

Eberhard Karls Universität Tübingen

Mathematisch-Naturwissenschaftliche Fakultät

Wilhelm-Schickard-Institut für Informatik

Dokumentation Praktikum Bildverarbeitung

Automatisches ausschneiden von Bildern aus Fotoalben

Mariella Dreißig, Tom Eckardt, Stefan Bergmann

6. Januar 2019

Betreuer

Andreas Karge, Prof. Andreas Schilling

Mariella Dreißig, Tom Eckardt, Stefan Bergmann:

Automatisches ausschneiden von Bildern aus Fotoalben

Dokumentation Praktikum Bildverarbeitung

Eberhard Karls Universität Tübingen

Zusammenfassung

Das hier entwickelte Kommandozeilenprogramm extrahiert Bilder aus einer eingescannten Fotoalbumseite und kann Gesichter detektieren. Zum extrahieren wird der Floodfill-Algorithmus verwendet, der anhand der Hintergrundfarbe des Fotoalbums die Bilder ausschneidet. Um noch einen vorhandene Rahmen zu entfernen werden die Kanten des Bildes mit dem Canny-Algorithmus detektiert und an den Seiten des Bildes nach einem Bereich gesucht an dem keine Kanten sind. Danach kann mit Hilfe eines Cascade Classifier Gesichter in den Bildern detektiert werden. Dies geschieht sowohl für Gesichter im Profil als auch für Gesichter die direkt in die Kamera schauen. Um die Ergebnisse zu überprüfen werden die Bilder auf Größe, Featureposition und Strukturelle Ähnlichkeit hin überprüft. Das Programm arbeitet sehr gut bei Bildern mit einem deutlichen Rahmen und die gerade in der Seite ausgerichtet sind. Bilder deren Farbwerte stark dem Hintergrund ähneln oder die sich schräg in dem Fotoalbum befinden werden schlecht ausgeschnitten. Für die Gesichtserkennung ist die Qualität und Auflösung der Bilder ausschlaggebend. Bei schlechter Qualität werden wesentlich weniger Gesichter erkannt als bei Bilder mit guter Qualität.

Inhaltsverzeichnis

1	Einleitung	5
1.1	Problemstellung	5
1.2	Benutzte Technologien	5
1.3	Programmaufbau	6
2	Hintergrundentfernung	7
2.1	Floodfill	7
2.2	Ausschneiden	8
2.3	Validierung	8
3	Rahmenentfernung	9
4	Gesichtserkennung	11
4.1	Cascade Classifier	12
5	Auswertung	14
5.1	Metriken zur Auswertung	14
5.1.1	Bildgröße	14
5.1.2	Bildmerkmale	14
5.1.3	Struktureller Ähnlichkeit (SSIM)	15
5.2	Ausgeschnittene Bilder	15
5.3	Gesichtserkennung	16
6	Fazit	22
7	Benutzung und Konfiguration	23
7.1	Ausführen des Programmes	23
7.2	Konfigurationsdatei und Parameter	24
	Literaturverzeichnis	27

1 Einleitung

1.1 Problemstellung

Die hier entwickelte Software soll aus einer eingescannten Fotoalbumseite die darin enthaltenen Fotos extrahieren. Dazu sollte die Seite des Fotoalbums sauber eingescannt sein mit einem möglichst gleichfarbigen Hintergrund und ohne Spiegelungen. Die Fotos können einen Rahmen haben, der von dem Programm entfernt wird. Sobald die Fotos ausgeschnitten wurden kann für jedes einzelne Foto innerhalb der Fotoalbumseite eine Gesichtserkennung durchgeführt werden, bei der die erkannten Gesichter markiert und ausgeschnitten werden. Um das Ergebnis der Software zu überprüfen soll es möglich sein die ausgeschnittenen Fotos mit Ground Truth Bildern auf Ähnlichkeit zu vergleichen. Das Programm soll über die Kommandozeile ausgeführt werden und in einzelne Module aufgeteilt sein, sodass einzelne Elemente jeder Zeit angepasst werden können.

1.2 Benutzte Technologien

Für die Entwicklung dieser Software wurde als Programmiersprache Python 3.6 verwendet zusammen mit der Bibliothek numpy in der Version 1.14.2. Numpy ermöglicht die schnellere Berechnung von Matrizen und wird für die Bibliothek OpenCV in der Version 3.4.0.12 benötigt. OpenCV bietet eine Reihe von Algorithmen für die Bildbearbeitung. Um eine Auswertung anzufertigen wurde aus der Bibliothek SciKit-image 0.13.1 die Methode zur Berechnung der strukturellen Ähnlichkeit (SSIM) verwendet.

1.3 Programmaufbau

Das Programm teilt sich in fünf Hauptkomponenten auf. Die erste ist in der *main.py* zu finden, die das Programm startet und die anderen Komponenten ausführt. Die zweite Komponente kümmert sich um das entfernen des allgemeinen Hintergrunds und ist in der *backgroundremover.py* zu finden. In dieser Komponente werden die eigentlichen Fotos aus dem Fotoalbum grob ausgeschnitten. Danach wird in der nächsten Komponente für jedes Foto der Rahmen entfernt, falls ein Rahmen vorhanden ist. Das entfernen des Rahmens geschieht in der *rectextract.py*. Die letzte Komponente ermöglicht es Gesichter in den ausgeschnittenen Bildern zu erkennen. Dies geschieht in der *facetedetection.py*. Möchte man die ausgeschnittenen Bilder mit Beispielbildern vergleichen, so kann man die letzte Komponente in der Datei *compare.py* verwenden. Mit deren Hilfe Metriken aufgestellt werden zum Vergleich der Bilder. In dem Flussdiagramm 1.1 ist der Ablauf des Programmes nochmals genauer als Diagramm dargestellt.

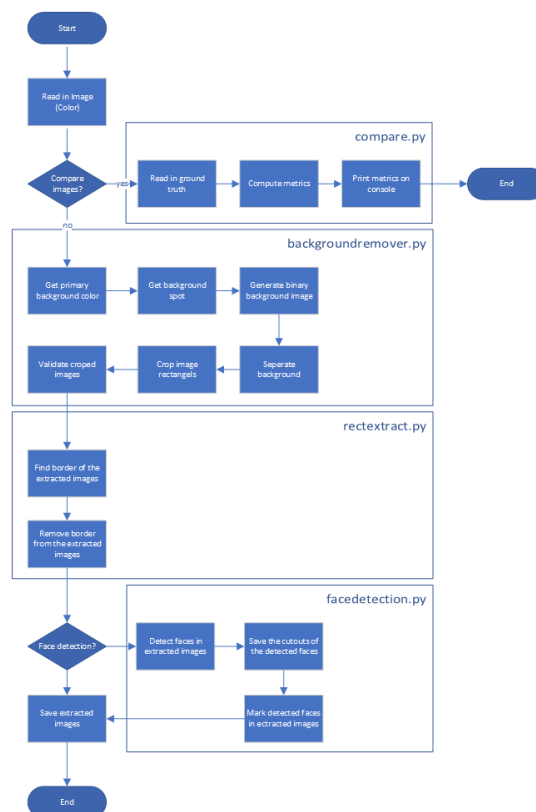


Abbildung 1.1: Flussdiagramm, das den Ablauf des Programmes darstellt. Verzweigungen können mit Hilfe von Parametern bei der Ausführung des Programmes gesteuert werden.

2 Hintergrundentfernung

2.1 Floodfill

Um den Hintergrund zu detektieren wird der Floodfill-Algorithmus verwendet. Dieser färbt eine Fläche zusammenhängender Pixel neu ein. Die Pixel sind zusammenhängend, wenn sich der Farbwert der Pixel nicht unterscheidet. Der Algorithmus untersucht dabei rekursiv, von einem Startpunkt beginnend, die benachbarten Pixel und färbt diese neu ein, wenn der Farbwert, der untersuchten Pixel, dem Farbwert des Startpixel entsprechen([5]).

Der Floodfill-Algorithmus ermöglicht es eine Maske zu erstellen mit deren Hilfe der Hintergrund des Fotoalbums von den Bildern unterschieden werden kann. Um den Hintergrund zu erkennen muss dessen Farbe erkannt werden. In einem Fotoalbum ist davon auszugehen, dass die Hintergrundfarbe am häufigsten innerhalb des Bildes vorkommt. Deshalb verwenden wir die Farbe die im Histogramm des Bildes am meisten vorkommt.

Nachdem wir die Farbe bestimmt haben müssen wir einen Startpunkt suchen von dem aus der Floodfill-Algorithmus ausgeführt wird. Für den Startpunkt ist es nicht möglich ein willkürliches Pixel mit der Hintergrundfarbe zu wählen, da diese Farbe auch innerhalb der Bilder, die ausgeschnitten werden sollen, vorkommen kann.

Deshalb suchen wir nach einem Bildausschnitt der fast ausschließlich mit der Hintergrundfarbe gefüllt ist. Dazu muss beachtet werden, dass der Hintergrund nicht perfekt die gleiche Farbe hat. Aus diesem Grund verwendet das Programm die quadratische Differenz zwischen dem gesuchten Bildausschnitt und einem Ausschnitt gefüllt mit der Hintergrundfarbe. Der Bildausschnitt mit der geringsten Differenz wird dann als Startpunkt ausgewählt.

Um die Effizienz des Floodfill-Algorithmus zu steigern wird ein Binärbild erzeugt indem die Hintergrundfarbe weiß dargestellt wird und der gesamte Rest schwarz. Dabei wird ein Threshold angewandt um ähnliche Farben mit einem niedrigeren oder höheren Farbwert ebenfalls weiß darzustellen.

Auf diese Binärbild wird dann der Floodfill-Algorithmus angewandt um die zusammenhängen-

de Fläche des Hintergrunds zu separieren, sodass die auszuschneidenden Bilder, welche die gleiche Farbe beinhalten wie der Hintergrund, nicht betroffen sind. Daraus resultiert ebenfalls ein Binärbild bei dem der Hintergrund schwarz ist und die auszuschneidenden Bilder weiß sind.

2.2 Ausschneiden

Zur Detektion von Konturen wird das Verfahren von Satorshi Suzuki und Keiichi Abe von OpenCV verwendet, dass das Bild mit einem Rasterscan durchsucht um Kantenstartpunkte zu finden. Den Kanten wird dann gefolgt und sie werden markiert. Der Algorithmus markiert die Kanten topologisch indem er sich die vorher gefundenen Kanten merkt und so einen Baum aufspannen kann mit dem man umschlossene Kanten feststellen kann. Das Ergebnis ist eine Reihe von Konturpunkten für die Ecken der jeweiligen Kontur([7]).

Die so gefundenen Konturpunkte müssen dann überprüft werden, ob sie die kleinstmögliche Ecke für das Bild darstellen. Wenn dies geschehen ist wird das Bild an dieser Stelle mit einem kleinen Rand ausgeschnitten um Fehler in der Auswahl zu umgehen.

2.3 Validierung

Schon während des Ausschneidens der gefundenen Bilder findet eine Validierung statt. Bei dieser Validierung werden Bilder aussortiert die entweder zu groß oder zu klein sind oder ein unnatürliches Breiten zu Höhen Verhältnis haben. Nach dem Ausschneiden wird noch einmal separat überprüft, ob es sich um Fotos handelt indem Kanten und Ecken detektiert werden. Dies geschieht indem man zweimal einen Gaussfilter mit unterschiedlicher Größe auf das Bild anwendet und dann die beiden Resultate voneinander abzieht. Daraus entsteht eine Approximation eines Gradientenbildes, mit dessen Hilfe man bestimmen kann, ob das Bild genug Merkmale aufweist. Ein normales Foto weist wesentlich mehr Merkmale auf als ein Ausschnitt indem sich kein oder nur teilweise ein Foto befindet.

3 Rahmenentfernung

$$G = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}, S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix},$$

Abbildung 3.1: Die Figur zeigt den Gausfilterkernel sowie die Faltungsmasken des Sobeloperators, die beim Canny-Algorithmus verwendet werden

Die Extraktion eines Bildes aus einem monotonen Rahmen geschieht in mehreren Schritten. Zu erst werden die Kanten des Bildes detektiert um mit dem daraus resultierenden Binärbild festzustellen, ob sich ein monotoner Rahmen um das eigentliche Bild befindet.

Zur Detektion von Kanten gibt es mehrere Möglichkeiten, da es aber wichtig ist, dass die Kanten möglichst Pixel genau detektiert werden, ist der Canny-Kantendetektionsalgorithmus am besten dazu geeignet. Bei diesem Verfahren wird zu erst rauschen aus dem Bild durch einen Gaußfilter entfernt. Der Kernel für einen 5x5 Gausfilter ist in Abbildung 3.1 dargestellt. Mit Hilfe der Sobeloperatoren für die X-, Y-Richtung werden danach die Gradienten des Bildes errechnet. Die dazu benötigten Faltungskerne sind ebenfalls in Abbildung 3.1 dargestellt. Mit den so gewonnen Ableitungen kann man dann die Richtung des Gradienten berechnen und die absolute Kantenstärke. Damit nun jede Kante nicht mehr als ein Pixel breit ist, soll mit Hilfe einer Non-maximum-suppression nur die Maxima entlang einer Kante erhalten bleiben. Bei der Non-maximum-suppression wird der Wert eines Pixels der Kante mit den Pixeln rechts und links neben der Kante verglichen. Sollte einer der Werte größer sein wird der betrachtete Pixel auf null gesetzt. Danach wird noch festgestellt, ab welcher Kantenstärke ein Pixel überhaupt zu einer Kante gehört. Das Ergebniss des Kantendetektionsalgorithmus ist dann ein Binärbild indem jede gefundene Kante mit weißen Pixeln dargestellt ist ([3]).

Um nun im nächsten Schritt den monotonen Rahmen zu erkennen muss man beachten, dass sich innerhalb des Rahmens keine Kanten befinden, nur zu beginn des Rahmens und direkt danach. Um diese Fläche ohne Kanten zu detektieren wird zuerst ein Bildausschnitt entlang

3 Rahmenentfernung

einer der vier Seiten erzeugt. Der Bildausschnitt erstreckt sich über die komplette Seite des Bildes und reicht eine bestimmte Anzahl an Pixeln in das Bild. Die Tiefe mit der der Ausschnitt in das Bild reicht wird dann Schritt für Schritt vergrößert. In jedem Schritt wird die Anzahl der weißen Pixel innerhalb des Ausschnittes gezählt. In Abbildung 3.2 kann man die Zunahme der weißen Pixel in den einzelnen Bildausschnitten ablesen.

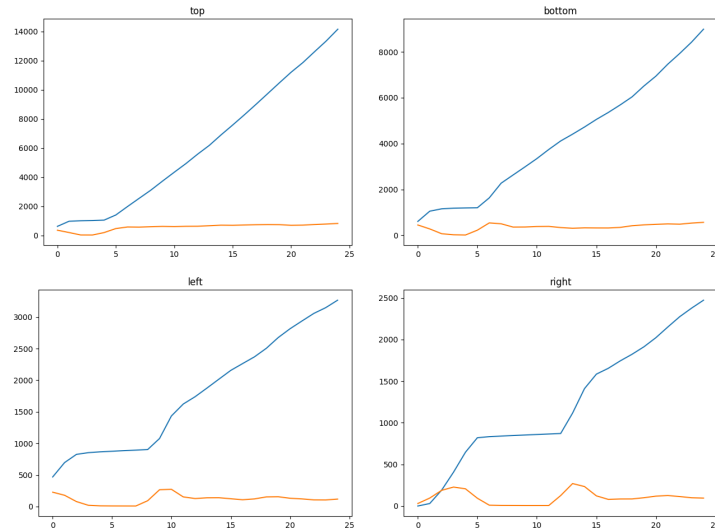


Abbildung 3.2: Die obere Linie zeigt die Zunahme weißer Pixel in den schrittweise größer werdenden Bildausschnitten. Die untere Linie zeigt die Gradient der Zunahme zum vorhergehenden Schritt.

Da sich innerhalb des Rahmen kaum weiße Pixel befinden kann man in der Abbildung deutlich sehen das in einem bestimmten Bereich die Anzahl der weißen Pixel fast gleich bleibt. In diesem Bereich muss sich der Rahmen befinden. Direkt danach nimmt die Anzahl deutlich zu. Dies muss der Beginn des eigentlichen Bildes sein. Um nun erkennen zu können, wann das eigentliche Bild beginnt wird der Gradient, der Zunahme von weißen Pixeln, verwendet. Dieser ist ebenfalls in der Abbildung 3.2 dargestellt. Der Gradient ist am geringsten in den Bereichen des Rahmens und hat dann ein kleines Maximum direkt nach dem Rahmen.

Im nächsten Schritt kann man dann mit Hilfe des davor erzeugten Gradienten das Bild soweit abschneiden, dass kein Rahmen mehr enthalten ist. Dazu sucht man den Schritt an dem der Gradient am kleinsten ist und geht die folgenden Schritte weiter bis zu dem nächsten Maximum. Anhand der Anzahl von Schritten die vom Rand des Bildes bis zum Maximum gegangen wurden kann man die Anzahl an Pixeln bestimmen, die an dieser Seite des Bildes abgeschnitten werden müssen. Sollte kein Maximum gefunden werden oder sollte das Maximum zu weit entfernt sein vom Minimum der Gradienten, dann wird kein Schnitt ausgeführt.

4 Gesichtserkennung

Die Aufgabenstellung beinhaltet, in den extrahierten Fotos Gesichter zu erkennen. Hier muss die Gesichtserkennung (in der Fachliteratur "Face Detection") von der Wiedererkennung von bereits identifizierten Gesichtern (in der Fachliteratur "Face Recognition") unterschieden werden. In der vorliegenden Arbeit geht es ausschließlich darum, die Position von Gesichtern in den eingescannten Fotos zu erkennen und nicht, bereits erkannte Gesichter wiederzuerkennen oder zuzuordnen ¹.

In der klassischen Bildverarbeitung wurden lange einfache 2D-Verfahren verwendet, um Gesichter in Bildern zu erkennen ([1], [10]). Beim Template Matching werden beispielsweise Vorlagen (von Teilen) von Gesichtern erzeugt um diese dann im Anwendungsfall mit den Zielbildern zu vergleichen. Dieses Verfahren ist nicht sehr robust, besonders bei sich verändernden Lichtverhältnissen, aber gleichzeitig extrem rechenaufwändig.

Eine einfachere Methode der Gesichtserkennung in der klassischen Bildverarbeitung ist die Nutzung von geometrischen Merkmalen in den Gesichtern. Hierbei werden beispielsweise Grauwerte in einem bestimmten Bereich des Bildes aufsummiert oder Gradientenoperatoren zur horizontalen und vertikalen Kantenerkennung angewandt. So kann die Ausrichtung, die relative Lage und ähnliche Merkmale der Nase, der Augen und des Mundes identifiziert und zusammen als Gesicht klassifiziert werden.

Doch auch diese Methode ist rechen- und zeitintensiv. Deswegen wird häufig eher auf komplexere Methoden zur Gesichtserkennung gesetzt, wie beispielsweise das Elastic Bunch Graph Matching, das eine robuste Waveletanalyse nutzt ([11]) oder die auf der Hauptkomponentenanalyse basierende Eigenfaces-Methode ([8]). Weiterentwicklungen dieser beiden Methoden werden heute noch eingesetzt, obwohl die Ansätze schon vergleichsweise alt sind.

Im Folgenden wird ein populärer Ansatz vorgestellt, der in den vorliegenden Arbeit verwendet wurde.

¹ Aufgrund der geringen Datenmenge und der schlechten Qualität der alten Fotos ist es nicht möglich, Gesichter wiederzuerkennen.

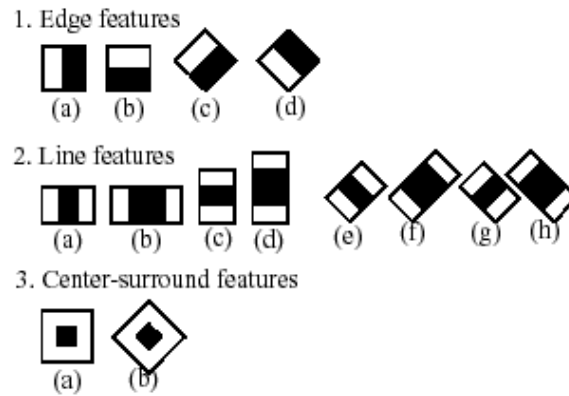


Abbildung 4.1: Alle vom Cascade Classifier Algorithmus genutzten Haar-ähnlichen Features zur Objekterkennung. Zur Gesichtserkennung werden nur fünf Features genutzt: 1(a), 1(b), 2(a), 2(c) und eine Kombination aus zwei 1(a) Features, die zusammen ein schachbrettartiges Muster ergeben.

4.1 Cascade Classifier

Die OpenCV Programmbibliothek beinhaltet vorimplementierte Funktionen zur Gesichtserkennung. Sie basiert auf einem 2001 entwickelten Ansatz zur Objekterkennung in Bildern ([9], [2]). Es ist ein trainierter Classifier, der einfache geometrische Merkmale im Bild erkennt und auf deren Basis die gewünschten Objekte erkennen kann ([4]).

Die Methode nutzt sogenannte Haar-ähnliche Features, mit denen grundlegende Strukturen (Helligkeitsunterschiede) im Bild erkannt werden. Im Bild 4.1 werden alle Features aufgelistet, die der Classifier nutzen kann. Für die Gesichtserkennung werden in der Praxis aber nicht so viele genutzt, sondern nur einige wenige. Mit diesen können allgemeine Eigenschaften von menschlichen Gesichtern repräsentiert werden. So ist zum Beispiel die Nase heller als die Bereiche um die Augen, das entspricht dem Feature 2(a) in Bild 4.1.

Nachdem die zu nutzenden Features ausgewählt wurden, wird ein Integralbild erzeugt, das in der Lage ist, die Features in konstanter Zeit auszuwerten. Dadurch wird eine relativ geringe Laufzeit erzielt.

In der Trainingsphase wird der Classifier mit Positiv- und Negativbeispielen von dem zu erkennenden Objekt trainiert, in diesem Fall mit Gesichtern². Dieser Vorgang wird mit verschiedenen Arten des AdaBoostings beschleunigt und verbessert. Dadurch werden die besten Features für die Klassifizierung identifiziert.

²In der OpenCV Bibliothek ist sowohl ein Trainer als auch ein Detector enthalten. Der Detector ist vortrainiert kann direkt zur Gesichtserkennung verwendet werden. Wenn man genug Daten zur Verfügung hat, kann man aber auch einen selbst trainieren.

4 Gesichtserkennung

Danach wird der Classifier in mehreren Stufen auf die Bilder angewandt ("cascade"), in denen das Objekt identifiziert werden soll. Jeder Classifier funktioniert dabei wie ein Filter, der die negativen Regionen aus dem Bild herausfiltert. Die verbleibenden Bereiche werden Stück für Stück durch immer komplexere Classifier gegeben, bis sie entweder aussortiert werden oder das Objekt (ein Gesicht) identifiziert wurde. Somit hat der Cascade Classifier die Struktur eines Entscheidungsbaumes und Rechenzeit wird gespart. Die Methode ist so robust, dass das Objekt in verschiedenen Größen in dem Bild erkennen kann, unabhängig von der Größe der Bilder, mit der der Classifier trainiert wurde.

Wenn die OpenCV Methode `detectMultiScale` ein Gesicht im Input-Bild erkennt, gibt es die Position in x- und y-Koordinaten und dessen Abmessungen (Höhe, Breite) zurück. In dem vorliegenden Programm werden zwei verschiedene Classifier auf die Bilder angewandt: einmal frontale Gesichter und einmal Gesichter im Profil. Die Rückgabewerte der Funktionen werden dafür genutzt, die Gesichter auszuschneiden und Dopplungen zu erkennen. Da es passieren kann, dass beide Classifier die gleichen Gesichter detektieren, wird durch eine Hilfsfunktion festgestellt, ob dieses Gesicht bereits in der Liste der erkannten Gesichter vorhanden ist. Dafür wird überprüft, ob der Mittelpunkt des kleineren Ausschnitts in dem größeren liegt. Wenn das der Fall ist, kann angenommen werden, dass es sich um das gleiche Gesicht handelt und eines davon wird verworfen. So wird sichergestellt, dass keine Gesichter verworfen werden, die lediglich nah beieinander liegen, so wie beispielsweise in einem Gruppenbild.

5 Auswertung

5.1 Metriken zur Auswertung

5.1.1 Bildgröße

Da es in dem Projekt um das Ausschneiden von Bildern geht ist die Bildgröße eine nahe liegende Metrik um die Genauigkeit unserer Software zu messen. Die Größe wird berechnet indem die Höhe mit der Breite multipliziert wird. Von der so ermittelten Größe des ausgeschnittenen Bildes wird die Größe des Ground Truth Bildes abgezogen. Aus dieser Differenz wird der Prozentuale unterschied zum Ground Truth Bild berechnet. Sollte das Ergebnis negativ sein ist das ausgeschnittene Bild kleiner als der Ground Truth.

5.1.2 Bildmerkmale

Eine weitere Metrik ist das erkennen gleicher Bildmerkmale. Dazu wird der patent freie ORB (Orientated FAST and Rotated BRIEF) detektor von OpenCV verwendet der Rotations invariant ist. ORB ist eine fusion zwischen dem FAST Schlüsselpunkt Detektor and BRIEF Beschreibung der Schlüsselpunkte mit vielen Modifikationen um die Geschwindigkeit des Algorithmus zu verbessern ([6]). Die so erkannten Merkmale werden dann der reihe nach anhand der Distanz zum Bildzentrum verglichen. Es wird für jedes erkannte Bildmerkmal die Distanz zum Zentrum des dazu gehörigen Bildes berechnet und das dann mit den Ergebnissen des Ground Truth verglichen. Solange zwischen den beiden Merkmalen keine zu großer Unterschied in der Distanz ist werden sie als gleich angesehen. Durch dieses Verfahren kann erkannt werden das sich der Ausschnitt des ausgeschnittenen Bildes an der selben stelle befindet, wie der Ground Truth und nicht nur die gleiche Größe ausgeschnitten wurde.

5.1.3 Struktureller Ähnlichkeit (SSIM)

Um ein Bild auf Gleichheit zu untersuchen verwenden wir den Strukturellen Ähnlichkeitsindex (Structural Similarity index, SSIM). Dieser berechnet die Ähnlichkeit zwischen verschiedenen Bildausschnitten mit folgender Gleichung:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

mit μ als Durchschnitt und σ als Varianz bzw. Kovarianzmatrix des Bildausschnittes. c sind da um die Division zu stabilisieren ([12]). Das Verfahren lässt sich nur auf gleich große Bilder anwenden. Aus diesem Grund verkleinern wir die Bilder mit einer linearen Interpolation auf eine Größe von 64 x 64 Pixeln. Auf die so verkleinerten Bilder wird dann der SSIM angewandt. Um so näher das Ergebnis an 1 ist um so ähnlicher sind die beiden Bilder.

5.2 Ausgeschnittene Bilder

Bild	Größe	Merkmale	SSIM
01	1.92	3.64	0.82
02	1.60	2.39	0.77
03	-5.52	10.12	0.36
04	3.15	16.54	0.68
05	1.48	2.65	0.81
06	-4.42	23.58	0.55
07	2.96	4.40	0.80
08	8.82	8.07	0.39
09	-1.24	2.06	0.70
10	-0.58	5.57	0.76
11	-4.35	28.46	0.49
12	2.96	1.74	0.80
13	1.26	3.24	0.88
14	3.01	1.94	0.72
15	0.76	19.51	0.60

Tabelle 5.1: Ergebnisse der Vergleichsmetriken der ausgeschnittenen Bilder.

In der Tabelle 5.1 sind die Ergebnisse für einige der Testbilder im Vergleich zu Ground Truth Bildern dargestellt. Die Ground Truth Bilder wurden per Hand ausgeschnitten ohne die Bilder zu rotieren oder andere Veränderungen vorzunehmen. Den direkten Vergleich zwischen

5 Auswertung

ausgeschnittenem Bild und Ground Truth kann man in Abbildung 5.1 sehen dort ist das Ergebnis von Bild 1 aus der Tabelle dargestellt. In Abbildung 5.2 kann man das Ergebnis von Bild 11 sehen, bei dem die Erkennung des Rahmens schwer gefallen ist durch die gegebene Rotation des Bildes.



Abbildung 5.1: Links ist das Ergebnis Bild im Vergleich zum Ground Truth Bild rechts davon. Das verglichene Bild ist Bild 01 aus der Tabelle 5.1

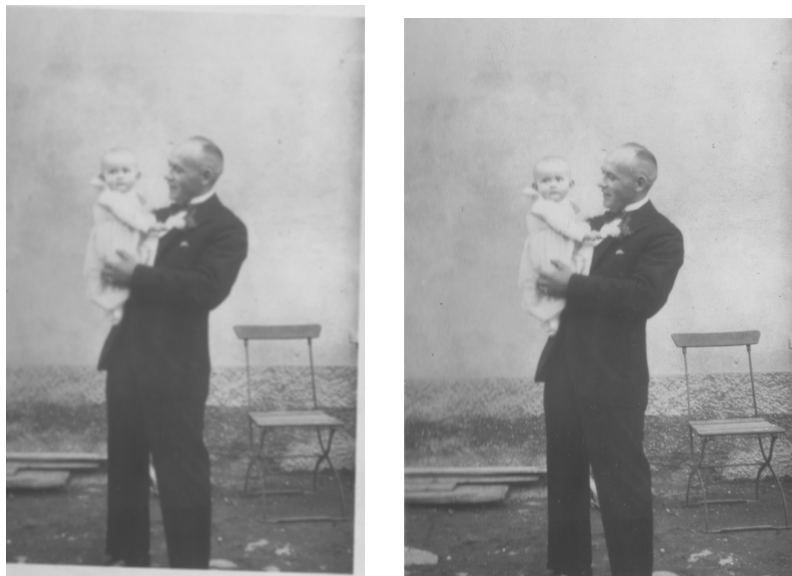


Abbildung 5.2: Links ist das Ergebnis Bild im Vergleich zum Ground Truth Bild rechts davon. Das verglichene Bild ist Bild 11 aus der Tabelle 5.1

5.3 Gesichtserkennung

Die von OpenCV mitgelieferten Methoden zur Gesichtserkennung auf Bildern funktioniert grundsätzlich gut. Sie ist einfach anwendbar, hat eine vertretbare Laufzeit (der Algorithmus

5 Auswertung

läuft nur einige Sekunden auf einer ganzen eingescannten Albumseite) und erkennt zumindest einen Großteil der Gesichter in den Fotos.

Um die Performanz der Classifier einzeln und in Kombination auszuwerten, wurden Wahrheitsmatrizen aufgestellt. Ausgewertet werden jeweils die korrekt erkannten Gesichter (True Positives (TP): positive - positive), die nicht erkannten Gesichter (False Negatives (FN): positive - negative) und die fälschlicherweise erkannten Gesichter (False Positives (FP): negative - positive). Korrekt nicht erkannte Gesichter (True Negatives (TN): negative - negative) gibt es in unserem Anwendungsfall nicht.

Tabellen 5.2 und 5.3 zeigen die Metriken in dem ersten und dem zweiten Datensatz wenn nur der frontal faces Classifier angewendet wird. Die daraus errechnete Genauigkeit ¹ beträgt im ersten Datensatz 0.75 und im zweiten 0.59.

Wenn der zweite Classifier zur Erkennung von Profilen mit dazu genommen wird, erhöht sich

	Reference	
	Positive	Negative
	Prediction	
	Positive	270
	Negative	88

Tabelle 5.2: Wahrheitsmatrix im ersten Datensatz, wenn nur der frontal-Classifier zum Einsatz kommt. Die Genauigkeit der Gesichtserkennung beträgt 0.75.

	Reference	
	Positive	Negative
	Prediction	
	Positive	282
	Negative	172

Tabelle 5.3: Wahrheitsmatrix im zweiten Datensatz, wenn nur der frontal-Classifier zum Einsatz kommt. Die Genauigkeit der Gesichtserkennung beträgt 0.59.

die Anzahl der erkannten Gesichtern ein wenig. Das ist in Tabellen 5.4 und 5.5 zu sehen. Die Gesichter, die bereits von dem frontal-Classifier erkannt wurden, werden automatisch heraus gefiltert. Danach bleibt ein kleiner Mehrgewinn durch die Anwendung des profile-Classifiers übrig, die Performanz verbessert sich auf 0.76 und 0.65 jeweils.

Festlegung der Ground Truth

Die Referenz für die Auswertung der Gesichtserkennung festzulegen ist nicht trivial. Auf den alten Fotos sind aufgrund der schlechten Qualität der Aufnahmen viele Gesichter nicht tatsäch-

¹Formel für die Berechnung der Genauigkeit ("Accuracy") ist: $\frac{TP+TN}{TP+FN+TN+FP}$, ($TN = 0$ in unserem Fall).

5 Auswertung

Reference			
Prediction	Positive	274	2
	Negative	84	-

Tabelle 5.4: Wahrheitsmatrix im ersten Datensatz, wenn frontal- und profile-Classifer kombiniert werden. Die Genauigkeit der Gesichtserkennung beträgt 0.76.

Reference			
Prediction	Positive	301	8
	Negative	153	-

Tabelle 5.5: Wahrheitsmatrix im zweiten Datensatz, wenn frontal- und profile-Classifer kombiniert werden. Die Genauigkeit der Gesichtserkennung beträgt 0.65.

lich zu erkennen. Vielmehr erschließt sich ein menschlicher Betrachter aus dem Kontext die Information, dass sich an gewissen Stellen ein Gesicht befinden muss. Das wurde bei der Festlegung der Ground Truth berücksichtigt. So wurden beispielsweise besonders kleine, verdeckte oder stark verschwommene Gesichter für die Berechnung der Metriken nicht berücksichtigt. Die Classifier haben außerdem Probleme, Gesichter zu erkennen, wenn starke Okklusionen auftreten, selbst wenn das Gesicht nur im Profil zu erkennen ist oder die Person eine Kopfbedeckung trägt. Diese wiederum wurden in die Gesamtsumme der Gesichter mit einberechnet. Im Folgenden werden exemplarisch einige Bilder gezeigt, die der Gesichtserkennungsalgorithmus nicht erkennen kann und die deswegen nicht in die Auswertung mit einbezogen wurden.

Bild 5.3 zeigt ein kleines Mädchen, jedoch ist die Aufnahme sehr verschwommen. Jeder Mensch, der das Bild betrachtet, weiß, dass das Mädchen ein Gesicht haben muss. Wenn jedoch die Kontextinformationen entfernt werden würde, wäre es selbst für einen Menschen schwer, ein Gesicht zu identifizieren.

Auf Bild 5.4 ist eine ähnliche Situation zu sehen: alle abgebildeten Menschen haben Gesichter, jedoch sind sie objektiv nicht wirklich zu sehen. Sie sind relativ klein, was normalerweise für den Classifier kein Problem darstellt, aber eine schlechte Beleuchtung und geringer Kontrast tragen dazu bei, dass die Gesichter nicht erkannt werden.

Um dennoch eine repräsentative Auswertung der Performanz der Gesichtserkennung zu gewährleisten, wurden viele nicht erkannte Gesichter mit in die Metriken einberechnet. Beispielsweise können die Classifier keine Gesichter mit Kopfbedeckungen erkennen. Ansonsten sind aber die Gesichter gut zu identifizieren, was in Bild 5.5 zu sehen ist.

Bild 5.6 zeigt einen problematischen Grenzfall aus dem zweiten Datensatz. In diesem Bild wurden insgesamt drei Gesichter erkannt (zwei vom frontal-Classifer (grün) und eins vom



Abbildung 5.3: Starke Bewegung im Bild verhindert, dass das Gesicht des Mädchens tatsächlich zu sehen ist, auch wenn menschliche Betrachter durchaus ihr Gesicht erkennen könnten.

profile-Classifer (rot)). Da alle Gesichter in etwa die gleiche Größe und Schärfe haben, sollte angenommen werden, dass auch die restlichen Personen erkannt werden. Es ist nicht ersichtlich, warum auf diesem Bild beide Classifier keine weiteren Gesichter erkennen. Von daher fließen alle Gesichter auf diesem Foto mit in die Bewertung ein und erhöhen die Anzahl der False Negatives enorm. Besonders im zweiten Datensatz kommen solche Fälle häufiger vor, was die schlechte Performanz des Algorithmus' im Vergleich zum ersten Datensatz erklärt.



Abbildung 5.4: Die Gesichtserkennung schlägt fehl, wenn die Gesichter zu schlecht beleuchtet sind und/oder zu wenig Kontrast aufweisen. Auch, wenn normalerweise auch Gesichter in dieser Größe von dem Algorithmus problemlos erkannt werden.



Abbildung 5.5: Kein Classifier erkennt Gesichter mit Hüten, auch wenn sie nur einen kleinen Teil vom Gesicht verdecken. Alle anderen Gesichter werden problemlos identifiziert.



Abbildung 5.6: In diesem Gruppenfoto werden nur wenige Gesichter erkannt, obwohl sie alle ähnlich aussehen. Das verschlechtert die Kennwerte für die Genauigkeit besonders im zweiten Datensatz stark.

6 Fazit

Das vorliegende Programm extrahiert einzelne Fotos aus eingescannten Albenseiten mithilfe von Methoden aus der klassischen Bildverarbeitung. Das passiert in zwei Schritten. Im ersten Schritt werden die Fotos mithilfe der Hintergrundfarbe und einem Binärbild grob ausgeschnitten. Diese grob ausgeschnittenen Bilder haben allerdings oft noch einen Rahmen, der dann im zweiten Schritt entfernt wird.

Das Programm wurde auf zwei Fotoalben angewandt mit jeweils 32 und 49 Seiten. Insgesamt wurden 126 von 136 Bildern (92.65 %) im ersten und 192 von 211 Bildern (91.00 %) im zweiten Album erfolgreich extrahiert. Die Performanz des Programms kann anhand von verschiedenen Parametern ausgewertet werden.

Die Maße des Bildes (Bildgröße), die Bildmerkmale (ORB) und die strukturelle Ähnlichkeit (SSIM) verglichen mit der Ground Truth zeigen, wie erfolgreich der Code ist. Besonders der SSIM ist ein gutes Indiz für den Erfolg des Algorithmus, und in den meisten Bildern werden gute Werte erzielt. Insgesamt lässt sich sagen, dass in dem Datensatz die meisten Fotos erkannt und gut ausgeschnitten werden, bis auf einige Ausnahmen. In diesen Fällen sind oft schlechte Lichtverhältnisse oder ungewöhnliche Formen der Fotos der Grund dafür.

Optional kann das Programm menschliche Gesichter erkennen und extrahieren. Dafür wird ein einfacher machine learning Ansatz verwendet. Dieser erkennt erfolgreich die meisten Gesichter, trotz des Alters der Fotos und der mangelnden Qualität. Dennoch werden einige Gesichter nicht erkannt, meist, wenn sie zu klein und verschwommen sind oder die abgebildete Person eine Kopfbedeckung trägt.

In kommenden Arbeiten könnte das Programm verbessert werden, indem Sonderfälle wie ovale oder schlecht belichtete Fotos anders behandelt werden. Der aktuelle Stand kann gut für die Extraktion von Fotos aus eingescannten Albumseiten verwendet werden, ohne dass viel manuelle Nachbearbeitung nötig ist. Das Programm wurde vollständig in Python unter der Verwendung von OpenCV geschrieben.

7 Benutzung und Konfiguration

7.1 Ausführen des Programmes

Für die Installation ist Python 3.6 notwendig mit dem Paketmanager Pip. Um die Abhängigkeiten zu installieren muss man mit der Kommandozeile in das Hauptverzeichnis navigieren indem sich die Datei *requirements.txt* befindet. Dort führt man folgenden Befehl aus:

```
pip3 install -r requirements.txt
```

Pip installiert nun sämtliche Abhängigkeiten, die zur Ausführung des Programmes benötigt werden. Um das Programm dann auszuführen genügt folgender Befehl:

```
python3 main.py /path/to/photo/album/page.tif
```

Man kann dann mit Parametern steuern, wo die ausgeschnittenen Fotos gespeichert werden sollen, wo sich die Konfigurationsdatei befindet, ob man Gesichter in den Fotos detektiert haben möchte oder ob man ein Bild mit einem Groud Truth Bild vergleichen möchte. Die Anleitung zum Ausführen dieser Befehle erhält man mit folgendem Befehl:

```
python3 main.py -h
usage: main.py [-h] [--result RESULT] [--compare COMPARE]
               [--config CONFIG] [--face] image

positional arguments:
  image                path to the image of the photo album page

optional arguments:
  -h, --help            show this help message and exit
  --result RESULT       path to the directory to store the
                        results
  --compare COMPARE     path to the image to compare
                        the given image to
  --config CONFIG       path to the config file
  --face               enables face detection
```

7.2 Konfigurationsdatei und Parameter

In der Konfigurationsdatei *config* befinden sich Parameter mit denen man das Programm steuern kann. Sämtliche Parameter die man setzen kann sind im folgenden aufgeführt mit einer Beschreibung was mit diesem Parameter gesteuert wird.

```
#-----
# Parameter for removing the background
#-----
[BackgroundRemover]
# The size of the searched spot in the background of the images
# to start the flood fill algorithm. The higher the value,
# the slower the search but up to a certain size it is more stable.
SpotSize = 200

# The threshold around the primary background color,
# which still should belong to the background to create a
```


7 Benutzung und Konfiguration

```
# binary image which is separating the background
# colors from the rest.
BinaryThreshold = 25

# The size an image must at least have to be considered as an
# image.
MinImageSize = -100

# The maximum relation between the width and the height of an
# image to prevent the finding of strips.
MaxRelationImageDimensions = 2.5

# The padding with which image is cut out of the original photo.
ImagePadding = 10

# The necessary amount of features needed to be regarded as image.
# To check if the found image is legit.
FeatureThreshold = 10
```

```
#_____
# Parameter for extracting the image
#_____
[ImageExtraction]
# The maximum size of the window within the program searches
# a frame around the image.
MaxWindowSize = 0.1

# The number of steps to search for the frame.
# The maximum window size and the number of steps determine
# the size of the steps fitting to the image size.
Steps = 25

# The ignored steps. This is done to prevent the program
# of finding the already given image boarder.
Offset = 4
```

```
#-----  
# Parameter for the face detection  
#-----  
[FaceDetection]  
# path for the Haarcascade files for frontal faces  
CascadePathFrontal = ./imextract/cascade/haarcascade_frontalface.xml  
  
# path for the Haarcascade files for profile faces  
CascadePathProfile = ./imextract/cascade/haarcascade_profileface.xml  
  
# specifies how much the image size is reduced at each image scale  
ScaleFactor = 1.2  
  
# specifies how many neighbors each candidate rectangle  
# should have to retain it  
Neighbors = 5
```

Literaturverzeichnis

- [1] Kees, Benjamin: *Gesichtserkennung*, 2012. <https://www.algoropticon.de/Dateien/Seminararbeit{ }Gesichtserkennung.pdf>.
- [2] Lienhart, Rainer, Alexander Kuranov, Vadim Pisarevsky und M R L Technical Report: *Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection*. Technischer Bericht, 2002. <http://www.multimedia-computing.de/mediawiki//images/5/52/MRL-TR-May02-revised-Dec02.pdf>.
- [3] OpenCV: *Canny Edge detector*. https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.htmls, besucht: 2018-12-19.
- [4] OpenCV: *Haar Feature-based Cascade Classifier for Object Detection*. <https://docs.opencv.org/2.4/modules/objdetect/doc/cascade{ }classification.html{#}grouprectangles>, besucht: 2018-12-09.
- [5] OpenCV: *Miscellaneous Image Transformations*. https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html, besucht: 2018-12-19.
- [6] OpenCV: *ORB: An efficient alternative to SIFT or SURF*. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html, besucht: 2018-12-19.
- [7] Suzuki, Satoshi und Keiichi Abe: *Topological structural analysis of digitized binary images by border following*. Computer Vision, Graphics, and Image Processing, 30:32–46, 1985.
- [8] Turk, Matthew und Alex Pentland: *Eigenfaces for Recognition*. Journal of Cognitive Neuroscience, 3(1):71–86, 1991. <https://doi.org/10.1162/jocn.1991.3.1.71>.

- [9] Viola, Paul: *Rapid Object Detection using a Boosted Cascade of Simple Features*. In: *Conference on Computer Vision and Pattern Recognition*, 2001. http://wearables.cc.gatech.edu/paper{}_of{}_week/viola01rapid.pdf.
- [10] Wächter, Carsten und Stefan Römer: *Gesichtserkennung I - Biometrische Systeme*, 2001. http://www.informatik.uni-ulm.de/nl/Lehre/WS01/HS-Biometrische-Systeme/ausarbeitungen/Gesichtserkennung{}_1{}_block.pdf.
- [11] Wiskott, Laurenz, Norbert Kr und Christoph Von Der Malsburg: *Face Recognition by Elastic Bunch Graph Matching*. *Intelligent Biometric Techniques in Fingerprint and Face Detection*, (11):355–396, 1999. <http://www.face-rec.org/algorithms/ebgm/wisfelkrue99-facerecognition-jainbook.pdf>.
- [12] Z. Wang, A. C. Bovik, H. R. Sheikh und E. P. Simoncelli: *Image quality assessment: From error visibility to structural similarity*. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. <http://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf>.