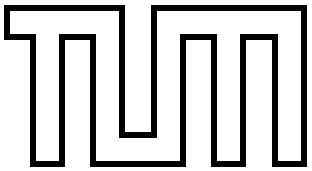


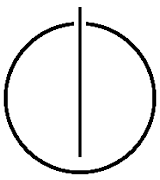
FAKULTÄT FÜR INFORMATIK

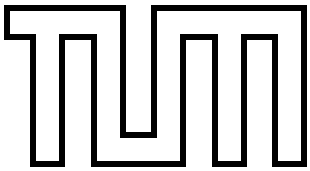
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Tracking für Augmented Reality Anwendungen auf Mobiltelefonen

Frieder Pankratz





FAKULTÄT FÜR INFORMATIK

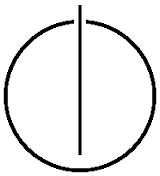
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Tracking für Augmented Reality Anwendungen auf Mobiltelefonen

Tracking for augmented reality applications on mobile phones

Author:	Frieder Pankratz
Aufgabenstellerin:	Prof. Gudrun Klinker, PhD.
Betreuer:	Björn Schwerdtfeger
Datum:	Juli 15, 2009



Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Juli 2009 Frieder Pankratz

Danksagung

Ich danke Björn Schwerdtfeger für sein Engagement und seine ausgezeichnete Betreuung während der Arbeit an dieser Master's Thesis.

Peter Keiter danke ich für seine wertvollen Tips. Weiter möchte ich Daniel Pustka und Wolf Rödiger danken für die Codebasis des Markertrackers, der Kameraansteuerung und der AR-Anwendung auf dessen Basis diese Arbeit beruht. Zudem möchte ich Vodafone für ihre Unterstützung während dieser Arbeit danken.

Zusammenfassung

Mobiltelefone bieten immer mehr Grafik- und Rechenleistung und haben damit das Potential einer nahezu idealen AR-Plattform. Sie sind weit verbreitet und die Benutzer sind mit ihrer Handhabung vertraut. Daher werden sie als Plattform für Augmented Reality (AR) Anwendungen immer interessanter. Sie bieten zwar nur einen Bruchteil der Leistung eines Desktop-PCs, aber dennoch genügend, um verschiedene Trackingverfahren durchzuführen. Die Trackingverfahren aus dem Desktopbereich können allerdings nicht einfach übernommen werden. Es muss darauf geachtet werden, dass die eingesetzten Techniken unter anderem rechen- und speichereffizient sind. In dieser Master's Thesis werden verschiedene Techniken zum Erkennen und Verfolgen von Objekten oder Kamerabewegungen in allen sechs Freiheitsgraden für AR-Anwendungen auf ihre Tauglichkeit für Mobiltelefone untersucht. Neben anderen technischen Einschränkungen mussten auch die Anforderungen, die die Benutzer an das Trackingsystem stellen berücksichtigt werden (Robustheit, Geschwindigkeit). Es wurden (kleine) Benutzerstudien durchgeführt um das Verhalten und die Bedürfnisse der Benutzer bei der Verwendung einer AR-Anwendung auf einem Mobiltelefon zu untersuchen. Um diese Aspekte zu berücksichtigen, wurde ein Markertrackingsystem so erweitert, dass der Marker nicht immer sichtbar sein muss. Wird der Marker verloren, erfolgt eine Approximation der Bewegung des Mobiltelefons über die Verschiebung des Kamerabildes in horizontaler und vertikaler Richtung. Die neue Position des Markers wird anschließend über die Bewegung des Mobiltelefons und der letzten bekannten Position des Markers bestimmt. Dadurch soll der Benutzer eine höhere Bewegungsfreiheit erhalten, um mit der AR-Anwendung auf natürlichere Weise interagieren zu können als bei einem herkömmlichen Markertrackingsystem.

Abstract

Mobile phones offer more and more gaphic and computing power and have the potential to become a nearly ideal AR platform. They are widespread and most people are comfortable using them. That's why mobile phones are getting more interesting as a platform for Augmented Reality(AR) applications. Even through they offer only a fraction of the processing power of a desktop pc, they offer sufficient processing power for different tracking

techniques. The tracking techniques used for desktop computing can't simply be applied to mobile phones. The used techniques must be, among other things, computing and memory efficient. In this master's thesis different techniques for detecting and tracking objects or camera movement in all six degrees of freedom for AR applications are presented. They are evaluated for their use on mobile phones. Among other technical restrictions, the user requirements for the AR system had to be considered (robustness, speed). Small user studies have been performed to observe the behavior and needs of the users, while using a mobile AR application. In respect to these requirements, a marker tracking system was extended, to allow the loss of the marker. If the marker gets lost, the movements of the mobile phone are approximated, using the horizontal and vertical movement of the camera image. The new marker pose is estimated using the movements of the mobile phone and the last known marker pose. Thereby the user should gain more mobility, which should allow him to interact with the AR application more naturally, compared to a standard marker tracking system.

Inhaltsverzeichnis

Teil I.

Einführung

1. Umfang der Arbeit

In diesem Abschnitt wird der Umfang der Arbeit beschrieben. Das erste Kapitel beinhaltet eine kurze Einführung in das Thema. In Kapitel 1.2 werden die Hintergründe und Motivation für diese Arbeit beschrieben. Kapitel 1.3 beschreibt den Aufbau der restlichen Arbeit.

1.1. Einführung

Bei AR handelt es sich um eine Mensch-Maschine Schnittstelle. Dabei wird die Realität durch digitale Informationen angereichert. Anders als bei der Virtuellen Realität, in der sich der Benutzer durch eine komplett künstliche Welt bewegt, werden bei AR virtuelle Objekte lagerichtig in die reale Umgebung eingefügt. Wichtig hierbei ist, dass der Benutzer mit den Objekten in Echtzeit interagieren kann. Auf der Website des Lehrstuhls für Informatikanwendungen in der Medizin & Augmented Reality der Technischen Universität München können etliche Beispiele betrachtet werden[1].

Es gibt eine Vielzahl an verschiedenen Einsatzmöglichkeiten für AR. Zum Beispiel kann AR dazu dienen, zusätzliche Information zu realen Objekten zu liefern. Klinker et al. [13] beschreiben eine AR-Anwendung, die mittels einer Webcam ein Sudoku-Spiel erkennt und dem Spieler zusätzliche Informationen liefert, um das Rätsel zu lösen. Eine andere Einsatzmöglichkeit ist das Behandeln von Phobien, wie in [10] beschrieben. Hierbei können die Patienten, die z.B. an einer Phobie vor Küchenschaben leiden, mit virtuellen Küchenschaben interagieren, um ihre Ängste zu Überwinden. AR kann auch zur wissenschaftlichen[18] oder medizinischen [23] Visualisierung dienen.

Um AR einsetzen zu können, muss das Trackingsystem die Position von realen Objekten erkennen. Mit Hilfe dieser Positionen können dann virtuellen Objekte in die reale Umgebung eingefügt werden. Anschließend müssen die virtuellen Objekte dem Benutzer auf passende Weise präsentiert werden. Hierfür gibt es eine Vielzahl an verschiedenen Trackingverfahren und Darstellungsmöglichkeiten. Auf die Trackingverfahren wird in Kapitel 2.2 näher eingegangen. Für die Darstellung werden in der Regel Bildschirme, Leinwände oder Head Mounted Display (HMD) verwendet.

In den letzten Jahren hat sich die Rechen- und Grafikleistung von Mobiltelefonen erheblich gesteigert. Zudem besitzt nahezu jedes aktuelle Mobiltelefon eine integrierte Kamera. Daher werden die Mobiltelefone immer interessanter als Plattformen für Augmented Reality, denn alle für AR nötigen Schritte können auf ihm ausgeführt werden. Ein weiterer Grund für ihre Attraktivität als Plattform, ist ihre weite Verbreitung und der sichere Umgang der Benutzer mit den Geräten. Es besteht keine Notwendigkeit für weitere Hardware. Dies ermöglicht es, AR-Anwendungen für einen sehr breiten Markt zu entwickeln, was vorher so nicht möglich war.

[1] http://campar.in.tum.de/WebHome, letzter Zugriff 15.07.2009

Die integrierte Kamera eines Mobiltelefons dient als Videoquelle, sein Prozessor der Verarbeitung und das Display der Ausgabe. Ein solches AR-System wird als Video-See-Through Display bezeichnet. Video-See-Through bedeutet, das der Benutzer die Realität nur indirekt über ein Video auf einem undurchsichtigen Display wahrnimmt. Bevor das Videobild im Display dargestellt wird, werden die virtuellen Objekten in das Videobild gezeichnet. Im Gegensatz dazu gibt es noch Optical-See-Through, bei dem der Benutzer die Realität direkt sieht. Die virtuellen Objekte werden mittels optischer Verfahren, z.B. durch ein HMD mit halbdurchsichtigem Display dargestellt. Damit Überlagern sich Realität und virtuelle Objekte direkt.

AR ist seit mehreren Jahrzehnten ein intensives Forschungsgebiet und bietet eine Vielzahl an Verfahren für den Desktopbereich. Jedoch können die meisten der Verfahren nicht direkt auf Mobiltelefone übertragen werden, da Mobiltelefone unter anderem nur über einen Bruchteil der Rechenleistung eines durchschnittlichen Desktop-PCs verfügen.

In dieser Arbeit soll untersucht werden welche Anforderungen an ein Trackingsystem für Mobiltelefone gestellt werden und welche Verfahren dazu geeignet sind, diese Anforderungen zu realisieren.

1.2. Hintergrund und Motivation

Diese Arbeit beruht auf einem Gemeinschaftsprojekt zwischen der Technischen Universität München, Fachgebiet Augmented Reality und der Vodafone Group R&D, an welchem ich maßgeblich beteiligt war. Ergebnis des Projekts war eine AR-Anwendung für Mobiltelefone, die mit Hilfe von ARToolKitPlus [27] die Position eines quadratischen Markers im aktuellen Kamerabild erkennt. Das Kamerabild wurde mit einem virtuellen Objekt auf der Position des Markers augmentiert und auf dem Display dargestellt. Die AR-Anwendung wird im weiteren als AR-Viewer bezeichnet. Ein Bild von ihr ist in Abbildung 1.1 zu sehen. Der AR-Viewer wird in Kapitel 4.1 näher beschrieben. Im Grunde konnte der Benutzer Objekte laden und auf der durch den Marker definierten Ebene verschieben und um ihre Z-Achse drehen. Es wurde eine informale explorative Benutzerstudie mit der AR-Anwendung durchgeführt. Dabei wurde das Verhalten der Benutzer im Umgang mit der AR-Anwendung und dem Mobiltelefon beobachtet. Die 5 Studienteilnehmer mussten Einrichtungsgegenstände(Tisch, Stuhl, Bild, Topfpflanze) im Raum platzieren, an realen Objekten ausrichten, auf eine bestimmte Position rotieren und die Objekte aus verschiedenen Blickwinkeln betrachten.

Dabei habe ich folgende Beobachtungen bezüglich des Trackingsystems gemacht. Den Studienteilnehmern wurden die Stärken und Schwächen des Trackingsystems vor Beginn der Studie nicht mitgeteilt, um das natürliche Verhalten der Benutzer beobachten zu können. Die meisten Benutzer sahen eine AR-Anwendung zum ersten Mal und hatten daher keine Vorkenntnisse mit optischem Marker-Tracking, was der Studie zugute kam. Die meisten Benutzer bewegten das Mobiltelefon am Anfang zu schnell, was zu verzerrten Bildern führte (mehr dazu in Kapitel 2.1.4), die das Marker-Tracking aussetzen ließen. Ein weiteres Problem bestand in dem kleinen Öffnungswinkel der integrierten Kamera. Wurden große Objekte wie Tische und Stuhle betrachtet, verdeckten die Objekte fast den gesamten Darstellungsbereich, wodurch die Benutzer den Bezug zur realen Welt verloren. War der Marker nah am Benutzer, musste er das Mobiltelefon bewegen um das ganze Ob-

jekt betrachten zu können. Dies führte zu einem Verlust der Markerposition, da der Marker aus dem Bild geschoben wurde. War der Marker weiter entfernt vom Benutzer, dann war der Öffnungswinkel kein Problem mehr. Allerdings begann dann, abhängig von Entfernung zum Marker, größe des Markers und Blickwinkel, die Pose des Markers zu zittern. Mehr dazu in Kapitel 3.1.3. Durch diese Einschränkungen wurde die AR-Anwendung für den Benutzer weniger intuitiv und frustrierte ihn teilweise. Dies führte zu der Schlussfolgerung, dass es dem Benutzer möglich sein muss Teile des Objekts zu betrachten, ohne den Marker dabei im Bild haben zu müssen. Des weiteren konnte beobachtet werden, dass die Benutzer meist vor dem Marker stehen bleiben. Betrachtet wurde das Objekt, indem das Mobiltelefon um die Achsen in Handgelenk, Ellenbogen und Hüfte rotiert wird. Die Achsen sind in Abbildung 3.12 illustriert. Das Mobiltelefon verändert dabei räumlich kaum seine Position. In den Interviews mit den Benutzern wurde außerdem festgestellt, dass eine stabile Markerposition für das zufriedenstellende Verwenden des AR-Viewers von hoher Bedeutung ist. Eine ungenaue Positionserkennung des Markers hingegen wurde von den Benutzern als nicht sonderlich störend empfunden.

Die Beobachtungen, die ich während der Benutzerstudie gemacht habe dienen als Grundlage für diese Arbeit. Unter der Annahme, das sich Benutzer bei betrachten eines Markers nicht wesentlich von der Stelle bewegen ist die Aufgabe dieser Arbeit die folgende.

Das Trackingsystem soll so erweitert werden, dass es nicht immer notwendig ist, den Marker im Videobild zu haben. Das Trackingsystem soll die Markerpose weiterverfolgen, wenn der Marker nicht erkannt werden kann. Es soll dem Benutzer möglich sein, Teile des virtuellen Objekts zu betrachten, selbst wenn die Sichtlinie der Kamera nicht auf den Marker zeigt. Die Beobachtungen über das Verhalten der Benutzer sollen in das neue Trackingsystem einfließen.

Es sollen bereits vorhandene Techniken eingesetzt werden, um das Marker-Tracking zu erweitern und zu stabilisieren. Zudem sollen für die Trackingverfahren und die Fusion möglichst wenig Ressourcen eingesetzt werden.

Abbildung 1.1.: AR-Viewer mit vituellen Modell einer Spardose

1.3. Überblick

In diesem Kapitel wird ein Überblick über die restlichen Kapitel dieser Arbeit gegeben. Als nächstes werden die Grundlagen und verwandte Arbeiten für diese Arbeit vorgestellt.

Kapitel 2.1 befasst sich zunächst mit den Möglichkeiten und Einschränkungen eines Mobiltelefons als AR-Plattform. In Kapitel 2.2 werden Verschiedene Trackingverfahren erläutert und auf ihre Tauglichkeit für Mobiltelefone bewertet. Zudem wird kurz auf verschiedene Systemarchitekturen und Möglichkeiten der Fusion verschiedener Trackingverfahren eingegangen. Abschließend wird in Kapitel 2.2.6 beschrieben, welche Trackingverfahren eingesetzt werden und warum.

Kapitel II befasst sich mit der Konstruktion und Evaluation des in dieser Arbeit entwickelten Trackingsystems. In Unterkapitel 3.1 wird die dynamische und statische Struktur des neuen Trackingsystems sowie Optimierungen beschrieben. Die Konfiguration des neuen Trackingsystems wird in Unterkapitel 3.2 behandelt. Als letztes Unterkapitel dieses Kapitels wird in 3.3 auf die Schnittstellen des neuen Trackingsystems und der Kameraansteuerung für C# eingegangen.

In Kapitel 4 wird das neue Trackingsystem evaluiert. Zunächst wird in 4.1 der AR-Viewer, auf dessen Basis die Evaluation durchgeführt wurde, genauer Vorgestellt. Neben der Benutzerstudie in 4.3 wurden auch Zeitmessungen durchgeführt, die in 4.2 zu sehen sind. Abschließend wird in Unterkapitel 4.4 eine Auswertung und Beurteilung der Ergebnisse der Benutzerstudie gegeben.

Als letztes wird in den Kapiteln 6 und 5 noch eine Zusammenfassung über diese Arbeit und ein Ausblick über mögliche Weiterentwicklungen gegeben.

2. Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die grundlegenden Eigenschaften von Mobiltelefonen und verschiedenen Trackingverfahren betrachtet. Dabei soll ein kurzer Überblick, über die Möglichkeiten und Einschränkungen, die die einzelnen Verfahren für den Einsatz auf Mobiltelefonen mit sich bringen, entstehen. Zunächst werden in Kapitel 2.1 die Eigenschaften der Mobiltelefone untersucht. Anschließend werden in Kapitel 2.2 die Tracking Verfahren untersucht und auf ihre Eignung bewertet. Zudem werden in dem Kapitel verschiedene Möglichkeiten der Systemarchitektur und der Fusion verschiedener Verfahren betrachtet.

2.1. Mobiltelefon

Neben grundlegenden Systemkomponenten wie CPU, Arbeitsspeicher, Display und Betriebssystem bieten Mobiltelefone, je nach Modell, verschiedene Möglichkeiten Informationen zu sammeln. Im folgenden werden die einzelnen Komponenten näher betrachtet.

2.1.1. CPU und Arbeitsspeicher

Die CPU ist eine der wichtigsten Komponenten in dem System, da von ihrer Leistungsfähigkeit im Wesentlichen die Leistung des gesamten AR-Systems abhängt. Da die wenigsten Mobiltelefone eine GPU besitzen (Nokia N95 ist eines der wenigen mit GPU), müssen alle Berechnungen für Tracking und Rendering auf dem selben Prozessor ausgeführt werden. Zudem ist der interne Bus langsam, welches zu langsamen Speicherzugriffen führt. Der Cache der CPU ist klein, was mehr Speicherzugriffen zur folge hat. Der verfügbare Arbeitsspeicher ist zudem stark begrenzt. Einige Mobiltelefone und ihre CPUs sind in Tabelle 2.2 zu sehen. Für vier aktuelle Geräte wurde ein Benchmark mit dem Programm SPB Benchmark durchgeführt. Da SPB Benchmark nur Windows Mobile Geräte unterstützt konnten nur Windows Mobile Geräte getestet werden. Die Ergebnisse des Benchmarks sind in Tabelle 2.1 zu sehen. Alle Indexwerte sind in Bezug auf das Compaq iPAQ 3650 normalisiert, mit 1000 als dem Standardwert. Eine Beschreibung der Tests, die für den Benchmark durchgeführt wurden, ist in [24] zu sehen.

Mobiltelefon	CPU Index	Grafik Index	Dateisystem Index
HTC Touch Diamond	2258	1664	190
Asus P552w	3559	10550	626
Asus P565	4110	2677	670
Dell Axim X51v	2044	616	142

Tabelle 2.1.: Sbp Benchmark verschiedener Mobiltelefone

Mobiltelefon	CPU	Display
HTC Touch Diamond	Qualcomm MSM7201A 528MHz	2.8 Zoll 480x640
HTC Touch Pro	Qualcomm MSM7201A 528MHz	2.8 Zoll 480x640
IPhone	Samsung S5L8900 400MHz	3.5 Zoll 320x480
GooglePhone	Qualcomm MSM7201A 528MHz	3.2 Zoll 320x480
Asus P552w	Marvell PXA930 624MHz	2.8 Zoll 480x640
Asus P565	Marvell PXA930 800MHz	2.8 Zoll 480x640
Dell Axim X51v	Intel XScale PXA270 624 MHz	3.8 Zoll 480x640

Tabelle 2.2.: CPU und Display ausgewählter Mobiltelefone und des Dell X51v

Ein entscheidender Punkt der die Entwicklung eines AR-Systems für Mobiltelefone erschwert, ist das Fehlen einer FPU. Floating-Point Berechnungen müssen entweder kostspielig simuliert oder durch Fixed-Point Berechnungen ersetzt werden. Eine Einführung in Fixed-Point Arithmetik ist in [28] zu finden. Fixed-Point Berechnungen sind zwar schnell genug um den Nachteil der fehlenden FPU auszugleichen, jedoch muss man bei ihrer Verwendung die Grenzen der Fixed-Point Zahlen berücksichtigen. Je nach dem, wo der "Dezimalpunkt" gesetzt wird, hat man entweder einen größeren gültigen Wertebereich mit geringer Genauigkeit, oder einen geringeren gültigen Wertebereich mit höherer Genauigkeit.

2.1.2. Display

Über das Display des Mobiltelefons wird dem Benutzer die augmentierte Szene gezeigt. Immer mehr moderne Mobiltelefone verwenden einen Touchscreen zur Steuerung, wie es z.B. vom IPhone bekannt ist. In Tabelle 2.2 sind die Displaygrößen und Auflösungen verschiedener Mobiltelefone mit Touchscreen-Steuerung zu sehen. Mittlerweile bieten die meisten Displays moderner Mobiltelefone eine Auflösung von 320x480 Pixeln oder größer. Diese Auflösung ist ausreichend, um auch detailreiche Objekte gut darstellen zu können. Meistens aber ist die Grafikleisung der Mobiltelefone zu gering, um detailreiche Objekte zu Rendern und Tracking zu betreiben. Wichtiger als die Auflösung des Displays ist daher seine Größe. In der Regel wird eine Displaygröße von 2.8 Zoll bei Touchscreen Mobiltelefonen nicht unterschritten. Auf Grund der Bauform der Mobiltelefone ist die Höhe der Displays meist größer als ihre Breite. Es wird zwar das bekannte 4:3 Sitenverhältnis beibehalten, jedoch sind die Seiten vertauscht.

2.1.3. Betriebssystem

Es gibt eine Vielzahl von verschiedenen Betriebssystemen für Mobiltelefone. Die Bandbreite reicht von proprietären Betriebsystemen der Mobiltelefonhersteller bis hin zu Open Source Betriebssystemen. Laut dem Marktforschungs- und Analyse- Unternehmen Gartner [6] war Nokia in zweiten Quartal 2008 mit ihrem Betriessystem Symbian OS Marktführer mit einem Marktanteil von 57,1%(18.405.057 verkaufe Smartphones). An zweiter stelle kommt RIM, die mit ihrer BlackBerry Serie bekannt wurden, mit einem Marktanteil von 17,7%(5.594.159). Microsoft Windows Mobile belegt den dritten Platz mit 12.0%(3.873.622)

Marktanteil. Anschließend kommen auf Linux basierte Betriebssysteme. Eines der Bekanntesten ist das Android Betriebssystem des Google Mobiltelefons G1. Zur Zeit ist der Marktanteil von Apple mit ihrem IPhone und dem Mac OS X Betriessystem noch sehr gering (2,3%), da das Betriebssystem nur mit den eigenen Mobiltelefonen ausgeliefert wird. Access[1] mit ihrem Betriessystem Garnet OS[2] haben nur einen Marktanteil von 1,1%. Durch die große Anzahl an Betriessystemen und Mobiltelefonmodellen haben sich kaum Standards durchsetzen können. Einer der wenigen Standards der von allen Betriessystemen unterstützt wird ist die Java ME Plattform, definiert in JSR 30 und 37, sowie Erweiterungen für diese Plattform. Da Java keine optimale Umgebung für die rechenintensiven Trackingalgorithmen bietet, werden diese meist in dedizierten Sprachen implementiert und müssen für das jeweilige Betriessystem portiert werden.

2.1.4. Kamera

Wie bereits in Kapitel 1.1 erwähnt, wird das Mobiltelefon im Video-See-through Modus verwendet werden. Nahe zu alle Mobiltelefone verfügen heutzutage über eine integrierte Kamera. Jedoch kann die Ansteuerung der Kamera aus einer Anwendung heraus Problematisch sein. Zum Beispiel gab es für das IPhone zum Zeitpunkt der Erstellung dieser Arbeit keine offizielle API, mit der die Kamera in einem Videomodus ansprechbar war. Es gab lediglich eine Komponente, mit der man einzelne Bilder aufnehmen konnte. Windows Mobile bietet hingegen mit DirectShow eine standardisierte Schnittstelle zur Ansteuerung der integrierten Kamera. Neben der Verwendung als Hintergrundbild kann noch eine Reihe optischer Trackingverfahren verwendet werden, sobald ein Videostream von Einzelbildern in ausreichender Qualität verfügbar ist. Werden die integrierten Kameras in einem Videomodus betrieben, erreichen sie bei weitem nicht die Auflösungen die bei Einzelbildern möglich wären. Jedoch ist für die meisten optischen Trackingverfahren eine Auflösung von 120x160 Pixeln ausreichend. Glücklicherweise haben die Bilder heutiger Kameras kaum eine radiale Verzerrung, wodurch man auf rechenintensive Entzerrungen des Bildes unter Umständen verzichten kann. Die beiden Geräte, die in dieser Arbeit benutzt wurden weisen geringe radiale Verzerrungen auf. In Abbildung 2.1 ist die radiale und tangiale Verzerrung des HTC Touch Diamond Mobiltelefons zu sehen. Im Vergleich dazu ist in Abbildung 2.2 die radiale und tangiale Verzerrung der Logitech QuickCam Pro 4000 Webcam zu sehen. Alle Angaben in den Bildern haben die Einheit Pixel. Die Isolinien beschreiben die Verschiebung der Pixel vom Bildmittelpunkt. Obwohl die Kamera des Mobiltelefons eine qualitativ schlechte Linse verwendet sind die Verzerrungen kaum größer als bei der Webcam. Dies liegt daran, dass die Bilder des Mobiltelefons bereits auf Hardwareebene entzerrt werden. Um den Bildmittelpunkt (Bildmitelpunkt = X, Principal Point = Kreis) sind die Verzerrungen sehr gering, erst in den Ecken des Bildes werden die Verzerrungen stärker.

Allerdings besitzen die integrierten Kameras auch einige Schwächen. So sind die Öffnungswinkel der Kameras relativ gering. Dies kann dazu führen, dass große virtuelle Objekte den ganzen Darstellungsbereich des Displays einnehmen und so der Zusammenhang zwischen virtueller und realer Welt für den Benutzer verloren geht. Zudem kommt es bei

[1] PalmSource wurde von Access übernommen
[2] neuer Name des Palm OS

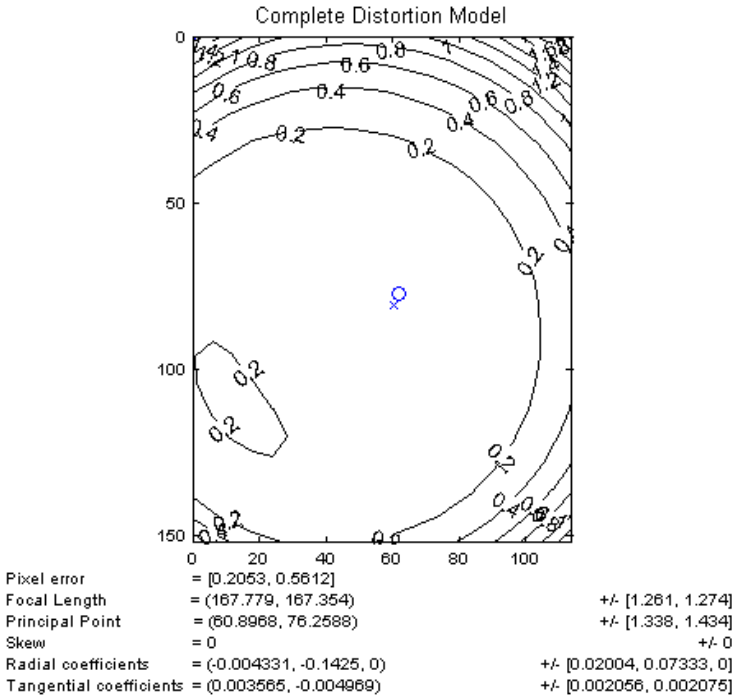


Abbildung 2.1.: Radiale und tangiale Verzerrung des HTC Touch Diamond bei 120x160 Pixeln

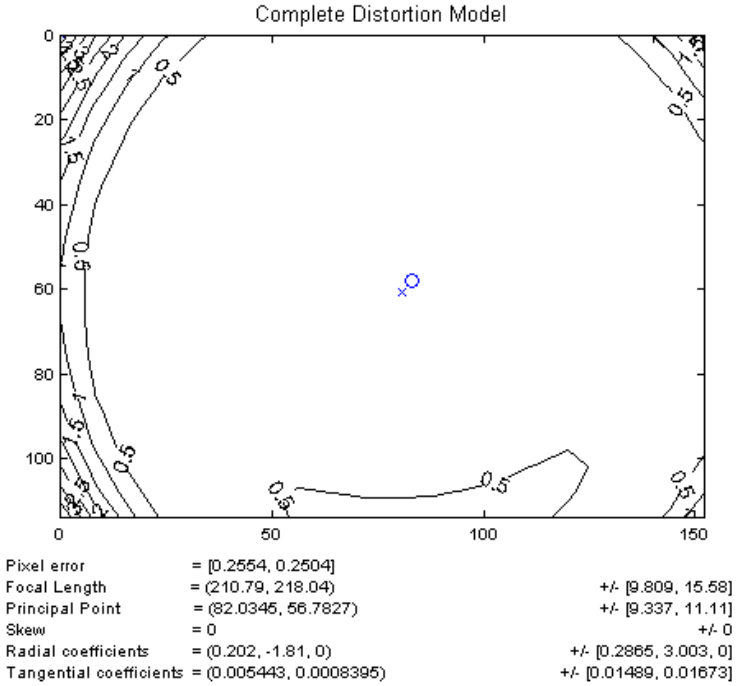


Abbildung 2.2.: Radiale und tangiale Verzerrung der Logitech QuickCam Pro 4000 bei 160x120 Pixeln

schnellen Bewegungen zu einer starken Verzerrung in den einzelnen Bildern, was eine zuverlässige Auswertung durch optische Trackingverfahren erschwert. Dieser Effekt wird teilweise dadurch ausgelöst, dass die Kamera keine Blende hat. Dadurch wird die Sensorfläche die ganze Zeit belichtet. Durch die langen Belichtungszeiten tritt der Effekt der Bewegungsunschärfe (motion blur) ein. Des weiteren tritt der Rolling-Shutter-Effekt ein. Bei den eingesetzten CMOS Sensoren werden die Bilddaten zeilen- bzw. spaltenweise abgetastet. Dadurch werden Linien die senkrecht zur Abtastrichtung stehen (z.B. bei einem Kameraschwenk) als Treppenlinien dargestellt.

Die Kamera und das Display sind kompakt in dem Mobiltelefon verbaut und damit räumlich nicht weit voneinander entfernt. Bei einem Mobiltelefon kann der Offset ignoriert werden, weil das Kamerabild verwendet wird, um die reale Welt im Display darzustellen. Der Offset der dadurch zwischen realer Welt und der Darstellung im Display entsteht, wird nicht bemerkt und beeinflusst das Gefühl der Immersion [19] nicht.

2.1.5. Inertiale Sensoren und Kompass

In modernen Mobiltelefone sind Inertialsensoren verbaut. Bei den Inertialsensoren, die zur Zeit hauptsächlich in Mobiltelefonen verbaut werden, handelt es sich um Beschleunigungssensoren, die die Beschleunigung in einer Achse messen können. Von diesen Sensoren sind in der Regel drei in einem Mobiltelefon verbaut, wie in Abbildung 2.3 zu sehen ist.

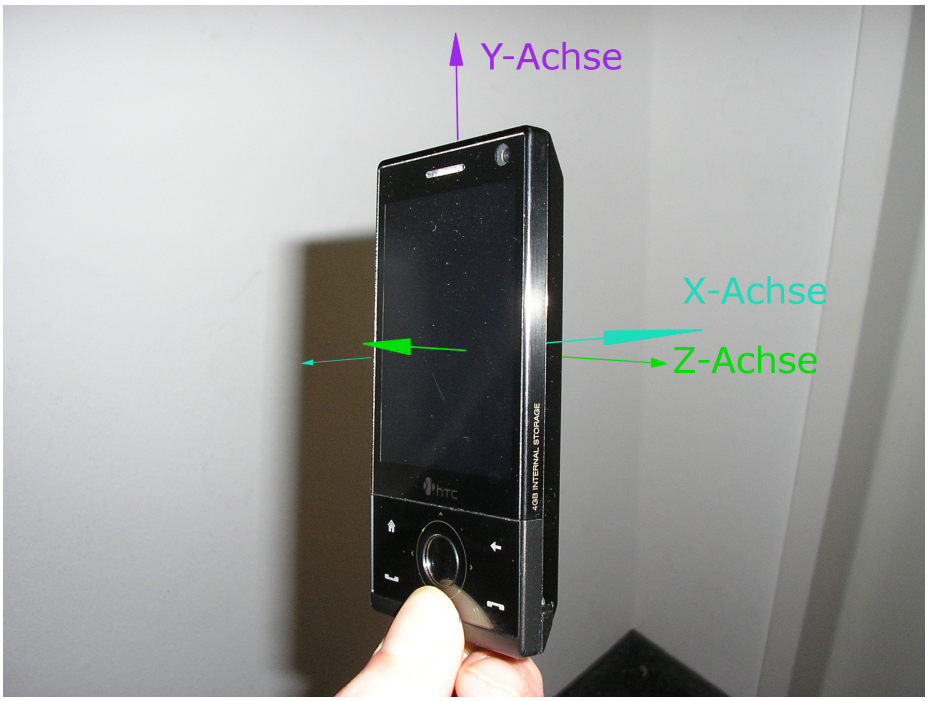


Abbildung 2.3.: Achsen der Beschleunigungssensor des HTC Touch Diamond

Indem die Erdbeschleunigung gemessen wird, können verschiedene Orientierung und Neigungen des Mobiltelefons gemessen werden. Wird das Mobiltelefon aufrecht gehalten,

kann die Rotation des Mobiltelefons um die Z-Achse über die Sensoren der X- und Y-Achse gemessen werden. Der Sensor der Z-Achse kann in diesem Fall dafür benutzt werden, die Neigung des Mobiltelefons zu bestimmen (Rotation um X-Achse). Wird das Mobiltelefon so gehalten, dass das Display nach oben oder unten zeigt, können die Sensoren der X- und Y-Achse dazu verwendet werden, die Neigung des Mobiltelefons in diesen Richtungen zu bestimmen. Diese Funktionalität wird z.B. bei dem von HTC entwickelten Spiel Teeter (Abbildung 2.4) als Eingabemethode verwendet. Durch neigen des Mobiltelefons wird die virtuelle Spielfläche geneigt und der Ball fängt an zu rollen.

Beschleunigungssensoren bieten den Vorteil, dass kaum Rechenzeit für das Erstellen und Auslesen der Beschleunigungswerte benötigt wird. Zudem liefern sie gute Werte über die Neigung des Mobiltelefons, abhängig von der derzeitigen Orientierung. Jedoch eignen sie sich nicht so gut dafür die typischen Bewegungen des Mobiltelefons zu verfolgen, wenn der Benutzer das Mobiltelefon bewegt. Hält der Benutzer z.B. das Mobiltelefon aufrecht vor sich und schwenkt es nach links oder rechts, dann sind die Bewegungen über die Beschleunigungssensoren nicht nachzuvollziehen. Am Anfang der Bewegung wird das Mobiltelefon zwar beschleunigt, jedoch nur für sehr kurze Zeit. Der Großteil der Bewegung wird mit mehr oder weniger konstanter Geschwindigkeit ausgeführt. Die Beschleunigung sollte durch den Sensor der X-Achse messbar sein. Jedoch ist die Beschleunigung nur sehr kurz und schwach im Vergleich zur Erdbeschleunigung und wird daher durch Rauschen der Sensordaten unterdrückt. Erschwerend kommt hinzu, dass die Bewegung vom Benutzer nie Perfekt ausgeführt werden kann. Beim schwenken wird das Mobiltelefon immer etwas um die Z-Achse rotiert. Die Erbeschleunigung die auf den Sensor der X-Achse wirkt, kann dann fälschlicher Weise als Beschleunigung der Bewegung interpretiert werden. Um aus der Beschleunigung die Positionsveränderung bestimmen zu können, muss die Beschleunigung doppelt integriert werden. Durch die ungenauen Beschleunigungswerte ergibt sich durch die doppelte Integration ein starker Drift der Position.

Gyroskope würden sich besser eignen, um die Rotationen des Mobiltelefons zu verfolgen. Die Benutzerinteraktion, die den größten Einfluss auf die Augmentierung hat, ist die Rotation des Mobiltelefons. Bisher waren ihre Produktionskosten zu hoch um sie im Massenmarkt einzusetzen. Dies könnte sich in nächster Zeit ändern [17], was weitere Möglichkeiten zum Verfolgen der Bewegungen eröffnet.

Neuerdings werden Mobiltelefone auch mit Kompassen ausgestattet. Eines der ersten Mobiltelefone mit Kompass war das Google G1 Mobiltelefon. Das Apple IPhone soll in einer neueren Revision ebenfalls einen Kompass erhalten. Ein Kompass würde die Möglichkeit bieten, jederzeit die Richtung in die das Mobiltelefon gehalten wird, zu ermitteln.

2.1.6. GPS

Einige Mobiltelefone besitzen mittlerweile einen integrierten GPS Empfänger. Dadurch ist es Möglich die Position des Mobiltelefons im Rahmen der Genauigkeit von GPS zu bestimmen. Obwohl man damit seine Position kennt, reichen diese Informationen für AR nicht aus. Kombiniert man aber GPS mit einem Kompass, kann die aktuelle Position und Blickrichtung ermittelt werden. Diese Informationen würden z.B. ausreichen um dem Be-

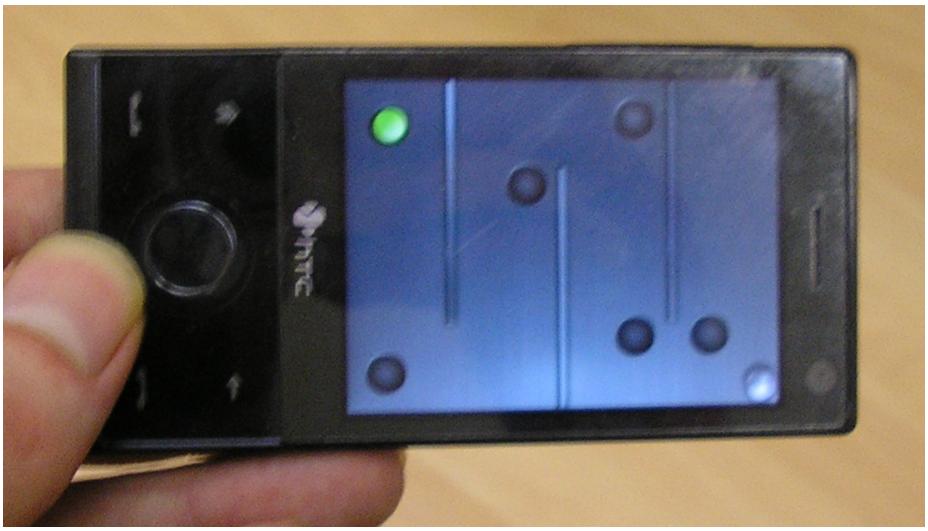


Abbildung 2.4.: HTC Teeter Spiel

nutzer ortsbezogene Informationen zu präsentieren. Beispielsweise die Namen der Berge, die man gerade sieht. GPS eignet sich gut um die Position in großen Arealen zu bestimmen. Bewegt sich der Benutzer allerdings nur ein paar Meter um seinen Ursprungsort, dann ist GPS zu ungenau, um sinnvoll eingesetzt werden zu können.

2.1.7. Datenverbindungen

Heutige Mobiltelefone bieten verschiedene Möglichkeiten, Daten mit anderen Rechnern auszutauschen. Neben GPRS und UMTS Datenverbindungen zu weit entfernten Rechnern, bieten immer mehr Mobiltelefone die Möglichkeit über Bluetooth und Wireless Lan Verbindungen zu lokalen Netzwerken herzustellen. Diese Verbindungen können neben dem Datentransfer auch für Trackingverfahren verwendet werden. Wenn mehrere WLan- oder Bluetooth-Sender in der nähe sind und ihre Positionen bekannt sind, kann aufgrund der empfangenen Signalstärke die Position mittels triangulation bestimmt werden. Problematisch ist hierbei, dass wie bei GPS nur eine Position mit einer gewissen ungenauigkeit ermittelt werden kann. Soll eine AR-Anwendung, die auf diesem Trackingverfahren beruht, in ständig wechselnden Umgebungen eingesetzt werden, müssen immer wieder neue Sender platziert werden und das Trackingsystem auf ihre neuen Positionen kalibriert werden.

2.1.8. Fazit

Durch die Beschränkungen in Sachen Leistung und Peripherie ist die Entwicklung einer AR-Anwendung für Mobiltelefone sehr herausfordernd. Zur Zeit haben GPUs noch nicht den einzug in Smartphones gefunden, daher müssen alle Berechnungen und Renderaufgaben auf der CPU durchgeführt werden. Da Tracking und Rendering auf der CPU laufen, muss während der Entwicklung stark darauf geachtet werden, dass Tracking so performant wie möglich zu gestalten. Dadurch soll der Anwendung genug Zeit übrig bleiben,

um ihre Aufgaben zu erfüllen und die Bildwiederholrate hoch zu halten. In den Dissertationen von Wagner [3] und Henrysson [8] ist zu sehen, dass Markertracking auf Mobiltelefonen und PDAs möglich ist.

Was Mobiltelefone als AR-Plattform so interessant macht, ist ihre kompakte Bauweise. Anders als bei AR-Systemen, die PCs verwenden, hat man volle Bewegungsfreiheit und ist nicht an einen Ort gebunden. Selbst AR-Systeme, die mit Laptops und HMDs arbeiten, sind um ein vielfaches sperriger. Ein Mobiltelefon vereint alle nötigen Komponenten für ein AR-System und kann dabei mit einer Hand gehalten und bedient werden. Zudem sind Mobiltelefone und ihre Bedienung vielen Leuten bekannt, wohingegen AR den meisten Leuten unbekannt ist. Die Hemmschwelle, eine AR-Anwendung zu verwenden, sollte sinken, da die Benutzer bereits mit der Hardware vertraut sind. AR-Anwendungen für Mobiltelefone könnten dazu verwendet werden, die bekanntheit von AR in der Bevölkerung zu steigern und die Konzepte von AR näher zu bringen.

Als Testplattform für die weitere Entwicklung des Trackingsystems fiel die Wahl auf das HTC Touch Diamond Mobiltelefon. Es besitzt einen 528 MHz Qualcomm MSM7201A CPU und 192 MB RAM. Zum Kaufzeitpunkt war HTC Touch Diamond eines der rechenstärksten Mobiltelefone. Das 2.8 Zoll Touchscreen-Display mit einer Auflösung von 480x640 Pixeln bietet genügend Platz zum Darstellen der Objekte. Zudem bietet das Mobiltelefon eine 3.1 MegaPixel Kamera, GPS-, Bluetooth- und WLan-Empfänger, sowie integrierte Inertialsensoren. Durch das eingesetzte Windows Mobile Betriebssystem, kann die DirectShow Schnittstelle verwendet werden, um die Kamera anzusteuern. Die Kamera kann in einem 15 FPS Videomodus angesprochen werden. Das Video hat dabei eine Auflösung von 240x320 Pixeln und ist im RGB565 Format[3] kodiert.

2.2. Trackingverfahren und AR-Systeme

Jedes AR-System benötigt ein Trackingverfahren, um die Position des Benutzers bzw. von Objekten zu ermitteln. Grundsätzlich werden dafür magnetische, mechanische, akustische, inertiale und optische Trackingverfahren sowie Kombinationen dieser Verfahren eingesetzt. In Kapitel 2.2.1 werden kurz die Trackingverfahren präsentiert, die prinzipiell nicht für das hier angestrebte Trackingsystem in Frage kommen. Anschließend werden in Kapitel 2.2.2 optische Tracking-Verfahren präsentiert. Kapitel 2.2.3 befasst sich mit inertialem Tracking. Neben den Tracking-Verfahren werden in Kapitel 2.2.4 verschiedene Möglichkeiten der Systemarchitektur und in Kapitel 2.2.5 die Fusion verschiedener Tracking-Verfahren besprochen. Abschließend werden in Kapitel 2.2.6 die einzelnen Tracking-Verfahren bewertet und eine Auswahl der Verfahren für das neue Trackingsystem getroffen.

2.2.1. Nicht geeignete Trackingverfahren

Die Trackingverfahren, die eingesetzt werden sollen, sollen bei jedem Benutzer sofort ohne großen Aufwand eingesetzt werden können. Daher fallen Verfahren wie magnetisches, mechanisches und akustisches Tracking aus, da hierfür zusätzliche Hardware benötigt wird. Zudem besitzen diese Verfahren noch andere Nachteile, die ihren Einsatz nicht sinnvoll erscheinen lassen.

[3] 16 Bit pro Pixel, 5 Bit für Rot und Blau, 6 Bit Grün

Magnetisches Tracking Für magnetisches Tracking müssen Magnetfelder an festen Positionen erzeugt werden. Magnetfeldsensoren werden auf das zu verfolgende Objekt angebracht und auf die Magnetfelder kalibriert. Wird das Objekt nun Bewegt, verändert sich die Magnetfeldstärke an den Sensoren. Über Triangulation kann die Position des Objekts errechnet werden. Von Nachteil ist hierbei, dass sich keine metallischen Gegenstände im Magnetfeld befinden dürfen, da ansonsten die Magnetfelder gestört würden und so das Tracking nicht mehr funktionieren kann.

Mechanisches Tracking Beim mechanischem Tracking wird das zu verfolgende Objekt über mechanische Glieder mit einem Referenzpunkt verbunden. Zwischen den einzelnen Gliedern sind Sensoren angebracht, die den Winkel zwischen den Gliedern bestimmen. Mit Kenntnis des Referenzpunktes kann so die Position des Objekts bestimmt werden. Dieses Tracking-Verfahren bietet zwar eine hohe Präzision, jedoch ist der Aktionsradius beschränkt und die Aufbauten sind meistens sperrig.

Akustisches Tracking Im Prinzip ähneln sich akustisches und mechanisches Tracking. Anstelle der Magnetfeldstärke wird die Laufzeit von Ultraschall-Signalen zwischen Sender (Lautsprecher) und Empfänger (Mikrofon) gemessen. Problematisch ist hierbei, dass die Geschwindigkeit des Schalls von der Umgebung (Temperatur, Druck, relative Luftfeuchte und Turbulenzen) abhängt.

Parasitäres Tracking Es würde auch die Möglichkeit bestehen, über Bluetooth oder WLan eine Positionsbestimmung des Mobiltelefons durchzuführen. Wie bereits in Kapitel 2.1.7 erwähnt, erfolgt hier die Ortung über Triangulation. Jedoch sollen die Trackingverfahren für jedem Benutzer sofort einsetzbar sein. Um parasitäres Tracking betreiben zu können, müssen die Sendestationen vorhanden und das System auf sie Kalibriert sein. Das macht das angestrebte Trackingsystem vorerst uninteressant für unseren Anwendungsfall. Verfügbare Systeme erreichen meist nur eine mit GPS vergleichbare Genauigkeit.

2.2.2. Geeignete Trackingverfahren 1: Optische Verfahren

Die Augmentierung findet im Kamerabild statt. Deswegen bietet es sich an, dieses auch gleichzeitig für optische Trackingverfahren zu benutzen. Da Trackingsensor und Augmentierung auf den gleichen Bilddaten stattfinden ist mit einer einfachen Kalibrierung eine ziehmlich genaue Registrierung zu erreichen. In [14, 25] ist zu sehen, wie schwarze Vierecke, die in einer Bekannten Konfiguration aufgebaut sind, durch optische Trackingverfahren erkannt werden und die Pose der Kamera relativ zu diesen errechnet wird. Die Trackingverfahren haben sich mit der Zeit weiterentwickelt, die Problemstellung blieb aber die gleiche. In diesem Kapitel werden drei verschiedene Verfahren vorgestellt. Bei dem ersten Verfahren (Markertracking) werden Kanten gesucht. Im zweiten Verfahren (Featuretracking) werden eindeutig wiedererkennbare Bildpunkte gesucht. Im dritten Verfahren werden die Grauwerte von Pixelblöcken verglichen. Optische Trackingverfahren haben den Nachteil, dass immer eine Sichtlinie zwischen Kamera und den zu verfolgenden Objekten bestehen muss. Je nach verwendetem Trackingverfahren können Teile des Objekts verdeckt sein. Jedoch scheitert jedes Trackingverfahren ab einem gewissen Grad der

Verdeckung. Weiterhin sind optische Trackingverfahren im Vergleich zu mechanischem oder magnetischem Tracking sehr Rechenintensiv. Von Vorteil ist, dass selbst mit den Bildern der Mobiltelefonkamera eine hohe Präzision bei der Erkennung aller sechs Freiheitsgrade eines Objekts möglich ist.

Markertracking

Eines der verbreitetsten optischen Trackingverfahren ist das Erkennen von quadratischen Markern. Die Marker bestehen aus einem schwarzen Rand, der den Marker komplett umschließt und einem rotationsinvarianten Muster in der Mitte des Markers. Dieses Muster kann zum Beispiel eine Textur oder eine binär codierte Zahl sein, wie in der Abbildung 2.5 zu sehen. Das Muster in der Mitte des Markers dient der eindeutigen Identifikation des Markers, wenn mehrere Marker verwendet werden und der Erkennung der Orientierung des Markers. Bei der Verarbeitung eines Bildes wird zuerst nach Quadraten gesucht. Es wird anschließend versucht, eine ID innerhalb des Quadrats zu finden. Wird keine ID gefunden, handelt es sich bei diesem Quadrat nicht um einen Marker. Wurde eine ID gefunden, kann aufgrund der rotationsinvarianten ID die Orientierung des Markers bewstimmt werden. Damit kann man feststellen, welche Eckpunkte des gefundenen Quadrats zu welchen Eckpunkten des Markers korrespondieren. Anschließend werden über Homographie alle 6 Freiheitsgrade der Markerposition bestimmt. Von Vorteil ist beim Markertracking, dass Marker einfach zu detektieren sind. Es gibt kein Initialisierungsproblem und es sind keine 3D Modelle der Umgebung oder des zu trackenden Objekts nötig. Eine weit verbreitete Markertracker Bibliothek ist ARToolKit von Kato und Billinghurst [11].

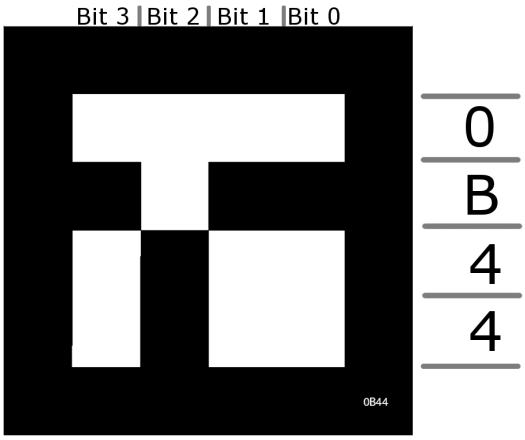


Abbildung 2.5.: Marker mit binärer ID 0x0B44. Die vierstellige ID ist als hexadezimale Zahl in 4 Zeilen codiert.

Featuretracking

Feature tracking Algorithmen basieren auf dem Wiederfinden lokaler Bildmerkmale. Um die Bildmerkmale über die Zeit verfolgen und wiederfinden zu können, sollten die Bildmerkmale unempfindlich gegenüber Koordinatentransformationen wie Translation, Rota-

tion und Skalierung sein. Die bekanntesten feature tracking Algorithmen sind der Scale-Invariant Feature Transform (SIFT) von Lowe [16] und Speeded Up Robust Features (SURF) von Bay et al. [1]. Ein guter Vergleich der Eigenschaften zwischen SIFT und SURF ist in [9] zu finden. Danach findet der SIFT Algorithmus mehr lokale Bildmerkmale. SURF findet zwar weniger, jedoch ist das Verhältnis zwischen korrekt und inkorrekt wiedergefundenen Bildmerkmalen nur geringfügig schlechter als bei SIFT. Die Qualität der Bildmerkmale ist bei SIFT geringfügig besser. SURF kann die Bildmerkmale dagegen schneller finden. Um wie beim Marker-Tracker die Position eines physikalischen Objekts zu erhalten, müssen die Bildmerkmale des Objekts dem System bekannt sein. Über diese Bildmerkmale können wie bei dem Markertracker mittels Homographie die sechs Freiheisgrade bestimmt werden.

Optischer Fluss

Als optischer Fluss wird ein Vektorfeld bezeichnet, dass die Bewegungsrichtung und -Geschwindigkeit für jeden Bildpunkt einer Bildsequenz angibt. Die meisten der hier eingesetzten Verfahren sind sehr Rechenaufwendig. Darum suchen wird für Mobiltelefone eine Alternative.

Daher wird der Projection Shift Analysis (PSA) Algorithmus [4] vorgestellt, der in einer abgewandelten Form hier zur Anwendung kommt. Den PSA Algorithmus kann man als Spezialfall eines Full Search Block Matching Algorithm (FSBM) betrachten. Er wurde speziell für mobile Geräte entwickelt. FSBM Algorithmen sind aus der Videokompression bekannt. Hierbei werden die Bewegungen von mehreren Pixelblöcken aus einem Bild über ie Zeit verfolgt. Es wird versucht, den Pixelblock so im nächsten Bild zu platzieren, dass sich die Farb- bzw. Grauwerte der einzelnen Pixel kaum oder gar nicht mehr von den Pixeln im aktuellen Bild unterscheiden. Die Bewegungen der Blöcke werden für die Videokompression verwendet. Bei dem PSA Algorithmus gibt es nur einen Block und zwar das gesamte Bild. Dadurch wird das Problem der Platzierung und Überprüfung wesentlich vereinfacht. In Kapitel 3.1.4 wird auf die Funktionsweise des PSA Algorithmus näher eingegangen. Als Ergebnis erhält man beim PSA Algorithmus die Verschiebung des Bildes in vertikaler und horizontaler Richtung. Bei dieser Verarbeitung wird weder auf die Objekte im Bild, noch ihre evtl. unterschiedlichen Bewegungen Rücksicht genommen. Es werden nur die Verschiebung der Pixel in horizontaler und vertikaler Richtung betrachtet. Daher wird diese Technik im weiteren als „Pixel Flow“ bezeichnet. Dadurch benötigt der Algorithmus zwar wenig Rechenzeit, wird aber anfällig für Fehler. Sind Bewegungen oder gleichfarbige Muster im Hintergrund, ist es wahrscheinlich, dass der Algorithmus fehlerhafte Ergebnisse produziert. PSA wurde ursprünglich als Interaktionsmethode für Mobiltelefone entworfen. Die Verschiebungen des Bildes wurden als virtuelle Mausbewegungen interpretiert [4].

2.2.3. Geeignete Trackingverfahren 2: Inertiale Verfahren

Über inertiale Tracking-Verfahren lässt sich die Position und Orientierung eines Objekts nicht bestimmen. Lediglich eine Verfolgung der Bewegungen des Objekts ist möglich. Um alle sechs Freiheitsgrade der Bewegungen bestimmen zu können sind drei Gyroskope und drei Beschleunigungssensoren notwendig. Inertiales Tracking ist eine gute Möglichkeit zu-

sätzliche Informationen über die Bewegungen eines Objekts zu erhalten. Sie sind völlig unabhängig von der Umgebung und können daher in ihrer Funktion nicht gestört werden, was bei anderen Tracking-Verfahren möglich ist. Die Informationen, die inertiales Tracking liefern kann, können verwendet werden, um andere Trackingverfahren zu stabilisieren oder zu ergänzen. In [12] werden Gyroskope eingesetzt, um die schnellen Kamerabewegungen einer am Kopf befestigten Kamera zu verfolgen, selbst wenn die Bilder der Kamera durch die Bewegungsunschärfe verzerrt sind.

2.2.4. Systemarchitektur

Wie bereits in Kapitel 2.1.1 erwähnt, ist die CPU des Mobiltelefons stark belastet, da Anwendungslogik, Tracking und Rendering von einer CPU bewältigt werden müssen. Bei einer Standalone Architektur werden alle Verarbeitungsschritte auf dem Mobiltelefon ausgeführt. Der Vorteil dieser Technik liegt darin, dass eine maximale Mobilität erreicht werden kann. Um die Last auf die CPU zu verringern, kann das AR-System oder die gesamte AR-Anwendung als Client-Server System entworfen werden. Dies bring verschiedene Vor- und Nachteile mit sich.

Eine AR-Anwendung mit optischem Tracking kann grob in vier zeitaufwendige Schritte unterteilt werden. Diese wären:

- Konvertieren des aktuellen Bildes (Umwandlung in Graubild, Schwarzweissbild, reduzieren der Auflösung)
- optisches Tracking
- Anwendungslogik
- Rendering

Jeder dieser Schritte kann auf einen Server ausgelagert werden.

In [20] werden alle Schritte bis auf die Bildkonvertierung auf einen Server ausgelagert. Der eingesetzte PDA ist mit dem Server über WLan verbunden. Es sollen große 3D Modelle mittels Marker-Tracking auf PDAs angezeigt werden. Dabei dient der PDA hauptsächlich als Ein- und Ausgabegerät der Videobilder. Der PDA nimmt einzelne Bilder auf, wandelt sie in Schwarzweissbilder um und sendet sie an den Server. Dieser Schritt ist der am wenigsten rechenaufwendigste und verringert die zu übertragenden Daten um ein vielfaches. Der Server wertet die Bilder aus, erstellt ein neues Farbbild, das nur die virtuellen Objekte enthält und sendet dieses zurück an den PDA. Dort wird nur das Aufgenommene Farbbild mit dem argumentierten Bild von Server überlagert und angezeigt. Dadurch wurden die rechenintensivsten Aufgaben auf den Server ausgelagert und die CPU des PDA kann für andere Aufgaben verwendet werden. Von Nachteil ist allerdings, dass ein leistungsstarker Server benötigt wird, wenn er von mehreren Benutzern gleichzeigig verwendet werden soll. Zudem ist eine schnelle Verbindung zum Server nötig, da ansonsten die Latenzzeiten zu groß werden.

2.2.5. Fusion verschiedener Verfahren

Verschiedene Tracking-Verfahren können verschiedene Arten von Ergebnissen liefern. So liefert z.B. ein Marker-Tracker alle sechs Freiheitsgrade eines Markers, während ein inertiales Verfahren nur die relativen Veränderungen liefert.

Eine der am häufigsten verwendeten Methoden um die verschiedenen Ergebnisse zu fusionieren, ist ein Extended Kalman Filter (EKF). Der EKF erlaubt im Gegensatz zu einem normalen Kalmanfilter auch nichtlineare Zustands- oder Beobachtungsgleichungen. Dies ist notwendig, da z.B. die Bewegungen eines Benutzers nicht linear sind.

Ein Beispiel hierfür ist in [29] zu finden. In diesem Beispiel werden optische Tracking-Verfahren mit Gyroskopen über einen EKF fusioniert um die Stabilität des Trackings und die Robustheit des Systems zu erhöhen. Um den EKF verwenden zu können, muss ein Zustandsmodell entworfen werden, dass das System beschreibt, in dem er Eingesetzt wird. In diesem Fall soll die Position der Kamera, an der auch die Gyroskope befestigt sind, bestimmt werden. Das Zustandsmodell beschreibt also die Position der Kamera in allen sechs Freiheitsgraden. Zudem werden Bewegungsmodelle benötigt, die die Daten der Tracking-Verfahren in Positionsveränderungen der Kamera umwandeln. Als letztes wird noch eine Matrix benötigt, die das Rauschen der am Prozess beteiligten Sensoren beschreibt. Diese Matrix muss im Vorhinein bestimmt werden und hat wesentliche Auswirkungen auf die Qualität der durch den EKF bestimmten Zustände.

Durch den Einsatz eines EKF können die Ungenauigkeiten der verschiedenen Tracking-Verfahren sehr gut kompensiert werden. Allerdings muss die Matrix, die das Rauschen der am Prozess beteiligten Sensoren beschreibt, für jedes Hardwaresetup neu bestimmt werden. Weiterhin ist der Rechenaufwand für einen EKF der alle sechs Freiheitsgrade einer Bewegung bestimmen soll, nicht zu vernachlässigen.

2.2.6. Fazit

In Tabelle 2.3 sind die Vor- und Nachteile der verschiedenen Trackingverfahren zusammengefasst. Von den optischen Trackingverfahren ist Featuretracking das mächtigste. Man ist nicht auf Marker angewiesen und kann prinzipiell jedes Objekt verfolgen. Jedoch macht der hohe Rechenaufwand es schwer, das Verfahren auf Mobiltelefonen zu implementieren. Wagner et al. haben es bereits geschafft, SIFT auf mobilen Geräten zu verwenden [26]. Bei ihnen ist es möglich, alle sechs Freiheitsgrade beliebig texturierter planarer Flächen zu bestimmen. Die Texturen müssen dem System jedoch vorher bekannt sein. Hierfür mussten starke Veränderungen am SIFT Algorithmus vorgenommen werden. Der Aufwand der Nachimplementation wurde für diese Arbeit als zu hoch eingeschätzt. Wagner und Schmalstieg haben mit ihrer Portierung von ARToolKit für mobile Geräte (ARToolKitPlus [27]) gezeigt, dass Markertracking effizient auf mobilen Geräten laufen kann. Im Vergleich zu dem Featuretracking System von Wagner bietet Markertracking auch keine wesentlichen Nachteile, da nur planare Flächen erkannt werden können. Weiterhin bietet Markertracking den Vorteil, dass nur Rechtecke erkannt werden, wodurch die Bestimmung der Markerpose mathematisch vereinfacht werden kann. In [7] wird dieses Vorgehen für die Berechnung der Homographie beschrieben. Markertracking ist ein erprobtes Verfahren mit moderatem Rechenaufwand. Zudem existierte am Lehrstuhl bereits eine Implementierung, auf die Aufgebaut werden konnte. Daher wurde Markertracking eines

der eingesetzten Verfahren.

Um nun die Position des Markers verfolgen zu können, müssen die Bewegungen des Mobiltelefons verfolgt werden, wenn der Marker nicht Sichtbar ist. Hierfür könnten der Pixel Flow und die Inertialsensoren verwendet werden. Wie bereits in Kapitel 2.1.5 angesprochen, verfügen Mobiltelefone zur Zeit nur über Beschleunigungssensoren, mit denen es nicht möglich ist, alle Bewegungen immer verfolgen zu können. Es kann nur in bestimmten Situationen die Neigung des Mobiltelefons bestimmt werden. Der Pixel Flow hingegen liefert immer brauchbare Ergebnisse. Da wie in Kapitel 1.2 besprochen, die Benutzer das Mobiltelefon immer nur um bestimmte Achsen Rotieren, bietet sich der Pixel Flow zusammen mit einem Bewegungsmodell an, um die Rotationen des Mobiltelefons zu bestimmen. Daher wird der Pixel Flow als zweites Tracking-Verfahren eingesetzt.

Ein EKF ist bei dieser Problemstellung nicht die beste Möglichkeit, die Ergebnisse der verschiedenen Trackingverfahren zu Fusionieren. Entweder wird der Marker erkannt, was zu einer genauen Bestimmung aller sechs Freiheitsgrade führt oder der Marker ist nicht sichtbar, wodurch der Markertracker keine Daten liefert. Es besteht keine Notwendigkeit, die Markererkennung durch den Pixel Flow zu verbessern. Dies würde lediglich zu einem erhöhtem Rechenaufwand führen. Daher werden die Ergebnisse von Marker-Tracker und Pixel Flow nur über ein einfaches Bewegungsmodell vereint und das auch nur, wenn der Marker nicht sichtbar ist.

Da eine schnelle und konstante Verbindung zu einem Server unrealistisch erscheint, wird die Standalone Architektur gewählt. Dadurch kann auch die maximale Mobilität gewährleistet werden.

Tracking-Verfahren	Vorteile	Nachteile
Magnetisches Tracking	keine Sichtlinie, geringer Rechenaufwand	anfällig für Störungen, zusätzliche Hardware
Mechanisches Tracking	keine Sichtlinie, geringer Rechenaufwand	sperriger Aufbau, zusätzliche Hardware
Akustisches Tracking	keine Sichtlinie, geringer Rechenaufwand	von Umwelt beeinflusst, zusätzliche Hardware
Parasitäres Tracking	keine Sichtlinie, geringer Rechenaufwand, keine Zusatzhardware	nur Position
Markertracking	bestimmung aller 6 Freih., moderater Rechenaufwand, kein Initialisierungsproblem, keine 3D Modelle nötig	Sichtlinie erforderlich
Featuretracking	bestimmung aller 6 Freih., alle Objekte verfolgbar, alle Bewegungen verfolgbar	Sichtlinie erforderlich, hoher Rechenaufwand, 3D Modell nötig,
Pixel Flow	sehr geringer Rechenaufwand	anfällig für Bewegungen, anfällig für Muster, nur Verschiebung des Bildes in X/Y Richtung
Inertiales Tracking	geringer Rechenaufwand, robust	nur relative Ergebnisse, ein Sensor pro Achse erforderlich

Tabelle 2.3.: Vor- und Nachteile verschiedener Trackingverfahren

Teil II.

Entwicklung/Evaluation des Trackingsystems

3. Entwicklung des Trackingsystems

Dieses Kapitel stellt das in dieser Arbeit entwickelte Trackingsystem vor. In Kapitel 3.1 wird die Architektur des Trackingsystems beschrieben. Es wird erläutert, warum das Trackingsystem so konstruiert wurde und welche Optimierungen vorgenommen wurden. Kapitel 3.2 befasst sich mit der Konfiguration des Trackingsystems. Abschließend wird in Kapitel 3.3 der C# Wrapper für den AR-Viewer beschrieben.

3.1. Entwicklung

In diesem Kapitel werden logische Zusammenhänge und Funktionsweisen der einzelnen Teile beschrieben. Am Anfang von Kapitel 3.1.1 wird ein Überblick über die Struktur des Trackingsystems geschaffen. Mittels eines Klassendiagramms und verschiedenen Aktivitätsdiagrammen wird das Verhalten des Trackingsystems erklärt. In den nachfolgenden Kapiteln wird die Funktionsweise von Markertracker, Pixel Flow und der Fusion erklärt.

3.1.1. Architektur

Das Trackingsystem ist so entworfen, dass es prinzipiell auf jedem Mobiltelefon betrieben werden kann. Um dies zu gewährleisten, besitzt das Trackingsystem keine Komponenten für die Ansteuerung der Kamera, sondern erwartet ein Bild in RGB888, RGB565 oder 8 Bit Graustufen. Die Kamera wird über eine externe Komponente angesteuert. Als Resultat gibt das Trackingsystem die Pose der gefunden Marker zurück. Bereits in der Einführung wurde der AR-Viewer angesprochen, auf dessen Basis die Evaluierung des neuen Trackingsystems durchgeführt wurde. Der AR-Viewer wird in Kapitel 4.1 näher erklärt. Der AR-Viewer verwendet eine Kamerakomponente um sich neue Bilder im RGB565 Format zu holen und sie dem Trackingsystem zu übergeben.

Der Mainloop der Anwendung ist in Abbildung 3.1 zu sehen. Für jeden Frame wird ein neues Bild aus der Kamera geholt, dem Trackingsystem übergeben und anschließend auf Basis der Posen aus dem Trackingsystem die Objekte gezeichnet.

In Abbildung 3.2 ist die statische Struktur des Trackingsystems zu sehen. Als Schnittstelle für das Trackingsystem nach außen dient die Klasse *„TrackingSystem“*. Sie besitzt Funktionen, um ein neues Bild in das System zu geben und die Ergebnisse abzufragen. Initialisiert wird das Trackingsystem durch eine Konfigurationsdatei, die in Kapitel 3.2 beschrieben wird. Sobald ein neues Bild in das Trackingsystem gegeben wird, wird die Verarbeitung des Bildes gestartet. Die Funktion kehrt erst zurück, nachdem alle Verarbeitungsschritte vollzogen sind.

In Abbildung 3.3 wird die Dynamik des Systems beschrieben. Das neue Bild wird in Verarbeitungsschritt 1 von den Klassen *„GrayImage“* und *„BlackWhiteImage“* zuerst in ein 8 Bit Graustufen Bild mit halber Auflösung konvertiert und anschließend mit einem Schwellwertverfahren in ein Binärbild gewandelt.

Um die Leistung des Systems zu erhöhen, wurde die Umwandlung des RGB565 kodierten Kamerabildes in 8 Bit Graustufen mittels einer Look-Up Table (LUT) optimiert. Der RGB565 Farbwert eines Pixels wird als ein vorzeichenloser 16 Bit Wert interpretiert und als Index verwendet. Über den Index wird der 8 Bit Grauwert des Pixels aus der LUT entnommen. Die LUT benötigt 64kb zusätzlichen Arbeitsspeicher. Durch die LUT wurde die Zeit, die für die Farbkonvertierung nötig war, um 24% verringert. Detailiertere Zeitmessungen sind in Kapitel 4.2 zu sehen. Das Kamerabild mit einer Auflösung 240x320 Pixel wird um die Hälfte auf 120x160 Pixel reduziert. Durch die gerade Anzahl von Pixeln kann eine effektive lineare Interpolation implementiert werden. Dabei werden die Mittelwerte von 2x2 Pixelblöcken für jedes Pixel des kleineren Ergebnissbildes errechnet. Die niedrigere Auflösung des Bildes erhöht nicht nur die Leistung des Markertrackers, es reduziert ebenfalls die Effekte die durch Rauschen und Bewegungsunschärfe hervorgerufen werden und erlaubt ein stabileres Tracking.

Anschließend erfolgt die Verarbeitung durch den Markertracker (2), der das Graubild und das Schwarzweissbild verwendet, und den Pixel Flow, der nur das Graubild erfordert. Wenn der Marker gefunden wurde, werden durch den aktuellen Pixel Flow (3) und die Position des Markers im aktuellen und vorherigen Bild die Faktoren für die Fusion angepasst (4). Die Fusion wird in Kapitel 3.1.5 genauer beschrieben. Je nachdem, wie der Benutzer sein Mobiltelefon bewegt und welche Objekte im Raum stehen, können sich bei gleicher rotation des Mobiltelefons, unterschiedliche Werte beim Pixel Flow ergeben. Wird der Marker nicht gefunden, so wird aus dem Pixel Flow die Veränderung der Orientierung des Mobiltelfons ermittelt (5) und mit der letzten gefunden Markerpose des Markertrackers fusioniert (6). Während das System noch in der Entwicklung war, war bereits abzusehen, dass sich ein Drift über die Zeit, in der der Marker nicht sichtbar ist, akkumuliert. Wenn der Marker wieder gefunden wurde, sprang das Objekt an die korrekte Position. War der Drift zwar klein aber sichtbar, erweckte das springen des Objekts den Eindruck, die Markerpose würde zittern. Um diesen Effekt zu verringern, wird überprüft, ob die aktuelle und die vorherige Pose einen gewissen Abstand überschreiten. Wird der Abstand überschritten, so wird zwischen den zwei letzten Posen innerhalb einer vorher definierten Zeit interpoliert (7). Dabei wird das Ende der Interpolation, also die aktuelle Markerpose, in jedem Zeitschritt aktualisiert.

3.1.2. Kamera

Obwohl die Kameraansteuerung nicht Teil des Trackingsystems ist, kann sie die Leistung der gesamten AR-Anwendung stark beeinflussen. Die Codebasis der Kameraansteuerung wurde von Daniel Pustka zur Verfügung gestellt. Die DirectShow Schnittstelle zur Ansteuerung der Kamera verlangt das setzen einer Callback-Funktion, um die neuen Bilddaten zu empfangen. Anfangs wurde der Kopiervorgang bei jedem Aufruf der Funktion gestartet. Wurden in der Anwendung mehrere Objekte geladen, fiel die Bildwiederholrate unter 8 FPS. Bei einem Videostream von 15 FPS bedeutet das, dass fast doppelt so viele Bilder kopiert wurden als benötigt. Um unnötiges Kopieren zu verhindern, wurde die Callback-Funktion mit der Funktion zum Anfordern eines neuen Bildes über zwei Semaphoren synchronisiert. Der Ablauf ist in Abbildung 3.4 zu sehen. Ein neuer Kopiervorgang wird nur gestartet, wenn ein neues Bild angefordert wird und verfügbar ist. Dadurch konnte die Bildwiederholrate bei gleicher Anzahl von geladenen Objekten von 8 FPS auf

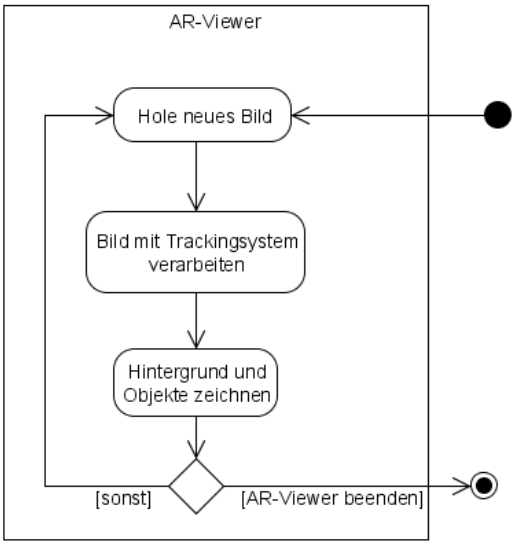


Abbildung 3.1.: Aktivitätsdiagramm AR Viewer, Mainloop

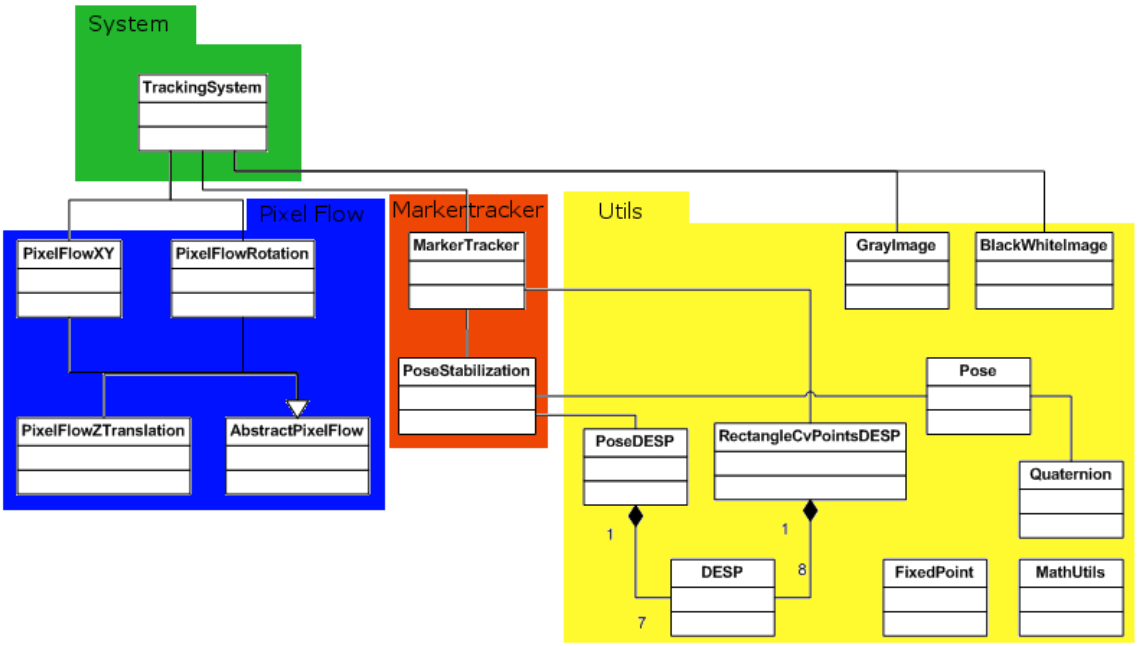


Abbildung 3.2.: Klassendiagram des Trackingsystems

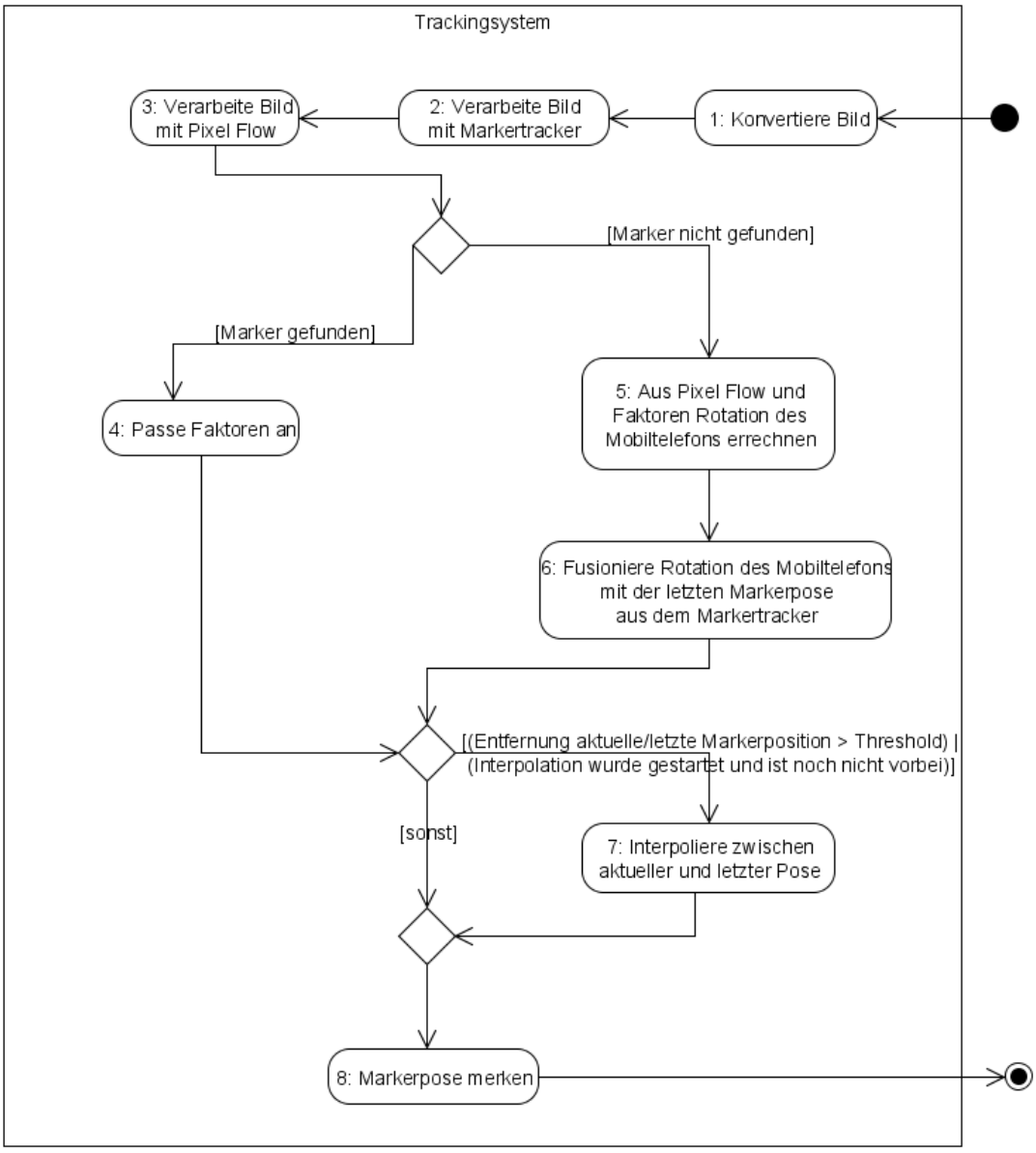


Abbildung 3.3.: Aktivitätsdiagramm Trackingsystem, neuer Trackingdurchgang

10 FPS gesteigert werden.

Die Kamerakomponente wurde weiter Optimiert, um möglichst wenige Ressouren zu verwenden. Zum Einen wurde dafür gesorgt, dass alle Puffer und Bildspeicher nur bei Initialisierung der Kamera alloziert werden. Des weiteren verwendet der AR-Viewer eine Textur um das Hintergrundbild darzustellen. Durch eine Einschränkung von OpenGL ES müssen zur Zeit noch Texturen in der Größe einer Zweierpotenz verwendet werden. Für das Hintergrundbild wird eine Textur der Größe 256x512 Pixel verwendet, da das Video eine Auflösung von 240x320 Pixel hat. Um den Prozess des Umkopierens zu beschleunigen, wurde er aus der Anwendung in die Kamera ausgelagert.

3.1.3. Markertracker

Der verwendete Markertracker basiert auf einer Übung der 3D Computer Vision Vorlesung der Technischen Universität München. Die Studenten sollten mit Hilfe von OpenCV einen Markertracker implementieren. Es wird auf die Musterlösung der Übung aufgebaut, die von Daniel Pustka bereitgestellt wurde. Um OpenCV auf dem Mobiltelefonen verwenden zu können, wurde die Bibliothek für Windows Mobile kompiliert. Neben ein paar Plattformabhänigen Präprozessoranweisungen mussten keine größeren Modifikationen an der Bibliothek vorgenommen werden. Die Funktionsweise des Trackingalgorithmus ist vergleichbar mit der des Algorithmus aus [3]. Der Markertracker produzierte zwar meistens stabile Ergebnisse, bei gewissen Betrachtungswinkeln auf den Maker und kleinen Markergrößen im Videobild begann jedoch die Markerpose extrem stark zu zittern. Ähnliche Effekte wurden bereits von Freeman [5] in Zusammenhang mit ARToolKit beschrieben. Eine mathematische Erklärung wird von Schweighofer [21] präsentiert. Daher wurde die Trackingpipeline des Markertrackers von mir erweitert, um die Markerpose zu stabilisieren. Die derzeitige Trackingpipeline ist in 3.5 zu sehen. Die Funktionalität des Markertrackers ist in der Klasse *„MarkerTracker“* implementiert. Zunächst werden in Verarbeitungsschritt 1 über eine OpenCV-Funktion die Konturen aus dem Schwarzweissbild extrahiert. Aus den Konturen werden über eine weitere OpenCV-Funktion Polygone extrahiert. Es wird überprüft ob die Polygone Quadrate darstellen und ob diese Quadrate einen Marker beschreiben (2). Die erste Optimierung, die durchgeführt wurde, um das Zittern der Markerpose zu unterdrücken, besteht darin, die Positionen der vier Eckpunkte des Markers über einen Double Exponential Smoothing Predictor (DESP) zu glätten. Hierfür wird die Klasse *„RectangleCvPointsDESP“* verwendet. Sie beinhaltet für jeden Eckpunkt und jede Dimension des Eckpunktes eine Instanz der *„DESP“* Klasse, in der der DESP Algorithmus implementiert wurde. Je nachdem, ob im letzten Bild ein Marker gefunden wurde oder nicht, werden die Eckpunkte geglättet (3) oder der DESP neu initialisiert (4). Anschließend wird mittels Homographie in Verarbeitungsschritt 5 die Markerpose bestimmt. Um das Gleichungssystem der Homographie zu lösen, wird eine weitere OpenCV-Funktion verwendet. Durch das Glätten der Eckpunkte hängt die Markerpose der wahren Position immer etwas hinterher. Um diesen Effekt zu kompensieren, wird die Markerpose nun ein zweites mal in Verarbeitungsschritt 6 geglättet, wenn im letzten Zeitschritt bereits ein Marker gefunden wurde. Ansonsten werden die Komponenten für die Glättung reinitialisiert (7). Die Glättung wird über die Klasse *„PoseStabilization“* durchgeführt. Sie kapselt die Funktionalität die in [15] beschrieben wird. Neben der Glättung wird die Markerpose extrapoliert, um das Hinterherhängen der Markerpose zu kompensieren.

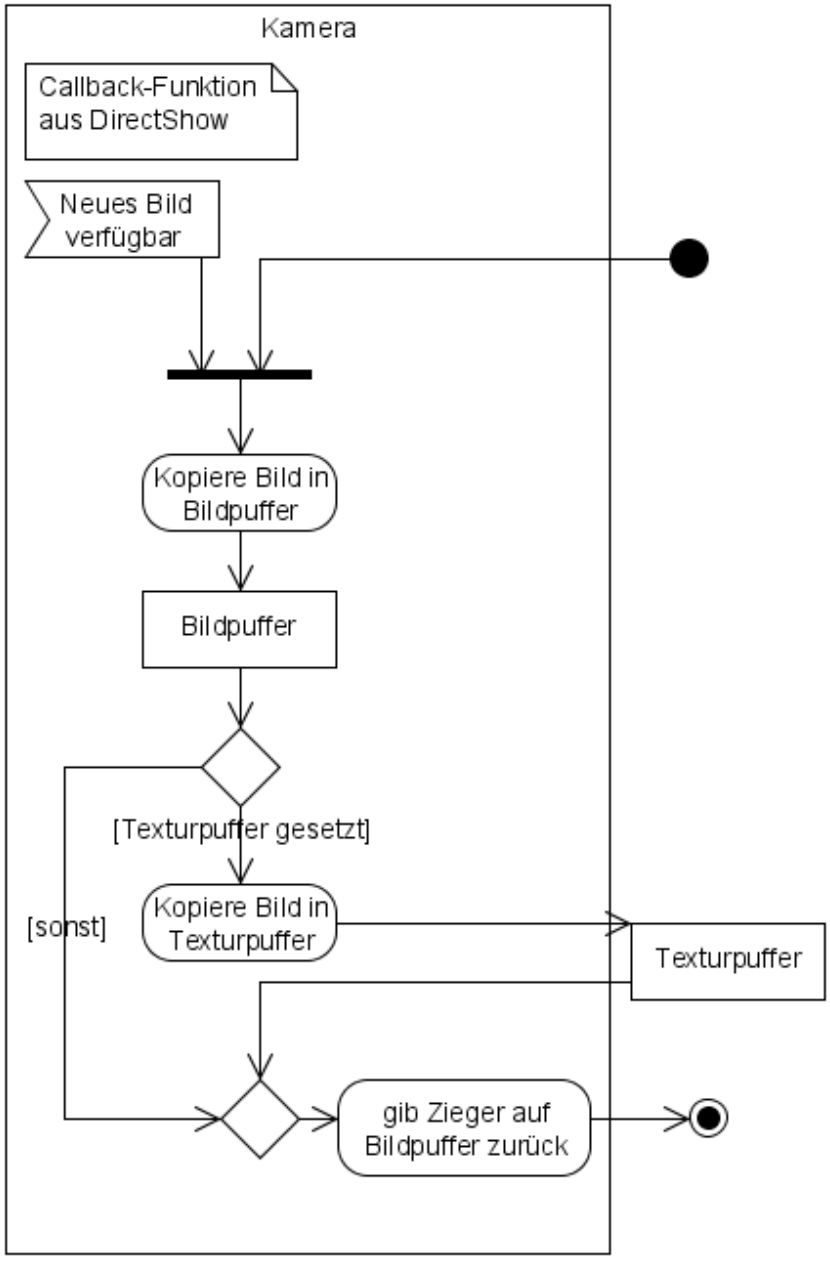


Abbildung 3.4.: Aktivitätsdiagramm Kamera, Bild holen

Die aktuelle Markerpose wird durch Interpolation zwischen der geglätteten Markerpose und der extrapolierten Markerpose bestimmt. Dabei wird die Position linear und die Rotation über den SLERP Algorithmus [22] interpoliert. Die einzelnen Faktoren für Interpolation, Extrapolation und das Glätten durch den DESP wurden durch ausgiebige Tests bestimmt. Die Faktoren können über die Konfigurationsdatei modifiziert werden. Nach dem die Markerpose nun bestimmt wurde, ist die Verarbeitung im Markertracker abgeschlossen.

Um die Leistung des Markertrackers zu steigern, wurden alle dynamisch angelegten Puffer durch statische Puffer ersetzt, um die Rechenzeiten für die Allokierung des Speichers zu sparen. Zudem wurden die meisten Berechnungen von Floating-Point auf Fixed-Point umgestellt. Für die Berechnung der Markerpose konnten allerdings Fixed-Point Zahlen nicht verwendet werden, da es Probleme mit der Genauigkeit und deren Wertebereich gab. Durch diese Maßnahmen konnte die Leistung des Markertrackers im Vergleich zur ursprünglichen Implementierung um 23% gesteigert werden.

3.1.4. Pixel Flow

Der PSA Algorithmus wurde speziell für Mobiltelefone als Interaktionsmethode entworfen [4]. Er ermöglicht es, die Verschiebung des Bildes in horizontaler und vertikaler Richtung zu bestimmen. Die Implementierung zeichnet sich vor allem dadurch aus, dass der gesamte Algorithmus nur Integer Arithmetik verwendet. Dadurch ist der Algorithmus Recheneffizient auf Mobiltelfonen zu implementieren, da Mobiltelefone in der Regel keine FPU besitzen. Im folgenden wird die Funktionsweise des Algorithmus näher erklärt.

Der Algrorithmus verwendet ein 8 Bit Graubild um die Verschiebung eines Bilder in einer Bildersequenz zu bestimmen. Dafür werden, wie in Abbildung 3.6 zu sehen, ein vertikaler (Y) und horizontaler (X) Projektionspuffer erstellt. In den jeweiligen Projektionspuffern werden die Summen der Grauwerte, die über Spalten und Zeilen des Bildes erstellt wurden, gespeichert. In Abbildung 3.7 ist der zweite Verarbeitungsschritt zu sehen. Für jede Verschiebungsrichtung existieren nun zwei Projektionspuffer. Ein Projektionspuffer für das aktuelle Bilde und einer für das vorherige Bilde. Es wird nun versucht, die aktuelle Verschiebung des Bildes über die Projektionspuffer zu bestimmen. Hierfür wird eine Verschiebung (z.B. ein Pixel) angenommen und die absoluten Differenzen zwischen den Projektionspuffern gebildet. Über die Differenzen wird ein Fehlerwert erstellt indem die quadrierten Differenzen aufsummiert werden. Dieses Verfahren ist auch als Least Square Method bekannt. Je nachdem, um wie viele Pixel ein Projektionspuffer verschoben wurde, sind unterschiedlich viele Differenzen bei der Erstellung des Fehlerwertes beteiligt. Um den Fehlerwert zu normalisieren, wird er durch die Anzahl der an der Fehlerwertbildung beteiligen Differenzen geteilt. Im ursprünglichen PSA Algorithmus wurde für die Normalisierung noch die Wurzel des Fehlerwerts gezogen. Da der Fehlerwert in diesem System nur für vergleichende Operationen mit anderen Fehlerwerten verwendet wird, ist es nicht nötig die rechenaufwendige Wurzeloperation durchzuführen. Der Fehlerwert wird nun für alle möglichen Verschiebungen errechnet. Da jede Verschiebungsachse getrennt betrachtet wird, entspricht die Anzahl der möglichen Verschiebungen höchstens der Anzahl der Elemente im Projektionspuffer. Es wird Angenommen, dass die Verschiebung mit dem niedrigsten Fehlerwert der aktuellen Verschiebung entspricht. Um die Leistung und Stabilität des Algorithmus zu erhöhen, werden die Verschiebungen nur für einen vorher

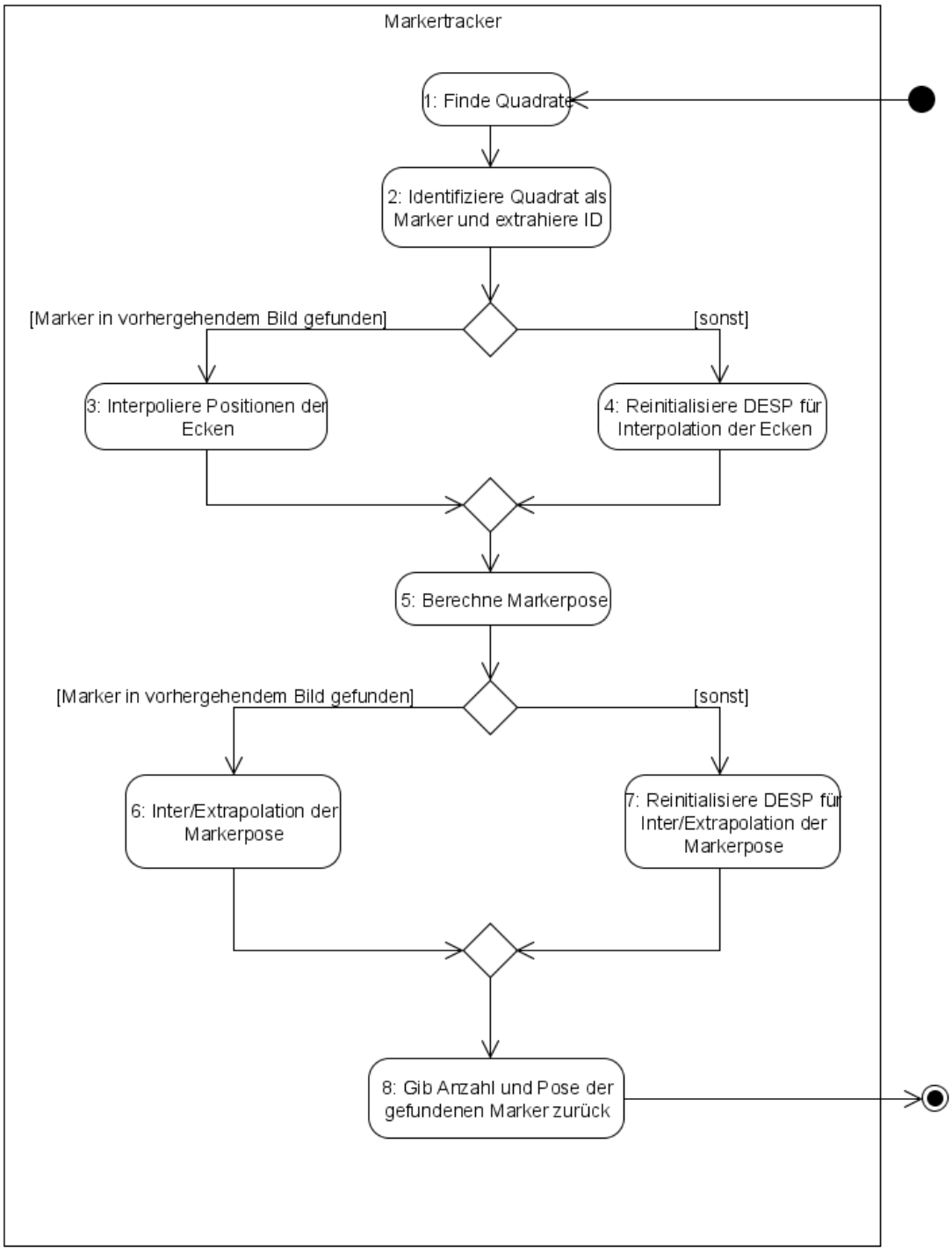


Abbildung 3.5.: Aktivitätsdiagramm Markertracker, neuer Trackingdurchgang

definierten Bereich durchgeführt. Von mir durchgeführte Tests haben gezeigt, das bei einer Verschiebung um mehr als 50% des Bildes öfter fehlerhafte Ergebnisse produziert werden. Dies liegt daran, dass nur noch wenige Differenzen für das Erstellen des Fehlerwerts verwendet werden und der Algorithmus gennerell anfällig auf gleichfarbige Hintergrunde und sicht wiederholende Muster ist. Der Bereich für die Verschiebungen wird über die Konfigurationsdatei bestimmt.

Um die Verarbeitung zu Beschleunigen, wird das gleiche Graubild mit reduzierter Auflösung verwendet, das bereits für den Markertracker verwendet wurde. Durch die Verarbeitung des Graubildes mit halber Videoauflösung können die Verschiebungen in der Regel sicher erkannt werden. Bewegt der Benutzer das Mobiltelefon jedoch relativ schnell, wird das Bild stärker Verzerrt und der Algorithmus liefert falsche Ergebnisse. Diese Situation ist durch einen überdurchschnittlich hohen Fehlerwert zu erkennen, der über die Konfigurationsdatei bestimmt wird. Um in diesem Fall verlässliche Ergebnisse zu erhalten, habe ich folgende Lösung: Die Auflösung wird erneut um 50% linear reduziert, was die Effekte der Verzerrung im Bild reduziert, und der Algorithmus ausgeführt. Wird wieder ein überdurchschnittlich hoher Fehlerwert erkannt, wird die Auflösung noch ein letztes mal reduziert und der Algorithmus erneut durchgeführt. Im schlimmsten Fall wird der Algorithmus also mit 1/2, 1/4 und 1/8 der Videoauflösung durchgeführt. Dadurch variert die Laufzeit und die Genauigkeit der Verarbeitung durch den Pixel Flow dynamisch. Zeitmessungen hierzu sind in Kapitel 4.2 zu sehen.

Um die Verarbeitung zu Beschleunigen, werden die Bilder für die Verarbeitung bei 1/4 und 1/8 der Videoauflösung nicht einzeln herunterskaliert, sondern die Projektionspuffer der vorhergehenden Verarbeitungsstufe mit höherer Auflösung verwendet. Dabei wird ein Element des Projektionspuffers mit niedrigerer Auflösung definiert durch den Mittelwert zweier Elemente des Projektionspuffer mit höherer Auflösung, die zu diesem Element korrespondieren. Mathematisch sind diese Operationen äquivalent zu dem linearen Herunterskalieren der Bilder auf die entsprechende Auflösung und Errechnen des Projektionspuffers. Durch diese Optimierung sind die geringen Laufzeiten bei 1/4 und 1/8 der Auflösung zu erklären.

Da sich dieser Algorithmus in kleinen Experimenten als recheneffizient und zuverlässig herausstellte, wurde versucht auch andere Verschiebungen des Bildes über diesen Ansatz zu bestimmen. Um die Programmierung zu vereinfachen, wurde die abstrakte Klasse „AbstractPixelFlow“ eingeführt, die die Funktionen zum Errechnen des Fehlerwertes und der Verschiebung enthält.

Rotation um Z-Achse

Es wurde versucht die Rotation des Mobiltelefons um die Z-Achse der Kamera über dieses Verfahren zu bestimmen. Hierfür musste der Projektionspuffer anders modelliert werden. Das Bild wird ausgehend von dem Bildmittelpunkt in Tortenstücke unterteilt, wie in Abbildung 3.8 zu sehen ist. Dabei bildet der Projektionspuffer einen Ring. Abbildung 3.8 ist zwar nur eine schematische Darstellung, jedoch ist zu sehen, dass Pixel, die nah am Bildmittelpunkt liegen nicht gleichmäßig auf die Tortenstücke verteilt werden können. Daher werden die Pixel, die nah am Bildmittelpunkt sind, für die einzelnen Tortenstüke ignoriert. Pixel, die durch das rechteckige Bildformat außerhalb des Kreises mit maximaler Ausdehnung liegen, werden ebenfalls ignoriert, um eine stabile Erkennung der Rotati-

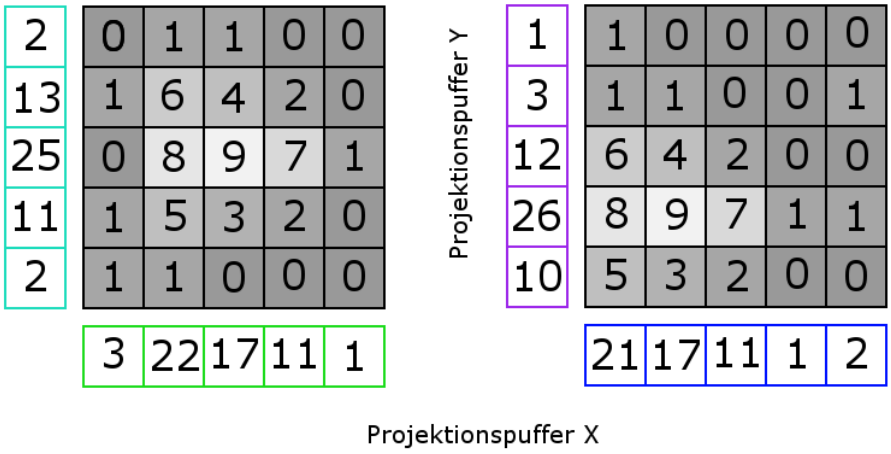


Abbildung 3.6.: Horizontale und vertikale Puffer für zwei aufeinanderfolgende Bilder

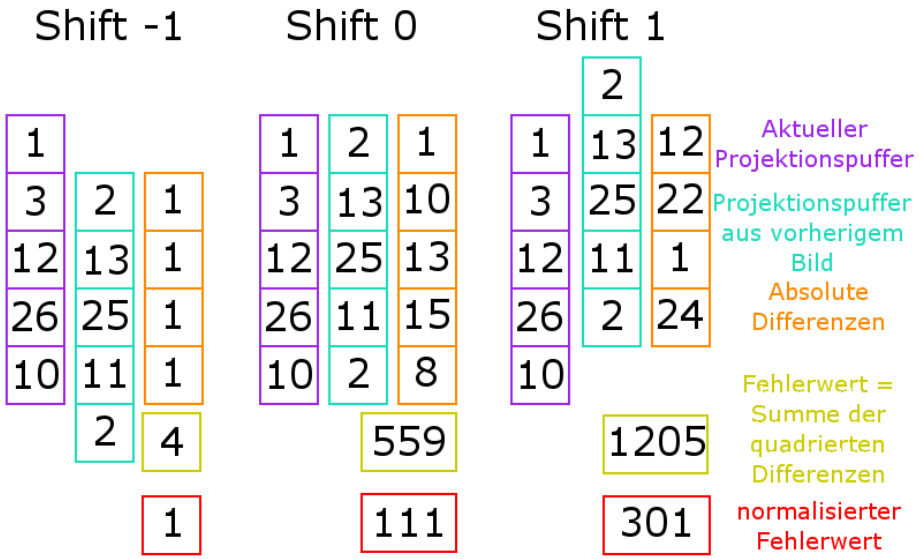


Abbildung 3.7.: Beispiel für die Bestimmung des Fehlerwertes

on zu ermöglichen. Da der Projektionspuffer nun einen Ring bildet, wurde das Verfahren zum Vergleichen der Projektionspuffer leicht modifiziert. Nun werden immer alle Werte der Projektionspuffer bei jeder möglichen Verschiebung ausgewertet, weil es immer einen Wert im anderen Projektionspuffer gibt, mit dem dieser verglichen werden kann. Eine Normalisierung über die Anzahl der beteiligten Differenzen ist nicht mehr notwendig. Problematisch hierbei ist die Wahl der Anzahl von Tortenstücken, die erzeugt werden sollen. Wird eine geringe Anzahl von Tortenstücken gewählt, ist die Erkennung einer Rotation relativ stabil, jedoch sind die Winkel der Tortenstücke dann relativ groß. Dies führt zu sprunghaften Rotationen. Wird eine feinere Winkeleinteilung gewählt, ist die Erkennung einer Rotation nicht stabil. Erschwerend kommt hinzu, dass sich bei einer natürlichen Handbewegung der Rotationsmittelpunkt verändert. Eine Rotation um den Bildmittelpunkt ist dabei nicht realistisch. Realistischer bei einer Bewegung des Benutzers ist eine Rotation um einen Punkt der ausserhalb des Bildes liegt. In Abbildung 3.9 ist die Aufteilung des Bildes in so einem Fall zu sehen. Selbst bei einer perfekten Rotation um den Bildmittelpunkt konnte keine zufriedenstellende Konfiguration gefunden werden. Zudem konnte bei einer beliebigen Bewegung nicht zuverlässig zwischen einer Rotation und einer Translation des Bildes unterschieden werden. Für Demonstrationszwecke wurde die Rotation um die Z-Achse in das Trackingsystem aufgenommen, jedoch ist sie standardmäßig ausgeschaltet.

Um die Pixel den Tortenstücken zuordenen zu können, müssen trigonometrische Berechnungen durchgeführt werden. Da sich die Zuordnung der Pixel zu den Tortenstücken zur Laufzeit nicht verändert, wurde die Verarbeitung durch Einführen einer LUT beschleunigt. Die LUT beschreibt, welches Pixel welchem Tortenstück zugeordnet ist.

Translation in Z-Richtung

Neben der Rotation um die Z-Achse wurde auch versucht eine Translation in Z-Richtung der Kamera über dieses Verfahren zu implementieren. Wird das Mobiltelefon in Z-Richtung bewegt, werden die sichtbaren Objekte in der Mitte größer. Objekte, die sich am Rand befinden, werden weiter in diese Richtung verschoben. Die Idee war nun, den Projektionspuffer über Ringe zu erstellen, wie in Abbildung 3.10 zu sehen. Wird das Mobiltelefon in Z-Richtung bewegt, sollten die Ringe nach außen oder innen wandern. Dieser Ansatz zeigte sich als wenig erfolgreich. Die Veränderung der Grauwerte der einzelnen Pixel war zu gering, um zuverlässig eine Translation feststellen zu können.

Wie bei der Rotation um die Z-Achse wurde die Verarbeitung über eine LUT beschleunigt.

3.1.5. Fusion

Wie bereits in Kapitel 3.1.1 erklärt, erfolgt eine Fusion der Markertracker Ergebnisse mit den Ergebnissen des Pixel Flows nur wenn der Marker nicht gefunden wurde. Hierfür wird ein Bewegungsmodell angenommen, in dem das Mobiltelefon nur um die X- und Y-Achse rotiert wird. Die Ergebnisse aus dem Pixel Flow werden mit einem Faktor multipliziert, der die Rotation in Radiant pro Pixel Bildverschiebung beschreibt. Die X- und Y-Achsen werden hierbei wie beim Pixel Flow getrennt betrachtet. Aus den beiden Winkeln der Rotation wird eine homogene 4x4 Rotationsmatrix erstellt. Die neue Markerpose

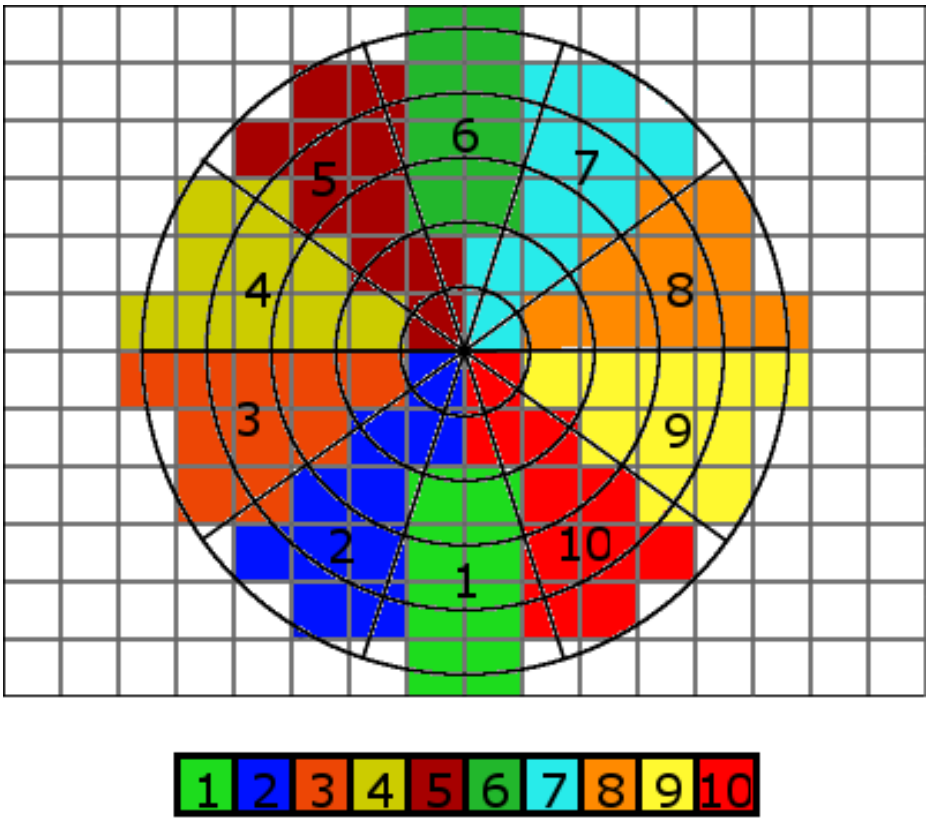


Abbildung 3.8.: Bildaufteilung für PSA Variante Rotation

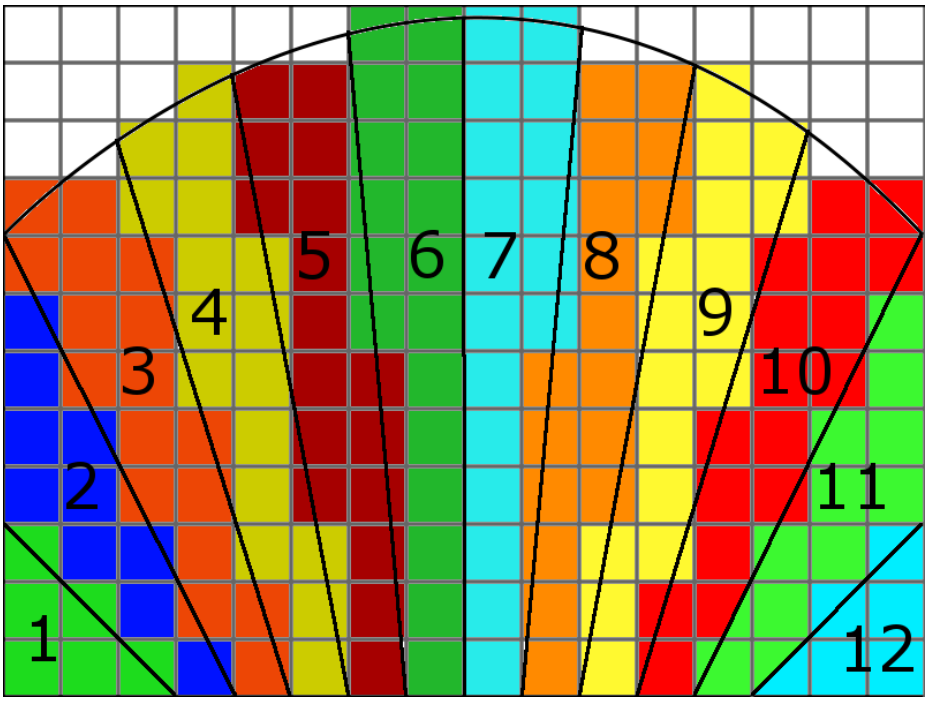


Abbildung 3.9.: Bildaufteilung für PSA Variante Rotation, Rotationspunkt ausserhalb des Bildes

(P_{Neu}) ergibt sich aus der multiplikation der Rotationsmatrix (M_{Rot}) mit der 4x4 Posematrix (P_{Alt}) der letzten gefundenen Markerpose aus dem Markertracker, zu sehen in Formel 1. Die Faktoren, die für die Fusion verwendet werden, verändern sich zur Laufzeit. Je nach dem wie weit die Objekte im Raum von dem Mobiltelefon entfernt sind, ergeben sich bei gleicher Rotation des Mobiltelefons unterschiedliche Werte beim Pixel Flow. Sind die Objekte nah an der Kamera, werden die Ergebnisse des Pixel Flows größer. Je weiter die Objekte entfernt sind, um so kleiner werden die Ergebnisse. Zudem verändert sich der Winkel zwischen Mobiltelefon und Marker wenn die gleiche Rotation einmal im Handgelenk und einmal im Ellenbogen durchgefürt wird. Dieses hängt, wie später zu sehen ist, stark vom jeweiligen Benutzer ab. In Abbildung 3.11 sind die Auswirkungen skizziert. Daher werden die Ergebnisse aus Markertracker und Pixel Flow für die Kalibrierung der Faktoren verwendet, wenn der Marker gefunden wurde. Es werden die Winkel zwischen den letzten beiden Markerposen errechnet und durch den derzeitigen Pixel Flow geteilt. Das Single Exponential Smoothing Verfahren wird verwendet, um die Faktoren der Fusion zu glätten. Die Formel des Verfahrens ist in 2 zu sehen. Der Faktor wird durch S_t repräsendiert. Die Ergebnisse des aktuellen Zeitschritts dienen dabei als neue Beobachtungen für das Glättungsverfahren (y_{t-1}). Dadurch ist es dem Trackingsystem möglich, sich automatisch auf die Umgebung und das Verhalten des Benutzers einzustellen.

$$P_{Neu} = M_{Rot} P_{Alt} \tag{3.1}$$

Formel 1: Multiplikation der Rotations- und Posematrix

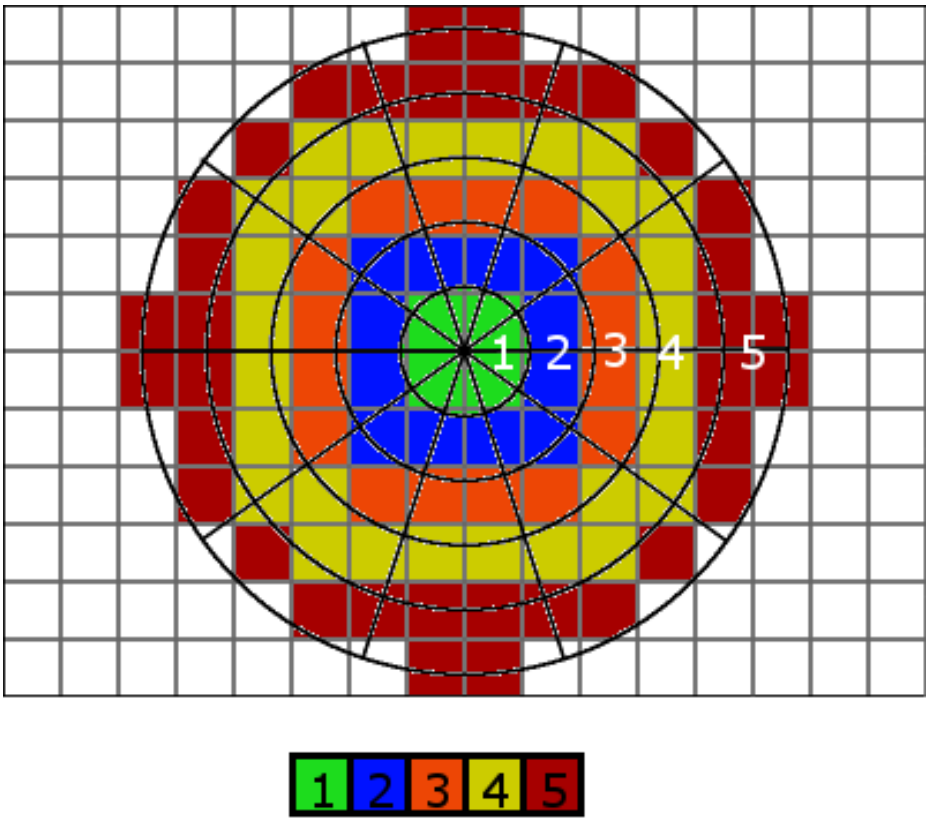


Abbildung 3.10.: Bildaufteilung für PSA Variante Translation

$$S_t = \alpha y_{t-1} + (1-\alpha)S_{t-1} \quad 0 < \alpha \leq 1 \quad t \geq 3 \tag{3.2}$$

Formel 2: Single Exponential Smoothing

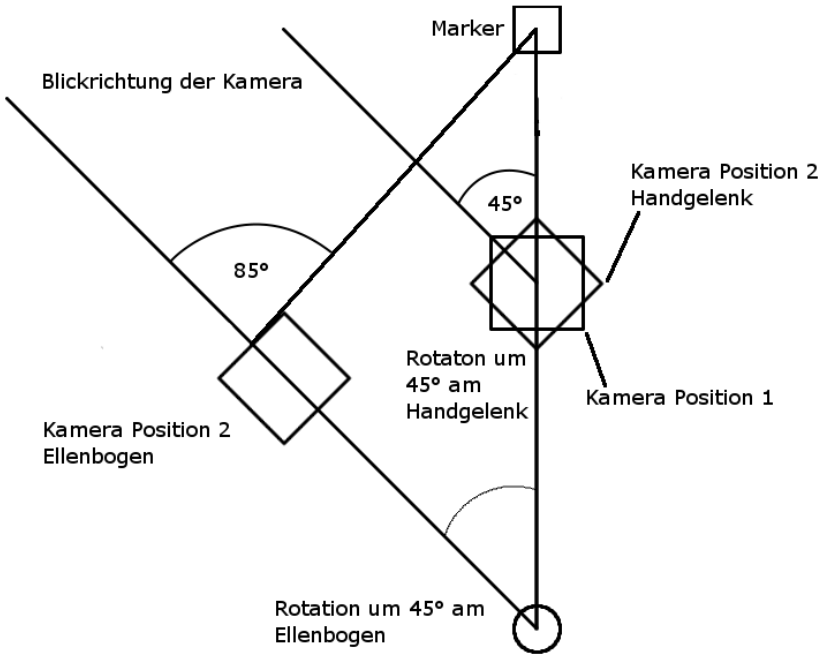


Abbildung 3.11.: Unterscheidliche Winkel zwischen Kamera und Marker bei unterschiedlichen Rotationspunkten

3.1.6. Kalibrierung

Zur Zeit muss lediglich die Kalibrierung der integrieten Kamera vor Benutzen des Trackingsystems durchgeführt werden. Das verwendete Lochkameramodell wird bestimmt durch den Bildmittelpunkt (principal point) und die Brennweite. Radiale und tangiale Verzerrungen sowie der Parameter der beschreibt wie windschief die Sensorelemente sind werden ignoriert. Diese Parameter können ignoriert werden, da wie in Kapitel 2.1.4 gezeigt, die Verzerrung der Bilder nicht stark ist und eine Entzerrung viel Rechenzeit benötigen würde. Das Trackingsystem besitzt keine Komponenten, um die Kalibrierung durchführen zu können, daher muss sie mit Hilfe eines externen Programms durchgeführt werden. Für diese Arbeit wurde die „Camera Calibration Toolbox for Matlab®“ [2] von Jean-Yves Bouguet, California Institute of Technology verwendet. Wie der Name schon sagt, handelt es sich hierbei um Module für Matlab, die aus Bildern eines Schachbrettmusters aus verschiedenen Perspektiven die Kameraparameter errechnet. Um die Kamera zu Kalibrieren, müssen daher die Bilder des Schachbrettmusters mit dem Smartphone aufgenommen und auf einem PC verarbeitet werden.

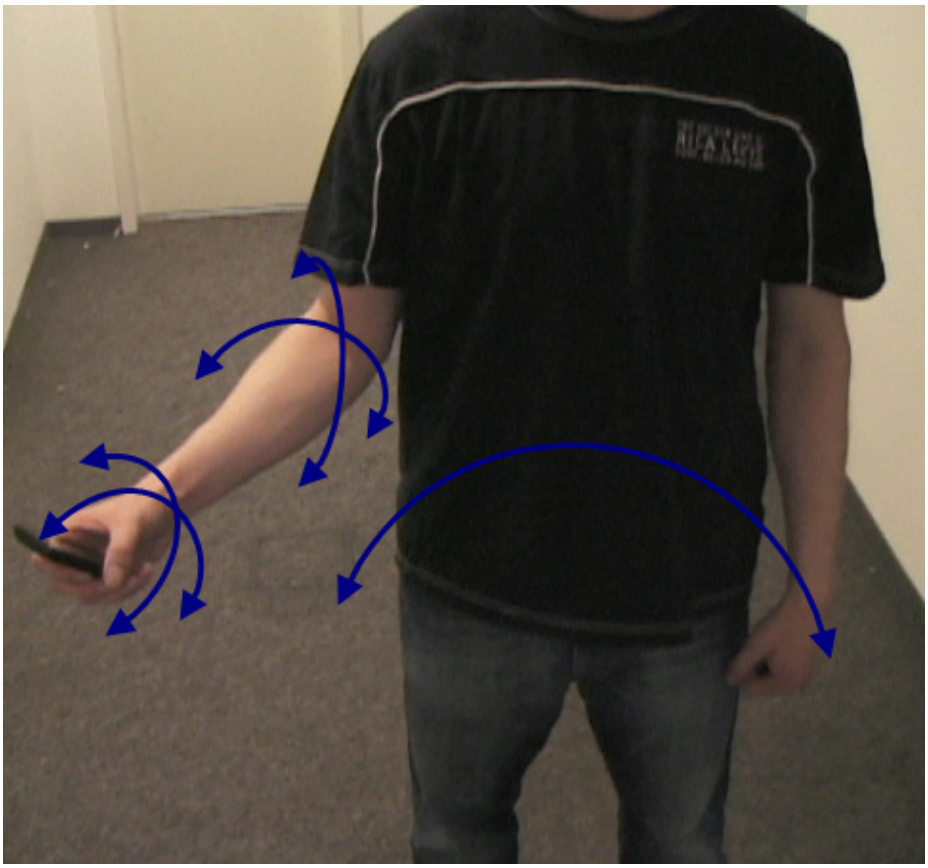


Abbildung 3.12.: Typische Rotationsachsen bei Verwendung des AR-Viewers. Durch welche Achsen gedreht wird, ist vom Benutzer abhängig

3.2. Konfiguration des Trackingsystems

Das Trackingsystem ist über eine ASCII-Textdatei konfigurierbar. Es wurde eine Textdatei, XML vorgezogen, da Textdateien einfacher auf Mobiltelefonen zu editieren sind. XML Dateien werden meist automatisch in einem Browser geöffnet, der keine Manipulationsmöglichkeiten bietet. Textdateien hingegen mit einem Texteditor (Windows Mobile Word). Ein Beispiel einer Konfigurationsdatei kann in Kapitel A betrachtet werden. Im folgenden werden die einzelnen Parameter beschrieben.

ccx, ccy Bildmittelpunkt des Lochkammeramodells bei halber Auflösung des Videostreams in Pixeln.

fcx, fcy Brennweite des Lochkammeramodells bei halber Auflösung des Videostreams in Pixeln.

BWThreshold Schwellenwert für das Schwellenwertverfahren um das Schwarzweissbild zu erzeugen.

MarkerSize Größe des Markers in Millimetern.

RotationSegments Anzahl der Tortenstücke für die Rotation um die Z-Achse.

UsePixelFlowXY Gibt an, ob der Pixel Flow für Rotationen des Mobiltelefons um die X- und Y-Achse verwendet werden soll.

UsePixelFlowZ Gibt an, ob der Pixel Flow für Rotationen des Mobiltelefons um die Z-Achse verwendet werden soll.

PixelFlowThreshold Schwellenwert, der bestimmt, ab welchem Fehlerwert des Pixel Flows die mehrstufige Verarbeitung gestartet werden soll.

MaximumShift Bestimmt die Größe des Bereichs um den Bildmittelpunkt, in dem Verschiebungen in horizontaler und vertikaler Richtung möglich sein sollen. Der Bereich wird in Pixeln angegeben und bezieht sich auf die Auflösung des Videostreams.

Origin Bestimmt die Ausrichtung des Kamerabildes (flip).

InterpolatePoses Bestimmt ob zwischen zwei Markerposen interpoliert werden soll, wenn sie zu weit von einander entfernt sind.

InterpolationThreshold Maximaler Abstand der zwischen zwei Markerpose bestehen kann, bevor sie interpoliert werden.

InterpolationMSecDuration Dauer, in der zwischen den zwei Markerposen interpoliert werden soll.

UseExtrapolation Bestimmt, ob eine extrapolation der Markerpose im Markertracker stattfinden soll.

SlerpFactorFuture Faktor für die SLERP Interpolation zwischen aktueller und extrapolierter Markerpose.

DESPAlphaRot Faktor für das Glätten der Rotation der Markerpose über den DESP Algorithmus.

DESPAlphaTrans Faktor für das Glätten der Translation der Markerpose über den DESP Algorithmus.

DESPAlphaPoints Faktor für das Glätten der Eckpunkte des Markers über den DESP Algorithmus.

MarkerCount Anzahl der Marker, die vom System erkannt werden sollen.

Marker Für jeden Marker der erkannt werden soll, wird ein „Marker“ Eintrag erstellt. Nach „Marker“ wird die ID des Markers als Hexzahl angegeben. Z.B. 0x0B44

3.3. C# Wrapper

Um das Trackingsystem zu testen, wurde der AR-Viewer (siehe Kapitel 4.1) verwendet. Da dieser aber in C# entwickelt wurde, habe ich einen C# Dll Wrapper entwickelt.

Kamera Die Ansteuerung der integrierten Kamera erfolgt ebenfalls über eine C++ Dll. Angesprochen wird die Kamera über die DirectShow Schnittstelle. Neben einer *„getCurrentImage“* Funktion, die einen Pointer auf den von der Kamera verwalteten Speicherbereich zurückgibt, in dem sich das aktuelle Bild befindet, gibt es noch eine *„setTextureBuffer“* Funktion. Ihr übergibt man einen Pointer auf den Speicherbereich der Textur sowie die Höhe und Breite der Textur. Wird nun die „getCurrentImage“ Funktion aufgerufen, wird wie in Kapitel 3.1.2 beschrieben automatisch auch die Hintergrundtextur aktualisiert.

Trackingsystem Auch für das Trackingsystem wurde eine Schnittstelle entwickelt, die auf zwei Funktionen reduziert werden kann. Eine *„init“* Funktion, der eine Konfigurationsdatei des Trackingsystems, Höhe, Breite und Pixelformat (RGB888, RGB565, GRAY) übergeben wird. Auf die Konfigurationsdatei wurde in Kapitel 3.2 eingegangen. Mit dieser Funktion wird das Trackingsystem initialisiert. Der Bildverarbeitung während der Laufzeit dient die Funktion *„processImage“*. Ihr wird ein Pointer auf den Speicherbereich des aktuellen Bildes übergeben, den man aus der *„getCurrrentImage“* Funktion der Kamera

erhält. Als Rückgabewert der Funktion erhält man ein Array von Objekten, die zur Zeit lediglich die 4x4 float Matrix der aktuellen Pose als eindimensionales Array in Column-Major Format und die ID des Markers Speichern. Alternativ wurde für die einfachere weitere Verarbeitung zusätzlich noch die Funktion *„processImageOpenGL“* eingeführt, die das Ergebnis zwar in gleicher Weise, allerdings in einer 16.16 Fixed-Point Repräsentation zurückgibt. Dadurch kann die Matrix direkt für das Rendering in OpenGL ES verwendet werden.

Zusätzlich wurden noch Klassen zum Speichern und Laden von Kamerabildern, und Posen entwickelt. Dadurch ist es möglich kurze Videos aufzunehmen um sie als Ersatz für die interne Kamera herzunehmen, um damit vergleichbare Zeitmessungen durchzuführen. Die gespeicherten Posen wurden dazu verwendet, bei Zeitmessungen, die kein optisches Feedback liefern, zu überprüfen, ob die aktuelle Pose korrekt errechnet wurde.

4. Evaluation des Trackingsystems

Um das Trackingsystem zu testen und die Annahmen über die typischen Bewegungen der Benutzer beim Verwenden einer mobilen AR-Anwendung zu überprüfen, wurde eine zweite Benutzerstudie mit dem neuen Trackingsystem durchgeführt. Im ersten Kapitel wird der AR-Viewer vorgestellt, auf dessen Basis die Evaluation des Trackingsystems durchgeführt wurde. Während der Benutzung des AR-Viewers wurden immer zwischen 9 - 15 FPS auf dem HTC Touch Diamond erreicht. Um festzustellen, wo die meiste Rechenzeit für die Bearbeitung benötigt wird, wurden Zeitmessungen vorgenommen. In Kapitel 4.2 werden die Ergebnisse der Zeitmessungen präsentiert. Kapitel 4.3 befasst sich mit der durchgeführten Benutzerstudie. Die Ergebnisse der Studie werden in Kapitel 4.4 präsentiert und bewertet.

4.1. Augmented Reality Viewer

Der AR-Viewer entstand im Rahmen eines Gemeinschaftsprojekts der Technischen Universität München, Fachgebiet Augmented Reality und der Vodafone Group R&D. Da der AR-Viewer in C# geschrieben wurde, bestand die notwendigkeit für den C# Wrapper. Neben marginalen Veränderungen um die Leistung des AR-Viewer zu verbessern, wurde im Rahmen dieser Arbeit eine weitere Interaktionsmöglichkeit implementiert, auf die später im Kapitel eingegangen wird.

Das Benutzerinterface des AR-Viewers besteht aus einer Toolbar, die über das Hintergrundbild gelegt wurde, wie zu sehen in Abbildung 4.1. Die Toolbar enthält Schältflächen zum Laden, Auswählen und Löschen von virtuellen Objekten. Sowie Schaltflächen um zwischen den Einstellungen, einem Betrachtungsmodus und verschiedenen Objektmanipulationsmodi zu wechseln. Die Objektmanipulationsmodi umfassen das Verschieben des Objekts in X, Y (bewegen) und Z-Richtung(anheben), ausgehend von dem Markermittelpunkt, drehen des Objekts um die Z-Achse des Markers und das Skalieren des Objekts. Die verschiedenen Objektmanipulationsmodi gingen aus den Wünschen der Benutzer aus der ersten Benutzerstudie bei Vodafone hervor, die in Kapitel 1.2 angesprochen wurde.

Schlagschatten werden auf dem Boden, auf dem die Objekte stehen gerendert, um den Eindruck der Immersion zu erhöhen. Solange der Marker vom Trackingsystem erkannt wird, wird er durch eine grüne Umrandung hervorgehoben. Wird der Marker nicht erkannt, z.B. weil Teile des Markers verdeckt sind, so markiert die grüne Umrandung die Position an der das Trackingsystem den Marker vermutet. Während der Interaktion mit einem bestimmten Objekt werden alle anderen Objekte zu 50% transparent gezeichnet um die derzeitige Auswahl hervorzuheben. Während der Objektmanipulation stehen dem Benutzer zwei Interaktionstechniken zur Auswahl. Zweitere wurde dabei im Rahmen dieser Arbeit entwickelt.

- *Touch manipulation:* Nachdem eine Funktion selektiert wurde, muss der Benutzer mit

seinem Finger eine Linie auf dem Touchscreen ziehen. Die relative Bewegung auf dem Bildschirm wird verwendet, um die Attribute der Objekte direkt zu manipulieren. Zum Beispiel indem die Positionattribute des Objekts auf der durch den Marker definierten Ebene verändert werden (bewegen). Dabei wird die relative Bewegung in Pixeln mit einem zuvor definierten Faktor multipliziert und auf das entsprechende Positionsattribut des Objekts addiert. Für jede Funktion wurde ein eigener Faktor durch ausgiebige Tests bestimmt.

- *Flow manipulation:* Um den Touchscreen für die Manipulationstechniken zu verwenden, sind meistens beide Hände erforderlich. Der PSA Algorithmus wurde ursprünglich als Interaktionstechnik entworfen. Daher lag es nahe den Pixel Flow aus dem Trackingsystem zu nutzen um ein Benutzerinterface zu entwerfen, dass mit nur einer Hand bedient werden kann. Zuerst muss die gewünschte Manipulationstechnik aus der Toolbar selektiert werden, am besten mit den Daumen. Dann, anstelle den Touchscreen zu verwenden, um z.B. das Objekt zu bewegen, muss der Benutzer den zentralen Knopf des Mobiltelefons drücken, während er das Mobiltelefon in die gewünschte Richtung bewegt. Wie auch bei der Touch manipulation, wird die relative Bewegung für die gewählte Manipulationstechnik verwendet, jedoch mit anderen Faktoren.

4.2. Zeitmessungen

Die Zeitmessungen des Trackingsystems wurden durch manuell eingefügte Befehle gemessen. Für die Zeitmessungen des AR-Viewers in C# war bereits eine Klasse vorhanden und in den Code integriert. Gemessen wurde anhand eines mit der Anwendung aufgenommenem Videos. Um das Video speichern und wieder laden zu können, wurde die entsprechenden Kameraimplementierung aus dem C# Wrapper verwendet. Durch nebenläufige Prozesse im AR-Viewer wurden die Zeitmessungen verfälscht. Daher wurden weitere Zeitmessungen mit einer zweiten Anwendung durchgeführt. Die zweite Anwendung besitzt selber keine Funktionalität und diente nur dem Messen des Trackingsystems. Auf Grundlage der verschiedenen Zeitmessungen konnte der Messfehler aus dem Messungen des AR-Viewers herausgerechnet werden. In Tabelle 4.1 sind die Ergebnisse der Zeitmessungen des AR-Viewers auf einem HTC Touch Diamond zu sehen. Es wurde ein Modell zum Rendern gewählt, durch das der AR-Viewer mit genau 10 FPS lief. Wie an den Zeiten zu sehen ist, wird knapp drei Viertel der Zeit zum Rendern benötigt, obwohl das gewählte Modell mit 1125 Dreiecken nicht sehr detailreich ist. Das Rendering ist der Flaschenhals der Anwendung, der höhere FPS verhindert. Die meiste Zeit des Trackingsystems wird für den Markertracker verwendet. Hierbei wird die meiste Zeit zum finden der Quadrate im Bild benötigt. Die OpenCV-Funktionen zum finden der Konturen und approximieren der Polygone verbrauchen dabei nur wenig Zeit. Die hälfte der Zeit verbringt der Markertracker mit dem Verfeinern der Quadrate, insbesondere der Interpolation der Positionen der Stützpunkte, um Geraden an die Kanten der Quadrate zu fitten um damit die Ecken der Quadrate zu optimieren. Die Funktion selber verwendet Fixed-Point Arithmetik und besteht aus nur wenigen mathematischen operationen. Daher kann die Funktion kaum noch

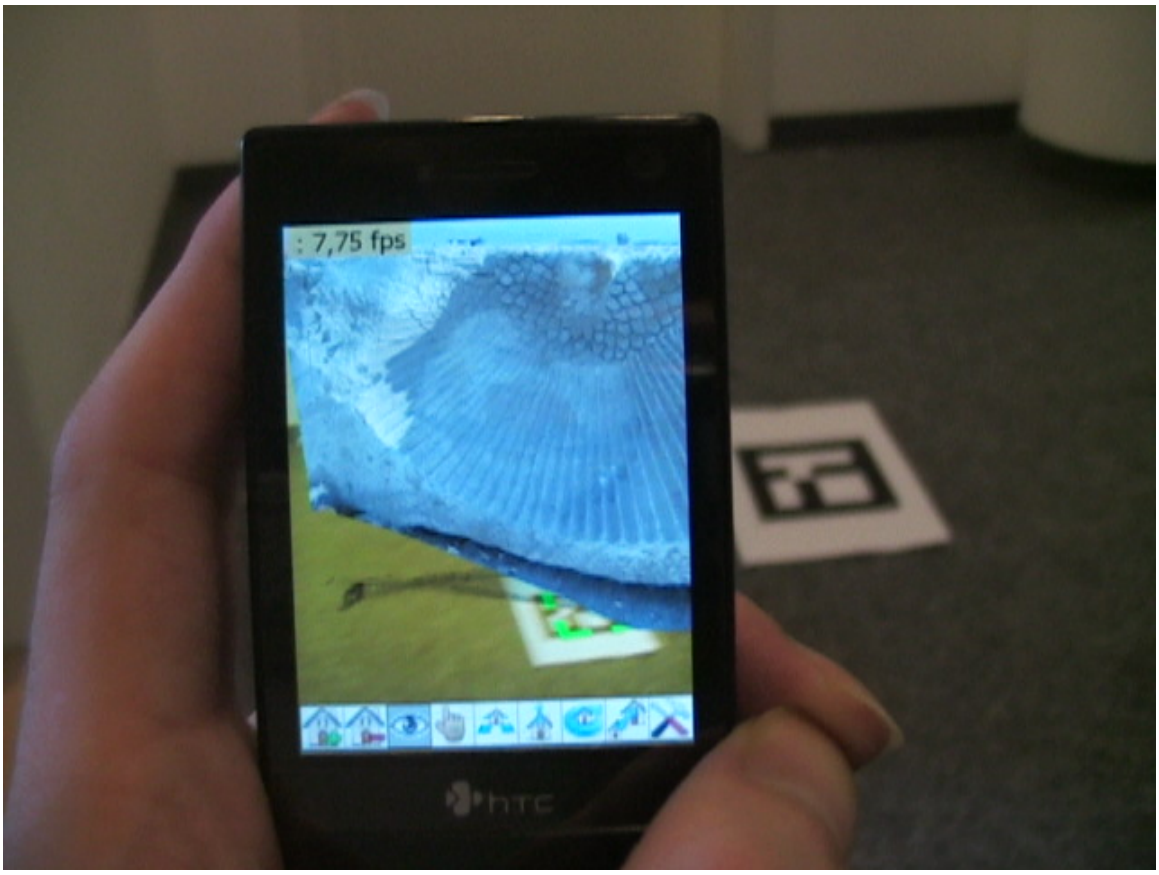


Abbildung 4.1.: AR-Viewer mit Modell eines türkischen Monuments. Drift nach dem der Marker schnell von rechts wieder in das Bild gekommen ist, bevor der Markertracker den Marker erkennen konnte

Komponente	*Zeit [ms]*
AR-Viewer	100.0
Kamera	12.10
Trackingsystem	13.0
Bildkonvertierung	2.20
Markertracker	9.85
Finde Quadrate	7.00
Finde Konturen (OpenCV)	1.00
Approximiere Polygone (OpenCV)	0.63
Verfeine Quadrate	5.10
Interpoliere Pixel Werte	2.26
Identifiziere Marker	0.0058
Errechne Pose	1.90
Pixel Flow	0.87
1/2 Auflösung	0.86
1/4 Auflösung	0.00098
1/8 Auflösung	0.00033
Rendering	74.91
Zeichne Hintergrund	17.56
Zeichne Objekt	57.35

Tabelle 4.1.: Typsiche Zeitwerte für den AR-Viewer mit einem Modell mit 1125 Dreiecken bei 10.0 FPS auf dem HTC Touch Diamond. Dell Axim 51v liefert ähnliche Ergebnisse

weiter optimiert werden. Jedoch wird durch das eingesetzte iterative Optimierungsverfahren diese Funktion über 1000 mal pro Bild aufgerufen, wodurch sie so viel Zeit verbraucht. Der Pixel Flow benötigt kaum Rechenzeit. Die meiste Rechenzeit wird für die Erstellung der Projektionspuffer bei halber Auflösung verwendet. Die Zeiten für die Errechnung des Pixel Flows bei $1/4$ und $1/8$ der ursprünglichen Auflösung sind vernachlässigbar gering. Da für die Fusion ein vereinfachtes Bewegungsmodell verwendet wird, sind die benötigten Rechenzeiten für die Fusion ebenfalls vernachlässigbar gering.

4.3. Benutzerstudie

Durch die zweite Benutzerstudie mit dem neuen Trackingsystem, sollte die Richtigkeit der Annahmen über die typischen Bewegungen der Benutzer überprüft werden (Rotation um Achsen in Hand, Ellenbogen und Hüfte). Zudem sollte überprüft werden, ob die „flow manipulation“ Interaktionsmethode von den Benutzern akzeptiert wird und für welche Manipulationsmethoden sie geeignet ist. Die Benutzerstudie wurde als explorative Studie ausgelegt. Wie bereits bei der ersten Benutzerstudie wurde den Probanden vorher nicht erklärt wie das neue Trackingsystem funktioniert. Erst wenn Probleme auftraten oder die Benutzer an die Grenzen des neuen Systems gestoßen waren, wurden sie über die Einschränkungen informiert. Die Benutzer sollten ohne Vorkenntnisse mit der Anwendung interagieren um ihr natürliches Verhalten beobachten zu können.

Aufbau/Ablauf Die Probanden hatten in einem within-subject design drei Aufgaben zu erfüllen. Die erste Aufgabe bestand darin, ein Gemälde, das auf einem an einer Wand platzierten Marker eingeblendet wurde, korrekt auszurichten und zu verschieben. Diese Aufgabe diente hauptsächlich dazu, den Probanden mit der Benutzeroberfläche des AR-Viewers bekannt zu machen. In der zweiten Aufgabe hatten die Probanden verschiedene virtuelle Gegenstände auf einem virtuellen Tisch zu platzieren. Hierfür wurde der Marker auf den Boden gelegt. Diese beiden Aufgaben wurden von den Probanden sowohl mit dem neuen Trackingsystem, als auch mit dem alten Trackingsystem (nur Markertracker) erfüllt, um Unterschiede in den Verhaltensweisen mit beiden System beobachten zu können und dem Probanden vergleichende Fragen stellen zu können. Die dritte Aufgabe bestand darin, eine ca. 2 Meter hohe Litfaßsäule zu betrachten. Die Litfaßsäule ist in Abbildung 4.2 zu sehen. Am oberen Ende der Litfaßsäule wurden 3 Poster verschiedener AR Konferenzen platziert. Die Probanden sollten die 3 Konferenzen identifizieren. Diese Aufgabe konnte nur mit dem neuen Trackingsystem erfüllt werden, denn hierfür musste das obere Ende der Litfaßsäule betrachtet werden, während der Marker auf dem Boden lag. Der Sinn dahinter war, den Probanden dazu zu zwingen, den Pixel Flow des neuen Trackingsystem zu verwenden. Während die Probanden die Aufgaben erfüllten wurde ihr verhalten beobachtet und protokolliert. Nach dem die Probanden die Aufgaben beendet hatten, wurde ein Interview geführt, in dem sie ihre eindrücke schildern sollten. Die einzelnen Fragen und Antworten der Benutzer sind in Kapitel B zu sehen. Die unabhängigen Variablen in dieser Benutzerstudie sind das alte und neue Trackingsystem. Als abhängige Variable wurde die Zufriedenheit der Benutzer mit der AR-Anwendung definiert. In Tabelle 4.2 sind die Einschätzungen der Probanden über ihre Erfahrung um Umgang mit Mobiltelefonen, 3D Spielen und AR zu sehen.

Proband	Alter	Beruf/Ausbildung	Erfahrung mit Mobiltelefonen	Erfahrung mit 3D Spielen	Erfahrung mit AR
1	25	Informatiker	10	10	1
2	22	IT Projektassistenz	5	10	1
3	25	IT Projektmanager	8	10	1
4	25	Informatiker	8	8	3
5	25	Informatiker	7	9	7
6	32	Architekt	6	2	1
7	30	Bürokaufmann	6	7	1
8	24	Lehrer (Sport/Eng)	6	4	1

Tabelle 4.2.: Probanden der Benutzerstudie (10 Experte, 1 Anfänger)

4.4. Auswertung und Beurteilung der Ergebnisse

Die erste Frage die den Probanden gestellt wurde, war ob die Erledigung der Aufgaben angenehm oder unangenehm für die Probanden mit dem jeweiligen Trackingsystem war. Die Ergebnisse sind in Tabelle 4.3 und 4.4 zu sehen. Die weite Streuung der Bewertungen des Systems kann durch dem Umstand erklärt werden, dass bis auf zwei Probanden (4 und 5), die Probanden zuvor noch nie ein AR-System verwendet hatten. Daher hatten sie

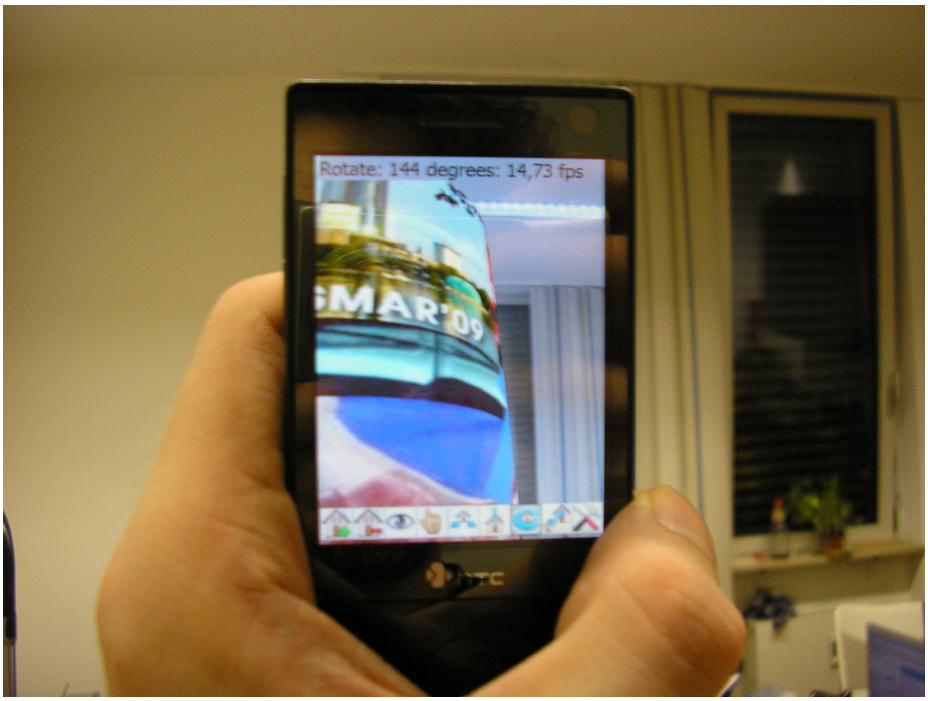


Abbildung 4.2.: Litfaßsäule aus Aufgabe 3 der Benutzerstudie

keine Erfahrung mit anderen System um das System einzuschätzen zu können. Die Zufriedenheit mit dem Gesamtsystem konnte drastisch gesteigert werden (im Durchschnitt +3). Dies lag zum Teil an der stabilisierten Markerpose. Zum Großteil aber lag es daran, dass sich der Proband keine Gedanken mehr über die Position des Markers machen musste. Er konnte das Mobiltelefon frei bewegen ohne Gefahr zu laufen, den Marker zu verlieren.

Proband	1	2	3	4	5	6	7	8	9	10
1			x							
2							x			
3		x								
4					x					
5			x							
6					x					
7						x				
8				x						

Tabelle 4.3.: Wie angenehm waren die Aufgaben zu lösen? (nur Markertracking, 1 unangenehm, 10 angenehm)

Weiterhin sollten die Probanden bewerten, ob sie Drift/Jitter der Markerpose und die Bildwiederholfrequenz generell als störend empfanden. Während der Benutzerstudie wurden zwischen 9 und 15 FPS erreicht, je nach größe und Anzahl der geladenen Modelle. Die Ergebnisse sind in den Tabellen 4.5, 4.6 und 4.7 zu sehen. Die meisten Probanden bemerkten den Drift der Markerposition zu Anfang nicht oder fühlten sich dadurch nicht gestört.

Proband	1	2	3	4	5	6	7	8	9	10	Änderung
1							x				+4
2										x	+3
3						x					+4
4							x				+2
5						x					+3
6									x		+4
7									x		+3
8								x			+4

Tabelle 4.4.: Wie angenehm waren die Aufgaben zu lösen? (neues Trackingsystem, 1 unangenehm, 10 angenehm)

Ein typischer Drift ist in Abbildung 4.1 zu sehen. Durch die Glättungen der Eckpunkte des Quadrats konnte der Jitter der Markerposition erheblich reduziert werden. Jedoch fing die Markerpose aus größeren Entfernungen jeden 30-40 Frame an zu flackern. Dies wurde von den Probanden als viel störender empfunden als der Drift der Position oder die niedrigen FPS. Mit dem alten Trackingsystem standen die Probanden meist unbewegt vor dem Marker. Alle virtuellen Objekte waren unanimiert, wodurch die niedrigen FPS nicht auffielen. Mit dem neuen Trackingsystem hatten die Probanden viel mehr Bewegungsfreiheit, die sie auch ausnutzten, wodurch die niedrigen FPS stärker auffielen.

1	2	3	4	5	6	7	8	9	10
	2	3	2		1				

Tabelle 4.5.: Wie sehr hat der Drift gestört? (neues Trackingsystem, 1 wenig, 10 sehr)

1	2	3	4	5	6	7	8	9	10
		2	2		1		1		2

Tabelle 4.6.: Wie sehr hat der Jitter gestört? (neues Trackingsystem, 1 wenig, 10 sehr)

Verhalten und Beobachtungen bei System mit altem Markertracking

- Rotation am Handgelenk x4
- es wird ignoriert, dass der Maker verlohren wurde x2
- durch ruhiges halten des Handys wurde nicht gemerkt, dass der Marker verloren wurde
- mit dem Handy um den Marker bewegt
- relativ starres Stehen vor dem Marker
- Handy starr vor dem Körper x4

1	2	3	4	5	6	7	8	9	10
	1	1	2			1	1		

Tabelle 4.7.: Wie störend waren die niedrigen FPS? (neues Trackingsystem, 1 wenig, 10 sehr)

- zwei Hände um Menü zu bedienen x5
- es wird versucht durch Rotation das Objekt wiederzufinden x2 (aus Bild verschoben)

Verhalten und Beobachtungen bei neuem Trackingsystem

- Rotation am Ellenbogen x5
- Rotation am Handgelenk x4
- Rotation aus dem Körper x2
- es wird immer wieder überprüft ob das Objekt noch auf dem Marker steht
- relativ starres Stehen vor dem Marker x3
- bei Touch manipulation geblieben, Flow manipulation gar nicht versucht
- Touch manipilation bevorzugt x2
- Flow manipulation bevorzugt x5
- versucht, das Handy um die Z-Achse zu drehen
- für Aufgabe 2 wurde von Flow manipulation auf Touch manipulation gewechselt, weil genauer x2
- mit Marker Anfangsposition finden, Marker verdecken bzw. aus Bild drehen, anschließend Aufgabe 2 bearbeitet
- versucht, näher an das Objekt heran zu gehen x4

Beurteilung der Ergebnisse Die Probanden standen immer noch vor dem Marker und Rotierten das Mobiltelefon um eine Achse liegend in Handgelenk, Ellenbogen oder Hüfte, wodurch die Annahmen über die Bewegungen der Benutzer bestätigt wurden. Mit den Bewegungsfeiheiten die das neue System brachte, versuchten manche Probanden, das obere drittel der Litfaßsäule zu betrachten und währenddessen um das Objekt zu gehen, um alle Poster auf der Säule zu erkennen. Das Trackingsystem unterstüzt jedoch diese Art von Bewegungen nicht, daher erhielt der Proband nicht das gewünschte Ergebnis. Nachdem die Einschränkungen des neuen Systems den Probanden erklärt wurden, stellten sie sich schnell darauf ein. Sie gingen nun entweder um den Marker, betrachteten den Marker für einige Sekunden, um das System die neue Position feststellen zu lassen und betrachteten Anschließend die Poster. Oder sie benutzen die Interaktionsmöglichkeiten der Anwendung um die Litfaßsäule rotieren zu lassen. Weiterhin wurde beobachtet, dass einige

Benutzer versuchten näher an das Objekt heranzugehen, wenn der Marker nicht sichtbar war, um mehr Details des Objekts zu sehen. Diese Bewegung wurde allerdings vom System nicht unterstützt.

Die Flow manipulation Interaktionsmethode wurde von den meisten Benutzern bevorzugt, obwohl ihre Entwicklung nicht so ausgereift war wie bei der Touch manipulation. Je nach Hintergrund produzierte der Pixel Flow nicht ganz korrekte Ergebnisse, wodurch die Steuerung ungenau wurde. Das wurde von den Benutzern vor allem beim erfüllen der zweiten Aufgabe bemängelt. Zudem hat sich gezeigt, dass die Flow manipulation nicht für alle Objektmanipulationsmodi geeignet ist. Für die Verschiebung des Objekts in X-,Y- und Z-Richtung war die Flow manipulation gut einsetzbar. Für die Skalierung des Objekts eignete sie sich nicht, da er das Objekt aus dem Blickfeld verlor. Teilweise verlor der Benutzer auch den Bezug zu anderen Objekten und war sich nicht mehr sicher, wie weit er das Objekt skalieren wollte. Für die Rotation des Objekts um die Z-Achse des Markers war die Flow manipulation ebenfalls nicht geeignet, weil während der Rotation des Mobiltelefons das Objekt aus dem Sichtbaren bereich verschwand.

5. Zusammenfassung

Eine Benutzerstudie, die vor dieser Arbeit durchgeführt wurde hatte gezeigt, dass eine AR-Anwendung für Mobiltelefone, die nur Markertracking einsetzt, für den Benutzer kein zufriedenstellendes Verwenden erlaubt. Das Trackingverfahren wird zum Showstopper. Im Rahmen dieser Master's Thesis wurde ein Markertrackingsystem erweitert, um das weiterverfolgen der Markerpose zu ermöglichen, selbst wenn der Marker nicht mehr sichtbar ist. Durch die Benutzerstudie wurde festgestellt, dass sich der Benutzer während der Verwendung der AR-Anwendung selten von seiner Position entfernt. Zudem Rotiert der Benutzer das Mobiltelefon nur über die Achsen in Handgelenk, Ellenbogen oder Hüfte. Über die Annahme, dass sich alle Benutzer so verhalten, konnte ein Bewegungsmodell erstellt werden, das erlaubt, die Rotationen des Mobiltelefons über eine Analyse des Pixel Flows zu ermitteln. Das Trackingsystem stellt sich dabei automatisch auf die vom Benutzer unterbewusst verwendeten Rotationsachsen ein. Dadurch konnte die Zufriedenheit der Benutzer mit dem Trackingsystem erheblich gesteigert werden. Durch die zweite durchgeführte Benutzerstudie wurden die Annahmen über die typischen Bewegungen der Benutzer bestätigt. Der theoretisch und praktisch einfache PSA Algorithmus, der verwendet wurde um den Pixel Flow zu ermitteln, hat sich als äußerst recheneffizientes Trackingverfahren erwiesen. Über eine Glättung der Ecken des Markers und der gesamten Markerpose konnte die Qualität des Markertrackings so gesteigert werden, dass es bei erkennen des Markers keine weitere Stabilisierung durch andere Trackingverfahren nötig war. Da der Pixel Flow für jeden Frame errechnet wurde, wurde der AR-Viewer um eine weitere Interaktionsmethode erweitert. Durch die Steuerung der Anwendung über den Pixel Flow ist es nun möglich, die Anwendung mit nur einer Hand zu bedienen. Während dieser Arbeit haben sich Mobiltelefone als brauchbare AR-Plattform erwiesen. Zur Zeit gibt es kaum eine andere Plattform die auf so kompaktem Raum AR-Anwendungen ermöglicht.

Das in dieser Arbeit entwickelte Trackingsystem hat während der Benutzerstudie sehr gut funktioniert, so lange sich der Benutzer nicht im Raum bewegt hat. Er konnte seine Arme aber frei bewegen und so das Objekt aus den Blickwinkeln betrachten, die er wollte. Er war nicht mehr gezwungen, große Kompromisse in seinem Verhalten auf Grund des Trackingsystems einzugehen. Wenn der Benutzer seine Position veränderte und die Markerpose nicht mehr stimmte, musste er nur den Marker wieder „einfangen", indem er ihn einige Sekunden betrachtete.

6. Ausblick

Die Benutzerstudie hat gezeigt, dass die Benutzer durch das neue Trackingsystem viel mehr Möglichkeiten haben und diese auch ausnutzen. Sie probierten neue Dinge aus, wodurch sie teilweise an die neuen Grenzen des Trackingsystems stießen. Eine der häufigsten Bewegungen, die die Benutzer ausführten, die nicht vom Trackingsystem unterstützt wurde, war die Translation in Z-Richtung. Würde das Trackingsystem um diese Funktionalität erweitert, könnten die meisten relevanten Benutzerbewegungen vom Trackingsystem abgedeckt werden. Zur Zeit werden die Translationen, die das Mobiltelefon bei der Bewegung durch den Benutzer erfährt vernachlässigt. Die Benutzerstudie hat gezeigt, dass der Drift, der dadurch entsteht, die Benutzer nicht stört. Um das Trackingsystem zu verbessern, könnte das Bewegungsmodell, das verwendet wird, um den Pixel Flow mit dem Markertracker zu fusionieren, ausgetauscht werden. Wird als Bewegungsmodell z.B. die Rotation des Mobiltelefons auf einer Kugeloberfläche angenommen, könnte die Translation bei Rotation um Achsen, die im Ellenbogen liegen, besser integriert werden. Während der Benutzerstudie hat sich das Trackingverfahren über den Pixel Flow als robust erwiesen. So robust, dass ein Benutzer während der Studie den Marker verdeckte und nur noch das Tracking durch den Pixel Flow verwendete. Da dies Problemlos funktionierte, könnte die Leistung des Trackingsystems gesteigert werden, indem z.B. nur für jedes zweite Bild das Markertracking durchgeführt wird. In den Frames dazwischen kann die Verfolgung der Position über den Pixel Flow kompensiert werden. Obwohl sich der Pixel Flow als robust erwiesen hat, besitzt er dennoch einige Schwachstellen. Bei der dritten Aufgabe der Benutzerstudie, in der die Bentuzer das obere drittel einer Litfaßsäule betrachten sollten, waren immer große Teile der Zimmerdecke zu sehen. Da die Zimmerdecke und die Wände meist einfarbig gestrichen und selten Objekte am oberen Ende der Wände befestigt sind, gab es Probleme mit dem Pixel Flow. In diesem Bereich wären die Beschleunigungssensoren von Vorteil. Sie eigenen sich zwar nicht, um alle Bewegungen des Mobiltelefons zu verfolgen, jedoch könnte man sie verwenden, um die Neigung des Mobiltelefons um die X-Achse der Sensoren bei aufrechter Haltung des Mobiltelefons zu ermitteln.

Anhang

A. Beispiel einer Konfigurationsdatei

ccx 60.94888
ccy 76.29551
fcx 167.79970
fcy 167.37371
BWThreshold 100
MarkerSize 150.0
RotationSegments 60
UsePixelFlowXY 1
UsePixelFlowZ 0
PixelFlowThreshold 250
MaximumShift 64
Origin 0
InterpolatePoses 0
InterpolationThreshold 100.0
InterpolationMSecDuration 250.0
UseExtrapolation 1
SlerpFactorFuture 0.5
DESPAlphaRot 0.2
DESPAlphaTrans 0.8
DESPAlphaPoints 0.2
MarkerCount 1
Marker 0x0B44

B. Antworten auf den Fragebogen aus der Benutzerstudie

Was hat Ihnen an der Anwendung gefallen?

- AR ist cool x4

Was hat Sie gestört? Allgemein

- Bedienung der Schaltflächen und das Laden schwierig mit nur dem Finger
- Steuerung unpräzise (Konzept nicht richtig verstanden, z.B. rotieren über Kreisbewegung
- Steuerung unpräzise (Konzept nicht richtig verstanden, z.B. Bild einmal oben und einmal unten angefasst und rotiert-> unterschiedliche Rotationen , Benutzer war verwirrt)
- Steuerung sollte insgesamt präziser arbeiten
- Steuerung über Pixelflow reagiert zu langsam
- Steuerung über Pixelflow hat nicht gefallen -> Programm beherrscht Benutzer
- Steuerung über Pixelflow gut für Verschieben
- Objekt schnell aus dem Blickfeld

Was hat Sie gestört? Nur Markertracking

- Marker geht oft und schnell verloren x3
- Position des Markers wackelt zu sehr x4

Was hat Sie gestört? Neues Trackingsystem

- Jitter der Position des Markers x2

Warum haben Sie das Programm auf diese weise bedient? Nur Markertracker

- Möglichst wenig Bewegungen des Handys, damit der Marker nicht verloren geht x5
  Als text schreiben

Wie würden Sie das Programm bedienen wollen?

- HMD für Darstellung
- Steuerung über “Gesten” auf dem Display, z.B. Kreis zeichnen -> rotieren, von links unten nach rechts oben ziehen -> vergrößern, Objekt direkt durch klick auswählen

Wie war der gefühlte Unterschiede zwischen Markertracker und neuem Trackingsystem?

1. Eindruck von besserer Markererkennung (Kompensation über Pixelflow)
2. Weniger Jitter

C. ISMAR Paper

Folgendes Paper entstand im Rahmen der Arbeit. Es wurde zur ISMAR Konferenz 2009 eingreicht.

Mobile Augmented Reality based 3D Snapshots

Peter Keitler* Frieder Pankratz Björn Schwerdtfeger Wolf Rödiger Gudrun Klinker
Technische Universität München
Christian Rauch† Anup Chathoth‡
Vodafone Group Services GmbH – Vodafone Group R&D Germany
John Collomosse§ Yi-Zhe Song
University of Bath

ABSTRACT

In this paper, we present a mobile augmented reality application that is based on the acquisition user-generated content obtained by 3D snapshotting. To take a 3D snapshot of an arbitrary object, a point cloud is reconstructed from multiple photographs taken by a mobile phone. From this, a textured polygon model is computed automatically. Other users can view the 3D object in the environment of their choosing by superimposing it on the live video taken by the cell phone camera. Optical square markers provide the anchor for virtual objects in the scene. To extend the viewable range and to improve overall tracking performance, a novel approach based on pixel flow is used to recover the orientation of the phone. This dual tracking approach also allows for a new single-button user interface metaphor for moving virtual objects in the scene. The Development of the AR viewer was accompanied by user studies and a further summative study evaluates the result, confirming our chosen approach.

Index Terms: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities—; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles;

1 MOTIVATION

We present a mobile augmented reality (AR) platform based on user-generated content. The core idea is to enable a user of our system to generate a 3D model of arbitrary small or mid-sized objects, based on photographs taken with their mobile phone camera. Thereupon, another user can inspect the object integrated in their natural environment, e.g. their home, using our mobile AR viewer application. We refer to this capture and viewing process as "3D Snapshotting"

3D models can be generated on-the-fly, using a pair of photographs of an object from different perspectives and reconstruct its 3D structure from them. This results in a dense 3D point cloud. However, they cannot be rendered directly, especially on mobile platforms, since graphics hardware is optimized for polygon models. One challenge for 3D snapshotting is therefore to reduce model complexity by automatically computing a textured mesh without significantly deteriorating aesthetic quality.

Another user can then inspect an object created by 3D snapshotting using the AR viewer installed on their mobile phone. It overlays the object on the live video, as shown in Figure 1. An optical square marker serves as an anchor for the object in the natural environment of the user. An extension based on pixel flow enhances the robustness of tracking and enlarges the usable viewing range.

Figure 1: Mobile AR viewer

This poses interesting questions regarding the usability of such a system. Can mobile phones perform sufficiently well to satisfy user's expectations with respect to the described AR application? How can users interact with the 3D scene most effectively? A demonstrator running on an HTC Touch Diamond has been developed to investigate those issues.

Wagner [2] and Henrysson [5] investigated various technical aspects of mobile AR applications. One of the fiercest barriers for developing AR tracking systems for mobile phones is their limited computing power. Therefore, porting existing tracking methods to mobile phones is quite challenging. A successful port of the ARToolKit marker tracking library was done by Wagner et al., released under the name ARToolKitPlus [12].

Another restriction of mobile platforms is the lack of a floating point unit (FPU). This forces the developer to either rely on a computationally expensive software implementation or to use fixed point arithmetic instead, which however introduces restrictions in the range of representable values and also in the achievable precision. Drab et al. [3] introduced a motion detection algorithm called projection shift analysis (PSA), designed particularly for mobile devices with limited resources. The algorithm computes the shift in x and y direction between consecutive frames of a video, using only integer arithmetic.

2 THE 3D RECONSTRUCTION

We reconstruct a 3D mesh model of an arbitrary object, using a pair of photographs supplied by the user. These photographs are taken under similar lighting conditions, from different points of view, and are typically captured as JPEG images using their camera phone. SIFT features [7] are identified within the photographs, and matched to identify parts of the object common to both viewpoints. The fundamental matrix, encoding the geometry between

*{keitler | pankratz | schwerdt | roediger | klinker}@in.tum.de
†christian.rauch@vodafone.com
‡Anup.Chathoth@vodafone-rnd.de
§{jpc | yzs20}@cs.bath.ac.uk

Figure 2: Steps of the reconstruction process. SIFT features are used to match images (upper left), a dense point cloud is reconstructed from matching points (bottom left), finally a textured mesh is produced (right).

the views, is recovered by analyzing the SIFT point matches via MAPSAC [10]. Camera intrinsic parameters are estimated either using the EXIF data captured in the JPEG file [11], or using a calibration rig. We input these data with standard dense reconstruction algorithms to infer the 3D positions of the matched SIFT points. Additional 3D points are recovered using dense match propagation, yielding a cloud of 5k reconstructed 3D points for a typical small or mid-sized object. We create a triangular mesh over the 3D points by backprojecting those points to their 2D positions in one of the original photographs, and performing a Delaunay triangulation. Texture for each 3D mesh triangle is cut from the corresponding 2D triangle within the original photograph. At this stage the mesh is simplified by merging adjacent mesh triangles where color and surface normal are near-identical. A graph is constructed where each node corresponds to a mesh face, and weights correspond to color and surface normal similarity. Graph cut [9] is applied to perform simplification of the mesh. Simplification is essential to reduce the rendering complexity for the mobile AR viewer. The profiling results in Table 1 show that even with simple objects, the application is mostly occupied with rendering. The important stages of the reconstruction process are depicted in Figure 2. Due to complexity of the 3D reconstruction and triangulation algorithms, the reconstruction is executed as a web service instead of on the mobile phone. Modern cellular bandwidth enables photographs and mesh geometry to be easily transferred.

3 AR Viewer Requirements

For a first informal user study, we developed an AR viewer based solely on optical square marker tracking, as described in Section 4. The main focus of the study was to observe how the participants interact with the phone and the test application, to gather requirements for the tracking system and application interface (see Section 5). The HTC Touch Diamond has a Qualcomm®MSM7201A™528 MHz CPU and runs Windows Mobile. The integrated camera provides an RGB565 video stream of 240x320 pixel at 15 fps. The 2.8 inch touch screen display has a resolution of 480x640 pixel (260 dpi).

The participants where not told how the marker tracking system worked and what flaws it had. They were just informed that a virtual object would appear on top of the marker. The intention was

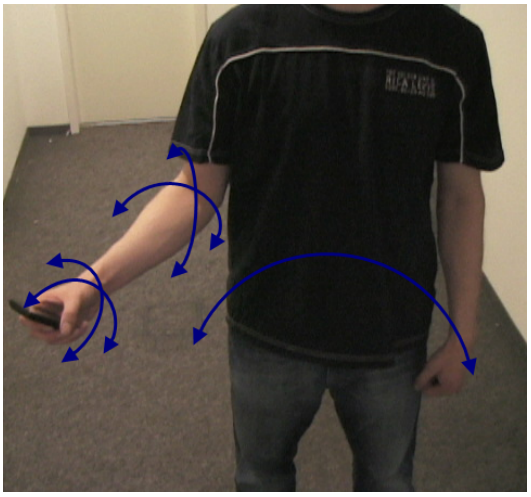


Figure 3: Commonly used rotation axes

to observe how the users would naturally interact with the system. Five participants had to arrange simple furniture objects in a room. More concretely, the tasks were to align them with real objects, orient them in a certain direction or inspect them from different perspectives.

The participants, most of them seeing an AR application for the fist time, did not know about the flaws of optical marker tracking and moved the mobile phone too fast in the beginning. This resulted in strong motion blur causing the marker tracking to fail, an effect that was even amplified by the long shutter times of the integrated phone camera.

Another big problem occurred in conjunction with big objects such as tables and chairs. Since the field of view of our phone camera is rather small, the virtual objects occluded almost the complete display, making the participant loose the context provided by the real world. When the marker was close to the participant, he had to move the phone to inspect the whole object which resulted in the loss of the marker tracking because the marker went out of scope. In case the marker was further away from the participant, the field of view wasn't as much of a problem, but the pose of the marker began to jitter, depending on its size, which the participants felt as very troublesome. Similar effects have already been reported by Freeman [4] in conjunction with ARToolKit and explained mathematically by Schweighofer [8]. In the end, the participants developed strategies to compensate for this, for example, standing completely still in front of the marker, interacting only with the methods provided by the AR viewer (see Section 5). Thus the application became less intuitive. From this first user study, we also learned that the participants preferred an inaccurate but stable pose over an unstable pose.

During the user study, we also noticed that users typically do not translate the phone but rather rotate it around axes lying in the wrist, elbow or hip, see Figure 3. This offers the opportunity for simplified pose estimation when the marker is absent. Summarizing the results, the study suggests that optical square marker tracking alone is not enough to provide a satisfactory user experience.

4 The Tracker

Based on the experiences of the formative user study described in Section 3, a tracking system suited for mobile devices was developed. The drawbacks of the marker tracker that had already been used for the study were then compensated for by integrating the efficient projection shift analysis (PSA) algorithm to recover the

orientation of the mobile phone even without a visible marker.

4.1 Marker Tracker

The tracking algorithm is functionally similar to known methods [2], we therefore concentrate on some optimization details. The first steps in the marker tracking pipeline are a conversion of the source image from color to grayscale, reduction of the resolution and a thresholding, resulting in a binary image suitable for corner detection. To improve the performance of the system, the conversion of the RGB565 formatted camera image into an 8 bit gray scale image was optimized using a look up table. The RGB565 color value of a pixel is interpreted as an unsigned short value and used as an index to get the corresponding 8 bit gray value from the look up table. 64kb additional memory is required for the look up table, but a speed up of 24% was observed. For more detailed time measurements see Table 1. Furthermore, the resolution is cut in half from 320x240 to 160x120 pixels. This allows for efficient implementation of linear interpolation by calculating the mean values of 2x2 pixel blocks for each pixel in the smaller result image. The lower resolution not only improves the performance of marker tracking, it also reduces the effects of noise and motion blur and allows more stable tracking results. The scaled grayscale image is not directly discarded since it is also used by the PSA algorithm (see below).

Then, quadrangles are found using OpenCV functions [1]. OpenCV has been partially ported to Windows Mobile for this purpose. After detecting a rectangle, its corners are refined with subpixel accuracy by fitting lines to the edges of the marker. Finding and refining the quadrangles takes up about 71% of the total time needed for marker tracking. The OpenCV functions for finding contours and approximating polygons aren't that computationally expensive and take only 23% of the needed time. 73%, however, is spent optimizing the corners of the quadrangle with sub-pixel accuracy, in particular for the interpolation between pixels for the refinement of edges. While the interpolation of one pixel value takes only 0.0027 ms, this function is called hundreds of times during one marker detection cycle.

The next steps are to detect the ID of the marker coded in the interior of the rectangle and to compute a unique marker orientation in 2D from it. Having uniquely determined the four corner points, an initial guess of the marker pose in 3D can be computed and optimized using the Levenberg-Marquardt algorithm. This takes up about 19% of the overall time needed for the marker tracker.

A fixed point implementation is used for decimal arithmetic in most cases. Floating point numbers are only used where high precision is necessary, e.g. in the pose estimation. To improve computational accuracy, it does not have a static 16.16 bit format but allows for the adaption of the number of decimal places according to the problem at hand. Quaternion operations for example highly benefit from this technique.

To reduce the jitter of the pose, a double exponential smoothing predictor[6] is used. First, the positions of the four corner points of the marker are smoothed to reduce jitter when the marker becomes relatively small. After that, the pose results from the marker tracker are extrapolated based on a linear motion model and smoothed to reduce the lag of the pose that is introduced by the smoothing of the corner points.

4.2 Pixel Flow

The projection shift analysis (PSA) algorithm, originally designed to use the motion of the phone as an interaction technique, was adapted to compute the orientation of the phone [3]. The algorithm computes the horizontal and vertical shift between two consecutive frames. First, it calculates the sums of gray values for every row and column in the current image, resulting in two vectors of accumulated gray values, one for the horizontal and one for the vertical shift. These vectors are compared with the vectors from the previous image to determine the shift of the current image. The optimal shift is found by minimizing the sum of squared differences (SSD) over all possible shifts. To increase the performance and stability of the algorithm, the SSD is only calculated for a predefined range of shifts. Tests suggest a range of 50% of the image size. This eliminates the risk of detecting huge shifts caused by similar background colors and patterns. Only integer operations are needed.

This produces stable and fine grained results most of the time but the algorithm is still vulnerable to motion blur. Thus, the PSA is invoked by default on the image with halved resolution already used by the marker tracker. Depending on the SSD value and a predefined threshold, the image is maybe scaled down again by 50% and PSA is repeated. This is done two more times, if necessary, up to one eighth of the original resolution in the worst case. This can be implemented efficiently by just scaling down the row and column intensity vector results of the previous PSA execution instead of scaling down the whole image.

4.3 Fusion

As long as the marker is being tracked by the system, the marker pose is considered to be accurate. Once the marker is lost, the horizontal and vertical shift of the current image with respect to the previous image is used to estimate the orientation change of the mobile phone around its X and Y axes. The last known marker pose is incrementally updated using these values. Translation of the phone is assumed to be negligible compared to changes in orientation (as noted in our earlier study). The conversion from shift in the image to the rotation of the phone is done by using two factors, one for the X and one for the Y axis. They describe the degree of rotation corresponding to one pixel shift. When choosing these factors, it has to be considered that depending on the objects and their distance to the camera, one degree of rotation can result in different pixel flows. Objects further away from the camera result in a higher pixel flow. To take this into account, the factors are constantly being adjusted as long as the marker is still visible. This way, the system automatically adapts to the current scene.

The advantages of this approach are that the users don't need to keep the marker in the visible area all the time. They can interact more naturally with the application and their environment, even when the marker is lost. Another advantage is that the results of the pixel flow can be used for other purposes, e.g. an alternative interaction method for the application (see Section 5).

PSA also has some drawbacks, too. If the background is flat or consists of repeating patterns (such as checker board), the it produces inaccurate results. Movements in the background are problematic, too. Since the pixel flow isn't 100% accurate, a drift will accumulate over time, resulting in an instantaneous jump of the pose as soon as the marker becomes fully visible again.

5 The Augmented Reality Viewer

The user interface of the AR viewer consists of a toolbar overlay. It contains buttons to load, select and erase virtual objects, to change settings, to toggle between view and object manipulation mode as well as several tools to arrange objects in space. In particular, this comprises the operations move, rotate, lift and scale. They were identified in our first user study described in Section 3.

Drop shadows are rendered at the bottom of the objects to improve the sense of immersion. While being tracked, the square marker is highlighted with a green selection to provide visual feedback. During user interaction with one particular object all other objects are rendered transparently to emphasize the current selection. In object manipulation mode, two different interaction techniques are provided:

- *Touch manipulation:* Once a function has been selected, the user has to drag a line on the screen with their finger. The

relative movement on the screen is used to directly manipulate the attributes of the selected object, e.g. the position in the plane defined by the marker (move) or the height (lift), using a predefined factor for each function. These factors where defined using extensive testing.

- *Flow manipulation:* Using the touch screen for user input requires both hands most of the time. Using the pixel flow provided by the tracking system, we can implement an interaction method for only one hand. First, the desired manipulation function is selected from the toolbar, preferably using the thumb. Then, instead of using the touch screen to, e.g. move an object, the user has to press a central button on the phone while moving the phone in the according direction(s). Again, the relative movement is used as input for the selected manipulation function, except that other conversion factors are used.

Early testing with some users also showed that imprecise results from the pixel flow, e.g. if downscaling was disabled, have a greater impact on the object manipulation than on the augmentation itself. While a drift of the augmentation was tolerated by most of the users, flow manipulation, for example while moving an object, is highly sensitive to errors.

6 Evaluation and Results

Based on the functionality described so far, we performed a second informal summative user study with 8 subjects. Again, the focus of the study was on the interaction of the participants with the mobile phone and the AR viewer. Like in our first user study, the participants where not told at first how the tracking system worked and what flaws it still had. This time the participants had to place different objects onto other objects, view an about 2 meters tall virtual advertising column and identify 3 posters that where placed around the top of it, thereby forcing them to use the pixel flow tracking.

The overall satisfaction when using the application increased drastically, partly because of the stabilized marker position, but mostly because the participants did not have to pay much attention to the marker and could freely move the phone without worrying about loosing it. The participants were still standing in front of the marker, rotating the phone about an axis in the wrist, elbow or body, approving our assumption about the movements of the user.

But with the freedom the new system granted, some users tried to view the top of the advertising column and move around the object at the same time. The tracking system doesn't support such movements, so the users didn't get the result they expected. After explaining to them the restrictions of the new tracking system, the users quickly adapted to it. They either went around the marker, looked at the marker for a few moments to let the system register the new position or they used the interaction methods provided by the application to rotate the advertising column.

Most of the participants didn't notice the drift at first or where not bothered by it. Only a few participants repeatedly checked whether the object was still at the marker position.

The user study showed that the flow manipulation is particularly suited for translating the object in the X/Y (move) and Z (lift) directions, but unusable for rotation since the required movement of the phone (rotating the phone around its Y axis) would make the object leave the field of view. Scaling the object using the flow manipulation is possible, but since it requires the phone to move, the point of view changes and so the users where not sure anymore to which size they wanted to scale the object since they lost their reference points.

Even though the touch manipulation was at a much better stage of implementation at this time, the users preferred the flow manipulation for moving the object. Only when the given task required precision, the users switched to the touch manipulation.

Component	*Time [ms]*
AR Viewer	100.0
Camera	12.10
Tracking system	13.0
Image conversion	2.20
Marker tracker	9.85
Find quadrangles	7.00
Find contours (OpenCV)	1.00
Approximate polygons (OpenCV)	0.63
Refine Quadrangles	5.10
Interpolate pixel values	2.26
Identify marker	0.0058
Estimate pose	1.90
Pixel flow	0.87
1/2 resolution	0.86
1/4 resolution	0.00098
1/8 resolution	0.00033
Rendering	74.91
Draw background	17.56
Draw object	57.35

Table 1: Typical time measurements for AR viewer rendering a model with 1125 triangles at 10.0 FPS

Acknowledgements

This topic has been initiated at the Vodafone Group R&D academic flagship conference in November 2007. The research activities have been supported and supervised by Vodafone Group R&D.

References

[1] G. Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[2] W. Daniel. *Handheld Augmented Reality*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, 2007.

[3] S. A. Drab and N. M. Artner. Motion detection as interaction technique for games & applications on mobile devices. In *Pervasive Mobile Interaction Devices (PERMID 2005) Workshop at the Pervasive 2005*, Munich, DR, May 2005.

[4] R. Freeman, S. Julier, and A. Steed. A method for predicting marker tracking error. 2007.

[5] A. Henrysson. *Bringing Augmented Reality to Mobile Phones*. PhD thesis, Linköping UniversityLinköping University, Department of Science and Technology, The Institute of Technology, 2007.

[6] J. J. LaViola Jr. An experiment comparing double exponential smoothing and kalman filter-based predictive tracking algorithms. In *VR '03: Proceedings of the IEEE Virtual Reality 2003*, page 283, Washington, DC, USA, 2003. IEEE Computer Society.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[8] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2024–2030, 2006.

[9] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[10] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24:271–300, 1997.

[11] N. S. University, N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM Transactions on Graphics*, pages 835–846. Press, 2006.

[12] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In H. G. Michael Grabner, editor, *Computer Vision Winter Workshop*, February 2007.

Literaturverzeichnis

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[2] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, 2008. [Online; letzter Zugriff 11 Juli 2009].

[3] Wagner Daniel. *Handheld Augmented Reality*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, 2007.

[4] Stephan A. Drab and Nicole M. Artner. Motion detection as interaction technique for games & applications on mobile devices. In *Pervasive Mobile Interaction Devices (PERMID 2005) Workshop at the Pervasive 2005*, Munich, DR, May 2005.

[5] Russell M. Freeman, Simon J. Julier, and Anthony J. Steed. A method for predicting marker tracking error. In *ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Gartner. Gartner says worldwide smartphone sales grew 16 per cent in second quarter of 2008. http://www.gartner.com/it/page.jsp?id=754112, 2008. [Online; letzter Zugriff 06 Juli 2009].

[7] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, March 2004.

[8] Anders Henrysson. *Bringing Augmented Reality to Mobile Phones*. PhD thesis, Linköping UniversityLinköping University, Department of Science and Technology, The Institute of Technology, 2007. On the day of the defence date the status on articles III and VIII was: Accepted.

[9] Peter Protzel Johannes Bauer, Niko Sünderhauf. Comparing several implementations of two recently published feature detectors. In *International Conference on Intelligent and Autonomous Systems*, 2007.

[10] MC Juan, D Joele, R Baños, C Botella, M Alcañiz, and C van der Mast. A markerless augmented reality system for the treatment of phobia to small animals. In *The 9th international workshop on presence – PRESENCE 06, Cleveland (USA)*, pages 71–74, 2006.

[11] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, page 85, Washington, DC, USA, 1999. IEEE Computer Society.

[12] Georg Klein and Tom Drummond. Robust visual tracking for non-instrumented augmented reality. In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 113, Washington, DC, USA, 2003. IEEE Computer Society.

[13] G. Klinker and F. Echtler. 3d visualization and exploration of relationships and constraints at the example of sudoku games. Technical Report TUM-I0722, Technische Universität München, Department of Computer Science, November 2007.

[14] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST-97)*, Lausanne, Switzerland, September 15–17, 1997. ACM Press.

[15] Joseph J. LaViola. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, pages 199–206, New York, NY, USA, 2003. ACM.

[16] David Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[17] Macnews. Gyroskope erobern die smartphones. http://www.macnews.de/news/113333, 2009. [Online; letzter Zugriff 06 Juli 2009].

[18] Patrick Maier. Augmented chemical reactions - small molecules with a large impact. Demonstration at IEEE International Symposium on Mixed and Augmented Reality (ISMAR'08), September 2008.

[19] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12), December 1994.

[20] Wouter Pasman, Charles Woodward, Mika Hakkarainen, and Jouko Hyväkkä Honkamaa, Petri Honkamaa. Augmented reality with large 3d models on a pda: implementation, performance and use experiences. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 344–351, New York, NY, USA, 2004. ACM.

[21] Gerald Schweighofer and Axel Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2024–2030, 2006.

[22] Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM.

[23] Tobias Sielhorst, Marco Feuerstein, and Nassir Navab. Advanced medical displays: A literature review of augmented reality. *IEEE/OSA Journal of Display Technology; Special Issue on Medical Displays*, 4(4):451–467, December 2008.

[24] Spb Software. Spb benchmark, test descriptions. http://www.spbsoftwarehouse.com/pocketpc-software/benchmark/tests_descriptions.html, 2008. [Online; letzter Zugriff 11 Juli 2009].

[25] D. Stricker, G. Klinker, and D. Reiners. A fast and robust line-based optical tracker for augmented reality applications. In *1. International Workshop on Augmented Reality (IWAR'98)*, pages 129–145, San Francisco, November 1998. AK Peters.

[26] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*, 2008.

[27] Schmalstieg Dieter Wagner Daniel. Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, February 2007.

[28] Randy Yates. Fixed-point arithmetic: An introduction. *Digital Sound Labs*, 74, 2001.

[29] Suya You and Ulrich Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)*, page 71, Washington, DC, USA, 2001. IEEE Computer Society.

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

AR-Plattform
Eine Zusammenstellung an Hardware, die alle nötigen Komponenten für Tracking, Anwendungslogik und Darstellung enthält.

AR-System
Ein Softwaresystem zusammen mit einer AR-Plattform bilden ein AR-System. Ein AR-System kann direkt von einem Benutzer verwendet werden.

Markerpose
Die Pose eines Markers.

Pose
Eine Pose beschreibt alle sechs Freiheitsgrade eins Objekts.

Trackingsystem
Ein Softwaresystem, das eines oder mehrere Trackingverfahren verwendet. Das Trackingsystem wird mit Sensordaten gespeist und liefert Daten über Position und Orientierung verschiedener Objekte.

Trackingverfahren
Ein Trackingverfahren wird definiert durch seinen Algorithmus. Es ist festgelegt, welche Art an Sensordaten für den Algorithmus notwendig ist, wie die prinzipielle Verarbeitung der Daten abläuft und welche Ergebnisse der Algorithmus liefert.

Abkürzungsverzeichnis

AR	Augmented Reality
DESP	Double Exponential Smoothing Predictor
EKF	Extended Kalman Filter
FSBM	Full Search Block Matching Algorithm
HMD	Head Mounted Display
LUT	Look-Up Table
PSA	Projection Shift Analysis
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features