Automatische Gesichtserkennung an der Haustür

Robert Malz
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Email: robert.malz@smail.inf.h-brs.de

Idris Nawid
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Email: idris.nawid@smail.inf.h-brs.de

***Zusammenfassung*—Die automatische Gesichtserkennung bietet eine Vielzahl von Anwendungsmöglichkeiten, aber vor allem wird in dieser Seminararbeit betrachtet, wie eine Gesichtserkennung speziell an der Haustür realisiert werden kann. Hierfür werden sowohl theoretische Grundlagen für die Erkennung als auch für die Identifikation von Personen vorgestellt. Anschließend wird eine praktische Umsetzung mittels OpenCV vorgestellt, welche Hürden es zu bewältigen gilt, inklusiv der Schwierigkeiten ihrer Realisierung**

I. Einleitung

Die automatische Gesichtserkennung ist bei den Sicherheitsbehörden für die Ergreifung von gesuchten Personen oder die Identifizierung bei der Einreise in die Vereinigten Staaten von Amerika nicht mehr wegzudenken. Durch die Tatsache, dass im öffentlichen Raum diese Technik schon eingesetzt wird, stellt sich die Frage nach der automatischen Gesichtserkennung von Personen an der Haustür.

Ziel dieser Arbeit ist es mittels der freien Programmbibliothek OpenCV, die Algorithmen für die Bildverarbeitung beinhaltet, ein System aufzubauen, welches die Gesichtserkennung an der Haustür automatisiert durchführt. Durch die Doppeldeutigkeit des Begriffs wird in dieser Arbeit die Lokalisierung eines Gesichts in einem Bild (engl. face detection) als Gesichtslokalisierung bezeichnet. Bei der Wiedererkennung eines bekannten Gesichtes (engl. face recognition) wird der Begriff Gesichtsidentifikation verwendet.

Kapitel 2 stellt einen allgemeinen Überblick über den generischen Ablauf, der Gesichtserkennung dar. Nachfolgend wird darauf eingegangen, wie die Lokalisierung eines Gesichts in einem Bild (Gesichtslokalisierung) abläuft, welche Verfahren es gibt und wie das verwendete Verfahren in OpenCV funktioniert. Nachfolgend werden der merkmalsbasierte Ansatz und der holistische Ansatz vorgestellt, welche die zwei Hauptgruppen der gegenwärtigen Forschungsansätze sind. Im Anschluss werden zwei bekannte Verfahren zur automatischen Gesichtsidentifikation. Zuerst wird die Eigenface Methode und als zweites die Fisherface Methode dargestellt.

In Kapitel 3 wird die Implementierung mittels OpenCV (Open Source Computer Vision Library) vorgestellt. Begonnen mit OpenCV. Im Anschluss werden die Installation und Konfiguration Bibliothek beschrieben. Darauf folgend wird die Wahl des Algorithmus für die Identifizierung der Person erläutert. Zuletzt wird die Implementierung der Gesichtserkennung mit OpenCV erläutert.

II. Gesichtserkennung

In diesem Kapitel wird ein allgemeiner Überblick über den typischen Prozess der Gesichtserkennung gegeben. Auf den generischen Prozess aufbauend, wird auf die Gesichtslokalisierung, also die Erkennung eines Gesichtes und typische Verfahren die hierfür existieren eingegangen, speziell wird hier der Fokus auf die Viola-Jones Methode [1] gelegt. Für die Gesichtsidentifikation, also die Wiedererkennung eines bekannten Gesichtes, wird ein Überblick über die existierenden Methoden gegeben. Darauf aufbauen werden die Eigenface und Fisherface Methode genauer betrachtet. Hierfür wird deren Funktionsweise, um ein Gesichts zu Identifizieren, vorgestellt.

A. Allgemeiner Überblick

Für die automatische Gesichtserkennung wird einer Leistungsfähiger Rechner und Kamera benötigt. Um die Erkennung von Personen an der Haustür zu bewerkstelligen, wird die Umgebung permanent mit der Kamera beobachtet. Den generischen Ablauf, ist in Abbildung 1 dargestellt.

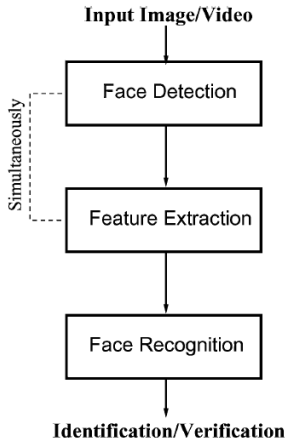

Abbildung 1. Generischer Ablauf eines Gesichtserkennungssystems aus [2]

Um Personen zu erkennen und zu identifizieren müssen zuerst biometrische Daten in Bildern oder Videoaufnahmen gefunden werden (Face Detection). Nachdem ein Gesicht erkannt wurde, wird dieses extrahiert (Feature Extraction) und mit schon bekannten Bildern zur Identifikation verglichen (Face Recognition).

B. Gesichtslokalisierung

Bei der Gesichtslokalisierung geht es um die Erkennung von Gesichtern in Bildern oder Videoaufnahmen. Diese bildet

somit die Basis zur Identifizierung einer Person. Exemplarisch werden die Gesichtslokalisierung durch das Entfernung von statischen Hintergrund, Bewegungsinformationen und die Viola-Jones Methode vorgestellt.

Weitere Methoden zur Gesichtslokalisierung können in [3] gefunden werden.

1) Gesichtslokalisierung durch Entfernung der Hintergrundinformation: Ein Gesichts kann auf einfache Art lokalisiert werden. Wenn der Hintergrund des Bildes bekannt ist, so lässt sich die Hintergrundinformation entfernen und das Gesicht lokalisieren. Hierfür muss allerdings der Hintergrund statisch sein und es darf sich nur ein Gesicht im Bild befinden [3]. Dieses Verfahren ist für eine Erkennung an der Haustür nicht praktisch, da sich ggf. auch Bäume im Hintergrund bewegen oder sich andere Gegenstände verändern. Zusätzlich lässt sich mit diesem Verfahren nur eine Person erkennen.

2) Gesichtslokalisierung durch Bewegungsinformationen: Die Gesichtslokalisierung mittels Bewegungsinformationen bildet einen weitere Methode, die sich jedoch nur bei Videoaufnahmen anwenden lässt. Zuerst wird die Differenz von Bild zu Bild berechnet und ist hierbei die Differenz größer als ein festgelegter Wert, wird von einem neuen Bild ausgegangen. Die sich veränderten Pixel werden als schwarz dargestellt und Hintergrundrauschen wird entfernt. Diese Methode produziert allerdings zahlreiche Fehler, da auch andere bewegte Objekte als Gesichter erkannt werden [4].

3) Viola-Jones Methode: Eine weitere Art der Gesichtslokalisierung ist die Viola-Jones Methode. Bei dieser Methode werden Fenster fixer Größe über das Bild geschoben und entschieden, ob sich darunter ein Gesicht befindet. Viola und Jones kombinieren in ihrem Objekterkennungsalgorithmus drei unterschiedliche Konzepte miteinander. Es setzt sich aus aus dem *integral image*, einer modifizierten *AdaBoost Prozedur* und einer *cascade classifier* zusammen, die im folgenden genauer erläutert werden.

Beim ersten Schritt, dem *integral image*, wird von einem Grauwertbild ausgegangen, d.h. das Eingabebild wird zuerst in ein Graustufenbild konvertiert. Der Grauwert wird durch die folgende Formel errechnet.

$$Grauwert = 0,299 * Rot + 0,587 * Grün + 0,144 * Blau \quad (1)$$

Um die Berechnung zu beschleunigen nutzen Viola und Jones das Konzept des Integralbildes, welches vorher im Bereich des texture mapping von Crow genutzt wurde [5]. Der Wert des Pixels (x,y) im Integralbild ist die Summe aller Pixel innerhalb eines Rechtecks, welches zwischen dem aktuellen Pixel (x,y) und dem Ursprung (0,0) aufgespannt wird.

$$I_{\sum}(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j) \quad (2)$$

Abbildung 2 verdeutlicht die Berechnung eines Integralwertes aus den Werten der Pixel, die zwischen Ursprung und dem markierten Pixel (x,y) liegen. Bei der Berechnung des Integralwertes reduziert sich die Rechenzeit, da die Berechnung der Pixelsummen auf Elementaroperationen basiert und so der Rechner weniger Zeit und Speicher benötigt. Eine wichtige Erkenntnis, die aus dem Wert eines Integralbildes gezogen wird ist, dass je heller der Bereich ist, desto höher ist die

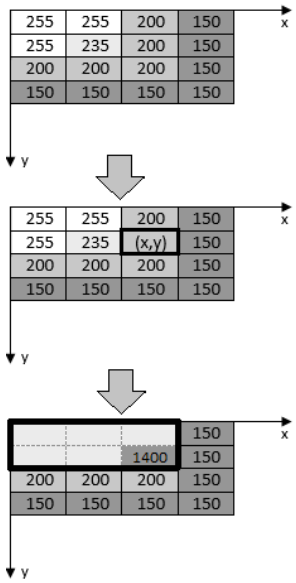

Abbildung 2. Berechnung des Integralbildes, anhand eines Grauwertbildes

Summe. Wenn Gesichter genauer betrachtet werden, dann ist diese Information eine wichtige Grundlage, da dort typische Helligkeitsunterschiede erkannt werden. So sind die Augenbrauen in der Regel dunkler als die Augen, bzw. die Nase heller als die Augen (siehe Abb.3). Aus diesen Erkenntnissen

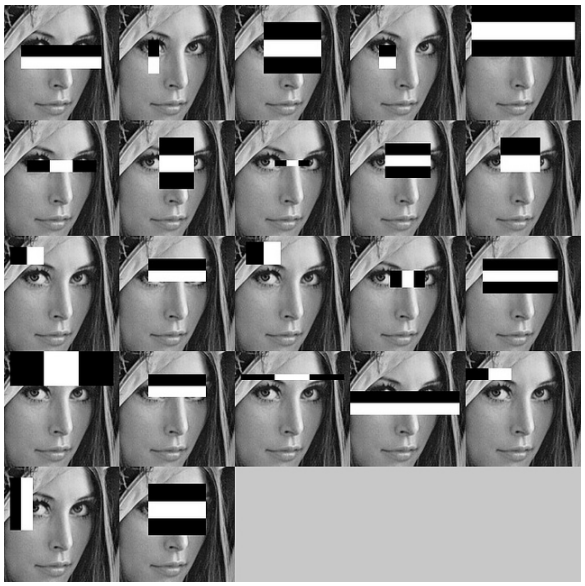

Abbildung 3. Haarmerkmale eines Gesichts aus [6]

für die Detektion von Gesichtern, welche in Abb. 4 zu sehen sind, stammen die Haar-like features.

Um den Wert eines Haar-like features zu berechnen werden die Summen der Pixel des jeweils schwarzen und weißen Bereichs gebildet und deren Differenz gebildet Daraus wird ein ein Vergleichswert errechnet.

Nachdem nun der Vergleichswert berechnet wurde, folgt der zweite Schritt der modifizierten *AdaBoost Prozedur*. Das AdaBoost wurde 1996 von Schapire und Freund vorgestellt

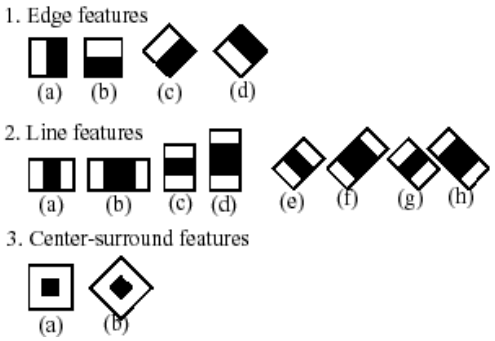

Abbildung 4. Haarmerkmale der Viola-Jones Methode aus [7]

[8] und wird verwendet, um die Leistung schwacher Klassifikatoren zu verstärken. Ein schwacher Klassifikator ist gekennzeichnet dadurch, dass der Algorithmus einen Fehler von weniger als 50% generiert [9]. Die AdaBoost Prozedur findet in der Viola-Jones Methode Anwendung um, darin die Klassifikatoren zu finden, deren Merkmale zu einer zuverlässigen Gesichtslokalisierung führen. Ein Klassifikator kann z.B. sein, dass die Augen dunkler als die Nasenwurzel oder die Augenbrauen dunkler als die Augen sind. Um die Klassifikatoren zu finden wird eine Trainingsmenge von positiven (bsp. Gesichtern) und negativen (bsp. Hintergründe) Bildern benötigt. Bei einem Unterfenster mit einer Auflösung von 24 x 24 Pixeln kann man über 180.000 Merkmale zuordnen, mit welchem das Bild nach Merkmalen durchsucht wird. So bekommt man mehr Merkmale als Pixel in dem Unterfenster, aber dies führt zu hohen Kosten. Die AdaBoost Prozedur wurde daraufhin von Viola und Jones optimiert, das man nur eine kleine Merkmalsmenge nimmt und diese kombiniert, um einen starken Klassifikator zu bilden.

Das letzte Konzept ist der *cascade classifier*, der die Detektionsleistung steigert und die Rechenzeit massiv verkürzt. Zwar kann man durch eine höhere Anzahl an Klassifizieren das gleiche erreichen, aber dies würde sich negativ auf die Rechenzeit auswirken. Stattdessen werden mehrere starke Klassifizierer trainiert und nacheinander auf das Unterfenster bzw. Bild angewendet. Die Entscheidung ob sich ein Gesicht in dem Unterfenster bzw. Bild befindet erfolgt mittels der Klassifikatoren sequenziell. Hierbei wird erst mit dem gröbsten Klassifikator begonnen und nur bei positiven Ergebnis mit den feineren Klassifikatoren fortgefahren. Bei einem negativen Ergebnis wird das Unterfenster verworfen und das nächste Fenster betrachtet. In der Arbeit von Viola und Jones wurden 38 Stufen trainiert, um ein Gesicht zu identifizieren.

Weitere Informationen zur Funktionsweise der Methode finden sich in [1].

Bei der Implementierung der Gesichtserkennungssoftware wird die Viola-Jones Methode zur Gesichtslokalisierung angewendet.

C. Methoden der Gesichtsidentifikation

Im vorherigen Kapitel wurde Methoden zur Lokalisierung von Gesichtern vorgestellt worden, nun kann man die lokalisierten Gesichter weiter verarbeiten.

Der nächste Schritt ist, dass unbekannte Gesicht zu Identifizieren. Bei der Identifikation lässt sich generell zwischen der merkmalsbasierten und holistischen Gesichtsidentifikation unterscheiden [10].

Die merkmalsbasierten Gesichtsidentifikation identifiziert und extrahiert Merkmale des Gesichts wie Augen, Nase und Mund. Anhand dieser Merkmale lässt sich die Person identifizieren, hierbei gibt es verschiedene Methoden. Eine der ersten Gesichtserkennungstechniken beruht auf den merkmalsbasierten Ansatz und wurde von Kanada im Jahr 1973 in seiner Doktorarbeit veröffentlicht [11]. In dem von Kanade entwickelten verfahren wurden 16 Parameter des Gesichts genommen und Abstände, Flächen und Winkel gemessen, worauf dann der Abgleich mit der Datenbank erfolgt. In der Literatur erwähnte und auch in der Praxis eingesetzte verfahren sind im merkmalsbasierten Verfahren die Hidden-Markov Methode (HMM) und Elastic Bunch Graph Matching (EBGM). Beim HMM wird jede Person mittels des HMM in der Gesichtserkennungsdatenbank gespeichert. Zuerst werden die Trainingsbilder in Blöcke geteilt und beinhalten markante Punkte wie Haare, Stirn, Augen, Nase und Mund. Diese Blöcke werden einer Kosinustransformation unterzogen und aus den Koeffizienten ein HMM gebildet. Sowohl die Gesichtsdatenbank geht wie genannt vor, um das HMM zu bilden, als auch das zu Vergleichende Bild, wobei hierbei ein Schwellwert eingefügt wird, um Fehler zu minimieren [12]. Das EBGM werden Graphen auf das Gesicht gelegt und markante Punkte werden gekennzeichnet, als Jets. Bei dem Verfahren werden die Längen der Linien zu den Jets gespeichert und später verglichen [13]. Eine detaillierte Beschreibung des Verfahrens findet sich in dem Artikel von Wiskott, et al. [13].

Bei der holistischen Gesichtsidentifikation werden hingegen das Gesicht als ganzes analysiert und klassifiziert. Zu diesem Verfahren gehört unter anderem die Eigenfaces Methode, Fisherfaces Methode und die Local Binary Pattern. In den folgenden Unterkapiteln, werden die Eigenfaces und Fisherfaces Methode genauer beschrieben. Hierbei erfolgt im Gegensatz zu der merkmalsbasierten Gesichtsidentifikation eine detailliertere Beschreibung, da diese Verfahren in der Software „OpenCV“ implementiert sind [7].

Die genannten verfahren bilden einen Ausschnitt über eine vielzahl von Verfahren, die über die Jahre entwickelt wurden. Um sich über weitere Verfahren zu informieren sind die Literaturübersichten von Jafri und Arabnia [10]; Zhao, Chellappa, Phillips und Rosenfeld [2]; Brunelli und Poggio [14] hilfreich.

1) Eigenfaces Methode: Die Eigenfaces Methode ist eine auf der Hauptkomponentenanalyse oder im engl. „Principal Component Analysis“ (PCA), basierende Methode zur Gesichtserkennung und wurde von Matthew Turk und Alex Pentland am Massachusetts Institute of Technology entwickelt [15].

Der erste Schritt, um Gesichter zu erkennen, ist das Erlernen der Gesichter. Hierfür werden mehrere Grauwertbilder pro Person benötigt. Diese Bilder sollten eine quadratische Größe von $N * N$ Pixeln besitzen und vier Bildern pro Person bestehen, die sich durch Veränderungen von Lichtverhältnissen und Emotionen unterscheiden, so ist die Empfehlung in [15].

Um nun ein Durchschnittsgesicht Ψ zu bilden, wird jedes Bild $(\Gamma_1, \Gamma_2, ..., \Gamma_n)$ durch einen N^2-Dimensionalenvektor dar-

gestellt. Das Durchschnittsgesicht Ψ wird wie folgt gebildet:

$$\Psi = \frac{1}{n} * \sum_{i=1}^{n} \Gamma_i \tag{3}$$

Anschließend wird wie folgt der Differenzvektor Φ für jedes Bildes zu dem Durchschnittsgesicht berechnet:

$$\Phi_i = \Gamma_i - \Psi \tag{4}$$

Diese Differenzvektoren werden dann der PCA unterzogen, um die Vektoren zu finden, die innerhalb der n Bilder die Verteilung am besten darstellen. Die linearen Unterschiede lassen sich mit der Kovarianzmatrix aufstellen die sich wie folgt bildet:

$$C = \frac{1}{n} \sum_{i=1}^{n} \Phi\Phi^T \tag{5}$$

Aus der Kovarianzmatrix lassen die Eigenwerte und Eigenvektoren (Eigenfaces) bilden, welche dann auch abgespeichert werden. Kurzum die PCA dient dazu die Zusammenhänge sichtbar zu machen und auf zweidimensionaler Ebene zu projizieren.

Nachdem nun die Eigenvektoren gefunden wurden, die den größten Teil der Varianz zum Durchschnittsgesicht repräsentieren, wird für jede Person wie die Hauptkomponente ausgewählt:

$$\lambda_k = \frac{1}{n} * \sum_{i=1}^{n} (u_k^t * \Phi_i)^2 \tag{6}$$

In der Praxis werden die Eigenvektoren u_k und Eigenwerte λ_k gefunden, indem die Eigenwerte und -vektoren der Kovarianzmatrix aller Differenzbilder berechnet werden.

Abbildung 5 zeigt oben links ein Durchschnittsgesicht Ψ, die beiden anderen oberen Gesichter zeigen Eigenfaces aus den ersten Eigenwerten. Die untere reihe zeigt Eigenfaces aus den letzten Eigenwerten.

Abbildung 5. Eigenfaces aus [16]

Nachdem nun die Datenbank Trainiert, ist kann eine Person wie folgt identifiziert werde. Das zu identifizierende Bild Γ wird wie folgt in seine Eigenface-Komponenten oder Gewichtungen ω zerlegt:

$$\omega_k = u_k^T(\Gamma - \Psi) \tag{7}$$

Die einzelnen Gewichtungen ω_w bilden einen Vektor $\Omega^T = [\omega_1, \omega_2, ..., \omega_n]$. Dieser Vektor wird dann verwendet, um festzustellen ob das Gesicht bekannt ist, oder nicht. Hierbei wird der Euklidisch Abstand vom Gewichtungsvektor Ω_k und dem zu erkennenden Gesicht Ω berechnet.

$$e_k = ||\Omega - \Omega_k||^2 \tag{8}$$

Befindet sich e_k unter einer bestimmten Entscheidungsschwelle, wird das Gesicht als bekannt eingestuft. Die Zuordnung erfolgt bei der der Ω-Vergleich das geringste e_k^2 ergibt [15].

2) Fisherfaces Methode: Bei der Fisherfaces Methode wird ähnlich wie bei der Eigenfaces Methode das Bild Γ durch einen N^2-Dimensionalvektor dargestellt. Im Gegensatz zur Anwendung des PCA, wird hier die Fisher's Linear Discriminant Analysis (FLD) angewendet [17]. Die Fisherfaces Methode wurde an der Yale University entwickelt und wies mit geringen Datenmengen eine höhere Erkennungsrate als bei Eigenfaces auf [17].

Im unterschied zu Eigenfaces, bei der die Varianz aller Bilder zueinander betrachtet wird, wird bei der Fisherfaces Methode eine Gesichtsklasse, also alle Bilder derselben Person betrachtet und so die Streuung zwischen den unterschiedlichen Gesichtsklassen maximiert, um so eine höhere Erkennungsrate zu erzielen.

Um nun hier die Datenbank zu Trainieren wird zuerst die Streuung zwischen den Gesamten Gesichtsklassen berechnet. Wobei N_i die Anzahl der Bilder in der jeweiligen Klasse, μ_i das Durchschnittsgesicht (Berechnung analog zu Formel (3)) der jeweiligen Klasse, μ das Durchschnittsgesicht Klassenübergreifend

$$S_B = \sum_{i=1}^{c} N_i * (\mu_i - \mu) * (\mu_i - \mu)^T \tag{9}$$

Gesichtsklassen intern wird die Streuung wie folgt berechnet, wobei x_k die jeweiligen Einzelbilder der Klassen bezeichnet und X_i, das Durchschnittsgesicht der jeweiligen Klasse:

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i) * (x_k - \mu_i)^T \tag{10}$$

Die optimale Transformierte Matrix einer Gesichtsklasse W_opt die den maximale Streuung ergibt wird wie folgt berechnet, womit W den Transformationsvektor einer Gesichtsklasse bezeichnet:

$$W_{opt} = arg\ max \frac{|W^T * S_B * W|}{|W^T * S_W * W|} \tag{11}$$

Die Eigenwerte und -vektoren ergeben sich wie folgt:

$$S_W^{-1} S_B)E = E\lambda \tag{12}$$

Abbildung 6 zeigt den Vergleich von FLD und PCA bei Tranformation vom 2D-Raum in den 1D-Raum. Die getrichelte linie von links unten nach rechts oben stellt den PCA-Raum dar und die von links oben nach recht unten den FLD-Raum. Es ist hierbei klar erkennbar, dass die FLD eine größere Streuung zwischen den Klassen erreicht und die Daten linear trennbar. Beim PCA wird eine größere Streuung erreicht und die Klassen sind vermischt.

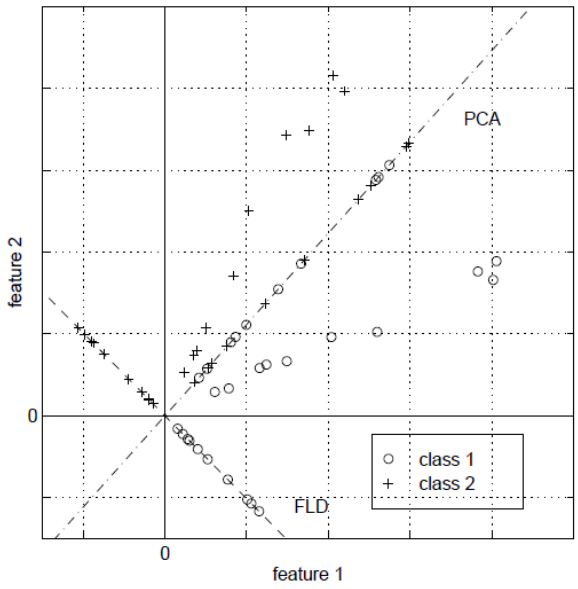

Abbildung 6. Vergleich von FLD und PCA [17]

In der praktischen Umsetzung wurde die Fisherface-Methode zur Gesichtsidentifikation verwendet. Warum diese Methode verwendet wurde und nicht die Eigenface-Methode findet sich in Kapitel III-D.

III. Software

In diesem Kapitel wird die Software „OpenCV“ [7] kurz vorgestellt. Des Weiteren wird eine Schritt für Schritt Installation und Konfiguration unter Mac OSX vorgestellt. Darüber hinaus wird eine Liste von Software-Paketen vorgestellt, um die Installation reibungslos durchzuführen.

Die Entscheidung zwischen die beiden Algorithmen (Fisherface und Eigenface) wird hier in diesem Kapitel erläutert. Danach wird ein Beispiel Code vorgestellt, um die Installation sowie Konfiguration zu testen. Zum Schluss wird die Implementierung und Problemstellung vorgestellt.

A. OpenCV

OpenCV steht für „Open Source Computer Vision Library” und wurde erstellt, um eine gemeinsame Infrastruktur für die Computer-Vision-Anwendungen zur Verfügung zu stellen. OpenCV ist ein BSD-lizenziertes Produkt und wurde im September 2006 in der Version 1.0 rausgebracht. Die Bibliothek beinhaltet mehr als 2500 optimierte Algorithmen wie z.B. für die Gesichtserkennung [7].

OpenCV läuft auf fast allen Plattformen (Linux/Mac, Windows, Android sowie Apple iOS) und ist online mit der Version 3.0.0 erhältlich [7].

Um das Projekt, Betriebssystem unabhängig zu halten und auch eine Portierung z.B. für Raspberry PI zu ermöglichen, haben wir uns für die Programmiersprache C++ entschieden.

B. Installation

Um mit OpenCV arbeiten (Unter MAC OSX) zu können, müssen folgende Software Pakete Installiert werden:

- MacPorts (https://www.macports.org/)
- Cmake (http://www.cmake.org)
- OpenCV (http://opencv.org/)

Wir beginnen die Konfiguration für Mac OSX mit der Installation von Macports. Nach der Installation kann über das Terminal Fenster mit dem Befehl `port` geprüft werden, ob die Installation erfolgreich war.

Nach der Installation von Macports, widmen wir uns „cmake” und tippen in das Terminal Fenster `\$sudo port install cmake`. Damit wird die Installation von Cmake über die Software Macports gestartet. Wenn die Installation von Cmake fehlschlagen sollte, muss vor der Installation ein clean durchgeführt werden `$sudo port clean cmake`.

Nach der Installation von Macports und Cmake, Entpacken und Kopieren des Verzeichnisses von OpenCV beispielsweise nach `.../FH_BRS/Seminar/OpenCV`.

Wir wechseln über das Terminal-Fenster in das Verzeichnis von OpenCV und erstellen ein Verzeichnis mit dem Namen „build“ und wechseln in das Verzeichnis Build. Soweit alles erfolgreich war, kann das build für OpenCV mit dem Befehl `$cmake -G "Unix Makefiles".` erstellt werden. Im Verzeichnis von OpenCV Build führen wir über das Terminal Fenster den Befehl `$cmake -j8` um mehr OpenCV Bibliotheken zu erstellen. Die Installation schließen wir mit dem Befehl `$sudo cmake install`.

C. Konfiguration

Die Konfiguration von OpenCV in Xcode beginnt mit dem Projekt, Erstellung von Typ „Command Line Tool“ unter Application (zu finden unter OSX). Auf der nächsten Seite ist zu beachten, dass die Programmiersprache „C++“ ausgewählt ist. Nach dem das Projekt erstellt worden ist, klicken wir auf das Projekt und wählen „Build Settings“ aus und suchen nach search paths.

Die folgenden Einstellungen müssen erstellt werden:

```
Always Srarch User Paths: YES
Framework Search Paths:/usr/local/lib
Header Search Paths:/usr/local/include
Library Search Paths:
../OPenCV/opencvFW/build/lip
```

Des Weiteren sollten im Projekt die OpenCV Bibliotheken hinzugefügt werden. Hierfür klicken wir mit der rechten Maustaste auf dem Projektverzeichnis und wählen „Add File to...“ um die benötigten Bibliotheken zum Build Verzeichnis hinzuzufügen.

Ein Test Programm soll prüfen, ob die Einstellungen ohne Fehler übernommen wurden und wir mit OpenCV Framework arbeiten können. Hier ein Code

Listing 1. TestProgramm

```
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/core/core.hpp"
```

```
using namespace cv;

int main(){
    VideoCapture cap(0);
    while (true){
        Mat Webcam;
        cap.read(Webcam);
        imshow("Webcam", Webcam);
    }
}
```

Mit dem Code starten wir mit Hilfe von OpenCV die Kamera und zeigen das Bild in einem Frame. Zu beginn inkludieren wir dir nötigen Bibliotheken, um auf die entsprechende Klassen bzw. Funktionen zugreifen zu können, um diese in unserem Programmcode nutzen können.

Die `core.hpp` Bibliothek liefert die Kernfunktionalitäten unter anderem ein Multi-Dimensional Feld `Mat` und weitere nützliche Funktionen (OpenCV Introduction). Die nächste Bibliothek nennt sich `highgui.hpp`. Sie dient nicht nur als GUI sondern auch als Bilder und Video Codecs. Mit der nächsten Bibliothek `imgproc.hpp` wird ein Bildverarbeitungsmodul geliefert. Unter anderem nicht lineare und lineare Bildfilterung, geometrische Bildtransformationen, Farbraumumwandlung und auch Histogramme.

Mit der Funktion `VideoCapture cap(0);` wird auf die Kamera (Webcam) zugegriffen. Die Null steht für die Kamera ID. Ein Objekt `Webcam` vom Typ `Mat` wird erstellt . Das Objekt wird mit der Funktion `cap` aufgerufen und gelesen. Die nächste bzw. letzte Zeile `imshow("Webcam", Webcam);`, wird ein Fenster erstellt mit dem Titel `Webcam` und mit dem Objekt `Webcam`.

D. Algorithmen

Die Algorithmen sind für OpenCV schon implementiert und befinden sich in der `contrib.hpp` Bibliothek. Wie im Kapitel II-C2 beschrieben arbeitet Eigenfaces mit PCA und Fisherfaces mit LDA. Dies spielt für die Gesichtserkennung eine sehr wichtige rolle. Da Fisherfaces mit wenig Bildern eine bessere Erkennung liefert wurde diese Implementierung verwendet [17], wobei sich wie in der Arbeit von Belhumeur et al. [17], die starke Empfindlichkeit von der Helligkeit bemerkbar macht bei der Erkennungsrate.

E. Implementierung

Die folgende Abbildung 7 stellt grafisch die Gesichtserkennung und die Zuordnung funktioniert dar. In der Abbildung als „Face Detection“ bezeichnete Bereich repräsentiert das Gesichtserkennungsmodul. Hier wird die Viola-Jones Methode angewendet. Sobald ein Gesicht erkannt ist, erfolgt die Identifikation. Dies passiert in der „Face Recognition“, mit Hilfe des Fisherfaces-Algorithmus, welcher in der Datenbank nach bekannten Gesichtsmustern sucht.

Für die Umsetzung einer Gesichtserkennungs-Software, werden Gesichtsmuster benötigt. Diese Gesichtsmustererkennung kommt mit der OpenCV Software als Beschreibungsdateien. Die Beschreibungsdatei ist zu finden in `data/haarcascades` vom OpenCV Ordner. Hieraus wird aus vielen hunderten oder tausenden von Bildern erstellt und als XML Datei gespeichert. Dieses Verfahren existiert seit

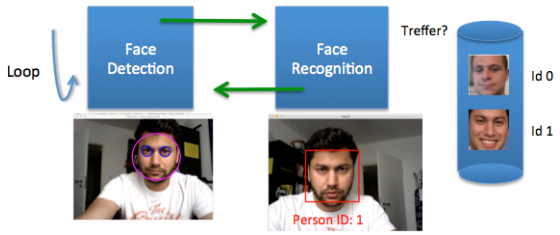

Abbildung 7. Gesichtserkennung Ablauf

2001 und ist von Paul Viola und Michael Jones entwickelt worden. Bekannt sind die Haarcascade Dateien auch als „Viola-Jones Detection“.

Folgende Beschreibungs-dateien werden Standardmäßig von OpenCV mitgeliefert:

```
haarcascade_eye_tree_eyeglasses.xml
haarcascade_eye.xml
haarcascade_frontalface_alt_tree.xml
haarcascade_frontalface_alt.xml
haarcascade_frontalface_alt2.xml
haarcascade_frontalface_default.xml
haarcascade_fullbody.xml
haarcascade_lefteye_2splits.xml
haarcascade_lowerbody.xml
haarcascade_mcs_eyepair_big.xml
haarcascade_mcs_eyepair_small.xml
haarcascade_mcs_leftear.xml
haarcascade_mcs_lefteye.xml
haarcascade_mcs_mouth.xml
haarcascade_mcs_nose.xml
haarcascade_mcs_rightear.xml
haarcascade_mcs_righteye.xml
haarcascade_mcs_upperbody.xml
haarcascade_profileface.xml
haarcascade_righteye_2splits.xml
haarcascade_smile.xml
haarcascade_upperbody.xml
```

Neben der Beschreibungsdateien wird eine Art Datenbank benötigt, woraus die Software Bilder zum Vergleich entnehmen kann. Hierfür benötigen wir mindestens 10 Bilder pro Person und Speichern diese in 50x50px als JPEG ab.

Um Bilder an die Gesichtserkennungs-Software weitergeben zu können, brauchen die Bilder eine ID. Die Realisierung wird mit Hilfe einer CSV Datei durchgeführt. Die CSV Datei soll eine Datenbank repräsentieren und soll Bild Ort (Bild Pfad) und die ID enthalten.

Beide müssen durch ein Semikolon getrennt werden. Hier ein Beispiel:

```
.../db/robert_malz1.jpg;0
.../db/robert_malz2.jpg;0
.../db/robert_malz3.jpg;0
.../db/idris_nawid1.jpg;1
.../db/idris_nawid2.jpg;1
.../db/idris_nawid3.jpg;1
```

Eine Methode, um die CSV zu lesen, heißt `read_csv`. Diese Funktion kann mit einer Datenbank Lesefunktion verglichen werden. Hinzugefügt wurde im Code String Felder, um die ID der CSV-Datei Personennamen anzeigen zu können.

Danach muss noch angegeben werden, wo sich die Beschreibungsdatei befindet. Diese werden dann beim Programmausführung ausgeführt [18].

Der Algorithmus, hier „FisherFaceRecognizer“ wird aus der OpenCV Bibliothek aufgerufen und auf die gelesenen Bilder sowie ID aus der CSV-Datei justiert:

```
Ptr<FaceRecognizer> model =
createFisherFaceRecognizer();
model->train(images, labels);
```

Um die Beschreibungsdatei laden zu können, muss ein „CascadeClassifier“ Objekt erstellt werden. Die Methode bekommt dann die Beschreibungsdatei zugewiesen, um die Datei zu laden.

```
CascadeClassifier haar_cascade;
haar_cascade.load(fn_haar);
```

Die Bilderkennungsprognose bekommt hier ein rotes Rechteck um das Gesicht und ein Text mit der ID bzw. mit dem Namen über dem Rechteck angegeben. Hier der Code.

```
int prediction=
model->predict(face_resized);
detected face:
rectangle(original,face_i,
        CV_RGB(0,255,0),1);
string box_text;
box_text = format("Name=");
if (prediction >= 0)
{
box_text.append(personen[prediction]);
}
```

Zum Schluss wird in der FOR-Schleife die Webcam aufgerufen mit `imshow("Gesichtserkennung", original);` und bleibt so lange aktiv bis das Programm beendet wurde.

IV. Zusammenfassung

In der vorliegenden Seminararbeit wurde eine kurze Einführung in das Thema der automatischen Gesichtserkennung gegeben, die über die implementierten Techniken hinausgeht.

Als Hilfswerkzeug für die Implementierung wurde OpenCV verwendet. Zwar ist die Konfiguration von OpenCV nicht trivial, liefert dafür eine Vielzahl von implementierten Algorithmen. Nach der Implementierung eines Beispiel Programmes, liefert die Software leider nicht immer zufriedenstellende Ergebnisse. Dies muss untersucht werden und das Beispiel Programm optimiert werden.

In zukünftigen Arbeiten kann man die holistische und merkmalsbasierte Gesichtsidentifikation kombinieren, um ggf. die Erkennung zu verbessern. Auch eine Implementierung auf ein RaspberryPi 2 sollte analysiert und umgesetzt werden.

Bei dem Szenario der Gesichtserkennung an der Haustür, werden zwar Gesichter erkannt die bekannt sind, aber unbekannte Gesichter können nicht eingeschätzt werden was deren Ziel ist. Hier kann man analysieren, wie man die Laune und Aggressionspotential anhand von Bildern erkennen kann und in die Software implementiert.

Literatur

[1] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb

[2] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face Recognition: A Literature Survey,” in *ACM Computing Surveys*, Dec. 2003, pp. 399–458.

[3] R. Frischholz, “Face Detection Techniques,” 2015. [Online]. Available: https://facedetection.com/techniques/

[4] E. Hjelmås, C. B. Lerøy, C. B. Ler#y, and H. Johansen, *Detection and Localization of Human Faces in the ICI System: A First Attempt*, 1998.

[5] F. C. Crow, “Summed-area Tables for Texture Mapping,” in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’84. New York, NY, USA: ACM, 1984, pp. 207–212. [Online]. Available: http://doi.acm.org/10.1145/800031.808600

[6] “kylemcdonald/AppropriatingNewTechnologies,” Jun. 2014. [Online]. Available: https://github.com/kylemcdonald/AppropriatingNewTechnologies

[7] “OpenCV.” [Online]. Available: http://opencv.org/

[8] Y. Freund and R. E. Schapire, *Experiments with a New Boosting Algorithm*, 1996.

[9] H. Ritter, “Vorlesung Lernarchitekturen,” 2006. [Online]. Available: https://ni.www.techfak.uni-bielefeld.de/files/script-ML2.pdf

[10] R. Jafri and H. R. Arabnia, “A Survey of Face Recognition Techniques.” *JIPS*, vol. 5, no. 2, pp. 41–68, 2009. [Online]. Available: http://www.cosy.sbg.ac.at/~uhl/face_recognition.pdf

[11] T. Kanade, “Picture Processing System by Computer Complex and Recognition of Human Faces,” in *doctoral dissertation, Kyoto University*, Nov. 1973.

[12] A. Nefian and I. Hayes, M.H., “Face detection and recognition using hidden Markov models,” in *1998 International Conference on Image Processing, 1998. ICIP 98. Proceedings*, vol. 1, Oct. 1998, pp. 141–145 vol.1.

[13] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg, “Face recognition by elastic bunch graph matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, Jul. 1997.

[14] R. Brunelli and T. Poggio, “Face recognition: features versus templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, Oct. 1993.

[15] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, Jan. 1991. [Online]. Available: http://dx.doi.org/10.1162/jocn.1991.3.1.71

[16] S. Zhang and M. Turk, “Eigenfaces,” *Scholarpedia*, vol. 3, no. 9, p. 4244, 2008. [Online]. Available: http://www.scholarpedia.org/article/Eigenfaces

[17] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997. [Online]. Available: http://dx.doi.org/10.1109/34.598228

[18] “Face Recognition in Videos with OpenCV — OpenCV 2.4.11.0 documentation.” [Online]. Available: http://docs.opencv.org/modules/contrib/doc/facerec/tutorial/facerec_video_recognition.html