Masterprojekt 1 - Projektbericht

Edo Kriegsmann

Contents

1 Einleitung **3**
- 1.1 Einleitung . . . 3
- 1.2 Vison . . . 3
- 1.3 Szenarien dieser Vision . . . 4

2 Teilprojekte **5**
- 2.1 Kinect auf Basis von OpenNI / NITE . . . 5
- 2.2 Kinect auf Basis des Microsoft SDKs . . . 6
  - 2.2.1 Color Image Data . . . 6
  - 2.2.2 Depth Data . . . 7
  - 2.2.3 Skeleton Information . . . 9
  - 2.2.4 Ansprechen der Sensoren . . . 11
- 2.3 Emotionserkennung . . . 11
  - 2.3.1 SHORE-Bibliothek . . . 12
  - 2.3.2 Auswertung erkannter Gesichter . . . 13
  - 2.3.3 Fazit der Gesichtserkennung . . . 13

3 Zusammenfassung **14**
- 3.1 Zusammenfassung . . . 14
- 3.2 Ausblick . . . 14

Bibliography **15**

1 Einleitung

1.1 Einleitung

Die Technisierung des Lebensraums zur Steigerung von Aspekten wie Wohnkomfort oder Werbeerfolg schreitet laufend voran. Ein Teilgebiet dieser Technisierung ist das Ermitteln und Bewerten von menschlichen Interaktionen und Emotionen. In diesem Kontext entstand im Jahr 2010 das ”Living Place” Projekt an der HAW, in welchem es unter anderem möglich ist, zukünftige Entwicklungen unter realen Bedingungen zu erproben (vgl. Rahimi, 2010) . Zu diesen Entwicklungen gehören Bilderkennungssysteme, wie reguläre Farb-Kameras, als auch TTL-Kameras und seit einiger Zeit die Farb- und 3D-Kamera „Micrsosoft Kinect“ (vgl. Microsoft, 2011b) Aus diesen Bereichen entstand der Kern dieser Ausarbeitung. So sollen die 3D und Farbinformationen der Kinect Kamera verbunden und zur Ermittlung menschlicher Interaktion und Emotion genutzt werden.

1.2 Vison

Meine Vision dieser Sensoraus- und Bewertung auf Grundlage der Kinect-Kamera sieht vor, einen kostengünstigen Aufbau zur Sensordatenerweiterung weiterer Systeme zu schaffen. Im Gegensatz zu bisherigen Systemen zur Emotionserkennung, welche ausschließlich Kendaten wie Alter und Geschlecht ermitteln können, würde dieses, durch die Zusatzinformation aus dem 3D-Kamerasystem auch auf Interaktionen (Gesten) reagieren können. Auch die Bestimmung von Körpergröße, Bewegungseingeschränktheit etwa durch einen Rollstuhl oder die Entfernung zum System wären möglich.

1.3 Szenarien dieser Vision

Szenario eins: So könnten diese Daten die Basis einer reaktiven Werbetafel schaffen. Diese könnte anhand der Sensordaten Personen erkennen und den Werbekontext entsprechend anpassen. Eine vorbeigehende Mutter (Ermittlung des Alters und Geschlechts über die Auswertung der Farb-Kamerainformationen) mit Kind würden als diese erkannt und der Werbekontext der Situation angepasst. Während dem Kind in etwa eine Schokoladenwerbung mit kindgerechter Aufmachung auf Augenhöhe des Kindes präsentiert wird, so suggeriert man der Mutter mit entsprechend angepasster Werbeinformation, ebenfalls auf Augenhöhe dieser, die Vorteile eines Kaufs des Produktes.

Szenario zwei: Angebracht an einen Fahrkartenautomaten könnte das System etwa Rollstuhlfahrer erkennen, und diesen direkt einen barrierefreien Weg zum Gleis vorschlagen. Auch könnte der Bildschirminhalt des Automaten dem Alter der bedienenden Person angepasst oder entsprechend erweitert werden. Jungen Personen könnte eine höhere technische Affinität unterstellt werden, was komplexere Bedienstrukturen erlauben würde. Bei höherem Alter könnten Bedienelemente (Berührungssensitiver Bildschirm vorausgesetzt) vergrößert werden, um ihnen die Bedienung zu erleichtern. Selbstverständlich müssen die Anpassungen so erfolgen, dass sich auch bei fehlerhafter Erkennung die zu erkennende Person nicht diskriminiert fühlt oder die Bedienung durch Anpassung eher erschwert wird.

2 Teilprojekte

2.1 Kinect auf Basis von OpenNI / NITE

Der erste Abschnitt beschäftigt sich mit dem Microsoft Kinect-Kamerasystem. Da die von Microsoft zwar angekündigte, aber zum Projektbeginn nicht veröffentlichte Entwicklungsumgebung nicht zur Verfügung stand und das rechtzeitige Erscheinen dieser nicht absehbar war, arbeitete ich mich in die Open-Source-Version von ”PrimeSense” ein (vgl. OpenNI, 2011) . Hierbei handelt es sich um eine Kombination der Programme NITE und OpenNI. Diese ermöglichen eine Auswertung der Farb- und 3D-Sensorinformationen der Kinect-Kamera. Ein Auslesen der Mikrofondaten und ein Setzen des Kamera-Servos ist nicht vorgesehen. Die Software basiert auf der Programmiersprache C++. Leider verstrich viel Zeit bei der Umsetzung dieser Lösung, da es regelmäßig zu Problemen mit der Software kam. Diese Probleme traten sowohl bei der Installation der Treiber auf, als auch beim programmieren von Software in der komplexen Softwarestruktur, bedingt durch die mehrteilige Bibliothek. Aufgrund der Verfügbarkeit des Microsoft SDKs, welches deutlich leistungsfähiger und unkomplizierter in der Installation ist, wurde dieser Ansatz nicht weiter verfolgt.

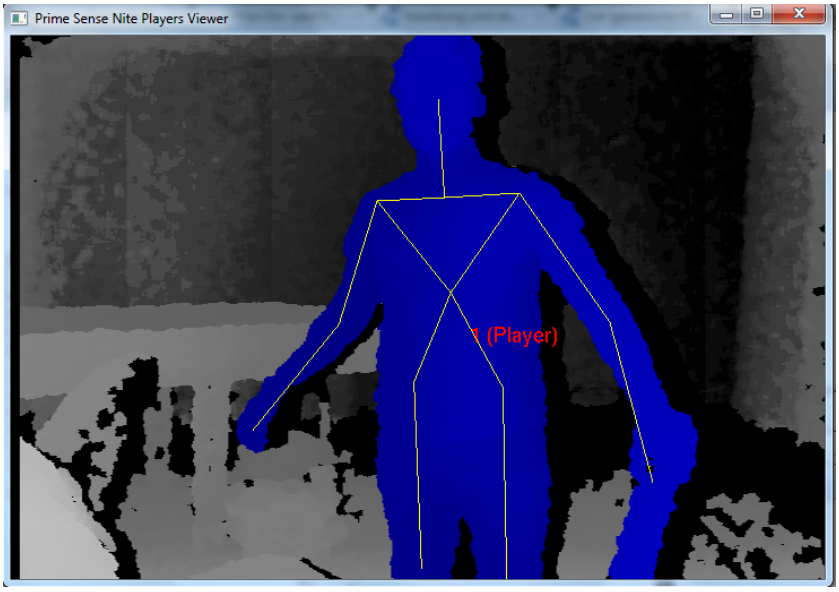

Figure 2.1: *Skeletterkennung über Prime Sense NITE*

2.2 Kinect auf Basis des Microsoft SDKs

Nachdem etwa drei viertel des Projekts abgeschlossen waren, erschien das Microsoft SDK zur Entwicklung von Anwendungen auf Basis der Kinect. (vgl. Microsoft, 2011b) Dieses ermöglicht das Erstellen von Software auf Basis der Programmiersprachen C++ und C#. Im Gegensatz zur Open-Source-Lösung waren die mitgelieferten Treiber des SDKs einwandfrei und ermöglichten einen raschen Beginn der Testphase.

Microsoft stellt in diesem SDK eine NUI-Bibliothek - NUI steht für natural user interface - bereit, welche es dem Entwickler je nach Initialisierung dieser ermöglicht, in der Programmierumgebung auf die Schnittstellen ”Color Image Data”, ”Depth Data” und dem in diesem Projekt nicht genutzten ”Audio Stream” zuzugreifen. (vgl. Microsoft, 2011a, Seite 14) Aus den Werten der ”Color Image Data”- und ”Depth Data”-Schnittstelle, stellt die Kinect des Weiteren die ”Skeleton Information” bereit. Diese Gebiete werden in folgenden Abschnitten genauer betrachtet.

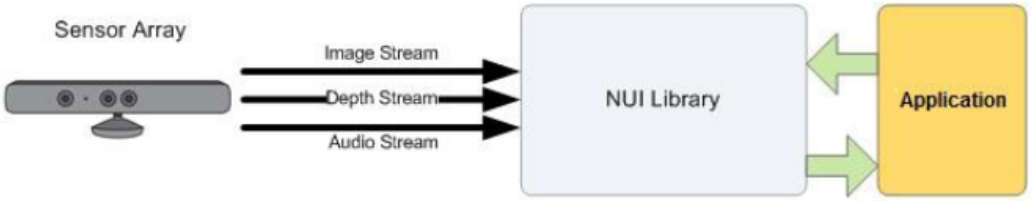

Figure 2.2: *Anbindung der Module in die Software (vgl. Microsoft, 2011a, Seite 14)*

2.2.1 Color Image Data

In diesem Modus ist es möglich, die Werte der Farbkamera der Kinect abzufragen. Hierbei unterstützt die Kinect zwei Modi, den 32-Bit-RGB-Farbmodus und den 16-Bit-YUV-Farbmodus. (vgl. Microsoft, 2011a, Seite 17)

Im Gegensatz zum RGB-Modus ist der YAV-Modus recourcenschonender, steht aber nur im 640x480pixel Format bei 15 Bildern pro Sekunde zur Verfügung.

Diese Farbinformationen sollen später extrahiert und an die Softwareschnittstelle der Emotionserkennung weiter gegeben werden. Dies ermöglicht es beide Systeme mit nur einer Hardware zu betreiben.

2.2.2 Depth Data

Mit dieser Initialisierung des NUI-Interfaces ist es Möglich die Tiefendaten des Kinect-Sensors auszulesen. Hier stehen drei verschiedene Auflösungen zur Auswahl: 640x480, 320x240 und 80x60 Pixel. (vgl. Microsoft, 2011a, Seite 18)

Hierbei besteht jedes Pixel aus einem 16-Bit Datensatz, aus welchem zwei Informationen extrahiert werden können. In den ”upper 12 Bit” befindet sich der Tiefenwert in Millimeter zum nächstgelegenen Objekt dieses Pixels. Aus den ”lower 3 Bit” kann der ”player index” von 0 bis 4 ausgelesen werden, welcher Preis gibt ob und zu welcher Person dieser Tiefenwert gehört. Dieser ”player index” wird automatisch vom ”Player Segmentation”-System der Kinect ermittelt und der erkannten Person zugeordnet. Sind alle 16 Bit auf 0 gesetzt, so ist auf diesem Pixel keine Tiefeninformation verfügbar, da das nächstgelegene Objekt zu nah an der Kamera steht, oder zu weit von dieser entfernt ist.

Diese Tiefeninformationen habe ich im Projekt mit mehreren Demoprogrammen untersucht. Ein Ziel war es, diese Tiefeninformationen in einer variabel großen Matrix erfassen, zu mitteln und diese so gefilterten Daten über ein neues Interface nach außen zu geben. Dies sollte es externer und verhältnismäßig langsamer Hardware, wie einem Arduino, ermöglichen, auf diese Sensorinformationen zuzugreifen. In folgendem Bild 2.3 ist dieser Filter auf dem PC grafisch aufbereitet zu erkennen.

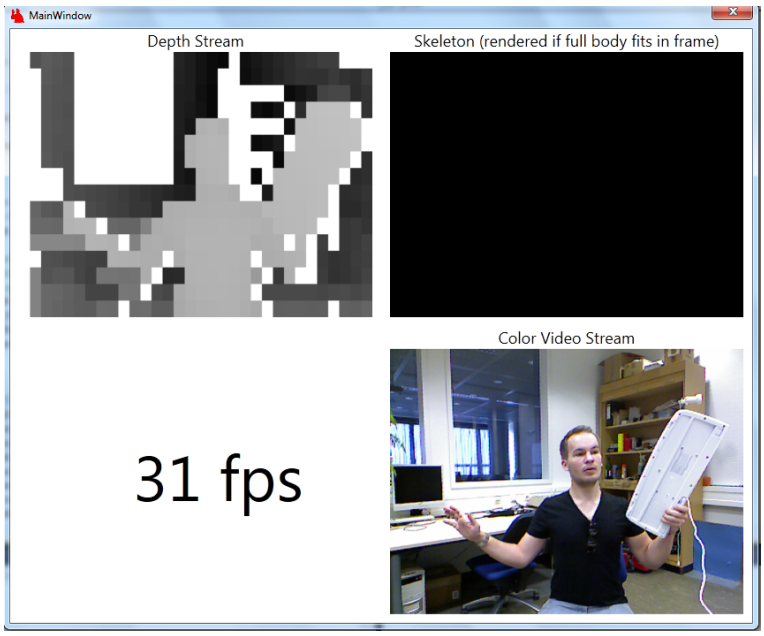

Figure 2.3: *Filterung auf 32x32px a 255-Stufigem Tiefenbild*

Auch war es möglich die Tiefeninformationen durch verschiedene Algorythmen zu filtern. In Grafik 2.4 ist das Tiefenbild neu eingefärbt - ähnlch einer Wärmebildkamera.

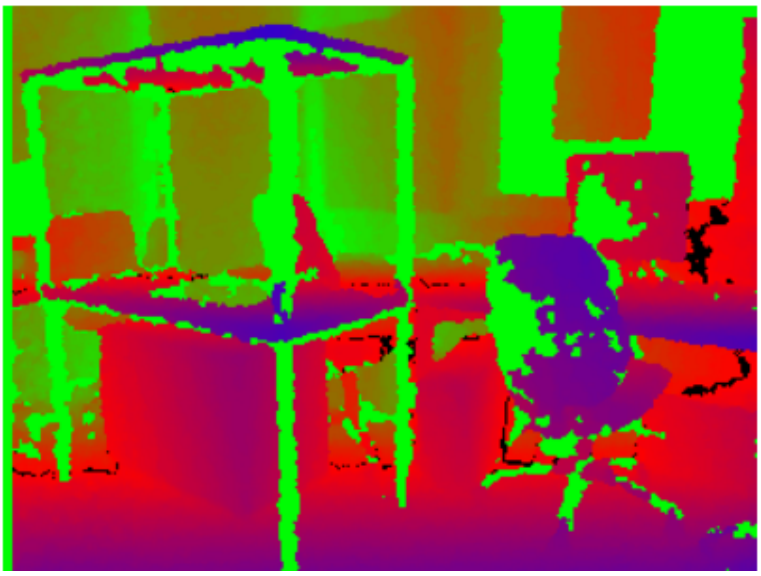

Figure 2.4: *Farbfilterung der Tiefeninformationen*

2.2.3 Skeleton Information

Besonders spannend ist die Kinect-Skeleton-Engine. Diese ermöglicht es einzelne Skelett-Punkte im 3D-Raum der erkannten Personen abzufragen und diese Informationen weiter zu verarbeiten. Wie auch bei der Auswertung der Tiefeninformationen erfolgt die Erfassung und Verarbeitung dieser Daten in Echtzeit. Das ermöglicht es, diese Daten unmittelbar zu verwenden und auf Eingaben direkt reagieren zu können.

Hierbei ist es möglich für jeden Spieler - bis zu zwei - folgende 21 Skelett-Punkte abzufragen. (vgl. Microsoft, 2011a, Seite 20)

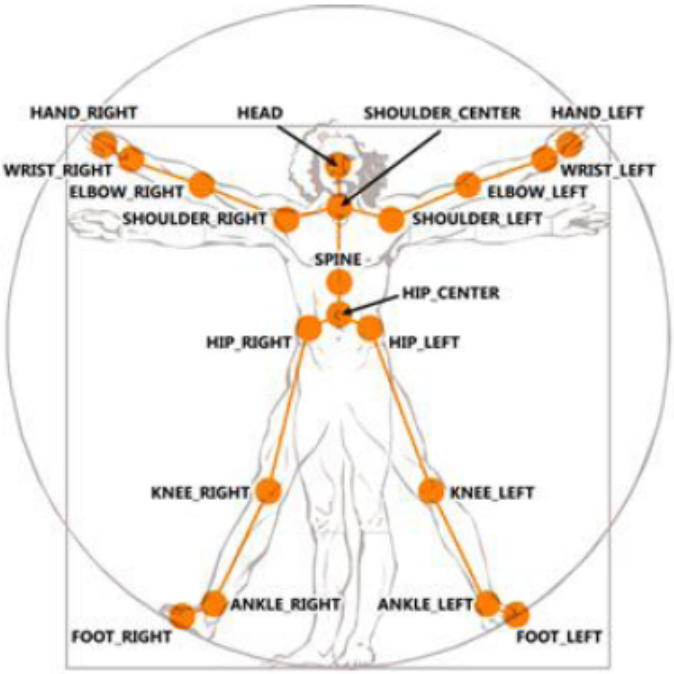

Figure 2.5: *Abfragbare Skelett-Punkte (vgl. Microsoft, 2011a, Seite 20)*

Im Projekt habe ich eine Art Spiel entwickelt, um diese Möglichkeit der Skeletterkennung zu erproben. Dabei las ich die Skelettinformationen der bis zu zwei Spieler aus und rechnete in das auf dem Monitor angezeigte Skelettbild ein ”Laserschwert” ein. Dies war ein um Faktor zwei verlängerter Vektor, erstellt aus der Differenz zwischen XYZ-Koordinate des Ellenbogens und des Handgelenks, addiert zur Koordinate des Handgelenks. Der Effekt war ein sich im 3D-Raum korrekt nach dem Unterarm gerichtetes ”Laserschwert’ welches auf Bild 2.6 zu sehen ist.

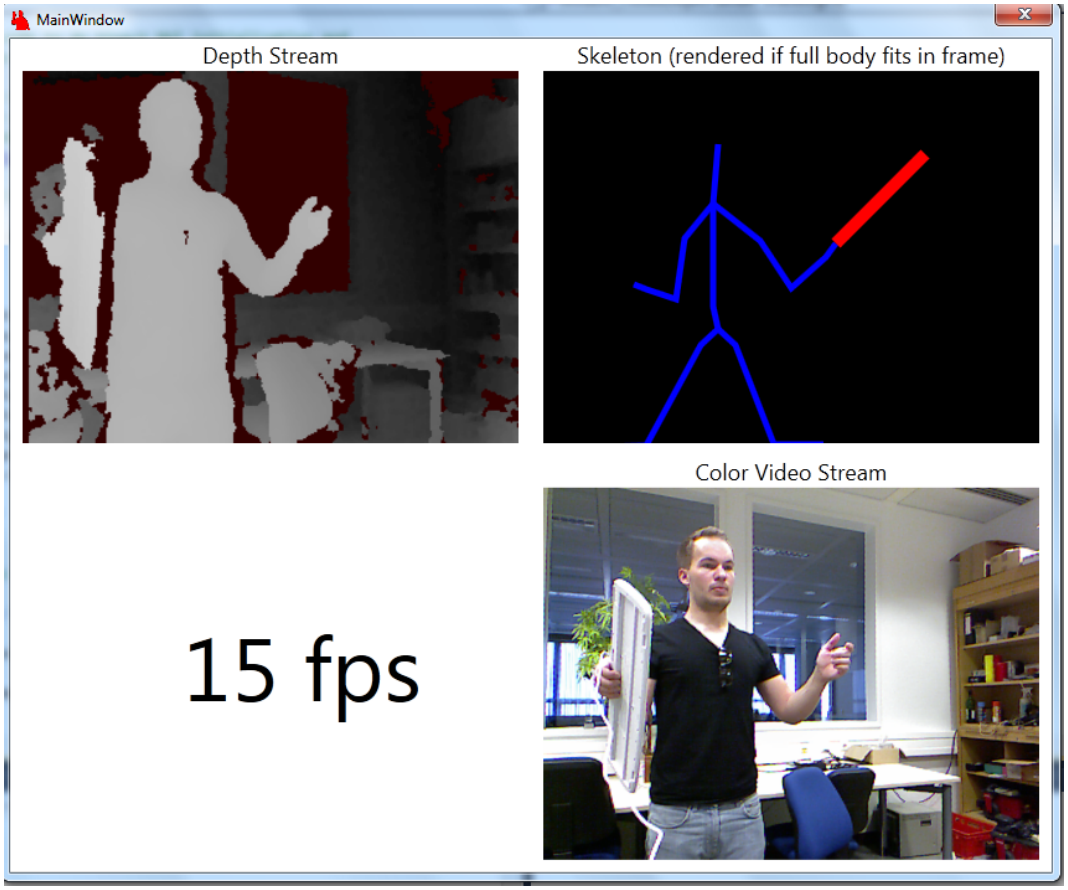

Figure 2.6: *Modifizierung der Skelettinformationen*

2.2.4 Ansprechen der Sensoren

Auszulesen sind die Kinect-Sensorinformationen nach dem Polling- oder Event-Model. Im Polling-Modus öffnet die zu programmierende Anwendung einen Stram zu dem Farb- oder Tiefensensor, bestimmt einen Timeout und fragt den Sensor über die NUI ab. Diese gibt, sobald die aufbereitete Information innerhalb der Timeout-Zeit zur Verfügung steht, diese zurück. Das Event-Model stößt, im Gegensatz zum Polling-Model, bei Fertigstellung der aufbereiteten Information ein Event an, welcher von einem Eventhandler abgefragt wird. Die im Projekt entstandene Software nutzt das recourcenschonende Event-Model.

2.3 Emotionserkennung

Hier kommt die vom Frauenhofer Institut entwickelte Engine names ”SHORE” - Sophisticated High-speed Object Recognition Engine- zur Verwendung. (vgl. IIS, 2011b) Diese stellt eine Bibliothek bereit, welche es ermöglicht in Farbbild-Informationen menschliche Gesichter und deren Emotionen zu erkennen. Eine beigefügre Demo-Software stellt dies eindrucksvoll da, zeigt jedoch auch Schwächen. So lässt sich diese leicht durch ein Bild eines Menschen täuschen oder verfehlt Alter und Geschlecht weit.

Figure 2.7: *Personen- und...*

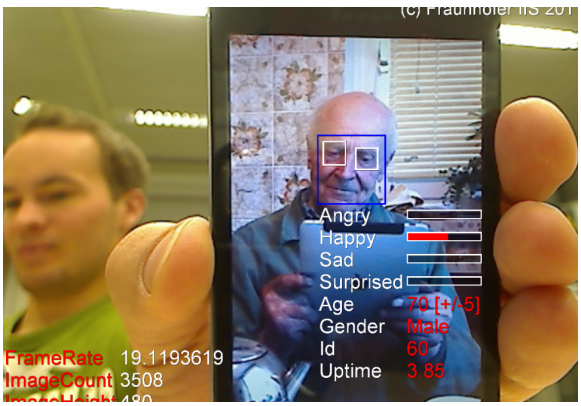

Figure 2.8: *... Emotionserkennung*

2.3.1 SHORE-Bibliothek

Zu Beginn habe ich mich mit den Möglichkeiten, welche mir die SHORE-Bibliothek zur Verfügung stellt, näher angesehen. Diese besteht im Grundlegenden aus folgenden fünf Klassen (vgl. IIS, 2011a, Seite 17) :

- **class Marker** A point in the image.
- **class Region** A region in the image.
- **class Object** An object within the image.
- **class Content** The content of an image.
- **class Engine**The workhorse of the library.

Der Hauptbestandteil der Bibliothek ist in der Engine-Klasse (vgl. IIS, 2011a, Seite 22) untergebracht und wird über eine Vielzahl von Attributen gesteuert. So ist es möglich verschiedene Erkennungsverfahren zu nutzen. In jeden Fall ist eine minimale Größe des Gesichts von 24x24 Pixel aus dem eingehenden Bildstrom vorausgesetzt. In einigen Fällen werden auch Auflösungen von 36x36 Pixeln benötigt Merkmale wie ”Position des linken Auges” auszuwerten.

Im ”face detection mode” ist die Engine (vgl. IIS, 2011a, Seite 23) darauf ausgelegt Gesichter zu erkennen. In Fall eines erkannten Gesichts gibt die Engine ein Objekt des Typs ”face” zurück, welche zu den XY-Grenzen des erkannten Gesichts und den Informationen wie Roll- und Neigewinkel auch die Wahrscheinlichkeit mit der sich das erkannte Gesicht wirklich um eines handelt - genannt ”score” - zurück. Zudem wird, ähnlich der Personenerkennung im Microsoft Kinect SDK, jedem erkannten Gesicht eine ID zugewiesen.

Dieser ”face detection mode” lässt sich durch Modelle, welche je nach Wunsch dem Modus hinzugefügt werden müssen, konfigurieren. Diese Modelle wurden ausführlich in der Dokumentation der Bibliothek beschieben (vgl. IIS, 2011a, Seite 23) . Im Folgenden einige Beispiele: Bei Hinzufügen der ’FaceFront_24x24_2008_08_29_161712_7.cpp’ erfolgt eine Suche eines frontal in die Kamera schauenden Gesichts. Zurückgegeben werden zudem die Position der Augen. Die ’FaceRotated_24x24_2008_10_15_-180432_24.cpp’-Datei erlaubt auch ein erkennen von Gesichtern, welche bis zu 60 Grad in positive oder negative Richtung zur Kamera gedreht stehen.

2.3.2 Auswertung erkannter Gesichter

Die durch den ”face detection mode” erkannten Gesichter, können nun durch die Bibliothek auf vorgegebene Kriterien durchsucht werden. Hier stehen folgende Optionen (vgl. IIS, 2011a, Seite 25) bereit:

- **searchEyes** Suche der Augen und Rückgabe der Position dieser.
- **searchNose** Suche der Nase und Rückgabe der Position dieser.
- **searchMouth** Suche des Mundes und Rückgabe der Position dieses.
- **analyzeEyes** Analysiert ob und welches Auge geschlossen ist und gibt eine Wahrscheinlichkeit von 0 bis 100 zurück
- **analyzeMouth** Analysiert ob der Mund geschlossen ist und gibt eine Wahrscheinlichkeit von 0 bis 100 zurück
- **analyzeGender** Analysiert das Geschlecht der erkannten Person und ist und gibt je Geschlecht eine Wahrscheinlichkeit von 0 bis 100 zurück
- **analyzeAge** Analysiert das Alter der erkannten Person und ist gibt einen Wert von 0 bis 90 zurück. Des Weiteren wird eine mögliche Abweichung ermittelt und als ’AgeDeviation’ zurückgegeben.

Zudem ist eine Analyse der Stimmung, jeweils in einem Wahrscheinlichkeitsbereich von 0 bis 100, der erkannten Person an folgenden Optionen ermittelbar:

- **analyzeHappy** Wie glücklich sieht die erkannte Person aus.
- **analyzeSad** Wie traurig sieht die erkannte Person aus.
- **analyzeSurprised** Wie überrascht sieht die erkannte Person aus.
- **analyzeAngry** Wie böse sieht die erkannte Person aus.

2.3.3 Fazit der Gesichtserkennung

Die Gesichts- und Emotionserkennung hat sich als überaus nutzbar erwiesen. Die durch die Bibliothek ermittelten Werte glichen zumeist der Realität. Die Erkennung arbeitet Robust, überaus schnell und wie die Microsoft Kinect Kamera-Auswertung in Echtzeit, was eine Verarbeitung der Daten zur Laufzeit ermöglicht.

3 Zusammenfassung

3.1 Zusammenfassung

Im Projekt 1 konnten erfolgreich die Grundlagen der Auswertung der unterschiedlichen Sensortypen geschaffen werden. Durch das noch rechtzeitige Erscheinen der Microsoft Programmierumgebung zur Kinect war es möglich alle gesteckten Ziele an dieses System zu erreichen. Es ist sowohl die auf 12 Bit genaue Verarbeitung der Tiefeninformationen, als auch eine Auswertung der Skelettmodelle, wie auf Seite 9 beschrieben, möglich gewesen. Gerade die Auswertung der Skelettmodelle mit der Möglichkeit, Punkte dieses Models im 3D-Raum zu ermitteln, stellte sich als besonders vielversprechend heraus. Auch durch die vorsorgliche Einarbeitung in die Programmiersprache C# war es möglich, rasch Erfolge zu erzielen.

Das vom Frauenhofer Institut veröffentliche Werkzeug zur Emotonserkennung konnte ebenfalls getestet werden. Lediglich die Implementation der zur Verfügung gestellten Bibliotheken, in eine mit der Kinect kombinierten Sensorverarbeitung, konnte aus zeitlichen Gründen im Projekt 1 noch nicht fertig gestellt werden.

3.2 Ausblick

Diese Grundlage wird es im Projekt 2 erlauben, die vorgestellten Techniken in einer Software zu kombinieren und eine sinnvolle Sensorauswertung zu ermöglichen. So wäre es eine mögliche Zielsetzung, eine Software zur Simulierung eines auf Seite 4 beschriebenen Szenarien zu entwickeln.

Bibliography

[IIS 2011a] IIS, Fraunhofer: *Fraunhofer IIS - Shoore 1.4*. Fraunhofer IIS, 2011

[IIS 2011b] IIS, Fraunhofer: *Fraunhofer IIS Website*. 2011. – URL http://www.iis.fraunhofer.de/bf/bv/ks/gpe/. – Zugriffsdatum: 02.07.2011

[Microsoft 2011a] MICROSOFT: *Microsoft Kinect SDK Programming Guide*. 2011. – URL http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/docs/ProgrammingGuide_KinectSDK.pdf. – Zugriffsdatum: 02.07.2011

[Microsoft 2011b] MICROSOFT: *Microsoft Kinect SDK Website*. 2011. – URL http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/. – Zugriffsdatum: 02.07.2011

[OpenNI 2011] OPENNI: *OpenNI*. 2011. – URL http://www.openni.org/. – Zugriffsdatum: 02.07.2011

[Rahimi 2010] RAHIMI, Voigt: *HAW Living Place*. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-proj/rahimi-vogt.pdf. – Zugriffsdatum: 02.07.2011