

**Magisterarbeit**

Zur Erlangung des akademischen Grades  
Magister Artium  
durch den Fachbereich  
Historisch-Kulturwissenschaftliche Informationsverarbeitung  
der Universität zu Köln

**Zur Erweiterungsfähigkeit bestehender OCR Verfahren  
auf den Bereich extrem früher Drucke**

Vorgelegt von

Jan Gerrit Wieners  
Peter-Warnecke-Weg 8  
51061 Köln

Köln, den 29. August 2008

Gutachter: Prof. Dr. Manfred Thaller

## Inhaltsverzeichnis

<b>1. Einleitung: Digitale Bibliotheken und Digitalisierungsprojekte .....</b>	<b>4</b>
<b>2. Automatische Texterkennung .....</b>	<b>8</b>
<b>3. Der Bereich extrem früher Drucke: Inkunabeln.....</b>	<b>10</b>
3.1. Die verteilte digitale Inkunabelbibliothek .....	11
3.2. Inkunabeln, Herausforderung für OCR .....	12
<b>4. „TED“: Ein Beitrag zur OCR von Inkunabeln .....</b>	<b>14</b>
4.1. „TED“: Intention und Workflow .....	14
<b>5. „TED“: Algorithmen und Arbeitsweise .....</b>	<b>15</b>
5.1. Vorverarbeitung .....	15
5.1.1. Farb- und Layoutanalyse .....	16
5.1.2. Umwandlung Farbbild zu Graustufenbild .....	18
5.1.3. Graustufenumwandlung: das IHS Farbmodell .....	19
5.1.4. Bild zuschneiden .....	20
5.1.5. Automatischer Histogrammausgleich .....	21
5.1.6. Medianfilter .....	23
5.2. Segmentierung: Schwellwertverfahren.....	26
5.2.1. Segmentierung über globalen Schwellwert.....	27
5.2.2. Histogrammbasierte Segmentierung: Otsu.....	28
5.2.3. Iterative Auswahl: Ridler, Calvard, Thrussel.....	30
5.2.4. Evaluation und Wahl des bevorzugten Algorithmus.....	30
5.3. Vorverarbeitung II .....	35
5.3.1. Erodierung .....	35
5.3.2. <i>k</i> Fill Filter .....	35
5.4. Skew-Korrektur .....	37
5.4.1. Glyphen und Grundlinien: Das Verfahren von Baird.....	38
5.4.2. Hough-Transformation .....	39
5.5. Objektwahrnehmung.....	41
5.5.1. Kantenerkennung und Kantenisolierung .....	41
5.5.2. Kantenverfolgung: Pfadfinder .....	43

5.5.3.	Speicherung von Objektinformation .....	44
5.6.	Glyphenbündelung und Glyphenerkennung .....	46
5.6.1.	Exkurs: Künstliche Neuronale Netze .....	46
5.7.	Glyphenbündelung: Selbstorganisierende Karten .....	49
5.7.1.	Trainings- und Lernprozess der SOM .....	50
5.7.2.	Implementation der SOM in „TED“ .....	52
5.7.3.	SOM: Normalisierung der Eingabemuster .....	52
5.7.4.	SOM: Training .....	53
5.7.5.	„TED“ / SOM: Ergebnisse und Bewertung.....	54
5.8.	Mustererkennung: mehrschichtiges Perzeptron.....	55
5.8.1.	Mehrschichtiges Perzeptron: Definition und Arbeitsweise.....	55
5.8.2.	Mehrschichtiges Perzeptron: Implementation in „TED“ .....	56
5.8.3.	MLP und „TED“: Evaluation .....	58
5.9.	Aufbereitung der OCR-Ergebnisse.....	60
<b>6.</b>	<b>„TED“ – TED Enhances Digitization .....</b>	<b>62</b>
6.1.	„TED“ – Workflow.....	63
<b>7.</b>	<b>Fazit und Ausblick.....</b>	<b>67</b>
<b>8.</b>	<b>Literaturverzeichnis .....</b>	<b>68</b>
<b>9.</b>	<b>Verzeichnis externer Ressourcen .....</b>	<b>73</b>
	<b>Appendix I: C++ Quellcode.....</b>	<b>74</b>
1.	Kantenisolierung ( <i>./TED/src/edge_detection/grad1.cpp</i> ).....	74
2.	Konvertierung Tiff nach ASCII (Auszug aus <i>./TED/src/nmmlp.cpp</i> ) .	75
	<b>Appendix II – Inhalt der DVD .....</b>	<b>78</b>
	<b>Appendix III: Abbildungsverzeichnis .....</b>	<b>79</b>
	<b>Erklärung.....</b>	<b>81</b>

## 1. Einleitung: Digitale Bibliotheken und Digitalisierungsprojekte

Schon im siebzehnten Jahrhundert ist das gedruckte Buch der souveräne Herrscher, unumschränkt, siegreich und im Siege befestigt genug, um der Welt das große wissenschaftliche Jahrhundert bieten zu können. [...] Seit drei Jahrhunderten spiegelt so die Buchdruckerkunst den menschlichen Gedanken wider, um ihn auszudrücken und weiterzugeben in tiefgehender allseitiger Bewegung. Ohne Riß und Lücke lagert sie über dem Menschengeschlechte, diesem ungeheuren, stets nach vorne drängenden Tausendfüßler.<sup>1</sup>

Wo Victor Hugo in seinem „Der Glöckner von Notre Dame“ achtungserfüllte Gedanken über den Bestand des Buchdrucks über die Fortdauer materieller Konstrukte der Baukunst formuliert, lässt sich erahnen, welche herausragende Stellung die Erfindung und Entwicklung des Buchdrucks in der geistesgeschichtlichen Entwicklung inne hat. Der Mensch, dieser nach vorne drängende „Tausendfüßler“, wie ihn Victor Hugo so pittoresk illustriert, scheint sich auch heute noch, rund 550 Jahre nach dem langsam, aber stetig erblühenden Buchdruck und mehr als 170 Jahre nach Veröffentlichung des „Glöckner von Notre Dame“, der tiefgreifenden Bedeutung bewusst zu sein, die das Buch verwahrt.

So macht die im Januar 2007 vom Fraunhofer Institut für Intelligente Analyse- und Informationssysteme veröffentlichte „Bestandsaufnahme zur Digitalisierung von Kulturgut und Handlungsfelder“<sup>2</sup> das Kulturgut „Buch“ zum Hauptbestandteil ihrer Untersuchung und weist auf den „unschätzbare[n] materielle[n] und ideelle[n] Wert“<sup>3</sup> hin, der sich in deutschen und europäischen Bibliotheken, Archiven und Museen verbirgt – und ruft die Vergänglichkeit von materiell manifestiertem Kulturgut in Erinnerung: Freilich ist das Buch – analog

---

<sup>1</sup> Hugo, Victor: Der Glöckner von Notre-Dame. Stuttgart, Hamburg, München: Deutscher Bücherbund; S. 191.

<sup>2</sup> Fraunhofer Institut für Intelligente Analyse- und Informationssysteme: Bestandsaufnahme zur Digitalisierung von Kulturgut und Handlungsfelder. Erstellt im Auftrag des Beauftragten für Kultur und Medien (BMK) unter finanzieller Beteiligung des Bundesministerium für Bildung und Forschung, Januar 2007.

<<http://www.bundesregierung.de/Content/DE/Artikel/2007/01/anlagen/2007-01-11-fraunhofer-studie-pdf-format,property=publicationFile.pdf>> (27.08.2008)

<sup>3</sup> Ebenda, S. 18.

den Produkten der Baukunst – der Endlichkeit untergeben; Bücher vergehen, sie verwelken zwar, ohne jedoch – sofern ihre Intention nicht in Vergessenheit gerät – ihre Blüte zu verlieren.

„Originale und Handschriften, [...] Nachdrucke und Kopien mit einem hohen kulturellen Wert“<sup>4</sup>, auch Dokumente in hoher Auflage stellt die Studie vor als Materialfundus, der vor dem nagenden Zahn der Zeit bewahrt werden muss. Der praktikable und der Sache angemessene Weg, um das Kulturgut zu konservieren, ist die Digitalisierung. Als das „wichtigste Instrument zur Sicherung dieser wertvollen Kulturgüter Deutschlands und deren Vermittlung mit aktuellen Medien“<sup>5</sup> soll die Digitalisierung – neben der Konservierung – das Kulturgut „weltweit für jedermann, jederzeit“<sup>6</sup> über das Internet zugänglich machen.

Digitalisierung (engl. „Digitization“ oder auch „Digitalization“) bezeichnet das Verfahren, über ein Bildaufnahmesystem, wie es durch einen Scanner oder eine Kamera bereitgestellt wird, abzubildendes Material (z.B. eine Buchseite, eine Photographie, ein Bild) in ein digitales Bild umzuwandeln, das vom Rechner gespeichert, bearbeitet und verarbeitet werden kann.<sup>7</sup> Das „Digitalisat“ ist das Resultat der Umwandlung und bezeichnet eine zweidimensionale Matrix, auch als „Rasterbild“ bezeichnet, deren einzelne Elemente (auch als „Bildpunkte“ oder „Pixel“ bezeichnet) die Farbwerte des Bildes repräsentieren.

Die Zahl der Digitalisierungsprojekte allein in Deutschland ist hoch – Digitalisierung erfreut sich großer Beliebtheit und ist ein Thema, das laut der Studie des Fraunhofer Institut für Intelligente Analyse- und Informationssysteme noch einige Jahrzehnte aktuell sein wird<sup>8</sup>. Die in Göttingen und München im Rahmen des Förderprojektes „Retrospektive Digitalisierung von

---

<sup>4</sup> Ebenda.

<sup>5</sup> Ebenda.

<sup>6</sup> Ebenda, S. 31.

<sup>7</sup> Für eine ausführliche Beschreibung der Komponenten eines Bildverarbeitungssystems sei hingewiesen auf: Jähne, Bernd: Digitale Bildverarbeitung. Berlin, Heidelberg: Springer-Verlag 2005; S. 19-27.

<sup>8</sup> Vgl. Fraunhofer Institut für Intelligente Analyse- und Informationssysteme: Bestandsaufnahme zur Digitalisierung von Kulturgut und Handlungsfelder. Erstellt im Auftrag des Beauftragten für Kultur und Medien (BMK) unter finanzieller Beteiligung des Bundesministerium für Bildung und Forschung, Januar 2007; S. 44.

Bibliotheksbeständen“<sup>9</sup> eingerichteten Kompetenz- und Digitalisierungszentren bündeln das Wissen um Digitalisierungsprozesse und bieten Hilfestellung bei Digitalisierungsvorhaben. Da immer mehr Museen, Bibliotheken und Archive ihr Kulturgut digitalisieren und der interessierten Öffentlichkeit zugänglich machen, formt sich (mitunter) der Wunsch nach einem zentralen Portal, das Fortschritte und Material verteilter Digitalisierungsprojekte bündelt, wie erst jüngst in der Mitteilung der EU-Kommission mit dem Portal „Europeana“ formuliert:

„Mit Hilfe von Europeana können Nutzer digitalisierte Materialien aus Museen, Archiven, Bibliotheken sowie Ton- und Bildarchiven europaweit durchsuchen und kombinieren, ohne dazu verschiedene Websites kennen oder aufsuchen zu müssen. Sie können unmittelbar auf digitalisierte Bücher, Zeitungen, Archivmaterial, Fotografien und Ton- und Bilddateien zugreifen und diese in ihrer Freizeit oder für Beruf und Studium konsultieren und nutzen.“<sup>10</sup>

Einzig ist der Nutzen von Digitalisaten beschränkt, sofern die Digitalisate nicht in einen Kontext eingebettet sind, der sie auflegen lässt, ihnen gerecht wird. Metadaten, also „Daten, die strukturierte Information über andere Daten enthalten und die diese damit in Informationssystemen besser auffindbar machen“<sup>11</sup>, erweitern das Informationsangebot der reinen Bilddatei des Digitalisates – und auch die Volltextdigitalisierung ist wünschenswert, um den Text des Digitalisates zu extrahieren und durchsuchbar zu machen. Die vage Formulierung „wünschenswert“ deutet hierbei voraus auf die Problematik der Volltextdigitalisierung: Die manuelle Erschließung ist zeit- und kostenintensiv<sup>12</sup>.

Die manuelle Arbeit am Text des Digitalisates zu reduzieren – und somit Zeit und Kosten zu minimieren – sucht die automatische Texterkennung. Die folgenden

---

<sup>9</sup> Eine Übersicht über Retrodigitalisierungs-Projektserver des DFG-Förderschwerpunktes „Retrospektive Digitalisierung von Bibliotheksbeständen“ ist zu finden unter <<http://www.hki.uni-koeln.de/retrodig/>> (27.08.2008)

<sup>10</sup> Kommission der Europäischen Gemeinschaften: Europas kulturelles Erbe per Mausklick erfahrbar machen, Stand der Digitalisierung und Online-Verfügbarkeit kulturellen Materials und seiner digitalen Bewahrung in der EU. Brüssel, 11.08.2008; S. 3. <[http://ec.europa.eu/information\\_society/activities/digital\\_libraries/doc/communications/progress/communication\\_de.pdf](http://ec.europa.eu/information_society/activities/digital_libraries/doc/communications/progress/communication_de.pdf)> (27.08.2008)

<sup>11</sup> Thaller, Manfred et. al.: Retrospektive Digitalisierung von Bibliotheksbeständen - Evaluierungsbericht über einen Förderschwerpunkt der DFG. Köln, 2005; S. 27. <[http://www.dfg.de/forschungsfoerderung/wissenschaftliche\\_infrastruktur/lis/download/retro\\_digitalisierung\\_eval\\_050406.pdf](http://www.dfg.de/forschungsfoerderung/wissenschaftliche_infrastruktur/lis/download/retro_digitalisierung_eval_050406.pdf)> (27.08.2008)

<sup>12</sup> Für die Evaluation der Digitalisierungskosten vgl. ebenda, S. 25-29.

---

Kapitel erläutern die automatische Zeichen- und Texterkennung, berichten über die „verteilte digitale Inkunabelbibliothek“ und betrachten schließlich den Problemkreis der automatischen Texterkennung auf extrem frühen Druckerzeugnissen am Arbeitsbeispiel von Inkunabeln.

## 2. Automatische Texterkennung

Der Begriff „automatische Texterkennung“ bezeichnet den Komplex der computergestützten automatischen Erkennung kleinster bedeutungstragender Textpartikel in Rasterbildern und ihre Transformation in einen Gesamtkontext, den Text. Auch als „Optical Character Recognition“ (OCR) bezeichnet, lässt sich bei der automatischen Texterkennung unterscheiden zwischen „on-line“ und „off-line“ Erkennung. Kommt die on-line Erkennung durch ihren Fokus auf die zeitliche Entstehungskomponente von Handschriften vorwiegend auf mobilen Geräten zum Einsatz, so arbeitet die off-line Erkennung auf den zweidimensional vorliegenden Daten von Rasterbildern.

Automatische Texterkennung ist kein isoliertes Verfahren, sondern konstituiert sich durch eine Kette von Prozessen, die den Erkennungsvorgang einleiten, durchführen und abschließen. So bereitet die Vorverarbeitungsphase das Quellenmaterial auf die folgenden Prozesse vor, unterteilt die Objektwahrnehmungsphase das Material in Einheiten, die dem Prozess der Erkennung von Merkmalen zugeführt werden, und transformiert die Erkennungsphase Bildinformation in Textinformation. Die folgenden Erläuterungen skizzieren zentrale Phasen der automatischen Texterkennung<sup>13</sup>. Für detaillierte Informationen sei an dieser Stelle auf die entsprechenden Kapitel der vorliegenden Arbeit verwiesen.

Die Vorverarbeitungsphase legt die Weichen für die Qualität aller folgenden Prozesse. In dieser ersten Phase der automatischen Texterkennung wird die Farbenvielfalt des Digitalisates reduziert auf ein schwarz-/weiß-Bild (auch als „Monochrombild“ bezeichnet), das von überflüssiger Information befreit ist und ausschließlich bedeutungstragende Elemente enthält. Ergänzt wird die Farbreduktion um Arbeitsschritte, die Bildstörungen und Rauschen beseitigen, fehlerhaft ausgerichtetes Quellenmaterial drehen („Skew Correction“) und

---

<sup>13</sup> Einen umfangreichen Überblick über OCR und Handschriftenerkennung bietet: Feldmann, Bianca: OCR von Handschriften. Ein Forschungsüberblick. In: Thaller, Manfred (Hrsg.): Codices Electronici Ecclesiae Coloniensis. Eine mittelalterliche Kathedralbibliothek in digitaler Form. Göttingen: Fundus - Forum für Geschichte und ihre Quellen. Beiheft 1, 2001 <<http://webdoc.gwdg.de/edoc/p/fundus/1/feldmann.pdf>> (27.08.2008)



ungleichmäßige Ausleuchtung korrigieren, die aus dem Scan- oder Photographievorgang resultiert.

Die Arbeitsphase der Objektwahrnehmung macht sich das vorbereitete Quellenmaterial zu Nutze und bündelt Pixelinformation zu bedeutungstragenden Einheiten; das Resultat sind Buchstaben und Wörter. Im Folgenden wird der Begriff „Glyph“ verwendet, um die graphische Darstellung einzelner Zeichen oder Symbole zu deklarieren.<sup>14</sup> Auf die Objektwahrnehmung folgt die eigentliche Erkennungsphase: Glyphen werden erkannt und mit entsprechenden Zeichen aus einem Zeichensatz, beispielsweise dem ASCII-Code<sup>15</sup>, belegt. Der letzte Arbeitsschritt besteht darin, die erkannten Zeichen und Wörter zu präsentieren. Hierbei bietet sich – je nach Verwendungszweck – die Abbildung der erkannten Information als Webdokument über die Auszeichnungssprache XHTML an.

Die vorliegende Ausarbeitung beschäftigt die Frage, inwiefern sich Verfahren der automatischen Texterkennung auf extrem frühe Drucke anwenden lassen. Die besondere Schwierigkeit in der automatisierten Betrachtung extrem früher Druckerzeugnisse besteht in der Vielzahl von Eigenheiten und Eigenschaften des Quellenmaterials, die in den folgenden Kapiteln erläutert werden.

---

<sup>14</sup> Vgl. Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997; S. 276: „Glyph“ als „images of individual symbols“.

<sup>15</sup> Der ASCII-Code (American Standard Code for Information Interchange) ist eine Zeichenkodierung, die 128 verschiedene Zeichen definiert.

### 3. Der Bereich extrem früher Drucke: Inkunabeln

Extrem frühe Drucke par excellence sind „Inkunabeln“. Der Begriff „Inkunabel“ stammt aus dem lateinischen („*incunabula*“) und bezeichnet „früheste typographische Erzeugnisse“<sup>16</sup>, die vor dem 1. Januar 1501 fertiggestellt wurden. Zwar stellt sich mit Geldner<sup>17</sup> die Frage, ob die Datierung auf das Jahr 1501 gerechtfertigt und der Vielfalt des beginnenden europäischen Buchdruckes angemessen ist, eines jedoch ist allen Inkunabeln gemein: Inkunabeln folgen sehr stark ihrem handschriftlichen Vorbild, den mittelalterlichen Codizes. Spätere Druckerzeugnisse des beginnenden 16. Jahrhunderts emanzipieren sich von ihrem handschriftlichen Vorbild und präsentieren



**Abbildung 1:** Bertholdus: *Horologium devotionis*. Zeitglöcklein des Lebens und Leidens Christi, Basel: [Johann Amerbach], 1492.

sich in einem völlig anderen Druckbild als Inkunabeln. Abbildung 1 illustriert die Ausführungen mit der 1492 gedruckten Inkunabel „Zeitglöcklein des Lebens und Leidens Christi“ (Bertholdus).

Wo Gutenberg und die ältesten Drucker „sehr erfolgreich bemüht [waren], die Handschrift bis in alle Einzelheiten nachzuahmen“<sup>18</sup>, findet sich eine Vielfalt an unterschiedlichen Schriftarten, wie die Textura, die Rotunda, die zwischen Antiqua und Fraktur stehende Bastardschrift, die Gotico-Antiqua und die

<sup>16</sup> Geldner, Ferdinand: *Inkunabelkunde*. Eine Einführung in die Welt des frühesten Buchdrucks. Wiesbaden: Reichert, 1978. (Elemente des Buch- und Bibliothekswesens; Band 5); S. 1.

<sup>17</sup> Vgl. Ebenda, S. 2: „Die Jahrhundertwende 1500 / 1501 erweist sich zwar für die Inkunabelforschung als ein äußerst praktisches Hilfsmittel, sie wird aber der Sache selbst nur sehr annähernd gerecht, denn erstens gibt es kein absolutes Kriterium, das uns ermöglichte, zwischen ‚Inkunabel‘ und ‚Nichtmehrinkunabel‘ eine sichere Grenze zu ziehen und zweitens verläuft die Grenze, die wir mit Hilfe einer Summe von Kriterien ziehen können, in den einzelnen Ländern und Städten zeitlich verschieden.“

<sup>18</sup> Ebenda, S. 2.

Humanistenschrift Antiqua<sup>19</sup>. An die Seite der vielfältigen Schriftarten gesellen sich feine Einzelheiten wie die Nachahmung und Umsetzung von Ligaturen (Verschmelzung zweier Drucktypen zu einer Einheit), dem Abkürzen ganzer Silben oder Wörter durch Abbrüviaturen, der Gestaltung besonderer Kürzungszeichen, gar der mechanischen Wiedergabe von Illustrationen und Buchschmuck durch Holz- oder Metallschnitt oder durch Kupferstich.

Inhaltlich beginnt mit den frühen Druckerzeugnissen eine langsame aber stetige Bewegung der Emanzipation – ein Grund hierfür lässt sich in den stark gesunkenen Produktionskosten für das Medium Buch ausmachen. Gehörten Inkunabeln vorwiegend der „religiösen Sphäre“<sup>20</sup> an, so beginnen weltliche Texte zunehmend mit religiös-kirchlichen Büchern zu konkurrieren<sup>21</sup> und als Katalysator für eine Breitenwirkung des Buches zu fungieren: Der Buchdruck steht „nun nicht mehr nur im Dienste der ‘Litterati’, die die lateinische Sprache beherrschten, er wandte sich an alle, die lesen (wenn auch nicht schreiben) konnten und auch an die, die Bücherlesungen zuhörten“<sup>22</sup>

Die gesamte Vielfalt der „Löbliche[n] Kunst, die der ewige Gott in seiner unergründlichen Weisheit erweckt hat“<sup>23</sup> offenbart sich in der „verteilten digitalen Inkunabelbibliothek“, einem Projekt der Kölner Universitäts- und Stadtbibliothek und der Herzog August Bibliothek Wolfenbüttel.

### 3.1. Die verteilte digitale Inkunabelbibliothek

Die „verteilte digitale Inkunabelbibliothek“ (vdlb) ist ein von der Deutschen Forschungsgemeinschaft (DFG) gefördertes Kooperationsprojekt zwischen der Kölner Universitäts- und Stadtbibliothek und der Herzog August Bibliothek in Wolfenbüttel (HAB). Angestrebt ist mit dem Projekt eine „möglichst repräsentative Digitalisierung der jeweiligen Inkunabelbestände unter

---

<sup>19</sup> Vgl. Mazal, Otto: Paläographie und Paläotypie. In: Hellinga, Lotte; Härtel, Helmar (Hrsg.): Buch und Text im 15. Jahrhundert: Arbeitsgespräch in d. Herzog-August-Bibliothek Wolfenbüttel vom 1.-3. März 1978. Hamburg: Hauswedell, 1981; S. 65.

<sup>20</sup> Geldner, Ferdinand: Inkunabelkunde. Eine Einführung in die Welt des frühesten Buchdrucks. Wiesbaden: Reichert, 1978. (Elemente des Buch- und Bibliothekswesens; Band 5); S. 4.

<sup>21</sup> Ebenda.

<sup>22</sup> Ebenda.

<sup>23</sup> Kölner Chronik, zitiert nach Ebenda, S. 5.

Berücksichtigung konservatorischer Gutachten“<sup>24</sup>. Über den Förderzeitraum von drei Jahren wurde aus den Gesamtbeständen der Bibliotheken jeweils rund 500 Inkunabeln digitalisiert: aus der Kölner Universitäts- und Stadtbibliothek ältere Inkunabeln (zwischen 1460 und 1484) und aus der HAB spätere Drucke der Veröffentlichungsjahre zwischen 1485 und 1500. Die im Rahmen des Kooperationsprojektes digitalisierten Inkunabeln machen einen Bestandteil von 10-12% der weltweit erhaltenen Wiegendrucke aus<sup>25</sup>.

Ergänzt um Metadaten, transkribierte Textpassagen und weiterführende Informationen, macht die „verteilte digitale Inkunabelbibliothek“ ihren umfangreichen Fundus von mehr als 300.000 Digitalisaten unter der Internetadresse <http://www.inkunabeln.ub.uni-koeln.de> frei verfügbar. Aufgrund der leichten Zugänglichkeit und der hohen Qualität der Digitalisate dient das Bildmaterial der vdIb als materielle Grundlage der Frage, wie sich automatische Texterkennung auf den Bereich extrem früher Drucke erweitern lässt.

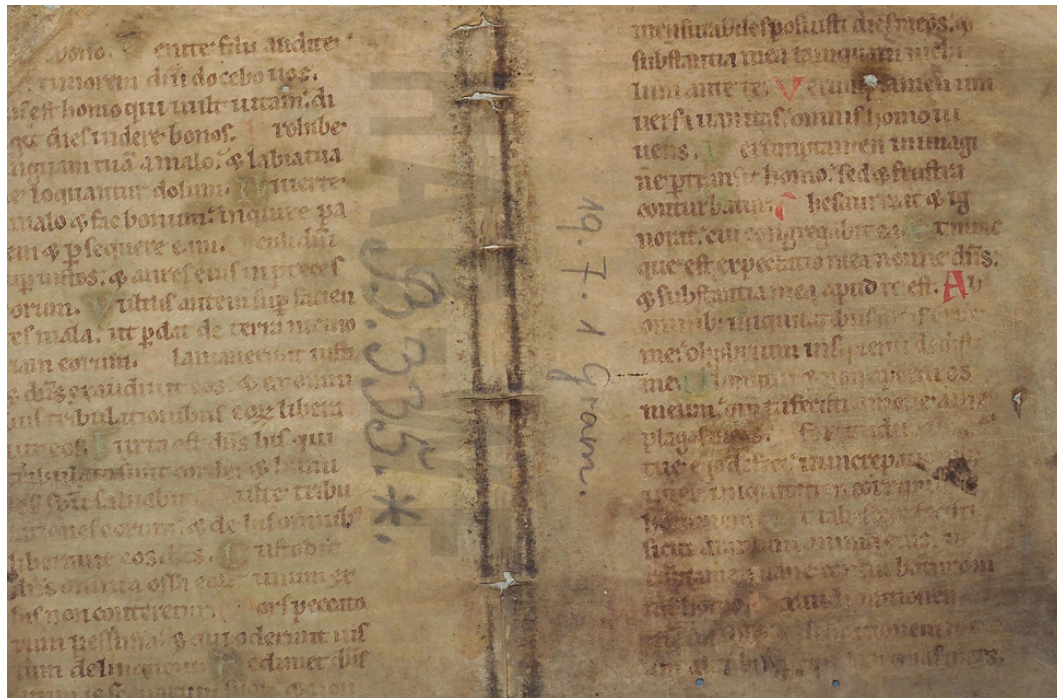
### 3.2. Inkunabeln, Herausforderung für OCR

Durch ihre spezifischen Eigenschaften stellen Inkunabeln eine große Herausforderung für automatische Texterkennung dar. Neben typographischen Eigenheiten erschweren besonders physikalische Schäden an dem Quellenmaterial die Volltexterkennung, machen sie oftmals gar unmöglich. So verwahren sich Glyphen, die durch Beschädigung (beispielsweise durch Abrieb und Abnutzung oder durch partielle Brandschäden) des Trägermaterials zerstört wurden, der automatisierten Glyphenerkennung und bleiben semantische Leerstellen im Gesamttext, die sich nur schwerlich mit Bedeutung füllen lassen. Abbildung 2 zeigt einen Ausschnitt einer Inkunabel, gedruckt gegen Ende des 15. Jahrhunderts, deren Erschließung zerstörter Textstellen selbst das geschulte menschliche Auge nicht mehr möglich ist.

---

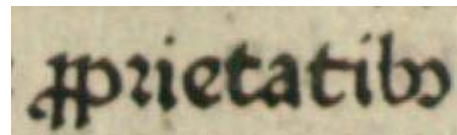
<sup>24</sup> Steyer, Timo: Verteilte Digitale Inkunabelbibliothek – Ein Baustein zur Gesamtdigitalisierung aller Inkunabelausgaben. In: Historisches Forum. Themenhefte von Clio-online Bd. 10/I (2007), S. 295. <[http://www.clio-online.de/site/lang\\_\\_de/40208224/Default.aspx](http://www.clio-online.de/site/lang__de/40208224/Default.aspx)> (27.08.2008)

<sup>25</sup> Vgl. die Dokumentation des Projektes unter <<http://inkunabeln.ub.uni-koeln.de/vdibDevelop/dokumentation/dokumentation.html>> (27.08.2008).



**Abbildung 2:** Beispiel eines stark beschädigten Druckes (Löcher, Druck verblasst, Verschmutzung, etc.): Alexander de Villa Dei: Doctrinale (Pars I) (Comm: Johannes Synthen), Deventer: Jacobus de Breda, [gedruckt zwischen 31 Oktober 1489 und 20 Dezember 1491].

Nicht minder problematisch stellen sich semantische Feinheiten wie Abbriviaturen dar (Abbildung 3): Wo sich Sprache und Schrift über Jahrhunderte entwickelte und Schriftproduzenten eigene Abbriviaturen einsetzten, ist es häufig nicht mehr möglich, den Sinn der Abkürzungen von Silben oder ganzer Sinnabschnitte zu rekonstruieren.



**Abbildung 3:** mehrere Abbriviaturen in einem Wort.

Ligaturen kennzeichnen einen weiteren Problemkomplex, mit dem sich automatische Text- und Glyphenerkennung auf Basis extrem alter Drucke zu beschäftigen hat: Die Bedeutung von Ligaturen, also mehreren Glyphen, die unter einer Drucktype zusammengefasst wurden, soll erschlossen werden – ein Vorhaben, das sich allein schon aufgrund der Vielzahl unterschiedlicher Drucktypen als schwierig erweist. Abbildung 4 zeigt die Verschmelzung der Glyphen „s“ und „t“ zu einer Glyphe.



**Abbildung 4:** Beispiel einer Ligatur. Hier sind s und t verschmolzen zu einer Drucktype.

## 4. „TED“: Ein Beitrag zur OCR von Inkunabeln

Um sich der „Herausforderung Inkunabel“ mit ihren zuvor angedeuteten spezifischen Eigenheiten zu nähern, fokussiert der in dieser Arbeit vorgestellte Ansatz auf die folgenden Prämissen, wie sich Texterschließungsarbeit gestalten soll:

- 1.) Texterschließung soll automatisiert verfahren: Die manuelle Arbeit auf dem Digitalisat, sprich: die Zuordnung von ASCII Code zu Glyphen, soll sich auf ein Minimum reduzieren.
- 2.) Texterschließung soll Zeit- und arbeitersparend sein, somit die Kosten der Volltextgenerierung minimieren.
- 3.) Texterschließung soll die spezifischen Eigenschaften des Quellenmaterials berücksichtigen und herausarbeiten: Generierung von Metadaten, etc.

Im Rahmen der vorliegenden Arbeit wurde ein selbstlernendes System entworfen und umgesetzt, das die Arbeit auf dem Digitalisat auf ein Arbeitsminimum zu reduzieren sucht. Mit „TED“ – „TED Enhances Digitization“ verfolgt die entwickelte Anwendung einen praxisnahen, plattformunabhängigen<sup>26</sup> Ansatz, um Glyphen zu trainieren und digitalisatübergreifende Automatisierung zu ermöglichen.

### 4.1. „TED“: Intention und Workflow

Die in der vorliegenden Ausarbeitung entwickelte und vorgestellte Applikation „TED“ will eine praktikable Arbeitshilfe bieten, um die Volltextgenerierung auf Basis extrem alter Drucke zu erleichtern, zu beschleunigen und den extrahierten Text für die Darstellung, im vorliegenden Falle als XHTML Dokument, vorzubereiten. Zentrale Arbeitsprozesse sollen gebündelt werden und ein signaturweit lernendes System den Arbeitsfluss der Text- und Metadatengenerierung unterstützen. Die Arbeit von „TED“ ist gekennzeichnet durch eine Vielzahl unterschiedlicher Algorithmen und Verfahren, die im Laufe dieser Ausarbeitung auf ihre Tauglichkeit zur automatischen Vorbereitung der digitalisierten Inkunabeln erprobt werden.

---

<sup>26</sup> Auf Basis der Qt-Bibliothek lässt sich die Anwendung sowohl unter Linux, als auch Windows kompilieren und nutzen.

## 5. „TED“: Algorithmen und Arbeitsweise

Den Beginn der Prozesskette gründet die Vorverarbeitung mit ihrem Fokus auf die Besonderheiten des Quellenmaterials. Das vorliegende Kapitel sucht die Frage zu beantworten, inwiefern sich die Digitalisate extrem früher Drucke vorbereiten lassen auf Texterkennungsprozesse – und wie sich jene Prozesse zur automatisierten Gewinnung von (Text)Information möglichst flexibel gestalten lassen, um der „Herausforderung Inkunabel“ gerecht zu werden.

### 5.1. Vorverarbeitung

Analog den Zahnrädern eines Uhrwerks greifen in der vorbereitenden Bearbeitung des Quellenmaterials zahlreiche Präparationsschritte ineinander. Die anfängliche Analyse des Digitalisates isoliert Buchschmuck und andersfarbig – im Gegensatz zu den zumeist schwarzen Lettern – gekennzeichnete Glyphen und Objekte. Daran anschließend wird das Digitalisat zugeschnitten: Schwarze Ränder werden entfernt, so dass im Mittelpunkt der Arbeit einzig der Text steht. Auf Basis des zugeschnittenen Digitalisates schließt sich das Verfahren an, den Informationsgehalt des Bildmaterials auf tatsächlich bedeutende kleinste Einheiten zu reduzieren: Am Schluss der Kette ist das Digitalisat auf bitonale Farbinformation reduziert, die jedwede Information abbildet, die sich auch im Farbbild findet.

Die folgenden Ausführungen stellen zentrale Schritte, Methoden und Algorithmen vor, die der Vorverarbeitung essentiell sind und in der Applikation „TED“ für eine Optimierung der sich an die Vorverarbeitung anschließenden Verfahren der Objektwahrnehmung und der Objekterkennung Sorge tragen. Erläutert wird zunächst ein Verfahren zur farbbasierten Layoutanalyse, daran anschließend wird die Verbesserung des Ausgangsmaterials über den Histogramm-, bzw. Kontrastausgleich und die Umwandlung des Farbbildes in Graustufen, schließlich in ein bitonales Bild über Schwellwertalgorithmen besprochen. Abschließend werden Verfahren zur Beseitigung von Störungen und zur Filterung von irrelevanter Information im Bild vorgestellt.

### 5.1.1. Farb- und Layoutanalyse

Die Forschungsliteratur der Bildbearbeitung und -verarbeitung<sup>27</sup> verzeichnet eine reiche Vielfalt an Algorithmen und Methoden, um bereits im ersten Stadium der Vorverarbeitung, der Layoutanalyse und Betrachtung der spezifischen Charakteristika des Digitalisates, relevante Information zu extrahieren und der weiteren Verarbeitung zugänglich zu machen. „Relevante Information“ bezeichnet Bildinformation, die den Text des Digitalisates repräsentiert. Der Text des Digitalisates ist gewebt aus sprachlichen Zeichen – den gedruckten Buchstaben – und nicht-sprachlichen Zeichen, wie Illustrationen oder Buchschmuck in ihren vielfältigen Manifestationen. Irrelevante Bildinformation ist Information, die für die Erscheinung des Textes nicht von Nutzen, gar hinderlich ist; als Beispiel für irrelevante Information sei auf die schwarzen Seitenränder verwiesen, die aus dem Scan-, oder Photographievorgang resultieren.

Hochwertige Vorverarbeitung trägt Sorge dafür, dass nicht-sprachliche Information und durch Einfärbung besonders gekennzeichnete sprachliche Information – als Beispiel sei an dieser Stelle auf eingefärbte Buchstaben und Glyphen verwiesen – gesondert betrachtet, kontextualisiert und weiteren Verarbeitungsprozessen isoliert zugeführt wird. Die Analyse der Text- und Bildgestaltung sieht sich durch eine Vielzahl von Algorithmen und Methoden unterstützt, auf die an dieser Stelle, des begrenzten Umfanges der Ausarbeitung wegen, nur stichpunktartig hingewiesen werden kann. So widmet sich die Arbeit von Bulacu et al. der Layoutanalyse handschriftlicher Aufzeichnungen des „Kabinet der Koningin“<sup>28</sup>, arbeitet die von Hirano et al. publizierte Arbeit mit der Metasprache PDL zur Beschreibung digitalisiert vorliegender Information<sup>29</sup>, oder gibt die von Shafait et al. vorgestellte Arbeit einen Überblick über zahlreiche

---

<sup>27</sup> Als erste Anlaufstelle für das derzeitige Mögliche der Bildanalyse und -verarbeitung mag die im zweijährigen Turnus stattfindende Konferenz „International Conference on Document Analysis and Recognition“ (ICDAR) dienen.

<sup>28</sup> Bulacu, Marius; van Koert, Rutger; Schomaker, Lambert; van der Zant, Tijn: Layout Analysis of Handwritten Historical Documents for Searching the Archive of the Cabinet of the Dutch Queen. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol. 1 (S. 357-361).

<sup>29</sup> Takashi Hirano, Yuichi Okano, Yasuhiro Okada, Fumio Yoda: Text and Layout Information Extraction from Document Files of Various Formats Based on the Analysis of Page Description Language. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol. 1 (S. 262-266).



Algorithmen zur Erkennung und Kennzeichnung zusammenhängender Sinnabschnitte in Digitalisaten<sup>30</sup>. Die Leistungsfähigkeit und Anknüpfungspunkte zur Verbesserung von Methoden zur Segmentierung von Digitalisaten macht der 2007 im Rahmen der ICDAR ausgetragene Wettbewerb zur automatischen Erkennung von Text- und Bildabschnitten in Alltagspublikationen wie Magazin- oder Zeitungsseiten offenbar<sup>31</sup>.

Was die vorliegende Arbeit leistet, ist die basale Extraktion farblich auffällig gekennzeichnete Schriftcharakteristika – in Abgrenzung zu den schwarzen Drucklettern – wie Rot- oder Blaudruck oder eingefärbter graphischer Elemente (Buchschnuck). Die im Rahmen der Arbeit entwickelte Applikation fokussiert hierbei auf rot und blau eingefärbte Zeichen. Grund für die Beschränkung auf die Isolierung von Rot- und Blaudruck ist der endliche Zeitrahmen, der dem Projekt gegeben ist und die an anderer Stelle zu klärende Frage, wie reich verzierte Einleitungsglyphen adäquat analysiert und als einzelne Glyphen erkannt werden können.

Aktiviert wird die gesonderte Behandlung von eingefärbten Objekten und Buchstaben über die Funktion *void isolatedecoration(QImage \*image, QImage \*ornamentimage)* in der Quelldatei *./TED/src/basics/basics.cpp*. Der Algorithmus betrachtet die drei Farbkanäle Rot, Grün und Blau des RGB-Bildes für jedes Pixel des übergebenen Arbeitsbildes *image*: Besitzt das jeweilige Pixel einen hohen Rotwert und gleichzeitig niedrige Blau- und Grünwerte, so wird das Pixel als Rotdruck wahrgenommen und im temporären Bild *ornamentimage* rot eingefärbt. Analog betrachtet und verarbeitet der Algorithmus blaugedruckte Objekte: Besitzt das betrachtete Pixel einen hohen Blauwert und niedrige Rot- und Grünwerte, so deutet das Pixel auf ein blau eingefärbtes Objekt hin. Die als Rot- und Blaudruck erkannten Pixel werden zunächst aus dem Arbeitsbild *image*

---

<sup>30</sup> Shafait, Faisal; Keysers, Daniel; Breuel, Thomas M.: Performance Evaluation and Benchmarking of Six Page Segmentation Algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence. Juni 2008 (Vol. 30, No. 6) S. 941-954.

<sup>31</sup> Antonacopoulos, A.; Gatos, B.; Bridson, D.: ICDAR2007 Page Segmentation Competition. 9th International Conference on Document Analysis and Recognition (ICDAR'07). Curitiba, Brazil: September 2007, S. 1279-1283.  
<[http://www.iit.demokritos.gr/~bgat/Icdar2007\\_PageSegmentationCompetition.pdf](http://www.iit.demokritos.gr/~bgat/Icdar2007_PageSegmentationCompetition.pdf)> (27.08.2008)

entfernt, im Bild *ornamentimage* zwischengespeichert, um an späterer Stelle der Verarbeitung gesondert betrachtet werden zu können.

Aus der schlichten Arbeitsweise des Algorithmus resultieren mitunter alleinstehende Pixel, die keinem farbigen Objekt zugeordnet werden können. Um solche Pixel zu beseitigen, schließt sich an den Prozess der Isolierung eingefärbter Objekte die Erodierung<sup>32</sup>. Hierbei wird für jedes Pixel des Bildes *ornamentimage* überprüft, ob sich im unmittelbaren Umfeld Nachbapixel befinden. Pixel, in deren Nachbarschaft sich weniger als drei Pixel befinden, werden als unwichtige Information interpretiert und gelöscht, d.h. mit dem Farbwert „weiß“ belegt. Das Resultat der Operation ist ein um Störungen bereinigtes Bild.

### 5.1.2. Umwandlung Farbbild zu Graustufenbild

Auf die Isolierung des Buchschmucks und andersfarbiger Objekte erfolgt die Umwandlung des Farbbildes in ein Graustufenbild. Die Reduktion der Farbinformation basiert auf der Annahme, dass das vorliegende Bildmaterial mehr Information bereitstellt, als für die folgenden Prozesse nötig; folglich soll das Digitalisat von Ballast befreit werden, um die weiteren Verarbeitungs- und Erkennungsprozesse zu optimieren.

Die Digitalisate der „verteilten digitalen Inkunabelbibliothek“ liegen als JPG Dateien in verschiedenen Auflösungen vor. Allen Digitalisaten gemein ist die Farbtiefe: Jede Inkunabel wird durch eine Bilddatei im RGB-Format mit 24-Bit repräsentiert. Jeweils acht Bit stehen dem RGB-Modell für jeden Farbkanal zur Verfügung; der darstellbare Farbraum erstreckt sich somit über  $2^8$ , folglich 256 Farbnuancen pro Farbkanal.

Die praktische Grundlage für die Arbeit auf den Digitalisaten bildet die bildverarbeitende Klasse *QImage* der Qt-Bibliothek. *QImage* repräsentiert jedes Pixel durch die Mischung der drei Farbkanäle Rot, Grün, Blau und stellt einen Alphakanal bereit, der die Transparenz des Pixels speichert. Die Mischung der Farbkanäle verfährt additiv: je höher der Anteil der Grundfarbe, desto heller das Mischergebnis. So ergibt die Mischung der drei RGB-Kanäle mit maximalen Farbwerten Weiß, die gemischten RGB-Kanäle mit minimalen Farbwerten

---

<sup>32</sup> Der ausführlichen Darstellung der Erodierung ist Kapitel 5.3.1 gewidmet.

Schwarz. Über die insgesamt 24 Bit der drei Farbkanäle lassen sich somit  $256^3=16.777.216$  (16,7 Millionen) Farben darstellen.

16,7 Millionen Farben – das ist weitaus mehr, als den weiteren Verarbeitungs- und Erkennungsprozesse dienlich ist. Für die automatische Erkennung von Objekten, gar von Text, sind Millionen von Farben schlichtweg redundante Information. Die Vorverarbeitung der Digitalisate fokussiert darauf, den Überfluss an Bildinformation zu beseitigen und auf möglichst charakteristische und der Weiterverarbeitung aussagekräftige Elemente zu reduzieren. Die Umwandlung von Mehrkanalbildern (RGB) in Graustufenbilder gründet den Anfang einer Prozesskette, um das Gros an Information auf Wesentliches zu reduzieren.

Für die Umwandlung von Farb- in Grauwertinformation lassen sich verschiedene Farbmodelle zu Rate ziehen. Auf den folgenden Seiten wird das IHS Farbmodell vorgestellt, das mit seiner Komponente der Intensität eine solide Ausgangsbasis für weitere Verarbeitungsschritte bietet.

### 5.1.3. Graustufenumwandlung: das IHS Farbmodell

We can summarize by saying that RGB is ideal for image color generation (as in image capture by a color camera or image display in a monitor screen), but its use for color description is much more limited.<sup>33</sup>

Das IHS Farbsystem – bei Jähne<sup>34</sup> und Gonzalez / Woods<sup>35</sup> als HSI Farbsystem bezeichnet – bildet die Farbmischung der drei Farbkanäle des RGB Modells ab über die Leuchtstärke (**I**ntensity), den Farbton (**H**ue) und die Farbreinheit, bzw. Sättigung (**S**aturation). Mit seiner Komponente der Helligkeit (Intensity) beschreibt das IHS Farbsystem explizit die erste Stufe der vorzunehmenden Reduktion der Bildinformation: die Umwandlung von Mehrfarbenbildern in Graustufenbilder.

Die Transformation des RGB Systems nach IHS erfolgt über die folgenden drei Annahmen:

---

<sup>33</sup> Gonzalez, Rafael C.; Woods, Richard E.: Digital Image Processing – Third Edition. New Jersey: Pearson Education, 2008; S. 407.

<sup>34</sup> Jähne, Bernd: Digitale Bildverarbeitung. Berlin, Heidelberg: Springer-Verlag 2005; S. 177.

<sup>35</sup> Gonzalez, Rafael C.; Woods, Richard E.: Digital Image Processing – Third Edition. New Jersey: Pearson Education, 2008; S. 407 ff.

- 1) Die Intensität ist die Maßeinheit der Helligkeit, resultierend aus dem Durchschnitt der Farbwerte:

$$I = \frac{R + G + B}{3}$$

- 2) Sättigung beschreibt die Farbreinheit, ausgedrückt durch den Term

$$S = 1 - \frac{\min(R, G, B)}{I}$$

- 3) Der Farbton ist proportional zu der durchschnittlichen Wellenlänge der Farbe und basiert auf der Abbildung der Farben in einem Polarkoordinatensystem (vgl. O’Gorman<sup>36</sup>):

$$H = \cos^{-1}\left(\frac{(R - G) + (R - B)}{2} \cdot ((R - G)^2 + (R - B)(G - B))^{\frac{1}{2}}\right)$$

Einzig die IHS-Komponente der Intensität ist relevant für den weiteren Verlauf der Vorverarbeitung des Digitalisates. Für jedes Pixel der Bilddatei wird die Intensität über den Mittelwert der RGB Farbwerte bestimmt und anschließend den drei Kanälen des RGB-Bildes zugewiesen. Aus der Umwandlung resultiert das Graustufenbild. Um weitere Optionen der Kennzeichnung einzelner Pixel direkt auf dem Bildmaterial zu ermöglichen, wird das resultierende Graustufenbild intern nicht als ein solches, sondern als ein RGB-Bild gespeichert.

Für die Rücktransformation des IHS in das RGB Farbsystem sei an dieser Stelle auf die detaillierten Ausführungen von Gonzalez / Woods<sup>37</sup> verwiesen.

#### 5.1.4. Bild zuschneiden

Der Scan- oder Photographievorgang bildet das Quellenmaterial nicht nur ab, sondern bildet es um, ergänzt es gar um überflüssige Information. Neben Störungen wie Rauschen oder unscharfer Wiedergabe des Materials findet sich das Digitalisat oftmals bereichert um die (zumeist schwarze) Unterlage des Scantisches, den Vermerk auf die Herkunft des Materials oder einer Farbtafel zum

<sup>36</sup> O’Gorman, Lawrence; Sammon, Michael J.; Seul, Michael: Practical Algorithms for Image Analysis – Description, Examples, Programs, and Projects. Cambridge: Cambridge University Press, 2008; S. 55.

<sup>37</sup> Gonzalez, Rafael C.; Woods, Richard E.: Digital Image Processing – Third Edition. New Jersey: Pearson Education, 2008; S. 411f.

späteren An- und Abgleich von Farbprofilen. Für die weitere Verarbeitung und die Objekterkennung jedoch ist solche Information weniger relevant, gar hinderlich. So arbeitet eine Vielzahl der im späteren Verlauf der Ausarbeitung vorgestellten Schwellwertalgorithmen bedeutend besser, wenn ihnen ein um überflüssige Information bereinigtes Digitalisat vorgelegt wird.

Um solche überflüssige Information zu reduzieren, schneidet die vorgestellte Applikation das Digitalisat vor der grundlegenden Verarbeitung zurecht. Der entwickelte und im Folgenden vorgestellte Algorithmus macht sich die spezifischen Eigenschaften des Bildmaterials zunutze: Da der Großteil der Digitalisate über breite schwarze Ränder verfügt, sucht der Algorithmus nach der ersten, jeweils längsten, horizontalen und vertikalen schwarzen Linie jeweils unterhalb und oberhalb, sowie links und rechts von der Mitte des Digitalisates. Um die schwarzen Ränder möglichst ausgeprägt wiederzugeben, wird zuvor das Quellenmaterial über einen schnell und einfach arbeitenden Schwellwertalgorithmus auf Schwarz- und Weißwerte reduziert – für die detaillierte Erläuterung des verwendeten Schwellwertalgorithmus sei an dieser Stelle auf Kapitel 5.2.1. verwiesen. Ausgehend von den ermittelten längsten schwarzen Linien, sowohl horizontal als auch vertikal, sucht der Algorithmus anschließend, jeweils von der Hälfte der Höhe und der Breite des Digitalisates ausgehend, nach dem ersten Weißpunkt, der – im Idealfall – auf den Zentralteil des Digitalisates hinweist. Aus dieser Arbeitsweise resultieren vier Koordinatenpunkte, über die das Digitalisat zugeschnitten wird.<sup>38</sup>

#### 5.1.5. Automatischer Histogrammausgleich

Das Histogramm eines Bildes dokumentiert über eine Liste von 256 Elementen (8-Bit Graustufenbild) die Anzahl der Pixel des Bildes, die mit dem entsprechenden Graustufenwert belegt sind. Zur Verdeutlichung mag das folgende Beispiel dienen: Ist ein Monochrombild mit einer Auflösung von 10x10 Pixeln komplett schwarz eingefärbt, so verzeichnet das Histogramm keinen Zähler für die Farbwerte 1 (nahezu schwarz) bis 255 (weiß). Index 0 (schwarz) des

---

<sup>38</sup> Für den C++ Quellcode des automatischen Zuschneidens vgl. die Funktion *QImage cutrelevant(QImage \*image, QImage \*copyimage)* in *./TED/src/basics/basics.cpp*.

Histogramms dokumentiert 100 Werte, resultierend aus der Bildauflösung von  $10 \times 10 = 100$  Pixel.

Bildmaterial, das den möglichen Grauwertraum nicht vollständig ausschöpft, erscheint arm an Kontrast und von einem „Grauschleier“ überzogen. Die vorliegende Ausarbeitung verwendet den bei O’Gorman et. al.<sup>39</sup> beschriebenen Algorithmus „Histogram Transformation Specified“ zur Streckung der Grauwerte und Kontrasterhöhung. Das über den Histogrammausgleich verbesserte Quellenmaterial präsentiert sich nicht nur dem Auge kontrastreicher, sondern hat auch positive Folgen für die späteren Arbeitsschritte der Glyphenisolierung. Abbildung 5 zeigt das Digitalisat vor und nach der automatischen Kontrastverbesserung mit entsprechenden Histogrammen.

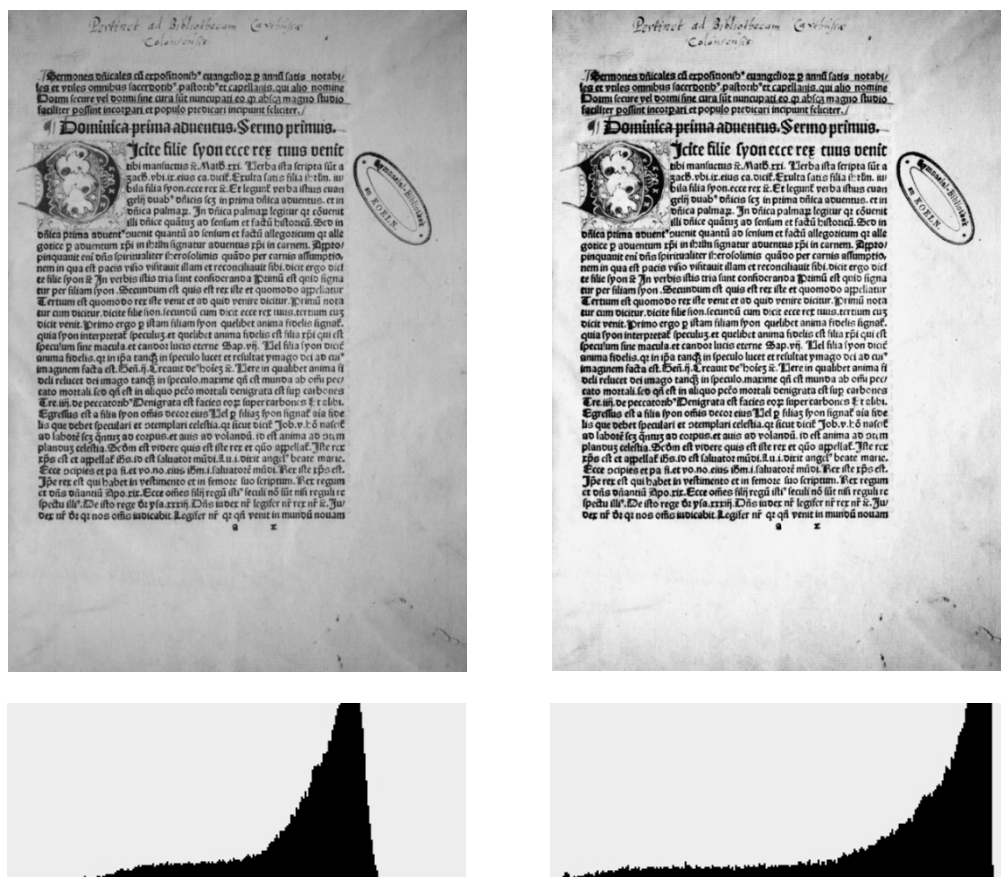


Abbildung 5: Johannes de Verdena: Sermones "Dormi secure" de tempore (ISTC Nr.: ij00454000). Digitalisat vor (links) und nach automatischem Histogrammausgleich (rechts) und zugehörige Histogramme.

<sup>39</sup> O’Gorman, Lawrence; Sammon, Michael J.; Seul, Michael: Practical Algorithms for Image Analysis – Description, Examples, Programs, and Projects. Cambridge: Cambridge University Press, 2008; S. 34.

### 5.1.6. Medianfilter

Das Medianfilter ist der Gruppe der Rangordnungsfilter zuzuordnen. Rangordnungsfilter betrachten die Nachbarschaftspixel jedes Pixels, speichern die gefundenen Graustufenwerte und sortieren die so gewonnenen Graustufenwerte in aufsteigender Reihenfolge. Aus einer 3x3 Pixelmatrix resultiert eine Liste, bestehend aus insgesamt neun Werten. Relevant für das Medianfilter ist das Element, das sich in der Mitte der sortierten Liste der Pixelwerte befindet. Das Medianfilter selektiert den Pixelwert des fünften Listenelements und weist dem Pixel in der Mitte der Matrix den Median, d.h. den fünften Wert der Liste zu. Der Flaschenhals, der die Performanz des Verfahrens einengt, ist hierbei die Sortierung der Liste: Für jedes Pixel müssen die Nachbarschaftspixel und das Pixel selbst gesammelt und in Rangordnung gebracht werden; bei einer Bildauflösung von 1000x1000 Pixel sind folglich 9.000.000 Werte zu sortieren. Die vorliegende Implementierung verwendet den Quicksort Algorithmus<sup>40</sup>, um die Sortierung der Werte zeitnah umzusetzen.

Seinen Einsatz findet das Medianfilter, um Bildstörungen zu verringern, gleichzeitig aber die Kanten von Objekten nur marginal zu beeinflussen. Befindet sich ein schwarzes Pixel in der Nachbarschaft von hellen Pixeln, so darf dieses Pixel mit großer Wahrscheinlichkeit als ein überflüssiges Pixel kategorisiert werden. Analog verfährt das Medianfilter mit hellen Pixeln in dunkler Umgebung: Findet sich ein weißes Pixel in unmittelbarer Nähe von dunklen Grauwerten, so wird dieses Pixel mit dem entsprechenden dunklen Medianwert eingefärbt.

Das skizzierte Medianfilter bringt bei leichten Bildstörungen praktikable Ergebnisse und empfiehlt sich durch seine kantenbewahrende Arbeitsweise für die Vorverarbeitung der Digitalisate alter Drucke als Standardeinstellung. Starkem Rauschen jedoch ist das vorgestellte Filter nicht gefeit. Eine Möglichkeit, stark verrauschtes Quellenmaterial der Weiterverarbeitung zugänglich zu machen, besteht darin, die Arbeitsmatrix dynamisch zu vergrößern, so dass das Mittelpunktpixel einem umfangreicheren Arbeitskontext zugeführt werden kann. Diese Arbeitsweise charakterisiert das „Adaptive Median Filter“.

---

<sup>40</sup> Verwendet wurde die Implementation des QuickSort Algorithmus von <http://www.dreamincode.net/forums/showtopic31409.htm> (27.08.2008)

Der Algorithmus zur adaptiven Medianfilterung arbeitet in zwei Arbeitsschritten. Zunächst erstellt der Algorithmus für ein  $k \times k$  großes Fenster die zuvor besprochene Rangliste der Pixelgrauwerte und extrahiert den Medianwert. In Phase A, die sich an die Erstellung der Rangliste anschließt, prüft der Algorithmus, ob der Medianwert ein Impuls<sup>41</sup> ist, d.h. den Wert 0 (schwarz) oder 255 (weiß) repräsentiert. Ist der Medianwert kein Impuls, so vergrößert sich das Abtastfenster mit jeder Iteration von Phase A bis zu einer zuvor festgelegten maximalen Fenstergröße.

Ist der Medianwert jedoch ein Impuls, so tritt der Algorithmus in Phase B: Hierbei vergleicht der Algorithmus den zuvor als Impuls charakterisierten Medianwert mit dem Wert des Pixels im Mittelpunkt des Arbeitsfensters. Ist der Grauwert des Mittelpunktpixels größer als der kleinste Grauwert im gesamten Arbeitsfenster und zugleich kleiner als der maximale Grauwert im Arbeitsfenster, so erfährt das Mittelpunktpixel keine Änderung; andernfalls wird dem Mittelpunktpixel der Medianwert zugewiesen.

Abschließend mag die folgende Kurzbeschreibung des Algorithmus in Pseudocode die Arbeitsweise des adaptiven Medianfilters veranschaulichen, basierend auf den Ausführungen von Gonzalez / Woods<sup>42</sup>:

### Termini

$Z_{\min}$  = Minimaler Grauwert in Fenster  $S_{xy}$   
 $Z_{\max}$  = Maximaler Grauwert in Fenster  $S_{xy}$   
 $Z_{\text{med}}$  = Medianwert in Fenster  $S_{xy}$   
 $Z_{xy}$  = Grauwert von zentralem Pixel an Position  $(x, y)$

### Phase A:

$A1 = Z_{\text{med}} - Z_{\min}$   
 $A2 = Z_{\text{med}} - Z_{\max}$   
 If  $(A1 > 0 \text{ AND } A2 < 0)$  : Starte Phase B  
 Else : Erhöhe Fenstergröße um Pixel  $(x, y)$   
 If Fenstergröße  $\leq S_{\max}$  : Wiederhole Phase A  
 Else : Gebe  $Z_{\text{med}}$  zurück

### Phase B:

$B1 = Z_{xy} - Z_{\min}$

<sup>41</sup> Vgl. Gonzalez, Rafael C.; Woods, Richard E.: Digital Image Processing – Third Edition. New Jersey: Pearson Education, 2008; S. 332 ff.

<sup>42</sup> Ebenda.



```
B2 =  $Z_{xy} - Z_{\max}$   
If ( $B1 > 0$  AND  $B2 < 0$ ) : Gebe  $Z_{xy}$  zurück  
Else : Gebe  $Z_{\text{med}}$  zurück
```

## 5.2. Segmentierung: Schwellwertverfahren

Schwellwertverfahren dürfen als das ausschlaggebende Moment gelten, das die Qualität weiterer Verfahren, wie der Isolierung von Objekten oder der automatischen Texterkennung, beeinflusst, gar determiniert. Auch als „Thresholding“, „Binarization“ oder „Segmentierungsmethoden“ bezeichnet, ordnen Schwellwertalgorithmen jedes Pixel eines Bildes einer eindeutigen Pixelsemantik zu. Die zentrale Frage: Repräsentiert ein Pixel bloße Hintergrundinformation, ist somit irrelevant für weitere Schritte in der Nachbearbeitung und Texterkennung des vorliegenden Digitalisates, oder birgt das Pixel die Information, die für weitere Algorithmen essentiell ist?

Schwellwertalgorithmen arbeiten – im vorliegenden Falle – auf Graustufenbildern und reduzieren das Bildmaterial auf ein zweifarbiges Bild<sup>43</sup>. Repräsentiert ein Pixel im Graustufenbild 256 verschiedene Graustufen, so reduzieren Schwellwertverfahren die Pixelinformation auf zwei Werte: relevante Information, repräsentiert über den Wert 0 (schwarz) oder Hintergrundinformation (255, weiß). Der Schwellwert entscheidet hierbei darüber, wann ein Pixel schwarz, oder weiß gesetzt wird.

Ob ein Pixel einem der automatischen Texterkennung relevanten Objekt zugehörig, oder minder relevanter Hintergrundinformation zuzuordnen ist, stellt Algorithmen zur grundlegenden Vorverarbeitung von Bildinformation vor zahlreiche Probleme: Wie lässt sich ohne manuellen Eingriff, feststellen, ob ein Pixel für die Erkennung von semantischen Strukturen wie Text oder Formatierungen (Layout, etc.) relevant ist? Wann ist ein Bildpunkt, das kleinste, bedeutungskonstituierende Partikel der digitalen Bildinformation, zugehörig zu vernachlässigender Hintergrundinformation, wann zugehörig zu dem Text des Digitalisates?

Im Idealfall enthält das über Schwellwertmethoden vereinfachte Bildmaterial ausschließlich relevante Information und sieht sich befreit von überflüssiger Hintergrundinformation. In der Praxis bleibt der Idealfall oftmals jedoch bloße Idee. Zahlreiche Störungen, die aus dem Scan- oder Photographievorgang

---

<sup>43</sup> Auch als Monochrombild oder bitonales Bild bezeichnet.

resultieren, wie Bildrauschen, heterogene Beleuchtung und Objekte mit stark differierenden Grauwerten, machen die Berechnung eines optimalen Schwellwertes zu einem komplexen Unterfangen.

Die folgenden Kapitel stellen signifikante Algorithmen und Ansätze zur Schwellwertbestimmung und der Reduktion auf ein bitonales Bild vor. Der Fokus der Analyse ist gerichtet auf die Untersuchung der Praxistauglichkeit der jeweiligen Algorithmen, im Hinblick auf die Vorbereitung extrem früher Druckzeugnisse für die Automatische Texterkennung (OCR). Als wesentliches Maß für die Leistungsfähigkeit eines jeweiligen Algorithmus gilt die Geschwindigkeit der Verarbeitung des vorliegenden Graustufenbildes, sowie die konkreten, subjektiv beurteilten Resultate der Segmentierung.

Die im Folgenden vorgestellte Auswahl von drei Algorithmen zehrt von der Arbeit Parkers, die trotz ihres Alters von nunmehr elf Jahren einen zeitgemäßen Überblick über Methoden und Algorithmen der Schwellwertverfahren bietet – und zudem durch ihre Zugänglichkeit der vorliegenden Ausarbeitung dienlich ist. Als Basismaterial der Segmentierung dienen die in Graustufenbilder konvertierten und automatisch zugeschnittenen Digitalisate der verteilten digitalen Inkunabelbibliothek.

Für eine umfassende und differenzierte Analyse der beachtlichen kreativen Vielfalt an Schwellwertalgorithmen sei an dieser Stelle auf die Arbeit von Sezgin und Sankur<sup>44</sup> verwiesen, die 40 Algorithmen vorstellt und auf Performanz und Tauglichkeit überprüft.

### 5.2.1. Segmentierung über globalen Schwellwert

Die einfachste und grundlegendste Methode, die Bildinformation eines 8-Bit Graustufenbildes auf ein Monochrombild zu reduzieren, ist die Wahl eines global gültigen Schwellwertes. Ungeachtet der besonderen Gegebenheiten des vorliegenden Bildes vergleichen Methoden, die über den globalen Schwellwert arbeiten, jedes einzelne Pixel mit dem global, d.h. für das jeweilige Digitalisat gesetzten Schwellwert. Besitzt das Pixel einen größeren oder gleichen

---

<sup>44</sup> Sezgin, Mehmet; Sankur, Bülent: Survey over image thresholding techniques and quantitative performance evaluation. In: Journal of Electronic Imaging Vol. 13, 146 (2004); S. 146-165.

Graustufenwert als der Schwellwert, so ist es als Hintergrundinformation zu betrachten und erhält den Wert 255 (weiß). Niedrigere Graustufenwerte als der Schwellwert resultieren in der Belegung des Pixels mit dem Wert 0 (schwarz).

Die zwei unterschiedlichen Pixelklassen (schwarzes Pixel, weißes Pixel) werden beschrieben durch folgende zwei Ausdrücke:

$$\text{a) } I(i, j) < T$$

$$\text{b) } I(i, j) \geq T$$

Ist der Graustufenwert des betrachteten Pixels an Position  $i, j$  des Bildes  $I$  kleiner als der gesetzte Schwellwert  $T$ , so wird es schwarz eingefärbt (a). Pixel mit einem Graustufenwert größer oder gleich dem Schwellwert  $T$  werden auf weiß gesetzt (b). Die Implementation des globalen Schwellwertalgorithmus ist simpel: Jedes Pixel des Bildes wird verglichen mit dem zuvor gesetzten Schwellwert aus dem Wertebereich zwischen 0 und 255 und nach entsprechender Prüfung neu belegt.

Schnell jedoch stößt der globale Schwellwertalgorithmus an seine Grenzen. Besonders verrauschte Bildinformation oder unregelmäßige Beleuchtung bzw. heterogene Helligkeitsverläufe machen deutlich, dass der globale Schwellwertalgorithmus durch seine Einfachheit zwar performant, aber nicht von Relevanz für anspruchsvolle Segmentierung ist, da er den Besonderheiten des Digitalisates nicht gerecht wird. Als Alternative zum global arbeitenden Schwellwertalgorithmus bietet sich die Segmentierung des Bildmaterials über histogrammbasierte Schwellwertalgorithmen an. Diese werden den spezifischen Eigenschaften des Bildmaterials gerecht – und generieren hochwertige Ergebnisse.

### 5.2.2. Histogrammbasierte Segmentierung: Otsu

Ein Anwendungsfall der histogrammbasierten Segmentierung ist das 1978 von Otsu vorgestellte Verfahren zur Segmentierung von Graustufenbildern<sup>45</sup>. Otsu geht davon aus, dass das Bildmaterial durch zwei Gruppen von Pixeln repräsentiert wird: Pixel mit einem Grauwert unterhalb (0 bis Schwellwert  $T$ ),

---

<sup>45</sup> Vgl. Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997; S. 119 f.

oder einem Grauwert oberhalb des Schwellwertes ( $T+1$  bis 255). Ausgehend von den zwei Klassen berechnet Otsu die Varianz der Grauwerte des Digitalisates, ausgedrückt durch  $\sigma_t^2$ . Die Varianz bezeichnet die Streuung der Grauwerte im vorliegenden Bildmaterial. Für jeden Schwellwert  $T$ , der die Graustufenwerte in die zwei Klassen Vorder- und Hintergrundpixel teilt, lässt sich die Varianz der beiden Pixelklassen bestimmen ( $\sigma_w^2$ ). Otsu zufolge ist der (möglichst) optimale Schwellwert zu finden durch Minimierung der Varianz innerhalb der Klassen und Maximierung der Varianz zwischen den Klassen ( $\sigma_b^2$ ):

$$n(t) = \frac{\sigma_b^2}{\sigma_t^2}$$

Die globale Varianz  $\sigma_t^2$  lässt sich aus dem Bild bestimmen über den Mittelwert  $\mu_T$ ; die Varianz zwischen den Klassen berechnet sich Parker<sup>46</sup> zufolge über

$$\sigma_b^2 = \omega_0 \omega_1 (\mu_0 - \mu_1)^2$$

Wobei für  $\omega_0$  und  $\omega_1$  gilt:

$$\omega_0 = \sum_{i=0}^t p_i \quad ; \quad \omega_1 = 1 - \omega_0$$

Für  $\mu_0$ ,  $\mu_1$  und  $\mu_t$  gilt:

$$\mu_0 = \frac{\mu_t}{\omega_0} \quad ; \quad \mu_1 = \frac{\mu_T - \mu_t}{1 - \omega_0} \quad ; \quad \mu_t = \sum_{i=0}^t i \cdot p_i$$

Bei einem histogrammbasierten Segmentierungsansatz wie dem von Otsu vorgestellten, wird deutlich, wie wichtig das vorherige Zuschneiden des Digitalisates ist: Wären die schwarzen Scanränder in das Histogramm aufgenommen worden, so würden die Ergebnisse der Varianzberechnungen auf einem Histogramm arbeiten, das einen sehr hohen Anteil seiner Werte im dunklen Graustufenbereich hat und somit gänzlich andere, mitunter schlechtere Ergebnisse liefert, als das automatisch zugeschnittene Digitalisat.

---

<sup>46</sup> Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997; S. 120.

### 5.2.3. Iterative Auswahl: Ridler, Calvard, Thrussel

Das von Ridler und Calvard vorgestellte Verfahren zur Segmentierung von Bildinformation<sup>47</sup> arbeitet iterativ: Auf die Wahl eines Anfangsschwellwertes – der Mittelwert der möglichen Graustufen – folgt die Berechnung des durchschnittlichen Grauwertes der durch den Schwellwert gesonderten zwei Pixelklassen. Der Mittelwert der Graustufen unterhalb des Schwellwertes wird als  $T_b$  bezeichnet, der Mittelwert der Graustufen oberhalb des Schwellwertes als  $T_o$ . Der neue Schwellwert berechnet sich aus dem Durchschnitt von  $T_b$  und  $T_o$ . Dieses Verfahren wird mit dem gewonnenen Schwellwert so lange wiederholt, bis keine Änderung des Schwellwertes festgestellt werden kann.

Der von Parker<sup>48</sup> implementierte Algorithmus ergänzt das Verfahren von Ridler und Calvard um den Vorschlag von Trussell<sup>49</sup>, das Histogramm bei den Berechnungen zu berücksichtigen:

$$T_k = \frac{\sum_{i=0}^{T_k} i \cdot h[i]}{2 \sum_{i=0}^{T_k-1} h[i]} + \frac{\sum_{j=T_{k-1}+1}^N j \cdot h[j]}{2 \sum_{j=T_{k-1}+1}^N h[j]}$$

Hierbei ist  $h$  das Graustufenhistogramm des Bildes. Der optimale Schwellwert ist Parker zufolge gefunden, wenn sich der Wert  $T_k$  nicht mehr ändert,  $T_k$  folglich gleich  $T_{k+1}$  ist.

### 5.2.4. Evaluation und Wahl des bevorzugten Algorithmus

Um die Digitalisate des frühen Buchdruckes vorzubereiten auf die folgenden Wahrnehmungs- und Erkennungsprozesse, wurde eine Vielzahl der bei Parker beschriebenen Algorithmen implementiert und auf ihre Praxistauglichkeit untersucht. Abbildungsmatrix 2 und die Vergleichstabelle 1 illustrieren, dass die Methode von Otsu zu bevorzugen ist: Zwar bewegt sich die Performanz des Algorithmus im flinken Mittelfeld der getesteten Algorithmen, die Ergebnisse der Umwandlung jedoch überzeugen. Um einen möglichst heterogenen Testsatz an Digitalisaten zu gewährleisten, wurden drei Bilddateien ausgewählt, die sich

<sup>47</sup> Ridler, T.W.; S. Calvard: Picture Thresholding Using an Iterative Selection Method. 1978. Zitiert nach Ebenda.

<sup>48</sup> Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997. S. 119

<sup>49</sup> Trussell, H. J.: Comments on „Picture Thresholding Using an Iterative Selection Method“. Systems, Man and Cybernetics, IEEE Transactions on. 1979. Zitiert nach Ebenda.

durch aufwändige farbige Illustration<sup>50</sup>, einer Vielzahl von (teils zerstörten) Glyphen<sup>51</sup> und einer geringen Anzahl gleichmäßig verteilter Objekte<sup>52</sup> auszeichnen. Die Spalte „Performanz“ gibt den Zeitaufwand für die Verarbeitung der drei Digitalisate in Sekunden wieder, die Spalte „Qualität“ die subjektive Beurteilung der Umwandlungsergebnisse.

---

<sup>50</sup> Methodius, S ADB Zedler: *Revelationes divinae a sanctis angelis factae*. Add: Wolfgang Aytinger: *Tractatus super Methodium*. 1498 (Ed: Sebastian Brant) (ISTC Nr.: im00524000); Auflösung: 2000x3383 Pixel. Im Folgenden zitiert als „Methodius“.

<sup>51</sup> Albertus Magnus ADB Zedler Wiki: *Compendium theologiae veritatis*. Add: Bernoldus de Caesarea: *Distinctiones de tempore et de sanctis quarum declarationes ex compendio ... capiuntur* (ISTC Nr.: ia00234000); Auflösung: 3072x3840 Pixel. Im Folgenden zitiert als „Magnus I“.

<sup>52</sup> (Albertus Magnus ADB Zedler Wiki: *Secreta mulierum et virorum (cum commento novo)* 1490 (ISTC Nr.: ia00318000); Auflösung: 2000x3350 Pixel. Im Folgenden zitiert als „Magnus II“.







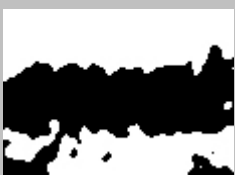







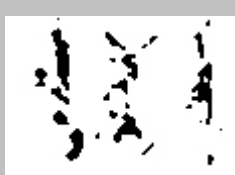



**Tabelle 1:** Schwellwertverfahren in der Praxis I – Analyse hinsichtlich Performanz und Qualität des Verfahrens. Die Wertungsskala reicht von sehr schlechten (--), hin zu sehr guten Ergebnissen (++). Die in runde Klammern gesetzten Performanzwerte bezeichnen die Laufzeit des jeweiligen Algorithmus in Sekunden.




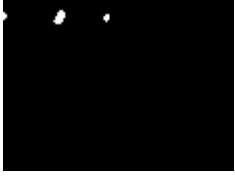






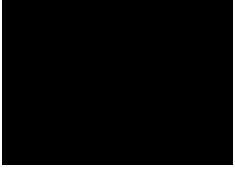













Algorithmus	Performanz <sup>53</sup>	Qualität
<b>Globaler Schwellwert</b>	++ (0,25)	- (stark Abhängig von Quellenmaterial)
<b>P-Tile Methode</b>	+ (0,45)	--
<b>Eckpixelbasiert (Laplacian)</b>	-- (2,51)	-
<b>Iterativ (Ridler)</b>	-- (2,10)	++
<b>Otsu</b>	+ (0,42)	++
<b>Entropiebasiert: Pun</b>	+ (0,37)	--
<b>Entropiebasiert: Kapur</b>	+ (0,47)	-
<b>Entropiebasiert: Johannsen</b>	+ (0,49)	+
<b>Fuzzy: Entropy</b>	+ (0,49)	-
<b>Fuzzy: Yager</b>	+ (0,48)	+
<b>Minimum Error</b>	+ (0,49)	-
<b>Relaxation</b>	-- (153,23)	--

<sup>53</sup> Das Testsystem: Intel Core2Duo 8400@3GHz; 4GB RAM; Windows Vista 32-Bit. „TED“, kompiliert mit der Debug-Bibliothek von Qt 4.4.1.



**Tabelle 2:** Schwellwertverfahren in der Praxis II – Visualisierung der Ergebnisse der Schwellwertverfahren.

Algorithmus	Magnus I	Methodius	Magnus II
Glob. Schwellwert ( $T=128$ )			
Glob. Schwellwert ( $T=155$ )			
Glob. Schwellwert ( $T=192$ )			
P-Tile Methode			
Eckpixelbasiert (Laplacian)			
Iterativ (Ridler)			

Otsu			
Entropie: Pun			
Entropie: Kapur			
Entropie: Johannsen			
Fuzzy: Entropy			
Fuzzy: Yager			
Minimum Error			
Relaxation			

### 5.3. Vorverarbeitung II

Der abschließende Teil der Vorverarbeitung beseitigt Störungen im Bildmaterial und optimiert das Quellenmaterial über die Verdrehungskorrektur („Skew Correction“), um anschließende Isolierungs- und Erkennungsprozesse zu optimieren.

#### 5.3.1. Erodierung

Über Schwellwertverfahren umgewandeltes Bildmaterial enthält häufig einzelne verstreute schwarze oder weiße Pixel, die als Salz-und-Pfeffer-Rauschen<sup>54</sup> bezeichnet werden. Diese überflüssigen Pixel zu beseitigen nimmt sich – in begrenztem Umfang und mit eingeschränkter Wirksamkeit – der Arbeitsschritt der Erodierung an.

Das einfachste Verfahren, jene überflüssigen Pixel „auszuwaschen“, erfolgt über eine 3x3 Pixelmatrix, die jedes Pixel des Digitalisates in den Mittelpunkt stellt und die unmittelbare Nachbarschaft von acht Pixeln betrachtet. Steht das Zentrumspixel isoliert im Raum, d.h. sind die Nachbarpixel ausschließlich andersfarbig als das Zentrumspixel, so wird das Zentrumspixel mit dem Wert der Umgebungspixel belegt. An seine Grenzen stößt das Verfahren bei großen Löchern in (schwarzen) Objekten. Hierbei ist mittels 3x3 Matrix nicht festzustellen, ob sich die jeweiligen Pixel eingebettet finden in einen größeren Kontext. Dieses Problem zu lösen sucht das von O’Gorman vorgestellte „kFill“ Filter.

#### 5.3.2. kFill Filter

Das  $k$  im Namen deutet auf die Arbeitsweise des  $k$ Fill Filters hin: Ein  $k \times k$  großes Arbeitsfenster tastet jedes Pixel des vorliegenden Bildmaterials ab und reagiert auf die Gegebenheiten in der unmittelbaren und näheren Umgebung des Pixels. Charakterisiert ist das Arbeitsfenster durch seinen Kern und die Nachbarschaft des Kerns: Der Kern des Fensters besteht aus einer  $(k - 2) \cdot (k - 2)$  großen Region;

---

<sup>54</sup> Auch bezeichnet als: „salt-and-pepper-noise“, „speckle noise“, „shot noise“ oder „dirt“: O’Gorman, Lawrence; Sammon, Michael J.; Seul, Michael: Practical Algorithms for Image Analysis – Description, Examples, Programs, and Projects. Cambridge: Cambridge University Press, 2008; S. 139.

die äußeren zwei Reihen und Spalten des Fensters  $4(k - 1)$  bezeichnen die Nachbarschaft des Kerns.

Bestrebt ist das  $k$ Fill Filter, den Kern in Abhängigkeit der Nachbarschaft zu füllen. Ob der Kern schwarz oder weiß gefüllt wird, ist abhängig von den drei Variablen  $n$ ,  $c$  und  $r$ . Für die Füllung des Kerns mit schwarzen Pixeln bezeichnet  $n$  die Anzahl der schwarzen Pixel in der Nachbarschaft,  $c$  gibt die Menge der verbundenen schwarzen Pixel in der Nachbarschaft wieder und  $r$  birgt die Summe der schwarzen Eckpixel des  $k \times k$  großen Arbeitsfensters. Analog verfährt das Verfahren für die Füllung des Kerns mit weißen Pixeln.

Der Kern wird gefüllt, wenn folgende Bedingung zutrifft:

$$c = 1 \wedge [(n > 3k - 4) \vee (n = 3k - 4) \wedge r = 2]$$

Die Arbeitsweise des  $k$ Fill Filters verfährt iterativ. Jeder Iterationsschritt ist gekennzeichnet durch zwei Operationen: das Füllen des Kerns zum einen mit weißen, zum anderen mit schwarzen Pixeln. Erfolgt kein Füllen in zwei aufeinander folgenden Operationen, so beendet das  $k$ Fill Filter seine Arbeit.

O’Gorman betont die Effizienz des  $k$ Fill Filters zur Reduzierung von Bildstörungen und irrelevanten Pixeln – in seiner Ausarbeitung berichtet O’Gorman von der Reduzierung des Ausgangs-



**Abbildung 6:** Ergebnis des  $k$ Fill Filters mit einem Arbeitsfenster von  $k=3$ . Links: Originalbild; rechts: nach Anwendung des  $k$ Filters.

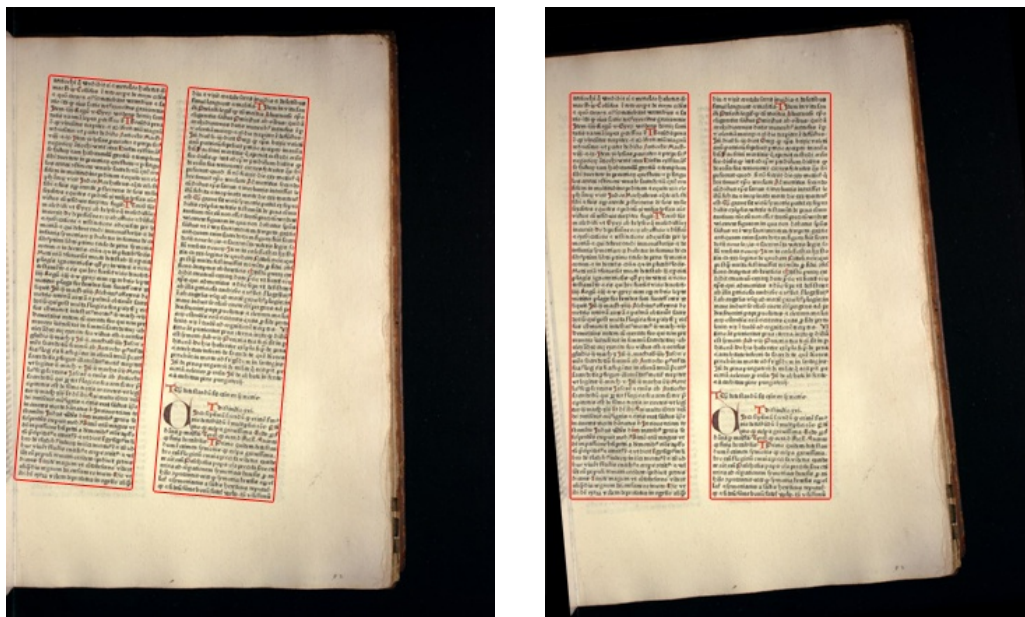
materials um 5% bis 20%<sup>55</sup>. Für die Umsetzung des  $k$ Fill Filters in „TED“ wurde auf eine externe Implementation des Filters zurückgegriffen<sup>56</sup>; durch seine bildverbessernden Eigenschaften (vgl. Abbildung 6) empfiehlt sich das  $k$ Fill Filter als Standardeinstellung der vorliegenden Applikation.

<sup>55</sup> Vgl. O’Gorman, Lawrence; Sammon, Michael J.; Seul, Michael: Practical Algorithms for Image Analysis – Description, Examples, Programs, and Projects. Cambridge: Cambridge University Press, 2008; S. 141.

<sup>56</sup> Für die Implementation des  $k$ Fill Filters in „TED“ wurde zurückgegriffen auf die Arbeit von Wakahara, Toru: <http://cis.k.hosei.ac.jp/~wakahara/kfill.c>.

## 5.4. Skew-Korrektur

Oftmals bildet der Scan- oder Photographievorgang das Quellenmaterial verdreht ab – nur selten ist es möglich, das abzubildende Dokument akribisch genau auszurichten. Ergänzt wird die vorbereitende Verarbeitung des Quellenmaterials häufig durch die automatische Verdrehungskorrektur, auch als „Deskewing“ oder „Skew-Correction“ bezeichnet. Seine Anwendung findet die Verdrehungskorrektur bei rotiertem, verdrehtem Quellenmaterial, um die Erkennungsrate von Text- und Objekterkennung zu verbessern, oftmals gar erst zu ermöglichen. So stellen verdrehte Objekte, Textabschnitte, Glyphen und Buchstaben mitunter ein großes Problem dar für die automatische Texterkennung – Ist der um  $45^\circ$  im Uhrzeigersinn gedrehte Buchstabe „p“ noch immer als ein „p“ zu erkennen, oder stößt algorithmenbasierte Texterkennung bereits hier an ihre Grenzen? Abbildung 7 illustriert die (manuelle) Verdrehungskorrektur anhand des Beispiels einer verdreht abgebildeten Inkunabel mit manuell eingezeichneten umschließenden Rechtecken (rot).



**Abbildung 7:** Originaldigitalisat<sup>57</sup> (links) mit manuell lokalisierten Bounding Boxes (rot gekennzeichnete Bereiche) und manuell um  $5^\circ$  gegen den Uhrzeigersinn gedrehtes Quellenmaterial mit umschließenden Rechtecken (rechts).

<sup>57</sup> Vincentius Bellovacensis: Speculum morale (1477).

Mit bis zu fünf Grad Verdrehung<sup>58</sup> weisen die betrachteten Digitalisate der vdlb eine Rotation des Quellenmaterials auf, die der implementierten OCR nicht hinderlich ist. In der vorliegenden Ausarbeitung wird somit vom Einsatz der Verdrehungskorrektur abgesehen. Seine Erläuterung und Besprechung soll das „Deskewing“ jedoch an dieser Stelle erfahren.

Die zentrale Frage der Verdrehungskorrektur ist die Frage, um welchen Winkel das Quellenmaterial im Gegensatz zum Original rotiert ist. Bei Kenntnis dieses Rotationswinkels ist es möglich, das Bild entsprechend zu rotieren und eine getreue Wiedergabe des Quellenmaterials zu rekonstruieren. Für die Bestimmung des Rotationswinkels bieten sich nach Parker<sup>59</sup> zwei Verfahren an: Zum einen das Verfahren von Baird<sup>60</sup>, das auf Glyphenebene versucht, Aussagen über den Rotationswinkel zu treffen, zum anderen das Verfahren der Hough-Transformation. Im Folgenden wird zunächst das Verfahren Bairds erläutert; daran anschließend folgt die Erläuterung der Hough-Transformation.

#### 5.4.1. Glyphen und Grundlinien: Das Verfahren von Baird

Das Verfahren von Baird versucht, zusammenhängende Bereiche zu erkennen: Ein zusammenhängender Bereich besteht aus unterschiedlichen Elementen einer bestimmten Größe, manifestiert in Glyphen ohne Unterlängen (Glyphen wie g, j, p, q). Für solche Ansammlungen von Glyphen lässt sich eine Grundlinie identifizieren, die Textzeile. Lässt sich eine Glyphe von einem rechteckigen Kasten umschließen (die sog. „Bounding Box“), so dürfte, laut Parker, die untere Gerade des Rechtecks auf die Ausrichtung der Glyphe und folglich auf die Ausrichtung der gesamten Textzeile schließen lassen. Für die Bestimmung des Verdrehungswinkels verfährt Baird in vier Schritten:

1. Generiere eine „Bounding Box“, die die Glyphe einschließt.

---

<sup>58</sup> Für die Analyse des Quellenmaterials konnte nur ein kleiner Teil, rund 15.000 Digitalisate, des umfangreichen Gesamtfundus von mehr als 300.000 Bilddateien der vdlb flüchtig überblickt und auf verdrehtes Material geprüft werden.

<sup>59</sup> Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997; S. 294-300.

<sup>60</sup> Baird, H. S.: The Skew Angle of Printed Documents. In: Proceedings of the Conference of the Society of Photographic Scientists and Engineers. SPIE, Bellingham, WA (1987); S. 14-21. Zitiert nach Ebenda, S. 295.

2. Finde für jede Glyphe den Mittelpunkt der unteren Geraden der Bounding Box.
3. Für jeden möglichen Rotationswinkel  $\theta$ : Berechne aus dem um den Rotationswinkel  $\theta$  gedrehten Bildmaterial die Y-Projektion der durch Schritt 2 gefundenen Punkte.  $P_i(q)$  bezeichnet den Wert von Bounding Box Nummer  $i$  für den Rotationswinkel  $\theta$ .
4. Maximiere die Funktion:  $A(\theta) = \sum_{i=1}^n P_{i(\theta)}^2$ . Der Rotationswinkel  $\theta$ , der den größten Wert der Funktion generiert, ist der korrekte Rotationswinkel.

Der Flaschenhals an dem Verfahren Bairds besteht in der intensiven Suche nach dem Rotationswinkel, da für eine Vielzahl von Winkelwerten die Funktion  $A(\theta)$  berechnet werden muss. Die Genauigkeit des Verfahrens von Baird liegt laut Parker im Bereich eines halben Grades.

#### 5.4.2. Hough-Transformation

Die Hough-Transformation bezeichnet eine Methode, um geometrische Figuren – im vorliegenden Falle Geraden – in einem Monochrombild zu erkennen. Basierend auf der Annahme, dass ein Pixel dadurch charakterisiert ist, dass eine unendliche Menge von Geraden durch das Pixel gehen können, bildet die Hough-Transformation diese möglichen Geraden in einem als „Hough-Raum“ bezeichneten Modellraum ab. Jeder Punkt im Hough-Raum entspricht einer Geraden; die Intensität eines Punktes im Hough-Raum gibt darüber Aufschluss, wie viele Geraden den Punkt im Datenraum des vorliegenden Monochrombildes durchdringen.

Die Transformation der Bildpunkte in den Hough-Raum findet über folgende Annahmen statt:

Jeder Punkt einer Geraden muss die Bedingung  $Y = mX + b$  erfüllen<sup>61</sup>. Hierbei bezeichnet  $m$  die Steigung und  $b$  den Achsenabschnitt der Geraden; die Koordinaten des zu untersuchenden Pixels sind durch  $X$  und  $Y$  ausgedrückt. Werden  $X$  und  $Y$  als Konstanten interpretiert, so lässt sich der Ausdruck in den Hough-Raum mit den Koordinaten  $m$  und  $b$  transferieren:  $b = -Xm + Y$ . Ein

<sup>61</sup> Vgl. hierzu auch Jähne, Bernd: Digitale Bildverarbeitung. Berlin, Heidelberg: Springer-Verlag 2005; S. 481.

Bildpunkt im zweidimensionalen Raum, lokalisiert durch die Koordinaten  $X$  und  $Y$ , entspricht einer Geraden im Hough-Raum mit den Koordinaten  $m$  und  $b$ .

Da eine Gerade eine unendliche Steigung besitzen<sup>62</sup> und somit nicht über die Formel  $Y = mX + b$  in dem diskreten Modellraum abgebildet werden kann, wird die Gerade durch ihren Steigungswinkel (der Winkel zur  $X$ -Achse des Bildkoordinatensystems)  $\omega$  und den Abstand der Geraden vom Ursprung des Koordinatensystems ( $r$ ) charakterisiert:

$$r = x \cos(\omega) + y \sin(\omega)$$

Jähne weist auf den hohen Aufwand der Hough-Transformation hin:

„Der Nachteil der Houghtransformation für die Liniendetektion ist der hohe Rechenaufwand. Für jeden Bildpunkt ist eine Gerade im Parameterraum zu berechnen, und jeder Punkt im Modellraum, durch den die Gerade läuft, muss inkrementiert werden.“<sup>63</sup>

---

<sup>62</sup> Vgl. Ebenda, S. 482.

<sup>63</sup> Ebenda, S. 483.



## 5.5. Objektwahrnehmung

Mit dem vorbereiteten Quellenmaterial ist der Wahrnehmung und Vorbereitung von Glyphen eine solide Arbeitsgrundlage gegeben: Jedes schwarze Pixel, das sich im Digitalisat findet, ist einem Objekt (einer Illustration, einem Bild) oder einer Glyphe zugehörig. Das Ziel der Objektwahrnehmung besteht in der Isolierung solcher Objekte, um sie für den Prozess der Glyphen- bzw. Texterkennung verfügbar zu machen.

Der folgende Abschnitt erläutert, wie die entwickelte Applikation „TED“ die Objektwahrnehmung umsetzt und wahrgenommene Objekte vorbereitet für anschließende Erkennungsprozesse. Zu Beginn der Erläuterung steht die vorbereitende Erkennung von (Objekt)Kanten im Digitalisat, über die der Pfadfindalgorithmus Objekte ausfindigt macht.

### 5.5.1. Kantenerkennung und Kantenisolierung

Der erste Arbeitsschritt der Objektwahrnehmung besteht darin, das bitonal vorliegende Quellenmaterial auf die Kantenpixel der Objekte zu reduzieren. Parker<sup>64</sup> folgend, ist die Ecke eines Objekts definiert durch eine Änderung der Graustufenwerte – im bitonal vorliegenden Bildmaterial durch die Änderung des Wertes von 0 (schwarz) auf 255 (weiß). Ein Eckpixel lässt sich folglich feststellen durch die Betrachtung der Änderung des Gradienten (Farb- bzw. Grauwertverlaufs). Das Verfahren zur Berechnung des Gradienten tastet jedes Pixel im Bild ab und setzt das aktuell verarbeitete Pixel an Position (x,y) in Relation zu seinem direkten linken Nachbapixel, ausgedrückt durch den Term

$$\nabla_x A(x, y) = A(x, y) - A(x - 1, y)$$

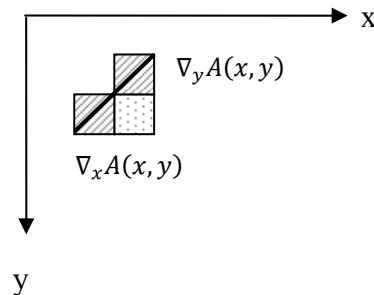
Im zweiten Schritt wird das unmittelbar angrenzende Pixel oberhalb des verarbeiteten Pixels betrachtet:

$$\nabla_y A(x, y) = A(x, y) - A(x, y - 1)$$

---

<sup>64</sup> Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997; S.10-16.

Die Stärke des Gradienten ist die Länge der Hypotenuse des rechtwinkligen Dreiecks mit den Seiten  $\nabla_x$  und  $\nabla_y$  (vgl. Abbildung 8)<sup>65</sup>.



**Abbildung 8:** Bestimmung der Gradientenstärke des verarbeiteten Pixels (gepunktet) aus den Nachbarpixeln (gestreift).

Die Stärke der Ecke berechnet sich über den Term

$$G_{mag} = \sqrt{\left(\frac{\partial A}{\partial x}\right)^2 + \left(\frac{\partial A}{\partial y}\right)^2}$$

Ob ein schwarzes Pixel ein Eckpixel ist, bestimmt der Schwellwert  $T$ , der sich aus dem Maximalwert  $z_{max}$  und Minimalwert  $z_{min}$  der gewonnenen Werte der Eckstärken berechnet:

$$T = \frac{z_{max} - z_{min}}{2}$$

Auf Basis der vorangegangenen Ausführungen über die Berechnung der Gradientenstärke lassen sich Kantenpixel leicht und performant identifizieren.

Das so auf Objektkanten reduzierte Digitalisat (siehe Abbildung 9) bildet die Arbeitsgrundlage des Algorithmus zur Pfadfindung. Für die Implementation des Eckpixelalgorithmus sei an dieser Stelle auf den C++ Quellcode in Anhang I.1 verwiesen.



**Abbildung 9:** Glyphe (links) und auf Kantenpixel reduzierte Glyphe (rechts).

<sup>65</sup> Ebenda, S. 11.

### 5.5.2. Kantenverfolgung: Pfadfinder

Objekte anhand ihrer Kontur zu isolieren, das ist die Aufgabe des Pfadfindalgorithmus. Pfadfindalgorithmen finden ihren Einsatz häufig in Computerspielen, wo eine virtuelle Spielfigur sich durch eine virtuelle Landschaft arbeitet, Hindernissen ausweicht und einem vorgegebenen Weg folgt<sup>66</sup>. Auf die Bildverarbeitung übertragen, besteht die Aufgabe des Pfadfindalgorithmus in der vorliegenden Ausarbeitung darin, die Eckpixel von Glyphen und Objekten im Bildmaterial zu verfolgen und von einem Startpunkt ausgehend das Objekt zu umgehen, bis der Pfadfinder am Ausgangspunkt der Bewegung angelangt ist.

Ausgehend von der linken oberen Ecke des Digitalisates bewegt sich der Algorithmus zeilenweise von links nach rechts durch das vorbereitete Bildmaterial und sucht nach schwarz eingefärbten Pixeln. Findet der Algorithmus ein solches schwarzes Pixel, so beginnt er, nach weiteren schwarzen Pixeln in der unmittelbaren Umgebung des gefundenen schwarzen Pixels zu suchen. Hierbei stellen weiße Pixel undurchdringbare Hindernisse für den Pfadfinder dar; die Aufgabe des Pfadfindalgorithmus stellt einzig die Verfolgung des durch schwarze Pixel ausgedrückten Weges dar.

Die Navigation auf der Objektkante erfolgt über ein gewichtetes Koordinatensystem, das die Bestrebung des Algorithmus repräsentiert, die Kante im Uhrzeigersinn zu verfolgen. Die möglichen Bewegungsrichtungen schließen neben den geraden Bewegungen (Nord, Ost, Süd, West) auch diagonale Navigation (Nordost, Südost, Südwest, Nordwest) auf der Objektkante ein. Die Kantenverfolgung agiert hierbei stark regelbasiert: In der Implementation wird über eine Vielzahl an *if*-Bedingungen die vorhergehende Bewegungsrichtung abgefragt, die nähere Umgebung des Pfadfinders betrachtet und in Abhängigkeit von diesen Umgebungsvariablen die nächste Bewegungsentscheidung getroffen. Elementare Regeln wie „gehe nicht nach Osten, wenn der vorherige Schritt eine Bewegung in Richtung Westen war“, finden sich ergänzt um tiefergehende

---

<sup>66</sup> Vgl. das entsprechende Kapitel über Methoden und Algorithmen zur Pfadfindung in Computerspielen bei: Bourg, David M.; Seeman, Glenn: AI for Game Developers – Creating Intelligent Behavior in Games. Sebastopol: O'Reilly Media, Inc., 2004.

kontextsensitive Regeln: „Besteht sowohl die Möglichkeit, nach Norden, als auch nach Nordosten zu gehen, dann gehe zuerst nach Norden.“

Basierend auf dem Satz an Regeln bewegt sich der Pfadfindalgorithmus im Uhrzeigersinn um das Objekt. Ausgehend vom Startpunkt, beendet der Algorithmus seine Arbeit, wenn er wieder am Ausgangspunkt angelangt ist. Um die Arbeitsweise des Algorithmus zu vereinfachen, wird in der Implementation jeder bereits gegangene Wegpunkt mit einem weißen Pixel belegt, so dass ein konsistentes Arbeiten des Algorithmus gewährleistet ist.

### 5.5.3. Speicherung von Objektinformation

Ist das Objekt erfolgreich umgangen, so folgt darauf die Generierung von Objektmustern, die der anschließenden Objekterkennung zugeführt werden: Für jedes erkannte Objekt wird ein neues Bild erzeugt, das mit den entsprechenden schwarzen Objektpixeln aus dem reduzierten Quellenmaterial nachgezeichnet und anschließend auf ein 15 x 15 großes Bild<sup>67</sup> reduziert wird. Die so generierten Objektmuster werden im entsprechenden Digitalisatordner abgelegt und dienen als Basismaterial für Mustervergleiche.

Auf die Generierung der Musterbilder folgt die Speicherung der Objektinformation im XML Format. Neben der eindeutigen Zuordnungsnummer des Objektes, sowie der Koordinaten der objektumgebenden Bounding Box werden die Resultate von Berechnungen für anschließende Prozesse der Glyphenbündelung (vgl. Kapitel 5.7.3: „Normalisierung der Eingabemuster“) in der Datei *digitalisatnummer\_index.xml* abgespeichert, die der anschließenden Glyphenzuordnung dienen. Der folgende Auszug aus der XML Datei illustriert die Speicherung von Glypheninformationen für eine Mustergröße von 3 x 3 Pixel:

```
<?xml version="1.0" encoding="UTF-8"?>
<objects>
<glyphobject id="1" winindex="">
<recognitiondata>none</recognitiondata>
<normlength value="0.00246762">
0.372611280996352;0.283776803407818;0.478719129227102;
0.0419496144168079;0.340532164089382;0.283776803407818;
0.308453047182411;0.370143656618893;0.34793503722176;
</normlength>
<boundingbox minx="88" maxx="200" miny="52" maxy="147" />
```

<sup>67</sup> Die Mustergröße lässt sich im Dialog „Einstellungen“ der Applikation festlegen.

```
</glyphobject>
<glyphobject id="2" winindex="">
<recognitiondata>none</recognitiondata>
<normlength value="0.00222527">
0.400549275800586;0.0467307488434017;0.19359881663695;
0.022252737544477;0.0066758212633431;0.391648180782795;
0.376071264501661;0.427252560853958;0.567444807384163;
</normlength>
<boundingbox minx="247" maxx="259" miny="79" maxy="87" />
</glyphobject>
</objects>
```

**Quellcode 1:** Speicherung der Objektinformation in der XML-Datei des verarbeiteten Digitalisates.

Hierbei leitet die XML Deklaration das XML Dokument ein und definiert die XML-Version, sowie den Zeichensatz („UTF-8“) für das Dokument. Das Tag *<objects>* verzeichnet alle gefundenen Objekte – hier wurden zwei Objekte erkannt und gespeichert. Das Objekt mit der *id* 1 wird umschlossen von einem Rechteck (Tag *boundingbox*), das sich über die Koordinaten *minx*, *maxx*, *miny* und *maxy* konstruieren lässt. Vorbereitend auf die Zuordnungs- und Erkennungsprozesse, die im folgenden Kapitel erläutert werden, stellt die Datei *index.xml* Platzhalter bereit für die Zuordnung des Objekts zu ASCII Zeichen (das Tag *recognitiondata*) und der Einordnung in ähnliche Objektkategorien über das Attribut *winindex* im Tag *glyphobject*.

## 5.6. Glyphenbündelung und Glyphenerkennung

Die automatische Objektwahrnehmung generiert einen umfangreichen Fundus von Objekten. Jedes Objekt aus dieser Vielzahl von Objekten muss nun mit Bedeutung belebt, d.h. als eine spezifische Glyphe „erkannt“ werden. Was für den menschlichen Verstand (zumeist) intuitiv funktioniert, die Zuordnung von schwarzen Pixeln zu einem sinnerfüllten Ganzen, stellt algorithmenbasiertes Erkennen vor große Probleme: Wann lässt sich eine Verteilung von Pixeln im zweidimensionalen Raum als ein „n“ erkennen, wann als ein „u“? Eine Möglichkeit des Erkennens bestünde darin, für jede Glyphe eine Vielzahl an Mustern zu erstellen, die als Grundlage für umfangreiche Vergleiche mit den wahrgenommenen Objekten dienen könnte. In Anbetracht der sehr hohen Anzahl von unterschiedlichen Glyphen, die das Bild extrem alter Druckerzeugnisse charakterisiert, wird schnell deutlich, dass der Weg der umfangreichen Mustervergleiche keine adäquate Lösung für das Problem der Glyphenerkennung bietet.

Die vorliegende Ausarbeitung verwendet zwei Arten von künstlichen neuronalen Netzen, um Glyphen zum einen anhand ihrer spezifischen Merkmale zu bündeln, um eine möglichst umfangreiche und konsistente manuelle Zuordnung von Glyphen zu ermöglichen, zum anderen, um noch nicht erkannte Glyphen mit dem Bestand von bereits erkannten Glyphen zu vergleichen. Zu Beginn der folgenden Ausführungen steht ein kurzer Exkurs, der über künstliche Neuronale Netze informiert. Anschließend wird das Glyphenclustering mittels selbstorganisierender Karten und ihre praktische Anwendung in „TED“ erläutert; zum Schluss des Kapitels über die Glyphenerkennung wird das mehrschichtige Perzeptron zur Mustererkennung vorgestellt.

### 5.6.1. Exkurs: Künstliche Neuronale Netze

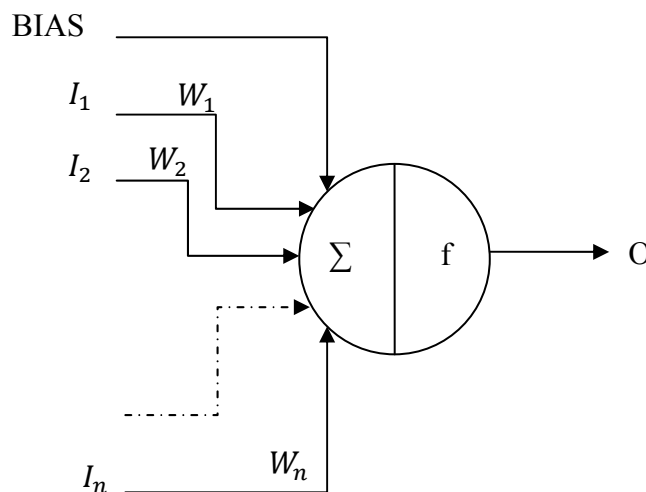
Künstliche neuronale Netze lassen sich beschreiben als „informationsverarbeitende Systeme, deren Struktur und Funktionsweise dem Nervensystem und speziell dem Gehirn von Tieren und Menschen nachempfunden sind.“<sup>68</sup> Basierend auf kleinsten informationstragenden und

---

<sup>68</sup> Borgelt, Christian et al.: Neuro-Fuzzy-Systeme. Wiesbaden: Vieweg Verlag, 2003; S.3.

informationsvermittelnden Einheiten, den Neuronen, bieten künstliche neuronale Netze mit ihrer Fähigkeit, „auf Eingabereize zu reagieren [...], zu lernen und sich der Umgebung entsprechend anzupassen“<sup>69</sup>, ein adäquates Instrumentarium, um komplexe Probleme zu lösen.

Ein künstliches neuronales Netz besteht aus Neuronen (oder „Knoten“<sup>70</sup>), die – je nach Art des Netzes – über vordefinierte Strategien miteinander zu unterschiedlichen Strukturen verbunden sind. Kommunikation innerhalb des Netzes findet statt über die Eigenschaft der Neuronen, Eingabereize zu verarbeiten und abhängig von den Eingabereizen Signale an andere Neuronen zu senden – dieses Verhalten eines Neurons, Signale kontextabhängig zu senden, wird auch als „feuern“ bezeichnet. Jede Eingabe- und Ausgabeverknüpfung eines Neurons ist dabei durch eine Gewichtung gekennzeichnet, die die Stärke der Verknüpfung determiniert. Abbildung 10 zeigt das Modell eines künstlichen Neurons nach McCulloch und Pitts<sup>71</sup>.



**Abbildung 10:** Neuronenmodell nach McCulloch und Pitts. Hierbei bezeichnen  $I_1$  bis  $I_n$  die Eingänge des Neurons,  $O$  den Ausgang,  $BIAS$  den Schwellwert,  $W_1$  bis  $W_n$  die Gewichte,  $\Sigma$  die Eingabefunktion und  $f$  die Aktivierungsfunktion.

Künstliche neuronale Netze verschalten – je nach Anwendungsfall – Neuronen miteinander zu mehreren gegenseitig agierenden Schichten. Lernen innerhalb

<sup>69</sup> Patterson, Dan: Künstliche neuronale Netze – Das Lehrbuch. München: Prentice Hall Verlag, 1997; S. 13.

<sup>70</sup> Vgl. Callan, Robert: Neuronale Netze – Im Klartext. München: Pearson Studium, 2003; S. 17.

<sup>71</sup> Abbildung nach Tizhoosh, Hamid R.: Fuzzy Bildverarbeitung: Einführung in Theorie und Praxis. Berlin / Heidelberg, 1998; S. 155.

dieses Komplexes findet über die Kalibrierung und Änderung einzelner Gewichte statt. Unterschieden wird zwischen überwachtem (supervised) und unüberwachtem (unsupervised) Lernen.

Überwachtes Lernen ist Lernen, das durch den Vergleich des gewünschten Ausgabevektors (zum Beispiel ein zu suchendes Pixelmuster) mit der konkreten Ausgabe des Netzwerks funktioniert. Aus dem Vergleich wird „ein Fehler berechnet, der im Laufe des Lernvorgangs minimiert wird.“<sup>72</sup> Ein künstliches neuronales Netz, das überwacht lernt, ist das mehrschichtige Perzeptron, das an späterer Stelle dieser Ausarbeitung vorgestellt wird.

Unüberwachtes Lernen findet statt, indem es dem Netzwerk „überlassen [wird], sich selbstorganisierend zu ordnen. Eine Lernrate bestimmt dabei die Veränderlichkeit der Neuronen des Netzes.“<sup>73</sup> Je länger der Lernprozess dauert, desto mehr nimmt die Lernrate ab, desto weniger Neuronen werden bei jedem Prozess aktiviert. „Das nicht überwachte Lernen ist ein selbst organisiertes Lernen. Hier wird von außen kein gewünschtes Ergebnis vorgegeben, sondern das Netz versucht selbst, regelhafte Strukturen in den Eingangsdaten zu finden.“<sup>74</sup>

---

<sup>72</sup> Speckmann, Heike: Dem Denken abgeschaut – Neuronale Netze im praktischen Einsatz. Braunschweig / Wiesbaden: Vieweg Verlag, 1996; S. 7.

<sup>73</sup> Ebenda, S. 8.

<sup>74</sup> Vowinkel, Bernd: Maschinen mit Bewusstsein – Wohin führt die künstliche Intelligenz? Weinheim: Wiley-VCH Verlag, 2006; S. 142.

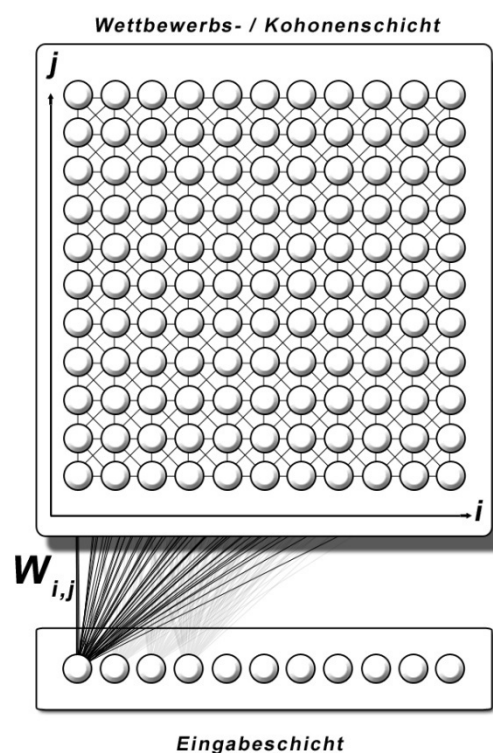


### 5.7. Glyphenbündelung: Selbstorganisierende Karten

Die selbstorganisierende Karte, nach ihrem Entwickler Teuvo Kohonen auch als „Kohonenkarte“, oder „Self-organizing Map“ (SOM) bezeichnet, ist ein unüberwacht lernendes künstliches neuronales Netz. Mit seiner spezifischen Eigenschaft, Eingabedaten nach Ähnlichkeit zu klassifizieren, bietet die selbstorganisierende Karte das Handwerkszeug, um die durch die Objektwahrnehmung gefundenen Objekte zu gruppieren – und somit die manuelle Glyphenzuordnung auf ein Arbeitsminimum zu reduzieren.

Charakterisiert ist die selbstorganisierende Karte durch zwei Schichten: Einer Eingabeschicht und einer Wettbewerbsschicht (auch als „Kohonenschicht“ bezeichnet). Jedes Neuron der Eingabeschicht ist mit jedem Neuron der Wettbewerbsschicht verbunden. Innerhalb der Wettbewerbsschicht ist jedes Neuron mit seinen unmittelbaren Nachbarneuronen verbunden. Determiniert ist die Verbindung jedes Neurons mit seinen benachbarten Neuronen durch ihren jeweiligen Gewichtsvektor  $W_{i,j}$ . Aus dieser Verknüpfung der Neuronen in der

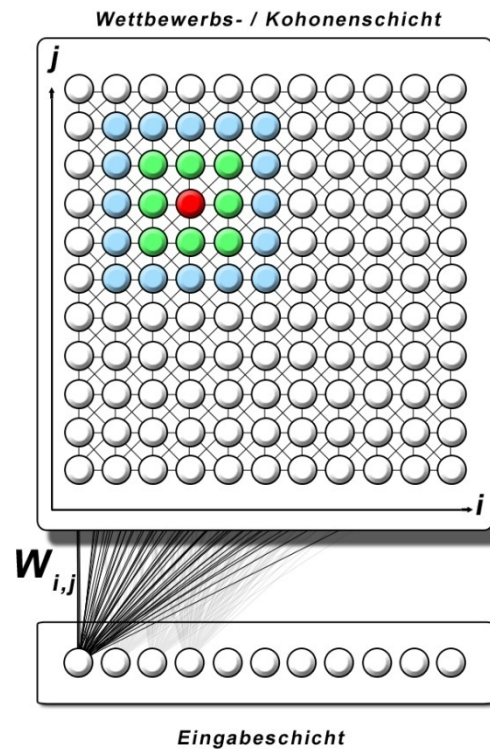
Wettbewerbsschicht resultiert eine topographische Abbildung der Eingabereize. Mit fortlaufendem Training der selbstorganisierenden Karte findet in der Wettbewerbsschicht die Organisation der Eingabereize statt: Ähnliche Reize positionieren sich an ähnlichen Koordinatenpunkten auf der zweidimensionalen Wettbewerbskarte, gleiche Reize lokalisieren sich an gleicher Position auf der Kohonenkarte.



**Abbildung 11:** Selbstorganisierende Karte, bestehend aus Eingabeschicht (unten) und Wettbewerbsschicht (oben).

### 5.7.1. Trainings- und Lernprozess der SOM

Zu Beginn des Trainingsprozesses wird jedes Neuron der Wettbewerbsschicht mit einem zufälligen Gewichtsvektor  $W_{i,j}$  initialisiert. Anschließend wird bei jedem Trainingsschritt ein zufällig ausgesuchter Eingabereiz aus der Eingabeschicht gewählt und der Wettbewerbsschicht präsentiert. Über „die gesamte (Wettbewerbs)Karte wird nach dem Neuron gesucht, das bezüglich eines Ähnlichkeitsmaßes  $D_{i,j}$  den nächsten Gewichtsvektor zum Eingabevektor gespeichert hat.“<sup>75</sup> Über die Koordinaten  $i_{min}$  und  $j_{min}$  des entsprechenden Neurons in der Wettbewerbsschicht lässt sich die Lernfunktion berechnen. Die Lernfunktion bestimmt, inwiefern das Feuern des Neurons seine unmittelbare und mittelbare Nachbarschaft affiziert. Das Feuern des Neurons in der Kohonenschicht hat somit zur Folge, dass die Gewichtung der Nachbarneuronen verändert wird.



**Abbildung 12:** Feuernes Neuron (dunkelrot) und seine Wirkung auf unmittelbar umgebende Neuronen (grün und blau) in der Kohonenschicht.

Die Änderung der Gewichtung erfolgt nach der „Mexican Hat“ Funktion, oder der Gaußschen Glockenfunktion; hierbei werden die Neuronen um das Erregungszentrum herum immer weniger aktiviert, je weiter sie von dem Zentrum entfernt liegen. Für die Berechnung des Ähnlichkeitsmaßes  $D_{ij}$  bietet sich die Euklidische Abstandsbestimmung<sup>76</sup> an:

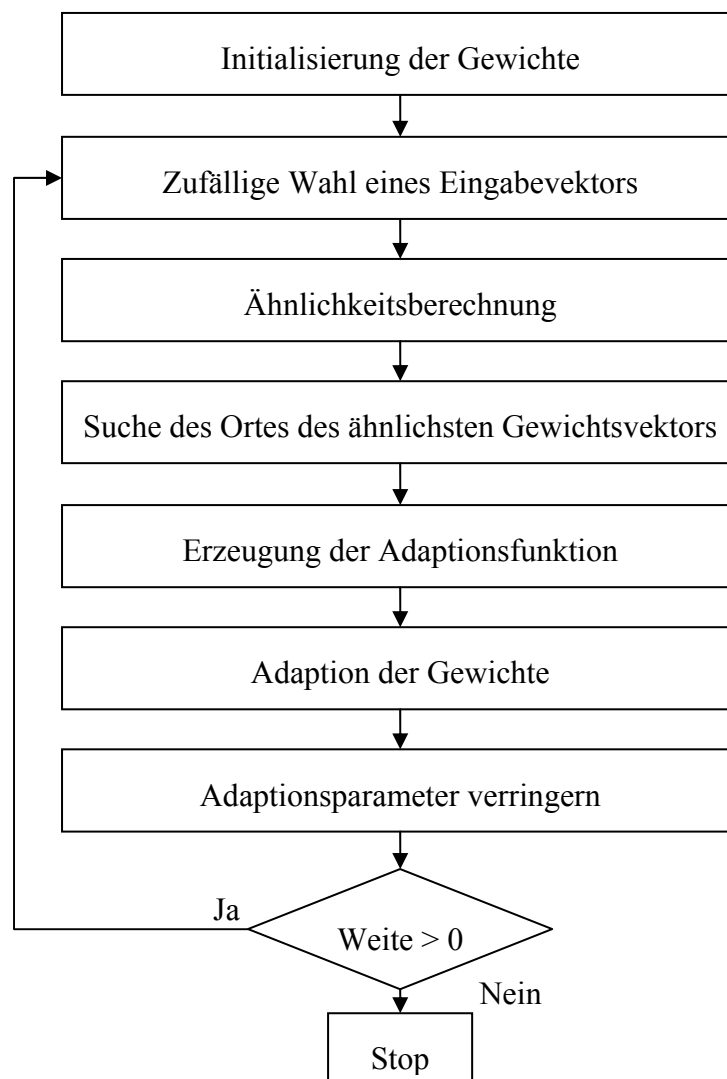
$$D_{ij} = \sum_{k=1}^n (W_{ijk}(t) - x_k(t))^2$$

<sup>75</sup> Speckmann, Heike: Dem Denken abgeschaut – Neuronale Netze im praktischen Einsatz. Braunschweig / Wiesbaden: Vieweg Verlag, 1996; S. 18.

<sup>76</sup> Vgl. Ebenda, S. 19.

Mit steigendem Trainingsaufwand werden die Gewichte dem Eingabereiz ähnlicher. Der Lernprozess endet, wenn die Anzahl der vordefinierten Trainingsschritte erreicht, oder die Kohonenschicht ausgeglichen ist, sodass das Feuern der Neuronen in der Wettbewerbsschicht keinen Einfluss mehr auf die benachbarten Neuronen hat.

Abbildung 13 illustriert den Lernalgorithmus der selbstorganisierenden Karte:



**Abbildung 13:** Lernalgorithmus der Selbstorganisierenden Karte. Quelle: Speckmann, Heike: Dem Denken abgeschaut – Neuronale Netze im praktischen Einsatz. Braunschweig / Wiesbaden: Vieweg Verlag, 1996; S. 18.

### 5.7.2. Implementation der SOM in „TED“

Die vorliegende Ausarbeitung nutzt die selbstorganisierende Karte, um die Vielzahl der nach dem Prozess der Objektisolierung unidentifizierten Glyphen automatisiert zu klassifizieren, zu gruppieren und gebündelt der manuellen Glyphenzuordnung zur Verfügung zu stellen.

Für die Umsetzung der selbstorganisierenden Karte wurde auf die Implementation von Rao<sup>77</sup> zurückgegriffen, die für „TED“ stark angepasst und verändert wurde. Der folgende Abschnitt berichtet über die Umsetzung der selbstorganisierenden Karte in „TED“ und weist auf Vorteile der SOM – und ihre Probleme – hin.

### 5.7.3. SOM: Normalisierung der Eingabemuster

Jede bei der Objektwahrnehmung gefundene Glyphe wird zunächst auf das Vergleichsmaß von (standardmäßig) 15x15 Pixel skaliert und im entsprechenden Digitalisatverzeichnis des Projektordners als unkomprimierte Tiff-Datei gespeichert. Anschließend wird jede Glyphe normalisiert, um als Eingabe für die selbstorganisierende Karte zu fungieren. Rao<sup>78</sup> berechnet hierbei den normalisierten Eingabevektor über die Multiplikation der jeweiligen Pixelwerte mit der Summe aller quadrierten Pixelwerte des 15x15 Pixel großen Eingabemusters: Der normalisierte Eingabevektor  $N_{xy}$  für das Pixel an Position

$$I(x, y) \text{ berechnet sich über den Term } N_{xy} = I(x, y) \cdot \frac{1}{\sqrt{\sum_{i=0}^n \eta_i^2}}.$$

Bereits in der Phase der Objektwahrnehmung werden die normalisierten Eingabevektoren berechnet und in der XML-Datei des Digitalisates abgelegt<sup>79</sup>.

<sup>77</sup> Rao, Valluru B.; Rao, Hayagriva V.: C++ Neural Networks and Fuzzy Logic. M&T Books, IDG Books Worldwide Inc., 1995: Vgl. Die Kapitel 11 (“The Kohonen Self-Organizing Map”) und 12 (“Application for Pattern Recognition”).

<sup>78</sup> Vgl. Ebenda, S. 271-274.

<sup>79</sup> Für den entsprechenden Quellcode zur Berechnung der Normalisierung vgl. die Quell- und Headerdateien zur Objektwahrnehmung im Verzeichnis `./TED/src/boxes/objectfinder.cpp` und `./TED/src/boxes/objectfinder.h`.

#### 5.7.4. SOM: Training

An die Berechnung der normalisierten Eingabevektoren schließt sich das Training der selbstorganisierenden Karte an. Hierbei wird die SOM über die Funktion *int nnkohonen(...)* initialisiert und über die Funktionsargumente kalibriert. Der folgende Quellcodeausschnitt zeigt die Funktion samt Argumentliste<sup>80</sup>:

```
int nnkohonen(QString workdir, QString imageid,  
int inputlayersize, int outputlayersize, float alpha,  
int neighborhood_size, int period, int max_cycles,  
int countobjects)
```

**Quellcode 2:** Initialisierung der Kohonenkarte über die Funktion *int nnkohonen()*.

Neben dem Arbeitsverzeichnis und der Digitalisatnummer innerhalb der Signatur nimmt die Funktion *int nnkohonen(...)* die Größe der Eingabeschicht (*int inputlayersize*) und die Größe der Wettbewerbsschicht (*int outputlayersize*) als Argument entgegen. Die Größe der Eingabeschicht sollte der Gesamtpixelanzahl des Eingabemusters entsprechen – aus der Standardauflösung von 15x15 Pixel folgt eine Eingabeschichtgröße von 225 Neuronen; die Wettbewerbsschicht sollte von gleicher Größe gewählt werden.

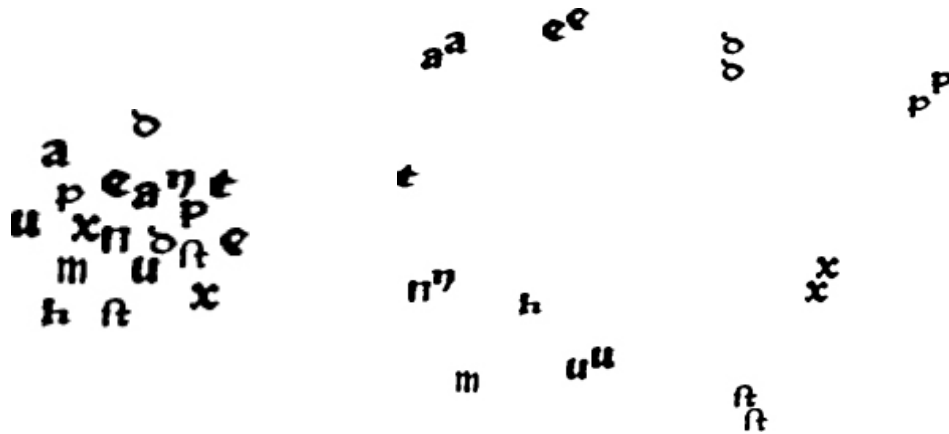
Die folgenden fünf Argumente dienen der Kalibrierung der selbstorganisierenden Karte und wurden von dem Autor der vorliegenden Ausarbeitung empirisch gewonnen. Als Wert für die anfängliche Lernrate  $\alpha$  (*float alpha*), die sich im Laufe des Trainings verringert, leistet ein Wert von 0,25 gute Dienste. Die Größe der Nachbarschaft (*int neighborhood\_size*) verringert sich analog zu  $\alpha$  im Laufe des Trainings und bestimmt, welche Neuronen von dem Feuern eines Neurons in der Wettbewerbsschicht profitieren; gute Ergebnisse ließen sich bei einem Nachbarschaftswert von fünf erzielen. Die verbleibenden drei Argumente bestimmen das Verhalten der selbstorganisierenden Karte: *int period* definiert die Anzahl der Lernschritte, nach denen  $\alpha$  und die Nachbarschaftsgröße dekrementiert werden (gute Ergebnisse bei einem Wert von 100); *int max\_cycles* determiniert die Anzahl der maximalen Lernschritte und *int countobjects* gibt die Anzahl der gesamten, über die Kohonenkarte abzubildenden Objekte an.

---

<sup>80</sup> Für den kompletten Quellcode der SOM-Implementierung sei auf die entsprechenden Quell- und Headerdateien im Verzeichnis *./TED/src/nnkohonen/* der beigelegten DVD verwiesen.

### 5.7.5. „TED“ / SOM: Ergebnisse und Bewertung

Bereits mit 100 Lernschritten bietet die vorliegende selbstorganisierende Karte eine Klassifizierungsleistung mit maximal 30-40% falsch zugeordneten Glyphen – in Anbetracht der Vielzahl unterschiedlicher Glyphen von gleicher Bedeutung ein praktikables Ergebnis. Abbildung 14 zeigt beispielhaft, wie die SOM seine Eingabemuster klassifiziert und auf der Karte verortet:



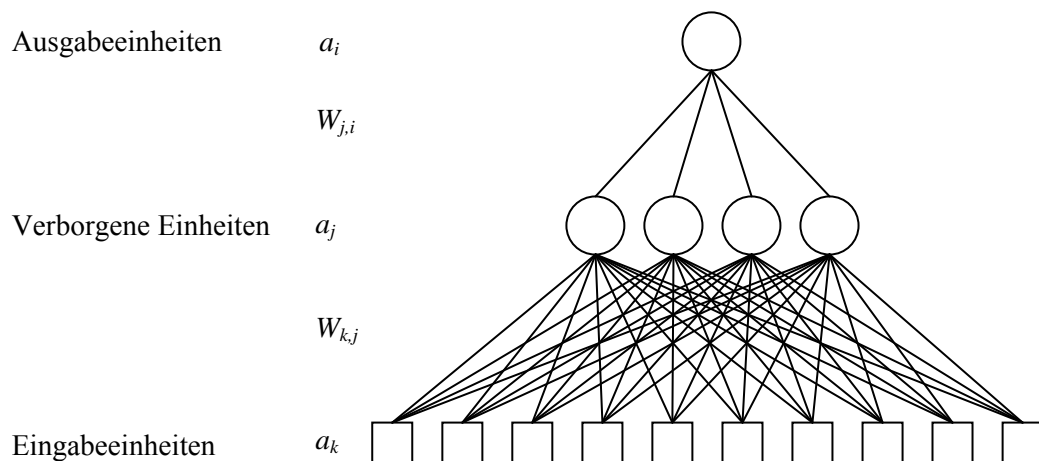
**Abbildung 14:** Beispielhafte Verteilung der Eingabemuster im Raum der Kohonenkarte: Vor dem Training (Links) und nach dem Training (rechts).

Die Erhöhung der Lernschritte resultiert zwar in einer feineren Klassifizierungsleistung der selbstorganisierenden Karte, offenbart zugleich jedoch ihre Problematik: Die selbstorganisierende Karte ist – zumindest in der vorliegenden Implementation – ein mitunter extrem inperformantes Verfahren, um den komplexen und vielfältigen Eingabevektor der Pixelmuster adäquat zu klassifizieren. Eine Alternative für die selbstorganisierende Karte deutet der Ausblick in Kapitel 7 an.

## 5.8. Mustererkennung: mehrschichtiges Perzeptron

Die Glyphenbündelung mittels selbstorganisierender Karte generiert Wissen, das in den signaturweiten Kontext überführt werden soll. Hierfür soll jede noch nicht zugewiesene Glyphen mit jeder bereits erkannten, d.h. zugewiesenen Glyphen verglichen werden. Finden sich äquivalente Glyphen, so soll die Glyphen mit dem ASCII-Wert der bereits bekannten Glyphen belegt werden. Die gestellte Aufgabe der Glyphenvergleiche und -erkennung wird im Folgenden über das mehrschichtige Perzeptron gelöst, ein künstliches neuronales Netz, das überwacht lernt. Auf die Erläuterung des mehrschichtigen Perzeptrons im folgenden Kapitel schließt sich die Besprechung der Umsetzung in „TED“ an. Erfahrungen mit dem mehrschichtigen Perzeptron im Praxiseinsatz schließen das Kapitel zur Mustererkennung ab.

### 5.8.1. Mehrschichtiges Perzeptron: Definition und Arbeitsweise



**Abbildung 15:** Mehrschichtiges Netzwerk, bestehend aus Eingabeschicht, verborgener Schicht und Ausgabeschicht.

Quelle: Russell, Stuart; Norvig, Peter: Künstliche Intelligenz – Ein moderner Ansatz. München: Pearson Studium, 2004; S. 905.

Auch als „mehrschichtiges Feedforward-Netzwerk“<sup>81</sup> bezeichnet, besteht das mehrschichtige Perzeptron aus einer Eingabeschicht, einer – oder mehreren – verborgenen Schichten und einer Ausgabeschicht, deren Neuronen miteinander vorwärtsgerichtet (engl. „feedforward“) verbunden sind: Alle Neuronen der Eingabeschicht sind mit allen Neuronen der verborgenen Schicht verknüpft, jedes

<sup>81</sup> Vgl. Patterson, Dan: Künstliche neuronale Netze – Das Lehrbuch. München: Prentice Hall Verlag, 1997; S. 160ff.

Neuron in der verborgenen Schicht wiederum ist mit jedem Neuron in der Ausgabeschicht verbunden; die Verbindung der Neuronen untereinander ist determiniert über die entsprechenden Gewichte  $W_{j,i}$  und  $W_{k,j}$ . Lernen innerhalb des mehrschichtigen Perzeptrons findet über den Backpropagation Algorithmus statt. Abbildung 15 illustriert den Aufbau eines mehrschichtigen Netzwerkes.

Der Backpropagation Algorithmus vergleicht die Ausgabe des mehrschichtigen Perzeptrons mit der Eingabe, dem trainierten Muster, und bezieht den Fehler, resultierend aus dem Vergleich, in die folgenden Netzdurchläufe ein. Lernen findet somit statt, indem der Backpropagation Algorithmus die Gewichtungen zwischen den Eingabe- und Ausgabeschichten entsprechend anpasst: Der „verborgene Knoten  $j$  [ist] ‘verantwortlich‘ für einen Teil des Fehlers  $\Delta_i$  in jedem der Ausgabeknoten [...], zu dem er eine Verknüpfung herstellt. Die  $\Delta_i$ -Werte sind also entsprechend der Verknüpfungsstärke zwischen dem verborgenen Knoten und dem Ausgabeknoten aufgeteilt und werden zurückgereicht, um die  $\Delta_j$ -Werte für die verborgene Schicht bereitzustellen.“<sup>82</sup>

Russell und Norvig folgend, lässt sich der Backpropagation Algorithmus über folgende Arbeitsschritte zusammenfassen:

- „Berechne die  $\Delta$ -Werte für die Ausgabeeinheiten unter Verwendung des beobachteten Fehlers.
- Wiederhole beginnend mit der Ausgabeschicht das Folgende für jede Schicht im Netzwerk, bis die erste verborgene Schicht erreicht ist:
  - Gib die  $\Delta$ -Werte zurück an die vorhergehende Schicht.
  - Aktualisiere die Gewichtungen zwischen den beiden Schichten.“<sup>83</sup>

### 5.8.2. Mehrschichtiges Perzeptron: Implementation in „TED“

Für die Realisierung des mehrschichtigen Perzeptrons (MLP) in „TED“ zur Glyphenerkennung wurde auf die externe C++ Bibliothek „Libann“<sup>84</sup> zurückgegriffen. „Libann“ stellt – neben der Implementation eines Hopfield-Netzwerkes, einer Boltzmann Maschine und einem Kohonennetzwerk – ein

<sup>82</sup> Russell, Stuart; Norvig, Peter: Künstliche Intelligenz – Ein moderner Ansatz. München: Pearson Studium, 2004; S. 906.

<sup>83</sup> Ebenda.

<sup>84</sup> „Libann“ < <http://www.nongnu.org/libann/index.html>>, Version 1.4 (Mai 2003).



mehrschichtiges Perzeptron bereit, das mit Eingabemustern trainiert werden kann und die Möglichkeit bietet, aus dem Fundus an trainierten Mustern das dem Suchmuster ähnlichste Muster zurückzugeben.

Als Trainings- und Suchdaten nimmt die Implementation des mehrschichtigen Perzeptrons ausschließlich Muster im ASCII-Format entgegen. Vor dem Einsatz des Netzwerkes erfolgt somit zunächst die Umwandlung der in *mlptrain.xml* und *mlpcheck.xml*<sup>85</sup> verzeichneten Glyphen in ASCII Zeichen, die über die Memberfunktion *prepareglyphs(void)* der Klasse *mlpnet* geleistet wird. Hierbei wird jedes schwarze Bildpixel dem Zeichen „#“ und jedes weiße Bildpixel dem Zeichen „.“ (Punkt) Zugeordnet<sup>86</sup>. Abbildung 16 veranschaulicht den Umwandlungsprozess:



**Abbildung 16:** Konvertierung des 15x15 großen Pixelmusters in ASCII-Code.

Auf die Umwandlung der Pixelmuster folgt die Trainingsphase des MLP, bei der jede erkannte Glyphe in das Netz eingespeist wird. Hierbei werden alle in der Datei *mlptrain.xml* verzeichneten Glyphen der Datenstruktur „glyph“ zugeführt, die neben dem Glyphenmuster den Namen (im Folgenden als „Klassifikationsname“ bezeichnet) des zugeordneten ASCII-Zeichens

<sup>85</sup> Die XML Dateien *mlptrain.xml* und *mlpcheck.xml* werden vor der Mustererkennung mit dem mehrschichtigen Perzeptron über die Funktion *ted::performsignaturewidedclustering(...)* dynamisch generiert und im Projektverzeichnis abgelegt. Zum Inhalt haben die beiden Dateien den Index aller zugeordneten (*mlptrain.xml*) und aller unbekannten Glyphen (*mlpcheck.xml*). Für den C++ Quellcode der Glyphenindizierung sei auf die Quelldatei *tedmainwindow.cpp* im Verzeichnis *./TED/src/gui/* verwiesen.

<sup>86</sup> Für den entsprechenden C++ Quellcode zur Umwandlung des Pixelmusters in ASCII-Daten sei an dieser Stelle auf den entsprechenden Quellcode in Anhang I.2 verwiesen.

entgegennimmt. Das Training startet mit der Initialisierung des MLP durch die Funktion *net(...)*. Neben Funktionsargumenten zur Beeinflussung der Lernrate (alpha) oder der Aktivierungsfunktion (float k) ist die Größe der versteckten Schicht von besonderer Relevanz: Wird die versteckte Schicht zu klein oder zu groß dimensioniert, so ufernt die Trainingslaufzeit der Implementation aus ins Unpraktikable – ohne bessere Ergebnisse in der Erkennungsleistung zu generieren. Sehr gute Ergebnisse wurden im Experiment mit einer Größe der versteckten Schicht von einem Zweiunddreißigstel der Eingabemustergroße erzielt – aus der Standardeinstellung 15x15 Pixel folgt so eine Größe der versteckten Schicht von sieben Neuronen.

Ist das MLP trainiert, so lassen sich über die Memberfunktion *recall(...)* Muster in das Netz einspeisen, die klassifiziert werden sollen. Als Rückgabewert liefert die Funktion den Klassennamen des gefundenen Suchmusters.

Auf die Klassifizierung aller Suchmuster folgt schließlich die Zuweisung der erkannten Glyphen: Jede über das MLP klassifizierte Glyphe wird in die Indexdatei des Digitalisates im Projektverzeichnis eingetragen und in der Applikation „TED“ durch einen grünen Rahmen um die Glyphe gekennzeichnet.

### 5.8.3. MLP und „TED“: Evaluation

Analog der selbstorganisierenden Karte hat auch das mehrschichtige Perzeptron mit starken Performanzproblemen zu kämpfen. Um den Trainingsaufwand möglichst gering zu halten, wurde das Training in „TED“ eindimensional gestaltet. Eindimensionales Training bezeichnet hierbei das Training von nur einer Glyphe aus einem Fundus an äquivalenten Glyphen einer Glyphenklasse. Ein Beispiel mag die Eindimensionalität des Trainings verdeutlichen: Wurden mittels selbstorganisierender Karte zahlreiche Glyphen **S** mit dem ASCII-Wert „s“ belegt, so schöpft das MLP-Training aus dieser Vielfalt an unterschiedlichen graphischen Repräsentationen für „s“ nur eine einzige aus.

Trotz dieser Einschränkung liefert das mehrschichtige Perzeptron eine Erkennungsrate von rund 80-90%<sup>87</sup> korrekt klassifizierter Glyphen – jedoch mit der Eigenschaft, auch nur das zu erkennen, was es trainiert hat und keine explizite Rückmeldung über Glyphen zu geben, die dem Netz unbekannt sind. Jene unbekannten Glyphen werden folglich mit sehr hoher Wahrscheinlichkeit falsch klassifiziert. Aus diesem Grunde wird die Mustererkennung des MLP ergänzt durch einen direkten Mustervergleich, der das „erkannte“ Muster mit dem Trainingsmuster zeichenweise vergleicht und über den Grad der direkten Übereinstimmung entscheidet, ob das vom MLP erkannte Muster korrekt klassifiziert wurde.

---

<sup>87</sup> Die Erkennungsleistung ist vorsichtig formuliert aufgrund der Vielfältigkeit des zu analysierenden Materials; eine hinreichend genaue Evaluation der Klassifizierungsleistung muss diese Ausarbeitung schuldig bleiben.

### 5.9. Aufbereitung der OCR-Ergebnisse

Die Ergebnisse der OCR sind freilich minder interessant, wenn sie nicht visualisiert werden. Um die Ergebnisse der OCR darzustellen, kombiniert „TED“ die Auszeichnungssprache XHTML mit Cascading Style Sheets (CSS). Hierbei wird für jedes Digitalisat eine separate XHTML- und CSS Datei generiert, die im Ordner *output* des Projektverzeichnisses abgelegt wird. Für jede Glyphe im Digitalisat, die erkannt wurde, d.h. der ein Zeichen aus dem ASCII-Code zugeordnet ist, wird in der CSS-Datei ein Selektor<sup>88</sup> erzeugt, der mit seinen Eigenschaftswerten die Position der Glyphe in der XHTML Datei definiert. Die Position der Glyphe ist definiert als die rechte untere Ecke der Boundingbox. Anschließend wird jede Glyphe über das *<div>* Tag mit der entsprechenden GlyphenID in das XHTML Dokument auf dem XHTML Dokument verortet. Die folgenden Quellcodeauszüge mögen das Verfahren erläutern; Abbildung 17 illustriert die Darstellung der OCR Ergebnisse.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="de" lang="de">
<head><title>output</title>
<link rel="stylesheet" href="1.css"
type="text/css" />
</head><body>
<div id="object_1">I</div>
<div id="object_3">fi</div>
<div id="object_4">r</div>
<div id="object_8">t</div>
<div id="object_9">de</div>
</body></html>
```

```
#object_1
{
position:
absolute;
left: 88px;
top: 147px;
}
#object_3
{
position:
absolute;
left: 340px;
top: 132px;
}
```

**Quellcode 3:** XHTML Markup (links) und entsprechende CSS Selektoren (rechts).

<sup>88</sup> Der „Selektor“ dient in der CSS Datei der Auszeichnung des entsprechenden Elementes in der XHTML Datei. So wählt der Selektor `#object_1` im Quellcodebeispiel 3 das Element mit der `id="object_1"` in der XHTML Datei aus und positioniert es den Eigenschaften „position: absolute“, „left: 88px“ und „top: 147px“ entsprechend.

De voluntate dei . xxxij .  
 De iusticia dei . xxxij .  
 De misericordia dei . xxxij .

D e v o l u n t a t e d e i . x x x i j .  
 D e i u s t i c i a d e i . x x x i j .  
 D e m i s e r i c o r d i a d e i . x x x i j .

**Abbildung 17:** Bearbeitetes Digitalisat<sup>89</sup> (oben) mit zugeordneten, d.h. grün markierten Glyphen und die entsprechende Ausgabe über die XHTML Datei mit absolut positionierten Glyphen (unten).

<sup>89</sup> Albertus Magnus ADB Zedler Wiki: Compendium theologiae veritatis. Add: Bernoldus de Caesarea: Distinctiones de tempore et de sanctis quarum declarationes ex compendio ... capiuntur (ISTC Nr.: ia00234000).

## 6. „TED“ – TED Enhances Digitization

Wie Abbildung 18 veranschaulicht, kapselt „TED“ die Arbeitsschritte der Vorverarbeitung, Glyphenisolierung, des Glyphenclustering, der Glyphenerkennung und Glyphenzuweisung über zentrale Funktionsgruppen, über große Symbole (Buttons), die sich am oberen Rand des Fensters befinden:

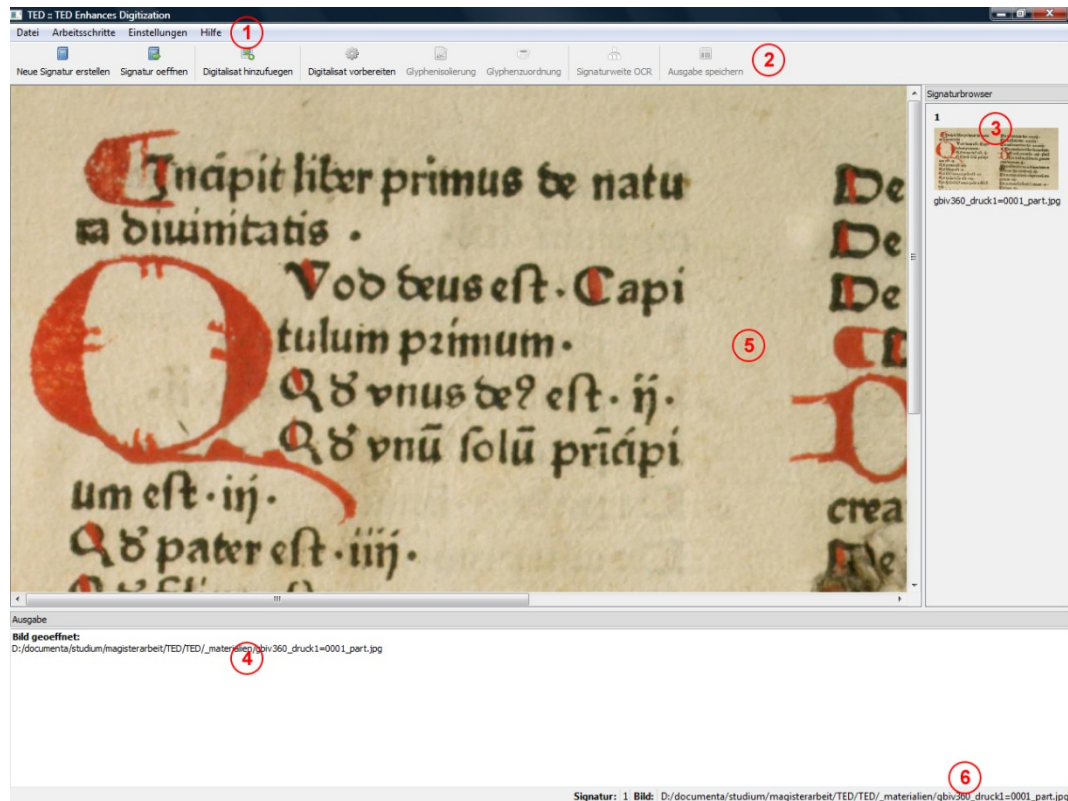


Abbildung 18: Funktionseinheiten von „TED“.

- 1.) Menüleiste: Hier finden sich alle Funktionen zur Arbeit mit „TED“, wie das Anlegen neuer Signaturen, das Starten der verschiedenen Arbeitsschritte, die Beeinflussung und Kalibrierung der Arbeitsschritte über den „Einstellungen“-Dialog oder weiterführende Informationen über die Applikation („Hilfe“).
- 2.) Die Werkzeugleiste macht alle Arbeitsschritte über große Buttons zugänglich.
- 3.) Einzelne Digitalisate einer Signatur lassen sich im Signaturbrowser ansteuern.
- 4.) Über den Vorgang der spezifischen Arbeitsprozesse informiert das Ausgabefenster.

- 5.) Die zentrale Arbeitsfläche ist Mittelpunkt der Applikation. Hier findet die Arbeit mit Digitalisaten statt, die über den Signaturbrowser ausgewählt wurden.
- 6.) Die aktuell bearbeitete Signatur und das aktuell bearbeitete Digitalisat werden in der Statusleiste angezeigt.

### 6.1. „TED“ – Workflow

Die Arbeit mit „TED“ gestaltet sich über acht Arbeitsschritte, die im Folgenden erläutert werden.

- 1.) Neue Signatur anlegen oder bestehende Signatur öffnen: Über den Button „Neue Signatur erstellen“ wird eine Projektdatei mit der Dateiendung \*.ted angelegt und um einen Ordner ergänzt, der die Digitalisate des Projekts aufnimmt. Das Anlegen einer neuen Signatur erfordert die Angabe eines Signaturnamens und eines Projektverzeichnis, das als Container für die weitere Arbeit mit der Signatur und ihren Digitalisaten dient. Verankert in der Projektdatei werden Informationen über das Arbeitsverzeichnis, das Erstellungsdatum der Signatur und die enthaltenen Digitalisate (siehe Quellcodeauszug 4).

```
<?xml version='1.0' encoding='UTF-8'?>
<project workdirectory="D:/_tedtestdata" created="Do 7. Aug
18:10:33 2008" name="gbiv360" >
  
</project>
```

**Quellcode 4:** Projektdatei mit Angabe des Arbeitsverzeichnisses, des Erstellungsdatums und der Projektdigitalisate.

„TED“ geht – wie in Kapitel 4 angedeutet – davon aus, dass verschiedene Digitalisate einer Signatur dadurch gekennzeichnet sind, dass sie in ihrer Typographie ähnliche Merkmale aufweisen. Die Arbeitshypothese von „TED“ gründet die Praktikabilität: Lassen sich hinreichend genaue Aussagen über ein Digitalisat einer Signatur treffen, so lässt sich dieses Wissen auch auf die anderen Digitalisate der Signatur anwenden.

- 2.) Digitalisate der Signatur hinzufügen: Über den entsprechenden Button werden Digitalisate dem Projekt und Signaturbrowser hinzugefügt. Für jedes Digitalisat wird ein entsprechender Unterordner im Projektverzeichnis erstellt und das Digitalisat mit der Projektdatei verknüpft.

- 3.) Über den Signaturbrowser wird das Digitalisat ausgewählt und anschließend über den Button „Digitalisat vorbereiten“ für die weiteren Isolierungs- und Erkennungsprozesse vorbereitet.
- 4.) Auf die Digitalisatvorbereitung folgt die Glyphenisolierung. Hierbei werden Glyphen im Digitalisat wahrgenommen, daraufhin im Digitalisatordner des Projekts für jede Glyphe ein Muster gespeichert und schließlich eine Indexdatei im XML-Format für das entsprechende Digitalisat im Projektordner erstellt.
- 5.) „TED“ versucht, ähnliche und gleiche Glyphen zu bündeln, um eine möglichst konsistente und einfache Zuordnung zu ermöglichen. Über den Button „Glyphenzuordnung“ beginnt die Suche nach ähnlichen und gleichen Glyphen.
- 6.) Nachdem Glyphen mit ähnlichen Merkmalen durch die selbstorganisierende Karte gebündelt wurden, lässt sich über einen Mouseklick auf die gewünschte Glyphe in der Arbeitsfläche der Informations- und Bearbeitungsdialog aufrufen (Abbildung 19).

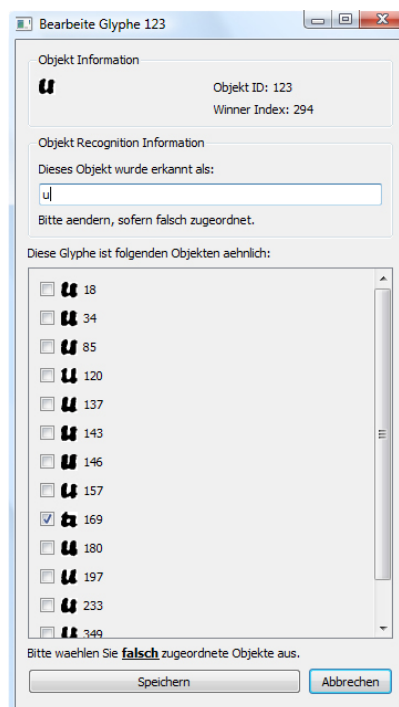


Abbildung 19: Dialog zur Klassifizierung von Glyphen in „TED“.

Neben Informationen zur Objektnummer oder dem Winner Index der Glyphe, resultierend aus dem Kohonenttraining, lässt sich über den Bearbeitungsdialog die gewählte Glyphe mit den vorgeschlagenen ähnlichen Glyphen vergleichen,



falsche Glyphen entfernen und die gewählte Glyphe mit einem Zeichen aus dem ASCII Code belegen (standardmäßig mit der Zeichenkette „none“ belegt für noch nicht zugeordnete Glyphen). Abbildung 19 zeigt den Dialog, der für die gewählte Glyphe „u“ zahlreiche Vorschläge für ähnliche Glyphen macht. Der falsche Vorschlag mit der Objektnummer 169 wird über einen Klick auf die nebenstehende Checkbox ausgewählt, um die offensichtlich falsch zugeordnete Glyphe aus dem Fundus der „u“-Glyphen auszuschließen. Nach Zuweisung der Glyphe und ihren automatisch erkannten ähnlichen Glyphen wird der Dialog über den Button „speichern“ geschlossen und die Änderung digitalisatweit angewendet. Die Einfärbung der die Glyphen umgebenden Bounding Box gibt Aufschluss über den Status der Glyphe. Hierbei bezeichnet ein grünes umgebendes Rechteck eine erfolgreich zugewiesene Glyphe, ein blaues Rechteck eine Glyphe, die als falsch zugeordnet erkannt wurde und ein rotes Rechteck eine Glyphe, die noch nicht einem Zeichen des ASCII Codes zugewiesen wurde (siehe Abbildung 20).

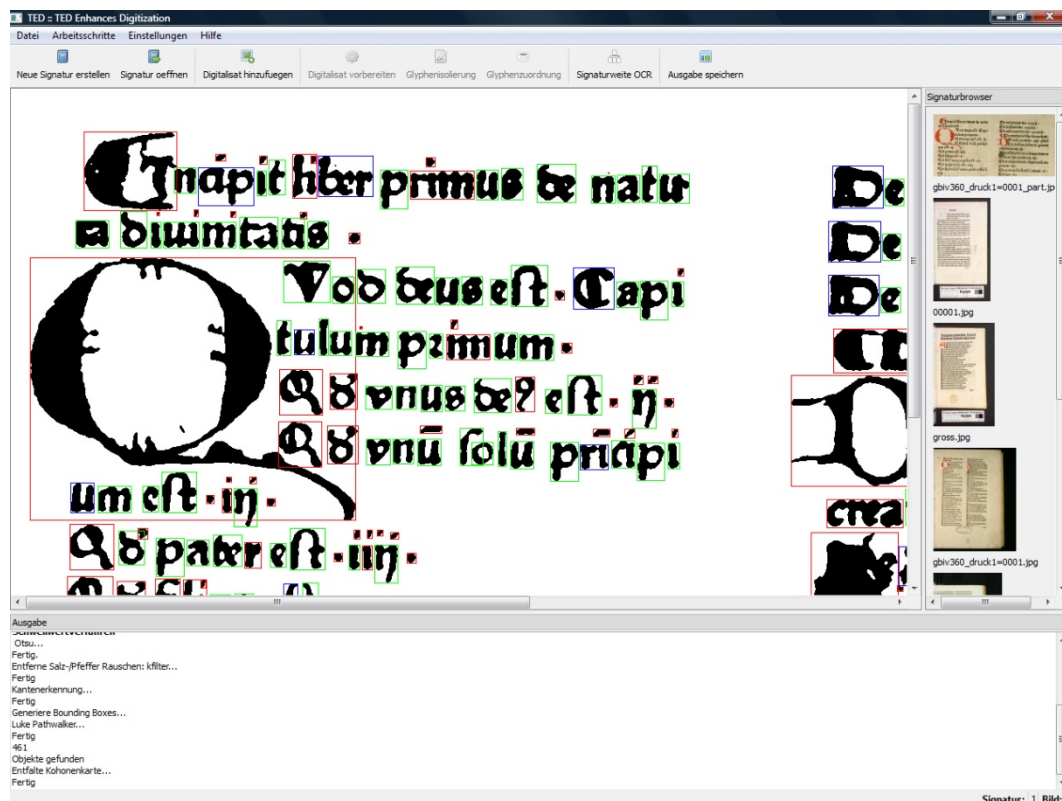


Abbildung 20: Das Arbeitsfenster von „TED“ mit zugeordneten (grün), nicht zugeordneten (rot) und als falsch klassifiziert erkannten Glyphen (blau).

- 7.) Im Laufe der Arbeit auf dem Digitalisat bildet sich eine konsistente Wissensbasis, die über den Arbeitsschritt „Signaturweite OCR auf die gesamte Signatur übertragen wird.
- 8.) Über den Button „Ausgabe speichern“ wird das Ergebnis der OCR als XHTML Datei im Ordner „*output*“ des Projektverzeichnisses gespeichert.

## 7. Fazit und Ausblick

Auf den vorangegangenen Seiten wurde das System „TED“ beschrieben und entwickelt, das auf die komplexen Herausforderungen extrem früher Drucke zu reagieren versucht über intensive Vorbereitung des Quellenmaterials und der automatischen Texterkennung über die Kombination der selbstorganisierenden Karte zur Glyphenklassifizierung und dem mehrschichtigen Perzeptron zur Glyphenerkennung. Die Arbeit mit „TED“ ist zunächst charakterisiert durch hohen manuellen Aufwand: Einzelne Zeichen der Digitalisate einer Signatur werden dem System bekannt gemacht und im Verlauf der Arbeit eine (möglichst) konsistente Wissensbasis generiert, die auf alle Digitalisate der Signatur angewendet wird.

Deutlich wurde im Rahmen der Ausarbeitung, dass die Kombination aus selbstorganisierender Karte und mehrschichtigem Perzeptron – neben durchschnittlicher Erkennungsleistung – durch starke Performanceprobleme gekennzeichnet ist. Mit Rückgriff auf Russell / Norvig<sup>90</sup> lässt sich ausblicken auf Support-Vektormaschinen (SVM), bzw. Kernelmaschinen, um performante und erkenntungsstarke Musterklassifizierung und -erkennung zu leisten<sup>91</sup>. Support-Vektormaschinen transformieren Merkmale in einen mehrdimensionalen Raum und ermöglichen die lineare Trennung der Daten, vereinen das „Beste aus beiden Welten“<sup>92</sup> einschichtiger und mehrschichtiger neuronaler Netzwerke.

---

<sup>90</sup> Russell, Stuart; Norvig, Peter: Künstliche Intelligenz – Ein moderner Ansatz. München: Pearson Studium, 2004.

<sup>91</sup> Vgl. die Fallstudie „handschriftliche Ziffernerkennung“ in: Ebenda, S. 914 ff. Hier positioniert sich das Verfahren einer selbst verbessernden, virtuellen Support-Vektormaschine als äußerst performante und leistungsfähige Alternative zu neuronalen Netzwerkstrukturen.

<sup>92</sup> Ebenda, S. 910.

## 8. Literaturverzeichnis

**Antonacopoulos, A.;** Gatos, B.; Bridson, D.: ICDAR2007 Page Segmentation Competition. 9th International Conference on Document Analysis and Recognition (ICDAR'07). Curitiba, Brazil: September 2007, S. 1279-1283.  
<[http://www.iit.demokritos.gr/~bgat/Icdar2007\\_PageSegmentationCompetition.pdf](http://www.iit.demokritos.gr/~bgat/Icdar2007_PageSegmentationCompetition.pdf)> (27.08.2008)

**Borgelt, Christian;** Klawonn, Frank; Kruse, Rudolf; Nauck, Detlef: Neuro-Fuzzy-Systeme. Wiesbaden: Vieweg Verlag, 2003.

**Bourg, David M.;** Seeman, Glenn: AI for Game Developers – Creating Intelligent Behavior in Games. Sebastopol: O'Reilly Media, Inc., 2004.

**Brill, Manfred:** Mathematik für Informatiker – Einführung an praktischen Beispielen aus der Welt der Computer. München: Carl Hanser Verlag, 2001.

**Bulacu, Marius;** van Koert, Rutger; Schomaker, Lambert; van der Zant, Tijn: Layout Analysis of Handwritten Historical Documents for Searching the Archive of the Cabinet of the Dutch Queen. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol. 1 (S. 357-361).

**Callan, Robert:** Neuronale Netze – Im Klartext. München: Pearson Studium, 2003.

**Calvert, David;** Guan, Jiawen: Distributed Artificial Neural Network Architectures. In Proceedings of the 19<sup>th</sup> International Symposium on High Performance Computing Systems and Applications (HPCS'05), 2005.

**Chen, Chih-Ming;** Chen, Chien-Chang; Chen, Char-Chin: A Comparison of Texture Features Based on SVM and SOM. In: The 18<sup>th</sup> International Conference on Pattern Recognition (ICPR'06).

**Cronin, Alex;** Fitzgerald, John; Kechadi, Tahar: A Hybrid Recogniser for Handwritten Symbols Based on Fuzzy Logic and Self-Organizing Maps. In: Proceedings of the 18<sup>th</sup> International Conference on Tools with Artificial Intelligence (ICTAI'06).

**Ertel**, Wolfgang: Grundkurs Künstliche Intelligenz. Eine praxisorientierte Einführung. Wiesbaden: Vieweg Verlag, 2008.

**Feldmann**, Bianca: OCR von Handschriften. Ein Forschungsüberblick. In: Thaller, Manfred (Hrsg.): Codices Electronici Ecclesiae Coloniensis. Eine mittelalterliche Kathedralbibliothek in digitaler Form. Göttingen: Fundus – Forum für Geschichte und ihre Quellen. Beiheft 1, 2001.

<<http://webdoc.gwdg.de/edoc/p/fundus/1/feldmann.pdf>> (27.08.2008)

**Fraunhofer** Institut für Intelligente Analyse- und Informationssysteme: Bestandsaufnahme zur Digitalisierung von Kulturgut und Handlungsfelder. Erstellt im Auftrag des Beauftragten für Kultur und Medien (BMK) unter finanzieller Beteiligung des Bundesministerium für Bildung und Forschung, Januar 2007.

< <http://www.bundesregierung.de/Content/DE/Artikel/2007/01/anlagen/2007-01-11-fraunhofer-studie-pdf-format,property=publicationFile.pdf>> (27.08.2008)

**Fukunaga**, Keinosuke: Introduction to Statistical Pattern Recognition – Second Edition. Boston: Academic Press, 1990.

**Geldner**, Ferdinand: Inkunabelkunde: Eine Einführung in die Welt des frühesten Buchdrucks. Wiesbaden: Reichert, 1978. (Elemente des Buch- und Bibliothekswesens; Band 5).

**Giersberg**, Dagmar: Inkunabeln wandern ins Netz. Berlin: Goethe-Institut, 2006. <<http://www.goethe.de/wis/med/thm/cpw/de1607878.htm>> (18.07.2008)

**Gonzalez**, Rafael C.; Woods, Richard E.: Digital Image Processing – Third Edition. New Jersey: Pearson Education, 2008.

**Hartmann**, Peter: Mathematik für Informatiker – Ein praxisbezogenes Lehrbuch. Braunschweig / Wiesbaden: Vieweg Verlag, 2003.

**Hugo**, Victor: Der Glöckner von Notre-Dame. Stuttgart, Hamburg, München: Deutscher Bücherbund.

**Jähne**, Bernd: Digitale Bildverarbeitung. Berlin, Heidelberg: Springer-Verlag, 2005.

**Kommission** der Europäischen Gemeinschaften: Europas kulturelles Erbe per Mausklick erfahrbar machen, Stand der Digitalisierung und Online-Verfügbarkeit kulturellen Materials und seiner digitalen Bewahrung in der EU, Brüssel, 11.08.2008.

<[http://ec.europa.eu/information\\_society/activities/digital\\_libraries/doc/communications/progress/communication\\_de.pdf](http://ec.europa.eu/information_society/activities/digital_libraries/doc/communications/progress/communication_de.pdf)> (27.08.2008)

**Lippman**, Stanley B.; Lajoie, Josée: C++. Bonn: mitp-Verlag, 2003.

**Marinai**, Simone; Marino, Emmanuele; Soda, Giovanni: Transformation invariant SOM clustering in Document Image Analysis. In: 14<sup>th</sup> International Conference on Image Analysis and Processing (ICIAP 2007).

**Mazal**, Otto: Paläographie und Paläotypie. In: Hellinga, Lotte; Härtel, Helmar (Hrsg.): Buch und Text im 15. Jahrhundert: Arbeitsgespräch in d. Herzog-August-Bibliothek Wolfenbüttel vom 1.-3. März 1978. Hamburg: Hauswedell, 1981.

**Michie**, D.; Spiegelhalter, D. J.; Taylor, C. C.: Machine Learning, Neural and Statistical Classification. 1994.

<<http://www.amsta.leeds.ac.uk/~charles/statlog/whole.pdf>> (27.08.2008)

**O’Gorman**, Lawrence; Sammon, Michael J.; Seul, Michael: Practical Algorithms for Image Analysis – Description, Examples, Programs, and Projects. Cambridge: Cambridge University Press, 2008.

**Pandya**, Abhijit S.; Macy, Robert S.: Pattern Recognition with Neural Networks in C++. Boca Raton: CRC Press, 1995.

**Parker**, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997.

**Patterson**, Dan: Künstliche neuronale Netze – Das Lehrbuch. München: Prentice Hall Verlag, 1997.

**Pitas**, Ioannis: Digital Image Processing – Algorithms and Applications. Chichester: John Wiley & Sons, 2000.

**Rao**, Valluru B.; Rao, Hayagriva V.: C++ Neural Networks and Fuzzy Logic. M&T Books, IDG Books Worldwide Inc., 1995.

**Russell**, Stuart; Norvig, Peter: Künstliche Intelligenz – Ein moderner Ansatz. München: Pearson Studium, 2004.

**Sezgin**, Mehmet; Sankur, Bülent: Survey over image thresholding techniques and quantitative performance evaluation. In: Journal of Electronic Imaging Vol. 13, 146 (2004). S. 146-165.

**Shafait**, Faisal; Keysers, Daniel; Breuel Thomas M: Performance Evaluation and Benchmarking of Six Page Segmentation Algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence. Juni 2008 (Vol. 30, No. 6); S. 941-954.

**Speckmann**, Heike: Dem Denken abgeschaut – Neuronale Netze im praktischen Einsatz. Braunschweig / Wiesbaden: Vieweg Verlag, 1996.

**Steyer**, Timo: Verteilte Digitale Inkunabelbibliothek – Ein Baustein zur Gesamtdigitalisierung aller Inkunabelausgaben. In: Historisches Forum. Themenhefte von Clio-online Bd. 10/I (2007), S. 294-308.  
<[http://www.clio-online.de/site/lang\\_\\_de/40208224/Default.aspx](http://www.clio-online.de/site/lang__de/40208224/Default.aspx)> (27.08.2008)

**Stroustrup**, Bjarne: Die C++ Programmiersprache. Bonn: Addison-Wesley-Longman Verlag, 1998.

**Takashi** Hirano, Yuichi Okano, Yasuhiro Okada, Fumio Yoda: Text and Layout Information Extraction from Document Files of Various Formats Based on the Analysis of Page Description Language. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol. 1 (S. 262-266).

**Thaller**, Manfred et. al.: Retrospektive Digitalisierung von Bibliotheksbeständen – Evaluierungsbericht über einen Förderschwerpunkt der DFG. Köln, 2005.  
<[http://www.dfg.de/forschungsfoerderung/wissenschaftliche\\_infrastruktur/lis/download/retro\\_digitalisierung\\_eval\\_050406.pdf](http://www.dfg.de/forschungsfoerderung/wissenschaftliche_infrastruktur/lis/download/retro_digitalisierung_eval_050406.pdf)> (27.08.2008)

**Tizhoosh**, Hamid R.: Fuzzy Bildverarbeitung: Einführung in Theorie und Praxis. Berlin / Heidelberg, 1998.

**Webb**, Andrew R.: Statistical Pattern Recognition – Second Edition. Chichester: John Wiley & Sons, 2002.

**Wolf**, Jürgen: Qt 4 GUI-Entwicklung mit C++. Bonn: Galileo Computing, 2007.

**Vowinkel**, Bernd: Maschinen mit Bewusstsein – Wohin führt die künstliche Intelligenz? Weinheim: Wiley-VCH Verlag, 2006.

**Yan Yu**, Pilian He; Yushan Bai; Zhenlei Yang: A Document Clustering Method Based on One-Dimensional SOM. In: Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008); S. 295-300.

Die Erreichbarkeit aller Verweise auf externe Quellen im WWW wurde überprüft am 27.08.2008.



## 9. Verzeichnis externer Ressourcen

*k*Fill Algorithmus von Wakahara, Toru: <http://cis.k.hosei.ac.jp/~wakahara/kfill.c>  
(27.08.2008)

Trolltech Qt 4.4.1 (31.07.2008)

Libann - Artificial Neural Network Library: <http://www.nongnu.org/libann/>  
(27.08.2008).

“TED” verwendet Icons von <http://www.famfamfam.com>

QuickSort Algorithmus:

<http://www.dreamincode.net/forums/showtopic31409.htm> (27.08.2008)

## Appendix I: C++ Quellcode

### 1. Kantenisolierung (*./TED/src/edge\_detection/grad1.cpp*)

Basierend auf der Implementation des Verfahrens von: Parker, James R.: Algorithms for Image Processing and Computer Vision. Chichester: John Wiley & Sons, 1997; S. 10-12 / G1.

```
#include <QImage>

#include "grad1.h"
#include <math.h>

void grad1(QImage *image)
{
    int i,j,k;
    double z, dx, dy;
    int pmax=0, pmin=255, thresh = 128;

    QRgb *RGB;
    QRgb *RGBTemp;
    int width=image->width();
    int height=image->height();

    for (i=height-1; i>=1; i--)
        for (j=width-1; j>=1; j--)
        {
            RGB=(QRgb *)image->scanLine(i)+j;
            RGBTemp=(QRgb *)image->scanLine(i-1)+j;
            dx=qRed(*RGB)-qRed(*RGBTemp);

            RGB=(QRgb *)image->scanLine(i) + j;
            RGBTemp=(QRgb *)image->scanLine(i) + j-1;
            dy =qRed(*RGB)-qRed(*RGBTemp);

            z=sqrt(dx*dx + dy*dy);

            if (z > 255.0) z=255;
            else if(z< 0.0) z = 0;

            RGB=(QRgb *)image->scanLine(i) + j;
            *RGB=qRgb(z, z, z);

            if (pmax < (unsigned char)z) pmax = (unsigned char)z;
            if (pmin > (unsigned char)z) pmin = (unsigned char)z;
        }
    thresh = (pmax-pmin)/2;

    for (i=0; i<height; i++)
    {
        RGB=(QRgb *)image->scanLine(i)+0;
        *RGB=qRgb(0, 0, 0);

        RGB=(QRgb *)image->scanLine(i)+width-1;
        *RGB=qRgb(0, 0, 0);
    }
}
```

```

    for (j=0; j<width; j++)
    {
        RGB=(QRgb *)image->scanLine(0)+j;*RGB=qRgb(0, 0, 0);

        RGB=(QRgb *)image->scanLine(height-1)+j;
        *RGB=qRgb(0, 0, 0);
    }

    /*    Threshold    */
    for (i=0; i<height; i++)
        for (j=0; j<width; j++)
        {
            RGB=(QRgb *)image->scanLine(i)+j;
            if (qRed(*RGB) > thresh)
            {
                RGB=(QRgb *)image->scanLine(i)+j;
                *RGB=qRgb(0, 0, 0);
            }
            else
            {
                RGB=(QRgb *)image->scanLine(i)+j;
                *RGB=qRgb(255, 255, 255);
            }
        }
    }
}

```

## 2. Konvertierung Tiff nach ASCII (Auszug aus *./TED/src/nnmlp/nnmlp.cpp*)

```

void mlpnet::prepareglyphs(void)
{
    // Create ASCII Glyphs based on input patterns
    // Step 1) Prepare Training Glyphs
    QString mlptraindirectory="";
    QString charglyph="";
    QDomElement docElem=trainingxml.documentElement();
    QDomNode node=docElem.firstChild();
    while(!node.isNull())
    {
        QDomElement e=node.toElement();
        if(!e.isNull())
        {
            QDomAttr curdocid=e.attributeNode("docid");
            QDomAttr
curobjectid=e.attributeNode("objectid");

            // Open Image files; write pattern into RAM
            QString tempimgstring=workdirectory;
            tempimgstring+=curdocid.value();
            tempimgstring+="/";
            mlptraindirectory=tempimgstring;
            mlptraindirectory+="mlptrain";
            tempimgstring+=curobjectid.value();
            tempimgstring+="tif";
            QRgb *RGB;
            QImage tempimage;
            tempimage.load(tempimgstring);

            charglyph="";
            for (int y=0; y < tempimage.height(); y++)

```

```

        {
            for (int x=0; x < tempimage.width(); x++)
            {
                RGB=(QRgb *)tempimage.scanLine(y) +
x;

                int test=qRed(*RGB);
                if(qRed(*RGB)>155)
                {
                    charglyph+=".";
                }
                else
                {
                    charglyph+=".";
                }
            }
            charglyph+="\n";
        }
        // If not created: create TrainingDir
        QDir dir(mlptraindirectory);
        if(!dir.exists())
        {
            // Create Directory
            dir.mkpath(mlptraindirectory);
        }
        // Save tif as ascii
        QString curmlptrainglyph=mlptraindirectory;
        curmlptrainglyph+="/";
        curmlptrainglyph+=curobjectid.value();
        curmlptrainglyph+=".char";
        QFile file_mlptrainfile(curmlptrainglyph);
        file_mlptrainfile.open(QIODevice::WriteOnly);
        QTextStream
out_file_mlptrainfile(&file_mlptrainfile);
        out_file_mlptrainfile << charglyph;
        file_mlptrainfile.close();
        // ***
    }
    node=node.nextSibling();
}

// Step 2) Prepare Glyphs to recognize
QString mlptoclassifydirectory="";
docElem=checkxml.documentElement();
node=docElem.firstChild();
while(!node.isNull())
{
    QDomElement e=node.toElement();
    if(!e.isNull())
    {
        QDomAttr curdocid=e.attributeNode("docid");
        QDomAttr
curobjectid=e.attributeNode("objectid");

        // Open Image files; write pattern into RAM
        QString tempimgstring=workdirectory;
        tempimgstring+=curdocid.value();
        tempimgstring+="/";
        mlptoclassifydirectory=tempimgstring;
        mlptoclassifydirectory+="mlptoclassify";
        tempimgstring+=curobjectid.value();
    }
}

```

```

        tempimgstring+=".tif";
        QRgb *RGB;
        QImage tempimage;
        tempimage.load(tempimgstring);
        QImage testing=tempimage;
        charglyph="";
        for (int y=0; y < tempimage.height(); y++)
        {
            for (int x=0; x < tempimage.width(); x++)
            {
                RGB=(QRgb *)tempimage.scanLine(y) +
x;

                int test=qRed(*RGB);
                if(qRed(*RGB)>155)
                {
                    charglyph+=".";
                }
                else
                {
                    charglyph+=".";
                }
            }
            charglyph+="\n";
        }
        // If not created: create toClassifyDir
        QDir dir(mlptoclassifydirectory);
        if(!dir.exists())
        {
            // Create Directory
            dir.mkpath(mlptoclassifydirectory);
        }
        // Save tif as ascii
        QString
curmlptoclassifyglyph=mlptoclassifydirectory;
        curmlptoclassifyglyph+="/";
        curmlptoclassifyglyph+=curobjectid.value();
        curmlptoclassifyglyph+="char";
        QFile
file_mlptoclassifyfile(curmlptoclassifyglyph);

        file_mlptoclassifyfile.open(QIODevice::WriteOnly);
        QTextStream
out_file_mlptoclassifyfile(&file_mlptoclassifyfile);
        out_file_mlptoclassifyfile << charglyph;
        file_mlptoclassifyfile.close();
        // ***
    }
    node=node.nextSibling();
}
}

```

## Appendix II – Inhalt der DVD

<b>/ Beispielmaterial</b>	Vier Ordner mit Digitalisaten aus der vdlb: enne19, enne24, gbii+c627+a und gbiv360.
<b>/ doc</b>	Die Magisterarbeit im PDF-Format.
<b>/ ext</b>	Lässt sich die Applikation nicht starten, müssen womöglich benötigte Microsoft Visual C++ 2005 SP1 Bibliotheken über das beigelegte Paket vc_redist_x86.exe installiert werden.
<b>/ TED</b>	
<b>/ bin_win32</b>	Ausführbare Windows-Applikation (x32); getestet unter Windows XP SP2 & SP3, Windows Vista Business SP1.
<b>/ bin_ubuntu32</b>	„TED“, getestet unter Ubuntu Linux 8.04.1 (32-Bit)
<b>/ bin_ubuntu64</b>	„TED“, getestet unter Ubuntu Linux 8.04.1 (64-Bit)
<b>/ src</b>	„TED“-Quellcode
<b>/ src_qtproject</b>	Quellcode mit Projektdatei TED.pro.
<b>/ src_vsstudio_2005</b>	Projektdateien einschließlich Quellcode für die IDE Microsoft Visual Studio 2005.

## Appendix III: Abbildungsverzeichnis

<b>Abbildung 1:</b> Bertholdus: Horologium devotionis. Zeitglöcklein des Lebens und Leidens Christi, Basel: [Johann Amerbach], 1492. ....	10
<b>Abbildung 2:</b> Beispiel eines stark beschädigten Druckes (Löcher, Druck verblasst, Verschmutzung, etc.): Alexander de Villa Dei: Doctrinale (Pars I) (Comm: Johannes Synthen), Deventer: Jacobus de Breda, [gedruckt zwischen 31 Oktober 1489 und 20 Dezember 1491]. ....	13
<b>Abbildung 3:</b> mehrere Abbraviaturen in einem Wort. ....	13
<b>Abbildung 4:</b> Beispiel einer Ligatur. Hier sind s und t verschmolzen zu einer Drucktype. ....	13
<b>Abbildung 5:</b> Johannes de Verdena: Sermones "Dormi secure" de tempore (ISTC Nr.: ij00454000). Digitalisat vor (links) und nach automatischem Histogrammausgleich (rechts) und zugehörige Histogramme. ....	22
<b>Abbildung 6:</b> Ergebnis des $k$ Fill Filters mit einem Arbeitsfenster von $k=3$ . Links: Originalbild; rechts: nach Anwendung des $k$ Filters. ....	36
<b>Abbildung 7:</b> Originaldigitalisat (links) mit manuell lokalisierten Bounding Boxes (rot gekennzeichnete Bereiche) und manuell um $5^\circ$ gegen den Uhrzeigersinn gedrehtes Quellenmaterial mit umschließenden Rechtecken (rechts). ....	37
<b>Abbildung 8:</b> Bestimmung der Gradientenstärke des verarbeiteten Pixels (gepunktet) aus den Nachbarn (gestreift). ....	42
<b>Abbildung 9:</b> Glyphe (links) und auf Kantenpixel reduzierte Glyphe (rechts). ....	42
<b>Abbildung 10:</b> Neuronenmodell nach McCulloch und Pitts. Hierbei bezeichnen $I_1$ bis $I_n$ die Eingänge des Neurons, $O$ den Ausgang, $BIAS$ den Schwellwert, $W_1$ bis $W_n$ die Gewichte, $\Sigma$ die Eingabefunktion und $f$ die Aktivierungsfunktion. ....	47
<b>Abbildung 11:</b> Selbstorganisierende Karte, bestehend aus Eingabeschicht (unten) und Wettbewerbsschicht (oben). ....	49
<b>Abbildung 12:</b> Feuerndes Neuron (dunkelrot) und seine Wirkung auf unmittelbar umgebende Neuronen (grün und blau) in der Kohonenschicht. ....	50
<b>Abbildung 13:</b> Lernalgorithmus der Selbstorganisierenden Karte. Quelle: Speckmann, Heike: Dem Denken abgeschaut – Neuronale Netze im praktischen Einsatz. Braunschweig / Wiesbaden: Vieweg Verlag, 1996; S. 18. ....	51
<b>Abbildung 14:</b> Beispielhafte Verteilung der Eingabemuster im Raum der Kohonenkarte: Vor dem Training (Links) und nach dem Training (rechts). ....	54
<b>Abbildung 15:</b> Mehrschichtiges Netzwerk, bestehend aus Eingabeschicht, verborgener Schicht und Ausgabeschicht. Quelle: Russell, Stuart; Norvig, Peter: Künstliche Intelligenz – Ein moderner Ansatz. München: Pearson Studium, 2004; S. 905. ....	55

---

<b>Abbildung 16:</b> Konvertierung des 15x15 großen Pixelmusters in ASCII-Code. ....	57
<b>Abbildung 17:</b> Bearbeitetes Digitalisat (oben) mit zugeordneten, d.h. grün markierten Glyphen und die entsprechende Ausgabe über die XHTML Datei mit absolut positionierten Glyphen (unten). ....	61
<b>Abbildung 18:</b> Funktionseinheiten von „TED“ .....	62
<b>Abbildung 19:</b> Dialog zur Klassifizierung von Glyphen in „TED“ .....	64
<b>Abbildung 20:</b> Das Arbeitsfenster von „TED“ mit zugeordneten (grün), nicht zugeordneten (rot) und als falsch klassifiziert erkannten Glyphen (blau). ....	65



**Erklärung**

Köln, den 29. August 2008

Hiermit versichere ich, dass ich diese Magisterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen, Karten und Abbildungen.