FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Eine berührungslose Benutzerschnittstelle in der Gebäudeautomation und Analyse ihrer Benutzbarkeit

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Michael Jahoda
Matrikelnummer 0325590

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: O. Univ.-Prof. Dipl.-Ing. Dr. Dietmar Dietrich
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Dietmar Bruckner

Wien, 23.9.2013

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Michael Jahoda

Kaiser-Ebersdorfer Straße 79+85/15/15

1110 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit –einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Unterschrift

Wien, 23.9.2013

Kurzfassung

Die Mehrheit der Benutzerschnittstellen im Bereich der Gebäudeautomation basiert auf handbetätigten Steuerelementen. Körperlich eingeschränkten Personen hingegen fehlt oft die Fähigkeit bzw. die Kraft, Schalter oder Knöpfe selbstständig zu betätigen. Eine neue Form der Interaktion mit Systemen der Gebäudeautomation wird benötigt, um Personen mit körperlichen Einschränkungen in ihrem Alltag zu unterstützen. In dieser Arbeit wurde eine berührungslose Benutzerschnittstelle erstellt, die es Benutzern mit einem Mangel an physischer Kraft und mangelnder Bewegungsfreiheit erlaubt, mit klassischen Kontrollelementen der Gebäudeautomation zu interagieren. Die Benutzerschnittstelle kann berührungsfrei mit Kopfgesten bedient werden. Der Benutzer kann mit definierten Gesten, die mit einer Standard-Webcam erfasst werden, mit dem System interagieren. Weiters soll es jedem Benutzer ermöglicht werden, individuelle Befehlsgesten zu definieren, um unabhängig von der Art der Beeinträchtigung, die bestmögliche Benutzbarkeit sicherzustellen. Die Umsetzung der Gestenerkennung basiert auf den empfangenen Daten der Facetracking-Engine faceAPI von SeeingMachines. Die Bewegungen des Kopfs werden erfasst und mit den im System gespeicherten Befehlsgesten verglichen. Wurde eine Geste eindeutig identifiziert, wird der korrespondierende Befehl vom System ausgeführt. Es wurde weiters ein Aufzugssimulator implementiert, mit dem das Verhalten eines echten Aufzugs simuliert werden soll. Anhand des Aufzugsimulators wurde das System mit Hilfe von Testpersonen, unter anderem mit körperlich beeinträchtigten Benutzern, bezüglich ihrer Benutzbarkeit evaluiert. Im Rahmen des Benutzertests hat sich die Praxistauglichkeit dieser Benutzerschnittstelle erwiesen. Durchschnittlich wurden 95% aller durchgeführten Gesten korrekt von der gestenbasierten Benutzerschnittstelle erfasst.

Abstract

The majority of user interfaces in the field of building automation is based on manual interaction. Physically challenged people often lack the ability or the strength to interact with switches or buttons by their own. A new form if interaction with building automation systems is needed to support physically challenged people in their daily routine. In this thesis a non-haptic user interface was created that allows users with a lack of physical strength or an insufficient freedom of movement to interact with typical user interfaces in the field of building automation. The user interface created in this work can be operated by head movements of a user. Users can interact with the system by defined head gestures that are recorded by a standard-webcam. To ensure a better usability of the system, users also have the possibility to define individual head gestures. The recognition and tracking of the users head was implemented with the facetracking engine faceAPI by SeeingMachines. The movements of the head are apprehended and compared with the defined gestures. Is a gesture identified, the corresponding command is executed by the system. To simulate the behavior of an elevator, a visual elevator similar was implemented. With the support of test persons, some of them physically challenged, the user interface was evaluated regarding the usability of the system. In the course of the tests the practical utility of the system was proved. On average, 95% of the executed gestures were recognized by the gesture based user interface.

Inhaltsverzeichnis

Abkürzungen

API	Application Programming Interface
DOF	Degrees of Freedom
TCP/IP	Transmission Control Protocol / Internet Protocol
XML	Extensible Markup Language

1. Einleitung

Gestenbasierte Computerinteraktion erlebt aktuell weltweit einen großen Aufschwung. Meist durch Entwicklungen in der Heimelektronik hält die gestenbasierte Steuerung von Geräten immer mehr Einzug in die Privathaushalte. Dank gestengesteuerter Fernseher und Hifi-Anlagen (z. B. [Pre10]) sowie gestenbasierter Eingabesysteme in der Videospielbranche, wie die Kinect-Technologie von Microsoft[1], gelten berührungsfreie, gestenbasierte Steuerungen heutzutage nicht mehr als Utopie, sondern setzen sich, auch auf Grund ihrer gesteigerten Zuverlässigkeit, immer mehr im Alltag durch. Auch Maus- bzw. Fingergesten haben sich in den letzten Jahren immer mehr im Alltag, nicht zuletzt durch den großen Erfolg moderner Smartphones, durchgesetzt. Aufgrund der hohen Komplexität sind gestenbasierte berührungsfreie Steuerungs- und Interaktionssysteme noch nicht weit verbreitet. Erste praxistaugliche Umsetzungen solcher berührungsfreien Benutzerschnittstellen gibt es vor allem im Bereich der Unterstützung körperlich eingeschränkter Personen. Der Einsatz von Gesten zur Steuerung eines Rollstuhls wurde in [Jia07], [Mat01], [Bai07] und [Yut07] beschrieben. Insbesondere im Bereich der Gebäudeautomation gibt es in diesem Bereich aber noch enormes Entwicklungspotential.

Häufig ist zur Verwendung eines gestenbasierten, berühungsfreien Steuerungssystems spezielle Hardware und darauf aufbauende Software erforderlich. SeeingMachines[2] hat mit der Facetracking-Engine „faceAPI“ 2008 erstmals eine Software zur dreidimensionalen Erfassung eines Kopfs vorgestellt, die auf üblichen PCs in Kombination mit einer handelsüblichen Webcam funktioniert. Mit Hilfe dieser Software ist es möglich, neben diversen weiten Anwendungsmöglichkeiten wie zum Beispiel der Einsatz in Driver Assistence Systemen (siehe [Mur07b], [Dos09]), die Daten dieser Software für eine Interaktionsmöglichkeit mit einer Benutzerschnittstelle umzusetzen.

Im Rahmen dieser Diplomarbeit soll mit Hilfe der Facetracking-Engine faceAPI eine berührungsfreie Benutzerschnittstelle für körperlich eingeschränkte Personen im Kontext der Gebäudeautomation erstellt und bezüglich ihrer Benutzbarkeit mit Hilfe eines Aufzugssimulators untersucht werden. Es sollen mit einer Webcam die Bewegungen eines Kopfes (Gesten) erfasst, und diese Bewegungen in konkrete Befehle interpretiert werden.

[1] http://www.xbox.com/de-AT/Kinect

[2] www.seeingmachines.com

Diese Arbeit ist wie folgt strukturiert. In Kapitel 2 werden die Problemstellung sowie die konkreten Ziele dieser Diplomarbeit formuliert. Aktuelle Entwicklungen im Bereich der berührungsfreien Benutzerschnittstellen, Grundlagen des Facetrackings, Facetrackingmethoden sowie ein Vergleich der verschiedenen Facetracking-Bibliotheken werden in Kapitel 3 nähergebracht. Kapitel 4 beschäftigt sich mit der praktischen Umsetzung der Benutzerschnittstelle und des Aufzugssimulators. Es wird die Systemarchitektur und die Implementierung der berührungsfreien Benutzerschnittstelle sowie des Aufzugssimulators beschrieben. Weiters wird aufgezeigt, wie die Kommunikation zwischen den verschiedenen Systemmodulen - von Kopfgesten in konkrete, maschinenverständliche Befehle - aufgebaut ist. In Kapitel 5 wird beschrieben, wie die kopfgestengesteuerte Benutzerschnittstelle mit Hilfe eines Benutzertests evaluiert wurde und ob sich die Praxistauglichkeit erwiesen hat. Im abschließenden Kapitel 6 werden die Ergebnisse dieser Diplomarbeit diskutiert sowie verfügbare Entwicklungspotentiale aufgezeigt.

2. Problemstellung, Ziel und Methode

Es wurden schon zahlreiche Ansätze verwirklicht, körperlich eingeschränkten Personen das Alltagsleben möglichst zu vereinfachen. Innerhalb der eignen Wohnung gibt es schon diverse Möglichkeiten und Hilfestellungen, die „einfachen“ Aufgaben des Alltags auch ohne Hilfspersonen bewältigen zu können (z. B. [Kim06], [Yam04]). Begibt man sich aber außerhalb der eignen Wohnung, ist es für zahlreiche Aufgaben nahezu unmöglich, diese ohne Assistenz erledigen zu können. Seien es bauliche Gegebenheiten wie Stufen, Rolltreppen, nicht abgeschrägte Gehsteigkanten oder diverse Benutzerschnittstellen, die man im Laufe eines Tages als nicht-eingeschränkte Person wie selbstverständlich benutzt, aber im Falle einer staken körperlichen Einschränkung (z. B. Lähmung) nicht bedienen kann. Beispiele hierfür wären das Betätigen des Tasters einer Ampelanlage, das Öffnen einer Straßenbahn- oder U-Bahn-Tür oder das Rufen und Bedienen eines Aufzuges anhand der vor dem und im Aufzug angebrachten Bedienelemente. Zur Vereinfachung des Lebens körperlich eingeschränkter Personen wäre eine Benutzerschnittstelle erforderlich, die möglichst ohne manuelle Interaktion bedienbar ist.

Ist man nur leicht körperlich eingeschränkt oder sitzt man im Rollstuhl, ist man mit zahlreichen Barrieren konfrontiert. Ist man durch starke körperliche Einschränkungen oder Lähmung in der Bewegungsfreiheit der Arme eingeschränkt, kann ein selbstständiges Leben oft nur durch technische Hilfsmittel ermöglicht werden.

Das Ziel dieser Diplomarbeit ist es, eine berührungsfreie Benutzerschnittstelle zu implementieren, die es Personen mit einem Mangel an physischer Kraft und mangelnder Bewegungsfreiheit erlaubt, mit klassischen Kontrollelementen in der Gebäudeautomation zu interagieren. Diese Benutzerschnittstelle soll dazu dienen, sowohl innerhalb als auch außerhalb des Wohnraums ohne Hilfe durch unterstützende Personen wie Pflegepersonal oder Familienmitglieder Mobilität und Selbständigkeit im täglichen Leben zurückzugewinnen. Konkret sollen im Rahmen dieser Diplomarbeit folgende Punkte umgesetzt werden:

- Entwicklung und Implementierung einer berührungsfreien Benutzerschnittstelle mit Hilfe eines vorgegebenen Facetracking-Systems FaceAPI unter Verwendung von Standard-Hardware
- Schaffen einer Möglichkeit, dem System Befehle individuell zu lernen – mit der Möglichkeit, auch mehrere Benutzer anlegen zu können
- Entwicklung und Implementierung eines Aufzugssimulators zum Testen der Schnittstelle
- Evaluierung der Benutzbarkeit der Benutzerschnittstelle im Form eines Benutzertests

Eine der wichtigsten Ziele der Implementierung ist die Modularität der Steuerung, sodass der Einsatz dieser Steuerung auch außerhalb der geplanten Nutzung als Aufzugssteuerung möglich ist. Mögliche Einsatzgebiete können „Benutzerschnittstellen“ innerhalb der Wohnung, wie beispielsweise Lichtschalter, Bedienung des Sonnenschutzes, Heizungssteuerung und dergleichen sein. Außerhalb der Wohnung kann die Steuerung beispielsweise bei Ampelanlagen, Türöffnungssysteme, Gegensprechanlagen und dergleichen eingesetzt werden.

In dieser Arbeit wird eine berührungslose Benutzerschnittstelle erstellt, die es Benutzern mit einem Mangel an physischer Kraft und mangelnder Bewegungsfreiheit erlaubt, mit klassischen Kontrollelementen der Gebäudeautomation zu interagieren.

Es soll ein Aufzugssimulator implementiert werden, der bei der Definition und Evaluierung möglicher Kommandosets zur Interaktion mit einer existierenden Aufzugssteuerung unterstützt. Hierfür wird eine visuelle Benutzerschnittstelle erstellt, welche basierend auf der Methode des Face-Trackings (eingebettet in der Engine faceAPI von SeeingMachines) Befehle interpretieren kann. faceAPI stellt eine Library zur Verfügung, welche mit Hilfe einer Webcam ein menschliches Gesicht erkennen und Daten über Position und Bewegungen des Gesichts erfassen kann. Diese Daten können anschließend weiterverarbeitet werden. Teil dieser Arbeit soll ebenfalls ein Usability-Test sein, der die Gebrauchstauglichkeit der entwickelten Benutzerschnittstelle analysieren soll.

Eine der größten Herausforderungen dieser Diplomarbeit war es, eine Lösung anhand von üblicher Standardtechnologien, im konkreten Fall wurde ein Notebook mit einer Standard-Webcam zur Gestenerfassung benutzt, zu realisieren. Die Benutzerschnittstelle sollte sich an die individuellen Bedürfnisse der einzelnen Benutzer, vor allem der körperlich eingeschränkten, anpassen und fehlertolerant sein. So sollte bei normal durchgeführten Bewegungen, die nicht als Eingabe gemeint sind, kein Befehl ausgeführt werden.

3. Grundlagen und State of the Art

Neben den klassischen Interaktionsformen wie Maus/Tastatur, Knöpfe und Tastern erfreuen sich gestenbasierte Interaktionssysteme in den letzten Jahren immer größerer Beliebtheit. Durch den technischen Fortschritt werden gleichzeitig auch technische Systeme zur Unterstützung körperlich beeinträchtigter Personen, vor allem in der Heim- und Gebäudeautomation, bereits öfter eingesetzt. Es liegt dabei nahe, diese beiden Entwicklungsgebiete miteinander zu kombinieren. In diesem Kapitel werden, neben den allgemeinen Stand der Forschung im Bereich gestenbasierter Interaktionsformen, auch aktuelle Konzepte in der Gebäudeautomation beschrieben und wie diese System zur einfacheren Absolvierung des täglichen Lebens körperlich beeinträchtigter Personen beitragen können.

Da die Steuerung des in dieser Arbeit implementierten Systems auf Basis von Kopfgesten erfolgt, werden weiters existierende Facetrackingsysteme verglichen sowie verschiedene Methoden zur Abschätzung der aktuell eingenommen Pose angeführt.

3.1 Systeme der Gebäudeautomation

Im Bereich der Gebäudeautomation und der Automatisierung von Vorgängen im Haushalt wurden in den letzen Jahren zahlreiche Projekte zur Verbesserung der Selbstständigkeit körperlich beeinträchtigter Personen umgesetzt, die sich als praxistauglich erwiesen.

In einem Haushalt stehen zahlreiche Anwendungsmöglichkeiten zur Verfügung, um einen Großteil der in der Regel manuell zu betätigenden Steuerungselemente automatisiert zu steuern. Allgemein können folgende aktive (manuelle Interaktion erforderlich) sowie passive (keine manuelle Interaktion erforderlich, meist Datenerfassung und Überwachung) Anwendungsmöglichkeiten für Lösungen der Gebäudeautomation zur Verfügung stehen (vgl. [2]):

Aktiv

- Beleuchtung bedarfs-, tageszeit- bzw. jahreszeit- und bewegungsabhängig schalten bzw. dimmen
- Heizung, Lüftungsanlage oder Klimaanlage bedarfs- und zeitgerecht steuern
- Verschattungseinrichtungen in Abhängigkeit von Sonnenlicht und Wind zeit- und bedarfsgerecht steuern

- Zutrittskontrollsysteme realisieren
- schalten bzw. dimmen mit Funk- oder Infrarotfernbedienung
- Laststeuerung auf Basis der Verbrauchsdatenerfassung durch sequenzielles Einschalten von Beleuchtungen
- Steuern der Mediengeräte, Multiraumsysteme in den Schulungs-, Seminar- und Medienräumen
- Steuerung elektrischer Geräte des Alltags, wie Kaffeemaschine oder Radio

Passiv

- alle Steuerungsvorgänge im Gebäude zentral erfassen und anzeigen
- Sicherheit erhöhen durch die Überwachung von Fenster- und Türkontakten, sowie von Bewegungsmeldern
- Fernüberwachung und Fernsteuerung über das Telefonnetz oder über das Internet (Fernwirken)
- Verbrauchsdatenerfassung von Wärmezählern, Wasserzählern, Gaszählern und Stromzählern.
- Simulation von Anwesenheit erhöht Sicherheit

Bei diesen zahlreichen Anwendungsmöglichkeiten kann gerade die Anwendung eines unterstützenden Interaktionssystems für körperlich eingeschränkte Personen im Bereich der aktiven Anwendungsmöglichkeiten von hohem Nutzen sein.

3.2 Berührungsfreie Interaktionsformen

In [Mit07] werden folgende gestenbasierte Interaktionsmethoden definiert:

1) Hand- und Armgesten: Erkennung von Handposition und Zeichensprache.
2) Kopf- und Gesichtsgesten: z. B. Nicken, Kopfschütteln. Zu diesen Gesten zählen auch das Erkennen der Blickrichtung, Aktionen der Gesichtsmerkmale (Augenbrauen, Mund, Blinzeln) und emotionale Ausdrücke (Freude, Trauer, Angst, etc.).
3) Körpergesten: Bewegungen des gesamten Körpers.

Während Hand- und Armgesten sowie Kopf und Gesichtsgesten primär für eine direkte Interaktion mit bestimmten Systemen vorgesehen sind, wird die Erkennung von Körpergesten primär zur Analyse bestimmter Bewegungsmuster und Bewegungsabläufe eingesetzt.

Eine weitere Form berührungsfreier Interaktion stellt die Spracherkennung dar. Hier kann durch gesprochene Befehle mit Systemen interagiert werden.

Eine der ersten praktischen Umsetzungen einer berührungsfreien Benutzerschnittstelle wurde in den frühen 90er Jahren implementiert. [Shaw90] stellte 1990 mit „The Eye Wink Control Interface“ (EWCI) eine durch Öffnen und Schließen der Augen steuerbare Benutzerschnittstelle vor. Mit dieser Benutzerschnittstelle war es möglich, einen Rollstuhl mit Hilfe von durch Blinzeln ausgelöste Befehle durch ein Labyrinth zu bewegen. Für die Erkennung, ob ein Auge geschlossen oder offen war, wurde eine Brille mit eingebauten Infrarotsensoren eingesetzt. In [Kauf93] wurde ein paar Jahre später eine Benutzerschnittstelle vorgestellt, die mit Hilfe von Augenbewegungen bedienbar war. Die Augenbewegungen wurden via Elektrookulografie (EOG) erfasst. Bei dieser Methode werden Elektroden in der Nähe der Augen platziert, die die durch Augenbewegungen verursachten Spannungsänderungen der Haut erfassen. Mit dieser Methode war es möglich, eine Vielzahl an Augengesten zu verarbeiten (z. B. Blinzeln, Augenbewegungen, etc.). Bei durchgeführten Tests lag die Erkennungsrate zwischen 73% und 92% bei komplexeren Menüs (3x2 Elemente), eine höhere Erkennungsgenauigkeit (90%-99%) konnte bei vereinfachten Menüdesigns erreicht werden.

Ein System zur Erkennung von Handgesten unter der Verwendung mehrerer Kameras wurde in [Uts96] vorgestellt, welches die Grundlage darstellte in [Uts99] ein Trackingsystem zur Erfassung der Pose mehrere Hände einzusetzen. Mit Hilfe dieses Trackingsystems wurde ein Anwendungsfall realisiert, bei dem sieben verschieden Gesten bei einer Erkennungsrate von 92%-100% vom System erkannt wurden. Mit Hilfe dieses Systems war es möglich, mit virtuellen 3-dimensionalen Objekten zu interagieren (z. B. „Erzeugen“, “Löschen“, „Farbe und Textur ändern“, etc.).

Eine der ersten berührungsfreien Benutzerschnittstellen, die ohne tragbare Hardware bedienbar waren, wurde in[Tak95] implementiert. Augenbewegungen wurden in Kombination mit Kopfbewegungen erfasst, die Bewegungen selbst hingegen mit einer Kamera und einer Bildverarbeitungssoftware. Zur Steuerung des Systems musste die gewünschte Funktion aus einer Menge an verfügbaren Funktionen, die auf einem Bildschirm durch quadratische Elemente dargestellt wurden, mit den Augen fokussiert werden. In der Benutzerschnittstelle, die in [Tak96] veröffentlicht wurde, kam ebenfalls eine Steuerung über Kopfgesten zum Einsatz. Es wurde ein ähnliches Menü wie in [Tak95] verwendet. Allerdings mussten die Benutzer eine Brille mit drei daran befestigten Leuchtdioden tragen. Anhand der Position der Dioden wurden die Bewegungen vom System erkannt. Dieses konnte erfolgreich mit körperlich eingeschränkten Personen getestet werden.

Das Konzept der durch Armgesten steuerbaren Benutzerschnittstelle verfolgte [Yam04] im Jahr 2004. Die Erfassung der Arme und deren Lage sowie deren Position erfolgte durch mehrere Kameras. In einem Testaufbau mit vier Kameras, die in den Ecken positioniert wurden, wurde die Oberfläche eines Raums in 20 gleich große Elemente aufgeteilt. Zeigte der Arm auf eines dieser virtuellen Elemente, wurde der entsprechende Befehl ausgeführt. So konnte beispielweise ein Fernseher eingeschaltet werden, indem mit dem Arm auf diesen gezeigt wurde. Mit zusätzlichen Armgesten, zum Beispiel durch Bewegen des Arms nach links oder rechts, konnte mit dem Fernseher interagiert und so etwa der Kanal erhöht oder vermindert werden.

Eine weitere Möglichkeit zur berührungsfreien Interaktion für körperlich beeinträchtigte Personen wurde in [LB06] implementiert. In diesem Ansatz sollte es möglich sein, den Mauszeiger eines Computers mit Hilfe von Kopfbewegungen zu steuern. Einen ähnlichen Ansatz verfolgte [Bet02].

Hier konnte ebenfalls der Mauszeiger eines Computers gesteuert werden. Bei dieser Lösung ist es, neben der Steuerung mit Kopfbewegungen, auch möglich, den Mauszeiger mittels eines Fingers zu bewegen.

Zur Erhöhung der Unterstützung der Mobilität körperlich beeinträchtigter Personen wurden schon zahleiche Projekte realisiert, in denen ein Rollstuhl mit berührungsfreien Benutzerschnittstellen gesteuert werden konnte (siehe [Jia07], [Mat01], [Bai07] sowie [Yut07]). Ein Ansatz, in dem eine bewegliche Prothese mit Hilfe von Kopfbewegungen gesteuert werden konnte, wurde in [Bak08] dargestellt. Welche Ansätze im Gebiet der Gebäudeautomation existieren, wird im folgenden Kapitel vorgestellt.

3.3 Konzepte in der Gebäudeautomation zur Unterstützung körperlich beeinträchtigter Personen

In [Ada13] werden folgende Möglichkeiten beschrieben, körperlich beeinträchtigte Personen in ihrem täglichen Leben zu unterstützen:

- **Mobilität und Unterstützung bei Tätigkeiten im Haushalt**

  Eine körperlich eingeschränkte Person sollte sich selbstständig im Haushalt bewegen und Tätigkeiten des Alltags (z. B. kochen, Wäsche waschen, duschen) verrichten können.

- **Steuerung von Heimelektronik**

  Es soll körperlich eingeschränkten Personen möglich sein, mit Elementen der Heimelektronik zu interagieren. Dies beinhaltet neben Musik- und Videowiedergabegeräten auch die Steuerung von Licht, Tür und Fenstermechanismen, etc.

- **Unterstützung bei Sicherheit, Gesundheit und in Notfällen**

  In gesundheitlichen oder sicherheitstechnischen Notfällen sollen körperlich eingeschränkte Personen um Hilfe rufen können.

- **Kommunikation**

  Um die soziale Isolation zu vermindern soll es körperlich eingeschränkten Personen möglich sein, mit Freunden, Familie, aber auch der ganzen Welt zu kommunizieren.

Um die zentrale Steuerung dieser Systeme zu ermöglichen, wird ein Netzwerk mit entsprechenden Schnittstellen benötigt. Dieses Netzwerk wird meist in Form eines Feldbus-Systems realisiert, an dem die einzelnen Geräte angeschlossen werden können. An ein Feldbussystem können sowohl Aktoren als auch Sensoren angeschlossen werden. [Die00] nimmt schon 2000 an, dass die Anzahl der Feldbus-Knoten in allen Systemen massiv ansteigen wird. In [Die10] werden folgende Anforderungen bzw. Ziele an ein Feldbus-Systems für den Einsatz in der Gebäudeautomation definiert: Kostenoptimierung, verbesserte Energieeffizienz, gesteigerte Sicherheit sowie erhöhter Komfort.

Zur Erhöhung des Komforts sowie der Selbstständigkeit körperlich eingeschränkter Personen wird in [Roes10] und [Zer11] ein System vorgestellt, dass auf Basis eines Feldbus-Systems (LON-Bus) eine praktische Umsetzung einer gestenbasierten Aufzugssteuerung ermöglicht. Mit fixen Kopfgesten kann die Benutzerschnittstelle einer Aufzugssteuerung bedient werden. Die generierten Befehle werden über ein Feldbussystem an den Aufzug gesendet und dort ausgeführt (z. B: „Fahre in den 1. Stock“).

Ein anderes Konzept verfolgt das Projekt ATTEND in [Bru09]. Hier soll über verschiedenste in einem Haushalt installierte Sensoren das Verhalten und der Tagesablauf der im Haushalt lebenden Person laufend erfasst werden. Aus den über einen Zeitraum oder laufend gesammelten Sensordaten generiert das System automatisch ein Modell (Hidden Markov Model; [Rab86]), auf Basis dessen es möglich ist, zwischen verschiedenen Szenarien zu unterscheiden. Wird nun ein Verhalten beobachtet, dass vom gelernten „Normalverhalten“ abweicht, kann im Notfall Hilfe gerufen werden [Bru10].

3.4 Facetracking

Unter Facetracking versteht man die automatisierte Messung bzw. Verfolgung der Lage und Position eines Kopfs in Relation zu einem Bilderfassungsgerät, in der Regel durch eine Kamera (engl. „Head Pose Estimation“). Zur Abschätzung der aktuellen Kopfposition stehen mehrere Methoden zur Verfügung, die im Folgenden näher beschrieben werden.

3.4.1 Methoden der Head Pose Estimation in Facetracking-Systemen

Die exakte Abschätzung der Kopfposition im Raum (z. B. Abstand zum Bilderfassungsgerät) bzw. der Pose (Lageveränderung des Kopfs, z. B. drehen) ist ein komplexes Problem der Bildverarbeitung. Es muss nicht nur ein Kopf innerhalb eines Bilds, sondern auch dessen Position im Raum bzw. Pose erkannt werden. Im Folgenden werden nun acht mögliche Methoden zur Kopfposenschätzung (evaluiert und zusammengefasst in [Mur09]) näher beschrieben, auf Basis dessen Facetracking-Systeme implementiert werden:

- Vorlagenvergleich
- Detektor-Array Methode
- nichtlineare Regression
- Mannigfaltigkeitseinbettung
- flexible Methoden
- geometrische Methoden
- Trackingmethoden
- hybride Methoden

Vorlagenvergleich

Das aktuell empfangene Bild wird mit einer Menge von Referenzbildern verglichen, wobei jedem Referenzbild eine konkrete Pose schon im Vorhinein zugewiesen wurde. Diese Methode ermittelt das dem aktuellen Bild ähnlichste Referenzbild und liefert die dazu korrespondierende Pose zurück [Niy96] (siehe Abbildung 1).

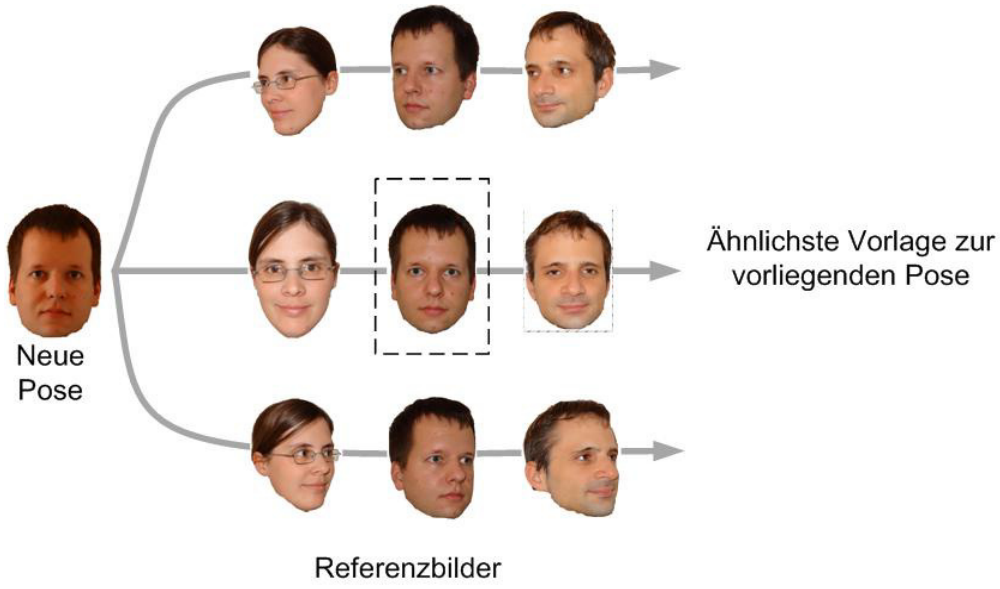

Abbildung 1 – Vorlagenvergleich: Vergleich eines neuen Bilds mit abgespeicherten Referenzbildern

Ein Vorteil dieser Methode ist die Adaptierbarkeit der Menge der Referenzbilder im Falle von geänderten Umgebungsbedingungen. Nachteile dieser Methode sind unabdingbare Notwendigkeit von Interpolationsalgorithmen für eine granulare Kopfposenschätzung sowie Effizienzprobleme aufgrund des rechnungsintensiven Bildervergleichs mit einer großen Menge der zu durchsuchenden Referenzbilder [Lab08].

Detektor-Array Methode

Detektor-Arrays basieren auf der oben beschriebenen Methode des Vorlagenvergleichs. Im Vergleich zur vorigen Methode wird hier eine Menge von Gesichtsdetektoren für jeweils eine bestimmte Pose spezialisiert (siehe Abbildung 2). Je ein Detektor umfasst alle für eine bestimmte Pose klassifizierten Referenzbilder, die durch einen übergeordneten Lernalgorithmus dem System gelehrt wurden. Der Detektor mit dem größten Ausschlag liefert die zur aktuellen Kopf-Ansicht korrespondierende Pose.

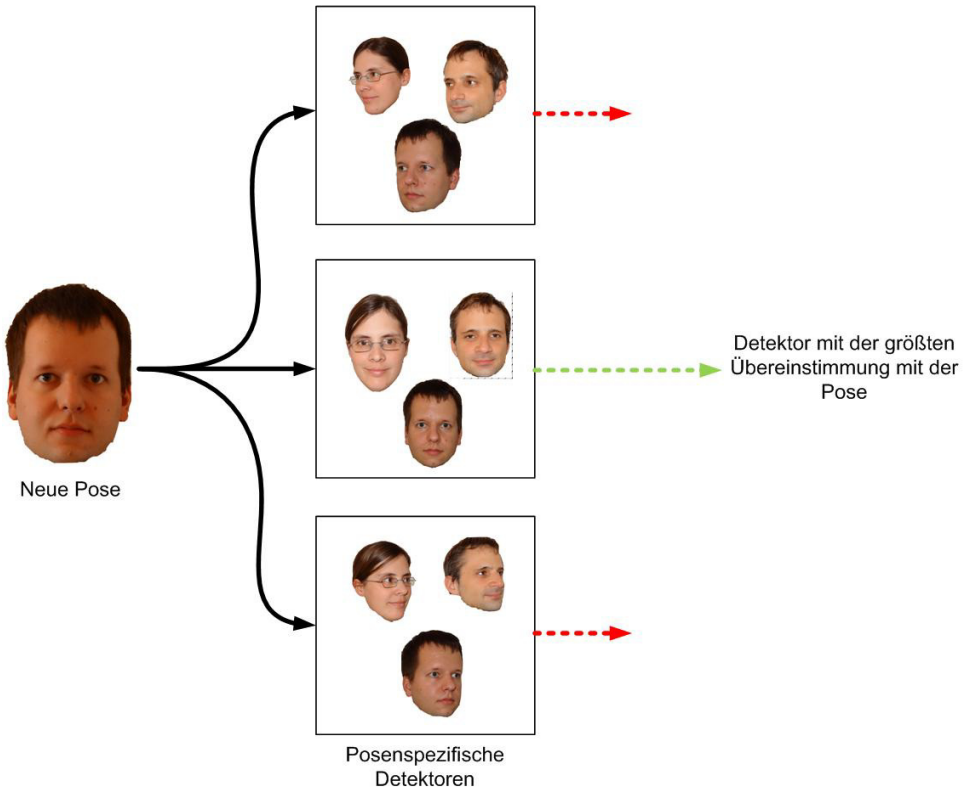

Abbildung 2 - Detektor-Array Methode: Mehrere Detektoren sind auf eine bestimmte Pose spezialisiert

Einer der Nachteile bei dieser Methode ist der hohe Aufwand, um viele Detektoren entsprechend jeder einzelnen Pose zu trainieren. Der notwendige hohe Rechenaufwand bei einer größeren Menge an Detektoren erschwert bei dieser Methode die Realisierung eines in Echtzeit arbeitenden Systems. Für diese Methode spricht der Wegfall eines getrennten Schritts zur Kopferkennung, da jeder Detektor des Arrays in der Lage ist, einen Kopf eindeutig als solchen zu identifizieren. Im Gegensatz zur Vorlagenvergleichs-Methode ist ein Detektor mit Hilfe von Trainings-Algorithmen auch in der Lage zu lernen, Änderungen in der Erscheinung von einer Änderung der Pose zu unterscheiden.

Nichtlineare Regression

Die Methode der nichtlinearen Regression basiert auf Schätzung des Mappings eines Bilds zu einer Pose (siehe Abbildung 3). Anhand von nichtlinearer Regression [Mur07a] oder der Anwendung von neuronalen Netzwerken [Voi07] kann so bei einer gegebenen Menge von mit einer Pose gekennzeichneten Bildern, die Beziehung zwischen Bild und Pose gelernt werden [Bai09]. Mit diesen Daten kann ein Modell erzeugt werden, mit Anwendung dessen eine Kopfposenschätzung für jeden neuen Input durchgeführt werden kann.

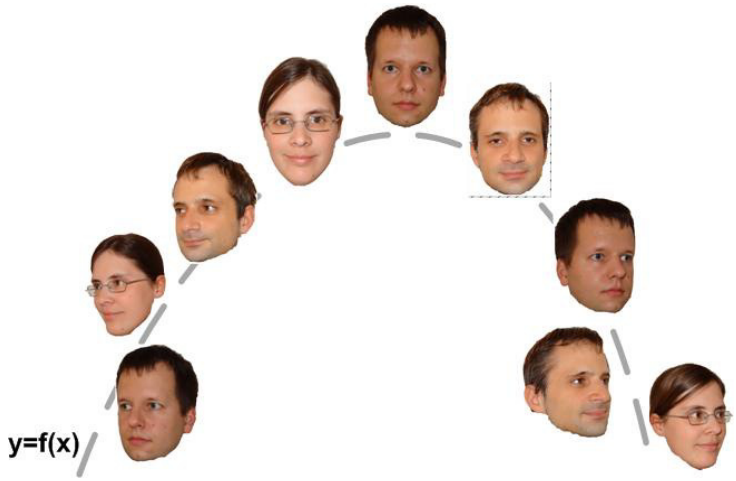

Abbildung 3 – nichtlineare Regression: Funktionale Mapping von einem Bild zu einer diskreten Position

Ein Nachteil dieser Methode ist, dass wie bei der Detektor-Array Methode bzw. dem Vorlagenvergleich nur diskrete Posenschätzungen möglich sind. Der Vorteile dieser Methode bei der Verwendung von neuronalen Systemen ist, dass diese äußerst schnell arbeiten. Sie liefern, auch unabhängig von der Entfernung zur Kamera, die genauesten Ergebnisse in der Praxis [Mur09].

Manifold Embedding

Bei Anwendung der Methode des Manifold Embedding werden niedrigdimensionale Mannigfaltigkeiten gesucht, in denen höher dimensionale Abbildungen eines Kopfes (3-dimensional) projiziert werden können [Mur09] (siehe Abbildung 4). Durch die Lage des neuen Inputs auf der Mannigfaltigkeit kann auf die Pose geschlossen werden.

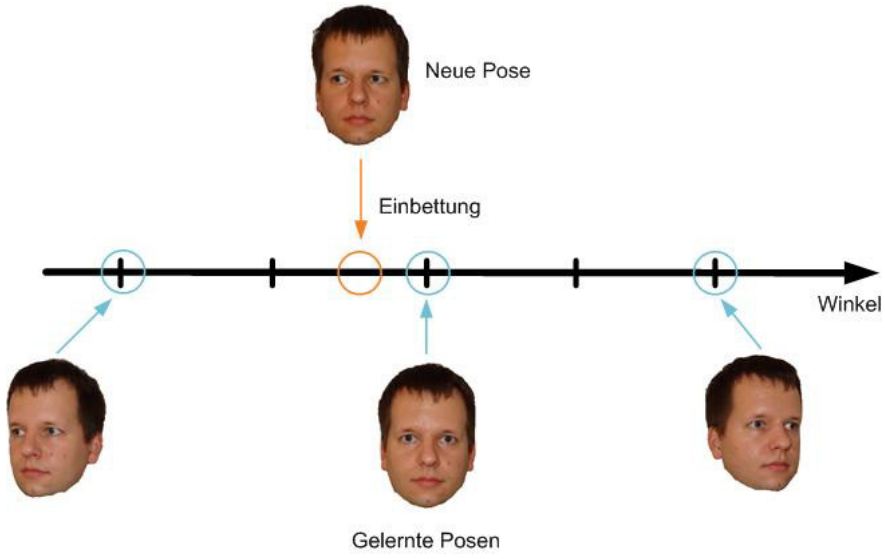

Abbildung 4 – Manifoldembedding: Einbettung des neuen Kopfes in die Mannigfaltigkeit

Flexible Methoden

Ein flexibles Modell, bestehend aus einem Graphen mit mehreren untereinander verbundenen Punkten, wird an einen Kopf bzw. ein Gesicht angepasst (siehe Abbildung 5). Das System basiert auf einer Menge an Musterbildern verschiedenen Aussehens und unterschiedlichen Posen, bei denen die Knoten des Graphs manuell an die Gesichtsmerkmale (z. B. Mundwinkel links, Mundwinkel rechts, Nasenspitze etc.) angepasst werden müssen[Wis97]. Wird die Pose eines Kopfs abgefragt, werden alle gespeicherten Graphen mit dem aktuellen Abbild des Kopfs mithilfe eines Matching-Algorithmus für elastische Graphen verglichen. Der Graph mit der geringsten Abweichung der Knoten zu den aktuellen Gesichtsmerkmalen liefert die entspreche diskrete Pose zurück [Krue97]. Aufgrund dessen sollte eine große Menge an markierten Graphen verfügbar sein, um eine möglichst genaue Schätzung der Pose erreichen zu können. Der Rechenaufwand, der notwendig ist, um alle verfügbaren Graphen anzupassen, ist wiederum im Vergleich mit anderen Methoden zur Kopfposenschätzung sehr rechenaufwendig [Mur09].

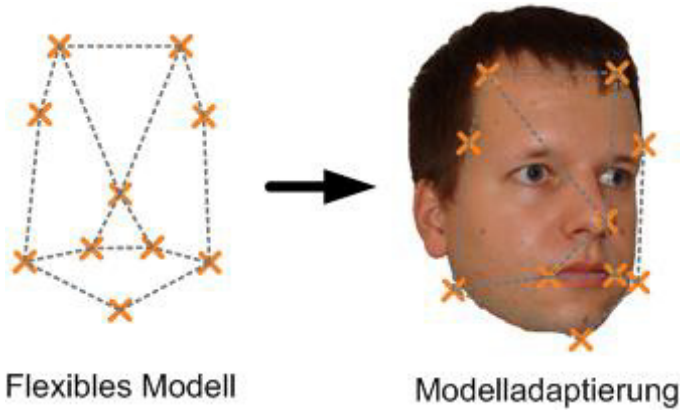

Abbildung 5 - Flexible Methoden: Anpassung des Modells an den Kopf

Geometrische Methoden

Geometrische Methoden zur Kopfposenschätzung benutzen die Kopfform bzw. die Position von Gesichtsmerkmalen zur Schätzung der Kopfpose. Werden, wie in Abbildung 6, Linien vom rechten Augenwinkel des rechten Auges zum linken Augenwinkel des linken Auges bzw. von der linken zur rechten Mundecke gezogen, kann eine Verbindung der Mittelpunkte der Linien gezogen werden. Dies ist die Symmetrieachse des Gesichts. Weicht nun beispielsweise die Nasenspitze von dieser Achse ab, kann von einer Drehbewegung ausgegangen werden [Gee94].

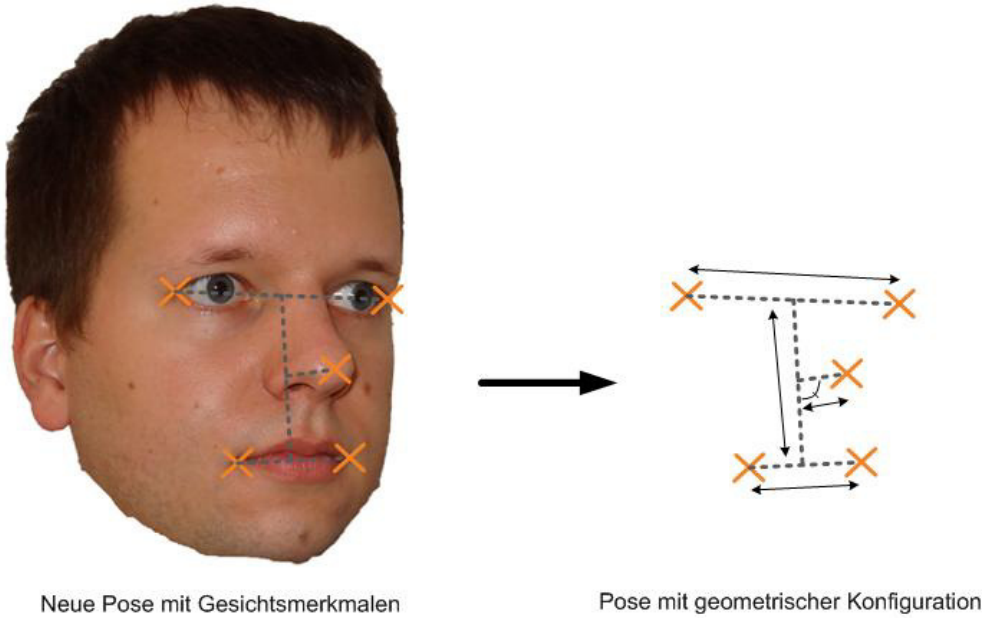

Abbildung 6 - Geometrische Methoden: Benutzen die Position von Gesichtsmerkmalen und die Kopfform zur Posenschätzung

Da nur einige Gesichtsmerkmale zum Schätzen der Pose notwendig sind, funktioniert diese Methode schnell und einfach. Eine Schwierigkeit dieser Methode liegt im korrekten Erkennen der Gesichtsmerkmale. Beispielsweise führt das Tragen von Schmuck bzw. Sehbehelfen (z. B. Brillen) zu Problemen bei der korrekten Lokalisation der Gesichtsmerkmale.

Trackingmethoden

Die Kopfposenschätzung auf Basis von Tracking basiert auf dem kontinuierlichen Verfolgen von Gesichtsmerkmalen innerhalb einer Videosequenz (siehe Abbildung 7). Es kann so die Summe relativer Bewegung zwischen einzelnen Bildern der Sequenz in eine Kopfposenschätzung überführt werden [Gee96].

Die Initialisierung des Erfassungssystems in einer frontalen Pose ist obligatorisch. Ebenso muss, im Falle des Verlusts des zu verfolgenden Kopfs, das Tracking neu initialisiert werden. Ohne diesen Initialisierungsvorgang ist das System nur in der Lage, relative Änderungen zwischen den einzelnen Bildern zu erfassen. Damit ist es nicht möglich, die Kopfpose in Relation zu einem initialen Ausgangszustand zu schätzen, sondern nur die relativen Bewegungsänderungen wiederzugeben.

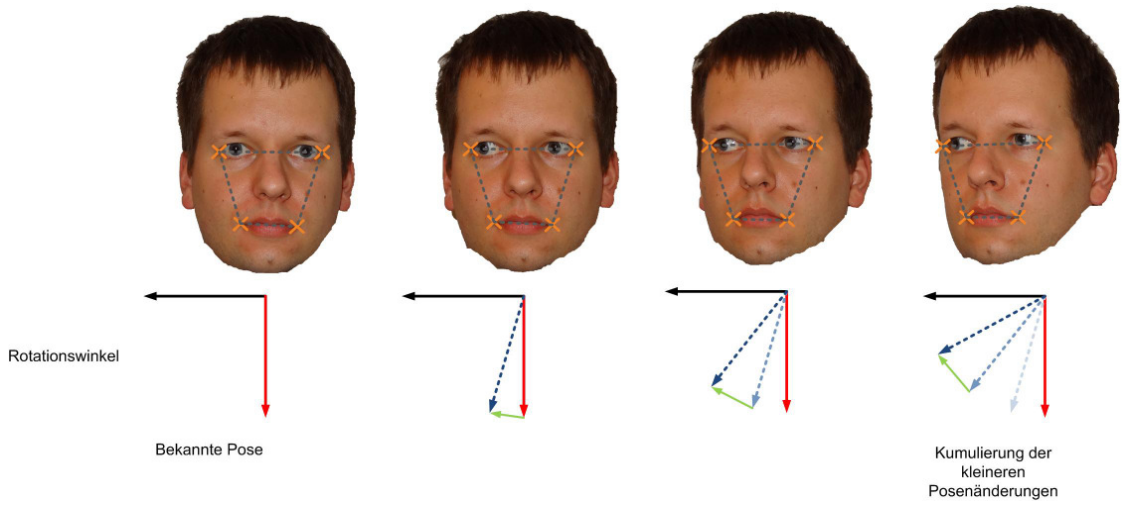

Abbildung 7 - Tracking Methode: Die Pose wird anhand der relativen Posenänderungen in einer Videosequenz ermittelt

Hybride Methoden

Da einzelne Methoden alleine nicht immer in der Lage sind, entsprechend genau bzw. effizient die Kopfposition bzw. Pose zu schätzen, werden in der Praxis häufig hybride Methoden angewandt, welche mehrere der oben erwähnten Methoden zu einem leistungsstarken Head Pose Esimation System kombinieren, um die Stärken der einzelnen Systeme zu kumulieren. Hierbei kann beispielweise eine Methode, welche besonders genau einen Kopf als solches erkennen kann, in der Initialisierungsphase eingesetzt werden, die laufende Kopfposenschätzung wird aber von einer geometrischen bzw. Tracking-Methode durchgeführt (siehe Abbildung 8).

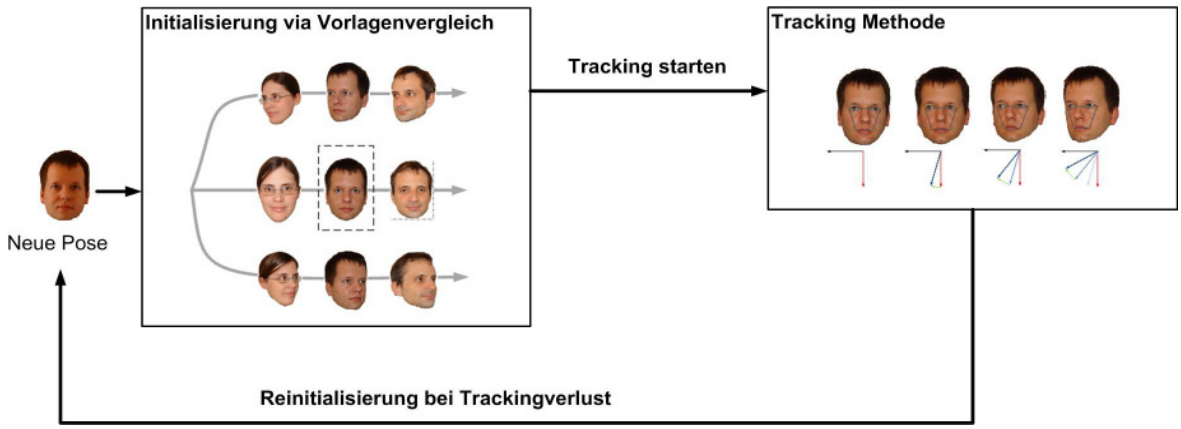

Abbildung 8 - Hybride Systeme kombinieren mehrere der verfügbaren Methoden

3.4.2 Facetracking Bibliotheken

Auf Basis dieser beschrieben Methoden für die Umsetzung eines Head Pose Estimation-Systems wurden schon unzählige Softwarepakete bzw. Programmierbibliotheken veröffentlicht. Im Folgenden werden nun verschiedene Anforderungen an Facetracking-Systeme beschrieben und ein Überblick über unterschiedliche in der Praxis einsetzbare bzw. am Markt befindliche Produkte zum Facetracking gegeben.

3.4.3 Anforderungen an Facetracking Bibliotheken

Folgende Anforderungen bzw. Designkriterien an Facetracking-Systeme (vgl. [Mur09]) sollten umgesetzt werden, um brauchbare Ergebnisse in der praktischen Anwendung von Facetracking erzielen zu können:

- **Akkurat:** Das System sollte in der Lage sein, verwendbare und korrekte Kopfposenschätzungen abgeben zu können.
- **Monokular:** Obwohl Systeme mit mehreren Kameras genauere Ergebnisse liefern könnten, sollte das System für die praktische Benutzbarkeit auch schon mit einer Kamera die Kopfposition korrekt abschätzen können.
- **Autonom:** Autonome Initialisierung ist obligatorisch für ein praktikables Facetrackingsystem. Manuelle Initialisierung, Erkennung bzw. Lokalisierung des Kopfes soll nicht Notwendig sein.
- **Multipersonen-Unterstützung:** Das System sollte in der Lage sein, bei mehreren Personen gleichzeitig die Kopfposition bestimmen zu können. Falls in bestimmten Fällen ein alleiniger Benutzer benötigt wird, soll der Fokus des Systems, bei Registrierung mehrerer Köpfe, an der aktuell steuernden Person bleiben.
- **Identitäts- und Beleuchtungsinvariant:** Unabhängig von der Identität der Person bzw. der Beleuchtungsverhältnisse sollte das System in der Lage sein, korrekt zu arbeiten.
- **Auflösungsunabhängig:** Das System sollte, auch aufgrund möglicher Mobilität des Systems, unabhängig von der Auflösung bzw. der Entfernung des Kopfs zur Kamera arbeiten.
- **Unterstützung aller möglichen Kopfbewegungen:** Alle sechs Freiheitsgrade (engl. „DOF“, degrees of freedom) - sollten vom System unterstützt werden (siehe Abbildung 9). Sowohl die Rotationsbewegungen eines Kopfes (Roll-, Nick- und Gier-Bewegungen) sowie das Bewegen des Kopfes im 3-dimensionalem Raum.
- **Echtzeit:** Das System sollte kontinuierliche Änderungen in Lage und Position des Kopfes möglichst schnell verarbeiten und die aktuellen Daten zur Verfügung stellen.

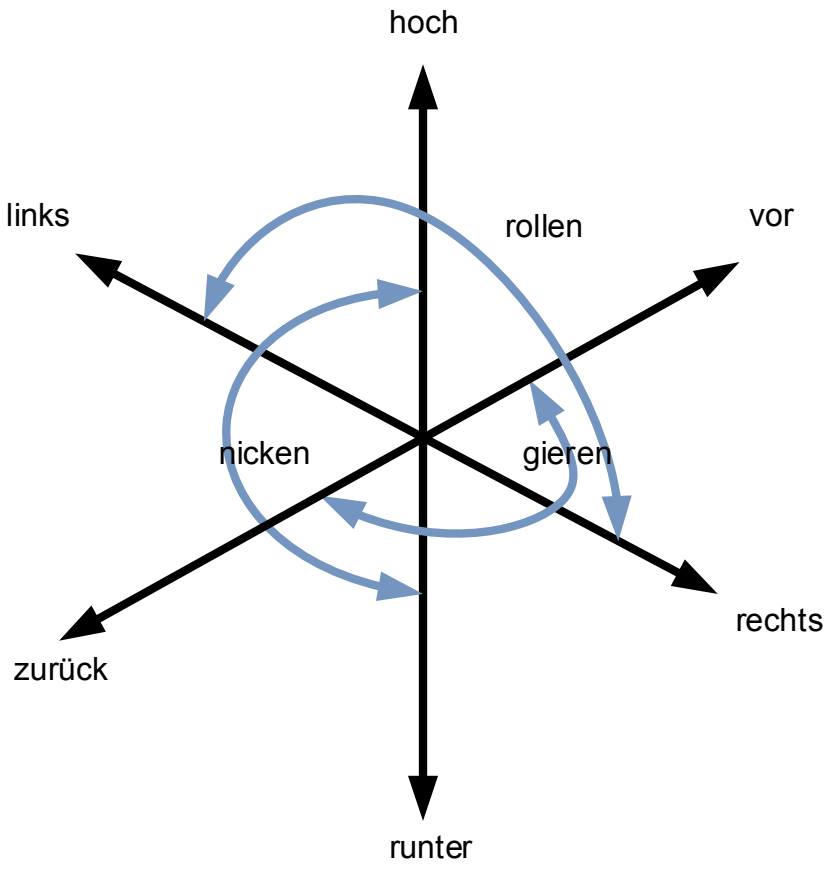

Abbildung 9 – Die sechs Freiheitsgrade (engl. "DOF" - degrees of freedom)

3.4.4 FaceAPI

FaceAPI (von Seeing Machines) ist eine kommerzielle Programmierschnittstelle (API), die Gesichtstracking mit Hilfe einer Standard-Webcam ermöglicht. FaceAPI unterstützt alle sechs Freiheitsgrade und kann somit sowohl alle Rotationsbewegungen als auch Bewegungen im 3-dimensionalen Raum erfassen und verarbeiten. FaceAPI ist außerdem in der Lage, Gesichtsmerkmale, beispielsweise die Ecken von Augen und Mund, Augenbrauen, und die Nasenspitze (siehe Abbildung 10), zu erkennen und zu verfolgen. Weiters ist es mit FaceAPI möglich, durch Lippen und Augenbrauen-Tracking die Gesichtsmimik zu erfassen.

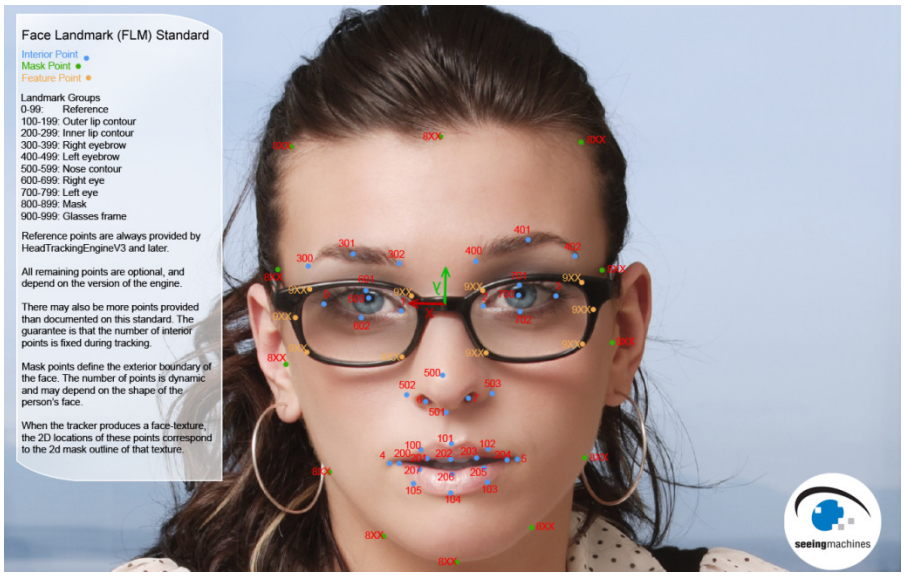

Abbildung 10 - Durch FaceAPI benutzter Face Landmark Standard (FLS) (aus [1])

FaceAPI Erfassungssystem - Koordinatensysteme

Alle Bewegungs- bzw. Merkmaldaten werden in Relation zu den jeweils betreffenden Koordinatensystemen erfasst. (Pixel- (2D), Welt- und Kamera- (3D), Kopf- (3D), Gesichtskontur- und Merkmalskoordinatensystem (2D)). Während beim Pixelkoordinatensystem sowie dem Gesichtskontur- und Merkmalskoordinatensystem nur zwei Achsen (x,y) abgebildet werden können, basiert das Welt- & Kamerakoordinatensystem sowie das Kopfkoordinatensystem auf einem dreidimensionalen Koordinatensystem (siehe Abbildung 11). Die einzelnen von Faceapi verwendeten Koordinatensysteme sind wie folgt aufgebaut:

- **Pixel Koordinatensystem**: Über das von der Webcam aufgenommene Bild wird ein 2d-Koordinatensystem gelegt, dessen Ursprung in der linken-oberen Ecke des Bilds liegt.
- **Kamera- & Welt-Koordinatensystem**: Zentrum dieses Koordinatensystems ist der Kameraprojektionspunkt. Ausgehend vom Zentrum dieses Koordinatensystems werden die Lage und Position des Kopfs ermittelt.
- **Kopf-Koordinatensystem**: Das Zentrum dieses 3-dimensionalen Koordinatensystems ist der Mittelpunkt der imaginären Linie zwischen den Augenwinkeln. Ausgehend von diesem Koordinatensystem werden die Gesichtsmerkmale lokalisiert.
- **Gesichtskontur- und Merkmalskoordinatensystem**: Das Zentrum des 2-dimensionalen Gesichtskontur- und Merkmalskoordinatensystems liegt im Ursprung des Kopfkoordinatensystems. Ausgehen vom Ursprung des Gesichtskontur- und Merkmalskoordinatensystems wird die Lage der Gesichtsmerkmale angegeben.

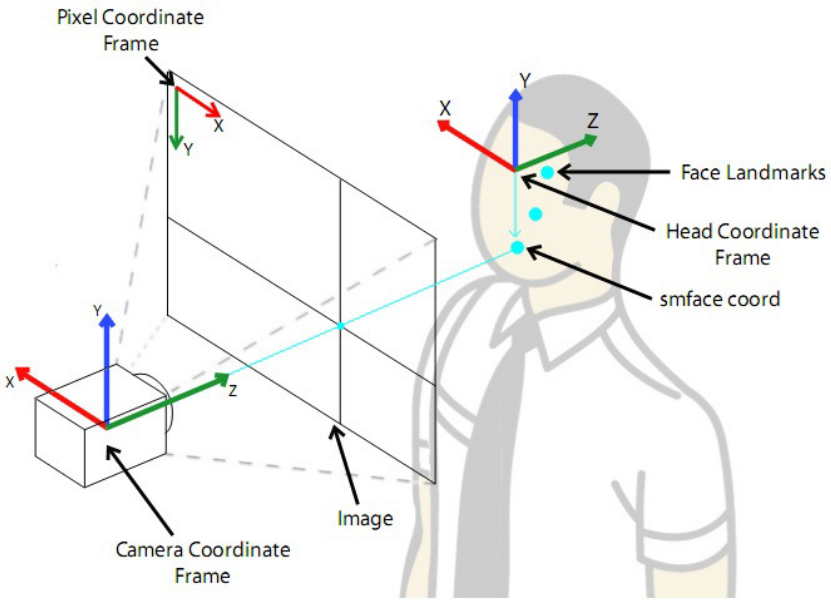

Abbildung 11 – Koordinatensystem in FaceAPI (aus [3])

Funktionalität und technische Details

FaceAPI ist in der Lage, Rotationsbewegungen und Positionsänderungen innerhalb definierter Grenzen zuverlässig zu erfassen. Die vorgegeben Grenzen bzw. Beschränkungen der Facetracking-Engine faceAPI für Bewegungen und Rotationen ermöglichen folgenden Bewegungsspielraum für das Erfassen von Rotationsgesten des Kopfes:

- Rotationen um die X-Achse: -30° - +60°
- Rotationen um die Y-Achse: -90° - +90°
- Rotationen um die Z-Achse: -90° - +90°
- Genauigkeit für Rotationsänderungen: <3°
- Genauigkeit für Positionsänderungen: <1 cm

In Abbildung 12 zeigt sich die technische Spezifikation von FaceAPI mit allen vorgegeben Beschränkungen und Informationen für die Anwendung der Trackingsystems von FaceAPI.

Tracking-Spezifikationen FaceAPI		
	Headtracker v1	**Headtracker v2**
Minimale Gesichtsgröße	24 Pixel	
Maximale Gesichtsverdeckung	50%	
Rotation x-Achse	-20°<X<45°	-30°<X<60°
Rotation y-Achse	-30°<Y<30°	-90°<Y<90°
Rotation z-Achse	-90°<Z<90°	
Fehlertoleranz Position	<1cm	
Fehlertoleranz Rotation	<3°	
Initialstatus- Spezifikationen		
Rotation x-Achse	<15°	
Rotation y-Achse	<15°	
Rotation z-Achse	<30°	
Dauer	0,3 – 3,0 Sekunden	
Suchmodus- Spezifikationen		
Minimalzeit f. Wiedererkennung	1 Frame	
Wiedererkennungs-Bedingungen	Frontalansicht Gesicht nicht verdeckt	Jede Pose Gesicht nicht verdeckt

Abbildung 12 – technische Spezifikation von FaceAPI [1]

3.4.5 OpenCV

OpenCV (Open Source Computer Vision) ist eine von Intel entwickelte Programmbibliothek, die über 500 Funktionen zur Echtzeit-Bildverarbeitung zur Verfügung stellt. Neben Methoden und Funktionen wie Bildverarbeitungsroutinen, Tranformationsmethoden und Objektsegmentierungsmethoden stellt OpenCV auch Gesichtserkennungs- und Trackingmethoden zur Verfügung. Diese in OpenCV verfügbaren Trackingmethoden erkennen Formen, beispielsweise ein Gesicht, und können dem erkannten Objekt im Raum folgen. Mit Hilfe der OpenCV-Funktion POSIT kann die Head Pose Estimation, also die Abschätzung der aktuell von einem Kopf eingenommenen Pose, durchgeführt werden.

4. Design und Implementierung

Die Implementierung der Aufzugssteuerung besteht, als Resultat der dem System vorausstehenden Anforderungen der Modularität und Flexibilität, aus drei größeren voneinander unabhängigen Modulen: Dem Facetracking-Modul für die Erfassung der Bewegungen, der Benutzerschnittstelle für die Interaktion mit dem Benutzer und dem Aufzugsimulator für die Visualisierung der Folgen der ausgewählten Befehle. Die Nutzung von FaceAPI als Facetracking-Engine ergab sich durch die gegebene Aufgabenstellung durch das Institut für Computertechnik der TU Wien.

In diesem Kapitel werden, nachdem auf die allgemeinen Anforderungen an das System eingegangen wird, die Systemarchitektur und die Implementierung der Benutzerschnittstelle sowie des Aufzugssimulators im Detail beschrieben.

4.1 Anforderungen

Aufgrund der im Rahmen dieser Diplomarbeit vorgestellten Ziele ergeben sich im Rahmen der Implementierung folgende Anforderungen:

- Einfache Benutzbarkeit: Das System soll unter der Annahme eines großen Anwendungsgebiets unter verschiedenen Rahmenbedingungen einfach zu benutzen sein und Anwendungsfehler möglichst vermeiden.
- Flexibilität: Die einzelnen Systemelemente (Facetracking-Engine, Benutzerschnittstelle, Befehlserkennung und –interpretation) sollen, je nach Anwendungsfall, auch ausgelagert werden können. Hierfür soll die Kommunikation zwischen den einzelnen Modulen netzwerkbasiert erfolgen.
- Modularität: Durch die Anforderung, einzelne Module durch andere Systeme ersetzen zu können, soll es möglich sein, sowohl den Einsatz anderer Facetracking-Module als auch eine Vielzahl von weiteren Anwendungsfällen zu unterstützen.
- Individualisierung: Den Benutzern soll es ermöglicht werden, dem System individuelle Befehle zu lernen, um so besser auf die speziellen körperlichen Einschränkungen der Benutzer Rücksicht nehmen zu können und die Erkennungsgenauigkeit im allgemeinen zu erhöhen.

4.2 Systemarchitektur

Die Systemarchitektur der Benutzerschnittstelle besteht, unter anderem aufgrund der abgezielten Modularität, aus mehreren voneinander unabhängigen Systemen (siehe Abbildung 13):

- Der Laptop inkl. Webcam zur Erfassung von Bewegungsdaten und zur Anzeige der Benutzerschnittstelle
- Steuerungssoftware mit Datenaufbereitung, Gestenerkennung und Befehlsgenerierung
- Aufzug zum Empfang und zur Ausführung der Befehle

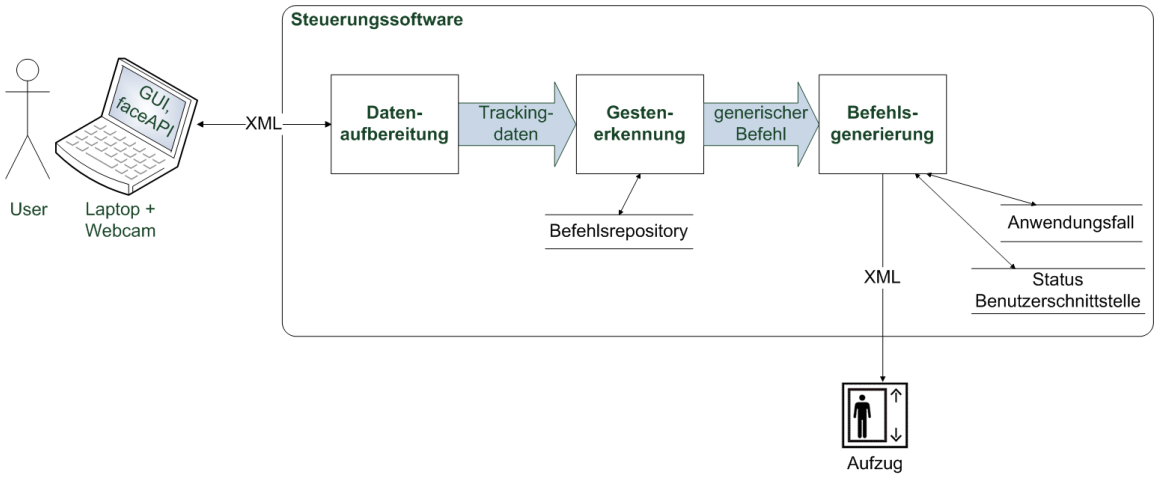

Abbildung 13 - Systemaufbau

Um eine möglichst hohe Modularität und Flexibilität zu gewährleisten, kommunizieren die verschiedenen Module mit XML-Nachrichten über TCP-Sockets. So soll es unter anderem möglich sein, einzelne Module des Systems auszutauschen, um so auch andere Anwendungsfälle oder Facetracking-Systeme zu unterstützen. Diese Architektur ermöglicht es, die einzelnen Module auch in räumlich getrennten Systemen zu betreiben, sodass beispielweise die Kamera mit dem Facetracking-Modul an einem Rollstuhl befestigt wird, der Bildschirm zur Darstellung der Benutzerschnittstelle aber fix im Aufzug verbaut ist.

Es werden zwei unterschiedliche Arten von XML-Nachrichten zur Kommunikation zwischen den Systemmodulen verwendet:

- **Facetracking Daten:** Die im Facetracking-Modul erzeugten Daten auf Basis der aktuell eingenommenen Pose des Benutzers.
- **Konkreter Befehl:** Der konkrete, verarbeitbare Befehl zur Steuerung beispielsweise eines Aufzugs. Dieser Befehl wird vom Aufzug verstanden und kann von ihm ausgeführt werden.

Die einzelnen Elemente des Systemaufbaus sowie die konkreten Inhalte der jeweiligen Datenströme werden in diesem Kapitel noch ausführlich beschrieben.

4.3 Facetracking-Modul

Im Facetracking-Modul (siehe Abbildung 14) werden mit der Facetracking-Engine FaceAPI von Seeing Machines über eine Webcam aufgezeichnete Lage- und Positionsdaten erfasst und zur weiteren Verarbeitung zur Verfügung gestellt.

Abbildung 14 - Facetracking-Modul

Aus der Engine resultiert ein Datenstrom aus über die Zeit aufgezeichneten Datensätzen, bestehend aus Posen- und Lagedaten des über die Webcam aufgenommenen Bilds. Dieser Datenstrom wird an die Benutzerschnittstelle weitergeleitet und über die Befehlserkennung und -verarbeitung in einen konkreten Befehl umgewandelt. Das Facetracking-Modul arbeitet unabhängig von anderen Systemteilen und kann jederzeit, unter Anpassung unter Verwendung der gleichen Kommunikationsschnittstelle, von einer anderen Facetracking-Engine ersetzt werden.

Die Lage- und Positionsdaten, die von faceAPI mit 30Hz erfasst werden, werden über ein TCP/IP-Socket an die Benutzerschnittstelle übermittelt. Dadurch ist es möglich, das Facetracking-Modul mit der Webcam sowohl am selben Computer wie die Benutzerschnittstelle als auch, beispielsweise als fix-verbautes Bauteil eines Gerätes (z. B. Aufzug), zu betreiben. Im Falle eines fix-verbauten Moduls kann die Benutzerschnittstelle via WLAN auf die Daten der Facetracking-Engine zugreifen.

In den Positionsdaten des Datenstroms wird der Abstand des Kopfes im Raum zum virtuellen Nullpunkt in Bezug zur Webcam auf der x, z und z-Achse von der Facetracking-Engine erfasst und übermittelt {posx, posy, posz}. Die Lagedaten beinhalten die aktuellen Rotationswinkel in x, y und z-Achse des Kopfes in Radiant {rotx, roty, rotz}. Die Rotationswinkel werden vor der Weiterverarbeitung von Radiant in Grad konvertiert. Weiters wird eine geschätzte Erkennungswahrscheinlichkeit der Engine erfasst und weitergegeben (0 … 1 = 0% … 100%).

In Listing 1 ist die Struktur einer xml-Nachricht aus der Facetracking-Engine ersichtlich. In Listing 2 ist der Inhalt einer generierten und verschickten Nachricht von faceAPI an die Benutzerschnittstelle dargestellt.

```
<facedata type="faceAPI">
     <rotx>RotX-Value</rotx>
     <roty>RotY-Value</roty>
     <rotz>RotZ-Value</rotz>
     <posex>PoseX-Value</posex>
     <posey>PoseY-Value</posey>
     <posez>PoseZ-Value</posez>
     <confidence>Condifence-Value</confidence>
</facedata>
```

Listing 1 - Struktur einer XML-Nachricht aus dem Facetracking-Modul faceAPI

```
<facedata type="faceAPI">
        <rotx>-15,59537</rotx>
        <roty>-1,820465</roty>
        <rotz>0,400794</rotz>
        <posex>-0,008274991</posex>
        <posey>-0,006181477</posey>
        <posez>0,4115051</posez>
        <confidence>0,6095423</confidence>
</facedata>
```

Listing 2 – Beispiel einer Nachricht aus faceAPI

In der Methode *receiveHeadPose* (siehe Listing 3) werden die Daten der Facetracking-Engine ausgelesen (Pointer **smEngineHeadPoseData*). Anschließend werden die Daten von Radiant in Grad konvertiert, eine XML-Nachricht generiert und diese über den generierten TCP/IP-Port zur Benutzerschnittstelle gesendet.

Wird das Facetracking-Modul und die Benutzerschnittstelle nicht am selben Computer ausgeführt , müssen die IP-Adresse und der Port an die des Ziel-Computers angepasst werden.

```
void STDCALL receiveHeadPose(void *,smEngineHeadPoseData head_pose,
smCameraVideoFrame video_frame){
	[…]

	//Convert headrotation-data from radiant in degrees
	head_rot_x=head_pose.head_rot.x_rads;
	head_rot_y=head_pose.head_rot.y_rads;
	head_rot_z=head_pose.head_rot.z_rads;

	try {
		//Definition of port for TCP/IP-communication with client (e.g.
		elevator user interface)
		Int32 port = 815;
		String^ ipaddress = "127.0.0.1";
		String^ message;
		TcpClient^ client = gcnew TcpClient(ipaddress, port);

		// Create xml-message
		message = "<facedata type=\"faceAPI\"><rotx>" +
		rad2deg(head_pose.head_rot.x_rads) + "</rotx><roty>"+
		rad2deg(head_pose.head_rot.y_rads) +
				"</roty><rotz>" +
		rad2deg(head_pose.head_rot.z_rads) +"</rotz><posex>"+
		head_pose.head_pos.x + "</posex><posey>"
				+ head_pose.head_pos.y + "</posey><posez>"+
		head_pose.head_pos.z +"</posez><confidence>"+ head_pose.confidence
		+"</confidence></facedata>";

		array<Byte>^ data = Text::Encoding::ASCII->GetBytes(message);

		// Create network-stream and write message
		NetworkStream^ stream = client->GetStream();
		stream->Write(data, 0, data->Length);
		client->Close();

	} catch (SocketException ^e)
	{
		Console::WriteLine("SocketException {0}", e);
	}
}
```

Listing 3 - Auslesen der Facetrackingdaten; Erzeugen und Übermittleln der XML-Nachricht

4.4 Benutzerschnittstelle

Mit Hilfe der Benutzerschnittstelle erfolgt die Eingabe der Befehle mit Hilfe von Kopfgesten, die über die Facetracking-Engine erfasst werden. Ein Laptop und eine Webcam dienen hierbei zur Erfassung der Bewegungsdaten und zur Anzeige der Benutzerschnittstelle (siehe Abbildung 15). Die zentralen Steuerelemente sind die vier Befehle im rechten Bereich der Benutzerschnittstelle.

Mit der korrespondierenden Befehlsgeste können die vier Steuerelemente angesprochen werden. So führt beispielweise das Ausführen der Geste von Befehl 1 zum Erhöhen des Stockwerkszählers um einen Stock und der Befehl zum Bestätigen des Befehls (Befehl 3) zur Auführung des gewünschten Befehls.

Die Benutzerschnittstelle beinhaltet weiters Funktionen zur Benutzerverwaltung. Über die Benutzerverwaltung ist es möglich, dem System indivuelle Befehle zu lernen oder bestehende Benutzer aus dem System zu entfernen.

Weiters wurde eine Informationsleiste implementiert, in der wichtige Hinweise und Systemrückmeldungen angezeigt werden. Um den Benutzer über den aktuellen Status der Benutzung der Benutzerschnittstelle zu informieren, ihn aber nicht von der Verwendung abzulenken, werden diese Meldungen farbig im unteren Bereich der Benutzerschnittstelle angezeigt, müssen aber nicht vom Benutzer explizit bestätigt werden. Wartet beispielsweise die Benutzerschnittstelle auf einen Befehl, wird das durch die entsprechende Meldung den Benutzer angezeigt. Folgende Meldungen werden je nach aktuellem Status angezeigt:

- *Warte*: Wurde die Verbindung zur Facetracking-Engine verloren, wird diese Statusmeldung angezeigt.
- *Bewegen Sie sich zurück zur Ausgangsposition!*: Die Befehlserkennung wurde abgeschlossen. Der Benutzer soll den Kopf wieder Richtung Ausgangposition bewegen.
- *Lernmodus aktiv*: Diese Meldung wird angezeigt, falls dem System neue Gesten gelernt werden.
- *Warte auf einen Befehl*: Die Befehlserkennung ist aktiv und wartet auf die nächste Befehlsgeste.
- *Führe Befehl aus! Bitte warten Sie kurz!:* Eine Befehlsgeste wurde erkannt und wird an den Empfänger gesendet.
- *Bitte einen Benutzer auswählen!*: Es ist noch kein Benutzer ausgewählt.
- *Befehlserkennung*: Wird eine mehrteilige Geste ausgeführt, wird diese Meldung angezeigt.

Für das Betätigen des Notstopps und des Notrufs muss in das Menü für Spezialbefehle gewechselt werden. Um in dieses Menü zu gelangen, muss die Geste des vierten Befehls ausgeführt werden. Die Steuerelemente der Benutzerschnittstelle werden angepasst. Statt den Befehlen zum Erhöhen und Vermindern des Stockwerkzählers stehen nun die Befehle „Notstop“ und „Notruf“ zur Auswahl

(siehe Abbildung 16). Anschließend kann, wie gewohnt, die gewünschte Funktion ausgewählt und die Auswahl mit *Befehl 3* bestätigt werden.

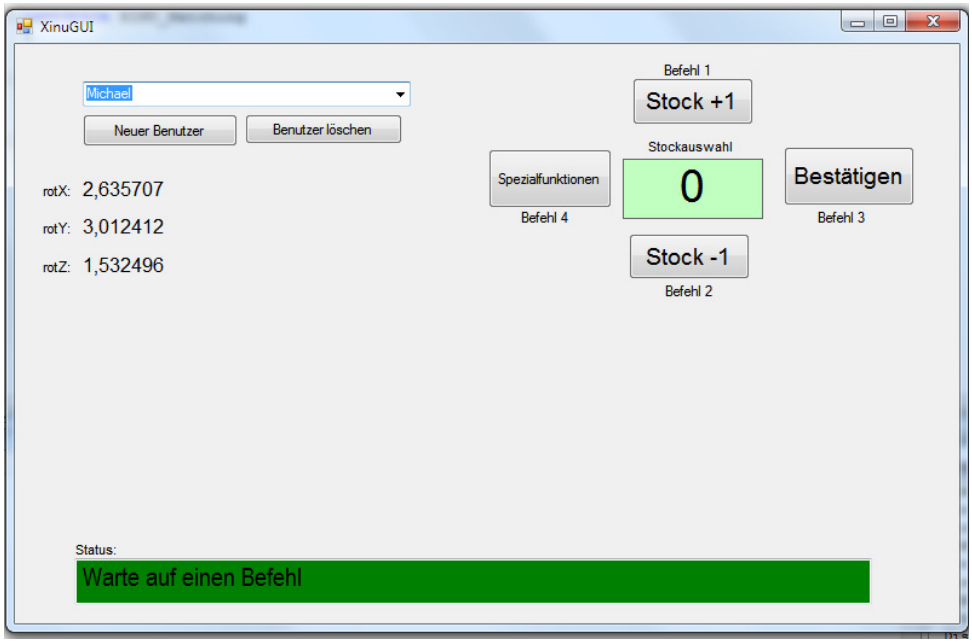

Abbildung 15 - Zentrale Benutzerschnittstelle

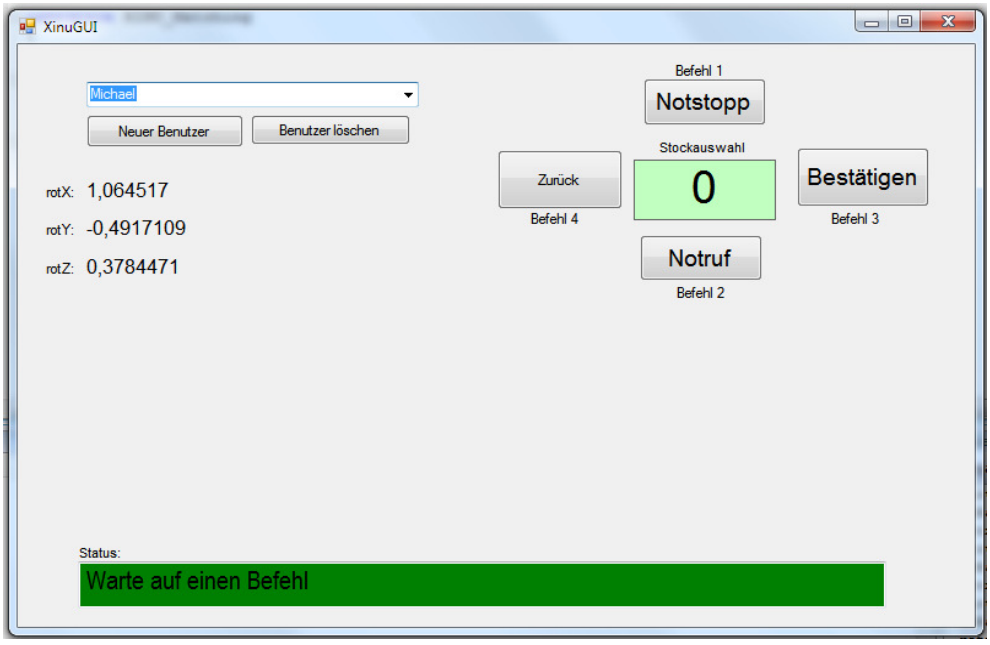

Abbildung 16 - Benutzerschnittstelle mit Befehlen für den Notfall (Modus für „Spezialfunktionen)

Werden vom Benutzer zwischen den durchgeführten Gesten Kopfbewegungen durchgeführt, wird der Benutzer darauf hingewiesen, wieder in die Ausgangsposition zurückzukehren. Dies ist dann der Fall, wenn die Ruheposition mehr als fünf Grad vom Nullpunkt abweicht, aber keine Geste durchgeführt wird. Dies wird dem Benutzer durch die Darstellung des Abbilds eines Gesichts, dass gerade in die Kamera blickt, und einer entsprechenden Meldung verdeutlicht („Bitte kehren Sie wieder in Ihre Ausgangsposition zurück!“).

Abbildung 17 - Benutzer ist etwas außerhalb des besten Erfassungsbereichs

Beim Programmstart wird eine Benutzerabfrage durchgeführt. Anhand des ausgewählten Benutzers werden die jeweils abgespeicherten Gesten je Befehl ins System geladen (siehe Listing 4). Die Befehle selbst sind im XML-Format als Textdatei im Dateisystem abgelegt. In Kapitel 4.4.3 „Lernen von Gesten“ (Seite 33) wird das Lernen der Befehle und deren Abspeicherung im Detail beschrieben.

```
public int setVectors(LinkedList<MovementVector> vectors1, LinkedList<MovementVector>
vectors2, LinkedList<MovementVector> vectors3, LinkedList<MovementVector> vectors4){
        vl1 = vectors1;
        vl2 = vectors2;
        vl3 = vectors3;
        vl4 = vectors4;

        Console.WriteLine("Vector 1 (" + vectors1.Count + ") setted. length " +
        vl1.Count);
```

```
        Console.WriteLine("Vector 2 (" + vectors2.Count + ") setted. length " +
        vl2.Count);
        Console.WriteLine("Vector 3 (" + vectors3.Count + ") setted. length " +
        vl3.Count);
        Console.WriteLine("Vector 4 (" + vectors4.Count + ") setted. length " +
        vl4.Count);

        return 1;
}
```

Listing 4 - Setzen der aktuell gültigen Befehlsgesten im Modul zur Befehlserkennung

Die von der Facetracking-Engine FaceAPI erzeugten Rohdaten werden erfasst und aufbereitet. Hierfür werden relevanten Daten auf den empfangenen Daten der Facetracking-Engine extrahiert (Posen- und Lagedaten, Erkennungswahrscheinlichkeit) und um einen Timestamp angereichert. Diese Daten werden anschließend zur besseren Verarbeitung gegenüber einem virtuellen Nullpunkt normalisiert (siehe Listing 6). Die aktuelle Pose des Kopfs in Relation zur Webcam im Ruhezustand beträgt meistens nicht exakt 0° auf allen drei Rotationsachsen. Zur einfacheren Verarbeitung der Daten wird die jeweilige Abweichung der aktuellen Pose von allen neu erfassten Posen abgezogen, sodass rechnerisch mit auf 0° normalisierten Daten gearbeitet werden kann. Für das Normalisieren der Daten wird in regelmäßigen Abständen (lt. aktueller Parametereinstellung, default: zwei Sekunden) die Methode *setIdlePose* ausgeführt, mit der die aktuelle Position im Ruhezustand als der aktuelle Initialzustand gesetzt (siehe Listing 5). Wurde länger als zwei Sekunden keine Geste ausgeführt, wird angenommen, dass sich der Kopf des Benutzers noch in der Ruheposition befindet und die aktuell eingenommene Pose als Initialzustand gesetzt. Die Facetracking-Daten werden mit der Methode *normalizeFacedata* anhand dieses Initialzustandes normalisiert (siehe Listing 6) und werden mit der Methode *workOnFacedata* an die Befehlserkennung und -verarbeitung weitergeleitet (siehe Listing 7).

```
public static void setIdlePose(Facedata myIdlePose)
{
        Console.Write("New Idlepose setted: ");
        Facedata.printFacedata(myIdlePose);

        idlePose = myIdlePose;
}
```

Listing 5 - Setzen eines neuen Initialzustands

```
public static Facedata normalizeFacedata(Facedata idleFacedata, Facedata
myFacedata)
{
	Facedata normalizedFacedata = new Facedata();

	normalizedFacedata.rotx = myFacedata.rotx – idleFacedata.rotx;
	normalizedFacedata.roty = myFacedata.roty – idleFacedata.roty;
	normalizedFacedata.rotz = myFacedata.rotz – idleFacedata.rotz;
	normalizedFacedata.posex = myFacedata.posex – idleFacedata.posex;
	normalizedFacedata.posey = myFacedata.posey – idleFacedata.posey;
	normalizedFacedata.posez = myFacedata.posez – idleFacedata.posez;
	normalizedFacedata.confidence = myFacedata.confidence;
	normalizedFacedata.timestamp = myFacedata.timestamp;

	return normalizedFacedata;
}
```

Listing 6 - Normalisieren der Trackingdaten

Die aufbereiteten Facetracking-Daten werden anschließend via XML-Datenstrom an die Befehlserkennung weitergeleitet.

```
workOnFacedata(Facedata.normalizeFacedata(idlePose, myFacedata));
```

Listing 7 - Weiterleiten der normierten Trackingdaten an die Befehlserkennung

4.4.1 Interaktion mit der Benutzerschnittstelle

Die Benutzerschnittstelle kann mit vier Befehlsgesten gesteuert werden. Jede dieser vier Befehlsgesten bedient eine der vier Bedienelemente, mit der die Benutzerschnittstelle bedient werden kann. Eine Anforderung an das System war, die Befehle so zu gestalten, dass diese für die Benutzer leicht zu merken sind. Da es möglich sein soll, dem System auch selber Befehle zu lernen, hat sich die Anzahl von vier Befehlen als am praktikablsten erwiesen. Mit weniger Befehlen ist es schwierig, Die Benutzerschnittstelle effiezient bedienen zu können. Bei mehr Befehlen ist die Wahrschnlichkeit groß, dass der Benutzer die verfügbaren Befehlsgesten vertauscht oder einzelne Gesten vergisst.

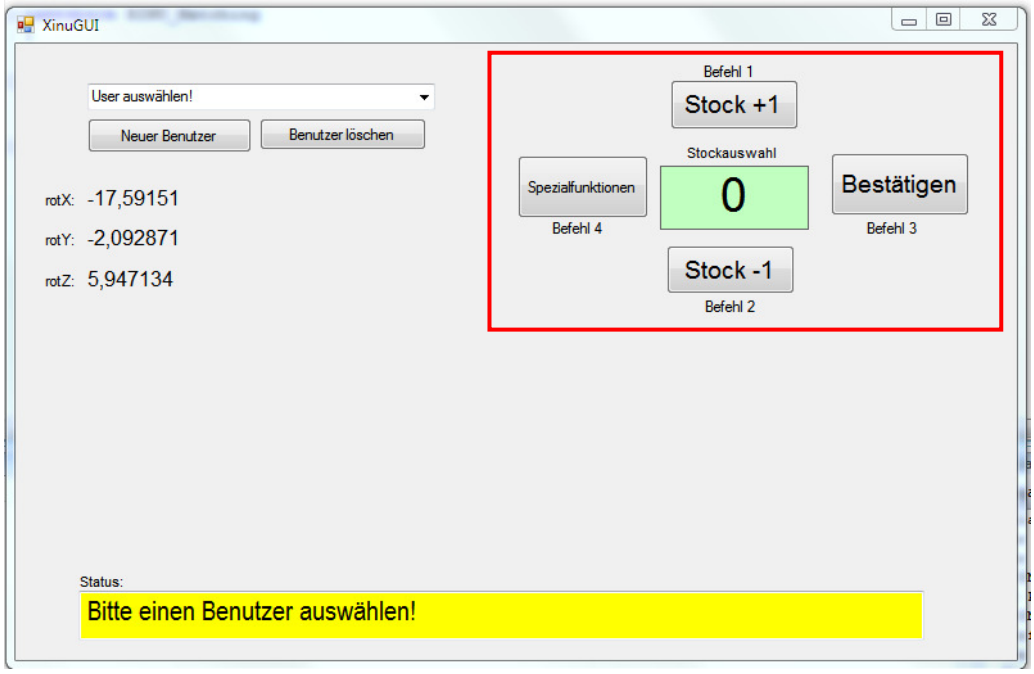

Abbildung 18 – Interaktion mit der Benutzerschnittstelle (rote Markierung)

4.4.2 Standardbefehlssatz

Zur Interaktion mit der Benutzerschnittstelle wurde ein Standardbenutzer im System hinterlegt. Diesem Standardbenutzer wurden vier Befehlsgesten, eine Geste je Befehl, zugewiesen, mit denen die Benutzerschnittstelle bedient werden kann. Im Rahmen des Anwendungsfalls (Aufzug) wurde versucht, diese Befehle möglichst intuitiv zu wählen. So wurde beispielweise der Befehl für das Erhöhen des Stockwerkzählers durch ein Nicken des Kopfes nach oben definiert. Zum Vermindern des Stockwerkszählers wurde ein Nicken nach unten definiert. Diese vier Befehle bestehen aus folgenden (Teil-)Gesten (siehe auch Abbildung 19):

- **Befehl 1:** Nicken um 20° nach oben
    - Teilgeste 1: (rotx, royx, rotz) = (10.00, 0.00, 0.00)
    - Teilgeste 2: (rotx, royx, rotz) = (10.00, 0.00, 0.00)
- **Befehl 2:** Nicken um 20° nach unten.
    - Teilgeste 1: (rotx, royx, rotz) = (-10.00, 0.00, 0.00)
    - Teilgeste 2: (rotx, royx, rotz) = (-10.00, 0.00, 0.00)
- **Befehl 3:** Kopf um 20° nach links drehen (gieren).
    - Teilgeste 1: (rotx, royx, rotz) = (0.00, 10.00, 0.00)
    - Teilgeste 2: (rotx, royx, rotz) = (0.00, 10.00, 0.00)
- **Befehl 4:** Kopf um 20° nach rechts drehen (gieren).
    - Teilgeste 1: (rotx, royx, rotz) = (0.00, -10.00, 0.00)
    - Teilgeste 2: (rotx, royx, rotz) = (0.00, -10.00, 0.00)

Abbildung 19 - Vier Befehlsgesten

Die im System gespeicherten, aber auch die neu gelernten Gesten, werden in Vektorform weiterverarbeitet. Die Befehlsgesten können zur besseren Visualisierung nun in einem dreidimensionalen Koordinatensystem aufgetragen werden. Jede Achse spiegelt eine Rotationsachse wieder.

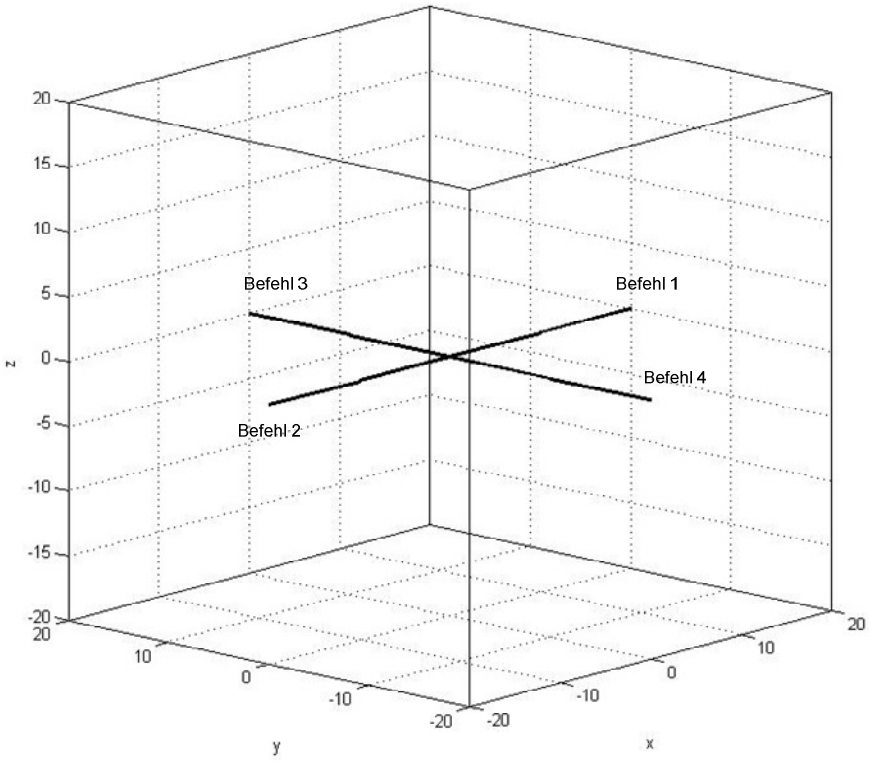

Abbildung 20 - Visualisierung der Standardbefehlsgesten

4.4.3 Lernen von Gesten

Für die bessere Benutzbarkeit ist es möglich, die Benutzerschnittstelle mit individuell definierten Befehlsgesten bedienen zu können. Individuelle Befehle sind für Personen mit körperlicher Einschränkung besonders hilfreich, die die Befehle des Standardbefehlssatzes aufgrund ihrer Einschränkung nicht ausführen können. Aber auch bei Personen ohne körperliche Einschränkung kann das Anlegen eines eigenen Benutzers dabei helfen, die Erkennungsgenauigkeit beim Ausführen der Befehle zu erhöhen.

Das Lernen von Befehlen wird während der Benutzeranlage durchgeführt. Zum Anlegen eines neuen Benutzers muss in der Benutzerschnittstelle die Funktion „Neuer Benutzer“ gestartet werden (siehe Abbildung 21).

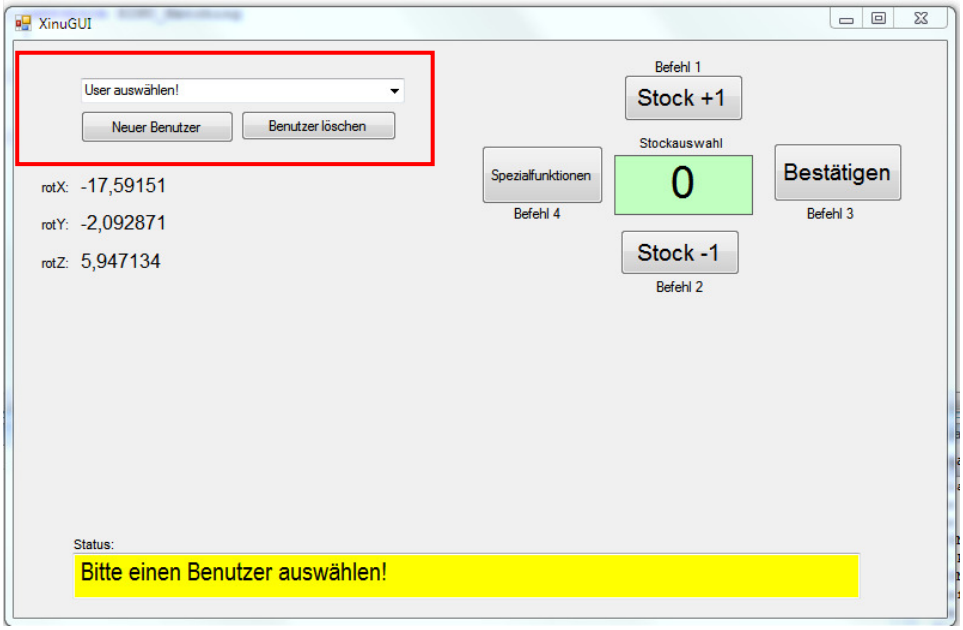

Abbildung 21 - Benutzerverwaltung (rote Markierung)

Nach Eingabe und Bestätigen des Benutzernamens wird der Benutzer aufgefordert, nacheinander vier Gesten, je eine Geste für einen Befehl, durchzuführen und so dem System zu lernen. Wurde ein Befehl komplett erfasst, wird die Erfassung der Geste mit der Meldung „OK!“ bestätigt (siehe Abbildung 22).

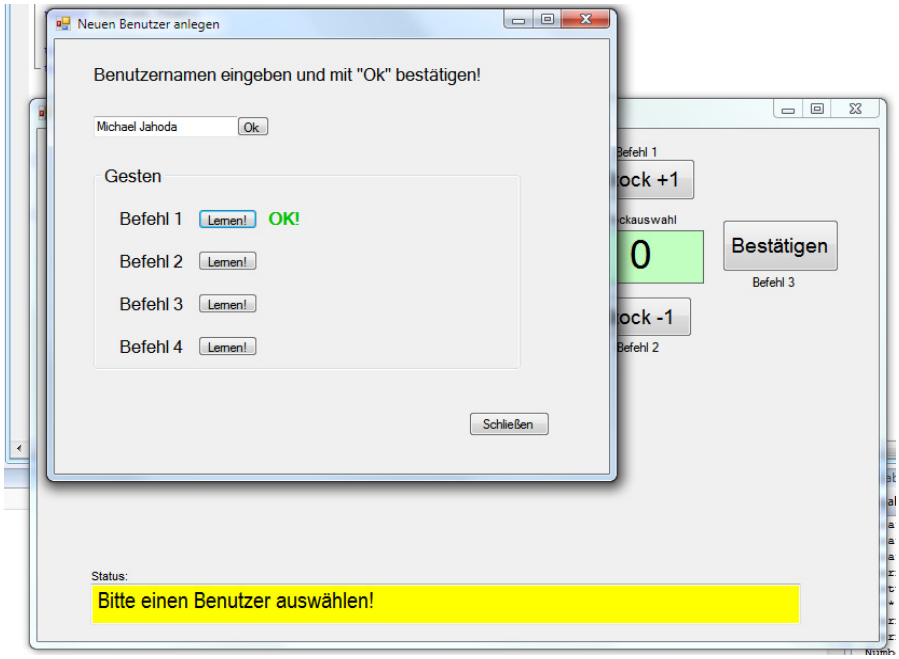

Abbildung 22 - Benutzeranlage mit Lernvorgang

Die durchgeführten Gesten werden in einer benutzerspezifischen Datei zusammengefasst und im System gespeichert.

4.4.4 Abspeicherung von gelernten Gesten

Wurden dem System Gesten gelernt, werden diese im Dateisystem als benutzerspezifische Textdatei abgelegt. Die einzelnen Gesten werden als XML-Datensatz abgespeichert. Wurden alle Befehlsgesten vom Lernmodul erfasst, wird die fertige Befehlsdatei erzeugt und im Arbeitsverzeichnis abgelegt.

In Listing 8 wird eine gespeicherte Geste für ein Drehen des Kopfs um 10° nach rechts um die y-Achse dargestellt.

Beim Lernen einer Geste wird die Bewegung des Kopfes in zehn-Grad Schritten erfasst. Die Gestenerkennung startet, sobald sich die Pose des Kopfs mehr als zehn Grad von der Initialposition (x,y,z = (0,0,0) abweicht. Die Befehlserkennung zum Lernen einer Geste endet, sobald die aktuelle Pose nach Durchführen einer Geste wieder weniger als zehn Grad der Initialpose abweicht. Eine gelernte Geste besteht somit aus mindestens einer oder beliebig vielen Teil-Gesten. Sobald eine Geste komplett erkannt wurde, wird sie im XML-Format in die Befehlsdatei des angelegten Benutzers geschrieben. Wurden im Lernmodul alle vier Befehle durchgeführt, wird die Befehlsdatei im System abgespeichert.

```
[...]
2:
<movement>
      <rotx>0</rotx>
      <roty>-10</roty>
      <rotz>0</rotz>
      <posex>0</posex>
      <posey>0</posey>
      <posez>0</posez>
      <duration>5</duration>
</movement>;
[...]
```

Listing 8 - Gespeicherte Geste im XML-Format

4.5 Ablaufsteuerung

Wurden die Bewegungsdaten von der Benutzerschnittstelle empfangen und normalisiert, werden sie an die Ablaufsteuerung weitergeleitet. Diese Ablaufsteuerung wurde mit Hilfe einer internen Statemachine realisiert. Der aktuelle State dieser Statemachine entscheidet, in welches Programmmodul die Daten weitergeleitet werden. Die Statemachine besteht aus acht verschieden States, in die je nach Benutzerauswahl in der Benutzerschnittstelle oder aktueller Pose gewechselt wird. Die einzelnen States und deren Wechselbedingungen werden im Folgenden beschrieben. Die States und

deren Übergangsbedingungen für die Befehlserkennung und –verarbeitung werden in Abbildung 23 dargestellt. In Abbildung 24 werden die States und Übergangsbedingungen für das Lernen von Gesten abgebildet.

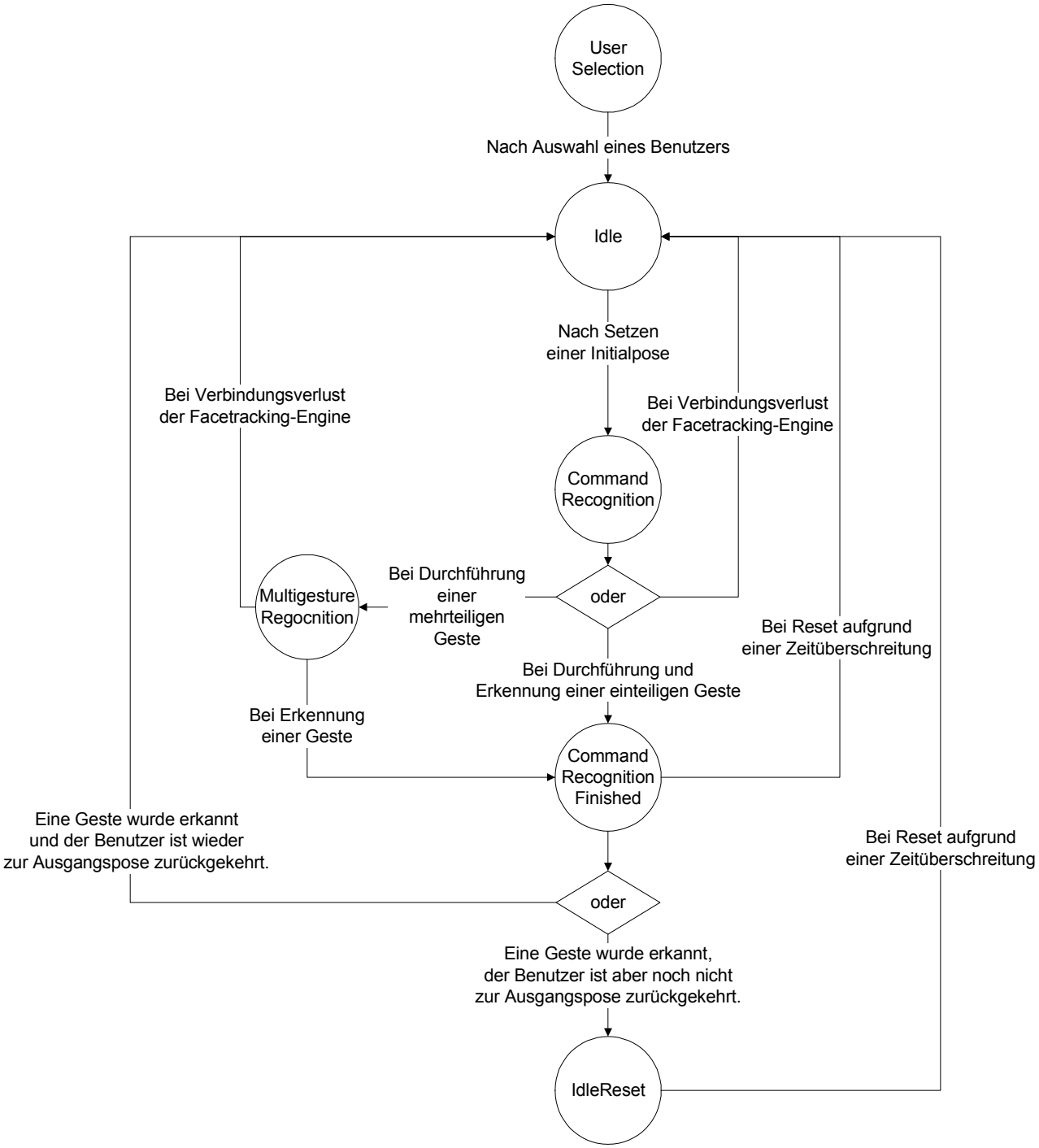

Abbildung 23 – Statediagramm für das Erkennen und Verarbeiten von Befehlen

Im System wird im Detail mit folgenden States gearbeitet:

- **UserSelection**
  Dieser State ist zu Programmstart aktiv. Es wird auf eine Auswahl des gewünschten Benutzers durch den Anwender gewartet. Die Befehlserkennung ist in diesem State noch nicht aktiv, obwohl die Bewegungsdaten laufend empfangen werden.

- **Idle**
  In den State „Idle“ wird gewechselt, falls die Verbindung zur Facetracking-Engine unterbrochen ist. Wurde die Verbindung wieder hergestellt, wechselt die Statemachine zurück in den Status „CommandRecognition“.

- **IdleReset**
  Wird die im System hinterlegte Resetzeit überschritten (z. B. zwei Sekunden), wird in diesen State gewechselt. Es wird angenommen, dass mit der aktuell eingenommenen Pose kein Befehl ausgeführt werden soll. In diesem State wird die Befehlserkennung neu gestartet und die Pose für den Nullpunkt, also die Parameter für das Normalisieren der empfangenen Bewegungsdaten, neu auf die aktuelle Position gesetzt.

- **CommandRecognition**
  In diesem State wird auf eine Geste gewartet, die als Befehl interpretiert werden könnte. Sobald einer der Rotationsachsen über 7° vom Nullpunkt abweicht, wird die Befehlserkennung gewartet. Folgen auf eine Bewegung weitere Bewegungen, wird in den State „Multigesture-Regocnition“ gewechselt. Wurde schon nach nur einer durchgeführten Geste ein gelernter Befehl erkannt, wird in den State „CommandRegcignitionFinished“ gewechselt.

- **MultigestureRecognition**
  Besteht eine Befehlsgeste aus mehreren Teilbewegungen, wird in diesen State gewechselt. In diesem werden alle durchgeführten Bewegungen erfasst und mit dem vorgegebenen gelernten Gesten verglichen. Wird ein Befehl erkannt, wird die Erkennung abgeschlossen und in den State „CommandRecognitionFinished“ gewechselt.

- **CommandRecognitionFinished**
  Sobald ein Befehl erkannt wurde, wird in diesen State gewechselt. Der zur Geste passende Befehl (z. B. Befehl „1“) wird generiert und von der Benutzerschnittstelle ausgeführt (z. B. „Erhöhe den Zähler um 1“). Wird der Befehl zum Bestätigen der aktuellen Auswahl („Befehl „3“) von der Benutzerstelle ausgeführt, wird die Auswahl (z.B. Feld Stockwerkzähler: „2“) zusätzlich an den Befehlsinterpreten weitergeleitet. Dieser wandelt den allgemeinen Befehl (z. B. „2“) auf Basis des aktuellen Anwendungsfalls („Aufzug“) in einen maschinenverständlichen Befehl um und sendet diesen über die Kommunikationsschnittstelle zum Aufzug, von dem der Befehl anschließend ausgeführt wird.

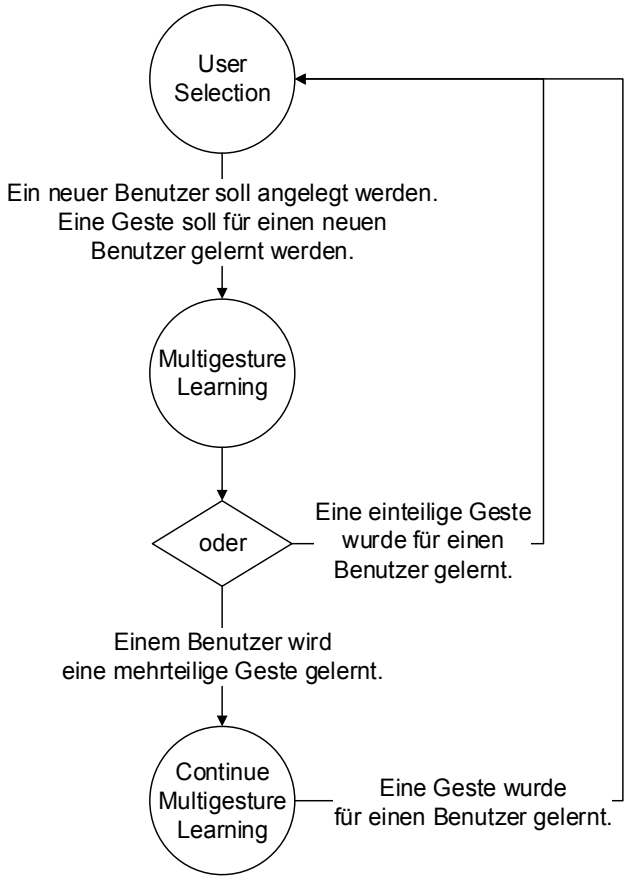

Abbildung 24 - Statediagramm für das Lernen von Gesten

Neben den State „Userselection“ werden für das Lernen von Gesten folgende States verwendet:

- **MultigestureLearning**
  Wird in den Lernmodus gewechselt, also soll dem System eine neue Geste für einen der vier möglichen Befehle gelernt werden, wird in diesen State gewechselt. Sobald eine Bewegung außerhalb der definierten „freien“ Bewegungsgrenzen (default: 10°) erfasst wird, wird das Lernen der Geste gestartet. Es wird in den State „ContinueMultigestureLearning“ für das tracken weiterer Gesten gewechselt.

- **ContinueMultigestureLearning**
  Sobald das Erfassen einer zu lernenden Geste gestartet wurde, wird in diesen State gewechselt. In jeweils 10°-Schritten wird ein neuer Bewegungsvektor erzeugt. Dieser State ist so lange aktiv, bis sich der Benutzer zurück in den Ruhezustand-Bereich bewegt. Dies ist der Fall, falls keine der Rotationachsen mehr als 15° vom Nullpunkt abweicht. Die bis dahin er-

fassten Bewegungsvektoren werden zwischengespeichert und anschließend in der jeweiligen Benutzerdatei abgespeichert.

Der Workflow der Befehlserkennung und –verarbeitung basiert auf der eben vorgestellten Statemachine. Je nach aktuellen Status wird so entweder die Ausführung einer Geste durch den Benutzer gewartet, eine durchgeführte Geste erkannt oder im Lernmodus eine neue Geste dem System gelernt. Werden die Grenzen zur Erfassung einer neuen Befehlsgeste überschritten, wird die Gestenerkennung gestartet. Die Daten werden an das nächste Modul zur Erkennung und Verarbeitung von Befehlsgesten weitergeleitet.

Für die programminterne Koordinierung des Ablaufs dient die Methode *workOnFacedata* (siehe Listing 9). Je nach aktuellem State (Variable *actualState*) wird in den jeweiligen if-Zweig gewechselt. Darin werden die Daten, entsprechend des aktuellen States, weiterverarbeitet.

```
public static void workOnFacedata(Facedata newFacedata){

    gestureRecognition = GestureRecognitionMulti.getInstance();

    String myCommand = "";

    if (actualState == States.UserSelection){
          // Waits, until a user is selected in the user-interface
          [...]
    }

    else if (actualState == States.Idle){
          // Starts a new command recognition
          // Sets first idle-pose after start of the user interface
          [...]
    }

    if (actualState == States.IdleReset){
          // Starts timer, when entered (e. g. Gesturerecognition
          // completed)
          // If timelimit is matched, idle-pose is reseted

          // Changes to state "CommandRecognition" when threshold is
          // undershot
          [...]
    }

    if (actualState == States.CommandRecognition){
          // Waits, until a new gesture is started (threshold is exceeded)
          // Forwards actual pose to recognition-module

          // Revieves command if any is recognized.
          [...]
```

```
    }

    else if (actualState == States.MultigestureRecognition){
        // Forwards following tacking-data to recognition module if a
        // corresponding command isn't recognized after the first gesture
        // Revieces command if any is recognized by the recognition-module.
        [...]
    }

    else if (actualState == States.CommandRecognitionFinished){
        // Waits, until a new gesture is started (threshold is exceeded)
        // Forwards actual pose to recognition-module
        [...]
    }

    else if (actualState == States.MultigestureLearning){
        // Waits, until a new gesture is started.
        // Forwards pose to learning-method.
        [...]
    }

    else if (actualState == States.ContinueMultigestureLearning){
        // Forwards the actual pose to learning-method as long the complete
        // command-gesture is returned
        [...]
    }

    // Return, if no command is recognized.
    if (myCommand.Equals("")) return;

    // Else: Send command
    else {

        getInstance().movementRevieved(myCommand);
        Console.WriteLine("******* Command " + myCommand + " recognized!
        **********");
        return;

    }

}
```

Listing 9 - Implementierung der Statemachine

4.6 Befehlserkennung und -verarbeitung

Nachdem die Daten erfasst, aufbereitet und normalisiert wurden, werden diese an die Befehlserkennung und –verarbeitung weitergeleitet. Die Befehlserkennung empfängt die aktuell empfangene Pose und vergleicht diese laufend mit den Standard-Befehlsgesten oder bei entsprechender Benutzerauswahl mit den Befehlsgesten des aktuellen Benutzers. Die aktuell gültigen Befehlsgesten werden bei der Auswahl des Benutzers in die Befehlserkennung geladen und dort in Vektorenform abgelegt.

Beginnt der Benutzer eine Geste auszuführen und gelangt bei mindestens einer Rotationsachse über eine Schwelle von 10° Abweichung vom Ruhezustandsnullpunkt, wird die Gestenerkennung aktiviert.

Gestenerkennung

Die Gestenerkennung selbst basiert auf Abweichungsmessungen zwischen den vier gespeicherten Vektoren der Befehlsgesten und der aktuell eingenommenen Pose. Beträgt der Abstand zwischen der aktuellen Pose und der gelernten Geste weniger als 10°, gilt eine Geste als erkannt und der dazugehörige generische Befehl (*Befehl1*, *Befehl2*, *Befehl3* oder *Befehl4*) wird von der Gestenerkennung retourniert (siehe Abbildung 25). Ist die Gestenerkennung nicht eindeutig, befinden sich also noch mehr als eine Geste in der Menge der möglichen Befehlsgesten, wechselt diese in den Modus für mehrteilige Befehlsgesten. Hier werden die noch möglichen Gesten laufend mit der aktuell empfangenen Pose verglichen, bis eine Geste eindeutig erkannt wurde. Wurde nach einer definierten Zeitspanne keine Geste eindeutig erkannt, wird die Gestenerkennung abgebrochen und zurück in den Ruhezustand gewechselt. Da nicht mit Sicherheit angenommen werden kann, dass die Geste eine unkorrekt durchgeführte Befehlsgeste gewesen ist, wird die aktuell eingenommene Pose automatisch als neuer Nullpunkt gesetzt.

Im Rahmen der Gestenerkennung sind folgende Parameter im System definiert:

- Toleranzbereich um dem Nullpunkt: 10°
- Grenzbereich für das Beenden der Gestenerkennung: 15°
- Grenzbereich für das Starten der Gestenerkennung für eine neuen Befehl: 10°
- Zeitspanne bis Reset: 3 Sekunden

Der allgemeine Toleranzbereich (10°) sowie der Toleranzbereich zum Beenden der Gestenerkennung (15°) erklärt sich dadurch, da nach Durchführung einer Geste ein Zurückfedern des Kopfs auftritt. Dadurch soll verhindert werden, dass zum Beispiel unmittelbar nach Durchführen des Befehls zum Erhöhen des Stockwerkzählers, aufgrund des Zurückfederns im selben Bewegungsablauf, der Befehl zum Vermindern des Stockwerkszählers erkannt wird.

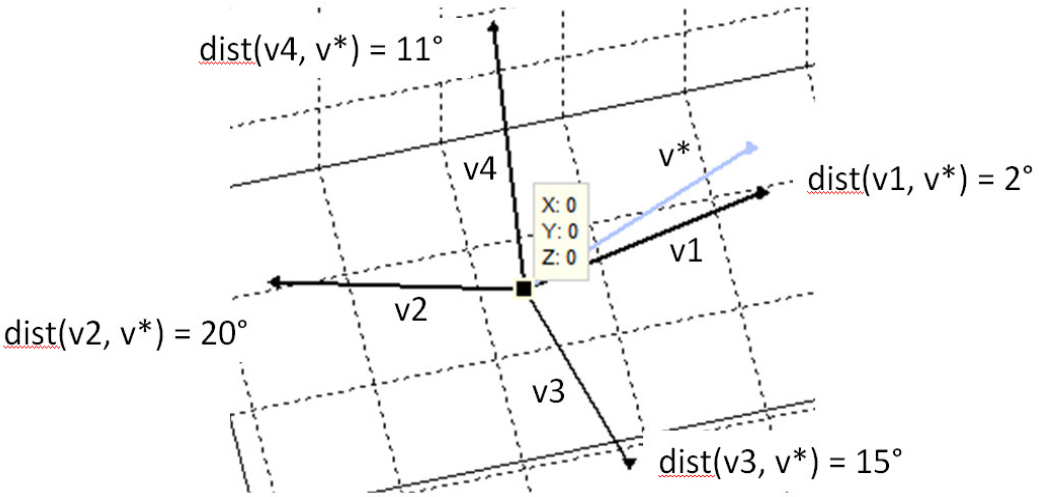

Abbildung 25 – Beispiel Gestenerkennung

Im Beispiel (siehe Abbildung 25) wurden vier Gesten (v1 – v4) im System hinterlegt. Eine neue Geste *v** wird von der Befehlserkennung empfangen. Die Befehlserkennung berechnet nun die Abweichung der neu empfangenen Geste mit jeder der vier im System hinterlegten Gesten. Im Beispiel beträgt, bis auf die Befehlsgeste *v1*, die Abweichung bei jeder anderen Befehlsgeste mehr als 10°. Bei *v1* beträgt die Abweichung 2°. Da sich in der Menge der möglichen Befehlsgesten nur mehr eine Geste befindet, gilt diese in diesem Beispiel als identifiziert. Es wird der zu *v1* korrespondierende Befehl (Befehl 1) zurückgegeben.

In Listing 10 ist der Programmablauf für Starten der Gestenerkennung und die Evaluierung ersichtlich. Es wird bei Empfang einer neuen Kopfbewegung laufend überprüft, ob die Grenze des Toleranzbereichs des Nullpunkts überschritten wird. Wird diese überschritten, wird überprüft ob es in Relation zu einer der vier im System verfügbaren Gesten zu einer Übereinstimmung kommt. Ist dies der Fall, wird diese Befehlsgeste in die Liste der möglichen Lösungen aufgenommen.

```
public String logPoseData(Facedata myFacedata) {
      double distance = 0;
      String result = "";

      Vector3 actualVector = makeVector3(myFacedata);

      if (((myFacedata.rotx < -threshold_start) ||  (myFacedata.rotx >
      threshold_start)) ||
      ((myFacedata.roty < -threshold_start) || (myFacedata.roty >
      threshold_start)) ||
      ((myFacedata.rotz < -threshold_start) || (myFacedata.rotz >
      threshold_start))){

            // Calculating the distance, between
            distance = Vector3.Distance(new Vector3(vl1.First.Value.chgrotX,
            vl1.First.Value.chgrotY, vl1.First.Value.chgrotZ), actualVector);

             if (distance <= threshold_cont) {
                  result = "1";
             }
             else removeVector("1");
            [...]
      [...]

}
```

Listing 10 - Beispiel einer Gestenerkennung bei der erstmaligen Wahrnehmung einer neu ausgeführten Geste

Die Berechnung des Abstands der vorgegebenen Gesten und der neu empfangenen Gesten erfolgt nach dem Prinzip des Satzes von Pythagoras (siehe Abbildung 26). In der Abbildung wird ein Beispiel mit 2-dimensionalen Vektoren dargestellt. Analog dazu verhält sich die Berechnung auch im Falle von 3-dimensionalen Vektoren.

Die Verwendung des Vektorabstands zur Bestimmung, ob eine durchgeführte Geste einer im System gespeicherten Geste entspricht, stellt eine, in Vergleich zu weiteren Möglichkeiten wie zum Beispiel Hidden-Markov-Modellen, relativ einfach zu realisierende aber trotzdem sehr zuverlässige Möglichkeit dar.

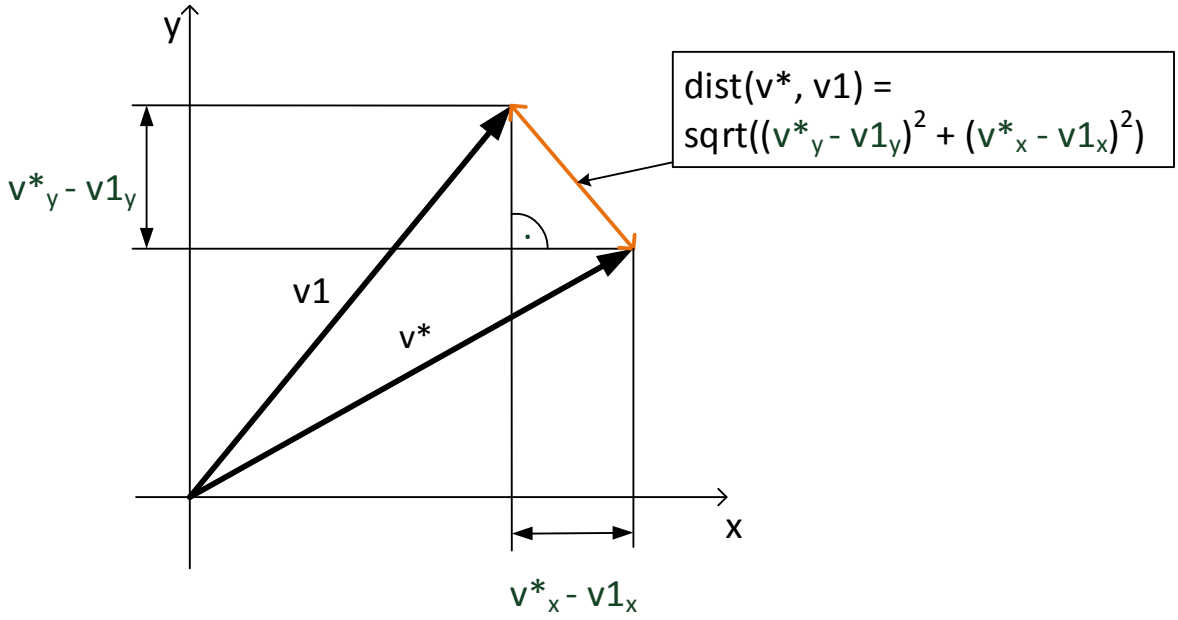

Abbildung 26 - Berechnung des Abstands zwischen zwei Vektoren

4.7 Befehlsgenerierung

Wurde ein Befehl vom Benutzer bestätigt, wird dieser in eine XML-Nachricht gepackt und via eines TCP-Sockets an den Empfänger des Befehls weitergeleitet (siehe Listing 11). Der Empfänger des Befehls kann sowohl ein Bus-System als auch ein direkt empfangsbereites Gerät sein.

```
String message = "<Command><DestinationFloor>" + messageCommand +
"</DestinationFloor></Command>";

Console.WriteLine("Sent Command: " + messageCommand);

SocketThread.send(message);
```

Listing 11 – Quellcode zur Befehlsgenerierung

4.8 Aufzugssimulator

Der Aufzugssimulator (siehe Abbildung 27) empfängt den Befehl der Benutzerschnittstelle und führt ihn aus. Wurde eine xml-Nachricht empfangen, wird diese an die Methode *convertXmlData* übergeben (siehe Listing 12). In dieser Methode wird die xml-Nachricht in eine weiterverarbeitbare Datenstruktur (Datentyp xmlDocument) gespeichert. Es wird überprüft, ob die empfangene Nachricht ein

Element des Typs „Command“ beinhaltet. Ist dies der Fall, wird der Befehl ausgelesen und ein Event des Typs *commandRecieved* ausgelöst, in dem der Befehl zur weiteren Verarbeitung beinhaltet ist. Der Event wird vom grafischen Auszugssimulator empfangen und der darin enthaltene Befehl ausgeführt.

```
[...]
public event CommandRevieved commandRevieved;

public static void convertXmlData(String myData){

        String destinationFloor = null;
        XmlDocument xmlDoc = new XmlDocument();

        //Loads the revieved String to an xml-datastructure and get the first
        element.
        xmlDoc.Load(new StringReader(myData));
        XmlElement element = xmlDoc.DocumentElement;

        //Checks, if the message contains a "command", extracts the command and
        throws event “commandRevieved” containing the extracted command
        if (element.Name.Equals("Command")){
              if (element.HasChildNodes){
                    for (int j = 0; j < element.ChildNodes.Count; j++){
                          if(element.ChildNodes[j].Name.Equals("DestinationFloor
                                ")){

                          destinationFloor = element.ChildNodes[j].InnerText;

                          Console.WriteLine("Destination-Floor: " +
                                element.ChildNodes[j].InnerText);

                          }
                    }

                    getInstance().commandRevieved(destinationFloor);

                    }
              }
         }
}
```

Listing 12 - Auslesen der Daten der XML-Nachricht und Weiterverarbeitunng des darin enthaltenen Befehls

Die Stockwerke werden in der Rechenfolge des Empfangs von der virtuellen Aufzugkabine angefahren. Befindet sich der Aufzug im Normalbetrieb, wird dies durch einen schwarzen Hintergrund und einem Statusfeld („Normalbetrieb“) dargestellt. Die Aufzugkabine bewegt sich schrittweise zum gewünschten Zielstockwerk.

Abbildung 27 – Aufzugsimulator

Der Notruf und Notstop wird grafisch im Aufzugsimulator visualisiert. Beim Notstop färbt sich der Simulator rot (siehe Abbildung 28) und das Statusfeld zeigt den entsprechenden Status an („Notstop“). Im Falle eines Notrufs färbt sich der Aufzugsimulator Violett (siehe Abbildung 29). Wird in der Benutzerschnittstelle anschließend ein neues Stockwerk ausgewählt, wechselt der Aufzug wieder zurück in den Modus für den Normalbetrieb.

Abbildung 28 – Aufzugsimulator Visualisierung Notstop

Abbildung 29 – Aufzugsimulator Visualisierung Notruf

5. Evaluierung der Steuerung anhand des Aufzugsimulators

Zur Evaluierung der Benutzerschnittstelle soll diese anhand eines Tests unter realistischen Bedingungen getestet werden. Mit Hilfe des Tests sollen sowohl die praktische Benutzbarkeit aber auch mögliche Verbesserungspotentiale der Steuerung ermittelt werden. Im Rahmen des Tests soll weiters ein Fragebogen mit den Testpersonen durchgegangen werden. In diesem Kapitel soll der Testaufbau, die Testpersonen, die Testfälle sowie die Ergebnisse des Tests beschrieben werden.

5.1 Testaufbau und -ablauf

Die Benutzerschnittstelle soll anhand eines aufgabenorientierten Benutzertests evaluiert werden. Im Laufe der Testdurchführung soll parallel ein Fragebogen durchgearbeitet werden. Während des Benutzertests werden die Testpersonen aufgefordert, an reale Bedingungen angelehnte Aufgabenstellungen mit Hilfe der Benutzerschnittstelle zu absolvieren, z. B.: „Fahren Sie in den 2. Stock“. Der Benutzertest wird als „Thinking aloud“-Test durchgeführt. Die Testpersonen werden aufgefordert, ihre Gedanken laut auszusprechen. Sie sollen dadurch die Möglichkeit bekommen, auftretende Fragen und Probleme während des Testvorgangs direkt anzusprechen.

Mit Hilfe eines Fragebogens sollen die Testpersonen während und am Ende des Tests interviewt werden. Im Laufe dieses Fragebogens werden sie zu ihrer Zufriedenheit, zur Zuverlässigkeit des Systems, zur Steuerung, zu ihren bisherigen Erfahrungen mit gestenbasierten Steuerungen sowie zu ihrer persönlichen Meinung zum Entwicklungspotential derartiger Steuerungen befragt.

Inklusive Vor- und Nachbesprechung soll ein Benutzertest in etwa 30 Minuten dauern. Für die Durchführung des Benutzertests wird ein Notebook mit eingebauter Webcam verwendet, auf dem alle notwendigen Module des Steuerungssystems (Benutzerschnittstelle, Gestenerkennung, Aufzugssimulator) ausgeführt werden. Für die spätere Evaluierung des Benutzertests sowie zur besseren Nachvollziehbarkeit der erhaltenen Antworten werden Video- und Audioaufzeichnungen des Tests angefertigt. In Abbildung 30 wird der Versuchsaufbau während eines Benutzertests dargestellt.

Der Ablauf eines Benutzertests teilt sich in mehrere Phasen: Nach der Vorstellung und einer kurzen Einführung in den Umgang mit der Benutzerschnittstelle werden die Benutzer gebeten, vorgegebene

Aufgaben mit Hilfe der Benutzerschnittstelle zu lösen. Die Absolvierung dieses ersten Aufgabenblocks soll dabei helfen, Erkenntnisse zu der Benutzbarkeit und der Erkennungsgenauigkeit der Benutzerschnittstelle zu gewinnen, wobei die Benutzer in dieser Phase ohne detaillierte Kenntnis über die Handhabung der Benutzerschnittstelle verfügen sollen. Nach einer kurzen Befragung zur subjektiv wahrgenommenen Erkennungsgenauigkeit der durchgeführten Befehlsgesten dieses ersten Aufgabenblocks wird den Benutzern einige Minuten Zeit gegeben, um sich an das System zu gewöhnen. Die Benutzer haben in diesem Zeitraum die Möglichkeit, frei mit dem System zu interagieren und sich die notwendigen Befehle besser einzuprägen. Nach der Eingewöhnungszeit werden die identen Aufgaben des ersten Aufgabenblocks ein zweites Mal absolviert. Hier soll sich zeigen, ob und wie sich die Erkennungsgenauigkeit im Vergleich zur ersten Absolvierung verändert. Wieder werden die Testpersonen zur subjektiven Wahrnehmung zur Erkennungsgenauigkeit befragt. Anschließend wird mit den Testpersonen ein dritter Aufgabenblock absolviert. Sie sollen nun eigene Befehlsgesten definieren und das System mit diesen selbst gelernten Gesten steuern. Diese Aufgabenstellung soll zeigen, ob eine individuelle Gestenerkennung bei verschiedenen Benutzern möglich ist. Abschließend werden, neben Fragen zur diesmal wahrgenommenen Erkennungsgenauigkeit, auch weitere allgemeine Fragen gestellt (siehe Fragebogen im Anhang, Seite 52).

Abbildung 30 – Versuchsaufbau

Die Benutzertests orientieren sich im Detail an folgendem Ablauf:

1. Begrüßung der Testperson
2. Vorstellung des Systems und der Testumgebung
3. Erklärung an die Versuchsperson von Ziel und Zweck des Tests; Hinweis für Testperson: Nicht die Person sondern das System wird getestet, dadurch keine „schlechten" Ergebnisse oder „falsches" Verhalten von den Testpersonen möglich
4. Erläuterung der Testmethode (inkl. Thinking aloud)
5. Kurze Einführung in die Aufgaben
6. Anfängliche Fragen der Testpersonen beantworten
7. Beantwortung des ersten Fragenblocks (Frageblock A: Vor dem Benutzertest) mit allgemeinen Fragen
8. Absolvierung der Aufgaben (Aufgabenblock 1**)** mit vorgegeben Gesten
9. Beantwortung des zweiten Fragenblocks (Frageblock B: Nach der Eingewöhnungsphase)
10. 5-minütige Gewöhnungsphase (Freies Benutzen der Steuerung)
11. Absolvierung der Aufgaben (Aufgabenblock 1**)** mit vorgegeben Gesten
12. Beantwortung des dritten Fragenblocks (Frageblock C: Nach Absolvierung der Aufgaben nach einer kurzen Eingewöhnungsphase)
13. Lernen individueller Gesten und Absolvierung der Aufgaben (Aufgabenblock 2)
14. Beantwortung des vierten und letzten Fragenblocks (Frageblock D: Nach Absolvierung aller Aufgaben (inkl. selbst definierten Befehlen)
15. Abschließende Kommentare der Testpersonen erfassen
16. Verabschiedung

5.1.1 Testfälle

Der Test wird als Benutzertest durchgeführt. Im Rahmen dieses Benutzertests sollen die Testpersonen vorgegebene Aufgaben mit Hilfe der berührungsfreien Benutzerschnittstelle durchführen. Die Aufgaben setzen sich aus realitätsnahen Anwendungsfällen zusammen, z. B. „Fahren Sie mit dem Aufzug in den zweiten Stock".

Im Folgenden werden die Aufgaben dargestellt, die von den Testpersonen durchgeführt wurden:

Aufgabenblock 1:

Vorbedingungen:

Ein Testbenutzer mit einem vorgegebenen Set an vier Befehlen wurde angelegt und im Benutzer „Standard" abgespeichert.

Aufgaben:

- **Starten Sie das Programm und wählen Sie den Benutzer „Standard“ aus.**
  1. Auswahl des Benutzers „Standard“ aus dem DropDown-Menü.

- **Der Aufzug befindet sich im Erdgeschoß. Sie wollen in den ersten Stock fahren.**
  1. Betätigen des Befehls für „Stock +1“.
  2. Bestätigen des Befehls.

- **Der Aufzug befindet sich im 1. Stock. Sie wollen wieder zurück ins Erdgeschoß fahren.**
  1. Betätigen des Befehls für „Stock -1“.
  2. Bestätigen des Befehls.

- **Der Aufzug befindet sich im Erdgeschoß. Sie wollen in den dritten Stock fahren.**
  1. Betätigen des Befehls für „Stock +1“.
  2. Betätigen des Befehls für „Stock +1“.
  3. Betätigen des Befehls für „Stock +1“.
  4. Bestätigen des Befehls.

- **Der Aufzug befindet sich im dritten Stock. Sie wollen ins Erdgeschoß fahren.**
  1. Betätigen des Befehls für „Stock -1“.
  2. Betätigen des Befehls für „Stock -1“.
  3. Betätigen des Befehls für „Stock -1“.
  4. Bestätigen des Befehls.

- **Betätigen Sie den Not-Stop.**
  1. Betätigen des Befehls für das Wechseln in den Modus für Spezialbefehle.
  2. Betätigen des Befehls für „STOP“.
  3. Bestätigen des Befehls.

Aufgabenblock 2:

- **Legen Sie einen neuen Benutzer im System an und lernen Sie dem System vier Befehle.**
  1. Öffnen der Funktion zum Anlegen eines neuen Benutzers.
  2. Definieren Sie einen Benutzernamen.
  3. Betätigen der Geste für „Befehl1“.
  4. Betätigen der Geste für „Befehl2“.
  5. Betätigen der Geste für „Befehl3“.
  6. Betätigen der Geste für „Befehl4“.
  7. Schließen des Formulars.

- **Wählen Sie den neu angelegten Benutzer aus.**
  1. Auswahl des angelegten Benutzers aus dem Drop-Down Menü.

- **Der Aufzug befindet sich im Erdgeschoß. Sie wollen in den ersten Stock fahren.**
  1. Betätigen des Befehls für „Stock +1“.
  2. Bestätigen des Befehls.

- **Der Aufzug befindet sich im 1. Stock. Sie wollen wieder zurück ins Erdgeschoß fahren.**
  1. Betätigen des Befehls für „Stock -1“.
  2. Bestätigen des Befehls.

- **Der Aufzug befindet sich im Erdgeschoß. Fahren Sie in den zweiten Stock.**
  1. Betätigen des Befehls für „Stock +1“.
  2. Betätigen des Befehls für „Stock +1“.
  3. Bestätigen des Befehls.

- **Der Aufzug befindet sich im zweiten Stock. Fahren Sie ins Erdgeschoß.**
  1. Betätigen des Befehls für „Stock -1“.
  2. Betätigen des Befehls für „Stock -1“.
  3. Bestätigen des Befehls.

- **Betätigen Sie den Notruf.**
  1. Betätigen des Befehls für das Wechseln in den Modus für Spezialbefehle.
  2. Betätigen des Befehls für den „Notruf“.
  3. Bestätigen des Befehls.

5.1.2 Fragebogen

Vor, während und nach Absolvierung der Aufgaben durch die Testpersonen wurden Fragen eines Fragebogens abgefragt. Mit Hilfe des Fragebogens wurden allgemeine Fragen zu gestenbasierten Steuerungen (bisherige Erfahrungen, persönliche Meinung zur Benutzbarkeit und Praxistauglichkeit, etc.) sowie die subjektive Wahrnehmung der Testpersonen zur Zuverlässigkeit der Benutzerschnittstelle abgefragt. Der Fragebogen, der während und nach dem Benutzertest mit den Testpersonen durchgegangen wurde, befindet sich im Anhang (Seite 52).

5.1.3 Testpersonen

Für den Test der Benutzerschnittstelle wurden fünf Testpersonen ausgewählt. Um die verschiedenen physischen Voraussetzungen der künftigen Benutzer realistisch nachzubilden, wurden Testpersonen aus drei Personengruppen ausgewählt:

- Testpersonen ohne relevante physische Beeinträchtigung

- Testpersonen mit relevanter physischer Beeinträchtigung
- Testpersonen mit schwerer relevanter physischer Beeinträchtigung

Als relevante physische Beeinträchtigung für die Durchführung des Benutzertests wird eingeschränkte Mobilität, beispielsweise das Angewiesen sein auf einen Rollstuhl oder Rollator, oder die fehlende Möglichkeit, Arme und Hände zielgerichtet einsetzen zu können, betrachtet.

Im Folgenden werden Testpersonen sowie deren physische Beeinträchtigung, falls vorhanden, näher beschrieben.

Testpersonen ohne relevante physische Beeinträchtigung

- Testperson1 (weiblich, 28 Jahre): Brillenträgerin
- Testperson2 (weiblich, 56 Jahre): Brillenträgerin

Testpersonen mit relevanter physischer Beeinträchtigung

- Testperson 3 (Weiblich, 80 Jahre): Auf Rollstuhl angewiesen, schwere Augenkrankheit (Makuladegeneration) und dadurch auf einem Auge blind, am zweiten Auge 10% Sehleistung. Arme und Hände können ohne Einschränkung verwendet werden.

Testpersonen mit schwerer relevanter physischer Beeinträchtigung

- Testperson 5 (Männlich, 40 Jahre): Multiple Sklerose, zeitweise schwere Spastiken. Auf Rollstuhl angewiesen. Keine Zielgerichteten und koordinierten Arm- und Handbewegungen möglich. Ganztägige Betreuung durch persönliche Assistenten. Brillenträger
- Testperson 6 (Weiblich, 41 Jahre): Multiple Sklerose, zeitweise schwere Spastiken. Auf Rollstuhl angewiesen. Keine Zielgerichteten und koordinierten Arm- und Handbewegungen möglich. Ganztägige Betreuung durch persönliche Assistenten.

5.2 Ergebnisse und Diskussion

Mit den Testpersonen wurden in Summe 95 Aufgaben (19 Aufgaben pro Testperson) absolviert. Zur korrekten Absolvierung dieser Aufgaben mussten insgesamt 265 Befehlsgesten (53 pro Person) ausgeführt werden. Alle Aufgaben und Befehlsgesten konnten von den Testpersonen erfolgreich durchgeführt werden.

Schon beim ersten Durchlauf der Aufgaben ohne Eingewöhnungszeit konnten die Testpersonen alle Aufgaben problemlos absolvieren. Der gefühlte Erkennungsgrad der einzelnen vom Benutzer durchgeführten Gesten war mit 70-100% bereitsauf einem hohen Niveau (siehe Tabelle 1). Tatsächlich

wurden in dieser Phase schon durchschnittlich 91% aller Gesten korrekt vom System erkannt (siehe Tabelle 2 und Tabelle 3).

Nach Absolvierung der gleichen Aufgaben nach einer kurzen Eingewöhnungsphase konnte die Erkennungsrate weiter verbessert werden. Die Evaluierung des Benutzertests ergab, dass die Benutzerschnittstelle eine hohe Praxistauglichkeit aufwies. Sogar jene Testpersonen mit schwerer körperlicher Beeinträchtigung konnten problemlos mit der Benutzerschnittstelle umgehen. Aufgrund ihrer körperlichen Beeinträchtigung war die Erkennungsgenauigkeit im Vergleich zu den nicht-beeinträchtigten Testpersonen zwar etwas geringer, es konnten dennoch alle Aufgaben problemlos absolviert werden.

Erkennungsrate je Testperson (TP)	**TP1**	**TP2**	**TP3**	**TP4**	**TP5**
Aufgabenblock 1 ohne Eingewöhnungszeit	70-90%	70-90%	70-90%	90-100%	90-100%
Aufgabenblock 1 nach einer Eingewöhnungsphase	70-90%	70-90%	90-100%	70-90%	90-100%
Aufgabenblock 2 mit individuellen Gesten (ohne Eingewöhnungszeit)	90-100%	70-90%	90-100%	70-90%	70-90%

Tabelle 1 - gefühlte Erkennungsrate des Benutzertests je Testperson (TP)

Erkennungsrate je Testperson (TP)	**TP1**	**TP2**	**TP3**	**TP4**	**TP5**
Aufgabenblock 1 ohne Eingewöhnungszeit	100%	94%	69%	100%	94%
Aufgabenblock 1 nach einer Eingewöhnungsphase	88%	100%	100%	88%	100%
Aufgabenblock 2 mit individuellen Gesten (ohne Eingewöhnungszeit)	100%	90%	95%	95%	96%

Tabelle 2 – tatsächliche Erkennungsrate des Benutzertests je Testperson (TP)

Erkennungsrate je Testperson (TP)	Durchschnitt tatsächliche Erkennungsrate
Aufgabenblock 1 ohne Eingewöhnungszeit	91%
Aufgabenblock 1 nach einer Eingewöhnungsphase	95%
Aufgabenblock 2 mit individuellen Gesten (ohne Eingewöhnungszeit)	93%

Tabelle 3 – durchschnittliche tatsächliche Erkennungsrate des Benutzertests je Testperson (TP)

Bei den körperlich eingeschränkten Testpersonen stellten sich am Ende des Benutzertests Ermüdungserscheinungen ein. Da die Benutzerschnittstelle in der Praxis nicht durchgehend sondern nur im Bedarfsfall benutzt wird, wird die körperliche Anstrengung, die zur Bewegung des Kopfs notwendig ist, in Grenzen gehalten. Im aktuellen Anwendungsfall eines Aufzugs sind alle möglichen Funktionen in zwei (z. B. „Stock+1“, „bestätigen“) bis maximal fünf Gesten (vier Mal „Stock+1“, „bestätigen“) durchführbar.

Alle vom System vorgegebenen Befehlsgesten konnten von allen Benutzern nach einer einmaligen kurzen Erklärung durchgeführt werden. Die Testpersonen konnten sich die häufigsten gebrauchten Gesten zum Erhöhen und Vermindern des Stockwerkzählers sowie die Geste zum Bestätigen der Auswahl sofort einprägen und bei Bedarf spontan ohne Überlegungszeit anwenden. Die vorgegeben Befehle wurden von allen Personen als intuitiv beschrieben. Jene Aufgaben, die die vierte und am seltensten benutzte Befehlsgeste erforderten, konnten alle Testpersonen spätestens nach einer kurzen Denkpause von wenigen Sekunden ebenfalls erfolgreich absolvieren.

Wurden die Benutzer gebeten, im Rahmen des Tests individuelle Gesten zu definieren, wurden meist dieselben Gesten des Standardbenutzers gewählt. Jene Benutzer, die dem System davon abweichende Gesten gelernt haben, hatten teilweise Probleme, sich an die selbst definierten Gesten zu erinnern. Jene Benutzer, die dem System vertraute Gesten gelernt haben, konnten die Erkennungsrate leicht steigern, da die gelernten Befehlsgesten nun mehr den individuellen Bewegungsablauf des Benutzers entsprachen, als die vergleichsweise „perfekt“ definierten Gesten des Standardbenutzers. Jene Benutzer, die dem System komplett neue Gesten gelernt haben, haben beim Lösen der Aufgaben teilweise die davor verwendeten und gewohnten Befehlsgesten des Standardbenutzers verwendet. Hier wäre es sinnvoll, die aktuell verfügbaren Gesten visuell darzustellen.

Im Rahmen des Benutzertest wurden die Benutzer gebeten, Verbesserungsvorschläge zu nennen. Die relevantesten Verbesserungsvorschläge lauteten wie folgt:

- **visuelle Darstellung der Befehlsgesten**

  Es wäre sinnvoll, dem Benutzer in einer künftigen Version die Möglichkeit zu geben, sich die gelernten Gesten beispielweise anhand eines 3-dimendional animierten Modells eines Kopfs vorführen zu lassen. Dies würde auch Benutzen, welche die Standard-Befehlsgesten nach längerer Verwendungspause vergessen haben, helfen, sich an die Befehle zu erinnern.

- **größere Darstellung der Elemente der Benutzerschnittstelle**

  Für Menschen mit einer Sehbeeinträchtigung wäre eine größere Schrift hilfreich. Für die letzten beiden Testpersonen wurde dieser Verbesserungsvorschlag schon umgesetzt. Testperson 3 mit nur 10% Sehleistung konnte somit die relevanten Bereiche der Benutzerschnittstelle (Statusfeld) problemlos lesen.

- **Verwendung von Signalfarben**

  Zur besseren visuellen Wahrnehmung der Befehle und der Ergebnisse wäre eine Darstellung mit Signalfarben sinnvoll. So könnte zum Beispiel das Bestätigen eines Befehls mit grüner Farbe, der Notstop mit roter Farbe usw. hinterlegt werden. Dies ist allerdings nur anwendungsfallspezifisch konfigurierbar.

Nach Evaluierung der von den Testpersonen angeführten Verbesserungsvorschlägen wurde die Benutzerschnittstelle entsprechend angepasst. Die Größe der Benutzerschnittstelle wurde optimiert und ist nun 1300 Pixel breit und 800 Pixel hoch. Die Steuerelemente wurden für eine bessere Orientierung auf der Benutzerschnittstelle umpositioniert. Die Reihenfolge der Befehls-Steuerelemente wurde an die Standardgesten angepasst. So befindet sich nun das Steuerelement für den dritten Befehl (Befehl 3 = Kopf nach links drehen) entsprechend der Standardgeste an die linke Position gerückt. Die Schriftgröße aller Steuerelemente wurden für eine bessere Lesbarkeit von 12 Pixel auf 16 bzw. 24 Pixel vergrößert. Die Schrift für die Darstellung der aktuellen Benutzerauswahl wurde auf 150 Pixel vergrößert (siehe Abbildung 32; alte Variante siehe Abbildung 31).

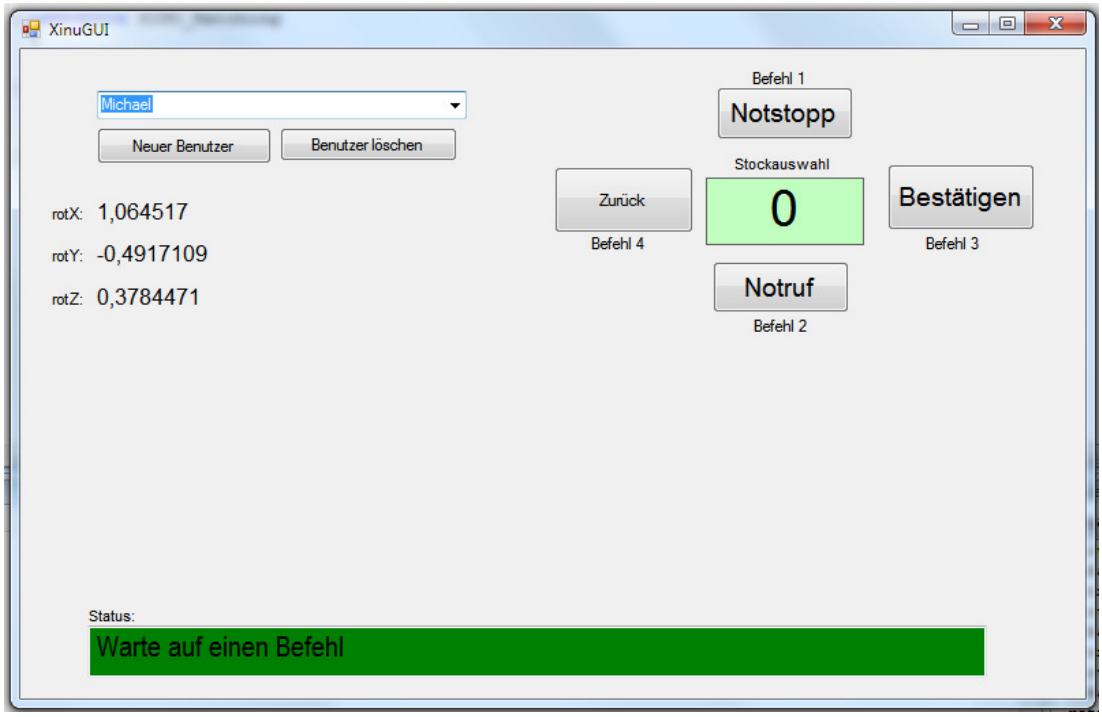

Abbildung 31 – ursprüngliche Variante der Benutzerschnittstelle

Abbildung 32 - Verbesserte Benutzerschnitstelle

Zusammenfassend wurde die Benutzerschnittstelle von allen Testpersonen positiv aufgenommen. Die Testpersonen hoben vor allem die leichte Benutzbarkeit und die flüssige Bedienung positiv hervor. Alle Antworten des Fragebogens (inkl. aller Verbesserungsvorschläge) sind im Anhang zu finden (siehe Anhang, Kapitel „Ergebnisse des Fragebogens“, Seite 72).

6. Zusammenfassung und Ausblick

Ursprung dieser Arbeit war die Problematik, dass die meisten Benutzerschnittstellen in der Praxis eine physische Interaktion, beispielweise das Drücken eines Knopfes, erfordern. Diese Art der Interaktion macht es physisch eingeschränkten Personen nahezu unmöglich, selbstständig den Alltag zu bewältigen. Im Rahmen dieser Diplomarbeit ist es gelungen, mit Standardsoftware und Hardware eine berührungsfreie Benutzerschnittstelle zu konzeptionieren und im Rahmen eines Usecases, der Simulation eines Aufzugs, auf ihre Benutzbarkeit zu testen. Mit Hilfe dieser Benutzerschnittstelle ist es Personen mit eingeschränkter Bewegungsfreiheit nun möglich, mittels Kopfgesten, beispielweise durch Nicken, mit dem System zu interagieren.

Zu Beginn der Arbeit (Kapitel 1 und 2) wurde der Kontext dieser Arbeit und die Problemstellung behandelt. In Kapitel 3 wurde auf die allgemeine Grundlagen und aktuelle Entwicklungen im Kontext der Arbeit eingegangen. Nachdem Systeme der Gebäudeautomation (Abschnitt 3.1) sowie Konzepte alternativer Steuerungsmethoden (Abschnitt 3.2) vorgestellt wurden, wurden in Abschnitt 3.3 Lösungen präsentiert, wie berührungsfreie Benutzerschnittstellen Systeme der Gebäudeautomation das Leben für physisch eingeschränkte Personen vereinfachen können.

In 3.4 wurde das Thema „Facetracking“ behandelt. In Abschnitt 3.4.1 wurden verschiedene Methoden der Head Pose Estimation dargestellt. Im Anschluss daran wurden in Abschnitt 3.4.2 Facetracking-Bibliotheken, insbesondere FaceAPI von SeeingMachines, vorgestellt und die Anforderungen an derartige Systeme dargestellt. In Kapitel 4 wurde beschrieben, wie die gestenbasierte Benutzerschnittstelle konzeptioniert wurde und wie die praktische Umsetzung erfolgt ist. Es wurden die Anforderungen beschrieben. Weiter wurde auf die interne Statemachine eingegangen und im Anschluss daran jedes Modul – das Facetracking-Modul, die Benutzerschnittstelle und der Aufzugsimulator – im Detail beschrieben. Im Zuge der Benutzerschnittstelle wurde auch auf den Algorithmus zur Gestenerkennung eingegangen. Im abschließenden Kapitel 5 wurde die Evaluierung der Benutzerschnittstelle behandelt. Es wurde das Konzept des Benutzertests mit dem Versuchsaufbau (Abschnitt 5.1), den Testfällen (Abschnitt 5.1.1) und den Testpersonen (Abschnitt 5.1.3) dargestellt. Abschließend wurden in Abschnitt 5.2 die Ergebnisse der Evaluierung detailliert beschrieben und diskutiert.

Die aktuelle Software ermöglicht es, vier Befehle entweder durch vorgegebene oder auch individuell definierbare Gesten aufzurufen. Durch die Kommunikation mit Hilfe von XML-Nachrichten, die via TCP/IP-Sockets versendet und empfangen werden, ist auch der Einsatz in zahlreichen weiteren Anwendungsfällen und Situationen denkbar. Beispielsweise könnten mit dieser neuartigen Benutzerschnittstelle, durch Einsatz eines zentralen Steuerungsmoduls zum Empfang der Befehle, diverse

Geräte und Mechanismen in einer Wohnung bedient werden. Es könnten so die Heizung, die Lichtschalter der Wohnräume oder diverse Wohnungsverdunkelungsvorrichtungen wie elektrisch verstellbare Rollläden gesteuert werden. In Verbindung mit einer Infrarotschnittstelle könnten sogar Fernseher und diverse andere TV-, Audio- und Video-Wiedergabegeräte bedient werden. Die Einsatzmöglichkeiten sind nahezu grenzenlos.

Im Rahmen eines Benutzertests wurde die berührungsfreie Benutzerschnittstelle mit Testpersonen mit und ohne physische Einschränkung getestet. Es hat sich gezeigt, dass schon nach einer kurzen Eingewöhnungsphase die Erkennungsgenauigkeit zwischen 70-100% liegt. Mit individuell definierten Befehlsgesten ließ sich die Erkennungsgenauigkeit auf bis zu 90-100% steigern. Durch die Konzeptionierung der Benutzerschnittstelle werden Befehle nur nach expliziter Bestätigung der gewünschten Funktion ausgeführt. Dadurch konnten unabsichtlich oder falsch ausgeführte Befehle verhindert und so alle Testfälle des Benutzertests erfolgreich durchgeführt werden. Eventuell falsch erkannte Befehle konnten sofort rückgängig gemacht werden.

Zur besseren Verwendungsmöglichkeit der Interaktion mit der Benutzerschnittstelle wurden einige Verbesserungsvorschläge im Rahmen des Benutzertests erfasst. Durch eine fehlende Möglichkeit, sich die verfügbaren Gesten grafisch darstellen lassen zu können, ist es nach längeren Verwendungspausen schwierig, sich an die Befehle, sowohl die vorgegeben Befehle aber auch die individuell gelernten, zu erinnern. Die Benutzer verwechselten oft die Befehle untereinander oder hatten Probleme, sich an alle Befehle zu erinnern. Eine Darstellung anhand eines virtuellen Kopf-Abbildes oder die Aufzeichnung der gelernten Gesten mit Hilfe der Webcam für eine spätere Darstellung wäre hierbei hilfreich. Weiters könnte die Erfassung eines individuell gelernten Befehls durch mehrmaliges Durchführen von diesem verbessert werden. Der Durchschnitt der mehrfach durchgeführten Gesten könnte berechnet werden und so helfen, eventuell auftretende Abweichungen bei der Durchführung der Gesten zu kompensieren.

Durch die Integration eines Gesichtserkennungsmoduls wäre es zusätzlich noch möglich, die Benutzerverwaltung weitestgehend zu automatisieren. Das System wäre in der Lage, anhand des Gesichts die Benutzer automatisch zu identifizieren, die gelernten Befehle des Benutzers zu laden sowie bei neuen, fremden Benutzern automatisiert das Lernprogramm zu starten.

Literatur

[Ada13] Adami, I., Antona, M., Stephanidis, C. (2013, forthcoming). "Ambient Assisted Living for the Motor Impaired"., *In G. Koroupetroglou (Ed.) Assistive Technologies, Disability Informatics and Computer Access for Motor Limitations., 2013*

[Bai07] Bailey, M., Chanler, A., Maxwell, B., Micire, M., Tsui, K., & Yanco, H., "Development of vision-based navigation for a robotic wheelchair", *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on,* pp. 951-957, 2007.

[Bai09] Kevon Bailly, Maurice Milgram: "Head Pan Angle Estimation by a Nonlinear Regression on Selected Features", *ICIP 2009*, pp.79-88, Nov. 2009.

[Bet02] Betke, M.; Gips, J.; Fleming, P., "The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on ,* vol.10, no.1, pp.1-10, 2002

[Bru09] D. Bruckner and G. Yin, "Project ATTEND - adapTive scenario recogniTion for emergency and need detection," *Ambient Assisted Living (AAL) Forum,* 2009.

[Bru10] Bruckner, D.; Velik, R., "Behavior Learning in Dwelling Environments With Hidden Markov Models," *Industrial Electronics, IEEE Transactions* on , vol.57, no.11, pp.3653-3660, 2010

[Die00] Dietrich, D., "Evolution potentials for fieldbus systems*," Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on* , pp.145,146, 2000

[Die10] Dietrich, D.; Bruckner, D.; Zucker, G.; Palensky, P., "Communication and Computation in Buildings: A Short Introduction and Overview*," Industrial Electronics, IEEE Transactions on* , vol.57, no.11, pp. 3577-3584, 2010

[Dos09] A. Doshi, S. Y. Cheng, and M. Trivedi, "A novel active heads-up display for driver assistance," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 1, pp. 85 –93, 2009.

[Gee94] A. Gee, R. Cippola: "Determining the gaze of faces in images", *Image and Vision Computing*, vol. 12, no. 10, pp. 639-647, 1994.

[Gee96] A. Gee, R. Cipolla, "Fast Visual Tracking by Temporal Consensus*", Image and Vision Computing*, vol. 14, no. 2, pp. 105-114, 1996.

[Jia07] Pei Jia, Huosheng H. Hu, Tao Lu, Kui Yuan, "Head gesture recognition for hands-free control of an intelligent wheelchair*", Industrial Robot: An International Journal,* vol. 34, no. 1, pp. 60-68, 2007.

[Lab08] Lablack, A.; Zhongfei Zhang; Djeraba, C. :"Supervised Learning for Head Pose Estimation Using SVD and Gabor Wavelets", *Tenth IEEE International Symposium on Multimedia*, pp.592-596, 2008.

[LB06] LeBlanc, D.; Ahmed, Y.B.; Selouani, S.; Bouslimani, Y.; Hamam, H., "Computer Interface by Gesture and Voice for Users with Special Needs," *Innovations in Information Technology 2006,* pp.1-5, Nov. 2006

[Kauf93] Kaufman, A.E.; Bandopadhay, A.; Shaviv, B.D., "An eye tracking computer user interface," *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*, pp.120,121, 1993

[Kim06] Daehwan Kim; Daijin Kim, "An Intelligent Smart Home Control Using Body Gestures", Hybrid Information Technology, 2006. ICHIT '06. International Conference on , vol.2, pp.439,446, 2006.

[Krue97] N. Krüger , M. Pötzsch , C. V. D. Malsburg: "Determination of Face Position and Pose With a Learned Representation Based on Labelled Graphs", Image and Vision Computing, Vol. 15, Issue 8, pp. 665-673, August 1997.

[Mat01] Yoshio Matsumoto, Tomoyuki Ino, Tsukasa Ogasawara, "Development of Intelligent Wheelchair System with Face and Gaze Based Interface*", Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pp. 262-267, 2001.

[Mit07] Mitra, S., Acharya, T., "Gesture Recognition: A Survey", *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, issue 3, pp. 311-324, 2007

[Mur07a] Erik Murphy-Chutorian, Anup Doshi, Mohan Manubhai Trivedi: "Head Pose Estimation for Driver Assistance Systems: A Robust Algorithm and Experimental Evaluation", Proc. of the *IEEE Intelligent Transportation Systems Conference*, pp. 709-714, 2007.

[Mur07b] Erik Murphy-Chutorian, Anup Doshi, Mohan Manubhai Trivedi, "Head Pose Estimation for Driver Assistance Systems: A Robust Algorithm and Experimental Evaluation", *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp.709,714, 2007 .

[Mur09] Erik Murphy-Chutorian, Trivedi Mohan Manubhai: "Head Pose Estimation in Computer Vision: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31 no. 4, pp.607-626, April 2009.

[Niy96] Sourabh Niyogi, William T. Freeman: "Example-Based Head Tracking", *Proc. of the Second International Conference on Automatic Face and Gesture Recognition*, pp.374-378, Okt. 1996.

[Rab86] [18] L. R. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.,* vol. 3, no. 1, pp. 4–16, 1986.

[Roes10] Roesener, C.; Perner, A.; Zerawa, S.; Hutter, S., "Interface for non-haptic control in automation," *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on,* pp.961,966, 2010

[Pre10] Prashan Premaratne, Quang Nguyen, Markin Premaratne, "Human Computer Interaction Using Hand Gestures*", in Proc. of 6th International Conference on Intelligent Computing, ICIC 2010, pp. 381-386., 2010.*

[Shaw90] Shaw, R.; Crisman, E.; Loomis, A.; Laszewski, Z., "The eye wink control interface: using the computer to provide the severely disabled with increased flexibility and comfort," *Computer-Based Medical Systems, 1990., Proceedings of Third Annual IEEE Symposium on*, pp.105,111, 1990

[Tak95] Takami, O.; Morimoto, K.; Ochiai, T.; Ishimatsu, T., "Computer interface to use head and eyeball movement for handicapped people," *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, vol.2, pp.1119-1123, 1995

[Tak96] Takami, O.; Irie, N.; Kang, C.; Ishimatsu, T.; Ochiai, T., "Computer interface to use head movement for handicapped people*," TENCON '96. Proceedings., 1996 IEEE TENCON. Digital Signal Processing Applications*, vol.1, pp.468-472 , 1996

[Uts96] Utsumi, A.; Miyasato, T.; Kishino, F.; Nakatsu, R., "Hand gesture recognition system using multiple cameras*," Pattern Recognition, 1996., Proceedings of the 13th International Conference on* , vol.1, pp.667-671, 1996

[Uts99] Utsumi, A.; Jun Ohya, "Multiple-hand-gesture tracking using multiple cameras," *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.,* vol.1, pp.,478, 1999

[Voi07] Michael Voit, Kai Nickel, Rainer Stiefelhagen: „Neural Network-Based Head Pose Estimation and Multi-view Fusion", *Multimodal Technologies for Perception of Humans: Proc. First Int'l Workshop Classification of Events, Activities and Relationships (CLEAR)*, pp. 291–298, 2007.

[Wis97] Laurenz Wiskott, Jean-Marc Fellous, Norbert Kruger, Christoph von der Malsburg: "Face Recognition by Elastic Bunch Graph Matching", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, Issue 7, pp. 775-779, 1996.

[Yam04] Yamamoto, Y.; Yoda, I.; Sakaue, K., "Arm-pointing gesture interface using surrounded stereo cameras system," *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* , vol.4, pp.965-970, 2004

[Yut07] Satoh, Yutaka, and Katsuhiko Sakaue. "An omnidirectional stereo vision-based smart wheelchair", *EURASIP Journal on Image and Video Processing 2007*, Article ID 87646, 2007

[Zer11] Zerawa, S.-A.; Roesener, C.; Perner, A., "Non-haptic interaction system," *Industrial Electronics (ISIE), 2011 IEEE International Symposium on,* pp.1341-1346, 2011

Internetreferenzen

[1] Seeing Machines – FaceAPI, Seeing Machines Face Tracking API Documentation, http://www.seeingmachines.com, 2010.

[2] Wikipedia, "Gebäudeautomation", http://de.wikipedia.org/wiki/Gebäudeautomation, abgerufen am 20.5.2013.

[3] Seeing Machines, faceAPI Technical Specifications Brochure, www.SeeingMachines.com, abgerufen am 7.4.2013.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Anhang

Fragebogen-Vorlage für den Benutzertest

Name: _____

Frageblock A: Vor dem Benutzertest

A.1 Hatten Sie schon Erfahrung in der Benutzung von gestenbasierten Steuerungen? (Spielekonsole, Handy-Gesten, Mausgesten etc.)

Ja Nein

A.1a Wenn ja, womit haben sie schon Erfahrung?

A.1b Wie würden Sie die bisherigen Erfahrung mit diesen Steuerungen beschreiben?

Schlecht eher schlecht gut sehr gut

Frageblock B: Nach der Eingewöhnungsphase

B.1 Wie war der erste Eindruck dieser Benutzerschnittstelle?

Schlecht eher schlecht gut sehr gut

B.2 Was gefällt Ihnen besonders gut?

B.3 Was gefällt Ihnen nicht?

B.4 Wie zuverlässig hat die Steuerung aus Ihrer Sicht funktioniert?

Nicht Zuverlässig eher zuverlässig zuverlässig sehr zuverlässig

B.5 Wie würden sie den Erkennungsgrad der durchgeführten Gesten einschätzen? Wie viele Befehle von 10 Befehlen wurden korrekt erkannt?

<3 3-5 5-7 7-9 9-10

Frageblock C: Nach Absolvierung der Aufgaben nach einer kurzen Eingewöhnungsphase

C.1 Wie hat die Steuerung aus Ihrer Sicht funktioniert?

Nicht Zuverlässig eher zuverlässig zuverlässig sehr zuverlässig

C.2 Wie würden sie den Erkennungsgrad der durchgeführten Gesten einschätzen? Wie viele Befehle von 10 Befehlen wurden korrekt erkannt?

<3 3-5 5-7 7-9 9-10

Frageblock D: Nach Absolvierung aller Aufgaben (inkl. selbst definierten Befehlen)

D.1 Wie ist ihr abschließender Eindruck dieser Benutzerschnittstelle?

Schlecht eher schlecht gut sehr gut

D.2 Was gefällt Ihnen besonders gut?

D.3 Was gefällt Ihnen nicht?

D.4 Wie hat die Steuerung mit individuell gelernten Befehlen aus Ihrer Sicht funktioniert?

Nicht Zuverlässig eher zuverlässig zuverlässig sehr zuverlässig

D.5 Wie würden sie den Erkennungsgrad der durchgeführten Gesten einschätzen? Wie viele Befehle von 10 Befehlen wurden korrekt erkannt?

<3 3-5 5-7 7-9 9-10

D.6 Wie hat die Steuerung mit selbst definierten Gesten im Vergleich zu den Gesten des Standardbenutzers funktioniert?

Schlechter eher schlechter kein Unterschied eher besser viel besser

D.7 Wie empfinden Sie die Steuerung über Kopfgesten?

Schlecht eher schlecht gut sehr gut

D.8 Finden Sie, dass die Steuerung über Kopfgesten eine hilfreiche Möglichkeit für körperlich beeinträchtigte Personen darstellt?

nicht hilfreich eher hilfreich hilfreich sehr hilfreich

D.9 Hat Ihnen das Benutzen dieser Benutzerschnittstelle Spaß gemacht?

Nein eher nicht eher ja ja

D.10 Wie würden Sie abschließend das Potential gestenbasierter Steuerungen für Sie selbst beschreiben?

negativ eher negativ positiv sehr positiv

D.11 Wie würden Sie abschließend das Potential gestenbasierter Steuerungen für körperlich eingeschränkte Personen beschreiben?

Negativ eher negativ positiv sehr positiv

D.12 Gibt es ihrerseits Vorschläge zur Verbesserung der Benutzerschnittstelle oder der Steuerung?

Ergebnisse des Fragebogens

Frageblock A: Vor dem Benutzertest

A.1 Hatten Sie schon Erfahrung in der Benutzung von gestenbasierten Steuerungen? (Spielekonsole, Handy-Gesten, Mausgesten etc.)

Ja	Nein
2	3

A.1a Wenn ja, womit haben sie schon Erfahrung?

-) Von früheren Tests dieser Steuerung

-) Handy-Wischgesten

A.1b Wie würden Sie die bisherigen Erfahrung mit diesen Steuerungen beschreiben?

Schlecht	eher schlecht	gut	sehr gut
	1	1	

Frageblock B: Nach der Eingewöhnungsphase

B.1 Wie war der erste Eindruck dieser Benutzerschnittstelle?

Schlecht	eher schlecht	gut	sehr gut
		2	3

B.2 Was gefällt Ihnen besonders gut?

-) Einfache Bedienung

-) Flüssige Bedienung

-) Bei öfterer Verwendung, keine Probleme

B.3 Was gefällt Ihnen nicht?

-) Die Befehle müssen zügig durchgeführt werden.

-) Teilweise Problematisch bei Brillenträgern und schlechten Lichtverhältnissen.

B.4 Wie zuverlässig hat die Steuerung aus Ihrer Sicht funktioniert?

Nicht Zuverlässig	eher zuverlässig	zuverlässig	sehr zuverlässig
		2	3

B.5 Wie würden sie den Erkennungsgrad der durchgeführten Gesten einschätzen? Wie viele Befehle von 10 Befehlen wurden korrekt erkannt?

<3	3-5	5-7	7-9	9-10

			3	2

Frageblock C: Nach Absolvierung der Aufgaben nach einer kurzen Eingewöhnungsphase

C.1 Wie hat die Steuerung aus Ihrer Sicht funktioniert?

Nicht Zuverlässig	eher zuverlässig	zuverlässig	sehr zuverlässig
		1	4

C.2 Wie würden sie den Erkennungsgrad der durchgeführten Gesten einschätzen?
Wie viele Befehle von 10 Befehlen wurden korrekt erkannt?

<3	3-5	5-7	7-9	9-10
			2	3

Frageblock D: Nach Absolvierung aller Aufgaben (inkl. selbst definierten Befehlen)

D.1 Wie ist ihr abschließender Eindruck dieser Benutzerschnittstelle?

Schlecht	eher schlecht	gut	sehr gut
		1	4

D.2 Was gefällt Ihnen besonders gut?

-) Es können individuelle Gesten gelernt werden.
-) Die Steuerung ist sehr Anwenderfreundlich.
-) Sehr gute Alternative für körperlich eingeschränkte Personen.
-) Steuerung mit individuellen Befehlen besser.

D.3 Was gefällt Ihnen nicht?

-) Teilweise Problematisch bei Brillenträgern und schlechten Lichtverhältnissen.
-) Die Benutzerschnittstelle könnte bunter gestaltet sein.
-) Die Benutzerschnittstelle könnte ansprechender gestaltet werden.
-) Schrift zu klein.

D.4 Wie hat die Steuerung mit individuell gelernten Befehlen aus Ihrer Sicht funktioniert?

Nicht Zuverlässig	eher zuverlässig	zuverlässig	sehr zuverlässig
		2	3

D.5 Wie würden sie den Erkennungsgrad der durchgeführten Gesten einschätzen?
Wie viele Befehle von 10 Befehlen wurden korrekt erkannt?

<3	3-5	5-7	7-9	9-10
			3	2

D.6 Wie hat die Steuerung mit selbst definierten Gesten im Vergleich zu den Gesten des Standardbenutzers funktioniert?

Schlechter	eher schlechter	kein Unterschied	eher besser	viel besser
	1	2	2	

D.7 Wie empfinden Sie die Steuerung über Kopfgesten?

Schlecht	eher schlecht	gut	sehr gut
		2	3

Kommentare:

-) Steuerung ohne Hände bedienbar
-) Zahlreiche Anwendungsmöglichkeiten für eine derartige Steuerung
-) Gut für Menschen, die ohne diese Art der Steuerung nicht mehr selbstständig leben können.
-) Die Steuerung über Kofgesten kann bei entsprechender Behindertung schnelll ermüdend sein.

D.8 Finden Sie, dass die Steuerung über Kopfgesten eine hilfreiche Möglichkeit für körperlich beeinträchtigte Personen darstellt?

nicht hilfreich	eher hilfreich	hilfreich	sehr hilfreich
		1	4

D.9 Hat Ihnen das Benutzen dieser Benutzerschnittstelle Spaß gemacht?

Nein	eher nicht	eher ja	ja
			5

D.10 Wie würden Sie abschließend das Potential gestenbasierter Steuerungen für Sie selbst beschreiben?

Negativ	eher negativ	positiv	sehr positiv
	1		4

D.11 Wie würden Sie abschließend das Potential gestenbasierter Steuerungen für körperlich eingeschränkte Personen beschreiben?

Negativ	eher negativ	positiv	sehr positiv
			5

D.12 Gibt es ihrerseits Vorschläge zur Verbesserung der Benutzerschnittstelle oder der Steuerung?

-) Schrift größer
-) Mehr Farben (Rot für Notstop, Grün für Bestätigen, etc.)
-) Darstellung der möglichen Befehlsgesten.