Freie Universität Berlin
Institut für Informatik

Diplomarbeit

Schnipseljagd
Maschinelle Verarbeitung von Puzzeln

vorgelegt von Zaidoun Samman
April 2011

Betreuer:
Prof. Dr. Raúl Rojas und
Prof. Dr. Marco Block-Berlitz
Arbeitsgruppe Künstliche Intelligenz
Institut für Mathematik und Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin
Deutschland

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Diplomarbeit mit dem Titel „*Schnipseljagd - Maschinelle Verarbeitung von Puzzeln*“ von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben. Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Ferner erkläre ich, dass die vorliegende Arbeit in keinem anderen Studiengang und an keiner anderen Stelle als Prüfungsleistung verwendet wurde.

Berlin, im April 2011

Zaidoun Samman

Danksagungen

Diese Arbeit ist Teil meines Studiums der Informatik an der Freien Universität in Berlin. Mein erster Dank geht an Herrn Prof. Dr. Raúl Rojas, der mir dieses Thema als Diplomarbeit bewilligt hat. Als Leiter der Arbeitsgruppe Künstliche Intelligenz hat er mich mit seinen Vorträgen und Vorlesungen gefördert, mein Wissen in vielen Bereichen zu erweitern und mir wissenschaftliche Herangehensweisen anzueignen

Mein Dank gilt ebenfalls meinem Betreuer Prof. Dr. Marco Block-Berlitz, der mich weitestgehend unterstützt und immer wieder auf neue Ideen gebracht hat. Seine vorbildliche und kontinuierliche Unterstützung hat wesentlich zum Gelingen dieser Diplomarbeit beigetragen.

Zu guter Letzt möchte ich mich bei meiner Familie für Ihre volle Unterstützung bedanken.

Inhaltsverzeichnis

1. Kapitel

Einführung

Diese Diplomarbeit ist dem Gebiet der Rekonstruktion von zerstörten Papierdokumenten zuzuordnen. Ziel dieser Arbeit ist es, den Forschern und Entwicklern eine Testumgebung zu diesem Zweck zur Verfügung zu stellen.

In dieser Arbeit werden zwei Ansätze genauer betrachtet. Der erste Ansatz ist ein Algorithmus, der zur Generierung von Dokumenten bedacht ist und diese in verschiedenen Formen und mit willkürlichem Inhalt erzeugen soll. Ein zweiter Algorithmus soll diese Dokumente in mehrere Teile bzw. Dokumentenschnipsel zerstückeln.

Eine Analysephase klassifiziert die Information auf einem Dokumentenschnipsel, sodass die folgende Rekonstruktionsphase eine höhere Matchquote erzielen kann.

Viele in dieser Arbeit entwickelte Ansätze basieren auf Java und ImageJ und sind zu dem Bereich der Digitalen Bildverarbeitung zu zählen.
Zur Veranschaulichung und Darstellung von Ergebnissen wurde ein GUI Framework basierend auf Java Swing entwickelt.

1.1 Überblick

Die Rekonstruktion von zerstörten Dokumenten ist von großer Bedeutung auf dem Gebiet der Forensik und investigativen Wissenschaft. Folgendes Beispiel zeigt die Relevanz:
In den letzten Tagen der DDR hat die Stasi versucht, so viele Akten wie möglich zu vernichten. Es wurden etwa 15000 mit Akten gefüllte Säcke gefunden, die von Hand zerrissen wurden. Ein Blatt Papier wurde typischerweise in 20 bis 30 Schnipsel zerlegt. Gelegentlich sind es im Minimum nur etwa 6 Schnipsel pro Blatt. Den Rekord bildet (bis jetzt) ein Blatt Papier, das aus 98 Schnipseln bestand (1).

Aus Gründen der Geschichtsforschung sowie der Gerechtigkeit gegenüber den Betroffenen und Opfer ist es von besonderem Interesse, diese Akten wieder zu rekonstruieren. Zurzeit puzzelt ein Team mit ca. 35 Leuten die Schnipsel wieder zusammen. Ein Mensch kann etwa 10 Seiten pro Tag wieder zusammenfügen. Es wird davon ausgegangen, dass diese auf dieser Weise „von Hand“ etwa 375 Jahre beschäftigt wären (1).
Die Zerstörung von Dokumenten kann neben dem händischen Zerreißen auch durch den Gebrauch eines Schredders erfolgen. Wobei, je nach verwendeten Typ des Schredders, das Papier entweder in dünne Streifen oder sehr kleine Rechtecke zerschnitten wird. Abhängig von der Größe und Anzahl der Schnipsel kann die Wiederherstellung der originalen Dokumente sehr zeitaufwendig sein, womit dies für Menschen beinahe unmöglich ist. Aus diesem Grund werden automatische Rekonstruierungsmethoden benötigt. Vor der rechnerbasierten Rekonstruktion müssen die Schnipsel beidseitig digitalisiert werden. Dabei werden sie mit einem Scanner erfasst und entsprechend gespeichert.
Die gespeicherten und gescannten Schnipsel dienen als Grundlage für die virtuelle Rekonstruktion, die mit Hilfe von Mustererkennungs- und Bildverarbeitungsprozessen durchgeführt werden kann. Dazu werden Merkmale wie Papierfarbe, Schriftfarbe, Schriftart, etc. extrahiert. Um besonders robust zu sein und zusammengehörige Schnipsel sicher zu erkennen, müssen die Algorithmen der virtuellen Rekonstruktion flächen- und konturbasiert arbeiten, das heißt sie berücksichtigen sowohl den Inhalt auf der Schnipseloberfläche, als auch die Form oder die Kontur der Schnipsel. Der in dieser Arbeit vorgestellte Algorithmus folgt dieser Voraussetzung.

1.2 Motivation

Die Arbeiten in dem Gebiet der Rekonstruktion von zerstörten Papierdokumenten und das Testen von dazu entwickelten Algorithmen erfordert eine umfangreiche Datenbank von Schnipseln und Puzzelteilen. Das Beschaffen geeigneter Trainings- und Testmengen von Puzzeldokumenten – also von vielen Dokumenten mit vielen Objekten - ist nicht ohne Weiteres möglich, denn oftmals sind solche Dokumente nicht für alle zugänglich und können für die Öffentlichkeit ungeeignete Informationen enthalten. Und auch wenn solche Dokumente zur Verfügung stehen, ist das Erstellen solcher Datenbanken sowohl zeitaufwendig als auch mit hohen Kosten verbunden. Dokumente müssen mühselig digitalisiert werden und können private personenbezogene Daten enthalten. Also ist es von Vorteil, solche Datenbanken automatisiert generieren zu können.
Im Zuge dieser Diplomarbeit werden zwei Algorithmen, die jeweils in zwei verschiedene Vorgänge aufgeteilt sind, für die Lösung dieses Problems vorgestellt. Der erste Algorithmus „Automatische Generierung von Dokumenten “ ermöglicht es dem Anwender mit einem Mausklick tausend Dokumente in völlig verschiedenen Maßen und mit willkürlichem Inhalt zu erstellen. Die generierten Dokumente sehen einer beliebigen Vorlage wie Briefen oder Rechnungen ähnlich. Der zweite Algorithmus „zufällige Zerlegung von Dokumenten“ ist für das Zerschnipseln der vom ersten Algorithmus erstellten Dokumente zuständig. Dabei kann zwischen Scheren, Handzerreißen oder Schreddern ausgewählt werden. Das Ergebnis ist eine Datenbank mit beliebig vielen Schnipseln. Diese sind in Größe und Form unterschiedlich und

haben realitätsnahen Inhalt. Die so gefüllten Datenbanken kann man zum Testen verschiedener Rekonstruktionsalgorithmen verwenden.
Um möglichst verlässliche Ergebnisse bei den Rekonstruktionen von originalen Dokumenten zu erhalten, ist es notwendig sowohl flächen- als auch konturbasiert zu arbeiten. Man stelle sich die 15000 mit Akten gefüllten Säcke oder ein ganz einfaches Beispiel vor: einen Stapel aus fünf Dokumenten übereinander Ecke an Ecke per Hand zerrissen. In diesem Fall würde ein allein Konturbasierter Algorithmus zu falschen Ergebnissen führen, weil je fünf Schnipsel (nämlich jeweils die übereinander liegenden) denselben Konturverlauf besitzen.
Daher beschäftigt sich ein Kapitel dieser Diplomarbeit mit einem Lösungsansatz, bei dem der Inhalt der Schnipsel betrachtet wird und diese mit Hilfe von Mustererkennung unterschiedlichen Klassen zugeordnet werden. Insgesamt ist diese Diplomarbeit in drei Hauptteile gegliedert: die automatische Generierung von Dokumenten, die zufällige Zerlegung von Dokumenten und die Vorkleben Prozess. Den Gesamtprozess des Generierens, Zerlegens und Vorklebens nennen wir der Einfachheit halber GZV.

2. Kapitel

Theorie und verwandte Arbeiten

In diesem Kapitel sollen einige Themen und verwandte Arbeiten angesprochen werden, die in Zusammenhang mit dieser Diplomarbeit stehen und für ein besseres Verständnis hilfreich sind. Im Folgenden wird das MVC (Model-View-Controller) Konzept detailliert vorgestellt. Zudem werden die theoretischen Grundlagen und Techniken erläutert, welche die Basis für die GZV Vorgänge stellen.

2.1 Theorie

2.1.1 Das Model-View-Controller Konzept

Das Model-View-Controller Konzept ist ein Architekturmuster zur Strukturierung von Software-Entwicklung in den drei Einheiten Datenmodell (model), Präsentation (view) und Programmsteuerung (controller) (2).
Ziel des Musters ist ein flexibler Programmentwurf, der eine spätere Änderung oder Erweiterung erleichtert und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglicht. In der Softwarewelt existiert schon lange die Trennung von Eingabe, Verarbeitung und Ausgabe. Das MVC-Modell überträgt diese Trennung auf GUI-basierte Systeme. Dabei entspricht der Controller der Eingabe, das Model der Verarbeitung und der View der Ausgabe. Die drei Komponenten werden dabei separat behandelt und implementiert. Das MVC-Modell zeichnet sich dadurch aus, dass Änderungen im Modell zu den entsprechenden Änderungen in den Views führen. Die drei Komponenten hängen je nach Realisierung unterschiedlich stark voneinander ab:
Modell (model): Das Modell enthält die darzustellenden Daten und gegebenenfalls (abhängig von der Implementierung des MVC-Patterns) auch die Geschäftslogik. Es ist von Präsentation und Steuerung unabhängig. Die Bekanntgabe von Änderungen an relevanten

Daten im Modell geschieht nach dem Entwurfsmuster „Beobachter“. Das Modell ist das zu beobachtende Subjekt, auch Publisher, also „Veröffentlicher“, genannt. Dieses kann mit beliebig vielen Views verbunden sein.

Präsentation (view): Die Präsentation übernimmt die grafische Darstellung der Daten. Es stellt eine mögliche visuelle Abbildung des Modells dar. Ein View-Objekt ist dabei immer mit genau einem Modell verbunden. Für den Fall, dass sich dieses ändert, erstellt das View-Objekt automatisch eine neue passende Darstellung des Modells und stellt diese grafisch dar. Im Regelfall wird die Präsentation über Änderungen von Daten im Modell mithilfe des Entwurfsmusters „Beobachter“ unterrichtet und kann daraufhin die aktualisierten Daten abrufen. Die Präsentation verwendet das Entwurfsmuster „Kompositum“.

Steuerung (controller): Der Controller definiert wie ein Benutzer mit der Applikation interagieren kann. Er nimmt Eingaben vom Benutzer entgegen und bildet diese auf Funktionen des Modells oder des Views ab. Damit dies funktionieren kann, muss der Controller natürlich Zugriff sowohl auf das Modell als auch auf den Views haben.
Es ist nicht die Aufgabe der Steuerung Daten zu manipulieren. Die Steuerung entscheidet aufgrund der Benutzeraktion in der Präsentation, welche Daten im Modell geändert werden müssen. Diese enthält weiterhin Mechanismen, um die Benutzerinteraktionen der Präsentation einzuschränken. Präsentation und Steuerung verwenden zusammen das Entwurfsmuster „Strategie“, wobei die Steuerung der Strategie entspricht. Die Steuerung kann in manchen Implementierungen ebenfalls zu einem „Beobachter“ des Modells werden, um bei Änderungen der Daten den View direkt zu manipulieren.

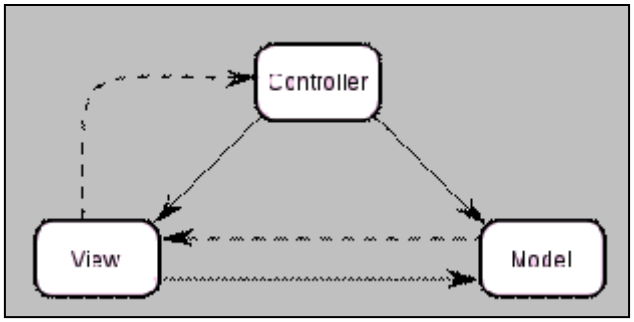


Abbildung 1: Das MVC Konzept, durchgezogene Linie symbolisiert eine direkte Assoziation, die gestrichelte eine indirekte Assoziation

Das MVC Modell wird als Architekturmuster in dieser Arbeit für das „Schnipsel-Jagd“ Framework benutzt, worauf in Kapital 4 näher eingegangen wird.

2.1.2 Distanz und Ähnlichkeitsmaße

Bei der Klassifizierung von Objekten mittels einer Clusteranalyse kommt dem Konzept der Ähnlichkeit eine bedeutsame Rolle zu.
Gegeben ist eine n-elementige Objektmenge *M*, wobei jedes Objekt durch *p* Merkmale repräsentiert wird. Weiterhin ist zu bestimmen, wie ähnlich sich diese Objekte sind, um diese

Menge in homogene Gruppen zu zerlegen. Je größer die Distanz zwischen den Merkmalswerten zweier Objekten in einem p-dimensionalen reellen Zahlenraum ist, umso unähnlicher sind die durch sie repräsentierten Objekte.
Hierbei wird angenommen, dass $i, j\ und\ h$ Objekte aus einer endlichen Objektmenge $I = \{1..N\}$ sind.
Ein Distanzmaß ist dann als Funktion $d: I \times I \rightarrow R$ wenn folgendes gilt:

- Identische Objekte: $d(i,j) = 0\ falls\ i = j$
- Symmetrie: $d(i,j) = d(j,i) \geq 0$
- Dreiecksungleichung: $d(i,j) \leq d(i,h) + d(h,j)$

Die erste Eigenschaft gibt wieder, dass identische Objekte eine Distanz von null haben. Nach der zweiten Eigenschaft ist die Distanz zwischen zwei beliebigen Objekten größer oder gleich Null. Die Dreiecksgleichung besagt, dass die direkte Distanz zwischen zwei Objekten nicht größer sein darf als die über den Umweg über ein drittes Objekt gemessene Distanz.
Die Funktionswerte $d(i,j)$ lassen sich zu einer symmetrischen $(N \times N)$ Matrix anordnen. Diese Matrix heißt Distanzmatrix. Zwei Distanzmaße wurden in dieser Arbeit eingesetzt.

Die Mahalanobis Distanz

Die Mahalanobis-Distanz (nach Prasanta Chandra Mahalanobis) ist ein Distanzmaß zwischen Punkten in einem mehrdimensionalen Vektorraum. Die Mahalanobis-Distanz wird speziell in der Statistik zum Beispiel im Zusammenhang mit multivariaten Verfahren verwendet. Bei multivariaten Verteilungen werden die *m* Koordinaten eines Punktes als m-dimensionaler Spaltenvektor dargestellt, welcher als Realisation eines Zufallsvektors *X* mit der Kovarianzmatrix *S* aufgefasst wird. Die Mahalanobis-Distanz ist skaleninvariant und translationsinvariant. (3). Der Abstand zweier so verteilter Vektoren x_i und x_k wird dann bestimmt durch die Mahalanobis-Distanz.

$$d(x_i, x_k) = \sqrt{d(x_i - x_k)^T S^{-1} d(x_i - x_k)}$$

Ist die Kovarianzmatrix die Einheitsmatrix (dies ist genau dann der Fall, wenn die einzelnen Komponenten des Zufallsvektors X paarweise unabhängig sind und jeweils Varianz 1 besitzen), so entspricht die Mahalanobis-Distanz dem euklidischen Abstand.

Die euklidische Distanz

Sind die Vektoren *x* und *y* gegeben durch die Koordinaten $x = (x_{1,} \dots, x_{n,})$ und $y = (y_{1,} \dots, y_{n,})$ so gilt:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

In den folgenden drei Abschnitten werden verwandte Techniken zu den GZV Vorgängen vorgestellt.

2.2 Automatische Generierung von Dokumenten

Bei der automatischen Generierung von Dokumenten werden Dateien erstellt. Diese Dokumente können verschiedener Art sein, wie beispielsweise ein Brief, ein Bild, eine Buchseite oder eine Rechnung. Diese Dateien bilden zusammen eine Datenbank, die als Basis für den folgenden Algorithmus – dem Zerlegen von Dokumenten – benutzt werden kann. Der Inhalt generierter Dokumente ist willkürlich und wird zufällig erstellt. In der Literatur findet man ein ähnliches Konzept, das sich mit der Thematik des Fingerabdrucks beschäftigt.

2.2.1 Das Generieren von synthetischen Fingerabdrücken

Der Fingerabdruck ist ein Abdruck der so genannten Papillarleisten am Endglied eines Fingers (Fingerkuppe bzw. Fingerbeere). Die Papillarleisten bilden Schleifen-, Bogen- oder Wirbelmuster, die in Verbindung mit den Unterbrechungen der Papillarleisten (Minutien) von Finger zu Finger und von Mensch zu Mensch unterschiedlich sind (4).
Als *Minutien* (von lat. Minuzien, „Kleinigkeiten“) werden die Endungen und Verzweigungen der Papillarleisten des menschlichen Fingerabdrucks bezeichnet. Sie sind unveränderlich, weshalb sie für die Authentifizierung von Nutze sind (Abbildung 2).

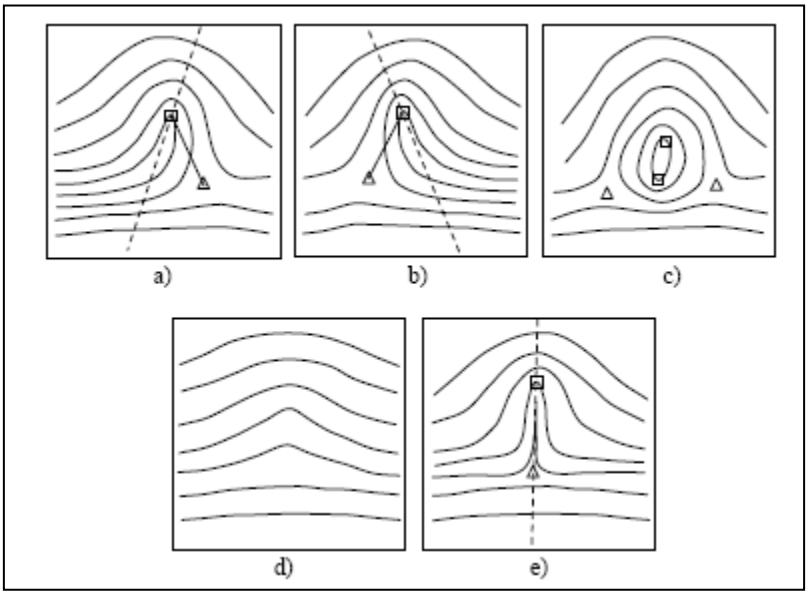


Abbildung 2 aus (5): Fingerabdruckmuster: a) Schlafe links b) Schlafe rechts c) Wirbel d) Bogen e) gewölbter Bogen

Als *Delta* wird ein Papillarlinienbild bezeichnet, das dem griechischen Großbuchstaben Delta ähnelt. Es wird aus zwei auseinander laufenden Papillarlinien oder aus einer sich gabelnden Papillarlinie und einer dritten aus einer anderen Richtung kommenden konvex verlaufenden Papillarlinie gebildet (5). Beispiele einer Delta-Konfiguration werden in der Abbildung 3 gezeigt.

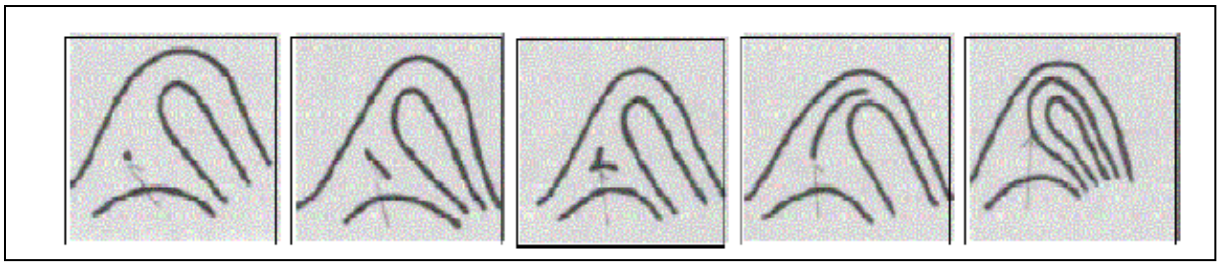


Abbildung 3 aus (5): Delta Konfigurationen

Der Kern eines einzelnen Fingerabdruckes ist aufgrund der großen Variation in der Krümmung der inneren Linien recht schwer definierbar. Deshalb wählt man den Kern als einen spezifischen Punkt, an dem sich das Zentrum des zugehörigen Musters befindet (5). Die Abbildung 4 zeigt einige Beispiele der Kernkonfiguration.

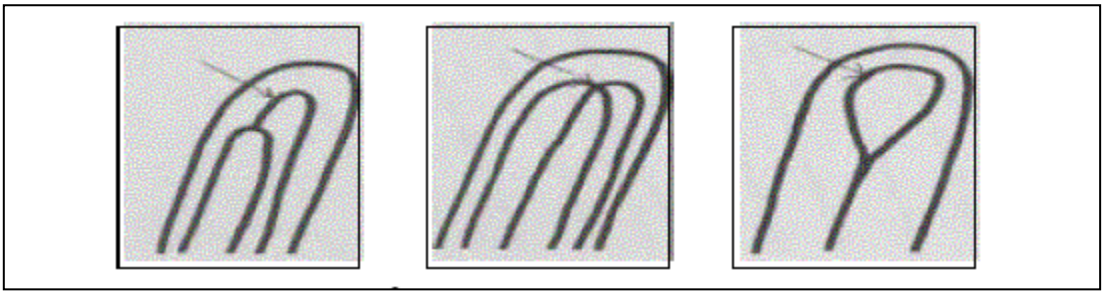


Abbildung 4 aus (5): Kern Konfiguration

Diese Merkmale bilden unter anderem die Einzigartigkeit des menschlichen Fingerabdrucks.
Aus diesem Grund war der Fingerabdruck, eine der ersten biometrischen Eigenschaften, die entdeckt und wissenschaftlich untersucht wurden.
Die Liste der Anwendungen von Fingerabdrücken ist lang. So sind unter anderen die wichtigsten Beweismittel der Kriminalistik die am Tatort gefundenen Fingerabdrücke. Sie werden mit einer Datenbank abgeglichen, um Verdächtige zu identifizieren. Das deutsche Bundeskriminalamt führt eine AFIS-Datenbank mit den Fingerabdrücken von über drei Millionen Personen (6). Ein weiteres Beispiel sind die biometrischen Systeme, wobei der Fingerabdruck ebenfalls zur Identifikation eingesetzt wird. Der Abdruck wird zumeist optisch oder elektrisch (z. B. kapazitiv) gelesen, um berechtigte von unberechtigten Nutzern zu unterscheiden. Fingerabdruckverfahren zur Identifikation von Personen gibt es in manueller Form seit etwa 100 Jahren und werden vorwiegend im Bereich der Strafverfolgung verwendet.
In der IT-gestützten, automatisierten Form ist das digitale Fingerabdruckverfahren ein biometrisches Verfahren mit hoher Erkennungsleistung. Mit Hilfe optischer, kapazitiver (Halbleiter), thermischer oder direkt-optischer Sensoren werden Fingerabdrücke bei automatischer Fingerabdruckerkennung erfasst. Nach der Erfassung des Fingerabdrucks steht dem Verfahren stets ein Graustufenbild des Fingers, der Fingerabdruck, zur Verfügung. Dieses Bild wird durch Schritte, wie etwa Verminderung des Bildrauschens, und Erkennung einzelner Abdruckmerkmale, weiterverarbeitet. Mit dem verbesserten Bild können Fingerabdrücke mit einer sehr hohen Quote korrekt verglichen werden. Durch unterschiedliche Algorithmen zur Fingerabdruckerkennung werden die charakteristischen Kennzeichen aus dem Bild extrahiert und klassifiziert. Der Vergleich dieser Kennzeichen mit

den personenspezifischen Sollwerten zeigt, ob die Abdrücke vom selben Finger und somit von genau einer Person stammen.
Fingerabdruckbasierende Identifikation wird funktionell nach drei grundlegenden Aufgaben aufgeteilt (5):

a) Fingerabdruckabtastung
b) Fingerabdruckklassifizierung
c) Fingerabdruckvergleich

Schritt (b) und (c) zusammen machen die Fingerabdruckerkennung aus. Die Lösungsansätze bzw. die Algorithmen in diesem Gebiet werden ständig sowohl durch akademische als auch industrielle Institutionen weiterentwickelt.
Für Algorithmen zur Fingerabdruckerkennung ist die Größe der Referenzdatenbank von Fingerabdrücken ein limitierender Faktor. Häufig sind die entwickelten Algorithmen stark auf eine bestimmte Datenbank optimiert. Um die Erkennungsrate solcher Algorithmen erhöhen zu können, werden wesentlich größere Datenbanken benötigt. Diese aufzubauen erfordert jedoch einen massiven Finanz-, Zeit-, und Personalaufwand.
Da Fingerabdrücke zur Identifizierung von Personen verwendet werden, besteht ein Datenschutzproblem. Selbst wenn sich eine Behörde oder eine Institution die Arbeit machen würde, eine solche Datenbank aufzubauen, wären internationale Richtlinien zu überwinden, um diese Datensätze veröffentlichen zu können. Um dennoch schnell, effizient und kostengünstig eine solche Datenbank für Testzwecke bestimmter Algorithmen zu erstellen, werden Fingerabdrücke künstlich erstellt (5).
SFinGe ist ein Programm zum Generieren von Fingerabdrücken. Es erstellt automatisch Datenbanken mit einer riesigen Anzahl von Fingerabdrücken, die zum Testen, Optimieren und Vergleichen verschiedener Fingerabdruckerkennungsalgorithmen verwendet werden können. Es existieren zur Erstellung künstlicher Fingerabdrücke im Wesentlichen zwei Hauptgruppen von Ausgangsmodellen: die physischen sowie die statistischen Modelle. Die physischen Modelle stützen sich auf Beobachtungen der Embryogenese. Danach ist das Wachstum der Epidermis durch äußere Einflüsse ungleichmäßig. Der Fingerabdruck entwickelt sich durch Kräfte, die bei ungleichmäßigem Wachstum der Epidermis entstehen.
Die statistischen Modelle zielen darauf ab, unabhängig von den physischen Modellen realistisch aussehende Fingerabdrücke zu erstellen. Solche Modelle basieren auf empirischen Analysen von echten Fingerabdrücken. Aus statistischen Datensätzen werden die Hauptmerkmale von Fingerabdrücken als Parameter an entsprechende Synthesealgorithmen übergeben. Dabei werden die 2D Fourierspektren aufgebaut, aus denen eine Vielzahl von Sinusoiden entsteht. Diese machen letztlich den Fingerabdruck aus. Wie die nachfolgende Abbildung veranschaulicht, nehmen die Fingerabdrücke in Ihrer Komplexität zu, je mehr Startpunkte man hat (5).

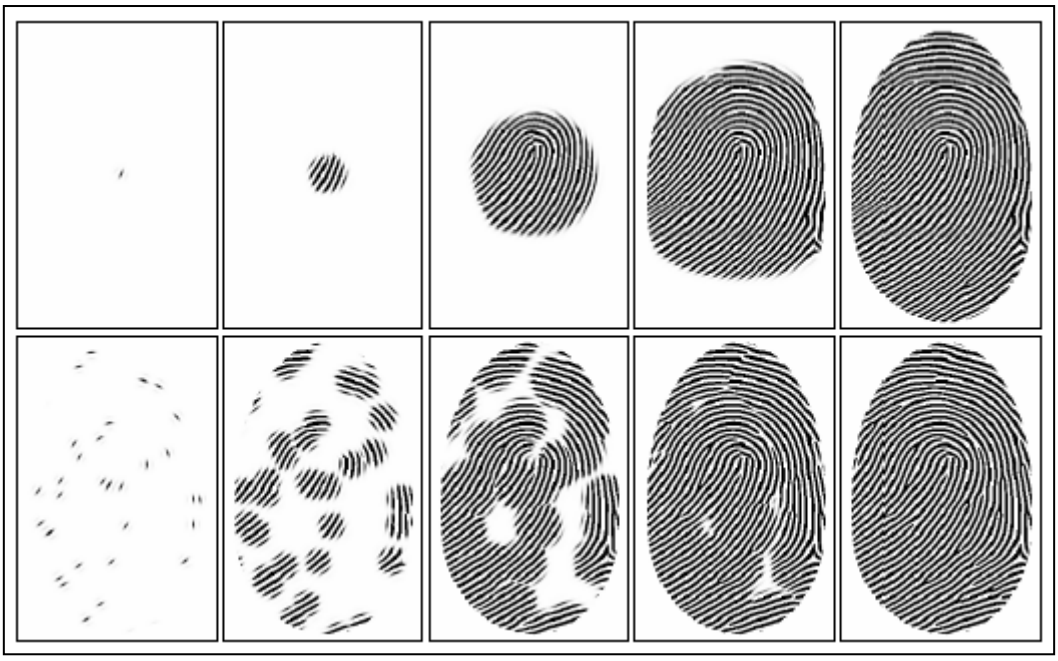


Abbildung 5 aus (5): Zwischenschritte eines Fingerabdrucks: Generierung ausgehend von einem einzelnen zentralen Punkt (obere Reihe) und aus zufällig platzierten Punkten (untere Reihe).Je mehr Startpunkte man hat, desto unregelmäßiger und reicher an Minutien ist das Muster.

Das Hauptziel der SFinGe (5) Methode ist es, beliebig große Datenbanken mit möglichst geringem Aufwand zu erzeugen. Ausgangspunkt dafür ist im ersten Schritt die Erstellung eines sog. „*Masterfingerabdrucks*“ aus einem echten Fingerabdruck. Dieser synthetische „*Masterfingerabdruck*“ ist einmalig und der Ausgangspunkt für die Erstellung multipler synthetischer Fingerabdrücke, die durch explizierte Modifikationen (Verschieben, Rotation, Drehung, Änderung der Hautbeschaffenheit und des Hintergrundes) generiert werden können. Die Generierung eines Fingerabdrucks beansprucht insgesamt neun Schritte; die ersten vier davon generieren den „*Masterfingerabdruck*“ (5):

1) Das Generieren eines Fingerabdruckbereiches: In diesem Schritt wird ein rechteckiger Rahmen definiert, welches dem Umriss des vom Sensor erfassten Fingerabdrucks enthält.
2) Das Generieren eines Orientierungsbildes: Ausgehend von der Position der Merkmale eines Fingerabdrucks (Delta und Wirbel) wird mit Hilfe von mathematischen Modellen (Bsp. komplexe rationale und sinusförmige Funktionen) die Orientierung des Fingerabdrucks berechnet.
3) Das Generieren der Dichteverteilung: Papillarleisten haben unterschiedliche Abstände zwischen einander, je nach dem wo sie sich auf dem Fingerabdruck befinden. Mit heuristischen Methoden wird in diesem Schritt eine Dichteverteilung der Papillarleisten, die einem echten Fingerabdruck ähnlich sieht, generiert.
4) Das Generieren eines Papillarlinienmusters: In einem weißen Bild werden mehrere schwarze Punkte zufällig platziert. Auf dieses Bild wird iterativ der Gaborfilter angewandt, wobei auch die lokalen Werte des Orientierungsbildes und der Dichteverteilung als Eingabe in die Berechnung fließen. Das Ergebnis ist ein Fingerabdruck mit realistischen Papillarlinien und Minutien von verschieden Arten, die an zufälligen Stellen zur Erscheinung kommen.

Um der Realität näherzukommen wird der „*Masterfingerabdruck*“ weiter verarbeitet. Dabei werden mehrere unterschiedliche Abdrücke von demselben Finger generiert. Diese enthalten Störungen, die bei einer echten Fingerabdruckerfassung auftreten können:

5) Definition des Bereiches auf dem Finger, welcher tatsächlich von dem Sensor erfasst wird.
6) Veränderung der Dicke der Papillarlinien, die als Folge von Hautfeuchtigkeit und Druck des Fingers gegen den Sensor auf dem erfassten Bild unterschiedlich dick aussehen können.
7) Hinzufügen von Verzerrungen, die in dem vom Sensor erfassten Bild durch die Hautelastizität verursacht werden.
8) Hinzufügen von Störungen und Koordinatentransformation um verschiedene Faktoren nachzuahmen. Diese können zur Verschlechterung der Bildqualität führen, in dem eine unreine Finger- bzw. Sensoroberfläche benutzt wird. Darüber hinaus ist der Fingerabdruck in der Regel nicht im Bild zentriert und kann eine gewisse Rotation haben.
9) Generieren eines passenden Hintergrundes, um den unrealistischen weißen Hintergrund zu füllen.

Der so generierte „*Masterfingerabdruck*“ und die darauf angewandten Bearbeitungsschritte liefern einen realistischen Fingerabdruck. Die folgende Abbildung 6 gibt einen zusammenfassenden Überblick über die genannten neun Schritte.

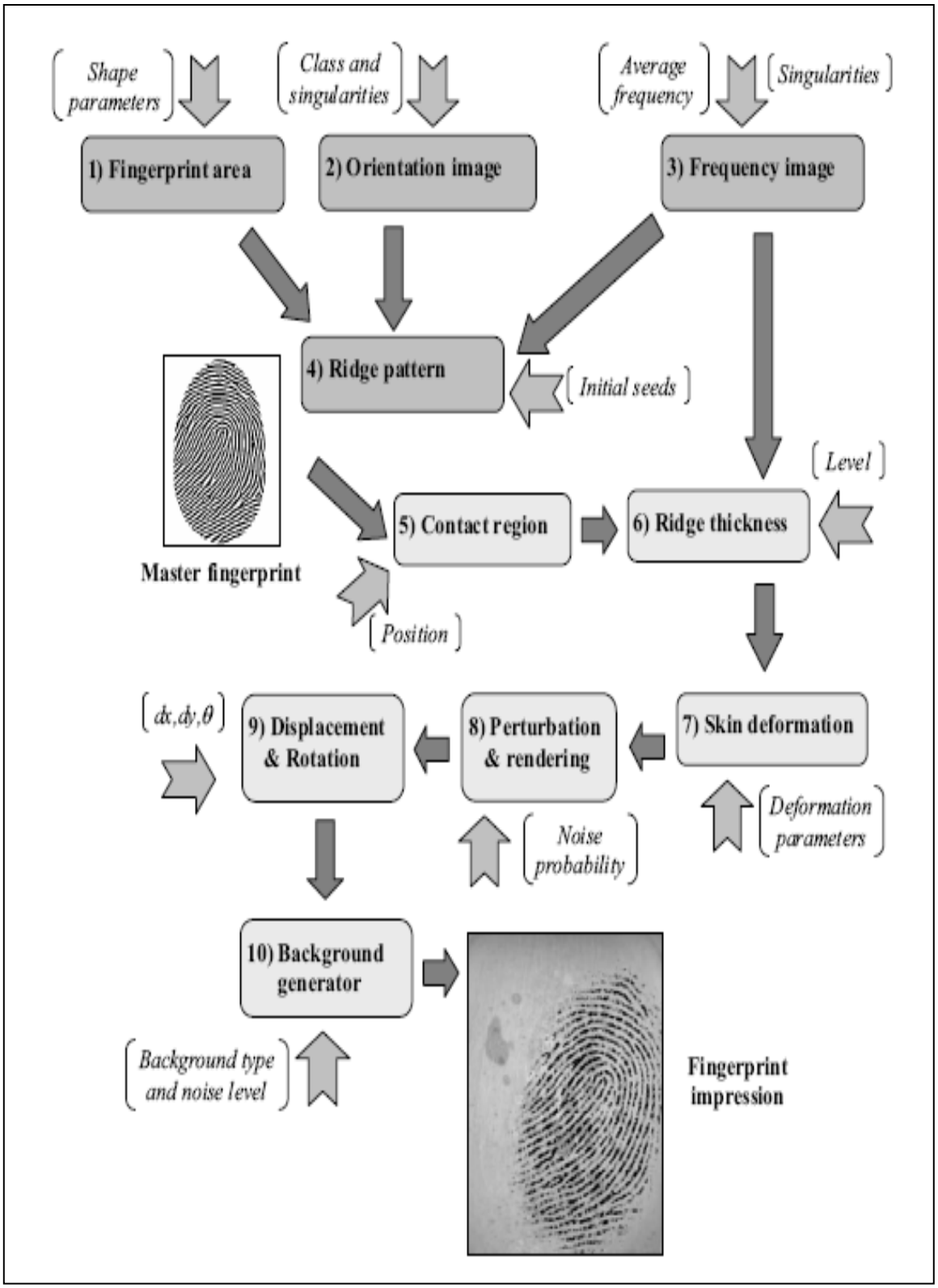


Abbildung 6 aus (5): Das funktionale Schema der synthetischen Fingerabdruckgenerierung SFinGe: Jedes Rechteck stellt einen Generationsschritt dar. In Klammer sind die wichtigsten Input-Parameter, welche jeden Schritt kontrollieren. Die Schritte 1 bis 4 erzeugen ein Master- Fingerabdruck. Schritte 5 bis 10 erzeugen einen möglichst realistischen synthetischen Fingerabdruck von dem Master-Fingerabdruck

2.3 Die zufällige Zerlegung von Dokumenten

Papierschnipsel, die zum Trainieren und Testen von Rekonstruktionsalgorithmen verwendet werden, entstehen entweder durch Schreddern, durch Schnipseln oder durch Scheren. Diese drei Möglichkeiten werden im Folgenden näher betrachtet

Das Schreddern

Ein Schredder, auch Aktenvernichter ist ein Gerät zum Vernichten von Dokumenten. Einfachere Aktenvernichter schneiden das Papier lediglich in schmale Streifen (Streifenschnitt). Bessere Aktenvernichter schneiden zusätzlich horizontale Schnitte (Kreuzschnitt/Partikelschnitt).

In Deutschland wird die Datensicherheit eines Aktenvernichters nach den fünf Sicherheitsstufen der DIN-Norm DIN 32757 bewertet. Die höchste Sicherheitsstufe 5 findet in der Praxis kaum Anwendung, am häufigsten ist die Sicherheitsstufe 2 oder 3 anzutreffen. Vereinzelt bewarben einige Hersteller auch die Sicherheitsstufe 6, welche aber nicht in der DIN 32757 geregelt ist. Im Einzelnen werden die sechs Sicherheitsstufen für Papier wie folgt definiert (DIN 32757-1:1995-01) (7):

Sicherheitsstufe 1 (empfohlen für allgemeines Schriftgut)

- bei Streifenschnitt: max. 12 mm Streifenbreite.
- bei Cross Cut: max. 1000 mm² Partikelfläche.

Sicherheitsstufe 2 (empfohlen für internes, nicht besonders vertrauliches Schriftgut)

- bei Streifenschnitt: max. 6 mm Streifenbreite.
- bei Cross Cut: max. 400 mm² Partikelfläche.

Sicherheitsstufe 3 (empfohlen für vertrauliches Schriftgut)

- bei Streifenschnitt: max. 2 mm Streifenbreite.
- bei Cross Cut: max. 4 mm Breite auf max. 60 mm Partikellänge (240 mm² Partikelfläche).

Sicherheitsstufe 4 (empfohlen für geheimzuhaltendes Schriftgut)

- Cross Cut: max. 2 mm Breite auf max. 15 mm Partikellänge (30 mm² Partikelfläche).

Sicherheitsstufe 5 (für maximale Sicherheitsanforderungen).

- Cross Cut: max. 0,8 mm Breite auf max. 15 mm Partikellänge (12 mm² Partikelfläche).
- zerkleinerte grobe Asche, Suspension, Lösung oder Fasern.

Sicherheitsstufe 6 (ungenormt) (für geheimdienstliche Sicherheitsanforderungen)

- Cross Cut: max. 1,0 mm Breite auf max. 5,0 mm Partikellänge (5 mm² Partikelfläche).
- zerkleinerte feine Asche, Suspension, Lösung oder Fasern.

Nach dem aktuellen Stand der Technik ist die Reproduktion der auf dem Informationsträger wiedergegebenen Informationen bis zur vierten Sicherheitsstufe möglich.

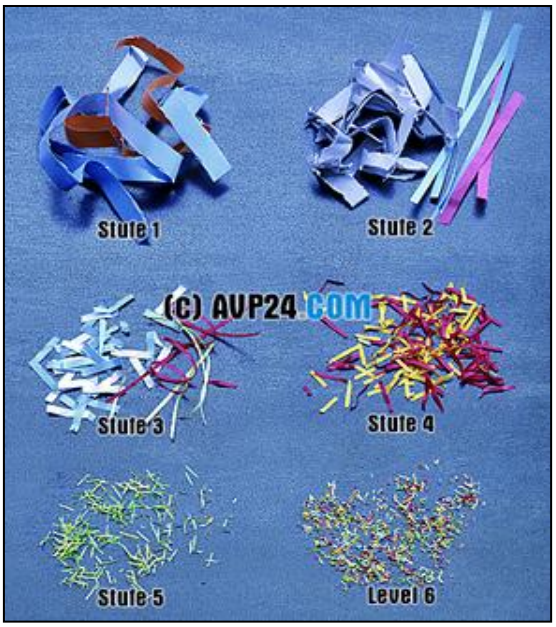


Abbildung 7 aus (7): Ergebnis vom Schreddern nach den verschiedenen Sicherheitsstufen

Das Schnipseln

Das bekannteste Beispiel für diesen Prozess kommt aus der ehemaligen DDR. Um die Geheimnisse der Stasi zu bewahren, wurden zwischen Herbst 1989 und Januar 1990 im ehemaligen Ministerium für Staatssicherheit systematisch Akten vernichtet. Die Menge der Dokumente war so enorm, dass die Reißwölfe ausfielen. Ein großer Teil der Unterlagen musste per Hand zerrissen werden. Geschätzte 45 Millionen DIN-A4 Seiten wurden in je 8 bis 30 Teile zerlegt (1).

Abbildung 8 aus (8): Unsortierte Schnipsel

Das Scheren
Das Scheren ähnelt dem Schnipseln, jedoch fallen die Ränder eines Schnipsels geradlinig aus und haben im Allgemeinen die Form eines Polygons.

Die moderne Computer-Technologie hat den Prozess der Reproduktion erheblich beschleunigt. Die Streifen bzw. Schnipsel werden auf beiden Seiten eingescannt, im Computer gespeichert und dann durch verschiedene Algorithmen wieder zusammengeführt, wobei die gescannten Schnipsel als Test bzw. Trainingsmenge für die Rekonstruktions-algorithmen dienen. Im nächsten Abschnitt wird hierauf näher eingegangen.
Beim Entwickeln von guten Algorithmen zur Rekonstruktion von zerschnipselten Dokumenten treten die gleichen technischen Schwierigkeiten, die auch beim Entwickeln von Algorithmen zum Vergleich von Fingerabdrücken auftreten. Das Aufbauen von Trainings und Testdatensätzen ist ein kostenintensiver Prozess.
Aus diesem Grund stellt das folgende Szenario eine intelligente Lösung dar: sei *In_M* eine Menge bzw. eine Datenbank von automatisch generierten Dokumenten (vgl. Abschnitt 3.1). Dabei ist auch *zerlege(In_M)* eine Methode, die *In_M* als Eingabe benutzt, auf diese Menge einen Zerlegungsprozess (Schreddern, Schnipseln oder Scheren) simuliert und *Out_M* als Ergebnis erzeugt. *Out_M* stellt also eine Menge von zerschnipselten Dokumenten dar, die als Test- bzw. Trainingsmenge für die Rekonstruktionsalgorithmen zur Verfügung steht.
Die herkömmlichen Bildverarbeitungsprogramme – Bsp. Photoshop – bieten die Möglichkeit, anhand ihrer vielen unterschiedlichen Werkzeuge ein Bild in verschiedene Formen zu zerschneiden. Trotzdem ist diese Funktionalität nicht in der Lage das obengenannte Szenario zu erfüllen.

2.4 Rekonstruktion zerrissener Dokumente

Bei diesem Prozess wird von einer Menge von Schnipsel S ausgegangen, die von unterschiedlichen Seiten stammen. Erzeugt wurden die Schnipsel durch händisches bzw. maschinelles Zerreißen von mehreren Papierseiten.
Es gibt viele verwandte Arbeiten, die sich mit diesem Thema beschäftigen, insbesondere wegen seines breiten Spektrums.
Eine davon, die jedoch sehr unterschiedliche Herausforderungen beinhaltet, beschäftigt sich mit dem automatischen Lösen von Puzzles (Jigsaw Puzzles). Der Hauptunterschied liegt darin, dass beim Puzzle beinahe alle Teile eine einheitliche und klare Form besitzen. Aus diesem Grund können ihre geometrischen Daten sehr gut für die Rekonstruktion verwendet werden. Außerdem werden auch Farbinformationen der Puzzleteile verwendet. Denn im Gegensatz zu den meisten Textdokumenten können zum Lösen von Puzzles diese Informationen effektiv beitragen (9).
In der Thematik, die sich mit der Rekonstruktion von handzerrissenen Dokumenten beschäftigt, stellt das Projekt „virtuelle Rekonstruktion“ eines der größten Projekte bisher (1)

Ziel dieses Projektes ist die Zusammensetzung der vorvernichteten Dokumente des Staatssicherheitsdienstes (*Stasi*) der DDR.
Die Struktur des Gesamtablaufs zur automatischen Rekonstruktion gliedert sich in die Aufbereitung der Schnipsel sowie deren Digitalisierung, virtuelle Rekonstruktion sowie Übertragung im Workflow-System und Archivierung (1)

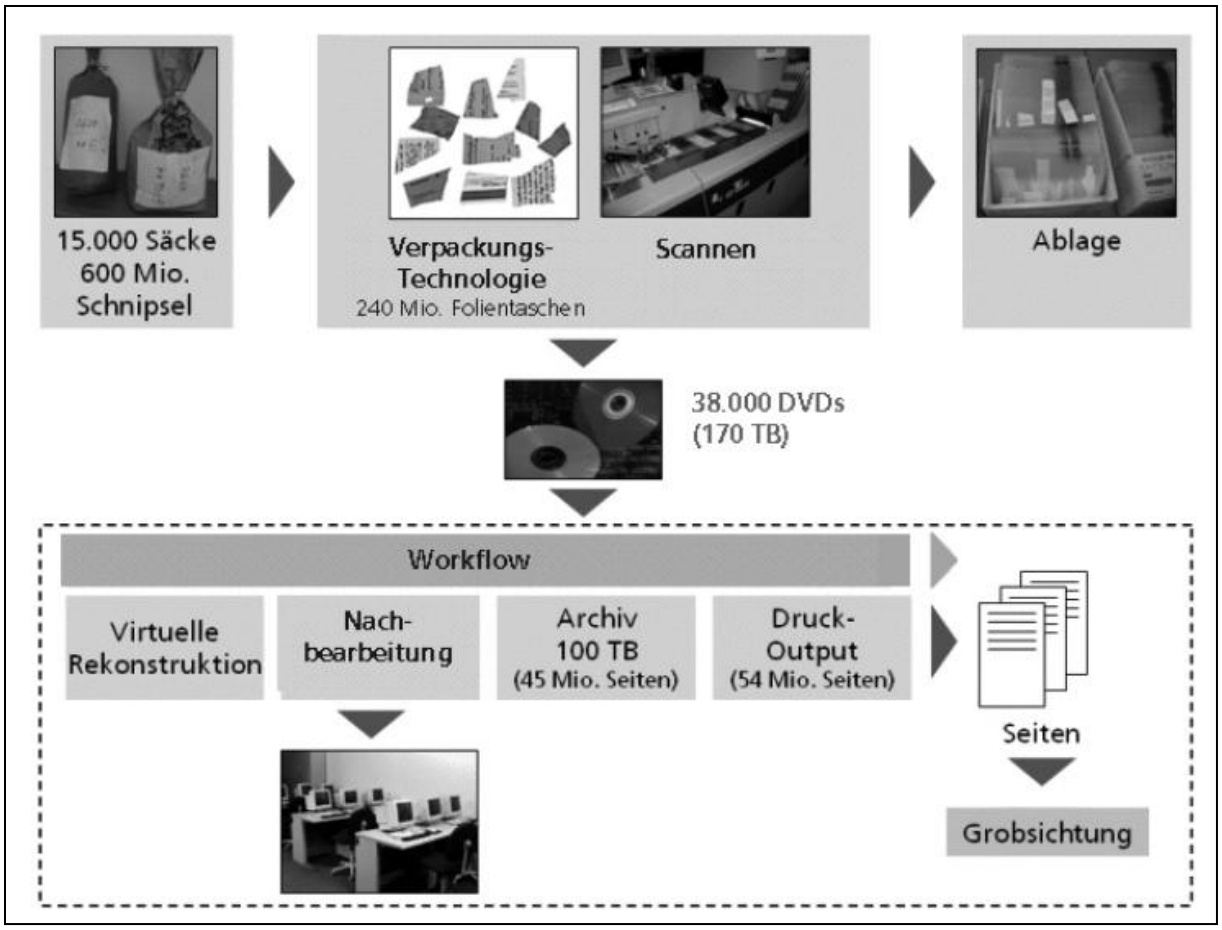


Abbildung 9 aus (1): Übersicht des Gesamtablaufs der Rekonstruktion von zerstörten Dokumenten

Grundlage der virtuellen Rekonstruktion sind die digitalisierten, segmentierten Schnipsel, die dem Workflow in Form von DVDs oder einer Datenbank übergeben werden.

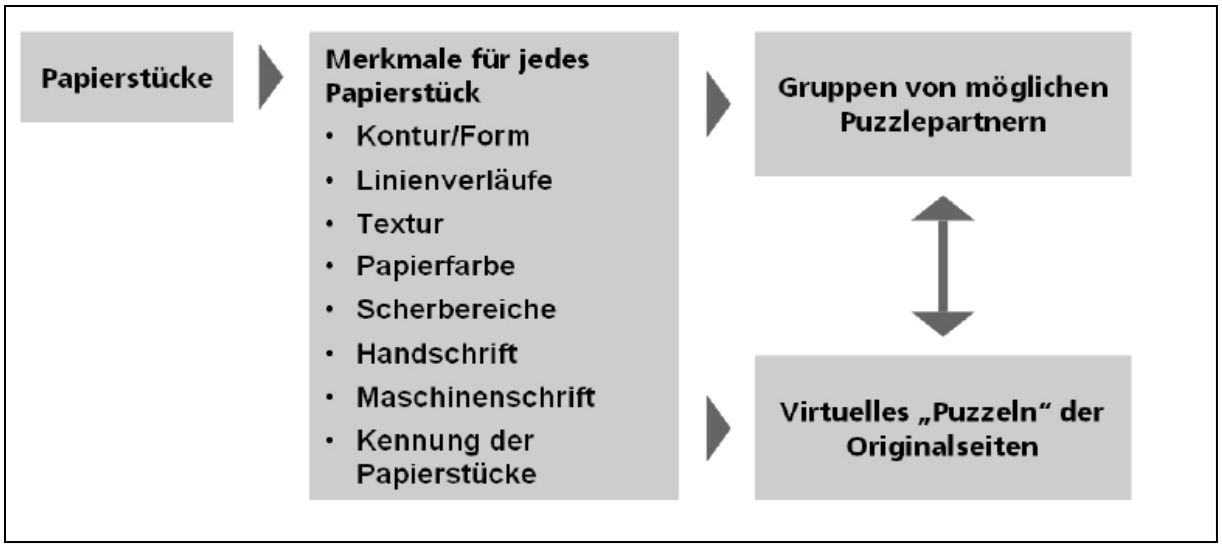


Abbildung 10 aus (1): Bestandteile des Prozesses der virtuellen Rekonstruktion

Die virtuelle Rekonstruktion kann in die folgenden drei Teilbereiche zerlegt werden (1), deren Zusammenhang in der Abbildung 10 dargestellt ist.

i. **Merkmalsextraktion**: Für das Zusammensetzen eines Puzzles wird von einer Vielzahl von Merkmalen Gebrauch gemacht, anhand derer entschieden wird, ob zwei Teile zueinander passen oder nicht. Dies sind zum einen Merkmale, die die äußere Form der Puzzlestücke beschreiben und zum anderen innere Merkmale wie Farbe oder Textur sowie kontextbezogene Merkmale wie Detailabbildungen auf den Puzzlestücken. Analog zum menschlichen Vorgehen werden für Vorder- und Rückseiten der Schnipsel verschiedene beschreibende Mustermerkmale herangezogen und in einer Datenbank abgelegt. Insbesondere sind dies Papierfarbe, mittlere Schnipselgröße, Form der Risslinie, Ausrichtung der Schrift, Ausrichtung der Risslinie, Texturmerkmale, geometrische Merkmale des Schnipsels, Schriftart auf dem Schnipsel, Schriftgröße, Zeilenabstand, Schnipsel einseitig oder zweiseitig bedruckt und Bild oder grafisches Element auf Schnipsel (vgl. Abbildung 11).

ii. **Suchraumreduktion:** Aufgrund der sehr großen Datenmenge, die in diesem Projekt zu beherrschen ist, ist es sehr aufwendig, alle genannten Merkmale in den weiteren Berechnungsschritten in Betracht zu ziehen. Die Kombinationsmöglichkeiten der zu überprüfenden Teile wären zu umfangreich. Das zu rekonstruierende Material ist nicht zufällig über alle Säcke verteilt. Die Einzelteile eines Dokumentes befinden sich sehr wahrscheinlich im selben Sack. Darüber hinaus haben sich durch das manuelle Befüllen der Säcke Schichten ergeben, die es wiederum wahrscheinlich machen, dass Rekonstruktionspartner in den jeweiligen Säcken relativ dicht beieinander liegen. Diese Schichtung wird bei der Digitalisierung berücksichtigt, sodass die Schnipselgruppen den Eingang in den aktuellen Suchraum bilden. Dieser wird nun mit Hilfe der oben beschriebenen Merkmale weiter untergliedert. So stellen Schnipsel mit ähnlichen Merkmalen aussichtsreiche Kandidaten für eine Rekonstruktion dar.

iii. **Virtuelles Puzzeln:** In dem reduzierten Suchraum werden Schnipsel entlang ihrer Konturen auf Merkmalsübereinstimmung hin verglichen. Werden passende Schnipsel gefunden, so werden diese zu einem größeren Teil des Dokumentes zusammengefasst, erneut die Merkmale des zusammengesetzten Stücks berechnet und dieses als neuer Schnipsel in der weiteren Rekonstruktion berücksichtigt. Somit kann dieses Stück so

lange weiter mit anderen Schnipseln und Teilergebnissen verglichen und gegebenenfalls zusammengefasst werden, bis ein vollständiges Dokument entsteht. Wurden alle möglichen Rekonstruktionsschritte auf der durch die Suchraumreduktion ermittelten Gruppe durchgeführt, wird die Gesamtmenge der Schnipsel durch die erzielten Ergebnisse aktualisiert.

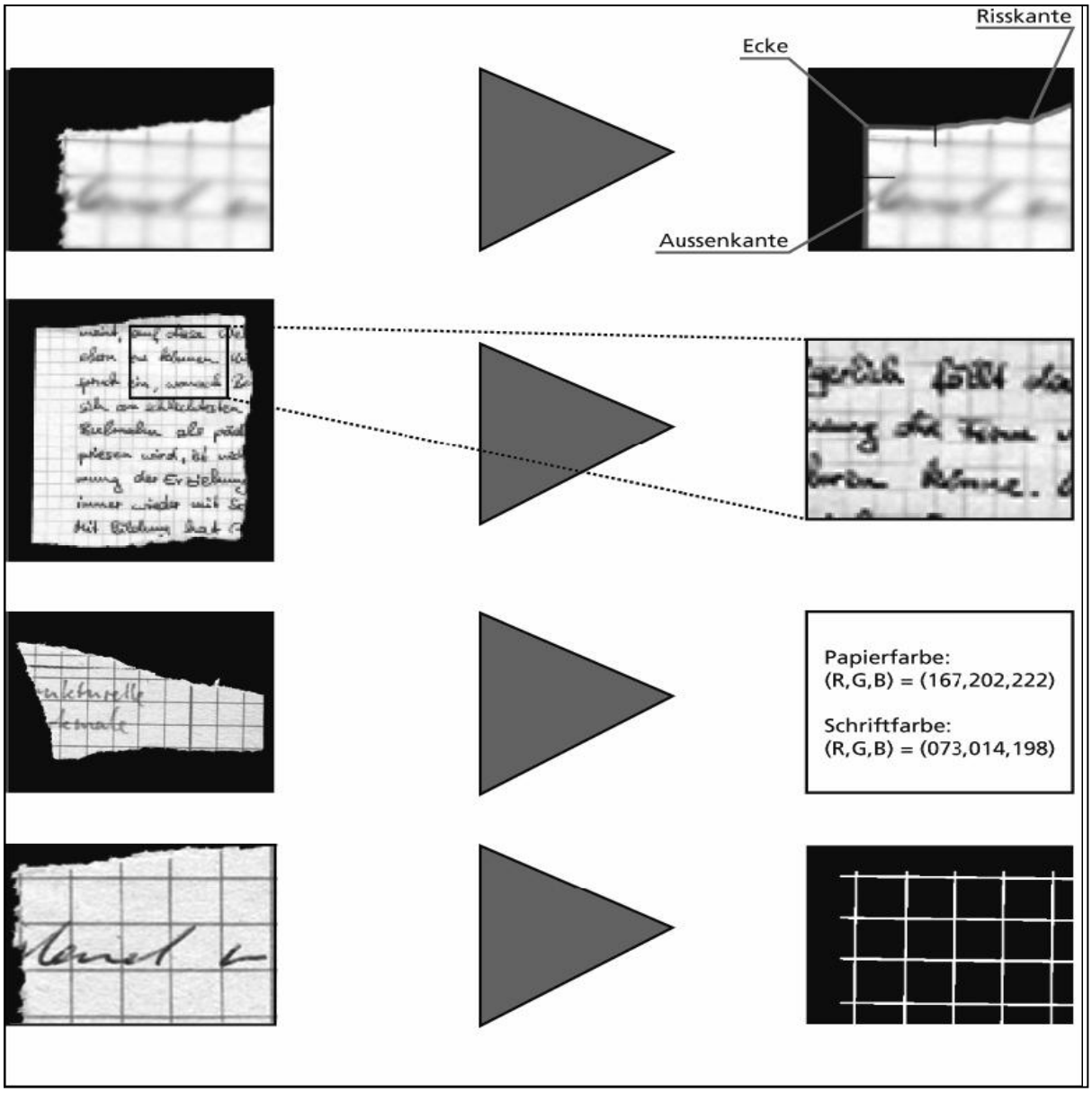


Abbildung 11 aus (1): Kontur-, Schrift-, Farb- und Linienmerkmale eines Schnipsels

Es ist durchaus möglich, dass die automatische Rekonstruktion für manche Schnipsel keine Entscheidung mehr trifft, und somit keinen passenden Schnipsel-Partner findet, so wird das erzielte Teilergebnis an einen manuellen Nachbearbeitungsplatz gegeben. Hier unterstützt das System den menschlichen Bearbeiter durch eine Auswahl der wahrscheinlichsten Rekonstruktionspartner. Mittels dieser Auswahlliste kann ein Mensch interaktiv am Rechner die Schnipsel und Teilergebnisse vervollständigen.

3. Kapitel

Das Schnipseljagd Projekt

In diesem Kapitel werden die Algorithmen und die Techniken der GZV Prozesse dargestellt, die im Rahmen dieser Diplomarbeit entwickelt wurden. Diese bilden außerdem den Kern des Schnipseljagd-Frameworks.
Die Problematik der einzelnen GZV Prozesse wird in jedem der folgenden Abschnitte dieses Kapitels beschrieben. Anschließend werden die entwickelten Algorithmen und Lösungsansätze vorgestellt sowie Experimente zum Evaluieren erläutert.
In den ersten beiden Abschnitten (3.1 und 3.2) werden das Generieren und das Zerschnipseln von Dokumenten behandelt. Die zentrale Idee dabei ist das Aufbauen von großen Referenzdatenbanken, die es ermöglichen sollen, die Leistung verschiedener Algorithmen zu vergleichen und zu optimieren. Ist eine große Referenzdatenbank vorhanden, so sollte dies mit geringem Finanz-, Zeit-, und Personalaufwand realisierbar sein.
Somit ist dieselbe Herausforderung zu bewältigen, die bei Algorithmen für Fingerabdruckerkennung auftritt und bereits unter Abschnitt 2.2.1 erörtert wird.

3.1 Der Prozess des (G)enerierens

Die Forschungsbereiche der Analyse und Klassifikation von Dokumenten beschäftigen sich mit der Erkennung von Strukturen in Dokumenten wie Überschriften, Absätzen und Tabellen, welche der menschliche Betrachter sofort und ohne Aufwand erkennen kann. Außerdem spielen die Erkennung des thematischen Inhaltes und die Textanalyse sowie die Zonen Klassifikation in Dokumenten wie Texte, Bilder, mathematische Formel etc. eine sehr große Rolle in diesen Forschungsbereichen.
In diesem Abschnitt wird eine Methode vorgestellt, deren Aufgabe darin besteht, eine beliebig große Referenzdatenbank aufzubauen, welche zufällig generierte Dokumente enthält. Das Format und der Inhalt der generierten Dokumente sind beliebig. Diese Referenzdatenbank hat

zwei wichtige Funktionen. Zum einen stellt sie eine geeignete Test- und Trainingsmenge für die Algorithmen zur Analyse und Klassifikation von Dokumenten (10) (11) (12) zur Verfügung, zum anderen wird sie als Input für den danach folgenden Prozess und zwar das Zerschnipseln von Dokumenten verwendet.

3.1.1 Der Dokumenten-Generator

Durch die visuelle Betrachtung von 50 der alltäglichen Briefe fiel der Entschluss, diese in vier verschiedenen Kategorien zu unterteilen. Dafür wurde ein Model konstruiert, welches die gewählte Herangehensweise darstellt. Da dieses Model mit der Methode der Entscheidungsbäume verwandt ist, wird diesem die Bezeichnung „*Baummodel*“ gegeben. Das Baummodel ist nicht nur für Briefe sondern für jede andere Dokumentart wie Kontoauszüge oder Bestellungskarten einsetzbar und erleichtert hierbei die verschiedenen Versionen einer Dokumentart herauszufinden. Dies wird anhand von Abbildung 13 veranschaulicht.
Durch die Überprüfung der 50 Briefe konnten insgesamt fünf gemeinsame Blöcke identifiziert werden. Diese Blöcke werden nach ihrem Inhalt benannt und bilden zusammen einen kompletten Brief. Diese sind das Logo-Adresse-, das Betreff-, das Anrede-, das Text-und das Endeblock.

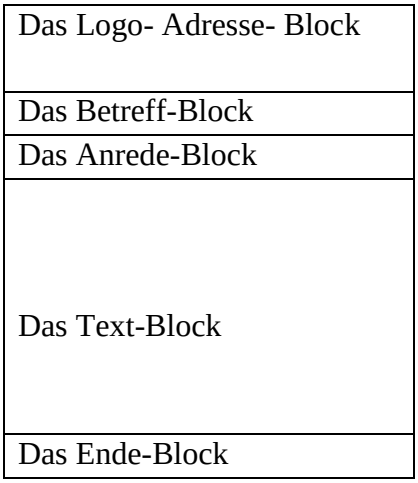


Abbildung 12 : Die fünf Blöcke eines Briefes

Wird dem Prinzip des Baummodels bei einen Brief gefolgt, so ist die folgende Frage nach der Existenz des *XX*-Blocks mit JA oder Nein zu beantworten, wobei *XX* für eines der zuvor genannten Blöcke steht. Dieses Szenario wird in einem Baum modelliert, welcher immer aus einem Wurzelknoten und beliebig vielen inneren Knoten sowie mindestens zwei Blättern besteht. Dabei repräsentiert jeder Knoten einen der fünf Blöcke eines Briefes und jedes Blatt eine Kategorie. Diese Kategorie wird dann als Dokument generiert. Man geht vom Wurzelknoten entlang des Baumes abwärts. Bei jedem Knoten wird die Existenz eines Blockes abgefragt und eine Entscheidung über die Auswahl des folgenden Knoten getroffen. Diese Prozedur wird so lange fortgesetzt, bis ein Blatt erreicht wird, welches der Kategorisierung entspricht. Eine Kategorie setzt sich aus den Blöcken zusammen, die mit der Antwort „JA“ gekennzeichnet wurden. Im betrachteten Fall liegen insgesamt vier Blätter vor.

Ein großer Vorteil dieses Baummodels ist, dass es gut erklärbar und nachvollziehbar ist. Dies ermöglicht, das Ergebnis auszuwerten und die unterschiedlichen Kategorien einer Dokumentart zu erkennen. Das ist vor allem nützlich, wenn später diese Kategorien als Grundlage für das Generieren von Dokumenten verwendet werden.

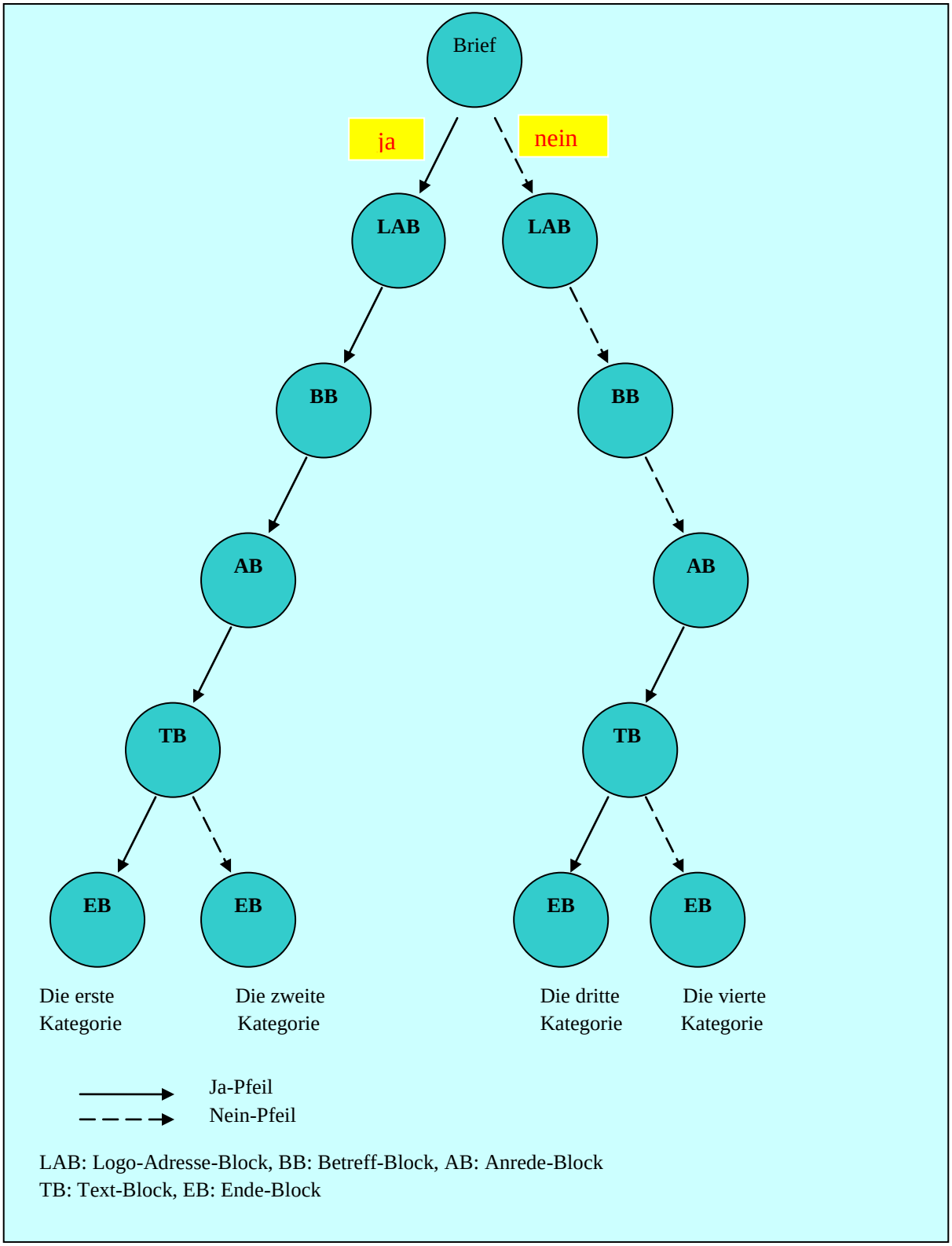


Abbildung 13: Das Baummodel für einen Brief. Viele Innere Knoten werden nicht betrachtet, denn diese Knoten repräsentieren keinen realen Zustand. Zum Beispiel, existiert auf einem Blatt kein Logo-Adresse-Block, dann existiert ebenso kein Betreff-Block, oder ein Blatt kann nur bei einem mehrseitigen Brief nur Text Enthalten.

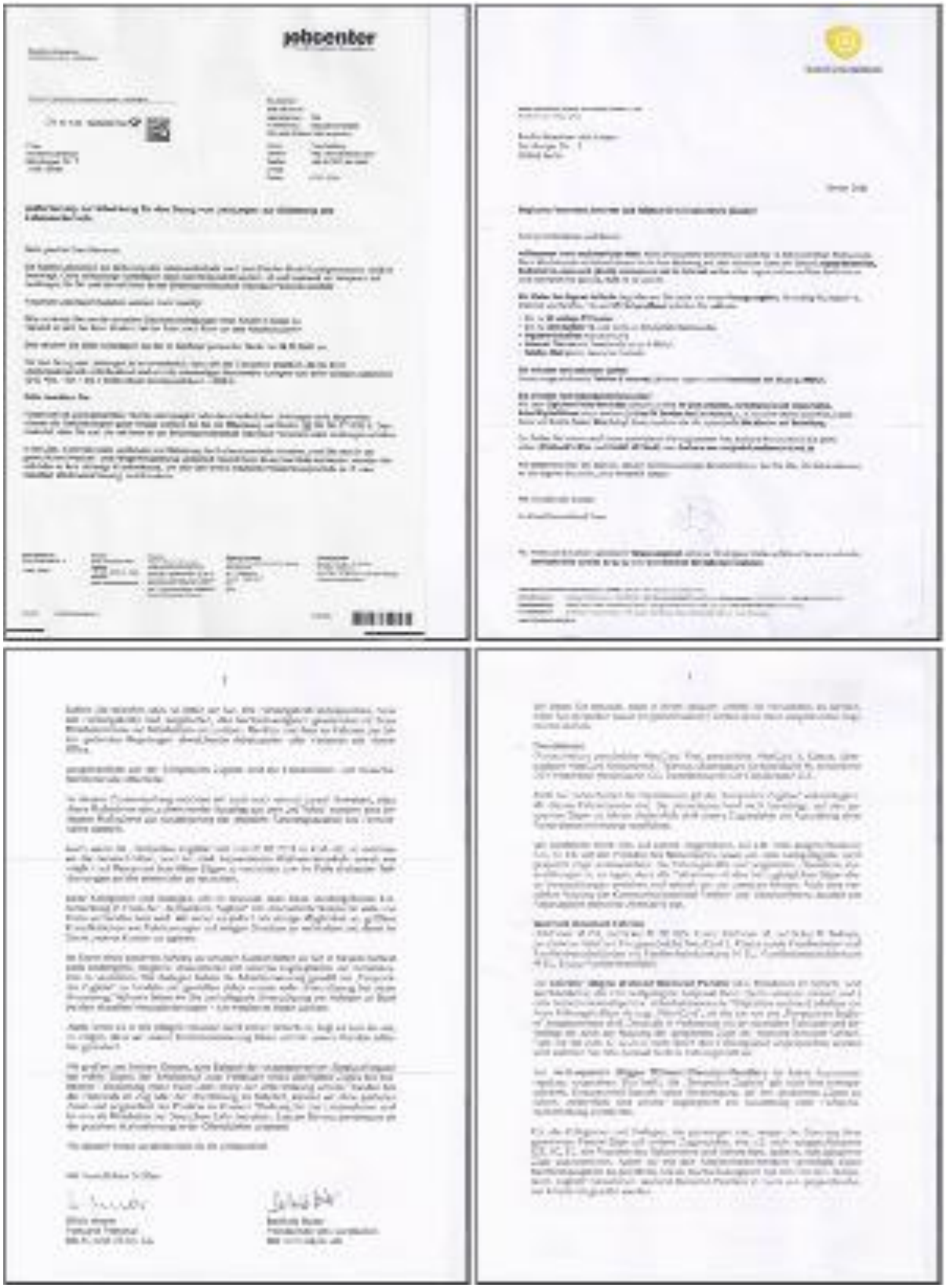


Abbildung 14 : Die vier Kategorien eines Briefes nach dem Baummodel in Abb. 13. Erste Kategorie (o.l) Zweite Kategorie (o.r) Dritte Kategorie (u.l) Vierte Kategorie (u.r)

Im Allgemeinen besteht die Generierung eines Dokumentes aus folgenden Schritten:

1) Das Generieren eines Dokument-Bereiches: In diesem Schritt wird ein rechteckiger Rahmen definiert, welcher das zu generierende Dokument enthält. Die Rahmenmaße entsprechen einer der Standardgrößen für Papierformate, die vom Deutschen Institut für Normen in den DIN-Norm DIN 476 festgelegt worden sind. Danach erfolgt die Formatangabe grundsätzlich mit Breite × Höhe, und zwar immer in dieser Reihenfolge. Deshalb lässt sich daraus schließen, ob es sich um ein Hoch- oder ein Querformat handelt.

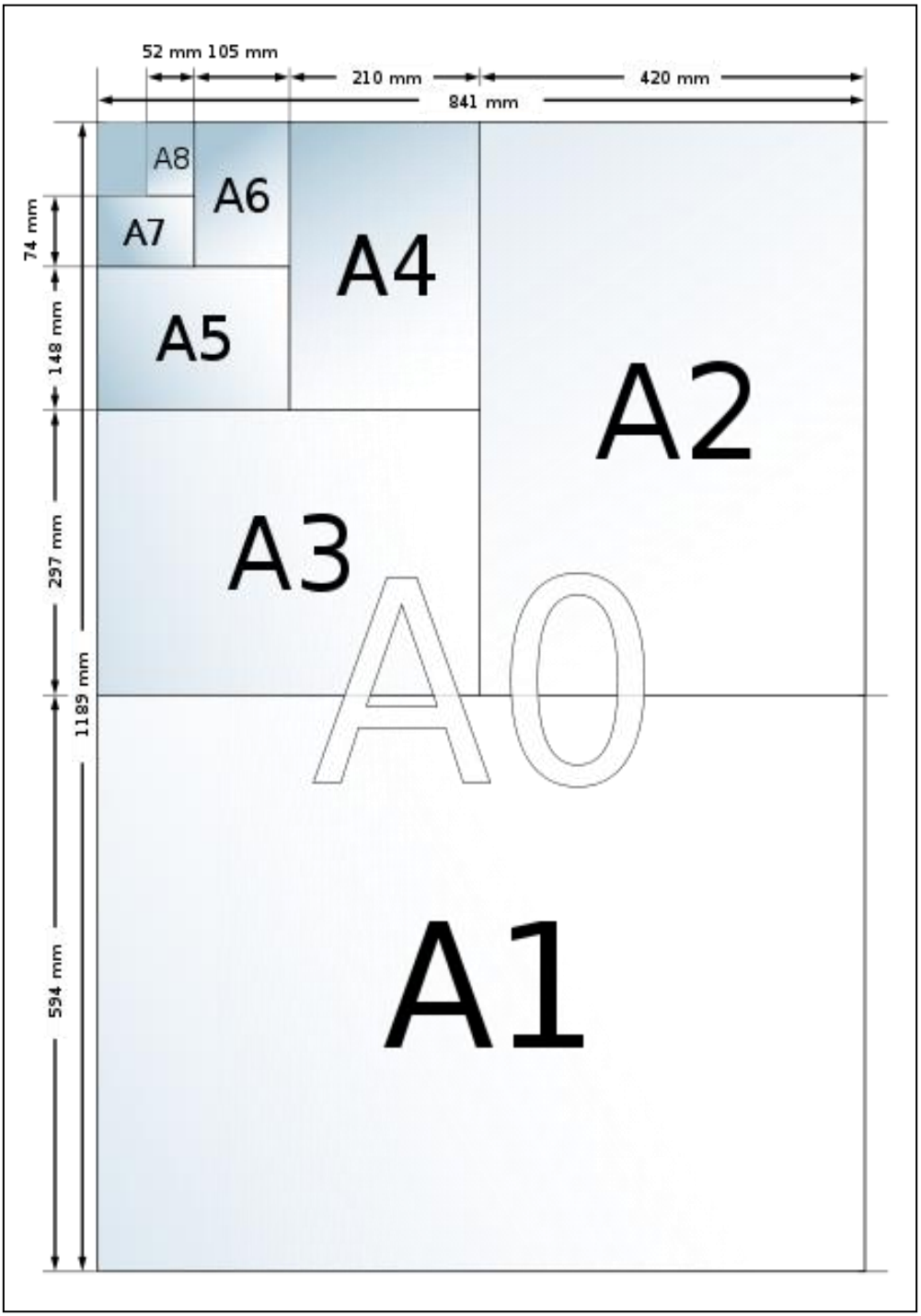


Abbildung 15 aus (13): Aufteilung eines A0-Bogens. Die Formate ergeben sich jeweils durch Halbierung des nächst größeren.

Alle Formate lassen sich durch die folgenden Bedingungen herleiten:

- Alle Formate innerhalb einer Reihe sind einander geometrisch ähnlich.
- Die Halbierung des Formats X(n) an der langen Seite ergibt das Format X(n + 1).
- Das Format A0 hat einen Flächeninhalt von 1 m². Die B-, C- und D-Reihe werden aus der A-Reihe abgeleitet (13).

In der folgenden Tabelle (13) stehen Beispiele für die unterschiedlichen Anwendungen der ISO/DIN Reihen, wobei viele der folgenden Beispiele als Dokument betrachten und generiert werden.

A0, A1	Technische Zeichnungen, See-/Landkarten, Druckbogen, Aushang-Fahrpläne, Poster, Filmplakate, Wahlplakate
A1, A2	Flipcharts, Geschenkpapier, Filmplakate, Fahrpläne, Kalender, Zeitungen, Meisterbrief
A2, A3	Zeichnungen, Diagramme, große Tabellen, Kalender, Karten, Filmplakate
B4, A3	Zeitungen, Noten, Karten
A4	Briefpapier, Formulare, Hefte, Zeitschriften
A5	Notizblöcke, Schulhefte, Prospekte
D5	DVD-Hüllen
A5 bis A8	Karteikarten
A6	Flyer, Postkarten, Taschenbücher, Überweisungsträger, Notizhefte
B5, A5, B6, A6, A4	Bücher
A7	Flugblätter, Taschenkalender, Personalausweis
B7	Spielkarten
B8, A8	Spielkarten, Visitenkarten, Etiketten
C4, C5 C6, C4	Umschläge, Kontoauszüge

2) Das Generieren eines Dokument-Formats: Dieser Schritt hängt vom ersten Schritt ab, da für manche Papierformate auch eine oder mehrere vordefinierte Normen existieren, welche bei der Gestaltung des Papierformats eine Verwendung finden. Es ist aber nicht zwingend, einer bestimmten Norm zu folgen, denn es könnte sich um ein Papierformat handeln, welches auch ohne eine Norm generiert werden kann. Dies könnte eine einfache Buchseite oder eine Karteikarte sein. Aus diesem Grund ist es sehr wichtig vor der Generierung des Dokument-Formats festzulegen, um welche Art von Dokument es sich handelt (Brief, Rechnung, einfache Buchseite etc.). Als Beispiel betrachten wir die DIN-Norm 676, in der die genauen Maßangaben für die DIN-gerechte Aufteilung von DIN-A4-Seiten angegeben sind. Gemeint ist das Briefbogen-Layout, das in der Regel durch einen Grafiker oder Mediengestalter erstellt wird und als Vordruck im Offsetdruck oder Digitaldruck vorproduziert wird. Es gibt zwei Vordrucke, der erste Vordruck A ist für Geschäftsbriefe mit kleinem Briefkopf während der zweite Vordruck B für Geschäftsbriefe mit erweitertem Briefkopf ist. Beide Vordrucke passen auf ein Blatt der Größe DIN A4 und sind in

verschiedene Bereiche eingeteilt, wie Briefkopf, Adressfeld, Informationsblock, Bezugszeichenzeile, Textbereich und Brieffuß (vgl. Abbildung 16).

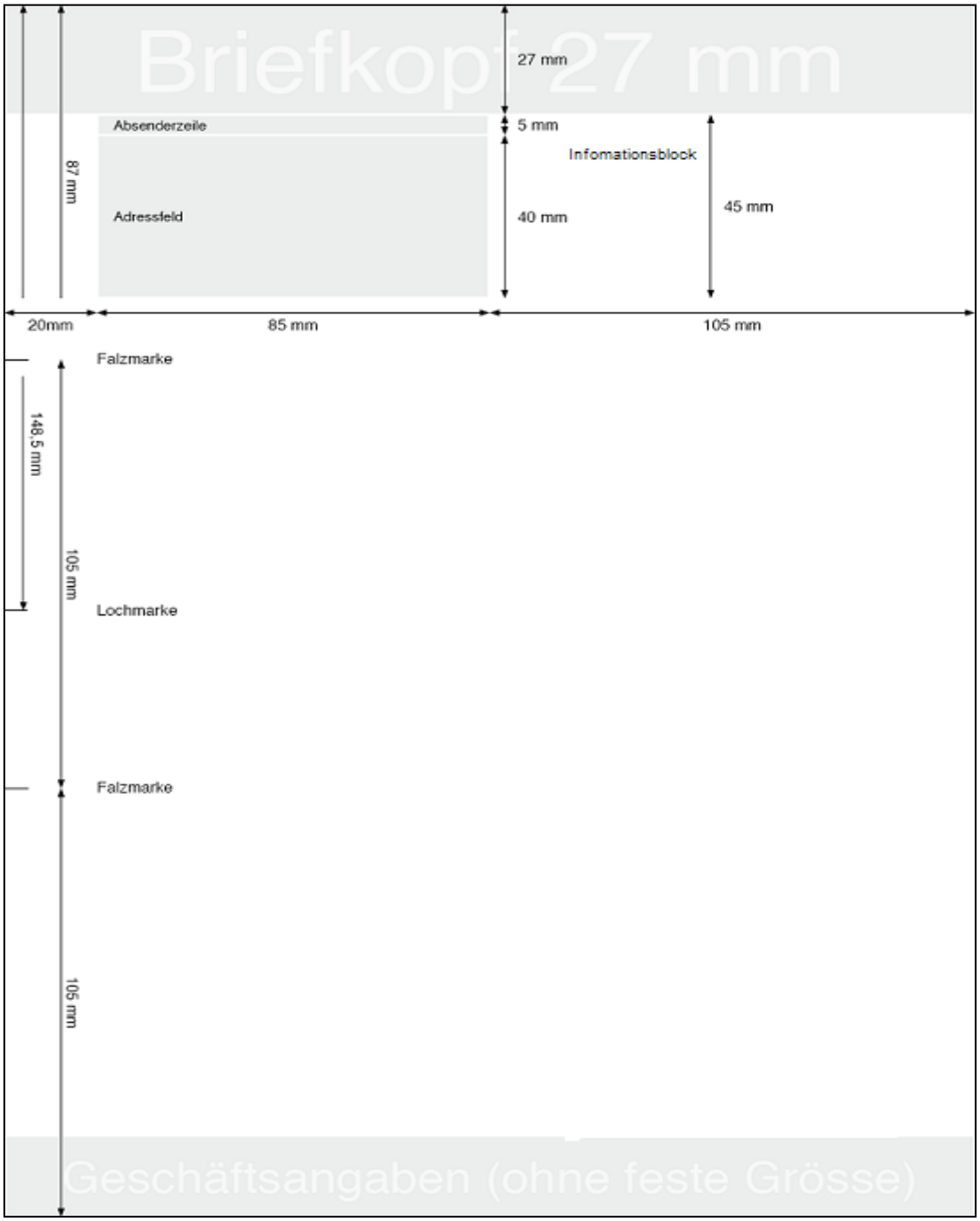


Abbildung 16: Bereiche und deren Maße in Vordruck A nach DIN676. Der Unterschied zum Vordruck B liegt nur bei den erweiterten Bereichmaßen des Briefkopfs (45mm).

Dementsprechend werden innerhalb des Dokument-Bereiches mehrere rechteckige Flächen definiert, wobei diese auch überlappend sein können und im nächsten Schritt mit zufällig generiertem Inhalt befüllt werden.

3) Das Generieren eines Dokument-Inhalts: Das leere Dokument wird in diesem Schritt befüllt. Der Inhalt besteht aus unterschiedlichen Elementen und wird ganz willkürlich und zufällig generiert. Diese Elemente sind Texte, Bilder, Diagramme, Handschrift etc. Ebenfalls gibt es hierfür unterschiedliche vordefinierte Normen, die bei der

Gestaltung des Inhalts benutzt werden können. Die DIN-Norm DIN 5008 ist solch ein Beispiel und legt Schreib- und Gestaltungsregeln für die Textverarbeitung fest. Diese gehört zu den grundlegenden Normen für Arbeiten im Büro- und Verwaltungsbereich. Somit ist diese mit der Norm DIN 676 anzuwenden und legt nicht den Inhalt sondern die Form fest.

Beispielsweise besteht das Adressfeld aus drei Teilen: der Anschrift des Absenders, der Zusatz- und Vermerkzone und der Anschrift des Empfängers. Diese drei Teile kommen zusammen auf neuen Zeilen, die detaillierte Informationen zur Adresse enthalten (14). Ebenso werden die Abstandzeilen zwischen den verschiedenen Briefteilen und die Informationen, die der Brieffuß enthalten kann, festgelegt. Dies sind nur ein paar Beispiele von vielen Regeln und Details, die die Norm DIN 5008 festgelegt hat.

Um diese Regeln bestmöglich umsetzen zu können, wird der Inhalt nach den folgenden zwei Schritten generiert: Im ersten Schritt werden lediglich die Teile des Dokuments gefüllt, die nicht komplett zufällig generiert sind, sondern aus Elementen bestehen, die sich aus Listen mit vordefinierten Vokabeln zusammensetzen. Dieses sind zum Beispiel die Bezugszeichenzeile oder das Adressfeld. Dazu werden mehrere Listen für Straßen-, Stadt- und Personennamen und Ordner für Bilder, Logos und Abbildungen benutzt. Im zweiten Schritt wird der Inhalt von Dokumentteilen, deren Aufbau keinem bestimmten Muster entspricht – z.B. der Textbereich -, zufällig generiert. Dieser setzt sich aus Sätzen zusammen, welche wiederum aus zufällig erzeugten Wörtern entstehen. Die Wörter werden durch willkürliche Zusammensetzung von Buchstaben generiert und haben somit in der Regel keine sprachliche Bedeutung.

4) Hinzufügen von Störungen und Hintergründe: Nach dem dritten Schritt des Generierungsprozesses bilden sich vollständige Dokumente, die in den meisten Fällen einen weißen Hintergrund haben. Daher werden verschiedene Bildfilter eingesetzt, um Störungen hinzu zufügen und somit realitätsnahe Dokumente zu generieren. Zusätzlich wird der weiße Hintergrund modifiziert so, dass Dokumente mit unterschiedlichen Hintergrundfarben und Hintergrundmustern erzeugt werden.

Die Menge der generierten Dokumente wird als Eingabe für den nächsten Prozess, das Zerschnipseln von Dokumenten, verwendet.

3.1.2 Das Speichern der generierten Dokumenten

Beim Starten des Prozesses wird ein neues Verzeichnis mit einer eindeutigen fortlaufenden Nummerierung erstellt, in dem die neu generierten Dokumente gespeichert werden.
Der Name jedes neu generierten Dokumentes besteht aus 18 Zeichen und ist in sechs Teile aufgeteilt.

- Teil 1 besteht aus drei Stellen und enthält die Nummer des aktuellen Verzeichnisses, in dem die generierten Dokumente gespeichert sind. Sein Startwert beträgt „001“.
- Teil 2 besteht aus drei Stellen und enthält eine eindeutige und fortlaufende Nummerierung für das neu generierte Dokument. Sein Startwert beträgt 001.

- Teil 3 besteht ebenso aus drei Stellen und enthält eine eindeutige und fortlaufende Nummerierung für einen neu generierten Schnipsel. Dies folgt jedoch im nächsten Prozess. Daher ist sein Wert in diesem Prozess noch „000“
- Teil 4 besteht aus drei Stellen und gibt das Format des Dokumentes an (Bsp. A04, B10 etc.).
- Teil 5 besteht aus fünf Stellen und gibt die Art des Dokumentes an. Jedoch wird eine Abkürzung für die Namen gespeichert falls diese aus mehr als fünf Zeichen besteht (Bsp. Brief, Rechn, Other, etc.).
- Teil 6 besteht aus einer Stelle und gibt die Art des Zerschnipselns an, wobei 1 für Schreddern, 2 für Scheren und 3 für Handzerreißen steht. In diesem Prozess wird der Wert 0 gespeichert, da aktuell noch ein ganzes Dokument vorhanden ist. Abbildung 17 veranschaulicht dies etwas genauer.

Als Beispiel bezeichnet der Name „012144000A04Brief0“ das 144te generierte Dokument, welches im zwölften Ordner gespeichert ist, noch nicht zerschnipselt ist und einen Brief in DIN A4 Format enthält.

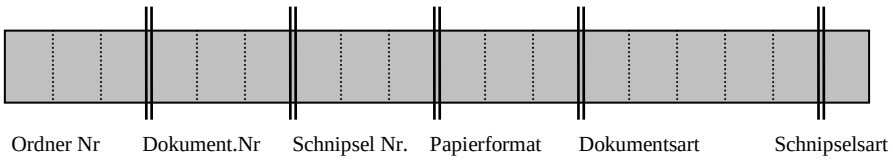


Abbildung 17: Die Felder des Speicherungsnamens eines Dokumentes

Der Vorteil bei diesem Schema ist, dass es die eindeutige Adressierung von bis zu eine Million zufällig generierte Dokumente und deren knapp eine Milliarde Schnipseln ermöglicht und zusätzlich optimale Test- und Trainingsmengen für die unterschiedlichen Erkennungs- und Rekonstruktionsalgorithmen erstellt. Somit kann mit Hilfe dieses Schemas die Erfolgsquote solcher Algorithmen ausgewertet werden, was den Vergleich und die Optimierung ihrer Leistung ermöglicht.

3.1.3 Ein führendes Beispiel

Sei DIN A4 das ausgewählte Papierformat. Sei ebenso die Norm DIN 676 in Vordruck A das ausgewählte Dokumentenformat sowie Brief die Art des Dokumentes. Abbildung 18 zeigt ein Beispiel eines zufällig generierten Dokuments. Dabei fallen insbesondere die blau markierten Rechtecke auf, die im zweiten Schritt erzeugt werden und im dritten Schritt mit zufällig generiertem und ausgewähltem Text befüllt wurden. Das Dokument oben links in der Abbildung 18 stellt ein Ergebnis des Dokumenten-Generators bis zum dritten Schritt dar. Die restlichen Dokumente illustrieren drei verschiedene Ergebnisse des vierten Schritts ausgeführt auf demselben Dokument.

Abbildung 18: Ein Ergebnisbeispiel des Dokumenten-Generators. Das Beispieldokument (o.l) nach dem dritten Schritt, (o.r) nach dem vierten Schritt, hier einfache Störung hinzugefügt, (u.l) ebenso nach dem vierten Schritt, hier wurde das Gauß-filter verwendet, (u.r) nach der Binarisierung.

3.2 Der Prozess des (Z)erschnipselns

Bei den Algorithmen für die Rekonstruktion von geschredderten bzw. handzerrissenen Dokumenten werden vorerst die Dokumente gescannt (vgl. Abschnitt 2.4). Ziel ist es, einen Ersatz für den Scanvorgang zu finden. Daher wird in diesem Abschnitt eine Methode beschrieben, deren Aufgabe darin besteht, Dokumente und Bilder virtuell zu zerschnipseln und diese in einer geeigneten Referenzdatenbank für die Rekonstruktionsalgorithmen zu speichern.

3.2.1 Das virtuelle Schreddern

Die geschredderten Teile eines Dokumentes besitzen den Vorteil, dass sie die gleichen Schnittmuster haben. Das hilft dabei, das virtuelle Schreddern einfach zu realisieren. Gegeben sind ein beliebiges Dokument *D* aus der Menge der zufällig generierten Dokumente und eine der ersten vier Schredder-Sicherheitsstufen. Die Sicherheitsstufe wird als ein rechteckiges Raster *R* betrachtet. Hierbei seien h_D, w_D und h_R, w_R die Höhe und die Bereite des Dokumentes bzw. des Rasters. Wobei $w_D > w_R$ und $h_D \geq h_R$ Es wird von einem Koordinatensystem aus gegangen, welches seinen Ursprungspunkt (0,0) in der oberen linken Ecke hat. Wobei positive *X*-Werte nach rechts und positive *Y*-Werte nach unten verlaufen. Seien (x_0, y_0) die Koordinaten der linken oberen ecke des Dokuments und (x_R, y_R) die Koordinaten der linken oberen Ecke des Rasters, dann werden die folgenden Schritte unter den Bedingungen, w_D ist ein Vielfaches von w_R und h_D ist ein Vielfaches von h_R, durchgeführt:

---

setze $(x_R, y_R) = (x_0, y_0)$
while $(x_R \leq x_0 + w_D)$
    while $(y_R \leq y_0 + h_D)$
        Alle Pixel, die innerhalb des Rasters sind, in einem neuen Bild kopieren
        $y_R = y_R + h_R$ // Y-Koordinaten des Rasters verschieden
        $x_R = x_R + w_R$ // X-Koordinaten des Rasters verschieden
    $y_R = y_0$
ende

Die neu erzeugten Dokumentenstücke werden entsprechend dem Abschnitt 3.1.2 gespeichert, indem jedes eine eindeutige Nummer zugewiesen bekommt, wobei die 18te Stelle den Wert Eins hat. So wird es möglich eine Trainingsmenge aus mehreren geschredderten Dokumenten in mehreren Ordnern zu bilden.
Die Abbildung 19 illustriert ein Beispiel des virtuellen Schreddern.

Abbildung 19: Ein Ergebnisbeispiel für das virtuelle Schreddern.

3.2.2 Das virtuelle Scheren

Im Vergleich zum virtuellen Schreddern ist das virtuelle Scheren nicht ohne Weiteres realisierbar, insbesondere da die zerlegten Dokumentenstücke keine einheitlichen Muster besitzen. Außerdem wird danach gestrebt, alle Schneidszenarien bestmöglich und realitätsnah zu simulieren.
Die Lösung des Problems beruht darauf, von der Transparenzeigenschaft einiger Bildformate Gebrauch zu machen. So werden die generierten Dokumente in PNG-Format gespeichert. Portable Network Graphics (PNG) ist ein Grafikformat für Rastergrafiken mit verlustfreier Bildkompression, die neben unterschiedlichen Farbtiefen auch Transparenz per Alphakanal unterstützt. So können PNG-Dateien Transparenzinformationen enthalten, entweder in Form eines Alphakanals oder für jede Farbe der Farbpalette. Ein Alphakanal ist eine zusätzliche Information, die für jedes Pixel angibt, wie viel vom Hintergrund des Bildes durchscheinen soll. PNG unterstützt Alphakanäle von 8 oder 16 Bit, was 256 beziehungsweise 65536 Abstufungen der Transparenzstärke entspricht (15).
Um das Dokument zu zerlegen, wird eine Schablone zum Ausschneiden aufgebaut. Diese Schablone hat die gleichen Maße des rechteckigen Dokumentes und ist in viele unterschiedliche Polygone geteilt. Die folgende Abbildung stellt ein Beispiel für eine Schablone mit acht Polygonen dar. Das Ergebnis des Ausschneidens nach dieser Schablone ist in der Abbildung 21 illustriert.

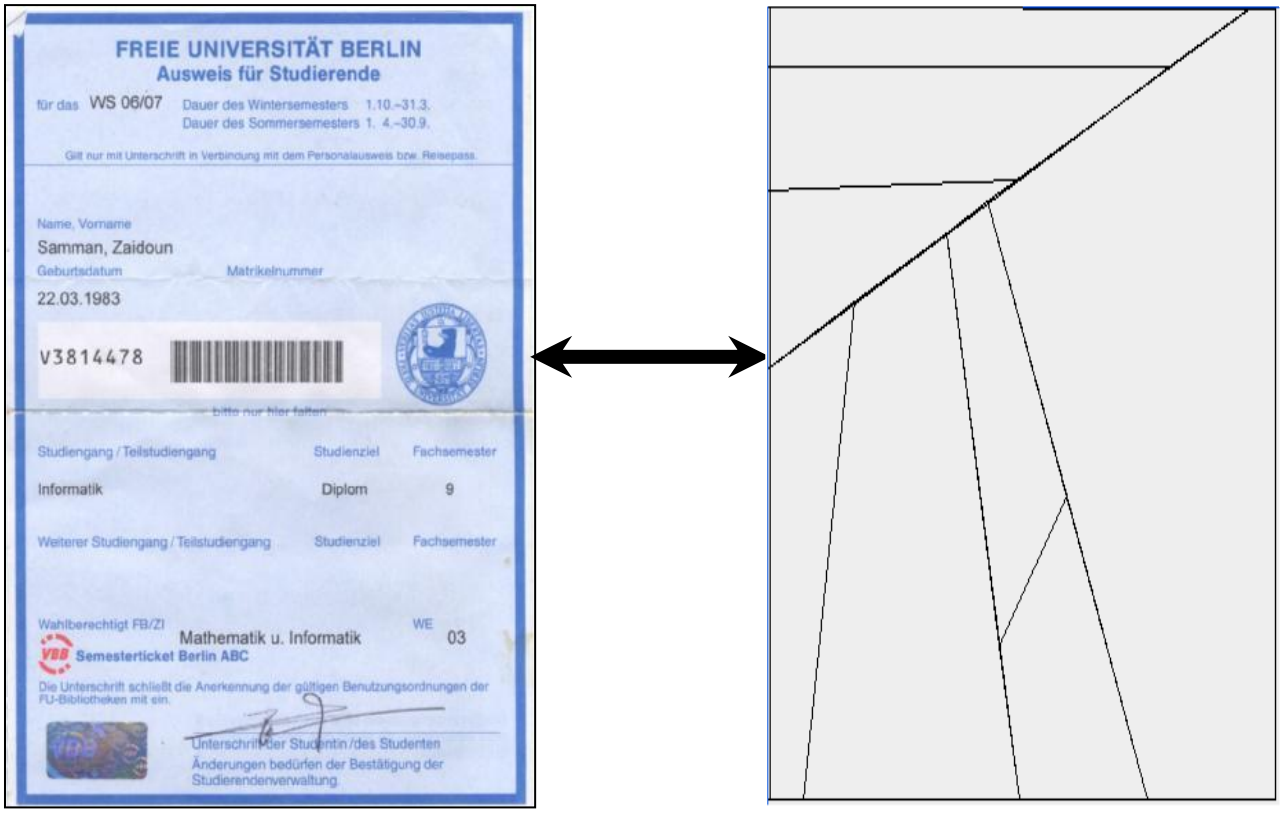


Abbildung 20: Eine Schablone zum Ausschneiden eines Dokumentes.

Es besteht die Möglichkeit für jedes Dokument eine Schablone zu erzeugen oder auch für mehrere Dokumente. Durch die Anwendung der Schablone kann ein Dokument in mindestens

zwei Teile zerlegt werden, wobei die maximale Anzahl der Teile von der Größe des Dokumentes abhängig ist. Wie diese Schablone entsteht, wird im folgenden Algorithmus erläutert.
Sei $PQ = \{P_i \mid 0 < i \leq n\ ,\ \forall P_i\ gilt\ MUR(P_i) \geq min,\ min \in \mathcal{R} \wedge min > 0\}$ eine n-elementige Menge von Polygonen. Wobei $MUR(P_i)$ die fläche des minimalen umgebenden Rechtecks, welches das Polygon P_i umschließt. Sei P das Startpolygon, welches dieselben Maße wie das zu zerlegende Dokument besitzt. Die Menge PQ enthält vorm Start des Algorithmus keine Elemente.

$Schablone(P, n, min)$
$if(n > 1)$
    $teile\ (P, min)$
    $while(n > 2)$
        wähle ein zufälliges Polygon P_i aus PQ aus
        $if\ (MUR(P_i) > min \times \varphi)$
            $teile\ (P_i, min)$
            entferne P_i aus PQ
            $n - -$
else füge P zu PQ

Wobei $\varphi > 1$ eine natürliche Zahl ist, damit nur Polygone mit mindestens doppelter Fläche geteilt werden. $teile\ (P, min)$ ist eine Methode, die ein Polygon in zwei Polygonen zerlegt und in PQ hinzufügt, unter der Bedingung, dass beide Polygone eine *MUR* Fläche haben, die größer als *min* ist.

$teile\ (P, min)$
wähle zufällig eine Polygonseite *s* aus.
wähle zufällig einen Punkt *x* aus *s* aus.
wähle zufällig ein Winkel *w* zwischen 10 und 170 Grad aus.
teile P in zwei Polygone P_i und P_j , anhand der Gerade, die durch den Punkt *x* geht und mit der Seite *s* einen Winkel *w* bildet.
$while\ (\ MUR(P_i < min\ ) \vee\ MUR(P_j < min\ ))$
    $teile\ (P, min)$
füge P_i und P_j zu PQ

Der Algorithmus terminiert, wenn die gewünschte Anzahl von kleinen Polygonen erreicht wird. Diese bilden zusammen den Ausgangspolygon und sind in der Menge PQ gespeichert, wie in Abbildung 20 gezeigt wird. Die erzeugte Schablone wird auf das Dokument überlappt. Dabei werden die einzelnen Polygone in PQ als eine Art Fenster betrachtet. Auf diese Weise bildet ein transparentes Polygon eine Art geöffnetes Fenster. Das Dokumentteil, welches durch dieses Fenster erscheint, wird als neues Schnipsel gespeichert. Das neue Schnipsel trägt den Namen entsprechend dem Abschnitt 3.1.2, in dem jedes neu erzeugte Dokumentstück eine eindeutige Nummer zugewiesen bekommt, wobei die 18te Stelle den Wert 2 hat.

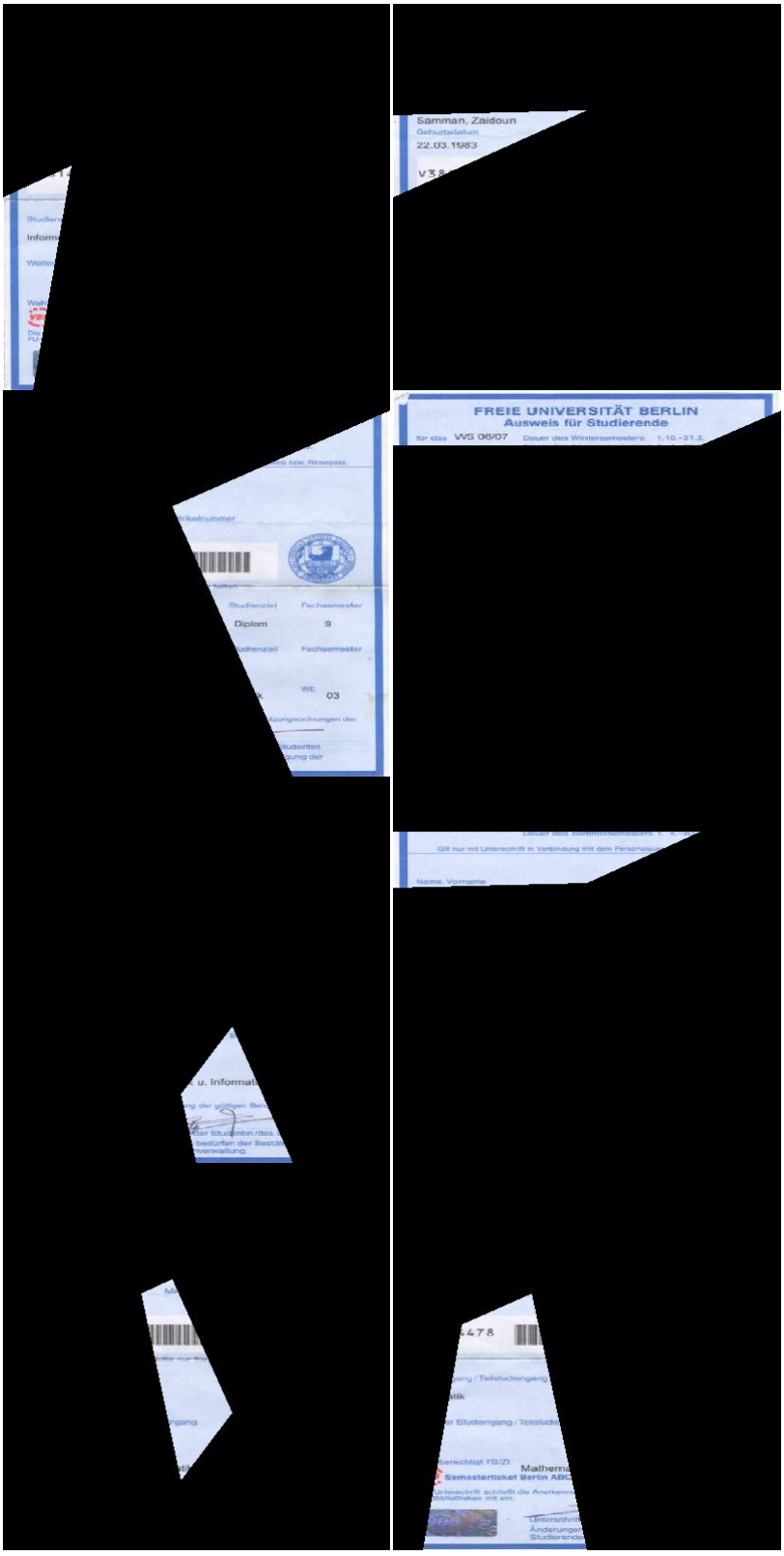


Abbildung 21: Das Ergebnis des Ausschneidens nach der Schablone in Abbildung 20.

3.2.3 Das virtuelle Handzerreißen

Für die Erzeugung handzerrissener Dokumente wird das gleiche Verfahren wie im vorherigen Abschnitt angewendet. Jedoch treten in diesem Fall zwei Probleme auf. Das erste liegt bei der Form der handzerrissenen Schnipsel, denn diese weisen eine willkürliche und uneinheitliche geometrische Form auf. Das zweite Problem stellen die Konturen solcher Schnipsel, da diese nicht gradlinig sind, wie es in der Abbildung 22 zu sehen ist.

Abbildung 22: Konturverlauf eines handzerrissenes Dokuments

Die Veränderung der Innenstruktur von der Ausschneideschablone stellt eine Möglichkeit dar, das erste Problem zu lösen. Das Prinzip dieses Vorgehens besteht drin, dass nacheinander, ausgehend von einem festgelegten Punkt zusammengesetzte Segmente gezeichnet werden. Diese Segmente bestehen nicht wie bei Polygonen ausschließlich aus Linien, sondern enthalten auch quadratische und kubische Kurven. Der Endpunkt eines Segmentes ist wiederum der Startpunkt für ein nachfolgendes Segment. Dadurch werden die Segmente mit einander verbunden und bilden zusammen die Innenstruktur der Ausschneideschablone (vgl. Abbildung 23). Um dies zu realisieren wird die Klasse *java.awt.geom.GeneralPath* benutzt, welche verschiedene Methoden wie *lineTo(), curveTo(), quadTo()*zur Verfügung stellt. Diese Methoden bauen die Segmente Schritt für Schritt auf. Mit der Methode *append()* werden die entstehenden geometrischen Formen miteinander verbunden.

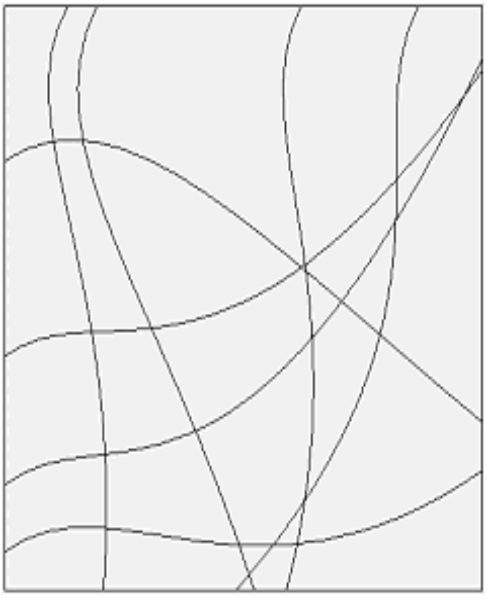


Abbildung 23: Die optimierte Schablone zum Ausschneiden von Dokumenten

Abbildung 23 zeigt eine generierte Ausschneideschablone, deren Innenstruktur aus unterschiedlichen geometrischen Formen zusammengesetzt ist. Jedoch haben die Konturen dieser geometrischen Formen und damit auch die Konturen der erzeugten Schnipsel nicht den gewünschten Handzerrissen-Effekt (vgl. Abschnitt 5.2.2).

3.3 Der Prozess des (V)orklebens

In diesem letzten Prozess beschreiben wir einen Algorithmus, dessen Aufgabe darin besteht, die Informationen auf einem handzerrissenen Schnipsel zu analysieren und sie zu klassifizieren. Dieser Algorithmus soll nicht die herkömmlichen Algorithmen zur Rekonstruktion von handzerrissenen Dokumenten ersetzen sondern erweitern und den Fällen, in denen diese Algorithmen zu nichts führen eine Alternative darstellen. Damit ist ein weiteres Ziel die Quote der erfolgreich zusammengesetzten Dokumente zu erhöhen. Wie der Name des Prozesses schon andeutet, wird dieser Ansatz vor dem Zusammensetzen der Schnipsel ausgeführt. Bei einer enorm hohen Anzahl von Schnipseln ist die Wahrscheinlichkeit der Existenz von zwei oder mehr ähnlichen Schnipselkonturen relativ hoch, insbesondere können solche Schnipsel beim gleichzeitigen Zerreißen von zwei oder mehreren Dokumenten entstehen. Abbildung 24 illustriert ein ähnliches Szenario. Dabei deuten die zwei Farben auf den Unterschied des Dokumenteninhaltes. Die Rekonstruktion anhand desselben Kontourverlaufs, ohne den Inhalt des Dokumentes zu analysieren, könnte dazu führen, nicht zusammenpassende Schnipsel zusammenzuführen. Aufgrund dieser Tatsache wird an dieser Stelle der Vorkleben-Prozess vorgestellt, der aus drei Schritten besteht:

1) die Clusteranalyse: bei der die Informationen auf einem Schnipsel in mehrere homogene Gruppen geteilt werden,
2) die Klassifikation: diese Cluster werden in Klassen voneinander unterschieden, die Klassen, Texte, Bilder und Diagramme sein können und
3) die Projektion: die Position einiger Klasse wird auf dem Schnipsel berechnet und auf die Konturen projiziert.

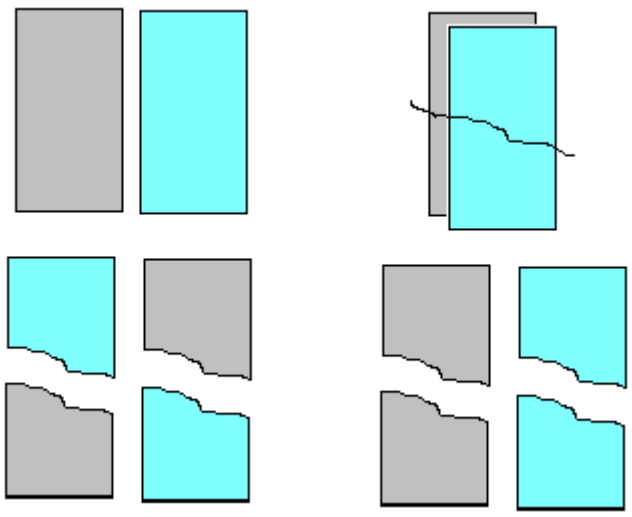


Abbildung 24: ein mögliches Szenario für die Rekonstruktion von Schnipseln. l.u zeigt ein falsches Ergebnis der Rekonstruktion

3.3.1 Die Clusteranalyse

Ausgegangen von einer Sammlung nicht klassifizierter Objekte und einer Methode, die die Ähnlichkeit von Objekten misst, besteht das Ziel darin, diese Objekte in eine Hierarchie von Klassen einzuordnen, die einem bestimmten Qualitätsstandard genügen, wie z. B. die Ähnlichkeiten von Objekten zu maximieren, die der gleichen Klassen angehören. Eine Vielzahl von Clusteralgorithmen ist bereits entwickelt worden, welche sich durch unterschiedliche Vorgehensweisen und Strategien auszeichnen.
Der Vielfalt und die Verteilung des Schnipselinhalts, der im Vorhinein unbekannt ist und die mögliche Existenz von Rauschen, sind die entscheidenden Gründe für die Anwendung von dichte-basierte Clusterverfahren.

Der DBSCAN Algorithmus
Density-Based Spatial Clustering of Applications with Noise, bedeutet auf Deutsch so viel wie dichtebasierte räumliche Clusteranalyse mit Rauschen. Dieser Algorithmus arbeitet dichtebasiert und ist in der Lage mehrere Cluster zu erkennen. Rauschpunkte werden als eine spezielle Gruppe angesehen, die Objekte beinhaltet, welche keinem Cluster zugeordnet werden können und somit ignoriert werden (16). Für das Verständnis des DBSCAN Algorithmus werden im Folgenden einige notwendige Grundbegriffe definiert und erläutert.

Grundlagen
Sämtliche folgende Definitionen (16) beziehen sich auf die beiden Parameter ε und *MinPts*, wobei *MinPts* eine natürliche Zahl ist und ε eine reelle Zahl größer Null ist.

Definition 1: (*ε-Umgebung*) Die ε-Umgebung eines Punktes *o* ist wie folgt definiert

$$N_{\mathcal{E}}(o) = \{o' \in O \mid dist(o, o') \leq \mathcal{E}\}$$

Die verwendete Metrik *dist* legt die Form der ε-Umgebung fest. Wird die euklidische Distanz eingesetzt, hat die ε-Umgebung die Form einer Hypersphäre; die Mahalanobis-*Distanz* führt zu einem Ellipsoid.

Definition 2: *(Kernobjekte)* Ein Punkt $o \in O$ wird als *Kernpunkt* bezeichnet, wenn die folgende Bedingung $|N_{\mathcal{E}}(o)| \geq MinPts$ erfüllt ist. Demnach ist ein Kernpunkt ein Punkt, in dessen ε-Umgebung sich mindestens *MinPts* Punkte befinden. Punkte, die nicht als Kernobjekte identifiziert werden, können entweder Randobjekte eines Clusters sein oder zur speziellen Gruppe von Rauschen gehören.

Definition 3: *(Direkte Dichte-Erreichbarkeit)* Ein Punkt *o* ist direkt dichte-erreichbar von einem Punkt p, wenn $o \in N_{\mathcal{E}}(p)$ und p ein Kernpunkt ist.

Definition 4: *(Dichte-Erreichbarkeit)* Ein Punkt *o* ist dichte-erreichbar von Punkt p wenn es einen Weg von Punkten $P_1 \ldots . P_n \wedge P_1 = o \wedge P_n = p$ gibt, sodass P_{i+1} direkt dichte-erreichbar von P_i ist

Definition 5: *Dichte-Verbundenheit)* Ein Punkt *o* ist mit einem Punkt *p* verbunden, wenn es einen anderen Punkt *q* gibt, sodass sowohl *o* als auch *p* von *q* dichte-erreichbar sind. Diese Definitionen sind in der Abbildung 25 illustriert.

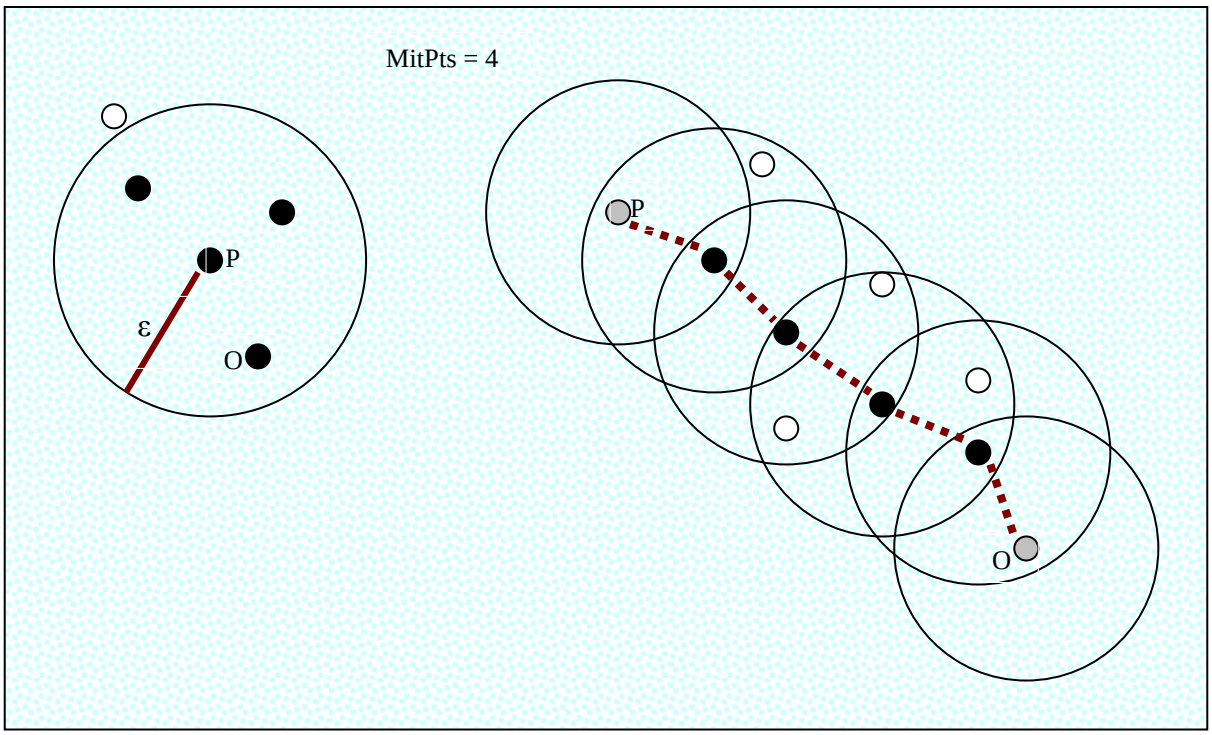


Abbildung 25: links, P illustriert die Definition des Kernpunkts. Rechts, P und O sind dichte-verbunden, P bzw. O sind dichte-erreichbar und zugleich auch Randpunkte; Die schwarze Punkt sind alle direkt dichte-erreichbar

Definition 6: *(Cluster)* Sei *O* eine Datenbank von Punkten. Ein Cluster *C* ist eine nicht leere Menge von *O*, welche die folgenden Bedingungen erfüllt:

i. $\forall p, q : p \in C$ und q ist dichte-erreichbar von $p \Longrightarrow q \in C$.
ii. $\forall p, q \in C$: p ist verbunden mit q (Verbundenheit).

Also stellt ein Cluster im DBSCAN Algorithmus eine Menge von verbunden Punkten dar. Diese Punkte sind von einem beliebigen Kernpunkt dieses Cluster dichte-erreichbar bzgl. der beiden Parameter ε und *MinPts*.

Beschreibung des Algorithmus

Der Algorithmus startet mit einem beliebigen Punkt *p* aus der Punktmenge *O*. Zusätzlich bekommt der Algorithmus in der Eingabe die beiden Parameter ε und *MinPts* festgelegt. Am Anfang sind alle Punkte als unklassifiziert gekennzeichnet und werden zunächst nacheinander besucht. Sollte ein Punkt nicht bereits zu einem Cluster gehören, so wird ein neuer Cluster erzeugt und die Unterfunktion *ExpandCluster* aufgerufen (16).

```
DBSCAN (SetOfPoints, Eps, MinPts)
// SetOfPoints is UNCLASSIFIED
     ClusterId := nextId(NOISE);
     FOR i FROM 1 TO SetOfPoints.size DO
          Point := SetOfPoints.get(i);
          IF Point.ClId = UNCLASSIFIED THEN
```

```
                IF ExpandCluster(SetOfPoints,Point,ClusterId,
                Eps, MinPts) THEN
                     ClusterId := nextId(ClusterId)
                END IF
           END IF
      END FOR
END; // DBSCAN
```

DBSCAN Algorithmus aus (16).

NOISE ist stellt eine spezielle *ClusterId* für Rauschen dar. *SetOfPoints.get(i)* holt den i-ten Punkt aus der Menge *O*. Die Unterfunktion untersucht zunächst die ε-Umgebung des aktuellen Punktes. Handelt es sich hier um einen Kernpunkt, so fügt sie alle Punkte in dieser Umgebung dem neuen Cluster hinzu, andernfalls wird der aktuelle Punkt als Rauschen klassifiziert. Es kann aber durchaus in einem späteren Durchlauf zu einem andern Cluster hinzugefügt werden. In diesem Fall stellt dieser Punkt einen Randpunkt dar.

```
ExpandCluster(SetOfPoints,Point,ClId,Eps,MinPts): Boolean;
   seeds:=SetOfPoints.regionQuery(Point,Eps);
   IF seeds.size < MinPts THEN // no core point
      SetOfPoint.changeClId(Point,NOISE);
      RETURN False;
   ELSE //all points in seeds are density-reachable from Point
      SetOfPoints.changeClIds(seeds,ClId);
      seeds.delete(Point);
      WHILE seeds <> Empty DO
         currentP := seeds.first();
         result := SetOfPoints.regionQuery(currentP,Eps);
         IF result.size >= MinPts THEN
            FOR i FROM 1 TO result.size DO
               resultP := result.get(i);
               IF resultP.ClId IN {UNCLASSIFIED, NOISE} THEN
                  IF resultP.ClId = UNCLASSIFIED THEN
                     seeds.append(resultP);
                  END IF;
                  SetOfPoints.changeClId(resultP,ClId);
               END IF; // UNCLASSIFIED or NOISE
            END FOR;
         END IF; // result.size >= MinPts
         seeds.delete(currentP);
      END WHILE; // seeds <> Empty
      RETURN True;
   END IF
END; // ExpandCluster
```

ExpandCluster aus (16).

Die Menge *seeds* beinhaltet die Punkte aus der ε-Umgebung des aktuellen untersuchten Punktes *currentP*. Diese Punkte werden mit dem Aufruf *SetOfPoints.regionQuery* untersucht. Dadurch werden alle neu gefundenen Punkte, welche noch unklassifiziert sind, ebenfalls der Menge *seeds* hinzugefügt.
Die Umsetzung der Methode *regionQuery* hat einen großen Einfluss auf die Laufzeit vom DBSCAN Algorithmus. Eine naive Implementierung führt im schlimmsten Fall bei einer Menge $||O|| = n$ zu einer Gesamtlaufzeit von $O(n^n)$. Durch den Einsatz von komplexen Datenstrukturen wie etwa den R*-Baum lässt sich die Gesamte Laufzeit auf $O\ (n \log n)$ verbessern, da der Aufruf *regionQuery* in schlimmsten Fall $O\ (\log n)$ Zeit benötigt (16).

Die Abbildung 26 illustriert einen Durchlauf des DBSCAN Algorithmus für eine Menge *O* mit sieben Punkten und *MinPts* = 4. Am Anfang sind alle Punkte weiß markiert und somit nicht klassifiziert; Menge *seeds* ist auch leer (Bild 1). Der erste Punkt aus der Menge O wird ausgewählt. Die Unterfunktion *ExpandCluster* wird mit dem Punkt *a* aufgerufen. Alle Punkte in dessen ε-Umgebung werden schwarz markiert und der Menge *seeds* hinzugefügt. Also beinhaltet *seeds* folgende Punkte, *a*,*b*,*e*, und *d*. (Bild 2). Der Punkt *a* bzgl. *MintPts* und ε ist ein Kernpunkt, daher wird dieser mit der ersten *ClusterId* versehen, in dem Fall grün markiert. Damit gehören alle Punkte in *seeds* zum selben Cluster. Zunächst wird *a* aus *seeds* entfernt. Der nächste Punkt *b* aus *seeds* wird betrachtet und dessen ε-Umgebung untersucht. (Bild 3). Sobald die Menge *seeds* keine Punkte mehr beinhaltet, terminiert die while Schleife, und liefert *ExpandCluster* den Wert True zurück. Alle Punkte des ersten gefunden Clusters sind grün markiert. Die *ClusterId* wird nun erhöht und *ExpandCluster* wird mit dem nächsten noch weiß markierten Punkt *f* wieder aufgerufen. (bild 4). Der Punkt *f* ist bzgl. *MintPts* und ε kein Kernpunkt, daher wird er grau markiert bzw. als Rauschen klassifiziert und aus *seeds* entfernt. *ExpandCluster* liefert false zurück. Der letzte Punkt *g* aus der Menge *O* wird ausgewählt und ebenfalls als Rauschen klassifiziert. (Bild 5). Der Algorithmus terminiert, da alle Punkte aus *O* bereits bearbeitet wurden. (Bild 6).

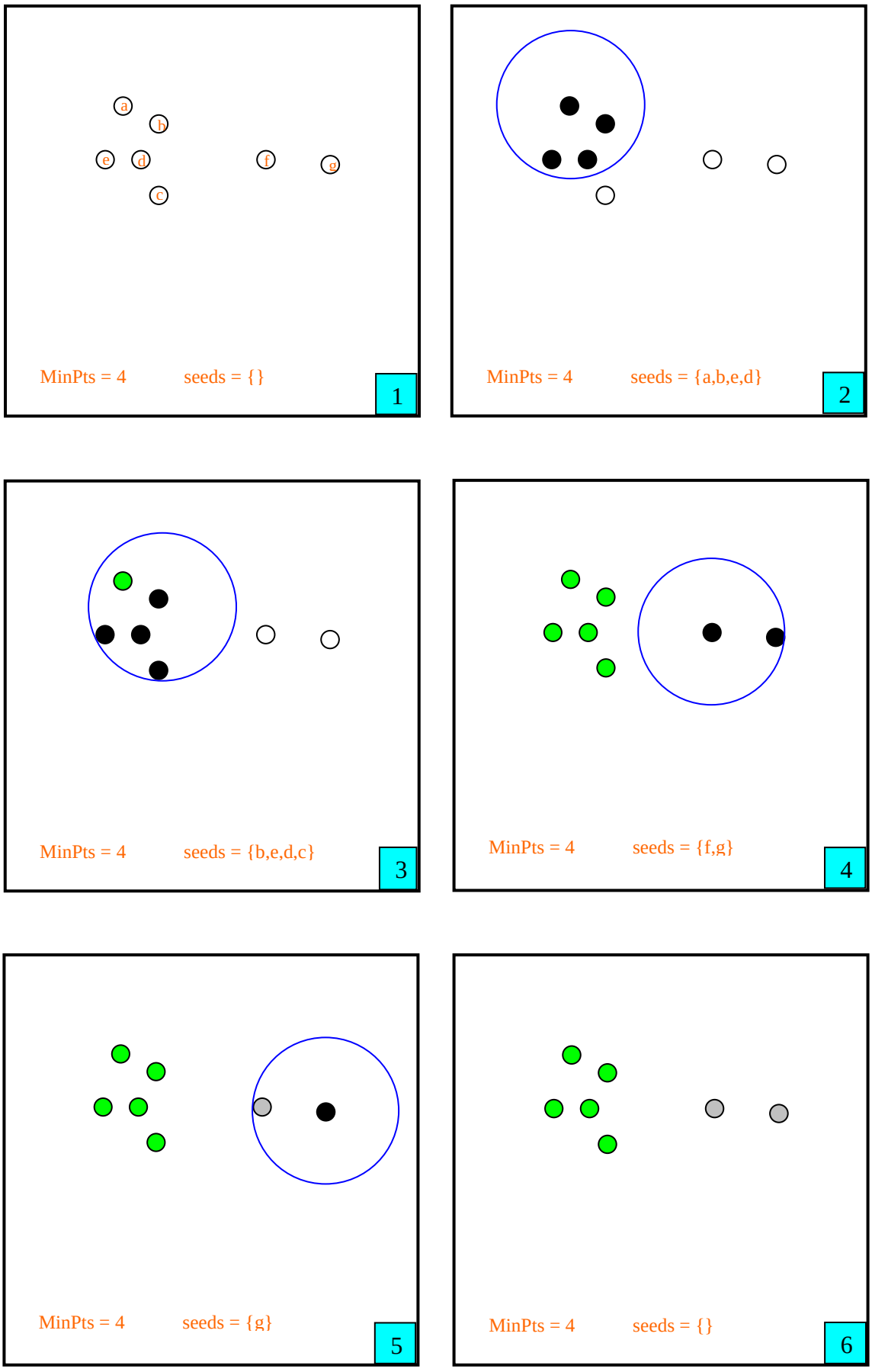


Abbildung 26: Ein Beispiel für den Ablauf des DBSCAN Algorithmus mit MinPt=4

Bestimmung der Parameter ε und MinPts

Bevor der DBSCAN Algorithmus für die vorliegenden Schnipsel zum Einsatz kommt, müssen vorerst geeignete Werte für die beiden Parameter *MintPts* und *ε* bestimmt werden.

Zu diesem Zweck wurde (16) in eine Heuristik vorgestellt. Diese geht aus der Eigenschaft der k-nächsten Nachbarn hervor. Sei *d* der Abstand eines Punktes *p* zu seinem k-nächsten Nachbarn, so befinden sich in der d-Nachbarschaft von *p* genau *k* Punkte. Dies gilt jedoch nicht, wenn mehrere Punkte exakt den gleichen Abstand zu *p* haben, was im Allgemeinen eher die Ausnahme darstellt. Nun wird eine Funktion *k-dist* definiert, die für jeden Punkt die Distanz zu seinem k-nächsten Nachbarn bestimmt. Danach werden alle Punkte bzgl. ihres *k-dist* Wertes absteigend sortiert und in einem Koordinatensystem dargestellt. Diese Darstellung gibt einige Hinweise über die Verteilung der Dichte in der Menge der Punkte, wie aus Abbildung 27 zu entnehmen ist.

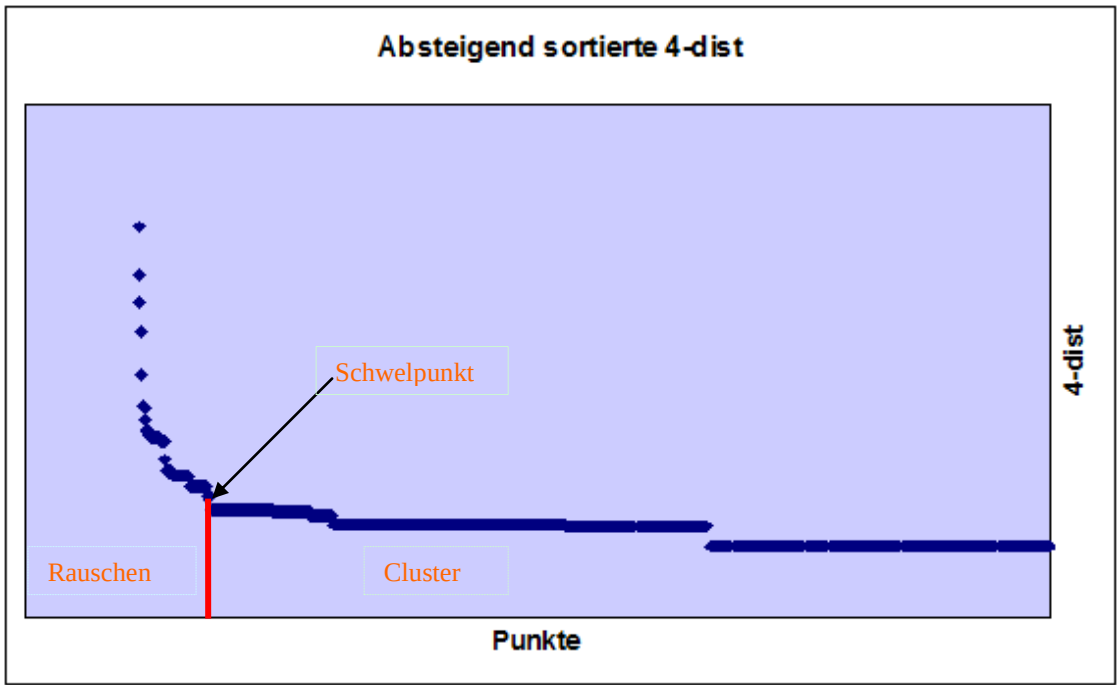


Abbildung 27: Absteigende sortierte 4-dist Funktion für den Schnipsel in Abbildung 29b

Das Diagramm in Abbildung 27 beschreibt für einen zuvor festgelegten Wert *k*, hier ist *k* gleich vier, wie nahe der k-nächste Punkt an einem beliebigen Punkt gelegen ist. Gesucht ist nun ein Schwelpunkt, dessen Wert ermöglicht, die Punkte, die zum Rauschen gehören, von den übrigen Punkten zu trennen. Dieser Wert ist nicht anders als der erste Parameter *ε*. Also der maximale *k-dist* Wert eines Punktes in dem dünnsten Cluster. Dieser Punkt ist der erste Punkt im ersten „Tal“ im sortierten *k-dist* Diagramm. Punkte mit größerem *k-dist* Wert, die links im Diagramm zu sehen sind, werden somit als Rauschen klassifiziert. Während alle anderen Punkte, die rechts im Diagramm zu sehen sind, zu den verschiedenen Clusters hinzugefügt werden.

Nun besteht die Schwierigkeit darin, ein geeignetes *k* zu finden, welches dem zweiten Parameter *MinPts* entspricht. In (16) wird ein Wert *k=4* vorgeschlagen, da in Experimenten keine signifikanten Unterschiede zu größeren Werten für *k* festgestellt werden konnten. Dieser Wert gilt nur für zwei dimensionale Daten.

Ergebnisse vom DBSCAN Algorithmus auf Schnipselbilder

Vorerst werden die Schnipselbilder in einem Vorschritt binarisiert, sodass dadurch ein Binärbild erzeugt wird. Die Pixel dieses Binärbildes haben zwingend entweder den Wert 0 oder 1. Die Vordergrundpixel eines Schnipselbildes stellen die Punktemenge dar, die als Eingabe für den DBSCAN Algorithmus benötigt werden. Die Features der einzelnen Punkte entsprechen der X und Y Koordinaten der Vordergrundpixel. Der Wert vom Parameter *MinPts* wird gleich vier festgelegt. Die Mahalanobis-Metrik wird zur Messung der Distanz zwischen den Pixeln eingesetzt.

Zur Motivation der Mahalanobis Metrik sei die Abbildung 28 mit zwei identischen Punkteschwärmen betrachtet. Nach der euklidischen Metrik liegen alle Punkte, die von einem festen Bezugspunkt denselben Abstand haben, auf einem Kreis mit diesem Bezugspunkt als Mittelpunkt und einem Radius von der Größe der euklidischen Distanz. Demnach seien die beiden Punkte (-3,-3) und (+3,-3) im Sinne der euklidischen Distanz äquivalent. In qualitativer Hinsicht ist der durch den Dreieck markierte Punkt (+3,-3) jedoch anders zu beurteilen als der Punkt (-3,-3). Letztere liegt nämlich in Richtung der langen Hauptachse des ellipsenförmig verteilten Punkteschwarms und fällt nicht aus dem Rahmen der positiven Korrelation der beiden Variablen, während der Punkt (+3,-3) als Ausreißer aufzufassen ist, da er hinsichtlich der Korrelation anormal liegt. Die Mahalanobis Metrik bezieht über die Kovarianzmatrix die Korrelationsstruktur der Features in die Entfernungsmessung ein. Im Sinne dieser Metrik liegen die Punkte gleicher Entfernung vom Schwerpunkt auf einer Ellipse, die bei gleicher Varianz und Unkorreliertheit der Variablen zu einem Kreis wird (3) Die Punkte repräsentieren nichts anderes als die Vordergrundpixel eines Schnipselbildes, somit stellt der Schwerpunkt ein Kernobjekt und der Ausreißer ein Punkt, welcher zum Rauschen gehört, dar.

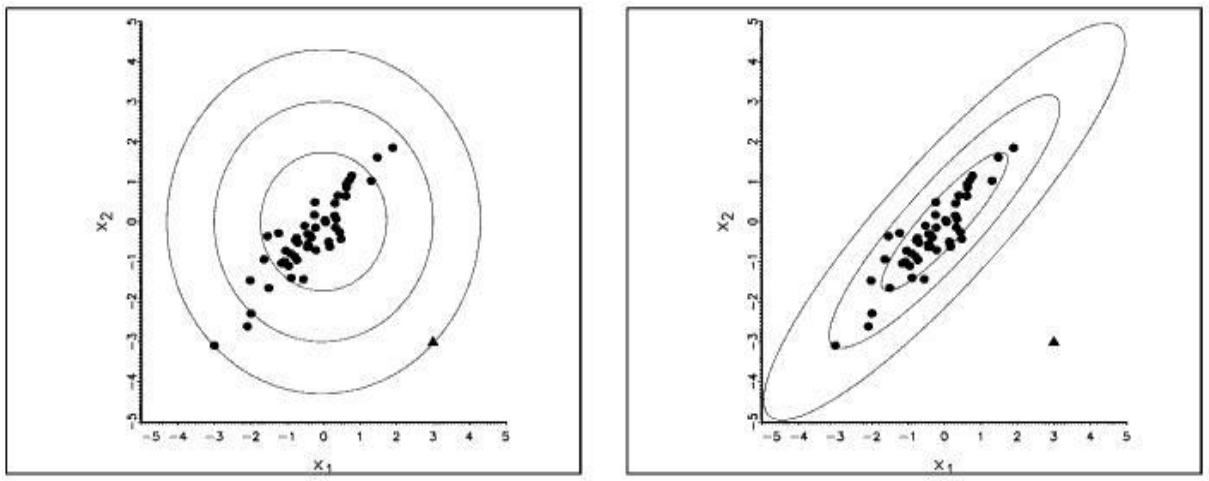


Abbildung 28 aus (3): links, Euklidische Distanz zum Mittelwert. rechts, Mahalanobis-Distanz zum Mittelwert

Zum Schluss wird der Wert vom Parameter ε berechnet und festgelegt. Problematisch hierbei ist jedoch die Durchführung der Berechnung für jedes Schnipselbild sowie das Ablesen des Wertes des *k-dist* „Tals“ aus dem dazugehörigen Diagramm, wie in der Abbildung 27 veranschaulicht wird. Außerdem lässt sich dieses „Tal“ im Allgemeinen nicht einfach automatisch finden. Aus dieser Tatsache wird abgeschätzt, wie viel Prozent der Punkte zum Rauchen gehört (16). Anhand dieser Berechnung wird der Wert vom Parameter ε festgelegt.

Die Abbildung 30 zeigt ein Ergebnisbeispiel für die Anwendung des DBSCAN Algorithmus auf einem binarisierten Schnipselbild (Abbildung 29b). Die Pixel, die zu einem Cluster gehören, sind blau markiert, während die zum Rauschen gehörenden Pixel gelöscht werden. Die grün markierten Rechtecke veranschaulichen die Grenze der einzelnen gefundenen

Cluster und bilden eine konvexe Hülle für die Pixel mit derselben *ClusterId*. Diese Rechtecke wurden ebenso für nachfolgende Schritte mit der jeweiligen *ClusterId* verseht. Im weiteren Verlauf dieser Arbeit wird die Bezeichnung „Bounding Box“ für diese Rechtecke verwendet. Somit bezeichnet ein „Bounding Box“ einen rechteckigen Bereich, welcher ausschließlich Pixel mit derselben *ClusterId* enthält.

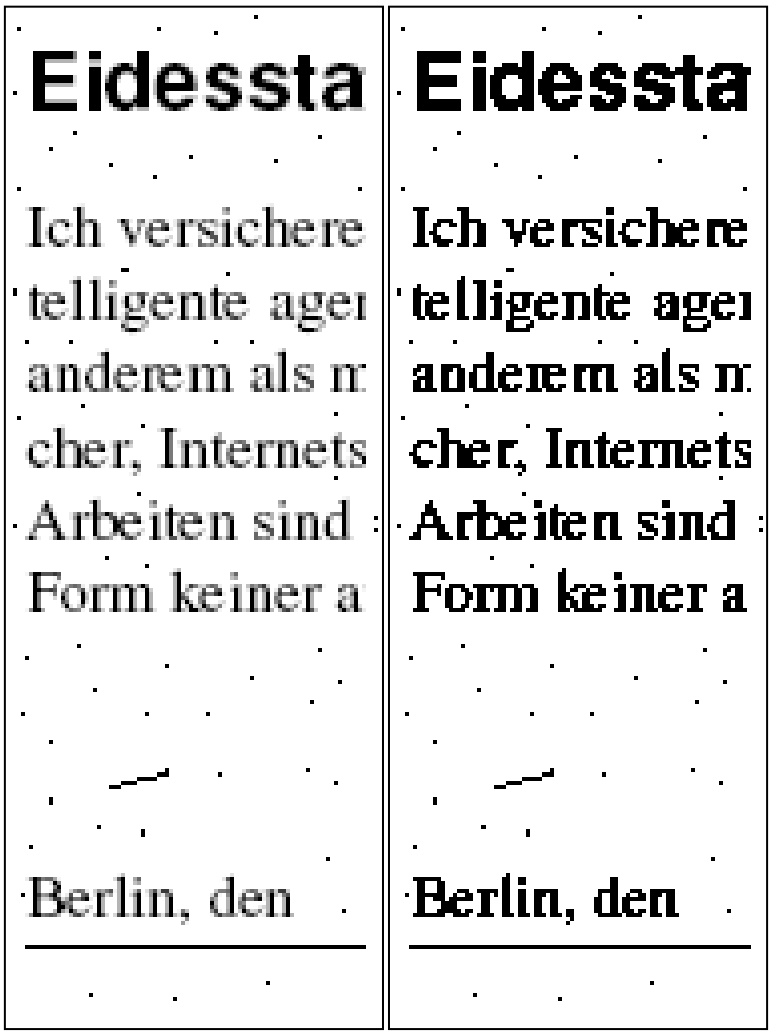
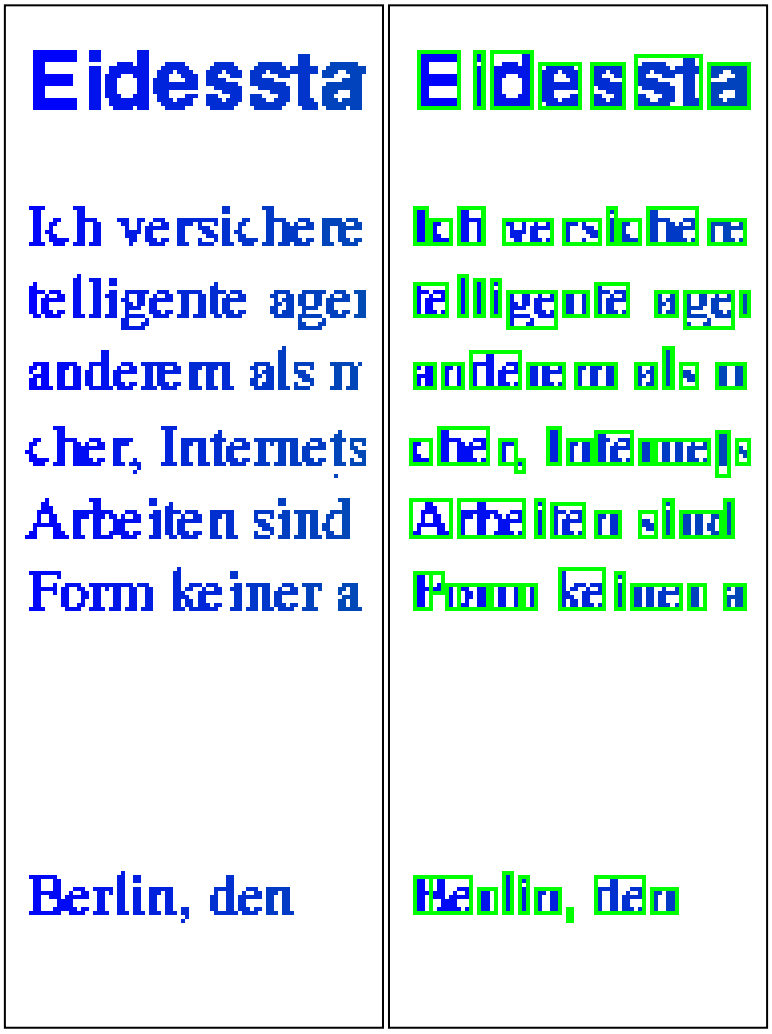


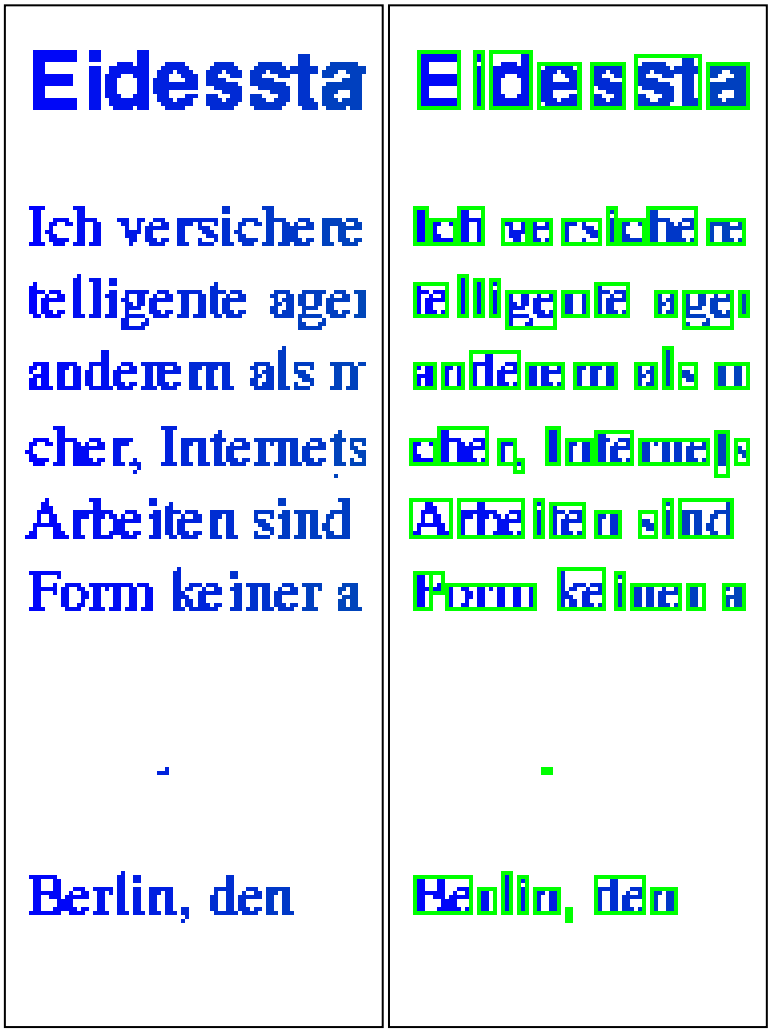
Abbildung 29a: Ein originales Schnipselbild **Abbildung 29b: Das Schnipselbild binarisiert**

In der Abbildung 30a wurde *MitPts* gleich vier und ε gleich 0,06 festgelegt, wobei ε aus dem Diagramm in der Abbildung 27 abgelesen wird. Der Algorithmus konnte 76 Cluster finden. Diese sind in Abbildung 30b deutlicher zu sehen. Zum Vergleich wurde der Algorithmus auch mit den Werten *MinPts* gleich 4 und ε gleich 0.05 ausgeführt. In diesem Fall wurden 72 Cluster gefunden, welche in der Abbildung 31b dargestellt sind.



(a)Ergebnisbeispiel des DBSCAN (b): Die grenzen der einzelnen gefunden Cluster

Abbildung 30: Ergebnisbeispiel des DBSCAN Algorithmus aufm Schnipsel in Abb.29b. Wobei MinPts=4 und *ε=0.06*



(a)Ergebnisbeispiel des DBSCAN (b): Die grenzen der einzelnen gefunden Cluster

Abbildung 31: Ergebnisbeispiel des DBSCAN Algorithmus aufm Schnipsel in Abb.28b. Wobei MinPts=4 und *ε=0.05*

$M = \{B_i | \, i = 1..n \, \wedge \, B_i \, \in S\}$ sei eine Menge aus den erzeugten Bounding Boxen auf einem Schnipsel *S*. Gesucht wird eine Menge $m \subset M$,wobei $m = \{B_i | \, B_i \, \in M \, \wedge \, B_i \, ist \, ein \, Rand - Cluster\}$. Mit Rand-Cluster ist ein Bounding Box gemeint, welches direkt an einer Kontur des Schnipsels *S* liegt.

Entscheidend ist, ob ein bereits klassifiziertes Bounding Box relevante Informationen für die spätere Projektion auf den Konturen wiedergibt. Die Projektion spielt eine wichtige Rolle beim Zusammensetzen von zwei oder mehreren Schnipseln. (mehr dazu im Abschnitt 3.3.3). Aufgrund dieser Tatsache führt dieser Vorgang zur Reduzierung der Anzahl der zu klassifizierenden Bounding Boxen und den damit verbundenen Klassifizierungsaufwand. Abbildung 32 veranschaulicht diejenigen Bounding Boxe, welche von der Klassifizierung betrachtet wurden. Dies sind nämlich diejenigen, die rot markiert sind, und in einer unmittelbaren Nähe der Schnipselkontur liegen. Obwohl die blau markierten Bounding Boxen sehr nah an der Schnipselkontur liegen, sind sie für die Klassifizierung uninteressant, genau wie die grau markierten, denn Sie geben keine relevant wichtige Informationen für die Projektion, welche für das Zusammensetzen von mehreren Schnipseln eine große Rolle spielt.

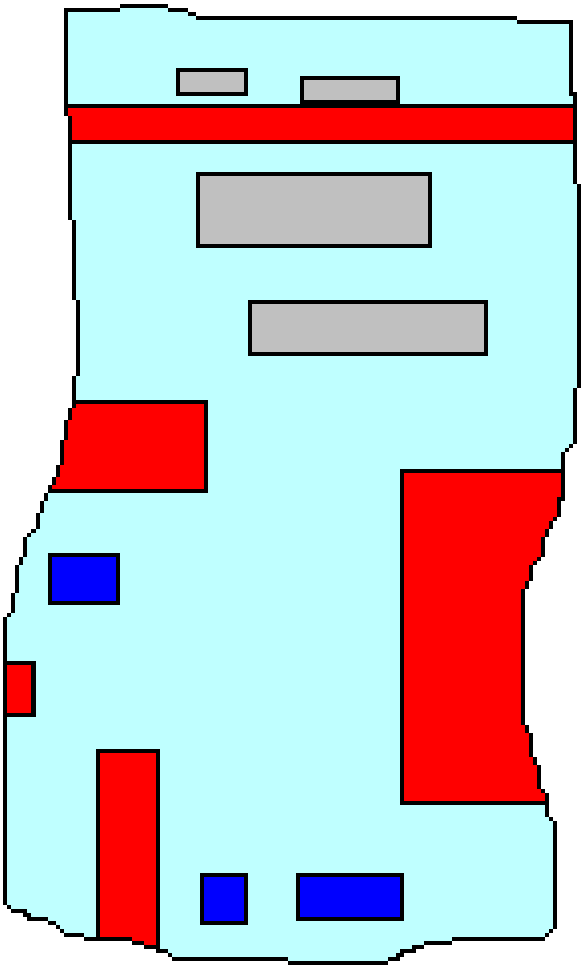


Abbildung 32: Rand-Cluster auf einem Schnipsel

3.3.2 Die Klassifikation

In diesem Schritt werden die im vorherigen Schritt erzeugten Bounding Boxen in den beiden Kategorien Bild und Text klassifiziert. Zu diesem Zweck wurde ein Klassifizierungsalgorithmus entwickelt. Interessant hierbei sind jedoch nicht alle Bounding-Boxen auf einem Schnipsel, sondern diejenigen, die in der Menge *m* enthalten sind.

$$m = \{B_i | \, B_i \in M \ \wedge \ B_i \, ist \, ein \, Rand - Cluster\}$$

Zunächst wird nach einem kurzen Überblick die Problemstellung geschildert. Anschließend wird die Extraktion von Features erläutert und abschließend die Klassifikationsverfahren genauer bestimmt.

Überblick

Unter den Oberbegriff „Zone Klassifikation“ fällt der in dieser Arbeit entwickelte Klassifizierungsalgorithmus, welcher als wichtiger Teilprozess der „Document Image Processing“ betrachtet wird.

In (17) wird das Zonen Klassifikationsproblem als maschinelles Lernproblem betrachtet. Die 13831 Zonen aus der „English Document Image database“ von der Universität von Washington -UW-Database I- wurden in drei Klassen unterschieden: Text, Druckraster und Bild. Basierend auf die Connected Component-Analyse und Run-Length stellten die Autoren sieben Features zusammen, und erreichten unter der Verwendung des C.4.5 Entscheidungsbaum eine Fehlerrate von 6,7%.
Weitere verwendete Features sind unter anderem die Kreuz-Korrelation zwischen den Scan-Zeilen (10), wobei die Fischer Diskriminante zur Klassifizierung eingesetzt wurde. Auch hier wurde zwischen den drei Klassen: Text, Bild und Diagramme, unterschieden.
Die Autoren von (11) präsentierten einen der detailliertesten Überblicke über die Entwicklung in der Zonen Klassifikation. Zusätzlich wurde ein Algorithmus vorgestellt, welcher mit einem optimierten Entscheidungsbaum insgesamt 24117 Zonen aus der UW-Database III in neuen verschiedenen Klassen klassifizierte. Diese Klassen sind unteranderem Text, Bild, Mathe, Linie und Table. Jede einzelne Zone wurde durch einen 25 dimensionalen Features Vektor repräsentiert. Run-length-, Autokorrelation- und räumliche Features sind in diesen Vektoren präsent. Die Fehlerrate des Algorithmus lag bei 1,55%.

Problemstellung

M sei eine Menge von Bounding Boxen auf einem Schnipsel. Zudem sei *L* ebenso die Menge der Klassenlabels, also Text und Bild. Die Funktion $f: M \rightarrow L$ weist jeder Bounding Box aus *M* einen Label zu. Sei *N* ein Wertbereich, dann beschreibt die Funktion $v: M \rightarrow N$ den Wert für jeden Element aus *M.* Wobei *v(m)* der generierte Featuresvektor für $m \in M$ ist. Infolgedessen kann das Bounding Box Klassifikationsproblem wie folgt formuliert werden: Gegeben ist eine Menge von Bounding Boxen *M* und eine Menge von Klassenlabels *L;* gesucht wird eine Klassifikationsfunktion $f: M \rightarrow L$, welche die maximale bedingte Wahrscheinlichkeit $P(f(M) \mid v(M))$ erfüllt .

Features Extraktion

Die Grundidee besteht darin, zu jeder Bounding Box eine Reihe von Merkmalen bzw. Features zu berechnen und diese in einem Vektor zu speichern. Der dadurch entstehende Featuresvektor wird dann verwendet, um dessen Bounding Box ein Klassenlabel zuzuweisen. Insgesamt werden 15 Features extrahiert. Diese stammen aus drei Gruppen, Run-length-, Kreuzkorrelation- und räumliche Feature. Hierbei werden sowohl Hintergrund- (weiße Pixel) als auch Vordergrundpixel (schwarze Pixel), beachtet. Die folgenden Features werden von jeder Bounding Box extrahiert:

a) *Räumliche Features:* Die ersten zwei Features beschreiben den sogenannten Füllverhältnis in einer Bounding Box und werden mit *b/w* bzw. *b/t* bezeichnet. *b/w* gibt das Verhältnis zwischen der Anzahl der schwarzen Pixel zu der Anzahl der weißen Pixel wieder. Während *b/t* dem Anteil der schwarzen Pixel in dem Bounding Box entspricht.
   b/w = #Vordergrundpixel / #Hintergrundpixel (1)
   b/t = #VordergrundPixel / #totale Pixel (2)

b) *Run-length Features:* Ein *Run* ist eine Liste von zusammenhängenden Hintergrund- bzw. Vordergrundpixel in einer bestimmten Richtung. Run-length bezeichnet die Anzahl der Pixel in einem *Run* (18). In dieser Gruppe werden die Anzahl von den *Runs*, der Mittelwert und die Varianz von den Run-length ermittelt. Diese Berechnungen erfolgen sowohl auf horizontaler als auch auf diagonaler sowie für die Hintergrund- als auch für die Vordergrundpixel.

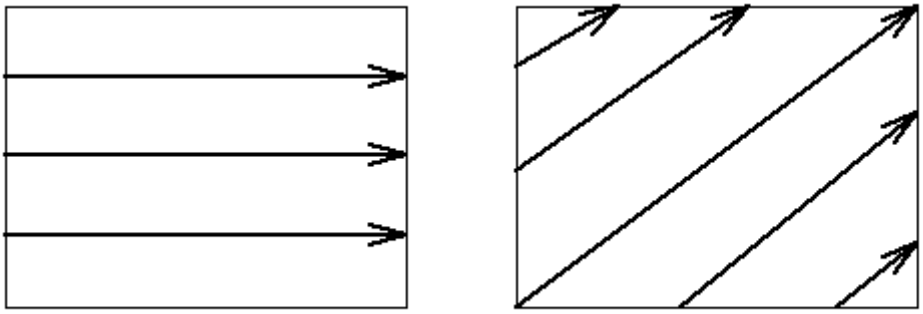


Abbildung 33: veranschaulicht die Richtungen der Extraktion von den Run-length Features. Horizontale Richtung wird mit h, und diagonale Richtung mit d bezeichnet

Mit folgendem Beispiel wird die Extraktion von Runs dargestellt: In einer Bounding Box I wird ein Hintergrundpixel mit 0 und ein Vordergrundpixel mit 1 bezeichnet.

$$I = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Für die zwei ausgewählten Richtungen wird eine Run-length ($N_g \times N_r$) Matrix definiert (18). N_g ist gleich zwei und entspricht der Anzahl der in der Bounding Box vorkommenden Pixelwerte {0,1}.Während N_r die mögliche maximale Länge eines Runs entspricht Im vorliegenden Beispiel ist N_r gleich vier. In horizontaler Richtung

$$Q_{RL}(h) = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{bmatrix}$$

Die erste Zahl in der ersten Zeile der Matrix $Q_{RL}(h)$ bezeichnet, wie oft das Hintergrundrun der Länge eines in einem Bounding Box vorgekommen ist - im vorliegenden Beispiel gleich 3-. Ebenso bezeichnet die zweite Zahl, wie oft ein Paar Hintergrundpixel vorgekommen ist, also im vorliegenden Beispiel gleich 1 usw.

Die Zahlen in der zweiten Zeile der Matrix $Q_{RL}(h)$ geben dieselben Informationen, jedoch für Vordergrundruns. Für die diagonale Richtung gilt:

$$Q_{RL}(d) = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 2 & 3 & 0 & 0 \end{bmatrix}$$

Basierend auf die oben erwähnten Definitionen werden die folgenden acht Features wie folgt berechnet:

|*RLh0*| = # der horizontale Hintergrundruns (3)
|*RLh1*| = # der horizontale Vordergrundruns (4)
|*RLd0*| = # der diagonalen Hintergrundruns (5)
|*RLd1*| = # der diagonalen Vordergrundruns (6)

Der Mittelwert der Run-length in den beiden Richtungen und für die Hintergrund- und Vordergrundpixel werden wie folgt berechnet:

$$RLmeanh0 = \frac{1}{|RLh0|}\sum_{rl\in RLh0} rl \quad (7)$$

$$RLmeanh1 = \frac{1}{|RLh1|}\sum_{rl\in RLh1} rl \quad (8)$$

$$RLmeand0 = \frac{1}{|RLd0|}\sum_{rl\in RLd0} rl \quad (9)$$

$$RLmeand1 =. \frac{1}{|RLd1|}\sum_{rl\in RLd1} rl \quad (10)$$

Die Varianz der Runlength in den beiden Richtungen und für die Hintergrund- und Vordergrundpixel werden wie folgt berechnet (11):

$$RLvarh0 = \frac{\sum_{rl\in RLh0} rl^2}{|RLh0|} - \left(rlmeanh0\right)^2 \quad (11)$$

$$RLvarh1 = \frac{\sum_{rl\in RLh1} rl^2}{|RLh1|} - \left(rlmeanh1\right)^2 \quad (12)$$

$$RLvard0 = \frac{\sum_{rl\in RLd0} rl^2}{|RLd0|} - \left(rlmeand0\right)^2 \quad (13)$$

$$RLvard1 =. \frac{\sum_{rl\in RLd1} rl^2}{|RLd1|} - \left(rlmeand1\right)^2 \quad (14)$$

Für das betrachtete Beispiel, angefangen beim Feature mit der Nummer (3) in der gleichen Reihenfolge bis Nummer(14) ergeben sich die folgenden Werte:

|RLh0|	*|RLh1|*	*|RLd0|*	*|RLd1|*	*RLmeanh0*	*RLmeanh1*
3	3	3	2	1,66	1,33
RLmeand0	***RLmeand1***	***RLvarh0***	***RLvarh1***	***RLvard0***	***RLvard1***
1,66	2,5	0,88	0,22	0,88	0,25

c) *Kreuzkorrelation Features*: In der Signalanalyse wird die Kreuzkorrelation zur Beschreibung der Korrelation zweier Signale *f* und *g* eingesetzt. Es gilt für binäre Signale, deren Werte aus der Menge {0,1} sind (10).

$$C\,(f,g) = 1 - 2\,(f\ XOR\ g)$$

Infolgedessen wird die normalisierte Kreuzkorrelation zwischen zwei Zeilen *z* und *z+r* in einer Boundnig Box als *C(r,z)* definiert. Es gilt:

$$C(r,z) = \frac{1}{L}\sum_{k=0}^{L-1}\left[1 - 2\,p(z,k)\ XOR\ p(z+r,k)\right]$$
$$= 1 - \frac{2}{L}\sum_{k=0}^{L-1} p(z,k)\ XOR\ p(z+r,k)$$

wobei $L=|z|$ die Anzahl der Pixel in der Zeile *z* bezeichnet und *p(z,k)* den Wert des k-ten Pixels in der Zeile *z* entspricht (10).
Der Wertebereich von *C* liegt zwischen {-1,1}.Somit erhalten wir die maximale Korrelation *C(r,z) = 1* , wenn *r = 0*, also die Korrelation einer Zeile mit sich selbst oder , wenn $p(z,k) = p(z+r,k) : \forall k \in \{0, L-1\}$ gilt.
In einer Bounding Box, welche nur Text enthält, zeigen die Zeilen eine größere Ähnlichkeit als jene, die nur Bild enthalten und somit auch einen höheren Korrelationswert. Aus dieser Beobachtung wird das letzte Feature definiert. Dies ergibt sich aus dem Verhältnis der Anzahl der Zeilen mit $C(1,z) > 0{,}9$ zu der totalen Anzahl der Zeilen in einer Bounding Box (10).
$C(1,z) > 0{,}9/T$ wobei *T* die Höhe der Bounding Box angibt (15)

Das Resultat der Features Extraktion ist eine Menge von Featuresvektoren. Diese Vektoren werden im nächsten Schritt klassifiziert. Jeder Vektor repräsentiert genau eine Bounding Box und enthält insgesamt 15 Features.

Der KNN Klassifikator

Zur Klassifizierung der Featuresvektoren wird der Nächste-Nachbar-Klassifikator eingesetzt, welcher ein mathematisch deutlich einfaches Klassifikationsverfahren ist. Dieser Klassifikator ordnet ein zu klassifizierendes Objekt mit dem Merkmalsvektor *x* derjenigen Klasse zu, zu der das Objekt mit dem ähnlichsten Merkmalen im Trainingsdatensatz gehört (19). Der nächste Nachbar *y* von *x* ist dabei bestimmt durch

$$d(x, y) = \min_{n=1..N} d(x, y_n)$$

wobei *d* ein Distanzmaß darstellt. Hier kommt die euklidische Metrik zum Einsatz. Eine naheliegende Erweiterung der Nächsten-Nachbar-Klassifikator ist der k- Nächsten-Nachbar-Klassifikator, bei dem nicht nur der nächstgelegene Merkmalsvektor im Trainingsdatensatz berücksichtigt wird, sondern die *k* am nächsten bei dem Merkmalsvektor *x* liegenden Merkmalsvektoren des Trainingsdatensatzes (19).
Der Trainingsdatensatz bzw. die Referenzdatenbank, besteht aus 100 Featuresverktoren, welche 100 unterschiedliche Bilder repräsentieren. Diese Bilder sind alle binarisiert und untergeteilt in folgende Gruppen:

- 40 unterschiedliche Bilder in verschiedenen Größen, darunter auch Logos,
- 26 Bilder für die 26 Buchstaben im Alphabet, sowie
- 10 Bilder für die Zahlen 0 bis 9 und
- 24 Bilder für willkürlich erzeugte Textblöcke, welche ein oder mehrere Wörter, ebenso ein oder mehrere Zeilen enthalten können.

Aus diesen Referenzbildern werden die 15 obengenannten Features extrahiert. Jeder der 100 erzeugten Featursvektoren ist mit seinem Klassenlabel versehen. Diese zusammen bilden die Referenzdatenbank.

3.3.3 Die Projektion

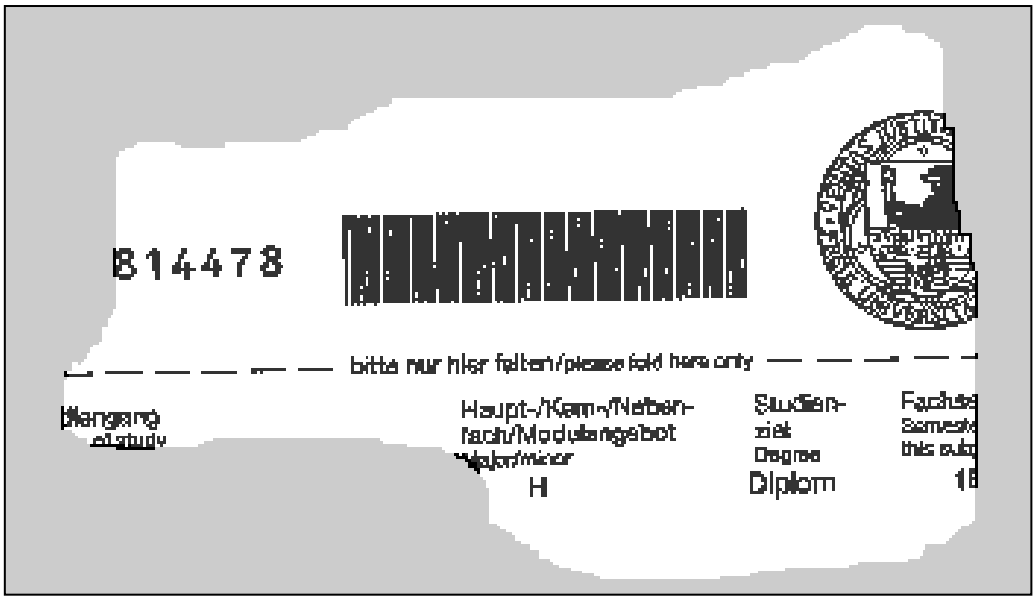
Die Projektion stellt den Abschluss des Vorkleben-Prozesses dar. Sie bezeichnet die Abbildung des Ergebnisses der Klassifizierung auf die Schnipselkonturen. Das Ziel liegt darin, das geschilderte Szenario in *3.3* zu vermeiden. Dies könnte bei einer großen Anzahl von Schnipseln zu besseren Konstruktionsergebnissen führen. Die Ergebnisse der Projektion könnten unterschiedliche Formen haben, denn diese Formen hängen weitgehend von den Algorithmen ab, die diese Ergebnisse zur Rekonstruktion verwenden. Die Folgende Menge stellt eine mögliche Lösung vor.

$$P = \{\, c, k, (x_i\,, y_i\,, h_i\,, w_i\,, k_j) \,|\, c = \|C\| \,\wedge k = \|Kl\| \,\wedge i = 1..c \,\wedge j = 1..k\},$$

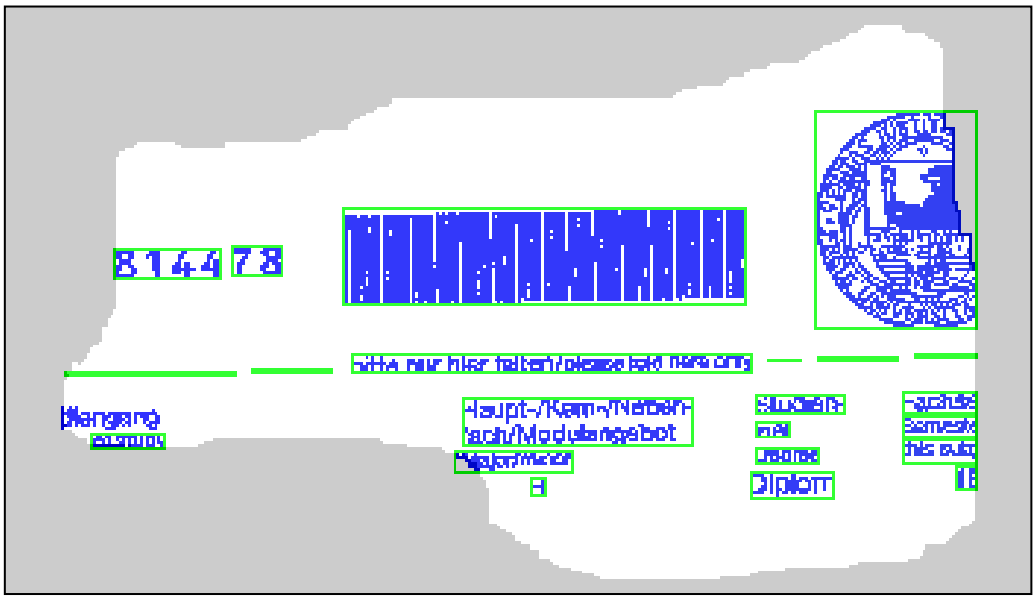
wobei *C* die Menge der klassifizierten Rand-Cluster und *Kl* die Menge der gegebenen Klassenlabels bezeichnen. Ebenso stehen $x_i\,, y_i\,, h_i\,, w_i$ für die x-, y-Koordinaten, Höhe und Bereite des i-ten Bounding Box. k_j bezeichnet das j-te Klassenlabel.

3.3.4 Auswertung des (V)orkleben Prozesses

In diesem Abschnitt werden Experimente mit den oben beschriebenen Methoden des Vorkleben Prozesses betrachtet und Ergebnisse vorgestellt sowie evaluiert. Als Begleitbeispiel werden zwei identische Schnipselbilder verwendet, die sich durch Rauschen voneinander unterscheiden. Wobei Abbildung 34a das Schnipselbild ohne Rauschen darstellt. Während Abbildung 35a dasselbe Schnipselbild jedoch mit Rauschen illustriert. Für alle Tests und Beispiele des DBSCAN Algorithmus wird die Mahalanobis Metrik als Distanzfunktion verwendet. Der *Parameter MinPts wird* gleich vier festgelegt.



(a)Originales Rauschfreies Schnipselbild

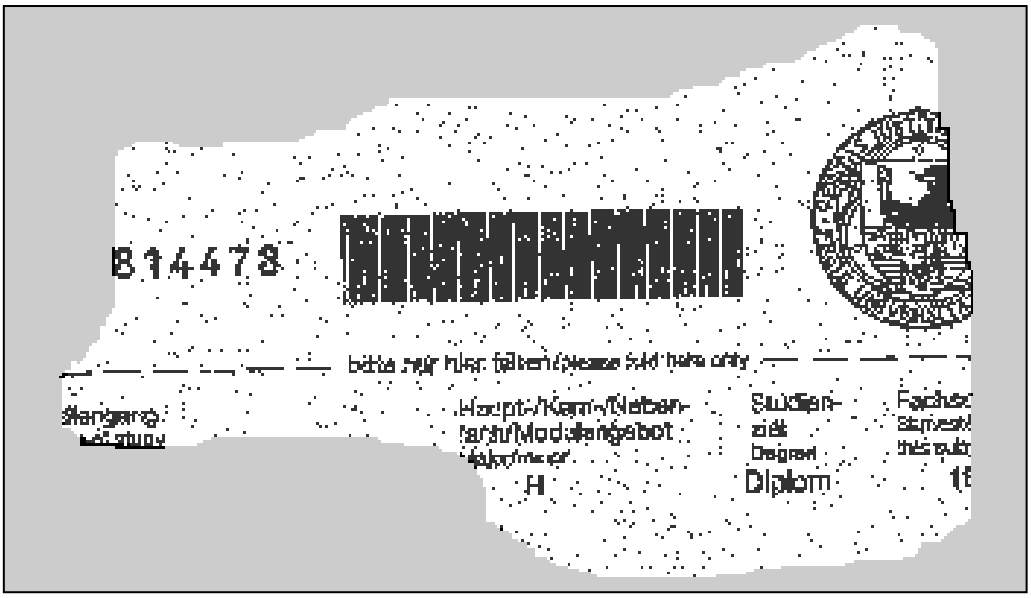


(b)Resultatbild mit 24 Cluster

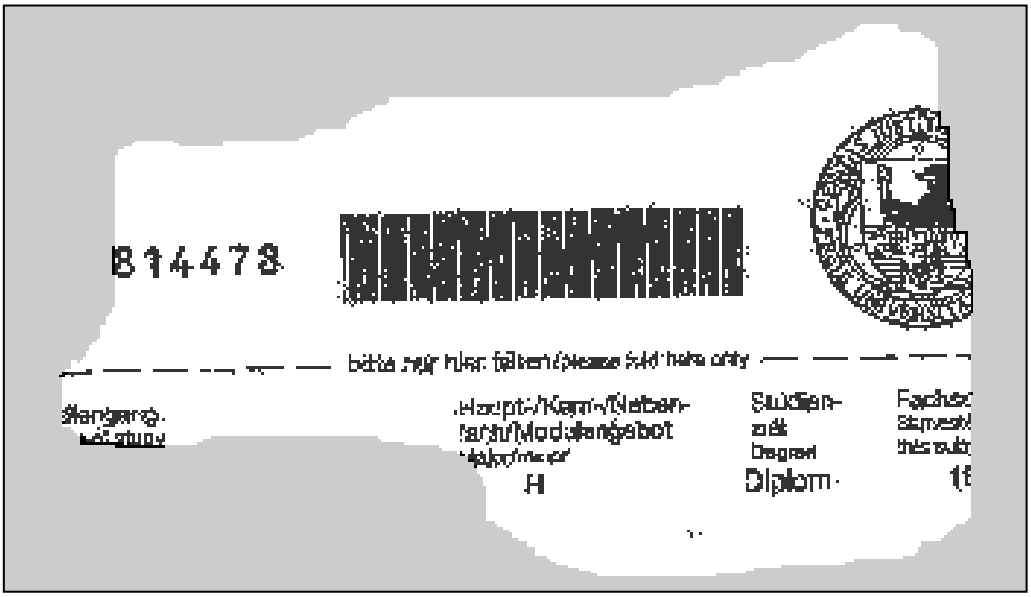
Abbildung 34: Das Ergebnis des DBSCAN Algorithmus auf einem Schnipselbild ohne Rauschen, wobei *MinPts*=4 und *ε=0.07*

Das in der Abbildung 34a dargestellte Schnipselbild wurde vorm Ausführen des DBSCAN Algorithmus binarisiert. Für das Clustern wurden 0,07 für ε als Wert eingesetzt. Das Ergebnis lässt sich aus der Abbildung 34b entnehmen. Entstanden sind insgesamt 24 Cluster. Tests auf eine Menge von 50 verschiedenen rauschfreien Schnipselbilder haben gezeigt, dass die besten Ergebnisse des Clusternprozesses bei $\mathcal{E} \in [0.07, 0.2]$ entstehen.

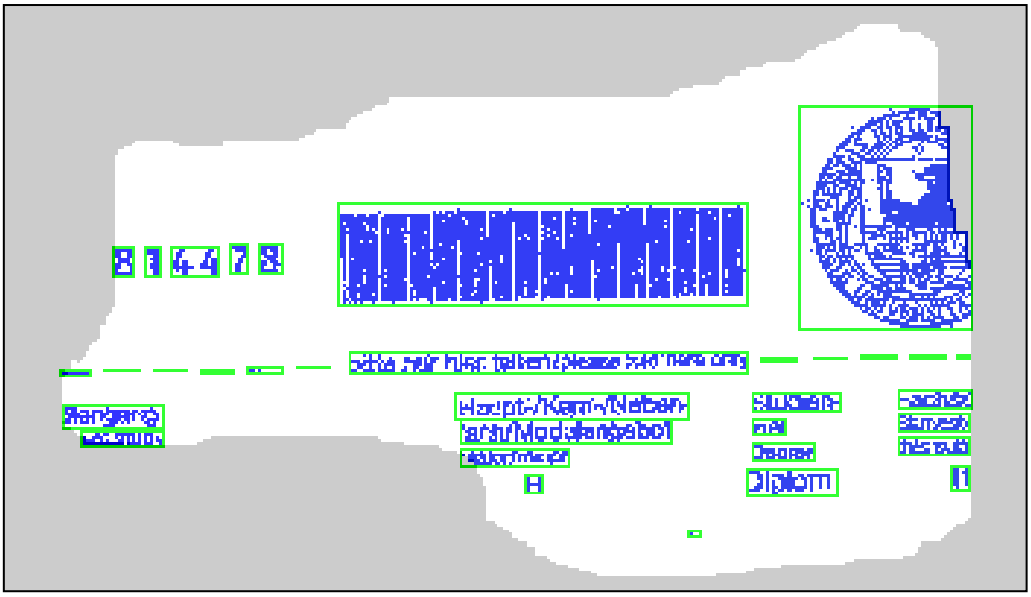
Der DBSCAN Algorithmus wurde ebenfalls auf dem Schnipsel in der Abbildung 35a eingesetzt. Bei einem Rauschbetrag von 10% haben Tests gezeigt, dass die besten Ergebnisse bei einem Wert von $\mathcal{E} \leq 0.045$ erzielt wurden. Das Resultatbild ist in der nachstehenden Abbildung 35zu sehen. Wobei die Beseitigung des Rauschens in Abb. 35a deutlich zu erkennen ist, während Abb. 35b. die entstandenen 34 Bounding Boxe demonstriert.



(a)Originales Schnipselbild mit 10% Rauschen



(b)Resultatbild

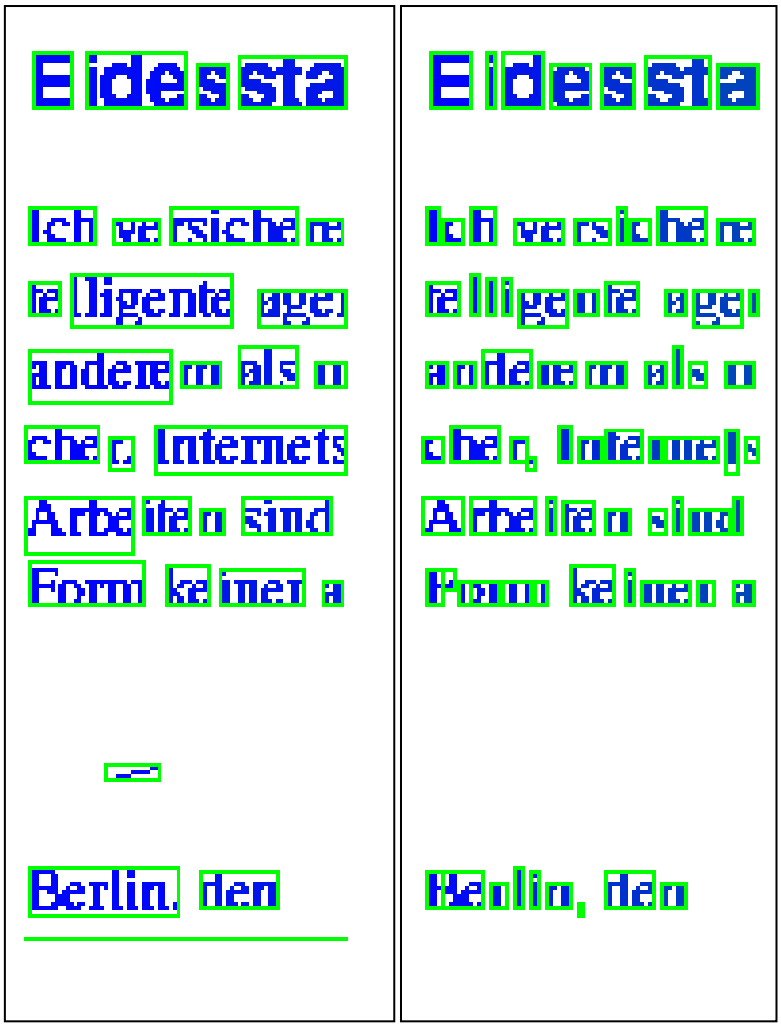


(c) Resultatbild mit 34 Cluster

Abbildung 35: DBSCAN Algorithmus auf einem Schnipselbild mit Rauschen, wobei *MinPts*=4 und $\varepsilon=0.04$

Obwohl es sich um dasselbe Schnipselsbild handelt, haben die Testergebnisse deutlich gemacht, welche negative Auswirkung das Rauschen auf den Clusterprozess ausübt.
Das „Überclustern" stellt das erste Problem dar. Damit ist die Entstehung von zusätzlich vielen kleinen Cluster gemeint, welche im Normalfall –ohne Rauschen- jedoch zusammen gehören. Dadurch sind auf dem Schnipselbild in der

Abbildung 35a zehn Cluster mehr als auf dem Schnipselbild in der Abbildung 34a entstanden. Die Anzahl der zu erkennenden Cluster in einem Schnipsel ist proportional zum prozentualen Anteil des Rauschens in einem Schnipsel. Dies führte zu einer erhöhten Laufzeit bei der Klassifizierung. Ein zweiter Durchlauf des DBSCAN Algorithmus auf dem Resultatbild wurde als eine Lösung für das „Überclustern" Problem vorgeschlagen. In diesem Fall wurden die bei einem rauschfreien Schnipselbild verwendeten Parameter eingesetzt. Zum Beispiel stellt das Schnipelbild in der Abbildung 34b ebenso ein Resultetbild dar, welches bei einem zweiten Durchlauf des DBSCAN Algorithmus auf dem Schnipselbild in der Abbildung 35b entsteht.
Das zweite Problem liegt darin, dass einige Vordergrundinformationen, die nicht zum Rauschen gehören, eventuell als Rauschen geclustert wurden, was zum Verlust von wichtigen Informationen bzw. Strukturen im Resultatbild geführt hat. Im allgemeinen Fall sind „dünne" Strukturen, zum Beispiel Linien, davon betroffen (vgl. Abbildung 31). In diesem Fall führt die Werterhöhung von ε zum Lösen dieses Problems, was bisher nicht ohne Weiteres möglich war, da in manchen Fällen kleine unabhängige Cluster entstanden sind, deren Inhalt eigentlich zum Rauschen gehört. Diese Cluster aus dem Rauschen wurden in Kauf genommen und bei der Klassifizierung ignoriert. Als Beispiel wird das Schnipselbild in der Abbildung 36 betrachtet, welches durch den DBSCAN Algorithmus auf dem Schnipselbild in der Abbildung 29b entstanden ist. Dabei wurde der Wert von ε auf 0.08 erhöht. Die Linie wurde in diesem Resultatbild als Cluster erkannt.



(a) Resultatbild mit von $\varepsilon=0.08$ (b) Verlustbehaftetes Resultatbild aus Abb.30b

Abbildung 36: Das Ergebnis des DBSCAN Algorithmus auf dem Schnipselbild in Abb 29b

Das Bestimmen der Rand-Cluster bewirkt die Reduzierung der Anzahl der Bounding Boxe, welche im nächsten Schritt klassifiziert werden. Dieser Schritt ist grafisch in der Abbildung 37 abgebildet.

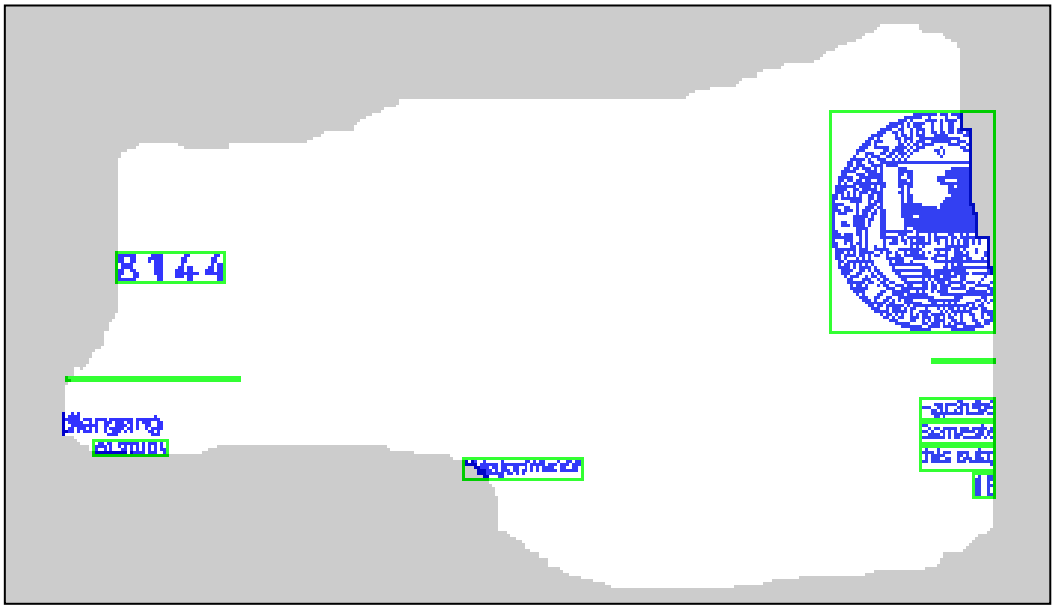


Abbildung 37: Rand-Cluster

Um die besten Ergebnisse bei der Klassifizierung der Bounding Boxen zu erzielen, wurden zwei Bedingungen aufgestellt. Diese spielten eine entscheidende Rolle bei der Auswahl der Features und bei der Erstellung der Referenzdatenbank. In diesem Kapitel wurde der Inhalt von Dokumentenschnipsel analysiert. Mit anderen Worten, es wird nicht von ganzen vollständigen Dokumenten ausgegangen. Hinzukommend wurde ein bestimmter Algorithmus zum Clustern des Inhalts eines Schnipsels eingesetzt und infolgedessen die Ausgabe dieses Algorithmus klassifiziert.

Die zwei Klassenlabels Text und Bild stellten die Zielklassen dar. Das Testen des Klassifikators wurde in drei Phasen durchgeführt. In jeder Phase wurde ein neuer Datensatz generiert. Der Datensatz bestand aus 20 Dokumenten und wurde mit Hilfe des Dokumenten-Generators erzeugt (vgl. 3.1.1) und nach dem Ansatz in 3.2.2 virtuell geschnipselt. Auf dieser Schnipselmenge wurde der DBSCAN Algorithmus angewendet und für jedes dadurch entstandenes Bounding Box ein Vektor mit den vorgeschlagenen Features berechnet. Diese Vektoren wurden einzeln durch die Mehrheitsentscheidung, welche durch die k-Nächste-Nachbarn aus der vorgeschlagenen Referenzdatenbank getroffen wurde, klassifiziert. Dabei wurde für jeden Wert von $k \in \{1,3,5,7,9\}$ getestet.

Ein großer Teil der fehlklassifizierten Bounding Boxen wurde durch einen unglücklichen Schnitt in den Dokumenten verursacht. Dies hat dazu geführt, dass einige grafische Objekte so geschnitten wurden, dass Teile davon als Text klassifiziert wurden (vgl. Abbildung 38). Ein anderer Grund stellte das Clusternprozess dar. Dieser Prozess hatte bei einigen Grafiken, die aus nicht zusammenhängenden Teilen bestehen, den Effekt gehabt, unabhängige Cluster zu bilden. Dabei bestand bei der danach folgenden Klassifizierung eine hohe Wahrscheinlichkeit, dass einige dieser neu entstanden Cluster nicht als Bild erkannt wurden. Daraus wird sichtbar, dass die Treffgenauigkeit der Klassifizierung anhand der beiden oben genannten Punkte variiert. Um den mit der Fehlklassifizierung entstandenen negativen Effekt zu reduzieren, wurde der Schnipseln-, Clustern- und Klassifikationprozess fünfmal bei jeder Testphase wiederholt. Diese Testphasen werden folgend näher erläutert:

- Die Dokumente in der ersten Testphase enthielten nur Zeichensätze. Durch den Einsatz des vorgeschlagenen Klassifikators hat sich eine durchschnittliche Fehlerrate von $fr \approx 0.05\%$ ergeben. Zum Schluss wurden die Featuresvektoren dieser Testphase zur Referenzdatenbank hinzugefügt.
- Es wurde ein neuer Datensatz generiert, wobei sich diesmal der Inhalt der Dokumente auf zufällig ausgewählte Logos, Bilder, und geometrische Figuren, welche aus dem Internet stammten, beschränkt hat. Mit dieser Gruppe wurde die zweite Testphase gestartet. Es wurde festgestellt, dass die Existenz von Zeichensätzen, was im Allgemeinen bei Logos der Fall ist, proportional zur Erhöhung der Fehlerrate beigetragen hat. Aus diesem Grund wurden nur Logos und Bilder ausgewählt, die keine Zeichensätze enthalten. Die Fehlerrate betrug in diesem Fall $fr \leq 15\%$ Abschließend wurden auch hier die Featuresvektoren zur Referenzdatenbank hinzugefügt.

- Der Inhalt der Dokumente für die letzte Testphase kann als eine Mischung aus den ersten beiden Testphasen betrachtet werden. So bestand der Inhalt jedes Dokumentes aus ≈50% Text und ≈50% Bild. Es wurde eine durchschnittliche Treffgenauigkeit von 94,9% erreicht.

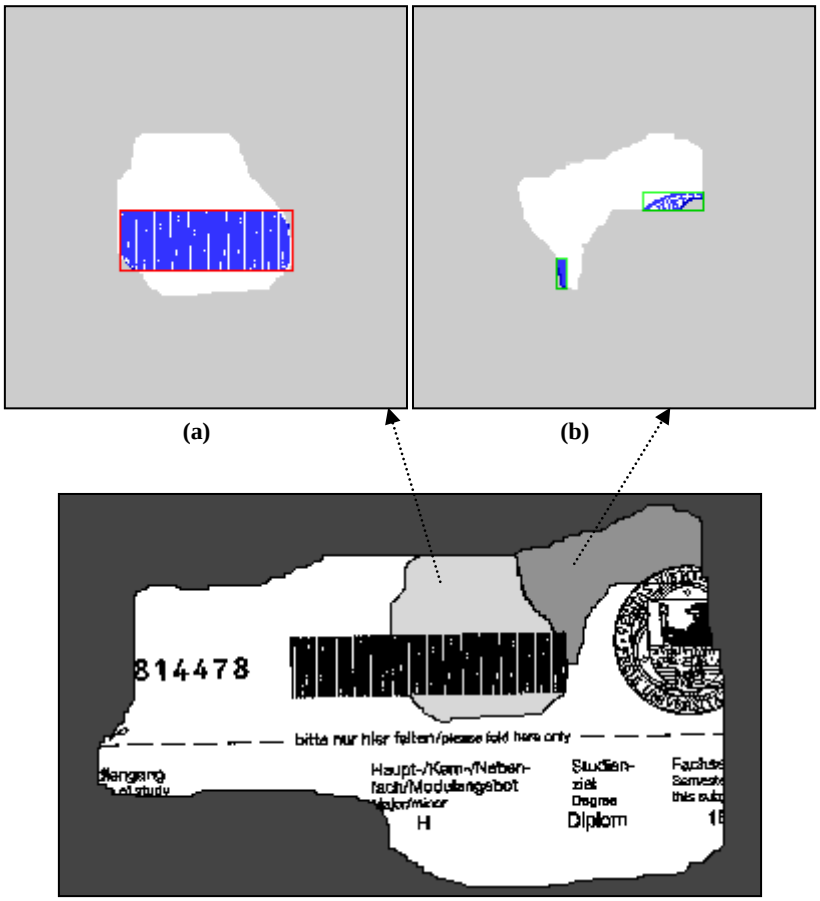


Abbildung 38: Fehlklassifizierung aufgrund einem unglücklichen Schnitt
(a) Bounding Box wurde als Bild klassifiziert (b) Bounding Boxe wurden als Text falsch klassifiziert

4. Kapitel

Das Schnipseljagd Framework

Für die in dieser Arbeit vorgestellten Algorithmen wurde ein Framework entwickelt. In diesem Kapital wird die graphische Benutzeroberfläche dieses Frameworks erläutert und deren Funktionalität beschrieben.

4.1 Die Benutzeroberfläche

Zunächst wird der Aufbau der Benutzeroberfläche beschrieben. Folgende Komponenten in der Abbildung 39 zu erkennen:

(1) Die Menüleiste, die mehrere Funktionen zur Verfügung stellt.
(2) Das Schnipsel-Fenster, welches sowohl die Dokumenten als auch die Dokumentenschnipsel grafisch darstellt.
(3) Das Eigenschaftenfenster, das die Informationen eines markierten Bildes zeigt. Dieses Bild wird im Schnipsel-Fenster markiert.
(4) Das Zielordnerfeld, welches bestimmt, in welchem Ordner die Ergebnisse eines Prozesses gespeichert werden.
(5) Das Modifikationsfenster, das einige Prozesse zur Modifizierung der Bilder zur Verfügung stellt.
(6) Der Schnipsel-Jagd Button, der das Prozessfenster öffnet, welches in der Abbildung 40 zu sehen ist.

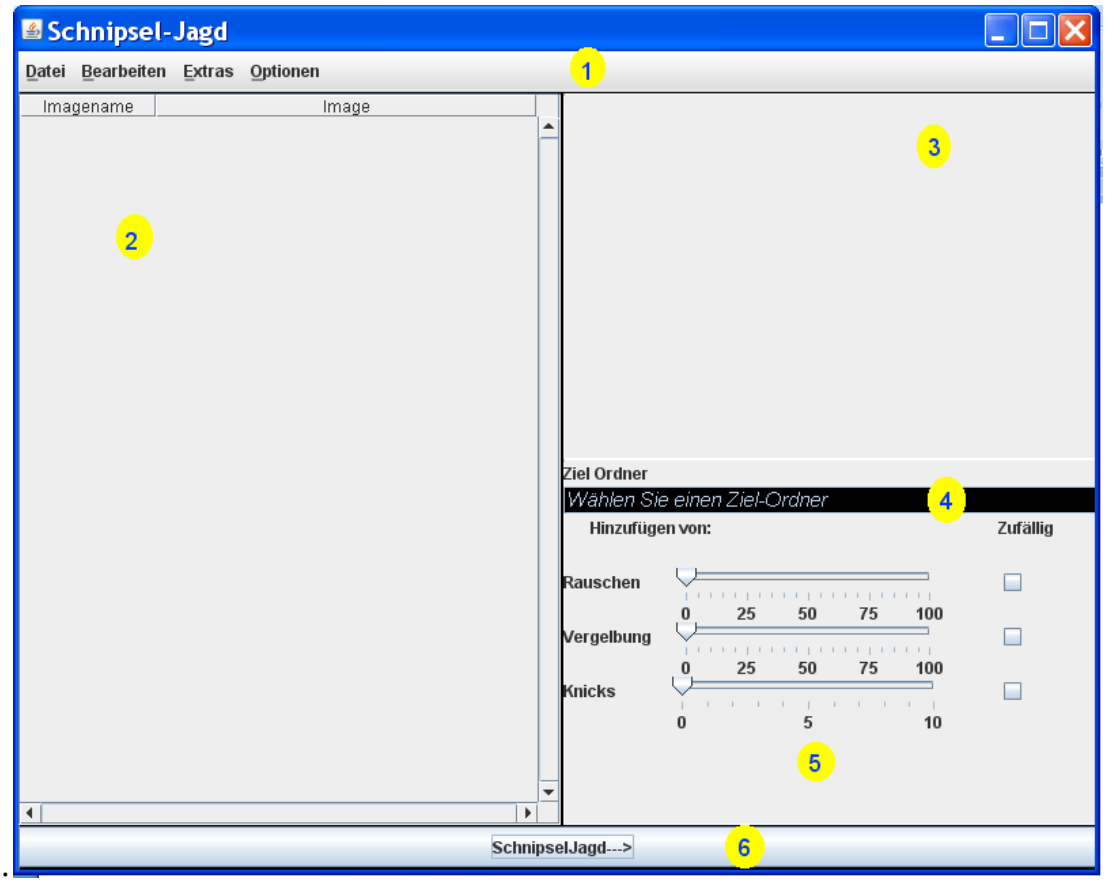


Abbildung 39: Hauptfenster des Schnipseljagd Frameworks

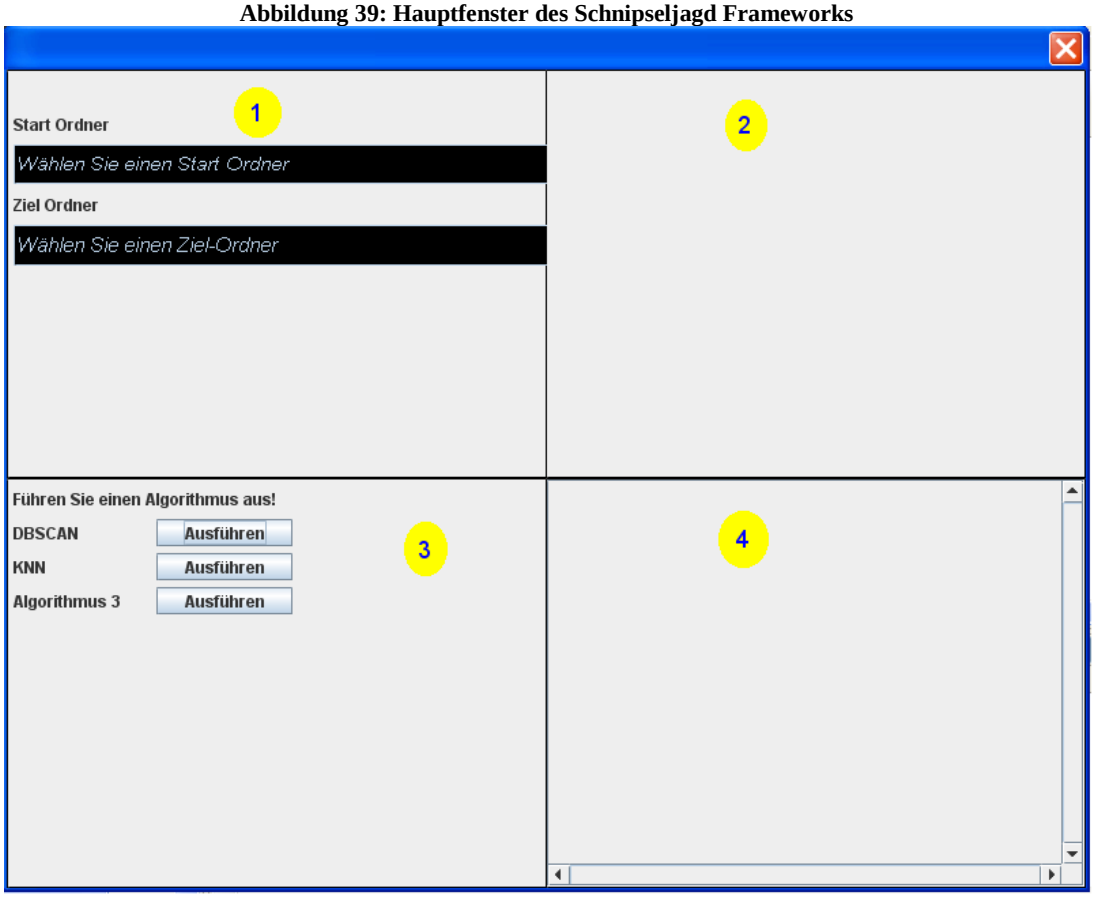


Abbildung 40: Das Prozessfenster

Das Prozessfenster besteht aus folgenden Komponenten:

(1) Das Adressenfenster, welches den Ordner der zu bearbeitenden Schnipsel auswählt und den Zielordner der bearbeiteten Schnipsel festlegt.
(2) Das Informationsfenster, welches einige Details über die Schnipsel gibt.
(3) Das Algorithmenfenster, das einige Algorithmen zur Verfügung stellt.
(4) Das Ergebnisfenster, das zum Veranschaulichen von Beispielen bearbeiteter Schnipsel dient.

4.2 Die Funktionen des Frameworks

Das vorgestellte Framework ermöglicht dem Benutzer unter anderem das Laden von Bildern, das Löschen, die Modifikation, die Erstellung von Dokumenten, das Schnipseln und die Analyse von Schnipseln.

4.2.1 Die Menüleiste

Nachfolgenden werden die Funktionen in der Menüliste kurz erläutert und Beispiele vorgestellt.

Datei
Unter dieser Funktion kann der Benutzer sowohl ein Bild als auch alle Bilder in einem Ordner öffnen. Diese werden in einer Tabelle mit zwei Spalten angezeigt. Die erste Spalte beinhaltet den Pfad des Bildes, während in der zweiten Spalte das Bild skaliert angezeigt wird. Diese Bilder stehen dann für weitere Bearbeitungsschritten bereit (vgl. Anhang A).

Bearbeiten
Hier stehen folgende Funktionen zur Verfügung. Um eine Funktion ausführen zu können, muss ein Bild mit einem Mausklick markiert werden.

- Hinzufügen: Nach der Auswahl „Hinzufügen“ kann ein neues Bild zu den Bildern in der Tabelle hinzugefügt werden.
- Löschen: Nach der Auswahl „Löschen“ wird das markierte Bild von der Tabelle gelöscht.
- Rotation: Nach der Auswahl „Rotation“ wird das markierte Bild um 90 Grad Rotiert.

Anhang A zeigt die Tabelle nachdem das Bild, welches im Anhang A ganz oben stand, gelöscht wurde und das danach folgende Bild rotiert wurde.

Extras
Der Generier-Prozess (vgl.3.1) und der Zerschnipseln-Prozess (vgl. 3.2) kann der Benutzer unter Extras starten. Der Benutzer kann beim Zerschnipseln-Prozess zwischen drei Modi entscheiden (vgl. Anhang A). Zusätzlich ist es möglich, die gewünschte Anzahl der Schnipsel abzulesen. Im Feld „min Größe“ wird die minimale Flächengröße des Rechtecks festgelegt, welches ein Schnipsel besitzen muss und schließt somit die Erzeugung von winzig kleinen Schnipseln. Der Prozess wird für alle in der Tabelle enthaltenen Bilder ausgeführt. Beim

Generier-Prozess besteht die Möglichkeit, die gewünschte Anzahl der zu generierenden Dokumenten festzulegen (vgl. Anhang A). Außerdem kann der Benutzer einen Format, z.B. Brief oder einen Standard auswählen.

Optionen
Nach der Auswahl von „Schnipsel Anzeigen“ werden die Tabelle und deren Inhalt angezeigt. Diese Option ist bei einer großen Menge von Daten sinnvoll. Die Auswahl von „Schnipsel Eigenschaften“ aktiviert das Eigenschaftenfenster (vgl. Anhang A).

4.2.2 Das Prozessfenster

In diesem Fenster sind die weiteren Funktionalitäten in dieser Arbeit auszuführen (vgl. Anhang A). Nachdem der Benutzer ein Start- und Zielordner festlegt, kann der DBSCAN Algorithmus gestartet werden. Infolgedessen wird unter anderem eine Datei erzeugt, welche die Featuresverktoren der entstehenden Bounding Boxen enthält. Der Benutzer kann diese Datei und die Datei der Referenzdatenbank laden und die Klassifikation starten (vgl. Anhang A).

4.2.3 Das Eigenschaftenfenster

Beim Mausklick auf ein Bild in der Tabelle können mehrere Informationen über das Bild angezeigt werden. Anhang A zeigt als Beispiel das Histogramm des markierten Bildes

5. Kapitel

Beiträge und zukünftige Arbeiten

Die Entwicklung eines universalen Programms, welches mit seinen Funktionen alle Prozesse in der Abbildung 41 deckt, ist noch nicht abgeschlossen. Mit dieser Arbeit wurde ein prinzipieller Ansatz vorgestellt, dessen Ziel es ist, insbesondere den ersten zwei Prozessen näher zu kommen. In diesem Sinne wurden neue Lösungsideen vorgeschlagen. Zusätzlich wurde eine Zusammensetzung aus bekannten Algorithmen vorgeschlagen, um für den dritten Prozess effiziente Lösungswege zu finden. Trotzallem müssen noch viele Punkte geklärt und weiter entwickelt werden.

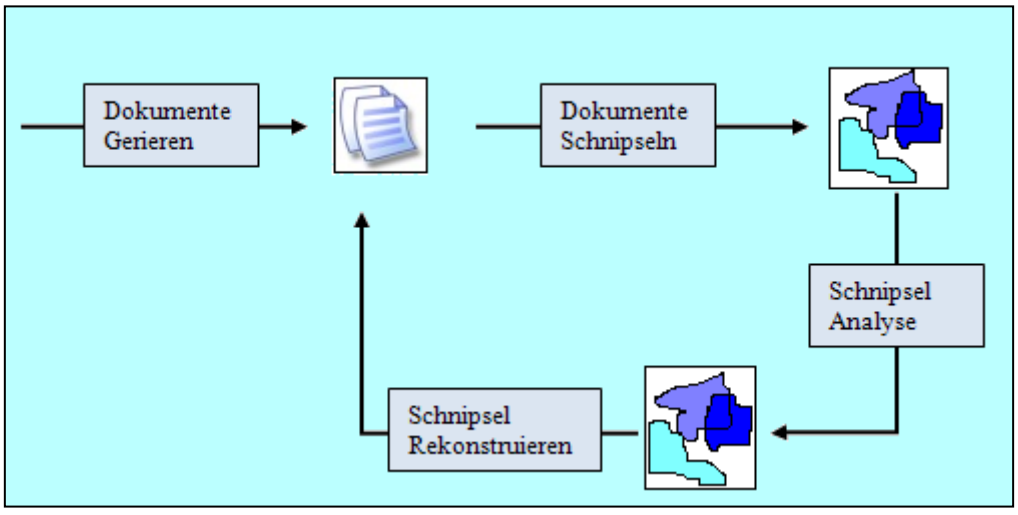


Abbildung 41: Ziel zukünftiger Arbeiten

5.1 Beiträge der Arbeit

In dieser Arbeit wurden sowohl eine Methode zur Generierung von Dokumenten als auch ein Verfahren zum Zerschnipseln von Dokumenten und ein Algorithmus zur Klassifizierung des Inhaltes auf einem Schnipsel vorgestellt. Die Möglichkeit, dass die Ergebnisse für andere Projekte zur Verfügung stehen, wurde berücksichtigt. Dafür wurde ein Muster zur Speicherung der Ergebnisse vorgeschlagen.

Ein dichtebasiertes Clusterverfahren wurde zur Clusteranalyse auf einem Schnipsel untersucht und mit einem Mustererkennung Algorithmus kombiniert. Dabei wurde der Einsatz vom k-nächten-Nachbarn Verfahren für die Klassifizierung der gebildeten Cluster diskutiert.

Eine umfangreiche Auswertung der vorgestellten Methoden wurde durchgeführt. Für die Auswertung wurde eine Testmenge aus Dokumenten mit den vorgestellten Methoden generiert und zerschnipselt. Das Ergebnis wurde als Teil der Testdaten für die Clusteranalyse und Klassifizierung verwendet. Auf diese Weise wurde die Erkennungsrate bestimmt.

Die hier vorgestellten Methoden wurden in einem Framework realisiert, dem zusätzliche Funktionen hinzugefügt wurden und der Benutzerfreundlichkeit dienen.

5.2 Ausblick auf zukünftige Arbeiten

Um den Ausblick auf zukünftige Arbeiten detaillierter zu beschreiben, wird dieser Aspekt in drei Unterabschnitte aufgeteilt, welche jeweils die drei Hauptprozesse in dieser Arbeit repräsentieren. Diese Prozesse wurden mit dem Name GZV Prozesse bezeichnet. Alle zur Realisierung der GZV Prozesse vorgestellten Methoden bedürfen zahlreichen Erweiterungen.

5.2.1 Der Prozess des (G)enerierens

Hinsichtlich der Ergebnisse steht den zukünftigen Arbeiten viel Arbeit vor. Denn es gibt ein breites Spektrum von vordefinierten Standards, die sich nicht nur auf die Gestaltung von Geschäftsbriefen oder deren Inhalt beschränken. Außerdem unterscheiden sich diese Normen von Land zu Land. So gelten z.B. in China oder in den USA andere Normen als solche die in Deutschland Anwendung finden. Des Weiteren werden diese Normen immer aktualisiert und weiterentwickelt.
In Hinsicht auf die Realisierung können Techniken aus der XML Welt verwendet werden und mit der hier verwendeten Programmiersprache Java eingebettet werden, um eine beliebige Anzahl von Standards zu definieren. Es wird viel Wert darauf gelegt, dass die Erstellung und das Hinzufügen von neuen Standards effizient und einfach zu realisieren sind.

5.2.2 Der Prozess des (Z)erschnipselns

Die Erweiterung in diesem Prozess beschäftigt sich hauptsächlich mit dem Konturverlauf eines Schnipsels. Gemeint sind vor allem die handzerrissenen Dokumente. Das Ziel besteht darin, eine realitätsnahe Gestaltung der Konturen zu erreichen. Bei der Realisierung sollte die Laufzeit und die Effizienz für eine beliebig große Anzahl von Dokumenten beachtet werden.

Das Ziel der beiden eben genannten Prozesse wird es weiterhin bleiben, virtuelle realitätsidentische bis hin zu optimale Alternativen von Datenmengen zu entwerfen, welche zwecks des Trainings, der Auswertung und der Optimierung von Algorithmen verwendet werden. Diese Algorithmen gehören zum Bereich der „Page Segmentation“ Classification“ sowie „Rekonstruktion von zerstörten Papierdokumenten“.

5.2.3 Der Prozess des (V)orklebens

Die Erweiterungen in diesem Prozess sind in zwei Teile zu betrachten, welche durch die zwei Teilprozesse, das Clustern und die Klassifizierung, repräsentiert werden. Für das Clusternprozess erscheinen folgende Ansätze denkbar:

- Die Farbwerte der einzelnen Pixels, seien es die Grauwerte oder auch die RGB Werte als Merkmale in dem DBSCAN Algorithmus untersucht werden. Dies kann zusätzlich zu den bereits verwendeten X-Y Merkmalen geschehen.
- Die Effizienz des Algorithmus kann bei der Erweiterung des Einsatzbereiches von Schnipseln auf vollständige Dokumente untersucht werden.

Für die Klassifizierung sind folgende Ansätze zu untersuchen:

- Es ist Interessant zu untersuchen, wie sinnvoll es ist, die Menge der gesuchten Klassen auf einem Schnipsel zu erweitern. Dabei wird die Klassifizierung verfeinert, indem die obengenannten Klassen in mehrere Teilklassen unterteilt werden. Auf diese Weise kann der Inhalt eines Schnipsels einem Element der folgenden Klassenmengen angehören $K = \{T1, T2, M, T, D, B, G, L, O\}$,
  wobei $T1, T2, M, T, D, B, G, L, O$, Text mit Font-Größe $\leq$ 18pt, Text mit Font-Größe $\geq$ 19pt, Mathe, Table, Druckraster, Bild, Gerade, Logo und Others repräsentieren (11). Eine weitere Verfeinerung die in Betracht käme, wäre die Unterscheidung zwischen Maschinen-und Handschrift (20).
- Andere Klassifikationsverfahren könnten eingesetzt werden. Denn bei einer Klassenmenge solcher Größenordnung ist die Anwendung von Verfahren wie Fuzzylogik oder maximale Entropie durchaus denkbar.

5.2.4 Das Framework

Alle in dieser Arbeit vorgestellten Methoden wurden in dem „Schnipseljagd“ Framework realisiert. Die folgenden Erweiterungen beschreiben einen Ausblick auf das, was in den zukünftigen Arbeiten möglich sein wird. Als erstes werden die vorgeschlagenen Ideen in den letzten drei Abschnitten mit dem Framework einzubetten sein.

Zusätzlich werden weitere Funktionen zur digitalen Bildverarbeitung hinzugefügt. Es wird die Definition einer Schnittstelle für Plugin‘s bezweckt (vgl. 4.1 das Algorithmenfenster). Sinn und Zweck dieser Schnittstelle beruht darauf, die Funktionalität des Frameworks durch weitere Algorithmen, die auf die erzeugten Datenmengen angewandt werden, zu erweitern.

6. Anhang A

6.1 Screenshots

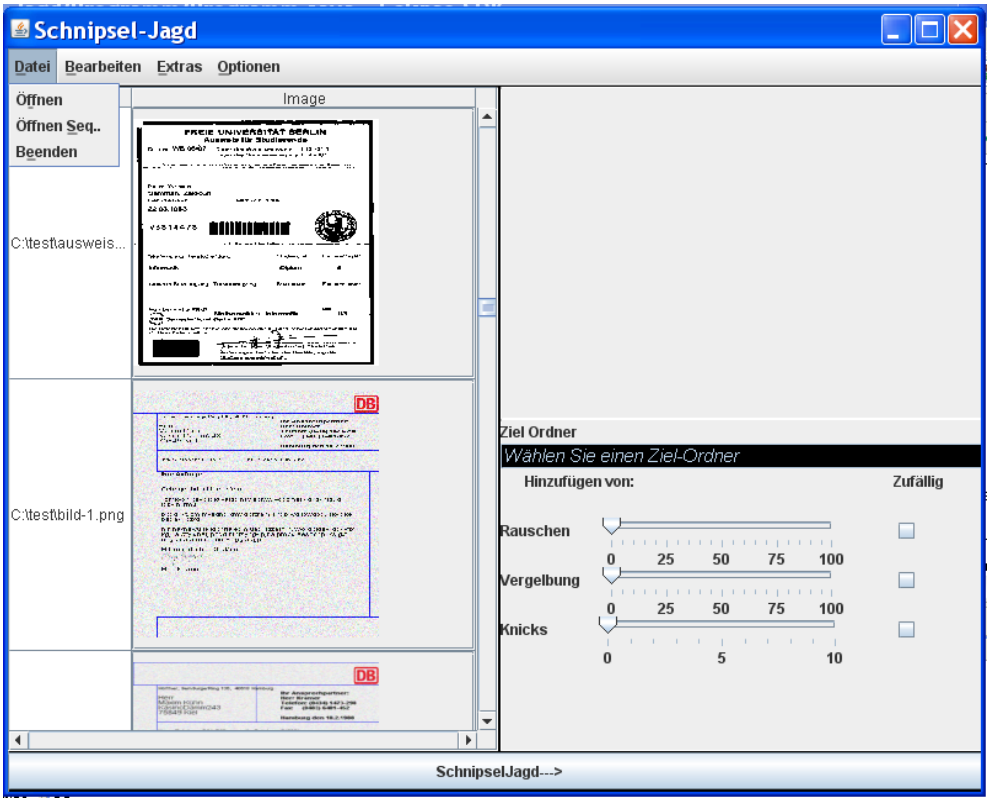


Abbildung 42: Die Datei-Funktion

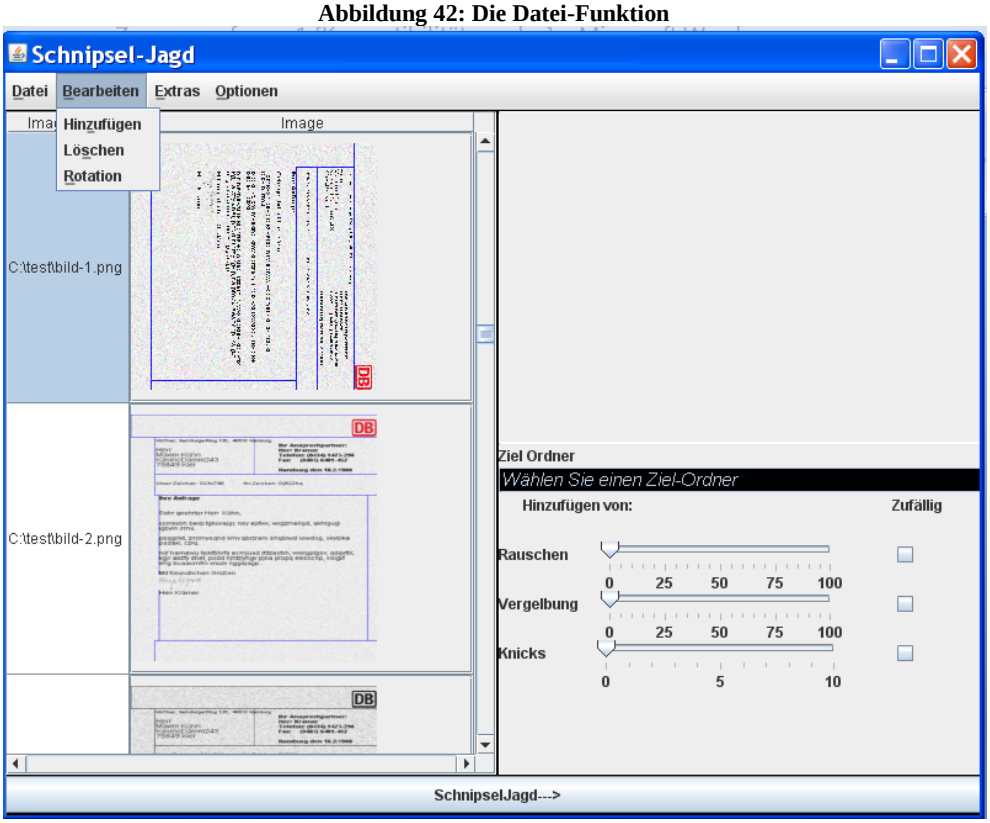


Abbildung 43: Die Bearbeitung-Funktion

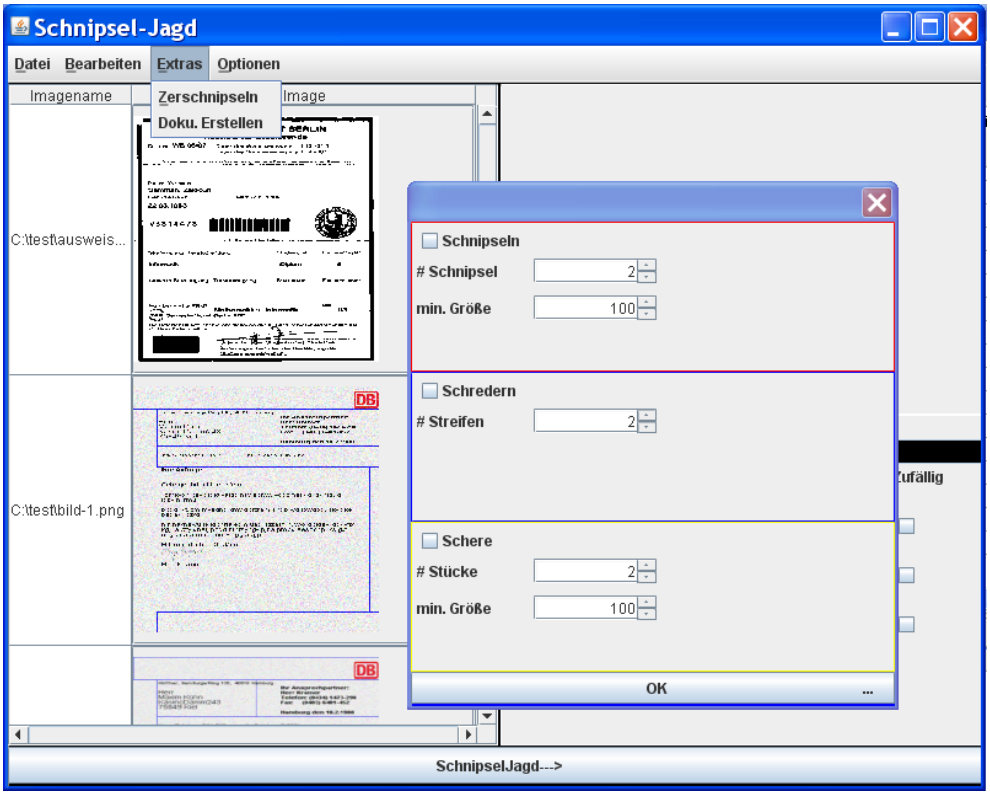


Abbildung 44: Die Extra-Funktion in der Menüleiste mit dem Zerschnipselnfenster

Abbildung 45: Das Dokumentengenerator

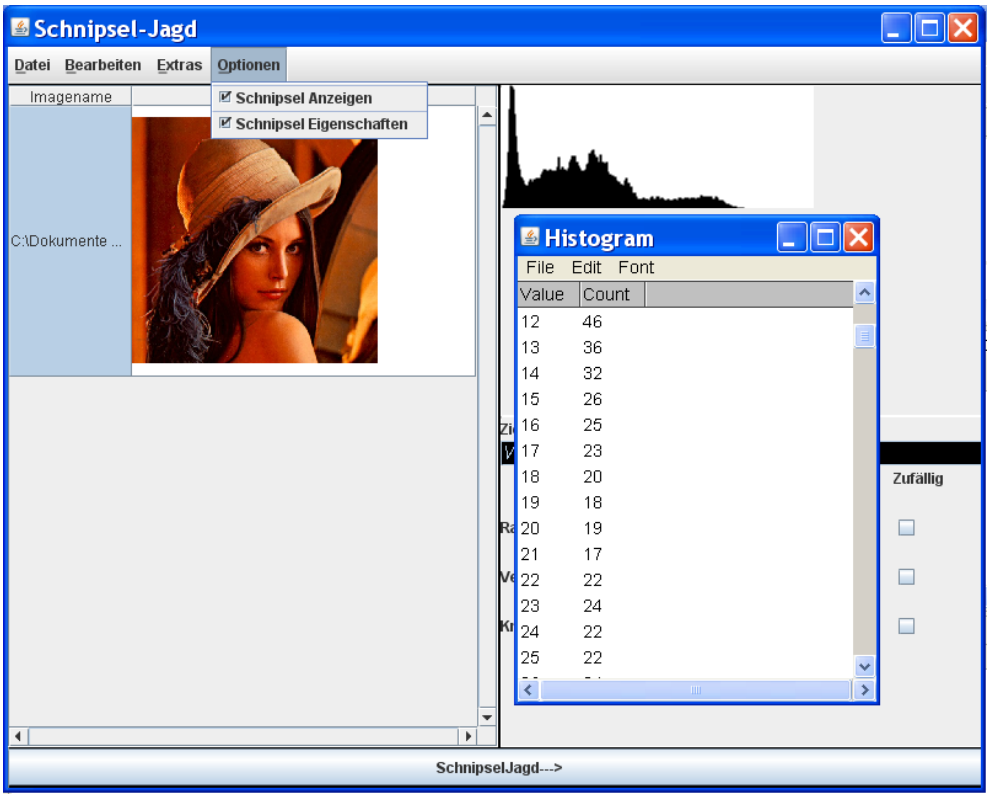


Abbildung 46: Das Eigenschaftenfenster

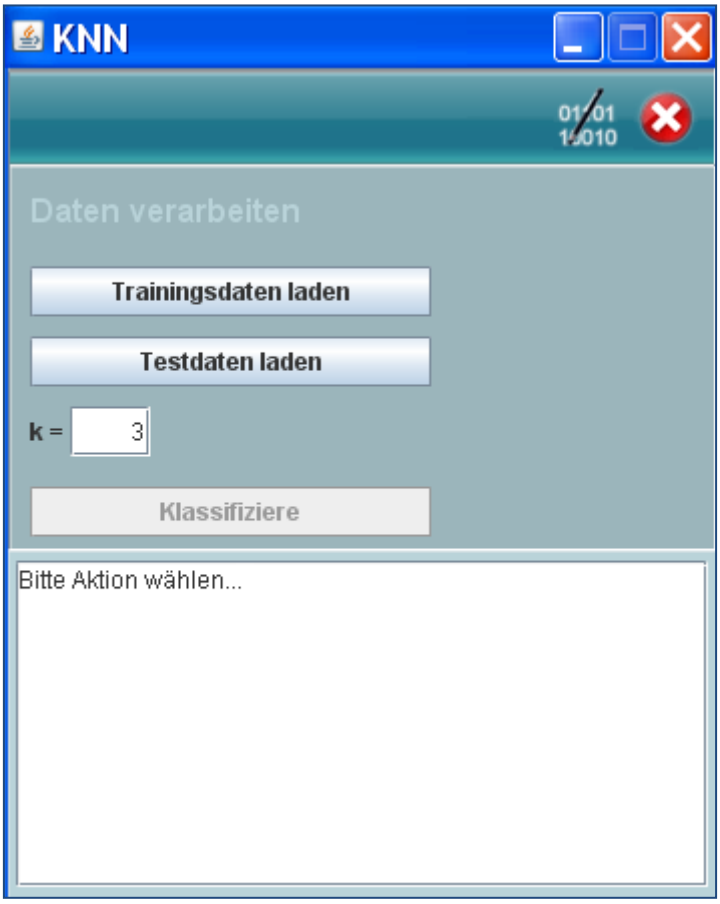


Abbildung 47: Das Klassifikatorfenster

7. Anhang B

7.1 Klassendiagramm

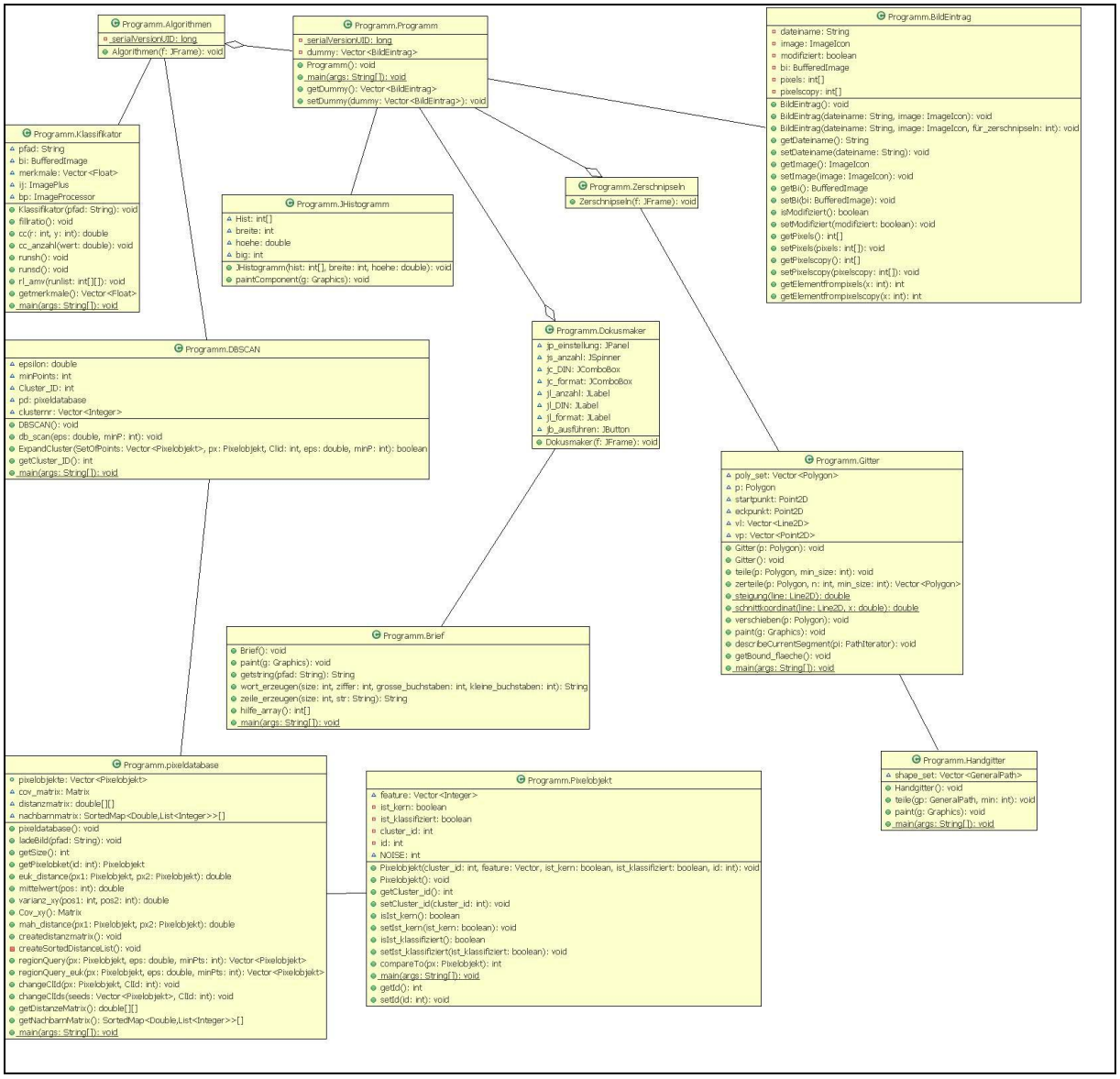


Abbildung 48: UML Klassendiagramm des „Schnipseljagd“ Projekts

8. Literaturverzeichnis

[1] DDR, Berliner Landesbeauftragten für die Unterlagen des Staatssicherheitsdienstes der ehemaligen. *Virtuelle Rekonstruktion „vorvernichteter" Stasi-Unterlagen.* Berlin, 2007. Bd. 2. ISBN: 978-3-934085-23-7.

[2] [Online] http://de.wikipedia.org/wiki/Model_View_Controller.

[3] Rinne, Horst. *Taschenbuch der Statistik.* : Harri Deutsch Verlag, 2008. 3817118279.

[4] http://de.wikipedia.org/wiki/Fingerabdruck. [Online]

[5] Davide Maltoni, Dario Maio, Anil K. Jain, Salil Prabhakar. *Handbook of Fingerprint Recognition.* : Springer, 2009. 1848822537.

[6] http://www.bka.de/pressemitteilungen/hintergrund/hintergrund1.html. Das Bundeskriminalamt. [Online]

[7] http://www.aktenvernichtung.de/pdf/merkblatt-sicherheitsstufen.pdf. Rhenus Data Office. [Online]

[8] http://www.bstu.bund.de/cln_028/nn_891038/DE/Archiv/Rekonstruktion/Bildergalerie. [Online]

[9] Chung, M.G., Fleck, M., Forsyth, D. *Jigsaw puzzle solver using shape and color.* Fourth International Conference of on Signal Processing., 1998. S. 877-880. Bd. Volume 2. 0-7803-4325-5.

[10] Theo Pavlidis, Jiangying Zhou. *Page segmentation and classification.* [Hrsg.] Graphical Models and Image Processing. 1992. S. 484-496. Bd. Vol. 54.

[11] Y. Wang, I.T. Phillips, R.M. Haralick. *Document Zone Content Classification and Its Performance Evaluation.* [Hrsg.] Pattern Recognition. 2006. S. 57-73. Bd. 39.

[12] Daniel Keysers, Faisal Shafait , Thomas M. Breuel. *Document image zone classification - a simple high-performance approach.* 2007.

[13] http://de.wikipedia.org/wiki/Papierformat. [Online]

[14] http://de.wikipedia.org/wiki/DIN_5008. [Online]

[15] http://www.libpng.org/pub/png/. [Online]

[16] Jörg Sander, Martin Ester, Hans-Peter Kriegel und Xiaowei Xu. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *DATA MINING AND KNOWLEDGE DISCOVERY.* : Springer, 1998, Bd. Vol. 2, S. 169-194.

[17] Stuart Inglis, Ian H. Witten. Document Zone Classification Using Machine Learning. *Proceedings of Digital Image Computing: Techniques and Applications.* Australia s.n., 1995, S. 631-636.

[18] Sergios Theodoridis, Konstantinos Koutroumbas. *Pattern recognition.*: Academic Press, 2006. S. 335-336. Bd. 3rd ed. 978-0123695314.

[19] Fahrmeir, Ludwig/Brachinger, Wolfgang. *Multivariate statistische Verfahren.* 1996. S. 418-421. Bd. 2. Auflage.

[20] Kavallieratou, E. Stamatatos, S. *Discrimination of Machine-Printed from Handwritten Text Using Simple.* [Hrsg.] 2004. ICPR 2004.Proceedings of the 17th International Conference Pattern Recognition. 2004. S. 437-440. Bd. Vol. 1. 0-7695-2128-2.

[21] Burger W., Burge J.B, "Digitale Bildverarbeitung - Eine Einführung mit Java und ImageJ ", ISBN 978-3540309406, Springer Verlag, 2006