Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Institut für Simulation und Graphik

Generierung einer semantischen Repräsentation aus Abbildungen handschriftlicher Kirchenbuchaufzeichnungen

Diplomarbeit

Verfasser: Markus Feldbach
geboren am 27. November 1974 in Magdeburg
Matrikel-Nr. 150 756

Betreuer: Prof. Dr. Klaus D. Toennies

Beginn: 4. Februar 2000

Abgabe: 31. Juli 2000

Zusammenfassung

Um die Information aus Kirchenbüchern und anderen historischen Aufzeichnungen automatisch erfassen zu können, muss die Schrift dieser Dokumente von einem Schrifterkennungsprogramm erkannt werden. Die Toleranzen sind bei dieser Art von Schrift besonders hoch. Diese Arbeit liefert einen Lösungsweg der Schritte, die notwendig sind, um eine Repräsentation des auf dem Dokument befindlichen Textes zu erhalten, die als Grundlage für einen Erkenner dient.

Die Besonderheiten dieser alten Schriften sind zum Einen der sehr freie Verlauf der Textlinien und zum Anderen die Verbindungen zwischen den einzelnen Zeilen. Die hier beschriebenen Algorithmen liefern eine Rekonstruktion des Zeilenverlaufes. Des Weiteren werden die einzelnen Zeilen segmentiert und als Rasterbild ausgegeben.

Abstract

For being able to automatically acquire the information recorded in church chronicles and other historical scriptures, the writing on these documents has to be recognized. The tolerances of this kind of writing are especially high. This thesis provides a solution to this problem. It describes algorithms for transforming the paper documents into a representation of the text that is usable as an input for an automatic text recognizer.

The automatic recognition of old hand-written scriptures is especially difficult for two main reasons. These are the comparatively unrestricted path of the text lines, on the one side, and the presence of connections between the lines on the other. The algorithms described in this thesis provide ways to reconstruct the path of the text lines. In addition, the single lines are segmented and an output in form of a raster image is provided.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich bei der Erstellung dieser Arbeit unterstützten.

Für die hervorragende Betreuung der Diplomarbeit danke ich Herrn Prof. Dr. Klaus D. Toennies, der mich in der sehr wichtigen Zeitplanung unterstützte und des öfteren Denkanstöße gab.

Großer Dank gebührt der Firma Graphikon, insbesondere dem Geschäftsführer Prof. Saedler, der mir den Sourcecode des Vektorisierungsprogrammes VECTORY zur Verfügung stellte. Das gilt auch für Guido Eberhardt, der für die Entwicklung von VECTORY verantwortlich ist und der sich mehrmals die Zeit nahm, um mir Fragen zum Sourcecode zu beantworten.

Ein herzliches Dankeschön an Herrn Mewes und an die Kirchengemeinde Wegenstedt, die es mir ermöglichten, ihre alten Kirchenbücher zu digitalisieren und mir damit den Zugriff auf historische Schriften ermöglichten.

Tobias Isenberg hat ein besonderes Dankeschon verdient, da er sich die große Mühe machte, die Arbeit trotz Zeitnot korrekturzulesen.

Selbständigkeitserklärung

Hiermit versichere ich, Markus Feldbach, die vorliegende Arbeit selbstständig und nur unter der Verwendung der angegebenen Quellen angefertigt zu haben.

Magdeburg, 31. Juli 2000 Markus Feldbach

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 1

1.2 Ziele der Arbeit 2

1.3 Gliederung der Arbeit 3

2 Erkennungssysteme und -verfahren **5**

2.1 Begriffserläuterung 5

2.1.1 Kettencode nach Freeman 5

2.1.2 Hough-Transformation 6

2.2 Erkennung von Druckschrift 7

2.3 Handschrifterkennung 7

2.3.1 Online-Erkennung 7

2.3.2 Offline-Erkennung 8

3 Merkmale der Kirchenbuchaufzeichnungen **11**

3.1 Idealisierter Aufbau einer Schriftseite 11

3.2 Ausnahmefälle 12

4 Vorverarbeitung **15**

4.1 Digitalisierung 15

4.1.1 Flachbettscanner 15

4.1.2 Digitalfotografie 16

4.1.3 Analogfotografie & Durchlichtscanner 16

4.2 Binarisierung 17

4.2.1 Vom Farb- zum Grauwertbild 17
4.2.2 Vom Grauwert- zum Binärbild 17
4.3 Skelette als Kettencode 21

5 Finden der Textlinien 23
5.1 Unterschiedliche Lösungswege 24
5.1.1 Zonenmethode 24
5.1.2 Hough-Transformation und Lernalgorithmus 24
5.1.3 Lokale Referenzlinien 25
5.2 Formulierung des Algorithmus 25
5.2.1 Finden der lokalen Minimum- und Maximumpunkte 26
5.2.2 Finden der Liniensegmente unter einem großen Winkelbereich . 28
5.2.3 Rotation der Textzeilen bestimmen 29
5.2.4 Finden der Textliniensegmente 30
5.2.5 Entfernen der schlechten BLS-Alternativen 30
5.2.6 Verschmelzen der Basisliniensegmente 31
5.2.7 Verbinden der Segmente zu den fertigen Basislinien 33
5.2.8 Entfernen von falschen Basislinien 34
5.2.9 Verbinden von unterbrochenen Basislinien 34
5.2.10 Finden der Mittellinie 35
5.3 Verlängerung der Textlinien 36
5.4 Zusammenfassung 38

6 Zeilensegmentierung 39
6.1 Zuordnung der Kontinuen zu den Zeilen 39
6.2 Trennung verbundener Zeilen 41
6.2.1 Finden der Verbindungsstellen 42
6.3 Analyse der Kreuzungspunkte 43
6.3.1 Der Algorithmus 45
6.3.2 Beispiel 46
6.4 Erzeugung von Grauwertbildern 47

6.4.1 Einfärben ganzer Kontinuen 48
6.4.2 Einfärben von Kontinuumteilen 48
6.4.3 Richtiges Einfärben der Kreuzungsbereiche 48

7 Test der Textlinienfindung 51
7.1 Schriftproben 51
7.2 Parameter 51
7.3 Ergebnisse 54
7.4 Fehlerstellen 55
7.4.1 Abweichung der Basislinie 55
7.4.2 Abweichung der Mittellinie 55
7.4.3 Nichtfinden der Mittellinie 56

8 Besonderheiten der Implementierung 59
8.1 Warum VECTORY? 59
8.2 Zur Textlinienfindung 60
8.2.1 Objekte und deren wechselseitigen Verweise 60
8.2.2 Hinweis zum Finden der Basisliniensegmente 62
8.2.3 Zur Zeilensegmentierung 62
8.3 zur Erzeugung der Grauwertbilder 62

9 Zusammenfassung und Ausblick 63
9.1 Fazit 63
9.2 Ausblick 64
9.2.1 Verbesserungen und Erweiterungen 64
9.2.2 Zum Handschrifterkenner 66

Literaturverzeichnis 69

A Testergebnisse 73

B Beispielbilder: Phasen der Textlinienfindung 77

C Beispielbilder: Textlinienfindung **83**

Kapitel 1

Einleitung

„Denn die Bücher, die Schatzkammern der Sprache, sind ja Testamente, in denen unsere besten Ahnen ihren Besitz vererben, sie sind die Samen, die wir selbst in die Zukunft streuen.“

Richard Müller-Freienfels

1.1 Motivation

Zu den Aufgaben von Historikern und Genealogen gehören vor allem Recherchen in alten Aufzeichnungen, wie sie in alten Bibliotheken, Stadt- oder Kirchenarchiven zu finden sind. Die große Zahl dieser Bücher macht ein komplettes Erfassen sehr schwierig – wenn nicht gar unmöglich. Denn neben dem Problem der sehr langwierigen und teuren manuellen Übertragung der Inhalte in Datenbanken (über Jahre hinweg wären Personen mit der sehr monotonen Aufgabe beschäftigt, die Inhalte dieser Bücher in den Computer einzugeben) besteht außerdem die Gefahr, dass in absehbarer Zeit kaum noch jemand diese alte Schrift lesen kann – nicht aus dem Grund, weil vielleicht die Aufzeichnungen unlesbar werden, sondern weil diese Schrift nicht mehr gelehrt wird und somit bis auf wenige Schriftexperten niemand mehr diese Schrift kennt. Daher wäre es sehr wünschenswert, wenn die Informatik zur Lösung dieses Problems beitragen könnte. Dabei ist es natürlich ein langer Weg vom vergilbten Kirchenbuch zum fertigen Datensatz der Ahnenforschungssoftware.

Handschrift ist zwar von Schreiber zu Schreiber anders, wir haben es hier aber mit Symbolen zu tun, die auf bestimmten Konventionen zwischen den Menschen eines Kulturkreises beruhen. D. h. es existieren Gesetzmäßigkeiten, die eingehalten werden, damit das Ziel von Schrift – nämlich Kommunikation – erreicht wird. Diese Gesetze oder Regeln sind sehr vielschichtig. Angefangen von den notwendigen Eigenschaften eines Buchstabens, damit beispielsweise ein „a“ von einem „d“ unterschieden werden kann, erstrecken sich diese Regeln über die Anordnung und Größenverhältnisse der Buchstaben zueinander bis hin zu kontextuellen Zusammenhängen zwischen der Verwendung von Worten oder ganzen Ausdrücken. Erst wenn es uns gelungen ist, all diese Regeln, die wir als Menschen teilweise unbewusst benutzen, zu erfassen, zu formalisieren und

in eine computergerechte Form zu übertragen, können wir ein System entwickeln, das einen beliebigen Text genauso fehlerfrei „versteht“, wie wir Menschen.

Für die Informatik besteht hier der zentrale Punkt in der Entwicklung eines geeigneten Erkennungssystems. Erkennungssoftware benötigt man immer dann, wenn ein Dokument in analoger Form vorliegt und dessen Inhalt digital verwalten oder weiterverarbeiten möchte. Im Allgemeinen ist dies die Papierform, es kann sich aber auch um belichtetes Filmmaterial oder um eine Audioaufnahme handeln. Das Hauptmerkmal dieser analogen Form der Dokumente ist die Tatsache, dass die darin enthaltene Information nur vom Menschen *direkt* erkannt werden kann.

1.2 Ziele der Arbeit

Der Ausgangspunkt sind Kirchenbücher, in denen über Jahrhunderte hinweg Eintragungen über die Mitglieder der Gemeinde getätigt wurden. Es handelt sich hierbei um altes, vergilbtes Papier mit einer Vielzahl von mehr oder weniger geschwungenen Handschriften. Deren Inhalt zu erkennen und ihn in eine entsprechende Datenbank zu übertragen, ist hier das *ferne* Ziel.

Bevor ein spezieller Erkenner sich an die Arbeit machen kann, um einzelne Worte oder Wortgruppen zu erkennen, sind eine Vielzahl von Arbeitsschritten zur Vorverarbeitung nötig. Ganz am Anfang steht hier der Schritt des *Digitalisierens*. Ausgehend von dem real existierenden Original muss eine digitale Kopie der darin enthaltenen Information erzeugt werden. Danach kann die *Dokumentenverarbeitung* beginnen. Dieser Begriff wird oft im Zusammenhang mit Erkennungsverfahren genannt [TYCS91]. Er umfasst alle notwendigen Schritte, die – ausgehend vom vorliegenden Dokument – zum elektronischen Erfassen der Information erforderlich sind [TCL+99]. Als Ausgangssituation liegen die Dokumente in unstrukturierter, digitaler Form vor, wie das bei einem durch Scannen erzeugten Pixelbild der Fall ist. Grundsätzlich lässt sich der Ablauf in zwei Phasen unterteilen. Nach dieser Einteilung besitzt jedes Dokument eine geometrische und eine logische Struktur. In der ersten Phase, der Dokumentenanalyse, wird die geometrische Struktur analysiert, während in der zweiten Phase, dem Dokumentverstehen, die geometrische Struktur auf eine logische umgesetzt wird.

Kim et al. erläutern in [KGS99] ein komplettes Erkennungssystem, das aus fünf Modulen besteht:

1. der Vorverarbeitung,
2. der Textlinienerkennung,
3. dem Bestimmen von Wortgrenzen,
4. der Worterkennung und
5. der linguistischen Nachverarbeitung.

In der strukturellen Unterteilung kann man diese Module folgendermaßen einordnen:

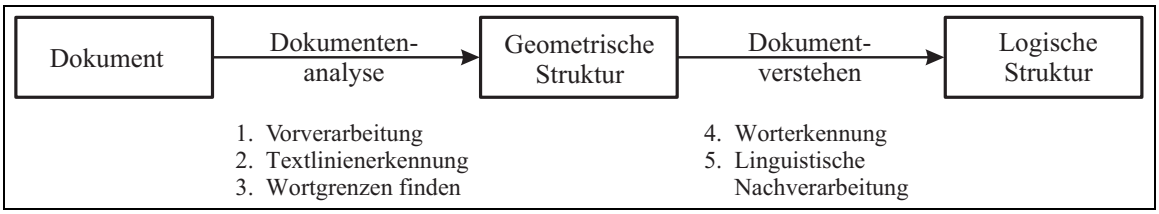


Abbildung 1.1: Dokumentenverarbeitung [TYCS91]

Auf die Kirchenbücher bezogen ist die Dokumentenanalyse das Finden von Textlinien und Schriftzügen. Es gilt also zu bestimmen, wo Federzüge sind und in welchem Verhältnis sie zueinander stehen. Daran schließt sich das Dokumentverstehen an, d. h. ein Erkenner erkennt den auf dem Dokument befindlichen Text auf der Grundlage der zuvor analysierten geometrischen Struktur. Je nach Art der Schrift ist das Finden der Wortgrenzen allein auf der Grundlage von geometrischen Analysen kaum möglich und muss dem Dokumentverstehen zugeordnet werden.

Das Ziel dieser Arbeit ist es, für die Verarbeitung der Kirchenbuchaufzeichnungen den Teil der Dokumentenanalyse zu entwickeln. Das entspricht ungefähr den ersten beiden Modulen in der oben genannten Einteilung. D. h. es soll eine geeignete Repräsentation des Kirchenbuchtextes erzeugt werden, damit ein für dieses Problem entwickelter Erkenner den Inhalt für weitere Anwendungen wie Ahnendatenbanken extrahieren kann.

1.3 Gliederung der Arbeit

Kapitel 2 gibt nach der Einleitung einen Überblick über das Gebiet der Schrifterkennung. Zuvor werden grundlegende Begriffe geklärt, die für die Arbeit wichtig sind. Es werden die Teilbereiche und Lösungsansätze erläutert und die Unterschiede zu dem hier anstehenden Problem der Kirchenbuchaufzeichnungen aufgezeigt.

Der Frage nach den Merkmalen der Schrift in Kirchenbüchern wird in Kapitel 3 nachgegangen. Dabei wird zunächst der ideale Aufbau einer Seite erläutert und systematisiert, bevor dann die Eigenheiten und Ausnahmen näher betrachtet werden.

Alle notwendigen Schritte, die vor der eigentlichen Textlinienfindung durchgeführt werden müssen, um entsprechende Daten zu erhalten, werden in Kapitel 4 erläutert. Dazu gehören die *Digitalisierung*, die *Binarisierung* inklusive der Beseitigung von Störungen und die *Skelettierung*.

Mit der Textlinienfindung beschäftigt sich das Kapitel 5 und ist damit das bedeutendste dieser Arbeit. Schrittweise wird beschrieben, wie zunächst lokale Extrema, Liniensegmente und ganze Textlinien eines Textblockes gefunden werden.

Wie mit Hilfe der Textlinien die einzelnen Zeilen segmentiert werden, wird in Kapitel 6 behandelt. Dabei geht es um die Bewertung der Zugehörigkeit von Kontinuen und Kontinuumteilen zu den einzelnen Zeilen. Außerdem wird gezeigt, wie durch Ana-

lyse von Kreuzungspunkten zweier Linien eine verbesserte Trennung von miteinander verbundenen Textzeilen erreicht werden kann.

Da der Erfolg der Textlinienfindung sowohl von der vorliegenden Schrift als auch von den zuvor einzustellenden Parametern abhängt, wird in Kapitel 7 die Robustheit des Algorithmus anhand von Tests geprüft.

In Kapitel 8 werden Besonderheiten der durchgeführten Implementierung der in den vorigen Kapiteln beschriebenen Algorithmen analysiert.

Kapitel 9 bietet zum Einen noch einmal einen Überblick über die in dieser Arbeit beschriebenen Algorithmen, zum Anderen werden Ideen zur weiteren Verbesserung gegeben sowie Gedanken zu einem zukünftigen Erkenner beschrieben.

Kapitel 2

Erkennungssysteme und -verfahren

Das Gebiet der allgemeinen Schrifterkennung ist sehr groß. Für jeden speziellen Problemfall gibt es genauso spezielle Lösungen, die sich in ihren Ansätzen zum Teil stark unterscheiden. Dieses Kapitel gibt einen Überblick über die verschiedenen Bereiche der Schrifterkennung. Als erste große Differenzierung kann die Unterteilung der Schrifterkennung in Druckschrift- und Handschrifterkennung angesehen werden. Während die Druckschrifterkennung u. a. aufgrund ihrer geringeren Komplexität schon recht ausgereift ist, ist es für die Handschriftenerkennung nicht möglich, *ein* Verfahren als „Alleskönner“ für *alle* handschriftlichen Probleme einzusetzen. So unterteilt sich das Gebiet der Handschrifterkennung in Zeichenerkennung und Gesamtworterkennung. Je nach Problemfeld unterscheiden sich außerdem die notwendigen Vorverabeitungsschritte der Erkennungssysteme. Des Weiteren lässt sich die Handschrifterkennung durch die Art der Datengewinnung in die so genannte „On-line“- und „Off-line“-Erkennung einteilen.

2.1 Begriffserläuterung

Bevor näher auf die Erkennungssysteme und -verfahren eingegangen wird, werden zunächst grundlegende Begriffe erläutert, auf die sich meine Arbeit stützt. Dies sind die Begriffe des *Kettencodes nach Freeman* [Fre61] und der *Hough-Transformation*.

2.1.1 Kettencode nach Freeman

Als Datenbasis für die Textlinienfindung und Federzug-Rekonstruktion dient das von VECTORY erzeugte Skelett der Objekte, das im *Kettencode* vorliegt.

Kettencode ist eine beliebte Repräsentationsform für Konturen und Linien. Das erste Mal wurde er von Freeman 1961 vorgeschlagen [Fre61]. Über eine vorliegende Kurve wird ein Gitter gelegt. Die Kurve wird durch die Kreuzungspunkte des Gitters approximiert, die am nächsten am Verlauf der Kurve liegen und miteinander verbunden sind. Es entsteht eine Kette aus Nachbarschaftspunkten. Angefangen von dem ersten Referenzpunkt ist der nächste Punkt jeweils einer der acht Nachbarschaftspunkte. Welcher Nachbar das ist wird durch eine Nummer zwischen Null und Sieben angegeben. Hierbei gilt die Festlegung, dass der rechte Nachbar die Nummer Null trägt und gegen den Uhrzeigersinn die Nachbarn bis Sieben durchnummeriert werden (siehe Abbildung 2.1).

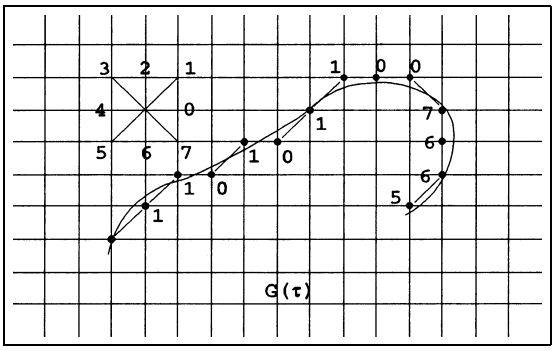


Abbildung 2.1: Kettencodierung einer Kurve [Liu92]

Da sich die Bildverarbeitung mit digitalisierten Bildern beschäftigt, werden die Kreuzungspunkte des Gitters durch Pixel ersetzt, die im Gegensatz zu den oben betrachteten Gitterpunkten eine örtliche Ausdehnung besitzen. Dies macht aber für die Kodierung keinen Unterschied, da hier die Nachbarschaftsverhältnisse identisch sind.

2.1.2 Hough-Transformation

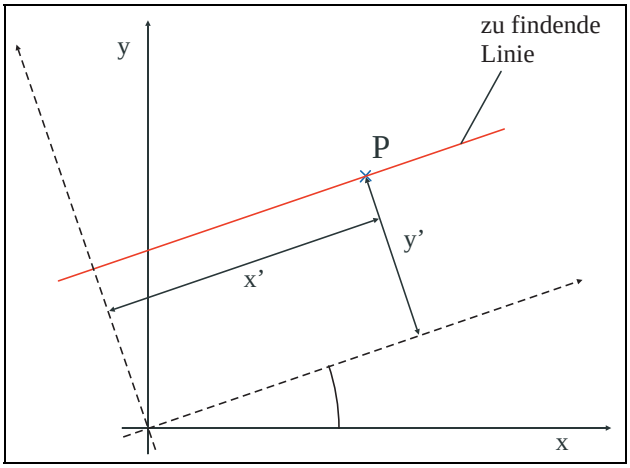


Abbildung 2.2: Linienfindung durch Hough-Transformation

Im Allgemeinen wird die Hough-Transformation dazu genutzt, um Linien, Kreise oder ähnliche primitive Objekte, die sich durch wenige Parameter beschreiben lassen, zu erkennen bzw. um festzustellen, ob die Anordnung von Punkten in einem kartesischen Koordinatensystem diesen geometrischen Objekten entsprechen. Dazu werden diese Punkte in einem so genannten Parameter-Raum (Hough-Raum) abgebildet.

Für Geraden heißt das, dass man sie sie durch zwei Parameter, dem Anstieg bzw. Winkel und dem Offset, definieren kann. In einem zweidimensionalen Koordinatensystem, das in der einen Dimension den Winkel und in der anderes Dimension den Anstieg darstellt, repräsentiert ein Punkt genau eine Gerade.

2.2 Erkennung von Druckschrift

Wenn man von Optical Character Recognition (OCR) spricht, so bezieht sich dieser Ausdruck meist auf das Erkennen von *gedrucktem* Text. Dieser Bereich der Schrifterkennung kann als sehr ausgereift bezeichnet werden. Heutige Erkennungssoftware erzielt selbst bei schlechten Vorlagen, wie beispielsweise bei Faxen Erkennungsraten von fast 100% [Ebe00]. Der Grund für die gute Erkennungsleistung aktueller Erkennungssoftware für Druckschrift liegt neben dem großen Absatzmarkt dieser Software und den damit verbundenen hohen Anstrengungen vor allem bei den festen Gesetzmäßigkeiten und engen Toleranzen der Merkmale. Auf einer Seite mit Druckschrift sind die Textlinien gerade und parallel zueinander. Sowohl die Zeilen als auch die Wörter und Buchstaben sind im Allgemeinen gut und mit konstanten Abständen voneinander getrennt und ein jedes Exemplar eines bestimmten Buchstabens sieht ungefähr gleich aus.

2.3 Handschrifterkennung

Auf dem Gebiet der Handschriftenerkennung sind die Voraussetzungen andere. Wenn es um die Erkennung von handgeschriebenem Text geht, so wurden für unterschiedliche Probleme unterschiedliche Strategien entwickelt. Eine grundlegende Einteilung der Verfahren resultiert aus der Gewinnung der Daten [PS00]. Hieraus ergibt sich eine Einteilung in „on-line“ und „off-line“.

2.3.1 Online-Erkennung

Der im Englischen als „on-line recognition“ bezeichnete Bereich der Handschrifterkennung basiert auf der Verarbeitung der Information, die durch das Aufzeichnen des Pfades des Schreibgerätes während des Schreibvorgangs gewonnen wird. In der Praxis findet man dies beispielsweise bei so genannten PDAs[1], die mit einem Stift oder Griffel bedient werden.

Im Vorfeld einer Erkennung müssen die entsprechenden Daten vorverarbeitet werden, um Störungen zu minimieren, Merkmale zu normalisieren und das Signal in bedeutungsvolle Einheiten zu zerlegen. Die Normalisierung ist notwendig bei Algorithmen, die auf den Vergleich mit standardisierten Grafen und Formen bauen. Zum Normalisierungsschritt gehören u. a. die Korrektur der Basislinien-Bewegung, die Neigung der Schrift und der Schriftgröße.

Die *Rauschminimierung* dient zur Glättung des Linienverlaufes, der zum Einen durch die Eingabegeräte und zum Anderen durch eine nicht perfekte Stiftführung kleine Stö-

[1] „personal digital assistent“ – kleine handliche Geräte mit einem druckempfindlichen Flüssigkristall-Bildschirm, der gleichzeitig als Aus- und Eingabeschnittstelle dient, wird vor allem zur Termin- und Adressverwaltung genutzt

rungen und Haken aufweisen kann. Dies geschieht durch eine kubische Spline-Funktion. Die *Normalisierung* reduziert die geometrische Varianz der Daten. Es handelt sich hierbei um das Standardisieren der Größen, die Drehung der Textlinien und die Korrektur der Neigung der Schrift [HLB00].

Die *Segmentierung* unterteilt sich meist in zwei Stufen. In der ersten Stufe geht es darum, die Textlinien zu finden, Worte zu segmentieren und Text von anderen Objekten zu trennen. In der zweiten Stufe werden einzelne Zeichen oder Teile von Zeichen wie Striche segmentiert. Bei Systemen, die die Erkennung auf Wortebene durchführen, wird diese zweite Stufe nicht durchgeführt[DD96, PAO99].

2.3.2 Offline-Erkennung

Im Gegensatz zur Online-Erkennung hat man in der Offline-Erkennung keine Informationen über den zeitlichen Verlauf des des Pfades des Schreibgerätes. Nach der Digitalisierung des Schriftbildes liegt hier ein zweidimensionales Feld mit Farb- oder Grauwerten vor, worin die Informationen der Schrift enthalten sind.

Der Hauptteil der Vorverarbeitung bildet hier die Binarisierung, die die Pixel der Bildes in Vorder- und Hintergrund einteilt. Fast alle heute gebräuchlichen Systeme setzten ein Binärbild für die Weiterverarbeitung voraus (z. B. [MKG99, GK96]). Nur in einem Fall, bei dem es um das Auffinden von Schrift in Farbbildern geht, werden Grau- oder Farbbilder genutzt [WMR99].

Viel Interesse an solchen Verfahren kommt von Seiten der Banken und Kreditinstitute oder anderer Institutionen, die eine möglichst schnelle Verarbeitung von handschriftlich ausgefüllten Formularen anstreben. Die meisten spezialisierten Erkennungssysteme, wie z. B. für die Betragserkennung auf Schecks, bauen auf einem System auf, das die Worte in ihrer Gesamtheit zu erkennen versucht [SLG+96, EYGSS97]. Dies ist möglich, da hier ein relativ kleines Vokabular vorliegt. Ein spezialisierter Fall ist die Unterschriften-Verifikation. Auch hier wird das Problem durch eine ganzheitliche Betrachtung des Schriftzuges gelöst [Sab97]. Hier findet ein Vergleich mit gespeicherten Unterschriftenmustern statt, der auf bestimmten extrahierten Eigenschaften basiert.

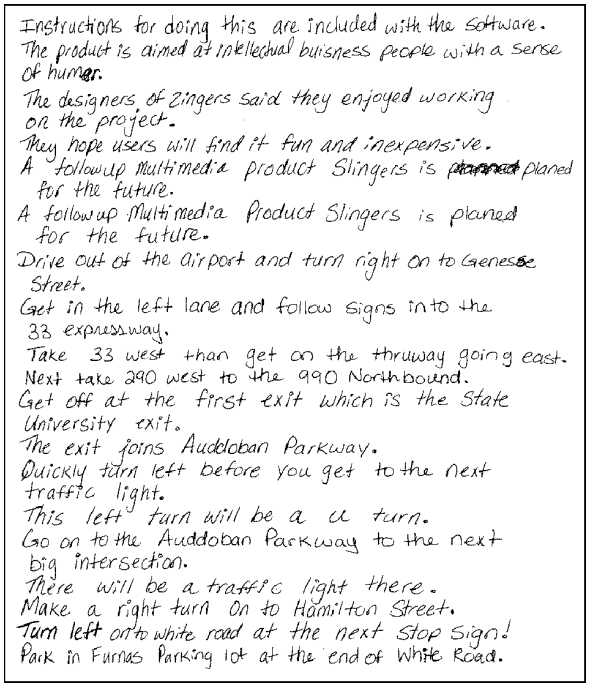
Bei Adresserkennungsverfahren muss anders vorgegangen werden [DSS+90, SXL99]. Hier ist das Vokabular der möglichen Worte erheblich größer. Die Worte werden in kleinere Einheiten zerlegt. Dies sind in dem meisten Fällen die Buchstaben. Es können aber auch Buchstabenkombinationen sein.

Bei all diesen Einsatzgebieten handelt es sich um die Klasse der Erkennung, die im Englischen als „off-line recognition“ bezeichnet wird. D. h. das Datenmaterial für das Erkennungsverfahren ist das Bild einer fertigen Schriftvorlage. Bei der so genannten „on-line recognition“ wird die Bewegung des Schreibgerätes beim Schreiben mit aufgezeichnet. Diese zusätzliche Information wird dann bei der Erkennung genutzt. Ein typisches Anwendungsgebiet bildet hier der Markt der schon erwähnten elektronischen

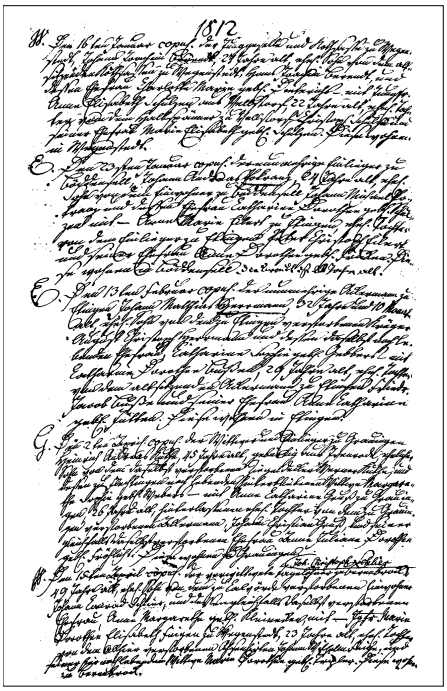
Terminplaner (PDAs), bei denen mit einem Griffel auf einem druckempfindlichen Bildschirm geschrieben wird.

Es gibt meines Erachtens zwei Gründe, warum sich auf dem Gebiet der automatisierten Erkennung historischer Dokumente noch nicht all zu viel getan hat: Zum Einen steht hinter den Interessenten dieser Technik nicht das große Kapital, wie dies bei den Banken und Versicherungen der Fall ist. Zum Anderen ist dies der wohl anspruchsvollste Bereich der Schrifterkennung.

Die oben genannten Gebiete der Handschriftenerkennung behandeln Probleme, die gegenüber der Erkennung von Kirchenbuchaufzeichnungen einen entscheidenden Vorteil haben: Es ist von vornherein bekannt, wo sich der zu erkennende Text befindet und welche Ausmaße er hat. Daher ist zwar die Zahl der entwickelten Verfahren für Zeichen- oder Worterkenner relativ groß, für das davor gelagerte Problem, den Texte auf einer Dokumentenseite zu finden und sich berührende Zeilen voneinander zu trennen, gibt es nur sehr wenige Ansätze. In Kirchenbüchern kann die Größe der Schrift und der Abstand der Zeilen stark variieren, da mitunter auf einer Seite Eintragungen zu finden sind, die mehrere Jahrzehnte auseinander liegen und von unterschiedlichen Personen stammen. Hier gibt es keine festgelegten Bereiche oder Felder, die einfach herausgegriffen und einem Zeichen- oder Worterkenner übergeben werden können.


Instructions for doing this are included with the software.
The product is aimed at intellectual buisness people with a sense of humor.
The designers of Zingers said they enjoyed working on the project.
They hope users will find it fun and inexpensive.
A followup multimedia product Slingers is planned for the future.
A followup Multimedia Product Slingers is planed for the future.
Drive out of the airport and turn right on to Genesee Street.
Get in the left lane and follow signs into the 33 expressway.
Take 33 west than get on the thruway going east.
Next take 290 west to the 990 Northbound.
Get off at the first exit which is the State University exit.
The exit joins Audoloban Parkway.
Quickly turn left before you get to the next traffic light.
This left turn will be a u turn.
Go on to the Audoloban Parkway to the next big intersection.
There will be a traffic light there.
Make a right turn on to Hamilton Street.
Turn left onto white road at the next Stop Sign!
Park in Furnas Parking lot at the end of White Road.

(a) aus [KGS99]



(b) aus Wegenstedter Kirchenbuch, 1812

Abbildung 2.3: Vergleich zwischen Kirchenbuch und gängigen handschriftlichen Vorlagen

Selbst Arbeiten, die sich mit der Erkennung von handgeschriebenen Textseiten als so genannte *unconstrained data*[2] beschäftigen, beziehen sich meistens auf einfacher gestrickte Dokumentvorlagen, wenn man sie mit den hier vorliegenden Kirchenbuchaufzeichnungen vergleicht. Dies ist z. B. bei dem oben schon erwähnten Paper von Kim et al. der Fall [KGS99]. Hier wird zwar eine Methode zur Textlinienfindung beschrieben, doch wie in Abbildung 2.3(a) ersichtlich ist, ist hier die Schrifthöhe und Zeilenabstand relativ konstant und alle Zeilen sind gut durch Zwischenräume voneinander isoliert. Dies ist ein gutes Beispiel für den Gegenstand der heutigen Forschung im Bereich „Offline"-Handschriftenerkennung.

[2] Dieser Ausdruck beschreibt die Herkunft der Eingangsdaten näher und besagt, dass der Schreiber dieses Textes nicht weiß, dass seine Schrift maschinell gelesen werden soll und daher auch keine Kooperation vorliegen kann. [Ste95]

Kapitel 3

Merkmale der Kirchenbuchaufzeichnungen

Um dem Problem der Segmentierung einer Schriftseite begegnen zu können, ist es zunächst erforderlich, die grundlegenden Eigenschaften einer solchen Seite und der sich darauf befindlichen Schrift zu betrachten. Dies geschieht zunächst in idealisierter Weise, um die Struktur systematisch erfassen zu können. In einem nächsten Schritt werden dann die doch sehr zahlreichen Ausnahmen erläutert.

3.1 Idealisierter Aufbau einer Schriftseite

Eine **beschriebene Seite** setzt sich aus einer oder mehreren *Textblöcken* zusammen. Diese erscheinen entweder in Form von Absätzen, die übereinander angeordnet sind, oder in Form von Feldern einer Tabelle, die über- und nebeneinander liegen. In beiden Fällen sind die Textblöcke gut voneinander getrennt.

Ein **Textblock** besteht aus mehreren übereinander angeordneten *Textzeilen*, die nicht miteinander verbunden sind und erwartungsgemäß nicht absolut waagerecht und gerade verlaufen.

Eine **Textzeile** besteht aus mehreren hintereinander angeordneten *Worten*, die sich nicht berühren. Ein Wort wird im Folgenden auch als Schriftzug bezeichnet. Außerdem gehören zur Textzeile Zeichen der Interpunktion.

Ein **Wort** besteht aus miteinander verbundenen *Federzügen* und Objekten, wie i-Punkte und Punkte auf Umlauten und „Verzierungen“, die zu der jeweiligen Zeit üblich waren (siehe Abbildung 3.1).

Abbildung 3.1: „Verziehrung“ über dem u

Federzüge sind das Ergebnis, das durch das Aufsetzen, Bewegen und Absetzten einer mit Tinte getränkten Feder entsteht. Dabei kann ein Federzug aus einem oder mehreren **kontinuierlichen Segmenten** bestehen. Das heißt, man kann davon ausgehen, dass bei einer Federbewegung die Richtungsänderung kontinuierlich verläuft. In einem Wendepunkt, in dem die Feder gestoppt wird, wechselt die Richtung relativ stark und ein neues kontinuierliches Segment beginnt.

Zeichen der Interpunktion und die zusätzlichen Objekte der Worte werden im Folgenden unter dem Begriff der *Kleinobjekte* zusammengefasst. Daraus folgt für das digitale Bild, dass ein Wort eine Menge von miteinander verbundenen dunklen Pixeln darstellt. Im Idealfall sind die Worte einer Zeile nicht miteinander verbunden und im Idealfall sind auch die Zeilen einer Seite nicht miteinander verbunden.

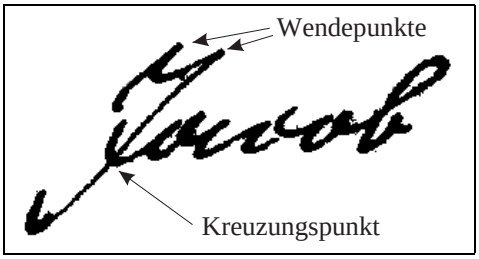


Abbildung 3.2: Schriftzug „Jacob“

Beim Schriftzug „Jacob“ in Abbildung 3.2 kann man erkennen, dass das „J“ einen Federzug darstellt, der aus drei kontinuierlichen Segmenten und zwei Wendepunkten besteht. Eine Verbindung in Form eines Kreuzungspunktes besteht zwischen den beiden Federzügen des Buchstaben „J“ und des Buchstaben „a“.

Im idealen Fall besitzt eine handschriftliche Zeile genau wie die Druckschrift vier Textlinien[1]: Die *untere Textlinie* wird von der unteren Grenze der Zeichen mit Unterlänge gebildet, wie z. B. „y“ oder „ j“. Die *Basislinie* ist die imaginäre Linie, auf der geschrieben wird, und wird von der Untergrenze aller Zeichen ohne Unterlänge gebildet. Die *Mittellinie* wird von der Obergrenze der kleinen Zeichen erzeugt während die *obere Textlinie* von den großen Zeichen gebildet wird.

3.2 Ausnahmefälle

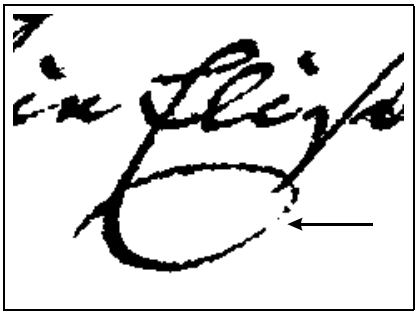
Wie schon erwähnt, handelt es sich bei den oben geschilderten Merkmalen um eine idealisierte Betrachtung. Dies sind ideale Eigenschaften, deren Gültigkeit stark eingeschränkt werden muss. Es ergeben sich in der Realität eine große Anzahl von Ausnahmefällen, die das Problem verkomplizieren.

So kann ein Federzug aufgrund einer Unterbrechung des Tintenflusses beim Schreiben Lücken aufweisen. Es besteht die Möglichkeit durch den kontinuierlichen Verlauf des Federzuges diesen zu rekonstruieren.

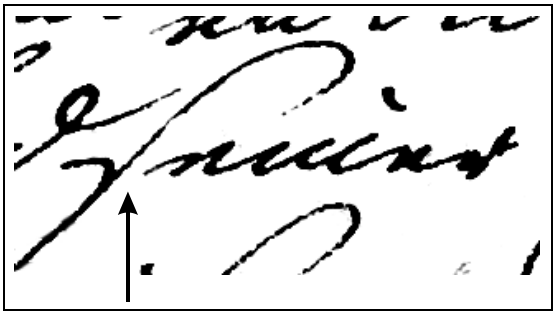
Weiterhin müssen sich Federzüge eines Wortes nicht immer berühren, was wiederum zwischen zwei hintereinander stehenden Wörtern durchaus der Fall sein kann.

Im Gegensatz zu heutiger Schreibschrift oder gar Druckschrift ist es bei diesen alten Dokumenten häufig der Fall, dass einzelne Federzüge sehr stark nach unten in die

[1] Man kennt sie vielleicht noch aus den Schreibübungsheften der 1. Klasse.

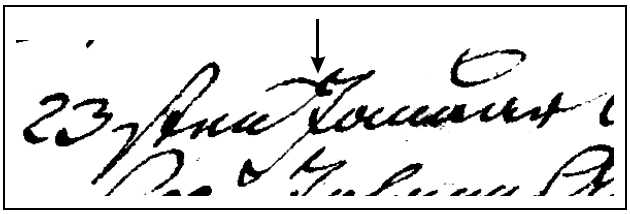


(a) in einem kontinuierlichen Federzug

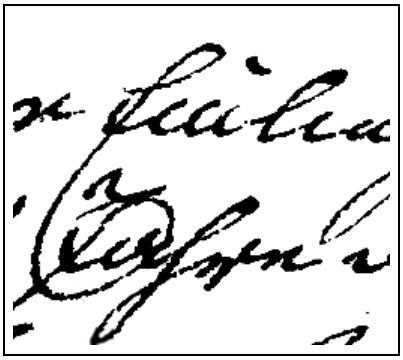


(b) zwischen zwei Federzügen

Abbildung 3.3: Lücken im Schriftbild



(a) Berührung zwischen zwei hintereinanderstehenden Worten



(b) Hineinragen einer Schriftlinie in die darunter liegende Zeile

Abbildung 3.4: Berührungen

benachbarten Textzeilen ragen können – nicht jedoch nach oben[2]. Das liegt vermutlich daran, dass beim Schreiben einer Zeile die darüber liegende Zeile schon geschrieben wurde und zu sehen ist, während das für die folgende, darunter liegende Zeile nicht der Fall ist.

Dieses Hineinragen hat zur Folge, dass Schriftlinien einer Zeile in den Zeilenraum einer anderen Zeile ragen, dabei aber *erstens* keine Schriftlinien dieser Zeile berühren oder aber *zweitens* eine Berührung stattfindet. Es hängt von der Art dieser Berührung ab, ob eine korrekte maschinelle Trennung der beiden Zeilen problematisch ist oder nicht.

Obwohl theoretisch vier Textlinien existieren, sind oft nur die Basis- und Mittellinie genügend ausgeprägt. Die obere Textlinie der großen Zeichen und die untere Textlinie der Unterlängen ist bei den meisten alten Schriften kaum vorhanden. Zum Einen ist

[2] Ausnahme: Auf einer Wegenstedter Kirchenbuchseite von 1741 berührt an zwei Stellen ein Federzug die Basislinie der darüber liegenden Zeile.

die Zahl diese Zeichen relativ gering und zum Anderen ist die vertikale Abweichung dieser Extrempunkte zu groß, um daraus eine Linie zu rekonstruieren.

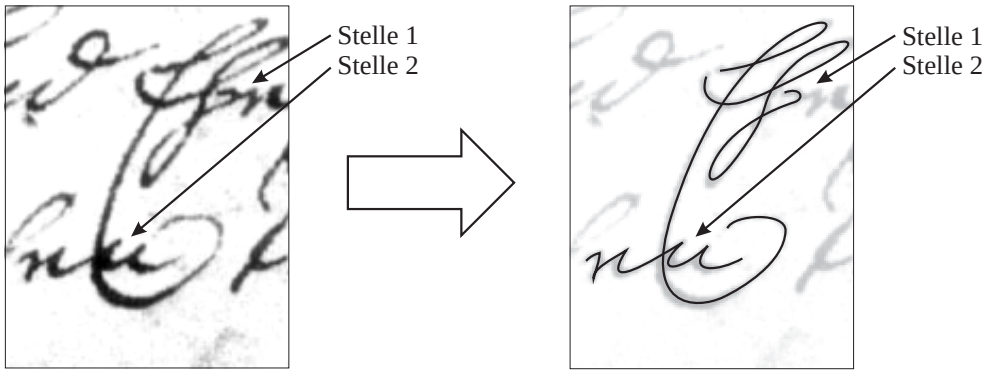


Abbildung 3.5: Idealvorstellung einer Spline-Repräsentation

In Abbildung 3.5 kann man zwei sehr typische Problemfälle erkennen. So sind an Stelle (1) mehrere Linienverläufe denkbar. Selbst als Mensch hat man hier Probleme, den korrekten Verlauf zu erkennen. Aber auch an Stellen, wo der Mensch keine großen Probleme hat, den Linienverlauf zu rekonstruieren, ist für den Computer ein sehr komplexes Problem. Hier muss zunächst untersucht werden, was die Merkmale sind, die einen Menschen den Verlauf erkennen lassen. So muss auch die Frage gestellt werden, inwieweit das Kontextwissen und Informationen, die nicht unmittelbar mit dem Schriftbild zu tun haben, in die Analyse mit hineinfließen. Ein ähnlich gelagertes Problem ist an Stelle (2) zu erkennen. Im Gegensatz zu einer eindeutigen Überkreuzung „verkleben“ hier mehrere Schriftlinien zu einem dunklen Objekt, bei dem es sehr schwierig ist, den Verlauf der Linien im Inneren zu erkennen.

Kapitel 4

Vorverarbeitung

In diesem Kapitel werden die notwendigen Schritte behandelt, die erforderlich sind, um aus einer realen (analogen) Kirchenbuchseite das Skelett der darauf befindlichen Objekte zu erhalten.

Als Erstes ist dazu die *Digitalisierung* erforderlich, um die so gewonnenen Daten anschließend im Rechner weiterverarbeiten zu können. Danach muss das gewonnene Farbbild in ein *Binärbild* konvertiert werden, indem eine Einteilung in Vordergrund und Hintergrund erfolgt. VECTORY erzeugt daraus des Skelett als Kettencode, das als Ausgangspunkt für die in den folgenden Kapiteln behandelte Textlinienfindung und Zeilentrennung genutzt wird.

4.1 Digitalisierung

Im Folgenden werden die unterschiedlichen Möglichkeiten aufgezeigt, um von dem realen (analogen) Dokument eine digitale Kopie zu erhalten, die die Voraussetzung für die folgende rechnergestützte Verarbeitung darstellt. Diese digitale Kopie bildet die Grundlage aller weiteren Verarbeitungsschritte. Sie hat die Form eines Raster-Farbbildes mit 24 Bit Farbtiefe und einer Ortsauflösung von 300 dpi (188,11 Pixel/cm), die für die weitere Verarbeitung vollkommen ausreichend ist. Im Mittel bewegen sich die Linienbreiten in einem Bereich von 0,5 bis 0,8 mm. An sehr schmalen Stellen beträgt die Breite kaum weniger als 0,2 mm. Im Rasterbild werden solche Linien mit ausreichenden 3 bis 4 Punkten dargestellt.

Es werden drei unterschiedliche Wege untersucht, um diese digitale Datenbasis zu erhalten. Bei dem Vergleich ist es neben den qualitativen Aspekten der Kopie vor allem wichtig, die mechanische Belastung der Originaldokumente zu betrachten. Es handelt sich bei den drei Wegen um die Verwendung eines Flachbettscanners, einer Digitalkamera und einer Analogkamera in Verbindung mit einem Durchlichtscanner.

4.1.1 Flachbettscanner

Die Klasse der Scanner kann in vier Unterkategorien unterteilt werden: Flachbettscanner, Einzugscanner, Handscanner und Trommelscanner [DIPS97]. Von vornherein

auszuschließen sind die Einzugs- und Trommelscanner. Zum Einen erfordern beide Arten das Heraustrennen der Seiten aus den Buch und zum anderen kann die mechanische Beanspruchung während des Scannens nicht in Kauf genommen werden. Handscanner, die noch bis Mitte der 90er-Jahre für den Privatgebrauch die einzige erschwingliche Möglichkeit darstellten, spielen heutzutage für die Digitalisierung von Schrift oder Bildern kaum noch eine Rolle. Zu häufig kommt es während des Scannens zu Verzerrungen, da eine sehr exakte Führung des Scannens über das Dokument erforderlich ist. Auch die mechanische Belastung der Dokumentenoberfläche ist bei einem Handscanner nicht zu unterschätzen.

Die Klasse Flachbettscanner schneidet nach dem Abwägen von Vor- und Nachteilen am besten ab. Hier wird zwar ein fest gebundenes Buch durch das Auflegen auf den Scanner relativ stark belastet, dies liegt aber bei sorgsamer Handhabung im Bereich des Akzeptablen. Dieses Verfahren hat neben den sehr geringen Kosten den Vorteil des geringen Zeitaufwands und der vergleichsweise guten Qualität. Die Kosten für einen Flachbettscanner belaufen sich heutzutage auf unter 500 DM. Nach dem Scanvorgang, der pro A3-Seite nicht länger als 10 Minuten beansprucht[1], liegt das digitale Farbbild zur Weiterverarbeitung bereit.

4.1.2 Digitalfotografie

Die relativ junge Möglichkeit der Digitalfotografie hat – wie auch das vorige Verfahren – den Vorteil, relativ wenig Zeit in Anspruch zu nehmen. Es muss hierbei aber auf die korrekte Ausleuchtung geachtet werden. Darüber hinaus wird das Originalmaterial mechanisch nur gering belastet. Allerdings muss eine eventuelle optische Verzerrung durch die Krümmung einer Seite eines aufgeschlagenen Buches nachträglich korrigiert werden.

Aufgrund des gegenwärtigen technischen Standes kann man dieses Verfahren wegen der zu geringen Ortsauflösung heute noch nicht empfehlen. Moderne Kameras der so genannten Consumer-Klasse liefern heutzutage Bilder mit 1600x1200 Pixel (ca. 2 Millionen Pixel). Das heißt, um eine Auflösung der Schrift von 300 dpi zu erreichen, würde man mit einer Aufnahme lediglich einen Ausschnitt von 85x64 mm erfassen. Außerdem bewegen sich die Kosten für eine Digitalkamera in Regionen um 1500 bis 3000 DM. Professionelle Modelle, mit denen das Vorhaben problemlos möglich wäre, kosten üblicherweise ca. 60 000 DM.

4.1.3 Analogfotografie & Durchlichtscanner

Ähnlich wie bei der Digitalfotografie wird auch bei der Analogfotografie das Originalmaterial mechanisch nur gering belastet. Der Vorteil gegenüber der Digitalfotografie ist jedoch die sehr gute Ortsauflösung. So liegt die Auflösung eines 100-ISO-Kleinbildfilms

[1] Dies ist auch abhängig vom verwendeten Computer, der vor allem über ausreichend Arbeitsspeicher verfügen sollte

bei ungefähr 12 Millionen Pixel [SL99]. Die Preise für einen geeigneten Durchlichtscanner liegen derzeit zwischen 3000 und 5000 DM. Dafür erhält man ein Gerät, das die Negative oder Dias mit einer Auflösung von bis zu 4000 dpi einließt. So eine Auflösung ist allerdings auch erforderlich, da das Abbild einer A4-Seite auf einem 36mm-Film (24 x 36 mm) um fast das 9-fache verkleinert wird.

4.2 Binarisierung

Das Ergebnis des Digitalisierens ist ein 24-Bit-Farbbild, d. h. es enthält drei Farbkanäle zu je 8 Bit. Es geht nun um das Ziel, daraus ein Binärbild zu erzeugen, das die Einteilung in Vorder- und Hintergrund repräsentiert – also in Pixel, die zur Schrift gehören, und Pixel, die nicht zur Schrift gehören. Dazu muss zunächst aus dem Farbbild ein Grauwertbild erzeugt werden, indem man den besten der drei Kanäle auswählt. Anschließend wird durch die Verwendung verschiedener linearer Filter eine möglichst optimale Binarisierung durchgeführt.

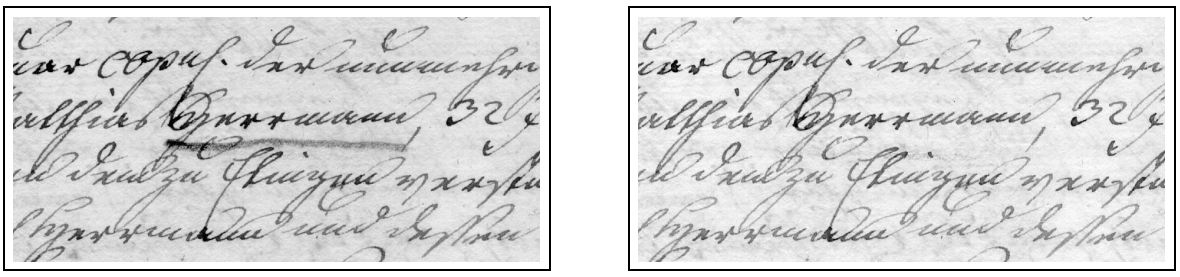
4.2.1 Vom Farb- zum Grauwertbild

Um aus dem Farbbild ein Grauwertbild zu erhalten,wird einer der drei Känale ausgewählt. Durch die farblichen Eigenschaften des Papiers und der verwendeten Tinte wirkt sich der Kontrast unterschiedlich auf die drei Farbkanäle aus. Ein Grauwertbild aus dem Durchschnitt aller Farbkanäle zu erzeugen, wie dies bei Bildverarbeitungsprogrammen üblich ist, würde auch einen durchschnittlichen Kontrast zur Folge haben. Je nach Dokument zeigte entweder der blaue oder der grüne Kanal die besten Werte. Dies ist zum Einen mit dem bloßen Auge zu erkennen, zum Anderen ist die Standardabweichung der Helligkeitswerte im Histogramm für die Entscheidung ausschlaggebend. D. h. je größer die Standardabweichung der Helligkeitswerte ist, umso größer ist der Kontrast.

In einigen Fällen kann die Entscheidung jedoch auf einen Kanal fallen, der nicht den größten Kontrast bietet. In einigen Aufzeichnungen wurden in jüngerer Zeit mit farbigen Text-Markern Namen hervorgehoben. Wie in Abbildung 4.1 ersichtlich ist, wirkt sich diese Störung je nach verwendeter Farbe sehr unterschiedlich auf die einzelnen Kanäle aus.

4.2.2 Vom Grauwert- zum Binärbild

Um eine gute Einteilung in Vorder- und Hintergrund zu erreichen, wurden für die unterschiedlichen Einsatzgebiete entsprechende Verfahren entwickelt. Sollen beispielsweise auf Bank-Schecks handschriftliche Eintragungen erkannt werden, besteht hier das Problem, dass u. a. auf Grund der Fälschungssicherheit ein Linienmuster auf dem gesamten Papier aufgebracht wurde. Erkennungssysteme, die dafür entwickelt wurden



(a) grüner Kanal (b) roter Kanal

Abbildung 4.1: unterschiedliche Auswirkung farbiger Störungen auf die Kanäle

nutzen den Vorteil, dass diese „Störung“ bekannt ist [FK97]. Es wird eine so genannte *morphologische Subtraktion* durchgeführt, um eine Differenz zwischen einem leeren und einem beschriebenen Scheck zu erzeugen. Da bei den Kirchenbüchern kein bekanntes Muster für die Störung verantwortlich ist, kann dieses Verfahren nicht verwendet werden.

Das automatische Setzen eines Schwellwertes, wie in [PS00] beschrieben, ist nur bedingt für das vorliegende Problem tauglich. Es wird davon ausgegangen, dass der Verlauf des Histogramms zwei lokale Maxima besitzt – eins für den hellen Hintergrund und eins für den dunklen Vordergrund. Es wird dann nach bestimmten Verfahren ein Schwellwert zwischen diesen beiden Maxima festgelegt.

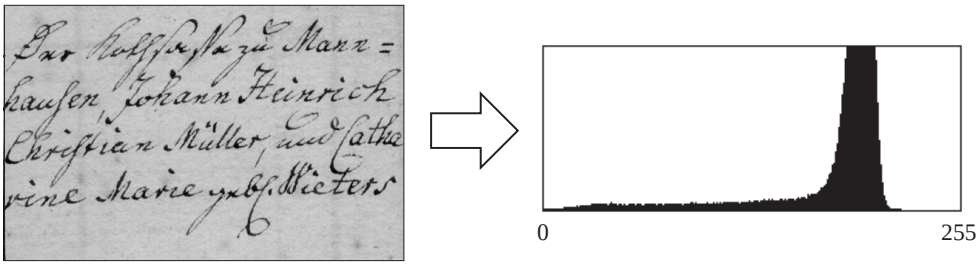


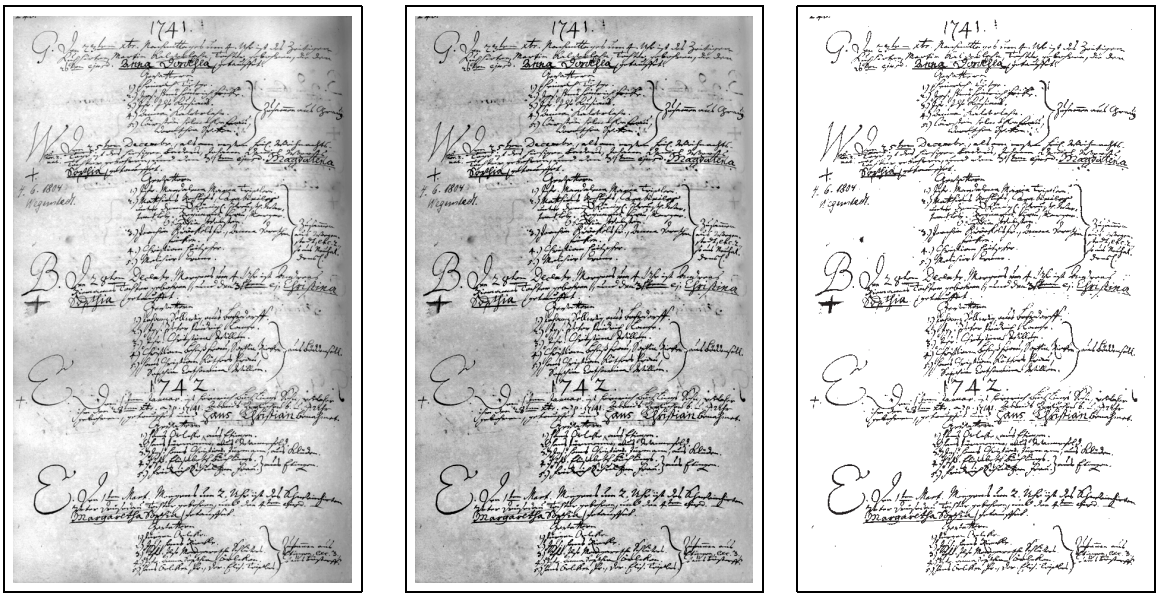
Abbildung 4.2: Blauer Kanal eines Kirchenbuchausschnitts und dessen Histogramm

Wie im Histogramm der Abbildung 4.2 zu sehen ist, ist zwar ein Maximum des hellen Hintergrundes vorhanden, für den dunklen Vordergrund ist dies jedoch nicht der Fall. An dieser Stelle lassen diese Merkmale keine ausreichende Unterscheidung zwischen Schrift und Störung zu.

Unabhängig von der gewählten Digitalisierungsform ist es zunächst erforderlich, durch unterschiedliche Ausleuchtung entstandene Helligkeitsunterschiede der Seite zu egalisieren. Da zu Beginn der Verarbeitung keine weiteren Informationen zur Strukturierung und Segmentierung vorliegen, ist es ausreichend, mittels eines dynamischen Schwellwertverfahrens eine Einteilung der Pixel in Vorder- und Hintergrund durchzuführen. Praktisch wurde zunächst eine Hochpass-Filterung mit einem Radius von 20 Punkten

durchgeführt. Damit werden die niederfrequenten Helligkeitsunterschiede des Dokumentes beseitigt. Die Werte bewegen sich nach der Normierung auf ein 8-Bit-Bild um den „Nullpunkt“ 127.

Anschließend wird mit einem konstanten Schwellwert binarisiert, der je nach Papierqualität bei 43 bis 45% (Grauwert 111 bis 115) trennt[2]. Damit werden vor allem die zur Schrift gehörenden Pixel als Vordergrundpixel markiert.



(a) grüner Kanal (b) Hochpassfilter, Radius 20 (c) Binarisierung mit Schwellwert 111

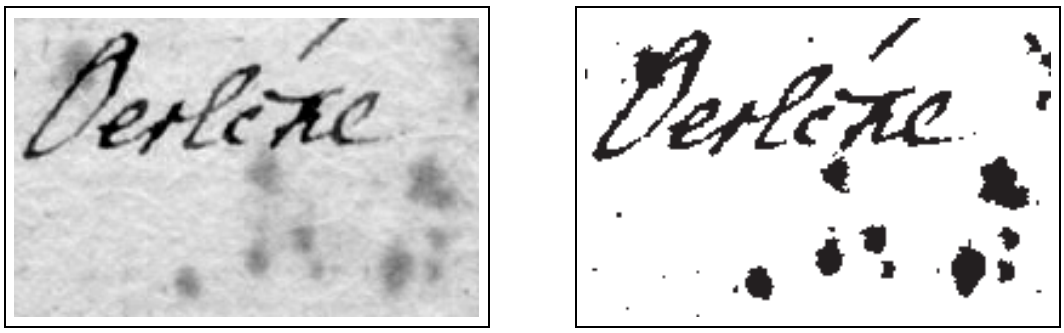
Abbildung 4.3: Binarisierung eines Beispiel-Dokumentes aus Wegenstedt, 1741

Störungen beseitigen

Die von mir digitalisierten Kirchenbuchseiten verfügen über einen relativ guten Kontrast zwischen Papier und Tinte. Gelegentlich treten Störungen in Form von Verunreinigungen, Stockflecken oder Durchscheinen der rückseitigen Schrift auf. Bei der Schwellwert-Verarbeitung können dadurch an diesen Stellen falsche Vordergrundpixel markiert werden.

Diese Störungen im Grauwertbild unterscheiden sich von der Schrift darin, dass ihre Objektkanten unschärfer sind. Diese Eigenschaft wird ausgenutzt, um eine Maske zu erzeugen, die nur die Störungen aus dem Grauwertbild herausfiltert, die Schrift jedoch passieren lässt.

[2] Diese Werte haben sich bei Tests an Kirchenbuchseiten unterschiedlichen Alters als optimal ergeben.



(a) blauer Kanal

(b) nach Hochpassfilterung und Schwellwert

Abbildung 4.4: Stockflecke werden fälschlicherweise als Vordergrundobjekte markiert

Durch den Einsatz eines Hochpassfilters mit einem kleinen Radius von einem Pixel werden niederfrequente Störungen wie z. B. Stockflecken herausgefiltert. Mit anschließender 8-Bit-Normierung hat dieser Filter die Form:

$$H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \cdot \frac{1}{8} + 127 \tag{4.1}$$

Ein anschließender Schwellwertfilter, der bei 43 bis 45% (Grauwert 111 bis 115) trennt, markiert nur die Pixel an den Schriftlinienkanten. Von den Hintergrundpixeln werden dadurch weniger als 0,1% als Vordergrund markiert.

Durch Erosion des Hintergrundes werden Lücken in breiten Linien geschlossen. Praktisch erreicht man diese Wirkung durch den Einsatz eines Mittelwertfilters der Größe 9 oder 11 (Radius 4 oder 5) und anschließender Schwellwertfilterung bei ca. 95% (Grauwert 243) . Durch ODER-Verknüpfung werden die Störungen aus dem Bild herausgefiltert.

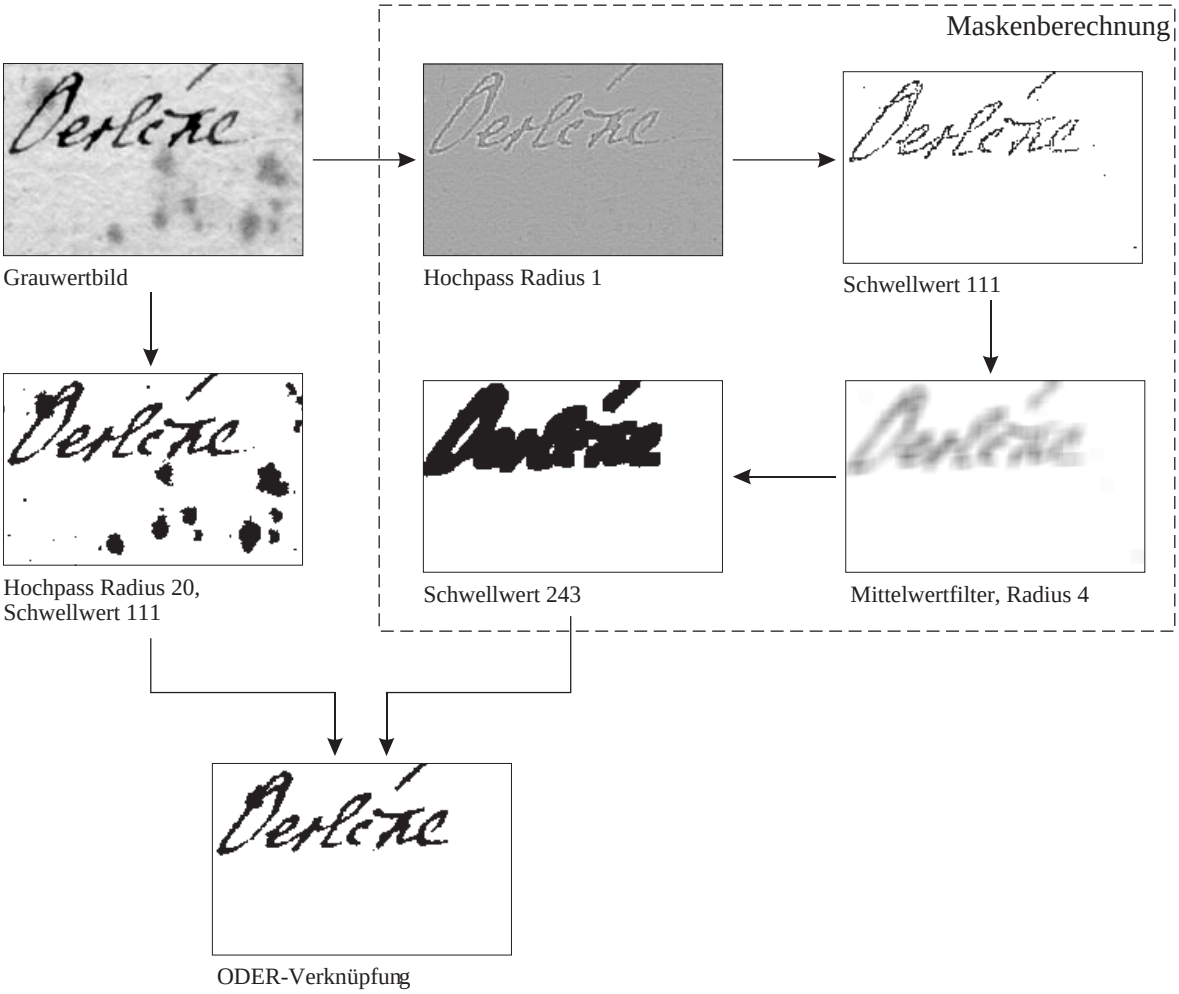


Abbildung 4.5: Erzeugung der Maske zur Beseitigung der Störungen

4.3 Skelette als Kettencode

Das gewonnene Binärbild stellt eine unstrukturierte Matrix aus Binärwerten dar. In diesem Bild müssen nun alle Vordergrundobjekte erfasst und analysiert werden. Miteinander verbundene Vordergrundpixel stellen solche Vordergrundobjekte oder Kontinuen dar.

VECTORY verfügt über optimierte Algorithmen der Skelettierung, die in einer Datenstruktur das Skelett als Kettencode erzeugen (siehe Abschnitt 2.1.1). Dieser Kettencode wird die Basis der weiteren Verarbeitungsschritte sein. Der Grund dieser Repräsentationsform liegt darin, dass der Gegenstand dieser Untersuchungen Linien sind. Im Gegensatz zur binären Pixelmatrix beinhaltet der Kettencode bereits wichtige strukturelle Informationen über den Verlauf der Linien, deren Breite und welche Linie mit welcher verbunden ist. Im Gegensatz zur Kontur, die ähnlich von den Objekten extrahiert werden könnte, repräsentiert das Skelett den Pfad des Schreibwerkzeugs besser.

Auf dieser Datenbasis werden bestimmte Parameter wie Ausdehnung und Position der Objekte erfasst, mit deren Hilfe kleine Störungen entfernt werden können. Genauer gesagt wird die x- und y-Ausdehnung eines Objektes bestimmt. Sobald beide Werte unterhalb einer Schwelle liegen, werden diese Objekte als mögliche Störungen markiert und für die weitere Verarbeitung nicht weiter betrachtet. Es wird hier natürlich auch passieren, dass Teile von Federzügen markiert werden. Das ist jedoch für die Textlinienfindung unerheblich. Lediglich für die Zeilensegmentierung müssen diese Daten wieder betrachtet werden.

Kapitel 5

Finden der Textlinien

Das Finden der Textlinien stellt einen der Hauptpunkte dieser Arbeit dar, da zur Beantwortung der Frage, *was* auf einer Seite mit Kirchenbuchaufzeichnungen geschrieben steht, natürlich die Frage beantwortet werden muss, *wo* sich auf der Seite die Schrift befindet. Außerdem setzen viele Erkennungsverfahren den Verlauf der Textlinien als bekannt voraus. In [GK96] wird ein Wort-Erkenner beschrieben, der die Start- und Endpositionen der Striche relativ zu den Textlinien zur Erkennung nutzt. Dabei werden die aus dem Bild abgeleiteten Eigenschaften mit vorher berechneten Eigenschaftsvektoren aus Lexikoneinträgen verglichen.

Ein grundsätzliches Problem bei allen Erkennungsaufgaben ist der erforderliche Rechenaufwand. Obwohl in dieser Arbeit die Geschwindigkeit des Verfahrens nicht im Mittelpunkt stehen soll, soll der Algorithmus auch in absehbarer Zeit das gewünschte Ergebnis liefern. Bezogen auf das Finden der Textlinien wäre es theoretisch machbar, dass jeder nur denkbare Verlauf einer Textlinie erzeugt wird (z. B. durch Permutation der Parameter wie Koordinatenpunkte). Danach werden bestimmte Bewertungskriterien herangezogen, die es ermöglichen, die schlechteren Alternativen zu verwerfen. Dies entspräche einem kompletten Aufbau eines Entscheidungsbaums, bei dem die Blätter die möglichen Verläufe der Textlinien darstellen. Eine Entscheidung für einen bestimmten Verlauf wird erst zum Ende getroffen. Dieser Ansatz ist auf Grund des exponentiellen Anstiegs des Rechenaufwandes praktisch nicht durchführbar. In der Praxis muss ein Pfad durch diesen Baum gesucht werden, wobei an mehreren Verzweigungen auf Grund einer kleinen Anzahl von Alternativen eine Entscheidung getroffen werden muss. Dieser Ansatz ist der einzig machbare, beinhaltet aber die Tatsache, dass das gefundene Resultat nur mit einer Wahrscheinlichkeit < 1 richtig ist. Es ist also das Ziel, möglichst gute Eigenschaften zu finden, an denen Entscheidungen gefällt werden können, die mit einer hohen Wahrscheinlichkeit richtig sind.

Vor dem Erläutern weiterer Arbeitsschritte ist es zunächst erforderlich, zwei Begriffe zu klären, die im Zusammenhang mit den Merkmalen der Schrift stehen:

Die Schriftlinie ist das Ergebnis, das durch ein entsprechendes Schreibwerkzeug auf das Papier gebracht wurde. Es ist der Bestandteil der Buchstaben und erscheint in der dunklen Vordergrundfarbe.

Die Textlinie definiert die imaginäre Linie, auf denen die Textzeile geschrieben wurde. Dies sind bei Druckschrift vier Linien einer Zeile, die sich aus den oberen und unteren Kanten der verschiedenen Zeichen ergeben.

5.1 Unterschiedliche Lösungswege

5.1.1 Zonenmethode

Auf der Basis des gewonnenen Binärbildes wäre eine Möglichkeit, die Textlinien zu finden, die so genannte Zonen-Methode [SK97]. Hierbei wird die Seite zunächst in mehrere vertikal verlaufende Streifen unterteilt. Für jeden Streifen wird nun in horizontaler Richtung die Häufung der dunklen (Vordergrund-) Pixel untersucht, sodass man eine Aussage über die Position der Textlinien und den weißen Zwischenräumen machen kann. Dieses Prinzip wird häufig für die Erkennung von Druckschrift genutzt, da hier alle notwendigen Parameter, wie Schriftgröße, Zeilenabstand und Zeilen-Neigung, gewonnen werden können. Für die Textlinienfindung auf handschriftlichen Dokumenten oder gar historischen Kirchenbuchaufzeichnungen ist dieses Verfahren jedoch nicht anwendbar, da diese Parameter hier an verschiedenen Stellen einer Seite anders sein können.

5.1.2 Hough-Transformation und Lernalgorithmus

In [PS98] wird eine andere Methode vorgestellt, die ein genaueres Ergebnis für die Rekonstruktion des Verlaufes der Textlinien liefert. Die Grundidee basiert hierbei auf dem Suchen der lokalen Minima der Vordergrundobjekte. Dadurch erhält man Punkte in einer bestimmten räumlichen Anordnung auf der Seite. Es wird weiterhin eine Hough-Transformation und ein Lernalgorithmus vorgeschlagen, um letztendlich den korrekten Verlauf der Basislinie rekonstruieren zu können. Dazu wird zunächst aus einem kleinen Ausschnitt des Bildes eine Ausgangsbasis an Textlinien erzeugt, die über Hough-Transformation gefunden werden. Mit dieser Ausgangsbasis wird das gesamte Bild mit all seinen Minimapunkten nach Textlinien analysiert.

Hier wird der natürliche Lernalgorithmus angewandt. D. h. Elemente, die zu einer bestehenden Gruppe (Cluster) ähnlich sind[1], werden dieser Gruppe zugeordnet. Elemente, die zu keiner bestehenden Gruppe ähnlich genug sind, werden zwischengespeichert. Sobald sich unter den zwischengespeicherten Elementen eine bestimmte Anzahl von zueinander ähnlichen Elementen befinden, wird eine neue Gruppe mit diesen Merkmalen erzeugt.

Die Elemente sind in diesem Fall die Minimumpunkte, deren Eigenschaft darin besteht, auf einer bzw. mehreren Linien zu liegen. Es erfolgt eine Gruppierung von Punkten, die ungefähr auf einer Linie liegen – einer Textlinie.

Prinzipiell ist dies ein vielversprechender Ansatz. Dieser Algorithmus ist dafür ausgelegt, die Basislinie einer Textzeile zu finden. Es wird hierbei aber vorausgesetzt, dass die Textlinien zwar nicht waagerecht, aber gerade verlaufen. Der Verlauf der Textlinien der Kirchenbuchaufzeichnungen weicht mitunter sehr stark von dem einer geraden

[1] Die Elemente weichen in ihren Merkmalen höchstens um eine bestimmte Toleranz ab.

Linie ab. Hierfür ist nicht die Krümmung einer Textzeile verantwortlich. Stattdessen weisen die Textlinien der einzelnen Worte, die in einer Zeile nebeneinander stehen, einen Unterschied in der Richtung sowie einen vertikalen Versatz auf. Hinzu kommt die durch die Aufnahme bedingte Verzerrung, vor allem bei fest gebundenen Büchern.

Ein weiterer Nachteil dieser Methode ist das Nichtbeachten der horizontalen Entfernung der Minimumpunkte. Dieser Algorithmus würde auch Linien finden, deren Minimumpunkte sehr weit auseinander liegen. Dass solche Punkte tatsächlich eine Textlinie bilden, ist eher unwahrscheinlich.

5.1.3 Lokale Referenzlinien

Der in meinen Augen am vielversprechenste Ansatz berücksichtigt die Eigenschaft von handschriftlichen Texten, dass jedes Wort eine eigene Basislinie ausbildet und eine Textzeile sich sodann aus diesen so genannten lokalen Referenzlinien zusammensetzt [MG99]. Hier werden relative Bedingungen von Richtung und Entfernung zwischen zwei lokalen Minimumpunkten genutzt, um Basislinien zu finden.

Iterativ wird hierbei von links nach rechts versucht, eine lokale Referenzlinie durch einen weiteren Minimumpunkt zu verlängern. Dabei wird auf ein so genanntes *„Gutes Kontinuitätskriterium“* geachtet. D. h. zum Einen muss der maximale x-Abstand der Minima zueinander eingehalten werden, zum Anderen darf der Anstieg der Linie, die den letzten Minimumpunkt mit dem neuen Minimumpunkt verbindet, nicht größer sein als eine bestimmte Schwelle. Diese Grundidee Idee wird aufgegriffen und im folgenden Abschnitt beschrieben.

Eine wichtige einschränkende Eigenschaft dieser Methode ist die Grundannahme, dass nicht mehrere Zeilen übereinander stehen. Hinzu kommt, dass die Verteilung der Minimumpunkte sich bei Kirchenbuchaufzeichnungen nicht so eindeutig darstellt, wie in den dort angeführten Beispielen.

5.2 Formulierung des Algorithmus

Der hier vorgestellte Algorithmus basiert auf dem Skelett der Objekte der Kirchenbuchseite. Dieses Skelett wird von VECTORY erzeugt und steht in einer Datenstruktur aus Kettencode-Objekten[2] zur Verfügung.

Es wird davon ausgegangen, dass die Richtung der zu findenden Textlinien ungefähr horizontal verläuft. Eine eventuelle Neigung, die entweder durch den Schreiber selber oder durch das Digitalisieren entstanden ist, wird ermittelt und durch Rotation korrigiert.

[2] siehe Abschnitt 2.1.1

Da, wie in Abschnitt 3.2 beschrieben, die obere und untere der vier Textlinien nicht ausgeprägt sind, werden für jede Textzeile eines Textblockes die Basis- und die Mittellinie ermittelt.

Der Algorithmus unterteilt sich in sieben Arbeitsschritte:

1. Finden der lokalen Minimum- und Maximumpunkte
2. Finden aller möglichen Basisliniensegmente (BLS) unter einem großen Winkelbereich
3. Errechnen der Rotation und deren Korrektur
4. Erneutes Finden der BLS unter kleinerem Winkelbereich
5. Bewertung und Entfernen der schlechten BLS-Alternativen
6. Verbinden der BLS zu den fertigen Basislinien
7. Finden der Mittellinien auf Grundlage der Basislinien

5.2.1 Finden der lokalen Minimum- und Maximumpunkte

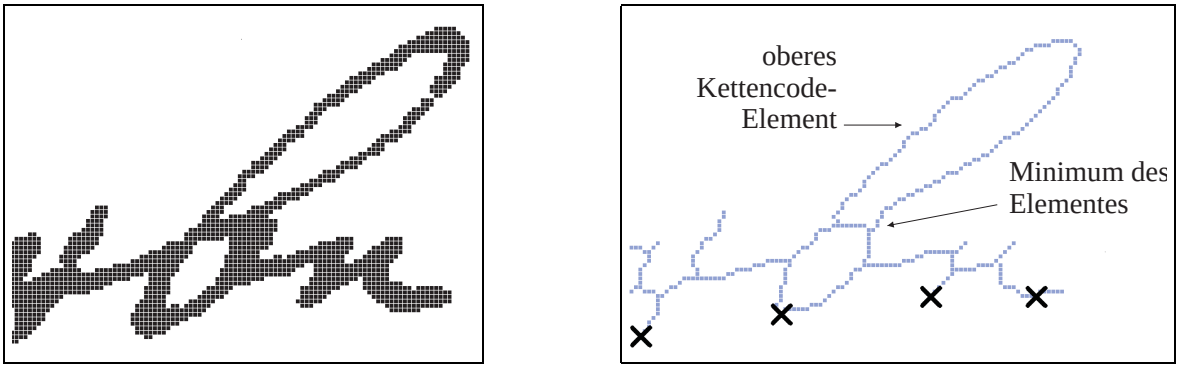
Das Ergebnis der Skelettierung von VECTORY ist eine Datenstruktur mit Kettencode-Elementen. Ein solches Element kann an beiden Enden jeweils mit zwei anderen Kettencode-Elementen verbunden sein.[3] Elemente, die so miteinander verbunden sind, bilden ein Vordergrundobjekt oder *Kontinuum*. Wie gewinnt man daraus eine Menge von lokalen Minimum- bzw. Maximumpunkten?

Zunächst einmal werden alle die Kettencode-Elemente ignoriert, die zu Kontinuen gehören, deren x- und y-Ausdehnung unterhalb einer bestimmen Schwelle liegt. Diese Kleinobjekte resultieren entweder aus einer Unterbrechung eines Schriftzuges oder sie haben ihren Ursprung in Zeichen der Interpunktion, in Punkten von Umlauten o. Ä. – wie in Abschnitt 3.1 beschrieben. In allen Fällen ist die Wahrscheinlichkeit gering, dass diese Kleinobjekte zur Textlinienfindung beitragen.

Der folgende Ablauf ist für das Finden der Minima sowie für das Finden der Maxima identisch, daher wird im Folgenden der Begriff *Extremum* für beide Fälle benutzt.

Da die Kettencode-Elemente meist relativ kurz sind und ihre Krümmungsrichtung nicht ändern, ist es für das Finden eines lokalen Extremums ausreichend, jeweilige Extremum des Kettencode-Elementes zu bestimmen. Allerdings würde diese Methode neben den gewünschten auch viele andere Punkte erzeugen, die keine lokalen Extrema sind. Um das zu verhindern, wird kontrolliert, ob das Extremum an einem Endpunkt liegt, an dem sich weitere Kettencode-Elemente anschließen. Ist dies der Fall, so wird dieser Punkt nicht als Extremum markiert.

[3] In seltenen Fällen sind es drei Elemente.

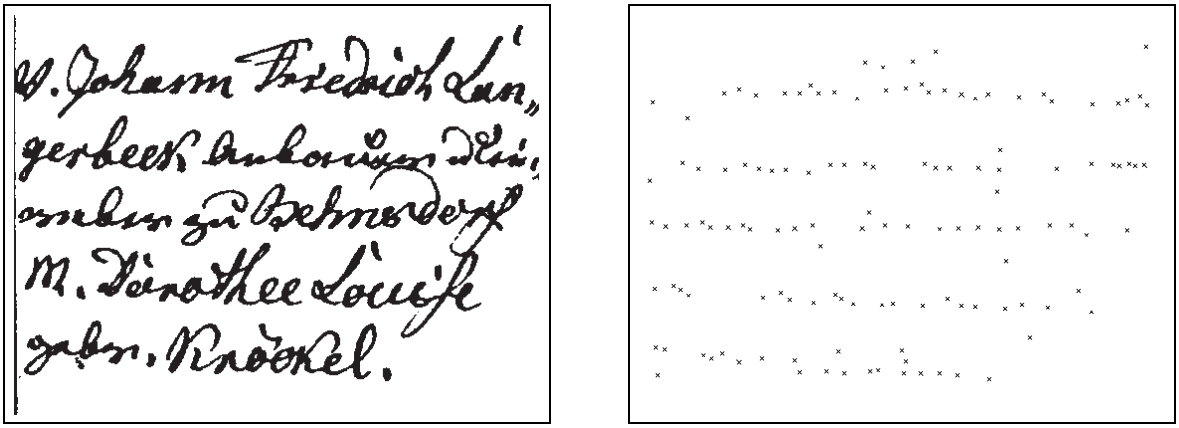


(a) Pixelbild

(b) Kettencode mit markierten lokalen Minima

Abbildung 5.1: lokale Minima aus Kettencode generieren

In Abbildung 5.1(b) ist eine der oberen Kettencode-Elemente gekennzeichnet. Es ist zu erkennen, dass dessen Minimum an einem Ende liegt, an dem sich weitere Kettencode-Elemente anschließen. Es ist nicht als lokales Minimum des Kontinuums markiert worden.



(a) Pixelbild

(b) gefundene Minimumpunkte

Abbildung 5.2: Textblock einer Tabelle von 1838 und seine Minimumpunkten

Dieses Verfahren schließt zwar nicht aus, dass Punkte markiert werden, die für die Textlinienfindung unerheblich sind, es ist jedoch ausreichend, um an der potentiellen Basislinie eine Dichte von Minimumpunkten und der potentiellen Mittellinie eine Dichte von Maximumpunkten zu erzeugen, mit der im weiteren Verlauf diese Linien gefunden werden können.

5.2.2 Finden der Liniensegmente unter einem großen Winkelbereich

Um die Rotation des Textblockes zu ermitteln, wird unter einem größeren Winkelbereich nach Liniensegmenten gesucht. Dazu wird die Anordnung der Minimumpunkte[4] analysiert. Wenn eine bestimmte Mindestanzahl von Punkten ungefähr auf einer Linie liegt und dabei einen bestimmten maximalen Abstand zueinander einhält, dann bilden diese Minimumpunkte ein potentielles Liniensegment[5]. Ich habe den zu untersuchenden Winkelbereich auf $\pm 20°$ zur Horizontalen festgesetzt, um unnötigen Rechenaufwand zu vermeiden. Eine größere Abweichung ist von einem nach Augenmaß ausgerichteten Dokument nicht zu erwarten.

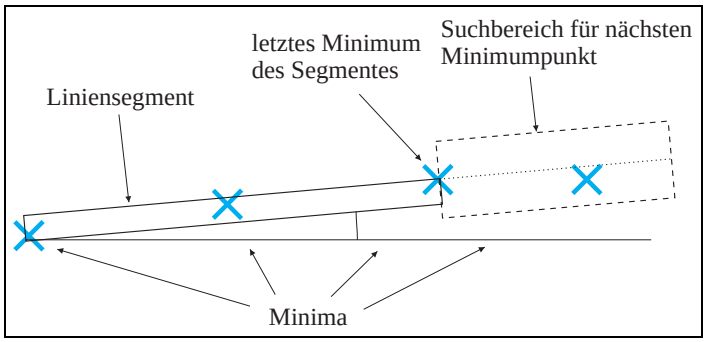


Abbildung 5.3: Konstruieren der Basisliniensegmente

Für jeden Winkel in diesem Winkelbereich werden die unter diesem Winkel nach x sortierten Minimumpunkte von links nach rechts durchlaufen. Für jeden Minimumpunkt (MP), der unter diesem Winkel noch keinem Segment angehört, wird versucht, ein BLS zu erzeugen, indem unter dem aktuellen Winkel nach einem weiteren MP gesucht wird, der rechts vom aktuellen MP liegt. D. h. es wird nach dem nächsten MP gesucht, der innerhalb eines kleinen Rechtecks liegt (siehe Abbildung 5.3). Dabei muss zusätzlich eine Höhenbeschränkung des BLS eingehalten werden. D. h. der größte vertikale Abstand (unter dem aktuellen Winkel) zwischen zwei Minimumpunkten, die zum BLS gehören, darf einen bestimmten Grenzwert nicht überschreiten. Dieser Schritt wird mit dem jeweils letzten MP solange wiederholt, bis kein weiterer MP auf diese Weise gefunden wird und das BLS nicht weiter verlängert werden kann.

Für jeden Winkel des untersuchten Winkelbereiches existiert nun eine Menge von BLS mit unterschiedlichen Längen und einer unterschiedlichen Anzahl von Minimumpunkten.

[4] Da die Basislinie die bessere Ausprägung als die Mittellinie hat, werden die *Minimumpunkte* betrachtet und somit *Basisliniensegmente* erzeugt

[5] Diese Werte werden als Parameter gesetzt (siehe Abschnitt 7.2)

5.2.3 Rotation der Textzeilen bestimmen

Winkel und Länge[6] sind die Eigenschaften eines Segmentes, auf die es bei der Bestimmung der Rotation[7] des Textblockes ankommt. Segmente mit kurzer Länge entstehen vor allem quer zur eigentlichen Textlinienrichtung und liegen dabei innerhalb einer Zeile oder verlaufen über zwei oder drei Zeilen hinweg. Dass lange Segmente ungleich zur Textlinienrichtung entstehen, ist eher unwahrscheinlich. Je länger ein BLS ist, umso wahrscheinlicher ist dessen Verlauf in Textrichtung. Um die durchschnittliche Richtung zu bestimmen, werden daher nur die längeren Segmente betrachtet. Genauer gesagt kommen nur die Segmente in Frage, deren Länge mindestens die Hälfte der maximalen Länge beträgt.

Ein Segment, das aus vielen Minimumpunkten besteht, hat ein größeres Gewicht als ein Segment mit wenigen Minimumpunkten. Deshalb wird die durchschnittliche Richtung nicht über die Anzahl der Segmente, sondern über die Anzahl der Minimumpunkte der entsprechenden Segmente ermittelt. Damit ein langes Segment ein größeres Gewicht erhält als mehrere kleine, wird zusätzlich die Anzahl der Minimumpunkte quadriert.

Sei $S_{1/2}$ die Segment-Menge der n Segmente s_i mit $1 \leq i \leq n$, deren Länge mindestens die Hälfte der Maximallänge beträgt. Weiterhin besitzt jedes Segment s_i einen Winkel w_i und eine bestimmte Anzahl m_i von Minimumpunkten. Der durchschnittliche Richtungswert $\bar{w}$ errechnet sich folgendermaßen:

$$\bar{w} = \frac{\sum_{i=1}^{n} (w_i \cdot m_i^2)}{\sum_{i=1}^{n} m_i^2} \tag{5.1}$$

Die Koordinaten der ermittelten Extrempunkte werden nun um $-\bar{w}$ gedreht. Damit schwanken die Winkelwerte der Textliniensegmente um 0°, und der Winkelbereich für die im Folgenden betrachteten Textliniensegmente kann entsprechend verkleinert werden.

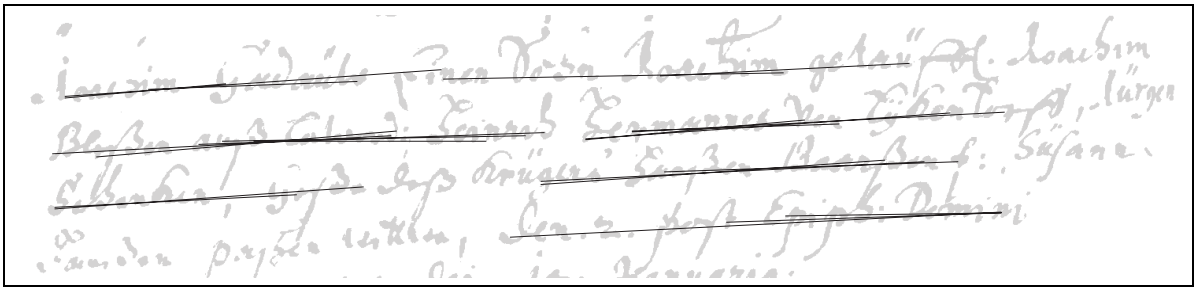


Abbildung 5.4: die langen Segmente, aus denen die Rotation bestimmt wird

[6] Alternativ zur Länge kann auch die Zahl Minimumpunkte verwendet werden, da sich diese beiden Eigenschaften ungefähr proportional zueinander verhalten.

[7] Der Begriff der *Rotation* bezieht sich auf den Verlauf der Textzeilen und darf nicht mit der *Neigung* der Schrift – wie z. B. bei kursiver Schrift – verwechselt werden.

In Abbildung 5.4 ist ein Textblock aus dem Jahre 1649 abgebildet, in den die 22 langen Segmente eingezeichnet wurden, die zur Bestimmung der Rotation betrachtet wurden. Sie beträgt $+2°$. Zuvor wurde in einem Bereich von $\pm 20°$ nach Segmenten gesucht.

5.2.4 Finden der Textliniensegmente

Es wird jetzt noch einmal der Schritt aus Abschnitt 5.2.2 wiederholt. Der entscheidende Unterschied ist nun der kleinere Winkelbereich, der jetzt bei ungefähr $\pm 6°$ liegt. Dieser Bereich ist ausreichend, da jetzt nur noch die Richtungsschwankungen innerhalb der Zeilen erfasst werden müssen. Durch Beobachtungen und Tests habe ich festgestellt, dass sich diese Werte nie mehr als um 8 bis 9° voneinander unterscheiden. Damit ist die Toleranz von 12° ausreichend.

Als Ergebnis dieses Schritts erhält man ein Zahl von möglichen Basisliniensegmenten, die zwar geringer ist als beim ersten Durchlauf mit großem Winkelbereich, aber immer noch viel zu groß, um an dieser Stelle auf den Verlauf der Basislinien zu schließen zu können.

Ein MP gehört meist zu mehreren Basisliniensegmenten, die unter unterschiedlichen Winkeln durch diesen Punkt verlaufen. Deshalb muss die Zahl der erzeugten Basisliniensegmente reduziert werden.

5.2.5 Entfernen der schlechten BLS-Alternativen

Ein Minimumpunkt soll nur dann zu zwei Basisliniensegmenten gehören, wenn das eine BLS an dieser Stelle endet und das andere beginnt.

Die Reduzierung der Segmentzahl erfolgt in drei Stufen:

1. Entfernen von kurzen Basisliniensegmenten
2. Entfernen von Basisliniensegmenten, die in anderen Segmenten vollständig enthalten sind und
3. Entfernen von Basisliniensegmenten, die sich mit anderen Segmenten kreuzen.

Das Ziel dieser Schritte ist es, nur die längsten Segmente zu behalten, die hoffentlich den Verlauf der Basislinien repräsentieren.

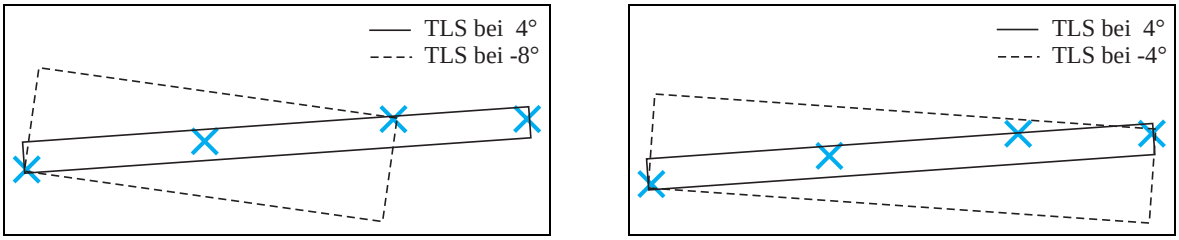
Entfernen von kurzen Basisliniensegmenten

Alle zu kurzen Basisliniensegmente werden gelöscht. Abbildung 5.2(b) zeigt ein Beispiel einer Minimumpunktverteilung. Hier kann man erkennen, dass auch kurze Liniensegmente möglich sind, die mit dem Textlinienverlauf nichts zu tun haben. Wo setzt man die Grenze für zu kurze Segmente? Segmente mit zwei Minimumpunkten treten sehr

zahlreich auf und besitzen keinerlei Aussagekraft, denn sobald zwei Minimumpunkte den maximalen Abstand einhalten, entsteht solch ein Segment. Je nach Schriftart können einerseits Segmente mit drei Minimumpunkten, vor allem bei kurzen Worten, für die Linienfindung entscheidend sein. Andererseits können bei einer anderen Schrift diese Segmente zum größten Teil an „falschen“ Stellen entstehen, wobei die „echten“ Textliniensegmente je nach Zeilenlänge und Schriftqualität bis zu 14 Minimumpunkte miteinander verbinden.

So werden in diesem Schritt zumindest alle BLS mit zwei Minimumpunkten entfernt.

Entfernen von Basisliniensegmenten, die in anderen Segmenten vollständig enthalten sind



(a) ein BLS ist in einem anderen enthalten

(b) zwei identische Basisliniensegmente

Abbildung 5.5: Basisliniensegmente, die in anderen Basisliniensegmenten enthalten sind

Für ein BLS kann unter einem benachbarten, schlechteren Winkel ein zweites BLS existieren, das entweder kürzer und damit in dem ersten BLS enthalten ist oder das auf Basis der Minimumpunkte identisch ist, aber einen schlechteren Qualitätswert besitzt, da die durchschnittliche vertikale Abweichung der Minimumpunkte größer ist. In beiden Fällen wird dieses zweite BLS gelöscht.

Entfernen von Basisliniensegmenten, die sich mit anderen Segmenten kreuzen

Es existieren immer noch einige Fälle, in denen ein MP zu mehreren Basisliniensegmenten gehört. In diesem Schritt wird untersucht, ob in solchen Fällen die horizontalen Grenzen des einen Segmentes innerhalb des anderen Segmentes liegen. Ist dies der Fall, wird das kürzere Segment gelöscht.

5.2.6 Verschmelzen der Basisliniensegmente

Zu behandeln sind nun noch Überschneidungen von Basisliniensegmenten, die unterschiedliche horizontale Bereiche abdecken. Hier ist es nicht möglich, ein BLS einfach

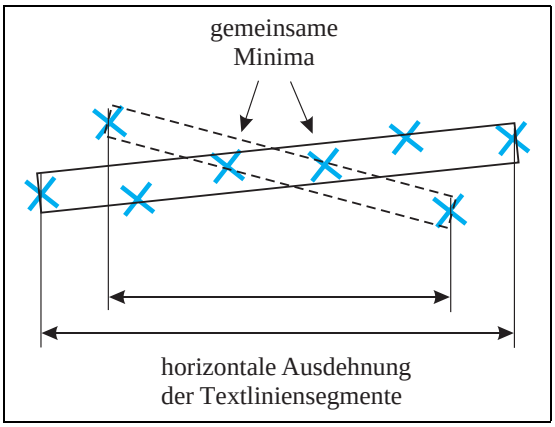
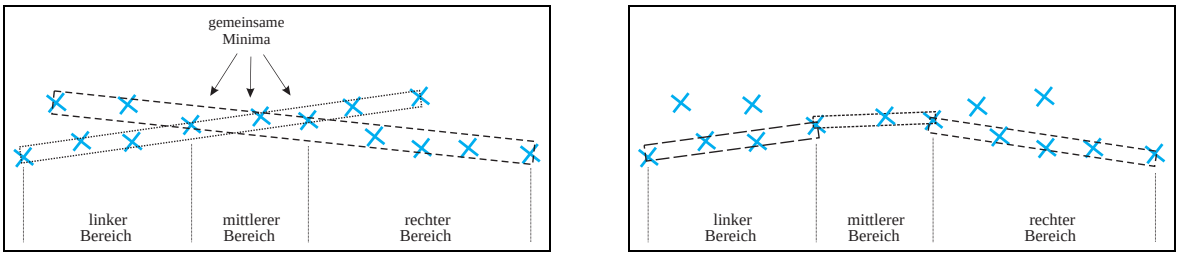


Abbildung 5.6: sich überkreuzende Segmente mit gemeinsamen Minima



(a) Einteilung in drei Bereiche

(b) drei neue Basisliniensegmente

Abbildung 5.7: sich kreuzende Basisliniensegmente mit unterschiedlichen Bereichen

zu löschen. Aus diesen zwei sich kreuzenden Segmenten werden zwei bis drei neue Segmente erzeugt, die hintereinander liegen.

Durch die Überschneidung entstehen bis zu drei Abschnitte. Der Mittelabschnitt wird durch die Minimumpunkte erzeugt, die in beiden Basisliniensegmenten enthalten sind. Daraus wird ein neues BLS erzeugt. Im linken und rechten Abschnitt haben die Basisliniensegmente einen eigenen Verlauf. Aus jeweils einem Verlauf auf einer Seite wird ein neues BLS erzeugt. Welcher Verlauf gewählt wird, entscheidet dessen höhere Anzahl der Minimumpunkte. Ist sie identisch, wird der tiefer liegende Verlauf gewählt.

Als Ergebnis erhält man maximal drei neue Basisliniensegmente, während die „alten“ gelöscht werden. Zwei neue Basisliniensegmente entstehen, wenn es nur einen gemeinsamen Minimumpunkt gibt oder wenn auf einer Seite kein getrennter Verlauf existiert. Nur ein neues BLS entsteht, wenn die beiden Basisliniensegmente nur ihren letzten Minimumpunkt gemeinsam haben.

5.2.7 Verbinden der Segmente zu den fertigen Basislinien

Nun bestehen keine mehrdeutigen Überschneidungen von Basisliniensegmenten mehr. Es werden die bestehenden Basisliniensegmente nun zu kompletten Basislinien kombiniert. Dabei werden Textlinienobjekte (TLO) erzeugt, die aus Basisliniensegmenten bestehen, die jeweils an ihren Enden miteinander verbunden sind.

Dies läuft ähnlich dem Verlängern der Basisliniensegmente durch ein weiteren Minimumpunkt ab. Die horizontal sortierte Liste der Basisliniensegmente wird von links nach recht durchlaufen. Ist ein BLS noch kein Bestandteil eines Textlinienobjekts, dann wird ein neues Textlinienobjekt mit diesem BLS erzeugt. Ausgehend vom letzten (rechten) Minimumpunkt des letzten BLS des Textlinienobjekts, wird nach Basisliniensegmenten gesucht, die mit ihren Minimumpunkten in einem rechteckigen Bereich liegen, der größer dimensioniert ist als beim Erweitern der Basisliniensegmente. Werden mehrere Segmente gefunden, wird eine Entscheidung durch des Vergleichen der Positionen der Minimumpunkte getroffen.

Seien (u, v) die Koordinaten des letzten Minimumpunktes und (x, y) die Koordinaten des Minimumpunktes des zu bewertenden Basisliniensegments. Durch Tests hat sich die Bestimmung des Qualitätswertes Q durch folgende Formel[8] bewährt:

$$Q = -10 \cdot \mathrm{abs}(y - v) - (x - u) \tag{5.2}$$

Drei Eigenschaften spielen hierbei eine Rolle:

$-\mathrm{abs}(y - v)$	Je kleiner der Höhenunterschied zwischen Endpunkt des Textlinienobjekts und dem jeweiligen Minimumpunkt des Segments ist, umso höher ist der Qualitätswert.
$-(x - u)$	Je horizontal näher ein Segment liegt, umso höher ist der Qualitätswert.

Der Faktor 10 bewirkt, dass ein Segment auch dann gewählt wird, wenn es horizontal weiter weg liegt als ein alternatives Segment, es aber die Basislinie optimal weiterführt. Diese Formel stellt vermutlich noch nicht das Optimum der Bewertung dar und bietet Möglichkeiten der Verbesserung des Algorithmus.

Wurde ein Folgesegment gefunden, wird – falls sich dieses Segment nicht direkt anschließt – ein BLS erzeugt, das den Endpunkt des TLO mit dem Anfangspunkt des BLS verbindet. Das TLO wird dann um beide Segmente erweitert.

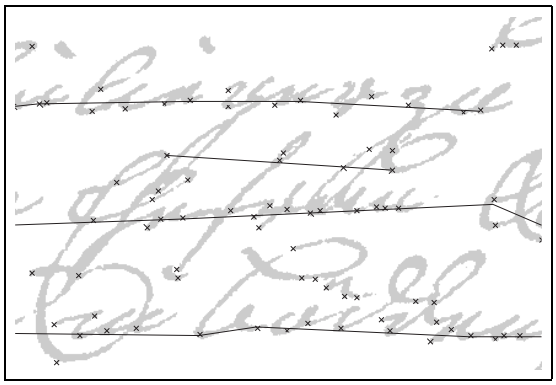


Abbildung 5.8: falsche Basislinien müssen entfernt werden

5.2.8 Entfernen von falschen Basislinien

Nachdem alle Basislinien erzeugt wurden, lassen sich Parameter der Zeilen wie durchschnittliche Zeilenlänge und Zeilenabstand ermitteln. Mit diesen Durchschnittswerten lassen sich falsche Basislinien, die aus ungünstigen Minimumpunkten entstanden sind, finden und löschen. Im Detail: Eine Basislinie wird dann gelöscht, wenn sie kürzer als eine bestimmte Prozentzahl der Durchschnittslänge ist und wenn der Abstand zur nächsten langen Textzeile kleiner als ein bestimmter Prozentwert des durchschnittlichen Zeilenabstandes ist. Die beiden Werte werden zuvor durch Parameter gesetzt und bewegen sich im Bereich von 50%. Die Konjunktion dieser beiden Bedingungen stellt sicher, dass wirklich nur falsche Basislinien entfernt werden.

5.2.9 Verbinden von unterbrochenen Basislinien

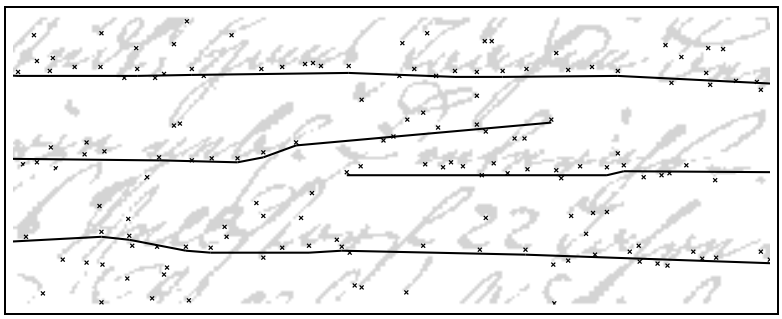


Abbildung 5.9: unterbrochene Basislinie durch ungünstige Konstellation

Durch ungünstige Konstellationen von horizontalem Versatz nebeneinander stehender Worte einer Zeile und der Position von Minimumpunkten, die nicht auf der Basislinie liegen, kann es passieren, dass in *einer* Textzeile zwei (oder gar mehrere) Basislinien gefunden werden (siehe Abbildung 5.9). Diese gilt es zu verbinden.

[8] Der Koordinatenursprung liegt bei dieser Betrachtung oben links.

Dazu müssen diese Stellen zunächst gefunden werden. Die nach y sortierten Basislinien werden dahingehend überprüft, ob sie sich horizontal mit ihrem Nachfolger überlappen. Ist dies der Fall und ist der durchschnittliche Abstand im Überlappungsbereich kleiner als 50% des durchschnittlichen Zeilenabstandes des untersuchten Textblockes, dann wird diese Basislinie mit ihrem Nachfolger verbunden.

Ähnlich der Situation in Abschnitt 5.2.6, der sich mit dem Verschmelzen von sich überkreuzenden Segmenten beschäftigt, bestehen auch hier wieder drei horizontale Bereiche. Es gibt eine linke und eine rechte Basislinie, die sich in einem horizontalen Bereich überlappen. Der Verlauf der neuen Basislinie wird links vom Überlappungsbereich von der linken Basislinie bestimmt und rechts davon von der rechten Basislinie. Für den Überlappungsbereich wird der bessere Verlauf gewählt. Dazu wird die Zahl der Minimumpunkte in diesem Bereich von beiden Basislinien verglichen.

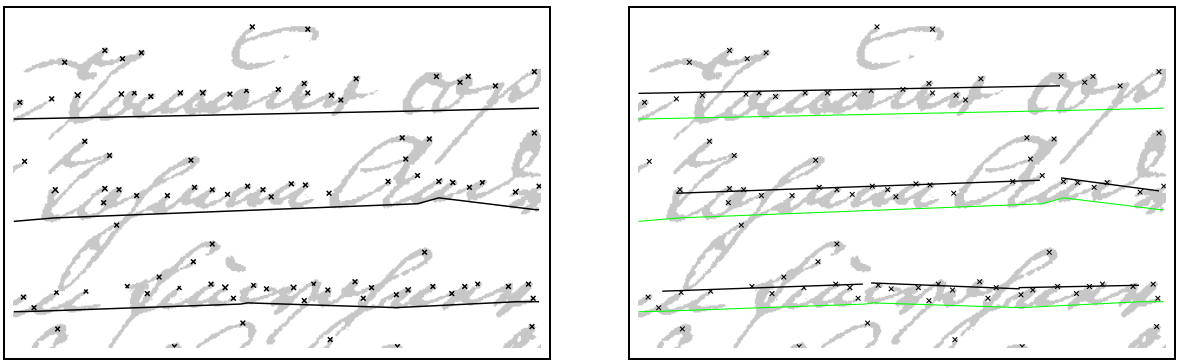
Das Ergebnis ist eine durchgehende Basislinie vom linken Ende der linken Basislinie bis zum rechten Ende der rechten Basislinie.

5.2.10 Finden der Mittellinie

Neben der Basislinie existieren theoretisch noch drei weitere Textlinien, die durch die unterschiedlich großen Buchstaben und Buchstaben mit Unterlängen gebildet werden. Je mehr man über den Verlauf dieser Linien in Erfahrung bringen kann, umso besser können Zeilensegmentierungen durchgeführt werden und umso bessere Ergebnisse liefert die Erkennung. Es hängt hierbei sehr stark von der Schrift ab, wie deutlich sich diese Linien ausprägen und ob sie gefunden werden können. Die mittlere Textlinie, die von den Oberkanten der kleinen Zeichen gebildet wird, hat nach der Basislinie im Allgemeinen die zweitbeste Ausprägung.

Eine Möglichkeit, diese Linie zu finden, wäre die Verwendung des gleichen Algorithmus, der zur Findung der Basislinien genutzt wird, diesmal jedoch auf Grundlage der lokalen Maxima. Da es aber sehr häufig der Fall ist, dass sich diese zweite Textlinie im Vergleich zur Basislinie nicht so stark herausbildet und der Verlauf der Basislinie bereits bekannt ist, nutzte ich dieses Wissen, um die mittlere Textlinie zu finden. Der Algorithmus basiert auf folgenden Annahmen:

- In der unmittelbaren Umgebung der mittleren Textlinie ist eine Häufung der lokalen Maxima zu verzeichnen.
- Die mittlere Textlinie verläuft nahezu parallel zur entsprechenden Basislinie und zwischen der aktuellen Basislinie und der darüber liegenden Basislinie.
- Im Zweifelsfall ist der Abstand zwischen Basis- und Mittellinie innerhalb einer Zeile ungefähr konstant.



(a) Basislinie und Maximumpunkte

(b) gefundene Mittelliniensegmente

Abbildung 5.10: Finden der Mittelliniensegmente

Formulierung des Algorithmus

Für jedes Segment einer Basislinie, das eine gewisse Mindestlänge besitzt, wird ein entsprechendes Mittelliniensegment (MLS) gesucht, indem eine Linie bestimmt wird, die folgende Eigenschaften aufweist:

- Sie liegt zwischen dem BLS und der darüber liegender Textzeile.
- Sie verläuft parallel zum BLS .
- Sie überdeckt mit ihrem Toleranzbereich die meisten lokalen Maxima.

Ist die Zahl der Maximumpunkte, die von der gefundenen Linie überdeckt werden, zu klein, wird die Position dieses Segmentes aus dem durchschnittlichen Abstand der anderen Mittelliniensegmente zum jeweiligen BLS der Textzeile bestimmt.

Als Letztes werden die MLS zu einer kompletten Mittellinie verbunden.

Sollte zu einer existierenden Basislinie keine Mittellinie gefunden werden, d. h. zu keinem Segment der Basislinie wurde ein Mittelliniensegment mit einer ausreichenden Zahl von Maximumpunkten gefunden, dann handelt es sich mit hoher Wahrscheinlichkeit um einen falschen Verlauf und die Basislinie wird gelöscht.

5.3 Verlängerung der Textlinien

Bevor die Zeilen voneinander getrennt werden können, müssen die gefundenen Textlinien noch bis zum Rand des untersuchten Bereiches erweitert werden. Dies liegt daran, dass der Verlauf einer Textlinie einer Zeile den Zeilenbereich[9] der benachbarten Zeile mitbestimmt. Ist die aktuelle Zeile relativ kurz, dann gibt es rechts oder links

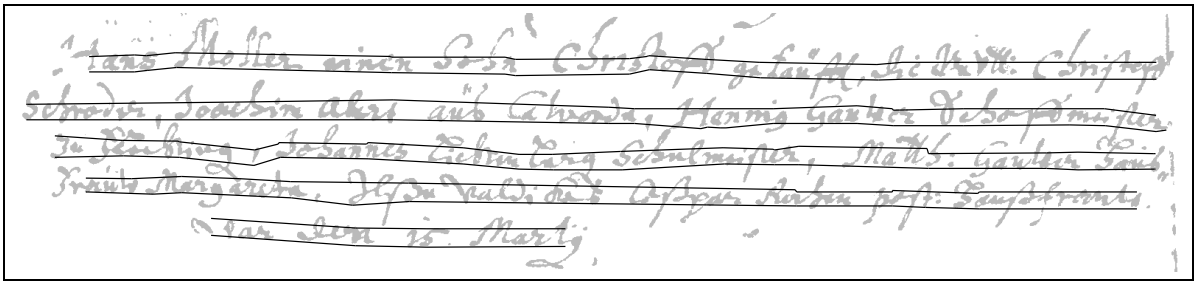
[9] siehe Abschnitt 6.1

von der Zeile einen undefinierten Bereich. Um für eine benachbarte längere Zeile den Zeilenbereich bestimmen zu können, darf es keinen undefinierten Bereich geben. Um dies sicherzustellen, werden alle Textlinien bis zum Rand des untersuchten Bereiches verlängert.

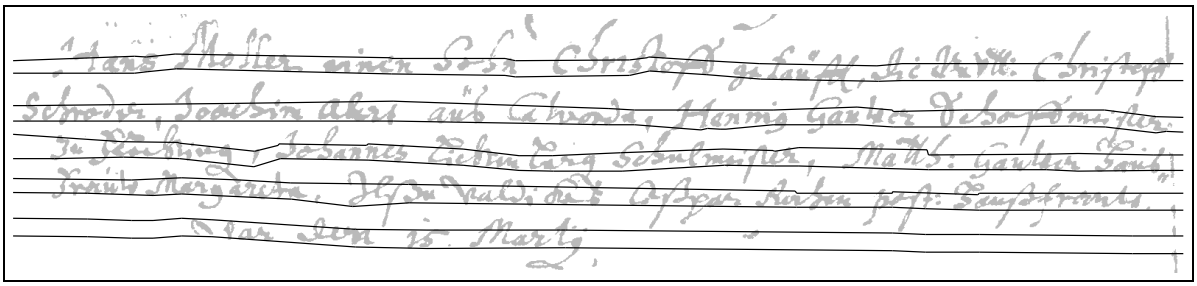
Dabei wird unterschieden, ob es sich um eine lange oder kurze Textlinie handelt. Dabei wird wie in Abschnitt 5.2.8 geprüft, ob eine Linie länger oder kürzer ist als ein bestimmter Prozentwert der Durchschnittslänge der Textlinien.

Zunächst werden alle langen Textlinien verlängert. Dazu wird für jeweils jede Seite der Anstieg zwischen Endpunkt und Linienmitte ermittelt, um in diesem Winkel die Textlinie zum Rand zu verlängern.

Danach werden alle kurzen Textlinien verlängert. Dazu wird der Verlauf bis zum Rand aus dem Verlauf der darüber und der darunter liegenden langen Textlinien konstruiert. Dabei wird der durchschnittliche Abstand zwischen der zu verlängernden und der langen Nachbarlinie bestimmt, um in diesem Abstand die kurze Textlinie bis zum Rand fortzuführen. Fehlt bei einer kurzen Textlinie sowohl die obere als auch die untere lange Linie (d. h. es befinden sich nur kurze Linien in der Nachbarschaft), wird diese Linie zunächst ausgelassen und in einem erneuten Durchlauf verlängert. Dieser Fall tritt beispielsweise dann auf, wenn ein Textblock mit zwei kurzen Textzeilen beginnt.



(a) ohne Verlängerung



(b) nach Verlängerung

Abbildung 5.11: 5. Textblock der Seite 111 aus dem Jahre 1649 mit Textlinien

5.4 Zusammenfassung

Ausgehend von der von VECTORY erzeugten Kettencode-Struktur des Skelettes wurde in diesem Kapitel ein Weg aufgezeigt, um in einem Textfeld für jede dort befindliche Zeile den Verlauf der Basis- und der Mittellinie zu rekonstruieren.

Nachdem für alle interessanten Kontinuen[10] die Minimum- und Maximumpunkte bestimmt wurden, entstehen an den Stellen Basisliniensegmente, an denen unter einem bestimmten Winkel Minimumpunkte auf einer Geraden liegen.

Durch die Analyse der erzeugten BLS wird die Rotationsrichtung des Textblockes bestimmt. Dadurch kann der zu untersuchende Winkelbereich eingeschränkt werden. Die Zahl der BLS, die sich in diesem Bereich befinden, wird in mehreren Schritten reduziert, bis es möglich ist, daraus durchgängige Basislinien zu erzeugen. Dazu werden hintereinander liegende BLS miteinander zu einer Basislinie verbunden.

Ausgehend vom Verlauf der Basislinien wird anschließend der Verlauf der Mittellinie rekonstruiert, die ungefähr parallel zur Basislinie der Zeile verläuft. Dazu werden die Häufungen der Maximumpunkte gesucht.

Abschließend werden die Textlinien bis zum Rand des Textblockes verlängert, um den Ausgangspunkt für die Zeilensegmentierung zu schaffen.

Das Ergebnis ist eine vollständige vertikale Einteilung des untersuchten Bereiches. Es stehen nun alle Informationen für einen Zeilensegmentierung zur Verfügung.

[10] Es werden nur Kontinuen betrachtet, die größer als die Mindestgröße sind.

Kapitel 6

Zeilensegmentierung

Da für jede Zeile des untersuchten Textblockes die Basis- und Mittellinie bekannt ist, können die Zeilen nun voneinander isoliert werden.

Die digitalisierte und binarisierte Textseite besteht aus Kontinuen, d. h. aus Vordergrundpixeln, die direkt oder über andere Vordergrundpixel miteinander verbunden sind. Die horizontale und vertikale Ausdehnung und Position dieser Kontinuen relativ zu den ermittelten Textlinien gibt darüber Auskunft, zu welcher Zeile das Kontinuum gehört bzw. ob es ein Problem des *Ineinanderragens* oder des *Berührens* von Zeilen gibt.

Die Kontinuen, die nicht von den Textlinien mehrerer Zeilen geschnitten werden, müssen den einzelnen Zeilen zugeordnet werden. Das wird im folgenden Abschnitt beschrieben.

Ist die vertikale Ausdehnung eines Kontinuums so groß, dass es sich über mehrere Zeilen erstreckt, dann liegt eine Berührung der Zeilen vor und das Kontinuum muss an bestimmten Stellen getrennt werden. Dies wird im darauf folgenden Abschnitt aufgezeigt.

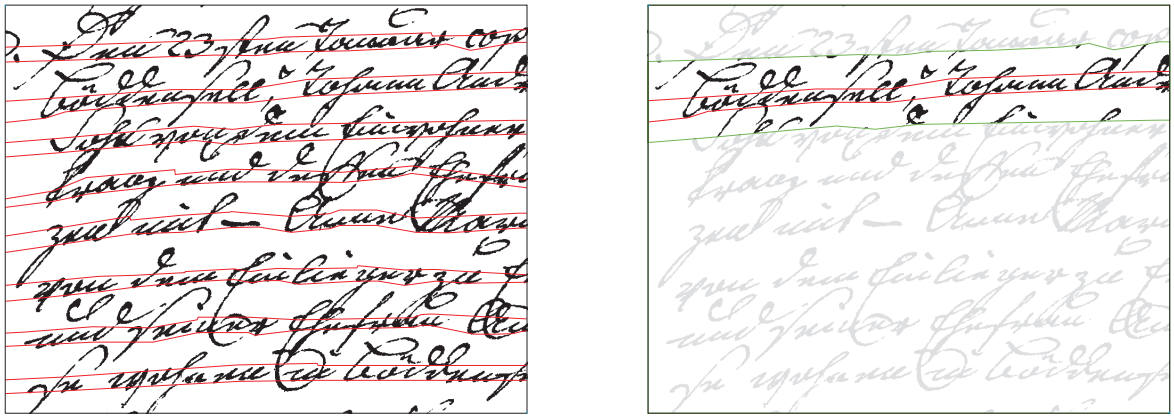
Durch die Analyse von bestimmten Verbindungssituationen zweier Zeilen lassen sich in einigen Fällen die Trennungsergebnisse verbessern. Dies wird im dritten Abschnitt anhand der Kreuzungspunkt-Analyse aufgezeigt.

Das Ergebnis ist eine Repräsentation einer jeden Zeile durch die zu dieser Zeile gehörenden Objekte. Um letztendlich für jede Zeile wieder ein Rasterbild zu erhalten, sind Schritte notwendig, die im letzten Abschnitt dieses Kapitels beschrieben werden.

6.1 Zuordnung der Kontinuen zu den Zeilen

Durch den bekannten Verlauf der Textlinien lässt sich für eine Zeile der interessante Bereich des Bildes horizontal eingrenzen, der für die Erkennung wichtig ist. Dieser *Zeilenbereich* wird nach oben durch die Basislinie der darüber liegenden Zeile und nach unten durch die Mittellinie der darunter liegenden Zeile begrenzt. Alle für die Erkennung dieser Zeile wichtigen Kontinuen liegen in diesem Bereich.

Im Folgenden gilt es, für jede Zeile eine Einteilung der Kontinuen zu treffen. Jedes Kontinuum wird dabei einer von drei Gruppen zugeordnet. Diese Gruppen sind:



(a) Ausgangsbild mit Textlinien

(b) Zeilenbereich der zweiten Zeile

Abbildung 6.1: Bestimmung des Zeilenbereiches

- gehört sicher zur Zeile,
- gehört möglicherweise zur Zeile und
- gehört nicht zur Zeile

Die Kontinuen, die von der Basislinie der betreffenden Zeile geschnitten werden oder zwischen dieser und der Mittelline liegen, gehören sicher zu dieser Zeile. Alle Kontinuen, die oberhalb und unterhalb des oben beschriebenen Bereiches liegen werden der letzten Gruppe zugeordnet.

Aus der in Abschnitt 3.2 beschriebenen Eigenschaft dieser Schriften, dass Federzüge nur nach unten in die nächste Zeile ragen aber nicht nach oben, folgt, dass ein Kontinuum, das die obere Bereichsbegrenzung schneidet, zur darüber liegenden Zeile gehört. Für die aktuelle Zeile wird es daher als nicht zur Zeile gehörend eingestuft[1].

Die anderen Kontinuen, die sich im Zeilenbereich über der Mittellinie oder unter der Basislinie befinden, werden in die Gruppe der fraglichen Objekte eingeordnet.

In Abbildung 6.2 ist die resultierende Einteilung der Kontinuen an einem Beispiel farblich dargestellt. Dabei repräsentiert die hellgraue Farbe die Gruppe der nicht zu dieser siebenten Zeile gehörenden Kontinuen, die dunkelgrau markierten Objekte sind die Kontinuen, die möglicherweise dazugehören und die schwarze Farbe zeigt, dass diese Kontinuen sicher zur Zeile gehören.

[1] Es sei denn, das Kontinuum erstreckt sich bis über die aktuelle Basislinie. In dem Falle handelt es sich um eine Zeilenberührung und wird im nächsten Abschnitt behandelt.

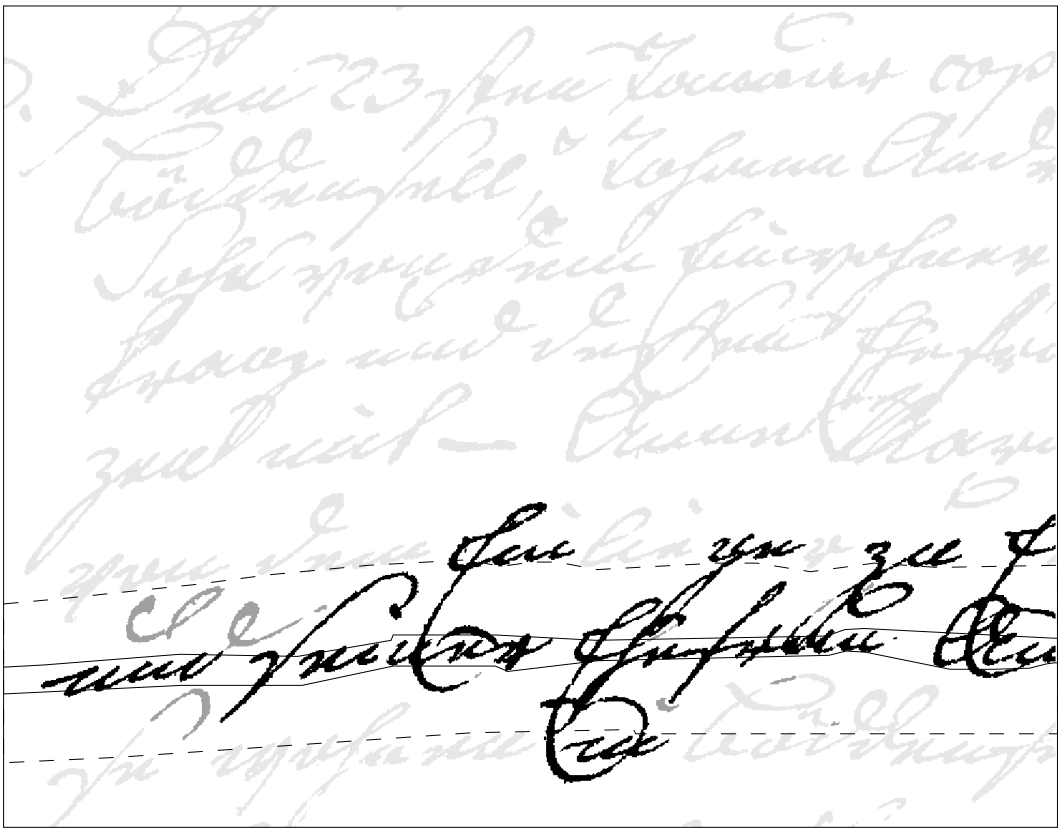


Abbildung 6.2: Einteilung der Kontinuen in drei Gruppen

6.2 Trennung verbundener Zeilen

Wie man in Abbildung 6.2 erkennen kann, führen Verbindungen zwischen Zeilen dazu, dass große Kontinuen entstehen, die sich über mehrere Zeilen erstrecken.

Das entscheidende Problem besteht in der möglichst korrekten Trennung von zwei sich berührenden Textzeilen. Diese Kontinuen müssen an einer oder mehreren Stellen getrennt werden. In vielen Fällen wird eine automatische, korrekte Zuordnung der einzelnen *Federzugsegmente*[2] nicht möglich sein. D. h. in vielen Fällen ist es nicht möglich, durch automatisierte Verfahren mit Sicherheit entscheiden zu können, dass ein bestimmtes Federzugsegment zu einem Federzug der darüber liegenden Zeile mit Unterlänge gehört. Hier wäre durch Interaktion eine manuelle Zuordnung möglich. Aber zum Einen soll in dieser Arbeit die automatische Verarbeitung im Vordergrund stehen, und zum Anderen ergäbe sich ein sehr großer manueller Aufwand, da beispielsweise auf einer Wegenstedter Kirchenbuchseite von 1812 ca. 200 Berührungen zwischen Zeilen bestehen.

[2] Teile eines Federzuges; sie werden durch Berührungen mit anderen Federzügen gebildet und sind äquivalent zu den erzeugten Kettencode-Elementen des Skeletts

Daher muss zunächst eine Methode existieren, die alle Verbindungsfälle so gut wie möglich behandelt. Danach können für bestimmte Klassen von Berührungssituationen spezielle Algorithmen zur Trennung entwickelt werden.

6.2.1 Finden der Verbindungsstellen

Bevor der Versuch einer Trennung erfolgen kann, muss zunächst die Stelle lokalisiert werden, die zwei Zeilen miteinander verbindet. In Frage kommen hier all die Kontinuen, die zu mehreren Zeilen gehören, da sie deren Textlinien schneiden. Wurde so ein Kontinuum gefunden, müssen nun die Federzugsegmente gefunden werden, die für die Verbindung der Zeilen verantwortlich sind. Das sind die Federzugsegmente, die sich im Zeilenzwischenraum befinden. D. h. sie liegen oberhalb der Mittellinie der unteren Zeile und unterhalb der Basislinie der oberen Zeile. Über diese Federzugsegmente kann zunächst keine eindeutige Aussage über ihre Zugehörigkeit zur oberen oder unteren Textzeile gemacht werden. Für diese Teile des Kontinuums wird für beide Zeilen der Status „unsicher“ gesetzt – ähnlich der in Abschnitt 6.1 beschriebenen Gruppierung der Kontinuen. Der Erkenner hat dann die Möglichkeit, diese Informationen für die Erkennung zu nutzen.

Um auf Basis des Kettencodes des Skelettes die unterschiedlichen Teile eines Kontinuums zu bestimmten, werden zunächst so genannte *Trennelemente* gesucht. Das sind die Kettencode-Elemente, die eine Textlinie schneiden und dabei den Teil des Kontinuums, der über dieser Textlinie liegt, mit dem darunterliegenden Teil verbindet. Es wird ein Pfad der Kettencode-Elemente gesucht, der den oberen mit dem unteren Teil verbindet und dabei durch diese Element verläuft.

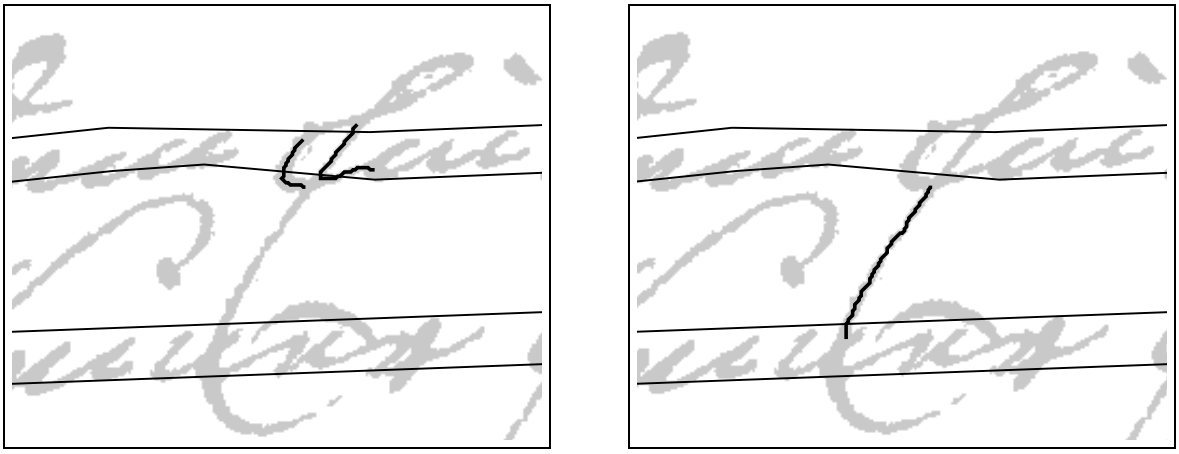
Um nun den genauen Schnittpunkt festzulegen, wird zunächst geprüft, ob sich zwei Trennelemente der Basisline auf ein Element zusammenfassen lassen. Dies ist sowohl nach oben als auch nach unten möglich. Dabei werden die Pfade zweier Trennelemente verglichen. Laufen diese beiden Pfade an einer Stelle zusammen, die nicht näher an einer anderen Textlinie liegt, wird dieses gemeinsame Element als Trennelement statt der zwei Ursprungselemente markiert.

Der genaue Trennpunkt ist jeweils am unteren Ende[3] des Trennelementes. Für die Trennelemente der Mittellinie gilt das, weil der unsichere Bereich im Zweifelsfall besser etwas zu groß als zu klein gewählt wird, während für die Trennelemente der Basislinie die in Abschnitt 6.1 genannte Feststellung zum Tragen kommt, dass Berührungen der Zeilen in der Regel durch Unterlängen der oberen Zeile verursacht werden.

Es entsteht ein unsicherer Kontinuumteil, der sich im Zeilenzwischenraum befindet und durch Trennelemente von den sicheren Kontinuumteilen der oberen und unteren Zeile getrennt ist.

Ist das Trennelement der oberen Basislinie identisch mit dem Trennelement der unteren Mittellinie, dann werden die beiden Zeilen an dieser Stelle lediglich durch ein

[3] Dies gilt abzüglich der Linienbreite, um vor einer Verzweigung oder Kreuzung zu schneiden



(a) ohne Zusammenfassung

(b) mit Zusammenfassung

Abbildung 6.3: Bildausschnitt mit Textlinie und markierten Trennelementen

Federzugsegment verbunden, welches zur oberen Zeile gehört, ohne dass ein unsicherer Kontinuumteil entsteht.

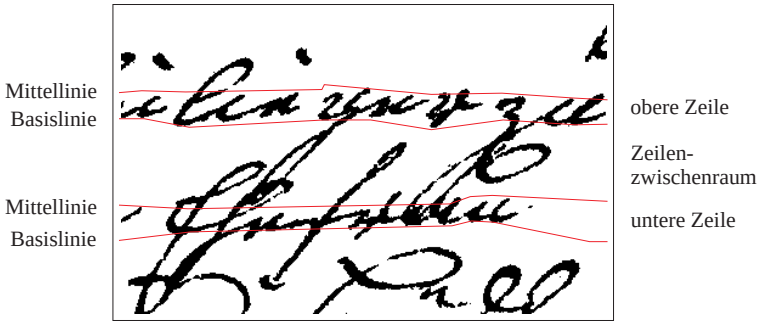
In den Abbildungen 6.4(b) und 6.4(c) ist die unterschiedliche Bewertung der Kontinuen bzw. Kontinuenteile durch unterschiedliche Grauwerte dargestellt. Für die entsprechende Zeile sind die sicheren Kontinuen schwarz und die unsicheren hellgrau eingefärbt, während die nicht zur Zeile gehörenden Kontinuen ganz gelöscht wurden. Die unsicheren Kontinuumteile sind durch ein dunkleres Grau abgebildet.

Durch dieses Verfahren lassen sich auch all die Zeilen segmentieren, die durch Berührung mit anderen Zeilen verbunden sind.

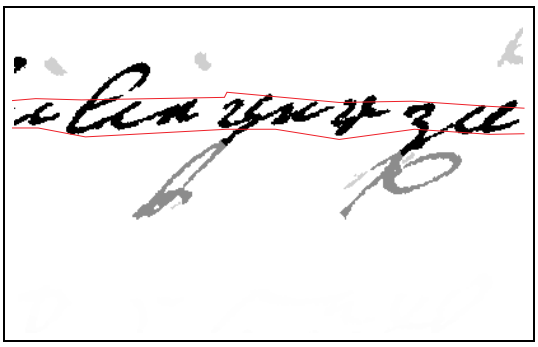
6.3 Analyse der Kreuzungspunkte

Um für einen späteren Erkenner die Erkennungsrate zu verbessern, müssen Möglichkeiten gefunden werden, um bei Zeilenberührungen möglichst viele Federzugsegmente der richtigen Zeile zuzuordnen. Wenn die Komplexität der Zeilenverbindung nicht zu groß ist, ist dies grundsätzlich möglich. Um dies zu verdeutlichen, werde ich im Folgenden eine Möglichkeit aufzeigen, um eine bestimmte Form der Zeilenverbindung zu verarbeiten. Es handelt sich bei dieser Form um eine Kreuzung von Federzügen, die durch *einen* Federzug der Unterlänge der oberen Zeile verursacht wird.

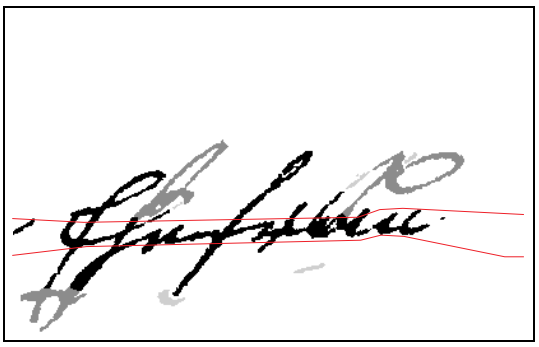
Die Idee einer Repräsentation *kompletter* Schriftzüge durch parametrische B-Splines habe ich nach genauerer Betrachtung der Skelette einiger Beispiele als nicht durchführbar eingeschätzt. Der Grund dafür wie auch für die Probleme der Zeilentrennung ist das Zusammentreffen von zu vielen Linien, die sich überkreuzen, berühren oder zu einer Linie zusammenlaufen. Das Ergebnis im Skelett ist eine sehr hohe Zahl von Abzweigun-



(a) Ausschnitt mit einem „mehrzeiligen“ Kontinuum



(b) obere Zeile



(c) untere Zeile

Abbildung 6.4: Trennung zweier Zeilen

gen, die sehr nahe beieinander liegen. Dies trifft auch auf komplexe Verbindungsstellen von Textzeilen zu.

Was die Analyse von Kreuzungspunkten betrifft, so beschreiben Nakajima et al. interessante Ansätze, in dem Möglichkeiten beschrieben werden, wie Linien, die sich kreuzen, segmentiert werden können [NMTS99]. Neben einer Methode, die auf Analyse der Kontur basiert, wird eine Methode beschrieben, die auf der Basis des Skelettes arbeitet. Das typische Problem bei der Skelettierung ist das Finden und Rekonstruieren von Kreuzungspunkten. In der Regel entstehen dabei zwei nebeneinander liegende Abzweigungen. In dem Algorithmus wird nach solchen Kombinationen gesucht und diese als potenzielle Kreuzungen markiert. Im Folgenden wird unabhängig vom Aussehen des unmittelbaren Kreuzungsgebiets für jede Kreuzung ein Satz hypothetischer B-Splines erzeugt, der sich aus den verschiedenen Kombinationsmöglichkeiten der beteiligten Liniensegmente ergibt. Es werden dabei also jeweils zwei Liniensegmente miteinander verbunden. Anschließend wird anhand der Krümmung verglichen, welche Spline-Kombination die wahrscheinlichste ist.

Ohne auf B-Splines zurückzugreifen, habe ich diese Methode aufgegriffen, um zunächst eine einfache Form der Überkreuzungen zu lösen. Es handelt sich dabei um eindeutige

Kreuzungen, bei denen sich die Linien in einem großen Winkel (nahe 90°) überschneiden.

Da meistens Unterlängen von Zeilen für Verbindungen verantwortlich sind, gilt es, diese zu rekonstruieren – zumindest in einigen Fällen . Es geht dabei um Federzugsegmente, die, ausgehend von der Basislinie, nach unten in die darunter liegende Zeile ragen und die Federzüge dieser unteren Zeile kreuzen. Da diese Federzugsegmente die Basislinie kreuzen und eine Verbindung zwischen „oben“ und „unten“ herstellen, wurden sie zuvor als Trennelement markiert. Im Folgenden werden alle Trennelemente der Basislinie daraufhin untersucht, ob an ihrem unteren Ende eine Kreuzung vorhanden ist. Wenn dies der Fall ist, wird das weiterführende Federzugsegment entsprechend markiert. Dies wird dann bei der Segmentierung der Zeile berücksichtigt.

6.3.1 Der Algorithmus

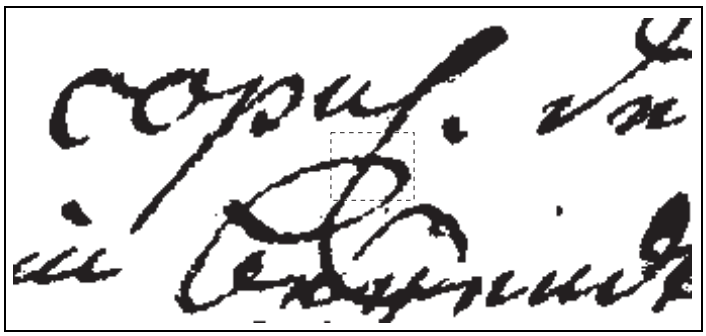
Für alle Trennelemente der Basislinien wird folgende Prozedur durchlaufen:

Auf Grund der Charakteristik des Skelettes schließen sich am unteren Ende des Trennelementes in der Regel zwei Kettencode-Elemente an.[4] Da ein Kreuzungspunkt im Skelett durch zwei Abzweigungen charakterisiert ist, die durch ein kurzes Liniensegment miteinander verbunden sind, wird untersucht, ob eines der beiden Elemente ein kurzes Element ist. Das ist dann der Fall, wenn es eine größere Breite als Länge besitzt. Wird so ein Element gefunden, wird abermals an dessen Ende überprüft, ob sich dort ein Abzweig befindet, von dem ein langes Element ausgeht, das gegenüber vom Trennelement liegt (siehe Abbildung 6.5(b)). Es muss ein langes Element sein, um seine Richtung bestimmen zu können. Das ist möglich, wenn seine Länge größer ist als die doppelte Breite. Das Element liegt dann gegenüber des Trennelementes, wenn entweder erst rechts und dann links oder erst links und dann rechts über dem Verbindungselement abgezweigt wurde.

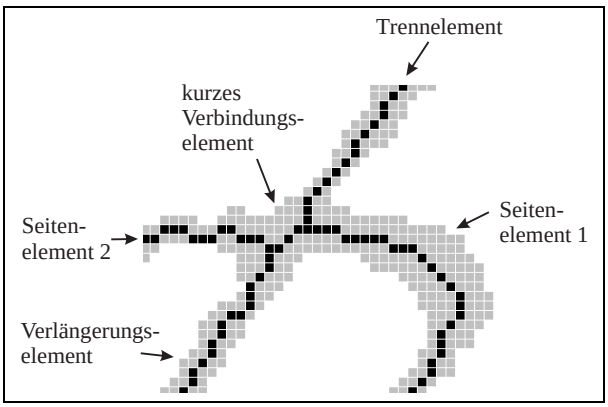
Ist so ein potentieller Kreuzungspunkt gefunden worden, wird die Richtungsqualität überprüft. Dazu werden jeweils zwei Punkte auf dem Trennelement und dem Verlängerungselement festgelegt. Ist w die durchschnittliche Breite des jeweiligen Federzugsegments, dann werden vom Abzweigpunkt beginnend die Kettenglieder mit der Nummer $m = w$ und mit der Nummer $n = 2w$ markiert. Durch das Verbinden der Punkte entstehen Vektoren, deren Winkel zur Waagerechten bestimmt wird (siehe Abbildung 6.5(c)). Es wird nun die Summe der Differenzbeträge zwischen den Winkeln der Vektoren 1 und 2 und zwischen den Winkeln der Vektoren 1 und 3 bestimmt. Anders ausgedrückt: Seien $a1$, $a2$ und $a3$ die Winkelwerte der drei Vektoren, so errechnet sich der Winkeldifferenzwert D folgendermaßen:

$$D = \mathrm{abs}(a1 - a2) + \mathrm{abs}(a1 - a3) \tag{6.1}$$

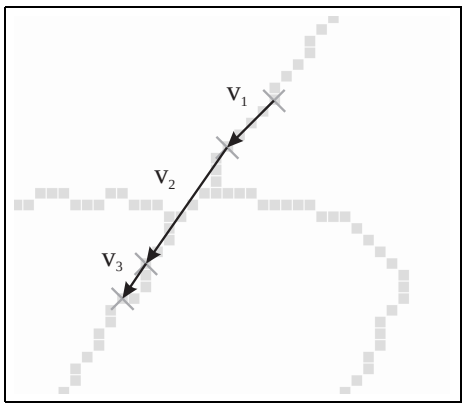
[4] Sollte der seltene Fall eintreten, dass hier vier Elemente aufeinander treffen, dann handelt es sich um einen idealen Kreuzungspunkt und die nächsten Schritte können übersprungen werden.



(a) Beispiel einer Zeilenverbindung durch mehrfache Überkreuzung



(b) Elemente des skelettierten Kreuzungspunktes



(c) Vektoren zur Qualitätsbestimmung

Abbildung 6.5: Analyse eines Kreuzungspunktes

Dieser Differenzwert muss unterhalb einer Schwelle liegen, damit dieser potentielle Kreuzungspunkt als „echter“ Kreuzungspunkt markiert wird. Um die Schwelle für diesen Wert zu setzen habe ich die errechneten Werte einiger echter Kreuzungspunkte mit denen falscher Kreuzungen verglichen. Erstere erlangten Werte, die sich im Bereich von 20° bewegten und stets kleiner als 36° waren. Die falschen Kreuzungspunkte ergaben Werte von 45° und mehr. Momentan ist die Schwelle auf 40° eingestellt und trennt somit gut zwischen wahren und falschen Kreuzungspunkten.

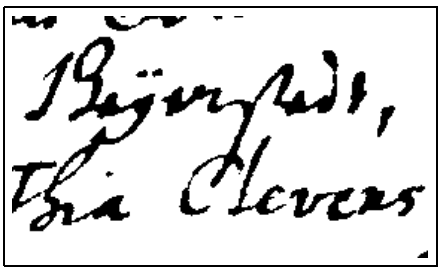
Der in Abbildung 6.5 dargestellte Kreuzungspunkt ergab einen Winkeldifferenzwert von 23°.

6.3.2 Beispiel

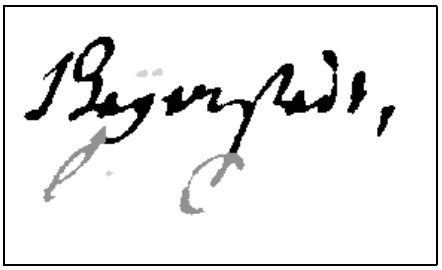
Dieser spezielle Kreuzungsart kommt nicht sehr häufig vor und hängt von der Schriftart ab. Auf der Seite aus dem Jahre 1812 tritt er im Mittel zweimal auf, während die

Tabelle der Seite 9 aus dem Jahre 1838, bei denen die Zahl der Zeilenberührungen gering ist, keinen solchen Kreuzungsfall aufweisen.

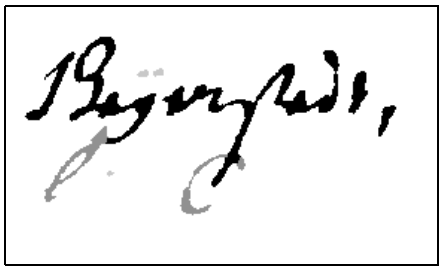
In Abbildung 6.6 wird die Verbesserung der Zeilentrennung deutlich, die der Algorithmus bei diesen Situationen erzielt.



(a) Ausgangsbild



(b) obere Zeile, vorher



(c) obere Zeile, nachher

(d) untere Zeile, vorher

(e) untere Zeile, nachher

Abbildung 6.6: Bildausschnitt aus Seite 111 aus dem Jahre 1649

6.4 Erzeugung von Grauwertbildern

Um einerseits eine gute Darstellungsform der Ergebnisse zu erzeugen und andererseits eine mögliche Datengrundlage für eine spätere Erkennung zu schaffen, ist es nun das Ziel, aus den gewonnen Daten des Kettencodes eine entsprechende Bildrepräsentation einer jeden isolierten Textzeile zu erzeugen.

Da für jede Zeile die Kontinuen und Teile der Kontinuen in bestimmte Klassen eingeteilt wurden, habe ich mich dazu entschlossen, statt eines Binärbildes ein Grauwertbild zu erzeugen. Die Zugehörigkeit eines Kontinuums bzw. Kontinuumteils wird hier durch einen bestimmten Grauwert repräsentiert.

6.4.1 Einfärben ganzer Kontinuen

Das Problem, das hierbei bestand, lag darin, dass keine direkte Verbindung zwischen einem Kettencode-Element und einem Vordergrundpixel besteht. Daher habe ich zunächst aus dem Binärbild des Textblockes mit Pixelwerten von 0 und 1 ein Grauwertbild mit den Pixelwerten 0 und 255 erzeugt. Über einen normalen Füllalgorithmus habe ich ausgehend von den bekannten Koordinaten der Kettencode-Elemente die Kontinuen des Rasterbildes entsprechend eingefärbt. Kontinuen, die nicht zur Zeile gehören, werden mit dem Grauwert 255 gefüllt und damit gelöscht, unsichere Kontinuen erhalten den Grauwert 180, der einem hellen Grau entspricht, und Kontinuen, die sicher zur Zeile gehören, behalten ihren Grauwert 0. Dadurch entstehen Bilder, wie eines in Abbildung 6.2 auf Seite 41 zu sehen ist. Hier wurden lediglich etwas andere Grauwerte benutzt und die Textlinien nachträglich hinzugefügt.

6.4.2 Einfärben von Kontinuumteilen

Schwieriger verhält es sich mit „mehrzeiligen“ Kontinuen, die nach der Zeilentrennung aus unterschiedlichen Kontinuumteilen bestehen. Für ein Kettencode-Element ist bekannt, zu welchem Kontinuumteil es gehört. Es gibt nun mehrere Möglichkeiten, die Pixel des Kontinuums entsprechend einzufärben. Denkbar wäre hier beispielsweise ein Füllalgorithmus, der nur vertikal zum Verlauf des Kettencode-Elementes arbeitet. Ich habe folgenden Weg gewählt:

Die Grenze zwischen zwei unterschiedlichen Teilen eines Kontinuums werden durch die unteren Enden der gefundenen Trennelemente bestimmt. An dieser Stelle wird im Rasterbild der Federzug durchtrennt, indem dort die Werte der Pixel von 0 auf 255 gesetzt werden. Nun werden die einzelnen Kontinuumteile durch den Füllalgorithmus entsprechend ihrem Status eingefärbt. Anschließend werden die Lücken mit dem entsprechenden Grauwert wieder geschlossen.

6.4.3 Richtiges Einfärben der Kreuzungsbereiche

Wenn ein Kreuzungspunkt gefunden wurde, muss außerdem entschieden werden, wie der Kreuzungsbereich eingefärbt werden soll. Ausgehend von der *oberen Textzeile* gehören das Trennelement, der Kreuzungsbereich und das Verlängerungselement mit Sicherheit zur Zeile und müssen anders eingefärbt werden, als die unsicheren Federzugsegmente, die sich im Zeilenzwischenraum befinden. Hier wird also an den Anfängen der beiden Seitenelemente und am Ende des Verlängerungselementes geschnitten. Für

die *untere Zeile* muss dieses Trennelement mit dem Verlängerungselement gelöscht werden, während der Kreuzungsbereich als „unsicher“ erhalten bleibt. Dafür wird das Trennsegment an seinem Ende durchtrennt, als würde kein Kreuzungspunkt existieren. Zusätzlich wird das Verlängerungselement an seinem Anfang und seinem Ende durchtrennt. So wie der Teil des Kontinuums, der zur oberen Zeile gehört, kann nun auch des Trenn- und das Verbindungselement mit dem Grauwert 255 gelöscht werden.

Kapitel 7

Test der Textlinienfindung

7.1 Schriftproben

Dank der Kirchengemeinde in Wegenstedt war es mir möglich, Schriftproben mit einem größeren Spektrum an verschiedenen Schriftarten zu digitalisieren. Es handelt sich dabei um 61 Textblöcke von 7 Kirchenbuchseiten aus dem Jahren 1649 (2 Seiten), 1727, 1741, 1812, 1817, 1838. Alle 61 Blöcke zusammen ergeben eine Zeilenzahl von 301. Man kann ungefähr sagen, dass sich auf jeder Dokumentenseite eine eigene Schriftart befindet. Auf der Seite von 1838 sind es sogar zwei. Daraus ergeben sich 8 unterschiedliche Handschriften mit unterschiedlichen Eigenschaften.

Alle Dokumente habe ich mit einem Flachbettscanner in einer Ortsauflösung von 600 dpi und einer Farbtiefe von 3×8 Bit gescannt. Nachdem klar wurde, dass 300 dpi für diese Schrift ausreichend sind, habe ich die Bilder auf diesen Wert herunter skaliert.

7.2 Parameter

Damit die Linienfindung für einen Textblock mit einer bestimmten Schrift korrekt funktioniert, müssen einige Parameter möglichst optimal eingestellt werden. Diese Parameter definieren bestimmte Toleranzen von geometrischen Merkmalen. Praktisch geschieht diese Einstellung in einer Konfigurationsdatei. Die 13 Parameter seien im Folgenden kurz erläutert:

Mindestgröße der Kontinuen Diese Schwelle regelt das Herausfiltern von Kleinobjekten wie, in Abschnitt 5.2.1 beschrieben. D. h. alle Objekte, die in ihrer horizontalen und vertikalen Ausdehnung kleiner sind als dieser Wert (in Pixel), werden für die Textlinienfindung nicht beachtet. *Ist er zu klein*, dann führen die Extrempunkte der Kleinobjekte und Verunreinigungen zu Störungen bei der Textlinienfindung. *Ist er zu groß*, dann fallen kleine Worte oder einzelne Zeichen unter diese Grenze und werden nicht beachtet.

maximaler horizontaler Abstand der Minimumpunkte Dieser Wert begrenzt den horizontalen Bereich, in dem beim Bilden der Basisliniensegmente nach weiteren Minima gesucht wird, um ein Segment zu erweitern (siehe Abschnitt 5.2.2). *Ist*

er zu klein, dann werden keine langen Segmente gebildet und der Textlinienverlauf kann nicht gefunden werden. *Ist er zu groß*, dann entstehen lange Segmente, die quer über mehrere Zeilen reichen.

Höhentoleranz der Basisliniensegmente Wie auch der vorige Parameter, bezieht sich dieser Toleranzwert auf das im Abschnitt 5.2.2 beschriebene Bilden der Basisliniensegmente und gibt die maximale Höhe an, die ein solches Segment haben kann. D. h. unter dem Winkel des Segmentes betrachtet, darf der vertikale Abstand zwischen zwei Minimumpunkten nicht größer sein als dieser Wert. *Ist er zu klein*, werden ebenfalls keine langen Segmente gebildet. *Ist er zu groß*, entstehen auch wieder lange Segmente an Stellen, an denen keine Textlinien verlaufen.

Mindestzahl der Minimumpunkte eines Segmentes Nachdem die Bildung aller Basisliniensegmente abgeschlossen wurde, werden – wie in Abschnitt 5.2.5 beschrieben – alle Segmente gelöscht, deren Minimumpunktanzahl kleiner ist als dieser Wert. Sinnvoll ist ein Wert von drei oder vier. *Ist er zu klein*, dann könnten zu kurze Segmente, die in großer Zahl entstehen, zum Anwachsen des Rechenaufwandes führen. *Ist er zu groß*, dann werden wichtige Segmente entfernt und die Textlinien können nicht gefunden werden.

Erster Winkelbereich In Abschnitt 5.2.3 wird beschrieben, wie die Rotation des vorliegenden Textblockes ermittelt wird. Dazu werden in diesem Winkelbereich alle Liniensegmente auf Grund der Minimumpunkte gebildet. $\pm 20°$ ist ein Wert, der standardmäßig eingestellt werden kann. *Ist er zu klein*, wird eine eventuelle Rotation nicht erkannt und ein zu kleiner Rotationswinkel ermittelt. *Ist er zu groß*, wird eine zu große Zahl von Segmenten erzeugt, und der Rechenaufwand steigt unnötig an.

Zweiter Winkelbereich Nachdem die Rotation des Textblockes ermittelt wurde, besteht nur noch die Richtungsabweichung innerhalb des Textblockes. D. h. es geht um die Frage: Wie weit weicht ein Textliniensegment von der durchschnittlichen Richtung des Textblockes ab? *Ist er zu klein*, werden eben diese abweichenden Textliniensegment nicht erkannt und ein falscher Textlinienverlauf gefunden. *Ist er zu groß*, können Segmente entstehen, die quer zur Textrichtung verlaufen und zu einem falschen Textlinienverlauf führen. Dies kann trotz optimal eingestelltem Abstandswert der Minimumpunkte und Höhentoleranz bei der Segmentbildung auftreten.

Maximaler horizontaler Abstand der Segmente Dieser Parameter wird bei der Bildung der kompletten Basislinie genutzt (siehe Abschnitt 5.2.7). Genauer gesagt definiert er, wie weit ein Minimumpunkt eines Segmentes vom aktuellen Endpunkt der Basislinie entfernt sein darf, damit dieses Segment zur Verlängerung der Basislinie beiträgt. *Ist er zu klein*, können die gefundenen Segmente nicht zu einer durchgehenden Basislinie kombiniert werden, und für eine Textlinie werden mehrere „Bruchstücke“ von Basislinien erzeugt. *Ist er zu groß*, erhöht es lediglich den Rechenaufwand, da bei der Entscheidung zwischen Segmentalternativen die Entfernung mit berücksichtigt wird (siehe Formel 5.2).

Maximaler vertikaler Abstand der Segmente Wie der vorige Parameter wird auch dieser Wert bei der Bildung der Basislinien genutzt (siehe Abschnitt 5.2.7). Dabei legt er fest, wie weit ein Minimumpunkt eines Segmentes vertikal vom Endpunkt der Basislinie entfernt sein darf, damit es für eine Verlängerung in Betracht kommt. *Ist er zu klein*, werden auch hier keine langen Basislinien aus den Segmenten gebildet. *Ist er zu groß*, kann in ungünstigen Fällen ein falsches Segment zur Verlängerung der Basislinie genutzt werden.

Kurze Textlinien Dieser Wert gibt an, ab wieviel Prozent der durchschnittlichen Zeilenlänge eine Textlinie als kurze Textlinie gilt. Dies wird beim Entfernen von falschen Textlinien genutzt (siehe Abschnitt 5.2.8). Er ist momentan auf 80% eingestellt und es gab bisher keine Veranlassung, dies zu ändern. *Ist er zu klein*, werden falsche Linien nicht erkannt. *Ist er zu groß*, werden auch Linien mit durchschnittlicher Länge als kurz eingestuft und u. U. entfernt.

Zu kleiner Zeilenabstand Analog zum vorigen Parameter ist dies der zweite Wert, der beim Entfernen von falschen Textlinien herangezogen wird (siehe Abschnitt 5.2.8). Er gibt an, ab wieviel Prozent des durchschnittlichen Zeilenabstandes zwei Textlinien zu nahe beieinander liegen. Er ist auf 60% eingestellt und braucht auch nicht geändert zu werden. *Ist er zu klein*, werden Problemfälle nicht erkannt. *Ist er zu groß*, werden kurze Zeilen entfernt, wie z. B. Absätze.

Vertikales Offset der Textlinien Die Basislinie werden um diesen Wert nach unter verschoben, Mittellinien nach oben. Damit werden die Folgen korrigiert, dass die Extrempunkte vom Skelett stammen und damit um eine halbe Linienbreite verschoben sind und dass die Liniensegmente vertikal mittig durch die Extrempunkte verlaufen. Der Wert von 6 Pixel hat sich bei 300 dpi bewährt. *Ist er zu groß oder zu klein*, wirkt sich das negativ auf die Zeilentrennung und spätere Erkennung aus.

Höhentoleranz der Mittelliniensegmente Analog zur Höhentoleranz der Basisliniensegmente wird hier der Wert für die Mittelliniensegmente festgelegt. Er ist im Allgemeinen etwas größer als bei den Basisliniensegmenten, da die Ausprägung der Mittellinie meist nicht so gut ist wie die der Basislinie. *Ist er zu klein*, werden die Mittelliniensegmente und damit u. U. die gesamt Mittellinie nicht gefunden, was dazu führt, dass auch die entsprechende Basislinie gelöscht wird. *Ist er zu groß*, liegt das gefundene Mittelliniensegment zu hoch oder zu niedrig.

Maximaler Abstand der beiden Textlinien Dieser Parameter wird bei der Suche der Mittellinie genutzt (siehe Abschnitt 5.2.10) und besagt in Prozent, wie weit die Mittellinie von der Basislinie im Verhältnis zum durchschnittlichen Zeilenabstand entfernt sein darf. *Ist er zu klein*, wird die Mittellinie nicht gefunden und damit auch die Basislinie für diese Zeile gelöscht. *Ist er zu groß*, wird ein zu hoher Verlauf der Mittellinie durch große Zeichen oder Unterstreichungen der darüberliegenden Zeile erzeugt.

7.3 Ergebnisse

Wie schon erwähnt, müssen die Parameter speziell für eine Schriftart eingestellt werden, um optimale Ergebnisse zu erhalten. Dennoch funktioniert der Algorithmus selbst ohne manuelle Anpassung der Parameter recht gut auf den meisten Schriftproben, wie dies aus folgenden Darlegungen ersichtlich ist.

Ich unterteile die auftretenden Fehler in zwei Stufen. Der *leichte Fehler* besteht dann, wenn für eine Zeile beide Textlinien gefunden wurden, diese aber an einer Stelle etwas nach oben oder unten vom eigentlichen Zeilenverlauf abweichen. Dies hat für die Trennung der Zeilen keine negativen Folgen, es kann allerdings bei der späteren Erkennung zu einer Verschlechterung kommen, wenn die Information des Textlinienverlaufs mit einbezogen wird. Ein *schwerer Fehler* liegt vor, wenn die gefundenen Textlinien stärker vom eigentlichen Verlauf abweichen, eine Zeile gar nicht erst gefunden wurde oder zusätzliche Textlinien an einer falschen Stelle erzeugt werden.

Die Schriften der beiden Seiten aus dem 18. Jh. möchte ich hier gesondert betrachten, da diese Schrift sich stärker von den anderen unterscheidet und zum Teil auch größer ist. Deshalb müssen hier die Parameter speziell abgestimmt werden.

Für alle anderen Seiten habe ich versucht, einen optimalen Kompromiss der Parameter-Einstellung zu finden:

Mindestgröße der Kontinuen	15
Max. horiz. Abstand der Minimumpunkte	80
Höhentoleranz der Basisliniensegmente	6
Min. Zahl der Minimumpunkte eines Segmentes	4
Erster Winkelbereich	15
Zweiter Winkelbereich	6
Max. horiz. Abstand der Segmente	300
Max. vertik. Abstand der Segmente	30
kurze Textlinien in %	80
Zu kleiner Zeilenabstand in %	60
Höhentoleranz der Mittelliniensegmente	8
Vertikaler Offset der Textlinien	6
Max. Abstand der beiden Textlinien in %	60

Mit dieser festen Einstellung der Parameter werden von 246 Zeilen aus 49 Blöcken 222 richtig gefunden, 17 mit leichtem Fehler und 7 mit schwerem Fehler. Das ergibt eine Erkennungsrate von 90%.

Lässt man die Anpassung der Parameter auf die Schriften zu, so können nun auch die Dokumente des 18. Jh. mit in die Betrachtung einbezogen werden. Von 300 Zeilen werden 291 richtig erkannt, 5 leichte und 4 schwere Fehler traten auf. Die Erkennungsrate beträgt nun 97%.

In den Tabellen A.1 und A.2 sind die Ergebnisse detailiert dargestellt.

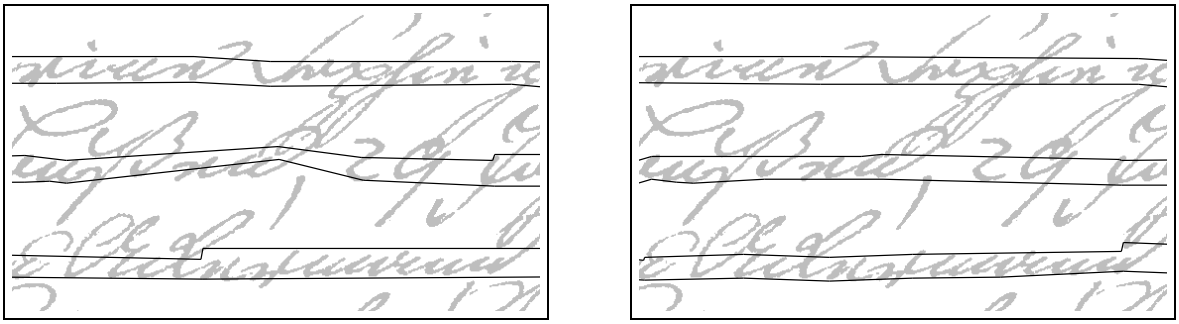
7.4 Fehlerstellen

Die im Folgenden beschrieben Fehler lassen sich meistens durch eine Anpassung der Parameter beseitigen. Wie an den Ergebnissen zu sehen ist, ist es mir in drei Prozent der Fälle nicht gelungen, eine Einstellung zu finden, bei der der Fehler nicht auftritt.

7.4.1 Abweichung der Basislinie

Es kommt zu einer Abweichung der Basislinie, wenn über oder unter der eigentlichen Basislinie Segmente erzeugt wurden, die dann zur Basislinienbildung genutzt werden.

Durch Optimierung der Parameter, die zum Einen zur Bildung der Segmente und zum Anderen zum Erzeugen der Textlinien genutzt werden, kann dieser Fehler in den meisten Fällen korrigiert werden. D. h. die Verringerung der Höhentoleranz der Segmente, des Abstands der Minimumpunkte bei der Segmentbildung und des vertikalen Abstands der Segmente bei der Basislinienerzeugung können das Problem lösen.



(a) mit obigen Einstellungen

(b) mit ±5 Grad Winkelbereich

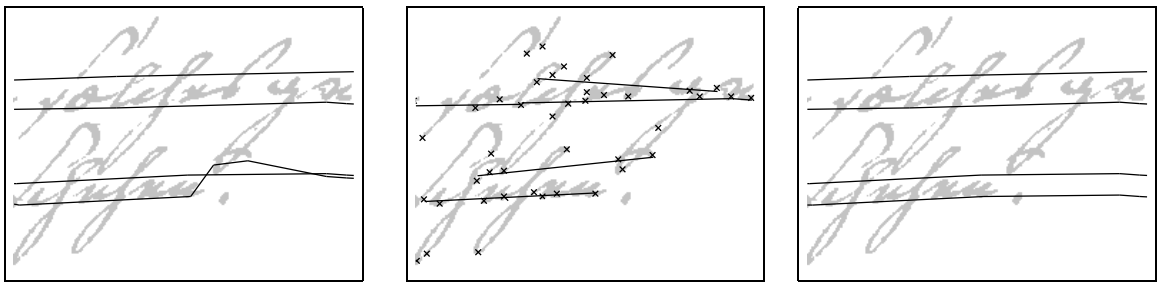
Abbildung 7.1: Bildausschnitt aus Textblock 3 von S. 16 von 1812

Während bei diesem Problem auch ein Eingrenzen des Winkelbereiches bei der Bildung der Segmente helfen kann, kann in einem anderen Fall ein Vergrößern dieses Bereiches notwendig sein. Dies ist dann der Fall, wenn der Winkelbereich zuvor zu klein gewählt worden ist und deswegen der wahre Basislinienverlauf nicht gefunden werden kann.

7.4.2 Abweichung der Mittellinie

Dies tritt oft an Stellen auf, wo viele Zeichen mit Oberlänge nebeneinander stehen und somit an der oberen der vier möglichen Textlinien eine größere Häufung von Maximumpunkten entsteht als an der Mittellinie.

Meist hilft hier eine Einschränkung des vertikalen Bereichs, in dem sich die Mittellinie befinden darf. Bei den meisten Schriften ist das Verhältnis von Zeilenabstand zum



(a) mit obigen Einstellungen

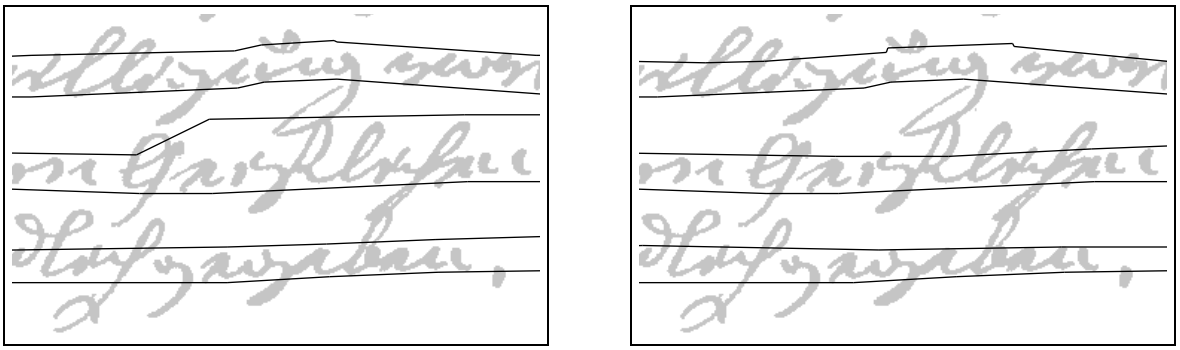
(b) Grund: falsche Segmente

(c) mit 20 Pixeln vertikaler Toleranz Segmentabstand

Abbildung 7.2: Bildausschnitt aus Textblock 3 von S. 30 von 1817

Abstand der Mittellinie zur Basislinie relativ konstant und gut einzuschätzen. Den entsprechenden Parameter kann man somit auf 50 oder 40 % absenken.

Ein anderer Grund kann ein zu kleiner Toleranzbereich bei der Suche nach Mittelliniensegmenten sein. Bei in dieser Beziehung „schlechten“ Schriftarten muss der Toleranzbereich entsprechend vergrößert werden.



(a) mit obigen Einstellungen

(b) mit 12 Pixeln Höhentoleranz der Mittelliniensegmente

Abbildung 7.3: Bildausschnitt aus Tabellenfeld 3. Zeile, 3. Spalte von S. 9 von 1838

7.4.3 Nichtfinden der Mittellinie

Dieser Fall tritt ein, wenn von keinem Segment der Basislinie aus ein Mittelliniensegment gefunden werden kann, das mindestens die Mindestzahl der Maximumpunkte besitzt. Die Folge ist das Entfernen der Basislinie und damit der gesamten Zeile für den weiteren Zeilentrennungsprozess.

Hierfür gibt es zwei Gründe. Entweder ist die Höhentoleranz der Mittelliniensegmente zu klein oder der Maximalabstand zwischen Basislinie und Mittellinie ist zu eng bemessen. In beiden Fällen kann eine entsprechende Änderung der Parameter helfen.

(a) mit obigen Einstellungen

(b) mit 12 Pixeln Höhentoleranz der Mittelliniensegmente

Abbildung 7.4: Tabellenfeld 4. Zeile, 4. Spalte von S. 9 von 1838

Kapitel 8

Besonderheiten der Implementierung

In diesem Kapitel gehe ich auf Punkte ein, die mit dem eigentlichen Algorithmus nichts zu tun haben, sondern die praktische Umsetzung betreffen.

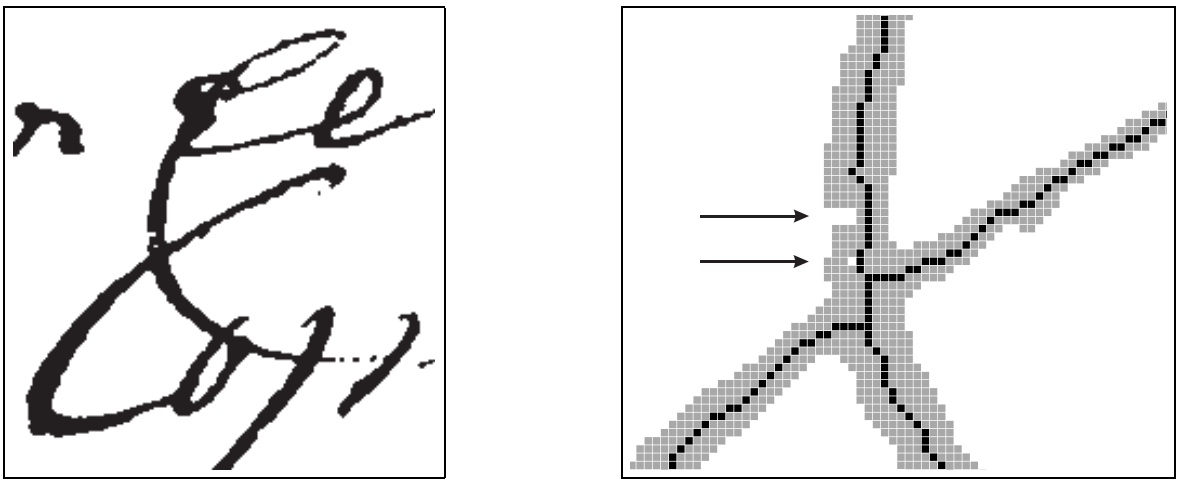
8.1 Warum Vectory?

Das Programm VECTORY der Firma Graphikon ist ein Konvertierungsprogramm, um aus Rasterdaten Vektordaten zu erhalten. Es ist auf Rasterbilder spezialisiert, die aus den Bereichen Architektur, Elektrotechnik oder Kartographie stammen. Als Ergebnis liefert das Programm Vektordateien im DXF- oder DNG-Format. Es erkennt geometrische Grundelemente wie Linien, Kreise und Kreisbögen aber auch Standard- und Kursivschrift.

Es gibt zwei Argumente, warum VECTORY für das Problem der Kirchenbuchaufzeichnungen geeignet ist. Zum Einen liegt hier ein komplettes GUI[1] vor – inklusive Mausunterstützung, Zoom- und Scroll-Funktionen – in dem eine kombinierte Darstellung des Rasterbildes mit den gewonnenen Vektordaten möglich ist. Zum Anderen wurde das Programm für die Verarbeitung von Linienzeichnungen als Rasterbilder konzipiert und verfügt daher über einen sehr ausgereiften Algorithmus zur Skelettierung und erzeugt eine gut zu handhabende interne Datenstruktur des Kettencodes.

Abbildung 8.1 demonstriert die Vorteile der Skelettierung von VECTORY. Selbst Störungen, wie starke Einbuchtungen (obere Pfeil in Abbildung 8.1(b)) oder gar Lücken innerhalb einer Linie (unterer Pfeil), führen nicht zur Bildung von falschen Linien im Skelett.

[1] Graphical User Interface – graphische Benutzeroberfläche



(a) Ausschnitt aus Wegenstedter Kirchenbuch, 1727

(b) vergrößerter Ausschnitt mit Skelett

Abbildung 8.1: Beispiel für den optimierten Skelettierungsalgorithmus in VECTORY

8.2 Zur Textlinienfindung

8.2.1 Objekte und deren wechselseitigen Verweise

Das Kernstück der Implementierung auf Grundlage des Quellcodes von VECTORY ist die Verwaltung der Objekte in doppelt verketteten Listen[2], die ein dynamisches Hinzufügen und Entfernen, Sortieren und schnelles Durchsuchen ermöglichen. Desweiteren kann eine Liste auch aus Zeiger-Objekten bestehen, die auf andere Objekte einer Liste verweisen. Die Vorverarbeitung von VECTORY liefert nach der Skelettierung eine Liste mit *Kontinuum-Objekten* und eine Liste mit *Kettencode-Objekten*. Ein Kontinuum-Objekt besitzt eine Liste mit Zeigern auf seine Kettencode-Objekte.

Auf Basis der Liste der Kontinuen wird in dem hinzugefügten Kirchenbuch-Modul zunächst eine Verbindungselement-Liste erzeugt, die eine wechselseitige Verkettung ermöglicht und zusätzliche Informationen des Kontinuums speichert. Darauf aufbauend werden dann die beiden Listen der Minimum- und Maximumpunkte mit Elementen gefüllt.

Die Liste der Vatersegmente beinhaltet die Textliniensegment-Elemente, die direkt aus der Analyse der Minimumpunkte gewonnen werden und mit denen die Rotationsbestimmung erfolgt.

Sie dienen ebenfalls als Grundlage der Basislinienbestimmung. Dabei wird die Liste der Basislinien mit Polylinien-Elementen gefüllt. Gleichzeitig wird bei jedem Polylinienelement die eigene Textliniensegment-Liste mit Elementen gefüllt. D. h. jedes

[2] Jedes Element besitzt einen Zeiger auf seinen Vorgänger und seinen Nachfolger.

Polylinienelement besitzt neben einer Liste mit Zeigern auf die Vatersegmente eine Liste mit seinen eigenen Segmenten.

Beim Füllen der Mittellinien-Liste werden Polylinien-Elemente inklusive eigener Textliniensegmente erzeugt und wechselseitig mit den entsprechenden Polylinien-Elementen der Basislinienliste verkettet.

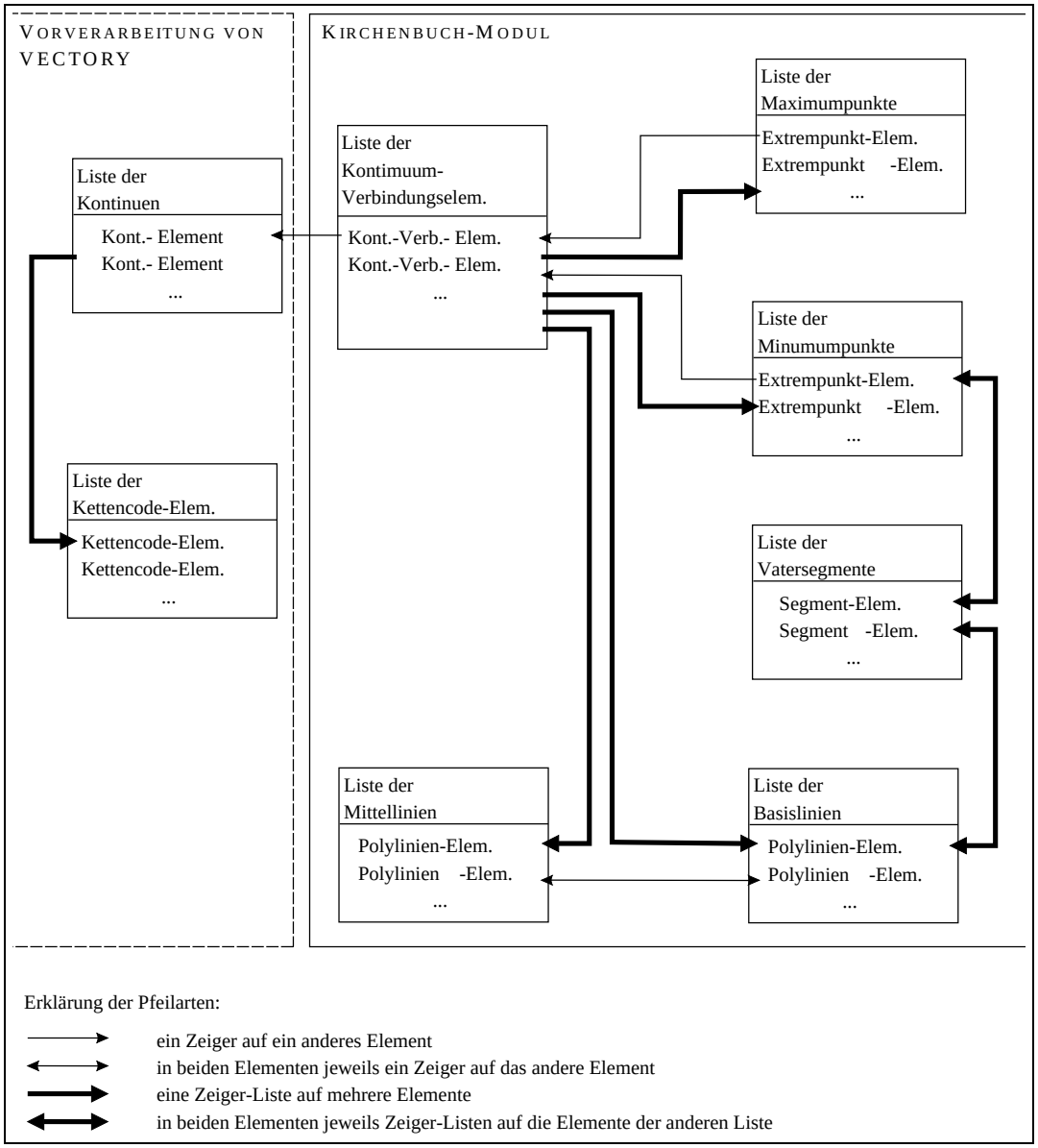


Abbildung 8.2: Listenstruktur zur Textlinienfindung

Abbildung 8.2 verdeutlicht den Zusammenhang der Listen und ihrer Objekte. Die Textliniensegment-Listen der Polylinien wurden aus Gründen der Übersichtlichkeit weggelassen.

8.2.2 Hinweis zum Finden der Basisliniensegmente

Bevor mit der eigentlichen Erzeugung der Basisliniensegmente begonnen wird, wird für jedes Extrempunkt-Objekt neben den „normalen“ Koordinaten die rotierten Koordinaten in Gradschritten von $-90°$ bis $+90°$ berechnet und in einem Feld abgelegt. Dadurch lässt sich im Folgenden sehr schnell überprüfen, wie groß der Abstand eines Punktes zu einer Linie mit einem bestimmten Winkel ist – angelehnt an die Methode der Hough-Transformation (siehe Abschnitt 2.1.2). Der Rotationmittelspunkt ist dabei für das Prinzip unerheblich. Hier ist es der Koordinatenursprung.

8.2.3 Zur Zeilensegmentierung

Für die Trennung von verbundenen Zeilen besitzen die „mehrzeiligen“ Kontinuen eine Liste mit Trennelement-Objekten, die einen Zeiger auf das jeweilige Kettencode-Element besitzen. Außerdem besitzen diese Trennelement-Objekte eine Liste mit Zeigern auf die Kettencode-Elemente, die den Pfad zwischen dem oberen und dem unteren Teil des Kontinuums angeben. Sie besitzen einen Zeiger auf die Textlinie, die das Element schneidet und gegebenenfalls einen Zeiger auf das Element, das diesen Federzug nach einem Kreuzungspunkt verlängert.

8.3 zur Erzeugung der Grauwertbilder

Wie schon in Abschnitt 6.4.1 erwähnt, besteht in VECTORY keine direkte Verbindung zwischen den Elementen des Kettencodes und den Elementen des Pixelbildes. Darüber hinaus ist es an der Stelle des Verarbeitungsablaufs, an dem mein Kirchenbuch-Modul einsetzt, nicht ohne weiteres möglich, auf das Pixelbild zuzugreifen, da dies bei der Vektorisierung nicht notwendig ist. Um nicht unnötig Zeit zu verlieren, entschied ich mich, diesen Teil mit der Entwicklungsumgebung IDL (Interactive Data Language) umzusetzen, da diese Sprache für die Verarbeitung von Rasterbildern sehr gut geeignet ist.

Als Schnittstelle zwischen VECTORY und dem IDL-Programm dienen Dateien, die die Koordinaten und Richtungen der Schnittstellen sowie die Koordinaten der einzufärbenden Kontinuen enthalten.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Fazit

Durch die ständig wachsende Leistungsfähigkeit der Rechner können heute sehr komplexe Berechnungen in kürzester Zeit durchgeführt werden, wie sie vor allem bei Erkennungsverfahren notwendig sind. Auch aus diesem Grund wurden auf dem Gebiet der Handschrifterkennung große Fortschritte erzielt. Daher bestehen auch gute Chancen, dass auch für mehrere hundert Jahre alte historische Aufzeichnungen, die wohl zu den schwierigsten Schriftvorlagen gehören, eine maschinelle Erkennung möglich ist.

Um diese Ziel zu verwirklichen, wurde in dieser Arbeit ein Lösungsweg gesucht, der – ausgehend von den realen Dokumenten – die Grundlage für einen Handschrifterkenner liefert, der auf diese alten Schriften abgestimmt ist.

Als Erstes muss das vorliegende Dokument digitalisiert werden. Es wurde gezeigt, dass verschiedene Techniken Vor- und Nachteile besitzen, die gegeneinander abgewogen werden müssen. Mit dem von mir verwendeten Flachbettscanner habe ich sehr gute Ergebnisse erhalten.

Das resultierende 24-Bit-Farbbild muss in mehreren Schritten zu einem Binärbild konvertiert werden. Um den besten Kontrast zwischen Hintergrund und Schrift zu erhalten, wird der Farbkanal ausgewählt, der die größte Standardabweichung der Helligkeitswerte aufweist. Gegebenenfalls wird ein anderer Kanal gewählt, wenn farbige Unterstreichungen in den Kanälen unterschiedlich stark stören. Dieses resultierende Grauwertbild wird durch die Verwendung eines Hochpassfilters und eines Schwellwertfilters in ein Binärbild konvertiert. Störungen durch Stockflecken können durch die Berechnung einer Maske und anschließender ODER-Verknüpfung mit dem Bild entfernt werden.

Die optimierten Algorithmen von VECTORY erzeugen aus diesem Binärbild eine Datenstruktur, die die Skelette der Vordergrundobjekte (Kontinuen) als Kettencode darstellt.

In idealer Betrachtung existieren in einer Textzeile vier Textlinien. Für die alten Schriften der Kirchenbücher lassen sich lediglich die Baislinie und die Mittellinie feststellen. Diese sind keine Geraden, die waagerecht und im konstanten Abstand voneinander in einem Textblock liegen. Vielmehr besitzt jedes Wort seine eigenen Textlinien. Diese Textliniensegmente müssen gefunden und miteinander verbunden werden. Das wird zunächst für die Basislinie durchgeführt. Dazu werden durch eine Art Hough-Transformation nach lokalen Minima der Objekte gesucht, die unter einem bestimmten

Winkel hintereinander liegen. Nach mehreren Reduktionsschritten bleibt eine gewisse Zahl von Segmenten übrig, die anschließend zu kompletten Textlinien kombiniert werden. Auf der Grundlage der Basislinien werden Segmente der Mittellinie gesucht, die parallel zur Basislinie verlaufen und durch Häufungen lokaler Maxima gefunden werden. Auch diese Segmente werden anschließend zu kompletten Mittellinien kombiniert.

Zum Einen können diese Linien zu einer besseren Erkennung beitragen, zum Anderen werden damit die einzelnen Zeilen voneinander segmentiert. Dazu wird für jede Zeile eine Einteilung der Kontinuen getroffen, ob sie zur Zeile gehören, ob sie nicht zur Zeile gehören oder ob dies nicht entschieden werden kann.

Wenn sich zwei Zeilen berühren, entstehen Kontinuen, die sich vertikal über mehrere Zeilen erstrecken. Diese müssen getrennt werden. Dies geschieht ungefähr dort, wo die Textlinien die Kontinuen schneiden. Die entstandenen Kontinuumteile werden wieder jeweils einer Kategorie der Zugehörigkeit zugeordnet.

Das bei diesem Trennungsprozess Federzüge der einen Zeile einer anderen zugeordnet werden, lässt sich nicht vermeiden. Für bestimmte Fälle kann diese Zuordnung aber verbessert werden. Verläuft ein Federzug einer Zeile in den Bereich der darunter liegenden Zeile, so kann es zu einer Kreuzung der Linien kommen. In einigen Fällen lässt sich dieser Kreuzungspunkt erkennen und ein weiteres Segment eines Federzuges der richtigen Zeile zuordnen.

Nachdem die Zuordnung der Kontinuen und Kontinuumteile erfolgt ist, wird anschließend für jede Zeile ein Grauwertbild erzeugt. In diesem wurden die Kontinuen entfernt, die nicht zu dieser Zeile gehören, während die zur Zeile gehörenden Kontinuen erhalten bleiben und als schwarze Objekte erscheinen. Kontinuen bzw. Kontinuumteile, bei denen die Zugehörigkeit sehr unsicher ist, werden in einem Grauton dargestellt. Dies dient dazu, dem Erkenner zusätzliche Informationen über Problemstellen zu geben.

Eine anschließende Wortsegmentierung wurde in diesem Rahmen nicht durchgeführt, da allein bestimmte Abstandswerte zwischen Objekten bei vielen Schriftarten nicht ausreichend sind.

9.2 Ausblick

9.2.1 Verbesserungen und Erweiterungen

Das vom mir hier dargelegte Verfahren zur Vorverarbeitung von Kirchenbuchaufzeichnungen bietet an einigen Stellen noch Potential zur Optimierung.

Automatisiertes Setzen der Parameter

Letztendlich soll ein Programm entstehen, das so wenig wie möglich zusätzliche manuelle Einstellungen erfordert. Dies ist momentan nicht der Fall. Dieser angestrebte

Automatismus kann erreicht werden, indem im Vorfeld aus dem Bild spezifische Eigenschaften der Schrift gewonnen werden, mit deren Hilfe dann die entsprechenden Parameter automatisch angepasst werden können. So wie mit der Gewinnung des Rotationswinkels eine entsprechende Korrektur möglich ist, so ist es auch denkbar, die Größe der Kontinuen und die Abstände der Extrempunkte in horizontaler und vertikaler Richtung auszuwerten, um dann die Toleranz- und Maximalwerte entsprechend zu setzen.

Verfeinerung der Ausgabedaten

Des Weiteren bietet die von mir gewählte Ausgabeform des Grauwertbildes die Möglichkeit einer differenzierteren Wahrscheinlichkeitsangabe über Zugehörigkeit oder Problemstellen, die für jedes Pixel in einer Skala von 0 bis 255 angegeben werden kann.

Verwendung unscharfer Grenzen

Momentan wird mit Toleranzbereichen gearbeitet, die entweder eingehalten wurden oder nicht. Hier könnten Veränderungen in Richtung Fuzzy-Logic helfen ein besseres Ergebnis zu erhalten, indem die Grenzen der Toleranzbereiche durch lineare Funktionen der Abstandswerte gebildet werden. Beispielsweise liefert dann die Anfrage, ob ein Extrempunkt in einem bestimmten Bereich liegt, nicht mehr 0 oder 1, sondern $0,3$ oder $0,8$ – je nachdem, wie weit er vom Zentrum dieses Bereiches entfernt ist.

Verbesserte Analyse von Kreuzungspunkten

Die hier gezeigte Kreuzungspunkt-Analyse hilft nur bei einer relativ kleinen Zahl von Zeilenberührungen. Das schon erwähnte Verfahren von Nakajima et al. könnte auch kompliziertere Kreuzungsfälle lösen [NMTS99]. Durch die Verwendung von B-Splines ließen sich vermutlich Federzüge über mehrere Kreuzungen hinweg verlässlich verfolgen. Es werden hier Hypothesen für den Verlauf eines Spline aufgestellt, die anhand ihrer Krümmung bewertet werden. Eine Verbesserung dieser Bewertung könnte folgendes Verfahren bewirken: Für eine Hypothese, wie in einem Kreuzungspunkt zwei Linien verlaufen könnten, wird mit der Information der Linienbreite für den Kreuzungsbereich eine Pixelmatrix erzeugt, die diesen Bereich im Binärbild repräsentiert, wenn die Federzüge so verlaufen. Durch das Aufsummieren der Differenzpixel mit dem Bereich im Originalbild erhält man ein vergleichbares Qualitätskriterium für die verschiedenen Hypothesen.

Lücken überbrücken

Genauso wie Kreuzungen erkannt und überbrückt werden können, trifft das auch auf Lücken in Federzugsegmenten zu. Das ganze Verfahren inklusive der Erkennung kann

verbessert werden, indem der Verlauf eines Federzugsegments über Lücken hinweg rekonstruiert wird.

Typische Schleifen

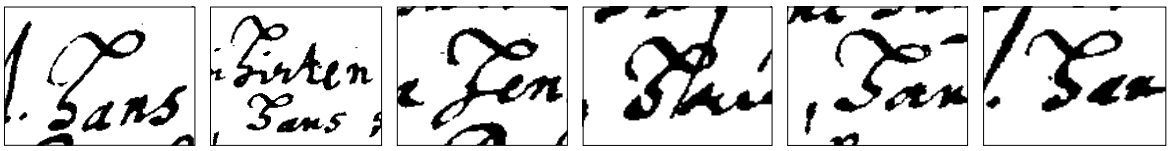


Abbildung 9.1: schreiberspezifische Eigenschaften

Es gibt Eigenschaften, die nur bei einem bestimmten Schreiber auftreten bzw. von Schreiber zu Schreiber variieren (siehe Abbildung 9.1). Hier wäre zwar die Berücksichtigung wünschenswert, es besteht allerdings das Problem, dass diese Merkmale erst einmal extrahiert werden müssen. Ein möglicher Weg wäre der Folgende: Wenn es Speicherplatz und Rechenzeit erlauben, können hypothetischen Splines für eine ganze Textseite gebildet werden. (Wenn sich auf der Seite Schriften unterschiedlicher Schreiber befinden, muss ein entsprechender Bereich gleicher Handschrift von Hand markiert werden.) Durch das Vergleichen bestimmter Parameter, wie z. B. Krümmung und Position zur Textlinie, ließen sich ähnliche Splines finden. Je größer die Zahl der zu einem bestimmten Verlauf ähnlichen Splines ist, umso größer ist die Wahrscheinlichkeit, dass dieser Verlauf korrekt ist.

Unterstreichungen finden

Es wäre wünschenswert, wenn Unterstreichungen auf dem Dokument gefunden werden könnten. Zum Einen könnten sie für die spätere Erkennung als Störung identifiziert und entfernt werden, zum Anderen dienten die Unterstreichungen als Hervorhebung von wichtigen Namen. Auch dadurch könnte eine verbesserte Erkennung erreicht werden.

9.2.2 Zum Handschrifterkenner

Für dieses vorliegende Problem der Kirchenbuchaufzeichnungen ist es erforderlich, dass ein spezieller Erkenner konstruiert wird.

Als Eingangsdaten erhält der Erkenner die isolierten Zeilen eines Dokumentes mit den gewonnenen Informationen über die Textlinien und Problemstellen. Als Ergebnis liefert er eine Reihe von Hypothesen der erkannten Worte mit jeweiligen Wahrscheinlichkeitswerten.

Wenn bei der Segmentierung Zeilenberührungen nicht eindeutig getrennt werden konnten oder sich kleine Objekte nicht eindeutig zuordnen ließen, so wäre ein Ablauf denkbar, bei dem der Erkenner die betreffenden Zeilen mehrmals analysiert und dabei die

unsicheren Objekte einmal mit einbezieht und einmal nicht. Durch Vergleich der entsprechenden Wahrscheinlichkeiten kann so die Erkennung inklusive Objektzuordnung durchgeführt werden.

Auf welchem Datenformat die Erkennung erfolgt, ist von untergeordneter Bedeutung. Bei dem Verfahren von Kim et al. wird bereits in der Vorverarbeitung der Kettencode des Skeletts erzeugt, der in der weiteren Verarbeitung genutzt wird [KGS99]. Bei dem von Côté et al. vorgestellten Handschriftenerkenner-System PERCEPTO werden bestimmte Eigenschaften aus der Kontur der Schrift gewonnen [CLCS98].

Bevor einem Erkenner ein Schriftzug übergeben wird, sind in den meisten Erkennungssystemen noch letzte Vorverarbeitungsschritte notwendig. Dies sind so genannte Normalisierungsschritte wie das Begradigen der Textlinien und das Entfernen der Neigung der Schrift. Hierfür gibt es ausgereifte und gut funktionierende Algorithmen, die prinzipiell auch auf die Kirchenbuchschriften angewandt werden können. Dabei muss allerdings berücksichtigt werden, dass Fehlkorrekturen (die mit hoher Wahrscheinlichkeit auftreten werden) zu einer schlechteren Erkennungsleistung führen. Ich danke dabei z. B. an eine Zeichenhöhenkorrektur, so dass aus einem „d“ oder „h“ fälschlicherweise ein „a“ oder „n“ wird.

Auf dem Gebiet der Handschrifterkenner gibt es zwei unterschiedliche grundsätzliche Ansatzpunkte. Der eine Teil zerlegt ein Wort in kleinere Segmente. Dies sind meistens die Buchstaben. Diese werden einzeln erkannt und danach das komplette Wort zusammengesetzt. Diese Verfahren wird vor allem in der Druckschrifterkennung eingesetzt. Es ist meiner Meinung nach für das hier anstehende Problem nicht gut geeignet. Da die Erscheinung eines Buchstaben in Verbindung mit unterschiedlichen Vorgängern und Nachfolgern stark variieren kann, ist ein anderer Ansatz besser geeignet. Ein anderer Teil der Handschrifterkenner betrachtet das Wort als Einheit und extrahiert daraus bestimmte Eigenschaften. Dieses Verfahren basiert auf Untersuchungen des menschlichen Erkennens handschriftlicher Worte. Es werden beispielsweise Ober- und Unterlängen (Ascenders und Descenders) oder so genannte Täler, die entweder nach oben oder unten offen sind, analysiert [CLCS98]. Dafür ist eine vorherige Bestimmung der Textlinien unentbehrlich.

Speziell bei der Erkennung von Kirchenbüchern ist das relativ kleine Vokabular von Vorteil. Dieses setzt sich aus einer Reihe von Vor-, Nach-, Städte- und Gemeindenamen zusammen, die in unterschiedlicher Schreibweise auftreten können. Hinzu kommen lediglich Datumsangaben, Begriffe über Verwandtschaftsverhältnisse (*„Vater“, „Tochter“, „Großmutter“, ...*) und Verben, die über das Ereignis informieren (*„starb“, „wurde geboren“, „heiratete“, ...*). Mit Hilfe dieser lexikalischen Basis lassen sich auch bei Dokumenten schlechter Qualität sicher gute Ergebnisse erzielen.

Literaturverzeichnis

[CLCS98] M. Côté, E. Lecolinet, M. Cheriet und C. Y. Suen. Automatic Reading of Cursive Scripts Using a Reading Model and Perceptual Concepts. *International Journal on Document Analysis and Recognition*, 1(1):3–17, Februar 1998.

[DD96] L. Duneau und B. Dorizzi. On-Line Cursive Script Recognition: A User-Adaptive System for Word Recognition. *Pattern Recognition*, 29(12):1981–1994, Dezember 1996.

[DIPS97] G. Dimauro, S. Impedovo, G. Pirlo und A. Salzo. Handwriting Recognition: Sate of the Art and Future Trends. In N. A. Murshed und F. Bortolozzi, Hrsg., *Advances in Document Image Analysis – First Brazilian Symposium on Document Analysis, BSDIA'97*, Band 1339 aus *Lecture Notes in Computer Science*, Seiten 1–18. Springer-Verlag, November 1997.

[DSS+90] V. Demjanenko, Y. C. Shin, R. Sridhar, P. Palumbo und S. Srihari. Real-Time Connected Component Analysis for Address Block Location. In *Proceedings, USPS Fourth Advanced Technology Conference*, Band 3, Seiten 1059–1071, 1990.

[Ebe00] A. Ebeling. Lesestunde – Fünf OCR-Klassiker im Vergleich. *c't Magazin für Computertechnik*, 2000(4):196–200, Februar 2000.

[EYGSS97] A. El-Yacoubi, M. Gilloux, R. Sabourin und C. Y. Suen. Objective Evaluation of the Discriminant Power of Features in an HMM-Based Word Recognition System. In N. A. Murshed und F. Bortolozzi, Hrsg., *Advances in Document Image Analysis – First Brazilian Symposium on Document Analysis, BSDIA'97*, Nummer 1339 in Lecture Notes in Computer Science, Seiten 60–73. Springer-Verlag, November 1997.

[FK97] K. Franke und M. Köppen. Towards an Universal Approach to Background Removal in Images of Bankchecks. In S.-W. Lee, Hrsg., *Advances in Handwriting Recognition*, Seiten 91–100. World Scientific, Februar 1997.

[Fre61] H. Freeman. On the Encoding of Arbitrary Geometric Configurations. In *IRE Transactions on Electronic Computers*, Band EC–10, Seiten 260–268, Juni 1961.

[GK96] V. Govindaraju und R. K. Krishnamurthy. Holistic Handwritten Word Recognition Using Temporal Features Derived From Off-line Images. *Pattern Recognition Letters*, 17(5):537–540, Mai 1996.

[HLB00] J. Hu, S. G. Lim und M. K. Brown. Writer Independent On-line Handwriting Recognition Using an HMM Approach. *Pattern Recognition*, 33(1):133–147, Januar 2000.

[KGS99] G. Kim, V. Govindaraju und S. N. Srihari. An Architecture for Handwritten Text Recognition Systems. *International Journal on Document Analysis and Recognition*, 2(1):37–44, Februar 1999.

[Liu92] Kun Liu. *On Chain Coding of Line Drawings*. Dissertation, Delft University Press, 1992.

[MG99] S. Madhvanath und V. Govindaraju. Local Reference Linies for Handwritten Phrase Recognition. *Pattern Recognition*, 32:2021–2028, 1999.

[MKG99] S. Madhvanath, G. Kim und V. Govindaraju. Chain Code Processing for Handwritten Word Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):928–932, 1999.

[NMTS99] Y. Nakajima, S. Mori, S. Takegami und S. Sato. Global Methods for Stroke Segmentation. *International Journal on Document Analysis and Recognition*, 2(1):19–23, 1999.

[PAO99] T. Paquet, M. Avila und C. Olivier. Word Modeling for Handwritten Word Recognition. In *Vision Interface '99*, Seiten 49–56. Trois-Rivières, Mai 1999.

[PS98] Y. Pu und Z. Shi. A Natural Learning Algorithm Based on Hough Transform for Text Lines Extraction in Handwritten Documents. In *Proceedings of the Sixth International Workshop on Frontiers of Handwriting Recognition (IWFHR VI),Taejon, Korea*, Seiten 637–346, 1998.

[PS00] Réjean Plamondon und Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, Januar 2000.

[Sab97] R. Sabourin. Off-Line Signature Verification: Recent Advances and Perspectives. In N. A. Murshed und F. Bortolozzi, Hrsg., *Advances in Document Image Analysis – First Brazilian Symposium on Document Analysis, BSDIA'97*, Band 1339 aus *Lecture Notes in Computer Science*, Seiten 84–98. Springer-Verlag, November 1997.

[SK97] M. Shridar und F. Kimura. Segmentation-Based Cursive Handwriting Recognition. In H. Bunke und P. S. P. Wang, Hrsg., *Handbook of Character Recognition and Document Image Analysis*, Seiten 123–156. World Scientific, Februar 1997.

[SL99] R. Seetzen und C. Loebich. Chemische Alternative. *c't Magazin für Computertechnik*, 1999(22):158–161, 1999.

[SLG+96] C. Y. Suen, L. Lam, D. Guillevic, N. W. Strathy, M. Cheriet, J. N. Said und R. Fan. Bank Check Processing System. *International Journal of Imaging Systems and Technology*, 7:392–403, 1996.

[Ste95] Peter Steiner. Zwei ausgewählte Probleme zur Offline-Erkennung von Handschrift. Diplomarbeit, Universität Bern, August 1995.

[SXL99] C. Y. Suen, Q. Xu und L. Lam. Automatic Recognition of Handwritten Data on Cheques – Fact or Fiction? *Pattern Recognition Letters*, 20:1287–1295, 1999.

[TCL+99] Y. Y. Tang, M. Cheriet, J. Liu, J. N. Said und C. Y. Suen. Document Analysis and Recognition by Computers. In C. H. Chen, L. F. Pau und P. S. P. Wang, Hrsg., *Handbook of Pattern Recognition & Computer Vision*, Seiten 579–612. World Scientific, London, 2. Auflage, 1999.

[TYCS91] Y. Y. Tang, C. D. Yan, M. Cheriet und C. Y. Suen. Document Analysis and Understanding: A Brief Survey. In *Proceedings of First International Conference on Document Analysis and Recognition*, Seiten 17–31, Saint-Malo, France, 1991. Februar.

[WMR99] V. Wu, R. Manmatha und E. M. Riseman. TextFinder: An Automatic System to Detect and Recognize Text In Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1224–1229, November 1999.

Anhang A

Testergebnisse

Tabelle A.1: Testlauf mit konstanten Parametern

Dokumentenseite	Textblock-Nr.	Anzahl der Zeilen	kleine Fehler	große Fehler
S. 110 von 1649	1	4	0	0
	2	6	1	0
	3	4	0	0
	4	4	2	0
	5	4	0	0
S. 111 von 1649	1	4	0	0
	2	6	1	0
	3	4	0	0
	4	4	0	0
	5	5	0	0
S. 16 von 1812	1	8	2	0
	2	8	0	0
	3	9	1	0
	4	9	0	1
	5	8	0	0
S. 30 von 1817	1	3	0	0
	2	4	0	0
	3	6	1	0
	4	4	0	0
	5	3	1	0
	6	4	0	0
	7	3	0	0
	8	5	0	1
	9	3	0	0
	10	6	0	0
	11	4	0	0
	12	5	1	0
	13	6	0	1
	14	3	0	0
	15	5	1	0
S. 9 von 1838	1	5	0	0

Fortsetzung nächste Seite...

Dokumentenseite	Textblock-Nr.	Anzahl der Zeilen	kleine Fehler	große Fehler
	2	5	0	0
	3	5	0	0
	4	4	0	0
	5	5	0	1
	6	5	0	1
	7	4	0	0
	8	5	1	0
	9	6	1	0
	10	5	0	0
	11	4	1	0
	12	6	0	0
	13	6	1	0
	14	4	0	1
	15	6	1	0
	16	5	0	0
	17	6	0	1
	18	5	0	0
	19	4	1	0

Tabelle A.2: Testlauf mit angepassten Parametern

Dokumentenseite	**Textblock-Nr.**	**Anzahl der Zeilen**	**kleine Fehler**	**große Fehler**
S. 110 von 1649	1	4	0	0
	2	6	1	0
	3	4	0	0
	4	4	0	0
	5	4	0	0
S. 111 von 1649	1	4	0	0
	2	6	0	0
	3	4	0	0
	4	4	0	0
	5	5	0	0
S. 18 von 1727	1	3	0	0
	2	3	0	0
	3	3	0	0
	4	4	0	0
	5	4	0	1
	6	3	1	0
S. 240 von 1741	1	3	0	0
	2	7	0	1
	3	10	0	0
	4	4	0	0
	5	7	1	0
	6	3	0	0
S. 16 von 1812	1	8	0	0
	2	8	0	0
	3	9	0	0
	4	9	0	1
	5	8	0	0
S. 30 von 1817	1	3	0	0
	2	4	0	0
	3	6	0	0
	4	4	0	0
	5	3	0	0
	6	4	0	0
	7	3	0	0
	8	5	0	0
	9	3	0	0
	10	6	0	0
	11	4	0	0
	12	5	0	0
	13	6	0	0

Fortsetzung nächste Seite...

Dokumentenseite	Textblock-Nr.	Anzahl der Zeilen	kleine Fehler	große Fehler
	14	3	0	0
	15	5	0	0
S. 9 von 1838	1	5	0	0
	2	5	0	0
	3	5	0	0
	4	4	0	0
	5	5	0	1
	6	5	0	0
	7	4	0	0
	8	5	0	0
	9	6	1	0
	10	5	0	0
	11	4	0	0
	12	6	0	0
	13	6	0	0
	14	4	0	0
	15	6	0	0
	16	5	0	0
	17	6	0	0
	18	5	0	0
	19	4	1	0

Anhang B

Beispielbilder: Phasen der Textlinienfindung

Am Beispiel eines Ausschnittes der Seite 16 von 1812 werden die einzelnen Phasen der Textlinienfindung illustriert. Sie entstanden durch Kombination der von VECTORY erzeugten DXF-Dateien mit dem Binärbild.

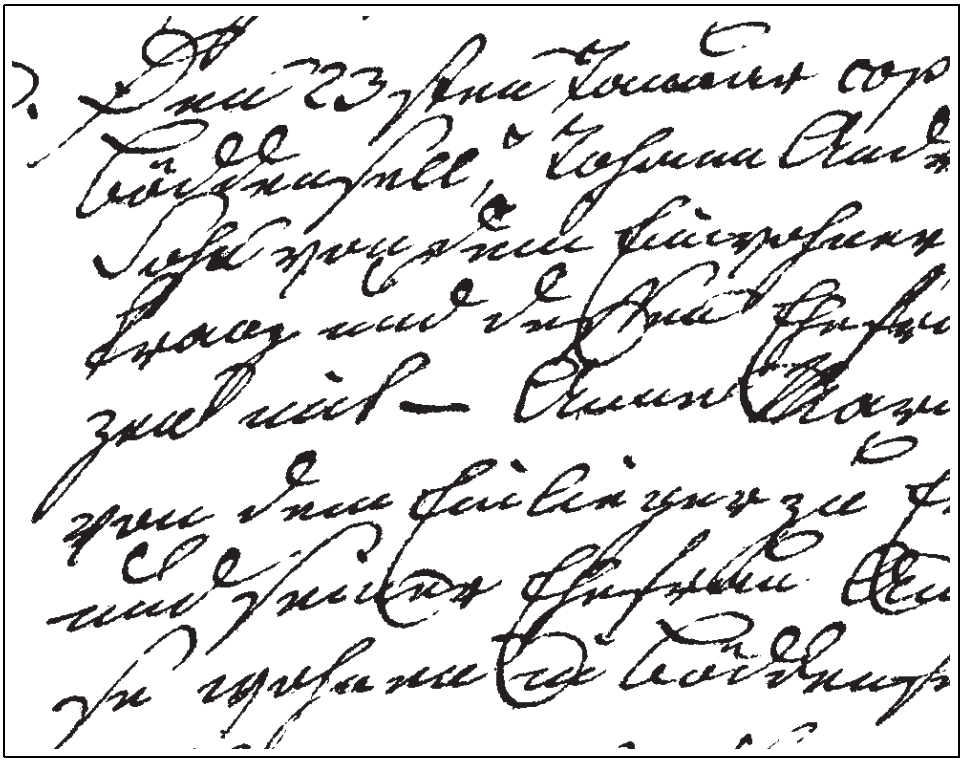


Abbildung B.1: binäres Ausgangsbild

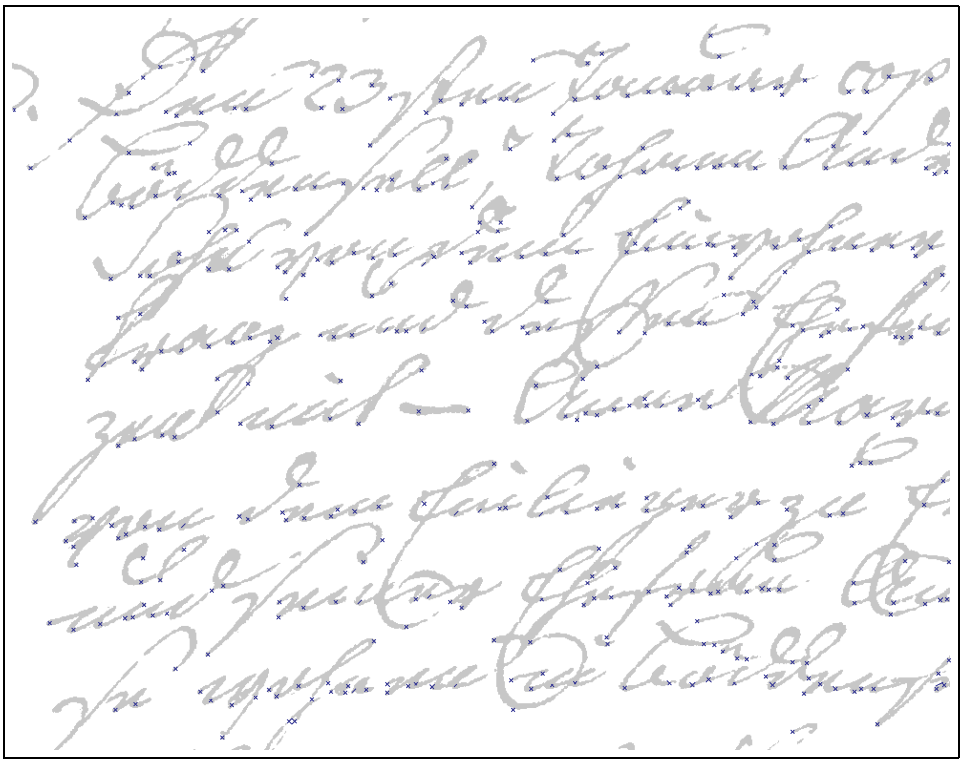


Abbildung B.2: 442 lokale Minimumpunkte

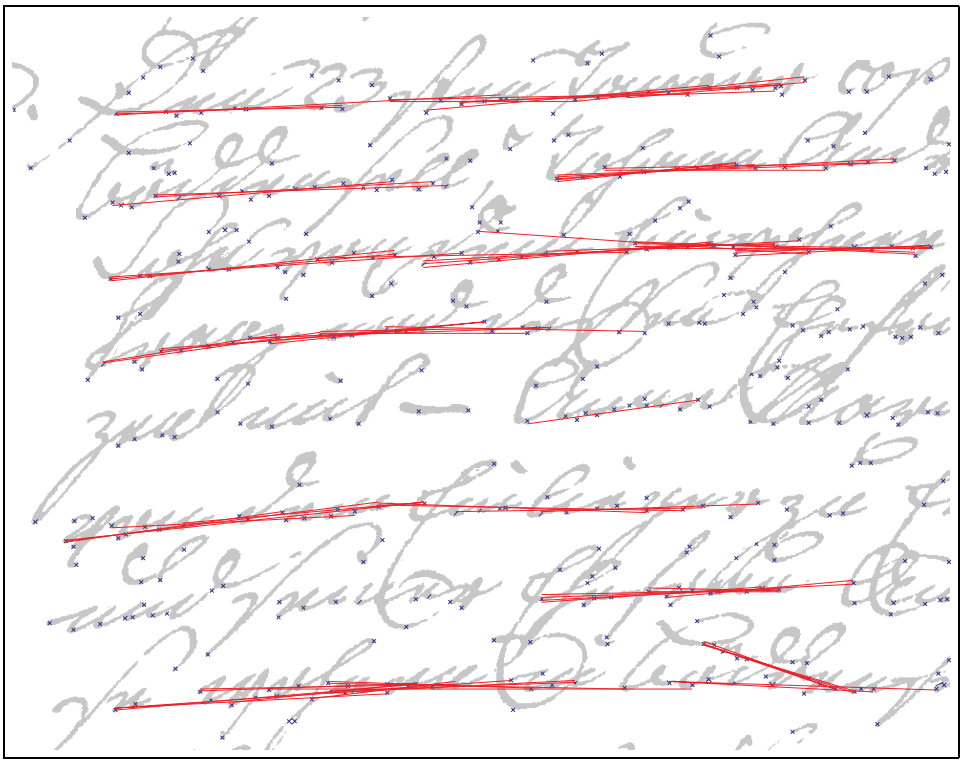


Abbildung B.3: 83 lange Segmente zur Rotationsbestimmung

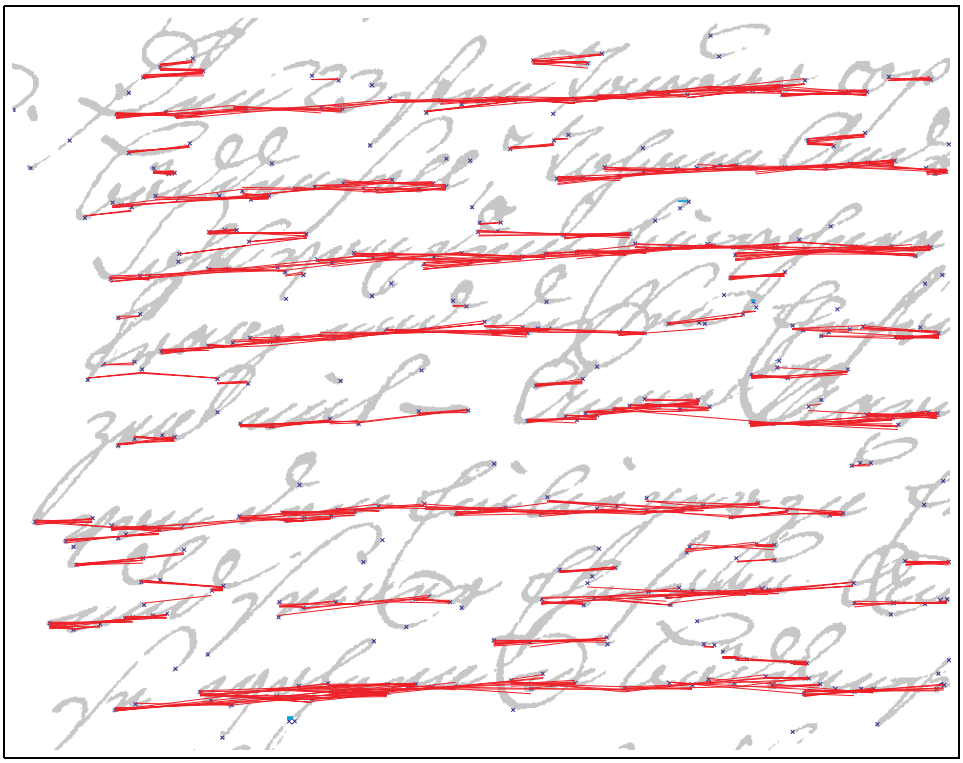


Abbildung B.4: 1267 Liniensegmente bei einem Winkelbereich von ±6°

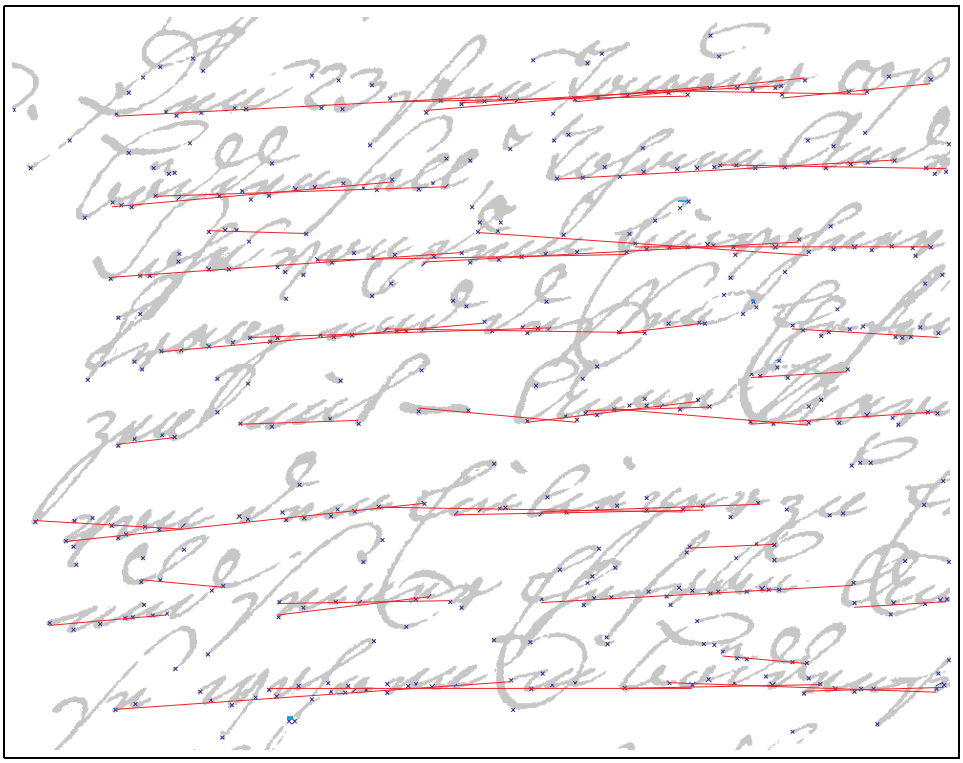


Abbildung B.5: 48 Liniensegmente nach der Reduzierung

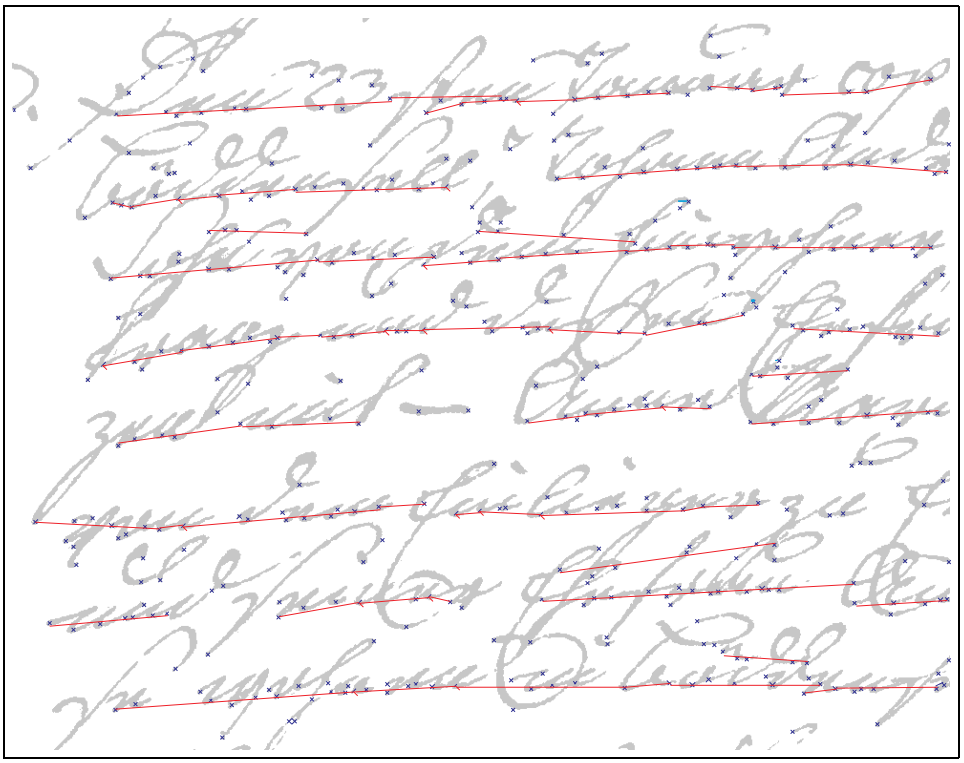


Abbildung B.6: 72 Liniensegmente nach dem Verschmelzen

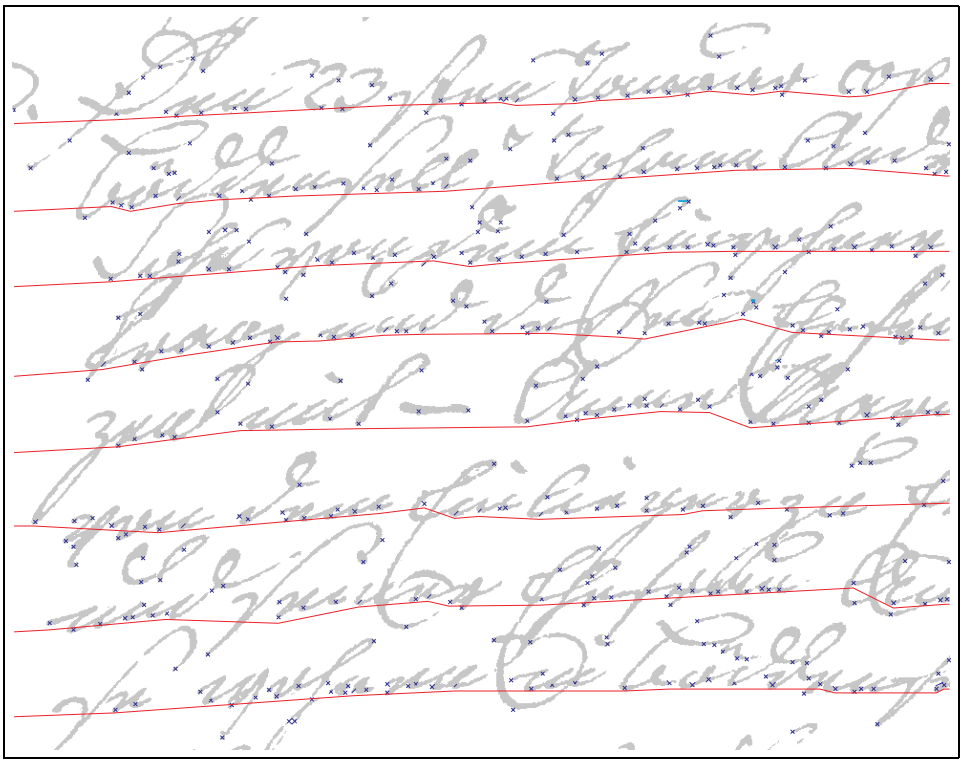


Abbildung B.7: die 8 gefundenen Basislinien

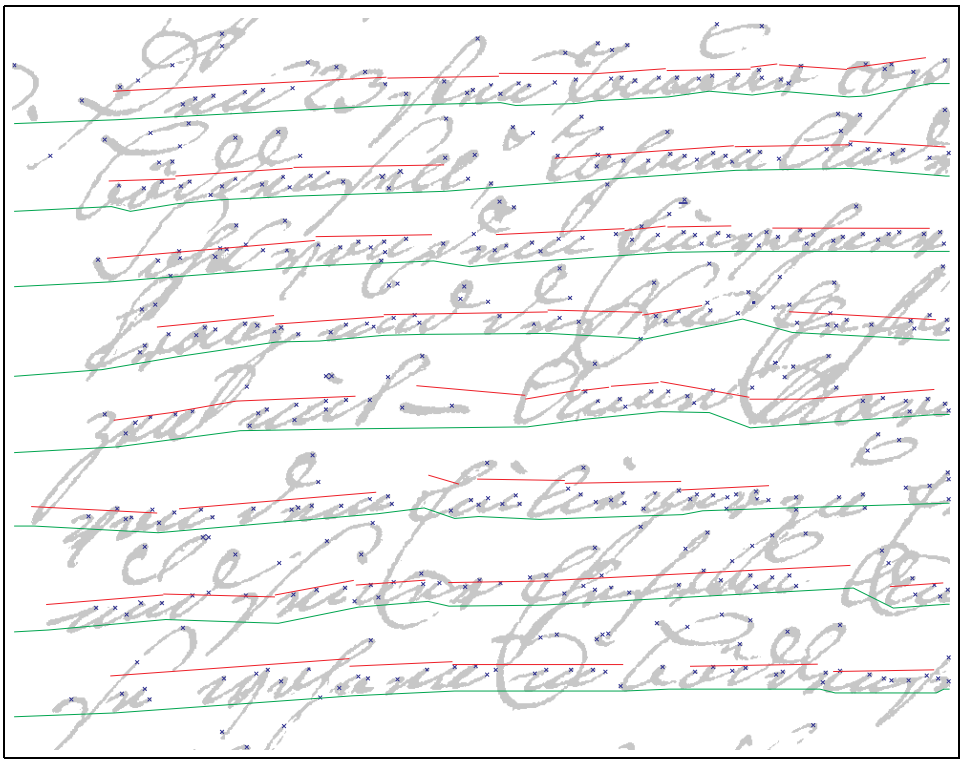


Abbildung B.8: Basislinien, Maximumpunkte und 53 gefundene Mittelliniensegmente

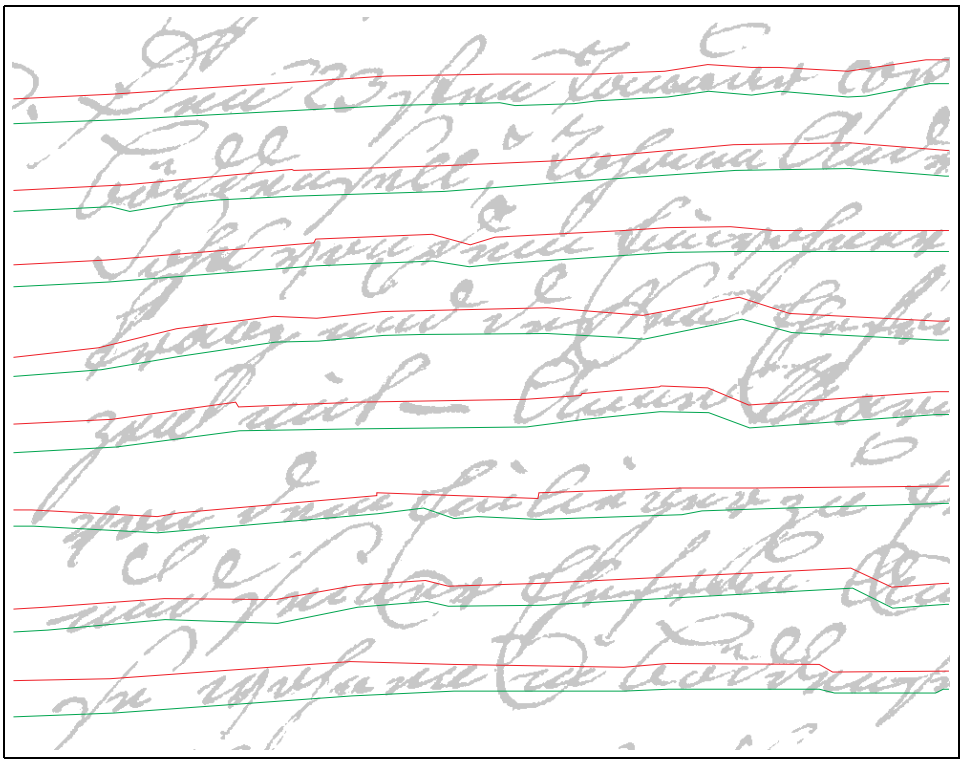


Abbildung B.9: Basislinie und Mittellinie

Anhang C

Beispielbilder: Textlinienfindung

Als Beispiel für die korrekte Zeilenfindung wird im Folgenden von jeder Schriftart ein Textblock dargestellt.

Nach dem Erzeugen der Textlinien gibt VECTORY die Möglichkeit, diese im gängigen DXF-Format zu exportieren. Zur Darstellung habe ich diese Dateien mit den Binärbildern kombiniert.

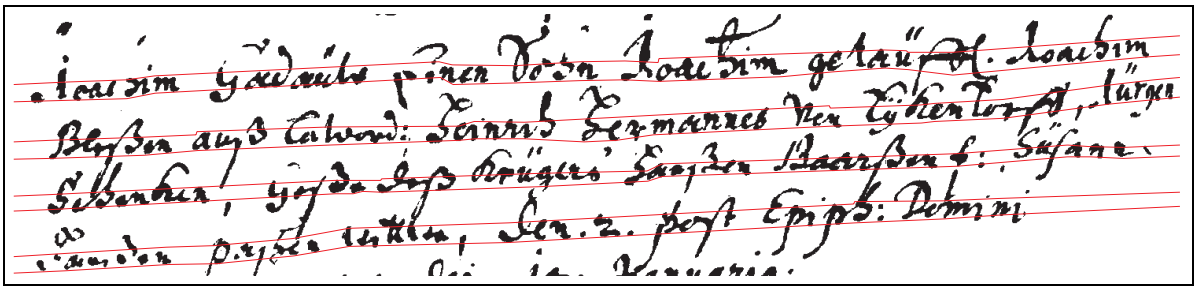


Abbildung C.1: 3. Textblock der Seite 110 von 1649

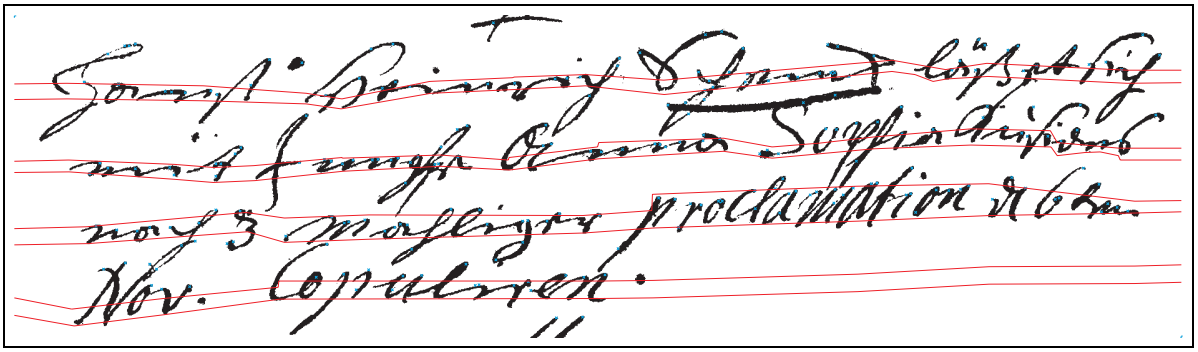


Abbildung C.2: 4. Textblock der Seite 18 von 1727

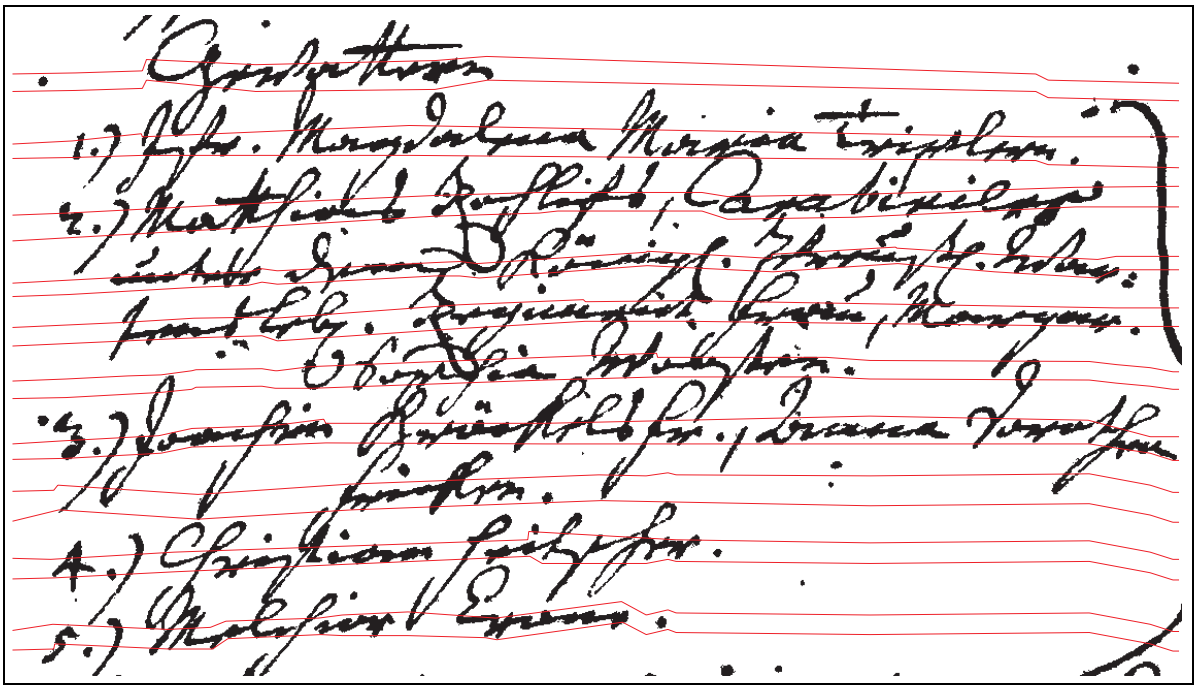


Abbildung C.3: 3. Textblock der Seite 240 von 1741

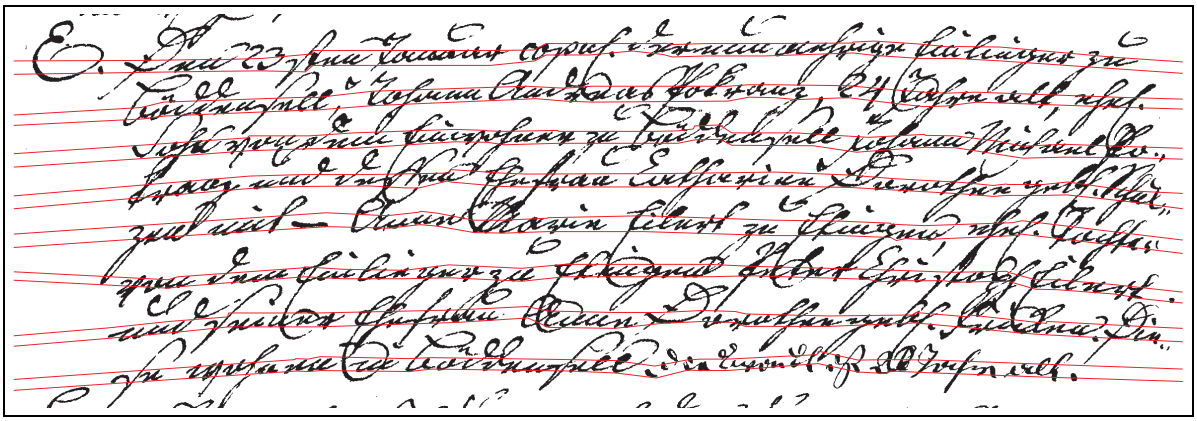


Abbildung C.4: 2. Textblock der Seite 16 von 1812

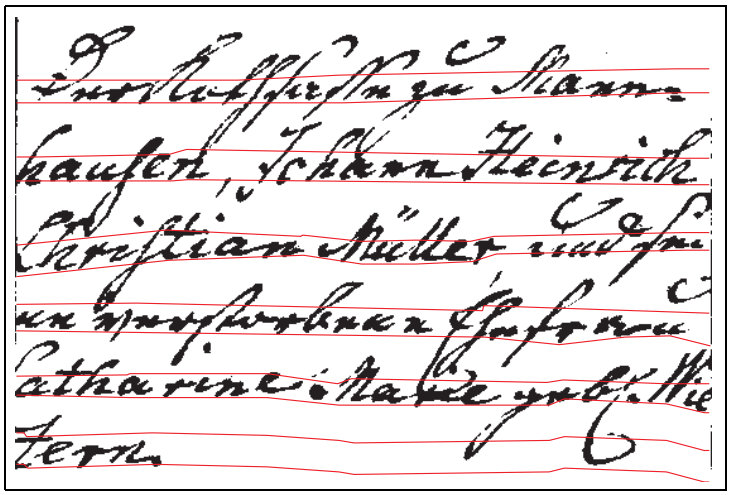


Abbildung C.5: Tabellenfeld: 6. Zeile, 2. Spalte der Seite 30 von 1817

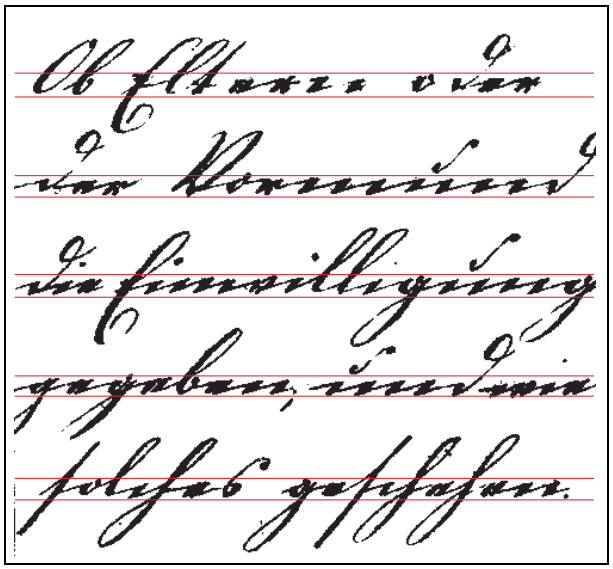


Abbildung C.6: Tabellenfeld: 1. Zeile, 3. Spalte der Seite 9 von 1838

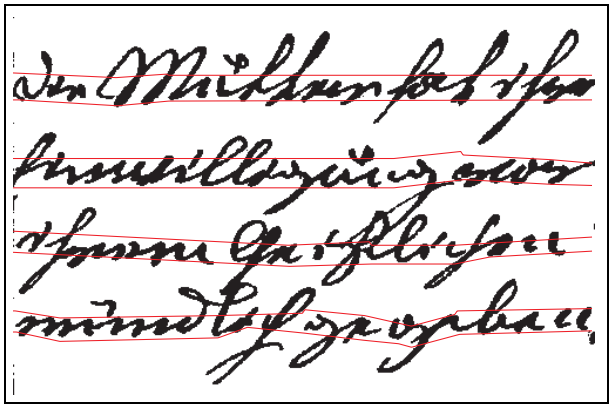


Abbildung C.7: Tabellenfeld: 4. Zeile, 4. Spalte der Seite 9 von 1838