Face-to-Face Communication

Erkennung von Gesichtsparametern mittels der Kamera eines mobilen Endgerätes und Übertragung dieser auf einen virtuellen Avatar zur Simulation der Mimik auf dem Display des Gesprächspartners

Diplomarbeit

vorgelegt von
Patrick Gentzcke

UNIVERSITÄT KOBLENZ · LANDAU
Institut für Computervisualistik
Arbeitsgruppe Computergrafik

SIEMENS
Siemens AG
Bereich Com
Geschäftsgebiet Mobile Devices

Betreuer: Dipl.-Ing. (BA) Technische Informatik Ansgar Tröster, Siemens AG
Prüfer: Prof. Dr. Stefan Müller, Universität Koblenz-Landau

November 2004

Zusammenfassung

In weiten Bereichen unseres täglichen Lebens findet zunehmend Kommunikation mittels mobiler Plattformen statt. Dabei gibt es sehr viele Anwendungen, die auf Bildverarbeitung und Computergrafik basieren. Dazu gehören auch virtuelle Welten, in denen man auf elektronisch generierte Figuren, sogenannte Avatare trifft. Das Aufeinandertreffen von Menschen und digitalen, computergesteuerten Akteuren eröffnet dem Kunden die Möglichkeiten, die Kommunikation spielerisch und neu zu erleben. Waren die Benutzer in der Vergangenheit noch auf ihre Vorstellungskraft angewiesen, um aus rein textbasierter Kommunikation, das Verhalten und Aussehen ihres Gesprächspartners zu visualisieren, so wird dies durch die Integration von Computergrafik und digitalen Animationen in naher Zukunft auf einem Mobiltelefon möglich. Der Benutzer sieht die Welt nun aus den Augen seiner digitalen Repräsentation und kann mit den sichtbaren Objekten der virtuellen Welt interagieren und auf mehreren Ebenen kommunizieren.
Inhalt dieser Arbeit ist die Auseinandersetzung mit den verschiedenen Verfahren zur Gesichtslokalisierung, Extraktion von Gesichtsmerkmalen und der Avatar-Animation. Es wird evaluiert, welche Verfahren zur Gesichtslokalisierung am besten geeignet sind. Dazu wurde eine Marktübersicht bezüglich Software zur Gesichtserkennung und Avatar-Animation erstellt um daraus eine optimale Lösungsidee für diese Arbeit abzuleiten. Der praktische Schwerpunkt liegt auf der prototypischen Erstellung einer Java 2 Micro Edition (J2ME) Anwendung, die mittels der Kamera eines Mobiltelefons die Mimik einer Person erkennt und anschließend auf einen Avatar überträgt. Dadurch ist die Simulation einer Face-to-Face Kommunikation über Netze mit geringen Kapazitäten möglich. Weiterhin wird ein Modell vorgestellt, das beschreibt, wie und unter welchen Voraussetzungen diese Anwendung bei einem Netzbetreiber in Betrieb gehen könnte.

Erklärung:

Hiermit versichere ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

München, den 3. November 2004

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Ziele der Arbeit . . . 2
- 1.3 Aufbau der Arbeit . . . 3

2 Grundlagen **4**
- 2.1 Grundlagen der Bildverarbeitung . . . 4
  - 2.1.1 Farbmodelle . . . 4
  - 2.1.2 Segmentierung . . . 5
  - 2.1.3 Gradientenfilter . . . 6
  - 2.1.4 Morphologische Operationen . . . 8
  - 2.1.5 Hough-Transformation . . . 10
- 2.2 Grundlagen der Computergrafik . . . 11
  - 2.2.1 Mesh . . . 11
  - 2.2.2 Avatare . . . 12

3 Marktübersicht **13**
- 3.1 SeeStorm - Face Analysis SDK . . . 13
- 3.2 SeeStorm - Mobile . . . 13
- 3.3 Charamel - Mocito . . . 14
- 3.4 Fraunhofer Institut für Nachrichtentechnik - 3D-Videokonferenz . . . 14
- 3.5 XeneX-Group - XG-Emotion . . . 15
- 3.6 Neven Vision - Facial Analysis SDK . . . 15
- 3.7 Zusammenfassung . . . 16

4 Symbian vs. J2ME **19**
- 4.1 Symbian . . . 19
- 4.2 J2ME . . . 20
- 4.3 Vergleich von Symbian und J2ME CLDC 1.0 . . . 21

5 Gesichtserkennung **23**
5.1 Verfahren zur Gesichtslokalisierung 23
5.1.1 Ellipse Fit 23
5.1.2 Deformable Line Templates 25
5.1.3 Augen-Mund Abstand 25
5.1.4 Farbklassifikation 27
5.2 Extraktion von Gesichtsmerkmalen 28
5.2.1 Finden der Iris 28
5.2.2 Finden des Mundes 30
5.3 Face Finding und Feature Extraction 32
5.4 Face Tracking 34
5.5 Implementierung 35
5.6 Ergebnisse 37

6 Avatar Animation **39**
6.1 Facial Animation in MPEG-4 39
6.2 Gesichtsmodell 40
6.3 Face Animation 41
6.4 Implementierung 44
6.5 Ergebnisse 45

7 Modell einer Face-to-Face Applikation **47**
7.1 Applikation 47
7.2 Technische Vorraussetzung 47
7.3 Use-Cases 49
7.3.1 Use-Case: Empfänger ist zum Server verbunden 49
7.3.2 Use-Case: Empfänger ist nicht zum Server verbunden . . 49
7.4 Fazit 50

8 Fazit **51**
8.1 Zusammenfassung 51
8.2 Bewertung 52
8.3 Ausblick 53

A Workflow der Gesamtapplikation **55**

B Siemens Mobility Toolkit 2.00b **57**

C Face Animation Player **59**

Abbildungsverzeichnis

2.1 RGB-Raum (vgl. [35], RGB-Farbraum) 5
2.2 HSV-Raum (vgl. [35], HSV-Farbraum) 6
2.3 Segmentierung mit Schwellwert 7
2.4 Gradientenfilter 7
2.5 Sobel-Operator 8
2.6 Morphologische Masken 9
2.7 Erosion 9
2.8 Dilatation 10
2.9 Mesh 11
2.10 Wobble (vgl. [41]) und Robert T-Online (vgl. [40]) 12

3.1 SeeStorm - Face Analysis (vgl. [1]). 14
3.2 Peter Eisert - 3D-Videokonferenz (vgl. [9]) 15
3.3 Neven Vision - Facial Feature Tracking (vgl. [12]) 16

5.1 Parametrisierung einer Ellipse 24
5.2 Ellipse Fit 24
5.3 Feature Erkennung durch Augen-Mund Abstand (vgl. [17]) 26
5.4 Farbklassifikation im HSV-Raum 27
5.5 Bounding-Boxes für Iris-Erkennung 28
5.6 Iriserkennung 29
5.7 Lippenerkennung 31
5.8 Berechung der Augen-BB durch das Tracking 35
5.9 Berechung der Mund-BB durch das Tracking 35

6.1 Drahtgittermodell des Kopfes (vgl. [28]) 41
6.2 Texturiertes Kopfmodell (vgl. [28]) 42
6.3 Auszug aus den Face Animation Parameter 43
6.4 Auszug aus der Face Definition Table 44

A.1 Workflow der gesamten Face-to-Face Applikation 56

B.1 Siemens Mobilty Toolkit - Manager 57
B.2 Siemens Mobilty Toolkit - Emulator Launcher 58

C.1 Face Animation Player - Process und Load Face Model 59

Tabellenverzeichnis

3.1 Marktübersicht Vergleich 17

4.1 Vergleich von Symbian- mit J2ME-Anwendungen 21

5.1 Face Finding 33
5.2 Feature Extraction 34
5.3 Face Tracking 35
5.4 Vergleich von Verfahren zur Gesichtslokalisierung 36
5.5 Abweichung der Gesichtslokalisierung 38
5.6 Abweichung der Augenparameter 38
5.7 Abweichung der Mundparameter 38

6.1 Normalisierte Gesichtsknotentabelle 40
6.2 Triangulierung des Meshes 42
6.3 Face Animation 43

Kapitel 1

Einleitung

1.1 Motivation

Virtuelle Welten erlauben ihren Benutzern unbekannte, neue Gebiete in Gestalt eines digitalen Avatars zu erschließen. Das Aufeinandertreffen von Menschen und digitalen, computergesteuerten Akteuren öffnet die Türen zu neuen Möglichkeiten und Erfahrungen. Waren die Benutzer in der Vergangenheit noch auf ihre Vorstellungskraft angewiesen, um aus rein textbasierter Kommunikation das Verhalten und Aussehen ihres Gegenübers zu visualisieren, so wird dies durch die Integration von Computergraphik und digitalen Animationen bald auf handelsüblichen Mobiltelefonen möglich. Der Benutzer sieht die Welt nun aus den Augen seiner digitalen Repräsentation und kann mit den sichtbaren Objekten der virtuellen Welt interagieren und auf mehreren Ebenen kommunizieren.

Die Diplomarbeit beschäftigt sich mit der Animation eines digitalen Avatars zur realistischen Mimikrepräsentation eines Gesprächspartners. Zusammen mit den Erkenntnissen aus der Bildverarbeitung, der Gesichtsanimation und einer GPRS-Verbindung (General Packet Radio Service) soll ein System erstellt werden, welches speziell auf die Repräsentation eines Gesprächspartners durch einen virtuellen Repräsentanten ausgerichtet ist. Dazu muss zuerst das Gesicht und die damit verbundenen Eigenschaften erkannt werden. Im nächsten Schritt werden spezifische Parameter des Gesichtes extrahiert und auf einen Avatar übertragen. Dadurch kann eine Face-to-Face Unterhaltung simuliert werden.

Eine weitere Motivation in dem von starkem Wettbewerb geprägten Mobilfunkmarkt ist das Bestreben, mit innovativen Diensten auch neue Geschäftsmodelle anzubieten. Darüber hinaus können Virtual Reality Anwendungen helfen, die begrenzten Netzressourcen optimal zu nutzen, da hierauf basierende audiovisu-

elle Konferenzen mit erheblich geringerem Datendurchssatz angeboten werden können (vgl. Kapitel 3.4 - statt 128kbps nur 1kbps).

Die Netzbetreiber sind bemüht, durch die Attraktivität der Angebote und Geräte den Zugang zu ihren Diensten überall und einfach zu ermöglichen. Dazu muss die Leistungsfähigkeit der mobilen Endgeräte stetig erweitert werden. Für die Endgerätehersteller ist diese Anforderung an eine überragende Innovationskraft nur erreichbar, wenn Anwendungen rechtzeitig evaluiert und dadurch Prioritäten für die Weiterentwicklung der Plattformen entsprechend gesetzt werden können.

Dies gilt gleichermaßen für die Anbieter der Infrastruktur. Hier ist insbesondere Siemens Com verpflichtet und aufgestellt, dem Netzbetreiber Infrastruktur und passende Endgeräte aus einer Hand anzubieten (vgl. [7]).

1.2 Ziele der Arbeit

Ziel der Arbeit ist , die momentanen technischen Möglichkeiten und Grenzen der mobilen Java-Plattform anhand einer Beispielapplikation zu untersuchen. Dabei spielen einerseits die verwendete Java Virtual Machine (JVM) sowie andererseits die Hardware eine Rolle. Des Weiteren soll ein Prototyp einer Applikation zur Face-to-Face Kommunikation erstellt werden. Die Applikation basiert auf einem mobilen Endgerät mit integrierter Kamera. Die Bildverarbeitungsoperationen ermitteln die Mimik eines Teilnehmers. Anschließend wird die Mimik auf einen Avatar übertragen und entsprechend des Verhaltens des anderen Teilnehmers animiert.
Letztlich müssen noch die Rahmenbedingungen untersucht werden, unter denen solch eine Applikation erfolgreich am Markt eingesetzt werden kann. Hierzu wird zuerst eine Marktübersicht über bereits bestehende, ähnliche Verfahren erstellt. Daraus wird die Anforderungsliste für den im Rahmen dieser Arbeit entwickelten Software-Prototypen abgeleitet.
Während der Entwicklung der Java 2 Mobile Edition-Anwendung (J2ME) werden die verschiedenen Verfahren zur Gesichtslokalisierung evaluiert. Anschließend wird ein geeignetes Verfahren zur Gesichtslokalisierung, Augenerkennung und der Munderkennung implementiert. Darüber hinaus wird ein Algorithmus zur Platzierung eines Drahtgitternetzes (Mesh) auf den relevanten Gesichtsdaten entwickelt, um die Animation eines Avatars zu ermöglichen.
In dieser Arbeit muss ein besonderer Wert auf die Performance der Algorithmen gelegt werden, da die Leistungsfähigkeit der mobilen Java-Engine aufgrund des Kostendrucks auf die Komponenten limitiert ist (vgl. Kapitel 4.2). Daher ist es oft nicht möglich, bereits existierende Verfahren zu verwenden.

1.3 Aufbau der Arbeit

Die Arbeit ist in acht Kapitel gegliedert. Nach dem ersten einleitenden Kapitel werden im zweiten Kapitel die Grundlagen bezüglich Bildverarbeitung und Computergrafik vermittelt. Hier wird ein Überblick über die unterschiedlichen Farbräume gegeben und es werden grundlegende Bildverarbeitungsoperationen und deren Funktionsweise erklärt. Dies ist notwendig um die Techniken und Methoden in den weiteren Kapiteln dieser Diplomarbeit zu verstehen.
Im dritten Kapitel wird eine Marktübersicht, über bereits existierende, ähnliche Produkte gegeben. Dabei wird auf die verschiedenen Plattformen und Funktionsweisen sowie die Features der Anwendungen eingegangen.
Das vierte Kapitel beschäftigt sich mit dem Vergleich der mobilen Java Plattform und der Symbian Plattform. Zuerst werden die Vor- und Nachteile der jeweiligen Systeme erläutert.
Das fünfte Kapitel behandelt die Erkennung der Mimik mittels Bildverarbeitungsoperationen. Zuerst werden die verschiedenen Verfahren zur Gesichtlokalisierung evaluiert, danach werden Face Finding, Feature Extraction und Face Tracking im Detail vorgestellt. Im sechsten Kapitel wird die Avatar-Animation behandelt. Das vorletzten Kapitel zeigt ein mögliches Modell zur Face-to-Face Kommunikation.
Abschließend werden im achten Kapitel die Ergebnisse dieser Arbeit zusammengefasst und es wird ein Ausblick auf technische Weiterentwicklungen sowie Veränderungen des zukünftigen Mobilfunktmarktes gegeben.

Zur Terminologie: Bei der Beschreibung der verschiedenen Inhalte stellte sich immer wieder die Frage, wie die korrekten Übersetzungen der englischen Originalbegriffe lauten. Einige Begriffe können dabei eindeutig übersetzt werden, während dies bei anderen kaum sinnvoll ist. Aus diesem Grund werden neben den deutschen auch die englischen Bezeichnungen angegeben. Der Einfachheit halber wird in dieser Arbeit nur von Anwendern, Benutzern usw. gesprochen, obwohl selbstverständlich auch immer die Anwenderinnen und Benutzerinnen damit gemeint sind.

Kapitel 2

Grundlagen

In diesem Kapitel werden die wichtigsten Grundlagen bezüglich Bildverarbeitung, Computergrafik und Avataren erklärt. Dies dient der besseren Verständlichkeit der restlichen Kapitel.

2.1 Grundlagen der Bildverarbeitung

2.1.1 Farbmodelle

RGB - Farbmodell

Innerhalb des RGB-Farbraums wird eine Farbe als additive Mischung der Primärfarben Rot, Grün und Blau gesehen (vgl. [2] S.170). Dies entspricht auch der Anzeigetechnik von Monitoren oder Handydisplays.
Die Abbildung 2.1 zeigt das Würfel-Modell des RGB-Farbraums. Auf der Diagonalen des Würfels liegen die Farbtöne mit gleichen Rot-, Grün- und Blau-Anteilen (Grauwerte). Jede der drei Primärfarben wird mit 256 Werten dargestellt. Daher erhält man im RGB-Farbraum 16,7 Mio. verschiedene Farbtöne.

HSV - Farbmodell

Der HSV-Farbraum wird durch die Attribute Hue (Farbton), Saturation (Sättigung) und Value (Helligkeit) beschrieben. Hue ist der Farbwinkel, der im Bereich von 0° bis 360° liegt (vgl. [4]). Das HSV-Modell ermöglicht es somit sehr leicht Farbtöne zu extrahieren, da diese von der Helligkeit und Sättigung unabhängig sind.
Der Farbwinkel V des HSV-Farbraums ist an der V-Achse (R=G=B) undefiniert. Zusätzlich ist der Farbwinkel in der Nähe der V-Achse sehr instabil (vgl. [3]).

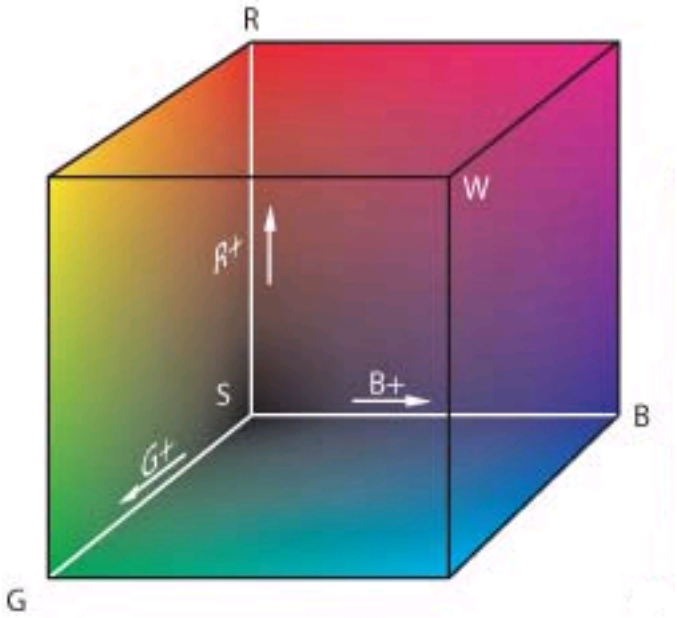

Abbildung 2.1: RGB-Raum (vgl. [35], RGB-Farbraum)

2.1.2 Segmentierung

Das Ziel der Segmentierung eines Bildes ist die Zerlegung in unterschiedliche Regionen, die in ihren Eigenschaften gleich sind. Somit können aus einem Bild einzelne, relevante Bereiche bestimmt werden, die für die weitere Verarbeitung von Bedeutung sind. Als Resultat verwirft man frühzeitig uninteressante Teile des Bildes, und für spätere Operationen wird weniger Performance benötigt.
Die Segmentierung ist innerhalb der Bildverarbeitung eine der wichtigsten Schritte, da hier einzelne Objekte aus dem Bild herausgelöst und identifiziert werden können.

Bei der punktorientierten Segmentierung ist die Entscheidung, ob ein Pixel dem Objekt angehört oder nicht, abhängig von der Eigenschaft des Bildpunktes. Ein Farbwinkel oder der Sättigungsgrad sind Beispiele für ein solches Entscheidungskriterium (vgl. [21]). Die Gleichung 2.1 zeigt die Regel für die Binärisierung eines Bildes. Dabei ist A_T das Pixel an der Stelle x, y im Bild. Anhand des Schwellwertes T wird die Binärisierung B durchgeführt.

$$A_T(x,y) = \begin{cases} 1 : B(x,y) > T \\ 0 : B(x,y) \leq T \end{cases} \tag{2.1}$$

Dunkle Objekte auf hellem Hintergrund oder helle Objekte auf dunklem Hinter-

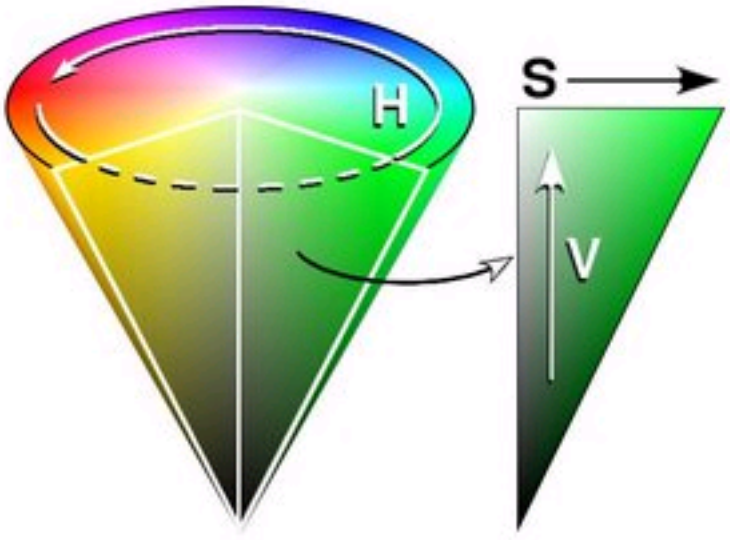

Abbildung 2.2: HSV-Raum (vgl. [35], HSV-Farbraum)

grund können mit Schwellwertverfahren einfach segmentiert werden (vgl. Abb. 2.3). Die Qualität der Ergebnisse hängt entscheidend vom Schwellwert T ab. In der Regel wendet man vor dem Schwellwertverfahren einen Filter an, um Rauschen und Störpixel zu entfernen. Hierzu eignen sich besonders nichtlineare Filter (z.B. Median), da diese die Kanten erhalten.

2.1.3 Gradientenfilter

Objektkanten und Linien sind wichtige Merkmale für die Bewertung von Bildern. Als Kante bezeichnet man die Grenze zwischen zwei homogenen Flächen in einem Bild. Es gibt Grauwert-, Textur- und Farbkanten. Im Nachfolgenden wird mit Grauwert- und Farbkanten gearbeitet.
Gradientenfilter beruhen auf der Approximation der 1. partiellen Ableitung der Bildfunktion (vgl. [4]). Die Matrix 2.2 zeigt die Maske für die Detektion von Kanten in x-Richtung. Dieser Filter wird auch Prewitt-Operator genannt. Der Prewitt-Operator enthält sowohl eine Glättung durch Mittelwertbildung als auch eine Glättung der symmetrischen Gradienten. Die Abbildung 2.4 zeigt die Wirkung eines Gradientenfilters in x-Richtung.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.2}$$

Abbildung 2.3: Segmentierung mit Schwellwert

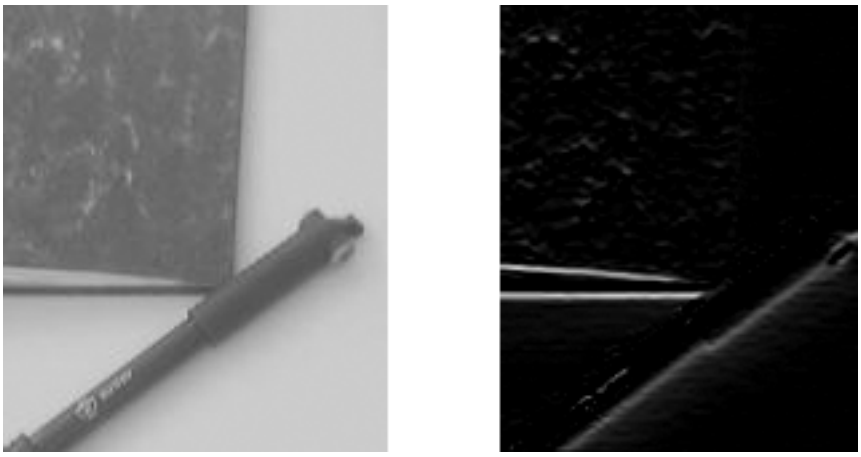

Abbildung 2.4: Gradientenfilter

Sobel-Operator

Der Sobel-Operator (Abb. 2.5) ist ein einfacher Kanten-Detektions Algorithmus der Bildverarbeitung (vgl. auch im Weiteren [2] S.103). Hier wird die erste Ableitung der Bildpunkt-Helligkeitswerte mit einer gleichzeitigen Glättung berechnet. Der Operator nutzt zur Faltung eine 3x3 Matrix (vgl. Matrix 2.3), die aus dem Originalbild ein Gradienten-Bild erzeugt. Die Bereiche der größten Inten-

sität sind dort, wo sich die Helligkeit des Originalbildes am stärksten ändert. An diesen Stellen liegen wahrscheinlich auch Kanten. Die Abbildung 2.5 zeigt deutlich eine Hervorhebung von horizontalen Kanten, durch die Verwendung eines horizontalen Sobel-Operators (Abb. 2.3).

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.3}$$

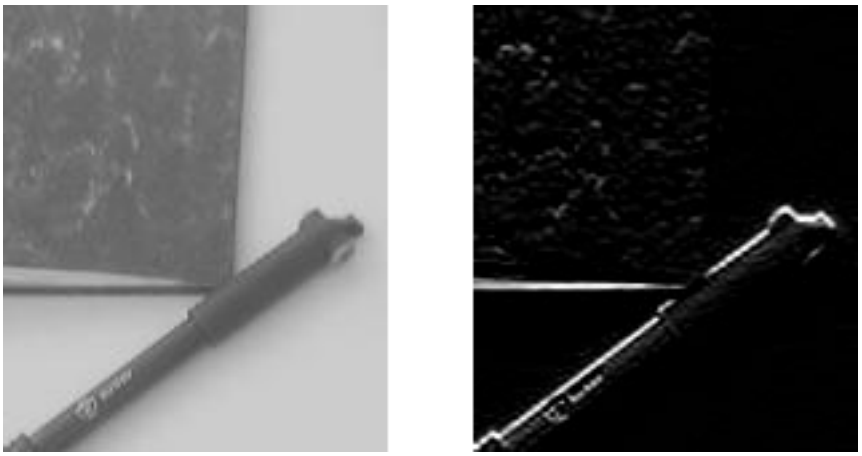

Abbildung 2.5: Sobel-Operator

2.1.4 Morphologische Operationen

Morphologische Filter nutzen Informationen bezüglich der Gestalt von Objekten zur Beseitigung von Störungen. Die Wirkungsweise lässt sich durch die Form der Maske steuern (vgl. [2] S.107). Die Abbildung 2.6 zeigt typische nichtquadratische Masken. Die morphologischen Filter unterteilen sich in Erosion und Dilatation.

Erosion

Bei der Bildverarbeitung bezeichnet die Erosion das selektive Löschen von Bildpunkten, indem jeder Bildpunkt eines dargestellten Objekts mit einer verschiebbaren Maske verglichen wird (vgl. auch im Weiteren [21]). Es bleiben nur die Bildteile des Objekts erhalten, die vollständig von der Maske überdeckt werden.

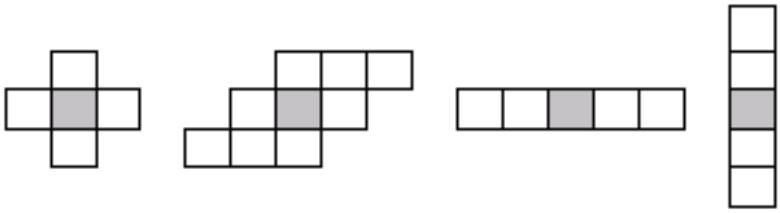

Abbildung 2.6: Morphologische Masken

Die Erosion liefert also diejenige Menge von Bildpunkten, die gleich dem Durchschnitt von Bildelementen und Maskenelementen ist (vgl. Abbildung 2.7). Mit dem Erosionsoperator lassen sich beispielsweise horizontale Linien mittels einer geeigneten Maske extrahieren. Nur wenn sich das Maskenelement B vollständig im Objekt A befindet, wird der Bezugspunkt „UND“-verknüpft und damit auf „1“ gesetzt (vgl. Gleichung 2.4). C stellt das Ergebnis der Erosion dar.

$$C = A \ominus B \tag{2.4}$$

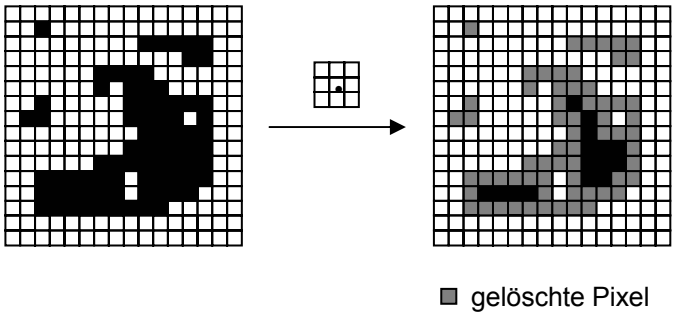

Abbildung 2.7: Erosion

Dilatation

Die Dilatation ist der duale Operator der Erosion (vgl. [25]). Die Menge der Dilatation sind alle Punkte, die die Maske berühren. Innerhalb der Dilatation (Gleichung 2.5) werden Bildpunkte am Rande des Objektes hinzugefügt, die von der Form und Größe der Maske abhängig sind. Das Ergebnis einer Dilatation ist in Abbildung 2.8 zu sehen. Wenn das Maskenelement B mindestens ein Pixel im

Objekt A abdeckt, wird eine „UND“-Verknüpfung durchgeführt und der Bezugspunkt erhält den Wert „1“ (vgl. Gleichung 2.5). C stellt das Ergebnis der Dilatation dar.

$$C = A \oplus B \tag{2.5}$$

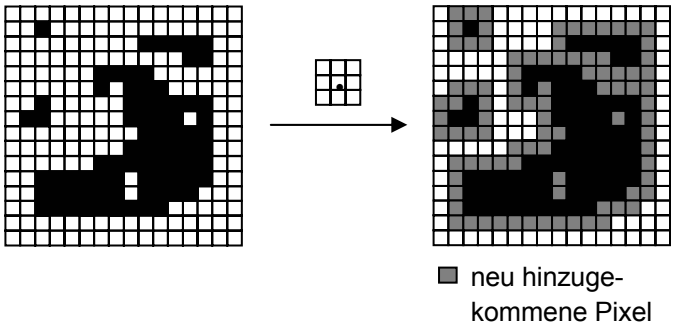

Abbildung 2.8: Dilatation

Closing

Die Kombination von Erosion und Dilatation werden abhängig von der Reihenfolge Opening beziehungsweise Closing genannt. Die Gleichung 2.6 zeigt, dass beim Closing zuerst eine Dilatation und anschließend eine Erosion durchgeführt wird (vgl. [21]). Dies führt zum Schließen von kleinen Lücken an den Objekträndern.

$$C = (A \oplus B) \ominus B \tag{2.6}$$

Opening

Bei der Opening-Operation wird zunächst eine Erosion mit nachfolgender Dilatation durchgeführt (vgl. Gleichung 2.7). Dabei können ausgefranste Objektränder begradigt werden und Pixelstörungen oder vereinzelte Linien werden komplett unterdrückt.

$$C = (A \ominus B) \oplus B \tag{2.7}$$

2.1.5 Hough-Transformation

Mittels Hough-Transformation können parametrisierbare Kurven in einem Bild sehr leicht gefunden werden (vgl. auch im Weiteren [6] S.2). Die Hough-Transformation bildet alle Punkte von Randkurven in einem Punkt des Abbildungsraumes ab. In diesem Falle ist die Randkurve eine Gerade, die durch die

Gleichung 2.8 bestimmt wird (vgl. [19]). Dabei gehören x und y zur Menge der reellen Zahlen. Die Steigung m sowie die Konstanten b beschreiben die Kurve.

$$y = m * x + b \tag{2.8}$$

Es werden die Farbwerte aller Pixel, die auf einer Geraden liegen addiert und in einer zweidimensionalen Matrix, dem accumulator array, abgelegt. Jeder Wert an der Stelle m_0, b_0 innerhalb des Abbildungsraumes beinhaltet also die Summe der Grauwerte aller Bildpunkte, die im Ausgangsbild auf der Geraden $y = m_0 * x + b_0$ liegen. Beispielsweise stellt der Punkt (7, 3) im Akkumulator Array die aufsummierten Bildpunkte der Geraden $y = 7x + 3$ dar.

2.2 Grundlagen der Computergrafik

2.2.1 Mesh

Ein Mesh/Netz besteht aus mehren Punkten, die durch Kanten miteinander verbunden sind (vgl. [20]). Abhängig von der Definition des Meshes wird der Pfad vorgegeben, nach dem das Mesh durchlaufen werden kann. Die Abbildung 2.9 zeigt ein einfaches Mesh. Die Zahlen bestimmen den Knoten, um später eine Triangulierung (Verbinden von drei Punkten) zu ermöglichen. In dieser Arbeit sind die verschiedenen Arten der Triangulierung nicht relevant, da die Punkte und die Reihenfolge fest vorgegeben sind.

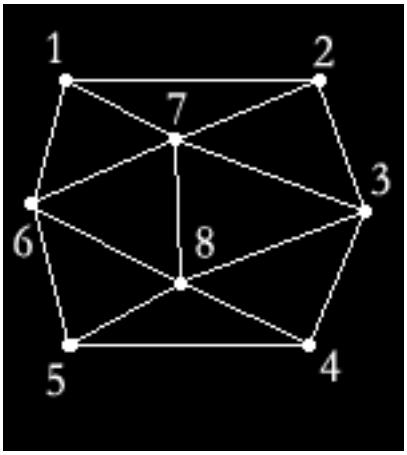

Abbildung 2.9: Mesh

2.2.2 Avatare

Es gibt verschiedene Definitionen von Avataren. In dieser Arbeit wird bei dem Begriff Avatar von einer elektronisch generierten Figur, die Personen illustriert ausgegangen.
Im ursprünglichen Sinne ist ein Avatar die Verkörperung einer Gottheit auf Erden, im engeren Sinne meist bezogen auf eine der 10 Inkarnationen des Gottes Vishnu (vgl. [35],Avatar).
Zu den bekannten Avataren gehören Wobble (vgl. [41]) (siehe Abb. 2.10 links) sowie Robert T-Online (vgl. [40])(siehe Abb. 2.10 rechts). Durch diese elektronisch generierten Figuren soll dem Benutzer die Interaktion vereinfacht werden. Allgemein fühlen sich Personen in Umgebungen mit Avataren persönlich mehr angesprochen als in Umgebungen ohne Avatare oder persönlicher Betreuung (vgl. [8]).

Abbildung 2.10: Wobble (vgl. [41]) und Robert T-Online (vgl. [40])

Kapitel 3

Marktübersicht

Zum Thema Gesichtserkennung und Animation von Avataren gibt es bereits einige kommerzielle Lösungen. In einem ersten Schritt werden die verschiedenen Hersteller und die Eigenschaften der Software kurz vorgestellt und verglichen. Der zweite Schritt beinhaltet die Ableitung einer Anforderungsliste für diese Diplomarbeit anhand der bestehenden Lösungen.

3.1 SeeStorm - Face Analysis SDK

Das SeeStorm Face Analysis Software-Development-Kit (SDK) (vgl. auch im Weiteren [1]) ermöglicht eine Gesichtserkennung auf frontalen Fotos. Die Erkennung ist weder in Echtzeit noch auf einem mobilen Endgerät möglich. Hier wird mindestens ein Pentium-III PC mit 128MB RAM benötigt. Das Eingabebild muss eine Größe von mindestens 60x80 Pixel haben. Wie Abbildung 3.1 zeigt, beinhaltet das Ergebnis der Gesichtserkennung Bounding-Boxes für den Kopf und die Augen. Des Weiteren werden Gesichtsmerkmale wie Augenbrauen, Nase, Augen, Mund und Pupillen durch Spline-Kurven dargestellt.
Der bei diesem Verfahren zu Grunde liegende Algorithmus basiert auf künstlichen neuronalen Netzen und wurde in C++ implementiert. Die Ergebnisse der Merkmalsextraktion werden in einem Feature-Vektor gespeichert.

3.2 SeeStorm - Mobile

Mit SeeStorm Mobile können Avatare automatisch durch Sprache und Text animiert und auf einem mobilen Endgerät abgespielt werden (vgl. auch im Weiteren [1]). SeeStorm Mobile ist eine Client-Server Anwendung. Bei dieser muss der Nutzer einen Text und ein Foto an den Server übertragen. Dabei wird das Gesicht aus dem Foto extrahiert und ein Avatar wird erzeugt. Der generierte Avatar ist

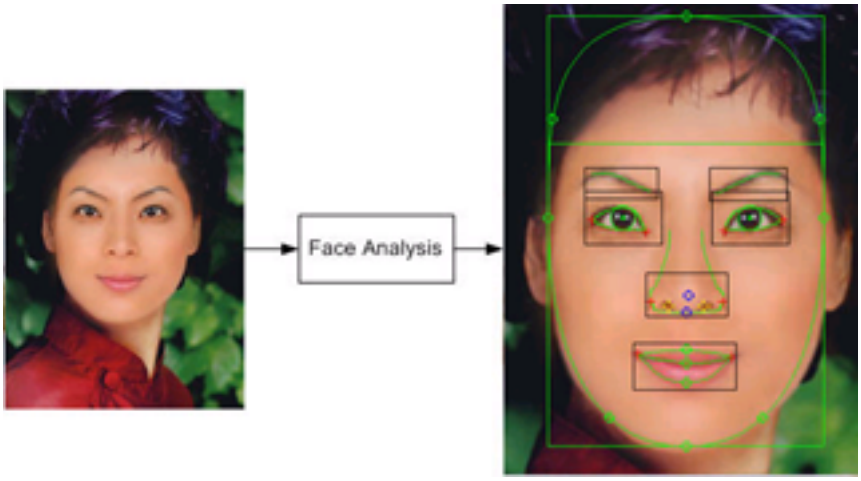

Abbildung 3.1: SeeStorm - Face Analysis (vgl. [1]).

innerhalb einer Multimedia Messaging Service-Nachricht (MMS) zu sehen Dabei wird der textuelle Inhalt durch den Avatar vorgelesen (Animation des Fotos durch Morphing). Die Animation des Avatars findet jedoch auf einem zwischengeschalteten Server statt. Das Ergebnis, die MMS-Nachricht ist eine Video-Datei die auf dem mobilen Endgerät abgespielt wird.

3.3 Charamel Mocito

Charamel Mocito (vgl. auch im Weiteren [8]) ist eine Software, die es ermöglicht, virtuelle Charaktere mittels Texteingabe zu animieren. Zuerst wird der Charaktertyp ausgewählt. Im nächsten Schritt wird durch die Auswahl von Emotionen die Gefühlslage des Avatars bestimmt. Diese Daten werden an einen Server übermittelt, der den Charakter entsprechend der Texteingabe und der gewählten Emotion animiert. Zusätzlich wird der Text in Sprache umgewandelt. Schließlich wird die Video-Datei als MMS-Nachricht oder als E-Mail an den Empfänger versendet.

3.4 Fraunhofer Institut für Nachrichtentechnik - 3D-Videokonferenz

Am Fraunhofer Institut für Nachrichtentechnik wurde ein Videokonferenzsystem entwickelt, das basierend auf dem MPEG-4 Standard die Übertragung von Schulterszenarien mit niedriger Bandbreite ermöglicht (vgl. auch im Weiteren [9]). Ein

Schulterszenario ist ein Bildausschnitt, bei dem der Kopf vollständig zu sehen ist und der Rest des Körpers knapp unterhalb der Schulter abgeschnitten wird.
Die 3D-Objektbeschreibungen werden einmal codiert und übermittelt. Ab diesem Zeitpunkt müssen nur noch die Parameter, die die Veränderungen der Objekte beinhalten, übertragen werden. Der Decoder auf dem mobilen Endgerät rekonstruiert die Videosequenz und animiert das 3D-Kopfmodell, um unterschiedliche Gesichtsausdrücke darzustellen. Hierbei werden Bitraten von weniger als 1kbps erreicht.
Die Abbildung 3.2 zeigt ganz links das original Ausgangsbild. In der Mitte ist

Abbildung 3.2: Peter Eisert - 3D-Videokonferenz (vgl. [9])

das künstliche Drahtmodell zu sehen. Rechts wurde das Ausgangsbild decodiert und in einem künstlichen Modell neu generiert.

3.5 XeneX-Group - XG-Emotion

Innerhalb des Projektes XG-Emotion der XeneX-Group (vgl. auch im Weiteren [11] S.8) wird die Emotionsvisualisierung an einem virtuellen Gesicht mittels des MPEG-4 Standards untersucht und beispielhaft in Java/Java-3D umgesetzt. Der Schwerpunkt des XG-Emotion Projektes liegt in der Verwendung von Facial-Animation-Parameters (FAP) zur Darstellung von errechneten Emotionen (nach MPEG-4 Standard) und der Bewegung des Gesichtes durch Feature-Points (FP). Eine von den Eingabewerten abhängige Visualisierung der Emotionen im Gesicht ist als Ziel zu sehen.

3.6 Neven Vision - Facial Analysis SDK

Das Neven Vision Facial Analysis SDK (vgl. auch im Weiteren [12]) besteht aus mehreren Teilen. Für diese Arbeit sind nur das Face-Detection Modul und das Facial-Feature-Tracking Modul relevant.

Das Face-Detection Modul ist die Grundlage für alle allen anderen Module. Aus einem Grauwertbild mit der minimalen Größe von 128x128 Pixeln wird das Gesicht extrahiert. Für die Gesichtsextraktion werden neuronale Netze und die Gabor-Wavelet-Transformation verwendet, die eine Datenbank mit Eigenschaften über Beispielgesichter als Grundlage haben.
Innerhalb des Facial-Feature-Detection Moduls ist eine Erkennung von Gesichtsmerkmalen in Echtzeit ohne Marker auf einem Videodatenstrom möglich. Darüber hinaus werden in jedem Frame weitere Informationen wie die Position des Kopfes und der Zustand der Augenöffnung extrahiert. Zuerst wird das Face-Detection Modul eingesetzt, um das Gesicht zu finden. Danach findet nur noch ein Tracking statt, das ebenfalls neuronale Netze und die Gabor-Wavelet-Transformation verwendet. Zusätzlich wird ein 3D-Modell des menschlichen Gesichtes eingesetzt, um Tracking-Fehler zu korrigieren.
Zuletzt werden die erkannten Gesichtsparameter (Koordinaten von 22 vordefinierten Punkten) auf einen Avatar übertragen. Dieser kann durch die Bewegungen des Benutzers gesteuert werden. Abbildung 3.3 zeigt die Steuerung eines Avatars entsprechend der Mimik des Benutzers (vgl. [12]).

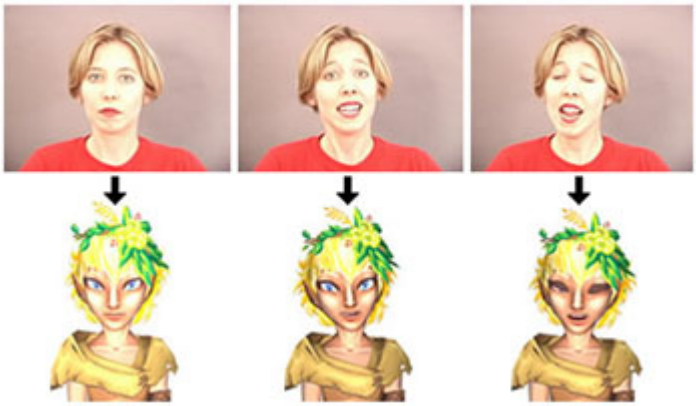

Abbildung 3.3: Neven Vision - Facial Feature Tracking (vgl. [12])

3.7 Zusammenfassung

Nachfolgend werden die wichtigsten Features der einzelnen Anwendungen, die in dieser Arbeit relevant sein werden, miteinander verglichen. Aus der Übersichtstabelle 3.1 lassen sich sehr einfach die Stärken und Schwächen der einzelnen Systeme erkennen. Die mit einem „*“ gekennzeichneten Felder bedeuten, dass der Hersteller oder Autor keine Informationen zu dieser Kategorie angibt. Die pro-

	SeeStorm - Face Analysis SDK	**SeeStorm - Mobile**	**Charamel - Mocito**	**FhG - 3D-Videokonferenz**	**XeneX-Group - XG-Emotion**	**Neven Vision - Facial Analysis SDK**
Verwendete Programmiersprache	C++ C	C++ ASP	*	*	Java Java3D	C++ C
Anwendung auf mobilem Endgerät	-	+	+	+	-	+
Echte 3D-Animation	-	-	+	+	+	-
Animation durch Morphing	-	+	-	-	-	-
Mimikerkennung ohne Marker	+	+	-	+	-	+
Mimikerkennung in Echtzeit	-	-	-	+	-	+
Animationsparameter im MPEG-4 Format	-	-	-	+	+	*

Tabelle 3.1: Marktübersicht Vergleich

totypische Implementierung innerhalb dieser Arbeit wird ausschließlich in der mobilen Java Umgebung erfolgen, da die J2ME-Plattform mittlerweile sehr weit verbreitet ist und somit eine große Zahl von Nutzern angesprochen werden kann. Des Weiteren ist dies in keinem der Vergleichssysteme bislang realisiert worden. Jedoch gibt es bereits Implementierungen für PDAs und Symbian Mobiltelefone. Hierbei ist jedoch keine vollständige Geräteunabhängigkeit gewährleistet und der Marktanteil der Geräte zu gering, um für Netzbetreiber interessant zu sein.
Um ein sehr gutes Resultat bei der Bewegung von Augenlidern sowie beim Öffnen und Schließen des Mundes zu erhalten, sollte ein 3D-Modell verwendet werden. Die Entscheidung über den zu wählenden Ansatz in dieser Arbeit wird von der Leistungsfähigkeit der Java-Umgebung und der Komplexität der Erstellung eines 3D-Schultermodels abhängig sein. Alternativ zum 3D-Modell wird die Animation anhand eines 2D-Modell erfolgen.
Eine unabdingbare Anforderung ist, dass die Mimik ohne Marker erkannt wird. Idealerweise findet dies in Echtzeit statt. Um zwischen verschiedenen Systemen eine Schnittstelle zu schaffen, wäre es von Vorteil, wenn die Animationsparameter

im MPEG-4 Format extrahiert werden können.
Die nachfolgenden Punkte sollen während dieser Diplomarbeit untersucht werden:

- Implementierung unter J2ME
- 3D-Modell (Alternativ: 2D-Modell oder Morphing/Warping)
- Mimikerkennung ohne Marker
- Mimikerkennung in Echtzeit
- Extraktion von Animationsparameter im MPEG-4 Format
- Keine Zwischenschaltung eines Servers für Berechnungen

Kapitel 4

Symbian vs. J2ME

Mit Symbian und J2ME stehen zwei potentielle, weit verbreitete Softwareplattformen zur Verfügung, die hier verglichen werden. Es werden die Vor- und Nachteile von Symbian und J2ME aufgezeigt. Symbian ist ein Betriebssystem, dass in diesem Falle für den Betrieb auf mobilen Endgeräten (Mobiltelefone, PDAs und Smartphones) entwickelt wurde. Wenn in dieser Arbeit von Symbian-Applikationen gesprochen wird, dann handelt es sich um C-basierte Programme (*.sis-Dateien). Im Gegensatz zu Symbian ist J2ME kein Betriebssystem sondern eine Laufzeitumgebung für Applikationen, die speziell für den Betrieb auf Geräten mit wenig Ressourcen entwickelt wurde. Daher ist es möglich, dass Geräte mit einem Symbian-OS gleichzeitig auch über eine J2ME-Laufzeitumgebung verfügen.

4.1 Symbian

Das Symbian Operating-System (OS) ist ein Derivat der 32-Bit EPOC-Plattform von Psion (vgl. auch im Weiteren [13]). Zu Beginn setzten von den großen Herstellern nur Nokia und Sony-Ericsson auf diese Plattform. Mittlerweile sind unter anderem Siemens und Samsung nachgezogen. Das erste Mobiltelefon/Smartphone mit Symbian OS war der Nokia 9210 Communicator, damals mit der Symbian Version 6.0 (vgl. [42]). Auf Symbian basiert auch die Series-60-Benutzeroberfläche, die bei einigen Mobiltelefonen von Nokia (3650, 7650, ...), Siemens (SX1) und anderen Hersteller eingesetzt wird. Außer beim Nokia 6600 (V7.0s) ist bei den genannten Handys die Version 6.1 installiert. Symbian setzt dabei auf offene Standards, um eine zügige Verbreitung auf dem Markt zu erreichen.

4.2 J2ME

Die Grundlage von J2ME bilden die Java Virtuelle Maschine (JVM), die Konfigurationen und die Profile (vgl. auch im Weiteren [43]). Eine Konfiguration ist entweder eine Connected Device Configuration (CDC), die über eine full-featured JVM verfügt, oder für mobile Endgeräte mit wenig Ressourcen ist es die Connected Limited Device Configuration (CLDC), die über die Kilobyte Virtual Machine (KVM) verfügt. Bei den CLDCs gibt es die am verbreitesten Version 1.0 sowie die Version 1.1, die auf vielen neuen Geräten angekündigt ist (z.B. Nokia 6230)(vgl. [5]). CLDCs spezifizieren die Zugriffsmöglichkeiten auf die Hardware eines Java-fähigen Endgerätes. Bei den Profilen unterscheidet man zwischen der Mobile Information Device Profile (MIDP) Version 1.1 sowie 2.0. In den Profilen wird die Grundmenge der Software-Funktionen spezifiziert. Der wesentliche Unterschied von MIDP 1.1 zu MIDP 2.0 ist die Erweiterungen des Application Programming Interfaces (API) um zum Beispiel:

- Wireless Messaging API
- Mobile Media API
- Security and Trust Services API
- Mobile 3D-Graphics
- Bluetooth API

Aus der Sicht eines Softwareherstellers hat J2ME den Vorteil, dass Anwendungen auf allen Geräten lauffähig sind, die die entsprechende API installiert haben. Somit ist eine geräteunabhängige Entwicklung möglich.
Ein großes Manko bei J2ME-Applikationen ist die mangelnde Performance. Eine Möglichkeit die Performance zu verbessern besteht darin, dass ein Hersteller eigene Bibliotheken definiert (z.B. Image Processing). J2ME stellt dann einen Funktionsaufruf und einen Ausgabewert zur Verfügung, die eigentliche Berechnung erfolgt aber auf Ebene des Betriebssystems. Somit können Funktionen, die sehr viel Leistung benötigen, schnell auf Betriebssystemebene berechnet werden, gleichzeitig kann aber ein Zugriff durch J2ME erfolgen. Entweder werden solche Bibliotheken von den führenden Hersteller in eine Spezifikation übernommen, oder es entsteht das Problem, dass keine Geräteunabhängigkeit mehr garantiert werden kann.
Weiterhin könnten durch das Java Native Interface (JNI), das in der KVM nicht vorhanden ist, performancelastige Teile von Anwendungen ausgelagert werden. Jedoch ist eine Realisierung eines JNI schwierig, da es kein herstellerübergreifendes Betriebssystem gibt. Des Weiteren möchten die Hersteller einen tiefen Eingriff in das Betriebssystem aus Sicherheitsgründen verhindern.

4.3 Vergleich von Symbian und J2ME CLDC 1.0

Der Grundlegende Unterschied zwischen J2ME mit der CLDC 1.0 und Symbian ist, dass Symbian-Anwendungen eine wesentlich bessere Performance als J2ME-Anwendungen haben. Des Weiteren ist bei Symbian-Anwendungen der Zugriff auf sämtliche Hardwarekomponenten gegeben. Bei J2ME mit CLDC 1.0 wird dies aus Sicherheitsgründen momentan nicht ermöglicht. Es besteht aktuell bei J2ME CLDC 1.0 beispielsweise keine Möglichkeit, während der Laufzeit Dateien zu schreiben (vgl. auch im Weiteren [44] S.121). Daten können nur in sogenannten Records persistent gespeichert werden. Ein Record wird als Bytefeld gespeichert und wird über die `recordId` identifiziert. Während der Laufzeit können Daten nur aus der *.jar-Datei oder aus einem Record geladen werden.
Symbian-Anwendungen, die keine gerätespezifischen APIs benötigen sind auch auf jedem Symbian Gerät lauffähig. Jedoch ist der Umfang von Geräteunabhängingen APIs im Vergleich zu Java recht gering. J2ME-Anwendungen sind ebenfalls an kein Endgerät gebunden, solange sich der Hersteller an die entsprechende Java-Spezifikation gehalten hat. In diesem Zusammenhang sollte auch berücksichtigt werden, dass es zu Kompatibilitätsproblemen aufgrund unterschiedlicher Hardware kommen kann. Wenn zum Beispiel eine Software für ein Gerät mit einer Displaygröße von 150x250 Pixeln entwickelt wurde und diese Applikation auf einem Gerät mit einer Displaygröße von 100x200 Pixeln dargestellt werden soll, kann es sehr schnell zu Problemen kommen. Im CLDC-

	Symbian	J2ME CDLC 1.0
Unabhängig von der Hardware	-	+
Zugriff auf sämtliche Hardware-Komponenten	+	-
Lesen und Schreiben von Dateien während der Laufzeit	+	-
Hohe Performance	+	-
Echte Floating Point Operations	+	-
Native Interface	+	-

Tabelle 4.1: Vergleich von Symbian- mit J2ME-Anwendungen

Profil 1.1 werden einige Einschränkungen der J2ME-Plattform aufgehoben (vgl. auch im Weiteren [23] S.3). Unter Anderem wird die Gleitkomma-Berechnung unterstützt (Klassen Float und Double) und die Klassen Calendar, Date, TimeZone werden J2SE-ähnlich. Zusätzlich wird die minimale Größe des permanenten Speichers für die VM und die CLDCs von 128kB auf 160kB erhöht. Die Größe des flüchtigen Speichers für die Laufzeitumgebung der VM beträgt 32kb.
Ein wichtiger Faktor für die Entscheidung, den Software-Prototypen in J2ME

zu implementieren, ist die wesentlich weitere Verbreitung von javafähigen Endgeräten. Um die Performance von J2ME-Plattformen zu steigern, hat SUN unter dem Namen „Lip Java Technology for Smartphones Hot-Spot Implementation“ eine Hot-Spot Implementierung angekündigt, die etwa um den Faktor sechs schneller sein soll als die bisherige J2ME-VM (vgl. [39]).

Kapitel 5

Gesichtserkennung

Das Ziel der Gesichtserkennung ist die Bestimmung sämtlicher Gesichter auf einem frei gewählten Bild. Die Gesichtslokalisierung ist eine Untermenge der Gesichterkennung, da dabei von einem Bild ausgegangen wird, auf dem genau ein Gesicht vorhanden ist. Von diesem Gesicht muss nur noch der Bereich im Bild bestimmt werden.
In diesem Kapitel werden zwei Schritte betrachtet: Die Gesichtslokalisierung und die Extraktion von Gesichtsmerkmalen.

5.1 Verfahren Gesichtslokalisierung

Ein Ziel dieser Arbeit ist, ein einfaches, robustes und ressourcenschonendes Verfahren zur Erkennung der Mimik zu finden. Um die Mimik zu erkennen, muss zuerst das Gesicht im Eingangsbild lokalisiert werden. Dabei wird die Annahme getroffen, dass immer ein Gesicht auf dem Bild vorhanden ist. Daher gibt es keine Bilder mit zwei Gesichtern oder komplett ohne Gesicht. Nachfolgend werden vier verschiedene Verfahren zur Gesichtslokalisierung evaluiert.

5.1.1 Ellipse Fit

Die Umrisskanten des menschlichen Kopfes ähneln einer Ellipse. Daher erweist es sich als praktikabel, bei der Objektsuche nach einem Kopf, eine Ellipse als Modell zu wählen. Somit basiert dieses Verfahren auf der Suche nach einem elliptischen Objekt im Bild. Um eine Objektsuche zu ermöglichen, muss zuerst ein Sobel-Filter für die Kantenextraktion auf das Bild angewendet werden. Eine Ellipse wird neben dem Mittelpunkt (x, y) noch durch die zwei Halbachsen a und b parametrisiert (siehe Abb. 5.1). Da eine Ellipse ebenfalls, wie in Kapitel 2.1.5

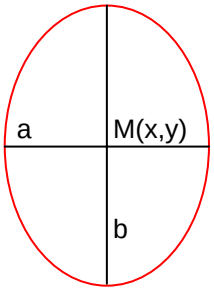

Abbildung 5.1: Parametrisierung einer Ellipse

beschrieben, eine parametrisierbare Randkurve ist, lässt sich diese auch mittels der Hough-Transformation finden.
Bei diesem Ansatz sind die Parameter a und b bekannt, da sonst der Rechenaufwand exponentiell ansteigt. Das Eingangsbild hat immer dieselbe Größe, daher können mit vier unterschiedlichen Parametern a und b bereits gute Ergebnisse erreicht werden. Die Abbildung 5.2 zeigt links das Eingangsbild. Im nächsten Schritt wird der Sobelfilter zur Kantendetektion eingesetzt. Danach wird die ähnlichste der vier Ellipsen auf das Gesicht abgebildet. Das Ellipse Fit Verfahren eig-

Abbildung 5.2: Ellipse Fit

net sich sehr gut für die Gesichtslokalisierung, wenn der Kontrast zwischen Kopf und dem Hintergrund stark genug ist. Solange der Sobel-Filter brauchbare Kanten liefert, funktioniert das Verfahren gut. Um die Geschwindigkeit von Ellipse Fit zu beschleunigen, sind die Parameter a, b vorgegeben, da sonst die Möglichkeiten, eine Ellipse zu finden einen zeitlich zu hohen Faktor benötigen würden.

5.1.2 Deformable Line Templates

Ein Gesicht kann mittels des Template-Matching lokalisiert werden. Hierbei werden einzelne Merkmalspunkte oder auch Umrisslinien des Gesichtes lokalisiert. Zu diesem Zweck werden Kurvenvorlagen zu den entsprechenden Linien im Bild gesucht.
Eine Kurvenvorlage (Line Template) besteht aus einzelnen Segmenten (vgl. auch im Weiteren [14] S.10). Diese müssen entsprechend der Vorlage angepasst werden, da die Line Templates in den seltensten Fällen ohne Anpassung mit der Vorlage übereinstimmen. Die einzelnen Segmente eines Line Templates können verschoben werden und ermöglichen somit durch Verzerrung eine Anpassung an das Original. Die Möglichkeit der Verzerrung ist begrenzt, um sich nicht zu weit vom Original zu entfernen. Im Prinzip muss eine möglichst exakte Überdeckung mit einer im Bild vorhandenen Kontur gefunden werden, ohne dabei zu sehr von dem Template abzuweichen.
Der Aufwand für die Berechnung ist abhängig von der Anzahl der Segmente und den erlaubten Veränderungen. Der Aufwand für die Änderung eines Templates ist exponentiell, da jede Änderung eines Templates, eine Änderung aller anderen Templates nach sich zieht.
Aufgrund der geringen Performance auf mobilen Endgeräten ist dieses Verfahren für das Lokalisieren eines Gesichtes nicht geeignet, da es bereits auf handelsüblichen PCs zu Performanceproblemen kommt.

5.1.3 Augen-Mund Abstand

Ein Verfahren für die Lokalisierung von Gesicht, Augen sowie Mund ist die Kombination aus Abstand und räumlicher Lage von Augen und Mund. Innerhalb dieser Methode wird das Wissen genutzt, dass beide Augen auf ungefähr gleicher Höhe sind und unterhalb davon, in der Mitte der Mund liegt.
Nachfolgend wird der „Lip Finding Algorithm“ (vgl. auch im Weiteren [17]) detailliert beschrieben. Auf das Eingangsbild wird zunächst ein horizontaler Gradientenfilter angewendet. Als nächster Schritt findet eine Threshold-Filterung statt. In der anschließenden Segmentierung werden alle unrelevanten Segmente (zu klein, zu groß oder falsche Form) eliminiert. Für jedes der restlichen Segmente wird eine Bounding-Box erstellt. Pro Frame sind jetzt noch ungefähr 10-25 Segmente übrig. Nun wird berechnet, welche Gruppe von Segmenten am besten auf das Gesichtsmodell zutreffen. Es wird der Abstand zwischen dem Mittelpunkt der Bounding-Box der linken $C(e_l)$ und der rechten Augenbraue $C(e_r)$ gemessen sowie der Abstand vom Mittelpunkt der Verbindung der beiden Augenbrauensegmente zum Mundsegment $C(m)$. Ausgewählt werden schließlich die Segmente, die den minimalen Abstand bilden. Nach Klaus Lukas (vgl. [17]) ist das Verhält-

nis von a (Abstand der Augensegmente) und b (Mittelpunkt der Augenbrauensegmente zum Mundsegment) ungefähr $K = b/a = 1,2$. Der gesamte Prozess ist in Abbildung 5.3 dargestellt. Das Bild im unteren rechten Viertel zeigt die Abstandsmessung der einzelnen Parameter. Dieser Ansatz wird bereits auf Symbian-Basis

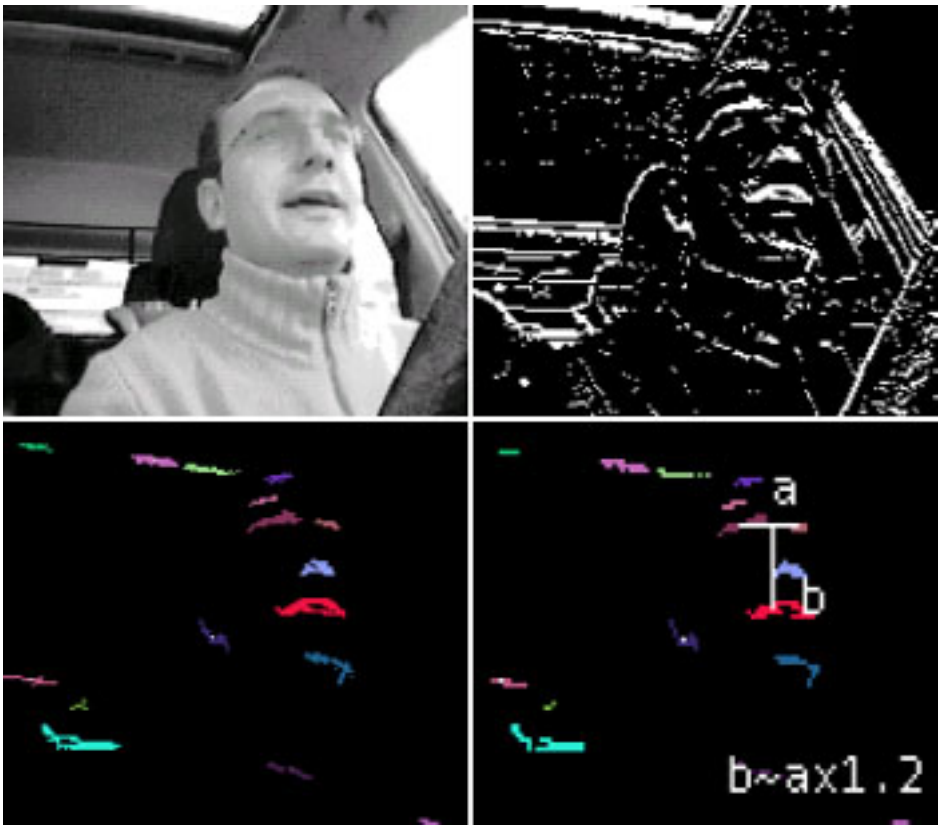

Abbildung 5.3: Feature Erkennung durch Augen-Mund Abstand (vgl. [17])

von Klaus Lukas und Jesus Guitarte bei Siemens Corporate Technology - Speech Center verfolgt (vgl. [17]). Daher wird der Mund-Augen Abstand Ansatz in dieser Arbeit nicht implementiert, dient aber als Vergleich zu Symbian-Systemen.
Das Verfahren funktioniert bereits sehr gut, jedoch liegt der Schwerpunkt bei der Erkennung von Mund-Parametern und nicht bei der Erkennung von unterschiedlichen Gesichtsfeatures. Ziel ist ein Lippenlesen durchzuführen und nicht die Animation von Gesichtern durch Mimikparameter.

5.1.4 Farbklassifikation

Die Erkennung von Hautfarbe ist eine sehr effektive und weit verbreitete Bildverarbeitungsmethode für das Finden und Verfolgen von Gesichtern. In vielen Arbeiten wird gezeigt, dass die Hautfarbe eine markante Farbe ist und für alle Menschen als ungefähr gleiche Farbe angenommen werden kann. In [27] wurde beispielsweise bewiesen, dass die Hautfarbwerte aller ethnischer Gruppen sich innerhalb eines kleinen Gebiets im HSV-Farbraum häufen. Die Hautfarbe ist ein Merkmal, dass sich sehr schnell und stabil berechnen lässt.
Die Versuche während dieser Arbeit und zahlreiche Beispiele aus der Literatur beweisen, dass das Merkmal Hautfarbe sehr spezifisch ist. S. Gong (vgl. [27]) hat in seiner Arbeit Hautfarbbilder untersucht und diese in den HSV-Farbraum übertragen. Der Autor vertritt auch die Meinung, dass das Ignorieren von Luminanz (Intensität) sofort eine Unabhängigkeit von Lichtverhältnissen liefert. Deswegen werde die Intensitätskomponenten der Pixel nicht betrachtet und nur die H und S-Komponenten in die Polarkoordinaten eingetragen. Demzufolge befindet sich die menschliche Hautfarbe im HSV-Farbraum hauptsächlich zwischen H=6° und H=36°. Die statistischen Auswertungen die durch 20 verschiedene Personen, unter differenzierten Beleuchtungsverhältnissen gemacht wurden bestätigen diese Angabe weitestgehend. Die Durchschnittswerte betragen H=353° bis H=39° und $S < 0{,}57$. Die Abbildung 5.4 zeigt beispielhaft eine Segmentierung nach Hautfarbe durch die Parameter H=353° bis H=39° und $S < 0{,}57$. Die Berechnung der Bounding-Box erfolgte durch das Wissen, dass der Hals zum Hautfarbensegment gehört. Das Verfahren ist sehr gut für eine Anwendung auf mobilen Endgeräten

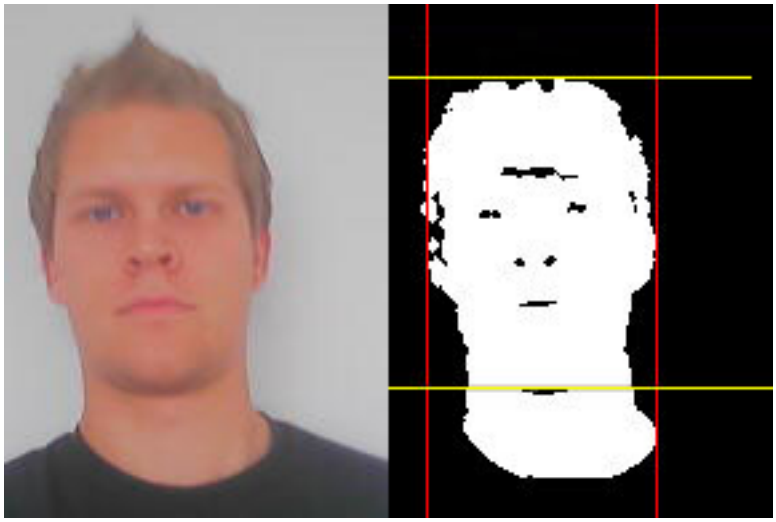

Abbildung 5.4: Farbklassifikation im HSV-Raum

geeignet, da die Abfrage der Farbkomponente ohnen größeren Rechenaufwand erfolgen kann. Die Vorverarbeitung des Bildes hat einen entscheidenden Einfluss auf die Qualität und Geschwindigkeit des Verfahrens.

5.2 Extraktion von Gesichtsmerkmalen

Bei der Extraktion von Gesichtsmerkmalen geht es um das Lokalisieren von bestimmten Referenzpunkten eines gegebenen Gesichtes. In dieser Arbeit wird ein bestehendes Drahtgittermodell anhand der gefundenen Merkmale wie Augen (Iris) und Mund (Lippen) angepasst. Im vorherigen Abschnitt wurden Verfahren zur Gesichtslokalisierung beschrieben, die eine Face Bounding-Box (BB) als Resultat liefern. Die nachfolgenden Methoden gehen daher von einem lokalisierten Gesicht als Eingabe aus.

5.2.1 Finden der Iris

Die Augen befinden sich in der oberen Hälfte des Gesichtes. Daher findet die Suche nach der Iris nur in der oberen Hälfte der Gesichts-BB statt. Um die Suche noch weiter zu vereinfachen, gibt es eine Bounding-Box für das linke und für das rechte Auge. Die obere Hälfte der Gesichts-Bounding Box wird horizontal nochmals halbiert. Die Abbildung 5.5 zeigt in orange: die BB des Gesichtes, in pink die BB für die Iris gesamt, in grün BB Iris links und in rot wird die gefundene Iris auf der linken Seite dargestellt. Mit den nachfolgenden Eigenschaften lässt

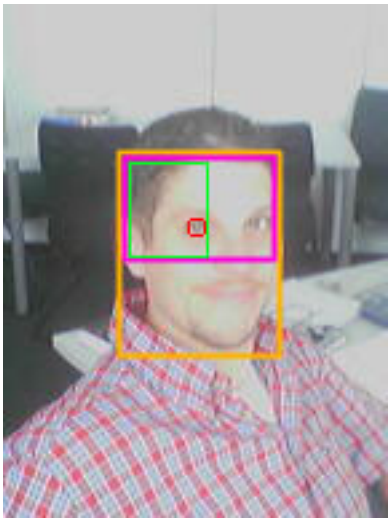

Abbildung 5.5: Bounding-Boxes für Iris-Erkennung

sich die Iris recht einfach finden:

- sie ist fast kreisförmig
- beide Augen liegen im Bild auf ungefähr gleicher Höhe
- dunkle Iris auf hellem Hintergrund

Die Augen liegen auf ungefähr gleicher Höhe, daher ist es recht einfach, fehlerhafte Erkennungen auszuschließen. Die Iris wird hauptsächlich durch drei Parameter beschrieben (vgl. auch im Weiteren [18]): Zentrumskoordinaten (x_0, y_0) sowie Radius (r). Im Kapitel Grundlagen wurde die Hough-Transformation vorgestellt, die sich für beliebig parametrisierbare Randkurven anwenden lässt. Daher ist es möglich, einen Kreis der durch die Gleichung 5.1 beschrieben ist, sehr leicht finden. Bevor man jedoch die Hough-Transformation für Kreise auf die Region of Interest (ROI) anwendet, wird die ROI zuerst mit einem Sobel-Operator gefiltert.

$$(x - x_0)^2 + (y - y_0)^2 = r_0^2 \tag{5.1}$$

Dazu wir für jeden Punkt (x_0, y_0) der ROI die Summe aller Grauwerte der Punkte (x, y), die um den Radius r_0 von (x_0, y_0) entfernt sind und die Gleichung 5.1 erfüllen, in ein dreidimensionales Akkumulator Array an die Stelle (x_0, y_0, r_0) eingetragen.
Es wird für jede mögliche Kombination von Mittelpunkt (x_0, y_0) und Radius (r_0) die komplette ROI durchlaufen und muss es muss gleichzeitig die Kreisgleichung erfüllt sein. Die gesuchten Parameter werden durch die maximalen Werte im Akkumulator dargestellt.
Für das Gesichtsmodell (Abb. 6.1) sind die Punkte 14 und 16 sowie 18 und 20 relevant. Die Abbildung 6.1 stellt als blauen Punkte die Knoten im Gesichtsmodell dar. Die grünen Punkte repräsentieren die Iris-Mittelpunkte.

Abbildung 5.6: Iriserkennung

$$P18_x = x_l \tag{5.2}$$

$$P18_y = y_l + r_l \tag{5.3}$$

Die Gleichung 5.2 zeigt die Berechung des x-Wertes von Punkt 18. Dabei wird der x-Wert des linken Mittelpunktes (x_l) als x-Wert für Punkt 18 gewählt. Der y-Wert des Punkt 18 (vgl. Abb. 5.3) berechnet sich durch die Addition y-Wert des linken Mittelpunktes (y_l) und dem Radius (r_l) der linken Iris. Beide Augenmittelpunkte haben ungefähr den gleichen y-Wert. Daher wird nach der Erkennung der beiden Mittelpunkte eine Differenz des y-Wert gebildet. Falls die Differenz größer als $2 * r_l$ ist, dann war die Erkennung fehlerhaft und die Operation muss noch einmal neu durchgeführt werden.

5.2.2 Finden des Mundes

Durch die Vorverarbeitung (Lokalisierung des Gesichtes), sind die Möglichkeiten, nach dem Mund zu suchen eingeschränkt. Der Mund muss beispielsweise in der unteren Hälfte der Bounding-Box des Gesichtes liegen (sofern diese korrekt ist). Die Lippen heben sich besonders durch die Farbeigenschaft vom Umfeld ab. Hierbei wird nur der Farbwert H innerhalb des HSV-Farbraums berücksichtigt. Es wurden bei 20 verschiedenen Fotos die Lippenregionen untersucht. Das Ergebnis war, dass die Lippen am häufigsten im Farbbereich von H=345° bis H=9° zu finden sind. Die Abbildung 5.7 zeigt die Lippenerkennung innerhalb der Bounding-Box des Gesichtes bei geschlossenem sowie bei geöffnetem Mund. Die Bounding-Box des Mundes ist in Abbildung 5.7 pink dargestellt. Diese wird zur Platzierung der angegebenen Punkte benötigt. Die restlichen Punkte in der Mundregion (Punkte 42-49 vgl. Abb. 6.1) werden berechnet, um eine bessere Performance zu erreichen, da sonst weitere Bildverarbeitungsschritte für die Erkennung notwendig wären. Aufgrund der geringen Auflösung des Eingangsbildes ist es teilweise schwierig, alle Punkte mittels Bildverabeitungsoperationen zu finden.

Die Punkte 58 und 59 liegen immer am rechten beziehungsweise linken Rand innerhalb der Mund-Bounding-Box. Wenn man die Box horizontal in der Mitte teilt, dann finden sich die Punkte 52 und 56 auf den Schnittpunkten zwischen Trennlinie und Mundkante wieder. Um die Punkte 54, 57 und 50, 55 zu platzieren wird die Breite der Mund-BB als Grundlage genommen. Dabei wird von der Breite (von links ausgehend) 10% ermittelt, und von diesem Abstand aus wird vertikal mit der Mundkante geschnitten. Dabei wird der obere Schnittpunkt der 54 (50) und der untere der 57 (55) zugeordnet. Für die rechtsliegenden Punkte kommt das gleiche Verfahren zum Einsatz, nur dass 90% der Breite, von links ausgehend, gewählt wird. Auf der oberen Kante der Mund-BB liegen die Punkte 51 und 53.
Nun wird die Berechnung der noch fehlenden Punkte 42-49 dargestellt. Der Punkt 43 bekommt den x-Wert von Punkt 52 zugewiesen (vgl. Gleichung 5.4). Der y-Wert ist der Wert des Punkt 52 addiert mit 20% der Höhe der Bounding Box (vgl. Gleichung 5.5).

$$P43_x = P52_x \tag{5.4}$$

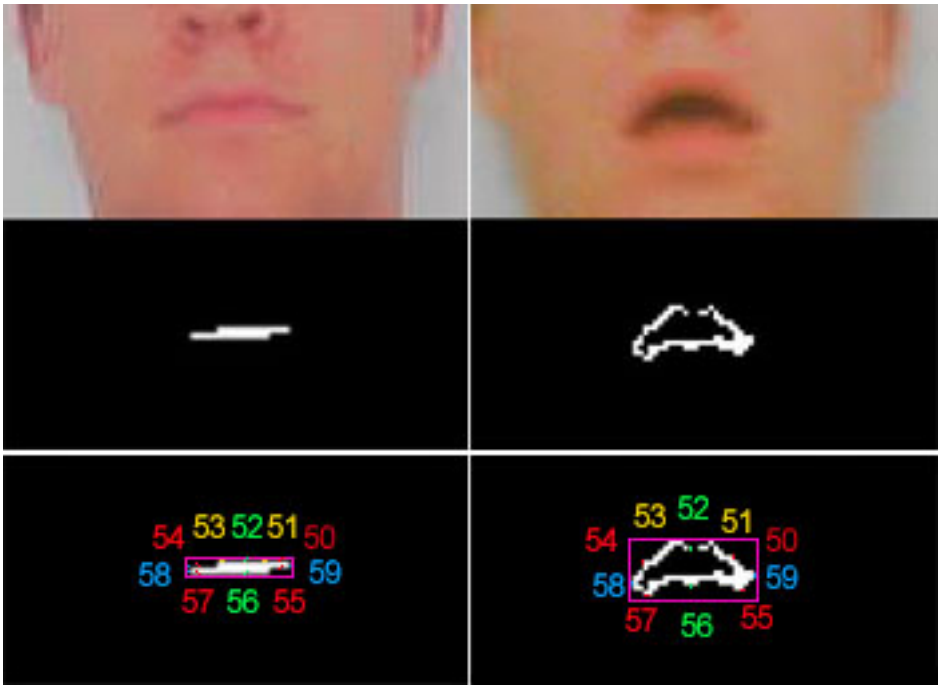

Abbildung 5.7: Lippenerkennung

$$P43_y = P52_y + BB_H * 0,2 \tag{5.5}$$

Analog hierzu orientiert sich der Punkt 46 an Punkt 56 mit dem Unterschied, dass 20% der Höhe vom y-Wert des Punktes P56 subtrahiert werden.
Der Punkt 42 berechnet sich anhand der des x-Wertes des Punkt 43 (vgl. Gleichung 5.6) und des y-Wertes von Punkt 51:

$$P42_x = P51_x \tag{5.6}$$

$$P42_y = P43_y \tag{5.7}$$

Der Punkt 44 wird analog in Abhängigkeit von Punkt 43 und 53 berechnet. Um den x-Wert von 49 zu berechnen werden die x-Werte der Punkte 43 und 46 addiert und danach halbiert (vgl. Gleichung 5.8). Der y-Wert des P49 ist identisch mit dem des Punktes 59 (vgl. Gleichung 5.9). Der Punkt 48 wird analog berechnet.

$$P49_x = (P43_x + P46_x)/2 \tag{5.8}$$

$$P49_y = P59_y \tag{5.9}$$

Zuletzt fehlen noch die Punkte 45 und 47. Der Punkt 47 berechnet sich aus der Addition des y-Wertes von P49 und P46. Danach muss noch eine Division durch Zwei erfolgen (vgl. Gleichung 5.11). Der x-Wert wird von Punkt 44 übernommen (vgl. Gleichung 5.10). Der Punkt 45 wird analog berechnet.

$$P47_x = P44_x \tag{5.10}$$

$$P47_y = (P49_y + P46_y)/2 \tag{5.11}$$

Die restlichen Punkte sind in einer normalisierten Tabelle abgelegt. Die Beschreibung hierzu erfolgt in Kapitel 6.

5.3 Face Finding und Feature Extraction

Zu Beginn der Anwendung findet zuerst ein Face Finding statt, da zunächst keine Informationen über vorhergegangene Frames vorhanden sind. Des Face Finding wird zu Beginn der Anwendung und bei erfolglosem Face Tracking durchgeführt, da dieser Prozess wesentlich aufwändiger als ein Face Tracking ist.
Die Tabelle 5.1 zeigt den Prozess des Face Findings. Hier wird das Gesicht im Bild gesucht. Das Ziel ist, eine Bounding-Box für das Gesicht auszugeben. Gleichzeitig wird eine auf der BB des Gesichtes basierende, separate BB für den Mund berechnet.
Die Eingabe besteht aus einem Einzelbild im png-Format. Da es in J2ME es nicht möglich ist, direkt ein HSV-Bild zu erzeugen, wird zunächst ein RGB-Bild erstellt. Dieses wird dann in den HSV-Farbraum transformiert, um leichter anhand des Farbwinkels und der Helligkeit segmentieren zu können. Auf das binarisierte Bild wird zuerst eine Opening- und dann eine Closing Operation ausgeführt. Dies dient dazu, Störpixel, die bei der Berechnung der Bounding-Box zu falschen Ergebnissen führen würden, zu eliminieren. Die Face Bounding-Box wird anhand der maximalen Ausdehnung des erhaltenen Segmentes ermittelt. Während der Arbeit hat sich herausgestellt, dass die unteren 25% der Bounding-Box abgeschnitten werden können, da hier der Hals zu sehen ist. Im nächsten Schritt werden die Seitenverhältnisse der Bounding-Box geprüft. Das ideale Verhältnis der Seitenhöhe zur Seitenbreite entspricht 3:2. Die Box muss immer höher als breit sein. Als gültiger Bereich hat sich alles zwischen 2:1 und 4:3 herausgestellt. In dieser Arbeit werden beim Gesicht die Augen sowie der Mund als relevante Merkmale angesehen. Es gibt noch weitere Merkmale wie Nase und Kinn, mit denen das Gesicht noch besser beschrieben werden könnte, jedoch ist dies durch die zeitliche Begrenzung der Arbeit nicht möglich.
In Tabelle 5.2 wird der Ablauf der Merkmalsextraktion beschrieben. Die Merkmale sind in dieser Arbeit auf Augen und Mund beschränkt. Als Eingabe wird eine

Face Finding
Eingabe: Bildsequenz png-Bilder
↓
Erzeugung eines RGB Image-Objektes
↓
Transformation von RGB nach HSV
↓
Segmentierung anhand des Farbwinkels und der Helligkeit der Hautfarbe
↓
Kombination von Opening- und Closing
↓
Berechnung der Face Bounding Box
↓
Prüfen der Seitenverhältnisse der BB und Anpassung an Seitenverhältnis
↓
Ausgabe: Gesichts-BB, Augen-BB und Mund-BB

Tabelle 5.1: Face Finding

Bounding Box für die Augen und für den Mund erwartet. Die BBes können entweder vom Face Finding- oder vom Tracking-Modul kommen. Zunächst wird das Vorgehen für die Augen beschrieben. Der Prozess ist für beide Augen gleich. Daher folgt die Erkennung des rechten analog zum linken Auge. Die Bounding-Box des Auges wird vertikal geteilt, um einen Suchbereich sowohl für das linke und das rechte Auge zu bestimmen. Später wird nach einer Kreisform (Iris: dunkler Kreis auf hellem Untergrund) gesucht. Aus diesem Grund muss zuerst ein Sobel-Filter angewendet werden. Dieser unterdrückt das Bildrauschen und bestimmt die Kanten. Danach kommt die Hough-Transformation zu Bestimmung der Kreise zum Einsatz. Die Performance kann durch die Vorgabe des minimalen und maximalen Kreisradius deutlich verbessert werden. Als Ergebnis wird der Mittelpunkt des Auges sowie der Radius zurückgeliefert. Die detaillierte Beschreibung für die Berechnung der Mesh-Punkte sind in Kapitel 5.2.1 zu finden. Nach der Berechnung werden die Punkte 14,16,18 und 20 ausgegeben.
Für die die Extraktion des Merkmals Mund werden die unteren 50% der Face Bounding-Box herangezogen. Das entscheidende Merkmal des Mundes ist der Farbkontrast gegenüber der umgebenen Haut. Daher ist die effizienteste Methode, die Lippen anhand des Farbwinkels zu segmentieren. Die Segmentierung liefert ein binäres Bild zurück, auf dem die Umrisse des Mundes zu sehen sind (vgl. Abb. 5.7). Anhand der maximalen Ausdehnung des Segments wird die Bounding-Box des Mundes berechnet. Diese wiederum hilft, die enstprechende Punkte der

Feature Extraction	
Auge	Mund
Eingabe: Augen-BB	*Eingabe: Mund-BB*
↓	↓
Aufteilen der Iris-BB 50% vertikal für linkes und rechtes Auge	Suchen des Mundes innerhalb der Mund-BB
↓	↓
Sobel Filter	Segmentierung anhand der Lippenfarbe
↓	↓
Hough-Transformation für Kreise	Berechnung der Mund-Bounding-Box anhand des Segmentes
↓	↓
Mittelpunkt und Radius	Erkennung der Mundpunkte
↓	↓
Berechnung der Mesh-Punkte	Berechnung der fehlenden Punkte
↓	↓
Ausgabe: Punkte 18, 20 und 14, 16	*Ausgabe: Punkte 42-59, Mund-BB*

Tabelle 5.2: Feature Extraction

Lippen zu erkennen. Fehlende Punkte werden auf Basis der Bounding-Box sowie der anderen Punkte berechnet. Da bei einem idealen Modell von einem annähernd symmetrischen Mund ausgegangen wird, können nicht erkannte Punkte teilweise auch berechnet werden. Es folgt die Ausgabe der Mesh-Punkte 42-59 sowie die Ausgabe der Mund-BB.

5.4 Face Tracking

In diesem Abschnitt wird das Face Tracking beschrieben. Sobald der Face Finding Prozess erfolgreich abgeschlossen wurde, startet der Feature Extraction Prozess. Damit ist die Initialisierungphase abgeschlossen. Wenn die Feature Extraction gültige Werte für Mund und Augen ausgegeben hat, wird für den nächsten Frame das Face Tracking verwendet. Für das Auge werden die absoluten Mittelpunkte der Iriszentren sowie die Augenradien als Eingabe erwartet. Aus diesen Werten wird eine Bounding-Box um das erkannte Feature gezeichnet. Die neue Bounding-Box des Auges wird vom Mittelpunkt der Iris ausgehend berechnet. In Abbildung 5.8 wird in rot die neue Bounding-Box dargestellt. Der Eckpunkt der neuen Bounding-Box berechnet sich wie folgt: $x = M_x - 2 * r$ und $y = M_y - 2 * r$. Dabei ist M der Mittelpunkt und r der Radius. Die Berechnung für den Eckpunkt unten links ist analog. Diese neue BB dient dann als Eingabe für die Feature Ex-

Face Tracking	
Auge	Mund
Eingabe: Mittelpunkt und Radius beider Augen ↓ Berechnung der ROI für das Tracking ↓ *Ausgabe: Augen-Tracking-BB*	*Eingabe: Mund-BB* ↓ Berechnung der ROI für das Tracking ↓ *Ausgabe: Mund-Tracking-BB*

Tabelle 5.3: Face Tracking

traction. Bei dem Mund wird ähnlich verfahren, da auch hier der Suchbereich für

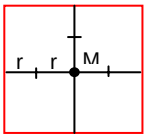

Abbildung 5.8: Berechung der Augen-BB durch das Tracking

die Lippen eingeschränkt wird. Die Abbildung 5.9 zeigt, die Vergrößerung des Suchbereiches der alten Bounding-Box (schwarz) gegenüber der neu Bounding-Box (rot). Die alte BB wird um 50% ihrer Höhe nach oben und unten erweitert. Für die Breite gilt das Gleiche analog.

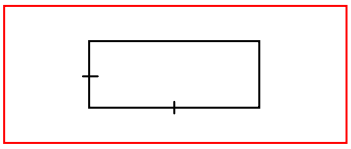

Abbildung 5.9: Berechung der Mund-BB durch das Tracking

5.5 Implementierung

Es wurde das Ellipse Fit Verfahren und die Farbwertsegmentierung zur Gesichtslokalisierung unter J2ME implementiert, und auf einem Siemens CX65 Mobiltelefon getestet. Die Eingabebilder haben alle eine Größe von 144x192 Pixeln und werden im png-Format eingelesen. Die Ergebnisse werden hinsichtlich Zeit, Qualität und Robustheit in Tabelle 5.4 verglichen. Die Qualität sowie die Robustheit

der Ergebnisse wird in der Skala von 1-3 gemessen, wobei 3 für den schlechtesten Wert steht.

Verfahren	Zeit (ms)	Qualität	Robustheit
1.) Farbwinkel	15	3	3
2.) Farbwinkel+Helligkeit	17	2	2
3.) Ellipse Fit	44	1	1

Tabelle 5.4: Vergleich von Verfahren zur Gesichtslokalisierung

Das erste Verfahren (Merkmal Farbwinkel) hat den Vorteil, dass es am schnellsten ist, jedoch ist Qualität und Robustheit im Vergleich zu den beiden anderen Verfahren schlechter, da beispielsweise die Farbintensität bei der Segmentierung nicht berücksichtigt wird. Wenn zum Farbwinkel noch die Helligkeit hinzugenommen wird, ist das Verfahren geringfügig langsamer, aber die Qualität und Robustheit nehmen deutlich zu. Das Verfahren Ellipse Fit liefert zwar die qualitativ besten Resultate, jedoch ist es zu langsam. Ellipse-Fit benötigt im Schnitt fast die dreifache Zeit für die Gesichtslokalisierung
Das Attribut Farbwinkel in Kombination mit der Helligkeit liefert das beste Ergebnis, wenn man die Faktoren Zeit, Qualität und Robustheit bewertet. Daher wurde in dieser Arbeit zur Gesichtslokalisierung der Farbwinkel in Kombination mit der Helligkeit als Entscheidungskriterium gewählt.
Im nächsten Absatz werden die Implementierungen zur Feature Extraktion vorgestellt. Die Erkennung der Augen findet mittels der Hough-Transformation für Kreise statt. Die Kreiserkennung funktioniert ähnlich der Geradenerkennung (vgl. Kapitel 2.1.5). Durch jeden Punkt im Kantenbild wird ein Kreis mit vorher festgelegtem Radius gelegt. Die Punkte in denen sich die meisten Kreise schneiden, werden als Kreismittelpunkte in den Originalbereich rücktransformiert. Während der Implementierung hat es sich als vorteilhaft erwiesen, den Durchmesser des Kreises auf ein Minimum und ein Maximum zu beschränken, da hierdurch ein Performancegewinn von 30% ermöglicht wird. Der maximale Radius beträgt 10 und der minimale zwei Pixel. Je nach erwartetem Eingangsbild können diese Werte angepasst werden. Insbesondere beim Tracking sind diese Informationen relevant. Das Verfahren beginnt bei der Suche immer mit dem zuletzt erfolgreichen Radius. Die Implementierung der Mundsegmentierung anhand der Farbe basiert auf der Segmentierung der Hautfarbe. Es werden nur andere Parameter für die Selektion des Farbwinkels übermittelt.

5.6 Ergebnisse

Der gewählte Ansatz zeigt, dass es auf einem mobilen Endgerät möglich ist, Gesichtsfeatures zu extrahieren. Bisher jedoch ist dies noch nicht in Echtzeit möglich, da einerseits die Bildverarbeitungsoperationen sehr aufwendig sind und andererseits die J2ME-Plattform momentan nicht genügend Performance zur Verfügung stellt. Ein weiteres Problem ist die direkte Nutzung einer Video-Sequenz, die mit der Kamera des mobilen Endgerätes aufgenommen wird. Mit der aktuellen Hardware ist es unmöglich, den RGB-Datenstrom der Kamera abzugreifen. Entweder wird der Datenstrom direkt an einen hardwarebasierenden MPEG-Encoder geleitet oder am Display dargestellt (z.B. Sucher der Kamera). Des Weiteren ist ein MPEG-Datenstrom nicht für Bildverarbeitungsoperationen geeignet, da ein vollständiges Bild vorhanden sein muss. Bei einem MPEG-Datenstrom sind immer nur die Key-Frames vollständig, die nachfolgenden Bilder werden durch Veränderungen gegenüber des vorangegangenen Key-Frames beschrieben. Zusätzlich wird eine JPEG-Komprimierung verwendet - diese würde zu ungewollten Fragmenten bei der Segmentierung führen. Daher wird zur Simulation eines Videostroms eine Reihe von Einzelbildern im png-Format verwendet. Dieses Bildformat ist wesentlich besser geeignet, da hier keine Verluste bei der Kompression auftreten.

Für jedes Gesichtslokalisierungsverfahren gibt es einige Bilder, die zu besonders schlechten, und andere, die zu besonders guten Ergebnissen führen. Wenn als entscheidendes Argument der Farbwinkel gewählt wird, so ist es beinahe unmöglich eine gute Segmentierung durchzuführen, wenn der Hintergrund einen hautfarbähnlichen Ton hat. Ebenfalls problematisch sind Bilder, die zu hell oder zu dunkel aufgenommen wurden. Bei den eben genannten Szenarien hat das Ellipse Fit Verfahren weniger Probleme. Sobald jedoch keine eindeutige Ellipsenkante oder mehrere ähnlich große Ellipsen gefunden werden, zeigen sich die bekannten Grenzen des Ellipse-Fit Verfahren.
Ein ideales Bild ist optimal belichtet, der Hintergrund ist einfarbig und bietet farblich einen deutlichen Kontrast zum restliche Bild. Um ein robusteres Resultat zu erhalten, werden beide Verfahren kombiniert. Auf aktuellen mobilen Endgeräten ist die Performance jedoch noch zu gering, um dies zu realisieren.

Das Resultat aus der Untersuchung von drei unterschiedlichen Einzelbild-Sequenzen (40 Einzelbilder) wird in Tabelle 5.5 dargestellt. Im ersten Schritt wird die Bounding-Box des automatisch gefundenen Gesichts mit einer von Hand erstellten Gesichts-Bounding-Box verglichen. Dabei wird die Differenz über die Abweichung in Pixeln angegeben. Bei einer Abweichung von mehr als 6 Pixeln ist die Box nicht zu verwenden.

	≤ 3 Pixel	> 3 && ≤ 6 Pixel	> 6 Pixel
Sequenz 1	11	23	6
Sequenz 2	17	16	7
Sequenz 3	14	21	5

Tabelle 5.5: Abweichung der Gesichtslokalisierung

Bei den Augen wird der Radius des linken und rechten Auges sowie der Mittelpunkt betrachtet. In Tabelle 5.6 wird die Abweichung beider Radien und Mittelpunkte über 40 Frames dargestellt.

		$3 \leq$ Pixel	> 3 && ≤ 6 Pixel	> 6 Pixel
Sequenz 1	Radius	68	9	3
	Mittelpunkt	64	13	3
Sequenz 2	Radius	45	23	2
	Mittelpunkt	56	20	4
Sequenz 3	Radius	59	17	5
	Mittelpunkt	65	9	6

Tabelle 5.6: Abweichung der Augenparameter

Im Mundbereich sind pro Frame zehn Punkte definiert. In der Tabelle 5.7 wird die Abweichung der 10 Punkte auf 40 Frames verglichen. Zusammenfassend kann ge-

	≤ 3 Pixel	> 3 && ≤ 6 Pixel	> 6 Pixel
Sequenz 1	253	123	24
Sequenz 2	307	44	49
Sequenz 3	280	83	37

Tabelle 5.7: Abweichung der Mundparameter

sagt werden, dass die Gesichts-Bounding-Box in 10,0% der Fälle nicht brauchbar ist. Dies hängt meist mit einer schlechten Beleuchtung oder mit weiteren Hautsegmenten im Bild zusammen. In 8,3% der Fälle wurde der Radius des Auges falsch ermittelt. Der Mittelpunkt zeigte bei 10,8% der Fälle eine Abweichung von mehr als sechs Pixeln. Dies ist meistens auf eine zu geringe Auflösung des Kopfes zurückzuführen. Die Mundpunkte zeigten 9,1% der Fälle eine Abweichnung von über sechs Pixeln. In diesen Fällen sind die Punkte unbrauchbar. Die fehlerhaften Punkte sind auf eine schlechte Segmentierung aufgrund von Bärten oder unzureichender Beleuchtung zurückzuführen.

Kapitel 6

Avatar Animation

In diesem Kapitel wird beschrieben, wie die aus Kapitel 5 erhaltenen Gesichtsparameter zur Animation verwendet werden. Es erfolgt einerseits eine Beschreibung des Gesichtsmodells und andererseits wird auf die MPEG-4 basierte Animation eingegangen.

6.1 Facial Animation in MPEG-4

MPEG-4 ist ein ISO-Standard, definiert von der Moving Picture Expert Group (MPEG). MPEG-4 entstand durch den Wunsch einen Multimedia-Standard zu haben, der kleinere Bitraten als die bestehenden Standards zur Videoübertragung, MPEG-1 und MPEG-2, erzeugt. Es wurden Überlegungen getätigt, wie dies zu realisieren sei. So stand fest, dass der zu Grunde liegende Video-Encoder verbessert werden müsse. Weiterhin griff man Konzepte von Forchheimer (vgl. auch im Weiteren [33]) aus den 80er Jahren auf, die sich mit einer modellbasierten Bildbeschreibung befassen. Durch die Verwendung eines parametrisierbaren und animierbaren Gesichtsmodells konnten die zu übertragenden Daten erheblich verringert werden. So wurde die Gesichtsanimation in den MPEG-4-Standard integriert. MPEG-4 spezifiziert allerdings nur das Übertragungsformat und die Bedeutung der übertragenen Daten. Die Gewinnung der Parameter, aus Bildmaterial und vor allem synthetischen Anwendungen und deren Interpretation, bleibt den Entwicklern der speziellen Anwendungen überlassen. Ziel war es einerseits, niedrige Datenübertragungsraten und andererseits Anwenderinteraktionen zu ermöglichen.
Der MPEG-4 Standard beschreibt ein parametrisiertes Modell sowohl für das Gesicht als auch für den gesamten menschlichen Körper, mit dem es möglich ist Facial beziehungsweise Body- Animation durchzuführen (vgl. [32]). Im Folgenden wird nur die Facial Animation betrachtet.
Das 3D-Modell des Kopfes/Gesichts in MPEG-4 besteht aus Polygonen, welche

auch Attribute wie beispielsweise Farben oder Texturen haben können (vgl. auch im Weiteren [31] S.5). Die Parameter verändern die Feature Points (FP) und somit das Modell des Kopfes. Durch die Veränderung wird der Kopf animiert. Dabei werden die Eckpunkte der Polygone indirekt durch die Parameter in ihrer Lage verändert. Die Verschiebung der Punkte ist abhängig von den Werten der Parameter. So bedeutet zum Beispiel ein großer Wert für die Bewegung der Mundwinkel-Parameter ein breites Lächeln. Insgesamt gibt es 84 Feature Points im neutralen Modell des menschlichen Gesichts von MPEG-4 (vgl. [30] S. 10). Die Feature Points dienen hauptsächlich als räumliche Referenz für die Facial Animation Parameters (FAPs). Durch eine Lageveränderung der Feature Points können alle möglichen Gesichtsausdrücke und Animationen darstellen werden.

6.2 Gesichtsmodell

Alle Menschen haben eine unterschiedliche Form des Gesichtes und diese haben ebenso verschiedene Anordnungen der Gesichtsmerkmale wie beispielsweise Augen, Mund und Nase. Es gibt Menschen mit einer besonders hohen Stirn, besonders eng angeordneten Augen - wenn man jedoch von diesen Ausnahmen absieht, kann man einen durchschnittliches Gesicht ermitteln, welches auf den größten Teil der Menschen annähernd passt. In Abbildung 6.1 wird ein Drahtgittermodell eines solchen Gesichtes dargestellt. Aufgrund der eingeschränkten Leistungsfähigkeit der mobilen Endgeräte ist das Modell auf 60 Knoten beschränkt. Das Modell wurde von Thomas Riegel (vgl. [28]) entwickelt und zur Verwendung in dieser Diplomarbeit zur Verfügung gestellt. Die rot markierten Punkte werden durch Bilderkennungsverfahren erkannt. Die grünen Koordinaten der grünen Punkte werden in Abhängigkeit der Bounding-Box des Mundes sowie den erkannten Punkten berechnet. Die restlichen Punkte sind einer Tabelle hinterlegt. In der Tabelle 6.1 werden die Gesichtsknoten der Mesh-Punkte normalisiert gespeichert. Dabei wird von einer Bounding-Box mit der Seitelänge 100 ausgegangen. Der Ursprung der Punkte liegt oben links in der BB. Somit werden die Punkte des Meshes entsprechend der Größe der Bounding-Box platziert.
Die Triangulierung des Meshes ist in einer weiteren Tabelle (vgl. 6.2) festgelegt. Insgesamt besteht das Mesh aus 94 Dreiecken.

Knoten	x-Koordinate	y-Koordinate
12	28	59
13	51	63
14	58	65

Tabelle 6.1: Normalisierte Gesichtsknotentabelle

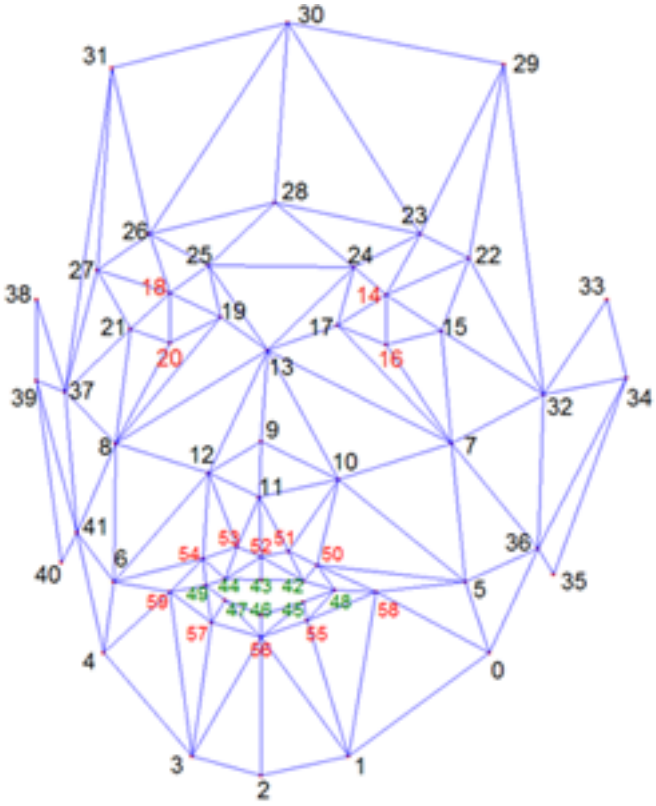

Abbildung 6.1: Drahtgittermodell des Kopfes (vgl. [28])

Die Abbildung 6.2 zeigt ein texturiertes Kopfmodell, dem enstprechend ein Drahtgittermodell (Abb. 6.1) modelliert ist. Das texturierte Modell wurde ebenfalls von Thomas Riegel (vgl. [28]) für diese Diplomarbeit zur Verfügung gestellt. Beide Modelle zeigen leicht unterschiedliche Anpassungen. Dies kommt daher, dass die Gesichtsknotentabelle auf jedes Gesicht skaliert werden kann. Das Mesh in Abbildung 6.2 zeigt, dass gegenüber dem Ausgangsmesh in Abbildung 6.1 zum Beispiel die Ohren weggelassen wurden. Dies wird dadurch realisiert, dass die Punkte 38 und 39 auf Punkt 37 gelegt werden. Punkt 40 wird auf Punkt 41 gelegt. Auf der rechten Seite findet die Anpassung analog statt.

6.3 Face Animation

In diesem Abschnitt wird auf die Animation des Avatars durch die in Kapitel 5 ermittelten Parameter eingangen. Vorneweg bleibt zu sagen, dass die Animation auf einem mobilen Endgerät aufgrund der Restriktionen der Hardware sowie der

Knoten 1	Knoten 2	Knoten 3
0	1	58
1	58	55
1	55	56

Tabelle 6.2: Triangulierung des Meshes

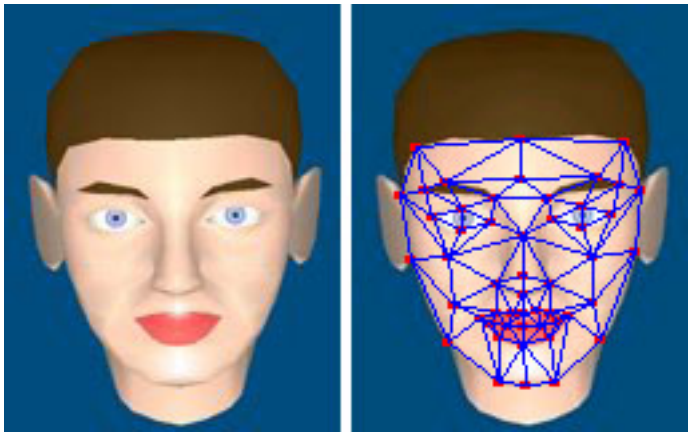

Abbildung 6.2: Texturiertes Kopfmodell (vgl. [28])

J2ME Plattform (vgl. Kapitel 4) nicht realisiert werden konnte. Die Tabelle 6.3 zeigt den Ablauf der Face Animation im Überblick. Zusätzlich zu den ermittelten Parametern erhält das Modul die Bounding-Box des Gesichtes. Die Bounding-Box wird für die Anpassung der normierten Gesichtknoten (vgl. Abb. 6.1) auf die Größe der ermittelten Bounding-Box benötigt. Angenommen die Gesichts-Bounding-Box hat eine Höhe von 127 Pixel und eine Breite von 85 Pixel (vgl. Abb. 5.4) und der Punkt 0 hat die Koordinaten $x = 74$, $y = 78$. Weiterhin gelte, dass die Tabelle mit den normierten Gesichtsknoten einen minimalen x-Wert bei $0,14$ (y-Wert $0,31$) sowie einen maximalen x-Wert von $0,86$ (y-Wert $0,89$) hat. Um die absoluten Bildkoordinaten zu erhalten muss nun für den x-Wert die Gleichung 6.3 und für den y-Wert die Gleichung 6.3 angewendet werden.

$$P1_x = ((((x_{norm} - x_{min}) * 100)/(x_{max} - x_{min})) * img_{width})/100 \qquad (6.1)$$

$$P0_y = ((((y_{norm} - y_{min}) * 100)/(y_{max} - y_{min})) * img_{height})/100 \qquad (6.2)$$

Die Klammerung ist notwendig, da unter J2ME keine Floating-Point Berechnungen unterstützt werden. Der angegebene Wert wird auf eine Ganzzahl abgerundet. Sobald man die absoluten Koordinaten der einzelnen Punkte hat, kann geprüft

Face Animation
Eingabe: Face Animation Parameter (FAP) und Gesichts-BB: *1. FAP der Merkmalsextraktion* *2. Restliche FAP aus Gesichtsknotentabelle* ↓ Skalierung der FAP auf absolute Koordinaten der Gesichts-BB ↓ Überprüfen, welche FAPs sich geändert haben Übertragen der geänderten FAPs ↓ Face Definition Table: Nachschlagen, welche Punkte sich durch geänderte FAPs in Folge ändern ↓ Rendering neuer Frame ↓ *Ausgabe: neues Bild*

Tabelle 6.3: Face Animation

werden, welche der Punkte sich zum vorherigen Frame überhaupt verändert haben. Es werden nur die Punkte erneut übertragen, die sich geändert haben. So lassen sich weitere Ressourcen während der Übertragung einsparen. Die Abbil-

```
#Frame 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0

#Frame 1
0 0 1 1 1 1 1 1 1 1 1 1 0 0
11 104 -61 -92 -47 -47 -11-11 -26 -26 26 26 111
```

Abbildung 6.3: Auszug aus den Face Animation Parameter

dung 6.3 zeigt einen Auszug aus den Face Animation Parameter. Die erste Zeile gibt immer den Frame an. In der zweiten Zeile gibt es für jeden der 59 Meshknoten ein Flag - wenn das Flag „0“ ist, dann wurde dieser Punkt im Mesh nicht geändert, falls das Flag jedoch auf „1“ gesetzt wurde gibt es eine Änderung. In der dritten Zeile stehen die Änderungswerte. Wenn ein Punkt keine Veränderung erfahren hat, so wird auch kein Wert eingetragen. Dieser Datenstrom würde auch frameweise bei der Verbindung zu einem anderen Mobiltelefon übertragen werden. Die Face Definition Table beschreibt die Art der Veränderung bei den FAPs (vgl. Abb. 6.4). Die `fapID` gibt den Punkt im Mesh an, von dem die Veränderung

```
FaceDefTables{
        fapID               54
        highLevelSelect         0
        faceDefMesh[
            FaceDefMesh{
                intervalBorders[-600, 0, 600]
                coordIndex[53, 54, 57, 59]
                displacements[-237 0, -237 0,
                        -497 0, -497 0,
                        -497 0, -497 0,
                        -993 0, -993 0]
            }
        ]
    }
FaceDefTables{
        fapID               57
        highLevelSelect         0
        faceDefMesh[
            FaceDefMesh{
                intervalBorders[-1860, 0, 1860]
                coordIndex[55]
                displacements[0 -999, 0 -999]
            }
        ]
    }
```

Abbildung 6.4: Auszug aus der Face Definition Table

ausgeht. Die `intervalBoarders` geben die Schrittweite der Bewegungsänderung an. Abhängig von der Stärke und Richtung der Bewegung wird der Wert für einen FAP berechnet.
Je nachdem in welchem Intervall sich der berechnete Wert befindet, findet eine Verschiebung der unter `coordIndex` benannten Punkte statt. Die Stärke der Verschiebung ist unter dem Punkt `displacements` zu finden. Hier wird die Verschiebung für jedes Intervall angebeben.
Die Ergebnisse aus der `FaceDefTable` dienen der Neuberechnung des nächsten Frames. Der PC-Avatar Player wurde für diese Arbeit von Siemens CT, München Perlach zur Verfügung gestellt. Dabei erfolgt das Rendering im Player gemäß den Regeln des MPEG-4 Standards (vgl. [38]). Im letzten Schritt der Face Animation wird der neue Frame angezeigt.

6.4 Implementierung

Zur Realisierung der Mesh-Platzierung wurden zwei verschiedene Verfahren evaluiert. Im ersten Verfahren wird das Mesh nur anhand der Größe der gefundenen Gesichts-Bounding-Box plaziert. Dabei werden die Punkte aus dem Augen und Mundbereich, die per Bilderkennung gefunden wurden, nicht berücksichtigt. Dieses Verfahren ist sehr schnell, jedoch taucht das Problem auf, dass das normierte

Gesicht nur für sehr wenige Gesichter exakt passt. Hieraus wurde auch das zweite Verfahren abgeleitet.
Zuerst werden die Punkte im Augenbereich und im Mundbereich erkannt. Auf Basis von diesen Koordinaten werden die restlichen Punkte des Meshes positioniert. Bei den Skalierungsangaben wird immer davon ausgegangen, dass links oben der Ursprung liegt. Die Koordinaten der Punkte 15, 17, 19, 22, 23, 24, 25, 26, 27, 28, 29, 30 und 31 werden anhand des y-Wertes von Punkt 20 vertikal auf die Größe angepasst. Die horizontale Anpassung erfolgt ausgehend von den x-Werten der Punkte 16 und 20.
Im nächsten Schritt wird geprüft, welche Werte sich gegenüber dem vorherigen Frame verändert haben. Falls die Werte nicht direkt an ein Animationsmodul weitergegeben werden, können diese in ein FAP-File gespeichert werden (vgl. Abb. 6.3). Um MPEG-4 konform zu sein, muss für jeden Punkt, der sich geändert hat, ein Änderungswert berechnet werden. Der Wert ist abhängig von der Verschiebung des Punktes in x und y-Richtung.
Um den nächsten Animationsframe zu erzeugen, wird in der Face Definition Table nachgeschlagen, wie sich der Punkt verändert hat, und welche anderen Punkte davon ebenfalls betroffen sind. Gegenüber der MPEG-4 Definition mussten alle Fließkommawerte in Ganzzahlen skaliert werden, um die Funktionalität unter J2ME zu ermöglichen.

6.5 Ergebnisse

Der Ansatz zeigt, dass es generell möglich ist, die erkannte Mimik auf einen Avatar zu übertragen und diesen zu animieren. Innerhalb der Diplomarbeit konnte die Avatar Animation nicht auf einem mobilen Endgerät durchgeführt werden. Die Ursachen liegen in der Performance des Endgerätes der begrenzten Größe des Java-Heaps und der *.jar-Datei. Die zu geringe Größe der *.jar-Datei hat es verhindert, dass gleichzeitig das Avatar-Modell und die Face Definition Table geladen werden konnten.
Die Skalierung des Meshes funktioniert sehr gut. Zu Testzwecken wurden schrittweise kleinere Bilder verwendet. Die Mesh-Platzierung war ab einer Bildhöhe von 25 Pixel problematisch, da durch Integer-Rundungen die Genauigkeit verloren geht und sich somit Abrundungen bemerkbar machen.
Die Mesh-Platzierung wurde an 10 verschiedenen Gesichter getestet. Die Skalierung ist durchgängig sehr passend. Es gibt jedoch Sonderfälle, wie beispielsweise eine besonders große Nase. Hier passt dann das Mesh nicht genau. Dies ist aber nicht weiter problematisch, da bei der Avatar-Animation keine größeren Unterschiede sichtbar sind.
Für die Animation in Echtzeit werden auf dem mobilen Endgerät mindestens 8-10

Frames pro Sekunde (Fps) benötigt. Im Avatar-Player auf einem Desktop-Rechner kann diese Rate problemlos übertroffen werden. Sobald man dort aber die zur Verfügung gestellte Performance drosselt (auf Niveau eines Mobiltelefons), ist keine fließende Animation mehr möglich. Zusätzlich ist zu erwähnen, dass die Animationsparameter vorher extrahiert wurden und durch eine Face Animation Table bereit gestellt werden.

Kapitel 7

Modell einer Face-to-Face Applikation

7.1 Applikation

Es gibt eine Vielzahl von Anwendungen auf dem Markt die bei verschiedenen Zielgruppen auf Interesse stoßen. Daher ist vor der Positionierung einer Anwendung eine Markterhebung notwendig, die zeigt, welche Zielgruppe Interesse an einer Anwendung hat und was diese Gruppe bereit ist dafür zu bezahlen.
In diesem Abschnitt wird ein Vorschlag unterbreitet, welche Zielgruppen an einer Face-to-Face Applikation Interesse haben könnten, und wie die Applikation entsprechend positioniert werden könnte. Zunächst sind die technikbegeisterten Nutzer zu nennen. Dieser Benutzergruppe folgt den neuesten Trends, und probiert gerne neue Innovationen aus. Dabei sind bis zu einem gewissen Punkt die Kosten unabhängig von der gebotene Leistung. Es bietet sich ein Tarif mit inklusiven Datenvolumen oder sogar eine Datenflatrate an.
Die zweite Benutzergruppe ist im jüngeren Bereich zu finden. Diese Gruppe möchte neue Features testen, ohne sich direkt an einen Tarif zu binden. Daher ist ein Tarif ohne Grundgebühr ideal, der nur nach aufkommenden Datenvolumen abrechnet.

7.2 Technische Vorraussetzung

Endgerät

Für eine rein J2ME-basierte Anwendung muss das Gerät eine Java-Unterstützung haben. Es wird eine Java-Version, die mindestens MIDP 2.0 (siehe Kapitel 4.2) und die CLDC 1.0 (siehe Kapitel 4.2) unterstützt, benötigt. Die in MIDP 2.0

enthaltene Multimedia APIs sind für die Java Applikation erforderlich, da ein Face-to-Face Decoder auf die JSR-000184 für das Rendering des Avatars zugreift. Zusätzlich wird eine Kamera mit einer Auflösung von mindestens 192 x 144 Pixel benötigt. Gegenüber den aktuellen Möglichkeiten der J2ME-Plattform muss noch der Zugriff auf die Kameradaten per Java ermöglicht werden. Das Gerät muss ebenfalls den General Packet Radio Service (GPRS) unterstützen, welcher die Klasse 5,6,7,9 oder größer aufweist, da sonst nicht genügend Sende- und Empfangsschlitze angesprochen werden können.

Datenverbindung

Für eine erfolgreiche Face-to-Face Kommunikation stehen momentan das Global System for Mobile Communications (GSM) sowie das Universal Mobile Telecommunications System (UMTS) zur Verfügung. In beiden Netzen wird für die Übertragung eine Verbindung benötigt, die Sprache und Daten parallel und synchron überträgt. Das Voice-over-IP (VoIP) Verfahren erscheint am geeignetsten, da Sprache und Daten synchron übertragen werden können. Dabei wird eine Netzkapazität von ungefähr 10 kBit/s für die Face-to-Face Applikation benötigt.
Das High Speed Circuit Swiched Data (HSCSD) Verfahren ist gegenüber GPRS benachteiligt, da einerseits sehr wenige Endgeräte dieses Verfahren unterstützen und andererseits auch nur E-plus und Vodafone dies anbieten.
GPRS eignet sich für eine Face-to-Face Applikation am besten, da der Benutzer am Server eingeloggt sein muss, um einen Face-to-Face Anruf empfangen zu können. GPRS rechnet nach dem Datenvolumen und nicht nach der Zeit ab.
Die theoretisch mögliche maximale Datenübertragung nach den aktuell durch die Netzbetreiber verwendeten Codecs beträgt bei GPRS 53,6 kBit/s (vgl. auch im Weiteren [24]). Im praktischen Einsatz sind 10 kBit/s in beide Richungen (Senden und Empfangen) auch unter schlechten Bedingungen möglich. GPRS ist abhängig von der Auslastung des Netzwerkes. Dies kann zwischenzeitlich zu einer gringeren Datendurchsatzquote führen. GPRS unterstützt zur Zeit kein Quality of Service (QoS), wodurch Verbindungsabbrüche und mangelnde Datendurchsätze entstehen können.

Server

Die Kommunikation der Face-to-Face Anwendung läuft über einen Server, da im GSM-Netz keine Punkt-zu-Punkt Verbindungen möglich sind. Der Server leitet zunächst die Anfrage zur Datenkommunikation eines Teilnehmers an den Empfänger weiter. Dazu muss geprüft werden, ob der Empfänger den Face-to-Face Decoder installiert hat. Wenn dies nicht der Fall ist, so wird die Kompatibilität der Hardware geprüft und gegebenenfalls wird der Encoder zum kostenlosen

Download angeboten. Falls das Gerät des Empfängers nicht kompatibel ist, wird dies dem Absender mitgeteilt und es wird eine reine Sprachverbindung aufgebaut. Darüber hinaus läuft die Registrierung von neuen Benutzern über den Server ab. Sobald ein Benutzer erfolgreich registriert wurde, wird diesem die Software durch den Server übermittelt. Wenn die Installation abgeschlossen wurde, muss keine weitere Konfiguration durch den Benutzer vorgenommen werden. Ein für den Netzbetreiber wichtiger Punkt ist das Abrechnungsverfahren. Der Server ermittelt für alle Verbindungen das verbrauchte Datenvolumen. Anhand des durch den Benutzer gewählten Tarifs kann so eine sehr einfache Abrechnung erfolgen.

7.3 Use-Cases

Die Vorbedingung bei allen Use-Cases ist, dass sowohl der Sender als auch der Empfänger an einem Kommunikations-Server des Dienstbetreibers registriert sind und dass der Empfänger die Face-to-Face Software auf seinem Endgerät installiert hat. Probleme können auftreten, wenn der Empfänger die Applikation deinstalliert oder auf ein Endgerät wechselt, das diese Leistungsmerkmale nicht zeigt.

7.3.1 Use-Case: Empfänger ist zum Server verbunden

In diesem Szenario sind sowohl Sender als auch Empfänger mittels einer GPRS-Verbindung am Kommunikationsserver des Dienstbetreibers aktiv angemeldet und haben die Face-to-Face Kommunikationssoftware gestartet. Der Sender schickt eine Nachricht an den Server, dass er eine Face-to-Face Verbindung zu einem Empfänger aufbauen möchte. Der Server prüft vorab, ob der Empfänger für diesen Dienst registriert ist. Falls ja, wird der Status „Empfänger ist mit dem Server verbunden“ abgerufen. Somit kann der Server die Verbindung vom Sender zum Empfänger herstellen.

7.3.2 Use-Case: Empfänger ist nicht zum Server verbunden

Der Server ermittelt wieder den Status, „Empfänger ist nicht zum Server verbunden“, des Empfängers. Wenn keine Verbindung zum Server vorhanden ist, kann dies mehrere Gründe haben: Entweder hat der Empfänger kein Netz, das Gerät ist ausgeschaltet oder der Empfänger ist im Netz eingebucht und hat die Applikation nicht gestartet. Falls der Empfänger nicht im Netz eingebucht ist, sendet der Server eine Hinweis-SMS, dass versucht wurde eine Face-to-Face Verbindung aufzubauen. Wenn der Benutzer im Netz eingebucht ist, schickt der Server eine Push-SMS, die den Empfänger darauf hinweist, dass eine Face-to-Face Verbindung eingeht, und die entsprechende Applikation gestartet werden soll. Durch

das Starten der Applikation sendet der Empfänger eine Bestätigung an den Server. Dieser kann dann die Verbindung aufbauen. Startet der Empfänger die Applikation nicht, bricht der Server nach einem gewissen Time-out den Versuch ab, eine Verbindung aufzubauen.

7.4 Fazit

Letztendlich entscheidet weder der Software-Entwickler noch der Netzbetreiber über den Erfolg einer Anwendung sondern der Kunde. Daher sollten die Funktionalitäten der Anwendung dem Kunden ausgiebig erläutert werden und anfangs zu einem günstigen Einstiegspreis am Markt positioniert werden. Der Kunde möchte ein vorkonfiguriertes System erhalten, bei dem keine weiteren Anpassungen notwendig sind. Die Qualität der Übertragung von Sprache und Daten sowie der Preis werden ein ausschlaggebendes Kriterium für den Erfolg einer solchen Anwendung am Markt sein.

Kapitel 8

Fazit

8.1 Zusammenfassung

Das Ziel dieser Arbeit war die Erkennung von Gesichtsparametern mittels der Kamera eines mobilen Endgerätes und die Übertragung der Parameter auf einen Avatar zur Simulation einer Face-to-Face Kommunikation unter Verwendung von J2ME. Hierzu wurde zuerst eine Marktübersicht über ähnliche Systeme erstellt und daraus die Anforderungsanalyse für diese Arbeit abgeleitet. Im nächsten Schritt wurde J2ME mit den Konfigurationen, wichtigen Profilen und optionalen Paketen vorgestellt und mit Symbian als Plattform für mobile Software Applikationen verglichen. J2ME hat den Vorteil der Plattformunabhängigkeit, jedoch sind Einschnitte bei der Performance zu machen. Zukünftige CLDCs (V.1.1) zeigen einen deutlich erweiterten Funktionsumfang wie beispielsweise Fließkommarechnung.
Die Evaluierung der Verfahren zur Gesichtserkennung ergab, dass das „Ellipse-Fit“ und die Segmentierung anhand der Hautfarbinformation die praktikabelsten Möglichkeiten für die Gesichtslokalisierung auf einem mobilen Endgerät sind. Daher wurden diese beiden Verfahren prototypisch implementiert. Aufgrund der besseren Performance wurde die Segmentierung anhand der Hautfarbe für das Gesamtprogramm gewählt. Für die Erkennung der Augen wurde ein Verfahren auf Basis der Hough-Transformation implementiert. Der Mund wurde wie das Gesicht anhand der Farbinformation der Lippen erkannt.
Für die Animation wurden die Parameter durch die in Kapitel 5 verwendeten Methoden zur Verfügung gestellt. Aufgrund von technischen Problemen wie zu kleinem Java-Heap, zu kleiner maximalen *.jar-Dateigröße und zu geringen CPU-Performance konnte der Avatar-Player nicht wunschgemäß auf einem mobilen Endgerät realisiert werden. Die Parameter der extrahierten Gesichts-Features konnten frameweise zwischengespeichert werden. Für eine Face-to-Face

Kommunikations- Anwendung können diese Daten mit Voice over IP parallel zur Sprache übertragen werden. Die Parameter dienen als Eingabe des Avatar-Players. Somit ist eine MPEG-4 basierte Animation des Avatars möglich.
Im letzten Kapitel wurde die Funktionsweise einer prototypischen Anwendung beschrieben. Es wurden insbesondere notwendigen Handlungsschritte hinsichtlich Hard- und Software aufgezeigt, die für den allgemeinen Betrieb noch erforderlich sind.

8.2 Bewertung

In der nachfolgenden Bewertung des gewählten Ansatzes werden Verbesserungsmöglichkeiten und Probleme aufgezeigt. Die Prämisse dafür ist jedoch, dass sowohl Hard- als auch Software fehlerfrei funktionieren und die Hardwareressourcen ausreichend sind.
Prinzipiell wird festgestellt, dass dieser Ansatz zielführend ist. In dieser Arbeit wurde nur die Erkennung von Augen und Mund sowie die Animation des Avatars auf dem PC-Player realisiert. Die alleinige Animation von Mund und Augen wirkt sich negativ auf das visuelle Gesamtresultat aus. Um die Qualität auf ein Videokonferenz-ähnliches Niveau zu heben, müssten alle definierten Mesh-Punkte animiert werden und zusätzlich noch Bewegungen des gesamten Kopfes realisiert werden.
Bei falsch erkannten Punkten kommt es zu Fehlern in der Darstellung. Hierbei verschieben sich ganze Vertices und weichen vom Original ab. Durch eine gegenseitige Korrektur beziehungsweise Überprüfung der Punktkoordianten könnten diese Fehler vermieden werden.
Ein weiteres, mit durchaus hoher Wahrscheinlichkeit auftretendes Problem ist die fehlerhafte oder nicht erfolgreiche Erkennung eines Gesichtes. Wenn die Anwendung kein Gesicht oder fehlerhafte Daten erkennt und somit keine Parameter für die Animation liefert, führt dies zu kurzen Pausen mit anschließenden über die Verbindungszeit andauerndem Fehlverhalten. Eine Bewegungsvorhersage könnte die Pausen unterbinden und durch eine Keyframe-Interpoltation könnten die anschließenden Sprünge unterbunden werden. Ostermann (vgl. [37]) beschreibt eine Interpolationsfunktionen, die eine maximale Bewegung eines Mesh-Knotens pro Frame festlegt.
Die Gesichtslokalisierung und Feature Extraktion (vgl. Kapitel 5.6) funktionieren gut - im Gesamtschnitt wird in 90,45% der Fälle ein gutes Ergebnis geliefert. Insbesondere bei optimierten Bildsequenzen (ausreichende Beleuchtung und weisser Hintergrund) kann mit einem noch besseren Ergebnis gerechnet werden.
Der gesamte Prototyp wurde unter J2ME erstellt und kann noch ausgebaut und optimiert werden. Das Kapitel 8.3 zeigt weiterführende Verbesserungs- und Opti-

mierungsvorschläge.

8.3 Ausblick

Während der Erstellung dieser Arbeit wurden Sachverhalte sichtbar und entstanden Ideen, die aufgrund der Komplexität dieser Arbeit nicht vollständig weiterverfolgt werden konnten. Folgende Themen, die innerhalb dieser Arbeit nur kurz oder gar nicht behandelt wurden, werden hier ausdrücklich zur weiteren wissenschaftlichen und/oder praktischen Vertiefung empfohlen:

- Die Bewertung der Leistungsfähigkeit von mobilen Endgeräten, insbesondere von Mobiltelefonen muss stetig neu durchgeführt werden. Dabei muss einerseits die Hardware betrachtet werden, andererseits ist die J2ME Plattform zu beobachten. Wie bereits in Kapitel 1.1 erwähnt, ist wegen der bestehenden und fortschreitenden Fragmentierung des Endgerätemarkets die Standardisierung über Modelle und Anbieter hinaus ein Schlüsselerfolgsfaktor für neue Anwendungen. Daher sind für die Hardwarefunktionen der nächsten Generation der Mobiltelefone bereits eine erhebliche Leistungssteigerung zu erwarten. Ebenfalls sollten Funktionen wie der Kamerazugriff von der Hardware entkoppelt werden.

- Momentan ist für die Darstellung der FAP ein 2D-Modell die Grundlage. Eine verbesserte Darstellung erfolgt mittels eines 3D-Modells. Hier können dann Zähne, Zunge usw. modelliert werden. Dadurch erscheint der Avatar bei Mundbewegungen realistischer. Zusätzlich kann eine Drehung und Neigung des Kopfes dargestellt werden. (Vgl. hierzu [10])

- Das Tracking in dieser Arbeit beschränkt sich auf einen reduzierten Suchraum. Hier sind Verbesserungen durch z.B. Differenzbilder und Bewegungsvorhersage möglich. (Vgl. hierzu [16], [15], [22] und [26]).

- Aktuell werden nur zwei Knoten im jeweiligen Augenbereich (links und rechts) und zehn Punkte im Mundbereich per Bilderkennung erkannt. Um das Mesh noch besser an das jeweilige Gesicht anzupassen, ist eine Erkennung weiterer Punkte notwendig. Dadurch wirkt die gesamte Mimik noch realistischer.

- Die J2ME Plattform wird erweitert. Einerseits werden weitere Klassen (z.B. Bildverarbeitungsoperationen) als API zur Verfügung gestellt und andererseits werden über die CLDC 1.1 Profile (vgl. Kapitel 4.1) weitere Datentypen, wie z.B. Float und Double ermöglicht. Hierzu ist ein Vergleich mit

dem aktuellen System und die entsprechende Anpassung des Quellcodes von großem Interesse, da ein deutlicher Performance Gewinn erwartet wird.

- Innerhalb dieser Arbeit wurde zwar ein Avatar animiert, jedoch nicht unter J2ME. Die Untersuchungen dieser Arbeit haben ergeben, dass die Heap-Größe unter J2ME sowie die *.jar-Dateien Größe noch nicht ausreichend sind. Zusätzlich bieten die Endgeräte noch nicht genügend Performance. Daher wurde ein Avatar-Player für einen PC von Siemens CT, München genutzt. Auf diesem Player konnten die extrahierten Mimik-Parameter interpretiert werden. Ein weiterer Schritt wäre die bestehende Konfiguration zu verschlanken, um einen Player auf der mobilen Java Plattform zu integrieren.

- Das in dieser Arbeit verwendete Gesichts-Mesh besteht nur aus 60 Knoten. Laut dem MPEG-4 Standard können optional noch weitere Knoten eingefügt werden (vgl. [36]). Entsprechend erweiterbar oder zumindest tolerant muss eine auf dieser Arbeit aufbauende Anwendung realisiert werden.

- Individuelle Avatarprofile sind im Rahmen des Personalisierungs- und Individualisierungstrends eine kurzfristig realisierbare Erweiterung. Hierzu wird zuerst ein Bild vom Gesprächspartner übertragen , und dann als Textur für den Avatar herangezogen (vgl. [34]).

- Für die Bilderkennung ist es teilweise problematisch, wenn das Eingangsbild nicht der idealisierten Annahme entspricht. Daher müssen Abweichungen wie beispielsweise Brille, Bart und dunkle Hautfarben explizit ergänzt werden.

- Bei den Eingabebildern wird von einem Bild ausgegangen, das genau ein Gesicht enthält. Eine Gesichtsdetektion könnte das Gesamtverfahren verbessern. Hierbei würde automatisch geprüft werden, ob überhaupt ein Gesicht im Bild vorhanden ist.

Über die in Kapitel 1.1 bereits erwähnte Dynamik im mobilen Dienstleistungs- und Endgerätemarkt werden in Zukunft Dienste in den gleichzeitig weiter bestehenden Netzen harmonisiert. Dies wird insbesondere durch die Verbreitung von IP-basierten Diensten im Festnetz unterstützt (vgl. [29]). Ein weiteres Indiz ist die Weiterentwicklung originärer Unterhaltungselektronik zu in Heimnetzwerke integrierte intelligente Kommunikationsgeräten. Hierbei wird deutlich, wie unterschiedlichste Leistungsanforderungen zusammenspielen müssen.

Anhang A

Workflow der Gesamtapplikation

Der Workflow der Gesamtapplikaion (vgl. Abb. A.1) zeigt wie die Prozesse Face Finding, Feature Extraction, Face Animation und Face Tracking zusammenhängen und welche Daten über die Schnittstellen ausgetauscht werden. Die Funktionsweise der einzelnen Module ist in den jeweiligen Kapiteln nachzulesen. Als Eingabe dienen *.png Bilder, die aus einer Videosequenz erstellt wurden. Im ersten Schritt findet eine Prüfung statt, ob eine Face-BB von einem vorherigen Frame vorhanden ist. Wenn „ja“, startet direkt die Feature Extraction. Im Fall „nein“, startet das Face Finding Modul und übergibt die Face-BB, Eye-BB und die Mouth-BB an die Feature Extraction. Als Resultat liefert die Feature Extraction die Mesh-Punkte der Augen, die Punkte des Mundes sowie eine aktuelle Bounding-Box des Mundes an das Face Animation Modul. Dort werden anhand der Punkteveränderungen die Daten für den nächsten Frame gerendert, ausgegeben und am Display dargestellt. Gleichzeitig erhält das Tracking Modul die Punkte der Augen die Mund-BB und die Face-BB. Das Face Tracking berechnet die neuen Regionen, in denen die Features Extrahiert werden sollen. Falls die Berechnung erfolgreich war, wird eine neue Eye-BB und Mund-BB an das Feature Extraction Modul übergeben. Dort startet dann der Prozess wieder von vorne. Falls das Tracking nicht erfolgreich verlaufen ist, wird das Face Finding erneut gestartet, um die entsprechenden Bounding-Boxes neu zu erkennen und die Feature Extraction zu starten.

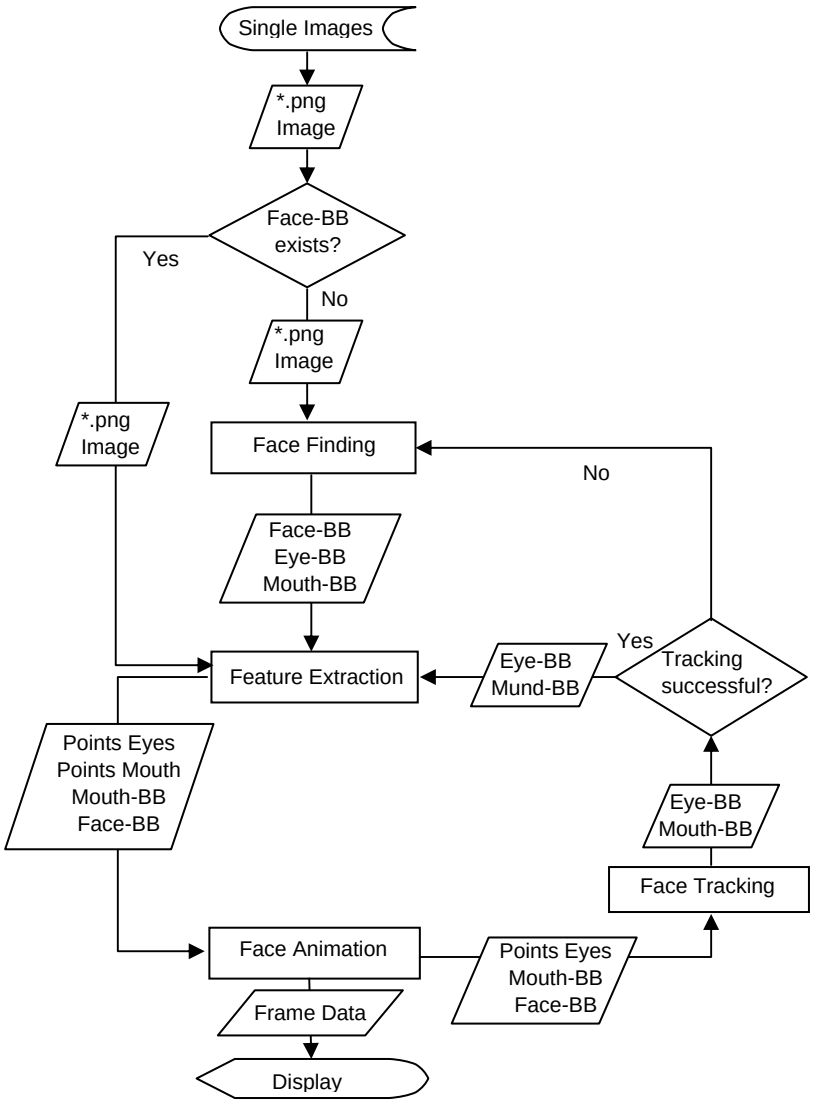

Abbildung A.1: Workflow der gesamten Face-to-Face Applikation

Anhang B

Siemens Mobility Toolkit 2.00b

Das Siemens Mobility Toolkit (SMTK) besteht aus dem SMTK Manager (siehe Abb. B.1) und dem SMTK Emulator Launcher (siehe Abb. B.2). Der SMTK Manager dient zur Verwaltung der Emulatoren der unterschiedlichen Endgeräte. Im SMTK Manager können beliebig viele Profile für andere Siemensgeräte hinzugefügt werden.

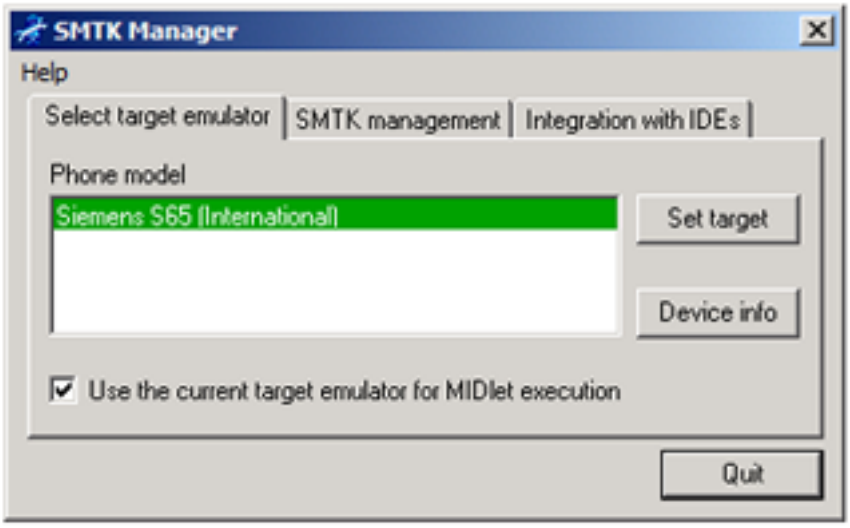

Abbildung B.1: Siemens Mobilty Toolkit - Manager

Der SMTK Emulator Launcher simuliert die J2ME-Laufzeitumgebung eines mobilen Endgerätes. Dabei können über den Menüpunkt $PhoneCommands - StartJavaApplication$ die *.jar Dateien interpretiert werden. Die *.jar Datei enthält die kompilierten *.class Dateien, die Bildsequenzen als *.png Dateien und eine Meta-Datei `Manifest.mf`.

Abbildung B.2: Siemens Mobilty Toolkit - Emulator Launcher

Anhang C

Face Animation Player

Mit dem Face Animation Player können eingegebene Face Animation Parameter in Animation eines Modelles umgesetzt werden. Zuerst muss das Modell und die dazugehörigen FAPs geladen werden (siehe Abb. C.1). Optional können auch noch Audiodaten hinzugefügt werden. Über die Eingabemaske unter dem

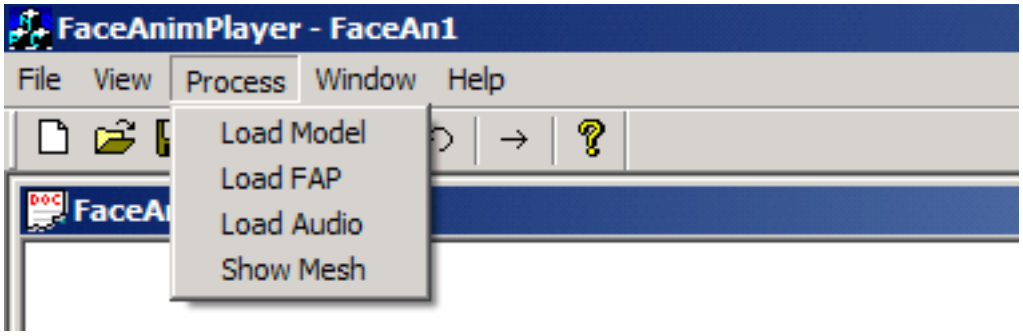

Abbildung C.1: Face Animation Player - Process und Load Face Model

Menüpunkt $Process, LoadModel$ erfolgt das Laden des Modells. Es wird ein *.jpg Bild benötigt, bei dem die Vertex-Daten als Meta-Tags in der Datei gespeichert wurden. Zusätzlich müssen noch drei weitere Dateien geladen werden: Die Datei *.mes enthält die Informationen über das Mesh. Die Face Definition Table ist in der Datei *.fdt gespeichert. Der Teeth Path ist in einer *.dat Datei abgelegt. Sobald alle Parameter geladen und diese auch gültig sind, kann die Animation mittels des $Play - Buttons$ gestartet werden. Über den Menüpunkt $Process, ShowMesh$ kann das Mesh während der Animation ein- und ausgeblendet werden.

Literaturverzeichnis

[1] SeeStorm, *SeeStorm Product Guide*, http://www.seestorm.com/products, Telefon: 02102/8801-32, E-Mail: sales@seestorm.com, SeeStorm, 2004.

[2] Kopp, Herbert, *Bildverarbeitung interaktiv*, B.G. Teubner Stuttgart, 1997.

[3] Rehrmann, Volker, *Foliensammlung Digitale Bildverarbeitung*, Universität Koblenz, 1999.

[4] Toennies, Klaus, *Grundlagen der Bildverarbeitung*, http://isgwww.cs.uni-magdeburg.de/bv/gbv/bvskript.html, Universität Magdeburg, 2004.

[5] Glahn, Kay, *Java ist im mobilen Telefonmarkt bislang eine sehr große Erfolgs-Story*, Java Magazin Nr. 7, 2003.

[6] Christmann, Constantin, *Geradenbestimmung mit der Hough-Transformation*, Proseminar Computervision, Universität Ulm, 2003.

[7] Hill, Jürgen, *Mobilfunker geben Vodafone einen Korb*, http://www.computerwoche.de, Computerwoche, 07.05.2004.

[8] Charamel, *Charamel Software, Tools, mocito*, http://www.charamel.com, Charamel Software GmbH, Richard-Wagner-Str. 39, 50674 Köln, Telefon: 0221/33664-0, E-Mail: info@charamel.com, Charamel Software GmbH, 2004.

[9] Eisert, Peter, *Very Low Bit-Rate Video Coding Using 3-D Models*, Fraunhofer-Institut für Nachrichtentechnik Heinrich-Hertz-Institut, Einsteinufer 37, 10587 Berlin, Telefon: 030/31002-614, E-Mail: eisert@hhi.fraunhofer.de, Shaker Verlag Aachen, Germany, 2001.

[10] Ahlberg, Jörgen, *Model-based Coding - Extraction, Coding and Evaluation of Face Model Parameters*, LTAB, Linköping, 2002.

[11] Wiech, Christian; Naphtali, Tino, *Visualisierung von Emotionen an einem virtuellen 3D-Gesicht (Projekt J3D Emotion)*, XeneX Group, Mehrower Str.23, 15366 Hönow, Telefon: 0176/20806542, E-Mail: naphtali@xenex.de, Humbold Universität Berlin, 2004.

[12] Neven Vision, *Facial Analysis SDK Suite*, http://www.nevenvision.com/products.html, Neven Vision Inc., 1452 2nd St., Santa Monica, CA 90401, Telefon: (310)458-2053, E-Mail: info@nevenvision.com, Neven Vision Inc., 2004.

[13] Symbian, *About Symbian*, http://www.symbian.com, Symbian, 2004.

[14] Weber, Matthias, *Extrahierung und Kodierung von Gesichtsmodellen*, Universität Mannheim, 2002.

[15] Vezhnevets, Vladimir, *Face and Facial Feature Tracking for Natural Human-Computer Interfaces*, Moscow State University, 2002.

[16] Yilmaz, Alper; Shah, Mubarak, *Automatic Feature Detection and Pose Recovery for Faces*, 5th Asian Conference on Computer Vision, 2002.

[17] Guitarte, Jesus; Lukas, Klaus; Frangi, Alejandro, *Low Resource Lip Finding and Tracking Algorithm for Embedded Device*, Siemens AG München, 2004.

[18] Mayer, Stephan, *Automatische Iris-Detektion und Merkmalsextraktion in digitalen Farbbildern des menschlichen Auges*, RWTH Aachen, 2002.

[19] Hengstl, Robert; Maier, Peter; Meerwald, Christof; Stockhammer, Martin, *PS-Bildverarbeitung*, Universität Salzburg, 2000.

[20] Albrecht, Dirk, *Facial Animation Coding*, TU Ilmenau, 2003.

[21] Deutschmann, Christina, *Morphologische Bildverarbeitung*, Universität Ulm, 2003.

[22] Pandzic, Igor; Kalra Prem; Thalmann, Nadia, *Real Time Facial Interaction*, Universität Genf, 1998.

[23] Sun Microsystems Inc., *Connected Limited Device Configuration - Specification Version 1.1*, http://bf.monis.ch/prog/java/midp/files/cldc-11/CLDCSpecification1.1.pdf, 2003.

[24] teltarif.de, *Der General Packet Radio Service - kurz GPRS* , http://www.teltarif.de/i/gprs.html, teltarif.de Onlineverlag GmbH, 2004.

[25] Frey, Jochen, *Bildsegmentierung mit der Topologischen Merkmalskarte in MR- und CT-Bildern*, http://mbi.dkfz-heidelberg.de/mbi/TR/TR49/dipl.bch.html, DKFZ-Heidelberg, 1993.

[26] Kondratyuk, Yevgeniya, *Verfolgung und Segmentierung von Händen in Bildsequenzen zur Unterstützung der 3D-Analyse in einem Echtzeit-Videokonferenzsystem*, TU Berlin, 2003.

[27] Raja, Yogesh; McKenna, Stephen; Gong, Shaogang, *Tracking and segmenting people in varying lighting conditions using colour*, IEEE International Conference on Face and Gesture Recognition, Proc. 3 ICFGR, S.228-233, Japan, 1998.

[28] Riegel, Thomas, *Das lachende Handy*, Siemens AG, Pictures of the Future Frühjahr 2003, 2003.

[29] Haldemann, Alexander; Giussani, Bruno , *Branding für das Mobile Internet - Das Handy als Teil eines konsistenten Marktauftritts*, Neue Zürcher Zeitung, 4. Februar 2003.

[30] von Angern, Ute; Bredow, Stefan; Gey, Christian, *Facial Animation in MPEG-4*, TU Berlin, 2000.

[31] Höhne, Heike, *MPEG-4 Gesichtsanimation: Das Gesichtsmodell und seine Parameter*, TU Dresden, 2004.

[32] Pötschke, Julia , *Gesichtsanimation mit MPEG-4 Teil II*, TU Dresden, 2004.

[33] Pandzic, Igor; Forchheimer, Robert, *MPEG-4 Facial Animation: The Standard, Implementation and Applications*, John Wiley and Sons, 2002.

[34] Demann, Tim, *3D-Rekonstruktion von Objekten aus Fotos am Beispiel von Gesichtern*, TU-Braunschweig, 2000.

[35] Wikipedia, *Die Freie Enzyklopädie*, http://de.wikipedia.org/wiki/, Wikpedia, 2003.

[36] Ostermann, Jörn, *Animation of Synthetic Faces in MPEG-4*, AT&T Labs, Philadelphia,1998.

[37] Ostermann, Jörn; Beutnagel, Mark; Fischer, Ariel; Wang, Yao, *Integration of Talking Heads and Text-to-Speech Synthesizers for Visual TTS*, AT&T Labs, Conference on Speech and Language Processing, Sydney, 1998.

[38] Ostermann, Jörn; Tekalp, Murat, *Face and 2D Mesh Animation in MPEG-4*, AT&T Labs, Philadelphia, 1998.

[39] Lauchner, Albert, *3GSM: Java dominiert, Symbian stärkt Position*, http://www.tecchannel.de/internet/1344/1.html, 04.03.2004.

[40] Ray, Christopher, *Die Wahrheit über Robert T-Online*, FAKTuell - Die Online Zeitung, Görlitz, 2000.

[41] Donath, Andreas, *Siemens packt Knuddel-Avatare aufs Handy*, Golem.de, Berlin, 2004.

[42] Nokia, *Nokia Produkte mit Symbian-Betriebssystem*, http://www.nokia.de/de/mobiltelefon/technologie/symbian, Nokia, 2004.

[43] Sun Microsystems, *Java 2 Platform, Micro Edition*, http://java.sun.com/j2me, Sun Microsystems Inc., 2004.

[44] Knudsen, Jonathan, *Wireless Java*, Springer Verlag, 2003.