FLUIDUM

Fluid Beam
Konzeption und Realisierung eines Display-Kontinuums mittels einer steuerbaren Projektor-Kamera-Einheit

Diplomarbeit im Rahmen des Projekts FLUIDUM

Fakultät für Informatik

Universität des Saarlandes

von

Lübomira Spassova

Betreuer: Prof. Dr. Andreas Butz

24. September 2004

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Lübomira Spassova Saarbrücken, den 24. September 2004

Danksagung

An dieser Stelle möchte ich mich bei den Menschen bedanken, deren Unterstützung die Entstehung dieser Arbeit ermöglicht hat. Den Mitgliedern der Nachwuchsforschergruppe FLUIDUM und allen Mitarbeitern des Lehrstuhls von Prof. Wahlster danke ich sowohl für die vielen fachlichen Tipps als auch für die zahlreichen Flur-Frühstücke und andere soziale Events, an denen ich teilnehmen durfte. Ganz besonderer Dank gilt meiner Familie, die mich während meines gesamten Studiums sowohl finanziell als auch moralisch unterstützt hat. Nicht zuletzt bedanke ich mich auch bei denjenigen, die sich die Mühe gemacht haben, die ersten Versionen dieser Arbeit Korrektur zu lesen, und mir viele nützliche Hinweise geben konnten.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit spielt Visualisierung eine immer stärkere Rolle. Sowohl an öffentlichen Plätzen als auch im privaten und beruflichen Umfeld sind Displays aller Art aus unserem täglichen Leben nicht mehr wegzudenken. Bedingt durch den ständig wachsenden technologischen Fortschritt zeichnet sich ein Trend zu immer größeren und flacheren Computermonitoren und Fernsehbildschirmen ab. Zur Zeit wird an der Entwicklung von hauchdünnen, flexiblen Displays (Paper-like Displays [4]) geforscht. Mit ihrer Hilfe sollen in Zukunft instrumentierte Räume realisiert werden, deren Wände vollständig mit einer Art elektronischer Tapete bedeckt sind. Der ganze Raum wird somit in ein kontinuierliches Display verwandelt; man spricht daher von einem Display-Kontinuum. Solange die Technologie zur Realisierung der elektronischen Tapete noch nicht ausgereift ist, stellt die Verwendung von Projektion die einzige Alternative dar, um Informationen auf beliebigen Flächen im Raum anzuzeigen.

Vor allem im Bereich des so genannten Ubiquitous Computing [1] werden Projektoren verstärkt als visuelle Ausgabemedien eingesetzt. Der Begriff des Ubiquitous Computing wurde bereits 1988 von Mark Weiser [2] geprägt und bezeichnet die Allgegenwärtigkeit von Informationstechnik und Rechenleistung [3]. Die Rechner der Zukunft sollen weitestgehend in die Umgebung integriert werden und somit für den Benutzer als solche nicht mehr wahrnehmbar sein. Der Benutzer soll die Funktionalitäten einer instrumentierten Umgebung intuitiv bedienen können, ohne das Gefühl zu haben, mit einem Rechner zu arbeiten. Durch ihre phy-

sikalische Entkoppelung von erzeugendem Gerät und erzeugtem Bild erfüllen Projektoren diese Bedingung des Ubiquitous Computing, da sie so angebracht werden können, dass sie für den Benutzer unsichtbar bleiben. Darüber hinaus können gewöhnliche Gegenstände durch Projektion instrumentalisiert werden. So kann z.B. eine Wand oder eine Schranktür in ein Display verwandelt werden, indem ein beliebiges Bild darauf projiziert wird.

2 Grundlagen

2.1 Aufgabenstellung

Das Ziel dieser Diplomarbeit war es, ein System zu entwickeln und zu implementieren, das Projektionen auf beliebige Flächen in einer instrumentierten Umgebung realisiert. Dadurch sollte es möglich sein, ein Display-Kontinuum im gesamten Raum zu erzeugen. Der Benutzer sollte in der Lage sein, mit den projizierten Displays zu interagieren. Darüber hinaus sollten geeignete Anwendungen konzipiert und implementiert werden, in denen das System zum Einsatz kommen konnte.

Zunächst musste die geeignete Hardware für die Realisierung des Systems ausgesucht werden. Dabei mussten die Vor- und Nachteile von ähnlichen bereits existierenden Systemen berücksichtigt werden.

Für die praktische Umsetzung der erarbeiteten Verfahren mussten verschiedene Kalibrierungsprobleme gelöst werden.

Schließlich musste das System in die Java-Architektur der instrumentierten Umgebung integriert werden. Dadurch sollte es von beliebigen Rechnern aus ferngesteuert werden können.

2.2 Verwandte Arbeiten

Zum Thema "bildentzerrende Verfahren" wurden in den letzten Jahren mehrere Arbeiten veröffentlicht. Im Folgenden werden drei davon vorgestellt, die einen Überblick über die Thematik liefern.

2.2.1 "Smarter Presentations"

In [5] wird ein Verfahren zur automatischen Trapezkorrektur des projizierten Bildes mittels einer nicht kalibrierten Kamera mit niedriger Auflösung vorgestellt. Die Kamera muss so platziert sein, dass die gesamte (rechtwinklige) Projektionsfläche überblickt wird. Das zu projizierende Bild wird entsprechend vorverzerrt, so dass es auf der Projektionsfläche rechtwinklig und möglichst groß erscheint. Darüber hinaus bietet das Programm die Möglichkeit, einen Laserpointer oder einen Zeiger, der farblich mit dem Hintergrund kontrastiert, als eine Art Mauszeiger zu verwenden, um virtuelle Schaltflächen zu betätigen oder auf den projizierten Folien zu zeichnen.

Um die richtige Verzerrungsmatrix P zu bestimmen, müssen die Transformationen - so genannte Homographien - zwischen Ausgangsbild- und Kamerabildebene (Matrix T) sowie zwischen Projektionsfläche und Kamerabildebene (Matrix C) berechnet werden. Daraus ergibt sich die Transformation zwischen Originalbild und Projektionsfläche als $P = C^{-1}T$ (siehe Abb. 2.1 links). Um T zu berechnen werden bestimmte Kalibrierungsmuster projiziert und einzelne Punkte im Kamerabild lokalisiert, aus deren Position die Matrix T bestimmt wird. Zur Bestimmung der Matrix C werden mit Hilfe eines Bilderkennungsverfahrens die Umrisse der Projektionsfläche im Kamerabild lokalisiert. Aus der Lage der vier Eckpunkte der Projektionsfläche im Kamerabild ergibt sich die Transformation C.

Die Inverse P^{-1} der so berechneten Transformationsmatrix P wird nun zur Verzerrung des Ausgangsbildes verwendet. Damit die Projektionsfläche optimal genutzt wird, wird das Bild zusätzlich mit einer geeigneten Matrix S skaliert. Auf diese Weise wird ein größtmögliches vorverzerrtes Bild erzeugt, das auf der Projektionsfläche rechtwinklig erscheint (siehe Abb. 2.1 rechts).

Dieselben Transformationsmatrizen, die für die Trapezkorrek-

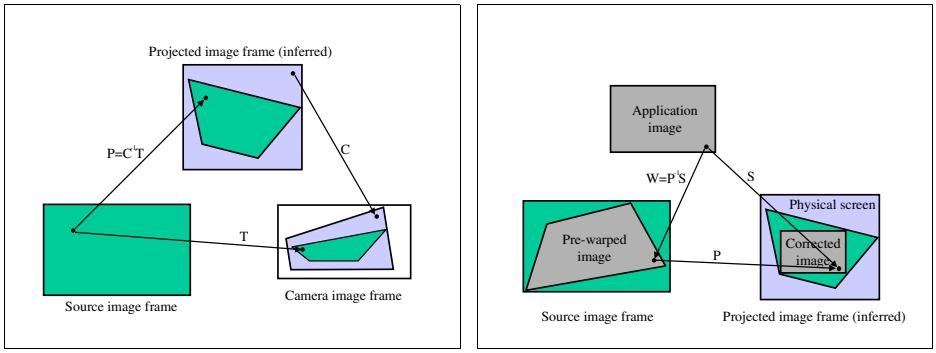

Abbildung 2.1: links: Transformation T zwischen Ausgangsbild (source image frame) und Kamerabild (camera image frame) und Transformation C zwischen Projektionsfläche (projected image frame) und Kamerabild; daraus ergibt sich die Abbildung zwischen Ausgangsbild und projiziertem Bild $P = C^{-1}T$.
rechts: Vorverzerrung des Ausgangsbildes durch Anwendung der Transformation $W = P^{-1}S$, wobei S eine Skalierungsmatrix ist und P die zuvor berechnete Transformation. (Bildquelle: [5])

tur berechnet wurden, werden auch zur Bestimmung der Zeigerposition im projizierten Bild verwendet. Der Laserpunkt bzw. die Zeigerspitze wird im Kamerabild lokalisiert und in die entsprechende Position im Ausgangsbild transformiert.

Der entscheidende Vorteil der oben beschriebenen Vorgehensweise ist, dass keine Kamerakalibrierung notwendig ist. Die einzige Bedingung, die erfüllt sein muss, ist, dass die Umrisse der Projektionsfläche im Kamerabild erkennbar sind. Das bedeutet, dass zum einen die mögliche Projektionsfläche klar definiert sein muss und zum anderen die Kamera so platziert werden muss, dass sie die gesamte Fläche überblickt. Dadurch ist das System unflexibel, da man oft keine klar definierte Projektionsfläche zur Verfügung hat, sondern an eine Raumwand projiziert. In diesem Fall kann man dieses Verfahren nicht verwenden.

Abbildung 2.2: Everywhere Displays Projektor
(Bildquelle: www.research.ibm.com)

2.2.2 "Everywhere Displays" Projektor

Das Everywhere Displays Projekt von Claudio Pinhanez [6] lieferte die maßgebliche Motivation für diese Diplomarbeit. Der Everywhere Displays Projektor - im Folgenden auch ED-Projektor genannt - besteht aus einem lichtstarken LCD-Projektor und einem Spiegel, der an einer computergesteuerten Drehvorrichtung befestigt ist (siehe Abb. 2.2). Mit Hilfe dieses schwenkbaren Spiegels kann der Projektorstrahl in beliebige Richtungen innerhalb seines Drehbereiches abgelenkt werden. Wenn der ED-Projektor in der oberen Ecke eines Raumes montiert wird, können auf diese Weise die beiden gegenüberliegenden Wände, die Hälfte der beiden anliegenden Wände und nahezu der gesamte Boden als Projektionsfläche genutzt werden.

Natürlich muss auch bei der Verwendung des ED-Projektors das Ausgangsbild korrigiert werden, damit es auf der entspre-

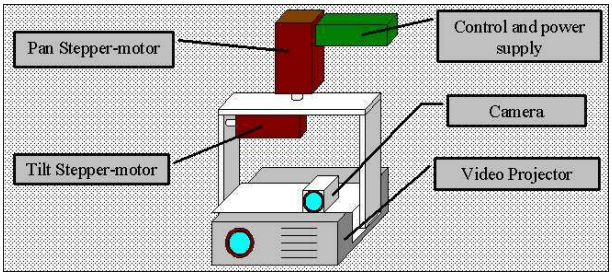

Abbildung 2.3: steuerbarer Videoprojektor (SVP) bestehend aus: Schrittmotoren für horizontale und vertikale Drehung (rot), Kontrolleinheit (grün), Kamera und Projektor (Bildquelle: [8])

chenden Projektionsfläche nicht verzerrt erscheint. Hier wird jedoch ein anderes Verfahren als das oben beschriebene verwendet. Die Methode basiert auf der Tatsache, dass aus geometrischer Sicht ein Projektor und eine Kamera mit den gleichen optischen Eigenschaften (gleicher Bildwinkel) identisch sind. Dieses Verfahren wird später ausführlich beschrieben (siehe Kap. 3.1).

2.2.3 "Steerable Video Projector" (SVP) und "Portable Display Screen" (PDS)

Bei dem in [8] beschriebenen System wird der Strahl des Projektors nicht wie in [6] mit Hilfe eines Spiegels in eine bestimmte Richtung abgelenkt, sondern der ganze Projektor, der in einer Drehvorrichtung befestigt ist, wird zur jeweiligen Projektionsfläche bewegt. Dieser steuerbare Projektor wird im Folgenden SVP (Steerable Video Projector) genannt (siehe Abb. 2.3). Über dem Projektor ist eine Kamera befestigt, die das projizierte Bild aufnimmt und somit eine Kalibrierung und Interaktionen ermöglicht. Die Einheit ist an der Decke des Raumes montiert und kann horizontal und vertikal gedreht werden. Zusätzlich sind in jeder der vier oberen Raumecken steuerbare Kameras installiert.

In dem Artikel werden zwei Kalibrierungsverfahren vorgestellt:

- offline Lokalisierung potenzieller stationärer Projektionsflächen
- online Lokalisierung einer tragbaren Projektionsfläche

Für das erste Verfahren werden der SVP und eine der stationären Kameras benötigt. Mit Hilfe des steuerbaren Projektors werden rechtwinklige Muster auf die zu untersuchenden Flächen projiziert. Im Kamerabild wird auf Grund der Verzerrung des projizierten Bildes festgestellt, ob es sich um eine stetige ebene Fläche handelt. Eine Unterbrechung einer projizierten Linie deutet z.B. auf eine Kante auf der Projektionsfläche hin und eine Krümmung der Linie auf eine gekrümmte Projektionsfläche. Wenn geeignete Projektionsflächen erkannt werden, wird durch Berechnung der entsprechenden Homographien ähnlich zu [5] eine geeignete Vorverzerrungsmatrix bestimmt. Diese wird zusammen mit den aktuellen Drehwinkeln gespeichert und später zum Entzerren des Ausgangsbildes verwendet. Mit dieser Methode können jedoch nur Projektionsflächen erkannt werden, die mindestens so groß sind wie das projizierte Muster.

Um kleine bewegte Projektionsflächen zu lokalisieren, hat man ein zweites Verfahren entwickelt, das die SVP-eigene Kamera benutzt. Die tragbare Projektionsfläche PDS (Portable Display Screen) (siehe Abb. 2.4) muss einen hellen Hintergrund haben, der dunkel umrandet ist. Dieser dunkle Rahmen wird im Kamerabild extrahiert und auf Grund seiner Form die entsprechende Vorverzerrung berechnet.

Dadurch ist es möglich, Bilder von einem stationärem Display zum anderen zu transportieren. Dazu wird in der oberen rechten Ecke der Projektion ein rotes Viereck eingefügt. Hält man die tragbare Projektionsfläche über dieses Viereck wird sie erkannt und das Bild wird von nun an auf das PDS projiziert.

Abbildung 2.4: tragbare Projektionsfläche (PDS) (Bildquelle: [8])

2.3 Offene Probleme

Die im vorangegangenen Abschnitt beschriebenen Arbeiten geben einen Einblick in die Entwicklung von bildentzerrenden Projektionsverfahren.

Das Projekt "Smarter Presentations" [5] beginnt zunächst mit einem stationären Projektor, der sich mittels einer Kamera selbst kalibrieren kann, d.h. die Projektionsfläche wird automatisch erkannt und das Bild entsprechend darauf angepasst. Die Umrisse der Projektionsfläche müssen jedoch klar definiert (rechtwinklig) und im Kamerabild sichtbar sein. Diese Bedingung macht das Verfahren unflexibel, da man in der Regel in einer instrumentierten Umgebung Projektionsflächen beliebiger Form und Ausdehnung hat, beispielsweise Wände, runde Tischoberflächen usw.

Beim Everywhere Displays-Projektor [6] wird dieser Nachteil durch die Erstellung eines 3D-Modells der Umgebung behoben. Ein weiterer Pluspunkt dieses Ansatzes ist, dass durch die Verwendung einer virtuellen Kamera die zeitaufwändigen Berechnungen von Transformationsmatrizen entfallen. Das Vorverzerren des Ausgangsbildes wird in diesem Fall von der Graphikhardware übernommen, was sehr effizient ist. Die Möglichkeit, den Projektorstrahl durch einen beweglichen Spiegel auf beliebige Flächen

zu richten, stellt einen weiteren Fortschritt dar. Jedoch bleibt die Reichweite des ED-Projektors innerhalb eines Kegels beschränkt, bedingt durch den maximalen Drehwinkel zwischen Projektionsstrahl und Spiegel.

Um in möglichst viele Richtungen im Raum projizieren zu können, wurde der steuerbare Videoprojektor SVP [8] entwickelt. Die zwei Freiheitsgrade der Dreheinheit ermöglichen - je nach Installation der Vorrichtung - eine Projektion auf nahezu jede Fläche in der Umgebung. Jedoch wird die Bildentzerrung beim SVP ähnlich wie bei [5] mittels Homographien realisiert.

In dieser Diplomarbeit wird ein Verfahren vorgestellt, das die Ansätze aus [6] und [8] kombiniert. Ein Projektor und eine Kamera befinden sich in einer Dreheinheit ähnlich der in [8]; zur Bildentzerrung wird das in [6] beschriebene Verfahren mit einer virtuellen Kamera angewendet.

2.4 Hardware

Zunächst haben wir uns dafür entschieden, den steuerbaren Projektor aus einzelnen Komponenten zusammenzusetzen. Die mir von der *FLUIDUM* Gruppe [9] zur Verfügung gestellte Hardware besteht aus einem lichtstarken Multimedia-Projektor und einer hochauflösenden Digitalkamera, die in einer Dreheinheit angebracht sind. Die Dreheinheit ist kopfüber an der Decke des instrumentierten Raums montiert.

Desweiteren wurde das in dieser Arbeit vorgestellte System auch mit einer anderen steuerbaren Einheit (*BeaMover*) verwendet, die als Gesamtpaket bestehend aus Drehvorrichtung und Projektor von der Firma *publitec Präsentationssysteme & Eventservice GmbH [10]* angeboten wird. Die einzelnen Geräte werden im Folgenden beschrieben.

Abbildung 2.5: Moving Yoke(Bildquelle: [7])

2.4.1 Dreheinheit

Die von der Firma *Amptown Lichttechnik GmbH* [7] unter dem Namen *Moving Yoke* hergestellte Dreheinheit besteht aus einer Basisplattform und einem drehbaren Bügel (siehe Abb. 2.5). In diesem Bügel kann ein Projektor platziert werden, der dann um zwei Achsen gedreht werden kann.

Um die horizontale Achse (*Pan*) kann die Einheit um ca. 340° gedreht werden, um die vertikale (*Tilt*) um ca. 270°. Die Bewegungen der Dreheinheit werden über 4 DMX Kanäle gesteuert. Dazu wurde zwischen Rechner und Dreheinheit ein USB/DMX-Interface geschaltet, das die entsprechenden Befehle weiterleitet (siehe Abb. 2.6).

DMX ist ein 8 bit Protokoll, das normalerweise im Bühnenbereich zur Lichtsteuerung (Dimmer) eingesetzt wird. Für die Steuerung der Dreheinheit werden jeweils 2 Kanäle für eine Drehachse verwendet, wodurch man eine 16 bit Auflösung erreicht. Somit ist eine sanfte Bewegung des Projektors möglich.

Die Genauigkeit der eingestellten Position wird durch die me-

Abbildung 2.6: USB/DMX-Interface
(Bildquelle: www.bullight.de)

chanischen Eigenschaften der zwei Motoren und der Getriebe bestimmt. Bei einer Entfernung von ca. 2 m zwischen Projektor und Projektionsfläche kann die Ungenauigkeit 1-2 cm betragen, weshalb keine exakte Positionierung des projizierten Bildes möglich ist.

2.4.2 Projektor

Bei der Wahl des Projektors müssen mehrere Bedingungen beachtet werden:

- Er muss die richtigen Maße haben, damit er in der Dreheinheit angebracht werden kann.
- Er soll möglichst lichtstark sein, damit man das projizierte Bild auch bei normalem Tageslicht sehen kann.
- Es muss ein LCD und kein DLP Projektor sein, da bei der DLP Technologie die drei Grundfarben (Rot, Grün und Blau) nacheinander projiziert werden, d.h. ein Bild wird aus drei aufeinander folgenden Einzelbildern zusammengesetzt. Wird ein DLP Projektor bewegt, werden diese Einzelbilder gegeneinander verschoben, und dadurch werden Farbränder in der Projektion sichtbar. Dieses Problem besteht bei einem LCD Projektor nicht, da alle Farben eines zu projizierenden Bildes gleichzeitig erzeugt werden.

- Alle Funktionen des Projektors (Ein-/Ausschalten, Fokus, Zoom,...) sollten auch über einen Rechner ansteuerbar sein.

Ein Gerät, das alle diese Eigenschaften in sich vereint, ist der *Sharp XG-P10XE* LCD-Projektor. Er projiziert mit einer Lichtstärke von 3000 ANSI Lumen und kann über eine RS-232C Schnittstelle an einen Rechner angeschlossen werden. Während der Arbeit mit dem Projektor wurde jedoch festgestellt, dass die ferngesteuerten Fokuseinstellungen nicht reproduzierbar sind, d.h. wenn man zweimal denselben Fokuswert einstellt, kann der tatsächliche Fokusabstand unterschiedlich sein. Dadurch wird eine automatische Fokussierung des Projektors unmöglich.

2.4.3 Kamera

Die an der Dreheinheit angebrachte Digitalkamera *PowerShot S45* von *Canon* hat eine Auflösung von 4 Megapixel und einen 3-fachen optischen Zoom. Zusätzlich kann sie ein kontinuierliches Videosignal mit 15 Bildern pro Sekunde liefern. Dieses wird mit Hilfe einer Framegrabber-Karte ausgelesen. Die Kamera wird über USB- bzw. Videokabel an den PC angeschlossen und kann mit Hilfe der von der Firma *Canon* im Internet zur Verfügung gestellten Software vom Rechner aus bedient werden.

2.4.4 BeaMover

Das Gerät *BeaMover* besteht aus einer Dreheinheit (*Moving Yoke*), in der ein Projektor (*Sanyo PLC-XP41*) mit 3.300 ANSI Lumen montiert ist. Zusätzlich haben wir auch am *BeaMover* unter dem Projektorobjektiv eine Digitalkamera (wie in Abschnitt 2.4.3 beschrieben) befestigt. Der wesentliche Unterschied zu unserer ersten Installation besteht darin, dass beim *BeaMover* nicht nur die Bewegungen der Dreheinheit, sondern auch alle wesentlichen Funktionen des Projektors (Fokus, Zoom, Trapezkorrektur, Ein-

und Ausschalten usw.) über ein DMX-Interface gesteuert werden können. Außerdem verfügt der BeaMover über eine höhere Positionierungsgenauigkeit als die in Abschnitt 2.4.1 beschriebene Dreheinheit, und der Fokus des Projektors kann sehr präzise eingestellt werden, was bei dem von uns zusammengestellten Gerät nicht der Fall war.

3 Lösungsansatz

Im Folgenden werden die im Rahmen dieser Diplomarbeit entwickelten Verfahren zur Lösung der gestellten Aufgabe beschrieben. Dabei liegt der Schwerpunkt dieses Kapitels in der Erläuterung der allgemeinen Konzepte; in Kapitel 4 wird auf die praktische Implementierung eingegangen.

3.1 Entzerrung des projizierten Bildes

Um auf beliebige Flächen im Raum projizieren zu können, kann man die Tatsache ausnutzen, dass der Prozess der Bildaufnahme mit einer Kamera eine Umkehrung der Projektion ist, unter der Annahme, dass Kamera und Projektor dieselben optischen Eigenschaften haben.

Um sich dieses Phänomen zu veranschaulichen, kann man folgendes Gedankenexperiment durchführen: Man stelle sich ein rechteckiges Bild (Ausgangsbild) an einer Wand vor, das mit einer Kamera aus einer bestimmten Position fotografiert wird. Falls das Bild nicht frontal aufgenommen wird, sind die Umrisse des Ausgangsbildes im Kamerabild nicht mehr rechtwinklig sondern verzerrt. Könnte man nun die Kamera durch einen Projektor mit den gleichen optischen Eigenschaften (Bildwinkel) an genau der gleichen Position mit genau derselben Ausrichtung ersetzen und das aufgenommene Kamerabild an die Wand zurückprojizieren, würden sich die Kamera- und die Projektionsverzerrung gegenseitig auslöschen und man hätte eine Projektion, die sich exakt

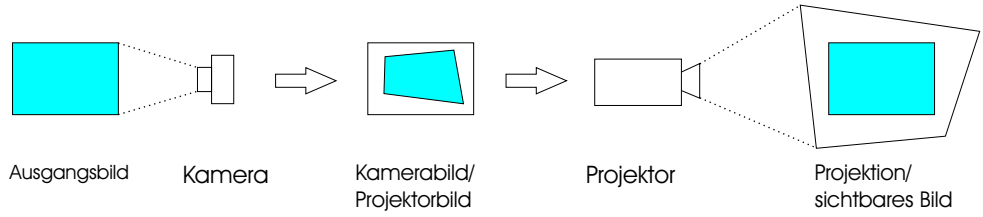

Abbildung 3.1: Gegenseitige Auslöschung von Projektions- und Kameraverzerrung

mit dem Ausgangsbild decken würde (siehe Abb. 3.1).

In der Praxis ist es natürlich nicht möglich das Objekt, das projiziert werden soll, mit einer Fotokamera aus allen Projektor-Perspektiven zu fotografieren. Zum einen wäre es sehr aufwändig, die Position und den Bildwinkel der Kamera exakt an die des Projektors anzupassen, und zum anderen müsste man theoretisch unendlich viele Bilder aufnehmen. Aus diesem Grund verwendet man statt der normalen Fotokamera eine virtuelle Kamera, die sich in einem 3D-Modell der realen Umgebung befindet und optisch und räumlich den Projektor modelliert, d.h. sie besitzt die gleichen optischen Parameter und ihre Position im 3D-Modell entspricht der Position des Projektors im realen Raum.

Auf diese Weise erhält man eine virtuelle Schicht, die den realen Raum überzieht, auf der virtuelle Objekte an bestimmte Positionen platziert werden können. Diese Objekte werden sichtbar, sobald der Projektorstrahl auf die entsprechende Stelle im Raum gerichtet wird.

3.2 Kalibrierung der Projektorposition

Damit wir das oben beschriebene Verfahren zur Bildentzerrung in der Praxis anwenden können, müssen wir zur Platzierung der virtuellen Kamera in unserem 3D-Modell die genaue Position des Projektors im instrumentierten Raum kennen. Aus physikalischen

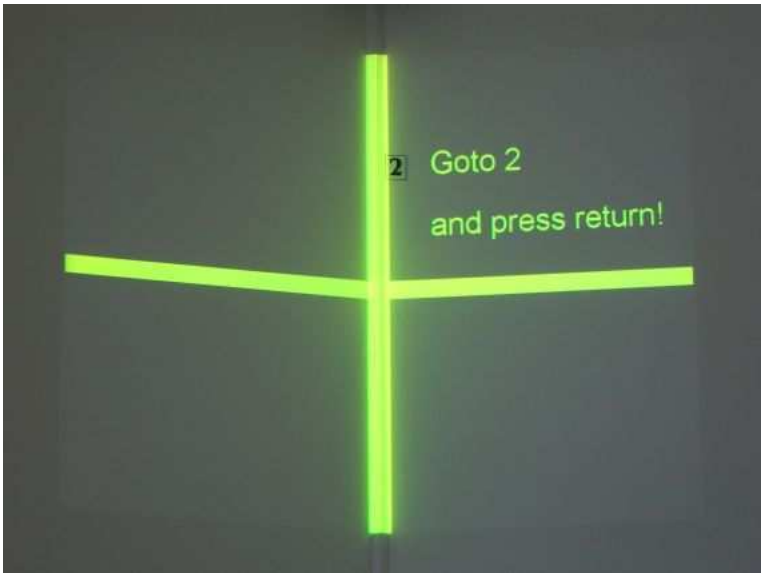

Abbildung 3.2: Kalibrierungsmuster

Gründen ist es nicht möglich, die Projektorposition durch einfaches Abmessen zu bestimmen, da wir nicht wissen, wo genau das optisches Zentrum bzw. der Drehpunkt des Projektors liegen. Aus diesem Grund haben wir ein Verfahren entwickelt, mit dem man den Drehpunkt des Projektors bestimmen kann, und das folgendermaßen funktioniert: Ein kreuzförmiges Muster wird nacheinander in drei Raumecken an Stellen, die in einer Ebene liegen, projiziert (siehe Abb. 3.2).

Aus der Änderung der Positionseinstellungen (DMX-Werte) der Dreheinheit bei der Bewegung von einer Raumecke zur nächsten wird der bei dieser Drehung überstrichene Winkel abgeleitet. Somit erhält man zwei Winkel (α und β), aus denen man durch Lösen eines Gleichungssystems die Position des Projektors in der betrachteten Ebene bestimmen kann (siehe Abb. 3.3).

Aus der in Abb. 3.3 dargestellten Messsituation lässt sich folgendes Gleichungssystem ableiten:

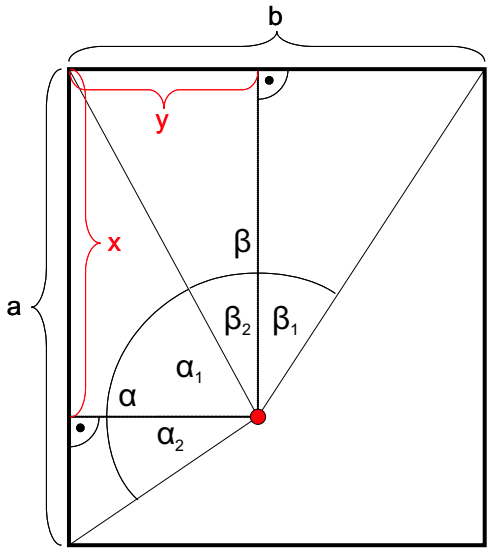

Abbildung 3.3: Kalibrierungsmessung in der xy-Ebene: Die Längen a und b sind bekannt, die Winkel α und β werden gemessen, daraus werden x und y berechnet.

$$
\begin{aligned}
\tan\alpha_1 &= \frac{x}{y} \\
\tan\alpha_2 &= \frac{a-x}{y} \\
\tan\beta_1 &= \frac{b-y}{x} \\
\tan\beta_2 &= \frac{y}{x} \\
\alpha_1 + \beta_2 &= \frac{\pi}{2} \\
\alpha_1 + \alpha_2 &= \alpha \\
\beta_1 + \beta_2 &= \beta
\end{aligned}
$$

Die geschlossene Lösung des obigen Gleichungssystems für x und y lautet:

$$
\begin{aligned}
x &= \frac{-ab\tan(\beta)(a - b\tan(\alpha))(\tan(\alpha)\tan(\beta) - 1)}{d} \\
y &= \frac{ab\tan(\alpha)(a\tan(\beta) - b)(\tan(\alpha)\tan(\beta) - 1)}{d} \\
d &= a^2\tan(\beta)^2 + b^2\tan(\alpha)^2 + (a^2 + b^2)\tan(\alpha)^2\tan(\beta)^2 \\
&\quad -2ab(\tan(\alpha)^2\tan(\beta) + \tan(\alpha)\tan(\beta)^2)
\end{aligned}
$$

Diese Messungen werden jeweils in der xy-, xz- und yz-Ebene durchgeführt, wodurch man je zwei Werte für die gesuchten Koordinaten des Drehpunktes D = $(x, y, z)^T$ erhält. Idealerweise sind die beiden x-, y- und z-Werte jeweils gleich und somit der Drehpunkt eindeutig bestimmt. In der Praxis weichen die Werte auf Grund von Messfehlern leicht voneinander ab. In diesem Fall erhält man anstatt eines einzelnen Punktes einen Quader, in dem sich der gesuchte Drehpunkt befindet. Die Größe des Quaders ist ein Maß für Größe des Messfehlers. Als beste Näherung an den tatsächlichen Drehpunkt nimmt man den Mittelpunkt des Quaders.

Die so berechneten Koordinaten werden in einer Datei gespeichert und zum Positionieren der virtuellen Kamera bei der Erstellung des 3D-Raummodells benutzt.

3.3 Das 3D-Raummodell

Das 3D-Modell des instrumentierten Raums wird so angelegt, dass der Koordinatenursprung in einer Raumecke liegt. Dadurch ist die spätere Platzierung der virtuellen Objekte für den Benutzer einfacher, da die Abstände für die Positionierung der Objekte bezüglich dieser Raumecke angegeben werden können. Wenn man nicht die Raumecke, sondern z.B. die virtuelle Kamera im Ursprung des Koordinatensystems legen würde, müsste man die

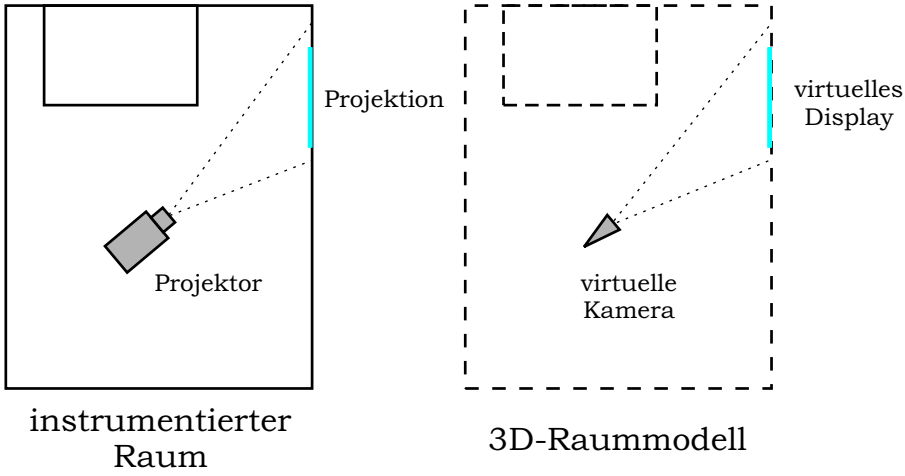

Abbildung 3.4: 3D-Raummodell

virtuellen Objekte bezüglich der Kameraposition platzieren, was nicht intuitiv wäre.

Die Wände und Tischoberflächen selbst werden nicht als 3D-Objekte dargestellt, sondern nur ihre Koordinaten werden festgelegt. Somit erhält man die Ebenen, auf denen virtuelle Objekte erzeugt werden können (siehe Abb. 3.4). Die virtuelle Kamera wird an der Position erzeugt, die mit Hilfe der Projektorkalibrierung aus Abschnitt 3.2 berechnet wurde und der Lage des Projektors im realen Raum entspricht.

3.4 Kamerakalibrierung

Die am steuerbaren Projektor befestigte Digitalkamera wird zur Marker- bzw. Gestikerkennung eingesetzt. Dazu muss die Transformation zwischen Projektor- und Kamerakoordinaten bekannt sein. Um diese Transformationsmatrix zu bestimmen, kann man folgendermaßen verfahren: Ein Referenzobjekt (Marker) wird in einer bekannten Entfernung vom Projektor platziert. Da die Position des Markers im Raum fest ist und die des Projektors mit

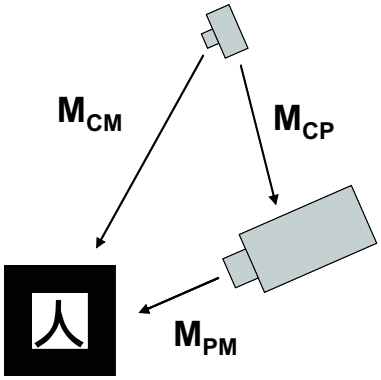

Abbildung 3.5: Kamerakalibrierung

der in Abschnitt 3.2 beschriebenen Kalibrierung bestimmt werden kann, ist die Transformation zwischen Projektor und Marker (M_{PM}) bekannt. Mit Hilfe einer Markererkennungssoftware (*z.B. JARToolkit* [11]) kann die Transformationsmatrix zwischen Kamera und Marker (M_{CM}) bestimmt werden. Daraus ergibt sich die Transformation zwischen Kamera und Projektor als M_{CP}=$M_{PM}^{-1}M_{CM}$ (siehe Abb. 3.5).

3.5 Anpassung des Bildwinkels der virtuellen Kamera

Wie bereits in Abschnitt 3.1 erläutert wurde, müssen die Bildwinkel der virtuellen Kamera und des verwendeten Projektors übereinstimmen, um eine exakte Bildentzerrung zu erreichen. Um den Bildwinkel des Projektors zu berechnen, wird folgende Messung vorgenommen: Der Projektor wird in Abstand a senkrecht zu einer Wand aufgestellt, dann wird die Breite b des projizierten Bildes gemessen. Daraus ergibt sich der Bildwinkel α als

$$\alpha = 2\arctan(\frac{b}{2a})$$

(siehe Abb. 3.6).

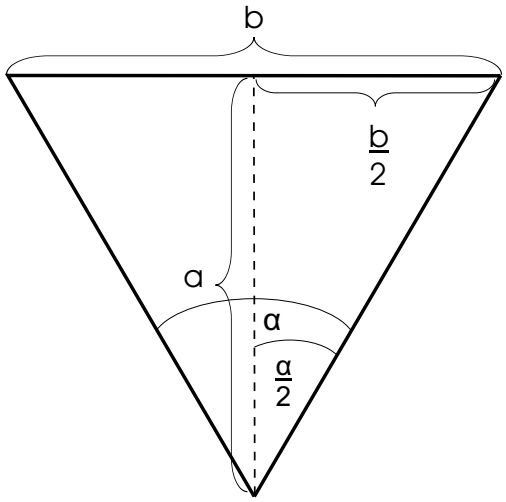

Abbildung 3.6: Bildwinkelmessung: $\tan(\frac{\alpha}{2}) = \frac{b}{2a}$

3.6 Berücksichtigung des vertikalen Lens Shifts

Manche Projektoren verfügen über eine so genannte *Lens Shift*-Funktion, mit der das Objektiv mechanisch aus der optischen Achse des Projektors bewegt werden kann. Dadurch wird die Projektion unter Beibehaltung der Bildgeometrie (d.h. ohne Trapezverzerrung) horizontal bzw. vertikal verschoben. Der horizontale Lens Shift ist eher selten und ermöglicht eine Positionierung des Projektors außerhalb der Mittelachse des Raumes, z.B. wenn diese durch eine Säule oder ähnlichem verstellt ist. Der vertikale Lens Shift wird oft verwendet, um den Projektor auf einem Tisch platzieren zu können, ohne dass das Bild zu tief projiziert wird (siehe Abb. 3.7).

Obwohl ein vertikaler Lens Shift bei einer herkömmlichen Nutzung des Projektors von Vorteil sein kann, ist er bei der Fluid Beam-Anwendung nicht erwünscht, da er die optischen Eigenschaften des Projektors verändert und diese somit nicht mehr mit denen der virtuellen Kamera übereinstimmen.

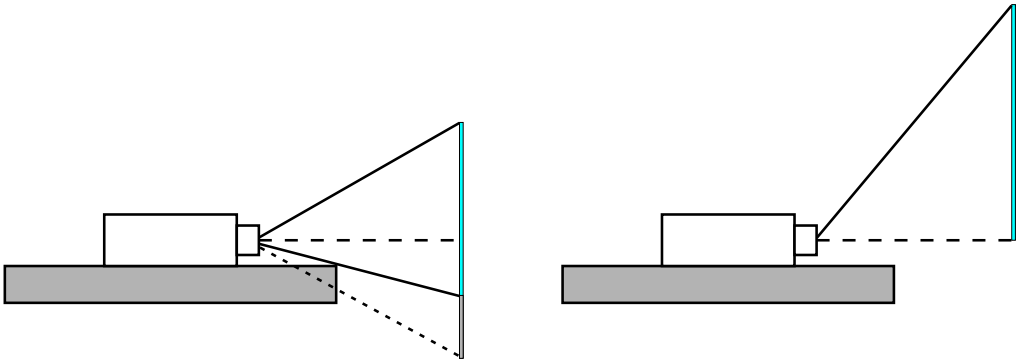

Abbildung 3.7: links: Projektion ohne Lens Shift, von Tischkante überschattet; rechts: Projektion mit vertikalem Lens Shift

Erst nach der Anschaffung des XG-P10XE von Sharp haben wir festgestellt, dass bei diesem Projektor ein fester vertikaler Lens Shift eingestellt ist, der nicht variiert werden kann. Durch diese senkrechte Verschiebung der Projektion wird das von der virtuellen Kamera aufgenommene Bild nicht an der richtigen Stelle im instrumentierten Raum projiziert und somit auch nicht korrekt entzerrt. Um dieses Problem zu umgehen, wird zunächst der Bildwinkel der virtuellen Kamera so vergrößert, dass er dem *virtuellen Bildwinkel* des Projektors entspricht. Damit wird hier der Bildwinkel bezeichnet, den der Projektor hätte, wenn man ihn als einen Projektor ohne vertikalen Lens Shift ansieht, der entlang seiner optischen Achse projiziert, wobei nur der obere Teil des projizierten Bildes sichtbar ist (siehe Abb. 3.8 rechts). Diese Projektor-Konstruktion wird durch die virtuelle Kamera simuliert, indem nur der entsprechende obere Ausschnitt des aufgenommenen Bildes vom Projektor angezeigt wird (siehe Abb. 3.8 links).

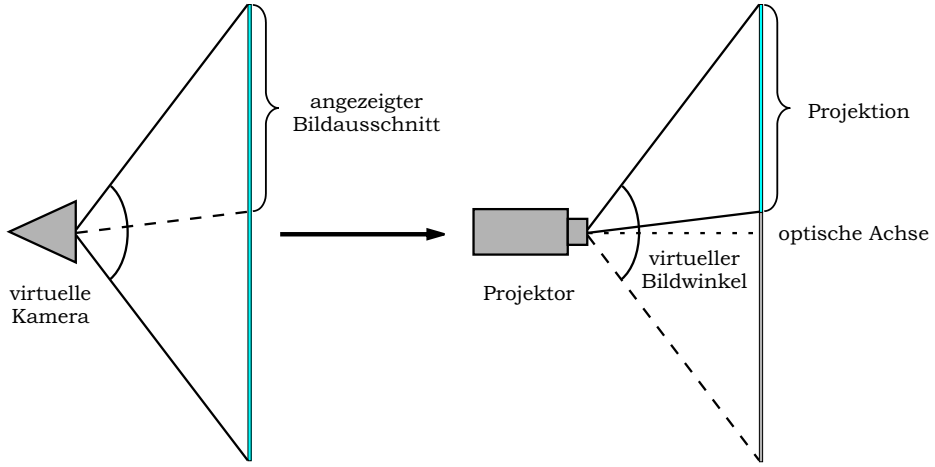

Abbildung 3.8: Anpassung des Bildwinkels der virtuellen Kamera und des angezeigten Bildausschnitts an den vertikalen Lens Shift des Prejktors

3.7 Ausrichtung des Projektors auf ein virtuelles Objekt

Virtuelle Objekte können mit unserem System an beliebige Stellen im Raum erzeugt werden, aber sie sind nur dann sichtbar, wenn der Projektor und somit auch die virtuelle Kamera auf sie gerichtet ist. Um den Projektor auf einen bestimmten Punkt im Raum zu richten, müssen aus den Koordinaten dieses Punktes die entsprechenden Pan- und Tilt-Werte der Dreheinheit berechnet werden. Aus der Position des Projektors bzw. der virtuellen Kamera und den Koordinaten des angesteuerten Punktes ergibt sich der Richtungsvektor R. Der Winkel zwischen der Projektion von R auf die horizontale xz-Ebene (R_H) und dem Richtungsvektor des Projektors in seiner Ausgangsposition (R_0) bestimmt die horizontale Ablenkung der Dreheinheit und somit den Pan-Wert. Der Tilt-Wert wird aus dem Winkel zwischen den Vektoren R_H und R abgeleitet (siehe Abb. 3.9).

Theoretisch kann der Projektor auch auf einem anderen Weg auf das virtuelle Objekt ausgerichtet werden. Dazu wird der Vek-

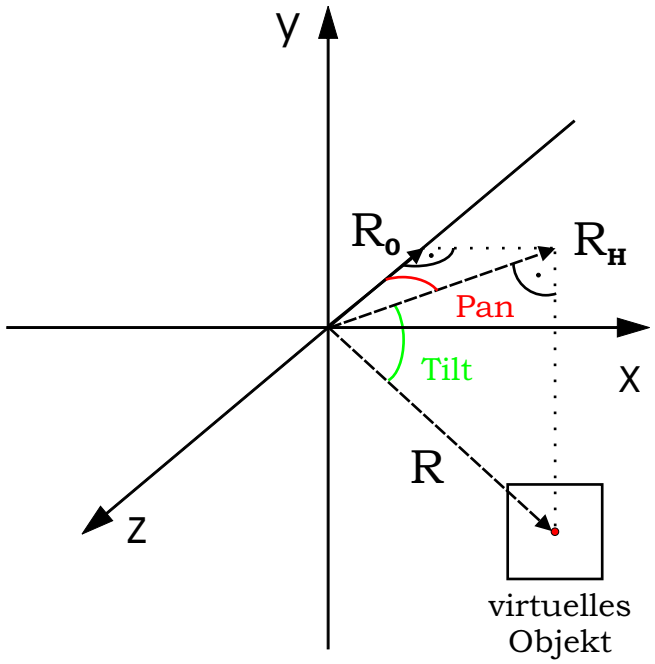

Abbildung 3.9: Bestimmung der Pan- und Tilt-Winkel

tor R_H in der xz-Ebene um 180° gedreht und man erhält den Vektor R'_H. Der Winkel zwischen R_0 und R'_H ist der alternative Pan-Winkel Pan', der alternative Tilt-Winkel $Tilt'$ wird von den Vektoren R'_H und R eingeschlossen (siehe Abb. 3.10). Bei dieser Ausrichtung wird der Projektor kopfüber gedreht. Da die virtuelle Kamera analog zum Projektor auch umgedreht wird, hat das projizierte Bild immer noch die richtige Ausrichtung. Deswegen ist die letztendlich sichtbare Projektion unabhängig davon, auf welchem Weg der Projektor ausgerichtet wurde.

In der Praxis muss man beachten, dass die Fluid Beam-Einheit bei beiden Drehachsen einen so genannten toten Winkel hat, der nicht eingestellt und über den nicht weitergedreht werden kann. Es kann also vorkommen, dass eine große Pan-Bewegung ausgeführt werden muss, obwohl die physikalische Entfernung zwischen Start- und Endposition klein ist. Wenn z.B. zwei Positionen in der horizontalen Ebene 30° voneinander entfernt sind, in die-

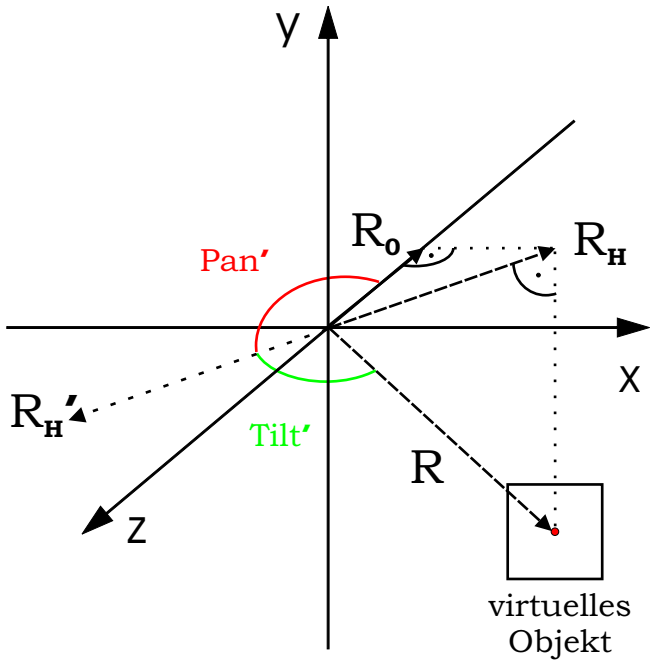

Abbildung 3.10: alternative Pan- und Tilt-Winkel

sen 30° jedoch der tote Winkel für die Pan-Bewegung liegt, muss im Endeffekt eine entgegengesetzte Pan-Drehung um 330° ausgeführt werden. In diesem Fall würde man mit der alternativen Ausrichtung einen kürzeren Weg zurücklegen.

Auf diese Weise kann man ein bereits erzeugtes virtuelles Objekt sichtbar machen, indem man die virtuelle Kamera auf dessen Mittelpunkt ausrichtet. Selbstverständlich muss darauf geachtet werden, dass die erzeugten virtuellen Objekte nicht größer sind, als die vom Projektionsstrahl überdeckte Fläche, da sie sonst nicht vollständig angezeigt werden können.

Falls das projizierte Bild vertikal geshiftet wird und der Bildwinkel der virtuellen Kamera wie in Abschnitt 3.6 beschrieben an den virtuellen Bildwinkel des Projektors angepasst wurde, muss berücksichtigt werden, dass in diesem Fall nur der obere Abschnitt des aufgenommenen Bildes angezeigt wird (siehe Abb. 3.11 links). Damit das angesteuerte virtuelle Objekt vollständig

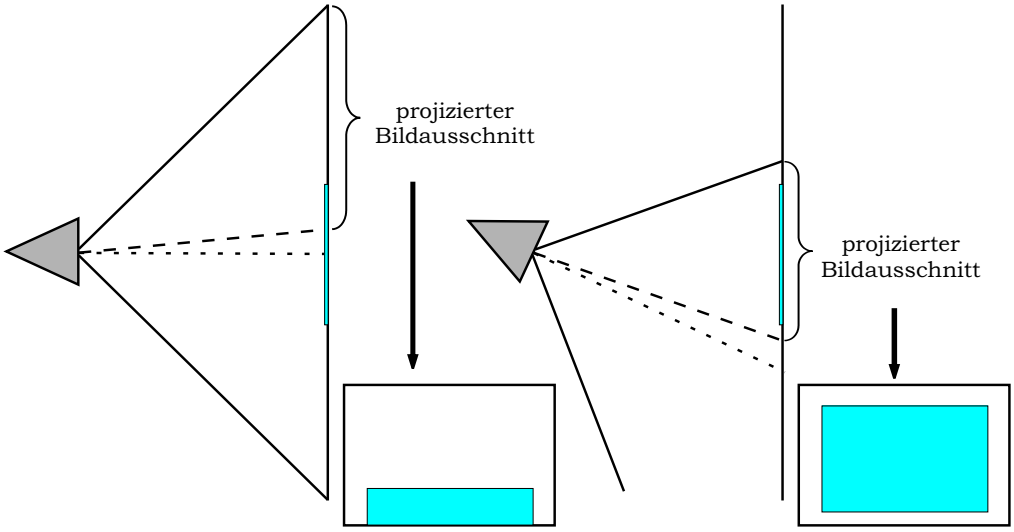

Abbildung 3.11: Korrektur des Tilt-Winkels bei vertikalem Lens Shift

angezeigt wird, muss also der berechnete Tilt-Winkel so korrigiert werden, dass das Objekt sich nicht in der Mitte des aufgenommenen Bildes, sondern in der Mitte des projizierten Bildausschnittes befindet (siehe Abb. 3.11 rechts).

3.8 Automatische Anpassung der Fokussierung des Projektors

Im Allgemeinen ändert sich der Abstand des Fluid Beam-Projektors zur jeweiligen Projektionsfläche, sobald dieser gedreht wird. Das bedeutet, dass das projizierte Bild unscharf wird und der Projektor neu fokussiert werden muss. Da sich bei einer Bewegung des Projektors der Projektionsabstand sehr oft und schnell ändern kann, ist eine automatische Fokussierung des Projektors notwendig.

Wie in Abschnitt 2.4.2 bereits erwähnt, konnte mit dem XG-P10XE-Projektor von Sharp aus technischen Gründen keine automatische Fokussierung realisiert werden. Bei unserem zwei-

ten Gerät, dem BeaMover, wird die Fokussierung über ein DMX-Kanal gesteuert, d.h. der Fokus kann auf einen DMX-Wert zwischen 0 und 255 eingestellt werden. Aus dem 3D-Raummodell kann zu jedem Punkt auf einer beliebigen Projektionsfläche der Projektionsabstand berechnet werden. Somit muss man, um den Fokus einstellen zu können, die Zuordnung zwischen Fokusabstand und DMX-Wert kennen.

Um diese Zuordnung zu bestimmen wurde eine Projektionsfläche nacheinander in unterschiedliche Entfernungen vom Projektor aufgestellt und der DMX-Wert jeweils so eingestellt, dass die Projektion korrekt fokussiert war. Dabei wurde festgestellt, dass für eine bestimmte Entfernung nicht nur ein DMX-Wert existiert, bei dem das projizierte Bild scharfgestellt ist, sondern es gibt einen Spielraum von etwa 10 DMX-Einheiten, in dem sich die Bildschärfe nicht wesentlich ändert. Um trotzdem eine eindeutige Zuordnung bestimmen zu können, wurde jeweils der mittlere Wert des DMX-Intervalls für den entsprechenden Abstand genommen.

Abbildung 3.12 zeigt die gemittelten DMX-Werte (rote Kreuze) für die jeweiligen Projektionsabstände. Aus diesen Daten wurde mit Hilfe des Programms *Origin [15]* folgende Logarithmus-Funktion als bestmögliche Näherung berechnet, die Projektionsabstände auf DMX-Werte abbildet:

$$f(x) = 98.48423 + 70.22683 * \ln(x - 0.90254)$$

Bei schräger Projektion ist es nicht möglich, das gesamte Bild mit der gleichen Schärfe zu projizierten, deswegen wird der Projektor auf die Mitte des angezeigten virtuellen Objekts fokussiert. Da man bei der Fokuseinstellung jedoch einen gewissen Spielraum hat, erscheint das projizierte Bild trotzdem nicht nur in der Mitte, sondern auch an den Rändern scharf, falls der Projektionswinkel nicht zu flach ist.

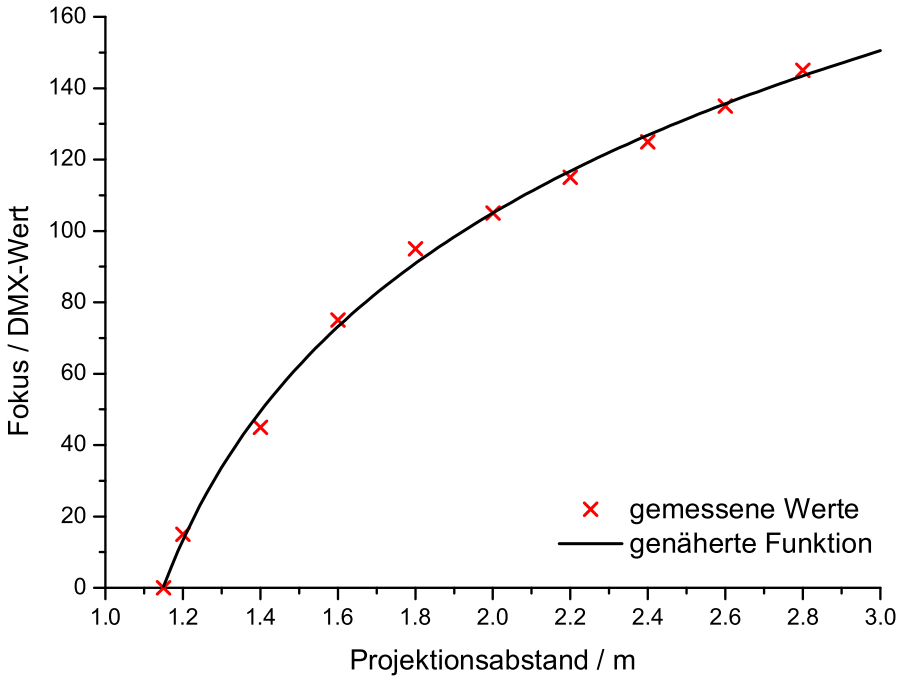

Abbildung 3.12: Abbildung von Projektionsabständen auf DMX-Werte

4 Implementierung

Für die praktische Umsetzung der bereits vorgestellten Methoden haben wir uns für die Programmiersprache *Java* entschieden. Für die Erstellung des virtuellen Modells wurde die Bibliothek *Java3D* [12] verwendet. Um das aktuelle Desktop-Bild oder Videos von einem beliebigen Rechner auf den Fluid Beam-Server zu übertragen wurden Klassen aus der Bibliothek *Java Media Framework* (*JMF*) [13] benutzt. Das System wurde schließlich in ein *Java RMI*-Interface (*Java Remote Method Invocation* [14]) eingebettet und kann somit von einem beliebigen PC aus bedient werden.

4.1 Synchronisation des Projektors mit der virtuellen Kamera

Damit das projizierte Bild immer korrekt entzerrt wird, müssen die Position und die Orientierung der virtuellen Kamera zu jedem Zeitpunkt mit denen des steuerbaren Projektors übereinstimmen. Wenn der Projektor also in der Dreheinheit bewegt wird, muss die virtuelle Kamera eine entsprechende Bewegung in der 3D-Welt ausführen. Diese Synchronisation wird mit Hilfe von linearen Interpolatoren implementiert. Diese werden in Java von der abstrakten Klasse *Interpolator* realisiert.

Ein Interpolator ist ein Objekt, das für zwei gegebene Werte x_1 und x_2 Zwischenwerte nach der Formel

$$x = (1 - \alpha) * x_1 + \alpha * x_2$$

berechnet, wobei α im Wertebereich von 0 bis 1 stetig variiert. Diese Alpha-Werte werden von einem Java-Objekt der Klasse *Alpha* geliefert. Das Alpha-Objekt bekommt unter anderem folgende Parameter: Startzeit, Anstiegsdauer (Zeit, in der der Alpha-Wert von 0 bis 1 ansteigt).

Für die Interpolation der Drehbewegungen der virtuellen Kamera wird die Unterklasse *RotationInterpolator* verwendet. Dabei wird für jede Drehachse (horizontal und vertikal) ein Rotationsinterpolator-Objekt erzeugt.

Um die DMX-Signale zu interpolieren, die die Dreheinheit des Fluid Beam-Systems steuern, wurde eine eigene Interpolator-Klasse (*FluidBeamInterpolator*) implementiert. Sie bekommt als Parameter die Anfangs- und Endpositionen der Dreheinheit (jeweils Pan- und Tilt-Winkel in rad) und ein Alpha-Objekt und sendet die entsprechenden Steuersignale an das DMX-Interface.

Damit die Drehungen des Projektors und der virtuellen Kamera synchron ablaufen, werden alle drei Interpolatoren mit demselben Alpha-Objekt gestartet, d.h. sie haben dieselben Start- und Laufzeiten.

4.2 Erzeugung virtueller Displayflächen

Um nun virtuelle Objekte an bestimmte Stellen im instrumentierten Raum projizieren zu können, muss man diese Objekte an den entsprechenden Positionen im 3D-Raummodell erzeugen. Die virtuellen Objekte müssen natürlich so geformt sein wie die physikalischen Objekte, auf die sie projiziert werden, da sie sonst verzerrt erscheinen würden. In unserem Fall werden als Projektionsflächen nur planare Flächen wie Wände oder Tischoberflächen verwendet, deswegen werden auch die virtuellen Objekte auf ebene, rechteckige Flächen reduziert, die mit unterschiedlichen Texturen versehen werden. Dabei können sowohl Bilder als

auch Videos oder sogar laufende Videosignale von einem beliebigen Rechner in Texturen umgewandelt werden (siehe Abb. 4.1). Theoretisch kann das System auf Projektionen auf beliebig geformte Objekte erweitert werden, wenn diese in Java3D modelliert und texturiert werden können.

4.2.1 VirtualDisplay

Zur Repräsentation dieser virtuellen Displayflächen wurde die Klasse *VirtualDisplay* implementiert, die aus den vier Eckpunkten oder mit Hilfe einer vorgegebenen Matrix (initiale Transformation) und der Displaygröße (Breite und Höhe) ein virtuelles Display an der entsprechenden Position im 3D-Modell erzeugt. Mit Hilfe der Funktionen *displayImage*, *displayVideo* oder *displayStream* werden die gewünschten Bilder, Videos bzw. Live-Streams als Textur auf der virtuellen Fläche angezeigt. Die Funktion *displayImage* bekommt als Parameter entweder die *URL* des anzuzeigenden Bildes oder ein Objekt vom Typ *Image*. Bei *displayVideo* muss man ebenfalls die *URL* der betreffenden Videodatei (in MPEG-Format) als Parameter übergeben. Soll ein Live-Stream mit Hilfe von *displayStream* angezeigt werden, muss der Port, an dem der Stream empfangen wird, angegeben werden. Die eigentliche Realisierung von animierten Texturen in Java3D wird in Abschnitt 4.2.3 beschrieben.

Es ist auch möglich die Position der virtuellen Displays zu verändern. Diese Positionsänderung kann sowohl abrupt als auch kontinuierlich über einen angegebenen Zeitraum erfolgen. Über die Funktion *setPoints* können die Eckpunkte eines Displays neu gesetzt werden. Auf diese Weise kann die Größe oder auch die Position des Displays verändert werden. Um ein Display in einer bestimmten Zeit auf eine neue Position zu verschieben, steht die Funktion *moveToPosition* zur Verfügung. Dazu wird ebenfalls ein Interpolator verwendet, nämlich ein Objekt der Klasse *Position-*

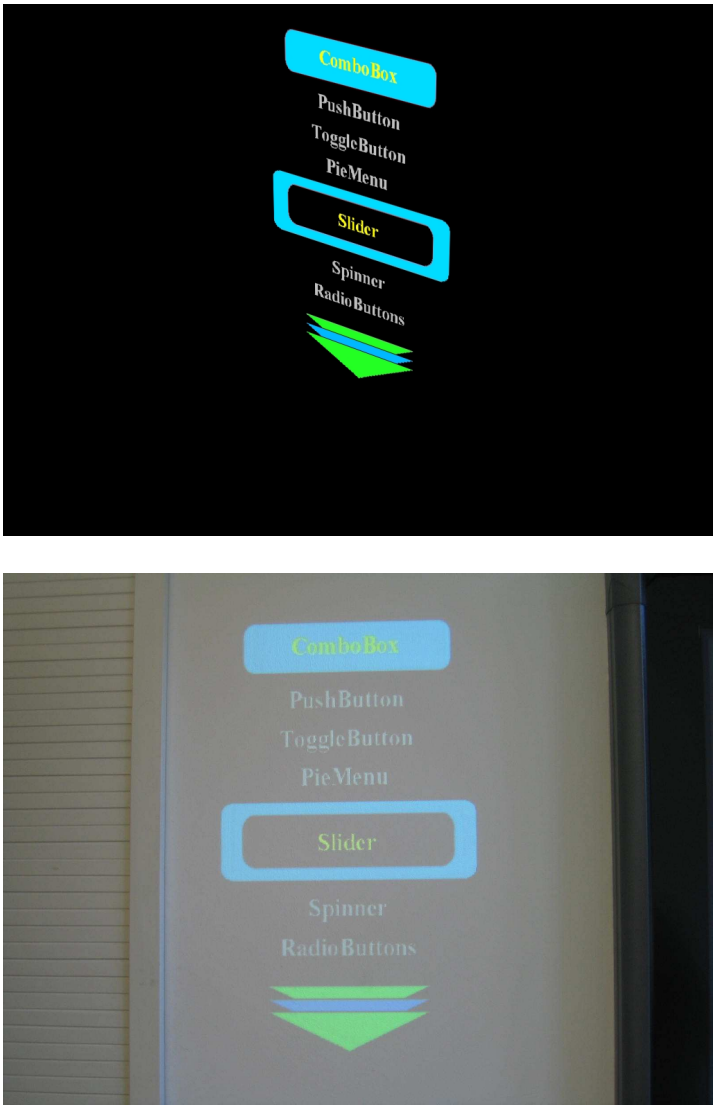

Abbildung 4.1: virtuelles Display mit Textur: Das von der virtuellen Kamera aufgenommene Bild (oben) wird an die Wand projiziert (unten).

PathInterpolator. Die Funktion bekommt als Parameter die neue Position für den Mittelpunkt des virtuellen Displays, die Zeit (in Millisekunden), in der die Bewegung erfolgen soll und eventuell die Startzeit für die Bewegung.

Somit hat man die Möglichkeit die projizierten Displays im instrumentierten Raum "wandern" zu lassen. Damit das Display auf seinem Weg immer sichtbar bleibt, muss der steuerbare Projektor stets auf seine aktuelle Position im Raum gerichtet sein. Auch in diesem Fall wird die Synchronisation der Interpolatoren über ein gemeinsames Alpha-Objekt erreicht.

4.2.2 MultipleDisplay

Im 3D-Raummodell können prinzipiell beliebig viele virtuelle Displays erzeugt werden. Um diese Displays verwalten zu können, wurde die Klasse *MultipleDisplay* implementiert. Sie stellt unter anderem Funktionen zur Verfügung, mit denen Objekte vom Typ *VirtualDisplay* erzeugt, gelöscht, mit Texturen versehen und bewegt werden können. Jedes virtuelle Display wird unter einem eigenen Namen erzeugt und kann später über diesen Namen referenziert werden. Auf diese Weise wird der instrumentierte Raum durch eine Instanz der Klasse *MultipleDisplay* als Display-Kontinuum repräsentiert.

4.2.3 Erzeugung von Bild- und Video- und Live-Stream-Texturen

In der Java3D-Klasse *TextureLoader* werden Funktionen zur Erstellung von Bildtexturen aus Objekten vom Typ *Image* zur Verfügung gestellt. Die Erzeugung von animierten Texturen hingegen wird in Java3D noch nicht explizit unterstützt.

Um trotzdem Videos und Live-Streams auf den virtuellen Displays anzeigen zu können, haben wir die Klassen *ProcessorManager*

und *AnimatedTextureRenderer* entwickelt. Mit Hilfe einer Instanz der Klasse *ProcessorManager* wird ein kontinuierlicher Datenstrom von einer URL (MPEG-Video) oder einem lokalen Port (Live-Stream) empfangen und an ein Objekt vom Typ *AnimatedTextureRenderer* weitergeleitet. Hier werden aus den empfangenen Videodaten Einzelbilder extrahiert, die sukzessiv als Texturen auf dem virtuellen Display angezeigt werden.

4.2.4 Manuelle Display-Kalibrierung

Falls kein exaktes Modell des instrumentierten Raums erstellt werden kann, besteht auch die Möglichkeit, die virtuellen Displays manuell zu kalibrieren. Zu diesem Zweck wurde die ausführbare Klasse *DisplayAdjustment* entwickelt. Sie erzeugt ein virtuelles Display, auf dem ein Testbild angezeigt wird, an einer Default-Position und richtet den steuerbaren Projektor darauf. Nun kann der Benutzer unter Verwendung folgender Tasten bzw. Tastenkombinationen die gewünschte Position und Orientierung des Displays manuell einstellen: Durch einmaliges Drücken der Tasten <x>, <y> oder <z> wird jeweils die lokale x-, y- oder z-Achse des virtuellen Displays als Dreh- und Verschiebungsachse festgelegt. Danach kann man das Display über die Pfeiltasten auf der voreingestellten Achse verschieben (↑/↓) oder drehen (←/→). Durch einmaliges Drücken der <0>-Taste werden alle bis dahin vorgenommenen Drehungen um die aktuell eingestellte Achse wieder rückgängig gemacht.

Falls das Display beim Verschieben aus dem Projektionsbereich bewegt wird, kann der Projektor wieder darauf gerichtet werden, indem man über die <p>-Taste die Dreheinheit als Manipulationsobjekt auswählt und über die Pfeiltasten den Pan- bzw. Tilt-Winkel verändert. Über die ←/→-Tasten wird die Pan-Bewegung gesteuert, der Tilt-Winkel wird über die ↑/↓-Tasten eingestellt. Dabei wird die Dreheinheit in Schritten von 0.01 rad gedreht.

Wird zusätzlich die <Shift>-Taste gedrückt, wird die Schrittgröße bei der Drehung auf 0.1 rad erhöht, um schnellere Bewegungen ausführen zu können.

Über die Taste <f> kann der Fokus ebenfalls als Manipulationsobjekt ausgewählt und mit den Pfeiltasten (↑/↓) justiert werden.

Nachdem das virtuelle Display passend kalibriert wurde, wird das Programm durch Drücken der <Return>-Taste beendet und die vorgenommenen Einstellungen werden in einer Kalibrierungsdatei gespeichert. Später kann man diese Datei zur Erstellung eines virtuellen Displays verwenden, indem man den Dateinamen als Parameter für die Funktion *createDisplay* in der Klasse *MultipleDisplay* (siehe Abschnitt 4.2.2) einsetzt.

Die zur Display-Kalibrierung verwendbaren Tasten und Tastenkombinationen und ihre Funktionen werden in Tabelle 4.1 nochmal zusammengefasst.

Manipulationsobjekt	Auswahltaste	Pfeiltasten (↑/↓)	Pfeiltasten (←/→)
Display (x-Achse)	x	Verschiebung	Rotation
Display (y-Achse)	y	Verschiebung	Rotation
Display (z-Achse)	z	Verschiebung	Rotation
Dreheinheit/ Projektor	p	Tilt-Rotation	Pan-Rotation
Fokus	f	Verschiebung	-
<Shift>+Pfeiltasten: große Schritte bei Pan- und Tilt-Drehung			
0: Ausgangsdrehung (0 rad) in aktueller Achse wiederherstellen			
<Return>: Einstellungen speichern und Programm beenden			

Tabelle 4.1: Tasten und Tastenkombinationen zur manuellen Display-Kalibrierung

4.3 Einbettung in die RMI-Architektur des Umgebungsmanagers

Damit die Fluid Beam-Applikation von beliebigen Rechnern aus bedient werden kann, wurde sie in ein RMI-Interface eingebettet. Durch Starten der ausführbaren Klasse *FluidBeam* wird die Anwendung initialisiert und bei einem so genannten Umgebungsmanager (*Fluid Manager* [18]) angemeldet. Dieser wurde im Rahmen einer Doktorarbeit im FLUIDUM-Projekt entwickelt und verwaltet die verschiedenen Geräte bzw. Anwendungen in einer instrumentierten Umgebung. Der Umgebungsmanager regelt unter anderem die Verteilung der verfügbaren Ressourcen (z.B. Displays) an die laufenden Anwendungen. Jede neu gestartete Anwendung wird beim Umgebungsmanager registriert und liefert dabei eine Beschreibung von bestimmten Diensten (z.B. visuelle Ausgabe, Audioausgabe, Bild-Input, Video-Input usw.), die sie anderen Anwendungen zur Verfügung stellt.

Die Fluid Beam-Anwendung stellt die Dienste "steuerbar" (Dreheinheit), "visuelle Ausgabe" (Projektor), "Bild-Input" (Kamera/Bildaufnahme) und "Video-Input" (Kamera/Videoaufnahme) bereit. Diese werden durch die Java-Schnittstellen *Steerable*, *VideoOut*, *ImageIn* und *VideoIn* repräsentiert. Das Interface *Steerable* wird bei der Fluid Beam-Applikation durch die Klasse *SteerableUnit* implementiert. Entsprechend werden *VideoOut* durch die Klasse *MultipleDisplay*, *ImageIn* durch die Klasse *CanonImageIn* und *VideoIn* durch die Klasse *CanonVideoIn* realisiert.

4.4 Verknüpfung mit weiteren Komponenten

Schon während der Entwicklung der Fluid Beam-Applikation ergaben sich vielfältige Einsatzmöglichkeiten für die steuerbare Projektor-Kamera-Einheit. Durch Verknüpfung mit anderen im Rah-

men des FLUIDUM-Projekts bzw. am Lehrstuhl für Künstliche Intelligenz von Prof. Wahlster entwickelten Programmen konnten neue Anwendungen entstehen, die im Folgenden vorgestellt werden.

4.4.1 Markererkennung (Search Light)

Search Light [16] ist eine Anwendung, mit der man nach physikalischen Objekten in einem instrumentierten Raum suchen kann. Sie kann als physikalische Analogie zur allgemein gebräuchlichen Suchfunktion nach Dateien und Ordner auf einem PC angesehen werden. Durch eine Ergänzung der PC-Suchfunktion mit der Search Light-Funktionalität würde also die Unterscheidung zwischen virtuellen und physikalischen Objekten entfallen, und man würde eine allgemeine Suchfunktion für den instrumentierten Raum erhalten.

Die Search Light-Anwendung wird in zwei grundlegende Modi unterteilt: den Scan-Modus und den Such-Modus. Die zu suchenden Objekte werden mit optischen Markern (z.B. *JARToolkit* [11]) gekennzeichnet. In einem ersten Schritt wird die gesamte instrumentierte Umgebung mit Hilfe der Fluid Beam-Kamera nach möglichen Markern abgescannt. Dabei wird die Dreheinheit in kleinen Schritten bewegt und an jeder Position wird die Kamera ausgelöst. Die Schrittgröße muss so gewählt werden, dass sich die Bilder leicht überlappen, so dass keine Marker beim Scannen übersehen werden. Abbildung 4.2 zeigt mehrere Bilder, die beim Abscannen eines Bücherregals aufgenommen wurden. Jedes Bild wird mit der entsprechenden Markererkennungssoftware analysiert und falls Marker gefunden wurden, werden ihre IDs zusammen mit der Markerposition im Bild und der aktuellen Pan- und Tilt-Werte der Dreheinheit in einer Liste gespeichert.

Abbildung 4.2: Beim Scan-Vorgang aufgenommene Bilder

Nach dem initialen Scan-Vorgang kann der Benutzer nach bestimmten Objekten im Raum suchen. Ein Objekt wird über die ID des ihm zugeordneten Markers identifiziert. Falls in der beim Scannen erzeugten Liste ein Eintrag für den gesuchten Marker existiert, wird die Dreheinheit auf die gespeicherte Position gefahren und es wird ein heller Kreis um das gesuchte Objekt herum projiziert (siehe Abb. 4.3). Auf diese Weise wird die Aufmerksamkeit des Benutzers auf den gesuchten Gegenstand gelenkt.

Da sich im Laufe der Zeit die Positionen der markierten Gegenstände ändern können, müssen in gewissen Zeitabständen neue Scans vorgenommen werden, um die gespeicherten Daten zu aktualisieren. Es ist anzunehmen, dass es dabei Bereiche geben wird, in denen mehr Positionsänderungen auftreten werden als in anderen. Um das System möglichst effizient zu machen, muss man also diese Bereiche verstärkt abscannen. Die Häufigkeit der Bewegungen, die in einer Region (z.B. mit Hilfe einer zusätzlichen Kamera) beobachtet werden, kann als Maß für die dort stattfindenden Positionsänderungen angenommen werden. Man würde dann Bereiche, in denen viel Bewegung registriert wurde, öfter abscannen als solche mit wenig Bewegung. Wenn in einem Bereich für eine bestimmte Zeit überhaupt keine Bewegung erfasst wurde, braucht man diesen nicht neu zu scannen, da man sicher sein kann, dass dort keine Änderungen stattgefunden haben.

Abbildung 4.3: Durch Projektion hervorgehobenes Buch

Im Gegensatz zur normalen Fluid Beam-Applikation wird für die Search Light-Anwendung kein 3D-Modell der instrumentierten Umgebung benötigt. Die Position des projizierten Kreises wird bestimmt durch die Position des gesuchten Markers im Bild, in dem dieser erkannt wurde, und durch die Pan- und Tilt-Werte der Dreheinheit, die bei der Bildaufnahme eingestellt waren.

Mögliche Einsatzbereiche für die Search Light-Funktion sind Bibliotheken, Buchläden, größere Büros usw. Die gesuchten Objekte müssen groß genug sein, damit sie mit einem Marker versehen werden können, und sie müssen für die Kamera sichtbar sein. Die benötigte Markergröße ist abhängig von der Auflösung der verwendeten Kamera, dem eingestellten Zoom und der Entfernung zwischen Kamera und Marker. Je höher die Auflösung, je größer der Zoom und je kleiner der Abstand zwischen Marker und Kamera, desto kleiner dürfen die Marker sein.

Abbildung 4.4: Wischgesten-Demo: Touchscreen mit Bilderordner, projiziertes Tisch-Display mit zwei ausgewählten Bildern und großes Präsentationsdisplay (weiße Fläche über der Tür), CeBit 2004

4.4.2 Gestikerkennung (Wischgesten)

Eine erste Anwendung, die Interaktionen mit dem projizierten Bild möglich macht, ist die Wischgesten-Demo, die wir für die Computermesse CeBit 2004 entwickelt haben. Bei dieser Applikation kann der Benutzer durch einfache Gesten virtuelle Objekte (Bilder und Videos) zwischen physikalischen und projizierten Displays hin und her verschieben.

Das Szenario sieht folgendermaßen aus: Dem Benutzer stehen drei verschiedene Displays zur Verfügung - ein (physikalischer) Touchscreen, ein kleines virtuelles Display auf der Tischoberfläche vor dem Touchscreen und ein großes virtuelles Display an

einer freien Wand (siehe Abb. 4.4). Auf dem Touchscreen wird in einem Fenster ein Symbol angezeigt, das das virtuelle Tisch-Display repräsentiert. So kann der Benutzer Bilder und Videos vom Touchscreen auf den Tisch übertragen, indem er ihre Icons nach dem Drag-and-Drop-Prinzip auf das Tisch-Symbol zieht. Sobald man ein Icon auf das Tisch-Symbol bewegt, verschwindet es auf dem Touchscreen und wird auf dem projizierten Tisch-Display in Kleinformat angezeigt. So kann man auf der Tischoberfläche eine bestimmte Auswahl von Bildern sammeln. Nun kann der Benutzer die ausgewählten Bilder von der Tischoberfläche auf das große Display an der Wand übertragen, indem er seine Hand über das Tisch-Display in Richtung Wand bewegt (Wischgeste). Dort werden die Bilder in Großformat in einer Art Diavorführung nacheinander projiziert. Es besteht auch die Möglichkeit durch eine andere Wischbewegung, nämlich in Richtung Monitor, die Bilder von der Tischoberfläche wieder auf den Touchscreen zu verschieben.

Mit dieser Applikation können z.B. PowerPoint-Präsentationen oder Diashows zusammengestellt und angezeigt werden. Auf seinem privaten Tisch-Display kann der Benutzer schnell die gewünschten Folien, Bilder oder Filme auswählen und sie dann durch eine Geste an die Wand "wischen", wo sie in Großformat nacheinander präsentiert werden.

4.4.3 Räumliches Audio (SAFIR)

Im Rahmen einer Diplomarbeit im FLUIDUM-Projekt wurde das System SAFIR (Spatial Audio For Instrumented Rooms) zur 3-dimensionalen Positionierung von virtuellen Schallquellen entwickelt [17]. Mit Hilfe von acht handelsüblichen Lautsprechern, die kreisförmig unter der Decke des instrumentierten Raums angeordnet sind, kann der Eindruck erzeugt werden, dass ein Klang aus einer bestimmten Position im Raum kommt. Es können prin-

zipiell beliebig viele Schallquellen an beliebige Positionen erzeugt werden.

Durch Verknüpfung der Fluid Beam-Applikation mit dem SAFIR-System ist es möglich, virtuellen Displays eine Audioausgabe räumlich zuzuordnen. Wenn die Position des Displays verändert wird, kann der dazugehörige Klang mitbewegt werden.

Das räumliche Audiosystem kann auch zur Erweiterung der bereits beschriebenen Search Light-Funktion (siehe Abschnitt 4.4.1) eingesetzt werden. In diesem Fall würde der Benutzer sowohl durch die Projektion als auch durch einen Klang von der entsprechenden Position auf das gesuchte Objekt aufmerksam gemacht werden.

4.4.4 Virtueller Bewohner der instrumentierten Umgebung

In einer instrumentierten Umgebung werden dem Benutzer viele - oft neuartige - Funktionalitäten und Geräte zur Verfügung gestellt. Jedoch können vor allem unerfahrene Benutzer von diesen Neuerungen sehr oft nicht profitieren, da sie ihre Handhabung nicht kennen oder nicht einmal etwas von ihrer Existenz wissen.

Um dem Benutzer den Umgang mit einem Programm zu erleichtern, werden bei komplexen Desktop-Anwendungen oft virtuelle Assistenten eingesetzt (z. B. "Karl Klammer" bei Microsoft Word), die dem Benutzer in bestimmte Situationen nützliche Tipps und Hinweise geben. Dieses Konzept kann auch auf den instrumentierten Raum übertragen werden.

Eine aktuelle Doktorarbeit am Lehrstuhl von Prof. Wahlster beschäftigt sich mit der Entwicklung eines solchen virtuellen Assistenten für den instrumentierten Raum, der neue Benutzer mit den Funktionalitäten und Geräte in der instrumentierten Umgebung vertraut machen und ihnen in bestimmten Situationen Hilfestellungen geben soll. Für die Visualisierung dieses Assistenten

Abbildung 4.5: Cyberella (links) neben einem Plasma-Bildschirm im instrumentierten Raum

wird der Fluid Beam-Projektor eingesetzt. Somit kann man den virtuellen Bewohner über die Wände des instrumentierten Raums ”wandern” lassen und ihn jeweils so positionieren, dass er sich in unmittelbarer Nähe der Geräte befindet, die er gerade beschreibt.

In einem ersten Demo-Szenario wird der Benutzer beim Betreten des instrumentierten Raums zunächst von Cyberella - einem animierten weiblichen Avatar - begrüßt. Danach stellt sie einzelne Geräte im Raum vor, wobei sie sich über die Wände von einem Gerät zum nächsten bewegt (siehe Abb. 4.5).

Die Bewegungen des Avatars werden durch eine Flash-Animation realisiert, die von einer Character Engine (CE) gesteuert wird. Der CE-Server greift über den Umgebungsmanager auf die Fluid Beam-Applikation zu (siehe Abschnitt 4.3), erzeugt ein virtuelles Display, auf dem der Avatar angezeigt wird und steuert über RMI-Aufrufe die Bewegungen des Projektors und des virtuellen Displays. Die visuelle Ausgabe der Flash-Animation wird über ei-

ne RTP-Verbindung als JPEG-Stream an den Fluid Beam-Server geschickt.

Für die räumliche Audio-Ausgabe wurde das SAFIR-System, wie in Abschnitt 4.4.3 beschrieben, mit der Fluid Beam-Applikation gekoppelt, so dass die Sprachausgabe immer aus der aktuellen Position des Avatars stammt und mit ihm bewegt wird.

5 Zusammenfassung und Ausblick

Das im Rahmen dieser Arbeit realisierte Fluid Beam-System ermöglicht eine verzerrungsfreie Projektion auf beliebig orientierte ebene Flächen (Wände, Tischoberflächen). Die dazu verwendete Hardware besteht aus einem Projektor und einer Kamera, die in einer Dreheinheit unter der Decke des instrumentierten Raums angebracht sind. Das System kann über RMI-Aufrufe ferngesteuert werden. Es werden Funktionen zur Verfügung gestellt, um virtuelle Objekte (virtuelle Displays) an beliebigen Stellen im instrumentierten Raum zu erzeugen, auf denen sowohl Bilder als auch Videos oder Live-Streams angezeigt werden können. Durch die Verwendung eines 3D-Raummodells wird die instrumentierte Umgebung mit einer virtuellen Schicht überzogen, auf der die erzeugten virtuellen Objekte positioniert und verschoben werden können. Dieses Display-Kontinuum ist nur an der Stelle sichtbar, die vom Projektorstrahl beleuchtet wird. Somit ist der Projektor methaphorisch gesprochen eine Taschenlampe, mit der man die virtuellen Objekte in der instrumentierten Umgebung sichtbar machen kann, indem man sie darauf richtet. Auf diese Weise ist es möglich, Informationen lokal im Raum zu speichern und anzuzeigen (siehe [19]).

Darüber hinaus wurden verschiedene Kalibrierungsverfahren (Projektor- und Kameraposition, Bildwinkel des Projektors bzw. der virtuellen Kamera, manuelle Positionierung der virtuellen Displays) vorgestellt, die für die Realisierung des gewählten Ansat-

zes notwendig sind. Anschließend wurden mehrere bereits implementierte Anwendungen (Markererkennung, Gestikerkennung, Verknüpfung mit räumlichem Audio, virtueller Assistent für den instrumentierten Raum) beschrieben.

Nachfolgend werden einige mögliche Verbesserungen und Erweiterungen des Systems erläutert, die Gegenstand einer weiterführenden Doktorarbeit sein können (siehe [20]).

5.1 Selbstkalibrierung

In der aktuellen Version des Fluid Beam-Systems muss das 3D-Raummodell manuell erstellt werden, d.h. der instrumentierte Raum wird vermessen und die Werte werden in einer Datei eingetragen. Die Bestimmung der exakten Projektorposition (siehe Abschnitt 3.2) erfolgt halbautomatisch.

Um die Bedienung des Systems für den Benutzer zu erleichtern, kann man ein Verfahren entwickeln, mit dem das 3D-Modell der instrumentierten Umgebung automatisch erfasst wird. Dabei kann man, ähnlich zu den in [21], [22] und [23] beschriebenen Methoden, strukturiertes Licht verwenden, um die Geometrie des Raums festzuhalten. Mit der Fluid Beam-Einheit können bestimmte Muster im Raum projiziert werden, die mit einer Kamera aufgenommen und später analysiert werden. Aus der beobachteten Veränderung (Verzerrung) des projizierten Bildes kann die Form des Projektionshintergrunds rekonstruiert werden.

Es muss untersucht werden, ob die Fluid Beam-Kamera geeignet ist, um die projizierten Muster aufzunehmen, oder eine zusätzliche Kamera verwendet werden muss. Man muss dabei beachten, dass die Kamera das gesamte projizierte Bild erfassen muss und dass die optischen Achsen von Kamera und Projektor einen genügend großen Abstand voneinander haben sollten, damit die Verzerrung der Projektion im aufgenommenen Bild sicht-

bar ist. Aus diesem Grund könnte es notwendig sein, eine externe Kamera zu verwenden, da Kamera und Projektor der Fluid Beam-Einheit einen sehr kleinen Abstand voneinander haben.

Um den gesamten Raum zu erfassen, kann man prinzipiell auf zwei verschiedene Arten vorgehen. Zum einen kann man mehrere Einzelbilder bei unterschiedlichen Einstellungen der Dreheinheit aufnehmen, wobei die Fluid Beam-Einheit in kleinen Schritten bewegt wird, ähnlich dem Scan-Vorgang bei der Search Light-Anwendung (siehe Abschnitt 4.4.1). Alternativ wäre ein kontinuierlicher Scan-Vorgang denkbar, bei dem das projizierte Bild langsam über die Flächen im Raum bewegt wird. Dabei kann die Kamera ein kontinuierliches Video vom projizierten Muster aufnehmen.

Wie in Abschnitt 2.4.1 angedeutet, können beim Positionieren der Fluid Beam-Einheit mechanisch bedingte Abweichungen auftreten. Eine höhere Genauigkeit bei der Positionierung der Dreheinheit kann durch den Einsatz von Winkelmessern erreicht werden, die an beide Drehachsen der Einheit befestigt werden. So kann die aktuelle Ausrichtung des Projektors genauer bestimmt und mit der virtuellen Kamera synchronisiert werden.

5.2 Interaktionen

Das vorliegende Fluid Beam-System ist hauptsächlich zum Anzeigen von Informationen geeignet. Für viele Anwendungen ist es jedoch wichtig einen Rückkanal zu haben, über den der Benutzer mit dem System interagieren kann. Die vom Desktop-PC bekannten Eingabe-Geräte, nämlich Tastatur und Maus, werden im instrumentierten Raum durch intuitive, alltägliche Interaktionstechniken, wie Gesten und Sprache, abgelöst.

5.2.1 Gestikerkennung

Eine Rückmeldung von der Umgebung oder vom Benutzer ist beim aktuellen Stand des Fluid Beam-Systems nur in Form von optischen Markern oder durch einfache Wischgesten möglich. Um komplexere Interaktionen mit dem System zu ermöglichen, kann die Gestikerkennung verfeinert werden, so dass nicht nur die Bewegungsrichtung der Hand, sondern auch einzelne Finger und ihre Position bezüglich des projizierten virtuellen Objekts erkannt werden können. Auf diese Weise wird das projizierte Bild wie ein Touchscreen bedient werden können. Man wird einzelne Objekte verschieben oder Icons durch bestimmte Bewegungen oder Gesten, analog zum Mausklick, aktivieren können. In [24] und [25] werden solche Fingererkennungsverfahren vorgestellt, die für die Fluid Beam-Applikation adaptiert werden können. Beide Verfahren wurden bereits für Interaktionen mit projizierten Bildern eingesetzt. Ein System zur Gestik-Interaktion mit dem Everywhere Displays-Projektor wird in [26] beschrieben.

5.2.2 Spracherkennung

Eine weitere Möglichkeit der Interaktion zwischen Benutzer und instrumentierter Umgebung ist die Sprachsteuerung. Die heutigen Spracherkennungssysteme sind sehr robust und für vielseitige Themenbereiche einsetzbar. Einer der führenden Anbieter auf dem Gebiet der Sprachverarbeitung ist die Firma ScanSoft [27], aber es gibt auch wissenschaftliche Projekte, die in diesem Bereich forschen und ihre Software frei zur Verfügung stellen, so wie das Sphinx Projekt an der Carnegie Mellon Universität [28].

Auch die Fluid Beam-Applikation könnte über Sprache gesteuert werden. Dafür müsste der Benutzer eventuell ein Headset mit eingebautem Mikrofon tragen, das über Bluetooth an das System angeschlossen ist. Dadurch könnte der Benutzer Anweisungen geben, um z.B. die Position eines neu erzeugten virtuellen Dis-

plays festzulegen oder Fragen an den virtuellen Assistenten zu stellen.

Auch kombinierte Sprach- und Gestikeingaben wären denkbar. So könnte der Benutzer z.B. eine Frage an den virtuellen Raumbewohner stellen, die sich auf ein bestimmtes Gerät bezieht und dabei auf dieses deuten. Eine aktuelle Doktorarbeit am Lehrstuhl von Prof. Wahlster beschäftigt sich mit der Entwicklung einer solchen multimodalen Interaktionsschnittstelle für mobile Systeme, die auch zur Steuerung der Fluid Beam-Einheit eingesetzt werden könnte.

5.2.3 Interaktionen zwischen stationären Displays und Projektionen

In einer instrumentierten Umgebung werden die stationären physikalischen Displays durch das projizierte Display-Kontinuum ergänzt und erweitert. Deshalb sollte es auch möglich sein, zwischen herkömmlichen Displays und Projektionen zu interagieren, so als gäbe es keinen Unterschied zwischen den beiden Ausgabemedien. So wie man derzeit an einem PC schon mehrere Monitore anschließen kann, die zu einem Gesamtdisplay zusammengefügt werden, sollen bald auch stationäre Displays und Projektion einander ergänzen können. Es können Techniken entwickelt werden, mit denen man Objekte zwischen Displays und Projektionen nahtlos verschieben kann, so dass das Display-Kontinuum des Fluid Beam-Systems als Transportmedium für virtuelle Objekte zwischen zwei stationären Displays dienen kann.

5.3 Kombination mehrerer steuerbarer Einheiten

Der instrumentierte Raum, in dem das Fluid Beam-System bisher entwickelt wurde, ist relativ klein (ca. 4 m x 4,5 m x 3 m), weshalb man auf alle Wände projizieren kann, wenn man die Fluid Beam-Einheit in der Mitte der Decke montiert. Da der Projektor von etwa 1,4 m bis 14,4 m fokussiert werden kann, ist es in größeren oder verwinkelten Räumen jedoch nicht möglich, die Fluid Beam-Einheit so zu platzieren, dass alle Flächen vom Projektorstrahl erreicht werden können.

Um das Fluid Beam-System auch in solche Räume einsetzen zu können, müssen zwei oder mehrere steuerbare Einheiten in der Umgebung platziert werden. So können virtuelle Objekte über längere Distanzen hin verschoben werden, indem die Projektion bildlich gesehen von einer steuerbaren Einheit an die nächste übergeben wird. Diese müssen in ihren Bewegungen so synchronisiert werden, dass der Übergang zwischen zwei Projektionsbereichen für den Benutzer nicht sichtbar wird. Mit dieser Methode kann z.B. die Visualisierung eines Navigationssystems implementiert werden, mit dem der Benutzer durch projizierte Pfeile in einer größeren instrumentierten Umgebung (z.B. Lagerraum, Einkaufszentrum usw.) zu seinem Ziel geleitet wird (siehe Abb. 5.1).

Weiterhin kann mit mehreren steuerbaren Einheiten durch geeignetes Zusammenfügen der einzelnen projizierten Bilder eine größere Projektion erzeugt werden. In [23] wird ein solches Verfahren anhand eines Beispiels vorgestellt, bei dem durch ein Cluster aus fünf Projektoren ein breites homogenes Panoramabild erzeugt wird.

Abbildung 5.1: projizierter Pfeil als Navigationshilfe

6 Codeauszüge

Das Interface **Steerable**:

```
public interface Steerable extends DeviceProperty {

   /** Richtet den Projektor auf einen vorgegebenen
     * Punkt im instrumentierten Raum.
     * @return: true nachdem die Bewegung erfolgreich
     * ausgeführt wurde, sonst false
     * @param point: der Punkt, auf den der Projektor
     * gerichtet wird
     * @param time: die Zeit, in der die Bewegung
     * ausgeführt werden soll
     * @throws RemoteException
     */
    public boolean moveToPosition(Point3d point,
    long time) throws RemoteException;

   /** Bewegt ein virtuelles Display auf eine
     * vorgegebene Position;
     * der Projektor wird parallel dazu bewegt,
     * so dass das Display
     * immer sichtbar bleibt.
     * @param point: der neue Mittelpunkt des Displays
     * @param time: die Zeit, in der die Bewegung
     * ausgeführt werden soll
     * @param df: die Display Factory, in der das Display
     * erzeugt wurde
     * @param id: der Name des Displays
     * @throws RemoteException
     * @return: true nachdem die Bewegung erfolgreich
     * ausgeführt wurde, sonst false
```

```
     */
    public boolean moveToPositionWithDisplay(Point3d point,
    long time, DisplayFactory df, String id)
    throws RemoteException;

   /** Bewegt die Dreheinheit auf eine vorgegebene
     * Position.
     * @param pan: die horizontale Ablenkung (in rad)
     * @param tilt: die vertikale Ablenkung (in rad)
     * @param time: die Zeit, in der die Bewegung
     * ausgeführt werden soll
     * @throws RemoteException
     * @return: true nachdem die Bewegung erfolgreich
     * ausgeführt wurde, sonst false
     */
    public boolean moveToDirection(double pan, double tilt,
    long time) throws RemoteException;

   /** Liefert die aktuelle Position der Dreheinheit.
     * @throws RemoteException
     * @return: horizontale Ablenkung (in rad),
     * verikale Ablenkung (in rad)
     */

    public double[] getDirection() throws RemoteException;

   /** Stellt fest, ob der Projektor auf einen bestimmten
     * Punkt gerichtet werden kann.
     * @param point: der Punkt, auf den der Projektor
     * gerichtet werden soll
     * @throws RemoteException
     * @return: true falls die Ausrichtung möglich ist,
     * sonst false
     */
    public boolean canMoveTo(Point3d point)
    throws RemoteException;

   /** Stellt fest, ob die Dreheinheit auf eine bestimmte
     * Position bewegt werden kann.
```

```
     * @param pan: die horizontale Ablenkung (in rad)
     * @param tilt: die vertikale Ablenkung (in rad)
     * @throws RemoteException
     * @return: true falls die Ausrichtung möglich ist,
     * sonst false
     */

    public boolean canMoveTo(double pan, double tilt)
    throws RemoteException;

   /** Bestimmt den horizontalen Ablenkbereich der
     * Dreheinheit.
     * @throws RemoteException
     * @return: der minimale und maximale horizontale
     * Ablenkwinkel (in rad)
     */
    public double[] getPanRange() throws RemoteException;

   /** Bestimmt den vertikalen Ablenkbereich der
     * Dreheinheit.
     * @throws RemoteException
     * @return: der minimale und maximale vertikale
     * Ablenkwinkel (in rad)
     */
    public double[] getTiltRange() throws RemoteException;

   /** Liefert die initiale Position der Dreheinheit.
     * @throws RemoteException
     * @return: die initiale Transformationsmatrix
     */
    public Matrix4d getHomeTransformation()
    throws RemoteException;
}
```

Das Interface **VirtualDisplayInterface**:

```
public interface VirtualDisplayInterface extends VideoOut {
   /** Liefert den Mittelpunkt des virtuellen Displays.
     * @throws RemoteException
```

```
     * @return: der Mittelpunkt
     */
    public Point3d getMidpoint() throws RemoteException;

   /** Liefert die Branch Group an der das virtuelle
     * Display angehängt ist.
     * @throws RemoteException
     * @return: die Branch Group
     */
    public BranchGroup getBranchGroup()
    throws java.rmi.RemoteException;

   /** Liefert die Transform Group des virtuellen
     * Displays.
     * @throws RemoteException
     * @return: die Transform Group
     */

    public TransformGroup getTransformGroup()
    throws java.rmi.RemoteException;

   /** Setzt die Eckpunkte des virtuellen Displays neu;
     * die Punkte werden gegen den Uhrzeigersinn
     * angegeben.
     * @param p0: links unten
     * @param p1: rechts unten
     * @param p2: rechts oben
     * @param p3: links oben
     * @throws RemoteException
     */
    public void setPoints(Point3d p0, Point3d p1, Point3d
    p2, Point3d p3) throws java.rmi.RemoteException;

   /** Bewegt das virtuelle Display auf eine neue Position.
     * @param point: der neue Mittelpunkt des Displays
     * @param time: die Zeit, in der die Bewegung
     * ausgeführt werden soll
     * @throws RemoteException
     */
```

```
        public void moveToPosition(Point3d point, long time)
        throws java.rmi.RemoteException;

       /** Bewegt das virtuelle Display auf eine neue Position.
         * @param point: der neue Mittelpunkt des Displays
         * @param startTime: die Zeit, in der die Bewegung
         * ausgeführt werden soll
         * @param time: die Startzeit der Bewegung
         * @throws RemoteException
         */
        public void moveToPosition(Point3d point, long
        startTime, long time) throws java.rmi.RemoteException;
    }
```

Das Interface **DisplayFactory**:

```
    public interface DisplayFactory extends DeviceProperty {

       /** Erzeugt ein neues virtuelles Display mit den
         * vorgegebenen Eckpunkten, die gegen den
         * Uhrzeigersinn angegeben werden.
         * @param ID: der Name des Displays
         * @param p0: links unten
         * @param p1: rechts unten
         * @param p2: rechts oben
         * @param p3: links oben
         * @throws RemoteException
         */
        public void createDisplay(String ID, Point3d p0, Point3d
        p1, Point3d p2, Point3d p3) throws RemoteException;

       /** Erzeugt ein neues virtuelles Display mit der
         * vorgegebenen Größe.
         * @param ID: der Name des Displays
         * @param matrix: initiale Transformation
         * @param width: die Breite des Displays
         * @param height: die Höhe des Displays
         * @throws RemoteException
         */
```

```
 public void createDisplay(String ID, Matrix4d matrix,
 float width, float height) throws RemoteException;

/** Erzeugt ein neues virtuelles Display mit der
  * vorgegebenen Kalibrierung.
  * @param ID: der Name des Displays
  * @param calibData: die Kalibrierungsdaten
  * @throws RemoteException
  */
 public void createDisplay(String ID, CalibrationData
 calibData) throws RemoteException;

/** Liefert das virtuelle Display das unter dem
  * vorgegebenen Namen erzeugt wurde.
  * @param ID: der Name des Displays
  * @throws RemoteException
  * @return: das virtuelle Display
  */
 public VirtualDisplayInterface getDisplay(String ID)
 throws RemoteException;

/** Löscht das virtuelle Display das unter dem
  * vorgegebenen Namen erzeugt wurde.
  * @param ID: der Name des Displays
  * @throws RemoteException
  */
 public void deleteDisplay(String ID)
 throws RemoteException;

/** Zeigt einen Live-Stream auf dem
  * vorgegebenen virtuellen Display an.
  * @param ID: der Name des Displays
  * @param port: der Port, an dem der Live-Stream
  * empfangen wird
  * @throws RemoteException
  */
 public void displayStream(String ID, int port)
 throws RemoteException;
```

```
/** Zeigt ein Bild auf dem
  * vorgegebenen virtuellen Display an.
  * @param ID: der Name des Displays
  * @param url: die URL des Bildes
  * @throws RemoteException
  */
 public void displayImage(String ID, URL url)
 throws RemoteException;

/** Zeigt ein Bild auf dem
  * vorgegebenen virtuellen Display an.
  * @param ID: der Name des Displays
  * @param image: das Image-Objekt
  * @throws RemoteException
  */
 public void displayImage(String ID, Image image)
 throws RemoteException;

/** Zeigt ein Video auf dem
  * vorgegebenen virtuellen Display an.
  * @param ID: der Name des Displays
  * @param url: die URL des Videos
  * @throws RemoteException
  */
 public void displayVideo(String ID, URL url)
 throws RemoteException;

/** Liefert die Namen aller
  * erzeugten virtuellen Displays.
  * @throws RemoteException
  * @return: ein Vektor mit den Displaynamen
  */
 public Vector getDisplayNames() throws RemoteException;

/** Liefert den Mittelpunkt des
  * vorgegebenen virtuellen Displays.
  * @param ID: der Name des Displays
  * @throws RemoteException
  * @return: der Mittelpunkt des Displays
```

```
         */
        public Point3d getMidpoint(String ID)
        throws RemoteException;

       /** Setzt die Eckpunkte des virtuellen Displays neu;
         * die Punkte werden gegen den Uhrzeigersinn
         * angegeben.
         * @param ID: der Name des Displays
         * @param p0: links unten
         * @param p1: rechts unten
         * @param p2: rechts oben
         * @param p3: links oben
         * @throws RemoteException
         */
        public void setPoints(String ID, Point3d p0, Point3d p1,
        Point3d p2, Point3d p3) throws RemoteException;

       /** Bewegt das virtuelle Display auf eine neue Position.
         * @param ID: der Name des Displays
         * @param pos: der neue Mittelpunkt des Displays
         * @param time: die Zeit, in der die Bewegung
         * ausgeführt werden soll
         * @throws RemoteException
         */
        public void moveToPosition(String ID, Point3d pos,
        long time) throws RemoteException;
    }
```

Abbildungsverzeichnis

Literaturverzeichnis

[1] Ubiquitous Computing: http://www.ubiq.com/hypertext/weiser/UbiHome.html

[2] Mark Weisers Homepage: http://www.ubiq.com/weiser/

[3] M. Weiser: The Computer for the 21st Century, *Scientific American, 1991*

[4] Paper-like Displays: http://www.research.philips.com/technologies/display/ov_elpap.html

[5] R. Sukthankar, R. G. Stockton, M. D. Mullin: Smarter Presentations: Exploiting Homography in Camera-Projector Systems, *International Conference on Computer Vision, 2001*

[6] Claudio Pinhanez: The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces, *3rd International Conference on Ubiquitous Computing, 2001*

[7] Amptown Lichttechnik GmbH Homepage: http://www.amptown-lichttechnik.de

[8] Stanislaw Borkowski, Oliver Riff, James L. Crowley: Projecting Rectified Images in an Augmented Environment, *ProCams Workshop, IEEE Computer Society Press, 2003*

[9] FLUIDUM Projekt Homepage: http://www.fluidum.org

[10] publitec Präsentationssysteme & Eventservice GmbH Homepage: http://www.publi.de

[11] JARToolkit Homepage: http://www.c-lab.de/jartoolkit/

[12] Java 3D API: http://java.sun.com/products/java-media/3D/

[13] Java Media Framework API (JMF): http://java.sun.com/products/java-media/jmf/index.jsp

[14] Java Remote Method Invocation (Java RMI): http://java.sun.com/products/jdk/rmi/

[15] OriginLab Homepage: http://www.originlab.com

[16] A. Butz, M. Schneider, M. Spassova: SearchLight - A Lightweight Search Function for Pervasive Environments, *Pervasive, 2004*

[17] M. Schmitz: Ein Framework für räumliches Audio in instrumentierten Umgebungen, *Diplomarbeit*, 2003

[18] Fluid Manager Homepage: http://www.fluidum.org/de/manager.shtml

[19] A. Butz, A. Krüger: A Generalized Peephole Metaphor for Augmented Reality and Instrumented Environments, *International Workshop on Software Technology for Augmented Reality Systems (STARS), 2003*

[20] L. Spassova: Fluid Beam - A Steerable Projector and Camera Unit, *ISWC/ISMAR Newbie and Student Colloquium, 2004*

[21] R. Yang, G. Welch: Automatic and Continuous Projector Display Surface Calibration Using Every-Day Imagery, *9th International Conference in Central Europe in Computer Graphics, Visualization, and Computer Vision, 2001*

[22] F. Devernay, O. Bantiche, È. Coste-Manière: Structured Light on Dynamic Scenes Using Standard Stereoscopy Algorithms, *2002*

[23] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, C. Forlines: iLamps: Geometrically Aware and Self-Configuring Projectors, *ACM SIGGRAPH, 2003*

[24] J. Crowley, F. Bérard, J. Coutaz: Finger Tracking as an Input Device for Augmented Reality, *International Workshop on Gesture and Face Recognition, 1995*

[25] C. von Hardenberg, F. Bérard: Bare-Hand Human-Computer Interaction, *Proceedings of Perceptual User Interfaces, 2001*

[26] R. Kjeldsen, C. Pinhanez, G. Pingali, J. Hartman, T. Levas, M. Podlaseck: Interacting with Steerable Projected Displays, *Proceedings of 5th International Conference on Automatic Face and Gesture Recognition, 2002*

[27] ScanSoft Homepage: http://www.scansoft.de/

[28] CMUSphinx: The Carnegie Mellon Sphinx Project: http://www.speech.cs.cmu.edu/sphinx/

Index