HAW HAMBURG

BACHELORARBEIT

Analyse und Optimierung von Tracking- und Synchronisationsprozessen in Virtual-Production-Workflows

vorgelegt am 27. August 2026

Tim Kronshage
[redacted]

Erstprüfer: Prof. Dr. Marco Grimm
Zweitprüfer: Thorsten Wagener

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

Diese Bachelorarbeit untersucht Latenzen und instabile Trackingdaten in einem Virtual-Production-Aufbau des Videolabors der HAW Hamburg. Dafür wurden zwei echtzeitfähige Anwendungen entwickelt. Die erste Anwendung bestimmt den zeitlichen Versatz zwischen realem Kamerabild und virtuellem Hintergrund, indem Bewegungen in zwei SDI-Videosignalen erfasst und durch Kreuzkorrelation verglichen werden. Verzögerungen und ihre Richtung konnten grundsätzlich erkannt werden. Die Messergebnisse waren jedoch noch nicht ausreichend reproduzierbar. Die zweite Anwendung stabilisiert den FreeD-Trackingdatenstrom der VIVE Mars. Eine kalibrierte Stillstandserkennung unterscheidet zwischen Kamerabewegung und Stillstand. Wenn die Kamera still steht, wird die Kamerapose eingefroren, während Bewegungen mit einem One-Euro-Filter geglättet werden. Dadurch werden Drift und Mikrobewegungen zuverlässig unterdrückt. Die Trackingstabilisierung wurde weitgehend erfolgreich umgesetzt, während die Latenzbestimmung noch weiter validiert werden muss.

Abstract

This bachelor’s thesis investigates latency and unstable tracking data in a virtual production setup in the video laboratory at Hamburg University of Applied Sciences. Two real-time applications were developed for this purpose. The first application determines the temporal offset between the real camera image and the virtual background by detecting motion in two SDI video signals and comparing it using cross-correlation. Delays and their direction could generally be detected. However, the measurement results were not yet sufficiently reproducible. The second application stabilises the FreeD tracking data of the VIVE Mars. A calibrated standstill detection method distinguishes between camera movement and tracking noise. During standstill, the camera pose is frozen, while movements are smoothed using a One Euro filter. This reliably suppresses drift and small unintended movements. Tracking stabilisation was largely implemented successfully, while the latency measurement requires further validation.

Inhaltsverzeichnis

I Abbildungsverzeichnis I
II Tabellenverzeichnis II
III Abkürzungsverzeichnis III
1. Einleitung 1
1.1 Allgemeine Einführung in das Thema 1
1.2 Problemstellung 2
1.3 Zielsetzung der Arbeit 3
1.4 Bewertungskriterien der Softwarelösungen 3
1.5 Forschungsfrage und leitende Unterfragen 4
1.6 Vorgehen 5
1.7 Aufbau der Arbeit 5
2. Grundlagen und Aufbau der VP-Umgebung 6
2.1 Aufbau der Testumgebung 6
2.2 Das VIVE-Mars-Kamera-Tracking-System 6
2.2.1 Trackingprinzip der SteamVR-Technologie 6
2.2.2 Systemkomponenten 7
2.2.3 Datenfluss und Ausgabeprotokolle 8
2.2.4 Kalibrierung und Begleitsoftware 8
2.3 Blackmagic Ultimatte 12 HD 9
2.3.1 Einstellung des Delays in der Ultimatte 9
2.3.2 Relevante Anschlüsse und Schnittstellen 10
2.4 Unreal Engine 10
2.4.1 Echtzeit-Rendering in der UE 10
2.4.2 Datenfluss und Latenzquellen 10
2.5 Blackmagic DeckLink Duo 2 11
2.6 Signalfluss und Systemarchitektur 11
2.7 Verwendete Programmiersprachen und Softwarebibliotheken 12
3. Entwicklung eines Verfahrens zur Latenzbestimmung und Synchronisation 13
3.1 Synchronisationsverfahren in der VP 13
3.1.1 Problematik der Signalverzögerung 13
3.1.2 Das Flash-Verfahren 13
3.1.3 Bewegungsbasierte Latenzbestimmung als alternativer Ansatz 14
3.2 Problemdefinition und Zielsetzung 14
3.3 Messprinzip und Signalabgriffspunkte 15

3.3.1 Grundprinzip der Latenzbestimmung 15
3.3.2 Wahl der Signalabgriffspunkte 15
3.4 Erste Entwicklungsphase der Latenzberechnung 16
3.4.1 Ziel 16
3.4.2 Vorgehen 16
3.4.3 Wahl der Verfahren zur Versatzbestimmung 17
3.4.4 Wahl der Trackingdatensätze 19
3.4.5 Filterung und Normierung 20
3.4.6 Messungen 21
3.4.7 Betrachtung der Ergebnisse 22
3.4.8 Verrechnung zu einem Latenzwert 23
3.4.9 Zusammenfassung 24
3.5 Zweite Entwicklungsphase Latenzberechnung 25
3.5.1 Ziel 25
3.5.2 Funktionsweise der Live-Latenzberechnung 25
3.5.3 Umstellung auf eine threadbasierte Programmstruktur und Downscaling 30
3.5.4 Schnittstelle zur DeckLink Duo 2 32
3.5.5 Zusammenfassung 33
3.6 Dritte Entwicklungsphase 33
3.6.1 Ziel 33
3.6.2 Zusammenführung des Programms mit der Schnittstelle für die DeckLink Duo 2 33
3.6.3 Entwicklung eines Bewegungstriggers 34
3.6.4 Weitere Verbesserungen zur Delaybestimmung 36
3.6.5 Entwicklung der grafischen Benutzeroberfläche 36
3.6.6 Finaler Ablauf des Programms 38
3.6.7 Letzte Testphase und Fehleranalyse 39
3.6.8 Zusammenfassung 42
3.7 Abschließendes Ergebnis der Latenzberechnung 43
4 Entwicklung eines Verfahrens zur Stabilisierung von Trackingdaten 43
4.1 Trackingfehler innerhalb der VP 43
4.1.1 Problematik der Trackingfehler 43
4.1.2 Unterscheidung von Trackingfehlern 44
4.2 Vive-Mars-Kamera-Tracking-System 44
4.2.1 Trackingfehler des Vive-Mars-Kamera-Tracking-Systems 44

4.2.2 Filtersystem des Vive Mars 46
4.3 Ziel der Trackingdatenstabilisierung 46
4.4 Erste Entwicklungsphase der Trackingdatenstabilisierung 47
4.4.1 Ziel 47
4.4.2 Vorauswahl der Filter 47
4.4.3 Versuchsaufbau zur Auswahl der Filter 49
4.4.4 Ergebnisse des Filtertests 51
4.4.5 Analyse der Filter und Trackingdaten 53
4.4.6 Ergebnisse 56
4.5 Zweite Entwicklungsphase der Trackingdatenstabilisierung 56
4.5.1 Ziel und Vorgehen 56
4.5.2 FreeD 57
4.5.3 Stillstandserkennung 58
4.5.4 Erstes entwickeltes Programm zur Stillstandserkennung 60
4.5.5 Ergebnis 61
4.6 Dritte Entwicklungsphase der Trackingdatenstabilisierung 62
4.6.1 Ziel 62
4.6.2 Anpassungen des Programms an den Datenstrom der Vive Mars 62
4.6.3 Senden der Trackingdaten an UE 64
4.6.4 Einbauen eines Filters 65
4.6.5 Entwicklung der grafischen Benutzeroberfläche 66
4.6.6 Ergebnisse 67
4.7 Abschließendes Ergebnis der Trackingstabilisierung 68
5. Evaluation der entwickelten Lösungen 69
5.1 Evaluation der Latenzbestimmung 69
5.2 Evaluation der Trackingstabilisierung 70
5.3 Kompatibilität der Anwendungen mit einem Mehrkamera-Aufbau 72
6. Fazit 73
6.1 Zusammenfassung der Ergebnisse 73
6.2 Beantwortung der Forschungsfrage 73
6.3 Kritische Reflexion der Arbeit und mögliche Weiterentwicklungen 74
7. Literaturverzeichnis 76
8. Anhang 80
Anlage 1 80
Rechenzeiten der Korrelationsverfahren 80

Durchschnittliche Güte-Rohwerte nach Korrelationsverfahren....81
Konsistenz der Delay-Ergebnisse....82
Anlage 2....84
Ein- und Ausgabelemente der Bedienoberfläche zur Quellenauswahl....84
Eingabeelemente der Bedienoberfläche zur Latenzberechnung....86
Ausgabeelemente der Bedienoberfläche zur Latenzberechnung....88
Anlage 3....90
Eingabeelemente der Bedienoberfläche zur Trackingstabilisierung....90
Ausgabeelemente der Bedienoberfläche zur Trackingstabilisierung....92
9. Hilfsmittel....93
10. Eigenständigkeitserklärung....99

I Abbildungsverzeichnis

Abbildung 1: Vereinfachter Aufbau samt Signalfluss 12
Abbildung 2: Testvideo Vordergrund 16
Abbildung 3: Testvideo Hintergrund 16
Abbildung 4: Kamerageschwindigkeit Vorder- und Hintergrund ungefiltert und nicht normiert 21
Abbildung 5: Kamerageschwindigkeit Vorder- und Hintergrund gefiltert und normiert 21
Abbildung 6: Gefilterte und normierte Geschwindigkeitskurven nach Anwendung des berechneten Delays zur optischen Überprüfung 25
Abbildung 7: Frameraten der Innen- und Außenkamera bei 30 festen fps und einer festen Belichtung von -7 26
Abbildung 9: Delay- und Korrelationswerte über einen Berechnungszeitraum von 30 s des linearen Programms 30
Abbildung 8: X- und Y-Achse eines Berechnungsintervalls 30
Abbildung 10: Delay- und Korrelationswerte über einen Berechnungszeitraum von 30 s des threadbasierten Programms 32
Abbildung 11: Delayausgabe einer Übergangs-GUI ohne Bewegungstrigger 35
Abbildung 14: GUI 2 Delayberechnung, Livevorschau-Tab 37
Abbildung 13: GUI 2 Delayberechnung, Verlaufs-Tab 37
Abbildung 12: GUI 1 Quellenauswahl 37
Abbildung 15: GUI 2 Delayberechnung, Einstellungs-Tab 38
Abbildung 16: Erkennung von Frame-Sprüngen und Verwerfen betroffener Berechnungsfenster 40
Abbildung 17: Gültiges Messfenster mit einem Delay von -6,359 ms 41
Abbildung 18: Gültiges Messfenster mit einem Delay von -66,433 ms 41
Abbildung 19: Vergleich der Geschwindigkeitskurven des Testvideos mit kreisender Kamerabewegung und starker Filterung 54
Abbildung 20: Vergleich der Geschwindigkeitskurven eines Testvideos mit stiller Kamera mit starker Filterung 55
Abbildung 22: Konsolenausgabe des Stillstandserkennungs-Programms. Zustandswechsel von „Still“ zu „Moving“ 61
Abbildung 21: Konsolenausgabe des Stillstandserkennungs-Programms. Initialisierung 61
Abbildung 24: Konsolenausgabe des Stillstandserkennungsprogramms nach den Änderungen 64
Abbildung 23: Konsolenausgabe des Stillstandserkennungsprogramms vor den Änderungen 64
Abbildung 25: Vergleich ausgewählter One-Euro-Filterparameter anhand der Pan-Geschwindigkeit während einer schnellen Kamerabewegung. 66
Abbildung 26: GUI des Programms zur Stillstandserkennung 67

II Tabellenverzeichnis

Tabelle 1: Ergebnisse der Kreuzkorrelation für die gefilterten und normierten Geschwindigkeitsdaten der Kreisbewegung 24
Tabelle 2: Ergebnisse der abschließenden SDI-Messreihe 39
Tabelle 3: Kontrollmessung mit identischem, über einen SDI-Splitter verteiltem Signal 42
Tabelle 4: Parameter der im Filtertest untersuchten Filterstärken 50
Tabelle 5: Vergleich der Filterwirkung des Stillstands bei starker Filterung 53
Tabelle 6: Delay im Testszenario MPP69_Stillstand 53
Tabelle 7: Vergleich der One-Euro-Filterparameter anhand der Pan-Achse beim Rütteltest 66
Tabelle 8: Kalibrierte Rauschschwellen der drei Testkonfigurationen 68
Tabelle 9: Vergleich der Rechenzeiten der Korrelationsverfahren 80
Tabelle 10: Güte-Rohwerte – Kreuzkorrelation 81
Tabelle 11: Güte-Rohwerte – FFT-Kreuzkorrelation 81
Tabelle 12: Güte-Rohwerte – Peak Matching 82
Tabelle 13: Güte-Rohwerte – Least Squares 82
Tabelle 14: Konsistenz der Delay-Ergebnisse 84
Tabelle 15: Ein- und Ausgabeelemente der Bedienoberfläche (GUI 1 – SDI-Quellenauswahl) 85
Tabelle 16: Eingabeelemente der Bedienoberfläche (GUI 2 – Delay-Berechnung) 87
Tabelle 17: Ausgabeelemente der Bedienoberfläche (GUI 2 – Delay-Berechnung) 90
Tabelle 18: Eingabeelemente Trackingstabilisierung 91
Tabelle 19: Ausgabeelemente Trackingstabilisierung 92

III Abkürzungsverzeichnis

MPP…….……. Medienproduktions-Projekt

SDK…….…..….Software Development Kit

UE…….…….Unreal Engine

VP…….…..…Virtual Production

1. Einleitung

1.1 Allgemeine Einführung in das Thema

Die Film- und Fernsehindustrie befindet sich in einem fortschreitenden technologischen Wandel, der unter anderem durch die zunehmende Nutzung digitaler und echtzeitbasierter Technologien geprägt ist (vgl. Swords & Willment 2024, S. 1557). Einen bedeutenden Ansatz stellt dabei die Virtual Production (VP) dar. Sie verbindet digitale Inhalte mit physischen Produktionsprozessen und ermöglicht es, computergenerierte Umgebungen bereits während der Aufnahme in Echtzeit zu visualisieren und zu kontrollieren (vgl. Swords & Willment 2024, S. 1557f.). Dadurch werden Produktionsprozesse stärker miteinander verbunden und Entscheidungen können zunehmend über die klassischen Grenzen von Vorproduktion, Produktion und Postproduktion hinweg getroffen werden (vgl. Dean et al. 2026, S. 437). Ein wesentliches Merkmal von VP besteht darin, dass virtuelle Umgebungen bereits während der Aufnahme „in camera“ sichtbar gemacht und aufgezeichnet werden können, anstatt sie erst nachträglich in der Postproduktion hinzuzufügen (vgl. Swords & Willment 2024, S. 1558). Hierfür werden verschiedene Technologien miteinander kombiniert, darunter Echtzeit-Game-Engines, Kamera-Tracking, Motion Capture, dynamische Beleuchtung sowie Green- und Bluescreens und LED-Bildschirme (vgl. Swords & Willment 2024, S. 1558). Die Bedeutung von VP ergibt sich dabei nicht allein aus einzelnen Technologien, sondern insbesondere aus deren Zusammenspiel innerhalb eines gemeinsamen Produktionsprozesses (vgl. Dean et al. 2026, S. 437). VP findet mittlerweile in unterschiedlichen Bereichen der Medienproduktion Anwendung. Dazu zählen Film- und Serienproduktionen, Fernsehproduktionen sowie Werbe- und Musikvideoproduktionen (vgl. Swords & Willment 2024, S. 1558). Ein bekanntes Beispiel ist die Serie The Mandalorian, bei deren Produktion virtuelle Umgebungen mithilfe großflächiger LED-Wände in die physische Studioumgebung integriert wurden (vgl. Dean et al. 2026, S. 437). Auch im deutschen Fernsehen werden vergleichbare Verfahren eingesetzt. So werden die Nachrichtensendungen der ZDF-heute-Familie in einem virtuellen Studio produziert, in dem unter anderem ein Greenscreen zur Darstellung des virtuellen Studioumfelds eingesetzt wird (vgl. Gebhard 2021). Die Technologie bietet dabei verschiedene Vorteile für die Gestaltung und Organisation von Produktionsprozessen. Virtuelle Umgebungen können während der Produktion schnell verändert werden, ohne dass hierfür ein Wechsel des physischen Drehortes notwendig ist. Auch Beleuchtungssituationen lassen sich digital steuern und für spätere Aufnahmen wiederherstellen, wodurch sich praktische Vorteile ergeben (vgl. Swords & Willment 2024, S. 1562f.). Darüber hinaus können virtuelle Schauplätze schnell gewechselt und angepasst werden,

wodurch sich insbesondere logistische Aufwände reduzieren lassen (vgl. Swords & Willment 2024, S. 1563). Ein weiterer Vorteil wird im Bereich der Nachhaltigkeit gesehen. Durch den Einsatz virtueller Produktionsverfahren kann insbesondere der Reise- und Transportaufwand reduziert werden, da bestimmte Dreharbeiten in kontrollierten Studioumgebungen stattfinden können, anstatt an verschiedenen realen Drehorten durchgeführt zu werden (vgl. Swords & Willment 2024, S. 1565). Dadurch besteht das Potenzial, den mit Film- und Fernsehproduktionen verbundenen CO_2-Ausstoß zu verringern (vgl. Swords & Willment 2024, S. 1565). Insbesondere während der COVID-19-Pandemie gewann Virtual Production an Bedeutung. Reisebeschränkungen und erschwerte Bedingungen für Dreharbeiten trugen dazu bei, dass virtuelle Produktionsverfahren verstärkt entwickelt und eingesetzt wurden (vgl. Swords & Willment 2024, S. 1568). Gleichzeitig ist VP nicht als universell einsetzbare Lösung zu verstehen. Die Kombination verschiedener Technologien und Produktionsbereiche befindet sich weiterhin in einer dynamischen Entwicklungsphase und bringt neue technische und organisatorische Anforderungen mit sich (vgl. Dean et al. 2026, S. 436). Insbesondere die Einführung neuer Workflows erfordert eine frühzeitige Abstimmung zwischen den beteiligten Fachbereichen. Zudem verlagern sich bestimmte kreative und technische Entscheidungen in frühere Phasen des Produktionsprozesses, wodurch insbesondere die Previsualisierung an Bedeutung gewinnt (vgl. Swords & Willment 2024, S. 1567). Durch die Zusammenarbeit von Fachkräften aus unterschiedlichen Bereichen entstehen darüber hinaus neue Anforderungen an Kommunikation, Kompetenzen und die Abstimmung innerhalb des Produktionsprozesses (vgl. Dean et al. 2026, S. 436).

1.2 Problemstellung

Die zuvor beschriebenen technischen Anforderungen der VP zeigen sich auch bei deren Einsatz in der praktischen Lehre. An der HAW Hamburg wird im Rahmen des Moduls „Medienproduktions-Projekt“ (MPP) ein VP-Aufbau eingesetzt, bei dem anstelle einer LED-Wand ein Greenscreen verwendet wird. Der virtuelle Hintergrund wird in Echtzeit in der Unreal Engine (UE) gerendert und mithilfe des VIVE-Mars-Kamera-Tracking-Systems sowie der Blackmagic Ultimatte mit dem realen Kamerabild zusammengeführt. Dieser Aufbau ermöglicht es Studierenden, grundlegende VP-Workflows unter realen Produktionsbedingungen zu erlernen. Gleichzeitig bringt er spezifische technische Herausforderungen mit sich, die in den Produktionen der vergangenen Jahre wiederholt zu Problemen geführt haben.

Da die Berechnung des virtuellen Bildes in der UE eine gewisse Rechenzeit in Anspruch nimmt, entsteht eine zeitliche Abweichung zwischen dem realen Vordergrund

(Kamerabild) und dem gerenderten Hintergrund. Dieser Versatz muss an der Ultimatte manuell durch Augenmaß ausgeglichen werden, was in der Praxis zu einer ungenauen Synchronisation führt und den Qualitätsansprüchen einer professionellen Produktion nicht ausreichend genügt. Neben der unzureichend präzisen Synchronisation trat im MPP des Wintersemesters 2025/2026 ein anderes technisches Problem auf: Der virtuelle Hintergrund ruckelte wiederholt sichtbar, obwohl die reale Kamera nicht bewegt wurde. Ursache hierfür sind Instabilitäten im Trackingdatenstrom, die sich als Mikrobewegungen im gerenderten Bild äußern und die visuelle Qualität der virtuellen Umgebung erheblich beeinträchtigen. Die beschriebenen Herausforderungen bilden den Ausgangspunkt für die vorliegende Arbeit und führen zu der Frage, wie die Synchronisation zwischen realem und virtuellem Bild sowie die Stabilität der Trackingdaten verbessert werden können. Die daraus abgeleiteten Zielsetzungen werden im folgenden Kapitel dargestellt.

1.3 Zielsetzung der Arbeit

Ziel dieser Arbeit ist es, die beschriebenen Probleme im VP-Workflow des MPP an der HAW Hamburg systematisch zu untersuchen und durch geeignete Softwarelösungen zu reduzieren. Im Mittelpunkt stehen zwei eigenständige Anwendungen: ein Verfahren zur präzisen Bestimmung des zeitlichen Versatzes zwischen Vordergrund und virtuellem Hintergrund sowie ein Verfahren zur Stabilisierung des Trackingdatenstroms, um Mikrobewegungen und Drift im virtuellen Hintergrund zu verringern. Für beide Problemstellungen werden unterschiedliche Lösungsansätze konzipiert, prototypisch umgesetzt und in mehreren Entwicklungsphasen unter realitätsnahen Produktionsbedingungen erprobt. Anhand festgelegter Bewertungskriterien wird untersucht, welche Ansätze sich für den Einsatz im Livebetrieb am besten eignen. Abschließend sollen geeignete Parameter- und Grenzwerte bestimmt und die entwickelten Programme zu praxistauglichen Anwendungen mit grafischer Benutzeroberfläche weiterentwickelt werden. Um die entwickelten Lösungsansätze hinsichtlich ihrer Eignung für den praktischen Einsatz vergleichen zu können, werden im Folgenden fünf zentrale Bewertungskriterien definiert.

1.4 Bewertungskriterien der Softwarelösungen

Für eine einheitliche und nachvollziehbare Bewertung der entwickelten Softwarelösungen werden fünf Kriterien festgelegt. Diese orientieren sich an den Anforderungen eines Live-Produktionsbetriebs und dienen in Kapitel 5 als Grundlage für die abschließende Bewertung der entwickelten Anwendungen:

1. **Echtzeitfähigkeit:** Die Verarbeitung muss innerhalb der zeitlichen Anforderungen des Livebetriebs erfolgen, ohne den Produktionsablauf merklich zu verzögern.
2. **Stabilität:** Die Lösung soll unter unterschiedlichen Produktionsbedingungen zuverlässig arbeiten und möglichst konstante Ergebnisse liefern.
3. **Genauigkeit:** Die ermittelten Latenzwerte beziehungsweise die stabilisierten Trackingdaten müssen eine für den praktischen Einsatz ausreichende Genauigkeit aufweisen.
4. **Rechenaufwand:** Die Verarbeitung soll möglichst ressourcenschonend erfolgen, damit die Systemlast den laufenden Produktionsbetrieb nicht beeinträchtigt.
5. **Bedienbarkeit und Integrationsfähigkeit:** Die Anwendungen sollen intuitiv und ohne vertiefte technische Kenntnisse bedienbar sein. Zudem müssen sie sich mit möglichst geringem Aufwand in den bestehenden VP-Workflow integrieren lassen.

Die beschriebenen Problemstellungen und Zielsetzungen bilden den Ausgangspunkt für die folgende Forschungsfrage, die durch leitende Unterfragen weiter konkretisiert wird.

1.5 Forschungsfrage und leitende Unterfragen

Die übergeordnete Forschungsfrage dieser Arbeit lautet:

Wie können Latenz und Trackinginstabilitäten in virtuellen Produktionsumgebungen zuverlässig und echtzeitfähig kompensiert werden?

Zur strukturierten Bearbeitung dieser Frage werden folgende leitende Unterfragen herangezogen:

- Welche Ursachen führen zu den beschriebenen Synchronisations- und Stabilitätsproblemen?
- Welche Lösungsansätze sind grundsätzlich denkbar, und welche davon sind für den Livebetrieb unter Einsatz des VP-Setups der HAW geeignet?
- Welche Parameter und Grenzwerte ermöglichen eine zuverlässige Unterscheidung zwischen Kamerabewegung und Stillstand?
- Welche Kamerabewegungen eignen sich zur Kalibrierung und Synchronisation der Videoströme?

- Welche Softwarelösungen erweisen sich anhand der definierten Bewertungskriterien als die geeignetsten?

1.6 Vorgehen

Die Bearbeitung der Arbeit folgt einem iterativen Entwicklungsansatz, der die Softwareentwicklung eng mit der praktischen Erprobung verbindet. Zunächst werden der bestehende VP-Aufbau untersucht, die auftretenden Probleme analysiert und geeignete Testdaten aufgenommen. Anschließend erfolgt die Entwicklung der beiden Softwarelösungen jeweils in drei aufeinander aufbauenden Phasen. Die Erkenntnisse einer Entwicklungsphase fließen dabei unmittelbar in die jeweils nachfolgende Phase ein. Die Umsetzung erfolgt überwiegend in Python. Dabei werden unter anderem OpenCV zur Bild- und Videoverarbeitung, NumPy für numerische Berechnungen und Tkinter zur Entwicklung der grafischen Benutzeroberflächen eingesetzt. Für den Zugriff auf die SDI-Eingänge der DeckLink Duo 2 wird ergänzend eine in C++ entwickelte Schnittstelle verwendet. Abschließend werden die fertiggestellten Anwendungen anhand der festgelegten Kriterien evaluiert und als ausführbare Programme bereitgestellt.

1.7 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in sechs Kapitel. Nach der Einleitung werden in Kapitel 2 die theoretischen und technischen Grundlagen sowie der verwendete VP-Aufbau vorgestellt. Dabei werden insbesondere das VIVE-Mars-Kamera-Tracking-System, die UE, die Blackmagic Ultimatte 12 HD und die DeckLink Duo 2 genauer betrachtet. Darüber hinaus werden der Signalfluss der Testumgebung sowie die eingesetzten Entwicklungswerkzeuge und Softwarebibliotheken beschrieben. Kapitel 3 behandelt die Entwicklung des Verfahrens zur Latenzbestimmung. Ausgehend von der Analyse des Synchronisationsproblems werden zunächst das zugrunde liegende Messprinzip und die gewählten Signalabgriffspunkte erläutert. Anschließend wird die Entwicklung von der Auswertung aufgezeichneter Videos über die Verarbeitung von Live-Videoquellen bis zur Einbindung der SDI-Signale und der grafischen Benutzeroberfläche dargestellt. In Kapitel 4 wird das Verfahren zur Stabilisierung des Trackingdatenstroms entwickelt. Zunächst werden die auftretenden Trackingfehler untersucht und verschiedene Filterverfahren erprobt. Darauf aufbauend werden eine Stillstandserkennung sowie die Verarbeitung und Weiterleitung der FreeD-Trackingdaten umgesetzt. Abschließend wird für das Programm eine grafische Oberfläche entwickelt. Kapitel 5 umfasst die Evaluation der entwickelten Softwarelösungen anhand der zuvor festgelegten Bewertungskriterien. Darüber hinaus wird ihre Eignung für den Einsatz in einem Mehrkamera-Aufbau betrachtet. Kapitel 6 fasst die wesentlichen Ergebnisse der Arbeit zusammen und ordnet diese im Hinblick

auf die Forschungsfrage ein. Abschließend werden die Grenzen der entwickelten Lösungen kritisch reflektiert und mögliche Ansätze für zukünftige Weiterentwicklungen aufgezeigt.

2. Grundlagen und Aufbau der VP-Umgebung

2.1 Aufbau der Testumgebung

Für die Durchführung der Untersuchungen wird ein reduzierter Versuchsaufbau mit einer realen und einer virtuellen Kamera verwendet. Da der Aufbau im Rahmen des MPP der HAW Hamburg, für dessen Einsatz die entwickelten Lösungen vorgesehen sind, in der Regel zwei Kameras umfasst, wird abschließend untersucht, inwiefern sich das entwickelte Setup auf einen solchen Mehrkamera-Aufbau übertragen lässt. Der Versuchsaufbau besteht aus mehreren Komponenten: Als reale Bildquelle dient eine Sony NEX-FS700 mit Exmor-Super-35-CMOS-Sensor. Die Kamerabewegungen werden mithilfe des VIVE-Mars-Kamera-Tracking-Systems erfasst. Für die Berechnung des virtuellen Hintergrunds kommt ein leistungsfähiger Rechner mit UE zum Einsatz. Die Signalübertragung und -verarbeitung erfolgt über eine Blackmagic DeckLink Duo 2, während eine Blackmagic Ultimatte 12 HD die Zusammenführung des realen und virtuellen Bildmaterials übernimmt. Auf Grundlage des beschriebenen Versuchsaufbaus werden im Folgenden die für das Verständnis der eingesetzten VP-Umgebung relevanten technischen Grundlagen und Komponenten näher erläutert. Dabei werden sowohl die eingesetzten Hardwaresysteme als auch die für die virtuelle Bildgenerierung, Signalverarbeitung und Entwicklung der später vorgestellten Anwendungen relevanten Softwarekomponenten betrachtet.

2.2 Das VIVE-Mars-Kamera-Tracking-System

Das VIVE-Mars-Kamera-Tracking-System ermöglicht die präzise Erfassung der Kamerabewegung und ist speziell für den Einsatz in der virtuellen Produktion ausgelegt (vgl. HTC VIVE o. J.). Das System nutzt dafür SteamVR Tracking 2.0 in Verbindung mit VIVE-Trackern (vgl. HTC VIVE o. J.). Die Komponenten Mars und Rover verbinden das Tracking-System mit der Produktionsumgebung und ermöglichen die Übertragung der Trackingdaten an Anwendungen wie Unreal Engine (vgl. HTC VIVE o. J.).

2.2.1 Trackingprinzip der SteamVR-Technologie

Die SteamVR-Basisstationen bilden die räumliche Referenz des Trackingsystems. Sie senden Infrarot-Signale aus, die von den Sensoren des Trackers erfasst werden. Auf Grundlage dieser Messungen wird die räumliche Lage des Trackers bestimmt. Dabei

werden sechs Freiheitsgrade erfasst, die sich aus drei Positions- und drei Rotationsachsen zusammensetzen. Da der Tracker starr mit der Kamera verbunden ist, kann dessen Bewegung auf die virtuelle Kamera übertragen werden (vgl. Kuhlmann de Canaviri et al. 2023, S. 12ff.). Die Genauigkeit des Trackings wird unter anderem durch die Anzahl und Anordnung der Basisstationen sowie durch mögliche Verdeckungen beeinflusst. Kuhlmann de Canaviri et al. (2023) zeigen, dass der Einsatz von vier Basisstationen gegenüber einer Konfiguration mit zwei Basisstationen grundsätzlich eine höhere Trackinggenauigkeit ermöglicht. Dies gilt insbesondere für die dynamische Positionsbestimmung und die Rotationsgenauigkeit. Zudem zeigen die Untersuchungen, dass vier Basisstationen bei der teilweisen Verdeckung einzelner Basisstationen eine robuste Trackinggenauigkeit ermöglichen, wobei die Positionierung der Basisstationen weiterhin eine wichtige Rolle spielt (vgl. Kuhlmann de Canaviri et al. 2023, S. 12ff.). Für das VIVE-Mars-Kamera-Tracking-System werden mindestens zwei Basisstationen benötigt. Mit zwei Basisstationen ist laut Hersteller eine Trackingfläche von bis zu 5 m × 5 m möglich. Durch den Einsatz von vier Basisstationen kann die Trackingfläche auf bis zu 10 m × 10 m erweitert werden. Gleichzeitig kann die Verwendung zusätzlicher Basisstationen die Trackingabdeckung verbessern (vgl. HTC VIVE 2022, S. 44).

2.2.2 Systemkomponenten

Das VIVE-Mars-Kamera-Tracking-System besteht aus mehreren Hardwarekomponenten, die gemeinsam die Position und Orientierung der Kamera erfassen und die Trackingdaten an die virtuelle Produktionsumgebung weiterleiten (vgl. HTC VIVE 2022, S. 5).

- Die **Base Stations 2.0 (Lighthouse)** bilden die räumliche Referenz des Trackingsystems und ermöglichen die Positionsbestimmung der VIVE Tracker und Rover. Für den verwendeten Aufbau werden zwei Basisstationen so im Raum positioniert, dass eine möglichst durchgehende Erfassung des Trackers gewährleistet wird (vgl. HTC VIVE 2022, S. 8, 19).
- Der **VIVE Tracker 3.0** wird am Rover beziehungsweise an der Kamera befestigt und übermittelt dessen Positionsdaten an den Rover. Ein weiterer Tracker wird im Rahmen der Kamerakalibrierung am Kalibrierungsboard angebracht, um dessen Position innerhalb des Trackingbereichs zu bestimmen (vgl. HTC VIVE 2022, S. 9, 12, 26).
- Der **Rover** ist mit dem VIVE Tracker verbunden und leitet dessen Trackingdaten an die Mars-Einheit weiter. Darüber hinaus kann der Rover FIZ-Daten (Focus,

Iris, Zoom) von einem unterstützten Objektiv-Encoder erfassen und übertragen (vgl. HTC VIVE 2022, S. 9f.).

- Die **Mars-Einheit** sammelt die Trackingdaten der angeschlossenen Rover und übermittelt sie über einen IP-Router an den Produktionsrechner. Gleichzeitig dient sie als zentrale Schnittstelle zur Verwaltung der angeschlossenen Geräte und ermöglicht über das integrierte Mars-Dashboard die Kontrolle verschiedener System- und Netzwerkeinstellungen. Darüber hinaus verfügt Mars über Anschlüsse für Timecode und Genlock und unterstützt den Betrieb mehrerer Rover-Einheiten (vgl. HTC VIVE 2022, S. 10ff.).

2.2.3 Datenfluss und Ausgabeprotokolle

Die vom VIVE Tracker erfassten Positionsdaten werden über den Rover an die Mars-Einheit weitergeleitet (vgl. HTC VIVE 2020, S. 9f.). Mars übermittelt die Trackingdaten anschließend über einen IP-Router an den Produktionsrechner (vgl. HTC VIVE 2020, S. 10). Für die Weitergabe der Trackingdaten stehen insbesondere Live Link und FreeD zur Verfügung. Live Link ermöglicht es der Unreal Engine, Daten externer Quellen fortlaufend zu empfangen und innerhalb der Engine weiterzuverarbeiten (vgl. Epic Games o. J.). Das VIVE-Live-Link-Plugin ermöglicht die Einbindung von Mars als Datenquelle in Unreal Engine (vgl. HTC VIVE 2020, S. 20). Die mit Mars verbundenen Rover werden dabei im Live-Link-Fenster als einzelne Subjects angezeigt (vgl. HTC VIVE 2020, S. 36). FreeD ist ein standardisiertes Protokoll zur Übertragung von Kamera-Trackingdaten an virtuelle Produktionssysteme. Dabei können unter anderem Positions-, Rotations-, Fokus- und Zoomdaten übertragen werden (vgl. HTC VIVE 2020, S. 20). Für die in Kapitel 3 beschriebene Latenzbestimmung wird Live Link verwendet, während die in Kapitel 4 entwickelte Trackingstabilisierung einen FreeD-Datenstrom verarbeitet.

2.2.4 Kalibrierung und Begleitsoftware

Damit die reale und virtuelle Kamera aufeinander abgestimmt werden können, müssen zunächst die Kameraparameter und die räumliche Beziehung zwischen Kamera und Tracker bestimmt werden. Hierfür wird das VIVE Camera Calibration Tool eingesetzt, das anhand mehrerer Aufnahmen eines Schachbrettmusters die erforderlichen Kalibrierungsdaten ermittelt (vgl. HTC VIVE 2022, S. 26ff.). Dazu gehören unter anderem die Brennweite, die Objektivverzeichnung, der Bildmittelpunkt und der Nodal Offset (vgl. HTC VIVE 2022, S. 32). Die erzeugten Kalibrierungsdaten werden anschließend in Unreal Engine übernommen. Hierzu wird ein Lens File angelegt und mit den aus der Kalibrierung ermittelten Werten für Brennweite, Verzeichnung, Bildmittelpunkt und Nodal Offset versehen (vgl. HTC VIVE 2022, S. 29ff.). Das Lens File wird anschließend der

CineCameraActor zugewiesen, wodurch die Kalibrierungsdaten auf die virtuelle Kamera angewendet werden (vgl. HTC VIVE 2022, S. 33).

2.3 Blackmagic Ultimatte 12 HD

Die Ultimatte 12 HD ist ein professioneller Echtzeit-Compositor zur Integration von Vorder- und Hintergrundbildern (vgl. Blackmagic Design 2026, S. 315f.). Im vorliegenden Versuchsaufbau wird das reale Kamerabild vor dem Greenscreen als Vordergrundsignal und das in der Unreal Engine gerenderte Bild als Hintergrundsignal zugeführt (vgl. Blackmagic Design 2026, S. 320). Für die Trennung von Vorder- und Hintergrund erzeugt der Ultimatte eine sogenannte Matte. Diese legt fest, welche Bereiche des Quellbildes sichtbar, entfernt oder teilweise transparent dargestellt werden (vgl. Blackmagic Design 2026, S. 315). Zur weiteren Optimierung des Compositings stehen unter anderem Funktionen zur Bearbeitung der Mattenkanten sowie zur Unterdrückung von Spill-Effekten zur Verfügung (vgl. Blackmagic Design 2026, S. 350). Darüber hinaus können Helligkeit, Kontrast und Sättigung der einzelnen Bildquellen unabhängig voneinander angepasst werden (vgl. Blackmagic Design 2026, S. 359). Für die nachfolgenden Versuche ist insbesondere die Möglichkeit relevant, das Vordergrund-Eingangssignal innerhalb der Ultimatte zeitlich zu verzögern. Über die Einstellung **FRAME DELAY FG INPUT** kann das Signal um eine definierte Anzahl von Frames verzögert werden. Der Einstellbereich reicht von 0 bis 14 Frames (vgl. Blackmagic Design 2026, S. 368).

2.3.1 Einstellung des Delays in der Ultimatte

Das virtuelle Hintergrundbild wird im verwendeten Versuchsaufbau über einen anderen Signalweg als das direkte Kamerabild zur Ultimatte geführt. Dadurch kann zwischen beiden Signalen ein zeitlicher Versatz entstehen. Zur zeitlichen Angleichung wird das Vordergrundsignal über den Parameter *Frame Delay FG Input* verzögert. Die Eingabeeinstellungen der Ultimatte ermöglichen eine Verzögerung des Vordergrundsignals, um dieses mit dem Hintergrundsignal zu synchronisieren (vgl. Blackmagic Design 2026, S. 367 f.). Der für die verwendete Ultimatte 12 HD verfügbare Einstellbereich reicht von 0 bis 14 Frames (vgl. Blackmagic Design 2026, S. 368). Bei einer Framerate von 50 fps entspricht ein Frame einer Dauer von 20 ms. Somit kann das Vordergrundsignal um bis zu 280 ms verzögert werden. Überschreitet der ermittelte zeitliche Versatz zwischen den beiden Signalpfaden diesen Wert, kann eine zusätzliche externe Verzögerung des Kamerasignals erforderlich sein. Wie der für den jeweiligen Versuchsaufbau benötigte Verzögerungswert bestimmt werden kann, wird in Kapitel 3 untersucht.

2.3.2 Relevante Anschlüsse und Schnittstellen

Für den Versuchsaufbau werden der FG-In-Eingang für das reale Kamerabild, der BG-In-Eingang für den virtuellen Hintergrund sowie Program Out für das zusammengesetzte Ausgangsbild verwendet. Ein zusätzlicher Monitor-Out-Ausgang dient der Signalkontrolle. Über den REF-Anschluss kann die Ultimatte an eine externe Referenzquelle angebunden werden. Die Steuerung des Geräts erfolgt über Ethernet. Die Ultimatte 12 HD unterstützt 3G-SDI-Signale bis 1080p60 (Blackmagic Design, 2026, S. 324–327).

2.4 Unreal Engine

2.4.1 Echtzeit-Rendering in der UE

Die UE ist eine von Epic Games entwickelte Echtzeit-Game-Engine, die ursprünglich für die Entwicklung von Videospielen konzipiert wurde und inzwischen auch in anderen Bereichen eingesetzt wird. Dazu zählt unter anderem die Filmindustrie, in der Echtzeit-Rendering und Game Engines zunehmend für die Umsetzung von Virtual Production eingesetzt werden (vgl. Jia et al. 2026, S. 461f.). Für VP ist dabei insbesondere relevant, dass virtuelle Inhalte bereits während der Produktion in Echtzeit dargestellt und unmittelbar angepasst werden können, wodurch sich der traditionelle lineare Produktionsablauf verändert (vgl. Jia et al. 2026, S. 462f.). Für die Darstellung komplexer virtueller Szenen stehen in der UE 5 unter anderem die Rendering-Technologien Nanite und Lumen zur Verfügung. Nanite ermöglicht die Verarbeitung hochdetaillierter 3D-Geometrien, indem die Darstellung von Objekten an deren Entfernung zur Kamera und die jeweilige Perspektive angepasst wird. Dadurch können auch detailreiche Assets verarbeitet werden, ohne dass für unterschiedliche Detailstufen mehrere Versionen eines Objekts manuell erstellt werden müssen (vgl. Sobchyshak et al. 2026, S. 3). Lumen dient der dynamischen Berechnung der Beleuchtung innerhalb der virtuellen Szene. Im Gegensatz zu Verfahren, bei denen Beleuchtungsinformationen bereits im Vorfeld berechnet werden, kann Lumen Veränderungen innerhalb der Szene während der Laufzeit berücksichtigen (vgl. Sobchyshak et al. 2026, S. 3). Dadurch können sich beispielsweise Licht- und Schattenverhältnisse bei Veränderungen der Lichtquellen, bewegten Objekten oder einer veränderten Tageszeit unmittelbar anpassen, ohne dass die Beleuchtung zuvor vollständig vorberechnet werden muss (vgl. Sobchyshak et al. 2026, S. 3).

2.4.2 Datenfluss und Latenzquellen

Die Positions- und Rotationsdaten des VIVE-Mars-Systems werden über das VIVE-Live-Link-Plugin in die UE übertragen. Dort steuern sie einen CineCameraActor, der die erfasste Bewegung der realen Kamera im virtuellen Raum nachbildet. Die virtuelle Kamera folgt somit den gemessenen Trackingdaten. Durch Mess-, Übertragungs- und

Verarbeitungszeiten kann es jedoch zu einer zeitlichen Abweichung zwischen der Bewegung der realen und der virtuellen Kamera kommen. Für den hier untersuchten Aufbau lässt sich die Gesamtlatenz auf mehrere aufeinanderfolgende Verarbeitungsschritte zurückführen. Dazu zählen die Erfassung und Übertragung der Trackingdaten, deren Verarbeitung in der UE, die Aktualisierung der virtuellen Szene, das Rendering auf der GPU, mögliche Frame-Puffer und schließlich die Videoausgabe über die DeckLink Duo 2. Auch wenn jedes Einzelbild innerhalb des für die Produktionsframerate verfügbaren Zeitfensters berechnet wird, kann die gesamte Verarbeitungskette einen zusätzlichen zeitlichen Versatz verursachen. Das resultierende Bild wird als SDI-Signal an den BG-In-Eingang der Ultimatte ausgegeben.

2.5 Blackmagic DeckLink Duo 2

Die Blackmagic DeckLink Duo 2 ist eine PCIe-Karte zur Aufzeichnung und Ausgabe von Videosignalen. Sie verfügt über vier bidirektionale 3G-SDI-Anschlüsse und unterstützt SD- und HD-Formate bis 1080p60 (vgl. Blackmagic Design o. J.-a). Abhängig von der gewählten Duplex-Konfiguration können die Anschlüsse entweder als vier unabhängige Ein- oder Ausgabekanäle oder als zwei Eingangs- und zwei Ausgangskanäle betrieben werden (vgl. Blackmagic Design o. J.-a). Zusätzlich verfügt die Karte über einen Referenzeingang für Black-Burst- oder Tri-Sync-Signale (vgl. Blackmagic Design o. J.-a). Im Produktionsaufbau übernimmt die DeckLink Duo 2 die SDI-Ausgabe des in der UE gerenderten Hintergrundbilds. Für die Latenzbestimmung werden außerdem zwei SDI-Signale über die Karte eingelesen und der entwickelten Anwendung bereitgestellt. Damit bildet die DeckLink Duo 2 die zentrale Schnittstelle zwischen der softwarebasierten Bildverarbeitung und der SDI-Signalkette.

2.6 Signalfluss und Systemarchitektur

Der Versuchsaufbau besteht aus zwei parallel verlaufenden Signalpfaden, die an der Ultimatte 12 HD zusammengeführt werden (vgl. Abbildung 1). Im Vordergrundpfad wird das reale Kamerabild der Sony NEX-FS700 über SDI direkt an den FG-In-Eingang der Ultimatte übertragen. Im Hintergrundpfad erfasst das VIVE-Mars-System die Position und Orientierung der Kamera und überträgt die Trackingdaten über Live Link an die UE. Dort steuern die Trackingdaten die virtuelle Kamera, aus deren Perspektive der Hintergrund in Echtzeit gerendert wird. Das erzeugte Bild wird anschließend über die DeckLink Duo 2 als SDI-Signal ausgegeben und dem BG-In-Eingang der Ultimatte zugeführt. Aufgrund der zusätzlichen Tracking-, Rendering- und Ausgabeschritte weist der Hintergrundpfad eine höhere Latenz als der Vordergrundpfad auf. Zur zeitlichen Angleichung der beiden Signalpfade wird das früher eintreffende Vordergrundsignal in der Ultimatte

über den Parameter *Frame Delay FG Input* verzögert. Der beschriebene Signalfluss ist in Abbildung 1 schematisch dargestellt und verdeutlicht den Aufbau der beiden Signalpfade.

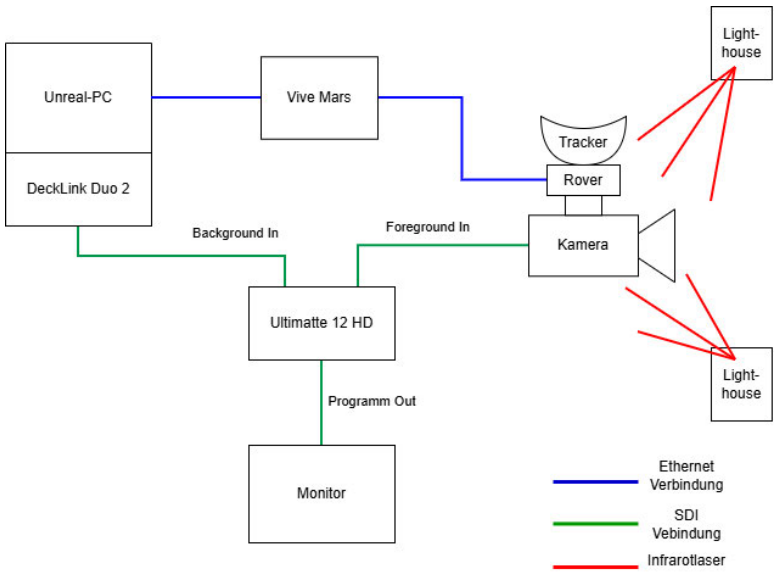


Abbildung 1: Vereinfachter Aufbau samt Signalfluss

2.7 Verwendete Programmiersprachen und Softwarebibliotheken

Die beiden Anwendungen werden überwiegend in Python entwickelt. Die Programmiersprache hat sich insbesondere im wissenschaftlichen Umfeld für explorative und interaktive Anwendungen sowie für die prototypische Entwicklung etabliert. Ergänzt wird Python dabei durch ein umfangreiches Ökosystem spezialisierter Bibliotheken, das unter anderem die Verarbeitung wissenschaftlicher und numerischer Daten unterstützt (vgl. Millman & Aivazis, 2011, S. 11f.). Die tatsächliche Eignung der entwickelten Verfahren für die zeitkritische Verarbeitung wird im Rahmen der Arbeit anhand ihrer Laufzeiten überprüft.

Die Open-Source-Bibliothek **OpenCV** wird zum Einlesen und Verarbeiten der Videobilder sowie zur bewegungsbasierten Auswertung der Kamerasignale eingesetzt. Neben grundlegenden Funktionen zur Verarbeitung von Videosequenzen stellt OpenCV auch Verfahren zur Merkmals- und Objektverfolgung bereit. Die dabei ermittelten Bewegungsverläufe werden anschließend für die Latenzberechnung verwendet (vgl. Culjak et al., 2012, S. 1728f.).

NumPy stellt die für die Anwendung benötigten Datenstrukturen und Funktionen zur numerischen Verarbeitung bereit. Im Mittelpunkt steht dabei das mehrdimensionale Array als zentrale Datenstruktur für die Verarbeitung numerischer Daten. Darüber hinaus

ermöglicht NumPy unter anderem vektorisierte Berechnungen sowie verschiedene Funktionen zur Bearbeitung und Analyse von Arrays (vgl. Harris et al., 2020, S. 357f.). In den entwickelten Anwendungen wird NumPy unter anderem für die Verarbeitung und Auswertung der Bewegungs- und Trackingdaten eingesetzt.

Für die Erstellung der Bedienoberflächen wird **Tkinter** eingesetzt. Das Modul stellt eine Schnittstelle zwischen Python und dem GUI-Toolkit Tcl/Tk bereit und ermöglicht damit die Entwicklung grafischer Benutzeroberflächen. In den entwickelten Anwendungen dient Tkinter unter anderem zur Auswahl von Quellen, zur Konfiguration von Parametern und zur Darstellung der berechneten Ergebnisse (vgl. Python Software Foundation, o. J.).

C++ und DeckLink Software Development Kit (SDK): In der entwickelten Anwendung zur Latenzbestimmung wird ergänzend eine C++-Schnittstelle auf Basis des DeckLink SDK eingesetzt. Diese übernimmt den hardwarenahen Zugriff auf die SDI-Eingänge der DeckLink Duo 2 und liest die eingehenden Videoframes ein. Die Bildanalyse erfolgt anschließend in Python, während die Anbindung an die Hardware in C++ umgesetzt wird. Die Kommunikation zwischen beiden Anwendungsteilen erfolgt über einen gemeinsamen Speicherbereich. Dadurch können die zeitkritische Hardwareanbindung und die eigentliche Auswertung der Bilddaten getrennt voneinander umgesetzt werden.

3. Entwicklung eines Verfahrens zur Latenzbestimmung und Synchronisation

3.1 Synchronisationsverfahren in der VP

3.1.1 Problematik der Signalverzögerung

Die zeitliche Angleichung von realem Vordergrund und virtuellem Hintergrund gehört zu den grundlegenden und in nahezu jeder VP-Umgebung wiederkehrenden Problemen. Da die Berechnung des virtuellen Bildes, von der Übertragung der Trackingdaten über das Rendering bis zur Signalausgabe, eine gewisse Zeit in Anspruch nimmt, trifft das Hintergrundsignal in aller Regel später ein als das direkte Kamerabild. Ohne geeignete Kompensation führt dieser Versatz zu einer sichtbaren zeitlichen Diskrepanz zwischen Vorder- und Hintergrund, insbesondere bei schnellen Kamerabewegungen.

3.1.2 Das Flash-Verfahren

Ein visuelles Kalibrierverfahren zur Bestimmung der Latenz wird beispielsweise im xR-Workflow von Disguise eingesetzt. Dabei gibt die LED-Wand zwischen mehreren

schwarzen Einzelbildern genau ein weißes Einzelbild aus. Die Kamera nimmt diese Bildfolge auf, und das im Kamerasignal erkannte weiße Einzelbild dient zur Bestimmung des sogenannten *Video Receive Delay*. Dieser Wert beschreibt die Zeit zwischen der Berechnung beziehungsweise Ausgabe eines Bildes durch das Render-System und dem Empfang des entsprechenden Kamerabildes. Der ermittelte Versatz wird anschließend zur zeitlichen Abstimmung der LED-Wand und der virtuellen Set-Erweiterung verwendet (vgl. Disguise 2026). Das Verfahren basiert somit auf einem kurzzeitigen und eindeutig identifizierbaren Helligkeitsereignis, das den vollständigen Signalweg über die LED-Wand und die Kamera durchläuft. Voraussetzung ist daher, dass das ausgegebene Referenzbild von der Kamera aufgenommen wird. Im Greenscreen-Aufbau des MPP wird der virtuelle Hintergrund jedoch nicht auf einer LED-Wand dargestellt, sondern separat erzeugt und erst in der Ultimatte mit dem Kamerabild kombiniert. Das Referenzereignis des Hintergrundsignals durchläuft somit nicht die Kamera. Das beschriebene Flash-Verfahren kann deshalb nicht unmittelbar zur Bestimmung des Versatzes zwischen Vorder- und Hintergrund eingesetzt werden.

3.1.3 Bewegungsbasierte Latenzbestimmung als alternativer Ansatz

Für die vorliegende Arbeit ergibt sich daraus die Notwendigkeit eines alternativen Ansatzes, der auch bei getrennt geführten Signalen und ohne gemeinsames Kalibrierereignis funktioniert. Die zugrunde liegende Idee ist, den Delay nicht anhand eines einmaligen visuellen Ereignisses, sondern anhand der Bewegungsverläufe von realer und virtueller Kamera zu bestimmen: Da beide auf denselben Trackingdaten basieren, sind ihre Bewegungen im Idealfall identisch und unterscheiden sich lediglich um den gesuchten zeitlichen Versatz. Durch eine Korrelation der jeweils aus dem Videosignal extrahierten Bewegungsverläufe lässt sich dieser Versatz messwertbasiert bestimmen, ohne dass ein separates Kalibrierereignis oder ein gemeinsames Kamerabild erforderlich wäre. Im Gegensatz zum Flash- bzw. Klappen-Verfahren ist dieser Ansatz zudem nicht auf eine einmalige Messung vor Drehbeginn beschränkt, sondern lässt sich fortlaufend während des laufenden Betriebs anwenden, wodurch auch zeitlich veränderliche Latenzanteile erfasst werden können.

3.2 Problemdefinition und Zielsetzung

Da die Delay-Einstellung zwischen Vorder- und Hintergrundvideosignal an der Ultimatte derzeit ausschließlich manuell und visuell (per Augenmaß über die Timing-Kontrolle im Inputs-Menü) erfolgt, kann die zeitliche Abstimmung des ausgegebenen Videosignals ungenau sein. Das im Rahmen dieses Abschnitts zu entwickelnde Programm soll diese manuelle Einstellung durch eine automatisierte, messwertbasierte Delay-Berechnung

ersetzen. Dabei soll es möglichst recheneffizient, framegenau und robust gegenüber Störeinflüssen arbeiten. Ein zentrales Teilproblem ist hier die Wahl geeigneter Signalabgriffspunkte: Entscheidend ist nicht ein bestimmter Abgriffspunkt per se, sondern dass der softwareseitig berechnete Delay-Wert exakt dem an der Ultimatte tatsächlich einzustellenden Korrekturwert entspricht – wählbar sind somit grundsätzlich mehrere Abgriffspunkte (z. B. an den Eingängen oder auch am Ausgangssignal der Ultimatte), solange sich daraus zuverlässig der richtige Einstellwert ableiten lässt.

3.3 Messprinzip und Signalabgriffspunkte

3.3.1 Grundprinzip der Latenzbestimmung

Da sich die reale und die virtuelle Kamera aufgrund des gemeinsamen Trackingsystems synchron bewegen, liegt es nahe, die Synchronisation der beiden Videosignale anhand der jeweiligen Bewegungsverläufe vorzunehmen. Hierzu bietet sich ein bildbasiertes Tracking der Kamerabewegung entlang der x- und y-Achse an, das mithilfe der Bibliothek OpenCV realisiert werden kann. Da die Bewegungsinformationen des realen und des virtuellen Bildes auf demselben Trackingdatenstrom beruhen, lassen sich beide Signale über eine Korrelation ihrer Bewegungsverläufe – etwa anhand von Positions-, Geschwindigkeits- oder Beschleunigungsdaten – zeitlich zueinander in Bezug setzen. Aus der zeitlichen Verschiebung des Korrelationsmaximums lässt sich der bestehende Delay zwischen beiden Signalen ableiten.

3.3.2 Wahl der Signalabgriffspunkte

Ein weiteres zentrales Teilproblem ist die Wahl geeigneter Signalabgriffspunkte, da der softwareseitig berechnete Delay-Wert exakt dem an der Ultimatte tatsächlich einzustellenden Korrekturwert entsprechen muss. Grundsätzlich lassen sich zwei mögliche Abgriffskonzepte unterscheiden: zum einen das synchrone, voneinander getrennte Abgreifen von Vorder- und Hintergrundsignal, zum anderen das Abgreifen des bereits von der Ultimatte zusammengesetzten Ausgangsbildes mit anschließendem separatem Tracking beider Bildanteile innerhalb desselben Bildes.

Gegen die zweite Variante spricht jedoch, dass sich Vorder- und Hintergrund im zusammengesetzten Ausgangsbild – gerade im Livebetrieb – nur schwer zuverlässig voneinander trennen lassen, da beide Bildanteile im Signal bereits ineinander verschmolzen sind und keine eindeutige Trennlinie mehr vorliegt. Aufgrund dieser technischen Unsicherheit wird für die weitere Entwicklung der erste Ansatz gewählt: das synchrone, getrennte Abgreifen von Vorder- und Hintergrundsignal über SDI-Splitter vor deren Eingängen.

3.4 Erste Entwicklungsphase der Latenzberechnung

3.4.1 Ziel

In der ersten Entwicklungsphase steht die Konzeption und prototypische Umsetzung eines ersten Programms zur Bestimmung des Delays zwischen Vorder- und Hintergrundsignal im Mittelpunkt. Grundlage hierfür bilden aufgenommene Testvideos von Vorder- und Hintergrund (siehe Abbildung 2 und 3) mit unterschiedlichen Bewegungsmustern, darunter horizontale, vertikale und diagonale Kamerabewegungen, kreisende Bewegungen sowie Rütteln, anhand derer sich die Verfahren in einer kontrollierten Umgebung testen lassen, bevor sie im weiteren Verlauf der Arbeit auf den Livebetrieb übertragen werden.

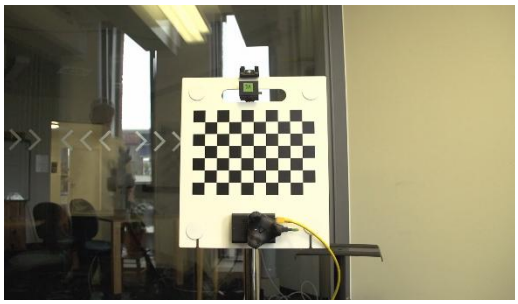


Abbildung 2: Testvideo Vordergrund



Abbildung 3: Testvideo Hintergrund

Zentraler Bestandteil dieser Phase ist die Entwicklung eines Videotrackings, mit dem sich die Bewegung der Kamera in beiden Signalen frameweise erfassen lässt. Aus der getrackten Bewegung wird direkt die Geschwindigkeit abgeleitet, aus der anschließend durch numerische Differentiation die Beschleunigung berechnet wird. Auf dieser Datengrundlage sollen zwei zentrale Fragestellungen beantwortet werden: zum einen, welche der beiden Datenarten – Positionsänderung, Geschwindigkeit oder Beschleunigung – sich besser für die Korrelation der beiden Signale eignet. Zum anderen, welches der vier untersuchten Korrelationsverfahren – Kreuzkorrelation, Peak Matching, die Methode der kleinsten Quadrate sowie die FFT-Kreuzkorrelation – die zuverlässigsten Ergebnisse liefert. Ergänzend werden die berechneten Ergebnisse in geeigneter Form grafisch aufbereitet, um die Bewertung der einzelnen Verfahren nachvollziehbar zu gestalten. Sofern sich die entwickelten Verfahren in dieser ersten Phase als grundsätzlich funktionsfähig erweisen, wird das Programm in der zweiten Entwicklungsphase für den Einsatz mit Live-Videosignalen umgebaut.

3.4.2 Vorgehen

Zunächst wird eine grundlegende Programmstruktur entworfen, die die einzelnen Verarbeitungsschritte in voneinander getrennte, wiederverwendbare Funktionseinheiten gliedert. Darauf aufbauend werden folgende Funktionen umgesetzt.

Das Programm liest zunächst zwei Videos ein, die das Vordergrund- und das Hintergrundsignal enthalten. Dabei werden die Einzelbilder sowie die jeweilige Bildrate erfasst. Letztere dient später dazu, einen in Frames bestimmten Versatz in einen Zeitwert umzurechnen. Zur Erfassung der Kamerabewegung wird eine OpenCV-Implementierung des Lucas-Kanade-Verfahrens verwendet. Das zugrunde liegende Verfahren leitet die Verschiebung zwischen Bildinhalten aus den räumlichen Helligkeitsverläufen ab und kann für die Bestimmung mehrdimensionaler Verschiebungen erweitert werden (vgl. Lucas & Kanade 1981, S. 676f.). In der hier entwickelten Anwendung werden die ermittelten Bewegungsvektoren anschließend durch eine robuste affine Transformationsschätzung zusammengeführt. Aus deren Translationsanteilen wird für jedes aufeinanderfolgende Bildpaar die Verschiebung entlang der x- und y-Achse bestimmt. Durch Division dieser Verschiebung durch das Zeitintervall zwischen den Einzelbildern ergeben sich die Geschwindigkeitskomponenten $v_x$und $v_y$sowie deren Betrag. Die Beschleunigung wird anschließend durch numerische Ableitung der Geschwindigkeit nach der Zeit berechnet. Um die Vergleichbarkeit unterschiedlich skalierter Signale herzustellen, werden die Daten im Anschluss mittels eines Tiefpassfilters geglättet und anschließend normiert. Auf Basis der so vorverarbeiteten Daten werden die vier Korrelationsverfahren (Kreuzkorrelation, Methode der kleinsten Quadrate, Peak Matching und FFT-Kreuzkorrelation) implementiert und zur Bestimmung des zeitlichen Versatzes zwischen den beiden Signalen eingesetzt. Abschließend werden die Ergebnisse grafisch dargestellt: einerseits die x- und y-Komponente sowie der Betrag der jeweiligen Größe für beide Videos im direkten Vergleich, andererseits dieselben Größen unter Berücksichtigung der jeweils berechneten Verschiebung, um die erzielte Ausrichtung der beiden Signale visuell überprüfen zu können. Für die durchgängige Verarbeitung der Daten wird zudem eine einheitliche Datentupel-Struktur (Zeit, Framerate, x- und y-Komponente, Betrag) festgelegt, mit der die Werte konsistent zwischen den einzelnen Verarbeitungsschritten weitergereicht werden.

3.4.3 Wahl der Verfahren zur Versatzbestimmung

Zur Bestimmung des zeitlichen Versatzes zwischen den Bewegungsverläufen von Vorder- und Hintergrundsignal werden im Rahmen dieser Arbeit folgende Verfahren zur Versatzbestimmung untersucht: die klassische Kreuzkorrelation, das Peak Matching, die Methode der kleinsten Quadrate sowie die FFT-basierte Kreuzkorrelation. Bei der direkten und der FFT-basierten Kreuzkorrelation handelt es sich nicht um zwei unterschiedliche Korrelationsverfahren, sondern um verschiedene Berechnungswege derselben Korrelationsfunktion. Lyon beschreibt die Berechnung im Frequenzbereich mithilfe des Korrelationstheorems und zeigt, dass sie dasselbe Ergebnis wie die direkte Berechnung

liefert (vgl. Lyon 2010, S. 20). Coakley und Hale zeigen außerdem, dass die Minimierung der quadrierten Differenzen zwischen einem Referenzsignal und einem zeitlich verschobenen Signal in der von ihnen untersuchten Formulierung zum gleichen Ergebnis wie die Maximierung der Kreuzkorrelation führt (vgl. Coakley & Hale 2001, S. 145). Die Ansätze werden in dieser Arbeit dennoch getrennt umgesetzt, um ihre praktische Berechnung und die jeweils benötigte Rechenzeit zu vergleichen.

Die **direkte Kreuzkorrelation** dient als Ausgangspunkt des Vergleichs. Hierzu wird eines der beiden Signale gegenüber dem anderen verschoben und für die untersuchten Verschiebungen die jeweilige Übereinstimmung bestimmt. Als geschätzter Versatz wird die Verschiebung mit dem höchsten Korrelationswert verwendet. Ein entsprechendes Vorgehen zur Ermittlung der relativen Verschiebung zweier eindimensionaler Signale beschreiben Coakley und Hale (vgl. Coakley & Hale 2001, S. 145).

Beim ereignisbasierten **Peak-Matching** werden markante lokale Extremstellen in beiden Bewegungsverläufen erkannt und anhand ihrer Zeitpositionen einander zugeordnet. In der hier entwickelten Umsetzung wird der Versatz aus den zeitlichen Abständen der zugeordneten Peaks bestimmt. Eine verwandte Herangehensweise findet sich bei Quian Quiroga, Kreuz und Grassberger. Ihr Verfahren reduziert die untersuchten Zeitreihen auf die Zeitpunkte ausgewählter Ereignisse, die im Anwendungsbeispiel durch lokale Maxima gebildet werden. Anhand der zeitlichen Reihenfolge dieser Ereignisse wird beurteilt, welches Signal dem anderen vorausgeht (vgl. Quian Quiroga et al. 2002, S. 1). Die konkrete Peak-Erkennung und Berechnung eines einzelnen Versatzwertes wird für die vorliegende Anwendung eigenständig umgesetzt.

Bei der **Minimierung der Summe der quadrierten Abweichungen** wird für jede untersuchte Verschiebung die Differenz zwischen den einander zugeordneten Signalwerten berechnet. Anschließend werden die Differenzen quadriert und aufsummiert. Die Verschiebung mit der kleinsten Fehlersumme wird als zeitlicher Versatz verwendet. Coakley und Hale wenden dieses Kriterium auf die Ausrichtung zeitlich verschobener Signale an. In der dort beschriebenen Berechnung stimmt das Minimum der Fehlerquadratsumme mit dem Maximum der Kreuzkorrelation überein (vgl. Coakley & Hale 2001, S. 145).

Bei der **FFT-basierten Kreuzkorrelation** werden beide Signale zunächst in den Frequenzbereich überführt. Dort wird die Fourier-Transformierte eines Signals mit der komplex konjugierten Fourier-Transformierten des anderen Signals multipliziert. Die inverse Fourier-Transformation dieses Produkts liefert anschließend die Korrelationsfunktion. Lyon beschreibt diesen Berechnungsweg und weist nach, dass die direkte und die FFT-basierte Berechnung dasselbe Korrelationsergebnis erzeugen (vgl. Lyon 2010, S. 20).

In der vorliegenden Anwendung wird der zeitliche Versatz aus der Position des Maximums dieser Korrelationsfunktion bestimmt.

Um die vier Verfahren unter vergleichbaren Bedingungen zu bewerten, werden sie im weiteren Verlauf jeweils auf drei unterschiedliche, aus den Trackingdaten abgeleitete Datensätze angewendet: die Positionsänderung (Pixel pro Frame), die Geschwindigkeit (Pixel pro Sekunde) sowie die Beschleunigung (Pixel pro Sekunde²). Dadurch lässt sich beurteilen, welche dieser Datenreihen sich am besten zur Korrelation der Vorder- und Hintergrundsignale eignet und ob sich die Eignung je nach eingesetztem Korrelationsverfahren unterscheidet. Die Bewertung der Verfahren erfolgt anhand der in Abschnitt 1.4 definierten Kriterien Genauigkeit, Geschwindigkeit und Robustheit. Im Rahmen der Genauigkeitsbewertung wird insbesondere geprüft, ob die von den einzelnen Verfahren jeweils berechneten Delay-Werte untereinander übereinstimmen. Die Geschwindigkeit wird anhand der Rechenzeit ermittelt, wobei jedes Verfahren dreimal auf denselben Datensätzen ausgeführt wird, um zufällige Schwankungen der Messwerte auszugleichen und einen belastbaren Vergleich der mittleren Berechnungsdauer zu ermöglichen.

3.4.4 Wahl der Trackingdatensätze

Zur Berechnung des zeitlichen Versatzes zwischen realem und virtuellem Kamerabild werden drei aus der bildbasierten Bewegungsschätzung abgeleitete Datensätze untersucht: die Verschiebung zwischen aufeinanderfolgenden Bildern (px/Frame), die daraus berechnete Geschwindigkeit (px/s) und die Beschleunigung (px/s²). Der in OpenCV implementierte Lucas-Kanade-Algorithmus verfolgt ausgewählte Merkmalspunkte über aufeinanderfolgende Bilder und ermittelt ihre Positionen im jeweils folgenden Bild. Aus der Differenz zwischen den vorherigen und den neu bestimmten Positionen ergeben sich die zweidimensionalen Verschiebungsvektoren (vgl. OpenCV o. J.). Die Verschiebung stellt innerhalb der verwendeten Verarbeitungskette die unmittelbar aus der Bewegungsschätzung hervorgehende und damit am wenigsten weiterverarbeitete Größe dar. Die Geschwindigkeit ergibt sich durch Division der Verschiebung durch den zeitlichen Abstand zwischen den Bildern. Die Beschleunigung wird anschließend durch numerische Differentiation der Geschwindigkeit mithilfe der NumPy-Funktion np.gradient bestimmt. Diese verwendet innerhalb des Datensatzes zentrale Differenzen und an dessen Rändern einseitige Differenzen (vgl. NumPy Developers o. J.). Bei videobasierten Bewegungsmessungen können die ermittelten Verschiebungswerte durch Messrauschen beeinflusst werden. Bei der Differentiation kann sich dieses Rauschen auf die daraus bestimmten Geschwindigkeits- und Beschleunigungswerte übertragen und verstärkt hervortreten (vgl. Alonso et al. 2005, S. 1085). Als einen gebräuchlichen Ansatz zur

Verarbeitung kinematischer Daten beschreiben Crenna et al. die numerische Differentiation mit anschließender Tiefpassfilterung (vgl. Crenna et al. 2021, S. 2). Entsprechend werden in dieser Arbeit zunächst die drei Datensätze gebildet und anschließend jeweils separat gefiltert. Welcher der drei Datensätze bei der Latenzbestimmung die höchste Genauigkeit und die zuverlässigsten Korrelationsergebnisse liefert, lässt sich anhand dieser theoretischen Eigenschaften nicht abschließend bestimmen. Dies wird daher im folgenden Kapitel anhand der durchgeführten Testreihen empirisch untersucht.

3.4.5 Filterung und Normierung

Als Filter für die Datenkurven wird ein mit scipy.signal.butter erzeugter Butterworth-Tiefpass gewählt. Dieser Filtertyp besitzt einen maximal flachen Frequenzgang im Durchlassbereich (vgl. SciPy Community o. J.a). Die Filterung erfolgt mit scipy.signal.filtfilt, wobei der Filter zunächst vorwärts und anschließend rückwärts auf das Signal angewendet wird. Der resultierende Filter besitzt dadurch einen Phasengang von null und einen Betragsfrequenzgang, der dem Quadrat des ursprünglichen Betragsfrequenzgangs entspricht (vgl. Gustafsson 1996, S. 988). Gleichzeitig verdoppelt sich die effektive Filterordnung (vgl. SciPy Community o. J.b). Durch den Phasengang von null wird verhindert, dass die Filterung die zeitliche Lage der Signalanteile verändert und dadurch die Delaybestimmung beeinflusst.

Anschließend werden die Datensätze durch ihren jeweiligen maximalen Absolutwert dividiert. Dadurch liegen die unterschiedlich skalierten Bewegungswerte von Kamera- und UE-Tracking im Bereich zwischen −1 und 1. Da diese Normierung einer Multiplikation mit einem positiven konstanten Faktor entspricht, verändert sie die Position des Kreuzkorrelationsmaximums und damit den ermittelten Delay nicht. Sie vereinheitlicht jedoch die Größenordnung der Eingangssignale. Die Vergleichbarkeit der als Gewichte verwendeten Korrelationswerte ergibt sich aus der im Programm verwendeten normierten Kreuzkorrelation.

Die Unterschiede zwischen den ungefilterten sowie den gefilterten und normierten Geschwindigkeitsdatensätzen sind auf der folgenden Seite in Abbildungen 4 und 5 zu betrachten.

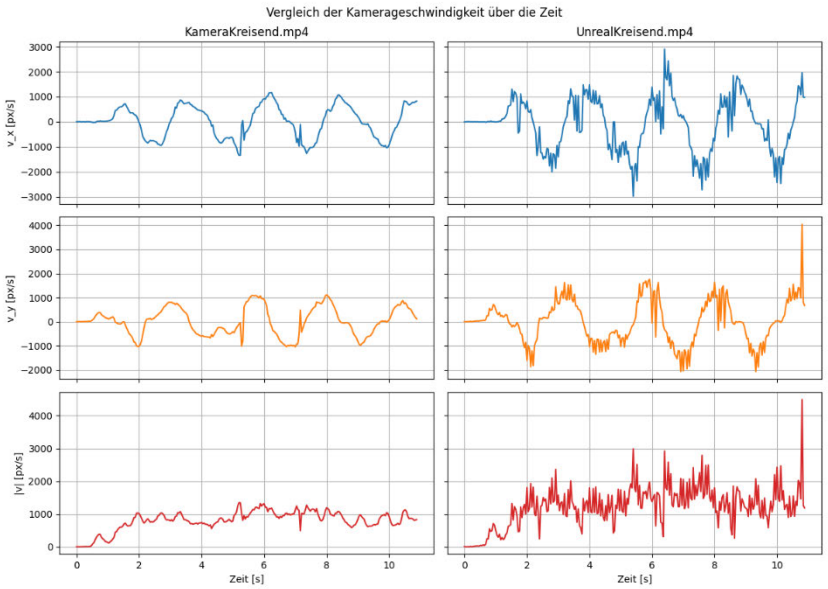


Abbildung 4: Kamerageschwindigkeit Vorder- und Hintergrund ungefiltert und nicht normiert

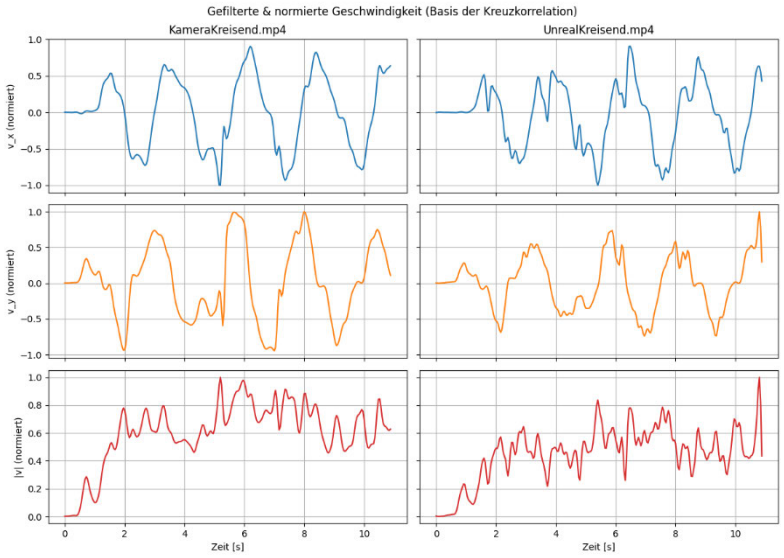


Abbildung 5: Kamerageschwindigkeit Vorder- und Hintergrund gefiltert und normiert

3.4.6 Messungen

Zur Auswahl einer geeigneten Kombination aus Korrelationsverfahren und Trackingdatensatz wurden die vier Verfahren Kreuzkorrelation, FFT-Kreuzkorrelation, Least Squares und Peak Matching mit sechs Varianten der Bewegungsdaten untersucht.

Berücksichtigt wurden Position, Geschwindigkeit und Beschleunigung, jeweils ungefiltert sowie gefiltert und normiert. Als Testmaterial dienten fünf aufgezeichnete Bewegungsmuster: horizontale, vertikale und diagonale Bewegung, Kreisbewegung und Rütteln. Für jede Kombination wurde der zeitliche Versatz getrennt für die x-Achse, die y-Achse und den Bewegungsbetrag berechnet. Die Rechenzeit wurde dreimal bestimmt; für die Auswertung wurde jeweils der niedrigste Wert verwendet. Güte-Rohwert und Delaywerte selbst wurden nicht wiederholt gemessen. Aus den fünf Bewegungsmustern wurden für jede Kombination aus Korrelationsverfahren und Datensatzvariante drei Kennwerte gemittelt: die Rechenzeit (Anlage 1, Tabelle 9), der verfahrenseigene Güte-Rohwert (vgl. Anlage 1, Tabelle 10-13) sowie die Konsistenz als mittlere paarweise Abweichung zwischen den Delaywerten von x-Achse, y-Achse und Betrag. Sie wurde als (|d_x − d_y| + |d_x − d_Betrag| + |d_y − d_Betrag|) / 3 berechnet (vgl. Anlage 1, Tabelle 14). Da kein exakter Soll-Delay als Referenz vorlag, diente die Konsistenz ausschließlich als internes Übereinstimmungsmaß; eine Aussage zur absoluten Messgenauigkeit war damit nicht möglich. Da sich die Güte-Rohwerte der vier Verfahren in Definition und Wertebereich unterscheiden, sind sie nur innerhalb desselben Verfahrens vergleichbar (vgl. Anlage 1, Tabellen 10-13).

3.4.7 Betrachtung der Ergebnisse

Bei der Betrachtung der Abweichungen zwischen x-Achse, y-Achse und Bewegungsbetrag erreichten die Kreuzkorrelation, die FFT-Kreuzkorrelation und Least Squares in Verbindung mit der gefilterten und normierten Geschwindigkeit jeweils den niedrigsten Wert von 0,0267 s (vgl. Anlage 1, Tabelle 14). Die Übereinstimmung der beiden Kreuzkorrelationsvarianten ist nachvollziehbar, da sie dieselbe Korrelationsfunktion auf unterschiedlichen Berechnungswegen bestimmen (vgl. Kapitel 3.4.3). Insgesamt lieferten die Geschwindigkeitsdaten konsistentere Ergebnisse als die Positionsdaten. Die Beschleunigungsdaten wiesen mit Abweichungen von bis zu 0,7947 s die größten Schwankungen auf. Eine mögliche Ursache ist die Verstärkung des Messrauschens durch die zusätzliche numerische Ableitung. Peak-Matching erzielte bei allen untersuchten Datensätzen deutlich größere Abweichungen als die anderen Verfahren. Bei der Kreuzkorrelation und der FFT-Kreuzkorrelation erreichte die gefilterte und normierte Position mit 0,843 den höchsten mittleren Güte-Rohwert. Die gefilterte und normierte Geschwindigkeit lag mit 0,823 nur etwa 2,4 % darunter (vgl. Anlage 1, Tabellen 10-13). Bei Least Squares war lediglich der Gütewert der gefilterten und normierten Geschwindigkeit positiv. Beim Peak Matching erzielte die ungefilterte Beschleunigung zwar den höchsten verfahrensinternen Gütewert, gleichzeitig waren die daraus bestimmten Delaywerte jedoch wenig konsistent. Der Güte-Rohwert allein reicht bei diesem Verfahren daher nicht zur Bewertung der

Ergebnisqualität aus. Mit einer mittleren Rechenzeit von 0,192 ms war die direkte Kreuzkorrelation das schnellste der untersuchten Verfahren (vgl. Anlage 1, Tabelle 9). Peak Matching benötigte ungefähr die doppelte Rechenzeit. Die FFT-Kreuzkorrelation war bei den verwendeten kurzen Datenreihen etwa 4,23-mal langsamer. Kaso (2018, S. 2) beschreibt die FFT-basierte Berechnung der normalisierten Kreuzkorrelation und weist darauf hin, dass mögliche Geschwindigkeitsvorteile von der Größe der untersuchten Daten- und Suchbereiche abhängen. Bei den hier verwendeten kurzen Datenreihen überwog daher vermutlich der zusätzliche Aufwand der Transformation. Least Squares wies mit etwa dem 36-Fachen der Rechenzeit den höchsten Rechenaufwand auf.

Für die weitere Entwicklung wird deshalb die direkte Kreuzkorrelation in Verbindung mit der gefilterten und normierten Geschwindigkeit verwendet. Diese Kombination erreichte eine geringe Abweichung zwischen den drei Bewegungsgrößen, einen hohen Güte-Rohwert und die niedrigste Rechenzeit. Die FFT-Kreuzkorrelation lieferte vergleichbare Ergebnisse, benötigte bei den verwendeten Datenfenstern jedoch mehr Rechenzeit. Least Squares und Peak Matching boten im untersuchten Aufbau keine entsprechenden Vorteile. Die vollständigen Messwerte und der Vergleich sämtlicher Kombinationen sind im Anhang aufgeführt.

3.4.8 Verrechnung zu einem Latenzwert

Die Kreuzkorrelation liefert für die x-Achse, die y-Achse und den Bewegungsbetrag jeweils einen eigenen Delaywert. Diese Ergebnisse können aufgrund unterschiedlich stark ausgeprägter Bewegungsrichtungen, des Messrauschens sowie mehrerer ähnlich hoher Korrelationsmaxima voneinander abweichen. Für die Latenzberechnung wird jedoch ein einzelner Gesamtwert benötigt. Daher werden die drei Delaywerte mithilfe eines korrelationsgewichteten arithmetischen Mittels zusammengeführt.

Im Unterschied zum ungewichteten Mittel gehen die Einzelergebnisse beim gewichteten arithmetischen Mittel mit unterschiedlichen Gewichtungsanteilen in die Berechnung ein (vgl. Stegen 2021, S. 10, 13). Im vorliegenden Verfahren dienen die nichtnegativen normierten Korrelationswerte als Gewichte. Delaywerte mit einer höheren Korrelationsgüte erhalten dadurch einen größeren Einfluss auf das Gesamtergebnis. Das Verfahren wurde gewählt, weil es einfach zu implementieren ist, nur einen geringen zusätzlichen Rechenaufwand verursacht und bei gültigen Eingangswerten einen einzelnen Gesamtwert liefert. Die Verwendung der Korrelationswerte als Gewichte stellt eine für den vorliegenden Anwendungsfall festgelegte heuristische Entscheidungsregel dar und ist nicht als statistisch optimale Gewichtung zu verstehen.

Der korrelationsgewichtete Delaywert wird folgendermaßen berechnet:

Allgemeine Formel

$$d_{final} = \frac{d_x \cdot r_x + d_y \cdot r_y + d_B \cdot r_B}{r_x + r_y + r_B}$$

Dabei bezeichnen d_x, d_y und d_B die ermittelten Delays der x-Achse, der y-Achse und des Bewegungsbetrags sowie r_x, r_y und r_B die zugehörigen normierten Korrelationsfaktoren.

Rechenbeispiel: Kreisbewegung

Für die gefilterten und normierten Geschwindigkeitsdaten der Kreisbewegung wurden die folgenden Werte bestimmt:

Auswertung	Delay d_i [s]	Korrelationsfaktor r_i
x-Achse	−0,24	0,918
y-Achse	−0,16	0,955
Bewegungsbetrag	−0,20	0,780

Tabelle 1: Ergebnisse der Kreuzkorrelation für die gefilterten und normierten Geschwindigkeitsdaten der Kreisbewegung

Einsetzen der Messwerte

$$d_{final} = \frac{(-0{,}24 \cdot 0{,}918) + (-0{,}16 \cdot 0{,}955) + (-0{,}20 \cdot 0{,}780)}{0{,}918 + 0{,}955 + 0{,}780}$$

$$d_{final}'' = \frac{-0{,}52912}{2{,}653}$$

$$d_{final} = -0{,}1994\ \text{s} \approx -0{,}20\ \text{s}$$

3.4.9 Zusammenfassung

In der ersten Entwicklungsphase wurde ein Programm zur automatisierten Latenzbestimmung zwischen zwei offline aufgenommenen Videos entwickelt. Für die Auswertung wurden die Kreuzkorrelation gefilterter und normierter Geschwindigkeitswerte sowie der korrelationsgewichtete Mittelwert zur Bestimmung des Gesamtergebnisses ausgewählt.

In der dargestellten Beispielmessung wurde eine Verzögerung von fünf Frames beziehungsweise 0,199 s ermittelt. Nach der entsprechenden zeitlichen Verschiebung stimmen die Bewegungsverläufe weitgehend überein. Der mittlere Korrelationskoeffizient von 0,884 unterstützt die optische Bewertung (siehe Abbildung 6) und zeigt, dass sich

das entwickelte Verfahren grundsätzlich zur Latenzbestimmung zwischen den beiden Videos eignet.

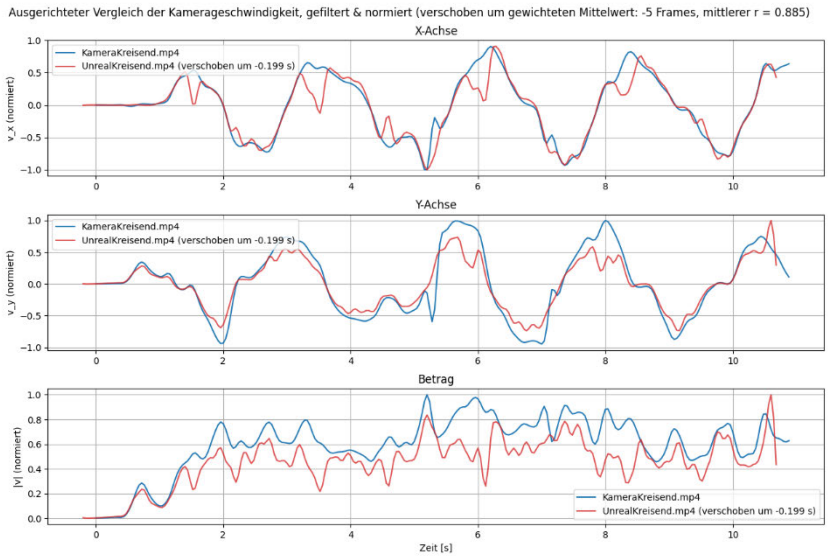


Abbildung 6: Gefilterte und normierte Geschwindigkeitskurven nach Anwendung des berechneten Delays zur optischen Überprüfung

3.5 Zweite Entwicklungsphase Latenzberechnung

3.5.1 Ziel

In der zweiten Entwicklungsphase wird das in der ersten Phase entwickelte Programm von vorab aufgezeichnetem Videomaterial auf live eingehende Videoquellen übertragen. Dabei wird weiterhin die Kreuzkorrelation der gefilterten und normierten Geschwindigkeitsdaten verwendet. Der resultierende Delaywert wird wie zuvor als korrelationsgewichteter Mittelwert aus den Einzelergebnissen der x-Achse, der y-Achse und des Bewegungsbetrags berechnet. Gleichzeitig soll in dieser Entwicklungsphase bereits mit den Vorarbeiten für das Einlesen der über die SDI-Eingänge der Blackmagic DeckLink Duo 2 eingehenden Videosignale begonnen werden, sodass die entsprechende Schnittstelle frühzeitig vorbereitet und in der nächsten Phase in die Gesamtanwendung integriert werden kann.

3.5.2 Funktionsweise der Live-Latenzberechnung

Als Live-Videoquellen werden in der zweiten Entwicklungsphase der Latenzerkennung die Innen- und Außenkamera eines Surface Pro 7 verwendet. Anders als beim Offline-Verfahren müssen die Kameraeinstellungen unmittelbar beim Öffnen der Streams festgelegt werden. Da die automatisch ausgehandelten Framerates deutlich schwankten und keine zuverlässige zeitliche Zuordnung der Bewegungsdaten ermöglichten, wird für

beide Kameras eine feste Zielframerate von 30 fps angefordert. Zusätzlich wird die automatische Belichtungsregelung deaktiviert und ein kurzer fester Belichtungswert eingestellt. Abbildung 7 zeigt die Framerates der Innen und Außenkamera bei diesen festgelegten Werten über einen Zeitraum von 30 Sekunden. Dadurch soll die Bewegungsunschärfe reduziert werden, die als wesentliche Ursache der anfänglich beobachteten Sprünge bei der Delaybestimmung vermutet wurde. Da die Übernahme dieser Einstellungen vom jeweiligen Kameratreiber abhängt, werden die vom Treiber zurückgegebenen Werte beim Programmstart ausgegeben.

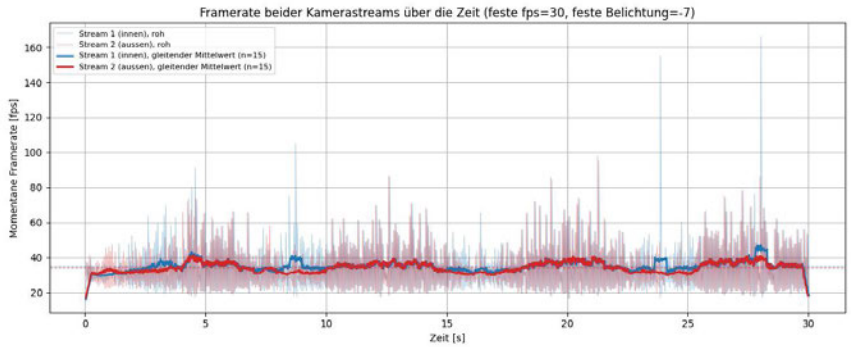


Abbildung 7: Frameraten der Innen- und Außenkamera bei 30 festen fps und einer festen Belichtung von -7

Als Treiberschnittstelle wird DirectShow verwendet. Die zunächst getestete Media-Foundation-Schnittstelle lieferte die Frames teilweise stoßweise mit hohen kurzzeitigen Ausgaberaten und anschließenden Pausen. Zu Beginn jedes Programmlaufs wird einmalig die beim seriellen Auslesen beider Kameras effektiv erreichte Framerate gemessen. Dieser Wert dient als Rückfallwert, solange noch nicht genügend Zeitstempel für eine laufende Schätzung der Framerate vorliegen. Die Größe der Ringpuffer ist davon unabhängig fest auf 150 Einträge eingestellt. Dies entspricht bei der angenommenen Framerate von 30 fps einem Zeitraum von ungefähr fünf Sekunden.

In dieser ersten Liveversion erfolgt die gesamte Verarbeitung seriell im Hauptthread. Pro Schleifendurchlauf wird zunächst ein Frame der Innenkamera und anschließend ein Frame der Außenkamera eingelesen. Direkt nach der Rückgabe des jeweiligen Lesevorgangs wird der Frame mit einem hochauflösenden Zeitgeber versehen. Dadurch entsteht ein kleiner, von der Auslesereihenfolge abhängiger zeitlicher Abstand zwischen den Frames. Die Zeitstempel beschreiben den Zeitpunkt, zu dem OpenCV den jeweiligen Frame bereitstellt. Interne Kamera- oder Treiberpuffer können daher nicht vollständig berücksichtigt werden.

Während beim Offline-Verfahren das vollständige Video vorliegt, müssen die Bewegungsdaten im Livebetrieb zunächst über einen bestimmten Zeitraum gesammelt

werden. Für jede Kamera speichert ein Ringpuffer die Verschiebungen in x- und y-Richtung gemeinsam mit den zugehörigen Zeitstempeln. Beim Eintreffen eines neuen Wertes wird der älteste Eintrag automatisch entfernt. Da die Puffergröße auf 150 Werte begrenzt ist, beträgt der erfasste Zeitraum bei 30 fps ungefähr fünf Sekunden. Bei einer abweichenden tatsächlichen Framerate verändert sich die zeitliche Länge des gespeicherten Ausschnitts entsprechend. Um den Speicherbedarf und die Rechenlast gering zu halten, werden ausschließlich die bereits berechneten Bewegungswerte und keine Rohbilder gespeichert. Der in Phase 1 entwickelte Trackingalgorithmus bleibt im Kern unverändert. Während das Offline-Verfahren ein vollständiges Video in einem Durchlauf verarbeitet, wird der Trackingalgorithmus im Livebetrieb inkrementell auf jeden neu eingelesenen Frame angewendet. Auch das in Entwicklungsphase 1 hergeleitete Korrelationsverfahren bleibt grundsätzlich bestehen, wird jedoch an die fortlaufende Auswertung angepasst. Sobald beide Ringpuffer vollständig gefüllt sind, wird ungefähr einmal pro Sekunde ein neuer Delaywert berechnet. Bei einer tatsächlichen Framerate von 30 fps umfasst das gleitende Analysefenster etwa fünf Sekunden. Zwei aufeinanderfolgende Fenster überschneiden sich damit um ungefähr vier Sekunden beziehungsweise 80 %. Die Ergebnisse beruhen folglich überwiegend auf denselben Bewegungsdaten und werden bei jedem Durchlauf um die neu hinzugekommenen Werte aktualisiert. Da die Delayberechnung synchron im Hauptthread erfolgt, wird die Kameraerfassung während eines Auswertungsdurchlaufs kurzzeitig unterbrochen. Diese Blockierung stellt eine Einschränkung der linearen Programmstruktur dar und bildet einen Ausgangspunkt für die anschließende threadbasierte Weiterentwicklung. Für jeden Auswertungsdurchlauf wird die effektiv erreichte Framerate beider Kamerastreams anhand der Zeitstempel im aktuellen Analysefenster neu geschätzt. Anschließend werden beide Werte gemittelt, da für die Kreuzkorrelation eine gemeinsame Framerate benötigt wird. Diese gemittelte Framerate wird anstelle der stark schwankenden Zeitdifferenzen einzelner Frames zur Berechnung der Bewegungsgeschwindigkeit verwendet. Dadurch führen stoßweise vom Treiber bereitgestellte Frames nicht unmittelbar zu unrealistisch hohen Geschwindigkeitsspitzen. Die Zeitstempel bleiben dennoch für die zeitliche Einordnung der Bewegungswerte erhalten. Da sie unregelmäßig verteilt sein können, werden die Bewegungsdaten beider Kameras durch lineare Interpolation auf ein gemeinsames, gleichmäßiges Zeitraster innerhalb ihres zeitlich überlappenden Bereichs übertragen. Die geschätzte Framerate bestimmt sowohl den Abstand als auch die Anzahl der Rasterpunkte. Darüber hinaus wird sie zur Normierung der Grenzfrequenz des Tiefpassfilters und zur Umrechnung des ermittelten Sample-Versatzes in Sekunden verwendet.Vor der Kreuzkorrelation werden die Geschwindigkeitskurven, wie im Programm für voraufgenommene

Videos auch, mit einem Butterworth-Tiefpass gefiltert, nur hier bei einer Grenzfrequenz von 3 Hz. Anschließend werden die Geschwindigkeitskomponenten auf ihren jeweiligen maximalen Absolutwert normiert. Liegt das Maximum einer Komponente nahezu bei null, wird anstelle einer Division durch null ein Nullsignal erzeugt. Dadurch wird verhindert, dass fehlende Bewegung zu ungültigen NaN-Werten in der weiteren Berechnung führt. Der bei der Kreuzkorrelation untersuchte Verschiebungsbereich wird auf ±1 Sekunde um den Nullpunkt begrenzt. Bei annähernd periodischen Bewegungen können andernfalls mehrere ähnlich starke Nebenmaxima im Abstand der Bewegungsperiode auftreten, zwischen denen das Ergebnis unvorhersehbar wechseln kann. Die Begrenzung ist vertretbar, da für den untersuchten Aufbau eine Systemlatenz von unter einer Sekunde erwartet wird. Die Kreuzkorrelation bestimmt den Versatz jeder Bewegungsgröße zunächst in ganzen Samples. Nicht ganzzahlige Endwerte können anschließend durch die gewichtete Kombination der unterschiedlichen Achsenergebnisse entstehen.

Für die x- und y-Richtung sowie den Bewegungsbetrag werden jeweils ein Delaywert und ein normierter Korrelationsfaktor berechnet. Eine Größe wird nur dann im korrelationsgewichteten Mittelwert berücksichtigt, wenn ihr Faktor mindestens 0,5 beträgt. Erreicht keine der drei Größen diesen Schwellwert, wird kein gemeinsamer Delaywert berechnet und der Durchlauf verworfen. Der Gesamtfaktor wird ausschließlich aus den tatsächlich berücksichtigten Größen gebildet und ebenfalls gegen einen Schwellwert von 0,5 geprüft. Verworfene Durchläufe werden weiterhin zu Diagnosezwecken in der Konsole und im Ergebnisplot dargestellt, fließen jedoch weder in die exponentielle Glättung noch in den abschließenden Mittelwert ein.

Gültige Delaywerte werden mit einem exponentiell gleitenden Mittelwert geglättet. Bei einem Glättungsfaktor von $\alpha = 0{,}5$ gehen der neu berechnete Wert und das bisherige geglättete Ergebnis jeweils mit einem Anteil von 0,5 ein. Der erste gültige Delaywert initialisiert die Glättung unmittelbar, sodass kein Einschwingen ausgehend von null entsteht. Zusätzlich wird die Anzahl der bisher gültigen Durchläufe mitgezählt und in der Konsolenausgabe angezeigt.

Für die grafische Auswertung werden die ersten 30 Berechnungsdurchläufe zwischengespeichert. Der Ergebnisplot zeigt die rohen Delaywerte und die zugehörigen Gesamtfaktoren, wobei gültige und verworfene Durchläufe unterschieden werden. Anschließend werden der mittlere Delaywert und der mittlere Korrelationsfaktor ausschließlich aus den gültigen Durchläufen berechnet. Die Zeitachse des Diagramms basiert auf der Nummer des Messwertes und dem vorgesehenen Analyseintervall von einer Sekunde. Sie entspricht daher nicht exakt der tatsächlich vergangenen Zeit. Ein zweiter Diagnoseplot

stellt die gefilterten und normierten Geschwindigkeitskurven beider Kameras dar. Die Kurve der zweiten Kamera wird um den für die jeweilige Achse berechneten Versatz verschoben, sodass die Übereinstimmung der Kurven für jeden errechneten Delaywert optisch überprüft werden kann.

Eine Berechnung wird nur durchgeführt, wenn der zeitlich überlappende Bereich beider Puffer mindestens zehn Werte enthält, da die Filterung andernfalls nicht zuverlässig ausgeführt werden kann. Insgesamt überträgt die lineare Liveversion die im Offline-Verfahren entwickelte Berechnungsmethode auf fortlaufend eingelesene Kamerastreams. Das serielle Auslesen und die synchrone Latenzberechnung führen jedoch weiterhin zu kurzen Blockierungen und einem von der Auslesereihenfolge abhängigen zeitlichen Abstand. Diese Einschränkungen werden im folgenden Entwicklungsschritt durch eine threadbasierte Programmstruktur adressiert. Abbildung 9 zeigt den Verlauf von 30 Aufeinander folgenden Latenzmessungen mit Korrelationswerten. Abbildung 8 zeigt beispielhaft einen errechneten Berechnungsintervall mit einem Delaywert von 0 ms der x- und y-achsen der Kameras.

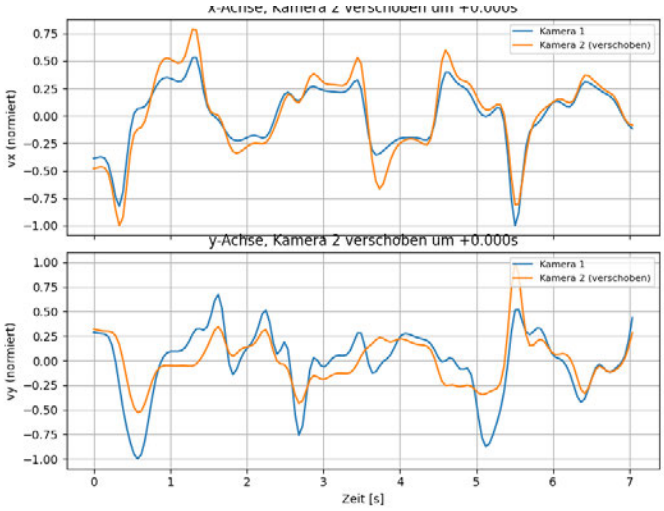


Abbildung 9: X- und Y-Achse eines Berechnungsintervalls

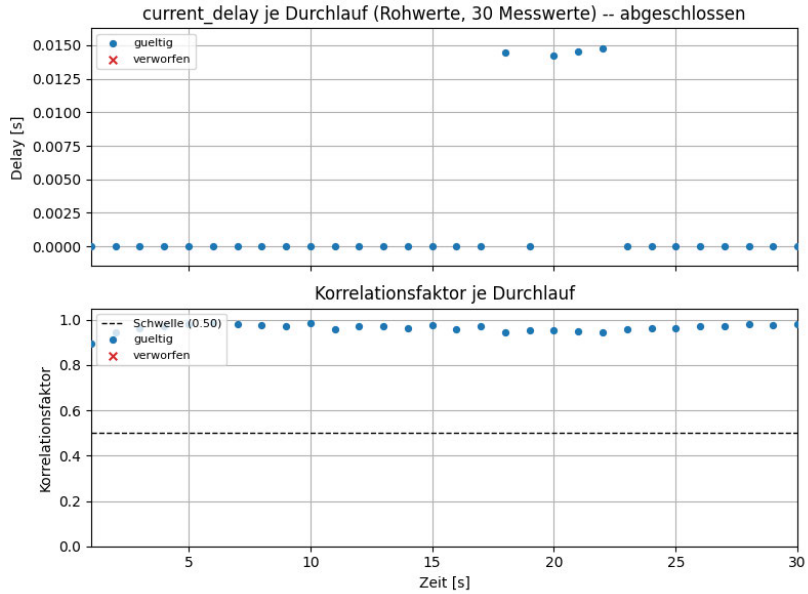


Abbildung 8: Delay- und Korrelationswerte über einen Berechnungszeitraum von 30 s des linearen Programms

3.5.3 Umstellung auf eine threadbasierte Programmstruktur und Downscaling

In der linearen Version wurden die Frames beider Kameras nacheinander im Hauptthread eingelesen. Dadurch entstand ein kleiner, von der Auslesereihenfolge abhängiger Zeitversatz. In der threadbasierten Variante besitzt jede Kamera einen eigenen CameraReader, der kontinuierlich Frames erfasst und unmittelbar nach dem Einlesen mit einem Zeitstempel versieht. Eine threadsichere Warteschlange mit einer Größe von einem Element hält jeweils nur den neuesten Frame. Wurde der vorherige Frame noch nicht abgeholt, wird er verworfen. Dadurch entsteht bei einer kurzzeitig langsameren Verarbeitung kein wachsender Rückstau. Der Hauptthread ruft die Frames mit einem

festgelegten Timeout ab, während beide Kameras unabhängig voneinander weitergelesen werden. Die getrennt erfassten Zeitstempel bleiben bei der anschließenden Verarbeitung erhalten.

Zusätzlich wurde ein Downscaling auf eine feste Bildbreite von 640 Pixeln implementiert. Dabei bleibt das Seitenverhältnis erhalten. Feature-Erkennung, optischer Fluss und affine Bewegungsschätzung werden auf dem verkleinerten Graustufenbild ausgeführt. Die ermittelten Verschiebungen werden anschließend mit dem Skalierungsfaktor auf die ursprüngliche Auflösung zurückgerechnet, bevor sie in den Ringpuffern gespeichert werden.

In der verwendeten Konfiguration hatte das Downscaling zunächst keine praktische Auswirkung. Obwohl die Kameras des Surface Pro 7 laut Hersteller Full-HD-Video unterstützen, wurden die Kamerabilder über OpenCV lediglich mit einer Auflösung von 640 × 480 Pixeln eingelesen. Da die Bildbreite bereits dem Zielwert von 640 Pixeln entsprach, fand keine tatsächliche Verkleinerung statt und der Skalierungsfaktor betrug 1. Eine Verringerung des Rechenaufwands konnte in diesem Aufbau daher nicht untersucht werden. Das Downscaling bleibt dennoch Bestandteil des Programms, da es in der folgenden Entwicklungsphase bei der Verarbeitung höher aufgelöster SDI-Signale eingesetzt werden soll. Sinkt die Anzahl erfolgreich verfolgter Bildmerkmale unter den festgelegten Grenzwert, führt der Tracker automatisch eine erneute Merkmalserkennung durch.

Auch die Delayberechnung wurde aus dem Hauptthread ausgelagert. In der vorherigen Version blockierte der synchrone Aufruf von analyze_once() während der gesamten Berechnung die Frameverarbeitung. Nun ruft ein DelayWorker dieselbe Analysefunktion in regelmäßigen Abständen in einem eigenen Thread auf. Das jeweils aktuelle Ergebnis wird zugriffsgeschützt gespeichert und vom Hauptthread nicht blockierend abgefragt. Anhand eines Ergebniszeitstempels wird erkannt, ob eine neue Berechnung vorliegt.

Beim Beenden werden zunächst der Analyse- und die Kamera-Threads gestoppt und erst anschließend die Kameraobjekte freigegeben. Dadurch wird verhindert, dass ein Thread noch auf eine bereits geschlossene Videoquelle zugreift. In den durchgeführten Tests lief die threadbasierte Programmversion insgesamt stabiler und lieferte gleichmäßigere Delaywerte. In der im Anhang dargestellten Messung wurde in 29 von 30 Berechnungsdurchläufen ein Versatz von 0 Frames ermittelt. Gleichzeitig lagen die Korrelationsfaktoren wie im linearen Programm konstant über 0,9 und somit deutlich oberhalb des festgelegten Schwellwerts von 0,5. Die parallele Kameraerfassung und die ausgelagerte Delayberechnung reduzieren somit Blockierungen und Schwankungen im Programmablauf.

Abbildung 10 zeigt den Verlauf 30 aufeinander folgender Latenzmessungen des threadbasierten Programms mit Korrelationswerten. Dabei ist eine etwas höhere Kontinuität der Ergebnisse zu beobachten.

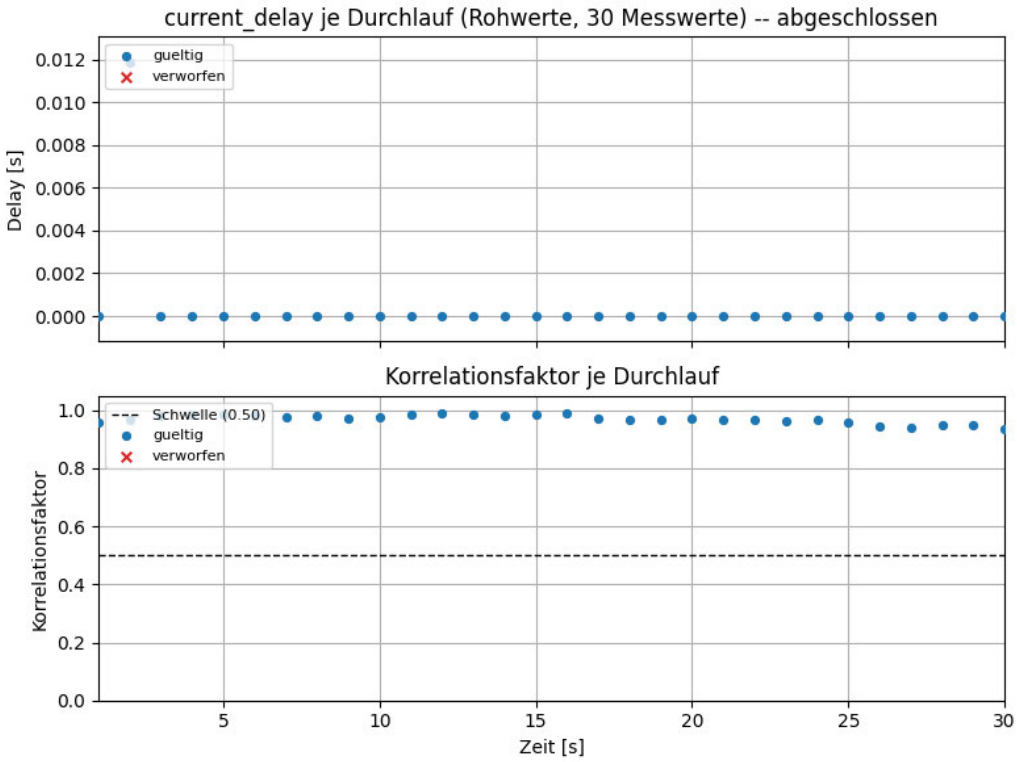


Abbildung 10: Delay- und Korrelationswerte über einen Berechnungszeitraum von 30 s des threadbasierten Programms

3.5.4 Schnittstelle zur DeckLink Duo 2

Nach dem erfolglosen direkten Zugriff mit OpenCV wurde zunächst versucht, die SDI-Signale über FFmpeg einzulesen. Die installierte FFmpeg-Version enthielt jedoch keine native DeckLink-Unterstützung. Die dafür notwendige eigene Kompilierung mit dem Blackmagic DeckLink SDK erwies sich unter Windows als zu aufwendig. Alternativ wurden die DeckLink-Eingänge über DirectShow als WDM-Geräte angesprochen. FFmpeg wandelte die eingelesenen Signale in das Pixelformat BGR24 um und übergab die Frames über eine Rawvideo-Pipe an Python. Dieser Ansatz funktionierte grundsätzlich, führte beim parallelen Betrieb zweier Eingänge jedoch zu Pufferüberläufen und verworfenen Frames. Daher wurde schließlich ein C++-Beispielprogramm des DeckLink SDK mithilfe von Claude Code angepasst und um eine Python-Bridge ergänzt. Das C++-Programm übernimmt den direkten Zugriff auf beide SDI-Eingänge und stellt die empfangenen Frames über Shared Memory bereit. Die Python-Bridge liest diese Daten aus und übergibt sie als NumPy-Arrays an OpenCV. Mit diesem Ansatz konnten beide SDI-

Videoströme schließlich zuverlässig in die Anwendung zur Latenzbestimmung eingebunden werden.

3.5.5 Zusammenfassung

Das in Entwicklungsphase 1 entwickelte Programm zur Latenzmessung zwischen zwei voraufgenommenen Videos wurde erfolgreich auf die kontinuierliche Verarbeitung zweier Live-Videostreams übertragen. Durch eine threadbasierte Ausführungsstruktur und das Downscaling vor dem Tracking konnten der Ressourcenbedarf und gegenseitige Blockierungen der Verarbeitungsschritte reduziert werden. Ein auf ±1 Sekunde begrenzter Verschiebungsbereich, die Bewertung anhand der Korrelationsfaktoren sowie das Verwerfen unsicherer Ergebnisse verbessern die Stabilität der Latenzbestimmung. Gültige Delaywerte werden zusätzlich exponentiell geglättet, um kurzfristige Schwankungen zwischen aufeinanderfolgenden Berechnungen zu reduzieren. Für die Einbindung der SDI-Signale wurde zudem mit Unterstützung von Claude Code eine C++-Anwendung mit Python-Bridge entwickelt, die beide DeckLink-Eingänge über Shared Memory zuverlässig für die weitere Verarbeitung bereitstellt.

3.6 Dritte Entwicklungsphase

3.6.1 Ziel

Ziel der dritten Entwicklungsphase ist die vollständige Einbindung der beiden SDI-Eingänge der DeckLink Duo 2 in ein Programm zur Delayberechnung. Die Videoströme sollen zuverlässig eingelesen und in festgelegten Intervallen zur Berechnung des zeitlichen Versatzes verwendet werden. Dabei sollen sowohl die aktuell berechneten als auch die geglätteten Delaywerte ausgegeben werden.

Zusätzlich soll eine grafische Benutzeroberfläche entwickelt werden, um die Bedienung der Anwendung zu vereinfachen. Dafür müssen die für den Betrieb relevanten Ein- und Ausgabewerte ausgewählt sowie sinnvolle Wertebereiche für die veränderbaren Parameter festgelegt werden. Abschließend sollen die einzelnen Programmkomponenten zusammengeführt, getestet und zu einer funktionsfähigen Gesamtanwendung finalisiert werden.

3.6.2 Zusammenführung des Programms mit der Schnittstelle für die DeckLink Duo 2

Als Grundlage dient die in Kapitel 3.5.4 entwickelte Schnittstelle aus einer C++-Anwendung auf Basis des DeckLink SDK und einer Python-Bridge. Für Vorder- und Hintergrund wird jeweils eine eigene Instanz der C++-Anwendung gestartet. Jede Instanz öffnet genau einen SDI-Eingang und schreibt das zuletzt vollständig empfangene Bild in einen getrennten Shared-Memory-Bereich. Neben den Bilddaten enthält der Header unter

anderem die Bildnummer, einen Hardware-Zeitstempel, Bildgröße, Zeilenbreite und den aktuellen Zustand der Quelle. Anhand einer Sequenznummer kann die Python-Seite erkennen, ob ein Schreibvorgang abgeschlossen ist, sodass kein nur teilweise aktualisiertes Bild übernommen wird. Die Übernahme eines neuen Bildes wird über ein Frame-Ready-Ereignis signalisiert. Die Python-Bridge muss den Speicher dadurch nicht in einem festen Intervall abfragen, sondern wird beim Eintreffen eines Bildes geweckt. Falls das Ereignis nicht verfügbar ist, bleibt ein kurzes Polling-Intervall als Rückfalllösung erhalten. Für die Delayberechnung wird aus jedem neuen YUV-Bild unmittelbar die Luminanzkomponente extrahiert und mit Bildnummer und Hardware-Zeitstempel weitergegeben. Für die Bildvorschau der Bedienoberfläche wird eine rechenintensivere RGB-Dekodierung ausgeführt, welche auf ungefähr 15 Aktualisierungen pro Sekunde begrenzt ist. Damit beeinflusst die Vorschau die für das Tracking benötigte Framerate möglichst wenig. Zwischen SDI-Reader und Delay-Engine befindet sich pro Quelle eine begrenzte Warteschlange mit zwölf Bildern. Kurze Verarbeitungsspitzen können dadurch abgefangen werden. Ist die Warteschlange vollständig belegt, wird das älteste noch nicht verarbeitete Bild verworfen. Ein stilles Duplizieren des letzten Bildes findet nicht statt. Die Bildnummern bleiben deshalb bis zum Tracking erhalten und ermöglichen später eine eindeutige Erkennung solcher Verluste. Die Quellenauswahl, die SDI-Bridge und die Delayberechnung sind in getrennten Komponenten gekapselt. Der zentrale Controller startet und beendet die beiden Quellen, prüft ihre Konfiguration und führt die Bilder über jeweils einen Feed-Thread der Delay-Engine zu. Vorder- und Hintergrund werden nur übernommen, wenn beide Quellen aktiv sind und dieselbe Framerate liefern. Für den vorgesehenen Produktionsbetrieb ist die Berechnung auf 50 Vollbilder pro Sekunde ausgelegt.

3.6.3 Entwicklung eines Bewegungstriggers

Die Delayberechnung war ursprünglich als durchgehender Prozess geplant: Nach jedem Berechnungszeitraum sollte ein neuer Delaywert ausgegeben werden. Der erste Testlauf am realen Versuchsaufbau zeigte jedoch, dass eine kontinuierliche Korrelation nicht sinnvoll ist. Im Stillstand enthalten beide Signale keine ausreichend ausgeprägte Bewegungsinformation, sodass kein eindeutiges Korrelationsmaximum entsteht. Die Korrelation lieferte dadurch unbrauchbare Ergebnisse: Der berechnete Delay nahm unplausibel hohe Werte an und wechselte unvorhersehbar zwischen positivem und negativem Vorzeichen (siehe Abbildung 11).



Abbildung 11: Delayausgabe einer Übergangs-GUI ohne Bewegungstrigger

Abhilfe schafft eine Bewegungstriggerung. Sie stellt sicher, dass eine Korrelation nur dann ausgeführt wird, wenn in beiden Videoquellen tatsächlich Bewegung vorliegt.

Der Bewegungstrigger funktioniert wie folgt: Jede der beiden Kameras erhält einen eigenen kalibrierten Schwellwert. Dazu wird der Geschwindigkeitsbetrag im Stillstand für beide Kameras einzeln per Knopfdruck in der GUI gemessen. Aus den Messwerten wird jeweils der Mittelwert gebildet und mit dem Faktor 2 multipliziert. Dieser Faktor wurde empirisch für den Versuchsaufbau gewählt und stellt keinen allgemeingültigen Grenzwert dar. Zur Laufzeit wird der Geschwindigkeitsbetrag pro Frame kontinuierlich und ungefiltert berechnet und mit dem jeweiligen kameraspezifischen Schwellwert verglichen. Solange kein Berechnungszeitraum aktiv ist, wird laufend geprüft, ob eine der beiden Kameras ihren Schwellwert überschreitet. Sobald dies eintritt, startet ein Zeitraum fester Länge (standardmäßig 3 Sekunden), und für die auslösende Kamera wird ein Marker („Latch") gesetzt. Dieser Marker bleibt für die gesamte Dauer des Zeitraums bestehen, unabhängig vom weiteren Verlauf des Rohsignals. Innerhalb des laufenden Zeitraums wird weiterhin geprüft, ob auch die zweite Kamera ihren Schwellwert überschreitet; ist dies der Fall, wird auch für sie der Marker gesetzt. Entscheidend ist, dass der Zeitraum bereits mit dem ersten Auslösen beginnt und nicht erst, wenn beide Kameras ausgelöst haben. Andernfalls würde die Anstiegsflanke der zeitlich früher auslösenden Kamera abgeschnitten. Diese wird jedoch für die spätere Korrelation benötigt.

Am Ende des Zeitraums erfolgt die Auswertung. Sind beide Marker gesetzt, heißt das, dass in beiden Videoquellen Bewegung registriert wurde. In diesem Fall wird der Zeitraum seit dem ersten ausgelösten Latch aus beiden Trackingdatenpuffern korreliert und ein neuer Delay-Wert berechnet. Wurde dagegen nur die eine Kamera ausgelöst, wird der Zeitraum verworfen. Es erfolgt weder eine Korrelation noch ein neues Ergebnis. In beiden Fällen werden anschließend sofort beide Marker zurückgesetzt. Ein neuer Berechnungszeitraum kann direkt beginnen, sobald erneut eine Kamera ihren Schwellwert überschreitet.

3.6.4 Weitere Verbesserungen zur Delaybestimmung

Für die SDI-Verarbeitung wurde die Zeitbasis erweitert. Jedes Tracking-Sample enthält neben der fortlaufenden Bildnummer den Hardware-Zeitstempel der DeckLink-Karte. Dadurch wird die Geschwindigkeit anhand des tatsächlichen Zeitabstands zwischen zwei Bildern berechnet. Doppelte Bilder werden ignoriert, unplausible Zeitstempel führen zum Zurücksetzen des Trackings und Sprünge der Bildnummer werden als Frameverluste erfasst. Da Vorder- und Hintergrund trotz gleicher Sollframerate nicht zeitgleich eintreffen, werden beide Geschwindigkeitsverläufe auf einen gemeinsamen Zeitbereich mit 50 Abtastungen pro Sekunde interpoliert. Fenster mit weniger als 50 gemeinsamen Samples, Zeitlücken über 60 ms oder mehr als 20 verlorenen Bildern je Quelle werden verworfen. Ein Ergebnis wird nur bei einem Korrelationsfaktor von standardmäßig mindestens 0,5 übernommen. Andernfalls bleiben der letzte gültige Delaywert sowie dessen Glättung, Minimal- und Maximalwert erhalten, während der Status des aktuellen Fensters auf „Nicht gültig“ gesetzt wird. Gültige Delaywerte können zusätzlich durch einen exponentiell gleitenden Mittelwert mit standardmäßig $\alpha = 0{,}3$ geglättet werden.

3.6.5 Entwicklung der grafischen Benutzeroberfläche

Um die Bedienbarkeit des Programms zu verbessern, wurde mit Tkinter eine grafische Benutzeroberfläche entwickelt. Für den ersten Layoutentwurf wurde Claude Code (Anthropic) auf Grundlage eigener Skizzen, der festgelegten Ein- und Ausgabewerte sowie der beschriebenen Programmfunktionen eingesetzt. Die Bedienoberfläche besteht aus zwei aufeinanderfolgenden Fenstern. Im ersten Fenster werden die SDI-Eingänge für Vorder- und Hintergrund ausgewählt und anhand einer Vorschau kontrolliert. Außerdem lassen sich das Bildformat und weitere Quelleinstellungen festlegen. Nach erfolgreicher Prüfung beider Quellen öffnet sich das zweite Fenster für die Delayberechnung. Dieses zeigt den aktuellen beziehungsweise geglätteten Delaywert, die Richtung des Versatzes, die Korrelationsgüte und den Status des letzten Berechnungsfensters. Über drei Registerkarten können die Live-Vorschau, die Geschwindigkeitsverläufe und die einstellbaren Berechnungsparameter aufgerufen werden. In der Verlaufsdarstellung werden die x- und y-Geschwindigkeiten sowie der Bewegungsbetrag beider Quellen übereinander dargestellt. Dadurch kann unmittelbar überprüft werden, wie gut die Kurven nach Anwendung

des berechneten Delays übereinstimmen. Die entstandenen Bedienoberflächen sind in den Abbildungen 12 bis 15 dargestellt.

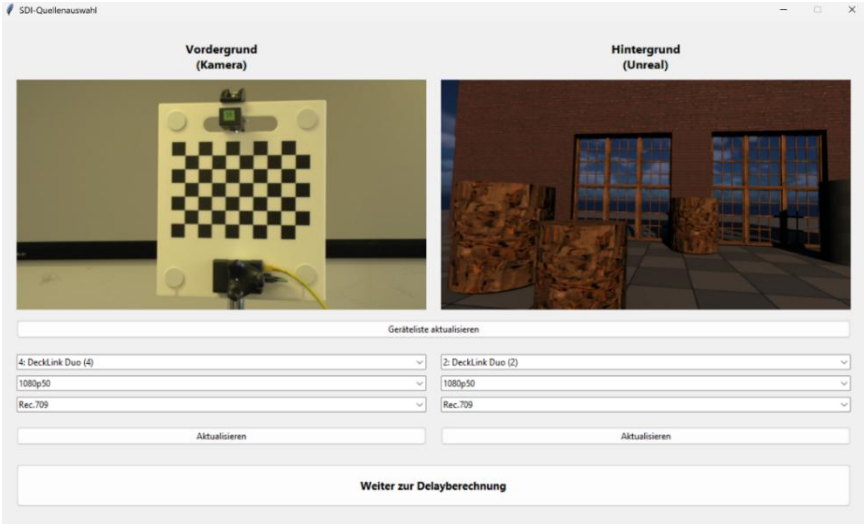


Abbildung 14: GUI 1 Quellenauswahl

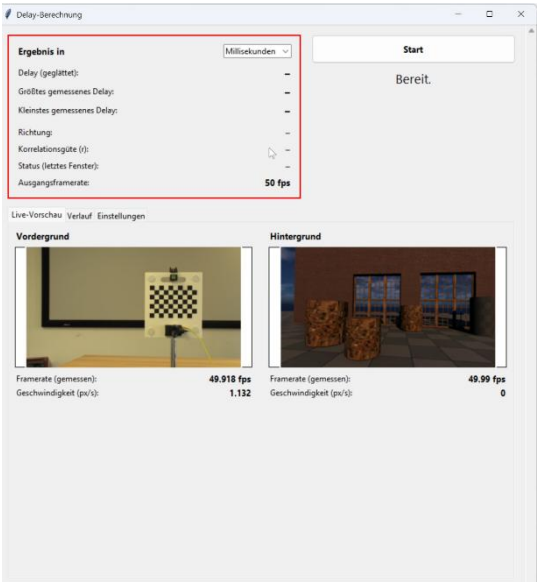


Abbildung 12: GUI 2 Delayberechnung, Livevorschau-Tab

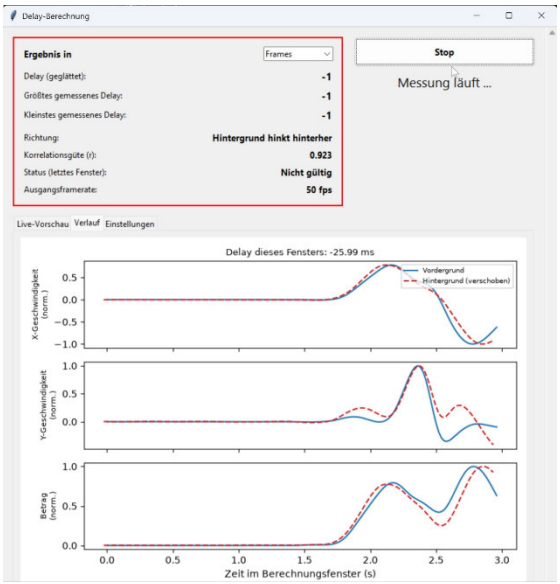


Abbildung 13: GUI 2 Delayberechnung, Verlaufs-Tab

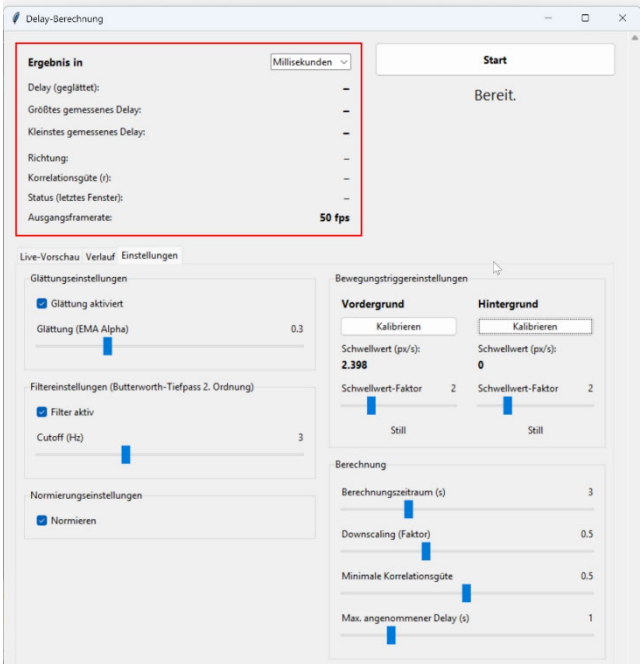


Abbildung 15: GUI 2 Delayberechnung, Einstellungs-Tab

3.6.6 Finaler Ablauf des Programms

Nach dem Start der Anwendung wird zunächst die Geräteliste eingelesen. Der Nutzer ordnet Vorder- und Hintergrund jeweils einem DeckLink-Eingang zu, legt das Videoformat fest und kontrolliert beide Bilder in der Vorschau. Mit dem Wechsel zur Latenzberechnung übernimmt der Controller die bestätigte Konfiguration. Die SDI-Prozesse und Feed-Threads laufen bereits zu diesem Zeitpunkt weiter, sodass die Bewegungstrigger vor dem eigentlichen Start der Messung kalibriert werden können. Für die Kalibrierung wird jede Quelle einzeln für die angezeigte Dauer stillgehalten. Anschließend können die Schwellwertfaktoren bei Bedarf angepasst werden. Beim Start der Messung erzeugt der Controller eine neue Delay-Engine mit den aktuellen Einstellungen. Jedes eintreffende SDI-Bild wird in Luminanzdaten umgewandelt, heruntergerechnet und verarbeitet. Aus Bildnummer und Hardware-Zeitstempel entstehen ein Geschwindigkeitswert sowie Diagnoseinformationen zu Zeitabständen und möglichen Frameverlusten. Überschreitet eine Quelle ihren kalibrierten Schwellwert, öffnet der Bewegungstrigger ein Berechnungsfenster. Nach dessen Ende werden nur dann zwei Segmente an den DelayWorker übergeben, wenn auch die zweite Quelle ausgelöst hat. Der DelayWorker prüft die Zeitbasis, bildet ein gemeinsames 50-Hz-Raster, filtert und normiert beide Signale und führt die Kreuzkorrelation für x, y und den Betrag aus. Erreicht das gewichtete Ergebnis die Mindestkorrelationsgüte, werden Rohwert, geglätteter Wert, Verlauf und Historie aktualisiert. Andernfalls bleibt das letzte gültige Ergebnis bestehen und nur der Fensterstatus

wechselt auf „Nicht gültig“. Über „Stop“ wird die Ergebniserzeugung beendet, während die Quellen für Vorschau und eine erneute Kalibrierung verfügbar bleiben.

3.6.7 Letzte Testphase und Fehleranalyse

In der abschließenden Testreihe wurden 13 gültige Berechnungsfenster bei 50 fps ausgewertet. Wie Tabelle 2 zeigt, besitzen alle Delaywerte dasselbe Vorzeichen und weisen darauf hin, dass der Hintergrund gegenüber dem Vordergrund verzögert ist. Der Mittelwert beträgt −41,042 ms beziehungsweise −2,052 Frames bei einer mittleren Korrelationsgüte von 0,906.

Messung	Delay (ms)	Delay (Frames)	Korrelationsgüte r
1	-34,251	-1,713	0,928
2	-40,000	-2,000	0,977
3	-60,000	-3,000	0,910
4	-33,336	-1,667	0,979
5	-53,539	-2,677	0,908
6	-6,359	-0,318	0,932
7	-33,205	-1,660	0,963
8	-45,904	-2,295	0,798
9	-46,745	-2,337	0,847
10	-46,995	-2,350	0,873
11	-26,776	-1,339	0,961
12	-40,000	-2,000	0,952
13	-66,433	-3,322	0,753
Durch-schnitt	**-41,042**	**-2,052**	**0,906**

Tabelle 2: Ergebnisse der abschließenden SDI-Messreihe

Die Einzelwerte liegen zwischen −6,359 und −66,433 ms, während die Korrelationsgüte zwischen 0,753 und 0,979 schwankt. Alle Messungen überschreiten damit den festgelegten Mindestwert von 0,5. Die Ergebnisse zeigen eine eindeutige Verzögerungsrichtung, zugleich aber eine Spannweite von 60,074 ms beziehungsweise ungefähr drei Bildperioden. Da kein unabhängig festgelegter Referenzversatz vorlag, können die absolute Messgenauigkeit und Wiederholpräzision nicht abschließend bestimmt werden.

Während der Tests traten zudem sporadische Sprünge der Bildnummern auf, die überwiegend das Hintergrundsignal betrafen. Dabei wurden unter anderem 14 beziehungsweise 9 fehlende Hintergrundbilder sowie ein fehlendes Vordergrund- und acht fehlende Hintergrundbilder erkannt. Obwohl die mittleren Bildabstände weiterhin 0,0200 s und die ermittelten Framerates jeweils 50,00 fps betrugen, wurden die betroffenen Fenster verworfen und nicht in die Messwerttabelle aufgenommen. Die genaue Ursache der Verluste innerhalb der SDI-Erfassung, Shared-Memory-Übergabe, Warteschlange oder weiteren Verarbeitung konnte nicht eindeutig bestimmt werden. Abbildung 16 zeigt die entsprechende Programmausgabe, die Abbildungen 17 und 18 stellen die gültigen Fenster mit dem kleinsten und größten gemessenen Delaybetrag dar. In Abbildung 18 ist zudem ein Bug des Programms zu erkennen: Die Y-Verläufe sind gespiegelt, sodass eine Bewegung nach oben in einem Signal als positive und im anderen als negative Bewegung erfasst wird.

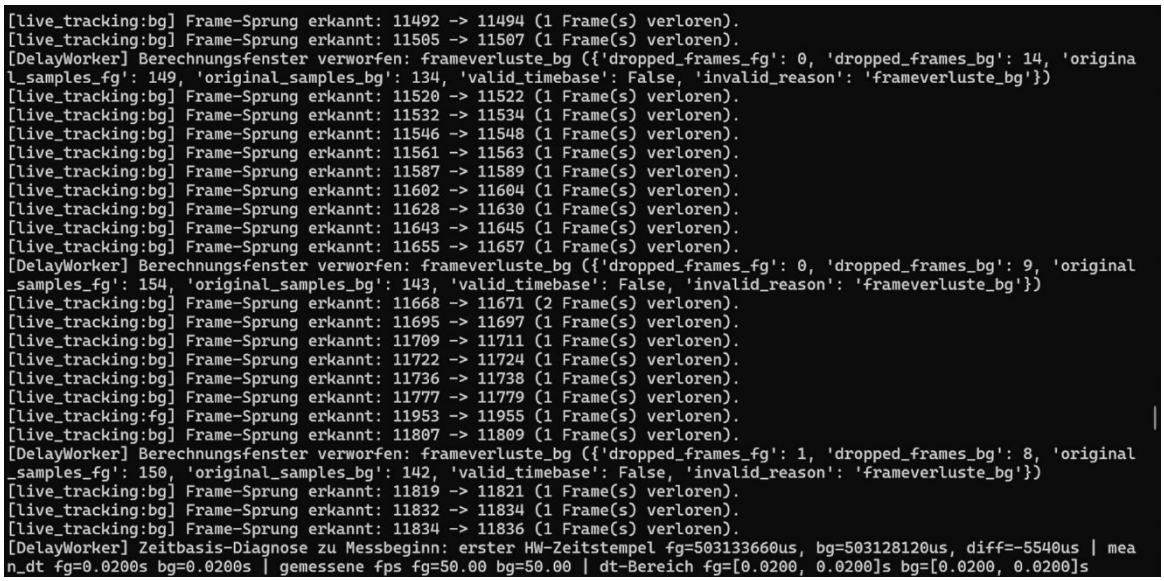


Abbildung 16: Erkennung von Frame-Sprüngen und Verwerfen betroffener Berechnungsfenster

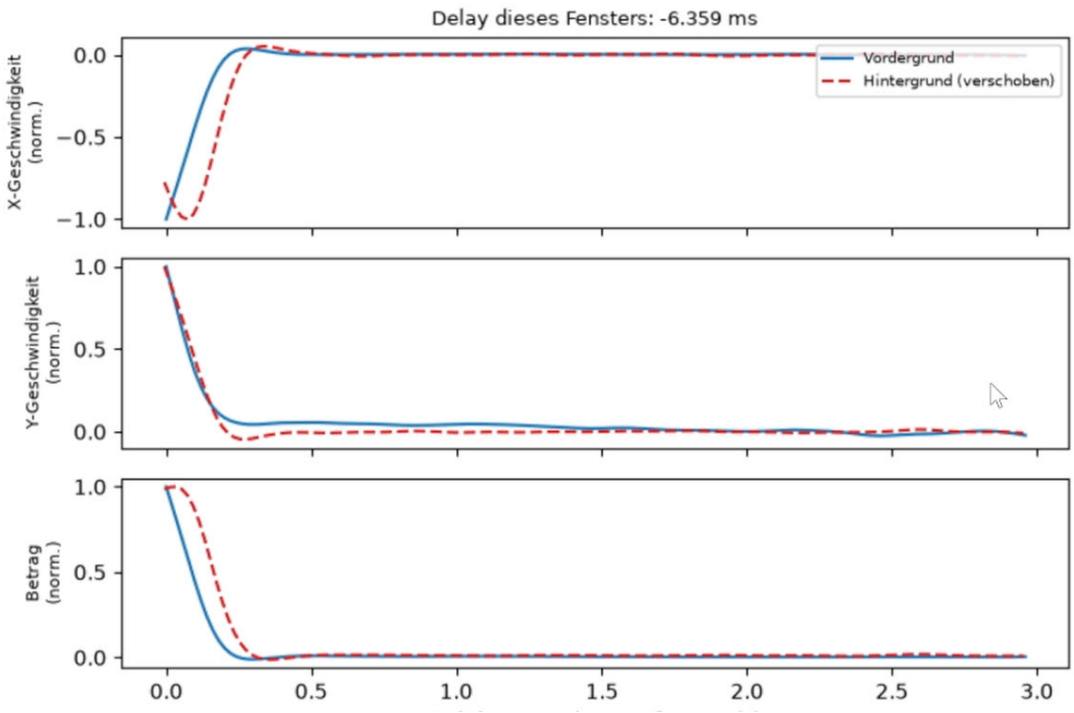


Abbildung 17: Gültiges Messfenster mit einem Delay von -6,359 ms

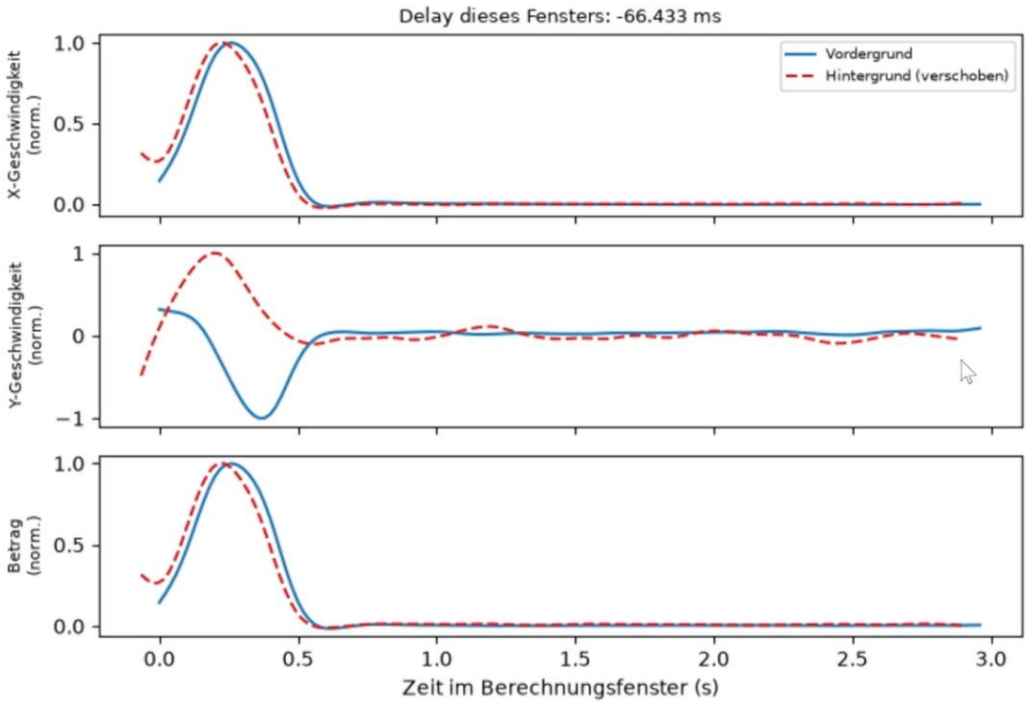


Abbildung 18: Gültiges Messfenster mit einem Delay von -66,433 ms

Zur Plausibilitätsprüfung wurde anschließend dieselbe SDI-Quelle über einen SDI-Splitter auf beide DeckLink-Eingänge verteilt. Fünf der sechs Messungen ergaben erwartungsgemäß 0 ms bei einer Korrelationsgüte von 1,000. Eine Messung lieferte dagegen trotz identischer Quelle einen Versatz von 33,379 ms beziehungsweise 1,669 Frames bei einer Korrelationsgüte von 0,985. Dadurch ergeben sich die in Tabelle 3 aufgeführten Mittelwerte von 5,563 ms, 0,278 Frames und r = 0,998. Als mögliche Ursachen kommen eine unterschiedliche zeitliche Bildübernahme, nicht erkannte Lücken oder die Auswahl eines falschen Korrelationsmaximums infrage. Die Kontrollmessung zeigt, dass auch eine hohe Korrelationsgüte keine fehlerfreie Delaybestimmung garantiert.

Messung	Delay (ms)	Delay (Frames)	Korrelationsgüte r
1	0,000	0,000	1,000
2	0,000	0,000	1,000
3	0,000	0,000	1,000
4	0,000	0,000	1,000
5	0,000	0,000	1,000
6	33,379	1,669	0,985
Durch-schnitt	**5,563**	**0,278**	**0,998**

Tabelle 3: Kontrollmessung mit identischem, über einen SDI-Splitter verteiltem Signal

3.6.8 Zusammenfassung

In der dritten Entwicklungsphase wurden die SDI-Erfassung, das optische Tracking, der Bewegungstrigger, die Delayberechnung und die grafische Benutzeroberfläche zu einem funktionsfähigen Prototyp zusammengeführt. Unzureichende Bewegung, eine zu geringe Korrelationsgüte sowie Fehler der Zeitbasis werden erkannt; betroffene Berechnungsfenster werden kontrolliert verworfen, ohne das letzte gültige Ergebnis zu überschreiben. Die abschließende Testreihe zeigte eine konsistente Richtung des gemessenen Versatzes, zugleich aber eine deutliche Streuung. Da ein festgelegter Referenzdelay fehlte, konnten die absolute Genauigkeit und die Wiederholpräzision noch nicht abschließend bewertet werden.

3.7 Abschließendes Ergebnis der Latenzberechnung

Über drei Entwicklungsphasen wurde das zunächst für voraufgezeichnete Videos entwickelte Verfahren auf Live-Videostreams und anschließend auf die SDI-Signale des Versuchsaufbaus übertragen. Die finale Anwendung wählt geeignete Bewegungsfenster aus, bestimmt den zeitlichen Versatz mittels Kreuzkorrelation und bewertet die Ergebnisse anhand der Korrelationsgüte. Erkannte Fehler der Zeitbasis und Frameverluste führen zum Verwerfen des betroffenen Berechnungsfensters. Die letzten Tests zeigen, dass das Programm zeitliche Abweichungen grundsätzlich erfassen und deren Richtung bestimmen kann. Bei der Kontrollmessung mit einer über einen SDI-Splitter verteilten Quelle trat jedoch trotz hoher Korrelationsgüte eine abweichende Messung auf. Eine zuverlässige Bestimmung des tatsächlichen Delays kann auf Grundlage der bisherigen Versuche daher noch nicht abschließend bestätigt werden. Das Verfahren stellt somit einen funktionsfähigen Prototyp für die weitere Untersuchung von Latenzen im verwendeten VP-Aufbau dar.

4 Entwicklung eines Verfahrens zur Stabilisierung von Trackingdaten

4.1 Trackingfehler innerhalb der VP

4.1.1 Problematik der Trackingfehler

Damit in der VP ein stimmiges Gesamtbild aus realem Vordergrund und virtuellem Hintergrund entsteht, muss die Bewegung der realen Kamera mit geringer Latenz und hinreichender Genauigkeit auf die virtuelle Kamera übertragen werden. Trackingdaten können jedoch auch bei stillstehender Kamera geringfügig zwischen aufeinanderfolgenden Messungen schwanken. Niehorster et al. erfassen solche als Jitter bezeichneten Präzisionsabweichungen anhand der Positions- und Orientierungswerte des HTC-Vive-Trackingsystems (2017, S. 10f). Dass bereits sehr kleine Bildschwankungen wahrnehmbar sein können, zeigen Wilmott et al. in einer psychophysischen Studie zu Jitter in AR-Head-Mounted-Displays. Unter bestimmten Betrachtungsbedingungen konnten die Versuchspersonen Bewegungen im Bereich weniger Bogenminuten und teilweise unterhalb der Pixelgröße des verwendeten Displays erkennen (vgl. Wilmott et al. 2022, S. 475f). Obwohl sich die Untersuchung auf AR-Head-Mounted-Displays und nicht unmittelbar auf VP bezieht, verdeutlicht sie die hohe Wahrnehmungsempfindlichkeit gegenüber kleinen Bildinstabilitäten. Auf die VP übertragen können entsprechende Schwankungen die räumliche Übereinstimmung zwischen realem Vordergrund und virtuellem Hintergrund beeinträchtigen und damit den Eindruck eines gemeinsamen Bildraums stören.

4.1.2 Unterscheidung von Trackingfehlern

Trackingfehler lassen sich anhand ihrer Ursache und ihres zeitlichen Verlaufs unterscheiden. Kurzfristige Schwankungen zwischen aufeinanderfolgenden Positions- und Orientierungswerten werden als **Jitter** bezeichnet. Niehorster et al. erfassen diese Schwankungen anhand des RMS zwischen aufeinanderfolgenden Messwerten eines stillstehenden HTC-Vive-Trackers (2017, S. 10f.). Davon zu unterscheiden ist **Drift**, bei der sich kleine Fehler der Positions- oder Orientierungsbestimmung über die Zeit ansammeln. Beim untersuchten HTC-Vive-System werden Position und Orientierung zwischen den Lighthouse-Messungen mithilfe von Beschleunigungs- und Drehratensensoren fortgeschrieben. Die Lighthouse-Signale dienen dazu, die dabei entstehende Fehlerakkumulation regelmäßig zu begrenzen und zu korrigieren (vgl. Niehorster et al. 2017, S. 3). **Latenz** bezeichnet dagegen eine zeitliche Abweichung zwischen der tatsächlichen Bewegung eines Trackers und der entsprechenden Reaktion beziehungsweise Darstellung im System. Eine zu hohe Latenz kann dazu führen, dass die virtuelle Umgebung während einer Bewegung räumlich instabil erscheint (vgl. Niehorster et al. 2017, S. 18f.). Eine weitere Fehlerart ist der **Trackingverlust**. Wird die Sichtverbindung zu den Lighthouse-Basisstationen unterbrochen, können nach der Wiederaufnahme des Trackings deutliche Versätze der ausgegebenen Position und Orientierung auftreten. In den Untersuchungen von Niehorster et al. waren diese Versätze besonders groß, wenn der Tracker während des Verlusts und der Wiederaufnahme des Trackings bewegt wurde (2017, S. 14f.). Für das in dieser Arbeit entwickelte Stabilisierungsverfahren ist vor allem die Unterscheidung zwischen kurzfristigem Jitter und einer längerfristigen Drift relevant.

4.2 Vive-Mars-Kamera-Tracking-System

4.2.1 Trackingfehler des Vive-Mars-Kamera-Tracking-Systems

Wie in Kapitel 2.2 beschrieben, basiert das VIVE-Mars-Kamera-Tracking-System auf SteamVR Tracking 2.0. Kuhlmann de Canaviri et al. (vgl. 2023, S. 9f) ermittelten für SteamVR Tracking 2.0 bei statischen Messungen Translationsfehler im Submillimeterbereich und Rotationsfehler im Subgradbereich. Dies galt sowohl für den Aufbau mit zwei als auch für den Aufbau mit vier Basisstationen, wobei die Fehler und deren Streuung im Vier-Stationen-Aufbau insgesamt geringer ausfielen. Für die dynamischen Messungen bewegte ein Roboterarm den Tracker mit drei Geschwindigkeiten von 12,5, 25 und 50 mm/s entlang einer Spiralbahn (vgl. Kuhlmann de Canaviri et al. 2023, S. 7f). Mit zunehmender Geschwindigkeit nahm die Genauigkeit in beiden Aufbauten ab. Nur bei der niedrigsten Geschwindigkeit erreichte der Vier-Stationen-Aufbau im Mittel noch eine Genauigkeit im Submillimeterbereich von 0,57 mm. Bei mittlerer und hoher Geschwindigkeit stiegen die mittleren Fehler auf 2,16 beziehungsweise 4,24 mm. Im Zwei-

Stationen-Aufbau betrugen sie bereits bei der niedrigsten Geschwindigkeit 2,82 mm und stiegen bei der höchsten Geschwindigkeit auf 11,09 mm (vgl. Kuhlmann de Canaviri et al. 2023, S. 10). Aus den Untersuchungsergebnissen lassen sich mögliche Fehlerquellen und geeignete Gegenmaßnahmen ableiten. Zugleich ergeben sich daraus Testszenarien, mit denen vergleichbare Trackingfehler gezielt hervorgerufen und das in diesem Kapitel entwickelte Programm unter kontrollierten Bedingungen überprüft werden können.

Erstens zeigten Niehorster et al. (2017, S. 14f) für das ursprüngliche HTC-Vive-Trackingsystem, dass ein Verlust und die anschließende Wiederherstellung des Trackings während einer Bewegung zu großen sprunghaften Versätzen der erfassten Position und Orientierung führen können. Da diese Untersuchung nicht mit SteamVR Tracking 2.0 durchgeführt wurde, lässt sie sich nicht unmittelbar auf das VIVE-Mars-Kamera-Tracking-System übertragen. Sie beschreibt jedoch ein mögliches Fehlerszenario, das auch bei der praktischen Prüfung des Programms berücksichtigt werden sollte. Die Sichtverbindungen zwischen Tracker und Basisstationen sollten deshalb während der Produktion möglichst durchgehend freigehalten werden.

Zweitens stellten Sansone et al. (2022, S. 910) bei SteamVR Tracking 1.0 Sichtverluste und Trackingartefakte an Messpositionen nahe der vertikalen Sichtfeldgrenze der Basisstationen fest. Außerdem waren die Messungen im zentralen Bereich des Trackingvolumens genauer als in dessen Randbereichen, während entlang der Höhenachse größere Abweichungen auftraten. Die Untersuchung zeigte zudem eine vom System geneigt erfasste Bodenreferenzebene (vgl. Sansone et al. 2022, S. 909, 912). An den Grenzen des Sichtfeldes führten eine eingeschränkte Sichtbarkeit und die äußere Lage der Messpunkte zu einer höheren Variabilität und einem Genauigkeitsverlust (vgl. Sansone et al. 2022, S. 913). Ob diese für SteamVR Tracking 1.0 beobachteten Effekte in vergleichbarer Form auch bei SteamVR Tracking 2.0 auftreten, wurde von Sansone et al. nicht untersucht. Da beide Versionen auf dem optischen Lighthouse-Funktionsprinzip beruhen, erscheinen ähnliche Effekte jedoch grundsätzlich möglich. Die Kamera sollte daher möglichst zentral im gemeinsamen Trackingbereich der Basisstationen positioniert und die Bodenreferenzebene sorgfältig kalibriert werden, um das Risiko entsprechender Trackingabweichungen zu reduzieren.

Drittens verschlechterte sich in den dynamischen Versuchen von Kuhlmann de Canaviri et al. (2023, S. 10) die Trackinggenauigkeit mit zunehmender Bewegungsgeschwindigkeit. Dieser Zusammenhang sollte bei der Bewertung beobachteter Trackingfehler

berücksichtigt und durch Tests mit unterschiedlichen Bewegungsgeschwindigkeiten überprüft werden.

Viertens erreichte der Aufbau mit vier Basisstationen insgesamt eine höhere statische und dynamische Genauigkeit als der Aufbau mit zwei Basisstationen. Die Verdeckungsversuche wurden allerdings ausschließlich mit vier Basisstationen durchgeführt. Beim Verdecken jeweils einer Basisstation blieben die statischen Translationsfehler überwiegend unter einem Millimeter, lagen jedoch über den Ergebnissen der Messungen ohne Verdeckung (vgl. Kuhlmann de Canaviri et al. 2023, S. 12f). Ein direkter Vergleich der Verdeckungsrobustheit von Zwei- und Vier-Stationen-Aufbauten wurde somit nicht vorgenommen. Die Autoren nehmen dennoch an, dass vier Basisstationen für Anwendungsfälle mit möglichen Verdeckungen besser geeignet sind (vgl. Kuhlmann de Canaviri et al. 2023, S. 14). Sofern der Produktionsraum dies zulässt, können daher vier statt zwei Basisstationen eingesetzt werden, um die Trackinggenauigkeit und die Redundanz gegenüber der Verdeckung einer einzelnen Basisstation zu erhöhen. Dies ist mit den Materialien des Videolabors möglich und entspricht bereits dem Aufbau im MPP. Zur Maximierung der Abdeckung des Produktionsbereichs empfiehlt der Hersteller außerdem, die Basisstationen mindestens zwei Meter hoch und mindestens fünf Meter voneinander entfernt zu montieren sowie um 30 bis 45 Grad nach unten zu neigen (vgl. HTC VIVE 2022, S. 19).

4.2.2 Filtersystem des Vive Mars

Die Software der VIVE Mars verfügt bereits über einen integrierten Filter für die Trackingdaten, der über die Einstellungen aktiviert werden kann. Im Benutzerhandbuch des Herstellers finden sich dazu jedoch keinerlei spezifische Angaben (vgl. HTC VIVE, 2022). Welcher Algorithmus dem Filter zugrunde liegt und welche Parameter dabei angepasst werden, bleibt somit offen. Eigenen Beobachtungen der Einstellungsoberfläche zufolge lässt sich der Filter lediglich ein- und ausschalten sowie in fünf Filterstufen abstufen (low bis high), ohne dass die Unterschiede zwischen diesen Stufen näher spezifiziert werden.

4.3 Ziel der Trackingdatenstabilisierung

Ziel der folgenden Entwicklungsabschnitte ist die Entwicklung eines Programms, das aus dem Trackingdatenstrom Rauschen herausfiltert und Drift soweit wie möglich unterbindet, um die im virtuellen Hintergrund aufgetretenen Trackingfehler während des MPP zu unterbinden. Die Anwendung soll dabei livefähig arbeiten und direkt auf dem eingehenden Trackingdatenstrom operieren, statt auf zuvor aufgezeichnetem Material zu basieren. Da die Filterung unmittelbar in den laufenden Produktionsbetrieb eingreift, soll dabei möglichst keine beziehungsweise nur eine minimale zusätzliche Latenz entstehen

und tatsächliche Kamerabewegungen möglichst wenig verfälscht werden. Wie bereits bei der Entwicklung der Latenzkompensation orientiert sich auch dieses Verfahren an den zu Beginn der Arbeit definierten Bewertungskriterien, sodass sich die entstehende Softwarelösung anhand dieser einheitlich evaluieren lässt.

4.4 Erste Entwicklungsphase der Trackingdatenstabilisierung

4.4.1 Ziel

Ziel der ersten Entwicklungsphase ist es, einen für die vorliegende Problemstellung geeigneten Filter zu finden, mit dem sich Rauschen und soweit es geht auch Drift wirksam reduzieren lassen. Im Mittelpunkt steht dabei zunächst die Auswahl und Erprobung mehrerer grundsätzlich infrage kommender Filterverfahren, deren Wirksamkeit anschließend anhand geeigneter Tests belegt werden soll. Auf Grundlage dieser Tests soll am Ende der Phase eine Entscheidung für ein konkretes Filterverfahren getroffen sowie dessen Koeffizienten näherungsweise festgelegt werden, sodass eine erste funktionsfähige Grundlage für die in den folgenden Phasen vorgesehene Verfeinerung entsteht. Zur Erprobung der Filter werden zunächst erneut die bereits im Rahmen der ersten Phase der Latenzberechnung aufgezeichneten Testvideos herangezogen, die sowohl Phasen des Stillstands als auch starke Kamerabewegungen abbilden und damit eine Einschätzung der Filterwirkung unter beiden Bedingungen ermöglichen. Ergänzend wird ein Videoausschnitt aus dem MPP-69 herangezogen, der einen isolierten, ruhenden Hintergrund bei stillstehender Kamera zeigt, an dem jedoch das bereits beschriebene Ruckeln deutlich sichtbar ist. Dieser reale Produktionsclip erlaubt es, die Wirksamkeit der getesteten Filter nicht nur an den kontrolliert aufgezeichneten Testdaten, sondern auch am tatsächlich beobachteten Fehlerbild zu überprüfen.

4.4.2 Vorauswahl der Filter

Da die Trackingdaten während des Betriebs fortlaufend verarbeitet werden sollen, werden für die Untersuchung ausschließlich kausal implementierbare Filter berücksichtigt. Ein zeitdiskretes System ist kausal, wenn sein aktueller Ausgabewert nur von gegenwärtigen und vergangenen, nicht aber von zukünftigen Eingabewerten abhängt (vgl. Oppenheim & Schafer 2010, S. 22). Verfahren, die zukünftige Abtastwerte benötigen, beispielsweise eine zentrierte Glättung oder eine Vorwärts-Rückwärts-Filterung, lassen sich nur mithilfe einer zusätzlichen Pufferung auf einen laufenden Datenstrom anwenden. Die dadurch entstehende Verzögerung widerspricht den Anforderungen des hier betrachteten latenzkritischen Trackingprozesses.

Auf dieser Grundlage werden drei kausal implementierbare Filter für den Vergleich ausgewählt: der exponentielle gleitende Mittelwert, ein Butterworth-Tiefpass zweiter Ordnung und der One-Euro-Filter:

Der **exponentielle gleitende Mittelwert (EMA)** ist ein rekursiver Tiefpass erster Ordnung. Sein Ausgabewert ergibt sich aus dem aktuellen Messwert und dem vorherigen Filterwert:

$$y_n = \alpha \cdot x_n + (1-\alpha) \cdot y_{n-1}$$

Mit abnehmendem α wird das Eingangssignal stärker geglättet. Gleichzeitig reagiert der Filter langsamer auf Änderungen, sodass die Unterdrückung von Jitter mit einem zunehmenden Lag verbunden ist (vgl. Casiez et al., 2012, S. 2528). Für die Berechnung muss lediglich der vorherige Ausgabewert gespeichert werden. Rekursive Tiefpässe erster Ordnung benötigen daher nur einen geringen Rechen- und Speicheraufwand, besitzen allerdings eine begrenzte Fähigkeit zur Trennung benachbarter Frequenzbereiche (vgl. Smith, 1999, S. 322ff.).

Der **Butterworth-Tiefpass zweiter Ordnung** gehört zu den IIR-Filtern. Butterworth-Filter zeichnen sich durch einen maximal flachen Betragsfrequenzgang im Durchlassbereich aus. Mit zunehmender Filterordnung wird der Übergang zwischen Durchlass- und Sperrbereich steiler, sodass ein Filter zweiter Ordnung Frequenzanteile oberhalb der Grenzfrequenz stärker dämpfen kann als ein Tiefpass erster Ordnung (vgl. Oppenheim & Schafer 2010, S. 513). Diese Eigenschaft ermöglicht eine deutlichere Trennung zwischen langsamen Bewegungsanteilen und höherfrequenten Störungen, sofern sich deren Frequenzbereiche hinreichend unterscheiden. Als IIR-Filter berechnet der Butterworth-Filter seinen aktuellen Ausgabewert aus dem aktuellen Eingangswert sowie einer begrenzten Anzahl vergangener Ein- und Ausgangswerte und kann daher kausal implementiert werden. Seine Phasenverschiebung kann jedoch schnelle Signaländerungen zeitlich verzögern und muss bei einer Echtzeitanwendung berücksichtigt werden (vgl. Türkler & Akkan 2025, S. 149).

Der **One-Euro-Filter** wurde für verrauschte Eingabedaten interaktiver Systeme entwickelt und soll den Zielkonflikt zwischen Jitter-Unterdrückung und geringer Verzögerung berücksichtigen (vgl. Casiez et al. 2012, S. 2527). Dazu wird die Grenzfrequenz des Tiefpasses in Abhängigkeit von einer geglätteten Schätzung der Signalgeschwindigkeit angepasst:

$$f_c = f_{c,min} + \beta|x|.$$

Bei geringen Signalgeschwindigkeiten liegt die Grenzfrequenz nahe $f_{c,min}$, wodurch eine stärkere Glättung erreicht wird. Mit zunehmender Geschwindigkeit erhöht der Parameter β die Grenzfrequenz, sodass der Filter schneller auf Bewegungen reagieren kann (vgl. Casiez et al. 2012, S. 2529). Die beiden wesentlichen einstellbaren Parameter sind somit die minimale Grenzfrequenz $f_{c,min}$ und der Geschwindigkeitsfaktor β. Die Berechnung verwendet lediglich einfache arithmetische Operationen und einen konstanten internen Zustand, sodass der Filter nur einen geringen Rechen- und Speicheraufwand benötigt (vgl. Casiez et al. 2012, S. 2527, 2529f.).

Alle drei Verfahren können ohne zukünftige Eingangswerte und mit einem begrenzten, konstanten Zustand pro Abtastwert ausgeführt werden. Sie eignen sich daher grundsätzlich für eine fortlaufende Verarbeitung der Trackingdaten. Dennoch verursachen alle Filter eine von ihrer Auslegung und Parametrierung abhängige Verzögerung. Der EMA stellt die einfachste Lösung dar, besitzt jedoch nur eine geringe Frequenzselektivität. Der Butterworth-Filter ermöglicht eine steilere Dämpfung oberhalb der Grenzfrequenz, erfordert aber eine sorgfältige Bestimmung seiner Koeffizienten und eine Berücksichtigung seiner Phasenverschiebung. Der One-Euro-Filter passt seine Glättungswirkung dagegen an die geschätzte Bewegungsgeschwindigkeit an. Ein vergleichender Test ist deshalb erforderlich, um das für die Trackingdaten dieser Arbeit günstigste Verhältnis zwischen Rauschunterdrückung und Bewegungsverzögerung zu bestimmen.

4.4.3 Versuchsaufbau zur Auswahl der Filter

Die ausgewählten drei kausalen Filter (der exponentielle gleitende Mittelwert (EMA), der Butterworth-Tiefpass 2. Ordnung und der One-Euro-Filter) werden implementiert und miteinander verglichen. Dazu wird auf die bereits im Rahmen der ersten Phase der Latenzberechnung aufgezeichneten Testvideos zurückgegriffen (vgl. Kapitel 3.4.1), die zu Testzwecken in der UE abgespielt werden, sodass die Filterwirkung unter produktionsnahen Bedingungen untersucht werden kann, ohne einen vollständigen Live-Trackingaufbau zu benötigen. Es kommen drei Testszenarien zum Einsatz: Stillstand, bei dem ein verbleibender Positionsversatz ausschließlich auf Rauschen und Drift zurückzuführen ist, ein Videoausschnitt aus dem MPP-69 mit isoliertem, ruhendem Hintergrund und wahrnehmbarem Drift, der eine Überprüfung am real beobachteten Fehlerbild ermöglicht, sowie Bewegung in Form einer kreisenden Kamerabewegung, um die Filterwirkung auch unter dynamischen Bedingungen zu beurteilen. Die Unterscheidung zwischen Stillstand und Bewegung stellt sicher, dass jeder der drei Filter sowohl im Ruhezustand als auch während einer tatsächlichen Kamerafahrt bewertet wird, passend zum in Kapitel 4.3 formulierten Ziel, dass die eigentliche Bewegung nicht verfälscht werden darf. Um

die Positionsverläufe der Videos auswerten zu können, werden diese mit dem bereits im Rahmen der Latenzbestimmung entwickelten, OpenCV-basierten Trackingverfahren (vgl. Kapitel 3.4.2) entlang der x- und y-Bildachse getrackt und die Trackingdaten anschließend mit den drei Filtern in einer niedrigen, einer mittleren und einer starken Filterung unterzogen. So liegt für jedes Testvideo ein aufgelöster Positionsverlauf vor gefiltert mit den exponentiell gleitenden Mittelwert (EMA), dem Butterworth-Tiefpass 2. Ordnung und dem One-Euro-Filter in jeweils drei Filterstärken, was die Grundlage für die folgenden Bewertungsgrößen bildet. Die verschiedenen Filterparameter sind in Tabelle 4 zu sehen.

Filter	**Niedrige Filterung**	**Mittlere Filterung**	**Hohe Filterung**
Butterworth-Filter (2. Ordnung)	Grenzfrequenz: 5 Hz	Grenzfrequenz: 1 Hz	Grenzfrequenz: 0,3 Hz
EMA-Filter (1. Ordnung)	Grenzfrequenz: 5 Hz	Grenzfrequenz: 1 Hz	Grenzfrequenz: 0,3 Hz
One-Euro-Filter (d_cutoff: 1,0 Hz β: 0,01)	min_cutoff: 2,0 Hz	min_cutoff: 0,5 Hz	min_cutoff: 0,15 Hz

Tabelle 4: Parameter der im Filtertest untersuchten Filterstärken

Jeder der drei Filter soll zwei teils gegenläufige Anforderungen erfüllen: Er soll insbesondere Rauschen und Drift möglichst vollständig entfernen, sodass z. B. die im MPP beobachteten Mikroruckler unterbunden werden, dabei aber keine allzu große zusätzliche Verzögerung verursachen. Zur objektiven Bewertung werden vier Kennwerte herangezogen: Erstens die Standardabweichung der Positionswerte während der Stillstandsszenarien als Maß für die Rauschreduktion. Zweitens der Drift als Abweichung zwischen Anfangs- und Endposition der jeweiligen Aufnahme, insbesondere des MPP-69-Clips. Drittens die Driftgeschwindigkeit als diese Abweichung bezogen auf die verstrichene Zeit, wodurch auch unterschiedlich lange Aufnahmen vergleichbar werden. Viertens der Delaywert, ermittelt mit dem bereits in der ersten Entwicklungsphase der Latenzbestimmung entwickelten, korrelationsbasierten Verfahren (vgl. Kapitel 3.4). Zusätzlich soll auch überprüft werden, wie stark die jeweiligen Filter die kreisende Kamerabewegung aus dem dritten Video verfälscht.

Die Überprüfung erfolgt sowohl anhand der errechneten Kennwerte als auch durch eine optische Kontrolle des gefilterten Videomaterials. Verglichen wird, welcher der drei Filter anhand der Standardabweichung das Rauschen am wirksamsten reduziert, welcher anhand der Driftabweichung dem Drift am wirksamsten entgegenwirkt und welcher dabei den geringsten Delay verursacht, wobei diese drei Kennwerte gegeneinander abgewogen werden müssen, da stärkere Glättung meist auch größeren Delay bedeutet. Unabhängig davon wird geprüft, welcher Filter die eigentliche Kamerabewegung am wenigsten verfälscht. Da sich EMA, Butterworth-Tiefpass und One-Euro-Filter bereits konzeptionell im Verhältnis von Rauschunterdrückung zu Latenz unterscheiden, zeigt dieser Vergleich, ob sich die dort formulierten Erwartungen auch empirisch bestätigen. Erst im Zusammenspiel aller vier Kriterien lässt sich am Ende der Testphase eine begründete Entscheidung für das weiter verwendete Filterverfahren treffen.

4.4.4 Ergebnisse des Filtertests

Die drei ausgewählten Filter wurden entsprechend dem beschriebenen Versuchsaufbau in jeweils drei Filterstärken auf die ermittelten Positionsverläufe angewendet. Für die Auswertung wurden die Standardabweichung σ, die Positionsabweichung zwischen Anfang und Ende D, die daraus resultierende Driftgeschwindigkeit D/T sowie die durch die Filterung hervorgerufene zeitliche Verschiebung betrachtet. Die Ergebnisse zeigen deutliche Unterschiede sowohl zwischen den Filterstärken als auch zwischen den drei Filterverfahren.

Im selbst aufgenommenen Stillstandsszenario beträgt die Standardabweichung des ungefilterten Bewegungsbetrags σ_r = 0,74 px. Mit zunehmender Filterstärke nimmt dieser Wert bei allen drei Filtern ab. Bei der stärksten untersuchten Filterung erreicht der Butterworth-Filter 0,4 px, der EMA 0,43 px und der One-Euro-Filter 0,28 px. Dies entspricht gegenüber dem ungefilterten Signal einer Reduktion um etwa 45,8 %, 41,8 % beziehungsweise 61,8 %. Der One-Euro-Filter erreicht damit in diesem Szenario die stärkste Reduktion der Positionsschwankungen. Bei der Positionsabweichung zwischen Anfang und Ende zeigt sich dagegen eine deutlich geringere Wirkung. Ungefiltert beträgt Dr = 7,55 px. Bei starker Filterung werden für Butterworth 6,89 px, für EMA 6,7 px und für One-Euro 6,45 px gemessen. Selbst beim One-Euro-Filter entspricht dies lediglich einer Reduktion von etwa 14,5 %. Auffällig ist zudem die unterschiedliche Ausprägung in den beiden Bildachsen. Während der ungefilterte Positionsversatz in x-Richtung lediglich 1,24 px beträgt, werden in y-Richtung 8,04 px erreicht. Auch nach starker One-Euro-Filterung verbleiben in y-Richtung 6,95 px. Die Positionsverschiebung findet somit überwiegend entlang der y-Achse statt.

Auch beim zweiten Testszenario dem Stillstand aus dem isolierten Hintergrund des MPPs reduzieren alle drei Filter mit zunehmender Filterstärke die Standardabweichung. Ausgehend von σ_r= 3,52 px werden bei starker Filterung 2,74 px mit Butterworth, 2,98 px mit EMA und 2,77 px mit One-Euro erreicht. Gleichzeitig steigt die durch die Filterung hervorgerufene zeitliche Verschiebung. Beim Butterworth-Filter erhöht sich der Betrag des Delays von 16,7 ms bei niedriger auf 79,2 ms bei mittlerer und 238,1 ms bei starker Filterung. Beim EMA werden 5,6 ms, 39,0 ms und 94,8 ms gemessen. Der One-Euro-Filter erreicht 16,7 ms, 55,7 ms und 121,5 ms.

Besonders deutlich wird der Einfluss der Filterung bei der kreisenden Kamerabewegung. Bei niedriger Filterstärke folgen alle Filter der ursprünglichen Bewegung noch weitgehend. Für Butterworth und EMA wird eine Verzögerung von jeweils etwa 13 ms gemessen. Bei zunehmender Filterstärke unterscheiden sich die Verfahren jedoch deutlich. Der Butterworth-Filter erreicht bei mittlerer Filterung bereits eine Verzögerung von 200,5 ms und bei starker Filterung 617,6 ms.

Beim EMA fällt die Beeinflussung geringer aus. Bei mittlerer Filterung beträgt der Delay 120,2 ms, während bei starker Filterung 254,6 ms erreicht werden. Auch hier zeigt sich somit eine zunehmende zeitliche und strukturelle Veränderung der ursprünglichen Bewegung mit steigender Filterstärke.

Der One-Euro-Filter zeigt bei der kreisenden Kamerabewegung ein abweichendes Verhalten. Für alle drei untersuchten Filterstärken wird mit der zeitlichen Auflösung des verwendeten Auswertungsverfahrens keine zeitliche Verschiebung festgestellt. Auch die Standardabweichung des Bewegungsbetrags verändert sich nur geringfügig von ungefiltert 268,37 px auf maximal 271,64 px.

Zusammenfassend zeigen die Messungen, dass eine Erhöhung der Filterstärke bei allen Verfahren die Positionsschwankungen in den ruhigen Szenarien reduzieren kann. Gleichzeitig nimmt insbesondere bei den statischen Filterverfahren Butterworth und EMA die zeitliche Beeinflussung tatsächlicher Kamerabewegungen zu. Der One-Euro-Filter weist in den untersuchten Szenarien insgesamt das günstigste Verhältnis zwischen der Reduktion kurzfristiger Schwankungen und der Erhaltung dynamischer Bewegungen auf. Ein verbleibender langsamer Positionsversatz wird jedoch auch durch diesen Filter nicht vollständig entfernt.

Filter	σr [px]	Reduktion σr	Dr [px]	Reduktion Dr	Delay [ms]
Ungefiltert	0,7383	–	7,5457	–	0,0

Butterworth	0,4003	45,8 %	6,8915	8,7 %	142,7
EMA	0,4297	41,8 %	6,7007	11,2 %	50,6
One-Euro	0,2819	61,8 %	6,4481	14,5 %	37,2

Tabelle 5: Vergleich der Filterwirkung des Stillstands bei starker Filterung

Filter	Filterstärke	Delay [ms]	σr [px]
Butterworth	niedrig	16,7	3,4820
Butterworth	mittel	79,2	3,2833
Butterworth	hoch	238,1	2,7400
EMA	niedrig	5,6	3,4842
EMA	mittel	39,0	3,3285
EMA	hoch	94,8	2,9754
One-Euro	niedrig	16,7	3,4278
One-Euro	mittel	55,7	3,1907
One-Euro	hoch	121,5	2,7710

Tabelle 6: Delay im Testszenario MPP69_Stillstand

4.4.5 Analyse der Filter und Trackingdaten

Die Ergebnisse zeigen, dass alle drei untersuchten Filter die Schwankungen der Trackingdaten reduzieren können. Gleichzeitig besteht jedoch ein Zielkonflikt zwischen einer starken Glättung im Stillstand und einer möglichst unverfälschten Übertragung tatsächlicher Kamerabewegungen. Besonders deutlich wird dies beim Butterworth-Tiefpass. Bei starker Filterung sinkt die Standardabweichung des Bewegungsbetrags im Stillstand von 0,74 px auf 0,4 px. Bei der kreisenden Kamerabewegung führt dieselbe Filterstärke jedoch zu einem Delay von 617,6 ms. Auch der EMA zeigt diesen Zusammenhang, wenn auch schwächer. Bei starker Filterung werden 254,6 ms Delay erreicht. Die starke Verzögerung und Glättung der tatsächlichen Bewegung ist auch im Verlauf der kreisenden Kamerabewegung deutlich erkennbar. Insbesondere der Butterworth-Filter folgt schnellen Richtungswechseln nur verzögert, während der EMA näher am ursprünglichen Verlauf liegt.

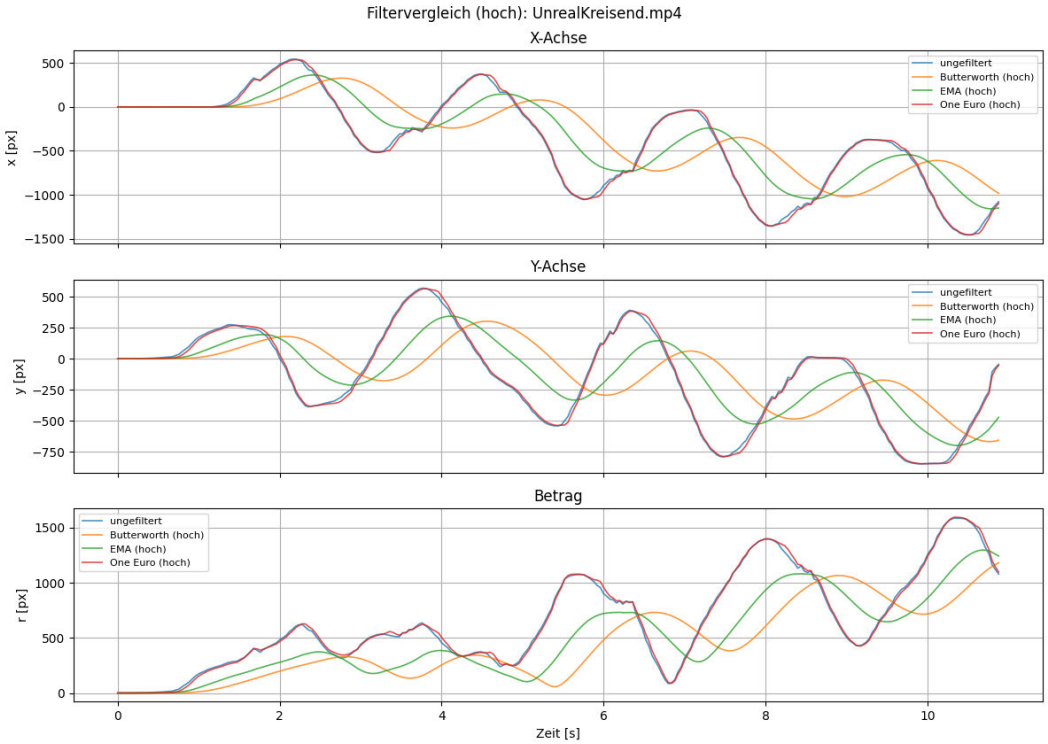


Abbildung 19: Vergleich der Geschwindigkeitskurven des Testvideos mit kreisender Kamerabewegung und starker Filterung

Der One-Euro-Filter zeigt demgegenüber das günstigste Verhalten. Im Stillstand reduziert er bei starker Filterung die Standardabweichung auf 0,28 px und damit um etwa 61,8 %. Gleichzeitig folgt sein Verlauf bei der kreisenden Kamerabewegung nahezu dem ungefilterten Signal. Für dieses Szenario wird keine messbare zeitliche Verschiebung festgestellt. Im MPP69-Stillstandsszenario tritt mit 121,5 ms bei starker Filterung allerdings ebenfalls eine Verzögerung auf. Der One-Euro-Filter ist somit nicht grundsätzlich verzögerungsfrei, passt seine Filterwirkung jedoch besser an die vorhandene Bewegung an als die beiden statischen Filter.

Die Betrachtung des Stillstandsszenarios zeigt gleichzeitig die Grenze einer reinen Filterung. Zwar werden die kurzfristigen Schwankungen deutlich geglättet, der langsame Positionsversatz bleibt jedoch erhalten. Dies ist insbesondere entlang der y-Achse sichtbar: Der ungefilterte Verlauf wandert während der fünfsekündigen Aufnahme kontinuierlich um mehrere Pixel und auch die gefilterten Verläufe folgen langfristig dieser Verschiebung. Die Filter reduzieren somit die schnellen Schwankungen um den Positionsverlauf, können dessen langsames Wandern jedoch nicht verhindern.

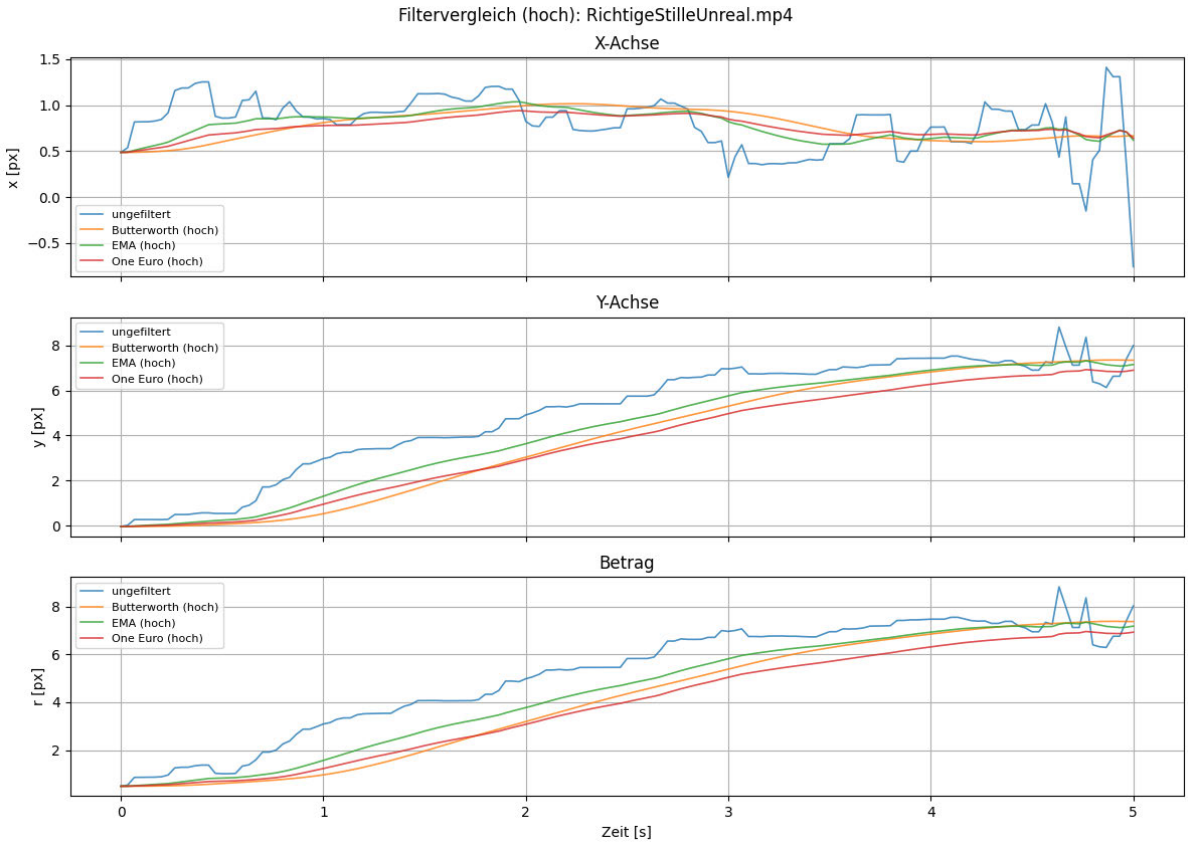


Abbildung 20: Vergleich der Geschwindigkeitskurven eines Testvideos mit stiller Kamera mit starker Filterung

Dies bestätigt sich auch quantitativ. Während der One-Euro-Filter die Standardabweichung des Bewegungsbetrags um etwa 61,8 % reduziert, sinkt der Positionsversatz lediglich von 7,55 px auf 6,45 px, was einer Reduktion von nur etwa 14,5 % entspricht. Eine weitere Erhöhung der Filterwirkung stellt keine geeignete Lösung dar, da dadurch die Gefahr einer zunehmenden Verzögerung und Verfälschung tatsächlicher Kamerabewegungen entsteht.

Von den untersuchten Verfahren bietet der One-Euro-Filter daher den geeignetsten Kompromiss zwischen Stabilisierung und Erhaltung der Kamerabewegung und wird für die weitere Entwicklung verwendet. Die Ergebnisse zeigen jedoch zugleich, dass eine

reine Filterung nicht ausreicht, um den beobachteten Drift zuverlässig zu unterdrücken. Für die weitere Stabilisierung muss deshalb zusätzlich erkannt werden, ob sich die Kamera tatsächlich bewegt oder stillsteht. Auf Grundlage einer Unterscheidung zwischen still und bewegt kann im Stillstand gezielt stärker in die Trackingdaten eingegriffen werden, ohne reale Kamerabewegungen gleichermaßen zu beeinflussen.

4.4.6 Ergebnisse

Aus der ersten Entwicklungsphase lässt sich ableiten, dass eine reine Filterung der Trackingdaten die Anforderungen an die Stabilisierung nicht vollständig erfüllen kann. Insbesondere die Unterscheidung zwischen unerwünschtem Drift und einer tatsächlichen langsamen Kamerabewegung ist allein anhand der Trackingdaten problematisch, da beide als Positionsänderung auftreten können. Für die weitere Entwicklung wird daher die klare Abgrenzung der beiden Kamerazustände Stillstand und Bewegung in den Mittelpunkt gestellt. Dadurch soll ermöglicht werden, die Trackingdaten abhängig vom erkannten Zustand unterschiedlich zu behandeln. Die folgende Entwicklungsphase beschäftigt sich entsprechend mit der Umsetzung und Untersuchung einer geeigneten Stillstanderkennung.

4.5 Zweite Entwicklungsphase der Trackingdatenstabilisierung

4.5.1 Ziel und Vorgehen

Die in der ersten Entwicklungsphase durchgeführten Filtertests zeigen, dass eine reine Glättung der Trackingdaten nicht ausreicht, um den auftretenden Drift zuverlässig zu unterdrücken. Daher wird in der zweiten Entwicklungsphase zusätzlich eine Stillstandserkennung entwickelt. Ziel des Programms ist es, die Trackingdaten der Kamera kontinuierlich einzulesen und daraus den aktuellen Bewegungszustand zu bestimmen. Dabei soll zuverlässig zwischen Stillstand und Bewegung unterschieden werden. Die Erkennung erfolgt ausschließlich anhand der eingehenden Trackingdaten und muss diese während des laufenden Betriebs in Echtzeit verarbeiten können. Da die Stillstandserkennung außerhalb der UE entwickelt wird, greift das Programm unmittelbar auf den FreeD-Datenstrom zu. Live Link ist kein eigenständiges Trackingprotokoll, sondern eine in die UE integrierte Schnittstelle zur Übertragung und Verarbeitung externer Animations- und Trackingdaten (vgl. Epic Games o. J.). Die über FreeD übertragenen UDP-Pakete können dagegen von einer eigenständigen Anwendung direkt empfangen und ausgewertet werden. Für die Entwicklung und erste Erprobung unabhängig vom VIVE-Mars-Studioaufbau wird ein iPhone vorübergehend als FreeD-Sender eingesetzt. Die App *FREED Tracker* ermittelt die Position und Orientierung des Geräts und überträgt diese

in Echtzeit als FreeD-Daten über UDP an einen frei konfigurierbaren Empfänger im lokalen Netzwerk (vgl. Ryzhak, 2026).

4.5.2 FreeD

Im beschriebenen Produktionsaufbau werden die Trackingdaten des VIVE-Mars-Systems über Live Link an die virtuelle Kamera in der Unreal Engine (UE) übertragen (vgl. HTC VIVE 2022, S. 36f.). Für die zusätzlich entwickelte, von der UE unabhängige Software wird dagegen die FreeD-Ausgabe des VIVE-Mars-Systems verwendet. Dafür lassen sich eine Ziel-IP-Adresse und ein UDP-Port festlegen (vgl. HTC VIVE 2022, S. 20ff.). Live Link stellt in der UE eine erweiterbare Schnittstelle für Datenströme externer Anwendungen und Aufnahmesysteme bereit. Abhängig von der Quelle können die Daten beispielsweise über ein Netzwerkprotokoll, eine Geräte-API oder den UE Message Bus übertragen werden (vgl. Epic Games o.J.-a). Die entwickelte Software empfängt die FreeD-Pakete hingegen unmittelbar über einen UDP-Socket und verarbeitet sie unabhängig von der UE. FreeD wurde ursprünglich von BBC Research and Development entwickelt (vgl. Vinten Radamec 2004, S. 4) und wird weiterhin von verschiedenen Kamera- und Trackingsystemen unterstützt (vgl. Epic Games o.J.-b). Für die Übertragung müssen insbesondere Ziel-IP-Adresse, UDP-Port und Kamera-ID passend eingestellt werden (vgl. Epic Games o.J.-b). Im Versuchsaufbau sendet das VIVE-Mars-System die Trackingdaten fortlaufend an die festgelegte Ziel-IP-Adresse und den konfigurierten Port. UDP baut keine dauerhafte Verbindung zwischen den Endsystemen auf und gewährleistet selbst weder eine zuverlässige noch eine geordnete Zustellung der Datagramme (vgl. Eggert et al. 2017, S. 3, 20f.). Der fortlaufende Versand ist daher keine Eigenschaft von UDP, sondern wird durch die FreeD-Ausgabe des VIVE-Mars-Systems gesteuert. Für die Trackingdaten wird der Nachrichtentyp D1 verwendet. Ein D1-Paket umfasst 29 Byte und enthält Nachrichtentyp, Kamera-ID, Pan, Tilt, Roll, X, Y und Z sowie Felder für Zoom, Fokus, benutzerdefinierte Daten und die abschließende Prüfsumme (vgl. Vinten Radamec 2004, S. 30). Die Prüfsumme c wird aus den 28 vorangehenden Paketbytes b_i berechnet (vgl. Vinten Radamec 2004, S. 28):

$$c = (0x40 - \Sigma_{i=0}^{27} b_i) \bmod 256$$

Die drei Byte langen Positions- und Rotationsfelder werden mit dem höchstwertigen Byte zuerst, also im Big-Endian-Format, zusammengesetzt und als vorzeichenbehaftete 24-Bit-Werte im Zweierkomplement interpretiert. Kamera-ID und Prüfsumme werden jeweils als einzelnes Byte verarbeitet. Aus den Rohwerten ergeben sich Rotation φ in Grad und Position p in Millimetern wie folgt (vgl.Vinten Radamec 2004, S. 28, 40f.):

$$\varphi\,[°] = \frac{r_\varphi}{32768}$$

$$p\,[mm] = \frac{r_p}{64}$$

Die Spezifikation definiert für Pan und Roll Werte von −180° bis +180°, für Tilt −90° bis +90° und für jede Positionsachse −131.072 mm bis annähernd +131.072 mm (vgl. Vinten Radamec, 2004, S. 40f.). Für die erneute Ausgabe werden die Verarbeitungsschritte umgekehrt: Die Werte werden mit dem jeweiligen Skalierungsfaktor multipliziert, in 24-Bit-Werte umgewandelt und in ein neues D1-Paket eingefügt; anschließend wird die Prüfsumme neu berechnet. Als ergänzende Orientierung diente eine quelloffene Python-Implementierung des Protokolls (vgl. Exscriber 2020). Bytepositionen, Skalierungen und Wertebereiche wurden anhand der FreeD-Spezifikation überprüft. Für den Versuchsaufbau wurde der in der LiveLinkFreeD-Quelle der UE voreingestellte Standardport 40000 verwendet (vgl. Epic Games o.J.-b). Derselbe Port wurde in der entwickelten Software konfiguriert und durch eine eingehende Ausnahmeregel in der Windows-Firewall freigegeben.

4.5.3 Stillstandserkennung

Zur Stillstandserkennung wertet das entwickelte Modul den beschriebenen FreeD-Datenstrom fortlaufend aus und unterscheidet dabei zwischen den Zuständen „Still“ und „Moving“. Diese Zustandsentscheidung bildet die Grundlage dafür, die Trackingdaten im Stillstand und während einer Kamerabewegung unterschiedlich zu behandeln.

Die Entscheidung erfolgt nicht anhand eines einzelnen festen Schwellenwerts, sondern mithilfe einer Hysterese mit zwei Grenzwerten. Für jedes gültige Trackingpaket werden zunächst aus der Veränderung gegenüber dem vorherigen Paket die translatorische Geschwindigkeit $v_{t,i}$ aus den Positionswerten x, y und z sowie die rotatorische Geschwindigkeit $v_{r,i}$ aus Pan, Tilt und Roll bestimmt. Da beide Größen unterschiedliche Einheiten und Größenordnungen besitzen, werden sie durch die zuvor kalibrierten RMS-Normwerte σ_t und σ_r dividiert. Anschließend werden die normierten Anteile zu einem dimensionslosen Kennwert e_i zusammengeführt, der im Folgenden als Bewegungsenergie bezeichnet wird:

$$e_i = \sqrt{\left(\frac{v_{t,i}}{\sigma_t}\right)^2 + \left(\frac{v_{r,i}}{\sigma_r}\right)^2}.$$

Der Begriff Bewegungsenergie bezeichnet hier somit keine physikalische Energie, sondern einen für diese Arbeit definierten, normierten Bewegungskennwert. Um kurzzeitige

Ausschläge zu dämpfen, wird über höchstens 15 aufeinanderfolgende Werte ein kausaler gleitender Effektivwert gebildet. Mit $M_i = \min(15, i)$ ergibt sich:

$$E_i = \sqrt{\frac{1}{M_i} \sum_{k=i-M_i+1}^{i} e_k^2}.$$

Die Zustandsentscheidung verwendet den geglätteten Kennwert E_i. Ein einzelner Schwellenwert könnte dazu führen, dass das System bei Werten nahe der Entscheidungsgrenze wiederholt zwischen „Still“ und „Moving“ wechselt. Ein verwandtes Problem beschreibt Canny bei der Kantenerkennung: Schwankt eine verrauschte Filterantwort um einen einzelnen Schwellenwert, können zusammenhängende Kanten unterbrochen werden. Canny reduziert diesen Effekt durch eine Hysterese mit einem oberen und einem unteren Schwellenwert (vgl. Canny, 1986, S. 689–690). Dieses Grundprinzip dient hier lediglich als konzeptionelle Orientierung. Anders als bei Canny werden die Schwellenwerte im vorliegenden Verfahren nicht zur räumlichen Verfolgung von Kanten, sondern zum zeitlichen Wechsel zwischen zwei Bewegungszuständen eingesetzt.

Aus dem Zustand „Moving“ wird erst dann in „Still“ gewechselt, wenn E_i mindestens 0,3 Sekunden lang durchgehend unter dem unteren Schwellenwert von 2,2 liegt. Umgekehrt erfolgt der Wechsel von „Still“ zu „Moving“, wenn E_i mindestens 0,1 Sekunden lang durchgehend über dem oberen Schwellenwert von 3,0 liegt. Die längere Bestätigungszeit beim Eintritt in „Still“ soll verhindern, dass eine kurze Bewegungspause fälschlich als Stillstand gewertet wird. Die kürzere Bestätigungszeit beim Austritt aus „Still“ dient dazu, eine beginnende Bewegung möglichst früh zu erkennen. Beide Schwellenwerte und Bestätigungszeiten wurden anhand der vorliegenden Trackingdaten in eigenen Tests empirisch bestimmt. Sie sind daher als Parameter des entwickelten Verfahrens und nicht als allgemeingültige Grenzwerte zu verstehen.

Voraussetzung für die Normierung ist eine vorgelagerte Kalibrierung des im Stillstand auftretenden Rauschniveaus. Dazu hält der Nutzer die Kamera still und startet die Messung manuell. Anschließend zeichnet das Programm fünf Sekunden lang die Positions- und Rotationswerte auf und berechnet aus jeweils zwei aufeinanderfolgenden Messwerten die translatorische und die rotatorische Geschwindigkeit. Um den Einfluss einzelner Ausreißer zu begrenzen, wird für beide Geschwindigkeitsreihen jeweils ein Grenzwert bestimmt, unterhalb dessen 95 % der gemessenen Werte liegen. Alle darüberliegenden Werte werden auf diesen Grenzwert gesetzt. Aus den so begrenzten

Geschwindigkeitswerten werden anschließend getrennt für Translation und Rotation die RMS-Normwerte berechnet.

$$\tilde{v}_{t,i} = \min\left(v_{t,i}, q_{t,0,95}\right), \qquad \tilde{v}_{r,i} = \min\left(v_{r,i}, q_{r,0,95}\right).$$

Aus den begrenzten Geschwindigkeitswerten werden getrennt für Translation und Rotation die RMS-Normwerte berechnet:

$$\sigma_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \tilde{v}_{t,i}^2}, \qquad \sigma_r = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \tilde{v}_{r,i}^2}.$$

Dabei bezeichnet N die Anzahl der während der Kalibrierung gebildeten Geschwindigkeitswerte. Die Größen σ_t und σ_r sind trotz ihrer Bezeichnung keine Standardabweichungen, sondern RMS-basierte Normwerte. Sie beschreiben die typische Größenordnung der im Stillstand auftretenden translatorischen beziehungsweise rotatorischen Geschwindigkeit und dienen anschließend als Bezugsgrößen für die Berechnung von e_i. Da sich das Rauschniveau mit dem Produktionsaufbau und den jeweiligen Trackingbedingungen verändern kann, wird die Kalibrierung zu Beginn jeder Sitzung erneut durchgeführt.

4.5.4 Erstes entwickeltes Programm zur Stillstandserkennung

Auf Grundlage des im Kapitel zuvor beschriebenen Verfahrens wurde ein erstes lauffähiges Programm entwickelt, das den FreeD-Datenstrom einliest, den Zustand zwischen „Still“ und „Moving“ bestimmt und über die Konsole ausgibt. Zu Beginn öffnet das Programm einen UDP-Socket auf Port 40000 (vgl. Kapitel 4.5.2) mit einem Timeout von 0,5 Sekunden, wobei als Verbindungsnachweis nicht bereits ein einzelnes gültiges Paket genügt. Erst wenn 50 Pakete der festgelegten Kamera-ID, Prüfsumme und ID-Kontrolle bestehen, meldet das Programm „Verbindung steht", während ungültige oder fremden Kameras zugeordnete Pakete verworfen werden, ohne den Zähler zu erhöhen. Diese Schwelle verhindert, dass ein einzelnes, zufällig korrektes Paket bereits als stabile Verbindung fehlinterpretiert wird. Erst nach einer expliziten Bestätigung per Eingabetaste beginnt anschließend die Kalibrierung, sodass sichergestellt ist, dass die Kamera bereits ruhig gehalten wird, bevor Daten aufgezeichnet werden. Zuvor wird der UDP-Empfangspuffer geleert, damit dort noch verbliebene, während der Positionierung empfangene Pakete nicht fälschlich in die Messung einfließen. Sammlung (5 s) und Berechnung von σ_t/σ_rerfolgen anschließend exakt nach dem in 4.5.3 beschriebenen Verfahren, wobei die Werte zur Dokumentation auf der Konsole ausgegeben werden.

In der laufenden Hauptschleife wird jedes Paket erneut auf Prüfsumme, Kamera-ID und Timeout geprüft, wobei anders als beim Verbindungsaufbau ein einzelner Fehler hier nur zum Überspringen des Durchlaufs führt, nicht zu erneutem Warten. Gültige Pakete werden an die Stillstandserkennung übergeben, die den Zustand „Still“ und „Moving“ liefert. Jeder Durchlauf schließt mit einem Konsolen-Log aus Zustand, Bewegungsenergie sowie Rohposition und -rotation, das während der Entwicklung zur Überprüfung der Schwellenwerte diente, bevor das Programm mit Eingabe von „q“ beendet wird und den Port für einen erneuten Start sofort wieder freigibt.

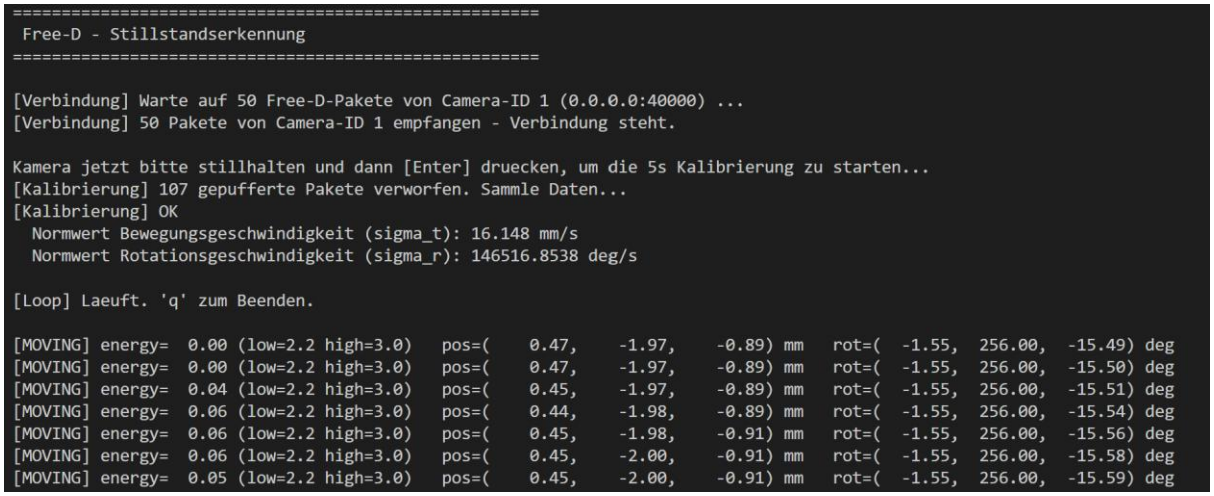


Abbildung 22: Konsolenausgabe des Stillstandserkennungs-Programms. Initialisierung

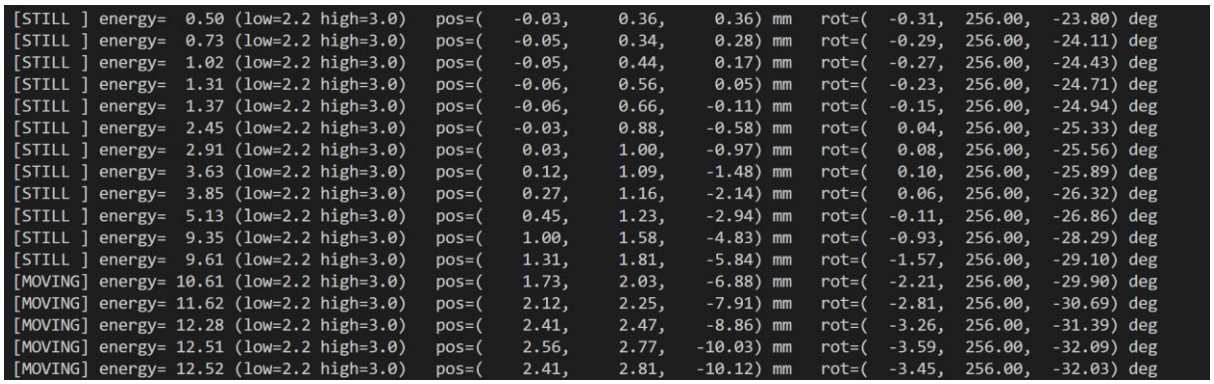


Abbildung 21: Konsolenausgabe des Stillstandserkennungs-Programms. Zustandswechsel von „Still“ zu „Moving“

4.5.5 Ergebnis

Die zweite Entwicklungsphase liefert ein lauffähiges Programm, das die formulierte Zielsetzung erfüllt: Es liest die Trackingdaten live über das FreeD-Protokoll ein und unterscheidet daraus kontinuierlich und zuverlässig zwischen den Zuständen „Still" und „Moving". Die Auswertung der geglätteten Bewegungsenergie mittels zweier Schwellwerte verhindert das in 4.5.3 beschriebene Chattering an der Entscheidungsgrenze, ohne die Erkennung spürbar zu verzögern.

4.6 Dritte Entwicklungsphase der Trackingdatenstabilisierung

4.6.1 Ziel

Das erste Ziel dieser Phase ist zunächst die Anpassung des Programms an den FreeD-Datenstrom der VIVE Mars (vgl. Kapitel 4.5.2). Darauf aufbauend sollen manipulierte FreeD-Datenpakete erstmals an die UE weitergesendet werden, sodass nicht mehr der ursprüngliche, sondern ein gefilterter Trackingdatenstrom in die Berechnung des virtuellen Kamerabildes einfließt. Da bislang nur zwischen den Zuständen „Still“ und „Moving“ unterschieden, jedoch noch keine Filterung vorgenommen wurde, soll ein Verfahren zur Filterung entwickelt werden. Damit einher geht das Ziel, den anhand der Hysterese erkannten Übergang zwischen den Zuständen „Still“ und „Moving“ flüssig zu gestalten, sodass beim Wechsel zwischen beiden Zuständen keine sichtbaren Sprünge oder Verzögerungen im virtuellen Hintergrund auftreten. Weiterhin soll der durch die Filterung entstehende Delay über beide Zustände hinweg möglichst klein gehalten werden, um eine gleichbleibende und vorhersehbare Reaktionszeit des Systems sicherzustellen. Abschließend ist die Entwicklung einer grafischen Benutzeroberfläche vorgesehen, über die sich die Anwendung im Produktionsbetrieb ohne tiefergehende technische Vorkenntnisse bedienen und konfigurieren lässt.

4.6.2 Anpassungen des Programms an den Datenstrom der Vive Mars

Im Testbetrieb mit der Vive Mars zeigte sich, dass die Netzwerk-Paketrate der ausgesendeten Free-D-Pakete (zumindest im Stillstand) deutlich höher liegt als die tatsächliche Aktualisierungsrate der Trackingdaten. Insbesondere im Stillstand enthielt daher nur ein Bruchteil der empfangenen Pakete tatsächlich neue Positions- und Rotationswerte. Ein Großteil der Pakete war bit-identisch mit dem jeweils vorherigen Paket und stellte somit keine neue Messung dar, sondern lediglich eine Wiederholung des zuletzt bekannten Zustands auf Netzwerkebene. Da die Kalibrierungsroutine zur Bestimmung des Rauschwerts (Noise Floor) ursprünglich jedes empfangene Paket als eigenständige Messung interpretierte, führte dieses Verhalten zu einer systematischen Verfälschung der berechneten Standardabweichungen σ_t und σ_r. Da ein Großteil der einbezogenen „Messungen" keine reale Bewegung zwischen aufeinanderfolgenden Samples widerspiegelte, kollabierten die berechneten Geschwindigkeitswerte v_t und v_r gegen null und die daraus abgeleiteten Rauschwerte fielen auf einen viel zu niedrigen Wert. In der Folge wurde bei der anschließenden Energieberechnung bereits durch minimales, aber reales Sensorrauschen eine massive Überschreitung der Rauschschwelle registriert, was zu einer berechneten Bewegungsenergie in den Millionenbereich führte, obwohl sich das Trackingobjekt tatsächlich im Stillstand befand.

Um dieses Problem zu beheben, wurden drei Anpassungen am Kalibrierungs- und Fehlerbehandlungsprozess vorgenommen.

In der Kalibrierungsfunktion wurde eine Duplikat-Filterung eingeführt, die bit-identische, aufeinanderfolgende Pakete vor der Geschwindigkeitsberechnung erkennt und entfernt. Innerhalb einer Sequenz wiederholter, identischer Trackingdaten wird dabei ausschließlich das jeweils letzte Sample beibehalten, da dieses den korrekten Zeitstempel für den nachfolgenden, tatsächlich neuen Messwert liefert. Dadurch fließen in die Berechnung von $v_t$und v_r nur noch Paare aufeinanderfolgender Samples ein, zwischen denen sich die Trackingdaten der Vive Mars tatsächlich verändert haben. Ohne diese Filterung wäre, wie beschrieben, ein Kollaps der berechneten Geschwindigkeiten auf nahezu null die Folge, wodurch auch σ_t und σ_r auf einen intern definierten Sicherheits-Minimalwert (Floor) abfielen und keine belastbare Grundlage für die spätere Energie-Normierung mehr darstellten. Da die Duplikat-Filterung dazu führen kann, dass bei sehr geringer effektiver Update-Rate nur wenige reale Bewegungsmessungen für die Kalibrierung zur Verfügung stehen, wurde zusätzlich die Exception-Klasse CalibrationError eingeführt. Diese wird ausgelöst, sobald nach der Duplikatfilterung weniger echte Bewegungsmessungen vorliegen als der konfigurierte Mindestwert min_motion_samples (Standardwert: 10). Vor dieser Änderung fiel die Kalibrierung in einem solchen Fall auf einen Sicherheits-Minimalwert für σ zurück, was zu einer instabilen Energie-Normierung führte. Ergänzend wurde die Dauer der initialen Kalibrierungsphase von 5 auf 15 Sekunden erhöht. Der Grund hierfür liegt ebenfalls in der geringen tatsächlichen Update-Rate der Vive Mars im Stillstand: In Testmessungen wurde beobachtet, dass real nur etwa eine neue Pose pro Sekunde übertragen wird, während die Netzwerk-Paketrate deutlich darüber liegt. Eine Kalibrierungsdauer von 5 Sekunden lieferte dementsprechend zu wenige tatsächlich neue Bewegungs-Deltas, um eine ausreichende Schätzung von σ_t und σ_r zu ermöglichen, und begünstigte damit fehlerhaft niedrige Rauschwerte. Mit einer Kalibrierungsdauer von 15 Sekunden stehen auch bei der geringen effektiven Update-Rate der Vive Mars ausreichend viele reale

Bewegungsmessungen zur Verfügung, um eine robuste Schätzung des Rauschniveaus zu gewährleisten.

```
[MOVING] energy=347043836.32 (low=2.2 high=3.0)   pos=( -375.08,  -301.23,  1262.31) mm   rot=(   2.21,    0.71,   -0.82
) deg
[MOVING] energy=282789275.70 (low=2.2 high=3.0)   pos=( -375.08,  -301.23,  1262.31) mm   rot=(   2.21,    0.71,   -0.82
) deg
[MOVING] energy=  0.00 (low=2.2 high=3.0)   pos=( -375.08,  -301.23,  1262.31) mm   rot=(   2.21,    0.71,   -0.82) deg
[MOVING] energy=  0.00 (low=2.2 high=3.0)   pos=( -375.08,  -301.23,  1262.31) mm   rot=(   2.21,    0.71,   -0.82) deg
[MOVING] energy=  0.00 (low=2.2 high=3.0)   pos=( -375.08,  -301.23,  1262.31) mm   rot=(   2.21,    0.71,   -0.82) deg
[MOVING] energy=582332054.05 (low=2.2 high=3.0)   pos=( -375.05,  -301.20,  1262.30) mm   rot=(   2.21,    0.71,   -0.82
) deg
[MOVING] energy=619729368.29 (low=2.2 high=3.0)   pos=( -375.05,  -301.19,  1262.30) mm   rot=(   2.21,    0.71,   -0.82
) deg
[MOVING] energy=651169444.67 (low=2.2 high=3.0)   pos=( -375.05,  -301.17,  1262.30) mm   rot=(   2.21,    0.71,   -0.82
) deg
```

Abbildung 24: Konsolenausgabe des Stillstandserkennungsprogramms vor den Änderungen

```
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
[STILL ] energy=  0.00 (low=2.2 high=3.0)   pos=( -374.84,  -283.97,  1220.52) mm   rot=(   1.97,    3.49,   -1.09) deg
```

Abbildung 23: Konsolenausgabe des Stillstandserkennungsprogramms nach den Änderungen

4.6.3 Senden der Trackingdaten an UE

Nachdem das entwickelte Programm anhand der eingehenden Trackingdaten zwischen den Zuständen „Still“ und „Moving“ unterschieden hat, werden die verarbeiteten Daten an die UE weitergeleitet. Wenn das Programm und die UE auf demselben Rechner ausgeführt werden, erfolgt der Versand an die Loopback-Adresse 127.0.0.1. Im Versuchsaufbau wird dafür nicht der zum Empfang der ursprünglichen VIVE-Mars-Daten verwendete Port 40000, sondern der separate Port 40002 genutzt. Die Trennung verhindert eine Überschneidung zwischen dem eingehenden und dem neu erzeugten FreeD-Datenstrom. Gegenüber der UE tritt das Programm somit als eigenständiger FreeD-Sender auf. Für den Empfang werden in der UE die Plug-ins „Live Link“ und „Live Link FreeD“ aktiviert. Anschließend wird im Live-Link-Fenster eine „LiveLinkFreeD Source“ eingerichtet und mit der verwendeten Netzwerkschnittstelle sowie dem Empfangsport konfiguriert. Epic Games nennt den UDP-Port 40000 als Standardwert; im vorliegenden Versuchsaufbau wird davon abweichend Port 40002 verwendet. Nach erfolgreichem Empfang erscheint der FreeD-Datenstrom in Live Link als Subject. Dieses wird über eine Live-Link-Controller-Component einer CineCamera zugeordnet, sodass die empfangenen Positions- und Rotationsdaten auf die virtuelle Kamera übertragen werden (vgl. Epic Games 2026a). Vor dem Versand werden die Trackingdaten abhängig vom erkannten Bewegungszustand verarbeitet. Beim Wechsel in den Zustand „Still“ wird die zu diesem Zeitpunkt aktuelle Kamerapose gespeichert und während des gesamten Stillstands unverändert ausgegeben. Dadurch werden die im Stillstand auftretenden Mikrobewegungen und langsamen Positionsänderungen nicht an die virtuelle Kamera weitergegeben. Im

Zustand „Moving“ werden dagegen fortlaufend die aktuellen Trackingdaten verwendet. Diese werden zusätzlich mit dem im folgenden Unterkapitel beschriebenen One-Euro-Filter geglättet, bevor sie an die UE übertragen werden. Für die Ausgabe werden die verarbeiteten Positions- und Rotationswerte wieder in das binäre Format eines FreeD-D1-Pakets übertragen. Dazu werden die Werte mit den vorgesehenen Skalierungsfaktoren multipliziert, als vorzeichenbehaftete 24-Bit-Werte kodiert und an den entsprechenden Bytepositionen des Pakets eingefügt. Anschließend wird die im FreeD-Protokoll vorgesehene Prüfsumme neu berechnet und als letztes Byte ergänzt (vgl. Vinten Radamec 2004, S. 28, 30; vgl. Kapitel 4.5.2). Das vollständige Paket wird schließlich über einen UDP-Socket an die konfigurierte IP-Adresse 127.0.0.1 und den Port 40002 gesendet.

4.6.4 Einbauen eines Filters

Die Filterung der Trackingdaten wird ausschließlich während erkannter Bewegungsphasen eingesetzt. Wird durch die zuvor entwickelte Stillstandserkennung der Zustand „Still“ festgestellt, werden die zuletzt übertragenen Positions- und Rotationswerte eingefroren. Eine zusätzliche Glättung der Trackingdaten im Stillstand ist daher nicht erforderlich. Der Filter soll stattdessen während des Zustands „Moving“ kleinere Schwankungen und ruckartige Änderungen der Trackingdaten reduzieren. Gleichzeitig soll die tatsächliche Kamerabewegung möglichst erhalten bleiben und durch die Filterung nur eine geringe zusätzliche Verzögerung entstehen. Hierfür wird der bereits zuvor untersuchte One-Euro-Filter eingesetzt. Zur Bestimmung geeigneter Filterparameter wurden FreeD-Trackingdaten für Stillstand, einen gleichmäßigen Schwenk sowie schnelle ruckartige Bewegungen aufgezeichnet. Die identischen Rohdaten wurden anschließend mit verschiedenen Kombinationen der Parameter min_cutoff und β verarbeitet. Für die Parameterwahl wurden insbesondere die Bewegungssequenzen betrachtet. Als Bewertungskriterien dienten die erreichte Glättungswirkung, die Erhaltung der Bewegungsspitzen sowie die durch den Filter verursachte zeitliche Verzögerung. Der Filterdelay wurde über eine Kreuzkorrelation zwischen der aus den Rohdaten bestimmten Bewegungsgeschwindigkeit und der Geschwindigkeit des gefilterten Signals ermittelt. Dadurch konnten alle Filterkonfigurationen anhand derselben aufgezeichneten Eingangsdaten reproduzierbar miteinander verglichen werden.

min_cutoff	β	Glättungswirkung* [%]	Peak-Erhaltung [%]	Delay [ms]	Korrelation
1,0	0	35,2	67,2	61,9	0,72
1,0	0,01	29,3	77,8	61,9	0,72

1,0	0,1	9,0	97,8	≈ 0	0,79
3,0	0	18,7	89,2	31,0	0,77
3,0	**0,01**	**16,4**	**92,1**	**31,0**	**0,77**

Tabelle 7: Vergleich der One-Euro-Filterparameter anhand der Pan-Achse beim Rütteltest

** Glättungswirkung als Reduktion der Signalvariation gegenüber den ungefilterten Trackingdaten.*

Wie Tabelle 7 zeigt, führen stärker glättende Einstellungen zu einer deutlicheren Abschwächung der tatsächlichen Bewegung und teilweise zu einem höheren Delay.

Als Kompromiss wurde daher die Kombination min_cutoff = 3,0 und β = 0,01 ausgewählt. Beim Rütteltest auf der Pan-Achse wurde die Signalvariation um 16,4 % reduziert, während gleichzeitig 92,1 % der maximalen Bewegungsamplitude erhalten blieben. Die mittels Kreuzkorrelation bestimmte zusätzliche Verzögerung betrug etwa 31 ms. Stärker filternde Einstellungen erzielten zwar eine höhere Glättungswirkung, führten jedoch zu einer stärkeren Abschwächung der Bewegung und teilweise zu einer höheren Verzögerung. Schwächere Einstellungen erhielten die ursprüngliche Bewegung dagegen besser, erzielten jedoch eine geringere Glättungswirkung. Die gewählte Parameterkombination stellt damit einen geeigneten Kompromiss zwischen der Stabilisierung der Trackingdaten, dem Erhalt der Kamerabewegung und einer möglichst geringen zusätzlichen Verzögerung dar und wird für die weitere Verarbeitung während der Bewegungsphasen verwendet.

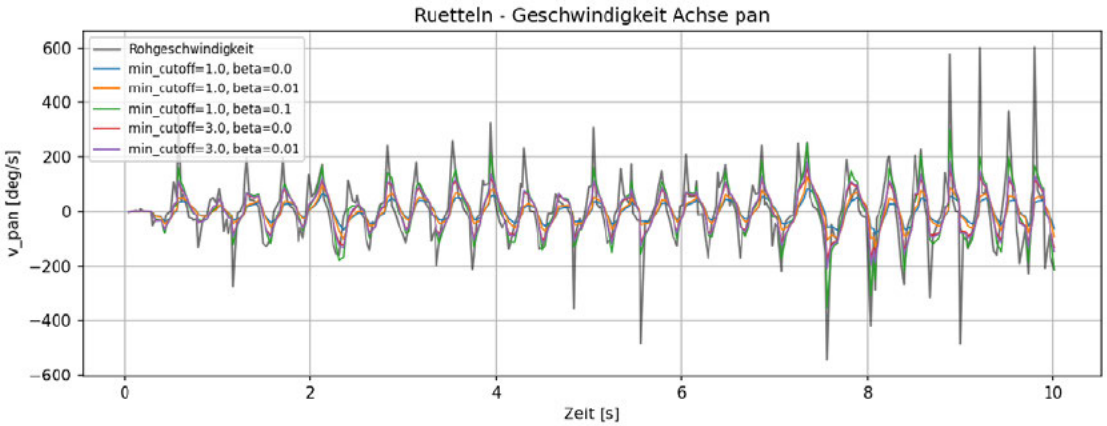


Abbildung 25: Vergleich ausgewählter One-Euro-Filterparameter anhand der Pan-Geschwindigkeit während einer schnellen Kamerabewegung.

4.6.5 Entwicklung der grafischen Benutzeroberfläche

Um die Bedienbarkeit zu vereinfachen, wird für das bislang nur über die Kommandozeile bedienbare Programm eine grafische Benutzeroberfläche entwickelt. Die Umsetzung erfolgt mit Tkinter, dem GUI-Toolkit der Python-Standardbibliothek, sodass keine

zusätzliche Abhängigkeit zum bereits in Python realisierten Programm entsteht. Für einen ersten Layoutentwurf wurde ergänzend ein KI-Sprachmodell (Claude, Anthropic) auf Grundlage der im Folgenden aufgeführten Ein- und Ausgabewerte (Anlage 3, Tabelle 18, 19), einer Skizze sowie der Funktionsweise des Programms herangezogen. Zusätzlich zeigt ein Live-Liniendiagramm den in Kapitel 4.5.3 beschriebenen aktuellen Bewegungsenergiewert und den des ausgehenden Trackingstroms, fortlaufend über die Zeit, in das der obere und der untere Schwellenwert als Referenzlinien eingezeichnet sind. Dadurch wird unmittelbar sichtbar, wie sich der Bewegungsenergiewert im Verhältnis zu beiden Schwellenwerten verhält, was deren Wahl beziehungsweise Nachjustierung erleichtert. Abbildung 26 zeigt die so entstandene Oberfläche.

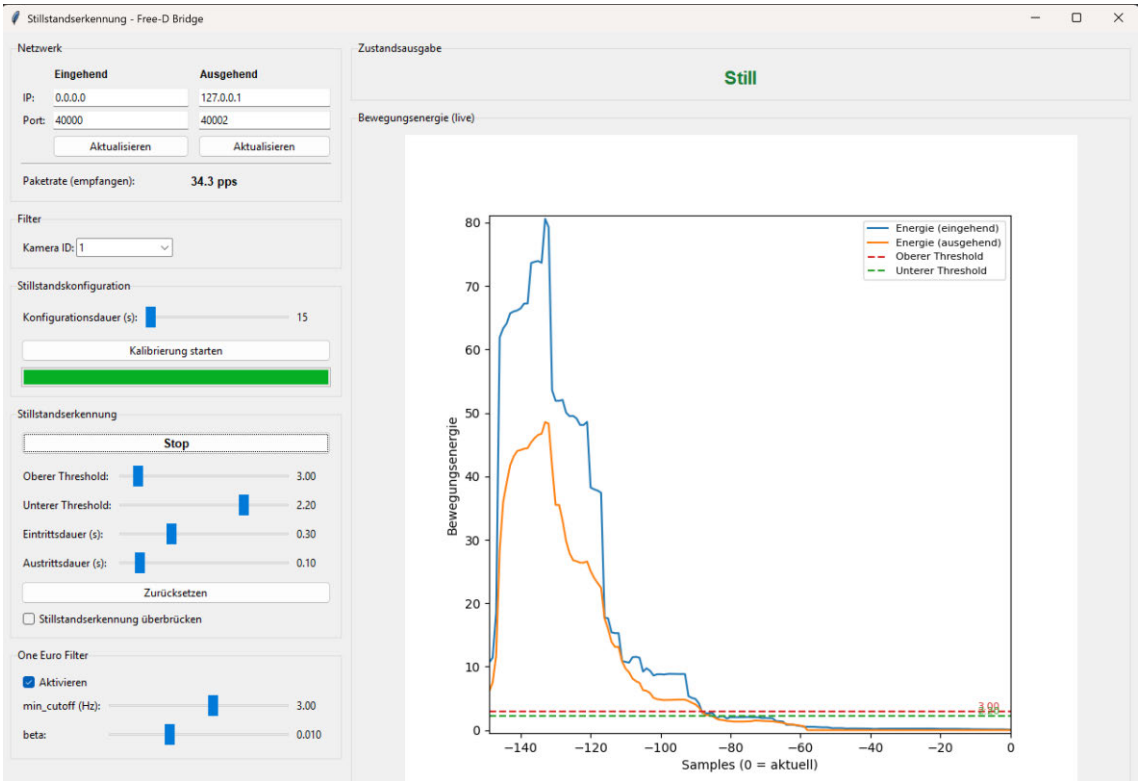


Abbildung 26: GUI des Programms zur Stillstandserkennung

4.6.6 Ergebnisse

In der letzten Testphase wurde das vollständige Programm unter drei unterschiedlichen Trackingbedingungen geprüft. Neben einem Aufbau mit vier Lighthouses wurden sowohl der ungefilterte Datenstrom der VIVE Mars als auch deren interne Filterung auf Stufe 3 von 5 untersucht. Zusätzlich erfolgte ein Test mit nur zwei Lighthouses und deaktiviertem Trackingdatenfilter der VIVE-Mars. Nach jeder Änderung des Aufbaus wurde das System neu kalibriert, damit die Stillstandserkennung an das jeweils vorhandene Rauschniveau angepasst wurde. Anschließend wurden Stillstandsphasen und Kamerabewegungen mehrfach nacheinander ausgeführt. Das Programm ließ sich in allen drei

Konfigurationen erfolgreich kalibrieren und wechselte zuverlässig zwischen den Zuständen „Still“ und „Moving“. Die während der Kalibrierung bestimmten Rauschschwellen sind in Tabelle 8 zusammengefasst. σ_t beschreibt den Rauschwert der Translationsgeschwindigkeit und σ_r den entsprechenden Wert der Rotationsgeschwindigkeit. Diese Werte bilden die an den jeweiligen Aufbau angepasste Grundlage für die anschließende Normierung der Bewegungsenergie und damit für die zustandsabhängigen Schwellwerte.

Test	Lighthouses	VIVE-Mars-Filter	σ_t [mm/s]	σ_r [Grad/s]
1	4	deaktiviert	7,633	0,2582
2	4	Stufe 3/5	0,439	0,0099
3	2	deaktiviert	7,326	0,2650

Tabelle 8: Kalibrierte Rauschschwellen der drei Testkonfigurationen

Die deutlich niedrigeren Werte bei aktivierter Filterstufe 3/5 zeigen, dass die interne Filterung der VIVE Mars bereits einen großen Teil der kleinen Positions- und Rotationsänderungen unterdrückt. Die ähnlichen Rauschschwellen dieser beiden Messungen zeigen, dass die Reduzierung von vier auf zwei Lighthouses im Stillstand keinen erkennbaren Einfluss hatte.

4.7 Abschließendes Ergebnis der Trackingstabilisierung

Als abschließendes Ergebnis liegt ein funktionsfähiges Programm vor, das den FreeD-Datenstrom der VIVE Mars vollständig verarbeitet und die Trackingdaten stabilisiert an UE weiterleitet. Nach der automatischen Kalibrierung unterscheidet die Anwendung zuverlässig zwischen Stillstand und Bewegung. Im Zustand „Still“ wird der zuletzt übertragene Wert eingefroren, sodass Drift und Trackingrauschen im virtuellen Kamerabild vollständig unterdrückt werden. Während einer Bewegung werden die Daten mit dem ausgewählten One-Euro-Filter geglättet. Der Zustandswechsel erfolgt ohne sichtbare Sprünge, während reale Kamerabewegungen weiterhin zeitnah übertragen werden. Die Daten werden anschließend wieder zu gültigen Free-D-Paketen zusammengesetzt und über den konfigurierten Port an UE gesendet.

Die abschließenden Tests bestätigen, dass das entwickelte Programm zuverlässig arbeitet. Durch die erneute Kalibrierung nach einer Änderung des Trackingaufbaus werden die Schwellwerte an das jeweilige Rauschniveau angepasst. Damit erfüllt das Programm die für die Trackingstabilisierung formulierten Ziele: Es reduziert unerwünschte Bewegungen im Stillstand, glättet Bewegungsphasen mit geringer zusätzlicher Verzögerung

und stellt die stabilisierten Trackingdaten über eine grafische Benutzeroberfläche für den VP-Workflow bereit.

5. Evaluation der entwickelten Lösungen

Die beiden entwickelten Anwendungen werden im Folgenden anhand der in Kapitel 1.4 festgelegten Kriterien Echtzeitfähigkeit, Stabilität, Genauigkeit, Rechenaufwand sowie Bedienbarkeit und Integrationsfähigkeit bewertet. Da sich Funktionsumfang und Reifegrad der Lösungen unterscheiden, erfolgt die Evaluation getrennt.

5.1 Evaluation der Latenzbestimmung

Echtzeitfähigkeit: Die Anwendung verarbeitet die beiden SDI-Signale mit 50 fps und führt die Bewegungsanalyse während des laufenden Betriebs aus. Durch die threadbasierte Erfassung, das Herunterskalieren der Bilder und die getrennte Berechnung geeigneter Bewegungsfenster konnte der vollständige Ablauf auf dem Testsystem in Echtzeit ausgeführt werden. Ein einzelner berechneter Wert sollte jedoch nicht unmittelbar als endgültiger Korrekturwert übernommen werden. Für eine belastbarere Aussage sind mehrere gültige Bewegungsfenster erforderlich, deren Ergebnisse geglättet beziehungsweise gemeinsam ausgewertet werden. Die Messung ist deshalb livefähig, benötigt für ein stabiles Gesamtergebnis aber mehrere Testbewegungen.

Stabilität: Alle 13 gültigen Messungen der abschließenden Testreihe zeigten dieselbe Verzögerungsrichtung. Der Mittelwert lag bei -41,042 ms beziehungsweise -2,052 Frames und die mittlere Korrelationsgüte bei 0,906. Die Einzelwerte streuten allerdings zwischen -6,359 und -66,433 ms. Darüber hinaus traten sporadische Bildnummernsprünge auf. Die Anwendung erkennt solche Fehler der Zeitbasis und verwirft die betroffenen Fenster kontrolliert, sodass das letzte gültige Ergebnis erhalten bleibt. Diese Fehlerbehandlung erhöht die Betriebssicherheit, beseitigt jedoch nicht die Ursache der Frameverluste. Die Stabilität der Messwerte ist daher noch nicht ausreichend für eine uneingeschränkt zuverlässige Nutzung.

Genauigkeit: Das Verfahren kann vorhandene Abweichungen grundsätzlich erfassen und ihre Richtung bestimmen. Eine abschließende Aussage zur absoluten Genauigkeit ist jedoch nicht möglich, da in der Hauptmessreihe kein unabhängig festgelegter Referenzversatz zur Verfügung stand. Auch die Kontrollmessung mit einer identischen, über einen SDI-Splitter verteilten Quelle zeigt die verbleibende Unsicherheit: Fünf von sechs Messungen ergaben erwartungsgemäß 0 ms, eine Messung dagegen trotz einer Korrelationsgüte von 0,985 einen Versatz von 33,379 ms. Eine hohe Korrelationsgüte allein

garantiert somit kein korrektes Ergebnis. Zusätzlich werden die Bewegungen der y-Achse in den beiden Quellen teilweise mit entgegengesetztem Vorzeichen erfasst. Dieser noch bestehende Fehler kann die achsenbezogene Korrelation beeinträchtigen und muss vor einer abschließenden Genauigkeitsbewertung behoben werden.

Rechenaufwand: Im Vergleich zur Trackingstabilisierung ist der Rechenaufwand hoch, da zwei SDI-Videoströme fortlaufend erfasst und mit jeweils 50 Bildern pro Sekunde verarbeitet werden. Neben der Umwandlung in Luminanzdaten und dem optischen Tracking werden Vorschau, Bewegungstrigger, Filterung, Normierung und Kreuzkorrelation ausgeführt. Das Downscaling und die Verwendung der direkten Kreuzkorrelation reduzieren den Aufwand jedoch deutlich. Zugleich beanspruchen insbesondere die Bildverarbeitung und die RGB-Vorschauen weiterhin erhebliche Systemressourcen. Auf dem verwendeten Testsystem blieb die Anwendung lauffähig, für eine parallele Nutzung mit weiteren rechenintensiven Produktionsanwendungen wurde die Leistungsreserve jedoch nicht systematisch untersucht.

Bedienbarkeit und Integrationsfähigkeit: Die grafische Benutzeroberfläche vereinfacht die Auswahl und Kontrolle der SDI-Quellen, die Kalibrierung der Bewegungstrigger sowie die Anzeige der Messwerte und Diagnoseinformationen. Die Anwendung kann damit ohne direkte Bedienung einzelner Konsolenmodule genutzt werden. Wegen der Vielzahl an Parametern, der getrennten Quellenauswahl und der notwendigen Kalibrierung ist der Ablauf jedoch noch nicht vollständig intuitiv. Für einen regelmäßigen Produktionseinsatz wären stärker geführte Arbeitsschritte und klarere Standardwerte sinnvoll. Die Integration in den vorhandenen Aufbau ist über die DeckLink Duo 2 grundsätzlich möglich. Der ermittelte Wert muss anschließend weiterhin manuell als Kompensation in der Ultimatte eingestellt werden.

Insgesamt erfüllt die Latenzanwendung die Kriterien Echtzeitfähigkeit und grundsätzliche Integrationsfähigkeit, während Stabilität und Genauigkeit nur teilweise erreicht werden. Sie ist als funktionsfähiger Prototyp und Diagnosewerkzeug geeignet, jedoch noch nicht als alleinige Grundlage für eine verlässlich reproduzierbare Latenzkompensation im Produktionsbetrieb.

5.2 Evaluation der Trackingstabilisierung

Echtzeitfähigkeit: Die Anwendung verarbeitet die eingehenden FreeD-Pakete fortlaufend und leitet die stabilisierten Daten unmittelbar an UE weiter. Da ausschließlich numerische Positions- und Rotationswerte verarbeitet werden, kann die Zustandsbestimmung ohne Unterbrechung des Datenstroms erfolgen. Die gewählte One-Euro-

Filterkonfiguration aus min_cutoff = 3,0 und β = 0,01 verursachte im Rütteltest eine zusätzliche Verzögerung von etwa 31 ms. Hinzu kommen die bewusst gewählten Bestätigungszeiten der Hysterese von 0,1 s beim Wechsel zu „Moving“ und 0,3 s beim Wechsel zu „Still“. Diese Zeiten verhindern Fehlumschaltungen und waren in den Versuchen kurz genug, um keine sichtbaren Sprünge beim Beginn einer Bewegung hervorzurufen. Das Kriterium der Echtzeitfähigkeit wird damit erfüllt.

Stabilität: Das vollständige Programm konnte in allen drei geprüften Konfigurationen erfolgreich kalibriert werden und wechselte wiederholt zuverlässig zwischen „Still“ und „Moving“. Dies galt sowohl für vier Lighthouses mit deaktiviertem VIVE-Mars-Filter als auch für vier Lighthouses mit Filterstufe 3/5 und für einen auf zwei Lighthouses reduzierten Aufbau ohne internen Filter. Die RMS-basierten Kalibrierwerte passten sich dabei deutlich an das jeweilige Rauschniveau an: Bei deaktivierter Filterung lagen σ_t und σ_r mit vier Lighthouses bei 7,633 mm/s und 0,2582 Grad/s, mit zwei Lighthouses bei 7,326 mm/s und 0,2650 Grad/s. Die automatische Neukalibrierung ermöglicht somit einen stabilen Betrieb unter unterschiedlichen Bedingungen. Die bisherige Testdauer und Zahl der Konfigurationen reichen allerdings noch nicht aus, um die Langzeitstabilität abschließend zu bewerten.

Genauigkeit: Während einer Bewegung reduziert der One-Euro-Filter bei der ausgewählten Parametrierung die Signalvariation im Rütteltest um 16,4 %, während 92,1 % der maximalen Bewegungsamplitude erhalten bleiben. Der Zustandswechsel erfolgte in den Versuchen ohne sichtbare Sprünge. Damit konnten die zuvor beobachteten Trackinginstabilitäten im Stillstand wirksam unterdrückt werden. Die Genauigkeit hängt vor allem von einer korrekten Kalibrierung und einer zum Aufnahmeverhalten passenden Wahl der Schwellwerte ab.

Rechenaufwand: Der Rechenaufwand ist gering, da pro Paket nur die FreeD-Werte dekodiert, Geschwindigkeiten und Bewegungsenergie berechnet, die Zustandslogik ausgeführt und die Daten erneut kodiert werden. Anders als bei der Latenzmessung ist keine kontinuierliche Bildverarbeitung erforderlich. Das Programm eignet sich daher grundsätzlich für den parallelen Betrieb mit UE und weiteren Komponenten des VP-Aufbaus.

Bedienbarkeit und Integrationsfähigkeit: Die grafische Benutzeroberfläche fasst Kalibrierung, Schwellwerte, Filtereinstellungen und Zustandsanzeige zusammen. Durch die Darstellung von ein- und ausgehender Bewegungsenergie, Paketrate und aktuellem Zustand kann der Datenfluss während des Betriebs überwacht werden. Die Einbindung erfordert jedoch, dass die VIVE Mars ihre FreeD-Daten an das Programm sendet und UE auf den neuen Ausgangsport eingestellt wird. Dieser Konfigurationsschritt setzt

grundlegende Kenntnisse zu IP-Adressen, Ports und der FreeD-Quelle voraus. Da das Programm zwischen Trackingquelle und UE liegt, stellt es außerdem einen zusätzlichen Ausfallpunkt dar: Wird die Anwendung geschlossen oder fällt der zugehörige Rechner aus, erreichen keine Trackingdaten mehr die UE. Für einen dauerhaften Produktionsbetrieb wäre daher ein automatischer, vom Programmzustand unabhängiger Bypass beziehungsweise eine Ausfallüberwachung sinnvoll.

Insgesamt erfüllt die Trackingstabilisierung die festgelegten Kriterien weitgehend. Sie arbeitet in Echtzeit, benötigt nur geringe Rechenleistung und beseitigt unerwünschte Bewegungen im Stillstand zuverlässig. Die wesentlichen Einschränkungen liegen in der noch begrenzten Langzeiterprobung, der möglichen Fehleinordnung sehr langsamer Bewegungen und der fehlenden Ausfallsicherheit der zwischengeschalteten Anwendung.

5.3 Kompatibilität der Anwendungen mit einem Mehrkamera-Aufbau

Die entwickelte Latenzanwendung ist in ihrer aktuellen Form für den Vergleich eines Vordergrund- und eines Hintergrundsignals ausgelegt. Für jede zu prüfende Kamera werden damit zwei SDI-Signale benötigt. Die Zahl gleichzeitig anschließbarer Signalpaare wird durch die verfügbaren Eingänge der DeckLink Duo 2 und durch die Auslegung der Software auf ein einzelnes Quellenpaar begrenzt. In einem Mehrkamera-Aufbau könnten die Kameras nacheinander geprüft und die benötigten Signale bspw. über eine SDI-Kreuzschiene auf die verwendeten Eingänge geschaltet werden. Eine parallele Messung mehrerer Kameras würde dagegen zusätzliche Ein- und Ausgänge sowie eine Erweiterung der Anwendung um mehrere voneinander getrennte Erfassungs- und Berechnungsinstanzen erfordern. Wegen des hohen Rechenaufwands müsste dabei auch die verfügbare Systemleistung erneut bewertet werden.

Bei der Trackingstabilisierung ist eine Erweiterung grundsätzlich über getrennte FreeD-Datenströme möglich. Die VIVE Mars kann unterschiedliche Kameratracker über verschiedene Kamera-IDs und Zielports ausgeben. Für einen ersten Mehrkamerabetrieb könnten daher mehrere Instanzen des Programms mit jeweils eigenen Empfangs- und Sendeports gestartet werden. Alternativ ließe sich die bestehende Anwendung um eine zentrale Verwaltung mehrerer Tracker ergänzen. Dabei müssten Rauschboden, Zustandslogik und Filterzustand für jede Kamera getrennt geführt werden.

6. Fazit

6.1 Zusammenfassung der Ergebnisse

Im Rahmen dieser Arbeit wurden zwei Softwarelösungen für den VP-Aufbau des Videolabors der HAW Hamburg entwickelt. Die erste Anwendung bestimmt den zeitlichen Versatz zwischen realem Vordergrund und gerendertem Hintergrund. Dazu werden Bewegungen in beiden SDI-Signalen erfasst, aufbereitet und durch Kreuzkorrelation miteinander verglichen. Die zweite Anwendung stabilisiert den FreeD-Trackingdatenstrom der VIVE Mars durch eine zustandsabhängige Verarbeitung. Die Latenzanwendung konnte Verzögerungen grundsätzlich erkennen und ihre Richtung bestimmen. SDI-Erfassung, Bewegungstrigger, Qualitätsprüfung und grafische Benutzeroberfläche wurden erfolgreich zu einem funktionsfähigen Prototyp zusammengeführt. Die Streuung der Messwerte, sporadische Frameverluste und ein fehlerhafter Wert in der Kontrollmessung zeigen jedoch, dass die Wiederholgenauigkeit für eine verlässliche Kompensation noch nicht ausreicht. Bei der Trackingstabilisierung erwies sich die Kombination aus kalibrierter Stillstandserkennung, Einfrieren der Pose und One-Euro-Filter als geeignet. Trackingrauschen und Drift werden im Stillstand unterdrückt, während reale Kamerabewegungen weitgehend erhalten bleiben. Die Anwendung arbeitete in den geprüften Konfigurationen zuverlässig.

6.2 Beantwortung der Forschungsfrage

Die in Kapitel 1.5 formulierte Forschungsfrage lautete, wie Latenz und Trackinginstabilitäten in virtuellen Produktionsumgebungen zuverlässig und echtzeitfähig kompensiert werden können. Die Ergebnisse zeigen, dass dafür zwei unterschiedliche, an den jeweiligen Datenstrom angepasste Verfahren notwendig sind. Latenzen lassen sich anhand übereinstimmender Bewegungsverläufe beider Videobilder bestimmen, während Trackinginstabilitäten durch eine kalibrierte Erkennung von Stillstand und Bewegung sowie eine zustandsabhängige Verarbeitung der Trackingdaten reduziert werden können.

Als wesentliche Ursachen der untersuchten Probleme wurden einerseits die Verarbeitungs- und Renderzeit der UE sowie weiterer Komponenten im Signalweg und andererseits Rauschen und Drift des Trackingsystems festgestellt. Für den Livebetrieb erwiesen sich bei der Latenzmessung die direkte Kreuzkorrelation gefilterter und normierter Geschwindigkeitsverläufe und bei der Trackingstabilisierung die Kombination aus doppeltem Schwellwert, Einfrieren der Pose und One-Euro-Filter als geeignete Ansätze. Beide Verfahren arbeiten in Echtzeit. Die Latenzmessung ist im aktuellen Entwicklungsstand jedoch noch nicht ausreichend reproduzierbar.

Für die Trackingstabilisierung hat sich insbesondere gezeigt, dass eine individuelle Kalibrierung des Rauschbodens erforderlich ist, da sich dieser abhängig vom jeweiligen Trackingaufbau verändert. Als geeignete Grenzwerte für die berechnete Bewegungsenergie wurde eine untere Schwelle von 2,2 und eine obere Schwelle von 3,0 ermittelt. Der Wechsel zu „Moving“ erfolgt nach 0,1 s oberhalb der oberen Schwelle, während der Wechsel zu „Still“ nach 0,3 s unterhalb der unteren Schwelle erfolgt. Für die Bewegungsphasen erwies sich eine One-Euro-Filterkonfiguration mit min_cutoff = 3,0 und β = 0,01 als geeignet. Die konkrete Parametrierung ermöglicht eine zuverlässige Unterscheidung zwischen Stillstand und Bewegung sowie eine Glättung der Bewegungsdaten. Da sich der Rauschboden abhängig vom Aufbau verändert, sind die ermittelten Werte jedoch nicht allgemein übertragbar und müssen nach Änderungen des Trackingsystems neu bestimmt werden. Für die Latenzbestimmung eignen sich deutliche, zusammenhängende und möglichst nichtperiodische Kamerabewegungen, die in beiden Videobildern eindeutig sichtbar sind. Bewegungen in mehreren Bildachsen verbessern die Plausibilitätsprüfung der Einzelergebnisse. Sehr kurze, langsame oder gleichförmig wiederholte Bewegungen sind weniger geeignet, da sie kein eindeutiges Korrelationsmaximum erzeugen können. Für die Kalibrierung der Bewegungstrigger und des Trackingrauschbodens muss die Kamera dagegen bewusst stillgehalten werden. Zusammenfassend wurde die Trackingstabilisierung im verwendeten Aufbau weitgehend zuverlässig umgesetzt. Sie unterdrückt Drift und Trackingrauschen im Stillstand, ohne reale Kamerabewegungen wesentlich zu beeinträchtigen. Die Latenzanwendung liefert eine geeignete echtzeitfähige Messmethodik und erkennt die Verzögerungsrichtung, benötigt vor einem produktiven Einsatz aber weitere Validierungen und eine robustere Zusammenfassung mehrerer Messfenster. Die übergeordnete Forschungsfrage kann daher für die Trackingstabilisierung weitgehend, für die zuverlässige Latenzkompensation jedoch nur teilweise beantwortet werden.

6.3 Kritische Reflexion der Arbeit und mögliche Weiterentwicklungen

Die größte Einschränkung der Latenzmessung ist das Fehlen eines exakt bekannten Referenzdelays. Dadurch konnten Plausibilität und Verzögerungsrichtung, nicht aber der absolute Messfehler bestimmt werden. Die geringe Zahl der Kontrollmessungen, ungeklärte Frameverluste und die teilweise gespiegelte y-Achse schränken die Aussagekraft zusätzlich ein. Als nächster Schritt ist daher eine systematische Validierung mit definierten Referenzwerten erforderlich. Gleichzeitig müssen die Achsenzuordnung korrigiert und die Ursache der Frameverluste entlang der gesamten Verarbeitungskette untersucht werden.

Der fehlerhafte Wert bei identischen Splitter-Signalen zeigt außerdem, dass eine hohe Korrelationsgüte allein kein korrektes Ergebnis garantiert. Eine robustere Ergebnisbildung sollte deshalb mehrere unabhängige Bewegungsfenster berücksichtigen und Ausreißer sowie uneindeutige Korrelationsmaxima erkennen. Dadurch könnte die Wiederholgenauigkeit verbessert werden. Langfristig wäre auch eine direkte Übergabe eines geprüften Korrekturwerts an die Ultimatte denkbar.

Die Trackingstabilisierung wurde unter drei praxisnahen Bedingungen erfolgreich geprüft, jedoch nicht über längere Produktionszeiträume oder mit allen relevanten Kamerabewegungen. Besonders langsame beabsichtigte Bewegungen können einem Stillstand ähneln. Weitere Versuche sollten daher Langzeittests, langsame Kamerafahrten, Verdeckungen und zusätzliche Lighthouse-Konfigurationen umfassen. Auf dieser Grundlage könnten die Schwellwerte stärker an die aktuelle Datenqualität angepasst und die Wirkung zusätzlich quantitativ am gerenderten Bild bewertet werden.

Der Integrationsaufwand begrenzte die Zeit für abschließende Tests. Für einen dauerhaften Einsatz sind einfachere Bedienabläufe, automatische Prüfungen und eine Ausfallsicherung der Trackingweiterleitung sinnvoll. Eine Mehrkamera-Erweiterung über zusätzliche SDI-Signalpaare und FreeD-Streams muss hinsichtlich Ressourcen und Stabilität gesondert erprobt werden.

7. Literaturverzeichnis

Alonso, F. J.; Del Castillo, J. M.; Pintado, P. (2005): Application of singular spectrum analysis to the smoothing of raw kinematic signals. In: Journal of Biomechanics, Vol. 38, Issue 5, S. 1085–1092. Verfügbar unter: https://doi.org/10.1016/j.jbiomech.2004.05.031 (Zugriff am: 26.08.2026).

Blackmagic Design (o. J.-a): DeckLink – Tech Specs. Verfügbar unter: https://www.blackmagicdesign.com/de/products/decklink/techspecs/W-DLK-31 (Zugriff am: 08.08.2026).

Blackmagic Design (o. J.-b): Desktop Video Developer Support FAQs. Verfügbar unter: https://www.blackmagicdesign.com/developer/support/faq/desktop-video-developer-support-faqs (Zugriff am: 23.08.2026).

Blackmagic Design (2026): Ultimatte 12 – Operations Manual. Verfügbar unter: https://documents.blackmagicdesign.com/UserManuals/UltimatteManual.pdf?_v=1772697610000 (Zugriff am: 26.08.2026).

Canny, J. (1986): A Computational Approach to Edge Detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, Issue 6, S. 679–698. Verfügbar unter: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4767851 (Zugriff am: 26.08.2026).

Casiez, G.; Roussel, N.; Vogel, D. (2012): 1 € Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12), S. 2527–2530. Verfügbar unter: https://dl.acm.org/doi/pdf/10.1145/2207676.2208639 (Zugriff am: 26.08.2026).

Coakley, K. J.; Hale, P. (2001): Alignment of Noisy Signals. In: IEEE Transactions on Instrumentation and Measurement, Vol. 50, Issue 1, S. 141–149. Verfügbar unter: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=903892 (Zugriff am: 20.07.2026).

Crenna, F.; Rossi, G. B.; Berardengo, M. (2021): Filtering Biomechanical Signals in Movement Analysis. In: Sensors, Vol. 21, Issue 13, Artikel 4580, S. 1–17. Verfügbar unter: https://doi.org/10.3390/s21134580 (Zugriff am: 26.08.2026).

Culjak, I.; Abram, D.; Pribanic, T.; Dzapo, H.; Cifrek, M. (2012): A brief introduction to OpenCV. In: 2012 Proceedings of the 35th International Convention MIPRO, Opatija, Kroatien, S. 1725–1730. Verfügbar unter: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6240859 (Zugriff am: 09.08.2026).

Dean, L.; Gilardi, F.; Jamil, S.; Tu, S.; Zhu, Z. (2026): Developing an industrial workflow for virtual production filmmaking. In: Convergence: The International Journal of Research into New Media Technologies, Vol. 32, Issue 2, S. 435–460. Verfügbar unter: https://journals.sagepub.com/doi/epub/10.1177/13548565261433131 (Zugriff am: 21.07.2026).

Disguise (2026): Video Receive Delay. In: Disguise User Guide. Verfügbar unter: https://help.disguise.one/workflows/xr/video-receive-delay (Zugriff am: 16.07.2026).

Eggert, L.; Fairhurst, G.; Shepherd, G. (2017): UDP Usage Guidelines. BCP 145, RFC 8085. Internet Engineering Task Force. Verfügbar unter: https://doi.org/10.17487/RFC8085 (Zugriff am: 26.08.2026).

Epic Games (o. J.-a): Live Link in Unreal Engine. Verfügbar unter: https://dev.epicgames.com/documentation/en-us/unreal-engine/live-link-in-unreal-engine (Zugriff am: 08.08.2026).

Epic Games (o. J.-b): Live Link FreeD in Unreal Engine. In: Unreal Engine Documentation. Verfügbar unter: https://dev.epicgames.com/documentation/unreal-engine/live-link-freed--in-unreal-engine (Zugriff am: 26.08.2026).

Exscriber (2020): TD-FreeD – FreeD-Protokoll-Parser in Python [GitHub-Repository]. Verfügbar unter: https://github.com/exscriber/TD-FreeD/blob/master/FreeD.py (Zugriff am: 19.08.2026).

Gebhard, C. (2021): ZDF-Nachrichten in neuem Design. In: film-tv-video.de, 21.07.2021. Verfügbar unter: https://www.film-tv-video.de/productions/2021/07/21/zdf-nachrichten-in-neuem-design/ (Zugriff am: 21.07.2026).

Gustafsson, F. (1996): Determining the Initial States in Forward-Backward Filtering. In: IEEE Transactions on Signal Processing, Vol. 44, Issue 4, S. 988–992. Verfügbar unter: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=492552 (Zugriff am: 21.07.2026).

Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P. et al. (2020): Array programming with NumPy. In: Nature, Vol. 585, Issue 7825, S. 357–362. Verfügbar unter: https://doi.org/10.1038/s41586-020-2649-2 (Zugriff am: 23.08.2026).

HTC VIVE (o. J.): VIVE Mars CamTrack. Verfügbar unter: https://www.mars.vive.com/ (Zugriff am: 26.08.2026).

HTC VIVE (2022): VIVE Mars CamTrack – User Guide. Verfügbar unter: https://dl4.htc.com/Web_materials/Manual/CamTrack/HTC_VIVE_Mars_CamTrack_User_Guide.pdf (Zugriff am: 04.08.2026).

Jia, X.; Berry, A.; Johnston, A. (2026): The evolutionary disruption: A paradigm shift in film and animation industry driven by real-time rendering and virtual production. In: Convergence, Vol. 32, Issue 2, S. 461–481. Verfügbar unter: https://journals.sagepub.com/doi/epub/10.1177/13548565251356932 (Zugriff am: 08.08.2026).

Kaso, A. (2018): Computation of the normalized cross-correlation by fast Fourier transform. In: PLOS ONE, Vol. 13, Issue 9, Artikel e0203434. Verfügbar unter: https://doi.org/10.1371/journal.pone.0203434 (Zugriff am: 20.07.2026).

Kuhlmann de Canaviri, L.; Meiszl, K.; Hussein, V.; Abbassi, P.; Mirraziroudsari, S. D.; Hake, L.; Potthast, T.; Ratert, F.; Schulten, T.; Silberbach, M.; Warnecke, Y.; Wiswede, D.; Schiprowski, W.; Heß, D.; Brüngel, R.; Friedrich, C. M. (2023): Static and Dynamic Accuracy and Occlusion Robustness of SteamVR Tracking 2.0 in Multi-Base Station Setups. In: Sensors, Vol. 23, Issue 2, Artikel 725. Verfügbar unter: https://doi.org/10.3390/s23020725 (Zugriff am: 04.08.2026).

Lucas, B. D.; Kanade, T. (1981): An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, S. 674–679. Verfügbar unter: https://dl.acm.org/doi/10.5555/1623264.1623280 (Zugriff am: 20.07.2026).

Lyon, D. A. (2010): The Discrete Fourier Transform, Part 6: Cross-Correlation. In: Journal of Object Technology, Vol. 9, Issue 2, S. 17–22. Verfügbar unter: https://doi.org/10.5381/jot.2010.9.2.c2 (Zugriff am: 20.07.2026).

Millman, K. J.; Aivazis, M. (2011): Python for Scientists and Engineers. In: Computing in Science & Engineering, Vol. 13, Issue 2, S. 9–12. Verfügbar unter: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5725235 (Zugriff am: 23.08.2026).

Niehorster, D. C.; Li, L.; Lappe, M. (2017): The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research. In: i-Perception, Vol. 8, Issue 3, S. 1–23. Verfügbar unter: https://journals.sagepub.com/doi/epub/10.1177/2041669517708205 (Zugriff am: 26.08.2026).

NumPy Developers (o. J.): numpy.gradient. NumPy Reference. Verfügbar unter: https://numpy.org/doc/2.1/reference/generated/numpy.gradient.html (Zugriff am: 26.08.2026).

OpenCV (o. J.): Optical Flow. OpenCV Documentation, Version 3.4.15. Verfügbar unter: https://docs.opencv.org/3.4.15/d4/dee/tutorial_optical_flow.html (Zugriff am: 26.08.2026).

Oppenheim, A. V.; Schafer, R. W. (2010): Discrete-Time Signal Processing. 3. Auflage. Upper Saddle River: Pearson/Prentice Hall. Verfügbar unter: https://ocw.mit.edu/courses/res-6-dtsp-discrete-time-signal-processing/resources/mitres_6-dtsp_s26_thirdedition_pdf/ (Zugriff am: 26.08.2026).

Python Software Foundation (o. J.): tkinter — Python interface to Tcl/Tk. Verfügbar unter: https://docs.python.org/3/library/tkinter.html (Zugriff am: 09.08.2026).

Quian Quiroga, R.; Kreuz, T.; Grassberger, P. (2002): Event Synchronization: A Simple and Fast Method to Measure Synchronicity and Time Delay Patterns. In: Physical Review E, Vol. 66, Artikel 041904. Verfügbar unter: https://doi.org/10.1103/PhysRevE.66.041904 (Zugriff am: 20.07.2026).

Ryzhak, I. (2026): FREED Tracker. Version 1.3 [Mobile App]. Apple App Store. Verfügbar unter: https://apps.apple.com/de/app/freed-tracker/id6771544166 (Zugriff am: 26.08.2026).

Sansone, L. G.; Stanzani, R.; Job, M.; Battista, S.; Signori, A.; Testa, M. (2022): Robustness and Static-Positional Accuracy of the SteamVR 1.0 Virtual Reality Tracking System. In: Virtual Reality, Vol. 26, S. 903–924. Verfügbar unter: https://doi.org/10.1007/s10055-021-00584-5 (Zugriff am: 26.08.2026).

SciPy Community (o. J.-a): scipy.signal.butter. SciPy API Reference. Verfügbar unter: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html (Zugriff am: 26.08.2026).

SciPy Community (o. J.-b): scipy.signal.filtfilt. SciPy API Reference. Verfügbar unter: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.filtfilt.html (Zugriff am: 26.08.2026).

Smith, S. W. (1999): The Scientist and Engineer's Guide to Digital Signal Processing. 2. Auflage. San Diego: California Technical Publishing. Verfügbar unter: https://www.dspguide.com/pdfbook.htm (Zugriff am: 26.08.2026).

Sobchyshak, O.; Berrezueta-Guzman, S.; Wagner, S. (2026): Pushing the boundaries of immersion and storytelling: A technical review of Unreal Engine. In: Displays, Vol. 91, Artikel 103268, S. 1–16. Verfügbar unter: https://doi.org/10.1016/j.displa.2025.103268 (Zugriff am: 08.08.2026).

Stegen, R. (2021): Stochastik ohne Zufall und Wahrscheinlichkeit. Wiesbaden: Springer Spektrum. Verfügbar unter: https://doi.org/10.1007/978-3-658-33779-7 (Zugriff am: 26.08.2026).

Swords, J.; Willment, N. (2024): The emergence of virtual production – a research agenda. In: Convergence: The International Journal of Research into New Media Technologies, Vol. 30, Issue 5, S. 1557–1574. Verfügbar unter: https://journals.sagepub.com/doi/epub/10.1177/13548565241253903 (Zugriff am: 21.07.2026).

Türkler, L.; Akkan, L. Ö. (2025): Noise Reduction Techniques for Sensor Data: Comparative Analysis of Kalman, Butterworth, Savitzky-Golay, Median, and Moving Average Filters for UWB-Based Position Estimation. In: Celal Bayar University Journal of Science, Vol. 21, Issue 4, S. 146–159. Verfügbar unter: https://doi.org/10.18466/cbayarfbe.1682594 (Zugriff am: 26.08.2026).

Vinten Radamec (2004): free-d Installation Manual. Version 1.4.4. Verfügbar unter: https://manualmachine.com/vintenradamec/freed/1773651-user-manual/ (Zugriff am: 19.08.2026).

Willment, N.; Swords, J. (2023): What is Virtual Production? An Explainer & Research Agenda. Forschungsbericht. York: XR Stories, University of York. Verfügbar unter: https://eprints.whiterose.ac.uk/id/eprint/205754/ (Zugriff am: 21.07.2026).

Wilmott, J. P.; Erkelens, I. M.; Murdison, T. S.; Rio, K. W. (2022): Perceptibility of Jitter in Augmented Reality Head-Mounted Displays. In: 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), S. 470–478. Verfügbar unter: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9995228 (Zugriff am: 26.08.2026).

8. Anhang

Anlage 1

Rechenzeiten der Korrelationsverfahren

Die Auswertung umfasst fünf Bewegungsdatensätze (Horizontal, Vertikal, Diagonal, Kreisbewegung und Rütteln). Alle angegebenen Kennwerte wurden über diese fünf Datensätze gemittelt.

Anmerkung: Angegeben ist jeweils der Mittelwert der in den Datensätzen enthaltenen minimalen Rechenzeiten aus drei Wiederholungen. Der Faktor bezieht sich auf die direkte Kreuzkorrelation.

Rang	Korrelationsverfahren	Mittelwert [ms]	Minimum [ms]	Maximum [ms]	Faktor
1	**Kreuzkorrelation**	**0,1918**	**0,176**	**0,200**	**1,00**
2	Peak Matching	0,3824	0,217	0,572	1,99
3	FFT-Kreuzkorrelation	0,8116	0,708	1,067	4,23
4	Least Squares	7,0116	5,472	8,228	36,56

Tabelle 9: Vergleich der Rechenzeiten der Korrelationsverfahren

Durchschnittliche Güte-Rohwerte nach Korrelationsverfahren

Da die Güte-Rohwerte der Verfahren unterschiedliche Definitionen und Skalen besitzen, werden die Ergebnisse getrennt nach Verfahren dargestellt. Jede Tabelle ist nach absteigendem mittleren Güte-Rohwert sortiert.

Rang	**Datentyp**	**Aufbereitung**	**mittlerer Güte-Rohwert**
1	Position	gefiltert und normiert	0,843
2	Position	ungefiltert	0,842
3	Geschwindigkeit	gefiltert und normiert	0,823
4	Geschwindigkeit	ungefiltert	0,742
5	Beschleunigung	gefiltert und normiert	0,569
6	Beschleunigung	ungefiltert	0,384

Tabelle 10: Güte-Rohwerte – Kreuzkorrelation

Rang	Datentyp	Aufbereitung	mittlerer Güte-Rohwert
1	Position	gefiltert und normiert	0,843
2	Position	ungefiltert	0,842
3	Geschwindigkeit	gefiltert und normiert	0,823
4	Geschwindigkeit	ungefiltert	0,742
5	Beschleunigung	gefiltert und normiert	0,569
6	Beschleunigung	ungefiltert	0,384

Tabelle 11: Güte-Rohwerte – FFT-Kreuzkorrelation

Rang	**Datentyp**	**Aufbereitung**	**mittlerer Güte-Rohwert**

1	Beschleunigung	ungefiltert	5,850
2	Beschleunigung	gefiltert und normiert	4,839
3	Geschwindigkeit	ungefiltert	3,680
4	Geschwindigkeit	gefiltert und normiert	2,835
5	Position	ungefiltert	1,916
6	Position	gefiltert und normiert	1,912

Tabelle 12: Güte-Rohwerte – Peak Matching

Anmerkung: Die Güte-Rohwerte sind nur innerhalb desselben Verfahrens vergleichbar. Bei Peak Matching beschreibt der Rohwert die Prominenz des gefundenen Peaks als z-Score; ein hoher Wert belegt daher nicht automatisch ein korrektes Delay.

Rang	Datentyp	Aufbereitung	mittlerer Güte-Rohwert
1	Geschwindigkeit	gefiltert und normiert	0,055
2	Position	gefiltert und normiert	0,014
3	Geschwindigkeit	ungefiltert	−0,101
4	Beschleunigung	gefiltert und normiert	−0,818
5	Position	ungefiltert	−1,795
6	Beschleunigung	ungefiltert	−4,198

Tabelle 13: Güte-Rohwerte – Least Squares

Konsistenz der Delay-Ergebnisse

Die Konsistenz entspricht der mittleren paarweisen absoluten Abweichung zwischen dem Delay der x-Achse, dem Delay der y-Achse und dem Delay des Bewegungsbetrags. Ein kleiner Wert kennzeichnet eine hohe Übereinstimmung.

Rang	Verfahren	Datentyp	Aufbereitung	mittlere Abweichung [s]

1	Kreuzkorrelation	Geschwindig-keit	gefiltert und normiert	0,0267
2	FFT-Kreuzkorrelation	Geschwindig-keit	gefiltert und normiert	0,0267
3	Least Squares	Geschwindig-keit	gefiltert und normiert	0,0267
4	Kreuzkorrelation	Geschwindig-keit	ungefiltert	0,0533
5	FFT-Kreuzkorrelation	Geschwindig-keit	ungefiltert	0,0533
6	Kreuzkorrelation	Position	ungefiltert	0,0640
7	FFT-Kreuzkorrelation	Position	ungefiltert	0,0640
8	Kreuzkorrelation	Position	gefiltert und normiert	0,0693
9	FFT-Kreuzkorrelation	Position	gefiltert und normiert	0,0693
10	Kreuzkorrelation	Beschleunigung	gefiltert und normiert	0,1173
11	FFT-Kreuzkorrelation	Beschleunigung	gefiltert und normiert	0,1173
12	Least Squares	Geschwindig-keit	ungefiltert	0,1440
13	Least Squares	Beschleunigung	gefiltert und normiert	0,2933
14	Least Squares	Position	gefiltert und normiert	0,4907
15	Least Squares	Beschleunigung	ungefiltert	0,6133
16	Least Squares	Position	ungefiltert	0,7040
17	Kreuzkorrelation	Beschleunigung	ungefiltert	0,7947
18	FFT-Kreuzkorrelation	Beschleunigung	ungefiltert	0,7947
19	Peak Matching	Position	ungefiltert	0,9387
20	Peak Matching	Position	gefiltert und normiert	0,9653
21	Peak Matching	Beschleunigung	ungefiltert	1,5467
22	Peak Matching	Beschleunigung	gefiltert und normiert	1,8507

23	Peak Matching	Geschwindig-keit	ungefiltert	1,9360
24	Peak Matching	Geschwindig-keit	gefiltert und normiert	2,5707

Tabelle 14: Konsistenz der Delay-Ergebnisse

Berechnung: (|d_x − d_y| + |d_x − d_Betrag| + |d_y − d_Betrag|) / 3. Ohne bekannten Soll-Delay beschreibt diese Kennzahl die interne Konsistenz, nicht die absolute Mess-genauigkeit.

Anlage 2

Ein- und Ausgabelemente der Bedienoberfläche zur Quellenauswahl

Element	**Typ**	**Standardwert**	**Funktion**
Gerät (Vordergrund)	Dropdown	–	Auswahl der DeckLink-Quelle für das Vordergrundsignal (Kamera)
Gerät (Hintergrund)	Dropdown	–	Auswahl der DeckLink-Quelle für das Hintergrundsignal (UE)
Geräteliste aktualisieren	Button	/	Suchen nach verfügbaren DeckLink-Geräten und Aktua-lisieren der Geräte-Dropdowns beider Spalten
Videomodus (je SDI-Quelle)	Dropdown	1080p50	Auswahl von Auflösung und Framerate der jeweiligen Quelle (1080p50 / 1080p25 / 720p50)
Farbraum (je SDI-Quelle)	Dropdown	Rec. 709	Auswahl des Farbraums der jeweiligen Quelle (rein infor-mativ, aktuell nicht an cap-ture.exe übergeben)

Element	Typ	Standardwert	Funktion
Aktualisieren (je SDI-Quelle)	Button	/	Starten bzw. Ersetzen der jeweiligen SDI-Quelle mit den getroffenen Einstellungen (Gerät, Videomodus)
Weiter zur Delayberechnung	Button	/	Prüfen und Übernehmen der Konfiguration beider Quellen sowie Wechseln zu GUI 2 – Delay-Berechnung
Bildvorschau Vordergrund	Videovorschau	/	Live-Vorschau des Vordergrundsignals zur Kontrolle der Bildquelle
Bildvorschau Hintergrund	Videovorschau	/	Live-Vorschau des Hintergrundsignals zur Kontrolle der Bildquelle

Tabelle 15: Ein- und Ausgabeelemente der Bedienoberfläche (GUI 1 – SDI-Quellenauswahl)

Eingabeelemente der Bedienoberfläche zur Latenzberechnung

Element	Typ	Standardwert	Funktion
Ergebnis in	Dropdown	Millisekunden	Festlegen der Anzeigeeinheit der Delay-Ergebnisse (Millisekunden oder Frames)
Start-Stop-Knopf	Button	Start	Starten bzw. Stoppen der laufenden Delay-Berechnung
Glättung aktiviert	Tickbox	aktiviert	Aktivieren bzw. Deaktivieren der geglätteten Anzeige des Delay-Werts
Glättung (EMA Alpha)	Fader	0,3	Einstellen des EMA-Glättungsfaktors Alpha (0–1): kleinere Werte glätten stärker, größere Werte reagieren schneller
Kalibrieren (je Vorder- und Hintergrund)	Button	/	Starten der Kalibrierung des Bewegungstriggers der jeweiligen Quelle, unabhängig von der anderen Seite und bereits vor „Start“ möglich
Schwellwert-Faktor (je Vorder- und Hintergrund)	Fader	2	Einstellen des Sicherheitsfaktors, mit dem der kalibrierte Rauschgrenzwert zum tatsächlich verwendeten Schwellwert der Bewegungserkennung multipliziert wird
Filter aktiv	Tickbox	aktiv	Aktivieren bzw. Deaktivieren des Butterworth-Tiefpassfilters 2. Ordnung
Cutoff (Hz)	Fader	3 Hz	Einstellen der Grenzfrequenz (Cutoff) des Tiefpassfilters

Element	Typ	Standardwert	Funktion
Normieren	Tickbox	aktiv	Aktivieren bzw. Deaktivieren der Normierung der gemessenen Geschwindigkeiten
Berechnungszeitraum (s)	Fader	3 s	Einstellen der Länge des Berechnungsfensters (Trigger-Fensters), über das jeweils ein Korrelationsdurchlauf zur Delay-Berechnung ausgeführt wird
Downscaling (Faktor)	Fader	0,5	Einstellen des Downscaling-Faktors für die Videoanalyse zur Reduzierung der Rechenlast des Trackings
Minimale Korrelationsgüte	Fader	0,5	Einstellen des Mindestwerts der Korrelationsgüte (r); Berechnungsfenster mit geringerer Korrelationsgüte werden verworfen
Max. angenommener Delay (s)	Fader	1 s	Einstellen der maximal angenommenen zeitlichen Verschiebung zwischen den beiden Videoquellen (begrenzt den Suchbereich der Kreuzkorrelation)

Tabelle 16: Eingabeelemente der Bedienoberfläche (GUI 2 – Delay-Berechnung)

Ausgabeelemente der Bedienoberfläche zur Latenzberechnung

Element	Typ	Werte / Bereich	Funktion
Delay (aktuell, geglättet oder ungeglättet)	Textausgabe	Auf drei Nachkommastellen gerundet, bzw. auf ganze Frames	Anzeige des zuletzt berechneten Delay-Werts in der gewählten Einheit; die Beschriftung wechselt je nach Einstellung „Glättung aktiviert“ zwischen „Delay:“ und „Delay (geglättet):“
Größtes / kleinstes gemessenes Delay	Textausgabe	Auf drei Nachkommastellen gerundet, bzw. auf ganze Frames	Anzeige des bisher größten und kleinsten gemessenen (ungeglätteten) Delay-Werts zur Eingrenzung des Delay-Bereichs
Richtung	Textausgabe	z. B. „Hintergrund hinkt hinterher“	Anzeige, welche der beiden Quellen der anderen zeitlich hinterherhinkt
Korrelationsgüte (r)	Textausgabe	Drei Nachkommastellen	Anzeige der Korrelationsgüte des zugrundeliegenden Berechnungsfensters als Maß für die Zuverlässigkeit des angezeigten Delay-Werts
Status (letztes Fenster)	Textausgabe	Gültig / Nicht gültig	Anzeige, ob das zuletzt verarbeitete Berechnungsfenster die Güteschwelle erreicht hat und der angezeigte

Element	Typ	Werte / Bereich	Funktion
			Delay-Wert somit aktuell ist
Ausgangsframerate	Textausgabe	z. B. 50 fps	Anzeige der aus der Quellenauswahl übernommenen Soll-Framerate, auf deren Basis der Delay-Wert in Frames umgerechnet wird
Status des Programms	Textausgabe	z. B. „Bereit.“ / „Messung läuft ...“ / „Gestoppt.“	Anzeige des aktuellen Programmstatus
Schwellwert (px/s, je Vorder- und Hintergrund)	Textausgabe	/	Anzeige des aktuell verwendeten Schwellwerts der Bewegungserkennung zur Kontrolle, ob die Kalibrierung sinnvolle Werte geliefert hat
Status der Bewegungstrigger (je Vorder- und Hintergrund)	Textausgabe	Kalibrierung nötig / Kalibrierung läuft / Bewegung erkannt / Still / Signalstatus der Quelle (z. B. „Kein SDI-Signal“)	Anzeige des aktuellen Bewegungstrigger-Status zur Einschätzung, ob eine Nachkalibrierung nötig ist; bei Signalverlust wird stattdessen der Quellenstatus rot hervorgehoben
Bildvorschau (je Vorder- und Hintergrund)	Videovorschau	/	Live-Bild der jeweiligen Quelle im Tab „Live-Vorschau“ zur Kontrolle der Bildquelle während der Messung

Element	Typ	Werte / Bereich	Funktion
Framerate, gemessen (je Vorder- und Hintergrund)	Textausgabe	/	Anzeige der tatsächlich gemessenen Framerate zur Kontrolle gegenüber der in der Quellenauswahl festgelegten Soll-Framerate
Geschwindigkeit, gemessen (je Vorder- und Hintergrund)	Textausgabe	/	Anzeige des aktuellen, rohen Geschwindigkeitsbetrags des Trackings; derselbe Wert, aus dem nach Glättung auch der Bewegungstrigger-Status abgeleitet wird

Tabelle 17: Ausgabeelemente der Bedienoberfläche (GUI 2 – Delay-Berechnung)

Anlage 3

Eingabeelemente der Bedienoberfläche zur Trackingstabilisierung

Element	Typ	Standardwert	Funktion
Eingehend IP-Adresse	Eingabefeld	0.0.0.0	Empfangsadresse des FreeD-Datenstroms
Eingehend Port	Eingabefeld	40000	Empfangsport
Rausgehend IP-Adresse	Eingabefeld	127.0.0.1	Zieladresse der stabilisierten Daten (UE)
Rausgehend Port	Eingabefeld	40002	Zielport, bewusst abweichend vom Empfangsport

Element	Typ	Standardwert	Funktion
Filter Kamera-ID	Eingabefeld / Dropdown	–	Auswahl der zu verarbeitenden Kamera (Mehrkamera-Setups)
Stabilisator umgehen	Checkbox	deaktiviert	Unbearbeitete Durchleitung der FreeD-Pakete
Stillstandskonfiguration	Button	–	Startet die Kalibrierung des Rauschbodens
Konfigurationsdauer	Eingabefeld	15 s	Dauer der Kalibrierung
Stillstandserkennung Start/Stop	Button	–	Aktiviert/deaktiviert die Hystereseerkennung
Oberer Schwellenwert	Fader	3,0	Obere Hystereseschwelle der Bewegungsenergie
Unterer Schwellenwert	Fader	2,2	Untere Hystereseschwelle der Bewegungsenergie
Eintrittsgrenze	Fader	0,3 s	Bestätigungszeit für Wechsel zu Still
Austrittsgrenze	Fader	0,1 s	Bestätigungszeit für Wechsel zu Moving
One-Euro-Filter aktivieren	Checkbox	deaktiviert	Aktiviert den in Kapitel 4.4 ausgewählten Filter
min_cutoff (fc_min)	Fader	0,1–5,0 Hz	Grundgrenzfrequenz
β	Fader, logarithmisch	0,0–1,0	Geschwindigkeitskoeffizient

Tabelle 18: Eingabeelemente Trackingstabilisierung

Ausgabeelemente der Bedienoberfläche zur Trackingstabilisierung

Element	Typ	Werte / Bereich	Funktion
Zustandsausgabe	Textanzeige	Still / Moving / Kalibrierung: Kamera stillhalten / Fehler: Erst kalibrieren	Aktueller Programmzustand
Kalibrierungsfortschritt	Fortschrittsbalken	0–100 %	Fortschritt der laufenden Kalibrierung
Paketrate	Anzeige	Pakete/s	Indikator für die Verbindungsqualität

Tabelle 19: Ausgabeelemente Trackingstabilisierung

9. Hilfsmittel

Zusammengeführte und sprachlich vereinheitlichte Übersicht der bei der Erstellung der Bachelorarbeit verwendeten Hilfsmittel.

Tool / Versions-nummer / Urheber	**Kapitel / Programm**	**Verwendung**	**Bemerkung**
Claude Code / Version 2.1.234 / Anthropic	3.4 / 1. Entwicklungsphase / Latenzberechnung	Überarbeitung der Analysefunktion	Die Analysefunktion wurde angepasst und in die bestehende Auswertung integriert.
Claude Code / Version 2.1.234 / Anthropic	3.4 / 1. Entwicklungsphase / Latenzberechnung	Überarbeitung der Programmstruktur	Das Programm wurde in logisch getrennte Dateien und Module aufgeteilt.
Claude Code / Version 2.1.234 / Anthropic	3.4 / 1. Entwicklungsphase / Latenzberechnung	Visualisierung der Ergebnisse	Funktionen zur grafischen Darstellung und Kontrolle der Ergebnisse wurden entwickelt.
Claude Code / Version 2.1.234 / Anthropic	3.4 / 1. Entwicklungsphase / Korrelationsverfahren- und Datensatztest	Entwicklung eines Vergleichsprogramms für Korrelationsverfahren und Datensätze auf Grundlage des bestehenden Codes	Ein Programm zum systematischen Vergleich der Verfahren, Datensätze und Kombinationen wurde erstellt.
Claude Code / Version 2.1.234 / Anthropic	3.5 / 2. Entwicklungsphase / check_resolution	Entwicklung eines Testprogramms zur Prüfung der Auflösung von Innen- und Außenkamera	Die tatsächlich verwendeten Kameraauflösungen konnten ermittelt und überprüft werden.
Claude Code / Version 2.1.234 / Anthropic	3.5 / 2. Entwicklungsphase / framerate_analyse	Entwicklung eines Testprogramms zur Prüfung der Framerate beider Kameras über 30 Sekunden	Höhe und Stabilität der Frameraten konnten überprüft werden.
Claude Code / Version 2.1.234 / Anthropic	3.5 / 2. Entwicklungsphase / threadbasierte Live-Latenzberechnung	Umsetzung der Threadlogik	Die Aufnahme und Verarbeitung beider Kameraströme wurde threadbasiert umgesetzt.
Claude Code / Version 2.1.234 / Anthropic	3.5 / 2. Entwicklungsphase / lineare und threadbasierte Live-Latenzberechnung	Prüfung und Anpassung des Downscalings	Das fehlerhafte Downscaling wurde überprüft und an die Kameraauflösungen angepasst.
DeckLink SDK / Version nicht dokumentiert / Blackmagic Design	3.5 / 2. Entwicklungsphase / DeckLink-Schnittstelle	Beispielcode als Grundlage der C++-Schnittstelle	Ein SDK-Beispiel zur synchronisierten Erfassung zweier SDI-Quellen diente als technische Grundlage.

Tool / Versions-nummer / Urheber	Kapitel / Programm	Verwendung	Bemerkung
Claude Code / Version 2.1.234 / Anthropic	3.5 / 2. Entwicklungsphase / DeckLink-Schnittstelle	Entwicklung der DeckLink-Schnitt-stelle auf Grund-lage des SDK	Die Schnittstelle zur Erfassung und Über-gabe der SDI-Bilddaten wurde entwickelt.
Claude Code / Version 2.1.234 / Anthropic	3.6 / 3. Entwicklungsphase / GUI der Latenzberech-nung	Entwicklung einer GUI anhand vor-gegebener Ein- und Ausgabefel-der sowie Skizzen	Die GUI wurde um-gesetzt und anschlie-ßend manuell funkti-onal und gestalte-risch angepasst.
Claude Code / Version 2.1.234 / Anthropic	3.6 / 3. Entwicklungsphase / SDI-Latenzberechnung	Zusammenfüh-rung der Pro-gramme aus Phase 2 mit der GUI	Die Programmteile wurden zu einer SDI-Anwendung zusam-mengeführt; weitere Anpassungen waren bis zur vollständigen Funktion erforderlich.
Claude Code / Version 2.1.234 / Anthropic	3.6 / 3. Entwicklungsphase / SDI-Latenzberechnung	Trennung der Programmteile in Controller, Engi-nes und GUI	Die Programmstruk-tur wurde in klar ab-gegrenzte Kompo-nenten gegliedert.
Claude Code / Version 2.1.234 / Anthropic	3.6 / 3. Entwicklungsphase / SDI-Latenzberechnung	Einbau eines Be-wegungstriggers nach vorgegebe-ner Logik	Der Bewegungstrig-ger wurde in den Messablauf inte-griert.
Claude Code / Version 2.1.234 / Anthropic	3.6 / 3. Entwicklungsphase / SDI-Latenzberechnung	Taktung der Ver-arbeitung auf 50 fps	Die Verarbeitung wurde auf die Bild-rate der SDI-Signale abgestimmt.
Claude Code / Version 2.1.234 / Anthropic	3.6 / 3. Entwicklungsphase / SDI-Latenzberechnung	Einbau einer Er-kennung für Frameverluste und fehlerhafte Zeitbasen	Frameverluste und inkonsistente Zeitin-formationen werden gemeldet; betroffene Messungen werden verworfen.
Claude Code / Version 2.1.234 / Anthropic	4.4 / 1. Entwicklungsphase / Filtermessungen	Erweiterung des Testprogramms auf mehrere Vi-deos und Filter-stufen	Eine Logik zum auto-matisierten Einlesen, Analysieren und Ver-arbeiten mehrerer Vi-deos wurde umge-setzt.
Claude Code / Version 2.1.234 / Anthropic	4.4 / 1. Entwicklungsphase / Filtermessungen	Übernahme der Delayberechnung aus der ersten Entwicklungs-phase der La-tenzberechnung	Das Verfahren wurde in das Filtertestpro-gramm integriert, um Signalverschiebun-gen zwischen Roh- und Filterdaten zu bestimmen.
Claude Code / Version 2.1.234 / Anthropic	4.4 / 1. Entwicklungsphase / Filtermessungen	Ausgabe der Er-gebnisse als Text und Diagramme	Die textuelle und grafische Ergebnis-ausgabe wurde um-gesetzt.

Tool / Versionsnummer / Urheber	Kapitel / Programm	Verwendung	Bemerkung
ChatGPT / GPT-5.6 / OpenAI	4.5 und 4.6 / 2. Entwicklungsphase / Stillstandserkennung	Ideensammlung und Einordnung von Ansätzen zur Zustandssteuerung der Bewegungserkennung	Ein Konzept für Kalibrierung, gleitende Auswertung, Hysterese und Zustandsumschaltung wurde erarbeitet und anschließend eigenständig umgesetzt und geprüft.
FreeD Tracker / Version 1.3 / Ivan Ryzhak	4.5 und 4.6 / 2. Entwicklungsphase / Stillstandserkennung	Verwendung eines iPhones als FreeD-Datenquelle für Programmtests	Die App ermöglichte die einfache Nutzung des iPhones als FreeD-Sender.
TD-FreeD (GitHub-Quellcode) / Exscriber (2020)	4.5 und 4.6 / 2. Entwicklungsphase / FreeD-Verarbeitung	Anhaltspunkt für die Entwicklung einer FreeD-Bridge	Die frei verfügbare TouchDesigner-/Python-Implementierung diente als Orientierung für das Einlesen und Verarbeiten von FreeD-Paketen.
Claude Code / Version 2.1.234 / Anthropic	4.5 und 4.6 / 2. Entwicklungsphase / Stillstandserkennung	Überarbeitung der Kalibrierung	Die Kalibrierung wurde so angepasst, dass die höchsten 5 % der Geschwindigkeitswerte vor der RMS-Berechnung ausgeschlossen werden.
Claude / Version nicht dokumentiert / Anthropic	4.6 / 3. Entwicklungsphase / Stillstandserkennung	Analyse der FreeD-Pakete der VIVE Mars	Es wurde festgestellt, dass sich die Trackingwerte im Stillstand seltener ändern als Pakete übertragen werden.
Claude Code / Version 2.1.234 / Anthropic	4.6 / 3. Entwicklungsphase / Stillstandserkennung	Prüfung, ob genügend gültige Trackingdaten für die Kalibrierung vorliegen	Die Ursache der zuvor nicht funktionsfähigen Kalibrierung wurde identifiziert und behoben.
Claude Code / Version 2.1.234 / Anthropic	4.6 / 3. Entwicklungsphase / OneEuroFiltertest	Entwicklung eines Testprogramms zur Bestimmung geeigneter One-Euro-Filter-Parameter	Das Programm wurde erstellt; Messdaten und Vergleichswerte konnten automatisiert ausgegeben werden.
Claude Code / Version 2.1.234 / Anthropic	4.6 / 3. Entwicklungsphase / GUI der Stillstandserkennung	Entwicklung einer GUI anhand vorgegebener Ein- und Ausgabefelder sowie Skizzen	Die GUI wurde entsprechend den Vorgaben umgesetzt und anschließend manuell angepasst.

Tool / Versionsnummer / Urheber	Kapitel / Programm	Verwendung	Bemerkung
Claude Code / Version 2.1.234 / Anthropic	4.6 / 3. Entwicklungsphase / Stillstandserkennung	Zusammenführung des Programms mit der GUI	Stillstandserkennung, Filterung, FreeD-Empfang und GUI wurden zu einer Anwendung zusammengeführt.
Claude Code / Version 2.1.234 / Anthropic	Alle Entwicklungsphasen / Alle Programme	Vereinheitlichung der Programm-Header, Kommentare und Variablennamen sowie Prüfung der Namenskonventionen	Die Programme wurden für die Abgabe strukturell vereinheitlicht; die Änderungen wurden anschließend geprüft.
Claude Code / Version 2.1.234 / Anthropic	Alle Entwicklungsphasen / Alle Programme	Erstellung von README- und Requirements-Dateien	Für die Programmordner wurden Beschreibungen, Startanweisungen und Abhängigkeiten dokumentiert.
LanguageTool / Version nicht dokumentiert / LanguageTooler GmbH	Gesamter Text	Prüfung von Rechtschreibung, Grammatik und Satzbau	Erkannte Fehler wurden geprüft und anschließend im Text korrigiert.
ChatGPT / GPT-5.6 / OpenAI	Gesamter Text	Suche nach Kürzungspotenzial sowie Prüfung von Formulierungen und Begriffskonsistenz	Vorschläge und Fundstellen wurden eigenständig geprüft und überarbeitet.
Perplexity / Sonar 2 / Perplexity AI	Gesamter Text	Quellenrecherche	Das Werkzeug wurde zur Suche nach potenziell geeigneten Quellen eingesetzt; die Quellen wurden anschließend eigenständig geprüft.
ChatGPT / GPT-5.6 / OpenAI	Gesamter Text	Quellenrecherche und Prüfung des Quellenverzeichnisses	Quellenvorschläge, fehlende Angaben und mögliche Dubletten wurden anschließend eigenständig überprüft.
Claude / Claude Sonnet 5 / Anthropic	Gesamter Text	Quellenrecherche	Das Werkzeug wurde zur Suche nach potenziell geeigneten Quellen eingesetzt; die Quellen wurden anschließend eigenständig geprüft.

Tool / Versionsnummer / Urheber	Kapitel / Programm	Verwendung	Bemerkung
Google Übersetzer / Version nicht dokumentiert / Google	Gesamter Text	Orientierende Übersetzung englischsprachiger Quellen	Die Übersetzung wurde zum ersten inhaltlichen Überblick verwendet; maßgebliche Aussagen wurden anhand der Originalquellen geprüft.
ChatGPT / GPT-5.6 / OpenAI	Gesamter Text	Orientierende Übersetzung englischsprachiger Fachartikel	Die Übersetzung wurde zum ersten inhaltlichen Überblick verwendet; maßgebliche Aussagen wurden anhand der Originaltexte geprüft.
ChatGPT / GPT-5.6 / OpenAI	Gesamter Text	Prüfung auf Kontinuitätsfehler und wechselnde Bezeichnungen	Die Prüfung war nur teilweise zuverlässig; Fundstellen wurden eigenständig kontrolliert und korrigiert.
draw.io / Version 31.1.8 / JGraph Ltd	Abbildung 1	Erstellung eines Diagramms zum Signalfluss im Virtual-Production-Testaufbau	Das Diagramm wurde mit der Software erstellt und manuell an den Versuchsaufbau angepasst.
ChatGPT / GPT-5.6 / OpenAI	Kapitel 3.4 / 1. Entwicklungsphase / Excel-Auswertung	Aufbereitung der Datensätze und Berechnung von Durchschnitt, mittlerer Abweichung, durchschnittlicher und minimaler Korrelationsgüte, Mittelwert, Median und Standardabweichung der Konsistenz sowie durchschnittlicher Güte und Rechenzeit	Die berechneten Übersichten wurden für den Vergleich von Korrelationsverfahren und Trackingdatensätzen verwendet und anschließend geprüft.
ChatGPT / GPT-5.6 / OpenAI	Kapitel 4.4 / 1. Entwicklungsphase / Excel-Auswertung	Aufbereitung der Datensätze und Berechnung des absoluten Delays sowie der prozentualen Reduktion von Schwankungen und Drift gegenüber den ungefilterten Daten	Die berechneten Übersichten wurden für den Filtervergleich verwendet und anschließend geprüft.

Tool / Versionsnummer / Urheber	Kapitel / Programm	Verwendung	Bemerkung
ChatGPT / GPT-5.6 / OpenAI	Kapitel 4.6 / 3. Entwicklungsphase / Excel-Auswertung	Aufbereitung der Datensätze und Berechnung von durchschnittlichem Delay und durchschnittlicher Korrelation je Parametersatz, Reduktion der Standardabweichung, Peak-Erhaltung sowie Abweichung zwischen Roh- und Filterdaten	Die berechneten Übersichten wurden für den Vergleich der One-Euro-Filter-Parametersätze verwendet und anschließend geprüft.
ChatGPT / GPT-5.6 / OpenAI	Kapitel 3 bis 6	Sprachliche Überarbeitung und Formulierungsvorschläge für einzelne Abschnitte	Die Vorschläge wurden anhand der eigenen Ergebnisse, Messungen und Aussagen ausgewählt, angepasst und in die Arbeit übernommen.
ChatGPT / GPT-5.6 / OpenAI	Kapitel 5 und 6	Unterstützung bei der Ausformulierung von Evaluation und Fazit auf Grundlage vorgegebener Stichpunkte und vorhandener Kapitel	Die erzeugten Textvorschläge wurden inhaltlich geprüft, gekürzt und eigenständig an die Arbeit angepasst.
ChatGPT / GPT-5.6 / OpenAI	Kurzfassung und Abstract	Kürzung der deutschen Kurzfassung und Übersetzung ins Englische	Kurzfassung und Übersetzung wurden inhaltlich geprüft und eigenständig überarbeitet.

10. Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Analyse und Optimierung von Tracking- und Synchronisationsprozessen

in Virtual-Production-Workflows

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.



Datum