Entwicklung einer Gesichtserkennenden Software zur Erfassung und Analyse von Personenströmen

Bachelor Thesis

vorgelegt von

Artur Schens

Betreuer:	Dipl.-Inform. Markus Maron
Erstgutachter:	Dipl.-Inform. Markus Maron
Zweitgutachter:	Prof. Dr. Ulrich Furbach

Koblenz, im April 2014

Eidesstattliche Erklärung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet–Quellen – benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

Ort,Datum

Unterschrift

Zusammenfassung

Diese Bachelorarbeit befasst sich mit der Entwicklung und Implementierung einer Gesichtserkennenden Software, die in der Lage ist Personenströme zu erkennen und zu protokollieren. Dabei wird, ausgehend von den speziellen Anforderungen der Bildverarbeitung die entstandene Softwarearchitektur und deren Implementation vorgestellt. Zusätzlich wird zur Implementation ein Webinterface entwickelt welches die Verwaltung der Daten vereinfachen soll. Abschließend werden weitere Verfahren der Gesichtserkennung vorgestellt und gegen das eingesetzte Verfahren verglichen. Zum Schluss wird die implementierte Software evaluiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Gesichtserkennung ist eines von mehreren biometrischen bekannten Verfahren [26] die zur Identifizierung von Personen eingesetzt werden kann. In vielen Bereichen hat die biometrische Identifizierung und Authentifizierung durch die Gesichtserkennung entscheidende Vorteile. Im Bereich der Marktanalyse kann die Gesichtserkennung eingesetzt werden um Käufe und Wege von Kunden beobachten zu können. Im Gegensatz zu anderen Verfahren, wie das Tracking durch Bluetooth oder Wi-Fi, können so auch Kunden ohne Mobilfunkgerät ermittelt werden. In sicherheitskritischen Bereichen hat eine Gesichtserkennungssoftware entscheidende Vorteile gegenüber klassischen Verfahren. Angriffspunkte verschwinden oder werden minimiert, wie das Abfragen eines Passwortes oder das Nutzen von Schlüsseln. Gesichtserkennende Software wird bereits heutzutage eingesetzt. Seit Oktober 2010 setzt die Stadt Rotterdam in Bahnlinien solche Systeme ein um Gewalt und Vandalismus zu unterbinden [19]. Auch in sozialen Netzwerken wird gesichtserkennende Software eingesetzt. Facebook nutzt diese um Nutzer auf Fotos markieren zu können. So können neue Fotos direkt einer Person zugeordnet werden. Facebook ermöglicht dem Nutzer der Platform seine Fotos automatisch erkennen zu lassen, indem das Taggen automatisiert wird [27]. Solche autonomen Systeme werfen auch Fragen des Datenschutzes auf. Das persönlichste Merkmale, das Gesicht, landet in einer Datenbank und kann zu den verschiedensten Zwecken genutzt werden. Eine erfasste Person kriegt meistens von der Speicherung nichts mit. Wie und wo diese Daten abgelegt und verarbeitet werden ist in der Regel nicht nachvollziehbar. Das zu entwickelnde System soll zeigen das Personen erfolgreich durch ein Kamerasystem verfolgt und identifiziert werden können. Die Abschlussarbeit wurde entwickelt um eine eine Alternative zu Trackingverfahren durch Bluetooth oder WiFi aufzuzeigen.

1.2 Zielsetzung

Ziel der Arbeit ist die Entwicklung eines Systems das in der Lage ist Gesichter in Bildern, die durch eine Webcam aufgenommen werden, zu extrahieren und diese gegen eine Datenbank mit anderen Gesichtern abzugleichen. Wenn ein passendes Gesicht in der Datenbank gefunden wurde wird der ausgewählte Datensatz erweitert, ansonsten wird ein neuer angelegt. Das System soll in der Lage sein anhand von Gesichtern das Geschlecht der Person bestimmen zu können. Die gesammelten Daten sollen grafisch ausgewertet und angezeigt werden. Dazu müssen bekannte Verfahren zur Gesichtsdetektierung, Merkmalsextraktion und Geschlechtsklassifizierung analysiert und angewandt werden.

1.3 Aufbau der Arbeit

Diese Ausarbeitung soll den Entwurfs- und Entstehungsprozess einer Software dokumentieren, mit der Gesichter aus Aufnahmen einer Webcam, extrahiert, analysiert und verglichen werden. In Kapitel 2 werden Grundlagen von elementaren Verfahren der Bildverarbeitung und der Klassifikation von Strukturen erläutert. In Kapitel 3 wird die Anforderungsdefinition und das Konzept der Software behandelt. Hier wird beschrieben wie die Software aufgebaut ist und wie diese arbeitet. Anschließend wird in Kapitel 4 die Implementation erläutert. In diesem Kapitel werden die verwendeten Frameworks behandelt und die Vorgehensweise, die für die Implementierung der Software notwendig ist. In Kapitel 5 werden weitere mögliche Verfahren zur Gesichtserkennung vorgestellt und diese mit dem eingesetzten Verfahren verglichen. Anschließend wird der entwickelte Prototyp evaluiert und dessen Performanz und Effizienz ermittelt. Zum Schluss wird der Prozess der Entwicklung evaluiert. Dabei sollen Probleme, die sich innerhalb des Entwicklungsprozesses ergeben haben, dokumentiert werden um daraus ein Fazit ziehen zu können. Danach soll aufgezeigt werden, inwiefern sich dieses Projekt erweitern lässt und welche Möglichkeiten sich zukünftig anbieten würden.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen behandelt die zur Konzeption notwendig sind. Im ersten Abschnitt 2.1 werden die grundlegenden Operationen der Bildverarbeitung erläutert die im System eingesetzt werden. Dazu gehört die Aufnahme von Bildern, die Abbildung von Bildsignalen im Speicher sowie die Aufbereitung der Bilder. Die Aufbereitung der Bilder umfasst die Umwandlung der Bilder in Graustufen sowie die Ermittlung von Histogrammen. Beide Verfahren sind essentiell für die Anwendung im SURF Algorithmus der im nächsten Abschnitt 2.3 behandelt wird. Im letzten Abschnitt 2.4 werden die Supported Vector Machines vorgestellt die zur Bestimmung des Geschlechts genutzt werden.

2.1 Bildverarbeitung

Die Bildverarbeitung befasst sich mit der Aufnahme, Speicherung und Aufbereitung von Bildern. Bilder werden durch eine Kamera, den Bildsensor, erfasst und in Digitale Signale umgewandelt. Die Speicherung der Bilder beinhaltet die Abbildung von Bildsignalen auf Datenstrukturen und deren Persistierung. Die Aufbereitung dient der besseren Interpretation der Digitalen Bildsignale und ist Vorraussetzung für eine Weiterverarbeitung und Bildanalyse [23]. Die Bildweiterverarbeitung wird in eine Low-Level Verarbeitung und eine High-Level Verarbeitung unterschieden [21]. Bildanalysen werden meistens auf einem High-Level durchgeführt. Durch Methoden der Bildverarbeitung kann die Qualität des Bildes verbessert, der Datenumfang komprimiert oder der Bildinhalt analysiert werden. Letzteres kann dazu benutzt werden um Muster in Bildern zu finden und so Objekte auf Bildern ausfindig zu machen. Verschiedene Methoden der Bildverarbeitung werden in der Medientechnik, Medizintechnik, Automobilbranche oder der Sicherheitsbranche angewandt.

2.1.1 Aufbereitung von Bildern

Bei der Aufbereitung von Bildern wird zwischen bilderzeugenden Verfahren und den informationensgewinnenden Verfahren unterschieden. Bilderzeugende Verfahren verändern das Bild in seiner Struktur. Informationsgewinndende Verfahren zielen darauf ab neue Informationen anhand des Bildes zu erzeugen. Die Umwandlung eines Bildes in eine Graustufe ist ein erzeugendes Verfahren. Die Berechnung eines Histograms hingegen ist ein informationsgewinnendes Verfahren. Beide Operationen, die Umwandlung in eine Graustufe und die Berechnung des Histograms, werden im dem geplanten System eingesetzt und in den nächsten beiden Unterabschnitten weiter erläutert.

Graustufe

Die Graustufe eines Bildes ist die Abbildung der Pixelwerte auf die Lichtintensitätswerte. Der Intensitätswert liegt in einem numerischen Intervall, auch Graustufenintervall genannt. Dabei bildet der niedrigste Wert auf dem Intervall die niedrigste Lichtintensität und der größte Wert die höchste Lichtintensität ab. RGB Bilder enthalten für jeden Pixel Informationen über dessen Rot, Grün und Blauwert. Um den Grauwert vom RGB-Pixel zu finden wird der arithmetische Mittelwert der einzelnen Werte gebildet und auf das Graustufenintervall abgebildet. Üblicherweise ist der größte Wert im Graustufenintervall das maximale arithmetische Mittel der RGB Werte.

Histogram

Ein Histogram ist eine graphische Repräsentation der Verteilung der Daten. In der digitalen Bildverarbeitung werden Histogramme eingesetzt um Informationen über ein Graustufenbild zu erhalten und um diese zu normalisieren. Dabei gibt ein Histogram eines Graustufenbildes die Verteilung der einzelnen Graustufen an [7].

2.2 Objektdetektion

Die Detektion von Objekten durch Haar feature-based Cascade Klassifizierer ist eine sehr effiziente Methode um Objekte zu erkennen und wurde 2001 von Paul Viola und Michael Jones vorgestellt [22]. Die Methodik gehört zu den des maschinelles Lernens. Eine *Cascade* Funktion wird trainiert mit einer großen Menge von positiven und negativen Bildern. Die Funktion wird dann genutzt um Objekte zu detektieren. Dabei enthält ein positives Bild das Objekt das detektiert wird und das negative alles bis auf das was detektiert werden soll. In der geplanten Software sollen Gesichter detektiert werden, dazu werden viele positive Bilder mit Gesichtern und negative Bilder ohne Gesichter benötigt um den Klassifizierer zu trainieren. Danach werden von jedem Bild die Merkmale

extrahiert werden. Merkmale werden mit Haar-Features extrahiert, siehe Abbildung 2.1. Jedes Merkmal ist ein Wert der durch die Subtraktion der Summe

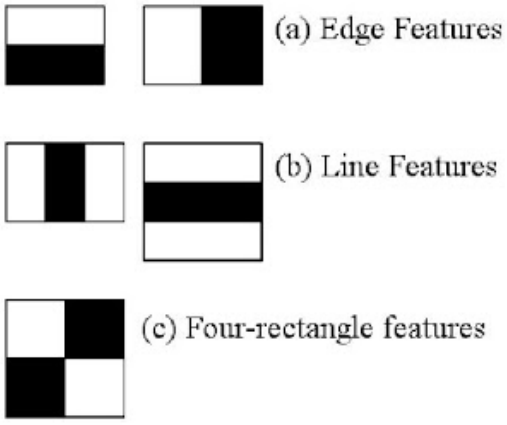

Abbildung 2.1: Benutzte Haar Features.
Quelle: [18]

der Pixel unter der weißen Fläche von der Summe der Pixel unter der Schwarzen Fläche errechnet wird. Jedes Haar-Features wird in allen Größen an jeder Position berechnet. Um diesen Prozess zu vereinfachen können Integralbilder genutzt werden. Dadurch kann enormer Rechenaufwand eingespart werden. Die besten Features werden mit Hilfe von Adaboost ausgewählt [15]. Anschließend wird der Prozess wiederholt. Die besten Haar-Features die ein Gesicht beschreiben oder nicht werden ausgewählt, der Rest verworfen. Der Prozess wird so lange wiederholt bis eine gewünschte Genauigkeit erreicht wird. Des Resultat ist eine Menge von geeigneten Haar-Features die ein Gesicht beschreiben können. Im Bild wird dann ein Fenster ausgewählt und überprüft ob dort ein Gesicht sein könnte. Die Features die in der ersten Stufe ausgwählt wurden werden hier benutzt. Sollte die erste Stufe kein Gesicht klassifizieren können so wird das Fenster verworfen und an der nächsten Stelle weitergesucht [18].

2.3 Speeded up Robust Features (SURF)

Die Identifikation von Objekten spielt eine große Rolle beim künstlichen Sehen. Typische Anwendungsfälle der Objektdetektion finden sich in mehreren Bereichen. In der Automobilbranche kann die Objektdetektion eingesetzt werden um Verkehrzeichen zu erkennen. In der Robotik ist eine Objektdetektion eines der wichtigsten Gebiete und dient dem Roboter zur Wahrnehmung und zur Interaktion mit seiner Umgebung. Um Objekte detektieren und identifizieren zu können müssen wichtige Bildmerkmale aus Bildern oder Bildsequenzen extrahiert werden. Diese Merkmale können mit anderen Merkmalen verglichen werden um so ein Objekt auf einem Bild identifizieren zu können. Die Bildmerkmale sollten möglichst invariant gegenüber Drehung, Verschiebung, Skalierung

und Verändung vom Lichtverhältniss sein. Um diese Aufgabe erfüllen zu können wird der Speeded up Robust Features Algorithmus, kurz SURF, vorgestellt. Der SURF-Algorithmus ist eine Erweiterung des SIFT-Algorithmus der von David Lowe vorgestellt wurde [14]. Da sich der SIFT-Algorithmus in der Praxis als zu langsam herausstellte wurde im Jahr 2006 der SURF-Algorithmus von Herbert Bay, Andreas Ess, Tinne Tuytelaars und Luc Van Gool vorgestellt [1]. Die erhöhte Performanz gegenüber dem SIFT-Algorithmus basiert darauf das die im SIFT Algorithmus verwendeten Gauß-Filter durch Mittelwertfilter ersetzt werden. Solche Mittelwertfilter können effizient durch Intergralbilder berechnet werden. Die nächsten Unterabschnitte erläutern die Berechnung von Integralbildern, die Detektion von Merkmalen mit Hilfe einer Hesse Matrix und wie Merkmale mit dem SURF-Algorithmus skalier- und rotationsinvariant gefunden werden können. Anschließend wird beschrieben wie die Merkmale benutzt werden können um Bilder zu identifizieren, das sogenannte Matching.

2.3.1 Integralbilder berechnen

Der Einsatz von Integralbildern dient der schnellen Berechnung von Pixelsummen in einem Bildausschnitt. Das Verfahren zur Schnellen berechnung von Pixelsummen wurde erstmals von Franklin C. Crow vorgestellt [5]. Ein Integralbild besteht aus Summen aus den vorherigen Pixelwerten die innerhalb eines aufgespannten Rechtecks liegen. Jeder Punkt ist die Summe aller Pixel innerhalb eines Rechtecks.

$$I(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} (i,j) \tag{2.1}$$

Die Berechnung 2.1 benötigt je nach Position $x * y$ Durchläufe. Durch eine rekursive Formulierung der Funktion 2.1 lässt sich die Summe in nur einem Durchlauf berechnen (vgl. [22]).

$$\sigma(x,y) = \sigma(x,y-1) + ß(x,y) \tag{2.2}$$

$$I(x,y) = I(x-1,y) + \sigma(x,y) \tag{2.3}$$

Wobei $\sigma(x,y)$ die Spaltensumme, $i(x,y)$ der Wert an der Stelle im Originalbild und $I(x,y)$ die Summe im Integralbild ist.

Die Abbildung 2.2 zeigt wie durch die rekursive Schreibweise die Pixelsumme Σ eines beliebigen Rechtecks durch vier Zugriffe berechnet werden kann.

$$\Sigma = D - B - C + A \tag{2.4}$$

2.3.2 Hesse Detektor

Der SURF-Algorithmus findet Merkmale mit Hilfe einer Hesse-Matrix. Dazu wird im Bild nach Stellen gesucht an denen die Determinante der Hesse-Matrix ein lokales Maximum hat. Um die einzelnen Einträge, die zu den x und

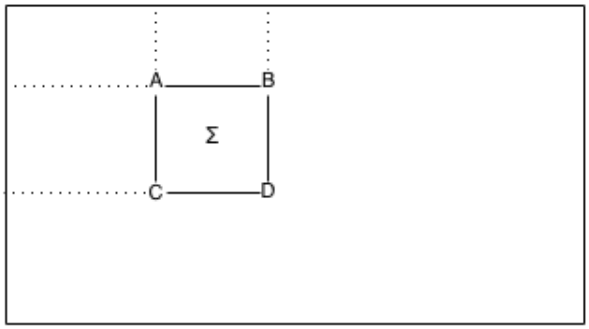

Abbildung 2.2: Berechnung vom Integralbild mit der rekursiven Formel.
Quelle: [1]

y Positionen im Bild korrespondieren, in der Hesse Matrix zu bestimmen wird beim SURF Algorithmus mit dem *Laplacian of Gaussians (LoG)* Operator gefaltet. Das heißt der Laplace-Operator wird auf eine Gauß-Funktion angewendet. Diese Operatoren sind partielle Ableitungen zweiter Ordnung und geben die Steigung einer Ebene am Punkt (x,y) in xy-Richtung an. Dadurch das eine Gauß Funktion benutzt wird kann die Glättung in jeder Skalierung angepasst werden und somit die Determinante auf jeder Skalierungsebene ermittelt werden. So kann der SURF Algorithmus Skalierungsinvariant arbeiten. Die Hesse Matrix am Punkt (x,y)ergibt sich aus der Funktion H die abhängig von der Position (x, y) und der Skalierung σ ist.

$$H((x,y),\sigma) = \begin{pmatrix} \Lambda_{xx}((x,y),\sigma) & \Lambda_{xy}((x,y),\sigma) \\ \Lambda_{xy}((x,y),\sigma) & \Lambda_{yy}((x,y),\sigma) \end{pmatrix} \tag{2.5}$$

Die Einträge Λ sind die zweiten partiellen Ableitungen mit der gefalteten Gauß-Funktion $\frac{\partial^2}{\partial x} g(\sigma)$ an der Position (x,y) vom Eingabebild. Ein lokales Extremum lässt sich anhand der Determinanten der Hesse Matrix ablesen.

$$det(H) = \Lambda_{xx} * \Lambda_{yy} * \Lambda_{xy}^2 \tag{2.6}$$

Ob dieser Punkt ein Extremum ist, also ein Merkmal darstellt, lässt sich mit folgender Vorschrift an der Determinanten ablesen.

$$(x,y) = \begin{cases} Sattelpunkt & \text{wenn } det(H) < 0; \\ Extremum & \text{wenn } det(H) > 0. \end{cases} \tag{2.7}$$

Wenn die Determinante von H negativ ist dann handelt es sich um einen Sattelpunkt. Wenn die Determinante positiv ist dann liegt ein Extremum vor. Um bestimmen zu können ob der Punkt (x,y) ein dunkler Punkt auf einem hellen Hintergrund oder ein heller Punkt auf einem dunklen Hintegrund ist, reicht die Berechnung der Determinanten von H nicht aus. Damit so eine Aussage getroffen werden kann wird die Spur der Hesse-Matrix H berechnet.

$$tr(H) = \Lambda_{xx} + \Lambda_{yy} \tag{2.8}$$

Wenn das Vorzeichen der Spur negativ ist handelt es sich um einen hellen Punkt auf einem dunklen Hintergrund, ansonsten anders herum. Um die Faltung Λ schneller berechnen zu können wird die Verwendung von Mittelwertfiltern, den sogenannten box filter, vorgeschlagen (vgl. [1]). Die Abbildung 2.3 zeigt die approximierten verwendeten Mittelwertfilter zu den LoG-Filtern. Durch die

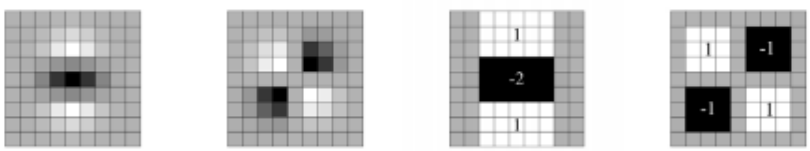

Abbildung 2.3: Von links nach rechts: die diskreten und zugeschnittenen partiellen Ableitungen, nach der Faltung mit einer Gauß-Funktion, in y und xy-Richtung und die dazu approximierten Mittelwertfilter. Die grauen Bereiche entsprechen einer 0. *Quelle: [1]*

Verwendung, der in Abbildung 2.3, gezeigten approximierten Mittelwertfiltern lässt sich die Determinante effizienter berechnen. Die Determinante an einer Position (x,y) wird als Blob bezeichnet.

2.3.3 Skalierungsinvarianz

Damit die gefunden Merkmale skalierungsinvariant sind müssen die Merkmale in verschiedenen Skalierungsebenen ermittelt werden. Im SIFT Algorithmus werden die Skalierungsebenen in einer Skalenraum-Pyramide, der sogenannten *scale-space pyramid*, ermittelt. Das Eingabebild wird inkrementell verkleinert und mit Gaußfiltern geglättet. Das inkrementelle Verkleinern der Bilder erzeugt eine Pyramide. Der SURF Algorithmus verkleinert nicht das Eingabebild, sondern vergrößert die Filtermasken die zur Berechnung der Blob-werte nötig sind. Durch die Benutzung von Integralbildern können die LoG-Filter unabhängig von ihrer Größe mit gleichbleibendem Aufwand berechnet werden. Die Abbildung 2.4 visualisiert beide Vorgehensweisen.

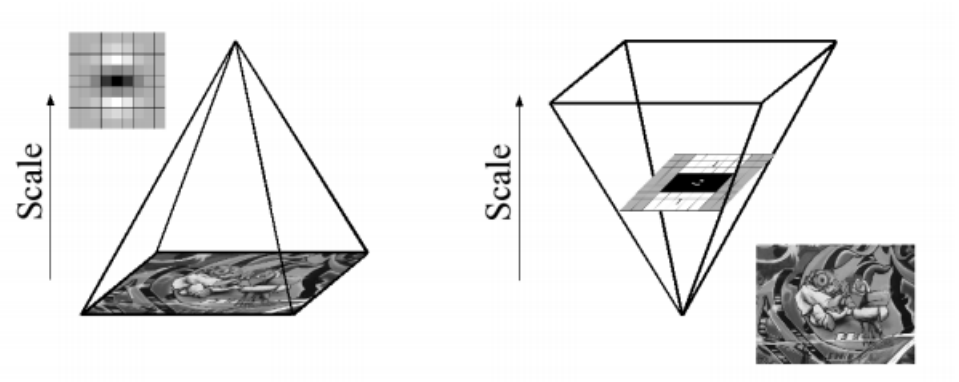

Abbildung 2.4: SIFT (links) verkleinert das Eingabebild bei gleich bleibendem Filter. SURF (rechts) vergrößert Filtermasekn bei gleich bleibendem Eingabebild.
Quelle: [1]

Der Skalenraum wir in *Oktaven* unterteilt. Jede Oktave entspricht der Filterung des Eingabebildes mit einer wachsenden Filtergröße. Dabei entspricht jede Oktave einem Skalierungsfaktor von 2 und wird weiter in eine konstante Anzahl von Skalierungsstufen unterteilt. Jede 9 x 9 Filtermaske von Abbildung 2.3 approximiert die LoGs mit der Varianz $\sigma = 1,2$ und berechnet das erste Intervall der ersten Oktave mit einer Skalierung von 1,2. Wenn der LoG-Filter vergrößert wird dann wird auch die Skalierung vergrößert. Die Skalierung und die Filtergröße stehen im Zusammenhang.

$$\sigma = \text{momentane Filtergröße} \frac{\text{Referenz-Filterskalierung}}{\text{Referenz-Filtergröße}} \tag{2.9}$$

Die Skalierungsschritte sind abhängig von der Teilfläche der LoG Filter, original auch $lobes$ genannt. Die Teilfläche bezeichnet entweder die schwarz oder weiß gekennzeichneten Flächen. Die kürzeste Linie einer Teilfläche beträgt stets ein Drittel der gesamten Filterlänge. Die Abbildung 2.6 zeigt wie die Vergrößerung vergenommen werden muss damit das Layout erhalten bleibt.

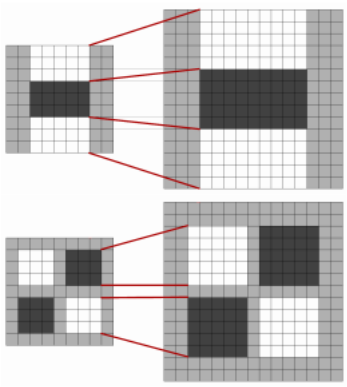

Abbildung 2.5: Filter D_{yy} (oben) und D_{xy} (unten) werden vergrößert. Der 9x9 Filter wird zu einem 15x15 Filter. Die länge der schwarzen Teilfläche kann nur um eine gerade Zahl an Pixeln vergrößert werden damit das Layout valide bleibt.
Quelle: [1]

Für jede folgende Oktave wird die Filtergröße verdoppelt. Daraus ergeben sich folgende Filtergrößen:

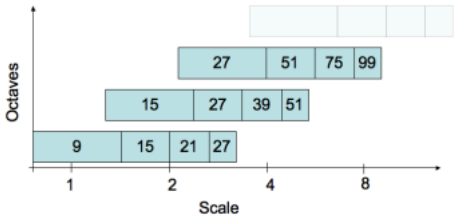

Abbildung 2.6: Verschiedene Filtergrößen in jeder Oktave.
Quelle: [1]

Nachdem die Oktaven erstellt wurden werden innerhalb der Skalenraum-Pyramide Interessenpunkte gefunden. Interessenpunkte sind die Punkte die ein lokales Extrema im Skalenraum bilden. Dazu wird in einer 3x3x3 Nachbarschaft jedes Blobs gesucht. Ist der Blob Wert betragsmäßig größer als seine 26 Nachbar-Blobs so handelt es sich um ein lokales Extremum.

2.4 Support Vector Machines (SVM)

Support Vector Machines, oder kurz SVMs, wurden erstmals 1992 von Boser, Guyon und Vapnik vorgestellt [3] und gehören zu den beaufsichtigten Lernalgorithmen die zur Klassifikation von Daten eingesetzt werden. Eine SVM gehört zu den linearen Klassifikatoren. Durch Methoden des maschinellen Lernens wird die Genauigkeit einer Klassifikation erhöht, während gleichzeitig Fehlklassifikationen verhindert werden. Man kann eine SVM auch als ein System

definieren, dass eine Menge von Objekten in einem Raum durch eine Hyperebene trennen kann. Wie in Abbildung 2.7 zu sehen ist können die Daten verschieden getrennt werden. Um die optimale Hyperebene zu bestimmen sucht man nach der Hyperebene die beide Klassen mit dem maximalsten Rand trennen kann.

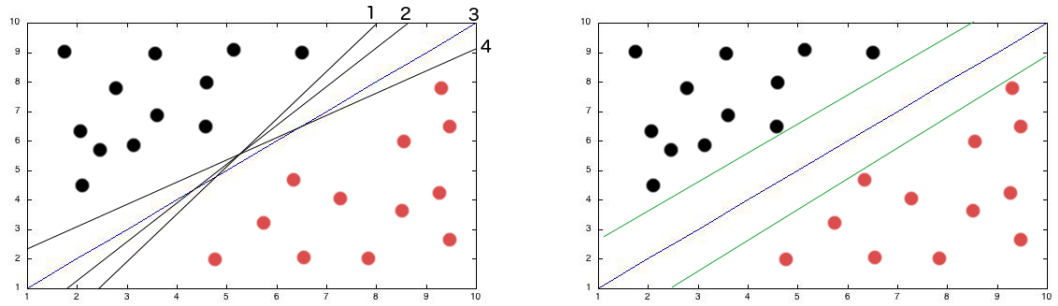

Abbildung 2.7: SVM mit linearen Trennungen und größtem Rand.
Quelle: [10]

Das Problem die optimale Hyperebene zu bestimmen ist ein Optimierungsproblem und kann wie folgt beschrieben werden. Die Daten bestehen aus der Menge δ.

$$\delta = \{(\vec{d_1}, c_1), ..., (\vec{d_i}, c_i) | i = 1..k, \vec{d_i} \in \mathbb{R}^n, c_i \in \{-1, 1\}\} \tag{2.10}$$

Die Menge δ dient als Trainingsmenge für die SVM. Sei nun δ eine durch eine Hyperebene linear separierbare Menge. Dann existiert eine Hyperebene der Form

$$\vec{w}^T \vec{x} + b = 0 \tag{2.11}$$

die beide Klassen linear trennen kann so, dass

$$\vec{w}^T \vec{x_i} + b < 0, \quad \text{für } y_i = -1$$
$$\vec{w}^T \vec{x_i} + b \geq 0, \quad \text{für } y_i = +1$$

wobei $\vec{w}^T$ der Normalenvektor zu der Hyperebene ist und b die senkrechte Distanz zum Ursprung der Hyperebene. Dann lässt sich eine Entscheidungsfunktion

$$g(\vec{x}) = \vec{w}^T \vec{x} + b \tag{2.12}$$

definieren die eine Distanz eines Elements zu der Hyperebene in dem gegebenen Raum beschreibt. Falls $g(\vec{x}) < 0$ dann wird das Element negativ klassifziert da es unter der Entscheidungsfläche liegt und falls $g(\vec{x}) \geq 0$ dann wird das Element positiv klassifiziert da es auf oder über der Entscheidungsfläche liegt. Da die Funktion 2.4 meistens nicht eindeutig lösbar ist, siehe Abbildung 2.7

(links), muss die optimale Hyperebene bestimmt werden die beide Klassen trennen kann. Um die optimale Hyperebene zu bestimmen wird die kürzeste Distanz d_+ und d_- von einem positiven und einem negativen Trainingelements zu der Hyperebene bestimmt. Die optimale Hyperebene ist dann die mit dem größten Abstand der Elemente zur Hyperebene. Die Bestimmung über den Abstand der Elemente zur Hyperebene wird auch *Maximum marigin algorithm* bezeichnet. Sobald man die Hyperebenen bestimmt hat kann man damit die optimale Hyperebene auswählen indem der Abstand zu den Elementen maximiert wird [10]. Da in fast allen praktischen Anwendungsfällen die meisten Daten nicht linear trennbar sind, siehe Abbildung 2.8, kann keine Hyperebene bestimmt werden. Damit ist eine lineare Trennung durch eine Hyperebene nicht möglich [12]. Um doch im praktischen Fall eine Hyperebene zu finden verwendet man den Kernel-Trick.

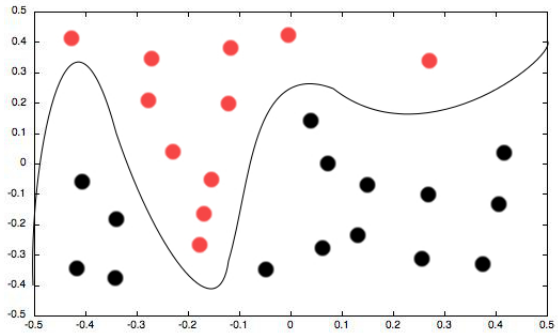

Abbildung 2.8: Nicht linear trennbare Daten

2.4.1 Kernel-Trick

Daten die in der Eingangsdimension nicht linear trennbar sind können in eine höhere Dimension projeziert werden, wodurch diese in einem Raum mit einer bestimmten Dimension linear trennbar werden. Jedes Element der Menge δ kann durch eine geeignete Funktion Φ in eine höhere Dimension abgebildet werden [10].

$$\Phi : \mathbb{R}^n \to \mathbb{R}^m : n, m \in \mathbb{N}; n < m \tag{2.13}$$

Die Funktion Φ wird auch Kernelfunktion genannt. In diesem höherdimensionalen Raum wird die Hyperebene bestimmt, die beide Klassen trennen kann. Bei der Rücktransformation der höherdimensionalen Daten in die Eingangsdimension wird die lineare Trennebene in der höheren Dimension zu einer komplexen Trennfläche im niedrigdimensionalen. Um das zu vermeiden werden Kernelfunktionen eingesetzt. Kernelfunktionen können in der höheren Dimension die Trennebene beschreiben und bleiben bei der Rücktransformation gutartig [25].

Kapitel 3

Konzept

In diesem Kapitel wird ausgehend von der Systemvision 3.1 die Anforderungen im Abschnitt 3.2 für das geplante System ermittelt. Dabei werden modulare Anforderungen ermittelt und funktionale und nicht-funktionale Anforderungen unterschieden. Anschließend wird das Konzept für die Implementierung erläutert. Im Abschnitt 3.4 wird die Aufnahme von Bildern über eine Webcam vorgestellt. Der Abschnitt 3.5 befasst sich mit dem Konzept der Gesichtsdetektierung und dessen Verfolgung im Aufnahmeverlauf. Der Abschnitt 3.6 erläutert das Konzept der Gesichtserkennung und beinhaltet die Ausrichtung von Gesichtern, die Extraktion von Merkmalen und den Abgleich mit bereits bekannten Merkmalen. Abschnitt 3.7 stellt die Komponente der Geschlechtserkennung vor. Im letzten Abschnitt 3.8 wird das geplante Webinterface entworfen.

3.1 Systemvision

Das Ziel der Bachelor Thesis ist die Entwicklung einer Gesichstserkennenungssoftware die in der Lage ist Personen eindeutig zu Identifizieren. Außerdem soll die Software in der Lage sein Geschlechter anhand von Gesichtsmerkmalen zu unterscheiden. Zeiten an denen eine Person erkannt wurde können benutzt werden um statistische Daten zu erheben. Verschiedene Trackingverfahren über Bluetooth oder WLAN können einen großen Teil der Personen nicht erfassen, da nicht alle Personen ein Bluetooth- oder WLAN fähiges mobiles Endgerät besitzen, bzw. die Schnittstellen nicht dauernd aktiv sind. Durch die biometrische Erfassung können auch Personenströme erfasst werden die nicht auf konventionelle Weise erfasst werden können. Die Software soll für jede erkannte Person ein Profil in einer Datenbank anlegen. Das Profil enthält alle Zeiten an denen die Person gesehen wurde und erweitert das Profil wenn die Person gesehen wurde. Die Visualisierung der gesammelten Daten geschieht über ein Webinterface. Das Webinterface bietet eine Schnittstelle um die gesammelten Daten zu Verwalten und gegebenenfalls zu korrigieren falls Fehlklassifikationen stattgefunden haben oder ein Profil mehrmals angelegt wurde.

3.2 Anforderungen

In den folgenden Listen werden Anforderungen an das geplante System festgelegt. Es wird zwischen funktionalen Anforderungen sowie nichtfunktionalen Anforderungen unterschieden. Die Anforderungen werden in Muss- , Soll- und Kann-Anforderungen aufgeteilt. Muss-Anforderungen müssen zwingend erfüllt werden. Soll-Anforderungen sollen erfüllt werden, sofern keine schwerwiegenden Argumente dagegen sprechen und Kann-Anforderungen können erfüllt werden, sofern es mit den gegebenen Ressourcen machbar ist.

3.2.1 Funktionale Anforderungen

Anforderungen an das Gesamtsystem

1. Es müssen neue Datensätze von Personen hinzugefügt werden können.
2. Bestehende Datensätze müssen erweitert werden können.
3. Das Webinterface und die Gesichtserkennung müssen auf der selben Datenbank arbeiten.
4. Das System kann eine bestimmte Person suchen lassen.

Anforderungen an die Aufnahme

1. Bilder müssen über eine Webcam aufgenommen werden.
2. Bilder können über die Microsoft Kinect aufgenommen werden.
3. Die Auflösung der aufgenommenen Bilder muss variabel sein.

Anforderungen an die Gesichtsdetektierung

1. Gesichter müssen auf Aufnahmen detektiert werden.
2. Das suchen von Gesichtern muss in Echtzeit passieren.
3. Die Gesichtsdetektierung muss Gesichter in einer Mindestentfernung von 1 Meter erkennen.
4. Gesichter müssen mit dem Viola-Jones Algorithmus erkannt werden.

Anforderungen an den Gesichtstracker

1. Gesichter müssen, sobald erkannt, verfolgt werden ohne das komplette Bild abzusuchen.
2. Die Gesichtsverfolgung soll bei schnellen Bewegungen hinterherkommen.
3. Es muss unterschieden werden ob die Person bereits verfolgt wird.

Anforderungen an die Gesichtserkennung

1. Wichtige Merkmale müssen extrahiert werden.
2. Extrahierte Merkmale müssen vergleichbar sein.
3. Merkmale müssen mit dem SURF-Algorithmus erkannt werden.
4. Eine unbekannte Person muss im System in einem neuen Datensatz erfasst werden.
5. Eine bekannte Person muss identifiziert und deren Datensatz erweitert werden.

Anforderungen an die Geschlechtserkennung

1. Das Geschlecht einer Person soll anhand der aufgenommenen Bilder klassifiziert werden.
2. Das Geschlecht muss mit Hilfe einer SVM ermittelt werden.
3. Die SVM kann mit den gesammelten Daten trainiert werden.
4. Der benötigte Klassifikator einer SVM muss austauschbar sein.

Anforderungen an das Webinterface

1. Das Webinterface muss alle gesammelten Daten anzeigen.
2. Jeder Datensatz muss einzeln abrufbar sein.
3. Das Webinterface muss eine Möglichkeit haben Datensätze manuell zu editieren.
4. Das Webinterface kann die Möglichkeit haben Personen manuell anzulegen.
5. Das Webinterface kann die Möglichkeit haben Personen als gesucht zu markieren.
6. Es soll ein Login System geben.

3.2.2 Nichtfunktionale Anfoderungen

Technische Anforderungen

1. C++ muss zur Implementation der Gesichtserkennung verwendet werden.
2. PHP kann zur Implementation des Webinterface verwendet werden.

3. Bilder müssen über eine Webcam aufgenommen werden.
4. Eine Datenbank muss zur Persistierung der Daten genutzt werden.
5. MySQL Datenbank kann zur Persistierung genutzt werden.
6. Das Boost C++ Framework soll für grundlegende Operationen genutzt werden.
7. Das OpenCV Framework soll für grundlegende Bildoperationen genutzt werden.
8. Das OpenBR Framework soll für grundlegende Analyseoperationen genutzt werden.

Anforderungen an die Benutzungsschnittstelle

1. Die Anwendung muss mindestens über ein Konsoleninterface verfügen.
2. Variable Parameter müssen in einer Konfigurationsdatei stehen.
3. Das Webinterface muss über jeden Browser zugänglich sein.
4. Das Webinterface muss von einem Apache Webserver ausgeliefert werden.

Anforderungen an den Entwicklungsprozess

1. Der Quellcode muss sinnvoll und verständlich dokumentiert sein.
2. Die Methoden können mit einem Test Framework getestet werden.
3. Eclipse kann als Entwicklungsumgebung verwendet werden.

3.3 Systemarchitektur

Das System soll Komponentenbasiert aufgebaut sein. Die Abbildung 3.1 skizziert die einzelnen Systembausteine. Die Architektur besteht aus 6 Komponenten und einer Datenbank. Die erste Komponente *Kameraaufnahme* kümmert sich um die Aufnahme von Bilddaten. Dazu gehört die Verbindung zur Kamera, die Aufnahme von Bildern und das umwandeln der aufgenommenen Bilder. Das Umwandeln behandelt die Abbildung von Bildsignalen auf Datenstrukturen. Diese Komponente wird im Abschnitt 3.4 behandelt und weiter ausgeführt. Das aufgenommene Bild wird weitergereicht an die nächste Komponente der *Gesichtsdetektierung*. Diese Komponente ist zuständig dafür das auf dem Bild Gesichter gefunden werden. Sobald ein Gesicht gefunden wurde wird es aus dem Bild ausgeschnitten und vorbereitet es an die nächste Komponente zu geben. Das vorbereiten beinhaltet das Gesicht auszurichten und

in einer Graustufe darzustellen. Sobald das Gesicht erfasst wurde benötigt der *Tracker* die Position vom erfassten Gesicht so wie die *Gesichtserkennung* das ausgerichtete Gesicht. An diesem Punkt arbeitet der *Tracker* und die *Gesichtserkennung* parallel. Der Tracker verfolgt das vom *Gesichtsdetektor* erkannte Gesicht anhand von Schnittmengen, diese Komponente wird Abschnitt 3.5 weiter erläutert. Die *Gesichtserkennung* extrahiert wichtige Merkmale vom Gesicht und gibt diese extrahierten Merkmale zur Komponente *Geschlechtserkennung* anschließend vergleicht die *Gesichtserkennung* die Merkmale gegen eine Datenbank mit bereits bekannten Gesichtern. Die *Datenbank* beinhaltet alle bekannten Gesichter sowie Zeiten an denen diese gesehen wurden. Sobald ein Gesicht klassifizert und gegen die Datenbank verglichen wurde wird entschieden ob ein neuer Datensatz angelegt wird oder ein bereits bestehender erweitert wird. Das *Webinterface* arbeitet ebenfalls auf der *Datenbank* und soll eine möglichst einfache Verwaltung von anfallenden Datenmengen bieten.

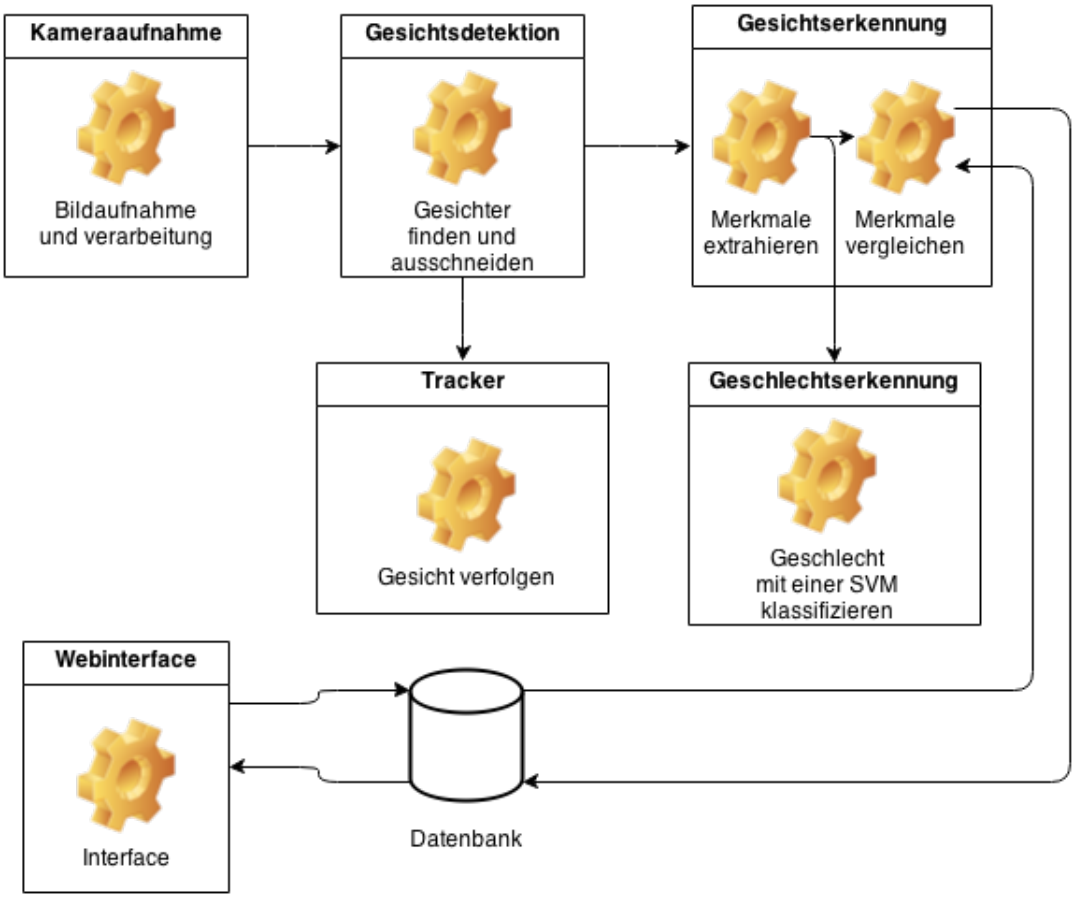

Abbildung 3.1: Systemarchitektur im Überblick

3.4 Bildakquisition

Die Bildaufnahme soll über eine Kamera, im praktischen Anwendungsfall durch eine Webcam, passieren. Dazu wird der Ablauf der Bildaufnahme in Abbildung 3.2 skizziert. Die Umgebung wird mittels einer Kamera erfasst. Das momentane Bild wird als ein Matrix Objekt repräsentiert. Das aktuelle Bild kann nun, in seiner Matrixrepräsentation, von anderen Komponenten angefordert werden. Die Kamera ist in der Lage immer nur ein Bild zu liefern. Das angeforderte Bild entspricht dem zuletzt aufgenommen Bild.

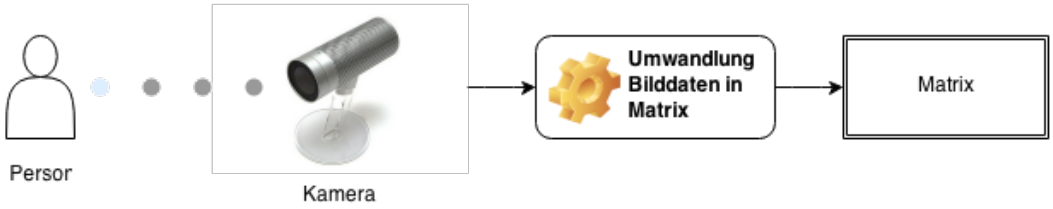

Abbildung 3.2: Ablauf der Bildaufnahme

Da die Aufnahme unabhängig von der verwendeten Hardware sein soll, wird die Klasse Capture eingeführt. Diese beinhaltet eine Methode die ein Bild von der der Kamera anfordert und dieses auf eine Matrize abbildet. Die aufgenommenen Daten werden als eine Matrize zurückgegeben. Das Bild liegt dann in einer vorher definierten Auflösung vor. Die Auflösung entspricht der Zeilen und Spalten der Matrix. Die Abbildung 3.3 zeigt in UML Notation die Abhängigkeit dieser Klasse von der Auflösung der angeforderten Daten. Es ist immer nur das letzte angeforderte Bild verfügbar. Wenn ein neues Bild angefordert wird, dann muss sich in anderen Modulen um die Persistierung der vorherigen Bilder gekümmert werden falls dies gefordert wird.

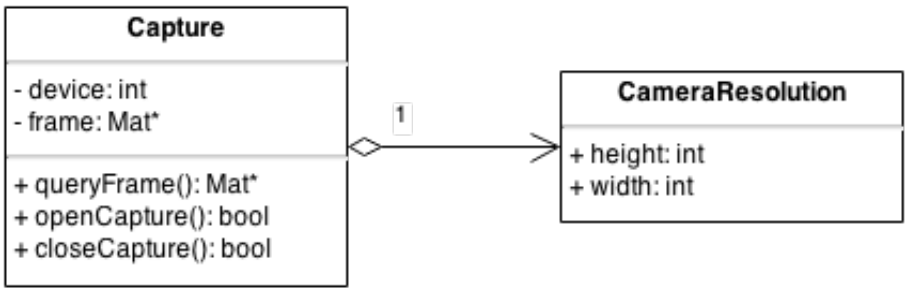

Abbildung 3.3: UML Diagramm der Capture Komponente

Sollte es spezifische Behandlungen zur Aufnahme von Bildern benötigen so dient die Klasse Capture als Schnittstelle. Kamerahardware die mehrere Sensoren verwendetet, kann so gezielt angefragt werden um die notwendigen Daten zu erhalten. Als Beispiel sei hier die Microsoft Kinect zu nennen. Dessen Behandlung zur Bildaufnahme unterscheidet sich deutlich von der einer Webcam,

da in der Kinect zusätzlich Tiefensensoren verbaut sind und die Kommunikation mit der Kamera anders passiert als in einer Webcam.

3.5 Gesichtsdetektierung und Tracking

Die aufgenommenen Bilder werden nach Gesichtsmustern abgesucht. Um nach Gesichtsmustern zu suchen wird das in Abschnitt 2.2 vorgestellte Verfahren verwendet. Eine Cascade Funktion wird mit möglichst vielen positiven und negativen Bildern trainiert. Positive Bilder enthalten Gesichter und negative Bilder enthalten keine Gesichter. Bei dem Verfahren aus Abschnitt 2.2 wird das komplette aufgenommene Bild abgesucht. Dies ist äußert ineffizient da beim nächsten ankommenden Bild wieder das komplette Bild abgesucht werden muss.

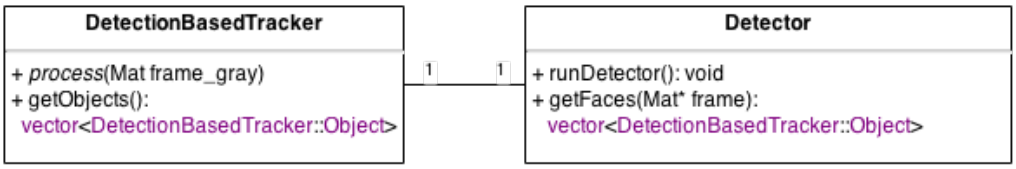

Abbildung 3.4: UML Diagramm der Gesichtsdetektierung und dem Tracker

Ein Gesicht wird initial durch die Gesichtsdetektierung erkannt und somit sind die x und y Positionen sowie die Breite und die Höhe des Gesichtes im Bild bekannt. Das daraus resultierende Rechteck kann genutzt werden um im nächsten Bild im näheren Umfeld des Rechtecks zu suchen. Jedes Gesicht kann persistent durch mehrere Bilder gehalten werden und bekommt so einen Bezug zum vorherigen Gesicht. Dieses Verhalten kann genutzt werden um den Zeitraum zu protokollieren an dem eine Person gesehen wurde. Der Tracker vergibt für jedes erfasste Gesichte eine variable Lebensspanne. Ein initial erfasstes Gesicht beginnt mit einer Lebenspanne von 0. Im nächsten Bild wird die Lebensspanne um eins erhöht wenn im näheren Bereich kein Gesicht detektiert werden konnte. Sobald das variable Maximum erreicht wurde wird das verfolgte Gesicht verworfen und wieder das komplette Bild abgesucht. Die Abbildung 3.5 stellt den oben erläuterten Ablauf bildlich da. Der Trackingalgorithmus baut für jedes individuelle Gesicht ein Trackingobjekt auf. Das Objekt existiert solange bis die Lebensspanne einen gewissen Wert überschreitet.

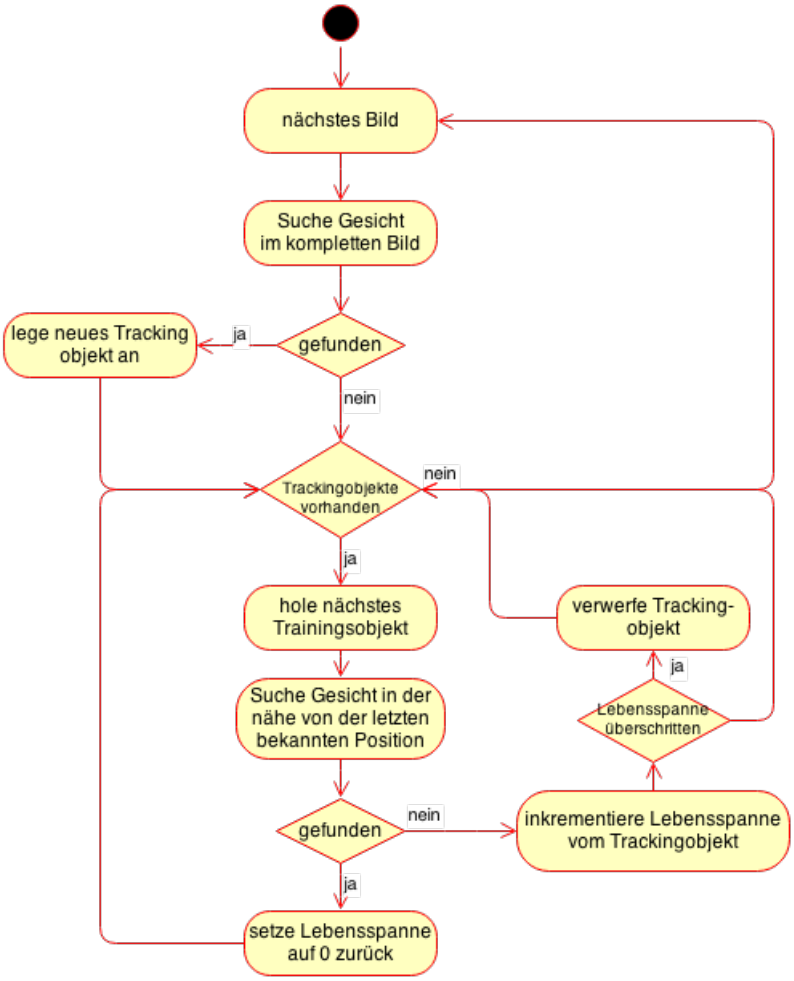

Abbildung 3.5: Ablauf des vorgegebenen Trackingalgorithmus

3.6 Gesichtserkennung

Alle erkannten Gesichter werden zuerst ausgerichtet, danach werden repräsentative Merkmale extrahiert und anschließend wird bestimmt ob der Datensatz bereits bekannt ist oder es sich um eine neue Person handelt. Der Abschnitt 3.6.1 beschreibt wie die erkannten Gesichter ausgerichtet werden. Im nächsten Abschnitt 3.6.2 wird beschrieben wie die wichtigen Merkmal extrahiert werden. Zum Schluss wird in Abschnitt 3.6.3 erläutert wie ein Datensatz identifiziert werden kann.

3.6.1 Gesichter ausrichten

Bevor die Merkmale mit dem SURF-Algorithmus extrahiert werden, müssen die Gesichter ausgerichtet werden. Die Ausrichtung geschieht in folgenden Schritten. Der Bereich in dem das Gesicht erfasst wurde, wird wenn möglich so

gedreht das die Augen auf einer horizontalen Linie sind. Abbildung 3.6 zeigt die Schritte die passieren wenn das Gesicht gedreht wird. In dem detektierten Bild werden die Augen erfasst. Anschließend kann der Augenmittelpunkt errechnet werden. Der Drehwinkel kann nun mittels einfacher Trigonometrie ausgerechnet werden. Zuerst wird der Quotient $\alpha = \frac{t_y}{\delta}$ ermittelt, wobei t_y der Höhenunterschied zwischen den Augenmittelpunkten und δ die Distanz zwischen den Augenmittelpunkten. Anschließend kann der Drehwinkel ω bestimmt werden, $\omega = asin(\alpha) * \frac{180}{\pi}$. Die Drehrichtung ist anschließend abhängig von den Augemittelpunkten. Sollte die Höhenposition vom rechten Augenmittelpunkt kleiner sein als die Höhenposition vom linken Augenmittelpunkt, dann wird im Uhrzeigersinn gedreht ansonsten gegen den Uhrzeiger sinn.

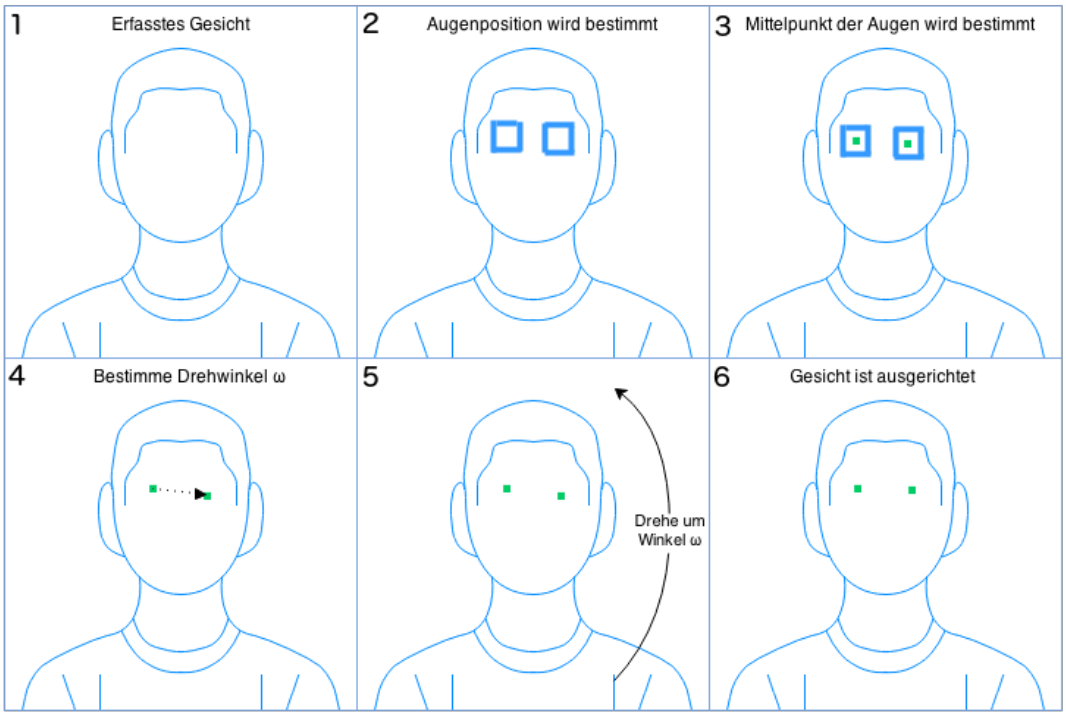

Abbildung 3.6: Ausrichtung vom Gesicht anhand der Augen

Sobald das Gesicht auf der horizontalen ausgerichtet wurde kann nun der Hintergrund entfernt werden. Die Abbildung 3.7 zeigt das der Bildausschnitt links und rechts (a und b), unten (c) und oben (d) geschrumpft wird. Viele unnötigen Informationen können damit schon vor der extraktion der Merkmale verworfen werden.

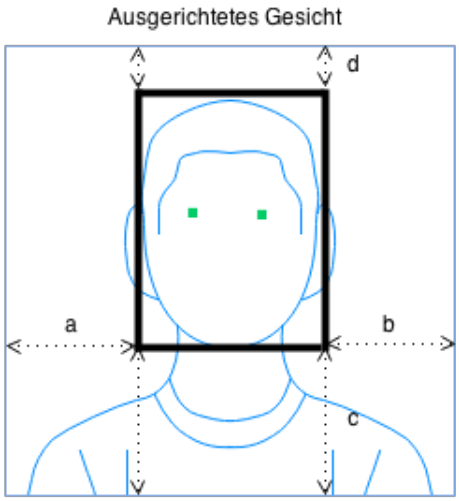

Abbildung 3.7: Ausschneiden vom Gesicht

Das ausgerichtete und ausgeschnittene Bild wird zur Merkmalsextraktion verwendet.

3.6.2 Gesichtsmerkmale extrahieren

Um die Merkmale zu extrahieren wird der SURF Algorithmus verwendet. Der SURF Algorithmus wurde bereits in Abschnitt 2.3 vorgestellt. Das ausgerichtete und ausgeschnittene Gesicht wird verwendet um die Merkmale zu extrahieren. Im ersten Schritt werden wichtige Schlüsselpunkte im Gesicht bestimmt. Anhand dieser Schlüsselpunkte können die repräsentativsten Merkmale berechnet werden.

3.6.3 Gesichtsmerkmale vergleichen

Um zwei Bilder oder Gesichter zu vergleichen müssen wir die Distanz der beiden zu vergleichenden Bildern ermitteln. Der SURF-Algorithmus wird verwendet um Merkmale zu extrahieren und wurde bereits in Abschnitt 2.3 vorgestellt. Der SURF-Algorithmus versucht die wichtigsten Schlüsselpunkte im Bild zu finden. Die wichtigsten Schlüsselpunkte sind die Punkte an denen die Varianz der Pixel maximal ist. An diesen Schlüsselpunkten werden Vektoren mit 64 Einträgen konstruiert um die Merkmale zu repräsentieren. In einem Bild wird es so viele Merkmale geben wie Schlüsselpunkte. Nachdem die Merkmale durch die 64-wertigen Vektoren konstruiert wurden, können diese genutzt werden um eine Distanz zu einem anderen Bild zu errechnen.

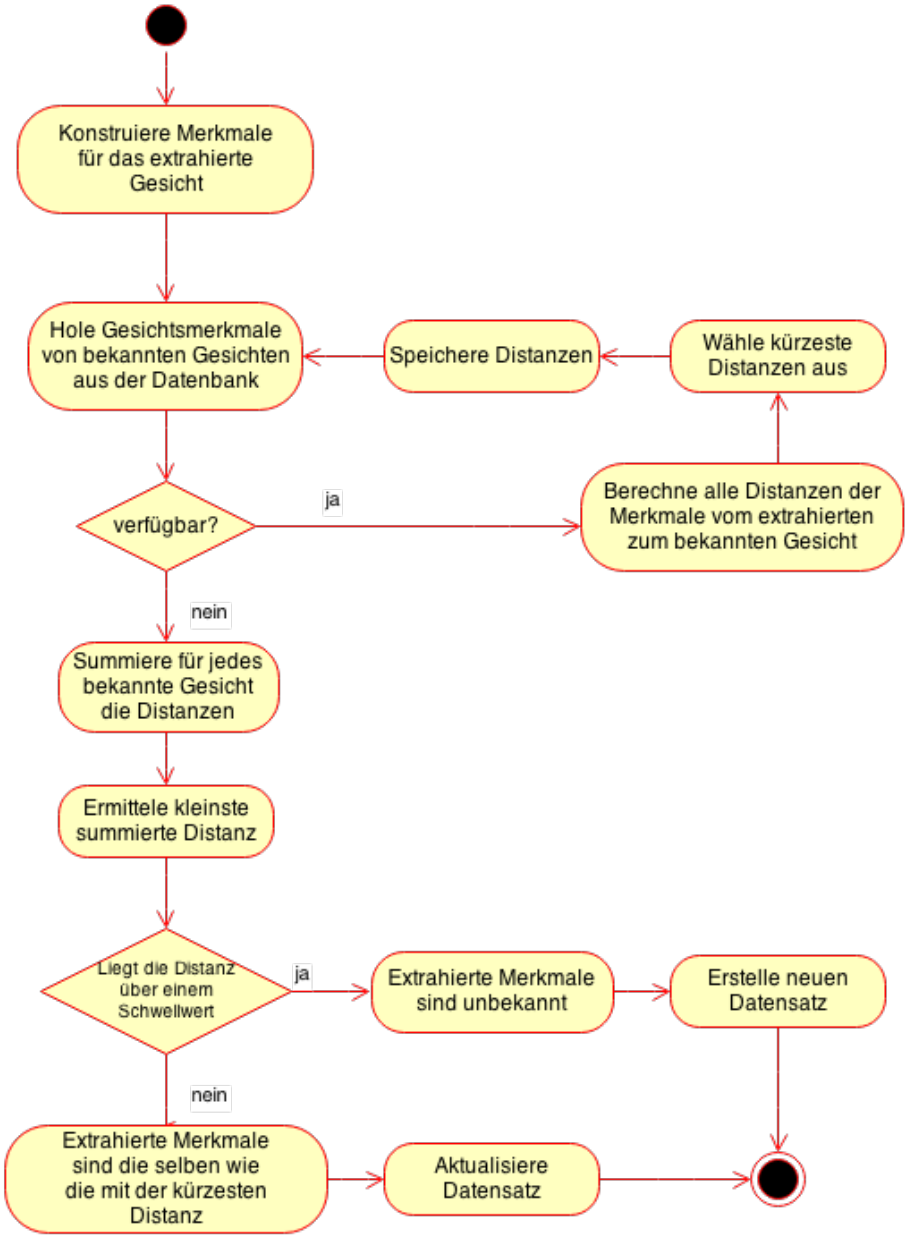

Abbildung 3.8: Gesichtsmerkmale vergleichen

Die Abbildung 3.8 stellt visuell den Ablauf vor wie die Merkmale der Gesichter verglichen werden sollen.

3.7 Geschlechtserkennung

Die Klassifizierung von Geschlechtern wird durch eine Support Vector Machine durchgeführt. SVMs wurden bereits in Abschnitt 2.4 eingeführt und vorgestellt. Der Einsatz von Support Vector Machines für die klassifizierung von Geschlechtern wird von Moghaddam und Yang in Ihrer Ausarbeitung vorgestellt (vgl. [17]).

Da Bilder von Gesichtern im allgemeinen Fall im niedrig dimensionalen Raum meistens nicht trennbar sind, helfen die Kernel Funktionen eine Trennung zu erreichen. Um entscheiden zu können ob eine Person männlich oder weiblich ist muss man vorher die repräsentativen Merkmale beider Geschlechter lernen. Anschließend muss mit einer passenden Kernel Funktion eine Trennlinie zwischen den repräsentativen Merkmalen gefunden werden. Sobald diese Trennlinie gefunden wurde kann diese verwendet werden um zukünftige Gesichter effizient klassifzieren zu können. Dazu werden die repräsentativen Merkmale extrahiert und in den Raum abgebildet. Je nach Position der Merkmale im Raum kann entschieden werden ob ein Gesicht männlich oder weiblich ist. Die Klasse 3.9 zeigt die Konzeptklasse zur Geschlechtserkennung. Die Methode getGender nimmt als Parameter ein Bild und liefert für männlich +1 und für weiblich -1 zurück.

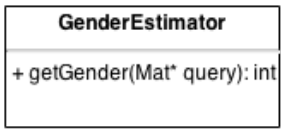

Abbildung 3.9: Klasse Genderestimator

Die Klasse $GenderEstimator$ benutzt eine bereits zuvor trainierte SVM. Wie bereits im Grundlagen Kapitel beschrieben besteht das Training einer SVM daraus die passende Trennline zwischen den Eingabedaten zu bestimmen. Der konzeptionelle Ablauf der Klassifizierung wird im nächsten Abschnitt vorgestellt.

3.7.1 Klassifizierung von Geschlechtern

Extrahierte Gesichter werden nach einem bestimmten Ablauf nach Geschlechtern klassifiziert. Die Abbildung 3.10 zeigt den schematischen Ablauf. Zuerst werden aus dem erfassten und ausgerichteten Gesicht die Merkmale extrahiert. Die Merkmale müssen nicht zwangsläuft durch SURF bestimmt werden, sondern können ebenfalls durch andere Verfahren ermittelt werden, beispielhaft sei hier die Principal Component Analysis (PCA) genannt. Die extrahierten Merkmale können in einen n-dimensionalen Raum abgebildet werden. Mit einem passenden Kernel kann das Eingabedatum in die gewünschte Dimension transformiert werden. Die zuvor trainierte SVM kann eingesetzt werden um bestimmen zu können zu welcher Klasse das Eingabedatum gehört. Sollte das Eingabedatum zur Klasse -1 gehören so wird das Gesicht als weiblich klassifziert, wenn das Gesicht zur Klasse +1 gehört dann wird das Gesicht als männlich klassifziert.

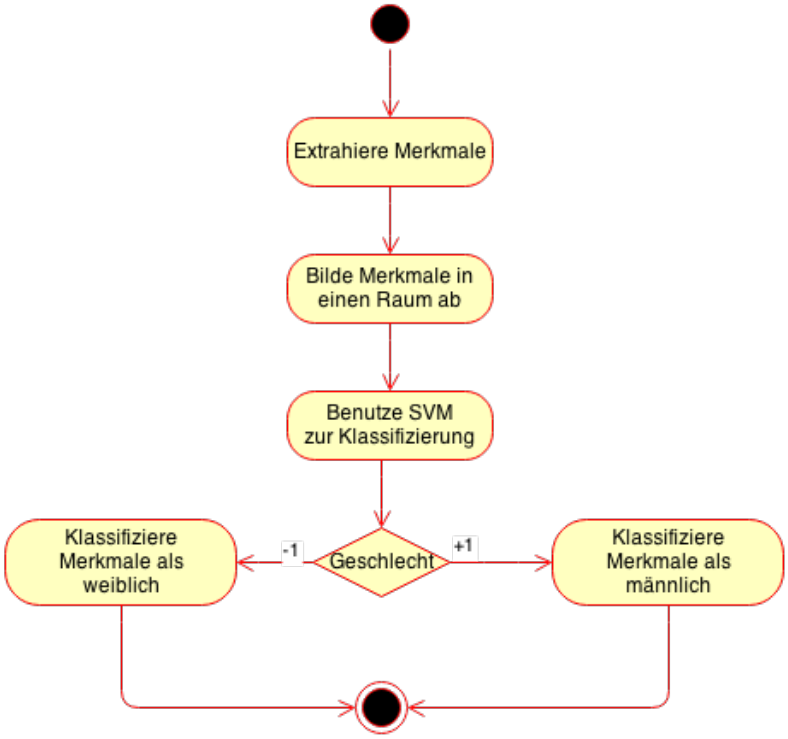

Abbildung 3.10: Ablauf der Klassifizierung von Geschlechtern

3.8 Entwurf eines Web Interface zur Verwaltung von gesammelten Datensätzen

Das geplante Web Interface soll in PHP entwickelt werden und folgt dem MVC - Muster. Das MVC-Muster steht für Model, View und Controller und unterteilt das System in Datenstrukturen, die Präsentation dieser und die Aktionen die durch einen Nutzer mit den Datenstrukturen entstehen. Das Muster zur unterteilung in Objektdomäne, Präsentation und Aktionen wurde erstmals von Steve Burbeck (vgl. [4]) vorgestellt und von Gamma, Helm, Johnson und Vlissides als Muster für die Objektorientierte Programmierung in Ihrem Buch *Design Patterns* beschrieben (vgl. [9]). Die Abbildung 3.11 beschreibt die Anordnung der Komponenten. Das Model verwaltet die Daten und das Verhalten der Applikation, antwortet auf Anfragen über den eigenen Status und antwortet dem Controller mit Instruktionen was sich geändert hat. Die View verwaltet die Darstellung der Daten und die Interaktion mit dem Benutzer. Jeder Interaktion wird dem Controller mitgeteilt. Der Controller interpretiert die Eingaben durch die View und informiert das Model über die Eingaben und die View über Änderungen im Model. Die View hat nur indirekten Einfluss über das Model so dass eine Änderung nur über den Controller möglich ist. Die View und der Controller sind beide vom Model abhängig, das Model selber hat keine Abhängigkeit

von den anderen Komponenten. Die unabhängigkeit des Models erlaubt das einfache Austauschen und hinzufügen von Interfaces und Applikationslogiken.

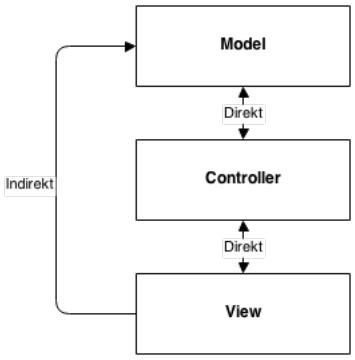

Abbildung 3.11: Das MVC Muster.
Quelle: [16]

Die verwaltung der gesammelten Datensätze soll über ein Webinterface passieren. Die Abbildung 3.12 zeigt die Hauptseite des geplanten Webinterfaces. Die Startseite soll die Möglichkeit bieten sich einzuloggen oder neu zu registrieren. Beide Funktionen können über die Navigationsleiste erreicht werden. Geplant ist ebenfalls eine Hilfefunktion die über die Grundlagen der Navigation und der Bedienung im Webinterface informiert. Auf der Startseite soll es möglich sein über den Button *Learn more* an die Projektdaten zu gelangen.

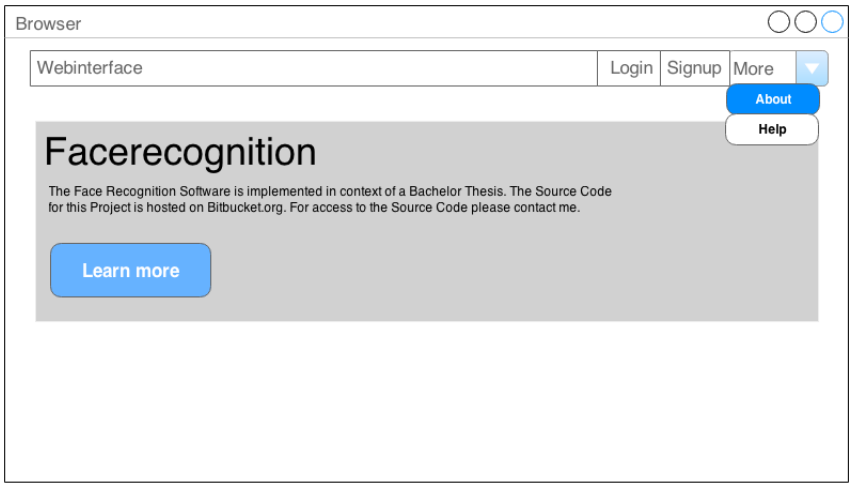

Abbildung 3.12: Mockup der Startseite

Nachdem man sich erfolgreich registriert und eingeloggt hat sieht man sein Dashboard das alle angefallenen Daten übersichtlich auflistet. Die Abbildung 3.13 zeigt wie das Dashboard aussehen soll. Die oberen Elemente im Dashboard zeigen eine grobe Auflistung aller im System verfügbaren Datensätze,

wieviele Bilder insgesamt aufgezeichnet wurden und wieviele dieser Datensätze männlich oder weiblich sind. Unter der groben Auflistung sieht man die letzten erkannten Gesichter.

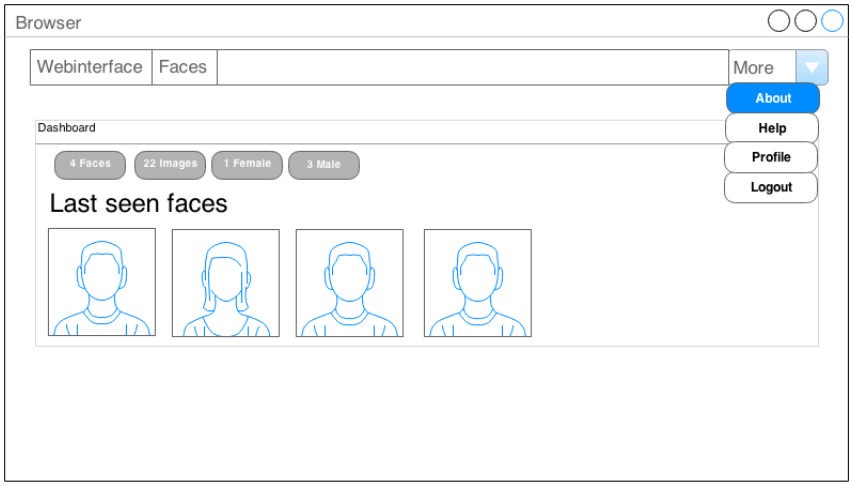

Abbildung 3.13: Mockup vom Dashboard

Jeder Datensatz besitzt eine Detailansicht. Diese Ansicht beinhaltet initial das geschätzte Geschlecht, alle aufgenommen Bilder und eine graphische Darstellung von den gesehenen Intervallen.

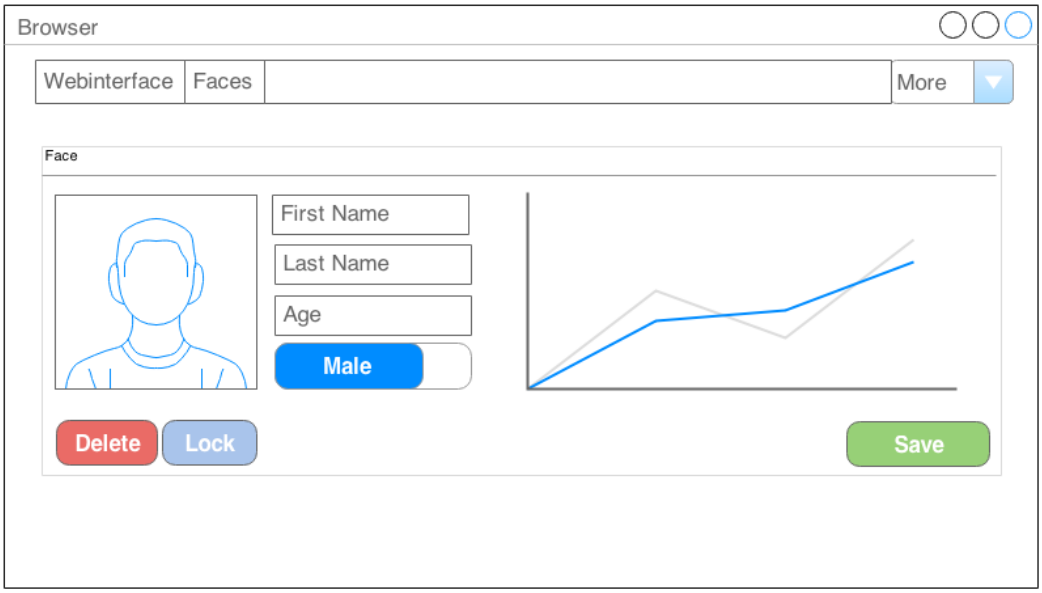

Abbildung 3.14: Mockup einer Detailseite eines Datensatzes

Der Datensatz lässt sich editieren und abspeichern. Daten wie der Vorname, Nachname, Alter und das Geschlecht lassen sich editieren. Zusätzlich lässt sich der Datensatz sperren. Die Sperrung hat zur Folge das der Datensatz nicht mehr editierbar oder löschbar ist.

Kapitel 4

Implementation

Das Kapitel Implementation befasst sich mit dem im Konzept beschriebenen System. Im Abschnitt 4.1 werden die benutzten Frameworks aufgelistet und vorgestellt. Der nächste Abschnitt 4.2 befasst sich mit der konkreten Aufnahme von Bildern. Im Abschnitt 4.3 wird beschrieben wie das Konzept zur Aufbereitung von Bildern umgesetzt wurde. Der nächste Abschnitt 4.4 beschreibt die Implementation der Objektdetektion und der Detektion von Gesichtern. Anschließend wird im Abschnitt 4.5 erläutert wie genau das detektierte Gesicht verfolgt wird. Der Abschnitt 4.6 stellt das extrahieren von Gesichtsmerkmalen und den Vergleich dieser zur Gesichtserkennung vor. Im vorletzten Abschnitt 4.7 wird die Implementation der Geschlechtserkennung erläutert. Der letzte Abschnitt 4.8 befasst sich mit der Implementation des Webinterfaces.

4.1 Verwendete Frameworks

Im Rahmen der Implementation wurden diverse Frameworks verwendet. Das OpenCV, ObenBR und Boost Framework wurden verwendet und werden in den folgenden Unterabschnitten weiter erläutert.

OpenCV

OpenCV, kurz für Open Computer Vision, ist eine Bibliothek und wurde entworfen um rechenaufwändige Aufgaben zu bewältigen. Die Bibliothek wurde von Intel entwickelt und steht unter der BSD Lizenz, was die Bibliothek frei einsetzbar macht. OpenCV ist für verschiedene Plattformen verfügbar. Derzeit liegt OpenCV in der Version 2.4.8 vor. Der Fokus der Bibliothek liegt auf der Verarbeitung von Daten in Echtzeit. Besonders im Bereich des maschinellen Sehens wird OpenCV häufig eingesetzt [11].

OpenBR

OpenBR, kurz für Open Biometric Recognition, ist ein Framework und dient als Middleware für Softwaresysteme die biometrische Identifikationsverfahren benutzen. Das Projekt ist Open Source und steht unter Apache 2 Lizenz und ist für mehrere Plattformen verfügbar. Derzeit liegt OpenBR in der Version 0.4.1 vor. Das Framework baut auf dem OpenCV und QT Framework auf und wird zur Evaluation von bestehenden biometrischen Systemen, verbesserung von bestehenden Algorithmen und von schnellem ausprobieren von neuen biometrischen Algorithmen eingesetzt [13].

Boost Framework

Das Boost Framework bietet zahlreiche Unterstützungsmöglichkeiten für den Programmierer und besteht aus über achtzig individuellen Bibliotheken, für die Programmiersprache C++. Die meisten Bibliotheken stehen unter der eigenen Boost Software Lizenz und erlauben den freien Einsatz. Derzeit liegt das Boost Framework in der Version 1.55 vor. Zahlreiche Aufgaben wie das Threading, die Netzwerkkommunikation, IO-Operationen oder Unit Tests können vom Boost Framework übernommen werden [24].

4.2 Kameraaufnahme

Um Bilder von der Kamera zu liefern, werden die beiden, in Abschnitt 3.4, vorgestellten Klassen *Capture* und *CameraResolution* verwendet. Der Algorithmus 1 beschreibt wie die Bilder aquiriert werden. Das *Capture*-Objekt wird zuerst erstellt. Anschließend wird die geforderte Auflösung initialisiert und an das *Capture*-Objekt gebunden. Danach kann das *Capture*-Objekt sich mit der Kamera verbinden. Das verbinden mit der Kamera passiert in der Methode *openCapture* des *Capture*-Objekts. Die Methode probiert jede bekannte und verfügbare Möglichkeit aus sich mit der Kamera zu verbinden. Wenn eine Verbindung hergestellt werden konnte tritt das Programm in eine Schleife ein die in jedem Zyklus ein neues Bild von der Kamera anfordert. Bei jedem Schleifendurchlauf wird geprüft ob die Kamera immer noch verbunden ist und Bilder liefern könnte. Der erste Aufruf in der Schleife holt sich ein konkretes Bild von der Kamera und wandelt es in ein Datenformat, das von OpenCV vorgegeben ist, um. Dieses Frame kann an die verschiedenen Komponenten zur Weiterverarbeitung gegeben werden.

4.3 Bildaufbereitung

Das aufgenommene Bild muss, bevor es an den Detektor für Gesichter weitergereicht wird, aufbereitet werden. Die Aufbereitung beinhaltet die Umwandlung

Algorithm 1 Bildaufnahme von einer Webcam

```
1:  procedure MAIN(argc, argv[])
2:      ...
3:      cap ← Capture                                  ▷ Initialisiere Capture Objekt
4:      resolution ← CameraResolution                  ▷ Init. CameraResolution Objekt
5:      cap.Resolution ← resolution
6:      if cap.openCapture() then
7:          while cap.isOpen() do
8:              frame ← cap.queryFrame()
9:              ...
10:             inputKey ← getKeyboardInput()          ▷ Hole Input von User
11:             if inputKey > 0 then
12:                 cap.close()
13:             end if
14:         end while
15:     end if
16:     ...
17:     return 0                                       ▷ Schließe Programm
18: end procedure
```

in Graustufen und die Normalisierung durch ein Histogram. Das Verfahren zur Umwandlung von Bilder in eine Graustufe wurde bereits in Abschnitt 2.1.1 erläutert. Die Umwandlung in eine Graustufe und die Normalisierung des Histogramms wurde mit dem OpenCV Framework realisiert. Der Quellcode 6.31 zeigt wie einfach sich die Umwandlung und die Normalisierung eines aufgenommenen Bildes durch das OpenCV Framework umsetzen lässt. Die Methode befindet sich in der Klasse *Detector*, die bereits im Abschnitt 3.5 vorgestellt wurde. Der Aufruf *cvtColor(...)* wandelt das ursprüngliche Bild in einen anderen Farbraum um. In dem Fall wird das aufgenommene Bild in eine Graustufe konvertiert. Die Methode *equalizeHist(...)* normalisiert das umgewandelte Bild anhand des Histogramms. Die Normalisierung wird eingesetzt um die Lichtunterschiede auf dem Graustufenbild zu komprimieren.

Listing 4.1: Bildaufbereitung mit OpenCV

```
1 vector<Faces*> Detector::getFaces(Mat* frame){
2     Mat frame_gray;
3     cvtColor(*frame, *&frame_gray, CV_BGR2GRAY);
4     equalizeHist(*&frame_gray, *&frame_gray);
5     ...
6 }
```

4.4 Gesichtsdetektierung

Wie in Abschnitt 2.2 bereits eingeführt können Objekte auf Bildern, durch das von Viola und Jones vorgestellte Verfahren, detektiert werden. In der Implemen-

tation wurde der Detektor bereits mit einer großen Anzahl von positiven Bildern und negativen Bildern trainiert und weiß wie Gesichter zu erkennen sind. Das OpenCV Framework stellt mehrere trainierte Detektoren zur Verfügung. Diese Detektoren müssen nicht erneut trainiert werden und können direkt verwendet werden. Die Klasse *Detector* wurde bereits im Abschnitt 3.5 konzeptionell vorgestellt und wird verwendet um den bereits trainierten Detektor einzusetzen um damit Gesichter auf Bildsequenzen erkennen zu können. Das detektieren von Gesichtern wird im Algorithmus 2 vorgestellt. Der Algorithmus 2 erläutert wie Gesichter durch das OpenCV Framework detektiert werden können, anhand der mitgelieferten trainierten Kaskaden. Die Routine *detectFaces* stellt exemplarisch die Detektion in einem Frame dar. Die Signatur der Routine nimmt als Parameter den Pfad zur Kaskade und das oben beschriebene *Capture* Objekt. Über das Capture Objekt wird ein Frame angefordert und das *CascadeClassifier* Objekt initialisiert. Das *CascadeClassifier* Objekt ist eine im OpenCV definierte Klasse und dient der einfachen Anbindung von trainierten Detektoren. Anschließend wird ein Vektor benötigt der die Rechtecke mit x und y Position beinhaltet. Der trainierte Detektor lässt sich über die *load*-Methode initialisieren. Anschließend werden über die *detectMultiScale* Methode alle trainierten Objekte detektiert. Da eine Gesichtssuche bei jedem neuen Bild ineffizient wäre kommt ein Tracker zum Einsatz. Die konkrete Implementation des Trackers wird im nächsten Abschnitt erläutert.

Algorithm 2 Gesichtsdetektierung mit OpenCV

```
1:  procedure DETECTFACES(pathToXML, capture)
2:      frame ← capture.queryFrame()
3:      CascadeClassifier cascade
4:      vector < Rect > faces
5:      if cascade.load(pathToXML) == FALSE then
6:          return                                  ▷ Beende Routine
7:      end if
8:      cascade.detectMultiScale(frame, faces, ...)
9:      return faces                          ▷ Liefere detektierte Gesichter
10: end procedure
```

4.5 Gesichtstracker

Durch den Einsatz eines Trackingalgorithmus lässt sich das Laufzeitverhalten positiv beeinflussen. Dadurch das ein Gesicht nicht erneut über der ganzen Fläche gesucht werden muss reduziert dies die benötigte Rechenleistung. Das OpenCV Framework bietet einen einfachen Tracker an, der anhand von Positionsdurchschnitten ein Gesicht erfassen und in der Bildsequenz verfolgen kann. In der vorgestellten Klasse *Detector* wird der Tracker eingebunden. Die Methode *getFaces* wird in der Methode 4.2 vorgestellt. Die Funktion nimmt als

Parameter ein Frame das durch das Capture Objekt geliefert wurde. Anschließend wird ein Vektor mit Elementen vom Typ *DetectionBasedTracker::Object* angelegt. Die Klasse im Namespace *DetectionBasedTracker* wird im OpenCV Framework definiert. Der Tracker versucht bekannte Gesichter zu verfolgen. Der Aufruf *process* des Tracker Objekts sucht nach bekannten Gesichtern und speichert diese dann im vorher definierten Vektor.

Listing 4.2: Einbindung des OpenCV Trackers

```
1  DetectionBasedTracker tracker;
2  ...
3  vector<DetectionBasedTracker::Object> Detector::getFaces(Mat* frame){
4
5      vector<DetectionBasedTracker::Object> objFaces;
6
7      this->tracker->process(frame);
8      this->tracker->getObjects(*&objFaces);
9
10     return objFaces;
11 }
```

Wie oben bereits beschrieben wird das interne Tracking durch Positionsdurchschnitte und Lebensspannen von erkannten Gesichtern realisiert. Der Algorithmus 3 skizziert den Ablauf der internen Methode *process*.

Algorithm 3 Process Methode

```
1: ...
2: vector < TrackedObjects > ← trackedObjects
3: ...
4: procedure PROCESS(frame)
5:     for each trackedObject in trackedObjects do
6:         lastPosition ← trackedObject.lastPosition
7:         detectInRegion(frame, lastPosition)
8:     end for
9: end procedure
```

Der Algorithmus 3 nimmt als Parameter das aufgenommene Frame. Zuvor wird für jedes erkannte Gesicht ein $TrackedObject$ angelegt. Das Objekt beinhaltet die letzte Position und die Lebensspanne des Objekts. Die Methode versucht für jedes bekannte TrackedObject das Gesicht in der Umgebung der letzten Position zu finden. Sollte das Gesicht gefunden werden wird die Lebensspanne des Objekts zurückgesetzt und die letzte bekannte Position aktualisiert. Im Falle das kein Gesicht im Umfeld der letzten Position erkannt wurde wird die Lebensspanne inkrementiert.

4.6 Gesichtserkennung

Um Gesichter erkennen zu können müssen diese vorher ausgerichtet werden. Die Ausrichtung dient der optimalen Erfassung von Gesichtern. Im Abschnitt 4.6.1 wird beschrieben wie die Ausrichtung von erfassten Gesichtern implementiert wurde. Nach dem erfassen und ausrichten können die Merkmale extrahiert

werden. Die Extraktion von Merkmalen geschieht mit dem SURF Algorithmus. Die Implementation der Extraktion von Merkmalen aus Gesichtern wird im Abschnitt 4.6.2 beschrieben. Sobald man repräsentative Merkmale vorliegen hat lassen sich diese mit einer Datenbank abgleichen. Das abgleichen und lernen von neuen Gesichtern wird in den Abschnitten 4.6.3 und 4.6.4 weiter ausgeführt.

4.6.1 Gesichter ausrichten

Wie im Abschnitt 3.6.1 beschrieben werden erkannte Gesichter anhand der Augen auf einer horizontalen Linie ausgerichtet. Im Algorithmus 4 wird beschrieben wie die Implementation umgesetzt wurde. Zuerst wird die Augenposition bestimmt. Die Bestimmung der Augenpositionen läuft genauso wie im Algorithmus 2. Das OpenCV Framework liefert trainierte Detektoren zur Detektion von Augen mit. Das erkannte Gesicht und die beiden erkannten Augen werden an die Methode *alignFace* übergeben. In der Methode wird das Gesicht um einen entsprechenden Winkel gedreht und anschließend so zurecht geschnitten das Hintergründe wegfallen. Zuerst werden die Mittelpunkte der Augenpositionen bestimmt. Der Algorithmus 5 beschreibt die Bestimmung der Augenmittelpunkte.

Algorithm 4 Algorithmus zum Gesichter ausrichten

procedure ALIGNFACE($face, leftEye, rightEye$)
  $leftEyeCenter \leftarrow getCenter(leftEye)$
  $rightEyeCenter \leftarrow getCenter(rightEye)$
  $rotationAngle \leftarrow getRotationAngle(leftEyeCenter, rightEyeCenter)$
  $rotationCenter \leftarrow Point2f(leftEyeCenter.x, leftEyeCenter.y)$
  $rotationMatrix \leftarrow getRotationMatrix2D(rotationCenter, rotationAngle)$
  $alignedFace \leftarrow Mat$
  $warpAffine(face, alignedFace, rotationMatrix)$
  $return\ cropFace(alignedFace, 10, 10)$
end procedure

Das Auge wird als ein Rechteck mit Koordinaten abgebildet. Der Mittelpunkt des Rechtecks ist die Hälfte der Breite an der x Position und analog dazu die Hälfte der Höhe an der y Position

Algorithm 5 Bestimmung der Augenmittelpunkte

procedure GETCENTER(eye)
  $center \leftarrow (eye.x + (eye.width/2), (eye.y + (eye.height/2))$
  $return\ center$
end procedure

Nachdem die Mittelpunkte der Augen gefunden wurden, lässt sich der Drehwinkel bestimmen. Die Methode *getRotationAngle* liefert abhängig von den

Mittelpunkten der Augen den Winkel zurück. Der Algorithmus 6 beschreibt die Implementation zur Berechnung des Drehwinkels.

Algorithm 6 Bestimmung des Drewhinkels

1: **procedure** GETROTATIONANGLE($leftEyeCenter, rightEyeCenter$)
2: $distance \leftarrow getDistance(left_point, right_point)$
3: $deltay \leftarrow leftEyeCenter.y - rightEyeCenter.y$ ▷ Bestimme delta der Höhe
4: $rotationAngle = asin(deltay/distance) * 180/PI$
5: **if** $rightEyeCenter.y < leftEyeCenter.y$ **then**
6: $rotationAngle = (-1) * rotationAngle$ ▷ Bestimme die Drehrichtung
7: **end if**
8: $return\ rotationAngle$
9: **end procedure**

Der Drehwinkel gibt an um wieviel Grad das Bild in welche Richtung gedreht werden muss, damit die Augen sich auf einer horizonatlen Linie befinden. Nachdem der Winkel bestimmt wurde wird das Gesicht affin gedreht. Das OpenCV Framework bietet hierfür bereits eine Implementationen an. Die Methode *warpAffine* dreht die Matrix um einen entsprechenden Winkel. Nachdem das Gesicht gedreht wurde werden unnötige Hintergrundinformationen herausgeschnitten. Die Methode *cropFace* ruft den Algorithmus 7 auf.

Algorithm 7 Zuschneiden des Gesichtes

1: **procedure** CROPFACE($alignedFace, top, left$)
2: $regionOfInterest \leftarrow newRectangle$
3: $subtractWidth \leftarrow alignedFace.cols * (left/100)$
4: $subtractHeight \leftarrow alignedFace.rows * (left/100)$
5: $regionOfInterest.width \leftarrow alignedFace.cols - subtractWidth$
6: $regionOfInterest.height \leftarrow alignedFace.rows - subtractHeight$
7: $regionOfInterest.x \leftarrow subtractWidth/2$
8: $regionOfInterest.y \leftarrow subtractHeight/2$
9: $croppedFace \leftarrow copy(aligendFace, regionOfInterest)$
10: $return\ croppedFace$
11: **end procedure**

Das ausgerichtete Gesicht wird um einen gewissen Prozentsatz von oben und von links verkleinert. Um das neue Rechteck bestimmen zu können wird zuerst berechnet um wieviel Pixel von der Breite und der Höhe abgezogen werden muss, anschließend werden die neuen x und y Positionen bestimmt. Nachdem das Rechteck ermittelt wurde wird der Inhalt des Gesichts in eine Matrix mit der Größe des ermittelten Rechtecks kopiert. Die konkrete Implementation befindet sich in der Klasse FeatureExtractor (vgl. $featureextractor.h$).

4.6.2 Gesichtsmerkmale mit SURF extrahieren

Merkmale werden mit dem SURF Algorithmus extrahiert. Das OpenCV Framework bietet bereits eine fertige Implementation des SURF Algorithmus an. Der Algorithmus 8 beschreibt den Einsatz des SURF Algorithmus zur Extraktion von Merkmalen. Die konkrete Implementation befindet sich in der Klasse $FeatureExtractor$ (vgl. $featureextractor.h$).

Algorithm 8 SURF Merkmale mit OpenCV

```
1: procedure EXTRACTFEATURES(frame)
2:     keypoints ← detectKeypoints(frame)
3:     features ← computeFeatures(frame, keypoints)
4:     return features
5: end procedure
```

4.6.3 Bekannte Gesichter finden

Wie im vorherigen Abschnitt beschrieben werden die Merkmale extrahiert und können gegen eine Datenbank von vorhandenen Gesichtern abgeglichen werden. Der Algorithmus 9 zeigt wie der Vergleich von Merkmalen umgesetzt wurde (vgl. $analyzer.cpp$).

Algorithm 9 Merkmale vergleichen

```
1: procedure ANALYZE
2:     for each face in faces do
3:         if face.isRecognized then
4:             Continue
5:         end if
6:         faceFeatures ← face.extractFeatures()
7:         distances ← vector
8:         while storedFace ← storedFaces.next do
9:             distances ← compare(faceFeatures, storedFace.getFeatures())
10:        end while
11:        idStoredFace ← findLeastDistance(distances, threshold)
12:        if idStoredFace = −1 then
13:            learnFace(face)
14:        else
15:            updateFace(face)
16:        end if
17:    end for
18: end procedure
```

Der Algorithmus überprüft jedes verfolgte Gesicht. Für jedes Gesicht das erfasst wurde wird überprüft ob das Gesicht bereits bekannt ist. Sollte das

der Fall sein wird beim nächsten Gesicht weitergemacht. Ansonsten werden die Merkmale vom Gesicht extrahiert. Anschließend gilt es die extrahierten Merkmale mit allen in einer Datenbank stehenden Gesichtern zu vergleichen. Für jedes Gesicht das in der Datenbank ist, werden die Merkmale von dem gespeicherten Gesicht mit denen des zu erkennenden Gesichts verglichen. Der Vergleich ist die Berechnung der Distanz beider Merkmalsvektoren zueinander. Nachdem jede bekannte Distanz Berechnet wurde wird die kleinste Distanz ermittelt. Sollte die kleinste Distanz über einem Grenzwert liegen dann ist das Gesicht nicht bekannt. In dem Fall wird das Gesicht vom System gelernt. Ansonsten wird der Datensatz erweitert.

4.6.4 Neue Gesichter lernen

Wenn die Distanz vom aktuellen Gesicht zu allen anderen bekannten Gesichtern über einem Grenzwert liegt wird das aktuelle Gesicht in die Datenbank aufgenommen. Dazu werden die Merkmale gespeichert damit diese nicht erneut Berechnet werden müssen. Anschließend werden alle mit dem Gesicht verknüpften Metadaten gespeichert. Die Metadaten enthalten die Zeitintervalle an denen das Gesicht gesehen wurde und das klassifizierte Geschlecht.

4.7 Geschlechtserkennung

Das Geschlecht einer Person kann mithilfe einer SVM bestimmt werden. Die Supported Vector Machines wurden bereits in Abschnitt 2.4 eingeführt und werden in der Implementation durch das OpenBR-Framework bereits implementiert (vgl. $genderestimator.cpp$). Jedes neue unbekannte Gesicht wird klassifiziert. Der im Konzept beschriebene Ablauf wird in Algorithmus 10 beschrieben.

Algorithm 10 Geschlecht klassifizieren

1: **procedure** GETGENDER($face$)
2: $query \leftarrow face$
3: $transform \leftarrow algorithm(GenderEstimation)$
4: $transform \leftarrow query$
5: $return\ query.file.get(Gender)$
6: **end procedure**

Im OpenBR-Framework stellt ein $query$ die Anfrage für einen bestimmten Algorithmus dar. Der query wird mit den Daten des Gesichtes initialisiert. Anschließend muss der query transformiert werden und durch einen Algorithmus laufen. Der $transform$ stellt die Schnittstelle zu den Algorithmen dar und wird mit dem Geschlechtsklassifizierungsalgorithmus initialisert. Der query wird an den transform gestellt und mit den Ergebnissen aus dem Algorithmus gefüllt. Das Ergebniss wird am Ende zurückgegeben. Wie im Konzept beschrieben wird

im Falle das die Person männlich ist +1 und im Falle das die Person weiblich ist -1 zurückgegeben.

4.8 Umsetzung des Web Interfaces

Das Webinterface wurde in PHP entwickelt und folgt dem im Abschnitt 3.8 vorgestellten MVC-Muster. Die Projektstruktur ist wie folgt definiert.

```
app
├── assets/
│   ├── css/*.css
│   ├── fonts/*
│   └── js/*.js
├── controller/
│   ├── Controller.php
│   ├── FaceController.php
│   └── IndexController.php
├── model/
│   ├── Database.php
│   ├── FaceEntity.php
│   ├── FaceEntityLoader.php
│   ├── LoginManager.php
│   └── User.php
├── view/
│   ├── html/*.html
│   └── View.php
└── index.php
```

Der komplette Quellcode zum Webinterface ist im Quellcode Anhang zu finden. Die Anfrage vom User wird an die index.php geleitet. Die Index Seite ruft den generischen Controller mit der Anfrage auf. In diesem generischen Controller wird bestimmt wie die Anfrage zu bearbeiten ist. Bevor die Anfrage das Model erreicht wird überprüft ob der Nutzer in dem System registriert und eingeloggt ist. Die Überprüfung findet im *LoginManager.php* statt. Anschließend wird geprüft welche Anfrage der Controller verarbeiten soll. Abhängig von der Anfrage wird eine bestimmte Seite von der View Komponente gerendert.

Kapitel 5

Evaluierung

In diesem Kapitel werden weitere Verfahren vorgestellt mit denen es möglich ist Gesichter darzustellen. Abschnitt 5.1 beschreibt zwei weitere populäre Gesichtserkennungalgorithmen und vergleicht diese in Abschnitt 5.2 mit dem eingesetzten SURF-Algorithmus. Der nächste Abschnitt 5.3 beschreibt die Wiedererkennungsraten von SURF und wertet mehrere Gesichtsdatenbanken aus im bezug auf die Geschlechtserkennung.

5.1 Andere Gesichtserkennungsalgorithmen

Außer dem vorgestellten SURF-Algorithmus zur Merkmalsextraktion existieren noch weitere zahlreiche Gesichtserkennungsalgorithmen. Die populärsten Gesichtserkennungsalgorithmen, Eigenfaces und Fisherfaces, werden in den nächsten beiden Unterabschnitten vorgestellt. Beide Algorithmen dienen zur Repräsentation von Gesichtern und können eingesetzt werden um Gesichter zu vergleichen.

5.1.1 Eigenfaces

Eigenfaces ist ein Verfahren zur Repräsentation von Gesichtsmerkmalen als eine Komposition von Komponenten und wurde von M.Turk und A.Pentland vorgestellt (vgl. [20]). Das Verfahren verwendet die Eigenmatrix und die Eigenwerte der Bildrepräsentation. Zuerst wird jedes Eingabebild in einen Vektor Φ, der Größe N, umgewandelt. Die Abbildung 5.1 zeigt eine Menge von Eingabebildern die beispielhaft verwendet werden.

Abbildung 5.1: Verschiedene Eingabebilder für Eigenfaces.
Quelle: [28]

Jedes Eingabebild wird benutzt um das Durchschnittsgesicht Ψ zu berechnen.

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Phi \tag{5.1}$$

Das Durchschnittsgesicht ist das arithmetische Mittel der Eingabebilder und wird in Abbildung 5.2 visuell dargestellt.

Abbildung 5.2: Durchschnittsgesicht.
Quelle: [28]

Wobei M die Anzahl der Eingabebilder ist. Ausgehend vom Durchschnittsgesicht kann die Differenz von jedem Eingabebild zum Durchschnittsgesicht ausgerechnet werden.

$$\Delta_i = \Phi_i - \Psi \tag{5.2}$$

Die Distanz Δ wird verwendet um die Kovarianz Matrix C zu berechnen.

$$C = \frac{1}{M} \sum_{i=1}^{M} \Delta_i \Delta_i^T = AA^T \in \mathbb{R}^{MxM} \tag{5.3}$$

Wobei $A = \Delta_1, \Delta_2, ..., \Delta_n$ ist. Die Eigenvektoren und Eigenwerte von AA^T sind definiert als

$$V = v_1, v_2, ..., v_r \tag{5.4}$$

und $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda r)$, $\lambda_1 \geq \lambda 2 \geq ...\lambda r > 0$, wobei r der Rang der Matrix A ist. Die Eigenvektoren und Eigenwerte von C sind Λ und $U = AV\Lambda^{\frac{1}{2}}$. Die Matrix U enthält die Eigengesichter, auch $Eigenfaces$ genannt. Zusammengefasst werden im Eigenface Verfahren die relevanten Gesichtsinformationen durch eine Eigenmatrix repräsentiert. Diese Repräsentation der Daten ist besonders effizient da nur die Varianz von Daten gespeichert wird, wobei unnötige Informationen verworfen werden. Eigenfaces eignet sich daher besonders gut Gesichter effizient zu speichern und zu repräsentierten.

5.1.2 Fisherfaces

Das Fisherface Verfahen beruht auf der linearen Analyse von Diskrimanten, auch Linear Discriminant Analysis (LDA) genannt. Die Idee besteht darin einen Unterraum zu finden der Vektoren der selben Klasse möglichst weit von anderen Vektoren einer anderen Klasse abbildet. Das Verfahren wurde von R.A. Fisher im Jahr 1936 vorgestellt und ist maßgebend für den Namen. Die resultierenden Basisvektoren die den Unterraum repräsentieren werden Fisherfaces genannt. Das Verfahren ist eine Erweiterung des Eigenface Verfahrens (vgl. [2]). Ähnlich wie beim Eigenface Verfahren wird das Durchschnittsgesicht Ψ berechnet. Der Unterschied besteht darin das zuerst die Durchschnittsgesichter jeder Klasse berechnet werden, anschließend wird die Streumatrix der Gesichter einer Klasse zu dem Durchschnittgesicht der Klasse berechnet.

$$S_w = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \Psi_i)(x_k - \Psi i)^T \tag{5.5}$$

Wobei c die Anzahl der verfügbaren Klassen, X_i die Menge der Gesichter die einer Klasse angehören und Ψ_i das Durchschnittgesicht einer Klasse ist. Danach wird die Streumatrix zwischen dem Durchschnittsgesicht Ψ, das aus allen trainierten Gesichtern berechnet wurde, und den Durchschnittsgesichtern Ψ_i der einzelnen Klassen berechnet.

$$S_w = \sum_{i=1}^{c} N_i(\Psi_i - \Psi)(\Psi i - \Psi)^T \tag{5.6}$$

Danach werden die Basisvektoren V gesucht die in S_w minimal werden und in S_b maximal werden. Dabei ist V eine Matrix und dessen Spalten v_i die Basisvektoren die den Unterraum beschreiben.

$$\frac{|V^T S_b V|}{|V^T S_w V|} \tag{5.7}$$

Die Lösung dazu ist ein Eigenwert Problem und wird durch

$$S_b V = S_w V \Lambda \tag{5.8}$$

gelöst. Wobei V die Matrix der Eigenvektoren ist und Λ die Diagonal Matrix der korrespondierenden Eigenwerte ist. Das Verfahren maximiert die Datenverteilung zwischen den Klassen. Den Unterschied zwischen beiden Verfahren sieht man auf Abbildung 5.3. Die Trainingsdaten wurden in einem 2-dimensionalen Raum abgebildet. Beide Verfahren bilden die Trainingsdaten auf einen 1-dimensionalen Punkt ab. Das Eigenface Verfahren das die Komponentenanalyse (PCA) nutzt verschmiert die Klassen miteinander, die damit nicht mehr linear separierbar bleiben. Damit wird eine größere Verteilung der Daten erreicht als im Fisherface Verfahren. Im Fisherface Verfahren werden die Klassen soweit wie möglich getrennt. Mit der Trennung der Klassen ist eine Klassifikation zwischen den Klassen deutlich einfacher als bei dem Eigenface Verfahren.

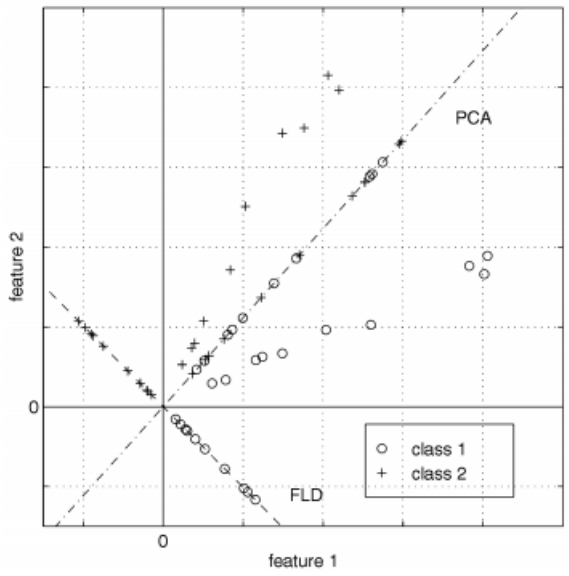

Abbildung 5.3: Vergleich zwischen Eigenfaces und Fisherfaces.
Quelle: [2]

5.2 Vergleich zu SURF

In diesem Abschnitt sollen die drei vorgestellten Verfahren verglichen werden. Da alle drei Verfahren unterschiedlich sind werden nur die generischen Punkte verglichen. Ein Vergleich auf Algorithmenebene findet nicht statt, da die Algorithmen verschiedene Ziele verfolgen. Verglichen werden Aspekte der Robustheit und Effizienz. Folgende Tabelle vergleicht welche Kernmethoden verwendet werden um Merkmale extrahieren zu können, ob die Merkmale invariant gegenüber Skalierungen, Rotationen und Helligkeit sind und ob sich aus den extrahierten Merkmalen das ursprüngliche Gesicht wieder ergeben kann. Außerdem wird verglichen ob sich die Merkmale wiederholen wenn sich die Gesichtsmimik verändert und Accessoirs, wie eine Brille, das Gesicht verändern.

	SURF	Eigenfaces	Fisherfaces
Methode	Hesse Detektor	PCA und Eigenmatrix	LDA
Skalierungsinvariant	Ja	Nein	Nein
Rotationsinvariant	Ja	Nein	Nein
Helligkeitsinvarianz	Hoch	Gering	Gering
Merkmalerekonstruktion	Nein	komplett	Nein
Wiederholbarkeit	Hoch	Gering	Mittel

5.3 Auswertung des entwickelten Systems

Um bestimmen zu können wie gut die entwickelte Software arbeitet wird die Wiedererkennungsrate bestimmt. Die Gesichter die zur Evaluierung genommen werden stammen aus der FERET Datenbank [6]. Der benutzte SURF Algorithmus verwendet Vektoren mit 64 Einträgen zur Abbildung der Gesichter. Möglich wäre auch der Einsatz von Vektoren mit 128 Einträgen, was zusätzliche Informationen über das Gesicht liefert und somit höhere Wiedererkennungsraten möglich macht. Die folgende Tabelle enthält die Wiedererkennungsraten vom SURF Algorithmus mit 64 und 128 Einträgen. Zusätzlich wird aufgeführt wie sich die Wiedererkennungsraten ändert wenn sich die Größe des Eingabebildes verdoppelt. Die Werte aus der Tabelle 5.1 beruhen auf den Messungen von Geng Du, Fei Su und Anni Cai (vgl. [8]).

Benutztes Merkmale	Wiedererkennungsrate (%)
SURF-64	95.6 %
SURF-64 doppelte Größe	95.2 %
SURF-128	96.0 %
SURF-128 doppelte Größe	96.6 %

Tabelle 5.1: Wiedererkennungsraten vom SURF Algorithmus

Den Messungen ist zu entnehmen das sich eine Verdoppelung der Merkmale oder der Größe kaum lohnt, da die Wiedererkennungsrate fast identisch bleibt. Von Interesse ist zudem die Rate der Geschlechtserkennung. Die Messungen zur Geschlechtserkennung wurden durch das OpenBR Framework erstellt. Durch den Einsatz von OpenBR können verschiedene Gesichtsdatenbanken als Eingabe zur Messung benutzt werden. Die Tabelle 5.2 zeigt welche Raten bei welcher Datenbank erreicht wurden.

Datenbank	Anz. Weiblich	Anz. Männlich	Gesamt	Fehlerrate (%)
AT & T	40	360	400	
Erkannt	121	279	400	
davon richtig	32	251	283	
davon falsch	89	28	117	29.9 %
UMIST	168	844	1012	
Erkannt	341	671	1012	
davon richtig	102	613	715	
davon falsch	239	58	297	29.3 %
MEDS-I	66	646	712	
Erkannt	95	617	712	
davon richtig	54	589	643	
davon falsch	41	28	69	9.6 %
MEDS-II	120	1186	1306	
Erkannt	185	1121	1306	
davon richtig	103	976	1079	
davon falsch	82	145	227	17.3 %

Tabelle 5.2: Raten der Geschlechtserkennung bei verschiedenen Datenbanken

Die erzielten Ergebnisse verdeutlichen das die Fehlerrate der Geschlechtserkennung abhängig von den Eingabedaten ist. Die Eingabedaten unterscheiden sich in der Größe der Bilder, den Lichtverhältnissen, der Kopfposition und der Mimik der Person. Eine Datenbank, wie etwa die AT&T oder UMIST Gesichtsdatenbank, die viel Variation in den Eingabedaten enthält, liefert schlechtere Ergebnisse als eine Gesichtsdatenbank die optimale Eingabedaten liefert mit wenig Variation. Zu sehen ist dies im Vergleich der AT&T und UMIST Datenbanken zu den MEDS-I und MEDS-II Datenbanken. Die MEDS Datenbanken enthalten Gesichter die gut für eine Geschlechtserkennung geeignet sind, da die Gesichter optimal ausgerichtet sind, sich die Größe der Bilder nicht ändert, das Lichtverhältnis konstant bleibt und die Gesichtsmimik sich kaum ändert. Aus den Messungen ergibt sich ebenfalls das die Klassifikation von Frauen im Durchschnitt schlechter ausfällt als die Klassifikation von Männern. Dies kann daran liegen das die Eingabemenge von weiblichen Gesichtern kleiner ist als die der männlichen Gesichter. Ebenfalls ist aus den Messungen zu entnehmen das die Gesichtsdetektion verlässlich arbeitet und alle Gesichter detektieren konnte.

Kapitel 6

Fazit und Ausblick

Die prototypische Realisierung einer Gesichtserkennenungssoftware im Rahmen der Abschlussarbeit zeigt, dass die Anforderungen eines Systems zur Erkennung, Klassifizierung und Wiedererkennung von Gesichtern umzusetzen waren. Das implementierte Webinterface ermöglicht eine einfache Verwaltung und Übersicht der gesammelten Daten. Die vorgenommene Testsimulation hat zwar eine gewisse Robustheit der Software gezeigt, jedoch konnten in dem Entwicklungszeitraum nicht genügend Daten gesammelt werden um eine endgültige Verifikation und Effizienzanalyse zu erstellen. Die in Kapitel 3.2 formulierten Anforderungen konnten durch die Implementierung folgendermaßen umgesetzt werden:

- Es wurde eine gemeinsame Datenbank benutzt um die Daten abzulegen, auf der sowohl die Gesichtserkennende Software als auch das Webinterface arbeitet.
- Durch das Webinterface können bestimmte Gesichter die erkannt wurde speziell gesucht werden. Sobald ein gesuchtes Gesicht von der Gesichtserkennenden Software erkannt wurde, ändert sich die Darstellung, bezüglich des Rahmens der um das Gesicht gezeichnet wird.
- Durch die generische Klasse $Capture$ können verschiedene Arten von Kameras angesprochen werden.
- Bevor das System gestartet wird kann eine bestimmte Auflösung angegeben werden in der die aufgenommenen Bilder zurückgeliefert werden.
- Durch den bereits implementierten Detektor der auf dem Verfahren von Viola und Jones basiert können Gesichter durchaus verlässlich auf jedem Bild detektiert werden. Die Testsimulation hat gezeigt das der Detektor eine Person, die eine Entfernung von mindestens 1 Meter zu der Kamera hat, verlässlich erfassen und deren Gesicht detektieren kann.
- Ein vom OpenCV bereitgestellter Tracker kann Gesichter effizient verfolgen ohne dabei das komplette Bild erneut nach Gesichtern abzusuchen.

Durch die Nutzung von Positionsdurchschnitten kann ein Bezug zum vorherig erkannten Gesicht erstellt werden um so unterscheiden zu können ob die Person bereits verfolgt wird.

- Der eingesetzte SURF Algorithmus ist in der Lage Merkmale als Vektoren zu extrahieren. Diese Vektoren können mit bereits bekannten Vektoren verglichen werden um bestimmen zu können ob das Gesicht bereits bekannt ist oder nicht.

- Die Implementation der Geschlechtserkennung wurde durch das OpenBR Framework umgesetzt. Die Fehlerquote in der Klassifizierung der Geschlechter ist jedoch hoch was auf schlechte oder zu wenige Trainingsdaten der SVM schließen lässt. Die SVM die benutzt wird kann erneut trainiert werden, was zu einer niedrigeren Fehlerquote führen könnte.

- Das geplante Webinterface wurde im MVC Muster umgesetzt und lässt im umgesetzten System gesammelte Datensätze anzeigen und editieren.

Das geplante Konzept wurde umgesetzt und zeigt das Personen mit Hilfe des SURF-Algorithmus verlässlich erkannt werden können, jedoch sei hier zu erwähnen das die Implementierung prototypisch ist und nicht für den produktiven Einsatz in den im Kapitel 1 genannten Anwendungsdomänen zu verwenden ist. Die implementierte Software bietet das Potential für verschiedene Erweiterungen. As Ausblick auf eine eventuelle Weiterarbeit an der Gesichtserkennungssoftware könnten die prototypisch implementierten Ansätze im Webinterface erweitert werden. Aus den Datensätzen könnte man Bewegungsprofile erstellen und gewisse Personen suchen lassen. Ebenfalls lässt sich die Gesichtserkennung erweitern. Die eingesetzte SVM könnte auf möglichst niedrige Fehlerquoten trainiert werden. Aufgenommene Datensätze könnten hier ebenfalls zum Training benutzt werden. Des Weiteren lässt sich die Erkennung so erweitern das Emotionen erkannt werden. Ein Einsatz in der Marktforschung wäre ebenfalls denkbar. Die Software könnte Personen die vor einem Werbebildschirm stehen erkennen und wichtige Daten für eine Produktevaluierung liefern. Das komplette Projekt wurde aber nur prototypisch umgesetzt und benötigt noch diverse Anpassungen damit ein produktiver Einsatz möglich wäre. Ein fortlaufender Test könnte noch genauere Ergebnisse liefern welche Komponenten weiterentwickelt werden müssten um einen produktiven Einsatz zu ermöglichen. Eine rechtliche Komponente müsste bei der Weiterarbeit ebenfalls bedacht werden, die des Datenschutzes. Die Software müsste so entwickelt werden das sie mit den Gesetzen des Datenschutzes konform ist. Eine Speicherung von biometrischen Merkmalen ohne Wissen der Person stellt eine Verletzung des momentanen Datenschutzes in Deutschland dar. Hierfür müssten Verfahren eingesetzt werden die nicht in Konflikt mit dem Gesetz zum Datenschutz kommen.

Quellcodeanhang

Listing 6.1: ageestimator.h

```
#ifndef AGEESTIMATOR_H
#define AGEESTIMATOR_H

#include <iostream>
#include <stdio.h>

#include <opencv2/core/core.hpp>
#include <openbr/openbr_plugin.h>


using namespace cv;
class AgeEstimator
{
public:
  AgeEstimator();
  int getAge(Mat *query);
private:
  QSharedPointer<br::Transform> transform;
};

#endif // AGEESTIMATOR_H
```

Listing 6.2: ageestimator.cpp

```
#include "ageestimator.h"

AgeEstimator::AgeEstimator()
{
 this->transform = br::Transform::fromAlgorithm("AgeEstimation");
}
int AgeEstimator::getAge(Mat* face){
  br::Template query(*face);
  query >> *this->transform;
  return query.file.get<int>("Age");
}
```

Listing 6.3: analyzer.h

```
#ifndef ANALYZER_H
#define ANALYZER_H

#include <iostream>
#include <vector>
#include <sstream>
#include <time.h>

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/objdetect/objdetect.hpp>

#include <boost/thread.hpp>

#include "configuration.h"
#include "person.h"
#include "featureextractor.h"
#include "trainer.h"
#include "database.h"
#include "logger.h"
#include "ageestimator.h"
#include "genderestimator.h"

using namespace std;
using namespace cv;

```

```
27 class Analyzer
28 {
29 public:
30     Analyzer(vector<Person*>* persons, Configuration *config, Trainer* trainer, Database* db, Logger* log,
           AgeEstimator* ageEstimator, GenderEstimator* genderEstimator);
31     ~Analyzer();
32     //make class callable for access to members
33     void operator() ();
34
35 private:
36     vector<Person*>* persons;
37     Configuration* config;
38     CascadeClassifier eyes;
39     FeatureExtractor* featureExtractor;
40     Trainer* trainer;
41     Database* db;
42     Logger* log;
43     AgeEstimator* ageEstimator;
44     GenderEstimator* genderEstimator;
45     FileStorage *fs;
46
47
48     pair<int, int> getGreatesMatch(vector<pair<int, int> > *matches);
49     void updateTimeSettings(int pid);
50     int insertNewPerson(Mat *personImage, Person *person);
51     void saveActiveInterval(Person* person);
52
53     void saveFeatures(string featureName, Mat* features);
54     Mat readFeature(string featureName);
55
56
57 };
58
59 #endif // ANALYZER_H
```

Listing 6.4: analyzer.cpp

```
1  #include "analyzer.h"
2
3  Analyzer::Analyzer(vector<Person*>* persons, Configuration* config, Trainer* trainer, Database* db, Logger*
       log, AgeEstimator* ageEstimator, GenderEstimator* genderEstimator)
4  {
5      this->persons = persons;
6      this->config = config;
7      this->eyes.load(config->getProperty("Haarcascade.Eyes"));
8      this->featureExtractor = new FeatureExtractor(&this->eyes, log);
9      this->trainer = trainer;
10     this->db = db;
11     this->log = log;
12     this->ageEstimator = ageEstimator;
13     this->genderEstimator = genderEstimator;
14     this->fs = new FileStorage();
15 }
16
17 Analyzer::~Analyzer(){
18     this->fs->release();
19 }
20
21 void Analyzer::operator ()(){
22     while(true){
23         if(persons->size() == 0){
24             //sleep for 5 second
25             boost::this_thread::sleep(boost::posix_time::millisec(500));
26             continue;
27         }else{
28
29            for(size_t person = 0; person < persons->size(); person++){
30
31                time_t now;
32                time(&now);
33
34                float diffTime = difftime(now, persons->at(person)->getLastSeen());
35                if(diffTime > 1){
36                    //save intervalls before erasing
37                     this->saveActiveInterval(persons->at(person));
38                     persons->erase(persons->begin()+person);
39                     continue;
40                }
41
42                //skip recognized faces
43                if(persons->at(person)->getRecId() != -1){
44                    continue;
45                }
46
47                //skip empty faces
48                if(persons->at(person)->getImage()->empty()){
49                    continue;
50                }
51
```

```
52                  Mat grayScaledPerson = featureExtractor->grayScale(persons->at(person)->getImage());
53                  Mat equalizedPerson = featureExtractor->equalizeMat(&grayScaledPerson);
54                  Mat alignedFace;
55                  vector<Rect> eyes = featureExtractor->searchEyes(&equalizedPerson);
56
57                  imshow("search eyes", equalizedPerson);
58                  if(eyes.size() == 2){
59                     alignedFace = featureExtractor->alignFace(&equalizedPerson, eyes[0], eyes[1]);
60
61                    if(!alignedFace.empty()){
62
63                       Mat* queryFeatures = featureExtractor->extractFeatures(&alignedFace, true, "query");
64                       alignedFace.release();
65                       persons->at(person)->setFeature(queryFeatures);
66                       vector<pair <int, int > > dbMatches;
67                       //get targets from the database
68                       string getTargetsQuery = "SELECT p.id, pi.image FROM people p JOIN people_information pi
                              ON p.id = pi.pid";
69                       if(db->query(getTargetsQuery)){
70                           MYSQL_RES* result = db->getResult();
71
72                           while(MYSQL_ROW row = mysql_fetch_row(result)){
73                               stringstream image;
74                               image << row[1];
75                               //load image
76
77                               Mat targetFromFile = this->readFeature("face_"+image.str().substr(0, image.str().
                                      size()-4));
78
79                               int matchesFound = trainer->match(queryFeatures, &targetFromFile);
80
81                               dbMatches.push_back(make_pair(matchesFound, atoi(row[0])));
82                           }
83
84                       }
85                       db->freeResults();
86
87                       //search for best match
88                       //(best match, pid)
89
90                       //debug messages
91                       for(size_t matchesCount = 0; matchesCount < dbMatches.size(); matchesCount++){
92                           stringstream debugMatches;
93                           debugMatches << dbMatches[matchesCount].first << " for pid " <<  dbMatches[
                                  matchesCount].second;
94                           log->debug(debugMatches.str());
95                       }
96
97                       pair<int, int> greatestMatch = this->getGreatesMatch(&dbMatches);
98
99                       if(greatestMatch.first >= atoi(config->getProperty("Recognition.MatchThreshold").c_str()))
                              {
100                          //known
101                          persons->at(person)->setRecId(greatestMatch.second);
102                          //check if enough pictures exists for this person
103                          stringstream getPidInformation ;
104                          getPidInformation << "SELECT count(image), age, gender, wanted FROM people_information
                                 pi JOIN people p ON p.id=pi.pid WHERE pid=" << greatestMatch.second;
105
106                          db->query(getPidInformation.str());
107                          MYSQL_RES* result = db->getResult();
108                          MYSQL_ROW row = mysql_fetch_row(result);
109
110                          int countNumber = atoi(row[0]);
111                          int age = atoi(row[1]);
112                          int gender = atoi(row[2]);
113                          bool wanted = false;
114
115                          if(atoi(row[3]) == 1 ){
116                              wanted = true;
117                          }
118
119                          persons->at(person)->setAge(age);
120                          persons->at(person)->setGender(gender);
121                          persons->at(person)->setWanted(wanted);
122
123                          db->freeResults();
124
125                          if(countNumber < atoi(config->getProperty("Recognition.ImagePerPerson").c_str())){
126                              stringstream faceDBPath;
127                              faceDBPath << config->getProperty("General.FacesDB") << greatestMatch.second << "_
                                     " << (countNumber+1);
128
129                              //write update image feature for face if known
130                              this->saveFeatures(boost::to_string(greatestMatch.second)+"_"+boost::to_string((
                                     countNumber+1)), queryFeatures);
131
132                              if(config->getProperty("Logging.SaveFace").compare("0") != 0){
133                                  faceDBPath << ".png";
134                                  imwrite(faceDBPath.str(), alignedFace);
135                              }
```

```
136
137                             //update db
138                             stringstream insertDBInformation;
139                             insertDBInformation << "INSERT␣INTO␣people_information␣(`pid`,␣`image`)␣VALUES␣('"
                                    <<greatestMatch.second << "',␣'" << greatestMatch.second << "_" <<(
                                    countNumber+1) << ".png')";
140                             db->query(insertDBInformation.str());
141                             db->freeResults();
142                         }
143                         this->updateTimeSettings(greatestMatch.second);
144                         time_t lastSeen;
145                         time(&lastSeen);
146                         persons->at(person)->setLastSeen(lastSeen);
147
148
149                     }else{
150                         //not known insert new person
151
152                         this->persons->at(person)->setRecId(this->insertNewPerson(&alignedFace, persons->at(
                                person)));
153                     }
154
155                 }
156                 }else{
157                     //face could not be aligned forget person and try next frame
158                     stringstream forgetStream;
159                     forgetStream << "Could␣not␣align␣person␣with␣Tracking␣ID:␣"<<persons->at(person)->
                            getTrackId();
160                     log->debug(forgetStream.str());
161                     //try next time
162                     //save image from person before clearing image
163                     stringstream tracksave;
164                     tracksave << config->getProperty("Logging.TrackFacePath")<< persons->at(person)->getTrackId
                            ()<<"_trackedId.png";
165                     imwrite(tracksave.str(), *persons->at(person)->getImage());
166                     persons->at(person)->clearImage();
167                 }
168
169             }
170
171         }
172         //prevent from overloading the cpu
173         boost::this_thread::sleep(boost::posix_time::millisec(100));
174         //make sure we can be interrupted
175         boost::this_thread::interruption_point();
176     }
177 }
178
179 pair<int, int> Analyzer::getGreatesMatch(vector<pair <int, int > > *matches){
180     pair<int, int> greatestMatch = make_pair(0,0);
181
182     for(size_t match = 0; match < matches->size(); match++){
183         if(matches->at(match).first > greatestMatch.first){
184             greatestMatch.first = matches->at(match).first;
185             greatestMatch.second = matches->at(match).second;
186         }
187     }
188
189     return greatestMatch;
190 }
191
192 void Analyzer::updateTimeSettings(int pid){
193     stringstream updateDBPeople;
194     updateDBPeople << "UPDATE␣people␣SET␣last_seen␣=␣NOW()␣WHERE␣id␣=" << pid;
195     log->debug(updateDBPeople.str());
196     db->query(updateDBPeople.str());
197     db->freeResults();
198
199     //TODO: insert interval visited
200 }
201
202 int Analyzer::insertNewPerson(Mat* personImage, Person* person){
203
204     //detect age
205     int age = ageEstimator->getAge(person->getImage());
206
207     //detect gender
208     int gender = genderEstimator->getGender(person->getImage());
209
210     person->setAge(age);
211     person->setGender(gender);
212     stringstream insertNewPerson;
213     insertNewPerson <<  "INSERT␣INTO␣people␣(`first_seen`,␣`last_seen`,␣`age`,␣`gender`)␣VALUES␣(NOW(),␣NOW(),
            ␣'"<< age << "',␣'"<<gender<<"')";
214     db->query(insertNewPerson.str());
215     db->freeResults();
216     int lastId = db->getLastInsertID();
217
218     stringstream faceDBPath;
219     faceDBPath << config->getProperty("General.FacesDB") << lastId << "_1" << ".png";
220
```

```
221     this->saveFeatures(boost::to_string(lastId)+"_1", person->getFeature());
222
223     if(config->getProperty("Logging.SaveFace").compare("0") != 0){
224         imwrite(faceDBPath.str(), *personImage);
225     }
226
227     stringstream insertImage;
228     insertImage << "INSERT INTO people_information ('pid', 'image') VALUES ('"<<lastId << "', '"<<lastId << "
            _1" << ".png')";
229     db->query(insertImage.str());
230     db->freeResults();
231     return lastId;
232 }
233
234 void Analyzer::saveActiveInterval(Person *person){
235         //TODO: do some more intelligent interval savings
236         if(person->getRecId() == -1){
237             return;
238         }
239         stringstream intervalQuery;
240         intervalQuery << "INSERT INTO people_interval ('pid', 'interval_start', 'interval_end') VALUES ('"<<
                person->getRecId()<<"', FROM_UNIXTIME("<<person->getFirstSeen()<<"), FROM_UNIXTIME("<<person->
                getLastSeen()<<"))";
241         db->query(intervalQuery.str());
242 }
243
244 void Analyzer::saveFeatures(string featureName, Mat* features){
245     FileStorage fs(config->getProperty("Logging.KeypointFile"), FileStorage::APPEND);
246     fs << "face_"+featureName << *features;
247     fs.release();
248 }
249
250 Mat Analyzer::readFeature(string featureName){
251     Mat feature;
252
253     FileStorage fs(config->getProperty("Logging.KeypointFile"), FileStorage::READ);
254     // FileNode featureNode = fs[featureName];
255     //read( featureNode, feature );
256     fs[featureName] >> feature;
257     fs.release();
258     return feature;
259 }
```

Listing 6.5: cameraresolution.h

```
1  #ifndef CAMERARESOLUTION_H
2  #define CAMERARESOLUTION_H
3
4
5  class CameraResolution
6  {
7  public:
8
9    CameraResolution(int width, int height);
10   int width;
11   int height;
12 };
13
14 #endif // CAMERARESOLUTION_H
```

Listing 6.6: cameraresolution.cpp

```
1  #include "cameraresolution.h"
2
3  CameraResolution::CameraResolution(int width, int height)
4  {
5    this->width = width;
6    this->height = height;
7  }
```

Listing 6.7: capture.h

```
1  #ifndef CAPTURE_H
2  #define CAPTURE_H
3
4  #include "cameraresolution.h"
5  #include <opencv2/imgproc/imgproc.hpp>
6  #include <opencv2/highgui/highgui.hpp>
7  #include <iostream>
8
9  using namespace cv;
10 using namespace std;
11
12 class Capture
```

```
{
public:
  Capture(int device);
  Capture();
  ~Capture();
  Mat* queryFrame();
  bool openCapture(CameraResolution* resolution);
  void closeCapture();
  double getFPS();
private:
  VideoCapture *cap;
  Mat frame;
  Mat modifiedFrame;
  int device;
  CameraResolution* resolution;

};

#endif // CAPTURE_H
```

Listing 6.8: capture.cpp

```
#include "capture.h"

Capture::Capture(int device)
{
  this->device = device;
}

Capture::Capture(){
    this->device = 0;
}

Capture::~Capture(){
  delete cap;
}

bool Capture::openCapture(CameraResolution* resolution){

  cap = new VideoCapture();

  cap->set(CV_CAP_PROP_CONVERT_RGB, false);
  cap->set(CV_CAP_PROP_FPS, 60.0);

  cap->open(this->device);

  if(!cap->isOpened()){
      cout << "ERROR cannot open Capture" << endl;
      return false;
  }
  else{
      this->resolution = resolution;
      //if capture supports it, then try to force it
      return true;
  }
}

void Capture::closeCapture(){
  cap->release();
}

Mat* Capture::queryFrame(){
  if(this->cap->isOpened()){
      cap->operator >>(frame);
      //check if camera supports frame resizing then resize by hand else copy pointer
      if(this->frame.cols != this->resolution->width
      || this->frame.rows != this->resolution->height){
          resize(this->frame, this->modifiedFrame, Size(this->resolution->width, this->resolution->height), 0,
                 0, INTER_LINEAR);
      }else{
          this->modifiedFrame = this->frame;
      }
  }
  return &this->modifiedFrame;
}
```

Listing 6.9: configuration.h

```
#ifndef CONFIGURATION_H
#define CONFIGURATION_H

//for windows compilation if ever
#ifdef _WIN32
    #include <direct.h>
    #define GetCurrentDir _getcwd
#else
    #include <unistd.h>
```

```
10      #define GetCurrentDir getcwd
11  #endif
12
13  #include <iostream>
14  #include <string>
15  #include <sstream>
16  #include <boost/property_tree/ptree.hpp>
17  #include <boost/property_tree/ini_parser.hpp>
18
19  namespace bpt = boost::property_tree;
20  using namespace std;
21
22  class Configuration
23  {
24  public:
25
26      Configuration();
27      void readIni(string iniPath);
28
29     string getProperty(const char *key);
30  private:
31      bpt::ptree properties;
32      void loadDefaultConfig();
33  };
34
35  #endif // CONFIGURATION_H
```

Listing 6.10: configuration.cpp

```
1   #include "configuration.h"
2
3   Configuration::Configuration()
4   {
5       loadDefaultConfig();
6   }
7
8   void Configuration::readIni(string iniPath){
9
10      bpt::ini_parser::read_ini(iniPath, properties);
11
12  }
13
14  string Configuration::getProperty(const char* key){
15          return properties.get<string>(key, "");
16  }
17
18  void Configuration::loadDefaultConfig(){
19      properties.put("Database.Host", "localhost");
20      properties.put("Database.Port", "3306");
21      properties.put("Database.User", "root");
22      properties.put("Database.Password", "");
23      properties.put("Database.Name", "Vizage");
24
25      char curDir[256];
26      GetCurrentDir(curDir, sizeof(curDir));
27      stringstream path;
28      path << curDir;
29
30      properties.put("Dependencies.OpenBR", "/home/aschens/Vizage/dependencies/openbr");
31      properties.put("Haarcascade.Face", path.str()+"/resources/haarcascade_frontalface_default.xml");
32      properties.put("Haarcascade.Eyes", path.str()+"/resources/haarcascade_eyes.xml");
33  }
```

Listing 6.11: database.h

```
1   #ifndef DATABASE_H
2   #define DATABASE_H
3
4   #include <iostream>
5   #include "configuration.h"
6   #include <mysql/mysql.h>
7
8   using namespace std;
9
10  class Database
11  {
12  public:
13      Database(Configuration* config);
14      ~Database();
15      bool connect();
16      bool query(string queryString);
17      MYSQL_RES* getResult();
18      void freeResults();
19      int getLastInsertID();
20      void close();
21  private:
22      Configuration* config;
```

```
23      MYSQL *connection;
24      MYSQL_RES *result;
25  };
26
27  #endif // DATABASE_H
```

Listing 6.12: database.cpp

```
1   #include "database.h"
2
3   Database::Database(Configuration* config)
4   {
5       this->config = config;
6       this->connection = mysql_init(NULL);
7   }
8
9   Database::~Database(){
10      mysql_free_result(this->result);
11      mysql_close(this->connection);
12  }
13
14  bool Database::connect(){
15      cout << "INFO Connecting to " << config->getProperty("Database.Host") << endl;
16     return mysql_real_connect(this->connection, config->getProperty("Database.Host").c_str(),
17                               config->getProperty("Database.User").c_str(),
18                               config->getProperty("Database.Password").c_str(),
19                               config->getProperty("Database.Name").c_str(), 0 , NULL, 0);
20  }
21
22  bool Database::query(string queryString){
23      if(mysql_query(this->connection, queryString.c_str())){
24          cout << "ERROR Query '"<< queryString << "' failed!" << endl;
25          return false;
26      }else{
27          result = mysql_use_result(this->connection);
28          return true;
29      }
30  }
31
32  MYSQL_RES* Database::getResult(){
33      return this->result;
34  }
35
36  void Database::freeResults(){
37      mysql_free_result(this->result);
38  }
39
40  int Database::getLastInsertID(){
41      string queryLastID = "SELECT LAST_INSERT_ID()";
42      this->query(queryLastID);
43      int lastId = atoi(mysql_fetch_row(getResult())[0]);
44      this->freeResults();
45      return lastId;
46  }
47
48  void Database::close(){
49      mysql_close(this->connection);
50  }
```

Listing 6.13: detector.h

```
1   #ifndef DETECTOR_H
2   #define DETECTOR_H
3
4   #include <iostream>
5   #include <string>
6   #include <vector>
7   #include <opencv2/core/core.hpp>
8   #include <opencv2/imgproc/imgproc.hpp>
9   #include <opencv2/highgui/highgui.hpp>
10  #include <opencv2/features2d/features2d.hpp>
11  #include <opencv2/contrib/detection_based_tracker.hpp>
12
13
14  using namespace cv;
15  using namespace std;
16
17  class Detector
18  {
19  public:
20    Detector(string cascadePath);
21    ~Detector();
22    bool runDetector();
23    vector<DetectionBasedTracker::Object> getFaces(Mat* frame);
24  private:
25    DetectionBasedTracker *tracker;
26    DetectionBasedTracker::Parameters param;
```

```
};

#endif // DETECTOR_H
```

Listing 6.14: detector.cpp

```
#include "detector.h"

Detector::Detector(string cascadePath)
{
  this->param.maxObjectSize = 400;
  //face survives 50 frames till not seen
  this->param.maxTrackLifetime = 1;
  this->param.minDetectionPeriod = 5;
  this->param.minNeighbors = 6;
  this->param.minObjectSize = 20;
  this->param.scaleFactor = 1.1;
  this->tracker = new DetectionBasedTracker(cascadePath, this->param);
}

Detector::~Detector(){
}

bool Detector::runDetector(){
  return this->tracker->run();
}

vector<DetectionBasedTracker::Object> Detector::getFaces(Mat* frame){

    vector<DetectionBasedTracker::Object> objFaces;
    Mat frame_gray;
    cvtColor(*frame, *&frame_gray, CV_BGR2GRAY);
    equalizeHist(*&frame_gray, *&frame_gray);

    this->tracker->process(*&frame_gray);
    this->tracker->getObjects(*&objFaces);

    return objFaces;
}
```

Listing 6.15: featureextractor.h

```
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/nonfree/features2d.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/contrib/detection_based_tracker.hpp>

#include "logger.h"

#ifndef FEATUREEXTRACTOR_H_
#define FEATUREEXTRACTOR_H_

using namespace std;
using namespace cv;

const static Size THUMBNAIL_SIZE = Size(200, 200);
const float PI = 3.141592654;

typedef  vector< pair< Mat , Rect > > FaceFeatures;

class FeatureExtractor {
public:
    FeatureExtractor(CascadeClassifier* eyes, Logger* log);
    virtual ~FeatureExtractor();
    FaceFeatures getFaceFeatures(Mat *frame, vector<DetectionBasedTracker::Object> *roi);
    Mat *extractFeatures(Mat *frame, bool showKeypoints, string windowName);
    Mat cropRegion(Mat region, Rect roi);
    Mat cropFace(Mat face, int percentTop, int percentRight);
    vector<Rect> searchEyes(Mat *frame);
    Mat alignFace(Mat *face, Rect leftEye, Rect rightEye);
    Mat grayScale(Mat *face);
    Mat equalizeMat(Mat *face);

private:
    //members
    CascadeClassifier* eyes;
    SurfFeatureDetector* fdetector;
    SurfDescriptorExtractor* dextractor;
    //functions
    Mat scaleFace(Mat face);
    Mat resizeFace(Mat face);
```

```
46      CvPoint getCenter(Rect rect);
47      float getRotationAngle(CvPoint leftPoint, CvPoint rightPoint);
48      float getDistance(CvPoint p1, CvPoint p2);
49      Logger* log;
50  };
51
52  #endif /* FACEEXTRACTOR_H_ */
```

Listing 6.16: featureextractor.h

```
1   #include "featureextractor.h"
2
3   FeatureExtractor::FeatureExtractor(CascadeClassifier* eyes, Logger* log) {
4       this->eyes = eyes;
5       this->fdetector = new SurfFeatureDetector(100);
6       this->dextractor = new SurfDescriptorExtractor();
7       this->log = log;
8   }
9
10  FeatureExtractor::~FeatureExtractor() {
11      // TODO Auto-generated destructor stub
12  }
13
14  FaceFeatures FeatureExtractor::getFaceFeatures(Mat *frame, vector<DetectionBasedTracker::Object> *faces) {
15
16      FaceFeatures featuresToPosition;
17
18      for (unsigned int i = 0; i < faces->size(); i++) {
19
20           Mat cropped_face;
21
22          if(faces->at(i).first.x == 0 || faces->at(i).first.y == 0){
23              cropped_face = *frame;
24          }else{
25              cropped_face = this->cropRegion(*frame, faces->at(i).first);
26          }
27
28          vector<Rect> eyes = this->searchEyes(&cropped_face);
29          //compensate failures
30          if (eyes.size() == 2) {
31             Mat alignedFace = this->alignFace(&cropped_face, eyes[0], eyes[1]);
32             alignedFace = grayScale(&alignedFace);
33
34             Mat* features = extractFeatures(&alignedFace, false, "");
35             featuresToPosition.push_back(make_pair(*features, faces->at(i).first));
36          }
37      }
38      return featuresToPosition;
39  }
40
41
42  Mat* FeatureExtractor::extractFeatures(Mat *frame, bool showKeypoints, string windowName){
43      Mat *features = new Mat();
44      vector<KeyPoint> keypoints;
45      fdetector->detect(*frame, keypoints);
46
47      if(showKeypoints){
48          Mat imgKeypoints;
49          drawKeypoints( *frame, keypoints, imgKeypoints, Scalar::all(-1), DrawMatchesFlags::DEFAULT );
50          imshow(windowName, imgKeypoints);
51      }
52
53      dextractor->compute(*frame, keypoints, *features);
54      return features;
55  }
56
57
58
59  Mat FeatureExtractor::alignFace(Mat *face, Rect left_eye, Rect right_eye) {
60
61      Mat aligned_face;
62
63      CvPoint left_center = getCenter(left_eye);
64      CvPoint right_center = getCenter(right_eye);
65
66      //compensate cascade failures
67      if(left_center.x > right_center.x){
68          CvPoint tmp = right_center;
69          right_center = left_center;
70          left_center =tmp;
71      }
72
73      float angle = getRotationAngle(left_center, right_center);
74
75      if(abs(angle) > 10){
76          return aligned_face;
77      }
78
79      stringstream debugRotationAngle;
```

```
80      debugRotationAngle << "Rotating Face at " << angle << "degrees";
81      log->debug(debugRotationAngle.str());
82      //need scale for crop and or mask
83      //float scale = get_distance(left_center, right_center);
84      Point2f src_center(left_center.x, left_center.y);
85      Mat rot_mat = getRotationMatrix2D(src_center, angle, 1.0);
86      warpAffine(*face, aligned_face, rot_mat, face->size());
87
88      //scale aligned face to size
89      Mat croppedAlignedFace = cropFace(aligned_face, 15, 35);
90
91      Mat resized = resizeFace(croppedAlignedFace);
92      return resized;
93  }
94
95  float FeatureExtractor::getDistance(CvPoint p1, CvPoint p2){
96      float dx = p1.x - p2.x;
97      float dy = p1.y - p2.y;
98      return sqrt(dx*dx+dy*dy);
99  }
100
101 CvPoint FeatureExtractor::getCenter(Rect rect) {
102     CvPoint center = cvPoint((rect.x + rect.width / 2),
103             (rect.y + rect.height / 2));
104     return center;
105 }
106
107 float FeatureExtractor::getRotationAngle(CvPoint left_point,
108         CvPoint right_point) {
109     //distance
110     float distance = getDistance(left_point, right_point);
111     //cout <<distance << " distance between eyes"<<endl;
112     float rotation_angle = 0;
113     //build triangle
114     float tx = right_point.x - left_point.x;
115     if(tx < 0){
116         tx = (-1)*tx;
117     }
118
119     float ty = left_point.y - right_point.y;
120
121     if(ty < 0){
122         ty = (-1)*ty;
123     }
124
125     rotation_angle = ty / distance;
126     float rotation_angle_pi = asin(rotation_angle)*180/PI;
127
128     //determine in wich direction to rotate
129     if(right_point.y < left_point.y){
130         rotation_angle_pi = (-1) * rotation_angle_pi;
131     }
132
133     return rotation_angle_pi;
134 }
135
136 vector<Rect> FeatureExtractor::searchEyes(Mat *face) {
137     vector<Rect> eyes;
138     this->eyes->detectMultiScale(*face, eyes, 1.1, 2,  CV_HAAR_DO_CANNY_PRUNING,
139             Size(30, 30));
140     return eyes;
141 }
142
143 Mat FeatureExtractor::cropRegion(Mat region, Rect roi) {
144     Mat cropped_image;
145     Mat(region, roi).copyTo(cropped_image);
146     return cropped_image;
147 }
148
149 Mat FeatureExtractor::cropFace(Mat face, int top, int left){
150     Rect crop_region_f;
151
152     float new_width = face.cols * (float) left/100;
153     float new_height = face.rows * (float) top/100;
154
155     crop_region_f.width = face.cols - new_width;
156     crop_region_f.height = face.rows- new_height;
157     crop_region_f.x = new_width/2;
158     crop_region_f.y = new_height/2;
159
160     return cropRegion(face, crop_region_f);
161 }
162
163 Mat FeatureExtractor::resizeFace(Mat face){
164     Mat resized;
165     resize(face, resized, THUMBNAIL_SIZE, 0, 0 , INTER_LINEAR);
166     return resized;
167 }
168
169 Mat FeatureExtractor::grayScale(Mat *face){
170     Mat grayImg;
```

```
171     cvtColor(*face, grayImg, CV_BGR2GRAY);
172     return grayImg;
173 }
174
175 Mat FeatureExtractor::equalizeMat(Mat *face){
176     Mat dst;
177     equalizeHist(*face, dst);
178     return dst;
179 }
```

Listing 6.17: genderestimator.h

```
1  #ifndef GENDERESTIMATOR_H
2  #define GENDERESTIMATOR_H
3
4  #include <iostream>
5  #include <stdio.h>
6
7  #include <opencv2/core/core.hpp>
8  #include <openbr/openbr_plugin.h>
9
10 using namespace cv;
11
12 class GenderEstimator
13 {
14 public:
15     GenderEstimator();
16     int getGender(Mat *query);
17 private:
18     QSharedPointer<br::Transform> transform;
19 };
20
21 #endif // GENDERESTIMATOR_H
```

Listing 6.18: genderestimator.cpp

```
1  #include "genderestimator.h"
2
3  GenderEstimator::GenderEstimator()
4  {
5      this->transform = br::Transform::fromAlgorithm("GenderEstimation");
6
7  }
8
9  int GenderEstimator::getGender(Mat* face){
10   br::Template query(*face);
11   query >> *this->transform;
12   std::cout << query.file.flat().toStdString();
13   return query.file.get<int>("Gender");
14 }
```

Listing 6.19: logger.h

```
1  #ifndef LOGGER_H
2  #define LOGGER_H
3
4  #include <locale>
5  #include <string>
6  #include <iostream>
7  #include <sstream>
8
9  #include <boost/date_time/posix_time/posix_time.hpp>
10 #include <boost/date_time/gregorian/gregorian.hpp>
11
12 #include "configuration.h"
13
14 using namespace std;
15
16 class Logger
17 {
18 public:
19
20     Logger(Configuration* config){
21         this->config = config;
22     }
23
24     virtual void info(string msg){cout << "[INFO␣"<< utcDate()<< "]␣" << msg << endl;}
25     virtual void debug(string msg){cout << "[DEBUG␣"<< utcDate()<<  "]␣"<< msg << endl;}
26     virtual void warning(string msg){cout << "[WARNING␣"<<  utcDate() << "]␣" << msg<< endl;}
27     virtual void error(string msg){cout << "[ERROR␣"<< utcDate()<< "]␣" << msg << endl;}
28
29 protected:
30     Configuration* config;
31
```

```
32      string utcDate(){
33          using namespace boost::posix_time; /* put that in functions where you work with time (namespace) */
34
35          ptime now = microsec_clock::local_time();
36
37          std::stringstream ss;
38          ss << now.date().year() << "-" <<static_cast<int>(now.date().month()) << "-" << now.date().day()
39             << "␣|␣" <<now.time_of_day().hours() << ":" << now.time_of_day().minutes() << ":" << now.
                   time_of_day().seconds();
40
41         return ss.str();
42      }
43  };
44
45  #endif // LOGGER_H
```

Listing 6.20: main.cpp

```
1   #include <QApplication>
2   #include <iostream>
3   #include <time.h>
4   #include <string>
5   #include <sstream>
6   #include <vector>
7
8   //declaration due to name mansing from the c++ compiler
9
10  #include <opencv2/core/core.hpp>
11  #include <opencv2/highgui/highgui.hpp>
12
13  #include <openbr/openbr_plugin.h>
14
15  #include <boost/thread.hpp>
16  #include <boost/lexical_cast.hpp>
17
18  #include "configuration.h"
19  #include "capture.h"
20  #include "kinectcapture.h"
21  #include "detector.h"
22  #include "person.h"
23  #include "analyzer.h"
24  #include "trainer.h"
25  #include "database.h"
26  #include "filelogger.h"
27  #include "ageestimator.h"
28  #include "genderestimator.h"
29  #include "server.h"
30
31  using namespace cv;
32  using namespace std;
33
34  Mat validateRegion(Mat* frame, Rect face, int percentToExtend);
35
36  int main(int argc, char *argv[])
37  {
38    //init configuration
39    Configuration* config = new Configuration();
40
41    if(argc == 2){
42
43        config->readIni(argv[1]);
44
45        cout << "INFO␣Configuration␣loaded."<<endl;
46    }else{
47
48        cout << "WARNING␣No␣Configuration␣provided.␣Using␣default␣Configuration,␣incompatible␣behavior␣may␣occur
                !"<<endl;
49    }
50
51    //init logger
52    Logger* log = new FileLogger(config);
53
54    //init database
55    Database* db = new Database(config);
56
57    if(db->connect() == false) {
58        log->error("Could␣not␣connect␣to␣Database,␣check␣your␣settings!");
59        return -1;
60    }else{
61        log->info("Connection␣established␣to␣"+
62                  config->getProperty("Database.User")+
63                  "@"+config->getProperty("Database.Host")+
64                  ":"+config->getProperty("Database.Port")+
65                  "/"+config->getProperty("Database.Name"));
66    }
67
68    //init openbr
69    br::Context::initialize(argc, argv, config->getProperty("Dependencies.OpenBR").c_str());
70
```

```
71  log->info("OpenBR initialized");
72
73  //init capture + detector + vectors
74  //check if we use kinect
75  Capture * capture;
76  if(config->getProperty("Capture.Device").compare("kinect") == 0){
77      log->info("Using Kinect Device");
78      capture = new KinectCapture();
79  }else{
80      capture = new Capture(0);
81  }
82
83  Detector* detector = new Detector(config->getProperty("Haarcascade.Face"));
84
85  AgeEstimator* ageEstimator = new AgeEstimator();
86  GenderEstimator* genderEstimator = new GenderEstimator();
87
88
89  vector<DetectionBasedTracker::Object> faces;
90  vector<Person*> persons;
91
92  //threading
93  boost::thread_group vizageThreadGroup;
94
95  Trainer faceTrainer(config, db, log);
96  Analyzer analyzer(&persons, config, &faceTrainer, db, log, ageEstimator, genderEstimator);
97  //ContentFetcher contentFetcher(config, db, log);
98
99  log->info("Starting Content Provider Server");
100 boost::asio::io_service ioService;
101 Server server(&ioService, 1339);
102
103 vizageThreadGroup.create_thread(analyzer);
104 vizageThreadGroup.create_thread(faceTrainer);
105 vizageThreadGroup.create_thread(server);
106
107 //vizageThreadGroup.create_thread(contentFetcher);
108
109 //open new thread for communication with vizage
110
111 log->info("Opening capture");
112 //TODO: Replace with config values
113 if (capture->openCapture(new CameraResolution(atoi(config->getProperty("Capture.Width").c_str()), atoi(
        config->getProperty("Capture.Height").c_str())))){
114     log->debug("Capture opened");
115 }
116 else
117 {
118     log->error("Could not open Capture");
119     return -1;
120 }
121
122 //FPS Variables
123 time_t start, end;
124
125 double fps = 0.0;
126 int frameCount = 0;
127
128 time(&start);
129
130
131 while(true){
132
133    Mat *frame = capture->queryFrame();
134    faces = detector->getFaces(frame);
135    int lastTrackId = 0;
136
137    //short term learning is done here
138    //for each face found
139    for(size_t faceCount = 0; faceCount < faces.size(); ++faceCount){
140        //check if person is known
141        //for each person known
142        bool found = false;
143        for(size_t person = 0; person < persons.size(); person++){
144            if(persons[person]->getTrackId() == faces[faceCount].second){
145               found = true;
146               time_t seen;
147               time(&seen);
148               persons[person]->setLastSeen(seen);
149
150               if(persons[person]->getImage()->empty()){
151                   Mat croppedFace = validateRegion(frame, faces[faceCount].first, 0);
152                  // Mat(*frame, faces[person].first).copyTo(croppedFace);
153                   persons[person]->setImage(croppedFace);
154               }
155
156               break;
157            }
158            lastTrackId = faces[faceCount].second;
159        }
160
```

```
161             if(found == false){
162                 log->info("Tracking new Person: "+boost::lexical_cast<string>(faces[faceCount].second));
163 
164                 Person* p = new Person();
165                 p->setTrackId(lastTrackId);
166                 time_t seen;
167                 time(&seen);
168                 p->setLastSeen(seen);
169                 p->setFirstSeen(seen);
170 
171                 log->debug("Short Term Learning Tracking ID: "+boost::lexical_cast<string>(faces[faceCount].
                        second));
172 
173                 Mat croppedFace = validateRegion(frame, faces[faceCount].first, 0);
174 
175 
176                 p->setImage(croppedFace);
177                 log->debug("Short Term learning done for Tracking ID: "+boost::lexical_cast<string>(faces[
                        faceCount].second));
178 
179                 persons.push_back(p);
180             }
181         }
182 
183         //fancy
184         for(unsigned int i = 0; i < faces.size(); i++)
185         {
186             stringstream trackidStream;
187             trackidStream << "Track ID: "<<faces[i].second;
188 
189             stringstream recIdStream;
190             recIdStream << "ID: ";
191 
192             stringstream ageStream;
193             ageStream << "Age: ";
194 
195             stringstream genderStream;
196             genderStream << "Gender: ";
197 
198             for(size_t person = 0 ; person < persons.size(); person++){
199                 if(persons.at(person)->getTrackId() == faces[i].second){
200                     if(persons.at(person)->getRecId() == -1){
201                         recIdStream << " processing...";
202                     }else{
203                         recIdStream << persons.at(person)->getRecId();
204                     }
205 
206                     if(persons.at(person)->getAge() == 0){
207                         ageStream << "estimating...";
208                     }else{
209                         ageStream << persons.at(person)->getAge();
210                     }
211 
212                     if(persons.at(person)->getGender() == -1){
213                         genderStream << "estimating...";
214                     }else{
215                         switch(persons.at(person)->getGender()){
216                           case 0:
217                             genderStream << "Male";
218                             break;
219                           case 1:
220                             genderStream << "Female";
221                             break;
222                         }
223                     }
224 
225                     if(persons.at(person)->isWanted()){
226                         putText(*frame, "WANTED", Point(faces[i].first.x, faces[i].first.y-5),
                                CV_FONT_HERSHEY_PLAIN, 1.0, Scalar(0, 0,255 ), 1, 8);
227                         rectangle(*frame, faces[i].first, CV_RGB(255, 0,0), 1);
228                     }else{
229                         rectangle(*frame, faces[i].first, CV_RGB(0, 255,0), 1);
230                     }
231                 }
232             }
233 
234             putText(*frame, recIdStream.str(), Point(faces[i].first.x+faces[i].first.width, faces[i].first.y+
                    faces[i].first.height-5), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar(0, 255, 255), 1, 8);
235             putText(*frame, trackidStream.str(), Point(faces[i].first.x+faces[i].first.width, faces[i].first.y+
                    faces[i].first.height-19), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar(0, 255, 255), 1, 8);
236             putText(*frame, ageStream.str(), Point(faces[i].first.x+faces[i].first.width, faces[i].first.y+faces
                    [i].first.height-33), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar(0, 255, 255), 1, 8);
237             putText(*frame, genderStream.str(), Point(faces[i].first.x+faces[i].first.width, faces[i].first.y+
                    faces[i].first.height-47), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar(0, 255, 255), 1, 8);
238         }
239 
240         //FPS calculations
241         time(&end);
242         double sec = difftime(end, start);
243         ++frameCount;
244 
```

```
245         fps = frameCount / sec;
246 
247         stringstream infostream;
248         infostream << "FPS: "<< round(fps);
249 
250         if(frameCount == 10000){
251             frameCount = 1;
252         }
253 
254 
255         putText(*frame, infostream.str(), Point(10, 20), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar::all(255), 1, 8);
256 
257         stringstream shortTermFacesStream;
258         shortTermFacesStream << "Short Term Faces: " << persons.size();
259         putText(*frame, shortTermFacesStream.str(), Point(10,40), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar::all(255),
                1, 8);
260 
261         //category watcher
262 
263         int posXContentCircle = atoi(config->getProperty("Capture.Width").c_str())*0.95;
264         int posYContentCircle = atoi(config->getProperty("Capture.Height").c_str())*0.05;
265         stringstream categoryWatching;
266         categoryWatching << "Category watching: " << server.getCategoryAiring();
267         CvScalar color = CV_RGB(0,255,0);
268         if(server.getClients() > 0){
269             color = CV_RGB(0,255,0);
270         }else{
271             color = CV_RGB(255,0,0);
272         }
273 
274         circle(*frame,cvPoint(posXContentCircle,posYContentCircle),10,color,-1);
275         putText(*frame, "C", Point(posXContentCircle-5, posYContentCircle+5), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar
                ::all(255), 1, 8);
276         putText(*frame, categoryWatching.str(), Point(10,60), CV_FONT_HERSHEY_PLAIN, 1.0, Scalar::all(255), 1,
                8);
277 
278 
279         imshow("Frame", *frame);
280 
281         //clear and prepare next frame
282         faces.clear();
283 
284         if (waitKey(1) >= 0){
285             server.stop();
286             vizageThreadGroup.interrupt_all();
287             vizageThreadGroup.join_all();
288 
289             break;
290         }
291     }
292 
293     vizageThreadGroup.join_all();
294     br::Context::finalize();
295 
296 
297     return 0;
298 }
299 
300 Mat validateRegion(Mat *frame, Rect face, int percentToExtend){
301 
302     Mat croppedFaceExtended;
303 
304     float percentFactor = (float) percentToExtend/100;
305 
306     //drag x/y by percent in 45 degree
307 
308     face.x = face.x - (face.x*percentFactor);
309     face.y = face.y - (face.y*percentFactor);
310 
311     face.width = face.width+ face.width*percentFactor;
312     face.height = face.height + face.height*percentFactor;
313 
314     if(face.x <= 0){
315         face.x = 0;
316     }
317 
318     if(face.width <= 0){
319         face.width = 0;
320     }
321     //width is over the frame
322     if(face.x + face.width >= frame->cols){
323         face.width = frame->cols - face.x;
324     }
325 
326     if(face.y <= 0){
327         face.y = 0;
328     }
329     if(face.height <= 0){
330         face.height = 0;
331     }
332 
```

```
333     if(face.y+face.height >= frame->rows){
334         face.height = frame->rows - face.y;
335     }
336
337     //extend width and height vector
338     Mat(*frame, face).copyTo(croppedFaceExtended);
339
340     return croppedFaceExtended;
341 }
```

Listing 6.21: person.h

```
1  #ifndef PERSON_H
2  #define PERSON_H
3
4  #include <iostream>
5  #include <time.h>
6
7  #include <opencv2/core/core.hpp>
8
9  using namespace std;
10
11 class Person
12 {
13 public:
14     Person();
15     ~Person();
16
17     int getTrackId();
18     int getRecId();
19     time_t getLastSeen();
20     time_t getFirstSeen();
21     cv::Mat* getImage();
22     cv::Mat* getFeature();
23     void clearImage();
24     int getAge();
25     int getGender();
26     bool isWanted();
27
28     void setTrackId(int trackId);
29     void setRecId(int recId);
30     void setLastSeen(time_t lastSeen);
31     void setFirstSeen(time_t firstSeen);
32     void setImage(cv::Mat image);
33     void setAge(int age);
34     void setGender(int gender);
35     void setWanted(bool wanted);
36     void setFeature(cv::Mat *feature);
37
38 private:
39     int trackId;
40     int recId;
41     int age;
42     int gender;
43     bool wanted;
44     cv::Mat image;
45     cv::Mat feature;
46     time_t firstSeen;
47     time_t lastSeen;
48 };
49
50 #endif // PERSON_H
```

Listing 6.22: person.cpp

```
1  #include "person.h"
2
3  Person::Person()
4  {
5      trackId = -1;
6      recId = -1;
7      lastSeen = 0;
8      age = 0;
9      gender = -1;
10     wanted = 0;
11 }
12
13 Person::~Person(){
14     this->image.release();
15 }
16
17 int Person::getTrackId(){
18     return this->trackId;
19 }
20
21 int Person::getRecId(){
22     return this->recId;
```

```
23  }
24
25  time_t Person::getLastSeen(){
26      return this->lastSeen;
27  }
28
29  time_t Person::getFirstSeen(){
30      return this->firstSeen;
31  }
32
33  cv::Mat *Person::getImage(){
34      return &this->image;
35  }
36
37  int Person::getAge(){
38      return this->age;
39  }
40
41  int Person::getGender(){
42      return this->gender;
43  }
44
45  cv::Mat* Person::getFeature(){
46      return &this->feature;
47  }
48
49  void Person::setTrackId(int trackId){
50      this->trackId = trackId;
51  }
52
53  void Person::setRecId(int recId){
54      this->recId = recId;
55  }
56
57  void Person::setLastSeen(time_t lastSeen){
58      this->lastSeen = lastSeen;
59  }
60
61  void Person::setFirstSeen(time_t firstSeen){
62      this->firstSeen = firstSeen;
63  }
64
65  void Person::setImage(cv::Mat image){
66      this->image.release();
67      this->image = image;
68  }
69
70
71  void Person::clearImage(){
72      cv::Mat newImage;
73      this->setImage(newImage);
74  }
75
76  void Person::setAge(int age){
77      this->age = age;
78  }
79
80  void Person::setGender(int gender){
81      this->gender = gender;
82  }
83
84  bool Person::isWanted(){
85      return this->wanted;
86  }
87
88  void Person::setWanted(bool wanted){
89      this->wanted = wanted;
90  }
91
92  void Person::setFeature(cv::Mat* feature){
93      this->feature.release();
94      this->feature = *feature;
95  }
```

Listing 6.23: index.php

```
1   <?php
2
3   include('app/config.php');
4   include('app/controller/Controller.php');
5   include('app/controller/IndexController.php');
6   include('app/controller/FacesController.php');
7   include('app/controller/FaceController.php');
8   include('app/controller/StatisticsController.php');
9   include('app/model/User.php');
10  include('app/model/FaceEntity.php');
11  include('app/model/FaceEntityLoader.php');
12  include('app/model/LoginManager.php');
13  include('app/model/Database.php');
```

```
14 include('app/view/View.php');
15 include('app/view/JSONView.php');
16
17 date_default_timezone_set('UTC');
18
19 $request = array_merge($_GET, $_POST);
20 //controller zum routen nutzen
21 $controller = new Controller($request);
22 echo $controller->display();
23
24 ?>
```

Listing 6.24: Controller.php

```
1  <?php
2
3  class Controller{
4
5          private $request;
6          private $template;
7          private $view;
8          private $loginManager;
9          private $database;
10         private $outputFormat;
11
12         public function __construct($request){
13
14                 $this->database = new Database(DBHOST, DBNAME, DBUSER, DBPASS);
15                 $this->loginManager = new LoginManager("Vizage_web", $this->database);
16                 $this->request = $request;
17                 $this->template = !empty($request['view']) ? $request['view'] : 'index';
18                 $this->outputFormat = !empty($request['output']) ? $request['output'] : 'html';
19                 $this->loginManager->secureSessionStart();
20         }
21
22         public function display(){
23                 switch(strtoupper($this->outputFormat)){
24                         case "HTML":
25                                 $this->view = new View();
26                                 return $this->displayHTML();
27                                 break;
28                         case "JSON":
29                                 $this->view = new JSONView();
30                                 return $this->displayJSON();
31                                 break;
32                         default:
33                                 return $this->displayHTML();
34                                 break;
35                 }
36         }
37
38         private function displayHTML(){
39
40                 $contentView = new View();
41
42                 //check for timeout
43                 if($this->loginManager->isLoggedIn()){
44                         //check for timeout
45                         if($this->loginManager->timeout()){
46                                 //redirect to index with warning,
47                                 //update settings for avoiding redirect loop
48                                 $this->loginManager->logout();
49                                 $this->view->assign("warning", "loggedout");
50                                 $this->view->assign("warning-message-strong", "You ");
51                                 $this->view->assign("warning-message", "have timed out! Please login again.");
52                                 $this->template = "index";
53                         }else{
54                                 $this->loginManager->updateLastAction();
55                         }
56                 }
57
58
59                 switch($this->template){
60                         case "index":
61                                 $indexController = new IndexController($this->loginManager, $this->request,
                                        $this->view);
62                                 $indexController->display($contentView);
63                                 break;
64
65                         case "faces":
66                                 $facesController = new FacesController($this->loginManager,  $this->request,
                                        $this->view);
67                                 $facesController->display($contentView);
68                                 break;
69                         case "face":
70                                 $faceController = new FaceController($this->loginManager, $this->request,
                                        $this->view);
71                                 $faceController->display($contentView);
72                                 break;
```

```
73                      case "statistics":
74                              $statisticsController = new StatisticsController($this->loginManager, $this->
                                    request, $this->view);
75                              $statisticsController->display($contentView);
76                              break;
77                      default:
78                              $indexController = new IndexController($this->loginManager,  $this->request,
                                    $this->view);
79                              $indexController->display($contentView);
80                              break;
81
82              }
83
84              $this->view->setTemplate("header");
85              $this->view->assign('loggedin', $this->loginManager->isLoggedIn());
86              $this->view->assign("content", $contentView->loadTemplate());
87
88
89              return $this->view->loadTemplate();
90      }
91
92      private function displayJSON(){
93
94              //check for timeout
95              if($this->loginManager->isLoggedIn()){
96                      //check for timeout
97                      if($this->loginManager->timeout()){
98                              //redirect to index with warning,
99                              //update settings for avoiding redirect loop
100                             $this->loginManager->logout();
101                             $this->template = "index";
102                     }else{
103                             $this->loginManager->updateLastAction();
104                     }
105             }
106
107             switch($this->template){
108                     case "face":
109                             $this->view->setTemplate("action");
110
111                             switch($this->request['action']){
112                                     case "saveFace":
113                                             $faceController = new FaceController($this->
                                                    loginManager, $this->request, $this->view);
114
115                                             $gender = $faceController->switchGender($this->request
                                                    ['id'], $this->request['gender']);
116                                             $firstname = $faceController->updateFirstName($this->
                                                    request['id'], $this->request['firstname']);
117                                             $lastname = $faceController->updateLastName($this->
                                                    request['id'], $this->request['lastname']);
118                                             $age = $faceController->updateAge($this->request['id'
                                                    ], $this->request['age']);
119                                             $wanted = $faceController->updateWanted($this->request
                                                    ['id'], $this->request['wanted']);
120
121                                             if($gender["error"] == 1){
122                                                     $this->view->assign("action", json_encode(
                                                            array("error" => 1, "errorDescription" =>
                                                             "could not set gender")));
123                                             }else if($firstname["error"] == 1){
124                                                     $this->view->assign("action", json_encode(
                                                            array("error" => 2, "errorDescription" =>
                                                             "could not set firstname")));
125                                             }else if($lastname["error"] == 1){
126                                                     $this->view->assign("action", json_encode(
                                                            array("error" => 3, "errorDescription" =>
                                                             "could not set lastname")));
127                                             }else if($age["error"] == 1){
128                                                     $this->view->assign("action", json_encode(
                                                            array("error" => 4, "errorDescription" =>
                                                             "could not set age")));
129                                             }else{
130                                                     $this->view->assign("action", json_encode(
                                                            array("error" => 0, "errorDescription" =>
                                                             "")));
131                                             }
132
133                                             return $this->view->loadTemplate();
134                                     break;
135                             case "toggleLock":
136                                     $faceController = new FaceController($this->loginManager,
                                            $this->request, $this->view);
137                                     $this->view->assign('action', json_encode($faceController->
                                            toggleLockFace($this->request['id'])));
138                                     return $this->view->loadTemplate();
139                                     break;
140                             case "toggleWanted":
141                                     $faceController = new FaceController($this->loginManager,
                                            $this->request, $this->view);
```

```
142                                             $this->view->assign('action', json_encode($faceController->
                                                    toggleWantedFace($this->request['id'])));
143                                             return $this->view->loadTemplate();
144                                             break;
145                                     case "getchartdata":
146                                             $faceController = new FaceController($this->loginManager,
                                                    $this->request, $this->view);
147                                             $this->view->assign("action", json_encode($faceController->
                                                    getChartData($this->request['id'])));
148                                             return $this->view->loadTemplate();
149                                             break;
150                                     default:
151                                             //no action is required
152                                             break;
153                                 }
154
155
156
157                 }
158
159
160
161         }
162 }
163
164 ?>
```

Listing 6.25: FaceController.php

```
1  <?php
2
3  class FaceController{
4
5          private $view;
6          private $request;
7          private $loginManager;
8          private $faceEntityLoader;
9
10         public function __construct($loginManager,  $request,$view){
11                 $this->view = $view;
12                 $this->request = $request;
13                 $this->loginManager = $loginManager;
14                 $this->faceEntityLoader = new FaceEntityLoader($this->loginManager->getDatabase());
15         }
16
17         public function display($contentView){
18
19                 $contentView->setTemplate("face");
20                 $loggedin = $this->loginManager->isLoggedIn();
21                 $contentView->assign('loggedin', $loggedin);
22
23                 if($loggedin){
24                         $contentView->assign("id", $this->request['id']);
25                         $face = $this->faceEntityLoader->loadFace($this->request['id']);
26                         $contentView->assign("face", $face);
27                         $contentView->assign("locked", $this->faceEntityLoader->isLocked($this->request['id'])
                                );
28                         $contentView->assign("wanted", $this->faceEntityLoader->isWanted($this->request['id'])
                                );
29                 }
30         }
31
32         public function switchGender($id, $gender){
33                 if($this->faceEntityLoader->switchGender($id, $gender)){
34                         return array("error" => 0);
35                 }else{
36                         return array("error" => 1);
37                 }
38         }
39
40         public function updateFirstName($id, $name){
41                 if($this->faceEntityLoader->setFirstName($id, $name)){
42                         return array("error" => 0);
43                 }else{
44                         return array("error" => 1);
45                 }
46         }
47
48         public function updateLastName($id, $name){
49                 if($this->faceEntityLoader->setLastName($id, $name)){
50                         return array("error" => 0);
51                 }else{
52                         return array("error" => 1);
53                 }
54         }
55
56         public function updateAge($id, $age){
57                 if($this->faceEntityLoader->setAge($id, $age)){
58                         return array("error" => 0);
```

```
59                  }else{
60                          return array("error" => 1);
61                  }
62          }
63
64          public function updateWanted($id, $wanted){
65                  if($this->faceEntityLoader->setWanted($id, $wanted)){
66                          return array("error" => 0);
67                  }else{
68                          return array("error" => 1);
69                  }
70          }
71
72          public function toggleLockFace($id){
73                  if($this->faceEntityLoader->toggleLockFace($id)){
74                          return array("error" => 0, "locked" => 1);
75                  }else{
76                          return array("error" => 0, "locked" => 0);
77                  }
78          }
79
80          public function toggleWantedFace($id){
81                  if($this->faceEntityLoader->toggleWantedFace($id)){
82                          return array("error" => 0, "wanted" => 1);
83                  }else{
84                          return array("error" => 0, "wanted" => 0);
85                  }
86          }
87
88          public function getChartData($id){
89                  $interval = $this->faceEntityLoader->getVisits($id);
90                  return $interval;
91          }
92
93  }
94  ?>
```

Listing 6.26: IndexController.php

```
1   <?php
2
3   class IndexController extends Controller{
4
5           private $loginManager;
6           private $view;
7           private $request;
8
9           public function __construct($loginManager, $request,$view){
10                  $this->loginManager = $loginManager;
11                  $this->view = $view;
12                  $this->request = $request;
13          }
14
15          public function display($contentView){
16
17                  $contentView->setTemplate("index");
18                  $contentView->assign("signedup", false);
19
20                  if(isset($this->request['action'])){
21                          switch($this->request['action']){
22                                  case 'login':
23                                          $this->loginManager->login($this->request['email-login'], $this->
                                                request['password-login']);
24                                          $contentView->assign('loggedin', $this->loginManager->isLoggedIn());
25                                          $this->loginManager->updateLastAction();
26                                          break;
27                                  case 'signup':
28                                          $this->loginManager->signup();
29                                          $contentView->assign("signedup", true);
30                                          break;
31                                  case 'logout':
32                                          $this->loginManager->logout();
33                                          $this->loginManager->updateLastAction();
34                                          $contentView->assign('loggedin', $this->loginManager->isLoggedIn());
35                                          $this->view->assign("warning", "loggedout");
36                                          $this->view->assign("warning-message-strong", "You ");
37                                          $this->view->assign("warning-message", "have logged out");
38                                          break;
39                          }
40                  }
41
42                  $loggedin = $this->loginManager->isLoggedIn();
43                  $contentView->assign('loggedin', $loggedin);
44
45                  if($loggedin){
46
47                          $faceLoader = new FaceEntityLoader($this->loginManager->getDatabase());
48                          $contentView->assign('face-count', $faceLoader->getFaceCount());
49                          $contentView->assign('image-count', $faceLoader->getImageCount());
```

```
50                          $contentView->assign('men-count', $faceLoader->getMenCount());
51                          $contentView->assign('female-count', $faceLoader->getFemaleCount());
52
53                          $faces = $faceLoader->loadAllFaces();
54                          $contentView->assign('faces', $faces);
55                          //load faces
56
57                  }
58
59          }
60  }
61  ?>
```

Listing 6.27: Database.php

```
1   <?php
2
3   class Database{
4
5           private $connection;
6
7           public function __construct($dbhost, $dbname, $dbuser, $dbpass){
8                   $this->connection = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
9           }
10
11          public function getConnection(){
12                  return $this->connection;
13          }
14  }
15
16  ?>
```

Listing 6.28: FaceEntity.php

```
1   <?php
2
3   class FaceEntity{
4
5           private $id;
6           private $firstName;
7           private $lastName;
8           private $firstSeen;
9           private $lastSeen;
10          private $gender;
11          private $age;
12          private $address;
13          private $images;
14          private $isWanted;
15
16          public function __construct($id, $firstName, $lastName, $firstSeen, $lastSeen, $gender, $age, $address
                , $wanted ,$images = array()){
17                  $this->id = $id;
18                  $this->firstName = $firstName;
19                  $this->lastName = $lastName;
20                  $this->firstSeen = $firstSeen;
21                  $this->lastSeen = $lastSeen;
22                  $this->gender = $gender;
23                  $this->age = $age;
24                  $this->address = $address;
25                  $this->isWanted = $wanted;
26                  $this->images = $images;
27          }
28
29
30          //plain getter and setter
31          public function getId(){
32                  return $this->id;
33          }
34
35          public function getFirstName(){
36                  return $this->firstName;
37          }
38
39          public function setFirstName($firstName){
40                  $this->firstName = $firstName;
41          }
42
43          public function getLastName(){
44                  return $this->lastName;
45          }
46
47          public function setLastName($lastName){
48                  $this->lastName = $lastName;
49          }
50
51          public function getFirstSeen(){
52                  return $this->firstSeen;
```

```
53          }
54
55          public function getImages(){
56                  return $this->images;
57          }
58
59          public function setFirstSeen($firstSeen){
60                  $this->firstSeen = $firstSeen;
61          }
62
63          public function getLastSeen(){
64                  return $this->lastSeen;
65          }
66
67          public function setLastSeen($lastSeen){
68                  $this->lastSeen = $lastSeen;
69          }
70
71          public function getGender(){
72                  return $this->gender;
73          }
74
75          public function setGender($gender){
76                  $this->gender = $gender;
77          }
78
79          public function getAge(){
80                  return $this->age;
81          }
82
83          public function setAge($age){
84                  $this->age = $age;
85          }
86
87          public function setImages($images){
88                  $this->images = $images;
89          }
90
91          public function setAddress($address){
92                  $this->address = $address;
93          }
94
95          public function getAddress(){
96                  return $this->address;
97          }
98
99          public function isWanted(){
100                 return $this->isWanted;
101         }
102
103         public function setWanted($wanted){
104                 $this->isWanted = $wanted;
105         }
106 }
107 ?>
```

Listing 6.29: FaceEntityLoader.php

```
1   <?php
2
3   class FaceEntityLoader{
4
5           private $database;
6
7           public function __construct($database){
8                   $this->database = $database;
9           }
10
11          public function loadAllFaces(){
12                  $faceEntities = array();
13
14                  if($stmt = $this->database->getConnection()->prepare("SELECT id, first_seen, last_seen,
                        address, age, gender, prename, lastname, wanted FROM people ORDER BY last_seen DESC")){
15
16                          $stmt->execute();
17
18                          $stmt->bind_result($id, $first_seen, $last_seen, $address, $age, $gender, $prename,
                                $lastname, $wanted);
19                          while($stmt->fetch()){
20                                  $face = new FaceEntity($id, $prename, $lastname, $first_seen, $last_seen,
                                        $gender, $age, $address, $wanted, array());
21                                  $faceEntities[] = $face;
22                          }
23                          $stmt->close();
24                  }
25
26
27                  foreach($faceEntities as $face){
28                          $images = array();
```

```
29                      if($stmtFaceImages = $this->database->getConnection()->prepare("SELECT image FROM
                            people_information WHERE pid = ?")){
30                          $stmtFaceImages->bind_param('i', $face->getId());
31                          $stmtFaceImages->execute();
32
33                          $stmtFaceImages->bind_result($image);
34                          while($stmtFaceImages->fetch()){
35                                  $images[] = base64_encode(file_get_contents(FACEPATH.$image));
36                          }
37                      }
38                      $face->setImages($images);
39              }
40
41              return $faceEntities;
42      }
43
44      public function loadFace($id){
45
46              if($stmt = $this->database->getConnection()->prepare("SELECT first_seen, last_seen, address,
                    age, gender, prename, lastname, wanted FROM people WHERE id = ".$id." LIMIT 1")){
47
48                      $stmt->execute();
49
50                      $stmt->bind_result($first_seen, $last_seen, $address, $age, $gender, $prename,
                            $lastname, $wanted);
51                      $stmt->fetch();
52                      $face = new FaceEntity($id, $prename, $lastname, $first_seen, $last_seen, $gender,
                            $age, $address, $wanted, array());
53
54                      $stmt->close();
55
56
57                      //fetch images for that face
58                      $images = array();
59                      if($stmtFaceImages = $this->database->getConnection()->prepare("SELECT image FROM
                            people_information WHERE pid = ?")){
60                          $stmtFaceImages->bind_param('i', $id);
61                          $stmtFaceImages->execute();
62
63                          $stmtFaceImages->bind_result($image);
64                          while($stmtFaceImages->fetch()){
65                                  $images[] = base64_encode(file_get_contents(FACEPATH.$image));
66                          }
67                      }
68                      $face->setImages($images);
69
70                      return $face;
71              }else{
72                      return false;
73              }
74
75      }
76
77      public function getFaceCount(){
78
79              if($stmt = $this->database->getConnection()->prepare("SELECT COUNT(id) as count FROM people
                    LIMIT 1")){
80                      $stmt->execute();
81                      $stmt->store_result();
82                      $stmt->bind_result($count);
83                      $stmt->fetch();
84              }
85
86              return $count;
87      }
88
89      public function getImageCount(){
90              if($stmt = $this->database->getConnection()->prepare("SELECT COUNT(id) as count FROM
                    people_information LIMIT 1")){
91                      $stmt->execute();
92                      $stmt->store_result();
93                      $stmt->bind_result($count);
94                      $stmt->fetch();
95              }
96              return $count;
97      }
98
99      public function getMenCount(){
100             if($stmt = $this->database->getConnection()->prepare("SELECT COUNT(id) FROM people WHERE
                    gender = 0 LIMIT 1")){
101                     $stmt->execute();
102                     $stmt->store_result();
103                     $stmt->bind_result($count);
104                     $stmt->fetch();
105             }
106             return $count;
107     }
108
109     public function getFemaleCount(){
110             if($stmt = $this->database->getConnection()->prepare("SELECT COUNT(id) FROM people WHERE
                    gender = 1 LIMIT 1")){
```

```
111                     $stmt->execute();
112                     $stmt->store_result();
113                     $stmt->bind_result($count);
114                     $stmt->fetch();
115             }
116             return $count;
117     }
118
119     public function switchGender($id, $gender){
120             if($stmt = $this->database->getConnection()->prepare("UPDATE people SET gender = ? WHERE id =
                    ? AND locked = 0")){
121                     $stmt->bind_param('ii', $gender, $id);
122
123                     $stmt->execute();
124                     return true;
125             }else{
126                     return false;
127             }
128     }
129
130     public function setFirstName($id, $name){
131             if($stmt = $this->database->getConnection()->prepare("UPDATE people SET prename = ? WHERE id =
                    ? AND locked = 0")){
132                     $stmt->bind_param('si', $name, $id);
133                     $stmt->execute();
134                     return true;
135             }else{
136                     return false;
137             }
138     }
139
140     public function setLastName($id, $name){
141             if($stmt = $this->database->getConnection()->prepare("UPDATE people SET lastname = ? WHERE id
                    = ? AND locked = 0")){
142                     $stmt->bind_param('si', $name, $id);
143                     $stmt->execute();
144                     return true;
145             }else{
146                     return false;
147             }
148     }
149
150     public function setAge($id, $age){
151
152             if($stmt = $this->database->getConnection()->prepare("UPDATE people SET age = ? WHERE id = ?
                    AND locked = 0")){
153                     $stmt->bind_param('ii', $age, $id);
154                     $stmt->execute();
155
156                     return true;
157             }else{
158                     return false;
159             }
160     }
161
162     public function setWanted($id, $wanted){
163             if($stmt = $this->database->getConnection()->prepare("UPDATE people SET wanted = ? WHERE id =
                    ? AND locked = 0")){
164                     $stmt->bind_param('ii', $wanted, $id);
165                     $stmt->execute();
166
167                     return true;
168             }else{
169                     return false;
170             }
171     }
172
173     public function toggleLockFace($id){
174
175             if($stmt = $this->database->getConnection()->prepare("SELECT locked FROM people WHERE id = ?")
                    ){
176                     $stmt->bind_param('i', $id);
177                     $stmt->execute();
178                     $stmt->store_result();
179                     $stmt->bind_result($locked);
180                     $stmt->fetch();
181             }
182
183             if($locked == 0){
184                     if($stmt = $this->database->getConnection()->prepare("UPDATE people SET locked = 1
                            WHERE id = ?")){
185                             $stmt->bind_param('i', $id);
186                             $stmt->execute();
187                             return true;
188                     }
189             }else{
190                     if($stmt = $this->database->getConnection()->prepare("UPDATE people SET locked = 0
                            WHERE id = ?")){
191                             $stmt->bind_param('i', $id);
192                             $stmt->execute();
193                             return false;
```

```
194                         }
195                 }
196
197         }
198
199         public function toggleWantedFace($id){
200                 if($stmt = $this->database->getConnection()->prepare("SELECT wanted FROM people WHERE id = ?")
                        ){
201                         $stmt->bind_param('i', $id);
202                         $stmt->execute();
203                         $stmt->store_result();
204                         $stmt->bind_result($wanted);
205                         $stmt->fetch();
206                 }
207
208                 if($wanted == 0){
209                         if($stmt = $this->database->getConnection()->prepare("UPDATE people SET wanted = 1
                                WHERE id = ?")){
210                                 $stmt->bind_param('i', $id);
211                                 $stmt->execute();
212                                 return true;
213                         }
214                 }else{
215                         if($stmt = $this->database->getConnection()->prepare("UPDATE people SET wanted = 0
                                WHERE id = ?")){
216                                 $stmt->bind_param('i', $id);
217                                 $stmt->execute();
218                                 return false;
219                         }
220                 }
221
222         }
223
224
225         public function isLocked($id){
226                 if($stmt = $this->database->getConnection()->prepare("SELECT locked FROM people WHERE id = ?
                        LIMIT 1")){
227                         $stmt->bind_param('i', $id);
228                         $stmt->execute();
229                         $stmt->store_result();
230                         $stmt->bind_result($locked);
231                         $stmt->fetch();
232                 }
233                 if($locked == 0){
234                         return false;
235                 }else{
236                         return true;
237                 }
238         }
239
240         public function isWanted($id){
241                 if($stmt = $this->database->getConnection()->prepare("SELECT wanted FROM people WHERE id = ?
                        LIMIT 1")){
242                         $stmt->bind_param('i', $id);
243                         $stmt->execute();
244                         $stmt->store_result();
245                         $stmt->bind_result($wanted);
246                         $stmt->fetch();
247                 }
248                 if($wanted == 0){
249                         return false;
250                 }else{
251                         return true;
252                 }
253         }
254
255         public function getVisits($id){
256
257                 $interval = array();
258
259                 if($stmtFaceImages = $this->database->getConnection()->prepare("SELECT DATE(interval_start) as
                        date, COUNT(*) total FROM people_interval WHERE pid = ? GROUP BY DATE(interval_start)")
                        ){
260                                 $stmtFaceImages->bind_param('i', $id);
261                                 $stmtFaceImages->execute();
262
263                                 $stmtFaceImages->bind_result($date, $total);
264                                 while($stmtFaceImages->fetch()){
265                                         $interval[] = array($date, $total);
266                                 }
267
268                 }
269                 return $interval;
270         }
271
272
273 }
274 ?>
```

Listing 6.30: LoginManager.php

```
1  <?php
2
3  class LoginManager{
4
5          private $sessionName;
6          private $database;
7
8          public function __construct($sessionName, $database){
9                  $this->sessionName = $sessionName;
10                 $this->database = $database;
11         }
12
13         public function getDatabase(){
14                 return $this->database;
15         }
16
17         public function secureSessionStart(){
18                 session_start();
19                 session_regenerate_id();
20         }
21
22         public function isLoggedIn(){
23                 if(isset($_SESSION['loggedin'])){
24                         return $_SESSION['loggedin'];
25                 }else{
26                         return false;
27                 }
28         }
29
30         public function login($email, $password){
31                 if($stmt = $this->database->getConnection()->prepare("SELECT id, username, password, name,
                        lastname, email FROM members WHERE email = ? LIMIT 1")){
32                         $stmt->bind_param('s', $email);
33                         $stmt->execute();
34                         $stmt->store_result();
35                         $stmt->bind_result($userid ,$username, $dbPassword, $name, $lastname, $email);
36                         $stmt->fetch();
37                 }
38
39                 if($stmt->num_rows == 1){
40                         if($dbPassword == $password){
41                                 $_SESSION['loggedin'] = true;
42                                 //create new user
43                                 $user = new User($userid, $name, $username, $lastname, $email);
44                                 $_SESSION['User'] = serialize($user);
45                         }else{
46                                 $_SESSION['loggedin'] = false;
47                         }
48                 }
49         }
50
51         public function logout(){
52                 $_SESSION['loggedin'] = false;
53         }
54
55         public function signup(){
56
57                 $this->database->getConnection()->query("INSERT INTO members(`username`,`email`, `name`, `
                        lastname`, `password`) VALUES
58 ('".$_POST['username-signup'])."', '".$_POST['email-signup'])."', '".$_POST['
       firstname-signup'])."', '".$_POST['lastname-signup'])."', '".$_POST['password-signup'])."')");
59                 return true;
60         }
61
62         public function timeout(){
63
64                 if(isset($_SESSION['User']) && isset($_SESSION['loggedin'])){
65
66                         if($_SESSION['loggedin'] == true){
67                                 //check for last action
68                                 $user = unserialize($_SESSION['User']);
69                                 if($stmt = $this->database->getConnection()->prepare("SELECT (lastaction - NOW
                                        ()) as difftime FROM members WHERE id = ? LIMIT 1")){
70                                         $stmt->bind_param('i', $user->getId());
71                                         $stmt->execute();
72                                         $stmt->store_result();
73                                         $stmt->bind_result($lastaction);
74                                         $stmt->fetch();
75                                 }
76
77                                 if(abs($lastaction) > TIMEOUT){
78                                         return true;
79                                 }else{
80                                         return false;
81                                 }
82
83                         }else{
84                                 return false;
85                         }
```

```
86                  }
87                  else{
88                          return false;
89                  }
90          }
91
92          public function updateLastAction(){
93                  if(isset($_SESSION['User']) && isset($_SESSION['loggedin'])){
94
95                          if($_SESSION['loggedin'] == true){
96                                  //check for last action
97                                  $user = unserialize($_SESSION['User']);
98                                  if($stmt = $this->database->getConnection()->prepare("UPDATE members SET
                                        lastaction = NOW() WHERE id = ?")){
99                                          $stmt->bind_param('i', $user->getId());
100                                         $stmt->execute();
101                                 }
102                         }
103                 }
104         }
105
106 }
107
108 ?>
```

Listing 6.31: User.php

```
1   <?php
2
3   class User{
4
5           private $id;
6           private $name;
7           private $lastname;
8           private $login;
9           private $email;
10
11          public function __construct($id, $name, $login, $lastname, $email){
12                  $this->name = $name;
13                  $this->login = $login;
14                  $this->lastname = $lastname;
15                  $this->email = $email;
16                  $this->id = $id;
17          }
18
19          public function getId(){
20                  return $this->id;
21          }
22
23          public function getName(){
24                  return $this->name;
25          }
26
27          public function setName($name){
28                  $this->name = name;
29          }
30
31          public function getLogin(){
32                  return $this->login;
33          }
34
35          public function setLogin($login){
36                  $this->login = login;
37          }
38
39
40          public function getLastName(){
41                  return $this->lastname;
42          }
43
44          public function setLastName($lastname){
45                  $this->lastname = $lastname;
46          }
47
48          public function getEmail(){
49                  return $this->email;
50          }
51
52          public function setEmail($email){
53                  $this->email = $email;
54          }
55  }
56
57  ?>
```

Glossar

High-Level Verarbeitung abstrahiert die Verarbeitung von Daten. Daten liegen nicht als Bits, Bytes oder Signale vor sondern sind gekapselt in Datenstrukturen. Die Datenstrukturen abstrahieren die darunterliegende Repräsentation. Oft werden bei der High Level Verarbeitung Methoden der künstlichen Intelligenz angewandt. In der Bildverarbeitung wird versucht durch eine High Level Verarbeitung die visuelle Wahrnehmung von Menschen zu imitieren.

Low-Level Verarbeitung hat normalerweise kein Wissen über den Inhalt oder den Kontext der Daten bzw. des Bildes.

SIFT steht für Scale-Invariant Feature Transform und beschreibt den von David Lowe vorgestellen Algorithmus zur Bestimmung von Merkmalen in Bildern [14].

SURF steht für Speeded up Robust Features und beschreibt die Erweiterung des SIFT Algorithmus. Der SURF Algorithmus wurde von Herbert Bay, Andreas Ess, Tinne Tuytelaars und Luc Van Gool im Jahr 2006 zum ersten Mal vorgestellt [1].

Literaturverzeichnis

[1] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

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

[3] Boser, Guyon, and Vapnik. A training algorithm for optimal margin classifiers. *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[4] Burbeck and Steve. Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC), 1992. Zugriffsdatum: 2014.04.13. URL: `http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html`.

[5] F. C. Crow. Summed-Area Tables for Texture Mapping. *Computer Graphics*, 18(3), Juli 1984.

[6] D. A. R. P. A. (DARPA). The Facial Recognition Technology (FERET) Database. Zugriffsdatum: 19.04.2014. URL: `http://www.itl.nist.gov/iad/humanid/feret/feret_master.html`.

[7] S. Das. Histogram Processing. Zugriffsdatum: 02.01.2014. URL: `http://www.cse.iitm.ac.in/~vplab/courses/CV_DIP/PDF/HIST_PROC.pdf`.

[8] G. Du, F. Su, and A. Cai. Face Recognition using SURF features, 2009. Zugriffsdatum: 10.04.2014. URL: `http://robotics.csie.ncku.edu.tw/HCI_Project_2009/Face_recognition_using_SURF_features.pdf`.

[9] Gamma, Helm, Johnson, and Vlissides. *Design Pattern: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[10] M. Hoffman. Support Vector Machines - Kernel and the Kernel Trick, Juni 2006. Zugriffsdatum: 03.01.2014. URL: `http:`

//www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf.

[11] Itseez. OpenCV (Open Source Computer Vision). Zugriffsdatum: 2014.03.11. URL: http://opencv.org/.

[12] M. I. Jordan. Advanced Topics in Learning & Decision Making :The Kernel Trick, 2014. Zugriffsdatum: 03.02.2014. URL: http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf.

[13] J. Klontz, B. Klare, S. Klum, A. Jain, and M. Burge. Open Source Biometric Recognition. In *Proceedings of the IEEE Conference on Biometrics: Theory, Applications and Systems*, 2013. Zugriffsdatum: 26.12.2013. URL: http://openbiometrics.org/publications/klontz2013open.pdf.

[14] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints, Januar 2004. Zugriffsdatum: 13.11.2013. URL: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf.

[15] J. Matas and J. Sochman. Adaboost. Zugriffsdatum: 12.03.2014. URL: http://www.robots.ox.ac.uk/~az/lectures/cv/adaboost_matas.pdf.

[16] Microsoft. Model-View-Controller. Zugriffsdatum: 11.04.2014. URL: http://msdn.microsoft.com/en-us/library/ff649643.aspx.

[17] B. Moghaddam and M.-H. Yang. Learning Gender with Support Faces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24, 2002.

[18] opencv dev team. Face Detection using Haar Cascades. Zugriffsdatum: 24.01.2014. URL: http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html.

[19] A. Schahidi. Gesichts-Scanner in Rotterdamer Straßenbahn, August 2010. Zugriffsdatum: 23.11.2013. URL: http://www.rp-online.de/politik/gesichts-scanner-in-rotterdamer-strassenbahn-aid-1.97734.

[20] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3, 1991.

[21] A. Uhl. Grundlagen Bildverarbeitung, 2005/2006. Zugriffsdatum: 02.01.2014. URL: http://www.cosy.sbg.ac.at/~uhl/imgProcess.pdf.

[22] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features, 2001. Zugriffsdatum: 20.01.2014. URL: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

[23] J. D. Wegner, S. Walk, M. Havlena, and K. Schindler. Bildverarbeitung: Einführung. Folien, 2013. Zugriffsdatum: 02.01.2014. URL: http://www.igp.ethz.ch/photogrammetry/education/lehrveranstaltungen/BildverarbeitungFS2013/coursematerial/1_BV_Einfuehrung_FS2013.pdf.

[24] Wikipedia. Boost (C++ libraries). Zugriffsdatum: 03.03.2014. URL: http://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries).

[25] Wikipedia. Support Vector Machine. Zugriffsdatum 03.01.2014. URL: http://de.wikipedia.org/wiki/Support_Vector_Machine.

[26] H. M. Winkels. Biometrische Verfahren, Oktober 2004. Zugriffsdatum: 2014.01.15. URL: http://www1.logistik.fh-dortmund.de/IT-Sicherheit/20_BiometrischeVerfahren.pdf.

[27] C. Wolter. Gesichtserkennung bei Facebook, Juni 2011. Zugriffsdatum: 23.11.2013. URL: http://bit.ly/1iiUP3u.

[28] S. Zhang and M. Turk. Eigenfaces. Zugriffsdatum 22.03.2014. URL: http://www.scholarpedia.org/article/Eigenfaces.