EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

GENERIERUNG SYNTHETISCHER ARABISCHER HANDSCHRIFT ZUR UNTERSTÜTZUNG VON AUTOMATISCHER ERKENNUNG HANDSCHRIFTLICHER TEXTE

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.–Ing.)

von **Dipl.–Ing. Laslo Dinges**

geb. am 18 März 1984 in Hildesheim

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik

der Otto-von-Guericke-Universität Magdeburg

Gutachter:

apl. Prof. Dr.–Ing. habil. Ayoub Al-Hamadi

Prof. Dr.–Ing. Abbas Omar

Prof. Dr.–Ing. Klaus Tönnies

Promotionskolloquium am: 12. November 2019

Zusammenfassung

Digitaler Text bietet viele Vorteile. Er ist kompakt, lässt sich ohne Aufwand vervielfältigen und zudem automatisiert analysieren und bearbeiten. Dennoch liegen auch heutzutage viele Dokumente in physischer Form vor. Zwar existieren robuste Lösungen zur automatischen Erkennung von Druckschrift, doch insbesondere die Erkennung arabischer Handschrift ist immer noch problematisch. Dies gilt insbesondere für zusammenhängende Texte, wie sie etwa in historischen Sammlungen vorkommen. Sowohl zur experimentellen Evaluierung als auch zum Trainieren von entsprechenden automatischen Lösungen zur Handschrifterkennung sind geeignete, umfangreiche Datenbanken von hoher Bedeutung. Aufgrund des mit deren Erstellung verbundenen Aufwandes weisen bestehende Datenbanken jedoch stets Einschränkungen des Umfangs, des verwendeten Vokabulars und des Detailgrades der beigefügten Grundwahrheiten auf. Eine Strategie, diesem Umstand zu begegnen, liegt darin, synthetische Proben zu erzeugen.

In dieser Arbeit wird ein neuartiger Ansatz zur Synthese arabischer Handschrift vorgestellt, welcher einen wesentlichen Beitrag zum Stand der Forschung darstellt. Bestehende Ansätze verknüpfen lediglich Bildausschnitte arabischer Buchstaben (was nur wenige unterschiedliche Synthesen ermöglicht) oder erzeugen keine vollständigen Handschriften. Dagegen ermöglicht es der hier vorgestellte Ansatz, eine Vielzahl unterschiedlicher Synthesen zu beliebigen arabischen Wörtern, Sätzen oder einspaltiger Textseiten zu erzeugen. Der Ansatz umfasst sowohl die Generierung neuer Buchstaben für jede Synthese, als auch Modifikationen des kompletten Schriftzuges, z.B. durch Änderung der Schriftneigung. Weiterhin werden zur Simulation von materialabhängigen Pigmentierungsschwankungen entwickelte Methoden vorgestellt.

Das Synthesesystem wird erfolgreich genutzt, um die IESK-arDB – eine während der Promotion entstandene, frei verfügbare Datenbank – zu ergänzen. Darüber hinaus werden in dieser Arbeit segmentierungsbasierte Verfahren zur automatischen

Erkennung arabischer Handschrift anhand realer und synthetischer Proben evaluiert und weiterentwickelt. Insbesondere die Klassifizierung von Buchstaben – durch optimierte Merkmale und einen auf Active-Shape-Models basierenden Klassifikator sowie eine kontextbasierte Fehlererkennung und -korrektur – stellt diesbezüglich einen zweiten wesentlichen Forschungsbeitrag dieser Dissertation dar. Außerdem wird ein verbesserter Ansatz zur Zeilensegmentierung mittels lokaler Gruppen vorgestellt, welcher auch die Detektion gekrümmter Textzeilen ermöglicht.

Abstract

Digital text provides a lot of advantages. It is compact and can automatically be duplicated, processed or analyzed. Nevertheless, even today many documents are given in physical form. Although there are robust solutions for recognizing machine printed text, the automatic recognition of handwritings is still challenging. This is especially true in case cohesive texts of historical collections. For both, the experimental evaluation and training of automatic solutions for handwriting recognition, comprehensive and suitable databases are crucial. Due to the accompanied costs, existing databases are limited in size, vocabulary and level of details of the added ground truth. One strategy to face this problem is to generate synthetic samples.

In this work, a novel approach of synthesizing Arabic handwritings is proposed, which is a major contribution to the state of the art.Existing approaches just concatenate images of Arabic characters (which just enables few different syntheses) or do not create complete handwritings. In contrast, the proposed approach enables to synthesize many variations of any Arabic words, sentences or one column text pages. The approach covers generation of new characters for each synthesis as well as modifications of the complete writing, as changing slant or skew. Furthermore, methods to simulate material depending variations in pigmentation are developed. The synthesis system is successfully used to augment the IESK-arDB, a database created during the PhD. Additionally, segmentation based procedures of automatic recognition of Arabic handwritings are refined and evaluated on real and synthetic data. Especially the classification of characters using optimized features and an approach based on Active Shape Modells as well as error detection and correction is a second major contribution of this thesis. Moreover, an improved approach of line segmentation using locale groups is proposed, which enables detection of curved text lines.

Schriftliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Insbesondere habe ich nicht die Hilfe einer kommerziellen Promotionsberatung in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, 04.02.2020

Laslo Dinges

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Akronym	Beschreibung
ASM	Active Shape Model (aktives Konturmodell)
ANNs	Artificial Neural Networks
BP	Branch Point (Verzweigungspunkt)
CC	Connected Component (zusammenhängende Komponente)
CNN	Convolutional-Neural-Networks (Deep learning)
GMM	Gaussian Mixture Model
GT	Grundwahrheiten
EP	End Point (Endpunkt in einem ausgedünntem Bild arabischer Handschrift)
HPAW	PAW ohne diakritische Zeichen.
ML	Machine Learning
LSTM	Long-Short-Term-Memory Netzwerke , zur Auswertung von Sequenziellen Proben konzipierte künstliche Neuronale Netze.
SVMs	Support Vector Machines
PAW	Piece of Arabic Word
PCA	Principal Component Analysis
SK	Segmentierungspunkt: Koordinate, an welcher zwei Buchstaben getrennt werden.

	Wichtige Formelzeichen:
$\acute{\mathbf{X}}$	Originale Trajektorien der online Buchstabendatenbank
$\mathbf{u}$	Durch ein ASM generierte Buchstabenprobe
N_b	Anzahl aller verwendeten Buchstabenklassen
$\mathcal{V}_{50k}, \mathcal{V}_{5k}, \mathcal{V}_{X}$	Bezeichnet das für ein Experiment verwendete Vokabular arabischer Wörter
$\mathbf{w}_r$	Die Sequenz von Buchstaben, welche das Ergebnis der automatischen Worterkennung darstellt.

$\mathbf{w}_t$	Die tatsächliche Sequenz von Buchstaben (entstammt den Grundwahrheiten)
$\mathcal{N}(\mu,\sigma)$	Eine Normalverteilung mit Erwartungswert μ und der Standardabweichung σ, die in dieser Arbeit oft zum beschreiben stochastischer Prozesse verwendet wird.
A_b	Treffergenauigkeit der Buchstabenerkennung (Prozentsatz der zum Testen reservierten Buchstabenproben, die korrekt klassifiziert wurden)
A'_w	Direkte Treffergenauigkeit der Worterkennung (Prozentualer Anteil von automatisch erkannten Wörtern mit $\mathbf{w}_r = \mathbf{w}_t$)
A_w	Treffergenauigkeit der Worterkennung nach Abgleich mit einem Vokabular
	Die wichtigsten über die Benutzeroberfläche steuerbare Parameter:
ς_P	Manipuliert den horizontalen Abstand benachbarter PAN (negative Werte führen zu Überlappenden PAW)
ς_{s_1}	Stauchung bzw. Streckung der Buchstaben
ς_{s_2}	Skalierung der Buchstaben und Manipulation der Auflösung der Wortsynthese
ς_l	Faktor zur Steuerung des Zeilenabstandes von Textsynthesen
ρ_r	Faktor zur Steuerung der Punktdichte beim Rendern von Handschriftsynthesen

Glossar

Begriff	Beschreibung
Basislinie	Bezieht sich in dieser Arbeit auf die untere Basislinie arabischer Handschrift. Diese tritt für gewöhnlich in Höhe der Kashidas auf, welche zwei Buchstaben miteinander verbinden.
Buchstaben Position	Die Position eines Buchstabens innerhalb eines PAW (isoliert, beginnend, mittig, endend). Die Gestalt eines Buchstabens hängt im Arabischen maßgeblich von seiner Position ab.
Diakritische Zeichen	Markierungen (meist Gruppen von Punkten) oberhalb oder unterhalb des Buchstabens.
Glyphen	Grafische Einheiten, hier für die Datensynthese aufbereitete Proben einzelner Buchstaben, Buchstaben mit Kashidas und Ligaturen.
Hauptkörper	Buchstabe ohne seine diakritischen Zeichen.
Hyperparameter	Parameter, dessen Wert extern bestimmt wird (wie z.B. die Lernrate, C und γ von SVMs etc.). Im Gegensatz dazu wird der Wert eines Parameters intern (innerhalb eines Modelles) bestimmt, z.B. die Gewichte eines ANNs.
Kashida	Verbindung eines Buchstaben mit seinem Vorgänger bzw. Nachfolger.
Ligatur	Verschmelzung zweier oder mehrerer Buchstaben zu einer Glyphe.
online	Datenaufnahme erfolgt während der Schreibvorgangs via Sensorik und/oder Videoaufnahme (Proben enthalten Trajektorien).
offline	Datenaufnahme erfolgt nach dem Schreibvorgang via. Scanner, Buchscanner etc. (Proben bestehen aus Abbildern, d.h. nur das Resultat des Schreibvorgangs liegt vor, keine Trajektorien).
Pen Down	Anfangspunkt eines Striches (Position an der der Stift zuerst das Papier berührt)
Pen Up	Endpunkt eines Striches

Unicode	Kodierung von Text (meist UCS-2 oder UTF-16), die im Gegensatz zu ASCI 16 Bit statt 8 Bit pro Symbol verwendet, so dass sich 65.536 statt 256 Zeichen dargestellen lassen. In dieser Dissertation wird der Begriff Unicode stellvertretend auch für UTF-8 Kodierten Text-Input verwendet, der intern in UTF-16 umgewandelt wird. Entscheidend ist, dass neben dem englischem Alphabet auch alle arabische Buchstaben dargestellt werden.
UTF-8	Kodierung, bei der Unicode Symbole durch Kombination mehrere 8-Bit Symbole (also mit dem ASCII Alphabet) gespeichert werden. Dies ist wichtig, da viele Programme und Dateiformate keinen herkömmliche Unicode Kodierung unterstützen.

KAPITEL 1

Einleitung

In der Menschheitsgeschichte waren Kulturen, die eine Schrift entwickelt hatten, entscheidend im Vorteil. Ihnen war es möglich, Informationen unverfälscht über mehrere Generationen hinweg zu bewahren. Doch weniger als zehn Prozent aller antiken Texte haben bis heute überdauert. Über die Jahrhunderte fielen die meisten Handschriften Zersetzungsprozessen, Katastrophen und Kriegen zum Opfer oder wurden gar systematisch vernichtet. Viele der noch erhaltenen Dokumente haben eine kulturelle oder historische Bedeutung, und ermöglichen Genealogen beispielsweise die Rekonstruktion von Familienstammbäumen. In anderen verbergen sich erstaunliche Erkenntnisse zu philosophischen oder wissenschaftlichen Disziplinen.

Technische Errungenschaften, die heute unseren Lebensstandard erhöhen, wurden schrittweise und teilweise auf hunderte Jahre älterem Wissen aufbauend entwickelt. Kontaktlinsen, die heute das Leben vieler erleichtern, basieren auf Theorien, die bereits im 16. Jahrhundert erarbeitet wurden. Auch moderne Teleskope, mit deren Hilfe wir heute die Beschaffenheit des Universums in ungeahnter Schärfe entdecken können, wären nicht denkbar, ohne die im 10 Jh. von Alhazan entdeckte Eigenschaft von gewölbtem Glas, eingehende Lichtwellen zu brechen. Einige bedeutende Erkenntnisse gerieten jedoch für viele Jahrzehnte oder Jahrhunderte in Vergessenheit. So wurden etwa erst vor wenigen Jahren alte medizinische Aufzeichnungen wiederentdeckt, die dokumentieren, wie sich aus Weihrauch ein Medikament gegen Rheuma entwickeln lässt. Man fand heraus, dass dieses den

modernen Pharmazeutika hinsichtlich Effizienz und Verträglichkeit deutlich überlegen ist. Sogar die für die moderne Physik so elementare String Theorie basiert auf philosophischem Gedankengut, das sich über 2000 Jahre zurückdatieren lässt. Daher ist anzunehmen, dass weitere Dokumente existieren, die solch wichtiges Gedankengut bis heute bewahren. Da historische Dokumente zum großen Teil nur fragmentarisch und ungeordnet vorliegen, stellt der bloße Erhalt solch bedeutender Informationen für sich genommen noch nicht deren Nutzbarkeit für die Gesellschaft sicher.

Sowie die Erfindung des Buchdrucks hohe Auflagen bedeutender Werke und somit deren Verbreitung und Sicherung ermöglichte, lassen sich Handschriften durch moderne bildgebende Verfahren digitalisieren und damit vervielfältigen, ohne hierzu eigens eine kostenintensive Abschrift anfertigen zu müssen. Obwohl dies prinzipiell einen kostenneutralen und direkten Zugriff auf eine Vielzahl von Handschriften ermöglicht, bliebe die gezielte Suche nach Dokumenten mit bestimmten Inhalten mit einem enormen Aufwand verbunden. Durch eine automatische Erkennung der in den digitalisierten Handschriften enthaltenen Wörtern ergäbe sich hingegen die Möglichkeit, deren Inhalte computergestützt zu analysieren.

Sogar im heutigem Zeitalter mit steigernder Verfügbarkeit von digitalen Medien und Mensch-Maschine-Schnittstellen, stellen Papier und Stift ein so weit verbreitetes Medium dar, dass für Verfahren zur Handschrifterkennung auch abseits historischer Dokumente vielfältige Anwendungsmöglichkeiten existieren. Dies gilt auch für arabische Handschrift. Ihre Eigenheiten erschweren zwar eine automatische Erkennung, da sich die Schrift jedoch im Laufe der letzten Jahrhunderte kaum verändert hat, lassen sich einmal gewonnene Erkenntnisse besonders vielseitig nutzen.

1.1 Motivation

Viele der bis heute erhaltenen Handschriften sind in arabischer Schrift verfasst. Das arabische Alphabet ist neben dem lateinischen das am weitesten verbreitete der Welt [64]. Arabisch ist die Muttersprache von über 300 und Zweitsprache von mehr als 160 Millionen Menschen. Auch umfangreiche Sammlungen nicht arabisch sprachiger Länder, wie die mehrere hundert Millionen Dokumente umfassenden Osmanischen Archive, sind in arabischer Schrift verfasst [98], [17]. Daher übersteigt eine manuelle Transkription solch umfangreicher Sammlungen – oder auch nur die

Einordnung aller Dokumente anhand der behandelten Themen, ihrer Verfasser der Epoche etc. – die verfügbaren Kapazitäten. Es liegt demnach nahe, diesen Prozess mit Hilfe von Methoden der Bildverarbeitung zu automatisieren.

Die Erkennung von Druckschrift war eines der ersten Projekte im Bereich der Mustererkennung. Da Druckschrift – im Gegensatz zur Handschrift – lediglich diskrete Variationen (Schriftart und Schriftgröße) und drucktechnisch bedingte, geringfügige Abweichungen aufweist, konnten auch mit eingeschränkten Ressourcen zufriedenstellende Ergebnisse erzielt werden. Die zunehmend leistungsfähigere Hardware ermöglichte in den letzten Jahren auch die Erkennung von Handschrift, während gleichzeitig viele Archive mit der Digitalisierung ihrer Dokumente begannen. Daher rückt auch die Erkennung von Handschrift immer mehr in den Forschungsmittelpunkt. Zudem lässt sich Handschrifterkennung auch im Alltag praktisch nutzen. Sind die Rahmenbedingungen bekannt und lassen sich – wie bei der Erkennung von Geldbeträgen in Checks oder Einträgen in Formularen – stark eingrenzen, erleichtert dies die Implementation eines robusten Erkennungssystems. Viele syrische Flüchtlinge können deutschen Behörden beispielsweise nur Dokumente vorlegen, die in arabischer Sprache verfasst und teilweise handschriftlich ausgefüllt sind. Auch hier ist eine automatische Lösung wünschenswert, welche wichtige Eckdaten relevanter Dokumente (wie beispielsweise Geburtsurkunden) erkennt und für den Sachbearbeiter übersetzt.

Bei einem geringen Vokabular von maximal wenigen hundert Wörtern lässt sich eine holistische Erkennung umsetzen, indem für jedes Wort Proben gesammelt werden. Vorzugsweise sind dies authentische Dokumente, die von der zuständigen Behörde oder Firma archiviert wurden (deren Veröffentlichung zu Forschungszwecken ist jedoch aufgrund der Datenschutzbestimmungen problematisch). Ab einem Vokabular von mehreren hundert oder tausend Wörtern, wird der Einsatz segmentierungsbasierter Handschrifterkennung erforderlich. Dies gilt auch, falls das Vokabular erst nach dem Erstellen der Datenbank oder des Trainingsprozesses definiert oder verändert wird: da hier zunächst Buchstaben und keine ganzen Wörter erkannt werden, ist es bei segmentierungsbasierten Ansätzen nicht erforderlich, dass das Vokabular der Trainings- und Testdatenbanken mit den in der Praxis vorkommenden Wörtern übereinstimmt.

Bei segmentierungsbasierten Verfahren wird aufgrund der erkannten Buchstaben auf das zugrundeliegende Wort geschlossen. Das hierzu verwendete Vokabular

ist im Falle zusammenhängender Texte deutlich größer, als bei der Erkennung von Handschrift in Vordrucken wie Checks. Da auch die notwendige Segmentierung der einzelnen Wörter fehleranfällig ist, resultiert eine im Vergleich zu holistisch lösbaren Problemen geringere Zahl korrekt erkannter Wörter. Die erforderliche, minimale Treffergenauigkeit einer rentablen automatischen Lösung hängt jedoch maßgeblich vom Anwendungsgebiet und der Effizienz der manuellen Verarbeitung ab. Um Handschrift zur Gänze automatisch in digitalen Text zu transkribieren oder in andere Sprachen zu übersetzen, ist etwa eine erheblich zuverlässigere Erkennung erforderlich, als bei der Suche nach einigen Schlüsselwörter.

Den unterschiedlichen Ansätzen der Schrifterkennung ist gemein, dass sie zum Training und Testen geeignete Datenbanken benötigen. Die bestehenden Datenbanken arabischer Handschrift weisen jedoch entweder ein sehr eingeschränktes Vokabular auf, oder bestehen aus Textseiten mit sehr vereinfachten Grundwahrheiten. Die verfügbaren Datenbanken allein sind daher nicht hinreichend, um die Güte eines Erkennungssystems im Hinblick auf das verwendete Vokabular zu untersuchen. Zudem lassen sich bestimmte Verfahren, wie die Segmentierung von Wörtern in Buchstaben, an diesen Datenbanken nicht akkurat oder nur manuell auswerten. Um die Einschränkungen der bestehenden Datenbanken zu umgehen, lassen sich Verfahren zur Synthese von handschriftlichen Proben einsetzen. Im Gegensatz zu statischen Datenbanken, ermöglicht die Datensynthese zudem eine dynamische Anpassung an neue Forschungsziele. Darüber hinaus lassen sich bestimmte Eigenschaften der Proben gezielt in ihrer gewünschten Intensität generieren, wodurch sich zusätzliche Forschungsaspekte ergeben. Weiterhin ist eine nachträgliche Erweiterung des Umfangs einer Datenbanken mit einem Synthesesystem unkompliziert und mit geringem Aufwand durchführbar.

1.2 Besonderheiten Arabischer Schrift

Wie die lateinische, so basiert auch die arabische Schrift auf einem Alphabet von Symbolen, die für verschiedene beim Sprechen erzeugte Laute stehen. Die arabische Schrift ist der lateinischen demnach sehr viel ähnlicher, als dies beispielsweise bei der japanischen der Fall ist, welche Silben und ganze Wörter durch spezifische Schriftzeichen ausdrückt. Dennoch wirkt arabische Schrift oft genauso fremd, da Buchstaben selbst in Druckschrift miteinander verbunden und zudem organischer

Tabelle 1.1. Das reguläre arabische Alphabet.

	i	e	m	b		i	e	m	b
Alif	ا	ا			Dhad	ض	ض	ضـ	ضـ
Ba	ب	ب	بـ	بـ	Taa	ط	ط	طـ	طـ
Ta	ت	ت	تـ	تـ	Dha	ظ	ظ	ظـ	ظـ
Tha	ث	ث	ثـ	ثـ	Ayn	ع	ع	عـ	عـ
Jim	ج	ج	جـ	جـ	Ghayn	غ	غ	غـ	غـ
Ha	ح	ح	حـ	حـ	Fa	ف	ف	فـ	فـ
Kha	خ	خ	خـ	خـ	Qaf	ق	ق	قـ	قـ
Dal	د	د			Kaf	ك	ك	كـ	كـ
The	ذ	ذ			Lam	ل	ل	لـ	لـ
Ra	ر	ر			Mim	م	م	مـ	مـ
Zai	ز	ز			Nun	ن	ن	نـ	نـ
Sin	س	س	سـ	سـ	He	ه	ه	هـ	هـ
Chin	ش	ش	شـ	شـ	Waw	و	و		
Sad	ص	ص	صـ	صـ	Ya	ي	ي	يـ	يـ
Tamabutra	ة	ة							

und weniger akzentuiert sind, als lateinische, griechische oder kyrillische Schriftzeichen.

In Tabelle 1.1 wird das reguläre arabische Alphabet dargestellt. Dieses grenzt, zusammen mit dem häufig gebrauchtem Sonderzeichen ة ة, die Menge der Zeichen ein, welche in den folgenden Kapiteln dargelegten Forschungsfragen zur Synthese und Erkennung von Handschrift verwendet wird. In arabischen Texten kommen zudem weitere Zeichen vor. Hierzu zählt in erster Linie die Menge der Satzzeichen {. ؛ ؟ ،}, die arabischen Ziffern, sowie verschiedene Sonderformen einiger Buchstaben wie z.B. ﺍ oder أ. Diese sind nicht Teil der folgenden Untersuchungen, sollen aber in zukünftigen Projekten die verwendete Zeichenmenge ergänzen.

Im arabischen Alphabet existieren sechs Buchstaben, die – ausgenommen am Ende eines Wortes – stets neue, sich zumeist überlappende Part-of-Arabic-Words

(PAWs) einleiten. Wie sich Tabelle 1.1 entnehmen lässt, existieren diese sechs Buchstaben nur in **i**solierter oder in **e**ndender Position, wogegen alle weiteren Buchstaben auch in **m**ittiger und **b**eginnender Position auftreten. Die Position bezieht sich dabei auf die Stelle im PAW. Ähnlich groß und kleingeschriebener lateinischer Buchstaben, hängt die Gestalt arabischer Buchstaben maßgeblich von deren Position ab. Aus der Buchstabenfolge ك ي ل و س ا ك ل ergibt sich so etwa das aus vier PAWs bestehende Wort الكَاشوليك.

Sofern sich PAWs nicht überlappen, erwecken sie häufig den Eindruck separater Wörter. Diese Eigenschaft des Schriftbildes spiegelt sich jedoch nicht in der Aussprache wieder, da PAWs keine Silben darstellen. Da es im Arabischen keine Blockschrift gibt, werden Buchstaben innerhalb eines PAW sowohl in Hand- als auch Druckschrift immer miteinander verbunden. Die Verbindung zweier Buchstaben wird Kashida genannt und erfolgt meist auf Höhe der unteren Basislinie. Die Länge der Kashida variiert dabei deutlich stärker, als dies bei lateinischer Schrift üblich ist. Ein Kashida kann die Breite eines Buchstabens übertreffen oder in einigen Fällen gänzlich verschwinden. Letzteres führt unter Umständen dazu, dass Buchstaben übereinander als Ligaturen geschrieben werden (wie beispielsweise ني).

Fünfzehn der achtundzwanzig arabischen Buchstaben weisen diakritische Zeichen über oder unterhalb der Basislinie auf [64]. Diese bestehen aus einzelnen Punkten oder Gruppen aus zwei bis drei, im Falle von auf dem arabischen Alphabet basierenden Sprachen wie Urdu, Farsi oder Persisch bis zu vier Punkten. Hierdurch wird unter anderem die Zeilensegmentierung von Textseiten erschwert, da sich die diakritischen Zeichen bei eng beschriebenen Dokumenten zwischen den Zeilen befinden. In Schreibschrift werden darüber hinaus Gruppen aus zwei oder drei Punkte gelegentlich als Striche oder Wellenlinie dargestellt, was die eindeutige Klassifizierung der diakritischen Zeichen erschwert. Hinzu kommt, dass bei arabischer Schrift vermehrt eine Ähnlichkeit zwischen Buchstaben und Teilen von anderen Buchstaben zu beobachten ist. Daher ist es nicht ohne weiteres möglich, funktionierende Erkennungssysteme für z.B. lateinische Schrift direkt für arabische Schrift anzupassen. Vielmehr müssen neue, spezielle Lösungen entwickelt werden, um den Rückstand bezüglich arabischer Handschrifterkennung aufzuholen.

Im Folgenden werden die für das Verständnis der folgenden Kapitel wesentlichen Besonderheiten der arabischen Schrift zusammengefasst:

- Im Arabischen wird von rechts nach links geschrieben.

- Das arabische Alphabet besteht aus 28 Buchstaben.

- Es wird nicht zwischen Groß- und Kleinschreibung unterschieden, allerdings hat die Position eines Buchstabe teilweise dramatische Auswirkungen auf seine Gestalt.

- Die Buchstaben ا, د, ذ, ر, ز und و treten nur in isolierter oder endender Position auf. Ein solcher Buchstabe wird niemals mit seinem Nachfolger verbunden, wodurch er ein Wort in Pieces of Arabic Words (**PAWs**) aufteilt (sofern es sich nicht ohnehin um den letzte Buchstabe des Wortes handelt).

- PAWs können sich überlappen und werden dann teilweise übereinander geschrieben.

- Zwei aufeinanderfolgende Wörter werden, im Gegensatz zu PAWs, in digitalem Text durch ein Leerzeichen eindeutig separiert. Diese Eigenschaft spiegelt sich in Handschrift jedoch nicht unbedingt durch den Abstand zwischen PAWs bzw. Wörtern wieder, wodurch die Segmentierung von Textzeilen in Wörter stark erschwert wird.

- Benachbahrte Buchstaben innerhalb eines PAW werden durch sogenannte Kashida verbunden.

- Etwa die Hälfte der arabische Buchstaben weisen diakritische Zeichen auf (zumeist über und bei einigen Buchstaben unterhalb der unteren Basislinie).

- Achtzehn Buchstaben unterscheiden sich ausschließlich durch die Anzahl oder Position der diakritischen Zeichen eindeutig von einem oder mehreren der anderen Buchstaben.

- Im Arabischen gibt es keinen so grundlegenden Unterschied zwischen Druckschrift, zeitgenössischer Schreibschrift und historischen Dokumenten, wie dies bei lateinischen Schriften der Fall ist. Dies liegt daran, dass die meisten handschriftlichen arabischen Dokumente im Naskh Stil geschrieben sind [69], welcher der Druckschriften wie ArabTex oder der Type „Traditional Arabic“ ähnelt.

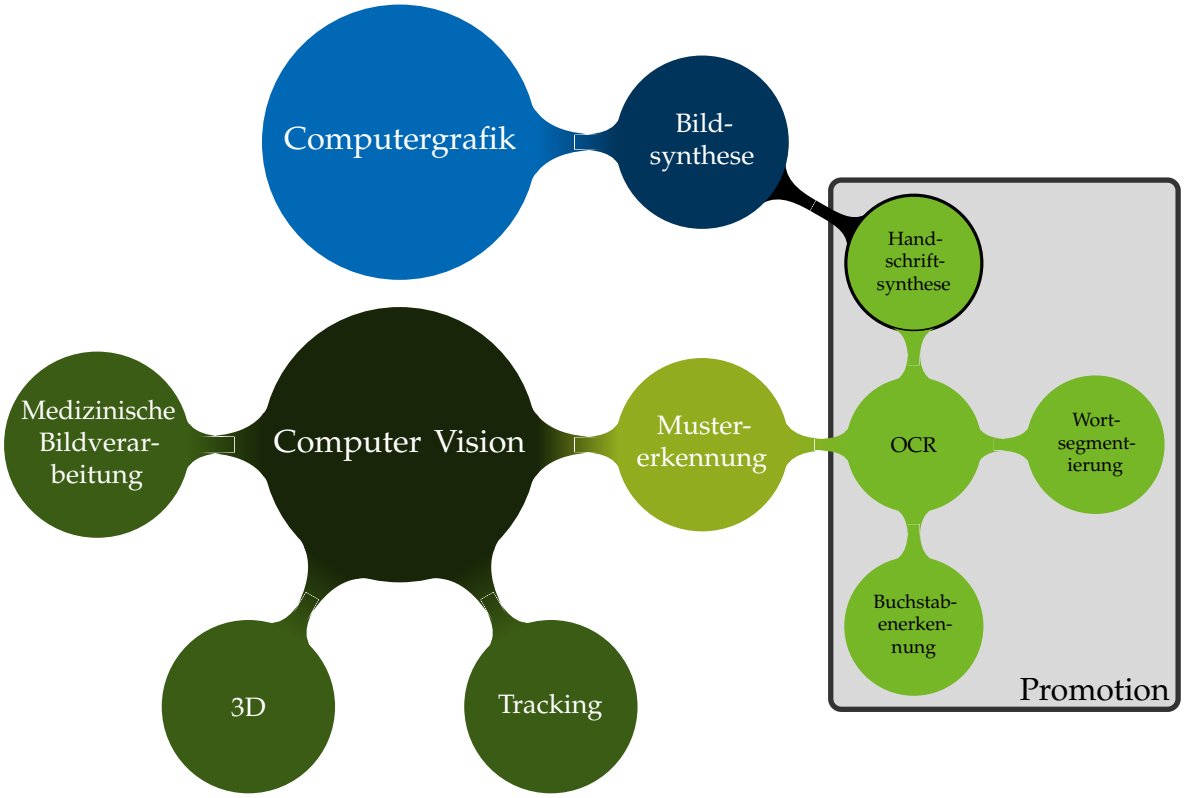


Abbildung 1.1. Einordnung der in dieser Dissertation behandelten Forschungsfelder.

1.3 Ziele und Einordnung der Arbeit

Die wesentlichen Forschungsbeiträge dieser Dissertation betreffen die segmentierungsbasierte Erkennung arabischer Handschrift und die Synthese von arabischen Handschriftproben. Abbildung 1.1 veranschaulicht, wie die Schwerpunkte der Arbeit den beiden Bereichen Computergrafik und Computer Vision zuzuordnen sind.

Das erste Ziel dieser Arbeit besteht darin, ein Verfahren zur Synthese arabischer Handschrift zu erforschen, welches die Generierung naturgetreuer Proben von beliebigen arabischen Wörtern ermöglicht, ohne dass für diese bereits Proben vorhanden sein müssen. Zudem sollen automatisch Grundwahrheiten erzeugt werden, welche die Validierung und Evaluierung möglichst vieler Verfahren aus den Bereichen Vorverarbeitung, Segmentierung und Klassifikation unterstützen. Hierdurch sollen die bestehenden Datenbanken nicht gänzlich ersetzt, jedoch deren Schwächen durch den vorgestellten Syntheseansatz ausgeglichen werden, um weitere Forschungsprojekte im Bereich arabischer Handschrifterkennung zu unterstützen.

Das zweite Ziel besteht in der Erforschung eines auf Segmentierung basierenden

Verfahrens zur Erkennung arabischer Handschrift. Hiermit soll zum einen ausgewertet werden, wie zuverlässig sich Datenbanken mit synthetischen Proben zur Evaluierung verschiedener Methoden eignen (dies betrifft sowohl die Eigenschaften der generierten Abbildungen als auch der Grundwahrheiten im Vergleich zu klassischen Datenbanken). Zum anderen sollen die Einsatzmöglichkeiten dieses Ansatzes zur Worterkennung im Hinblick auf Aufgaben der Dokumentanalyse erforscht werden. Hierzu werden verschiedene Verfahren zur Segmentierung und Klassifikation untersucht und verglichen. Die Datensynthese ermöglicht es hierbei unter anderem, gezielt den Einfluss bestimmter Eigenschaften der Handschrift auf die Güte dieser Verfahren sowie der resultierenden Worterkennung zu analysieren.

1.4 Gliederung der Arbeit

Die vorliegende Arbeit gliedert sich in mehrere Kapitel, die an dieser Stelle kurz erläutert werden sollen. Eine grafische Übersicht der Kapitel findet sich zudem in Abbildung 1.2 auf Seite 11.

Im Anschluss an dieses einleitende Kapitel folgen:

Kapitel 2 In diesem Kapitel wird der Stand der Technik beschrieben, um sowohl eine Übersicht über die wichtigsten Ansätze zur Handschrifterkennung und -synthese zu geben, als auch die wesentlichen Methoden vorzustellen, welche für das Verständnis der weiteren Kapiteln erforderlich sind.

Kapitel 3 Dieses Kapitel erläutert das vorgeschlagene Verfahren zur Synthese arabischer Handschrift. Dies umfasst die Akquisition und Aufbereitung der erforderlichen Proben handschriftlicher Buchstaben, Methoden zur Zusammenfügung derselben zu Wörtern oder Texten, sowie Ansätze, um unterschiedliche Eigenschaften diverser Schreibutensilien beim Rendering-Prozess zu simulieren.

Kapitel 4 Hier wird beschrieben, wie das Syntheseverfahren aus dem letzten Kapitel zur Generierung von Datenbanken eingesetzt wird. Zudem erfolgt eine Analyse verschiedener Datenbanken arabischer Handschrift. Abschließend wird der Aufbau der erzeugten synthetischen Datenbanken sowie der zugehörigen Grundwahrheiten erläutert.

Kapitel 5 In diesem Kapitel werden die wesentlichen Methoden zur Vorverarbeitung sowie zur Segmentierung handschriftlicher arabischer Wörter in PAWs und Buchstaben dargelegt. Zudem wird ein Ansatz zur Segmentierung von Zeilen mit nichtlinearen Basislinien vorgestellt.

Kapitel 6 Hier werden die im Rahmen dieser Promotion untersuchten Methoden zur Buchstabenerkennung vorgestellt. Diese umfassen strukturelle Merkmale zur Reduzierung der potentiellen Klassen sowie verschiedene statistische Merkmale. Weiterhin werden geeignete Klassifikatoren beschrieben und verglichen.

Kapitel 7 Dieses Kapitel legt dar, wie – ausgehend von den Ergebnissen der letzten beiden Kapitel – ein Verfahren zur Worterkennung entwickelt wird. Im Vordergrund steht hierbei die Detektion und Behebung von Segmentierungs- und Klassifikationsfehlern.

Kapitel 8 In diesem Kapitel werden die vorgestellten Verfahren experimentell evaluiert. Der Fokus liegt auf der Worterkennung und den hierbei beteiligten Verfahren zur Segmentierung der Wörter in Buchstaben und deren Klassifizierung.

Kapitel 9 Im letzten Kapitel werden alle gewonnenen Erkenntnisse zusammengefasst, bestehende Probleme des behandelten Forschungsfeldes diskutiert und ein Ausblick über potentielle Schwerpunkte zukünftiger Arbeiten gegeben.

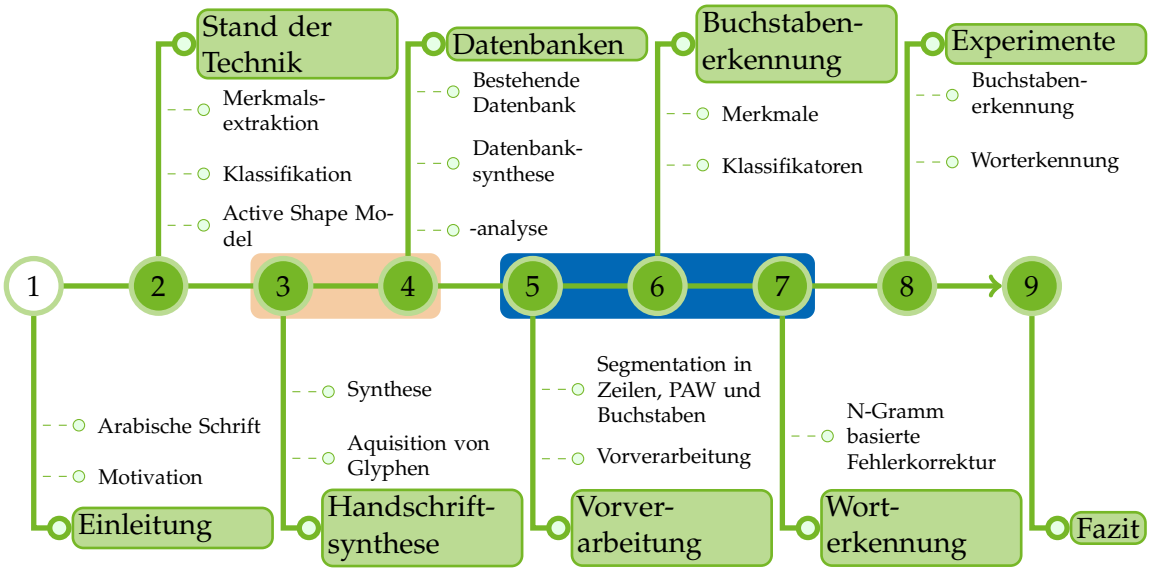


Abbildung 1.2. Vereinfachte Darstellung der Kapitel dieser Promotionsschrift. Die beige eingefassten Kapitel behandeln den Schwerpunkt der Handschriftsynthese und Datenbanken, die blau eingefassten den der Handschrifterkennung.

1.5 Bisherige eigene Veröffentlichungen

Die vorliegende Dissertation basiert zu Teilen auf den im Folgenden gelisteten Veröffentlichungen in internationalen Fachzeitschriften und Konferenzen.

Eigene internationale Journal-Paper

1) Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, and Sherif El-etriby. Synthesis of common arabic handwritings to aid optical character recognition research. *Sensors*, 16(3):346, 2016 (ISI Impact factor 2.677).

2) Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, Sherif El etriby, and Ahmed Ghoneim. Asm based synthesis of handwritten arabic text pages. *Scientific World Journal*, 2015 (5 Jahres ISI Impact Factor (2013): 1.300).

3) Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, Zaher Al-Aghbari, and Hassan Mustafa. Offline automatic segmentation based recognition of handwritten arabic words. *International Journal of Signal Processing, Image Processing and Pattern Recognition (IJSIP)*, 4:131–144, 2011.

Eigene internationale Konferenz-Paper

4) Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, and Andreas Nürnberger. Automatic recognition of common arabic handwritten words based on ocr and n-grams. In *ICIP*, 2017.

5) Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, and Sherif El-etriby. Synthetic based validation of segmentation of handwritten arabic words. *Manuscript Cultures*, 1:10–18, 2015.

6) Laslo Dinges, Ayoub Al-Hamadi, and Moftah Elzobi. A locale group based line segmentation approach for non uniform skewed and curved arabic handwritings. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 803–806, 2013.

7) Laslo Dinges, Ayoub Al-Hamadi, and Moftah Elzobi. An approach for arabic handwriting synthesis based on active shape models. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1292–1296, 2013.

8) Laslo Dinges, Ayoub Al-Hamadi, and Moftah Elzobi. An active shape model based approach for arabic handwritten character recognition. In *The 11th International Conference on Signal Processing (ICSP'2012)*, pages 1194–1198, 2012.

9) Laslo Dinges, Moftah Elzobi, Ayoub Al-Hamadi, and Zaher Al-Aghbari. Synthizing handwritten arabic text using active shape models. In *Image Processing and Communications Challenges 3*, volume 102 of *Advances in Intelligent and Soft Computing*, pages 401–408. Springer, 2011.

Internationale Publikationen als Co-Author

10) Moftah Elzobi, Ayoub Al-Hamadi, Laslo Dinges, and Sherif El-etriby. *Advances in Visual Computing: 11th International Symposium, ISVC 2015, Las Vegas, NV, USA, December 14-16, 2015, Proceedings, Part II*, chapter CRFs and HCRFs Based Recognition for Off-Line Arabic Handwriting, pages 337–346. Springer International Publishing, Cham, 2015.

11) Moftah Elzobi, Ayoub Al-Hamadi, Zaher Al Aghbari, Laslo Dinges, and Anwar Saeed. Gabor wavelet recognition approach for off-line handwritten arabic using explicit segmentation. In *Image Processing and Communications Challenges 5*, pages 245–254. Springer International Publishing, 2014.

12) Moftah Elzobi, Ayoub Al-Hamadi, and Laslo Dinges. A hidden markov model-based approach with an adaptive threshold model for off-line arabic handwriting recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 945–949, 2013.

13) Moftah Elzobi, Ayoub Al-Hamadi, Zaher Al Aghbari, and Laslo Dinges. Ieskardb: a database for handwritten arabic and an optimized topological segmentation approach. *IJDAR*, 16(3):295–308, 2013.

14) M. Elzobi, A. Al-Hamadi, A. Saeed, and L. Dinges. Arabic handwriting recognition using gabor wavelet transform and svm. In *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, volume 3, pages 2154–2158, 2012.

15) Moftah Elzobi, Ayoub Al-Hamadi, Laslo Dinges, and Bernd Michaelis. A structural feature based segmentation for off-line handwritten arabic text. In *5th International Symposium on Image/Video Communication over Fixed and Mobile Networks, ISIVC 2010 . - IEEE*, pages 1–4, (Rabat, Marocco),, 2010.

KAPITEL 2

Stand der Technik

In diesem Kapitel werden Methoden des Stands der Technik beschrieben, die im Bezug zu Schriftsynthese oder -erkennung stehen. Zunächst werden die für arabische Handschrift verfügbaren Datenbanken sowie Ansätze zur Synthetisierung entsprechender Daten vorgestellt. Darüber hinaus werden im Wesentlichen Methoden der Bildverarbeitung verglichen, die sich bereits hinsichtlich der automatischen Erkennung von arabischer oder anderer Handschrift bewährt haben.

2.1 Datenbanken

In diesem Abschnitt sollen die zur Schrifterkennung verfügbaren Datenbanken vorgestellt und deren Eignung für die automatische Schrifterkennung diskutiert werden.

Große Mengen an Datenproben sind insbesondere für Trainings- und Validierungsprozesse von entscheidender Bedeutung, um das Verhalten von einzelnen Methoden und ganzen Systeme in späteren realen Einsätzen verlässlich abschätzen zu können. Auch die durch ausreichend Stichproben ermöglichte Optimierung verschiedener Parameter und das Training von Klassifikatoren ist im hohen Maße von der Verfügbarkeit einer geeigneten Trainingsdatenbank abhängig. Genügend Stichproben sind allerdings nicht immer verfügbar und das Akquirieren neuer handgeschriebener Stichproben ist als fehleranfälliger, arbeits- und zeitintensiver Prozess bekannt [62].

Aufgabenfelder der Mustererkennung lassen sich in viele unterschiedliche Teilaspekte wie z.B. Personen-, Emotions- und Schmerzerkennung oder Briefkopf- und Texterkennung oder Word-Spotting in historischen Dokumenten untergliedern. Die Entwicklung von entsprechenden Lösungen ist dabei zumeist an eine konkrete praktischen Anwendungen gebunden, die spezifische Anforderungen an die zu verwendenden Datenbanken stellt. Daher werden Datenbanken oft nicht generalisiert, sondern für ein spezifisches Problem eines aktuellen Projektes entwickelt, was den Nutzen für andere Projekte mindert. Daher erfüllen bestehende Datenbanken – sofern überhaupt vorhanden – die Anforderungen neuer Projekte oft nur fragmentarisch. Reale Daten aus dem Einsatzgebiet der zu implementierenden Mustererkennungslösung unterliegen oftmals Datenschutzbestimmungen, was den mit dem Erstellen einer Datenbank verbundenen Aufwand und Schwierigkeitsgrad deutlich erhöht. So existiert beispielsweise eine verfügbare Datenbank, die den Verlauf der Mimik unter Einfluss unterschiedlicher Schmerzgrade darstellen soll [66], jedoch wurden diese von den Probanden lediglich durch Schauspiel simuliert. Um eine zuverlässige Schmerzerkennung trainieren zu könne, war es daher erforderlich, in der Arbeitsgruppe eine eigene, auf kontrolliert zugefügten Hitzeschmerzen basierende Datenbank zu erarbeiten [95]. Im Bereich der Schrifterkennung weisen die bestehenden, frei verfügbaren Datenbanken vergleichbare Einschränkungen auf. Dies gilt insbesondere bezüglich arabischer Handschrift, da sich vielfältige Variationen aufgrund regionaler Schreibweise, der verwendeten Wörter (z.B. Wortlänge) oder dem Fokus auf Wörter, Sätze oder ganze Dokumente ergeben. Auch die Art der benötigten Grundwahrheiten (GT) unterscheidet sich deutlich. Je nach Anwendung ist die Etikettierung einer Stichprobe durch Unicode hinreichend, oder es müssen die genauen Positionen aller PAWs, Buchstaben und diakritischen Zeichen dokumentiert werden. Trotz der Bemühungen während des letzten Jahrzehnts, geeignete, frei verfügbare Datenbanken als Forschungsgrundlage arabischer Handschrift zu schaffen, gibt es derer bisher nur wenige. Im Folgenden sollen sie vorgestellt und ihre wesentlichen Eigenschaften erläutert werden.

2.1.1 Bestehende arabische Schriftdatenbanken

Eine Übersicht der einschlägigen Datenbanken für arabische Schrifterkennung findet sich in Tabelle 2.1 auf Seite 17 (auf die im Rahmen dieser Promotion entstandene IESK-arDB wird in Kapitel 4 näher eingegangen). Die Datenbanken lassen sich

in die Kategorien Druckschrift, handschriftliche Dokumente und handschriftliche Wörter einteilen:

- **Druckschrift:** Es gibt mehrere einschlägige Datenbanken arabischer Druckschrift. Die *DARPA*-Datenbank aus dem Jahre 1997 umfasst 345 mit 600 dpi digitalisierte Seiten aus arabischen und persischen Dokumenten wie Büchern oder Zeitungen, wobei die Grundwahrheiten (GT) den Unicode zu allen enthaltenen Texten enthalten. Die Datenbank ist im Rahmen bestimmter Forschungsprojekte frei verfügbar. Die *Erim* Datenbank enthält 750 Seiten aus arabischen Büchern und Magazinen, welche unterschiedliche Schriftarten und Degradationsgrade aufweisen [84]. Die Daten wurden als Graubilder (300 dpi) digitalisiert, und werden durch umfangreiche hierarchische GT auf Zeilen Wort und Buchstabenebene ergänzt. Ursprünglich sollte die Datenbank als CD-Rom erwerblich sein, allerdings konnte sich die weniger umfangreiche *DARPA* Datenbank durchsetzen. Frei verfügbar ist zudem die mit 45.313.600 Wörtern sehr umfangreiche APTI Datenbank [89].

- **Handschriftliche Dokumente:** Die OpenHaRT Datenbank [7] eignet sich, um Verfahren der Texterkennung sowie der Zeilen- und Wortsegmentierung zu validieren. GT für die Validierung der Segmentierung von Wörter in Buchstaben sind jedoch nicht vorhanden. Die Lizenz bedingt zudem die Teilnahme an der OpenHaRT Evaluation und ist auf deren Dauer beschränkt, weshalb es sich nicht um eine frei verfügbare Datenbank handelt. Die auch für Forschungszwecke kostenpflichtige AMA Datensammlung enthält 5000 Seiten Formulare, Gedichte, Notizen und Diagramme, wobei als GT jeweils nur die Positionen der PAWs verfügbar sind.

- **Handschriftliche Wörter** Die erste frei verfügbare, umfangreiche Datenbank handschriftlicher arabische Wörter ist die IFN\ENIT Datenbank aus Braunschweig [62]. Die Datenbank wurde mit dem Hintergrund der automatischen Sortierung von Briefen innerhalb Tunesiens entwickelt. Sie enthält ein Vokabular von 937 verschiedener tunesischer Ortsnamen sowie der dazugehörigen Postleitzahlen. Von 411 Schreibern wurden jeweils 60 Stichproben zufällig ausgewählter Ortsnamen akquiriert, wobei die insgesamt 26.459 Stichproben in fünf Sets gegliedert wurden, um die einheitliche Auswahl von Trainings-

und Testdaten sicherzustellen. Zu allen Stichproben wurden GT erstellt, welche den Unicode für die Postleitzahl und den Stadtnamen und außerdem die manuelle Bestimmung der unteren Basislinie erhalten. Die Datenbank ist frei verfügbar und wird zum Vergleich verschiedener Worterkennungssysteme genutzt. Im Zusammenhang mit der Erkennung allgemeiner Wörter ist die IFN\ENIT Datenbank jedoch aufgrund des verwendeten Vokabulars nicht ideal. Für lateinische Schrift existiert beispielsweise die RIMES-Datenbank mit einem Vokabular von 5.334 allgemeinen französischen Wörtern, anhand derer verschiedene Erkennungssysteme verglichen werden können [56].

Tabelle 2.1. Übersicht der verschiedenen Datenbanken arabischer Schriftproben. Verfügbarkeit: ● frei verfügbar $ gegen Gebühr verfügbar ◖ teilweise verfügbar ◷zeitweise verfügbar ○ nicht verfügbar @ Für Online Verfahren (Trajektorien) d Druckschrift

Name		Inhalt	Beschreibung	Ref.
IESK-arDB	●	Handschrift: Buchstaben, Wörter, historische Dokumente & Synthetische Handschrift.	Während der Arbeit an dieser Promotion entstandene Datenbank (siehe Kapitel 4).	[30, 37]
IFN\ENIT	●	26.459 handschriftliche Wörter.	Handschriftliche Proben zu 937 verschiedenen tunesischen Städtenamen.	[62]
OpenHaRT	◷	Handschriftliche Textzeilen.	Lizenz zeitlich und thematisch auf OpenHart Evaluierung beschränkt, momentan nicht verfügbar.	[7]
ALtecOnDB	$@	39.151 unterschiedliche, gebräuchliche handschriftliche arabische Wörter (online).	Datenbank mit umfangreichem Vokabular, jedoch nicht frei verfügbar (auch nicht auszugsweise, daher ist keine Einschätzung der Proben möglich).	[3]
KHATT	●	4.000 handschriftliche arabische Paragraphen und aus ihnen extrahierte Textzeilen.	Enthält rudimentäre Grundwahrheiten (digitale Transkription der Handschrift).	[44]
AMA	$	5.000 Seiten mit arabischer Handschrift.	Datenbank für Zeilensegmentierung und Erkennung arabischer Handschrift (Offline).	[10]
Arabic-Handwritten 1.0	◖	5000 Textseiten in arabischer Handschrift.	Datenbank für offline Handschrifterkennung.	
APTI	●d	45.313.600 Wörter (Druckschrift).	Sammlung arabischer Druckschrift.	[89]

ERIM	○d	750 Seiten Text.	Datensammlung arabischer Bücher und Magazine.	[84]
CEDAR	○	100 Textseiten mit je 150-200 Wörtern.	Arabischer Teil der CEDAR Datenbank, offline Handschrift.	[60]
CENPARMI	$	29.498 PAWs, 15.175 Ziffern und 2.499 Nummern.	Handgeschriebene Beträge in Bankchecks (offline).	[8]

2.2 Schriftsynthese

Aufgrund der oben genannten Einschränkungen der verfügbaren Datenbanken wird ersatzweise auch auf synthetische Schriftproben zurückgegriffen [33]. Im Kontext der automatischen Handschrifterkennung kann dabei in erster Linie zwischen zwei Ansätzen unterschieden werden:

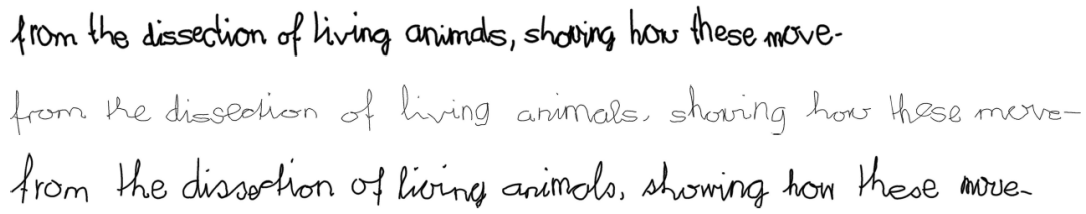
1. Methoden zur Generierung von neuen Buchstaben, Glyphen, Silben, Wörtern oder Sätzen auf Grundlage einiger bestehender Proben.

2. Methoden zur Synthese von Wörtern und Texten (für die noch keine Proben vorliegen), indem einzelnen Glyphen oder Modelle von Buchstaben zusammengefügt werden (kann 1. als Teilschritt enthalten).

Für lateinische Schrift existieren hierzu diverse Ansätze.

2.2.1 Verfahren zur Erweiterung des Probenbestandes

Ein erster Schritt der Schriftsynthese ist das Generieren einer Vielzahl synthetischer Buchstaben bzw. Glyphen anhand einiger Stichproben, was typischerweise durch die Perturbed Data Technik realisiert wird. Dabei werden bestehende Proben zufallsbasiert durch Rauschen deformiert [91]. In der Literatur finden sich jedoch auch Ansätze, die auf Modellen von Handschrift basieren und spezifischere, echter Handschrift nachempfundene Variationen erzeugen [24, 67, 79]. So beschreibt etwa das Delta Log-Normal Modell die physikalischen Vorgänge des Schreibens. Diese basieren auf einigen vom Schreiber angepeilten Zielpunkten und der Massenträgheit, aufgrund derer die Zielpunkte abhängig von neuromuskulären Prozessen nicht durch grade Linien, sondern durch Kurven verbunden werden [93, 96]. Dieser Ansatz wird nicht nur zur Generierung von synthetischen Proben, sondern auch



[92]

Abbildung 2.1. Beispiel dreier Proben eines englischen Satzes, welche mit dem in [91] und [92] beschriebenem Verfahren synthetisiert wurden.

zur Analyse von vermindertem Schreibvermögen verwendet, welches ein Symptom von neurologischen Krankheiten wie Alzheimer darstellt.

Ein anderer Ansatz lässt sich mit Active Shape Modells (ASMs) verfolgen [26]. Hierbei werden nicht die eigentlichen Schreibvorgänge simuliert, sondern vielmehr mit deren Resultaten deformierbare Modelle der einzelnen Buchstaben gebildet. Diese enthalten statistische Informationen über die Ausprägungen der Form der jeweiligen Buchstabenklasse und lassen sich auch zur Klassifizierung von Buchstaben einsetzen [71, 87, 88].

2.2.2 Verfahren zur Schriftsynthese aus Glyphen

Bei der Wortsynthese durch Verknüpfung von Glyphen (Silben, Ligaturen oder Buchstaben mit Kashida) wird ein digitaler Text in ein Bild umgewandelt, welches möglichst die gleichen Eigenschaften wie ein echtes handgeschriebenes Beispiel aufweisen soll [42, 91]. Für lateinische Schrift existiert beispielsweise ein Ansatz, bei dem zunächst neue Glyphen erzeugt werden, indem aus Bezierkurven bestehende Modelle zufällig verformt werden [92]. Diese lassen sich anschließend zu Textzeilen verknüpfen, wie sie in Abbildung 2.1 dargestellt sind. Zum Trainieren einer Zeilenerkennung wurden reale sowie die dreifache Menge an synthetischen Textzeilen verwendet. Dabei wurde bei den realen 71,8% und bei den synthetischen Zeilen 61,7% Treffergenauigkeit erzielt.

2.2.3 Ansätze zur Synthese arabischer Handschrift

Für die arabische Schrift sind nur zwei Ansätze zur Synthese von Handschrift bekannt. Ursachen hierfür liegen z.B. in der positionsabhängigen Gestalt der Buchstaben und den häufig auftretenden diakritische Zeichen, wodurch die Textsynthese zusätzlich erschwert wird. Weitere Ansätze basieren auf Druckschrift. Hier wurde der Einfluss von echten und simulierten Kopiervorgängen auf die Genauigkeit der

(a) [34] (b) [34] (c) [82] (d) [82]

Abbildung 2.2. Beispiele der Ergebnisse bestehender Ansätze zur Synthese arabischer Handschrift.

Klassifizierung sowie die Eignung von Datenbanken aus Pseudo-Wörtern untersucht [63,64] (Pseudo-Wörter sind zufällige Buchstabenpermutationen, die zumeist kein gültiges Wort bilden).

Elerian et al. geben an, das erste System zur Synthese handschriftlicher arabischer Wörter entwickelt zu haben [34]. Es wurden hierzu im Vorfeld aus der IFN/ENIT Datenbank einige Bildausschnitte von Buchstaben zweier Schreiber extrahiert. Zur Wortsynthese werden hieraus geeignete Proben selektiert und zu einem Wort zusammengefügt. Bei der Auswahl wird jeweils die beste Übereinstimmung von Winkel und Strichdicke der Verbindungsstellen benachbarter Buchstaben angestrebt, anstatt zufällige Proben zu verketten. Dieser Ansatz ermöglicht es daher, ein bis wenige realistische, nichtidentische Proben desselben Wortes zu synthetisieren. Ein Beispiel hierfür ist in Abbildung 2.2 (a) dargestellt.

Saabni et al. benutzen einen auf Online-Proben basierenden Ansatz zur Synthetisierung arabischer Handschrift. Es können so größere Mengen verschiedener Proben erzeugt werden. Diese umfassen jedoch keine vollständigen Wörter, sondern lediglich einzelne PAWs ohne diakritische Zeichen [82]. Die Resultate sind in Abbildung 2.2 (b) dargestellt.

2.2.4 Normalverteilte Zufallswerte

Die Streuungen von vielen Messungen – insbesondere solcher, die durch mehrere, unabhängige zufällige Faktoren beeinflusst werden – lassen sich exakt oder mit guter Näherung durch eine Normalverteilung beschreiben. Hierzu gehören Phänomäne der Naturwissenschaften (wie die durchschnittliche Körpergröße), aber auch wirtschafts- und ingenieurwissenschaftliche Vorgänge. Daher ist die Normalverteilung auch für die Simulation und Datensynthese von hoher Bedeutung. Mit dem Erwartungswert μ und der Standardabweichung σ hat eine Normalverteilung

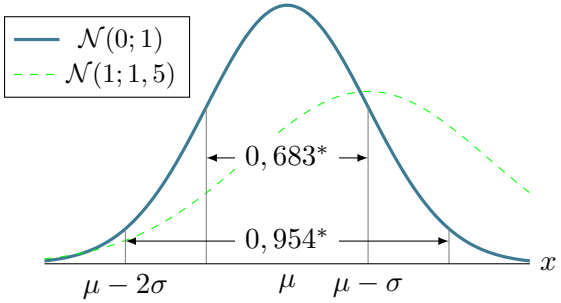


Abbildung 2.3. Darstellung der Dichtefunktion der Standardnormalverteilung und einer davon abweichenden Normalverteilung. *Anteil der durch die Dichtefunktion aufgespannten Fläche innerhalb der einfachen bzw. zweifachen Standardabweichung. Dieser entspricht der Wahrscheinlichkeit einer Zufallsvariabel x, sich innerhalb des Erwartungswertes $\mu \pm \sigma$ zu befinden.

Algorithmus 2.1 : Erzeugung eines Paares von Zufallswerten einer Normalverteilung $\mathcal{N}(\mu, \sigma^2)$.

Eingabe : Parameter μ, σ^2 (bzw. σ) der Normalverteilung[1]
Ausgabe : Zwei Zufallswerte x_1,$x_2 \in \mathcal{N}(\mu, \sigma^2)$
solange $q \notin (0,1]$ **tue**
  Erzeuge zwei zufällig Werte $u, v \in [-1,1]$ (gleichverteilt);
  Berechne $q = u^2 + v^2$;
Berechne $p = \sqrt{\frac{-2 \cdot \ln q}{q}}$;
Berechne die voneinander unabhängigen standardnormalverteilten Zufallszahlen $x_1' = u \cdot p$ und $x_2' = v \cdot p$ mit $x_1', x_2' \in \mathcal{N}(0,1)$;
Berechne x_1,$x_2 \in \mathcal{N}(\mu, \sigma^2)$: $x_i = \sigma \cdot x_i' + \mu$ [1];

[1] In den folgenden Kapiteln wird zur Definition von Normalverteilungen statt der Varianz σ^2 die Standardabweichung σ als Argument verwendet. Es ergibt sich demnach x_1,$x_2 \in \mathcal{N}(\mu, \sigma)$: $x_i = \sigma \cdot x_i' + \mu$

$\mathcal{N}(\mu, \sigma^2)$ folgende Dichtefunktion:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.1}$$

Eine grafische Darstellung der Dichtefunktion ist in Abbildung 2.3 gegeben.

In dieser Arbeit ist an verschiedenen Stellen die Berechnung von zufälligen Werten erforderlich, welche unterschiedlichen Normalverteilungen entsprechen. Zur Generierung entsprechender Zufallszahlen stehen mehrere Verfahren zur Auswahl, wobei sich die Polar-Methode aufgrund der geringen Kosten gut für die Umsetzung auf einem Computer eignet [65]. Ausgangspunkt hierfür sind jeweils zwei gleichverteilte Pseudozufallswerte u und v, welche ein Computer auf der Grundlage der aktuellen Uhrzeit berechnet (alternativ wird ein manuell definierter Saatwertes eingesetzt, um wiederholbare Zufallswerte erzeugen zu können). Das

weitere Vorgehen zur Berechnung eines Paares normalverteilter Zufallszahlen x_1,x_2 wird in Algorithmus 2.1 beschrieben.

2.3 Vorverarbeitung

Nach der Bildaufnahme oder Synthese von Handschriftproben erfolgt die Vorverarbeitung. Hier ist das vorherrschende Ziel, die Daten von Rauschen und für die folgenden Verfahren irrelevanten Informationen zu befreien. Bei Handschrift schließt dies in vielen Fällen auch die Farbinformationen sowie die Linienbreite mit ein. Diese Eigenschaften hängen von den verwendeten Schreibgeräten ab, und geben zumeist keinen Aufschluss über die dargestellten Wörter oder Zeichen. Zur Vorverarbeitung eingesetzte Methoden sind üblicherweise Gauss- und Medianfilter sowie die Normierung der Schriftneigung [77]. Neben der Binarisierung [72] spielen zudem Methoden zur Skeletonisierung und Extraktion der Kontur [51] sowie projektions- und Hough-Raum basierte Methoden zur Basislinienbestimmung [19,23,40] eine entscheidende Rolle. Geeignete Methoden für die bezüglich dieser Arbeit relevanten Vorverarbeitungsschritte werden in Kapitel 5.1 weiter ausgeführt.

2.4 Online und Offline Verfahren der Schrifterkennung

Ausgehend von der Signalaufnahme, lassen sich Systeme zur Schrifterkennung in online und offline Verfahren unterteilen.

Online Verfahren sind in erster Linie für die Handschrifterkennung von Interesse und ermitteln während des Schreibens in bestimmten Abtastintervallen die Position des Schreibgerätes. Erfasst werden hierdurch die Trajektorien des Schreibprozesses, welche eine zeitlich geordnete Abfolge der erfassten Sensorpositionen zwischen jeweils einem Ansetzen (*PenDow*n) und Absetzten (*PenUp*) des Stiftes darstellen. Je nach verwendeter Hardware wird gegebenenfalls auch der ausgeübte Druck erfasst (welcher beispielsweise wichtige Merkmale zur Verifikation von Unterschriften liefert).

Bei offline Verfahren erfolgt eine Bildaufnahme *nachdem* der Schreibprozess beendet wurde über geeignete Aufnahmesysteme (meist handelsübliche Scanner). Es wird, im Gegensatz zu online Verfahren, das optische Resultat des Schreibprozesses

digitalisiert. Daher eignet sich dieses Verfahren auch zur Bearbeitung bereits existierender, auch historischer, Dokumente und sowohl für Druck- als auch Handschrift aller Art [96]. Dynamische Bewegungsinformationen gehen jedoch verloren und die Aufnahmen enthalten zusätzlich Rauschen aufgrund des Materials und im Laufe der Zeit zunehmender Degradationen, wodurch sich die Segmentierung und Erkennung schwieriger als bei den online Verfahren gestaltet.

2.5 Zeilensegmentierung

Die Segmentierung von Absätzen in Textzeilen ist der erste wichtige Schritt einer Dokumentanalyse Pipeline [22]. Darüber hinaus treten in einigen historischen arabischen Dokumenten multiple, teilweise nachträglich am Rand eingefügte und unter Umständen schwer separierbare Absätze auf. Diese sollen in dieser Arbeit jedoch nicht behandelt werden.

Obschon das Problem der Zeilensegmentierung für Druckschrift als gelöst gilt, bestehen immer noch diverse Herausforderungen bei handschriftlichen Dokumenten [53]. Handschrift weist eine hohe Variabilität bezüglich der Abstände zwischen Wörtern, Wortlänge und Schreibstilen auf. Zudem sind die Basislinien häufig gekrümmt, und es treten auch Überlappungen zwischen benachbarten Zeilen auf. Bei arabischer Handschrift kommt noch die problematische Unterscheidung zwischen Wörtern und PAWs hinzu. Außerdem lassen sich die häufig auftretenden diakritischen Zeichen nicht eindeutig einer Zeile zuordnen, da sie sich sowohl oberhalb als auch unterhalb eines Wortes befinden können.

Im Wesentlichen existieren vier unterschiedliche Ansätze zur Zeilensegmentierung: 1. Projektionbasierte Ansätze, 2. Smearing, 3. Shredding und 4. auf lokalen Gruppierungen basierende Ansätze:

1. Projektionbasierte Ansätze berechnen die Minima der horizontalen Projektion, um die Abstände zwischen zwei Zeilen zu detektieren. Durch Nutzung der Hough Transformation können auch schräge Basislinien detektiert werden, sofern alle Basislinien einen ähnlichen Winkel und keine merkliche Krümmung aufweisen [57].

2. Zheng et al. [55] schlugen einen auf Smearing basierenden, von der verwendeten Schrift unabhängig einsetzbaren Ansatz vor. Die grundlegende Idee des Smearings ist hierbei, alle Lücken zwischen den benachbarten Wörtern einer Zeile zu schleißen, so dass die entstehenden zusammenhängenden Komponenten jeweils genau eine Zeile enthalten. Bei geringem Zeilenabstand besteht hierbei die Gefahr

der Verschmelzung mehrerer Zeilen, insbesondere wenn in einer Zeile Unterlängen und in der folgenden Zeile Oberlängen mit geringem örtlichem Versatz auftreten. Für Zeilen, deren Basislinien nicht annähernd horizontal ausgerichtet bzw. entsprechend normiert sind, ist dieser Ansatz zudem nicht geeignet.

3. Ein als Shredding bezeichneter Ansatz wird in [73] von Nicolaou et al. beschrieben. Hierbei wird der Text zunächst mit einem Unschärfefilter bearbeitet. Anschließend versuchen Tracer einen Pfad von der linken zur rechten Seite zu finden, der sich nicht mit zur Schrift gehörenden Bildpunkten kreuzt. Jeder erfolgreich gefundene Pfad separiert jeweils zwei benachbarte Zeilen. Der Ansatz wurde an englischsprachigen, handschriftlichen Dokumenten getestet und ist für arabische Schrift aufgrund der vielen diakritischen Zeichen weniger geeignet, da diese die Bestimmung eines korrekten Pfades zwischen den Zeilen erschweren.

4. Der vierte Ansatz basiert auf lokalen Gruppen, wie zusammenhängende Komponenten (Connected Components), die jeweils mit ihren benachbarten Einheiten verknüpft werden. Aufgrund der in Handschrift auftretenden Abweichungen ist die Definition eines geeigneten Maßes zur Bestimmung des Nächsten Nachbarn entscheidend [57]. So gehört der Nächste Nachbar, wenn die euklidische Distanz als Maß verwendet wird, oft zu einer anderen Textzeile. Auf lokalen Gruppen basierende Ansätze sind von der verwendeten Sprache abhängig. Im Arabischen wirken sich etwa die vielen kurzen PAWs und diakritischen Zeichen auf die Art der einzusetzenden lokalen Gruppe und deren statistische Verteilungen aus. Allerdings ermöglicht dieser Ansatz auch die Segmentierung von Zeilen mit gekrümmten Basislinien.

2.6 Analytische und holistische Worterkennung

Zur Erkennung von Wörtern lassen sich sowohl analytische als auch holistische Verfahren einsetzen. Analytische Verfahren basieren auf dem „teile und herrsche“ Prinzip. Um komplexe, aus einer Vielzahl potentieller Klassen stammende Gebilde – wie arabische Wörter – zu erkennen, erfolgt zunächst eine Segmentierung in deren Bestandteile. Aus der Erkennung der einzelnen Silben, Buchstaben oder kleinerer Einheiten wird dann auf das Wort geschlossen.

Eine zurzeit weniger verbreitete, alternative Lösung stellen die holistischen Verfahren dar, die gänzlich auf Segmentierung verzichten [5, 14]. Hier wird ein begrenztes Vokabular verwendet, da jedes Wort eine eigene Klasse darstellt. Für alle

Wörter müssen daher ausreichend Proben gesammelt werden, um Klassifikatoren, wie SVMs, zu trainieren und eine direkte Erkennung der Wörter zu ermöglichen.

2.6.1 Analytische Verfahren – Segmentierung der Wörter

Analytische Verfahren untergliedern sich wiederum in Verfahren mit expliziter und impliziter Segmentierung [21, 28]. Bei expliziter Segmentierung werden die Wörter zunächst in einzelne symbolische Einheiten (meist Buchstaben) zerlegt, um nach anschließender Klassifikation dieser Einheiten auf das vorliegende Wort zu schließen. Bei den impliziten Verfahren sind es kleinere, meist gleichgroße Einheiten.

Explizite Segmentierung. Bei der expliziten Segmentierung werden zunächst mittels topologischer Merkmale Kandidaten für Trennstellen lokalisiert, an denen zwei Buchstaben segmentiert werden können. Ein Ansatz für lateinische Handschrift nutzt etwa Minima und zusätzliche, durch ein künstliches Neuronales Netz detektierte Trennstellen [20]. Arica und Yarman-Vural schlagen in [16] einen anderer Ansatz vor. Dieser nutzt Über- und Unterlängen, welche jeweils zu einer Linie verbunden werden, die zwei benachbarte Buchstaben trennt. Bekannte Ansätze der expliziten Segmentierung arabischer Schrift nutzen nahe der Basislinie befindliche Minima als potentielle Segmentierungspunkte. Daraus resultiert eine erhöhte Übersegmentierung bei problematischen Buchstaben wie z.B. س, weshalb Wörter, welche entsprechende Buchstaben enthalten, oft aus den Testdatenbanken ausgeschlossen werden [21, 59]. Dies gilt sogar für Maschinenschrift [85].

Ein Vorteil der expliziten Segmentierung besteht darin, dass sich auftretende Probleme aufgrund der transparenten Architektur leicht lokalisieren und analysieren lassen. Zudem erlaubt sie eine modulare, sukzessive Bearbeitung von Segmentierung und Buchstabenerkennung. Daher könne neue, verbesserte Segmentierungsmethoden effektiv in ein bestehendes System integriert werden. Hierzu ist anzumerken, dass der Segmentierungsschritt bei der expliziten Segmentierung bislang wesentlich problematischer ist, als die folgende Erkennung oder die Nachverarbeitung.

Hinsichtlich Methoden, welche auftretende Segmentierungsfehler mittels a priori Wissen korrigieren, besteht für arabische Handschrifterkennung ein Forschungsrückstand. Die Zweckmäßigkeit solcher Methoden zeigt unter Anderem die von

Ray et al. eingesetzte explizite, auf multiplen Heuristiken basierende Segmentierungen für die Erkennung von gedruckten und handgeschriebenen indischen Wörtern [80]. Hierbei wird ausgehend von einem Lexikon die wahrscheinlichste Segmentierungshypothese ausgewählt, wodurch Segmentierungsfehler vermindert werden.

Implizite Segmentierung. Bei der impliziten Segmentierung erfolgt zunächst entweder eine Übersegmentierung in Buchstaben und kleinere Bestandteile oder aber eine Rasterung in gleichgroße Einheiten, wie die konstante Fenstergröße des Sliding-Window Ansatzes [6, 28, 64, 81]. Die aus diesen Segmenten extrahierten Merkmalsvektoren werden einem Klassifikator übergeben, der diese in Abhängigkeit mit den benachbarten Merkmalsvektoren auswertet. Im nächsten Schritt werden die Klassen der Einheiten zu einer Sequenz von Buchstaben zusammengefasst.

Die Anzahl an Segmenten, aus denen sich ein Buchstabe zusammensetzt, ist nicht von der Topologie abhängig. So kommt es vor, dass einfache, aber in arabischer Handschrift oft sehr langgezogene Buchstaben wie ب, durch mehr Einheiten repräsentiert werden, als komplexere Buchstaben wie س. Einheiten lassen sich zudem auch als Segmentierung (Kashida oder Leerzeichen) klassifizieren. Eine Segmentierung in Buchstaben erfolgt demnach indirekt als Nebenprodukt der Klassifizierung, weshalb eine klare Trennung dieser beiden Schritte nicht möglich ist. Auch werden mehr Trainingsdaten benötigt. Vorteile liegen in kompakten und gegebenenfalls auch schriftunabhängig realisierbaren Lösungen [18].

2.6.2 Holistische Verfahren

Holistische Verfahre benötigen keine Segmentierung und können auch schriftunabhängig implementiert werden. Sie setzen jedoch voraus, dass für jedes zu klassifizierende Wort ausreichend Stichproben zum Trainieren des Klassifikators vorhanden sind. Gute Ergebnisse konnten holistische Systeme für arabische Maschinenschrift [9] und – umfangreiche Trainingsdatenbanken vorausgesetzt – auch für Handschrift erzielen [68, 74]. Für historische Dokumente wurde beispielsweise von Lavrenko et al. ein schreiberabhängiges, holistisches Erkennungssystem entwickelt, welches die manuelle Extraktion von Proben für jedes zu klassifizierende Wort erforderlich macht [54]. Derartige Verfahren werden eingesetzt, um z.B. Historikern dabei zu unterstützen, innerhalb desselben Dokumentes nach dem erneuten

Vorkommen eines markierten Wortes zu suchen.

Holistische Verfahren sind psychologisch motiviert, da die visuelle Intelligenz eines Menschen eine holistische, kontextunterstützte Erkennung favorisiert [58, 96]. Voraussetzung hierzu ist allerdings ein umfangreiches Kontextwissen, insbesondere ein Sprachmodell, um aufgrund bereits erkannter Wörter eine Hypothese des Folgenden aufzustellen. Dies ermöglicht es, außerhalb der Fovea befindliche (und daher nur unscharf wahrnehmbare) Wörter anhand ihrer wesentlichen Merkmale zu klassifizieren. So reicht in Kombination mit a priori Informationen in vielen Fälle die Länge des Wortes, die Position von Ober- und Unterlängen und der Anfangs- oder Endbuchstabe für eine robuste Erkennung aus. Bei unbekannten bzw. selten gelesenen Wörtern oder ungewohntem Schriftbild ist es allerdings auch für den Menschen erforderlich, zunächst einzelne Buchstaben zu erkennen, um daraus auf das Wort zu schließen. Daher beschränken holistische Verfahren die Anzahl an unterscheidbaren Wörtern auf jene, für die Proben in Datenbanken existieren.

2.7 Merkmalsextraktion

Im Anschluss an die Vorverarbeitung, und eine gegebenenfalls stattgefundene Segmentierung des aufgenommenen Bildes, findet die Extraktion geeigneter Merkmale statt, mit denen in der nachfolgenden Klassifikation die Erkennung der Buchstaben realisiert wird. Bei der offline Schrifterkennung liegen die Daten, aus denen die Merkmale extrahiert werden sollen, in der Regel als Grau- oder Binärbild vor. Sofern keine Convolutional-Neural-Networks (CNNs) verwendet werden, empfiehlt es sich, aus den Bildpunkten eine kompaktere, leichter zu interpretierende Merkmalsrepräsentation zu extrahieren.

Geeignete Merkmale sind möglichst diskriminativ, d.h. Merkmale von Proben derselben Klasse sollen sich kaum, verschiedener jedoch stark unterscheiden (geringe *Intravarianz* bei hoher *Intervarianz*). Um dies zu erreichen, müssen die verwendeten Merkmale robust gegenüber den innerhalb einer Klasse auftretenden Schwankungen sein. Diese bestehen im Falle von Druckschrift aus Rauschen und diskreten Änderungen der Schriftart und -Größe, welche bereits im hohem Maße durch die Vorverarbeitung ausgeglichen werden können. Bei Handschrift hingegen sind diese Schwankungen generell stärker und lassen sich nicht so gut vorhersagen, da sie vom Schreiber und anderen Einflüssen abhängen.

Im Bereich der Schrift- hat sich, im Gegensatz zur Spracherkennung, bisher noch

kein fester Satz an Merkmalen herauskristallisiert. Daher soll hier eine Übersicht der verschiedenen Merkmalsgruppen erfolgen, die üblicherweise für die Erkennung von Schrift – insbesondere arabischer Handschrift – Verwendung finden. Diese gliedern sich in statistische und strukturelle Merkmale.

2.7.1 Statistische Merkmale

Statistische Merkmale sind Low-Level-Merkmale, die meist aus verschiedene Bildbereiche extrahiert und als Merkmalsvektoren $\mathbf{m} \in \mathbb{R}^n$ zusammengefasst werden. Typische statistische Merkmale für Schrifterkennungsprobleme sind Kantenmerkmale, die über Filter oder die 4er- bzw. 8er-Nachbarschaft extrahiert werden [47]. Weiterhin werden Merkmale aus horizontalen oder vertikalen Projektionen berechnet. Auch Momente oder momentähnliche Merkmale – die Eigenschaften aller Punkte eines Bildes bzw. des gewählten Bildausschnitts als Skalar festhalten – werden oft eingesetzt [78]. Der Vorteil von statistischen Merkmale liegt darin, dass sie sich robust und automatisch aus den Trainingsproben extrahieren lassen. Dadurch kann das Erkennungssystem durch die situationsabhängige Auswahl geeigneter Trainingsproben an ein spezielles Problem (wie einen regionalen Schreibstil) angepasst werden. Auch lassen sich verschiedene Merkmalsvektoren kombinieren oder durch unterschiedliche Klassifikatoren auswerten. Einzelne statistische Merkmale enthalten jedoch wenig Information und sind nicht sehr aussagekräftig. Zudem müssen hochdimensionale Merkmalsvektoren gegebenenfalls zunächst reduziert werden, um redundante und irrelevante Merkmale zu entfernen.

Im Kontext der Schrifterkennung werden statistische Merkmale aus Grau-, Binär- oder ausgedünnten Bildern extrahiert, die in den meisten Fällen normiert werden. Für holistische Verfahren werden die Merkmale aus einem ganzen Wort, ansonsten einzeln für jedes Segment extrahiert. Eine weitere Strategie besteht darin, das normierte Bild an der Basislinie aufzuteilen, so dass die Extraktion von Merkmalen aus Buchstaben mit Ober- bzw. Unterlängen erleichtert wird (für einzelne Buchstaben, PAWs oder sehr kurze Wörter lässt sich die Basislinie jedoch nicht zuverlässig berechnen).

Ein andere verbreitete Vorgehensweise besteht darin, das normierte Bild in einzelne Zellen aufzuteilen und für jede Zelle separate Merkmale zu berechnen [49]. Shanbehzadeh et al. beschreiben einen solchen Ansatz für die Erkennung von arabische Buchstaben [86]. Zunächst wird jedes Bild normiert, ausgedünnt und in $5x5$

Zellen eingeteilt, aus welchen jeweils 15 verschiedene Merkmale extrahiert werden. Diese werden anschließend spaltenweise zusammengefasst, um einen reduzierten, endgültigen Merkmalsvektor zu bilden. Elf Merkmale werden aus der Verteilung der Bildpunkte extrahiert, vier weitere – die lokale Konkavität beschreibende – mittels der 8er-Nachbarschaft. Ein ähnlicher Ansatz wird von Parkins und Nandi in [76] verfolgt. Allerdings erfolgt hier eine automatische Merkmalsreduktion mittels genetischer Programmierung, wobei der ursprüngliche Merkmalsvektor aus der Verteilung von vier möglichen Kanten (vertikal, horizontal, diagonal steigend und abfallend) für jede Zelle hervorgeht. Sollen ähnliche Richtungsmerkmale nicht aus ausgedünnten, sondern aus Graubildern gewonnen werden, lassen sich hierzu stattdessen Gabor-Merkmale verwenden [48].

Neben rein bildbasierten Merkmalen lässt sich auch eine aus der Kontur eines Buchstabens berechnete Chaincode-Repräsentation verwenden, aus der weitere Merkmale extrahiert werden [17,74]. Diese Repräsentation ist kompakter und geordneter als ein Bild, und ähnelt den Trajektorien von online Verfahren. Außerdem wurden für die Buchstabenerkennung zwanzig verschiedener bengalischer Maschinenschriftarten die zur Schrifterkennung ansonsten noch nicht untersuchten Curvelet Transformationen mit gutem Erfolg eingesetzt [15].

2.7.2 Strukturelle Merkmale

Strukturelle Merkmale sind High-Level-Merkmale, welche auf extrahierten, strukturellen Elementen, den sogenannten Primitiven, basieren. Geeignete Primitive sind beispielsweise Striche, Bögen, Schleifen, End- oder Verzweigungspunkte oder – im Fall des holistischen Ansatzes – Ober- und Unterlängen. Ein (heuristisches) Modell einer Klasse wird durch die Topologie [52], das Vorkommen, die Position und die Relation der dominanten Primitive definiert. Dies erfolgt in der Regel manuell, wobei bereits simple Klassen mehrere Modelle erfordern, um Variationen in der Schreibweise widerzuspiegeln. Für holistische Verfahren eignen sich prinzipiell geometrische Merkmale, wie die Orientierung eines Wortes sowie Unter- und Oberlängen [39,64]. Bei Arabischer Schrift lassen sich auch die Anzahl und Längen der PAWs sowie Anzahl und Position diakritischer Zeichen verwenden.

Manuell erstellte strukturelle Modelle für die 10 Ziffern und die zwölf Monatsnamen wurden von Feldbach [39] zur Datumsextraktion in deutschen Kirchenbüchern

Abbildung 2.4. Prototypen nach Feldbach [39] für die Ziffern 8 und 9 und den Monat Mai.

eingesetzt. Beispiele für die Ziffern 8 und 9 und den Monat Mai sind in Abbildung 2.4 dargestellt. Zur Bildung der Modelle wurden die Positionen von Strichen, Kreisen und Halbkreisen festgelegt, um verschiedene Prototypen für die einzelnen Klassen zu modellieren. Andere Ansätzen verwenden verformbare Modelle, um auftretende Varianzen handgeschriebener Buchstaben zu repräsentieren [46,79].

Für den Einsatz im Zusammenhang mit Tablets, Smartphones und ähnlichen Systemen, haben Al-Taani und Al-Haj einen Ansatz vorgeschlagen, bei dem auf online Proben basierende strukturelle Modelle isolierter arabischer Buchstaben semiautomatisch erstellt werden [11]. Die Klassifizierung basiert ausschließlich auf einem Entscheidungsbaum, dessen Blätter jeweils einzelne Klassen enthalten (außer im Fall von ذ und ز, zwischen denen nicht unterschieden werden kann). Die Hauptmerkmale sind die Anzahl der Segmente und Schleifen sowie Vorkommen, Ausrichtung und Position der scharfer Winkel (20-40°) eines Buchstabens. Entsprechend der Natur struktureller Merkmale, werden keine Abweichungen der Primitive toleriert (wie beispielsweise Rundungen statt klarer Winkel oder einem Strich statt zwei Punkten). Eine Kooperation bzw. Konditionierung des Schreibers wird daher vorausgesetzt.

Vorteile struktureller Merkmale liegen in ihrer hohen Diskriminanz und klaren Definition der Einzelmerkmale, die sich somit leichter interpretieren lassen. Da das manuelle Erstellen mehrerer, komplexer struktureller Modelle erforderlich sein kann, sind strukturelle Merkmale für Problemstellungen mit vielen Klassen generell weniger praktikabel. Dies gilt insbesondere für Probleme, deren Klassen eine hohe Intravarianz aufweisen, wie sie unter Anderem bei der schreiberunabhängigen Schrifterkennung zu erwarten sind.

2.7.3 Kombination statistischer und struktureller Merkmale

Sollen die positiven Eigenschaften statistischer und struktureller Merkmale in einem System vereint werden, ist im Zusammenhang mit arabischer Schrifterkennung die Auswahl einiger diskreter struktureller Merkmale $m_d \in M_d = \{m_1, m_2, ...m_n\}$ von Interesse. Geeignet sind bestimmte Primitive, die in möglichst allen Proben einer

Klasse vorkommen und sicher detektiert werden können. Hierzu gehören insbesondere die Anzahl von Fragmenten und Schleifen, aus denen sich ein Buchstabe einer bestimmten Klasse erwartungsgemäß zusammensetzt [99]. Diese enthalten eindeutige Eigenschaften einer Klasse, und weisen eine geringere Intravarianz und höhere Intervarianz auf, als komplexe, nur bestimmte Schreibweisen repräsentierende Modelle.

Eine Möglichkeit die beiden Merkmalsarten zu fusionieren besteht nun darin, den Zustand eines strukturellen Merkmals m_d dem statistische Merkmalsvektor $\mathbf{m}$ hinzuzufügen (entsprechend der Zelle, in welcher m_d lokalisiert wurde) [45]. Eine weitere Möglichkeit bietet sich, indem durch Auswertung von m_d mittels Entscheidungsbäumen zunächst nur die Zahl potentiellen Klassen reduziert wird und anschließend eine auf statistischen Merkmalen basierende Klassifikation erfolgt. Dies ist bei arabischer Buchstabenerkennung prinzipiell sinnvoll, da sich einige Buchstaben ausschließlich durch diakritische Zeichen unterscheiden [17].

2.8 Merkmalsselektion

Werden umfangreichen Merkmalsvektoren mit mehreren hundert Merkmalen verwendet, besteht eine hohe Wahrscheinlichkeit, dass diese redundante oder irrelevante Merkmale aufweisen. Je nach verwendetem Klassifikator ist es zur Verbesserung der Klassifikationsergebnisse (und zur Verringerung der Kosten) von Vorteil, die skalaren Merkmale zu gewichten und gegebenenfalls deren Anzahl zu reduzieren.

Wird beispielsweise ein einfacher k-NN Klassifikator eingesetzt, sollten die einzelnen Merkmale in jedem Fall gewichtet werden, damit weniger relevante Merkmale einen geringeren Einfluss auf die Klassifikationsergebnisse haben, oder gänzlich ignoriert werden. Die Gewichte lassen sich z.B. durch ein Singel-Layer-Perzeptron (SLP) trainieren. Auch der Einsatz von evolutionären Algorithmen, um eine optimale Kombination von Merkmalen zu finden, wird zur Klassifikation von Buchstaben eingesetzt [76] (hier haben die Gewichte entweder den Wert 0 oder 1).

Ein übliches, variablenorientiertes Verfahren zur Reduktion der Dimension des Merkmalsraums ist die Hauptkomponentenanalyse (PCA) [4]. Hier wird durch Linearkombination der ursprünglichen Merkmale eine geringere Zahl neuer Merkmale berechnet. Hierzu wird die Hauptkomponente des Merkmalsraums berechnet, die stärker ausgeprägt ist, je mehr die einzelnen Merkmale miteinander korrelieren. Im Idealfall geht keine Informationen verloren und Redundanzen der korrelierten

Ursprungsmerkmale werden entfernt. Falls die ursprünglichen Merkmale nicht korreliert sind, bringt der Einsatz der PCA jedoch keine Vorteile.

2.9 Klassifikation

Die Zuordnung von Proben zu einer bestimmten Klasse durch Auswertung der extrahierten Merkmale ist eine zentrale Aufgabe im Bereich der Mustererkennung. Hierfür eignen sich Klassifikationsverfahren, die – im Gegensatz zur Regression – nicht die kontinuierliche Intensität verschiedener Ausgangsmerkmale beschreiben, sondern eine Probe aufgrund extrahierter Merkmale einer konkreten Klasse zuweisen, oder entsprechende Wahrscheinlichkeiten für alle Klassen approximieren. Holistische Verfahren der Handschrifterkennung extrahieren Merkmale aus der gesamten Darstellung der Probe, wobei jedes Wort eine eigene Klasse bildet. Bei analytischen Verfahren wird ein Buchstabe oder der Teil eines Buchstabens einer der im vorherigem Kapitel aufgelisteten Buchstabenklassen zugeordnet.

Im Folgenden werden verschiedene zur Handschrifterkennung geeignete Klassifikatoren vorgestellt.

2.9.1 Generative und diskriminative Klassifikation

Die im vorangegangenem Schritt extrahierten Merkmalsrepräsentationen $\mathbf{m}$ (Beobachtungen) werden im anschließendem Klassifikationsschritt ausgewertet, um die Klasse C_i des zugehörigen Buchstabens zu bestimmen. Hierzu lassen sich sowohl generative als auch diskriminative Modelle einsetzen.

Bei generativen Modellen wird angestrebt, die multivariate Wahrscheinlichkeit $P(C_i, \mathbf{m})$ zu ermitteln. Hierdurch werden die Eigenschaften der Klassen gelernt. Bei der Klassifikation wird dann abgeschätzt, welche der Klassen die höchste Wahrscheinlichkeit aufweist, den beobachteten Merkmalsvektor $\mathbf{m}$ zu erzeugen. Wie der Name schon nahelegt, ermöglicht es dieser Ansatz zudem neue Merkmalsvektoren $\mathbf{m}$ zu den entsprechenden Klasse zu generieren. Beispiele für generative Modelle sind Gaussian Mixture Modelle (GMM), Hidden Markov Modelle (HMM) und Naive Bayes [35, 97].

Diskriminative Modelle sind ebenfalls statistische Modelle, basieren jedoch auf der bedingten Wahrscheinlichkeit $P(C_i|\mathbf{m})$. Sie kategorisieren die beobachteten Merkmale lediglich, indem die Grenzen zwischen den einzelnen Klassen modelliert

werden. Beispiele für diskriminative Modelle sind Support Vector Machines [21], Boosting und künstliche Neuronale Netze (ANN).

Diskriminative Modelle sind weniger flexibel als generative, eignen sich jedoch gut für Klassifikations- oder Regressionsaufgaben und erzielen hier oft bessere Ergebnisse [70].

2.9.2 Klassifikatoren zur Schrifterkennung

Zur Klassifizierung expliziter segmentierungsbasierter Verfahren werden SVMs, künstliche neuronale Netze, k-Nearest Neighbour Klassifikatoren [83] oder auch genetische Programmierung [13] verwendet. Bei der Verwendung statistische Merkmale sind ausreichend Trainingsproben für alle Buchstabenklassen erforderlich, um eine gute, schreiberunabhängige Klassifikation zu ermöglichen. In jedem Fall werden Stichproben zur Evaluation benötigt.

Im Bereich der impliziten Segmentierung hatten sich zunächst die ursprünglich zur Spracherkennung entwickelte Hidden Markov Modelle (HMM) etabliert. Dabei werden zunächst HMMs für alle Buchstabenklassen erzeugt, indem Merkmale aus über segmentierten Elementen durch Sliding-Windows mit der Buchstabensequenz der Grundwahrheiten verglichen werden. Anschließend lassen sich Modellen für Wörter Vokabular zusammenfügen [64]. Eine Erweiterung dieses Verfahrens wurde von Al-Hajj et. al. erforscht [68]. Zusätzlich zum horizontalen werden hier auch zwei diagonale Sliding-Windows eingesetzt, deren Ergebnisse mit künstlichen Neuronalen Netzen kombiniert werden. Die erzielten Ergebnisse weisen dabei eine geringfügige Verbesserung auf. Gegenwärtig werden statt HMMs zunehmend rekurrente neuronale Netze Long-Short-Term-Memory-Netzwerke (LSTMs) verwendet [3]. Ihre Funktionsweise ähnelt HMMs, allerdings gestalten sich Training und Initialisierung für konkrete Probleme weniger aufwendig, so dass sich bestehende Lösungen mit geringem Aufwand für andere Probleme anpassen lassen. LSTMs mit Connectionist Temporal Classification (CTC) Funktion lassen sich mit Sliding-Windows und somit als Erkennungssystem mit impliziter Segmentierung einsetzen; traditionelle LSTMs werden dagegen mit expliziter Segmentierung kombiniert. LSTMs wurden beispielsweise an der UPTI (Urdu Printed Text Images) Datenbank getestet, welche größere Mengen synthetischer Maschinenschrift-Textzeilen enthält (Urdu nutzt eine Erweiterung des arabischen Alphabets). Dabei wurde eine im Vergleich zum Stand der Technik (Shape Matching) um 5% verbesserte Erkennungsrate

erzielt [18].

Neben einfacher Klassifikation – der Rückgabe der wahrscheinlichsten Klasse C_i – lässt sich ein Ranking aller Klassen erstellen. Hierbei wird eine absteigend nach $P(C_i|\mathbf{m})$ bzw. $P(C_i, \mathbf{m})$ geordnete Liste aller Klassen ermittelt. Diese ermöglicht es anschließend, die Wahrscheinlichkeit einer Fehlklassifikation abzuschätzen, oder verschiedene Klassifikatoren zu kombinieren, um Klassifikationsfehler zu beheben.

2.10 Active Shape Models

Active Shape Modell (ASMs) wurden von Cootes mit der Absicht entwickelt, Objekte mit variierenden Größen, Ausrichtungen oder Formen in verrauschten Bildern zu lokalisieren [26]. ASMs gehören zu den statistisch verformbaren Modellen, welche die Kontur eines Objektes anhand einer festgelegten Menge signifikanter Punkte repräsentieren, die im Folgenden als *Landmarken* bezeichnet werden. Ein entscheidender Vorteil gegenüber anderen verformbaren Modellen ist dabei, dass ASMs nur statistische Abweichungen zulassen, welche den zum Training verwendeten Proben entsprechen.

ASMs lassen sich vielseitig einsetzen. Die Überprüfung industriell gefertigter Bauteile oder das Tracking von Gesichtern in einer Videosequenz stellen beispielsweise klassische Anwendung dar. Durch die Wahl der Trainingsproben lassen sich ASMs für bestimmte Probleme spezifizieren. Unterscheiden sich Proben von Gesichtern vornehmlich durch den Ausdruck, so eignen sich resultierenden ASMs für die Emotionserkennung oder verhalten sich beim Tracking robuster gegenüber auftretender Mimik. Auch lassen sich ASMs aus 3D-Datensätzen gewinnen, die unter anderem in der Medizintechnik zur Detektion von Abnormitäten in organischen Strukturen oder in der Automobilindustrie zur Oberflächeninspektion eingesetzt werden. Im Bereich der Handschrifterkennung wurden ASMs zur Klassifizierung chinesischer Schriftzeichen erfolgreich eingesetzt [87, 88]. Active Structural Models (ASSM) kombinieren statistisches und strukturelles wissen, und wurden zur Klassifikation von Zeichnungen, Insekten und bedingt zur Unterschriftverifizierung eingesetzt [12].

2.10.1 Training von Active Shape Modells

In diesem Abschnitt sollen die mathematischen Grundlagen zur Erstellung zweidimensionaler Active Shape Modells (ASMs) dargelegt werden. Erforderlich zum Training von ASMs sind m Repräsentationen einer Objektklasse der Form

$$\mathbf{x}_i = (x_1,y_1,x_2,y_2 \ldots,x_n,y_n)^T \mid i \in \{0, m\}, \tag{2.2}$$

die aus jeweils exakt n Landmarken (x,y) bestehen. Die Landmarken sollten die jeweilige Objektklasse ausreichend genau repräsentieren. Allerdings ist zu bedenken, dass mit steigendem n auch mehr Proben erforderlich sind, um statistisch verlässliche ASMs zu gewährleisten. Geeignete Landmarken müssen signifikant und in allen Proben lokalisierbar sein, wie beispielsweise Extrem- , Verzweigungs- Wende- oder Endpunkte der Kontur. Sollten es nicht möglich sein, ausreichend solcher Landmarken zu bestimmen, empfiehlt es sich nach Cootes zwischen den vorhandenen Landmarken zu interpolieren [27].

Sofern noch nicht bei der Bildaufnahme erfolgt, ist eine Normierung der Ausrichtung und Position aller $\mathbf{x}_i$ notwendig. Hiernach lässt sich der Erwartungswert $\bar{\mathbf{x}}$ und die Kovarianzmatrix $\mathbf{S}$ berechnen:

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{m} \mathbf{x}_i}{m}, \quad \mathbf{S} = \frac{1}{2m-1} \sum_{i=1}^{m} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \tag{2.3}$$

Aus der Kovarianzmatrix $\mathbf{S}$ ergeben sich die Eigenwerte λ_j und Eigenvektoren $\mathbf{e}_j$. Damit ist es möglich, jede gegebene Instanz $\mathbf{x}$ durch ein zur selben Klasse gehörendes ASM anzunähern (siehe Algorithmus 2.2). Außerdem lassen sich neue Instanzen $\mathbf{u}$ generieren:

$$\begin{gathered} \mathbf{u} = \bar{\mathbf{x}} + \mathbf{E}\mathbf{b} \\ \Big| \mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2 ..., \mathbf{e}_{m_t}), \mathbf{b} = (c_1, c_2 ..., c_{m_t})^T. \end{gathered} \tag{2.4}$$

Dabei gilt $m_t \leq m$ und $c_j \in \left[-2\sqrt{\lambda_j}, 2\sqrt{\lambda_j}\right]$. Jedes c_j definiert den Einfluss, den der Eigenvektor $\mathbf{e}_j$ auf $\mathbf{u}$ ausübt und $\sum |c_j|$ gibt an, wie stark $\mathbf{u}$ von $\bar{\mathbf{x}}$ abweicht. Dabei ist die Wahrscheinlichkeit, dass ein zufällig gewähltes $\mathbf{x}$ mit $\mathbf{u}$ korreliert, antiproportional zu $\sum |c_j|$.

Obschon es vorkommen kann, dass Eigenvektoren spezifische Klasseneigenschaften beschreiben (z.B. die Intensität von Lächeln oder die Stauchung eines bestimmten Bereiches), hängt dies von den Proben ab. Generell nimmt der Einfluss

Algorithmus 2.2 : Anpassen des ASM eines Buchstaben an eine Probe $\mathbf{x}$.

Eingabe : Probe $\mathbf{x}$, Erwartungswert $\bar{\mathbf{x}}$, Matrix der Eigenvektoren $\mathbf{E}$, Eigenwerte λ_1 bis λ_{m_t}
Ausgabe : Approximation $\mathbf{u}$
Initialisiere $\mathbf{b}$ mit 0;
Berechne $\mathbf{u} = \bar{\mathbf{x}} + \mathbf{E}\mathbf{b}$;
solange $|\mathbf{u} - \mathbf{x}| >$ *Schwellwert* **tue**
  wenn $\mathbf{u} \wedge \mathbf{x}$ *nicht normiert* **dann**
    Optimiere Position, Winkel, Skalierung und Kashidalänge von $\mathbf{u}$;
  sonst
    Optimiere Kashidalänge von $\mathbf{u}$;
  Modifiziere $\mathbf{b}$, so dass sich $|\mathbf{u} - \mathbf{x}|$ minimiert (durch Optimierungsverfahren oder Lösen der Gleichung (2.4));
  Berechne $\mathbf{u} = \bar{\mathbf{x}} + \mathbf{E}\mathbf{b}$;

der Eigenvektoren mit zunehmendem Index ab, weshalb es sich zur Senkung der Kosten empfiehlt, nur die ersten m_t Eigenvektoren in Gleichung 2.4 einfließen zu lassen. Der optimale Wert für m_t hängt jedoch von der Anzahl der Proben m, deren Varianz, der Anzahl Landmarken n sowie der gewünschten Genauigkeit der ASMs ab.

Es ist anzumerken, dass $|c_i|$ laut der Literatur $3\sqrt{\lambda_j}$ nicht überschreiten sollte [27]. Dies stellt sicher, dass $\mathbf{u}$ nicht mehr als die dreifache Standardabweichung zu $\bar{\mathbf{x}}$ aufweist, denn aufgrund der begrenzten Anzahl von Proben sind ASMs nur in der Umgebung des Erwartungswertes verlässlich. Da die Proben von organischen Strukturen oder Vorgängen stärker variieren als beispielsweise Maschinenbauteile, wird hier von $2\sqrt{\lambda_j}$ als Maximalwert ausgegangen.

Abschließend empfiehlt es sich zu überprüfen, inwiefern die ASMs die Stichproben, aus denen sie berechnet wurden, ausreichend genau repräsentieren. Hierzu werden ASMs aus jeweils $m - 1$ Proben gebildet, und hierauf an die nicht verwendete Probe $\mathbf{x}$ angepasst. Algorithmus 2.2 beschreibt dieses Vorgehen am Beispiel von ASMs von Buchstaben. Sofern sich die zu untersuchenden Proben als Vektor $\mathbf{x}$ der Länge $2n$ darstellen lassen, kann die Anpassung durch (numerische) Lösung der Gleichung (2.4) erfolgen. Andernfalls werden hierzu iterative Optimierungsverfahren eingesetzt.

2.11 Optimierungsverfahren

Die Ergebnisse vieler in der Bildverarbeitung eingesetzter Algorithmen hängen maßgeblich von einem oder mehreren Parametern oder Hyperparametern $c \in \mathbf{c} \in \mathbb{R}^n$ ab. Sofern eine repräsentative Datenbank mit geeigneten Grundwahrheiten (GT) vorliegt, lässt sich ein Algorithmus validieren und das Ergebnis in Abhängigkeit von den verwendeten Parametern als Fitnessfunktion $f_{\text{fit}}(\mathbf{c})$ bewerten. Falls nicht bekannt ist, ob die einzelnen c_i voneinander unabhängig sind, ist eine Optimierung einzelner Hyperparameter $f_{\text{fit}}(c_i)$ nicht zweckmäßig. Die Anzahl der möglichen Merkmalsvektoren $\mathbf{c}$ ist jedoch meist zu hoch, um sie alle zu überprüfen (*Brute Force*). Stattdessen kommen diverse Ansätze zur Optimierung von $\mathbf{c}$ zum Einsatz, die durch den Vergleich einiger $\mathbf{c}$ eine gute, jedoch nicht zwangsläufig bestmögliche Parametrisierung ermitteln [50].

2.11.1 Grid Search

Dieser Ansatz ist gut geeignet, um eine geringe Zahl Hyperparameter zu optimieren, wie beispielsweise C und γ von Support Vector Machines (SVMs) mit RBF-Kernel. Das Vorgehen ähnelt *Brute Force*, jedoch wird für jeden Hyperparameter ein minimaler und maximaler Wert sowie die Schrittweite festgelegt. Im Fall der Optimierung eines Klassifikators wird selbiger für alle Parametrisierungen $\mathbf{c}$ mit den Trainingsproben trainiert. Anschließend findet eine Kreuzvalidierung an Validierungsproben statt, deren bestes Ergebnis die optimale Parametrisierung indiziert. Hiernach lässt sich der Klassifikator mit optimaler Parametrisierung nochmals an Trainings- und Validierungsproben trainieren, und an Testproben evaluieren.

2.11.2 Gradientenabstiegsverfahren

Beim Gradientenabstiegsverfahren (GD) wird ein Gradient f' berechnet, um zusammen mit einer definierten Schrittweite die nächste Parametrisierung mit besserem Funktionswert $f(\mathbf{c})$ zu bestimmen [50]. Dies entspricht im zweidimensionalem Fall einer Kugel, welche einen Abhang hinunterrollt, bis sie eine Senke erreicht. Wird eine Fitnessfunktion f_{fit} verwendet, wird hingegen eine Maximierung des Funktionswertes angestrebt. Dieses Verfahren konvergiert immer in einem lokalem Optimum, welches allerdings gegebenenfalls einen schlechteren Funktionswert als das globale Optimum aufweist und/oder im Merkmalsraum relativ weit von

diesem entfernt ist.

Für GD sollte bereits ein Näherungswert bekannt sein, der sich als Initialisierung $\mathbf{c}_0$ einsetzen lässt. Bei der Optimierung der ASMs zur Klassifikation (siehe Kapitel 6) ist z.B. $\forall c_i = 0$ eine gute initiale Parametrisierung, da der Erwartungswert naturgemäß die beste Chance auf Übereinstimmung hat.

Die Berechnung der Richtung des steilsten Ab- bzw. Anstiegs erfolgt partiell:

$$f'_{\mathsf{fit}}(\mathbf{c}) = \begin{pmatrix} f'_{\mathsf{fit}}(c_0) \\ f'_{\mathsf{fit}}(c_1) \\ \vdots \\ f'_{\mathsf{fit}}(c_n) \end{pmatrix}, \; f'_{\mathsf{fit}}(c_i) = \frac{1}{2\epsilon} \left(f_{\mathsf{fit}} \begin{pmatrix} \vdots \\ c_{i-1} \\ c_i - \epsilon \\ c_{i+1} \\ \vdots \end{pmatrix} - f_{\mathsf{fit}} \begin{pmatrix} \vdots \\ c_{i-1} \\ c_i + \epsilon \\ c_{i+1} \\ \vdots \end{pmatrix} \right). \tag{2.5}$$

Bei den ersten Iterationen empfiehlt es sich, ϵ und gegebenenfalls auch die Schrittweite relativ hoch zu setzen, um lokale Minima zu ignorieren. Nachdem die maximale Anzahl an Iterationen erreicht oder eine festgelegten Fitness überschritten wurde, wird die aktuelle Parametrisierung als Optimierungsergebnis markiert.

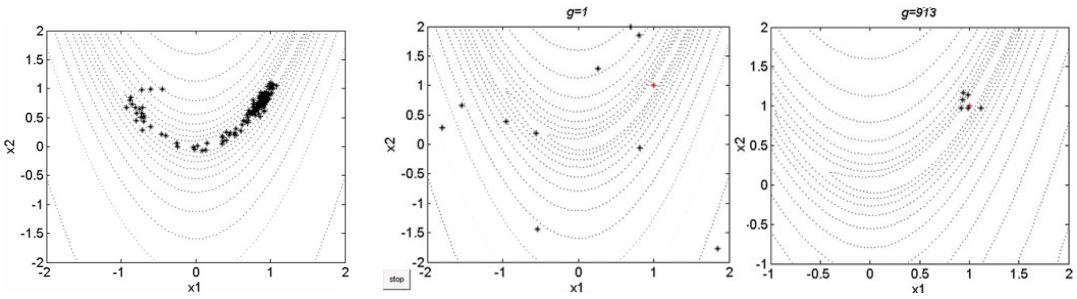
GD wird in Form von Backpropagation auch zum Trainieren künstlicher Neuronaler Netze eingesetzt.

2.11.3 Simuliertes Ausglühen

Ein alternatives, stochastisches Optimierungsverfahren ist simuliertes Ausglühen [50]. Simuliertes Ausglühen hängt stark vom Zufall ab und modelliert einen Vorgang der Thermodynamik, bei dem bei einer anfangs hohen Temperatur T eine entsprechend höhere Wahrscheinlichkeit besteht, bei der folgenden Iteration eine Parametrisierung auch dann zu wählen, wenn diese eine geringere Fitness als die aktuelle aufweist. Hierdurch lassen sich anfangs lokale Minima überwinden, was sich bei einem ungünstigem Startpunkt oder einer unruhigen Zielfunktion als Vorteil erweist. Die Optimierung erfolgt jedoch weniger stetig als bei GD.

Die Hyperparameter c der aktuellen Parametrisierung $\mathbf{c}$ werden beim simulierten Ausglühen in jeder Iteration zufällig variiert. Die neue Parametrisierung wird, abhängig von T und der Fitness $f_{\mathsf{fit}}(\mathbf{c_{i+1}})$, mit der Wahrscheinlichkeit

$$p = \begin{cases} 1 \textbf{ falls } f_{\mathsf{fit}}(\mathbf{c_{i+1}}) > f_{\mathsf{fit}}(\mathbf{c_i}) \\ e^{\frac{-\left(f_{\mathsf{fit}}(\mathbf{c_{i+1}}) - f_{\mathsf{fit}}(\mathbf{c_i})\right)}{k_B T}} \textbf{ sonst} \end{cases}, \tag{2.6}$$



(a) [50] (b) [50]

Abbildung 2.5. Visualisierung der Optimierungsergebnisse im zweidimensionalem Merkmalsraum. (a) Iterationen des simulierten Ausglühens. (b) Erste und letzte Generation einer Optimierung durch Evolutionäre Algorithmen.

angenommen (k_B ist die Boltzmann Konstante). Eine Visualisierung dieses und des nachfolgenden Verfahrens zeigt Abbildung 2.5.

2.11.4 Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) sind metaheuristische Optimierungsverfahren, die Prozesse der Evolutionstheorie nachempfinden [76] [50]. Hierzu wird zunächst der Genotyp von Individuen berechnet, welcher z.B. als Binär- oder Greycode (und somit in einer der DNA ähnlichen Repräsentation) vorliegen mag. Im Folgenden werden Genotypen der Form $\mathbf{c} \in \mathbb{R}^n$ betrachtet. Dabei stellt jedes $c_i \in [-1, 0]$ einen normierten Hyperparameter dar, um die EA internen Vorgänge zu vereinfachen.

EA ist gemein, dass sie, ausgehend von einer meist zufällig initialisierten Population POP_0, die Anzahl der Individuen iterativ durch verschiedene genetische Operatoren reduzieren und durch Rekombination eine neue Generation POP_i erzeugen. Dadurch wird der Hyperparameterraum gut abgedeckt, sodass die Wahrscheinlichkeit steigt, dass sich einige Individuen nahe am globalen Optimum befinden. Dieser Bereich wird in der folgenden Generationen gründlicher abgetastet, als Regionen mit initial schwachen Individuen. Eine auf das Wesentliche reduzierte Vorlage eines EA ist in Algorithmus 2.3 dargestellt.

Zur Rekombination kommen Verfahren wie 1- oder 2-Punkt Crossover zum Einsatz, die einen neuen Genotyp erzeugen, indem jeweils einige c_i von einem und die übrigen von einem anderen Eltern-Individium übernommen werden. Die Idee dahinter besteht darin, für gute Fitness verantwortliche Teillösungen zu kombinieren. Einige Individuen werden zudem mit einem Mutationsoperator modifiziert, um ansonsten nicht erreichbare Stellen des Hyperparameterraums abzudecken.

Algorithmus 2.3 : Grundlegendes Beispiel eines Evolutionären Algorithmus.

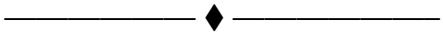
Eingabe : Funktion $f_{\mathsf{fit}}(\mathbf{c})$ (als Funktions-Pointer), Anzahl der Epochen n_{epoch}, Gewünschte Fitness t_{fit}
Ausgabe : Beste Parametrisierung $\mathbf{c}$
Initialisiere POP (Individuen mit zufälligem Genotyp $\mathbf{c}$) ;
Selektion;
solange *Anzahl der Epochen* $< n_{\mathsf{epoch}} \wedge$ *beste Fitness* $< t_{\mathsf{fit}}$ **tue**
    Rekombination;
    Mutation;
    Evaluation;
    Selektion;

Entscheidend ist, wie auch bei den anderen Optimierungsverfahren, die aufwendige Berechnungen erfordernde Fitnessfunktion $f_{\mathsf{fit}}(\mathbf{c_i})$. Diese wird vor dem Selektionsschritt aufgerufen, um alle Individuen zu bewerten. Hierzu wird etwa die *Wheel of Fortune* Methode eingesetzt, die n Individuen mit einer ihrer relativen Fitness entsprechenden Wahrscheinlichkeit (mit zurücklegen) auswählt. Hierdurch hat auch das Individuum mit der schlechtesten Fitness eine geringe Chance, in die nächste Generation übernommen zu werden. Bei der Turniermethode hingegen treten die Individuen paarweise gegeneinander an und nur dasjenige mit der jeweils höheren Fitness wird in die nächste Generation übernommen.

Durch Kombination der verschiedenen genetischen Operatoren lassen sich Evolutionäre Algorithmen für viele Probleme anpassen und bieten generell eine gute Näherungslösung. Dagegen besteht beim Gradientenabstieg die Gefahr, bereits in der ersten Iteration in einem lokalem Minima zu verharren.

2.12 Zusammenfassung

In diesem Kapitel wurden die grundlegenden Verfahren vorgestellt, die für das Verständnis der folgenden Kapitel erforderlich sind. Insbesondere wurde dabei auf für arabische Schrifterkennung geeignete Datenbanken sowie den Stand der Technik zur Schriftsynthese eingegangen. Zudem wurden gängige Verfahren zur Erkennung von Handschrift verglichen. Im nächsten Kapitel wird nun der eigene Ansatz zur Synthese von Proben arabischer Handschrift vorgestellt. Die hiermit generierten Datenbanken werden im darauffolgendem Kapitel beschrieben und mit herkömmlichen Datenbanken verglichen.



KAPITEL 3

Methode zur Synthese arabischer Handschrift

UMFANGREICHE und für unterschiedliche Anwendungsgebiete geeignete Datenbanken bilden eine essentielle Grundlage von Mustererkennungsproblemen. Sie sollten mit ausreichend Grundwahrheiten versehen sein, damit sich mit ihnen verschiedene Methoden der Bildverarbeitung trainieren sowie automatisch validieren und testen lassen. Im Fall von Schrifterkennung wird die Bereitstellung solcher Datenbanken durch die hohe Anzahl möglicher Klassen erschwert (die Anzahl verschiedener Wörter ist sogar unbegrenzt). Zudem ist es erforderlich, dass die Datenbanken und der Untersuchungsgegenstand in derselbe Sprache oder wenigstens in derselben Schrift verfasst sind. Mit über hundert Buchstabenklassen weist die Erkennung der arabischen Schrift, im Vergleich zu anderen Klassifikationsproblematiken, eine komplexe Konfusionsmatrix auf. Daher müssen umso signifikantere Merkmale aus den Buchstabenproben extrahiert werden, was jedoch aufgrund der in Handschrift auftretenden Variationen erschwert wird. Daher ist die Verfügbarkeit umfangreicher Datenbanken so entscheidend. Zudem spielt – wie am Ende des nächsten Kapitels gezeigt wird – auch die Wahl des der Datenbank zugrundeliegenden Vokabulars eine wichtige Rolle. Allerdings ist die Erstellung entsprechender Datenbanken zeit- und arbeitsaufwendig. Daher soll im Rahmen dieser Promotion das Potential der Synthese arabischer Handschrift untersucht werden. Der Anspruch, mit wenig Aufwand sehr viele unterschiedliche und zudem realistische Proben zu synthetisieren, steht dabei eben so im Mittelpunkt, wie die automatische Generierung umfassender Grundwahrheiten.

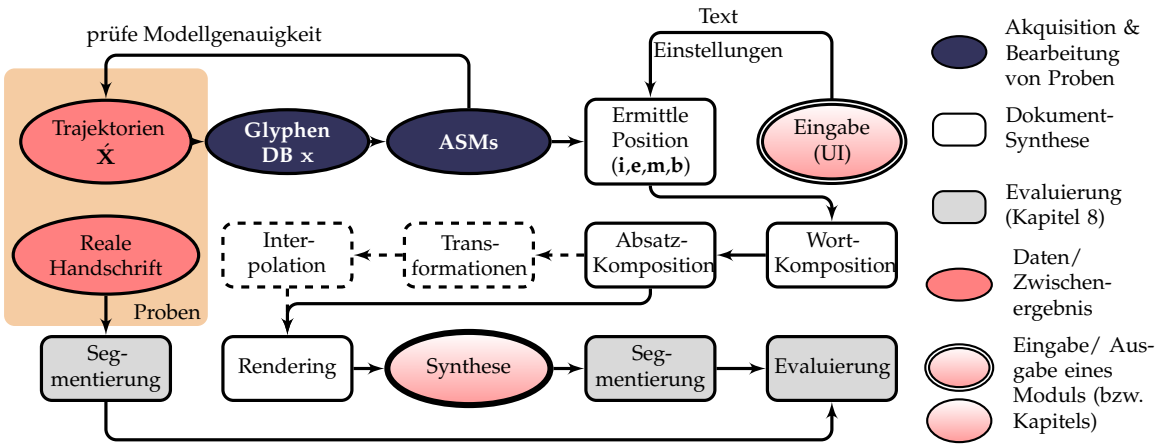


Abbildung 3.1. Flussdiagramm der in diesem Kapitel vorgestellten Methoden zur Synthese arabischer Handschrift. Zum Vergleich der synthetischen Proben mit einer realen Datenbank wurde eine Evaluierung an beiden Datensätzen vorgenommen.

Ausgangspunkt des im Folgenden vorgestellten Syntheseverfahren sind Trajektorien handschriftlicher arabischer Buchstaben, die anschließend zu Wörtern und Texten zusammengefügt und abschließend als digitales Bild oder Vektorgrafik gerendert werden. Zusätzlich können verschiedene Eigenschaften der arabischen Handschrift manipuliert und somit deren Auswirkungen auf diverse Methoden der Schrifterkennung untersucht werden. Abbildung 3.1 gibt eine Übersicht über alle wichtigen Bestandteile des Syntheseverfahrens. Die wesentlichen zugehörigen Konzepte und Ergebnisse wurden bereits auf einschlägigen internationalen Konferenzen vorgestellt [31], [29].

3.1 Glyphen-Akquisition

Die Basis des in dieser Arbeit vorgeschlagenen Verfahrens zur Schriftsynthese bilden handschriftliche Glyphen, aus denen Wörter und ganze Texte arabischer Handschrift zusammengesetzt werden. In der Topografie sind Glyphen grafische Einheiten, die Schriftzeichen wie Buchstaben, Silbenzeichen oder Ligaturen darstellen. Die hier verwendeten Glyphen bestehen jeweils aus einem einzelnen Buchstaben und – abhängig von der Position – bis zu zwei Kashidas, den Verbindungen zu dem Vorgänger und Nachfolger des Buchstaben. Um die Qualität der Synthesen zu verbessern, sollen zukünftig zudem Ligaturen akquiriert werden. Ligaturen sind Glyphen, die sich aus zwei Buchstaben zusammensetzen und eine besondere Gestalt annehmen. So ergeben beispielsweise ا und ل die Ligatur ﻻ (statt des Buchstabenpaares لَا). Aus den Glyphen werden Active Shape Modells (ASMs)

berechnet, die wiederum zur statistisch verlässlichen Berechnung neuer Glyphen eingesetzt werden. Durch Verknüpfung derselben lassen sich schließlich beliebig viele unterschiedliche Proben handschriftlicher Wörter und Texte synthetisieren.

3.1.1 IESK-arDB$_{\text{OnlineLetter}}$ – Akquisition einer Online- Buchstabendatenbank

Frühere Experimente haben gezeigt, dass offline Proben nicht optimal zur Extraktion von Glyphen geeignet sind, sofern aus diesen Active Shape Modells (ASMs) berechnet werden sollen. Zudem erschweren offline Proben diverse Nachtbearbeitungsschritte, wie etwa die Reduzierung der Länge der Kashida [31]. Weiterhin enthält die (mittels der vorgeschlagenen Segmentierungsmethode aus der Wortdatenbank IESK-arDB extrahierte) Buchstabendatenbank IESK-arDB$_{\text{Letter}}$ für viele Klassen nur wenige und für einige garkeine Proben (dies wird im nächsten Kapitel näher erläutert). Daher wurde eine separate Buchstabendatenbank konzipiert und mit Hilfe eines online Stiftes akquiriert. Dieser hat den Vorteil, dass er sich – im Gegensatz zu Stiften für Tablets – nahezu exakt wie ein gewöhnlicher Kugelschreiber benutzen lässt. Die Trajektorien werden dabei während des Schreibvorgangs mit Hilfe von Ultraschallsensoren aufgezeichnet, die sich in einem Clip befinden, welcher sich an einem Klemmbrett oder Block befestigen lässt. Der im Stift enthaltene Sender hat ein so geringes Gewicht und Volumen, dass der Schreiber dadurch nicht beeinflusst wird.

Zu Beginn des Promotionsvorhabens war nur ein Datensatz mit arabischen online Buchstaben bekannt, der 1.400 Proben von Buchstaben in isolierter Position enthält (von zehn Schreibern je fünf Proben pro Klasse) [11]. Da zur Synthese arabischer Wörter jedoch zwingend Buchstabenproben in allen vier Positionen benötigt werden, wurde eine eigene online Buchstabendatenbank erstellt. Im Rahmen der verfügbaren Mittel ließen sich Proben von vier verschiedenen Schreibern aufzeichnen. Diese dienen in dieser Arbeit zur experimentellen Abschätzung des Potentials des Handschriftsynthesesystems und sind zum Trainieren und Testen von z.B. schreiberunabhängigen Long-Short-Term-Memory Netzwerken noch nicht hinreichend (daher ist in zukünftigen Projekten eine Erweiterung der IESK-arDB$_{\text{OnlineLetter}}$ geplant).

Für jede der über hundert Buchstabenklassen wurden von jedem Schreiber mindestens fünfzig Proben aufgenommen (insgesamt 28.046), welche Rohdaten in Form

von Trajektorien $\acute{\mathbf{X}} \in \mathbb{R}^{n_\alpha \times 2}$ für jeden ausgeführten Strich enthalten ($n_\alpha \approx$ 100-200). Jeder Schreiber füllt bei diesem Prozess eine DIN A4 Seite pro Buchstabenklasse, wobei die resultierende Datei mit den Trajektorien eine Auflösung von 1000 dpi und 58 rps (reports per second) aufweist. So ergibt sich ein konstantes Zeitintervall

$$\acute{\triangle} t = t_{i+1} - t_i = 0.0172s \tag{3.1}$$

zwischen benachbarten Punkten $(\acute{x}_{i,1}, \acute{x}_{i,2}) \in \acute{\mathbf{X}}$.

Aus den Rohdaten werden automatisch Buchstabendatenbanken erstellt, so dass dem Synthesesystem zukünftig Proben von weiteren Schreibern zugänglich gemacht werden können, ohne dass manuelle Nachbearbeitungsschritte anfallen. Wesentlich ist hierfür zum einen die automatische Zuordnung der einzelnen Striche zu einer Buchstabeninstanz. Daher ist es erforderlich, dass die Schreiber immer erst den Hauptkörper und hiernach die diakritischen Zeichen und andere Bestandteile schreiben und ausreichend Platz zwischen den Buchstaben lassen. Zum anderen werden die extrahierten Daten in Form einer XML-Datei gespeichert, um schnellen Zugriff auf die Trajektorien der einzelnen Buchstabeninstanzen zu ermöglichen. Zusätzlich werden an dieser Stelle bereits Bilddateien erstellt, die zur offline Buchstabenerkennung eingesetzt werden können, denn die online Informationen der IESK-arDB$_{\mathsf{OnlineLetter}}$ werden im Folgenden Dissertation lediglich zur Berechnung der ASMs benötigt.

3.1.2 Berechnung von Active Shape Models

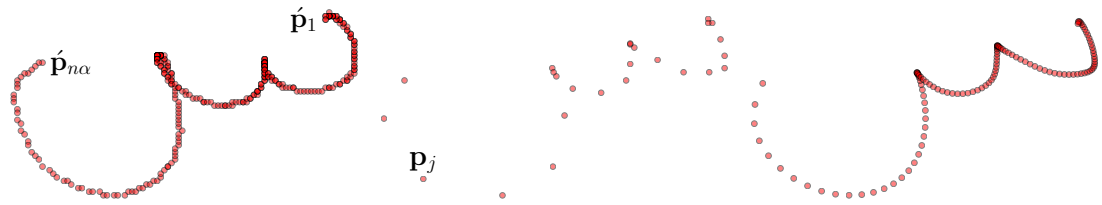
Entsprechend der obigen Betrachtungen und der Grundlagen aus Kapitel 2.10, ergibt sich zur Berechnung von Active Shape Modells (ASMs) die Notwendigkeit, die gemessenen Trajektorien $\acute{\mathbf{X}} \in \mathbb{R}^{n_\alpha \times 2}$ in normierte Vektoren $\mathbf{x} \in \mathbb{R}^{2n}$ zu konvertieren. Zur Berechnung eines ASM sind Vektoren mit einer einheitlicher Länge n erforderlich. Dabei sollte berücksichtigt werden, dass der Berechnungsaufwand $O(n^2)$ entspricht. Noch entscheidender ist jedoch, dass sich die Anzahl an Proben – die zur Berechnung von ASMs benötigt werden, die auch abseits des Erwartungswertes $\bar{\mathbf{x}}$ ASMs aussagekräftig sind – proportional zu n verhält. Daher ist n nach Möglichkeit so zu wählen, dass sowohl die Auflösung der Modelle als auch deren statistische Genauigkeit den jeweiligen Anwendungen entsprechen. In jedem Fall aber gilt $n < n_\alpha$.

Die gegebene Anzahl an Punkten n_α der Trajektorien $\acute{\mathbf{X}}$ hängt von der Zeit ab, die jeweils zum Schreiben aufgewendet wurde; die Anzahl der Landmarken

n muss hingegen für alle Proben einer Klasse identisch sein. Es konnte festgestellt werden, dass $n = 25$ eine ausreichende Genauigkeit für die Hauptkörper alle Buchstabenklassen des arabischen Alphabets bei akzeptabler Performanz sicherstellt (zuzüglich 3 Punkte für jeden Bestandteil der diakritischen Zeichen). Dabei kann es bei komplexeren Buchstaben wie س jedoch in einigen Fällen zu merklichem Informationsverlust kommen, wie Abbildung. 3.2 zeigt. Um dies zukünftig zu beheben, ließe sich eine separate Optimierung von n für jede Klasse konzipieren, z.B. auf Grundlage der auf auf Seite 129 vorgestellten Taxonomien. Andererseits ließe sich durch Einsatz von ASMs mit unterschiedlichem $n \in \{20{,}25{,}35\}$ die Varianz der zur Synthese verwendeten Glyphen weiter erhöhen, sofern dies nicht bereits in ausreichendem Maße durch eine Erweiterung der Buchstabenproben $\acute{\mathbf{X}}$ erreicht wird. Die Normierung der Trajektorien empfiehlt sich selbst dann, wenn keine ASM-basierten Proben verwendet werden sollen, da die Datensynthese und das initiale Laden der benötigten Glyphen hierdurch signifikant beschleunigt werden.

Als Ausgleich für die reduzierten Trajektorien wird gegebenenfalls eine Interpolation eingesetzt, die in Abschnitt 3.2.3 näher erläutert wird. Die Anzahl der Interpolationspunkte wird hierbei an die von der Skalierung abhängigen Auflösung und die verwendete Rendermethode angepasst, wobei bei einem Skalierungsfaktor ≤ 1 generell keine Interpolation notwendig ist. Für die Generierung einiger besonders natürlicher, ausgeprägter Wortsynthesen empfiehlt sich der Einsatz der original Buchstabenproben $\acute{\mathbf{X}}$.

Detektion von Landmarken. Entscheidend für die Güte von ASMs ist die Definition von Landmarken und deren Lokalisierung innerhalb der verwendeten Proben. Die Existenz von automatisch detektierbaren Merkmalen, die sich prinzipiell als Landmarken eignen (wie Verzweigungs- oder Endpunkte), hängt oft stark von handschriftlichen Eigenarten, der Schreibmaterialien und der Auflösung des Abbildes ab. Daher weichen diese Merkmale in realer Handschrift stark vom idealen Modell der zu erwartenden Naskh Handschrift ab. So fehlen beispielsweise einige KeyFeatures (KF) – wie Verzweigungs- (BPs) und Endpunkte (EPs) – bei Buchstaben wie س, während an anderen Stellen zusätzliche KF auftreten. Die Topologie der Buchstaben ist demnach nicht für eine automatische Bestimmung von Landmarken geeignet, denn alle Landmarken müssen zwingend in sämtlichen Proben einer



(a) $\acute{\mathbf{X}}$ (b) $\mathbf{x}$ (c) $\breve{\mathbf{x}}$

Abbildung 3.2. Beispiel für die Normierung und Nachbearbeitung der online Proben (a) Darstellung der Original Trajektorie. (b) Normierte Trajektorie (25 Koordinaten) (c) B-Spline-Interpolation der normierten Trajektorie (siehe Abschnitt 3.2.3).

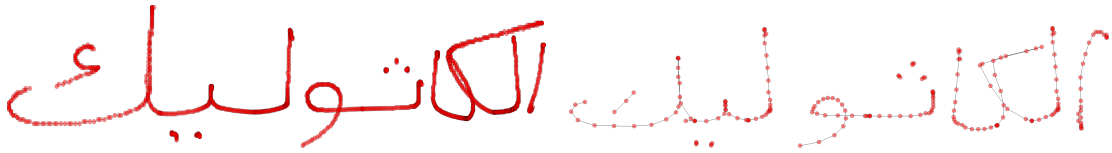
Klasse vorhanden sein und andernfalls manuell geschätzt und nachgetragen werden. Diese manuelle Nachbearbeitung für alle 28046 existierenden und sämtliche zukünftigen Proben durchzuführen, wäre allerdings im Vergleich zur Datenaufnahme unverhältnismäßig aufwändig und im Rahmen der verfügbaren Mittel kaum realisierbar. Daher wurde – entsprechend der von Cootes vorgeschlagenen Methode – zwischen den automatisch detektierbaren Landmarken interpoliert [27], um, wie oben beschrieben, insgesamt 25 Punkte zu erhalten und diese als zusätzliche Landmarken zu verwenden. Ein Beispiel hierfür zeigt Abbildung. 3.2 (b).

Sei $\acute{\mathbf{p}}_i = (\acute{x}_{2i},\acute{x}_{2i+1})$, so bestehen die automatisch detektierbaren Landmarken aus dem *PenDown* ($\acute{\mathbf{p}}_1$) und dem *PenUp* ($\acute{\mathbf{p}}_{n_\alpha}$) Punkt des Hauptkörpers sowie der *PenDown* und *PenUp* Punkte der Bestandteile der diakritischen Zeichen. Bei der Interpolation zwischen diesen Landmarken soll sichergestellt werden, dass die sich ergebenden Zeitintervalle $\triangle_t = \acute{\triangle}_t \frac{n_\alpha}{n}$ (siehe Gleichung (3.1)) zwischen zwei interpolierten Punkten $\mathbf{p}_j,\mathbf{p}_{j+1}$ konstant sind, die euklidische Distanz sich jedoch proportional zur Schreibgeschwindigkeit verhält:

$$\begin{gathered}\mathbf{p}_j = \big(1-\lambda_a\big)\acute{\mathbf{p}}_{\lfloor j\tilde{\triangle}_t\rfloor} + \lambda_a\acute{\mathbf{p}}_{\lfloor j\tilde{\triangle}_t\rfloor+1} \;\Big|\; \mathbf{p}\in\mathbf{x},\acute{\mathbf{p}}\in\acute{\mathbf{X}} \\ \text{mit } \tilde{\triangle}_t = \frac{\acute{\triangle}_t}{\triangle_t},\ \lambda_a = j\tilde{\triangle}_t - \lfloor j\tilde{\triangle}_t\rfloor,\ \lfloor j\tilde{\triangle}_t\rfloor \in [1,n_\alpha-1]\wedge j\in[1,n].\end{gathered} \tag{3.2}$$

So erhöht sich die Konzentration von Landmarken in langsam geschriebenen und somit informationsreicheren Gebieten, in denen wohl definierten Landmarken, wie Endpunkte, erwartet werden (siehe Abbildung 3.2 (b)).

Weiterhin wird die Größe und Position aller Vektoren $\mathbf{x}$ normiert. Sei W die Breite und H die Höhe des $\mathbf{x}$ umgebenden Rechtecks, so wird $\mathbf{x}$ unter Beibehaltung des Seitenverhältnisses mit dem Faktor $\frac{100^2}{WH}$ skaliert und anschließend zentriert, so dass der geometrische Mittelpunkt von $\mathbf{x}$ im Ursprung liegt. Die anschließende Berechnung der ASMs erfolgt nun entsprechend Kapitel 2.10.



(a) $\acute{\mathbf{X}}$ (b) $\mathbf{x}$

Abbildung 3.3. Beispiele eines synthetisierten Wortes aus (a) Online Proben (b) Normierten online Proben.

Generierung von Glyphen mit Active Shape Modells. Die Anzahl der Verwendeten Eigenvektoren wird im Folgenden auf $m_t = 8$ beschränkt, um so die zur Generierung von Buchstabenrepräsentationen $\mathbf{u}$ nötigen Kosten zu senken. Da für jeden Schreiber separate ASMs berechnet werden, ist die in ihnen enthaltene Varianz limitiert und die mit $m_t = 8$ erzielte Genauigkeit für den Zweck der Schriftsynthese ausreichend. Neue Trajektorien lassen sich nun wie folgt berechnen:

$$\mathbf{u} = \bar{\mathbf{x}} + \sum_{j=1}^{m_t} c_j \sqrt{2\lambda_j} \cdot \mathbf{e}_j \quad | c_j \in [0,1]. \tag{3.3}$$

Die Normierung der Faktoren c_j erleichtert später die Klassifikation mittels ASMs und genetischer Programmierung.

Die unterschiedlichen Schriftbilder aus Abbildung 3.3 machen zwar deutlich, dass die Normierung und ASM basierte Generierung noch Optimierungsbedarf aufweisen, jedoch lassen sich die resultierenden Abweichungen auch als zusätzliche Varianz einsetzen. Eine ausführliche Erläuterung der zur Verfeinerung des Schriftbilds verwendeten Interpolation wird in Abschnitt 3.2.3 gegeben.

3.1.3 Gütemaß für ASMs

Mit Hilfe der entsprechend Kapitel 2.10 für alle Buchstabenklassen berechneten ASMs, lassen sich nun nach Gleichung (3.3) neue Buchstabenrepräsentationen $\mathbf{u}$ generieren. Hierdurch wird erreicht, dass jede Bildsynthese jeweils einzigartige Buchstaben aufweist, und nicht teilweise identisch zu anderen Synthesen ist. Zudem lassen sich die Eigenschaften der Handschrift hierdurch kontrollieren, was später die Durchführung verschiedener in Kapitel 8 dokumentierter Experimente ermöglicht.

Einige Beispiele für Trainingsproben $\mathbf{x}$ und den resultierenden Glyphen $\mathbf{u}$ finden sich in Abbildung 3.4. Für den Vergleich von $\mathbf{x}$ und $\mathbf{u}$ ist die Definition eines

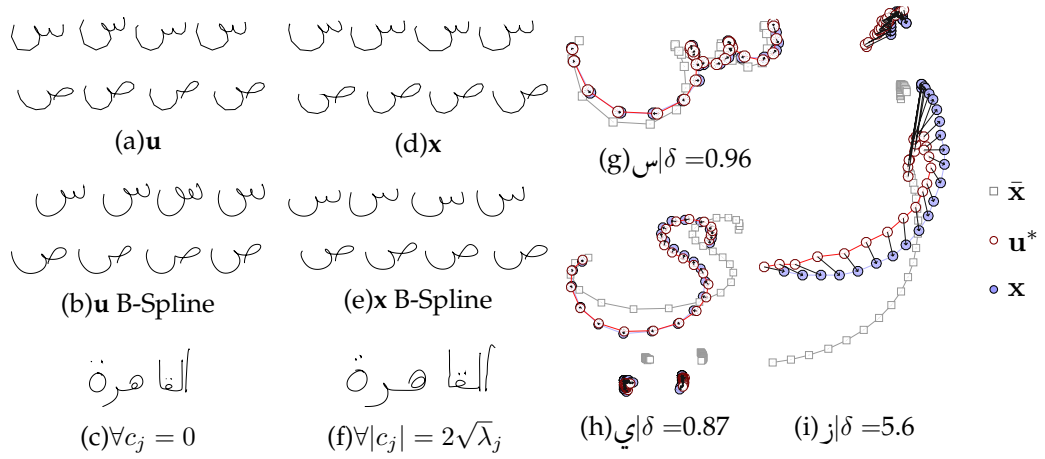


Abbildung 3.4. Beispiel verschiedener ASM Repräsentationen **u** und Original-Buchstabenproben **x**. Die Buchstaben in Abbildung b) und e) wurden mit B-Spline Interpolation geglättet. Abbildungen c) und f) zeigen Wörter, die aus ASM Glyphen **u** zusammengesetzt wurden. Beispiele für ASM Repräsentationen $\mathbf{u}^*$, welche sich optimal an Proben **x** mit der Abweichung δ anpassen, finden sich in g-i).

geeigneten Abstandsmaßes erforderlich. Aufgrund der vereinfachten Darstellung als Vektoren $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$, lässt sich der mittlere euklidische Abstand δ zwischen **x** und **u** wie folgt berechnen:

$$\delta(\mathbf{x},\mathbf{u}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(u_{2i-1}-x_{2i-1})^2+(u_{2i}-x_{2i})^2}. \tag{3.4}$$

Als Folge der durchgeführten Normierung von **x**, lässt sich $\delta(\mathbf{x},\mathbf{u})$ annähernd als Abweichung in % interpretieren. Dabei gilt, je geringer $\delta(\mathbf{x},\mathbf{u})$, desto besser die Güte des getesteten ASM. Im Fall der online Handschrifterkennung ließe sich dieses Maß auch zur Klassifikation einsetzen. In dieser Arbeit dient es jedoch lediglich dazu, die Güte der berechneten ASMs zu indizieren, da bei der später beschriebenen offline Erkennung eine normierte Abbildung eines Buchstabens statt des Vektors x vorliegt.

Für alle **x** wird die Repräsentation $\mathbf{u}^*$ mit der besten Übereinstimmung ermittelt, indem die Gleichung (2.4) numerisch gelöst wird. Die Unbekannte ist hierbei **c** mit der Initialisierung $\forall c_i = 0$. Einige Beispiele für $\mathbf{u}^*$ sind in Abbildung 3.4 (g-i) dargestellt. Im Gegensatz zu dem zu Beginn der Promotion verfolgten Ansatz, bei dem ASMs aus offline Proben gewonnenen wurden [31], repräsentieren die meisten auf online Proben basierenden ASMs – wie in Abbildung 3.4 (g) und (h) – ihre Trainingsdaten gut, auch wenn keine manuell definierte Landmarken verfügbar sind. Allerdings ist die durchschnittliche Abweichung $\bar{\delta}$ schreiberabhängig. Für Schreiber 1 ergibt sich $\bar{\delta} = 0.8$ (d. h. ca. ein Pixel Abweichung pro Landmarke). Für

Schreiber 2 $\bar{\delta} = 1.46$ und noch höhere Werte für die anderen Schreiber, deren Handschrift weniger sauber ist. Auch die Buchstabenklasse hat einen starken Einfluss auf $\bar{\delta}$. So benötigen die ASMs des Buchstaben ـج Verbesserung, wie Abbildung 3.4 (i) zeigt. Die Distanz zwischen dem Hauptkörper und dem diakritischem Zeichen variiert hier deutlich, was zu Ungenauigkeiten bei der Berechnung des ASMs und des entsprechenden $\mathbf{u}^*$ führen kann, falls $\mathbf{x}$ stark vom Erwartungswert $\bar{\mathbf{x}}$ abweicht. Dem lässt sich zukünftig durch die Verwendung von mehr Trainingsproben oder separaten ASMs für Hauptkörper und diakritische Zeichen entgegenwirken (z.B. eignen sich hierfür Active Shape Structural Models (ASSM) [12]). Allerdings sind die hier vorgestellten ASMs zur Vervielfältigung von Buchstabenproben für Handschriftsynthese gedacht, was üblicherweise durch bloßes Rauschen oder zufällige affine Transformationen realisiert wird (Perturbed Data Methode). Daher sind die auftretenden, moderaten Ungenauigkeiten akzeptabel und gefährden nicht die Qualität des Handschriftsynthesesystems.

3.2 Handschriftsynthese

Die grundlegende Idee des verwendeten Ansatzes zur arabischen Handschriftsynthese besteht darin, anhand eines gegebenen, digitalen arabischen Textes eine Sequenz kontextsensitiver polygonaler Glyphen zu selektieren und anschließend zu PAWs, Wörtern und ganzen Textseiten zu verknüpfen.

Um eine hohe Variabilität der Gestalt von Buchstaben und Wörtern sicherzustellen, werden durch ASMs generierte Glyphen $\mathbf{u}$ eingesetzt und optional durch affine Transformationen modifiziert, wie in den folgenden Abschnitten dargelegt werden soll.

3.2.1 Komposition arabischer Handschrift

Im Kontext der Bildverarbeitung – z.B. bei der Poseschätzung – werden Ansätze zur Datensynthese oft verwendet, um bereits bestehende Proben durch Methoden wie affine Transformationen zu modifizieren. Indem auf die Weise bereits umfangreiche Datenbanken durch Abwandlungen der enthaltenen Proben erweitert werden, profitieren Trainingsverfahren wie Convolutional-Neural-Networks oder Long-Short-Term-Memory Netzwerke davon, da hierdurch insbesondere das Risiko von Overfitting reduziert wird. Bei dem in dieser Arbeit vorgestellten Verfahren

wird dieser Aspekt hauptsächlich auf Buchstabenebene durch die oben beschriebene Verwendung der ASMs behandelt, auf Wort- und Textebene steht zunächst ein anderes Ziel im Fokus. Der wichtigste Syntheseschritt besteht hier vielmehr darin, Handschriften zu beliebigen Zeichenfolgen $\mathbf{w}$ zu erschaffen, für die bisher noch keinerlei Proben vorliegen. Darüber hinaus werden in Abschnitt 3.2.2 einige zusätzliche Modifikationen des Schriftzuges untersucht. Obwohl die Wortebene im Vordergrund dieser Dissertation steht, werden am Ende dieses Kapitels auch Ansätze zur Synthese von Sätzen und Textseiten untersucht.

Als Input des im Folgenden beschriebenen Schriftsynthesesystems werden Zeichenketten $\mathbf{w}$ aus arabischen Buchstaben erwartet. Dabei lässt sich die Zeichenkette direkt im User Interface des Schriftsynthesesystems eingeben (Unicode), oder aus einer UTF8 codierten Textdatei laden. Da sich die verfügbaren Glyphen zurzeit auf das reguläre arabische Alphabet und das häufig verwendete Zeichen Tamabuta (ۃ) beschränken, werden Spezialformen, wie Alif mit Hamza (أ), durch die entsprechende reguläre Form (ا) ersetzt. Zudem erfolgt eine Überprüfung auf eventuell im Text vorhandene unbekannten Zeichen – wie mathematische Symbole oder Buchstaben anderer Alphabete – die anschließend entfernt werden.

Ermittlung der positionsabhängigen Gestalt der Buchstaben. In Maschinenschrift werden Wörter auch im Arabischen explizit durch Leerzeichen getrennt, die in Uni- bzw. UTF8-Code eindeutig als ein Symbol dargestellt werden. Dies gilt allerdings nicht für die Trennung der PAWs, die sich implizit aus der Position der Buchstaben ergibt. Diese Positionen – und damit auch die positionsabhängigen Gestalten der Buchstaben – sind in digitalem arabischem Text, im Gegensatz zur expliziten Groß- und Kleinschreibung lateinischer Schriften, nicht definiert. Die konkrete Position und somit die korrekte Glyphe muss daher jeweils von dem Programm ermittelt werden, mit welchem der Text bildlich dargestellt wird. Dies muss auch das Synthesesystem leisten. Der Vorgang wird in Abbildung 3.5 an einem Beispiel verdeutlicht. Die hierzu eingesetzte Methode soll im Folgendem erörtert werden.

Sei $\mathfrak{b} \in \mathbf{B}$ ein Buchstabe der Menge aller Buchstabenklassen $\mathbf{B}$ (siehe Tabelle 1.1), so definiert k die Stelle, welche $\mathfrak{b}$ im Alphabet innehat. Weiterhin definiert $\mathfrak{q} \in \{\mathbf{i}, \mathbf{b}, \mathbf{m}, \mathbf{e}\}$ die Art der Position des Buchstabens (isoliert, am Beginn, in der Mitte oder am Ende eines PAW) und i seine Stelle innerhalb der Zeichenkette $\mathbf{w}$. Hieraus ergibt sich die Darstellung $\mathfrak{b}^k_{i\mathfrak{q}}$, wobei die Klasse der Buchstaben bzw.

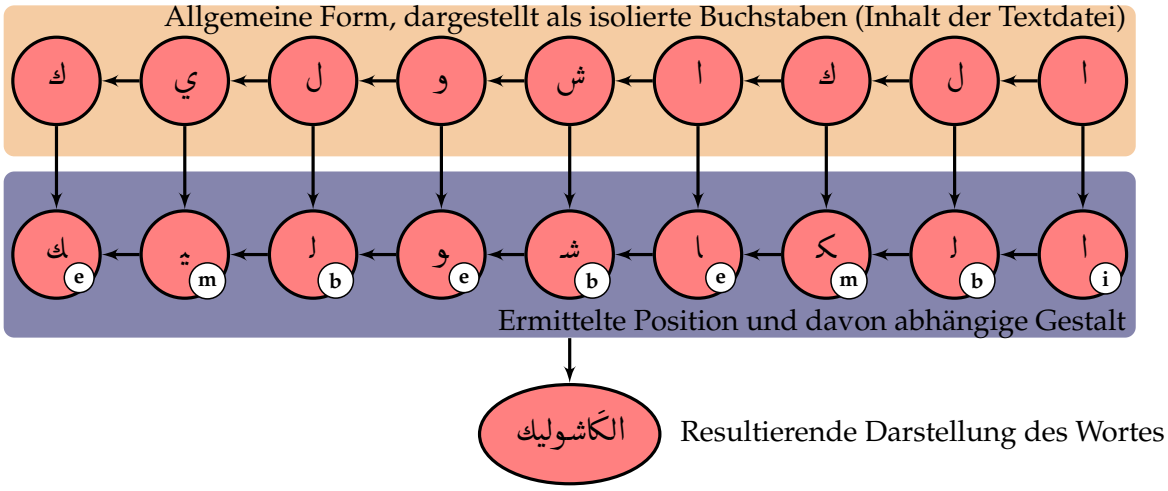


Abbildung 3.5. Die Schematische Darstellung zeigt, wie sich die Position – und somit auch die Gestalt – der Buchstaben eines arabischen Wortes bestimmen lässt.

Glyphen sowohl von $\mathfrak{q}$ als auch k abhängt (die Gesamtzahl der Buchstabenklassen, die das Systems verwendet, ist N_b). Neben den Buchstaben werden auch einige weitere Zeichen verwendet. In dieser Arbeit sind dies Leer- und Absatzzeichen, die zukünftig durch arabische Satzzeichen wie ؟ ، oder ., Ziffern oder Ligaturen erweitert werden sollen.

Obschon positionsspezifischer Unicode für arabische Buchstaben existiert, wird arabischer Text in der allgemeinen Form codiert (Unicode Block 0600–06FF). Da die Position eines arabischen Buchstabens einen starken Einfluss auf dessen Gestalt ausübt (ع عـ عـ ـع), muss diese zuerst ermittelt werden, damit sich geeignete Buchstaben bzw. ASM-Klassen für die Schriftsynthese selektieren lassen. Hierzu wird als erstes überprüft, ob ein Buchstabe $\mathfrak{b}_i$ zu den sechs Buchstaben gehört, die lediglich zwei der vier Positionen annehmen können:

$$W_{6\mathsf{F}} = \left\{ ا, د, ذ, ر, ز, و \right\}. \tag{3.5}$$

Hieraus ergibt sich bereits die Position von $\mathfrak{b}_0$, des ersten Buchstaben von $\mathbf{w}$:

$$\mathfrak{q}_0 = \begin{cases} \mathbf{i} \text{ falls } \mathfrak{b}_i \in W_{6\mathsf{F}} \\ \mathbf{b} \text{ sonst} \end{cases}. \tag{3.6}$$

Die Position der restlichen Buchstaben zu bestimmen ist etwas komplizierter. Sei $\mathfrak{q}_{i-1}$ die Position des vorherigen Buchstabens $\mathfrak{b}^k_{i-1}$ und definiere $\mathfrak{b}^k_{i+1} = \mathfrak{y}$, dass das nächste Symbol ein Leer- oder Absatzzeichen $\mathfrak{y}$ ist, so ergibt sich die Position $\mathfrak{q}_i$ des Buchstaben $\mathfrak{b}^k_i$ wie folgt:

$$\mathfrak{q}_i = \begin{cases} \mathbf{b} \iff \mathfrak{q}_{i-1} \in \{\mathbf{i}, \mathbf{e}\} \wedge \left(\mathfrak{b}^k_i \notin W_{6\mathsf{F}} \wedge \mathfrak{b}^k_{i+1} \neq \mathfrak{y}\right) \\ \mathbf{m} \iff \mathfrak{q}_{i-1} \in \{\mathbf{b}, \mathbf{m}\} \wedge \left(\mathfrak{b}^k_i \notin W_{6\mathsf{F}} \wedge \mathfrak{b}^k_{i+1} \neq \mathfrak{y}\right) \\ \mathbf{e} \iff \mathfrak{q}_{i-1} \in \{\mathbf{m}, \mathbf{b}\} \wedge \left(\mathfrak{b}^k_i \in W_{6\mathsf{F}} \vee \mathfrak{b}^k_{i+1} = \mathfrak{y}\right) \\ \mathbf{i} \text{ sonst} \end{cases}. \tag{3.7}$$

Verknüpfung der Glyphen zu Handschrift. Nachdem die Position für alle Buchstaben der Zeichenkette $\mathbf{w}$ bestimmt wurde, lässt sich für jeden Buchstaben $\mathfrak{b}_{i\text{ч}}^{k}$ ein geeignetes ASM der entsprechenden Glyphe auswählen. Die ASMs erlauben es, für alle $\mathfrak{b}_{i\text{ч}}^{k}$ und $\mathbf{w}$ einzigartige polygonal Repräsentationen $\mathbf{u}$ zu generieren und vermeiden somit die Entstehung von abschnittsweise identischen Synthesen. Alternativ werden an dieser Stelle die normierten Trainingsproben $\mathbf{x}$ oder die Originalproben $\acute{\mathbf{X}}$ verwendet. Der Einsatz letzterer verlangsamt die Handschriftsynthese jedoch erheblich und soll daher nicht weiter verfolgt werden.

Relation der Buchstaben zur Basislinie. Um die einzelnen Buchstaben sinnvoll zu Text zusammenzufügen, muss zunächst die Höhe des ersten Buchstaben eines PAW (d.h. aller Buchstaben in Position **i** oder **b**) in Relation zur Höhe der (unteren) Basislinie h_b bestimmt werden. Um diese auch für Schreibschrift authentisch wiederzugeben, wird hierbei die statistische Relation von $n(\mathrm{PAW})$, d.h. allen Buchstabenklassen, aus denen es sich das PAW zusammensetzt, mit einbezogen. Hierzu wurden zuvor für jede der Buchstabenklassen $\mathfrak{b}_{i\text{ч}}$ der relative Durchschnittswert und die Standardabweichung der Distanz $\delta_b = h_b - y_{\mathsf{min}}$ berechnet, die zwischen dem niedrigsten Punkt des Buchstaben und h_b ermittelt wurde

$$\mu_r = \frac{1}{N_k}\sum_{b=1}^{N_K}\frac{\delta_b}{H_{\mathsf{let}b}}, \sigma_r = \sqrt{\frac{1}{N_K}\sum_{b=1}^{N_K}\left(\frac{\delta_b}{H_{\mathsf{let}b}} - \mu_r\right)^2}, \tag{3.8}$$

wobei H_{let} die Höhe des Buchstaben und N_k die Anzahl von Proben der jeweiligen Buchstabenklasse angibt. Zur Berechnung wurde die IESK-arDB [37] verwendet, da die Grundwahrheiten der anderen verfügbaren Datenbanken nicht ausreichend sind. Aus diesen Angaben wird bei einer Synthese zunächst für jede verwendete Glyphe $\acute{\mathbf{U}}$ durch eine Normalverteilung $\mathcal{N}(\mu_{r_i}, \sigma_{r_i})$ und die Höhe $H_{\mathsf{let}i}$ der Glyphe ein konkreter Wert für die Entfernung zur Basislinie berechnet (siehe Abbildung 3.6). Über alle Glyphen des aktuellen PAW ergibt sich der gemittelten Wert

$$\delta_{b\mathsf{PAW}} = \frac{\sum_{i=o}^{o+n(\mathrm{PAW})}\mathcal{N}(\mu_{r_i}, \sigma_{r_i})H_{\mathsf{let}i}}{n(\mathrm{PAW})}. \tag{3.9}$$

Dabei ist o jeweils der Index der ersten Glyphe des betrachteten PAW. Zusammen mit der Matrixrepräsentation von $\mathbf{u}$

$$\acute{\mathbf{U}} = \begin{bmatrix} \acute{u}_{1,1} = u_1 & \acute{u}_{1,2} = u_2 \\ \acute{u}_{2,1} = u_3 & \acute{u}_{2,2} = u_4 \\ \vdots & \vdots \\ \acute{u}_{n,1} = u_{2n-1} & \acute{u}_{n,2} = u_{2n} \end{bmatrix}, \ \acute{\mathbf{u}}_i = \begin{pmatrix} u_{i,1} \\ u_{i,2} \end{pmatrix}, \tag{3.10}$$

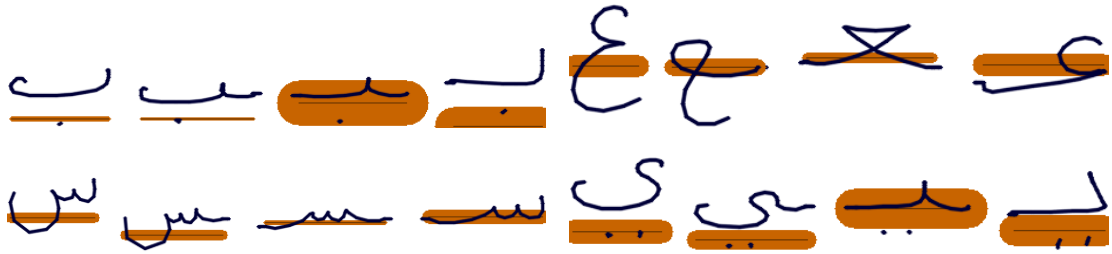


Abbildung 3.6. Es werden einige Beispiele für den mittels der IESK-arDB$_{\text{Letter}}$ berechneten und auf die Glyphen aus IESK-arDB$_{\text{OnlineLetter}}$ übertragenen Bereich dargestellt, in welchem jeweils die Höhe der Basislinie erwartet wird. Dieser Bereich ergibt sich aus der Höhe der jeweiligen zu IESK-arDB$_{\text{OnlineLetter}}$ bzw. des resultierenden ASM gehörenden Glyphe und – entsprechend Gleichung (3.8) – der aus IESK-arDB$_{\text{Letter}}$ berechneten Werte μ_r und σ_r.

und einer als Matrix $\mathbf{T}$ dargestellten Transformation $\mathbf{T}$, ergibt sich die Translation von $\acute{\mathbf{U}}$. Hierzu wird jede Zeile aus $\acute{\mathbf{U}}$ in homogenen Koordinaten mit der Translationsmatrix $\mathbf{T}_t$ multipliziert:

$$\mathbf{T}_t(\mathbf{d}) \cdot \acute{\mathbf{U}} = \mathbf{T}_t(dx,dy) \cdot \acute{\mathbf{U}} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \acute{u}_{i,1} \\ \acute{u}_{i,2} \\ 1 \end{bmatrix} \quad |\forall i \in \{1,2,...n\}. \tag{3.11}$$

Hiermit lässt sich die Position der ersten Glyphe $\acute{\mathbf{U}}^o$ des betrachteten PAW anpassen:

$$\acute{\mathbf{U}}^o := \mathbf{T}_{\text{PAW}_o} \acute{\mathbf{U}}^o \quad \left| \mathbf{T}_{\text{PAW}_o} = \mathbf{T}_t\Big(0, \delta_{b\text{PAW}} - h_b\Big). \right. \tag{3.12}$$

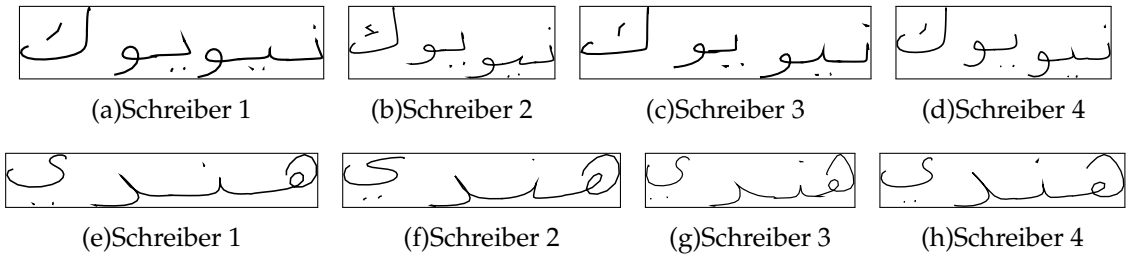
Soll die in Handschrift auftretende Variation bei der Synthese vernachlässigt werden, ist σ_{r_i} aus Gleichung 3.9 auf 0 zu setzten.

Verknüpfung von Buchstaben zu PAWs. Als nächstes sollen die restlichen $n(\text{PAW}) - 1$ Buchstaben des PAW zusammengefügt werden: عر ← ع ر, نيو ← ن ي و. Sei $\acute{\mathbf{U}} \neq \acute{\mathbf{U}}^o$ die aktuelle Glyphe des betrachteten PAW und $\acute{\mathbf{U}}^{-1}$ dessen Vorgänger, so lässt sich dies erreichen, indem $\acute{\mathbf{U}}$ mit der Matrix

$$\mathbf{T}_{t-\text{join}} = \mathbf{T}_t(\acute{\mathbf{u}}_n^{-1} - \acute{\mathbf{u}}_1), \tag{3.13}$$

multipliziert wird. Somit fällt jeweils der Anfangspunkt (*PenDown*) einer Glyphe mit dem Endpunkt (*PenUp*) seines Vorgängers zusammen. Beispiele von Wörtern, die aus Glyphen der Erwartungswerte $\bar{\mathbf{x}}$ verschiedener Schreiber zusammengesetzt wurden, finden sich in Abbildung 3.7. Um derartige Synthesen zu erzielen, müssen nur noch die finalen Positionen der einzelnen PAWs definiert werden.

Positionierung der PAWs eines Wortes. Der Abstand eines PAW zu seinem Vorgänger wird so gesetzt, dass zwar gegebenenfalls Überlappungen der umgebenden Rechtecke der PAWs, jedoch keine Intersektionen ihrer Trajektorien auftreten.



(a)Schreiber 1 (b)Schreiber 2 (c)Schreiber 3 (d)Schreiber 4

(e)Schreiber 1 (f)Schreiber 2 (g)Schreiber 3 (h)Schreiber 4

Abbildung 3.7. Beispiele für die Synthese des Wortes نيويورك (New York) und هنري mit den Glyphen unterschiedlicher Schreiber. Als Glyphen wurden jeweils die Erwartungswerte $\overline{x}$ der ASMs verwendet. Dies verdeutlicht die schreiberabhängigen Unterschiede der Handschrift, welche je nach Buchstabenklasse unterschiedlich stark ausgeprägt sind.

Sei w_P die Distanz der äußersten linken x-Koordinate des Vorgängers zur äußersten rechten x-Koordinate des aktuellen PAW, so lässt sich die Position der Glyphen des aktuellen PAWs mit folgender Translation anpassen:

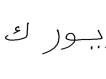
$$\mathbf{T}_{\mathsf{PAW-shift}} = \mathbf{T}_t(w_P - \varsigma_P, 0). \tag{3.14}$$

Durch den Eingabeparameter ς_P lässt sich die Distanz manipulieren, wobei sich mit $\varsigma_P < 0$ horizontal überlappende PAWs erzielen lassen, wie sie in arabischer Handschrift häufig vorkommen. Falls sich die umgebenden Rechtecke der benachbarten PAWs überschneiden, wird überprüft, ob sich auch deren Trajektorien überschneiden (siehe Abschnitt 3.3). Ist dies der Fall, so wird ς_P sukzessive um $1/4|\varsigma_P|$ erhöht, bis keine Überschneidung mehr auftritt.

3.2.2 Modulation der Wortausprägung

Active Shape Models enthalten bereits viele Variationen der Handschrift, beispielsweise der Stauchung bzw. Streckung bestimmter Buchstaben oder der Kashidas. Die Variationen entsprechen jedoch denen der zum Berechnen der ASMs verwendeten Proben und interpolieren lediglich zwischen deren Ausprägungen. Da die Proben der IESK-arDB$_{\mathsf{OnlineLetter}}$ sorgsam und nur von wenigen Schreibern geschrieben wurden, erscheint der Einsatz einer Methode zur Verfremdung der Glyphen Ú sinnvoll. Um hiermit das Spektrum der möglichen Variationen erweitern und gleichsam kontrollieren zu können, werden im Folgenden affine Transformationen eingesetzt. Diese beinhalten die bereits beschriebene Translationen zur Kontrolle der Zeilenabstände, Höhe und horizontaler Abstände der PAWs. Verwendet werden außerdem Rotationen zur Kontrolle der Abweichung von der Basislinie, Skalierungen der Buchstaben zur Stauchung, Streckung und Manipulation der Größe von

Tabelle 3.1. Beispiele für verschiedene, durch affine Transformationen und Manipulation der Trajektorien erzielbare Variationen, die sich über die Benutzeroberfläche steuern lassen.

Stauchung	Neigung	Größe	Schrägstellung	Position	Kashida
					
					
					

Buchstaben, sowie Scherungen zur Simulation der Schriftneigung.

Über die Benutzeroberfläche des Synthesesystems lassen sich zu jeder affinen Transformation die gewünschten Erwartungswerte und Standardabweichungen der zugehörigen Normalverteilungen einstellen. Hierdurch lassen sich auch globale Eigenschaften – wie die Schriftneigung eines Wortes – simulieren, die nicht durch ASMs repräsentiert werden. Die Parameter müssen jedoch innerhalb bestimmter, im Folgenden dargelegter Grenzwerte liegen. Dies soll die Bedienung der Benutzeroberfläche vereinfachen und entarteten Synthesen vorbeugen.

Entsprechend der bereits verwendeten Transformationsmatrix $\mathbf{T}_t$ für die Translation, werden weitere Matrizen für die angewendeten affinen Transformationen eingeführt. Der Einfluss der beschriebenen Transformationen auf das Syntheseergebnis ist in Tabelle 3.1 dargestellt.

Stauchung und Größe der Buchstaben. Stauchungen oder Dehnungen der Glyphen lassen sich durch Skalierung der x-Ebene realisieren. Mit der Transformationsmatrix der Skalierung

$$\mathbf{T}_s(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.15}$$

und der vom Anwender steuerbaren Normalverteilung $\mathcal{N}(\varsigma_{s_1\mu}, \varsigma_{s_1\sigma})$ – mit $\varsigma_{s_1\mu} \in [0{,}5; 2]$ und $\varsigma_{s_1\sigma} \in [0; 1]$ – ergibt sich die gewünschte Transformation eines Buchstaben durch

$$\mathbf{T}_s\big(\mathcal{N}(\varsigma_{s_1\mu}, \varsigma_{s_1\sigma}),0\big). \tag{3.16}$$

Durch eine gleichförmige Skalierung

$$\mathbf{T}_s\big(s_{\text{scale}}, s_{\text{scale}}\big) \,\big|s_{\text{scale}} = \mathcal{N}\left(\varsigma_{s_2\mu}, \varsigma_{s_2\sigma}\right), \tag{3.17}$$

mit $\varsigma_{s_2\mu} \in [0{,}1; 10]$, lässt sich sowohl eine Variation in der Buchstabengröße erzielen, als auch – mit $\varsigma_{s_2\sigma} = 0$ – die Auflösung der synthetisierten Bilder kontrollieren. Bei $\varsigma_{s_2} = 1$ beansprucht ein Buchstabe im Bild durchschnittlich eine Fläche von $2500pt^2$.

Position. Die Position der PAWs, und somit deren Abstand, lässt sich, wie im letzten Abschnitt beschrieben, mit dem Parameter ς_P kontrollieren.

Neigung. Da die Schriftneigung arabischer Handschrift (insbesondere bei Wörtern mit vielen Ober und Unterlängen) die Segmentierung und auch Klassifikation deutlich erschwert, wird sie bei der Vorverarbeitung oft normiert. Dies erfordert jedoch einen hohen Berechnungsaufwand und lässt sich bei kurzen Wörtern zudem nicht zuverlässig umsetzen. Umso wichtiger ist es daher, diesen Parameter, und die Bedingungen, unter denen eine Normierung gegebenenfalls vernachlässigt werden kann, genau zu untersuchen. Die Handschriftsynthese ermöglicht dies, durch gezielte Definition des gewünschten Neigungswinkels ς_α der Schrift mittels Scherungstransformation:

$$\mathbf{T}_{\mathsf{sch}} = \begin{bmatrix} 1 & \mathcal{N}(\varsigma_{\alpha\mu}, \varsigma_{\alpha\sigma}) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad |\varsigma_{\alpha\mu} \in [-45°, 45°]. \tag{3.18}$$

Da die Ausprägung der Schriftneigung für gewöhnlich schreiberabhängig ist, und daher innerhalb von einem Wort oder einer Zeile nicht stark variiert, ist es ratsam $\varsigma_{\alpha\sigma}$ niedrig zu halten.

Um eine authentische Schriftneigung zu simulieren, wurde zum Vergleich die IESK-arDB [37] analysiert. Dabei hat sich ergeben, dass die gemessene Schriftneigung sich gut durch eine Normalverteilung $\mathcal{N}(-3{,}8°; 7{,}1°)$ approximieren lässt (besteht den $\chi^2 - Test$ mit $\alpha = 0{,}05$, siehe Kapitel 5.1).

Rotation. Auch die Normierung der Basislinie ist ein wichtiger Vorverarbeitungsschritt, da einige Buchstabenklassen durch Rotation starke Ähnlichkeit mit anderen annehmen. Daher ist eine Normierung hinsichtlich Buchstabenklassifikation sinnvoller, als der Einsatz rotationsinvarianter Merkmale. Geringe Abweichungen von wenigen Grad sind jedoch auch nach einer solchen Normierung zu erwarten. Zudem sind die hierfür einsetzbaren Methoden – wie schon bei der Schriftneigung – im Falle kurzer Wörter sehr unzuverlässig. Deswegen wird diese Normierung in Kapitel 5 - 8 unterdrückt, falls die geschätzten Anzahl an Buchstaben weniger als vier

beträgt. Aus diesem Grund soll auch der Rotationswinkel der Buchstaben gezielt durch den Parameter ς_β gesteuert werden, um seinen Einfluss auf Segmentierung und Erkennung zu erforschen und den Toleranzbereich zu ermitteln.

Die Rotationsmatrix ist durch

$$\mathbf{T}_r(\Theta) = \begin{bmatrix} cos(\Theta) & -sin(\Theta) & 0 \\ sin(\Theta) & cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

gegeben. Hiermit lässt sich der Winkel $\mathcal{N}(\varsigma_{\beta\mu}, \varsigma_{\beta\sigma})$ anpassen, den die ideale (hier horizontale) und die lokale Basislinie einer – in endender oder mittiger Position befindlichen – Glyphe $\acute{\mathbf{U}}$ aufspannen. Dies geschieht durch eine Rotation um den *PenDow*n-Punkt $\acute{\mathbf{u}}_1$:

$$\mathbf{T}_{\dot{r}}(\Theta, \acute{\mathbf{u}}_1) = \mathbf{T}_t(\acute{\mathbf{u}}_1) \cdot \mathbf{T}_r(\Theta) \cdot \mathbf{T}_t(-\acute{\mathbf{u}}_1). \tag{3.20}$$

Die Anpassung der Glyphe erfolgt nun durch $\acute{\mathbf{U}} := \mathbf{T}_{\dot{r}}\big(\mathcal{N}(\varsigma_{\beta\mu}, \varsigma_{\beta\sigma}), \acute{\mathbf{u}}_1\big)\acute{\mathbf{U}}$. Dabei gilt $\varsigma_{\beta\mu} \in [-45°, +45°]$. Eine Analyse der IESK-arDB ergab, dass sich der tatsächliche Winkel der Basislinie grob mit der Normalverteilung $\mathcal{N}(-4.6°,14.4°)$ approximieren lässt.

Länge der Kashidas. Zusätzlich zu den auf affinen Transformationen basierenden Modifikationen, wird die Option geboten, die Länge der *Kashidas* zu manipulieren. Die Länge der Kashidas stellt eine wesentliche Eigenschaft arabischer Handschrift dar, denn sie variiert bisweilen so stark, dass Buchstabenpaaren nicht neben-, sondern übereinander geschrieben werden. Diese Schreibweise tritt oft bei bestimmten Buchstabenpaaren auf, die in Tabelle 3.2 zusammengefasst sind. Der rechte Buchstabe nimmt dabei die beginnende, der linke die endende oder mittige Form ein.

Da sich der Trennpunkt, an welchem eine Glyphe in ein Kashida übergeht, nicht exakt bestimmen lässt, wird zur Kürzung, je nach Buchstabenposition, bis zu $1/4$ der Trajektorie $\acute{\mathbf{U}}$ entfernt. Im Falle der endenden Position wird hiermit bei $\acute{\mathbf{u}}_1$, im Falle der beginnenden bei $\acute{\mathbf{u}}_n$ begonnen. Bei der mittigen Position werden entsprechend beide Kashidas gekürzt. Wie Abbildung 3.8 zeigt, lassen sich durch dieses Vorgehen auch unter Verwendung derselben Glyphen unterschiedliche Schriftbilder erzeugen.

Tabelle 3.2. Paare aller Buchstaben (Spalten: rechter B., Zeile: linker B.), die gemäß der Analyse der IESK-arDB oft als Ligaturen (übereinander) geschrieben werden.

	ب	ت	ث	ف	ل	م	ن	ه
ج	-	(تج تجـ)	-	-	(لج لجـ)	(مج مجـ)	(نج نجـ)	-
ح	-	(تح تحـ)	-	-	(لح لحـ)	(مح محـ)	(نح نحـ)	(هح)
خ	-	(تخ تخـ)	-	-	(لخ لخـ)	(مخ مخـ)	(نخ نخـ)	-
م	-	(تم)	(ثم)	-	(لم)	-	(نم)	(هم)
ه	-	(ته تهـ)	-	-	(له لهـ)	(مه مهـ)	(نه نهـ)	-
ي	(بي)	(تي)	(ثي)	(في)	(لي)	-	(ني)	(هي)

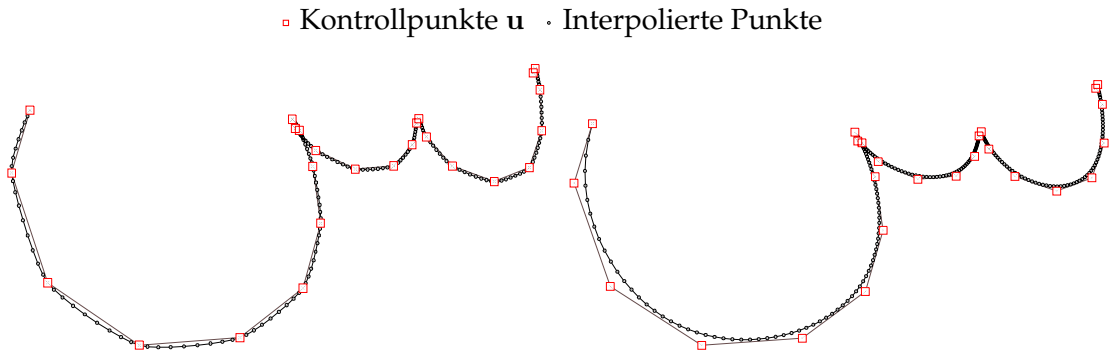
(a) (b) (c)

Abbildung 3.8. Darstellung eines synthetisierten Buchstabenpaares mit a) maximalem Kashida, b) reduziertem Kashida und c) negativem Kashida.

3.2.3 Interpolation – Optimierung von Ú

Die polygonale, unorganische Struktur der Buchstabenrepräsentationen Ú tritt mit dem Skalierungsfaktor zunehmend hervor. In extremen Fällen führt dies letztlich zu einer negativen Beeinflussung der durch das Synthesesystem generierten Proben auf die Segmentierung und Klassifikation. Daher empfiehlt es sich, Ú zu interpolieren, wobei die Koordinaten $\mathbf{ú}_i$ als Kontrollpunkte der Interpolationskurve $\breve{\mathbf{U}}$ dienen.

Sei $n(\mathbf{Ú})$ die Anzahl der Koordinatenpaare von Ú, so wird generell eine Interpolation mit $n(\breve{\mathbf{U}}) \leq n(\acute{\mathbf{X}})$ angestrebt, um eine den ursprünglichen Trajektorien ähnliche maximale Auflösung zu erzielen. Da diese Erhöhung des Datenaufkommens erst vor dem Renderungsprozess einsetzt, sind die Kosten der Datensynthese in diesem Fall – verglichen mit der Nutzung der ursprünglichen Proben $\acute{\mathbf{X}}$ – immer noch niedrig. Im Durchschnitt beträgt $\bar{n}(\acute{\mathbf{X}}) = 266{,}8$. Daher werden als Richtwert zehn Interpolationspunkte zwischen zwei Stützpunkten angesetzt (dies führt zu



(a)kubisch Hermite-Spline (b)B-Spline

Abbildung 3.9. Vergleich zweier Interpolationsmethoden zur Glättung von hochauflösenden Synthesen. Abbildung (a) zeigt stückweise kubischer Hermite- und Abbildung (b) B-Spline-Interpolation.

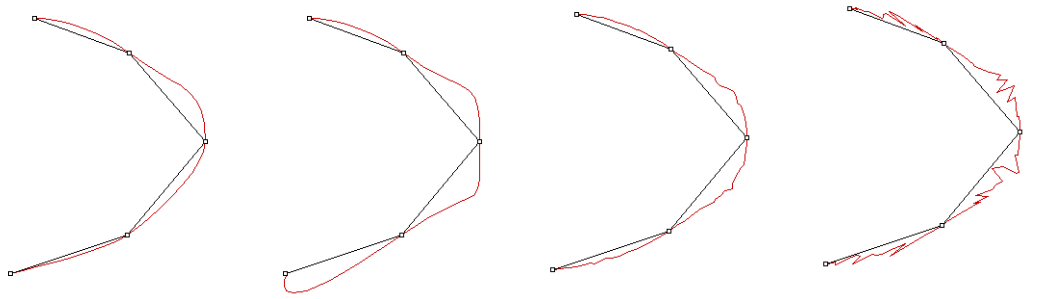
(a) (b)kubisch Hermite-Spline (c)B-Spline

Abbildung 3.10. Synthetisierte Trajektorien (نفس) (a) ohne Interpolation, (b) mit stückweise kubischer Hermite-Interpolation und (c) mit B-Spline Interpolation ($n(\acute{\mathbf{U}}) = 250$).

$n(\breve{\mathbf{U}}) = 250$). Um die Effizienz der zuvor beschriebenen Methoden sicherzustellen, sollte die Interpolation im Falle niedrig aufgelöster Bilder übersprungen bzw. die Anzahl der Interpolationspunkte über die Benutzeroberfläche angepasst werden.

B-Spline und kubisch Hermite Interpolation. Bei der Datensynthese werden in dieser Arbeit zwei verschiedene Interpolationsmethoden angewendet. Die C^1-stetigen, in Abbildung 3.9(a) und Abbildung 3.10(b) dargestellten, kubisch Hermiten Splines verlaufen zwar durch die Stützpunkte $\acute{\mathbf{u}}_i$, erzielen jedoch keine saubere, organischen Kurvatur. Zu Anfang des Promotionsvorhabens wurde dieses Interpolationsverfahren im Sinne des *Perturbed Data* Ansatzes modifiziert, um durch Verfremdung zusätzliche Buchstabenproben, wie in Abbildung 3.11, zu erhalten. Um weitere Glyphen mit geringerem Rauschen zu erstellen, lässt sich eine Fusion aus kubisch Hermiter mit Langrange Interpolation einsetzen [31]. In der aktuellen Version des Synthesesystems wird stattdessen B-Spline Interpolation eingesetzt.

Die C^2-stetigen B-Splines entsprechen dem Stand der Technik und werden unter



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

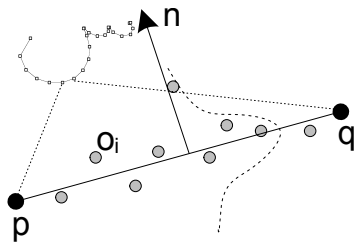
Abbildung 3.11. Modifizierte kubisch Hermite Interpolation zur Generierung von Rauschen mit (a) schwach und (b) stark ausgeprägter niedriger sowie (c) schwach und (d) stark ausgeprägter hoher Frequenz.

anderem in CAD-Anwendungen eingesetzt. Diese Interpolationsmethode erlaubt es, glatte, natürlich wirkende Kurven, wie in Abbildung 3.9(b) und Abbildung 3.10(c), zu erzeugen. Obwohl die Interpolationskurve nicht durch die inneren Stützpunkte verläuft, kommt sie diesen dennoch sehr nahe. Mit dem Maß δ aus Gleichung (3.4) lässt sich dies beziffern, indem zu jedem Stützpunkt aus $\acute{\mathbf{U}}$ die minimale euklidische Distanz zu $\breve{\mathbf{U}}$ berechnet wird. Eine Kreuzvalidierung mit $kfold = 10$ ergibt für $n(\acute{\mathbf{U}}) = 25$ eine durchschnittliche Abweichung von $\bar{\delta}(\acute{\mathbf{U}},\breve{\mathbf{U}}) = 0.45 \pm 0.058$.

3.2.4 Rendern von Handschriftsynthesen

Im vorhergehenden Abschnitt wurde ein Verfahren zur Komposition und Interpolation von handschriftlichen arabischen Glyphen zu Wörtern vorgestellt. Um für Methoden der Musterkennung geeignete Proben zu erhalten, müssen die aus Trajektorien $\breve{\mathbf{U}}$ bzw. $\acute{\mathbf{U}}$ bestehenden Wortsynthesen im nächsten Schritt in ein Bildformat überführt werden. Hierzu wird im Folgenden eine Technik zum Rendern von Handschrift vorgestellt, die insbesondere die Eigenheiten von Kugelschreibern oder grafitbasierten Schreibwerkzeugen wie Bleistiften reproduziert. Anschließend wird in Abschnitt 3.2.4.2 eine Modifikation vorgestellt, mit der sich mit Feder oder Füller geschriebene Handschrift simulieren lässt.

Der erste Schritt des Verfahrens besteht darin, Bildpunkte $\mathbf{o} = (x,y)^T \in \mathbf{I}^{H\times W} | x,y \in \mathbb{N}_0$ zu bestimmen, die an das Polygons $\breve{\mathbf{U}}$ angrenzen und dem Vordergrund zugeordnet werden sollen. Diese Bildpunkte weisen demnach eine durch ein Schreibutensil erzeugte Pigmentation auf, wogegen alle anderen zum Hintergrund gehören. Die Breite und Höhe der Matrix $\mathbf{I}^{H\times W}$ wird dabei so gewählt, dass sich letztlich alle Glyphen einer Probe innerhalb von $\mathbf{I}^{H\times W}$ befinden, und ihr umgebendes Rechteck



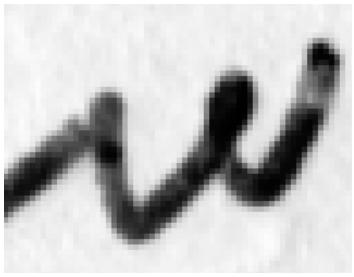
(a)

(b)

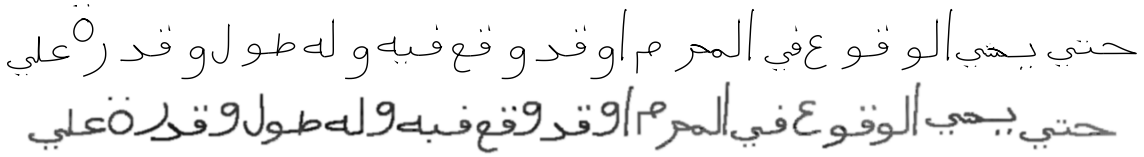
(c)

(d)

(e)Synthetische Probe



(f)Reale Probe



(g)Synthese von „حتي يحشي الوقوع في المحرم او قد وقع فيه وله طول وقدرة علي“

Abbildung 3.12. Render Methode I: (a) Schema zur Bestimmung der Pigmentierung nahe eines Polygons $\mathbf{U}$. (b) Ergebnis für $\rho = 6, \mu_{\mathsf{LW}} = 2, \sigma_{\mathsf{LW}} = 3.5$ (ohne Verwendung von Filtern). (c) Test Textur angewandt auf (b). (d) $\rho = 30, \mu_{\mathsf{LW}} = 2, \sigma_{\mathsf{LW}} = 1.0$ mit Kugelschreiber-Textur. (e-f) Vergleich von Synthese und realer Probe (Kugelschreiber). (g) Handschriftsynthese eines arabischen Satzes (oben als Visualisierung der Trajektorien durch eine Vektorgrafik, unten mit der vorgeschlagenen Rendering Methode).

dabei einen manuell definierten Rand freilässt. Hierzu werden zunächst mittels Vektorgrafik geeignete Koordinaten $\breve{\mathbf{o}} \in \mathbb{R}^2$ ermittelt. Wie aus Abbildung 3.12 (a) hervorgeht, wird zwischen zwei benachbarter Punkte $\mathbf{p}$ und $\mathbf{q}$ aus $\breve{\mathbf{U}}$ bzw. $\acute{\mathbf{U}}$ interpoliert. Es werden jeweils $\rho_r * ||\mathbf{p} - \mathbf{q}||$ zufällige neue Punkte $\breve{\mathbf{o}}'$ interpoliert, die auf dem Geradenabschnitt $\overline{\mathbf{pq}}$ liegen:

$$\breve{\mathbf{o}}' = \nu\mathbf{p} + (1 - \nu)\mathbf{q} | \nu \in [0; 1], \tag{3.21}$$

(ν ist gleichverteilt ist und ρ_r wird vom Anwender bestimmt) .

Sei weiterhin $\mathbf{n_{pq}}$ der Normaleneinheitsvektor zu $(\mathbf{q} - \mathbf{p})$, so ergeben sich die Koordinaten der pigmentierten Bildpunkte, indem alle $\breve{\mathbf{o}}'$ entlang $\mathbf{n_{pq}}$ verschoben werden:

$$\breve{\mathbf{o}} = \mathbf{o}' + \mathcal{N}(\mu_{\mathsf{LW}}, \sigma_{\mathsf{LW}})\mathbf{n_{pq}}. \tag{3.22}$$

Der Erwartungswert μ_{LW} regelt hierbei die Linienbreite. Diese wird um bis zu

20% reduziert wird, falls sich $\breve{\mathbf{o}}$ in der Nähe eines *Pen Up* oder *Pen Down* Punktes befindet, um zunehmenden bzw. abnehmenden Druck zu simulieren. Mit σ_{LW} lässt sich darüber hinaus die Schärfe des Schriftzuges steuern. Hieraus ergibt sich bereits der Prototyp des zu rendernden Bildes, welcher aus allen pigmentierten Bildpunkten des Vordergrundes

$$\mathbf{o} = \begin{pmatrix} \lfloor \breve{o}_x + 0.5 \rfloor \\ \lfloor \breve{o}_y + 0.5 \rfloor \end{pmatrix} \tag{3.23}$$

besteht (siehe Abbildung 3.12 (b)). Akzenturtiere Konturen lassen sich durch reduzierte Streuung erzielen, indem die Gleichung (3.22) durch die Bedingung $|\mathcal{N}(\mu,\sigma)| < \varpi\sigma$ ergänzt wird.

Soweit lassen sich einfache schwarzweiße (bei kumulativer Pigmentierung graue) Bilder wie in Abbildung 3.12(b) generieren. Im nächsten Abschnitt wird erläutert, wie sich Synthesen mit nicht wiederholenden (nicht gekachelten) Texturen berechnen lassen, wie sie für die Synthesen in Abbildung 3.12 (c) und (d) eingesetzt wurden.

3.2.4.1 Berechnung von Texturen im Frequenzraum

Viele gebräuchliche Schreibwerkzeuge, wie Kugelschreiber oder Bleistifte, erzeugen keine gleichmäßig gefärbten Linien, sondern übertragen ihre Pigmente vielmehr aufgrund von Unregelmäßigkeiten der Papieroberfläche, der Schreibgeschwindigkeit und des eingesetzten Drucks. Wie Abbildung 3.12(f) zeigt, hat dies auch im Falle eines herkömmlichen Kugelschreibers auf glattem, weißem Papier zur Folge, dass die Schrift im eingescannten Dokument durch sehr unterschiedliche Intensitäten $I(x,y)$ repräsentiert wird. Sie lässt sich daher nicht durch ein wiederholendes Muster (gekachelte Textur) darstellen. In Abhängigkeit der verwendeten Methoden der Vorverarbeitung, wie Binarisierung und Ausdünnung, kann diese unregelmäßige Pigmentierung zu abweichenden Ergebnissen bis hin zur Fragmentation von Pieces of Arabic Words (PAWs) führen. Daher sollen diese Eigenschaften auch in das System zur Schriftsynthese integriert werden, wozu im Folgenden ein geeigneter Lösungsansatz vorgestellt wird.

Inspiriert durch die Fähigkeit der Fourier-Transformation wesentliche Eigenschaften eines Bildes bereits durch die Überlagerung weniger Frequenzen wiederzugeben, werden dynamische Texturklassen gestaltet, welche die Eigenschaften unterschiedlicher Schreibwerkzeuge simulieren sollen. Hierzu werden zunächst bis

zu zehn Tripel (λ, θ, ϕ) im Frequenzraum ausgewählt, welche im Ortsraum durch die Funktionen

$$f_i(x,y) = 255cos\Big(\big(\lambda_i x + (1-\lambda_i)y\big)0.05\theta_i + \phi_i\Big) \mid f_i(x,y) \in \mathbb{R}, \tag{3.24}$$

repräsentiert werden. Jedes Tripel entspricht einem der Diagramme aus Abbildung 3.13 auf Seite 64. Die Auflösung des Bildes $\mathbf{I}^{H\times W}$, das aus $\mathbf{o}$ und $f_i(x,y)$ resultiert, wird durch dieses Verfahren nicht limitiert, so dass sich die Methode auch zur Texturierung ganzer Textseiten eignet.

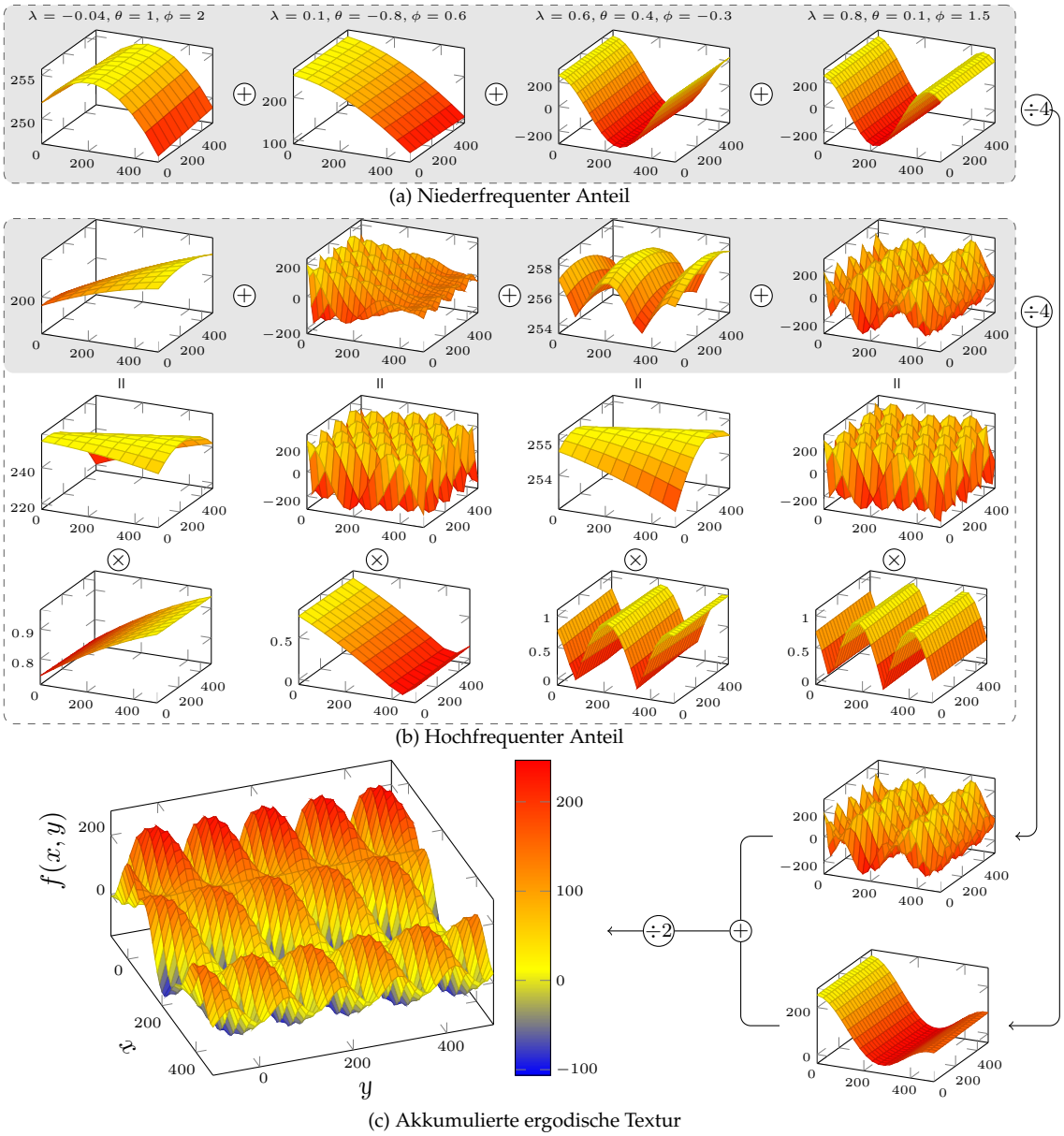
Hochfrequenter Anteil. Textur-Klassen werden definiert, indem für jede Funktion f_i experimentell Normalverteilungen für den Winkel θ_i, die Wellenlänge λ_i und die Phase ϕ_i festgelegt werden. Hierdurch wird erreicht, dass bei jeder generierte Synthese eine leicht abweichende Textur verwendet wird.

Wie in Abbildung 3.13(b) dargestellt, werden die f_i mit zusätzlichen niederfrequenten Funktionen multipliziert, um die Intensität der einzelnen hochfrequenter Texturanteile – und somit den Kontrast – lokal zu variieren. Anschließend werden alle f_i akkumuliert und hiernach durch ihre Anzahl dividiert, um eine finale hochfrequente Funktion $f(x,y)$ zu erhalten (wie oben rechts in Abbildung 3.13(c)).

Niederfrequenter Anteil. Pigmentierungsschwankungen hängen bei Kugelschreibern vom Druck, dem Untergrund und dem nicht immer perfektem Fluss der Tintenpaste ab. Sie treten daher nicht nur im Millimeterbereich, sondern auch in Regionen von bis zu ca. $3\,\mathrm{cm}^2$ auf. Durch Akkumulation verschiedener niederfrequenter Hilfsfunktionen $f_i(x,y)$, mit entsprechend höheren Wellenlängen λ_i, ergibt sich der niederfrequente Anteil der Textur, wie in Abbildung 3.13(a) und (c) veranschaulicht. Diese f_i haben einen Einfluss auf sämtliche – statt nur einzelne – hochfrequente Funktionen, und werden zu diesen addiert. So lassen sich besonders stark oder schwach pigmentierte Areale simulieren.

Beispiele für zwei Instanzen der Beispieltextur aus Abbildung 3.13, die jeweils auf das komplette Bild $\mathbf{I}_{o_x,o_y}$ angewandt wurden, finden sich in Abbildung 3.14. Beispiele zur Synthese von Handschrift mit simulierter Kugelschreiber-Textur wurden bereits in Abbildung 3.12(d, e) und (g) gegeben.

Berechnung der Texturmatrix. Die kombinierte Funktion $f(x,y)$ (siehe Abbildung 3.13(c)) soll die Eigenschaften bestimmter Schreibutensilien widerspiegeln. Hierzu

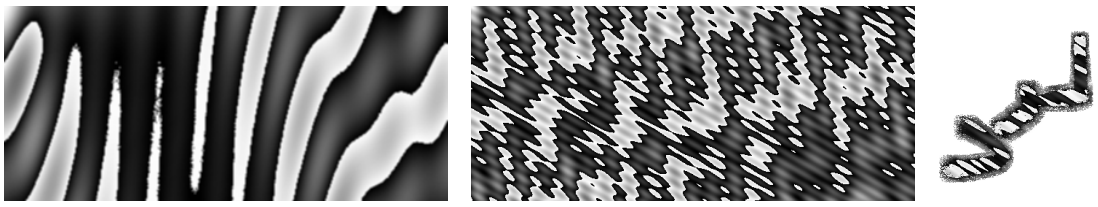


(a) Niederfrequenter Anteil

(b) Hochfrequenter Anteil

(c) Akkumulierte ergodische Textur

Abbildung 3.13. Beispiel für die Generierung einer Textur durch die Verknüpfung mehrerer Frequenzen.



(a) (b) (c)

Abbildung 3.14. Beispieltextur. (a,b) Instanzen der Beispieltexture, die auf das gesamte Bild angewandt werden. (c) Anwendung der Textur auf ein arabisches Wort mit hoher Linienstärke zur Generierung eines CAPTCHA.

erfolgt die Berechnung der Textur im Ortsraum für die zuvor berechneten Bildpunkte $\mathbf{o}$. Die sich im Ortsraum ergebenden Intensitäten des Vordergrundes werden nun für übliche Bildformate (BMP, PNG etc.) angepasst:

$$\mathbf{I}_{o_x,o_y} = I(\mathbf{o}) = \frac{f(o_x, o_y) + 250}{2} \; | I(\mathbf{o}) \in \{0{,}256\}. \tag{3.25}$$

Hierzu wird ein Farb- oder Graubild mit $\mathbf{I}_{x,y} = 256$ initialisiert, in das alle Intensitäten $I(\mathbf{o})$ der Schriftbildpunkte übertragen werden (wobei ein Rand von mindestens 3 Bildpunkten ausgespart wird).

Fazit zur vorgestellten Rendermethode. Durch den Einsatz polygonaler Glyphen und der beschriebenen Rendering-Methode lassen sich effizient multiple, einzigartige Synthesen mit stetigen Verknüpfungspunkten generieren. Weiterhin ermöglichen es die synthetisierten, dynamischen Texturen, für jede Handschriftsynthese verschiedene, nicht gekachelte Muster einzusetzen. Abhängig von der gewählten Textur-Klasse wird das synthetisierte Bild durch Median- bzw. Gaussfilter selektierbaren Grades modifiziert und anschließend als PNG-Datei gespeichert.

3.2.4.2 Erweiterte Rendering-Methode zur Simulation von historischer und kalligrafischer Handschrift

Die oben beschriebene Methode eignet sich dazu, mit Kugelschreibern, Bleistiften oder Kohle geschriebene Texte zu simulieren. Insbesondere ältere Dokumente wurden jedoch vorzugsweise mit Füllfederhaltern oder Federn verfasst. Diese werden – abhängig von der Breite der Federspitze – auch zum Anfertigen von Kalligrafien eingesetzt, und beeinflussen das Schriftbild mitunter deutlich. Professionelle arabische Kalligraphien werden im Rahmen dieser Arbeit nicht behandelt, und lassen sich aufgrund ihrer Komplexität auch nicht mit den zur Verfügung stehenden Glyphen simulieren. Das in diesem Abschnitt vorgestellte Renderingverfahren eignet sich jedoch zur Simulation von Alltagsschrift, die mit einem gewöhnlichen oder kalligraphischen Füllfederhalter geschriebene wurde, sowie historischer Handschrift (jeweils im Naskh Stil).

Um die Eigenschaften dieser Gruppe von Schreibwerkzeugen zu simulieren, sind einige Modifikationen und Erweiterungen vonnöten, die im Folgendem beschrieben werden. Die wesentlichen Modifikationen beziehen sich auf die *Schreibgeschwindigkeit* sowie Form und Eigenschaften der Spitze des Schreibinstrumentes,

welche im Weiteren kurz als *Stiftausprägung* bezeichnet werden. Diese Stiftausprägung entspricht grundlegend den Pinselmustern gängiger Mal- und Zeichenprogrammen.

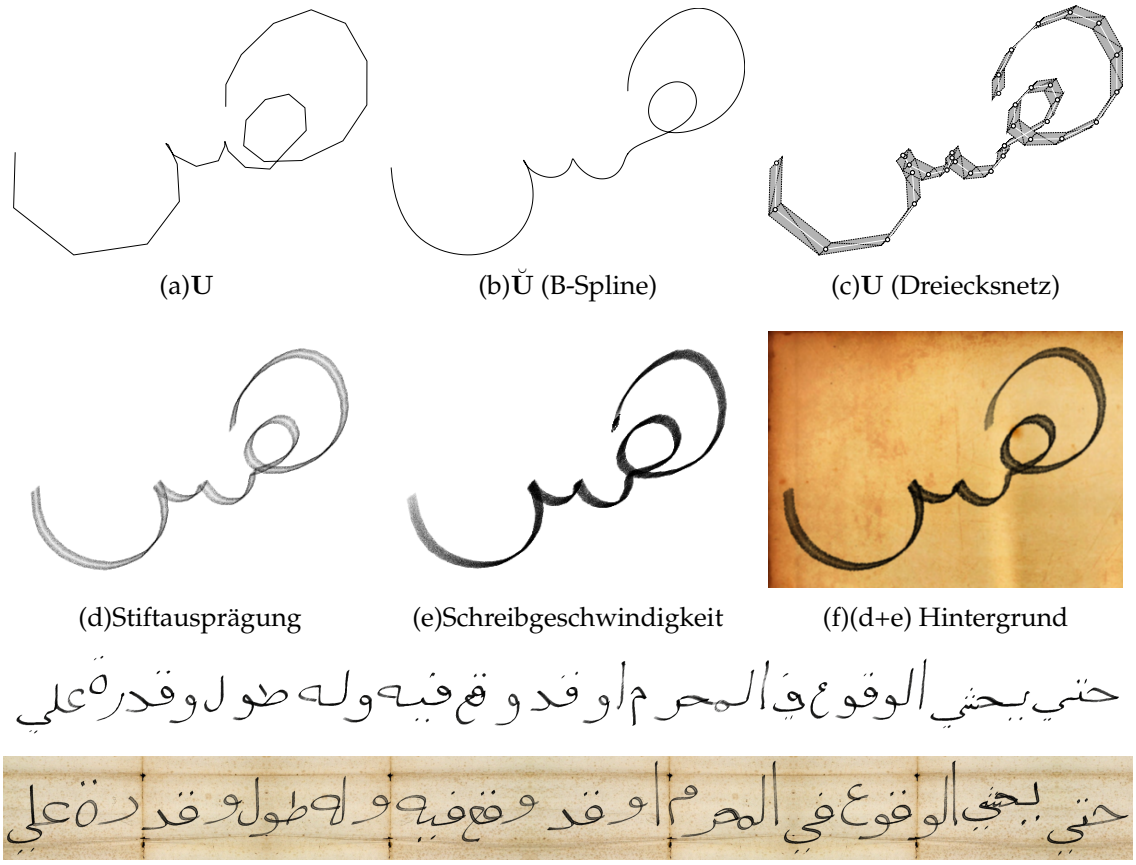
Stiftausprägung. Typischerweise lässt sich die Spitze eines Füllers oder vergleichbaren Schreibgerätes nicht als Kreis, sondern vielmehr als Linie oder Ellipsoid beschreiben. In diesem Fall hängt die tatsächliche, lokale Linienbreite nicht nur von der Stiftausprägung w_P (der Linienstärke des Stiftes) ab, sondern auch vom Winkel γ_P, in welchem der Stift um seine Längsachse rotiert, und dem Winkel $\gamma_{\breve{\mathbf{U}}}$ der Tangente der Trajektorie (gegeben durch $\mathbf{p}, \mathbf{q} \in \breve{\mathbf{U}}$). Durch Kombination dieser Winkel wird insbesondere die lokale Breite der Linie beeinflusst. Wie Abbildung 3.15(c) auf Seite 67 zeigt, kann dies bei einem breiten aber dünnen Federkopf stark ausgeprägt sein. Ausgehend von $\gamma_P = 45°$, wird γ_P kontinuierlich durch zwei überlagerte Kosinusfunktionen modifiziert, deren Amplituden, Wellenlängen und Phasen durch Normalverteilungen für jede Synthese neu berechnet werden. Die maximale Abweichung vom Initialwert wird auf $\gamma_P = 45° \pm 15°$ limitiert, um ein gemäßigtes Schriftbild sicherzustellen.

Entsprechend der Oberflächeneigenschaft eines Stiftes variiert der Kontakt mit der Papieroberfläche und folglich die verursachten Pigmentierung. Im vorgeschlagenem System wird diese Eigenschaft durch eine eindimensionale Funktion $f(x_P)\ |0 \leq x_P \leq w_P$ modelliert, welche das Pigmentierungspotential entlang der Längsachse der Stiftausprägung definiert. Dabei ist x_P die Entfernung von $\mathbf{o}$ zur Trajektorie $\breve{\mathbf{U}}$. Ein verdeutlichtes Beispiel findet sich in Abbildung 3.15(d) und (g), welchen die Funktion

$$f(x_P) = \begin{cases} 0.8\dfrac{x_P^2}{(0.8w_P)^2} + 0.2 & \textbf{falls } x_P < 0.8w_P \\ 1 & \textbf{sonst} \end{cases}, \tag{3.26}$$

zugrunde liegt.

Da für viele Experimente Binärbilder erwünscht sind, wurde eine vereinfachte, schnellere Methode implementiert. Zunächst erfolgt eine Extrusion der eindimensionalen Pinselausprägung entlang der Trajektorie $\acute{\mathbf{U}}$ bzw. $\breve{\mathbf{U}}$. Aus dieser wird, wie in Abbildung 3.15(c) dargestellt ist, ein Dreiecksnetz gebildet. Das Dreiecksnetz lässt sich anschließend durch Standardmethoden zum Füllen von Polygonen auf das Bild $\mathbf{I}$ übertragen. Abbildung 3.15(h) zeigt hierzu ein Beispiel, das mit der Bibliothek OpenCV gerendert wurde.



(a)**U** (b)$\breve{\mathbf{U}}$ (B-Spline) (c)**U** (Dreiecksnetz)

(d)Stiftausprägung (e)Schreibgeschwindigkeit (f)(d+e) Hintergrund

(g)Mit dem vorgestelltem Verfahren synthetisierte Textzeile.

(h)Effiziente Synthese eines Binärbildes derselben Textzeile)

Abbildung 3.15. Darstellung der einzelnen Schritte der erweiterten Rendering-Methode, von der polygonalen Darstellung **Ú** zur fertigen Synthese eines Bildes, das mit einem Füller angefertigte Handschrift simuliert.

Schreibgeschwindigkeit. Statistisch nimmt die Schreibgeschwindigkeit bei weitläufigen Linien oder Bögen zu und bei filigranen Strukturen ab. So wird beispielsweise der linke Bereich von س schneller als der rechte geschrieben (siehe Abbildung 3.15(e)). Eine hohe Schreibgeschwindigkeit ist oftmals Ursache für unregelmäßige Pigmentierung, die zu stellenweise blassen oder gesprenkelten Linien führt. Die zur Simulation dieser Eigenschaft benötigte Information über die Schreibgeschwindigkeit ist bereits in den Trajektorien enthalten, sie zu extrahieren erfordert allerdings einige Vorverarbeitungsschritte.

Aus den Trajektorien $\breve{\mathbf{U}}$ (oder **Ú**) lässt sich zunächst die relative, lokale Schreibgeschwindigkeit $v_r(i)$ für alle ASM basierten Glyphen in Bildpunkten per Sekunde

ableiten

$$v_r(i)\mathrm{pt/s} = \frac{\left\|\breve{\mathbf{u}}_{i+1} - \breve{\mathbf{u}}_i\right\|\mathrm{pt}}{\frac{n(\acute{\mathbf{U}})}{n(\breve{\mathbf{U}})}\Delta_t\mathrm{s}} \;\Big|\, 1 \leq i \leq n(\breve{\mathbf{U}}). \tag{3.27}$$

Die Opazität der Pigmentschicht α_a lässt sich nun anhand der normalisierten Schreibgeschwindigkeit $\dot{v} \in [0;1]$ abschätzen. Hierbei ist $\dot{v} = 0$ die minimale und $\dot{v} = 1$ die maximale Schreibgeschwindigkeit innerhalb aller Trajektorien der betrachteten Synthese (Wort, Satz oder Seite). Um sprunghafte Übergänge zu vermeiden, wird die Geschwindigkeit nahe der Verknüpfungen von Buchstaben interpoliert. Die Geschwindigkeit $\dot{v}$ der ersten $^{n(\breve{\mathbf{U}})}/_4$ Punkte $\breve{\mathbf{u}}_i$ von Buchstaben in mittlerer oder endender Position wird auf

$$\dot{v}(i) := \dot{v}(i)[1-\lambda_i] + \dot{v}_b\big(n(\breve{\mathbf{U}})\big)\lambda_i \;\big|\lambda_i = 4\frac{i}{n(\breve{\mathbf{U}})}, i \leq \frac{1}{4}n(\breve{\mathbf{U}}), \tag{3.28}$$

gesetzt, wobei $\dot{v}_b$ die Geschwindigkeit des *PenUp* Punktes des vorhergehenden Buchstaben angibt. Abbildung 3.16 zeigt den Verlauf von $\dot{v}(i)$ für ein PAW und einen Satz, wobei deutlich wird, dass die Schreibgeschwindigkeit bei einigen Buchstaben bzw. Teilen von Buchstaben erheblich zunimmt.

Mit der normierten Schreibgeschwindigkeit $\dot{v}$ lässt sich die Opazität der Pigmentierung nun durch

$$\alpha_a = 0.8(1-\dot{v}) + 0{,}2, \tag{3.29}$$

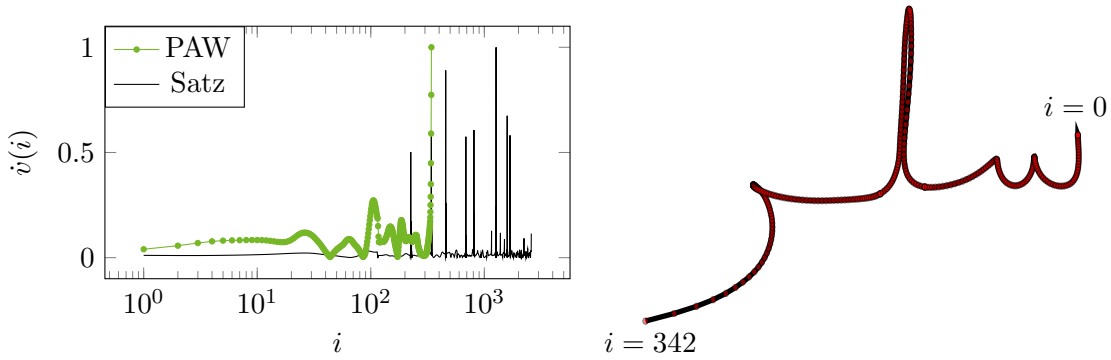
schätzen. Sei nun $\mathfrak{c}_0$ die aktuelle Farbe eines Bildpunktes im RGB Farbraum und $\mathfrak{c}_p$ die Farbe des verwendeten Stiftes, so ist der neue Farbwert

$$\mathfrak{c} = \mathfrak{c}_p\alpha_a + \mathfrak{c}_0\alpha_b(1-\alpha_a) \;|\mathfrak{c} \in \mathbb{R}^3. \tag{3.30}$$

Falls der Hintergrund keine Transparenz aufweist, wird α_b auf 1 gesetzt.

Obwohl zum Schreiben verwendete Pigmente (z.B. Ruß, oder Eisen(II,III)-oxid) abhängig vom Mahlgrad deckend oder halbdeckend sind, führt eine hohe Schreibgeschwindigkeit zu merklicher Transparenz oder ungleichmäßiger Färbung. Diese lassen sich durch eine entsprechende Reduzierung von α_a simulieren, wie Abbildung 3.15(e) zeigt.

Nachbearbeitung. Durch Kombination des eben beschriebenen Verfahrens mit den dynamischen Texturen aus Abschnitt 3.2.4.1, und der optionalen Verwendung texturierter Hintergründe, ergeben sich schließlich die in Abbildung 3.15(f) und (g)



(a) (b)

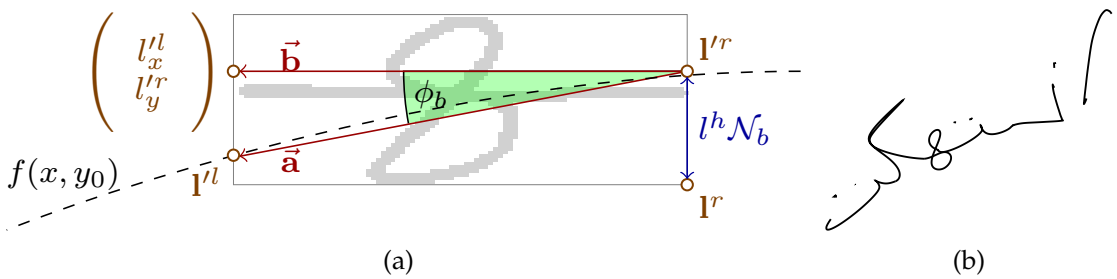
Abbildung 3.16. Darstellung der normierten Schreibgeschwindigkeit $\dot{v}(i)$. (a) Plot eines kompletten Satzes und des in (b) dargestellten synthetisierten PAW.

dargestellten Synthesen. Die dynamischen Texturen werden dabei mit der Opazität α_a aus Gleichung (3.29) multipliziert, wodurch sich sowohl vom Stift und Schreibvorgang abhängige als auch schwache, zufällige Unregelmäßigkeiten simulieren lassen.

3.3 Synthese einspaltiger Textseiten

In den vorhergehenden Kapiteln wurde beschrieben, wie sich einzelne Wörter oder Sätze arabischer Handschrift synthetisieren lassen. Der Forschungsfokus im Bereich arabischer Handschrifterkennung verschiebt sich jedoch zunehmend von der Buchstaben- und Worterkennung zu komplexeren Aufgaben der Dokumentanalyse, wie der Einordnung, Interpretation oder der Erkennung von Dokumenten bzw. Dokumentausschnitten. Daher wurden im Rahmen dieser Promotion auch Methoden zur Synthese ganzer Textseiten erforscht. Hierbei ist die Simulation der unteren Basislinien entscheidend, die im Zusammenhang mit vielen Problemen bei der Zeilen- und Wortdetektion steht.

Der hier vorgestellte Ansatz zur Synthese von Handschrift entlang gegebener, nichtlinearer Basislinien besteht aus drei Teilschritten. Zunächst wird für das jeweils erste PAW einer Zeile die Koordinate (x_0,y_0) bestimmt. Hiernach erfolgt die Berechnung der Basislinienkurvatur. Abschließend müssen alle PAWs entzerrt werden, deren Trajektorien sich überschneiden.



(a) (b)

Abbildung 3.17. Platzierung eines Buchstaben l entlang der Basislinie $f_b(x,y_0)$. (a) Schematische Darstellung zur Berechnung des Rotationswinkels ϕ. (b) Beispiel eines entlang einer stark gekrümmten Basislinie ausgerichteten synthetischen Wortes.

3.3.1 Position der Start PAWs

Entsprechend des üblichen Schriftbildes arabischer, handschriftlicher Dokumente, wird allen am Zeilenanfang befindlichen PAWs die maximale x-Koordinate zugewiesen: $x_0 = x_{max}$. Die y-Koordinate lässt sich mit Hilfe des Abstandes zur jeweils vorherigen Zeile ermitteln. Dieser Abstand setzt sich aus der durchschnittlichen Höhe der Buchstaben und einem benutzerdefinierten Faktor ς_l zusammen.

3.3.2 Basislinienkurvatur

Der zweite Teilschritt wird realisiert, indem die PAWs entlang der simulierten Basislinien positioniert werden, welche durch Funktionen $f_b(x,y_0)$ definiert sind (die Berechnung dieser Funktionen wird in Abschnitt 3.3.4 beschrieben). Um sicherzustellen, dass die Basislinienkurvaturen unabhängig von der Auflösung sind, werden x und y_0 normiert, indem durch die Breite bzw. Höhe der zu rendernden Synthese dividiert wird.

Um den Text an eine Basislinie f_i auszurichten, muss zunächst die Höhe der ersten Glyphe des betrachteten PAW entsprechend angepasst werden. Bestimmt $\mathbf{l}^r$ die untere rechte Ecke des umgebenden Rechteckes einer Glyphe l, so kann dies durch die Translationsmatrix

$$\mathbf{T}_{t\text{norm}} = \mathbf{T}_t \left(\begin{pmatrix} l^r_x \\ l^r_y \end{pmatrix} - \begin{pmatrix} l^r_x \\ f_b(l^r_x, y_0) \end{pmatrix} \right), \tag{3.31}$$

erreicht werden, so dass sich $\mathbf{l}'^r$ auf der Basislinie befindet.

Nach der groben Korrektur erfolgt nun die Feinabstimmung der y-Position in Abhängigkeit der Buchstabenklasse. Wie in Abbildung 3.17 dargestellt, geschieht

dies durch eine Translation

$$\mathbf{T}_{t\text{adapt}} = \mathbf{T}_t \begin{pmatrix} 0 \\ l^h \mathcal{N}_b \end{pmatrix} \; |\mathcal{N}_b = \mathcal{N}(\mu_r, \sigma_r). \tag{3.32}$$

Dabei gibt l^h die Höhe von $\mathbf{l}$ an (des $\breve{\mathbf{U}}$ umgebenden Rechtecks) während μ_r und σ_r entsprechend Gleichung 3.8 berechnet werden. Hiernach befindet sich die Glyphe entsprechend der Klasse teilweise oder gänzlich ober- bzw. unterhalb der Basislinie. Wie in Abschnitt 3.2.1 beschrieben, wurden mit Hilfe der IESK-arDB-Datenbank die normierten statistischen Relationen $\mathcal{N}(\mu_r, \sigma_r)$ für alle Buchstabenklassen berechnet. Damit lassen sich die jeweils ersten Glyphen eines PAW durch die Transformation

$$\mathbf{T}_{1.\text{Glyphe}} = \mathbf{T}_{t\text{adapt}} \cdot \mathbf{T}_{t\text{norm}}. \tag{3.33}$$

korrekt platzieren.

Abgesehen von der Position hängt auch der Winkel einer Glyphe von der Basislinie ab, da eine Abweichung von der idealen, waagerechten Basislinie auch während des Schreibens eines PAW geschieht. Dieses Verhalten wird simuliert, indem alle Punkte einer Glyphe $\breve{\mathbf{U}}$ um den *PenDow*n-Punkt $\breve{\mathbf{u}}_1$ rotiert werden. Sei $\mathbf{l}^l$ die untere linke Ecke des umgebenden Rechteckes des aktuellen Buchstaben $\mathbf{l}$, dann ergibt sich der Rotationswinkel wie folgt:

$$\phi_b = cos^{-1} \frac{\langle \vec{\mathbf{a}}, \vec{\mathbf{b}} \rangle}{|\vec{\mathbf{a}}||\vec{\mathbf{b}}|} \;\; \left| \vec{\mathbf{a}} = \mathbf{l}'^r - \mathbf{l}'^l, \; \vec{\mathbf{b}} = \mathbf{l}'^r - \begin{pmatrix} l'^l_x \\ l'^r_y \end{pmatrix} . \right. \tag{3.34}$$

Eine Glyphe $\breve{\mathbf{U}}_i$ in mittiger und endender Form wird – wie schon bei der Komposition einzelner Wörter – um den Verbindungspunkt mit ihrem Vorgänger $\breve{\mathbf{U}}_{i-1}$ rotiert, so dass $\mathbf{T}_{\text{Baseline}} \cdot \breve{\mathbf{u}}_1 = \breve{\mathbf{u}}_n^- 1$ erfüllt wird. So ergibt sich die Transformation zur Anpassung aller folgenden Glyphen eines PAW durch

$$\mathbf{T}_{\text{Baseline}} = \mathbf{T}_t \left(\breve{\mathbf{u}}_n^{-1} \right) \cdot \mathbf{T}_r(\phi_b) \cdot \mathbf{T}_t \left(-\breve{\mathbf{u}}_1 \right). \tag{3.35}$$

Aus den obigen Betrachtungen ergibt sich nun die Anpassung aller Glyphen an die Basislinien:

$$\breve{\mathbf{U}} := \begin{cases} \mathbf{T}_{1.\text{Glyphe}} \cdot \breve{\mathbf{U}} \textbf{ falls } \breve{\mathbf{U}} \text{ die erste Glyphe eines PAW ist} \\ \mathbf{T}_{\text{Baseline}} \cdot \breve{\mathbf{U}} \textbf{ sonst} \end{cases}, \tag{3.36}$$

Durch dieses Vorgehen wird erreicht, dass sich die synthetisierte Handschrift organisch an die Basislinien anpasst und zusätzliche Varianzen auf Satz- und Absatzebene erzielt werden.

3.3.3 Behebung von Intersektionen

Im letzten Teilschritt ist es erforderlich, Intersektionen zwischen den Trajektorien benachbarter Pieces of Arabic Words (PAWs) zu detektieren und anschließend mit einem iterativen Verfahren zu beheben. Dazu werden zunächst alle PAWs $\mathbf{B}$ ermittelt, die zur Zeile oberhalb des aktuellen PAW $\mathbf{A}$ gehören (sofern $\mathbf{A}$ der obersten Zeile angehört, wird dieser Schritt übersprungen).

Falls das umgebendes Rechteck eines PAW $\mathbf{B}$ mit dem von $\mathbf{A}$ überlappt (oder ein Spiel $< \epsilon$ aufweist), besteht die Gefahr, dass Intersektionen zwischen diesen PAWs auftreten. Für jedes $\mathbf{B}$ wird daher berechnet, ob ein Paar intersektierender Geradenabschnitte $\mathfrak{g}_\mathbf{A} = \overline{\mathbf{l}'^i_\mathbf{A}\mathbf{l}'^{i+1}_\mathbf{A}} \in \mathbf{A}$ und $\mathfrak{g}_\mathbf{B} = \overline{\mathbf{l}'^i_\mathbf{B}\mathbf{l}'^{i+1}_\mathbf{B}} \in \mathbf{B}$ existiert. Falls ja, wird die Position von $\mathbf{A}$ mit der Translationsmatrix $\mathbf{T}_t(0,\psi)$ korrigiert. Anschließend wird analog auf Intersektion mit dem Vorgänger-PAW von $\mathbf{A}$ getestet, denen gegebenenfalls durch $\mathbf{T}_t(\psi,0)$ entgegengewirkt wird. Dieser Vorgang wird wiederholt, bis keine Intersektionen zwischen $\mathbf{A}$ und einem anderen PAW detektiert werden.

Überprüfung auf potentielle Intersektionen. Um zu überprüfen, ob Intersektionen zwischen den Trajektorien von $\mathbf{A}$ und $\mathbf{B}$ existieren, genügt es zunächst, alle Paare von Geradenabschnitten auf Überschneidungen zu untersuchen. Die Geraden sind durch folgende Gleichung gegeben:

$$\begin{gathered} f_l^i(x) = m_i x + b_i \\ m_i = \frac{l'^i_x - l'^{i+1}_x}{l'^i_y - l'^{i+1}_y}\,, b_i = l'^i_y - m_i x. \end{gathered} \tag{3.37}$$

Sofern zwei Geraden nicht parallel liegen, haben sie den Schnittpunkt

$$\mathbf{z} = \begin{pmatrix} b_\alpha \\ m_1 b_\alpha + b_1 \end{pmatrix} \Leftarrow f_l^1(x) = f_l^2(x) \;\text{ mit }\; b_\alpha = \frac{b_2 - b_1}{m_1 - m_2}. \tag{3.38}$$

Die Darstellungen Abbildung 3.18 veranschaulichen, wie Intersektionen zwischen zwei Geradenabschnitten $\mathfrak{g}_\mathbf{A}$ und $\mathfrak{g}_\mathbf{B}$ benachbarter PAWs detektiert sowie potentielle Intersektionen zwischen den resultierenden Linienabschnitte im zu rendernden Bild $\mathbf{I}$ prädiktiert werden. Das abgebildete Raster hat die Größe $1pt$, ε beträgt $2pt$.

Im einfachsten Fall lässt sich $\mathbf{z}$ durch Linearkombination von $\mathfrak{g}$ repräsentieren. Dann liegt der Schnittpunkt $\mathbf{z}$ innerhalb der Geradenabschnitte $\mathfrak{g}_\mathbf{A}$ und $\mathfrak{g}_\mathbf{B}$, die sich, wie in Abbildung 3.18 (a), überschneiden. Hierdurch wird eine Intersektion zwischen den zugehörigen PAWs detektiert.

Da das vorgestellte Synthesesystem in erster Linie nicht zur Generierung von Trajektorien sondern von offline Proben konzipiert wurde, ist das Verfahren zur

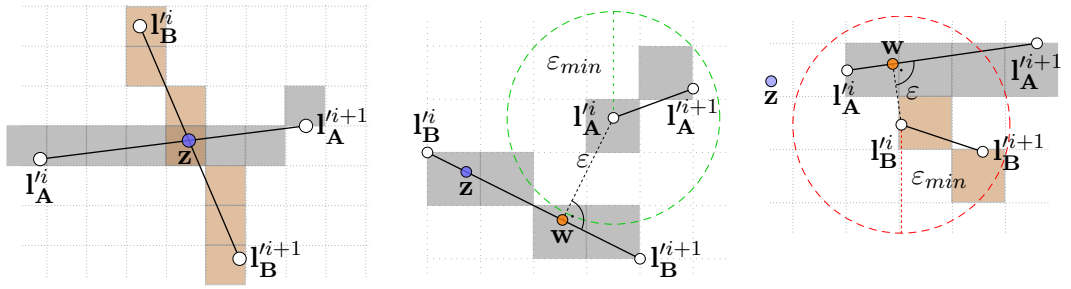


Abbildung 3.18. Darstellung der Überprüfung auf mögliche Intersektionen zwischen benachbarter Buchstaben. (a) Beispiel einer eindeutige Intersektion der beiden Geradenabschnitte. (b) Die Geradenabschnitte liegen sehr nahe beieinander, jedoch besteht keine Notwendigkeit das PAW **A** zu verschieben. (c) Obwohl sich die beiden Gradenabschnitte nicht überschneiden, wird eine Intersektion im korrespondierenden Bild **I** erwartet.

Detektion von Intersektionen an dieser Stelle noch nicht hinreichend. In einem synthetisierten Bild treten unter Umständen auch dann Intersektionen auf, wenn sich $\mathfrak{g}_\mathbf{A}$ und $\mathfrak{g}_\mathbf{B}$ mathematisch nicht überschneiden, jedoch nahe beieinander liegen. Daher ist es notwendig, ein Spiel ε_{min} zu definieren, dass den Mindestabstand zweier Geradenabschnitte festlegt. Hierzu wird zunächst der minimale Abstand ε zwischen den Geradenabschnitten, wie in Abbildung 3.18(b) und (c), berechnet. Entscheidend ist, dass ε_{min} wenigstens einen Bildpunkt größer als die zu erwartende Breite der Stiftausprägung ist. Sei $\mathbf{l}_\varepsilon \in \{\mathbf{l}'^{i}_\mathbf{A}, \mathbf{l}'^{i+1}_\mathbf{A}, \mathbf{l}'^{i}_\mathbf{B}, \mathbf{l}'^{i+1}_\mathbf{B}\}$ ein Endpunkt von einer der beiden Geradenabschnitte und $\mathbf{w}$ jener Punkt auf dem jeweils anderem Geradenabschnitt, der mit $\mathbf{l}_\varepsilon$ einen zu diesem orthogonalen Vektor bildet, so ist die euklidische Distanz $\|\mathbf{l}_\varepsilon - \mathbf{w}\|$ ein Kandidat für ε. Falls sich $\mathbf{z}$ – wie in Abbildung 3.18(a) und (b) – auf einem der Geradenabschnitte befindet, gehört der zu ε resultierende Punkt $\mathbf{l}_\varepsilon$ zum anderen Geradenabschnitt; andernfalls muss $\|\mathbf{l}_\varepsilon - \mathbf{w}\|$ für alle vier Punkte berechnet werden. Liegt $\mathbf{w}$ außerhalb des Geradenabschnittes, so wird stattdessen die kürzeste Entfernung zu einem der Endpunkte des gegenüberliegendem Geradenabschnittes berechnet.

Im nächsten Abschnitt wird erläutert, wie sich geeignet Basislinienfunktionen $f(x,y_0)$ ermitteln lassen.

3.3.4 Definition und Optimierung von Basislinienfunktionen

Die Basislinien arabischer Handschrift weichen, im Vergleich zu gedrucktem Arabisch, von idealen horizontalen oder schrägen Basislinien ab, so dass eine perfekte Basislinienkorrektur auf Satz und Wortebene nicht in jedem Fall erreichbar ist. Die

Eigenschaften solcher Basislinien beeinflussen daher in nicht unerheblichem Maße diverse zur automatischen Dokumentanalyse erforderlichen Methoden, angefangen von der Zeilensegmentierung bis hin zur Wort- bzw. Buchstabenklassifikation. Deswegen ist es im Kontext der Handschriftsynthese wichtig, Methoden zur Simulation von realitätsnahen Basislinien zu erforschen, um Schriftproben mit authentischem Herausforderungsgrad zu erschaffen. Im Folgenden wird hierzu ein Ansatz erläutert, bei dem eine zunächst heuristische Basislinienfunktion (mathematische Formel) durch evolutionäre Algorithmen so optimiert wird, dass eine maximale Ähnlichkeit zwischen den resultierenden Textzeilen und den Grundwahrheiten realer Zeilen erreicht wird.

3.3.5 Validierung einer Gruppe simulierter Basislinien

Ein Absatz Ξ besteht aus einer Gruppe von $n(Xi)$ Textzeilen, die jeweils aus einer bestimmten Folge PAWs bestehen. Im folgenden Abschnitt wird lediglich die Positionen der umgebenden Rechtecke beachtet, nicht die Glyphen, aus denen sich das PAW zusammensetzt. Ein synthetischer Absatz Ξ_s ist das Ergebnis der Basislinienfunktion $f(x,y_0)$. Daher lassen sich die Basislinienfunktionen validieren, indem die Korrelation $\rho = corr(\Xi_r, \Xi_s)$ zwischen realen Ξ_r und synthetischen Ξ_s Absätzen berechnet wird. Hierbei besteht Ξ_r aus 11295 manuell etikettierten PAWs aus handschriftlichen arabischen Dokumenten.

Um die globale Korrelation ρ_g zu erhalten, wird zunächst ein Gaussian Mixture Model (GMM) mit den realen Basislinien Ξ_r trainiert. Hierzu dienen die Merkmale $x_i \in \mathbf{x}$. Diese bestehen jeweils aus dem Durschnittswert und der Standardabweichung des:

x_1,x_2 normalisierten Abstandes zwischen zwei Textzeilen,

x_3,x_4 Winkels ϕ zwischen einer Textzeile und der Horizontalen,

x_5,x_6 der Änderung von ϕ relativ zu y_0.

Anschließend wird mithilfe des GMM die Log-Likelihood-Funktion $\log \mathcal{L}(\theta|\mathbf{x})$ berechnet. Diese gibt die Wahrscheinlichkeit an, mit der ein Ξ_s mit dem Merkmalsvektor $\mathbf{x}$ zur Klasse der realen Basislinien θ gehört, deren Nachahmung ja das Ziel dieses Abschnitts darstellt. Die globale Korrelation ergibt sich nun aus

$$\rho_g = n(\Xi)^-1 \sum^{n(\Xi)} \log \mathcal{L}(\theta|\mathbf{x}). \tag{3.39}$$

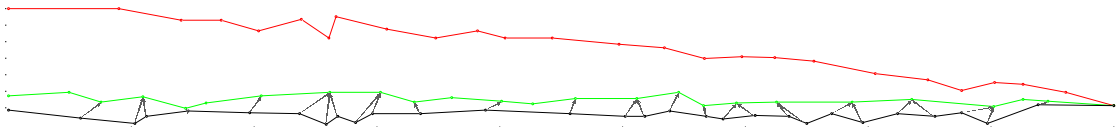


Abbildung 3.19. Validierung der normierten PAW Positionen einer synthetischen Textzeile (grau). Die grün dargestellte reale Basislinie weist (im Gegensatz zur roten) die maximale Ähnlichkeit zur synthetischen Basislinie auf.

3.3.6 Validierung einzelner Basislinien

Um außerdem einzelne Textzeilen zu detektieren, die eine entartete Basislinie aufweisen, wird der Abstand aller PAWs der synthetischen Textzeilen zu den realen Basislinien verglichen. Als Textzeilen-Repräsentation $\mathfrak{Z}$ dient hierbei die Folge aller geometrischen Mittelpunkte $\mathbf{s}$ der enthaltenen PAWs (von rechts nach links). Es werden alle realen und synthetischen Zeilen ($\mathfrak{Z}_r$ und $\mathfrak{Z}_s$) normiert und so verschoben, dass der Mittelpunkt $\mathbf{s_1}$ des ersten PAWs jeweils im Ursprung liegt. Nun werden zu allen $n(\mathfrak{Z}_s)$ Mittelpunkten $\mathbf{s} \in \mathfrak{Z}_s$ Nachbarn $\mathbf{s}^a, \mathbf{s}^b \in \mathfrak{Z}_r$ mit: $s_x^a > s_x > s_x^b$ ermittelt. Eine Visualisierung, in der alle $\mathbf{s}$ mit dem im euklidischem Raum jeweils nächsten Nachbarn verbunden werden, zeigt Abbildung 3.19. Nun lässt sich $\rho_l = corr(\mathfrak{Z}_s,\mathfrak{Z}_r)$ wie folgt berechnen:

$$\rho_l = log\left(1 - \left(\frac{\sum^{n(\mathfrak{Z}_s)} \|\mathbf{s} - \mathbf{s}^r\|}{n(\mathfrak{Z}_s)}\right)\right),\ \mathbf{s}^r = \lambda_r \mathbf{s}^a + (1-\lambda_r)\mathbf{s}^b,\ \lambda_r = \frac{s_x^a - s_x}{s_x^b - s_x^a}. \quad (3.40)$$

Indem der Mittelwert $\bar{\rho}_l$ der besten fünf Übereinstimmungen ρ_l eingesetzt wird, lässt sich die Korrelation der synthetischen mit den realen Basislinien durch

$$\rho = \bar{\rho}_l + \rho_g, \quad (3.41)$$

beschreiben. Hierbei gibt ρ die Güte an, mit der die Funktion $\tilde{f}(x,y_0)$ den Verlauf der realen Basislinien simuliert.

Definition und Optimierung der Basislinienfunktion. Die Funktionen $\tilde{f}_i$ werden zunächst manuell zu bestimmen. Anschließend erfolgt die automatische Optimierung der Funktionsparameter q_i. Das Ziel dieses Vorgangs liegt darin, dass sich für bestimmte Problemstellungen relevanten Merkmale manuell flexibel anpassen lassen, deren Ausprägung jedoch anschließend automatisch an reale Basislinienproben angeglichen wird. Dahingehend wurden, basierend auf der Studie historischer und zeitgenössischer arabischen Dokumente, verschiedene Funktionen $\tilde{f}_i$ definiert, welche die unterschiedlichen beobachtbaren Eigenschaften hinsichtlich Basislinien

(a)$\tilde{f}_1$ (b)$\tilde{f}_1$ (c)$\tilde{f}(x) = y_0^2\big(sin(5x) - 5x\big)$

Abbildung 3.20. Visualisierung der Basislinienfunktionen. (a-b) Zwei Beispiele der optimierten, dynamischen Funktion $\tilde{f}_1$. (c) Beispiel für eine statische, nicht optimierte Funktion, die durch Eingabe der angegebenen Formel in die dafür vorgesehene Schaltfläche der Benutzeroberfläche erzeugt wurde.

reflektieren. Hierzu gehören in erster Linie Dokumente, deren Basislinien in Schreibrichtung moderat (jedoch mit jeder Zeile zunehmend ausgeprägter) nach unten geneigt sind. Beispiele hierfür sind in Abbildung 3.20(a) und (b) dargestellt. Die Synthesen aus Abbildung 3.21 basieren auf der Funktion $\tilde{f}_1(x,y_0) =$

$$-2q_1xy_0 + y_0\Big[0{,}7\sin\big(2{,}5x - 1{,}2\big) + q_2 \cdot \sin\big(10x\big) + q_3 \cdot \sin\big(8x + 2\big)\Big]. \qquad (3.42)$$

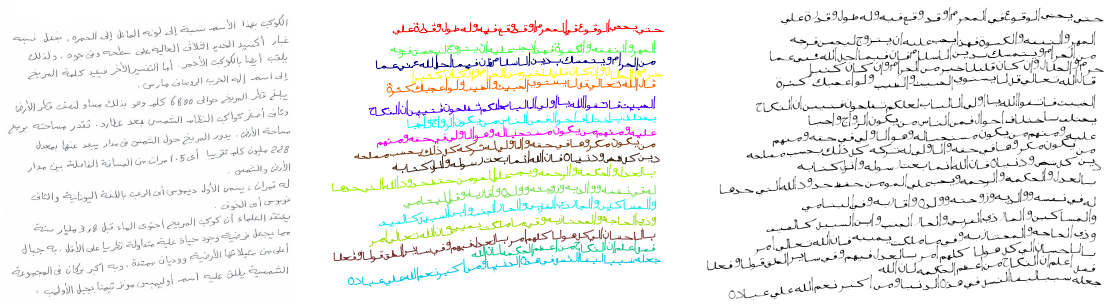
Damit alle mit dieser Funktion generierten Synthesen statistisch plausible Abweichungen aufweisen, werden die n_l Parameter q_i während des Optimierungsvorgangs durch Normalverteilungen $\mathcal{N}_i(\mu_i,\sigma_i)$ ersetzt. Nach der Optimierung werden hiermit vor jedem Synthesevorgang allen q_i zufällige Werte zugewiesen.

Geeignete Normalverteilungen $\mathcal{N}_i$ werden durch evolutionäre Algorithmen ermittelt. Die Repräsentation eines Individuums ist dabei durch $(\mu_1, \sigma_1, \ldots \mu_{n_l},\sigma_{n_l})$, dessen Fitness durch ρ gegeben. Um ρ für ein einzelnes Individuum zu berechnen, werden mithilfe von $\mathcal{N}_i$ mehrere Parametrisierungen für q_i generiert. Nach dem Einsatz von mehreren Epochen Mutation und Rekombination wird das am Besten an die realen Basislinien Ξ_r angepasste Individuum selektiert. Im Fall von $\tilde{f}_1$ ergibt sich z.B. die Parametrisierung:

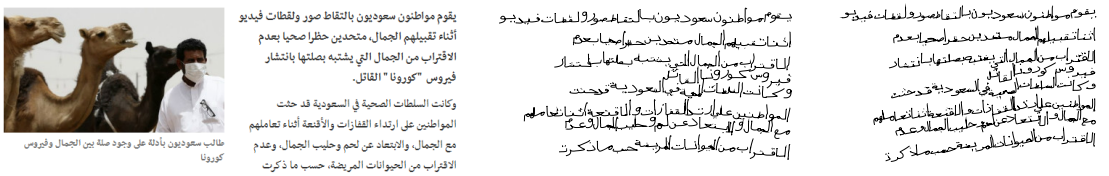
$$q_1 \Leftarrow \mathcal{N}(0{,}15; 0{,}84),\ q_2 \Leftarrow \mathcal{N}(0{,}67; 0{,}51),\ q_1 \Leftarrow \mathcal{N}(0{,}35; 0{,}17), \qquad (3.43)$$

die für die recht ähnliche Ausprägung der Basislinien in Abbildung 3.20 (a) und (b) verantwortlich ist.

Abhängig von der verwendeten Formel $\tilde{f}_i$ und den realen Basislinien Ξ_r, können Synthesen mit hohem oder niedrigem Schwierigkeitsgrad (bezüglich Zeilensegmentierung) erzeugt werden. Dies ist wichtig, um unterschiedlichen Forschungsschwerpunkten bzw. realen Bedingungen gerecht werden zu können.



(a)Reale Handschrift (b)Synthese (PDF) (c)Synthesis (PNG)



(d) (e) (f)

Abbildung 3.21. Ergebnisse der Handschriftsynthese bei Verwendung der simulierten Basislinien. (a) Eine Reale Probe einer der mit Grundwahrheiten versehenen handschriftlichen Textseiten, die zur Validierung der simulierten Basislinien verwendet wurde. (b) Einer mit der Funktion $\tilde{f}_1$ generierten Synthese mit farblich gekennzeichneten Zeilen. (c) Eine entsprechend mit der erweiterten Renderingmethode erzeugte Synthese. (d) Ausschnitt einer arabischen BBC Webseite. (e,f) Synthesen des aus der Webseite extrahierten Textes.

3.4 Zusammenfassung

In diesem Kapitel wurde ein Verfahren zur Synthese arabischer Handschrift vorgestellt. Ausgehend von Active Shape Modells zur Generierung neuer Glyphen für jede Synthese, wurde hierbei digitaler Text zunächst in Trajektorien übersetzt. Es wurde gezeigt, dass sich durch diverse anschließende Manipulationen zusätzliche handschriftliche Variationen simulieren lassen. Hiernach wurden Rendering Verfahren vorgestellt, mit welchen sich unterschiedlich ausgeprägte Proben arabischer handschriftlicher Wörter und Texte synthetisieren lassen. Diese lassen sich, wie im nächsten Kapitel dargelegt wird, zur Ergänzung bestehender Datenbanken einsetzen.

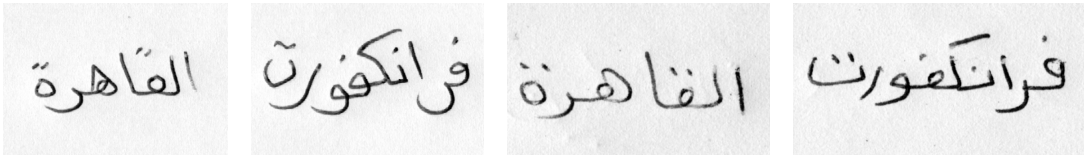
KAPITEL 4

Analyse und Synthese arabischer Handschrift-Datenbanken

Das folgende Kapitel gibt eine Übersicht über verschiedene Datenbanken arabischer Handschrift. Im ersten Abschnitt werden zunächst die herkömmlichen, während der Promotion angefertigten Datenbanken vorgestellt. Anschließend wird in Abschnitt 4.2 eine Methode zur Generierung von Pseudo-Texten vorgestellt, welche als Grundlage der synthetischen Datenbanken verwendet werden sollen. Hierauf erfolgt in Abschnitt 4.3 die Analyse und der Vergleich der Pseudo-Texte und einiger Datenbanken. Weiterhin wird die Erweiterung der herkömmlichen mit synthetischen Wortdatenbanken in Abschnitt 4.4 diskutiert. Abschließend wird in Abschnitt 4.5 kurz auf synthetische Datenbanken von handschriftlichen Textseiten eingegangen.

4.1 Die bestehende IESK-arDB Datenbank

Da das im letzten Kapitel beschriebene System zur Schriftsynthese lediglich eine Forschungsarbeit zur Ermittelung des Potentials des vorgestellten Ansatzes darstellt, erfolgte die Entwicklung der in den folgenden Kapiteln aufgeführten Bildverarbeitungsmethoden zunächst auf Grundlage traditioneller Datenbanken. Mangels einer für explizite Segmentierung geeigneten, frei verfügbaren und auf einem allgemeinen Vokabular aufbauenden Datenbank, wurde in der Arbeitsgruppe Neuro-Informationstechnik (NIT) die IESK-arDB (arabische Datenbank des Instituts für Informations- und Kommunikationstechnik(IIKT), ehemals IESK) geplant und



(a)A01_001 (b)A01_005 (c)A02_001 (d)A02_005

Abbildung 4.1. Beispiel für Proben der IESK-arDB des ersten und zweiten Schreibers.

realisiert [37]. Die IESK-arDB wurde insbesondere zur Evaluierung von Segmentierungsalgorithmen konzeptioniert und soll im folgenden Abschnitt erläutert werden. Einige Proben der IESK-arDB sind in Abbildung 4.1 dargestellt.

4.1.1 Aufbau der IESK-arDB Wortdatenbank

Das Vokabular der IESK-arDB umfasst 323 häufige arabische Wörter. Diese entstammen drei Rubriken: Häufig gebrauchte Nomen und Verben, internationale Städtenamen und im Zusammenhang mit innerer Sicherheit stehende Begriffe. Die Handschriften wurden von 22 Schreibern angefertigt, die aus verschiedenen arabischen Ländern (z.B. Ägypten und Jemen) sowie nicht arabischen Ländern (wie Pakistan) stammen. Alle Schreiber nutzten dabei den am weitesten verbreiteten Naskh Stil, bei dem die Charakteristiken der Schriftzeichen gut ausgeprägt sind.

Den Schreibern wurden Formblätter im Format DINA4 für jeweils 8 Wörter ausgehändigt, die mit Kugelschreibern ausgefüllt wurden. Die anschließende Datenaufnahme erfolgte in einer Auflösung von 300dpi mit handelsüblichen Scannern. Zusätzlich zu den Originalen wurden jedem Eintrag der Wortdatenbank jeweils eine binarisierte und eine ausgedünnte Version der Handschrift hinzugefügt.

Grundwahrheiten der IESK-arDB. Die Besonderheit der IESK-arDB liegt, neben dem verwendeten Vokabular, in den detaillierten Grundwahrheiten (GT). Zur Etikettierung der Wortproben wurde eine Benutzeroberfläche implementiert, in die verschiedene primäre und sekundäre Informationen eingegeben und hiernach als XML-Datei gespeichert werden.

Die primären Informationen umfassen die phonetischen Bezeichnungen aller arabischen Buchstaben entsprechend der Tabelle 1.1 auf Seite 5. Zudem wurden die umgebenden Rechtecke aller PAWs sowie die optimalen Segmentierungspunkte[1]

[1]Zum Zeitpunkt der Etikettierung wurde von der Verwendung eines gewöhnlichen Segmentierungsalgorithmus ausgegengen, der entsprechend manueller Segmentierung in der Mitte der Kashidas – vorzugsweise an einem Minima – trennt. Für zukünftige Datenbanken empfiehlt sich die

Abbildung 4.2. Beispiel für Proben der IESK-arDB$_{\text{Letter}}$.

zwischen zwei Buchstaben markiert. Außerdem wurde die untere Basislinie anhand zweier Punkte definiert.

Sekundäre Informationen betreffen vorwiegend den Schreiber der Wortprobe und geben Auskunft über seine Herkunft, schulische Ausbildung und Alter. Zudem wird die geschätzte Qualität der Wortprobe (saubere Schrift, normal, unleserlich) und die englische Übersetzung des Wortes angegeben.

4.1.2 Extraktion von Buchstabenproben – die IESK-arDB$_{\text{Letter}}$

Aus der IESK-arDB wurde die erste in der Arbeitsgruppe verwendete Buchstabendatenbank zum Trainieren und Testen von Buchstabenklassifikatoren extrahiert. Hierzu lassen sich die Segmentierungspunkte der Grundwahrheiten (GT) verwenden. Allerdings segmentiert die verwendete Segmentierungsmethode nicht in der Mitte zwischen zwei benachbarten, sondern jeweils nahe des linken Buchstaben (dies wird im nächsten Kapitel näher erläutert). Um eine Buchstabendatenbank zu erhalten, die besser mit den erwarteten Segmentierungsergebnissen harmoniert, wurde die IESK-arDB daher mit der automatischen Segmentierung in ihre Buchstaben zerlegt. Fehlerhafte Segmentierungen wurden anschließend manuell entfernt, wonach insgesamt 5.711 Buchstabenproben verfügbar sind.

Einige Proben der IESK-arDB$_{\text{Letter}}$ sind in Abbildung 4.2 dargestellt. Zusätzlich finden sich im Anhang in Abbildung A.3 weitere normierte Beispielproben. Eine aus Wörtern extrahierte Buchstabendatenbank enthält jedoch typischerweise nur wenige Proben zu selten in der jeweiligen Sprache vorkommenden Buchstabenklassen. In einigen Fällen sind gar keine Proben vorhanden (eine genaue Analyse erfolgt in Kapitel 4.3.1). Dies betrifft auch die IESK-arDB$_{\text{Letter}}$, was einen der Gründe

manuelle Ettiketierung von zwei Punkten zu beiden Seiten des optimalen Segmentierungspunktes, die einen Bereich aufspannen, zwischen denen eine Segmentierung nach menschlichem Ermessen korrekt erscheint.

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

Abbildung 4.3. Beispiel für Proben der Buchstabenklasse غ der IESK-arDB$_{\text{OnlineLetter}}$ (dargestellt ist die Variante für den Einsatz mit Vorklassifizierung zur Reduktion der Klassen, daher fehlen in Abbildung (b-d) die diakritischen Zeichen). (a) Visualisierung eines Ausschnitts einer von einem Schreiber ausgefüllten digitalen DIN A4 Seite. (b) Aus den Trajektorien berechnete offline Probe (c) Normierte offline Probe (d) Chamfer Distanz Transformation der Probe (siehe Kapitel 6.2.2)

darstellt, weshalb die aus separat geschriebenen Buchstaben bestehende Datenbank IESK-arDB$_{\text{OnlineLetter}}$ konzipiert wurde.

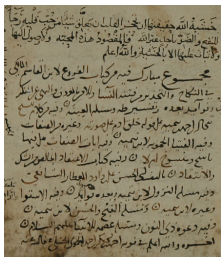
4.1.3 Online Buchstabenproben – die IESK-arDB$_{\text{OnlineLetter}}$

Die aus 28.046 Proben bestehende IESK-arDB$_{\text{OnlineLetter}}$ wurde bereits im letzten Kapitel in Abschnitt 3.1 vorgestellt. Dort wurden die Trajektorien verwendet, um die Active Shape Modells (ASMs) für die einzelnen Buchstabenklassen zu berechnen. Diese online Daten werden in den nächsten Kapiteln jedoch nicht genutzt. Abbildung 4.3 zeigt einige offline Repräsentationen von Proben der IESK-arDB$_{\text{OnlineLetter}}$, die – wie die Proben der IESK-arDB$_{\text{Letter}}$ – zum Trainieren und Testen der Buchstabenklassifikatoren in Kapitel 6 verwendet werden.

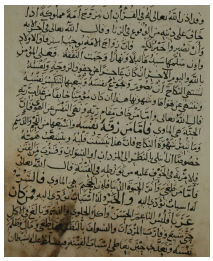
Da alle Buchstaben separat geschrieben wurden, ist die Verteilung der Proben auf die verschiedenen Klassen gut balanciert. Zumal seltene Klassen einen entsprechend hohen Informationsgehalt aufweisen (nur wenige Vokabeln enthalten diese Klasse), ist dies nicht nur für die Bildung der ASMs sondern auch für das Training der Support Vector Machines (SVMs) sinnvoll.

4.1.4 Historische arabische Dokumente – die IESK-arDB$_{\text{Hist}}$

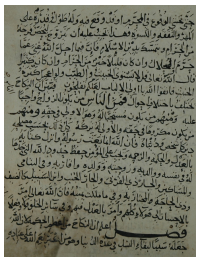
Als Erweiterung der eben vorgestellten zeitgenössischen Wort- und Buchstabendatenbanken, wurden über 300 Seiten Handschrift aus dem 14. Jahrhundert digitalisiert und veröffentlicht. Die hierzu angefertigten Grundwahrheiten bestehen aus Textdateien, die zu jeder Zeile Handschrift eine Unicode-Transkription enthalten. Extrem langgezogene Kashida werden dabei explizit gekennzeichnet, nicht lesbare Buchstaben mit einem „?“ substituiert. In Abbildung 4.4 sind einige Dokumente



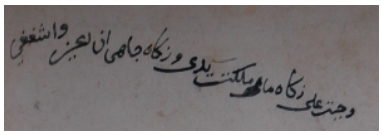
(a)im2_1



(b)im4_1



(c)im5_1



(d)im67_2

Abbildung 4.4. Beispiel für Proben der IESK-arDB$_{\mathsf{Hist}}$. (a-c) sind typische Proben, (d) zeigt hingegen eine Dokument, das nur eine einzelne Zeile mit deutlich gekrümmter Basislinie enthält.

der IESK-arDB$_{\mathsf{Hist}}$ dargestellt.

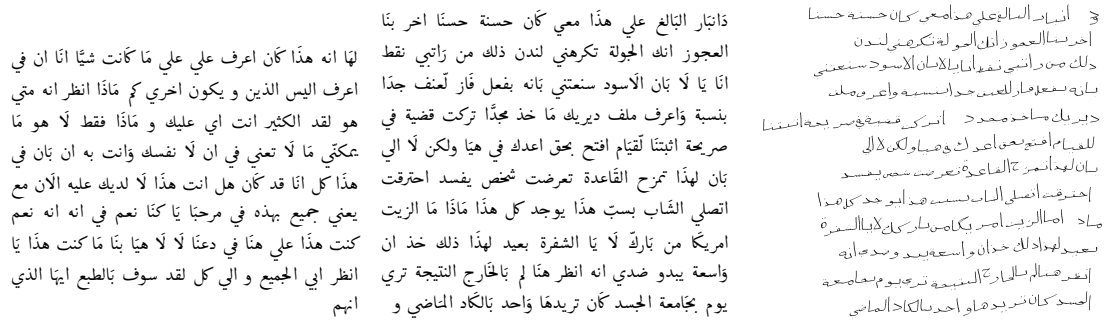
4.2 Generierung arabischer Pseudotexte

Das im letzten Kapitel erläuterte Verfahren zur Simulation der unteren Basislinien ist ein erster Schritt, um mit dem vorgestellte System zur Wortsynthese eine authentische Darstellung von handschriftlichen Textseiten zu erzielen. Darüber hinaus enthält aber auch das verwendete Vokabular wichtige Eigenschaften eines Textes. Beispielsweise unterscheidet sich die durchschnittliche Wortlänge in arabischen Texten deutlich von der eines für Formulare oder automatische Briefkopferkennung vorgesehen Vokabulars. Aus diesem Grund sollen arabische Pseudo-Texte erzeugt werden, welche die am häufigsten gebrauchten arabischen Wörter widerspiegeln. Hierzu wird $\mathcal{V}_{\mathsf{50k}}$, das aus den häufigsten 50.000 arabischen Wörtern bestehende Vokabular, mit den zugehörigen Auftrittshäufigkeiten $n(\mathbf{w}_{\mathcal{V}_i})$ der Einträge $\mathbf{w}_{\mathcal{V}_i}$ eingesetzt. Verwendet wurden hierzu mittels Web-Crawler ermittelte Daten [2]. Die Pseudo-Texte ermöglichen es, sowohl den Einfluss der Eigenschaften dieser häufig auftretenden Wörter auf Methoden wie Worterkennung zu untersuchen, als auch eine kontextbezogene Fehlererkennung zu konzeptionieren.

Pseudo-Texte lassen sich durch Verknüpfungen zufällig bestimmter Wörter $\mathbf{w}_{\mathcal{V}_i} \in \mathcal{V}_{\mathsf{50k}}$ erzeugen. Die Wahrscheinlichkeit, dass $\mathbf{w}_{\mathcal{V}_i}$ das nächste Wort des Pseudo-Textes ist, wird hierbei durch:

$$P(\mathbf{w}_{\mathcal{V}_i}) = \frac{n(\mathbf{w}_{\mathcal{V}_i})}{\sum_{j=1}^{50.000} n(\mathbf{w}_{\mathcal{V}_j})}, \tag{4.1}$$

definiert. Mit der Wheel-of-Fortune Methode werden iterativ Vokabeln gemäß



(a)$\mathcal{V}_{500}$ (ArabTEX) (b)$\mathcal{V}_{50k}$ (ArabTEX) (c)$\mathcal{V}_{50k}$ (IESK-arDB$_{\text{SynText}}$)

Abbildung 4.5. Arabischer Pseudo-Text. Mit ArabTEX dargestellter Unicode, der durch Verknüpfung der (a) 500 bzw. (b) 50.000 häufigsten arabischen Wörter generiert wurde. (c) Synthese des 50k-Pseudo-Textes (Zeilenumbruch des ArabTEX Absatzes kann variieren).

ihrer Auftrittswahrscheinlichkeit $P(\mathbf{w}_{\mathcal{V}_i})$ ausgewählt. Hierdurch lassen sich Pseudo-Texte, wie in Abbildung 4.5, erzeugen.

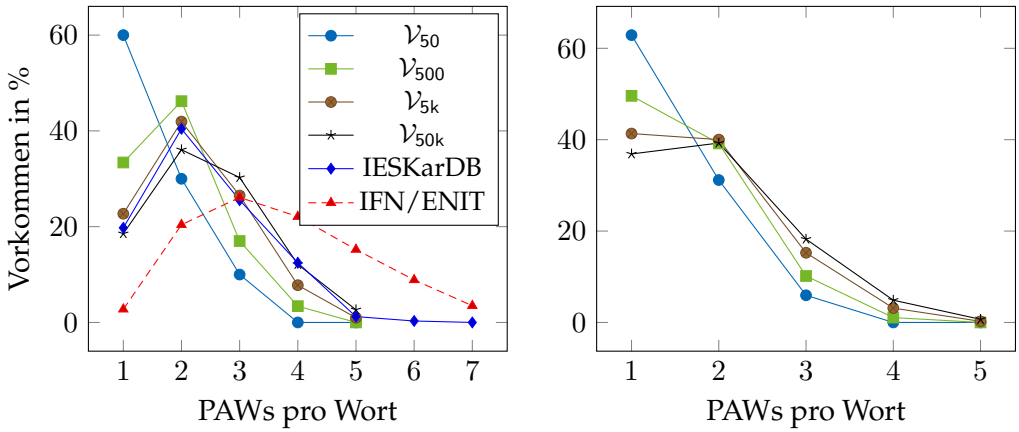
4.3 Analyse der Pseudo-Texte und Wortdatenbanken

Beim Trainieren, Validieren und Testen verschiedener zur Vorverarbeitung, Segmentierung und Erkennung von Texten eingesetzter Methoden, spielen neben der Handschrift auch die Eigenschaften der verwendeten Texte bzw. Pseudo-Texte eine entscheidende Rolle. Diese sollen an dieser Stelle daher zunächst analysiert werden, bevor im darauffolgenden Abschnitt der Aufbau einer synthetischen Datenbank beschrieben wird.

In dem Pseudo-Text in Abbildung 4.5(a), der auf den 500 häufigsten arabischen Wörtern basiert, sind offenkundig deutlich mehr kurze Wörter zu finden, als in dem Pseudo-Text in Abbildung 4.5(b), welcher alle Wörter aus $\mathcal{V}_{50k}$ als Grundlage hat. Wie Tabelle 4.1 verdeutlicht, besteht allerdings auch letzterer – im Gegensatz zu dem Vokabular einer automatisierten Briefkopferkennung (IFN\ENIT) – aus

	Vokabular					
	$\mathcal{V}_{50}$	$\mathcal{V}_{500}$	$\mathcal{V}_{5k}$	$\mathcal{V}_{50k}$	IESK-arDB	IFN\ENIT
$\overline{N_l}$	$2{,}77 \pm 0{,}83$	$3{,}89 \pm 1{,}16$	$4{,}63 \pm 1{,}32$	$5{,}36 \pm 1{,}42$	$4{,}74 \pm 1.42$	$8{,}18 \pm 3{,}10$
$\overline{\breve{N}_l}$	$2{,}56 \pm 0{,}86$	$3{,}11 \pm 1{,}39$	$3{,}52 \pm 1{,}68$	$3{,}90 \pm 2{,}02$	–	–
$\overline{N_{\text{PAW}}}$	$1{,}50 \pm 0{,}67$	$1{,}90 \pm 0{,}79$	$2{,}22 \pm 0{,}92$	$2{,}45 \pm 1{,}03$	$2{,}35 \pm 0{,}99$	$3{,}72 \pm 1{,}50$
$\overline{\breve{N}_{\text{PAW}}}$	$1{,}43 \pm 0{,}67$	$1{,}62 \pm 0{,}84$	$1{,}80 \pm 1{,}01$	$1{,}93 \pm 1{,}56$	–	–

Tabelle 4.1. Vergleich der durchschnittlichen ($\overline{N_l}$) und mit $P(\mathbf{w}_{\mathcal{V}_i})$ gewichteten ($\overline{\breve{N}_l}$) Buchstabenanzahl pro Wort sowie der entsprechenden durchschnittlichen Anzahl an PAWs pro Word.



(a)ungewichted (b)gewichted

Abbildung 4.6. Häufigkeit von arabischen Wörtern in Korrelation mit der Anzahl ihrer PAWs. (a) Ergebnisse für verschiedene Teilmengen des Vokabulars $\mathcal{V}_{50k}$ sowie für das Vokabular zweier Datenbanken. (b) Die Auftrittswahrscheinlichkeit $P(\mathbf{w}_{\mathcal{V}_i})$ der Wörter wird hier bei der Berechnung des Vorkommens beachtet, wodurch sich die Ergebnisse für $\mathcal{V}_{500} - \mathcal{V}_{50k}$ an die Serie $\mathcal{V}_{50}$ angleichen.

vielen kurzen Wörtern (wie Artikel oder Präpositionen). Arabische Wörter sind mit durchschnittlich 5 Buchstaben generell ziemlich kurz. Hinzu kommt, dass längere Wörter in mehrere PAWs aufbrechen, die – aufgrund des stärker variierenden Abstands vor allem bei Handschrift – gegebenenfalls mit kurzen Wörtern verwechselt werden.

Die durchschnittliche Anzahl an Buchstaben und PAW der IESK-arDB Datenbank entspricht in etwa jener der 5.000 häufigsten arabischen Wörter. Dies liegt daran, dass die IESK-arDB Datenbank, neben internationalen Städtenamen, auch viele allgemeine Vokabeln enthält. Anders verhält es sich bei der IFN\ENIT Datenbank, deren Einträge in einigen Fällen bis zu 20 (statt 8) Buchstaben und 9 (statt 5) PAWs aufweisen. Dies erhöht das Risiko von Segmentierungs- und Erkennungsfehlern, aber ebenfalls die Effektivität einer Fehlerkorrektur mittels Vokabular.

Die Diagramme in Abbildung 4.6 zeigen, dass die meisten Wörter gewöhnlicher Texte aus ein bis zwei PAWs bestehen; mehr als vier PAWs enthalten Wörter nur in Ausnahmefällen. Werden in einem handschriftlichen Wort 4 oder 5 PAWs detektiert, lässt sich die Anzahl der potentiellen Vokabeln stark eingrenzen, da die meisten Vokabeln lediglich 1-3 PAWs aufweisen. Entsprechend der Informationstheorie enthalten jedoch grade seltene Wörter viel Information, und sind daher z.B. besonders gut geeignet, um auf den Inhalt eines Textes zu schließen. Somit ermöglicht die Analyse der PAW-Anzahl eine Selektion potentieller Schlüsselwörter schon vor der

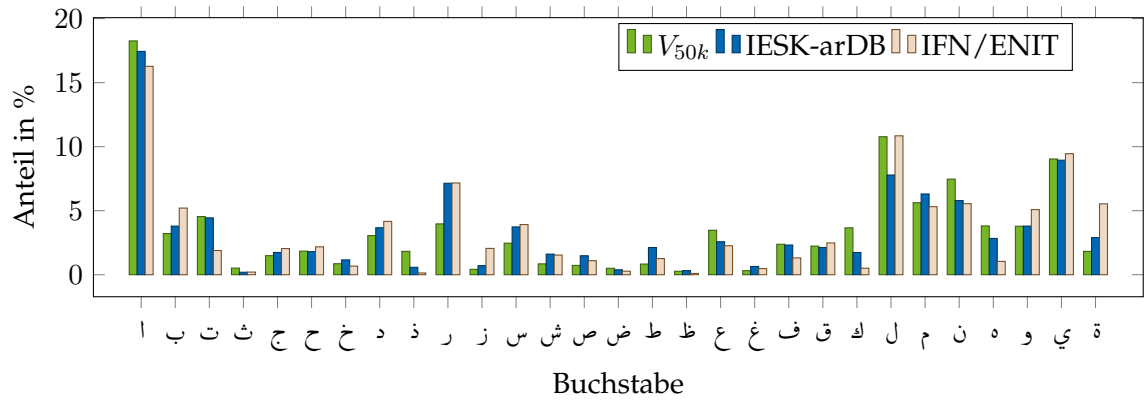


Abbildung 4.7. Verteilung der Buchstabenklassen verschiedener Datenbanken (alle Buchstaben-Positionen).

Handschrifterkennung.

4.3.1 Analyse der Buchstabenverteilung

Ein Problem, das unter anderem die Verfügbarkeit von Proben für alle Buchstaben betrifft, ist die unterschiedliche Häufigkeit, mit welcher die 29 Buchstaben des arabischen Alphabets (ا bis ة) in einem Text oder einer Wortdatenbank vorkommen. Wie Abbildung 4.7 zeigt, ist fast jeder fünfte Buchstabe in einem Wort des Vokabulars $\mathcal{V}_{50k}$ oder der IESK-arDB und IFN\ENIT Datenbank ein ا. Im Gegensatz dazu sind viele Buchstaben sehr selten (z.B. ث). Weil arabische Glyphen zudem stark von der Position des entsprechenden Buchstaben abhängen, fällt die Verteilung der Buchstabenklassen (die sich aus dem Buchstaben und der Position ergibt) noch stärker ins Gewicht. Aus Abbildung 4.8 geht hervor, dass einige Buchstaben, wie ل, meist in beginnender, andere hingegen öfter in mittiger Position auftreten. Da beispielsweise die generell raren Buchstaben س und ج besonders selten in der endenden und isolierten Form auftreten, gibt es in den Datenbanken nur sehr wenige Proben für die entsprechenden Glyphen. In der IFN\ENIT- sowie der IESK-arDB-Datenbank fehlen einige Buchstabenklassen sogar gänzlich! Daher ist zur Erstellung der ASMs die explizite Buchstabendatenbank IESK-arDB$_{\text{OnlineLetter}}$, mit in etwa gleichviel Proben für alle Buchstabenklassen, vorzuziehen.

Beim Trainieren von SVMs ist die geringe Anzahl von Proben für die seltenen Buchstabenklassen prinzipiell weniger dramatisch. Bezüglich Buchstabenerkennung besteht hier hauptsächlich eine erhöhte Gefahr einer False-Negative Klassifikation seltener Buchstabenklassen; jedoch eben nur in Fällen, bei denen ein Wort auch entsprechend seltene Buchstaben enthält. Zur Optimierung der SVM Parameter

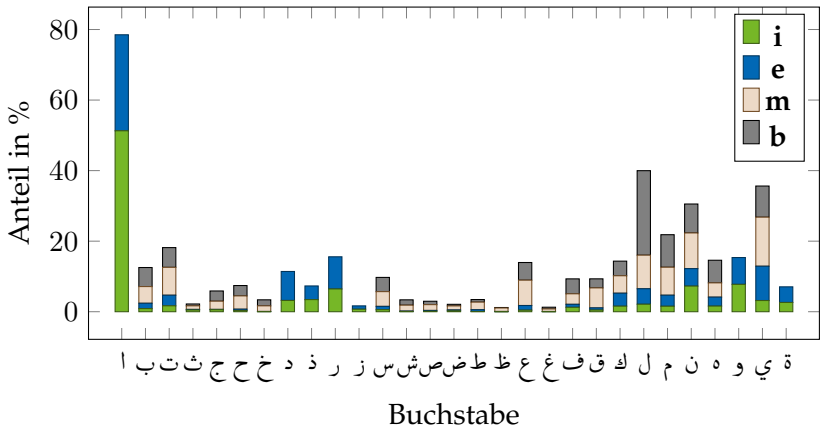


Abbildung 4.8. Verteilung der Buchstabenklassen des Vokabulars $\mathcal{V}_{50k}$ für die Buchstaben in isolierter, endender, mittiger und beginnender Position.

ist jedoch eine Kreuzvalidierung erforderlich, weshalb für jede Klasse wenigstens $kfold \geqq 5$ Proben vorliegen müssen. Dies ist bei Verwendung der IESK-arDB$_{\text{Letter}}$ nicht gewährleistet.

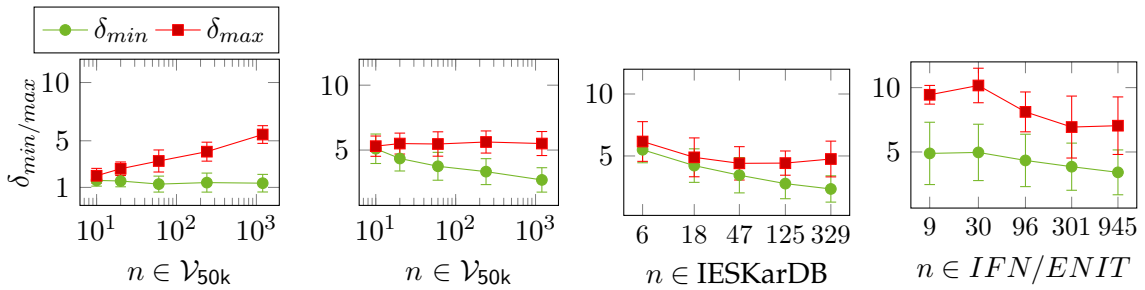
4.3.2 Distanzmaß zwischen erkannter Sequenz w und Vokabel $\mathbf{w}_{\mathcal{V}_i}$

Die Güte der Korrektur fehlerhaft erkannter Wörter – welche in Kapitel 6 vorgestellt wird – hängt maßgeblich von dem hierzu verwendeten Vokabular $\mathcal{V}$ ab. Einen offensichtlichen Einfluss übt die Anzahl der Vokabeln $n(\mathcal{V})$ aus. Je umfangreicher das Vokabular, desto höher ist die Wahrscheinlichkeit, dass mehrere Vokabeln die gleiche Ähnlichkeit zur erkannten Buchstabensequenz $\mathbf{w}_r$ aufweisen. Ein weit verbreitetes Maß für diese Ähnlichkeit ist die Levenstein Distanz δ, welche die Anzahl an Operationen – wie Änderung, Hinzufügen oder Löschen eines Buchstaben – angibt, die zur Umformung einer Zeichenkette in eine andere nötig sind [94].

Die durch die minimale Levenstein Distanz δ_{min} angegeben Ähnlichkeit der einzelnen Einträge des Vokabulars ist ein entscheidender Indikator für mögliche Verwechslungen. Abbildung 4.9 gibt für verschiedene Teilmengen unterschiedlicher Vokabulare $\delta_{min} = \arg\min_{\delta} \delta(\mathbf{w}_{\mathcal{V}_i},\mathbf{w}_{\mathcal{V}_j})$ an, wobei $(\mathbf{w}_{\mathcal{V}_i},\mathbf{w}_{\mathcal{V}_j})$ ein Wortpaar aus einer Vokabel und der jeweils nächst ähnlichsten Vokabel desselben Vokabulars bildet. Ist der Erkennungsfehler $\delta(\mathbf{w}_r, \mathbf{w}_t)$ zwischen dem erkannten und tatsächlichem Wort $\mathbf{w}_t$ größer oder gleich δ_{min}, so ist eine Verwechslung mit $\mathbf{w}_{\mathcal{V}_j}$ oder einer anderen Vokabel sehr wahrscheinlich. Somit ist $\bar{\delta}_{min}$ ein Maß dafür, wie viele Fehler pro Wort durchschnittlich auftreten dürfen, bevor bei der Zuordnung einer erkannten Sequenz zu einer Vokabel die akute Gefahr einer Verwechslung besteht. So lässt

Quelle	$\delta_{min}=1$	$\delta_{min}=2$	$\delta_{min}=3$	$\delta_{min}=4$
IFN	ةنَايرَا , ةنَايرف	روصنم يديس , رصن يديس	ةروصنلمَا , يرتسنلمَا	لمَاَا لزنم , رونلَا لزنم
IFN	ةبَارغلَا , طبَارغلَا	ةَاورب , ةنوردب	ةبقروب ريب , ةبيقروب يح	يَابلَا ريب , ةميلح ريب
IFN	ةنَايرف , ةنَانرف	فشرلخَا , مشلخَا	يَابلَا ريب , بيطلَا ريب	يتَاوتلَا يديس , ينَاوسلَا يدَاو
$\mathcal{V}_{5_k}$	لهذَا , ولهذَا	قَانون , مجنون	الَايَام , الَاحوَال	قَانون , يحتَاجون
$\mathcal{V}_{5_k}$	بلي , بيلي	مجنون , قَانون	الَامل , القنَابل	الوحيد , بَالتَاكيد
$\mathcal{V}_{5_k}$	عند , عندك	هنَاك , هنري	بين , صحيح	–

Tabelle 4.2. Beispiele für ähnlichste Wortpaare mit unterschiedlicher Levenstein Distanz δ_{min} innerhalb der IFN/ENIT Datenbank und der 5.000 häufigsten arabischen Wörter.



(a)häufigste (b)seltenste (c) (d)

Abbildung 4.9. Levenstein Distanzen innerhalb eines Vokabulars der (a) n häufigsten und (b) n seltensten Vokabeln aus aus $\mathcal{V}_{50k}$ sowie zum Vergleich n Vokabeln aus der (c) IESK-arDB und (d) IFN/ENIT Datenbank. Entscheidend ist das Maß δ_{min}, welches die durchschnittliche Levenstein Distanz zwischen je einer Vokabel $\mathbf{w}_{\mathcal{V}_i}$ und der zu dieser ähnlichsten Vokabel $\mathbf{w}_{\mathcal{V}_j}|j \neq i$ angibt (bzw. der unähnlichsten Vokabel für δ_{max}).

sich für eine konkrete Anwendung ein Kompromiss zwischen der Größe des verwendeten Vokabulars (bzw. der Anzahl ähnlicher Wörter) und der nötigen Güte der Fehlerkorrektur finden. Tabelle 4.2 zeigt einige Beispiele für Wortpaare $(\mathbf{w}_{\mathcal{V}_i},\mathbf{w}_{\mathcal{V}_j})$ mit unterschiedlichem δ_{min}.

Das Maß δ_{min} hängt nicht nur von der Anzahl der Vokabeln ab. Werden – wie in Abbildung 4.9(a) – nur die häufigsten arabischen Wörter verwendet, so gilt sogar bei sehr wenigen Einträgen $\bar{\delta}_{min} < 2$. Es besteht also eine hohe Verwechslungsgefahr. Dies liegt daran, dass die meisten sehr häufigen Vokabeln aus nur 3 Buchstaben bestehen, welche auch die Obergrenze von δ_{min} angeben (siehe Tabelle 4.1). Seltene Vokabeln unterscheiden sich durchschnittlich deutlich stärker voneinander. Wie Abbildung 4.9(b) zeigt, ist die antiproportionale Relation von δ_{min} und der Vokabulargröße hier deutlich stärker ausgeprägt, weshalb sich die Fehlerkorrektur durch

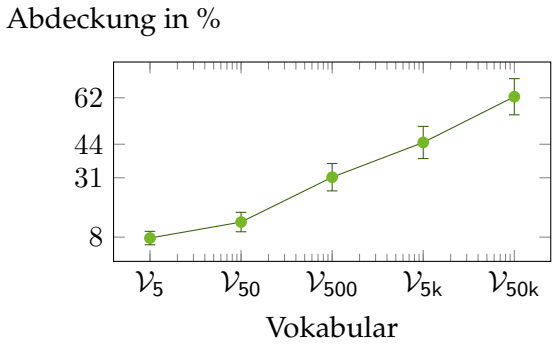


Abbildung 4.10. Darstellung des durchschnittlichen Anteils bekannter Wörter pro Textseite in Abhängigkeit des genutzten Vokabulars $\mathcal{V} \subseteq \mathcal{V}_{50k}$. Bekannte Wörter im Text sind diejenigen, die ohne Abweichung auch im Vokabular vorliegen (Buchstaben mit Supplemente wie ه̣ werden durch ihre Normalform ه ersetzt). Die Textseiten bestehend aus den Grundwahrheiten der IESK-arDB$_{\text{Hist}}$, die Dokumente aus dem 14. Jahrhundert enthalten.

eine Reduzierung der verwendeten Vokabeln wesentlich verbessern lässt. Das relativ kompakte Vokabular der IESK-arDB verhält sich, entsprechend Abbildung 4.9(c), ähnlich wie die seltensten Einträge aus $\mathcal{V}_{50k}$. Noch stärker ausgeprägt ist dieses Verhalten bei der IFN\ENIT Datenbank. Um eine sehr robuste Fehlerkorrektur zu erzielen, empfiehlt es sich daher, ein Vokabular zu benutzen, das wenige, eher seltene und vorzugsweise längere Wörter enthält.

Methoden zur Fehlererkennung auf Grundlage von Vokabularen verschiedener Größe werden in Kapitel 8 untersucht.

Abdeckung von Texten durch ein Vokabular. Bei freien Texten ist die Anzahl potentiell enthaltenen Wörtern nicht limitierbar und lässt sich daher nicht gänzlich durch ein Vokabular abdecken. Abbildung 4.10 zeigt jedoch, dass $\mathcal{V}_{50k}$ durchschnittlich bereits 60% aller Wörter einer Seite arabischen Text aus dem 14. Jahrhundert abdeckt. Auch mit einem kleinen Vokabular der 500 häufigsten arabischen Wörter werden noch über 30% Abdeckung erreicht.

4.4 Synthese handschriftlicher Wortdatenbanken

Datenbanken mit handgeschriebenen arabischen Textseiten sind essenziell, um zu erforschen, welche Möglichkeiten zur Analyse oder Übersetzung entsprechender Dokumente bestehen. Im Rahmen dieser Dissertation steht jedoch die Erkennung einzelner Wörter im Vordergrund. Daher werden die Pseudotexte nicht nur eingesetzt, um ganze Texteseiten zu generieren, sondern auch um Wortdatenbanken zu

erstellen, welche das Ergebnis einer optimalen Segmentierung der synthetisierten handschriftlichen Texte in Wörter simulieren. Um mit dem vorgestelltem Synthesesystem umfangreiche Datenbanken erzeugen zu können, werden zudem folgende Funktionen eingesetzt:

- Einlesen des Vokabulars (Wörter oder Texte) der Datenbank
- Einlesen und Verwenden mehrerer Einstellungsdateien
- Generierung von Grundwahrheiten zu jeder synthetisierten Probe

Das Vokabular, bzw. der zu synthetisierende Text, liegt hierbei in einer im UTF8 Format kodierten Textdatei vor und lässt sich durch die Benutzeroberfläche entweder als zusammenhängender Text oder als Liste von Vokabeln aufrufen. Die weiteren Punkte werden in den folgenden Abschnitten näher erläutert.

Gegenwärtig existiert noch keine frei verfügbare Version des vorgestellte Synthesesystems. Um mit diesem dennoch einen Beitrag zur Forschung zu leisten, wurde die IESK-arDB-Datenbank um synthetische Wort- und Textproben erweitert, die weiterhin als IESK-arDB$_{\mathsf{SynText}}$ und IESK-arDB$_{\mathsf{SynWords}}$ bezeichnet werden. In diesem Abschnitt werden diese synthetisierten Datenbanken anhand einiger Beispiele vorgestellt und deren Aufbau näher erläutert. Einige Proben der IESK-arDB$_{\mathsf{SynWords}}$ sind in Abbildung 4.11 dargestellt (zum Vergleich zeigt Abbildung 4.12 dieselben Wörter in realer Handschrift).

4.4.1 Erweiterung der IESK-arDB durch Wortsynthese

Bezüglich der meisten in dieser Arbeit behandelten Methoden ist die IESK-arDB$_{\mathsf{SynWords}}$ von großem Interesse. Der aktuelle Schwerpunkt der in der Arbeitsgruppe behandelten Schrifterkennung ist zwar die Erkennung einzelner arabischer Wörter, jedoch mit der Intention, in zukünftigen Projekten zur Analyse von umfangreicheren Texten eingesetzt zu werden. Die in der IESK-arDB$_{\mathsf{SynWords}}$ enthaltenen Wörter sollen daher die Eigenschaften solcher Texte repräsentieren, anstatt eine beliebige oder für eine einzelne, spezifische Anwendung konzipierte Liste von Wörtern darzustellen. Außerdem erleichtert es der hohe Detailgrad der zugehörigen Grundwahrheiten die Vorverarbeitung, Segmentierung, Erkennung und Fehlerkorrektur zu validieren und zu testen.

Die IESK-arDB$_{\mathsf{SynWords}}$ enthält 20.000 Proben synthetisierter arabischer Wörter, welche auf Pseudo-Text basieren, der auf der Grundlage des Vokabulars $\mathcal{V}_{5\mathsf{k}} \subset \mathcal{V}_{50\mathsf{k}}$

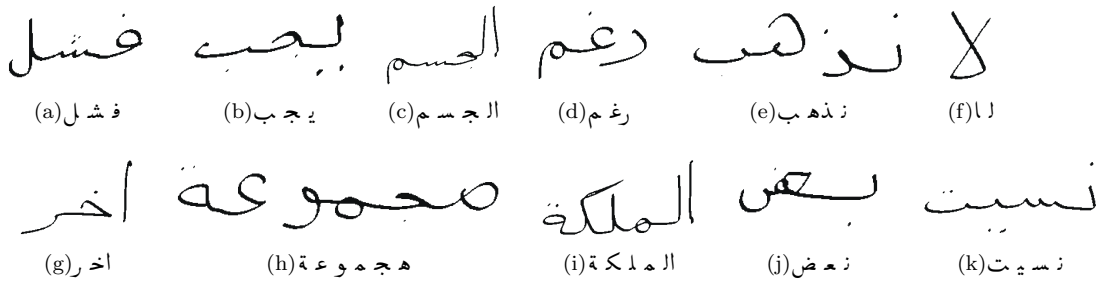


Abbildung 4.11. Beispielsynthesen aus der Datenbankenerweiterung IESK-arDB$_{\text{SynWords}}$.

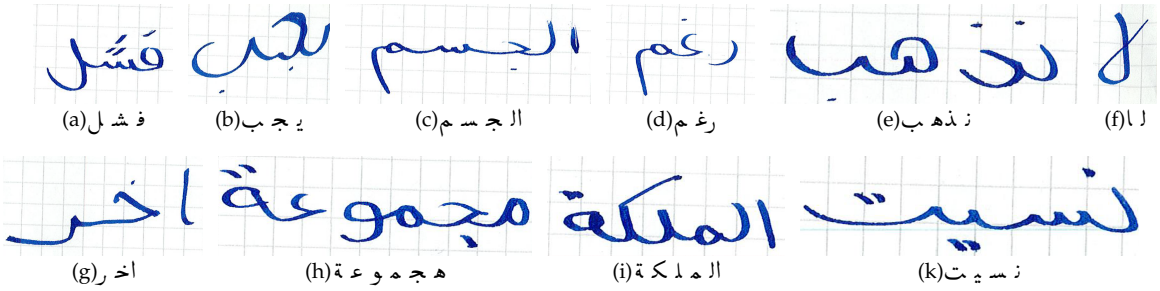


Abbildung 4.12. Beispiel für reale Handschriften zu Wörtern aus dem Vokabular der IESK-arDB$_{\text{SynWords}}$.

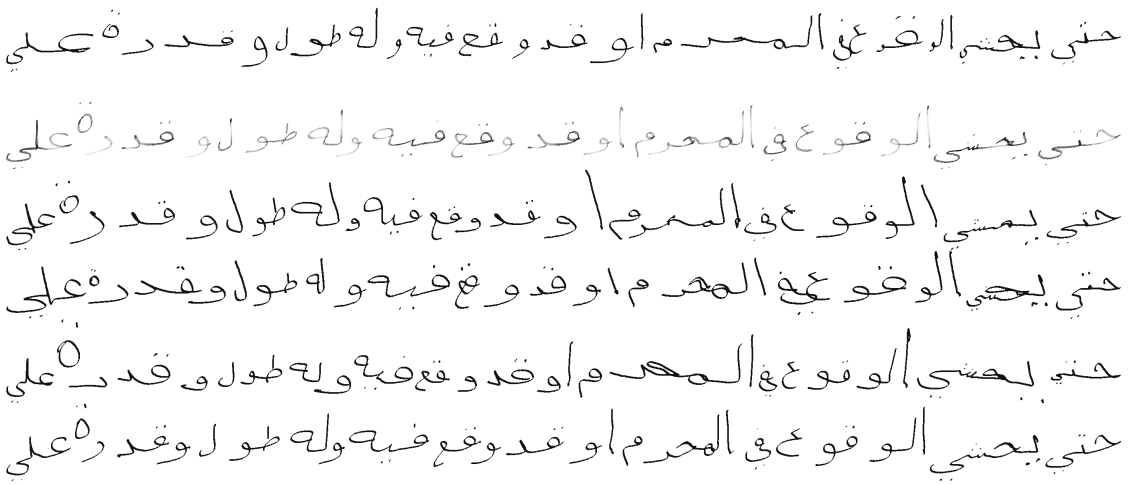


Abbildung 4.13. Mit unterschiedlichen Einstellungen generierte Synthesen desselben Schreibers.

erzeugt wurde. Dieses Vokabular mittlerer Größe bildet auch die Grundlage vieler Experimente in Kapitel 8. Jede Probe besteht aus einem Bild im PNG Format und einer zugehörigen XML-Datei, welche die Grundwahrheit (GT) enthält. Die GT umfassen zudem die Trajektorien der Buchstaben, so dass sich die Datenbank auch zur Validierung von online Methoden eignet.

Die aktuelle Version des Syntheseverfahrens verfügt bereits über Strategien, um Kollisionen von Textzeilen und benachbarter Wörter oder PAWs zu beheben. Vereinzelt auftretende Kollisionen zwischen Buchstaben innerhalb eines PAW– wie in Abbildung 4.11 (j) – werden jedoch lediglich detektiert und zur Dokumentation des Herausforderungsgrades in den GT vermerkt. Sie haben oft zur Folge, dass eine korrekte Segmentierung und Buchstabenerkennung nicht mehr möglich und eine Fehlerkorrektur daher zwingend notwendig ist. Da vergleichbare Mängel allerdings auch in realer arabischen Handschrift beobachtet wurden, werden die betroffenen Proben nicht von den Experimenten in Kapitel 8 ausgeschlossen.

Abbildung 4.13 zeigt mehrere Synthesen desselben Satzes, welche vom selben Schreiber stammen. Hierbei wurden mehrere Einstellungsdateien zur Erhöhung der Varianz innerhalb der Datenbank verwendet.

4.4.2 Grundwahrheiten

Zu allen synthetisierten Bilder werden automatisch XML-Dateien mit den Grundwahrheiten (GT) generiert. Abbildung 4.14 visualisiert die wichtigsten Angaben der GT zu zwei Proben. Die GT untergliedern sich in die Unterbäume `BasicData` und `OnlineData`, wobei letzterer die Trajektorien der einzelnen Buchstaben enthält. Obwohl die Trajektorien nicht für offline Anwendungen konzipiert wurden, können sie auch hier zur Validierung verwendet werden, beispielsweise um unerwünschte Änderungen der Topologie durch Methoden der Vorverarbeitung zu evaluieren. Da die Trajektorien jedoch eine erhebliche Datenmenge ausmachen, werden sie von den für die Validierung von offline Methoden zumeist hinreichenden `BasicData` separiert. Ein Beispiel einer solchen xml-Datei findet sich in Anhang A.1.

`BasicData` teilt sich wiederum in globale und lokale GT auf. Eine Übersicht der globalen GT, welche die wesentlichen Eigenschaften des zugehörigen Wortes beschreiben, findet sich in Tabelle 4.3. Zusätzliche lokale Angaben werden für jeden Textzeile, jedes PAW und jeden Buchstaben oder diakritisches Zeichen hierarchisch gespeichert. Alle Elemente besitzen eindeutige IDs, mit welchen z.B. ein Buchstabe

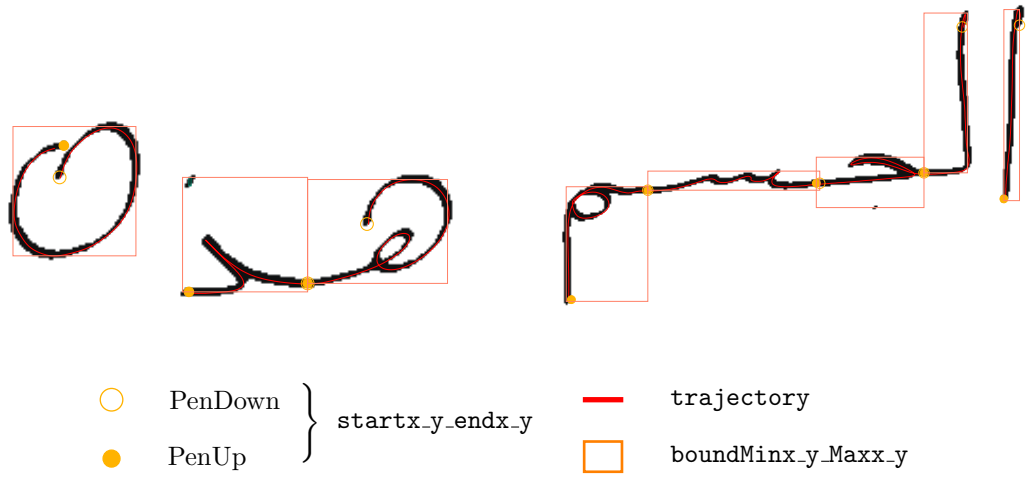


Abbildung 4.14. Visualisierung der wesentlichen Komponenten der Grundwahrheiten zu den synthetisierter Wörtern هذه und الجسم.

Variabel	Beispiel	Bedeutung
`Intersection`	0, 0,	1 falls Buchstabe $i > 0$ seinen Vorgänger kreuzt
`letterNames`	He The He	Name der Buchstaben in ASCI
`unicode`	هذه	Wort als Unicode(UTF8)
`Baseline`	63_98_	(y_1,y_2) Basislinienhöhe
`nDiacritics`	1	Anzahl der diakr. Z.
`NLetters`	3	Anzahl der Buchstaben
`NPAWs`	2	Anzahl der PAWs

Tabelle 4.3. `BasicData`

Variabel	Beispiel	Bedeutung
`name`	He	Name des Buchstaben in ASCI
`nDiacriticParts`	0	Anzahl der Fragmente des diakritischen Zeichens
`DiacriticID`	-1	ID Nr. des diakr. Zeichens (-1 falls `nDiacriticParts` $= 0$)
`startx_y_endx_y`	191_70_163_98	x,y-*PenDown*, x,y-*PenUp*
`boundMinx_y_Maxx_y`	163_49_229_98	$(x,y),(x,y)$ – umgebendes Rechteck von `letter0`

Tabelle 4.4. `letter0`

```
1 COM: Ground Truth file with the IFN/ENIT Style
2 X_Y: 269 201
3 BDR: begin data record
4 LBL: ZIP:3015;AW1:الجسم;AW2:aaA|laB|jaM|seM|maE|;QUA:YB1;ADD:P2
5 CHA: 5
6 BLN: 113,143
7 EDR: end of data record
```

Abbildung 4.15. Beispiel für eine zusätzliche Grundwahrheit im IFN/ENIT Stil.

auf sein zugehöriges diakritische Zeichen verweist. Es gilt die Hierarchie: `line` → `PAW` → `letter` → `diacritic`. Für diakritische Zeichen wird zusätzlich das umgebende Rechteck vermerkt, für Buchstaben weitere Informationen, wie in Tabelle 4.4 am Beispiel des ersten Buchstabens dargestellt wird.

Grundwahrheiten im IFN/ENIT Format. Da die IFN/ENIT Datenbank weit verbreitet ist, wurden zusätzlich Grundwahrheiten (GT) im entsprechendem Format generiert. Ein Beispiel ist in Abbildung 4.15 dargestellt. Diese GT enthalten keine Informationen über die genauen Positionen der Buchstaben, PAWs oder diakritische Zeichen, ermöglichen jedoch den sofortigen Einsatz der synthetischen Wortproben zum Trainieren und Testen von Systemen, welche bereits die IFN/ENIT Datenbank verwenden. Die ZIP ID wurde durch die Position des Wortes innerhalb der Liste der 50.000 häufigsten Wörter ersetzt. Die Basislinien wurden durch Lineare Regression der *PenDow*n Punkte von Buchstaben in mittiger und endender Form berechnet. Existieren weniger als zwei solcher Punkte, wird stattdessen die durchschnittliche Relation der Buchstabenklassen zur Basislinie verwendet.

4.5 Synthese handschriftlicher Textdatenbanken

Mit zunehmendem Erfolg bei der Erkennung einzelner arabischer Wörter werden immer häufiger Verfahren zur Erkennung komplexerer Handschrift, wie zusammenhängender Texte, erforscht. Hierzu gehören auch Verfahren zur Segmentierung von Textzeilen, oder die – im Fall arabischer Schrift herausfordernde – Segmentierung der Textzeilen in einzelne Wörter. Zwar existieren verschiedene Datenbanken mit handschriftlichen arabischen Zeilen oder Paragraphen, wie z.B. die KHATT-Datenbank [61] oder die historischen Dokumente der IESK-arDB$_{\text{Hist}}$, allerdings enthalten diese nur auf das Wesentlichste reduzierte Grundwahrheiten (in erster

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

Abbildung 4.16. Beispiele für synthetisierte Textseiten. (a-e) Komplexe Rendering Methode simuliert die aufgrund des Stiftes und der Geschwindigkeit erzeugte halbdeckende Pigmentierung, welche die verwendete Hintergrundtextur durchscheinen lässt. (f) Einfache, binäre Textseitensynthese mit horizontaler Basislinie.

Linie den Unicode). Präzise Zuordnungen der umgebenden Rechtecke aller Wörter, PAWs, Buchstaben und diakritischen Zeichen zu einer Zeile – wie sie bei der Datensynthese automatisch erstellt werden – sind jedoch hilfreich für die genaue Validierung und Evaluierung von Verfahren zur Zeilensegmentierung oder zur Vorverarbeitung (um beispielsweise zu überprüfen, wie Änderungen der verwendeten Binarisierungsmethode die Detektionsgüte von PAWs oder diakritischer Zeichen beeinflussen). Zudem ermöglichen die Synthesen eine präzise Einschätzungen des Herausforderungsgrades aufgrund der kontrollierbaren Intensität der Basislinienkrümmung und -schräge, des Zeilenabstandes und anderer das Schriftbild

betreffender Eigenschaften. Daher wurde das Synthesesystem im Rahmen dieser Dissertation auch für die Generierung von Datenbanken von Textseiten, wie in Abbildung 4.16, eingesetzt, um zukünftige Untersuchungen zu unterstützen. Es wurden bereits 712 synthetische Proben zu 178 unterschiedlichen Textseiten als Ergänzung der IESK-arDB$_{\text{Hist}}$ veröffentlicht (www.iesk-ardb.ovgu.de).

Eine genaue Auflistung aller veröffentlichten Bestandteile der IESK-arDB zeigt Tabelle 4.5.

Tabelle 4.5. Übersicht aller aktuellen Bestandteile der IESK-arDB.

Name	Datei	Erläuterung
IESK-arDB$_{\text{Hist}}$	m_DB.rar	Datenbank mit 285 historischen arabischer Textseiten.
IESK-arDB	words_DB.rar	Datenbank mit 2.540 handschriftlichen Proben zu 323 verschiedenen arabischen Wörtern von insgesamt 22 verschiedenen Schreibern.
IESK-arDB$_{\text{Letter}}$	DbChar.rar	5.708 Proben von Buchstaben (die aus der IESK-arDB extrahiert wurden).
IESK-arDB$_{\text{OnlineLetter}}$	ASMs.rar	Buchstabendatenbank auf Grundlage von 28.046 online Glyphen. Besteht aus XML Dateien, welche ASMs sowie normierte Trajektorien der Buchstaben enthalten. Aus letzteren wurden offline Proben zum Trainieren der Klassifikatoren gerendert.
IESK-arDB$_{\text{SynWords}}$	Synthetic-Database.rar	Synthetische Datenbank 20.000 Bildern zufällig (entsprechend ihrer Auftrittswahrscheinlichkeit, mit Zurücklegen) ausgewählter arabischer Wörter aus dem Vokabular $\mathcal{V}_{5k}$. Enthält zudem 712 synthetische Textseiten.
IESK-arDB$_{\text{SynWords}}$	SyntheticSamples-WithIFN-CompatibleGT.rar	Datenbanken mit verschiedenen Vokabularen, die jeweils in einer zusätzlichen mit der IFN/ENIT Datenbank kompatible Versionen vorliegen. Umfasst die 1.000 häufigsten und die 1.000 am wenigsten häufigen Vokabeln aus $\mathcal{V}_{50k}$ sowie das Vokabular der IESK-arDB und der IFN/ENIT-Datenbank.
IESK-arDB$_{\text{SynWords}}$	10k.rar	40.000 synthetisierte Bilder zu allen Wörtern des Vokabulars $\mathcal{V}_{10k}$ (nach Auftrittswahrscheinlichkeit abnehmend angeordnet).

4.6 Zusammenfassung

In diesem Kapitel wurden verschiedene herkömmliche sowie synthetische Datenbanken arabischer Handschrift vorgestellt. Zudem wurde erläutert, wie auf Grundlage des Vokabulars der 50.000 häufigsten arabischen Wörter – sowie der Vokabulare der IESK-arDB und IFN\ENIT Datenbank – eine synthetische Datenbank mit arabischen Wörtern und Textseiten generiert wurde. Weiterhin wurden die zugehörigen Grundwahrheiten beschrieben.

Durch eine Analyse der herkömmlichen und synthetischen Datenbanken wurde aufgezeigt, wie stark sich das Vokabular der für die Briefkopferkennung konzipierten IFN\ENIT Datenbank von dem allgemeiner arabischer Texte unterscheidet und welche Bedeutung dies für die in den nächsten Kapiteln beschriebene automatische Erkennung arabischer Handschrift hat.

KAPITEL 5

Vorverarbeitung und Segmentierung

Um komplexe Aufgaben, wie die Erkennung handschriftlicher Wörter, durchführen zu können, bedarf es zunächst der in Abbildung 5.1 dargestellten Vorverarbeitungs- sowie Segmentierungsschritte. Die Vorverarbeitung dient dazu, irrelevante Information, wie Rauschen, zu entfernen. Bei Handschrifterkennung sind auch Farbe und Textur in vernachlässigbar, sofern sie keinen Bezug zum Inhalt eines Dokumentes aufweisen. Methoden der Segmentierung wiederum können eingesetzt werden, um ein Problem, wie die Erkennung eines handschriftlichen Textes, in kleinere Probleme, wie die Erkennung einzelner Zeilen oder Wörter, aufzuteilen. In Abschnitt 5.2 bis 5.4 werden dahingehend Segmentierungsansätze vorgestellt, die für das in dieser Dissertation vorgeschlagene System zur Handschrifterkennung geeignet sind.

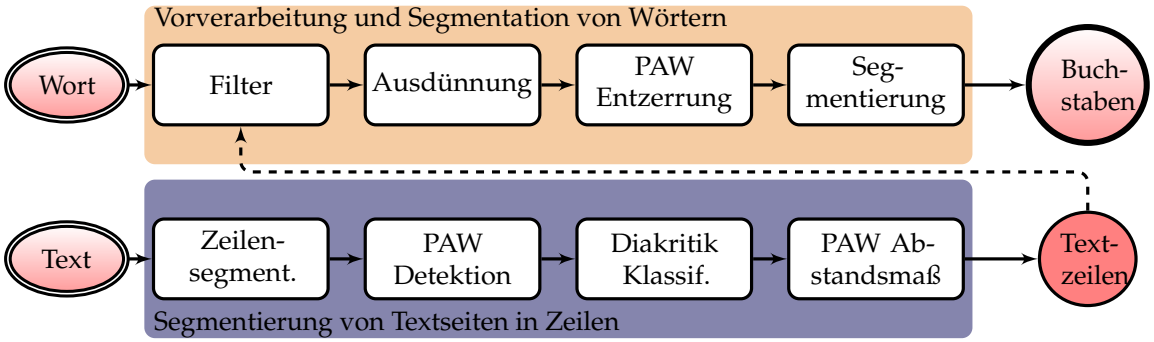


Abbildung 5.1. Übersicht zur Vorverarbeitung und zu den Segmentierungsverfahren.

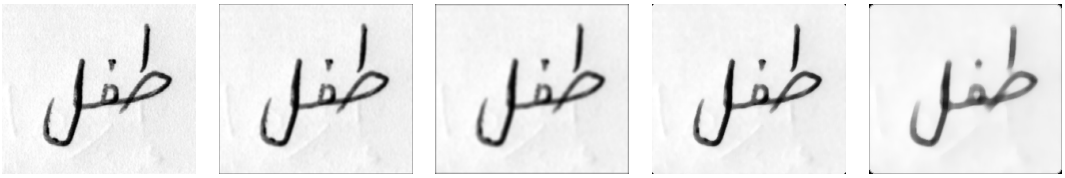
5.1 Vorverarbeitung

Zu Beginn einer Bildverarbeitungspipeline stehen üblicherweise gängige Filteroperationen, die zur Unterdrückung von Rauschen und Schärfen von Objektkanten eingesetzt werden. Hier werden Gauss- und Medianfilter mit Filtermasken der Größe $2n+1 \times 2n+1 \,|n \in \mathbb{N}$ auf ihre Eignung untersucht. Entscheidendes Kriterium ist hierbei, inwiefern Fehler der nachfolgenden Segmentierungsschritte, aber auch der Buchstabenklassifikation, minimiert werden.

Zur Rauschunterdrückung hat sich bei den meisten untersuchten Proben eine Gaußsche 3×3 Filtermaske als geeignet erwiesen (siehe Abbildung 5.2 (b)). Die Anwendung eines Medianfilters hebt Kanten hervor, und reduziert bezüglich Handschrifterkennung irrelevante Texturen im optimalem Fall zu Flächen. Daher lässt sich bei Proben, die mit Schreibutensilien wie Kugelschreibern erzeugt wurden und eine unregelmäßige Pigmentation aufweisen, durch den Einsatz von Medianfiltern ein klareres Schriftbild erzielen. Wie Abbildung 5.2 (e) allerdings erahnen lässt, führt eine zu große Filtermaske unter Umständen dazu, dass sich kleinere Schleifen schließen oder benachbarter Schriftbestandteile verschmelzen. Ähnliche Effekte entstehen auch auf natürliche Weise, etwa durch die Wahl eines Stiftes mit hoher Linienstärke. Für Handschrift mit ähnlichen Eigenschaften[1] wie der IESK-arDB ist (bei 300dpi) aber dennoch eine mäßig große Filtermaske zu empfehlen (3×3 oder 5×5).

5.1.1 Binarisation

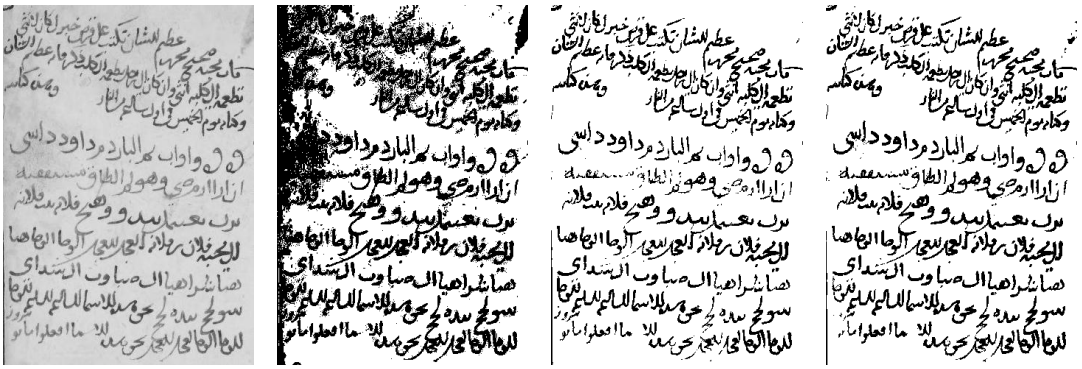
Im Anschluss an die Rauschunterdrückung erfolgt eine Segmentierung des Hintergrundes und des Vordergrundes, welcher in diesem Fall durch die Schrift gegeben



(a)Original (b)G3 $\times$ 3 (c)G5 $\times$ 5 (d)M5 $\times$ 5 (e)G3 $\times$ 3M11 $\times$ 11

Abbildung 5.2. Resultate von Gauss- (G) und Median (M) Filter mit unterschiedlich großen Filtermasken.

[1] Die Eigenschaften ergeben sich aus den verwendeten Stiften (hier Kugelschreibern), der Größe der Schrift und der Auflösung der Proben.



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

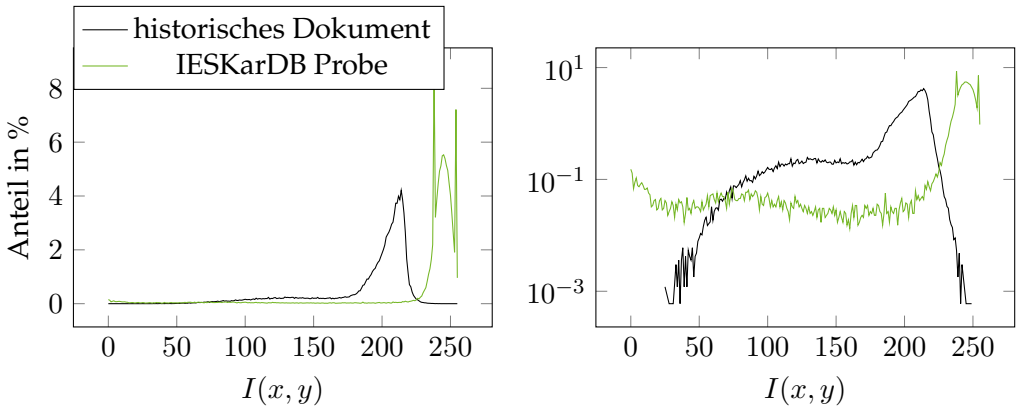
Abbildung 5.3. Binarisierung von Handschrift. (a) Grauwertbild. (b-c) Binarisierte Bilder: (b) Mit geschätztem Schwellwert von 200. (c) Mit dem mittels Histogramm und Rückkopplung mit der Validierung der in Abschnitt 5.4.3 beschriebenen Zeilensegmentierung optimierten Schwellwert 160. (d) Adaptiver Schwellwert.

ist. Hierzu wird ein Binarisierungsverfahren eingesetzt, welches im einfachsten Fall alle Bildpunkte, deren Grauwerte $I(x,y)$ einen bestimmten Schwellwert t_h unterschreiten, dem Vordergrund zuweist (siehe Abbildung 5.3 (b)). Da viele handschriftliche Dokumente nicht mit schwarzer Tinte, sondern mit Kugelschreibern oder Bleistiften erstellt wurden, lässt sich jedoch kein allgemein gültiger Schwellwert festsetzen. Daher empfiehlt es sich, entsprechend der Otsu Methode eine Analyse des Histogramms durchzuführen, um den optimalen Schwellwert abzuschätzen [75].

Da bei handschriftlichen Dokumenten ca. 90% aller Bildpunkte zum Hintergrund gehören, ist hier, wie Abbildung 5.4 zeigt, lediglich die Hintergrund-Klasse gut ausgeprägt. Indem die Hintergrund-Klasse als Normalverteilung $\mathcal{N}_h(\mu_h,\sigma_h)$ beschrieben wird, lässt sich der Schwellwert nun dynamisch in Abhängigkeit des Histogramms bestimmen. Um den in Abbildung 5.3 (c) eingesetzten Schwellwert t_h zu optimieren, wird die ermittelte Standardabweichung der Hintergrund-Klasse eingesetzt:

$$t_h = \mu_h - c_h \cdot \sigma_h \; | c_h \in [0,3]. \tag{5.1}$$

Ist c_h zu hoch gewählt, fragmentieren schlecht pigmentierte Schriftbestandteile. Bei zu niedrigem Wert besteht jedoch die Gefahr, dass dunkle Bereiche des Hintergrundes – wie in Abbildung 5.3 (b) – als Schrift klassifiziert werden. Da vordergründig der Erfolg der an die Vorverarbeitung anschließenden Methoden von Interesse ist, wird der Parameter c_h auf Grundlage der Genauigkeit der in Abschnitt 5.4.3 beschriebenen Zeilensegmentierung optimiert.



(a) (b)

Abbildung 5.4. Histogramm einer modernen ((Abbildung 5.2 (a)) sowie einer historischen Handschriftprobe (Abbildung 5.3 (a)) in (a) normaler und (b) logarithmischer Darstellung.

Auch mit optimalem t_h stößt die Otsu Methode bei älteren, stark degradierten Dokumenten an ihre Grenzen. In diesem Fall weicht die Helligkeit des Hintergrundes nicht nur stark vom idealen Weiß ab, vielmehr treten verstärkt regionale Abweichungen z.B. durch Stockflecken auf. In diesem Fall empfiehlt sich der Einsatz eines adaptiven Schwellwertes, der aufgrund eines regionalen Histogramms für jeden Bildpunkt berechnet wird [72]. Wie in Abbildung 5.3 (c) zu sehen ist, wird so eine gute aber – bei problematischen (historischen) Proben – nicht fehlerfreie Binarisierung erzielt.

Neben den beschriebenen automatischen Binarisierungsmethoden existieren weitere Ansätze, die manuelle Initialisierungen erfordern [90]. Da die folgenden Verfahren jedoch eine automatische Binarisierung vorauszusetzen, wird hierfür ein adaptives Schwellwertverfahren eingesetzt.

5.1.2 Ausdünnung der Schrift

Die primären Informationen der Handschrift sind in den aus den Trajektorien resultierenden Schriftzügen enthalten. Die Breite der Linien induziert lediglich den ausgeübten Druck und die Schreibgeschwindigkeit und hängt zudem stark vom Material oder der Schriftgröße ab. Daher ist eine Normierung der Linienbreite in vielen Fällen sinnvoll. Insbesondere zur Merkmalsextraktion ist dieser Vorgang oftmals sogar notwendig, um z.B. KeyFeatures (KF) zu detektieren.

In dieser Arbeit wird ein auf morphologischen Operatoren basierendes Verfahren zur Ausdünnung der Schrift verwendet, welches die Konturen iterativ erodiert, bis eine Linienstärke von einem Bildpunkt wie in Abbildung 5.5 (c) erreicht ist. Im

(a) (b) (c) (d)G3 $\times$ 3M11 $\times$ 11 (e) (f)

Abbildung 5.5. Berechnung des ausgedünnten Bildes eines Wortes. (a,d) Binarisierte IESK-arDB-Probe (طفل), (b,e) Skeletonisierung, (c,f) Ausdünnung (Thinning). (d-f) Zeigen die Zwischenergebnisse nach Einsatz von Gauss- und Medianfilter. Die Kontur in Abbildung (d) ist deutlich ruhiger als in (a), allerdings tritt in der resultierenden Abbildung (f) eine Verfremdung der Topologie auf.

Gegensatz zur Skeletonisierung (siehe Abbildung 5.5 (b)) bleiben bei der Ausdünnung nur die Hauptkomponenten erhalten.

Der Einsatz von Gauss- und Medianfiltern vor der Binarisierung führt, wie Abbildung 5.2 (d) zeigt, zu glatteren Konturen und letztlich zu ruhigeren Linien im ausgedünnten Bild, wie in Abbildung 5.5 (f). Um die damit einhergehenden Verfremdung minimal zu halten, werden im Folgendem jedoch Filtermasken mit moderater Größe eingesetzt (M3 $\times$ 3 statt M11 $\times$ 11).

5.1.3 Sequenzdarstellung zusammenhängender Komponenten

Die zusammenhängende Komponenten (englisch Connected Components – CCs) lassen als Sequenzen von Bildpunkten der 8-Nachbarschaft beschreiben, welche sich aus den ausgedünnten Bildern extrahieren lassen. Dazu wird, ausgehend von der obersten Zeile, jeweils der erste zum Vordergrund gehörende Bildpunkt als Startpunkt der Kontur eines noch nicht besuchten CC selektiert. Die Kontur wird anschließend im bzw. gegen den Uhrzeigersinn abgelaufen, wobei die Koordinaten als Folge von Bildpunkten gespeichert werden. In der arabischen Handschrifterkennung bilden folgende Elemente CCs:

1. Diakritische Zeichen oder deren Bestandteile (Punkte)
2. Supplemente wie أ ا خّ, die vor allem in historischen Dokumenten vorkommen
3. Fragmentierte bzw. nicht verbundene Oberlängen, z.B. bei ك oder ط
4. Pieces of Arabic Words (PAWs) die 1.-3. nicht enthalten
5. Durch Rauschen oder schwache Pigmentierung erzeugte Schriftfragmente

bzw. bei der Binarisierung als Schrift klassifizierte Störungen im Hintergrund (vorwiegend bei historischen Dokumenten)

Für einige Arbeitsschritte ist die Speicherung der zusammenhängende Komponenten als Sequenzen sinnvoller als eine Bildrepräsentation, da erstere einen gezielten und somit mit geringen Kosten verbundenen Zugriff auf alle Vordergrundbildpunkte ermöglichen.

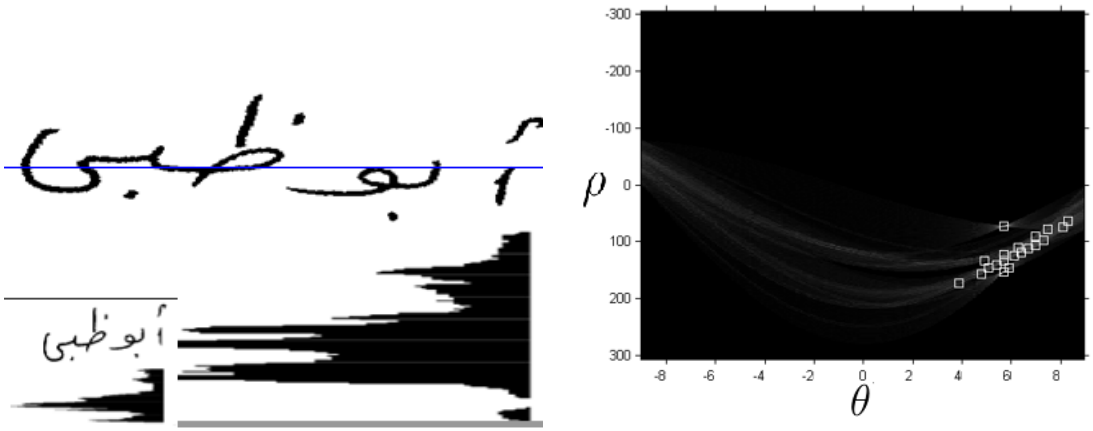
5.1.4 KeyFeatures

Anhand der 8-Nachbarschaft im ausgedünnten Bild lassen sich verschiedene KeyFeatures (KF) direkt bestimmen. Sei $\sum_{8N}(x,y)$ die Anzahl an Vordergrundbildpunkten in der 8-Nachbahrschaft des Bildpunktes an der Stelle (x,y), so gilt für Endpunkte (EPs) $\sum_{8N}(x,y) = 1$, für Verzweigungspunkte (BPs) $\sum_{8N}(x,y) \in \{3,4\}$ und für gewöhnliche Punkte stets $\sum_{8N}(x,y) = 2$.

5.1.5 Bestimmung der Basislinie

Bei arabischer Handschrift ist die Bestimmung der unteren Basislinien der Schriftzüge entscheidend für die anschließende Segmentierung sowie für die Klassifikation diakritischer Zeichen und Pieces of Arabic Words (PAWs). Zudem ist die Bestimmung der Basislinie erforderlich, um gegebenenfalls eine Rotation der Handschriftproben auszugleichen. Früher wurde davon ausgegangen, dass die Basislinien arabischer Handschrift annähernd horizontal verlaufen, und sich daher mit einer horizontalen Projektion ermitteln lassen. Dies tritt laut Pechwitz jedoch in vielen Fällen nicht ein, weshalb er stattdessen eine Lösung durch eine Hough-Transformation vorschlägt [64].

Hough-Transformationen werden zur Detektion von einfachen geometrischen Formen wie Kreisen oder Linien eingesetzt. Hierzu werden die Formen zunächst durch geeignete Parameter beschrieben und anschließend unterschiedliche Parametrisierungen mit einer Bildprobe verglichen [32]. Basislinien lassen sich durch Polarkoordinaten (ρ, θ) beschreiben. Bei den Proben der verwendeten Wortdatenbanken wird von einer annähernd geraden Basislinie mit einem nicht zu extrem von der Horizontalen abweichenden Winkel ausgegangen. Eine solche Basislinie lässt sich mit einer Hough-Transformation ermitteln. Diese vergleicht potentielle Basislinien, die – im Gegensatz zur einfachen Projektion – nicht nur in ihrer Höhe sondern auch in ihrem Winkel zur Horizontalen variieren. Die Basislinie ist nun



(a) (b)

Abbildung 5.6. Basisliniendetektion und -korrektur mittels Hough-Transformation. (a) Originalbild (unten links) und Bild nach Normierung der ermittelten Basislinie auf die Horizontale. (b) Hough-Raum mit Parametrisierungen zu möglichen Kandidaten der Basislinie.

jene Kombinationen von (ρ, θ), auf der sich die maximale Anzahl von Schriftpixeln befinden. Die zugrundeliegende Annahme ist hier, dass sich die Kashida größtenteils auf der Basislinie befinden. Dies ist für sehr kurze Wörter jedoch nicht oder nur teilweise der Fall, insbesondere dann, wenn diese viele isolierte Buchstaben und übereinandergeschriebene Buchstabenpaaren aus Tabelle 3.2 enthalten.

Der Hough-Raum in Abbildung 5.6 (b) enthält die Anzahl der Schriftbildpunkte unter der durch (ρ, θ) definierten Linie. Dabei wird ρ für jede mögliche y-Koordinate überprüft, θ hingegen für ein erwartetes Intervall. Da die Hough-Transformation relativ kostenintensiv ist, empfiehlt sich bei Schrift mit normaler Basislinienschräge eine Eingrenzung von $\theta \in [-25°,25]$. Zudem ist auch eine erhöhte Schrittweite von $2 - 4°$ sinnvoll, denn da die tatsächliche Basislinie ohnehin geringfügig von einer perfekten Geraden abweicht, ist eine Reduzierung der Kosten hier wichtiger, als maximale Präzision. Daher erfolgt im Weiteren eine entsprechende Bestimmung der Basislinien. Eine Analyse der IESK-arDB ergab, dass die meisten Basislinien im Intervall $[-15°,10°]$ liegen (siehe Abbildung 5.7 (a)). Weil die Bestimmung der Basislinie bei vielen der kürzeren Wörtern problematisch ist, wird die gemessene Verteilung verfälscht und lässt sich nur schlecht mit einer Normalverteilung approximieren.

Für den Fall, dass in der Praxis Basislinien mit einer deutlichen Kurvatur zu erwarten sind, ist es sinnvoll, eine auf KF basierende Methoden zu verwenden. Derartige Basislinien sind in erster Linie bei ganzen Textzeilen bzw. -Seiten zu

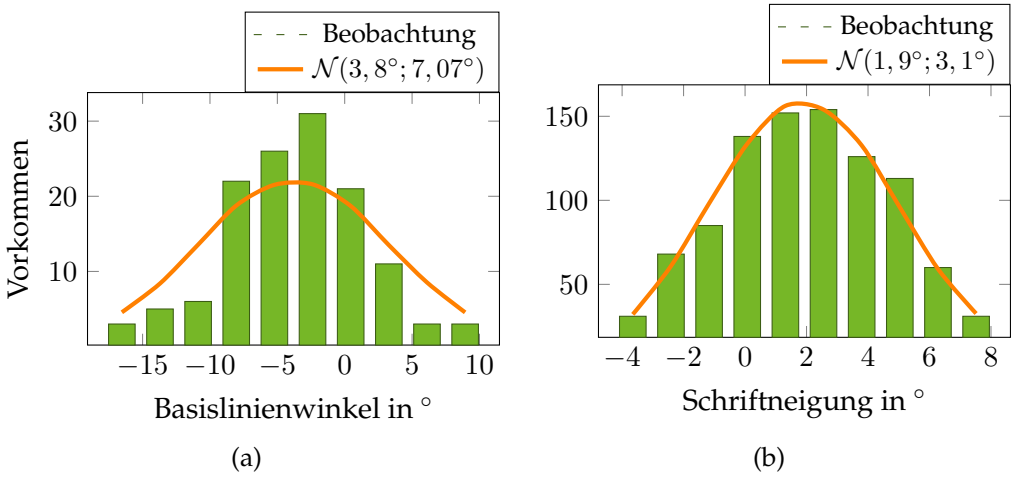


Abbildung 5.7. Analyse der Schriftneigung und des Winkels der Basislinien der IESK-arDB. (a) Analyse der Winkels zwischen Basislinie und Horizontaler, $\mathcal{N}(3{,}8; 7{,}07)$ besteht den χ^2-Test nicht. (b) Analyse der Schriftneigung, geschätze Gaussverteilung $\mathcal{N}(1{,}9; 3{,}1)$ besteht den χ^2-Test mit $\alpha = 0{,}05$.

erwarten. Da die KF basierte Methode geringere Kosten erfordert, lässt sie sich auch in Kombination mit der Hough-Transformation einsetzen, um zunächst den Parameters θ auf ein kleineres Intervall einzugrenzen [36].

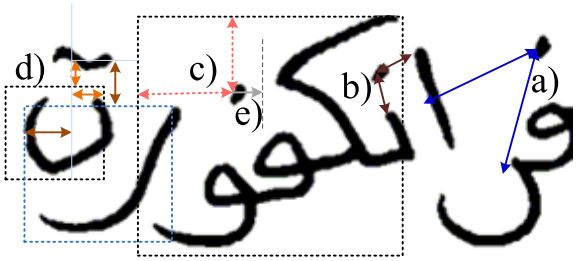
5.1.6 Korrektur der Schriftneigung

Eine Normierung der Schriftneigung verbessert gegebenenfalls die Ergebnisse der Segmentierung und Erkennung (abhängig von deren Ausprägung in realen Daten und Trainingsproben). Hierzu wird, analog zur Basisliniendetektion, die stärkste Ausprägung von Geraden (vorwiegend Ober- und Unterlängen) mit verschiedenen Winkeln detektiert. Ausgangspunkt ist hierbei die Vertikale. Eine genaue Beschreibung der verwendeten Implementierung zur Detektion und Korrektur der Schriftneigung wird in [36] gegeben.

Abbildung 5.7 (b) zeigt die gemessene Verteilung der Schriftneigung der IESK-arDB, welche sich gut durch die Normalverteilung $\mathcal{N}(1{,}9°; 3{,}1°)$ beschreiben lässt. Diese Werte lassen sich für das Synthesemodul verwenden, um realistische Variationen der Schriftneigung zu erzeugen.

5.2 Segmentierung von Wörtern in PAWs

In arabischer Handschrift überlappen sich benachbarte Pieces of Arabic Words (PAWs) häufig auf der x-Achse. Die Anzahl der PAWs stellt jedoch eine wichtige



(a)

Abbildung 5.8. Visualisierung der Distanzmaße zur Zuordnung von nicht-HPAW zu einem HPAW ("فرانكفورت", Probe A01_005 der IESK-arDB-Datenbank).

Information für weitere Methoden dar, und kann z.B. zur Einschränkung der Wörter verwendet werden, welche die Probe aufgrund ihrer Merkmale voraussichtlich darstellen könnte. Außerdem ist die Lokalisierung der PAWs für die spätere Segmentierung in Buchstaben entscheidend, weshalb die Überlappungen an dieser Stelle behoben werden müssen.

5.2.1 Verwendete Maße

Der erste Schritt der Segmentierung in PAWs besteht in der Klassifikation der zusammenhängende Komponenten (CCs) in HPAWs (Hauptkomponenten) und nicht-HPAWs (diakritische Zeichen, Hamza etc.). In Abschnitt 5.4 wird ein hierzu einsetzbarer, trainierbarer Ansatz beschrieben. Da die nötigen Grundwahrheiten (GT) jedoch nur für einige wenige Textproben vorliegen, nicht jedoch für die IESK-arDB, wird im Folgenden ein heuristischer Ansatz verwendet. Ein CC wird hierbei als HPAW deklariert, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Das umgebende Rechteck des CC schneidet die Basislinie [2]
- Die Länge des CC (in Bildpunkten) überschreitet 30% der durchschnittlichen CC-Länge (ermittelt aus mehreren Proben bzw. der aktuellen Textseite) *oder* der Steigungswinkel des umgebenden Rechtecks liegt über $35°$
- Das CC enthält mehr als drei Endpunkte (EPs) und mehr als einen Verzweigungspunkt (BP)

Allen übrigen CCs wird die Etikettierung „nicht-HPAW “ zugewiesen.

[2]Es wird davon ausgegangen, dass die Basislinie bereits mittels Hough-Transformation oder eines ähnliches Ansatzes berechnet wurde. Sofern die Berechnung der Basislinie erst nach der Klassifizierung der CCs erfolgen kann (Key Feature basierter Ansatz), wird dieses Merkmal nicht genutzt.

Der nächste Schritt besteht darin, alle nicht-HPAWs einem HPAW zuzuordnen. Hierzu werden die folgenden, in Abbildung 5.8 visualisierten, Abstandsmaße verwendet:

a) Der euklidische Abstand vom geometrischen Zentrum des HPAW zu jenem des nicht-HPAW

b) Der minimale Abstand vom geometrischem Zentrum des nicht-HPAW zu einem Punkt $(x,y) \in$ HPAW[3]

c) Geschätzte Wahrscheinlichkeit, dass sich das nicht-HPAW innerhalb des HPAW befindet:

   – 1, falls das nicht-HPAW in dem umgebendem Rechteck von genau einem HPAW liegt

   – sonst: Distanz vom nicht-HPAW zur jeweils näheren Begrenzung des umgebenden Rechtecks des HPAW auf der x- und y-Achse dividiert durch dessen Breite bzw. Höhe

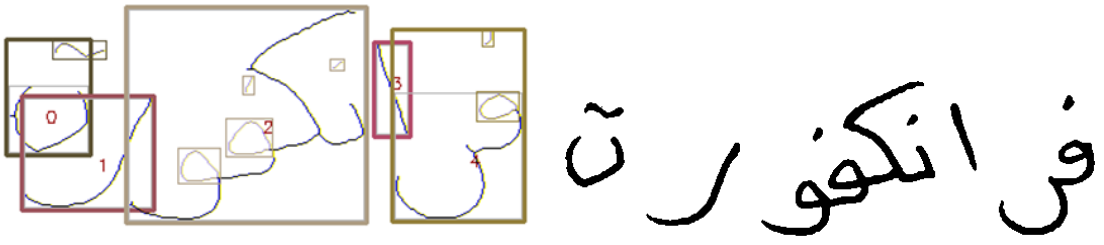
d) Analog zu c), jedoch mit möglicher negativer Distanz, falls das nicht-HPAW auf mindestens einer Achse außerhalb des HPAW liegt. Dabei wird die x-Achse stärker gewichtet, da sich diakritische Zeichen oft über oder unter einem HPAW befinden.

e) Optionale Korrektur der Position diakritischer Zeichen, bei Schrift die eine stark ausgeprägte Verschiebung diakritischer Zeichen in Schreibrichtung aufweist: Translation des nicht-HPAW auf der x-Achse um einen Wert kleiner b) und kleiner als die halbe durchschnittliche Buchstabenbreite.

5.2.2 Detektierte PAWs

Die Ausgabe der Ergebnisse erfolgt in zwei unterschiedlichen Formen. Die erste Form ist eine auf der Sequenzdarstellung basierende, sehr kompakte Datenstruktur, welche die Positionen der einzelnen CC nicht verändert, jedoch die Reihenfolge der HPAWs und die Zugehörigkeit der nicht-HPAWs speichert. Hierdurch lassen sich die Kosten für den weiteren Zugriff auf etwa 10% reduziert. Zudem enthält diese Darstellung einen erhöhten Informationsgehalt (Art und Relation der CCs,

[3]Hier wird die Sequenzdarstellung der zusammenhängende Komponenten verwendet, um die Kosten der Vorverarbeitung gering zu halten.



(a) (b)

Abbildung 5.9. PAW-Segmentierung (a) Visualisierung der auf der Sequenzdarstellung basierenden Datenstruktur zur Lösung überlappender PAWs. (b) Entzerrtes Bild, ohne überlappende PAWs.

umgebende Rechtecke), welcher die Umsetzung der in den nächsten Abschnitten beschriebenen Verfahren unterstützt. Abbildung 5.9 (a) visualisiert die Datenstruktur anhand eines Beispiels. Ein Nachteil ist jedoch, dass die anschließenden Methoden (Segmentierung und Klassifikation) auf diese Datenstruktur abgestimmt werden müssen. Daher wird das Ergebnis in einer alternativen bildbasierten Darstellung bereitgestellt, die als Eingabe und Ausgabe von Methoden eine universelle Schnittstelle darstellen. Für die vorliegende Forschungsarbeit wurde primär der innerhalb der Arbeitsgruppe von Herrn Moftah Elzobi entwickelte Ansatz zur Segmentierung in Buchstaben verwendet [37, 38], welcher zurzeit jedoch nur als MATLAB-Funktion vorliegt. Daher wurde als kompatibles Format ein entzerrtes Bild verwendet, in dem, wie Abbildung 5.9 (b) zeigt, keine überlappenden PAWs auftreten. Allerdings muss hier beachtet werden, dass ein solches entzerrtes Bild aufgrund der verschobenen PAWs nachträglich mit den Grundwahrheiten (GT) synchronisiert werden muss.

Die Segmentierung, sowie die Korrektur der Basislinien und der Schriftneigung, sind für den Großteil der Gesamtkosten des Schrifterkennungsvorgangs verantwortlich. Zur Beschleunigung empfiehlt es sich daher für zukünftige Arbeiten, diese Methoden auf Grundlage der Sequenzdarstellung umzusetzen, und Zwischenergebnisse entsprechend Abbildung 5.9 (a) zu verwenden.

5.2.3 Fazit zur Segmentierung in PAWs

Der Anteil an Wörtern, deren PAWs mit der beschriebenen Methode trotz Überlappung korrekt segmentiert werden, liegt bei 95%. Aufgrund der Variationen der Position, Form und Größe der in arabischer Handschrift vorkommenden CCs

erscheint ein heuristischer Ansatz dennoch nicht optimal. Ein zukünftige Optimierung der Parameter ist anhand der IESK-arDB$_{Syn}$ durchführbar, da deren GT die umgebenden Rechtecke aller CCs sowie deren Klassen enthalten. Obwohl im Vergleich zu anderen Datenbanken bereits mit umfangreichen GT ausgestattet, gilt dies nicht für die IESK-arDB, da der erforderliche Arbeitsaufwand die verfügbaren Mittel übersteigen würde. Dies verdeutlicht die Vorteile zusätzlicher synthetischer Datenbanken, wie sie mit dem in Kapitel 4.4 beschriebenem System generiert werden können.

5.3 Segmentierung von Wörtern in Buchstaben

Die Erkennung von Handschrift durch segmentierungsbasierte Verfahren lässt sich, im Gegensatz zur holistischen Erkennung, flexibel – d.h. ohne auf statische, mit der Trainingsdatenbank übereinstimmende Vokabulare limitiert zu sein – einsetzen. In dieser Arbeit werden hierzu an dieser Stelle Methoden zur expliziten Segmentierung von arabischer Handschrift eingesetzt die

1.) vor der Erkennung der Buchstaben eingesetzt werden,

2.) topologische Merkmale der PAWs, sogenannte KeyFeatures (KF)s, diakritische Zeichen und die Basislinie berechnen,

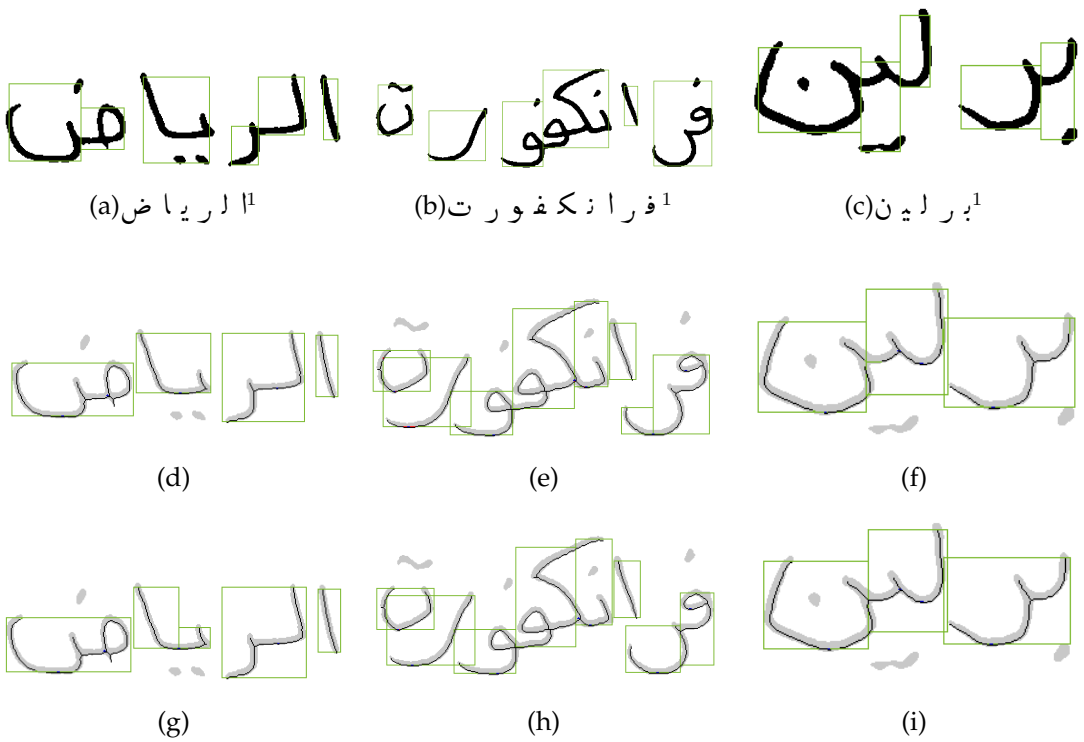
3.) aufgrund dieser extrahierten Informationen Kandidaten für Segmentierungspunkte (SK)s ermitteln und

4.) finale SKs aufgrund verschiedener (heuristischer) Regeln selektieren, welche von der Relation der KFs zueinander und zur Basislinie abhängen.

Im Rahmen dieser Dissertation wurden dabei zwei Ansätze aus dem Stand der Technik implementiert und mit dem in der Arbeitsgruppe NIT entstandenen Ansatz verglichen.

5.3.1 Beschreibung der Ansätze zur expliziten Segmentierung

Die Ansätze aus dem Stand der Technik wurden entsprechend ihrer Beschreibung implementiert, dabei jedoch dahingehend verändert, dass die Detektion der KeyFeatures (KFs) sowie die weiteren Schritte nicht bildbasiert sind, sondern anhand der berechneten Sequenzen und der in Abbildung 5.9 (a) dargestellten Datenstrukturen erfolgen. Dies hat jedoch ausschließlich Einfluss auf die Algorithmen zur



(a)ا[1] ل ر ي ا ض (b)ف ر ا ن ك ف و ر ت[1] (c)ب ر ل ي ن[1]

(d) (e) (f)

(g) (h) (i)

Abbildung 5.10. Beispiele für die Segmentierung einiger Proben (الرياض, فرانكفورت und برلين) der IESK-arDB mittels unterschiedlichen Segmentierungsmethoden. (a-c) Am IIKT entwickelte Methode [37]. (d-f) Lorigi et al. [59]. (g-i) Atici et al. [17]. [1]Im Wort vorkommende Buchstaben.

Berechnung der KFs. Zusätzlich zu den Verzweigungspunkten (BPs) und Endpunkten (EPs) wurden bei den Ansätzen aus der Literatur Minima als entscheidende KFs und potentielle Segmentierungspunkte (SK)s verwendet. Der hierfür Grund ist, dass zwei Buchstaben im Idealfall durch ein Kashida verbunden sind, welches genau ein Minima aufweist. Weitere KFs sind Punkte, die zu einer Schleife gehören (wie es z.B. in Abbildung 5.9 (a) bei den Buchstabe و und ف der Fall ist). Zudem werden diakritische Zeichen detektiert, wobei SKs, die sich unterhalb des umgebenden Rechtecks eines diakritischen Zeichens befinden, verworfen werden.

Der von Atici et al. [17] vorgestellte Ansatz ordnet alle KFs entsprechend ihrer x-Koordinaten und teilt sie in KF-Segmente ein. Diese sind jeweils durch einen BP oder EP begrenzt. Dazwischen befinden sich maximal zwei SKs: ausgehend von rechts und links jeweils das erste Minima. Sofern das benachbarte KF dieser SKs ein EP ist, muss der Winkel zwischen dem EP und dem SK und der Horizontalen einen Schwellwert überschreiten, damit der SK als finale Segmentierung festgelegt wird.

Der von Lorigo et. al. [59] beschriebene Ansatz sieht zunächst vor, jedes Minima als SK zu deklarieren und die Menge an SKs anschließend durch Anwendung

verschiedener Regeln auszudünnen. Ein SK wird verworfen, wenn

i er sich unterhalb der Basislinie befindet,

ii der SK sich innerhalb einer Schleife befindet oder

iii das nächste KF ein EP ist und der Winkel zwischen der Horizontalen und der Strecke zwischen SK und EP größer als ein gegebener Schwellwert ist.

Diese Regeln dienen vorrangig dazu abzuschätzen, ob sich ein SK innerhalb eines Buchstaben befindet, oder bereits andere SKs zwischen denselben Buchstaben detektiert wurden.

Der in der Arbeitsgruppe entwickelte Algorithmus wurde nach dem gleichem Prinzip konzeptioniert, nutzt jedoch andere Regeln. Insbesondere sind hier BPs statt Minima Kandidaten für Segmentierungen. BPs liegen zwar nicht mittig zwischen zwei Buchstaben, sind jedoch zuverlässiger detektierbar. Die genaue Beschreibung der Regeln findet sich in [36,37]. Ergebnisse zu allen drei Segmentierungsverfahren sind in Abbildung 5.10 dargestellt.

Obwohl die Idee des KF-basierten, expliziten Ansatzes zur Segmentierung in Buchstaben über zwei Jahrzehnte alt ist, werden auch heute noch neue auf zusätzlichen KFs und Regeln basierende Variationen erforscht. Es wird sogar von einer gegenüber impliziten Methoden verbesserten Worterkennung berichtet [3].

5.3.2 Fehlermaße zur Auswertung der Segmentierungsansätze

Zur Auswertung der vorgestellten Segmentierungsmethoden werden verschiedene Fehlermaße benutzt. Diese hängen von den optimalen Segmentierungspunkten (SPs), den Grundwahrheiten (GT) und den davon abgeleiteten Intervallen ab, in denen jeweils genau eine Segmentierung erwartet wird. Für jedes Intervall ohne SP, sowie für jeden zusätzlichen SP innerhalb oder außerhalb des Intervalls, wird ein Segmentierungsfehler induziert. Hieraus ergibt sich die durchschnittliche Anzahl an Segmentierungsfehlern pro Wort und pro Buchstabe, die in Tabelle 5.1 für die verschiedene Segmentierungsmethoden aufgelistet sind. Weiterhin ist der Anteil der übersegmentierten und untersegmentierten Wörter angegeben, die jeweils mindestens eine zusätzliche bzw. fehlende Segmentierungen aufweisen (d.h. ein Wort kann gleichzeitig unter- und übersegmentiert sein).

Ob eine Segmentierung korrekt ist, lässt sich nicht immer eindeutig bestimmen. Ist ein SP (ausgehend vom optimalem SP der GT) lediglich nach links oder

Tabelle 5.1. Automatischer Vergleich der Segmentierungsmethoden auf der IESK-arDB-Wortdatenbank (2738 Wortproben).

	Ansatz		
Fehlermaß	IIKT [37]	Lorigi et al. [59]	Atici et al. [17]
Fehler pro Buchstabe	$0{,}35 \pm 0{,}01$	$0{,}46 \pm 0{,}01$	$0{,}49 \pm 0{,}01$
Fehler pro Wort	$1{,}68 \pm 0{,}06$	$2{,}23 \pm 0{,}08$	$2{,}35 \pm 0{,}05$
Übersegmentierung	$79 \pm 9\%$	$34 \pm 4\%$	$31 \pm 2\%$
Untersegmentierung	$89 \pm 4\%$	$90 \pm 1\%$	$96 \pm 1\%$
Perfekt segmentierte Wörter	$17 \pm 4\%$	$4 \pm 2\%$	$3 \pm 1\%$
Korrekte Anzahl an Segmentierungen	$34 \pm 3\%$	$14 \pm 3\%$	$11 \pm 2\%$

rechts verschoben, besteht ein Segment gegebenenfalls nur aus dem Teil eines Buchstaben, wogegen sein Nachbarsegment zusätzlich den fehlenden Teil umfasst. Derartige Fehler treten in unterschiedlichen Intensitäten auf und lassen sich durch ausreichend Trainingsproben gegebenenfalls ausgleichen, so dass anschließend immer noch ein korrekte Klassifizierung möglich ist. Dies wird jedoch schwieriger, je näher der falsche SP an einer Buchstabenmitte liegt (auf der x-Achse). Bei fehlenden oder zusätzlichen Segmentierungen ist ein derartiger Ausgleich nicht möglich, es resultieren daher immer Fehler in der erkannten Sequenz von Buchstaben (zusätzliche Segmentierungen können jedoch durch entsprechend trainierte Long-Short-Term-Memory oder durch Nachverarbeitung korrigiert werden). Diese schweren Segmentierungsfehler lassen sich sowohl manuell als auch automatisch robuster detektieren, als fehlplazierte Segmentierungen. So lässt sich etwa der Anteil an Wörtern, welche die korrekten Anzahl an Segmentierungen aufweisen, auch automatisch eindeutig evaluieren (letzte Zeile in Tabelle 5.1). Dieses Fehlermaß enthält jedoch – im Gegensatz zum Anteil an perfekt segmentierter Wörter – keine Aussage darüber, ob sich die einzelnen Segmentierungen zwischen zwei oder innerhalb eines Buchstaben befinden, und ist daher als alleiniges Gütekriterium unzureichend.

5.3.3 Vergleich der Segmentierungsansätze

Die drei Ansätze zur expliziten Segmentierung wurden jeweils an der kompletten IESK-arDB getestet (inklusive Wörtern, die schwer segmentierbare Buchstaben

wie س oder problematische Buchstabenpaare wie ني enthalten). Dabei wurden, wie oben beschrieben, die Grundwahrheiten der Datenbank für eine automatische Berechnung verschiedener Fehlermaße benutzt, da sich manuell mit vertretbarem Aufwands zumeist nur wenige hundert Proben auswerten lassen [59]. Die so gemessenen Fehler sind im Vergleich zu manueller Auswertung höher, da das automatische Verfahren grenzwertige Segmentierungen eher als Fehler einstuft.

Wie Tabelle 5.1 zu entnehmen ist, liefert der in der Arbeitsgruppe entwickelte Ansatz im Vergleich zu den Ansätzen der Literatur gute Ergebnisse. Die höhere Übersegmentierungsrate lässt sich durch den Umstand erklären, dass dieses Verfahren bevorzugt an Verzweigungspunkten trennt, also nicht in der Mitte eines Kashida, sondern nahe dem linken Buchstaben. Daher sind viele Segmentierungen grenzwertig und werden gegebenenfalls als Übersegmentierung eingestuft. Die durchschnittliche Anzahl der Fehler, sowie die Wahrscheinlichkeit ein Wort perfekt zu segmentieren, fällt für das am IIKT entwickelte Verfahren jedoch wesentlich günstiger aus. Daher wird dieses Verfahren für das in Kapitel 7 beschriebene Worterkennungssystem und die Experimente in Kapitel 8 eingesetzt (hier erfolgt auch eine Auswertung des Segmentierungsverfahrens an synthetischen Proben).

5.4 Zeilensegmentierung

Sollen lediglich einzelne arabische Wörter, z.B. innerhalb von Formularen, erkannt werden, sind die bis hierhin diskutierten Segmentierungsmethoden hinreichend. Sofern jedoch Wörter innerhalb handschriftlicher Dokumente erkannt werden sollen, ist zumindest die Segmentierung einzelner Textzeilen von entscheidender Bedeutung.

Viele arabische Dokumente (auch historische) bestehen nur aus einem einzelnen Absatz und weisen keine zusätzlichen Elemente wie Abbildungen auf. Daher wird das im Folgenden vorgestellte Verfahren anhand entsprechender Dokumente konzipiert, validiert und getestet. Um in der Lage zu sein, auch Zeilen mit unregelmäßig gekrümmten Basislinien segmentieren zu können, basiert das Verfahren auf lokalen Gruppen, wobei als Elemente Pieces of Arabic Words (PAWs) eingesetzt werden (siehe Abschnitt 2.5). Zunächst muss das erste Element jeder Zeile gefunden werden, welches bei arabischer Schrift eines der am weitesten rechts befindlichen PAWs ist. Ausgehend von diesen Startpunkten erfolgt anschließend jeweils eine serielle Gruppierung der PAWs. In jedem Iterationsschritt wird einer der nächsten

Nachbarn des aktuellen Piece of Arabic Word (PAW) als Nachfolger ausgewählt, bis das Ende der Zeile erreicht wurde.

Neben Gütemaßen zum Vergleich der potentiellen Nachfolger des aktuellen PAW (siehe Abschnitt 5.4.1), werden beim vorgeschlagenen Verfahren auch die verschiedenen sich daraus ergebenden Zeilenkandidaten bewertet, um anschließend die plausibelste Lösung zu selektieren (Abschnitt 5.4.3). Der wesentliche Forschungsbeitrag liegt dabei darin, verschiedene Nächster-Nachbar Maße zu konzipieren und zu optimieren, um hierdurch das Risiko zu minimieren, den falschen Nächsten-Nachbarn zu selektieren.

5.4.1 Klassifikation zusammenhängender Komponenten

Der erste wichtige Schritt eines auf lokalen Gruppen basierenden Ansatzes zur Zeilensegmentierung besteht wie in Abschnitt 5.1.3 darin, alle zusammenhängende Komponenten (CCs) innerhalb eines Dokumentes zu detektieren. Die CCs werden als Knoten gespeichert, welche die Sequenzdarstellung sowie hieraus extrahierte Informationen wie das umgebende Rechteck und den geometrischen Mittelpunkt des CC enthalten. Anschließend werden alle CCs absteigend nach ihrer oberen y-Koordinate sortiert.

In der arabischen Schrift treten viele diakritische Zeichen (ت ث ش) und gegebenenfalls Supplemente (ا أ) auf. Daher ist es notwendig, alle CCs wie in Abschnitt 5.2.1 als HPAWs (in diesem Zusammenhang die Hauptkörper der zusammengeschriebenen Buchstaben) und nicht-HPAWs (diakritische Zeichen und sonstige Bestandteile) zu klassifizieren. Dies ist wichtig, da sich die diakritische Zeichen jeweils zwischen zwei Zeilen befinden. Sie sollen deshalb zunächst ausgeblendet werden, so dass ausschließlich HPAWs zur Detektion einer Zeile verwendet werden. Anschließen werden die nicht-HPAWs demjenigen HPAW zugeordnet, zu dem sie voraussichtlich gehören.

Anstatt der in Abschnitt 5.2.1 beschriebene heuristische Methode soll an dieser Stelle die Klassifikation der CCs mittels statistischer Merkmale erfolgen. Die IESK-arDB bietet hierzu keine hinreichenden Grundwahrheiten (GT), jedoch wurden 10 handschriftliche Textseiten mit detaillierten GT versehen (umgebende Rechtecke für alle HPAWs und nicht-HPAWs). Hierbei wurden $\frac{2}{3}$ der Daten zum Trainieren und der Rest zum Testen von Support Vector Machines (SVMs) eingesetzt, die mit einer Genauigkeit von 91,5% zwischen HPAWs und nicht-HPAWs zu unterscheiden

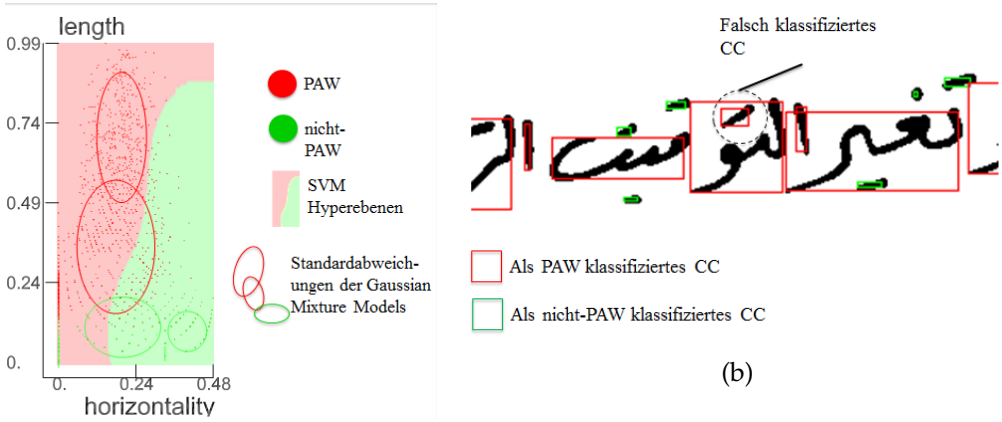


Abbildung 5.11. Die Abbildung zeigt, wie zusammenhängende Komponenten (CCs) mit Hilfe von Support Vector Machines bzw. Gaussian Mixture Models in PAW und nicht-PAW klassifiziert werden. (a) Darstellung der Separierung der Klassen im Merkmalsraum. (b) Ausschnitt aus einem Dokument mit klassifizierten CCs.

vermögen. Als Merkmale wurden Chaincode Histogramme eingesetzt [31]. Weitere Merkmale sind die Länge und die Fläche des CC, jeweils dividiert durch die durchschnittliche Länge bzw. Fläche im aktuellen Dokument. Hinzu kommen die Orientierung sowie die Exzentrizität. Beispiele und Ergebnisse der CC Klassifikation zeigt Abbildung 5.11.

Der statistische Ansatz hat gegenüber dem heuristischen Vorgehen den Vorteil, dass eine Erkennung auch bei problematischen Abweichungen der diakritischen Zeichen (durch den persönlichen Stil aber auch bei kleiner Schrift) durch entsprechende Trainingsdaten ermöglicht wird. Diese Abweichungen sind bei ganzen Textseiten ausgeprägter, als bei den einzelnen Wörtern der IESK-arDB.

5.4.2 Distanzmaße für lokale Gruppen

Nachdem alle CCs in HPAWs und nicht-HPAWs klassifiziert wurden, soll nun die Reihenfolge der HPAWs im Dokument ermittelt werden. Diese bilden (zusammen mit der späteren Zuordnung der nicht-HPAWs) die lokalen Gruppen, aus deren Permutationen sich letztendlich die segmentierten Zeilen ergeben. Dazu wird zunächst ein Distanzmaß $\rho \in [0,1]$ benötigt, welches sich im Idealfall antiproportional zur Wahrscheinlichkeit verhält, dass ein HPAW der Nachfolger des aktuellen HPAW ist. Von Kumar et al. wird vorgeschlagen, hierzu die gewichtete euklidische Distanz zu verwenden [53]. In dieser Arbeit sollen darüber hinaus zwei neue Distanzmaße untersucht werden.

Um die Berechnung der Distanzmaße zu beschleunigen, erfolgt eine Vorauswahl der HPAWs. So wird ρ nur für diejenigen berechnet, deren geometrische Zentren auf der x-Achse nicht weiter als das sechsfache der durchschnittlichen[4] Länge, und auf der y-Achse nicht weiter als das vierfache der durchschnittlichen Höhe vom Zentrum des aktuellen HPAW entfernt liegt.

Evolutionäre Sigmoidfunktionen. Das erste neuartige Distanzmaß wird aufgrund von beobachtbaren Eigenschaften arabischer Handschrift zunächst manuell konzipiert. In einem zweiten Schritt werden anschließend die verwendeten Hyperparameter $w_1 - w_8 \in \mathbf{w}$ durch evolutionäre Algorithmen optimiert.

Da im Arabischen von rechts nach links geschrieben wird, muss davon ausgegangen werden, dass der Nachfolger eines HPAWs sich immer links von diesem befindet. Weiterhin wird davon ausgegangen, dass die Basislinien der Zeilen sich einer Geraden mit dem Winkel α_0 annähern. Da auch gekrümmte Basislinien zulässig sein sollen, wird eine maximale positive $\omega_1 = \alpha_0 + \alpha_a$, und negative Abweichung $\omega_2 = \alpha_0 - \alpha_b$ definiert. Dabei gibt α_0 eine grobe Schätzung des Winkels zwischen der Horizontalen und den Basislinien an (dies wird beispielsweise im Vorfeld durch a priori Informationen oder mittels einer Analyse durch die Hough Transformation realisiert). Für die Untersuchungen in dieser Dissertation wird $\alpha_0 = 0$ gesetzt.

Um eine nichtlineare Änderung von α abhängig von der euklidischen Distanz l zum aktuellem HPAW zu ermöglichen, wird eine sigmoide Funktion eingesetzt:

$$\mathcal{S}(l,\omega_3,\omega_4) = \frac{1}{1 + e^{-\omega_3(l-\omega_4)}}. \tag{5.2}$$

Nun lässt sich das normierte Distanzmaß ρ_0 definieren:

$$\rho_0 = \begin{cases} 1 \quad \textbf{falls } \Big[l > \omega_5(\bar{l} + \sigma_l)\Big] \vee \Big[\alpha > \mathcal{S}(l,\omega_3,\omega_4)\omega_1\Big] \vee \Big[\alpha < \mathcal{S}(l,\omega_3,\omega_4)\omega_2\Big] \\ \omega_6 \dfrac{|l - \bar{l}|}{\omega_5} + \omega_7 \left(\dfrac{|\alpha - \alpha_0|}{\omega_2 - \omega_3}\right) \quad \textbf{sonst} \end{cases}, \tag{5.3}$$

mit dem sich für alle benachbarten HPAW die Wahrscheinlichkeit der Nachfolgerschaft abschätzten lässt. Hierbei ist $\bar{l}$ der Erwartungswert aller Distanzen l des aktuellen Dokumentes und σ_l die zugehörige Standardabweichung. Das finale Distanzmaß ergibt sich durch nochmaligen Einsatz der sigmoiden Funktion:

$$\rho_S = \mathcal{S}(\rho_0,0,\omega_8). \tag{5.4}$$

[4] Der Durchschnitt wird aus allen HPAWs der aktuellen Seite ermittelt.

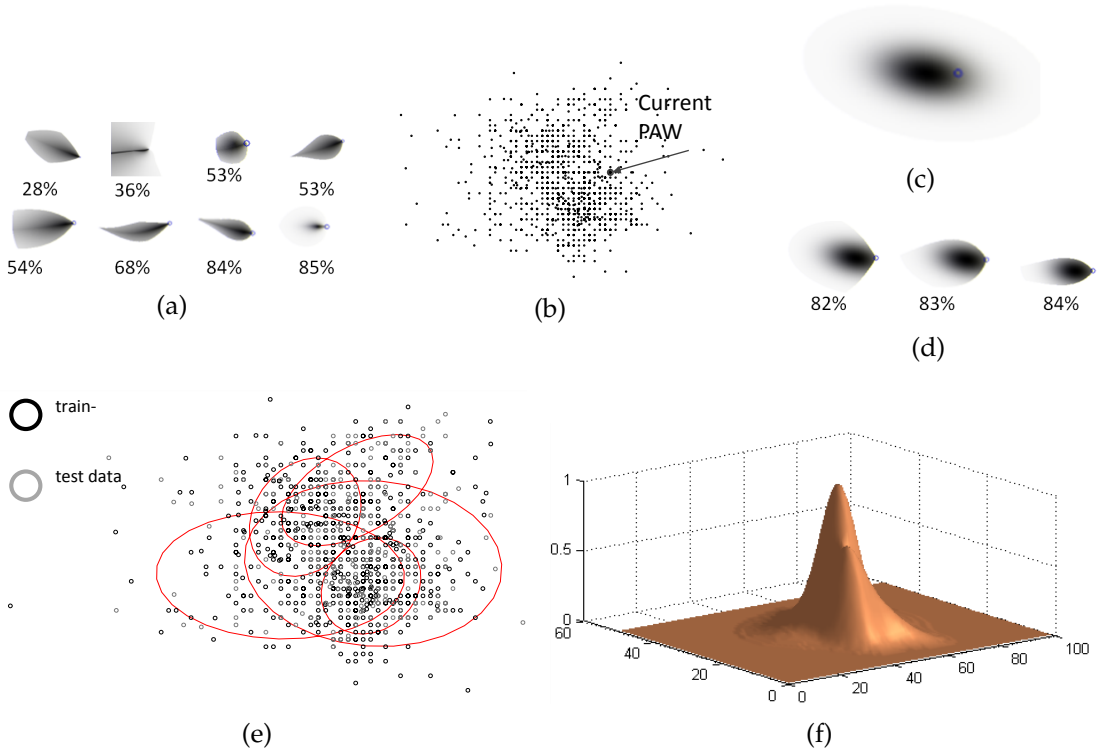


Abbildung 5.12. Maße für die Distanz vom aktuellem HPAW zu seinem Nachfolger: (a) Normiertes durch evolutionäre Algorithmen berechnetes Distanzmaß ρ_S um das aktuelle HPAW. (b) Statistische Verteilung der HPAWs (normierter Abstandsvektor). (c) Repräsentation als zweidimensionale normalverteilte Wahrscheinlichkeitsdichtefunktion. (d) Hybrides Distanzmaß. (e-f) Darstellung des Gaussian Mixture Models, zur Approximation der in (b) dargestellten Verteilung.

Alle HPAWs mit $\rho_S < 1$ sind potentielle Nachfolger des aktuellen HPAW und werden in einer aufsteigend geordneten Liste gespeichert.

Um die Parametrisierung $\mathbf{w}$ zu optimieren, wird ein evolutionärer Algorithmus mit Turnierauswahl, uniformem Crossover und Mutation eingesetzt. Eine Auswahl der resultierenden Phänotypen ist in Abbildung 5.12 (a) dargestellt. Die Fitness jedes Individuums $\mathbf{w}_i$ der Population entspricht der durchschnittlichen Genauigkeit der Zeilensegmentierung, die für jedes Individuum validiert wird.

Normalverteilte Wahrscheinlichkeitsdichtefunktion. Ein weiteres Distanzmaß,

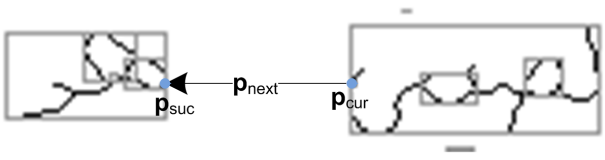


Abbildung 5.13. Schematische Darstellung der euklidischen Distanz zwischen einem HPAW und seinem potentiellen Nachfolger.

das im Rahmen dieser Arbeit untersucht wurde, basiert auf der statistischen Verteilung von HPAWs und deren Nachfolger, wie sie in Abbildung 5.12 (b) dargestellt ist. Aus dieser lässt sich eine zweidimensionale, normalverteilte Wahrscheinlichkeitsdichtefunktion berechnen, welche die tatsächliche Verteilung hinreichend approximiert. Zur genaueren Approximation wurden zudem Gaussian Mixture Models (GMM) berechnet, durch welche jedoch keine signifikante Verbesserung erzielt werden konnte (siehe Abbildung 5.12 (e-f)).

Zur Berechnung der zweidimensionalen normalverteilten Wahrscheinlichkeitsdichtefunktion werden – entsprechend der Darstellung in Abbildung 5.13 – die Vektoren

$$\mathbf{p}_{\mathsf{next}} = \mathbf{p}_{\mathsf{suc}} - \mathbf{p}_{\mathsf{cur}}, \tag{5.5}$$

zwischen allen aktuellen HPAWs und deren Nachfolgern berechnet (alternative Vektoren, wie z.B. die Distanz zwischen den geometrischen Zentren, wurden untersucht, führten jedoch zu weniger guten Ergebnissen). Alle $\mathbf{p}_{\mathsf{next}}$ lassen sich normieren, indem durch die durchschnittliche Länge $||\overline{\mathbf{p}_{\mathsf{next}}}||$ dividiert wird (welche jeweils für das aktuelle Dokument berechnet wird). Eine automatische Berechnung von $||\overline{\mathbf{p}_{\mathsf{next}}}||$ lässt sich (ohne Grundwahrheiten) allerdings erst nach der Zeilensegmentierung realisieren. Daher wird ersatzweise die durchschnittliche Breite $\overline{\mathfrak{w}}$ der HPAWs berechnet, welche mit $||\overline{\mathbf{p}_{\mathsf{next}}}||$ korreliert: $||\overline{\mathbf{p}_{\mathsf{next}}}|| \approx 0.7\ \overline{\mathfrak{w}}$. Nun werden anhand der $n_{\mathbf{p}}$ verfügbaren normierten Koordinaten

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \frac{\mathbf{p}_{\mathsf{next}}}{c_{\mathsf{norm}}} \left| c_{\mathsf{norm}} \in \left\{ ||\overline{\mathbf{p}_{\mathsf{next}}}||, 0.7\ \overline{\mathfrak{w}} \right\}, \right. \tag{5.6}$$

die Durchschnittswerte μ_1, μ_2, die Standardabweichungen σ_1, σ_2 sowie der Korrelationskoeffizient

$$\varrho_e = \frac{\frac{1}{n_{\mathbf{p}}} \sum_{i=1}^{n_{\mathbf{p}}} (x_{1_i} - \mu_1)(x_{2_i} - \mu_2)}{\sigma_1 \cdot \sigma_2}, \tag{5.7}$$

der normalverteilten Wahrscheinlichkeitsdichtefunktion berechnet. Hierzu werden Dokumente eingesetzt, für die manuell angefertigte Grundwahrheiten zur Verfügung stehen. Eine zweidimensionale normalverteile Wahrscheinlichkeitsdichtefunktion lässt sich dementsprechend wie folgt definieren:

$$\begin{gathered} g(x,y) = g(x_1,x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\varrho_e^2}} \cdot \exp\left(-\frac{1}{2\left(1-\varrho_e^2\right)} \left[\frac{{z_1}^2}{\sigma_1^2} + \frac{{z_2}^2}{\sigma_2^2} - \frac{2\varrho_e z_1 z_2}{\sigma_1\sigma_2} \right] \right), \\ |z_i = (x_i - \mu_i)\,. \end{gathered} \tag{5.8}$$

Damit lässt sich das zweite Distanzmaß berechnen:

$$\rho_G = 1 - \frac{g(x_1, x_2)}{g(\mu_1,\mu_2)}. \tag{5.9}$$

Hybrides Distanzmaß. Da die beiden Distanzmaße auf sehr unterschiedlichen Ansätzen mit verschiedenen Vor- und Nachteilen basieren, soll auch ein Hybrid aus beiden untersucht werden. Das erste Distanzmaß hat den Vorteil, auch globale Aspekte zu berücksichtigen, da die Parameter ω_i aufgrund der Ergebnisse der Zeilendetektion optimiert werden. Hierdurch lassen sich gegebenenfalls auch statistisch weniger wahrscheinliche Nachfolger favorisieren, wenn dies einem potentiell schwerwiegendem Fehler (wie einem Zeilensprung) entgegenwirkt. Das zweite Distanzmaß basiert hingegen auf einem statistischen Modell der Relation zwischen einem HPAW und seinem Nachfolger. Hierdurch wird eine bezüglich dieses Kriteriums lokal verlässliche Selektion des nächsten HPAW ermöglicht. Zudem ist das Modell weniger anfällig gegenüber Overfitting und lässt sich mit geringen Kosten erstellen.

Die beiden Ansätze werden kombiniert, indem die sigmoide Funktion für die äußere Begrenzung und die berechnete Wahrscheinlichkeitsdichtefunktion, als inneres Maß eingesetzt wird:

$$\rho = \begin{cases} \rho_0 & \textbf{falls} \quad \rho_0 = 1 \\ \rho_G & \textbf{sonst} \end{cases} \tag{5.10}$$

Hierbei werden die involvierten „äußeren“ Hyperparameter ω_1 bis ω_5 erneut optimiert, was zu den Resultaten führt, die in Abbildung 5.12 (d) auf Seite 118 dargestellt sind.

Mit den beschriebenen Distanzmaßen lassen sich im Folgenden alle potentiellen Nachfolger des betrachteten HPAW (d.h. seine Nächsten Nachbarn (NN)) ermitteln und sortieren.

5.4.3 Verfahren zur Zeilendetektion

Auf der Grundlage der vorhergehenden Arbeitsschritte soll nun der optimale Pfad vom ersten zum letzten HPAW jeder Zeile berechnet werden. Die Zeilensegmentierung ergibt sich dann implizit aus den detektierten Bestandteilen der Zeilen.

Start-HPAW Lokalisierung. Zunächst ist es erforderlich, vor der eigentlichen Zeilensegmentierung die ersten und optional auch alle letzten HPAWs einer Zeile zu

bestimmen (Start-HPAWs und End-HPAWs). Ein HPAW ist ein Start-HPAW, wenn sich kein HPAW rechts davon befindet (bzw. links im Fall von End-HPAW). Dabei wird im Winkel α_{start} bzw. α_{end} zum abgeschätzten Winkel der Basislinie α_0 gesucht: $\alpha_{start} = \alpha_0 \pm 40°$, $\alpha_{end} = \alpha_0 \pm 20°$ (aktuell wird von $\alpha_0 = 0$ ausgegangen).

Da arabische Dokumente in der Regel rechtsbündig sind, lassen sich die Start-HPAWs wesentlich zuverlässiger lokalisieren, als ihre Kontenparts am Zeilenende. Daher ist es sinnvoll, bei deutlich variierender Zeilenlänge und Basislinienkrümmung nur die Start-HPAWs explizit zu verwenden. In dieser Arbeit werden die End-HPAWs lediglich eingesetzt, um die Plausibilität verschiedener Zeilenkandidaten zu vergleichen, wogegen das verbindliche Ende eines Zeilenkandidaten dynamisch während seiner Sequenzierung detektiert wird.

Generierung von Zeilenkandidaten. Im nächsten Schritt sollen, ausgehend vom jeweils höchstgelegenem noch nicht besuchtem Start-HPAW, alle HPAWs einer Zeile besucht werden. Hierzu werden die HPAWs eines Dokumentes als Knoten $\mathfrak{n}_i$ eines Grafen beschreiben, die – wie in Abbildung 5.14 dargestellt – jeweils mit ihren Nächsten Nachbarn (NN) durch gerichtete und gewichtete Kanten $\mathfrak{e}_j$ verbunden sind (weist ein Knoten keine ausgehende Kante auf, so induziert dies das Ende der Textzeile). Hierdurch ergeben sich verschiedene Zeilenkandidaten $\mathfrak{N} = (\mathfrak{n}_1,\mathfrak{n}_i,...\mathfrak{n}_m)$ die miteinander verglichen werden, um anschließend den besten als finale Zeile $\mathfrak{N}_f$ zu selektieren.

Sei $\bar{n}(\mathfrak{n}_i)$ die durchschnittliche Anzahl an NN eines Knotens $\mathfrak{n}_i$ und $\bar{n}(\mathfrak{N})$ die durchschnittliche Anzahl von Knoten pro Zeile, so ergeben sich für jede Zeile $\bar{n}(\mathfrak{n}_i)^{\bar{n}(\mathfrak{N})}$ mögliche Kombinationen. Daher werden nicht alle, sondern nur eine Auswahl von $k_n < \bar{n}(\mathfrak{n}_i)^{\bar{n}(\mathfrak{N})}$ Zeilenkandidaten $\mathfrak{N}$ mit den zugehörigen Kanten $\mathfrak{E} = (\mathfrak{e}_1,\mathfrak{e}_i,...\mathfrak{e}_{\bar{n}(\mathfrak{N})-1})$ gebildet.

Das normierte Gewicht einer Kante $\mathfrak{e}_i$

$$p_i = \frac{1-\rho_i}{\sum_{j=1}^{\bar{n}(\mathfrak{n}_c)} 1-\rho_j} \in [0,1], \tag{5.11}$$

wird mit dem oben vorgestellten Distanzmaß ρ berechnet (oder alternativ mit ρ_S, ρ_G oder der gewichteten euklidischen Distanz). Es approximiert die Wahrscheinlichkeit, dass $\mathfrak{e}_i$ auf den korrekten Nachfolger des aktuellen Knotens $\mathfrak{n}_c$ zeigt. Zusätzlich wird auch die euklidische Distanz zwischen den Knoten in der Kante gespeichert.

Zur Generierung eines Zeilenkandidaten $\mathfrak{N}$ wird der aktuelle Knoten $\mathfrak{n}_c$ auf das

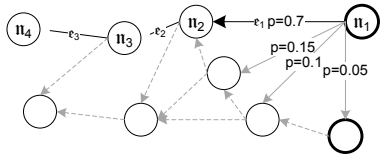


Abbildung 5.14. Schema der Selektion von Zeilenkandidaten. Alle Knoten $\mathfrak{n}_i$ weisen gewichtete Kanten zu ihren Nächsten Nachbarn auf, wodurch sich jede mögliche Folge von Knoten bewerten lässt.

nächste Start-HPAW gesetzt. Wird anschließend stets die Kante mit dem höchsten Gewicht selektiert, existiert nur ein Zeilenkandidat. Sofern eine robuste Detektion der End-HPAWs realisiert wird, ließen sich Algorithmen wie Dijkstra oder A* verwenden, um den günstigsten Pfad zu berechnen. Da die Gewichte aber lediglich ein lokales Bewertungsmaß darstellen, wird hierdurch keine hinreichend gute Repräsentation der Zeile sichergestellt. Daher wird, solange $\mathfrak{n}_\mathfrak{c}$ ausgehende Kanten aufweist, eine dieser Kanten zufällig mit der *Wheel of Fortune* Methode selektiert, so dass Kanten mit hohem Gewicht p_i bevorzugt werden. Hierbei werden alle Kanten ignoriert, die auf Knoten zeigen, welche bereits Teil des aktuellen Zeilenkandidaten $\mathfrak{N}$ oder einer finale Zeile $\mathfrak{N}_\mathfrak{f}$ sind. Dieser Prozess wird wiederholt, bis k Zeilenkandidaten generiert wurden.

Im Folgenden werden holistische Bewertungskriterien für die Auswahl von k_n Zeilenkandidaten vorgestellt.

Qualitätsmaße zur Selektion von Zeilenkandidaten. Zur Bewertung der Zeilenkandidaten $\mathfrak{N}$ werden die folgenden Maße ϱ verwendet:

ϱ_1 Das durchschnittliche Gewicht $\overline{p_i}$ der Kanten, dividiert durch die Anzahl der Kanten in $\mathfrak{N}$.

ϱ_2 0 falls $\sum \mathfrak{e}_i < \mathcal{W}$, $\frac{1}{\mathcal{W}} \sum \mathfrak{e}_i - 1$ sonst, , wobei $\mathsf{eukl}(\mathfrak{e}_i)$ die euklidische Distanz einer Kante $\mathfrak{e}_i \in \mathfrak{E}$ und $\mathcal{W}$ die Breite des aktuellen Dokumentes bezeichnet (Zeile krumm bzw. zickzackförmig, induziert Zeilensprünge).

ϱ_3 Anzahl der sich oberhalb des Zeilenkandidaten befindlichen, noch nicht besuchten (verpassten) Knoten dividiert durch die aktuelle Anzahl finaler Zeilen (induziert Zeilensprünge).

ϱ_4 Strafterm, der auf 0 gesetzt wird, falls $\mathfrak{N}$ mit einem zuvor als letztes HPAW der Zeile gelabelten Knoten endet (andernfalls gilt $\varrho_4 = 5$).

Durch einen Optimierung mit einem evolutionären Algorithmus ergeben sich für die Qualitätsmaße folgende Gewichte: $w_{\varrho 1} : -0.25, w_{\varrho 2} : 1.1, w_{\varrho 3} : 0.21, w_{\varrho 4} : 0.038$.

Algorithmus 5.1 : Selektion einer finaler Zeilen.

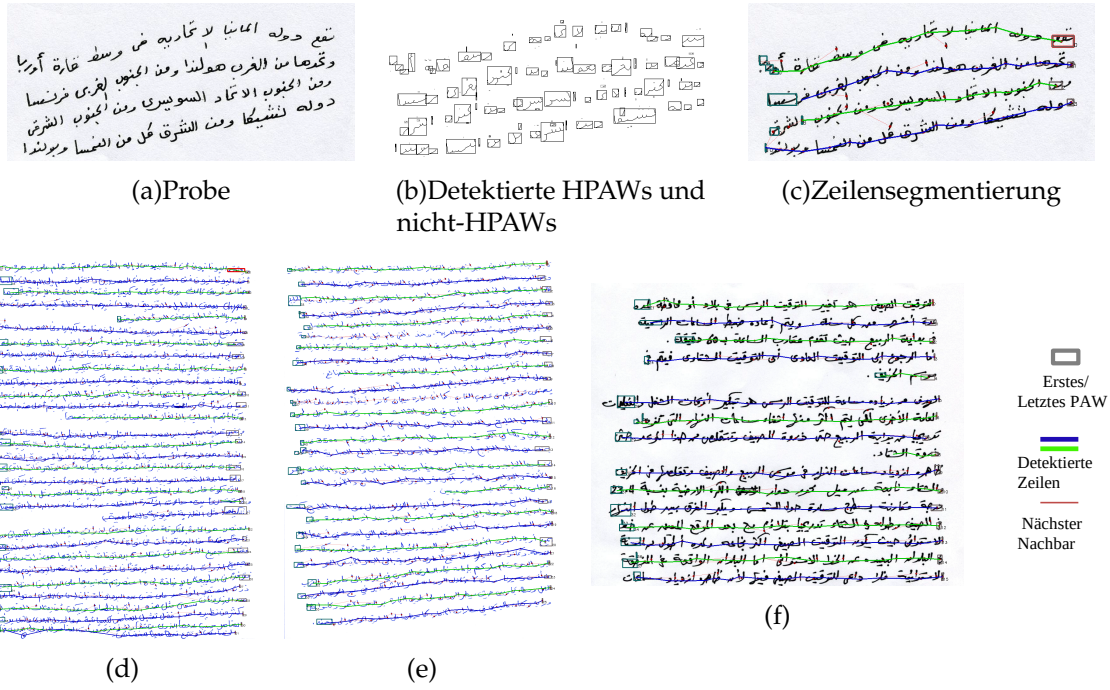
Eingabe : Liste aller Knoten $\mathfrak{n}$ und Kanten $\mathfrak{e}$ sowie Knoten der Start-HPAWs (unbesucht $\mathfrak{n}_{\mathsf{start}\bigcirc}$, besucht $\mathfrak{n}_{\mathsf{start}\bullet}$)
Ausgabe : Finale Zeilen $\mathfrak{N}_\mathfrak{f}$
solange $\exists \mathfrak{n}_{\mathsf{start}\bigcirc}$ **tue**
  für $a \leftarrow 1$ **bis** k_n **tue**
    Setze $\mathfrak{n}_c := \mathfrak{n}_{\mathsf{start}\bigcirc}$
    solange $\mathfrak{n}_c$ *eine Kante* $\mathfrak{e}_i$ *zu einem unbesuchten Nachbarn aufweist* **tue**
      wenn $k_n = 1$ **dann**
        $\mathfrak{n}_c :=$ Ziel von $\mathfrak{e}_i$ mit größtem p_i (siehe Gleichung 5.11) ;
      sonst
        $\mathfrak{n}_c :=$ Wähle zufällig das Ziel einer Kante (mit der Wahrscheinlichkeit p_i) ;
      Füge $\mathfrak{n}_c$ zu $\mathfrak{N}_a$ hinzu ;
  Berechne $\varrho_1 - \varrho_4$ für jedes $\mathfrak{N}_a$;
  Selektiere $\mathfrak{N}_a$ mit minimalem $\sum_{e=1}^{4} w_{\varrho e} \cdot \varrho_e$ als finale Zeile $\mathfrak{N}_\mathfrak{f}$ zum aktuellen $\mathfrak{n}_{\mathsf{start}\bullet}$;
  Markiere alle $\mathfrak{n} \in \mathfrak{N}_\mathfrak{f}$ permanent als besucht;

Alle finalen Zeilen des Dokumentes lassen sich entsprechend Algorithmus 5.1 ermitteln. Anschließend werden alle verpassten Knoten $\mathfrak{n}_i$ nachträglich einer finalen Zeile $\mathfrak{N}_\mathfrak{f}$ zugeordnet, um die Genauigkeit der Zeilensegmentierung zu erhöhen. Dies geschieht auf Grundlage der y-Distanz zwischen dem Knoten $\mathfrak{n}_i$ und der Basislinie, d.h. der Kantenfolge von $\mathfrak{N}_\mathfrak{f}$. Beispiele für detektierte finale Zeilen finden sich in Abbildung 5.15 auf Seite 124. Auch die nicht-HPAW sind hierdurch erfasst, da sie vorher bereits einem HPAW zugeordnet wurden. Somit ist die Zeilensegmentierung abgeschlossen.

5.4.4 Experimentelle Ergebnisse

Die zum Trainieren und Validieren der Zeilensegmentierung eingesetzte Datenbank wurde mit Grundwahrheiten (GT) versehen. Diese enthalten Informationen über die umgebenden Rechtecke aller HPAWs sowie deren Reihenfolge und Zeilen-Zugehörigkeit. Im Folgenden soll erläutert werden, wie damit das Verfahren zur Zeilensegmentierung evaluiert wird.

Falls ein HPAW zwar laut GT in der i-ten Zeile vorkommt, dieses HPAW jedoch nicht Teil der berechneten i-ten finalen Zeile $\mathfrak{N}_{\mathfrak{f}_i}$ ist, wird dieses HPAW als False Negativ (FN) deklariert. Kommt ein HPAW hingegen ausschließlich in $\mathfrak{N}_{\mathfrak{f}_i}$ vor,



(a)Probe (b)Detektierte HPAWs und nicht-HPAWs (c)Zeilensegmentierung

(d) (e) (f)

Abbildung 5.15. Visualisierung des Prozesses (a-c) sowie einiger Ergebnisse der Zeilensegmentierung (d-f).

wird es als False Positive (FP) deklariert. Ferner werden alle HPAWs, die sowohl in $\mathfrak{N}_{\mathfrak{f}_i}$ als auch in der entsprechenden GT-Zeile existieren, als True Positive (TP) und alle die in beiden fehlen als True Negative (TN) deklariert. Die Genauigkeit der Zeilensegmentierung in % ist dementsprechend durch

$$A_Z = 100\frac{TP + TN}{TP + TN + FP + FN}, \tag{5.12}$$

gegeben. Weiterhin gibt A'_Z die Genauigkeit an, nachdem alle verpassten HPAWs (bzw. Knoten $\mathfrak{n}$) einer Zeile zugeordnet wurden.

Optimierung wesentlicher Systemparameter. Ein wichtiger Hyperparameter des auf sigmoiden Funktionen basierenden Distanzmaßes ist ω_5, da durch diesen die maximal zulässige Distanz eines Nachfolgers $\mathfrak{n}_i$ gesteuert wird. Ist diese zu niedrig, kann ein exzessiver Abstand zwischen zwei Worten (innerhalb der Test-Proben) dazu führen, dass ein Zeilensprung eintritt, oder die restlichen HPAWs der Zeile verpasst werden. Wie Abbildung 5.16 (a) zeigt, lässt sich dies weitestgehend unterdrücken, indem bei der Kodierung des evolutionären Algorithmus $\omega_5 \geq 4$ sichergestellt wird.

Abbildung. 5.16 (b) zeigt den Einfluss der Anzahl verwendeter Zeilenkandidaten pro Zeile auf die Genauigkeit der Zeilensegmentierung. Wenigstens 50-100 Zeilenkandidaten sind für eine merkliche Erhöhung der Genauigkeit erforderlich. Dies

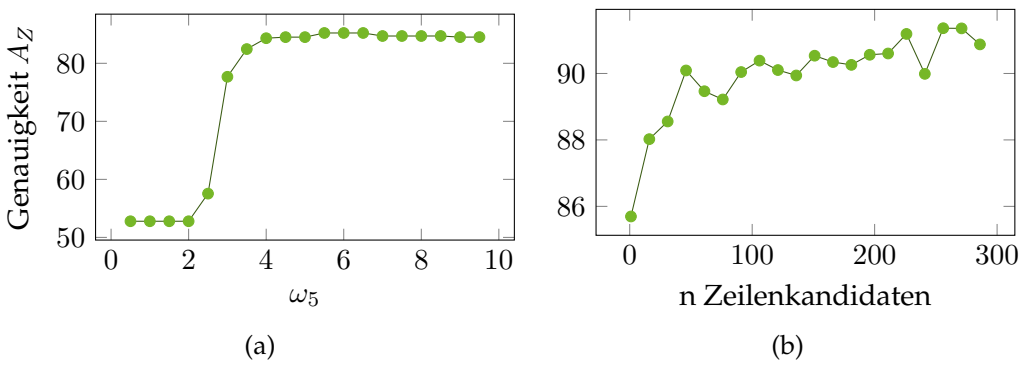


Abbildung 5.16. Optimierung der wichtigsten Parameter des Zeilensegmentierungssystems. (a) ω_5 steuert die maximal zulässige Entfernung zwischen zwei aufeinanderfolgenden HPAWs eines Zeilenkandidaten. (b) Die Anzahl der generierten Zeilenkandidaten, aus denen die finale Zeile ausgewählt wird.

sind allerdings deutlich weniger, als alle n^m möglichen Zeilen, was entsprechend der verwendeten Dokumente zwischen 1000 und 8^{30} Zeilenkandidaten entspräche.

Vergleich der Distanzmaße. Die Konzeption und der Vergleich verschiedener Maße für die Distanz zwischen zwei HPAWs ist ein entscheidender Beitrag dieses Kapitels.

Mit der normalverteilten Wahrscheinlichkeitsdichtefunktion werden zwar weniger HPAWs als mit den anderen Distanzmaßen verpasst (hoher A_Z in Tabelle 5.2), allerdings erhöht dies das Risiko eines Zeilensprunges, insbesondere, falls die Test- und Trainingsproben stark voneinander abweichen. Das auf Sigmoidfunktionen basierende Distanzmaß wird mit A_Z bzw. A'_Z als Zielfunktion optimiert, anstatt ein ausschließlich auf den Grundwahrheiten basierendes Modell der benachbarten HPAWs zu berechnen. Hierdurch lässt sich eine gute finale Genauigkeit A'_Z auch dann erzielen, wenn die Detektion der einzelnen NNs statistisch weniger zuverlässig ist. Eine Kombination beider Distanzmaße führt zu einer geringfügigen Erhöhung von A'_Z. In allen Fällen wird jedoch ein höherer Wert fürA_Z und A'_Z erzielt, als mit dem Standarddistanzmaß (der gewichteten euklidischen Distanz).

5.4.5 Zusammenfassende Betrachtung des Verfahrens zur Zeilensegmentierung

Wie die Beispiele in Abbildung 5.15 zeigen, lässt sich das vorgeschlagene Verfahren zur Zeilensegmentierung für handgeschriebene arabische Dokumente mit geraden aber auch unregelmäßig gekrümmten Textzeilen einsetzen. Die Untersuchungen

Tabelle 5.2. Treffergenauigkeit der Zeilensegmentierung in % in Abhängigkeit der verwendeten Distanzmaße.

		Distanzmaß			
		euklidisch	Norm.	Sigmoid	Komb.
verpasste HPAWs:	ignorieren (A_Z)	80,5	90,1	85,1	84,8
	Zeile zuordnen (A'_Z)	91,0	95,5	95,8	96,4

haben gezeigt, dass die beiden vorgeschlagenen Abstandsmaße bessere Ergebnisse erzielen, als dies mit dem Standardmaß für auf lokalen Gruppen basierender Zeilensegmentierung möglich ist.

Ein Problem von lokalen Gruppen besteht darin, dass in einigen Fällen – wie in Abbildung 5.15 (e) – Zeilensprünge auftreten, wenn dass zugehörige Start-HPAWs fälschlicherweise als diakritisches Zeichen erkannt wurde. Basierend auf der verwendeten Datanbank beträgt die Genauigkeit der Connected Component Klassifikation lediglich 91%. Allerdings bietet keiner der anderen Ansätze zur Zeilensegmentierung generell bessere Ergebnisse, wobei einige der Ansätze zudem annähernd gerade Basislinien voraussetzen [57].

5.5 Zusammenfassung des Kapitels

In diesem Kapitel wurden die wesentlichen Vorverarbeitungs- und Segmentierungsschritte vorgestellt, die zur Erkennung handschriftlicher arabischer Wörter mittels der in den folgenden Kapiteln vorgestellten Methoden zur Klassifikation und Nachverarbeitung erforderlich sind. Die Segmentierung von Wörtern in Buchstaben stellt dabei die größte Unsicherheit dar, auch wenn der in der Arbeitsgruppe entwickelte Ansatz bessere Ergebnisse liefert als die untersuchten Ansätze aus dem Stand der Technik. Eine zukünftige Optimierung des Segmentierungsansatzes hat demnach ein hohes Potential, das im Folgenden beschriebene Worterkennungssystem deutlich zu verbessern, da die im nächsten Kapitel erläuterte Klassifikation der Buchstaben stark von den Segmentierungsergebnissen abhängt.

KAPITEL 6

Buchstabenerkennung

NACHDEM im letzten Kapitel die Segmentierung von Bildern arabischer Wörter in einzelne Buchstaben beschrieben wurde, soll nun auf die Erkennung dieser Buchstaben eingegangen werden. Diese umfasst die Extraktion geeigneter Merkmale sowie die Verwendung und Konzeption von Klassifikatoren, um die Merkmale auszuwerten.

Hauptsächlich werden im Folgenden diskriminative Support Vector Machines (SVMs) zur Buchstabenklassifikation eingesetzt. Die als alternative konzipierten generativen Klassifikatoren auf Grundlage von Active Shape Models (ASMs) wurden zusammen mit speziellen Merkmalen entwickelt. ASMs bieten den Vorteil, dass sich auch während der Erkennungsphase Änderungen am Modell vornehmen lassen, verursachen jedoch ähnlich hohe Kosten wie das k-NN Verfahren.

Eine Übersicht der in diesem Kapitel beschriebenen Abläufe ist in Abbildung 6.1 gegeben. Im nächsten Abschnitt wird zunächst näher auf die Merkmalsextraktion eingegangen. Hiernach werden die verwendeten Klassifikatoren in Abschnitt 6.2 beschrieben.

6.1 Merkmalsextraktion

Zur Erkennung arabischer Buchstaben kommen in dieser Arbeit verschiedene Merkmale zum Einsatz. Für die optionale, heuristischen Vorklassifizierung eignen sich prinzipiell diskrete strukturelle Merkmale, wie die Art und Position diakritischer Zeichen. Die eigentliche Klassifikation erfolgt anhand verschiedener statistischer

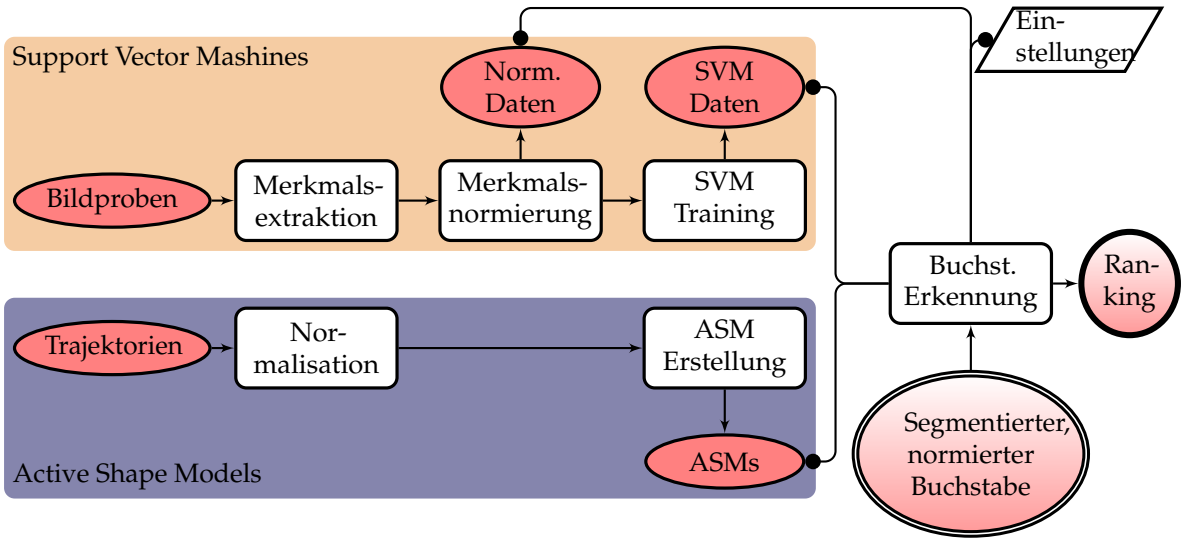


Abbildung 6.1. Flussdiagramm zum Ablauf der Buchstabenerkennung.

Merkmale, die in einem Merkmalsvektor zusammengefasst werden.

6.1.1 Diskrete strukturelle Merkmale zur Vorklassifizierung

Das deutsche Alphabet kennt drei Umlaute (ä,ü,ö), deren Hauptkörper jeweils mit einem Buchstaben des lateinischen Grundalphabets identisch sind (a,u,o). Bereits im regulärem arabischen Alphabet existieren hingegen drei unterschiedliche diakritische Zeichen (1-3 Punkte), wobei zwei sowohl über als auch unter dem Hauptkörper bzw. der Basislinie auftreten. Insgesamt unterscheiden sich hier 20 Buchstaben ausschließlich durch ihre diakritischen Zeichen eindeutig von allen anderen. Davon bilden vierzehn die Paare (ر ز), (د ذ), (س ش), (ص ض), (ط ظ), (ة ه), (ع غ) und sechs die beiden Dreiergruppen (ب ت ث) und (ج ح خ). Darüber hinaus unterscheiden sich die Hauptkörper der Buchstaben (ف ق) nur marginal. Weiterhin existieren zusätzliche Abwandlungen einiger Buchstaben, die nicht Teil des regulären Alphabetes sind, sowie weitere Abwandlungen für Farsi, Urdu oder persische Schrift. Daher bilden die diakritischen Zeichen wichtige Merkmale.

Bei der Handschrifterkennung ist eine ausschließlich auf strukturellen Modellen basierende Klassifikation aufgrund der vielen Klassen und der hohen Varianz nicht sehr verbreitet. Allerdings werden einfache strukturelle Merkmale zur Reduktion der potentiellen Klassen eingesetzt, um so der eigentlichen Klassifizierung eine Vorklassifizierung in Form einer Taxonomie vorzuschalten [39]. Aufgrund der oben dargelegten Eigenschaften erscheint dieses Vorgehen bei arabischer Handschrift sinnvoll, da sich viele der Buchstaben tatsächlich nur durch ihre diakritischen

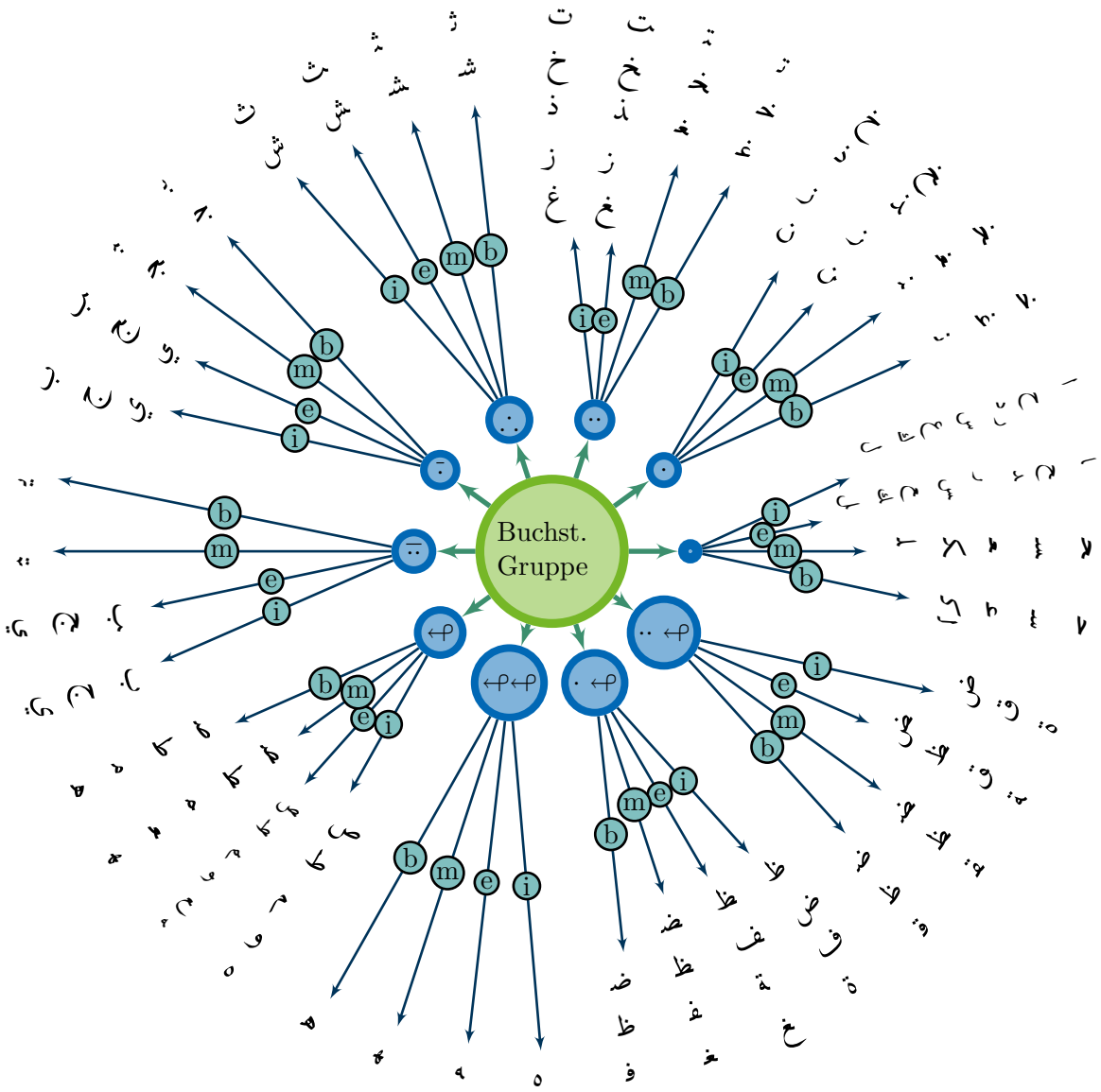


Abbildung 6.2. Taxonomien der arabischen Buchstabenklassen. Visualisiert wird der zur Vorklassifizierung eingesetzte Entscheidungsbaum, welcher folgende Merkmale einsetzt: diakritisches Zeichen · (1-3 Punkte), Position des diakritischen Zeichens befindet sich unterhalb der Basislinie ̄, Buchstabe enthält Schleife ↩ (1-2). Die Gruppen sind gegebenenfalls von den vier Positionen abhängig (**i**soliert, **b**eginnend, **m**ittig, **e**ndend).

Zeichen unterscheiden, welche theoretisch robust zu detektierende strukturelle Merkmale darstellen. Es ist jedoch zu bedenken, dass in der Praxis gegebenenfalls schlecht platzierte oder entsprechend des persönlichen Schreibstils stark abweichend bzw. falsch dargestellte diakritische Zeichen auftreten.

Konzeption der Vorklassifizierung. Aufgrund der beobachteten Eigenschaften arabischer Handschrift wurde entschieden, mit Hilfe der diakritischen Zeichen sowie auftretender Schleifen (z.B. bei ه oder ص), einen Entscheidungsbaum aufzustellen. Mit diesem wird die Anzahl der Klassen-Kandidaten eingeschränkt, indem jeder Buchstaben mindestens einer Gruppen zugewiesen wird. Die dafür benötigten strukturellen Merkmale werden durch einfache Heuristiken detektiert. Da die Existenz bzw. Anzahl der Schleifen in einigen Fällen von der Position des Buchstabens abhängt, erfolgt für alle vier möglichen Positionen eine separate Definitionen der Buchstabengruppen. Eine grafische Darstellung des verwendeten Entscheidungsbaumes findet sich in Abbildung 6.2.

Durch die Reduzierung der Klassen-Kandidaten auf Buchstaben mit unterschiedlichem Hauptkörper wird es ermöglicht, die diakritischen Zeichen bei der Normierung der Buchstabenproben zu entfernen. So lassen sich eindeutigere Merkmale extrahieren, da einige Störfaktoren entfallen, wie beispielsweise die variierende Distanz zwischen diakritischem Zeichen und Hauptkörper. Inwiefern sich in der Praxis eine robuste Vorklassifizierung realisieren lässt, hängt jedoch stark von der jeweiligen Handschrift ab. Werden diakritische Zeichen, insbesondere Gruppen von zwei oder drei Punkten, als Strich bzw. Kringel dargestellt, oder in Richtung des benachbarten Buchstabens verschoben, führt dies oft dazu, dass die falsche Gruppe von Klassen-Kandidaten selektiert wird. In diesem Fall ist eine anschließende korrekte Klassifizierung des Buchstabens ausgeschlossen. Daher ist es gegebenenfalls sinnvoller, die im Folgenden beschriebenen Merkmale und Klassifikatoren ohne Vorklassifizierung einzusetzen.

6.1.2 Statistische Merkmale

Da im arabischen über hundert Buchstabenklassen vorliegen, eignen sich statistische Merkmale prinzipiell besser für die Buchstabenerkennung, als strukturelle. Zudem lassen sie sich als Eingabe für verschiedene Klassifikatoren verwenden. Im Gegensatz zu Convolutional-Neural-Networks, bei denen die Merkmalsextraktion

und -selektion intern erfolgt, benötigen Klassifikatoren wie SVMs vorher berechnete und normierte Merkmalsvektoren. Eine Übersicht der hier untersuchten Merkmalsgruppen gibt Tabelle 6.1.

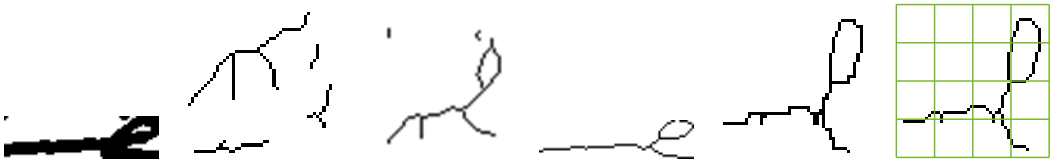
Tabelle 6.1. Übersicht der Verschiedenen Merkmalsgruppen.

Kategorie	Merkmalsvektor Bezeichnung	Länge des Merkmalsvektors
Global	KF-simple	40
	baseline	1
Momente	Momente	155
Gradienten	Gabor	100
	F74	83
	Im15	15
	CHi	96

Normierung der Proben. Das Seitenverhältnis der Buchstabenproben wird vor der Merkmalsextraktion für gewöhnlich normiert. Dabei treten bei rein bildbasierten Methoden jedoch oft Artefakte auf, insbesondere falls die Buchstaben eine komplexe Topologie aufweisen. Eine erhöhte Linienbreite und eine starke Streckung oder Stauchung der Buchstaben erhöhen dieses Risiko beträchtlich. Abbildung 6.3 (a-c) zeigt einen solchen Fall für einen Buchstaben, der aus einem Wort der IESK-arDB segmentiert wurde.

Um das Entstehen derartiger Artefakte zu verhindern, wurde ein Verfahren entwickelt, welches das Bild des segmentierten Buchstabens zunächst ausdünnt und seine Kontur anschließend in ein Polygon $\in \mathbb{R}^n$ umwandelt. Dieses wird hierauf auf das Zielformat skaliert (hier $46 \times 46 + 4$ Randbildpunkte). Das Polygon wird durch Interpolation auf ein Bild übertragen, so dass eine durchgängige Linie entsteht. Anschließend wird das Bild wiederum ausgedünnt, worauf die Normierung abgeschlossen ist und sich, wie in Abbildung 6.3 (f) dargestellt, Merkmale aus einzelnen Zellen extrahieren lassen.

Viele Merkmale, die bei der Buchstabenerkennung eingesetzt werden, erfordern eine solche Vorverarbeitung. Bei der Erkennung von Handschrift ist hauptsächlich die Linienführung entscheidend. Flächen, Farben und Texturen sind hier – im Gegensatz zu vielen anderen Klassifizierungsproblemen – irrelevant. Daher werden



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

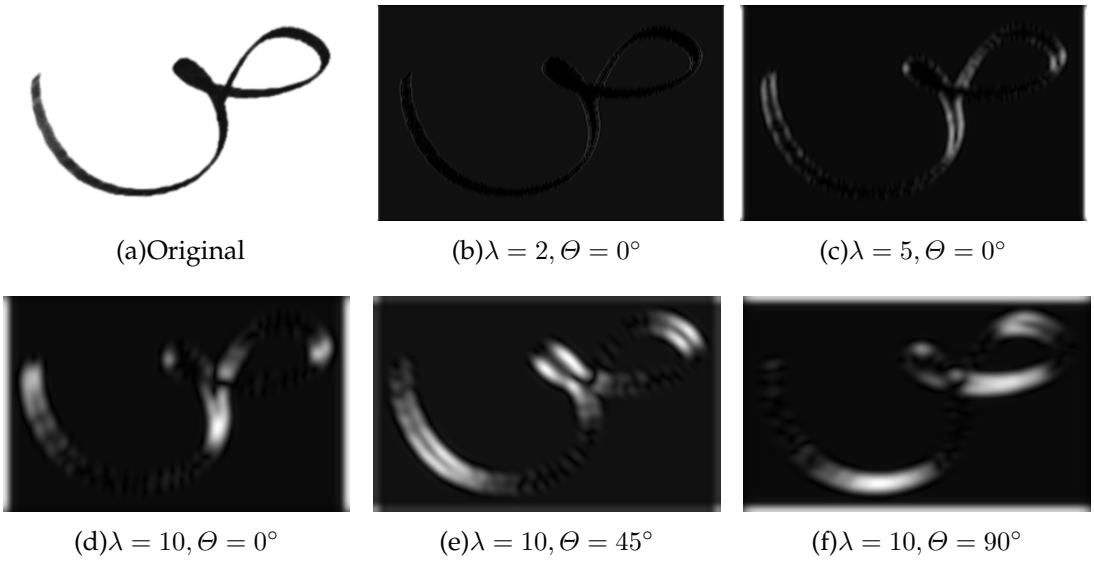
Abbildung 6.3. Normalisierung der Buchstabenproben für SVMs. (a) Segmentierter Buchstabe ـﺠـ. (b) Ergebnis nach Normierung des Bildes und anschließender Ausdünnung. (c) Ergebnis nach Ausdünnung, Normierung und erneute Ausdünnung. (d) Auf der Kontur des ausgedünnten Bildes basierendes Polygon. (e) Die aus (d) resultierende auf 50×50 Bildpunkte normierte Buchstabenprobe. (f) Nach einem weiteren Ausdünnen wird die Probe in (hier 4×4) Zellen aufgeteilt, aus denen anschließend verschiedene statistische Merkmale extrahiert werden.

oft Kantenmerkmale aus der 4er bzw. 8er Nachbarschaft des normierten Bildes extrahiert.

Momente. Momente fassen statistische Eigenschaften eines Bildes in jeweils einem Merkmal zusammen und können dabei translations- skalierungs- sowie rotationsinvarient sein. Es wird davon ausgegangen, dass wesentlichen Abweichungen der Rotation von Buchstaben bereits während der Vorverarbeitung behoben wurden und geringfügige Abweichungen ebenso in den verwendeten Trainingsdaten vorkommen. Zudem stellt die Orientierung ein wesentliches Merkmale vieler arabischer Buchstaben dar. Daher sind z.B. die rotationsinvarianten Momente von Hu hier nicht geeignet. Stattdessen werden für jede Zelle die translationsinvarianten zentralen und spatialen Momente berechnet [78].

Gradientbasierte Merkmale. Vielen der häufig für die Schrifterkennung eingesetzten Merkmalen ist gemein, dass sie die Winkel der im Schriftzug befindlichen Kanten repräsentieren. Eine allgemein einsetzbare Methode nutzt Gabor-Filter, um verschiedene Gradienten- aus Grauwertbildern zu extrahieren. Spezialisierte Methoden für die Schrifterkennung basieren dagegen meist auf der 8-er oder 4-er Nachbarschaft ausgedünnter Bilder, da Handschrift fast ausschließlich eindimensionale Merkmale enthält.

Gabor- Filter basierte Merkmale. Gabor-Filter $g(x,y;\lambda,\Theta,\psi,\sigma)$ erlauben die Extraktion von Kanten aus Grauwertbildern. Daher ist hier keine Binarisierung und Ausdünnung der Probe erforderlich. Der gesuchte Winkel Θ und die Wellenlänge λ (entspricht der Kantenstärke) sind stufenlos wählbar. Daher sind Gabor-Filter



(a)Original (b)$\lambda = 2, \Theta = 0°$ (c)$\lambda = 5, \Theta = 0°$

(d)$\lambda = 10, \Theta = 0°$ (e)$\lambda = 10, \Theta = 45°$ (f)$\lambda = 10, \Theta = 90°$

Abbildung 6.4. Ergebnisse nach Anwendung des Gabor-Filters auf ein Bild eines arabischen Buchstabens für unterschiedliche Parametrisierungen für λ und Θ. (b-d) Sofern der Gabor-Filter nicht auf ausgedünnte Bilder angewendet wird, ist die optimale Wellenlänge nicht bekannt oder variiert lokal. In diesem Fall werden unterschiedliche Wellenlängen verwendet, wodurch der resultierende Merkmalsvektor wächst. (e-f) Durch unterschiedliche Werte für Θ können Vorkommen von Kanten mit entsprechendem Winkel im Buchstaben detektiert werden.

vielseitig einsetzbar.

Abbildung 6.4 zeigt die Ergebnisse unterschiedlicher Parametrisierungen im Fall einer Probe, die nicht durch Vorverarbeitungsoperationen normiert wurde. Da sich Merkmale für mehrere Wellenlängen kombinieren lassen, sind Gabor-Filter in der Lage, auch mit der Linienstärke zusammenhängende Merkmale zu extrahieren (z.B. kleine Schleifen oder sich überschneidende Linien). Allerdings hängt die Linienbreite auch stark von dem verwendetem Schreibutensil ab, und ist daher kein zuverlässiger Indikator der Buchstabenklasse.

Für die Experimente in Kapitel 8 wird die Parametrisierung

$$\Theta = 5, \psi = 0.5, \sigma = 0.5, \tag{6.1}$$

verwendet, welche bei der IESK-arDB$_{\mathsf{OnlineLetter}}$ die besten Ergebnisse erzielt. Für die Winkel $0°, 45°, 90°, 135°$ und $180°$ wird je ein eigenes Bild berechnet. Anschließend wird dieses in 5×5 Zellen eingeteilt, aus denen jeweils die gemittelte Intensität als Merkmal extrahiert wird. Sofern die verwendeten Bilder vor Anwendung des Gabor-Filters nicht ausgedünnt wurden, ist die Berechnung für unterschiedliche Wellenlängen λ zur Detektion variabler Linienbreiten sinnvoll. Dies erhöht jedoch den Berechnungsaufwand und die Wahrscheinlichkeit redundanter und irrelevanter

Abbildung 6.5. Kodierung der 8-Nachbarschaft des mit x gekennzeichneten Bildpunktes als Chaincode.

Merkmale im Merkmalsvektor.

F74 – Chaincode basierte Kantenmerkmale. Ähnlich den Gabor-Filter basierten Merkmalen, lassen sich aus dem Histogramm des Chaincodes Merkmale extrahieren. Hier wird das normierte Bild zunächst in 3×3 Zellen eingeteilt. Das Histogramm des Chaincodes (d.h. der 8-er Nachbarschaft), welches sich aus den zur Schrift gehörenden Bildpunkten der jeweiligen Zelle ergibt, wird an den Merkmalsvektor angehängt. Da keine Richtungsinformation vorliegt, lässt sich das Histogramm entsprechend Abbildung 6.5 auf Chaincodes $\in \{0,1,2,3\}$ reduzieren.

Parkins et al. haben diese Merkmale zur Erkennung von Ziffern eingesetzt, indem zuvor eine Merkmalsreduktion mittels evolutionärer Algorithmen durchgeführt wird [76]. Der Informationsgehalt dieser Merkmale entspricht im wesentlichen denen der Gabor-Filter basierten Merkmalen, sie sind jedoch für ausgedünnte Bilder von Schriftzeichen konzipiert und lassen sich für diese effizienter berechnen.

Im15. Shanbehzadeh et al. beschreiben einen kompakten Satz von fünfzehn auf dem ausgedünnten Bild basierenden Merkmalen zur Erkennung von Buchstaben des Farsi Alphabetes [86]. Diese Merkmale basieren teilweise auf dem Prinzip der Momente. Es werden jedoch auch einfache Strukturen durch Filtermasken detektiert, um grade Linien und Ecken zu induzieren.

CHi – online Merkmale. Um einen Chaincode mit allen acht der in Abbildung 6.5 dargestellten Werte zu nutzen, ist eine Rekonstruktion der Trajektorien erforderlich, um so die verlorengegangene Schreibrichtung von offline Proben wiederherzustellen. Ähnlich wie bei der Basislinie, lässt sich eine solche Rekonstruktion zwar hinlänglich zuverlässig während der Segmentierung eines Wortes in Buchstaben durchführen, jedoch nicht beim Training mit einer Buchstabendatenbank, die keine speziell hierfür konzipierten Grundwahrheiten aufweist. Daher wird diese Merkmalsgruppe – im Gegensatz zu allen anderen – auf Grundlage der Trajektorien der IESK-arDB$_{\text{OnlineLetter}}$ berechnet, anstatt die binären oder normierten Bildrepräsentationen zu verwenden. Hiermit wird gleichsam bezweckt, den offline Ansatz mit einem simplen online Verfahren zu vergleichen.

Die Extraktion der Chaincodemerkmale ist mit den F74 Merkmalen vergleichbar, jedoch werden die Proben hier nicht in Zellen aufgeteilt. Stattdessen werden die Trajektorien in 5 gleichgroße Abschnitte zergliedert, aus denen jeweils das vollständige Histogramm des Chaincodes berechnet und dem Merkmalsvektor hinzugefügt wird.

Globale Merkmale – die Merkmalsgruppe KF-simple. Viele statistische Merkmale werden aus normalisierten Bildern der Buchstabenproben extrahiert, wie anhand Abbildung 6.3 (f) auf Seite 132 veranschaulicht wurde. Da bei der Normierung das Seitenverhältnis verlorengeht, dieses aber ein wichtiges Merkmal zur Unterscheidung einiger Klassen darstellt (wie z.B. bei ن und ت), wird es zuvor als zusätzliches Merkmal gespeichert (Merkmal 10). Alle weiteren globalen Merkmale, die in Tabelle 6.2 aufgelistet sind, werden aus dem normierten Bild gewonnen.

Tabelle 6.2. Auflistung aller Merkmale der Merkmalsgruppe "KF-simple".

Merkmal	Beschreibung
1	Anzahl der zusammenhängende Komponenten (CCs) (induziert Vorkommen diakritischer Zeichen)
2-3	Durchschnitt und Standardabweichung der Fläche der CCs (μ,σ)
4	Anzahl der Schleifen
5-6	Relative Position im normiertem Bild (x,y)
7	Anzahl der Endpunkte (EPs)
8	Anzahl der Verzweigungspunkte BPs
9	Normierte Anzahl der Bildpunkte des Buchstaben
10	Seitenverhältnis des Buchstaben vor der Normierung
11	Normierte Länge des größten CC (potentieller Hauptkörper)
12	Durchschnittliche Länge der restlichen CCs (0, falls keine existieren)
13	Merkmal 11 dividiert durch Merkmal 12
14-17	Relative Position (x,y), Breite und Höhe des größten Fragments
18	Relative Überschneidung des grössten mit den restlichen CCs auf der $y-$ Achse
19-22	Durchschnittliche relative Position der kleineren CCs (μ,σ von x und y), 0 falls nicht vorhanden
23-24	Relative Breite und Höhe des die kleineren CCs umgebenden Rechtecks
25	Relative Überschneidung des größten mit den restlichen CCs auf der $x-$ Achse

26-27	Differenz der durchschnittlichen relativen Breite und Höhe des umgebenden Rechtecks des Größten und der kleineren CCs
28-36	Anzahl an BPs und EPs im oberen, unterem, linkem und rechtem Bereich des Bildes (erlaubt z.B. die Differenzierung von ـٖـ und ـٓـ)
37	Relative Position des geometrischen Mittelpunkts aller Schleifen auf der $y-$Achse

Die ersten elf Merkmale wurden mit dem Ziel entwickelt, potentielle Kandidaten für strukturelle Merkmale als statistische Merkmale zu interpretieren. Der Grund hierfür ist, dass die involvierten Elemente (diakritische Zeichen und Key-Features wie Verzweigungspunkte) in den untersuchten handschriftlichen Proben zu unregelmäßig auftreten, um aus ihnen zuverlässige strukturelle Modelle zu bilden. Merkmal 9 ergänzt diese Information, indem die Anzahl an Vordergrundbildpunkten im normiertem Bild durch die Gesamtanzahl an Bildpunkten dividiert wird. Hierdurch lassen sich komplexe Buchstaben wie ض von schlichteren wie ر unterscheiden.

Es wurde festgestellt, dass mit den oben beschriebenen Merkmalsgruppen, wie Momenten oder gradientenbasierten Merkmalen, ohne Vorklassifizierung in erster Linie Buchstaben mit gleichem Hauptkörper verwechselt werden. Wie bereits erwähnt, erzielt die heuristische Vorklassifizierung jedoch keine guten Ergebnisse, wenn die diskreten strukturellen Merkmale zu sehr von der Norm abweichen. Daher werden die im normiertem Bild gefundenen zusammenhängende Komponenten (CCs) hier nicht explizit in diakritischen Zeichen, Supplemente und Hauptkörper bzw. Fragmente desselben eingeteilt. Stattdessen werden verschiedene Eigenschaften der CCs extrahiert, die insbesondere zur Differenzierung von Klassen mit gleichem Hauptkörper dienen sollen (Merkmale 11-37).

Basislinie. Ein spezielles Merkmal ist der Abstand der Basislinie zum oberen bzw. unteren Endes eines Buchstabens, denn er lässt sich nicht aus dem Bild eines segmentierten Buchstaben ermitteln. Dennoch handelt es sich um ein wichtiges Merkmal, durch das sich insbesondere die ansonsten sehr ähnlichen Buchstaben د und ر gut differenzieren lassen. Dieses Merkmal lässt sich zwar vor der eigentlichen Merkmalsextraktion als Nebenprodukt der Basislinienbestimmung und Segmentierung berechnen, allerdings liegt die benötige Information nicht für alle zum Training

konzipierten Buchstabendatenbanken wie z.B. der IESK-arDB$_{\text{OnlineLetter}}$ vor. Daher wird dieses Merkmal separat behandelt und nicht dem globalem Merkmalsvektor hinzugefügt.

6.1.3 Merkmalsnormierung und -selektion

SVMs benötigen für eine effektive Berechnung der Hyperebenen normierte Merkmalsvektoren. Da sich nicht alle Merkmale direkt normieren lassen, wird hierzu Standarisierung eingesetzt:

$$x_i = \frac{x'_i - \mu'_i}{\sigma'_i}. \tag{6.2}$$

Diese Normierung basiert auf den gemessenen Werten x'_i der Trainingsproben. Sie liefert gute Ergebnisse, sofern für sämtliche Klassen genügend Trainingsproben vorliegen und die Merkmale der Testproben keine dramatischen Abweichungen aufweisen. Gespeichert werden zu jedem Merkmal x_i der Erwartungswert μ'_i und die Standardabweichung σ'_i. Dies erfolgt separat für jede Kombination von verwendeten Merkmalen und für jede der vier Buchstaben Positionen.

Eine Gewichtung oder Merkmalsselektion ist bei vielen Klassifikatoren, wie SVMs, nicht zwingend erforderlich, da der Einfluss wenig relevanter Merkmale auf die Klassifizierung automatisch angepasst wird. Zur Reduktion von umfangreichen, mutmaßlich redundanten Merkmalsvektoren, wird jedoch die Hauptkomponentenanalyse (PCA) eingesetzt. In Kombination mit SVMs lässt sich hierbei jedoch weder eine Steigerung der Geschwindigkeit noch der Treffergenauigkeit der Buchstabenerkennung feststellen. Allerdings lässt sich der Umfang der meisten der oben beschriebenen Merkmalsvektoren auf ca. 30% reduzieren, bevor eine Verschlechterung der Treffergenauigkeit eintritt.

6.2 Klassifikation

Durch die Kombination der verschiedenen Merkmale, lässt sich das Bild eines segmentierten Buchstaben als beobachteter Merkmalsvektor x zusammenfassen. Das Ziel eines Klassifikators ist es nun, durch die Beobachtung x darauf zu schließen, welcher Klasse der Buchstabe zuzuordnen ist.

6.2.1 Auf Support Vector Machines basierte Klassifikation

Support Vector Machines (SVMs) gehören zu den etablierten Klassifikatoren zur Auswertung statistischer Merkmale. Sie lassen sich im Vergleich zu z.B. Hidden Markov Modellen oder neuronalen Netzen schnell trainieren, was die flexible Verwendung neuer Merkmalsvektoren oder Trainingsproben vereinfacht. Zudem erfolgt die Klassifikation sehr schnell und zuverlässig.

Im Rahmen dieser Doktorarbeit wurden SVMs mit nichtlinearem RBF-Kernel eingesetzt, welche im Allgemeinen eine gute Trennung der Klassen im Merkmalsraum ermöglicht und auch für das untersuchte Problem die besten Ergebnisse liefert. RBF basierte SVMs erfordern die Optimierung der Parameter C und γ, welche mittels GridSearch und Kreuzvalidierung für jeden Merkmalsvektor und für die jeweils verwendeten Trainings- und Validierungsproben umgesetzt wird (siehe Kapitel 8.2.1). Da unter anderem für die Fehlererkennung ein Ranking aller Klassen erforderlich ist, wurde die LIBSVM Implementation ausgewählt. Diese bietet Multiklassen SVMs, welche bei der Auswertung des Merkmalsvektors zu jeder Klasse die Log-Likelihood $\mathcal{L}(\theta|x) = \ln\big(L(\theta|x)\big)$ wiedergeben.

Für alle SVMs (und auch ASMs) wurden jeweils die Proben von einem Schreiber für die Evaluierung verwendet und der Rest zum Trainieren und Validieren eingesetzt. Hierdurch lässt sich ermitteln, wie robust die verwendeten Merkmale gegenüber schreiberabhängigen Variationen der Handschrift sind. Allerdings deckt dies nicht zwangsläufig alle beim Segmentieren von Wörtern in Buchstaben auftretenden Variationen ab. Problematisch ist insbesondere, wenn die Buchstaben an Verzweigungspunkten segmentiert werden, was eine im Vergleich zu den Trainingsproben erhöhte Varianz der Kashidas bewirkt. Wie in Kapitel 8 gezeigt wird, lässt sich diesem Problem durch Modifizierung der ursprünglichen Trainingsproben nur bedingt beikommen, weshalb der ASMs basierte Ansatz durch seine Adaptivität diesbezüglich von Vorteil ist.

Einsatz der Vorklassifizierung bei SVMs. Zunächst wurden ähnlichen Erkennungsraten mit und ohne vorgeschalteter Vorklassifizierung gemessen. Mit den optimierten Merkmale aus Tabelle 6.2 lassen sich jedoch ohne Vorklassifizierung bessere Ergebnisse erzielen, da sich auch Buchstaben mit irregulär dargestellten diakritischen Zeichen bei ausreichend Trainingsdaten korrekt klassifizieren lassen.

Außerdem ermöglichen die resultierenden, vollständigen Rankings eine zuverlässigere Korrektur von Erkennungsfehlern durch a priori Wissen. Daher werden im Weiteren SVMs ohne Vorklassifizierung favorisiert.

6.2.2 Active Shape Modell basierte Klassifikation

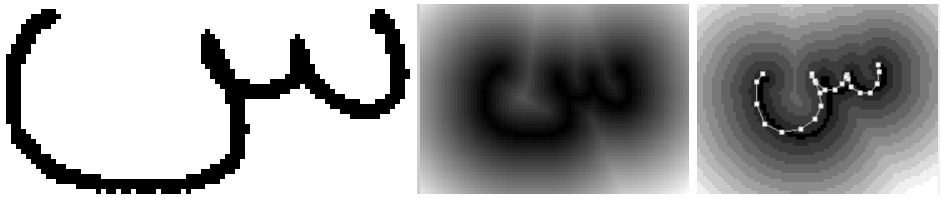
Im Allgemeinen werden Active Shape Modells (ASMs) häufig zur Lokalisation oder zum Tracking von Objekten und Gliedmaßen eingesetzt. Dies setzt eine geeignete Initialisierung sowie die iterative Anpassung des ASM an das Ziel voraus. Ist das Problem hingegen die Erkennung arabischer Buchstaben, so ist bereits eine geeignete Initialisierung bekannt, nicht jedoch die Klasse des Ziels. Daher werden die ASMs aller Buchstabenklassen (ohne diakritische Zeichen) initialisiert und iterativ an die Buchstabenprobe angepasst, um die Ähnlichkeit zu allen Klassen-Kandidaten zu ermitteln.

Die Kosten der ASM basierten Klassifikation sind abhängig von der Anzahl an Iterationsschritten der gewählten Optimierungsmethode, jedoch in jedem Fall wesentlich höher als bei Support Vector Machines (SVMs). Ein Vorteil der ASMs ist jedoch, dass sie sich auch nach ihrer Berechnung modifizieren lassen. So lassen sich etwa die Kashidas durch zusätzliche Freiheitsgrade des ASM manuell oder automatisch an die verwendete Segmentierungsmethode anpassen.

Einsatz der Vorklassifizierung bei ASMs. Aufgrund der rechenaufwendigen Klassifizierung mit ASMs wird diese im Folgenden immer mit Vorklassifizierung kombiniert. Hierdurch wird einerseits die Anzahl der Klassen auf einige Klassen-Kandidaten reduziert, andererseits lassen sich vereinfachte ASMs ohne diakritische Zeichen verwenden.

Um die ASMs zur Klassifikation einzusetzen, wird für alle Klassen-Kandidaten mittels des in Kapitel 2.10.1 beschriebenen Verfahrens eine geeignete Glyphe $\mathbf{\acute{U}}$ (siehe Gleichung 3.10) mit maximaler Korrelation zu einer Buchstabenprobe $\mathbf{I}$ gesucht. $\mathbf{I}$ muss hierzu zunächst in eine geeignetere Form überführt werden.

Chamfer Distanz Transformation. Um mit geringen Kosten eine Korrelation zwischen dem ASM und einer Buchstabenprobe $\mathbf{I}$ zu berechnen, wird die Probe zunächst unter Beibehaltung des Seitenverhältnisses normiert. Anschließend wird



(a) (b) (c)

Abbildung 6.6. Beispiel der Chamfer Distanz Transformation. (a) Buchstabenprobe (Binärbild). (b) Chamfer transformiertes Bild $\mathbf{I}_{\mathsf{chamf}}$ mit einem Rand von zwanzig Bildpunkten. (c) Initialisiertes ASM ($\acute{\mathbf{U}}$) über dem zur besseren Visualisierung quantisierten Chamfer Bild.

sie mit der Chamfer Distanz Transformation (CDT) in $\mathbf{I}_{\mathsf{chamf}}$ überführt, wie Abbildung 6.6 veranschaulicht. Der Vorteil der CDC liegt in ihrem robustem Verhalten gegenüber Rauschen und unruhigen Kanten [88]. Die Intensität jedes Bildpunktes $\mathbf{I}_{\mathsf{chamf}}(x,y)$ ergibt sich aus der Distanz von diesem Punkt zum nächstgelegenem Punkt $\mathbf{p_i}$ einer Kante aus $\mathbf{I}$:

$$\mathbf{I}_{\mathsf{chamf}}(\acute{u}_1, \acute{u}_2) = \delta(\acute{\mathbf{u}}) = \arg\min_{\mathbf{p}_i} \|\acute{\mathbf{u}} - \mathbf{p}_i\| \,. \tag{6.3}$$

Liegt die Buchstabenprobe als Sequenz vor, lässt sich die Berechnung von $\mathbf{I}_{\mathsf{chamf}}$ entsprechend beschleunigen. Für die Experimente wurde $\mathbf{I}_{\mathsf{chamf}}$ außerdem zu allen Buchstabenproben der IESK-arDB$_{\mathsf{OnlineLetter}}$ vorberechnet und der Datenbank hinzugefügt.

Da Punkte $\acute{\mathbf{u}} \in \acute{\mathbf{U}}$ des optimierten ASM oft auch außerhalb des umgebenden Rechtecks der ursprünglichen Initialisierung liegen, wird dem Bild $\mathbf{I}_{\mathsf{chamf}}(x,y)$ ein Randbereich hinzugefügt. Liegt ein Punkt $\acute{\mathbf{u}} \in \acute{\mathbf{U}}$ dennoch außerhalb von $\mathbf{I}_{\mathsf{chamf}}$, so wird stattdessen der Wert des nächstgelegenen Randpunktes von $\mathbf{I}_{\mathsf{chamf}}$ zuzüglich der Distanz zwischen diesem Punkt und $\acute{\mathbf{u}}$ verwendet.

Abstandsmaße zwischen ASM und Chamfer Bild. Bei dem ASM basierten Verfahren sind für optimale Ergebnisse gegebenenfalls viele Iterationsschritte erforderlich. Zudem liegen die aus den Trainingsdaten berechneten ASMs nicht in derselben Form vor, wie die Proben. Daher wird davon abgesehen, für jede Iteration i die in den vorherigen Abschnitten beschriebenen Merkmalsvektoren für $\acute{\mathbf{U}}_i$ zu berechnen. Stattdessen werden speziell für das ASM konzipierte Abstandsmaße zur Bestimmung der Unähnlichkeit zwischen $\acute{\mathbf{U}}_i$ und der Buchstabenprobe $\mathbf{I}$ eingesetzt. Das

erste Maß

$$\mathfrak{f}_1 = \frac{|m_1 - m_2|}{|m_1 + m_2|}, \tag{6.4}$$

ergibt sich aus der zur Initialisierung des ASM nötigen Streckung oder Stauchung. Diese hängt von der Steigung des umgebenden Rechtecks des ASM (m_1) und der Buchstabenprobe (m_2) ab. Für die Folgenden Abstandmaße werden die Darstellungen des ASM aus Kapitel 2.10.1 und Kapitel 3.1 vorausgesetzt.

Mit $\varnothing$ als Diagonale von $\mathbf{I}_{\mathsf{chamf}}$ und der Anzahl von Punkten des ASM $n(\acute{\mathbf{U}})$ ergibt sich das zweite Maß wie folgt:

$$\mathfrak{f}_2 = \frac{\sum_{i=1}^{n(\acute{\mathbf{U}})} \mathbf{I}_{\mathsf{chamf}}\left(\acute{\mathbf{u}}_i\right)}{\varnothing\, n(\acute{\mathbf{U}})} \; |\text{mit Matrix } \mathbf{M}(\acute{\mathbf{u}}_i) = m_{\acute{u}_{i,1},\acute{u}_{i,2}}. \tag{6.5}$$

Die Wahrscheinlichkeit für eine Buchstabenprobe, dem Erwartungswert $\bar{\mathbf{x}}$ des ASM am ähnlichsten zu sehen, ist statistisch am höchsten, wenn Probe und ASM zur selben Buchstabenklasse gehören. Daher wird der Abstand

$$\mathfrak{f}_3 = \frac{\|\mathbf{c}\|}{\sqrt{m_t}}, \tag{6.6}$$

vom optimierten ASM zu seinem Erwartungswert als drittes Fehlermaß verwendet (m_t beziffert die Anzahl und $\mathbf{c}$ den Einfluss der Eigenvektoren, entsprechend Formel 3.3).

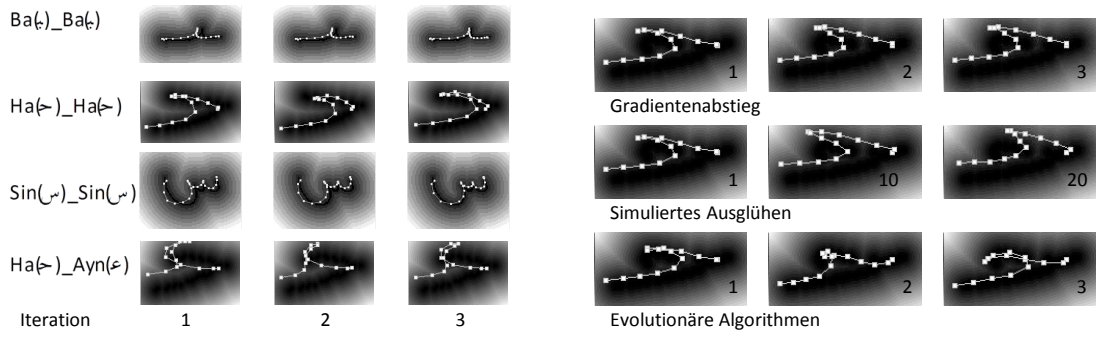
Das nächste Abstandsmaß ergibt sich aus den Abweichungen der Chamfer Distanzen zweier benachbarter Punkte des ASMs:

$$\mathfrak{f}_4 = \frac{1}{\varnothing} \sum_{i=1}^{n(\acute{\mathbf{U}})} \frac{|\mathbf{I}_{\mathsf{chamf}}\left(\acute{\mathbf{u}}_i\right) - \mathbf{I}_{\mathsf{chamf}}\left(\acute{\mathbf{u}}_{i+1}\right)|}{\mathbf{I}_{\mathsf{chamf}}\left(\acute{\mathbf{u}}_i\right) + \mathbf{I}_{\mathsf{chamf}}\left(\acute{\mathbf{u}}_{i+1}\right)}. \tag{6.7}$$

Eine hohe Differenz induziert die teilweise Übereinstimmung eines ASM, das nicht zur selben Klasse wie die Probe gehört.

ASMs einiger Buchstabenklassen (beispielsweise ر und د) ähneln Teilen anderer Buchstaben und würden sich daher gänzlich innerhalb eines Bereiches von $\mathbf{I}_{\mathsf{chamf}}$ mit niedriger Chamfer Distanz einfügen lassen. Daher wird ein weiteres Abstansmaß benötigt, welches ASMs, die $\mathbf{I}_{\mathsf{chamf}}$ nur teilweise abdecken, bestraft. Sei $\mathcal{I}$ die Gruppe aller Punkte aus $\mathbf{I}_{\mathsf{chamf}}$ mit $\mathbf{I}_{\mathsf{chamf}}(\mathfrak{i}) = 0, \mathfrak{i} \in \mathcal{I}$, so wird für jedes $\mathfrak{i}$ die Distanz zum nächsten $\acute{\mathbf{u}}$ wie folgt berechnet:

$$\mathfrak{f}_5 = \frac{1}{\varnothing n(\mathcal{I})} \sum_{i=1}^{n(\mathcal{I})} \arg\min_{\acute{\mathbf{u}}} \|\mathfrak{i}_i - \acute{\mathbf{u}}\| \tag{6.8}$$



(a) (b)

Abbildung 6.7. Darstellung von ASMs, die sich durch Optimierungsmethoden an eine Probe anpassen. (a) Beispiele für ASM basierte Klassifikation mittels Gradientenabstiegsverfahren (GD). (b) Vergleich der drei Optimierungsmethoden. Das ASM und die Buchstabenprobe gehören jeweils zur Klasse ح. Es ist jeweils die Iteration bzw. Epoche der Optimierung angegeben.

Aufgrund der hohen Kosten wird dieses Abstandsmaß nur für die letzte Iteration der im nächsten Abschnitt beschriebenen Optimierung der ASMs verwendet.

Optimierung der ASM Hyperparameter. Im nächsten Schritt werden auf Grundlage der Abstandsmaße $\mathfrak{f}_{1-5}$ die Hyperparameter $c_i \in \mathbf{c}$ zu den ASMs aller Klassen-Kandidaten dahingehend optimiert, dass das resultierende Ú die größtmögliche Übereinstimmung mit $\mathbf{I}_{\mathsf{chamf}}$ annimmt (siehe Kapitel 2.11). Hierbei gibt $\mathfrak{f}_i(\mathbf{c}, \mathbf{I}_{\mathsf{chamf}})$ den jeweiligen Abstand zwischen der Bildprobe und eines ASM mit der Parametrisierung $\mathbf{c}$ an. Somit lässt sich die Fitness von $\mathbf{c}$ wie folgt berechnen:

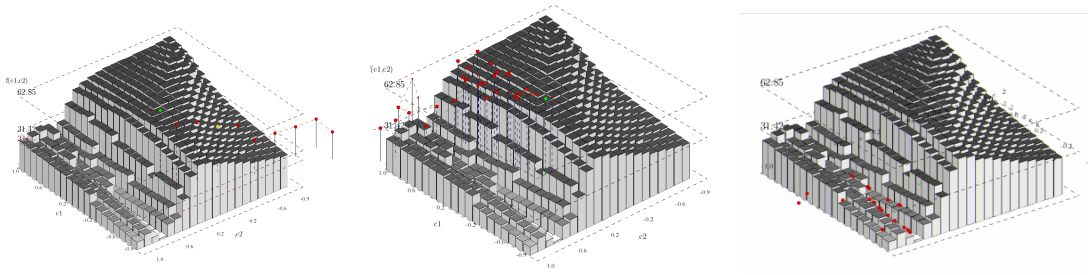
$$f_{\mathsf{fit}}(\mathbf{c}) = -\sum_{i=1}^{5} w_i \cdot \mathfrak{f}_i(\mathbf{c}, \mathbf{I}_{\mathsf{chamf}}) \; |c \in [-1{,}1]. \tag{6.9}$$

Hiernach lässt sich die normierte Fitness $f_{\mathsf{fit-norm}}(\mathbf{c})$ berechnen, indem $f_{\mathsf{fit}}(\mathbf{c})$ durch die höchste gemessene Fitness dividiert wird. Die Gewichte w_i der einzelnen Abstandsmaße wurden im Vorfeld durch einen Evolutionären Algorithmus optimiert. Das beste Ergebnis wurde hierbei mit

$$\mathbf{w} = (0{,}54; 0{,}59; 0{,}2; 0{,}69; 1) \tag{6.10}$$

erzielt.

Die Optimierung von $\mathbf{c}$ erfolgt wahlweise durch die im Stand der Technik beschriebenen Optimierungsverfahren: Graddientenabstieg (GD), Simuliertes Ausglühen (SA) oder Evolutionäre Algorithmen (EA). Dabei ist zu beachten, dass auch



(a)Gradientenabstieg (b)Simuliertes Ausglühen (c)Ev. Algorithmen, 3. Epoche

Abbildung 6.8. Visualisierung der Optimierungsverfahren mit einer Parametrisierung $\mathbf{c}$ der Länge zwei (die ASMs werden hier nur durch die ersten beiden Eigenwerte definiert, der Funktionswert entspricht $1 - f_{\text{fit\_rel}}(c_1, c_2)$, wobei $f_{\text{fit\_rel}}$ die über den Definitionsbereich normierte Fitness bezeichnet).

die Hyperparameter zu den Klassen-Kandidaten, die nicht mit der Klasse von $\mathbf{I}_{\text{chamf}}$ übereinstimmen, optimiert werden. Das Ziel besteht in diesem Kontext jedoch nicht allein in der bestmöglichen Optimierung von $\mathbf{c}$. Vielmehr wird bezweckt, dass die optimierte Fitness des zur selben Klasse wie die Probe gehörenden ASM höher als jene der anderen ASMs ist, wodurch die korrekte Klassifizierung von $\mathbf{I}_{\text{chamf}}$ resultiert. GD ist hier im Vorteil, da der Erwartungswert $\bar{\mathbf{x}}$ des ASM als Ausgangspunkt genutzt wird. Stammt $\mathbf{I}_{\text{chamf}}$ von derselben Klasse wie das ASM, so weist $\bar{\mathbf{x}}$ oft bereits eine hohe Ähnlichkeit zu $\mathbf{I}_{\text{chamf}}$ auf. Stammt $\mathbf{I}_{\text{chamf}}$ hingegen von einer anderen Klasse, so ist dies nicht der Fall, wodurch die Gefahr in einem lokalem Optimum hängen zu bleiben deutlich erhöht wird. Abbildung 6.7(a) zeigt einige Beispiele für die Optimierung der ASMs dreier Proben der gleichen und einer Probe mit abweichender Klasse. In Abbildung 6.7(b) ist dargestellt, wie unterschiedlich die Ergebnisse in Abhängigkeit des gewählten Optimierungsverfahrens ausfallen. Bei diesem Beispiel verharrt GD in einem lokalem Optimum nahe des Erwartungswertes, wohingegen EA nach drei Generationen eine deutlich bessere Anpassung an die Probe erzielen.

Abbildung 6.8 visualisiert den Hyperparameterraum für das auf c_1 und c_2 reduzierte Optimierungsproblem (es werden nur die beiden ersten Eigenvektoren genutzt). Das Verlassen des Intervalles $[-1,1]$ ist hierbei zum Erhalt der Stetigkeit erlaubt, wird jedoch mit einem Strafterm geahndet. Abbildung 6.8 (a) zeigt, dass das Gradientanabstiegsverfahren sehr direkt auf ein lokales Optimum zusteuert, das globale Optimum jedoch aufgrund des Startpunktes verpasst wird. Simuliertes Ausglühen ist vom Zufall abhängig, und hat daher mit dem selben Startpunkt eine

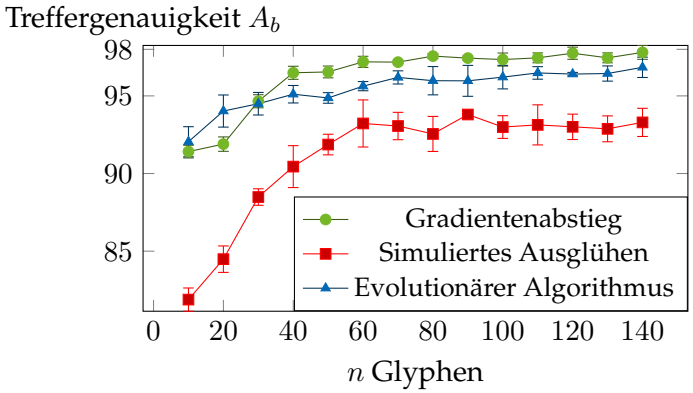


Abbildung 6.9. Optimierung der ASM basierten Klassifikation mit verschiedenen Verfahren. Hierzu wurde eine manuelle Vorklassifizierung verwendet. Zudem ist die Optimierung, im Gegensatz zu den Experimenten in Kapitel 8, nicht schreiberunabhängig (Trainings- und Testproben enthalten jeweils Proben aller Schreiber). Wie zu sehen ist, erzielt der verwendete Evolutionäre Algorithmus geringfügig und simuliertes Ausglühen merklich schlechtere Ergebnisse als das Gradientenabstiegsverfahren.

Chance, in die Nähe des globalen Optimums zu gelangen. Evolutionäre Algorithmen decken den Definitionsbereich in der ersten Epoche gut ab, und weisen in der dritten Epoche viele Individuen nahe des globalen Optimums auf.

Die Ergebnisse der verschiedenen Optimierungsverfahren in Abhängigkeit der nötigen Iterationen und Epochen – und der einhergehenden Anzahl generierter Glyphen $\acute{\mathbf{U}}$ – ist in Abbildung 6.9 dargestellt. Hieraus geht hervor, dass die besten Ergebnisse für das Gradientenabstiegsverfahren (GD) zu erwarten sind und mindestens 50-100 Generierungen von Glyphen für optimale Ergebnisse erforderlich sind.

Erweiterte Parametrisierung. Die bisher verwendeten Hyperparemeter c_i steuern den Einfluss der Eigenvektoren auf $\acute{\mathbf{U}}$. Falls es erforderlich ist, lassen sich zusätzliche Hyperparameter auch nach der Berechnung der ASMs hinzufügen. Diese eignen sich zum Beispiel zum Steuern affiner Transformationen. Bei den für die Experimente verwendeten Wortdatenbanken hat sich jedoch nur die Anpassung der Kashida-Länge als relevant erwiesen. Zusätzliche Hyperparameter – für Rotation, Translation und Skalierung – führten hingegen zu keiner weiteren Erhöhung der Treffergenauigkeit A_b. Daher wird $\mathbf{c}$ nur ein Hyperparameter hinzugefügt, der die Länge der Buchstabentrajektorie um 0-25% kürzt (siehe Kapitel 3.2.2). Eine manuelle Ettikettierung der Buchstabenproben bezüglich der exakten Trennpunkte von

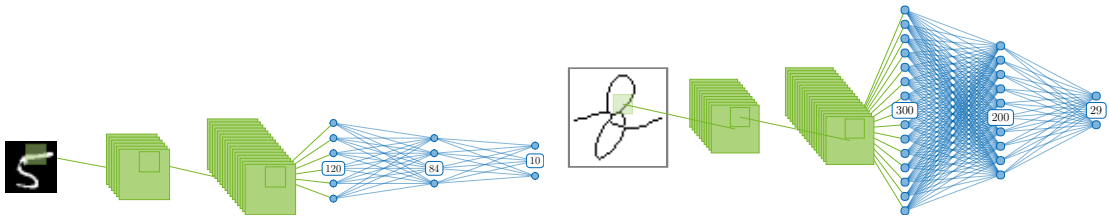


Abbildung 6.10. Darstellung der Architektur des Convolutional-Neural-Networks. Die Konvolutionsschichten werden grün, die vollverbundenen Schichten blau dargestellt. (a) Vorgegebene Architektur der Bibliothek Dlib zur Erkennung von Ziffern. (b) Zur Erkennung von arabischen Buchstaben modifiziert Architektur.

Kashida und eigentlichem Buchstaben war bisher nicht realisierbar, daher wurde der allgemeine Wert geschätzt und experimentell überprüft.

6.2.3 Convolutional-Neural-Networks basierte Klassifikation

Convolutional-Neural-Networks (CNNs) erfreuen sich als Klassifikatoren zunehmender Beliebtheit. Die Fähigkeit von CNNs, sowohl die Ermittlung von Merkmalen auf mehreren Ebenen, als auch deren Auswertung in den vollständig verbundenen Schichten zu trainieren, bietet viele Möglichkeiten. So können z.B. große, unsortierte Datenbanken zunächst zum Trainieren der Konvolutionsschichten eingesetzt werden, die sich später an für spezifische Probleme konzipierte *Fully Connected Layer* anfügen und weiter trainieren lassen (*Transfer Learning*). Die ersten Konvolutionsschichten erlauben die Detektion von einfachen Merkmale wie Kanten. Höhere Schichten setzen sich aus den jeweils zuvor detektierten Merkmalen zusammen, und sind deutlich komplexer, so dass gegebenenfalls Strukturen wie Augen oder sogar Gesichter resultieren (d.h. Merkmale können bereits Klassen sein). Daher bieten an umfangreichen Datenbanken vortrainierte Konvolutionsschichten ein großes Potential, um das Training diverser Erkennungsprobleme zu beschleunigen und gleichsam die Genauigkeit zu erhöhen, da bereits eine Vielzahl von komplexen Merkmalen gelernt wurde, die sich zur Lösung konkreter Probleme einsetzen lassen. Im Kontext sehr konkrete Klassifikationsprobleme mit weniger komplexen Merkmalen – wie die Erkennung einzelner arabischer Buchstaben oder Ziffern – ist jedoch davon auszugehen, dass sich die Vorteile von CNNs nicht optimal ausnutzen lassen.

Zum Vergleich der SVM und ASM basierten Ansätzen wurde die CNN Implementation der Dlib Bibliothek verwendet, welche beispielsweise zum Klassifizieren

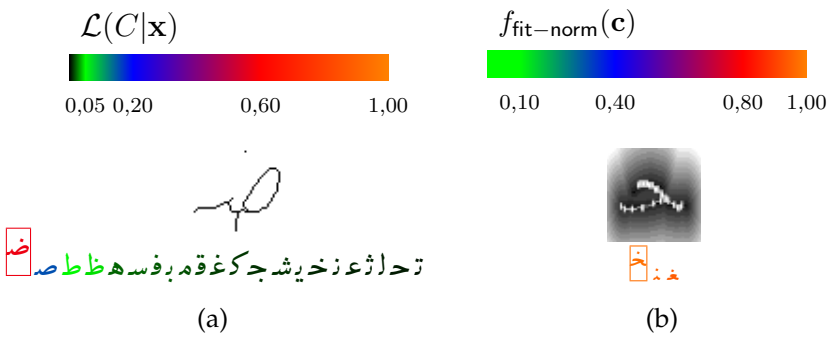


Abbildung 6.11. Beispiel für ein Ranking, das auf der (a) von einem SVM berechneten Likelihood und (b) auf der Korrelation eines ASM mit dem Buchstaben basiert (nachdem die Anzahl der Klassenkandidaten durch einen Entscheidungsbaum reduziert wurde).

von Ziffern eingesetzt wurde [1]. Da arabische Buchstaben eine ähnliche, jedoch etwas höhere Komplexität als Ziffern aufweisen, wurde die in Abbildung 6.10 (b) dargestellte Architektur des verwendeten CNN geringfügig modifiziert. Es wurde die Anzahl der parallelen Konvolutionsschichten erhöht, um mehr Merkmale zu trainieren. Zudem wurde die Knotenzahl der voll verknüpften Schichten entsprechend erhöht, auch um der höheren Anzahl an Klassen gerecht zu werden.

6.2.4 Ranking der Klassen-Kandidaten

Da sich handschriftliche Buchstaben in einigen Fällen weder durch Methoden der Bildverarbeitung noch von einem menschlichen Beobachter eindeutig einer bestimmten Klasse zuordnen lassen, ist eine Auflistung aller Klassen in einem Ranking sinnvoll. Dieses Vorgehen erlaubt es unter anderem, die Erkennungsergebnisse anschließend mit a priori Informationen wie Sprachmodellen zu kombinieren. Zudem ist zu jedem Eintrag des Rankings die Log-Likelihood $\mathcal{L}(C|\mathbf{x})$ der verwendeten Support Vector Machines beigefügt, aufgrund derer das Ranking erstellt wurde.

Ein Beispiel für ein komplettes SVM basiertes Ranking findet sich in Abbildung 6.11 (a). Die Höhe des erste Wertes $\mathcal{L}(C|\mathbf{x})$ des Rankings gibt hier Auskunft darüber, wie zuverlässig sich der Buchstabe klassifizieren lässt. Je geringfügiger dieser Wert jene aller folgenden Einträge des Rankings übersteigt, desto höher ist die Wahrscheinlichkeit, dass entweder das Klassifizierungsverfahren oder die Ausprägung der Handschrift für eine eindeutige Erkennung unzureichend ist und eine erhöhte Gefahr einer Fehlklassifizierung besteht. In der Praxis kommt es darüber hinaus vor, dass es sich bei der vermeintlichen Buchstabenprobe um das Resultat von Über- oder Untersegmentierungen handelt. Um diese Segmentierungsfehler in Zukunft

besser zu detektieren, ließen sich an dieser Stelle zusätzliche SVMs trainieren, die zwischen validen Buchstaben und einer Restklasse unterscheiden.

Werden Active Shape Modells zum Erstellen des Rankings eingesetzt, erfolgt zunächst eine Selektion der Klassen-Kandidaten durch die Vorklassifizierung. Hiernach wird zu allen Klassen-Kandidaten die normierte Fitness $f_{\mathsf{fit-norm}}(\mathbf{c})$ der zugehörigen optimierten ASMs für das Ranking eingesetzt. Einige der Gruppen enthalten nur wenige Klassen-Kandidaten, wie es z.B. bei Abbildung 6.11 (b) der Fall ist.

6.3 Zusammenfassung

In diesem Kapitel wurden verschiedene Merkmale und Klassifikatoren auf ihre Eignung zur Erkennung handgeschriebener arabischer Buchstaben hin untersucht.

Es wurden die in den Buchstaben vorhandenen strukturelle Merkmale analysiert, mit denen die Buchstabenklassen in einem Vorklassifizierungsschritt zunächst in einzelne Gruppen aufgeteilt werden. Zudem wurden diverse statistische Merkmale untersucht. Weitere statistische Merkmale wurden konzipiert, um die Vorklassifizierung zu ersetzen, falls die strukturellen Merkmale in der Handschrift nicht eindeutig ausgeprägt sind.

Als Alternative zur Auswertung statistischer Merkmalsvektoren mittels bekannter Klassifikatoren wie Support Vector Machines, wurde ein auf Active Shape Modells basierendes Klassifikationsverfahren vorgestellt. Beide Klassifikationsansätze erzeugen ein Ranking der Klassen, welches die Grundlage des im folgenden Kapitel beschriebenen Verfahrens zur Worterkennung bildet.

KAPITEL 7

Worterkennung mittels A Priori basierter Fehlerkorrektur

GEEIGNETE Verfahren zur Segmentierung arabischer Wörter in Buchstaben und zur anschließenden Erkennung derselben wurden in den vorhergehenden Kapiteln erläutert. Darauf aufbauend wird in diesem Kapitel die Nachverarbeitung zur Fehlererkennung und -behebung und letztlich zur Worterkennung dargelegt. Eine Übersicht hierzu gibt Abbildung 7.1.

In Abschnitt 7.1 wird zunächst ein n-Gramm basiertes Verfahren zur Fehlerdetektion und -korrektur vorgestellt. Hierauf behandelt Abschnitt 7.2 die Fehlerbehebung durch den Abgleich mit einem Vokabular. Vollständige Sprach- und Fehlermodelle werden im Rahmen dieser Dissertation nicht untersucht, sind jedoch als Bestandteil zukünftiger Projekte eingeplant. Abschließend wird in Abschnitt 7.3 – am Beispiel von Long-Short-Term-Memory Netzwerken – ein auf impliziter Segmentierung basierendes Verfahren zur Worterkennung beschrieben.

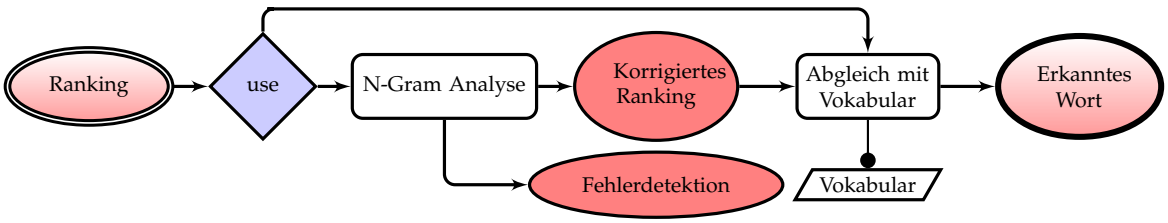


Abbildung 7.1. Vereinfachte Darstellung der Nachverarbeitung zur Worterkennung.

7.1 N-Gram basierte Fehlerdetektion und -korrektur

Um die Erkennung kompletter Texte zu unterstützen, empfiehlt es sich, neben Methoden der Bildverarbeitung auch a priori Wissen einzusetzen. In dieser Arbeit werden diesbezüglich n-Gramme von Buchstaben anhand der 50.000 häufigsten arabischen Wörter berechnet (siehe Kapitel 4.2) und zu Fehlerdetektion und -korrektur verwendet. Anhand dieses Vokabulars $\mathcal{V}_{50k}$ wurde Unigramme, Bigramme und Trigramme der arabischen Sprache untersucht. Die Verwendung höherer n-Gramme ist problematisch, da diese in zunehmend weniger Wörtern enthalten sind und der benötigte Speicherbedarf zudem exponentiell steigt.

7.1.1 Berechnung der n-Gramme

Unigramme enthalten die Wahrscheinlichkeit $P(C_i)$, mit der ein zufällig gewählter Buchstabe zur Klasse C_i gehört. Hier wird $P(C_i)$ aufgrund der Häufigkeit berechnet, mit der C_i in $\mathcal{V}_{50k}$ vertreten ist. Dazu wird das Maß $n(C_i, \mathbf{w}_{\mathcal{V}_j})$ benötigt, welches die Anzahl von Buchstaben der Klasse C_i innerhalb einer Vokabel $\mathbf{w}_{\mathcal{V}_j}$ angibt. Zusammen mit $n(\mathbf{w}_{\mathcal{V}_j})$, der Häufigkeit, mit der diese Vokabel im Textkorpus vorkommt, ergibt sich:

$$P(C_i) = \frac{\sum_{j=1}^{50k} n(C_i, \mathbf{w}_{\mathcal{V}_j}) n(\mathbf{w}_{\mathcal{V}_j})}{\sum_{C=ا}^{ي} \sum_{j=1}^{50k} n(C, \mathbf{w}_{\mathcal{V}_j}) n(\mathbf{w}_{\mathcal{V}_j})}. \tag{7.1}$$

Entsprechend werden die Wahrscheinlichkeiten der Bigramme $P(C_i|A)$ und Trigramme $P(C_i|BA)$ berechnet. Die Parameter A und B stehen hierbei für die Klasse der beobachteten vorherigen Buchstaben.

Es wurde festgestellt, dass $82{,}7\%$ aller theoretisch möglichen Bigramme[1] auch tatsächlich in $\mathcal{V}_{50k}$ vorkommen (d.h. es gilt $P(C_i|A) > 0$). Im Fall von Trigrammen sind es hingegen lediglich $33{,}98\%$. Außerdem weisen die Trigramme einen hohen Kontrast der Wahrscheinlichkeit $P(C_i|BA)$ auf. So ergibt sich bei gegebenem $A =$ص und $B =$غ eine bedingte Wahrscheinlichkeit von:

1. $P(\text{ي}|\text{غص}) = 0{,}88$ bei $P(\text{ي}) = 0{,}09$
2. $P(\text{ا}|\text{غص}) = 0{,}058$ bei $P(\text{ا}) = 0{,}18$

[1] Bei beliebiger Position der beiden Buchstaben. Die Verteilung von positionzspezifischen Bigrammen ist im Anhang in Abschnitt A.4 dargestellt.

Aufgrund dieses starken Kontrastes lassen sich Verwechslungen der Klasse des aktuellen Buchstaben (C) mit Trigrammen besser als mit Uni- oder Bigrammen prädizieren, und – sofern die Werte für A und B den tatsächlichen Buchstabenklassen entsprechen – auch korrigieren. Beispiele für valide Trigramme mit maximaler bis minimaler Wahrscheinlichkeit $P(C_i|BA) \neq 0$ finden sich in Tabelle 7.1.

7.1.2 Fehlerdetektion mit n-Grammen

Die a priori Wahrscheinlichkeit, dass drei zufällig gewählte, aufeinanderfolgende Buchstaben eines beliebigen Wortes die Parameter A, B und C aufweisen, ist durch

$$P(CBA) = P(C|AB) \cdot P(B|A) \cdot P(A), \tag{7.2}$$

gegeben. $P(CBA)$ gibt Aufschluss darüber, ob mindestens einer der Buchstaben falsch erkannt wurde, und eignet sich daher, um Rechtschreibfehler oder – im Fall von Handschrifterkennung – Segmentierungs- und Klassifikationsfehler zu detektieren. Diese sind beim Auftreten eines Trigramms mit $P(CBA) = 0$ anzunehmen, da ein solches nicht ein einziges Mal in $\mathcal{V}_{50k}$ vorkommt. Zudem ist die Wahrscheinlichkeit hoch, dass solche Trigramme auch in keinem sonstigen arabischen Wort vorkommen bzw. lediglich in Ausnahmefällen wie seltenen Namen oder Fremdwörtern.

Um auch bei validen Trigrammen mit $P(CBA) \neq 0$ eine Einschätzung von Fehlern zu ermöglichen, müsste zusätzlich ein umfangreiches Fehlermodell entsprechend der tatsächlich bei der Erkennung auftretenden Fehler erstellt werden. In dieser Promotion soll daher lediglich untersucht werden, wie viele der vorkommenden Fehler sich durch Detektion von Trigrammen mit $P(CBA) = 0$ ermitteln lassen. Dazu wird überprüft, ob in einer erkannten Sequenz von Buchstaben ein oder mehrere Trigramme mit $P(CBA) = 0$ vorhanden sind. Die Ergebnisse hierzu werden in Kapitel 8 vorgestellt.

Tabelle 7.1. Einige Beispiele für Trigrammen mit unterschiedlichen a priori Wahrscheinlichkeiten $P(C_i|BA)$.

Trigramm	ا ه غ	ر ط ض	ز د ظ	ل س غ	ة س ر	ي ح ش	ث م ت	ط ل ع
$P(C_i\|BA)$	1	0,98	0,78	0,5	0,21	0,08	0,016	0,00005

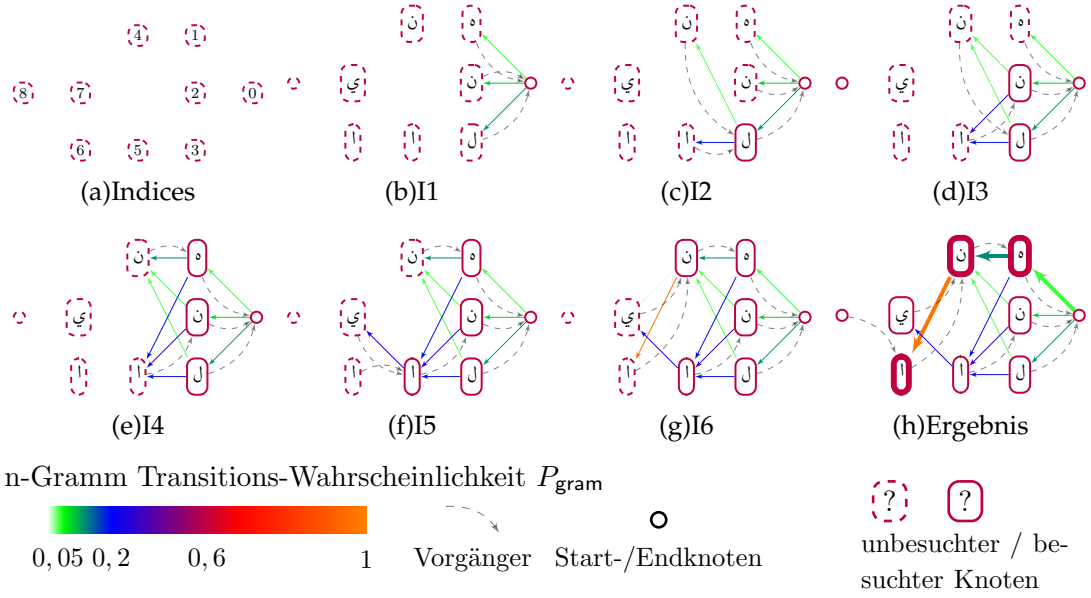


Abbildung 7.2. Vereinfachtes Beispiel zur Darstellung der Funktionsweise des modifizierten Dijkstra Algorithmus ($\forall \mathcal{L} = 0.5$, noch nicht gewichtet Kante werden ausgeblendet). (b-g) Iterationsschritte 1-6 (Iteration 5 ergibt den Pfad ل → ا=لَا, ein sehr häufiges Bigramm, welches zudem das Wort ‚nein' bildet.). (h) Visualisierung des Grafen nach Abschluss des Dijkstra Algorithmus: Der Pfad ه→ن→ا=هنَا ist ein häufiges Trigramm und zudem das arabische Wort für ‚hier'.

7.1.3 Fehlerkorrektur mit n-Grammen

In diesem Abschnitt wird erläutert, wie die erkennungsbasierten Rankings mit den a priori Wahrscheinlichkeiten der n-Gramme kombiniert werden. Hierzu werden Graphen konzipiert, deren Knoten die Log-Likelihoods $\mathcal{L}$ der Klassifikation enthalten und deren Kanten entsprechend der n-Gramme gewichtet werden. Eine abstrahierte Darstellung dieser Grafen ist in Abbildung 7.2 zu sehen. Abbildung 7.3 auf Seite 154 zeigt ein Beispiel für eine Fehlerkorrektur.

Unigramme $P(C)$ bilden die Gewichte der Kanten vom (leeren) Startknoten zu den Knoten des ersten Buchstaben $\mathfrak{b}_1$. Bigramme $P(C|A)$ werden hingegen zur Initialisierung der Gewichte der gerichteten Kanten eingesetzt, die je zwei vollwertigen Knoten verbinden. Bei der Traversierung wird das initiale Gewicht P_{gram} einer Kante (a priori Anteil) mit der Log-Likelihood $\mathcal{L}$ kombiniert, welche in dem Knoten gespeichert ist, auf den die Kante zeigt (merkmalsbasierter Anteil). Es ergibt sich hierdurch folgende Distanz zwischen einem Knoten und dem Startknoten:

$$\delta_{\mathfrak{n}_i} = 1 - \left(\lambda P_{\text{gram}} \cdot (1-\lambda)\mathcal{L}\right) + \delta_{\mathfrak{n}_{\text{pre}}}. \tag{7.3}$$

Dabei gibt $\delta_{\mathfrak{n}_{\text{pre}}}$ die aktuelle Distanz des Vorgängerknotens $\mathfrak{n}_{\text{pre}}$ an (0, falls dies der Startknoten $\mathfrak{n}_0$ ist). Die besten Ergebnisse werden hier mit $\lambda = 0{,}5$ erzielt. Für einen

Tabelle 7.2. Beispiel der Kantenliste für Knoten 4 (Iteration 6) aus Abbildung 7.2.

Indices der Knoten		Distanzmaße	
Nachfolger C	Vorgänger A	δ_{n_i}	$P(C\|BA)$
6	3	1,99	0,37
6	2	2,03	0,329
6	**1**	**1,69**	**0,96**
7	3	2,14	0,067
7	2	1,90	0,58
7	1	2,16	0,005

Knoten $\mathfrak{n}_i$ ergibt sich

$$P_{\mathsf{gram}} = \begin{cases} P(C) & \textbf{falls Vorgänger } \mathfrak{n}_{\mathsf{pre}} = \mathfrak{n}_0 \\ P(C|A) & \textbf{falls Vorvorgänger } \mathfrak{n}_{\mathsf{pre-pre}} = \mathfrak{n}_0\,. \\ P(C|BA) & \textbf{sonst} \end{cases} \tag{7.4}$$

Im letzten Fall bezieht P_{gram} sowohl die Kante von A nach B als auch jene von B nach C mit ein. Daher ist für Grafen, in denen dieser Fall eintritt, zur Berechnung des optimalen Pfades vom Start- zum Endknoten ein modifizierter Dijstra Algorithmus mit erweiterter Kantenliste erforderlich.

Wie Abbildung 7.2 anhand eines vereinfachten Beispiels zeigt, ist der Start- und Endknoten des Grafen jeweils leer. Alle Knoten dazwischen gehören entsprechend ihrer Reihe zum i-ten Buchstaben der ermittelten Buchstabensequenz. Die Nummer der zugehörigen Zeile entspricht der Position, welche die Buchstabenklasse des Knotens im Ranking einnimmt. Alle Kanten führen dabei von einem Knoten der Reihe i zur einem Knoten der Reihe $i + 1$ (beginnend mit Reihe 0, die nur den Startknoten enthält).

Erweiterte Kantenliste. Ein Trigramm wird nicht als einzelne Kante, sondern als Kombination von zwei Kanten zu drei vorgegebenen Knoten dargestellt ($A \rightarrow B \rightarrow C$). Betrachten wir den aktuell besuchten Knoten B, so hängen $P(C|BA)$ und δ_{n_i} des Nachfolgers C auch vom Vorgänger A ab! Daher müssen – im Gegensatz zum ursprünglichen Dijkstra-Algorithmus – bei jedem Schritt Kantenlisten entsprechend Tabelle 7.2 erstellt werden [2].

Der kürzeste Pfad lässt sich nun analog zum herkömmlichen Dijkstra-Algorithmus

[2] Für alle Knoten B der Reihe 2 oder höher, ansonsten genügen die herkömmlichen Kantenlisten.

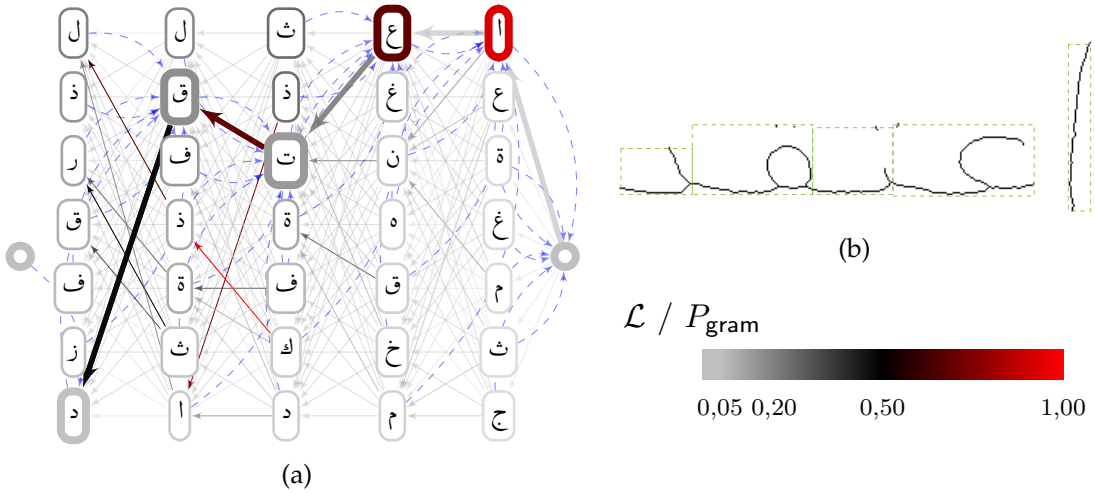


Abbildung 7.3. Reales, jedoch extremes Beispiel für die Korrektur der erkannten Sequenz mittels n-Grammen. Da hier SVMs mit sehr schwachen Merkmalen und entsprechend niedriger Treffergenauigkeit verwendet wurden, befinden sich die korrekten Buchstabenklassen der letzten drei Buchstaben relativ weit unten im Ranking.

ermitteln [43]. Bei zuverlässiger Klassifikation ist zu erwarten, dass dieser Pfad alle Knoten der ersten Reihe enthält, welche das Klassifikationsergebnis der SVMs darstellen. Bei uneindeutigen Log-Likelihoods $\mathcal{L}$ – wie in Abbildung 7.3 – enthält der Pfad gegebenenfalls auch Knoten weiterer Reihen. In diesem Fall sind die n-Gramm basierten a priori Informationen zuverlässiger als die Klassifikationsergebnisse.

7.2 Vokabular basierte Fehlerkorrektur

Bei der Erkennung von Handschrift wird oft auf ein Vokabular von Wörtern zurückgegriffen. Bei holistischen Ansätzen ist dieses Vokabular auf jene Wörter eingeschränkt, zu denen Trainingsdaten vorliegen. Allerdings wird auch bei segmentierungsbasierten Ansätzen häufig ein Vokabular von einigen hundert bis tausend Wörtern verwendet [96], [64].

Das Vokabular $\mathcal{V}_{50k}$ und dessen Untermengen sollen im Weiteren als Grundlage zur Erforschung von vokabularbasierter Fehlerkorrektur dienen. Im Gegensatz zu der oben beschriebenen, auf n-Grammen basierenden Methode, ist jedoch keine Korrektur von Wörtern möglich, die nicht Element des verwendeten Vokabulars sind. Im Fall der Erkennung allgemeiner arabischer Texte lassen sich auch mit sehr umfangreichen Vokabularen nicht alle Wörter abdecken. Dies gilt besonders für Eigennamen, seltene Fachbegriffe, Fremdwörter oder erst kürzlich eingeführte Begriffe. Wie jedoch in Kapitel 4.2 gezeigt wurde, werden bereits die meisten Wörter von historischen Texten durch die häufigsten 50.000 Wörter eines moderner

arabischen Textkorpus abgedeckt[3] .

7.2.1 Vergleich der erkannten Sequenz mit dem Vokabular

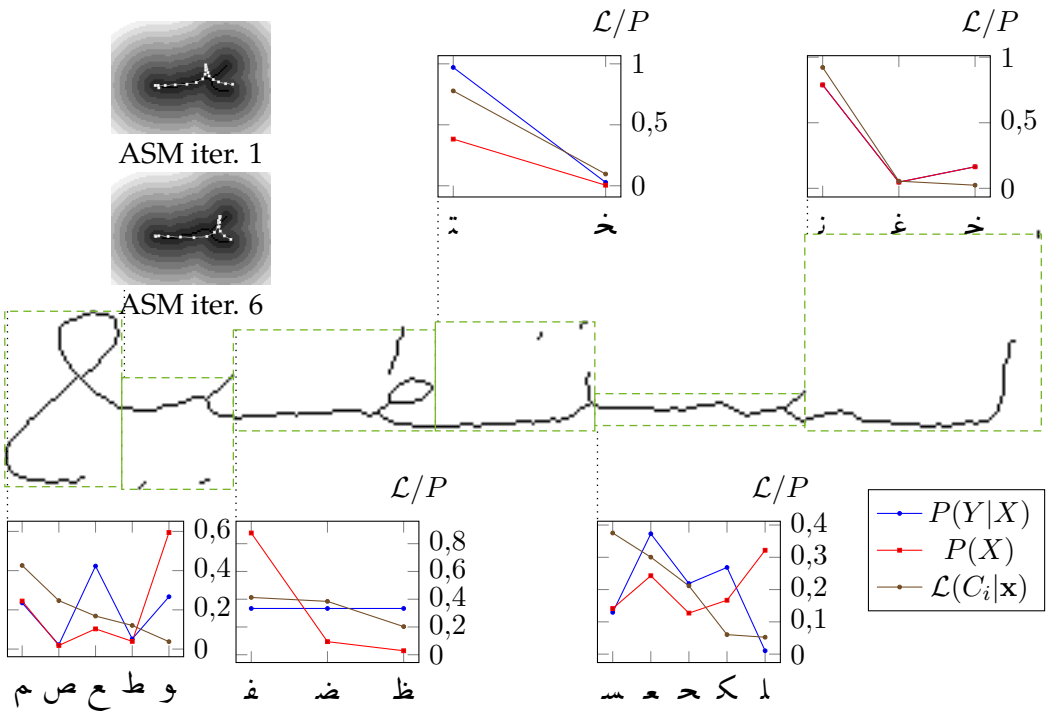
Zur Korrektur der erkannten Buchstabensequenz $\mathbf{w}_r$ erfolgt zunächst ein Vergleich von $\mathbf{w}_r$ mit allen Vokabeln $\mathbf{w}_{\mathcal{V}_i} \in \mathcal{V}$ des verwendeten Vokabulars $\mathcal{V}$. Hierzu wird die Levenstein-Distanz

$$\delta_l = \delta(\mathbf{w}_r, \mathbf{w}_{\mathcal{V}_i}), \tag{7.5}$$

verwendet (siehe Kapitel 4.3.2). Hierbei ist jedoch eine geringfügige Anpassung der Levenstein-Distanz an die Eigenheiten arabischer Schrift nötig: Da PAWs – im Gegensatz zu Wörtern – nicht explizit durch Leerzeichen separiert werden, müssen diese vor Berechnung von δ_l ermittelt und als zusätzliches Sonderzeichen eingefügt werden. Mit anderen Maßen, wie ISUB oder Soft TF-IDF, lassen sich zukünftig gegebenenfalls noch genauere Unterscheidungen treffen [25, 94]. Hierzu werden unter anderem die Auftrittswahrscheinlichkeiten der Buchstaben mit einbezogen, um die zum Überführen der Zeichenketten benötigte Operationen – im Vergleich zur Levenstein Distanz – unterschiedlich zu gewichten.

Zunächst werden alle Vokabeln $\mathbf{w}_{\mathcal{V}_i}$ nach ihrer Levenstein-Distanz $\delta(\mathbf{w}_r, \mathbf{w}_{\mathcal{V}_i})$ aufsteigend geordnet. Ein Beispiel zeigt Abbildung 7.4 mit $\mathbf{w}_r =$ نستفيم und der ähnlichsten Vokabel $\mathbf{w}_{\mathcal{V}_1} =$نستطيع. Vokabeln mit identischem $\delta(\mathbf{w}_r, \mathbf{w}_{\mathcal{V}_i})$ werden entsprechend ihrer a priori Auftrittswahrscheinlichkeit $P(\mathbf{w}_{\mathcal{V}_i})$ absteigend geordnet, sofern diese bekannt ist (im Fall von $\mathcal{V}_{50k}$ wird diese aus dem Vorkommen der Vokabeln im Textkorpus abgeleitet). Ein weiteres Beispiel einer Worterkennung, die das Ergebnis einer Segmentierung, Buchstabenerkennung und der anschließenden Korrektur mit Hilfe eines Vokabulars darstellt, zeigt Abbildung 7.5. Hier wurde keine Vorklassifizierung eingesetzt, jedoch sind die dargestellten Rankings auf die ersten sechs Klassen beschränkt. Es wurden sowohl mit der originalen IESK-arDB$_{\text{OnlineLetter}}$ trainierte SVMs verwendet, als auch SVMs, die mit Proben trainiert wurden, welche modifizierte Kashidas aufweisen (die resultierenden Log-Likelihoods $\mathcal{L}(C_i|\mathbf{x})$ wurden jeweils gemittelt). Die Konfidenz $conf'(\mathbf{w}_r) \in [0{,}1]$ dient zur Abschätzung von Fehlern in der erkannten Sequenz $\mathbf{w}_r$ und wird in Kapitel 8.3.1 näher erläutert. Für dieses Beispiel wurde eine Konfidenz von $conf'(\mathbf{w}_r) = 0{,}5$ berechnet, was auf

[3] Es ist anzumerken, dass sich die arabische Sprache und Rechtschreibung im Laufe der letzten Jahrhunderte kaum gewandelt hat. Probleme, wie sie z.B. bei der Analyse historischen Aufzeichnungen aus Deutschland mit teilweise sehr abweichender Schreibweise auftreten [39], sind hier also nur in sehr abgeschwächter Form zu erwarten.



Erkanntes Wort ($\mathbf{w}_r$): م ي ف ت س ن ⇒ نستفيم (ع ي ف ت س ن ⇒ نستفيع)$_{\text{ASM}}$

Tatsächliches Wort ($\mathbf{w}_t$): ع ي ط ت س ن ⇒ نستطيع L. Distanz δ: 2 (1)$_{\text{ASM}}$

Wort-Kandidaten ($\mathcal{V}'$): نستطيع $\delta = 2$ (korrekt), يستطيع $\delta = 3$, تفهم $\delta = 3$, مستحيل $\delta = 3$, نسيت $\delta = 3$, ستفعل $\delta = 3$, ستيف $\delta = 3$, نتكلم $\delta = 3$, مستشفي $\delta = 3$, تيم $\delta = 3$, سليم $\delta = 3$ ...

Abbildung 7.4. Ausgabe des Worterkennungssystems für ein handgeschriebenes arabisches Wort. Die Diagramme zeigen die Likelihoods der SVM basierten Rankings (nach der Vorklassifizierung) sowie die relativen Wahrscheinlichkeit für entsprechende Unigramme und Bigramme an.

Fehler hinweist. Tatsächlich wurde der letzte Buchstabens ب mit ج verwechselt.

Ist die tatsächliche Buchstabensequenz $\mathbf{w}_t$ durch die Grundwahrheiten gegeben, lässt sich ermitteln, ob auftretende Segmentierungs- oder Klassifikationsfehler durch den Abgleich mit dem Vokabular behoben werden. Dies ist genau dann der Fall, wenn

$$\mathbf{w}_r \neq \mathbf{w}_t \wedge \mathbf{w}_{\mathcal{V}_1} = \mathbf{w}_t, \tag{7.6}$$

gilt. Hier ist anzumerken, dass – entsprechend der Analyse in Kapitel 4.2 – die Erfolgsrate der Fehlerkorrektur nicht ausschließlich durch die Anzahl der Fehler und Größe des Vokabulars beeinflusst wird, sondern in erheblichem Maße von der Ähnlichkeit der verschiedenen Vokabeln zueinander abhängt.

Die Treffergenauigkeit der Worterkennung mit und ohne Abgleich mit einem Vokabular wird im nächsten Kapitel in Abschnitt 8.3.1 genauer untersucht. Sie

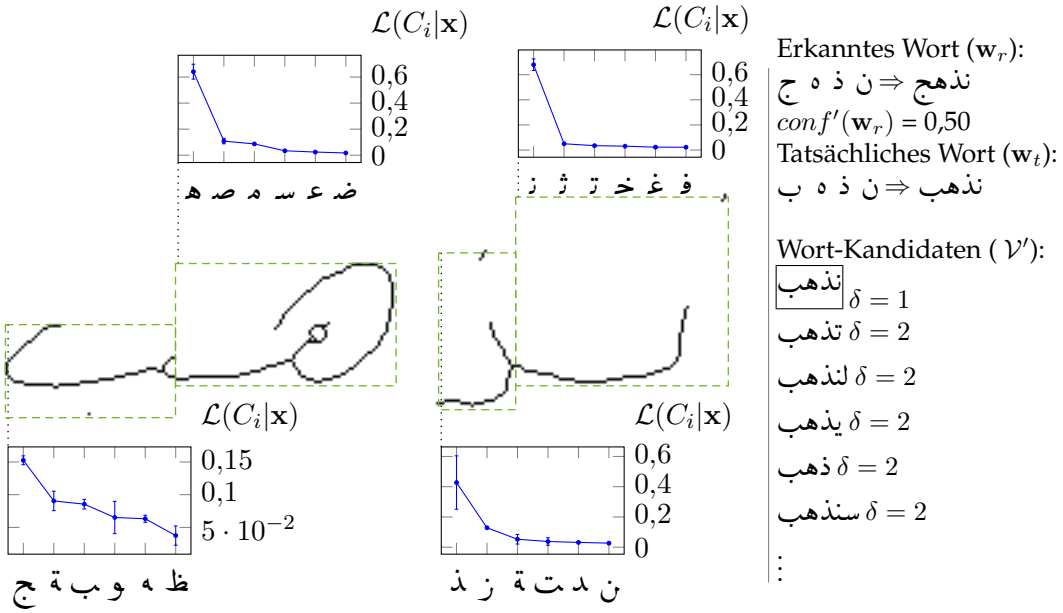


Abbildung 7.5. Beispiel für die Worterkennung mit multiplen SVMs ohne Vorklassifizierung.

stellt ein wesentliches Kriterium zur Evaluation des vorgestellten Ansatzes zur Erkennung arabischer Handschrift dar.

7.3 Sequenz-zu-Sequenz Verfahren

In diesem Abschnitt soll ein auf Sequenz-zu-Sequenz Verfahren basierender Ansatz zur Worterkennung untersucht werden. Sequenz-zu-Sequenz Verfahren, wie Long-Short-Term-Memory Netzwerke (LSTMs) oder Hidden-Markov-Modelle (HMMs), werden eingesetzt, um eine Eingabe-Sequenz in eine Ausgabe-Sequenz mit gegebenenfalls variierender Länge zu überführen. Dieser Ansatz ist vielseitig einsetzbar. Sofern ausreichend Trainingsproben vorliegen, lässt sich etwa eine Vorhersage von Aktienkursen oder der Toxizität von Molekülen treffen, oder eine Sequenz von Wörtern in eine andere Sprache übersetzen. Werden Buchstaben (oder aus ihnen extrahierte Merkmale) als Elemente der Sequenzen benutzt, lassen sich Rechtschreib- oder Erkennungsfehler beheben, die – ähnlich wie die in Abschnitt 7.1.3 beschriebenen n-Gramme – auf statistischen Wahrscheinlichkeiten beruhen. Allerdings erfolgt die Korrektur implizit und nur entsprechend der in den Trainingsdaten vorkommenden Sequenzen.

Der Vorteil von Sequenz-zu-Sequenz Verfahren gegenüber n-Grammen liegt darin, dass die Wahrscheinlichkeit von nacheinander auftretenden Klassen abhängig

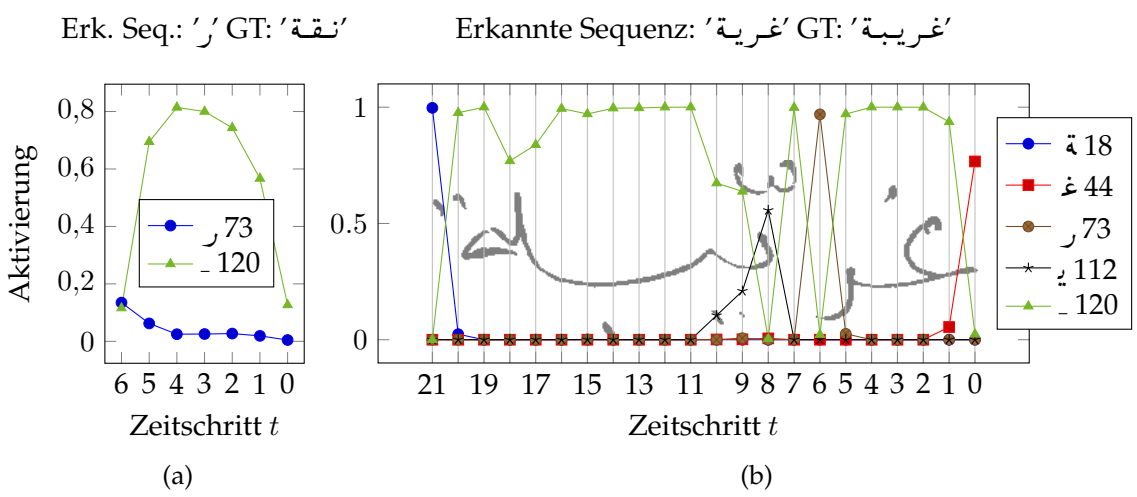


Abbildung 7.6. Darstellung der Ausgabeschicht des LSTMs (Aktivierungen der Buchstabenklassen mit Label-Nr. über die Zeit; Klassen mit geringen Aktivierungen wurden zur Wahrung der Übersicht ausgeblendet). (a) Beispiel aus Epoche 1: Zu beobachten ist zunächst eine starke Aktivierung der Rest-Klasse für Kashida-Verbindungen oder Leerzeichen, da diese in allen Proben vorhanden sind. (b) Das vorliegende Beispiel stammt aus der 17. Trainingsepoche, in welcher auch die Aktivierungen der restlichen Buchstabenklassen zunehmend zuverlässiger werden.

von den zur Handschrift gehörenden optischen Merkmalen ausgewertet wird. So lassen sich die Merkmale eines undeutlich geschriebenen Buchstaben gegebenenfalls besser auswerten, wenn dies im Kontext der vorherigen und nachfolgenden Merkmale geschieht. Dies ist insbesondere bei benachbarten Buchstaben der Fall, die ihre Schreibweise gegenseitig beeinflussen (im Arabischen beispielsweise die Buchstabenpaare aus Tabelle 3.2, im Deutschen doppelte Buchstaben wie „tt“). Ein Nachteil liegt in der Anfälligkeit zum Overfitting, weshalb zum Training sehr viele Proben mit ausreichender Varianz erforderlich sind.

7.3.1 LSTMs mit Connectionist Temporal Classification

Long-Short-Term-Memory Netzwerke (LSTMs) sind auch für Verfahren zur Erkennung arabischer Handschrift geeignet, die auf impliziter Segmentierung basierende. Durch die Connectionist Temporal Classification (CTC) Funktion als Transkriptionsschicht, erlaubt z.B. das rnnlib Framework die Klassifikation der in arabischer Handschrift vorhandenen Buchstabenklassen ohne explizite Segmentierung [41]. Dabei findet zunächst eine Übersegmentierung in Zeitschritte statt, zu denen jeweils Merkmale extrahiert werden. Wie Abbildung 7.6 zeigt, wird für alle Buchstabenklassen eine Aktivierungsfunktion berechnet, d.h., dass auch die Position der Buchstaben durch Klassifikation ermittelt wird. Jede Aktivierungsfunktion

approximiert zu allen Zeitschritten t_i die Wahrscheinlichkeit, dass die beobachteten Merkmale zur entsprechenden Buchstabenklasse gehören. Das LSTM entscheidet anschließend aufgrund aller Aktivierungsfunktionen, welche Sequenz von Buchstaben erkannt wurde. Hierbei bilden die Kashidas und die durch PAWs oder Wörter verursachten Zwischenräume eine Zusatzklasse, die zur Segmentierung der Aktivierungsfunktion in die Ausgabesequenz $\mathbf{w}_r$ eingesetzt wird.

Um einen Vergleich zu dem in dieser Dissertation vorgestellten auf expliziter Segmentierung basierenden System zu erhalten, wurde das rnnlin-Framework benutzt, um ein LSTM mit CTC Funktion auf der IFN\ENIT Datenbank zu trainieren und zu testen.

7.4 Zusammenfassung

In diesem Kapitel wurden Verfahren vorgestellt, mit denen sich die Ergebnisse der zuvor beschriebenen Segmentierungs- und Klassifikationverfahren interpretieren lassen, um letztlich handschriftliche arabische Wörter zu erkennen. Hierzu wurden statistische Informationen eingesetzt, die aus den fünfzigtausend häufigsten arabischen Wörtern extrahiert wurden.

Alternativ zu dem vorgeschlagenem System wurde, entsprechend der aktuellen Entwicklung im Bereich Mustererkennung, ein Long-Short-Term-Memory Netzwerk trainiert, welches das Segmentierungsproblem implizit löst.

Im anschließendem Kapitel erfolgt eine experimentelle Auswertung aller an der Worterkennung beteiligter Komponenten.

KAPITEL 8

Experimentelle Auswertung

DIESES Kapitel stellt die experimentellen Ergebnisse der in dieser Dissertation beschriebenen Ansätze zur Synthese und Erkennung von handschriftlichen arabischen Wörtern vor.

In Kapitel 5.3 wurden bereits verschiedene Segmentierungsmethoden miteinander verglichen. Die Methode, die dabei die besten Ergebnisse lieferte, wird nun in Abschnitt 8.1 genauer evaluiert, wozu verschiedenen Datenbanken verwendet werden. Eine Evaluation der untersuchten Methoden zur Buchstabenklassifikation erfolgt anschließend in Abschnitt 8.2. Hiernach wird in Abschnitt 8.3 die Effektivität der vorgeschlagenen Verfahren zur Nachverarbeitung und Worterkennung ausgewertet. In Abschnitt 8.3.1 wird zunächst, basierend auf statistischem Vorwissen und den Klassifizierungsergebnissen, die Konfidenz der Worterkennung bewertet. In den folgenden Abschnitten erfolgt eine detaillierte Evaluation der Worterkennung in Abhängigkeit der verschiedenen zuvor beschriebenen Methoden und Eigenschaften der Datenbank. Eine Übersicht hierzu ist in Abbildung 8.1 dargestellt.

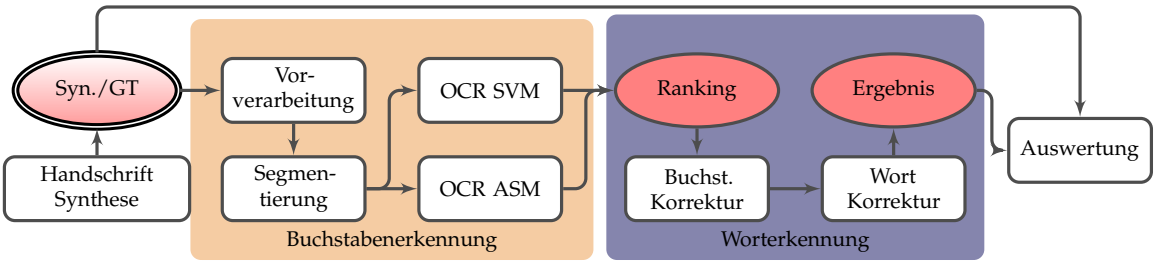


Abbildung 8.1. Übersicht zur Evaluierung des Worterkennungssystems.

8.1 Evaluierung der Segmentierung in Buchstaben mittels realer und synthetischer Proben

Eine Vergleich verschiedener Methoden zur Segmentierung von arabischen Wörtern in Buchstaben anhand der IESK-arDB fand bereits in Abschnitt 5.3 statt. Die Datensynthese ist ein wesentlicher Beitrag dieser Dissertation und bildet die Grundlage vieler der folgenden Experimente. Daher findet hier eine Evaluation der favorisierten Segmentierungsmethode an realen und synthetischen Datenbanken statt. Die einzige verfügbare Datenbank arabischer Handschrift mit Grundwahrheiten (GT), welche eine automatische Validierung und Testung der Segmentierungsverfahren zulassen, ist hierbei die am Institut erstellte Wortdatenbank IESK-arDB. Tabelle 8.1 zeigt die verschiedenen Fehlermaße für die IESK-arDB, die mit demselben Vokabular synthetisierte IESK-arDB$_{\mathsf{Syn}}$, sowie für synthetische Proben mit dem Vokabular $\mathcal{V}_{\mathsf{5k}}$.

Die Ergebnisse für die IESK-arDB und die IESK-arDB$_{\mathsf{Syn}}$ unterscheiden sich hauptsächlich aufgrund der Anzahl perfekt segmentierter Wörter. Dies mag daran liegen, dass sich die Schreibweisen der synthetisierten Wörter weniger drastisch unterscheiden[1]. Hierdurch treten bei der IESK-arDB$_{\mathsf{Syn}}$ weniger Wörter mit sehr vielen bzw. wenigen Segmentierungsfehlern auf. Dafür spricht auch die geringere Varianz der perfekt segmentierten Wörter der IESK-arDB bei gleichzeitig deutlich höherer Varianz an Segmentierungsfehlern pro Wort. Die mit $\mathcal{V}_{\mathsf{5k}}$ synthetisierte Datenbank führt aufgrund der im Durchschnitt deutlich kürzeren Wörter (siehe

Datenbank	IESK-arDB	IESK-arDB$_{\mathsf{Syn}}$	IESK-arDB$_{\mathsf{Syn}}$ $\mathcal{V}_{\mathsf{5k}}$
Anzahl der Wortproben	2540	9000	8000
Fehler pro Wort	1,67 $\pm$ 0,13	1,74 $\pm$ 0,024	0,96 $\pm$ 0,019
Fehler pro Buchstabe	0,35 $\pm$ 0,03	0,34 $\pm$ $5e^{-3}$	0,27 $\pm$ $6{,}16e^{-3}$
Übersegmentierung	80 $\pm$ 8%	86 $\pm$ 0,6%	41 $\pm$ 0,7%
Untersegmentierung	90% $\pm$ 7%	88 $\pm$ 1,9	55 $\pm$ 1,3%
Perfekte segmentierte Wörter	17 $\pm$ 0,2%	13 $\pm$ 0,7%	35 $\pm$ 0,8%

Tabelle 8.1. Auswertung der verwendeten Segmentierungsmethode an realen und synthetischen Datenbanken mittels der in Abschnitt 5.3 definierten Fehlermaße.

[1]Dem ließe sich entgegenwirken, indem Datenbanken mit mehreren Einstellungsdateien generiert werden, um sauber geschriebene mit undeutlichen Proben zu vermischen. Zudem spielt die Anzahl der Schreiber der IESK-arDB$_{\mathsf{OnlineLetter}}$ eine Rolle.

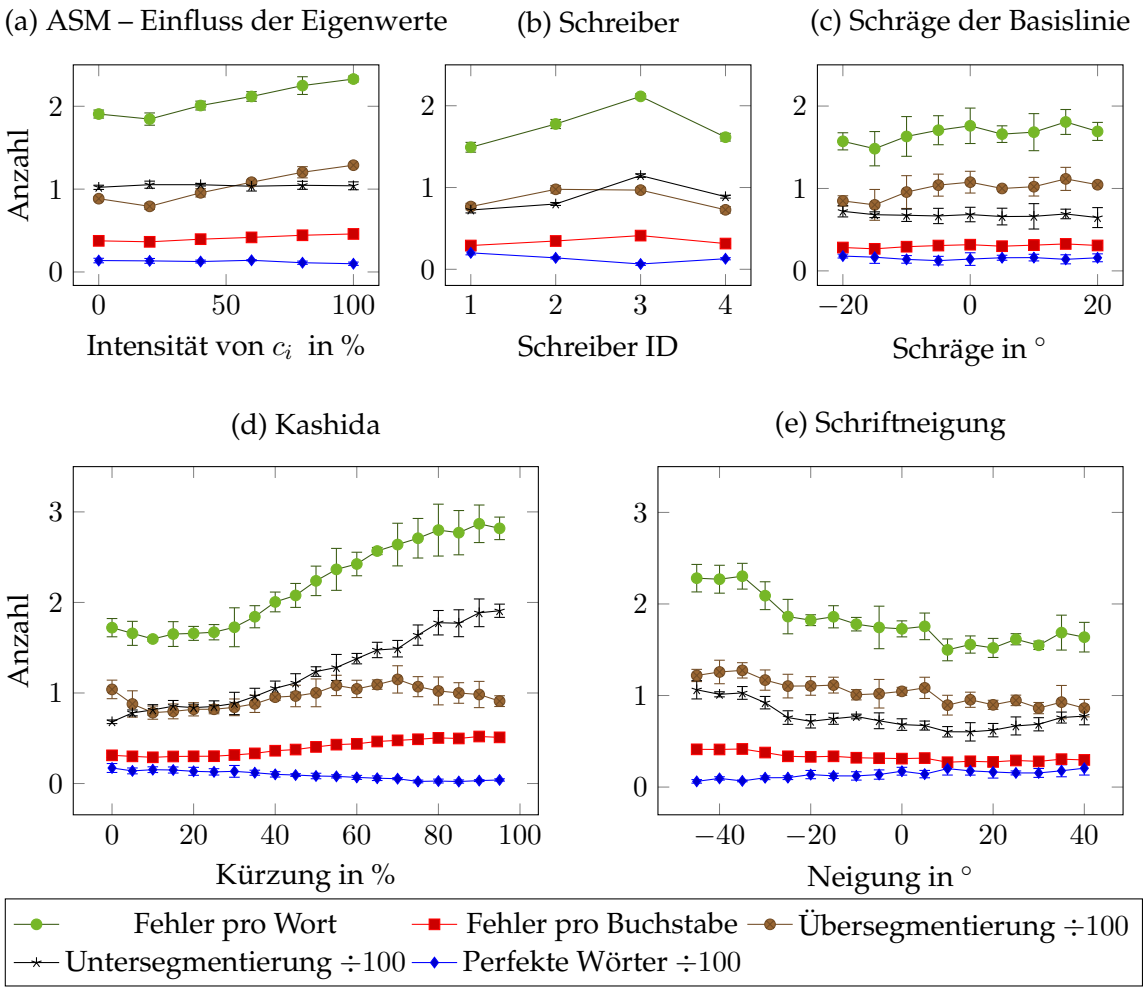


Abbildung 8.2. Untersuchung des Einflusses verschiedener Schrifteigenschaften (siehe Tabelle 3.1 aus Kapitel 3) auf die Güte der verwendeten Segmentierungsmethode.

Abschnitt 4.2) zu weniger Fehlern pro Wort.

8.1.1 Einfluss der handschriftlichen Eigenschaften auf die Segmentierungsergebnisse

In diesem Abschnitt wird die Robustheit der favorisierten Segmentierungsmethode gegen unterschiedliche Einflüsse untersucht. Hierzu wurden im Vorfeld mit Hilfe des Synthesesystems aus Kapitel 3 mehrere Datenbanken mit dem Vokabular der IESK-arDB generiert, deren Abweichungen bis auf jeweils eine Eigenschaft nur minimal ausgeprägt sind. Die Datenbanken wurden hierzu in mehrere Untereinheiten aufgeteilt, die jeweils eine bestimmte Ausprägung der untersuchten Eigenschaft enthalten (eine Untereinheit pro x-Koordinate der Diagramme in Abbildung 8.2).

(a) ASM – Einfluss der Eigenwerte. In Abbildung 8.2 (a) wird die Güte der Segmentierung in Abhängigkeit der Abweichung dargestellt, den die durch Active

Shape Modells (ASMs) generierte Probe $\mathbf{u}$ zum Erwartungswert $\bar{\mathbf{x}}$ aufweist. Hier entsprechen 100% einem maximalen Faktor von $\sqrt{2\lambda_j}$ und $c_j = \pm 1$ (siehe Gleichung 3.3), was in einigen Fällen bereits sehr undeutlich geschriebene Buchstaben zur Folge hat. Da diese Abweichungen aber weniger die potentiellen Segmentierungspunkte, sondern vielmehr die Gestalt der Buchstaben betreffen, ergeben sich dennoch vergleichsweise geringe negative Auswirkung auf die Güte der Segmentierung.

(b) Schreiber. Stärker ist der Einfluss des persönlichen Schreibstils auf die Güte der Segmentierung, wie Abbildung 8.2 (b) verdeutlicht. Obwohl stets der Naskh-Stil verwendet wird, und die Glyphen den Erwartungswert $\bar{\mathbf{x}}$ der ASMs aufweisen, treten hier deutliche Abweichungen auf, welche auch die Topologie – also die zur Segmentierung verwendeten Key Features (KF) – betreffen.

(c) Schräge der Basislinie. Aus 8.2 (c) geht hervor, dass Abweichungen der Basislinie von bis zu $\pm 20^\circ$ – wie sie auch der ursprünglichen IESK-arDB auftreten – keinen eindeutigen Einfluss auf die Segmentierung haben. Daher wird auch die optionale Korrektur der Basislinie in den folgenden Experimenten nicht eingesetzt.

(d) Kashida. Wie Abbildung 8.2 (d) zu entnehmen ist, hat die Länge der Kashida zwischen zwei Buchstaben von allen untersuchten Eigenschaften den deutlichsten Einfluss auf die Güte der Segmentierung. Obwohl das Segmentierungsverfahren bevorzugt die Verzweigungspunkte (und nicht die Minima) der Kashidas als potentielle Trennstellen nutzt, führen verkürzte Kashidas zu schwerwiegenden Problemen. So resultieren mehr überlappende Oberlängen benachbarter Buchstaben, oder – typisch für arabische Schrift – auch übereinander geschriebene Buchstaben.

(e) Schriftneigung. Die Schriftneigung wurde im Bereich von $\pm 40^\circ$ untersucht. Wie Abbildung 8.2 (e) zeigt, ist eine Neigung von $\leq -20^\circ$ problematisch, wogegen eine Neigung von $\geq 0^\circ$ sich positiv auf die Segmentierungsergebnisse auswirkt. Ursache hierfür mag sein, dass trotz der sorgfältig und möglichst neutral geschriebenen Buchstaben der Datenbank ein negativer Bias der Schriftneigung vorherrscht. Die künstlich erzeugte Neigung gleicht diesen – in erster Linie Oberlängen betreffenden – Bias aus. Zudem besteht bei negativem Neigungsgrad (Neigung nach rechts) die Gefahr, dass Trennstellen von Oberlängen vertikal überlappt werden. Da das verwendete Segmentierungsverfahren nicht in der Mitte zwischen zwei Buchstaben, sondern links davon trennt, führt ein negativer Neigungsgrad hierdurch zu vermehrten Untersegmentierungen.

Tabelle 8.2. Maße zur Bestimmung der Güte der Buchstabenklassifikation.

Maß	Berechnung	Bedeutung
Relevanz (Precision $\mathcal{P}$)	$\frac{TP}{TP+FP}$	Es wurde Klasse C_i erkannt: Anteil an Fällen, in denen die Klasse tatsächlich C_i ist
Sensitivität (Recall $\mathcal{R}$)	$\frac{TP}{TP+FN}$	Es ist Klasse C_i: Anteil an Fällen, in denen auch Klasse C_i erkannt wurde
F_1-Score	$\frac{2 \cdot \mathcal{P} \cdot \mathcal{R}}{\mathcal{P}+\mathcal{R}}$	Harmonisches Mittel
Treffergenauigkeit (A_b)	$\frac{100 \times (TP+TN)}{TP+TN+FP+FN}$	Anteil aller korrekt klassifizierten Proben in Prozent (für alle Buchstabenklassen)

Zusammenfassung. Obwohl das System zur Handschriftsynthese reale Datenbanken nicht vollständig ersetzen kann, weisen die Experimente in Tabelle 8.1 darauf hin, dass eine ausreichende Ähnlichkeit von synthetisierten und realen Wörtern besteht, um verschiedene Systeme damit validieren, testen, und vergleichen zu können. Um die im Vergleich zu nicht synthetischen Handschriftproben geringeren Abweichungen der einzelnen Proben auszugleichen, lässt sich eine erhöhte Probenanzahl verwenden.

8.2 Buchstabenklassifizierung

Als Maße für die Güte der Buchstabenklassifizierung haben sich neben der Treffergenauigkeit die Relevanz und Sensitivität sowie das F_1-Score Maß durchgesetzt. Die Maße lassen sich entsprechend Tabelle 8.2 mit Hilfe der True Positive (TP), True Negative (TN), False Positive (FP) und False Negative (FN) Raten berechnen. Für jede Buchstabenprobe des Testsets wird hierbei die Klasse mit dem höchsten Wert im Ranking als „Positive“ gesetzt. Alle weiteren erhalten das Label „Negative“. Stimmen diese Label mit den Grundwahrheiten (GT) überein, so wird ihnen „True“, andernfalls „False“ vorangestellt.

Im Folgenden werden mit den beschriebenen Fehlermaßen die verwendeten

Merkmale evaluiert. Darüber hinaus werden weitere Aspekte der Buchstabenerkennung untersucht. Hierzu zählt, wie die Anzahl der zum Training verwendeten Proben die Klassifikation beeinflusst. Auch die statistische Position der tatsächlichen Klasse eines Buchstabens im Ranking ist von Interesse.

8.2.1 Optimierung der Support Vector Machines

Für alle vier Positionen werden vor dem eigentlichen Training die Parameter C und γ der RBF Kernels der verwendeten Support Vector Machines (SVMs) optimiert (bei Vorklassifizierung zudem für alle Gruppen). Zunächst werden die zum Training verwendeten Proben in ein temporäres Trainings- und Validierungsset aufgeteilt, die keine Proben desselben Schreibers enthalten, um Overfitting zu reduzieren. Nach der Optimierung der Parameter erfolgt ein erneutes Training des jeweiligen SVM mit sämtlichen Trainingsproben. Diese werden anschließend an den übrigen Proben getestet. Dies erfolgt separat für alle Merkmalsgruppen oder Kombinationen derselben, welche im nächsten Abschnitt ausgewertet werden.

Wie in Abbildung 8.3 dargestellt ist, werden die beiden Parameter mittels Kreuzvalidierung und Gridsearch optimiert. Ohne Reduzierung der Klassen durch Vorklassifizierung lässt sich bei der Merkmalsgruppe „*KF-simple* “ die durchschnittliche Treffergenauigkeit durch Optimierung nur geringfügig erhöhen. Jedoch fällt auf, dass die umfangreiche, auf separat geschriebenen Glyphen basierende Buchstabendatenbank IESK-arDB$_{\text{OnlineLetter}}$, deutlich bessere Ergebnisse liefert, als die weniger umfangreiche IESK-arDB$_{\text{Letter}}$, welche als Nebenprodukt der Segmentierung entstanden ist. Dies liegt an der weniger sorgfältigen Schreibweise, welche die IESK-arDB$_{\text{Letter}}$ aufweist. Die unregelmäßige Anzahl an Proben je Klasse hingegen

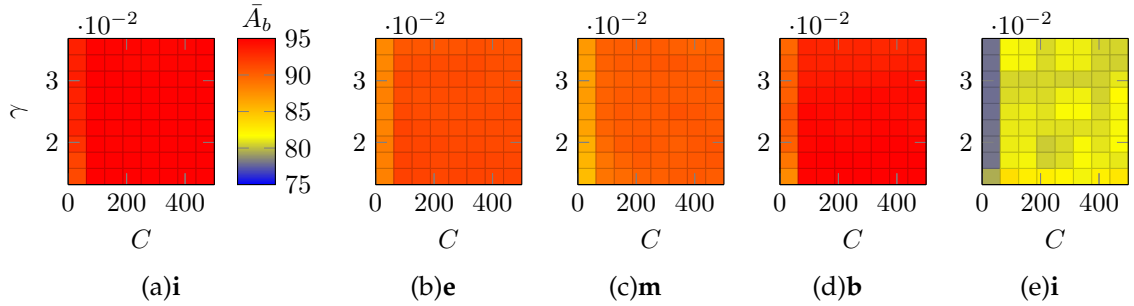


Abbildung 8.3. Optimierung der Parameter C und γ des RBF-Kernels der SVMs am Validierungsset.(a-d) Für die verschiedenen Positionen unter Nutzung der Merkmalsgruppe „*KF-simple* “ und der IESK-arDB$_{\text{OnlineLetter}}$. (e) Ergebnisse für die isolierte Position bei Nutzung der IESK-arDB$_{\text{Letter}}$.

Tabelle 8.3. Statistische Gütekriterien der Klassifikation einzelner Buchstabenklassen $\times 100$.

		Positionsspezifisch				
		i	e	m	b	∅
KF-simple:	Relevanz	94,82±3,08	91,23±3,94	90,91±2,22	94,53±2,29	92,95±1,23
	Sensitivität	95,28±1,14	90,68±5,06	90,74±4,59	94,58±3,37	92,94±0,97
	F_1-score	95,02±1,32	90,88±3,67	90,76±2,57	94,52±1,89	92,94±0,63
Momente:	Relevanz	78,31±5,64	73,69±6,57	65,56±3,66	72,41±9,39	72,98±2,73
	Sensitivität	78,26±5,15	73,50±4,20	65,83±4,01	72,48±11,31	72,94±2,80
	F_1-score	78,08±3,23	73,43±3,91	65,58±2,45	72,37±10,09	72,94±2,40
F74 [76]:	Relevanz	82,90±3,54	72,85±2,94	75,47±4,41	80,54±3,43	78,08±2,16
	Sensitivität	83,13±4,64	73,10±10,68	75,69±3,84	80,73±5,10	78,00±2,11
	F_1-score	83,00±4,02	72,65±5,64	75,55±3,76	80,58±3,63	78,04±1,99
Gabor:	Relevanz	88,98±3,62	80,71±3,70	81,58±6,06	86,49±2,93	84,60±1,22
	Sensitivität	89,34±3,89	80,22±5,56	81,68±4,38	86,83±5,23	84,43±2,41
	F_1-score	89,10±2,77	80,32±2,74	81,51±4,00	86,52±1,96	84,50±1,32

verbessert die Treffergenauigkeit sogar, da Klassen wie ا, mit vielen Trainings- und somit auch Validierungsproben, bei der Klassifikation bevorzugt werden. Problematisch ist dahingehend jedoch die Worterkennung: Zwar treten seltenere Klassen hier ebenfalls entsprechend selten auf, sie haben hierdurch allerdings auch einen höheren Informationsgehalt. Dieser wirkt sich positiv auf die Verfahren zur Fehlerkorrektur aus, sofern diese Buchstaben korrekt klassifiziert wurden.

8.2.2 Evaluierung der Support Vector Machines

Um die verschiedenen Gruppen von Merkmalen aus Kapitel 6 auszuwerten und miteinander zu vergleichen, wurden die – entsprechend des letzten Abschnitts optimierten – SVMs verwendet.

Bezüglich der getesteten Merkmale der Literatur erzielen Gabor-Filter, wie aus Tabelle 8.3 hervorgeht, die besten Ergebnisse. Jedoch sind diese in ihrer Berechnung mit Abstand am komplexesten, weshalb die für die Schrifterkennung optimierten, in [76] beschriebenen F74-Merkmale zu bevorzugen sind. Die Merkmalsgruppe der zentralen und spatialen Momenten liefert zwar bessere Ergebnisse, als die hier nicht aufgeführten, rotationsinvarianten Momente von Hu, sie sind aber dennoch schlechter als die anderen Merkmalsgruppen zur Klassifikation arabischer Buchstaben geeignet. Die Gruppe *KF-simple* setzt sich aus manuell konzipierten Merkmalen zusammen. Sie wurde ursprünglich nur zur Ergänzung der anderen

Merkmalsgruppen konzipiert, um im Zusammenhang mit diakritischen Zeichen stehende Schwächen bei der Klassifizierung auszugleichen. In ihrer finalen Form erzielt diese Merkmalsgruppe allerdings die beste Relevanz, Sensitivität und den besten F_1-Score.

Einfluss der Buchstabenposition. In der **m**ittleren und **b**eginnenden Position ist die Anzahl der Klassen um jene sechs Buchstaben reduziert, die ausschließlich in **i**nitialer und **e**ndender Form existieren (ا ا د د ذ ذ ر ر ز ز و و). Einerseits sinkt durch diese Verminderung der Klassen prinzipiell die Wahrscheinlichkeit, dass zwei Buchstaben miteinander verwechselt werden (veranschaulicht wird dies in den Konfusionmatrizen in Abbildung A.3 des Anhangs). Daher ist das F_1-score für Buchstaben in endender auch höher als für jene in beginnender Position (siehe Tabelle 8.3). Andererseits ist die Gestalt der Buchstaben von der Position abhängig. Diese ist in der isolierten Position am ausgeprägtesten. Die isolierte Position weist zudem keine Kashidas auf, die – da sie in allen Klassen vorkommen – aufgrund niedriger Intervarianz kaum zur Klassifikation geeigneten Merkmale aufweisen. Dies erklärt die guten Ergebnisse für Buchstaben in der isolierten im Vergleich zur mittleren Position, deren Gestalt stark reduziert ist.

Abhängigkeit der Worterkennung von der Buchstabenerkennung. Ein sinnvolles Maß zur Angabe der Güte, mit welcher die Klassifikation arabischer Buchstaben erfolgt, ist die Treffergenauigkeit A_b. Zusammen mit $\overline{|\mathbf{w_r}|}$, der durchschnittlichen Anzahl an Zeichen der erkannten Buchstabensequenz, lässt sich auch die zu erwartende Treffergenauigkeit der Worterkennung A'_w schätzen:

$$A'_w \approx 100 \left(\frac{A_b}{100} \right)^{\overline{|\mathbf{w_r}|}} . \tag{8.1}$$

Dies gilt unter der Annahme, dass keine Segmentierungsfehler auftreten und keine Fehlerkorrektur stattfindet. Abbildung 8.4 auf Seite 169 zeigt die geschätzte, maximale Treffergenauigkeit A'_w der Worterkennung bei Verwendung der *KF-simple* Merkmale und des Vokabulars $\mathcal{V}_{\mathsf{50k}}$ (rote Markierung). Zudem wird angezeigt, wie hoch die Treffergenauigkeit A_b der Buchstabenerkennung mindestens sein muss, um für verschiedene Vokabulare ein Worterkennung mit $A'_w = 40\%$ zu erzielen (grüne Markierungen).

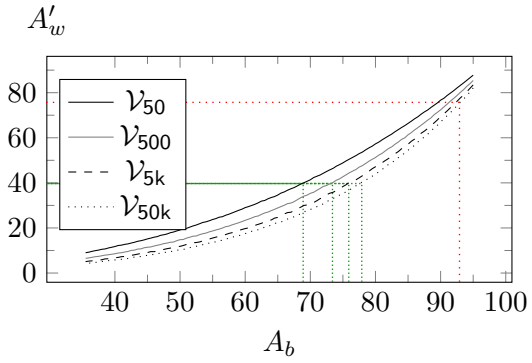


Abbildung 8.4. Aufgrund der Treffergenauigkeit der Buchstabenerkennung A_b geschätzte maximale (d.h. bei perfekter Segmentierung erwartete) Treffergenauigkeit der Worterkennung A'_w in Abhängigkeit des verwendeten Vokabulars.

Treffergenauigkeit und Analyse der Rankings für verschiedene Merkmale. Die Treffergenauigkeiten für verschiedene Merkmalsgruppen, sowie einiger kombinierter Merkmalsvektoren, ist in Tabelle 8.4 dargestellt. Zeile 11 zeigt, dass durch die Reduzierung der Klassen mittels Vorklassifizierung und anschließender ASM basierten Klassifikation eine, im Vergleich zum der Klassifikation mittels SVMs ohne Vorklassifizierung, geringe Treffergenauigkeit A_b erzielt wird. Die Merkmalsgruppe mit der höchste Treffergenauigkeit ist *KF-simpel* (Zeile 5). Auch durch Kombination mit den Gradientenmerkmalen *F74* lässt sich keine eindeutige Verbesserung erzielen (Zeile 7). Werden sämtliche Merkmale verwendet, tritt sogar eine Verschlechterung ein. Dies ist auch der Fall, wenn die Anzahl der Merkmale durch Principal Component Analysis (PCA) auf 30% bzw. 50% reduziert wird. Ursache hierfür mag sein, dass die aussagekräftige Merkmalsgruppe *KF-simple* nur aus sehr wenigen Merkmalen besteht.

Die Merkmalsgruppe *CHi* wird, im Gegensatz zu den anderen Merkmalen, nicht aus einem normierten Bild, sondern aus den Trajektorien der Buchstaben berechnet (online Ansatz). Wie aus Zeile 14 hervorgeht, erlaubt der hierdurch erhöhte Informationsgehalt eine im Vergleich zu den offline Merkmalen aus Zeile 1-5 bessere Erkennung. Da die Rekonstruktion der Trajektorien aus offline Proben jedoch insbesondere für die isolierte Form problematisch ist, und es zudem zu Abweichungen zwischen den zum Training verwendeten Buchstabendatenbanken und segmentierten Buchstaben kommt, spielt die Merkmalsgruppe *CHi* für die folgende Untersuchung der Worterkennung keine entscheidende Rolle.

Werden die generativen Gaussian Mixture Models (GMMs) mit den selben Merkmalen wie die SVMs trainiert, führt dies wie erwartet zu erheblich schlechteren

Tabelle 8.4. Treffergenauigkeit A_b und andere statistische Gütekriterien der Multiklassen-SVMs, ASMs und Gaussian Mixture Model (GMM) basierten Buchstabenerkennung.

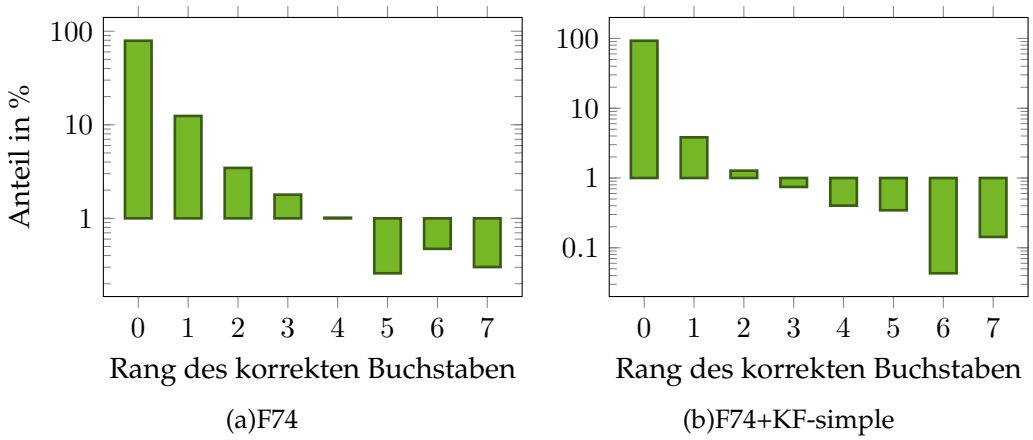
	Merkmalsgruppe	A_b in %	∅ Rang[1]	$\mathcal{L}_{real}$ [2]	$\mathcal{L}_0$ [3]	$\mathcal{L}_1$
1	Momente	$72{,}45 \pm 4{,}34$	$0{,}93 \pm 2{,}96$	$0{,}44 \pm 0{,}27$	$0{,}50 \pm 0{,}23$	$0{,}11 \pm 0{,}085$
2	*Im15* [86]	$76{,}89 \pm 3{,}02$	$0{,}43 \pm 1{,}53$	$0{,}50 \pm 0{,}26$	$0{,}57 \pm 0{,}22$	$0{,}11 \pm 0{,}08$
3	*F74* [76]	$78{,}27 \pm 3{,}10$	$0{,}52 \pm 1{,}81$	$0{,}51 \pm 0{,}26$	$0{,}57 \pm 0{,}21$	$0{,}11 \pm 0{,}09$
4	Gabor	$84{,}60 \pm 1{,}91$	$0{,}41 \pm 1{,}80$	$0{,}53 \pm 0{,}25$	$0{,}56 \pm 0{,}22$	$0{,}09 \pm 0{,}07$
5	*KF-simple*	$\mathbf{92{,}84 \pm 2{,}54}$	$0{,}22 \pm 1{,}26$	$0{,}68 \pm 0{,}21$	$0{,}69 \pm 0{,}18$	$0{,}07 \pm 0{,}07$
6	*F74*+Moments	$74{,}96 \pm 3{,}61$	$0{,}83 \pm 2{,}79$	$0{,}46 \pm 0{,}26$	$0{,}52 \pm 0{,}22$	$0{,}11 \pm 0{,}08$
7	*F74+ KF-simple*	$93{,}80 \pm 2{,}60$	$0{,}24 \pm 1{,}71$	$0{,}63 \pm 0{,}22$	$0{,}64 \pm 0{,}20$	$0{,}07 \pm 0{,}06$
8	Alle Merkmale	$77{,}56 \pm 2{,}21$	$0{,}75 \pm 3{,}08$	$0{,}44 \pm 0{,}24$	$0{,}49 \pm 0{,}21$	$0{,}09 \pm 0{,}06$
9	*F74+KF-simple* [4]	$96{,}6 \pm 2{,}31$	$0{,}05 \pm 0{,}28$	$0{,}81 \pm 0{,}19$	$0{,}82 \pm 0{,}17$	$0{,}09 \pm 0{,}10$
10	*F74+KF-simple* [5]	$85{,}4 \pm 3{,}4$	–	–	–	–
11	ASM[5]	$77{,}3 \pm 1{,}7$	–	–	–	–
12	GMM & *F74*	$52{,}9 \pm 7{,}6$	–	–	–	–
13	GMM & *KF-simple*	$76{,}1 \pm 3{,}9$	–	–	–	–
14	*CHi*	$96{,}55 \pm 0{,}8$	$0{,}05 \pm 0{,}30$	$0{,}69 \pm 0{,}10$	$0{,}69 \pm 0{,}10$	$0{,}05 \pm 0{,}06$
15	*KF-simple*[6]	$86{,}49 \pm 1{,}52$	$0{,}23 \pm 0{,}77$	$0{,}37 \pm 0{,}15$	$0{,}38 \pm 0{,}13$	$0{,}12 \pm 0{,}05$
16	Gabor[6]	$73{,}77 \pm 1{,}91$	$0{,}79 \pm 2{,}39$	$0{,}34 \pm 0{,}19$	$0{,}37 \pm 0{,}17$	$0{,}11 \pm 0{,}05$
17	CNN[7]	78,00				

[1] Durschnittlicher Rang der tatsächlichen Buchstabenklasse
[2] Durchschnittliche Log-Likelihood der tatsächlichen Buchstabenklasse
[3] Durchschnittliche höchste Log-Likelihhood (der Klasse mit Rang 0)
[4] Bei Nutzung der GT zur perfekten Vorklassifizierung
[5] Bei Nutzung des Entscheidungsbaums zur Vorklassifizierung
[6] Trainiert und getestet an der IESK-arDB$_{\text{Letter}}$ (statt IESK-arDB$_{\text{OnlineLetter}}$)
[7] Nur isolierte Position, siehe Abschnitt 8.2.4

Ergebnissen (Zeile 12&13 im Vergleich zu Zeile 3&4). Allerdings lassen sich die GMMs einsetzen, um gegebenenfalls zusätzliche Merkmalsvektoren zum Trainieren der SVMs zu generieren.

Wie Zeile 10 verdeutlicht, führt der Einsatz von Vorklassifizierung in Kombination mit den *KF-simple* Merkmalen zu einer verschlechterten Treffergenauigkeit. Dies liegt daran, dass diese Merkmalsgruppe gezielt als Alternative zur Vorklassifizierung konzeptioniert wurde. Eine entsprechende Differenzierung zwischen Buchstaben verschiedener Gruppen ist jedoch nach der Vorklassifizierung nicht mehr möglich. Dies hat zur Folge, dass sich einmal durch Vorklassifizierung verursachte Fehler nicht mehr durch die anschließende Klassifikation beheben lassen.

Zeile 15 & 16 zeigen die Ergebnisse für den Fall, dass die IESK-arDB$_{\text{Letter}}$ statt



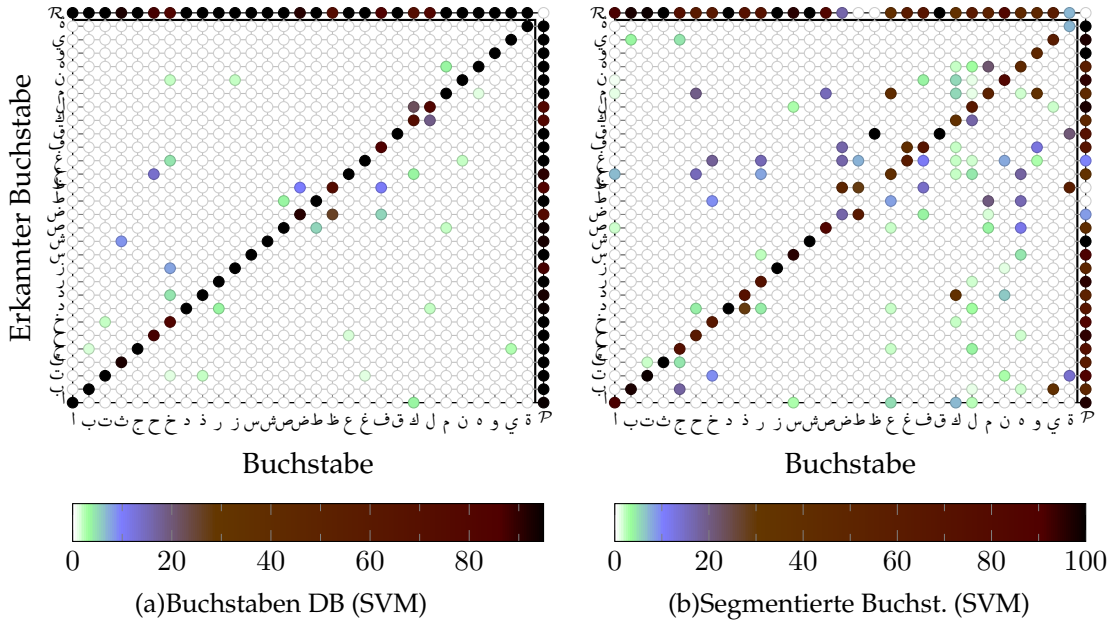
(a)F74 (b)F74+KF-simple

Abbildung 8.5. Histogramme zu den ersten sieben Rängen. Es wird jeweils angezeigt, wie häufig die korrekte Buchstabenklasse die entsprechende Position im Ranking der SVMs einnimmt. Insgesamt gibt es 29 mögliche Ränge (0-28).

der IESK-arDB$_{\text{OnlineLetter}}$ zum Training und zum Testen eingesetzt wird. Die Treffergenauigkeit ist hier im Vergleich zur IESK-arDB$_{\text{OnlineLetter}}$ geringer, was aufgrund der eher unsauber geschrieben Proben zu erwarten war (siehe Kapitel A.3). Zudem enthält die IESK-arDB$_{\text{Letter}}$ generell weniger Proben.

Weitere Gütekriterien zur Buchstabenerkennung. Da die Treffergenauigkeit A_b keine differenzierte Aussage über die Güte der Rankings und der von den SVMs berechneten Likelihoods $\mathcal{L}$ ermöglicht, sollen an dieser Stelle weitere Maße vorgestellt werden. Die dritte Spalte aus Tabelle 8.4 gibt Auskunft über den durchschnittliche Rang, den die tatsächliche Klasse der jeweiligen Probe im Ranking innehat. Hiermit lässt sich der voraussichtlich maximale Rang der korrekten Klasse ermitteln, was eine Reduzierung der Anzahl potentieller Klassen für die Nachverarbeitung ermöglicht. Bei einer korrekten Klassifikation hat die Klasse des tatsächlichen Buchstabens Rang 0, theoretisch beträgt der maximal Rang 28 (siehe S. 146 Abbildung 6.11(a)). Allerdings liegt der Rang bei den meisten Fehlklassifikationen zwischen 1 und 3 (bei Verwendung der *KF-simple* Merkmale). Abbildung 8.5 zeigt die entsprechende Verteilung der ersten sieben Ränge für zwei verschiedene Merkmalsvektoren. Hieraus geht hervor, dass die tatsächliche Klasse nur sehr selten Rang 5 oder höher innehat (es genügt demnach, die ersten 5 Ränge d.h. Rang 0 bis Rang 4 zu betrachten).

Von Interesse ist weiterhin die durchschnittliche Log-Likelihood $\mathcal{L}_{real}$ der tatsächlichen Buchstabenklasse, sowie die Log-Likelihood jener Klasse, die Rang 0 innehat. Da diese Klassen bei korrekter Klassifikation – also in den meisten Fällen



(a)Buchstaben DB (SVM) (b)Segmentierte Buchst. (SVM)

Abbildung 8.6. Konfusionsmatrizen der SVM basierten Klassifikation (schreiberunabhängig) mit Relevanz (Precision $\mathcal{P}$) und Sensitivität (Recall $\mathcal{R}$). (a) Trainiert und getestet an der IESK-arDB$_{\text{OnlineLetter}}$. (b) Trainiert an der IESK-arDB$_{\text{OnlineLetter}}$, getestet an Buchstaben, die mittels Grundwahrheiten aus Wörtern der IESK-arDB$_{\text{SynWords}}$ segmentiert wurden.

– übereinstimmen, unterscheiden sich $\mathcal{L}_{real}$ und $\mathcal{L}_0$ nur geringfügig voneinander. Dagegen weist bereits die Klassen mit Rang 1 im Durchschnitt eine deutlich geringere Log-Likelihood $\mathcal{L}_1$ auf. Ist dies nicht der Fall, so ist dies ein Indiz für eine Konfusion von Buchstabenklassen oder Segmentierungsfehler. Bestimmte Klassen werden besonders häufig miteinander verwechselt, wie die Konfusionsmatrizen in Abbildung 8.6 deutlich machen. Abbildung 8.6 (b) zeigt zudem, dass die Klassifikation von segmentierten Buchstaben auch bei bekannten Trennstellen problematisch ist, z.B. aufgrund überlappender Buchstabenpaare.

Weitere Untersuchungen zur Treffergenauigkeit der Buchstabenklassifikation

Tabelle 8.5. Abhängigkeit der Klassifikationsgüte von der Anzahl an Proben. Verwendete Merkmale: F74+KF-simple.

Proben	1253	2067	4102	6137	8172
A_b	84,19±1,30%	89,00±1,21%	91,66±1,43%	91,90±2,15%	92,63±1,97%
Zulässiger, maximaler Rang i ($A_b^0 = A_b$)	0	1	2	3	4
A_b^i	93,80±8,95	97,19±7,82	98,00±7,14	98,25±6,97	98,99±5,49

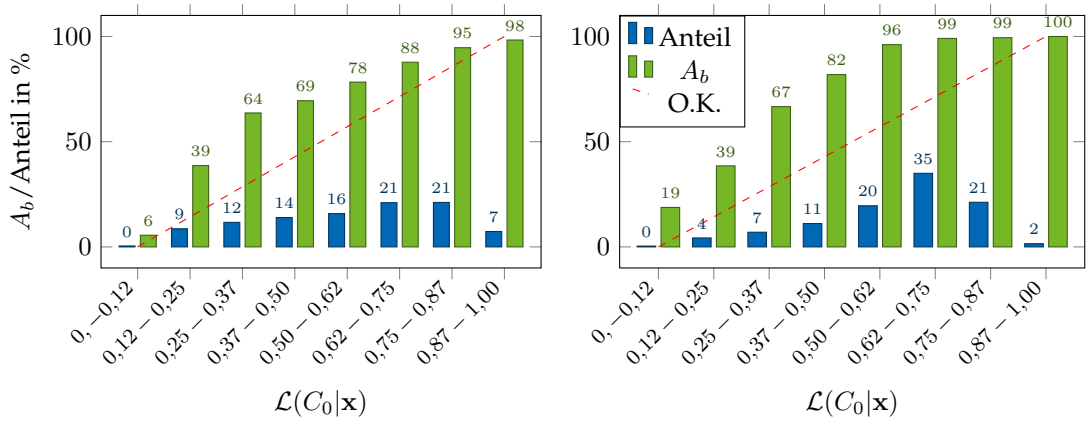
wurden bezüglich der Anzahl verwendeter Trainingsproben gemacht. Tabelle 8.5 zeigt, dass weniger als zweitausend Proben zu Problemen bei der Klassifizierung führen. Ab viertausend Proben lässt sich hingegen keine eindeutige Verbesserung der Treffergenauigkeit mehr durch Erhöhung der Probenanzahl erzielen (stattdessen sollte dann die Anzahl der Schreiber erhöht werden). In Tabelle 8.5 wird außerdem die alternative Treffergenauigkeit A_b^i untersucht. Bei dieser ist es für eine True Positive (TP) Wertung hinreichend, wenn die tatsächliche Klasse einen Rang $\leq i$ annimmt. Aus A_b^4 geht hervor, dass sich mittels Ranking die Anzahl der potentiellen Klassen in 99% aller Fälle auf 5 reduzieren lässt. Hierdurch lässt sich beispielsweise die Fehlerkorrektur stark beschleunigen. Allerdings lassen sich in diesem Fall 1% der erkannten Buchstaben nicht mehr durch die verwendeten n-Gramme korrigieren (jedoch durch den Abgleich mit dem Vokabular oder, bei Erkennung ganzer Sätze, durch n-Gramme auf Wortebene).

8.2.3 Verlässlichkeit der SVM-Likelihoods

Die Rankings der SVMs basieren auf der approximierten Log-Likelihood $\mathcal{L}$ (der LIBSVM Implementation). Diese ist entscheidend für die n-Gramm basierte Nachverarbeitung, weshalb in diesem Abschnitt die Zuverlässigkeit von $\mathcal{L}$ untersucht werden soll.

Bei der Handschrifterkennung unterscheiden sich die Test- stets merklich von den Trainingsproben. Dies gilt insbesondere für die Experimente in diesem Kapitel, da hier Trainings- und Testproben von unterschiedlichen Schreibern stammen. Da die SVMs ausschließlich die Merkmalsvektoren der Trainingsproben nutzen, um die Log-Likelihood $\mathcal{L}(C_i|\mathbf{x})$ zu berechnen, ist diese umso unzuverlässiger, je stärker die extrahierten Merkmale von den Testproben abweichen.

Es wurden Experimente durchgeführt, die bestätigen, dass – abhängig von den verwendeten Merkmalen sowie den Trainings- und Testproben – eine deutliche Abweichung der prädizierten Log-Likelihood $\mathcal{L}(C_0|\mathbf{x})$ und der tatsächlichen Treffergenauigkeit A_b auftritt. Abbildung 8.7 zeigt A_b über Intervallen von $\mathcal{L}(C_0|\mathbf{x})$, wobei A_b zumeist höher als der geschätzte Wert $\mathcal{L}(C_0|\mathbf{x})$ ist. Abbildung 8.7 (a) zeigt, dass für das Intervall $\mathcal{L}(C_0|\mathbf{x}) \in [0{,}62; 0{,}75]$ durchschnittlich 88% statt der erwarteten 69% der Klassifikationen korrekt sind. Bei Verwendung der *KF-simple* Merkmale ist dieses Verhalten noch stärker ausgeprägt (siehe Abbildung 8.7 (b)). Zudem liegen



(a)F74 (b)KF-simple

Abbildung 8.7. Vergleich der Log-Likelihood $\mathcal{L}(C_0|\mathbf{x})$ für die jeweils erste Klasse eines Rankings und der Häufigkeit, mit welcher diese Klasse tatsächlich korrekt ist (Treffergenauigkeit A_b). Der Anteil an Proben, deren $\mathcal{L}(C_0|\mathbf{x})$ sich in den entsprechenden Intervallen befinden, ist jeweils angegeben. Abbildung (a) zeigt die Ergebnisse für die Merkmalsgruppe *F74* und Abbildung (b) für KF-simple. Zum Vergleich ist die optimale Korrelation (O.K.) zwischen $\mathcal{L}(C_0|\mathbf{x})$ und A_b eingetragen.

57% aller Log-Likelihoods $\mathcal{L}(C_0|\mathbf{x})$ zwischen $0{,}62$ und 1, und haben eine Treffergenauigkeit A_b von mindestens 99%. Daher lässt sich mit akzeptabler Zuverlässigkeit schätzen, ob ein Buchstabe korrekt klassifiziert wurde.

8.2.4 Convolutional-Neural-Networks basierte Erkennung

Wie in Tabelle 8.6 zu sehen ist, sind die mit Convolutional-Neural-Networks (CNNs) erzielten Treffergenauigkeiten in etwa so hoch wie bei den mit den Merkmalen der Literatur trainierten Support Vector Machines (SVMs), jedoch geringer als bei mit *KF-simple* trainierten SVMs. Es wurden verschiedene CNN-Architekturen implementiert und getestet, allerdings tritt stets ein deutliches Overfitting auf. Zudem lässt sich die verwendete CNN Bibliothek (Dlib) nicht ohne größeren Aufwand

Tabelle 8.6. Genauigkeit der CNN basierten Buchstabenklassifikation (isolierte Form, Lernrate 0,01).

Abhängigkeit von der Trainingsdauer			Abhängigkeit von der Anzahl Proben		
Epochen	A_b Training in %	A_b Test in %	Min Batch Size	A_b Training in %	A_b Test in
20	77	59	30	100	60
50	100	72	128	96	71
100	100	75	300	100	78

in das Worterkennungssystem integrieren und ist zudem nicht mit älteren GPUs kompatibel. Daher werden im Weiteren nur die SVM und ASM basierten Ansätze untersucht.

8.3 Worterkennung

Die bisherigen Zwischenergebnisse der Segmentierung sowie der Klassifizierung und des Rankings der Buchstabenklassen stehen in direktem Bezug zu der erwarteten Güte der Erkennung handgeschriebener arabischer Wörter. Letztere hat eine große Bedeutung für verschiedene praktische Anwendungen, wobei in diesem Forschungsbeitrag die Erkennung von Wörtern in arabischer Prosa im Vordergrund steht. In diesem Abschnitt werden hierzu zwei wesentlichen Gütekriterien untersucht:

A'_w Die direkte Treffergenauigkeit

A_w Die Treffergenauigkeit nach Abgleich mit einem Vokabular

A'_w gibt den Prozentsatz von Wörtern an, deren erkannte Buchstabensequenzen $\mathbf{w}_r$ komplett mit $\mathbf{w}_t$ – den Buchstabensequenzen der Grundwahrheiten (GT) – übereinstimmen (d.h. es treten keinerlei Segmentierungs- oder Erkennnungsfehler auf). Hingegen definiert A_w die Treffergenauigkeit nach dem Abgleich mit einem vorgegebenem Vokabular. A_w lässt sich berechnen, indem zunächst jede Vokabel mit $\mathbf{w}_r$ verglichen und das Vokabular entsprechend der Levenstein Distanz aufsteigend geordnet wird. Die prozentuale Häufigkeit, mit welcher $\mathbf{w}_t$ mit dem ersten Eintrag des neu geordneten Vokabulars übereinstimmt, ergibt nun A_w.

Worterkennung mit SVMs und ASMs. In das System zur Worterkennung wurden zwei verschiedene Klassifikatoren zur Buchstabenerkennung eingebunden. SVMs, die mit verschiedenen Merkmalsvektoren trainiert wurden, haben entsprechend der Auswertung in Abschnitt 8.2 besser abgeschnitten, als der ASM basierte Klassifikator. Allerdings lassen sich die ASMs nachträglich entsprechend der bei den segmentierten Buchstaben auftretenden Abweichungen modifizieren, wodurch sich das Overfitting auch ohne Erweiterung der Trainingsdatenbank reduzieren lässt. Die folgenden Experimente zeigen, dass diese Fähigkeit bei der Worterkennung von Vorteil ist, insbesondere, um den Einfluss der Segmentierungsmethode auf die

resultierenden, zu klassifizierenden Buchstaben auszugleichen. Allerdings wird auch gezeigt, dass sich die gewonnenen Erkenntnisse einsetzen lassen, um die Trainingsdatenbank für die SVMs erfolgreich anzupassen.

Bevor die experimentellen Ergebnisse der Worterkennung vorgestellt werden, soll im nächsten Abschnitt zunächst eine Auswertung der n-Gramm basierten Fehlerdetektion und der Konfidenz einer erkannten Buchstabensequenz w_r erfolgen.

8.3.1 Fehlerdetektion und Konfidenz der Worterkennung

In der Praxis ist es unrealistisch, und gegebenenfalls auch nicht notwendig, sämtliche Wörter eines handgeschriebenen Textes zu erkennen. Daher wird ein Maß benötigt, welche es erlaubt, voraussichtlich falsch erkannte Wörter zu detektieren, um sie bei der Interpretation oder der Analyse eines Textes gegebenenfalls zu ignorieren oder gesondert zu behandeln. In diesem Abschnitt werden hierzu drei geeignete Bewertungskriterien untersucht. Diese basieren auf den Rankings der Support Vector Machines (SVMs), der stochastischen Analyse der n-Gramme sowie dem Vergleich der erkannten Buchstabensequenz w_t mit dem Vokabular.

Analyse der Trigramme. Durch die in Kapitel 6 vorgestellte Analyse der in $\mathcal{V}_{50k}$ vorkommenden Trigramme, lassen sich bereits viele Fehler sicher erkennen. Wie der untersten Zeile in Tabelle 8.7 zu entnehmen ist, werden etwa die Hälfte aller Wörter mit ein oder mehreren Fehlern auch als solche detektiert. Es ließen sich noch mehr Fehler detektieren, indem auch Trigramme mit geringer Wahrscheinlichkeit $P(CBA) \neq 0$ beachtet würden. Hierdurch träten dann jedoch auch False Positive Deklarationen auf (andernfalls werden beim Vokabular $\mathcal{V}_{50k}$ keine korrekt erkannten

Tabelle 8.7. Prozentualer Anteil an Wörtern mit Erkennungsfehlern, die durch die Analyse von Trigrammen detektiert wurden.

		Anzahl an Fehlern pro Wort		
	Vokabular	1	2	3
Simulierte Fehler:	$\mathcal{V}_{50k}$ komplett	$44{,}18 \pm 0{,}01$	$58{,}28 \pm 0{,}01$	$66{,}3 \pm 0{,}01$
	$\mathcal{V}_{50k}$ zufällige Wörter	$32{,}0 \pm 0{,}01$	$39{,}27 \pm 0{,}12$	$42{,}52 \pm 0{,}08$
Bildverarbeitung:	**SVM**$_{Gt}$**:**[1]	$44{,}0 \pm 0{,}48$	**SVM:**[2]	$54{,}7 \pm 0{,}5$

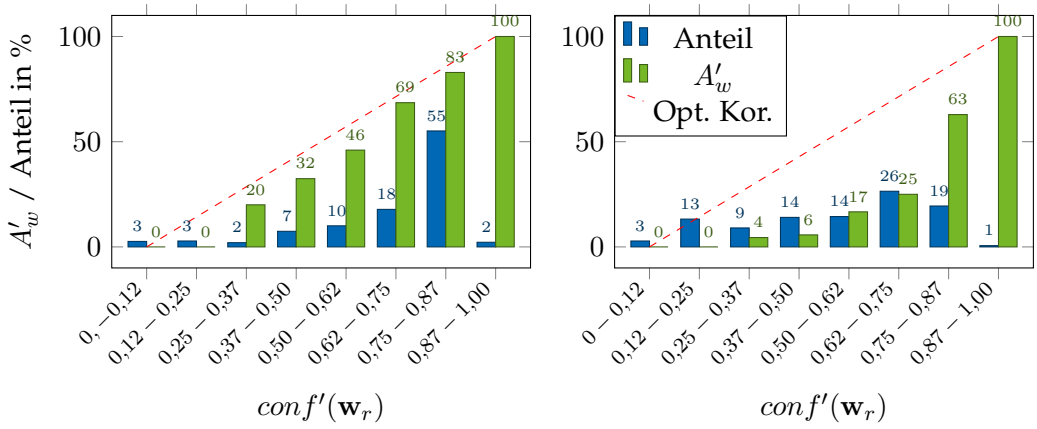
[1] Manuelle Segmentierung (hauptsächlich Substitutionsfehler).
[2] Automatische Segmentierung (zusätzliche Fehler durch Über- und Untersegmentierung).

Wörter als fehlerhaft eingestuft).

Ergänzend zu den Erkennungsfehlern, welche durch die vorgestellten Methoden der Bildverarbeitung resultieren, wurden Experimente an Buchstabensequenzen $\mathbf{w}_s$ mit simulierten Fehlern durchgeführt. Zum einen wurde hierzu das gesamte Vokabular $\mathcal{V}_{50k}$ verwendet. Zum anderen wurden 50.000 – analog zu dem in Kapitel 4.2 beschriebenem Verfahren zur Generierung von Pseudotexten – zufällig gewählte Vokabeln $\in \mathcal{V}_{50k}$ eingesetzt. Zur Simulation der Fehler wurde jede Vokabel $\mathbf{w}$ manipuliert, indem ein, zwei oder drei Buchstaben $w_i \in \mathbf{w}$ zufällig geändert wurden. Der Tabelle 8.7 ist zu entnehmen, dass die Detektionsrate fehlerhaft erkannter Wörter nur mäßig mit der Fehleranzahl steigt. Ein Grund hierfür mag sein, dass mehrere Fehler in denselben Trigrammen auftreten. Die Wahrscheinlichkeit hierfür hängt von der Länge der Wörter ab: Die zufällig gewählten Wörter sind häufig sehr kurz. Die in ihnen auftretende Fehler werden daher deutlich seltener detektiert, da sie weniger Buchstaben in der Wortmitte aufweisen (d.h. Buchstaben mit mindestens ein bis zwei Nachbarn auf jeder Seite). Es sind jedoch eben jene Buchstaben, die sich gleichzeitig in bis zu drei unterschiedlichen Trigrammen befinden. Die Wahrscheinlichkeit, dass zwei oder drei veränderte Trigramme (statt nur einem) zufällig eine Wahrscheinlichkeit $P(CBA) \neq 0$ aufweisen, und sich der Fehler deshalb nicht detektieren lässt, ist bei Buchstaben in der Wortmitte also entsprechend geringer.

Konfidenz aufgrund der Trigramme und Rankings. Durch das auf n-Grammen aufbauende Maß lässt sich prinzipiell abschätzen, ob eine erkannte Buchstabensequenz $\mathbf{w}_r$ ein Wort der arabischen Sprache repräsentiert. Allerdings werden dabei noch keine aus der Handschrift extrahierten Merkmale einbezogen. Um dies zu tun, werden nun die zu jedem Buchstaben w_i gehörigen Log-Likelihoods $\mathcal{L}(C_0^i|\mathbf{x}_i)$ hinzugezogen. Hiermit wird die Konfidenz berechnet, mit der $\mathbf{w}_r$ – in Abhängigkeit der beobachteten Merkmalsvektoren $\mathbf{x}_i \Rightarrow w_i = C_0^i \,|w_i \in \mathbf{w}_r$ – voraussichtlich die korrekte Transkription des handgeschriebenen Wortes darstellt. Sei E_{tri} die Anzahl derjenigen Trigramme innerhalb von $\mathbf{w}_r$, welche eine Wahrscheinlichkeit von 0 aufweisen, und C_0^i die Buchstabenklasse, welche im i-ten Ranking den Rang 0 innehat, so berechnet sich die Konfidenz von $\mathbf{w_r}$ wie folgt:

$$conf'(\mathbf{w}_r) = \frac{1}{2}\left(\frac{\sum_{i=1}^{n(\mathbf{w}_r)} \mathcal{L}(C_0^i|\mathbf{x}_i)}{n(\mathbf{w}_r)} + 1 - \frac{E_{\mathsf{tri}}}{n(\mathbf{w}_r)}\right). \tag{8.2}$$



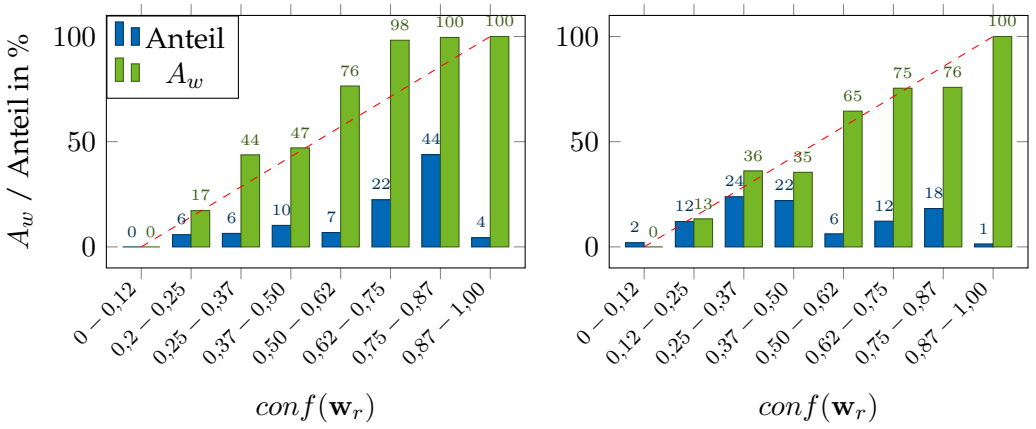
(a)manuelle Segmentierung (b)automatische Segmentierung

Abbildung 8.8. Prädiktion des Erfolges der direkten Worterkennung. Die Diagramme zeigen die Treffergenauigkeit A'_w über verschiedene Intervalle der berechneten Konfidenz (die verwendeten Merkmalsgruppen sind *KF-simple* + F74). Der prozentuale Anteil von Messungen innerhalb eines Intervalls ist jeweils angegeben, außerdem wird die optimale (angestrebte) Korrelation von $conf'(\mathbf{w})$ und A'_w visualisiert.

.

Diese Konfidenz $conf'(\mathbf{w}_r) \in [0{,}1]$ trifft nun aufgrund der Analyse der Trigramme und der Rankings eine Aussage darüber, wie ähnlich $\mathbf{w}_r$ dem tatsächlichem Wort $\mathbf{w}_t$ voraussichtlich ist. Im optimalem Fall gilt $conf'(\mathbf{w}_r) \approxeq P(\mathbf{w}_r = \mathbf{w}_t)$.

Abbildung 8.8 zeigt die direkte Treffergenauigkeit A'_w der Worterkennung über verschiedenen Intervallen für $conf'(\mathbf{w}_r)$. Dabei ist in Abbildung 8.8 (a) eine deutliche Korrelation zwischen der Konfidenz und der mittels Grundwahrheiten (GT) ermittelten Treffergenauigkeit ersichtlich. Wird eine manuelle Segmentierung eingesetzt, haben über die Hälfte aller $\mathbf{w}_r$ eine Konfidenz von $0{,}75$ bis $0{,}87$. Von diesen werden 83% korrekt erkannt ($A'_w = 83$), was auch ungefähr hundert mal der mittleren Konfidenz dieses Intervalles entspricht ($100 \cdot 0{,}81 = 81$). Daraus folgt, dass sich die Treffergenauigkeit A'_w hier verlässlich durch die Konfidenz voraussagen lässt. Ist die Konfidenz niedriger, so sinkt auch A'_w. Aufgrund der häufigen Segmentierungsfehler ist dieser Zusammenhang entsprechend weniger stark ausgeprägt, wenn das automatische Segmentierungsverfahren verwendet wird (siehe Abbildung 8.8 (b)). Dem ließe bei Bedarf sich entgegenwirken, indem zusätzliche One-vs-All SVMs mit den Proben aller Buchstabenklassen und zusätzlichen Über- und Untersegmentierungen als Restklasse trainiert würden.



(a)manuelle Segmentierung (b)automatische Segmentierung

Abbildung 8.9. Prädiktion des Erfolgs der Worterkennung nach Abgleich mit einem Vokabular. Die Diagramme zeigen die Treffergenauigkeit A_w der Worterkennung nach Abgleich mit dem Vokabular $\mathcal{V}_{5k}$ über der zuvor berechneten Konfidenz für (a) manuelle und (b) automatische Segmentierung.

Konfidenz aufgrund des Abgleichs mit dem Vokabular. Wird die erkannte Sequenz $\mathbf{w}_r$ mit dem verwendeten Vokabular verglichen, so wird statt A'_w die Treffergenauigkeit A_w verwendet. Außerdem wird die Konfidenz in diesem Fall aufgrund der Levenstein Distanzen δ_0 berechnet (dem Abstand zwischen $\mathbf{w}_r$ und dem ähnlichsten Eintrag des Vokabulars):

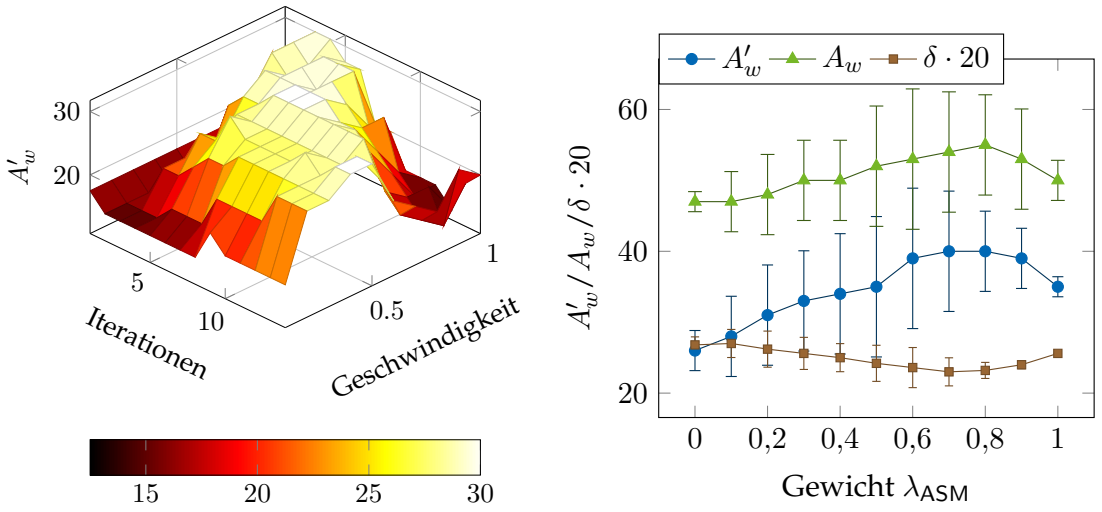
$$conf(\mathbf{w}_r) = \frac{1 - \delta_0}{n(\mathbf{w}_r)} \tag{8.3}$$

Abbildung 8.9 zeigt, dass sich auch in diesem Fall über die Konfidenz abschätzen lässt, ob ein Wort korrekt oder fehlerhaft klassifiziert wurde. Sofern manuelle Segmentierung verwendet wird, haben 70% aller Wörter eine Konfidenz von $conf(\mathbf{w}_r) \geq 0{,}62$ und weisen eine Treffergenauigkeit von $A_w \geq 98\%$ auf. Auch bei automatischer Segmentierung stimmen $conf(\mathbf{w}_r)$ und A_w gut überein.

Die Konfidenz ermöglicht es, zukünftige Arbeitsschritte zu unterstützen, wie beispielsweise die Korrektur mit Sprachmodellen. Wörter mit geringer Konfidenz sollten als nicht erkannt markiert oder – z.B. bei Verwendung von n-Grammen auf Wortebene – entsprechend gering gewichtet werden.

8.3.2 Optimierung der Worterkennung

Die Optimierung des Worterkennungssystems unterscheidet sich hinsichtlich des verwendeten Klassifikators. Die optimalen Hyperparameter C und γ der Support Vector Machines (SVMs) müssen vor dem eigentlichen Training durch Validierung



(a)ASM (b)Hybride Erkennung (SVM&ASM)

Abbildung 8.10. Optimierung und Fusion von SVM und ASM basierter Erkennung (a) Optimierung der Parameter des Gradientenabstiegverfahrens für ASM basierten Klassifikation. Die Zielfunktion ist hierbei allerdings nicht die Treffergenauigkeit der Buchstaben-, sondern der Worterkennung (A'_w). (b) Treffergenauigkeit der Worterkennung bei kombinierten Rankings von ASMs und SVMs: $(1 - \lambda_{\mathsf{ASM}})$SVM $+\lambda_{\mathsf{ASM}}$ASM.

bestimmt werden. Hierzu dient die Zielfunktion A_b (siehe Abschnitt 8.2.1). Bei Active Shape Modells (ASMs) ist es hingegen nur erforderlich, die Eigenwerte und -vektoren sowie die Erwartungswerte im Vorfeld mittels der Trainingsproben der IESK-arDB$_{\mathsf{OnlineLetter}}$ zu berechnen. Die Parameter des ASM basierten Erkennungsverfahrens lassen sich jedoch auch während der Worterkennung mit der Zielfunktion A'_w optimieren.

Abbildung 8.10 (a) zeigt die Optimierung der Geschwindigkeit (Lernrate) und der Anzahl an Iterationen des Gradientenabstiegsverfahrens (GD). Indem A'_w als Zielfunktion eingesetzt wird, lassen sich auch Segmentierungsfehler in den Optimierungsprozess mit einbeziehen. Hierdurch lässt sich beispielsweise vermeiden, dass sich ein ASM zu stark an einen über- oder untersegmentierten Buchstaben anpasst.

Andere Anpassungen, wie die Rotation des ASM oder die Modifikation der Kashida Länge, werden an dieser Stelle nicht mittel A'_w optimiert, sondern gehören (wie die verwendeten Eigenwerte) zu den Freiheitsgraden des ASM. Die Anpassung erfolgt hier also während der Erkennung eines einzelnen Buchstabens. Zu testen sind lediglich sinnvolle Grenzwerte, und welche Kombination von aktiven Parametern (Kashidalänge, Rotation, Scherung, Stauchung) für die untersuchten Testproben günstig ist. So wurde festgestellt, dass sich A'_w deutlich erhöht, wenn

die Länge der Kashida initial um 50% reduziert und bei den anschließenden Iterationen um maximal $\pm 50\%$ variiert wird. Hierdurch lässt sich mittels ASMs, trotz schlechterer Treffergenauigkeit A_b, eine bessere Worterkennung erzielen, als dies mit SVMs (die mit der originalen IESK-arDB$_{\mathsf{OnlineLetter}}$ trainiert wurden) möglich ist.

Fusion von ASM und SVM Rankings. Da der auf ASM basierende Klassifikator und die verwendeten Merkmale sich deutlich von den SVMs unterscheiden, erscheint eine Verknüpfung beider Rankings sinnvoll. Dem Diagramm in Abbildung 8.10 (b) ist jedoch zu entnehmen, dass sich durch eine solche Kombination der beiden Ansätze nur eine geringfügige und aufgrund der Varianz nicht eindeutige Erhöhung der Treffergenauigkeit A'_w und A_w erzielen lässt.

8.3.3 Experimente zur Worterkennung

Ein wesentliches Ziel dieser Dissertation besteht darin, Verfahren zur automatischen Erkennung handgeschriebener arabischer Wörter zu untersuchen, die auf expliziter Segmentierung basieren. Die hierzu eingesetzten Komponenten, wie Segmentierungs- und Klassifikationsverfahren, wurden bereits in den vorherigen Abschnitten untersucht. Hier wird nun deren Zusammenspiel unter verschiedenen Bedingungen genauer ausgewertet.

Vergleich verschiedener Klassifikatoren. Im ersten Abschnitt (a) der Tabelle 8.8 werden verschiedene Klassifikatoren miteinander verglichen.

Im Fall von **SVM**$_{\mathsf{precl}}$ wurde zunächst eine Reduzierung der potentiellen Klassen durchgeführt (mittels der in der Vorklassifizierung eingesetzten Taxonomien aus Abbildung 6.2 auf Seite 129). Hiernach wurde für jede Gruppe eine separate Support Vector Machine (SVM) mit den aus den Proben extrahierten Hauptkörpern der Buchstaben trainiert. Wird bei der Segmentierung ein diakritisches Zeichen teilweise oder gänzlich dem falschen Buchstaben zugeordnet, ist eine korrekte Klassifizierung nicht mehr möglich. Dieser Fall tritt in der Praxis wesentlich öfter auf, als ursprünglich angenommen wurde. Wird hingegen auf Vorklassifizierung verzichtet, so ist eine korrekte Klassifikation in einigen Fällen auch bei diakritischen Zeichen realisierbar, die nicht der erwarteten Taxonomie entsprechen (bei Buchstaben mit eindeutigem Hauptkörper sogar dann, wenn ein diakritisches Zeichen gänzlich fehlt). Somit sind mit allen Klassen und der Merkmalsgruppe *KF-simple* trainierte SVMs ohne Vorklassifizierung vorzuziehen.

SVM$_{\text{cut}}$ bezeichnet SVMs, die auf der modifizierten IESK-arDB$_{\text{OnlineLetter}}$ trainiert wurden. Hierzu wurden die Kashidas, analog zu dem im letzten Abschnitt beschriebenem Vorgehen für ASMs, angepasst. Jedoch wurde die Abweichung vom Initialwert hier zufällig durch eine Normalverteilung bestimmt

Tabelle 8.8. Ergebnisse der Worterkennung vor und nach dem Abgleich mit dem Vokabular (A'_w, A_w). Es wurde der Einfluss der verwendeten Klassifikatoren sowie unterschiedlicher Datenbanken und Vokabulare auf die Genauigkeiten (A'_w, A_w) untersucht.

(a) Automatische Segmentierung ($\mathcal{V}_{5k}$)

	SVM$_{\text{precl}}$	**SVM**	**SVM**$_{\text{cut}}$	**SVM**$_{\text{cut+nGr}}$	**ASM**	**ASM + SVM**$_{\text{cut}}$
A_w	$28{,}60 \pm 1{,}59$	$34{,}32 \pm 3{,}29$	$40{,}14 \pm 2{,}24$	$38{,}77 \pm 2{,}09$	$43{,}85 \pm 2{,}69$	$47{,}35 \pm 2{,}54$
A'_w	$11{,}44 \pm 2{,}1$	$16{,}10 \pm 2{,}69$	$21{,}18 \pm 0{,}6$	$25{,}63 \pm 0{,}89$	$25{,}31 \pm 0{,}74$	$27{,}43 \pm 1{,}14$

(b) Manuelle Segmentierung (GT, $\mathcal{V}_{5k}$)

	SVM$_{\text{precl}}$	**SVM**	**SVM**$_{\text{cut}}$	**SVM**$_{\text{nGr}}$	**ASM**
A_w	$73{,}41 \pm 1{,}64$	$84{,}76 \pm 2{,}34$	$81{,}96 \pm 2{,}42$	$82{,}01 \pm 1{,}89$	$79{,}87 \pm 2{,}09$
A'_w	$42{,}79 \pm 1{,}19$	$70{,}05 \pm 1{,}37$	$63{,}91 \pm 1{,}36$	$73{,}43 \pm 0{,}57$	$59{,}95 \pm 1{,}9$

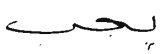
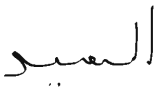
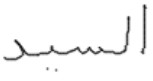
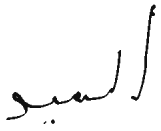
(c) Manuelle Segmentierung (GT, $\mathcal{V}_{5k}$) + SVM, verschiedene Merkmale

	F74 + KF-simple	*F74*	Momente
A_w	$84{,}76 \pm 2{,}34$	$69{,}27 \pm 4{,}79$	$67{,}40 \pm 4{,}15$
A'_w	$70{,}05 \pm 1{,}37$	$53{,}81 \pm 4{,}19$	$50{,}37 \pm 3{,}00$

(d) Automatische Segmentierung, unterschiedliche Vokabulare (SVM)

	$\mathcal{V}_{1k\_most}$*	$\mathcal{V}_{1k\_last}$*	IESK-arDB	IFN\ENIT
A_w	$40{,}74 \pm 5{,}75$	$50{,}95 \pm 1{,}65$	$51{,}36 \pm 6{,}69$	$55{,}61 \pm 6{,}74$
A'_w	$15{,}76 \pm 4{,}92$	$7{,}96 \pm 3{,}17$	$11{,}92 \pm 0{,}48$	$2{,}01 \pm 0{,}14$

(e) Automatische Segmentierung, unterschiedliche Eigenschaften der Handschrift (SVM)

	$\mathcal{V}_{5k}$	$\mathcal{V}_{5k}$ thin	$\mathcal{V}_{5k}$ stark deformiert
A_w	$40{,}14 \pm 2{,}24$	$23{,}19 \pm 1{,}04$	$32{,}80 \pm 4{,}41$
A'_w	$21{,}18 \pm 1{,}6$	$8{,}68 \pm 2{,}69$	$16{,}82 \pm 1{,}92$
Beispiel (يجب)			
Beispiel (السيد)			

*die häufigsten/seltensten Wörter aus $\mathbf{w}_{\mathcal{V}_{50k}}$ (ohne zurücklegen)

(die Anzahl der Proben wurde dabei nicht erhöht). Bei Verwendung der automatischen Segmentierung lässt sich hierdurch eine Verbesserung der Treffergenauigkeiten A'_w und A_w erzielen. Dagegen tritt bei der manuellen Segmentierung – wie aus Tabelle 8.8 (b) hervorgeht – eine Verschlechterung ein, da die Segmentierungsstellen hier eher den Kashidas der ursprünglichen Buchstabendatenbank entsprechen.

Die direkte Treffergenauigkeit A'_w lässt sich weiter erhöhen, indem anschließend die n-Gramm basierte Fehlerkorrektur durchgeführt wird ($\mathbf{SVM}_{\mathsf{cut+nGr}}$). In Kombination mit der vokabularbasierten Fehlerkorrektur ist hierdurch jedoch keine weitere Verbesserung ersichtlich.

Obwohl die SVMs eine deutlich höhere Treffergenauigkeit A_b beim Test an der Buchstabendatenbank aufweisen, ist es dennoch die ASM basierten Klassifikation, mit welcher sich – in Kombination mit der automatischen Segmentierung – die bessere Treffergenauigkeit (A'_w und A_w) der Worterkennung erzielen lässt. Dies gilt auch im Vergleich zu $\mathbf{SVM}_{\mathsf{cut}}$. Daraus lässt sich schließen, dass beim Training der SVMs auch bei Nutzung der modifizierten Proben ein Overfittung aufgrund variierender Kashidaausprägungen auftritt. Eine zukünftige Erweiterung der zum Training verfügbaren Buchstabenproben scheint daher sinnvoll.

Wie im letzten Kapitel bereits beschrieben, lässt sich durch Kombination der SVM und ASM basierten Klassifikationsergebnisse eine geringfügige Erhöhung der Treffergenauigkeit erzielen (**ASM + SVM**$_{\mathsf{cut}}$).

Einfluss der Segmentierungsergebnisse. Wie aus Tabelle 8.8 (b) hervorgeht, lassen sich bei Verwendung manueller Segmentierung 70% der Wörter korrekt klassifizieren (A'_w), nach Abgleich mit einem Vokabulars von 5.000 Wörtern sind es sogar über 80% (A_w). A'_w ist mit weniger als 30% dagegen deutlich geringer, wenn das vorgeschlagene Verfahren zur automatischen Segmentierung verwendet wird. Dies zeigt, dass sich auch mit einer deutlichen Verbesserung der Buchstabenerkennung auf annähernd 100% nur eine mäßig verbesserte Worterkennung erzielen ließe, solange nicht gleichsam die auftretenden Segmentierungsfehler minimiert werden.

Einfluss der Merkmale. Für die Ergebnisse der SVM basierten Klassifikation der Buchstaben zeigen sich neben den Trainingsdaten in erster Linie die verwendeten Merkmale verantwortlich. Dies spiegelt sich auch in der Worterkennung wieder, wie Abschnitt (c) der Tabelle 8.8 zeigt.

Einfluss des Vokabulars. Ein wesentlicher Unterschied zwischen der Erkennung einzelner Wörter, wie sie in Formularen vorkommen, und solcher innerhalb zusammenhängender Texte, betrifft das zu erwartende Vokabular. Zum einen ist ein Vokabular im letzteren Fall wesentlich umfangreicher, deckt aber dennoch nicht sämtliche Wörter ab, die in einem beliebigem Text vorkommen mögen. Zum anderen wurde in Abschnitt 3 gezeigt, dass sich viele der am häufigsten benutzten Vokabeln ähneln, wogegen eine Liste mit Städtenamen eine deutlich geringere Verwechslungsgefahr bedeutet. In Abschnitt (d) der Tabelle 8.8 wird nun untersucht, wie sich dieser Umstand auf die Erkennung der handschriftlichen arabischen Wörter auswirkt.

Es ist ersichtlich, dass $\mathcal{V}_{\mathsf{1k\_most}}$ – die tausend häufigsten arabischen Wörter – eine bessere Treffergenauigkeit A'_w aufweisen, als $\mathcal{V}_{\mathsf{1k\_last}}$, die tausend seltensten arabischen Wörter (aus $\mathcal{V}_{\mathsf{50k}}$). Mit der Treffergenauigkeit A_w verhält es sich jedoch anders herum. Dies entspricht der Erwartung, welche sich auf die zuvor berechnete Ähnlichkeit innerhalb des Vokabulars stützt (siehe Kapitel 4, Abbildung 4.9 auf Seite 88), und die demnach durchaus eine Abschätzung der erwarteten Güte der Worterkennung erlaubt. Es ist anzumerken, dass Wörter aus $\mathcal{V}_{\mathsf{1k\_last}}$ meist Aufschluss über den Inhalt des Textes geben. Dagegen enthalten die häufigen Wörter aus $\mathcal{V}_{\mathsf{1k\_most}}$ für sich genommen eher wenig Information, sind jedoch entscheidend für die Grammatik und daher gegebenenfalls nützlich für die Nachverarbeitung mit einem Sprachmodell. Bei der IFN/ENIT-Datanbank ist die beobachtete Diskrepanz zwischen A'_w und A_w noch sehr viel ausgeprägter, da die Einträge hier oft Städtenamen sind, die sich aus mehreren Wörtern zusammensetzen und sich sich zudem viele Vokabeln signifikant von allen anderen unterscheiden.

Abbildung 8.11 zeigt die Treffergenauigkeit A_w in Abhängigkeit der Vokabulargröße. Die ersten drei Reihen beziehen sich dabei auf simulierte Erkennungsergebnisse $\mathbf{w}_r$ mit je ein, zwei und drei Fehlern (d.h. die gemessene Levenstein Distanz entspricht den simulierten Fehlern in Tabelle 8.7 auf Seite 176). Bei der vierte Reihe (δ_{sim}) wird der Fehler der Worterkennung entsprechend der bei der Segmentierung und Klassifikation der Buchstaben gemessenen Verteilungen simuliert (d.h. es können sowohl $\mathbf{w}_r$ mit mehreren als auch ohne Fehler auftreten). Wie ersichtlich ist, weicht A_w für die Reihe mit realen Fehlern (δ_{real}) deutlich von den durch die Simulation geschätzten Ergebnissen ab (A_w ist weniger sensitiv bezüglich

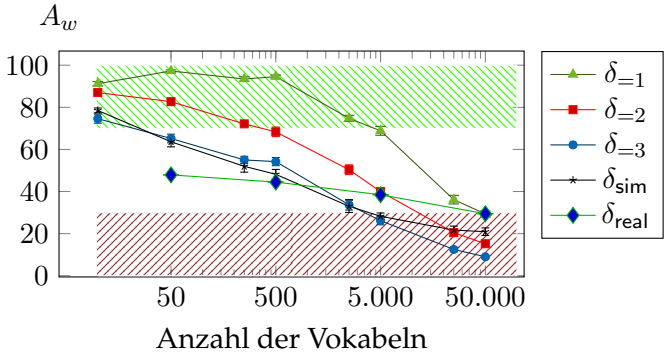


Abbildung 8.11. Abhängigkeit der Treffergenauigkeit der Worterkennung A_w vom Umfang des Vokabulars $\mathcal{V}_{\subseteq}\mathcal{V}_{\text{50k}}$ und der Levenstein Distanz δ zwischen der erkannten Buchstabensequenz $\mathbf{w}_r$ und dem tatsächlichem Wort $\mathbf{w}_t$.

der Größe des Vokabulars). Dies mag an einer weniger homogenen Verteilung der tatsächlich auftretenden Fehler liegen, die darüber hinaus in gewissem Maße von den einzelnen Vokabeln abhängt.

Einfluss der Handschrift. Eigenarten der Handschrift wirken sich sowohl auf die Segmentierung als auch die Klassifizierung von Buchstaben aus. Sowohl eine unsaubere Handschrift mit schwach ausgeprägten Merkmalen oder eine geringe Auflösung der Bildproben, als auch schlicht eine starke Abweichung von den Trainingsdaten, haben eine kritische Auswirkung auf die Treffergenauigkeit A_w. Im Vorfeld wurden beispielsweise mit dem für die Akquirierung der Buchstaben verwendeten online Stift einige deutsche und englische Sätze geschrieben, und mit einem mitgeliefertem OCR Programm erkannt (arabische Schrift wird durch die Software nicht unterstützt). Der Unterschied zwischen sehr sorgsamer, zwei Zeilen eines karierten Papiers ausfüllender Schrift, und gewöhnlicher, eher untersetzter Handschrift, war in diesem Fall eindeutig: Nur die sorgsame Handschrift wurde weitestgehend richtig erkannt, von der anderen oft nicht einmal ein einzelnes Wort. Da offline Worterkennung aufgrund der reduzierten Informationen noch anspruchsvoller ist, wird auch hier von einer entsprechende Anfälligkeit gegenüber unsauberer Handschrift ausgegangen. Wie Abschnitt (e) der Tabelle 8.8 zeigt, ist insbesondere eine stark reduzierte Auflösung problematisch.

8.4 Ergebnisse der Sequenz-zu-Sequenz Transkription

Zusätzlich zu dem vorgestellten auf expliziter Segmentierung basierenden System, wird an dieser Stelle der auf impliziter Segmentierung basierende Ansatz

mittels Long-Short-Term-Memory Netzwerk (LSTM) mit Connectionist Temporal Classification Funktion (CTC) ausgewertet. Verwendet wurde hierzu das rnnlib Framework für C++ und Python (siehe Abschnitt 7.3).

Zum Trainieren wurden die Partitionen a,b und c, zum Validieren Partition e und zum Testen Partition d der IFN\ENIT Datenbank eingesetzt. Zur Auswertung dienen folgende Fehlermaße:

ctc Fehler	Connectionist Temporal Classification (die zum Training verwendete Zielfunktion)
Löschungen in %	Anzahl aller fehlenden Buchstaben multipliziert mit $\frac{100}{N_t}$, wobei N_t die Anzahl an Buchstaben aller verwendeter Proben entsprechend der GT bezeichnet
Einfügungen in %	Zusätzliche Buchstaben multipliziert mit $\frac{100}{N_t}$,
Substitutionen in %	Anzahl falsch klassifizierter Buchstaben multipliziert mit $\frac{100}{N_t}$,
Label-Fehler in %	Summe der Löschungen, Einfügungen und Substitutionen multipliziert mit $\frac{100}{N_t}$,
Sequenzfehler in %	Prozentsatz der Wörter, deren Transkription w mindestens eine Löschung, Einfügung oder Substitution aufweist (entspricht $100 - A'_w$)

Im Gegensatz zu dem zuvor vorgestellten Ansatz, ist die Segmentierung hier Teil der Klassifizierung und muss daher trainiert werden. Dabei werden Kashida und Leerzeichen zwischen PAW oder Wörtern als zusätzliche (dominante) Klasse gesehen, wie in Kapitel 7.3 erläutert wurde. Auch die Korrektur der Klassifizierung wird hier nicht separat und explizit in der Nachverarbeitung (durch n-Gramme) vorgenommen, sondern simultan und in Abhängigkeit der verfügbaren Proben trainiert.

Abbildung 8.12 auf Seite 187 zeigt, wie sich die beschriebenen Fehlermaße über die Trainingsepochen hin entwickeln. Das Training erfordert bei 60 Epochen etwa 3 Tage. Hiernach werden jedoch auch die Testproben relativ sicher erkannt. Insbesondere Segmentierungsfehler (Einfügungen und Löschungen), treten nach erfolgreichem Training seltener auf, als dies bei den untersuchten Methoden zur expliziten Segmentierung der Fall ist. Dies bestätigt, dass die Segmentierung der Wörter in Buchstaben der wohl problematischste Aspekt der Erkennung allgemeiner arabischer Handschrift ist (Tests an manuell selektierten Proben, die problematische Buchstaben oder Buchstabenpaare ausschließen, zeigen dies weniger deutlich).

Daher scheint es sinnvoll, das Segmentierungsproblem als Klassifikationsproblem von Kashidas bzw. Leerzeichen zu umschreiben. Allerdings besteht beim impliziten Ansatz eine erhöhte Gefahr des Overfittings: Werden die verwendeten Trainingsproben auf ein Drittel reduziert, werden bereits sehr viel schlechtere Ergebnisse erzielt. Zudem berichten Gheith et al. von einem explizitem Segmentierungsansatz mit erweiterten Key Features und sehr geringer Über- und Untersegmentierung [3]. Dieser, oder andere neue Segmentierungsansätze, ließen sich in zukünftigen Projekten verwenden, um gegebenenfalls die bislang verwendeten zu ersetzen.

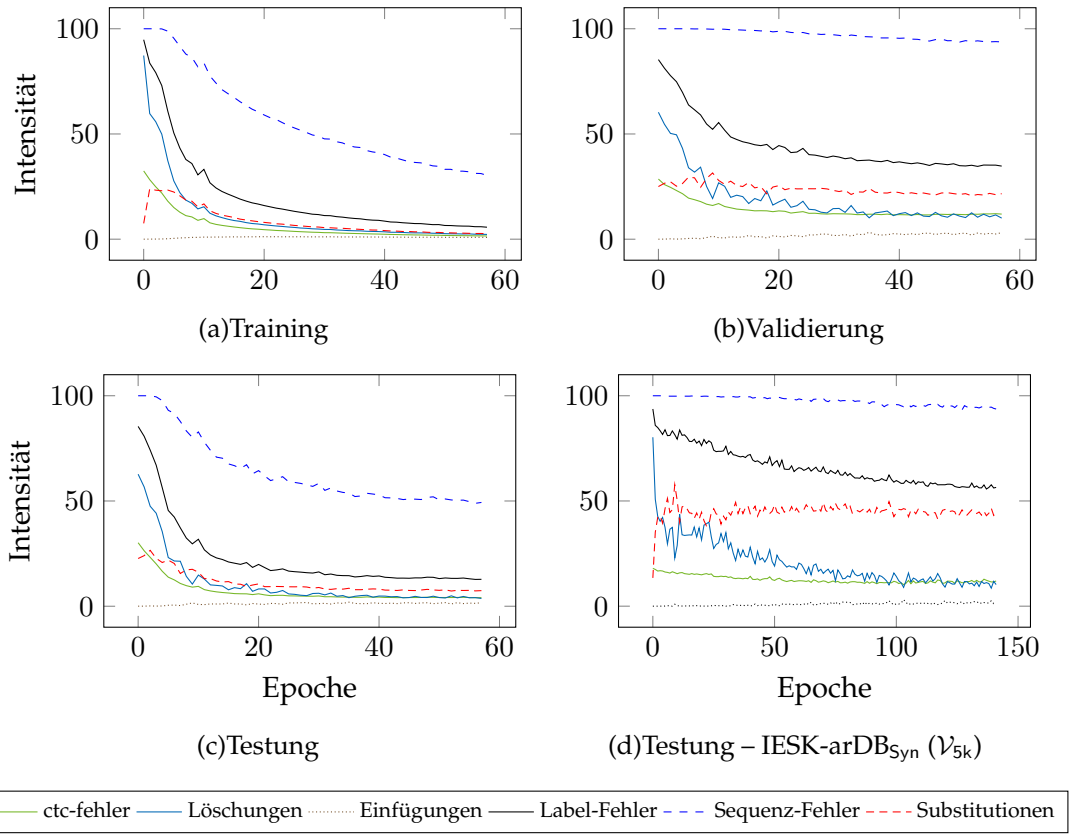


Abbildung 8.12. Verlauf (a) Training (b) Validierung und (c) Testung über mehrere Epochen. (d) Trainiert, validiert und getestet wurde das LSTM hier an der IESK-arDB$_{Syn}$ mit dem Vokabular $\mathcal{V}_{5k}$.

Abbildung 8.12 (d) zeigt die Testergebnisse für den Fall, dass – wie bei dem vorherigem Ansatz – die IESK-arDB$_{Syn}$ zum Trainieren und Testen verwendet wurde. Aufgrund der höheren Varianz der IFN\ENIT Datenbank, die Proben von mehreren hundert Schreibern enthält, liegt es nahe, dass die synthetischen Datenbanken in ihrer bestehenden Form nur bedingt zum Trainieren von LSTMs geeignet sind. Daher sollen in zukünftigen Arbeiten Glyphen von weiteren Schreibern aufgenommen und die Synthesen als Ergänzung der bestehenden Trainingsdatenbanken eingesetzt

werden. Zudem ist in diesem Zusammenhang auch die Verfremdung bestehender Proben (z.B. aus der IFN\ENIT) mittels Rauschen, simulierter Degradationen, affiner Transformationen, morphologischer Operatoren oder lokaler Modifikationen sinnvoll.

8.5 Zusammenfassung

In diesem Kapitel wurden die verschiedenen Komponenten eines Systems zur Erkennung arabischer Handschrift untersucht. Zunächst wurden Experimente an herkömmlichen und den vorgestellten synthetischen Datenbanken durchgeführt. Diese haben gezeigt, dass eine Evaluierung mit letzteren zu vergleichbaren Ergebnisse führt. Anschließend erfolgte eine detaillierte Auswertung der Verfahren zur Buchstabenklassifikation sowie der darauf aufbauenden Worterkennung. Es hat sich gezeigt, dass sich durch eine Reduzierung der Klassen-Kandidaten mittels Vorklassifizierung keine Verbesserung der Buchstabenerkennung erreichen lässt. Vielmehr lässt sich diese durch eine Kombination von Merkmalen aus der Literatur und einer Reihe von manuell konzipierter Einzelmerkmalen erzielen, die anschließend durch Support Vector Machines (SVMs) ausgewertet werden. Bei der Worterkennung wurde die beste Treffergenauigkeit hingegen durch eine Kombination von SVM und Active Shape Modells (ASMs) basierter Klassifikation erreicht.

In dieser Dissertation wurden voranging verschiedene auf expliziter Segmentierung basierende Verfahren der Schrifterkennung untersucht. Dabei wurde bestätigt, dass die häufigen Segmentierungsfehler die wesentliche Schwäche dieses Ansatzes darstellen. Das zum Vergleich untersuchte auf LSTMs basierende Verfahren, das implizite Segmentierung durch Klassifizierung der Segmentierungsstellen als zusätzliche Klasse einsetzt, ist bezüglich der Segmentierung deutlich überlegen, sofern ausreichend repräsentative Proben zum Training zur Verfügung stehen.

KAPITEL 9

Zusammenfassung und Ausblick

CHRIFT ist ein mächtiges Werkzeug zur Bewahrung von Informationen. Selbst heutzutage sind handschriftliche Aufzeichnungen weit verbreitet, und der Großteil historischer arabischer Dokumente liegt beispielsweise ausschließlich als Handschrift vor. Digitale Texte haben jedoch den Vorteil, dass sie sich leichter lesen, vervielfältigen, analysieren, klassifizieren und verarbeiten lassen. Daher hat die automatische Erkennung von handschriftlichen arabischen Texten ein hohes Anwendungspotential.

9.1 Zusammenfassung

In den letzten Jahren wurde hauptsächlich im Hinblick auf die Erkennung einzelner, einem speziellen Vokabular zugehöriger arabischer Wörter geforscht (die tunesischen Städtenamen der IFN\ENIT Datenbank sind hierfür ein gutes Beispiel). In dieser Arbeit wurde jedoch gezeigt, dass sich die Eigenschaften allgemeiner arabischer Texte hiervon drastisch unterscheiden. Doch obwohl innerhalb der letzten fünf Jahre mehrere neue Datenbanken mit arabischer Handschrift veröffentlicht wurden, sind die frei verfügbaren Datenbanken in diesem Kontext nicht ausreichend, da entweder das Vokabular stark limitiert ist oder die beigefügten Grundwahrheiten unzureichend sind. Diesen Problemen lässt sich durch die Synthese von Datenbanken begegnet.

In der Literatur sind zwei Ansätze zur Synthese arabischer Handschrift bekannt. Diese ermöglichen jedoch nicht die Generation genügend variabler Synthesen von

Wörtern und Texten. Daher wurde ein neuartiges Verfahren zur Synthese authentischer arabischer Handschrift entwickelt und in dieser Dissertation vorgestellt. Mit diesem wurde die innerhalb der Arbeitsgruppe NIT entworfene IESK-arDB durch synthetische Proben erweitert. Das in Kapitel 3 vorgestellte Verfahren basiert auf einer online Datenbank, welche zu allen Klassen des regulären arabischen Alphabetes Glyphen mehrerer Schreiber enthält. Es wurde erläutert, wie sich aus diesen Active Shape Modells (ASMs) berechnen lassen, die es ermöglichen, für jede synthetisierte Handschriftprobe neue Glyphen zu generieren. Damit wurde die Grundlage zur Synthese einer Vielzahl unterschiedlicher Handschriftproben gelegt, die – entsprechend echter Handschrift – ähnliche jedoch keine identischen Glyphen aufweisen. Eine weitere Erhöhung der Variabilität der Synthesen wurde erzielt, indem deren Erscheinungsbild durch diverse – größtenteils durch affine Transformationen realisierte – Modifikationen angepasst wurde. Zudem wurden Methoden zur Simulation unterschiedlicher Schreibwerkzeuge wie Füller und Kugelschreiber entwickelt.

Da die Erkennung von Wörtern innerhalb von handschriftlichen Textseiten zunehmend an Interesse gewinnt, wurde das Syntheseverfahren entsprechend erweitert. Um einspaltige handschriftliche Textseiten zu synthetisieren, ist in erster Linie die Simulation geeigneter Basislinien erforderlich. Hierzu wurde ein Ansatz vorgestellt, bei dem das wesentliche Verhalten der Basislinien – von links nach rechts und in Abhängigkeit der Zeilennummer – zunächst durch eine manuell erstellte Formel beschrieben wird. Hiernach wurde erläutert, wie sich die Formeln automatisch optimieren lassen, indem die Eigenschaften der resultierenden, simulierten Basislinien mit den Grundwahrheiten einiger realer Textseiten verglichen werden.

Das vorgestellte Verfahren zur Synthese arabischer Handschrift stellt einen wichtigen Beitrag dieser Dissertation dar. Daher wurden, auf Grundlage der 50.000 häufigsten arabischen Wörter, Datenbanken synthetisiert, welche in Kapitel 4 analysiert und mit bestehenden Datenbanken verglichen wurden. Hierbei wurde festgestellt, dass zwischen allen überprüften Datenbanken lediglich geringe Abweichungen bei der Verteilung der Buchstabenklassen auftreten, jedoch erhebliche bezüglich der durchschnittlichen Wortlänge und der Ähnlichkeit verschiedener Einträge innerhalb eines Vokabulars. Letztere beeinflusst deutlich die Wahrscheinlichkeit zwei

Vokabeln zu verwechseln. Die synthetischen Proben wurden den anderen Bestandteilen der in der Arbeitsgruppe NIT entwickelten IESK-arDB hinzugefügt und sind zu Forschungszwecken frei verfügbar. Da zur Datensynthese bisher nur auf Glyphen sehr weniger Schreiber zurückgegriffen werden konnte, sind die aktuell generierbaren Datenbanken im Vergleich zur IFN\ENIT Datenbank – die Proben mehrerer hundert Schreiber umfasst – weniger gut zum Trainieren von Klassifikatoren geeignet, die auf impliziter Segmentierung bzw. Deep Learning basieren.

Der zweite wesentliche Forschungsbeitrag dieser Dissertation besteht in der Entwicklung segmentierungsbasierter Verfahren zur automatischen Erkennung arabischer Handschrift. In Kapitel 5 wurden zunächst die wichtigsten Vorverarbeitungsschritte beschrieben. Bei arabischer Handschrift ist hier die Detektion und Behebung von sich vertikal überlappender Pieces of Arabic Words (PAWs) hervorzuheben. Anschließend wurde gezeigt, dass die in der Arbeitsgruppe NIT entwickelte Methode zur Segmentierung von Wörtern in einzelne Buchstaben bessere Ergebnisse erzielt, als vergleichbare Methoden aus dem Stand der Technik. Darüber hinaus haben verschiedene Experimente mit synthetischen und realen Datenbanken an dieser Stelle bestätigt, dass sich die synthetischen Datenbanken durchaus zum Testen von verschiedenen zur Schrifterkennung eingesetzten Verfahren eignen. Zudem wurde ein auf lokalen Gruppen basierendes Verfahren zur Segmentierung von Textzeilen mit gekrümmten Basislinien vorgestellt. Hier wurde gezeigt, dass sich durch die neu entwickelten Abstandsmaße für lokale Gruppen eine im Vergleich zum Standardmaß genauere Zeilensegmentierung erzielen lässt.

In Kapitel 6 wurden verschiedene Ansätze zur Klassifikation von Buchstaben vorgestellt. Hierzu zählt die Auswertung allgemeiner und speziell für die Schrifterkennung konzipierte Merkmalsvektoren durch Support Vector Machines (SVMs). Die beste Treffergenauigkeit wurde dabei mit einer Reihe neuer, manuell definierter Merkmale erzielt. Zudem wurde ein auf Active Shape Modells (ASMs) basierender Klassifikator entwickelt, der in der Lage ist, durch Segmentierung verursachte Abweichungen von den Trainingsproben auszugleichen. Es wurde gezeigt, dass sich hierdurch eine höhere Worterkennungsrate erzielen lässt, auch wenn die an der IESK-arDB$_{\mathsf{OnlineLetter}}$ und IESK-arDB$_{\mathsf{Letter}}$ erzielte Buchstabenerkennungsraten im Vergleich zu den SVMs niedriger sind.

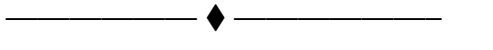
Weiterhin wurden in Kapitel 7 ein Verfahren zur Fehlerdetektion und -korrektur untersucht. Es wurde gezeigt, dass n-Gramme auf Wortebene eine zuverlässige

Detektion von Erkennungs- und Segmentierungsfehlern ermöglichen, jedoch nur eine mäßig erfolgreiche Fehlerkorrektur erzielen. Die Fehlerkorrektur durch den Abgleich der erkannten Sequenz von Buchstaben mit dem Vokabular ist hier wesentlich effektiver, hängt allerdings von der Anzahl der verwendeten Vokabeln sowie deren paarweiser minimaler Ähnlichkeit ab.

In dieser Dissertation wurden segmentierungsbasierte Verfahren zur Erkennung arabischer Handschrift untersucht. Den Schwerpunkt im Bereich der Bildverarbeitung bildete dabei das Teilproblem der Buchstabenerkennung, das zur auf expliziter Segmentierung basierenden Worterkennung gehört. Es wurde jedoch bestätigt, dass explizite Verfahren eine im Vergleich zu dem getesteten impliziten Ansatz schlechtere Worterkennung erzielen und dass dies maßgeblich den problematischen Segmentierungsschritten der expliziten Ansätze zuzuschreiben ist. Allerdings setzen implizite Verfahren , wie Long-Short-Term-Memory-Netzwerken (LSTMs) mit Connectionist Temporal Classification (CTC) Funktion, das Training an sehr umfangreichen und repräsentativen Wortdatenbanken voraus. Bei expliziter Segmentierung ist zum Training hingegen eine umfangreiche Buchstabendatenbank hinreichend. Zudem ist Overfitting bezüglich der enthaltenen Vokabeln und persönlichen oder regionalen Schreibstile hier deutlich geringer ausgeprägt.

9.2 Ausblick

Die Erkennung handschriftlicher Wörter mittels LSTMs erzielt – ausreichen geeignete Trainingsdaten vorausgesetzt – bereits gute Ergebnisse. Die verwendeten Datenbanken umfassen hier in der Regel ein mittelgroßen Vokabular, wie beispielsweise das der IFN\ENIT-Datenbank. Es wurde jedoch gezeigt, dass sich die Eigenschaften von Vokabularen allgemeiner arabischer Texte drastisch von ersterem unterscheiden. Da der Trainingsprozess von LSTMs mit CTC Funktion auch von den Permutationen der in den Proben vorkommenden Buchstabenklassen abhängt, sind Datenbanken mit einem umfangreichen Vokabular zu bevorzugen, welches für die jeweiligen Anforderungen des angestrebten Einsatzgebietes angemessen ist. An dieser Stelle soll zukünftig das Verfahren zur Handschriftsynthese eingesetzt werden, um den Umfang von verfügbaren Trainingsproben zu erhöhen. Hierzu kann auch eine Erweiterung für Sprachen wie Persisch oder Urdu erfolgen.



Appendices

ANHANG A

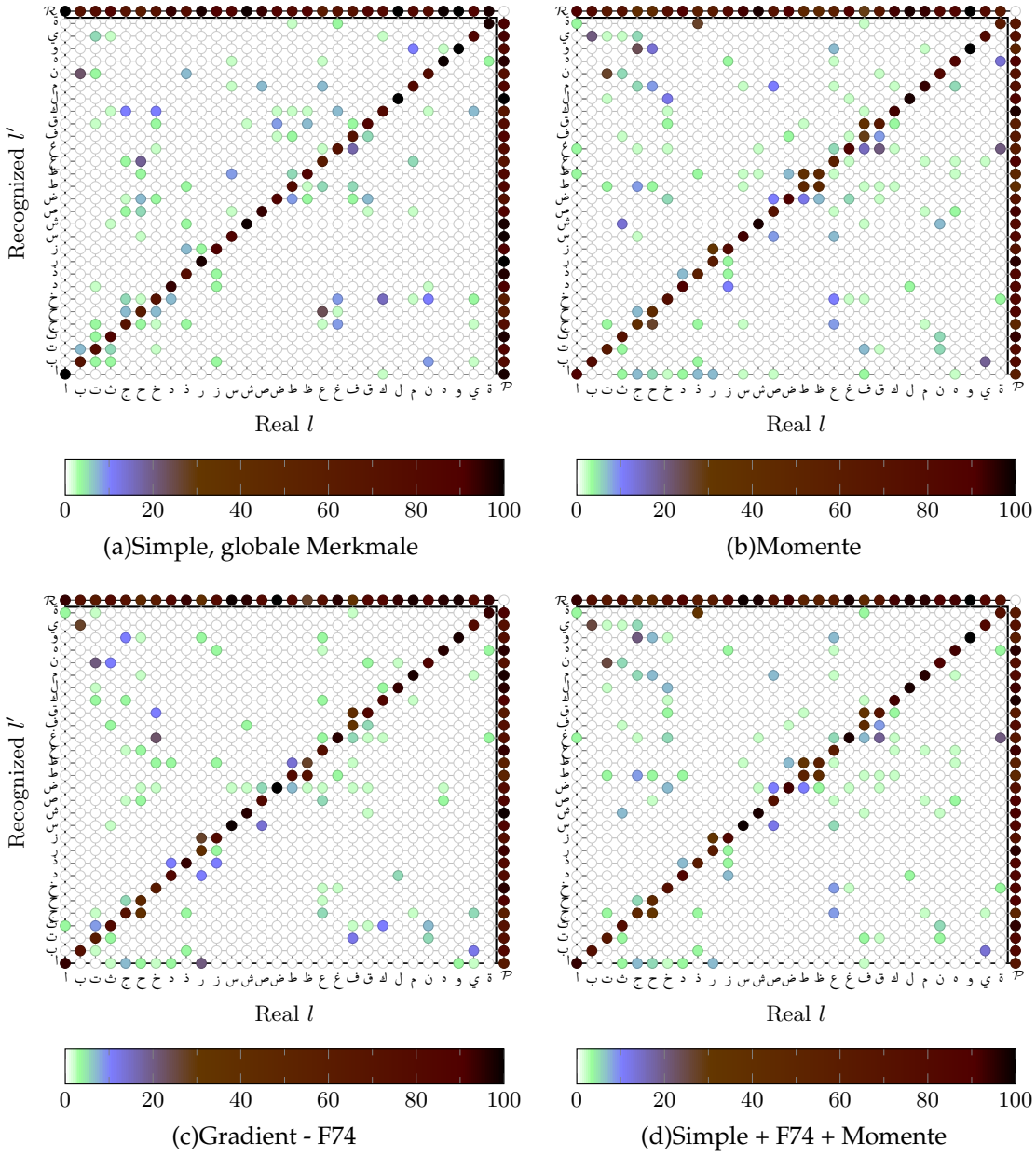
Tabellen

A.1 Schaubilder und Diagramme

Dieser Abschnitt zeigt das Beispiel einer kompletten Grundwahrheit der IESK-arDB$_{\text{SynWords}}$ sowie Konfusionsmatrizen zur Buchstabenerkennung für verschiedene Merkmalsvektoren und Buchstaben Positionen.


```
− <root>
  − <BasicData Intersectsion="0, 0, 0, 0, " letterNames="Alif Lam Ra Jim Lam " unicode="الرجل" ArabTeX="\<Alrjl>"
    Baseline="513_317_143_309_" nDiacritics="1" nLines="1" Nletters="5" NPAWs="3">
      <letter0 name="Alif" nDiacriticParts="0" DiacriticID="-1" startx_y_endx_y="674_11_652_324"
      boundMinx_y_Maxx_y="645_3_674_344"/>
      <letter1 name="Lam" nDiacriticParts="0" DiacriticID="-1" startx_y_endx_y="596_85_513_317"
      boundMinx_y_Maxx_y="513_71_596_317"/>
      <letter2 name="Ra" nDiacriticParts="0" DiacriticID="-1" startx_y_endx_y="513_317_307_449"
      boundMinx_y_Maxx_y="307_293_513_449"/>
      <letter3 name="Jim" nDiacriticParts="1" DiacriticID="0" startx_y_endx_y="208_266_143_309"
      boundMinx_y_Maxx_y="143_249_370_371"/>
      <diacritic0 boundMinx_y_Maxx_y="241_368_245_371"/>
      <letter4 name="Lam" nDiacriticParts="0" DiacriticID="-1" startx_y_endx_y="148_309_3_316"
      boundMinx_y_Maxx_y="3_190_148_380"/>
      <paw0 letterIDs="0_"/>
      <paw1 letterIDs="1_2_"/>
      <paw2 letterIDs="3_4_"/>
      <line0 PAWIDs="0_1_2_"/>
    </BasicData>
  − <OnlineData>
      <letter0
      trajectory="674.9_11.52_674._8.875_673.2_6.903_672.6_5.495_672.2_4.547_671.8_3.952_671.4_3.608_671.1_3.463_670.7_3.502_670.
      breakPointIndices=""/>
      <letter1
      trajectory="596.4_85.21_596.3_80.3_596.3_76.79_596.3_74.43_596.3_72.97_596.3_72.18_596.2_71.8_596.2_71.74_596.2_72.02_596.1_
      breakPointIndices=""/>
      <letter2
      trajectory="513.5_317.6_513.4_317.7_510.8_317.9_508.4_318.1_506._318.3_503.6_318.6_501.2_318.8_498.7_319._496.2_319.2_493.7_
      breakPointIndices=""/>
      <letter3
      trajectory="208.9_266.2_209.1_263._209.9_260.4_211._258.2_212.4_256.4_214._255._215.8_253.7_217.7_252.6_219.8_251.7_221.9_2
      breakPointIndices="107_266.2"/>
      <letter4
      trajectory="148._309.5_147.4_309.5_145._308.2_142.5_307._140._305.8_137.5_304.8_135._303.8_132.5_302.8_130.1_301.8_127.6_30
      breakPointIndices=""/>
    </OnlineData>
</root>
```


Abbildung A.1. Beispiel für eine Probe der IESK-arDB$_{\text{SynWords}}$ und der zugehörigen XML-Datei, welche die Grundwahrheit und zusätzlich die Trajektorien (online Daten) aller Buchstaben enthält.



(a)Simple, globale Merkmale (b)Momente

(c)Gradient - F74 (d)Simple + F74 + Momente

Abbildung A.2. Konfusionsmatrizen für verschiedene Merkmalsvektoren.

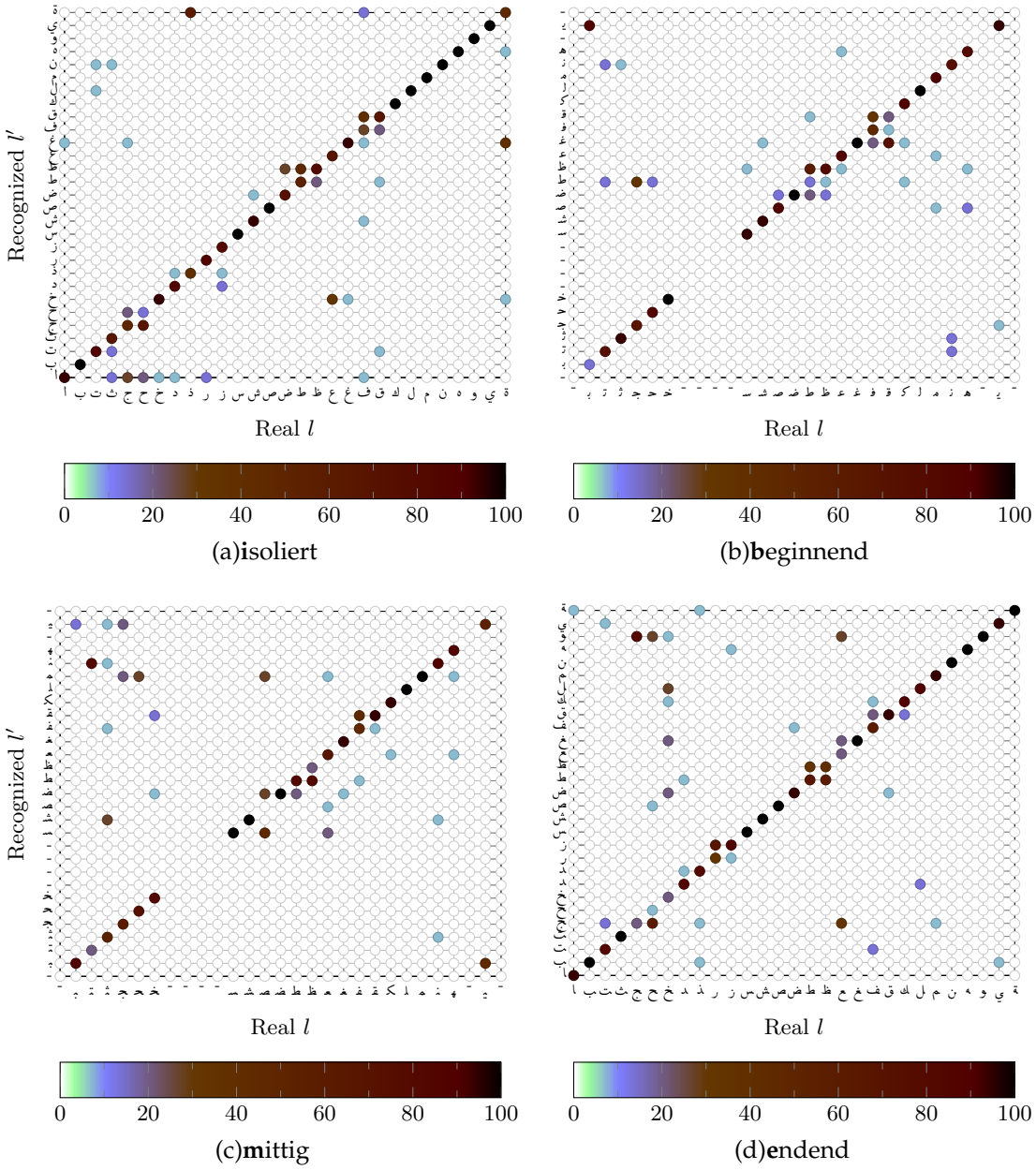


Abbildung A.3. Konfusionsmatrizen für verschiedene Buchstabenpositionen.

A.2 Online-Buchstabenproben – Auszug der verwendeten Datenbank

Im folgendem wird ein exemplarischer Anteil der verwendeten Online-Buchstabendatenbank dargestellt. Dabei handelt es sich um die aus den Online-Proben (Trajektorien) berechneten normierten Offline-Proben (ausgedünnte Bilder), die zum Trainieren der SVMs verwendet wurden. Die Online-Informationen werden lediglich für die Erstellung akkurater ASMs benötigt, und in den Grundwahrheiten gespeichert, um etwaige zukünftige Arbeiten im Bereich Online-Erkennung innerhalb aber auch außerhalb der Arbeitsgruppe zu ermöglichen.

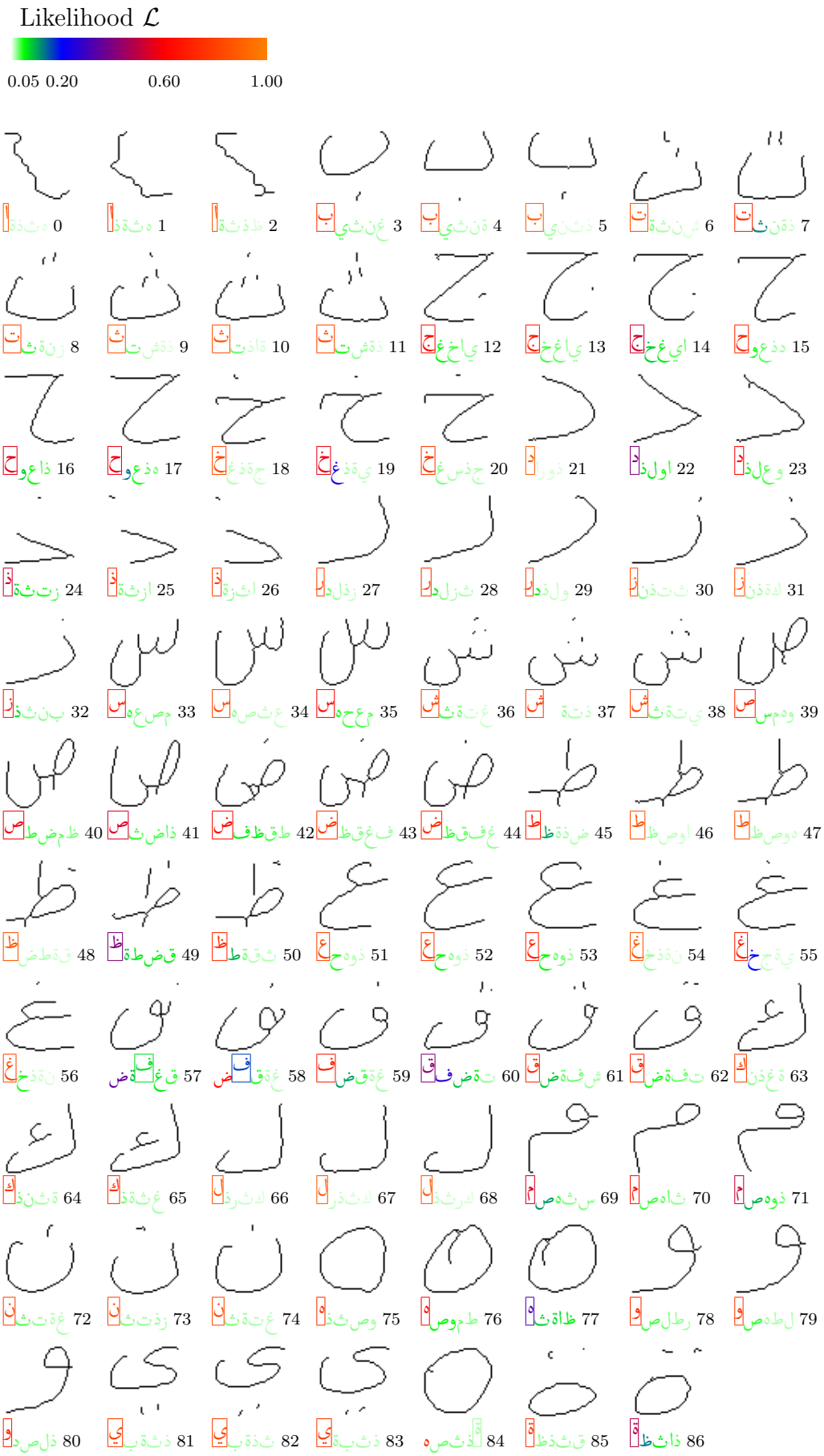


Abbildung A.4. Isolierte Position, KF-simpel. Beispiele für die SVM basierten Rankings. Dargestellt sind die besten 5 der insgesamt 29 möglichen Buchstabenklassen.

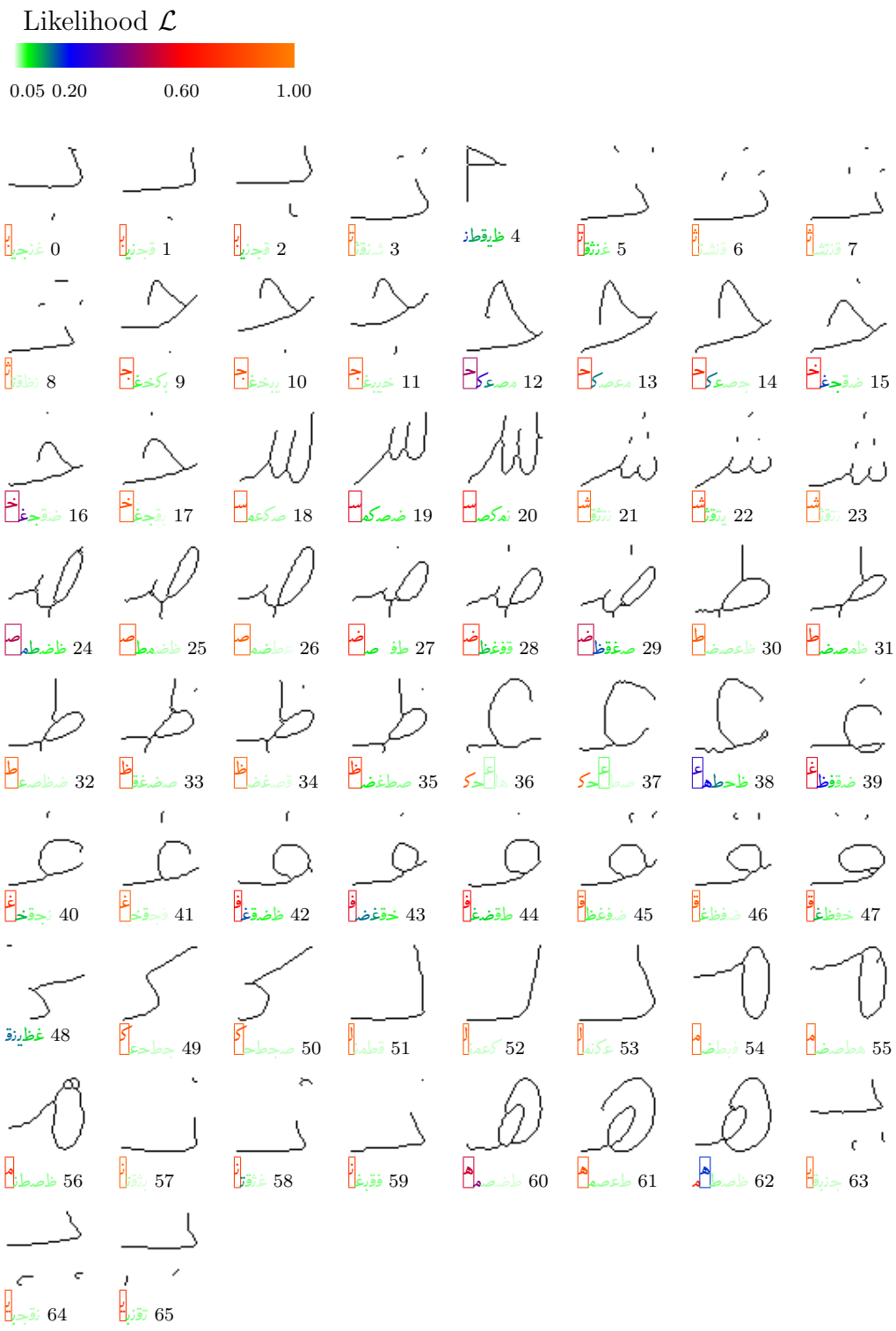


Abbildung A.5. Beginnende Position b, KF-simpel.

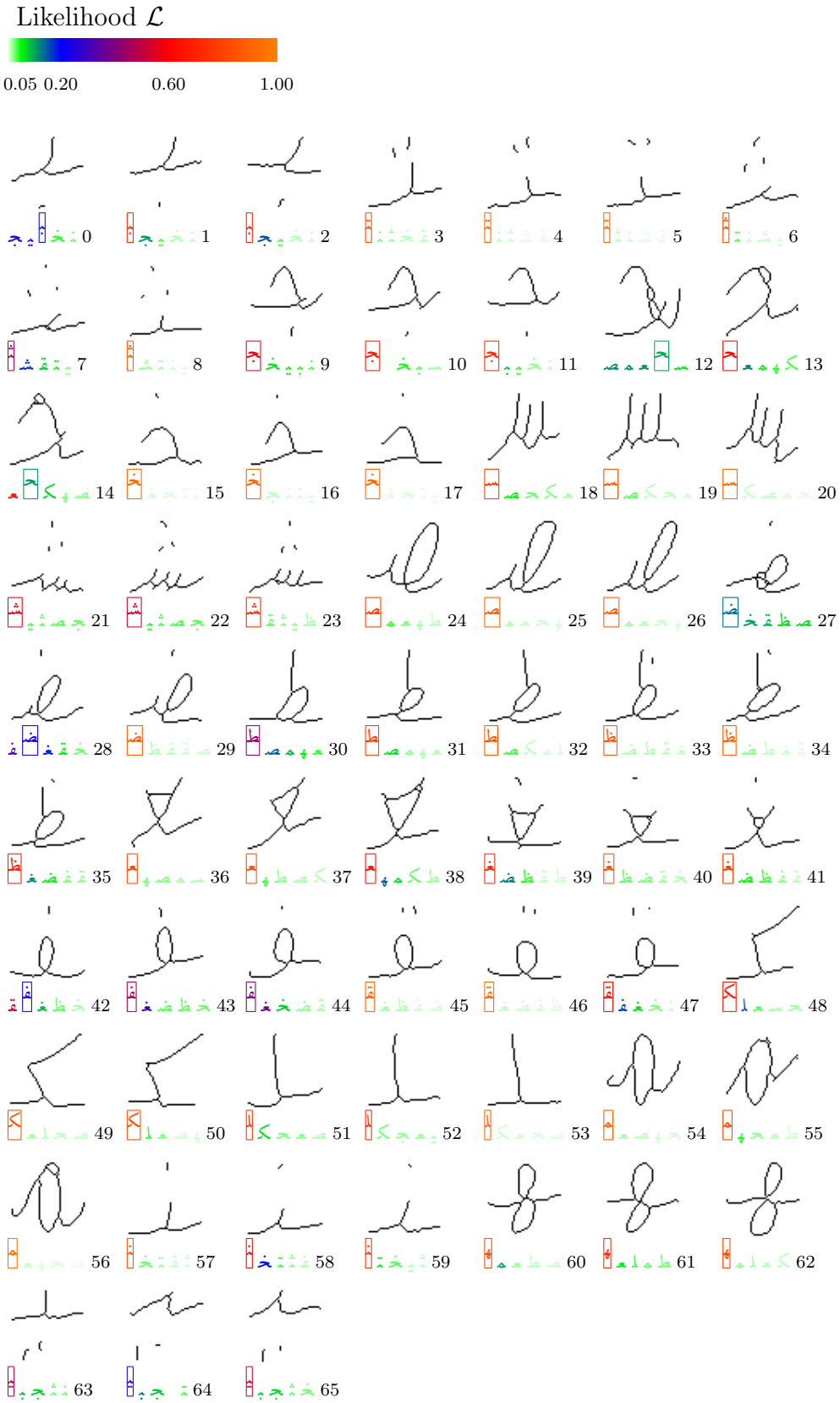


Abbildung A.6. Mittlere Form, KF-simpel.

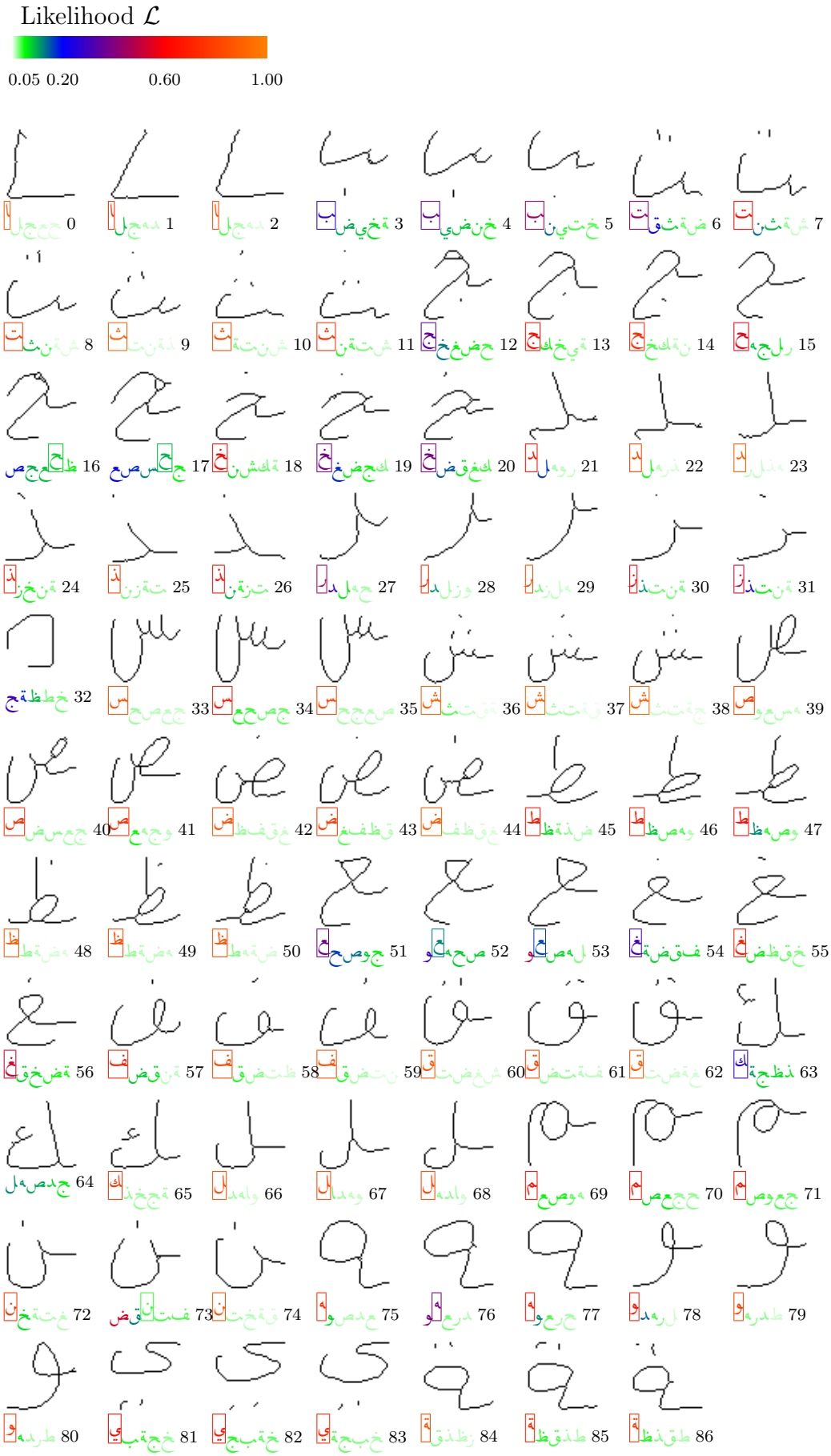


Abbildung A.7. Endende Position, KF-simpel.

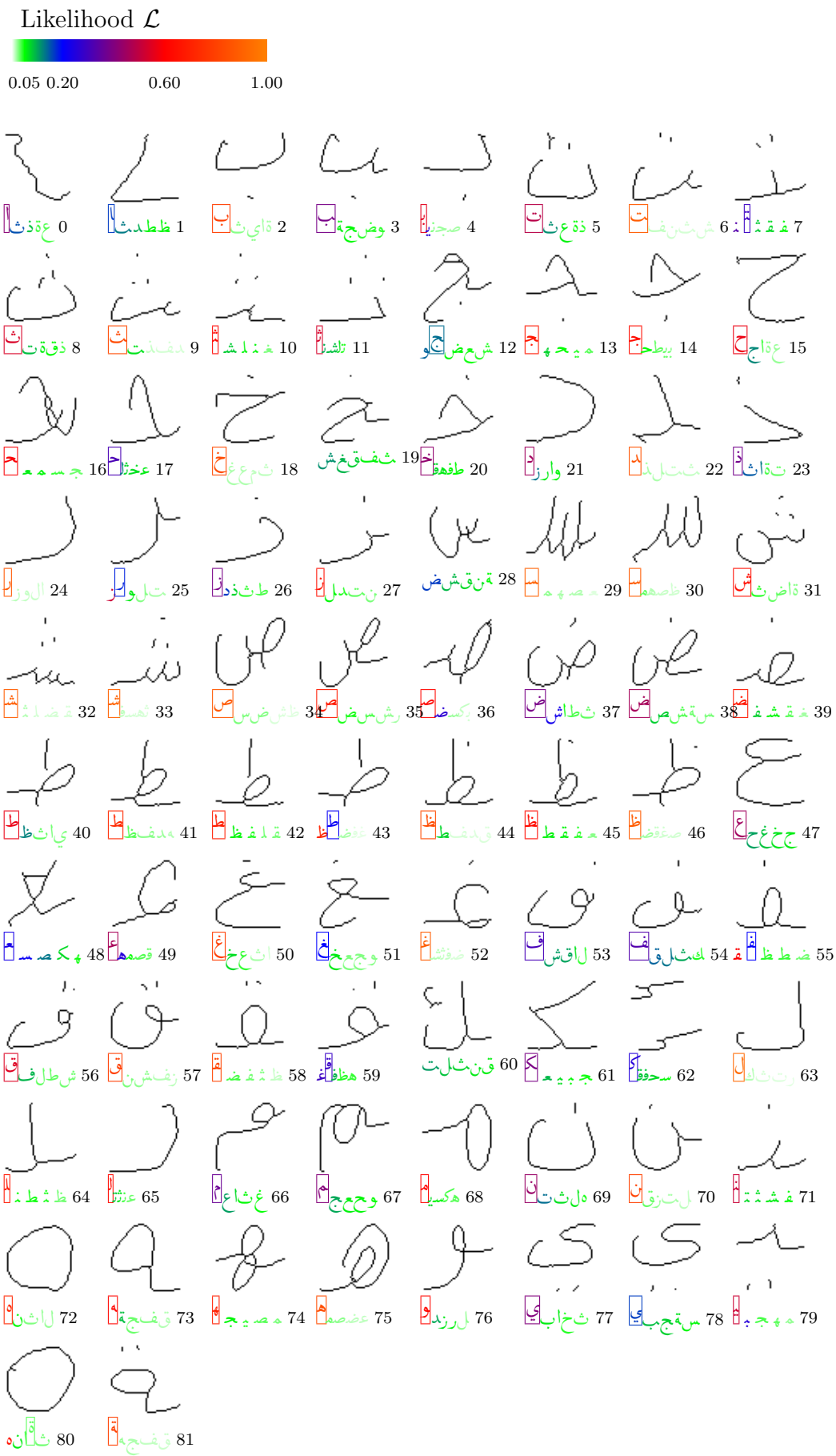


Abbildung A.8. Gemischte Positionen, Gabor Merkmale.

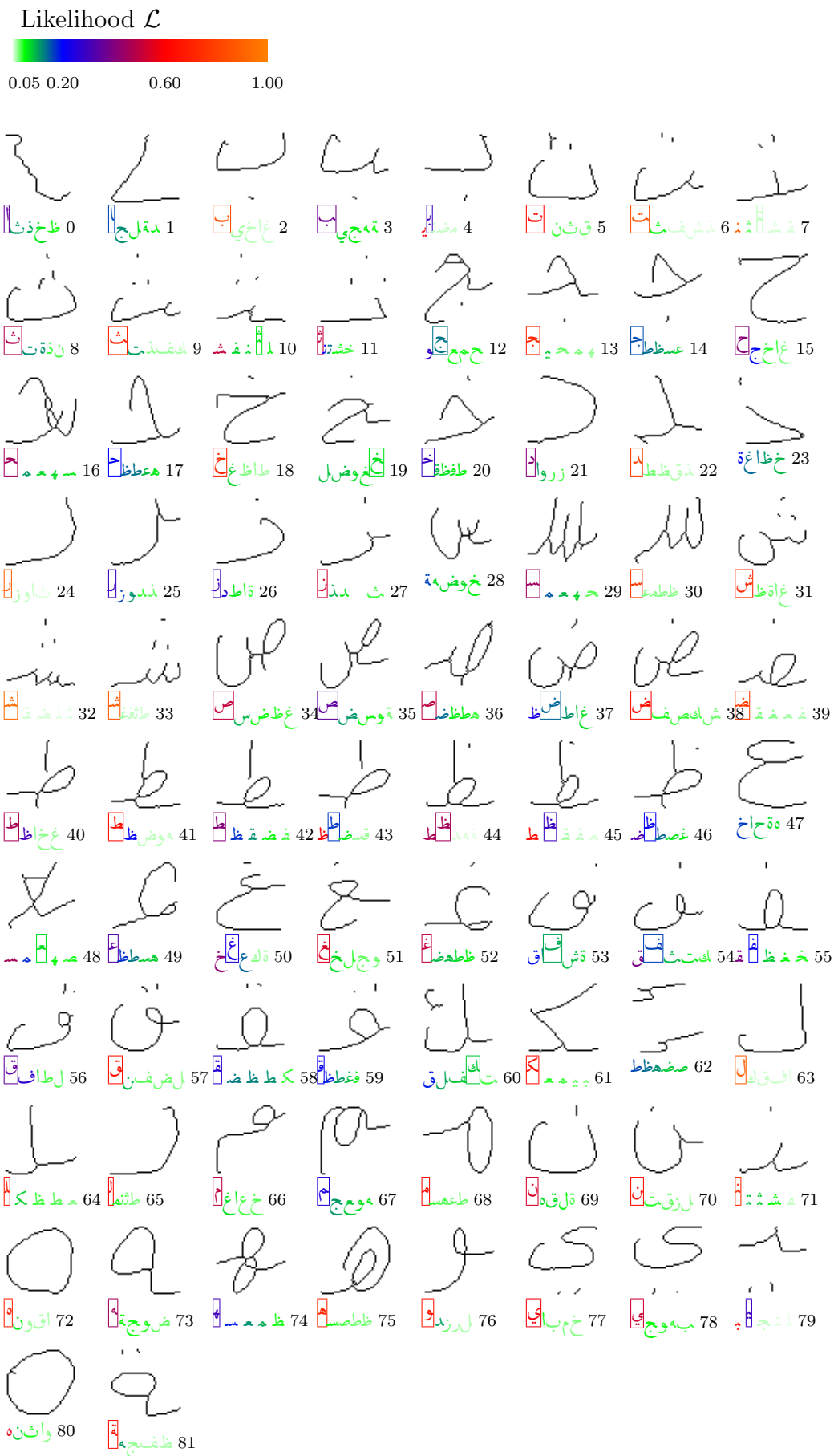


Abbildung A.9. Gemischte Positionen, Momente.

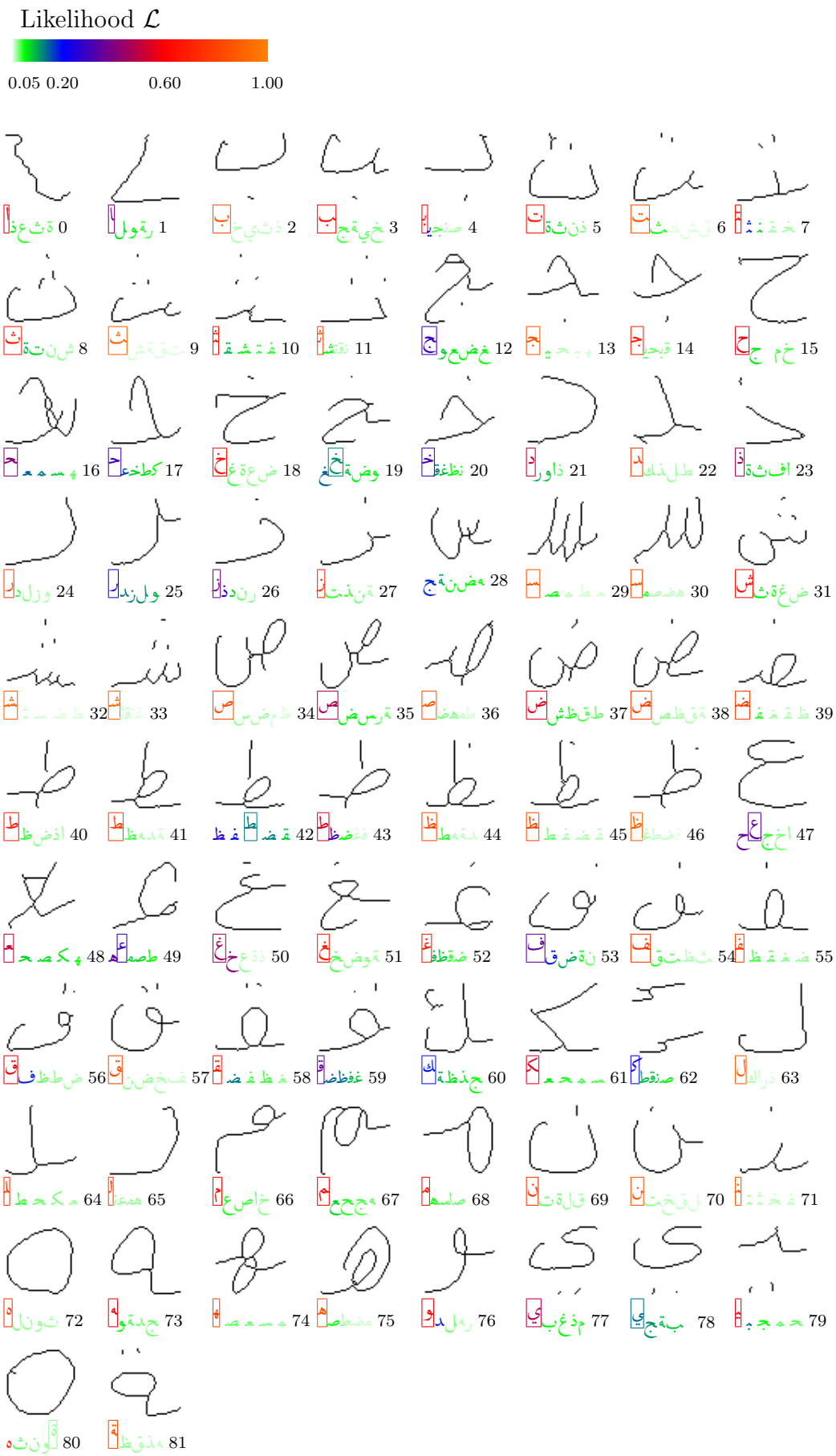


Abbildung A.10. Gemischte Positionen, F74 + KF-simpel.

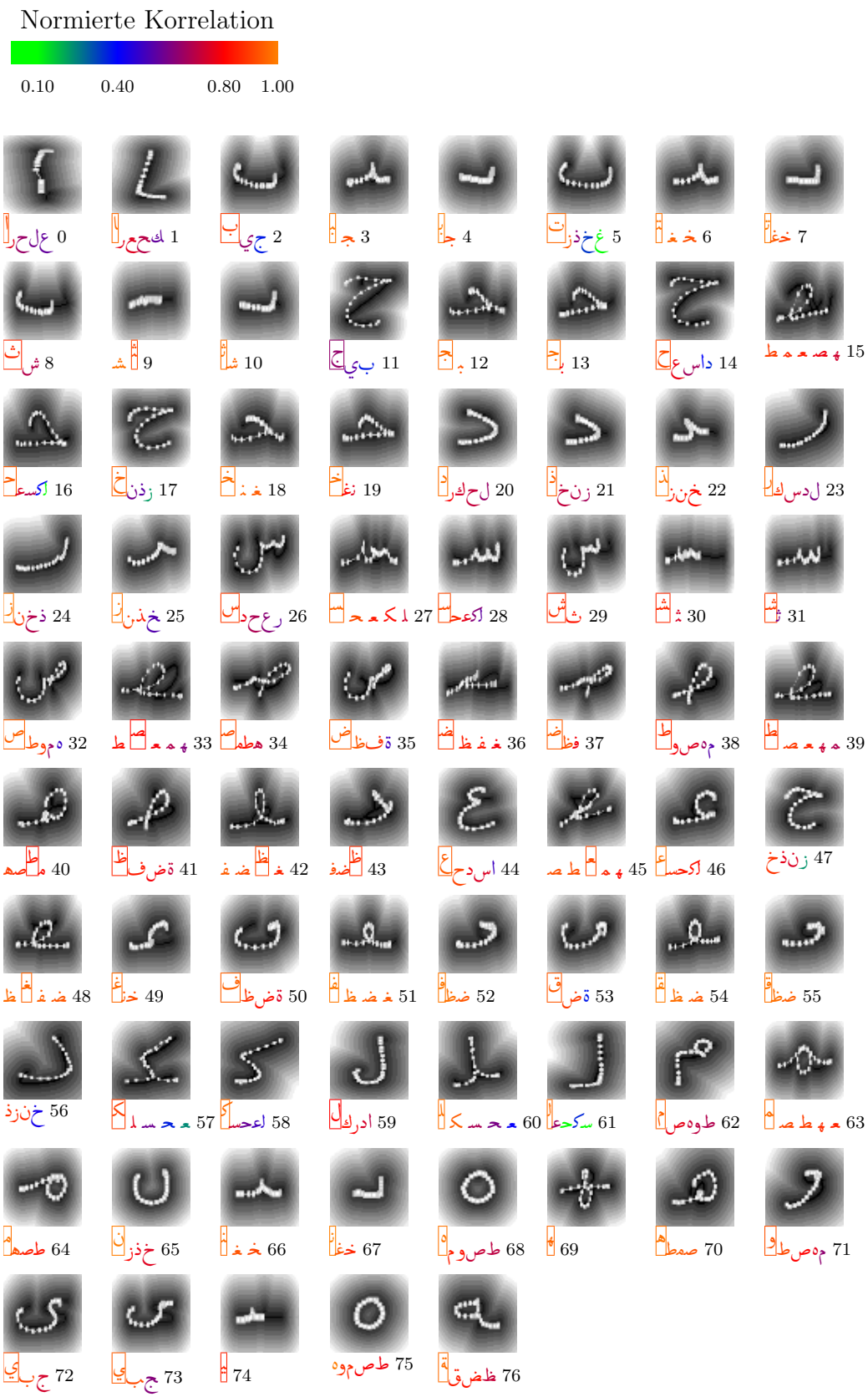


Abbildung A.11. Beispiele für die ASM basierten Rankings.

A.3 Offline-Buchstabenproben – Auszug der verwendeten Datenbank

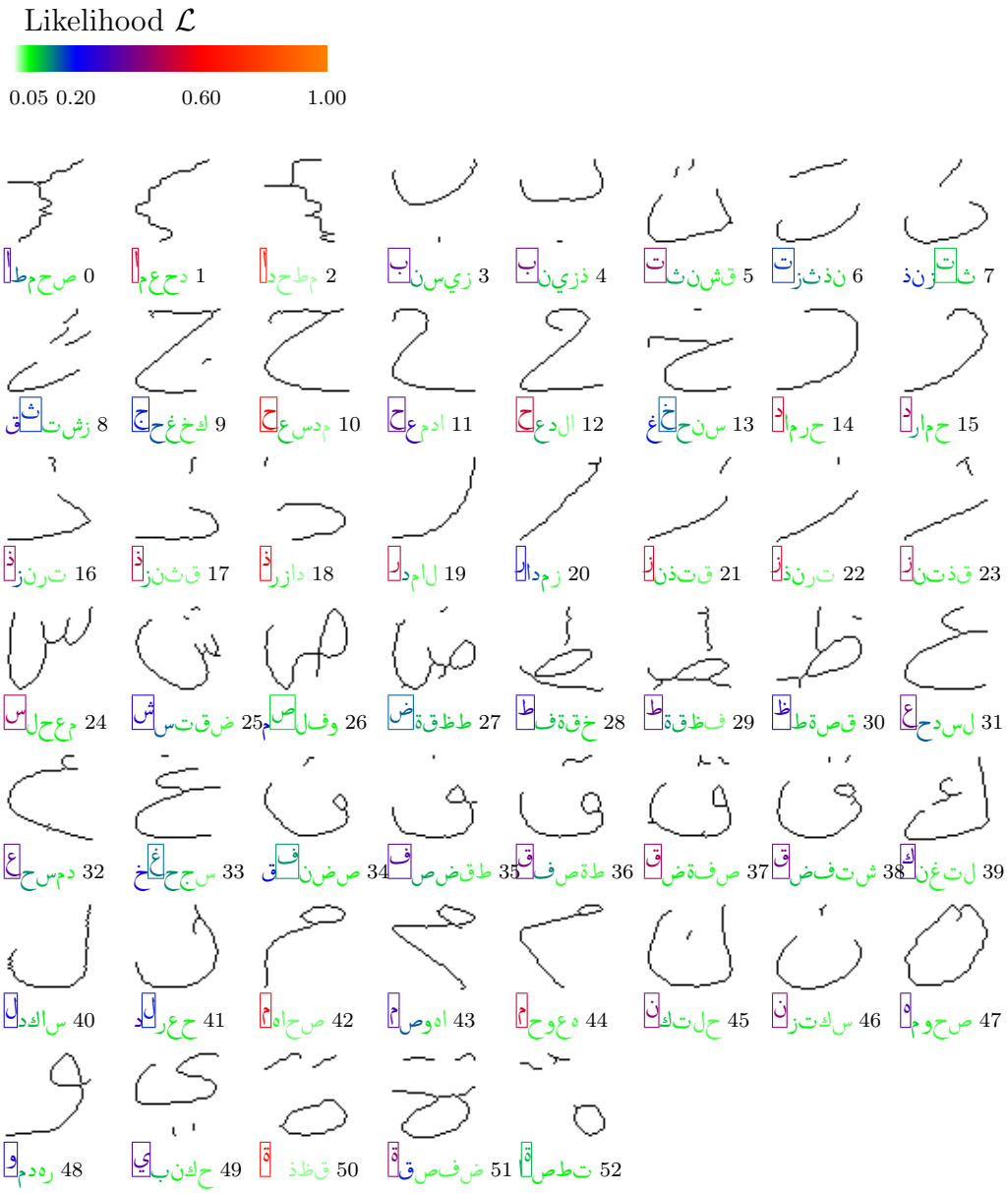


Abbildung A.12. Isolierte Form, KF-simpel. Beispiele für die SVM basierten Rankings. Dargestellt sind die besten 5 der insgesamt 29 möglichen Buchstabenklassen.

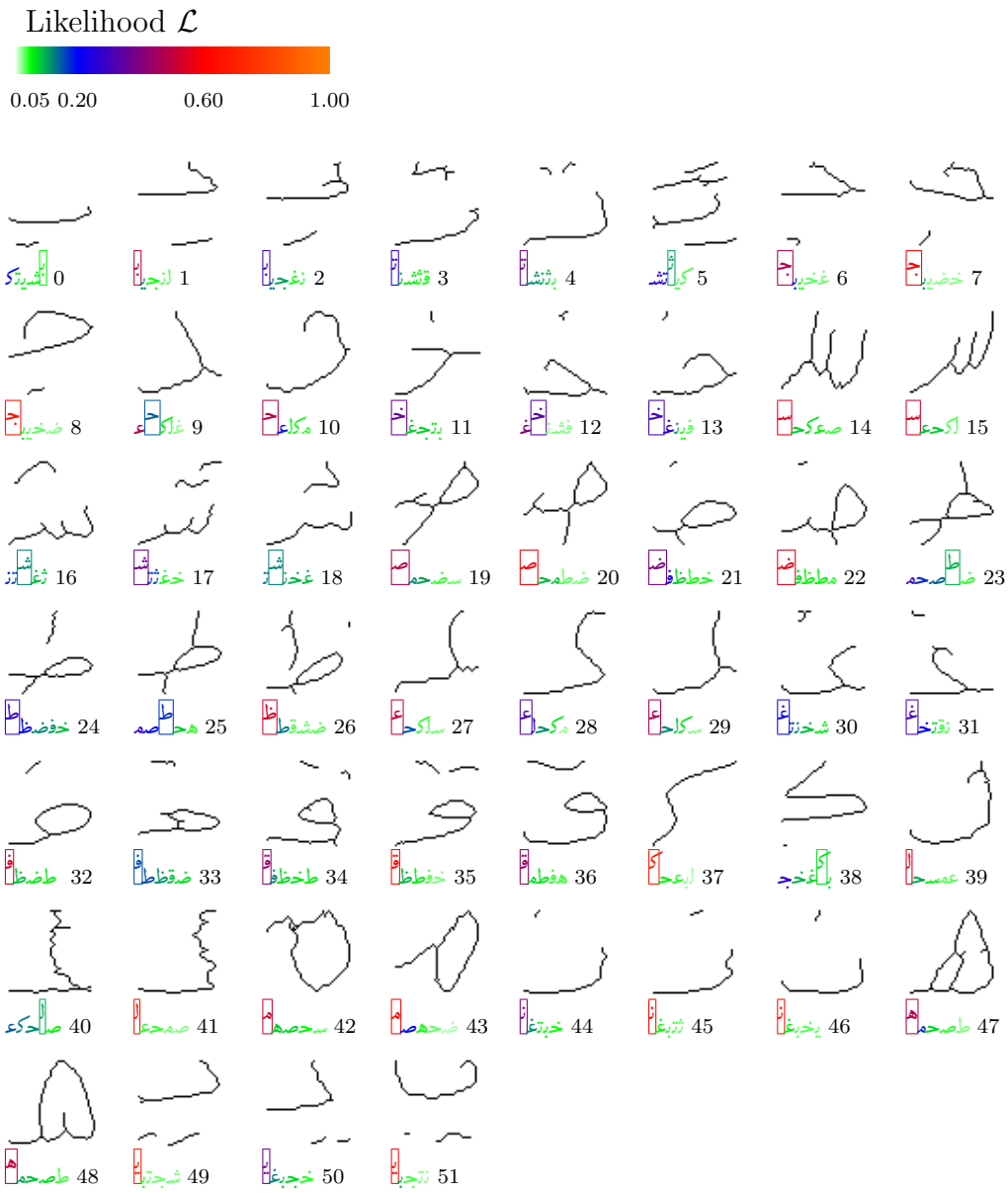


Abbildung A.13. Beginnende Form b, KF-simpel.

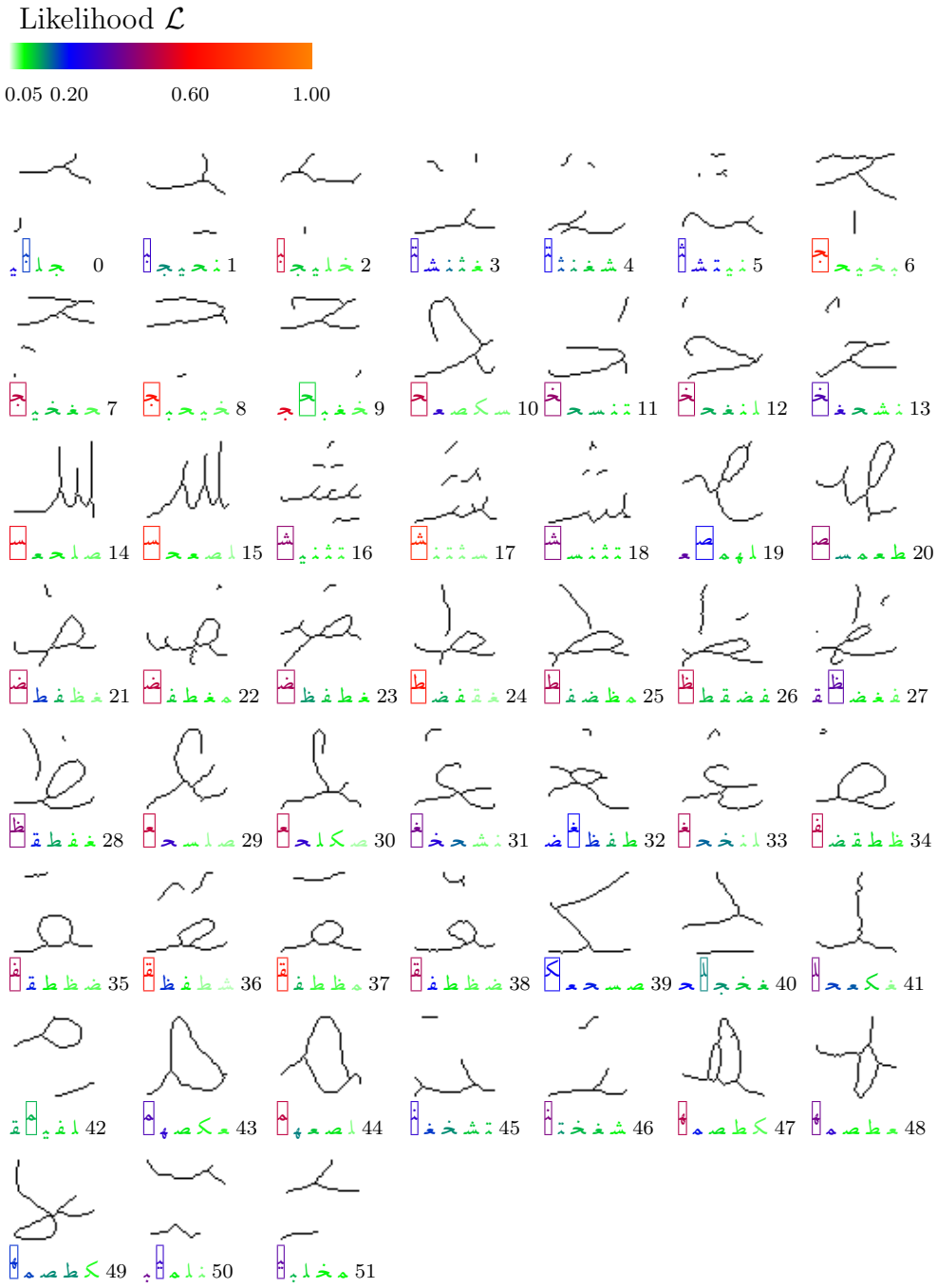


Abbildung A.14. Mittlere Form, KF-simpel.

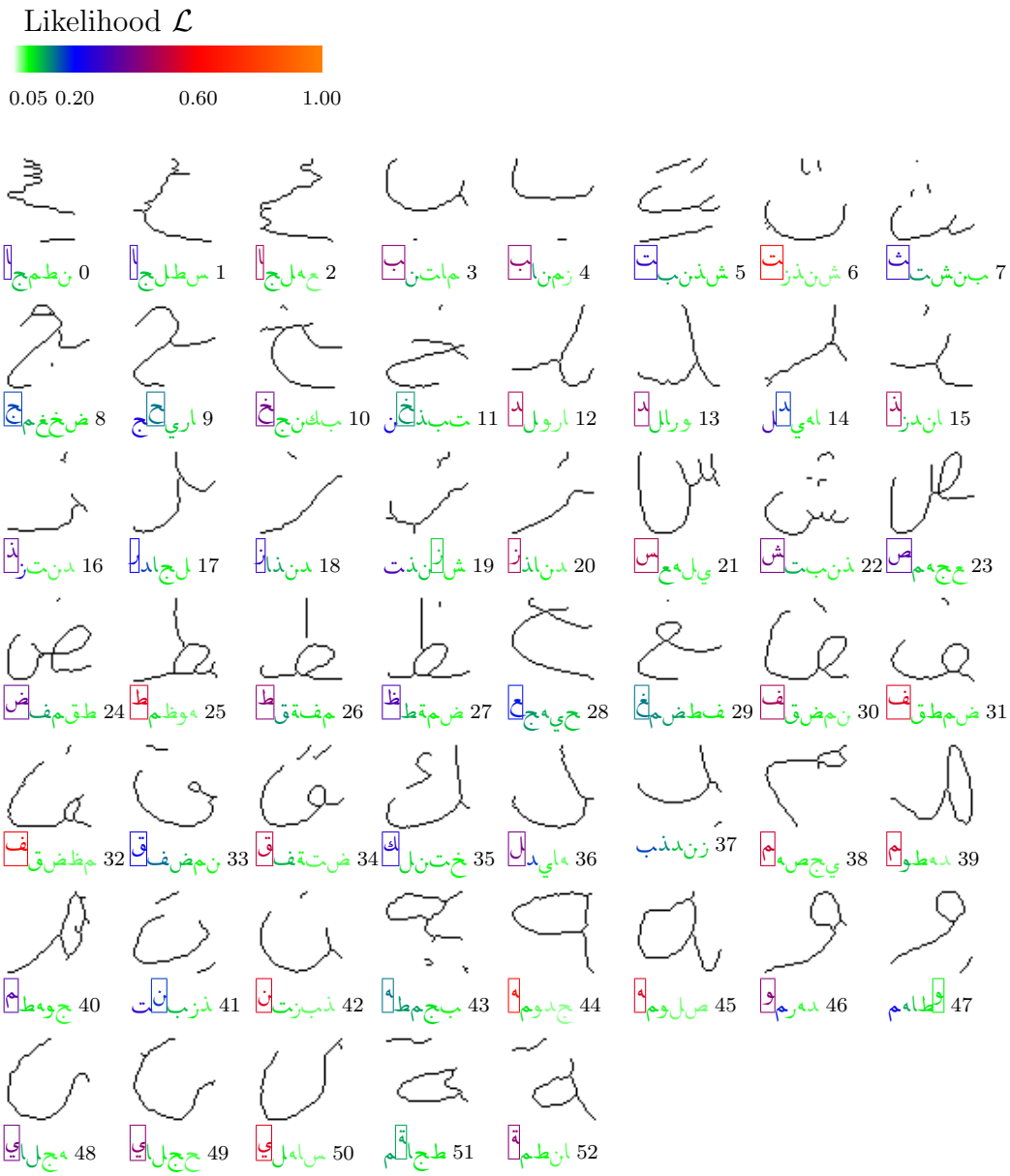


Abbildung A.15. Endende Form, KF-simpel.

A.4 Mögliche Paare arabischer Buchstaben

Die folgenden Tabellen zeigen alle möglichen Kombinationen von arabischen Buchstabenpaaren auf (Bigramme). Dabei wird im Folgenden zwischen den Positionen die innerhalb eins PAW auftreten können unterschieden: **b**eginnend (initial), **m**ittig und **e**ndend (die **i**solierte Position entfällt hier). Es ergeben sich die Kombinationen **be**, **bm**, **mm** und **me**. Alle in arabischer Schrift auftretende Kombinationen für Trigramme sind **bei**, **ibe**, **bme**, **eii**, **mei**, **bmm**, **mme**, **iii**, **meb**, **ebe**, **beb**, **ibm**, **iib**, **mmm**, **ebm** und **eib**. Sie sollen hier jedoch nicht näher behandelt werden, da sie neben der Fehlererkennung nur wenig Relevanz haben.

Da Buchstaben immer paarweise segmentiert werden, geben die Tabellen vor allem eine Übersicht über die verschiedenen Fälle, die beim Segmentierungs-Prozesss beachtet werden müssen. Weiterhin ist die zu erwartende Häufigkeit der Kombinationen farblich gekennzeichnet, da einige Kombinationen in der arabischen Sprache nicht vorkommen oder sehr selten sind, was jeweils auch stark von der Position der Buchstaben abhängen mag. Im allgemeinen Fall (die Position wird nicht beachtet) treten 82% aller Bigramme in arabischer Schrift auf, unter Beachtung der Positionen sind es nur 10–40%. Die angegebene Wahrscheinlichkeit $P(C_2|C_1) \cdot P(C_1)$ setzt sich aus der bedingten Wahrscheinlichkeit $P(C_2|C_1)$ für das Bigram und der einfachen Wahrscheinlichkeit $P(C_1)$ für das Unigram zusammen. Dabei ist C_1 jene Klasse zu welcher der erste Buchstabe und C_2 die Klasse zu welcher der zweite (linke) Buchstaben gehört.

Mögliche Buchstabenkombinationen. Der erste Buchstabe befindet sich in **b**-, der zweite in **e**-Position.
- Kombination nicht möglich
Rote Paare sind sehr häufig ($P(C_2|C_1) \cdot P(C_1) > 0.01$)
Grüne Paare sind sehr selten ($P(C_2|C_1) \cdot P(C_1) < 0.00015$).
Graue Paare treten nicht in der Arabischen Sprache auf ($P(C_2|C_1) \cdot P(C_1) = 0$).

	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ي
ا	-	با	تا	ثا	جا	حا	خا	-	-	-	-	سا	شا	صا	ضا	طا	ظا	عا	غا	فا	قا	كا	لا	ما	نا	ها	-	يا
ب	-	بّ	تب	ثب	جب	حب	خب	-	-	-	-	سب	شب	صب	ضب	طب	ظب	عب	غب	فب	قب	كب	لب	مب	نب	هب	-	يب
ت	-	بت	تّ	ثت	جت	حت	خت	-	-	-	-	ست	شت	صت	ضت	طت	ظت	عت	غت	فت	قت	كت	لت	مت	نت	هت	-	يت
ث	-	بث	تث	ثّ	جث	حث	خث	-	-	-	-	سث	شث	صث	ضث	طث	ظث	عث	غث	فث	قث	كث	لث	مث	نث	هث	-	يث
ج	-	بج	تج	ثج	جّ	ججـ	خج	-	-	-	-	سج	شج	صج	ضج	طج	ظج	عج	غج	فج	قج	كج	لج	مج	نج	هج	-	يج
ح	-	بح	تح	ثح	جح	حّ	خح	-	-	-	-	سح	شح	صح	ضح	طح	ظح	عح	غح	فح	قح	كح	لح	مح	نح	هح	-	يح
خ	-	بخ	تخ	ثخ	جخ	حخ	خّ	-	-	-	-	سخ	شخ	صخ	ضخ	طخ	ظخ	عخ	غخ	فخ	قخ	كخ	لخ	مخ	نخ	هخ	-	يخ
د	-	بد	تد	ثد	جد	حد	خد	-	-	-	-	سد	شد	صد	ضد	طد	ظد	عد	غد	فد	قد	كد	لد	مد	ند	هد	-	يد
ذ	-	بذ	تذ	ثذ	جذ	حذ	خذ	-	-	-	-	سذ	شذ	صذ	ضذ	طذ	ظذ	عذ	غذ	فذ	قذ	كذ	لذ	مذ	نذ	هذ	-	يذ
ر	-	بر	تر	ثر	جر	حر	خر	-	-	-	-	سر	شر	صر	ضر	طر	ظر	عر	غر	فر	قر	كر	لر	مر	نر	هر	-	ير
ز	-	بز	تز	ثز	جز	حز	خز	-	-	-	-	سز	شز	صز	ضز	طز	ظز	عز	غز	فز	قز	كز	لز	مز	نز	هز	-	يز
س	-	بس	تس	ثس	جس	حس	خس	-	-	-	-	سّ	شس	صس	ضس	طس	ظس	عس	غس	فس	قس	كس	لس	مس	نس	هس	-	يس
ش	-	بش	تش	ثش	جش	حش	خش	-	-	-	-	سش	شّ	صش	ضش	طش	ظش	عش	غش	فش	قش	كش	لش	مش	نش	هش	-	يش
ص	-	بص	تص	ثص	جص	حص	خص	-	-	-	-	سص	شص	صّ	ضص	طص	ظص	عص	غص	فص	قص	كص	لص	مص	نص	هص	-	يص
ض	-	بض	تض	ثض	جض	حض	خض	-	-	-	-	سض	شض	صض	ضّ	طض	ظض	عض	غض	فض	قض	كض	لض	مض	نض	هض	-	يض
ط	-	بط	تط	ثط	جط	حط	خط	-	-	-	-	سط	شط	صط	ضط	طّ	ظط	عط	غط	فط	قط	كط	لط	مط	نط	هط	-	يط
ظ	-	بظ	تظ	ثظ	جظ	حظ	خظ	-	-	-	-	سظ	شظ	صظ	ضظ	طظ	ظّ	عظ	غظ	فظ	قظ	كظ	لظ	مظ	نظ	هظ	-	يظ
ع	-	بع	تع	ثع	جع	حع	خع	-	-	-	-	سع	شع	صع	ضع	طع	ظع	»	غع	فع	قع	كع	لع	مع	نع	هع	-	يع
غ	-	بغ	تغ	ثغ	جغ	حغ	خغ	-	-	-	-	سغ	شغ	صغ	ضغ	طغ	ظغ	عغ	غّ	فغ	قغ	كغ	لغ	مغ	نغ	هغ	-	يغ
ف	-	بف	تف	ثف	جف	حف	خف	-	-	-	-	سف	شف	صف	ضف	طف	ظف	عف	غف	فّ	قف	كف	لف	مف	نف	هف	-	يف
ق	-	بق	تق	ثق	جق	حق	خق	-	-	-	-	سق	شق	صق	ضق	طق	ظق	عق	غق	فق	قّ	كق	لق	مق	نق	هق	-	يق
ك	-	بك	تك	ثك	جك	حك	خك	-	-	-	-	سك	شك	صك	ضك	طك	ظك	عك	غك	فك	قك	كّ	لك	مك	نك	هك	-	يك
ل	-	بل	تل	ثل	جل	حل	خل	-	-	-	-	سل	شل	صل	ضل	طل	ظل	عل	غل	فل	قل	كل	لّ	مل	نل	هل	-	يل
م	-	بم	تم	ثم	جم	حم	خم	-	-	-	-	سم	شم	صم	ضم	طم	ظم	عم	غم	فم	قم	كم	لم	مّ	نم	هم	-	يم
ن	-	بن	تن	ثن	جن	حن	خن	-	-	-	-	سن	شن	صن	ضن	طن	ظن	عن	غن	فن	قن	كن	لن	من	نّ	هن	-	ين
ه	-	به	ته	ثه	جه	حه	خه	-	-	-	-	سه	شه	صه	ضه	طه	ظه	عه	غه	فه	قه	كه	له	مه	نه	هّ	-	يه
و	-	بو	تو	ثو	جو	حو	خو	-	-	-	-	سو	شو	صو	ضو	طو	ظو	عو	غو	فو	قو	كو	لو	مو	نو	هو	-	يو
ي	-	بي	تي	ثي	جي	حي	خي	-	-	-	-	سي	شي	صي	ضي	طي	ظي	عي	غي	في	قي	كي	لي	مي	ني	هي	-	يّ

Mögliche Buchstabenkombinationen. Der erste Buchstabe befindet sich in **b**-, der zweite in **m**-Position.
- Kombination nicht möglich
Rote Paare sind sehr häufig ($P(C_2|C_1) \cdot P(C_1) > 0.01$)
Grüne Paare sind sehr selten ($P(C_2|C_1) \cdot P(C_1) < 0.00015$).
Graue Paare treten nicht in der Arabischen Sprache auf ($P(C_2|C_1) \cdot P(C_1) = 0$).

	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ي
ا	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ب	-	بّ	تب	ثب	جب	حب	خب	-	-	-	-	سب	شب	صب	ضب	طب	ظب	عب	غب	فب	قب	كب	لب	مب	نب	هب	-	يب
ت	-	بت	تّ	ثت	جت	حت	خت	-	-	-	-	ست	شت	صت	ضت	طت	ظت	عت	غت	فت	قت	كت	لت	مت	نت	هت	-	يت
ث	-	بث	تث	ثّ	جث	حث	خث	-	-	-	-	سث	شث	صث	ضث	طث	ظث	عث	غث	فث	قث	كث	لث	مث	نث	هث	-	يث
ج	-	بج	تج	ثج	جّ	حج	خج	-	-	-	-	سج	شج	صج	ضج	طج	ظج	عج	غج	فج	قج	كج	لج	مج	نج	هج	-	يج
ح	-	بح	تح	ثح	جح	حّ	خح	-	-	-	-	سح	شح	صح	ضح	طح	ظح	عح	غح	فح	قح	كح	لح	مح	نح	هح	-	يح
خ	-	بخ	تخ	ثخ	جخ	حخ	خّ	-	-	-	-	سخ	شخ	صخ	ضخ	طخ	ظخ	عخ	غخ	فخ	قخ	كخ	لخ	مخ	نخ	هخ	-	يخ
د	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ذ	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ر	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ز	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
س	-	بس	تس	ثس	جس	حس	خس	-	-	-	-	سّ	شس	صس	ضس	طس	ظس	عس	غس	فس	قس	كس	لس	مس	نس	هس	-	يس
ش	-	بش	تش	ثش	جش	حش	خش	-	-	-	-	سش	شّ	صش	ضش	طش	ظش	عش	غش	فش	قش	كش	لش	مش	نش	هش	-	يش
ص	-	بص	تص	ثص	جص	حص	خص	-	-	-	-	سص	شص	صّ	ضص	طص	ظص	عص	غص	فص	قص	كص	لص	مص	نص	هص	-	يص
ض	-	بض	تض	ثض	جض	حض	خض	-	-	-	-	سض	شض	صض	ضّ	طض	ظض	عض	غض	فض	قض	كض	لض	مض	نض	هض	-	يض
ط	-	بط	تط	ثط	جط	حط	خط	-	-	-	-	سط	شط	صط	ضط	طّ	ظط	عط	غط	فط	قط	كط	لط	مط	نط	هط	-	يط
ظ	-	بظ	تظ	ثظ	جظ	حظ	خظ	-	-	-	-	سظ	شظ	صظ	ضظ	طظ	ظّ	عظ	غظ	فظ	قظ	كظ	لظ	مظ	نظ	هظ	-	يظ
ع	-	بع	تع	ثع	جع	حع	خع	-	-	-	-	سع	شع	صع	ضع	طع	ظع	»	غع	فع	قع	كع	لع	مع	نع	هع	-	يع
غ	-	بغ	تغ	ثغ	جغ	حغ	خغ	-	-	-	-	سغ	شغ	صغ	ضغ	طغ	ظغ	عغ	غّ	فغ	قغ	كغ	لغ	مغ	نغ	هغ	-	يغ
ف	-	بف	تف	ثف	جف	حف	خف	-	-	-	-	سف	شف	صف	ضف	طف	ظف	عف	غف	فّ	قف	كف	لف	مف	نف	هف	-	يف
ق	-	بق	تق	ثق	جق	حق	خق	-	-	-	-	سق	شق	صق	ضق	طق	ظق	عق	غق	فق	قّ	كق	لق	مق	نق	هق	-	يق
ك	-	بك	تك	ثك	جك	حك	خك	-	-	-	-	سك	شك	صك	ضك	طك	ظك	عك	غك	فك	قك	كّ	لك	مك	نك	هك	-	يك
ل	-	بل	تل	ثل	جل	حل	خل	-	-	-	-	سل	شل	صل	ضل	طل	ظل	عل	غل	فل	قل	كل	لّ	مل	نل	هل	-	يل
م	-	بم	تم	ثم	جم	حم	خم	-	-	-	-	سم	شم	صم	ضم	طم	ظم	عم	غم	فم	قم	كم	لم	مّ	نم	هم	-	يم
ن	-	بن	تن	ثن	جن	حن	خن	-	-	-	-	سن	شن	صن	ضن	طن	ظن	عن	غن	فن	قن	كن	لن	من	نّ	هن	-	ين
ه	-	به	ته	ثه	جه	حه	خه	-	-	-	-	سه	شه	صه	ضه	طه	ظه	عه	غه	فه	قه	كه	له	مه	نه	هّ	-	يه
و	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ي	-	بي	تي	ثي	جي	حي	خي	-	-	-	-	سي	شي	صي	ضي	طي	ظي	عي	غي	في	قي	كي	لي	مي	ني	هي	-	يّ

Mögliche Buchstabenkombinationen. Der erste Buchstabe befindet sich in **m**-, der zweite in **m**-Position.
- Kombination nicht möglich
Rote Paare sind sehr häufig ($P(C_2|C_1) \cdot P(C_1) > 0.01$)
Grüne Paare sind sehr selten ($P(C_2|C_1) \cdot P(C_1) < 0.00015$).
Graue Paare treten nicht in der Arabischen Sprache auf ($P(C_2|C_1) \cdot P(C_1) = 0$).

	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ي
ا	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ب	-	بّ	تب	ثب	جب	حب	خب	-	-	-	-	سب	شب	صب	ضب	طب	ظب	عب	غب	فب	قب	كب	لب	مب	نب	هب	-	بي
ت	-	بت	تّ	ثت	جت	حت	خت	-	-	-	-	ست	شت	صت	ضت	طت	ظت	عت	غت	فت	قت	كت	لت	مت	نت	هت	-	تي
ث	-	بث	تث	ثّ	جث	حث	خث	-	-	-	-	سث	شث	صث	ضث	طث	ظث	عث	غث	فث	قث	كث	لث	مث	نث	هث	-	ثي
ج	-	بج	تج	ثج	جّ	حج	خج	-	-	-	-	سج	شج	صج	ضج	طج	ظج	عج	غج	فج	قج	كج	لج	مج	نج	هج	-	جي
ح	-	بح	تح	ثح	جح	حّ	خح	-	-	-	-	سح	شح	صح	ضح	طح	ظح	عح	غح	فح	قح	كح	لح	مح	نح	هح	-	حي
خ	-	بخ	تخ	ثخ	جخ	حخ	خّ	-	-	-	-	سخ	شخ	صخ	ضخ	طخ	ظخ	عخ	غخ	فخ	قخ	كخ	لخ	مخ	نخ	هخ	-	خي
د	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ذ	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ر	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ز	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
س	-	بس	تس	ثس	جس	حس	خس	-	-	-	-	سّ	شس	صس	ضس	طس	ظس	عس	غس	فس	قس	كس	لس	مس	نس	هس	-	يس
ش	-	بش	تش	ثش	جش	حش	خش	-	-	-	-	سش	شّ	صش	ضش	طش	ظش	عش	غش	فش	قش	كش	لش	مش	نش	هش	-	يش
ص	-	بص	تص	ثص	جص	حص	خص	-	-	-	-	سص	شص	صّ	ضص	طص	ظص	عص	غص	فص	قص	كص	لص	مص	نص	هص	-	يص
ض	-	بض	تض	ثض	جض	حض	خض	-	-	-	-	سض	شض	صض	ضّ	طض	ظض	عض	غض	فض	قض	كض	لض	مض	نض	هض	-	يض
ط	-	بط	تط	ثط	جط	حط	خط	-	-	-	-	سط	شط	صط	ضط	طّ	ظط	عط	غط	فط	قط	كط	لط	مط	نط	هط	-	يط
ظ	-	بظ	تظ	ثظ	جظ	حظ	خظ	-	-	-	-	سظ	شظ	صظ	ضظ	طظ	ظّ	عظ	غظ	فظ	قظ	كظ	لظ	مظ	نظ	هظ	-	يظ
ع	-	بع	تع	ثع	جع	حع	خع	-	-	-	-	سع	شع	صع	ضع	طع	ظع	عّ	غع	فع	قع	كع	لع	مع	نع	هع	-	يع
غ	-	بغ	تغ	ثغ	جغ	حغ	خغ	-	-	-	-	سغ	شغ	صغ	ضغ	طغ	ظغ	عغ	غّ	فغ	قغ	كغ	لغ	مغ	نغ	هغ	-	يغ
ف	-	بف	تف	ثف	جف	حف	خف	-	-	-	-	سف	شف	صف	ضف	طف	ظف	عف	غف	فّ	قف	كف	لف	مف	نف	هف	-	يف
ق	-	بق	تق	ثق	جق	حق	خق	-	-	-	-	سق	شق	صق	ضق	طق	ظق	عق	غق	فق	قّ	كق	لق	مق	نق	هق	-	يق
ك	-	بك	تك	ثك	جك	حك	خك	-	-	-	-	سك	شك	صك	ضك	طك	ظك	عك	غك	فك	قك	كّ	لك	مك	نك	هك	-	يك
ل	-	بل	تل	ثل	جل	حل	خل	-	-	-	-	سل	شل	صل	ضل	طل	ظل	عل	غل	فل	قل	كل	لّ	مل	نل	هل	-	يل
م	-	بم	تم	ثم	جم	حم	خم	-	-	-	-	سم	شم	صم	ضم	طم	ظم	عم	غم	فم	قم	كم	لم	مّ	نم	هم	-	يم
ن	-	بن	تن	ثن	جن	حن	خن	-	-	-	-	سن	شن	صن	ضن	طن	ظن	عن	غن	فن	قن	كن	لن	من	نّ	هن	-	ين
ه	-	به	ته	ثه	جه	حه	خه	-	-	-	-	سه	شه	صه	ضه	طه	ظه	عه	غه	فه	قه	كه	له	مه	نه	هّ	-	يه
و	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ي	-	بي	تي	ثي	جي	حي	خي	-	-	-	-	سي	شي	صي	ضي	طي	ظي	عي	غي	في	قي	كي	لي	مي	ني	هي	-	يّ

Mögliche Buchstabenkombinationen. Der erste Buchstabe befindet sich in **m**-, der zweite in **e**-Position.
- Kombination nicht möglich
Rote Paare sind sehr häufig ($P(C_2|C_1) \cdot P(C_1) > 0.01$)
Grüne Paare sind sehr selten ($P(C_2|C_1) \cdot P(C_1) < 0.00015$).
Graue Paare treten nicht in der Arabischen Sprache auf ($P(C_2|C_1) \cdot P(C_1) = 0$).

	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ي
ا	-	با	تا	ثا	جا	حا	خا	-	-	-	-	سا	شا	صا	ضا	طا	ظا	عا	غا	فا	قا	كا	لا	ما	نا	ها	وا	يا
ب	-	بّ	تب	ثب	جب	حب	خب	-	-	-	-	سب	شب	صب	ضب	طب	ظب	عب	غب	فب	قب	كب	لب	مب	نب	هب	وب	يب
ت	-	بت	تّ	ثت	جت	حت	خت	-	-	-	-	ست	شت	صت	ضت	طت	ظت	عت	غت	فت	قت	كت	لت	مت	نت	هت	وت	يت
ث	-	بث	تث	ثّ	جث	حث	خث	-	-	-	-	سث	شث	صث	ضث	طث	ظث	عث	غث	فث	قث	كث	لث	مث	نث	هث	وث	يث
ج	-	بج	تج	ثج	جّ	حج	خج	-	-	-	-	سج	شج	صج	ضج	طج	ظج	عج	غج	فج	قج	كج	لج	مج	نج	هج	وج	يج
ح	-	بح	تح	ثح	جح	حّ	خح	-	-	-	-	سح	شح	صح	ضح	طح	ظح	عح	غح	فح	قح	كح	لح	محّ	نح	هح	وح	يح
خ	-	بخ	تخ	ثخ	جخ	حخ	خّ	-	-	-	-	سخ	شخ	صخ	ضخ	طخ	ظخ	عخ	غخ	فخ	قخ	كخ	لخ	مخ	نخ	هخ	وخ	يخ
د	-	بد	تد	ثد	جد	حد	خد	-	-	-	-	سد	شد	صد	ضد	طد	ظد	عد	غد	فد	قد	كد	لد	مد	ند	هد	ود	يد
ذ	-	بذ	تذ	ثذ	جذ	حذ	خذ	-	-	-	-	سذ	شذ	صذ	ضذ	طذ	ظذ	عذ	غذ	فذ	قذ	كذ	لذ	مذ	نذ	هذ	وذ	يذ
ر	-	بر	تر	ثر	جر	حر	خر	-	-	-	-	سر	شر	صر	ضر	طر	ظر	عر	غر	فر	قر	كر	لر	مر	نر	هر	ور	ير
ز	-	بز	تز	ثز	جز	حز	خز	-	-	-	-	سز	شز	صز	ضز	طز	ظز	عز	غز	فز	قز	كز	لز	مز	نز	هز	وز	يز
س	-	بس	تس	ثس	جس	حس	خس	-	-	-	-	سّ	شس	صس	ضس	طس	ظس	عس	غس	فس	قس	كس	لس	مس	نس	هس	وس	يس
ش	-	بش	تش	ثش	جش	حش	خش	-	-	-	-	سش	شّ	صش	ضش	طش	ظش	عش	غش	فش	قش	كش	لش	مش	نش	هش	وش	يش
ص	-	بص	تص	ثص	جص	حص	خص	-	-	-	-	سص	شص	صّ	ضص	طص	ظص	عص	غص	فص	قص	كص	لص	مص	نص	هص	وص	يص
ض	-	بض	تض	ثض	جض	حض	خض	-	-	-	-	سض	شض	صض	ضّ	طض	ظض	عض	غض	فض	قض	كض	لض	مض	نض	هض	وض	يض
ط	-	بط	تط	ثط	جط	حط	خط	-	-	-	-	سط	شط	صط	ضط	طّ	ظط	عط	غط	فط	قط	كط	لط	مط	نط	هط	وط	يط
ظ	-	بظ	تظ	ثظ	جظ	حظ	خظ	-	-	-	-	سظ	شظ	صظ	ضظ	طظ	ظّ	عظ	غظ	فظ	قظ	كظ	لظ	مظ	نظ	هظ	وظ	يظ
ع	-	بع	تع	ثع	جع	حع	خع	-	-	-	-	سع	شع	صع	ضع	طع	ظع	عّ	غع	فع	قع	كع	لع	مع	نع	هع	وع	يع
غ	-	بغ	تغ	ثغ	جغ	حغ	خغ	-	-	-	-	سغ	شغ	صغ	ضغ	طغ	ظغ	عغ	غّ	فغ	قغ	كغ	لغ	مغ	نغ	هغ	وغ	يغ
ف	-	بف	تف	ثف	جف	حف	خف	-	-	-	-	سف	شف	صف	ضف	طف	ظف	عف	غف	فّ	قف	كف	لف	مف	نف	هف	وف	يف
ق	-	بق	تق	ثق	جق	حق	خق	-	-	-	-	سق	شق	صق	ضق	طق	ظق	عق	غق	فق	قّ	كق	لق	مق	نق	هق	وق	يق
ك	-	بك	تك	ثك	جك	حك	خك	-	-	-	-	سك	شك	صك	ضك	طك	ظك	عك	غك	فك	قك	كّ	لك	مك	نك	هك	وك	يك
ل	-	بل	تل	ثل	جل	حل	خل	-	-	-	-	سل	شل	صل	ضل	طل	ظل	عل	غل	فل	قل	كل	لّ	مل	نل	هل	ول	يل
م	-	بم	تم	ثم	جم	حم	خم	-	-	-	-	سم	شم	صم	ضم	طم	ظم	عم	غم	فم	قم	كم	لم	مّ	نم	هم	وم	يم
ن	-	بن	تن	ثن	جن	حن	خن	-	-	-	-	سن	شن	صن	ضن	طن	ظن	عن	غن	فن	قن	كن	لن	من	نّ	هن	ون	ين
ه	-	به	ته	ثه	جه	حه	خه	-	-	-	-	سه	شه	صه	ضه	طه	ظه	عه	غه	فه	قه	كه	له	مه	نه	هّ	وه	يه
و	-	بو	تو	ثو	جو	حو	خو	-	-	-	-	سو	شو	صو	ضو	طو	ظو	عو	غو	فو	قو	كو	لو	مو	نو	هو	وّ	يو
ي	-	بي	تي	ثي	جي	حي	خي	-	-	-	-	سي	شي	صي	ضي	طي	ظي	عي	غي	في	قي	كي	لي	مي	ني	هي	وي	يّ

Literaturverzeichnis

[1] http://dlib.net (besucht am 20.11.2018).

[2] https://github.com/zacharydenton/zaum/blob/master/public/data/wordfreq/ar/ar_50k.txt (besucht am 20.11.2018).

[3] ABANDAH, G. A., JAMOUR, F. T., AND QARALLEH, E. A. Recognizing handwritten arabic words using grapheme segmentation and recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR) 17*, 3 (2014), 275–291.

[4] ABDI, H., AND WILLIAMS, L. J. Principal Component Analysis, JohnWiley & Sons, Inc.WIREs CompStat 433-459 , 2010.

[5] AGHBARI, Z. A., AND BROOK, S. Hah manuscripts: A holistic paradigm for classifying and retrieving historical arabic handwritten documents. *Expert Syst. Appl. 36* (2009), 10942–10951.

[6] AL-JAWFI, R. Handwriting arabic character recognition lenet using neural network. *Int. Arab J. Inf. Technol. 6*, 3 (2009), 304–309.

[7] AL-KHOURY, I. *Arabic Text Recognition and Machine Translation*. PhD thesis, Universitat Politècnica de València, 2015.

[8] AL-MA'ADEED, S., ELLIMAN, D., AND HIGGINS, C. A data base for arabic handwritten text recognition research. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on* (2002), pp. 485–489.

[9] AL-MUHTASEB, H., MAHMOUD, S., AND QAHWAJI, R. Recognition of off-line printed arabic text using hidden markov models. *Signal Processing 88* (2008), 2902–2912.

[10] AL-OHALI, Y., CHERIET, M., AND SUEN, C. Databases for recognition of handwritten arabic cheques. *Pattern Recognition 36*, 1 (2003), 111 – 121.

[11] AL-TAANI, A., AND AL-HAJ, S. Recognition of on-line arabic handwritten characters using structural features. *JOURNAL OF PATTERN RECOGNITION RESEARCH 1* (2010), 23–37.

[12] AL-ZUBI, S. *Active Shape Structural Model*. PhD thesis, Otto-von-Guericke University of Magdeburg, 2004.

[13] ALJUAID, H., MUHAMMAD, Z., AND SARFRAZ, M. A tool to develop arabic handwriting recognition system using genetic approach. *Journal of Computer Science 6* (2010), 619–624.

[14] ALKHATEEB, J. H., JIANG, J., REN, J., KHELIFI, F., AND IPSON, S. Multiclass classification of unconstrained handwritten arabic words. *The Open Signal Processing Journal 2* (2009), 21–28.

[15] ANGSHUL, M. Bangla basic character recognition using digital curvelet transform. *Journal of Pattern Recognition Research 2*, 1 (2008), 17–26.

[16] ARICA, N., AND YARMAN-VURAL, F. Optical character recognition for cursive handwriting. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 6 (2002), 801–813.

[17] ATICI, A., AND YARMAN-VURAL, F. A heuristic algorithm for optical character recognition of arabic script;. *Signal Processing 62* (1997), 87–99.

[18] A.UL-HASAN, AHMED, S., RASHID, F., SHAFAIT, F., AND BREUEL, T. Printed urdu nastaleeq script recognition with bidirectional lstm networks. In *International Conference on Document Analysis and Recognition* (2013).

[19] BAGDANOV, A., AND KANAI, J. Projection profile based skew estimation algorithm for jbig compressed images. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on* (Aug 1997), vol. 1, pp. 401–405 vol.1.

[20] BLUMENSTEIN, M. *Cursive Character Segmentation Using Neural Network Techniques.*, vol. 90 of *Studies in Computational Intelligence*. Springer, 2008, pp. 259–275.

[21] CAMASTRA, F. A svm-based cursive character recognizer. *Pattern Recogn. 40* (2007), 3721–3727.

[22] CHARFI, M., BOUSSELLAA, W., AND ALIMI, M. A. An intelligent system for digitization and processing of historical arabic documents. *Int. J. Comput. Proc. Oriental Lang. 20*, 4 (2007), 259–288.

[23] CHERIET, M., KHARMA, N., LIU, C.-L., AND SUEN, C. *Character Recognition Systems: A Guide for Students and Practitioners*. Wiley-Interscience, 2007.

[24] CHEUNG, K.-W., MEMBER, S., CHIN, R. T., YEUNG, D.-Y., AND CHIN, T. A bayesian framework for deformable pattern recognition with application to handwritten character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20* (1998), 1382–1388.

[25] COHEN, W. W., RAVIKUMAR, P., AND FIENBERG, S. E. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation* (2003).

[26] COOTES, T. Active shape models. In *In Proc. British Machine Vision Conf* (1992), In Proc. British Machine Vision Conf, Springer-Verlag, pp. 266–275.

[27] COOTES, T. *An Introduction to Active Shape Models*. Image Processing and Analysis, 2000, ch. 7, pp. 223–248.

[28] DING, X., AND LIU, H. Segmentation-driven offline handwritten chinese and arabic script recognition. In *Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition* (Berlin, Heidelberg,, 2008), SACH'06, Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition, Springer-Verlag, pp. 196–217.

[29] DINGES, L., AL-HAMADI, A., AND ELZOBI, M. An approach for arabic handwriting synthesis based on active shape models. In *International Conference on Document Analysis and Recognition (ICDAR)* (2013), pp. 1292–1296.

[30] DINGES, L., AL-HAMADI, A., ELZOBI, M., AND EL-ETRIBY, S. Synthesis of common arabic handwritings to aid optical character recognition research. *Sensors 16*, 3 (2016), 346.

[31] DINGES, L., ELZOBI, M., AL-HAMADI, A., AND AGHBARI, Z. A. Synthizing handwritten arabic text using active shape models. In *Image Processing and Communications Challenges 3* (Berlin, Heidelberg, 2011), R. S. Choraś, Ed., Springer Berlin Heidelberg, pp. 401–408.

[32] DUDA, R. O., AND HART, P. E. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM 15*, 1 (Jan. 1972), 11–15.

[33] ELARIAN, Y., ABDEL-AAL, R., AHMAD, I., PARVEZ, M. T., AND ZIDOURI, A. B. C. Handwriting synthesis: classifications and techniques. *IJDAR 17*, 4 (2014), 455–469.

[34] ELARIAN, Y., AL-MUHSATEB, H. A., AND GHOUTI, L. M. Arabic handwriting synthesis. In *First International Workshop on Frontiers in Arabic Handwriting Recognition (http://hdl.handle.net/2003/27562)* (2011), First International Workshop on Frontiers in Arabic Handwriting Recognition (http://hdl.handle.net/2003/27562).

[35] ELMEZAIN, M., AL-HAMADI, A., APPENRODT, J., AND MICHAELIS, B. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on* (Dec 2008), pp. 1–4.

[36] ELZOBI, M. *Unconstrained recognition of offline Arabic handwriting using Generative and discriminative classification models*. PhD thesis, Otto-von-Guericke-University Magdeburg, 2017.

[37] ELZOBI, M., AL-HAMADI, A., AL AGHBARI, Z., AND LASLO, D. Iesk-ardb: a database for handwritten arabic and optimized topological segmentation approach. In *International Journal on Document Analysis and Recognition*. Springer, 2012.

[38] ELZOBI, M., AL-HAMADI, A., DINGES, L., AND MICHAELIS, B. A structural features based segmentation for off-line handwritten arabic text. In *I/V Communications and Mobile Network (ISVC), 2010 5th International Symposium on* (Sept 2010), pp. 1–4.

[39] FELDBACH, M. *Segmentierung und strukturbasierte adaptive Erkennung von Gebrauchsschrift in historischen Dokumenten*. 2006.

[40] FERNANDES, L. A., AND OLIVEIRA, M. M. Real-time line detection through an improved hough transform voting scheme. *Pattern Recognition 41*, 1 (2008), 299 – 314.

[41] GRAVES, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. PhD thesis, Technische Universität München, 2008.

[42] GUYON, I. Handwriting synthesis from handwritten glyphs. In *In Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition* (1996), In Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition, pp. 309–312.

[43] HÖHER, P. A. *Grundlagen der digitalen Informationsübertragung (2. Auflage, Kapitel 10)*. Springer Vieweg, 2013.

[44] HTTP//HTTP://KHATT.IDEAS2SERVE.NET/ (BESUCHT AM 27.09.2017).

[45] HU, J., ROSENTHAL, A. S., AND BROWN, M. K. Combining high-level features with sequential local features for on-line handwriting recognition. In *Image Analysis and Processing, 9th International Conference, ICIAP '97, Florence, Italy, September 17-19, 1997, Proceedings, Volume II* (1997), pp. 647–654.

[46] JAIN, A., AND ZONGKER, D. Representation and recognition of handwritten digits using deformable templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*, 12 (1997), 1386–1390.

[47] JAIN, A. K., W., R. P., AND MAO, J. Statistical pattern recognition: A review. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE 22*, 1 (2000), 4–37.

[48] KAMARAINEN, J.-K. Gabor features in image analysis. In *Int. Conf. on Image Processing Theory, Tools and Applications (IPTA2012)* (Istanbul, Turkey, 2012).

[49] KAVALLIERATOU, E., FAKOTAKIS, N., AND KOKKINAKIS, G. K. An unconstrained handwriting recognition system. *IJDAR 4*, 4 (2002), 226–242.

[50] KIRANYAZ, S., INCE, T., AND GABBOUJ, M. *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*, 1st ed. Springer Publishing Company, Incorporated, 2015.

[51] KLETTE, R., AND ZAMPERONI, P. *Handbuch der Operatoren für die Bildbearbeitung: Bildtransformationen für die digitale Bildverarbeitung*. Vieweg, 1992.

[52] KRUMKE, S. O., AND NOLTEMEIER, H. Graphentheoretische konzepte und algorithmen, 2009.

[53] KUMAR, J., ABD-ALMAGEED, W., KANG, L., AND DOERMANN, D. Handwritten arabic text line segmentation using affinity propagation. In *IAPR* (2010), pp. 135–142.

[54] LAVRENKO, V., RATH, T. M., AND MANMATHA, R. Holistic word recognition for handwritten historical documents. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)* (2004), DIAL '04, Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04), IEEE Computer Society, pp. 278–.

[55] LI, Y., ZHENG, Y., DOERMANN, D., JAEGER, S., AND LI, Y. Script-independent text line segmentation in freestyle handwritten documents. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 8 (Aug. 2008), 1313–1329.

[56] LIKFORMAN-SULEM, L., MOJAMMED, R. A. H., MOKBEL, C., MENERASI, F., BIANNE-BERNARD, A.-L., AND KERMORVANT, C. *Features for HMM-Based*

Arabic Handwritten Word Recognition Systems. Springer, London, UK,, 2012, ch. Chapter 6, pp. 215–254. ISBN 978-1-4471-4071-9.

[57] LIKFORMAN-SULEM, L., ZAHOUR, A., AND TACONET, B. Text line segmentation of historical documents: a survey. *IJDAR 9*, 2-4 (2007), 25.

[58] LIM, H.-Y., KIM, J.-H., AND KANG, D.-S. Development of paper automation recognition system using brain modeling of hippocampal neural network algorithm. In *The 23rd International Technical Conference on Circuit/Systems Computers and Communications* (2008), The 23rd International Technical Conference on Circuit/Systems Computers and Communications, pp. 1241–1244.

[59] LORIGO, L., AND GOVINDARAJU, V. Segmentation and pre-recognition of arabic handwriting. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)* (Seoul, Korea,,, 2005), I. C. Society, Ed., Eighth International Conference on Document Analysis and Recognition (ICDAR'05), pp. 605–609.

[60] MAEGNER, V., AND ABED, H. E. *Databases and competitions: strategies to improve Arabic recognition systems*. Springer-Verlag, Collage Park, MD, USA,, 2008, pp. 82–103.

[61] MAHMOUD, S. A., AHMAD, I., ALSHAYEB, M., AL-KHATIB, W. G., PARVEZ, M. T., FINK, G. A., MÄRGNER, V., AND ABED, H. E. Khatt: Arabic offline handwritten text database. In *ICFHR* (2012), pp. 449–454.

[62] MÄRGNER, V., AND ABED, H. E. *Databases and competitions: strategies to improve Arabic recognition systems*. Springer-Verlag, Collage Park, MD, USA,,, 2008, pp. 82–103.

[63] MÄRGNER, V., AND PECHWITZ, M. Synthetic data for arabic ocr system development. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on* (2001), Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on, pp. 1159–1163.

[64] MARIO, P. Automatische erkennung handgeschriebener arabischer wörter. Tech. rep., Dissertation, Institut für Kommunikationstechnik, Technische Universität Braunschweig,, 2004.

[65] MARSAGLIA, G., AND BRAY, T. A. A convenient method for generating normal variables. *SIAM Review, Vol. 6, No. 3 p.260-264* (1964).

[66] MATUSZEWSKI, B. J., QUAN, W., AND SHARK, L.-K. High-resolution comprehensive 3-d dynamic database for facial articulation analysis. In *in 1st Workshop on Benchmarking Facial Image Analysis Technologies, ICCV'2011, Barcelona* (2011), pp. 2128–2135.

[67] MIYAO, H., AND MARUYAMA, M. Virtual example synthesis based on pca for off-line handwritten character recognition. In *Document Analysis Systems* (2006), Document Analysis Systems, pp. 96–105.

[68] MOHAMAD, R. A.-H., LIKFORMAN-SULEM, L., AND MOKBEL, C. Combining slanted-frame classifiers for improved hmm-based arabic handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31* (2009), 1165–1177.

[69] NAWAZ, T., NAQVI, S., REHMAN, H., AND FAIZ, A. Optical character recognition system for urdu (naskh font) using pattern matching technique. *International Journal of Image Processing 3*, 3 (2008), 92–104.

[70] NG, A. Y., AND JORDAN, M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 841–848.

[71] NG, G., SHI, D., GUNN, S., AND DAMPER, R. Nonlinear active handwriting models and their applications to handwritten chinese radical recognition. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on* (2003), Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on, pp. 534 – 538 vol.1.

[72] NIBLACK, W. *An Introduction to Digital Image Processing*. Strandberg Publishing Company, Birkeroed, Denmark, Denmark, 1985.

[73] NICOLAOU, A., AND GATOS, B. Handwritten text line segmentation by shredding text into its lines. In *ICDAR* (2009), pp. 626–630.

[74] ONAT, A., YILDIZ, F., AND GÜNDÜZ, M. Ottoman script recognition using hidden markov model. *Proceedings of World Acedemy of Science, Engineering and Technology 14* (2006), , 71–73.

[75] OTSU, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics 9*, 1 (Jan. 1979), 62–66.

[76] PARKINS, A. D., AND NANDI, A. K. Simplifying hand written digit recognition using a genetic algorithm, european signal processing conference (eusipco), toulouse, france, 2002.

[77] PESCH, H., HAMDANI, M., FOSTER, J., AND NEY, H. Analysis of preprocessing techniques for latin handwriting recognition, Frontiers in Handwriting Recognition (ICFHR), 2012.

[78] PROKOP, R. J., AND REEVES, A. P. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graph. Models Image Process. 54* (1992), 438–460.

[79] RAO, P. Shape vectors: An efficient parametric representation for the synthesis and recognition of hand script characters, 1993.

[80] RAY, A., CHANDAWALA, A., AND CHAUDHARY, A. Character recognition using conditional random fields based matching engine. In *12th International Conference on Document Analysis and Recognition* (2013).

[81] REHMAN, A., MOHAMED, D., AND SULONG, G. Implicit vs explicit based script segmentation and recognition: A performance comparison on benchmark database. *Int. J. Open Problems Compt. Math 2*, 3 (2009), 352–364.

[82] SAABNI, R. M., AND EL-SANA, J. A. Comprehensive synthetic arabic database for on/off-line script recognition research. Springer-Verlag, 2012.

[83] SAHRHAN, A., AND HELALAT, O. A. Arabic character recognition using artificial neural networks and statistical analysis. *World Academy of Science, Engineering and Technology 27* (2007), 32–36.

[84] SCHLOSSER, S. Erim arabic database. Tech. rep., Document Processing Research Program, Information and Materials Applications Laboratory, Environmental Research Institute of Michigan, 1995.

[85] SHAIK, N., AHMED, Z., AND ALI, G. Segmentation of arabic text into characters for recognition. *IMTIC 20* (2008), 11–18.

[86] SHANBEHZADEH, J., PEZASHKI, H., AND SARRAFZADEH, A. Features extraction from farsi hand written letters. In *Proceedings of Image and Vision Computing* (New Zealand,,, 2007), Proceedings of Image and Vision Computing, pp. 35–40.

[87] SHI, D., GUNN, S., AND DAMPER, R. A radical approach to handwritten chinese character recognition using active handwriting models. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (2001), Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.

[88] SHI, D., GUNN, S. R., AND DAMPER, R. I. Handwritten chinese radical recognition using nonlinear active shape models. *IEEE Trans. Pattern Anal. Mach. Intell. 25*, 2 (2003), 277–280.

[89] STRASSEL, S. M. Linguistic resources for arabic handwriting recognition. In *Proceedings of the second International Conference for Arabic Handwriting Recognition* (2009).

[90] TOENNIES, K. D. *Guide to Medical Image Analysis: Methods and Algorithms*, 2nd ed. Springer Publishing Company, Incorporated, 2017.

[91] VARGA, T., AND BUNKE, H. Generation of synthetic training data for an hmm-based handwriting recognition system. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on* (2003), Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on, pp. 618 – 622 vol.1.

[92] VARGA, T., KILCHHOFER, D., AND BUNKE, H. Template-based synthetic handwriting generation for the training of recognition systems. In *In Proceedings of the 12th Conference of the International Graphonomics Society* (2005), In Proceedings of the 12th Conference of the International Graphonomics Society, pp. 206–211.

[93] WANG, J., CHENYUWU, YING-QINGXU, QING XU, Y., AND YEUNG SHUM, H. Combining shape and physical models for on-line cursive handwriting synthesis, 2003.

[94] WANG, Y. S. M. A comparative evaluation of string similarity metrics for ontology alignment. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE 12*, 3 (2015), 957.

[95] WERNER, P., AL-HAMADI, A., NIESE, R., WALTER, S., GRUSS, S., AND UND H. C. TRAUE. Towards pain monitoring: Facial expression, head pose, a new database, an automatic system and remaining challenges. In *in Proceedings of the British Machine Vision Conference* (2011), pp. 119.1–119.13.

[96] WIENECKE, M. Videobasierte handschrifterkennung. Tech. rep., Dissertation, Technische Fakultät, Universität Bielefeld,, 2003.

[97] XUE, J.-H., AND TITTERINGTON, D. Comment on "on discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes". *Neural Processing Letters 28*, 3 (2008), 169–187.

[98] YALNIZ, I. Z., ALTINGOVDE, I. S., GÜDÜKBAY, U., AND ULUSOY, Ö. Ottoman archives explorer: A retrieval system for digital ottoman archives. *ACM Journal on Computing and Cultural Heritage 2*, 3 (2009), 0–20, Article 8.

[99] ZIDOURI, A. Oran system: A basis for an arabic ocr. *The Arabian Journal for Science and Engineering 31* (2005), 91–100.