Diplomarbeit

cand. ing. Thomas Vierjahn

Verbesserte kamerabasierte Bilderkennung für interaktive Installationen und Spiele

FH D
Fachhochschule Düsseldorf
University of Applied Sciences

FB 5
Fachbereich Medien

[rmh]²
new media gmbh

Diplomarbeit

Verbesserte kamerabasierte Bilderkennung für interaktive Installationen und Spiele

cand. ing. Thomas Vierjahn

1. August 2007

Betreut durch:
Professor Dr.-Ing. Sina Mostafawy
Dipl.-Inform. Jérôme Thoma

Vorgelegt zur Erlangung des Grades eines Diplom-Ingenieurs
im Fachbereich Medien an der Fachhochschule Düsseldorf
durch cand. ing. Thomas Vierjahn

Angefertigt im Fachbereich Medien der Fachhochschule Düsseldorf
mit Unterstützung von [rmh] new media gmbh in Köln

Betreut durch:
Professor Dr.-Ing. Sina Mostafawy und
Dipl.-Inform. Jérôme Thoma

Fachhochschule Düsseldorf
Fachbereich Medien
Josef-Gockeln-Straße 9
40474 Düsseldorf

http://www.medien.fh-duesseldorf.de

[rmh] new media gmbh
Gilbachstraße 29a
50672 Köln

http://www.rmh.de

Diese Diplomarbeit wurde mit LaTeX 2ε unter Verwendung von
KOMA-Script, AMS-LaTeX und BibTeX gesetzt.

© Copyright 2007 by Thomas Vierjahn (tom@vierjahn.de)

Diese Diplomarbeit ist urheberrechtlich geschützt.
Alle Rechte vorbehalten.

Zusammenfassung

Für moderne interaktive Anwendungen wird es immer wichtiger, einen Benutzer durch zusätzlichen Ballast so wenig wie möglich einzuschränken. Daher bietet sich eine kamerabasierte Interaktionserkennung an. Viele existierende Verfahren benötigen dazu aber einen weitestgehend statischen Hintergrund. In der gegebenen Anwendung allerdings befindet sich im Sichtbereich der Kameras eine Projektion bewegter Inhalte, mit denen ein Benutzer interagieren kann.

Im Rahmen dieser Arbeit sollte ein bestehendes, auf Infrarotlicht basierendes System verbessert werden, das sich bisher als beleuchtungsabhängig erwiesen hat. Dazu wurden zunächst mehrere Verfahren zur Trennung von Vorder- und Hintergrund auf ihre Tauglichkeit untersucht. Das favorisierte Verfahren sollte anschließend durch stereoskopische Bildaufnahme – und damit tiefenbasierte Trennung – so verbessert werden, dass der bewegte Hintergrund sicher unterdrückt wird. Dies erwies sich für die gewünschten Anwendungen zwar als nicht praktikabel, dennoch wurden mögliche andere Anwendungsbereiche gefunden. Auch die Untersuchung der einzelnen Trennungsverfahren lieferte ein Ergebnis, das – mit weiteren Tests – zur Verbesserung der bestehenden Installation beitragen kann.

Abstract

Modern interactive environments are tending to not impose any restrictions by additional encumbrance on the user. Therefore it seems reasonable to utilize computer vision for detecting interactions. There are many techniques available to segment an image into fore- and background, but most of them require the scene to be static. In contrast this work applies to installations where an interactive projection is in the field of vision of a camera.

Within the bounds of this work an existing technique utilizing infra-red light, which has been shown to be illumination-variant, was to be improved. First several techniques for foreground-background-segmentation were checked for suitability. The most appropriate was to be refined by adding depth segmentation to suppress the animated background utilizing stereo-vision. This turned out to be impractical for the given installations, but more practicable ones were found. In addition to this even the favoured segmentation-technique may provide an improvement on the existing one.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Abkürzungsverzeichnis

CV ..... Computer-Vision

FBO ..... Framebuffer-Object
FDS ..... Fast Depth Segmentation

GLSL ..... OpenGL Shading Language
GPU ..... Grafic processing unit

HCI ..... Human-Computer-Interface
HMD ..... Head-Mounted-Display

IR ..... Infrarot

LUT ..... Look-Up-Table

POS ..... Point of Sale

URL ..... Uniform Resource Locator

VR ..... Virtual-Reality

1 Kamerabasierte berührungslose Interaktion – ein Überblick

Es begann mit *„Critter“*: Myron W. Krueger erforschte seit 1969 Interaktionsmöglichkeiten mit grafischen Umgebungen (Krueger *et al.*, 1985). Dabei entstand 1970 die Installation *„Videoplace“*, in der grafische Objekte mit dem Real-Bild des Betrachters kombiniert wurden. Krueger beschreibt ein „unwiderstehliches Verlangen“ der Betrachter, „das Objekt zu berühren“ sowie die Erwartung, dass „die Berührung die grafische Welt beeinflusse“. So entstand *„Critter“* – eine computergenerierte Figur, die mit dem Real-Bild des Betrachters interagieren konnte.

1.1 Videoplace

Für Echtzeit und *Virtual-Reality* (*VR*)-Installationen sowie interaktive Spiele werden neue Formen des *Human-Computer-Interface* (*HCI*) immer wichtiger. Diese Erkenntnis ist in der Literatur schon früh zu finden: Krueger erwartete, dass sich die Interaktion mit Computern weg von programmier-ähnlichen hin zu natürlichen, wahrnehmungsgestützten Verfahren entwickeln würde (Krueger *et al.*, 1985). Sutherland entwickelte beispielsweise ein stereoskopisches *Head-Mounted-Display* (*HMD*), das dem Benutzer einen von seiner Position abhängigen, perspektivisch korrekten Blick auf eine Szene ermöglichte (Sutherland, 1968). Dagegen entstand im Projekt *„GROPE“* ein haptisches Interface zur Darstellung von Kraftfeldern interagierender Protein-Moleküle (Batter und Brooks, Jr., 1971; Brooks, Jr. *et al.*, 1990).

Ebenso wie die 1971 von Myron W. Krueger entwickelte Installation *„Psychic Space“*, bei der die Position des Betrachters mittels Drucksensoren erfasst wurde (Krueger *et al.*, 1985), sind diese Interfaces alle kabelgebunden. Ein rein optisches Verfahren entwickelte Krueger 1970 mit *„Videoplace“*. *„Critter“* stellte dabei eine frühe Anwendung für diese Installation dar (Krueger *et al.*, 1985): Eine kleine, computergenerierte Figur interagiert mit dem Bild des Betrachters und reagiert auf dessen Bewegungen. Das Computersystem analysiert die Silhouette des Menschen und steuert das Verhalten von *„Critter“* abhängig von der Armhaltung des Betrachters (Abbildung 1.1).

Krueger beschreibt in „Artificial Reality“ die Anwendung von *„Videoplace“* für Video-Konferenzen, bei denen die Teilnehmer gleichzeitig miteinander mit denselben grafischen Objekten interagieren können (Krueger, 1983). Darüber hinaus sieht er Nutzungsmöglichkeiten als Interface für Spiele oder computergestützte Lehre (Krueger *et al.*, 1985).

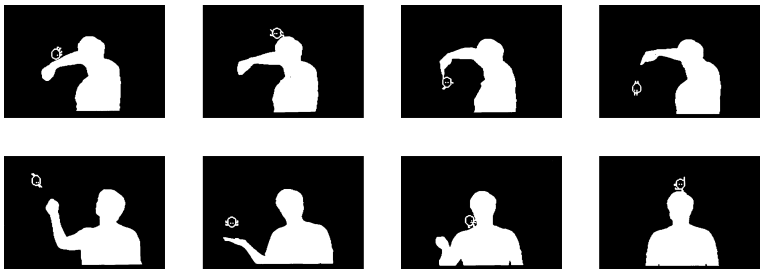


Abbildung 1.1: *„Critter“*: Eine interaktive Anwendung für *„Videoplace“* (aus: Krueger *et al.*, 1985)

1.2 Sehende Maschinen

Der von Krueger vorgeschlagene Wechsel weg von der Programmierung hin zu natürlicheren Interaktionsformen (Krueger *et al.*, 1985) setzt sich in späteren Arbeiten fort. Der von Pierre Wellner entwickelte *„DigitalDesk“* ergänzt einen Büro-Schreibtisch um elektronische Elemente, anstatt den elektronischen Computer-Desktop mehr und mehr an einen realen anzunähern (Wellner, 1991). Auf diese Weise ist die Interaktion mit realen Dokumenten und Objekten identisch mit der Interaktion mit computergenerierten Dokumenten und Objekten: Der projizierte virtuelle Taschenrechner wird beispielsweise wie sein reales Pendant durch Berühren der Tasten bedient. Auch ein Wechsel zwischen realer und virtueller Ebene kann nahtlos erfolgen: Ein reales Dokument kann mithilfe des in den Tisch eingebauten Scanners in die Projektion übernommen werden, ein virtuelles Dokument kann – an dieselbe Stelle verschoben – ausgedruckt werden. Ein weiterer großer Vorteil dieses *HCI* gegenüber einer Computermaus ist, dass die beschriebene Installation keinen Platz auf dem Schreibtisch benötigt.

Ebenfalls auf *Computer-Vision* (*CV*) basierende Interaktion wurde von Wren *et al.* untersucht (Wren *et al.*, 1998). Auch hier wird die Notwendigkeit natürlicher Interfaces deutlich hervorgehoben. Wren *et al.* vertreten aber die Ansicht, dass kabelgebundene Sensoren wie Datenhandschuh, *HMD* u. a. keine natürlichen Interfaces darstellen. Diese Geräte beeinträchtigen die vom Menschen aus der Realität gewohnte Interaktion mit dreidimensionalen Umgebungen durch zusätzlichen Ballast und Einschränkung der Beweglichkeit. Stattdessen wurde versucht, mit *„Perceptive Spaces“* eine natürliche virtuelle Umgebung zu schaffen, die darüber hinaus auch keine besonderen Randbedingungen wie spezielle Kleidung o. ä. erfordert. Die einzigen Bedingungen, die gestellt wurden, waren „konstante Beleuchtung und unbewegter Hintergund“ (Wren *et al.*, 1998).

Eine Anwendungsmöglichkeit dieser *„Perceptive Spaces“* stellt *„City of News“* dar (Sparacino *et al.*, 1997; Wren *et al.*, 1998) (Abbildung 1.2). Dabei handelt es sich um einen Webbrowser, der *URL*s in Gebäude einer virtuellen Stadt umsetzt. Er nutzt die Fähigkeit des Menschen, sich anhand von Landmarken und auf Plänen zu orientieren. Benutzer dieses Browsers navigieren so von einem Gebäude zum anderen. Die gewünschte *Website* wird durch Zeigen oder Sprachkommandos geöffnet. Ebenso erfolgt das *Scrollen* durch den Inhalt über Gesten oder Sprache. Ein ähnliches *HCI* untersuchten Bérard und Crowley *et al.* (Bérard, 1999b; Crowley *et al.*, 2000): Das *„Perceptual Window“* erlaubt Scrollen in

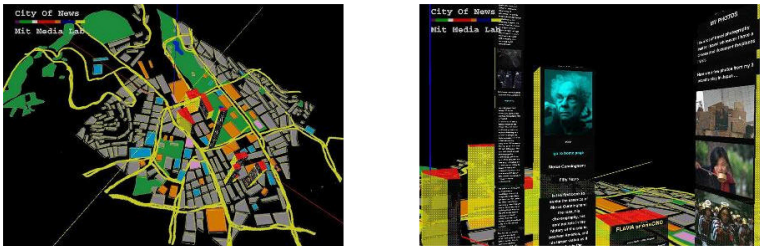


Abbildung 1.2: *„City of news“*: Anwendungsbeispiel für *„Perceptive Spaces“* (aus: Sparacino *et al.*, 1997)

Dokumenten mittels Kopf- und Mausbewegungen (siehe Bérard, 1999c).

An den *„DigitalDesk“* von Wellner ist das *„MagicBoard“* – je nach Anwendung auch *„Magic Table“* genannt – angelehnt (Bérard, 2003; Crowley *et al.*, 2000). Mit diesem Projekt wird ein White-Board um elektronische Funktionalität ergänzt. Mittels *CV* können Zeichnungen und Texte gescannt und anschließend neu angeordnet projiziert werden (siehe Bérard, 1999a). Im Gegensatz zu elektronischen White-Boards gibt es beim *„MagicBoard“* keine Einschränkungen zu normalen White-Boards wie beispielsweise geringe Eingabe-Geschwindigkeit oder Benutzeranzahl.

Um solche Interaktionen implementieren zu können, bedarf es allerdings keiner sehr aufwändigen Algorithmen. In Abhängigkeit von der Anwendung können bestimmte Beschränkungen definiert werden, die es ermöglichen, schnelle und einfache *CV*-Algorithmen einzusetzen. Eines von mehreren Beispielen, die von Freeman *et al.* zu diesem Thema vorgestellt werden, ist ein Computerspiel, das zu einem bestimmten Zeitpunkt erwartet, dass der Spieler läuft (Freeman *et al.*, 2000). Dies führt zu der Beschränkung, dass der Algorithmus lediglich die Laufgeschwindigkeit bestimmen muss – eine relativ einfach zu implementierende Aufgabe.

1.3 Ein Kinderspiel

Im *MIT Media Laboratory* entstand eine besondere interaktive Umgebung: der *„KidsRoom“* (Bobick *et al.*, 1999). Ziel dieses Projektes war es, eine Umgebung zu schaffen, die genau auf die Bedürfnisse von Kindern zugeschnitten ist. Der *„KidsRoom“* war einem Kinderzimmer nachempfunden – lediglich zwei Wände waren durch Rückprojektions-Leinwände ersetzt. Bis zu vier Kinder konnten sich in dieser für sie gewohnten Umgebung frei bewegen, miteinander spielen und an einer interaktiven Geschichte teilnehmen. Um die Kinder nicht zu beeinträchtigen, ist es hier besonders wichtig, auf *CV*-Technologien zurückzugreifen, anstatt die Kinder mit Datenhandschuh, spezieller Kleidung, Markern und Kabeln zu belasten. Bei dieser Installation lieferten drei Kameras die für die Interaktion nötigen Daten. Eine Kamera wurde als Top-Kamera zum Tracking der Personen benutzt (Abbildung 1.3), die beiden anderen dienten zur Gesten-Erkennung an zwei besonders markierten Punkten im Raum.

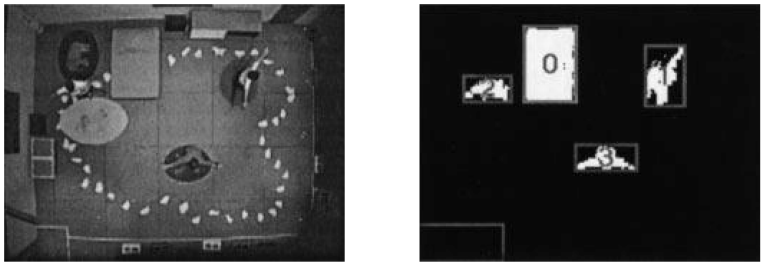


Abbildung 1.3: *„KidsRoom“*: Top-View und Tracking-Maske (aus: Bobick *et al.*, 1999)

Höysniemi *et al.* erforschten für das Spiel *„QuiQui's Giant Bounce“* (Hämäläinen, 2002) die Gesten von Kindern zur Steuerung (Höysniemi *et al.*, 2005). Eine einfache Webcam an einem Standard-PC ermöglicht hier die berührungslose Interaktion. Auch an die Umgebung werden keine speziellen Anforderungen gestellt (siehe Hämäläinen *et al.*, 2003).

Eine ebenfalls von Hämäläinen *et al.* untersuchte Anwendung *CV*-basierter Interfaces stellen *Video Mirrors* dar, die beispielsweise mit der Applikation *„Kick Ass Kung-Fu“* für Kampfsport-Training genutzt werden können (Hämäläinen *et al.*, 2005). Bei dieser Installation steht der Benutzer zwischen zwei Projektionsleinwänden, und sein Bild wird im Profil in die virtuelle Szene integriert. Auch dieses Spiel ist auf Standard-Hardware mit einer einfachen Webcam lauffähig.

Als weitere Beispiele für kamerabasierte Verfahren sind das *„EyesWeb Project“* (Camurri *et al.*, 2004) und *„Robot's Play“* (Brooks *et al.*, 2004) zu nennen. Ersteres untersucht Interaktion aufgrund von Gesten und Bewegungen in Verbindung mit Musik, Letzteres untersucht Interaktionsmöglichkeiten mit einem spielenden Roboter. Des Weiteren untersucht Rakkolainen das Tracking von Personen durch einen für Infrarotlicht transparenten Projektionsschirm (Rakkolainen, 2006); Eisenstein und Mackay vergleichen zwei *CV*-basierte Auswahltechniken (Eisenstein und Mackay, 2006).

Zuletzt dürfen natürlich auch kommerzielle Produkte wie *„Me2Cam“* von Intel und Mattel (D'Hooge und Goldsmith, 2001) sowie das sehr erfolgreiche *„EyeToy“* von Sony (Sony, 2003) als Beispiele für interaktive, kamerabasierte Anwendungen nicht vergessen werden.

Ein sehr großes Forschungsfeld für kamerabasierte Bilderkennung, die aber nicht in erster Linie als *HCI* dient, ist die Videoüberwachung. Hier finden sich sehr viele und sehr gute Ansätze im Hinblick auf Trennung von Vorder- und Hintergrund.

Die Entwicklung der Hardware im Verlauf der vergangenen Jahre ermöglicht es, *CV*-basierte interaktive Installationen mit immer geringerem materiellen und damit kleinerem finanziellen Aufwand zu erstellen. Während Krueger *et al.* für *„Videoplace“* noch eine *VAX 11/780* sowie eine „Gruppe eng verbundener, dedizierter Prozessoren auf einer spezialisierten Bus-Struktur“ benötigte (Krueger *et al.*, 1985), liegen die Anforderungen der zuvor genannten *„QuiQui's Giant Bounce“* (Hämäläinen, 2002), *„EyeToy“* (Sony, 2003) und *„Kick Ass Kung-Fu“* (Hämäläinen *et al.*, 2005) bei nur einem einfachen Prozessor und einer einfachen Webcam. Ebenso verhält es sich mit den Kosten: In einem internen Memo der *Bell Labs* wird der Preis für eine *VAX 11/780* auf 241.255 $ beziffert (London und Reiser, 1978), während die Preisempfehlung des Herstellers für eine *„Playstation 2“*

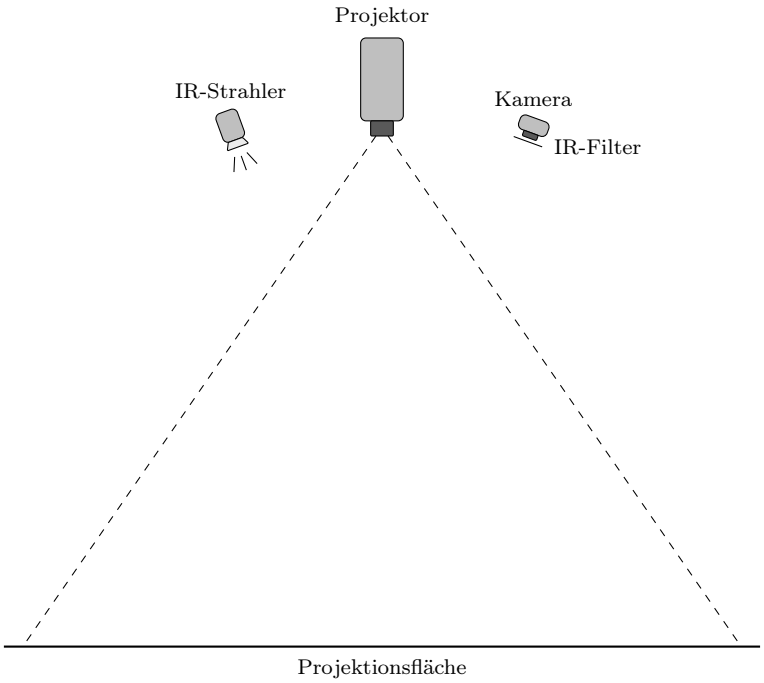


Abbildung 1.4: Aufbau der bisherigen Installation

129,95 € – also etwa 176 $ – beträgt (Sony, 2006).

1.4 Aufgabenstellung

Die Firma *rmh* produziert interaktive Spiele beispielsweise für Messen oder *Point of Sale* (*POS*)-Installationen, die auf einem Standard-PC laufen. Die Spielfläche wird dabei auf den Boden oder eine Wand projiziert, und die Bewegungen der Spieler werden mit einer Webcam aufgenommen. Zur Trennung von Vorder- und Hintergrund wird ein Bild des leeren Hintergrundes aufgenommen. Da die sich mit dem Spielverlauf ändernde Projektion prinzipbedingt ebenfalls im Sichtbereich der Kamera liegt, erhält die Kamera einen Filter, sodass nur noch nahes Infrarot (*IR*)-Licht aufgenommen wird. So ist die Projektion im Kamerabild nicht enthalten. Gleichzeitig wird die Szene mit einem *IR*-Scheinwerfer beleuchtet (Abbildung 1.4).

Die Trennung von Vorder- und Hintergrund erfolgt durch Differenzbildung des aktuellen Kamerabildes gegen den gespeicherten Hintergrund. Auf diese Differenz wird ein Schwellwert (*Threshold*) anhand einer für die Installation ermittelten Gauß'schen Wahrscheinlichkeitsverteilung angewendet und somit eine binäre Maske des Spielers erzeugt. Diese wird anschließend mit morphologischen Operatoren bereinigt.

Das oben beschriebene Verfahren ist zwar prinzipiell zur Spielsteuerung geeignet, es

treten aber folgende Probleme auf:

- Beleuchtungsabhängigkeit:
  Zu Beginn wird ein *IR*-Bild des leeren Hintergrundes aufgenommen. Ändert sich später die Beleuchtungsintensität im nahen *IR*-Bereich z. B. durch Sonneneinstrahlung, kommt es zu Veränderungen, die fälschlicherweise als Vordergrundobjekte interpretiert werden. Insbesondere umstehende Personen können von außen einen starken, sich bewegenden *IR*-Schatten auf die Spielfläche werfen, der ebenfalls als Spieler interpretiert wird.
- Kleidungsabhängigkeit:
  Bestimmte Materialien reflektieren *IR*-Licht anders als sichtbares Licht. Beispielsweise erscheint schwarzes Synthetik-Material im *IR*-Bild weiß und wird nicht mehr oder nur unzulänglich als Vordergrundobjekt erkannt.
- Manuelle Kalibrierung:
  Zur Kalibrierung der Installation müssen vier Marker auf die Spielfläche aufgebracht werden. Die Position wird zwar vorgegeben, die Marker können aber vom Benutzer nicht pixelgenau positioniert werden.

Im Rahmen dieser Diplomarbeit werden verschiedene Verfahren für die Trennung von Vorder- und Hintergrund recherchiert und auf ihre Tauglichkeit für die beschriebenen Installationen untersucht. Dabei soll insbesondere Abhilfe für o. g. Probleme geschaffen und das gewählte Verfahren zu Test-Zwecken sowohl als idealisierte Labor-Simulation als auch unter realen Bedingungen implementiert werden. Da die beschriebenen Spiele im niedrigen Preissektor angeordnet sind, soll aus wirtschaftlichen Gründen weiterhin nur Standard-Hardware verwendet werden – also ein handelsüblicher Computer und einfache Webcams.

1.5 Gliederung der Arbeit

Die in Abschnitt 1.4 aufgezeigten Anforderungen an die Verbesserung des bestehenden Verfahrens – sichere beleuchtungsunabhängige Maskierung eines Spielers vor bewegtem Hintergrund – legen ein zweigeteiltes Vorgehen nahe, an dem sich auch diese Arbeit orientiert:

- Kapitel 2 befasst sich mit monoskopischen Verfahren zur Trennung von Vorder- und Hintergrund. Dabei wird ein beleuchtungsunabhängiges Verfahren gesucht. Dieses Kapitel ist wie folgt weiter untergliedert:
  - Abschnitt 2.1 definiert formal die zuvor genannte Problemstellung.
  - In Abschnitt 2.2 werden einzelne Trennungsverfahren vorgestellt.
  - Abschnitt 2.3 beschreibt die Entwicklung einer geeigneten Software, um alle Verfahren vergleichbar testen zu können. In ihr wurden die beschriebenen Verfahren selbst implementiert.
  - Danach folgen in Abschnitt 2.4 die visuell erhaltenen Testergebnisse.
  - Abschließend für diesen Teil werden in Abschnitt 2.5 die Ergebnisse im Hinblick auf die Problemstellung diskutiert.

- Kapitel 3 befasst sich mit dem Problem des im Kamerabild sichtbaren bewegten Hintergrundes. Es ist wie folgt untergliedert:
  - Zunächst wird in Abschnitt 3.1 die vorliegende Problemstellung erläutert.
  - Anschließend wird in Abschnitt 3.2 ein stereoskopisches Verfahren vorgestellt und weiterentwickelt, das eine Lösung für dieses Problem darstellen könnte.
  - Abschnitt 3.3 stellt die Entwicklung einer geeigneten Software vor, die es ermöglicht, das Verfahren sowohl im idealisierten Modell als auch unter realen Bedingungen zu testen.
  - In Abschnitt 3.4 wird die Untersuchung der Genauigkeit des vorgestellten alten sowie des weiterentwickelten Verfahrens beschrieben.
  - Die Ergebnisse aus dieser Untersuchung und den Tests werden in Abschnitt 3.5 dargestellt.
  - Zuletzt werden in Abschnitt 3.6 die Ergebnisse dieses Verfahrens für das Problem des bewegten Hintergrundes diskutiert.
- Das diese Arbeit abschließende Kapitel 4 diskutiert die einzelnen Ergebnisse der beiden Kapitel 2 und 3 auf dem Hintergrund einer möglichen Kombination beider Verfahren, um die in Kapitel 1 aufgezeigten Probleme zu lösen. Weiterhin werden in diesem Kapitel Ausblicke auf sowie Anforderungen an zukünftige Untersuchungen gegeben, die auf dieser Arbeit basieren.

2 Trennung von Vorder- und Hintergrund

Die in Kapitel 1 aufgeführten Anwendungen – insbesondere die in Abschnitt 1.4 beschriebene Installation – benötigen ein Verfahren, das den Spieler oder Benutzer von anderen Inhalten im Kamerabild trennen kann. Diese Trennung lässt sich allgemein als eine Unterscheidung zwischen Vorder- und Hintergrund beschreiben, welche sich prinzipiell auf die Erkennung einer Bildveränderung zurückführen lässt: Sich ändernde Bildelemente stellen den Vordergrund oder Benutzer, unveränderte Elemente den Hintergrund dar.

Verfahren zu diesem Zweck finden sich in verschiedenen Anwendungsbereichen wie Video-Überwachung oder bildgebenden Methoden in der Medizin. Sie reichen von einfacher Differenzbildung aus zwei Bildern (Jain, 1981) bis zu den aufwendigeren Algorithmen, die mittels statistischer Methoden prüfen, ob ein bestimmtes Pixel wahrscheinlicher Hinter- oder Vordergrund darstellt. Grundlage für letztere Verfahren ist die Modellierung und ständige Aktualisierung eines Hintergrundes (Stauffer und Grimson, 1999). Dieses Entscheidungsprinzip lässt sich durch die Erweiterung um ein geeignetes Vordergrundmodell noch verfeinern (Withagen *et al.*, 2003).

2.1 Definition der Problemstellung

In diesem Kapitel soll zunächst nur die saubere Trennung von Vorder- und Hintergrund mittels beleuchtungsunabhängiger Verfahren untersucht werden. Dabei wird das Problem des bewegten Hintergrundes ausdrücklich nicht betrachtet. Dies geschieht später in Kapitel 3.

Wie eingangs erwähnt, lässt sich der Trennungsschritt als Erkennung einer Bildveränderung formulieren. Diesbezüglich geben Radke *et al.* eine gute Definition für die allgemeine Funktionsweise eines Algorithmus zur Erkennung einer Veränderung im Bild (Radke *et al.*, 2005): Es sei $\langle I_t \rangle$ eine Folge von Einzelbildern:

$$\langle I_t \rangle = I_1, \ldots, I_t, \ldots, I_T \qquad t, T \in \mathbb{N} \tag{2.1}$$

Dabei ordnet jedes Einzelbild einer Pixelkoordinate $X \in \mathbb{R}^d$ eine Intensität oder Farbe $I_t(X) \in \mathbb{R}^i$ zu. Prinzipiell erzeugt ein Algorithmus zur Änderungserkennung zum jeweils

letzten Einzelbild ein binäres Bild – die *Change Mask* B_t. Für diese Maske gilt:

$$B_t(X) = \begin{cases} 1 & \text{signifikante Veränderung an Pixel } X \text{ von } I_t \\ 0 & \text{anderenfalls} \end{cases} \tag{2.2}$$

Die inhaltliche Definition der Wendung „signifikante Veränderung“ ist dabei eindeutig anwendungsabhängig. Auch die Interpretation der Werte 0 und 1 von $B_t(X)$ als beispielsweise verändert und unverändert oder Vordergrund und Hintergrund ist abhängig von der Anwendung. Im Folgenden sollen diese beiden Zustände als verändert (*changed*) und unverändert (*unchanged*) bezeichnet werden.

Es existieren mehrere Möglichkeiten, durch die Änderungen im Bild zustande kommen können: Dies sind zum einen Bewegungen der Kamera und der Objekte, zum anderen können Veränderungen auch von schwankender Beleuchtung und Schattenwurf ausgehen. Der in Abschnitt 1.4 beschriebene Aufbau der Spiele, für die ein geeignetes Verfahren gefunden werden soll, schließt Kamerabewegungen als Ursache für Bildveränderungen aus. Hier könnten – je nach Befestigung – lediglich kleine Vibrationen auftreten, die aber unter unvermeidliche Störungen durch Rauschen gefasst werden können. Änderungen, die auf die Beleuchtung zurückzuführen sind, gelten aufgrund der Aufgabenstellung nicht als signifikante Veränderungen: Sind es doch gerade diese Störungen, die das neue Verfahren vermeiden soll. Übrig bleiben lediglich Objektbewegungen, die als Veränderung interpretiert werden dürfen: Bewegt sich der Spieler, muss dies erkannt werden. Dabei ist aber zu beachten, dass für eine Spielsteuerung nicht nur diese Bewegung erkannt werden muss: Auch ein über längere Zeit still stehender Benutzer muss sicher als vom Hintergrund verschieden erkannt werden.

Objektbewegungen bedürfen im Allgemeinen einer besonderen Betrachtung: Sie erzeugen sowohl an der alten als auch an der neuen Position eine Veränderung. Grundlage für die Interaktionserkennung ist aber ausschließlich die aktuelle Position. Dies bedeutet, dass durch Objektbewegung freigelegter Hintergrund auch als solcher erkannt werden muss. Weiterhin können auch Hintergrundobjekte bewegt werden. Dies wird vor allem wichtig bei Anwendungen der Video-Überwachung oder bei Installationen wie dem *„EyeToy“*. Der Vorteil der dieser Arbeit zugrunde liegenden Installation besteht darin, dass diese Objektbewegungen ausschließlich in der Projektion und damit innerhalb einer einzigen Ebene stattfinden. Kapitel 3 befasst sich mit einem Verfahren, das zum Ziel hat, diese Bewegungen zu unterdrücken.

Aus der Aufgabenstellung ergeben sich schließlich noch Vereinfachungen für die zu Beginn dieses Abschnitts aufgestellten allgemeinen Gleichungen:

- Im Rahmen dieser Arbeit wird lediglich mit zweidimensionalen Bildern gearbeitet, sodass für die Dimension eines Pixels gilt: $d = 2$ und damit $X \in \mathbb{R}^2$.

- Weiterhin werden hier nur ein bis drei Farbkanäle pro Bild betrachtet, sodass für deren Anzahl gilt: $i \in [1, 2, 3]$. Dass die Größen X und $I_t(X)$ prinzipbedingt einen diskreten Wertebereich besitzen, kann für die theoretische Betrachtung der Verfahren vernachlässigt werden. Es kann aber später zu Rundungsfehlern kommen, die die Genauigkeit der betreffenden Verfahren beeinträchtigen können.

2.1.1 Notation

Um die Beschreibung der Verfahren zu vereinfachen, gilt für die weitere Arbeit – soweit nicht anders angegeben – folgende Notation:

- I_t sei ein Einzelbild mit in der Regel drei Farbkanälen. Dabei gibt $t \in \mathbb{N}$ die zeitliche Position des Bildes in der Bildfolge $\langle I_t \rangle$ an. Das aktuelle Kamerabild wird mit I_t bezeichnet, ein gespeichertes Hintergrundbild mit I_B.
- $I_t(X)$ liefert die Farbe eines Pixels $X = (x, y)$. Ein dreifarbiges Pixel wird dabei nicht weiter bezeichnet. Soll jedoch ein einzelner Farbkanal explizit angegeben werden, so wird dies rechts oben notiert: $I_t^R(X)$, $I_t^G(X)$, $I_t^B(X)$, $I_t^Y(X)$, $I_t^U(X)$ und $I_t^V(X)$ bezeichnen den roten, gelben und grünen Anteil sowie die Luminanz bzw. die Chrominanz eines Pixels. Dabei werden hier aufgrund der kürzeren Schreibweise U und V anstelle der korrekten Bezeichnungen C_b und C_r verwendet. Weitere Kanalangaben sowie Kombinationen daraus sind möglich.
- Ω^X sei ein quadratischer Block von Pixeln mit der Breite $w \in \mathbb{N}$ und dem Mittelpunkt X. In diesem Block seien die Pixel Ω_n^X mit $n \in \mathbb{N}^{\leq w^2}$ enthalten.
- Die Variable, anhand derer zwischen Vorder- und Hintergrund unterschieden wird, wird als $D(X)$ bezeichnet.
- Da jedes Verfahren eine *Change Mask* erzeugt, wird auf eine explizite Angabe wie in Gleichung 2.2 verzichtet. Stattdessen wird die Bedingung für die Klassifizierung folgendermaßen angegeben:

  $$D(X) \underset{u}{\overset{c}{\gtrless}} \tau$$

  Die Bezeichnung erfolgt mit c für eine Veränderung (*changed*) und u für unveränderte Teile (*unchanged*). Eine weitere Bezeichnung kann i für unbestimmt (*indeterminate*) sein. Eine Einordnung der Gleichheit bleibt dem Leser überlassen.
- Zur Abgrenzung von Pixelfarben und Pixeln werden Vektoren als $\overrightarrow{v}$ geschrieben. Matrizen werden mit Großbuchstaben in Fraktur gesetzt ($\mathfrak{M}$).
- Der Betrag einer Zahl wird als $|\cdot|$ notiert. Der Betrag einer Pixelfarbe wird komponentenweise gebildet. Die Determinante einer Matrix wird ebenfalls mit $|\cdot|$ bezeichnet.
- Die Länge eines Vektors wird als $\|\cdot\|$ notiert.

2.2 Beschreibung der untersuchten Verfahren

Im Folgenden werden einige Verfahren zur Trennung von Vorder- und Hintergrund vorgestellt. Viele Verfahren betrachten dabei zur Änderungserkennung zwei aufeinanderfolgende Bilder I_{t-1} und I_t; die Forderung der sicheren Erkennung eines Benutzers schließt dies aber aus: Würde dieser stehen bleiben, träte keine Änderung des Bildes mehr auf und er

würde nicht mehr erkannt werden. Stattdessen muss für alle Verfahren ein Hintergrundbild I_B aufgenommen werden, demgegenüber eine Veränderung als Benutzer erkannt werden kann.

Die Reihenfolge der Verfahren richtet sich nach ihrer Komplexität: Es wird mit einfachen Differenzverfahren begonnen, danach folgen robustere Verfahren. Dieser Abschnitt schließt mit einem komplexen Verfahren, das mithilfe statistischer Methoden und definierter Anforderungen an die Form der Maske die Wahrscheinlichkeit der Hypothesen *changed* und *unchanged* gegeneinander abwägt.

Für die folgenden Beschreibungen wird ein Wertebereich der einzelnen Farbkomponenten von 0 bis 255 angenommen.

2.2.1 Einfache Differenzverfahren

Bei der Bildung der einfachen Differenz handelt es sich um ein sehr schnelles und weit verbreitetes Verfahren zur Erkennung bewegter Objekte (Yang und Levine, 1992; Radke *et al.*, 2005). Es beruht auf folgendem Prinzip:

$$D(X) = I_t(X) - I_{t-1}(X) \tag{2.3}$$

Unter Berücksichtigung der Tatsache, dass für die gesuchten Verfahren ein Hintergrundbild zur Änderungserkennung angelegt werden muss, folgt:

$$D(X) = I_t(X) - I_B(X) \tag{2.4}$$

Daraus ergibt sich eine geeignete *Change Mask* durch:

$$| D(X) | \begin{matrix} c \\ \gtrless \\ u \end{matrix} \tau \tag{2.5}$$

Der *Threshold* τ erhält der Einfachheit halber für jede Implementation einen festen Wert. Hierfür werden mehrere Werte getestet.

Differenz der Luminanz

Um einen ersten Eindruck von der Tauglichkeit einfacher Differenzbild-Verfahren zu erhalten, wurde die Differenz der Luminanzen von Kamerabild und gespeichertem Hintergrund gebildet und mit festgelegten Werten für den *Threshold* τ verglichen. Somit ergibt sich für dieses Verfahren das Trennungsprinzip aus den Gleichungen 2.4 und 2.5 als:

$$D^Y(X) = I_t^Y(X) - I_B^Y(X) \tag{2.6}$$

$$| D^Y(X) | \begin{matrix} c \\ \gtrless \\ u \end{matrix} \tau \tag{2.7}$$

Für den Test dieses Verfahrens wurden für den *Threshold* die drei Werte 25, 45 und 65 bei einem Wertebereich der Luminanz von 0 bis 255 gewählt, da für diese Werte visuell unterscheidbare Ergebnisse erzielt wurden.

Differenz der Farbe

Als nächstes Kriterium zur Bestimmung einer *Change Mask* wurde die Differenz der einzelnen RGB-Farbkanäle getestet. Es wurde dabei insbesondere geprüft, ob die zusätzliche Farbinformation eine höhere Genauigkeit der Unterscheidung liefert. Für dieses Verfahren ergibt sich das Prinzip der Trennung von Vorder- und Hintergrund aus den Gleichungen 2.4 und 2.5 als:

$$D(X) = I_t(X) - I_B(X) \tag{2.8}$$

$$| D(X) | \overset{c}{\underset{u}{\gtrless}} \tau \tag{2.9}$$

Dabei wird die Entscheidung zwischen *changed* und *unchanged* für jede Farbkomponente einzeln getroffen.

Auch dieses Verfahren wurde für die drei Werte 25, 45 und 65 für τ getestet.

Differenz der Chrominanz

Da der vorhergehende Test gegen die Farbdifferenz implizit auch gegen die Helligkeit testet, wurde für die nächsten Tests ein Entscheidungskriterium herangezogen, das die Helligkeit ausdrücklich nicht beinhaltet. Um dies zu erreichen, wurde die komponentenweise Differenz der jeweiligen Chrominanzen zur Maskenbildung verwendet. Für dieses Verfahren ergibt sich aus den Gleichungen 2.4 und 2.5:

$$D^{U,V}(X) = I_t^{U,V}(X) - I_B^{U,V}(X) \tag{2.10}$$

$$| D^{U,V}(X) | \overset{c}{\underset{u}{\gtrless}} \tau \tag{2.11}$$

Auch hier wird die Entscheidung komponentenweise getroffen.

Bei diesem Test wurden die drei *Thresholds* 10, 15 und 25 verwendet, da sich bereits dabei deutliche visuelle Unterschiede ergaben.

Differenz des Farbtons

Beim folgenden Test beruhte die Entscheidung nur noch auf einem Wert für die Farbigkeit, nicht wie zuvor auf zweien – den beiden Komponenten der Chrominanz. Dies legt einen Test gegen den Farbton nahe. Es ergibt sich aus den Gleichungen 2.4 und 2.5:

$$D^H(X) = I_t^H(X) - I_B^H(X) \tag{2.12}$$

$$| D^H(X) | \overset{c}{\underset{u}{\gtrless}} \tau \tag{2.13}$$

Hier wurde wieder mit den drei *Thresholds* 25, 45 und 65 gearbeitet. Zusätzlich wurde aufgrund der erhaltenen Ergebnisse noch ein wesentlich höherer *Threshold* von 100 gewählt.

2.2.2 Chrominanzabweichung

Kamkar-Parsi *et al.* entwickelten einen mehrteiligen Algorithmus zur Trennung von Vorder- und Hintergrund (Kamkar-Parsi *et al.*, 2005). Ein erster Teil darin legt der Entscheidung zwischen *changed* und *unchanged* ebenfalls die Chrominanz zugrunde. Dabei wird allerdings nicht wie bei den zuvor vorgestellten einfachen Differenzbild-Verfahren die komponentenweise Differenz gebildet, sondern der euklidische Abstand zweier Farbwerte in der UV-Farbebene gemessen. Aus Gründen der Verarbeitungsgeschwindigkeit wird aber nicht diese Distanz direkt, sondern deren Quadrat mit einem *Threshold* verglichen. Um dies zu verdeutlichen, wird die Entscheidungsvariable als D^2 anstelle von D geschrieben. Nach Kamkar-Parsi *et al.* ergibt sich damit aus den Gleichungen 2.4 und 2.5:

$$D^2(X) = \left[\left(I_t^U(X) - I_B^U(X)\right)^2 + \left(I_t^V(X) - I_B^V(X)\right)^2 \right] \cdot 0,5 \tag{2.14}$$

$$D^2(X) \underset{u}{\overset{c}{\gtrless}} \tau^2 \tag{2.15}$$

Dabei dient der Faktor 0,5 zur Skalierung des Wertebereichs der Entscheidungsvariable $D^2(X)$ auf 0 bis 65.025;

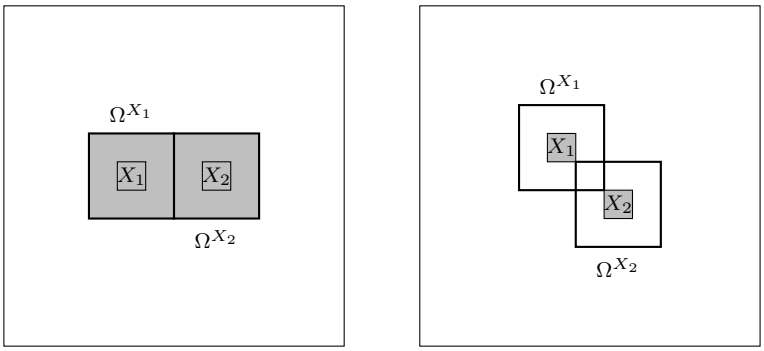
Kamkar-Parsi *et al.* weisen darauf hin, dass eine Chrominanz-basierte Entscheidung nicht getroffen werden kann, wenn die Sättigungen beider verwendeter Farbwerte zu niedrig sind. In diesem Falle enthalten die zugehörigen Chrominanzkomponenten keine aussagekräftige Farbinformation, und damit muss das zugehörige Pixel als *indeterminate* bezeichnet und mithilfe von weiteren Verfahren in eine *Change Mask* eingeordnet werden. Die Sättigung kann folgendermaßen bestimmt werden (Kamkar-Parsi *et al.*, 2005):

$$I_t^S(X) = \sqrt{\left(I_t^U(X) - W^U\right)^2 + \left(I_t^V(X) - W^V\right)^2} \cdot \sqrt{2} \tag{2.16}$$

Dabei geben W^U und W^V die Chrominanzkomponenten des gewählten Weißpunkts an. Für die vorliegende Implementation im Wertebereich von 0 bis 255 ergibt sich für beide der Wert 127,5. Der Faktor $\sqrt{2}$ dient lediglich zur Skalierung des Wertebereichs der Sättigung $I_t^S(X)$ auf 0 bis 255. Die mathematische Definition der Sättigung nach Gleichung 2.16 liefert zwar andere Werte als die Definition nach Smith, dennoch ist die inhaltliche Definition als „Abstand zu farblos, also weiß oder grau“ identisch (Smith, 1978). Die voneinander abweichenden Werte rühren lediglich von unterschiedlichem Aufbau der entsprechenden Farbräume her und wirken sich auch nicht prinzipiell auf die Beurteilung der Aussagekraft der Chrominanzwerte aus. Formal ergibt sich die Entscheidung für *indeterminate* demnach als:

$$I_t^S(X) + I_B^S(X) \underset{i}{\overset{c,u}{\gtrless}} 2 \cdot \tau^S \tag{2.17}$$

Dieses Verfahren wurde mit den drei Schwellwerten 10, 15 und 25 getestet. Somit ergab sich der Vergleich von D^2 mit den quadratischen *Thresholds* 100, 225 und 625. Als Sättigungsuntergrenze τ^S wurde ein Wert von 20 gewählt.



a b

Abbildung 2.1: Zwei Möglichkeiten der Block-Zerlegung:
a) Das Bild wird in nicht-überlappende Blöcke unterteilt und der gesamte Block als *changed* oder *unchanged* klassifiziert.
b) Die Blöcke sind überlappend und nur der Mittelpunkt wird aufgrund des Inhalts des Blocks klassifiziert.
(verändert nach: Radke *et al.*, 2005)

2.2.3 Abweichung des Luminanzvektors

In den meisten Fällen – wie auch für das gesuchte Verfahren – finden signifikante Bildveränderungen nicht an vereinzelten Bildpunkten, sondern in zusammenhängenden Regionen statt. Daher liegt es auf der Hand, zur Entscheidung zwischen Vorder- und Hintergrund, einen kleinen Ausschnitt eines Bildes als Ganzes zu betrachten. Dazu wird ein Bild in einzelne quadratische Blöcke Ω^X unterteilt, deren jeweiliger Mittelpunkt das Pixel X ist. Alle Pixel innerhalb eines Blocks tragen dann zur Entscheidung zwischen *changed* und *unchanged* bei. Das Ergebnis kann entweder dem gesamten Block (Abbildung 2.1a) oder nur dem Mittelpunkt zugewiesen werden (Abbildung 2.1b). Im Allgemeinen ist Letzteres vorzuziehen: Aufgrund der Zerlegung in Blöcke erfolgt eine räumliche Tiefpass-Filterung des Bildes, welche zum einen das Rauschen zum anderen aber auch die Auflösung reduziert. Die erste Variante würde durch die Zuweisung des Ergebnisses in den gesamten Block nochmals filtern. Sie wäre zwar deutlich schneller, da wesentlich weniger Blöcke verarbeitet werden müssten, die Auflösung würde aber für eine aussagekräftige Maske durch die zweifache Filterung zu weit reduziert werden (Aach *et al.*, 1993; Radke *et al.*, 2005).

In ihrem mehrteiligen Algorithmus zur Trennung von Vorder- und Hintergrund verwenden Kamkar-Parsi *et al.* ebenfalls ein solches Verfahren mit Blöcken der Breite 3: Für jedes Pixel eines Bildes wird ein Vektor erstellt, dessen Komponenten sich aus der Luminanz des zugehörigen Pixels sowie den Luminanzen der horizontal und vertikal benachbarten Pixel zusammensetzt (Abbildung 2.2). Formal ergibt sich dieser Vektor als:

$$\overrightarrow{y}_t(X) = \left[\, I_t^Y\left(\Omega_2^X\right),\ I_t^Y\left(\Omega_4^X\right),\ I_t^Y\left(\Omega_5^X\right),\ I_t^Y\left(\Omega_6^X\right),\ I_t^Y\left(\Omega_8^X\right) \,\right] \tag{2.18}$$

Da der zugehörige Block symmetrisch um das Pixel X liegt, ist $\Omega_5^X = X$. Kamkar-Parsi *et al.* verwenden als Entscheidungskriterium den Winkel zwischen dem Luminanzvektor des

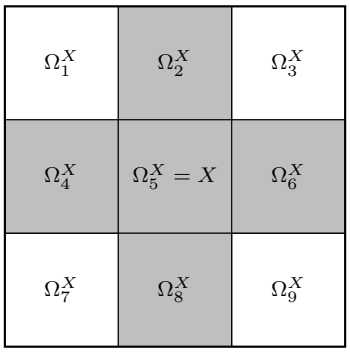


Abbildung 2.2: Für den Luminanzvektor verwendete Pixel

Vordergrundpixels und dem Luminanzvektor des zugehörigen Pixels im Hintergrundbild. Anstelle dessen dient hier der wiederum weniger aufwendig zu berechnende Cosinus dieses Winkels zur Bestimmung der *Change Mask*:

$$\begin{aligned} D(X) &= 1 - \frac{\overrightarrow{y}_t(X) \cdot [\overrightarrow{y}_B(X)]^T}{\| \overrightarrow{y}_t(X) \| \cdot \| \overrightarrow{y}_B(X) \|} \\ &= 1 - \overline{y}_t(X) \cdot [\overline{y}_B(X)]^T \end{aligned} \tag{2.19}$$

$$D(X) \underset{u}{\overset{c}{\gtrless}} \tau \tag{2.20}$$

Dabei ist zu beachten, dass die Vektoren für das aktuelle und das Hintergrundbild normalisiert vorliegen müssen. Damit diese Normalisierung durchgeführt werden kann, müssen beide Vektoren eine von Null verschiedene Länge haben. Anderenfalls sind die Pixel als *indeterminate* zu kennzeichnen:

$$\| \overrightarrow{y}_t(X) \|, \| \overrightarrow{y}_B(X) \| \overset{c,u}{\underset{i}{\geqq}} 0 \tag{2.21}$$

Dieses Verfahren wurde für die drei Werte 0,098, 0,118 und 0,138 als *Thresholds* getestet. Dies entspricht in etwa Winkeln von 25,6 °, 28,1 ° und 30,4 ° zwischen den Vektoren des aktuellen und des Hintergrundpixels.

2.2.4 Abweichung des Luminanzvektors – nötige Veränderungen

Das zuvor beschriebene Verfahren liefert bei einfarbigen Vordergrundobjekten, die sich vor andersfarbigem aber ebenfalls einfarbigem Hintergrund befinden, lediglich eine feine Konturlinie in der Maske. An diesen Stellen sind die Luminanzvektoren von Vorder- und Hintergrundpixel kollinear. Dies hat per Definition eine Klassifizierung als Hintergrund zur Folge. Daher schlagen Kamkar-Parsi *et al.* eine Erweiterung dieses Verfahrens vor:

Für Pixel, die zuvor als Hintergrund eingestuft wurden, wird das Verhältnis ihrer Längen mit einem weiteren Schwellwert verglichen.

$$\frac{\parallel \overrightarrow{y}_t(X) \parallel}{\parallel \overrightarrow{y}_B(X) \parallel} \overset{u}{\in} \left[\tau_L^{MR}\ ,\ \tau_U^{MR}\right] \qquad \frac{\parallel \overrightarrow{y}_t(X) \parallel}{\parallel \overrightarrow{y}_B(X) \parallel} \overset{c}{\notin} \left[\tau_L^{MR}\ ,\ \tau_U^{MR}\right] \tag{2.22}$$

Dabei kann $\tau_L^{MR} = \left(\tau_U^{MR}\right)^{-1}$ gewählt werden.

Aufgrund der Tatsache, dass die in dunklen Bereichen entstehenden kurzen Vektoren durch Kamerarauschen eine größere Winkelabweichung erfahren können als längere, wird hier der von Kamkar-Parsi *et al.* vorgeschlagenen Erweiterung noch eine weitere hinzugefügt: Damit die Aussage der *Change Mask* nicht durch dieses Rauschen beeinflusst wird, muss auch für die Vektorlänge ein Schwellwert eingeführt werden – ähnlich wie für die Sättigung bei der Chrominanzabweichung (Unterabschnitt 2.2.2). Dazu werden Regionen, in denen sowohl die Luminanzvektoren im aktuellen wie auch die im gespeicherten Hintergrundbild eine bestimmte Länge nicht überschreiten, als *indeterminate* bezeichnet. Damit ergibt sich für Gleichung 2.21 folgende Korrektur:

$$\parallel \overrightarrow{y}_t(X) \parallel + \parallel \overrightarrow{y}_B(X) \parallel \overset{c,u}{\underset{i}{\gtrless}} 2\tau^M \qquad \parallel \overrightarrow{y}_t(X) \parallel,\ \parallel \overrightarrow{y}_B(X) \parallel \neq 0 \tag{2.23}$$

Das so veränderte Verfahren wurde nochmals für einen *Threshold* von 0,098 getestet. Dies entspricht einem Winkel von etwa 25.6 °. Als τ_U^{MR} wurden die drei Werte 3,5, 10,0 und 35,0, als τ^M wurde eine Länge von 50 verwendet.

2.2.5 Blockbasierter Kollinearitätstest mittels Bayes'scher Entscheidung und Markov-Zufalls-Feldern

Ein ebenfalls blockbasierter Ansatz zur Trennung von Vorder- und Hintergrund wird von Aach *et al.* vorgestellt und durch ihre spätere Arbeit ergänzt und weiter verbessert (beispielsweise Aach *et al.*, 1993; Aach und Kaup, 1995; Mester *et al.*, 2001). Ausgehend von diesem Ansatz implementierten Griesser *et al.* diesen Algorithmus mit *OpenGL* und *Shader*-Programmierung auf der Grafikkarte (*GPU*) (Griesser *et al.*, 2005).

Auch für dieses Verfahren wird bei Mester *et al.* aus den Luminanzen eines Pixelblocks ein Zeilen-Vektor erstellt – allerdings sind in diesem sämtliche benachbarten Pixel berücksichtigt (Mester *et al.*, 2001). Mester *et al.* gründen dabei eine Trennungsentscheidung ebenfalls auf die Betrachtung zweier aufeinanderfolgender Bilder. Wie zuvor auch wird hier aber das aktuelle Kamerabild mit einem gespeicherten Hintergrund verglichen. Damit ergeben sich wiederum die Vektoren $\overrightarrow{y}_t(X)$ und $\overrightarrow{y}_B(X)$ für das aktuelle und das Hintergrund-Pixel.

Für den Fall, dass bei dem in Abschnitt 1.4 gegebenen Aufbau zwischen Hintergrund und aktuellem Bild keine Objektveränderung stattgefunden hat, kann eine Bildveränderung nur noch durch wechselnde Beleuchtung (Faktor k) und additive Rausch-Störungen (ε_1 und ε_2) entstanden sein. Mithilfe der Annahmen, dass sich die aufgenommene Luminanz als Produkt aus Beleuchtung und Reflexion der Objekte modellieren lässt, und dass das additive Rauschen Gauß-verteilt ist, lassen sich in diesem Fall die beiden beobachteten

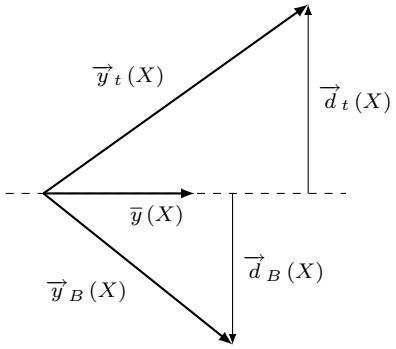


Abbildung 2.3: Geometrische Interpretation des Kollinearitätstests (verändert nach: Griesser *et al.*, 2005)

Vektoren auch von einem unbekannten gemeinsamen Signal $\overrightarrow{y}(X)$ ableiten:

$$\overrightarrow{y}_t(X) = k \cdot \overrightarrow{y}(X) + \varepsilon_1 \tag{2.24}$$

$$\overrightarrow{y}_B(X) = \overrightarrow{y}(X) + \varepsilon_2 \tag{2.25}$$

Daraus lässt sich – wie für den Test auf eine Luminanzvektor-Abweichung aus Unterabschnitt 2.2.3 – ein Test auf Kollinearität beider Vektoren herleiten. Mester *et al.* weisen aber ausdrücklich darauf hin, dass die zur Winkelbestimmung notwendige Normalisierung der Vektoren „statistische Informationen irreversibel unterdrückt" (Mester *et al.*, 2001). Stattdessen führen sie die geometrische Interpretation (Abbildung 2.3) des Tests auf Kollinearität auf eine *Maximum-Likelihood*-Schätzung der wahren Signalrichtung $\overline{y}(X)$ zurück. Diese kann mittels Minimierung der Summe aus Gleichung 2.26 geschätzt werden:

$$D^2(X) = \| \overrightarrow{d}_t(X) \|^2 + \| \overrightarrow{d}_B(X) \|^2 \tag{2.26}$$

Unter Berücksichtigung der folgenden Matrix

$$\mathfrak{Y} = \begin{bmatrix} \overrightarrow{y}_t(X) \\ \overrightarrow{y}_B(X) \end{bmatrix} \tag{2.27}$$

zeigen Mester *et al.*, dass $D^2(X)$ identisch ist mit dem kleinsten Eigenwert der Matrix $\mathfrak{Y} \cdot \mathfrak{Y}^T$ (Mester *et al.*, 2001). Griesser *et al.* leiten daraus mittels der Substitutionen

$$f := \overrightarrow{y}_t(X) \cdot [\overrightarrow{y}_t(X)]^T \tag{2.28}$$

$$b := \overrightarrow{y}_B(X) \cdot [\overrightarrow{y}_B(X)]^T \tag{2.29}$$

$$c := \overrightarrow{y}_t(X) \cdot [\overrightarrow{y}_B(X)]^T \tag{2.30}$$

die Entscheidungsvariable gemäß Gleichung 2.31 her (Griesser *et al.*, 2005):

$$D^2(X) = \frac{f + b - \sqrt{(f-b)^2 + 4 \cdot c^2}}{2} \tag{2.31}$$

Unter Zuhilfenahme der gewünschten Eigenschaft der *Change Mask*, dass sie nur kompakte, glatte Regionen maskiert, lassen sich *a priori*-Wahrscheinlichkeiten der Klassifizierungen *changed* und *unchanged* mit einem zweidimensionalen Markov-Feld beschreiben. Mit dieser Annahme wird die Entscheidungsvariable in das von Aach und Kaup entwickelte Bayes'sche *Framework* integriert (Aach und Kaup, 1995). Schließlich ergibt sich für diese Entscheidung (Mester *et al.*, 2001; Griesser *et al.*, 2005):

$$D^2(X) \underset{u}{\overset{c}{\gtrless}} \tau_s + \tau_a = \tau_s + (4 - n_c) \cdot C \tag{2.32}$$

Dabei stellen τ_s und τ_a einen statischen sowie einen adaptiven *Threshold* dar. n_c bezeichnet die Anzahl der benachbarten, als verändert eingestuften Pixel, bei C handelt es sich um eine positive Konstante. Die Tatsache, dass die Anzahl der als *changed* klassifizierten Pixel zur Festlegung des adaptiven *Thresholds* bekannt sein muss, legt iteratives Vorgehen nahe. Im ersten Iterationsschritt eines jeden Bildes wird zur Bestimmung von n_c die Maske des vorhergehenden Bildes verwendet.

Zum Test dieses Verfahrens werden nicht die Luminanzen herangezogen (Mester *et al.*, 2001), sondern die Farbwerte (Griesser *et al.*, 2005). Weiterhin wird die ebenfalls von Griesser *et al.* vorgeschlagene Ergänzung zur Behandlung sehr dunkler Stellen im Bild verwendet.

Mester *et al.* wiesen empirisch nach, dass es sich bei der Entscheidungsvariablen $D^2(X)$ um eine χ^2-verteilte Zufallsgröße mit w^2-1 Freiheitsgraden handelt (w: Breite des Blocks). Damit ließe sich für eine gewünschte Signifikanz der statische *Threshold* für eine spätere konkrete Anwendung bestimmen. Hier wurden zum Vergleich geeignete Werte empirisch ermittelt.

Das beschriebene Verfahren wurde mit einer Blockbreite von drei Pixeln für die drei Werte 2500, 4500 und 6500 für τ_s sowie 200 für C und einer Kompensation für dunkle Bereiche von 76 für $\sqrt{O_{dc}}$ mit sechs Iterationsschritten getestet.

2.2.6 Blockbasierter Kollinearitätstest – kleine Veränderung

Der im vorigen Unterabschnitt vorgestellte Kollinearitätstest geht ursprünglich auf ein Luminanz-basiertes Verfahren zurück (Mester *et al.*, 2001). Griesser *et al.* verwendeten zur Entscheidungsbildung die RGB-Farbwerte eines Pixelblocks. Dabei ist das ursprüngliche Verfahren bereits vor dem Hintergrund der Beleuchtungsunabhängigkeit erstellt worden.

Die RGB-Farbinformation ist der Entscheidungsbildung mittels Luminanz vorzuziehen, weil sie zusätzliche Unterscheidbarkeit bietet. Da sie dennoch implizit die Luminanz beinhaltet, wird der Algorithmus so verändert, dass stattdessen nur die Chrominanz verwendet wird.

Es verändern sich dabei im Verfahren lediglich die Dimensionen der Vektoren $\overrightarrow{y}_t(X)$ und $\overrightarrow{y}_B(X)$ – die für diese Abwandlung besser als $\overrightarrow{v}_t^{U,V}(X)$ und $\overrightarrow{v}_B^{U,V}(X)$ bezeichnet werden – sowie die Dimension der zugehörigen Matrix. Das übrige Prinzip bleibt identisch.

Diese Änderung in der Dimension der Vektoren zieht geänderte Werte für τ_s und C nach sich. Für die abschließenden Tests wurden die drei Werte 550, 1650 und 2750 für τ_s sowie 80 für C mit sechs Iterationsschritten getestet. Der Wert 76 für $\sqrt{O_{dc}}$ wurde beibehalten.

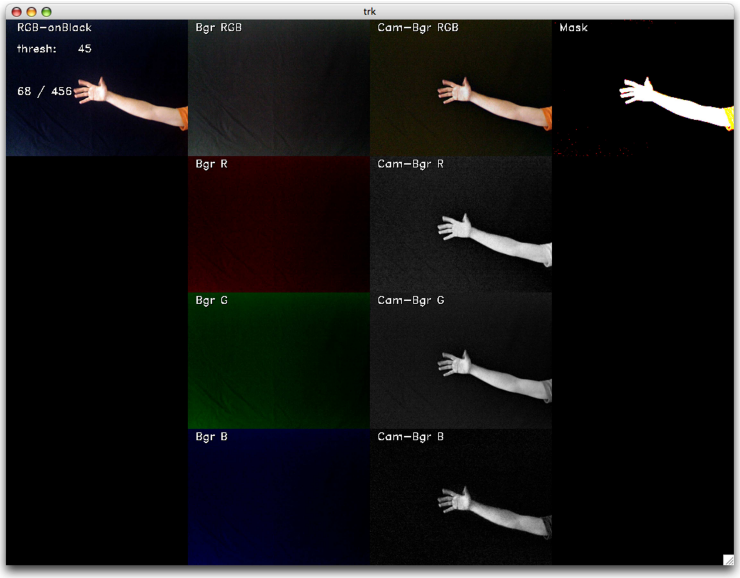


Abbildung 2.4: Screenshot der Software zum Test der Trennungsverfahren

2.3 Entwicklung einer geeigneten Testumgebung

Damit die verschiedenen Verfahren zur Trennung von Vorder- und Hintergrund vergleichbar auf ihre Tauglichkeit untersucht werden können, muss eine geeignete Umgebung geschaffen werden. In ihr muss die Möglichkeit existieren, die fraglichen Verfahren auf einfache Art und Weise mithilfe vorgefertigter Module zu implementieren. So lässt sich – durch sorgfältige Planung und Vorarbeit – erreichen, dass eine Vielzahl von Verfahren untersucht werden kann. Weiterhin kann die Testumgebung damit auch außerhalb dieser Arbeit Grundlage weitergehender Untersuchungen sein.

Um eine Vergleichbarkeit der Ergebnisse zu erreichen, bietet die Umgebung die Möglichkeit, Bildsequenzen aufzunehmen und diese dann zu einem späteren Zeitpunkt mit allen Verfahren zu verarbeiten. Die erzielten Bildergebnisse werden nach der Bearbeitung für eine Auswertung gespeichert. Neben dieser Möglichkeit, fertige Bildsequenzen zu verarbeiten, ist es unbedingt erforderlich, die Verfahren während ihrer Implementation zu testen. Dazu – und zur Verdeutlichung der einzelnen Verfahrensschritte – werden möglichst viele Zwischenschritte angezeigt.

Ein Beispiel der Anwendung dieser Software zeigt Abbildung 2.4.

2.3.1 Verwendetes Computersystem

Das für die Entwicklung zur Verfügung stehende Computersystem bestand aus einem *MacBook Pro* mit:

- 2,33 GHz Intel Core 2 Duo-Prozessor

- 2 GB 667 MHz DDR2 SDRAM
- ATI Radeon X1600 (PCIe-Bus)
- Mac OS X 10.4
- integrierter iSight Kamera (Version 1.84)

2.3.2 Wahl der Programmiersprache

Der Testumgebung soll ein möglichst modulares Konzept zugrunde liegen: Auf einfache Weise kann so ein Verfahren aus einzelnen Funktionsblöcken zusammengesetzt werden. Aus dieser Anforderung ergibt sich, dass die zu entwickelnde Anwendung objektorientiert programmiert werden muss. Die Entwicklung sollte aber nicht so weit gehen, dass für die Implementation der Verfahren ein eigenes grafisches *User Interface* zur Verfügung gestellt wird. Stattdessen werden die Verfahren aus einzelnen Bausteinen programmiert.

Bei dem im Rahmen dieser Arbeit zu entwickelnden Verfahren handelt es sich um eine zeitkritische Anwendung der Bildverarbeitung, die parallel zu einer übergeordneten Anwendungslogik ablaufen soll. Deshalb muss – zumindest in der späteren Implementation für eine konkrete Installation – großer Wert auf effiziente Programmiertechniken gelegt werden. Für einen Test der bildverarbeitenden Verfahren spielt der Aspekt Zeit in erster Linie eine untergeordnete Rolle. Soll sich aus der Test-Applikation aber auf möglichst kurzem Wege eine konkrete Anwendung ergeben, muss die gewählte Programmiersprache zumindest die Möglichkeit bieten, zeitlich effizient arbeitende Software zu erstellen. Dies – besonders auf dem Hintergrund der Bildverarbeitung – erfordert möglichst hardwarenahe Programmierung.

Die Programmiersprache *C++* bietet sowohl Objektorientierung als auch die Möglichkeit, sehr effizient zu programmieren. Die Alternative *Java* scheidet vor allem vor dem Hintergrund aus, dass die entstehenden Programme zur Laufzeit von einem Interpreter ausgeführt werden, wodurch sich ihre Ausführung verlangsamt. Die durch *Java* gegebene höhere Portabilität ist kein Kriterium, das dies aufwiegen würde, da für die Betriebssysteme, für die das zu entwickelnde Verfahren eingesetzt wird, *C++-Compiler* vorhanden sind. Darüber hinaus wird die Entwicklung in *C++* bevorzugt, da aufgrund der Erfahrungen des Autors wesentlich früher sichere Ergebnisse zu erwarten sind.

2.3.3 Wahl der Bibliothek zur Bildverarbeitung

Die in diesem Abschnitt beschriebene Testumgebung soll die Möglichkeit bieten, sowohl Bilder von einer Kamera aufzunehmen und zur Verfügung zu stellen als auch Verarbeitungsschritte an ihnen durchzuführen. In einem ersten Schritt bietet es sich an, für diese Aufgaben, die vielfach auch in anderem Zusammenhang Anwendung finden, eine bereits existierende Bibliothek zu verwenden. Werden dabei spezielle Probleme deutlich, können diese hinterher mit anderen Mitteln gelöst werden. Bei der Konzeption des Programms ist darauf zu achten, dass dies leicht möglich bleibt.

Die o. g. Anforderungen an die zu wählende Bibliothek werden von *OpenCV* von Intel erfüllt (Intel, 2006). Zum einen bietet diese Bibliothek die Möglichkeit, Bilder der integrierten sowie anderer angeschlossener Kameras einzulesen, zum anderen sind in ihr auch

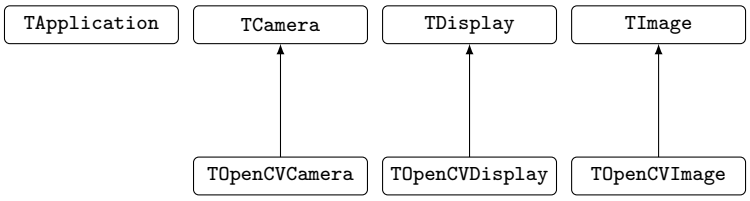


Abbildung 2.5: Klassenstruktur des Programms zum Testen der Trennungsverfahren

sehr effizient implementierte Algorithmen zur Bildverarbeitung enthalten. Darüber hinaus zielt diese Bibliothek auf Echtzeit-Applikationen im Bereich der kamerabasierten Mensch-Computer-Interaktion ab. Weiterhin ist diese Bibliothek in der Programmiersprache *C* geschrieben, wodurch eine enge Anbindung möglicht ist. Auch hier waren nicht zuletzt die Erfahrungen des Autors ausschlaggebend.

2.3.4 Klassenstruktur und Flussdiagramm

Die der im Rahmen dieser Arbeit entwickelten Software zugrunde liegende objektorientierte Programmierung ermöglicht es, das Projekt in funktionale Einheiten zu gliedern. Auf diese Weise wird eine später leicht zu erweiternde Struktur geschaffen. Die dazu gewählte Klassenstruktur geht aus Abbildung 2.5 hervor.

Die Klasse `TApplication` kontrolliert den Programmablauf gemäß Abbildung 2.6 und enthält die einzelnen Implementationen der Verfahren als jeweils eigene Funktion. Diese können der Übersicht halber auf mehrere Quellcode-Dateien verteilt werden.

Während die Klassen `TCamera` sowie `TOpenCVCamera` lediglich zur Bildaufnahme von der Kamera und die Klassen `TDisplay` sowie `TOpenCVDisplay` zur Anzeige von Bildergebnissen und der Aufnahme von Benutzereingaben dienen, sind in den Klassen `TImage` sowie `TOpenCVImage` der Speicherplatz für Bilddaten und die für die Tests wichtigen Funktionen zur Bildverarbeitung enthalten. Die scharfe Abgrenzung der Klassen in einzelne Funktionsblöcke geht soweit, dass `TCamera` und `TDisplay` sowie die davon abgeleiteten Klassen keinen eigenen Speicherplatz für Bilddaten zur Verfügung stellen. Dieser muss vor Erstellung der aus diesen Klassen hervorgehenden Objekte reserviert worden sein und das zugehörige Bildobjekt dem entsprechenden Konstruktor bei Erstellung eines Kamera- oder Anzeigeobjektes übergeben werden. Der große Vorteil dieser Trennung im Hinblick auf einfache Implementation der einzelnen Verfahren liegt darin, dass sich dadurch beispielsweise das Auslesen eines Kamerabildes nicht vom Auslesen eines weiterverarbeiteten Bildes unterscheidet. Die Kamera-Klasse sorgt automatisch dafür, dass das aufgenommene Kamerabild im vorgesehenen Bildobjekt korrekt abgelegt wird.

Die Klassen `TCamera`, `TDisplay` und `TImage` dienen dazu, eine von *OpenCV* unabhängige Daten- und Funktionsstruktur zu erzeugen. So sind spätere Erweiterungen auch mit Funktionen anderer Bibliotheken möglich. Dies wurde bisher lediglich bei der Klasse `TImage` genutzt: In ihr ist zum einen der Speicherplatz für die Bilddaten in einem eigenen Format definiert, zum anderen finden sich hier Konversions-, Schwellwert- und Insert-Funktionen. Die übrigen Funktionen wurden in den abgeleiteten Klassen implementiert.

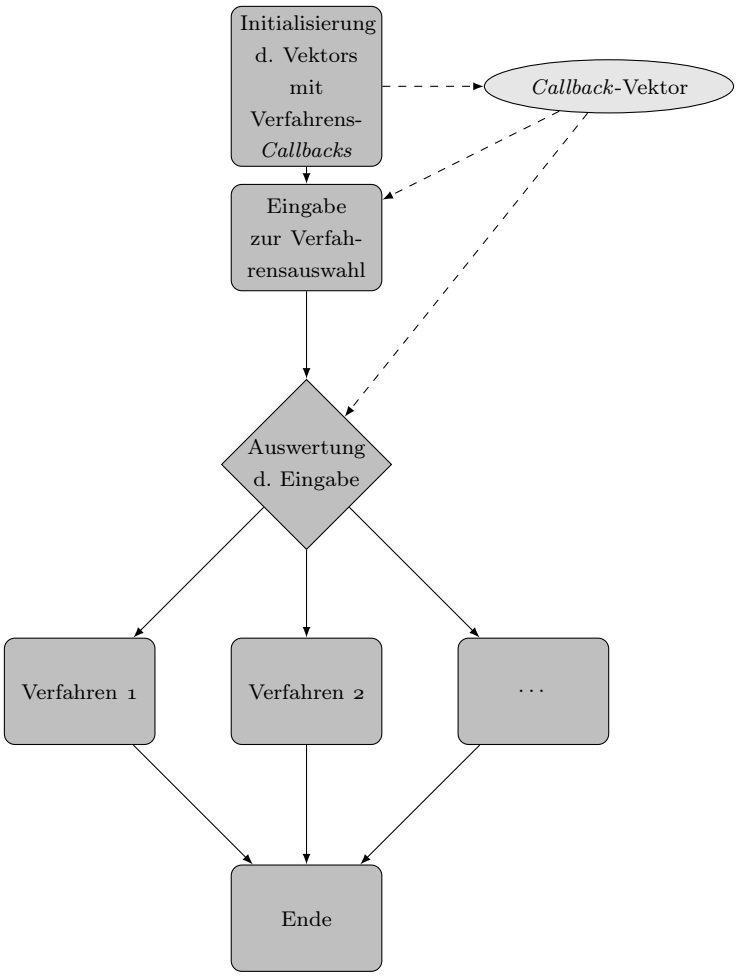


Abbildung 2.6: Flussdiagramm für die Testumgebung

Listing 2.1: Datenstruktur für den *Callback*-Vektor

```
struct Callbacks
{
  int (TApplication::*MethodFunction)( int, char* const* );
  std::string methodString;
  std::string methodName;
};
```

2.3.5 Modularisierung für die Verfahren

Die Aufteilung in die beschriebenen Klassen gibt einerseits eine starke Strukturierung innerhalb der einzelnen Verfahrens-Implementationen vor, andererseits wird in der Testumgebung jedes Verfahren selbst als einzelner Baustein programmiert. Dazu wird jedes Verfahren für sich in einer eigenen Funktion implementiert, die jeweils – in einem eigenständigen Programm – eine vollständige `main()`-Funktion darstellen könnte. Um dieses Baukasten-Prinzip zu unterstützen, sind bis auf die für eine `main()`-Funktion übliche Parameterliste und den Namen des Verfahrens keine Daten von außerhalb der Funktion notwendig. Der Name dient dabei lediglich zur Identifikation des gewählten Verfahrens in der Konsole.

Nachdem ein neues Verfahren implementiert und die zugehörige Funktion im *Header* der Klasse `TApplication` bereitgestellt wurde, kann es während der Initialisierung im Konstruktor der Anwendungsklasse zu einem *Callback*-Vektor hinzugefügt werden (siehe Abbildung 2.6). Dieser *STL*-Vektor – der Datenmember `fpMethodVector` der Klasse `TApplication` – enthält Einträge vom eigens dafür erstellten Typ `Callbacks` (Listing 2.1). Innerhalb dieser Datenstruktur enthält das erste Element den zugehörigen Funktions-Pointer, in `methodString` einen Kurz- und in `methodName` einen Langnamen. Dieser Vektor wird automatisch beim Programmstart in der Funktion `Run()` ausgewertet und aufgrund dessen wird dem Benutzer eine Auswahl von möglichen Verfahren gezeigt. Nach erfolgter Eingabe wird mithilfe des gespeicherten Funktionspointers automatisch in die gewählte Funktion verzweigt. Auf diese Weise können neue Verfahren leicht in diese Testumgebung integriert werden: Listing 2.2 zeigt den Befehl, der nötig ist, um ein in der Funktion `foo()` neu implementiertes Verfahren in der Testumgebung anzumelden.

2.3.6 Speicherverwaltung für die einzelnen Bilder

Es ist wichtig für einen Test sowie für die Beurteilung der ausgewählten Verfahren, dass viele Zwischenergebnisse – wie die Differenzbilder – angezeigt werden können. Dazu muss am Beginn eines Verfahrens eine ausreichende Menge an Speicher für Bildobjekte reserviert werden. Dieser Speicher wird – zumindest bei Verwendung von *OpenCV* – dynamisch zur Laufzeit reserviert. Daher ist eine explizite Freigabe dieser Bereiche am Programmende erforderlich. Um auch hier eine große Flexibilität zu erreichen und dem Programmierer damit die Arbeit zu erleichtern, wurde ein simpler Speicher-Manager implementiert, der

Listing 2.2: Hinzufügen der neu erstellten Funktion `foo()` zum *Callback*-Vektor. Nach diesem Schritt steht dem Benutzer das neue Verfahren zur Verfügung.

```
fpMethodVector.push_back(
  (Callbacks){
    &TApplication::foo,  // function pointer
    "method01",          // short name
    "Method one"         // longer name
  }
);
```

Listing 2.3: Gegenüberstellung von `new` und `NewImage()` zur Erstellung eines Bildobjektes

```
TOpenCVImage* imgA = new TOpenCVImage( 1024, 768, 3 );
TOpenCVImage* imgB = NewImage( 1024, 768, 3 );
```

den Bildspeicher mit Hilfe eines *STL-Stacks* verwaltet.

Anstatt jedes einzelne Bild mithilfe des Operators `new` zu erstellen, kann dies durch Aufruf der Funktion `NewImage()` erfolgen. Dabei unterscheidet sich die Syntax der Befehle nur unwesentlich; ein Beispiel ist in Listing 2.3 gegeben. Innerhalb der Funktion `NewImage()` erfolgt dann die gewohnte dynamische Erstellung eines Bildobjektes mithilfe von `new` – ein Zeiger auf den neuen Speicher wird zurückgegeben. Zusätzlich wird diese Speicherstelle auf dem dafür vorgesehenen *Stack* abgelegt (Listing 2.4). Der Vorteil gegenüber der direkten Anwendug von `new` und `delete` kommt dann am Ende einer jeden Implementation eines Verfahrens zum Tragen: Es ist nur ein einziger Befehl notwendig, um den gesamten reservierten Bildspeicher in der richtigen Reihenfolge freizugeben – ein Aufruf der Funktion `FreeImageMemory()`. Darin werden die auf dem *Stack* abgelegten Speicherbereiche einzeln mit `delete` wieder freigegeben (Listing 2.5).

Auf diese Weise wird es dem Programmierer, der ein Verfahren für Tests implementiert, ermöglicht, eine beliebige Anzahl von Bildobjekten nahezu wie gewohnt zu erstellen, ohne jedes einzelne explizit löschen zu müssen. Dies verringert die Fehleranfälligkeit deutlich.

2.3.7 Funktionen zur Bildverarbeitung

Für die Implementation der einzelnen Verfahren sind grundlegende Bildverarbeitungsschritte sowie der Zugriff auf einzelne Datenelemente nötig. Diese wurden teilweise in der Klasse `TImage` direkt, teilweise in `TOpenCVImage` mittels der von *OpenCV* zur Verfügung gestellten Funktionalität implementiert. Einen Überblick über die wichtigsten bildverarbeitenden Funktionen geben die Tabellen 2.1 für die direkten sowie 2.2 für die *OpenCV*-spezifischen Implementationen.

Listing 2.4: Funktion zur Erstellung von Bildobjekten

```
TOpenCVImage*
TApplication::NewImage
( int width, int height, int channels )
{
  TOpenCVImage* pReturnValue =
    new TOpenCVImage( width, height, channels );

  fpImagePointerStack.push( pReturnValue );

  return pReturnValue;
}
```

Listing 2.5: Funktion zur Freigabe von Bildspeicher

```
void
TApplication::FreeImageMemory
()
{
  while( !fpImagePointerStack.empty() )
  {
    if( fpImagePointerStack.top() )
      delete fpImagePointerStack.top();

    fpImagePointerStack.pop();
  }
}
```

Tabelle 2.1: Die wichtigsten bildverarbeitenden Funktionen mit Implementation in der Klasse `TImage`

Funktion	Beschreibung
`Threshold()`	Komponentenweiser Schwellwert
`Convert()`	Konvertiert den Farbmodus
`Clear()`	Löscht den Bildinhalt
`SetPixel()`	Zeichnet einen Pixel
`GetPixelChannel()`	Liefert einen Kanalwert eines Pixels
`SetPixelChannel()`	Setzt einen Kanalwert eines Pixels
`Insert()`	Kopiert den Bildinhalt in einen Bereich eines anderen Bildes

Tabelle 2.2: Die wichtigsten bildverarbeitenden Funktionen mit Implementation in der Klasse `TOpenCVImage`

Funktion	Beschreibung
`Add()`	Addition zweier Bilder
`Subtract()`	Subtraktion zweier Bilder
`Copy()`	Kopiert den Bildinhalt in ein anderes Bild
`Flip()`	Spiegelt ein Bild horizontal

Die Verwendung der von *OpenCV* bereitgestellten Funktionalität zur Bildverarbeitung ist im Hinblick auf die Effizienz der entwickelten Anwendung sicherlich vorteilhaft, dennoch ist es notwendig, an einigen Funktionen Veränderungen vorzunehmen. Ein Beispiel für eine solche Funktion ist `Threshold()`, aus der unmittelbar die gewünschte *Change Mask* beispielsweise gemäß Gleichung 2.5 hervorgeht:

$$\mid D\left(X\right) \mid \underset{u}{\overset{c}{\gtrless}} \tau$$

OpenCV stellt zur Anwendung eines Schwellwertes auf ein Bild eine Funktion zur Verfügung, die lediglich Bilder mit einem Farbkanal verarbeiten kann. Zum Test und zur Auswertung der beschriebenen Verfahren ist es aber notwendig, dass der *Threshold* komponentenweise angewendet werden kann. Um dies zu erreichen, ist die Trennung eines dreikanaligen Bildes in drei einfarbige Bilder sicher nicht effizient. Stattdessen kann aufgrund der vorliegenden Klassenstruktur in der Oberklasse ein generischer *Threshold*-Algorithmus für mehrkanalige Bilder erstellt werden, der auch bei Verwendung anderer Grafik-Bibliotheken weiter benutzt werden kann.

Eine weitere wichtige Funktion der Klasse `TImage` ist die Funktion `Convert()`. Mit ihr ist es möglich, die *RGB*-Bilder der Kamera zum einen in Luminanz und Chrominanz, zum anderen in Farbton, Sättigung und Helligkeit umzuwandeln. Die Umwandlung in Luminanz und Chrominanz erfolgt dabei gemäß *ITU-R BT.601-5* (ITU, 1995) (Gleichungen 2.33ff). Der Wertebereich wurde dabei auf 0 bis 255 verändert; die additive Komponente von 127,5 dient lediglich zur Korrektur:

$$I^Y\left(X\right) = 0{,}299 \cdot I^R\left(X\right) + 0{,}587 \cdot I^G\left(X\right) + 0{,}114 \cdot I^B\left(X\right) \tag{2.33}$$

$$I^U\left(X\right) = 0{,}564 \cdot \left(I^B\left(X\right) - I^Y\left(X\right)\right) + 127{,}5 \tag{2.34}$$

$$I^V\left(X\right) = 0{,}713 \cdot \left(I^R\left(X\right) - I^Y\left(X\right)\right) + 127{,}5 \tag{2.35}$$

Die Konvertierung in Farbton, Sättigung und Helligkeit kann nicht mithilfe einer Transformationsmatrix erfolgen. Stattdessen wird der Algorithmus von Smith mit einem ebenfalls auf 0 bis 255 veränderten Wertebereich verwendet (Smith, 1978).

Abschließend ist die Funktion `Insert()` zu nennen, die zur Montage mehrerer Einzelbilder in ein Gesamtbild dient. Sie zielt dabei auf die Erstellung eines Übersichtsbildes mit Kamerabild, Zwischenschritten und *Change Mask* ab. Bei ihrem Aufruf kann sowohl die Position der linken oberen Ecke des Bildes angegeben werden, das in ein anderes eingefügt

werden soll, als auch dessen Größe. Weiterhin bietet sie die Möglichkeiten, nur einzelne Kanäle in der zugehörigen Farbe oder in Graustufen anzuzeigen und die enthaltenen Werte mit einem Faktor für das gesamte Bild zu skalieren.

2.3.8 Exemplarische Implementation eines Verfahrens

Zur Verdeutlichung der Arbeitsweise bei der Erstellung von Funktionen für neue Verfahren wird hier exemplarisch die Implementation des Trennungsverfahrens mittels komponentenweiser Subtraktion zweier Bilder im RGB-Farbraum dargestellt. Listing 2.6 gibt dabei die Funktion im Pseudocode an, der als Bauplan für Implementationen anderer Verfahren dienen kann.

Zu Beginn einer jeden Funktion zur Implementation eines Verfahrens wird ein kleines Interface in der Konsole erzeugt, anhand dessen der Benutzer das gewählte Verfahren sowie die Funktionsmöglichkeiten erkennen kann. Anschließend müssen zuerst alle benötigten Bildobjekte und danach die Kamera- sowie Anzeigeobjekte erstellt werden, da Letztere bei ihrer Erstellung die jeweils zugehörigen Bildobjekte benötigen. Zuletzt verbleibt das Programm so lange in einer Schleife, bis der Benutzer eine Eingabe zum Programmende macht. In dieser Schleife wird zunächst das aktuelle Kamerabild erfasst. Danach werden daran die einzelnen Verarbeitungsschritte durchgeführt und die gewünschten Zwischenergebnisse in das Bildobjekt zur Anzeige geschrieben. Abschließend erfolgt die Erstellung der *Change Mask* sowie ein *Refresh* der Anzeige.

2.4 Ergebnisse

Die einzelnen Verfahren wurden jeweils mit denselben drei Sequenzen von Einzelbildern getestet, die vor schwarzem, dunkelgrünem sowie weißem Hintergrund aufgenommen wurden. Für Änderungen der Beleuchtung sorgten sowohl Schattenwurf einer Person als auch Schwankungen der Sonneneinstrahlung. Zusätzlich passte sich die verwendete Kamera automatisch den gegebenen Lichtverhältnissen im Bild an. Dies wirkt sich sehr ähnlich wie Beleuchtungsveränderungen aus, kann allerdings auch Farbfehler zur Folge haben.

Die Auswertung der einzelnen Verfahren kann im Rahmen dieser Arbeit lediglich rein qualitativ visuell erfolgen, da die objektive Konstruktion einer korrekten Maske, gegenüber der Fehler quantitativ bestimmt werden können, mit immensem Zeitaufwand verbunden und sehr schwierig zu realisieren ist (Hu *et al.*, 2001; Radke *et al.*, 2005). Selbst eine Maskierung von Hand liefert nicht per se eine völlig genaue Basis: Die Maske wird sich bei Erstellung durch mehrere Personen und sogar bei mehrfacher Maskierung durch eine Person unterscheiden (Radke *et al.*, 2005). In dieser Arbeit wird für die gegebenen Verfahren beurteilt, ob Beleuchtungsschwankungen, Schattenwurf und verschiedene Hintergründe die Trennung beeinträchtigen. Weiterhin wird entschieden, ob – insbesondere für höhere *Thresholds* – die erzeugte Maske von ihrer Struktur her für eine nachgelagerte Interaktionserkennung geeignet ist. Da für die Anwendung, für die ein neues Verfahren gesucht wird, eine exakt pixelgenaue Maske nicht erforderlich ist, genügt die visuelle Auswertung der Bildergebnisse völlig.

Listing 2.6: Pseudocode zur Trennung mittels komponentenweiser RGB-Differenz

```
int
TApplication::RunSimpleRGBdifferenceThreshold
( int argc, char* const argv[] )
{
  // print some kind of UI to console for description and usage

  // create necessary image-objects

  // create camera-object and assign image

  // create display-object and assign image

  // applicaion-main-loop
  int abortLoop = 0;
  while( !abortLoop )
  {
    // key-event-handling:
    // ** ESC: abort loop
    // ** 1:   store background image

    // grab a frame

    // insert camera-image into first column of display

    // insert backgound-image into second column of display

    // subtract background-image from camera-image channel by
       å channel

    // insert result of subtraction into third column of display

    // apply threshold

    // insert result into fourth column of display

    // refresh display

  }  // END: application-main-loop

  // delete display- and camera-objects

  // delete image-objects

  return 0;

} // END: RunSimpleRGBdifferenceThreshold()
```

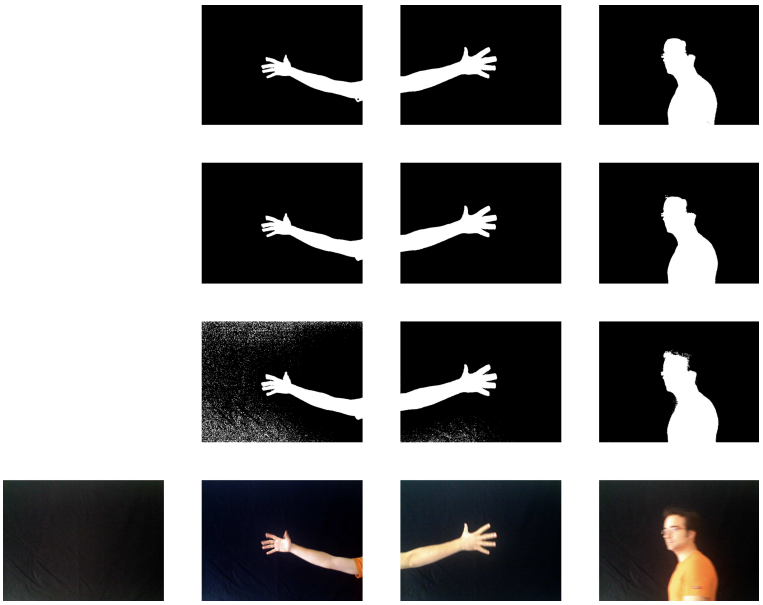


Abbildung 2.7: Ergebnis der Luminanzdifferenz vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 25, 45 und 65

2.4.1 Einfache Differenzverfahren

Zunächst werden hier die Ergebnisse der in Unterabschnitt 2.2.1 präsentierten einfachen Differenzverfahren dargestellt.

Differenz der Luminanz

Das Verfahren, das die *Change Mask* aufgrund der Luminanzdifferenz gegenüber einem gespeicherten Hintergrund bildet, erzeugte vor schwarzem Hintergrund bei kleinen *Thresholds* deutlich sichtbare Rausch-Störungen in Bereichen mit sich verändernder Helligkeit des Hintergrundes (Abbildung 2.7). Hinzu kamen eine Anfälligkeit gegenüber den eingangs benannten Farbschwankungen. Diese sind auf den ersten beiden Bildern gegenüber dem gespeicherten Hintergrund deutlich zu erkennen, wirkten sich aber unterschiedlich stark auf die Maske aus. Diese Empfindlichkeit nahm zu größeren Werten von τ hin ab, sodass dieses Verfahren beispielsweise bei einem *Threshold* von 45 eine saubere Maske erzeugte. Dabei gingen allerdings auch schon bei kleineren Werten mehr und mehr dunkle Strukturen der Vordergrundobjekte verloren. Auf schwarzem Hintergrund blieben die helleren Teile der Vordergrundobjekte auch für höhere *Thresholds* deutlich sichtbar: Es ergab sich eine saubere, vollständig ausgefüllte Maske.

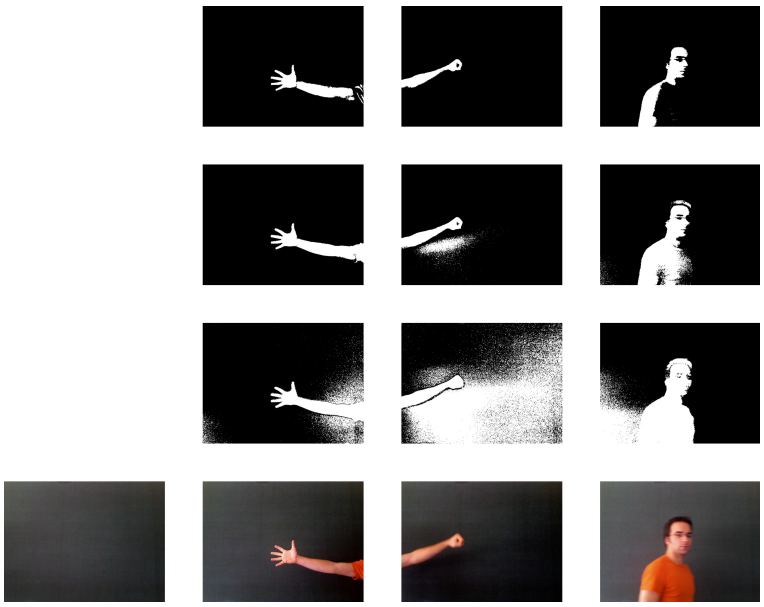


Abbildung 2.8: Ergebnis der Luminanzdifferenz vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 25, 45 und 65

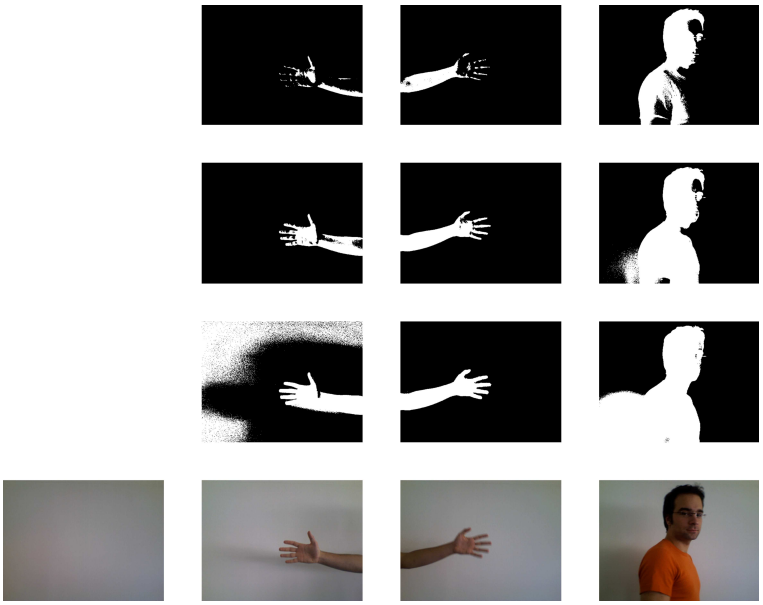


Abbildung 2.9: Ergebnis der Luminanzdifferenz vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 25, 45 und 65

Auch vor dunkelgrünem Hintergrund entstanden starke Störungen in der Maske, die hier aber beim Vergleich der Original-Bilder nicht auf Farbveränderungen, sondern auf Helligkeitsveränderungen und Schattenwurf auf dem Hintergrund zurückzuführen sind (Abbildung 2.8). Diese Störungen nahmen zwar auch wieder zu höheren Schwellwerten hin ab, aber nicht in dem Maße, wie dies vor schwarzem Hintergrund der Fall war: Erst für *Thresholds* ab etwa 50 fand die fehlerhafte Bezeichnung des beleuchteten Hintergrundes als *changed* nicht mehr statt. Dies ging allerdings einher mit starken Einbußen der Qualität der Maske: Sie erschien bei komplett unterdrückten Störungen nicht nur löchrig, hier wurden sogar bis in mittlere Helligkeitsbereiche hinein die Vordergrundobjekte nicht erkannt. Für kleine *Thresholds* war die Maske hingegen zwar noch ausgefüllt, aber für nachgelagerte Verfahren nicht von den beschriebenen Störungen zu unterscheiden.

Auch die Ergebnisse bei der Maskierung der Bildsequenz vor weißem Hintergrund zeigen die deutliche Beleuchtungsabhängigkeit dieses Verfahrens: Die aufgetretenen Änderungen in der Beleuchtung des Hintergrundes und der Schattenwurf der Vordergrundobjekte hinterließen in der Maske für kleine *Thresholds* ebenfalls starke Störungen (Abbildung 2.9). Diese Anfälligkeit konnte wiederum durch die Wahl eines höheren *Thresholds* umgangen werden, allerdings nahm sie auch für den weißen Hintergrund langsamer ab als für den schwarzen. Auch hier war die Maske für niedrige *Thresholds* zwar ausgefüllt, die vorhandenen Störungen machten eine sichere Erkennung aber unmöglich. Für höhere *Thresholds*

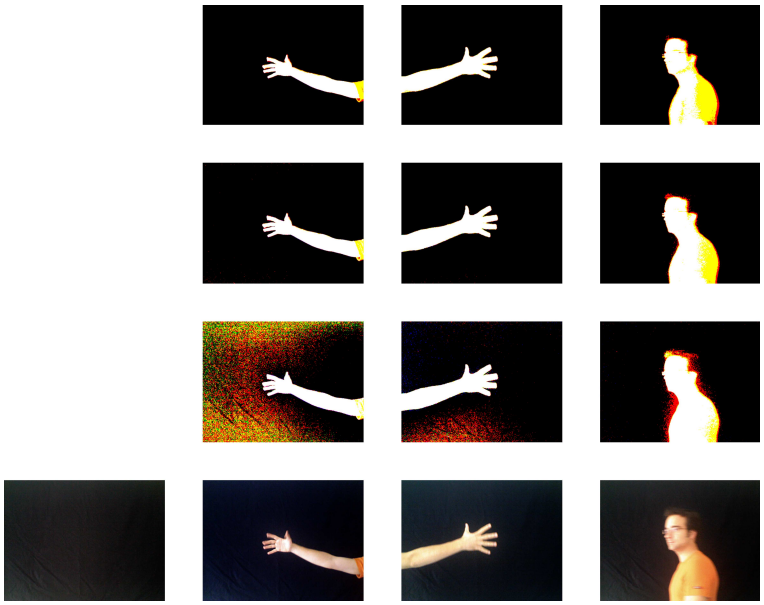


Abbildung 2.10: Ergebnis der komponentenweisen Farbdifferenz vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 25, 45 und 65

nahm die Qualität der Objektmaske rapide ab und machte bei vollständig unterdrückten Störungen eine Erkennung deshalb unmöglich.

Beim Vergleich der Masken für die einzelnen Bildsequenzen untereinander ließ sich eine deutliche Abhängigkeit dieses Verfahrens von der Helligkeit und Farbe des gewählten Hintergrundes beobachten: Bei dunklem Hintergrund wurden dunkle Vordergrundpartien für höhere *Thresholds* nicht mehr erkannt, hellere blieben sehr deutlich. Bei hellem Hintergrund wurden umgekehrt hellere Partien nicht zuverlässig maskiert. Im Allgemeinen nahm die Qualität der Vordergrundmaske mit zunehmendem *Threshold* ab. Diese Abnahme unterschied sich in ihrer Stärke für die verschiedenen Hintergründe. Ebenso wirkten sich Beleuchtungsänderungen abhängig vom Hintergrund unterschiedlich stark auf die Qualität des Trennungsergebnisses aus.

Differenz der Farbe

Auch bei der Maske, die mithilfe der komponentenweisen Differenzbildung im RGB-Farbraum entstanden ist, ließen sich bei niedrigem *Threshold* für die Bildsequenz vor schwarzem Hintergrund starke Störungen in den Hintergrundbereichen mit veränderter Helligkeits- und Farbverteilung beobachten (Abbildung 2.10). Auch hier wirkten sich die

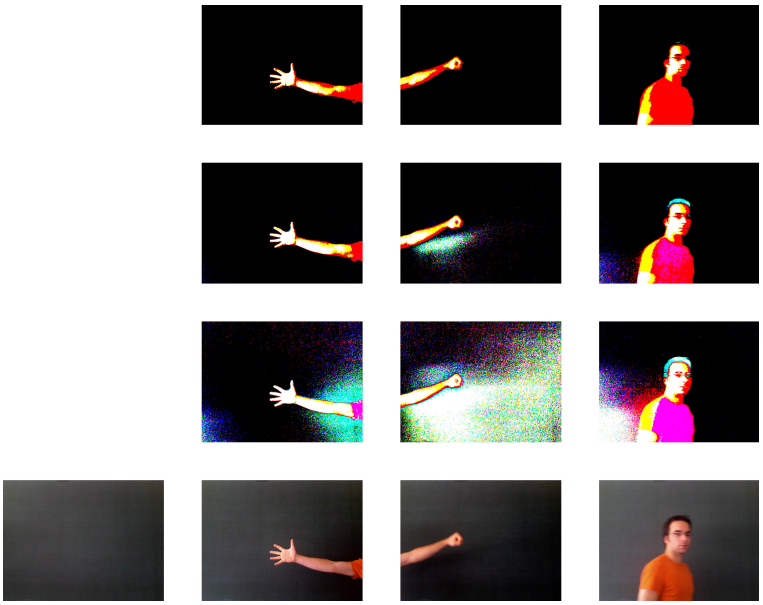


Abbildung 2.11: Ergebnis der komponentenweisen Farbdifferenz vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 25, 45 und 65

deutlich erkennbaren Farbveränderungen unterschiedlich stark auf die Maske aus. Störend für eine sichere Trennung zwischen Vorder- und Hintergrund war darüber hinaus auch ein durch hell beleuchtete, farbige Vordergrundobjekte erzeugter *Glow*. Für das letzte Bild ist dieser im Originalbild zu erkennen. Er hinterließ in der Maske für kleine *Thresholds* einen sehr unscharfen Rand, der zu Problemen bei der Interaktionserkennung führen kann. Dies wurde dadurch verstärkt, dass durch die helle Beleuchtung ein *Color Bleeding* selbst auf den schwarzen Hintergrund erfolgte und bei diesem Verfahren die Maske für den geringen *Threshold* nochmals verschlechterte. Auch bei dieser auf der Farbdifferenz basierenden Trennung konnte bei schwarzem Hintergrund für höhere Schwellwerte eine deutliche Reduktion der Anfälligkeit für die aufgetretenen Störungen beobachtet werden. Es ergab sich für den hier dargestellten Schwellwert von 45 ebenfalls eine saubere, ausgefüllte Maske, deren Qualität bei einer Steigerung des *Thresholds* zunächst nicht nennenswert nachließ. Allerdings wurden auch bei diesem Vefahren vor schwarzem Hintergrund dunkle Objektpartien nicht als Vordergrund erkannt.

Bei der Maskierung der Bildsequenz vor dunkelgrünem Hintergrund mithilfe dieses Verfahrens traten wiederum massive Störungen der Maske in Bereichen mit veränderter Beleuchtung bei kleinen Schwellwerten auf (Abbildung 2.11). Auch hier nahmen diese Störungen für höhere Werte von τ langsam ab. Diese Zunahme der Unempfindlichkeit ging

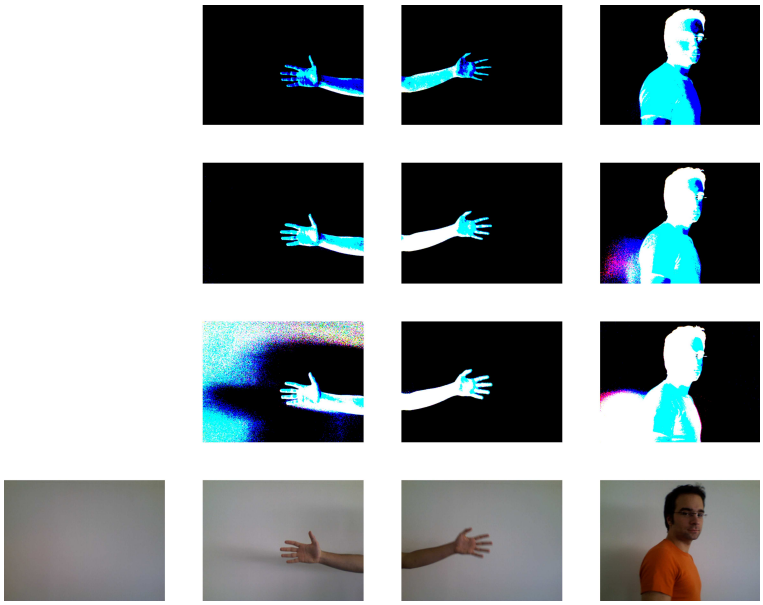


Abbildung 2.12: Ergebnis der komponentenweisen Farbdifferenz vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 25, 45 und 65

aber wie für den schwarzen Hintergrund nicht mit einer nennenswerten Verschlechterung der Maske einher, sodass sich auch hier *Thresholds* finden ließen, die störungsfreie Masken erzeugten. Dennoch: Dunkle Objektpartien wurden schon für kleine *Thresholds* dem Hintergrund zugeordnet.

Masken, die bei niedrigen *Thresholds* aus der Bildsequenz vor weißem Hintergrund entstanden waren, wiesen wiederum starke Störungen in Bereichen veränderter Hintergrundbeleuchtung sowie in Bereichen mit Schattenwurf auf (Abbildung 2.12). Wie zu erwarten war, ließen sich diese ebenfalls durch höhere Schwellwerte unterdrücken, doch gelang dies hier schlechter, da weißer Hintergrund deutlich anfälliger für wechselnde Beleuchtung und Schattenwurf ist als dunkler. Dennoch blieb bei weißem Hintergrund auch für höhere Schwellwerte die Maske noch sauber und ausgefüllt, wurde bei völliger Störungsunterdrückung aber löchrig.

Für dieses Verfahren ließ sich beim Vergleich der Masken der drei Bildsequenzen untereinander kein nennenswerter Einfluss der Helligkeit und Farbe des Hintergrundes feststellen. Beleuchtungsabhängige Störungen traten bei hellerem Hintergrund deutlicher auf als bei dunklem, aber die Qualität der erstellten Masken war bei gleichem *Threshold* für die unterschiedlichen Hintergründe ähnlich.

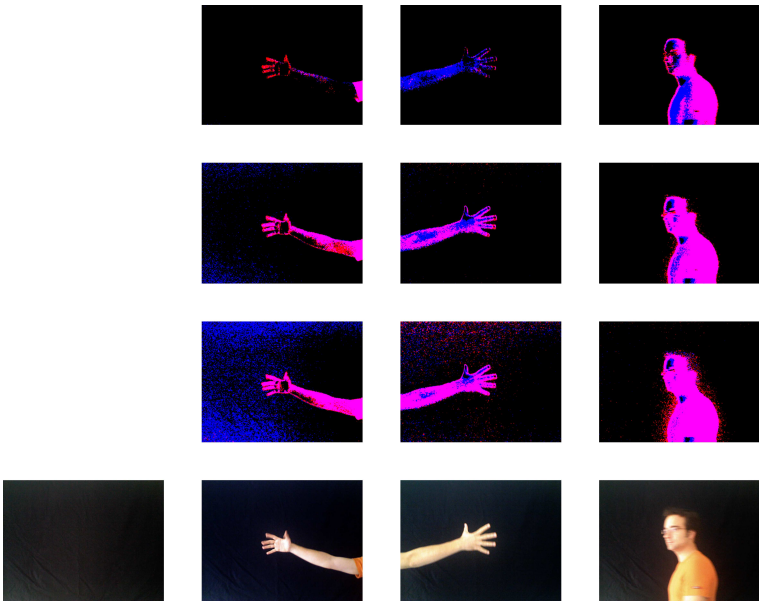


Abbildung 2.13: Ergebnis der komponentenweisen Chrominanzdifferenz vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 10, 15 und 25

Differenz der Chrominanz

Bei der Maskenbildung aus der komponentenweisen Differenz der Chrominanz vor schwarzem Hintergrund waren für kleine Werte von τ Störungen zu erkennen (Abbildung 2.13). Diese rührten wie bereits beschrieben aus Farbveränderungen aufgrund der verwendeten Kamera her. Auch das zuvor bereits erwähnte *Color Bleeding* und der *Glow* machten sich bei der Trennung des letzten Bildes störend bemerkbar. Zusätzlich zu diesen lokal begrenzten Störungen fand sich für kleine *Thresholds* ein gleichmäßig über die Maske verteiltes Rauschen, das in der Abbildung lediglich aufgrund drucktechnischer Gegebenheiten nicht zu erkennen ist. Diese Störungen konnten durch eine Erhöhung des Schwellwertes sehr gut unterdrückt werden. Dies galt besonders für das Rauschen. Vor schwarzem Hintergrund hatte diese Erhöhung allerdings zur Folge, dass mehr und mehr Anteile der Maske des Vordergrundobjektes verlorengingen: Zunächst fehlten bei niedrigen Schwellwerten lediglich die dunklen Objektteile, später wurden auch hellere Bereiche nicht mehr sicher vom Hintergrund unterschieden, und die Maske wurde löchrig oder ging vollständig verloren.

Wurde dieses Trennungsverfahren auf die Bildsequenz vor dunkelgrünem Hintergrund angewendet, ergaben sich schon für niedrige Schwellwerte – abgesehen vom schon beschriebenen Rauschen – keine nennenswerten Störungen (Abbildung 2.14). Selbst Bereiche, in denen die Beleuchtung schwankte, wurden sicher als Hintergrund erkannt. Im letzten Bild

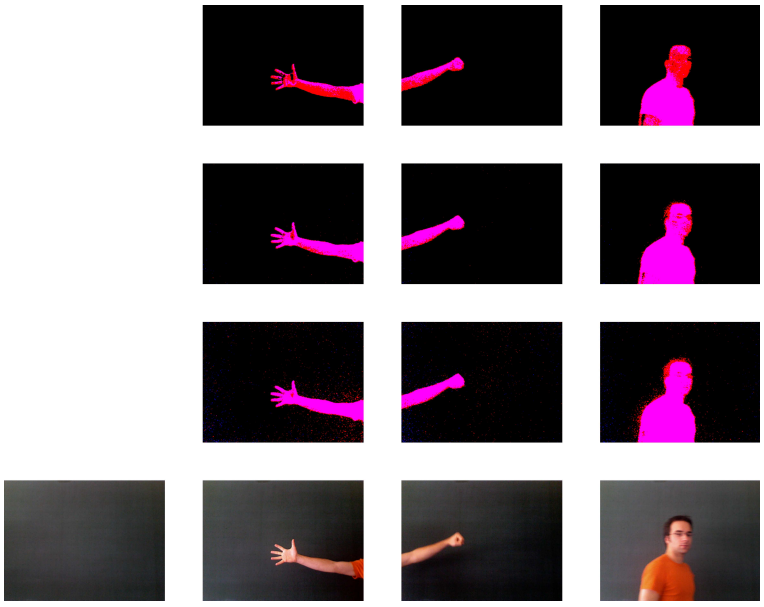


Abbildung 2.14: Ergebnis der komponentenweisen Chrominanzdifferenz vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 10, 15 und 25

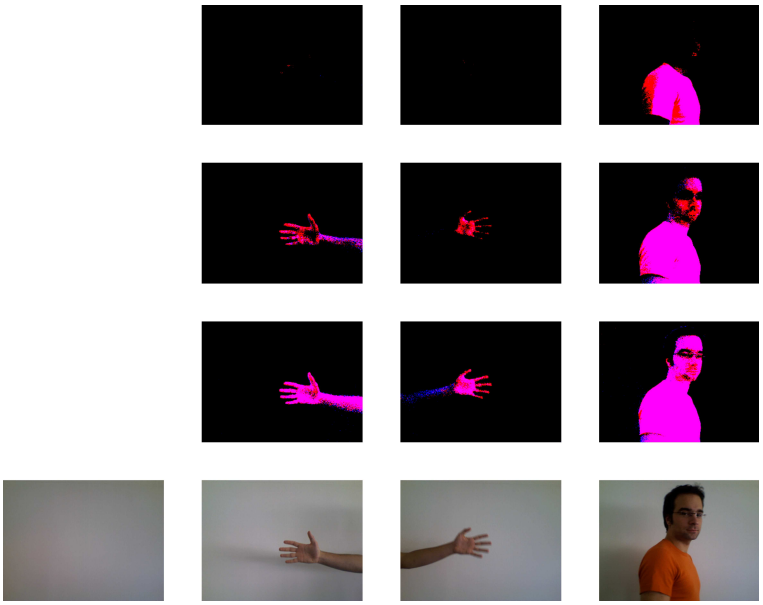


Abbildung 2.15: Ergebnis der komponentenweisen Chrominanzdifferenz vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 10, 15 und 25

ließen sich für den niedrigen *Threshold* die bereits beschriebenen Auswirkungen eines hell beleuchteten, farbigen Objektes erkennen. Diese Störungen konnten wiederum gut mithilfe einer Erhöhung von τ unterdrückt werden. Dabei wies die Maske insgesamt eine große Stabilität auf: Abgesehen vom Verlust dunkler Teile des Vordergrundobjektes wurde auch für den höchsten dargestellten *Threshold* das gesamte Vordergrundobjekt korrekt erkannt und durch eine gefüllte Maske dargestellt.

Die Trennung der Bildsequenz vor weißem Hintergrund lieferte für kleine Schwellwerte lediglich sporadisch auftretende Störungen, die von den Bildern in Abbildung 2.15 allerdings nicht erfasst sind. Diese traten in Bereichen besonders starker Abschattung des Hintergrundes durch ein Vordergrundobjekt auf, konnten aber durch Erhöhung des *Thresholds* vermieden werden. Eine solche Störungsunterdrückung hatte für die verwendete Bildsequenz aber zur Folge, dass die Qualität der Maske deutlich abnahm. Diese war – abhängig von Farbe und Helligkeit des Vordergrundobjektes – schon bei niedrigen *Thresholds* teilweise löchrig. Für höhere Schwellwerte wurde wiederum farbabhängig das Vordergrundobjekt nicht mehr erkannt. Auch vor weißem Hintergrund wurden dunkle Teile des Vordergrundobjektes schlecht bis gar nicht erkannt.

Ein Vergleich der Trennungsergebnisse der Bildsequenzen untereinander zeigte für dieses Verfahren eine Abhängigkeit der Maskenqualität vom verwendeten Hintergrund: Die

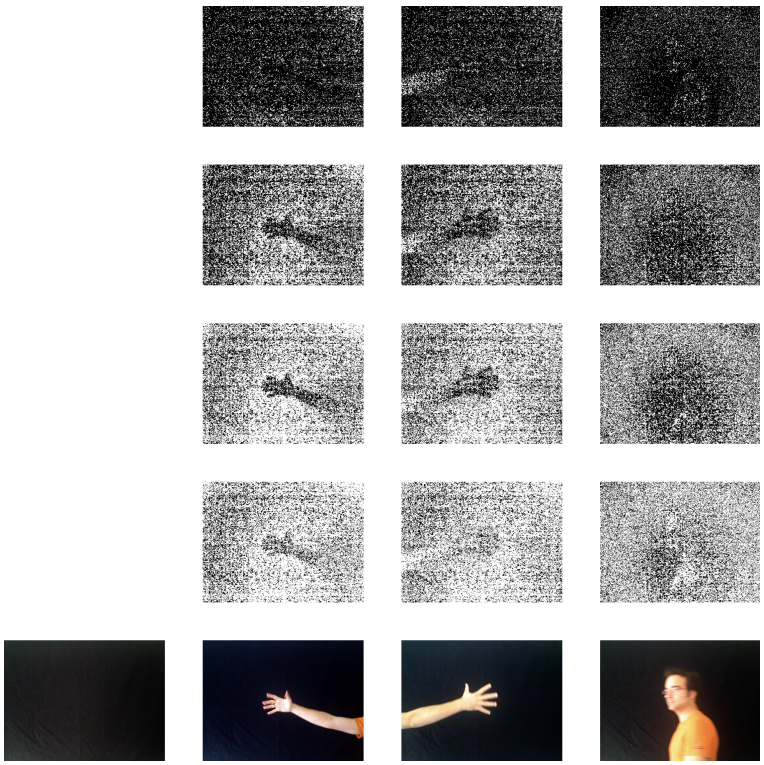


Abbildung 2.16: Ergebnis der Farbtondifferenz vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die vier *Thresholds* 25, 45, 65 und 100

Trennung vor schwarzem und vor weißem Hintergrund lieferten ähnliche Masken. Diese waren aber deutlich schlechter als die, die bei dunkelgrünem Hintergrund entstanden waren.

Differenz des Farbtons

Das getestete Verfahren zur Trennung von Vorder- und Hintergrund mittels pixelweiser Subtraktion der Farbton-Werte von aktuellem und gespeichertem Bild ergab sowohl für schwarzen als auch für dunkelgrünen Hintergrund lediglich Maskierungsergebnisse, die ein annähernd gleichmäßig verteiltes Rauschen enthielten, dessen Dichte zu höheren Werten von τ hin abnahm (Abbildungen 2.16 und 2.17). Eine klare Unterscheidung zwischen Vorder- und Hintergrund war nicht möglich.

Bei der Verarbeitung der Bildsequenz mit weißem Hintergrund trat in großen Teilen des Bildes ebenfalls Rauschen auf, dessen Dichte sich aber in etwa an der Bildhelligkeit

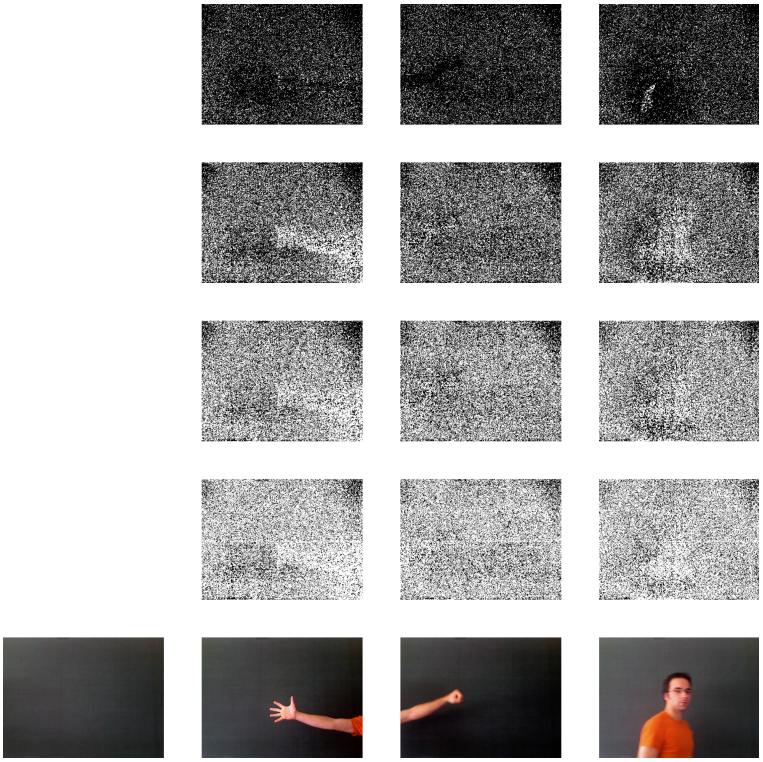


Abbildung 2.17: Ergebnis der Farbtondifferenz vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die vier *Thresholds* 25, 45, 65 und 100

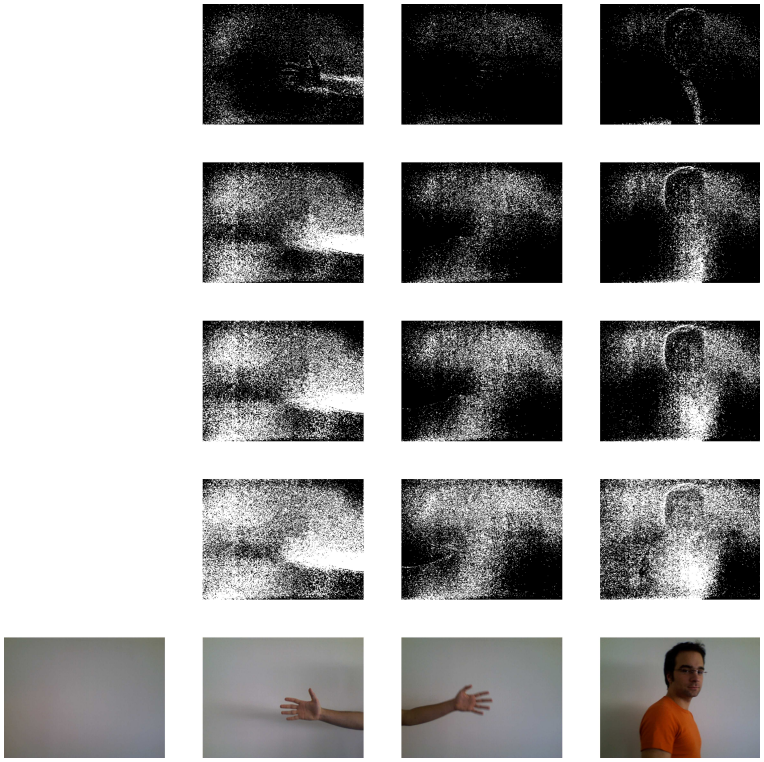


Abbildung 2.18: Ergebnis der Farbtondifferenz vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die vier *Thresholds* 25, 45, 65 und 100

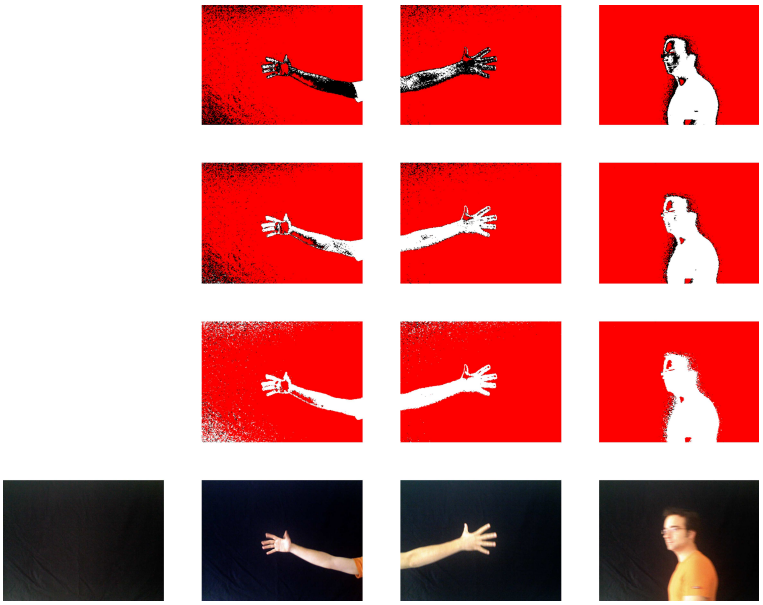


Abbildung 2.19: Ergebnis der Chrominanzabweichung vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 10, 15 und 25 – das entspricht den quadratischen Schwellwerten von 100, 225 und 625. Als Sättigungsuntergrenze wurde ein Wert von 20 gewählt. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

orientierte (Abbildung 2.18). Eine zusätzliche Verdichtung wurde für die Bereiche des Vordergrundobjektes beobachtet. Auch hier nahm das Rauschen für höhere Werte von τ ab – damit aber auch die Dichtehäufung an Vordergrundobjekten, sodass vor weißem Hintergrund eine Unterscheidung ebenfalls nicht möglich ist. Auch für eine Kamera der Firma *Unibrain* wurden ähnliche Ergebnisse erzielt.

2.4.2 Chrominanzabweichung

Bei der Trennung von Vorder- und Hintergrund durch Vergleich des euklidischen Abstands zweier Chrominanzen mit einen *Threshold* ergaben sich für die Bildsequenz vor schwarzem Hintergrund bei kleinen Werten für τ^2 wiederum deutliche Störungen aus Farbschwankungen im Hintergrund, die fälschlicherweise als Vordergrund interpretiert wurden (Abbildung 2.19). Auch die bereits aufgezeigten Auswirkungen des im letzten Bild hell beleuchteten, farbigen Vordergrundobjektes machten sich wieder in der Maske bemerkbar. Gleichzeitig wurden durch die Wahl einer Sättigungsuntergrenze große Flächen der Maske als *indeterminate* angegeben. Diese Kennzeichnung erfolgt zwar für große Teile des schwarzen Hintergrundes, aber auch einige Teile des Vordergrundobjektes wurden als unbestimmbar

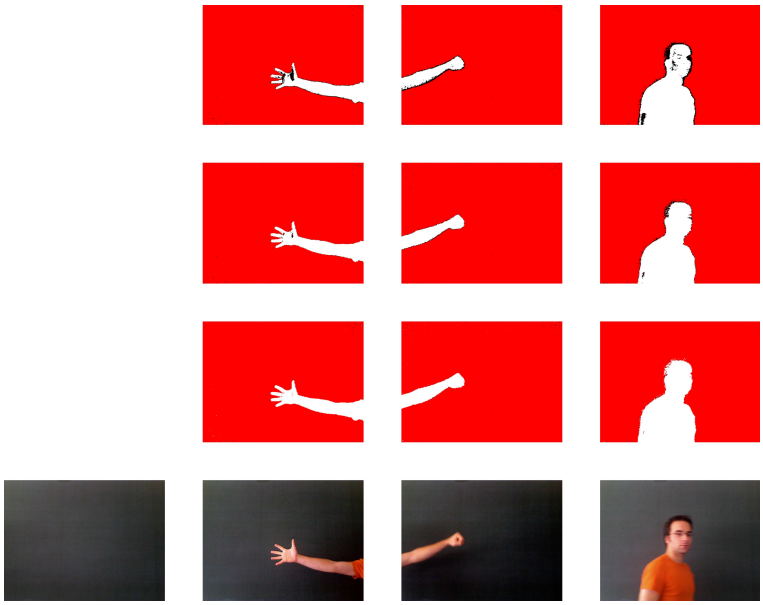


Abbildung 2.20: Ergebnis der Chrominanzabweichung vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 10, 15 und 25 – das entspricht den quadratischen Schwellwerten von 100, 225 und 625. Als Sättigungsuntergrenze wurde ein Wert von 20 gewählt. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

klassfiziert. Zu höheren *Thresholds* hin nahmen die Störungen ab. Dabei blieben die Bereiche, die von diesem Verfahren nicht bestimmt werden konnten, unverändert. Die Maske des Vordergrundobjektes, in der schon für niedrige Schwellwerte dunkle Objektteile als Hintergrund oder *indeterminate* eingeordnet wurden, verlor bei verbesserter Störanfälligkeit mehr und mehr an Qualität: Sie wurde löchrig. Weiterhin waren bei der verwendeten Bildsequenz überstrahlte Bereiche des Vordergrundobjektes nicht bestimmbar.

Eine Anwendung dieses Trennungsverfahrens auf die Bildsequenz vor dunkelgrünem Hintergrund ergab schon für niedrige Schwellwerte keine nennenswerten Störungen durch die – für diese Sequenz bereits beschriebenen – Beleuchtungsschwankungen auf dem Hintergrund (Abbildung 2.20). Die gewählte Sättigungsuntergrenze bewirkte auch bei diesem Hintergrund, dass große Flächen der Maske als unbestimmbar klassifiziert wurden. Die zwischenzeitlich noch vorhandenen Störungen konnten durch eine Erhöhung von τ^2 unterdrückt werden. Dabei war die Maske unanfällig gegenüber der Vergrößerung des Schwellwertes. Zwar gingen vor dunkelgrünem Hintergrund dunkle Objektteile nicht in die Vordergrundmaske ein, aber auch für höhere *Thresholds* wurde nahezu das gesamte übrige Vordergrundobjekt korrekt erkannt und durch eine gefüllte Maske dargestellt.

Dieses Trennungsverfahren lieferte für die Bildsequenz vor weißem Hintergrund schon

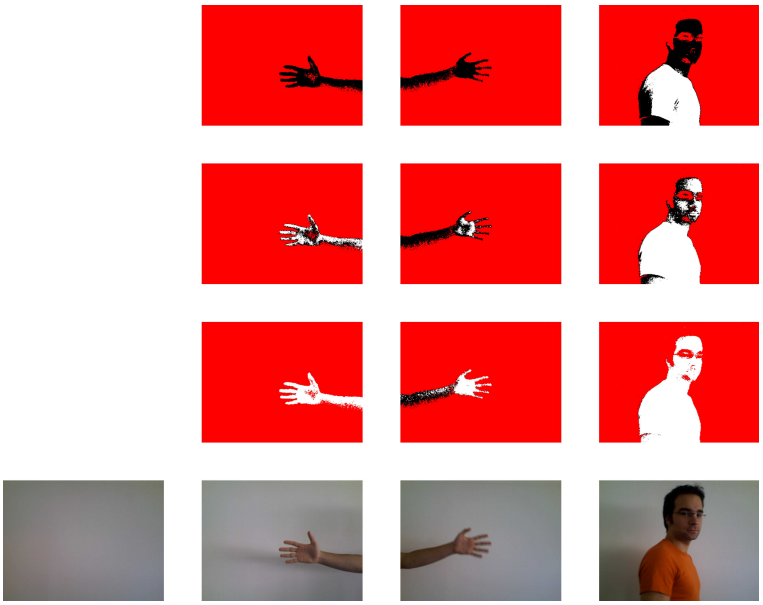


Abbildung 2.21: Ergebnis der Chrominanzabweichung vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 10, 15 und 25 – das entspricht den quadratischen Schwellwerten von 100, 225 und 625. Als Sättigungsuntergrenze wurde ein Wert von 20 gewählt. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

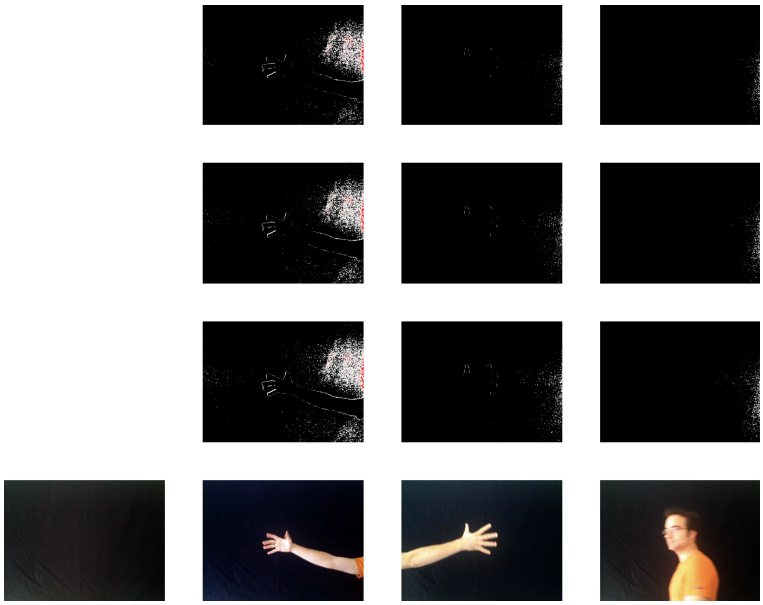


Abbildung 2.22: Ergebnis der Luminanzvektor-Abweichung vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 0,098, 0,118 und 0,138, die Winkeln von etwa 25,6 °, 28,1 ° und 30,4 ° entsprechen. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

bei der Wahl kleiner Schwellwerte keine nennenswerten Störungen mehr (Abbildung 2.21). Auch die Schwankungen der Hintergrundbeleuchtung, die u. a. durch Schattenwurf entstanden, wurden nicht als *changed* eingestuft. Eine Erhöhung der Werte für τ^2 bewirkte eine rapide Abnahme der Maskenqualität: Je dunkler ein Element des Vordergrundobjektes war, desto eher wurde es nicht mehr erkannt. Auch bei weißem Hintergrund ergaben sich durch das beschriebene Verfahren große Flächen der Maske als *indeterminate*.

Ein Vergleich der Masken für die drei Bildsequenzen untereinander – und damit für drei verschiedene Hintergründe – zeigte, dass das Ergebnis der Trennung bei einem weißen und einem schwarzen Hintergrund bei gleichen *Thresholds* ähnlich war. Wiederum waren die Masken, die aus der Sequenz vor dunkelgrünem Hintergrund entstanden, deutlich besser.

2.4.3 Abweichung des Luminanzvektors

Das in Unterabschnitt 2.2.3 beschriebene Verfahren, das die Entscheidung zwischen *changed* und *unchanged* aufgrund einer Winkelabweichung zwischen zwei normalisierten Luminanzvektoren trifft, lieferte für die Bildsequenz vor schwarzem Hintergrund bei kleinen Schwellwerten deutlich sichtbare Störungen (Abbildung 2.22). Vom Vordergrundobjekt

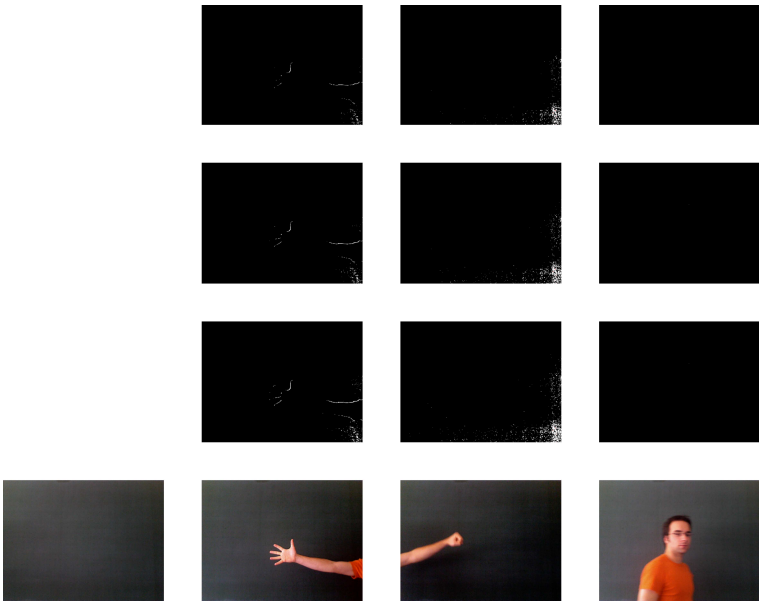


Abbildung 2.23: Ergebnis der Luminanzvektor-Abweichung vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 0,098, 0,118 und 0,138, die Winkeln von etwa 25,6 °, 28,1 ° und 30,4 ° entsprechen. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

wurde lediglich die Konturlinie korrekt erkannt, die aber abhängig von der Helligkeit des Objektes unterschiedlich deutlich ausgeprägt war. Das beschriebene *Color Bleeding* sowie der *Glow* im letzten Bild der Abbildung verursachten im Originalbild einen starken Helligkeitsverlauf anstelle einer harten Objektkante. Dadurch wurde in der zugehörigen Maske selbst für kleine *Thresholds* keine Kontur erkannt. Die Störungen in den Masken veränderten sich bei der Bildsequenz auf schwarzem Hintergrund für die getesteten höheren Schwellwerte nicht nennenswert. Stattdessen wurde die Objektkontur immer schlechter maskiert. Aufgrund der Bedingung, dass die Luminanzvektoren für die Normalisierung eine von Null verschiedene Länge aufweisen müssen, ergaben sich in Randbereichen kleine Flächen, die als *indeterminate* klassifiziert wurden.

Auch bei der Verarbeitung der Bildsequenz mit grünem Hintergrund durch dieses Verfahren ergaben sich bei kleinen Werten für τ Störungen in der Maske, die ebenfalls zu höheren Werten hin nicht wesentlich abnahmen (Abbildung 2.23). Wiederum zeigte sich, dass dieses Verfahren allenfalls die Objektkontur maskiert – bei dieser Bildsequenz allerdings nur an Kanten, die einen sehr großen Helligkeitssprung aufwiesen. Wie die Masken, die aus dem mittleren und letzten Bild hervorgegangen sind, zeigen, enthielt das Trennungsergebnis bei geringen Kontrasten keine Information über das Vordergrundobjekt.

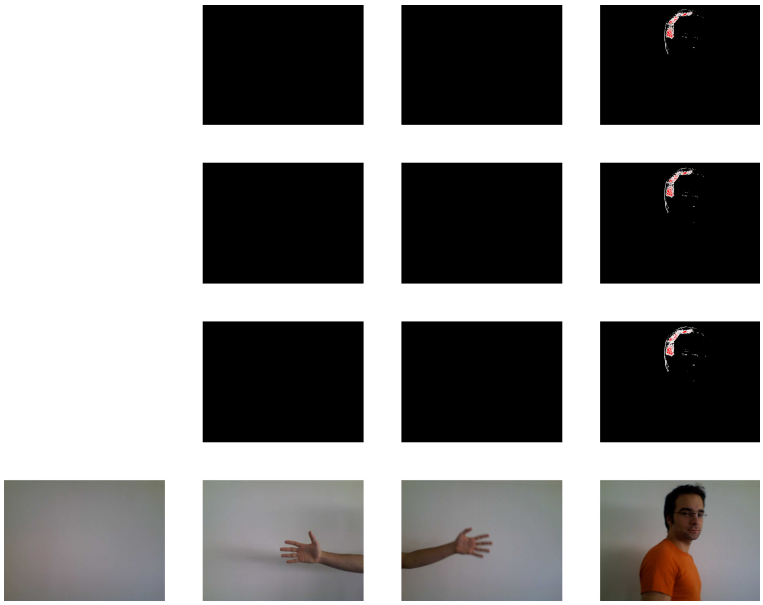


Abbildung 2.24: Ergebnis der Luminanzvektor-Abweichung vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 0,098, 0,118 und 0,138, die Winkeln von etwa 25,6 °, 28,1 ° und 30,4 ° entsprechen. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

Bei den Tests zur Trennung für die Bildsequenz mit weißem Hintergrund waren selbst für kleine *Thresholds* keine Störungen in den Masken zu erkennen, die auf Anfälligkeit gegenüber Schattenwurf oder veränderter Beleuchtung hindeuten (Abbildung 2.24). Dennoch zeigte sich auch hier, dass nur solche Teile der Objektkontur erkannt wurden, die eine große Helligkeitsänderung gegenüber dem Hintergrund darstellten: In diesem Fall wurden lediglich die Kanten zu sehr dunklen Teilen des Vordergrundobjektes maskiert, kleinere Helligkeitssprünge – die für beinahe sämtliche Teile des Vordergrundobjektes auftraten – wurden überhaupt nicht erkannt. Dabei veränderte sich dieses Ergebnis zu höheren *Thresholds* hin zunächst nicht.

Beim Vergleich der Ergebnisse des Verfahrens für die verschiedenen Hintergründe untereinander ergab sich unabhängig vom Hintergrund ein ähnlich schlechtes Ergebnis für alle Masken: Nur an deutlichen Helligkeitssprüngen wurde eine Kontur erkannt. Die in den Bildsequenzen enthaltenen Helligkeitssprünge sind aber nicht ausgeprägt genug, um eine sichere Konturbildung zu ermöglichen.

Für das gemäß Unterabschnitt 2.2.4 um zusätzliche Schwellwerte für die Vektorlänge sowie das Längenverhältnis ergänzte Verfahren ergaben sich andere Testergebnisse. Aufgrund der Maskierungsergebnisse des vorigen Versuchs wurden hier nicht mehr verschiedene Abweichungs-*Thresholds*, sondern unterschiedliche Schwellwerte für das Längenver-

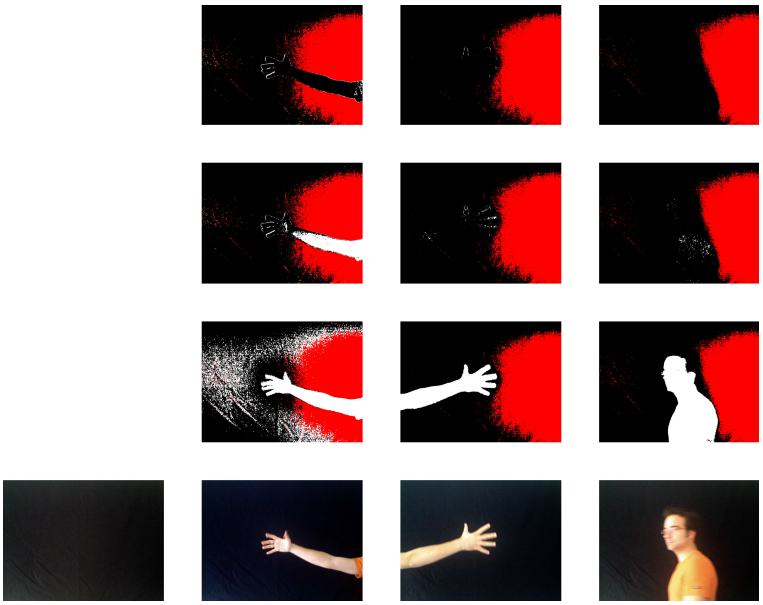


Abbildung 2.25: Ergebnis der Luminanzvektor-Abweichung mit Längenverhältnis vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_U^{MR}
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei Schwellwerte τ_U^{MR} des Längenverhältnisses von 3,5, 10,0 und 35,0. Als *Threshold* τ wurde ein Wert von 0,098 gewählt, der einem Winkel von etwa 25,6 ° entspricht. Als Mindestlänge der Vektoren wurde 50 angenommen. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

hältnis der Vektoren erprobt. Um dennoch eine Vergleichbarkeit zu schaffen, wurde als Schwellwert der Winkelabweichung derselbe verwendet, der die untere Reihe der Masken beim zuvor getesteten Verfahren erzeugte.

Vor schwarzem Hintergrund ergaben sich zunächst aufgrund der Mindestbedingung für die Vektorlänge große Flächen, die als unbestimmbar eingestuft wurden (Abbildung 2.25). Innerhalb dieser Flächen befanden sich die beim vorigen Test festgestellten Störungen, die nun zumindest keine fehlerhafte Klassifizierung als Vordergrund zur Folge hatten. Dennoch traten aufgrund der Weiterverarbeitung der zuvor als Hintergrund bezeichneten Pixel neue Störungen auf. Diese zeigten sich jedoch hauptsächlich bei niedrigen Werten für τ_U^{MR} in den Bereichen, die bei allen vorherigen Tests vor schwarzem Hintergrund schon durch Störungen aufgrund von Farbveränderungen der Kamera auffielen. Im Allgemeinen ergab sich hier aber für niedrige *Thresholds* eine klar abgegrenzte, ausgefüllte Maske, die allerdings sehr dunkle Objektpartien nicht beinhaltete. Zu höheren Schwellwerten für das Längenverhältnis hin wurde diese Maske dann aber – durch den Verlust der Erkennung großer Teile der Vordergrundobjekte – schnell unbrauchbar.

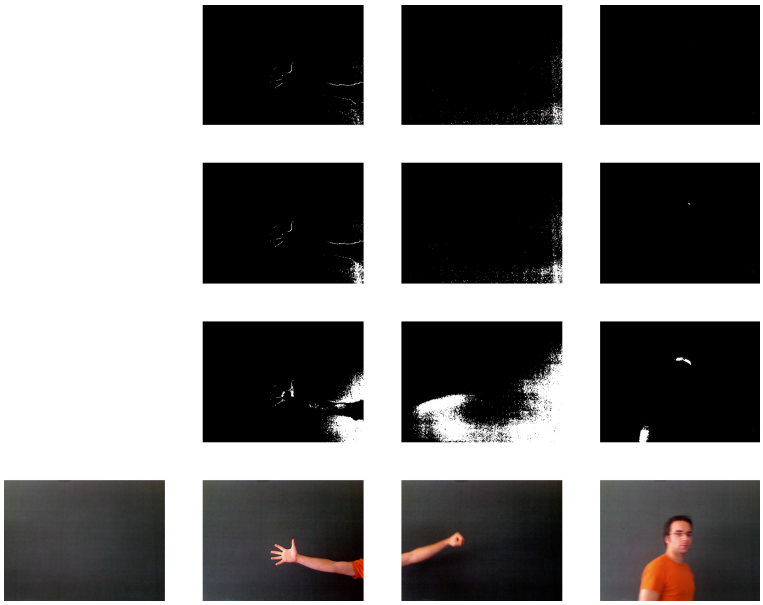


Abbildung 2.26: Ergebnis der Luminanzvektor-Abweichung mit Längenverhältnis vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_U^{MR}
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei Schwellwerte τ_U^{MR} des Längenverhältnisses von 3,5, 10,0 und 35,0. Als *Threshold* τ wurde ein Wert von 0,098 gewählt, der einem Winkel von etwa 25,6 ° entspricht. Als Mindestlänge der Vektoren wurde 50 angenommen. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

Bei der Verarbeitung der Bildsequenz vor dunkelgrünem Hintergrund mithilfe des abgewandelten Verfahrens ergaben sich bei kleinen Schwellwerten für das Längenverhältnis der Vektoren neue massive Störungen vor allem in Bereichen mit veränderter Beleuchtung und Schattenwurf (Abbildung 2.26). Bei dieser Bildsequenz traten in den Masken aufgrund der Helligkeitsverteilung in den einzelnen Bildern keine unbestimmbaren Elemente auf. Die beleuchtungsabhängigen Störungen ließen sich durch höhere *Thresholds* lediglich auf ein Maß reduzieren, das zuvor schon für das nicht abgewandelte Verfahren erzielt wurde. Die zuvor bei schwarzem Hintergrund erhaltene Füllung der Maske durch die Verfahrensabwandlung konnte bei dunkelgrünem Hintergrund nicht beobachtet werden: Es ergaben sich insgesamt keine brauchbaren Masken.

Wie zuvor für das unveränderte Verfahren traten auch hier vor weißem Hintergrund keine beleuchtungsabhängigen Störungen bei den gewählten *Thresholds* auf (Abbildung 2.27). Auch hier konnten alle Pixel eindeutig entweder als *changed* oder *unchanged* bestimmt werden. Eine durch die Veränderung des Verfahrens erfolgte Füllung der Maske konnte nur für die Elemente der Vordergrundobjekte erreicht werden, die sich in ihrer

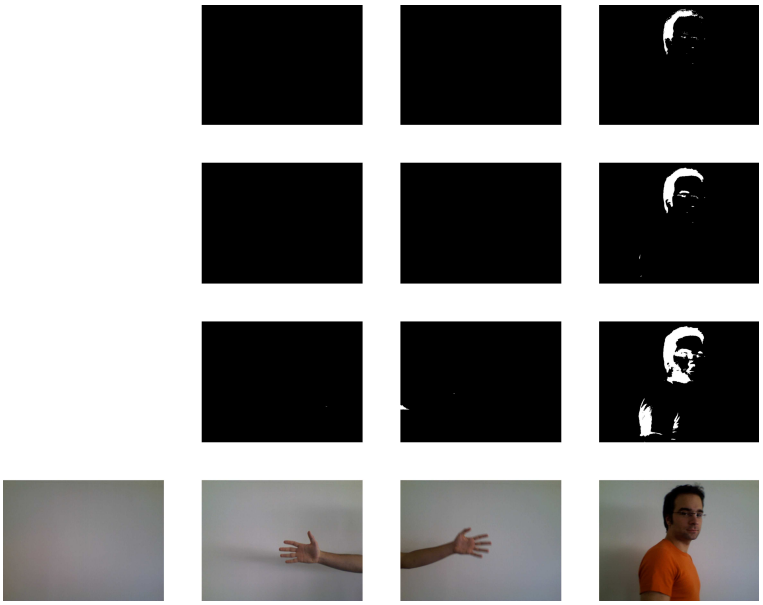


Abbildung 2.27: Ergebnis der Luminanzvektor-Abweichung mit Längenverhältnis vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_U^{MR}
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei Schwellwerte τ_U^{MR} des Längenverhältnisses von 3,5, 10,0 und 35,0. Als *Threshold* τ wurde ein Wert von 0,098 gewählt, der einem Winkel von etwa 25,6 ° entspricht. Als Mindestlänge der Vektoren wurde 50 angenommen. Rote Flächen in den Masken geben die Kennzeichnung als *indeterminate* an.

Helligkeit deutlich vom Hintergrund unterschieden: Nur dunkle Teile der Vordergrundobjekte waren in der Maske erkennbar. Diese Maskierung verschlechterte sich zu höheren *Thresholds* hin noch weiter.

Beim Vergleich der einzelnen Bildergebnisse für die verschiedenen Hintergründe untereinander ergaben sich – abhängig von der Hintergrundfarbe – unterschiedlich starke Störungen durch Beleuchtung und Schattenwurf. Auch die Füllung der Objektmasken war abhängig vom Hintergrund.

Ein Vergleich der ergänzten Methode mit dem ursprünglichen Verfahren zeigte eine Veränderung, aber keine nennenswerte Verbesserung der Masken. Der hinzugefügte Verarbeitungsschritt der als Hintergrund klassifizierten Pixel bewirkte zwar teilweise eine Füllung der Maske, führte andererseits aber auch zu neuen, oft massiven Störungen.

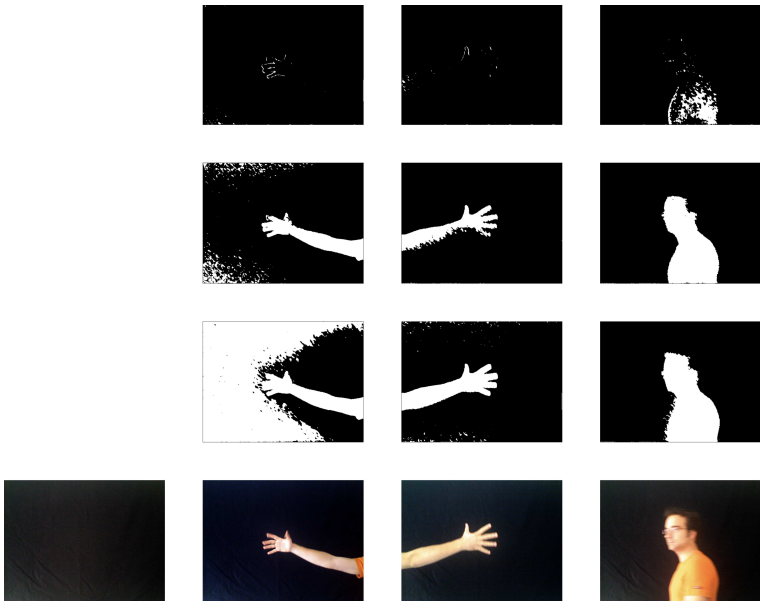


Abbildung 2.28: Ergebnis des blockbasierten Kollinearitätstests im RGB-Farbraum vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_s
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 2500, 4500 und 6500 sowie 200 für die positive Konstante C und einen Wert von 76 für $\sqrt{O_{dc}}$, die Kompensation für dunkle Bereiche

2.4.4 Blockbasierter Kollinearitätstest mittels Bayes'scher Entscheidung und Markov-Zufalls-Feldern

Tests des Verfahrens des blockbasierten Kollinearitätstests nach Mester *et al.* mit der von Griesser *et al.* vorgeschlagenen Verwendung der Farbe und zusätzlicher *Darkness Compensation* ergaben für die Bildsequenz mit schwarzem Hintergrund für kleine Werte von τ_s wiederum die zuvor beschriebenen Störungen, die von den Farbveränderungen der Kamera während dieser Bildsequenz herrührten (Abbildung 2.28) (Mester *et al.*, 2001; Griesser *et al.*, 2005). Aufgrund des hier gewählten Wertes der positiven Konstante fiel die fehlerhafte Maskierung dieser Störungen sehr massiv aus. Diese Störungen ließen sich zu größeren Schwellwerten hin deutlich reduzieren. Für kleine *Thresholds* ergab sich eine deutliche, vollständig ausgefüllte Maske für das Vordergrundobjekt, dessen dunkle Stellen wiederum nicht erkannt werden konnten. Der zuvor schon aufgefallene *Glow* und das *Color Bleeding* im letzten gezeigten Bild verursachten keine glatte, sondern eine stark gezackte Maske. Für die übrigen Bilder der Sequenz ergab sich eine deutliche, ausgefüllte Maske. Bei einer Erhöhung von τ_s ging zunächst die Konturschärfe der Maske verloren, später wurde sie löchrig.

Bei der Maskierung der Bildsequenz vor dunkelgrünem Hintergrund ergaben sich für

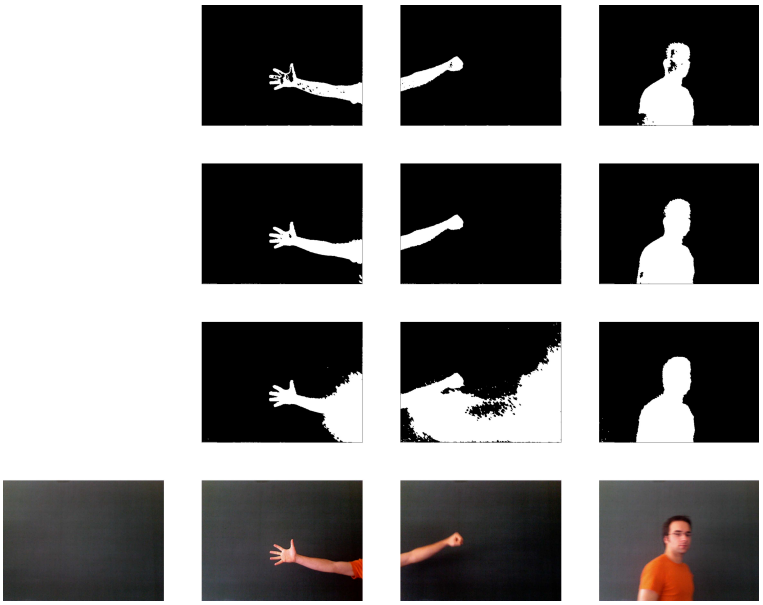


Abbildung 2.29: Ergebnis des blockbasierten Kollinearitätstests im RGB-Farbraum vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_s
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 2500, 4500 und 6500 sowie 200 für die positive Konstante C und einen Wert von 76 für $\sqrt{O_{dc}}$, die Kompensation für dunkle Bereiche

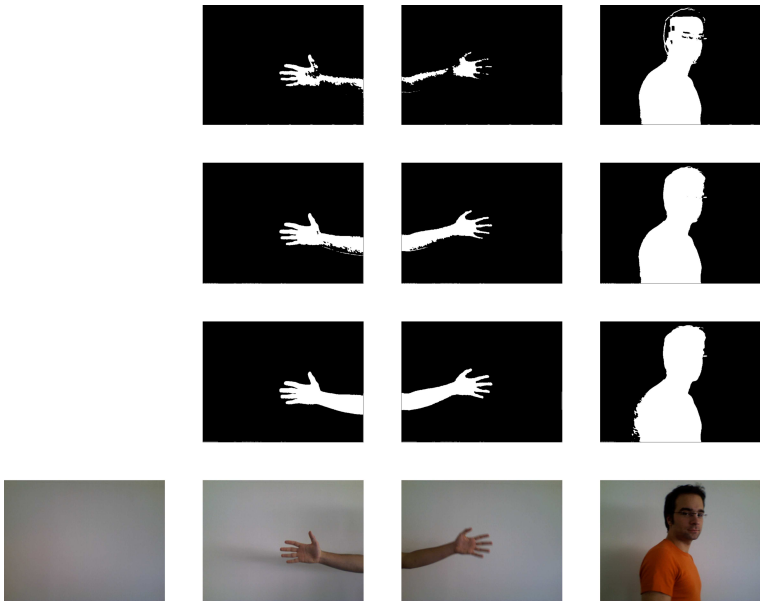


Abbildung 2.30: Ergebnis des blockbasierten Kollinearitätstests im RGB-Farbraum vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_s
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 2500, 4500 und 6500 sowie 200 für die positive Konstante C und einen Wert von 76 für $\sqrt{O_{dc}}$, die Kompensation für dunkle Bereiche

niedrige *Thresholds* die ebenfalls zuvor schon beobachteten Störungen durch stark schwankende Beleuchtung und Schattenwurf der Vordergrundobjekte (Abbildung 2.29). Diese Störungen ließen sich wiederum durch die Wahl größerer *Thresholds* unterdrücken. Für kleine Schwellwerte ergab sich eine – bis auf die angrenzenden störungsbedingten Fehler – saubere, scharf begrenzte, ausgefüllte Maske. Ihre Qualität nahm zunächst mit größer werdendem τ_s nicht nennenswert ab, sodass auch störungsfreie, scharfe, ausgefüllte Masken erzeugt werden konnten. Mit weiterer Erhöhung wurde die Maske zunehmend löchrig. Bei der Anwendung dieses Verfahrens auf die Bildsequenz vor dunkelgrünem Hintergrund wurden auch dunkle Partien der Vordergrundobjekte sicher maskiert.

Die Masken, die bei den Tests des Verfahrens aus der Bildsequenz mit weißem Hintergrund entstanden sind, zeigten bei kleinen *Thresholds* keine nennenswerten Störungen durch Schattenwurf oder wechselnde Beleuchtung – außer in Bereichen sehr großer Helligkeitsabweichung (Abbildung 2.30). Diese ließen sich aber sehr leicht durch größere Schwellwerte vermeiden; allerdings nahm auch hierbei die Qualität der Vordergrundmaske ab: Anfangs wurde das gesamte Vordergrundobjekt sicher maskiert, danach wurden als erstes die dunklen Bereiche nicht mehr erkannt.

Ein Vergleich der Maskierungsergebnisse für die verschiedenen Hintergründe ergab ei-

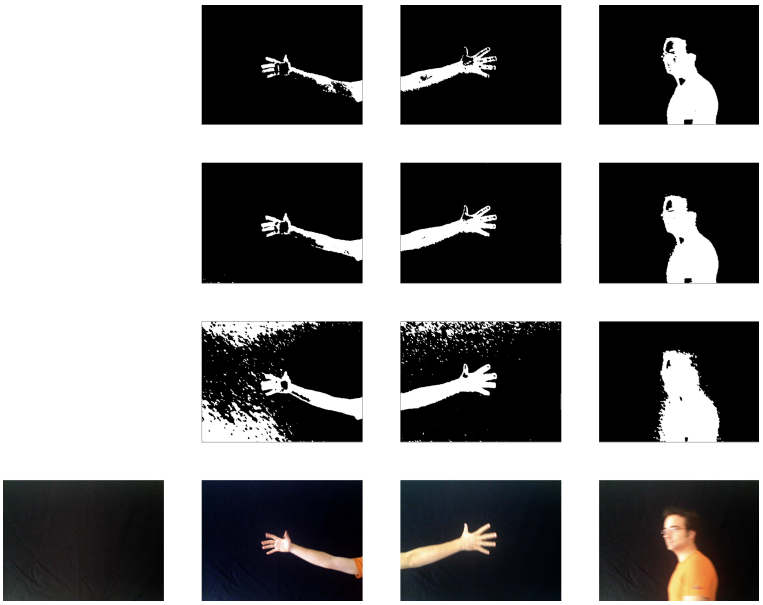


Abbildung 2.31: Ergebnis des veränderten blockbasierten Kollinearitätstests in der Chrominanzebene vor schwarzem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_s
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 550, 1650 und 2750 sowie 80 für die positive Konstante C und einen Wert von 76 für $\sqrt{O_{dc}}$, die Kompensation für dunkle Bereiche

ne hintergrundsabhängige Störanfälligkeit für Beleuchtungsschwankungen bei niedrigen *Thresholds*. Für alle Hintergründe wurde dabei das Vordergrundobjekt durch eine ausgefüllte Maske dargestellt. Die Abnahme ihrer Qualität war für alle Hintergründe zu höheren Schwellwerten ähnlich.

Die Tests der in dieser Arbeit vorgeschlagenen Änderung des Verfahrens nach Griesser *et al.* in eine Chrominanz-basierte Erkennung erzeugte aus der Bildsequenz vor schwarzem Hintergrund wiederum die schon bekannten Störungen (Griesser *et al.*, 2005) (Abbildung 2.31). Diese Störungen ließen sich durch die Wahl höherer Werte für τ_s reduzieren. Dabei wurde die Maske des Vordergrundobjektes an Stellen geringer Sättigung löchrig; auch dunkle Teile des Vordergrundobjektes gingen verloren. Es ließ sich aber feststellen, dass für höhere *Thresholds* die Maske trotzdem immer noch deutlich erkennbar blieb.

Die Masken zur Bildsequenz mit dunkelgrünem Hintergrund wiesen für kleine *Thresholds* nur leichte Störungen auf (Abbildung 2.32). Selbst die Bereiche, in denen die Beleuchtung des Hintergrundes deutlich schwankte, wurden korrekt als *unchanged* eingestuft. Für höhere Schwellwerte ergaben sich sehr scharf abgegrenzte Masken des Vordergrundobjektes, deren Qualität nicht wesentlich durch die höhere Störungsunterdrückung abnahm. Lediglich die Bereiche geringer Sättigung – also hell überstrahlte und sehr dunkle – gingen

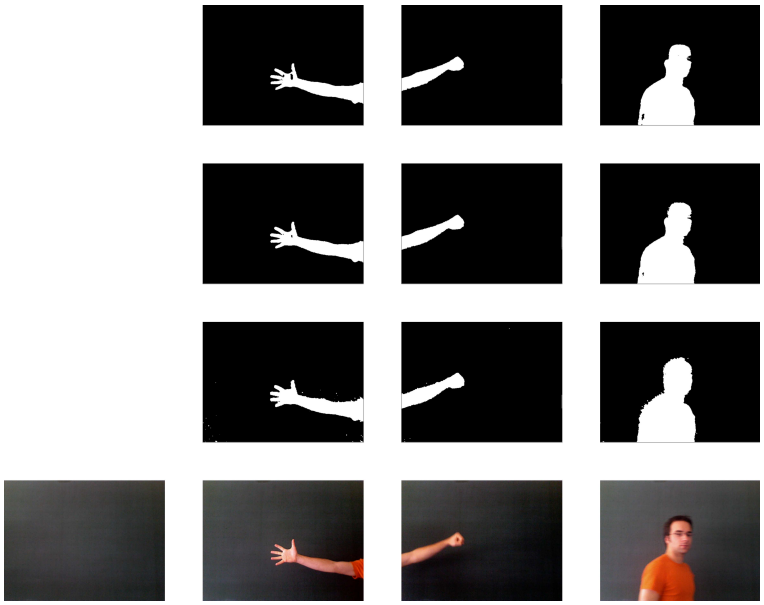


Abbildung 2.32: Ergebnis des veränderten blockbasierten Kollinearitätstests in der Chrominanzebene vor dunkelgrünem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_s
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 550, 1650 und 2750 sowie 80 für die positive Konstante C und einen Wert von 76 für $\sqrt{O_{dc}}$, die Kompensation für dunkle Bereiche

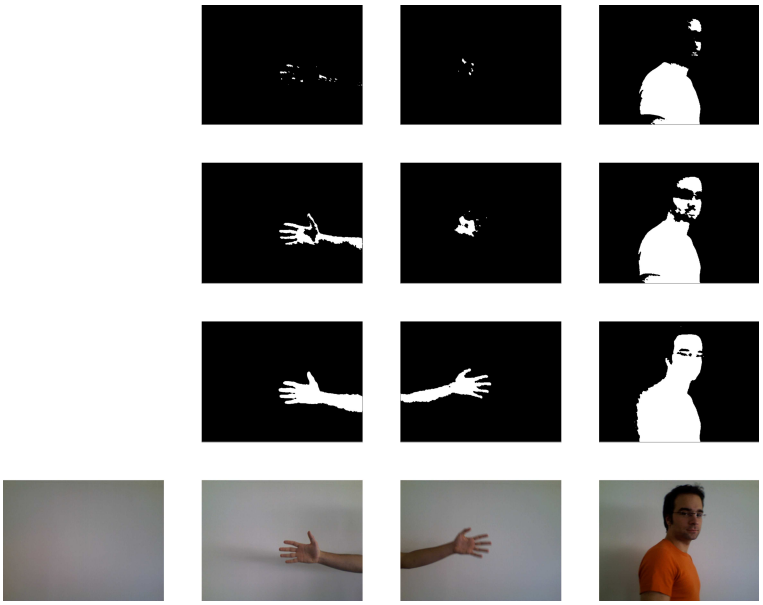


Abbildung 2.33: Ergebnis des veränderten blockbasierten Kollinearitätstests in der Chrominanzebene vor weißem Hintergrund zu drei verschiedenen Zeitpunkten und unterschiedlichen Werten für τ_s
links unten: gespeicherter Hintergrund
von unten nach oben: Originalbild, Masken für die drei *Thresholds* 550, 1650 und 2750 sowie 80 für die positive Konstante C und einen Wert von 76 für $\sqrt{O_{dc}}$, die Kompensation für dunkle Bereiche

verloren.

Auch aus der Bildsequenz vor weißem Hintergrund ergaben sich wiederum bei kleinen *Thresholds* keine nennenswert durch Beleuchtungsveränderungen und Schattenwurf gestörten Masken (Abbildung 2.33). Lediglich die Kontur der Masken war nicht glatt, konnte aber durch die Wahl höherer Schwellwerte verbessert werden. Dies ging allerdings wiederum einher mit Verlusten von Bereichen geringer Sättigung bis hin zu großen Teilen der Maske.

Lässt man die Störungen durch die Farbfehler der Sequenz vor schwarzem Hintergrund unberücksichtigt, ergab sich beim Vergleich der Masken untereinander ebenfalls eine ähnliche Unempfindlichkeit auf Beleuchtungsschwankungen. Für schwarzen und weißen Hintergrund war eine eine ähnliche Verschlechterung der Maske zu höheren Werten von τ_s hin zu beobachten. Dagegen wurden die Objekte vor dunkelgrünem Hintergrund deutlich besser erkannt.

Bei den Tests dieses Verfahrens – sowohl mit als auch ohne Veränderung – konnte eine deutliche Abhängigkeit der Maskenqualität von der positiven Konstante C erkannt werden: Hohe Werte für C lieferten im Allgemeinen glattere, kompaktere Masken. Allerdings verursachten zu hohe Werte zum einen ein deutliches Nachziehen der Maske, zum ande-

ren wurden kleine, lokal begrenzte Störungen als sehr große, teilweise sogar bildfüllende Vordergrundobjekte maskiert.

2.5 Diskussion

Ausgehend von den zuvor dargestellten Testergebnissen und den theoretischen Beschreibungen der einzelnen Verfahren wird nun deren Tauglichkeit für eine Trennung von Vorder- und Hintergrund in der in Abschnitt 1.4 beschriebenen Installation erörtert. Dazu muss bemerkt werden, dass für dise Beurteilung aller bisherigen Tests der durch die Projektion vorhandene veränderliche Hintergrund ausdrücklich nicht berücksichtigt wird. Dieses Problem lässt sich prinzipiell getrennt von den Trennungsverfahren betrachten. Ein diesbezüglicher Ansatz wird in Kapitel 3 untersucht. Am Ende dieser Arbeit wird dann die Eignung einer Kombination aus den beiden Teilverfahren erörtert. Dieses Vorgehen erscheint legitim, da in der aktuell von *rmh* verwendeten Installation der bewegte Hintergrund durch eine Beschränkung des aufgenommenen Spektrums auf den *IR*-Bereich unterdrückt wird und anschließend ein einfaches Differenz-Verfahren die Trennung in Vorder- und Hintergrund übernimmt.

Zunächst lässt sich bei Betrachtung der Ergebnisse feststellen, dass ein Trennungsverfahren, das auf der Farbtondifferenz basiert, mit den verwendeten einfachen Kameras gänzlich ungeeignet zur Erfassung von Interaktion ist: Die hierdurch erhaltenen Ergebnisse weisen lediglich auf ein sehr hohes Farbrauschen der verwendeten Kameras hin. Dies konnte für mehrere Kameratypen nachgewiesen werden, sodass ein dem Trennungsverfahren nachgelagerter Schritt zur Interaktionserkennung keine Ergebnisse liefern kann. Zwar würden wesentlich höherwertige Kameras dieses Problem sicher lösen, allerdings ist diese Investition unter wirtschaftlichen Gesichtspunkten für die beschriebene Anwendung untragbar. Dennoch muss für die Auswertung der weiteren Verfahren festgehalten werden, dass sich dieses Farbrauschen auch störend in den übrigen Ergebnissen ausgewirkt hat, wenn auch bei weitem nicht in dem Maße wie beim Test auf Farbtondifferenz. Daher erscheint es in jedem Falle sinnvoll, die verwendeten Kameras sorgfältig auszuwählen.

Beim Vergleich aller Ergebnismasken aus der Bildsequenz vor schwarzem Hintergrund zeigten sich Störungen durch Farbveränderungen gegenüber dem Hintergrundbild. Diese sind im jeweils zweiten gezeigten Originalbild der Sequenz zu erkennen, im ersten Bild sogar sehr deutlich sichtbar. Sie haben ihre Ursache in der automatischen Anpassung der Kamera auf die entsprechende Bildhelligkeit. Dabei erfolgt auch ein Weißabgleich, der bei schwarzem Hintergrund nicht erfolgreich ausgeführt werden kann. Da diese Störungen – zumindest für kleine *Thresholds* – in allen Masken des ersten Bildes der einzelnen Verfahren sehr ähnlich auftreten, werden sie für die weitere Diskussion vernachlässigt. Werden die für eine konkrete Installation verwendeten Kameras allerdings farbkalibriert, ist sichergestellt, dass diese Störungen nicht auftreten werden.

Zusammenfassend für alle erhaltenen Ergebnisse gibt Tabelle 2.3 einen Überblick über die Tauglichkeit der einzelnen Verfahren. Diese sind nach der Einschätzung ihrer Eignung von sehr gut bis ungenügend sortiert. Die einzelnen Kriterien wurden dabei aus den in Abschnitt 2.4 genannten Gründen rein qualitativ visuell beurteilt. In der Tabelle wurde darüber hinaus versucht, das Verhalten der einzelnen Verfahren gegenüber verschiedenen *Thresholds* zu vergleichen.

Tabelle 2.3: Zusammenfassung der Testergebnisse
Legende:
++ : sehr gut
+ : gut / ausgefüllt,
o : mittelmäßig / löchrig,
- : schlecht / große fehlende Teile
- - : ungenügend

Verfahren	**Eignung**	**niedrige τ**			**höhere τ**		
		Beleuchtungsunabhängigkeit	**Struktur d. Maske**	**Verwendbarkeit d. Maske**	**Beleuchtungsunabhängigkeit**	**Struktur d. Maske**	**Verwendbarkeit d. Maske**
Kollinearitätstest	++	o	+	o	+	+	+
Kollinearitätstest (veränd.)	++	+	+	+	+	o	o
Chrominanzabweichung	+	+	o	+	+	o	o
Chrominanzdifferenz	+	+	o	+	+	o	o
Luminanzdifferenz	o	-	+	-	o	o	o
Farbdifferenz	o	-	+	-	o	+	+
Abw. d. Luminanzvekt.	-	+	-	-	+	-	-
Abw. d. Luminanzvekt. (veränd.)	-	o	-	-	+	-	-
Farbtondifferenz	- -	-	-	-	-	-	-

Die in diesem Kapitel erhaltenen Ergebnisse deuten darauf hin, dass schon die untersuchten einfachen Differenzverfahren bei der Wahl geeigneter Schwellwerte brauchbare, beleuchtungsunabhängige Masken erzeugen. Allerdings ist diese Auswahl sehr schwierig zu treffen und von verschiedenen Randbedingungen abhängig, wie beispielsweise der gesamten zu erwartenden Helligkeit oder der zu erwartenden Helligkeit der Vordergrundobjekte: Es ergaben sich bei gleichen *Thresholds* deutliche Unterschiede der Masken bei verschiedenen Hintergründen und Helligkeiten. Es ist außerdem deutlich zu erkennen, dass Chrominanz-basierte Verfahren im Hinblick auf Beleuchtungsunabhängigkeit sicherere Erkennung ermöglichen als Verfahren, die die Helligkeit in ihre Entscheidung explizit oder – wie im Falle des Farbraumes RGB – implizit mit einbeziehen: Schon bei niedrigen *Thresholds* ergaben sich keine nennenswerten fehlerhaften Erkennungen in Bereichen mit schwankender Beleuchtung oder Schattenwurf. Dennoch liefert die einfache farbbasierte Differenzbildung die in der Verfahrensbeschreibung erwarteten besseren Ergebnisse als die Luminanzdifferenz. Für die einfachen Differenzverfahren lässt sich festhalten, dass im Allgemeinen durch Erhöhung der Schwellwerte eine gute Störungsunterdrückung erzielt werden konnte, die allerdings mit einem Verlust an Qualität der Masken einherging.

Eine sichere Erkennung von Vordergrundobjekten aufgrund der winkelbasierten Abweichung des Luminanzvektors innerhalb eines Blocks von Pixeln konnte nicht festgestellt werden: Die durch das Trennungsprinzip sehr feine Konturlinie der Vordergrundobjekte wurde teilweise durch Stöungen überlagert, die ohne Verlust der Maske nicht zu eliminieren waren. Abhängig von Helligkeit und Hintergrund konnte selbst diese Kontur teilweise nicht erzeugt werden. Auch die von Kamkar-Parsi *et al.* festgestellte Verbesserung der Trennungsergebnisse durch die Ergänzung um eine Beurteilung des Längenverhältnisses konnte – für die hier verwendeten einfachen Kameras – nicht bestätigt werden (Kamkar-Parsi *et al.*, 2005). Durch den nachgelagerten Verarbeitungsschritt bereits klassifizierter Hintergrundpixel, entstanden – zumindest für dunkle Hintergründe – massive Störungen im Bild, die teilweise durch Rauschen, teilweise durch Beleuchtungsänderungen und Schattenwurf verursacht wurden. Diese Störungen ließen sich durch höhere *Thresholds* nicht ohne den gleichzeitigen Verlust der Vordergrundmaske reduzieren. Die erwartete ausgefüllte Maske kontte lediglich vor schwarzem Hintergrund erzeugt werden, vor helleren Hintergründen ergab sich diese Verbesserung nicht. Einzig die in dieser Arbeit vorgesehene Prüfung auf eine Mindestlänge der Vektoren erbrachte eine Verbesserung dieses Verfahrens dahingehend, dass die zuvor aufgetretenen Störungen nicht mehr als Vordergrund erkannt wurden. Diese Bereiche waren dann aber mit diesem Verfahren nicht mehr bestimmbar, wodurch wiederum nachgelagerte Verarbeitungsschritte erforderlich sind.

Ebenfalls große unbestimmbare Bereiche entstanden bei der Auswertung des euklidischen Abstands in der UV-Ebene. Dieses Verfahren bestätigte aber die besondere Eignung Chrominanz-basierter Verfahren zur beleuchtungsunabhängigen Trennung von Vorder- und Hintergrund. Die Ergebnisse unterscheiden sich aber nicht nennenswert von denen der einfachen Chrominanzdifferenz, obwohl inhaltlich der euklidische Abstand aussagekräftiger in Bezug auf Bildveränderungen erscheint als die simple Differenz der jeweiligen Komponenten. Wie für das einfachere Chrominanz-basierte Verfahren auch nahm die Qualität der Masken für höhere Schwellwerte deutlich ab.

Das Verfahren, das aufgrund der Ergebnisse am geeignetsten zur beleuchtungsunabhängigen Trennung von Vorder- und Hintergrund vor verschiedenen Hintergründen erscheint, ist das von Mester *et al.* vorgeschlagene und von Griesser *et al.* erweiterte Verfahren des blockbasierten Kollinearitätstests mittels Bayes'scher Entscheidung und der Verwendung

von Markov-Zufalls-Feldern (Mester *et al.*, 2001; Griesser *et al.*, 2005). Hiermit lässt sich eine sehr gute Störungs-Unempfindlichkeit bei gleichzeitig hoher Maskenqualität im Hinblick auf die zu lösenden Probleme realisieren. Die aufgrund der vorherigen Ergebnisse der beiden Chrominanz-basierten Verfahren aufgestellte Vermutung, dass auch hier eine Entscheidung allein auf Grundlage der Chrominanzkomponenten eine Verbesserung mit sich bringt, konnte nicht bestätigt werden: Die Ergebnisse sind sehr ähnlich. Dies legt nahe, dass das von Mester *et al.* entwickelte rein Luminanz-basierte Verfahren ähnlich gute Ergebnisse liefern kann, da in den RGB-Farbkomponenten implizit auch die Luminanz enthalten ist.

Von besonders großer Wichtigkeit bei diesem Verfahren ist die geeignete Auswahl des Schwellwerts τ_s in Kombination mit der positiven Konstante C: Zu kleine τ_s machen das Verfahren störungsanfällig. Zu große Werte für C verursachen dann in einem sehr großen Bereich der Maske eine fehlerhafte Erkennung dieser Störungen als Vordergrund. Je größer C wird, desto größer ist auch der Einflussbereich der Störungen. Ein weiterer nachteiliger Effekt zu großer Werte für C ist das entstehende Nachziehen der Maske: Beim ersten Iterationsschritt dieses Verfahrens wird zur Berechnung des adaptiven *Thresholds* das Maskierungsergebnis des vorherigen Bildes verwendet. Ist C zu groß gewählt worden, ergeben sich so sehr große Bereiche, die als Vordergrund erkannt werden, obwohl sich das Vordergrundobjekt schon von dieser Stelle weg bewegt hat. Auf diese Weise kann ein sehr großer, störender Schweif der bewegten Vordergrundmaske entstehen. Daher sind bei einer späteren konkreten Installation ausgiebige Tests auf geeignete Kombinationen von τ_s und C unabdingbar.

In der Literatur – besonders im Bereich der Video-Überwachung – finden sich noch weitere Verfahren, die mehrfach als sehr vielversprechend bewertet werden. Dabei handelt es sich beispielsweise um statistische Verfahren, die einen möglichen Hintergrund oder mögliche Vorder- und Hintergründe mittels mehrerer Gauß-Verteilungen modellieren (beispielsweise: Stauffer und Grimson, 1999; Withagen *et al.*, 2003). Diese erscheinen aber aufgrund ihrer hohen Komplexität für die gegebene Anwendung als ungeeignet: Neben der Modellierung und Bildverarbeitung muss auf ein und demselben Computer zusätzlich noch die Spielelogik und Grafikerzeugung in Echtzeit ausgeführt werden können – eine Anforderung, die mit diesen Verfahren wohl nicht erfüllt werden kann. Sie werden allerdings ebenfalls gute Ergebnisse liefern für Anwendungen der Video-Überwachung mit typischerweise niedrigeren *Frame-Rates* oder Applikationen mit größerem Hardware-Budget.

Für das in dieser Arbeit favorisierte Verfahren des blockbasierten Kollinearitätstests liefern Griesser *et al.* eine sehr schnelle *GPU*-basierte Implementation (Griesser *et al.*, 2005). Dadurch ist es überaus geeignet, in der gegebenen Anwendung parallel zu den anderen Prozessen die Trennung von Vorder- und Hintergrund durchzuführen.

3 Stereoskopische Bildaufnahme

Im vorigen Kapitel wurde ein Verfahren zur Trennung von Vorder- und Hintergrund gefunden, dessen Beleuchtungsunabhängigkeit und Störungsunempfindlichkeit auf der Basis empirischer Ergebnisse nachgewiesen wurde. Dieses Verfahren ist bisher noch nicht in der Lage, die korrekte Maskierung eines Vordergrundobjektes vor einem sich bewegenden Hintergrund durchzuführen. Dieser entsteht bei der in Abschnitt 1.4 gegebenen Installation durch eine Projektion einer interaktiven Spielfläche, die sich mit dem Benutzer zusammen im Sichtbereich der Kamera befindet. In diesem Kapitel wird ein Verfahren hergeleitet, das in der Lage ist, diesen bewegten Hintergrund sicher zu unterdrücken und damit eine Trennung von Vorder- und Hintergrund nach dem favorisierten Verfahren zu ermöglichen.

Dazu wird zunächst die vorliegende Problemstellung genau definiert. Anschließend wird ein Verfahren, das als Ausgangspunkt der weiteren Entwicklung dient, vorgestellt und im Hinblick auf die Aufgabenstellung untersucht. Aus diesen Ergebnissen wird mit Blick auf die Problemstellung eine Veränderung des Verfahrens hergeleitet. Diese veränderte Methode wird mit der zu diesem Zweck entwickelten Testumgebung auf ihre Tauglichkeit untersucht. Am Ende dieses Kapitels werden die dabei erhaltenen Ergebnisse im Hinblick auf eine erfolgreiche Unterdrückung des Hintergrundes in der gegebenen Installation diskutiert.

3.1 Definition der Problemstellung

Wie erwähnt, wird in der nach Abschnitt 1.4 gegebenen Installation eine interaktive Spielfläche projiziert, die mit dem Benutzer zusammen im Kamerabild sichtbar ist. Wird nun das in Kapitel 2 vorgeschlagene Verfahren des blockbasierten Kollinearitätstests direkt zur Trennung von Vorder- und Hintergrund verwendet, werden auch projizierte Bewegungen innerhalb der Maske sichtbar. Dies kann so weit gehen, dass die gesamte Projektion als Vordergrundobjekt erkannt wird, wenn die leere Projektionsfläche als Hintergrund gespeichert worden ist. Auch die Verwendung zweier aufeinanderfolgender Kamerabilder kann nicht zu den gewünschten Ergebnissen führen: Zum einen würde – wie bereits erwähnt – ein stillstehender Benutzer nicht erkannt, zum anderen würde jede Bewegung – also auch die im projizierten Hintergrund – als Veränderung und damit als Interaktion erkannt.

Dieses Problem wurde in der ursprünglichen Installation durch zusätzliche Infrarotbeleuchtung sowie eine Begrenzung des von der Kamera aufgenommenen Spektrums auf den nahen *IR*-Bereich gelöst: Die Projektion war im Kamerabild nicht mehr sichtbar.

Davis und Bobick entwickelten ein sehr ähnliches Verfahren, bei dem allerdings vertikale Projektionsflächen verwendet wurden (Davis und Bobick, 1998).

Die ausschließliche Verwendung des Infrarotlichts führte bei der von *rmh* entwickelten Installation zu großer Fehleranfälligkeit bei Beleuchtungsschwankungen – insbesondere durch Sonneneinstrahlung. Daher soll ein Verfahren entwickelt werden, das die beleuchtungsunabhängige Trennung von Vorder- und Hintergrund bei sichtbarer Projektion bewegter Inhalte vornimmt. Da in Kapitel 2 bereits ein allgemeines, beleuchtungsunabhängiges Trennungsverfahren vorgeschlagen wurde, folgt hier ausschließlich die Untersuchung der Unterdrückung des bewegten Hintergrundes. Das dazu herangezogene Verfahren beinhaltet einen Trennungsschritt durch einfache Differenzbildung. Dies ist zur Beurteilung der Tauglichkeit in diesem Kapitel völlig ausreichend. Zusätzlich gibt die enthaltene Differenzbildung Aufschluss über die Qualitätsanforderungen an die verwendeten Kameras.

3.1.1 Erweiterte Notation

Für eine geeignete Beschreibung im weiteren Verlauf der Arbeit muss die Notation aus Abschnitt 2.1.1 erweitert werden:

- $I_{L,t}$ und $I_{R,t}$ seien Einzelbilder der linken und rechten Kamera.
- $I_{L,t}(X_L)$ liefert die Farbe des Pixels X_L im Bild der linken Kamera und $I_{R,t}(X_R)$ die Farbe eines Pixels X_R im Bild der rechten Kamera.

3.2 Das Verfahren

Die beschriebene Problematik legt nahe, mittels stereoskopischer Bildaufnahme eine tiefenbasierte Trennung vorzunehmen. Der bewegte Hintergrund, der sich prinzipbedingt in einer Ebene befindet, könnte so sicher unterdrückt werden. Ein diesbezüglich veränderter Aufbau der Installation ist Abbildung 3.1 zu entnehmen.

Ansätze zur Erstellung von Tiefenbildern lassen sich beispielsweise in der *„CMU-Video-Rate Stereo Machine“* (Kanade *et al.*, 1996), dem *„Small Vision Module“* (Konolige, 1997) oder einem von Darrell *et al.* entwickelten Algorithmus zum *Tracking* von Personen unter anderem mittels stereoskopischer Bildaufnahme finden (Darrell *et al.*, 2000). Diese wurden aufgrund ihrer Komplexität auf – wenn auch teilweise einfacher – eigens entwickelter Hardware implementiert. Auf Standard-Hardware dagegen läuft ein Verfahren von Bahadori *et al.*, das allerdings nur fünf bis zehn Bilder pro Sekunde verarbeiten kann und ebenfalls auf dem Algorithmus des *„Small Vision Modules“* basiert (Bahadori *et al.*, 2005). Alle diese Algorithmen sind bei der zur Verfügung stehenden Hardware und Rechenzeit ungeeignet: Die Spielelogik und Grafikerzeugung können nicht flüssig parallel ausgeführt werden.

Einen dagegen vielversprechenden Ansatz – ursprünglich von Ivanov *et al.* entwickelt und später von Wren und Ivanov erweitert – bietet der wesentlich simplere *Fast Depth Segmentation* (*FDS*)-Algorithmus, der im Folgenden untersucht und verändert wird (Ivanov *et al.*, 2000; Wren und Ivanov, 2001).

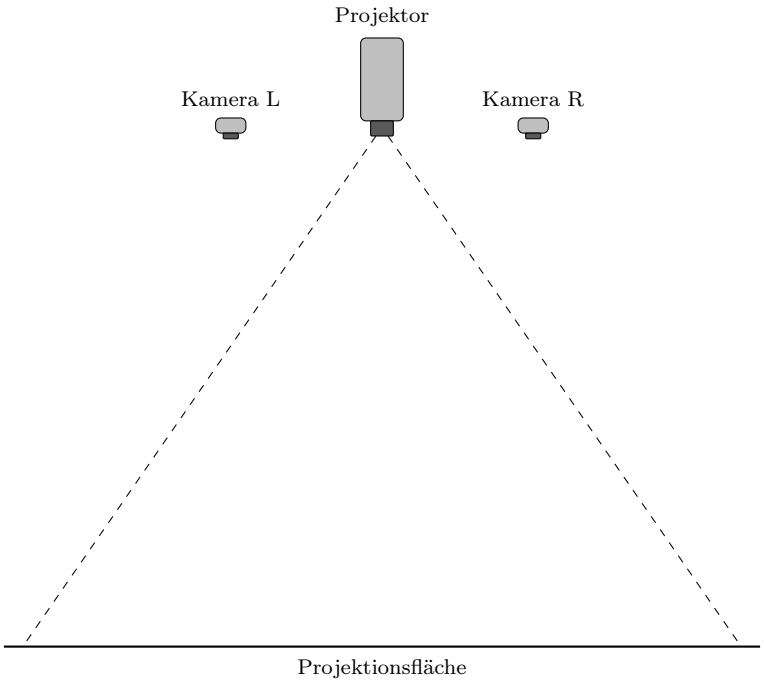


Abbildung 3.1: Aufbau der Installation mit stereoskopischer Bildaufnahme

3.2.1 Der FDS-Algorithmus

Für den in Abschnitt 1.4 beschriebenen Aufbau der Installation lässt sich – im Gegensatz beispielsweise zum *„Eye Toy“* (Sony, 2003) – feststellen, dass der gesamte Hintergrund in einer einzigen Ebene liegt. Diese geometrische Beschränkung lässt sich für eine tiefenbasierte Trennung ausnutzen (Wren und Ivanov, 2001).

Typischerweise misst ein Algorithmus zur Erstellung von Tiefenbildern die Deviationen zusammengehörender Bildelemente im linken und rechten Kamerabild:

$$\overrightarrow{d}\,(X_L) = X_R - X_L \tag{3.1}$$

Zusammen mit Kalibrierungsdaten der Kameras kann dann der Abstand des zugehörigen Objektes von der Kamera berechnet werden. Beim *FDS*-Algorithmus werden stattdessen in einem Initialisierungsschritt die Deviationen für alle Punkte der Hintergrund-Ebene bestimmt, die sich prinzipbedingt in einem festen Abstand zur Kamera befindet. Diese Deviationen werden in einer *Disparity-Map* angeordnet, aus der für jedes Pixel des linken Bildes die zugehörige Deviation abgelesen werden kann. Mithilfe dieser *Disparity-Map* lässt sich dann zu jedem Pixel im linken Bild das zugehörige Pixel des rechten Bildes berechnen, auf das derselbe Punkt der Projektionsfläche agebildet wird. Unter der Annahme, dass einerseits die Reflexionsparameter des sichtbaren Hintergrundes annähernd die eines Lambert'schen Materials sind und dass andererseits beide Pixel tatsächlich die

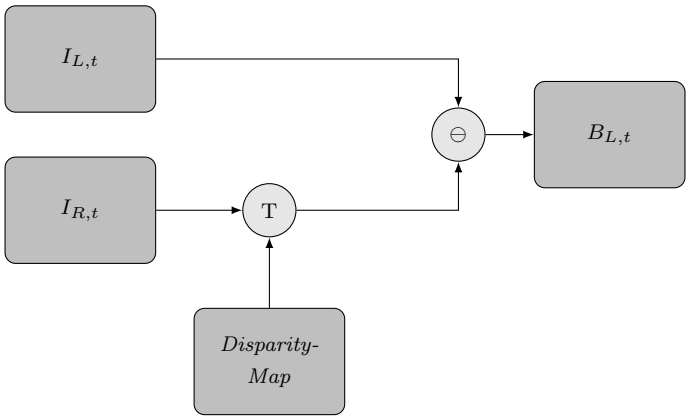


Abbildung 3.2: *FDS*-Algorithmus
T: Operator zur Transformation des Bildes mittels *Disparity-Map*
(verändert nach: Ivanov *et al.*, 2000)

zuvor vermessene Ebene zeigen, ergibt die Differenz der zugehörigen Farbwerte Null – unabhängig von der momentanen Beleuchtung. Zeigt ein oder zeigen beide Pixel ein Objekt, das sich nicht in der Projektionsebene befindet, ist die geometrische Bedingung, unter der die zugehörige Deviation berechnet wurde, verletzt. Somit zeigen beide Pixel nicht mehr denselben Objektpunkt: Die Differenz der zugehörigen Farbwerte ist von Null verschieden. Daraus folgt als Trennungsprinzip:

$$D_L\left(X_L\right) = I_{R,t}\left(X_L + \overrightarrow{d}\left(X_L\right)\right) \ominus I_{L,t}\left(X_L\right) \tag{3.2}$$

$$\mid D_L\left(X_L\right) \mid \underset{u}{\overset{c}{\geqq}} 0 \tag{3.3}$$

Dabei gibt $\ominus$ die einfache komponentenweise Differenz an; stattdessen kann aber auch jedes andere – und damit auch das in Kapitel 2 favorisierte – Trennungsverfahren benutzt werden. Einen Überblick über das Prinzip dieses Algorithmus gibt Abbildung 3.2.

Während Ivanov *et al.* zur Berechnung der Deviationen für alle Pixel mehrere unregelmäßig verteilte *Samples* triangulierten und anschließend interpolierten (Ivanov *et al.*, 2000), verwendeten Wren und Ivanov in ihrem Ansatz ein Schachbrettmuster zur Messung zusammengehöriger Punkte, das sie auf die Projektionsfläche legten. Die Punkte wurden mit dem von *OpenCV* bereitgestellten Algorithmus zur Erkennung von Schachbrettmustern erfasst und ein zweidimensionales Polynom nach Gleichung 3.4 an die Messwerte angepasst (Wren und Ivanov, 2001):

$$\overrightarrow{d}\left(X_L\right) = \overrightarrow{x}\left(X_L\right) \cdot \mathfrak{E} \tag{3.4}$$

Dabei gilt für $\overrightarrow{x}\left(X\right)$:

$$\overrightarrow{x}\left(X\right) = \left[\, x^2\,,\;\; y^2\,,\;\; xy\,,\;\; x\,,\;\; y\,,\;\; 1\,\right] \tag{3.5}$$

Zusätzlich wird aus m gemessenen Pixeln des linken Bildes zunächst die Matrix $\mathfrak{X}$ mit sechs Spalten und m Zeilen aufgestellt:

$$\mathfrak{X} = \begin{bmatrix} \overrightarrow{x}\,(X_{L,1}) \\ \overrightarrow{x}\,(X_{L,2}) \\ \cdots \\ \overrightarrow{x}\,(X_{L,m}) \end{bmatrix} \tag{3.6}$$

Außerdem wird die Matrix $\mathfrak{D}$ der zugehörigen Deviationen ins rechte Bild mit zwei Spalten und ebenfalls m Zeilen benötigt:

$$\mathfrak{D} = \begin{bmatrix} \overrightarrow{d}\,(X_{L,1}) \\ \overrightarrow{d}\,(X_{L,2}) \\ \cdots \\ \overrightarrow{d}\,(X_{L,m}) \end{bmatrix} \tag{3.7}$$

Damit ergibt sich die Matrix $\mathfrak{E}$ mittels der Methode der kleinsten Quadrate als:

$$\mathfrak{E} = \left(\mathfrak{X}^T\mathfrak{X}\right)^{-1}\mathfrak{X}^T\mathfrak{D} \tag{3.8}$$

Die Deviationen für sämtliche Pixel werden dann durch mehrfache Anwendung von Gleichung 3.4 ermittelt (Wren und Ivanov, 2001).

Dieser Algorithmus wurde von Wren und Ivanov noch um die Applikation *„Touch It“* zur Erkennung von Berührungen ergänzt: Dazu wurde eine zweite, virtuelle Ebene im Abstand von ungefähr 4 mm über der Projektionsfläche vermessen und der *FDS*-Algorithmus pro Bildpaar zweimal angewendet. Ergibt sich bei Verwendung der *Disparity-Map* für die Projektionsebene eine Klassifizierung als *changed* und gleichzeitig bei Verwendung der *Disparity-Map* für die zweite, virtuelle Ebene eine Klassifizierung als *unchanged*, so liegt am aktuellen Pixel eine Berührung vor (Wren und Ivanov, 2001).

3.2.2 Der neue Algorithmus

In Kapitel 2 wurde bereits ein geeignetes Trennungsverfahren hergeleitet. Daher wird in diesem Unterabschnitt lediglich ein geeigneter Vorverarbeitungsschritt für ein Stereo-Bildpaar entwickelt. Dieses Verfahren entsteht auf der Basis des soeben vorgestellten *FDS*-Algorithmus, beinhaltet aber selbst keinen eigenen Trennungsschritt und ist somit auf ein nachgelagertes Verfahren zur Maskierung angewiesen. Dieses Trennungsverfahren wird dazu in der folgenden Beschreibung und den Abbildungen weiterhin als $\ominus$ gekennzeichnet und der neue Algorithmus in den Abbildungen zusätzlich vom Trennungsschritt abgegrenzt.

Beim *FDS*-Algorithmus nach Wren und Ivanov wird das Bild einer Hilfskamera so in das Koordinatensystem der Hauptkamera transformiert, dass identische Pixelpositionen in beiden Bildern ein und denselben Punkt in der Projektionsebene zeigen (Wren und Ivanov, 2001). Diese Transformation ist aber für die in der Aufgabenstellung genannte Installation nicht sinnvoll: Auf diese Weise können zwar Vordergrundobjekte erkannt

werden, jedoch stehen deren Pixelkoordinaten im Bild der Hauptkamera in keinem bisher definierten Zusammenhang mit den Positionen der interaktiven Objekte in der Projektion. Für eine Interaktionserkennung ist dieser Zusammenhang aber zwingend erforderlich. Ein einfacher Weg, dies zu erreichen, ist die Transformation beider Kamerabilder in das Koordinatensystem der Projektion. Somit können die Pixelkoordinaten der erkannten Objekte direkt mit denen der interaktiven Objekte zur Interaktionserkennung verglichen werden. Damit ergibt sich aus Gleichung 3.2 das neue Trennungsprinzip als:

$$D(X) = I_{R,t}\left(X + \overrightarrow{d}_R(X)\right) \ominus I_{L,t}\left(X + \overrightarrow{d}_L(X)\right) \tag{3.9}$$

$$\mid D(X) \mid \overset{c}{\underset{u}{\geqq}} 0 \tag{3.10}$$

Dabei gilt:

$$\overrightarrow{d}_L(X) = X_L - X \tag{3.11}$$

$$\overrightarrow{d}_R(X) = X_R - X \tag{3.12}$$

Diese Deviationen werden in je einer *Disparity-Map* für das linke und rechte Kamerabild abgelegt.

Um die Deviationen aus den Gleichungen 3.11f. berechnen zu können, genügt es nicht mehr, einfach nur ein Schachbrettmuster auf die Projektionsfläche zu legen. Dieses muss sich vielmehr an einer genau definierten Position im projizierten Bild befinden. Ansonsten wäre X unbekannt und damit die Deviationen nicht bestimmbar. Zur Lösung dieses Problems wird vorgeschlagen, das Schachbrettmuster zu projizieren. Nur so lässt sich sicherstellen, dass X pixelgenau bekannt ist. Außerdem ist somit die Anforderung der automatischen Kalibrierung aus Abschnitt 1.4 erfüllt.

Ein weiterer Vorteil des projizierten Schachbrettmusters liegt darin, dass sich daraus ein Koordinatensystem erzeugen lässt, in dem sich intrinsische und extrinsische Kameraparameter bestimmen lassen, ohne die Abmessungen der konkreten Installation zu kennen. Zur Bestimmung dieser Parameter stellt *OpenCV* eine Funktion zur Kamerakalibrierung zur Verfügung. Das Koordinatensystem, für das die Installation automatisch vermessen werden kann, ergibt sich aus der zuvor genannten geometrischen Beschränkung für die Projektionsebene: Diese liege so in der x-y-Ebene, dass für alle Punkte in ihr die z-Koordinate als Null angenommen wird. Weiterhin befinde sich der Ursprung des Koordinatensystems an der Stelle der Projektionsebene, an die das Pixel $X = (0, 0)$ projiziert wird. Darüber hinaus fallen die x- und die y-Achse dieses Koordinatensystemes mit denen der Projektion zusammen. Damit ist sichergestellt, dass jedes Pixel $X = (x, y)$ an die Stelle $[X|0] = (x, y, 0)$ projiziert wird.

Für mehrere projizierte Schachbrettmuster lassen sich somit Pixel-Triplets mit den Elementen X, X_L und X_R erzeugen, für die sichergestellt ist, dass sie denselben Punkt der Projektionsebene zeigen. Ausreichend viele dieser Triplets liefern mithilfe des in *OpenCV* implementierten Algorithmus die intrinsischen und extrinsischen Kameraparameter $\mathfrak{A}$ und $\left[\mathfrak{R}|\overrightarrow{t}\right]$ (Intel, 2006). Zusätzlich lassen sich noch radiale (k_1 und k_2) und tangentiale Linsenverzerrungen (p_1 und p_2) berechnen. Die extrinsischen Parameter bestehen dabei aus

der Rotationsmatrix und dem Translationsvektor für die Kamera. Die Matrix der intrinsischen Parameter setzt sich wie folgt zusammen:

$$\mathfrak{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Dabei stellen f_x und f_y die Brennweite, c_x und c_y das Projektionszentrum in Pixeln dar.

Mit diesen Daten kann *OpenCV* die Pixelkoordinaten der Projektionsebene in die zugehörigen Pixelkoordinaten der Kamera umrechnen. Dies erfolgt nach folgenden Gleichungen:

$$\overrightarrow{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathfrak{R} \cdot [X|0]^T + \overrightarrow{t} \tag{3.14}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{cases} \frac{1}{z} \cdot \overrightarrow{p} & \text{wenn } z \neq 0 \\ \overrightarrow{p} & \text{wenn } z = 0 \end{cases} \tag{3.15}$$

$$\begin{aligned} x'' &= x' \cdot \left(k_1 r^2 + k_2 r^4 + 1\right) + 2p_1 x' y' + p_2 \left(r^2 + 2x^2\right) \\ y'' &= y' \cdot \left(k_1 r^2 + k_2 r^4 + 1\right) + 2p_2 x' y' + p_1 \left(r^2 + 2y^2\right) \end{aligned} \tag{3.16}$$

$$X_L = [f_x x'' + c_x \ , \ \ f_y y'' + c_y] \tag{3.17}$$

Dabei wurden zur Berechnung des Pixels im linken Bild die Parameter der linken Kamera verwendet. Entsprechendes gilt für das rechte Kamerabild.

Der zur Kalibrierung in *OpenCV* implementierte Algorithmus entspricht dabei im Wesentlichen der *„Camera Calibration Toolbox for Matlab“* von Bouguet (Bouguet, 2007). Letztere wiederum geht zurück auf die Kalibrierungsalgorithmen von Heikkilä und Silvén sowie Zhang (Heikkilä und Silvén, 1997; Zhang, 2000).

Anstelle der Schätzung der Deviationen kann das hier vorgeschlagene Verfahren nun die jeweils zugehörigen Kamerapixel direkt berechnen, stützt sich dabei auf wesentlich mehr Messpunkte als der *FDS*-Algorithmus und berücksichtigt zusätzlich vorhandene Linsenverzerrungen. Damit nicht für jedes Einzelbild eine Transformation berechnet werden muss, wird während des Initialisierungsschrittes eine *Look-Up-Table* (LUT) erstellt, die zu jedem Pixel der Projektion die zugehörige Speicherstelle in den Kamerabildern enthält: Es wird also keine *Disparity Map*, sondern eine *Pixel-LUT* verwendet. Das fertige neue Verfahren wird in Abbildung 3.3 dargestellt.

3.3 Entwicklung einer geeigneten Testumgebung

Um den hier vorgeschlagenen Algorithmus zur Vorverarbeitung eines Stereo-Bildpaares auf seine Tauglichkeit zu untersuchen, wird zunächst eine idealisierte Testumgebung geschaffen. Diese wird erweitert, um reale Kameras verwenden zu können. Im idealisierten Teil

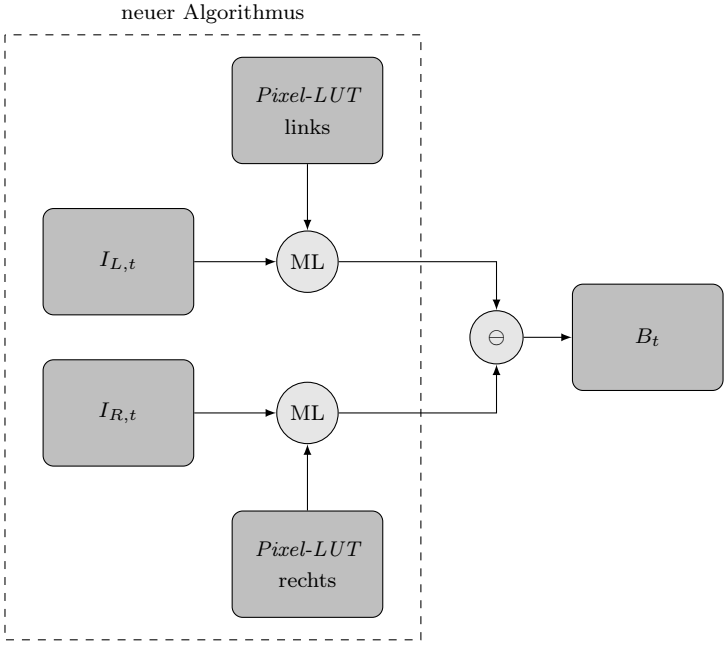


Abbildung 3.3: Der neue Algorithmus zur Vorverarbeitung der Stereo-Bildpaare
ML: Operator zum Auslesen der durch die *Pixel-LUT* gegebenen Speicherstellen

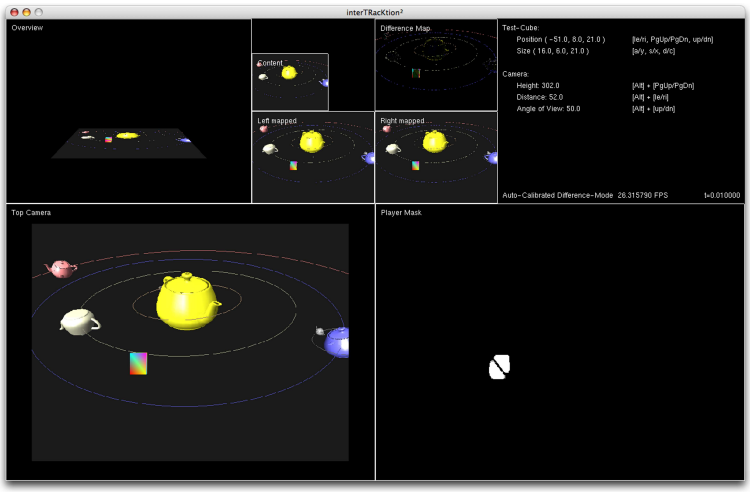


Abbildung 3.4: Screenshot der Software zum Test des neuen Algorithmus zur Vorverarbeitung der Stereo-Bildpaare

der Umgebung werden zunächst die Implementation sowie mögliche Probleme untersucht. Dazu wird auch hier ein bewegter Hintergrund erzeugt, der bei korrekter Transformation im Trennungsschritt komplett unterdrückt werden soll. Anschließend werden mit realen Kamerabildern die Auswirkungen der festgestellten Probleme sowie die Tauglichkeit für eine konkrete Installation untersucht.

Ein Beispiel der Anwendung dieser Software zeigt Abbildung 3.4.

3.3.1 Verwendetes Computersystem

Das für die Entwicklung zur Verfügung stehende Computersystem bestand aus einem *MacBook Pro* mit:

- 2,33 GHz Intel Core 2 Duo-Prozessor
- 2 GB 667 MHz DDR2 SDRAM
- ATI Radeon X1600 (PCIe-Bus)
- Mac OS X 10.4
- 2 Unibrain Fire-i Board Cameras

3.3.2 Wahl der Programmiersprache und der benötigten Bibliotheken

Das stereoskopische Kamerasystem soll möglichst automatisch kalibriert werden können. Da *OpenCV* die dafür benötigten Funktionen zur Verfügung stellt, wurde diese Bibliothek

erneut verwendet. Auch für diese Software fiel die Wahl der Programmiersprache aus den in Unterabschnitt 2.3.2 genannten Gründen auf *C++*.

Die beiden Kameras müssen – soweit möglich – synchron Bilder aufnehmen. *OpenCV* bietet allerdings nur begrenzte Möglichkeiten der Kamerasteuerung, weshalb dazu eine eigene Bibliothek verwendet wurde. Bei den zur Verfügung stehenden *Fire-i Board Cameras* der Firma *Unibrain* handelt es sich um *IIDC*-konforme *IEEE 1394*-Digitalkameras. Eine freie Bibliothek, die diesen Standard unterstützt, ist *libdc1394* in der Version 2.0.0-RC5 (Douxchamps, 2007). Zwar handelt es sich dabei nicht um ein *Final Release*, vom Autor der Bibliothek wird aber empfohlen, für Neuentwicklungen die aktuelle anstelle der alten Version 1.2.0 zu verwenden. Weiterhin ist nur die aktuelle Version auf dem zur Entwicklung zur Verfügung stehenden Betriebssystem *Mac OS X* lauffähig. Für *Windows* hingegen existiert noch keine Version. Dieser Umstand ist allerdings im Hinblick auf die konkreten Installationen unbedeutend, da für sie bereits Kamera-*Interfaces* existieren. Auch bei der Wahl dieser Bibliothek war die Erfahrung des Autors ausschlaggebend.

Die zu Testzwecken gewünschte idealisierte Simulation legt die Verwendung von *OpenGL* nahe. Um die realen Tests in die dann schon existierende Umgebung zu integrieren, wurden auch die dazu nötigen Bildverarbeitungsschritte auf der Grafikkarte und nicht mithilfe von *OpenCV* implementiert. Lediglich zur Kalibrierung wurde letztere Bibliothek verwendet. Zur Verarbeitung der Bilder wurde ein System aus *Framebuffer-Objects* (*FBOs*) und *Shadern* entwickelt, die in *OpenGL Shading Language* (*GLSL*) programmiert wurden.

Da *OpenGL* kein Anzeigesystem enthält, wurde zu diesem Zweck die freie Bibliothek *SDL - Simple Directmedia Layer* in der Version 1.2.11 verwendet (Lantinga, 2007). Mit ihr konnte sowohl das benötigte Fenster zum Anzeigen der Einzelbilder geöffnet als auch ein geeignetes *Event*-System geschaffen werden, um die Testumgebung zu bedienen. Weiterhin ist diese Bibliothek in *C* geschrieben und bietet eine direkte Anbindung an *C++*. Auch hier war die Erfahrung des Autors ausschlaggebend.

3.3.3 Klassenstruktur, Zustands- und Flussdiagramm

Klassenstruktur

Auch für diese Testumgebung wurde eine Strukturierung des Projekts in einzelne funktionale Einheiten durch die objektorientierte Programmierung ermöglicht. Die für diese Software entwickelte Klassenstruktur geht aus Abbildung 3.5 hervor.

Die Klasse `TApplication` kontrolliert den gesamten Programmablauf nach Abbildung 3.7 (Seite 75), die Klasse `TEvents` verarbeitet die Benutzereingaben und ordnet diesen eine programminterne *Event*-Bezeichnung zu. Das Grafiksystem wird von der Klasse `TGraphics` verwaltet und die Kontrolle über die einzelnen Programmzustände nach Abbildung 3.6 (Seite 73) übernimmt die Klasse `TStatemachine`. Alle vier Klassen sind durch entsprechende Vererbung als *Singleton* implementiert. Dies stellt zum einen sicher, dass es von jeder dieser Klassen nur eine Instanz geben kann, zum anderen hat es den Vorteil, dass die Instanzen global verfügbar sind.

Von der Klasse `TGeometry` leiten sich vier Klassen für Geometrieobjekte ab, die in der Simulation verwendet werden. Von ihr wird ebenfalls die Klasse `TCamera` abgeleitet, die ihrerseits Oberklasse für `TCameraGLTX` ist. Aus dieser entstehen die virtuellen Kameras für die Simulation, die somit wie Geometrieobjekte transformiert werden können.

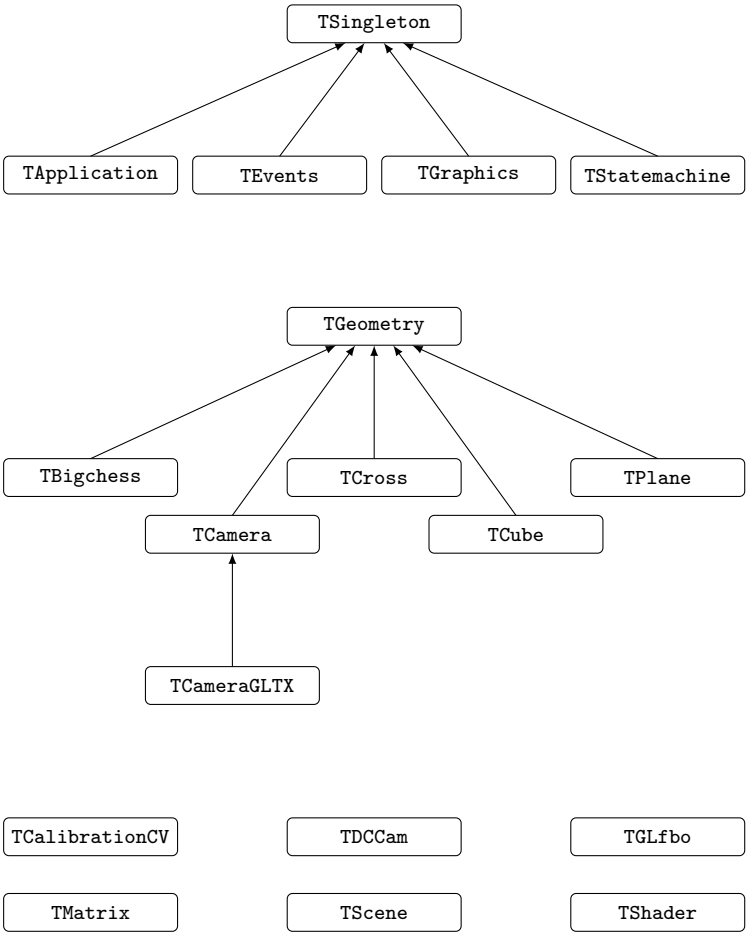


Abbildung 3.5: Klassenstruktur des Programms zum Testen des neuen Vorverarbeitungs-Algorithmus

Die Klasse `TCalibrationCV` beinhaltet den von *OpenCV* zur Verfügung gestellten Algorithmus zum Auffinden von Schachbrettmustern sowie die zur Kalibrierung der Kameras notwendigen Funktionen.

Objekte der Klasse `TGLfbo` bieten die Möglichkeit, mithilfe der in *OpenGL* enthaltenen *FBOs* außerhalb des sichtbaren Bildbereiches in eine Textur zu *rendern*. Diese kann für weitere *Renderings* in eine Textur oder auf den Bildschirm verwendet werden. Für Letzteres enthält diese Klasse eine eigene Funktion, mit der sich der Texturinhalt in beliebiger Größe darstellen lässt.

Objekte für reale Kameras, mit denen der neue Algorithmus getestet werden soll, entstehen aus der Klasse `TDCCam`. Sie stellt alle nötigen Funktionen zur Kamerabedienung zur Verfügung. Sie enthält außerdem die benötigte *Look-Up-Table*, um eine Transformation des Bildes nach dem neuen Algorithmus durchführen zu können. Das Bildergebnis wird in eine Textur eines Objektes der Klasse `TGLfbo` geschrieben. Diese muss dem Kameraobjekt aber explizit zugewiesen werden.

Die Klasse `TMatrix` enthält die für Transformationen nötigen Matrizen sowie die zugehörigen Rechenoperationen. Die Klasse `TScene` bietet die Möglichkeit, Geometrieobjekte in einer Liste abzulegen und dann mit einem einzigen Befehl zu *rendern*. Zuletzt werden mit Objekten der Klasse `TShader` *Vertex-* und zugehörige *Fragment-Shader* geladen und aktiviert. Darüber hinaus können *Shader-Codes* auch zur Laufzeit dynamisch erzeugt werden.

Zustandsdiagramm

Die entwickelte Software beinhaltet in der Klasse `TStatemachine` eine einfache *Finite-State-Machine*, die die verschiedenen Zustände nach Abbildung 3.6 verwaltet. Das Programm startet im Zustand „*Standard*“ und zeigt damit die Simulation. Durch Drücken der *Return*-Taste startet die automatische Kalibrierung: Im Zustand „*Auto-Chessboard*“ werden neun Schachbrettmuster aufgenommen und vermessen. Anschließend wird automatisch in den Zustand „*Auto-Calibration*“ gewechselt, in dem die zuvor erhaltenen Messdaten zur Kalibrierung der virtuellen Kameras verwendet werden. Danach wird dynamisch für die beiden virtuellen Kameras ein passender Transformations-*Shader* erzeugt. Ist dieser kompiliert und gelinkt, wechselt das System automatisch in den Zustand „*Auto-Difference*“ und zeigt u. a. eine Aufsicht auf die virtuelle Szene sowie die entstandene Maske. Die Rückkehr in den Zustand „*Standard*“ ist aus jedem anderen Zustand durch Drücken der *Escape*-Taste möglich.

Wird bei der Initialisierung festgestellt, dass zwei reale Kameras am System angeschlossen sind, so wird automatisch in den Zustand „*Real-Cam*“ gewechselt und die Bilder der realen Kameras angezeigt. Durch Drücken der Leertaste wird der Zustand „*Real-Calibration*“ gestartet, in dem der Anwender manuell neun vorbereitete Schachbrettmuster projizieren muss. Zur Vermessung eines jeden Musters muss jeweils wieder die Leertaste gedrückt werden. Nachdem das neunte Muster erkannt wurde, wechselt das System automatisch in den Zustand „*Real-Difference*“. Aus den letzten beiden Zuständen kann jederzeit durch Drücken der *Escape*-Taste in den Zustand „*Real-Cam*“ zurückgekehrt werden; aus diesem Zustand gelangt man ebenso in den Zustand „*Standard*“. Durch Drücken der *R*-Taste gelangt man dann wieder in den Zustand „*Real-Cam*“.

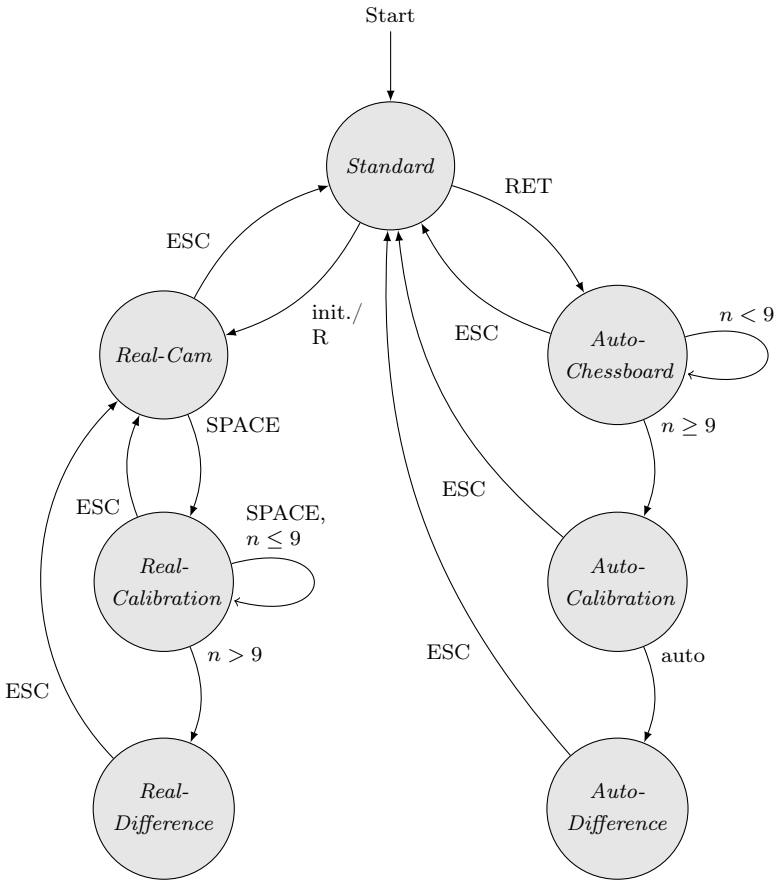


Abbildung 3.6: Zustandsdiagramm der Umgebung zum Testen des neuen Vorverarbeitungs-Algorithmus

Flussdiagramm

Der Programmablauf der Testumgebung ist in Abbildung 3.7 stark vereinfacht wiedergegeben. Nach dem Programmstart werden zunächst das Grafik-, das *Event-* und das Status-Subsystem erstellt und initialisiert. Anschließend werden die benötigten *Shader* geladen und kompiliert sowie die *FBOs* mit den zugehörigen Texturen erzeugt. Sind reale Kameras an das System angeschlossen, so werden diese ebenfalls initialisiert und das System wechselt in den Zustand „*Real-Cam*“. Anschließend werden die für die Simulation benötigten Geometrieobjekte erstellt.

Der *Main-Loop* beginnt zunächst mit der Behandlung aufgelaufener *Events*, danach werden sämtliche Texturinhalte erzeugt. Befindet sich das System in einem Simulationszustand, wird hier zunächst der bewegte Hintergrund für die virtuelle Projektionsfläche gerendert. Danach werden – abhängig vom aktuellen Zustand – beispielsweise die Bilder der virtuellen oder der realen Kameras in die Texturen des linken und rechten *FBO* geschrieben. Aus Gründen der Verarbeitungsgeschwindigkeit erfolgt hierbei im Zustand „*Real-Cam*“ die Transformation mithilfe der zugehörigen *LUTs*.

Nachdem die Bilder eingelesen sind, erfolgt für den Zustand „*Auto-Difference*“ deren Transformation. Für die Zustände „*Real-*“ und „*Auto-Difference*“ erfolgt anschließend die Trennung in Vorder- und Hintergrund sowie weitere Bildverarbeitungsschritte. Befindet sich das System im Zustand „*Real-Calibration*“, „*Auto-Chessboard*“ oder „*Auto-Calibration*“ werden die in den Bildern enthaltenen Schachbrettmuster erkannt und vermessen sowie die Kameras kalibriert. Dabei werden die *LUTs* gefüllt oder die Transformations-*Shader* erzeugt.

Anschließend erfolgt abhängig vom aktuellen Zustand die Darstellung der Inhalte der einzelnen *FBOs* am Bildschirm. Wurde beim *Event-Handling* kein Abbruch gewünscht, so verbleibt das Programm im *Main-Loop*.

3.3.4 Aufbau und Anwendung der Pixel-LUTs

Für den neuen Algorithmus wurde in Abschnitt 3.2.2 vorgeschlagen, eine *Pixel-Look-Up-Table* zur Transformation der Kamerabilder in das Koordinatensystem der Projektion zu verwenden. Diese *LUT* hat dieselbe Breite und Höhe wie das gewünschte Zielbild und ist mit ihm deckungsgleich. Die Größe und der Aufbau sind dem Auszug aus der Klassendeklaration von `TDCCam` in Listing 3.1 zu entnehmen. In dieser Implementation wird eine Größe von 640 mal 480 Pixeln verwendet. Jedes Pixel der *LUT* hat ebenso viele Farbkanäle wie die Pixel des Zielbildes – hier sind es vier. Der Grund dafür: Für jeden Farbkanal eines Pixels des Zielbildes gibt das zugehörige Pixel der *LUT* die Speicherposition im Kamerabild an. Für RGB-Bilder scheint diese aufwendige Dimensionierung übertrieben, da die Unterteilung der Pixel in Farbkanäle für Quell- und Zielbild identisch ist. Für unterschiedliche Bildformate erweist es sich aber als äußerst praktisch: Während der Transformation können die Formate angepasst werden. Beispielsweise kann ein Quellbild im Format BGR somit kanalweise in ein Bild im Format RGB verwandelt werden – je nachdem, welche Anforderungen die Anwendung stellt. Weiterhin lassen sich so auch Bilder im Format YC_bC_r 4:1:1 besser weiterverarbeiten. Die Bibliothek *libdc1394* ordnet diese Bilder im Speicher gemäß Tabelle 3.1 an. Zur standardisierten Weiterverarbeitung ist aber eine Anordnung nach Tabelle 3.2 günstiger. Wichtig ist dabei allerdings, dass das Bild

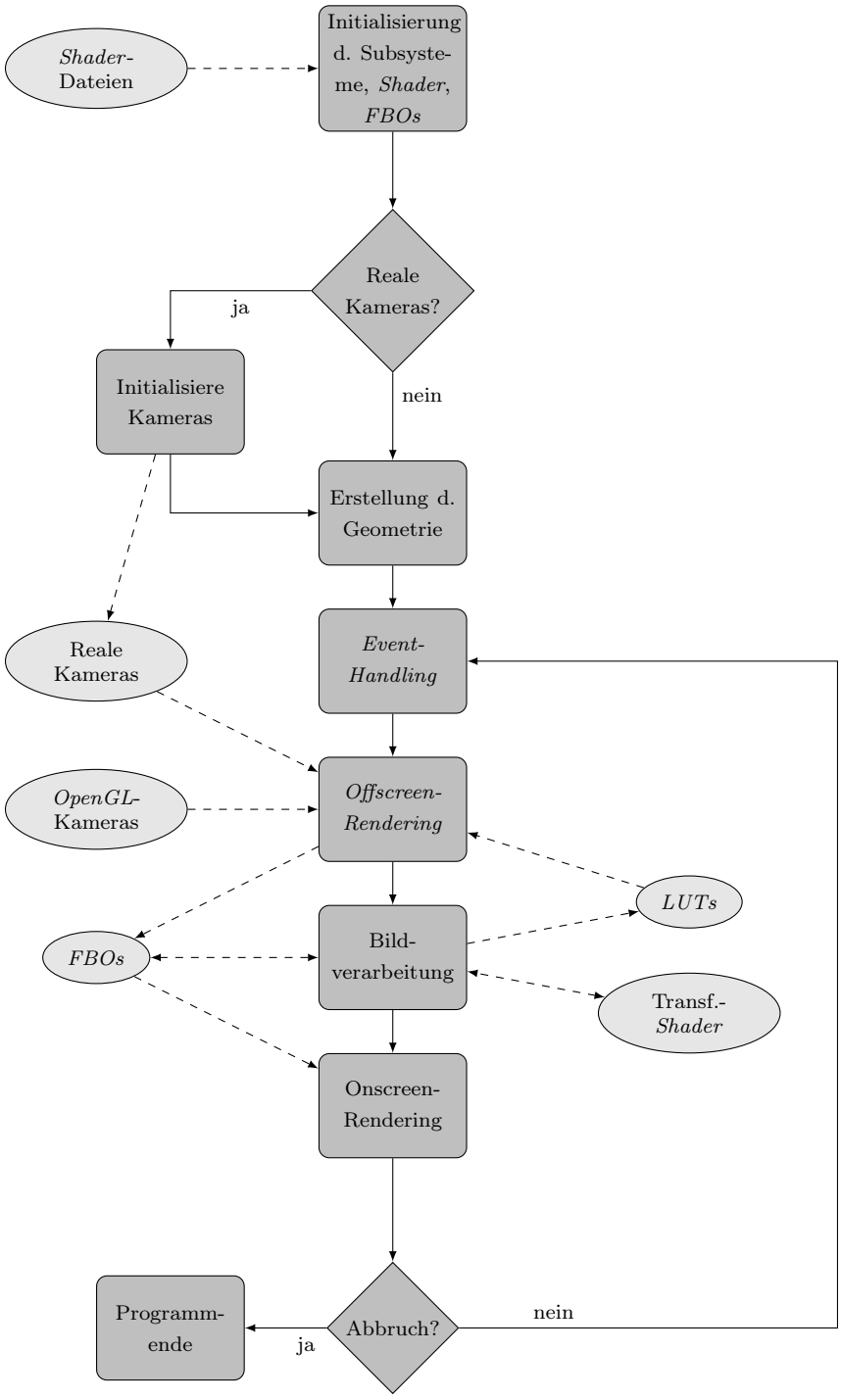


Abbildung 3.7: Flussdiagramm der Umgebung zum Testen des neuen Vorverarbeitungs-Algorithmus

Listing 3.1: Auszug aus `tdccam.h` – Aufbau und Größe einer Pixel-*LUT*

```
class TDCCam
{
public:
  // constructor
  TDCCam () ... {
    fpLUT = new unsigned int[640*480*4];
    ...
  }
  ...
 private:
   unsigned int* fpLUT;
   ...
}
```

Tabelle 3.1: Speicherpositionen innerhalb eines Bildes im Format YC_bC_r 4:1:1, das von *libdc1394* zur Verfügung gestellt wird

Byte-Offset	Pixel	Farbkanal
0x0000	0, 1, 2, 3	C_b
0x0001	0	Y
0x0002	1	Y
0x0003	0, 1, 2, 3	C_r
0x0004	2	Y
0x0005	3	Y
0x0006	4, 5, 6, 7	C_b
0x0007	4	Y
0x0008	5	Y
0x0009	4, 5, 6, 7	C_r
0x000A	6	Y
0x000B	7	Y

Tabelle 3.2: Zur Weiterverarbeitung günstigere Speicherpositionen innerhalb eines Bildes im Format YC_bC_r 4:1:1

Byte-Offset	Pixel	Farbkanal
0x0000	0	C_r
0x0001	0	Y
0x0002	0	C_b
0x0003	1	C_r
0x0004	1	Y
0x0005	1	C_b
0x0006	2	C_r
0x0007	2	Y
0x0008	2	C_b
0x0009	3	C_r
0x000A	3	Y
0x000B	3	C_b

Listing 3.2: Pseudocode der Transformation des Kamerabildes in das Koordinatensystem der Projektion

```
for( int n = 0; n < width * height * channels; n++ )
{
  targetImage[ n ] = camImage[ fpLUT[ n ] ];
}
```

immer noch prinzipiell im Format YC_bC_r 4:1:1 vorliegt und die Chrominanzen lediglich den entsprechenden Pixeln zugeordnet wurden. Bei der Implementation wurde zusätzlich ein vierter Farbkanal für die *LUT* gewählt, da das Zielbild – eine *OpenGL*-Textur – im Format RGBA vorliegt.

Abhängig vom gewählten Format von Kamera- und Zielbild werden die *LUTs* zu Beginn des Programms so initialisiert, dass vom Kamerabild ins Zielbild keine Transformation stattfindet. Nachdem die Kalibrierungsdaten erhalten wurden, wird mithilfe eines Objektes der Klasse `TCalibrationCV` für jedes Pixel des Zielbildes das zugehörige Pixel des Kamerabildes errechnet und es werden die Speicherpositionen der Farbkanäle in die *LUT* eingetragen. Die Transformation ist in Listing 3.2 in Pseudocode dargestellt. Hier lässt sich auch erkennen, dass dieser Transformationsschritt simpel und unabhängig vom Bildformat ist, wenn der Aufbau der *LUT* günstig gewählt wird.

3.3.5 Offscreen-Rendering in ein FBO

Eine der Aufgaben, die in der entwickelten Software immer wieder durchgeführt werden muss, um die Maske zu erstellen, ist das *Rendering* außerhalb des Bildschirmes in ein

Listing 3.3: Funktion zum Aktivieren eines *FBO* als *Rendertarget*

```
void
TGLfbo::Activate
()
{
  glBindFramebufferEXT( GL_FRAMEBUFFER_EXT, fFBO );
  glPushAttrib( GL_VIEWPORT_BIT | GL_COLOR_BUFFER_BIT |
                GL_DEPTH_BUFFER_BIT );
  glViewport( 0, 0, fWidth, fHeight );
}
```

Framebuffer-Object. Die entwickelte Umgebung bietet dazu mit der Klasse `TGLfbo` eine Struktur, die dies sehr erleichtert. Die aus dieser Klasse erstellten Objekte enthalten ein *FBO*, an das eine Textur angekoppelt ist. Wird das *Framebuffer-Object* als *Rendertarget* aktiviert, kann in diese Textur anstatt auf den Bildschirm geschrieben werden. Die dafür bereitgestellte Funktion zeigt Listing 3.3. Diese Textur kann danach als Objekttextur beispielsweise für eine *Plane* verwendet werden. Auch zur Texturaktivierung stellt die Klasse `TGLfbo` eine geeignete Funktion zur Verfügung. Wird zum *Rendering* der *Plane* ein *Shader*-Programm benutzt, können darin die bildverarbeitenden Schritte auf der Grafikkarte berechnet werden. Das führt in der Regel zu einer Beschleunigung der Verarbeitung. Listing 3.4 zeigt beispielhaft die für die Differenzbildung notwendigen Schritte.

3.3.6 Difference-Shader

Um den neuen Algorithmus zur Vorverarbeitung eines Stereo-Bildpaares zu testen, wurde in die Software ein einfacher Trennungsschritt integriert, der auf komponentenweiser Differenzbildung im RGB-Farbraum basiert. Dieser Schritt wurde in einem *Shader* implementiert. Der zugehörige *Vertex-Shader* ist Listing 3.5 zu entnehmen, der zugehörige *Fragment-Shader* Listing 3.6. Diese Differenz wird mithilfe eines weiteren *Shaders* durch morphologische Operatoren verbessert und danach so ausgewertet, dass innerhalb eines Blocks von Pixeln die Summe der quadratischen Differenzen gebildet und diese zur Maskenbildung mit einem *Threshold* verglichen wird. Dieser Schritt kann später für eine konkrete Anwendung durch höherwertige Trennungsverfahren – wie das in Abschnitt 2.5 empfohlene – ersetzt werden, genügt aber für einen aussagekräftigen Test der Vorverarbeitung.

Auf weitere Code-Beispiele im Rahmen dieser Arbeit wird aufgrund der Komplexität der Testumgebung verzichtet. Die grundlegenden Schritte, die die Vorverarbeitung charakterisieren, wurden jedoch ausführlich dargestellt.

Listing 3.4: Beispielhafte Implementation der Differenzbildung mithilfe eines *Shaders* durch *Offscreen-Rendering* in ein *FBO*

```
// activate render target
MyDestinationFBO.Activate();

// prepare orthographic view
glMatrixMode( GL_PROJECTION );
glPushMatrix();
glLoadIdentity();
gluOrtho2D( 0, 640, 0, 480 );
glMatrixMode( GL_MODELVIEW );
glPushMatrix();
glLoadIdentity();
glClearColor( 0.0f, 0.0f, 0.0f, 1.0f );
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

// activate shader
MyDifferenceShader.Activate();

// activate textures
MySourceFBO1.ActivateTexture( GL_TEXTURE0 );
MySourceFBO2.ActivateTexture( GL_TEXTURE1 );

// render quad
glBegin( GL_QUADS );
{
  glTexCoord2f( 0.0, 0.0 );          glVertex2i(  0, 0 );
  glTexCoord2f( 0.0, 0.46875 );      glVertex2i(  0, 480 );
  glTexCoord2f( 0.625, 0.46875 );    glVertex2i( 640, 480 );
  glTexCoord2f( 0.625, 0.0 );        glVertex2i( 640,   0 );\
}
glEnd();

// deactivate textures
MySourceFBO2.DeactivateTexture( GL_TEXTURE1 );
MySourceFBO1.DeactivateTexture( GL_TEXTURE0 );

// deactivate shader
MyDifferenceShader.Deactivate();

// restore previous view
glMatrixMode( GL_PROJECTION );
glPopMatrix();
glMatrixMode( GL_MODELVIEW );
glPopMatrix();

// deactivate FBO
MyDestinationFBO.Dectivate();
```

Listing 3.5: *Vertex-Shader* zur komponentenweisen Differenzbildung

```
void main
()
{
  gl_TexCoord[0] = gl_MultiTexCoord0;
  gl_Position = ftransform();
}
```

Listing 3.6: *Fragment-Shader* zur komponentenweisen Differenzbildung

```
uniform sampler2D texture0;
uniform sampler2D texture1;

void main
()
{
  vec4  colorDisplay =
          abs( texture2D(texture0,gl_TexCoord[0].st) -
               texture2D(texture1,gl_TexCoord[0].st)   );

  colorDisplay.a = 1.0;

  gl_FragColor = colorDisplay;
}
```

3.4 Untersuchung der Genauigkeit des neuen Verfahrens und des FDS-Algorithmus

Zunächst wurde die Genauigkeit der Transformation beider Algorithmen verglichen. Dazu wurden sowohl in der idealen Simulation als auch im realen Aufbau 273 Punkte mithilfe mehrerer Schachbrettmuster vermessen. Für diese bekannten Punkte wurden die quadratischen Fehler der Transformationen beider Verfahren für ideale sowie reale Bedingungen ermittelt:

$$\varepsilon^2 = (x' - x)^2 + (y' - y)^2 \tag{3.18}$$

Dabei geben x und y die Pixelkoordinaten in der Projektion, x' und y' die durch die Transformation errechneten Pixelkoordinaten im jeweiligen Kamerabild an.

Wren und Ivanov verwendeten der Beschreibung nach nur ein einziges Schachbrettmuster zur Kalibrierung (Wren und Ivanov, 2001). Um die Ergebnisse hier aber aussagekräftig vergleichbar zu machen, wurden für die Kalibrierung zu diesem Test dieselben, über die gesamte Null-Ebene verteilten Messpunkte für beide Algorithmen verwendet. Weiterhin wurden für den *FDS*-Algorithmus nicht die Deviationen vom linken ins rechte Bild berechnet, sondern – wie in Unterabschnitt 3.2.2 empfohlen – die Deviationen vom projizierten ins jeweilige Kamerabild. Damit sind die beiden Transformationen vergleichbar.

Für den realen Aufbau wurde die Projektion durch einen TFT-Monitor ersetzt. Dies hat aber keinen Einfluss auf die erhaltenen Ergebnisse. Die Breite des Monitors betrug 30 *cm*, seine Höhe 22,5 *cm*, der Kamera-Abstand 48 *cm* und die Kamerabasisbreite 12 *cm*. Die verwendeten Objektive hatten eine Brennweite von 4,3 *mm* – das entspricht einem vertikalen Öffnungswinkel von ungefähr 33 °. Die Kameras waren parallel ausgerichtet.

Durch diesen Aufbau erschien der Monitor in beiden Kamerabildern so groß, dass er am jeweils inneren Rand der Bilder gerade eben nicht abgeschnitten wurde. Eine kleinere Kamerabasisbreite konnte bedingt durch den Aufbau nicht gewählt werden. In der Simulation wurde – soweit möglich – derselbe Aufbau angelegt. Einen Überblick über die entstandenen Kamerabilder gibt Abbildung 3.8. In ihr sind die erkannten Ecken im Schachbrettmuster eingezeichnet.

3.5 Ergebnisse

Im folgenden Abschnitt werden sowohl die in der Simulation als auch die im realen Aufbau erhaltenen Ergebnisse vorgestellt. Dazu erfolgt zunächst die Auswertung der quadratischen Transformationsfehler sowohl des neuen als auch des *FDS*-Algorithmus. Anschließend folgen die Ergebnisse der Transformation in der Simulation sowie im realen Aufbau.

3.5.1 Quadratische Transformationsfehler des neuen Verfahrens und des FDS-Algorithmus

Wie in Abschnitt 3.4 beschrieben, wurden die quadratischen Fehler bei der Transformation im neuen sowie im *FDS*-Algorithmus ermittelt. Dazu wurde für 273 Messpunkte das arithmetische Mittel der jeweiligen quadratischen Fehler aus fünf Messungen ermittelt.

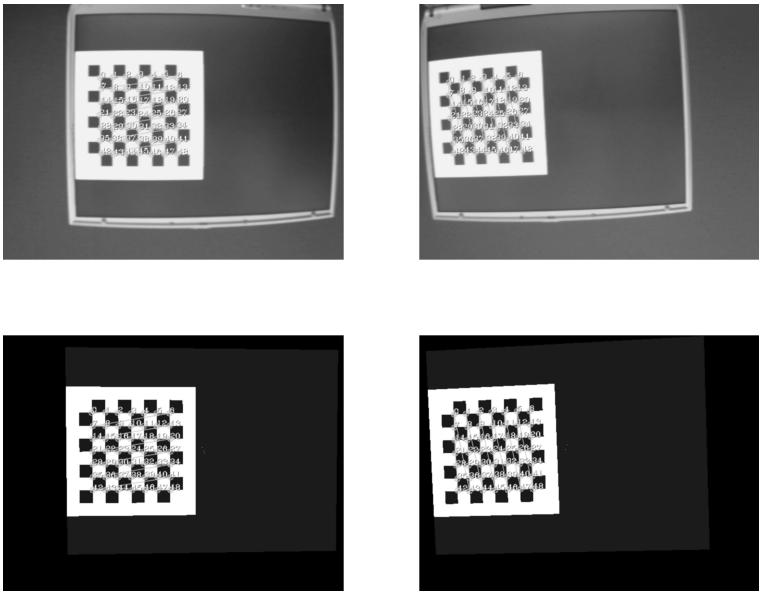


Abbildung 3.8: Kamerabilder aus realem Aufbau (oben) und Simulation (unten) Eingezeichnet sind die erkannten Ecken des Schachbrettmusters.

Tabelle 3.3: Mittlere quadratische Transformationsfehler des neuen und des *FDS*-Algorithmus in der Simulation und im realen Aufbau

	Simulation		**realer Aufbau**	
	neuer Alg.	***FDS*-Alg.**	**neuer Alg.**	***FDS*-Alg.**
linkes Bild	0,5099	0,5101	2,270	3,837
rechtes Bild	0,5099	0,5101	3,244	4,483

Der mittlere quadratische Fehler wurde über alle Punkte aller Messungen ebenfalls arithmetisch gemittelt. Rundungsfehler aufgrund der diskreten Pixelverteilung wurden bei der Berechnung bereits berücksichtigt. Einen Überblick über die erhaltenen mittleren quadratischen Fehler gibt Tabelle 3.3.

Beide Algorithmen erwiesen sich in der Simulation und damit unter rausch- und verzerrungsfreien Bedingungen als absolut gleichwertig. Dies ist sowohl am mittleren quadratischen Fehler als auch anhand der Diagramme in Abbildung 3.9 abzulesen. In diesen wurden die quadratischen Fehler der unteren, mittleren und oberen Messpunktreihe für das linke und rechte Bild aufgetragen. Zur Verdeutlichung wurden die Datenpunkte durch lineare Interpolation verbunden. Die entstandenen Kurven für den neuen Algorithmus (durchgezogene Linie) und den *FDS*-Algorithmus (gestrichelte Linie) waren deckungsgleich.

Im realen Aufbau ergaben sich dagegen deutliche Abweichungen: Der neue Algorithmus führte sowohl für das linke als auch für das rechte Bild zu einem geringeren mittleren quadratischen Transformationsfehler als der *FDS*-Algorithmus. Aus den Diagrammen in

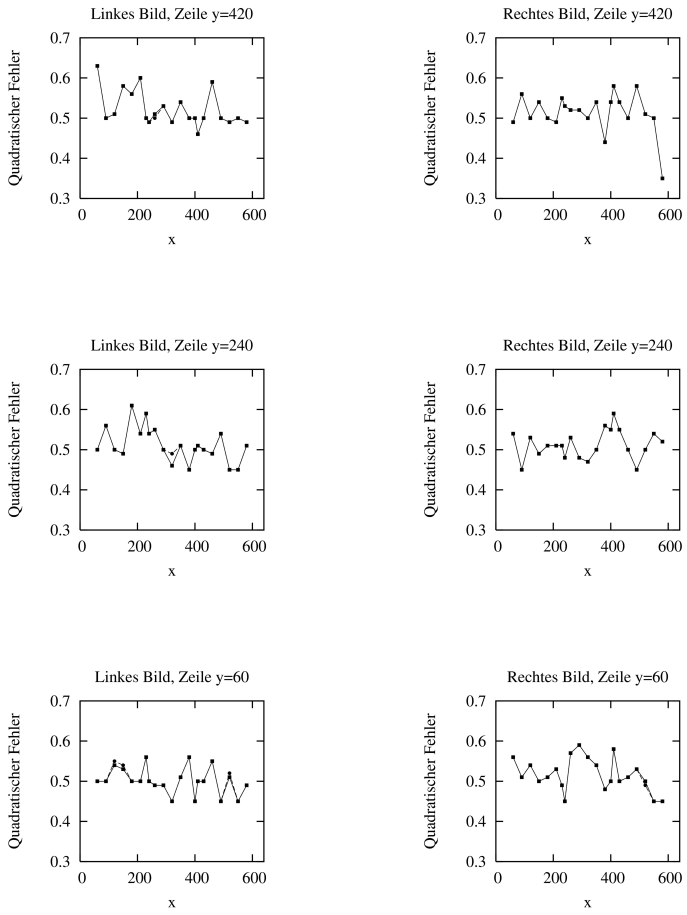


Abbildung 3.9: Quadratische Transformationsfehler von neuem und *FDS*-Algorithmus unter idealen Bedingungen in drei verschiedenen Bildzeilen
durchgezogene Linie: neuer Algorithmus
gestrichelte Linie: *FDS*-Algorithmus

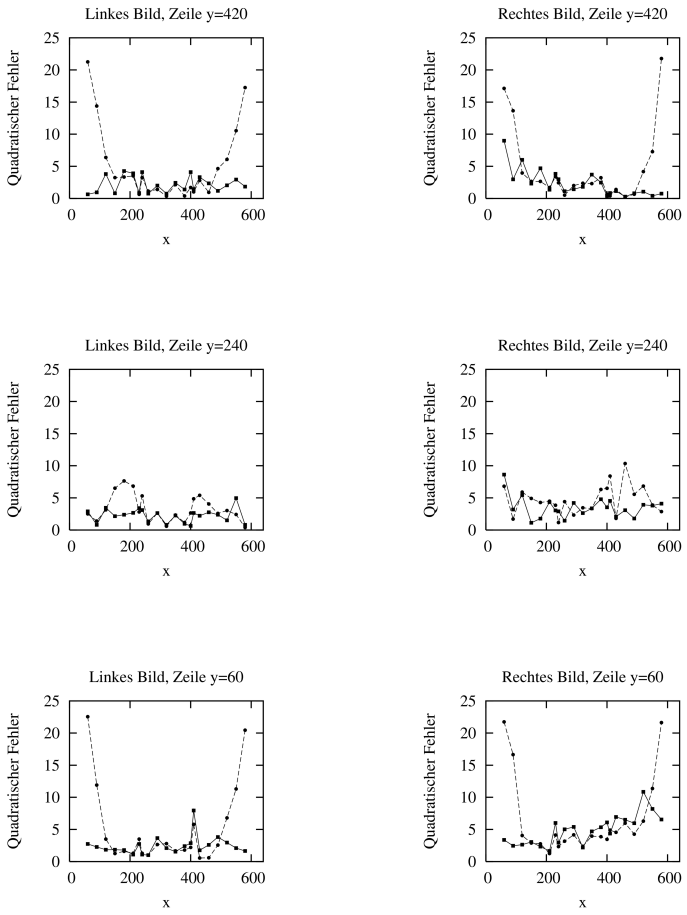


Abbildung 3.10: Quadratische Transformationsfehler von neuem und *FDS*-Algorithmus unter realen Bedingungen in drei verschiedenen Bildzeilen
durchgezogene Linie: neuer Algorithmus
gestrichelte Linie: *FDS*-Algorithmus

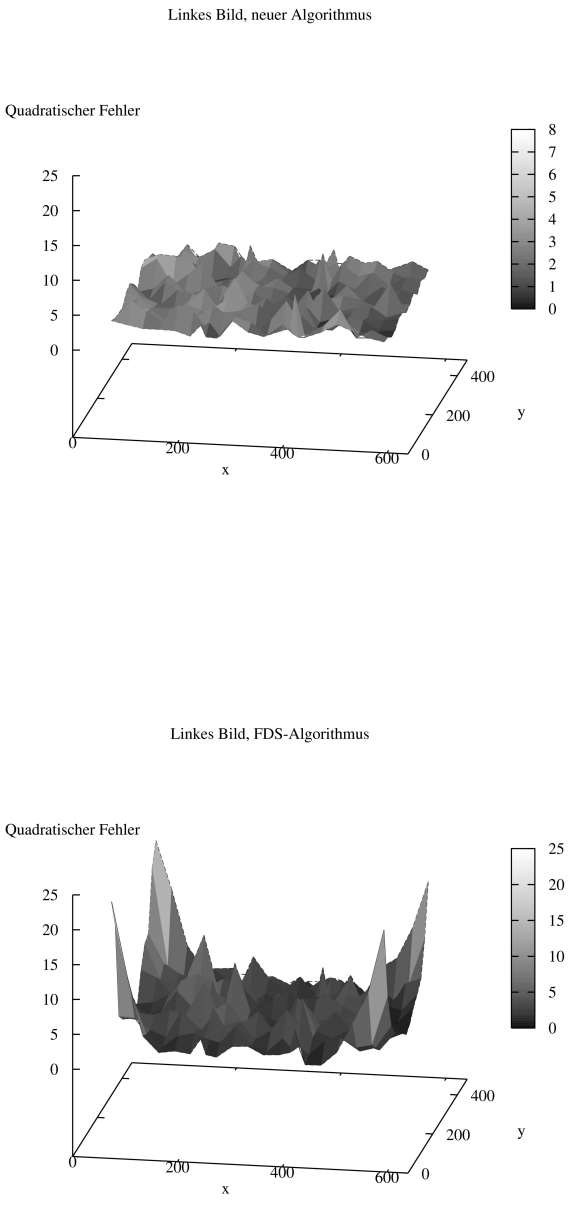
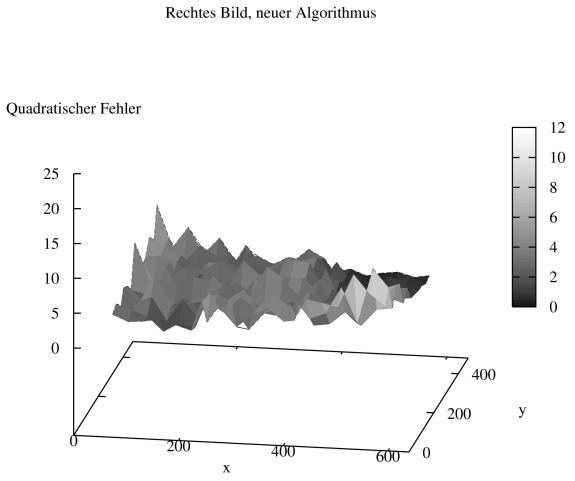


Abbildung 3.11: Quadratische Transformationsfehler von neuem und *FDS*-Algorithmus unter realen Bedingungen für das linke Bild
oben: neuer Algorithmus
unten: *FDS*-Algorithmus



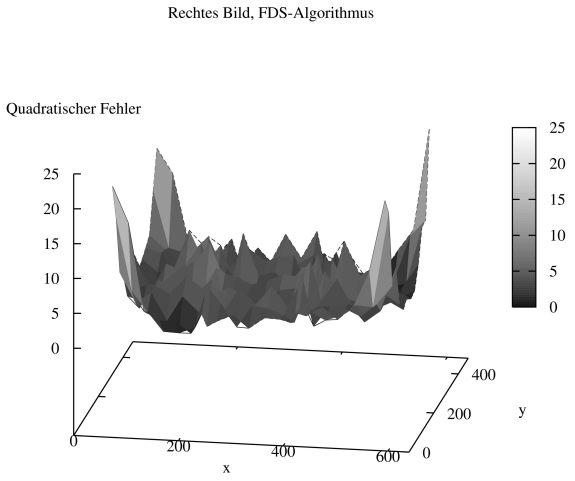


Abbildung 3.12: Quadratische Transformationsfehler von neuem und *FDS*-Algorithmus unter realen Bedingungen für das rechte Bild
oben: neuer Algorithmus
unten: *FDS*-Algorithmus

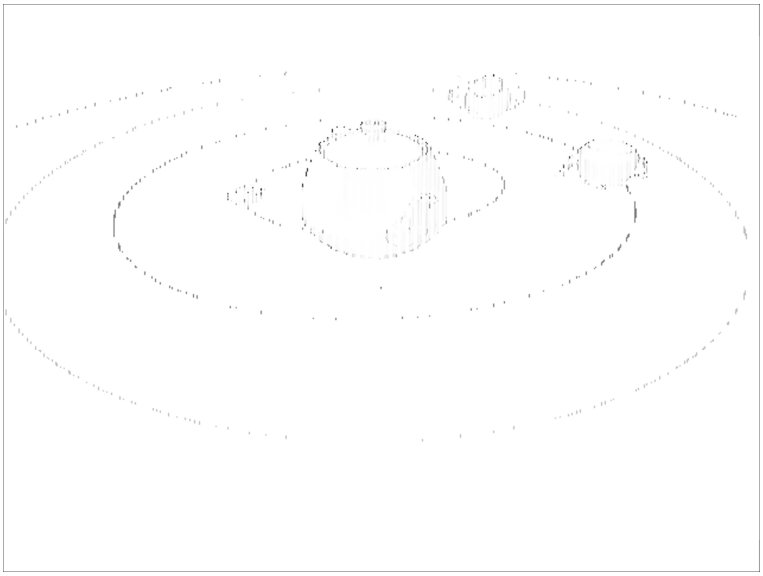


Abbildung 3.13: Differenzbild des kalibrierten Systems in der Simulation
Aus drucktechnischen Gründen wurde es lediglich invertiert und in Graustufen umgewandelt.

Abbildung 3.10 geht hervor, dass der quadratische Fehler des *FDS*-Algorithmus zu den Bildecken hin deutlich zunahm. Für den Messpunkt (60, 60) beispielsweise betrug dieser für die Transformation ins linke Kamerabild 22,55 – beinahe das Sechsfache des mittleren quadratischen Fehlers und damit das Maximum bei der Messung. Für die Bildmitte ergaben sich wiederum sehr ähnliche Ergebnisse wie beim neue Algorithmus. Letzterer liefert allerdings über die gesamte Bildfläche ein gleichmäßig gutes Transformationsergebnis. Aber auch hier ergaben sich für die Transformation ins rechte Kamerabild in der linken oberen und rechten unteren Bildecke etwas höhere Fehler.

Gleiches lässt sich auch aus den Diagrammen in den Abbildungen 3.11 und 3.12 entnehmen. Hier wurde für den neuen und den *FDS*-Algorithmus der quadratische Fehler der Transformation dreidimensional über der Bildfläche aufgetragen. Auch hier ist sehr deutlich die Zunahme des Transformationsfehlers beim *FDS*-Algorithmus in den Bildecken zu erkennen. Deutlich wird ebenfalls der nur leichte Anstieg des Fehlers beim neuen Algorithmus im rechten Bild.

Insgesamt fiel für den realen Aufbau auf, dass der Fehler bei beiden Algorithmen bei der Transformation ins linke Kamerabild geringer ist als bei der Transformation ins rechte.

3.5.2 Simulation

Nachdem das System in der Simulation für eine Kamerahöhe von 3 m und eine Projektionsfläche von3,2 m mal 2,4 m kalibriert worden war, war am Differenzbild deutlich zu erkennen, dass die gewünschte Transformation korrekt berechnet wurde: Die beiden

transformierten Kamerabilder waren deckungsgleich und hoben sich somit weitestgehend auf. Diese Differenz ist in Abbildung 3.13 dargestellt. An der Abbildung lässt sich der zuvor beschriebene – wenn auch sehr kleine – Fehler des Transformationsschrittes erkennen. Konturen und Bildbereiche, in denen ein Farbverlauf mit hohem Gradienten enthalten war, wurden nicht vollständig unterdrückt: Einzelne Pixel blieben übrig, wobei Konturen wesentlich größere Differenzwerte verursachten als Farbverläufe. Durch einen geeigneten kleinen *Threshold* ließen sich die Farbverläufe unterdrücken, die Konturpixel blieben allerdings erhalten und machten sich in der Maske störend bemerkbar.

Da es sich bei den beschriebenen Fehlern lediglich um Flächen von ein bis zwei Pixeln Breite handelte, wurden durch Anwendung des morphologischen Operators *Erode* auf die Differenz wesentlich bessere Ergebnisse erzielt als durch höhere *Thresholds*: Die Kontur in der Differenz verursachte keine Maskenfehler mehr. Das in der Simulation enthaltene Probe-Objekt – ein Quader von 10 *cm* Breite, 5 *cm* Höhe und 20 *cm* Tiefe mit einem dreidimensionalen Farbverlauf als Textur – wurde sicher erkannt.

Anhand der für dieses Objekt entstandenen Maske ließen sich die unterschiedlichen Auswirkungen von höheren *Thresholds* im Vergleich zur Anwendung des morphologischen Operators feststellen: Ein größerer Schwellwert verringerte zwar die Fehler durch die Kontur, verschlechterte aber auch die Objektmaske. Weiterhin konnte die Kontur nicht vollständig unterdrückt werden. Im Gegensatz dazu liefert die Anwendung des Operators *Erode* auf die Differenz schon bei sehr niedrigen *Thresholds* gute Maskierungsergebnisse (Abbildung 3.14). Die Anwendung des morphologischen Operators *Dilate* auf die binäre Maske, verbesserte deren Qualität weiter.

Das Verhalten von Differenz und Maske wurde auch für unterschiedliche Höhen des Probe-Objektes untersucht: Wie in Abbildung 3.15 zu sehen ist, entstand für die Probe, die sich in geringer Höhe über der Projektionsfläche befand, eine einzelne Region in der Maske. Je höher die Probe positioniert wurde, desto breiter wurde diese Region, bis schließlich zwei einzelne Objekte maskiert wurden. Gleiches Verhalten ließ sich den zugehörigen Differenzbildern entnehmen. Allerdings war hier zu erkennen, dass für geringe Höhen an der Objektkante viel größere Differenzen entstanden als auf der Objektfläche. Je größer die Höhe wurde, in der sich die Probe über der Projektionsfläche befand, desto mehr Anteile der Objektfläche verursachten große Differenzen. Aufgrund der in der Simulation möglichen sehr kleinen *Thresholds* entstand aber auch für die geringste Höhe eine ausgefüllte Maske der Objektfläche.

Zuletzt wurden in der Simulation die minimalen Erkennungshöhen eines Objektes ermittelt. Dazu wurde wiederum das oben beschriebene Probe-Objekt verwendet, das in seinen Außmaßen in etwa dem Spann eines durchschnittlichen Erwachsenen-Schuhs entspricht. Die minimalen Erkennungshöhen aus den Diagrammen in Abbildung 3.16 geben dabei die Höhen der oberen Fläche des Quaders an, die eine visuell eindeutig erkennbare Maskierung zur Folge hatten. Dabei wurden die Erkennungshöhen für die Kamerahöhen 2 *m* und 3 *m* bei verschiedenen Kamerabasisbreiten untersucht. Die zugehörigen Projektionsgrößen betrugen 1,6 *m* mal 1,2 *m* und 3,2 *m* mal 2,4 *m*, als Öffnungswinkel der Kameras wurden 50 ° und 60 ° gewählt, damit die Projektion bei den gegebenen Abständen noch vollständig zu sehen war. In den Diagrammen sind die gemessenen Höhen eingetragen. Zur Übersicht wurden die einzelnen Punkte zusätzlich durch ein kubisches Polynom angenähert.

Exemplarisch sei hier lediglich ein Ergebnis für jede Höhe angegeben: Für eine Kamerahöhe von 2 *m* und einen Öffnungswinkel von 50 ° wurde ab einer Kamerabasisbreite von ca. 40 *cm* das Probe-Objekt in einer Höhe von 2 *cm* erkannt. Für eine Kamerahöhe von

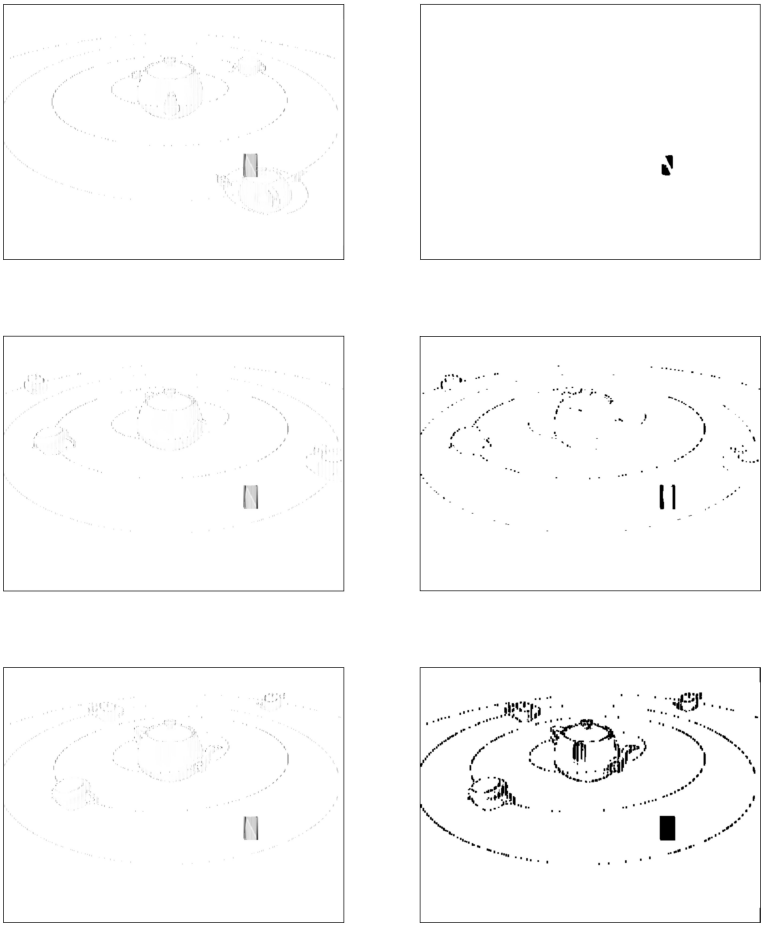


Abbildung 3.14: Differenzbild und Masken aus der Simulation
links: Differenzbilder
rechts – von unten nach oben: zugehörige Masken für die zwei *Thresholds* 0,01 und 1,00 ohne *Erode* sowie für den *Threshold* 0,01 mit *Erode*
Aus drucktechnischen Gründen wurden die Bilder zunächst invertiert und danach in Graustufen umgewandelt.

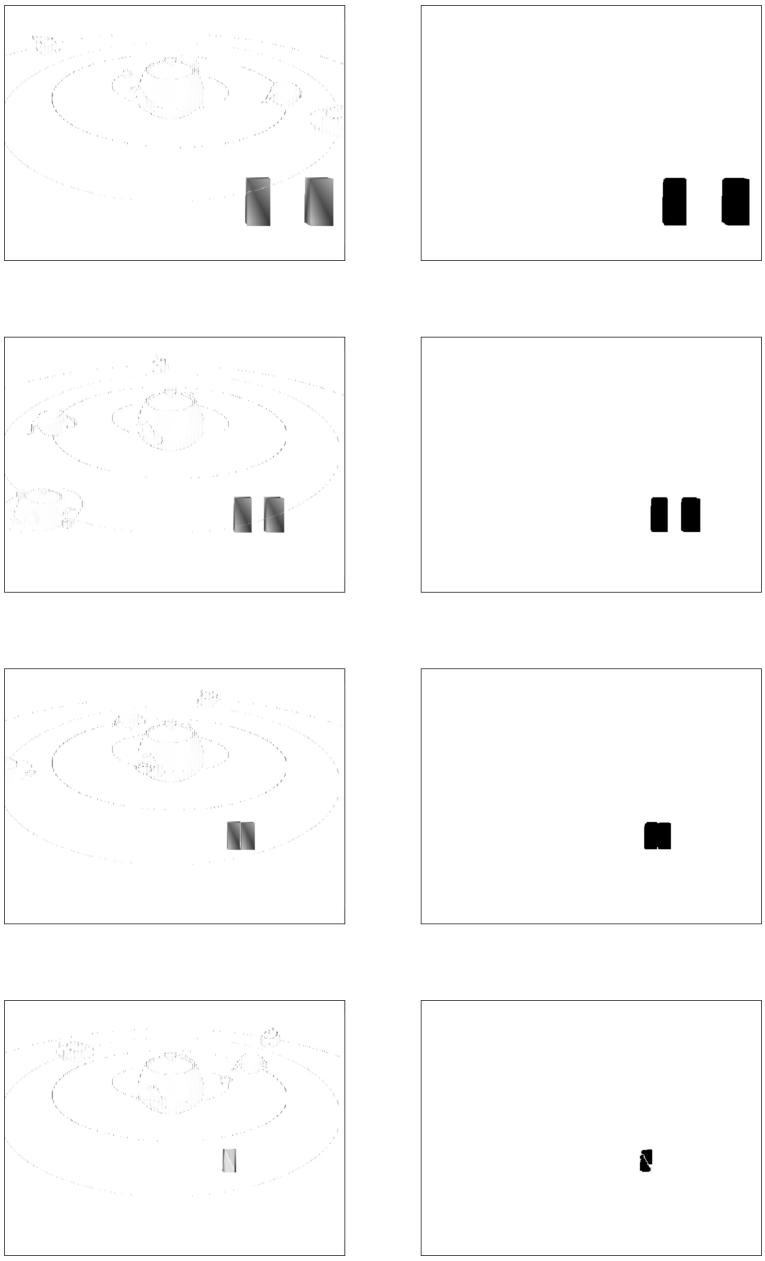


Abbildung 3.15: Differenzbild und Masken für verschiedene Höhen des Probe-Objektes
links: Differenzbilder; rechts: zugehörige Masken
von unten nach oben: vier verschiedene Höhen des Mittelpunktes der Probe über der Projektionsfläche (10, 60, 110 und 160 *cm*)
Aus drucktechnischen Gründen wurden die Bilder zunächst invertiert und danach in Graustufen umgewandelt.

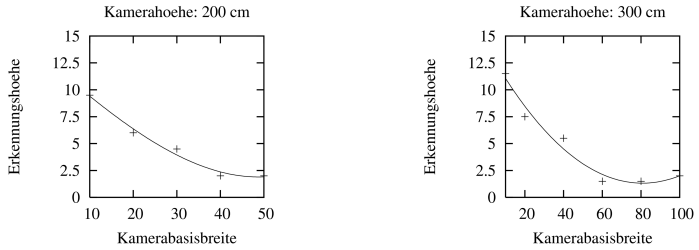


Abbildung 3.16: Minimale Erkennungshöhe eines 10 *cm* mal 5 *cm* mal 20 *cm* (BxHxT) großen Quaders in der Simulation
links: Kamerahöhe 2 *m*, Projektion 1,6 *m* mal 1,2 *m*, Öffnungswinkel 50 °
rechts: Kamerahöhe 3 *m*, Projektion 3,2 *m* mal 2,4 *m*, Öffnungswinkel 60 °

3 *m* und einen Öffnungswinkel von 60 ° betrug die minimale Kamerabasisbreite, ab der das Probe-Objekt in einer Höhe von 2 *cm* erkannt wurde, ca. 60 *cm*. Aus den Messungen geht hervor, dass die Erkennungshöhe zu kleineren Basisbreiten hin ansteigt. Sie steigt darüber hinaus zu größeren Kamerahöhen und damit auch zu abnehmender Fläche an, die die Projektion im Kamerabild einnimmt.

3.5.3 Realer Aufbau

Auch im getesteten realen Aufbau wurde nach der Kalibrierung mithilfe der Schachbrettmuster deutlich, dass die gewünschte Transformation korrekt berechnet wurde: Die transformierten Kamerabilder waren – abgesehen von den zuvor festgestellten kleinen Fehlern – im gesamten Bildbereich deckungsgleich. Die durch die Fehler übrigbleibenden Konturen ließen sich wiederum mittels des morphologischen Operators *Erode* aus der Differenz entfernen.

Allerdings traten durch die Verwendung zweier nicht farbkalibrierter Kameras Störungen im Differenzbild auf, die sich nur durch die Wahl eines höheren *Thresholds* in der Maske unterdrücken ließen. Der Versuch, beide Kameras manuell aneinander anzugleichen, gestaltete sich sehr aufwendig und führte nicht zu optimalen Ergebnissen. Abhängig von der Helligkeit der Bilder wichen die Intensitäten der Kamerabilder dennoch unterschiedlich stark voneinander ab.

Darüber hinaus ließen sich zum einen eine relativ große Latenz, zum anderen eine Asynchronität der beiden Kamerabilder erkennen. Beide schwankten aber von Test zu Test und über den Testzeitraum in sehr weiten Grenzen. Dazu muss bemerkt werden, dass die entwickelte Umgebung nicht auf Verarbeitungsgeschwindigkeit, sondern ausschließlich auf Flexibilität der Tests optimiert wurde.

In Abbildung 3.17 sind Differenzbilder und zugehörige Masken aus einem horizontalen realen Aufbau zu sehen. Bei diesem Test befand sich der Benutzer zwischen Kamera und senkrecht stehender Projektionsfläche. An den Differenzbildern ist zu sehen, dass wiederum nur kleine Transformationsfehler auftraten, die für Konturen im Hintergrund erhöhte

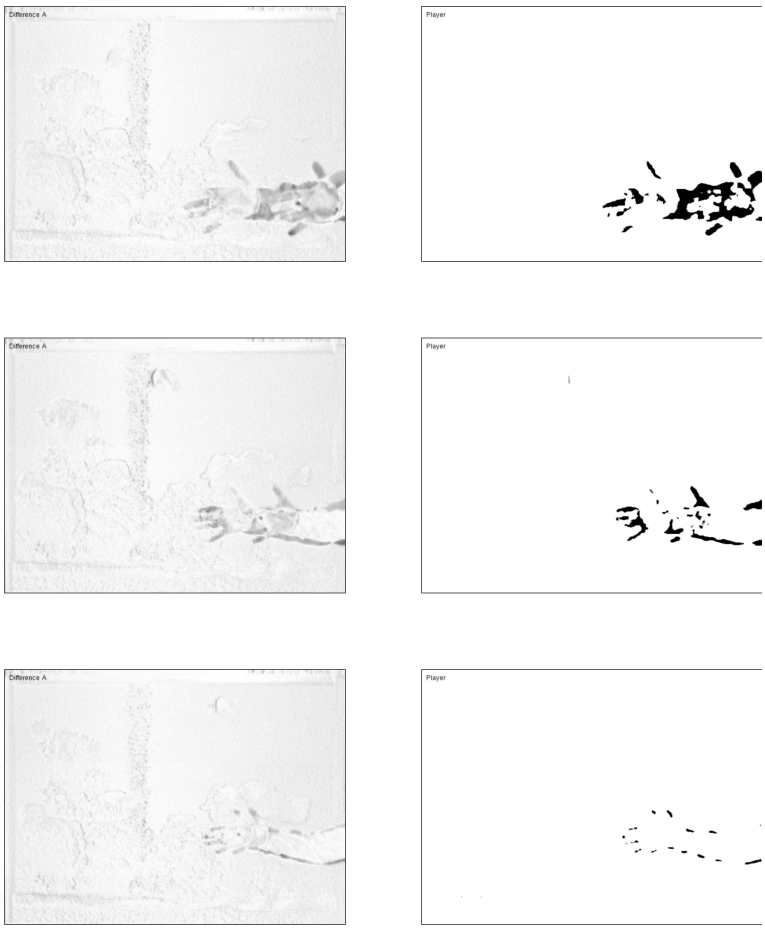


Abbildung 3.17: Differenzbilder und Masken aus einem horizontalen realen Aufbau
links: Differenzbilder; rechts: zugehörige Masken
von unten nach oben: größer werdender Abstand der Hand zur Projektionsfläche
Aus drucktechnischen Gründen wurden die Bilder zunächst invertiert und dann in Graustufen umgewandelt.

Differenzwerte entstehen ließen. Die übrigen Flächen erzeugten lediglich kleine Differenzwerte, die von den nicht kalibrierten Kameras herrührten. Bei einem kleinen Abstand einer Hand zur Projektionsfläche ergab sich nur für die Kontur eine große Differenz. Da der *Threshold* im realen Aufbau wegen der genannten Fehler relativ hoch gewählt und der morphologische *Erode*-Operator zur Unterdrückung der Transformationsfehler angewendet werden musste, wurde diese Konturlinie in der Maske lediglich stark unterbrochen dargestellt. Größere Abstände der Hand führten zu größeren Flächen mit hohen Differenzwerten, und damit größeren maskierten Bereichen. Der morphologische *Erode*-Operator hatte dadurch auf das Maskierungsergebnis keinen großen Einfluss mehr. Lediglich der notwendige hohe Schwellwert verursachte große Löcher in der Maske. Wie schon in der Simulation lässt sich auch in diesem realen Aufbau deutlich erkennen, dass prinzipbedingt Doppelbilder bei größeren Abständen eines Objektes zur Projektionsfläche entstehen. Bei diesem Test ließen sich – außer den beschriebenen Schwankungen der Kameras – keine beleuchtungsabhängigen Störungen bei der Maskenbildung beobachten.

Abbildung 3.18 zeigt Differenzbilder und zugehörige Masken aus einem vertikalen realen Aufbau. Dieser Aufbau entspricht einer späteren Installation. Hier konnte ebenfalls anhand der Differenzbilder die Korrektheit der Transformation bestätigt werden. Darüber hinaus wurde auch hier deutlich, dass die Kameras farblich nicht aufeinander abgestimmt waren. Das untere Bildpaar der Abbildung zeigt Differenzbild und Maske für den Fall, dass ein Benutzer seinen Fuß auf die Projektionsfläche setzt. Im Differenzbild ist eine schmale Kontur des Schuhs und der Hose zu erkennen, es erfolgt allerdings keine Maskierung. Wird der Fuß angehoben, werden die Bereiche großer Differenzen – wie zuvor auch schon beobachtet – immer größer, und auch die Maskierung erzeugt immer deutlichere Ergebnisse (zweites und drittes Bildpaar von unten). Im oberen Bildpaar der Abbildung sind Differenzbild sowie zugehörige Maske gezeigt, die entstehen, wenn ein Benutzer seine Hand über der Projektionsfläche ausstreckt. Auch hier entstanden sowohl in der Differenz als auch in der Maske Doppelbilder der Hand und des Armes. Diese Bilder waren räumlich voneinander getrennt und ermöglichten somit keine eindeutige Positionsbestimmung des Benutzers anhand der Maske. Bei diesem Test wirkte sich der notwendige hohe *Threshold* ebenfalls auf die Qualität der Maske aus: Nur Bereiche hoher Differenzen konnten sicher maskiert werden. Andere Teile, deren Differenzwerte kleiner waren, wurden nicht erkannt, und es entstand eine ebenfalls löchrige Maske. Wiederum ließ sich beobachten, dass das Maskierungsergebnis nicht von Beleuchtungsschwankungen beeinflusst wurde. Lediglich die beschriebenen Fehler aufgrund unzureichender Kalibrierung der Kameras traten auf.

Zuletzt wurde getestet, ob zur Vermeidung der auftretenden Doppelbilder der Erkennungs- und damit der Interaktionsbereich mithilfe der von Wren und Ivanov vorgeschlagenen Applikation *„Touch It“* eingeschränkt werden kann (Wren und Ivanov, 2001). Dabei ergab sich aber für die virtuelle Ebene nur ein sehr schmaler Bereich von etwa fünf Zentimetern, in dem sich die Differenz beider transformierter Kamerabilder Null näherte. Nur innerhalb dieses Bereiches konnte dann durch die in Abschnitt 3.2.1 beschriebene Kombination beider Masken eine Interaktion erkannt werden. Zusätzlich nahm die Qualität der so erzeugten Maske noch einmal deutlich ab: Zum einen konnten beide Masken – die der Null-Ebene und die der virtuellen Ebene – nicht genau zur Deckung gebracht werden. Dadurch wurden auch noch Bereiche mit großem Abstand zur Projektionsfläche maskiert. Zum anderen war die Maske der virtuellen Ebene ebenfalls löchrig, weshalb die Fläche der kombinierten Maske des beschränkten Bereiches nochmals wesentlich kleiner als die der einzelnen Masken war.

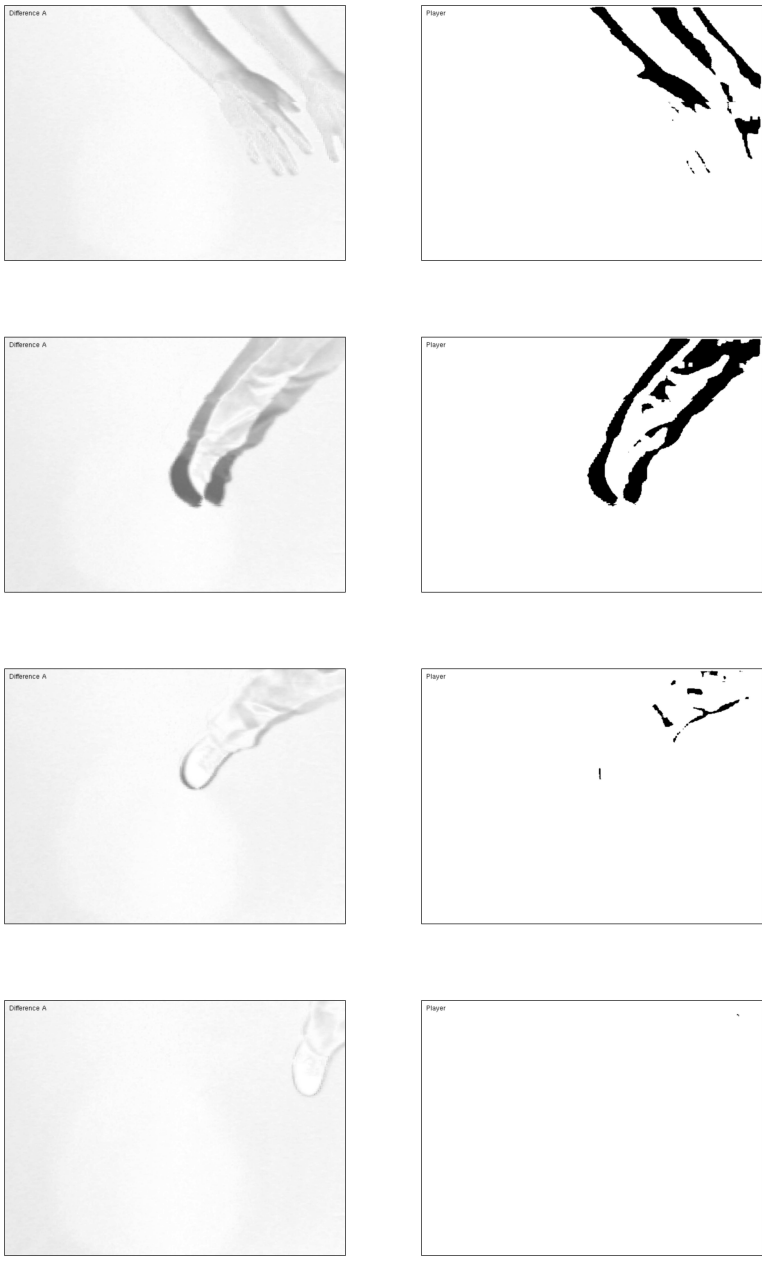


Abbildung 3.18: Differenzbilder und Masken aus einem vertikalen realen Aufbau
links: Differenzbilder; rechts: zugehörige Masken
von unten nach oben: größer werdender Abstand des Anwenders zur Projektionsfläche
Aus drucktechnischen Gründen wurden die Bilder zunächst invertiert und dann in Graustufen umgewandelt.

3.6 Diskussion

Zunächst wurde mithilfe von Tests nachgewiesen, dass der in dieser Arbeit vorgeschlagene Vorverarbeitungs-Algorithmus – insbesondere der darin enthaltene Kalibrierungsschritt – im realen Aufbau deutlich genauere Ergebnisse liefert als der *FDS*-Algorithmus. Dies gilt vor allem in Bereichen hoher Linsenverzerrung, da diese im neuenVerfahren ausdrücklich berücksichtigt werden. Dennoch ist auch der neue Algorithmus nicht vollständig genau: Aufgrund des diskreten Wertebereiches der Pixelkoordinaten ergaben sich in der Simulation Rundungsfehler. Im realen Aufbau kamen weitere Fehler durch Kamerarauschen und Linsenverzerrungen hinzu, die insgesamt zu kleinen Fehlern im Transformationsschritt führten. Es wurde gezeigt, dass sich diese mithilfe des morphologischen Operators *Erode* vollständig unterdrücken ließen und eine Maskierung möglich war.

Die in der Simulation empirisch ermittelten Erkennungshöhen deuteten darauf hin, dass auch im realen Aufbau kleine Objekte wie Schuhe auf der Projektionsfläche erkannt werden können. Aufgrund der großen auftretenden Störungen musste hier allerdings der *Threshold* so groß gewählt werden, dass eine saubere Erkennung nicht möglich war. Auch die entstandene Konturlinie in der Differenz für kleine Objekte war nicht so breit, dass nach erfolgter Erosion eine sichere Maskierungsentscheidung getroffen werden konnte: Die sicheren Erkennungshöhen liegen im realen Ausbau demnach deutlich höher als dies aus den simulierten Ergebnissen geschlossen werden kann. Es zeigte sich aber auch hier eine Abhängigkeit von der Kamerahöhe. Dies wiederum lässt den Schluss zu, dass die Erkennungshöhe neben der Kamerabasisbreite ausschließlich von der zur Verfügung stehenden Auflösung abhängt: Je weiter die Kamera von der Projektionsfläche entfernt ist, auf desto weniger Pixel wird ein kleines Objekt abgebildet und desto größer ist dessen Erkennungshöhe. Um diese also zu reduzieren, müssen entweder eine größere Basisbreite oder höher auflösende Kameras gewählt werden, die für die gegebene Anwendung aber aus wirtschaftlichen Gründen nicht sinnvoll sind. Ein weiterer Faktor, der die Erkennungshöhe bei diesem Verfahren heraufsetzte, war der *Erode*-Operator. Durch diesen werden schmale Konturen unterdrückt, die zum einen durch geringe Objekthöhen, zum anderen aber auch durch Transformationsfehler hervorgerufen werden. Um diese Transformationsfehler zu vermeiden und damit den notwendigen Einflussbereich des *Erode*-Operators zu reduzieren, wären genauere Verfahren zur geometrischen Kalibrierung erforderlich. Diesbezüglich gilt es zu untersuchen, ob und wie sich die von *OpenCV* bereitgestellten Verfahren verbessern lassen.

Auch die Tatsache, dass die verwendeten Kameras nicht farbkalibriert waren, wirkte sich störend auf eine sichere Maskenbildung aus. Aus diesem Grund entstanden auch für einfarbige Bereiche der Projektion noch große Differenzen, was einen hohen *Threshold* erforderlich machte, der wiederum die Maskenqualität reduzierte. Werden die Kameras im Kalibrierungsschritt farblich aufeinander abgeglichen, dürften sich hier eindeutig bessere Ergebnisse erzielen lassen: Es ist zu erwarten, dass der *Threshold* auch für einen realen Aufbau deutlich kleiner gewählt werden kann und somit die Maske deutlicher wird. Dies wurde in der Simulation nachgewiesen. Eine für den realen Aufbau ausreichende Farbkalibrierung konnte im Rahmen dieser Diplomarbeit jedoch nicht erfolgen. Vielversprechende Ansätze zur Kalibrierung zweier Kameras sowie zur Kalibrierung eines Systems aus Projektor und Kamera finden sich sowohl bei Porikli als auch bei Juang und Majumder (Porikli, 2003; Juang und Majumder, 2007). Darauf aufbauend könnten neben dem geometrischen Kalibrierungsschritt für die Transformation in der Initialisierungsphase auch

Farbkalibrierungen durchgeführt werden.

Ein Problem des hier vorgeschlagenen neuen Algorithmus zur Vorverarbeitung eines Stereo-Bildpaares stellen die entstehenden Doppelbilder bei großer Objekthöhe dar. Prinzipbedingt befindet sich der gesamte Körper des Benutzers bei dem gegebenen vertikalen Aufbau parallel zur optischen Achse der Kamera. Damit ist der Kopf eines Benutzers wesentlich weiter von der Null-Ebene entfernt als die Füße. Somit ergeben sich sehr große Bereiche, die maskiert werden, aber nicht mehr direkt mit der Position des Benutzers in Verbindung gebracht werden können. Größere Kamerabasisbreiten zur Verringerung der Erkennungshöhe zogen diesbezüglich eine größere räumliche Trennung dieser Doppelbilder nach sich.

Der Versuch, den Erkennungsbereich vertikal zu begrenzen, lieferte auch kein zufriedenstellendes Ergebnis. Zum einen entstanden dadurch weitere Störungen, zum anderen ist der Bereich, auf den die Erkennung reduziert würde, zu klein, um einen sinnvollen Einsatz dieses Verfahrens für die gegebenen Anwendungen zu ermöglichen: Beispielsweise wäre es für das von *rmh* entwickelte Fußballspiel nicht praktikabel, dass Schussbewegungen nur innerhalb eines etwa fünf Zentimeter hohen Bereiches über dem Boden erkannt werden. Um die beschriebenen Doppelbilder zu vermeiden, schlagen Ivanov *et al.* die Verwendung weiterer Kameras vor (Ivanov *et al.*, 2000). Danach ließen sich mit drei Kameras auch drei Masken für die möglichen Kamerapaare erzeugen, aus denen dann die Schnittmenge gebildet werden könnte. Dabei müsste geprüft werden, ob eine dritte Kamera wirtschaftlich tragbar wäre und ob der zusätzlich nötige Rechenaufwand eine gleichzeitige Ausführung der Anwendungslogik noch ermöglichen würde.

Bezüglich der Verarbeitungsgeschwindigkeit ist es für eine konkrete Anwendung zwingend erforderlich, dass beide Kameras – zumindest annähernd – synchron Bilder liefern. Ist dies nicht möglich, verursachen Bewegungen im Hintergrund deutliche Unterschiede zwischen den beiden transformierten Kamerabildern, und die gewünschte Unterdrückung des Hintergrundes kann nicht mehr erfolgen. Für eine konkrete Anwendung muss die Bildaufnahme soweit optimiert werden, dass zu jedem Zeitpunkt eine möglichst hohe Synchronität gewährleistet ist. Dennoch wird – zumindest bei Standard-Hardware – nicht der Fall eintreten, dass beide Bilder exakt im selben Moment aufgenommen werden. Das verwendete Projektionssystem muss innerhalb dieses – wenn auch sehr kurzen – Zeitraumes sicherstellen, dass sich das projizierte Bild nicht verändert. *DLP*-Projektoren mit Farbrad genügten dieser Anforderung bei den erfolgten Tests nicht: Zu den Farbungenauigkeiten der Kameras kamen die Farbveränderungen der Projektion während der Bildaufnahme noch hinzu und verschlechterten das Maskierungsergebnis. Diesbezüglich müssen noch weitere Tests mit verschiedenen Projektionssystemen erfolgen.

Aus den erhaltenen Ergebnissen lässt sich feststellen, dass das hier vorgeschlagene Verfahren zur Vorverarbeitung für die gegebenen Anwendungen mit vertikalem Aufbau nicht praktikabel ist, obwohl die grundsätzliche Funktionalität nachgewiesen wurde: Zum einen sind die Störungen durch Doppelbilder zu hoch, zum anderen ist bei dem o. g. Aufbau die Auflösung in der Nähe der Projektionsebene zu gering. Folglich darf in einer Anwendung der Abstand sowohl zwischen Benutzer und Null-Ebene als auch zwischen Kamera und Null-Ebene nicht zu groß werden. Weiterhin muss die Fläche der Projektion klein gehalten werden, damit in ihrer Nähe eine ausreichend große Auflösung zur Verfügung steht.

Ein Vergleich der im Rahmen dieser Arbeit erhaltenen Ergebnisse mit denen, die Wren und Ivanov für ihren *FDS*-Algorithmus und die Applikation *„Touch It“* erhalten haben (Wren und Ivanov, 2001), legt nahe, dass der Anwendungsbereich dieses Verfahrens sowie

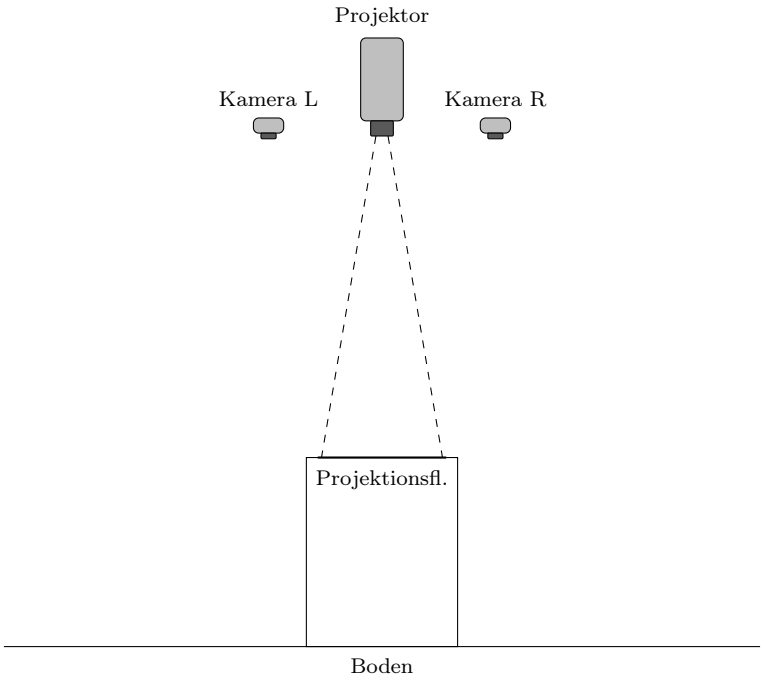


Abbildung 3.19: Möglicher vertikaler Aufbau für manuelle Interaktion

des hier entwickelten neuen Algorithmus auf Installationen mit kleinen Abmessungen beschränkt ist. Dadurch lassen sich mit Standard-Hardware hohe Auflösungen im Bereich der Projektionsebene und damit niedrige Erkennungshöhen erreichen. Weiterhin muss zur Vermeidung von Doppelbildern durch den Aufbau sichergestellt werden, dass der von den Benutzern eingenommene Raum entlang der optischen Achse der Kameras eine kleine Ausdehnung hat und sich in geringem Abstand zur Projektionsfläche befindet.

Bei horizontalem Aufbau ließe sich der Bewegungsbereich der Benutzer soweit einschränken, dass das Entstehen von Doppelbildern auf ein verträgliches Maß begrenzt wird. Auch vertikale Aufbauten scheinen möglich, aber dabei darf der Spieler nicht soweit in den Bereich der Projektion treten können, dass sein gesamter Körper in den Sichtbereich der Kameras gerät. Für eine Interaktion mit den Füßen wird sich diese Randbedingung schwer einhalten lassen. Einfacher wird sich dies jedoch bei Installationen nach Abbildung 3.19 gestalten. Hierbei kann der Benutzer mit seinen Händen interagieren, die sich in nur geringem Abstand über der erhöhten, kleinen Projektionsfläche befinden. Aufgrund des Aufbaus kann er aber die Projektionsfläche selbst nicht betreten.

4 Abschließende Betrachtung und Ausblick

Abschließend wird eine Kombination aus dem in Kapitel 3 hergeleiteten Vorverarbeitungsschritt und dem in Kapitel 2 favorisierten Trennungsverfahren erörtert. Dies geschieht unter der Annahme, dass die in Abschnitt 3.6 genannten Bedingungen eine praktikable Installation zulassen. Da sich diese Kombination aus den im vorhergehenden Abschnitt genannten Gründen für Aufbauten gemäß Abschnitt 1.4 allerdings nicht eignet, wird anschließend ein Vorschlag zur Verbesserung des existierenden Verfahrens lediglich auf Grundlage der Ergebnisse aus Kapitel 2 gemacht. Danach werden Anregungen für weitergehende Untersuchungen gegeben.

4.1 Kombination aus neuem Vorverarbeitungsschritt und blockbasiertem Kollinearitätstest

Der im Rahmen dieser Arbeit entwickelte Vorverarbeitungsschritt wurde so konzipiert, dass er mit dem blockbasierten Kollinearitätstest kombiniert werden kann. Dazu wird zunächst ein Stereo-Bildpaar mithilfe der Kalibrierungsdaten transformiert.

Da beide Bilder nun deckungsgleich sind, kann – wie aus Abbildung 4.1 ersichtlich – eines das Bild I_B, das andere das Bild I_t des Trennungsverfahrens ersetzen. Beide Bilder sind zur selben Zeit aufgenommen worden, zeigen also beide sowohl die Hintergrund-Ebene als auch eventuelle Objekte davor in immer exakt demselben Zustand – eine Voraussetzung, die ein gespeicherter Hintergrund nicht erfüllt. Somit wird die für den blockbasierten Kollinearitätstest bereits festgestellte Beleuchtungsunabhängigkeit noch weiter verbessert: In beiden Bildern erfährt die Null-Ebene dieselben Veränderungen durch wechselnde Beleuchtung. Dadurch, dass in Kapitel 2 gezeigt werden konnte, dass dieses Trennungsverfahren auch bei kleinen Störungen der einzelnen Bilder noch sehr stabile und verlässliche Ergebnisse liefert, werden sich auch kleine Ungenauigkeiten der Kameras bei Weitem nicht so deutlich bemerkbar machen wie dies bei den Tests des Vorverarbeitungsschrittes noch der Fall war. Dennoch muss großer Wert auf eine genaue gegenseitige Anpassung der Kameras gelegt werden. Dies konnte im Rahmen dieser Arbeit nicht erfolgen und erfordert somit noch weitere Untersuchungen.

Im Ganzen betrachtet, sollte sich also ein hochwertiges Verfahren ergeben, das eine sichere Trennung von Vorder- und Hintergrund bei wechselnder Beleuchtung und sicht-

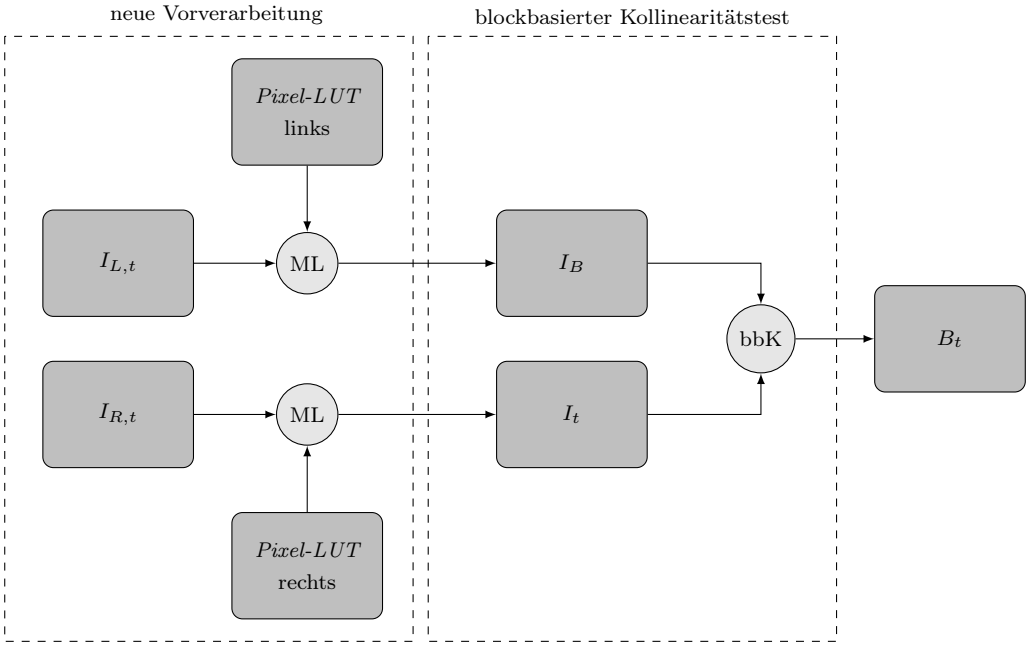


Abbildung 4.1: Kombination von Vorverarbeitung und Trennungsschritt zur Interaktionserkennung vor bewegtem Hintergrund
ML: Operator zum Auslesen der durch die *Pixel-LUT* gegebenen Speicherstellen
bbK: Operator für den blockbasierten Kollinearitätstest

barer Projektion bewegter Hintergründe ermöglicht. Da es sich für die in Abschnitt 1.4 beschriebenen Anwendungen aus den genannten Gründen aber nicht eignet, wurde auf eine kombinierte Implementation im Rahmen dieser Arbeit verzichtet.

4.2 Verbesserung des bestehenden Verfahrens auf Grundlage der Ergebnisse

Anstatt beide Verfahren zu kombinieren, wird hier erörtert, inwieweit die alleinige Anwendung des Trennungsverfahrens eine Verbesserung der bestehenden Installation darstellt.

Zur Interaktionserkennung bei den Spielen der Firma *rmh* wird die Differenz des aktuellen Kamerabildes gegen ein gespeichertes Hintergrundbild errechnet. Dabei werden die Bilder durch einen geeigneten Filter vor der Kamera auf das nahe *IR*-Spektrum begrenzt. Diese Differenz wird mit einer Gauß'schen Wahrscheinlichkeitsverteilung verglichen, die für jede konkrete Installation in der Initialisierungsphase ermittelt wird. Aus diesem Vergleich resultiert die Klassifizierung einzelner Pixel als *changed* oder *unchanged*. Prinzipiell handelt es sich also um ein einfaches Differenzverfahren. Für diese Gruppe von Verfahren wurden die Auswirkungen unterschiedlicher *Thresholds* auf das Maskierungsergebnis bei verschiedenen Hintergründen im Rahmen dieser Arbeit untersucht: Alle erwiesen sich bei wechselnder Beleuchtung – insbesondere durch Sonneneinstrahlung, die große spektrale Anteile im *IR*-Bereich hat – als störanfällig.

Zur beleuchtungsunabhängigen Trennung von Vorder- und Hintergrund ließ sich in Kapitel 2 mit dem blockbasierten Kollinearitätstest von Mester *et al.* in der Verbesserung von Griesser *et al.* ein geeignetes Verfahren finden und dessen Stabilität bestätigen (Mester *et al.*, 2001; Griesser *et al.*, 2005). Im Rahmen dieser Arbeit wurden zur Vektorbildung in diesem Verfahren lediglich die RGB-Farbkomponenten sowie die Chrominanzen verwendet. In beiden Fällen konnten ähnlich gute Testergebnisse erzielt werden. Mester *et al.* verwendeten für ihren Ansatz nur die Luminanzkomponenten (Mester *et al.*, 2001). Da diese Helligkeit in den Farbkomponenten implizit enthalten ist, werden sich dafür ähnlich gute Ergebnisse erzielen lassen.

Bei der existierenden Installation von *rmh* ist durch die Filterung ebenfalls nur die Helligkeit Grundlage der Trennung. Wenn sich das verwendete Infrarotlicht ähnlich verhält wie die Luminanzen im sichtbaren Licht, ist davon auszugehen, dass sich mit dem blockbasierten Kollinearitätstest bessere Ergebnisse – vor allem im Hinblick auf Beleuchtungsunabhängigkeit – erzielen lassen. Der bewegte Hintergrund würde dann weiterhin vom *IR*-Filter unterdrückt.

Diesbezüglich müssen aber weitere Tests erfolgen. Besonderes zu beachten ist dabei die Spezialisierung auf den nahen *IR*-Bereich. Darüber hinaus sind einige Tests notwendig, die den Zusammenhang von statischem *Threshold* τ_s und der positiven Konstante C verdeutlichen. Auch eine günstige Blockgröße muss für die speziellen Bedingungen unter *IR*-Beleuchtung ermittelt werden. Zusätzlich ist zu überlegen, ob der blockbasierte Kollinearitätstest als Ergänzung zum einfachen Differenzverfahren implementiert wird. Dies könnte die Genauigkeit der Erkennung noch weiter erhöhen.

4.3 Ausblick

Für die im vorigen Abschnitt beschriebene Verbesserung des existierenden Verfahrens sind noch einige Tests mit dem blockbasierten Kollinearitätstest nötig. Zu diesem Zweck kann die im Rahmen dieser Arbeit entstandene Software verwendet und weiterentwickelt werden. Mit dieser lassen sich beispielsweise Tests mit verschiedenen Kameras und Beleuchtungen durchführen.

Die für die Untersuchungen in Kapitel 3 entwickelte Simulation ist ebenfalls geeignet, diese Experimente zu unterstützen, muss aber auf die Verwendung einer einzelnen Kamera angepasst werden. Möglicherweise lässt sich damit auch die Kalibrierung noch weiter verbessern: Dazu sollten Schachbrettmuster als Marker verwendet werden, die sich von der Software automatisch erkennen lassen.

Abschließend ist es allem Anschein nach auch für die *IR*-basierte monoskopische Trennung sinnvoll, eine gute optische Kalibrierung der Kamera durchzuführen. Wie bereits erwähnt, finden sich in der Literatur vielversprechende Ansätze (Porikli, 2003; Juang und Majumder, 2007). Allerdings kann dazu aufgrund der Filterung nicht das projizierte Bild verwendet werden, möglicherweise aber ebenfalls Marker mit Schachbrettmustern.

In jedem Fall muss die hier entwickelte Software weiter getestet und auf Fehler untersucht werden: Im Rahmen dieser Arbeit konnte die Programmierung nur so weit gehen, dass ein lauffähiges Programm entstand. Es sollte lediglich die beschriebenen Tests ermöglichen, nicht aber eine stabile Ausführung unter allen denkbaren Bedingungen sowie Lauffähigkeit auf anderen Hardware-Systemen garantieren. Nicht enthalten sind beispielsweise Routinen zur Fehlerbehandlung, da Randbedingungen bei den durchgeführten Tests leicht eingehalten werden konnten. Für eine spätere Anwendung müssen aber geeignete Kontrollstrukturen vorhanden sein. Weiterhin verwendet die Testumgebung für den neuen Vorverarbeitungsschritt die noch nicht als stabil eingestufte Bibliothek *libdc1394* in der aktuellen Entwicklungsversion. Dies führte beim Start der Anwendung mit angeschlossenen Kameras oft zu Programmabstürzen, die sich durch erneute Ausführung umgehen ließen. Um die Software aber sinnvoll weiter nutzen zu können, sollte die Entwicklung von *libdc1394* verfolgt und das Programm an deren Neuerungen angepasst werden.

Danksagung

Zu allererst möchte ich Professor Dr.-Ing. Sina Mostafawy für die Stellung der Aufgabe und die Betreuung der Arbeit sowie ganz besonders für seine Unterstützung danken. Mein Dank gilt auch Dipl.-Inform. Jérôme Thoma für die Betreuung, seine guten Ratschläge sowie die Übernahme des Korreferats dieser Arbeit.

Herzlichen Dank auch an die Firma *rmh*, die mich großzügig mit dem nötigen Material zur Durchführung meiner Untersuchungen unterstützte.

Bei meinen Kommilitonen Christian Wallmeier und René Bannasch möchte ich mich für die schöne Zeit während des Studiums bedanken.

Thanks alot to Wayne Cowan for some really good thoughts on the subject of this thesis and his inspiring ideas. Many thanks also for proofreading the English abstract.

Mein Dank gilt auch meiner Tante Edeltraud Stumpe sowie meinen Eltern für die überaus mühsame Korrektur dieser Arbeit – ich möchte nicht mit Euch tauschen ... Meinen Eltern danke ich außerdem für die nicht nur finanzielle Unterstützung meines Studiums.

Mein ganz besonderer Dank geht an Birte Posch, die als Allererste meine Arbeit korrigiert hat. Danke, dass Du solche Ungetüme gezähmt hast:

Der von Kamkar-Parsi et al. *vorgeschlagenen Erweiterung wird hier noch eine weitere hinzugefügt: Aufgrund der Tatsache, dass die in dunklen Bereichen entstehenden kurzen Vektoren durch Kamerarauschen eine größere Winkelabweichung erfahren als längere, muss, damit die Aussage der* Change Mask *nicht durch solche Störungen beeinflusst wird, auch für die Vektorlänge – ähnlich wie für die Sättigung bei der Chrominanzabweichung (Unterabschnitt 2.2.2) – ein Schwellwert eingeführt werden.*

Und:

dabei nahm die Qualität zunehmend ab

Viel wichtiger ist aber die Tatsache, dass Du es immer ausgehalten hast, wenn ich über „normal-verteilte Grundgesamtheiten im n-dimensionalen Raum orthogonal zu $\overrightarrow{v}$“ gesprochen habe und Du mich in den richtigen Momenten immer wieder aufgebaut hast.

Danke!

Literaturverzeichnis

[Aach und Kaup 1995] Aach, Til ; Kaup, André: Bayesian Algorithms for Change Detection in Image Sequences Using Markov Random Fields. In: *Signal Processing: Image Communication* 7 (1995), Nr. 2, S. 147–160

[Aach *et al.* 1993] Aach, Til ; Kaup, André ; Mester, Rudolf: Statistical Model-Based Change Detection in Moving Video. In: *Signal Processing* 31 (1993), Nr. 2, S. 165–180

[Bahadori *et al.* 2005] Bahadori, Shahram ; Iocchi, Luca ; Leone, G. R. ; Nardi, Daniele ; Scozzafava, L.: Real-time people localization and tracking through fixed stereo vision. In: *IEA/AIE 2005: Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence*. London, UK : Springer-Verlag, 2005, S. 44–54

[Batter und Brooks, Jr. 1971] Batter, James J. ; Brooks, Jr., Frederick P.: GROPE-I: A computer display to the sense of feel. In: *IFIP 1997: Proceedings of the International Federation for Information Processing Congress* Bd. 1, 1971, S. 759–763

[Bérard 1999a] Bérard, François: *The MagicBoard Project*. 1999. – URL `http://iihm.imag.fr/demos/magicboard/`

[Bérard 1999b] Bérard, François: The perceptual window: Head motion as a new input stream. In: Sasse, A.M. (Hrsg.) ; Johnson, C. (Hrsg.): *IFIP 1999: Proceedings of the International Federation for Information Processing Conference on Human-Computer Interaction*, IOS Press, 1999, S. 238–244

[Bérard 1999c] Bérard, François: *The Perceptual Window Movies*. 1999. – URL `http://iihm.imag.fr/demos/pwindow/`

[Bérard 2003] Bérard, François: The Magic Table: Computer-Vision Based Augmentation of a Whiteboard for Creative Meetings. In: *PROCAMS 2003: CD-ROM proceedings of the IEEE International Conference in Computer Vision, Workshop on Projector-Camera Systems*, 2003

[Bobick *et al.* 1999] Bobick, Aaron F. ; Intille, Stephen S. ; Davis, James W. ; Baird, Freedom ; Pinhanez, Claudio S. ; Campbell, Lee W. ; Ivanov, Yuri A. ; Schütte, Arjan ; Wilson, Andrew: The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment. In: *Presence: Teleoperators and Virtual Environments* 8 (1999), Nr. 4, S. 369–393

[Bouguet 2007] Bouguet, Jean-Yves: *Camera Calibration Toolbox for Matlab*. 2007. – URL `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`

[Brooks *et al.* 2004] Brooks, Andrew G. ; Gray, Jesse ; Hoffman, Guy: Robot's play: interactive games with sociable machines. In: *ACE 2004: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA : ACM Press, 2004, S. 74–83

[Brooks, Jr. *et al.* 1990] Brooks, Jr., Frederick P. ; Ouh-Young, Ming ; Batter, James J. ; Kilpatrick, P. J.: Project GROPE - Haptic displays for scientific visualization. In: *SIGGRAPH 1990: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM Press, 1990, S. 177–185

[Camurri *et al.* 2004] Camurri, Antonio ; Coletta, Paolo ; Massari, Alberto ; Mazzarino, Barbara ; Peri, Massimiliano ; Ricchetti, Matteo ; Ricci, Andrea ; Volpe, Gualtiero: Toward real-time multimodal processing: EyesWeb 4.0. In: *AISB 2004: Proceedings of Artificial Intelligence and Simulation of Behaviour 2004 Convention: Motion, Emotion and Cognition*, Society for the Study of Artificial Intelligence and the Simulation of Behaviour (SSAISB), 2004, S. 22–26

[Crowley *et al.* 2000] Crowley, James L. ; Coutaz, Joëlle ; Bérard, François: Perceptual user interfaces: things that see. In: *Communications of the ACM* 43 (2000), Nr. 3, S. 54–ff.

[Darrell *et al.* 2000] Darrell, Trevor ; Gordon, Gaile G. ; Harville, Michael ; Woodfill, John: Integrated Person Tracking Using Stereo, Color, and Pattern Detection. In: *International Journal of Computer Vision* 37 (2000), Nr. 2, S. 175–185

[Davis und Bobick 1998] Davis, James W. ; Bobick, Aaron F.: A Robust Human-Silhouette Extraction Technique for Interactive Virtual Environments. In: *CAPTECH 1998: Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*. London, UK : Springer-Verlag, 1998, S. 12–25

[D'Hooge und Goldsmith 2001] D'Hooge, Herman ; Goldsmith, Michael: Game design principles for the Intel Play Me2Cam* virtual game system. In: *Intel Technology Journal* Q4 (2001)

[Douxchamps 2007] Douxchamps, Damien: *libdc1394*. 2007. – URL `http://damien.douxchamps.net/ieee1394/libdc1394/index.php`

[Eisenstein und Mackay 2006] Eisenstein, Jacob ; Mackay, Wendy E.: Interacting with communication appliances: an evaluation of two computer vision-based selection techniques. In: *CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM Press, 2006, S. 1111–1114

[Freeman *et al.* 2000] Freeman, William T. ; Beardsley, Paul A. ; Kage, Hiroshi ; Tanaka, Ken-Ichi ; Kyuma, Kazuo ; Weissman, Craig D.: Computer vision for computer interaction. In: *SIGGRAPH Computer Graphics* 33 (2000), Nr. 4, S. 65–68

[Griesser *et al.* 2005] Griesser, Andreas ; De Roeck, Stefaan ; Neubeck, Alexander ; Van Gool, Luc: GPU-Based Foreground-Background Segmentation using an Extended Colinearity Criterion. In: Greiner, G. (Hrsg.) ; Hornegger, J. (Hrsg.) ; Niemann, H. (Hrsg.) ; Stamminger, M. (Hrsg.): *VMV 2005: Proceedings of Vision, Modeling, and Visualization 2005*, IOS Press, November 2005, S. 319–326

[Hämäläinen 2002] Hämäläinen, Perttu: *QuiQui's Giant Bounce*, Medialab in the University of Art and Design Helsinki, UIAH, Master's Thesis, 2002

[Hämäläinen *et al.* 2003] Hämäläinen, Perttu ; Höysniemi, Johanna ; Rouvi, Teppo ; Turkki, Laura: *QuiQui's Giant Bounce*. 2003. – URL `http://www.cs.uta.fi/kukakumma/htmls_en/index.html`

[Hämäläinen *et al.* 2005] Hämäläinen, Perttu ; Ilmonen, Tommi ; Höysniemi, Johanna ; Lindholm, Mikko ; Nykänen, Ari: Martial arts in artificial reality. In: *CHI 2005: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 2005, S. 781–790

[Heikkilä und Silvén 1997] Heikkilä, Janne ; Silvén, Olli: A Four-step Camera Calibration Procedure with Implicit Image Correction. In: *CVPR 1997: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA : IEEE Computer Society, 1997, S. 1106

[Höysniemi *et al.* 2005] Höysniemi, Johanna ; Hämäläinen, Perttu ; Turkki, Laura ; Rouvi, Teppo: Children's intuitive gestures in vision-based action games. In: *Commununications of the ACM* 48 (2005), Nr. 1, S. 44–50

[Hu *et al.* 2001] Hu, Jianying ; Kashi, Ramanujan ; Lopresti, Daniel ; Nagy, George ; Wilfong, Gordon: Why Table Ground-Truthing is Hard. In: *ICDAR 2001: Proceedings of the Sixth International Conference on Document Analysis and Recognition*. Washington, DC, USA : IEEE Computer Society, 2001, S. 129

[Intel 2006] Intel: *Open Source Computer Vision Library*. 2006. – URL `http://www.intel.com/technology/computing/opencv/index.htm`

[ITU 1995] ITU: *Recommendation ITU-R BT.601-5: Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios*. 1995

[Ivanov *et al.* 2000] Ivanov, Yuri ; Bobick, Aaron ; Liu, John: Fast Lighting Independent Background Subtraction. In: *International Journal of Computer Vision* 37 (2000), Nr. 2, S. 199–207

[Jain 1981] Jain, Ramesh C.: Extraction of Motion Information from Peripheral Processes. In: *IEEE Transactions on Pattern Analysis and machine Intelligence* 3 (1981), September, Nr. 5, S. 489–503

[Juang und Majumder 2007] Juang, Ray ; Majumder, Aditi: Photometric Self-Calibration of a Projector-Camera System. In: *CVPR 2007: IEEE Conference on Computer Vision and Pattern Recognition*, 2007

[Kamkar-Parsi *et al.* 2005] Kamkar-Parsi, A. Homayoun ; Laganière, Robert ; Bouchard, Martin: A multi-criteria model for robust foreground extraction. In: *VSSN 2005: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*. New York, NY, USA : ACM Press, 2005, S. 67–70

[Kanade *et al.* 1996] Kanade, Takeo ; Tanaka, Masaya ; Oda, Kazuo ; Yoshida, Atsushi ; Kano, Hiroshi: A Video-Rate Stereo Machine and Its New Applications. In: *27th International Symposium on Industrial Robots*, October 1996, S. 671–676

[Konolige 1997] Konolige, Kurt: Small vision system. hardware and implementation. In: *Proceedings of the International Symposium on Robotics Research*, 1997, S. 111–116

[Krueger 1983] Krueger, Myron W.: *Artificial Reality*. Addison-Wesley, 1983. – 312 ff. S

[Krueger *et al.* 1985] Krueger, Myron W. ; Gionfriddo, Thomas ; Hinrichsen, Katrin: VIDEOPLACE–An Artificial Reality. In: *CHI 1985: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 1985, S. 35–40

[Lantinga 2007] Lantinga, Sam: *Simple Directmedia Layer*. 2007. – URL `http://www.libsdl.org/`

[London und Reiser 1978] London, Thomas B. ; Reiser, John F. ; Ritchie, Dennis M. (Hrsg.): A UNIX^{TM} Operating System for the DEC VAX-11/780 Computer / Bell Labs. URL `http://cm.bell-labs.com/cm/cs/who/dmr/otherports/32v.pdf`, 1978. – Memo

[Mester *et al.* 2001] Mester, Rudolf ; Aach, Til ; Dümbgen, Lutz: Illumination-Invariant Change Detection Using a Statistical Colinearity Criterion. In: *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*. London, UK : Springer-Verlag, 2001, S. 170–177

[Porikli 2003] Porikli, Fatih M.: Inter-camera color calibration by correlation model function. In: *ICIP 2003: International Conference on Image Processing* Bd. 2, 2003, S. 133–136

[Radke *et al.* 2005] Radke, Richard J. ; Andra, Srinivas ; Al-Kofahi, Omar ; Roysam, Badrinath: Image change detection algorithms: a systematic survey. In: *IEEE Transactions on Image Processing* 14 (2005), Nr. 3, S. 294–307

[Rakkolainen 2006] Rakkolainen, Ismo: Tracking users through a projection screen. In: *MULTIMEDIA 2006: Proceedings of the 14th annual ACM international conference on Multimedia*. New York, NY, USA : ACM Press, 2006, S. 101–104

[Smith 1978] Smith, Alvy R.: Color gamut transform pairs. In: *SIGGRAPH 1978: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM Press, 1978, S. 12–19

[Sony 2003] Sony: *EyeToy*. 2003. – URL `http://www.eyetoy.com`

[Sony 2006] Sony: *Pressemitteilung: Sony Computer Entertainment Europe gibt neuen Preis für die PlayStation 2 bekannt.* 2006. – URL `http://www.playstation-presse.de`

[Sparacino *et al.* 1997] Sparacino, Flavia ; Pentland, Alex ; Davenport, Glorianna ; Hlavac, Michal ; Obelnicki, Mary: City of News. In: *Ars Electronica 1997 Exhibition Catalogue*, 1997

[Stauffer und Grimson 1999] Stauffer, Chris ; Grimson, W. Eric L.: Adaptive background mixture models for real-time tracking. In: *CVPR 1999: IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 2, 23-25 June 1999, S. 246–252

[Sutherland 1968] Sutherland, Ivan E.: A head-mounted three dimensional display. In: *AFIPS/FJCC 1968: Fall Joint Computer Conference, American Federation of Information Processing Societies Conference Proceedings* Bd. 33, 1968, S. 757–764

[Wellner 1991] Wellner, Pierre: The DigitalDesk calculator: tangible manipulation on a desk top display. In: *UIST 1991: Proceedings of the 4th annual ACM symposium on User interface software and technology.* New York, NY, USA : ACM Press, 1991, S. 27–33

[Withagen *et al.* 2003] Withagen, Paul J. ; Groen, Frans C. A. ; Schutte, Klamer: EMswitch: A Multi-Hypothesis Approach to EM Background Modelling. In: *ACIVIS 2003: Proceedings of the IEEE Advanced Concepts for Intelligent Vision Systems Conference*, 2003, S. 199–206

[Wren und Ivanov 2001] Wren, Christopher R. ; Ivanov, Yuri: A Fast Algorithm for Depth Segmentation. In: *CVPR 2001: Conference on Computer Vision and Pattern Recognition, Technical Sketches*, 2001

[Wren *et al.* 1998] Wren, Cristopher R. ; Sparacino, Flavia ; Azarbayejani, Ali J. ; Darrell, Trevor J. ; Davis, James W. ; Starner, Thad E. ; Kotani, Akira ; Chao, Chloe M. ; Hlavac, Michal ; Russell, Kenneth B. ; Bobick, Aaron ; Pentland, Alex P.: Perceptive Spaces for Performance and Entertainment (Revised). In: *ATR Workshop on Virtual Communication Environments: Bridges over Art/Kansei and VR Technologies.* Kyoto, Japan, April 1998

[Yang und Levine 1992] Yang, Yee-Hong ; Levine, Martin D.: The background primal sketch: an approach for tracking moving objects. In: *Machine Vision and Application* 5 (1992), Nr. 1, S. 17–34

[Zhang 2000] Zhang, Zhengyou: A Flexible New Technique for Camera Calibration. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), Nr. 11, S. 1330–1334