3D-Simulation via Face Tracking

Johannes Ewald
Klinkerberg 32
86152 Augsburg

johannes.ewald@hs-augsburg.de

Stefanie Steinbach
Provinostraße 11 1/22
86153 Augsburg

stefanie.steinbach@hs-augsburg.de

Abstract

Ziel der Anwendung ist ein dreidimensionaler Eindruck beim Betrachten eines computergenerierten Bildes, indem die Kameraposition abhängig von der Gesichtsposition vor dem Bildschirm berechnet wird.

Categories and Subject Descriptors
J.5 [Computer Applications]: Arts and Humanities - *Arts, fine and performing*;
J.0 [Computer Applications]: General;

General Terms
Experimentation, Design

Keywords
Tracking, Face-Tracking, 3D, CG, Computer-Generated, Virtual Reality, OpenCV, Java, Processing

1. Motivation

3D-Kinofilme erfreuen sich immer größerer Beliebtheit. Die Zuschauer werden von der Dreidimensionalität der Bilder stärker in den Bann gezogen als von ihren „platten" Vorgängern.
Es verwundert daher nicht, dass das Interesse nach der dritten Dimension auch bei computergenerierten Bildern exisitiert.

In beiden Szenarien ist allerdings nach wie vor eine spezielle Brille notwendig, die die beiden Bilder für das jeweilige Auge trennt.

Wir möchten mit unserem Programm erste Versuche anstellen, die Illusion der Räumlichkeit nicht durch zwei getrennte Bilder, sondern durch eine betracherabhängige Darstellung zu

erzielen. Die Betrachterposition soll dabei über Facetracking bestimmt werden, so dass außer einer Webcam keine zusätzlichen Geräte notwendig sind.

2. Verwandte Arbeiten

Maßgeblich beeinflusst haben uns folgende zwei Arbeiten:

- *Head Tracking for Desktop VR Displays using the WiiRemote* von Johnny Chung Lee (http://johnnylee.net/projects/wii/)

- *Winscape* von Ryan Hoagland (http://www.rationalcraft.com/Winscape.html)

Beide Projekte haben gemeinsam, dass sie die Illusion der Räumlichkeit nur durch Bewegung erzeugen. Der Betracher muss sich also (leicht) bewegen, damit das Bild als räumlich wahrgenommen wird.
Im Gegensatz zu den Filmen muss also nicht auf eine Shutter- oder Polarisationstechnik zurückgegriffen werden, die wiederum die entsprechenden Brillen voraussetzt. Die Räumlichkeit der Bilder entsteht allein durch die Änderung des dargestellten Bildes, wenn der Betrachter sich bewegt.

Der Bildschirm verhält sich dadurch wie ein Fenster. Er ist lediglich Rahmen einer Szenerie. Bewegt sich der Betrachter so ändert sich auch das Panorama, dass er durch das Fenster sieht.

Alle bisher genannten Techniken haben allerdings den Nachteil, dass immer so etwas wie ein Sensor oder Gerät am Betracher befestigt werden muss. Wäre es daher nicht viel besser, wenn das System den Betrachter selbstständig erkennt und die Darstellung anhand seiner Position anpasst?

In diesem Punkt unterscheidet sich unsere Arbeit von den zwei genannten. Wir möchten mittels Gesichtserkennung ein menschliches Gesicht automatisch erkennen und daraus die Betrachterposition bestimmen. Eine Brille oder ein Infrarotemitter wären danach überflüssig.

3. Anwendungsszenario

3.1 Mögliche Szenarien

Prinzipiell sind alle Szenarien denkbar, in denen eine dreidimensionelle Darstellung den Eindruck bzw. die Intention verstärken kann und in denen nur eine Person mit der Anwendung interagiert. Diese Einschränkung leitet sich aus dem Ansatz ab, da die Darstellung immer nur relativ zu einem Betrachter angepasst werden kann.

Denkbare Szenarien dafür sind:

- Computer- und Konsolenspiele mit 3D Grafik

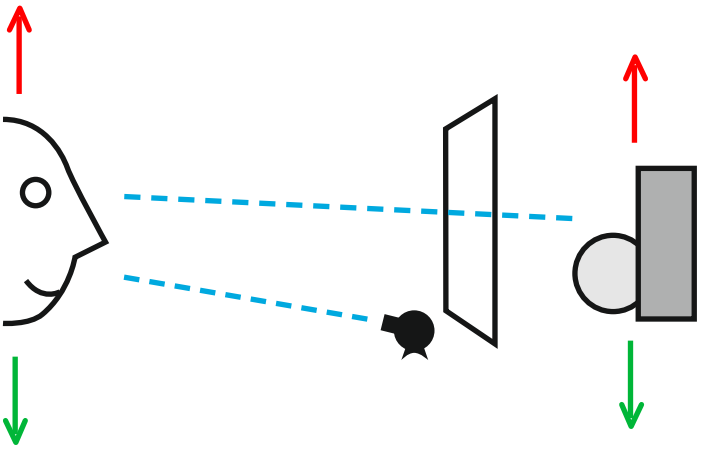
- GUIs, die auf Grund der Komplexität von einem dreidimensionalen Eindruck profitieren könnten

- Interaktive Installationen auf Messeständen oder Kunstaustellungen, die

die oben genannte Einschränkung in Kauf nehmen

3.2 Use-Cases

Betrachter: Bewegt sich
Webcam: Nimmt Bewegung auf
Bildschirm: Stellt die Szenerie in Abhängigkeit von der Bewegung dar



Kontextbedingungen:

- Prinzipiell dürfen mehrere Gesichter im Webcambild sichtbar sein. Um allerdings den Effekt gezielt anzuwenden, sollte zumindest zu Beginn nur die Zielperson zu sehen sein

- Die Tests haben ergeben, dass die Lichtbedingungen sehr schlecht sein dürfen. Ab zu extremen Bedingungen nimmt allerdings die Genauigkeit ab. Eine lichtstarke Webcam ist allerdings nie von Nachteil.

4. Entwurf eines Prototyps

4.1 Verwendete Technologien

Die Hauptarbeit bei der Gesichtserkennung übernimmt das Open-source-Framework OpenCV, welches ursprünglich von Intel entwickelt wurde und nun von vielen Entwicklern weltweit weiterentwickelt wird.

Die Gesichter werden dabei von den Standard Classifiern erkannt, die bei OpenCV mitgeliefert sind. Sie funktionieren bereits so zuverlässig, dass alternative Classifier nicht ausprobiert werden mussten.

Für die dreidimensionale Darstellung konnten wir das Wissen aus dem Fach Computergrafik hervorragend einsetzen.
Die Neuberechnungen der Kameraposition bzw. des Frustums und die Kalibrierung der Werte erforderte eine solide Kenntnis der Materie.

4.2 Konzeptioneller Aufbau

Der Aufbau unserer Anwendung lässt sich am besten durch den Programmfluss beschreiben, der beim Zeichnen eines Frames durchlaufen wird.

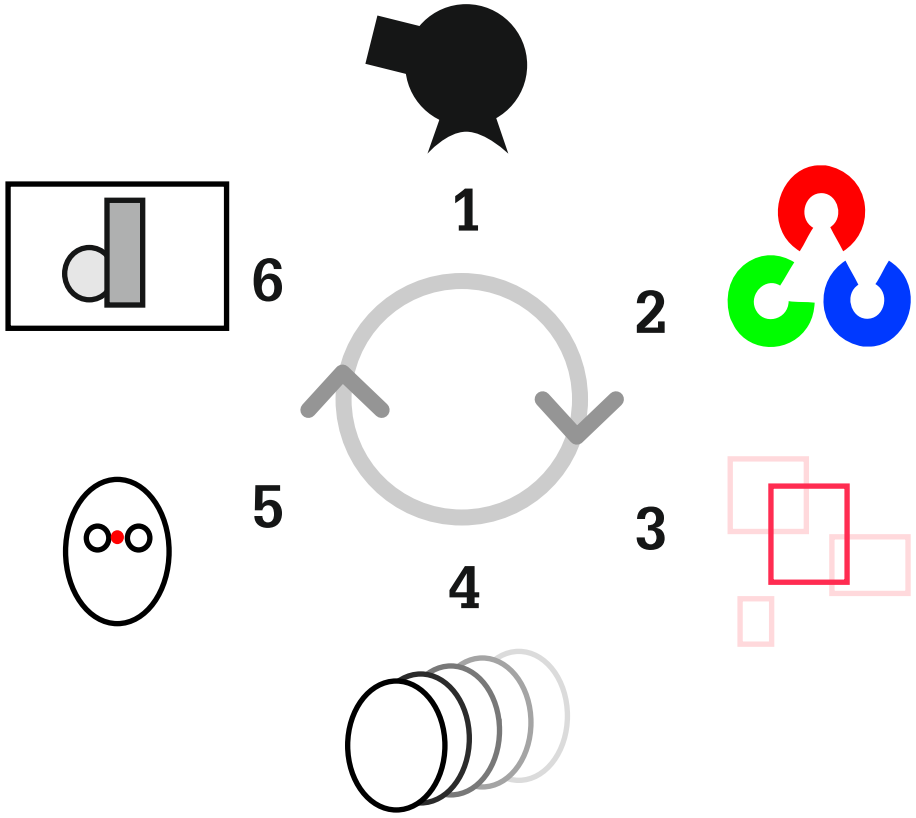
- (1) Das Kamerabild wird aus dem Webcampuffer gelesen

- (2) OpenCV bearbeitet das Bild und liefert die Koordinaten aller gefundenen Gesichter zurück

- (3) Aus diesen Koordinaten wird das wahrscheinlichste Gesicht ausgesucht. Das wahrscheinlichste ist jenes, welches der Position des Gesichts aus dem letzten Frame am nächsten ist

- (4) Die Gesichtskoordinaten werden interpoliert, um kleinere Ungenauigkeiten bei der Erkennung auszugleichen

- (5) Aus den interpolierten Gesichtskoordinaten wird die Position der Augen bzw. die Position zwischen



den Augen bestimmt

- (6) Anhand dieser Koordinaten wird Kameraposition und die Frustummatrix berechnet

- (6) Die 3D Szenerie wird gezeichnet

5. Implementierung

5.1 Hard- und Softwareumgebung
Da wir beide jeweils auf unterschiedlichen Betriebssystemen arbeiten (*WinXP* und *MacOS Leopard*) wurde die Anwendung auf diese beiden Systeme optimiert.

Dies bedeutete für uns allerdings auch, nahezu doppelt so viele Probleme zu lösen, die allein durch die Umgebung entstanden sind.
Im Zusammenspiel der fremden Komponenten entstanden so die meisten Fehlersituationen.

Als Platform diente uns *Processing*, eine leichte Modifikation von *Java*. Das ausschlaggebende Argument für *Processing* war neben der Platformunabhängigkeit vor allem aber die Flexibilität und Schnelligkeit in der Entwicklung.

Processing ist für seinen Ansatz des Rapid Development bekannt. Alle Programme für *Processing* werden in sogenannten Sketches angelegt, also Skizzen. Die Funktionen sind schlank gehalten und für eine unkomplizierte grafische Darstellung optimiert. Da wir sowohl 2D als auch 3D Bildoperationen zu erledigen hatten, war *Processing* für uns optimal.

Dennoch wollten wir nicht auf die Vorteile von einer großen IDE verzichten. Aus diesem Grund entwickelten wir nicht im *Processing* Sketchbook, sondern importierten die Coredateien von *Processing* in *Eclipse* und arbeiteten dort.
Da *OpenCV* in *C* geschrieben ist

und uns deshalb nur als kompilierte Bibliothek zu Verfügung stand, benötigten wir noch einen Wrapper, der die nativen Funktionen in *Java* zugänglich macht. Hier standen uns mehrere Wrapper zur Auswahl, wobei sich einer besonders auszeichnete.

Wir entschieden uns für die *OpenCV* Klasse von Stéphane Cousot und Douglas Edric Stanley (http://ubaa.net/shared/processing/opencv/), da diese bewusst schlank gehalten und außerdem noch gut dokumentiert ist. Eine weitere Möglichkeit wäre *JavaCV* von Samuel Audet gewesen. Diese Bibilothek hätte sogar noch deutlich mehr Funktionen nach Java portiert. Unsere Wahl fiel allerdings auf die *OpenCV* Klasse, da diese speziell für *Processing* optimiert. Sie liefert zum Beispiel als Kamerabild bereits den *Processing* Datentyp PImage zurück.

Prinzipiell ist die Software mit jeder Kamera verwendbar. In unserem Setup verwendeten wir die *PlayStation Eye Camera*, die eigentlich für die für die gleichnamige Spielekonsole entwickelt wurde.
Sony hat allerdings die Kamera derart auf das Tracking spezialisiert, dass sie auch für Computerentwickler von Interesse geworden ist. Es sind daher mitlerweile für alle aktuellen Betriebssysteme selbstgebaute Treiber entwickelt worden.
Durch ihre hohe Bildwiederholungsrate und Lichtstärke kann sie selbst bei schlechten Lichtbedingungen noch auswertbare Bilder liefern.
Der geringe Preis von ca. 35€ macht sie im Vergleich zu anderen Webcams unschlagbar.
Leider war es in unserem Setup beim *Windows* PC nicht möglich, die *PS EyeCam* nativ mit der *OpenCV* Bibliothek anzusprechen. Es musste daher der Umweg über eine kleine Treiberklasse aus dem SDK der Treiberhersteller genommen werden.

5.2 Kern der Implementierung
Einen wichtigen Bestandteil stellt der Stabilizer dar. Seine Aufgabe ist es, aus der Menge der erkannten Gesichter das richtige auszuwählen. Außerdem soll die Position interpoliert werden, um Ungenauigkeiten in der Gesichtserkennung auszugleichen und Bewegungen glatter zu machen. Schließlich soll er auch die Position schätzen können, falls das Gesicht für einen Frame nicht erkannt wurde.

Der Stabilizer teilt dafür das Bild in drei rechteckige Bereiche auf. Einmal das tatsächlich erkannte Gesicht, dann die sogenannte Shaking Area und schließlich die Moving Area. Die letzten beiden Bereiche beziehen sich jeweils auf den vorangegangen Frame und werden jeden Frame neu berechnet.

Nun wird das erkannte Rechteck anhand zweier Algorithmen interpoliert. Befindet sich das Gesicht in...

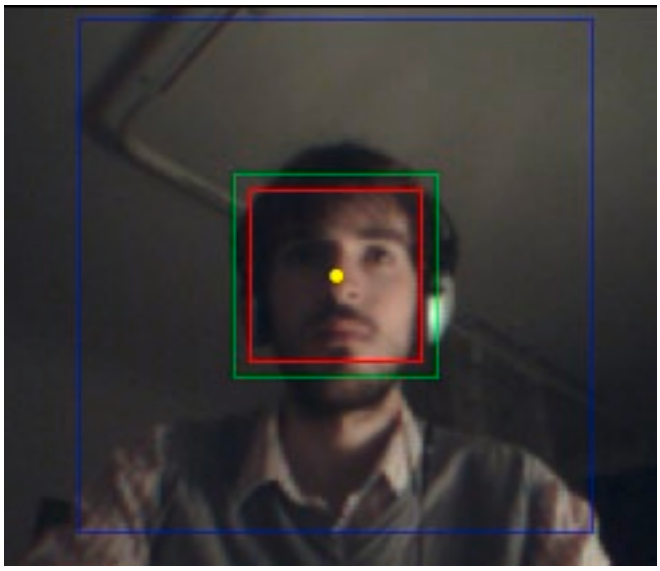
- der Shaking Area, so wird angenommen, dass der Betrachter sich kaum bewegt hat und jegliche Veränderung auf Ungenauigkeiten zurückzuführen sind. Es wird vom neuen Rechteck deshalb nur ein geringer Bruchteil übernommen.

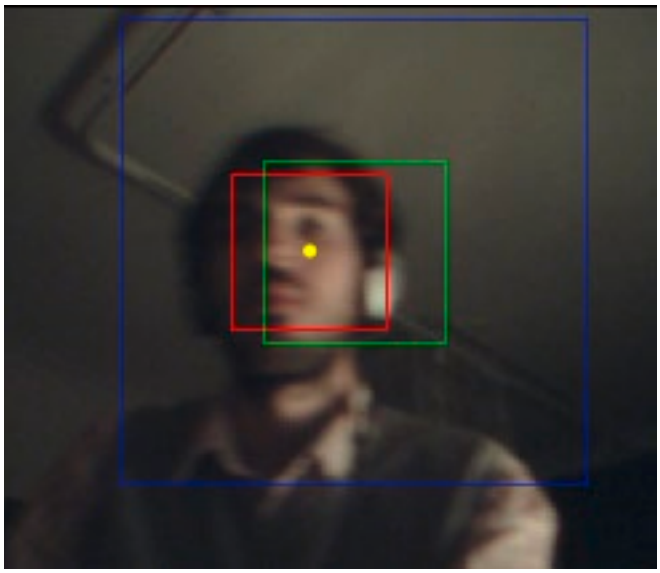
- der Moving Area, so wird angenommen, dass der Betrachter sich bewegt hat. Um Bewegungen zu glätten und stärkere Ungenauigkeiten

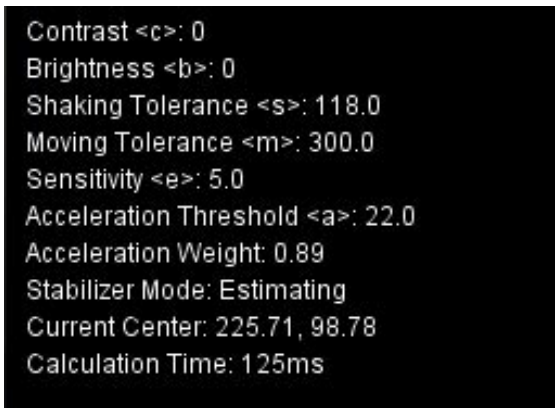
weiterhin zu interpolieren wird nun in Abhängigkeit der Entfernung der alten und der neuen Position ein Beschleunigungs-Gewichtungsfaktor ausgerechnet. Geringe Beschleunigungen werden geringer gewichtet, große Beschleunigungen hingegen mehr. Übersteigt die Entfernung einen gewissen Grenzwert, so wird die neue Position vollständig ohne Interpolation übernommen.
So wird sichergestellt, dass auch schnelle Bewegungen erkannt und verfolgt werden. Die Interpolation bei geringen Beschleunigungen dagegen ist notwendig, um im Grenzbereich zwischen Shaking Area und Moving Area ein abrubtes Springen zu verhindern.

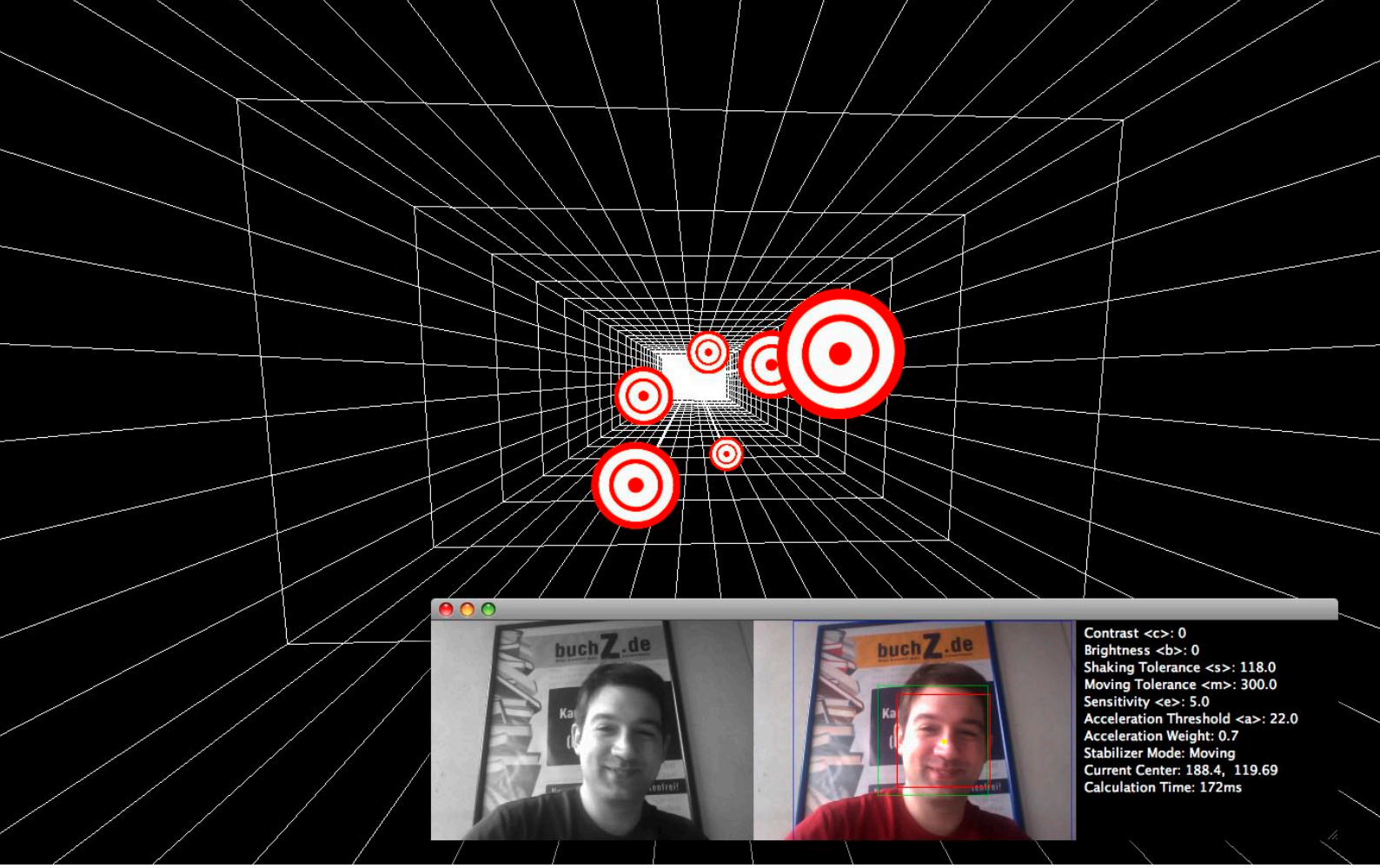
- außerhalb der Moving Area, so wird angenommen, dass es sich um ein neues Gesicht handelt. Der Stabilizer handelt jetzt so, als wäre gar kein Gesicht zu sehen. Er schätzt anhand der alten Geschwindigkeits- und Positionswerte die Position des Gesichtes und zählt intern mit, vor wie vielen Frames das Gesicht zuletzt zu sehen war. Wird ein bestimmter Grenzwert überschritten, so wird die alte Position des Gesichtes aufgegeben und jede neue akzeptiert.

Rot: Das interpolierte Rechteck
Grün: Die Shaking Area
Blau: Die Moving Area









6. Evaluation

6.1 Ziele der Evaluation

Bei der Evaluation unseres Projektes legten wir das Hauptaugenmerk auf die Genauigkeit des Face-Trackings und die damit zusammenhängende Qualität der 3D-Simulation.

Um die Evaluation möglichst objektiv und ohne Beeinflussung durchzuführen ließen wir einige Testpersonen die Simulation ausprobieren. Wir positionierten sie dabei vor den Bildschirm und die dort befestigte Kamera, ohne jegliche Erklärungen und nur mit der Anweisung, in den Bildschirm zu schauen. Dabei sahen sie zunächst nur das Fenster der 3D-Simulation und erst nach kurzer Zeit auch das Fenster mit der bildlichen Darstellung des eigenen Gesichts. Dadurch war es für uns einfacher zu kontrollieren, wie gut das Face-Tracking funktionierte.

Anschließend fragten wir sie als erstes nach dem allgemeinen ersten Eindruck und dann noch einmal spezifisch nach dem Empfinden der 3D-Simulation.
Außerdem testeten wir die Anwendung in unterschiedlichen Lichtverhältnissen und auf verschiedenen Rechnern. Hierbei überprüften wir die Qualität des Face-Trackings in erschwerteren Bedingungen sowie die Wirkung des 3D-Effekts in Abhängigkeit der Rechengeschwindigkeit.

6.2 Durchführung der Evaluation

Unsere Testpersonen besaßen zwischen normalen bis hin zu sehr guten Computerkenntnissen, welches allerdings auf unsere Anwendung wenig Auswirkung hatte, da die eigentliche Mensch-Maschine-Interaktion vom Face-Tracking übernommen wird.

Wir mussten oft beobachten, dass sich die Probanden zu sehr zur Seite bewegten oder den Kopf zu stark neigten, so dass unser Programm nicht immer fähig war, dass Geischt zu erkennen. Darunter litt dementsprechend auch die Wirkung der 3D-Simulation, da die Bewegungen manchmal stockten oder sehr ruckartig wieder einsetzten.

Ansonten konnten alle Testpersonen die 3D-Simulation erkennen und waren vor allem zu Beginn positiv überrascht, da sie mit einem solchen Effekt nicht gerechnet hatten. Erstaunlicherweise funktionierte die Gesichtserkennung auch bei schlechten Lichtverhältnissen recht gut, auch wenn die optimalen Ergbenisse natürlich bei guter Beleuchtung erzielt wurden.

Bei niedrigerer Rechnerleistung musste man schon leichte Qualitätsabstriche bei der 3D-Simulation machen, da die Berechnungen nicht immer schnell genug durchgeführt werden konnten.

6.3 Interpretation der Evaluationsergebnisse

Wir haben bei der Evalutation festgestellt, dass die Grundidee einer 3D-Simulation mit Hilfe von Face-Tracking erfolgreich umgesetzt werden kann. Dennoch ergab das Projekt auch, dass dafür noch einiges verbessert werden muss. Um diese Verbesserungen vorzunehmen, müsste man sich allerdings noch tiefer mit der Materie auseinandersetzen, was uns in diesem Rahmen nicht möglich war.

7. Zusammenfassung

7.1 Erzielte Ergebnisse

Die vorliegende Studie beschäftigte sich mit der Fragestellung, inwiefern man einen dreidimensionalen Eindruck nur durch die Bewegung des Bildes entsprechend der mit HIlfe von Face-Tracking berechneten Position des Betrachters erzielen kann. Als Inspiration dienten dazu ähnliche Anwendungen, die allerdings entweder den 3D-Simulationseffekt beinhalteten oder das Face- bzw. Eye-Tracking, nie jedoch beides vereint. Wir entwickelten einen Prototypen, der beide Komponenten enthielt, und nutzten für die Umsetzung einige Processing und OpenCV Bibliotheken und den darin enthaltenen Classifier zur Gesichtserkennung.
In einer anschließenden Evaluation wurde der Prototyp hinsichtlich der Qualität der Dreidimensionalität und der Gesichtserkennung getestet und eine Nutzerstudie mit 13 Testern durchgeführt. Die Studie ergab, dass es prinzipiell möglich ist, den gewünschten dreidimensionalen Effekt auch ohne Hilfsmittel wie Rot-Grün-Brillen oder ähnlichem und nur durch den Einsatz einer am Computer positionierten Kamera zu erzielen, jedoch für ein optimales Ergebnis noch einiges an Zeit investiert werden müsste.

7.2. Erweiterungsmöglichkeiten / Ausblick

Zunächst müsste die Genauigkeit des Face-Trackings verbessert werden und auch die Classifier noch einmal

überarbeitet, da diese momentan nur Gesichter erkennen, die der Kamera frontal und zumindest fast aufrecht entgegengerichtet sind. Im Zusammenhang damit müsste man nicht nur erkennen, dass das Gesicht im geneigten Zustand noch ein Gesicht ist sondern auch, wie stark es in welche Richtung geneigt ist. Letzteres stellen wir uns allerdings noch recht schwierig vor.
Die Performanceprobleme könnte man durch einen verbesserten Interpolationsalgorithmus verringern. Gerade die Gesichtserkennung über OpenCV beansprucht viele Rechnerzyklen. Eine Möglichkeit wäre daher die Verringerung der Framerate, mit der OpenCV die Webcambilder ausliest. In den dazwischenliegenden Zyklen, in denen kein neues Webcambild vorliegt, könnte die Position anhand der letzten Geschwindigkeit und Position vermutet werden.

Sobald die Gesichtserkennung optimiert ist und die 3D-Simulation entsprechend angepasst und verbessert wird, könnten wir uns vorstellen, dass unsere Anwendung eine Bereicherung in vielen der zuvor genannten Anwendungsszenarien, wie z.B. Computer- und Konsolenspiele, wäre.

Eine weitere interessante Herausforderungen wäre es, nicht nur die Gesichter zu erkennen sondern wiederzuerkennen, so dass die Anwendung weiß, wen sie vor sich hat.

8. Referenzen

[1] *Head Tracking for Desktop VR Displays using the WiiRemote*
http://johnnylee.net/projects/wii/ (Stand: 27.07.2010)
[2] *Winscape*
http://www.rationalcraft.com/Winscape.html (Stand: 27.07.2010)
[3] *OpenCV für Processing*
http://ubaa.net/shared/processing/opencv/ (Stand: 27.07.2010)
[4] *Processing*
http://www.processing.org/ (Stand: 27.07.2010)
[5] *Vorlesung über Tracking*
http://www.hs-augsburg.de/~tr/ss10/IP10/IP-12-Tracking.pdf (Stand: 27.07.2010)
[6] *PS3 EyeCam Windows Treiber*
http://codelaboratories.com/ (Stand: 27.07.2010)