FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Gesichtserkennung in ressourcen-beschränkten Umgebungen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Christoph Fischer
Matrikelnummer 9726405

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Privatdoz. Dipl.-Ing. Dr.techn. Martin Kampel

Wien, 24.11.2011

(Unterschrift Verfasser/in)

(Unterschrift Betreuer)

Christoph Fischer
Gartenweg 8, 3714 Sitzendorf

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

Wien, am 24.11.2011

Abstract

Security is a major concern in our highly-interconnnected world. In many areas, like e-government, e-commerce, online-banking etc. sensitive data is stored and transmitted. It has to ben ensured that only authorized persons have access to this data to prevent misuse. Biometrics can aid to securely identify persons to guarantee privacy. Face is besides finger print, iris, palm or vein, only to mention some, a biometric feature that can be exploited to ensure security. Nowadays security is mainly implemented by means of password or pins. This method has a number of drawbacks. The high distribution and application of this technique originates from a major advantage, that is the simplicity of use and implementation for both the user and the provider. Biometric methods can be an alternative or complement to this classic approach. With this master thesis, chosen face recognition approaches, namely Eigenfaces, Fisherfaces, Elastic Bunch Graph Matching and Enhanced Local Gabor Binary Patterns Histogram Sequence, are discussed, implemented and evaluated. The criteria are memory usage, storage usage, error rate (FAR and FRR) and computing time. In addition the work introduces a new algorithm (Marlies-Faces), that was developed together with a colleague at St. Pölten University of Applied Sciences with respect to the extremly constrained enviroment of a smart card. This approach is a combination of Enhanced Local Gabor Binary Patterns Histogram Sequence and Principal Component Analysis. Besides the theory of the approaches the for a whole face recognition system needed steps training phase, enrollment and authentification process are discussed for every method. With face recognition one has to deal with a number of challenges. Head pose, differences in brightness, mimic, changes caused by aging, beards, make-up, only to mention some. One has to deal with these problems when developing a successful face recognition system. Preprocessing steps can help to avoid or solve these problems. They cut hair or other irrelevant parts of the face, try to compensate illumination changes and to enhance the contrast. Important representatives of preprocessing steps (Histogram Equalization, Multi-Scale-Retinex, Daubechies Wavelets, TT) are discussed and evaluated.

Kurzfassung

In einer immer stärker vernetzten Welt ist Sicherheit ein entscheidendes Kriterium. In vielen Bereichen, z.B.: e-Government, E-Commerce, Online-Banking usw., werden sensible Daten gespeichert und übertragen. Es muss sichergestellt werden, dass nur autorisierte Personen Zugriff auf diese Daten haben, um Missbrauch vorzubeugen. Biometrie kann dabei helfen, Personen zu identifizieren und damit Sicherheit zu gewährleisten. Das Gesicht ist neben dem Fingerabdruck, der Iris, der Handfläche, der Vene usw. ein biometrisches Merkmal, das zur Authentifikation herangezogen werden kann. Heutzutage wird Sicherheit fast ausschließlich über Passwörter/PINs realisiert. Dieser Ansatz hat eine Reihe von Nachteilen. Die weite Verbreitung dieser Methode ist durch einen offensichtlichen Vorteil erklärbar, nämlich die Einfachheit der Benutzung und Umsetzung für sowohl den Anwender als auch den Anbieter. Biometrische Methoden können als Alternative oder Ergänzung dieses klassischen Ansatzes eingesetzt werden. Im Rahmen dieser Diplomarbeit werden ausgewählte Vertreter von Gesichtserkennungsverfahren, nämlich Eigenfaces, Fisherfaces, Elastic Bunch Graph Matching und Enhanced Local Gabor Binary Patterns Histogram Sequence, beschrieben, umgesetzt und evaluiert. Die Kriterien sind dabei RAM-Bedarf (Memory Requirements), Festspeicher-Bedarf (Storage Requirements), Erkennungsrate (Error Rate: FAR und FRR) und Computing Time (Aufwand). Die Arbeit behandelt auch ein neues Verfahren (Marlies-Faces), das im Hinblick auf die extrem ressourcen-beschränkte Umgebung Smartcard gemeinsam mit einer Kollegin an der FH St. Pölten entwickelt wurde. Diese neue Methode ist eine Kombination von Enhanced Local Gabor Binary Patterns Histogram Sequence und Princal Component Analysis. Neben der Theorie der Verfahren werden für jede Methode die für ein komplettes, praxistaugliches Gesichtserkennungssystem notwendigen Schritte Trainingsphase, Enrollment und Authentifzierungsvorgang erläutert. Bei der Gesichtserkennung hat man mit einer Reihe von Problemen zu kämpfen. Kopfhaltung, Helligkeitsunterschiede, Mimik, Veränderungen durch das Alter, Bärte, Make-Up, um nur einige zu nennen. All das muss für ein erfolgreiches Gesichtserkennungsystem bedacht und behandelt werden. Vorverarbeitungsschritte können helfen, solche Probleme zu umgehen oder zu lösen. Sie entfernen irrelevante Teile wie die Haare und den Rand des Gesichts und/oder versuchen, Helligkeitsunterschiede auszugleichen und den Kontrast zu erhöhen. Wichtigte Vertreter von Vorverarbeitungsschritten (Histogram Equalization, Multi-Scale-Retines, Daubechies-Wavelets, TT) werden beschrieben und evaluiert.

Contents

List of Figures

List of Tables

CHAPTER 1

Einleitung und Motivation

Biometrie bezeichnet "... die Lehre von den Meß- u. Zahlenverhältnissen der Lebewesen u. ihrer Einzelteile ..." [1]. In der IT-Sicherheitstechnik beschäftigt sie sich mit der Erfassung und Vermessung von Merkmalen des Menschen. Anhand dieser eindeutigen Merkmale können Menschen unterschieden und damit authentifiziert werden. Zu den biometrischen Merkmalen zählen u.a. der Fingerabdruck, die Iris, die Handgeometrie, die Unterschrift und das Gesicht.

In [2] beschreiben Jain et al. 7 Eigenschaften, die ein Merkmal haben muss, damit es in der Biometrie verwendet werden kann bzw. ein biometrisches System erfüllen muss. Ich verwende hier die englischen Bezeichnungen.

universality: "which means that every person should have the characteristic". Möglichst jede Person sollte über das Merkmal verfügen.

uniqueness: "which means that no two persons should be the same in terms of the characterstic". Das Merkmal soll bei allen Personen unterschiedlich sein.

permanence: "which means that the characteristic should be invariant with time". Das Merkmal soll sich im Laufe der Zeit möglichst wenig verändern.

collectability: "which indicates that the characteristic can be measured quantitatively". Das Merkmal muss erfassbar/messbar sein.

performance: "which refers to the achievable identification accuracy, the resource requirements to achieve an acceptable identification accuracy, and the working or environmental factors that affect the identification accuracy". Die erreichbare Identifizierungsgenauigkeit, die Ressourcen, die für eine akzeptable Genauigkeit benötigt werden, und die Faktoren, die die Genauigkeit beeinflussen.

Biometrics	Universality	Uniqueness	Permanence	Collectability	Performance	Acceptability	Circumvention
Face	High	LOW	Medium	High	LOW	High	LOW
Fingerprint	Medium	High	High	Medium	High	Medium	High
Hand Geometry	Medium	Medium	Medium	High	Medium	Medium	Medium
Keystrokes			LOW	Medium	LOW	Medium	Medium
Hand Vein	Medium	Medium	Medium	Medium	Medium	Medium	High
Iris	High	High	High	Medium	High	LOW	High
Retinal Scan	High	High	Medium	LOW	High	LOW	High
Signature	Low	Low	LOW	High	LOW	High	LOW
Voice Print	Medium	LOW	LOW	Medium	LOW	High	LOW
F. Thermograms	High	High	LOW	High	Medium	high	High
Odor	High	High	High	LOW	LOW	Medium	LOW
DNA	High	High	High	LOW	High	LOW	LOW
Gait	Medium	LOW	LOW	High	LOW	High	Medium
Ear	Medium	medium	High	medium	Medium	High	Medium

Figure 1.1: Jains Eigenschaften [2]

acceptability: "which indicates to what extent people are willing to accept the biometric system". Die Akzeptanz des biometrischen Systems beim Benutzer.

circumvention: "which refers to how easy it is to fool the system by fraudulent techiques". Wie leicht ist es, das System in betrügerischer Absicht zu täuschen?

Ein biometrisches Merkmal bzw. System sollte diese Eigenschaften erfüllen, um erfolgreich angewandt werden zu können. Die Merkmale wie Fingerabdruck, Iris etc. genügen nicht allen Eigenschaften zu 100%, jedoch zu einem akzeptablem Ausmaß. Figure 1.1 zeigt, wie gut die Eigenschaften von den verschiedenen biometrischen Merkmalen erfüllt werden.

Die Benutzerauthentifikation erfolgt in der Regel durch Passwörter/PINs. Dieser Ansatz hat eine Reihe von Nachteilen [3]. Passwörter sollten nicht aufgeschrieben werden, möglichst komplex sein, regelmäßig geändert werden und nicht weitergegeben werden [33]. Bei der Menge an Passwörtern, die sich ein jeder von uns für die verschiedensten Logins merken muss, gestaltet sich die Einhaltung dieser Ratschläge als schwierig [4]. Deshalb werden sie vom Großteil der Leute nicht befolgt. Hier kommt die Biometrie ins Spiel. Biometrische Merkmale wie der Fingerabdruck und das Gesicht können die klassischen Passwörter ersetzen oder zumindest ergänzen [5].

Natürlich haben Passwörter auch gewichtige Vorteile, die die weite Verbreitung als Authentifizierungsmechanismus erklären. Zu nennen ist dabei u.a. die Einfachheit der Anwendung

und Umsetzung sowohl beim Benutzer als auch beim Anbieter, das leicht mögliche Ändern des Schlüssels (des Passworts), die Akzeptanz unter den Benutzern. Der Einsatz von biometrischen Verfahren wird in der Regel als Eingriff in die Privatsphähre aufgefasst, es besteht ein gewisses "Unbehagen" [6]. Der Fingerabdruck beispielsweise wird oft mit Verbrechern assoziiert. Im Gegensatz zu biometischen Verfahren hat die Passwortauthentifizierung nicht den Makel des (gefühlten) Eingriffs in die Privatspähre. Dieses "Unbehagen" wird in Zeiten der zunehmenden Aushöhlung des Datenschutzes (Stichwort Facebook) und der zahlreichen erfolgten unautorisierten Zugriffe auf Datenbanken (vor einigen Monaten Sony Playstation Network [34] und ganz aktuell Steam von Valve Software [35]) noch verstärkt. Hier gilt es mit detailierten Informationen und klar definiertem Einsatzzweck und Regeln für Biometrie gegenzusteuern: Welche Daten werden gespeichert? Wielange werden die Daten gespeichert? Wer hat Zugriff auf die Daten? Wie werden die Daten vor unautorisiertem Zugriff geschützt? Wer kann die Daten verändern bzw. löschen? usw. Da diese Punkte für biometrische Systeme der Öffentlichkeit selten bekannt sind, hat Biometrie einen "gefühlten" schlechten Ruf.

Die wichtigste Voraussetzung für den praxistauglichen Einsatz von Biometrie ist die Erkennungsrate. Der Benutzer muss vom System erkannt und von anderen Personen unterschieden werden können. "Es muss funktionieren". Es muss möglichst vollständig ausgeschlossen sein, dass jemand beispielsweise beim Bankomaten kein Geld bekommt, weil das System versagt. Die Falschakzeptanzrate (False Acceptance Rate, FAR) d.h. die Wahrscheinlichkeit, mit der eine unautorisierte Person Zugriff erlangt, muss sehr gering (wenige Prozent) sein, um Missbrauch vorzubeugen. Noch viel wichtiger: Die Falschrückweisungsrate (False Rejection Rate, FRR), d.h. die Wahrscheinlichkeit, mit der eine autorisierte Person abgewiesen wird, muss sehr niedrig sein. Die Erkennungsrate ist für jedes biometrische System essentiell. Egal, welche Vorteile eine biometrische Methode hat, wenn die Erkennungsrate zu gering ist, ist an einen Einsatz in der Praxis nicht zu denken. Andere Aspekte wie Ressourcen-Verbrauch, Benutzerfreundlichkeit, Geschwindigkeit sind auch bedeutsam, jedoch überwiegt die Fehlerrate.

Die Forscher/Entwickler von Gesichtserkennungssystemen kämpfen mit einer Reihe von Problemen, z.B.: Heilligkeitsunterschiede, Kopfposition, Mimik, Veränderungen durch das fortschreitende Alter, Bärte, Make-Up, um nur einige zu nennnen [7]. Diese Probleme haben beträtchlichen Einfluss auf die Performance von Gesichtsserkennungssystemen [8] . "the variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity" [9]. Das Problem wird sehr gut durch Figure 1.3 verdeutlicht. Sie zeigt 4 Bilder derselben Person mit demselben neutralen Gesichtsausdruck unter unterschiedlichen Beleuchtungsverhältnissen. Es ist einsichtig, dass das für Gesichtserkennungsmethoden problematisch sein kann. Ein erfolgreiches Gesichtserkennungssystem muss mit diesen störenden Gegebenheiten umgehen können, entweder durch geeignete Vorverarbeitung (um beispielsweise Helligkeitsunterschiede auszugleichen) oder durch robuste Verfahren, bei denen die Erkennungsrate trotz dieser Probleme nicht übermäßig leidet.

Figure 1.2 zeigt den Anteil, den die verschiedenen biometrischen Merkmale am gesamten weltweiten Biometriemarkt haben. Die Gesichtserkennung liegt bei 11,4%.

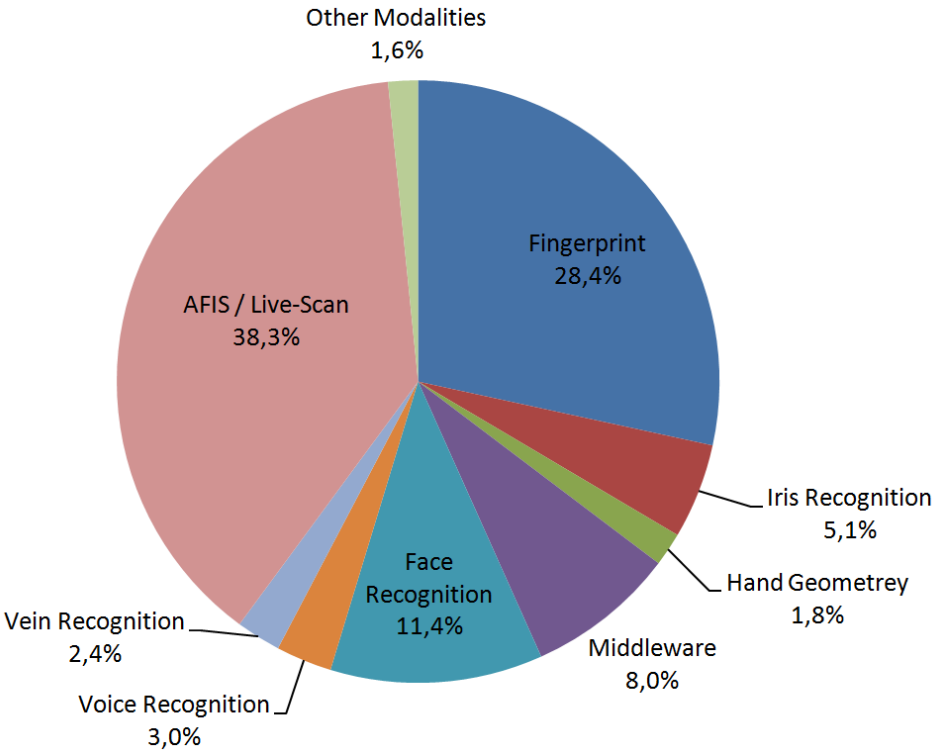

Figure 1.2: Der Anteil, den die verschiedenen biometrischen Merkmale am gesamten weltweiten Biometriemarkt haben [10]

Biometrie wird längst nicht mehr nur von Regierungsorganisationen oder Behörden eingesetzt. Authentifikation und Identifikation mittels biometrischer Merkmale boomt und ist ein Milliardengeschäft geworden. [36]. Der Umsatz von Biometrie soll für Europa, Asien und Afrika von 216,1 Millionen Euro im Jahr 2008 auf 1,058 Milliarden Euro im Jahr 2015 anwachsen [37]. Das entspricht einer Steigerung von über 25%. Die Voraussagen für Deutschland sind mit einer Steigerung von 20% bis 2014 ähnlich hoch [38]. Biometrie wird vermehrt an Arbeitsplätzen, aber auch im privaten Umfeld eingesetzt, um sichere Personenerkennung zu ermöglichen [39]. Laut [40] wollen drei Viertel der Arbeitnehmer in Zukunft mehr von zuhause aus arbeiten. Das erfordert am Heimarbeitsplatz die Erfüllung derselben Sicherheitsstandars wie im Unternehmen. E-Commerce, E-Government, Online-Banking usw. verlangen nach sicherer Authentifikation. Um vor Missbrauch durch unautorisierte Personen zu schützen, kann Biometrie eingesetzt werden.

Auf dem Gebiet der Gesichtserkennung gab es in den letzten Jahren seit der Veröffentlichung der Eigenfaces bedeutende Fortschritte. Zu nennen ist beispielsweise der Enhanced Local Gabor Binary Patterns Histogram Sequence [11], der einen neuen Ansatz in der Gesichtserkennung darstellt. Die Erkennungsrate ist jedoch nicht mit z.B. dem Fingerabdruck oder der Venenerken-

Figure 1.3: 4 Bilder derselben Person unter unterschiedlichen Beleuchtungsverhältnissen [7]

nung vergleichbar. Die Erkenngsrate ist im Vergleich zu diesen Merkmalen bedeutend niedriger.

Gesichtserkennung hat gegenüber anderen biometrischen Methoden einige Vorzüge. Die Authentifikation ist berührungslos und es ist keine Interaktion beim Authentifizierungsvorgang nötig. Das führt zu einer höheren Akzeptanz beim Benutzer. Weiters ist keine teure Hardware nötig, günstige Webcams sind ausreichend (ca. 20 €).

Im Rahmen der Diplomarbeit[1] wird eine Auswahl von Vorverarbeitungsmethoden und Gesichtserkennungsverfahren beschrieben, in einem C++-Framework umgesetzt und evaluiert. Die Kriterien sind Erkennungsrate, Memory Bedarf, Storage Bedarf und Computing Time (CPU-Zyklen). Der Aspekt der Erkennungsrate ist in der verfügbaren Literatur gut beschrieben. Die anderen Kriterien, die beim Einsatz in ressourcen-beschränkten Systemen essentiel sind, werden jedoch meistens nicht behandelt. Das sind jedoch neben der Erkennungsrate relevante Eigenschaften der Algorithmen, die ich im Rahmen dieser Arbeit untersuche. Beispielsweise ist im Embedded-Bereich (Smartcards, Smartphones, Handys) die Rechenkraft und der verfügbare Speicher stark eingeschränkt.

Anhand der benötigten Datenstrukturen wird der Memory-Bedarf berechnet (ohne Betriebssystem-Overhead). Der Storage Bedarf bezeichnet den Platz, den die Trainingsdaten und 3 Templates (je nach Verfahren die verarbeitete Repräsentation von 3 Gesichtsbildern der zu authentifizierenden Person) benötigen. Durch die Analyse von vielen Testläufen hat sich gezeigt, dass 3 Templates für die Authentifizierung ausreichen. Mehr bringen keine wesentliche Verbesserung der Erkennungsrate, und bei weniger steigt die Fehlerrate deutlich. Zur Ermittlung der Anzahl der benötigten CPU-Zyklen (Computing Time) verwendete ich Profiling. Alle Verfahren und

[1] Ich habe im Zeitraum von 15.07.2009 bis 25.07.2011 mein Praktikum an der FH St. Pölten absolviert. Ich arbeitete dort als wissenschaftlicher Mitarbeiter am geförderten Forschungsprojekt FaceMOC. Der Inhalt des Projektes war Gesichtserkennung. Das Thema hat mich fasziniert und es hat großen Spaß gemacht. Daher war es naheliegend, dass ich auch meine Diplomarbeit in diesem Bereich ansiedle. Das Wissen, das ich mir im Laufe des Praktikums erworben habe, hat mir bei der Erstellung der Diplomarbeit sehr geholfen.

Vorverarbeitungsschritte wurden unter Windows in Visual Studio 2008 Professional implementiert. Für jede Kombination von Vorverarbeitung und Verfahren wurden die 500 statischen Durchgänge (A.1) automatisch durchlaufen und das Ergebnis als Log-Datei (A.3) gespeichert. Mithilfe eines kleinen Programms (A.2) konnte dann im Nachhinein durch Parsen der Log-Datei der Schwellwert möglichst optimal gesetzt werden.

Der weitere Inhalt der Arbeit ist folgendermaßen gegliedert. Das nächste Kapitel behandelt nach einem Überblick zum Thema Gesichtserkennungsverfahren den State-of-the-Art. Ich beschreibe 3 aktuelle Veröffentlichungen zu diesem Thema. Darauf folgt eine detailierte Beschreibung der Funktionsweise der behandelten Vorverarbeitungsschritte und Gesichtserkennungsverfahren, die Theorie und der Ablauf des Einsatzes in einem Gesichtserkennungssystem. Kapitel 4 enthält die Evaluierung nach den vorher genannten Kriterien. Die Zusammenfassung rundet die Arbeit mit abschließenden Gedanken ab. Anhang, Literatur- und Linkverzeichnis beenden die Arbeit.

CHAPTER 2

State-of-the-Art

Gesichtserkennungsverfahren lassen sich in 3 Klassen unterteilen: holistische, merkmalsbasierte und hybride Methoden [5].

Holistische Verfahren verarbeiten die Gesichtsbilder als Ganzes, Pixel für Pixel, ohne im vorhinein markante Punkte wie die Pupillen, Nasenspitze etc. zu detektieren. Zu ihnen zählen u.a. Eigenfaces [12], Fisherfaces [13], Bayesian Intrapersonal/Extrapersonal Classifier [14], Enhanced Local Gabor Binary Patterns Histogram Sequence [11], Independent Component Analysis [15], Fourier Transform [16], Discrete Cosine Transform [17, 16].

Eigenfaces, Fisherfaces, Bayesian Intrapersonal/Extrapersonal Classifier und Independent Component Analysis sind subspace-basierte Verfahren. In der Trainingsphase wird ein Subspace erzeugt, mithilfe dessen sich die hochdimensionalen Input-Daten in eine weniger-dimensionale Darstellung projizieren lassen. Dabei erfolgt eine Merkmalsextraktion bei gleichzeitiger bedeutender Daten-Reduktion. Diese Projektionen können anschließend durch Distanzmetriken auf Ähnlichkeit untersucht werden.

Fourier Transform und Discrete Cosine Transform transformieren die Input-Bilder in die Frequency-Domain. So wird für das Auge unsichtbare Information extrahiert und zur weiteren Verarbeitung herangezogen. Eine Besonderheit dieser 2 Verfahren ist, dass keine Trainingsphase nötig ist.

Die Anwendung von Gabor Wavelets auf ganze Gesichtsbilder bzw. Teile davon ist bei einigen Verfahren zu finden. Beim Elastic Bunch Graph Matching, der zu den merkmalsbasierten Methoden gehört, werden markante Punkte im Gesicht detektiert und diese Knoten zusammen mit den umgegebenden Bereichen durch Gabor-Masken gefaltet. Diese Antwort der Gabor-Masken auf die Bildteile (Jets) bildet dann die Information anhand derer verglichen wird. Enhanced Local Gabor Binary Patterns verwendet ebenfalls, wie der Name schon sagt, die Gabor-Masken zur Informationsextraktion. Im Gegensatz zum Elastic Bunch Graph Matching werden

diese jedoch auf das ganze Bild angewandt, Histogramme dieser Bereiche erzeugt, und mittels Histogram Intersection gewichtete Ähnlichkeitswerte berechnet. Auch Dynamic Link Architecture [18] wendet Gabor-Wavelets an.

Untersuchungen [19] zeigen, dass Menschen sowohl das Gesicht als Ganzes als auch einzelne markante Stellen zum Wiedererkennen bzw. Unterscheiden von Gesichtern heranziehen. Dieser Umstand wird von den hybriden Methoden aufgegriffen. Sie kombinieren holistische mit merkmalsbasierten Ansätzen.

Im folgenden möchte ich 3 aktuelle Veröffentlichungen zum Thema Gesichtserkennung vorstellen.

2.1 Face Recognition Algorithms Surpass Humans Matching Faces Over Changes in Illumination

O'Toole et. al vergleichen in [20] die Fehlerrate, die sieben ausgewählte Gesichterkennungsalgorithmen erreichen, mit der menschlichen Fähigkeit, Gesichter Menschen zuzuordnen bzw. zu unterscheiden. Als Testgrundlage verwenden sie die standardisierte Face Recognition Grand Challenge (FRGC). Genauer, den schwierigsten Teil dieser Test-Suite, der aus dem Erkennen von Paaren von Gesichtsbildern, die unter verschiedenen Belechtungsverhältnissen aufgenommen wurden, besteht, d.h. je zwei Gesichtsbilder werden bearbeitet, um zu entscheiden, ob die Bilder dieselbe Person zeigen oder nicht. Figure 2.1 zeigt 2 Paare von Bildern. Oben dieselbe Person unter unterschiedlichen Beleuchtungsverhältnissen und unten 2 Bilder von verschiedenen Personen.

Die Algorithmen werden dazu mit allen möglichen Paaren von 16028 "target" und 8014 "probe" Gesichtsbildern getestet, das ergibt rund 128 * 10^6 Tests. Die "target" Geischtsbilder wurden unter kontrollierten Beleuchtungsverhältnissen aufgenommen, die "probe" Bilder unter nicht kontrollierten bzw. zufälligen. Die Algorithmen entscheiden mittels Ähnlichkeitswert, ob die Gesichter derselben Person angehören. Ist der Wert über dem Schwellwert, so ist das Ergebnis "dieselbe Person", andernfalls "unterschiedliche Personen". Die Matrix mit 16028 x 8014 Ähnlichkeitswerten wird mittels Receiver Operating Characteristic (ROC) Curve über alle mögliche Schwellwerte (alle Schwellwerte im möglichen Bereich) beschrieben.

Zum Vergleich mit der menschlichen Erkennungsfähigkeit werden Testpersonen "sehr leichte" und "sehr schwierige" Paare von Gesichtsbildern gezeigt. Die Probanden bewerten die Ähnlichkeit in einer 5-stelligen Ähnlichkeitsskala von "sicher dieselbe Person" bis "sicher unterschiedliche Personen". Die Ergebnisse der Algorithmen hinsichtlich der Paare von Bildern, die auch den Menschen gezeigt wurden, werden nun mit den menschlichen Entscheidungen verglichen. Dazu werden ROC Kurven der Algorithmen und Menschen erzeugt.

Die möglichen 128 * 10^6 Paare von Bildern werden in den Experimenten jedoch noch eingeschränkt.

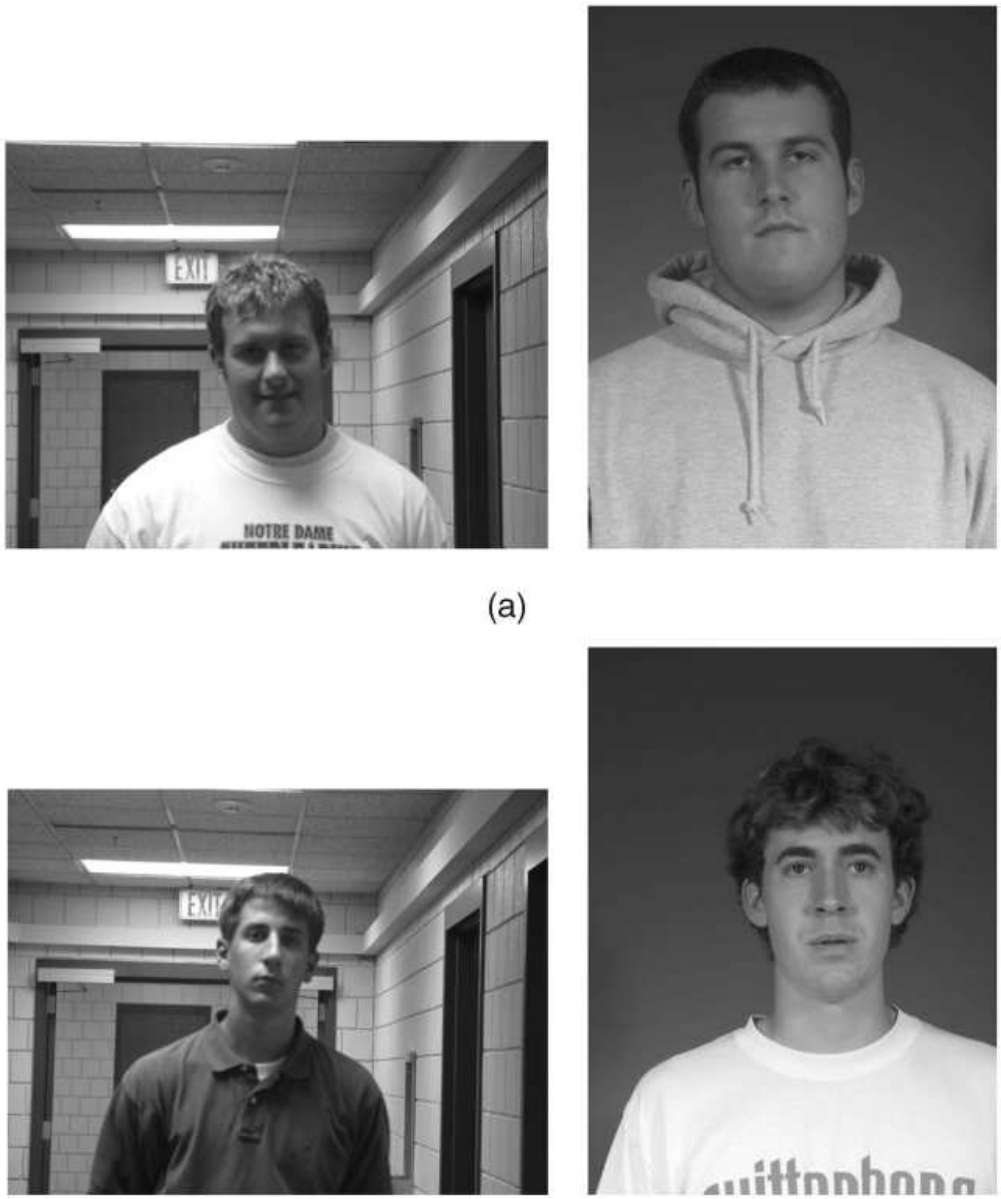

Figure 2.1: Paare von Gesichtsbildern: (a) dieselbe Person, (b) unterschiedliche Personen [20]

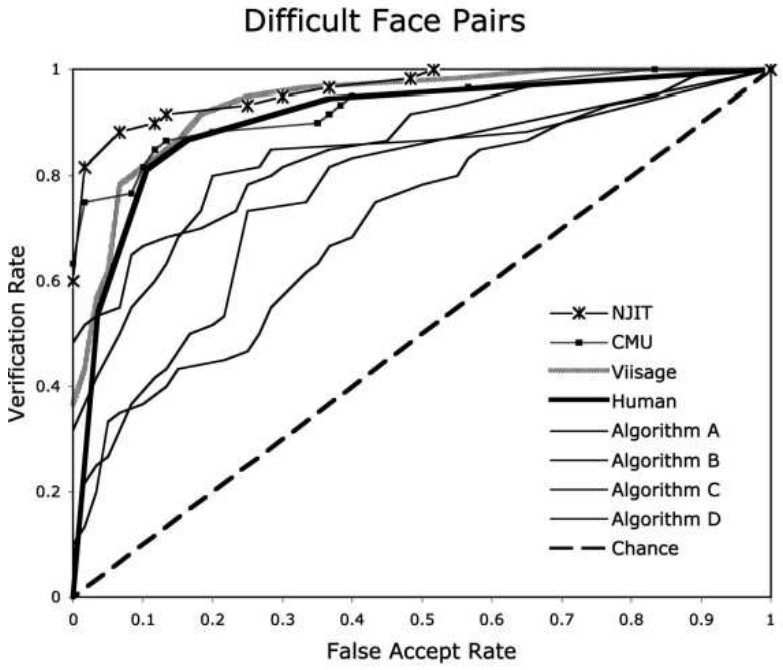

Figure 2.2: Erkennungsrate: schwierige Paare [20]

Es werden nur kaukasische Männer und Frauen bearbeitet und die Paare haben zusätzlich noch dasselbe Geschlecht. Durch diese Einschränkung wird das Experiment noch schwieriger. Die Algorithmen und Menschen haben keine Möglichkeit, die Paare anhand von Rasse oder Geschlecht zu unterscheiden.

Die Paare von Bildern werden durch einen Algorithmus, basierend auf PCA, anhand der Verteilung der Ähnlichkeitswerte, erzeugt durch die Gesichtserkennungverfahren, in schwierige und leichte Paare unterteilt. Das ist notwendig, um die Paare auszuwählen, die den Menschen gezeigt werden.

Die menschlichen Probanden waren 91 Studenten der "School of Behavorial and Brain Sciences at the University of Texas at Dallas". Das Experiment bestand nun im dem Erkennen von 120 männlichen und 120 weiblichen Paaren von Gesichtsbildern. Diese wurden 2 Sekunden lang auf einem Bildschirm gezeigt. Anschließend hatte man unbegränzt Zeit, die Antwort einzugeben. Die Entscheidungen wurden mit den Ergebnissen der Gesichsterkennungsverfahren für dieselben Paare von Bildern verglichen.

Figure 2.2 und 2.3 zeigen das Ergebnis des Experiments, die Erkennungsrate (ROC-Kurven) der Menschen und der 7 Algorithmen für die schwierigen und leichten Paare. Die Bewertung, ob "schwierig" oder "leicht", wurde, wie schon erwähnt, per PCA anhand der Ähnlichkeitswerte der Algorithmen bestimmt. 3 Algorithmen schneiden bei den schwierigen Paaren für fast alle

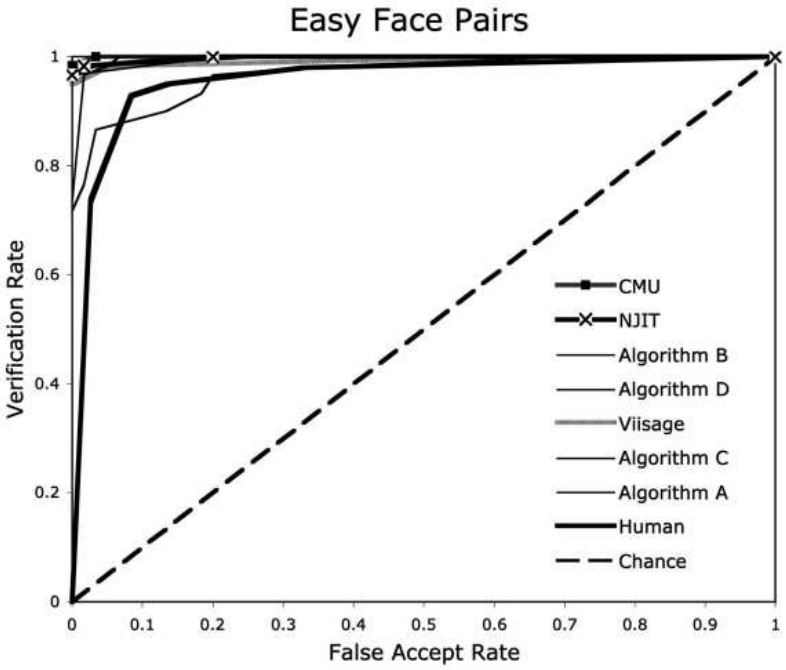

Figure 2.3: Erkennungsrate: leichte Paare [20]

möglichen Kombinationen von Verification Rate und False Accept Rate besser ab als die menschlichen Probanden. Die anderen 4 Algorithmen erzielen im Vergleich zum Menschen schlechtere Ergebnisse. 6 Algorithmen erreichen bei den leichten Paare bessere Resultate als die Menschen.

Zu noch einer Erkenntnis gelangte man im Rahmen des Versuchs. Man experimentierte auch mit der Dauer, mit der den Probanden die Gesichtspaare gezeigt wurden. Mit einer Verringerung auf 500ms stieg die Fehlerrate deutlich an. Jedoch hatte eine Verlängerung auf "unendlich" keinen positiven Effekt auf die Erkennungsrate. Man einigte sich daher, die 2 Sekunden zu belassen.

2.2 Approach of Human Face Recognition Based on SIFT Feature Extraction and 3D Rotation Model

Zhou et al. stellen in [21] ein neues Gesichtserkennungsverfahren vor, um die Probleme der Helligkeit und Unterschiede in der Kopfposition zu bewältigen. Es basiert auf der SIFT (Scale Invariant Feature Transform) feature extraction und 3D rotation model. Figure 2.4 zeigt den Ablauf eines Identifizierungsvorgangs.

Das System verwendet 3 Kameras, um Aufnahmen des Testgesichtes unter unterschiedlichen Betrachtungswinkeln zu machen. Nach der Detektierung des Gesichtes wird durch Vorverar-

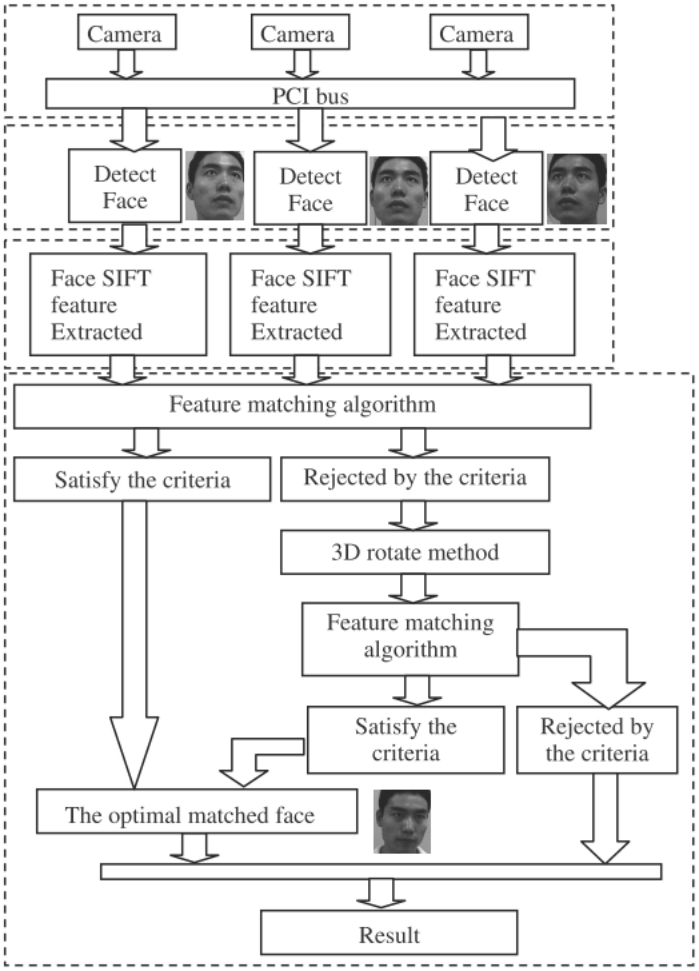

Figure 2.4: Ablauf des Identifizierungsvorgangs [21]

beitung störendes Bildrauschen entfernt und Helligkeitsunterschiede ausgeglichen. Anschließend werden die SIFT features extrahiert. Diese sind invariant zu verschiedenen Betrachtungswinkeln, Bildskalierung und Rotation. Die Features des Testbildes werden nun mit den Features der Datenbank verglichen. Das erfolgt durch Berechnung der Euklid-Distanz der Feature Vektoren. Mittels Nearast Neighbor Klassifizierung wird der "best matching" Kandidat ausgewählt. Die Datenbank wurde im vorhinein einmalig erstellt. Ist nun die Distanz des Nearast Neighbors unter einem vorher definierten Schwellwert, wird das in der Datenbank gefundene Gesicht als "erkannt" klassifiziert und der Identifizierungsvorgang ist damit abgeschlossen.

Ist die Distanz unter dem Schwellwert, wird das 3D Rotation Model zur weiteren Entscheidung herangezogen, indem eine Tiefennormalisierung durchgeführt wird. Es wird ein Durchschnitts-Frontal-Gesicht berechnet und der Unterschied des Betrachtungswinkels des Testbildes zu diesem

Figure 2.5: Durchschnitts Frontal Gesicht [21]

Figure 2.6: Angleichen des Testbildes an das Durchschnittsbild [21]

Durchschnittsbild ermittelt, 2.5. Mithilfe dieses Winkels lässt sich das Testgesicht an ein Frontal-Bild angleichen 2.6. Jetzt wird der erste Schritt mit diesem neuen Testbild wiederholt.

Mithilfe einer Datenbank, die 70 Personen mit Aufnahmen von je 9 Betrachtungswinkeln enthält, wurde die Erkennungsrate des vorgestellten Verfahrens für verschiedene Schwellwerte ermittelt, siehe 2.7 und 2.8 und 2.9. Die Fehlerrate der Methode wurde außerdem mit anderen Ansätzen verglichen, 2.10.

2.3 Human Face Recognition Based on Principal Component Analysis and Particle Swarm Optimazatin -BP Neural Network

In [22] stellen Du et al. ein neues Gesichtserkennungsverfahren vor. Es basiert auf der Kombination von Principal Component Analysis und Neural Networks.

PCA wird zur Extraktion von Features der Gesichtsbilder verwendet. Anschließend wird ein neurales Netz mit diesen Features trainiert. Die Gewichte werden mittels Particle Swarm Optimization verbessert.

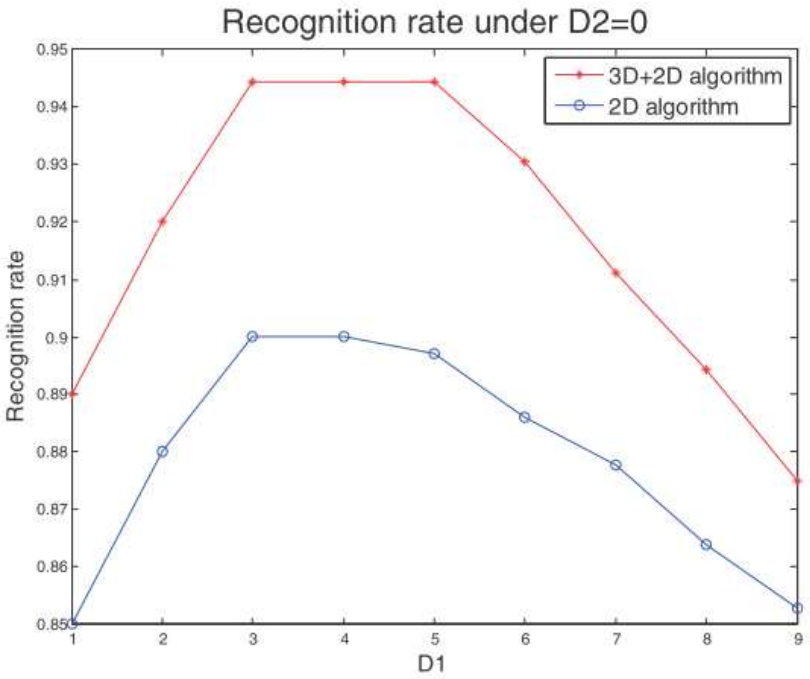

Figure 2.7: Erkennungsrate [21]

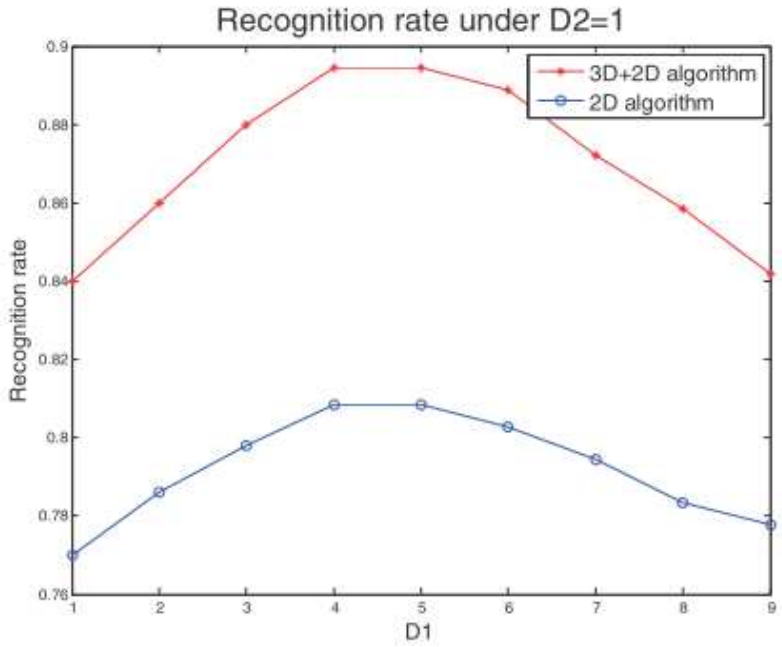

Figure 2.8: Erkennungsrate [21]

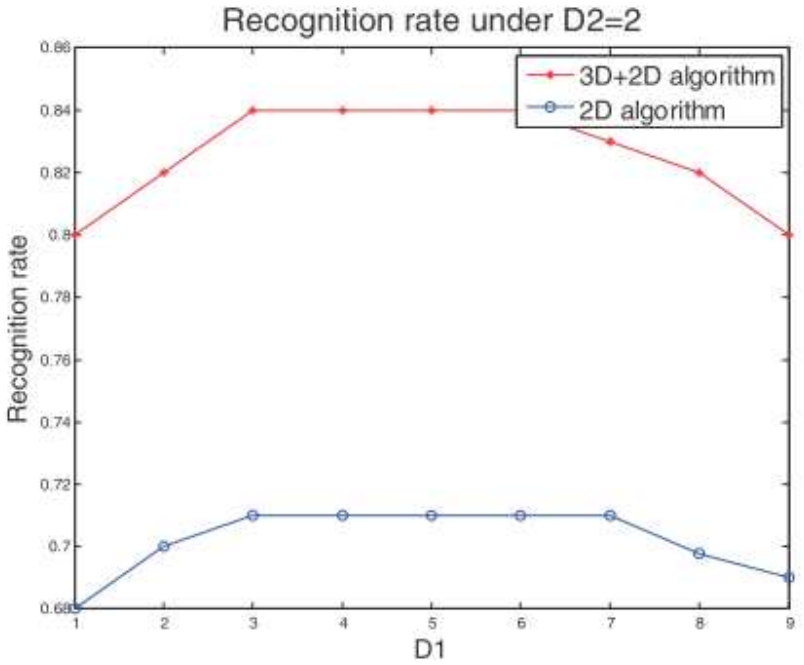

Figure 2.9: Erkennungsrate [21]

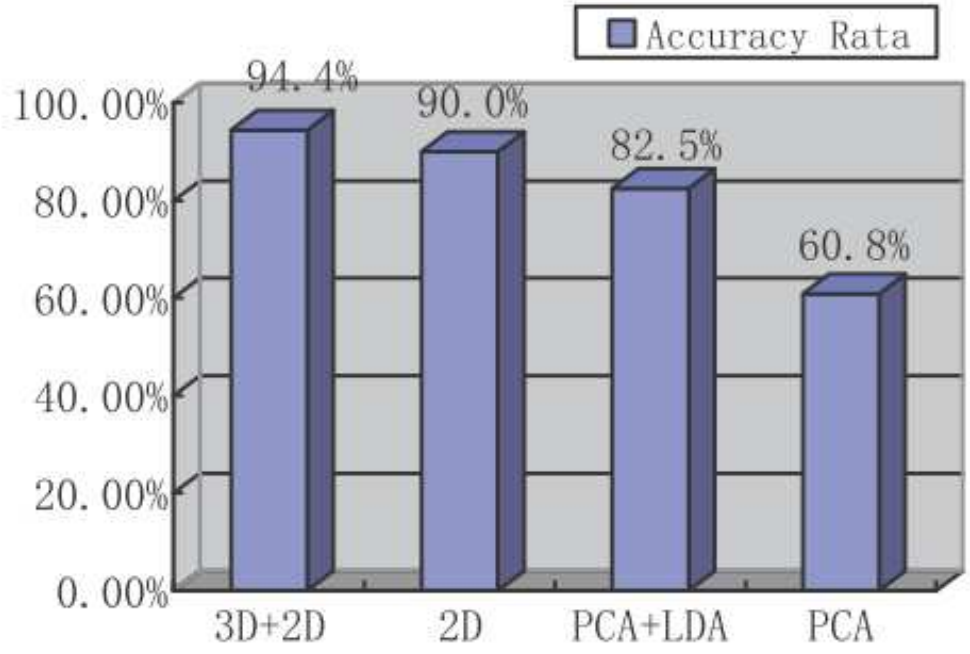

Figure 2.10: Vergleich mit anderen Ansätzen [21]

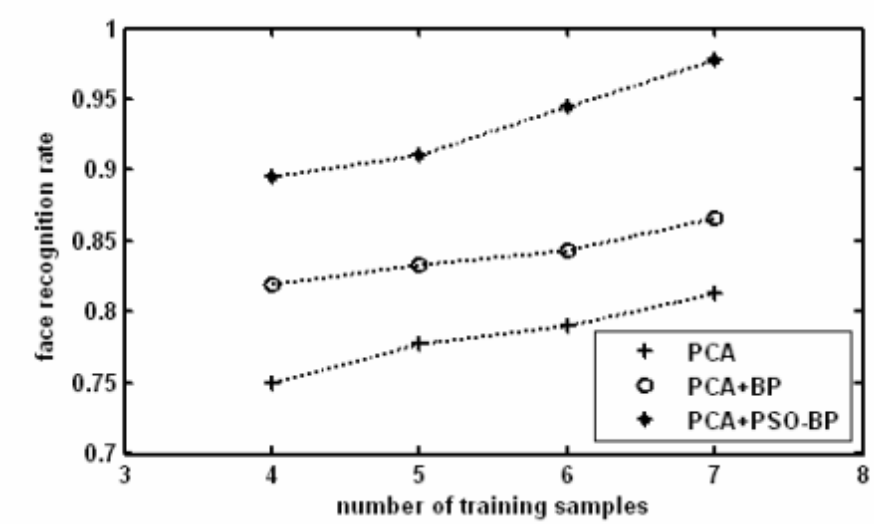

Figure 2.11: Erkennungsrate [22]

Die Erkennungsrate ist aus Figure 2.11 und 2.12 ersichtlich. Zur Evaluierung wurde die ORL Gesichtsdatenbank verwendet. Diese enthält 400 Bilder von insgesamt 40 Personen, damit 10 Bilder pro Person. 300 Bilder wurden zufällig ausgewählt und mit 4, 5, 6, und 7 Bildern pro Person trainiert. Die restlichen 3 Bilder wurden als Testbilder zur Ermittlung der Fehlerrate herangezogen.

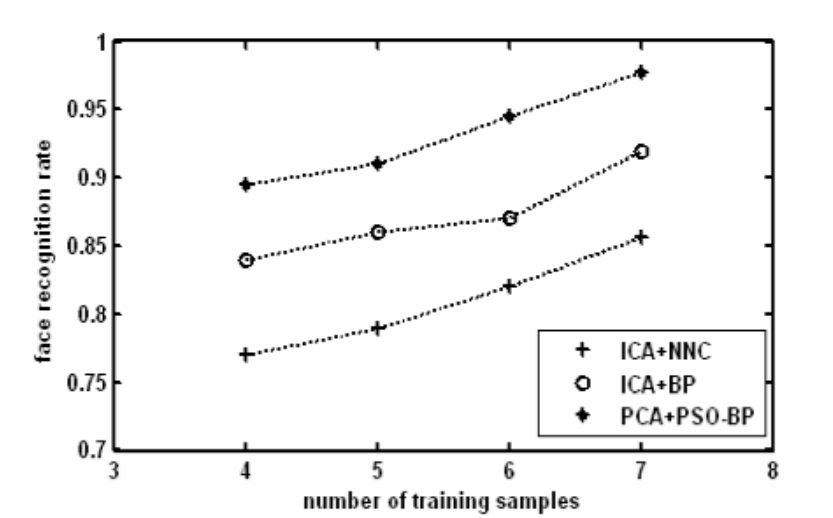

Figure 2.12: Vergleich zu anderen Methoden [22]

CHAPTER 3

Methoden

Dieses Kapitel beschreibt die Funktionsweise der behandelten Gesichtserkennungsverfahren und Bildvorverarbeitungsschritte inkl. aller Formeln.

3.1 Vorverarbeitung

Vorverarbeitung wird in der Gesichtserkennung wie auch in anderen Bereichen verwendet, um die Input-Bilder auf verschiedene Arten zu verbessern, beispielsweise durch Erhöhung des Kontrast, Verstärkung von Details und Kanten, eine gleichmäßigere Verteilung der Grauwerte, Entfernung von Bildrauschen etc.. Das Ziel ist eine Verminderung der Fehlerrate der Algorithmen. Störende Einflüsse wie der Hintergrund, Bildrauschen etc. werden entfernt oder zumindest in ihrem negativen Effekt abgeschwächt. Der Kontrast wird verstärkt und die Helligkeit angepasst. Bilddetails, dabei vor allem Kanten (abrupte hohe Änderungen der Grauwerte), werden betont. Im Allgemeinen werden die Input-Bilder so verändert, dass es einen positiven Effekt auf die Erkennungsrate des Algorithmus hat. Im Folgenden werden die im Rahmen der Diplomarbeit umgesetzten und getesteten Vorverarbeitungsschritte vorgestellt. Das Ausmaß der Verbesserung der Fehlerrate war jedoch entgegen vorheriger Annahmen bis auf Face Detection und Histogram Equalization eher gering.

Face Detection

Face Detection bezeichnet das Finden eines Gesichtes in einem Bild. Der detektierte Gesichtsbereich wird ausgeschnitten und ist der Input für die nachfolgenden Schritte. In den Testläufen wurde die Implemetierung des Haar Classifier Cascade (beschrieben in [23]) (Haar) der OpenCV Library [41] verwendet. Diese liefert sehr gute Ergebnisse und war in der Lage, in fast jedem Bild der FERET Gesichtsdatenbank das Gesicht zu identifizieren: Bei den ca. 12000 Anwendungen der Face Detection im Rahmen der 500 statischen Durchgänge konnte 21 mal kein Gesicht gefunden worden, das entspricht einer Fehlerrate von 0.2%. In Figure 3.1 sieht man die Anwendung beispielhaft für ein Bild der Datenbank. Ein großer Vorteil dieser Face Detection

Figure 3.1: Face Detection: links das Originalbild, rechts der gefundene Gesichtsbereich [42]

ist, dass die Positionen der Augen bei unterschiedlichen Gesichtsbildern sehr nahe beieinander liegen. Das ist hilfreich für die nachfolgenden Schritte. Außerdem hat das Ergebnisbild immer die Dimension 200 x 200, eine nachträgliche Größenänderung ist damit nicht notwendig. In allen Testläufen war der erste Schritt die Anwendung des Haar Classifier Cascade auf die in Graustufen konvertierten Bilder der FERET Gesichtsdatenbank. Die wenigen Fälle, wo kein Gesicht gefunden wurde, wurden als Erkennungsfehler klassifiziert.

Resize

Das Gesichtsbild hat nach der Face Detection die Dimension 200 x 200. Im Rahmen des Praktikums und der Diplomarbeit wurde mit verschiedenen Bildgrößen experimentiert, von 50 x 50 bis 200 x 200. Die Dimension von 200 x 200 hatte jedoch in den Tests im Vergleich zu 100 x 100 keine messbare niedrigere Fehlerrate zur Folge. Deshalb wurden alle Bilder auf diese Große reduziert (Ausnahme bei EBGM: 150 x 200). Dazu verwendete ich die Funktion cvResize() der OpenCV Library [41]. Diese interpoliert die neuen Pixel-Werte. Die Verminderung der Datenmenge auf 1/4 ist für die Verarbeitung durch die Algorithmen vorteilhaft, weil die Ausführungszeit wesentlich abnimmt.

Elliptisches Zuschneiden

Der dritte Schritt der Vorverarbeitung war für einige nachträglich vorgestellte Methoden das elliptische Zuschneiden des Gesichts (Elliptic). So wird ein Großteil des störenden Hintergrunds entfernt, oder besser gesagt: Der Hintergrund hat danach für alle Bilder dieselbe Farbe, bedeckt denselben Bereich und fällt damit nicht ins Gewicht. Vor allem bei den Eigenfaces führte das zu einer deutlichen Verbesserung der Erkennungsrate. Bei der Erkennung nicht relevanter Teile wie die Ohren, der Rand des Gesichtes und die Haare werden beim Zuschneiden entfernt. Figure 3.2 zeigt das Ergebnis dieses Schrittes auf ein Bild der Gesichtsdatenbank.

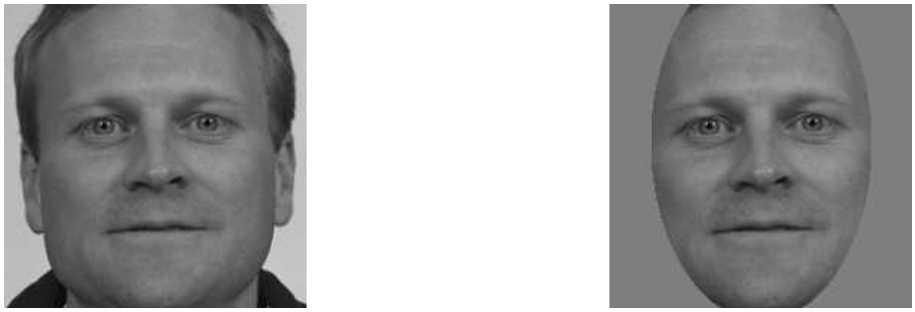

Figure 3.2: elliptisches Zuschneiden des Gesichtsbildes [42]

Histogram Equalization [24]

Die Pixel von 8-Bit-Grauwert-Bildern können die Werte von 0 bis 255 annehmen. In manchen Grauwert-Bildern liegt der Grßteil der Grauwerte innerhalb eines kleinen Bereiches in der Nähe der Mitte des möglichen Bereichs. Durch Histogramme solcher Bilder kann man sich das verdeutlichen. Histogram Equalization (HE) wird nun eingesetzt, um diesen kleinen Bereich in den Bereich von 0 bis 255 zu transformieren. Der Bereich wird dann beispielsweise von 24 bis 160 auf 0 bis 255 ausgedehnt, möglichst gleichverteilt. Das führt zu einer Erhöhung des Kontrasts und besser erkennbaren Details. Histogram Equalization erreicht dies, indem eine kumulative Verteilungsfunktion (cumulative distribution function) der Grauwerte erstellt wird. Die Werte dieser Funktion bilden die neuen Grauwerte. In Figure 3.3 sieht man den Effekt der Anwendung dieser Methode (Es wird dabei nur der Bereich innerhalb der Ellipse zur Berechnung herangezogen, der Hintergrund wird ignoriert). Man sieht den offensichtlich förderlichen Effekt sehr gut. Der Kontrast ist verstärkt und die Details (z.B. die Bartstoppeln) sind sehr viel besser erkennbar. In der OpenCV Library ist Histogram Equalization als fertige Funktion implementiert. Ich setzte diese Methode jedoch selbst um, weil die Ergebnisse der OpenCV-Funktion mangelhaft waren. In Figure 3.4 ist aus dem Beispiel ersichtlich, wie man die neuen, besser verteilten Grauwerte berechnet. Das Input-Bild ist links oben zu sehen. Mittig oben werden die Grauwerte gezählt. Rechts oben werden die kumulativen Summen (aktuelle Anzahl + letzte Summe) berechnet. Mithilfe der Formel unten werden die Grauwerte in neue, besser verteilte Grauwerte transformiert. $oldMin$ bezeichnet die kleinste kumulative Summe (in diesem Fall 2 von Grauwert 83), $oldMax$ die größte kumulative Summe (in diesem Fall 16 von Grauwert 93). Rechts das verbesserte Ergebnisbild.

Multi-Scale Retinex

Multi-Scale Retinex [25, 26, 27] (MSR) ist eine Methode, die Helligkeitsunterschiede in Bildern ausgleicht. Sie operiert als einzige Vorverarbeitungsmethode ursprünglich auf Farb-Bildern. Im Rahmen der Implementierung des Gesichtserkennungsframeworks wurde die Methode für Grau-Bilder adaptiert. Die nachfolgend beschriebene Vorgehensweise ist eine Kombination der Schritte/Informationen aus den 3 anfangs genannten Papers.

Figure 3.3: Histogram Equalization [42]

86	93	91	90
83	87	90	92
84	87	88	86
83	87	90	93

83: 2	87: 3	91: 1
84: 1	88: 1	92: 1
85: 0	89: 0	93: 2
86: 2	90: 3	

83: 2	87: 8	91: 13
84: 3	88: 9	92: 14
85: 3	89: 9	93: 16
86: 5	90: 12	

$$neuerGrauwert = (SummeAlterGrauwert - oldMin) * (255/oldMax)$$

48	223	175	159
0	80	159	191
16	80	112	48
0	80	159	223

Figure 3.4: Histogram Equalization: Berechnung der neuen Grauwerte

Als erstes wird das Input-Bild $S()$ in Reflexionsanteil $R()$ und Helligkeitsanteil $L()$ gespalten.

$$S(x, y) = R(x, y) * L(x, y) \tag{3.1}$$

Das Reflexionsbild $R()$ kann durch folgende Formel berechnet werden:

$$R_i(x, y) = \sum_{k=1}^{K} W_k(logI_i(x, y) - log[F_k(x, y) * I_i(x, y)]) \tag{3.2}$$

Bei Farbbildern geht i von 0 bis 2 (für die 3 Farbkanäle). Man erhält so 3 Reflexionsbilder. In unserem Fall (Graustufenbilder) haben wir nur einen Kanal und damit nur ein Reflexionsbild als Output. I_i bezeichnet den Farbwert an der Position (x, y) für Kanal i. $F_k()$ sind die Nachbarschaftsfunktionen. Sie können frei gewählt werden. In der Implementierung verwendete

ich 4 Gaussian Blur Funktionen mit unterschiedlichen Parametern sigma. W_k bezeichnet die Gewichtung der entsprechenden Nachbarschaftsfunktion. Ich verwendete für alle W_k den Wert 1, so sind alle Nachbarschaftsfunktionen gleich gewichtet. Die Werte von $R()$ befinden sich in der logarithmischen Domäne. Mithilfe von $S()$ und $R()$ lässt sich nun das Helligkeitsbild auf folgende Weise berechnen:

$$L(x,y) = logS(x,y) - R(x,y) \tag{3.3}$$

Nun werden das Helligkeitsbild und das Reflexionsbild verändert und anschließend wieder zum resultierenden Bild zusammengefügt. Auf das Reflexionsbild wird Color Restoration angewandt:

$$C_i(x,y) = \beta\{log[\alpha I_i(x,y)] - log[\sum I_j(x,y)]\} \tag{3.4}$$

Wie bei der Berechnung von $R()$ iteriert i über die Kanäle, bei Farbbildern von 0 bis 2. Man erhält daher einen verbesserten Farbwert für jeden Kanal. Im Fall von Graustufenbildern haben wir nur einen Kanal und damit nur einen Wert pro Pixel. Die Summe gegen Ende der Formel ist die Summe über die Farbwerte aller Kanäle an der entsprechenden Position. "β is the gain constant, and α controls the strength of nonlinearity" [25]. Für β verwendete ich den Wert 1 und für α 128. Im Fall von Graustufenbildern lässt sich obige Formel folgendermaßen anschreiben:

$$C(x,y) = \beta\{log[\alpha I(x,y)] - log[3 * I(x,y)]\} \tag{3.5}$$

Auf das Helligkeitsbild wird Gamma Correction angewandt. Die Werte befinden sich jedoch in der logarithmischen Domäne, daher ist zuerst folgendes notwendig (für alle Pixel):

$$L = e^L \tag{3.6}$$

Gamma Correction:

$$L' = W * (L/W)^{\frac{1}{\gamma}} \tag{3.7}$$

Figure 3.5: links das Original [27], rechts die Anwendung des modifizierten MSR

W bezeichnet den größtmöglichen Farbwert, hier 255. Für γ verwendete ich den Wert 1.2. $\gamma > 1$ erhöht die Helligkeit, $\gamma < 1$ vermindert die Helligkeit. Abschließend wird noch sichergestellt, dass die resultierenden Grauwerte den Bereich nicht überschreiten.

if pixelvalue > 255 then
pixelvalue = 255

In Figure 3.5 ist die Anwendung dieser Vorverarbeitung auf ein Bild von [27] dargestellt. Links das Input-Bild. Rechts das Ergebnis des (modifizierten) Multi-Scale-Retinex Algorithmus. Es sind viele Details sichtbar (die Bartstoppeln), die man im Originalbild nicht erkennen kann.

Daubechies-Wavelet-Transformation

Daubechies-Wavelet-Transformation (Daub4) ist eine Methode zur Verbesserung/Normalisierung der Helligkeitsverteilung von Bildern. Gleichzeitig werden dabei Details (Kanten) verstärkt. Das Bild wird zu diesem Zweck in die Wavelet-Domäne hinübergeführt. Dazu muss das Bild

LL	HL
LH	HH

Figure 3.6: Die 4 Sub-Bänder

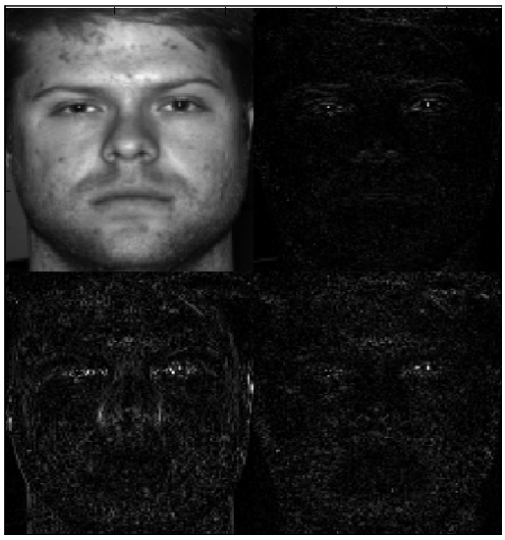

Figure 3.7: Die 4 Sub-Bänder eines Gesichtsbildes [28]

spaltenweise und dann zeilenweise durch die 1-level Daubechies-Wavelet-Transformation bearbeitet werden. Man erhält so vier Bänder, zwei low-frequency (low-low und low-high) und zwei high-frequency (high-low und high-high) Sub-Bänder, siehe Figure 3.6 und 3.7. Die Low-Bänder werden durch die "approximation coefficients" erzeugt, die High-Bänder durch die "detail coefficients" [28].

Um den Kontrast zu erhöhen, wird das LL Band mittels Histogram Equalization bearbeitet. Die Informationen in den anderen Sub-Bändern werden durch Multiplikation mit einem Skalar verstärkt. Abschließend werden die veränderten Bänder durch mehrfache inverse Daubechies-Wavelet Transformation wieder zum Gesamtbild zusammengeführt. Zuerst LL und LH zu L, dann HL und HH zu H und anschließend L und H zum Ergebnisbild. Es muss angemerkt werden, dass jede beliebige Wavelet-Dekomposition verwendet werden kann. Je nach Art erhält man unterschiedliche Subbänder. In [28] wird die einfache 1-level Daubechies-Wavelet-Transformation verwendet. Diese habe ich auch im Framework umgesetzt.

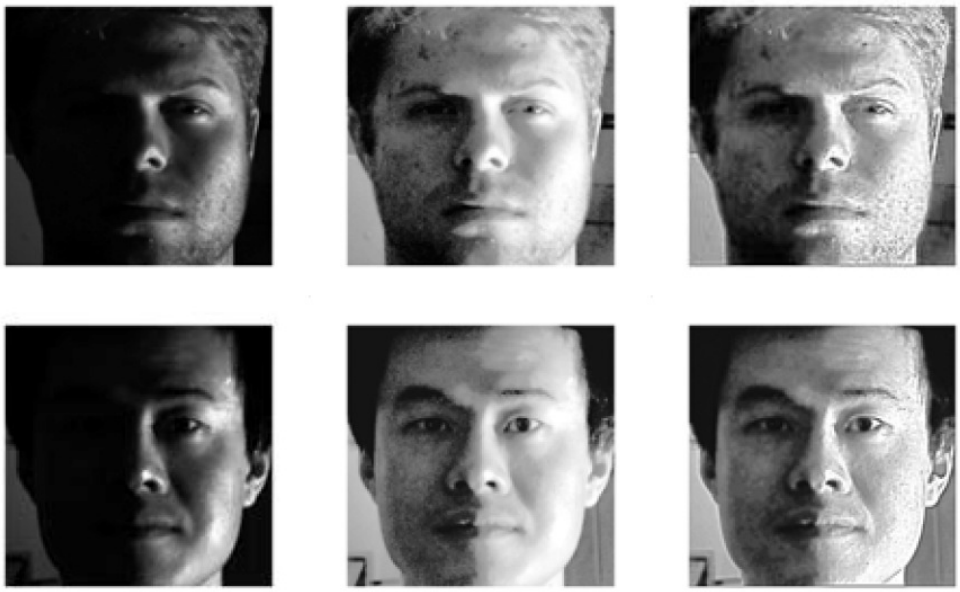

Figure 3.8: links Original, mitte Histogram Equalization, rechts Daubechies [28]

Gamma Correction, Difference of Gaussian Filtering, Contrast Equalization

In [29] beschreiben Tan et al. eine neues Verfahren zur Gesichtserkennung. Das Verfahren selbst wurde von mir nicht behandelt, jedoch die auch im Rahmen des Papers vorgestellte Vorverarbeitung. Die unbenannte Methode wird von mir TT genannt und besteht aus einer Kombination von Gamma Correction, Difference of Gaussian Filtering (DoG) und Contrast Equalization, in dieser Reihenfolge. Gamma Correction (siehe Formel 3.8) verbessert die Verteilung der Grauwerte in dunklen bzw. schattigen Abschnitten und komprimiert gleichzeitig den Bereich der Grauwerte in hellen Bereichen [29]. Es kommt also zu einer generell besseren Helligkeitsverteilung.

$$I = I^{\gamma} \tag{3.8}$$

γ liegt im Bereich $[0, 1]$. Wie im Paper vorgeschlagen, verwendete ich für γ 0.2.

Bei DoG werden zuerst 2 Gaussian Blur Bilder des Original-Bildes mit verschiedenen Parametern σ erzeugt. Für σ_0 verwendete ich 1.0 und für σ_1 2.0. Das stärker "verwischte" Bild (mit größerem σ) wird anschließend vom anderen Gauss Blur Bild subtrahiert. Es handelt sich dabei um einen Bandpass-Filter. In Formel 3.9 die Definition der ein-dimensionalen Gauss Blur Funktion. Sie wird zuerst in Y-Richtung und danach in X-Richtung auf das Bild angewandt. Die resultierenden Blur Bilder werden voneinander subtrahiert (Formel 3.10).

Figure 3.9: von links nach rechts: Original, Gamma Correction, DoG, Contrast Equalization [42]

$$G(x,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.9}$$

$$I = G(x,\sigma_0) - G(x,\sigma_1) \tag{3.10}$$

Der abschließende Schritt ist "Robust Contrast Equalization", um den globalen Kontrast und die Helligkeitsverteilung zu verbessern. Probleme machen dabei sehr helle oder sehr dunkle Bereiche. Es ist wichtig, einen robuste Schätzfunktion zu verwenden [29]. Laut [29] hat sich folgende einfache und schnelle Vorgehensweise als zielführend erwiesen (Formeln 3.11 bis 3.13). Auf diese Weise wurde es von mir im Rahmen der Diplomarbeit auch umgesetzt.

$$I(x,y) = \frac{I(x,y)}{(mean(|I(x',y')|^a))^{\frac{1}{a}}} \tag{3.11}$$

$$I(x,y) = \frac{I(x,y)}{(mean(min(\tau,|I(x',y')|^a))^{\frac{1}{a}}} \tag{3.12}$$

$$I(x,y) = \tau tanh(\frac{I(x,y)}{\tau}) \tag{3.13}$$

Als Werte für die Parameter wurden von mir $a = 0.1$ und $\tau = 10$ verwendet. In Figure 3.9 sieht man die Anwendung der Methode auf ein Bild der FERET Datenbank. Ganz links das Originalbild nach der Face Detection, daneben nach der Anwendung von Gamma Correction, Difference of Gaussian Filtering und Contrast Equalization.

3.2 Eigenfaces

1991 stellten Turk und Pentland in [12] ein Verfahren zur Gesichtserkennung, die Eigenfaces, vor. Der Ansatz basiert auf der Principal Component Analysis (PCA, Hauptkomponentenanalyse) zur Merkmalsextraktion bei gleichzeitiger Informationsreduktion.

Hauptkomponentenanalyse

In vielen Bereichen (inkl. der Gesichtserkennung) hat man es mit sehr großen Datenmengen zu tun. Die Bestimmung (Extraktion) der relevanten, aussagekräftigen Teile dieser Informationen ist eine Voraussetzung für die effiziente Verarbeitung der Daten. Hochdimensionale Daten enthalten normalerweise viel redundante Information und weniger relevante Teile. Die Hauptkomponentenanalyse kann nun eingesetzt werden, um die Komponenten mit größtem Informationsgehalt (größter Relevanz, die Hauptkomponenten) zu bestimmen. Das Maß für die Relevanz ist dabei die Varianz. Sind die Hauptkomponenten bekannt, kann ein geringere Anzahl (im Vergleich zur hochdimensionalen Ausgangsmenge) von Variablen gefunden werden, die die Daten mit möglichst wenig Informationsverlust beschreiben. Es findet also eine Informationsextraktion bei gleichzeitiger Datenreduktion statt. Die Eigenfaces wenden dieses Prinzip an, um die relevante Information aus den hochdimensionalen Gesichtsbildern zu extrahieren. Die PCA wird aber nicht nur bei den Eigenfaces angewandt, sondern in einer Vielzahl von anderen Bereichen als statistisches Verfahren erfolgreich eingesetzt.

Die Eigenfaces lassen sich in 2 Phasen unterteilen. Die Trainingsphase findet nur einmal statt. Hier werden Trainingsbilder (Gesichtsbilder) verarbeitet und mittels PCA Merkmale extrahiert, um einen sogenannten Eigenspace zu erzeugen. Diese Merkmale (Eigenfaces) dienen in der Test-Phase (Authentifikationsphase) dazu, die Gesichtsbilder in den Eigenspace zu projizieren. Dabei findet eine beachtliche Informationsreduktion statt. Projektionen mehrerer Bilder lassen sich durch Distanz-Metriken (z.B.: Euklidische Distanz) auf Ähnlichkeit untersuchen. Ist die Distanz zwischen zwei Projektionen größer als ein händisch festgelegter Schwellwert, so ist das Ergebnis "unterschiedliche Personen", ist sie kleiner als der Schwellwert, so folgt daraus "dieselbe Person".
Im folgenden wird der genaue Ablauf inklusive aller Formeln erläutert.

Ablauf

Die zu erfolgenden Schritte zur Authentifikation eines Gesichtsbildes lassen sich in Trainingsphase, Enrollment und Authenifizierungsvorgang unterteilen.

Trainingsphase

Die Trainingsbilder (Graustufenbilder) werden bei den Eigenfaces als Vektoren ihrer Grauwerte aufgefasst. Im ersten Schritt erfolgt eine Normalisierung der Gesichtsbilder x_i durch Subtraktion des Durchschittsbildes m. Dabei werden die Gemeinsamkeiten der Gesichter "entfernt" und die Unterschiede betont.

$$m = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.14}$$

$$\bar{x}_i = x_i - m$$

Die normalisierten Gesichtsvektoren $\bar{x}_i$ werden anschließend in einer Matrix (Datenmatrix $\bar{X}$) nebeneinander angeordnet, jeder Vektor als eine Spalte.

$$\bar{X} = [\bar{x}_1 | \bar{x}_2 | ... | \bar{x}_n] \tag{3.15}$$

Jetzt erfolgt die eigentliche Extraktion der Merkmale durch Berechnung der Eigenvektoren v_i und zugehörigen Eigenwerte λ_i der Kovarianzmatrix Ω der Datenmatrix. Die Eigenvektoren einer Abbildung (Matrix) sind jene vom Nullvektor verschiedenen Vektoren, die durch die Abbildung höchstens in ihrer Länge, jedoch nicht in ihrer Richtung verändert werden.

$$\Omega = X * X^T \tag{3.16}$$

$$\Omega v_i = \lambda_i v_i \tag{3.17}$$

Das Berechnen von Eigenvektoren von nicht symmetrischen Matrizen ist nicht trivial und wird hier nicht näher erklärt. Bei der Implementierung der Eigenfaces innerhalb des Frameworks habe ich zu diesem Zweck die C++-Version der Fortran-Bibliothek lapack verwendet. Diese ist auch für kommerziellen Einsatz freigegeben, d.h. lizenzverträglich mit sowohl akademischem als auch kommerziellem Einsatz. Die Berechnung der Eigenvektoren mittels OpenCV ist mir nicht gelungen. Anscheinend unterstützt diese Bibliothek nur die Berechnung der Eigenvektoren von symmetrischen Matrizen.

Zur Berechnung der Eigenvektoren und zugehörigen Eigenwerte muss die Kovarianzmatrix der Datenmatrix berechnet werden. Diese würde jedoch im Fall von Gesichtsbildern eine sehr große Matrix ergeben. Bei z.B. Input-Bildern der Dimension 100 x 100 hätte jedes Bild 10 000 Pixel. Die Datenmatrix hat damit 10 000 Zeilen. Die Kovarianzmatrix $X * X^T$ hat 10 000 Zeilen und 10 000 Spalten und damit 10^8 Elemente. Die Berechnung der Eigenvektoren dieser Matrix ist praktisch undurchführbar.

Bei den Eigenfaces bedient man sich daher eines Tricks. Man berechnet stattdessen die Eigenvektoren und Eigenwerte der Matrix $X^T * X$. Diese hat mit der Dimension 100 x 100 praktikable Ausmaße. Anschließend normalisert man die Eigenvektoren und multipliziert die Eigenmatrix mit der Datenmatrix X.

Diese Vorgehensweise ist aus folgendem Grund möglich:

$$XX^T v_i = \lambda_i v_i \tag{3.18}$$

$$X^T X u_i = \lambda_i u_i \tag{3.19}$$

$$XX^T X u_i = \lambda_i X u_i$$

Das ursprüngliche Eigenwertproblem 3.18 kann auch berechnet werden, indem man das veränderte Eigenwertproblem 3.19 linksseitig mit X multipliziert. Wenn nun u_i ein Eigenvektor von $X^T X$ ist, ist Xu_i ein Eigenvektor von XX^T und damit der gesuchte Eigenvektor v_i.

Es wird daher die "kleine" Kovarianzmatrix verwendet:

$$\Omega = X^T * X \tag{3.20}$$

Die Eigenvektoren werden absteigend nach der Größe ihrer Eigenwerte sortiert und eine bestimmte Anzahl der Vektoren mit den größten Eigenwerten behalten, die übrigen werden verworfen. Eine Möglichkeit, diese Anzahl zu bestimmen, ist die Betrachtung der größten Eigenwerte. Man behält z.B. soviele, dass die Summe der entsprechenden Eigenwerte 90% der Gesamtsumme aller Eigenwerte ausmacht. Damit hat man 90% der Information zur weiteren Verarbeitung übernommen. Die Eigenvektoren werden durch Division mit ihrem Betrag normalisiert und in einer Matrix (Eigenmatrix) angeordnet.

$$v_i = \frac{v_i}{\|v_i\|} \tag{3.21}$$

$$V = [v_1|v_2|...|v_n] \tag{3.22}$$

Abschließend muss die Eigenmatrix, wie schon erwähnt, mit der Datenmatrix multipliziert werden. So erhält man mit viel geringerem Aufwand die größten Eigenvektoren des ursprünglichen Eigenwertproblems 3.18.

$$V = X * V \tag{3.23}$$

Damit ist die Trainingsphase abgeschlossen. Das Ergebnis der Trainingsphase ist die Eigenmatrix.

Figure 3.10: die 4 Eigenfaces mit größten Eigenwerten

Enrollment

Beim Enrollment wird eine bestimmte Anzahl von Gesichtsbildern e_i (in den Testdurchgängen drei) der zu authentifizierenden Person normalisiert und in den Eigenspace projiziert. Man erhält pro Bild für jedes Eigenface (jede Spalte der Eigenmatrix) einen Projektionswert des Vektors. Diese drei Vektoren (Projektionen) p_{e_i} werden für den späteren Vergleich in der Authentifikationsphase gespeichert.

$$\bar{e}_i = e_i - m \tag{3.24}$$

$$p_{e_i} = V^T * \bar{e}_i \tag{3.25}$$

Authentifizierung

Jedes zu authentifizierende Testbild wird normalisiert und in den Eigenspace projiziert.

$$\bar{y} = y - m \tag{3.26}$$

$$p = V^T * \bar{y} \tag{3.27}$$

Die so erzeugte Projektion p wird mit den Projektionen des Enrollments verglichen.

$$distanz = min(DistanzMetrik(p, p_{e_i})) \tag{3.28}$$

Ist die neue Projektion zumindest einer Enrollment-Projektion ähnlich, so wird das Gesicht als "authentifiziert" klassifiziert, ansonsten ist das Ergebnis "nicht erkannt". Wie ähnlich sich die Projektionen im ersten Fall sein müssen, muss empirisch bzw. halbautomatisch ermittelt werden. Man muss einen Schwellwert setzen. Bei jeder Änderung der Bildgröße, der Vorverarbeitung oder Distanz-Metrik muss der Schwellwert von Neuem ermittelt werden.

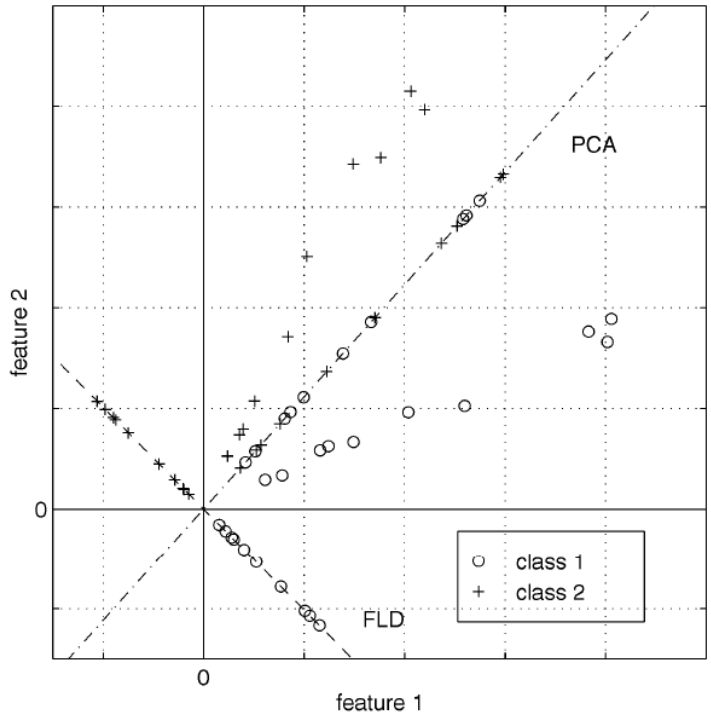

Figure 3.11: PCA vs. FLD [13]

3.3 Fisherfaces

Bei den Fisherfaces [13] handelt es sich wie bei den Eigenfaces um ein holistisches Verfahren, d.h. dass das Bild als Ganzes verarbeitet wird und nicht einzelne Merkmale wie Augen, Mund etc. identifiziert werden. Die Fisherfaces basieren auf der Idee der FLD (Fisher'sche Diskriminanzanalyse). Es wird versucht, eine Trennlinie zwischen unterschiedlichen Klassen (im 2-Dimensionalen eine Gerade, im Mehrdimensionalen eine Hyperebene) zu finden, um neue Objekte je nach Ähnlichkeit einer dieser Klassen zuordnen zu können. Dabei wird die Intraklassenvarianz (Abstand der Objekte innerhalb einer Klasse) minimiert und die Interklassenvarianz (Abstand zwischen den Klassen) maximiert [13].

Figure 3.11 zeigt den Vergleich zwischen PCA und FLD im 2-dimensionalen Fall. Die Objekte mit Klassenzugehörigkeit 1 sind als Kreise dargestellt. Die Objekte mit Zugehörigkeit 2 als Kreuze. Bei der PCA wird nun versucht, die Richtungen mit größter Streuung zu bestimmen (Hauptkomponenten).
Im Gegensatz dazu werden die Objekte bei den Fisherfaces auf eine Gerade projiziert, um eine lineare Trennlinie (im Mehrdimensionalen wäre es eine Hyperebene) bestimmen zu können, die die Klassen möglichst gut voneinander trennt. Die Objekte der Klasse 1 (Kreise) befinden sich rechts unterhalb der Trenngeraden, die Objekte der Klasse 2 (Kreuze) links oberhalb der Geraden. Bei diesem Beispiel ist eine perfekte Zuordnung der Objekte zu Klassen möglich. In der

Praxis hat man natürlich auch Falschzuweisungen, d.h. es befinden sich auch Objekte der Klasse 1 links oberhalb der Trennlinie und analog für Klasse 2.

Im folgenden wird der mathematische Ablauf des Verfahrens erläutert.

Ablauf

Zur Erstellung der Templates ist zuerst eine Trainingsphase notwendig.

Trainingsphase

Die Trainingsobjekte werden manuell in Klassen unterteilt, d.h. die Bilder einer Person bilden Klasse 1, die Bilder einer zweiten Person Klasse 2 usw. Es wird das Gesamtdurchschnittsbild über alle Klassen berechnet.

$$m = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.29}$$

Zusätzlich wird für jede Klasse das klassenweite Durchschnittsbild berechnet.

$$m_c = \frac{1}{|C|} \sum_{x \in C} x \tag{3.30}$$

Jetzt erfolgt die Berechnung der Eigenvektoren v_i und zugehörigen Eigenwerte λ_i der "Within Class Scatter Matrix" und der "Between Class Scatter Matrix". Die Eigenvektoren einer Abbildung (Matrix) sind jene vom Nullvektor verschiedenen Vektoren, die durch die Abbildung höchstens in ihrer Länge, jedoch nicht in ihrer Richtung verändert werden.

Die "Within Class Scatter Matrix" S_w wird gebildet. S_w dient der Minimierung der Unterschiede der Objekte innerhalb einer Klasse, d.h. des Abstandes von Objekten, die derselben Klasse angehören.

$$S_c = \sum_{x \in C} (x - m_c)(x - m_c)^T \tag{3.31}$$

$$S_w = \sum S_c \tag{3.32}$$

Die "Between Class Scatter Matrix" wird berechnet. Ihr Nutzen ist die Maximierung des Abstandes zwischen den unterschiedlichen Klassen. l_c bezeichnet die Anzahl der Elemente der Klasse c.

$$S_b = \sum l_c(m_c - m)(m_c - m)^T \tag{3.33}$$

Die Eigenvektoren der "Within Class Scatter Matrix" und der "Between Class Scatter Matrix" werden berechnet.

$$S_b V = \lambda S_w V \tag{3.34}$$

$$S_b S_w^{-1} V = \lambda S_w S_w^{-1} V$$

$$S_b S_w^{-1} V = \lambda E V$$

$$S_b S_w^{-1} V = \lambda V$$

Bei der Anwendung der FLD in der Gesichtserkennung stößt man auf das Problem, dass S_w singulär ist, d.h. sie besitzt keine inverse Matrix. Das ist der Fall, da $D > N - c$ immer erfüllt ist.
D ... Anzahl der Pixel der Input-Bilder
N ... Anzahl der Trainingsbilder
c ... Anzahl der Klassen

Um dieses Problem zu umgehen, wendet man bei den Fisherfaces vor der FLD PCA an, um die Dimension der Input-Daten auf N-c zu reduzieren. Damit existiert eine inverse Matrix zu Sw, weil $D == N - c$. Wie bei den Eigenfaces werden die Bilder gemittelt, in einer Datenmatrix angeordnet, und die Eigenvektoren und Eigenwerte der "kleinen" Kovarianzmatrix werden berechnet.

$$\bar{x}_i = x_i - m$$

$$\bar{X} = [\bar{x}_1|\bar{x}_2|...|\bar{x}_n] \tag{3.35}$$

$$\Omega = X^T * X \tag{3.36}$$

$$\Omega v_i = \lambda_i v_i \tag{3.37}$$

Die N-c größten Eigenvektoren werden behalten. Sie werden normalisiert, in einer Eigenmatrix angeordnet und diese mit der Datenmatrix multipliziert.

$$v_i = \frac{v_i}{\|v_i\|} \tag{3.38}$$

$$V = [v_1|v_2|...|v_n] \tag{3.39}$$

$$V = X * V \tag{3.40}$$

Die Input-Bilder werden zuerst in diesen gerade berechneten Eigenspace projiziert. So erreichen wir, dass die Dimension des Inputs für die folgende Eigenberechnung N-c ist und somit eine inverse Matrix zu S_w existiert!

$$\bar{e}_i = e_i - m \tag{3.41}$$

$$p_{e_i} = V^T * \bar{e}_i \tag{3.42}$$

Wie oben angegeben wird die "Between Class Scatter Matrix" und die "Within Class Scatter Matrix" berechnet.
Das 2. Eigenwertproblem ist dann:

$$S_b S_w^{-1} W = \lambda W$$

Die c-1 größten Eigenvektoren werden in der Fishermatrix F zusammengefasst und diese normalisiert.
Die Projektionen der obigen Eigenmatrix (V) werden in diesen Fisherspace projiziert und die Ähnlichkeit zwischen einem Testbild und bekannten Gesichtern durch eine Vergleichsfunktion bestimmt. In den Tests wurde die einfache Euklidische-Distanz verwendet.

$$f_{e_i} = F^T * p_{e_i} \tag{3.43}$$

$$distanz = min(DistanzMetrik(someProjection, f_{e_i})) \tag{3.44}$$

3.4 Elastic Bunch Graph Matching

Elastic Bunch Graph Matching (EBGM) ist ein 1997 vorgestelltes Verfahren [30], das im Gegensatz zu den anderen behandelten Ansätzen zu den merkmalsbasierten zählt, d.h. dass einzelne markante Punkte im Gesichtsbild gesucht werden (Ecken der Augen, Nasenspitze etc.), und diese und die umgebenden Bereiche für den Vergleich verarbeitet werden.

Gabor Wavelets

EBGM basiert auf den Gabor Wavelets. Sie werden in der Bildverarbeitung eingesetzt, um ein Bild in seinen Frequenzbereich zu transformieren. Um einen möglichst großen Teil der Information abzudecken, werden unterschiedliche Wavelets mit unterschiedlichen Parameteren verwendet. Da es sich bei Gesichtsbildern um 2-dimensionale Daten handelt, verwendet EBGM die 2-dimensionale Form der Gabor Wavelets. Diese sind folgendermaßen definiert:

$$W(x, y, \Theta, \lambda, \varphi, \sigma, \gamma) = e^{-\frac{x'^2+y^2y'^2}{2\sigma^2}} cos(2\pi\frac{x'}{\lambda} + \varphi) \tag{3.45}$$

$$x' = xcos(\Theta) + ysin(\Theta) \tag{3.46}$$

$$y' = -xsin(\Theta) + ycos(\Theta) \tag{3.47}$$

Mithilfe dieser Gleichungen können nun alle Punkte (x, y) der Gabor-Masken berechnet werden. Die Paramter sind dabei Orientierung Θ, Frequenz λ, Phase φ, Größe der Wavelets σ und Seitenverhältnis γ. Anschließend ist die Parameterbelegung nach Wiskott [30] und Bolme [31] dargestellt. Sie unterscheiden sich lediglich in den Werten von φ und σ. In den Testsdurchgängen wurde letztere verwendet.

Paramterbelegung nach Wiskott:

$$\Theta = \{0, \tfrac{\pi}{8}, \tfrac{2\pi}{8}, \tfrac{3\pi}{8}, \tfrac{4\pi}{8}, \tfrac{5\pi}{8}, \tfrac{6\pi}{8}, \tfrac{7\pi}{8}\}$$

$$\lambda = \{4, 4\sqrt{2}, 8, 8\sqrt{2}, 16\}$$

$$\varphi = \{0, \tfrac{\pi}{2}\}$$

$$\sigma = \lambda$$

$$\gamma = 1$$

Parameterbelegung nach Bolme:

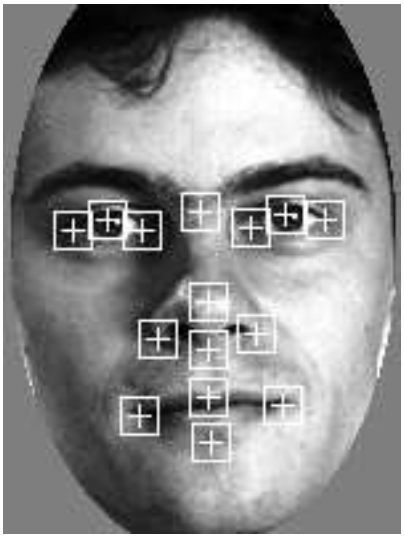

Figure 3.12: Bildbereich für die Berechnung der Jets bei Maskengröße 11 [42]

$$\Theta = \{0, \tfrac{\pi}{8}, \tfrac{2\pi}{8}, \tfrac{3\pi}{8}, \tfrac{4\pi}{8}, \tfrac{5\pi}{8}, \tfrac{6\pi}{8}, \tfrac{7\pi}{8}\}$$

$$\lambda = \{4, 4\sqrt{2}, 8, 8\sqrt{2}, 16\}$$

$$\varphi = \{-\tfrac{\pi}{4}, \tfrac{\pi}{4}\}$$

$$\sigma = \tfrac{3\lambda}{4}$$

$$\gamma = 1$$

Durch die 8 Orientierungen, 5 Frequenzen und 2 Phasen erhält man 80 Gabor-Masken.

Berechnung der Jets

Zur Informationsextraktion werden alle Masken auf alle detektierten Bildbereiche angewandt (Convolution - Faltung). Figure 3.12 zeigt die Bildbereiche um die Knoten, die Einfluss auf die Werte der Jets haben. Es ist anzumerken, dass nicht bei allen Masken der eingezeichnete Bildbereich verarbeitet wird, sondern sich der Bereich je nach Maskengröße unterscheidet.

Das Zentrum der Maske wird dabei auf den Knoten gelegt und die Produkte aus Maskenpunkt und Bildpunkt für alle Maskenpunkte werden aufsummiert. Die Summe wird dem Knoten als ein Koeffizient zugewiesen, exemplarisch zu sehen in Figure 3.13.

Auf diese Weise erhält man pro Knoten 80 Koeffizienten (für die 80 Masken), 40 reale und 40 imaginäre. Die später erläuterten Vergleichsfunktionen arbeiten jedoch mit den Polarkoordinaten der Koeffizienten. Daher wird für je 2 Koeffizienten (je ein realer und ein imaginärer,

86	93	107	108
83	87	102	99
84	91	95	89
83	87	101	90

80	103	72
90	91	75
91	92	80

	70816		

86*80 + 93*103 + 107*72 + 83*90 + 87*91 +102*75 + 84*91 + 91*92 + 95*80 = 70816

86	93	107	108
83	87	102	99
84	91	95	89
83	87	101	90

80	103	72
90	91	75
91	92	80

	70816	**74915**	

93*80 + 107*103 + 108*72 + 87*90 + 102*91 + 99*75 + 91*91 + 95*92 + 89*80 = 74915

Figure 3.13: Anwendung einer Maske (Mitte) auf ein Bild (links)

deren Parameter sich nur im Wert der Phase unterscheiden) die Polardarstellung berechnet. Die Zahlen (a, ϕ) nennt man Polarkoordinaten einer komplexen Zahl z.

$$z = x + iy = a(cos\phi) + isin\phi) \tag{3.48}$$

Wie schon gesagt, erhält man durch die Anwendung der Masken 40 reale und 40 imaginäre Koeffizienten:

$$a_{real} = acos(\phi) \tag{3.49}$$

$$a_{imag} = asin(\phi)$$

Aus den Koeffizienten lassen sich die Polarkoordinaten folgendermaßen berechnen:

$$a = \sqrt{a_{real}^2 + a_{imag}^2} \tag{3.50}$$

$$\phi = \begin{cases} arctan(\frac{a_{imag}}{a_{real}}) & a_{real} > 0 \\ \pi + arctan(\frac{a_{imag}}{a_{real}}) & a_{real} < 0 \\ \frac{\pi}{2} & a_{real} = 0 \,\&\, a_{imag} >= 0 \\ -\frac{\pi}{2} & a_{real} = 0 \,\&\, a_{imag} < 0 \end{cases}$$

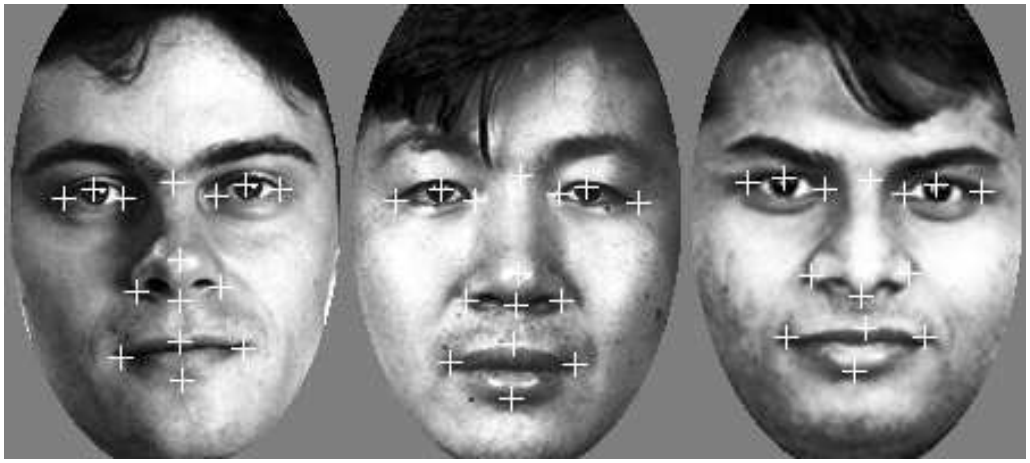

Figure 3.14: manuell gesetze Knoten [42]

Diese Polarkoordinaten werden als sogenannter Jet für jeden Knoten gespeichert. Man erhält pro Knoten 40 Jets (mit Magnitude a und Angle ϕ).

Ablauf

Die zu erfolgenden Schritte zur Authentifikation eines Gesichtsbildes lassen sich in Trainingsphase, Enrollment und Authenifizierungsvorgang unterteilen.

Trainingsphase

Bei allen Trainingsbildern werden manuell (oder halbautomatisch) die wichtigsten Punkte im Gesicht markiert. Für die Tests setzte ich manuell bei 88 Trainingsbildern je 15 Knotenpositionen wie aus Figure 3.14 ersichtlich.

Aus je einem Trainingsbild wird ein Facegraph erzeugt. Er enthält die Jets der 15 Knoten (siehe 6.2). Aus den 88 Facegraphen wird ein Bunchgraph erzeugt. Er enthält ebenfalls 15 Knoten (mit den durchschnittlichen Knotenkoordinaten der entsprechenden Knoten der Facegraphen) und zu jedem Knoten alle 88 Jets der Trainings-Facegraphen. Diese Daten werden später beim Finden eines Facegraphen in einem neuen Bild benötigt. Das Ergebnis der Trainingsphase ist der Bunchgraph. Dieser kann einmalig berechnet werden, weil er unabhängig von den Input-Bildern ist.

Finden eines Facegraphen im neuen Bild

Für ein neues Gesichtsbild wird mithilfe des Bunchgraphen ein Facegraph erstellt. Zuerst wird mit Hilfe der durchschnittlichen Knotenkoordinaten ein grober Facegraph positioniert. Die Position der Knoten wird durch Vergleich mit den Jets der 88 Trainings-Facegraphen verfeinert.

Dabei werden neue Jets extrahiert und mit den im Bunchgraph bestehenden verglichen. Als Displacement Estimator wurde dazu von mir bei allen Testläufen NarrowingLocalSearch verwendet. Abschließend werden die Jets an den endgültigen Knotenpositionen extrahiert und als neuer Facegraph gespeichert. Das Ergebnis dieser Phase ist ein möglichst gut positionierter Facegraph mit den entsprechenden Jets.

Enrollment

Für 3 ausgewählte Bilder der zu authentifizierenden Person wird ein Facegraph erstellt (siehe "Finden eines Facegraphen im Bild"). Die zugehörigen Jets werden für den späteren Vergleich gespeichert.

Authentifizierung

Für das neue Bild wird ein Facegraph erstellt. Dieser wird mit den Enrollment-Facegraphen durch eine Vergleichsfunktion auf Ähnlichkeit untersucht. Ist die Ähnlichkeit über einem bestimmten Schwellwert, so wird die Person als erkannt klassifiziert, sonst als nicht erkannt.

Vergleichsfunktionen

Mithilfe der Vergleichsfunktionen können einzelne Jets, Facegraphen mit Facegraphen, oder Facegraphen mit dem Bunchgraphen verglichen werden. Diese werden im Rahmen des Algorithmus an verschiedenen Stellen eingesetzt. Zuerst beim Finden eines Facegraphen im Bild und später beim Vergleich eines neuen Facegraphen mit den Enrollment-Facegraphen. Es existieren verschiedene Vergleichsfunktionen. Allen ist gemeinsam, dass sie die Polarkoordinaten der Jets verarbeiten und dass das Ergebnis zwischen 0 und 1 liegt. Je näher bei 1, desto ähnlicher sind sich die Jets. Alle Vergleichsfunktionen vergleichen 2 Jets, zum Vergleich von Facegraphen wird das Ergebnis über alle zugehörigen Jets aufsummiert.

Vergleichsfunktion "normalisierte Kreuzrelation" (Angle ϕ wird ignoriert!)

$$S_a(J, J') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}} \tag{3.51}$$

Vergleichsfunktion, bei der Angle miteinfließt, λ und Θ bezeichnen die Parameter der Maske, mithilfe deren die Koeffizienten berechnet wurden. Der Abstandsvektor $\vec{d}$ muss durch Maximierung der Vergleichsfunktion (siehe nächstes Kapitel) geschätzt werden.

$$S_\phi(J, J') = \frac{\sum_j a_j a'_j cos(\phi_j - \phi'_j - \vec{d}\vec{k}_j)}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}} \tag{3.52}$$

$$\vec{k} = \begin{pmatrix} \frac{2\pi cos(\Theta)}{\lambda} \\ \frac{2\pi sin(\Theta)}{\lambda} \end{pmatrix}$$

Displacement Estimator

Es gibt mehrere Möglichkeiten, den Abstandsvektor $\vec{d}$ in der Vergleichsfunktion S_ϕ zu schätzen. Allen Ansätzen ist gemein, dass der Abstandsvektor so gewählt wird, dass S_ϕ maximal ist. Die Colorado State University implementierte in ihrem Gesichtserkennungsframework CSU [43] verschiedene Methoden. Diese unterscheiden sich teilweise beträchtlich in Aufwand und Qualität. Im Rahmen der Diplomarbeit wurden alle Methoden umgesetzt und ausführlich getestet:

Narrowing Local Search
Predictive Step
Predictive Iter
Grid Sample
Fixed Local Search

In der Evaluierung beschränke ich mich auf die Ergebnisse von Predictive Step, Predictive Iter und Narrowing Local Search, weil diese das beste Aufwand/Nutzen Verhältnis haben.

3.5 Enhanced Local Gabor Binary Patterns Histogram Sequence

Enhanced Local Gabor Binary Patterns Histogram Sequence (ELGBPHS) wurde in [11] vorgestellt. Es handelt sich dabei um die Weiterentwicklung der Idee der Local Gabor Binary Patterns Histogram Sequence (LGBPHS) aus [32]. Der Algorithmus gehört wie die Eigenfaces und Fisherfaces zu den holistischen Verfahren und basiert wie einige andere Verfahren auch auf den Gabor Wavelets. Eine Besonderheit dieses Ansatzes ist, dass auch die Phase-Information der Gabor-Koeffizienten für das Matching herangezogen wird. Bei den anderen (früheren) Verfahren wie z.B. Dynamic Link Architecture (LDA) oder Elastic Bunch Graph Matching (EBGM) wurde diese verworfen und nur die Magnitude-Information weiterverarbeitet. Der Grund dafür ist, dass sich die Phase bei nur geringen Positionsänderungen stark verändern kann. Das bedeutet, das die Information bei ein und demselben Merkmal bei geringer Positionsänderung sehr unterschiedlich sein kann, obwohl es sich um dasselbe lokale Merkmal handelt. Bei Object Matching ist das verständlicherweise sehr problematisch.

Zhang et al. lösten dieses Problem, indem diese Daten (Phase-Information der einzelnen Pixels)

nicht direkt verglichen werden, sondern größere Bildbereiche (der Magnitude und Phase Maps) durch den LBP (Local Binary Patterns) Operator modelliert werden. Die Anwendung dieser Texturbeschreibungs-Methode wirkt dem genannten Problem entgegen.

Ablauf

Zuerst werden die Gabor Wavelets (8 Orientierungen, 5 Skalierungen, 2 Phasen) auf die Input-Bilder angewandt und man erhält 40 Magnitude und 40 Phase Maps (für die 80 Gabor Masken). Für weitergehende Information zu Gabor Wavelets siehe Kapitel 6.1. Diese Maps können wieder als Bilder aufgefasst werden, auf die nachfolgend der LBP Operator angewendet wird. LBP vergleicht den betrachteten Pixel mit seiner Nachbarschaft, fasst das Ergebnis als Zahl auf und weist dem betrachteten Pixel diesen Wert zu. Die LBP-Bilder werden in Sub-Bilder unterteilt. Abschließend werden Histogramme der Sub-Bilder berechnet, die durch Histogramm-Intersection auf Ähnlichkeit verglichen werden können. Durch die Gewichtung der Unterblöcke ist es möglich, bestimmte Bereiche des Gesichts durch höhere Gewichtung zu betonen oder andere durch Gewicht 0 völlig zu ignorieren. So haben bestimmte Bereiche wie z.B. die Augen, die für die erfolgreiche Gesichtserkennung wichtiger sind, größeren Einfluss auf den resultierenden Ähnlichkeitswert.

Local Binary Patterns

LBP ist eine einfache und zugleich mächtige Texturebeschreibungsmethode, die invariant gegen globale Helligkeitsunterschiede ist. Ausschlaggebend für den LBP-Wert ist nicht die globale Helligkeit (die Größe der Werte der Grauwerte), sondern der relative Unterschied zwischen den Pixeln, genauer zwischen dem betrachteten Pixel und seiner Umgebung. Im einfachsten Fall vergleicht LBP den Zentrumspixel mit den 8 angrenzenden Nachbarn und weist das Ergebnis dem betrachteten Pixel zu. Diese Vergleiche werden für alle Pixel des Input-Bildes durchgeführt.

$$S(f_p - f_c) = \begin{cases} 1, & f_p \geq f_c \\ 0, & f_p < f_c \end{cases} \tag{3.53}$$

$$LBP = \sum_{p=0}^{7} S(f_p - f_c)2^p \tag{3.54}$$

Formel 7.1 und 7.2 zeigen die Anwendung von LBP. f_c bezeichnet den Zentrumspixel, f_p die Nachbarpixel. f_c wird mit seinen 8 Nachbarn verglichen. Ist der Grauwert des Nachbarn größer als f_c , liefert $S()$ eine 1, sonst 0. Dieses binäre Ergebnis wird durch den Faktor 2^p schließlich als ganze Zahl aufgefasst und dem Ergebnispixel zugewiesen.

86	93	107
83	87	102
84	91	95

0	1	1
0	87	1
0	1	1

2^0	2^1	2^2
2^3	**214**	2^4
2^5	2^6	2^7

$11010110 = 2^7 + 2^6 + 2^4 + 2^2 + 2^1 = 214$

Figure 3.15: LBP angewendet auf Bild

Zur weiteren Verdeutlichung zeigt 3.15 ein einfaches Beispiel. Der Zentrumspixel 87 wird mit seinen Nachbarn verglichen und das Ergebnis als binäre Zahl aufgefasst.
Es muss hier angemerkt werden, dass die Faktoren 2^p nicht unbedingt in dieser Anordnung (von links oben nach rechts unten) zugewiesen werden müssen. Jede andere Reihenfolge ist genausogut. Es ist nur wichtig, dass innerhalb des Algorithmus für alle Anwendungen von LBP auf unterschiedliche Input-Bilder dieselbe Anordnung verwendet wird.

Vergleichsfunktion

Die Ähnlichkeit der erstellten Histogramme wird durch die Vergleichsfunktion bestimmt. Dabei wird für alle Orientierungen θ, Skalierungen λ, beide Phasen und alle Unterbereiche r der gewichtete Wert der Histogram-Intersection der entsprechenden Histogramme aufsummiert. Es handelt sich dabei im Gegensatz zur Euklid-Distanz um einen Ähnlichkeitswert, d.h. je größer der Wert, desto ähnlicher sind sich die Eingangsbilder. Formel 7.3 zeigt getrennt den Ähnlichkeitswert füer die Phase und die Magnitude. Sie werden jedoch durch einfaches Aufsummieren kombiniert.

$$\Re = (\Re^m, \Re^p) \tag{3.55}$$

$$S(\Re_1, \Re_2) = S^m(\Re_1, \Re_2) + S^p(\Re_1, \Re_2)$$

$$S^m(\Re_1, \Re_2) = \sum_{\theta=0}^{7} \sum_{\lambda=0}^{4} \sum_{r=0}^{R-1} W^m_{\theta,\lambda,r} \Psi(H1^m_{\theta,\lambda,r}, H2^m_{\theta,\lambda,r})$$

$$S^p(\Re_1, \Re_2) = \sum_{\theta=0}^{7} \sum_{\lambda=0}^{4} \sum_{r=0}^{R-1} W^p_{\theta,\lambda,r} \Psi(H1^p_{\theta,\lambda,r}, H2^p_{\theta,\lambda,r})$$

$\Psi(.)$ bezeichnet die Histogramm-Intersection der Histogramme $H1$ und $H2$ der entsprechenden Orientierung θ, Skalierung λ und des Unterbereichs r. $W^p_{\theta,\lambda,r}$ ist analog die Gewichtung für dieselben Parameter. Es muss angemerkt werden, dass die Gewichtung im Rahmen dieser Arbeit nur für die einzelnen Unterbereiche unterschiedlich gesetzt wurde, d.h. die Gewichtung für einen Unterbereich r ist für alle Werte der anderen Parameter derselbe. Daher könnte man diesen Teil der Formel durch W_r ersetzen.

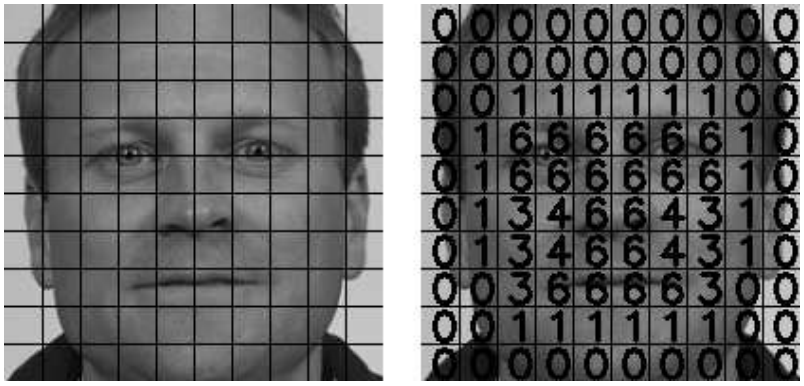

Figure 3.16: Links: Subblöcke, rechts: Gewichtung der Subblöcke [42]

Gewichtung

Durch die Gewichtung der einzelnen Sub-Bilder lassen sich bestimmte Bereiche betonen und durch Setzen der Gewichtung auf 0 ignorieren. Im Laufe der Tests hat sich folgende Unterteilung und Gewichtung als Verbesserung erwiesen (im Vergleich zur ungewichteten Intersection der Histogramme). Das Eingangsbild der Größe 100 x 100 wird dabei in 100 Unterbilder der Größe 10 x 10 unterteilt. Die Gewichtsfaktoren werden wie aus Figure 3.16 ersichtlich gesetzt. Wie aus der Formel zu sehen ist, lassen sich auch bestimmte Orientierungen, Skalierungen und Phasen stärker oder weniger stark gewichten. Dies wurde jedoch in den Tests nicht angewandt. Die für die Gesichtserkennung entscheidenden Bereiche im Zentrum des Gesichts, nämlich die Augen, Nase und Teile des Mundes werden betont (durch Setzen auf 3 - 6). Irrelevante Abschnitte wie der Hintergrund und der Rand des Gesichts werden schwächer gewichtet (durch Setzen auf 1) oder vollständig ignoriert (durch Setzten auf 0). Natürlich werden in der Implementierung aus Effizienzgründen die Histogramme der Bereiche mit Gewichtung 0 gar nicht erzeugt und nur diese mit Gewichtung ungleich 0 für die weitere Verarbeitung herangezogen.

3.6 MarliesFaces - ein neues Verfahren

Im Rahmen meiner Arbeit im Projekt FaceMOC an der FH St. Pölten ist in Zusammenarbeit mit Mag. DI Marlies Rybnicek ein neues Verfahren entstanden. Das Projektziel war die Entwicklung eines Gesichtserkennungssystems, das in der sehr beschränkten Umgebung einer Massen-Smartcard verwendbar ist.

Die Voraussetzungen waren:
RAM-Bedarf: max 4 KB
Storage-Bedarf: max 40 KB, je weniger desto besser
Erkennungsrate: > 90%
Dauer des Authentifizierungsvorgangs: wenige Sekunden

Das Projektziel wurde durch die Kombination von ELBGBPS mit PCA erreicht. Das Verfahren hat den Namen MarliesFaces.

Trainingsphase

In der Trainingsphase werden 88 Trainingsbilder der FERET Datenbank durch die Gabor Wavelets (8 Orientierungen, 5 Skalierungen, 2 Phasen) (siehe Kapitel 6.1) verarbeitet. Man erhält pro Trainingsbild 40 Magnitude und 40 Phase Maps (für die 80 Gabor Masken) mit derselben Dimension wie die Input-Bilder (100 x 100). Diese Maps werden in 10 x 10 große Subbereiche unterteilt. Mittels LBP wird die Information der einzelnen Bereiche modelliert (siehe Kapitel 7.2). Anschließend werden Histogramme der LBP-Subbereiche berechnet und diese als Trainingsbilder für die PCA angeordnet, d.h.: Für jedes Trainingsbild erhalten wir 100 x 80 (100 Unterbereiche, 80 Masken) = 8000 Histogramme. Nur die Hälfte der Histogramme wird weitergehend verarbeitet, die andere Hälfte (vor allem der Rand der Gesichtsbilder, die Haare und andere irrelevante Bereiche) werden verworfen.

Die entsprechenden Histogramme aller Trainigsbilder bilden die PCA-Bilder. Pro Trainingsbild und Bereich gibt es 80 Histogramme (aufgrund der Faltung mit den 80 Gabor Masken). Ein PCA-Bild (Vektor) besteht aus den nacheinander angeschriebenen Histogramm-Anzahl-Werten aller 80 Faltungen eines Trainingsbildes und eines Bereiches. Wir erhalten so 88 PCA-Bilder pro Bereich. Ein PCA-Bild besteht aus 88 x 80 = 7040 Histogrammen. Diese neuen "Trainingsbilder" sind der Input für die nachfolgende PCA. Pro Bereich (insgesamt 50) berechnen wir eine Eigenmatrix mit zugehörigen Eigenwerten. Die Eigenmatrizen werden für die spätere Projektion der Testbilder gespeichert.

Enrollment

Die Enrollment-Bilder werden mit den 80 Gabor Masken gefaltet, darauf LBP angewendet und die resultierenden Histogramme für jeden Bereich als Vektor angeordnet. Nach der Mittelung durch Subtraktion der Durchschnittsbilder werden die Vektoren (50, weil 50 Bereiche) in die entsprechenden 50 Eigenmatrizen projeziert. Die Projektionen werden für den späteren Vergleich gespeichert.

Authentifizierungsphase

Genau wie in der Trainingsphase wird das Input-Bild mit den 80 Gabor Masken gefaltet, darauf LBP angewendet und die resultierenden Histogramme für jeden Bereich als Vektor angeordnet. Nach der Mittellung durch Subtraktion der Durchschnittsbilder werden die Vektoren in die entsprechenden 50 Eigenmatrizen projeziert. Durch Euklidische Distanz wird die Ähnlichkeit zwischen den Enrollment-Templates und dem Test-Template bestimmt. Anhand eines vorher gesetzten Schwellwertes wird entschieden, ob die Person als "erkannt" oder "nicht erkannt" klassifiziert wird.

CHAPTER 4

Evaluierung

Alle Vorverarbeitungsschritte und Gesichtserkennungsverfahren wurden in C++ implementiert.

Zur Ermittlung der Erkennungsrate wurden die Frontalbider der frei verfügbaren FERET Gesichtsdatenbank [42] verwendet. Insgesamt ca. 2700 Bilder von 348 Personen, im Schnitt 8 Bilder pro Person. Anhand dieser Bilder wurden zufällig 500 "statische Durchgänge" erstellt. Ein Durchgang besteht aus Enrollment einer Person, 20 Reject-Tests mit Bildern anderer Personen und ca. 4 Accept-Tests mit anderen Fotos derselben Person. Die Falschzuweisungen werden dabei aufgezeichnet, um daraus die FAR und FRR zu berechnen. Die FAR ist der Prozentsatz der falsch als "authentifiziert" klassifizierten Tests (Bilder von fremden Person, die falsch als die Enrollment-Person klassifiziert werden). Die FRR ist der Prozentsatz der falsch als "nicht authentifiziert" klassifizierten Gesichter (Bilder der Enrollment-Person, die falsch als fremde Personen klassifiziert werden).

Der RAM-Bedarf wurde durch Analyse der benötigten Datenstrukturen der Implementierung ermittelt. Der größte Platzbedarf der für die Ausführung gleichzeitig im Speicher gehaltenen Variablen bildet den Speicherbedarf. Betriebssystem-Overhead wurde dabei nicht berücksichtigt. Der tatsächliche RAM-Bedarf wird daher etwas darüber liegen.

Der Festspeicher-Bedarf wurde durch die Betrachtung der Datenstrukturen der zu speichernden Daten berechnet, beim Training die Trainingsdaten und beim Matching die 3 Enrollment-Templates.

Der Rechenaufwand (CPU-Zyklen, Ausführungszeit) wurde mithilfe der Funktion QueryPerformanceCounter() der Windows-API ermittelt. Diese hat eine Auflösung von 1/1193182 Sekunden, ist daher hinreichend genau. Vor dem Beginn des zu analysierenden Code-Blocks wird QueryPerformanceCounter() aufgerufen, und direkt danach. Die Ausführungsdauer des Code-Blocks kann so gemessen werden. Die entsprechenden Code-Blöcke werden etwa 100 mal mit unterschiedlichen Eingabeparametern ausgeführt und die durchschnittliche Dauer berech-

net. Die CPU-Zyklen ergeben sich dann durch die Takt-Frequenz der CPU und die ermittelte Dauer.

Zusammengefasst:

- die Erkennungsrate (FAR und FRR) wird durch die Ausführung von 500 statischen Durchgängen, das sind etwa 2000 Accept-Tests und 10000 Reject-Tests, berechnet.
- der RAM-Bedarf ergibt sich durch Betrachtung der für die Ausführung notwendigen Datenstrukturen.
- der Festspeicherbedarf wird durch die Analyse der Datenstrukturen der zu speichernden Daten berechnet.
- Die Ausführungszeit und die notwendigen CPU-Zyklen werden durch Messung der Dauer der Code-Blöcke mithilfe der Funktion QueryPerformanceCounter() der Windows-API ermittelt.

Im folgenden wird der Speicherbedarf und Rechenaufwand aller Vorverarbeitungsmethoden und Gesichtserkennungsverfahren dargelegt.

Legende der Evaluierungstabellen

Vorverarbeitung:
without: Face Detection - Resize
HE: Face Detection - Resize - Elliptic - Histogram Equalization
MSR: Face Detection - Resize - Elliptic - Multi-Scale Retinex - Elliptic
Daub4: Face Detection - Resize - Elliptic - Daubechies Wavelet Transformation - Elliptic
TT: Face Detection - TT (Gamma Correction - Difference of Gaussian Filtering - Contrast Equalization) - Resize

Trainingsphase: Berechnung der Trainingsdaten
Authenticate 1: 1 Gesichtsbild verifizieren
Distance Function 3: die Berechnung der Ähnlichkeit zwischen den 3 Enrollment-Templates und dem Test-Template (3x Distanzfunktion)

4.1 Face Detection

Memory Requirements

Da für die Face Detection der fertige Algorithmus der OpenCV Library verwendet wurde, kann der RAM-Bedarf nicht genau beziffert werden.

Face Detection	Memory Requirements
Bedarf	?

Table 4.1: Haar Memory Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Haar	246498	251628000	89.87 ms	251.63 ms	6290.9 ms

Table 4.2: Haar Computing Time

4.2 Resize

Memory Requirements

Für Resize müssen 2 Bilder der Dimension des Ursprungsbildes (100 x 100) im Speicher sein, somit 19.5 KB.

Resize	Memory Requirements
Bedarf	19.5 KB

Table 4.3: Resize Memory Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Resize	4050	4134440	1.48 ms	4.14 ms	103.6 ms

Table 4.4: Resize Computing Time

4.3 Elliptisches Zuschneiden

Memory Requirements

Für das elliptische Zuschneiden müssen 2 Bilder der Dimension des Ursprungsbildes (100 x 100) im Speicher sein, soit 19.5 KB

Elliptic	Memory Requirements
Bedarf	19.5 KB

Table 4.5: Elliptic Memory Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Elliptic	3432	3504050	1.25 ms	3.50 ms	87.50 ms

Table 4.6: Elliptic Computing Time

4.4 Histogram Equalization

Memory Requirements

Im Nachfolgenden ist der RAM-Bedarf der Vorverarbeitungsmethode Histogram Equalization bei der Verarbeitung eines Graustufenbildes der Dimension 100 x 100 dargelegt.

Das Bild verbraucht 100 x 100 Bytes = 9.7 KB. Zur Zählung der Grauwerte werden 256 ganze Zahlen (int mit 4 Byte) benötigt und zur Berechnung der kumulativen Summen weitere 256 ganze Zahlen. Man sieht, dass die Methode mit extrem geringem Speicherbedarf punktet. Das meiste wird für das Halten des zu verarbeitenden Bildes im Speicher benötigt.

gesamter RAM-Bedarf: 100 x 100 + 256 x 4 + 256 x 4 = 11.7 KB

Das ist mit aktueller Hardware bei PCs und Handys kein Problem. Für die Anwendung der Methode auf einer Massen-Smart-Card jedoch viel zu viel (typischer RAM: 4 KB). Jedoch kann die Vorverarbeitung bei der Implementierung für Smart-Cards normalerweise ausgelagert und z.B. vom Terminal erledigt werden.

HE	Memory Requirements
Bedarf	11.7 KB

Table 4.7: HE Memory Requirements

Computing time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
HE	3924	4005950	1.43 ms	4.00 ms	100.1 ms

Table 4.8: HE Computing Time

4.5 Multi-Scale Retinex

Memory Requirements

Im nachfolgenden ist der RAM-Bedarf der Vorverarbeitungsmethode Multi-Scale-Retinex auf ein Graustufenbild der Dimension 100 x 100 erklärt.

Zu gleicher Zeit muss das zu verarbeitende Bild, die 4 GausMasken, die 4 durch die Anwendung der Gaus-Masken auf das Ursprungsbild erzeugten gefilterten Bilder und das Ergebnisbild im Speicher gehalten werden.

Die Gaus-Masken haben bei den gewählten Parametern eine Größe von 9 bzw. 11 Gleitkomma-Zahlen (double mit 8 bytes), somit insgesamt 2 x 9 x 8 + 2 x 11 x 8 Bytes (zwei Masken der Größe 9 und zwei Masken der Größe 11). Das ist vernachlässigbar.

Das Ursprungsbild benötigt 100 x 100 Bytes. Das Ergebnisbild und die gefilterten Bilder ebenfalls je 100 x 100 Bytes, somit insgesamt 6 x 100 x 100 = 58.6 KB.

Die für die Berechnungen notwendigen Datenstrukturen (hauptsächlich einige Gleitkommazahlen) sind vernachlässigbar.

MSR	Memory Requirements
Bedarf	58.6 KB

Table 4.9: MSR Memory Requirements

Der RAM-Bedarf ist für aktuelle Hardware bei PCs und Handys kein Problem, auch bei der

gleichzeitigen Verarbeitung von vielen Bildern. Nur in sehr eingeschränkten Umgebungen kann es dabei Probleme geben.

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
MSR	24704	25218400	9.00 ms	25.2 ms	630 ms

Table 4.10: MSR Computing Time

4.6 Daubechies-Wavelet-Transformation

Memory Requirements

Im nachfolgenden ist der RAM-Bedarf der Vorverarbeitungsmethode Daubechies-Wavelet-Transformation bei der Anwendung auf ein Graustufenbild der Dimension 100 x 100 dargelegt.
Gleichzeitig müssen maximal 2 Bilder der Dimension des Ursprungsbildes und 2 der halben Größe im Speicher gehalten werden.

damit gesamter RAM-Bedarf: 2 x 100 x 100 + 2 x 100 x 50 = 29.2 KB.

Daub4	Memory Requirements
Bedarf	29.2 KB

Table 4.11: Daub4 Memory Requirements

Das stellt für heutige Hardware kein Problem dar.

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Daub4	9522	9720580	3.47 ms	9.72 ms	242.9 ms

Table 4.12: Daub4 Computing Time

4.7 TT

Memory Requirements

Im nachfolgenden ist der RAM-Bedarf der Vorverarbeitungsmethode TT bei der Verarbeitung eines Graustufenbildes der Dimension 100 x 100 dargelegt.

Für Gamma Correction müssen 2 Bilder im Speicher gehalten werden, somit 2 x 100 x 100 = 19.5 KB.

Difference of Gaussian Filtering benötigt 2 Gaus-Masken, dieser Bedarf ist vernachlässigbar. Weiters müssen gleichzeitig 4 Bilder der Dimension des Ursprungsbildes verfügbar seien, somit 4 x 100 x 100 = 39.1 KB.

Zur Berechnung der Contrast Equalization müssen 2 Bilder und 200 Gleitkommahlen (double 8 bytes) gleichzeitig im Speicher seien, damit 2 x 100 x 100 + 200 x 8 = 21.1 KB.

Gamma Correction, Difference of Gaussian Filtering und Contrast Equalization werden unabhängig voneinander hintereinander ausgeführt. Der maximal RAM-Bedarf ist somit 39.1 KB.

TT	Memory Requirements
Bedarf	39.1 KB

Table 4.13: TT Memory Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Verarbeitung eines Graustufen-Bildes der Dimension 100 x 100 dargelegt.

Computing Time	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
TT	18434	18817900	6.72 ms	18.82 ms	470.4 ms

Table 4.14: TT Computing Time

4.8 Eigenfaces

Memory Requirements

Im Nachfolgenden ist der RAM-Bedarf der Eigenfaces dargelegt.

Beim Training müssen alle Trainingsbilder (hier 88) gleichzeitig im Speicher sein, somit 88 x 100 x 100 Bytes = 859 KB. Das Durchschnittsbild hat dieselben Dimensionen, 100 x 100 Bytes = 9.8 KB. Die Datenmatrix (double 8 bytes) hat die gemittelten Trainingsbilder als Spalten, 88 x 100 x 100 x 8 = 6875 KB. Die Kovarianzmatrix, die Matrix der errechneten Eigenvektoren und die Eigenwertematrix haben mit 88 Zeilen und 88 Spalten einen Bedarf von 3 x 88 x 88 x 8, somit von ca. 180 KB. Die resultierende normalisierte Matrix hat 10000 Zeilen und ca. 43 Spalten, somit 3359 KB.

insgesamter maximaler RAM-Bedarf in dieser Konfiguration: 7743.8 KB

Der Speicherbedarf ist mit ca. 8 MB denkbar gering.

Beim Authentifizierungsvorgang müssen die gesamte Eigenmatrix (3359 KB) und das zu testende Bild (9.8 KB) im Speicher gehalten werden. Zusätzlich benötigt man noch Platz für eine weitere Matrix mit der Dimension des Ursprungsbildes, 100 x 100 x 8 = 78 KB. Der weitere Bedarf ist mit wenigen kleinen Matrizen vernachlässigbar.

insgesamter RAM-Bedarf eines Authentifizierungsvorgangs: 3446.8 KB

Die Distanzfunktion Euklid-Distanz benötigt neben den zu vergleichenden Templates nur sehr geringen zusätzlichen Speicher.

insgesamter RAM-Bedarf der Distanzfunktion: 2 x 43 x 8 = 688 Bytes.

Eigenfaces	Trainingsphase	Authenticate 1	Distance Function 3
Bedarf	7743.8 KB	3446.8 KB	0.17 KB

Table 4.15: Eigenfaces Memory Requirements

Storage Requirements

Für den späteren Gebrauch müssen folgende Daten der Trainingsphase gespeichert werden. Die Eigenmatrix hat bei 88 Trainingsbildern der Dimension 100 x 100 10000 (Anzahl Pixel) Zeilen und ca. 43 (Anzahl Eigenvektoren) Spalten bei, somit 10000 x 43 x 8 (double) Bytes = 3359 KB. Das Durchschnittsbild hat die Dimension 100 x 100, damit 100 x 100 Bytes = 9.8 KB.
gesamter Bedarf: 3359 + 9.8 = 3368.8 KB

Die 3 Templates des Enrollments belegen 3 x 43 x 8 Bytes = 1 KB

Eigenfaces	Trainingsdaten	3 Templates
Bedarf	3368.8 KB	1 KB

Table 4.16: Eigenfaces Storage Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Trainingsphase (ohne Vorverarbeitung), das Authentifizieren eines Gesichtsbildes (ohne Vorverarbeitung) und das dreimalige Berechnen der Distanz dargelegt.

Eigenfaces	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Trainingsphase	2513310	2565620000	916.29 ms	2565.61 ms	64140.3 ms
Authenticate 1	9330	9525150	3.40 ms	9.52 ms	238 ms
Distance Function 3	5058	5163650	1.84 ms	5.15 ms	128.8 ms

Table 4.17: Eigenfaces Computing Time

Error Rate

Eigenfaces	FRR	FAR	Schwellwert
without	20.6	21.2	2000
HE	14.2	15.4	3750
MSR	16.9	18.6	3700
Daub4	18.8	19.3	3600
TT	33.6	35.9	1080

Table 4.18: Eigenfaces Error Rate

4.9 Elastic Bunch Graph Matching

Memory Requirements

Im nachfolgenden ist der RAM-Bedarf des EBGM bei Graustufen-Input-Bildern der Dimension 150 x 200 dargelegt.

Die 80 Gabor-Masken müssen zu jeder Zeit verfügbar sein. Dabei handelt es sich um quadratische Matrizen (double Werte 8 bytes) mit einer Größe von 19, 29, 39, 53 bzw. 77 (je 16). Diese belegen ca. 1432 KB.

In der Trainigsphase wird der Bunchgraph erzeugt. Er enthält 88 Facegraphen mit je 42 Knoten.

Für jeden Knoten wird seine Position gespeichert (2 ints). An jeder Knotenposition werden 80 Jets berechnet (für jede Maske ein Jet). Ein Jet belegt mit 2 Gleitkommazahlen 2 x 8 Bytes. Insgesamt belegt der Bunchgraph also 88 x 42 x 2 (Knotenpositionen) + 88 x 42 x 80 x 2 x 8 Bytes (Jets) = 4627 KB. Zur Extraktion der Jets werden die Trainingsbilder benötigt. Daher zusätzlicher RAM-Bedarf von 88 x 150 x 200 = 2578 KB.

Insgesamter RAM-Bedarf der Trainingsphase: 7205 KB.

Bei der Authentifizierung müssen die Informationen im Bunchgraphen zugreifbar sein, daher 4627 KB. Es wird ein möglichst gut positionierter Facegraph gesucht und dessen Jets werden berechnet, 42 x 80 x 2 x 8 = 52.5 KB (42 Knoten, 80 Masken, 2 x 8 Bytes pro Jet). Zusätzlich müssen zum Vergleich die Enrollment-Facegraphen im Speicher sein, 3 x 42 x 80 x 2 x 8 = 157.5 KB.

Insgesamter RAM-Bedarf des Authentifizierungsvorgangs: 664 KB (Masken) + 4627 KB (Bunchgraph) + 210 KB (4 Facegraphen) = 5501 KB.

Die Distanzfunktionen benötigen neben den Facegraphen, deren Ähnlichkeit berechnet werden sollen, nur sehr wenig zusätzlichen Speicher, dieser ist vernachlässigbar. Vier zu vergleichende Graphen belegen 210 KB. Dieser Bedarf kan noch verringert werden, wenn nur die zwei gerade betrachteten Graphen im Speicher sind, er beträgt dann 105 KB.

Insgesamter RAM-Bedarf der Distanzfunktionen: ca. 210 KB

EBGM	Trainingsphase	Authenticate 1	Distance Function 3
Bedarf	7205 KB	5501 KB	210 KB

Table 4.19: EBGM Memory Requirements

Storage Requirements

Für den späteren Authentifzierungsvorgang muss der Bunchgraph gespeichert werden, somit 4627 KB. Die Masken wurden in der Implementierung beim Start berechnet und mussten somit nicht gespeichert werden.

3 Templates (Facegraphen) belegen 3 x 42 x 80 x 2 x 8 Bytes = 157.5 KB.

EBGM	Trainingsdaten	3 Templates
Bedarf	4627 KB	157.5 KB

Table 4.20: EBGM Storage Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für die Trainingsphase (ohne Vorverarbeitung), das Authentifizieren eines Gesichtsbildes (ohne Vorverarbeitung) und das dreimalige Berechnen der Distanz dargelegt. Die Zeit für das Authentifizieren und die Distanzfunktion ist jeweils für Predicitive Step (ps), Predictive Iter (pi) und Narrowing Local Search (nl) angegeben.

EBGM	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Trainingsphase	19526500	19932900000	7118.89 ms	19932.9 ms	498322.3 ms
Authenticate 1 ps	1238450	1264230000	451.51 ms	1264.23 ms	31605.7 ms
Authenticate 1 pi	1235430	1261140000	450.41 ms	1261.15 ms	31528.7 ms
Authenticate 1 nl	1284920	1311670000	468.45 ms	1311.67 ms	32791.5 ms
Distance Function 3 ps	9993	10201700	3.64 ms	10.19 ms	254.8 ms
Distance Function 3 pi	10802	11027200	3.94 ms	11.03 ms	275.8 ms
Distance Function 3 nl	33288	33981300	12.14 ms	33.99 ms	849.8 ms

Table 4.21: EBGM Computing Time

Error Rate

Predictive Step	FRR	FAR	Treshold
without	13.6	15.4	0.615
HE	11	13.4	0.585
MSR	12.4	13.2	0.535
Daub4	14.9	15.8	0.553
TT	12.8	14	0.471

Predictive Iter	FRR	FAR	Treshold
without	12.3	13.9	0.66
HE	9.4	10.4	0.635
MSR	10.2	10.3	0.594
Daub4	11.7	13.6	0.605
TT	11.2	12.2	0.537

Narrowing Local Search	FRR	FAR	Treshold
without	9.7	10.2	0.71
HE	7.4	8.3	0.68
MSR	7.8	9.0	0.65
Daub4	8.9	9.5	0.663
TT	7.9	8.8	0.627

Table 4.22: EBGM Error Rate

4.10 Enhanced Local Gabor Binary Patterns Histogram Sequence

Memory Requirements

Im nachfolgenden ist der RAM-Bedarf des ELGBPHS Algorithmus bei Graustufen-Input-Bildern der Dimension 100 x 100 dargelegt.

Beim ELGBPHS gibt es keine Trainingsphase.

Beim Authentifizierungsvorgang müssen sich die 3 Enrollment-Templates und das zu vergleichende Template im Speicher befinden. Ein Template besteht aus 4000 Histogrammen (80 Masken mal 50 Bildbereiche mit Gewichtung > 0). Weil ein Bildbereich die Dimension 10 x 10 hat, kann die maximale Anzahl eines Grauwertes 100 sein, damit passt es in 1 Byte. Ein Histogramm belegt somit 256 Bytes (ohne Komprimierung). Für ein Template sind somit 80 x 50 x 256 Bytes = 1000 KB vonnöten. Die 3 Enrollment-Templates werden mit dem Test-Template verglichen, somit 4 x 1000 KB = 4000 KB. Die eigentliche Vergleichsfunktion kommt mit sehr wenig Speicher aus, dabei wird im Wesentlichen bloß eine Summe erhöht. Auch die Gewichtung lässt sich in wenigen Bytes speichern, das ist alles vernachlässigbar.

gesamter RAM-Bedarf des Authentifizierungsvorgangs: 4000 KB

Die Distanzfunktion benötigt neben den Histogrammen nur sehr wenig Speicher. Damit ist der RAM-Bedarf der Vergleichsfunktion 4 Templates, 4000 KB.

gesamter RAM-Bedarf der Vergleichsfunktion: 4000 KB

ELGBPHS	Trainingsphase	Authenticate 1	Distance Function 3
Bedarf	-	4000 KB	4000 KB

Table 4.23: ELGBPHS Memory Requirements

Storage Requirements

Beim ELGBPHS gibt es keine Trainigsphase und damit keine Daten zu speichern.

3 Templates belegen 3000 KB.

ELGBPHS	Trainingsdaten	3 Templates
Bedarf	0 KB	3000 KB

Table 4.24: ELGBPHS Storage Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für das Authentifizieren eines Gesichtsbildes (ohne Vorverarbeitung) und das dreimalige Berechnen der Distanz dargelegt. Beim ELGBPHS gibt es keine Trainingsphase.

ELGPBHS	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Trainingsphase	-	-	-	-	-
Authenticate 1	4256300	4344890000	1551.75 ms	4344.9 ms	108622.5 ms
Distance Function 3	31367	32020300	11.44 ms	32.03 ms	800.8 ms

Table 4.25: ELGBPHS Computing Time

Error Rate

ELGBPHS	FRR	FAR	Schwellwert
without	4.6	5	760000
HE	4.6	4.9	755000
MSR	5.4	5.9	744000
Daub4	5.8	6.1	741000
TT	3.2	3.5	754000

Table 4.26: ELGBPHS Error Rate

4.11 MarliesFaces

Memory Requirements

Im nachfolgenden ist der RAM-Bedarf der MarliesFaces bei Graustufen-Input-Bildern der Dimension 100 x 100 dargelegt.

Beim Training müssen die 88 Trainingsbilder (88 x 100 x 100 Bytes = 859 KB) geladen werden. Nach der Umwandlung in Histogramme eines Bereiches (88 x 80 x 256 Bytes = 1760 KB) kann der Speicher der Trainingsbilder wieder freigegeben werden. Für einen Bereich werden die Histogramme als Vektoren als PCA-Bilder angeordnet (7040 KB Bedarf, float 4 bytes) und daraus ein Eigenspace berechnet. Das wird für alle 50 Bereiche wiederholt. Eine normalisierte Eigenmatrix hat 20480 Zeilen und 53 Spalten, somit 4240 KB (float 4 bytes).

insgesamter max. RAM-Bedarf der Trainingsphase: 11280 KB

Beim Authentifizierungsvorgang sollten aus Performance-Gründen alle Eigenmatrizen im Speicher gehalten werden, somit 50 x 20480 x 53 x 4 = 207 MB. Das Bild wird gefaltet und in Histogramme umgewandelt, das erfordert 1000 KB. Die Projektion belegt 50 x 53 x 4 (50 Bereiche, 53 Eigenvektoren, float 4 bytes) = 10 KB.

gesamter max. RAM-Bedarf des Authentifizierungsvorgangs: 208 MB

Die Distanzfunktion benötigt neben den Histogrammen nur sehr wenig Speicher. Der Bedarf für den Vergleich von 2 Templates ist (nahezu) beliebig gering, weil die Templates streng sequentiell verarbeitet werden können. Typisch: 2 Templates = 2 x 50 x 53 x 4 = 20.6 KB

gesamter RAM-Bedarf der Vergleichsfunktion: 20.6 KB

MarliesFaces	Trainingsphase	Authenticate 1	Distance Function 3
Bedarf	11280 KB	208 MB	20.6 KB

Table 4.27: MarliesFaces Memory Requirements

Storage Requirements

Die Eigematrizen und zugehörigen Durschnittsbilder müssen (wenn das Training nicht jedesmal wiederholt werden soll) gespeichert werden. Die Eigenmatrizen belegen 207 MB. Die Durchschittsbilder 50 x 20480 x 4 Bytes = 4000 KB.

3 Templates belegen 50 x 53 x 4 Bytes = 10.3 KB

MarliesFaces	Trainingsdaten	3 Templates
Bedarf	211 MB	10.3 KB

Table 4.28: MarliesFaces Storage Requirements

Computing Time

Hier ist der Aufwand (CPU-Zyklen) für das Authentifizieren eines Gesichtsbildes (ohne Vorverarbeitung) und das dreimalige Berechnen der Distanz dargelegt.

MarliesFaces	Ticks	CPU-Cycles	2.8 GHz	1 GHz	40 MHz
Trainingsphase	-	-	146s	408s	10220s
Authenticate 1	1284140	1310860000	468.16ms	1310.85ms	32771.2ms
Distance Function 3	4844	4945240	1.77ms	4.96ms	123.9ms

Table 4.29: MarliesFaces Computing Time

Error Rate

MarliesFaces	FRR	FAR	Schwellwert
without	10.2	11.3	495000000
HE	8.2	9.8	550000000
MSR	8.6	9.9	540000000
Daub4	9.7	11.0	530000000
TT	10.1	13.7	360000000

Table 4.30: MarliesFaces Error Rate

4.12 Ergebnisse zusammengefasst

Memory Requirements

Verfahren	Trainingsphase	Authenticate 1	Distance Function 3
Eigenfaces	7743.8 KB	3446.8 KB	0.17 KB
EBGM	7205 KB	5501 KB	210 KB
ELGBHPHS	-	4000 KB	4000 KB
MarliesFaces	11280 KB	208 MB	20.6 KB

Table 4.31: Memory Requirements zusammengefasst

Der Speicherbedarf von Eigenfaces, EBGM und ELGBPHS ist durchwegs gering, d.h. kein Problem für aktuelle PCs oder Smartphones. Der Authentifizerungsvorgang der MarliesFaces

benötigt über 200 MB an RAM, falls alle Eigenmatrizen zur Projektion der Histogramme gleichzeitig im Speicher gehalten werden. Werden die Eigenmatrizen jedoch einzeln eingelesen, die Histogramme projiziert und der Speicher der Eigenmatrix wieder freigegeben, beträgt der RAM-Bedarf etwa 4 MB.

Storage Requirements

Verfahren	Trainingsdaten	3 Templates
Eigenfaces	3368.8 KB	1 KB
EBGM	4627 KB	157.5 KB
ELGBPHS	0 KB	3000 KB
MarliesFaces	211 MB	10.3 KB

Table 4.32: Storage Requirements zusammengefasst

Auffällig beim Festspeicherbedarf ist, dass die MarliesFaces etwa 211 MB an Trainingsdaten produzieren. Das liegt daran, dass 50 Eigenmatrizen gespeichert werden müssen, in die Histogramme projiziert werden.

Computing Time

Verfahren, 2.8 GHz	Trainingsphase	Athenticate 1	Distance Function 3
Eigenfaces	916.29 ms	3.40 ms	1.84 ms
EBGM nl	7118.89 ms	468.45 ms	12.14 ms
ELGBPHS	-	1551.75 ms	11.44 ms
MarliesFaces	146 s	468.16ms	1.77ms

Table 4.33: Computing Time zusammengefasst

Die Trainingsphase der MarliesFaces dauert bei 2.8 Ghz Taktfrequenz etwa 146 Sekunden, weil 50 Datenmatrizen erzeugt werden und deren Eigenvektoren berechnet werden müssen.

Best Error Rate

Verfahren	Vorverarbeitung	FRR	FAR
Eigenfacs	HE	14.2	15.4
EBGM nl	HE	7.4	8.3
ELGBPHS	TT	3.2	3.5
MarliesFaces	HE	8.2	9.8

Table 4.34: Error Rate zusammengefasst

Mit knapp 97% erreicht ELGBPHS mit der Vorverarbeitung TT die höchste Erkennungsrate. Eigenfaces ist mit etwa 85% weit abgeschlagen. MarliesFaces erzielen 91% und EBGM 90% Genauigkeit (jeweils bei HE Vorverarbeitung).

CHAPTER 5

Zusammenfassung

Im Rahmen der Diplomarbeit wurde festgestellt, dass im Vergleich zum Fingerabdruck, der Venenerkennung oder Iris das Gesicht als biometrisches Merkmal von einer höheren Fehlerrate gekennzeichnet. ELGBPHS in Kombination mit der Vorverarbeitung TT erreichte mit ca. 96,7% die höchste Erkennungsrate der behandelten Verfahren . Weit abgeschlagen liegen die Eigenfaces mit Vorverarbeitung HE bei 85%. EBGM erreichte mit Displacement Estimator "Narrowing Local Search" und Vorverarbeitung HE 92,5%. Außer Konkurrenz sind die MarliesFaces, weil diese für einen bestimmte Umgebung (Smart-Card) entwickelt wurden und mit 90,5% das Ziel von 10% Fehlerrate erreichten.

Der Grund für die niedrigere Erkennungsrate im Vergleich zu anderen biometrischen Merkmalen sind die Probleme, die man bei der Identifizierung einer Person über das Gesicht hat. Helligkeitsunterschiede, Make-Up, Änderungen durch das Alter, Kopfposition, Mimik, Brillen usw. erschweren die Arbeit der Gesichtserkennungssysteme und sind so bei anderen Merkmalen wie dem Fingerabdruck nicht zu finden.

Zum Aufwand bzw. Speicherbedarf ist zu sagen, dass Eigenfaces, EBGM und ELGBPHS auf einem PC oder aktuellem Smartphone uneingeschränkt lauffähig sind.

Das einzige Verfahren, das auch in einer extrem ressourcen-beschränkten Umgebung wie einer Smart-Card eine Fehlerrate von unter 10% erreicht, sind die MarliesFaces. In dieser Umgebung sind daneben nur die Eigenfaces lauffähig, jedoch mit einer inakzeptablen Erkennungsrate. Die Trainingsphase der MarliesFaces benötigt bei 2,8 GHz 146 Sekunden, der Authentifizierungsvorgang hat einen RAM-Bedarf von 208 MB und die Trainingsdaten belegen 211 MB. Die Trainingsphase, Vorverarbeitung und Projektion können bei Anwendung auf einer Smart-Card außerhalb erfolgen, weil keine personenbezogenen Daten außerhalb gespeichert werden müssen (beispielsweise ist die zu authentifizierende Person im Trainingset nicht enthalten). Nur die Enrollment-Projektionen werden auf dem Chip gespeichert und mit der von außen gelieferten Test-Projektion verglichen.

Histogram Equalization verbesserte die Fehlerrate der Eigenfaces um ca. 6%. Beim EBGM lag der Effekt bei 2,5%. Die Vorverarbeitung, die beim ELGBHS am besten abgeschnitten hat, ist TT mit 1,5% Verbesserung gegenüber den Testläufen ohne Vorverarbeitung (nur Face Detection und Elliptisches Zuschneiden). Die anderen Methoden verursachten keine Verbesserung bzw. verschlechteten das Ergebnis sogar.

Es muss für jede Anwendung separat entschieden werden, ob die geringere Erkennungsrate der Gesichtserkennungsverfahren (im Vergleich zu beispielsweise dem Fingerabdruck oder der Venenerkennung) einen Einsatz in der Praxis zulässt. Zukünftige Arbeiten könnten sich mit der Verbesserung der Erkennungsrate der MarliesFaces beschäftigen, z.B. durch Analyse weiterer nicht behandelter Vorverarbeitungsmethoden. Die Diplomarbeit beschränkte sich auf die Beschreibung und Evaluierung von 5 ausgewählten Verfahren. Darauf aufbauend wäre eine Beschäftigung mit weiteren Gesichtserkennungsverfahren / Kombinationen von Methoden und deren Evaluierung wichtig für den Praxis-Einsatz von Gesichtserkennungssystemen.

APPENDIX A

Appendix

A.1 500 statische Durchgänge

Figure A.1: 500 statische Durchgänge, FERET Gesichtsdatenbank [42]

A.2 Setzen des Schwellwerts

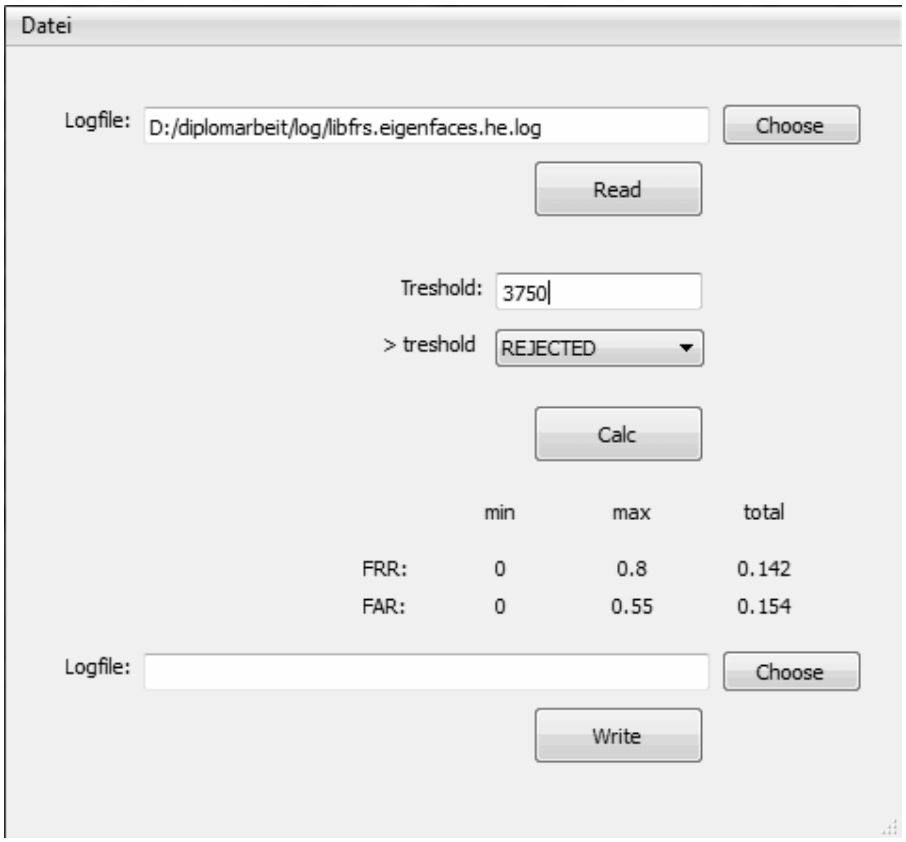

Figure A.2: Setzen des Schwellwertes

A.3 Logfile

```
frsInit(): new internal log object for lib created
frsInit(): create new internal frs object for lib
Computing Gabor-masks ...
Reading conf from ..\..\frs_data\conf\ELGBPHS_hist_properties.txt ...
*************************
** ELGBPHS
Reading training from directory ...
frsLoad() called
ELGBPHS::load(): There is no training-phase with ELGBPHS, so nothing to load here!
PreProcessing init ...
Loading training completed.
RUN 0
Reading ..\..\..\images\STATIC500\STATIC_000\enrollment
Reading ..\..\..\images\STATIC500\STATIC_000\accept
Reading ..\..\..\images\STATIC500\STATIC_000\reject
***************
** START ENROLL
Enrolling ..\..\..\images\STATIC500\STATIC_000\enrollment\00774_941201_fa.bmp ...
PreProcess image ...
Calculating hists of ...
Enrolling ..\..\..\images\STATIC500\STATIC_000\enrollment\00774_941201_fb.bmp ...
PreProcess image ...
Calculating hists of ...
Enrolling ..\..\..\images\STATIC500\STATIC_000\enrollment\00774_960620_fa.bmp ...
PreProcess image ...
Calculating hists of ...
** END ENROLL
***************
*****************
** ACCEPT TEST **
Verifying ..\..\..\images\STATIC500\STATIC_000\accept\00774_941205_fa.bmp ...
PreProcess image ...
Calculating hists ...
Converting enrollment templates to histograms ...
distance 0: 783963
distance 1: 866911
distance 2: 782995
max: 866911
ACCEPTED
```

Figure A.3: Log-Datei

Literaturverzeichnis

[1] Verlagsgruppe Bertelsmann GmbH. *Die Grosse Bertelsmann Lexikothek, Band 2*. Verlagsgruppe Bertelsmann GmbH, 1993.

[2] A. K Jain, R. Bolle, and S. Pankanti. *Biometrics: Personal Identification in Networked Society*. Kluwer Academic Publications, 1999.

[3] S. Furnell. *Computer Insecurity*. Springer, 2005.

[4] J. Bonneau and S. Preibusch. The password thicket: technical and market failures in human authentication on the web. In *Proceedings of The Ninth Workshop on the Economics of InformationSecurity - WEIS2010*, 2010.

[5] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399–458, 2003.

[6] J. J. Borking. *At Face Value: On Biometrical Identification and Privacy*. Registratiekamer (Niederlande), 1999.

[7] P. N. Belhumeur. Ongoing challenges in face recognition. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2005 Symposium*, pages 5–14, 2005.

[8] R. Gross, S. Baker, I. Matthews, and T. Kanade. *Handbook of Face Recognition*, chapter Face Recognition Across Pose and Illumination, pages 193–216. Springer-Verlag, 2005.

[9] Y. Moses, Y. Adini, and S. Ullman. Face recognition: The problem of compensating for changes in illumination direction. In *European Conference on Computer Vision*, pages 286–296, 1994.

[10] L. Xueyan and G. Shuxu. The fourth biometric - vein recognition. In *Pattern Recognition Techniques, Technology and Applications*, pages 537–546, 2008.

[11] W. Zhang, S. Shan, L. Qing, X. Chen, and W. Gao. Are gabor phases really useless for face recognition? *Pattern Analysis & Applications*, 12(3):301–307, 2009.

[12] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[13] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[14] M. L. Teixeira. The bayesian intrapersonal/extrapersonal classifier. Master's thesis, Colorado State University, 2003.

[15] M.S. Bartlett, J.R. Movellan, and T.J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450 – 1464, 2002.

[16] Z. Lihong, Z. Cheng, Z. Xili, S. Ying, and Z. Yushi. Face recognition based on image transformation. In *WRI Global Congress on Intelligent Systems*, pages 418–421, 2009.

[17] Ziad M. Hafed and Martin D. Levine. Face recognition using the discrete cosine transform. *International Journal of Computer Vision*, 43(3):167–188, 2001.

[18] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.

[19] P. Sinha, B. Balas, Y. Ostrovsky, and R.; Russell. Face recognition by humans: Nineteen results all computer vision researchers should know about. In *Proceedings of the IEEE*, volume 94, pages 1948–1962, 2006.

[20] A. J. O'Toole, J. P. Phillips, F. Jiang, J. Ayyad, N. Penard, and H. Abdi. Face recognition algorithms surpass humans matching faces over changes in illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1642–1646, 2007.

[21] R. Zhou, J. Wu, Q. He, C. Hu, and Z. Yu. Approach of human face recognition based on sift feature extraction and 3d rotation model. In *Information and Automation (ICIA '11)*, pages 476–479, 2011.

[22] L. Du, Z. Jia, and L. Xue. Human face recognition based on principal component analysis and particle swarm optimization-bp neural network. In *Natural Computation (ICNC'07)*, pages 287–291, 2007.

[23] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[24] Bradski G. and A. Kaehler. *Learning OpenCV*. O'Reilly Media, 2008.

[25] D. J. Jobson, Z. Rahman, and G. A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965 – 976, 1997.

[26] N. Kela, A. Rattani, and P. Gupta. Illumination invariant elastic bunch graph matching for efficient face recognition. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '06)*, page 42. IEEE Computer Society, 2006.

[27] Z. Rahman, D. J. Jobson, and G. A. Woodell. Retinex processing for automatic image enhancement. *Journal of Electronic Imaging*, 13(1):100–110, 2004.

[28] S. Du and R. Ward. Wavelet-based illumination normalization for face recognition. In *Proceedings of the 2005 IEEE International Conference on Image Processing (ICIP'05)*, pages 954–957. IEEE Computer Society, 2005.

[29] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635 – 1650, 2010.

[30] L. Wiskott, J. Fellous, N. Krüger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

[31] D. Bolme. Elastic bunch graph matching. Master's thesis, Colorado State University, 2003.

[32] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang. Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)*, volume 1, pages 786–791, Washington, DC, USA, 2005. IEEE Computer Society.

Linkverzeichnis

[33] S. Granger. The simplest security: A guide to better password practices. http://www.symantec.com/connect/de/articles/simplest- security-guide-better-password-practices. 17.01.2002. Zugriff: 14.11.2011.

[34] heise Nachrichten. Angriff auf playstation network: Persönliche daten von millionen kunden gestohlen. http://www.heise.de/newsticker/meldung/Angriff-auf-Playstation-Network-Persoenliche-Daten-von-Millionen-Kunden-gestohlen-1233136.html. Zugriff: 14.11.2011.

[35] heise Nachrichten. Hacker dringen in steam-plattform von valve software ein. http://www.heise.de/newsticker/meldung/Hacker-dringen-in-Steam-Plattform-von-Valve-Software-ein-1377214.html. Zugriff: 14.11.2011.

[36] Bundesamt für Sicherheit in der Informationstechnik. Einführung in die technischen grundlagen der biometrischen authentisierung. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Biometrie/Technische_Grundlagen_pdf.pdf?__blob=publicationFile, 2011. Zugriff: 14.11.2011.

[37] GIT SICHERHEIT. Emea-markt für biometrie bietet signifikante wachstumschancen. http://www.git-sicherheit.de/news/emea-markt-fuer-biometrie-bietet-signifikante-wachstumschancen, 2010. Zugriff: 15.11.2011.

[38] BITKOM Bundesverband Informationswirtschaft Telekommunikation und neue Medien e.V. Biometrie aus deutschland weltweit führend. http://www.bitkom.org/de/presse/62013_60912.aspx, 2009. Zugriff: 15.11.2011.

[39] BITKOM Bundesverband Informationswirtschaft Telekommunikation und neue Medien e.V. Biometrie referenzbroschüre 2009 - 2. auflage. http://www.bitkom.org/de/themen/55159_60402.aspx, 2009. Zugriff: 14.11.2011.

[40] BITKOM Bundesverband Informationswirtschaft Telekommunikation und neue Medien e.V. Berufstätige wünschen sich flexiblere arbeitsbedingungen. http://www.bitkom.org/de/markt_statistik/64054_63496.aspx, 2010. Zugriff: 14.11.2011.

[41] Opencv library. http://opencv.willowgarage.com/wiki/. Zugriff: 03.11.2011.

[42] Feret face database. http://www.itl.nist.gov/iad/humanid/feret/feret_master.html. Zugriff: 03.11.2011.

[43] Colorado state university: Evaluation of face recognition algorithms. http://www.cs.colostate.edu/evalfacerec/index10.php. Zugriff: 03.11.2011.