Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart

Bachelorarbeit

Bartentfernung mittels robuster Statistik und PCA

Christian Schreiner

Studiengang:	Medieninformatik
Prüfer/in:	Prof. Dr. -Ing Andres Bruhn
Betreuer/in:	Prof. Dr. -Ing Andres Bruhn
Beginn am:	9. April 2019
Beendet am:	9. November 2019

Kurzfassung

Der Bart ist ein sekundäres Geschlechtsmerkmal, welches das Aussehen des Gesichts stark beeinflusst. Deshalb stellt sich vielen die Frage, ob sie einen Bart tragen sollen oder nicht. Diese Arbeit verfolgt das Ziel jedem die Beantwortung dieser Frage leichter zu machen. Dazu werden drei Algorithmen implementiert, deren Ergebnisse unterschiedlich gut sind. Die Grundidee aller Algorithmen ist, das bärtige Gesicht in einem Unterraum aus bartlosen Gesichtern zu rekonstruieren. Dieser Unterraum wird aus Bildern von bartlosen Gesichtern konstruiert. Der erste Algorithmus nutzt zur Rekonstruktion einen einfachen Least-Squares-Fit. Der Zweite nutzt zur Lösung des Problems robuste statistische Methoden. Der dritte Algorithmus teilt das Bild in mehrere Schichten auf. Eine dieser Schichten ist die Bartschicht, welche dann einfach aus dem Bild entfernt werden kann. Zur Berechnung dieser Schicht wird zuerst ein zweiter Unterraum konstruiert. Dieser beschreibt nur Bärte. Dann wird das Originalbild in einer Kombination beider Unterräume mit einem Least-Squares-Fit rekonstruiert. Da die Konstruktion eines reinen Bartraums im Gegensatz zur Konstruktion eines Gesichtsraums ein kompliziertes Problem ist, welches die Qualität der berechneten Bartschicht stark beeinflusst, wird hier ebenso darauf eingegangen. Um das zu erreichen, wird eine Principal-Component-Analysis durchgeführt. Da das nicht ausreicht, um gute Ergebnisse zu erzielen, werden zusätzlich verschiedenen Entrauschungsverfahren auf die einzelnen Bilder angewendet. Das führt zu einer starken Verbesserung der Ergebnisse.

Inhaltsverzeichnis

Einleitung

Seit Tausenden von Jahren stellt sich der Mensch eine wichtige Frage: „Bart oder nicht Bart?“. Wenn man dieser Frage aus geschichtlicher Sicht nähern will, stößt man je nach Kultur und Epoche auf verschiedene Trends. Während sich Steinzeitmenschen vermutlich wenig Gedanken über diese Frage machten, trugen die Pharaonen Zeremonialbärte als Zeichen von Macht. Sie rasierten sich allerdings ihre echten Bärte ab, was einen nur zu dem Entschluss kommen lässt, dass sie auch keine Antwort auf diese Frage hatten. Griechische Philosophen trugen Vollbärte als Zeichen ihrer Weisheit. Die Römer sahen Bartträger als unzivilisierte Barbaren. So zieht es sich durch die Geschichte: Mal ist er im Trend, mal ist er es nicht. Man kommt unweigerlich zu dem Entschluss, dass es wohl jeder für sich selbst entscheiden muss und es keine allgemeingültige Antwort gibt.

Das Ziel dieser Bachelorarbeit ist es, aber jedem diese Entscheidung immerhin zu vereinfachen, indem eine Möglichkeit geboten werden soll, Bärte aus Bildern zu entfernen und dem Unentschiedenen damit zu zeigen, wie er (oder sie?) ohne Bart aussehen könnte. Das kann mit einem reinen Machine-Learning-Ansatz realisiert werden. Das würde aber einen großen Datensatz aus Bart/kein-Bart Bildpaaren voraussetzt. Wie man sich denken kann, ist es schwer, an so ein Datensatz zu kommen. Deshalb dient das Paper „Image-based Shaving“ [NLET08] als Basis für diese Arbeit. Hier wird ein Algorithmus präsentiert, der nur eine vergleichsweise geringe Menge an annotierten Daten voraussetzt. Ein Datensatz aus bartlosen Bildern wird als Model verwendet und Bärte werden als Ausreißer von diesem Model betrachtet. Das Verfahren funktioniert dabei folgendermaßen: Zuerst werden zwei Unterräume konstruiert, ein Gesichtsunterraum, der aus bartlosen Gesichtern besteht und ein Bartunterraum, der nur aus Bärten besteht. Dann wird eine Repräsentation des bartlosen Gesichts im Gesichtsunterraum und eine Repräsentation des Bartes im Bartunterraum berechnet. Daraus bilden sich zwei Schichten: eine Gesichts- und eine Bartschicht. Eine dritte Schicht, die Sonstigesschicht ergibt sich indem man vom Originalbild die Bart- und die Gesichtsschicht subtrahiert. Nun kann man die einzelnen Schichten unabhängig voneinander modifizieren und so Bärte entfernen, heller und dunkler machen oder sogar auf ein anderes Gesicht transferieren. Das war ein grober Überblick über den letzten von drei Algorithmen, die in [NLET08] beschrieben werden. Neben diesem Algorithmus werden in [NLET08] noch zwei weitere Verfahren beschrieben. Sie dienen als Basis für den letzten Algorithmus und sind für dessen Implementierung notwendig. In dieser Arbeit werden alle drei Algorithmen implementiert, evaluiert und gegebenenfalls verbessert.

Kapitel 1 beschäftigt sich mit den Grundlagen dieser Arbeit und geht dabei auf die notwendige Mathematik, relevante Computer-Vision-Themen und generelle Grundlagen dieser Arbeit ein. Kapitel 2 beschäftigt sich mit dem ersten Algorithmus, welcher auf dem Lösen eines Least-Squares-Problems basiert. Kapitel 3 beschäftigt sich mit dem zweiten Algorithmus, welcher auf robuster Statistik basiert. Kapitel 4 beschäftigt sich mit dem letzten Algorithmus. Dieser funktioniert so wie oben beschrieben. Kapitel 5 beschäftigt sich mit den implementierten Verbesserungen des letzten Algorithmus und wie sich diese auf das Ergebnis ausgewirkt haben. Das letzte Kapitel besteht aus einer

Zusammenfassung dieser Arbeit und einem Ausblick auf mögliche Verbesserungen und weitere Anwendungsmöglichkeiten des Verfahrens. Die Abbildungen in dieser Arbeit sollte man am besten auf einem Bildschirm betrachten.

1 Grundlagen

In diesem Kapitel geht es um die Grundlagen dieser Bachelorarbeit. Zuerst wird auf die mathematischen Grundlagen und Notationen, die verwendet werden, eingegangen. Darauf folgt ein Abschnitt über Morphologische Operationen. Dabei handelt es sich um Computer-Vision-Grundlagen. Dann wird auf die Open-Source-Bibliothek OpenCV eingegangen. Der nächste Abschnitt beschäftigt sich mit Traingular-Warping und Canonical-Faces sowie deren Bedeutung für diese Arbeit. Der vorletzte Abschnitt behandelt die Bilder, die in dieser Bachelorarbeit verwendeten werden. Am Ende dieses Kapitels wird dann noch darauf eingegangen wie Differenzbilder dargestellt werden.

1.1 Mathematik

In diesem Abschnitt geht es um die mathematischen Grundlagen und Notationen, die für das Verständnis der folgenden Kapitel vorausgesetzt werden. Zuerst wird die Notation besprochen. Dann folgt ein Unterabschnitt über die Least-Squares-Methode und darauf ein Unterabschnitt über die Tikhonov-Regularisierung. Schließlich geht es um die Choleski-Zerlegung, die Principal-Component-Analysis und dann noch um robuste Statistik.

1.1.1 Notation

In diesem Abschnitt wird die Notation besprochen, die in dieser Bachelorarbeit verwendet wird. Für die Schreibweise mathematischer Objekte gilt:

- $\boldsymbol{A}$: Matrizen sind fett gedruckte Großbuchstaben
- $\boldsymbol{I}$: Einheitsmatrizen werden mit $\boldsymbol{I}$ bezeichnet
- $\boldsymbol{b}$: Spaltenvektoren sind fett gedruckte Kleinbuchstaben
- h: Skalare werden nicht fett gedruckt

Für die Indizierung gilt:

- $\boldsymbol{a}_i$ ist der i-te Spaltenvektor der Matrix $\boldsymbol{A}$
- $\tilde{\boldsymbol{a}}_i^T$ ist der i-te Zeilenvektor der Matrix $\boldsymbol{A}$
- a_{ij} ist der Eintrag in der i-ten Zeile und der j-ten Spalte der Matrix $\boldsymbol{A}$
- b_i ist der i-te Eintrag des Vektors $\boldsymbol{b}$
- Bilder werden nicht in RGB (rot, grün, blau) sondern in BGR (blau, grün, rot) beschrieben, die Indizes b, g und r stehen hier nicht für Variablen sondern für die entsprechenden Farbkanäle.

Zur Schreibweise von Normen ist zu erwähnen, dass $\|\boldsymbol{b}\|_2^2$ die quadrierte L_2-Norm von $\boldsymbol{b}$ und $\|\boldsymbol{b}\|_1$ die L_1-Norm von $\boldsymbol{b}$ ist.

1.1.2 Least-Squares-Methode (Methode der kleinsten Quadrate)

Die Least-Squares-Methode ist ein wichtiges Werkzeug, zur Lösung von überbestimmten, linearen Gleichungssystemen. Da das Lösen solcher Gleichungssysteme ein wichtiger Teil dieser Bachelorarbeit ist, wird dieses Verfahren hier nun erklärt. Die folgende Erklärung basiert auf dem entsprechenden Kapitel aus dem Buch „Introduction to Linear Algebra“ von Strang [Str09]. Bei der „Least-Squares-Methode“ wird die Summe der quadrierten Fehler minimiert. Das bedeutet, dass für ein überbestimmtes Gleichungssystem der Form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \tag{1.1}$$

mit der Matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, den Unbekannten $\boldsymbol{x} \in \mathbb{R}^n$ und der rechten Seite $\boldsymbol{y} \in \mathbb{R}^n$, eine Optimallösung im Hinblick auf den kleinsten quadratischen Fehler gefunden werden kann, indem das folgende Optimierungsproblem gelöst wird:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{1.2}$$

Da es sich bei $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ um ein Polynom zweiten Grades handelt, hat es nur eine Minimalstelle. Diese Minimalstelle kann gefunden werden, indem man $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ ableitet, gleich 0 setzt und nach $\boldsymbol{x}$ auflöst:

$$\frac{\partial}{\partial \boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = 0 \tag{1.3}$$

$$-2\boldsymbol{A}^T(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) = 0 \tag{1.4}$$

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{y} = 0 \tag{1.5}$$

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{y} \tag{1.6}$$

$$\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y} \tag{1.7}$$

Dadurch ergibt sich ein lineares Gleichungssystem: die Normalengleichungen (siehe Gleichung (1.6)). Dieses Gleichungssystem aus Normalengleichungen kann beispielsweise mit der Cholesky-Zerlegung (siehe Abschnitt 1.1.4) gelöst werden, vorausgesetzt, dass $\boldsymbol{A}^T\boldsymbol{A}$ positiv definit ist. Falls das nicht der Fall ist, muss das Least-Squares-Problem regularisiert werden (siehe Abschnitt 1.1.3). Das kann passieren, wenn zu viele Spalten linear abhängig sind. Durch die Regularisierung kann die Matrix $\boldsymbol{A}^T\boldsymbol{A}$, welche per Konstruktion positiv semidefinit ist [LLM93], positiv definit gemacht werden. Wenn auf die Regularisierung verzichtet wird, kann man auch die Pseudoinverse berechnen. Dafür muss dann anstelle der Cholesky-Zerlegung eine Singulärwert-Zerlegung durchgeführt werden. Diese ist allerdings typischerweise deutlich langsamer als die Cholesky-Zerlegung.

1.1.3 Tikhonov-Regularisierung

Wie im vorherigen Abschnitt erwähnt wurde, kann Regularisierung dazu genutzt werden, Least-Squares-Probleme mittels Choleski-Zerlegung (siehe Abschnitt 1.1.4) lösbar zu machen. Die Erkärungen in diesem Kapitel basieren zum Teil auf der Vorlesung „Schlecht gestellte Probleme:

Einführung und numerische Methoden“ von Reinhardt [Rei02]. Zur Regularisierung muss die positiv semidefinite Matrix $\boldsymbol{A}^T\boldsymbol{A}$ (siehe Gleichung (1.7)) positiv definit gemachet werden. Das kann mit der Tikhonov-Regularisierung, einem der bekanntesten Regularisierungsverfahren, erreicht werden. Dazu wird ein Least-Squares-Problem um einen zusätzlichen Term erweitert:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \gamma\|\boldsymbol{x}\|_2^2 \tag{1.8}$$

Durch diesen Term kann die Lösung des Problems eingeschränkt werden. Der Vektor $\boldsymbol{x}$ muss so gewählt werden, dass $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ möglichst klein wird ohne, dass $\gamma\|\boldsymbol{x}\|_2^2$ zu groß wird. Mit $\gamma > 0$ wird die Wahl der Einträge von $\boldsymbol{x}$ beeinflusst. Ein großer Wert für γ sorgt dafür, dass wenige Einträge von $\boldsymbol{x}$ nicht 0 sind. Das kann sinnvoll sein, wenn man dafür sorgen will, dass nur wenige Spaltenvektoren von $\boldsymbol{A}$ gewählt werden, um den gesamten Term zu minimieren [Dav17]. In dieser Arbeit wird die Tikhonov-Regularisierung aber, wie zuvor erwähnt, dazu genutzt, $\boldsymbol{A}^T\boldsymbol{A}$ positiv definit zu machen. In diesem Fall wird γ möglichst klein gewählt, um das Ergebnis möglichst nicht zu verfälschen. Zur Regularisierung muss auch dieses Optimierungsproblem nach $\boldsymbol{x}$ abgeleitet, gleich 0 gesetzt und umgestellt werden:

$$\frac{\partial}{\partial \boldsymbol{x}}(\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \gamma\|\boldsymbol{x}\|_2^2) = 0 \tag{1.9}$$

$$-2\boldsymbol{A}^T(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) + 2\gamma\boldsymbol{I}\boldsymbol{x} = 0 \tag{1.10}$$

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} + \gamma\boldsymbol{I}\boldsymbol{x} = 0 \tag{1.11}$$

$$(\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{y} \tag{1.12}$$

$$\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I})^{-1}\boldsymbol{A}^T\boldsymbol{y} \tag{1.13}$$

Die Matrix $\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I}$ ist positiv definit, da $\boldsymbol{A}^T\boldsymbol{A}$ positiv semidefinit ist und $\gamma > 0$ ist. Das lässt sich zeigen, indem man $\boldsymbol{A}^T\boldsymbol{A}$ folgendermaßen zerlegt [LLM93]:

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{W}\boldsymbol{D}\boldsymbol{W}^{-1} \tag{1.14}$$

Die Spalten der Matrix $\boldsymbol{W}$ sind dabei die Eigenverktoren von $\boldsymbol{A}$ und $\boldsymbol{D}$ ist eine Diagonalmatrix auf deren Hauptdiagonale sich die Eigenwerte λ_i von $\boldsymbol{A}$ befinden. Für diese gilt $\lambda_i \geq 0$. Daraus folgt:

$$\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I} = \boldsymbol{W}\boldsymbol{D}\boldsymbol{W}^{-1} + \gamma\boldsymbol{I} \tag{1.15}$$

$$= \boldsymbol{W}(\boldsymbol{D} + \gamma\boldsymbol{I})\boldsymbol{W}^{-1} \tag{1.16}$$

Daraus lässt sich schließen, dass für alle Eigenwerte $\tilde{\lambda}_i$ von $\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I}$ folgendes gilt:

$$\tilde{\lambda}_i = \lambda_i + \gamma \geq \gamma > 0 \tag{1.17}$$

Die Matrix $\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I}$ ist damit positiv definit und das regularisierte Least-Squares-Problem hat auf jeden Fall eine Lösung.

1.1.4 Cholesky-Zerlegung

Die Choleski-Zerlegung ist ein effizientes Werkzeug zur Lösung von linearen Gleichungssystemen und dient in dieser Bachelorarbeit dem Zweck Least-Squares-Probleme (siehe Abschnitt 1.1.2) zu lösen. Die folgende Erklärung basiert auf der Arbeit „Zur Numerik linearer Gleichungssysteme“

[Vog04]. Die Cholesky Zerlegung dient dazu eine symmetrische, positiv definite Matrix $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ folgendermaßen zu zerlegen:

$$\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T \tag{1.18}$$

Hierbei ist $\boldsymbol{L}$ eine untere Dreiecksmatrix, deren Werte auf der Haupdiagonalen positiv sind. $\boldsymbol{L}$ wird folgendermaßen berechnet:

$$l_{ik} = \begin{cases} \sqrt{b_{kk} - \sum_{j=1}^{k-1} l_{kj}^2} & , i = k \\ \frac{1}{l_{kk}}(b_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj}) & , i > k \\ 0 & , \text{sonst} \end{cases} \tag{1.19}$$

Die Choleski-Zerlegung hat eine Komplexität von $O(n^3)$. Da $\boldsymbol{B}^{-1} = \boldsymbol{L}^{-1}(\boldsymbol{L}^T)^{-1}$ gilt, kann man zur Berechnung von $\boldsymbol{B}^{-1}$ Vorwärts- und Rückwärtssubstitution verwenden (siehe Abschnitt 1.1.5).

1.1.5 Invertierung von Dreiecksmatrizen mit Vorwärts- und Rückwärtssubstitution

In diesem Abschnitt geht es darum wie Dreiecksmatrizen einfach mit Vorwärts- und Rückwärtssubstitution invertiert werden können. Das ist nützlich wenn man die Inverse einer Matrix berechnen will, die sich in eine obere und eine untere Dreiecksmatrix zerlegen lässt. Dieser Abschnitt basiert auf [CK 11]. Die Inverse einer Dreicksmatrix $\boldsymbol{T} \in \mathbb{R}^{n \times n}$ ist folgendermaßen definiert:

$$\boldsymbol{T}\boldsymbol{O} = \boldsymbol{I} \tag{1.20}$$

Wobei es sich bei $\boldsymbol{O}$ um die Inverse von $\boldsymbol{T}$ handelt. Die Spalten der Matrix $\boldsymbol{O}$ werden folgendermaßen berechnet:

$$\boldsymbol{O} = [\boldsymbol{T}\boldsymbol{o}_1 = \boldsymbol{i}_1 \cdots \boldsymbol{T}\boldsymbol{o}_n = \boldsymbol{i}_n] \tag{1.21}$$

Die Gleichungssysteme $\boldsymbol{T}\boldsymbol{o}_k = \boldsymbol{i}_k$ werden mit Rückwärtssubstitution gelöst, wenn es sich bei $\boldsymbol{T}$ um eine obere Dreiecksmatrix handelt. Wenn es sich bei $\boldsymbol{T}$ um eine untere Dreicksmatrix handelt, nutzt man Vorwärtssubstitution. Beide Verfahren werden dazu verwendet Gleichungssystem der Form $\boldsymbol{T}\boldsymbol{x} = \boldsymbol{y}$ zu lösen.

Vorwärtssubstitution

Die Vorwärtssubstitution wird verwendet, wenn es sich bei $\boldsymbol{T}$ um eine untere Dreiecksmatrix handelt. Sie wird folgendermaßen berechnet:

$$x_1 = \frac{y_1}{t_{11}} \tag{1.22}$$

$$x_i = \frac{y_i - \sum_{j=1}^{i-1} t_{ij} x_j}{t_{ii}}, \quad i = 2, ..., n \tag{1.23}$$

Rückwärtssubstitution

Die Rückwärtssubstitution wird verwendet, wenn es sich bei $\boldsymbol{T}$ um eine obere Dreiecksmatrix handelt. Sie wird folgendermaßen berechnet:

$$x_n = \frac{y_n}{t_{nn}} \tag{1.24}$$

$$x_i = \frac{y_i - \sum_{j=i+1}^{n} t_{ij} x_j}{t_{ii}}, \quad i = n-1, ..., 1 \tag{1.25}$$

1.1.6 Principal-Component-Analysis (Hauptkomponentenanalyse)

In diesem Abschnitt geht es um die Principal-Component-Analysis, welche ein wichtiges Mittel zur Dimensionsreduktion von Vektorräumen ist. Sie stellt einen wichtigen Teil dieser Arbeit dar. Die folgende Erklärung basiert auf dem Paper „Principal Component Analysis - A Tutorial“ von Tharwat [Tha16]. Durch eine Dimensionsreduktion können Datensätze, die aus vielen korrelierenden statistische Variablen bestehen auf einige wenige nicht korrelierende Variablen reduziert werden. Dazu betrachtet man die Datenpunkte als Punkte in einem n-dimensionalen Raum. Bei n handelt es sich um die Anzahl aller Variablen des Datensatzes und m ist die Anzahl der Datenpunkte des Datensatzes $\boldsymbol{D}^{n \times m}$. Jetzt versucht man Vektoren in diesem Raum zu finden, die entlang der größten Varianz des Datensatzes liegen. Dazu berechnet man zunächst den Mittelwert $\boldsymbol{\mu}$ aller Datenpunkte:

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{d}_i \tag{1.26}$$

Dann zieht man $\boldsymbol{\mu}$ von allen Spaltenvektoren von $\boldsymbol{D}$ ab, um $\boldsymbol{E}$ zu berechnen:

$$\boldsymbol{E} = [(\boldsymbol{d}_1 - \boldsymbol{\mu}) \cdots (\boldsymbol{d}_m - \boldsymbol{\mu})] \tag{1.27}$$

Dadurch verschiebt man den Ursprung des Vektorraums in die Mitte des Datensatzes. Man berechnet dann die Kovarianzmatrix von $\boldsymbol{E}$ indem man $\boldsymbol{E}\boldsymbol{E}^T$ berechnet. Bei der Kovarianzmatrix handelt es sich um eine Verallgemeinerung der Varianz auf mehreren Dimensionen. Jetzt berechnet man die Eigenvektoren $\boldsymbol{v_i}$ der Matrix und sortiert diese absteigend nach ihren korrespondierenden Eigenwerten. Die Eigenvektoren liegen so im Raum, dass sie entlang der größten Varianzrichtungen des Datensatzes liegen. Je größer der Eigenwert eines Eigenvektors ist, desto größer ist die Varianz in der Richtung dieses Vektors. Die Eigenvektoren werden ”Principal-Components”(Hauptkomponenten) genannt. Der Ortsvektor aller Principal-Components ist der Mittelwert $\boldsymbol{\mu}$ des Datensatzes $\boldsymbol{D}$. Zur Dimensionsreduktion wählt man jetzt die ersten l Principal-Components, die zu den größten Eigenwerten gehören und erstellt daraus einen neuen Unterraum $\boldsymbol{U}^{n \times l}$.

$$\boldsymbol{U} = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_l] \tag{1.28}$$

1.1.7 Robuste Statistik

Ausreißer sind Datenpunkte, die weit entfernt vom Großteil der restlichen Datenpunkte liegen. Bei ihnen kann es sich um falsche Datenpunkte handeln. Solche Fehler können beim Messen oder Übertragen passieren. Ausreißer können aber auch wichtige und interessante Informationen enthalten,

weshalb sie immer untersucht und nicht verworfen werden sollten. Robuste statistische Methoden sind für einen Algorithmus in dieser Arbeit wichtig, weshalb hier in dieses Thema eingeführt werden soll. Die folgende Einführung basiert zum Teil auf einer Vorlesung [AR15]. Während in klassischen statistischen Methoden, davon ausgegangen wird, dass die observierten Daten nach einem Model verteilt sind, stellt das Modell in der robusten Statistik nur eine mathematische Abstraktion der Realität dar. Es sollen also Verfahren, auch Schätzer genannt, geschaffen werden, die auch bei großen Abweichungen einzelner Datenpunkte, noch gute Ergebnisse liefern. Der Mittelwert ist nicht robust, da einige wenige Ausreißer das Ergebnis stark beeinflussen. Der Median (Zentralwert) einer geordneten Stichprobe $\boldsymbol{p}$ von n Messwerten ist definiert als:

$$\tilde{\boldsymbol{p}} = \begin{cases} \boldsymbol{p}_{\frac{n+1}{2}} & , n \text{ ungerade} \\ \frac{1}{2}(\boldsymbol{p}_{\frac{n}{2}} + \boldsymbol{p}_{\frac{n}{2}+1}) & , n \text{ gerade} \end{cases} \tag{1.29}$$

Der Median ist robust, denn er wird fast nicht von Ausreißern beeinflusst. Für jeden robusten oder nicht robusten Schätzer $\hat{T}_n$, einen Datensatz $\boldsymbol{G}$, den zu schätzenden Wert v und den Schätzfehler ϵ gilt:

$$\hat{T}_n(\boldsymbol{G}) = v + \epsilon \tag{1.30}$$

$$\lim_{n\to\infty} \epsilon = 0 \tag{1.31}$$

Das bedeutet, je größer die Anzahl n der Datenpunkte aus $\boldsymbol{G}$, die betrachtet wird, desto präziser ist der Schätzer. Das wird Konsistenz genannt. Es gibt verschieden Klassen von Schätzern, für die jeweils weitere Eigenschaften gelten, hier wird aber nur auf die Klasse der „M-Estimator-of-Regression“ eingegangen. Diese Form des Schätzers kann genutzt werden, um Regressionsprobleme zu lösen, bei denen es darum geht, eine Beobachtung über die Datenpunkte eines Datensatzes zu erklären. Ein M-Estimator-of-Regression ist die Lösung des folgenden Optimierungsproblems:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \sum_{i=1}^{n} \rho(r_i) \tag{1.32}$$

$$\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{G}\boldsymbol{x} \tag{1.33}$$

Wobei ρ eine konvexe Funktion $\boldsymbol{r} \to \mathbb{R}$, $\hat{\boldsymbol{x}}$ der geschätzte Wert des M-Schätzers und $\boldsymbol{r}$ die Differenz zwischen der Beobachtung $\boldsymbol{y}$ und dem Produkt der Regressoren $\boldsymbol{G}$ und den Regressionskoeffizienten $\boldsymbol{x}$ ist. Für ρ muss Folgendes gelten:

- $\rho(0) = 0$
- $\rho'(u) > 0$ für $u > 0$
- $\rho'(u) < 0$ für $u < 0$
- $\rho'(u) = 0$ für $u = 0$

Diese Eigenschaften gelten beispielsweise die folgende Funktion (siehe Abbildung 1.1):

$$\rho_1(u) = u^2 \tag{1.34}$$

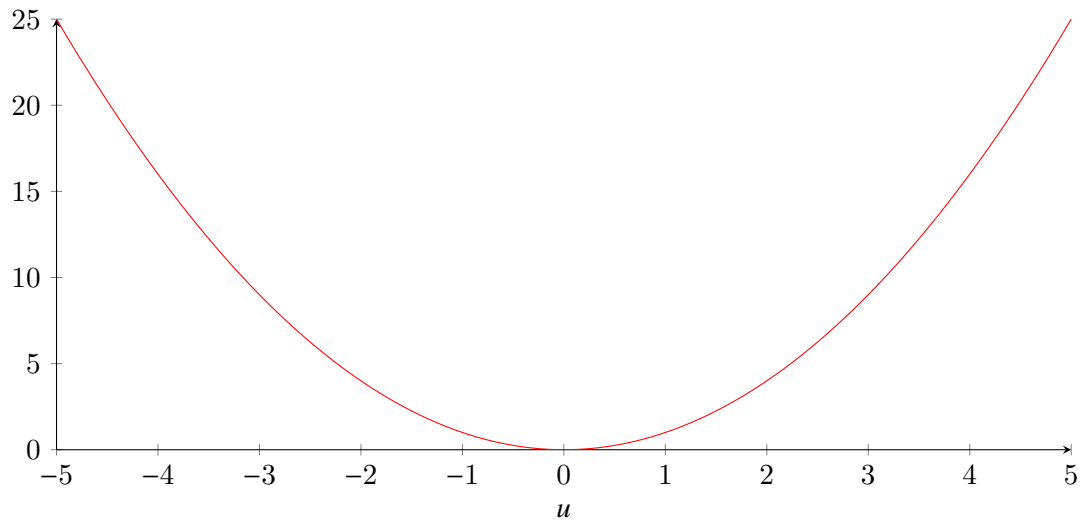


Abbildung 1.1: Nicht robuste Funktion ρ_1.

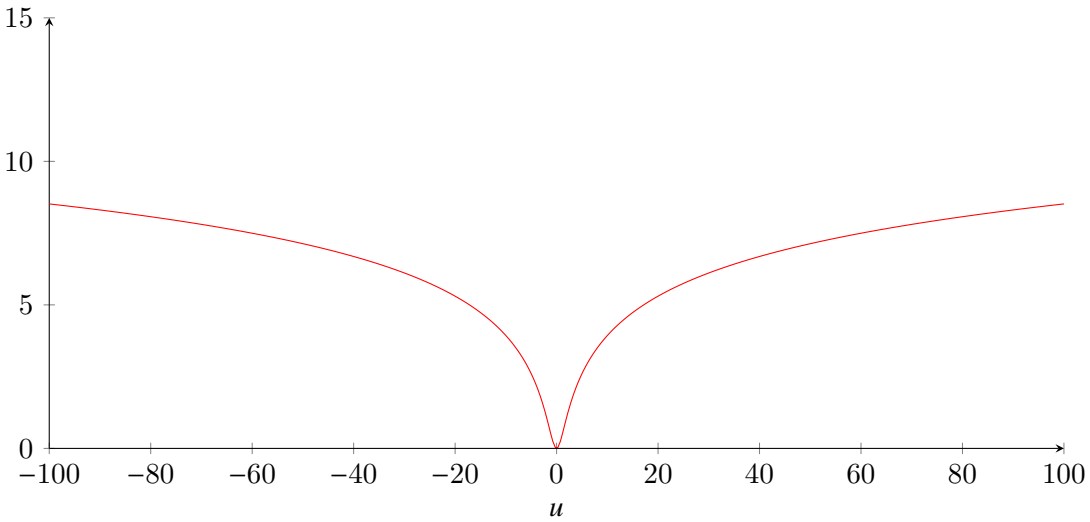


Abbildung 1.2: Lorentz-Straffunktion [Wei04]

Die Variable ϵ steht hierbei für einen kleinen Wert. Wenn der M-Schätzer robust sein soll, muss zusätzlich folgendes gelten:

$$\lim_{u \to \pm\infty} \frac{\rho'(u)}{2u} = 0 \tag{1.35}$$

Beispiele für Funktionen, die nach dieser Definition robust sind, sind die „Lorentz-Straffunktion“ (siehe Gleichung (1.36) und Abbildung 1.2) und die „Geman-McClure Funktion“ (siehe Gleichung (1.37) und Abbildung 1.3). Letztere wird in einem späteren Kapitel wichtig.

$$\rho_2(u) = \ln(1 + \frac{u^2}{2}) \tag{1.36}$$

$$\rho_3(u, \sigma) = \frac{u^2}{u^2 + \sigma^2} \tag{1.37}$$

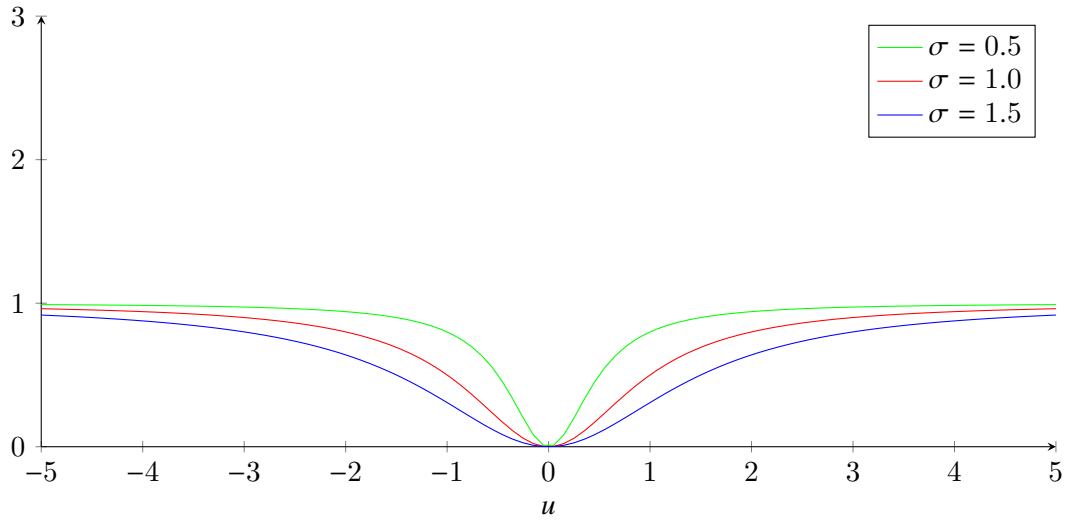


Abbildung 1.3: Geman-McClure Funktion [NLET08]

Die Optimierungsfunktion (siehe Gleichung (3.1)) kann mit der „Iteratively Re-Weighted-Least-Squares-Methode“ gelöst werden. Bei diesem Verfahren wird das Optimierungsproblem, wie der Name schon sagt, iterativ gelöst, indem man in jeder Iteration ein Least-Squares-Problem löst, dessen Gleichungen durch ρ' gewichtet werden. Gleichungen, die zu Ausreißern gehören, werden geringer gewichtet als die restlichen Gleichungen. Dadurch wird die Relevanz dieser verringert. Dazu wird die Gleichung (3.1) in das folgende Least-Squares-Problem umgeformt:

$$\boldsymbol{x}^{(k)} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{W}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x})\|_2^2 \tag{1.38}$$

Um dieses Problem lösen zu können, muss es nach $\boldsymbol{x}$ abgeleitet, gleich 0 gesetzt und nach $\boldsymbol{x}$ aufgelöst werden.

$$\frac{\partial}{\partial \boldsymbol{x}} \|\boldsymbol{W}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x})\|_2^2 = 0 \tag{1.39}$$

$$-2\boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x}) = 0 \tag{1.40}$$

$$(\boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{G}\boldsymbol{x}) - (\boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{y}) = 0 \tag{1.41}$$

$$\boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{G}\boldsymbol{x} = \boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{y} \tag{1.42}$$

$$\boldsymbol{x} = (\boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{G})^{-1}\boldsymbol{G}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{y} \tag{1.43}$$

Dieses Problem löst man nun in jeder Iteration k. Das tut man so lange bis $\boldsymbol{x}^{k+1} - \boldsymbol{x}^k$ für $k = 1, 2, \ldots$ konvergiert. Die Einträge der Gewichtungsmatrix $\boldsymbol{W}$ werden dabei in jeder Iteration folgendermaßen berechnet:

$$w_{ij} = \begin{cases} \frac{\rho'(r_i)}{2r_i} & , i = j \\ 0 & , \text{sonst} \end{cases} \tag{1.44}$$

Da für einen robusten M-Schätzer Gleichung (1.35) gilt, werden die Gleichungen für die der Wert r_i groß ist geringer gewichtet als die, für die der Wert r_i klein ist. Wenn der Wert von r_i groß ist, handelt es sich um einen Ausreißer, welcher nur geringen Einfluss auf den M-Schätzer haben soll. Somit

können diese Gleichungen vernachlässigt werden, um bessere Koeffizienten für die Gleichungen zu finden, für die bereits gute gefunden wurden. Dieser Effekt wird mit jeder Iteration größer, da die Differenz $\boldsymbol{y} - \boldsymbol{Gx}$ an den Ausreißergleichungen immer größer und bei den anderen Gleichungen immer kleiner wird.

1.2 Morphologische Operationen

Morphologische Operationen eignen sich oft dafür Rauschen zu entfernen und dafür sollen sie hier auch verwendet werden. Alle Erklärungen zu diesen Operationen basieren auf der Vorlesung [Bru18]. Morphologische Operationen wenden ein Strukturelement auf ein Eingabebild an, um ein Ausgabebild zu erzeugen. Dieses Strukturelement ist ein Kernel. Dieser hat für gewöhnlich eine quadratische oder eine Kreisform. Der Kernel hat einen Ankerpunkt. Dieser ist für gewöhnlich das Zentrum des Kernels. Der Kernel wandert mit seinem Ankerpunkt über alle Pixel des Bildes und die entsprechende Morphologische Operation wird durchgeführt. Abbildung 1.4 zeigt Beispiele der erklärten Operationen.

1.2.1 Erosion

Bei der Erosion wird dem Pixel am Ankerpunkt das Minimum aller Pixelwerte im Kernelbereich zugewiesen. Dabei wachsen dunkle Bereiche an und helle werden kleiner.

1.2.2 Dilation

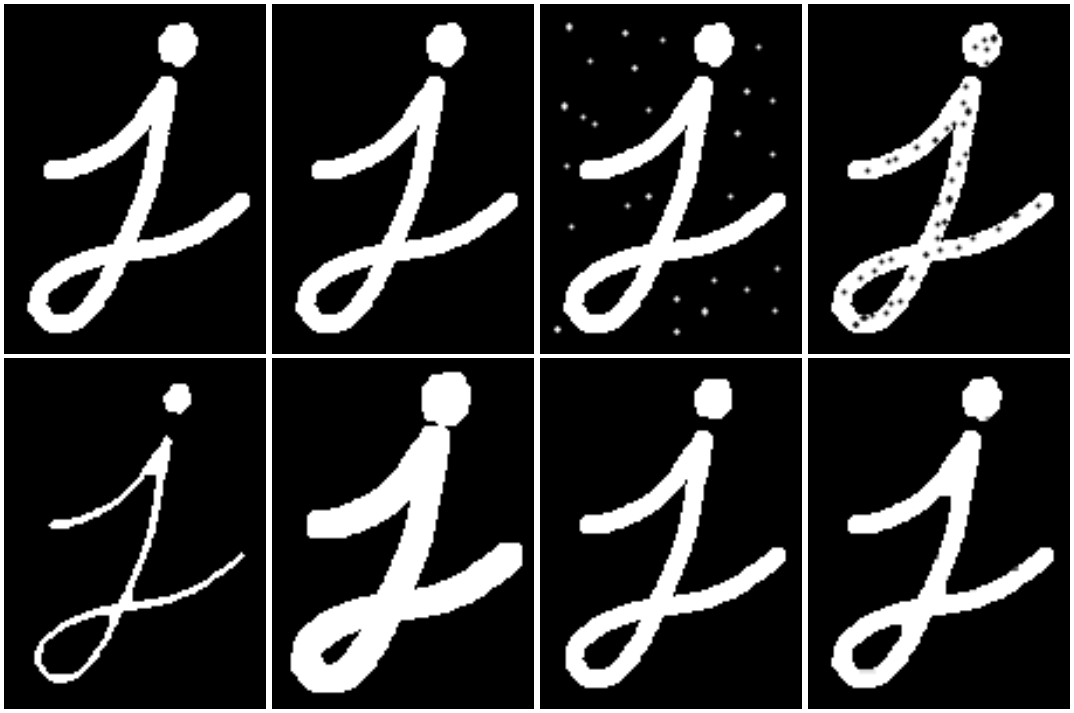
Bei der Dilation wird dem Pixel am Ankerpunkt das Maximum aller Pixelwerte im Kernelbereich zugewiesen. Dabei wachsen helle Bereiche an und dunkle werden kleiner.

1.2.3 Opening

Beim Opening wandert der Kernel erst mit einer Erosion über das Bild und dann mit einer Dilation. Dieses Verfahren eignet sich, um dunkles Rauschen zu entfernen.

1.2.4 Closing

Beim Closing wandert der Kernel erst mit einer Dilation über das Bild und dann mit einer Erosion. Dieses Verfahren eignet sich, um helles Rauschen zu entfernen.



(a) **(b)** **(c)** **(d)**

Abbildung 1.4: Hier sieht man die Auswirkungen verschiedener Morphologischer Operationen. In der oberen Zeile befinden sich die Ausgangsbilder und in der unteren Zeile sieht man das Ergebnis nach der Operation. Bei den Operationen handelt es sich um Erosion (a), Dilation (b), Opening (c) und Closing (d) [Ope19b].

1.3 OpenCV

OpenCV [Ope19a] ist eine Open-Source-Bibliothek im Bereich Computer-Vision und Machine-Learning. Sie stellt über 2500 optimierte Computer-Vision und Machine-Learning Algorithmen bereit. Dabei handelt es sich sowohl um klassische als auch neue State-of-the-Art Algorithmen. Die Library ist stark optimiert und für viele der Algorithmen existieren CUDA und OpenCL Implementierungen. Deshalb wird OpenCV oft für Echtzeitanwendungen verwendet. OpenCV wird unter der BSD-Lizenz verbreitet, weshalb auch Unternehmen die Library für kommerzielle Zwecke nutzen und anpassen können. OpenCV wurde in C++ entwickelt und verfügt über ein Template Interface, welches problemlos mit STL Containern funktioniert. Zusätzlich zum C++ Interface existieren Python, Java und MATLAB Interfaces. Da OpenCV hochperformant ist und viele, für diese Arbeit notwendige Algorithmen und Datenstrukturen bereitstellt, wurde sie zur Implementierung in dieser Bachelorarbeit verwendet.

Abbildung 1.5: Das sind die Positionen der 68 Landmarks die von OpenCVs Gesichtserkennung erkannt werden [RL18].

1.4 Triangular-Warping und Canonical-Faces (Dreieckstransformationen und Kanonische-Gesichter)

Die Algorithmen in dieser Arbeit setzen voraus, dass alle Gesichter gleichförmig sind. Das bedeutet: Alle Bilder sind gleich groß und alle Strukturen sind in allen Bildern an derselben Stelle. Das kann auf manuellem Weg erreicht werden, das wäre allerdings umständlich. In diesem Abschnitt wird ein Algorithmus beschreiben, mit dem sich das automatisieren lässt. Dazu wird eine funktionierende Gesichtserkennung vorausgesetzt, die in der Lage ist markante Punkte, die sogenannten „Landmarks“, in einem Gesicht zu markieren. Als Gesichtserkennung werden hier die OpenCV (siehe Abschnitt 1.3) Module „Object Detection“ und „Face Analysis“ verwendet. Das Object-Detection-Modul dient dazu, den Suchbereich für das Face-Analysis-Modul zu verkleinern. Dadurch wird das Finden und Markieren der Landmarks beschleunigt. In OpenCV werden dabei 68 Landmarks markiert (siehe Abbildung 1.5). Jedem Landmark muss eine Stelle im Canonical-Face zugewiesen werden. Die Landmarks werden dann so zu Dreiecken zusammengefasst, dass sich jedes Gesichtspixel in genau einem Dreieck befindet (siehe Abbildung 1.6). Diese Dreiecke werden nun an die vordefinierten Stellen im Canonical-Face transformiert. Das Ergebnis ist das Canonical-Face. Abbildung 1.7 einige Beispiele. Die Dreiecke können fast verlustfrei zurück in das Originalbild transformiert werden, wenn der Auflösungsunterschied zwischen Ausgangsbild und Canonical-Face nicht zu groß ist. So können Modifikationen des Canonical-Faces auf das Originalbild übertragen werden. Die Auflösung der Bilder beträgt in dieser Arbeit 95×95 Pixel.

1.5 Der Unterraum der bartlosen Gesichter

In diesem Abschnitt wird der bartlose Unterraum $\boldsymbol{V}^{d\times n}$ beschrieben. Dieser Unterraum ist ein elementarer Bestandteil dieser Arbeit. Zunächst muss ein Datensatz aus Canonical-Faces (siehe Abschnitt 1.4) erstellt werden. Alle diese Gesichter müssen bartlos sein. Dann wird jedes Bild in die folgende Vektorform gebracht:

$$\begin{bmatrix} (a_b, a_g, a_r) & (b_b, b_g, b_r) \\ (c_b, c_g, c_r) & (d_b, d_g, d_r) \end{bmatrix} \rightarrow \begin{bmatrix} a_b \\ a_g \\ a_r \\ b_b \\ b_g \\ b_r \\ c_b \\ c_g \\ c_r \\ d_b \\ d_g \\ d_r \end{bmatrix} \tag{1.45}$$

Die entstehenden Vektoren dienen dann als Spaltenvektoren der Matrix $\boldsymbol{V}$, d ist die Anzahl der Pixel mit der Anzahl der Farbkanäle multipliziert und n ist die Anzahl an Bildern. Der Datensatz enthält in dieser Arbeit 943 Bilder. Die Bilder, die für diese Arbeit verwendet wurden, kommen aus dem Aberdeen Face Set [Cra], der Chicago Face Database [Ma 15], der Color FERET Database [1] [PHJP98] [PHSP00] und Googles Open Images Dataset [KDA+16].

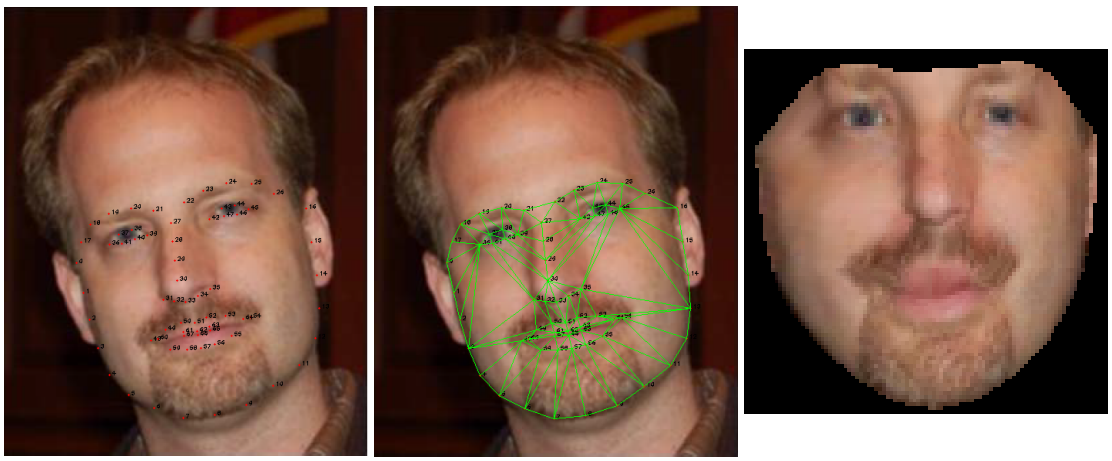


Abbildung 1.6: Diese Abbildung zeigt, wo sich die erkannten Landmarks befinden, wie sie zu Dreiecken zusammengefasst werden und wie das entsprechende Ergebnisbild des Traingular-Warping Verfahrens aussieht. Adaptiert von [NLET08].

[1] Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office

Abbildung 1.7: Einige erzeugte Canonical-Faces. Links ist das Ausgangsbild und rechts ist das Canonical-Face. Adaptiert von [Ma 15].

1.6 Darstellung von Differenzbildern

Pixelwerte können nur im Intervall $[0, 255]$ dargestellt werden. Differenzbilder können aber negative Pixelwerte haben. Alle Pixel p_{ij} eines Differenzbildes $\boldsymbol{P}_{\text{diff}}$ werden in dieser Arbeit folgendermaßen zur Darstellung skaliert.

$$p_{ij} = (\frac{p_{ij_b}}{2} + 127.5, \frac{p_{ij_g}}{2} + 127.5, \frac{p_{ij_r}}{2} + 127.5) \tag{1.46}$$

Diese Skalierung sorgt dafür, dass alle Werte im darstellbaren Bereich liegen. Negative Werte werden also dunkel dargestellt, während positive Werte hell dargestellt werden. Pixel mit dem Wert $(0, 0, 0)$ sind grau. Die Abbildungen 4.3 und 5.9 zeigen einige Beispiele, bei denen diese Skalierung verwendet wurde.

2 Bartentfernung mit der Least-Squares-Methode

In diesem Kapitel geht es um einen naiven Algorithmus zur Bartentfernung aus [NLET08], der zwar nicht sehr gut funktioniert, aber wichtige Erkenntnisse über das Problem liefert. Die Idee dieses Verfahrens ist, das bärtige Gesicht aus einer Linearkombination von bartlosen Canonical-Faces (siehe Abschnitt 1.4) zu rekonstruieren. Da bartlose Gesichter an den Stellen, an denen sich normalerweise Bärte befinden, keine bartfarbenen Pixel besitzen, können diese Stellen nicht rekonstruiert werden und nehmen deshalb die Hautfarbe des restlichen Gesichtes an.

2.1 Algorithmus

Um das bärtige Gesicht als Linearkombination von bartlosen Gesichtern darzustellen, müssen zunächst die Koeffizienten dieser Kombination bestimmt werden. Sei $\boldsymbol{x}$ das bärtige Canonical Face, $\boldsymbol{x}^*$ das bärtige Canonical Face ohne Bart, $\boldsymbol{V}$ der Unterraum der bartlosen Gesichter (siehe Abschnitt 1.5) und $\hat{\boldsymbol{c}}$ die Koeffizienten der Linearkombination. Dann wird $\boldsymbol{x}^*$ folgendermaßen berechnet:

$$\boldsymbol{x}^* = \boldsymbol{V}\hat{\boldsymbol{c}} \tag{2.1}$$

Da $\hat{\boldsymbol{c}}$ unbekannt ist, wird ein Least-Squares-Fit (siehe Abschnitt 1.1.2) durchgeführt:

$$\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c}} (\|\boldsymbol{x} - \boldsymbol{V}\boldsymbol{c}\|_2^2 + \gamma\|\boldsymbol{c}\|_2^2) \tag{2.2}$$

$$\hat{\boldsymbol{c}} = (\boldsymbol{V}^T\boldsymbol{V} + \gamma\boldsymbol{I})^{-1}\boldsymbol{V}^T\boldsymbol{x} \tag{2.3}$$

Jeder Koeffizient von $\hat{\boldsymbol{c}}$ gibt den Beitrag eines Bildes aus $\boldsymbol{V}$ zum Ergebnis an. Wenn $\boldsymbol{V}^T\boldsymbol{V}$ positiv definit ist, kann die Cholesky-Zerlegung (siehe Abschnitt 1.1.4) verwendet werden, um die Inverse der Matrix zu bestimmen. Falls die Matrix nur positiv semidefinit ist, kann man die Tikhonov Regularisierung (siehe Abschnitt 1.1.3) verwenden, um die Matrix positiv definit zu machen. Dazu wählt man für γ einen kleinen Wert. In dieser Arbeit wurde, für positiv semidefinite Matrizen, der Wert 0,1 gewählt. Dieser Wert ist groß genug, um die Matrix zu regularisieren und klein genug, um das Ergebnis nicht zu sehr zu beeinflussen. Wenn eine Regularisierung nicht notwendig ist, wird der Wert von γ auf 0 gesetzt. Anstatt der Cholesky-Zerlegung kann man auch eine Singulärwertzerlegung durchführen und die Pseudoinverse bestimmen. Die Singulärwertzerlegung setzt nicht voraus, dass die Eingabematrix positiv definit ist. Deshalb kann man in diesem Fall auf eine Regularisierung verzichten. Die Berechnung dauert allerdings lange und ist deshalb keine sinnvolle Alternative.

Abbildung 2.1: Einige Ergebnisse des Least-Squares-Fits. In der oberen Zeile befinden sich die Ausgangsbilder und in der unteren die Ergebnisbilder. Man sieht, dass die Gesichtsrekonstruktion bei diesen Bildern gut funktioniert. Der Bart wird allerdings nur heller.

2.2 Ergebnis

Wie man in den Abbildungen 2.1 und 2.2 sieht, funktioniert dieser Ansatz nicht sehr gut. Der Anteil der Bartpixel ist so groß, dass Bartstellen im Gesicht aus bartlosen Gesichtern rekonstruiert werden müssen. Der Bart wird nur heller und andere Gesichtsstellen werden nicht exakt rekonstruiert. Bei genauem Hinschauen fällt auf, dass vor allem Nasen und Münder schlecht rekonstruiert werden. Das liegt daran, dass es sich dabei um markante Teile des Gesichts handelt, die von Mensch zu Mensch stark variieren. Sie können also mit hoher Wahrscheinlichkeit nur schlecht in $\boldsymbol{V}$ rekonstruiert werden, wenn keine ähnlichen Gesichter vorhanden sind. Das gleiche gilt für die Augen. Der Effekt ist hier aber nicht so stark. Insgesamt funktioniert die Rekonstruktion des Gesichts aber in vielen Fällen relativ gut, auch wenn sie nicht exakt ist. Wenn der Bart fast nicht sichtbar ist, kann er mit dieser Methode sogar entfernt werden (siehe Abbildung 2.1 zweites Bild von links). Abbildung 2.2 zeigt einige Beispiele, bei denen die Gesichtsrekonstruktion schlecht funktioniert. Das Ergebnis ist in diesen Fällen verschwommen und die Rekonstruktion von Gesichtsteilen, die von Mensch zu Mensch stark variieren, funktioniert noch schlechter. Das gilt auch für den Bart, da keines der Gesichter aus $\boldsymbol{V}$ einen Bart hat.

Abbildung 2.2: Einige Ergebnisse des Least-Squares-Fit. In der oberen Zeile befinden sich die Ausgangsbilder und in der unteren die Ergebnisbilder. Man sieht, dass die Gesichtsrekonstruktion hier nur schlecht funktioniert. Der Bart wird auch hier nur heller.

3 Bartentfernung mit robuster Statistik

Dieses Kapitel beschäftigt sich mit einem weiteren Algorithmus zur Bartentfernung aus [NLET08]. Dieser basiert auf der in Kapitel 2 gewonnen Erkenntnis, dass markante Gesichtsteile oft nur schlecht in $\boldsymbol{V}$ rekonstruiert werden. Da das insbesondere für den Bart gilt, kann man Bartpixel als Ausreißer von $\boldsymbol{V}$ betrachten. Hier soll nun ein Optimierungsproblem aufgestellt werden, das Ausreißer nur gering bestraft, so dass sie bei der Rekonstruktion ignoriert werden können. Wenn Pixel exakt rekonstruiert werden, soll das dennoch belohnt werden. Dafür eignen sich robuste statistische Methoden (siehe Abschnitt 1.1.7).

3.1 Algorithmus

Wie zuvor sollen auch hier die Koeffizienten $\hat{\boldsymbol{c}}$ einer Linearkombination bestimmt werden. Diese werden mit einem „M-Estimator-of-Regression“ gefunden, welcher folgendermaßen aussieht:

$$\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c}} \sum_{i=1}^{n} \rho(r_i, \sigma) \tag{3.1}$$

$$\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{V}\boldsymbol{c} \tag{3.2}$$

Bei ρ handelt es sich um die zuvor genannte Geman-McClure Funktion (siehe Gleichung (1.37)). Da diese Funktion gegen 1 konvergiert, werden große Einträge in $\boldsymbol{r}$ maximal mit 1 bestraft. Wenn ein Pixel schlecht rekonstruiert, wird ist die maximale Strafe dafür also 3, da jedes Pixel über 3 Farbkanäle verfügt. Mit dem Wert von σ kann kontrolliert werden, wie schnell die Geman-McClure Funktion konvergiert. Ein großes σ sorgt dafür, dass die Geman-McClure Funktion langsam gegen 1 konvergiert. Ein kleiner Wert sorgt für eine schnellere Konvergenz.

Dieses Optimierungsproblem kann mit der Iteratively-Re-Weighted-Least-Squares-Methode (siehe Abschnitt 1.1.7) gelöst werden. Dazu wird das Problem umgeformt:

$$\boldsymbol{c}^{(k)} = \arg\min_{\boldsymbol{c}} (\|\boldsymbol{W}(\boldsymbol{x} - \boldsymbol{V}\boldsymbol{c})\|_2^2 + \gamma\|\boldsymbol{c}\|_2^2) \tag{3.3}$$

$$= (\boldsymbol{V}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{V} + \gamma\boldsymbol{I})^{-1}\boldsymbol{V}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x} \tag{3.4}$$

Zur Lösung verwendet man, wie zuvor, die Cholesky-Zerlegung (siehe Abschnitt 1.1.4). Da es auch hier möglich ist, dass die Matrix $\boldsymbol{V}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{V}$ positiv semidefinit ist, kann man wieder die Tikhonov Regularisierung verwenden, um die Matrix positiv definit zu machen. In dem Fall wird γ als 0,1 gewählt. Ansonsten wählt man für γ den Wert 0. Alternativ kann man auch hier die Pseudoinverse mit einer Singulärwertzerlegung bestimmen, anstatt die Inverse mit der Cholesky-Zerlegung zu berechnen. Da das Optimierungsproblem iteriert wird, ist das aufgrund der langen Berechnungszeit jedoch eine schlechte Alternative.

Die Gewichtungsmatrix $\boldsymbol{W}^{n\times n}$ wird in der ersten Iteration mit der Einheitsmatrix initialisiert. In der ersten Iteration wird deshalb derselbe Least-Squares-Fit wie in Kapitel 2 durchgeführt:

$$(\boldsymbol{V}^T\boldsymbol{I}^T\boldsymbol{I}\boldsymbol{V} + \gamma\boldsymbol{I})^{-1}\boldsymbol{V}^T\boldsymbol{I}^T\boldsymbol{I}\boldsymbol{x} = (\boldsymbol{V}^T\boldsymbol{V} + \gamma\boldsymbol{I})^{-1}\boldsymbol{V}^T\boldsymbol{x} \tag{3.5}$$

In den folgenden Iterationen wird $\boldsymbol{W}$ folgendermaßen neu berechnet:

$$w_{ij} = \begin{cases} \frac{1}{2r_i}\frac{\partial\rho(r_i,\sigma)}{\partial r_i} = \frac{1}{2r_i}\frac{2r_i\cdot\sigma^2}{(r_i^2+\sigma^2)^2} = \frac{\sigma^2}{(r_i^2+\sigma^2)^2} & , i = j \\ 0 & , \text{sonst} \end{cases} \tag{3.6}$$

$$\sigma = 1.4826 \cdot median(|r_i| : \forall i \in [i,d]) \tag{3.7}$$

Die Berechnung von $\boldsymbol{W}$ folgt aus der Geman-McClure Funktion und der Gleichung (1.44). Der Wert von σ wird mit jeder Iteration kleiner, bis ein Grenzwert erreicht wird. Deshalb wird eine schlechte Rekonstruktion in frühen Iterationen geringer bestraft und späteren Iterationen stärker. Anfangs werden schlecht rekonstuierte Werte also nicht schnell zu Ausreißern. Zu einem späteren Zeitpunkt werden sie dafür aber schneller zu Ausreißern. Gut rekonstruierte Werte werden deshalb immer wichtiger, wodurch ihre gute Rekonstruktion erhalten bleibt oder sogar verbessert wird.

In der Implementierung handelt es sich bei $\boldsymbol{W}$ nicht um eine Matrix, sondern um einen Vektor $\boldsymbol{h} \in \mathbb{R}^n$. Die gewichtete Matrix $\boldsymbol{F}$ einer Matrix $\boldsymbol{A}$ und einer Gewichtungsmatrix $\boldsymbol{W}$ wird deshalb folgendermaßen berechnet.

$$h_i = w_{ii} \tag{3.8}$$

$$\tilde{\boldsymbol{f}}_i = h_i\tilde{\boldsymbol{a}}_i \tag{3.9}$$

Diese Art der Berechnung hat eine Komplexität von $O(n)$, was wesentlich besser ist als $O(n^3)$. Es werden so viele Iterationen durchgeführt, bis $|\boldsymbol{c}^{(k)} - \boldsymbol{c}^{(k-1)}|$ gegen 0 konvergiert.

Abbildung 3.1 zeigt einige Ergebnisse des Verfahrens. Wie man sieht, funktioniert die Bartentfernung mit diesem Algorithmus nicht besser als mit der Least-Squares-Methode aus Kapitel 2 (vergleiche Abbildung 2.1). Die Rekonstruktion des Gesichts funktioniert sogar etwas schlechter.

3.2 Verifizierung der Implementierung

Die schlechte Gesichtsrekonstruktion kann an Fehlern in der Implementierung liegen. In diesem Abschnitt geht es darum, das auszuschließen. Dazu werden hier alle Experimente besprochen, die durchgeführt wurden, um das herauszufinden. Zusätzlich ermöglicht das einen Einblick in das Verhalten des Algorithmus.

3.2.1 Gewichtsveränderung

Dieser Abschnitt beschäftigt sich mit der Veränderung der Gewichte über die einzelnen Iterationen hinweg. Dazu wurde in jeder Iteration k ein Ergebnisbild $\boldsymbol{V}\boldsymbol{c}^{(k)}$ und ein Gewichtbild ausgegeben, um die Veränderungen zu veranschaulichen. Hier würde man erwarten, dass sowohl der Bart als auch andere markante Gesichtsstellen mit der Zeit ein geringeres Gewicht bekommen. In Abbildung 3.2 sieht man aber, dass das nicht der Fall ist. Der Bart bekommt zwar ein etwas geringeres Gewicht als das restliche Gesicht, aber dieses wird mit der Zeit kaum geringer. Deswegen verändert sich das Gesicht auch kaum.

Abbildung 3.1: Einige Ergebnisse der Iteratively-Re-Weighted-Least-Squares-Methode. In der oberen Zeile befinden sich die Ausgangsbilder und in der unteren die Ergebnisbilder. Man sieht, dass die Gesichtsrekonstruktion hier etwas schlechter funktioniert als bei der Least-Squares-Methode. Der Bart wird hier auch nur heller.

(a) **(b)** **(c)** **(d)** **(e)**

Abbildung 3.2: Man sieht hier, wie sich das Ergebnisbild und die Gewichtungsmatrix $\boldsymbol{W}$ über die Iterationen hinweg verändern. Die obere Zeile zeigt die Ergebnisbilder und die untere die Pixelgewichte. Jeder Farbkanal stellt das Gewicht des Kanals dar. Die Gewichte werden auf ein Intervall von 0 bis 255 abgebildet. Zu sehen sind die Iteration k: $k = 1$ (a), $k = 25$ (b), $k = 50$ (c), $k = 75$ (d) und $k = 100$ (e). Auffällig ist, dass sich die Gewichte des Bartes kaum verändern. Das hat zur Folge, dass sich die Ergebnisbilder auch kaum verändern.

3.2.2 Koeffizienten

In diesem Abschnitt geht es um Auffälligkeiten, bei den in jeder Iteration errechneten Koeffizienten $\boldsymbol{c}^{(k)}$. Dazu wird $\boldsymbol{c}^{(k)}$ in jeder Iteration k als Streudiagramm ausgegeben. Die y-Achse deckt dabei einen Bereich von -1 bis 1 ab. Die x-Achse beschreibt, um welches Bild aus $\boldsymbol{V}$ es sich handelt. Die Ergebnisse sind in Abbildung 3.3 zu sehen. Man sieht, dass sich der Beitrag einzelner Bilder verändert. Es gibt dabei keine besonders auffälligen Ausreißer d. h. kein Koeffizient hat einen Wert, der sehr weit von den Werten der restlichen Koeffizienten abweicht. Es gibt also kein Bild, dass einen überproportionalen Anteil zum Ergebnisbild beiträgt.

3.2.3 Teilweise Rekonstruktion

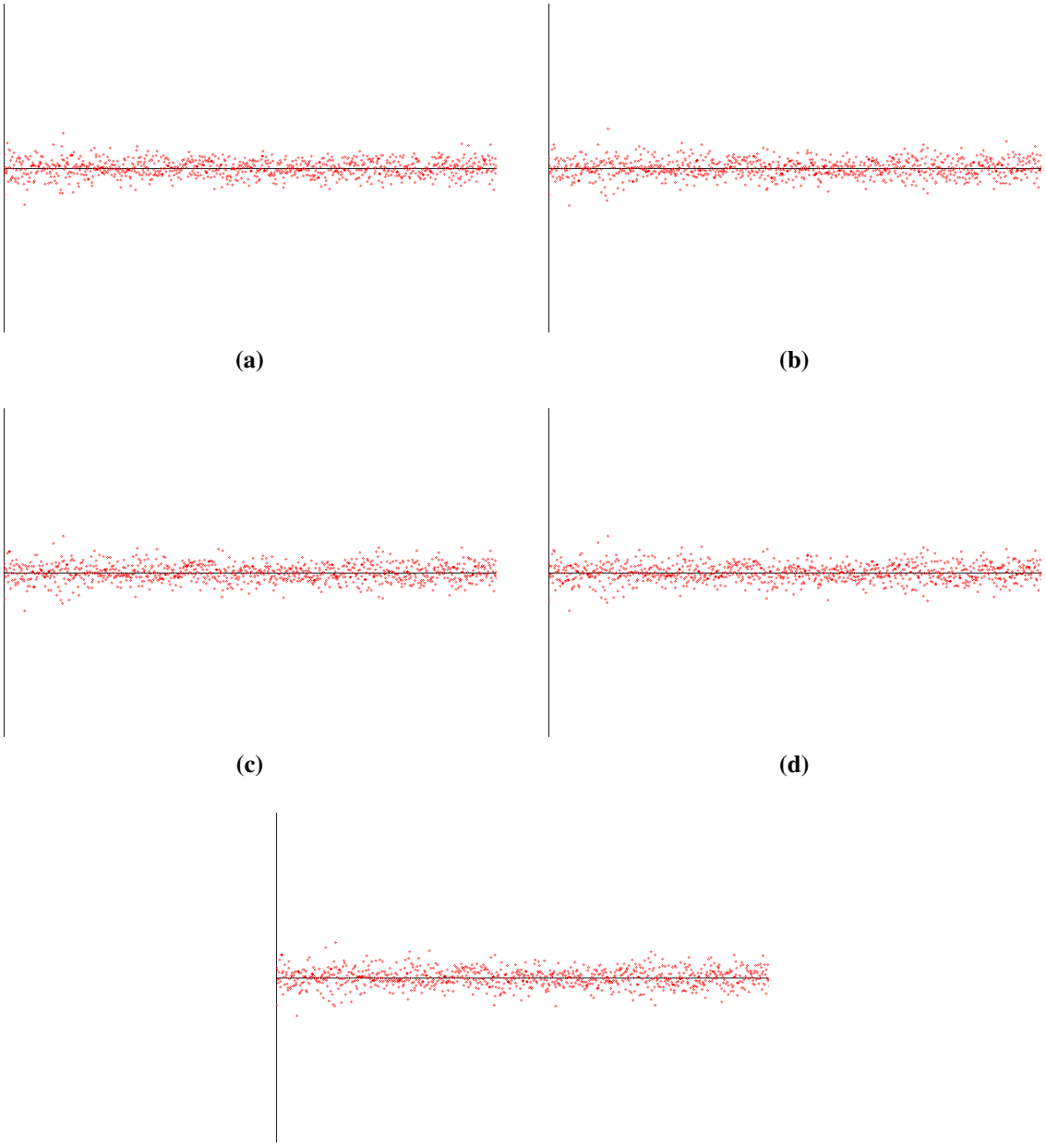
Hier wird überprüft, ob sich die Rekonstruktionsergebnisse verbessern, wenn man sich nur auf einen Teil des Gesichts beschränkt. Dafür setzt man alle Einträge von $\boldsymbol{W}$ auf 0, die nicht zu den oberen 35 Zeilen des Bildes gehören. Dadurch wird der untere Teil des Bildes ignoriert, da er keinen Beitrag zum Optimierungsproblem leistet. Wie man in Abbildung 3.4 sieht, ist das Rekonstruktionsergebnis des oberen Gesichtsteiles besser als der obere Gesichtsteil einer kompletten Rekonstruktion. Die Gewichte der Pixel im oberen Teil sind höher als bei der Rekonstruktion des gesamten Gesichts (vergleiche Abbildungen 3.2 und 3.4). Der untere Gesichtsteil sieht komplett anders aus, weil er bei der Rekonstruktion außer acht gelassen wurde. Der Bart wurde allerdings entfernt, was daran liegt, dass alle Gesichter in $\boldsymbol{V}$ keinen Bart haben und die Rekonstruktion, dessen nicht wichtig ist, da er sich im unteren Gesichtsteil befindet.

3.2.4 Entfernung von lokalen Modifikationen

In diesem Abschnitt geht es darum lokale Modifikationen aus Bildern zu entfernen. Dazu werden zunächst einige Bilder aus $\boldsymbol{V}$ ausgewählt und dann an einigen Stellen modifiziert. Das Experiment soll zeigen, dass der Algorithmus tatsächlich in der Lage ist Ausreißer zu entfernen. Wenn der Algorithmus korrekt funktioniert, werden die lokalen Modifikationen entfernt und das Gesicht perfekt rekonstruiert. Das soll passieren indem die entsprechenden unmodifizierten Bilder aus $\boldsymbol{V}$ ausgewählt werden. Wie man in Abbildung 3.5 sieht, funktioniert das auch.

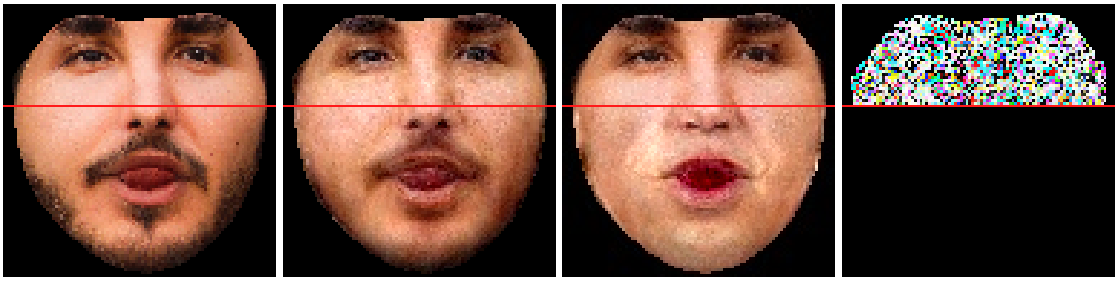
3.2.5 Den Unterraum der bartlosen Gesichter verkleinern

Dieser Abschnitt befasst sich mit der Verkleinerung von $\boldsymbol{V}$ d. h. was passiert, wenn der Raum den $\boldsymbol{V}$ abdeckt, verkleinert wird. Hierzu wurden zufällig 471 Gesichter aus $\boldsymbol{V}$ entfernt. Man würde hier erwarten, dass sich das Ergebnisse allgemein verschlechtern. Abbildung 3.6 zeigt aber, dass sich zwar die Gesichtsrekonstruktion verschlechtert, die Bartentfernung dafür aber besser funktioniert. Die Gewichtsveränderung verhält sich dabei entsprechend. Die Gewichte von Bartpixeln nehmen mit der Zeit ab, wie man in Abbildung 3.7 sieht. Das deutet darauf hin, dass der Algorithmus trotz allem korrekt implementiert ist.



(a) **(b)** **(c)** **(d)** **(e)**

Abbildung 3.3: Diese Abbildung zeigt, dass sich die Koeffizienten $\boldsymbol{c}^{(k)}$ über die Iterationen hinweg kaum ändern. Für die Iterationen $k = 1$ (a), $k = 25$ (b), $k = 50$ (c), $k = 75$ (d) und $k = 100$ (e) wurde ein Streudiagramm erstellt.



(a) **(b)** **(c)** **(d)**

Abbildung 3.4: Das Ergebnis einer teilweisen Rekonstruktion. Nur der Teil über der roten Linie ist der, der rekonstruiert wurde. Deshalb wird hier nur dieser Teil verglichen. Zu sehen sind das Ausgangsbild (a), das komplett rekonstruierte Gesicht (b), das zum Teil rekonstruierte Gesicht (c) und die Pixelgewichte in der letzten Iteration wenn nur der obere Teil rekonstruiert wird (d). Die Rekonstruktion der oberen Gesichtshälfte funktioniert hier besser und die Pixelgewichte sind höher als bei der kompletten Rekonstruktion.

Abbildung 3.5: Entfernung von lokalen Modifikationen. Die Bilder sind hierbei aus V und wurden lokal modifiziert. Die obere Zeile zeigt die Ausgangsbilder mit Modifikationen und die untere Zeile zeigt die Ergebnisbilder bei denen die Modifikationen entfernt wurden. Wie man sieht funktioniert die Entfernung dieser auch.

3.3 Ergebnis

Wie man sieht, funktioniert die Bartentfernung mit diesem Algorithmus nur dann besser als die Least-Squares-Methode (vergleiche Abbildungen 2.1 und 3.1), wenn V verkleinert wird. Das liegt daran, dass der Bart nicht schlechter rekonstruiert wird als der Rest des Gesichts. Deshalb werden Bartpixel nicht zu Ausreißern (siehe Abbildung 3.2). Wenn man den Unterraum V verkleinert, werden alle Pixel schneller zu Ausreißern. Das gilt hierbei aber vor allem für Bartpixel, da V keine bärtigen Gesichter enthält. Durch die Raumverkleinerung verschlechtert sich aber auch die Rekonstruktion des restlichen Gesichts.

Abbildung 3.6: Einige Ergebnisse der Iteratively-Re-Weighted-Least-Squares-Methode mit einer auf 472 Bilder verkleinerten Matrix $\boldsymbol{V}$. In der oberen Zeile befinden sich die Ausgangsbilder. In der mittleren Zeile befinden sich Ergebnisbilder mit einer normal großen Matrix und in der unteren Zeile befinden sich Ergebnisbilder mit einer verkleinerten Matrix $\boldsymbol{V}$. Man sieht, dass die Gesichtsrekonstruktion wie zu erwarten schlechter funktioniert als bei einer normal großen Matrix $\boldsymbol{V}$. Die Bartentfernung funktioniert dafür besser.

Unabhängig von der Größe von $\boldsymbol{V}$ werden Nasen, Augen und Münder wie bei der Least-Squares-Methode schlecht rekonstruiert, da sie stark von Mensch zu Mensch variieren. Sie können also mit hoher Wahrscheinlichkeit nur schlecht in $\boldsymbol{V}$ rekonstruiert werden, wenn keine Gesichter mit ähnlichen Augen und Nasen vorhanden sind. Dadurch werden auch sie zu Ausreißern und damit unwichtig für die Rekonstruktion. Zusätzlich kann ein Durchlauf des Algorithmus mehrere Minuten dauern und es lässt sich nicht vorhersagen, wie viele Iterationen für ein gegebenes Bild notwendig sind. Bei der Durchführung des Algorithmus gab es Fälle bei denen bis zu 2000 Iterationen notwendig waren, bis $|\boldsymbol{c}^{(k)} - \boldsymbol{c}^{(k-1)}|$ gegen 0 konvergiert ist.

(a) **(b)** **(c)** **(d)** **(e)**

Abbildung 3.7: Man sieht hier, wie sich das Ergebnisbild und die Gewichtungsmatrix $\boldsymbol{W}$ über die Iterationen hinweg verändern. Die Matrix $\boldsymbol{V}$ enthält hier nur 472 Bilder. Die obere Zeile zeigt die Ergebnisbilder und die untere die Pixelgewichte. Jeder Farbkanal stellt das Gewicht des Kanals dar. Die Gewichte werden auf ein Intervall von 0 bis 255 abgebildet. Zu sehen sind die Iteration k: $k = 1$ (a), $k = 25$ (b), $k = 50$ (c), $k = 75$ (d) und $k = 100$ (e). Hier verringert sich das Gewicht des Bartes. Deshalb verschwindet der Bart hier auch fast komplett.

4 Bartentfernung durch das Schichtenmodel

In diesem Kapitel geht es um den letzten Algorithmus aus [NLET08]. Die Iteratively-Re-Weighted-Least-Squares-Methode aus Kapitel 3 ist weder in der Lage, Bärte komplett zu entfernen noch den Rest des Gesichts zu rekonstruieren. Der Algorithmus in diesem Kapitel soll das besser machen. Hier wird das Gesicht komplett rekonstruiert und der Bart besser entfernt. Dabei sollen auch markante Strukturen, wie Muttermale und Narben erhalten bleiben. Dazu sollen nun nicht mehr nur die bartlosen Stellen rekonstruiert werden, sondern das gesamte Gesicht soll komplett durch 3 Schichten rekonstruiert werden. Diese Schichten werden komplett unabhängig voneinander modelliert. Bei den Schichten handelt es sich um die Gesichtsschicht, die Bartschicht und die Sonstigesschicht. So soll der Bart vom Gesicht trennbar gemacht werden.

4.1 Algorithmus

Wie bereits erwähnt betrachtet man das Gesicht nun als ein Schichtenmodell:

$$\boldsymbol{x} = \boldsymbol{V}\hat{\boldsymbol{\alpha}} + \boldsymbol{B}\hat{\boldsymbol{\beta}} + \bar{\boldsymbol{x}} \tag{4.1}$$

Der Vektor $\boldsymbol{V}\hat{\boldsymbol{\alpha}}$ ist die Gesichtsschicht, $\boldsymbol{B}\hat{\boldsymbol{\beta}}$ ist die Bartschicht und $\bar{\boldsymbol{x}}$ die Sonstigesschicht. Das bärtige Canonical Face ohne Bart $\boldsymbol{x}^*$ wird folgendermaßen berechnet:

$$\boldsymbol{x}^* = \boldsymbol{V}\hat{\boldsymbol{\alpha}} + \bar{\boldsymbol{x}} \tag{4.2}$$

Man addiert also nur die Gesichts- und die Sonstigesschicht und lässt dabei die Bartschicht weg.

Bei $\boldsymbol{B}$ handelt es sich um den Bartraum, welcher ausschließlich in der Lage ist Bärte zu rekonstruieren. Um $\boldsymbol{B}$ zu erstellen wird zunächst ein Datensatz aus bärtigen Canonical-Faces erstellt. Dann berechnet man mit dem Algorithmus aus Kapitel 3 zu jedem Gesicht $\boldsymbol{x}_i$ das bartlose Gesicht $\boldsymbol{x}_i^*$. Daraus erstellt man den Datensatz $\hat{\boldsymbol{B}}^{d\times n}$ folgendermaßen:

$$\hat{\boldsymbol{B}} = \left[(\boldsymbol{x}_1 - \boldsymbol{x}_1^*)...(\boldsymbol{x}_n - \boldsymbol{x}_n^*)\right] \tag{4.3}$$

Die Matrix $\hat{\boldsymbol{B}}$ enthält also theoretisch nur Bärte. Abbildung 4.1 zeigt einige extrahierte Bärte. Da der Algorithmus aus Kapitel 3 allerdings nicht in der Lage ist, das Gesicht exakt zu rekonstruieren, enthalten die Gesichter aus $\hat{\boldsymbol{B}}$ viel Rauschen. Zusätzlich enthält das Differenzbild die markante Gesichtsstrukturen der einzelnen Gesichter, da diese nicht rekonstruiert werden konnten. Um diese Strukturen zu entfernen, führt man eine PCA (siehe Abschnitt 1.1.6) auf dem Datensatz $\hat{\boldsymbol{B}}$ durch. Der Datensatz $\hat{\boldsymbol{B}}$ besteht hier aus 943 Bildern.

Man erhält dabei 95% der Energie. Das bedeutet, dass man so viele Prinicipal Components wählt, so dass folgendes gilt:

$$0.95 \approx \frac{\text{Summe der } \lambda\text{, der gewählten Prinicipal Components}}{\text{Summe aller } \lambda} \tag{4.4}$$

Abbildung 4.1: Einige extrahierte Bärte aus $\hat{\boldsymbol{B}}$. Die obere Zeile zeigt die Ausgangsbilder. In der Mitte sieht man die entsprechenden Ausgabebilder der Iteratively-Re-Weighted-Least-Squares-Methode. In der unteren Zeile sieht man die extrahierten Bärte.

Man beginnt dabei mit dem ersten Prinicipal Component. Das Rauschen kann auf diese Weise aber nicht aus dem Datensatz entfernt werden, da es in allen Bildern auftaucht. Das Ergebnis der PCA sind der Datensatz $\boldsymbol{B}$ welcher aus den gewählten Principal-Components besteht und der Meanbart $\boldsymbol{\mu}$. Beim Meanbart handelt es sich um den Mittelwert aller Datenpunkte aus $\hat{\boldsymbol{B}}$. Abbildung 4.2 zeigt die ersten 6 Prinicipal Components von $\boldsymbol{B}$. Zur Darstellung wurden sie so skaliert, dass sie sichtbar sind und mit $\boldsymbol{\mu}$ und dem Durchschnittsgesicht aus $\boldsymbol{V}$ addiert. Man sieht, dass nicht alle Prinicipal Components sichtbare Bärte enthalten, sondern auch andere Strukturen, wie Brillen, Augen und Nasen. Zudem enthalten sie viel Rauschen, da das in fast allen der zuvor berechneten Differenzbilder vorhanden ist.

Da es hier um die genaue Rekonstruktion geht, stellt man wie in Kapitel 2 ein Least-Squares-Problem auf:

$$\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\arg\min} \left(\|\boldsymbol{x} - \boldsymbol{\mu} - \boldsymbol{V}\boldsymbol{\alpha} - \boldsymbol{B}\boldsymbol{\beta}\|_2^2 + \gamma\|\boldsymbol{\alpha}\|_2^2 + \gamma\|\boldsymbol{\beta}\|_2^2 \right) \tag{4.5}$$

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\arg\min} (\|\boldsymbol{x} - \boldsymbol{\mu} - \begin{bmatrix} \boldsymbol{V} & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \|_2^2 + \gamma \| \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \|_2^2) \tag{4.6}$$

$$= \left(\begin{bmatrix} \boldsymbol{V}^T \\ \boldsymbol{B}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{V} & \boldsymbol{B} \end{bmatrix} + \gamma \boldsymbol{I} \right)^{-1} \begin{bmatrix} \boldsymbol{V}^T \\ \boldsymbol{B}^T \end{bmatrix} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{4.7}$$

Abbildung 4.2: Die ersten 6 Principal-Components von $\boldsymbol{B}$.

Zur Lösung kann wieder, wie in Kapitel 2 die Choleski-Zerlegung verwendet. Falls die Matrix $\begin{bmatrix} V^T \\ B^T \end{bmatrix} [V \quad B]$ positiv semidefinit ist, kann man wieder die Thikonov-Regularisierung verwenden. Dazu wählt man für γ wieder den Wert 0,1. Wenn die Regularisierung nicht notwendig ist, wird für γ wieder 0 gewählt. Als Alternative zur Choleski-Zerlegung, kann man auch wieder die Singulärwertzerlegung verwenden, welche allerdings eine längere Berechnungsdauer hat.

Nachdem man die Koeffizienten $\hat{\boldsymbol{\alpha}}$ und $\hat{\boldsymbol{\beta}}$ berechnet hat kann man die Sonstigesschicht $\bar{\boldsymbol{x}}$ berechnen:

$$\bar{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{V}\hat{\boldsymbol{\alpha}} - \boldsymbol{B}\hat{\boldsymbol{\beta}} \tag{4.8}$$

Die berechneten Koeffizienten und Schichten setzt man nun in Gleichung (4.2) ein und berechnet damit das bartlose Gesicht $\boldsymbol{x}^*$.

4.2 Ergebnis

Abbildung 4.3 zeigt einige Ergebnisse dieses Algorithmus. Wie man sieht, wird der Bart auch hier nicht entfernt, sondern nur heller. Das liegt daran, dass die Bartschicht nicht in der Lage ist, den Bart komplett zu rekonstruieren. Sowohl die Sonstiges- als auch die Gesichtsschicht enthalten Bartrückstände, die dann im Ergebnisbild addiert werden. Dadurch erscheint ein heller Restbart im Ergebnisbild. Ein möglicher Grund hierfür ist, dass der Unterraum $\boldsymbol{B}$ viel Rauschen enthält, welches die Rekonstruktion der Bartschicht negativ beeinflusst. Das Rauschen ist auch in den Principal-Components sichtbar, was man in Abbildung 4.2 sehen kann. Wenn der Bart komplett

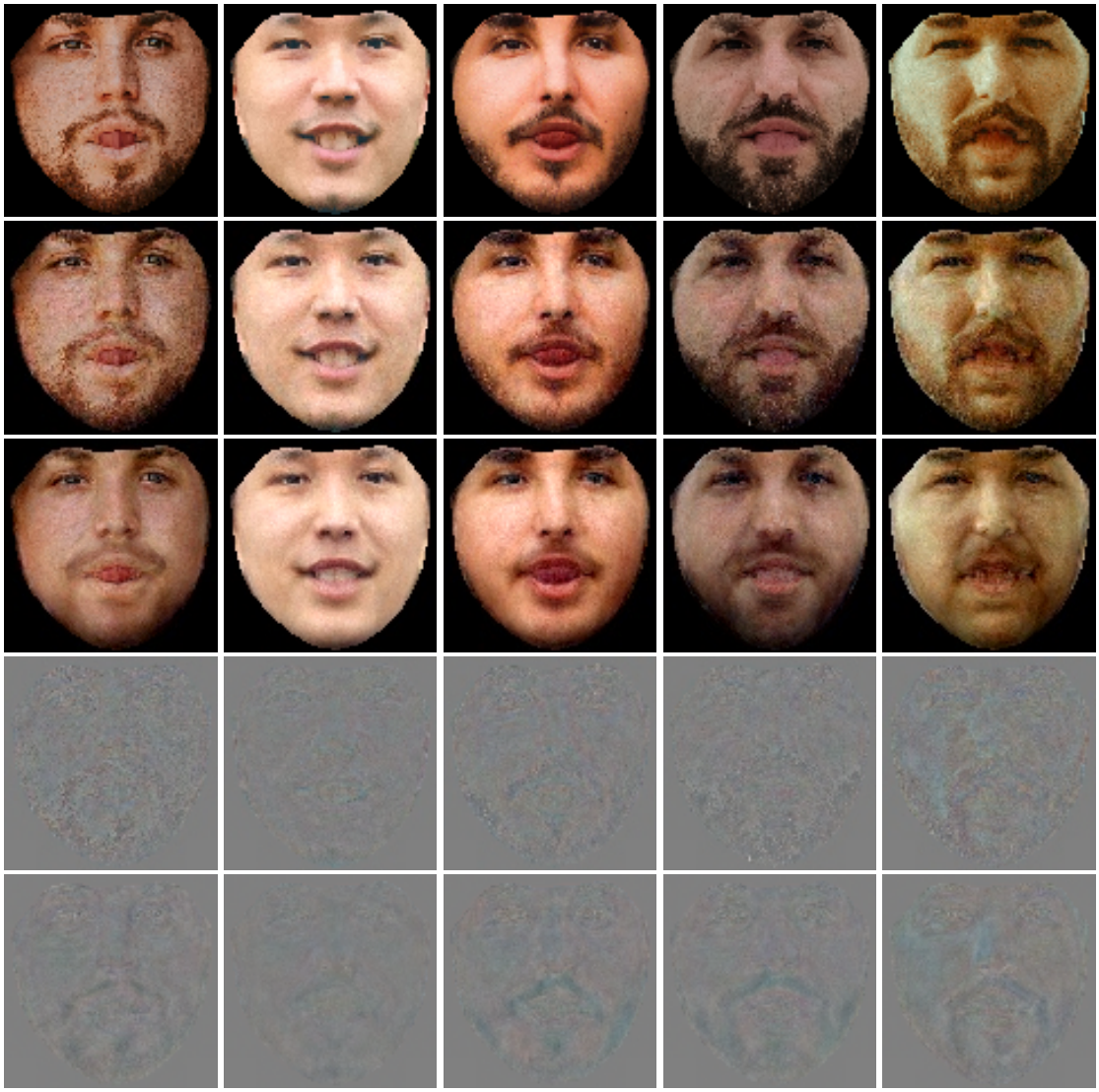


Abbildung 4.3: Einige Ergebnisse der Bartentfernung durch das Schichtenmodell. Die oberste Zeile zeigt die Originalbilder. Die zweite Zeile zeigt die Ergebnisbilder. Die dritte Zeile zeigt die Gesichtsschicht. Die vierte Zeile zeigt die Sonstigesschicht und die unterste Zeile zeigt die Bartschicht. Man sieht, dass sowohl die Gesichts- als auch die Sonstigesschicht Teile des Bartes enthalten. Deshalb taucht der Bart im Ergebnisbild wieder auf. Dennoch sieht das Ergebnis besser aus als die Ergebnisse der vorherigen Algorithmen.

rekonstruiert werden würde, muss das dadurch aufaddierte Rauschen, durch die Gesichtsschicht ausgeglichen werden, um das Optimierungsproblem zu lösen. Dadurch würde das restliche Gesicht schlechter rekonstruiert werden, was ein suboptimales Ergebnis zur Folge hätte. Deshalb wird der Bart zum Teil von der Gesichtsschicht rekonstruiert oder es wird auf eine exakte Rekonstruktion dessen verzichtet.

5 Verbesserungen

Wie man in Kapitel 4 bereits sah, funktioniert der letzte Algorithmus nicht so gut wie erwartet. Das liegt daran, dass die Iteratively-Re-Weighted-Least-Squares-Methode (siehe Kapitel 3) Bärte nur schlecht entfernt. Deshalb sind extrahierte Bärte kaum sichtbar und enthalten viel Rauschen (siehe Abbildung 4.1). Die Principal-Components reflektieren das auch (siehe Abbildung 4.2). Der Bartraum $\boldsymbol{B}$ kann deshalb kaum als Bartraum bezeichnet werden. In diesem Kapitel geht es deshalb darum, aus $\boldsymbol{B}$ einen echten Bartraum zu machen.

5.1 Modifikationen der Iteratively-Re-Weighted-Least-Squares-Methode

Abschnitt 3.2 zeigt, dass der Algorithmus korrekt implementiert wurde. Deshalb werden in diesem Abschnitt drei Modifikationen vorgestellt, die das Extraktionsergebnis verbessern. Die ersten beiden Modifikationen ändern die Art wie die Gewichtungsmatrix $\boldsymbol{W}$ initialisiert wird und die Letzte ändert die Art wie $\boldsymbol{W}$ berechnet wird. Alle drei Modifikationen erreichen dadurch ein besseres Rekonstruktions- und somit auch ein besseres Extraktionsergebnis.

5.1.1 Mean-Face-Initialisierung (Durchschnittsgesicht-Initialisierung)

Bei der Mean-Face-Initialisierung wird $\boldsymbol{W}$ nicht mit der Einheitsmatrix initialisiert, sondern die Initialisierung basiert auf der Differenz $\boldsymbol{r}$ zum Mean-Face von $\boldsymbol{V}$. Das Mean-Face kann mit Gleichung (1.26) berechnet werden und ist in Abbildung 5.1 zu sehen. Dann setzt man diese Differenz in Gleichung (3.6) und Gleichung (3.7) ein und berechnet die Einträge von $\boldsymbol{W}$. In den ersten Iterationen ist das Ergebnisbild sehr ähnlich zum Mean-Face. Mit weiteren Iterationen nähert es sich aber immer weiter $\boldsymbol{x}$ an. Der Bart ist bereits nach der ersten Iteration entfernt, da das Mean-Face keinen besitzt. Die Differenz ist an den Bartstellen also groß, weshalb $\boldsymbol{W}$ ein niedriges Gewicht an diesen Stellen hat. Stellen von $\boldsymbol{x}$, die ähnlich zum Mean-Face sind, bekommen ein hohes Gewicht von $\boldsymbol{W}$. Deshalb sind die Ergebnisbilder in den ersten Iterationen auch sehr ähnlich zum Mean-Face. Abbildung 5.2 zeigt einige Ergebnisse dieser Modifikation. Der Bart wird hier wesentlich besser entfernt als beim Standardalgorithmus, aber auch hier gehen markante Merkmale verloren. Es fällt auf, dass diese Merkmale dem Mean-Face sehr ähnlich sind.

Abbildung 5.1: Das Mean-Face von $\boldsymbol{V}$.

Abbildung 5.2: Einige extrahierte Bärte. Die obere Zeile zeigt die Ausgangsbilder. In der Mitte sieht man die entsprechenden Ausgabebilder der Iteratively-Re-Weighted-Least-Squares-Methode mit der Mean-Face-Initialisierung. In der unteren Zeile sieht man die extrahierten Bärte.

5.1.2 Nearest-Neighbour-Initialisierung (Nächster-Nachbar-Initialisierung)

Bei dieser Art der Initialisierung von $\boldsymbol{W}$ wird im Gegensatz zur Mean-Face-Initialisierung nicht die Differenz zum Mean-Face sondern die Differenz zum nächsten Nachbarn $\boldsymbol{q}$ aus $\boldsymbol{V}$ genutzt. Der nächste Nachbar wird folgendermaßen berechnet:

$$\hat{m} = \arg\min_{m} \|\boldsymbol{x} - \boldsymbol{v}_m\|_1 \tag{5.1}$$

$$\boldsymbol{q} = \boldsymbol{v}_{\hat{m}} \tag{5.2}$$

Die Gewichtungsmatrix $\boldsymbol{W}$ wird nun wie in Abschnitt 5.1.1 berechnet indem man die Differenz in Gleichung (3.6) und Gleichung (3.7) einsetzt. Auch hier sieht das Gesicht zu Beginn so aus wie das Gesicht, zu dem die Differenz gebildet wurde. Die Gründe hierfür sind dieselben wie in

Abbildung 5.3: Einige extrahierte Bärte. Die obere Zeile zeigt die Ausgangsbilder. In der Mitte sieht man die entsprechenden Ausgabebilder der Iteratively-Re-Weighted-Least-Squares-Methode mit der Nearest-Neighbour-Initialisierung. In der unteren Zeile sieht man die extrahierten Bärte.

Abschnitt 5.1.1. Die Ergebnise nähern sich auch mit späteren Iterationen an $\boldsymbol{x}$ an. Auffällig ist, dass zu Beginn $\boldsymbol{q}$ den größten Beitrag zum Ergebnisbild liefert. Das kommt daher, dass mit dem Bild $\boldsymbol{q}$ die Stellen gut rekonstruiert werden können, die in $\boldsymbol{x}$ bereits ähnlich zu $\boldsymbol{q}$ sind. Abbildung 5.3 zeigt einige Ergebnisse dieser Modifikation. Auch hier gehen markante Merkmale wieder verloren, da sie sich dem Nächsten-Nachbarn angleichen. Die Bartentfernung funktioniert ähnlich gut wie bei der Mean-Face-Initialisierung.

5.1.3 Geman-McClure mit 0.8-Quantil

Wie man in den Abbildungen 3.2 und 3.7 sieht, werden nicht nur Bartpixel zu Ausreißern sondern auch andere Pixel. Welche Pixel zu Ausreißern werden, kann mit der Geman-McClure Funktion (siehe Gleichung (1.37) und Abbildung 1.3) bestimmt werden. Damit Pixel nicht so schnell zu Ausreißern werden, muss σ größer gewählt werden. Der Wert von σ wird normalerweise mit Gleichung (3.7) berechnet. Hier soll der Wert aber folgendermaßen berechnet werden:

$$\sigma = 1.4826 \cdot 0.8\text{-}Quantil(|r_i| : \forall i \in [i, d]) \tag{5.3}$$

Abbildung 5.4: Einige extrahierte Bärte. Die obere Zeile zeigt die Ausgangsbilder. In der Mitte sieht man die entsprechenden Ausgabebilder der Iteratively-Re-Weighted-Least-Squares-Methode mit der Berechnung des 0.8-$Quantils$ anstatt des Medians. In der unteren Zeile sieht man die extrahierten Bärte.

Die Initialisierung von $\boldsymbol{W}$ ändert sich hier nicht. Man initialisiert $\boldsymbol{W}$ also mit der Einheitsmatrix. Ein p-$Quantil(\boldsymbol{x})$ wird folgendermaßen berechent:

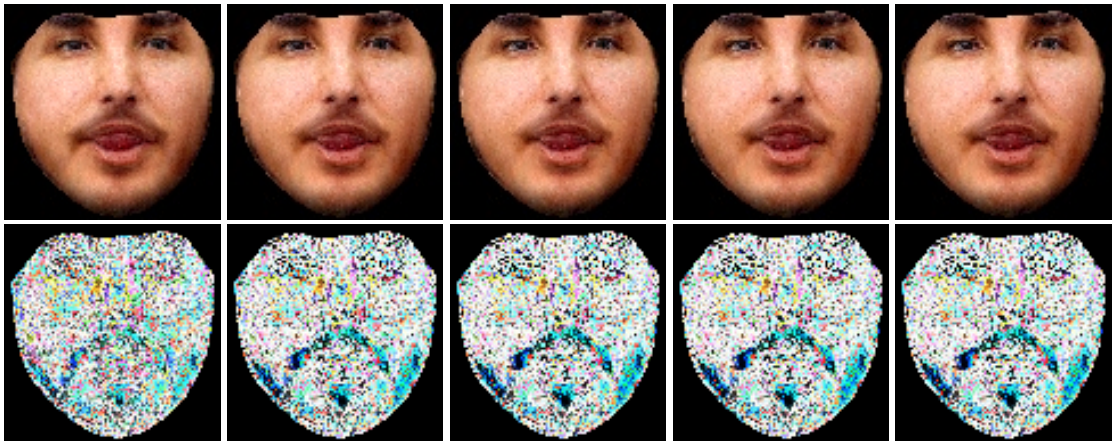
$$n = AnzahlEinträge(\boldsymbol{x}) \tag{5.4}$$

$$p\text{-}Quantil(\boldsymbol{x}) = \begin{cases} \frac{\boldsymbol{x}_{np}+\boldsymbol{x}_{np+1}}{2} & \text{, falls } np \text{ ganzzahlig} \\ \boldsymbol{x}_{\lfloor np \rfloor+1} & \text{, falls } np \text{ nicht ganzzahlig} \end{cases} \tag{5.5}$$

Abbildung 5.4 zeigt einige Ergebnisse. Man sieht, dass die Bartentfernung und somit auch die Bartextraktion mit der Modifikation besser funktioniert als ohne die Modifikation (vergleiche Abbildung 4.1). Wie zu erwarten werden weniger Pixel zu Ausreißern (siehe Abbildung 5.5). Charakteristische Gesichtsmerkmale werden besser erhalten als bei den anderen beiden Modifikationen, gehen aber dennoch mit der Zeit verloren. Ein weiterer Vorteil dieser Modifikation ist, dass die Berechnung wesentlich schneller konvergiert. Die Bartentfernung selbst funktioniert allerdings schlechter als bei den anderen beiden Methoden.

5.2 Entrauschung des Bartraums

Die Modifikationen aus Abschnitt 5.1 haben das Extraktionsergebnis verbessert. Der Bart ist im Differenzbild fast immer gut sichtbar. Dennoch enthalten die Bilder immer noch viel Rauschen. In diesem Kapitel geht es, darum dieses zu entfernen. Da sich morphologische Operationen zur



(a) (b) (c) (d) (e)

Abbildung 5.5: Hier sieht man wie sich das Ergebnisbild und die Gewichtungsmatrix $\boldsymbol{W}$ über die Iterationen hinweg verändern, wenn man das $0.8 - Quantil$ anstelle des Meadians berechnet. Die obere Zeile zeigt die Ergebnisbilder und die untere die Pixelgewichte. Jeder Farbkanal stellt das Gewicht des Kanals dar. Die Gewichte werden auf ein Intervall von 0 bis 255 abgebildet. Da weniger Iterationen bis zu Konvergenz notwendig sind als bei der Berechnung des Medians sieht man hier nur die Iteration k: $k = 1$ (a), $k = 8$ (b), $k = 16$ (c), $k = 24$ (d) und $k = 32$ (e). Hier verringert sich das Gewicht des Bartes. Deshalb verschwindet der Bart hier auch fast komplett.

Entrauschung eignen wird als Erstes auf diese eingegangen. Dann geht es um ein Schwellenwertverfahren, welches Pixel auf $(0, 0, 0)$ setzt, wenn deren Summe der Farbkanalbeträge kleiner als ein berechneter Schwellenwert ist. Der Schwellenwert wird dabei gefunden, indem ein Quantil berechnet wird. Als Letztes geht es dann um den Grab Cut. Das Verfahren dient dazu den Vordergrund eines Bildes vom Hintergrund zu segmentieren. Diese Verfahren werden auf die extrahierten Bärte angewendet, bevor die PCA durchgeführt wird. Bärte werden in den kommenden Abschnitten mit der Iteratively-Re-Weighted-Least-Squares-Methode in Kombination mit der Mean-Face-Initialisierung extrahiert. Große Teile der dabei entstehenden Bilder enthalten immer Informationen, die für die Bartrekonstruktion nicht relevant sind. Deshalb wird eine Maske über das Bild gelegt, die nur die relevanten Bildteile erhält. Die Maske ist in Abbildung 5.6 zu sehen und wird in den kommenden Abschnitten immer vor dem eigentlichen Verfahren angewendet. Dadurch wird bereits ein großer Teil des Rauschens auf einmal entfernt. Abbildung 5.7 zeigt, wie sich die Maske auf extrahierte Bärte auswirkt. Abbildung 5.8 zeigt, dass die Maske allein bereits einen starken positiven Effekt auf $\boldsymbol{B}$ und somit auch auf die Bartentfernung hat.

5.2.1 Morphologische Operationen

In diesem Abschnitt werden morphologische Operationen auf die extrahierten Bärte angewendet, um Rauschen zu entfernen, da es sich bei diesen um Standardverfahren zur Entrauschung handelt. Abbildung 5.9 zeigt einige Ergebnisse. Sowohl bei der Dilation als auch bei der Erosion wird das Rauschen nicht veringert. Das wirkt sich wie zu erwarten auf die Bartentfernung aus. Bei der Entfernung wird der Bart weiterhin nur heller, was zum Großteil aber an der Maske liegt. Die Erosion

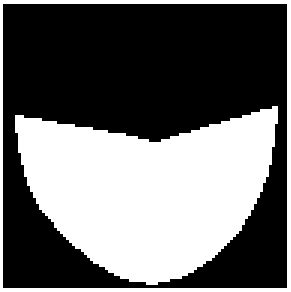


Abbildung 5.6: Die Gesichtsmaske, die nur den Teil des Gesichts erhält, der für den Bart relevant ist. Der erhaltene Teil ist weiß.

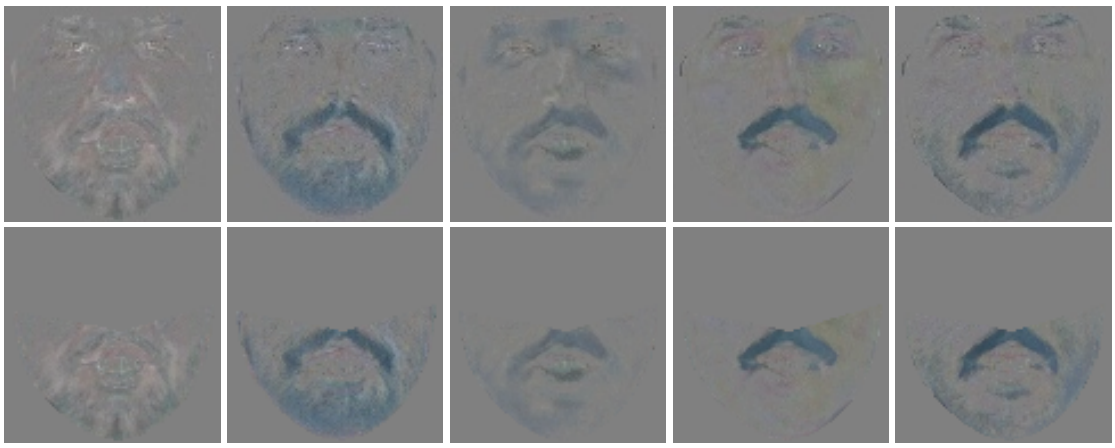


Abbildung 5.7: Hier sieht man, wie sich die Maske aus Abbildung 5.6 auf extrahiere Bärte auswirkt.

Abbildung 5.8: Einige Ergebnisse der Bartentfernung. Über die Bärte aus denen $\boldsymbol{B}$ besteht, wurde vor der PCA die Maske aus Abbildung 5.6 gelegt.

und die Dilation führen sogar zu einer kleinen Verschlechterung. Beim Opening und Closing wird das Rauschen zwar immer noch nicht entfernt, aber es wird in den meisten Fällen heller. Deshalb kann die Bartentfernung dadurch etwas verbessert werden. Einige Bartentfernungsergebnisse sind in Abbildung 5.10 zu finden.

5.2.2 Schwellenwertverfahren

In diesem Abschnitt wird ein einfaches Schwellenwertverfahren vorgestellt. Hierbei wird das Differenzbild $\boldsymbol{X}_{\text{diff}}$ folgendermaßen in einen Vektor $\boldsymbol{p}$ umgeformt:

$$\begin{bmatrix} (a_b, a_g, a_r) & (b_b, b_g, b_r) \\ (c_b, c_g, c_r) & (d_b, d_g, d_r) \end{bmatrix} \rightarrow \begin{bmatrix} |a_b| + |a_g| + |a_r| \\ |b_b| + |b_g| + |b_r| \\ |c_b| + |c_g| + |c_r| \\ |d_b| + |d_g| + |d_r| \end{bmatrix} \tag{5.6}$$

Nun wird das 0.8-$Quantil(\boldsymbol{p})$ berechnet und als Schwellenwert s genutzt. Der Schwellenwert wird so gewählt, da ein Großteil des Bildes aus einem schwarzen Rand besteht, der perfekt rekonstruiert wurde. Diese Randpixel haben im Differenzbild also den Wert $(0, 0, 0)$. Da es sich hierbei um viele Pixel handelt, sorgen sie dafür, dass der Median einen kleinen Wert hat. Ein zu kleiner Wert ist allerdings nicht groß genug, um ausreichend Rauschen zu entfernen. Der Schwellenwert s wird nun folgendermaßen auf das Ausgangsbild angewendet:

$$x_{ij} = \begin{cases} (0,0,0) & , (|x_{ij_b}| + |x_{ij_g}| + |x_{ij_r}|) < s \\ x_{ij} & , \text{sonst} \end{cases} \tag{5.7}$$

Man sieht in Abbildung 5.9 das immer noch viel Rauschen im Bild zurückbleibt, welches dann auch im Unterraum $\hat{\boldsymbol{B}}$ vorhanden ist. Wie man in Abbildung 5.10 sieht hat auch hier die Maske den größten Effekt, währrend das Schwellenwertverfahren kaum zu einer Verbesserung führt.

5.2.3 Grab Cut

Pixel gehören mit hoher Wahrscheinlichkeit zum Bart, wenn die meisten Nachbarpixel auch zum Bart gehören. In [NLET08] wird ein Nachbearbeitungsschritt vorgestellt, der diese Tatsache nutzt um Pixel auf $(0, 0, 0)$ zu setzen die nicht zum Bart gehören. Dazu wird ein Optimierungsproblem aufgestellt, welches dann mittels Graph Cuts effizient gelöst werden kann. In dieser Arbeit wird stattdessen ein Verfahren Namens Grab Cut [RKB04] verwendet. Der Grab Cut dient dazu den Vordergrund eines Bildes von dessen Hintergrund zu trennen. Da der Bart im Differenzbild meistens klar zu erkennen ist und sich vom restlichen bild abhebt, kann der Grab Cut genutzt werden um den Bart zu segmentieren. Dieses Verfahren ist iterativ und das Ergebnis kann durch Nutzereingaben verbessert werden. Der Grab Cut wird hier allerdings komplett automatisch ausgeführt, da die Segmentierung mit Nutzereingaben zu aufwendig wäre. Es werden 4 Iteration durchgeführt, da experimentell ermittelt wurde, dass mehr Iterationen kein besseres Ergebnis bei der Bartsegmentierung liefern. In Abbildung 5.9 sieht man, dass die Segmentierung auf diesem Weg nicht immer funktioniert. Das führt dazu, dass $\boldsymbol{B}$ dennoch Rauschen enthält. Deshalb verbesseren sich die Ergebnisse hiermit auch kaum, wie Abbildung 5.10 zeigt. Wenn man den Grab Cut mit Nutzereingaben durchführt, kann man aber sehr gute Ergebnisse erzielen, was sich dann auch positiv auf die Bartentfernung auswirken würde.

5.2.4 Ergebnis

Die Abbildungen 5.9 und 5.10 zeigen Beispiele, an denen man sehen kann wie sich die einzelnen Verfahren auf das Entrauschen der Bilder, die den Unterraum $\boldsymbol{B}$ bilden und die anschließende Bartentfernung auswirken. Das Entrauschen funktioniert nicht bei allen Verfahren gut. Die Bartentfernung funktioniert zwar immer noch nicht perfekt, aber der Bart wird wesentlich heller, was zum Großteil an der Maske liegt, die vor der Anwendung der einzelnen Verfahren über jeden Bart gelegt wird. Sowohl die Dilation als auch die Erosion in Kombination mit der Maske liefern ein schlechteres Ergebnis als die alleinige Anwendung der Maske. Das Schwellenwerverfahren und der Grab Cut in Kombination mit der Maske liefern ähnliche Ergebnisse, wie die alleinige Anwendung der Maske. Die Verfahren Opening und Closing liefern hier in Kombination mit der Maske die besten Ergebnisse. Auffällig ist, dass sich vor allem die Farbe des Mundes ändert. Das liegt daran, dass Münder oft in $\boldsymbol{B}$ zu finden sind, da keins der Verfahren diese entfernen kann. Die Bartschicht trägt deshalb oft zu Rekonstruktion des Mundes bei. Das Entfernungsergebnis kann aber wahrscheinlich weiter verbessert werden, wenn man den Grab Cut mit Nutzereingaben durchführt. Dadurch können alle nicht-Bart-Pixel entfernt werden, was zu einer Verbesserung des Bartentfernungsergebnisses führen sollte.

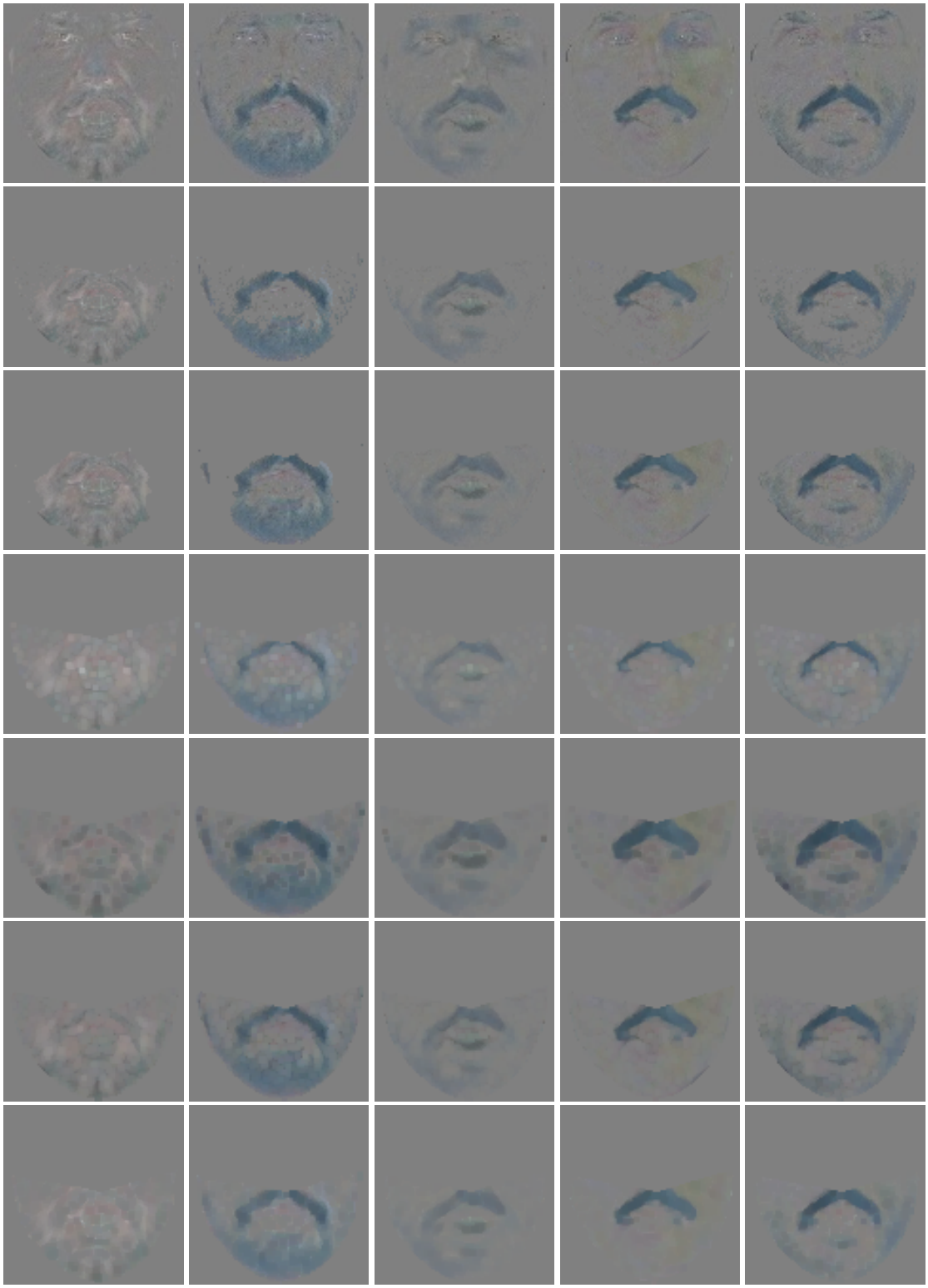


Abbildung 5.9: Hier werden die Auswirkungen, der vorgestellten Verfahren auf extrahierte Bärte gezeigt. Die erste Zeile zeigt die extrahierten Originalbärte. Die darauf folgenden Zeilen zeigen die Auswirkungen des Schwellenwertverfahrens, des Grab Cuts, der Dilation, der Erosion, des Openings und des Closings auf die extrahierten Bärte.

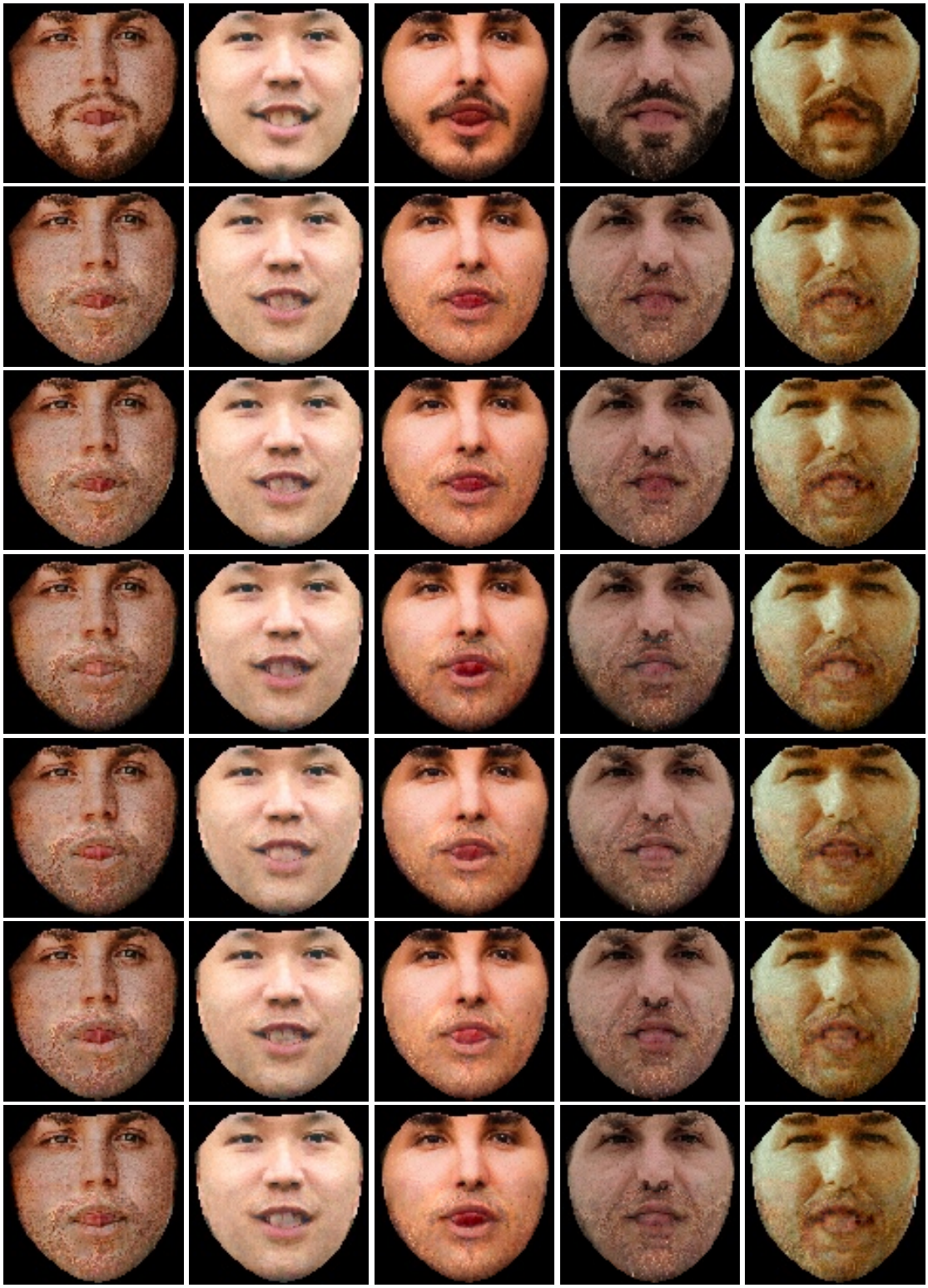


Abbildung 5.10: Hier sieht man, wie sich die Entrauschung von B mit den vorgestellten Verfahren auf die Bartentfernung auswirkt. Die erste Zeile zeigt das Originalbild. Die darauf folgenden Zeilen zeigen die Auswirkungen des Schwellenwertverfahrens, des Grab Cuts, der Dilation, der Erosion, des Openings und des Closings auf die Bartentfernung.

Abbildung 5.11: Einige Ergebnisse der Bartentfernung durch das Schichtenmodel. Die Bilder, aus denen die Matrix $\boldsymbol{B}$ besteht, wurden zuvor mit dem Opening Verfahren entrauscht.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurden die Bartentfernungsalgorithmen aus dem Paper „Image-based Shaving“ [NLET08] implementiert und evaluiert. Kapitel 1 beschäftigt sich ausführlich sowohl mit mathematischen als auch mit Computer-Vision-Grundlagen. Alle Algorithmen haben die Grundidee, das bärtige Gesicht im Unterraum der bartlosen Gesichter V (siehe Abschnitt 1.5) zu rekonstruieren. Es ging also darum, gute Koeffizienten für eine Linearkombination, zu finden. Der erste dieser Algorithmen (siehe Kapitel 2) versucht, diese Koeffizienten durch das Lösen eines Least-Squares-Problems zu finden. Dieser Ansatz funktioniert so allerdings nicht, da der Bart einen großen Teil des Gesichts darstellt. Er hat also einen großen Einfluss auf das Least-Squares-Problem. Das führt dazu, dass der Bart aus bartlosen Gesichtern rekonstruiert werden muss und die Koeffizienten entsprechend gewählt werden. Der Bart wird im Ergebnisbild dadurch nur heller. Dieses Ergebnis ist nicht zufriedenstellend. Deshalb wurde ein zweiter Algorithmus (siehe Kapitel 3) implementiert. Der zweite Algorithmus löst einen „M-Estimator-of-Regression“ mit der Iteratively-Re-Weighted-Least-Squares-Methode, um die gesuchten Koeffizienten zu finden. Dabei wird die Rekonstruktion von bereits gut rekonstruierten Pixel immer wichtiger, während die Rekonstruktion von schlecht rekonstruierten Pixel immer unwichtiger wird. Da Bartpixel nur schlecht rekonstruiert werden können, werden diese theoretisch mit der Zeit niedriger gewichtet als der Rest des Gesichts. Das gleiche gilt allerdings auch für markante Teile des Gesichts wie Augen, Nasen oder Muttermale. Diese werden ebenfalls niedriger gewichtet und damit schlechter rekonstruiert. Das funktioniert aber nicht, wenn der Bart ansatzweise rekonstruiert werden kann, da er dann nicht schlechter rekonstruiert wird als das restliche Gesicht. Da der letzte Algorithmus einen Datensatz aus Bärten benötigt und dieser Algorithmus dafür genutzt werden sollte, diesen Datensatz zu generieren, wurde mit einigen Experimenten sichergestellt, dass er korrekt implementiert wurde. Die Experimente haben gezeigt, dass die Implementierung korrekt ist, woraus man schließen konnte, dass der Unterraum der bartlosen Gesichter Bärte fast genauso gut rekonstruieren kann wie Gesichter. Der dritte und letzte Algorithmus (siehe Kapitel 4) unterteilt das Gesicht in drei Schichten: eine Gesichts-, eine Bart- und eine Sonstigesschicht. Die Gesichts- und die Bartschicht werden wieder erzeugt, indem die Koeffizienten einer Linearkombination gefunden werden. Diese werden wie beim ersten Algorithmus durch das Lösen eines Least-Squares-Problems gefunden. Die Linearkombination findet diesmal aber nicht nur im Unterraum der bartlosen Gesichter, sondern in einer Kombination aus dem Unterraum der bartlosen Gesichter und dem Bartraum statt. Der Bartraum wird erzeugt indem man mit dem zweiten Algorithmus Bärte extrahiert, welche dann wie beim Unterraum der bartlosen Gesichter zu einer Matrix zusammengefasst werden. Die extrahierten Bärte enthalten viele nicht-Bart-Strukturen wie Muttermale oder Narben. Um dieses zu entfernen, wird eine PCA durchgeführt. Die Sonstigesschicht wird erzeugt indem die Bart- und die Gesichtsschicht vom Ausgangsbild subtrahiert werden. Das Ergebnisbild erhält man, indem man die Gesichts- und die Sonstigesschicht addiert. Auch hier wird der Bart nur heller. Der Rest des Gesichts wird aber fast perfekt erhalten. Der Bart wird hier aber auch nur heller, da B viel Rauschen enthält, welches sich negativ auf die Gesichtsrekonstruktion auswirkt. Das Rauschen konnte mit der PCA nicht entfernt werden, da es in fast allen Bartbildern vorhanden ist. Der Grund hierfür ist, dass der zweite Algorithmus bei

der exakten Rekonstruktion und Bartentfernung versagt. Um dem entgegenzuwirken, wurden im darauf folgenden Kapitel drei Modifikationen des zweiten Algorithmus vorgeschlagen, die sowohl die Initialisierung der Gewichtungsmatrix $\boldsymbol{W}$ verändern als auch die Art wie die Einträge von $\boldsymbol{W}$ berechnet werden. Alle Modifikationen konnten die Ergebnisse verbessern. Da dennoch viel Rauschen vorhanden war, beschäftigte sich dieses Kapitel auch damit, wie man dieses verringern kann. Es konnten dennoch keine Ergebnisse erzielt werden die mit den Ergebnissen aus [NLET08] vergleichbar sind. Abbildung 5.11 zeigt einige Ergebnisse.

Ausblick

Wie auch in [NLET08] kann auch hier davon ausgegangen werden, dass sich die Ergebnisse stark verbessern, wenn man die Datensätze vergrößert. Dadurch können Gesichter und Bärte genauer rekonstruiert werden. Das führt dann zu genaueren Schichten, welche dann voneinander getrennt werden können. Ein großer Teil dieser Arbeit beschäftigte sich damit, Rauschen aus den Bildern zu entfernen, aus denen der Bartraum besteht. Im Zusammenhang damit wurde in Abschnitt 5.2.3 erwähnt, dass mit dem Grab Cut das gesamte Rauschen aus einem Differenzbild entfernt werden kann, vorausgesetzt dass ein Nutzer die Segmentierung überwacht und durch Eingaben korrigiert. Auf diese Weise kann ein nahezu perfekter Bartraum erzeugt werden, welcher dann den kompletten Bart rekonstruieren kann. Dadurch sollte eine restlose Entfernung des Bartes vom Gesicht möglich sein. Ein ähnlicher Effekt kann wahrscheinlich auch mit dem Segmentierungsschritt aus [NLET08] erzielt werden. Da die Ergebnisse des letzten Algorithmus insgesamt besser sind als die des zweiten Algorithmus, könnte man einen Bartraum aus den berechneten Bartschichten erzeugen. Dieser neue Bartraum würde weniger Rauschen enthalten, was zu einer besseren Bartrekonstruktion führen würde. Wenn nötig kann man diesen auch mit den vorgestellten Nachbearbeitungsschritten verbessern. Große Teile der Least-Squares-Methode und der Bartentferung durch das Schichtenmodel können vorberechnet werden. Dadurch kann man die Komplexität auf $O(n)$ senken, was eine Echtzeitanwendung möglich machen könnte.

Literaturverzeichnis

[AR15] A. Atkinson, M. Riani. *Vorlesung: Robust Methods and Multivariate Extremes*. 8th International Conference of the ERCIM WG on Computational and Methodological Statistics (CMStatistics 2015). Dez. 2015. URL: http://cmstatistics.org/CMStatistics2015/docs/WinterCourseAR_Regression.pdf?20190704011146 (zitiert auf S. 14).

[Bru18] A. Bruhn. *Vorlesung: Imaging Science*. Institut für Visualisierung und Interaktive Systeme Universität Stuttgart. 2018 (zitiert auf S. 17).

[CK 11] C.-K. Shene. *Vorlesung: Introduction to Computing with Geometry Notes*. Department of Computer Science Michigan Technological University. 2011. URL: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/CURVE-linear-system.html (zitiert auf S. 12).

[Cra] I. Craw. *Aberdeen Face Database*. URL: http://pics.stir.ac.uk/ (zitiert auf S. 20).

[Dav17] M. Davenport. *Vorlesung: Statistical Machine Learning*. Center for Machine Learning at Georgia Tech. 2017. URL: https://mdav.ece.gatech.edu/ece-6254-spring2017/notes/11-regression-regularization.pdf (zitiert auf S. 11).

[KDA+16] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes, A. Gupta, D. Narayanan, C. Sun, G. Chechik, K. Murphy. „OpenImages: A public dataset for large-scale multi-label and multi-class image classification.“ In: *Dataset available from https://github.com/openimages* (2016) (zitiert auf S. 20).

[LLM93] D. C. Lay, S. R. Lay, J. J. McDonald. *Linear Algebra and Its Applications*. Fifth edition. Wellesley-Cambridge Press, 1993 (zitiert auf S. 10, 11).

[Ma 15] Ma, Correll, & Wittenbrink. „The Chicago Face Database: A free stimulus set of faces and norming data“. In: *Behavior Research Methods* 47 (2015), S. 1122–1135 (zitiert auf S. 20, 21).

[NLET08] M. H. Nguyen, J.-F. Lalonde, A. A. Efros, F. de la Torre. „Image-based shaving“. In: *Computer Graphics Forum Journal (Eurographics 2008)* 27.2 (2008), S. 627–635 (zitiert auf S. 7, 16, 20, 23, 27, 35, 45, 51, 52).

[Ope19a] OpencV team. *OpenCV*. 2019. URL: https://opencv.org/ (besucht am 09. 07. 2019) (zitiert auf S. 18).

[Ope19b] OpencV team. *OpenCV 4.1.0 Documentation*. 2019. URL: https://docs.opencv.org/4.1.0/ (zitiert auf S. 18).

[PHJP98] P.J. Phillips, H. Wechsler, J. Huang, P. Rauss. „The FERET database and evaluation procedure for face recognition algorithms“. In: *Image and Vision Computing* 16.5 (1998), S. 295–306 (zitiert auf S. 20).

[PHSP00] P.J. Phillips, H. Moon, S.A. Rizvi, P.J. Rauss. „The FERET evaluation methodology for face recognition algorithms“. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 22.5 (2000), S. 1090–1104 (zitiert auf S. 20).

[Rei02] H.-J. Reinhardt. *Vorlesung: Schlecht gestellte Probleme: Einführung und numerische Methoden*. Fachbereich Mathematik Universität Siegen. 2002. URL: https://www.uni-siegen.de/fb6/numerik/skripts/skripte/scriptv02.pdf (zitiert auf S. 11).

[RKB04] C. Rother, V. Kolmogorov, A. Blake. „GrabCut -interactive foreground extraction using iterated graph cuts“. In: *ACM Transactions on Graphics (SIGGRAPH)* (Aug. 2004). URL: https://www.microsoft.com/en-us/research/publication/grabcut-interactive-foreground-extraction-using-iterated-graph-cuts/ (zitiert auf S. 45).

[RL18] R. Ramos, J. Luis. „Masterarbeit: Deep learning for universal emotion recognition in still images“. Faculty of Mathematics and Computer Science University of Barcelona. 2018. URL: https://pdfs.semanticscholar.org/0a99/87a43d85489afb62dd6db19b38c4c0819411.pdf?_ga=2.116267269.2030592780.1562499918-928058359.1562499918 (zitiert auf S. 19).

[Str09] G. Strang. *Introduction to Linear Algebra*. Fourth edition. Wellesley-Cambridge Press, 2009 (zitiert auf S. 10).

[Tha16] A. Tharwat. „Principal component analysis - a tutorial“. In: *International Journal of Applied Pattern Recognition* 3 (Jan. 2016), S. 197 (zitiert auf S. 13).

[Vog04] W. Vogt. *Vorlesung: Zur Numerik linearer Gleichungssysteme*. Institut für Mathematik Technische Universität Ilmenau. 2004. URL: https://www.tu-ilmenau.de/fileadmin/media/math/Preprints/2004/04_21_vogt.pdf (zitiert auf S. 12).

[Wei04] J. Weickert. *Vorlesung: Mathematik für Informatiker II - Stochastik*. Mathematical Image Analysis Group Universität des Saarlandes. 2004. URL: https://www.mia.uni-saarland.de/Teaching/MFI04/mfi2-stochastik.pdf (zitiert auf S. 15).

Alle URLs wurden zuletzt am 12. 10. 2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift