Detektion und Erkennung von Text in Videos mit niedriger Qualität

vorgelegt von
Dipl.-Ing.
Andreas Cobet
geb. in Berlin

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurswissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Orglmeister
Gutachter:	Prof. Dr. T. Sikora
Gutachter:	Prof. Dr. P. Eisert

Tag der wissenschaftlichen Aussprache: 12.06.2015

Berlin 2015

Inhaltsverzeichnis

1. Kurzfassung

Text wird oft in Bildern oder Videos zusätzlich übertragen. Der begleitende Text dient dem Betrachter zur schnellen Information über den sachlichen Inhalt des aktuellen Bildes oder Videos. Außerdem kann Text in einem Bild oder Video Informationen liefern, die ohne das Betrachten nicht zugänglich sind. Auf einem Foto kann zum Beispiel ein Schild für ein Straßennamen vorhanden sein, welches helfen kann, den Ort von der Aufnahme des Fotos zu ermitteln. Solche Informationen sind auch in Videos vorhanden. Die Internetseite YouTube bietet eine Vielzahl an Videos in Form einer Datenbank. Diese Videos wurden hauptsächlich von Privatpersonen erzeugt und auf der Seite veröffentlicht. Für jedes Video konnte der entsprechende Besitzer eine kleine Liste an Stichwörtern bereitstellen, welche das Video beschreiben. Eine Suche auf dieser Seite nach entsprechenden Videos kann nur über diese Stichwörter stattfinden In vielen dieser Videos ist aber zusätzlich weiterer Text in den Bildern integriert, der in der Liste der Stichwörter nicht enthalten ist. Könnte dieser Text automatisch aus dem Video in die Liste der Stichwörter aufgenommen werden, wäre eine genauere bzw. umfangreichere Suche in der Videodatenbank möglich. In der Literatur gibt es eine Reihe von Verfahren, die versuchen, Text aus solchen Videos zu extrahieren. Jedes dieser Verfahren hat eine messbare Genauigkeit, mit der es den in dem Video enthaltenen Text extrahieren kann. Keines dieser Verfahren wird dabei 100% von dem enthaltenen Text extrahieren. Je nach Art und Qualität der Videos fällt die Genauigkeit der Extraktion aus. Der Grund liegt hier in der Beschaffenheit der Bilder in dem Video bzw. dem Video an sich. Für eine ausreichende Antwort auf die Fehlerursache muss näher auf die Beschaffenheit der Bilder bzw. des Videos eingegangen werden.

Prinzipiell befinde sich in einem Video deutlich mehr Informationen als auf einem Bild. Ein Video besteht aus einer Reihe von Bildern, die immer neue Informationen, aber auch sich wiederholende Informationen besitzen können. So kann sich in den Bildern der Text im Laufe der Zeit ändern. Die einfachste Form von Text in einem Video ist die Einblendung. Dieser Text wurde nachträglich in die Bilder integriert und ist daher leichter von anderen Objekten in den Bildern zu trennen. Schwieriger wird es bei Text, der in der Szene enthalten ist. Dieser Text befinde sich meist auf anderen Objekten und ist damit schwerer auszumachen. Dieser Text ist außerdem stärker von dem Kontrast, der Helligkeit, der Rotation, der Skalierung oder Deformationen abhängig. Damit Text gefunden werden kann, muss für die Unterscheidung von Text zu anderen Objekten in den Bildern eine Analyse der Eigenschaften von Schriftzeichen stattfinden Oft verändern sich diese Eigenschaften durch Fehler in dem Bild oder Video. Durch die begrenzte Datenmenge für jedes Video sinkt automatisch die Qualität und es entstehen Fehler. Es gibt unterschiedliche Verfahren zur Reduktion der Datenmenge und damit verändern sich die Eigenschaften von Schriftzeichen in den Bildern. Beispiele hierfür wären die Videokomprimierung durch MPEG oder durch Interlacing. Das Kapitel 2 dieser Arbeit befasst sich mit der Problematik und vergleicht die Fähigkeit des Menschen, einen Text zu lesen, mit den Möglichkeiten eines Computers. Hierbei wird schnell klar, dass das Lesen von Text in Schritten vollzogen werden muss. Als erstes muss der Text gefunden werden. Danach müssen die Schriftzeichen gelesen werden und abschließend entstandene Fehler korrigiert

werden. Daraus resultiert im Folgenden eine Aufteilung dieser Arbeit in die Kapitel:

- Detektion von Textbereichen (siehe Kapitel 3)
- Erkennung von Schriftzeichen (siehe Kapitel 4)
- Korrektur von Fehlern (siehe Kapitel 5)

In einem Video kommen nahezu unendlich viele verschiedene Objekte vor. Jedes dieser Objekte besitzt eine charakteristische Form und Gestalt. Im ersten Schritt zum Lesen von Text muss dieser daher gefunden und von anderen Objekten wie Autos, Häusern und allem Anderen getrennt werden. Dieser Schritt wird als Detektion von Text bezeichnet. Für das Auffinde von Text sind die Konturen der Objekte und damit der Schriftzeichen wichtig. Durch die Kantenerkennung nach Canny können diese Konturen gefunden und in ein Konturbild gespeichert werden. In einem so entstandenen Bild können viele Fehler durch die Videokomprimierung entstanden sein. Durch geeignete Filter können hier die Fehler reduziert werden. Eine entscheidende Eigenschaft von Text in Bildern ist die hohe Dichte der Kantenbildpunkte. Auf diese Weise können die Bereiche mit Text gefunden und die meisten anderen Objekte davon getrennt werden. Durch geeignete Parameter können dann die Textbereiche in einzelne Wörter getrennt und damit im nächsten Schritt interpretiert werden. Die so automatisch gefundenen Textbereiche müssen nicht immer mit denen vom Menschen gefundenen übereinstimmen. Damit die Zuverlässigkeit der automatischen Detektion von Text bestimmt werden kann, wurde im Rahmen dieser Arbeit ein Programm zum manuellen Markieren von Textstellen in einem Video entwickelt. Eine solche Markierung kann auch als Annotation bezeichnet werden. Mit Hilfe dieses Programms ist es möglich, relativ schnell Text in einem Video per Hand zu markieren. Mit Hilfe des manuell markierten Textes kann der automatisch markierte Text verglichen und ausgewertet werden. Das Kapitel 3 umfasst die theoretische Ausarbeitung, die praktische Umsetzung und eine Messung mit der dazugehörigen Auswertung der Genauigkeit der Detektion von Text in einem realen Video.

Sind die Bereiche der Wörter ermittelt, muss ein Schwarz-Weiß-Bild von diesen Bereichen erstellt werden. Ziel dabei ist es, alle Bildpunkte, die zu dem Wort gehören, in Schwarz und alle anderen Bildpunkte in Weiß darzustellen. Diese Trennung der Wörter von dem Hintergrund wird auch Binarisierung bezeichnet. Auf diesen Schwarz-Weiß-Bildern kann nun eine Interpretation der Schriftzeichen stattfinden Dazu müssen die Schriftzeichen voneinander separiert werden. Von den einzelnen separierten Schriftzeichen werden dann Merkmale extrahiert werden, welche danach mit Hilfe von einem Klassifikations erfahren interpretiert werden. In einem in dieser Arbeit erstellten und gemessenen Versuch werden Hidden Markov Modelle verwendet. In diesem Klassifikations erfahren wird für jedes verschiedene Schriftzeichen im Vorfeld je ein Modell erstellt und mittels des Baum-Welch Algorithmus durch Beispiele trainiert. Ein zu interpretierendes Schriftzeichen wird mit allen erstellten Modellen mittels des Viterbi Algorithmus verglichen und das wahrscheinlichste gewählt. Da dieses Vorgehen für alle Schriftarten unzählige Trainingsdaten benötigt und das Hauptaugenmerk der Arbeit auf der Fehlerkorrektur liegt, wird für die abschließenden Experimente das Programm zur Texterkennung LEADTOOLS OCR SDK verwendet. Das Kapitel 4 beinhaltet die theoretische Ausarbeitung, die praktische Umsetzung und eine Messung mit der dazugehörigen Auswertung der Genauigkeit der Erkennung von Text in realen Videos für das eigene Verfahren im Vergleich zu dem kommerziell erhältlichen Programm LEADTOOLS OCR SDK.

Durch die Detektion und Erkennung von Text entsteht eine Reihe von Fehlern. Zum einen werden Textbereiche nicht gefunden oder umgekehrt als Textbereich markiert, obwohl kein Text vorhanden ist. Zum anderen kann es bei jedem zu erkennenden Schriftzeichen zu einer falschen Interpretation kommen. Zusätzlich können mehrere Schriftzeichen als eins oder ein Schriftzeichen als mehrere zusammengefasst werden. Daraus resultieren Fehler, die durch eine Fehlerkorrektur reduziert werden können. Ziel dieser Arbeit ist es, ein Verfahren an die Detektion und Erkennung von Text anzuschließen, welches die Fehlerrate reduziert. Es existieren bereits verschieden Verfahren um die Fehlerrate zu reduzieren. Häufi werden mit Hilfe eines Wörterbuchs die Ergebnisse abgeglichen. Ein großer Nachteil liegt hier in der Vollständigkeit des Wörterbuchs. Eine Reihe von Verfahren versucht über viele Bilder in einem Video eine bessere Qualität in einem Bild für die Erkennung von Text zu erreichen. Der Nachteil dieses Verfahrens liegt darin, dass der durch die Erkennung von Text entstehende Fehler erst anschließend entsteht. In dieser Arbeit wird ein Verfahren vorgestellt, welches die Ergebnisse der Erkennung von demselben Text, welcher in vielen aufeinanderfolgenden Bildern in dem Video entsteht, über die Zeit mittelt und damit insgesamt den Fehler reduziert. Das Kapitel 5 befasst sich mit diesem Thema. Anhand von einfachen Beispielen wird als erstes eine empirisch gefundene Lösung analysiert. Daraus wird mit Hilfe der Levenshtein Matrix eine berechenbare Lösung für die Mittelung der Wörter über viele, zeitlich aufeinander folgende Ergebnisse der Texterkennung erstellt. Für die Auswertung des in dieser Arbeit entwickelten Verfahrens werden dazu theoretische Messungen unter verschiedenen Bedingungen vorgenommen und anschließend mit Messungen in einem realen Video verglichen.

2. Einleitung

2.1 Verbreitung von Text in Bildern und Videos

Text wird oft in Bildern oder Videos zusätzlich übertragen. Der begleitende Text dient dem Betrachter zur schnellen Information über den sachlichen Inhalt des aktuellen Bildes oder Videos. Es gibt hierfür viele Beispiele. Schlagzeile eines Nachrichtenthemas, Information über den zeitlichen Ablauf und Stand von Sportveranstaltungen, Titel und Synchronisierung von Filmen, Produktbeschreibung in Werbespots oder Datum auf einem Foto. Auf diese Weise erhält der Betrachter zusätzliche Informationen zu den Bildern, die er ansieht. Diese erleichtern es ihm den Inhalt der Bilder zu verstehen. Auf die gleiche Weise wird auch der gesprochene Text in Videos näher durch eine Einblendung von Text im Video beschrieben.

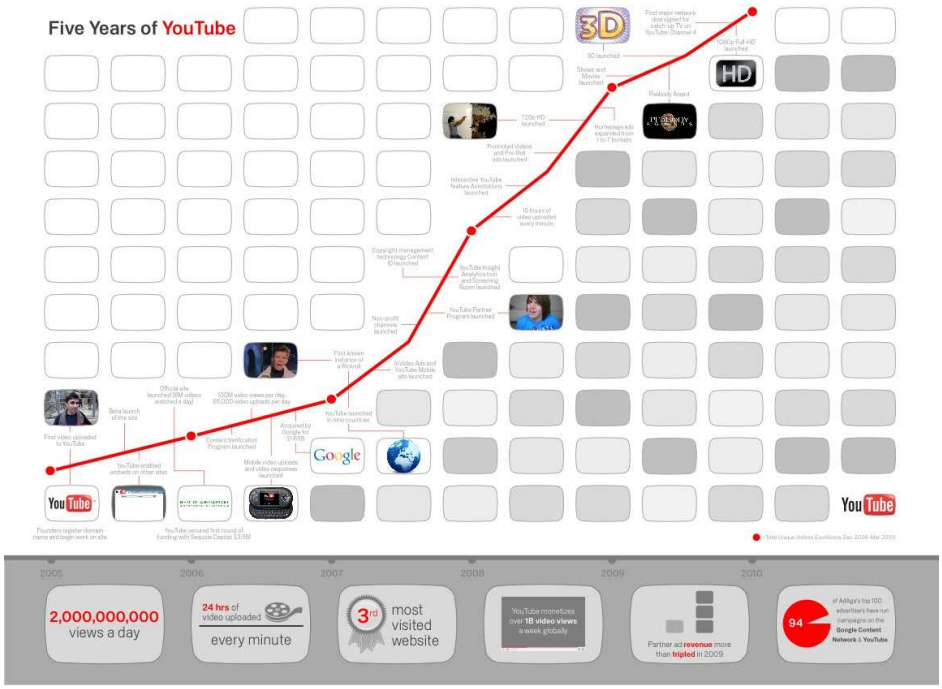


Abbildung 2.1: *Die Entwicklung von YouTube in den letzten 5 Jahren [1].*

Gegenwärtig wird ein steigender Gebrauch von Bildern speziell von Videos beobachtet. Hauptursache ist die ständige Verbesserung der Technik bildlicher Darstellungen. Die Anzahl an generierten Bildern und Videos pro Zeiteinheit erhöht sich erheblich. Eins der bekanntesten Beispiele hierfür ist YouTube. Im Jahr 2006 wurde YouTube etwa 100 mal pro Tag auf-

gerufen und täglich wurden rund 65.000 Videos hochgeladen. Bis 2009 hatte sich die Zahl bereits verzehnfacht [2, 3] und bis 2010 ein weiteres Mal verdoppelt [1] wie in Abbildung 2.1 zu erkennen ist. Der Gebrauch von Smartphones und Notebooks mit Gebrauch dieser Medien nimmt ebenfalls zu. Viele der so erreichbaren Videos haben nur wenige Informationen als Beschreibung über den Inhalt. Daher ist der Wunsch naheliegend, dass durch automatische Verfahren der Inhalt der Videos analysiert wird und damit das Video näher beschrieben werden kann, so dass eine Suchanfrage auf die Datenbank der Videos genauere Treffer liefert. Wertvolle Informationen sind dabei Ort und die Zeit der Handlung, Personen oder auch das Thema des Videos. Die Nachrichtenübertragung beschäftigt sich seit einiger Zeit damit, den Inhalt von Videos zu analysieren. Es wird versucht, Personen zu finde und sie auch zu erkennen [4], das Genre des Videos [5] zu klassifiziere und andere Charakteristika zu finden Diese Daten werden oft durch Texteinblendungen im Video mitgeliefert. Es liegt nahe, dass diese Informationen aus dem Video wieder ausgelesen werden und dann zur Beschreibung des Videos in der Datenbank gespeichert werden. So wird dem Nutzer eines diesbezüglichen Programmes die zeitaufwendige Suche nach einer bestimmten Thematik erspart.

2.2 Der Mensch als Vorbild für den Computer

2.2.1 Das menschliche Sehen

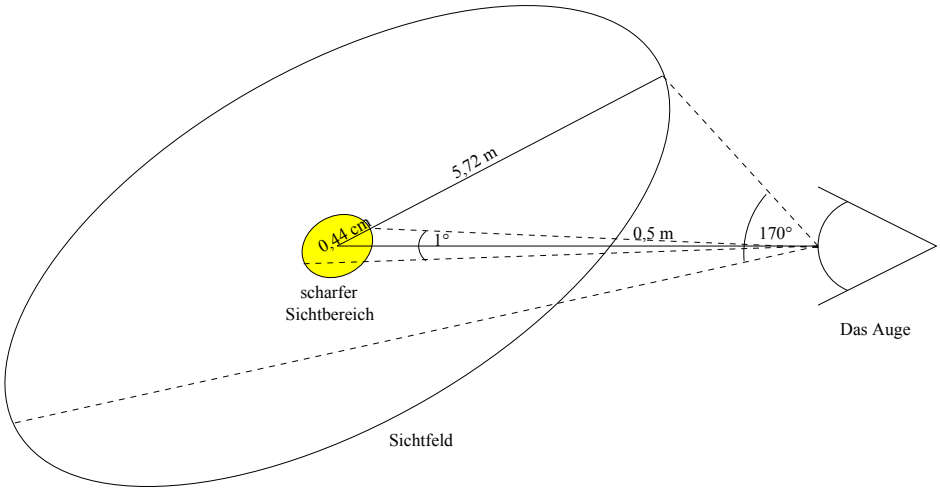


Abbildung 2.2: *Das Sichtfeld vom menschlichen Auge bei 0,5 Meter Abstand mit 170° horizontalem Blickwinkel beträgt 5,72 m. Der scharfe Teil dieses Bereiches mit 1° beträgt ungefähr 4,4 mm.*

Für den Menschen ist es vergleichsweise leicht diesen Text zu lesen und zu verstehen. Für den Computer ist der Text in Bildern oder Videos eine Information, die er nicht interpretieren kann. Erst durch eine Reihe an Algorithmen kann der Computer die Information mit relativ zum Menschen hoher Fehlerrate gewinnen. Hier stellt sich die Frage, wo der Unterschied

zwischen dem Menschen und dem Computer liegt. Um diese Frage zu beantworten, muss auf die unterschiedlichen Verarbeitungen der Informationen vom Menschen und vom Computer eingegangen werden.

In [6] wird die Funktionsweise des menschlichen Sehens beschrieben. Der Mensch kann Licht mit der Wellenlänge von rund 380 nm bis 780 nm in Form von Farben wahrnehmen. Das Licht dringt bei dem Mensch durch die Hornhaut und die Pupille in den Augapfel und fällt dann auf die Netzhaut. Die Netzhaut ist mit Sinneszellen bedeckt. Aber nur auf 0,02 Prozent der Fläche auf der Netzhaut kann der Mensch scharf sehen. Dieser kleine Bereich wird Gelber Fleck genannt. Das sind rund 5° unseres Gesichtsfeldes mit 170° horizontaler und 110° vertikaler Ausdehnung. Ein Gegenstand wird erst durch das Betrachten kleiner Abschnitte ersichtlich. Meist sind die Ausschnitte nur 1° des Blickfeldes groß. Jeder Bildbereich wird dabei 0,2 bis 0,6 Sekunden betrachtet. Auf diese Art wird ein Gegenstand oder die Umgebung durch das Auge abgetastet und so ein scharfes Bild erzeugt. In Abbildung 2.2 ist der Sichtbereich des menschlichen Auges am Beispiel mit einem Meter Abstand dargestellt. Nur ein kleiner Teil des sichtbaren Bereiches ist scharf. Für dieses Beispiel ergibt sich eine Fläche mit einem Durchmesser von 1,74 cm die scharf ist. Befind t sich ein 15“-Bildschirm in diesem Abstand, wird dieser ca. 240-mal abgetastet, wenn dieser vollständig erfasst werden soll. Damit wird eine Zeit von 48 bis 144 Sekunden benötigt. Ein Beispiel wäre eine Seite voll Text. Befinde sich nur wenig Text in dem Bild, werden nur entscheidenden Bereiche näher betrachtet und dementsprechend wird das Bild schneller erfasst.

2.2.2 Darstellung eines Bildes im Computer

Der Computer hat als Information ein Zahlenstrom, kombiniert aus den Zahlen Null und Eins. Erst durch eine Reihe von Definitione werden die Informationen verarbeitet. So werden je acht Zahlen, Bits genannt, zu einem Block zusammengefasst. Dieser wird dann ein Byte genannt und hat 2^8 Kombinationsmöglichkeiten. Ein Byte kann damit Werte zwischen 0000 0000 bis 1111 1111 im Binärsystem, 0 und 255 im Dezimalsystem bzw. 00 bis FF im Hexadezimalsystem annehmen. Mit diesem System sind auch Bilder und damit die Farben eines Bildes im Computer gespeichert. Es gibt verschiedene Methoden die Farben zu definie ren. Eine ist die Kombination aus den Farben Rot, Grün und Blau der auch Rot-Grün-Blau-Farbraum (RGB-Farbraum) genannt wird. Durch die Kombination dieser drei Grundfarben können 16.777.216 verschiedene Farben erzeugt werden. Ein Bild wird wie in Abbildung 2.3 in viele kleine Bildpunkte unterteilt, welche wie eine Matrix angeordnet sind, wobei ein Element der Matrix einem Bildpunkt entspricht. Ein Element der Matrix wird auf einem Bildschirm nicht als Punkt, sondern als kleine quadratische Fläche angezeigt. Jedes Matrix-Element wird durch drei Farbwerte erzeugt. Obwohl die Anzahl der Kombinationen in einem kleinen Bereich einer solchen Matrix begrenzt ist, ist die Anzahl zu groß, um jedes Objekt, das durch so eine Kombination entsteht, zu speichern bzw. zu definiere und so verbal zu beschreiben. Als Beispiel kann hier ein Album mit vielen Portraits herangezogen werden. Prinzipiell trifft auf alle Bilder die Beschreibung, es handelt sich um ein Gesicht, zu. Danach kann aber weiter unterschieden werden zwischen weiblich und männlich, der Haarfarbe, der Augenfarbe oder der Hautfarbe bishin zu kleinen Poren auf der Haut. So verändert jedes Detail das Bild und macht es einzigartig. Die so entstehende Menge an Bildern hat die Gemeinsamkeit, ein Gesicht zu enthalten, ist aber mathematisch nur schwer zu definieren Durch eine schlechte Qualität ist dann eventuell sogar fraglich, ob es sich um ein Portrait handelt. Daher

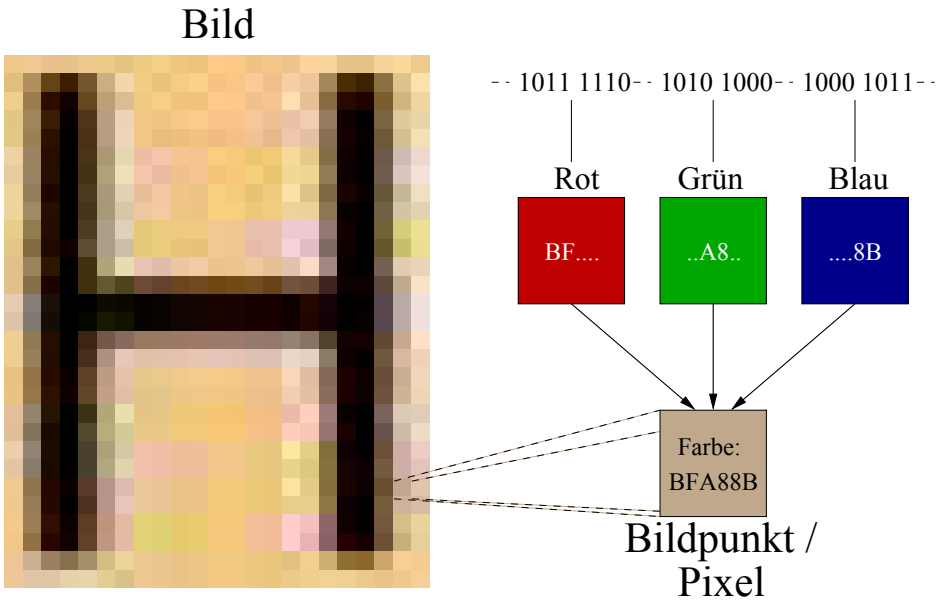


Abbildung 2.3: *Das Bild in einem Computer entsteht ausgehend von einem Bitstrom. Der Bitstrom (oben rechts) wird dabei immer wieder in 3 mal 8 Bit zerlegt. Jeder 3 mal 8 Bit Block erzeugt dabei eine Punkt in dem Bild, bestehend aus den 3 Farbanteilen Rot, Grün und Blau. Jede Farbkomponente kann auch im Hexadezimalsystem dargestellt werden (mitte rechts). Durch die Überlagerung der 3 Farbanteile wird dann die ursprüngliche Farbe erzeugt (unten rechts). Auf diese Weise werden alle Punkte im Bild nacheinander gespeichert und bei der Darstellung rekonstruiert.*

muss das Bild erst analysiert werden. Dazu wird, ähnlich wie bei dem menschlichen Sehen, durch Verfahren im Computer die Strukturen in der Matrix näher betrachtet und interpretiert und am Ende eine Entscheidung getroffen, die eventuell falsch ist.

2.2.3 Unterschied von Informationen für Mensch und Computer

Der generelle Unterschied zwischen dem Menschen und dem Computer besteht in der analogen Information für den Menschen und den digitalen Informationen für den Computer. Die Quantisierung der Daten in digitaler Form und damit der Verlust an Informationen wird bei der Vergrößerung des Schriftzeichens A im vergleich zu der idealen Form sichtbar (Abbildung 2.4). Während in der oberen Skalierung unendlich viele Zustände angenommen werden können und damit die Kanten des Schriftzeichens immer glatt erscheinen, ist die Anzahl der Zustände in der digitalen Form begrenzt, wodurch eine Kante bei näherer Betrachtung eine Blockstruktur besitzt. Deshalb ist die Größe des Schriftzeichens in Bezug auf die Auf ösung des Bildes zu beachten. Je mehr Bildpunkte, auch Pixel genannt, zur Beschreibung des Schriftzeichens zur Verfügung stehen, umso genauer können die Konturen beschrieben

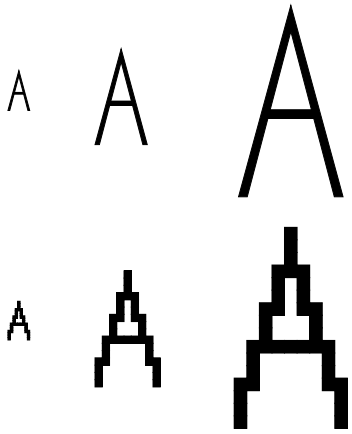


Abbildung 2.4: *Die Quantisierung der Daten in digitaler Form und damit der Verlust an Informationen wird bei der Vergrößerung des Schriftzeichens A im Vergleich zu der idealen Form sichtbar.*

werden. Durch eine Vergrößerung der Pixelanzahl ändert sich das Verhältnis von der Anzahl der Pixel zur Länge und Breite des Schriftzeichens. Da sich die Bildgröße über die Anzahl der Pixel definiert können Schriftzeichen deshalb alle denkbaren Größen in einem Bild annehmen. Deshalb sind Annahmen wie ein Schriftzeichen ist 20 mal 20 Bildpunkte groß nur eingeschränkt möglich. Die unterschiedlichen möglichen Größen eines Schriftzeichens ergeben auch das Problem, dass die Methoden der Suche von Text der Größe angepasst sein müssen. Ist ein Schriftzeichen sehr groß und es wird aus einem zu kleinen Abstand vom Menschen betrachtet, dann ist es nicht lesbar. Unter der Vorrausetzung eines Augenabstandes von 30 cm Entfernung zum Plakat mit einer Größe von 15 Metern Durchmesser und einem Plakat füllenden Schriftzeichen kann ein Mensch das Schriftzeichen nicht identifzieren. Erst durch einen größeren Abstand kann er das Schriftzeichen erkennen. Ein anderes Beispiel ist eine Ameise, die sich auf dem Rücken eines Elefanten befindet Sie kann ebenfalls nicht erkennen, dass sie sich auf einem Tier befindet Genauso ist es nicht möglich in einem Computerbild durch die Analyse zu kleiner Teilbereiche ein Schriftzeichen zu erkennen. Erst durch die Betrachtung unterschiedlicher Bereichsgrößen können alle Schriftgrößen gelesen werden. (siehe Abschnitt 3.2.6).

2.2.4 Erkennung von Objekten

Prinzipiell nimmt der Mensch und der Computer mit der Wahrnehmung bis dahin nur Strukturen wahr, kann sie aber nicht interpretieren. Zwischen dem Wahrnehmen bzw. Detektieren und dem Interpretieren bzw. Erkennen besteht ein Unterschied. Um die Strukturen zu erkennen müssen Erfahrungen zu Vergleichszwecken vorhanden sein, was diese Struktur bedeutet. Der Mensch muss jede Struktur und deren Bedeutung über viele Beispiele trainieren. Für jedes Objekt, wie Baum, Blume oder Haus werden unzählige Beispielbilder benötigt. So ergibt sich für diese abstrakten Begriffe Baum, Blume oder Haus je ein Modell. Wie dieses Modell

Abbildung 2.5: *Verschiedene Darstellungen eines Baumes.*

exakt entsteht ist noch nicht richtig geklärt und wird in der Informatik als Semantische Lücke beschrieben. Der Begriff Baum kann zum Beispiel in Abbildung 2.5 sehr unterschiedlich aussehen. Dennoch steht jede der drei Darstellung für denselben Begriff.

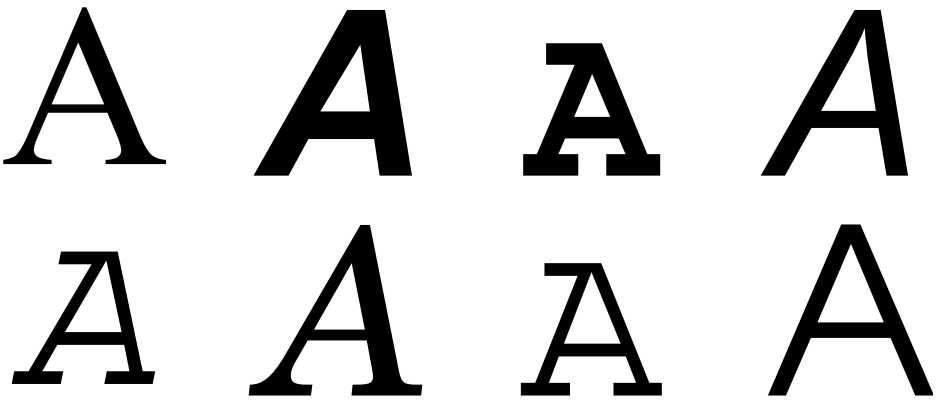


Abbildung 2.6: *Beispiel für verschiedene Schriftarten am Beispiel des Schriftzeichens A.*

Auf die gleiche Weise besitzen Schriftzeichen ebenfalls unterschiedliche Darstellungsformen. In Abbildung 2.6 sind 8 verschiedene Symbole. Dennoch stehen alle für das Schriftzeichen 'A' (siehe Abschnitt 4.4).

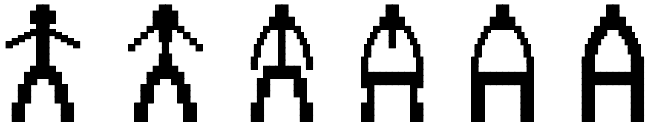


Abbildung 2.7: *Beispiel des Schriftzeichnens A (rechts) und des Strichmännchen-Symbols (links) mit Zwischenstufen.*

In Abbildung 2.7 sind verschiedene Symbole aufgelistet. Rechts ist das Schriftzeichen A und links das Symbol eines Strichmännchens. Dazwischen befinde sich vier weitere Symbole als Übergangsformen. Der jeweilige Nachbar hat dabei nur wenige Unterschiede. Hieraus ergeben sich zwei Probleme für die Erkennung. Zum einen ist es selbst mit gut trainierten Modellen nicht immer möglich, Objekte richtig zu erkennen und damit zu klassifizieren Durch Unschärfe im Bild kann es zu Übergangsformen kommen, wodurch dann das Objekt nicht immer exakt zugeordnet werden kann. Zum anderen besitzen Schriftzeichen ähnliche Eigenschaften wie andere Objekte im Bild (siehe Abschnitt 2.6). In diesem Fall ist es ein Schriftzeichen und ein abstraktes Symbol für einen Mann. Bei der Erkennung von Schriftzeichen muß also deren Variabilität berücksicht werden. Die Variabilität kann im Extremfall mit Ähnlichkeiten zu Verwechselungen führen. Beispielsweise kann ein Tischbein mit einem I verwechselt werden, wenn der Zusammenhang mit einem größeren Objekt übersehen wird. Die menschliche Erfahrung veranlasst ihn zu einer genaueren Überprüfung des Objektes. Der Computer hat zunächst keine Erfahrung. Das Wissen für andere Objekte muss dem Computer erst vermittelt werden und ist noch nicht gelöst. Um eine höhere Trefferquote zu erreichen, können für spezielle Situationen gewisse Annahmen getroffen werden, wie die Größe des Textes oder dessen Farbe. Eine weitere mögliche Annahme ist, dass der Text erst später in das Bild eingefügt wurde. Die Algorithmen im Computer können bisher nur potentielle Textstellen finde und diese dann näher interpretieren (siehe Abschnitt 3).

2.3 Unterschiede zwischen Bildern und Videos

In den bisherigen Erläuterungen sind nur Bilder betrachtet worden. Bilder haben eine feste Information, die sich in der Zeit nicht ändert. Ein Video besteht aus vielen einzelnen zeitlich aufeinander folgenden Bildern. Jedes Bild kann theoretisch völlig andere Informationen beinhalten. Praktisch sind in Videos Kameraeinstellungen erkennbar, in denen sich die Information in einem begrenzen Rahmen in den zeitlich aufeinander folgenden Bildern verändern. Die Kameraeinstellungen sind durch eine Kameraeinstellungsgrenze voneinander getrennt. Die Kameraeinstellungsgrenze wird dadurch definiert dass sich mehr Informationen von einem Bild zum nächsten ändern, als eine vorher festgelegte Grenze es zulässt. So befinde sich redundante Informationen in dem Video, welche für die Analyse, im Gegensatz zu einem einzelnen Bild, genutzt werden können. In einem Bild kann es durch Rauschen zu falschen Farben in einem Pixel kommen. Dieses Rauschen kann durch Flimmern der Luft oder ein Fehler im Objektiv entstehen. Auch denkbar ist, dass in einem Bild ein Objekt durch ein anderes Objekt kurzzeitig verdeckt wird. Ein Bild in einer Kamera entsteht durch eine feste Belichtungszeit. Die Belichtungszeit ist ein von der Lichtstärke abhängiger Zeitraum. Je stärker das Licht ist, umso kürzer ist die Belichtungszeit. Der Zeitraum ist notwendig, damit genug Lichtteilchen gesammelt werden können und das Objekt, auf das die Kamera gerichtet ist, erfasst wird. Es ist nicht möglich ein Bild nur durch einen festen Zeitpunkt zu erzeugen, da dann eine unendlich große Lichtstärke notwendig wäre. Im Umkehrschluss würde bei unendlicher Belichtungszeit ein perfektes Bild entstehen. Da das nicht möglich ist, wird ein Kompromiss für den Zeitraum gewählt, der ein zufriedenstellendes Bild ergibt, aber auch nicht zu lange Zeit in Anspruch nimmt. So ist die Bildqualität von der Länge der Belichtung abhängig. Ist die Belichtungszeit lang genug, wird ein Objekt, das sich im Verhältnis zur Belichtungszeit nur sehr kurz vor der Kamera befindet durch die zeitliche Mittlung nicht erfasst. Als Beispiel

können Sternenaufnahmen mit langen Belichtungszeiten genommen werden, wärenddessen in dem Zeitraum ein Flugzeug durch den Bereich des Bildes fli gt. Das Flugzeug würde auf dem Bild nicht zu erkennen sein. Angenommen, in einem Video ist die Kamera auf ein Objekt fixiert dann befinde sich in diesem Video in den einzelnen Bildern Informationen, die immer dasselbe repräsentieren, jedoch durch den begrenzten Belichtungszeitraum bzw. durch die zeitlich begrenzten Ereignisse wie Flimmern der Luft und anderen Einf üssen unterschiedlich sein können. Wenn diese Bilder zeitlich gemittelt werden, kann daraus eine bessere Qualität erzeugt werden. Diese Eigenschaft kann für eine höhere Rate der Detektion und Erkennung von Text ausgenutzt werden. Auch wenn die Kamera sich bewegt oder der Text sich bewegt, kann der Unterschied von einem Bild zum nächsten ausgenutzt werden. Diese Arbeit befasst sich im Kernpunkt um die Nutzung der redundanten Informationen von Videos im Vergleich zu Bildern.

2.4 Text als Einblendung

Abbildung 2.8: *In dem Bild sind verschiedene Texte nachträglich eingeblendet worden, um zusätzliche Informationen mit zu liefern.*

In vielen Videos bzw. Bildern werden nachträglich Information in Form von Text eingeblendet, um sie mit weiteren Informationen anzureichern. Es gibt hierfür viele Beispiele. Schlagzeile eines Nachrichtenthemas [7], Information über den zeitlichen Ablauf und Stand von Sportveranstaltungen [8], Titel und Synchronisierung von Filmen, Produktbeschreibung in Werbespots oder Datum auf einem Foto. In Abbildung 2.8 sind verschiedene Einblendungen erkennbar. Im unteren Bereich des Bildes befinde sich verschiedene Formen von Texteinblendungen. Hierbei ist der Satz 'Zivilschutz: Keine Schäden nach Erdbeben in Mexiko' in grauer Farbe auf weißem Hintergrund. Ganz links befinde sich im gleichen Farbschema

der Text 'S&P-Future 1.042 +0,7%' in deutlich kleinerer Schrift. Dazwischen befinde sich der Text 'Nachrichten' in invertierten Farben. Rechts oberhalb dieser Textzeile befinde sich der Text 'n-tv' in fetten Schriftzeichen und der Text '11:32 mit weißer Schrift und rotem Hintergrund.

Für den Text in diesem Bild können eine Reihe von Annahmen getroffen werden, welche die Erkennung vereinfachen. Die Einblendungen sind über die Zeit immer an denselben Orten. Dadurch muss nicht erst das ganze Bild auf Text abgesucht werden. Die Schriftgröße der einzelnen Einblendungen an den verschiedenen Orten ist immer identisch. Deshalb muss bei der Erkennung der Schriftzeichen keine Skalierung vorgenommen werden. Da, wie oben beschrieben, die Farbe der Schriftzeichen und die Farbe des Hintergrundes bekannt sind, erleichtert es die Schrift vom Hintergrundbild zu trennen. Diese Trennung wird auch Binarisierung genannt (siehe Abschnitt 4.1). Für die Binarisierung ist es sehr hilfreich, wenn die Schriftfarbe, wie in dem Beispielbild, einfarbig ist und die Schrift sich auf einem einfarbigen Hintergrund befindet Damit ist der Kontrast zwischen dem Schriftzeichen und dem Hintergrund hoch, was die Binarisierung vereinfacht. In dem Beispielbild ist aber auch die Texteinblendung 'Wahl des Bundespräsidenten' oben links ohne einen einfarbigen Hintergrund vorhanden. Diese Einblendung mit weißer Farbe kann in bestimmten Videoszenen mit hellem Hintergrund nur schwer bzw. gar nicht detektiert oder erkannt werden.

Für die Erkennung ist ebenfalls sehr wichtig, dass die Schrift waagerecht positioniert ist. Ist die Schrift rotiert, kann die Erkennung der Schriftzeichen ohne Kenntnis des Rotationswinkels fehlschlagen. In Abbildung 2.9 sind die Schriftzeichen P und d erkennbar. Erst durch den Bezug zu anderen Schriftzeichen (rechts) wird ersichtlich, dass es sich in beiden Fällen um dasselbe Schriftzeichen P handelt und nicht um das Schriftzeichen d. Die obere Zeile ist also nur um 180° gedreht.

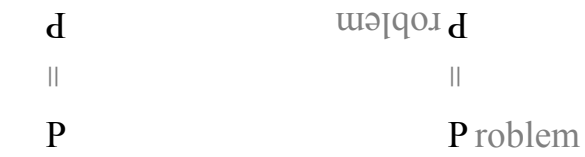


Abbildung 2.9: *Veränderung der Bedeutung eines Schriftzeichens durch Drehung um 180°.*

An diesem Beispiel ist ersichtlich, dass eine Rotation des Textes im Bild Einflus auf die Erkennung haben kann. Auch Rotationen von nur 90° können zu Fehlern in der Erkennung führen. Bei einer solchen Rotation können die Schriftzeichen Z und N identisch sein. Auch jede andere Rotation kann die Erkennung verändern. Die Modelle zur Erkennung von einzelnen Schriftzeichen sind so trainiert, dass die Schriftzeichen in der waagerechten liegen. Also muss die Erkennung eines Schriftzeichens auch in der Waagerechten stattfinden Deshalb muss vor der Erkennung des Textes ermittelt werden, um welchen Winkel der Text rotiert wurde (siehe Abschnitt 4.4). Zu den bisher aufgelisteten Annahmen müssen weitere Eigenschaften in Videos beachtet werden. Entscheidend ist hierbei der zeitliche Verlauf. Texteinblendungen können sich in den einzelnen, zeitlich aufeinander folgenden Bildern in einem Video ändern oder sich verschieben.

Eine spezielle Form der Texteinblendung befinde sich in dem Video (siehe Abschnitt 2.3). Hier wird ein Laufband eingeblendet, welcher sich von rechts nach links bewegt. Einen

Abbildung 2.10: *Das Wort 'hat' bewegt sich von rechts nach links. In den Bildern 1 bis 3 ist diese Bewegung markiert.*

ähnlichen Effekt ist auch oft am Ende von Filmen zu finden Im Abspann bewegt sich hier eine Liste mit den Namen der Akteure von unten nach oben. In beiden Fällen ändert sich die Position des Textes in der Zeit. In Abbildung 2.10 ist das Wort 'hat' markiert. Es ist erkennbar, dass sich die Position des Wortes in den Bildern 2 und 3 zum Bild 1 verschiebt. Durch die Verschiebung des Wortes können nicht direkt mehrere Bilder zur automatischen Erkennung berücksichtigt werden. Es muss erst festgestellt werden, wo das Wort sich in jedem Bild befindet Dafür kann eine Textverfolgung genutzt werden (siehe Abschnitt 3.3)

2.5 Text in Szenen

In einem Bild gibt es oft auch Text, der nicht nachträglich eingeblendet wurde. Oft befin det sich Text auf Objekten in der Szene. Als Beispiel dient hier in Abbildung 2.11 eine Fotografi eines Flugzeugsrumpfes. Das Bild hat einen geringen Kontrast und die verschiedenen Farben im Bild sind sehr ähnlich. Oben links ist hellblauer Himmel und darunter ist das Flugzeug in blaugrauen Farben. Da nur ein Teil des Flugszeugs auf dem Bild ist, sind nur einige Fenster und ein Schriftzug 'SILVERJET' unterhalb der Fenster erkennbar. Auch die Fenster und der Schriftzug besitzen einen geringen Kontrast zum Hintergrund. Anhand dieses Beispiels sind viele Probleme bei der Detektion und Erkennung von Text in Szenen erkennbar. Eine der wichtigsten Punkte war bisher, dass Text, der nachträglich in ein Bild eingefügt wurde, sich meist sehr gut vom Hintergrund durch einen hohen Kontrast abhebt. Text in Szenen besitzt dieselbe Beleuchtung wie die Umgebung. Dadurch gleichen sich die Farben von Text und Hintergrund oft an und der Kontrast verringert sich oder verschwindet unter Umständen vollständig. Für das Auffinde von Text in Bildern und Videos ist dieser Kontrast sehr wichtig. Durch die Beleuchtung der Szene ist der Text nicht immer monochrom bzw. besitzt oft sogar ein Farbverlauf. Auch diese Eigenschaft erschwert eine Detektion und

Erkennung gegenüber den einfarbigen Texteinblendungen.

Abbildung 2.11: *In dem Bild ist ein Text erkennbar, der sich auf einem Flugzeugrumpf befindet*

Eine weitere Eigenschaft von Texteinblendungen ist meist die waagerechte Ausrichtung. Auch diese Eigenschaft trifft selten bei Text in Szenen zu. Meistens ist der Text von der waagerechten Position um einen unbestimmten Winkel rotiert. Die meisten Algorithmen zum Auffinde von Text nutzen die Eigenschaft, dass Text mehr breit als hoch ist. Wenn der Text aber soweit gedreht ist, dass diese Eigenschaft umgekehrt wird, sinkt automatisch die Detektionsrate. Auch wenn der Text detektiert wurde, ist die Erkennung ohne Kenntnis des Rotationswinkels erschwert. Die meisten Algorithmen zur Erkennung von Schriftzeichen gehen davon aus, dass die Schriftzeichen ebenfalls waagerecht vorliegen. Ein weiteres Problem bei Text in Szenen ist, dass der Schriftzug nicht frontal aufgenommen wird. Oft befinde sich der Schriftzug auf einem Objekt, welches sich in einem Winkel von nicht 90° zur Kamera befindet Dieser Winkel verursacht eine perspektivische Verzerrung, so dass die Schriftzeichen, die weiter entfernt von der Kamera sind, kleiner in der Aufnahme dargestellt werden als diejenigen, die sich dichter befinden In dem Beispielbild ist so das Schriftzeichen 'S' im Schriftzug 'SILVERJET' durch den Winkel der Aufnahme zum Objekt kleiner als das Schriftzeichen 'T'. Diese Verzerrung führt dazu, dass die Schriftzeichen durch die digitale Struktur des Bildes unterschiedliche Eigenschaften besitzen. Als letzter, aber wichtigster Punkt bei der Detektion von Text in Szenen sind Objekte neben dem Text zu beachten. Wie in dem Beispiel zu erkennen, befinde sich oft Strukturen im Bild, die ähnliche Eigenschaften wie Text aufweisen. In diesem Beispiel sind es die Fenster des Flugzeugs. Die Fenster haben ebenfalls kleine Strukturen in der Größe von Text und einen vergleichbar hohen Kontrast zum Hintergrund. Sie könnten fälschlich als 'O' erkannt werden. Solche Objekte in Szenen verursachen oft Fehler in der Erkennung und sind nur schwer zu vermeiden. Es kann auch dazu kommen, dass solche Objekte Text überlagern und damit eine Detektion oder Erkennung unmöglich machen. Diese Eigenschaft wird heute oft bei Internetseiten genutzt, um eine automatische Textwiedergabe als Sicherheitsbarriere für Datenabfragen durch Computer zu verhindern. Dieser Test heist 'Completely Automated Public Turing test to tell Computers and Humans Apart' (Captcha) [9]. In Abbildung 2.12 ist ein solcher veränderter Text dargestellt.

Abbildung 2.12: *Die Textlesbarkeit im Bild wurde absichtlich reduziert, um die automatische Detektion und Erkenunng durch einen Computer zu verhindern. So werden Internetseiten vor automatischen Zugriffen geschützt.*

2.6 Eigenschaften von Text

Für die vollautomatische Detektion und Erkennung von Text ist eine Analyse der Eigenschaften von Schriftzeichen in Bildern und Videos notwendig. Die wichtigste Frage dabei ist, wie Text in Bildern gefunden werden kann. Anders gefragt, was unterscheidet den Text von anderen Objekten in einem Bild? In der Literatur [10, 11] hat sich gezeigt, das Text eine hohe Dichte an Kanten, gemessen auf ein fest definierte Bildbereich, besitzen. In Abbildung 2.13 ist links ein Bild aus einem Fußballspiel zu sehen. Rechts davon ist auf dieses Bild ein Kantenfilte angewendet worden. Außerdem wurde der Text durch rote Rahmen manuell markiert.

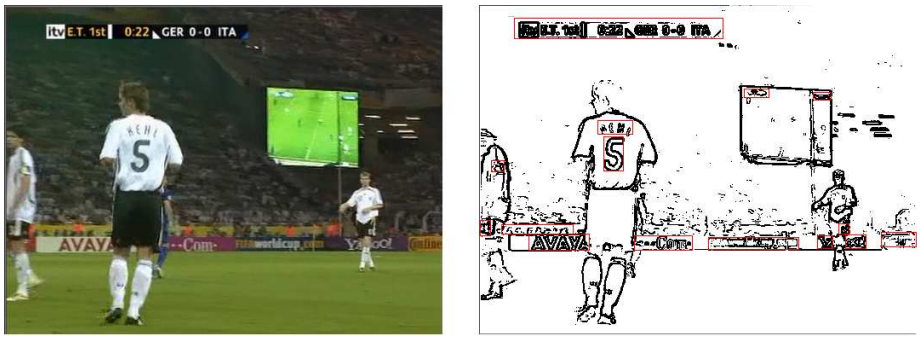


Abbildung 2.13: *Auf dem linken Bild ist eine Szene aus einem Fußballspiel zu sehen. Auf dem rechten Bild ist dieselbe Szene zu sehen, wobei hier ein Kantenfilte angewendet wurde. Durch das Kantenfilte sind die Umrisse aller Objekte im Bild schwarz gezeichnet worden.*

An diesem Beispiel ist zu erkennen, dass der Text im Verhältnis zu den anderen Objekten im Bild eine hohe Dichte an Kantenbildpunkten (schwarze Bildpunkte) aufweist. Durch Ausnutzen dieser Eigenschaften können potentielle Bildbereiche, die Text enthalten, selektiert und weiter analysiert werden. Es gibt aber auch, wie in 2.5 beschrieben, andere Objekte im Bild, die eine ähnliche Kantendichte vorzuweisen haben. In dem Beispiel sind rechts neben der Anzeige einige Lampen erkennbar, die bei dem Kantenbild eine ähnliche Dichte auf-

weisen (siehe Abschnitt 3). Erst durch eine nähere Analyse dieser potentiellen Textbereiche kann endgültig entschieden werden, ob es sich um Text handelt oder nicht. Hierfür müssen weitere Eigenschaften vom Text ausgenutzt werden. Eine dieser Eigenschaft von Text ist die Aneinanderreihung von kleinen Objekten bzw. Schriftzeichen. Diese Objekte sind durch eine kleine Lücke voneinander getrennt, so dass eine Art Kette entsteht. In den meisten Fällen ist die Kette von Objekten in einer Linie. Durch diese Eigenschaft könnte in dem Beispiel festgestellt werden, dass die Lampen aus einem Objekt und nicht aus einer Kette von kleinen Objekten bestehten. Im Folgenden sind weitere Eigenschaften von Text beschrieben, die weiter den Text vom Nicht-Text differenzieren. Auf diese Weise wird die Wahrscheinlichkeit für einen Fehler in der automatischen Detektion reduziert.

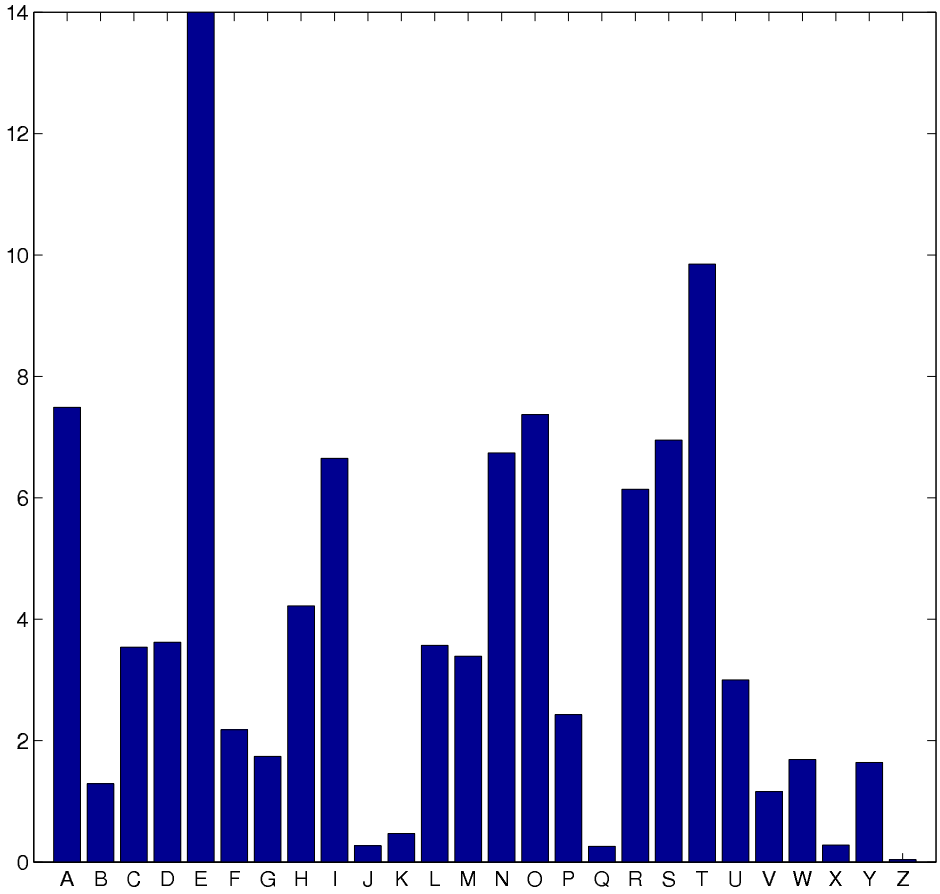


Abbildung 2.14: *Häufig eit der Schriftzeichen in der englischen Sprache.*

Bisher wurden vor allem grobe Eigenschaften von Text betrachtet wie zum Beispiel die Beschaffenheit ganzer Wörter gegenüber dem Rest von einem Bild. Hier wurde mit Hilfe von einer Kantenerkennung und der Berechnung der Dichte von Kanten in einem bestimmten Bereich eine Aussage getroffen, ob es sich um Text oder Nicht-Text handelt. Es existieren im

Detail noch viele weitere Merkmale. Unter anderem weisen Schriftzeichen oft vertikale und horizontale Lienen auf wie zum Beispiel die Schriftzeichen 'T', 'H' oder 'L'. Oft befinde sich in Schriftzeichen Einschlüsse wie zum Beispiel im 'O', 'D' oder 'B'. Wenn sich die Frage stellt, ob es sich bei den potentiellen Kandidaten um einen Textbereich handelt, kann das Auffinde solcher Eigenschaften eine Antwort auf die Frage liefern. Trotz Ausnutzung solcher Eigenschaften bleibt immer ein Restrisiko für die automatische Detektion eines Textbereiches und es werden fälschlich Bereiche erkannt. Es kann lediglich die Wahrscheinlichkeit für eine falsch detektierten Bereich reduziert werden, denn es gibt nahezu unendlich viele Variationen von Werten der Bildpunkte und damit nahezu unendlich viele Zwischenstufen von Text und Nicht-Text. Für die spätere Erkennung der Schriftzeichen sind die Häufig eiten der Schriftzeichen ebenfalls wichtig. Dazu muss die Häufig eit der einzelnen Schriftzeichen in der entsprechenden Sprache, hier englisch, analysiert werden. In Abbildung 2.14 sind in der Veröffentlichung [12] die Schriftzeichen A bis Z über viele Dokumente gezählt und damit die Häufig eit bestimmt worden. Es ist auffällig, dass bestimmte Schriftzeichen wie 'E' oder 'T' sehr viel häufige vorkommen als die Schriftzeichen 'J' oder 'Z'. Mit diesem Wissen kann ein Text besser auf seine Eigenschaften analysiert werden. Als Eigenschaften eines Schriftzeichens können neben den Einschlüssen, oder auch vertikalen und horizontalen Linien, die Art der Ausrichtung betrachtet werden. Es stellt sich die Frage, ob ein Schriftzeichen links oder rechts mehr Gewicht besitzt bzw. ob sie von links oder rechts mehr offen sind.

Wie in Abbildung 2.15 werden alle Schriftzeichen von links und von rechts analysiert. Bildlich gesehen wird jeder Buchstabe nach links oder rechts gekippt, von Balken eingerahmt und von oben geflute bis eine glatte Oberf äche entstanden ist. Danach wird die geflutet Fläche gemessen und für jedes Schriftzeichen gespeichert. Jedes Quadrat entspricht hierbei einem Bildpunkt und wird mit 1 gewertet. Gezählt werden die roten Quadrate. Auf diese Weise kann jedes Schriftzeichen zu drei verschiedenen Kategorien zugeordnet werden: Schriftzeichen die links oder rechts mehr offen und die gleich sind. In der Abbildung 2.15 sind die Schriftzeichen 'n', 'd' und 'b' als Beispiele aufgeführt. Das Schriftzeichen 'n' ist von beiden Seiten nahezu identisch und die Füllmenge liegt mit 0 und 1 fast auf dem gleichen Niveau. Dagegen weisen die Schriftzeichen 'd' und 'b' erhebliche Unterschiede auf. Beide haben die Fühlmenge 0 und 28 sind aber spiegelverkehrt. Da kleine und große Schriftzeichen zueinander unterschiedlich sein können, müssen sie getrennt voneinander betrachtet werden. In Tabelle 2.1 und 2.2 sind die Schriftzeichen mit ihren Eigenschaften aufgelistet.

Art	Schriftzeichen	Anzahl	Häufig eit in %
von links offen	a, d, j, q	4	11,64
von rechts offen	b, c, e, f, g, h, k, p, r, y	10	37,65
gleich	i, l, m, n, o, s, t, u, v, w, x, z	14	50,71

Tabelle 2.1: *Kleine Schriftzeichen in der Englischen Sprache die nach links oder rechts offen sind.*

Bei kleinen Schriftzeichen sowie bei den großen Schriftzeichen ist zu beobachten, dass die Anzahl der nach rechts offenen den links offenen deutlich überwiegt. Für diese Eigenschaft wurde bisher nur die Fläche betrachtet, die der Hintergrund der Schriftzeichen einnimmt. Es sind auch andere Eigenschaften für die Charakterisierung von Text möglich. In

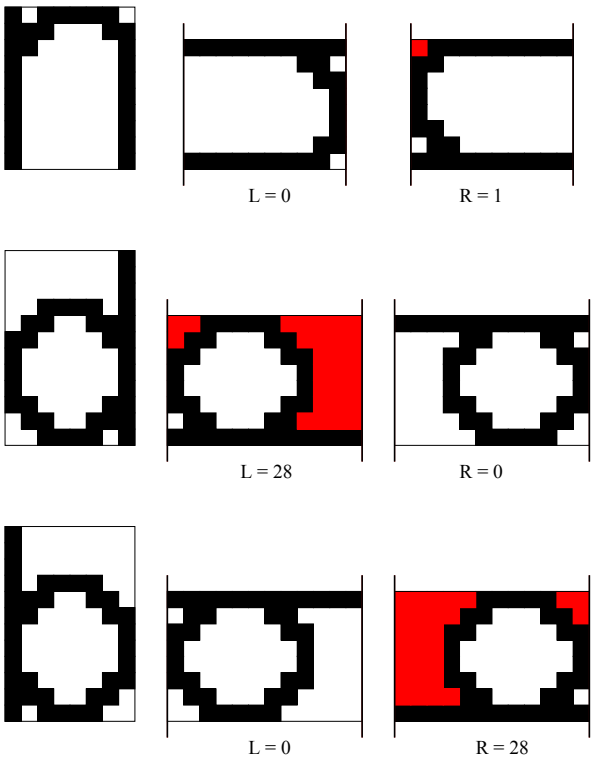


Abbildung 2.15: *Beispiel für links und rechts offene Schriftzeichen: Als Beispiel dienen hier von oben nach unten die Schriftzeichen n, d und b. Für die Veranschaulichung der Bedeutung von links oder rechts offen, werden die Schriftzeichen jeweils um 90° in beide Richtungen gekippt und dann von oben geflutet Der rot markierte Bereich ist die so gefüllte Fläche.*

Art	Schriftzeichen	Anzahl	Häufig eit in %
von links offen	J, Q	2	0,53
von rechts offen	C, E, F, G, K, L, P, R	8	34,07
gleich	A, B, D, H, I, M, N, O, S, T, U, V, W, X, Y, Z	16	65,4

Tabelle 2.2: *Große Schriftzeichen in der englischen Sprache die nach links oder rechts offen sind.*

dem nächsten Beispiel wird nicht die Fläche betrachtet, die von links oder rechts geflute werden kann, sondern die Fläche der Bildpunkte, die das Schriftzeichen selbst einnimmt. Hierfür wird das Schriftzeichen mittig in der Vertikalen getrennt und dann die Fläche vom Schriftzeichen eingenommen, in Form von kleinen Quadraten für die Bildpunkte, rechts und

links gemessen. In der Abbildung 2.16 sind ebenfalls die Schriftzeichen ’n’, ’d’ und ’b’ als Beispiele aufgeführt. Auch hier ist ersichtlich, dass das Schriftzeichen ’n’ in beiden Hälften mit 13 und 15 nahezu identisch viele Bildpunkte besitzt. Bei den Schriftzeichen ’d’ und ’b’ wird wieder ein deutlicher Unterschied mit 15 und 22 ersichtlich.

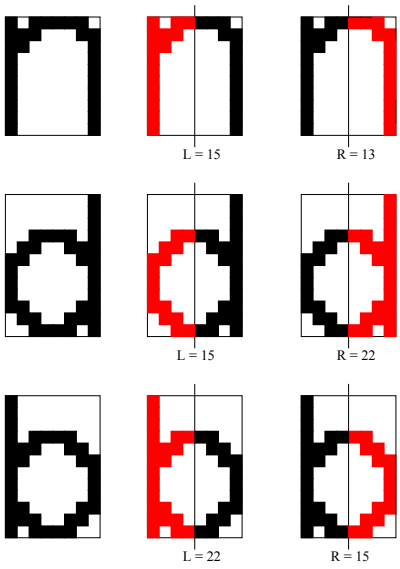


Abbildung 2.16: *Beispiel für links und rechts gewichtete Schriftzeichen: Als Beispiel dienen hier von oben nach unten die Schriftzeichen n, d und b. Für die Veranschaulichung der Bedeutung von links oder rechts gewichtet, wird jedes Schriftzeichen mittig in der Vertikalen gespalten und danach auf beiden Seiten die Menge der eingenommen Bildpunkte gezählt.*

In der Tabelle 2.3 sind die kleinen und großen Schriftzeichen je nach Gewichtung aufgelistet. Diese Auswertung kann in anderen Schriftarten zu leicht anderen Ergebnissen führen. Dennoch ist ein Trend erkennbar. Im Allgemeinen sind die Schriftzeichen in einem englischsprachigen Text mehr nach rechts offen und mehr links gewichtet.

Art	Schriftzeichen	Anzahl
links gewichtet (klein)	b, c, e, h, k, p, r, t, y	9
rechts gewichtet (klein)	a, d, g, j, q	5
gleich (klein)	f, i, l, m, n, o, s, u, v, w, x, z	12
links gewichtet (groß)	B, C, D, E, F, K, L, P, R	9
rechts gewichtet (groß)	J, Q	2
gleich (groß)	A, G, H, I, M, N, O, S, T, U, V, W, X, Y, Z	15

Tabelle 2.3: *Links und rechts gewichtete Schriftzeichen.*

Bisher wurden die Eigenschaften von Schriftzeichen nur von links und rechts betrachtet. Hierbei sind die Gewichtung und die Öffnung von Bedeutung gewesen. Diese Eigenschaften lassen sich natürlich auch auf die Betrachtung von oben und unten anwenden. Dazu werden weitere Eigenschaften in Bezug zueinander wichtig. In Abbildung 2.17 sind die kleinen Schriftzeichen 'a' bis 'z' nebeneinander aufgelistet und mit 3 horizontalen, getrennten Bereichen dargestellt. Es ist zu erkennen, dass die Schriftzeichen sich alle vorwiegend im mittleren Bereich befinde und sich in bestimmten Fällen in den oberen und unteren Bereich ausstrecken. Diese Eigenschaft wird erst im Vergleich zueinander ersichtlich.

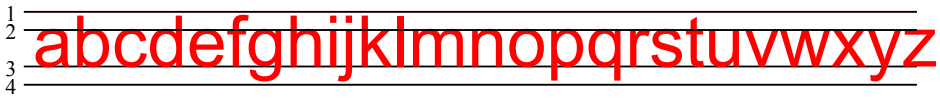


Abbildung 2.17: *Ausdehnung von Schriftzeichen nach oben und unten.*

In Tabelle 2.4 sind die Schriftzeichen nach ihrer Ausdehnung und der Häufig eit im englischsprachigen Text aufgelistet. Mit 25,2% zu 6,34% besitzen die Schriftzeichen im Text überwiegend eine Ausdehnung nach oben.

Art	Schriftzeichen	Anzahl	Häufig eit in %
nach unten gerichtet	g, j, p, q, y	5	6,34
nach oben gerichtet	b, d, f, h, k, l, t	7	25,2
gleich	a, c, e, i, m, n, o, r, s, u, v, w, x, z	14	68,46

Tabelle 2.4: *Kleine Schriftzeichen in der englischen Sprache, die nach oben oder unten gewichtet sind.*

Eine weitere Eigenschaft ist zu erkennen, wenn die beiden Grenzen 2 und 3 zwischen den Bereichen in Abbildung 2.17 betrachtet werden. Werden die Bildpunkt der Schriftzeichen, die auf dieser Grenze liegen, gezählt, können die Schriftzeichen ebenfalls sortiert werden. Genau auf diesen Grenzen liegen bei vielen Schriftzeichen zusätzlich Balken. In Tabelle 2.5 sind die Schriftzeichen nach Anzahl der Bildpunkte auf den Grenzen 2 und 3 getrennt. Aus dieser Tabelle ist eindeutig zu erkennen, dass auf der Grenze 2 mit 37,01% wesentlich mehr Schriftzeichen ein Übergewicht an Bildpunkten besitzen als auf der Grenze 3 mit 3%. Diese Eigenschaft kann für die Bestimmung der Orientierung von Text genutzt werden.

Art	Schriftzeichen	Anzahl	Häufig eit in %
oben	f, h, m, n, r, t, v, w, y	9	37,01
unten	u	1	3
gleich	a, b, c, d, e, g, i, j, k, l, o, p, q, s, x, z	16	59,99

Tabelle 2.5: *Schriftzeichen mit oberen oder unteren Balken.*

2.7 Bilder mit niedriger Qualität

Heute werden die meisten Videos in digitaler Form gespeichert und übertragen. Ein bekanntes Beispiel hierfür ist der so genannte Begriff 'Full HD'. Er steht für 'Full High Definition und bedeutet ins Deutsche übersetzt 'vollständig hochauf ösend'. Hinter diesem Begriff steht eine Definitio der Auf ösung jedes Bildes im Video mit 1.920 x 1.080 Bildpunkten in den Achsen x und y. Diese beiden Zahlen entsprechen dem Verhältnis 16 zu 9. Ebenfalls wird die Anzahl der Bilder pro Sekunde fps mit 24 50 oder 60 definiert Jeder Bildpunkt setzt sich dabei aus 3 Farbkomponenten zusammen, die je Werte zwischen 0 und 255 einnehmen können. Für jede Farbkomponente werden bei 'Full HD' deshalb 8 Bit benötigt, da $log_2(256) = 8$. Daraus ergibt sich die Farbtiefe $c = 3 * 8$. In der Formel 2.1 ergeben sich daraus sehr große Datenmengen mit 311.040.000 Bit/s bei 24 Bilder pro Sekunde und mehr.

$$D = x * y * c * fps \tag{2.1}$$
$$1.194.393.600 = 1.920 * 1.080 * 24 * 24 \tag{2.2}$$
$$2.488.320.000 = 1.920 * 1.080 * 24 * 50 \tag{2.3}$$
$$2.985.984.000 = 1.920 * 1.080 * 24 * 60 \tag{2.4}$$

Abbildung 2.18: *Beispiele für die Auf ösung eines Bildes.*

Diese Datenmenge übersteigt heutige Übertragungsmöglichkeiten von DSL (engl. Digital Subscriber Line) mit ca 8.000.000 Bit/s erheblich. Auf Grund dieses Problems wurde nach Möglichkeiten gesucht die Datenmenge zu reduzieren. Hierfür kann jeder Wert in der Formel 2.1 verkleinert werden. Es kann die Auf ösung von 1920 x 1080 verkleinert werden und damit die Anzahl der Bildpunkte auf derselben Fläche, wodurch die Schärfe des Bildes verringert wird. Weiter kann die Zahl von 8 Bit pro Farbkomponente verkleinert werden, wodurch die mögliche Farbanzahl reduziert wird. Es kann auch die Anzahl der Bilder pro Sekunde, verbunden mit einer geringeren Flüssigkeit der Bewegungen in dem betreffenden Video, verringert werden. In der Abbildung 2.18 ist das gleiche Bild in unterschiedlichen Auf ösungen 1 bis 4 dargestellt. Oben links mit der Nummer 1 befinde sich das Originalbild mit 686 x 514 Bildpunkten und einer Farbtiefe von 24 Bit bzw. 3 x 8 Bit für jede Farbkomponente. In Bild Nummer 2 oben rechts wurde die Farbtiefe auf insgesamt 4 Bit reduziert, wodurch nur noch 16 verschiedene Farben möglich sind. Dagegen wurde bei dem Bild 3 unten links die Farbtiefe beibehalten und dafür die Anzahl der Bildpunkte pro Flächeneinheit auf 20% reduziert. In Bild 4 unten rechts wurde dann die Farbtiefe und die Auf ösung gleichzeitig reduziert. In dem direkten Vergleich der Bilder ist zu erkennen, dass die Qualität der Bilder abhängig ist von der Bitrate. Diese Methoden verringert also in jedem Fall automatisch die Qualität des Videos und widerspricht dem Standart von 'Full HD', welcher die Auf ösung fest definiert Damit dieser Widerspruch behoben werden kann, muss eine Alternative gefunden werden die Datenmenge zu reduzieren. Die Lösung hierfür liegt in der Komprimierung der Daten, wobei zwischen verlustfreier und verlustbehafteter Komprimierung unterschieden werden muss.

2.7.1 Verlustbehaftete Komprimierung

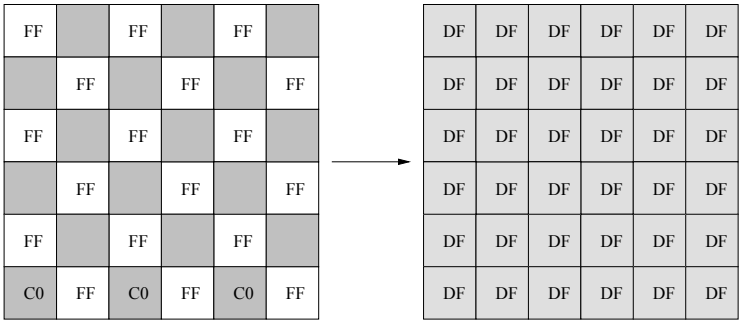


Abbildung 2.19: *Beispiel für verlustbehaftete Komprimierung.*

Eine andere Möglichkeit für die Datenkomprimierung ist, dass ein Verlust von Daten in Kauf genommen wird, wodurch bei der Wiederherstellung des Bildes dennoch eine akzeptable Qualität behalten wird. In Bildern kommt es oft zu großen Flächen mit derselben oder ähnlichen Farben, in denen es vereinzelt Bildpunkte mit ähnlichen Farben gibt. In anderen Fällen wird eine Fläche wie in Abbildung 2.19 durch eine Kombination von Bildpunkten mit verschiedenen Farben erzeugt, die als eine Farbe wahr genommen werden. In diesen Fällen kann die Datenmenge reduziert werden, indem ein Durchschnitt der Farbe in dieser Fläche

berechnet und benutzt wird. Die ursprüngliche Information wird durch diese Vorgehensweise verändert und kann nicht exakt wieder reproduziert werden. Die neue Information ist nun in einem gewissen Maße repräsentativ. Da die Qualität direkt von der Datenkompression abhängt muss ein Maß vorhanden sein, welches dieses Verhältnis beschreibt.

2.7.2 Interlacing

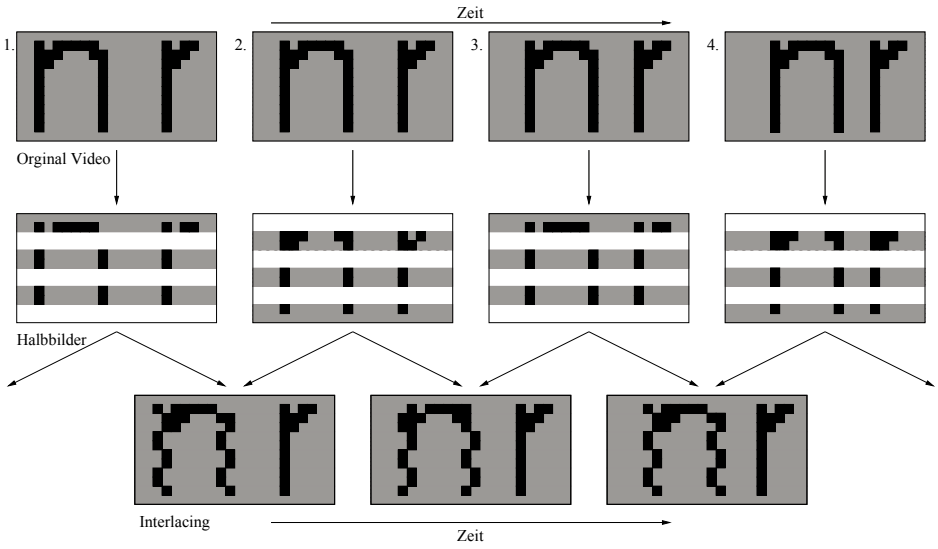


Abbildung 2.20: *Schematische Darstellung für das Verfahren Interlacing: In der oberen Zeile befinde sich die aufeinanderfolgenden Bilder eines Videos. Für die Datenübertragung bzw. Speicherung in der mittleren Zeile wird abwechselnd von jedem Bild nur jede gerade oder ungerade Zeile gespeichert. Für die Rekonstruktion und Darstellung des Videos werden nun in der unteren Zeile immer zwei aufeinanderfolgende Bilder verwendet. Auf diese Weise werden die geraden und ungeraden Zeilen abwechseln miteinander verschränkt.*

Eine weitere Form die Datenmenge zu reduzieren ist das Verfahren Interlacing, auch Zeilensprungverfahren genannt. In diesem Verfahren wird für jedes Bild in dem Video abwechselnd nur jede zweite Zeile übertragen. Auf diese Weise wird die Datenmenge halbiert. In Abbildung 2.20 wird am Beispiel der Bildfolge 1 bis 4 die Funktionsweise der Interlacing-Funktion dargestellt. In den einzelnen Bildern ist jeweils das Schriftzeichen 'n' und 'r' dargestellt. Jedoch bewegt sich das Schriftzeichen 'n' in jedem Bild weiter nach rechts und das Schriftzeichen 'r' behält die Position. Eine horizontale Bewegung entsteht bei einem Kameraschwenk von rechts nach links, wobei sich das gefilmt Objekt, in dem Fall das 'n', nicht bewegt. Dieser Fall tritt ein, wenn eine Szene gefilm wird. Dagegen bewegt sich das 'r' mit der Kamera mit. Dieser Fall ensteht oft, wenn Text nachträglich als Einblendung bzw. Overlay eingefügt wird.

Abbildung 2.21: *Beispiel für das Verfahren Interlacing.*

Im ersten Schritt des Verfahrens Interlacing wird von jedem Bild in dem Video abwechselnd von Bild zu Bild jede zweite Zeile ausgelesen und im zweiten Schritt gespeichert oder übertragen. Für jedes Bild wird dadurch nur die Hälfte der Bildpunkte gespeichert. Deshalb werden die so entstehenden Bilder Halbbilder genannt. Insgesamt wird durch dieses Verfahren die Datenmenge für das Video halbiert. Damit das Bild wieder dargestellt werden kann, müssen nun im dritten Schritt zwei zeitlich aufeinanderfolgende Halbbilder kombiniert werden. Dazu werden die Zeilen von zwei aufeinanderfolgenden Bildern ineinander verschränkt. Aus diesem Beispiel wird sofort ein Problem ersichtlich. Da das Objekt sich in den aufeinanderfolgenden Bildern horizontal bewegt, sind die Umrisse bei der Darstellung eines so gespeicherten und übertragenen Videos verzerrt. Dabei gilt: Je schneller die horizontale Bewegung ist, desto verzerrter wird das Objekt dargestellt. Dagegen sind Objekte, die im Vergleich zur Kamera still stehen, ohne Verzerrung zu erkennen. Der Grund für diese Verzerrung liegt in der zeitlich veränderten Position des Objektes. Diese Form der Datenreduzierung stellt für die Detektion und Erkennung von Schrift ein erhebliches Problem dar. Die zu erkennenden Schriftzeichen können durch die Verzerrung so stark verändert werden, dass sie vollständig unkenntlich werden. In der Abbildung 2.20 ist das ’n’ in der Zeile 3 kaum noch zu erkennen. Der Grund hierfür liegt in den veränderten Konturen des Schriftzeichens. Da der Text meistens nur wenige Bildpunkte groß ist, sind kleine Änderungen bei den Umrissen ausschlaggebend und führen zur nicht korrekten Erkennung des Textes. In Abbildung 2.21 ist ein Bild aus einem Video zu sehen, welches durch Interlacing in der Datenmenge halbiert,

übertragen und wieder hergestellt wurde. In den rot markierten Bereichen ist deutlich eine Verzerrung der Kontur durch die Bewegung zu erkennen. Dagegen sind die Konturen in den grün markierten Bereichen klarer zu erkennen, da diese Objekte sich nicht in dem Video bewegen. Diese Objekte zusammen mit dem Rahmen wurden erst später in das Video eingefügt und sind unabhängig von den Bewegungen der Objekte und der Kamera. In der Abbildung 2.21 ist noch ein weiteres Problem vorhanden. Neben der Bewegung von Objekten im Laufe der Zeit kann sich auch die Beleuchtung in der Zeit verändern. In diesem Fall kam es zu einer schlagartigen Lichtveränderung durch ein Blitzlicht von einem Fotoapparat. So haben sich in kurzer Zeit die Farben in der Szene und damit auch in den zeitlich aufeinanderfolgenden Halbbildern verändert. Diese Farbänderung äußert sich in dem Beispiel in einer Streifenbildung. Dadurch entstehen in dem gesamten Bild hohe Frequenzanteile. Da Text auch hohe Frequenzanteile besitzt ist dieser Effekt für die späteren Detektion und Erkennung von Text problematisch.

2.8 Aufteilung der Arbeit

Prinzipiell muss bei dem Lesen von Text durch den Menschen oder automatisch durch den Computer zwischen dem Auffinde von Text, also dem Detektieren im ersten Schritt, und dem Erkennen der Wörter bzw. genauer den Schriftzeichen, also dem Erkennen im zweiten Schritt, unterschieden werden. Oft unterlaufen dem Menschen und auch dem Computer beim Lesen Fehler, die eine Korrektur benötigen. Der Mensch erkennt solche Fehler durch sein Allgemeinwissen und korrigiert die Fehler dadurch. Dem Computer kann ebenfalls weiteres Wissen gegeben werden. Damit ist es möglich, solche Fehler zu korrigieren. Ein Beispiel hierfür ist ein falsch erkanntes Schriftzeichen innerhalb eines Wortes. Durch den Vergleich dieses Wortes mit einem Wörterbuch kann durch die Berechnung von Unterschieden aller bekannten Wörter eventuell der Fehler behoben werden. Sind zu viele Fehler in dem Wort durch die Erkennung entstanden, sind etliche Alternativen möglich und dann ist das passende Wort nur durch eine Analyse der Grammatik oder sogar des Inhaltes des Textes möglich und die Korrektur des Problems wird sehr komplex. Um diesen Punkt geht es im dritten und entscheidenden Schritt dieser Arbeit. Durch die Aufteilung der Problemstellung ergibt sich eine Teilung der Arbeit in folgende Bereiche:

- Detektion von Textbereichen (siehe Kapitel 3)
- Erkennung von Schriftzeichen (siehe Kapitel 4)
- Korrektur von Fehlern (siehe Kapitel 5)

Das Kapitel Detektion von Textbereichen (siehe Kapitel 3) und Erkennung von Schriftzeichen (siehe Kapitel 4) sind Stand der Forschung und dienen dem Verständnis der Problematik. Die Verfahren für die Detektion und Erkennung von Text basiert auf der von mir erstellten Diplomarbeit und wurden für diese Arbeit erweitert. Das Kapitel Korrektur von Fehlern (siehe Kapitel 5) beinhaltet die Beschreibung und Analyse des im Rahmen dieser Arbeit entwickelte Verfahrens zur Verbesserung der Texterkennung mittels Filterung über die Zeit.

3. Detektion von Textbereichen

In Kapitel 2 wurden viele Eigenschaften von Text in Bildern und Videos und deren Probleme aufgezeigt. Dieses Kapitel beschäftigt sich damit, Bilder oder Videos auf das Vorkommen von Text und deren Position zu bestimmen. Hierfür sind der Inhalt und damit die Erkennung des Textes noch nicht notwendig. Für die Erkennung von Textbereichen sind einige der Eigenschaften, die im Kapitel 2 erklärt wurden, wichtig. Eine der wichtigsten Eigenschaften ist die hohe Frequenzdichte der Bereiche mit Text im Vergleich zum gesamten Bild. Durch die Nutzung dieser Eigenschaft können viele potentielle Bereiche mit Text gefunden werden. Natürlich werden damit auch viele Bereiche erkannt, die die gleiche Eigenschaft haben, aber keinen Text enthalten. Diese fälschlich detektierten Bereiche müssen dann in weiteren Schritten aussortiert werden. In diesem Kapitel wird beschrieben, wie möglichst viele Textbereiche detektiert werden können und damit die Fehlerrate der fälschlich markierten Bereiche möglichst gering bleibt.

3.1 Struktur eines kompletten Systems zur Detektion und Erkennung von Text

Ein Detektions- und Erkennungs-Verfahren bzw. -System für Text kann unterschiedlich gestaltet sein. Die Komplexität kann ebenfalls verschieden groß sein. Ziel von jedem System ist es, den Text in den Bildern und Videos so gut wie möglich zu extrahieren. Ein System kann in verschiedene Teilsysteme gegliedert werden. In jedem Teilsystem können Fehler entstehen. Damit der Fehler innerhalb eines Systems möglichst gering gehalten werden kann, muss der Fehler jedes Teilsystems auf mögliche Fehlerquellen analysiert werden. Zusätzlich zu den Teilsystemen, in denen Fehler entstehen können, kann es auch Teilsysteme geben, die den Fehler wieder reduzieren. Insgesamt muss das Ziel sein, ein möglichst fehlerloses Ergebnis durch die Kombination der Teilsysteme zu erreichen.

3.1.1 Erkennung von Text auf Einzelbildern

In den 90er Jahren wurden Verfahren entwickelt, den Text in wenigen Schritten aus einem Bild oder Video zu extrahieren. In der Literatur werden, unter anderen in [13], [14], [15], [10], [16], [17] und [18] Systeme vorgestellt, welche Text aus einzelnen Bilder aus einem Video extrahieren. Das Prinzip in diesen vorgestellten Verfahren basiert immer auf der gleichen Struktur. In Abbildung 3.1 ist dieses System dargestellt.

In einem klassischen System zur Extraktion von Text aus einem Bild oder Video muss zuerst das Bild dekodiert werden. Dementsprechend muss auch ein Bild nach dem anderen aus einem Video dekodiert werden. Mit den Rohdaten des Bildes kann eine Detektion von Text durchgeführt werden. Die Detektion von Text basiert auf einem Wavelet-Verfahren oder einem Kantenerkennungs-Verfahren. Entscheidend sind hier hohe Frequenzen für die Existenz von Text. Sind die Koordinaten von einem potentiellen Bereich für Text ermittelt, wird

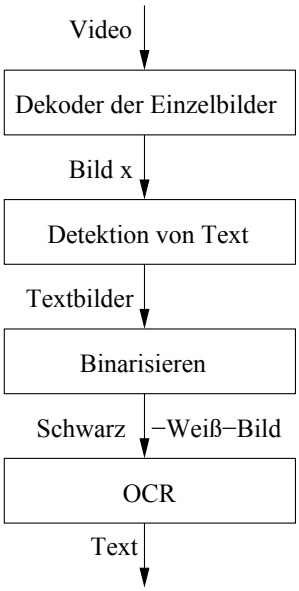


Abbildung 3.1: *In einem klassischen Detektions- und Erkennungssystem für Text gibt es vier einzelne Schritte, die aus dem Video den Text extrahieren. Die Dekodierung der Einzelbilder, die Detektion von Text, die Binarisierung der Bilder und das Erkennen der Schriftzeichen (OCR). Die entscheidenden Elemente des Systems sind dabei die Detektion von Text und die Erkennung der Schriftzeichen.*

dieser im nächsten Schritt in ein Schwarz-Weiß-Bild konvertiert. Das Ziel dabei ist, dass der Text mit Schwarz von allen anderen Elementen in dem Bild mit Weiß getrennt wird. Dieses entstehende Bild ist dann vergleichbar mit einem reinen Textdokument, welches durch eine Schreibmaschine entstanden ist. Im letzten Schritt werden dann die verschiedenen Objekte in dem Bild als Schriftzeichen interpretiert. Bildlich gesehen wird jedes Objekt in dem Schwarz-Weiß-Bild mit einem Modell eines Schriftzeichens, welches im Vorfeld durch viele Beispiele trainiert wurde, verglichen. Das Modell, welches die größte Übereinstimmung mit dem in dem Schwarz-Weiß-Bild vorhandenen Objekt besitzt, beschreibt das Objekt in diesem Bild. In jedem Teilsystem können hier Fehler entstehen. Bei der Dekodierung von verlustbehafteten, komprimierten Videos können nicht alle Daten vom Orginalvideo vollständig rekonstruiert werden. Diese niedrigere Qualität der resultierenden Bilder kann zu Fehlern führen. Im Teilsystem der Detektion von Textbereichen in dem Video können Bereiche übersehen werden oder auch Bereiche ausgewählt werden, die Ähnlichkeit mit Text besitzen, aber kein Text enthalten, welches später zu Fehlern führt. Durch die Konvertierung der Bilder in ein Schwarz-Weiß-Bild, bei der der Text von anderen Elementen im Bild getrennt werden soll, können ebenfalls viele Fehler entstehen, da die Zuordnung der Bildpunkte von Text und anderen Elementen nicht immer eindeutig ist. In der Erkennung der Schriftzeichen können nur solche erkannt werden, die durch ein Modell beschrieben sind. Befinde sich noch weitere

Schriftzeichen oder sogar andere Objekte in dem Schwarz-Weiß-Bild, werden diese ebenfalls als eines der vorhandenen Modelle und damit als falsches Schriftzeichen interpretiert. In diesem einfachen System sind keine Teilsysteme vorhanden, die Fehler korrigieren.

3.1.2 Erkennung von Text auf mehreren aufeinanderfolgenden Bildern

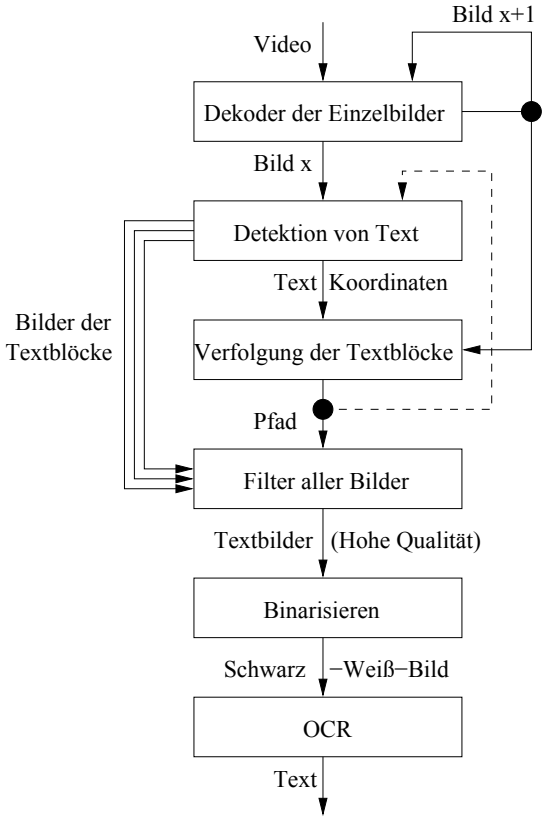


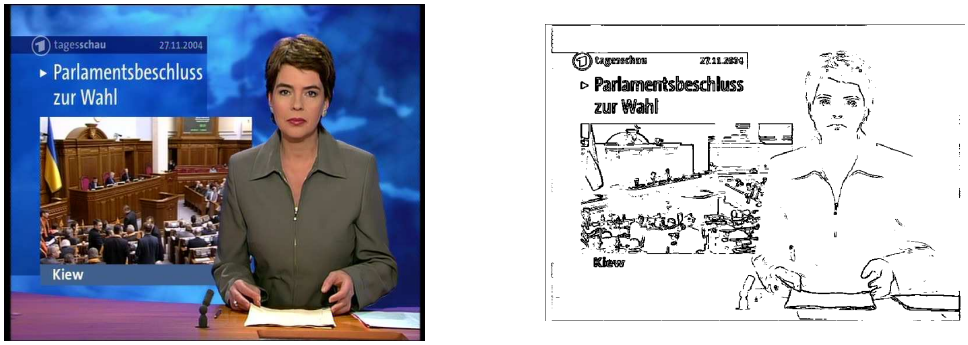
Abbildung 3.2: *Ähnlich wie in Abbildung 3.1 wird das Video erst dekodiert und auf den einzelnen Bildern eine Detektion von Text durchgeführt. Nach der Detektion der einzelnen Textblöcke in den Einzelbildern werden diese in den nächsten Bildern in dem Video verfolgt. Die so gefundenen Koordinaten der Textblöcke können optional die weitere Detektion von Text erleichtern und sind hier mit einem gestrichelten Weg gekennzeichnet. Nachdem die Einzelbilder von einem zu erkennenden Text ermittelt wurden, müssen diese in einem Filter zu einem Bild mit hoher Qualität vereinigt werden. Ist ein Bild mit hoher Qualität erzeugt, kann wie bei dem einfachen Verfahren eine Binarisierung und eine anschließende Erkennung von Schriftzeichen (OCR) durchgeführt werden.*

Eine weitere Möglichkeit, die Rate der Erkennung von Text zu verbessern bzw. die Fehler zu reduzieren, ist die Verwendung von mehreren aufeinanderfolgenden Bildern. Auf diese Weise wird versucht, die Qualität von Bildern, welche ein Wort enthalten, und auch damit die Rate der Schrifterkennung zu steigern. In Abbildung 3.2 ist das System aus Abbildung 3.1 um 2 Teilsysteme und mehrere neue Pfade erweitert. Nach der Detektion von Text muss dieser in den nächsten Bildern wieder gefunden werden. Hierfür wird eine Verfolgung der Textblöcke benötigt. Die Verfolgung der Textblöcke sucht in den jeweiligen nächsten Bildern in dem Video ein Bildbereich, der dem Bild vom Text in den davor liegenden Bildern möglichst nahe entspricht. Ist der Pfad, den das Bild durch die aufeinanderfolgenden Bilder im Video und damit die Koordinaten ermittelt, wird ein Filter für die Fusion der Bilder benötigt. Der Bildbereich von dem zu erkennenden Text kann in den aufeinanderfolgenden Bildern erheblich verändert werden. Einige Faktoren davon sind Position im Videobild, Größe, Rotation, Perspektive, Kontrast und Beleuchtung des Bildbereichs. Diese Bildveränderungen sind für die Erzeugung eines repräsentativen Bildes mit hoher Qualität für einen Text über mehrere auf einanderfolgende Bilder im Video zu beachten. Für die Detektion von Text können die Informationen aus der Verfolgung der Textblöcke und umgekehrt genutzt werden. Hiefür ist ein zusätzlicher Weg in Abbildung 3.2 als gestrichelter Weg eingetragen.

3.2 Detektion von Text durch Kantenerkennung

3.2.1 Kantenerkennung nach Canny

Schriftzeichen besitzen in Bildern eine hohe Kantendichte gegenüber Personen und anderen Objekten [10, 16, 19–21]. Dadurch können die Bereiche, in denen Text vorhanden ist, gefunden werden. Dazu ist eine Kantenerkennung notwendig. Kanten in Bildern sind Bereiche mit einem hohen Kontrast. Dieser Kontrast ist durch einen hohen Intensitätswechsel von einem Bildpunkt zum nächsten erkennbar[22]. In Abb. 3.3 ist ein Schlüsselbild mit dem resultierenden Kantenbild erkennbar.



(a) (b)

Abbildung 3.3: *Bild a zeigt ein Schlüsselbild. In Bild b befinde sich das dazugehörige berechnete Kantenbild.*

Die Kantenerkennung 3.4 kann auf die verschiedenen Farbwerte rot, grün oder blau an-

gewendet werden. Auf diese Art können verschiedenfarbige Buchstaben als Kontraste zu dem Hintergrund gefunden werden 3.5. Die zu erkennenden Zeichen in den zu untersuchenden Nachrichtenvideos sind ausschließlich weiß. Weiß beinhaltet alle Farbkomponenten. Damit ist es nicht notwendig, alle Farbkomponenten einzeln zu untersuchen. Deshalb kann das Bild vor der Kantenerkennung in ein Graustufenbild umgewandelt werden. Durch die Umwandlung in ein Graustufenbild werden die verschiedenen Farbkomponenten rot, grün und blau weiter berücksichtigt. Der weitere Aufwand ist damit um 2/3 reduziert.

Abbildung 3.4: *Das Bild wurde durch einen Algorithmus für Kantenerkennung berechnet. Eine gefundene Kante wurde als schwarz (Binär 1) und andererseits als weiß (Binär 0) abgespeichert.*

Abbildung 3.5: *In dem Bild sind Bereiche mit einer hohen Kantendichte (schwarze Bildpunkte) zu erkennen. Diese Bereiche sind durch Rechtecke eingerahmt und enthalten Text.*

Bevor die eigentliche Kantenerkennung durchgeführt wird, kann das Rauschen in einem Bild durch ein mathematisches Filter reduziert werden. Dazu wird eine diskrete Gaußmaske benutzt. Diese Gaußmaske ist 5x5 Werte groß (Abb. 3.6).

$$\frac{1}{115}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Abbildung 3.6: *Die diskrete Gaußmaske über 5x5 Bildpunkte wird für die Rauschreduktion benutzt.*

Diese Maske wird dann über jeden Bildpunkt P gelegt, so dass gleichzeitig in jede Richtung 2 Bildpunkte aufgenommen werden. Jeder dieser Matrixelemente ist der Multiplikator für die Bildpunkte, welche anschließend durch 115 dividiert werden. Die Summe aller 25 berechneten Punkte ergibt den neuen Bildpunkt für das rauschreduzierte Bild. Die Formeln für die Berechnung des neuen Bildpunktes P' lauten:

$$\begin{aligned}
P'_1 &= \frac{P_{x,y} * 2}{115} + \frac{P_{x,y+1} * 4}{115} + \frac{P_{x,y+2} * 5}{115} + \frac{P_{x,y+3} * 4}{115} + \frac{P_{x,y+4} * 2}{115} \\
P'_2 &= \frac{P_{x+1,y} * 4}{115} + \frac{P_{x+1,y+1} * 9}{115} + \frac{P_{x+1,y+2} * 12}{115} + \frac{P_{x+1,y+3} * 9}{115} + \frac{P_{x+1,y+4} * 4}{115} \\
P'_3 &= \frac{P_{x+2,y} * 5}{115} + \frac{P_{x+2,y+1} * 12}{115} + \frac{P_{x+2,y+2} * 15}{115} + \frac{P_{x+2,y+3} * 12}{115} + \frac{P_{x+2,y+4} * 5}{115} \\
P'_4 &= \frac{P_{x+3,y} * 4}{115} + \frac{P_{x+3,y+1} * 9}{115} + \frac{P_{x+3,y+2} * 12}{115} + \frac{P_{x+3,y+3} * 9}{115} + \frac{P_{x+3,y+4} * 4}{115} \\
P'_5 &= \frac{P_{x+4,y} * 2}{115} + \frac{P_{x+4,y+1} * 4}{115} + \frac{P_{x+4,y+2} * 5}{115} + \frac{P_{x+4,y+3} * 4}{115} + \frac{P_{x+4,y+4} * 2}{115} \\
P'_{x+2,y+2} &= P_1 + P_2 + P_3 + P_4 + P_5
\end{aligned} \tag{3.1}$$

Die Koordinaten x und y stehen für die aktuelle Position der Maske im Bild. Auf diese Weise wandert die Maske über das ganze Bild und berechnet jeden Bildpunkt neu. Nach der Glättung bzw. der Rauschreduzierung werden die Kantenstärken im Bild durch die Berechnung der Gradienten gefunden. Dafür wird ein Sobel-Operator zur räumlichen Gradientenmessung auf das Bild angewendet. So kann die absolute Gradientengröße für jeden Bildpunkt gefunden werden. Der Sobel-Operator benutzt ein Paar von 3x3 Faltungsmasken. Die erste Maske berechnet den Gradienten in der X-Achse (Abb. 3.7). Die zweite Maske berechnet den Gradienten in der Y-Achse (Abb. 3.8).

Die Sobel-Masken werden dann auf jeden Bildpunkt P angewendet, so dass die benachbarten Bildpunkte mit in die Berechnung aufgenommen werden. Die Koordinaten x und y

−1	0	+1
−2	0	+2
−1	0	+1

Gx

Abbildung 3.7:
Die Sobel-Maske wird zur Berechnung der Gradienten in der horizontalen Richtung (X-Achse) benutzt.

+1	+2	+1
0	0	0
−1	−2	−1

Gy

Abbildung 3.8:
Die Sobel-Maske wird zur Berechnung der Gradienten in der vertikalen Richtung (Y-Achse) benutzt.

repräsentieren die Position der Sobel-Masken im Bild. Die Werte der Masken sind die Faktoren für die Bildpunkte, welche danach addiert werden zu G_x und G_y. Die Formeln lauten:

$$\begin{aligned} G_x &= -P_{x,y} + P_{x+2,y} - 2P_{x,y+1} + 2P_{x+2,y+1} - P_{x,y+2} + P_{x+2,y+2} \\ G_y &= P_{x,y} + 2P_{x+1,y} + P_{x+3,y} - P_{x,y+2} - 2P_{x+1,y+2} - P_{x+2,y+2} \end{aligned} \tag{3.2}$$

Die Kantenstärke G_k errechnet sich aus der Summe der Beträge von G_x und G_y.

$$G_k = |G_x| + |G_y| \tag{3.3}$$

Falls die Summe eine Größe von 255 überschreitet, wird sie durch 255 ersetzt. Dieser Schritt ist notwendig, da das Bild nur maximal 256 Farbwerte annehmen kann (0 bis 255).

Nachdem die Gradienten der Kantenstärken in der X- und Y-Achse gefunden sind, kann die Richtung der Kanten berechnet werden. Der Winkel θ für die Richtung errechnet sich durch:

$$\theta = \arctan(\frac{G_x}{G_y}) \tag{3.4}$$

Es muss darauf geachtet werden, dass G_y nicht den Wert Null annimmt. Falls dieser Fall eintritt, muss eine Sonderbehandlung durchgeführt werden. Immer wenn der Gradient in der X-Achse Null annimmt, muss die Richtung der Kante zwischen 90° und 0° liegen in Abhängigkeit von dem Wert, welchen der Gradient G_y besitzt. Ist der Gradient G_y gleich Null, ist die Richtung der Kante ebenfalls 0°. Falls der Gradient G_y einen anderen Wert als Null besitzt, muss die Richtung der Kante genau 90° betragen.

Als Nächstes muss die Ausrichtung auf die Bildpunkte im Bild (Abb. 3.9) übertragen werden.

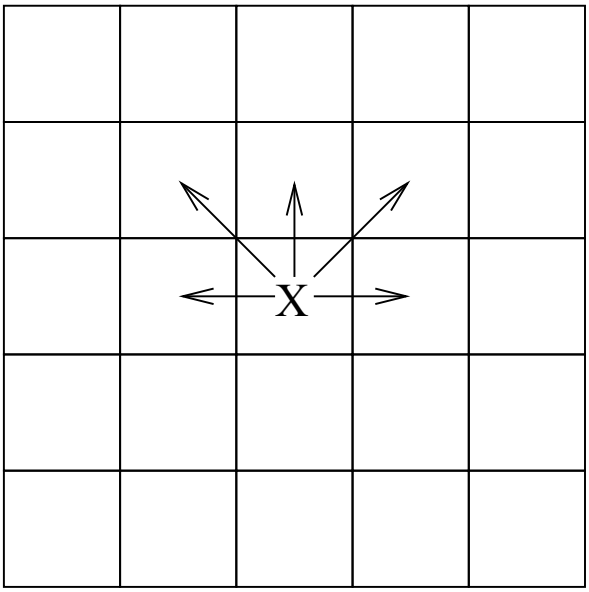


Abbildung 3.9: *Die Kante kann vier Ausrichtungen in der Bildstruktur besitzen (horizontal, vertikal, positiv und negativ diagonal).*

Es ist eine horizontale, eine positiv und negativ diagonale und eine vertikale Ausrichtung möglich. Die horizontale Ausrichtung besitzt dabei die Winkel 0° und 180°. Die positive Ausrichtung besitzt 45° und die negative Ausrichtung 135°. Die vertikale Ausrichtung besitzt 90°. Da das Zwischenergebnis alle möglichen Winkel ergeben kann, muss an dieser Stelle eine Diskretisierung vorgenommen werden. Das heißt, dass die Winkel in vier Bereiche zusammengefasst werden. Dadurch werden die Kanten zu den Bildpunkten zugeordnet. Für die Zuordnung in Bereiche werden Winkelgrenzen wie folgt benutzt. In dem horizontalen Bereich werden alle Winkel zwischen 0° bis 22,5° und 157,5° bis 180° eingeordnet. Die positiv diagonalen Winkel liegen im Bereich von 22,5° bis 67° und die negativ diagonalen Winkel liegen im Bereich von 112,5° bis 157,5°. Der vertikale Winkel liegt im Bereich von 67,5° bis 112,5°(Abb. 3.10).

Als letzter Schritt werden zwei Grenzwerte $T1$ und $T2$ eingeführt, die die Fehler durch das Rauschen in den Bildern reduzieren. Falls der errechnete Wert für die Kante den Grenzwert $T1$ überschreitet, wird dieser Bildpunkt als mögliche Kante markiert. Falls der Wert des Bildpunktes den Grenzwert $T2$ überschreitet, wird dieser als Kante gespeichert. Jeder benachbarte Bildpunkt, der über dem Grenzwert $T1$ liegt, wird ebenfalls als Kante gespeichert. Es muss also mindestens ein Bildpunkt in einem zusammenhängenden Bereich liegen, der über den Grenzwert $T2$ kommt, damit dieser Bereich als Kante gespeichert wird. Dadurch werden schwache Kanten als durchgehende Linien gespeichert, und es entsteht keine gestrichelte Linie.

Die Bildpunkte mit einer Kante werden mit einer 0 gespeichert und die Bildpunkte ohne eine Kante werden mit einer 1 gespeichert. Das so entstehende Bild beinhaltet jetzt nur noch

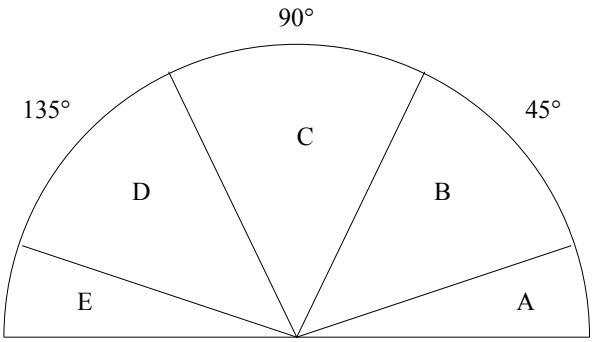


Abbildung 3.10: *Die Kante nimmt diskrete Wickel für die Ausrichtung an. Die vier Bereiche beinhalten die horizontale (A und E), positiv diagonale (B), negativ diagonale (D) und vertikale Ausrichtung (C).*

die Informationen über die Kanten des Orginalbildes.

3.2.2 Skalierung um Faktor 2 für Kantenerkennung

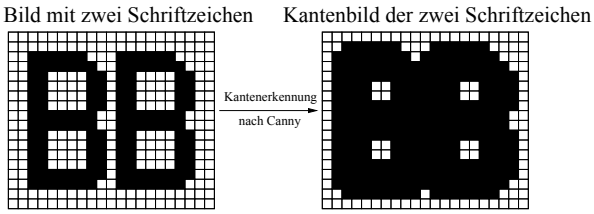


Abbildung 3.11: *Auf der linken Seite befinde sich ein Bild mit zwei Schriftzeichen „BB“. Auf der rechten Seite ist aus dem Bild ein Kantenbild nach dem Verfahren von Canny erzeugt worden.*

Besteht ein ausreichender Unterschied der Farben zwischen zwei benachbarten Bildpunkten innerhalb eines Bildes liegt eine Kante vor. Durch die Kanntenerkennung nach Canny müssen in diesem Fall beide Bildpunkte in einem neuen Bild als Kante markiert werden. Hier entsteht ein großer Nachteil für die spätere Detektion von Schriftzeichen bzw. Text. Befin den sich Schriftzeichen in dem Bild sehr dicht beieinander, kann es in dem Kantenbild dazu kommen, dass die Kanten von verschiedenen Schriftzeichen zusammenhängen. In diesem Fall können die Schriftzeichen nicht mehr von einander separiert werden und die Detektion und Erkennung kann nicht funktionieren. In Abbildung 3.11 ist auf der linken Seite ein Bild mit zwei sehr dicht aneinanderliegenden Schriftzeichen „BB“ zu erkennen. Die beiden Schriftzeichen trennt nur ein Bildpunkt. Auf der rechten Seite befinde sich das entsprechende Kantenbild, erzeugt nach dem Verfahren von Canny. Die beiden Schriftzeichen fließe in dem Kantenbild ineinander über und sind nicht voneinander zu trennen.

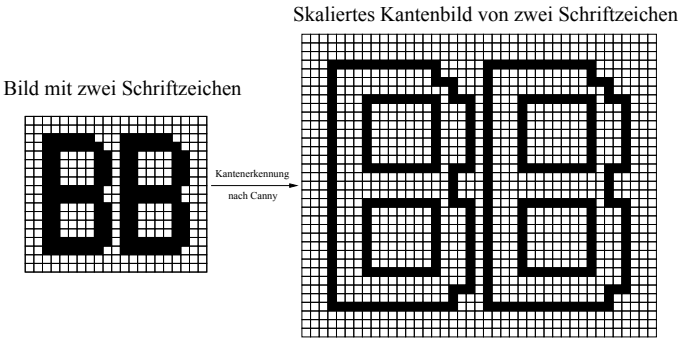


Abbildung 3.12: *Auf der linken Seite ist wieder das Bild der linken Seite aus der Abbildung 3.11 zu erkennen. Auf der rechten Seite wurden die Kantenbildpunkte mit Hilfe einer Verdoppelung der Auf ösung zwischen den ursprünglichen Bildpunkten gespeichert.*

Unterscheiden sich zwei Bildpunkte so stark, dass eine Kante erkannt wird, befinde sich die Kante theoretisch genau zwischen diesen beiden Bildpunkten. Damit die Kante genau zwischen diesen beiden Bildpunkte gespeichert werden kann, muss die Anzahl der Bildpunkte von dem zu erzeugenden Kantenbild vergrößert werden. Hier werden die horizontale und die vertikale Anzahl der Bildpunkte verdoppelt. Genau genommen ist die Anzahl der Bildpunkte in der Horizontalen und Vertikalen des Kantenbildes nicht exakt doppelt so groß. Da nur ein Bildpunkt je zwei aneinander grenzende Bildpunkte des Orginalbildes dazu kommt, fehlt in der Horizontalen und Vertikalen je ein Bildpunkt für eine echte Verdoppelung. In Abbildung 3.12 ist das gleiche Bild von der linken Seite der Abbildung 3.11 durch das Verfahren nach Canny als Kantenbild erzeugt worden. Der Unterschied liegt hier darin, das alle festgestellten Kanten nicht auf beide betreffenden Bildpunkte gespeichert worden sind, sondern in dem Zwischenraum, welcher durch die Verdoppelung der Auf ösung entstanden ist. Auf diese Weise ist auch nach der Kantenerkennung eine Separierung von Text bzw. Schriftzeichen möglich. Ein ähnliches Verfahren wurden in [23] und [24] veröffentlicht.

3.2.3 Geeignete morphologische Bildoperatoren

Damit die Textbereiche in den Kantenbildern gut erkannt werden ist es notwendig, dass möglichst wenig Fragmente neben den Texten enthalten sind. Die Reduzierung dieser Fragmente kann durch morphologische Bildoperatoren erreicht werden. Für diesen Zweck werden drei verschiedene morphologische Bildoperatoren verwendet. Alle drei morphologischen Bildoperatoren funktionieren auf eine ähnliche Weise. Es wird ein quadratisches Fenster in der Größe von 3 mal 3 Bildpunkten über das gesamte Kantenbild geschoben, so dass alle darin befindliche Bildpunkte untersucht werden. Jeder Bildpunkt, der in der Mitte dieses Fenster liegt, wird bei dem Verschieben des Fensters untersucht. Dabei wird entschieden, ob der Bildpunkt weiterhin als Kante endgültig anerkannt werden kann.

Zwischen den Schriftzeichen in Wörtern kann es zu Verbindungen bzw. Brücken durch Fragmente kommen. Diese meist kleinen Verbindungen können durch den ersten morpholo-

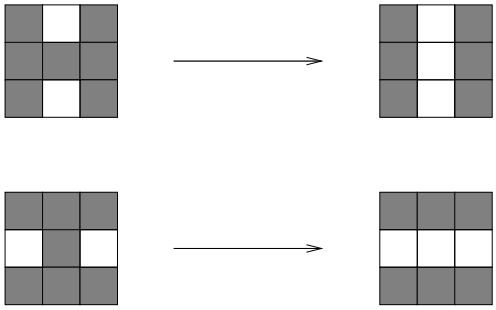


Abbildung 3.13: *Die Brücken zwischen den Kanten werden durch einen morphologischen Bildoperator in der Vertikalen und in der Horizontalen entfernt. Kantenbildpunkte sind durch eine graue Farbe gekennzeichnet.*

gischen Bildoperator entfernt werden 3.13. Befinde sich zwei Kanten parallel zueinander in einem Bild mit dem Abstand von einem Bildpunkt und befinde sich eine Verbindung zwischen den parallelen Kanten durch einen Kantenbildpunkt, wird durch den ersten morphologischen Bildoperator die Brücke zwischen den Kanten gelöscht. Die Operationen werden in der Vertikalen und in der Horizontalen durchgeführt.

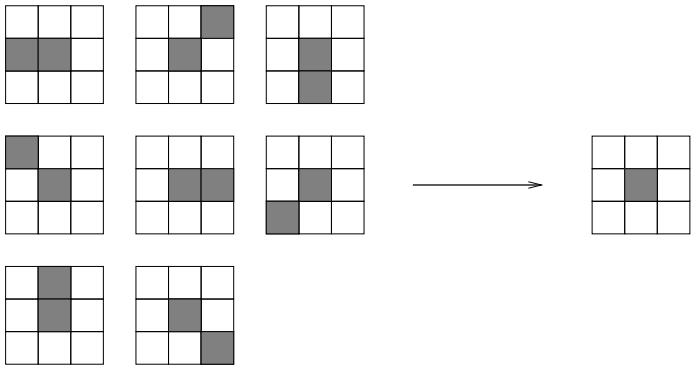


Abbildung 3.14: *Eng beieinander stehende Kantenbildpunkte werden entfernt. Kantenbildpunkte sind durch eine graue Farbe gekennzeichnet.*

Schriftzeichen besitzen in den Kantenbildern eine abgeschlossene Form und sind kompakt. Sie besitzen damit mindesten zwei Kantenbildpunkte in einer Schnittstelle. Es gibt Kanten, die als Linie im Raum enden und für die spätere Erkennung hinderlich sind. Durch diese Eigenschaft können weitere Fragmente aus dem Kantenbild entfernt werden 3.14. Dafür wird ein morphologischer Bildoperator verwendet, der eng beieinander stehende Kanten entfernt. Besitzt ein Kantenbildpunkt maximal ein Kantenbildpunkt als direkten Nachbar, wird er durch den morphologischen Bildoperator gelöscht.

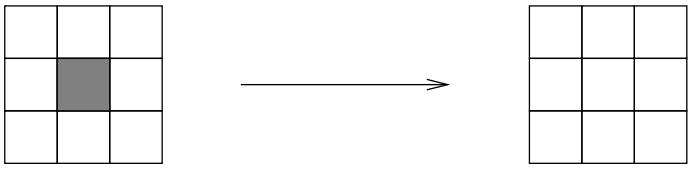


Abbildung 3.15: *Einzelne Kantenbildpunkte werden entfernt. Kantenbildpunkte sind durch eine graue Farbe gekennzeichnet.*

Dieser Operator wird mehrfach auf das gesamte Kantenbild angewendet. Dadurch können schwache Linien mit großer Sicherheit entfernt werden 3.15. Es bleiben kompakte Objekte übrig, die für die weiteren Untersuchungen von Bedeutung sind. Ein Beispiel für solche uninteressanten Linien sind die Unterstreichungen von Überschriften. Der Bildoperator kann so oft aufgerufen werden, bis keine Veränderung mehr stattfindet In den eigenen Experimenten hat sich gezeigt, dass 100 Aufrufe ausreichend sind 3.16 3.17.

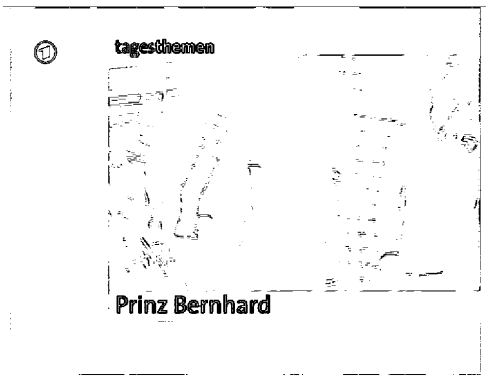


Abbildung 3.16:

In der Abbildung befinde sich ein Kantenbild vor der Anwendung der morphologischen Bildoperatoren. Es sind viele kleine Fragmente in dem Bild erkennbar. Die Kanten oben und rechts sind durch ein Fehler in der Übertragung durch das Terrestrische Fernsehen entstanden.

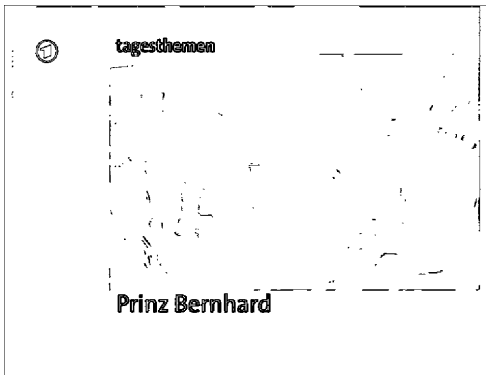


Abbildung 3.17:

In der Abbildung befinde sich ein Kantenbild nach der Anwendung der morphologischen Bildoperatoren. Die Kantendichte der Schriftzeichen ist gleich groß geblieben und die Kantendichte der anderen Bildbereiche ist reduziert. Dadurch ist die Erkennung von hohen Kantendichten von Schriftzeichen verbessert.

Durch Rauschen in den I-Bildern kann es zu der Erkennung von kleinen Kanten kommen, welche nur einen Bildpunkt groß sind. Damit diese Bildpunkte gelöscht werden können, wird ein weiterer morphologischer Bildoperator eingeführt. Er löscht Kantenbildpunkte, die keinen direkten Nachbarn besitzen. Dadurch sinkt die falsche Erkennung von dichten Kantenbereichen (vgl. 3.2.4).

3.2.4 Detektion von Textbereichen in Bildern

Die Kantenerkennung hat ein Schwarz-Weiß-Bild erzeugt. Schwarze Bildpunkte repräsentieren in diesem Bild Kanten. Objekte mit vielen Strukturen erzeugen in dem Kantenbild entsprechend viele schwarze Bildpunkte. Dagegen besitzen Objekte mit einer großen eintönigen Fläche kaum Kanten und damit wenige schwarze Bildpunkte. Eine Kante wird dann gefunden, wenn ein abrupter Wechsel in der Farbe stattfindet Schriftzeichen sind unter den gegebenen Bedingungen kleine detailreiche Objekte, die sich durch ihre Farbe von dem Hintergrund unterscheiden. Deshalb besitzen diese Schriftzeichen in den berechneten Kantenbildern eine große Kantendichte und heben sich gegenüber den meisten anderen Objekten hervor.

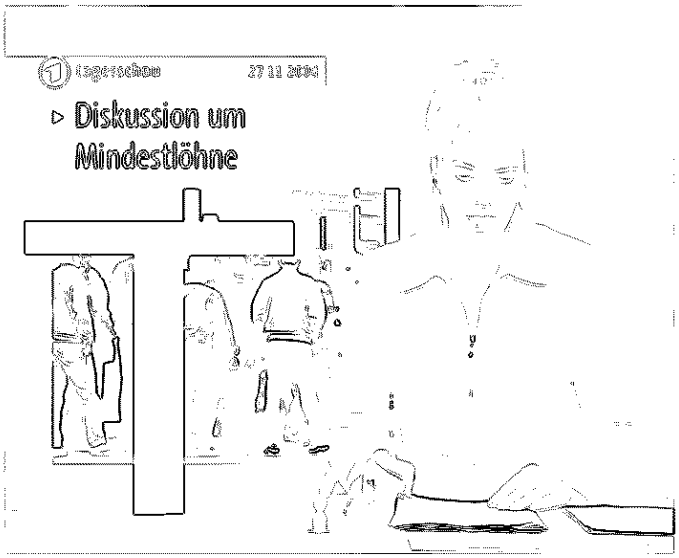


Abbildung 3.18: *Das durch Kantenerkennung berechnete Bild enthält verschieden große Schriftzeichen. Das große Schriftzeichen 'T' besitzt im Bezug zu seiner Größe weniger Kantenbildpunkte als die kleineren Schriftzeichen (Diskussion um Mindestlöhne). Die Personen im Hintergrund besitzen mehr Kantenbildpunkte als das Schriftzeichen 'T'.*

Text kann in allen Größen in dem Bild vorkommen (Abb. 3.18). Ein Buchstabe kann das ganze Bild ausfüllen oder auch nur wenige Bildpunkte groß sein. Für die vorliegende Aufgabe muss der zu suchende Text in der Größe der Schriftzeichen eingeschränkt werden, da eine Reihe von Parametern auf diese Größe eingestellt werden. Das Auffinde aller Größen von Schriftzeichen ist nicht mit Hilfe von einem Parametersatz möglich, da sich die Relationen der Kantendichten zu anderen Objekten in dem Bild deutlich verändern. Zum Beispiel haben sehr große Schriftzeichen in dem Bild weniger Kanten als eine gleichzeitig im Vordergrund stehende Person. Für die vorgesehene Aufgabe sind die Größen der Schriftzeichen in den Titeln der Nachrichtenbeiträge ausschlaggebend. Wenn die Größe der Schriftzeichen sich in einem vorher festzulegenden Rahmen befindet können deutlich größere bzw. kleinere Ob-

jekte ausgeschlossen werden (vgl. 3.2.4). Durch mehrere Arbeitsschritte mit verschiedenen Parametersätzen kann also jeder Text gefunden werden.

Auffinden von hohen Kantendichten

Der erste Arbeitsschritt für die Detektion von Text ist das Auffinde von Bereichen mit hohen Kantendichten im Kantenbild E. Dazu müssen kleine Bildabschnitte auf die Anzahl von Kantenbildpunkten untersucht werden. An dieser Stelle bietet sich ein quadratisches Fenster von der Kantenlänge W_{size} an, in dem die Kantenbildpunkte gezählt werden. Erreicht die Anzahl von Kantenbildpunkten eine vordefiniert Schwelle E_{max}, wird dieser Bereich als hohe Kantendichte markiert. Die Anzahl der Kantenbildpunkte E_w für ein Fenster errechnet sich aus der Gleichung:

$$E_w = \sum_{m=0}^{W_{size}} \sum_{n=0}^{W_{size}} E(m,n) \tag{3.5}$$

Das quadratische Fenster wird dann über das komplette Kantenbild geschoben, so dass jeder Bereich in dem Bild untersucht wird. Wenn die Fenster aneinander angrenzend gelegt werden, können Bereiche mit einer hohen Kantendichte übersehen werden (Abb. 3.19).

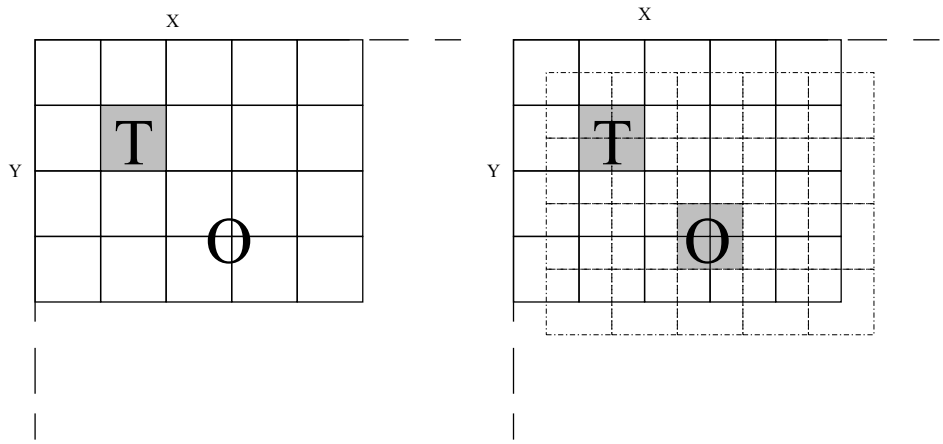


Abbildung 3.19: *Das Bild links zeigt eine Aneinanderreihung der Fenster zum Auffinde von hohen Kantendichten. Dadurch wird das Schriftzeichen O nicht gefunden. Erst durch die Überlappung wie im rechten Bild wird auch das Schriftzeichen O gefunden.*

Der Grund dafür ist, dass ein Objekt mit hoher Kantendichte zur Hälfte in einem Fenster und zur anderen Hälfte im nächsten Fenster liegt. Dadurch kann der Schwellenwert E_{max} nicht erreicht werden. Wenn das Objekt genau in einem Fenster liegt ist der Schwellenwert E_{max} überschritten. Damit dieser Fehler behoben werden kann, ist eine Überlappung der Fenster notwendig. Das heißt, dass jedes Fenster in die benachbarten Fenster hinein verschoben ist. Diese Verschiebung finde in der horizontalen und vertikalen Richtung statt. Diese

überlappende Verschiebung der Fenster wird in dieser Arbeit mit 50% gewählt. Durch die Überlappung werden mehr Fenster als durch das Aneinanderreihen benötigt. Die Anzahl der Fenster W_{max} errechnet sich aus der Gleichung:

$$W_{max} = 4 * (\frac{X}{W_{size}} - 1) * (\frac{Y}{W_{size}} - 1) \tag{3.6}$$

X und Y sind die Länge und die Breite des Kantenbildes.

Berechnung des Textbereiches im Bild

Nachdem das Kantenbild auf Bereiche mit hohen Kantendichten durchsucht wurde, müssen nun aus den gefundenen und somit markierten Fensterblöcken mit hoher Kantendichte rechteckige Bereiche, in denen Text enthalten sein kann, untersucht werden. Dazu müssen zwei Aspekte berücksichtigt werden. Erstens können die markierten Fenster für ein Wort zusammenhängend sein. Zweitens können zwei Wörter durch Fragmente im Bildhintergrund und somit auch durch markierte Fensterblöcke verbunden sein. Das Ziel ist, möglichst alle Wörter einzeln in einem Textbereich einzusortieren und auf diese Art wenige Bildfragmente als Textbereiche zu markieren.

Es werden nacheinander alle markierten Bereiche im Bild so lange in alle Richtungen um je eine Bildpunktzeile bzw. -spalte vergrößert, bis das Objekt mit den markierten Bereichen gänzlich erfasst ist. Ein Objekt kann über mehrere markierte Bereiche verfügen, so dass in diesem Fall das Objekt nach der Vergrößerung mehrfach vorliegt. Um diese unnötige Vervielfältigung zu vermeiden, werden alle markierten Bereiche, die bei der Vergrößerung erreicht werden, gelöscht. Es kann aber, wie später erläutert, auch ein schon erkanntes Objekt bei der Vergrößerung erreicht werden. An dieser Stelle wird die aktuelle Vergrößerung abgebrochen.

Neben den Kantenbildpunkten der Schriftzeichen befinde sich auch Kantenbildpunkte von Fragmenten aus dem Hintergrund in dem Bild. Diese Fragmente können unerwünscht die einzelnen Wörter miteinander verbinden. Diese Fragmente sind oft sehr schmale Linien. Es werden verschiedene Parameter (s.u.) eingeführt, die die Vergrößerung an diesen Punkten abbrechen. Die horizontale und vertikale Richtung der Vergrößerung wird unterschiedlich behandelt. Der Grund dafür sind die räumlichen Eigenschaften von Texten. Texte sind fast immer breiter als hoch. Diese Eigenschaft kann genutzt werden zum Ausschluss von Nichttextbereichen. In der horizontalen und vertikalen Richtung wird nur vergrößert, wenn in der nächsten Zeile bzw. Spalte die vorhandenen Kantenbildpunkte eine vordefiniert Prozentzahl oder minimale Kantenbildpunktzahl erreichen. Dafür werden als Parameter die prozentualen Schwellen P_{minx} und P_{miny} und die Schwellen für die minimalen Pixel L_{minx} und L_{miny} eingeführt (Abb. 3.20).

Zwischen den Schriftzeichen in einem Wort befinde sich Lücken. Außerdem gibt es Schriftzeichen, wie zum Beispiel das i, welche in der Vertikalen eine Trennung besitzen. Die so entstehenden Lücken müssen bei der Vergrößerung übersprungen werden. Dafür werden zwei weitere Parameter eingeführt. Es dürfen maximal bei der Vergrößerung in der Vertikalen G_{maxy} Zeilen und in der Horizontalen G_{maxx} Spalten übersprungen werden.

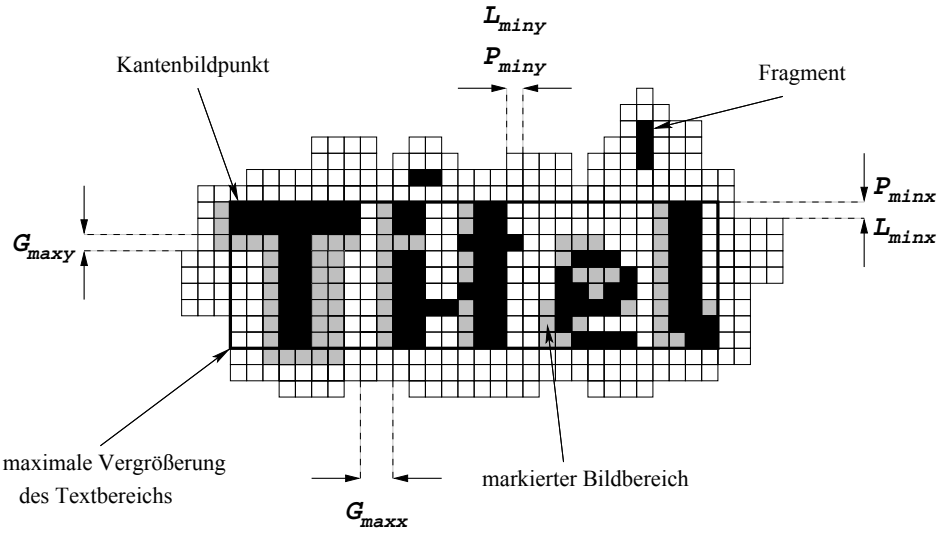
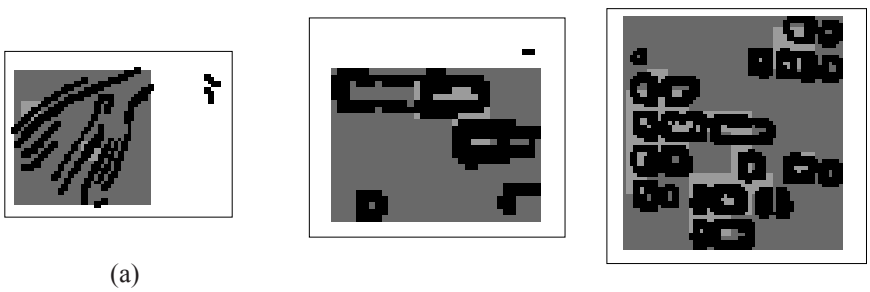


Abbildung 3.20: *In dem Bild ist das Wort Titel enthalten. Schwarze Bildpunkte sind Kanten, weiße Bildpunkte sind leer und graue Bildpunkte sind markierte Bereiche. Durch die Parameter P_{minx},P_{miny},L_{minx},L_{miny},G_{maxy} und G_{maxx} wird die Vergrößerung des Textbereiches (fetter Rahmen) begrenzt.*

3.2.5 Reduzierung der falsch erkannten Textbereiche

Es ist klar, dass nicht alle gefundenen Bildbereiche auch Text enthalten. In Bildern gibt es oft Strukturen die Schriftzeichen ähnlich sind. An dieser Stelle werden bestimmte Annahmen getroffen, um Bildbereiche ohne Text bzw. mit nicht erwünschtem Text zu löschen, damit diese in den weiteren Arbeitsschritten nicht unnötig weitergeführt werden.



(a) (b) (c)

Abbildung 3.21: *Bild a zeigt eine Bildstruktur, die als mögliche Textstelle markiert und durch das Verhältnis von Höhe zur Breite aussortiert werden konnte. Die Bilder b und c zeigen ebenfalls mögliche Textstellen, die durch die zu große Höhe ausgeschlossen werden können.*

Das wichtigste Kriterium ist hier das Verhältnis von Höhe zur Breite. Text ist immer breiter als hoch. Somit können gefundene Textbereiche, die höher als breit sind, sofort aus der Liste gelöscht werden (Abb. 3.21). Das Verhältnis Höhe zur Breite kann durch den Parameter T_{HB} eingestellt werden.

Für diese Arbeit ist es notwendig, nur bestimmte Textgrößen zu erkennen. Wörter können fast beliebig lang sein, haben aber immer eine ähnliche Höhe. Alle gefundenen Bildbereiche, die höher als der Schwellenwert T_{maxx} sind, können ebenfalls aus der Liste gelöscht werden.

Die letzte Kontrolle ist die Dichte der Kantenbildpunkte in dem vergrößerten Bildbereich. Sinkt die Dichte unter den Schwellenwert K_d kann ausgeschlossen werden, dass in diesem Bereich Text enthalten ist.

Einstellung der Kantenerkennung

Schriftzeichen besitzen meist klare Umrisse und sind von dem Hintergrund klar abzugrenzen. Außerdem besitzen sie gegenüber anderen Objekten im Bild viele kleine Strukturen und können dadurch in dem Bild durch mathematische Methoden aufgespürt werden. In der Kantenerkennung werden die Umrisse der Objekte eines Bildes gefunden. Der Algorithmus kann durch zwei Parameter $T1$ und $T2$ auf die gewünschten Resultate eingestellt werden. Die beiden Parameter geben die Schwellenwerte für die Entscheidung an, ob Löcher von Kanten der Objekte geschlossen werden sollen oder nicht, wenn keine Kante errechnet wurde. Schriftzeichen besitzen sehr viele Kanten und es kommt nur sehr selten dazu, dass aufgrund der kleinen Strukturen die oben beschriebenen Löcher vorkommen. Es ist wünschenswert, dass alle Objekte neben den Schriftzeichen möglichst wenig Kantenbildpunkte in dem zu errechnenden Kantenbild erhalten. Daher müssen diese beiden Parameter so eingestellt werden, dass sie möglichst wenig Verbindungen in den Kanten von Objekten herstellen 3.1.

Parameter	Symbol	Wert
untere Grenze für Kante	$T1$	254
obere Grenze für Kante	$T2$	255

Tabelle 3.1: *Die Parameter $T1$ und $T2$ sind so gewählt, das möglichst wenig Kanten von Objekten, die keine Schriftzeichen sind, gefunden werden.*

Es gibt in dem Graustufenbild für die Berechnung der Kanten 256 verschiedene Farben von 0 bis 255 gezählt und damit 256 verschiedene Ergebnisse für die Kantenstärke. Damit nur bei sehr hohen Werten ein Kantenbildpunkt erkannt wird, sind die Grenzen $T1$ mit 254 und $T2$ mit 255 zu wählen. Alle Kantenstärken von Farbbildpunkten eines Objektumrisses mit seiner Umgebung unterhalb dieser Grenze werden damit nicht als Kanntenbildpunkt erkannt.

Bestimmung der Parameter für Erkennung von Textstellen

Für das Auffinde des Textes in einem Kantenbild sind zuerst die Bereiche mit einer hohen Kantendichte zu finden Dafür werden mittels kleiner quadratischer Fenster die Bereiche des Kantenbildes abgesucht. Hier ist die Kantenlänge der quadratischen Fenster W_{size} und der Schwellenwert E_{max}, an dem ein Bereich als hohe Kantendichte markiert werden soll, zu bestimmen. Wichtig für die weiteren Module in dem System ist es, dass alle Schriftzeichen durch die Auffindun der hohen Kantendichten gefunden werden. Durch die Variation der

beiden Parameter und Überprüfung auf Vollständigkeit aller zu erkennenden Schriftzeichen wurden durch Versuche die Parameter empirisch bestimmt 3.2.

Für die Kantenlänge der quadratischen Fenster ist die Höhe bzw. Breite der zu erkennenden Schriftzeichen ausschlaggebend. Die Kantenlänge W_{size} der quadratischen Fenster wurde auf 12 Bildpunkte eingestellt. Das entspricht der durchschnittlichen Breite der Linien der Schriftzeichen.

Ausgehend von einer maximalen Dichte von 1 der Kantenbildpunkte in einem quadratischen Fenster wurde der Schwellenwert für die Markierung eines Fensters bestimmt. Der Schwellenwert wurde so weit verkleinert, dass alle Schriftzeichen durch ein Fenster markiert waren. Durch den Schwellenwert $E_{max} = \frac{(W_{size})^2}{2}$ wurde dieses Ziel erreicht.

Es ist klar, dass durch die gewählten Werte der Parameter nicht nur Schriftzeichen gefunden werden. Wichtig ist, dass die gesuchten Texte vollständig in die weiteren Teilschritte des gesamten Systems eingeordnet werden, so dass eine Vollständigkeit dieser garantiert werden kann. Die Fehler in Form von erkannten Fragmenten können durch die weiteren Teilschritte in dem System reduziert werden.

Parameter	Symbol	Wert
Kantenlänge vom quadratischen Fenster zum Auffinde der Kantendichten	W_{size}	12
Schwellenwert für das Auffinde der Kantendichten	E_{max}	$\frac{(W_{size})^2}{2}$
prozentualer Schwellenwert für die Vergrößerung in der X-Achse	P_{minx}	9
prozentualer Schwellenwert für die Vergrößerung in der Y-Achse	P_{miny}	10
minimaler Schwellenwert für die Vergrößerung in der X-Achse	L_{minx}	3
minimaler Schwellenwert für die Vergrößerung in der Y-Achse	L_{miny}	3
maximale Lücke für die Vergrößerung in der X-Achse	G_{maxx}	3
maximale Lücke für die Vergrößerung in der Y-Achse	G_{maxy}	0

Tabelle 3.2: *Die Parameter für die Detektion von Textbereichen wurden durch Annäherung gefunden.*

Nach der Markierung der Bereiche mit hoher Kantendichte sind sie in Bereiche mit enthaltenem Text zu erweitern. Dafür müssen weitere Parameter eingestellt werden, die dem entwickelten Algorithmus zugrunde liegen. Hier ist eine Vollständigkeit der Wörter zu gewährleisten und über die entstehenden Fehler zu stellen. Daher wurden die Parameter, ausgehend von minimalen bzw. maximalen Werten, so lange erweitert bzw. verkleinert, bis die Vollständigkeit empirisch gewährleistet ist. Ausgehend von einem Maximalwert wurden die Parameter P_{minx}, P_{miny}, L_{minx} und L_{miny} so lange verkleinert, bis alle Schriftzeichen der Wörter in der Vergrößerung der Bereiche enthalten waren. Die X-Achse und die Y-Achse

müssen getrennt von einander behandelt werden. Deshalb sind dafür je zwei Parameter notwendig. P_{minx} und P_{miny} begrenzen die minimale Prozentzahl der Kantenbildpunkte in einer zu erweiternden Bildpunktzeile. Die Parameter L_{minx} und L_{miny} stellen dabei die minimale Anzahl der Bildpunkte in der zu erweiternden Bildpunktzeile dar.

Die Parameter G_{maxx} und G_{maxy} bestimmen die maximale Lücke in Bildpunkten zwischen zwei Buchstaben bzw. die maximale Lücke innerhalb eines Schriftzeichens mit vertikaler Unterteilung. Die Bestimmung dieser Parameter ging von einem Minimalwert aus und wurde bis zur Vervollständigung der Wörter vergrößert.

Die Vollständigkeit der Wörter wurde durch die Einstellung der Parameter P_{minx} = 9%, P_{miny} = 10%, L_{minx} = 3, L_{miny} = 3, G_{maxx} = 3 und G_{maxy} = 0 erreicht.

3.2.6 Bildpyramide zur Erkennung unterschiedlicher Schriftgrößen

Die Detektion von Text ist bisher speziell auf eine Schriftzeichengröße optimiert. Die Größe umfasst dabei eine Schriftzeichenhöhe von 10 bis 100 Bildpunkten. Der Grund für diese Einschränkung ist der Anteil an Kanten bzw. Frequenzen pro fest definiert Fläche, welcher zur Detektion von Schriftzeichen wichtig ist. Werden die Schriftzeichen größer, sinkt die Anzahl an Kanten in einer fest definierte Fläche. Ab einer gewissen Größe kann kein Schriftzeichen mehr detektiert werden. Umgekehrt ist eine minimale Größe der Schriftzeichen ebenfalls entscheidend, da sonst ebenfalls die Anzahl an Kanten in einer fest definierte Fläche unzureichend ist. Prinzipiell muss für größere Schriftzeichen eine Erweiterung des bisherigen Verfahrens gefunden werden. Es muss Vollständigkeit alle Schriftzeichengrößen gefunden werden. An dieser Stelle gibt es zwei Alternativen zur Lösung dieses Problems. Zum einen kann für jede Schriftzeichengröße das Verfahren für die Detektion neu eingestellt werden. Hierbei entsteht das Problem, dass Bilder und damit Text beliebig groß sein können und damit unendlich viele Einstellungen nötig wären. Da dies nicht möglich ist, wäre das Verfahren immer noch auf eine maximale Schriftgröße limitiert.

Zum Anderen kann das Bild schrittweise durch Skalierung verkleinert werden und jedes Mal mit derselben Einstellung auf Schriftzeichen analysiert werden. Dieser Vorgang wird dann solange weitergeführt, bis das Bild nur noch ein Schriftzeichen enthalten kann. So entsteht in Abbildung 3.22 eine Pyramide von Bildern, die jeweils mit den selben Verfahren und Einstellungen analysiert wird. Eine erste Anwendung dieses Verfahrens wurde in [25] veröffentlicht. Die gefundenen Schriftzeichenbereiche können dann interpretiert bzw. erkannt werden.

Lanczos Filter

Die Skalierung eines Bildes ist ein wichtiger Aspekt zum Auffinde von Text in unterschiedlichen Größen. Die Berechnung der Bildpunkte im neuen skalierten Bild kann auf verschiedene Weise durchgeführt werden. Die einfachste Methode ist den örtlich nächst liegenden Bildpunkt aus dem Originalbild zu wählen und dessen Farbwerte zu kopieren. Diese Methode führt aber zu relativ schlechten Bildqualitäten. In der Veröffentlichung [26] wurden neben dieser Methode noch einige andere Verfahren verglichen. Als eines der besten Verfahren hat sich hier das Verfahren nach Lanczos [27] erwiesen. Für die in dieser Arbeit vorkommenden Skalierungen von Bildern wird ebenfalls das Verfahren nach Lanczos verwendet.

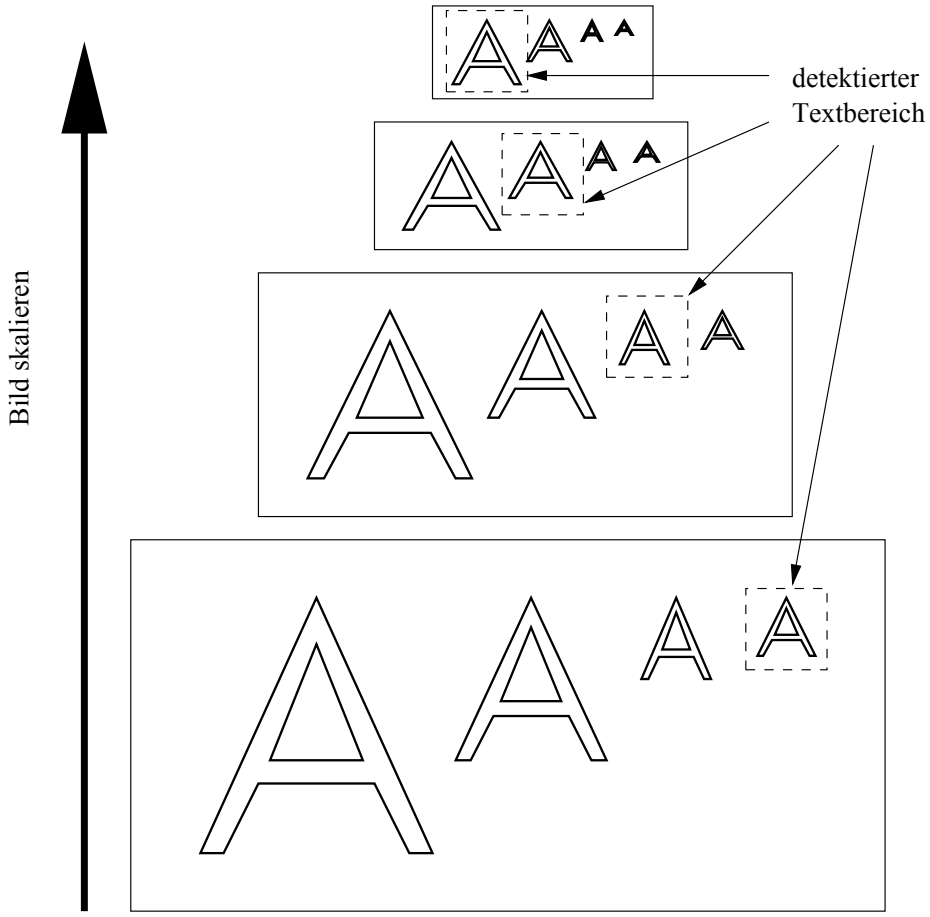


Abbildung 3.22: *In dem hier gezeigten Beispiel befinde sich vier verschiedene Größen von dem Schriftzeichen A. Das eingestellte Verfahren zur Detektion von Schriftzeichen kann nur Schriftzeichen in der Größe von dem gestrichelten Rechteck finden Für die Detektion muss das Bild schrittweise durch Skalierung verkleinert werden, so dass alle Schriftzeichen detektiert werden können.*

Die Grundlage des Verfahrens nach Lanczos basiert auf den Funktionen:

$$sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{3.7}$$

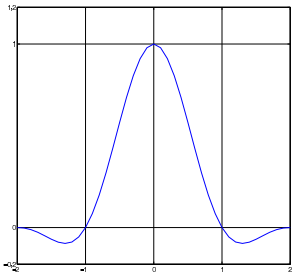


Abbildung 3.23:
Die Lanczos Funktion ist definier im Bereich -2 bis 2 und besitzt 4 Nullstellen.

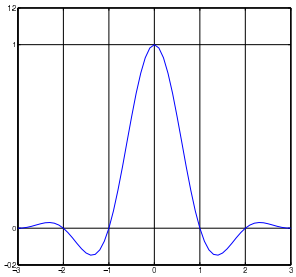


Abbildung 3.24:
Die Lanczos Funktion ist definier im Bereich -3 bis 3 und besitzt 6 Nullstellen.

Die Komplette Gleichung für den Lanczos Filter $L(x)$ lautet:

$$L(x) = \begin{cases} 1, & \text{falls } x = 0 \\ \frac{sin(\pi x) * sin(\frac{\pi x}{a})}{\pi^2 x^2}, & \text{falls } -a <= x < 0 \text{ und } 0 < x <= a \\ 0, & \text{sonst} \end{cases} \tag{3.8}$$

Hierbei beschreibt a die Anzahl der Nullstellen in positiver oder negativer Richtung der x-Achse. Üblicher Weise wird hier $a = 2$ oder $a = 3$ gewählt. Zur Begrenzung des Aufwandes der Berechnung werden alle Werte über a auf 0 gesetzt. An $x = 0$ entsteht eine Polstelle. In diesem Fall muss der Wert 1 für die Funktion definier werden. So ergibt sich für die Werte $a = 2$ in Abbildung 3.23 oder $a = 3$ in Abbildung 3.24 folgende Funktionsverlaufe.

Die Funktion $L(x)$ wird über jeden Bildpunkt im Ausgangsbild gelegt. Dabei ist der Abstand von zwei benachbarten Bildpunkten in der Vertikalen oder Horizontalen genau die Länge a. Wird ein neues Bild mit einem Skalierungsfaktor s berechnet, muss jeder neue Bildpunkt (x_{neu}, y_{neu}) in das Originalbild projiziert werden und so die virtuelle Position (x_{org}, y_{org}) berechnet werden. Hierfür gilt folgende Formel:

$$\begin{aligned} x_{org} &= \frac{x_{neu}}{s} \\ y_{org} &= \frac{y_{neu}}{s} \end{aligned} \tag{3.9}$$

Mit der so berechnete Position (x_{org}, y_{org}) kann dann die Distanz $dis_{x,y}$ zu jedem Bildpunkt im Originalbild (x, y) ermittelt werden:

$$dis_{x,y} = \sqrt{(|x_{org} - x|)^2 * (|y_{org} - y|)^2} \tag{3.10}$$

Werden die Distanzen x_{dis} von jedem Bildpunkt im Originalbild mit NxM Bildpunkten in die Funktion $L(x)$ eingesetzt, je mit dem Farbwert rot, grün und blau des Originalbildpunktes $(fr_{org}, fg_{org}, fb_{org})$ multipliziert, anschließend addiert und am Ende normiert,

ergibt sich der neue Bildpunkt (fr_{neu}, fg_{neu}, fb_{neu}):

$$
\begin{aligned}
fr_{neu} &= \sum_{x=1}^{N}\sum_{y=1}^{M} \frac{L(dis_{x,y}) * fr_{org}}{\sum_{x=1}^{N}\sum_{y=1}^{M} L(dis_{x,y})} \\
fg_{neu} &= \sum_{x=1}^{N}\sum_{y=1}^{M} \frac{L(dis_{x,y}) * fg_{org}}{\sum_{x=1}^{N}\sum_{y=1}^{M} L(dis_{x,y})} \\
fb_{neu} &= \sum_{x=1}^{N}\sum_{y=1}^{M} \frac{L(dis_{x,y}) * fb_{org}}{\sum_{x=1}^{N}\sum_{y=1}^{M} L(dis_{x,y})}
\end{aligned}
\tag{3.11}
$$

Da die Funktion $L(x)$ nur zwischen $-a >= 3$ und $a =< 3$ bzw. $-a >= 2$ und $a <= 2$ definier ist, sind die meisten Werte 0 und können bei der Berechnung des neuen Bildpunktes ignoriert werden, welches das Verfahren beschleunigt.

3.3 Die Verfolgung der Textblöcke

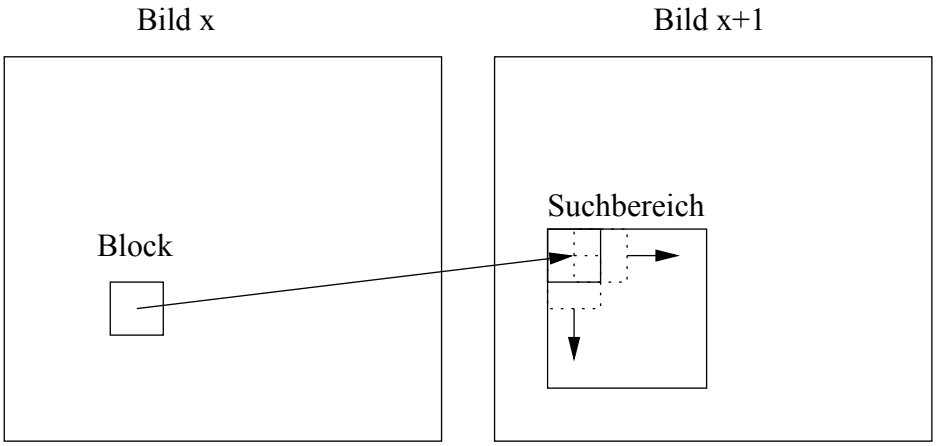


Abbildung 3.25: *Bei dem Block Matching Verfahren wird ein kleiner Block aus Bild x in dem Bild x+1 in einem Suchbereich mit den Bilddaten verglichen. Ist die Abweichung minimal ist der Block an diesen Koordinaten gefunden.*

Die Verfolgung von Textblöcken innerhalb mehrerer aufeinanderfolgender Bilder in einem Video kann auf verschiedene Weisen gelöst werden. Zum einen kann in jedem Bild erneut eine Detektion von Text durchgeführt werden. Nach der Detektion können die Resultate von den Einzelbildern verglichen werden. Sind die Koordinaten der von einem Bild

zum nächsten in gewissen Grenzen ähnlich, können diese miteinander verbunden werden. Die maximale Abweichung der Koordinaten muss vorher definier werden.

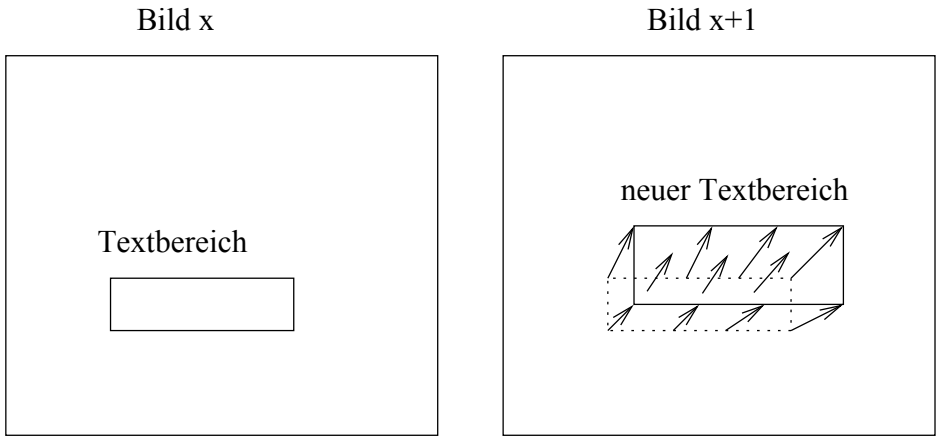


Abbildung 3.26: *In dem linken Bild ist ein Rechteck zu sehen, der Text enthalten kann. Im rechten Bild hat sich das Rechteck und damit der Text horizontal und vertikal bewegt. Außerdem ist das Rechteck größer geworden.*

Eine weitere Möglichkeit ist die Analyse des durch die Detektion von Text gefundenen Bildbereichs und das Suchen dieses Bereiches im nächsten Bild, die in Abbildung 3.25 dargestellt ist. Ein solches Verfahren wurde in [28] vorgestellt. Hierfür bietet sich ein Verfahren aus der Videokodierung an: Das Auffinde von Blöcken (Block Matching). In diesem Verfahren wird ein Block mit einer Aussagen definierte Größe, zum Beispiel mit 8 mal 8 Bildpunkten, in dem nächsten Bild gesucht. Verglichen wird dabei immer der zu suchende Block aus dem ersten Bild mit Bildbereichen um die Orginalposition herum im folgenden Bild. Es wird dabei eine maximale Abweichung der Position von einem Bild zum nächsten festgelegt, so dass nur ein Teil vom ganzen Bild im Video durchsucht werden muss. In diesem Bereich wird dann der Block hinein gelegt, verschoben und schrittweise mit den vorhandenen Bildwerten verglichen. Entscheidend ist hier die Abweichung der aufsummierten Abweichungen der einzelnen Werte der Bildpunkte. Die Position, die die geringste Abweichung aufweist, ergibt dann die neuen Koordinaten für den gesuchten Block. Auf diese Weise kann ein Bildbereich für einen Text in solche Blöcke zerlegt und im nächsten Bild im Video gesucht werden. Jeder Block ergibt dann einen Vektor, der angibt, wohin der Block im nächsten Bild gewandert ist. In Abbildung 3.26 ist das Ergebnis eines solchen Block Matching Verfahrens dargestellt. In diesem Beispiel hat sich der zu suchende Textbereich im nächsten Bild nach rechts und nach oben verschoben. Zusätzlich wurde der Textbereich vergrößert. Neben dem sehr einfachen hier vorgestellten Verfahren gibt es noch eine Reihe von komplexeren Verfahren zum Beispiel in [29], [30], [31] und [32]. In [33] wird auch die Deformation von Blöcken betrachtet.

3.4 Auswertung von automatischen Verfahren

Für die Auswertung der unterschiedlichen Verfahren muss eine einheitliche Analyse der Texte in einem Video gefunden werden. Damit alle Verfahren untereinander verglichen werden können, muss eine Referenz geschaffen werden. Ideal wäre hier eine fehlerlose Analyse jedes Videos in der festgehalten wird, wo, wann und welcher Text in einem Video zu finde ist. Es ist klar, dass durch jedes Verfahren Fehler entstehen werden. Daher muss das Verfahren mit der geringsten Fehlermenge als Referenz genutzt werden. Hierfür bietet sich die manuelle Beschriftung des Menschen an. Der Mensch dient für alle Verfahren als Vorbild und so ist die manuelle Beschriftung durch den Menschen als Grundwahrheit anzusehen. Die Analyse von Text in Videos muss einheitlich und auch durch den Computer verständlich sein. Entscheidend ist hierfür, in welchem Bild in dem Video Text ist, welche Koordinaten dieser hat und was es für Text ist. Weiterführend kann noch die Orientierung wichtig sein, welches in dieser Dokumentation nicht berücksichtig werden kann. Eine solche Beschreibung von Bildinhalten kann auch als Annotation bezeichnet werden.

3.4.1 Definition eines Bildbereiches

Als erstes muss klar definier werden, wie ein Textbereich aussehen muss. Im Gegensatz zu vielen anderen Objekten in einem Bild ist der Text nicht nur ein Objekt, sondern besteht aus vielen kleinen Objekten, den sogenannten Schriftzeichen. Ein Objekt, wie zum Beispiel in Abbildung 3.27 ein Ball oder eine Kiste, kann relativ einfach mit einer Kontur umrandet werden und so vom Rest des Bildes abgetrennt werden. Für einen Text ist das nicht ohne Probleme möglich.

Natürlich kann Text auch mit Hilfe der Konturen umrandet werden. Dabei entstehen aber einige Probleme. Zum einen sind die Flächen der Schriftzeichen relativ klein und damit kann der Hintergrund mit dem Schriftzeichen verschmelzen und eine klare Trennung ist nicht möglich. Des Weiteren können Schriftzeichen aus mehreren einzelnen Elementen bestehen wie zum Beispiel aus dem Schriftzeichen i. Auf diese Weise können einzelne Schriftzeichen geteilt werden. Außerdem bilden mehrere Schriftzeichen ein Wort und die Wörter dann einen Text. Also können hier verschiedene Ebenen der Markierung von Bildbereichen gewählt werden: die mit Schriftzeichen, die mit Wörtern oder die mit Texten. Für die Wahl der Ebene muss das weitere Vorgehen der Detektion und Erkennung von Text betrachtet werden. Für die Erkennung von Text müssen die Fehlerquellen möglichst weit reduziert werden. Daher muss der markierte Bereich im Bild möglichst klein um die Schriftzeichen sein. Dagegen sind für die meisten Erkennungsverfahren die benachbarten Schriftzeichen wichtig. Durch die in Abschnitt 2.6 beschriebenen Eigenschaften kann so die Orientierung des Textes aus der Größe zu den anderen Schriftzeichen, im Zweifelsfall zwischen Groß- und Kleinschreibung, unterschieden oder sogar anhand der Häufig eit der vorgestellten Schriftzeichen auf die weiteren Schriftzeichen geschlossen werden. Auch zwischen aufeinanderfolgenden Wörtern können bestimmte Eigenschaften genutzt werden. Es können aufgrund der Satzstruktur Rückschlüsse auf folgende Wörter geschlossen werden. Zum Beispiel kann durch die Kenntnis von Subjekt und Objekt erwartet werden, dass das dritte Wort ein Prädikat ist. So kann in einem Wörterbuch von Prädikaten nach bekannten Wörtern gesucht werden, die eventuell auf das dritte Wort passen und so eventuelle Fehler in der Erkennung vermieden werden. Ein solches Verfahren wurde in [34] veröffentlicht. Dieses Verfahren finde aber

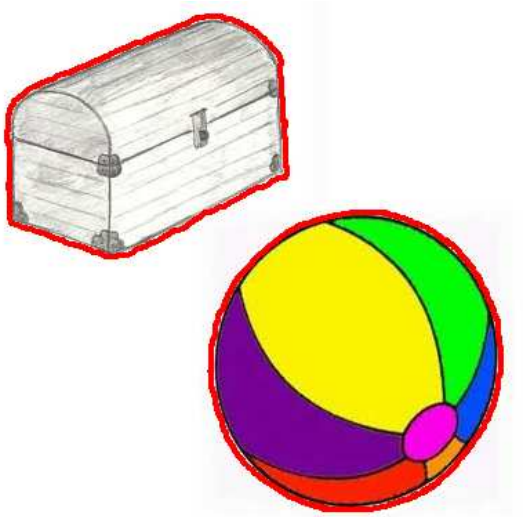


Abbildung 3.27: *In dem Bild sind zwei Objekte zu erkennen, eine Kiste und ein Ball. Beide Objekte sind mit einer roten Kontur markiert worden. Die Kontur ist durch ihre ausschließlich konvexe Wölbung relativ einfach.*

in dieser Arbeit keine Anwendung. Dagegen spielt der Zusammenhang von Schriftzeichen innerhalb eines Wortes zur späteren Korrektur von fehlerhaften Wörtern eine große Rolle. Deshalb wird in dieser Arbeit die Markierung von Bildbereichen mit einzelnen Wörtern verwendet.

Durch die Markierung von Bildbereichen mit Wörtern wird viel vom Bild mit markiert, welches keine Informationen für den Text beinhaltet. An dieser Stelle muss nun eine Methode gefunden werden, wie diese Markierung möglichst optimal aussehen muss. Ideal wäre ein frei verlaufender Umriss des Wortes, damit möglichst wenig von der Umgebung des Bildbereiches mit dem Wort erfasst wird. Leider kann diese Art der Markierung nur mit hohem mathematischen Aufwand oder einer großen Datenmenge beschrieben werden und ist daher für eine schnelle Auswertung nicht zu gebrauchen. Dagegen ist ein Kreis mathematisch sehr einfach zu beschreiben. Auch die Datenmenge wäre sehr gering. Es sind dabei lediglich ein Koordinatenpaar und ein Radius notwendig. Der Nachteil dieser Art der Markierung liegt in der großen Bildf äche, die keine wichtigen Informationen enthält. Außerdem können weitere Wörter in den Bereich von dem zu markierenden Wort herein ragen, welche dann zu Problemen in der Erkennung führen können. Text wird in den meisten Fällen in einer Zeile geschrieben. Nur in seltenen Fällen werden die Schriftzeichen eines Wortes auf einer Kurve dargestellt. Daher nimmt ein Wort unabhängig von der Länge in der Regel eine rechteckige Form an. Mit anderen Worten: Ein Wort spannt im Raum eine rechteckige Fläche auf. Da ein Rechteck ebenfalls mathematisch relativ einfach durch zwei Koordinatenpaare beschrieben werden kann, ist es für die Markierung von Textbereichen ideal. Auf diese Weise wird die Menge an nutzlosen Bildbereichen gering und der mathematische Aufwand klein gehalten. In Abbildung 3.28 sind die verschiedenen Varianten von Markierungen von Textberei-

Abbildung 3.28: *Als erste Möglichkeit einen Text zu markieren kann die Kontur von jedem Schriftzeichen genutzt werden. Die Kontur ist hier mit roter Farbe zu erkennen. Auf diese Weise ist der Text zwar perfekt umrahmt, aber die Schriftzeichen werden teilweise gespalten und eine Beschreibung der Markierung ist sehr aufwendig. Die zweite Möglichkeit, Text zu markieren, ist hier mit einer grünen Markierung dargestellt. Bei dieser Form wird der Textbereich grob umrandet. Hier sind die Schriftzeichen komplett vorhanden, aber die Beschreibung der Markierung ist ebenfalls sehr aufwendig. Durch einen Kreis bzw. eine Ellipse, hier in türkis, ist eine Beschreibung der Markierung deutlich einfacher. Da aber viel von der Umgebung mit markiert wird, welche aber keine Textinformationen enthält, ist das Rechteck besser geeignet für die Markierung von Text. In diesem Beispiel ist der Text 'Beispieltext' durch einen blauen rechteckigen Rahmen sehr gut markiert worden ohne zu viele unnötige Bereiche mit zu integrieren und gleichzeitig ist der Text vollständig markiert.*

chen dargestellt. Durch viele Faktoren wie Unschärfe, verlustbehaftete Videokomprimierung oder Objekte, die Wörter verdecken können, ist oft eine perfekte Markierung eines Wortes nicht möglich. Der Übergang von Bildpunkten, die zum Hintergrund oder dem Schriftzeichen gehören, ist fließend Da in der Detektion des Textes noch keine Aussage zu der Erkennung der Schriftzeichen und damit des Wortes stattfindet kann also auch darüber keine Aussage getroffen werden, welche Bildpunkte wichtig und welche unwichtig sind. Daher müssen in diesem Fall großzügig Bildpunkte in den markierten Bereich mit aufgenommen werden. Diese Tatsache widerspricht aber dem Ziel, möglichst wenig unwichtigen Bereich mit zu markieren. Daher muss hier ein Kompromiss gefunden werden. Als Ideal hat sich hier eine empirisch gefundene Abweichung von 30% der Höhe des Textes in alle Richtungen des Wortes gezeigt. Durch diese Abweichung wird in einem kompakten Text mit mehreren Zeilen keine Teile von Wörtern anderer Zeilen mit markiert und die Trennung der Wörter durch ein Leerzeichen berücksichtigt.

Abbildung 3.29: *In dem Bild sind mehrere Textbereiche durch einen blauen rechteckigen Rahmen markiert worden.*

3.4.2 Die manuelle Annotation von Text

Das Ziel ist eine Aussage über die Trefferrate von der Detektion von Textbereichen mittels der hier vorgestellten automatischen Verfahren. Damit die automatischen Verfahren ausgewertet werden können, muss zu den durch die Verfahren automatisch markierten Bereichen ein Vergleich manuell erstellt werden. Hier entstehen mehrere große Probleme. Als erstes spielt die Auswahl der Videos für die Ergebnisse eine große Rolle. Videos können sehr unterschiedliche Eigenschaften aufweisen. In einigen ist extrem viel Text vorhanden und in einigen nur sehr wenig bis gar kein Text. Ebenfalls ist die Qualität und Auf ösung entscheidend. Zusätzlich ist die Art des Vorkommens des Textes von Bedeutung. Diese Faktoren beeinflus sen das Ergebnis der automatischen Detektion von Text und werden in Abschnitt 3.5 näher betrachtet. Ebenfalls können manuell markierte Bereiche von den automatisch markierten Bereichen abweichen. Diese Abweichung muss aber im Rahmen bleiben und ein Grenzwert muss gefunden werden, der in Abschnitt 3.4.6 beschrieben wird. Das größte Problem ist die Erstellung der manuellen Markierungen. Ein Video kann pro Sekunde 24 Bilder besitzen. Das macht für eine halbe Stunde Videomaterial 43.200 einzelne Bilder, die einzeln manuell auf Textbereichen untersucht werden müssen. In jedem Bild können sehr viele Bildbereiche mit Text vorkommen, die alle einzeln markiert werden müssen. Wenn im Schnitt jedes Bild 10 Bildbereiche mit Text besitzt, dann ergeben sich insgesamt 432.000 markierte Bereiche. Da jeder Bildbereich durch zwei Koordinatenpaare angegeben werden muss, sind es am Ende 864.000 Koordinatenpaare bzw. 1.728.000 Koordinaten, die für dieses Video manuell gespeichert werden müssen. Zusätzlich muss dann der Text manuell eingegeben werden. Auf herkömmliche Weise müsste ein Mensch jedes einzelne Bild betrachten, mit einem Cursor die entsprechenden Koordinaten erfassen und in eine Datei mit dem entsprechenden Text schreiben. Dieses Vorgehen ist extrem zeitaufwendig und so ist die Menge an vergleichbaren Daten,

die manuell erstellt wurden, stark limitiert. Zusätzlich birgt diese Methode ein großes Fehlerpotential durch Tippfehler, die später den Vergleich von automatischen Verfahren verfälschen können. Damit dennoch eine möglichst große Zahl an vergleichbaren Daten erzeugt werden kann muss ein Weg gefunden werden, diese schneller und sicherer zu erstellen. Hierfür wird ein Annotation Tool benötigt, welches in Abschnitt 3.4.3 beschrieben wird.

3.4.3 Annotation Tool

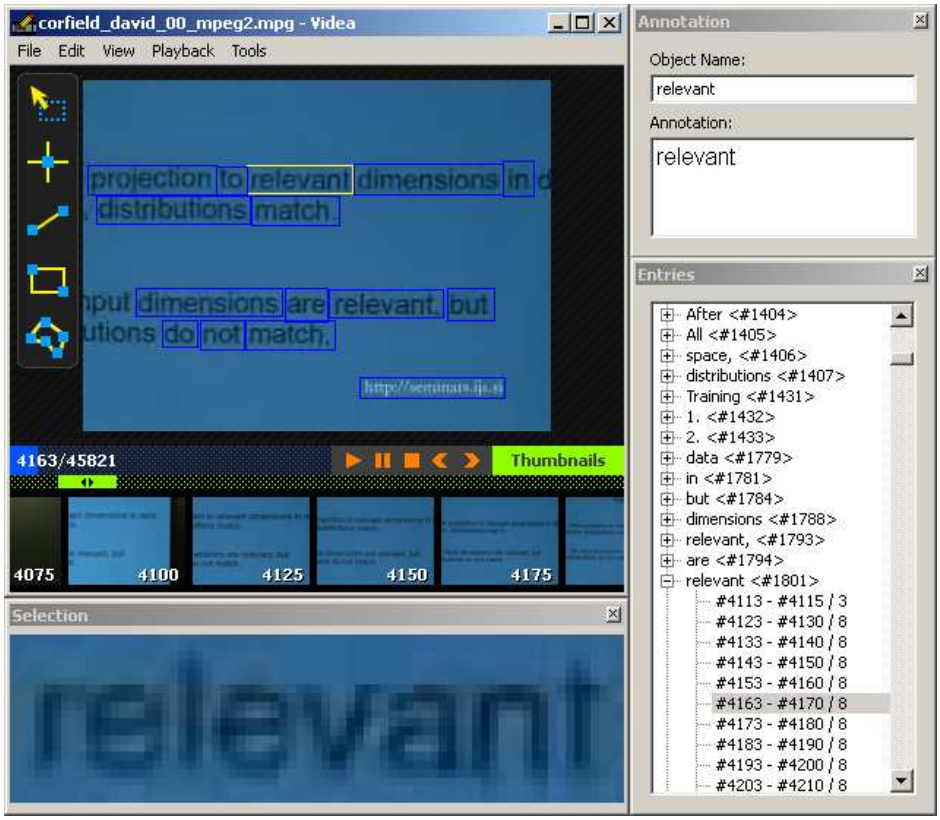


Abbildung 3.30: *In der Abbildung ist die GUI des Annotation Tools zu erkennen, in der gerade ein Video geladen ist und schon vorhandene Annotationen abgespeichert wurden.*

Auf herkömmliche Weise muss ein Video mit einem Programm zum Abspielen von Videos, in dem jedes zeitlich aufeinander folgende Bild einzeln dargestellt werden kann, geöffnet werden. Ebenfalls muss in diesem Programm das Ausmessen von Koordinaten innerhalb der einzelnen Bilder möglich sein. Erst durch diese Funktionen ist ein Erstellen von Daten manuell möglich. Der Mensch muss zum Erstellen eines Datensatzes jedes Bild einzeln betrachten. Auf jedem Bild müssen zu jedem einzelnen Text die Koordinaten der linken obe-

ren Ecke und die Koordinaten der rechten unteren Ecke des zu markierenden Textes gesucht werden. Als Eintrag in die Liste der Texte vom Video muss dann ein Datensatz, bestehend aus der Bildnummer im Video, den zwei Koordinatenpaaren oben rechts und unten links und dem Text geschrieben werden. Hierfür wird sehr viel Zeit benötigt. Für die manuelle Annotation von Text sind also Hilfsmittel nötig, damit eine große Datenmenge zum Vergleichen der Ergebnisse von automatischen Verfahren in angemessener Zeit erzeugt werden kann. Um die Eingabe von großen Datenmengen zu erleichtern sind Hilfsmittel wichtig, die in einem Annotation Tool vereint werden müssen. Als Grundlage dient hier ein Programm, welches wie bei der herkömmlichen Methode das Video laden, die einzelnen Bilder anzeigen und auch Koordinaten in den Bilder zur Verfügung stellen kann. Eine wichtige Zeitersparnis wird erreicht, wenn die durch den Menschen ausgemessenen Koordinaten eines Textbereiches durch Anklicken direkt gespeichert werden können und nicht erst durch den Menschen gelesen und dann in ein separate Datei gespeichert werden müssen. Auf diese Weise muss der Mensch nur 2 Punkte pro Text anklicken und braucht die Koordinaten nicht erst ablesen und wieder eingeben. Oft kommt es in der herkömmlichen Methode zu Fehlern durch Tippfehler oder mangelnde Konzentration. Diese Fehler können in dem Annotation Tool behoben werden, indem der markierte Bereich direkt auf dem Bild des Videos dargestellt wird. So können falsche Eingaben sofort erkannt und korrigiert werden. Nachdem ein Textbereich durch zwei Punkte markiert wurde, muss der entsprechende erkannte Text von diesem Bereich eingegeben werden. Ebenfalls ist bei der herkömmlichen Methode oft das zu ungenaue Ausmessen der Koordinaten ein Problem. Grund dafür ist die zu feine Auf ösung des Bildes. In einem Annotation Tool ist ein Zoom für den Bildbereich daher wichtig. Im Weiteren kann weiter viel Zeit eingespart werden. Oft kommt es vor, dass in zwei oder mehr zeitlich aufeinanderfolgenden Bildern in einem Video keine oder kaum eine Änderung stattfindet In diesem Fall können in dem Annotation Tool einfach die Markierungen der letzten Seite kopiert werden. Durch eine Kamerabewegung kommt es oft vor, dass sich zwar der Text und dessen relative Position zueinander nicht geändert haben, aber die globale Position verschoben wurde. In diesem Fall kann durch Kopieren und Verschieben der Auswahl aller oder bestimmter Markierungen in dem zeitlich vorhergehenden Bild ebenfalls ein Bild sofort analysiert werden.

3.4.4 Umsetzung und Funktionalität des Annotation Tools

Vorrangig für ein Annotation Tool ist die einfache und vor allem schnelle Bedienung. Daher müssen die sich immer wiederholenden Vorgänge so konzipiert werden, dass sie leicht zugänglich und möglichst schnell durchführbar sind. Eine grafisch Nutzeroberf äche (GUI - engl. Graphical User Interface) ist daher unumgänglich. Die GUI ist in Abbildung 3.30 dargestellt. Die wesentlichen Funktionen für die Annotation eines Videos müssen mit möglichst wenigen Schritten oder sogar direkt durch eine Taste oder Tastenkombination erreichbar sein. Für die Umsetzung der GUI dient die Bibliothek QT 4.2.0 [35]. Die Basis des Annotation Tools ist C++. Die Bibliothek zum öffnen des Videos ist FFmpeg [36]. Als Entwicklungsumgebung dient Visual Studio 2005 [37].

In Abbildung 3.31 ist die GUI des Annotationen Tools abgebildet, in dem ein Video mit vorhandenen Annotationen geladen ist. Der Name “corfiel _david_00_mpeg2.mpg“ der Video-Datei ist oben in der Kopfzeile zu erkennen, gefolgt von einer Zeile mit den wichtigen Funktionen innerhalb von Menüs. Unter dem Menü ist ein Bild aus dem Video zu erkennen. Das Bild entspricht der realen Auf ösung von dem Video. Auf dem Bild sind bereits

Abbildung 3.31: *Das Hauptfenster zeigt das Video mit den bereits existierenden Annotationen und den wichtigsten Werkzeugen zum Erstellen und Bearbeiten weiterer Annotationen.*

Annotationen in Form von blauen Rechtecken zu erkennen. Links von dem Bild aus dem Video befinde sich die Hilfsmittel für die Annotierung von Bildbereichen. Durch die schnelle Auswahlmöglichkeit bestimmter Funktionalitäten wie dem Zusammenfassen oder dem Setzen von Annotierungen ist eine schnelle Bearbeitung eines Bildes möglich. Das Annotation Tool bietet neben der Möglichkeit rechteckige Markierungen zu setzen auch weitere Markierungsvarianten. Durch die Schnellauswahl kann ein Punkt, eine Linie oder sogar ein Polygon verwendet werden. Unterhalb des Videobildes befinde sich die Funktionalitäten für die Navigation in dem Video. Wichtig ist hier das schrittweise Bild für Bild Vor- oder Zurücklaufen in dem Video. Während der Bearbeitung wird unter den Navigationswerkzeugen für jedes 25te Bild in dem Video eine verkleinerte Kopie des Bildes in einer Liste in chronologischer Reihenfolge eingetragen. Diese Liste kann genauso wie das Video an sich schnell durchspielt werden. So bekommt der Betrachter innerhalb kurzer Zeit einen Eindruck von dem Video oh-

ne sich den kompletten Inhalt anschauen zu müssen und kann direkt zu den wichtigen Stellen durch Anklicken eines verkleinerten Bildes gelangen. Auf dem Bild von dem Video sind 15 Annotationen vorhanden. Jede Annotation ist in einem blauen Rahmen dargestellt. Zur weiteren Bearbeitung kann ein so markierter Bildbereich erneut ausgewählt werden. Dieser wird dann in einer gelben Farbe dargestellt. In diesem Fall wurde ein Bereich markiert, in dem das Wort "relevant" enthalten ist.

Abbildung 3.32: *Jede Annotation eines Bildbereichs kann in einem extra Fenster vergrößert werden, damit eine genauere Bearbeitung der Koordinaten möglich wird.*

Für eine genauere Bearbeitung einer Annotation dient das Vergrößern eines Bildbereiches. In Abbildung 3.32 ist die Vergrößerung von dem ausgewählten Bildbereich des Wortes "relevant" zu erkennen. Erst in dieser Vergrößerung sind die einzelnen Bildpunkte zu erkennen und der Rahmen für die Annotation kann genauer festgelegt werden. Aus diesem Beispiel wird auch ersichtlich, dass die Grenzen einer Annotation variieren können. Es ist nicht genau ersichtlich wo ein Schriftzeichen anfängt oder endet. Die Übergänge sind hier fließend Sind die Koordinaten einer Annotation gefunden, kann der Inhalt beschrieben werden. Hierfür dient ein kleines Fenster, welches in Abbildung 3.33 dargestellt ist. In diesem Fenster sind zwei Felder editierbar. Einerseits kann der Annotation ein Objektname gegeben werden und andererseits kann der Inhalt beschrieben werden. In diesem Fall sind der Objektname und die Beschreibung des Inhaltes identisch.

Der Grund für die Unterteilung in den Objektnamen und die Objektbeschreibung liegt in der Beschaffenheit der Wörter in einem Video. Ein Wort kann in einem Video über viele Bilder vorkommen, ist aber immer das gleiche Wort. Für das Speichern von Annotationen ist der Zusammenhang zwischen Wörtern verschiedener Bilder wichtig. Auf diese Weise kann eine Annotation über viele Bilder geführt werden und trägt dabei immer denselben Objektnamen. Die Objektbeschreibung kann dagegen in jedem Bild der Annotation verschieden sein. Der Grund hierfür liegt in der Beschaffenheit des Videos. Durch Kamerabewegungen oder sich bewegende Objekte in der Szene können Teile eines Wortes für kurze Zeit aus dem Bild verschwinden bzw. verdeckt werden. Solche Situationen müssen in der Beschreibung der Annotation berücksichtigt werden. Da es sich trotz fehlender Wortteile immer noch um dasselbe Objekt handelt, werden diese unterschiedlichen Objektbeschreibungen zusammen gesetzt und mit dem Objektnamen gespeichert. Ist eine Annotation vollständig, wird sie in der Liste gespeichert. In Abbildung 3.34 ist das Fenster mit der Liste aller bereits existieren-

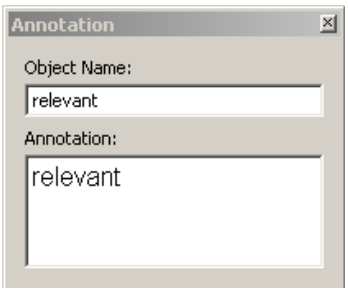


Abbildung 3.33: *Eine Annotation besteht aus dem Objektnamen, welcher über alle Bilder identisch ist und einer Objektbeschreibung, die sich je Bild unterscheiden kann.*

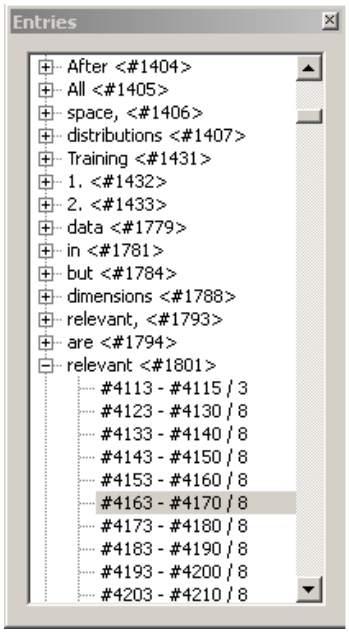


Abbildung 3.34: *Alle Annotationen sind in einer Liste gespeichert.*

den Annotationen ersichtlich. Jede Annotation hat eine Kennzahl (ID), die in den dreieckigen Klammern steht. Für weitere Informationen kann der Eintrag für die Annotation aufgeklappt werden. Auf diese Weise können die einzelnen Beschreibungen für die Bilder eingesehen werden. In jeder Beschreibung ist die Start- und Endbildnummer, gefolgt von der Anzahl der daraus resultierenden Bilder, dargestellt. In der Abbildung 3.34 ist ebenfalls die Annotation von dem Wort „relevant“ aufgeklappt. Durch die Auswahl eines Eintrags in der Liste kann direkt zu dem entsprechenden Bild in dem Video gesprungen werden. Ebenfalls kann die

Annotation nachträglich bearbeitet werden.

3.4.5 Speicherung der Daten

Für die Auswertung der manuellen und automatischen annotierten Daten ist eine einheitliche Datenstruktur notwendig. Für die Speicherung wird das Extensible Markup Language (XML) Format [38] benutzt. In dieser Sprache werden Elemente immer durch ein beschreibendes Wort wie „annotation“ umgeben mit den Klammern “⟨⟩“ eingeführt. Jedes Element muss speziell abgeschlossen werden. Das kann innerhalb der gleichen Zeile mit der Einführung durch ein „/“ am Ende vor dem „⟩“ geschehen oder in einer weiteren Zeile durch die Kombination des beschreibenden Wortes und dem „/“ am Ende. In jedem Element können weitere Spezifikatione aufgelistet werden. Ein Element hat damit eine der folgenden Strukturen:

```
<annotation>
  ...
</annotation>

<node ... />
```

Zwischen dem Anfang und dem Ende innerhalb eines Elementes können weitere Elemente definier werden. Diese sind hier durch ... symbolisiert. In dem XML Format sind ebenfalls Variablen möglich. Variablen werden durch ein = gefolgt von einer Zahl in „“ beschrieben. Mit Hilfe dieser Sprachelemente kann in dem XML Format eine Annotationsliste vollständig beschrieben werden. Eine so erstellte Datei kann wie folgt aussehen:

```
<annotation>
  <entry type="rectangle" id="1801" name="relevant">
    <reference end="4115" start="4113" />
    <shape>
      <text>relevant</text>
      <origin x="67" y="51"/>
      <nodes>
        <node x="-44" y="-12"/>
        <node x="43" y="12"/>
      </nodes>
    </shape>
  </entry>
  .
  .
  .
</annotation>
```

Mit dem Element „annotation“ wird die gesamte Annotationsliste beschrieben. In dieser Liste befinde sich alle Annotationen, die immer mit dem Elementtyp „entry“ definier werden. Eine Annotation besitzt ebenfalls weitere Elemente, die alle notwendigen Eigenschaften beschreiben. Die ersten Beschreibungen befinde sich schon in der Einführungzeile für die Annotation. Sie beschrieben mit „type“ die Art, mit „id“ die Kennzahl und mit „name“ die

verbale Beschreibung der Annotation. In diesem Fall ist die Art der Annotation ein Rechteck, die Kennzahl die 1801 und die verbale Beschreibung „relevant“. Innerhalb einer Annotation werden zwei weitere Elemente definiert Zum einen „reference“ zur Beschreibung des Zeitpunktes in dem Video und zum zweiten „shape“ zur örtlichen und inhaltlichen Beschreibung innerhalb des Bildes. In diesem Fall beginnt die Annotation in Bild 4113 und endet in Bild 4115. Die örtliche und inhaltliche Beschreibung wird durch 3 Elemente „text“, „origin“ und „nodes“ für den zu erkennenden Text, den Mittelpunkt und die Ausdehnung der Annotation definiert In diesem Fall ist der zu erkennende Text „relevant“. Der Mittelpunkt befinde sich bei den Koordinaten $x = 67$ und $y = 51$. Für die Ausdehnung werden die obere linke und die untere rechte Ecke des Rechtecks gewählt. Im Gegensatz zum Mittelpunkt werden für die Ausdehnung keine Koordinaten angegeben, sondern Abstände auf der x- und y-Achse. Für die Beschreibung eines Rechtecks sind minimal 2 Koordinatenpaare notwendig. Die Angabe des Mittelpunktes ist für das Annotation Tool wichtig.

3.4.6 Abweichung von manueller und automatisch markierten Bildbereichen

Es ist klar, dass die manuelle Markierung von Bereichen von der automatischen Markierung abweichen kann, obwohl derselbe Bereich gemeint ist. Diese Abweichung kommt durch die eher subjektive Einschätzung des Menschen bei der manuellen Eingrenzung der Bereiche mit Text. Werden die Koordinaten der manuellen und automatischen Markierung direkt vergleichen, wird kaum eine Übereinstimmung zu finde sein. Daher ist eine automatische Auswertung schwierig. Wichtig ist dabei die gemeinsame Schnittmenge der manuellen und automatisch erstellten Markierung desselben Textbereiches. Dazu müssen die Flächen der manuellen und automatischen Markierungen verglichen werden. In Abbildung 3.35 ist ein Beispiel für die unterschiedlichen Bereiche desselben Textes durch automatische und manuelle Markierungen ersichtlich.

Im weiteren Verlauf wird die Fläche des manuelle markierte Bereich als F_m und die Fläche des automatisch markierte Bereich als F_a bezeichnet. Der Flächeninhalt kann für die Fläche F_a durch die Koordinaten $((x1_a, y1_a), (x2_a, y2_a))$ und für die Fläche F_m durch die Koordinaten $((x1_m, y1_m), (x2_m, y2_m))$ berechnet werden:

$$F_a = (x2_a - x1_a) * (y2_a - y1_a) \tag{3.12}$$
$$F_m = (x2_m - x1_m) * (y2_m - y1_m) \tag{3.13}$$

Für den Vergleich beider Bereiche muss die gemeinsame Fläche gefunden werden. Die gemeinsame Fläche kann ebenfalls durch Koordinaten beschrieben werden, die durch folgende Formeln gefunden werden:

$$x1_s = Max(x1_a, x1_m) \tag{3.14}$$
$$x2_s = Min(x2_a, x2_m) \tag{3.15}$$
$$y1_s = Max(y1_a, y1_m) \tag{3.16}$$
$$y2_s = Min(y2_a, y2_m) \tag{3.17}$$

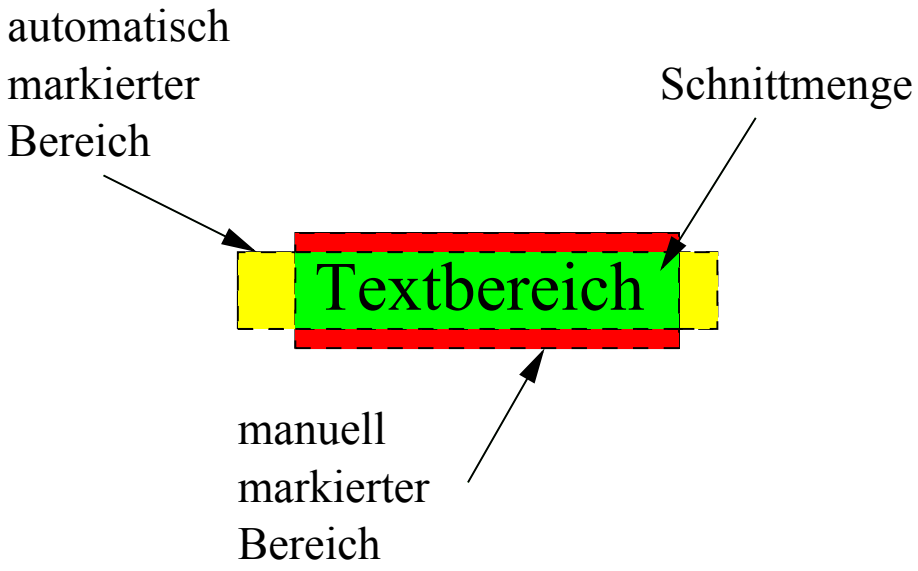


Abbildung 3.35: *In der Abbildung sind die Rechtecke der manuellen und automatischen Markierung erkennbar. Durch die verschiedenen Farben sind die Schnittmenge und die getrennten Bereiche hervorgehoben. Der grüne Bereich ist dabei die gemeinsame Schnittmenge beider Markierungen. Rot und Gelb dagegen sind die Bereiche ohne Übereinstimmung.*

Es wird angenommen, dass $x1_a < x2_a$, $y1_a < y2_a$,$x1_m < x2_m$ und $y1_m < y2_m$ ist. Die Funktion Max bestimmt hierbei das Maximum von zwei Zahlen und die Funktion Min das Minimum von zwei Zahlen. Die gemeinsame Fläche F_s ergibt sich aus der Formel:

$$F_s = Max(0, x2_s - x1_s)) * Max(0, y2_s - y1_s)) \tag{3.18}$$

Ist die Koordinate $x1_s > x2_s$ oder $y1_s > y2_s$, dann ist die gemeinsame Fläche 0. Aus der Flächen F_a, F_m und F_s lässt sich nun berechnen, wie viel Prozent des manuell markierten Bereiches im automatisch markierten Bereich liegt und umgekehrt:

$$P_{am} = F_s/F_a * 100 \tag{3.19}$$
$$P_{ma} = F_s/F_m * 100 \tag{3.20}$$

P_{am} beschreibt hier den prozentualen Anteil der Fläche F_s in der Fläche F_a und P_{ma} den prozentuellen Anteil der Fläche F_s in der Fläche F_m. Es reicht an dieser Stelle nicht, wenn einer dieser Werte maximal wird, da es vorkommen kann, dass ein markierter Bereich nur ein kleiner Teil in dem anderen markierten Bereich ist und damit nur ein Bruchteil des möglichen Bereiches gefunden wurde. Erst durch den Durchschnitt des prozentualen Anteils P_g von P_{am} und P_{ma} ist eine Aussage über die Analogie vom manuell markierten Bereich

zum automatisch markierten Bereich möglich. Die Größe des prozentualen Anteils P_g ist je nach Beschaffenheit des Videos und des Textes in dem Video sehr unterschiedlich. Hier spielt die Auf ösung des Videos und die Größe des Textes eine große Rolle. Daher kann P_g nur empirisch festgelegt werden.

3.5 Auswahl des Videomaterials für ausführliche Experimente

Aufgrund der zeitaufwendigen manuellen Markierung von Textbereichen in Videos muss im Vorfeld eine gute Auslese von Testvideos stattfinden Die Videos müssen bestimmte Kriterien erfüllen. Zum einen müssen sie viel Text enthalten, an dem man eine Aussage über die Trefferrate der automatischen Detektion von Text fällen kann. Andererseits müssen sie auch eine Herausforderung für die automatische Detektion sein. Einige Kriterien können dazu aufgelistet werden. Sie sind aber oft nur subjektiv zu bewerten.

- Auf ösung des Videos
- Framerate
- Bitrate bzw. Komprimierung des Videos
- Kontrast der Bilder
- Text als Einblendung oder in der Szene
- Häufig eit von Text
- Häufig eit von anderen Objekten neben Text
- Art der Wörter

Die Auf ösung des Videos hat direkten Einflus auf die Erkennungsrate. Je kleiner die Bilder sind, je weniger Bilddaten können für die Detektion und Erkennung genutzt werden. Da viele Videos durch Einsparung der Datenmenge sehr niedrige Auf ösungen besitzen, sollten hier Testvideos mit ebenfalls geringer Auf ösung genutzt werden. Die Framerate gibt an, wieviel Bilder pro Sekunde in dem Video enthalten sind. Umso mehr Bilder zur Verfügung stehen, umso mehr Bilder können für eine Detektion und Erkennung genutzt werden, was dann wieder zu einer höheren Detektions- und Erkennungsrate führt. Ebenfalls entscheidend ist die Bitrate. Die meisten Videos sind verlustbehaftet komprimiert. Je niedriger die Bitrate, also die Menge an Bits pro Zeiteinheit ist, je schlechter wird das Bild bzw. Video. Einer der Effekte ist eine blockartige Struktur in den Bildern und das Verschwimmen von Konturen, die für die Detektion und Erkennung von Text wichtig sind. Das Testvideo sollte daher möglichst nahe an der Grenze zum nicht Erkennbaren liegen. Der Kontrast ist ebenfalls wichtig für die Detektion und Erkennung. Sie ist stark von der Aufnahme abhängig. Je niedriger der Kontrast ist, um so weniger kann erkannt werden. Text als Einblendung ist einfacher zu erkennen als Text in der Szene, da dieser statisch (abgesehen von einem Laufband) und immer im Vordergrund ist. Text in der Szene kann wie in Abschnitt 2.5 von Objekten verdeckt werden, durch Lichteinf üsse unterschiedlichen Kontrast besitzen oder durch einen Kameraschwenk

sich verschieben. Um die Grenze in dieser Arbeit vorkommenden Algorithmen zu ermitteln ist Text in der Szene wichtig. Eine Komplexe Fragestellung ist die Mischung zwischen Text in dem Video und anderen Objekten. Wenn nur Text vorhanden ist, kann auch nur Text erkannt werden. Ist aber kein Text enthalten, dafür aber viele ähnliche Objekte, dann steigt automatisch die Zahl der falsch detektierten Bereiche. Da in dieser Arbeit der Fokus auf der nachträglichen Bearbeitung der erkannten Texte liegt, ist also an dieser Stelle ein Video zu wählen, welches viele Texte enthält. Dabei sind andere Objekte nebensächlich, da sie kaum eine Rolle spielen bei der Aussage der Qualitätsverbesserung durch eine nachträgliche Bearbeitung der Ergebnisse. In der Qualitätsverbesserung spielt die Art der Wörter eine große Rolle. Sind nur einfache Wörter vorhanden, kann die Qualität sehr leicht mit dem Abgleich durch ein Wörterbuch verbessert werden. Handelt es sich aber um komplizierte und seltene Wörter, ist die Wahrscheinlichkeit für Fehlschlagen des Abgleichens mit einem Wörterbuch deutlich höher. Besonders schwierig wird es bei Kunstworten oder Namen, da diese eine beliebige Varianz besitzen können und damit nicht in einem Wörterbuch enthalten sein können. Da oft Namen bei einer Suche sehr wichtig sind, sollten diese in der Arbeit mit berücksichtigt werden.

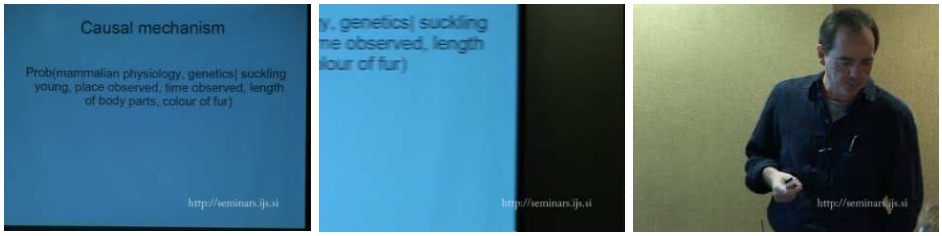


Abbildung 3.36: *In den drei Bildern sind verschiedene Momente aus dem Video für die Auswertung der Arbeit aufgezeigt. Im linken Bild ist eine Leinwand zu erkennen, auf der Text projiziert wird. Im mittleren Bild schwenkt die Kamera von der Leinwand zum Sprecher von links nach rechts. Durch die Bewegung kommt es zu Unschärfe im Bild und der Text verschwimmt. Außerdem ändert die unterschiedliche Beleuchtung von Leinwand und Sprecher den Kontrast. Auf dem rechten Bild ist der Sprecher zu erkennen.*

Für diese Arbeit wurde daher ein Video gewählt, welches möglichst gut zu den erwähnten Kriterien passt. In dem Video wurde eine wissenschaftliche Präsentation gefilm und die Kamera zeigt abwechselnd mit einem Kameraschwenk den Sprecher oder die Leinwand. Das Video ist in der Videoplattform videolectures.net [39] zu finden In Abbildung 3.36 sind drei verschiedene charakteristische Bilder aus dem Video zu erkennen. Insgesamt hat das Videos 45.821 einzelne Bilder. Das entspricht bei 25 Bildern pro Sekunde einer Videolänge von 30 Minuten und 33 Sekunden. Das Video hat eine Auf ösung von 320 mal 240 Bildpunkten und eine Bitrate von 291 kb/s.

In dem Video befinde sich 576 unterschiedliche Wörter. Dabei ist jedes Wort in vielen aufeinanderfolgenden Bildern zu erkennen. Die Bildanzahl pro Wort ist stark unterschiedlich und schwankt zwischen knapp 10 Bildern bis zu einigen 1.000 Bildern. Durch die Kamerabewegung kann sich die Position jedes Wortes über die aufeinanderfolgenden Bilder ändern.

In seltenen Fällen wird ein Wort kurzzeitig in einigen Bildern verdeckt und erscheint danach wieder. Insgesamt sind die 576 Wörter in dem Video in 484.820 einzelnen Bereichen zu finden Im Schnitt ist damit jedes Wort in 842 einzelnen Bildern vorhanden. Die Wörter stammen hauptsächlich aus der englischen Sprache. Es befinde sich ebenfalls einige Namen und Formelzeichen unter den Textbereichen, die hier ebenfalls als Wörter angesehen werden können. Neben den Wörtern, die auf die Leinwand projiziert werden, befinde sich auch Grafi en, die eine hohe Kantendichte aufweisen.

3.6 Zuverlässigkeit der Detektion von Text

Für die Bestimmung der Zuverlässigkeit der Detektion von Text wird das Video, beschrieben in Abschnitt 3.5, verwendet. Aus der manuellen Annotation des Videomaterials sind die Positionen der Wörter bekannt. Für die Bestimmung der Genauigkeit der Textdetektion werden an dieser Stelle die manuell erstellten Textkoordinaten mit den automatisch gefundenen verglichen. Wie in Abschnitt 3.4.6 beschrieben können die manuell erstellten Textkoordinaten von den automatisch detektierten abweichen, obwohl sie den selben Text beinhalten. Für die Auswertung der Übereinstimmung werden im Folgenden 50% und 70% Überdeckung gewählt.

Werden die durch die automatische Textdetektion gefundenen Textkoordinaten mit den von Hand annotierten Textkoordinaten verglichen, sind drei Aussagen über die jeweilige Detektion denkbar. Als erstes kommt eine Übereinstimmung in Frage. In diesem Fall wird von einer positiv richtigen Detektion (TP, engl. True Positive) gesprochen. Zweitens ist es möglich, dass die durch die Verfahren gefundenen Koordinaten durch die per Hand markierten Zeitpunkte nicht gefunden wurde. Diese somit nicht detektierten Schriftzeichen werden als negative falsche Detektion (FN, engl. False Negative) bezeichnet. In dem dritten Fall ist der Textbereich nicht per Hand, aber durch die Verfahren detektiert worden. Dieser Fall wird positive falsche Detektion (FP, engl. False Positive) genannt. Die unterschiedlichen Fälle können dann in N_{TP} für TP, N_{FN} für FN und N_{FP} für FP gezählt werden.

Die Detektionssrate (R, engl. Recall) ergibt sich dann aus:

$$R = \frac{N_{TP}}{N_{TP} + N_{FN}},\ P\ \in [0,1] \tag{3.21}$$

Die Präzision (P, engl. Precision) ergibt sich dann aus:

Zu jeder Rate der positiv richtig erkannten Textbereiche lässt sich auch die Rate der positiv falsch erkannten Schriftzeichen (F_{PR}, engl. False Positive Rate) berechnen:

$$P = \frac{N_{TP}}{N_{TP} + N_{FP}},\ P\ \in [0,1] \tag{3.22}$$

Die beiden Werte R und P befinde sich in Zahlen zwischen 0 und 1.

Aus dem Testmaterial ergeben sich die Messungen in Tabelle 3.3 für die Genauigkeit der Textdetektion. Die Detektionsrate R bei einer Überdeckung von 70% liegt bei 0,3509 bzw. 35,09% und besitzt dabei eine Präzision P von 0,8625 bzw. 86,25%. Die Detektionsrate R bei einer Überdeckung von 50% liegt dagegen höher mit 0,3762 bzw. 37,62% und besitzt dabei eine Präzision P von 0,9642 bzw. 96,42%. Auffällig ist hier die deutlich höhere Präzision.

Überdeckung	N_{TP}	N_{FN}	N_{FP}	R	P
70%	48132	89043	7671	0,3509	0,8625
50%	51604	85571	1914	0,3762	0,9642

Tabelle 3.3: *Aus den Zählungen von N_{TP}, N_{FN} und N_{FP} lassen sich die Detektionsrate R und die Präzision P für die Schriftzeichen bestimmen.*

Insgesamt werden also ca. 1/3 aller Textpositionen durch die Textdetektion automatisch gefunden und dabei ist nur 1 von 25 falsch detektiert.

4. Erkennung von Schriftzeichen

Die Detektion bzw. das Auffinde von Schriftzeichen und das Erkennung bzw. die Interpretation von Schriftzeichen unterscheiden sich stark. Durch die Detektion von Schriftzeichen werden ausschließlich Bereiche in einem Bild oder Video gefunden, die eventuell Text enthalten. Die Detektion kann aber keine Aussage über die Schriftzeichen selbst geben. Die Analyse dieser Bereiche geschieht durch die Erkennung bzw. Interpretation, welches in diesem Kapitel beschrieben wird.

4.1 Binarisierung eines Bildes zum Filtern der Schriftzeichen

In der Detektion von Text wurden Kantenbilder nach dem Verfahren von Canny verwendet. Als Resultat liefert die Detektion rechteckige Markierungen in dem Orginalbild. Für die anschließende Erkennung der Schriftzeichen in diesen Bereichen müssen die Schriftzeichen vom Hintergrund getrennt werden. Dazu wird ein Schwarz-Weiß-Bild erzeugt, bei dem die Bildpunkte der Schriftzeichen Schwarz und die restlichen Bildpunkte Weiß werden. Dieses Verfahren wird Binarisierung genannt. Die Binarisierung von solchen Bildern ist sehr komplex. Die Schwierigkeit in der Trennung von Text und Hintergrund liegt vor allem bei sehr strukturierten Hintergründen. Hier spielen die Faktoren Kontrast und Farben eine große Rolle. In der Literatur gibt es verschiedene Ansätze, um das Ziel zu erreichen. In [40] wird versucht, ein kleines Fenster über die entsprechenden Bereiche wandern zu lassen und durch vordefiniert Parameter Bildpunkte als Text oder Hintergrund zu definieren Andere Verfahren versuchen statt aufwendig zu bestimmende Parameter, mathematische Modelle zu definieren Hiefür werden in [41] Markov Feld Modelle oder in [42] Neuronale Netzwerke verwendet. In dieser Arbeit wird die Binarisierung der Software LEADTOOLS OCR SDK [43] verwendet.

4.2 Trennung der Schriftzeichen innerhalb eines Textbereiches

Nachdem die Textstellen in dem Bild markiert wurden (vgl. 3.2.4), müssen an dieser Stelle die einzelnen Schriftzeichen aus jeder markierten Textstelle separiert werden. Die Trennung der Schriftzeichen eines Textbereiches besitzt eine ähnliches Verfahren wie die Berechnung des Textbereiches im Bild (vgl. 3.2.4). Es werden mehrere Parameter verwendet, um Schriftzeichen abzugrenzen. Danach werden die erkannten Schriftzeichen auf die vollständige Größe kontrolliert und nach falsch erkannten Bereichen durchsucht.

4.2.1 Berechnung der Schriftzeichengrenzen in der horizontalen Ebene

Die markierten Textbereiche sind rechteckig und befinde sich in dem Kantenbild. An dieser Stelle werden alle markierten Bereiche einzeln auf Schriftzeichen durchsucht. Jedes gefundene Schriftzeichen wird in einer Liste gespeichert. Diese Liste befinde sich in einer weiteren Liste, welche die markierten Bereiche beinhaltet. Damit entsteht eine Liste von Textstellen, die eine Liste mit Bildern von Schriftzeichen enthält.

Jede markierte Textstelle wird von links nach rechts untersucht, damit der später entstehende Text in der richtigen Reihenfolge gespeichert werden kann. Schriftzeichen besitzen in der horizontalen Richtung keine Lücken. In der vertikalen Richtung können Lücken auftreten. Ein Beispiel dafür ist das Schriftzeichen i. Es ist getrennt in zwei Bereiche. Weiterhin sind zwischen Schriftzeichen in der Druckschrift keine Verbindungen. In manchen Schriftsätzen kann es zu stilistischen Verbindungen kommen. In dieser Arbeit wird von Schriftsätzen ohne Verbindungen ausgegangen. Auf Grund dieser Annahmen können die Schriftzeichen durch die Lücken in der horizontalen Richtung getrennt werden.

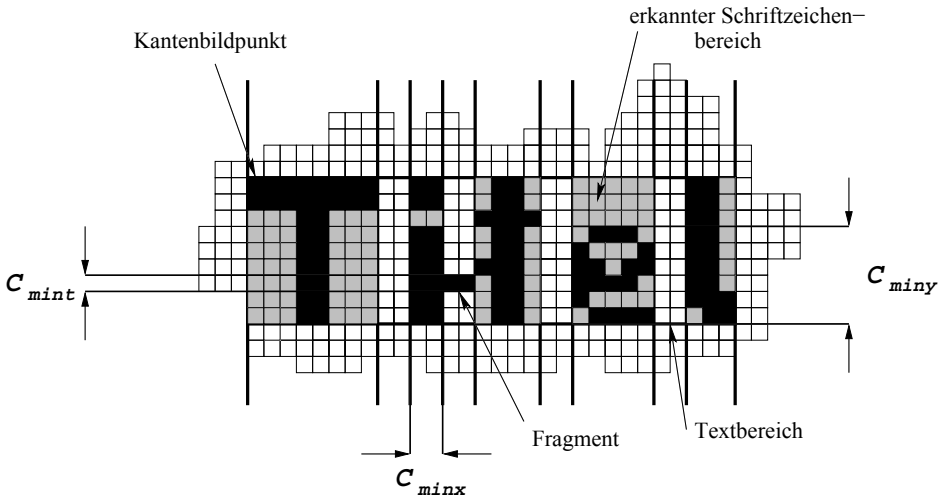


Abbildung 4.1: *In dem Bild ist das Wort Titel enthalten. Schwarze Bildpunkte sind Kanten, weiße Bildpunkte sind leer und graue Bildpunkte sind markierte Bereiche. Durch die Parameter C_{mint},C_{minx} und C_{miny} werden die Bereiche der Schriftzeichen begrenzt.*

Durch Bildfragmente und Rauschen können zwischen Buchstaben kleine Verbindungen existieren. Diese kleinen, oft sehr schmalen Verbindungen können mit Hilfe von Parametern berücksichtigt werden. Durch den Parameter C_{mint} werden Verbindungen von den Schriftzeichen, die kleiner als dieser Wert sind, als nicht vorhanden angesehen. Weiterhin kann ein Schriftzeichen nur eine minimale Größe besitzen, die durch die Parameter C_{minx} und C_{miny} begrenzt ist (Abb. 4.1).

4.2.2 Berechnung der Schriftzeichengrenzen in der vertikalen Ebene

Durch die Berechnung des Textbereiches (vgl. 3.2.4) sind für jedes markierte Schriftzeichen Grenzen in der vertikalen Richtung vorgegeben. Diese Grenzen können zu groß oder zu klein sein. Beispiele dafür sind die Wörter „Titel“ und „eigen“. In dem Wort „Titel“ ist das Schriftzeichen „e“ nicht füllend in den Grenzen. Dagegen besitzen die Schriftzeichen „i“ und „g“ in dem Wort „eigen“ eine größere Ausdehnung als die durch die Berechnung des Textbereiches entstandene Abgrenzung der Schriftzeichen (vgl. 3.2.4). Für die Merkmalextraktion müssen die Schriftzeichen bildfüllend in dem markierten Bereich sein (vgl. 4.3). Die Markierung muss genau auf das Schriftzeichen begrenzt werden. Deshalb muss die Abgrenzung der Schriftzeichen vergrößert bzw. verkleinert werden, so dass die Schriftzeichen komplett in der Begrenzung liegen und kein Freiraum zwischen der Begrenzung und dem Schriftzeichen existiert.

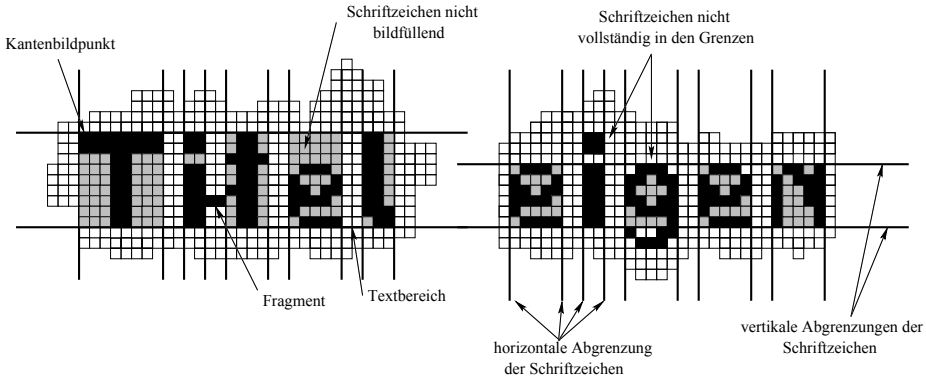


Abbildung 4.2: *Die Wörter „Titel“ und „eigen“ besitzen Schriftzeichen, die nicht vollständig von den Grenzen erfasst sind (Schriftzeichen „i“ und „g“), bzw. bei denen die Grenzen nicht abschließend sind und damit zu große Grenzen besitzen (Schriftzeichen „e“).*

Die Grenzen werden als erstes so weit verkleinert, das keine Freiräume zwischen den Grenzen und den Schriftzeichen in Form von Freizeilen vorkommen. Danach müssen noch die Grenzen der Schriftzeichen vergrößert werden, die nicht vollständig erfasst sind. Hier ist darauf zu achten, dass durch Bildfragmente die Grenzen der Schriftzeichen nicht beliebig erweitert werden. Damit das verhindert wird, muss der Parameter C_{mint} benutzt werden. Dieser Parameter kontrolliert die Dicke in Bildpunkten der Schriftzeichen. Falls der Wert der Dicke den Parameter C_{mint} unterschreitet, ist die Grenze des Schriftzeichens gefunden. Durch sehr große Bildfragmente kann es vorkommen, dass das Schriftzeichen beliebig erweitert werden kann. Damit dieser Effekt verhindert wird, muss die Erweiterung der Grenzen ab einer bestimmten Distanz abgebrochen werden. Der Parameter C_i gewährleistet die Begrenzung der Erweiterung.

Sind die Grenzen von einem Schriftzeichen gefunden, wird es in die Liste der Schriftzeichen des jeweiligen Wortes eingefügt.

4.3 Merkmalextraktion eines Schriftzeichens

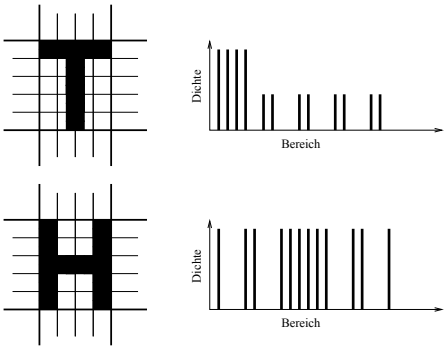


Abbildung 4.3: *Die zwei Schriftzeichen T und H werden in Bereiche unterteilt. Die Dichte der Bildpunkte der Schriftzeichen ergibt die Tabelle rechts.*

Nach der Trennung und Eintragung der Schriftzeichen (vgl. 4.2) in die Liste der Schriftzeichen eines Wortes, besitzen die Schriftzeichen verschiedene Größen in der X- und Y-Achse. Dafür gibt es verschiedene Ursachen. Neben dem Rauschen in den Bildern sind die Schriftzeichen auch unterschiedlich strukturiert. Es gibt Schriftzeichen, welche gegenüber anderen höher bzw. tiefer sind wie zum Beispiel 'M' und 'e' oder 'e' und 'g'. Außerdem haben sie unterschiedliche Längen wie zum Beispiel das 'M' und das 'l'. Je nach Größe der Schriftzeichen gibt es mehr oder weniger Bildpunkte innerhalb der Grenzen eines Schriftzeichens.

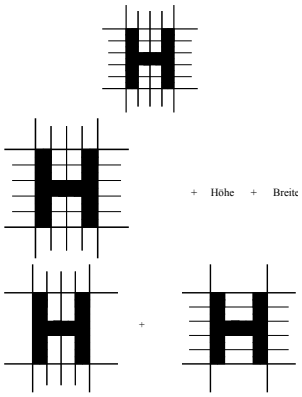


Abbildung 4.4: *Verschiedene Kombination der Merkmale sind für die Erkennung möglich. Die Dichte der Bildpunkte eines Schriftzeichens unterteilt in Bereiche kann allein (Oben) oder mit Kombination der Höhe und Breite (Mitte) des Schriftzeichens vollzogen werden. Eine mögliche Kombination ist auch die Nutzung der Aufteilung in vertikale bzw. horizontale Balken (Unten).*

Für die Schriftzeichenerkennung mit Hilfe von Hidden Markov Modellen (vgl. 4.4) sind nur bestimmte Informationen (Merkmale) wichtig. Es ist nicht notwendig, alle Bildpunkte von einem Schriftzeichen für die Erkennung zu benutzen. Also müssen die wichtigen Informationen in den Bildern der Schriftzeichen gefunden werden. Jedes Schriftzeichen muss außerdem dabei gleich viel Informationen besitzen.

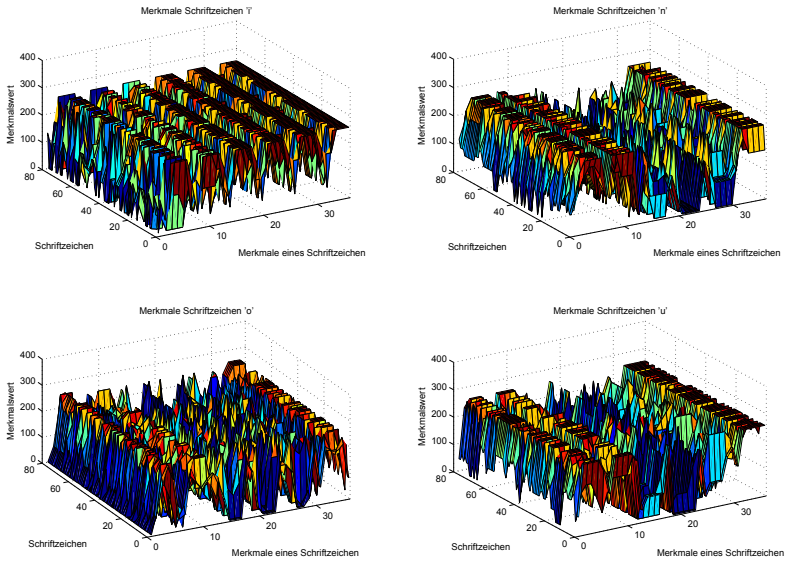


Abbildung 4.5: *Es sind Merkmale des Verfahrens (a) für die Schriftzeichen 'i','n','o' und 'u' zu erkennen. Es wurden 80 Bilder für jedes Schriftzeichen verwendet. Jedes Schriftzeichen besitzt 36 Merkmale.*

Nachdem die Grenzen jedes Schriftzeichens in dem Bild bekannt sind, wird das Originalbild des Videos verwendet. Dieses Bild wird in den Graustufen-Farbraum umgewandelt und auf 1 Bit Farbtiefe reduziert. Damit sind nur noch zwei Farben in dem Bild vorhanden. Weiß besitzt den Wert 1 und Schwarz besitzt den Wert 0. Die Reduzierung der Farben von 8 Bit auf 1 Bit ist so zu wählen, dass die weißen Schriftzeichen sich von dem Hintergrund unterscheiden. Die Unterteilung in einen weißen bzw. schwarzen Bildpunkt kann durch den Parameter B_t eingestellt werden.

Jedes Schriftzeichen befinde sich in rechteckigen Grenzen. Wenn das Schriftzeichen in kleinere Teile unterteilt wird, sind in diesen Bereichen mehr oder weniger weiße Bildpunkte. Die Dichte der weißen Bildpunkte in den Bereichen ist charakteristisch für die jeweiligen Schriftzeichen. So ist die Dichte des gleichen Schriftzeichens in einem Bildbereich immer ähnlich. Werden die Dichtegrade der einzelnen Bereiche eines Schriftzeichens mit denen anderer Schriftzeichen verglichen, können auf diese Weise Unterschiede erkannt werden. Diese Unterschiede können für die Erkennung der Schriftzeichen mit Hilfe von Hidden Markow Modellen (vgl. 4.4) benutzt werden(Abb. 4.3). Wichtig ist es, dass die Unterteilung bei allen

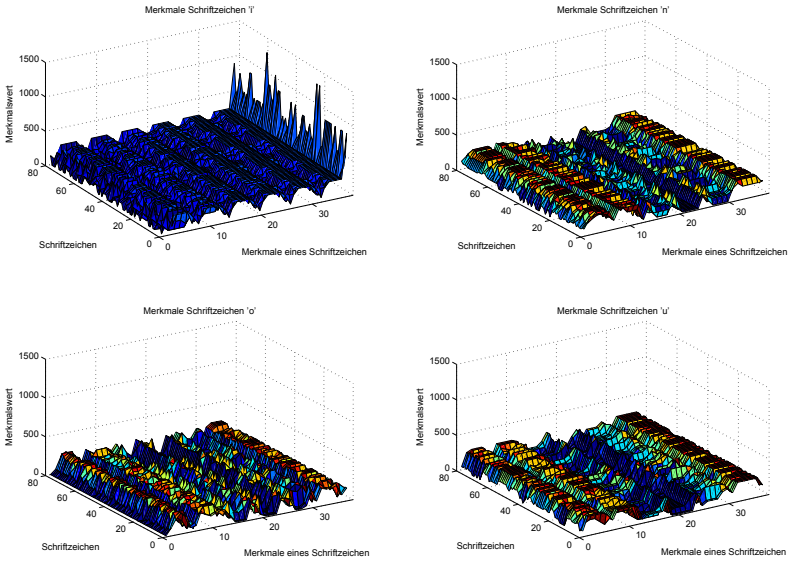


Abbildung 4.6: *Es sind Merkmale des Verfahrens (b) für die Schriftzeichen 'i','n','o' und 'u' zu erkennen. Es wurden 80 Bilder für jedes Schriftzeichen verwendet. Jedes Schriftzeichen besitzt 36+2 Merkmale.*

Schriftzeichen prozentual auf die Größe vollzogen werden. Es müssen immer gleich viele Merkmale entstehen. Es können also unterschiedlich viele Bildpunkte in einem unterteilten Bereich vorhanden sein. Eine weitere Unterscheidung der Schriftzeichen kann, wie oben beschrieben, durch die unterschiedliche Höhe und Breite durchgeführt werden.

Die Unterteilung der Bereiche der Schriftzeichen kann auf verschiedene Arten geschehen. Die erste Unterteilung ist die Aufteilung in ein Raster aus Quadraten. Die Schriftzeichen können auch durch vertikale oder horizontale Balken unterteilt werden.

Für die Symbolerkennung mit Hilfe von Hidden Markow Modellen (vgl. 4.4) sind verschiedene Kombinationen von Unterteilungen und anderen Eigenschaften (Abb. 4.4) möglich. Die Werte aus der Zusammenstellung der Merkmale jedes Schriftzeichen werden als Vektor gespeichert.

In der Merkmalextraktion (Abb. 4.5 (a)) werden die Dichten der weißen Bildpunkte für die einzelnen Bereiche in dem Bild des Schriftzeichens berechnet. Die Bereiche sind in kleine Quadrate unterteilt. Die Vektoren der Bilder gleicher Schriftzeichen werden in einer Textdatei gespeichert. In der Abb. 4.5 befinde sich eine Darstellung der Merkmale der Schriftzeichen 'i','n','o' und 'u' im Vergleich. Es wurden dafür je die Merkmale von 80 Schriftzeichen verwendet. Jedes Schriftzeichen besitzt dabei 36 Merkmale. Werden die Merkmale der Bilder eines Schriftzeichens aneinander gereiht, ist eine für Schriftzeichen spezifisch Struktur erkennbar. Zu den 36 Merkmalen der Merkmalsextraktion (Abb. 4.5 (a)) werden zusätzlich in der Merkmalsextraktion (Abb. 4.6 (b)) die Höhe und die Breite der Schriftzeichen abgespei-

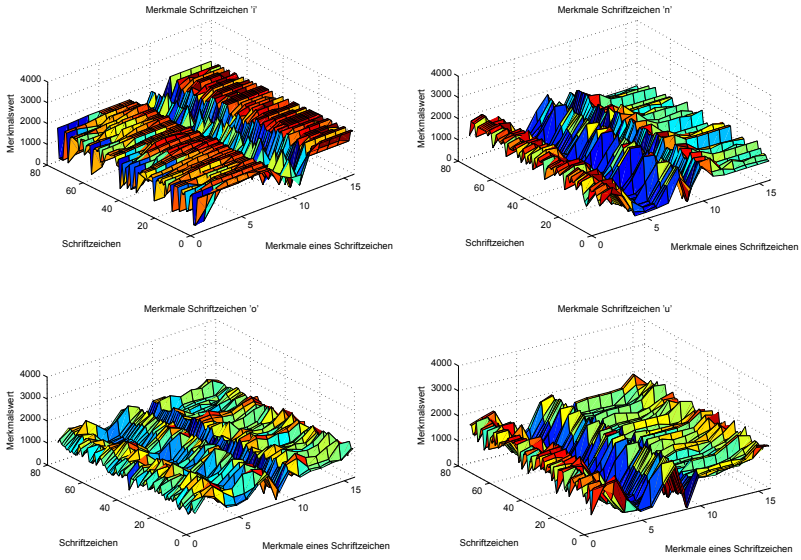


Abbildung 4.7: *Es sind Merkmale des Verfahrens (c) für die Schriftzeichen 'i','n','o' und 'u' zu erkennen. Es wurden 80 Bilder für jedes Schriftzeichen verwendet. Jedes Schriftzeichen besitzt 16 Merkmale.*

chert. Die Darstellung der Merkmale befinde sich in der Abb. 4.6. Es ist ein hoher Wert in den Merkmalen der Höhe bei den Bildern des Schriftzeichens 'i'. erkennbar.

Die Darstellung der Merkmale aus der Merkmalsextraktion (Abb. 4.7 (c)) befinde sich in der Abb. 4.7. Die Bereiche der Bilder für die Bestimmung der Merkmale wurden bei diesem Verfahren in je 8 vertikale und horizontale Streifen unterteilt. Insgesamt werden so für jedes Schriftzeichen 16 Merkmale gefunden.

4.4 Schriftzeichenerkennung mit Hilfe von Hidden Markov Modellen

Nachdem durch die Merkmalsextraktion eines Schriftzeichens (vgl. 4.3) ein Vektor mit den wichtigsten Informationen existiert, muss das Bild von dem Schriftzeichen interpretiert werden als American Standard Code for Information Interchange Zeichen (ASCII-Zeichen). ASCII-Zeichen wurden 1963 durch das American National Standards Institute (ANSI) definiert Es gibt 128 verschiedene ASCII-Zeichen.

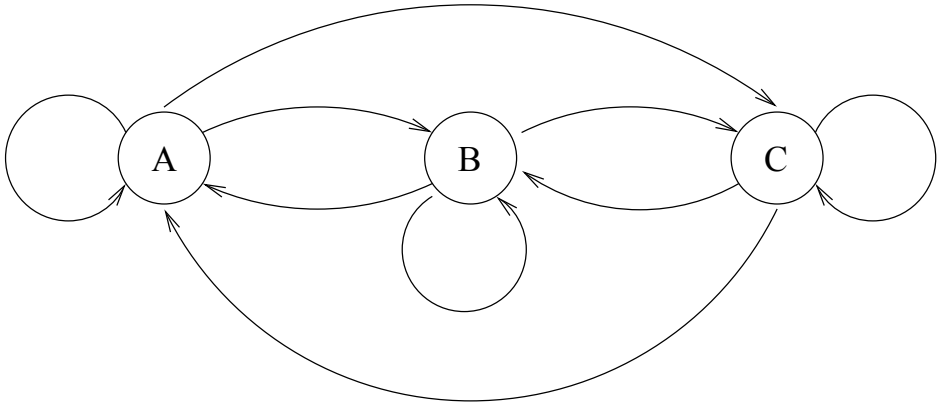


Abbildung 4.8: *Es gibt in diesem Markov Modell 3 Zustände. Jeder Zustand kann in einen anderen Zustand mit einer festen Wahrscheinlichkeit wechseln oder in dem selben Zustand bleiben.*

Die Merkmale in den Vektoren gleicher Schriftzeichen besitzen nicht die selben Werte. Durch Rauschen oder Verzerrungen des Bildes sind die Werte unterschiedlich. Es ist nicht möglich, die Veränderungen der Schriftzeichen durch Regeln zu erfassen. Dadurch lässt sich nur schwer eine deterministische Aussage treffen, zu welchem ASCII-Zeichen das jeweilige Schriftzeichen gehört. Für die Beschreibung solcher im Verborgenen ablaufenden nichtdeterministischen Vorgänge bieten sich statistische Modellierungen wie die Hidden Markov Modelle [44] an. Neben den Hidden Markov Modellen werden auch Support Vector Machine [45] und Neuronale Netzwerke [46] verwendet.

Ein Markov Modell besitzt n Zustände (Abb. 4.8). Zu diskreten Zeitpunkten t geht das System vom Zustand i in den Zustand j über mit folgender Übergangswahrscheinlichkeit:

$$a_{ij} = P(\omega_j(t+1)|\omega_i(t)) \tag{4.1}$$

Die Übergangswahrscheinlichkeiten können in einer Matrix A (Gl. 4.2) gespeichert werden. Jede Zeile in der Matrix repräsentiert alle Übergangswahrscheinlichkeiten eines Zustan-

des.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{4.2}$$

Die Summe der Übergangswahrscheinlichkeiten einer Zeile ergibt immer 1.

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{4.3}$$

In der Diagonalen von links oben nach rechts unten in der Matrix A stehen die Wahrscheinlichkeiten, in denen der Zustand i und j derselbe ist, und damit kein Wechsel des betrachteten Zustandes stattfindet Mit diesem System kann die Wahrscheinlichkeit, mit der ein bestimmter Zustand vorliegt, ausgerechnet werden. Es muss weiterhin bekannt sein, wie hoch die Wahrscheinlichkeiten zum Zeitpunkt der Initialisierung sind. Die Werte der Wahrscheinlichkeiten befinde sich in dem Vektor π. Das Markov Modell wird durch das Tupel (A, π) vollständig beschrieben.

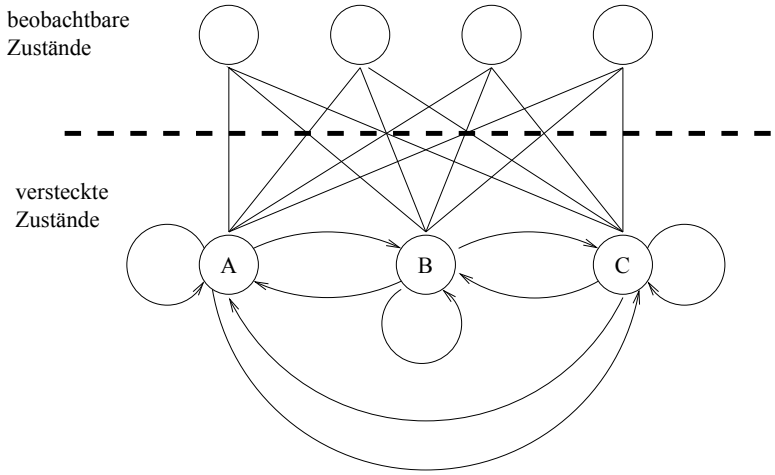


Abbildung 4.9: *Es gibt in diesem Hidden Markov Modell 3 Zustände. Jeder Zustand kann in einen anderen Zustand mit einer festen Wahrscheinlichkeit wechseln oder in denselben Zustand bleiben. Außerdem sind 4 sichtbare Zustände vorhanden.*

Das System kann aber nicht direkt beobachtet werden, da keine Informationen über die Zuordnung der Schriftzeichenmerkmale zu den Zuständen existieren. Daher müssen an dieser Stelle Hidden Markov Modelle verwendet werden, in denen die Zustände nicht sichtbar sind (Abb. 4.9). Hidden Markov Modelle bestehen aus n Zuständen, die jedoch nicht direkt beobachtet werden können. Jeder Zustand $\omega(t)$ emittiert zu jedem Zeitpunkt t ein zufällig

ausgewähltes sichtbares Symbol, das auch als sichtbarer Zustand bezeichnet wird. Das gesamte System generiert beim Durchlaufen von t verborgenen Zuständen die Sequenz

$$V^T = \{v(1), v(2), ..., v(T)\} \tag{4.4}$$

Für die Übergangswahrscheinlichkeiten gilt:

$$a_{ij} = P(\omega_j(t+1)|\omega_i(t)) \tag{4.5}$$

Die Wahrscheinlichkeit für die Emission eines Symbols v zum Zeitpunkt t, wenn sich das System im Zustand $\omega(t)$ befindet lautet:

$$b_j(v(t)) = P(v(t)|\omega_j(t)) \tag{4.6}$$

$b_j s$ kann in der Zustands-Symbolmatrix B zusammengefasst werden. Es gibt einen Vektor π, der die initiale Wahrscheinlichkeitsverteilung angibt. Ein Hidden Markov Modell ist daher über (A, B, π) definiert Außerdem gilt für die $a_{ij} s$ und analog für die $b_j s$ die Normalisierungsbedingung:

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{4.7}$$

4.4.1 Typen von Hidden Markov Modellen

Es gibt ergodische und Links-Rechts Hidden Markov Modelle. In den ergodischen Hidden Markov Modellen können alle Zustände von jedem Zustand aus innerhalb eines Schrittes mit der Wahrscheinlichkeit > 0 erreicht werden (Abb. 4.9). Die Zustandsmatrix darf an keiner Stelle 0 sein (Gl. 4.2). In einem Links-Rechts Modell geht das System mit jedem Schritt entweder in dem selben Zustand oder zu einem noch nicht besuchten Zustand über. Daraus folgt: $a_{ij} = 0$ für $j < i$. Das Hidden Markov Modell hat dann eine längliche Form.

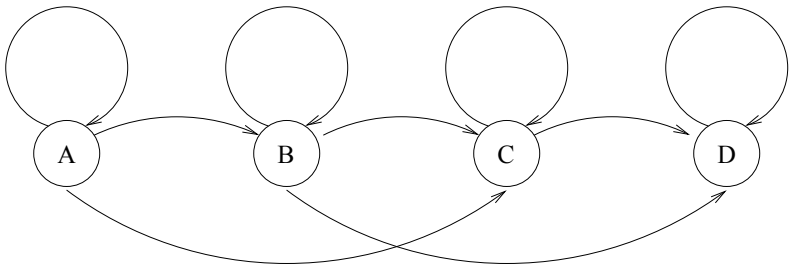


Abbildung 4.10: *Es ist ein Links-Rechts Hidden Markov Modell mit 4 verdeckten Zuständen dargestellt.*

Die Übergangswahrscheinlichkeitsmatrix lautet:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix} \tag{4.8}$$

4.4.2 Training mit dem Baum-Welch Algorithmus

Das Trainieren des Hidden Markov Modells geschieht mit der Neueinschätzung der vordefi nierten Modellwerte mit dem Baum-Welch Algorithmus. Dieser Algorithmus ist eine spezielle Form des Expectation-Maximization Algorithmus (EM). Er finde ein lokales Optimum von dem Parameterset λ für die klassenbedingte Wahrscheinlichkeit $P(O|\lambda)$. O ist hier die klassenvordefiniert Überwachungssequenz von den Trainingsdaten. Klassen sind hier die von Hand voreingeteilten, verschiedenen und zu erkennenden Schriftzeichen. Die Anzahl an Zuständen ist vordefiniert Für die vordefinierte Werte des Modells werden das ergodische oder Links-Rechts Modell (vgl. 4.4.1) benutzt mit der initialen Wahrscheinlichkeitsverteilung π.

Die nächsten zwei Schritte sind die Abschätzung der Wahrscheinlichkeiten des aktuellen Modells durch die Trainingdaten und die Überprüfung der Modellparameter A,σ_j und π für die besten Wahrscheinlichkeiten. Abschließend wird für jede Klasse ein Parametersatz für die spätere Erkennung gespeichert.

4.4.3 Klassifikation mittels Viterbi Algorithmus

Der Pfad Q mit der höchsten Wahrscheinlichkeit für eine gegebene Sequenz von Überwachungen O und der dazugehörigen maximalen Wahrscheinlichkeit für den Zustandsgraph ist definier durch den Viterbi Algoritmus [47]. Der wahrscheinlichste Pfad Q_i für das Modell λ von der Klasse i aus allen Klassen C errechnet sich aus:

$$Q_i = \arg\max_Q P(Q, O|\lambda_i) \tag{4.9}$$

Die maximale Wahrscheinlichkeit $P(Q_i, O|\lambda_i)$ wird dann durch die Abhängigkeiten von dem Zustandsgraph Q_i und der zu klassifizierende Daten berechnet. Die Klassenbeziehung von den Merkmalvektoren ist abhängig von der Klasse ω_i mit dem maximalen Wert der klassenabhängigen Wahrscheinlichkeit des besten Pfades:

$$w_i = \arg\max_{1 \leq i \leq c} P(Q, O|\lambda_i) \tag{4.10}$$

4.4.4 Zusammensetzung von Texten aus erkannten Symbolen

Für jedes Bild eines Schriftzeichens existiert nach der Klassifizierun ein ASCII-Zeichen (vgl. 4.4.3). Die Bilder der Schriftzeichen befinde sich in Listen. Jedes Wort ist eine solche Liste von Bildern. Die Wörter sind in einer weiteren Liste gespeichert, die den gesamten Text eines Bildes aus den Nachrichtenvideos enthält. Dieser Text ist repräsentativ für die ganze Kameraeinstellung. Damit der Text zusammengefügt wird, müssen die einzelnen

ASCII-Zeichen für die jeweiligen in der Liste befindliche Schriftzeichen in der vorgegebenen Reihenfolge herausgenommen werden und zu einem Wort zusammengefügt werden. Die so entstehenden Wörter, die nicht mehr aus Bildern, sondern aus ASCII-Zeichen bestehen, werden in einer zusätzlichen Liste gespeichert.

4.4.5 Für weitere Erkennung von Text verwendete Programme

Es hat sich außerdem gezeigt, dass die Anzahl und die Auswahl der Trainingsdaten für die Modelle aller Schriftzeichen entscheidend ist. Die in dieser Arbeit benutzten Trainingsdaten sind sehr gering und beziehen sich ausschließlich auf eine Schriftart. Für diese Art von Video und Schriftart ist die Erkennungsrate ausreichend. Da andere Schriftarten nicht in den Trainingsdaten vorhanden sind, ist der Einsatz der hier entwickelten Schriftzeichenerkennung unzureichend. Aufgrund der großen Vielfalt der verschiedenen Schriftzeichenarten übersteigt der zeitliche Aufwand einer Erstellung einer ausreichenden Bibliothek an Trainingsdaten für alle Schriftzeichenarten den Rahmen dieser Arbeit. Es existieren bereits gut entwickelte Programme für die Schriftzeichenerkennung. Zu diesen zählen Abbyy FineReader [48], GOCR [49] und LEADTOOLS OCR SDK [43]. Jede dieser Programme hat gute Erkennungsraten für Schriftzeichenerkennung. Die Detektion von Text in Bildern ist jedoch bei diesen Programmen nicht vorgesehen. Für das Gesamtsystem zur Detektion, Erkennung und Fehlerreduktion von Text wird die Erkennung von Text durch das Programm LEADTOOLS OCR SDK ersetzt. Für die Detektion und die Fehlerreduktion von Text werden die in dieser Arbeit entwickelten Verfahren verwendet.

4.5 Zuverlässigkeit der Schriftzeichenerkennung

Für die Bestimmung der Zuverlässigkeit der Erkennung von Text durch das Programm LEAD TOOLS OCR SDK wird das Video, beschrieben in Abschnitt 3.5, verwendet. Die Auswertung der Erkennungsrate der Texterkennung muss unabhängig von der Textdetektion durchgeführt werden. Deshalb werden an dieser Stelle nicht die von der Textdetektion gefundenen Textpositionen für die Messungen verwendet. Aus der manuellen Annotation des Videomaterials sind nicht nur die Wörter, sondern auch die Positionen im Video bekannt. Für die Bestimmung der Erkennungsrate der Texterkennung werden an dieser Stelle die bekannten Positionen aus der manuellen Textdetektion verwendet. Mit Hilfe der Koordinaten dieser manuellen Annotation können nun die Bildbereiche mit dem zu erkennenden Text aus dem Video ermittelt werden. Diese Bildbereiche werden dann mit Hilfe des Programms LEADTOOLS OCR SDK auf Text untersucht und anschließend mit den manuell annotierten Wörtern verglichen. Die Erkennungsrate kann unter zwei verschiedenen Aspekten ausgewertet werden. Zum einen ist die Anzahl der richtig erkannten Wörter und zum anderen die Anzahl der richtig oder falsch erkannten Schriftzeichen entscheidend.

Werden die durch die automatische Texterkennung gefundenen Schriftzeichen mit den von Hand annotierten Schriftzeichen verglichen, sind drei Aussagen über die jeweilige Erkennung denkbar. Als erstes kommt eine Übereinstimmung in Frage. In diesem Fall wird von einer positiv richtigen Erkennung (TP, engl. True Positive) gesprochen. Zweitens ist es möglich, dass die durch die Verfahren gefundenen Schriftzeichen nicht durch die per Hand markierten Zeitpunkte übereinstimmen oder nicht gefunden wurden. Diese somit falsch er-

kannten Schriftzeichen werden als negative falsche Erkennung (FN, engl. False Negative) bezeichnet. In dem dritten Fall ist ein Schriftzeichen nicht per Hand, aber durch die Verfahren erkannt worden. Dieser Fall wird positive falsche Erkennung (FP, engl. False Positive) genannt. Die unterschiedlichen Fälle können dann in N_{TP} für TP, N_{FN} für FN und N_{FP} für FP gezählt werden.

Die Erkennungsrate (R, engl. Recall) ergibt sich dann aus:

$$R = \frac{N_{TP}}{N_{TP} + N_{FN}},\ P\ \in [0,1] \tag{4.11}$$

Die Präzision (P, engl. Precision) ergibt sich dann aus:

Zu jeder Rate der positiv richtig erkannten Schriftzeichen lässt sich auch die Rate der positiv falsch erkannten Schriftzeichen (F_{PR}, engl. False Positive Rate) berechnen:

$$P = \frac{N_{TP}}{N_{TP} + N_{FP}},\ P\ \in [0,1] \tag{4.12}$$

Die beiden Werte R und P befinde sich in Zahlen zwischen 0 und 1.

N_{TP}	N_{FN}	N_{FP}	R_s	P_s
197.203	837.986	7310	0,1905	0,9050

Tabelle 4.1: *Aus den Zählungen von N_{TP}, N_{FP} und N_{FN} lassen sich die Erkennungsraten R_s und die Präzision P_s für die Schriftzeichen bestimmen.*

Aus dem Testmaterial ergeben sich die Messungen in Tabelle 4.1 für die Erkennung der Schriftzeichen. Die Erkennungsrate R_s liegt bei 0,1905 bzw. 19,05%. Das heißt 19,05% aller vorkommenden Schriftzeichen wurden gefunden. Der Wert P_s gibt Aufschluss über die dabei zusätzlich erkannten Schriftzeichen, die nicht im Video existieren, im Verhältnis der richtig gefundenen Schriftzeichen. P_s liegt hier bei 0,9042 bzw. 90,42%.

Im nächsten Schritt werden die gefundenen Wörter ausgewertet. Werden die durch die automatische Texterkennung gefundenen Wörter mit den von Hand annotierten Schriftzeichen verglichen, sind zwei Aussagen über die jeweilige Erkennung denkbar. Hier kann nur ausgesagt werden, ob das Wort richtig oder falsch ist. Durch die Vorgabe der Textpositionen aus der manuellen Annotation ist hier ausgeschlossen, dass Wörter zu viel oder zu wenig erkannt werden. Damit existieren nur die Werte N_{TP} und N_{FP}. Daher lässt sich auch nur die Erkennungsrate R_w bestimmen.

N_{TP}	N_{FP}	R
68.134	102.971	0,3982

Tabelle 4.2: *Aus den Zählungen von N_{TP} und N_{FP} lässt sich die Erkennungsrate R_w für die Wörter bestimmen.*

Die Erkennungsrate R der Wörter aus Tabelle 4.2 liegt mit 0,3982 bzw. 39,82% deutlich über der Erkennungsrate der einzelnen Schriftzeichen. Die Diskrepanz entsteht hier durch die

hohe Anzahl von falschen Schriftzeichen in einem nicht korrekt erkannten Wort. Mit knapp 40% der zu erkennenden Schriftzeichen und Wörter ist nur $2/5$ vom Text gefunden worden.

4.6 Fehler innerhalb der Erkennung auf optimalen Bildern

Die Schriftzeichenerkennung basiert auf Modellen für jedes zu erkennende Schriftzeichen. Jedes dieser Modelle muss mit Hilfe von Beispielen trainiert werden. Da es nahezu unendlich viele Formen und Größen für jedes Schriftzeichen gibt, sind fehlende Beispiele für ein Schriftzeichen zu erwarten. Aufgrund der mangelnden Beispiele für das Training der Modelle sind Fehler zu erwarten. Um die Fehler der Texterkennung auf idealen Bildern zu zeigen, werden im Folgenden ideale Bilder mit verschiedenen Schriftsätzen erstellt. In jedem Bild befinde sich die Schriftzeichen „a“ bis „z“ und „A“ bis „Z“ in der Schriftgröße 8. Die Schriftzeichen werden direkt als unkomprimiertes Bild gespeichert und anschließend durch die Texterkennung analysiert.

Schriftart	richtig	falsch	Erkennungsrate
Arial	48	4	0,92
Courier	35	17	0,67
MS Sans Serif	51	1	0,98
Segoe Ul	30	22	0,58
Times New Roman	43	9	0,82
Alle	207	53	0,80

Tabelle 4.3: *Ergebnisse der Texterkennung von jedem möglichen Schriftzeichen bei Schriftgröße 8 und 5 verschiedenen Schriftzeichenarten.*

In Tabelle 4.3 wurden 5 verschiedene Schriftzeichenarten getestet. Auffällig ist die unterschiedliche Erkennungsrate zwischen den Schriftzeichenarten. Die Schriftzeichenart „Arial“ und „MS Sans Serif“ erreichen eine sehr hohe Erkennungsrate. Dagegen ist die Erkennungsrate der Schriftzeichenart „Courier“ und „Segoe Ul“ deutlich niedriger. Aus dieser Messung lässt sich schließen, dass selbst bei optimalem Bildmaterial dennoch Fehler in der Texterkennung entstehen können.

5. Korrektur von Fehlern

Durch die Fehler, die in der Schriftzeichenerkennung entstehen, ist die vollständige und fehlerlose Suche von Wörtern in Videos nahezu unmöglich. Der entstehende Fehler muss so minimal wie möglich gehalten werden. Da in der Bildanalyse Fehler nicht ausgeschlossen werden können, müssen weitere Schritte nach der Detektion und Erkennung von Text gesucht werden, um die Fehler zu reduzieren. Das folgenden Kapitel befasst sich mit der Analyse der erkannten Schriftzeichen, deren Interpretation und damit der in dieser Arbeit entwickelten Methode der Fehlerreduktion.

5.1 Korrektur der Fehler durch ein Wörterbuch

Eine simple Methode zur Reduktion der Fehler ist, nach der Schriftzeichenerkennung jedes erkannte Wort mit einem Wörterbuch zu vergleichen. Befinde sich das Wort nicht innerhalb des Wörterbuchs bestehen mehrere Optionen. Zum einen kann das erkannte Wort gelöscht werden. Eine weitere Möglichkeit ist das Wort mit allen Wörtern in dem Wörterbuch zu vergleichen und es durch das, was am nächsten ist, zu ersetzen. Eine dritte Option ist die Kombination aus beiden Entscheidungen. Wenn das Wort zu verschieden zu allen vorhandenen Wörtern in dem Wörterbuch ist, wird es gelöscht und andernfalls durch das am nächsten liegende ersetzt. Auf diese Weise werden alle nicht bekannten Wörter nach der Schriftzeichenerkennung eliminiert. Das Ergebnis ist bei dieser Fehlerkorrektur stark vom Wörterbuch abhängig und bringt damit auch Nachteile mit sich. Befinde sich ein zu erkennendes Wort nicht in dem Wörterbuch, dann wird es zwangsläufi eliminiert. Dementsprechend muss ein Wörterbuch möglichst umfangreich sein, um möglichst viele Wörter erkennen zu können. Hier entstehen einige Probleme. Ein Problem eines Wörterbuches ist die Unvollständigkeit. Als Beispiel dient hier das zu erkennende Wort „abbauen“. Nach der Schriftzeichenerkennung kann es zu dem Ergebnis „ahhauen“ kommen. Befinde sich in dem Wörterbuch nur die Wörter „anbauen“ und „abtauen“, wird dieses Wort durch eines dieser beiden ersetzt und das eigentliche zu erkennende Wort kann nicht gefunden werden. Daher muss ein Wörterbuch möglichst umfangreich sein. Je mehr Wörter in einem Wörterbuch zusammengefasst werden, umso höher steigt die Chance, dass ein falsches Wort bei der Fehlerkorrektur gewählt wird. Als Beispiel dienen hier wieder die gleichen Wörter. Als zu erkennendes Wort ist hier wieder „anbauen“ und durch die Schriftzeichenerkennung wurde „ahhauen“ gefunden. Befinde sich in dem Wörterbuch nur das Wort „anbauen“, ist es eindeutig, dass das Wort die korrekte Lösung ist. Befinde sich aber noch das Wort „abtauen“ sind beide Wörter als Lösung möglich und es muss eins gewählt werden. Hier ist dann nur noch eine 50%ige Wahrscheinlichkeit gegeben, das richtige Wort zu wählen. Als Resultat gibt es also Vor- und Nachteile, wenn das Wörterbuch klein gehalten wird oder möglichst umfangreich ist. Fehlerhafte Wörter können auch aus dem Zusammenhang mit anderen Wörtern korrigiert werden. Ein solches Verfahren wurde in [50] und [34] vorgestellt.

5.2 Mitteln des Bildes über die Zeit

Sind die Koordinaten der einzelnen Textbereiche bekannt, können die einzelnen Bilder mit Hilfe eines Filters kombiniert werden. Bildlich gesehen werden alle Einzelbilder übereinander gelegt. Auf diese Weise werden mögliche Fehler innerhalb einzelner Bilder über die Zeit durch Mittelung eliminiert. Ein solches Verfahren wurde in [51] veröffentlicht. In diesem Verfahren wurde auf einzelnen Textblöcken eine Erkennungsrate von 53% erreicht. Durch Mitteln der Bilder der Textblöcke über die Zeit wurde eine Steigerung der Erkennung auf 88% möglich.

5.3 Erkennung von dem selben Text auf mehreren Einzelbildern

Bild im Video	Ergebniss der OCR
1	abtauen
2	anbauen
3	butly
4	rfgustly
5	roghhbustly
6	robustdfg
7	robusy
8	hjhhjhj
9	rustly
10	rdfgdfg
11	obustly
12	busghly
13	rdfbustly
14	robu
15	robnstly
16	robufgly
17	robust

Tabelle 5.1: *Nach der Detektion und Erkennung von Text in einzelnen aufeinanderfolgenden Bildern in einem Video von demselben Text ergibt sich eine Liste an möglichen Ergebnissen. In diesem Fall war das englische Wort „robustly“ zu erkennen. Aufgrund der hohen Fehlerrate ist keiner der möglichen Ergebnisse korrekt. Durch einen Fehler der Detektion und Verfolgung des Textes sind die ersten beiden Ergebnisse sogar von einem anderen zu erkennenden Wort in die Liste mit aufgenommen worden.*

In dem in Abschnitt 3.1.1 vorgestellten System zur Detektion und Erkennung von Text ergeben sich für jedes Bild in einem Video einzelne Ergebnisse. Mit Hilfe der in Abschnitt 3.3 eingeführten Verfolgung der Textblöcke über mehrere aufeinanderfolgende Bilder im Vi-

deo können die einzelnen Ergebnisse zugeordnet werden. So entsteht für ein Wort in dem Video eine Liste an möglichen Ergebnissen.

Nach der Detektion und Erkennung von Text in einzelnen aufeinanderfolgenden Bildern in einem Video von demselben Text ergibt sich eine Liste an möglichen Ergebnissen. In diesem Fall war das englische Wort „robustly“ zu erkennen. In Tabelle 5.1 ist eine Liste von Ergebnissen dargestellt. Aufgrund der hohen Fehlerrate ist keines der möglichen Ergebnisse korrekt. Durch einen Fehler der Detektion und Verfolgung des Textes sind die ersten beiden Ergebnisse sogar von einem anderen zu erkennenden Wort mit in die Liste mit aufgenommen worden. Dieses beiden Ergebnisse haben bis auf die örtliche Nähe keine Gemeinsamkeiten zu dem zu erkennenden Wort „robustly“. Dennoch ist bei den anderen Ergebnissen immer eine gewisse Menge an Schriftzeichen korrekt erkannt worden. Es liegt nahe, dass aus den einzelnen falschen Ergebnissen ein korrektes Gesamtergebnis gebildet werden kann.

5.4 Korrektur der Fehler durch Mitteln der Schriftzeichen

Die in Tabelle 5.1 aufgelisteten Ergebnisse der Texterkennung von einzelnen aufeinanderfolgenden Bildern besitzen viele Fehler. In diesem Fall ist keines der Ergebnisse richtig. Um ein korrekte Aussage zu dem zu erkennen Wort zu finde müssen also aus allen einzelnen Ergebnissen Informationen gewonnen werden. Eine grundlegende Idee dafür ist die Schriftzeichen an den Positionen über alle Einzelergebnisse zu mitteln. Ist die Länge der Wörter identisch, ist dies durch einfaches Abzählen und Selektieren der Schriftzeichen an den festen Positionen innerhalb des Wortes möglich.

r	o	b	n	s	t	l	y
r	u	b	u	s	t	l	y
r	o	b	u	s	l	l	y
j	o	b	u	s	t	l	n
r(3) j(1)	o(3) u(1)	b(4)	u(3) n(1)	s(4)	t(3) l(1)	l(4)	y(3) n(1)

Tabelle 5.2: *Zu erkennen ist das englische Wort „robustly“. Die ersten 4 Zeilen zeigen 4 einzelne Ergebnisse von der Texterkennung. Jedes dieser Ergebnisse besitzt einen Fehler und die Länge der Wörter ist identisch. In der letzten Zeile sind die vorhandene Schriftzeichen der jeweiligen Spalte und in Klammern das jeweilige Vorkommen aufgezählt. Durch Vergleich der Anzahl der vorkommenden Zeichen in der jeweiligen Spalte kann das dominierende Schriftzeichen ermittelt werden. Werden die Schriftzeichen mit der höchsten Anzahl je Spalte selektiert und aneinander gereiht, ergibt sich das gesuchte Wort.*

In Tabelle 5.2 ist eine solches Verfahren dargestellt. Trotz der in jedem Einzelergebnis vorkommenden Fehler kann durch Auszählen der Häufig eit der Schriftzeichen in einer Spalte das richtige Schriftzeichen ermittelt werden. Auf diese Weise kann aus allen falschen Teilergebnissen eine korrekte Gesamtaussage für das zu suchende Wort gefunden werden. Durch dieses Verfahren ist also ebenfalls eine Korrektur der Fehler durch die Detektion, Binarisierung und Erkennung von Text möglich und stellt damit eine Alternative zu der Filterung und

Fusion der Einzelbilder und der anschließenden einzelnen Erkennung des Textes dar.

	a	b	tau	e	n		
a	n	b	au	e	n		
		b	u		t	l	y
r	f	g	u	s	t	l	y
r	oghh	b	u	s	t	l	y
r	o	b	u	s	t	df	g
r	o	b	u	s			y
h	j	h	h	j	h	j	
r			u	s	t	l	y
r	d	f	g	d	f	g	
	o	b	u	s	t	l	y
		b	u	s	gh	l	y
r	df	b	u	s	t	l	y
r	o	b	u				
r	o	b	n	s	t	l	y
r	o	b	u	f	g	l	y
r	o	b	u	s	t		

Tabelle 5.3: *In der Tabelle sind alle Ergebnisse der Erkennung der Schriftzeichen von den Einzelbildern in Spalten gegliedert, um gemeinsame Schriftzeichenfolgen zu ermitteln.*

Problematisch wird es aber, wenn die Ergebnisse der Texterkennung auf den Einzelbildern eine unterschiedliche Länge aufweisen. In diesem Fall ist nicht klar, welches Schriftzeichen mit welchem aus den anderen Ergebnissen verglichen werden muss. Die Ursache für eine unterschiedliche Länge ist vielfältig. Einerseits kann es zu einem Fehler in der Detektion von Text kommen, bei dem ein Teil des zu erkennenden Wortes nicht erfasst wird und damit bei der Erkennung der Schriftzeichen schon fehlt. In Tabelle 5.1 ist in der Zeile 3 ein Teil vom Bild mit dem Text auf der linken Seite nicht detektiert worden und damit fehlt in dem Ergebnis der Anfang vom Wort. Es fehlt in diesem Beispiel „ro“ und das einzelne Ergebnis ist dann nur „butly“. Ebenfalls fehlt in dem Ergebnis das Schriftzeichen „s“, welches vermutlich bei der Erkennung eines benachbarten Schriftzeichens verschluckt wurde. Andersrum ist es auch möglich, dass ein Schriftzeichen als mehrere Schriftzeichen interpretiert wurde. In Zeile 14 wurde zum Beispiel das Schriftzeichen „o“ als „df“ interpretiert. Die so entstehende Abweichung der Länge der einzelnen Ergebnisse muss beim Auszählen der Schriftzeichen berücksichtigt werden. In Tabelle 5.3 sind die Schriftzeichen der Einzelergebnisse aus der Tabelle 5.1 in einzelne Spalten gegliedert worden. Ziel war es hier, in jeder Spalte möglichst viele gemeinsame Schriftzeichen zu erhalten. Für eine Einteilung der Spalten muss eine Logik vorhanden sein.

Entscheidend für eine Logik zur Erzeugung einer solchen Tabelle ist es, die Reihenfolge der erkannten Schriftzeichen in den einzelnen Ergebnissen beizubehalten und diese auf die anderen Ergebnisse abzubilden, also gemeinsame Schriftzeichen möglichst in eine Spalte zu bringen. Fehlt ein Schriftzeichen an einer Position im Vergleich zu den anderen Wörtern,

				a	b	t	a	u	e	n			
			a	n	b		a	u	e	n			
					b			u		t		l	y
r				f	g			u	s	t		l	y
r	o	g	h	h	b			u	s	t		l	y
r	o				b			u	s	t	d	f	g
r	o				b			u	s				y
h	j	h	h	j	h	j							
r								u	s	t		l	y
r	d	f	g	d	f	g							
	o				b			u	s	t		l	y
					b			u	s	g	h	l	y
r	d			f	b			u	s	t		l	y
r	o				b			u					
r	o				b			n	s	t		l	y
r	o				b			u	f	g		l	y
r	o				b			u	s	t			

Tabelle 5.4: *In der Tabelle sind alle Ergebnisse der Erkennung der Schriftzeichen von den Einzelbildern in Spalten gegliedert und gemeinsame Schriftzeichen teilen sich, so weit möglich, eine Spalte.*

bleibt an dieser Stelle die Spalte für das Wort leer und die weiteren wieder identischen Schriftzeichen können in dieselben Spalten eingetragen werden. Ähnlich zu diesen fehlenden Schriftzeichen können unterschiedliche Schriftzeichen in eine Spalte eingetragen werden, wenn dadurch nachfolgende Schriftzeichen wieder in den Spalten identisch sind. Sind alle Schriftzeichen grundsätzlich verschieden zu den Schriftzeichen in den anderen Wörtern, können die Spalten beliebig gewählt werden. Problematisch sind zu viele Schriftzeichen an einer Position zu den anderen Wörtern. In diesem Fall werden hier mehrere Schriftzeichen in eine Spalte als Kombination eingetragen. Da mehrere Zeichen an einer Stelle in der Tabelle als ein Element interpretiert werden, kann keine Aussage über eventuell gleiche Schriftzeichen innerhalb dieser Position und folgender Wörter an dieser Position getroffen werden. Hierfür bietet sich an, weitere Spalten in die Tabelle einzufügen, um jedes Schriftzeichen eines Wortes in eine Spalte zu überführen und mit den folgenden Wörtern zu vergleichen. In Tabelle 5.4 sind alle Schriftzeichen der einzelnen Wörter in Spalten zusätzlich getrennt. Für diese Tabelle lassen sich genauso wie in Tabelle 5.2 die Schriftzeichen in jeder Spalte zählen und das häufigst selektieren. Zusätzlich müssen hier die Leerstellen mit berücksichtigt werden. Sie können ebenfalls als Schriftzeichen interpretiert werden.

In jeder Spalte kann dann ein Zeichen mit der höchsten Häufig eit gefunden werden. Falls eine Leerstelle eine höhere Häufig eit als jedes Schriftzeichen aufweist, wird diese gewählt. Die so selektierten Schriftzeichen können in fester Reihenfolge zur Tabelle aneinandergehängt werden. Leerstellen werden dabei übersprungen. Auf diese Weise ergibt sich ein Wort, welches zugleich das Ergebnis ist. In diesem Fall ist das Ergebnis „robustly“. Dieses Beispiel zeigt, dass aus vielen einzelnen Wörtern, die jeweils mindestens einen

- 5	- 6	**- 14**	**- 13**	a 1	**b 13**	t 1	a 2	**u 14**	e 2	n 2	- **15**	**- 7**	- 6
r 11	**o 8**	g 1	a 1	n 1	g 1	**- 14**	**- 15**	- 2	- 4	**t 9**	d 1	**l 9**	**y 10**
h 1	j 1	h 1	h 2	**- 10**	h 1	j 1		n 1	**s 10**	- 4	h 1	f 1	g 1
	d 2	f 1	g 1	f 2	- 1	g 1			f 1	g 2			
				h 1	f 1								
				j 1									
				d 1									
r	o				b			u	s	t		l	y

Tabelle 5.5: *In dieser Tabelle sind die verschiedenen Schriftzeichen aus der Tabelle 5.4 mit ihrer entsprechenden Häufig eit dargestellt. In jeder Spalte ist das Schriftzeichen bzw. die Leerstelle mit dem häufigste Vorkommen in fetter Schrift hervorgehoben. Leerstellen sind hier durch ein „-“ gekennzeichnet. In der letzten Zeile ist dieses Schriftzeichen dann eingetragen. Im dem Fall, dass eine Leerstelle am häufigste war, ist die Zelle leer. Durch Aneinanderreihen alle Schriftzeichen aus der letzten Zeile ergibt sich ein Wort, welches zugleich das Ergebnis ist.*

Fehler beinhalten können, eine Aussage über alle getroffen werden kann, welche dann zu dem gewünschten Ergebnis führen. Das hier vorgeführte Beispiel zum Mitteln der Schriftzeichen über viele Einzelergebnisse der Texterkennung ist empirisch erzeugt. Daher können hier einige Fehler entstanden sein, die das Ergebnis verfälschen. Außerdem ist es möglich, dass einige Wörter nach den aufgestellten Regeln anders in die Tabellen eingebaut werden können bzw. dass es dafür viele verschiedene Möglichkeiten gibt. Das hier gezeigte Beispiel ist also nicht deterministisch. Das bedeutet, dass das Ergebnis sich bei einem erneuten Auswerten ändern kann. Damit das Ergebnis deterministisch und damit nachvollziehbar ist, muss ein fest definierte Verfahren entwickelt werden. Für dieses Verfahren können aber einige Sachverhalte aus diesem Beispiel festgehalten werden. Zum Einen ist die Position identischer Schriftzeichen in verschiedenen einzelnen Wörtern für die Positionsfindun in der Tabelle entscheidend. So ergeben sich bestimmte Positionen für Schriftzeichen in der Tabelle, die sich von Wort zu Wort wiederholen und eine Art Grundgerüst erzeugen. In Tabelle 5.4 sind zum Beispiel die Spalten 1 und 6 sehr konstant über alle Wörter. Zum Anderen sind verschiedene Operationen zum Erzeugen der Tabelle benutzt worden. Ist ein Schriftzeichen von einem Wort zum anderen identisch, dann wird es in dieselbe Spalte eingetragen. Diese Operation kann als Beibehalten bezeichnet werden. Ist an einer Position das Schriftzeichen unterschiedlich, wobei sich die Position aus den benachbarten Positionen der Schriftzeichen ergibt, dann wird das Schriftzeichen zwar in die gleiche Spalte eingetragen, aber ist dann

unterschiedlich zu dem verglichenen Schriftzeichen. Diese Operation ist dann als Ersetzen anzusehen. Befinde sich aber an dieser berechneten Position kein Schriftzeichen, dann muss in der Spalte eine Leerstelle eingetragen werden. Diese Operation steht dann für Löschen. Umgekehrt kann ein Schriftzeichen zuviel vorkommen. In diesem Fall muss in der Tabelle an dieser Position eine neue Spalte eingefügt werden, in der dann nur das eine Schriftzeichen enthalten ist. Diese Operation ist dann das Einfügen. So ergeben sich die vier Operationen Beibehalten, Ersetzen, Löschen und Hinzufügen. Auf Basis dieser Fakten kann nun ein Verfahren definier werden, welches deterministisch ist.

5.5 Zeitliche Filterung der Texterkennung

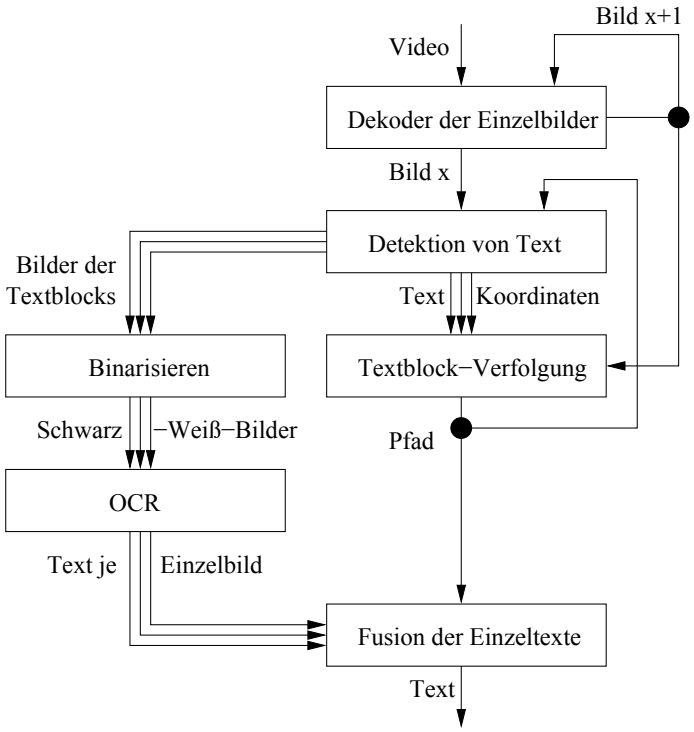


Abbildung 5.1: *Aus Abschnitt 5.4 ergibt sich eine neue Möglichkeit, die Fehler nach der Detektion und Erkennung von Text zu korrigieren. Anders als bei dem Mitteln der Bilder von einem zu erkennenden Text wird nun direkt auf den einzelnen Bildern für diesen Text je eine Erkennung von Text durchgeführt. So entstehen viel mögliche Ergebnisse von demselben Text. Durch die Verfolgung der Textblöcke können diese zueinander geordnet werden und durch Mitteln der Schriftzeichen zu einem einzigen Ergebnis mit hoher Trefferrate fusioniert werden.*

Das in Abschnitt 5.4 gefundene empirische Beispiel für die Mittelung der Schriftzeichen über Einzelergebnisse der Texterkennung für dasselbe Wort zeigt, dass aus vielen Wörtern, die jeweils Fehler beinhalten, ein korrektes Ergebnis gefunden werden kann. Ebenfalls wurden Eigenschaften für ein Verfahren zur Mittelung der Schriftzeichen gefunden. Zum Einen sind die Positionen der Schriftzeichen und zum Anderen die Operationen Beibehalten, Ersetzen, Löschen und Hinzufügen zum Vergleich eines Wortes mit einem oder mehreren anderen Wörtern entscheidend. Das System zur Detektion und Erkennung von Text muss dementsprechend wie in Abbildung 5.1 geändert werden. In dieser neuen Systemstruktur wird nun auf einzelnen statt auf mehreren fusionierten Bildern eine Texterkennung durchgeführt. Die Ergebnisse der Texterkennung werden dann über die Textblockverfolgung zueinander zugeordnet und in einem weiteren Schritt durch Mitteln der Schriftzeichen zu einem einzigen Ergebnis mit hoher Trefferrate fusioniert. Kern dieses neuen Verfahrens ist die Fusion der Einzeltexte. Da auf je in dem Video aufeinanderfolgenden Bild eine Texterkennung durchgeführt wird und diese Bilder zeitlich aufeinander folgen, kann dieses Fusion der Texte auch als zeitliche Filterung der Texterkennung angesehen werden. Das in Abschnitt 5.4 gefundene Beispiel zur Mittelung der Schriftzeichen zeigt einen möglichen Weg, ist aber nicht deterministisch. Daher muss ein Verfahren gefunden werden, welches eine deterministische Lösung zu diesem Problem erzeugt und die gefundenen Eigenschaften benutzt. Hierfür bietet sich als Basis des Verfahrens die Levenshtein Distanz [52] an.

5.5.1 Levenshtein Distanz

Die Levenshtein Distanz berechnet den Abstand von zwei Wörtern. Der Abstand besagt, mit welcher minimalen Anzahl an Schritten ein Wort in das andere umgewandelt werden kann. Hierfür stehen 4 Operationen zur Verfügung. Es kann ein Schriftzeichen beibehalten, entfernt, hinzugefügt oder ersetzt werden. Wenn ein Schriftzeichen beibehalten wird, dann bleibt die Distanz unverändert. Andernfalls wird die Distanz um 1 vergrößert. Dieses Verfahren wurde 1966 von V. Levenshtein in [52] veröffentlicht. Für die Erklärung des Verfahrens werden hier die Wörter u mit „abtauen“ und v mit „anbauen“ verglichen. Für die Berechnung der Levenshtein Distanz wird eine Matrix benötigt. Die Größe der Matrix ist abhängig von der Länge der Wörter. Die Länge des ersten Wortes mit 1 inkrementiert definier dabei die Länge der Matrix. Die Länge vom zweiten Wort mit 1 inkrementiert definier die Breite der Matrix. Die Länge von beiden Wörtern ist 7 und damit hat die Matrix eine Größe von 8x8.

Für die Berechnung muss die Matrix initialisiert werden. Die Zeilen und Spalten der Matrix werden von 0 an gezählt. Hierfür wird ganz oben links eine 0 eingetragen. In der Zeile 0 wird dann der Wert jedes weiteren Feldes von links nach rechts um 1 inkrementiert. Auf diese Weise wird von links nach rechts von 0 bis zu der Länge des ersten Wortes $m = 7$ gezählt. Auf die gleiche Weise wird ebenfalls die nullte Spalte von oben nach unten von 0 bis zur Länge des zweiten Wortes $n = 7$ gezählt. Für das Verständnis befinde sich die Wörter vertikal neben bzw. horizontal über der Matrix. Auf diese Weise besitzt jede Spalte und jede Zeile ein Schriftzeichen, welches diese Spalte oder Zeile charakterisiert. Eine Ausnahme sind die nullte Spalte und nullte Zeile. Die nullte Spalte und Zeile sind für das Verfahren die Basis und werden je mit einem ϵ gekennzeichnet. Die initialisierte Matrix mit den beiden zu vergleichenden Wörtern ist in Tabelle 5.6 dargestellt. Ist die Matrix auf diese Weise initialisiert, wird jedes weitere Feld zeilenweise von links nach rechts und von oben nach unten ermittelt. Dazu werden als erstes das charakteristische Schriftzeichen der Spalte und

	ϵ	a	b	t	a	u	e	n
ϵ	0	1	2	3	4	5	6	7
a	1							
n	2							
b	3							
a	4							
u	5							
e	6							
n	7							

Tabelle 5.6: *Die Levenshtein-Matrix hat die Länge und Breite, definier durch die Länge der vergleichenden Wörter, mit 1 inkrementiert. Die erste Zeile und die erste Spalte werden von links nach rechts bzw. von oben nach unten von 0 bis zur Länge des Wortes hochgezählt. Diese Zeile und Spalte sind für das Verfahren die Basis und werden je mit einem ϵ gekennzeichnet. Für das Verständnis befinde sich die Wörter vertikal neben bzw. horizontal über der Matrix.*

das charakteristische Schriftzeichen der Zeile an dieser Position verglichen. Sind die beiden Schriftzeichen identisch wird der Wert ein Feld weiter links oben in der Matrix übernommen. Sind die beiden Schriftzeichen nicht identisch wird das Minimum des Feldes links, oben bzw. links oben übernommen und um 1 inkrementiert. Das Verfahren ist in den Formeln 5.1 bis 5.6 dargestellt.

$$m = |u| \text{ (Länge von Wort 1)} \tag{5.1}$$

$$n = |v| \text{ (Länge von Wort 2)} \tag{5.2}$$

$$D_{0,0} = 0 \text{ (Matrix oben links auf 0 setzen)} \tag{5.3}$$

$$D_{i,0} = i, 1 \leq i \leq m \text{ (Zeile von 0 bis m durchzählen)} \tag{5.4}$$

$$D_{0,j} = j, 1 \leq j \leq n \text{ (Spalte von 0 bis n durchzählen)} \tag{5.5}$$

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} & +0 \text{ falls } u_i = v_j \\ D_{i-1,j-1} & +1 \text{ (Ersetzung)} \\ D_{i,j-1} & +1 \text{ (Einfügung)} \\ D_{i-1,j} & +1 \text{ (Löschung)} \end{cases}, 1 \leq i \leq m, 1 \leq j \leq n \tag{5.6}$$

Die so berechnete Matrix ist in Tabelle 5.7 dargestellt.

Es kann gesagt werden, dass jedes Element der Matrix aussagt, wie viele Operationen ausgeführt werden müssen, um das eine Worte mit Hilfe des anderen Wortes bis zu dieser Länge in der Spalte bzw. Zeile zu bilden. Die Operation Beibehalten wird hierbei als kein Aufwand betrachtet. Also wird in der ersten Spalte das Wort „anbauen“ gebildet und es sind noch kein Schriftzeichen von „abtauen“ bekannt. Demnach muss in jeder Zeile genau eine Operation durchgeführt werden, um ein weiteres Zeichen von dem zu bildenden Wort „anbauen“ zu erhalten. Insgesamt werden dafür 7 Operationen benötigt. In der nächsten Spalte

	ϵ	a	b	t	a	u	e	n
ϵ	0	1	2	3	4	5	6	7
a	1	0	1	2	3	4	5	6
n	2	1	1	2	3	4	5	5
b	3	2	1	2	3	4	5	6
a	4	3	2	2	2	3	4	5
u	5	4	3	3	3	2	3	4
e	6	5	4	4	4	3	2	3
n	7	6	5	5	5	4	3	2

Tabelle 5.7: *Die komplette Levenshtein-Matrix zeigt im Feld unten rechts den Abstand zu beiden Wörtern. In diesem Fall ist das Ergebnis 2.*

ist vom Wort „abtauen“ bereits das erste Schriftzeichen „a“ bekannt, welches für die Bildung von „anbauen“ genutzt werden kann. Auf diese Weise reduziert sich die Anzahl der Operationen zum Bilden des Wortes um 1 und damit werden 6 Operationen benötigt, um das eine Wort aus dem anderen zu bilden. Mit jeder weiteren Spalte kommt ein weiteres Schriftzeichen vom Wort „abtauen“ hinzu, was dann zur Bildung des Wortes „anbauen“ verwendet werden kann. Das Ergebnis in diesem Beispiel ist der Abstand von 2. Die Levenshtein Distanz finde in vielen täglichen Bereichen Anwendung. Die häufigste Anwendungen sind hier zum Beispiel Suchanfragen im Internet oder die Korrektur von Rechtschreibfehlern in Dokumenten. In beiden Fällen kann der Abstand eines fehlerhaftes Wortes zu einer Bibliothek bestimmt werden und durch ein Wort mit der geringsten Levenshtein Distanz ersetzt werden. Die Levenshtein Distanz besagt lediglich, wie viele Operationen minimal benötigt werden, um von dem einen Wort zum anderen zu gelangen. Hiefür stehen die Operationen Löschen, Einfügen oder Ersetzen zur Verfügung. Es ist aber nicht direkt ersichtlich, welche Operationen an welcher Stelle ausgeführt werden müssen. Diese Operationen sind erst indirekt durch das Analysieren der Matrix erkennbar.

5.5.2 Beweis für die Levenshtein Distanz

In der Publikation [52] wurde der Beweis für einen Algorithmus, welcher als Levenshtein Distanz bekannt ist, zum Löschen, Hinzufügen und Ersetzen von binären Zeichen in einem Code für eine Korrektur mit minimaler Schrittanzahl erbracht.

- Lemma 1

Jeder Algorithmus, welcher s Löschungen oder s Hinzufügungen korregiert, kann auch s Löschungen und Hinzufügungen korregieren.

- Beweis

Angenommen Wort z stammt von Wort x mit der Länge n mit i_1 Löschungen und j_1 Hinzufügungen mit $i_1 + j_1 \leq s$ und von Wort y mit der Länge n mit i_2 Löschungen und j_2 Hinzufügungen mit $i_2 + j_2 \leq s$ ab. Wort z kann mit maximal $max(i_2 + j_1, j_2 + i_1)$ Schritten

aus Wort x und Wort y gebildet werden. Da Wort x und Wort y die gleiche Länge besitzen gilt $j_1 - i_1 = j_2 - i_2$ und damit $i_2 + j_1 = j_2 + i_1 = \frac{1}{2}(i_1 + i_2 + j_1 + j_2) \leq s$.

B_n sei die Menge aller binären Wörter mit der Länge n. Für ein beliebiges Wort x aus B_n sei $|x|$ die Anzahl der Einser und $||x||$ die Anzahl doppelter Zeichen. Damit kann die Anzahl verschiedener Wörter $P_s(x)[Q_s(x)]$ durch s Löschungen oder s Hinzufügungen vorrausgesagt werden:

$$C^s_{||x||-s+1} \leq P_s(x) \leq C^s_{||x||+s-1'} \tag{5.7}$$

$$\sum_{i=0}^{s} C^i_n 2^{s-i} \leq Q_s(x) \leq \sum_{i=0}^{s} C^i_n C^i_s 2^{s-i} \tag{5.8}$$

- Lemma 2

$L_s(n)$ beschreibt die Anzahl der Wörter maximaler Länge in B_n, welche s Fehler korregieren kann. Für eine feste Anzahl an Korrekturen s und die Länge $n \to \infty$ soll gelten:

$$2^s(s!)^2 2^n/n^{2s} \lesssim L_s(n) \lesssim s!\ 2^n/n^s \tag{5.9}$$

- Beweis

Sei K ein maximaler Wort in B_n welches Hinzufügungen oder Löschungen korregieren kann und für ein zufälliges $k(1 \leq k \leq \frac{n}{2})$ sei $L_s(n) = L'_k + L''_k$, wobei L'_k die Anzahl der Wörter $x \in K$ ist, so dass $k < ||x|| < n - k$. Für K gilt $\sum_{x\in K} P_s(x) \leq 2^{n-s}$ und für das Maximum $\sum_{x\in K} R_{2s}(x) \geq 2^n$ wobei $R_{2S}(x)$ die Anzahl der Wörter mit $2s$ oder weniger von x ist. Aus Formel 5.7 und 5.8 folgt $2^{n-s} \geq L'_k C^s_{k-s}$ und $2^n \leq (L'_k C^s_{n-k+s} + L''_k C^s_{n+s-1}) \sum_{i=0}^{s} C^i_{n-1} C^i_s 2^{s-i}$. Aus diesen Ungleichungen lässt sich auf die Ungleichung 5.9 aus Lemma 2 folgern.

- Theorie 1

$$L_1(n) \sim 2^n/n \tag{5.10}$$

- Beweis

Aus Lemma 2 (5.9) ist eine Beweis ausreichend für:

$$L_1(n) \geq 2^n/(n+1) \tag{5.11}$$

Für eine Klasse von Wörtern $K^a_{n,m}$ sei jedes $K^a_{n,m}$ mit $(a = 0, 1, ..., m-1)$ definier als Gruppe von Codes $\sigma_1 ... \sigma_n$ in B_n so dass gilt $\sum_{i=1}^{n} \sigma_i i \equiv a(\mod m)$. Es muss gezeigt

werden das für $m \geq n+1$ jede Klasse $K_{n,m}^{a}$ eine Löschung korregieren kann. Als Folge einer Löschung kann geschlossen werden, dass ein Wort $x = \sigma_1 ... \sigma_n$ in $K_{n,m}^{a}$ in $x' = \sigma_1' ... \sigma_n'$ umgewandelt wurde. Daraus kann $|x'|$ und der kleinste positive Restbetrag aus $a - \sum_{I=1}^{n-1} \sigma_i' i \mod m$ bestimmt werden. Um ein Wort x wieder aus x' herzustellen ist es wichtig zu wissen, welche der Zeichentypen gelöscht wurde und wieviel Zeichen anderer Typen sich links von dieser Stelle befinden Aus der Definitio von $K_{n,m}^{a}$ und n_0 und n_1 folgt, dass wenn $m \geq n+1$ entweder $a' = |x'| + 1 + n_0$, wenn eine 1 gelöscht wurde, oder $a' = n_1$, wenn eine 0 gelöscht wurde, gilt. Als Folge, abhängig ob a' größer als —x'— ist oder nicht, kann geschlossen werden, welcher Zeichentyp gelöscht wurde. Aus dem Ergebnis von Lemma 1 kann jeder Code $K_{n,m}^{a}$ für $m \geq n+1$ eine Löschung oder Hinzufügung korregieren. Da jedes Wort in B_n zu den gleichen von m Codes $K_{n,m}^{a}(a = 0, 1, ..., m-1)$ gehört, muss einer dieser Codes nicht weniger als $2\frac{n}{m}$ Wörter beinhalten, was für $m = n+1$ 5.11 bestätigt.

Es wird angenommen, dass ein Code s Löschungen, Hinzufügungen und Ersetzungen korrigieren kann, wenn jedes binäre Word von nicht mehr als einem Wort in K mit s oder weniger Löschungen, Hinzufügungen und Ersetzungen erhalten werden kann. Es kann gezeigt werden, dass eine Funktion $r(x, y)$ definier durch ein Paar von binären Wörtern, gleich zu setzen mit der kleinesten Anzahl an Löschungen, Hinzufügungen und Ersetzungen, welches das Wort x in Wort y transformiert, eine Metrik ist. Außerdem kann gezeigt werden, dass ein Code K s Löschungen, Hinzufügungen und Ersetzungen korregieren kann, wenn $r(x, y) > 2s$ für jedes verschiedene Wort x ist. y in K. $M_s(n)$ bestimmt dabei die Menge der maximalen Codes in B_n, welche s Löschungen, Hinzufügungen und Ersetzungen korregieren kann.

- Theorie 2

$$2^{n-1}/n \leq M_1(n) \leq 2^n/(n+1) \tag{5.12}$$

- Beweis

Die obere Abgrenzung ist die Hamming Abschätzung für Codes, welche eine Ersetzung korrigieren kann. Um die untere Abgrenzung zu beweisen reicht es aus, dass alle Codes aus $K_{n,m}^{a}$, definier durch den Beweis aus Theorie 1, in der Lage sind eine Löschungen, Hinzufügungen und Ersetzungen zu korrigieren, falls $m \geq 2n$ ist. Es ist noch zu Beweisen, dass nur noch eine Ersetzung nötig ist zum Ändern eines Wortes $sigma_1 ... \sigma_n$ in $K_{n,m}^{a}$ in ein Wort $sigma_1' ... \sigma_n'$. Dabei gilt, dass das kleinste der nicht negativen Residuum von $a - \sum_{i=1}^{n} \sigma_i' i$ und $\sum_{i=1}^{n} \sigma_i' i - a \; mod \; 2n$ größer oder gleich zu j ist, wobei j der Index von den umgedrehten Symbolen ist. Unter Nutzung der selben Methode zum Beweis von Lemma 2 kann gezeigt werden, dass für eine feste Anzahl s und $n \to \infty$ gilt:

$$\left(\frac{(2s)!}{\sum_{i=0}^{s} 2^{-i} C_{2s}^{2i} C_{2i}^{i}} \right) \frac{2^n}{n^{2s}} \lesssim M_s(n) \lesssim ! \frac{s^n}{n^s} \tag{5.13}$$

5.5.3 Wegberechnung durch die Levenshtein Matrix

Da die Levenshtein Distanz keine Aussage über die nötigen Operationen zum Umformen eines Wortes in ein anderes Wort gibt, muss ein Verfahren gefunden werden, welche die

nötigen Operationen ermittelt. Prinzipiell ist es möglich, dass es mehrere verschiedene Umformungswege gibt, die alle die minimale Operationsanzahl erfüllen. In dem Beispiel der Wörter „abtauen“ und „anbauen“ ist die Levenshtein Distanz 2. Um von dem ersten Wort mit den Operationen Einfügen, Entfernen oder Ersetzen zu dem zweiten Wort in zwei Operationsschritten zu gelangen sind zwei Wege möglich. In Tabelle 5.8 und 5.9 sind die beiden Wege mit den nötigen Operationen aufgelistet.

Wort	Operation
abtauen	
abauen	Operation 1 (Entferne „t“ an Position 3)
anbauen	Operation 2 (Füge an Postion 2 „n“ hinzu)

Tabelle 5.8: *Das Wort 1 wird durch die Operationen Entfernen und Einfügen in Wort 2 umgewandelt.*

Wort	Operation
abtauen	
abbauen	Operation 1 (Ersetze „t“ Postion 3 durch „b“)
anbauen	Operation 2 (Ersetze „b“ Postion 2 durch „n“)

Tabelle 5.9: *Das Wort 1 wird durch die Operationen Ersetzen und Ersetzen in Wort 2 umgewandelt.*

Da es offensichtlich mehrere Lösungswege geben kann, reicht es nicht, nach einer Lösung zu suchen. Es muss sichergestellt werden, dass alle eventuellen Umformungswege ermittelt werden. Hierfür kann die berechnete Levenshtein Matrix genutzt werden. Ausgangspunkt für die Berechnung der möglichen Wege ist das Ergebnis der Levenshtein Distanz unten rechts in der Matrix. Ausgehend von diesem Element in der Matrix wird nun ein Weg nach ganz oben links gesucht, wobei immer nur nach oben, links oder links oben gelaufen werden kann. Die Richtungen stehen für je einen Schritt bei der Umformung von einem Wort zum anderen. Nach oben steht für das Einfügen, nach links für das Entfernen und nach links oben für das Ersetzen bzw. das Beibehalten eines Schriftzeichens. Ein Schriftzeichen wird dann ersetzt, wenn der Wert der aktuellen Position größer ist als der Wert oben links von der aktuellen Position. Sind beide Werte identisch, wird das Schriftzeichen beibehalten und bei der Umformung des einen Wortes in das andere ist an dieser Position keine Operation notwendig.

Für die Auswahl jede dieser Richtungen sind die Werte der aktuellen Position mit der nach oben, links bzw. oben links zu vergleichen. In Tabelle 5.10 ist die aktuelle Position $d0$. Von dieser Position können nur die Positionen $d1$, $d2$ und $d3$ erreicht werden. Prinzipiell gilt, dass immer die kleinsten Werte gewählt werden. Dazu muss als erstes das Minimum der drei Werte $d1$, $d2$ und $d3$ ermittelt werden. Hierbei ist es möglich, dass alle drei Werte gleichzeitig das Minimum sind und damit drei Operationen an dieser Position möglich sind. Daher muss jede Richtung markiert werden. Von dieser neuen Position bzw. diesen neuen Positionen muss dann nach dem gleichen Schema erneut ermittelt werden, welche Werte nach oben, links bzw. oben links möglich sind. Dieses Verfahren wird angewendet, bis man ganz oben links in der

...	...	...	...
...	$d3$	$d2$	...
		$\nwarrow$ $\uparrow$	
...	$d1$	$\leftarrow$ $d0$	...
...	...	...	...

Tabelle 5.10: *Für die Wegberechnung wird immer nur ein kleiner Teil einer Matrix betrachtet. Ausgehend von $d0$ kann nur noch nach oben $d2$, nach links $d1$ oder nach links oben $d3$ gegangen werden.*

Matrix angekommen ist. Die so markierten Richtungen ergeben dann alle möglichen Wege durch die Matrix mit den dazugehörigen Operationen, die nötig sind, das eine Wort aus dem anderen zu bilden. Es ist immer mindestens ein Weg vorhanden.

	ϵ	a	b	t	a	u	e	n
ϵ	0	1	2	3	4	5	6	7
a	1	$\nwarrow$ 0	1	2	3	4	5	6
n	2	$\uparrow$ 1	$\nwarrow$ 1	2	3	4	5	5
b	3	2	$\nwarrow$ 1	$\nwarrow$ $\leftarrow$ 2	3	4	5	6
a	4	3	2	2	$\nwarrow$ 2	3	4	5
u	5	4	3	3	3	$\nwarrow$ 2	3	4
e	6	5	4	4	4	3	$\nwarrow$ 2	3
n	7	6	5	5	5	4	3	$\nwarrow$ 2

Tabelle 5.11: *In der Levenshtein-Matrix ist der Weg zur Erzeugung des einen Wortes durch das andere markiert. In diesem Fall sind zwei Wege möglich.*

In Tabelle 5.11 wurden die möglichen Wege durch die Levenshtein Matrix für die beiden Wörter „abtauen“ und „anbauen“ gekennzeichnet. Der Start befinde sich unten rechts. Im Vergleich der Werte nach links, oben und links oben ist der minimale Wert 2. Damit kann nur die Richtung nach links oben gewählt werden. An der Position in der Spalte „t“ und Zeile „b“ ist der Wert 2. Im Vergleich der möglichen neuen Positionen sind die Werte links und links oben mit 1 minimal und beide Richtungen müssen deshalb markiert werden. An dieser Stelle sind also zwei verschiedene Wege durch die Matrix möglich. Diese beiden Wege enden oben

links in der Matrix.

Für die Markierungen der Richtungen wird in dieser Arbeit eine weitere Matrix erzeugt, die die identische Größe der Levenshtein Matrix besitzt. Alle Werte dieser Matrix werden am Anfang auf 0 gesetzt. Wird in dem oben beschriebenen Verfahren eine Richtung gefunden, wird in der neuen Matrix ein neuer Wert an dieser Stelle eingetragen. Da mehrere Wege durch die Matrix über eine bestimmte Stelle in der Matrix laufen können, müssen die Richtungen nachvollziehbar sein. Es reicht also nicht, immer nur den Wert 1 an jede Position in der Matrix zu schreiben, über den der Weg läuft. Damit die unterschiedlichen Richtungen identifizier werden können bieten sich hier Potenzen von der Zahl 2 an. Die unterschiedlichen Werte können alle miteinander addiert werden und danach ist immer noch ersichtlich, von welchen Richtungen diese Position in der Matrix erreicht wurde.

Richtung	Operation	Potenz von 2	Wert	Binär-Wert
kein Weg	-	0	0	0000
oben links	beibehalten	2^0	1	0001
oben links	ersetzen	2^1	2	0010
links	löschen	2^2	4	0100
oben	hinzufügen	2^3	8	1000

Tabelle 5.12: *Jede Operation ist ein Wert entsprechend einer Potenz von 2 zugeordnet. Werden diese Zahlen miteinander addiert, ist immer noch ersichtlich, welche Operation ausgeführt wurde.*

In Tabelle 5.12 ist jeder möglichen Operation ein Wert zugeordnet, welcher einer Potenz von 2 entspricht. Werden diese Zahlen miteinander addiert, ist immer noch ersichtlich welche Operation ausgeführt wurde. Auf diese Weise ist es innerhalb einer Matrix möglich mehrere Wege zu speichern, die sich überlagern können. Jeder Weg beginnt unten rechts und endet oben links. Der Wert unten rechts wird am Anfang auf den Wert 1 gesetzt.

In der Tabelle 5.13 ist für die Levenshtein Matrix der beiden Wörter „abtauen“ und „anbauen“ eine weitere Matrix erzeugt worden, in der alle möglichen Wege eingetragen wurden. Die Pfeile zeigen an, über welche Positionen in der Matrix gelaufen werden kann. Der Wert an der Position, wohin der Pfeil zeigt, gibt Aufschluss über die Operation die ausgeführt wurde. In dem Beispiel sind zwei verschiedene Wege möglich, wie sie schon in der Tabelle 5.8 und 5.9 vorausgesagt wurden.

Für dieses Beispiel mit den Wörtern „abtauen“ und „anbauen“ wurde bisher die Umformung vom ersten Wort in das zweite interpretiert. Durch die Levenshtein Matrix und der hier beschriebenen neu erstellen Matrix zur Kennzeichnung des Weges kann aber auch direkt abgelesen werden, wie sich das zweite Wort zu dem ersten umformen lässt. Hierfür müssen lediglich die Operationen Löschen und Hinzufügen vertauscht werden. Es ist auch möglich, die komplette Matrix einschließlich der beiden Wörter zur Beschreibung der Spalten und Zeilen diagonal von links oben nach rechts unten zu spiegeln und danach auf gleiche Weise zu interpretieren.

	ϵ	a	b	t	a	u	e	n
ϵ	1	0	0	0	0	0	0	0
a	0	↖ 10	0	0	0	0	0	0
n	0	↑ 1	↖ 2	0	0	0	0	0
b	0	0	↖ 4	↖ ← 1	0	0	0	0
a	0	0	0	0	↖ 1	0	0	0
u	0	0	0	0	0	↖ 1	0	0
e	0	0	0	0	0	0	↖ 1	0
n	0	0	0	0	0	0	0	↖ 1

Tabelle 5.13: *In der neuen Matrix, ausgehend von der Levenshtein Matrix, ist der Weg zur Erzeugung des einen Wortes durch das andere markiert. In diesem Fall sind zwei Wege möglich.*

5.5.4 Die Datenstruktur

Durch die Levenshtein Distanz und die Analyse der Matrix für die Levenshtein Distanz ist nun eine deterministischste Aussage über die Position der Schriftzeichen und den 4 Operationen Beibehalten, Ersetzen, Löschen und Hinzufügen zum Vergleich von zwei Wörtern möglich. Es ist ebenfalls mit Hilfe der 4 Operationen deterministisch möglich zu beschreiben, auf welchen Wegen ein Wort aus dem anderen gebildet werden kann. An dieser Stelle muss die Möglichkeit geschaffen werden mehr als nur 2 Wörter miteinander zu vergleichen. Hierfür werden ähnlich wie zu der in Abschnitt 5.4 benutzen Tabelle 5.4, eine Datenstruktur benötigt, die diese Anforderungen erfüllt.

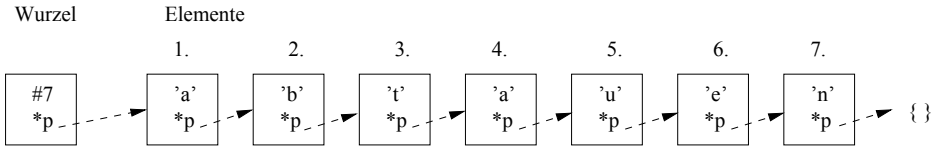


Abbildung 5.2: *Eine Liste besteht aus einer Wurzel und beliebig vielen Elementen. In diesem Beispiel sind es 7 Elemente. Die Wurzel kennt die Anzahl der Elemente und zeigt mit *p auf das erste Element. Jedes Element beinhaltet ein Schriftzeichen und ein Zeiger *p auf ein weiteres Element. Das letzte Element zeigt auf eine leere Menge* { }.

Die einfachste Datenstruktur für eine Tabelle mit Spalten und Zeilen ist eine Matrix. Eine Matrix hat bei der Implementierung in einer Programmiersprache immer eine feste Größe. Die Größe muss also vor der Initialisierung der Matrix bekannt sein. Für die Rekonstruktion von einem Wort aus vielen einzelnen Ergebnissen der Texterkennung ergeben sich bei der Matrix zwei Probleme. Zum einen ist die Länge des Wortes unklar und zum anderen ist die Anzahl der Ergebnisse der Texterkennung beliebig groß. Das bedeutet, dass bei jedem neuen Schriftzeichen eine neue Spalte und bei jedem neuen Ergebnis für die Rekonstruktion eine Zeile in der Matrix hinzu kommt. Wenn das passiert, muss eine neue Matrix erzeugt werden und die alte Matrix mit der neuen Spalte bzw. Zeile hinein kopiert werden. Dieser Vorgang wird im Laufe von vielen Ergebnissen der Texterkennung immer aufwändiger und verlangsamt das Verfahren erheblich.

Eine Alternative bieten hier Listen. In der Programmiersprache C++ können einzelne Elemente definier werden, die auf ein weiteres Element verweisen und damit eine Liste erzeugen. Der Verweis auf ein weiteres Element aus einem Element wird als Zeiger bezeichnet. Jedes Element in diesen Listen ist unabhängig von allen anderen Elementen. Dadurch können Elemente dynamisch, ohne das alle anderen Elemente kopiert werden müssen, in die Liste hinzugefügt oder sogar entfernt werden. Es müssen lediglich die Zeiger der alten angrenzenden Elemente verlegt werden. Für die Verwaltung der Liste ist eine Wurzel notwendig. Die Wurzel beinhaltet die Information über die Anzahl der Elemente und zeigt außerdem auf das erste Element. Eine solche Datenstruktur heißt einfach verkettete Liste. Wenn ein Wort in einer solchen Liste gespeichert werden soll, werden alle Schriftzeichen in dem Wort zu einem Element. Das erste Schriftzeichen von dem Wort ist zugleich der Start von der Liste. Dieses Element zeigt dann auf das zweite Element, welches das zweite Schriftzeichen enthält. So bilden die Elemente eine Kette bis zum letzten Schriftzeichen in dem Wort. Das letzte Element besitzt zwar auch einen Zeiger, der aber auf eine leere Menge zeigt. Es ist auch möglich, dass jedes Element zusätzlich auf das Vorgängerelement zeigt. Eine solche Liste heißt doppelt verkette Liste. Für die hier notwendige Datenstruktur wird lediglich eine einfach verkettete Liste benötigt. Mit den Listen können also Wörter mit beliebiger Länge gespeichert werden. Diese Datenstruktur ist nur eindimensional und reicht damit nicht für das Speichern einer Tabelle. Für eine zweidimensionale Datenstruktur müssen also noch Erweiterungen gefunden werden. Die Lösung hierfür ist, dass jedes Element in der Liste wiederum eine Wurzel für je eine weitere Liste ist. In Abbildung 5.3 ist auf diese Weise die Tabelle 5.5 gespeichert. Die waagerechte Liste ist hierbei die übergeordnete Liste und die senkrechten Listen sind die untergeordneten Listen. Ein großer Vorteil von Listen bzw. Listen von Listen besteht darin, dass leere Elemente weggelassen werden können. In diesem Fall besitzt die Tabelle 98 Zellen. Die Liste von Listen hat dagegen nur 53 Elemente. Es konnte also fast die Hälfte mit 45 Elementen gespart werden. Die Einsparung der Elemente wird durch die zusätzlich eingeführten Zeiger erkauft. In großen Datenstrukturen kann die Speicherersparnis der Zellen deutlich größer werden als der Speicheraufwand der Zeiger und so entsteht ein Gewinn. Durch die Nutzung von Listen ist es nun möglich auf schnelle Weise die Schriftzeichen eines neuen Wortes in die Datenstruktur einzugliedern. Es können für neue Positionen neue Spalten in die übergeordnete Liste eingefügt, neue Elemente in die untergeordneten Listen angehängt oder einfach die Anzahl eines Elements in den untergeordneten Listen erhöht werden.

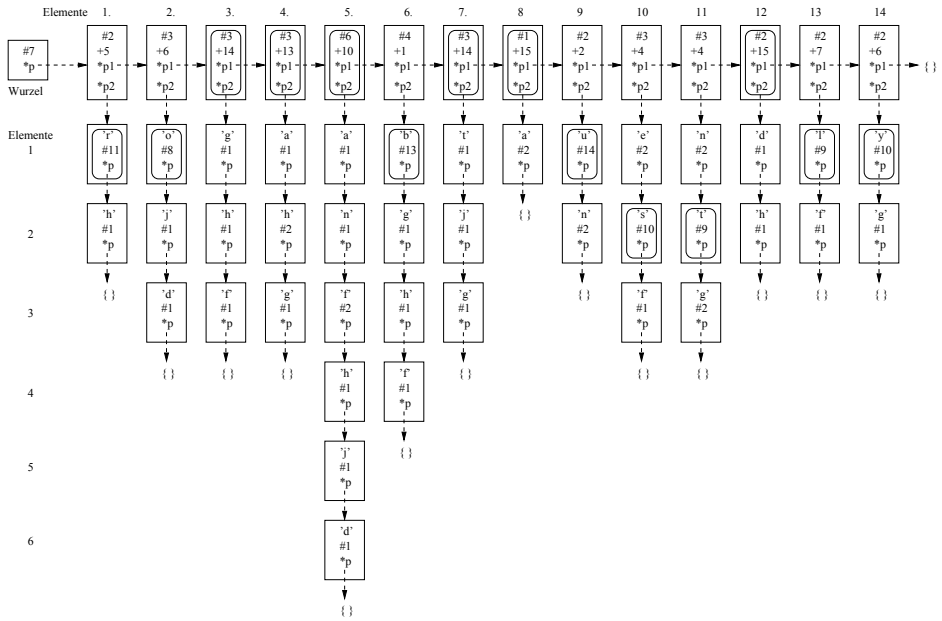


Abbildung 5.3: *Im Vergleich zu einer einfachen Liste sind hier die Elemente der Liste wiederum eine Wurzel zu je einer weiteren Liste. Jedes dieser Elemente hat wieder einen Zeiger p1 auf das nachfolgende Element. Zusätzlich müssen nun auch die Informationen über die Liste, auf welche das Element zeigt, gesichert werden. Hierfür wird der Zeiger p2 auf die neue Liste und die Anzahl, mit # vorangestellt, der Elemente in der Liste benötigt. Statt dem Schriftzeichen bei den Elementen der einfachen Liste wird in jedem Element die Anzahl der Leerstellen mit + vorangestellt gespeichert. Da die Leerstellen in der Tabelle nicht gleichzusetzen sind mit einem Schriftzeichen, müssen diese in der Wurzel vermerkt werden. Die untergeordneten Listen sind dann wieder einfach Listen. Jedes Element in den untergeordneten Listen besitzt wieder ein Schriftzeichen und einen Zeiger *p auf das nächste Element. Zusätzlich wird noch eine Anzahl für das Schriftzeichen gespeichert. In der Abbildung ist so die Tabelle 5.5 in Form von einer Liste von Listen gespeichert. Das häufigst Element bzw. die Leerstelle ist in jeder untergeordneten Liste durch ein abgerundetes Rechteck hervorgehoben. Werden diese hervorgehobenen Elemente hintereinander geschrieben, ergibt sich wieder das Wort „robustly“. Die Leerstellen werden dabei eliminiert.*

5.5.5 Integrieren von Wörtern in die Datenstruktur

Einer der entscheidensten Punkte in der zeitlichen Filterung der Texterkennung ist die Zuordnung der Schriftzeichen zu den Positionen in den anderen Ergebnissen der Texterkennung. Durch die Wegberechnung durch die Levenshtein Distanz ist es möglich, die Schriftzeichen

eines Wortes zu einem anderen Wort einzuordnen. Hierfür sind 4 Operationen möglich: Beibehalten, Ersetzen, Löschen und Hinzufügen. Die zuvor in Abschnitt 5.5.4 definiert Datenstruktur mit Listen von Listen dient hier als Speicher für die Ergebnisse. In diese können beliebig viele Ergebnisse der Texterkennung eingegliedert werden. An dieser Stelle ist es wichtig, jedes gefundene Wort durch die Texterkennung nicht mit einem anderen gefunden Wort der Texterkennung zu vergleichen, sondern diese Wörter mit der Datenstruktur zu vergleichen. Das erste Wort kann einfach in die Datenstruktur übertragen werden. Das zweite Wort kann dann einfach durch die Levenshtein Matrix eingegliedert werden. Bei jedem weiteren Wort wird es nun schwerer, da die Datenstruktur nicht wie notwendig nur aus einem Wort besteht, sondern aus vielen möglichen Schriftzeichen an den bestimmten Positionen. So kann keine Levenshtein Matrix erzeugt werden. Also muss ein repräsentatives Wort aus der Datenstruktur für jeden weiteren Vergleich mittels der Levenshtein Matrix gebildet werden.

Auswahl eines repräsentativen Wortes für die Datenstruktur

Für den Vergleich von einem neuen Wort mit der Datenstruktur muss ein repräsentatives Wort aus der Datenstruktur gefunden werden. In der Datenstruktur kann es an bestimmten Positionen bisher zu einer hohen Wahrscheinlichkeit einer Leerstelle gekommen sein, die aber mit weiteren Wörtern widerlegt wird. Da ein neues Wort beliebig verschieden zu den Daten in der Datenstruktur sein kann und damit Leerstellen gefüllt werden können, muss jede mögliche Position in der Datenstruktur für die spätere Einordnung berücksichtigt werden. Daher muss für jede Stelle ein repräsentatives Schriftzeichen ermittelt werden, egal ob an dieser Stelle die Wahrscheinlichkeit für eine Leerstelle spricht oder nicht. Wenn für eine Position die Leerstelle die höchste Wahrscheinlichkeit besitzt, wird anstelle der Leerstelle das nächste wahrscheinlichste Schriftzeichen gewählt. Sind mehrere Schriftzeichen mit der gleichen Wahrscheinlichkeit vorhanden, muss eines davon ausgewählt werden. In diesem Fall wird das erste aus der Liste gewählt. Umso mehr Wörter in die Datenstruktur integriert werden, umso unwahrscheinlicher werden gleiche Mengen von verschiedenen Schriftzeichen an derselben Position. Auf diese Weise wird an jeder Stelle ein Schriftzeichen nach einer Weile dominieren. Für das Beispiel in Abbildung 5.3 ergibt sich dann das Wort „roghfbtaustdly“. Eine solche Aussage für die Datenstruktur kann nach jedem in die Datenstruktur integrierten Wort getroffen werden. Mit diesem Resultat ist es möglich, mit einem beliebigen weiteren Wort eine Levenshtein Matrix und damit die Wegmatrix zum Umformen in das andere Wort zu erzeugen. Mit dem Auswerten der 4 Operationen Beibehalten, Ersetzen, Hinzufügen und Löschen kann dieses Wort in die Datenstruktur eingegliedert werden.

Auswahl der nötigen Operationen zum Eingliedern in die Datenstruktur

Nachdem durch das neue Wort und das repräsentative Wort eine Levenshtein Matrix und damit ebenfalls eine Matrix für die Wegfindun durch die Levenshtein Matrix erzeugt wurde, müssen nun anhand der Wegmatrix die Schriftzeichen des neuen Wortes mit Hilfe der 4 zur Verfügung stehenden Operation in die Datenstruktur eingegliedert werden. Die erzeugte Wegmatrix gibt die nötige Operation und die Position in der Datenstruktur an. Für die Erklärung des Verfahrens dient wieder das Beispiel der beiden Wörter „abtauen“ und „anbauen“ für eine Levenshtein Matrix 5.7 und deren Weg 5.13. Das Word „abtauen“ ist hier das repräsentative Word für die Datenstruktur und das Wort „anbauen“ ist das neu einzugliedernde Wort. Also

wird das senkrechte Wort mit Hilfe des waagerechten Wortes und der Nutzung der Levenshtein Distanz in die Datenstruktur eingegliedert. Jede Spalte in der Matrix spiegelt hierbei die Position in der Datenstruktur wieder. In der Wegmatrix können mehrere Wege parallel laufen und jeder führt zu einer anderen Art der Eingliederung der neuen Schriftzeichen. Für eine korrekte und deterministische Aussage müssen also alle Wege in die Datenstruktur eingetragen werden. Mit jedem parallelen Weg kommt es zu einer zusätzlichen möglichen Lösung an den entsprechenden Positionen. Sind zum Beispiel 2 Wege zum Eingliedern möglich ist die Chance, den richtigen zu wählen, 50%. Werden alle möglichen Lösungen an den entsprechenden Positionen eingegliedert, dann ergibt sich im Vergleich zu eingegliederten Wörtern mit nur einem Weg ein Ungleichgewicht der Wahrscheinlichkeit, da dort jedes Schriftzeichen zu 100% zutrifft. Das bedeutet, dass im Falle von zwei Wegen die neuen Schriftzeichen nur halb gewertet werden können. Bei 3 Wegen dann nur um 1/3 und so fortlaufend. Andersrum gesehen wird jede Position durch das Eingliedern des neuen Wortes trotz mehrerer Wege nur um 1 erhöht. So hat jedes Wort für jede Position die gleiche Menge an möglichen Lösungen hinzugefügt. Auch in dem Fall von „abtauen“ und „anbauen“ existieren mehrere Wege zum Eingliedern in die Datenstruktur. Um die Anzahl der möglichen parallelen Wege an einer Position zu ermitteln, müssen die Wege je Spalte gezählt werden. Ist die Anzahl der Wege an den entsprechenden Positionen bekannt, können die entsprechenden Operationen an den Positionen ausgewertet werden. In der Wegmatrix 5.13 sind hierfür zusätzlich zu den Zahlen Pfeile eingetragen, die das Interpretieren erleichtern. Ausgangspunkt für die Auswahl einer Operation ist immer eine Zahl größer 0 in der Wegmatrix.

Beibehalten: Unten rechts in der Wegmatrix steht eine 1. In dieser Spalte befinde sich von dem repräsentativen Wort der Datenstruktur ein „n“. In der Zeile für das neu einzugliedernde Wort befinde sich ebenfalls ein „n“. Die nötigen Operationen zum Eingliedern in die Datenstruktur können aus den zu erreichenden Punkten in der Wegmatrix abgelesen werden. In dem Fall ist nur ein Weg möglich und es kann nur nach oben links gegangen werden, wo ebenfalls eine 1 steht. Die Zahl, auf die der Pfeil zeigt, gibt an, welche Operation benötigt wird. Hier ist es die Operation Beibehalten. Wird ein Schriftzeichen beibehalten, wird das Element mit dem Schriftzeichen „n“ in der entsprechende Spalte um den Wert 1 erhöht.

Ersetzen: In der Spalte mit dem Schriftzeichen „t“ und der Zeile mit dem „b“ steht als Ausgangspunkt eine 1. Von diesem Punkt aus sind zwei Wege möglich. es können die Zahlen 2 und 4 erreicht werden. Die Zahl 2 steht für die Operation Ersetzen. Das bedeutet, dass das Schriftzeichen „t“ aus dem repräsentativen Wort der Datenstruktur durch das Schriftzeichen „b“ ersetzt wird. Das „b“ wird als Element in die Datenstruktur der entsprechenden Spalte von dem Schriftzeichen „t“ eingefügt. Unter Umständen kann in dieser Spalte in der Datenstruktur bereits ein Element mit dem Schriftzeichen „b“ von vorangegangenen Eingliederungen vorhanden sein. Ist es bereits vorhanden, wird die Anzahl um 1, dividiert durch die Anzahl der möglichen Wege, in der Spalte in dem Element „b“ erhöht. Ist es noch nicht vorhanden, wird ein neues Element mit dem Schriftzeichen „b“ in die Spalte eingefügt und der Wert 1 dividiert durch die Anzahl der möglichen Wege in der Spalte festgelegt. In diesem Fall wird der Wert 0,5 gespeichert.

Löschen: In der Spalte mit dem Schriftzeichen „t“ und der Zeile mit dem „b“ steht als Ausgangspunkt eine 1. Von diesem Punkt aus sind zwei Wege möglich. Es können die Zahlen 2 und 4 erreicht werden. Die Zahl 4 steht für die Operation Löschen. Das bedeutet, dass das Schriftzeichen „t“ aus dem repräsentativen Wort der Datenstruktur gelöscht wird. Hierfür muss nun in der Wurzel für die Spalte die Leerstelle um den Wert 1 dividiert durch die Anzahl

der möglichen Wege in der Spalte erhöht werden. In diesem Fall wird der Wert 0,5 auf die Anzahl der Leerstellen addiert.

Hinzufügen: In der Spalte mit dem ersten „a“ und in der Zeile mit dem ersten „b“ steht als Ausgangspunkt eine 1. Von diesem Punkt aus ist nur die Zahl 10 erreichbar. Die Zahl 10 kann in die Zweierpotenten 8 und 2 zerlegt werden. Die Zahl 8 steht für die Operation Hinzufügen. Das bedeutet in diesem Fall, dass das Schriftzeichen „b“ aus dem neu einzugliedernden Wort der Datenstruktur in eine neue Spalte gespeichert werden muss. In der Datenstruktur gab es für diese Position noch keine Schriftzeichen. Die neue Spalte muss als Anzahl der Leerstellen die Anzahl der bereits integrierten Wörter addiert um den Wert 1, dividiert durch die Anzahl der möglichen Wege in der Spalte der Wegmatrix festgelegt werden. Auch hier waren 2 Wege möglich und damit wird auf die Anzahl der integrierten Wörter der Wert 0,5 addiert. In der Spalte muss dann das neue Element mit dem Schriftzeichen „b“ eingetragen werden. Das neue Element erhält ebenfalls den Wert 1 dividiert durch die Anzahl der möglichen Wege in der Spalte. In diesem Fall wird der Wert 0,5 gespeichert.

In Abbildung 5.4 ist mit Hilfe der Levenshtein Matrix, der Wegmatrix und den 4 Operatoren die Datenstruktur für die Ergebnisse der Texterkennung aus Tabelle 5.1 automatisch gebildet worden. Gegenüber der per Hand gefundenen Datenstruktur in Abbildung 5.3 sind bei der automatisch gefundenen Struktur deutlich weniger Spalten nötig. Statt 14 Spalten werden hier nur noch 10 Spalten benötigt. Die Ursache hierfür liegt eventuell darin, dass in dem Versuch manuell eine Lösung zu finde möglicherweise Kombinationsmöglichkeiten übersehen wurden.

Auswahl der wahrscheinlichsten Schriftzeichen

Für ein Ergebnis muss aus jeder Spalte das Schriftzeichen mit der höchsten Wahrscheinlichkeit gewählt werden. Ist die Anzahl der Leerstellen größer als alle möglichen Schriftzeichen, dann wird diese Position gelöscht und kein Schriftzeichen gewählt. Auf diese Weise ergibt sich aus der Datenstruktur in Abbildung 5.3 wieder das Wort „robustly“. Durch das automatische Verfahren zum Eingliedern von Wörtern in die Datenstruktur ist es nun möglich, beliebig viele Wörter miteinander deterministisch zu analysieren. Nach jedem weiteren Wort, welches in die Datenstruktur integriert wurde, kann ein Ergebnis mit der höchsten Wahrscheinlichkeit für alle Schriftzeichen ausgegeben werden. Hierbei ist die Länge der integrierten Wörter insgesamt und untereinander beliebig.

Komplexität der zeitlichen Filterung

Die Berechnung der Levenshtein Distanz ist mit zwei in sich verschachtelten Schleifen nahezu quadratisch mit $O(mn)$ mit der Spaltenanzahl m und der Zeilenanzahl n. Für die zeitliche Filterung wird je zur Verfügung stehendem Wort l eine Levenshtein Distanz berechnet und in die Datenstruktur eingetragen. Damit ist der Aufwand für des gesamte Verfahren $O(mnl)$.

5.5.6 Zuverlässigkeit der zeitlichen Filterung der Texterkennung

Der in Abschnitt 5.5 vorgestellten Methode zur zeitlichen Filterung der Texterkennung sind Grenzen gesetzt. Diese Grenze muss getestet werden. Genauso ist die Zuverlässigkeit zu prüfen. Im Folgenden wird deshalb dieses Verfahren auf seine Zuverlässigkeit und deren Grenzen untersucht. Um eine Aussage hierfür zu finde werden Messungen mit sich

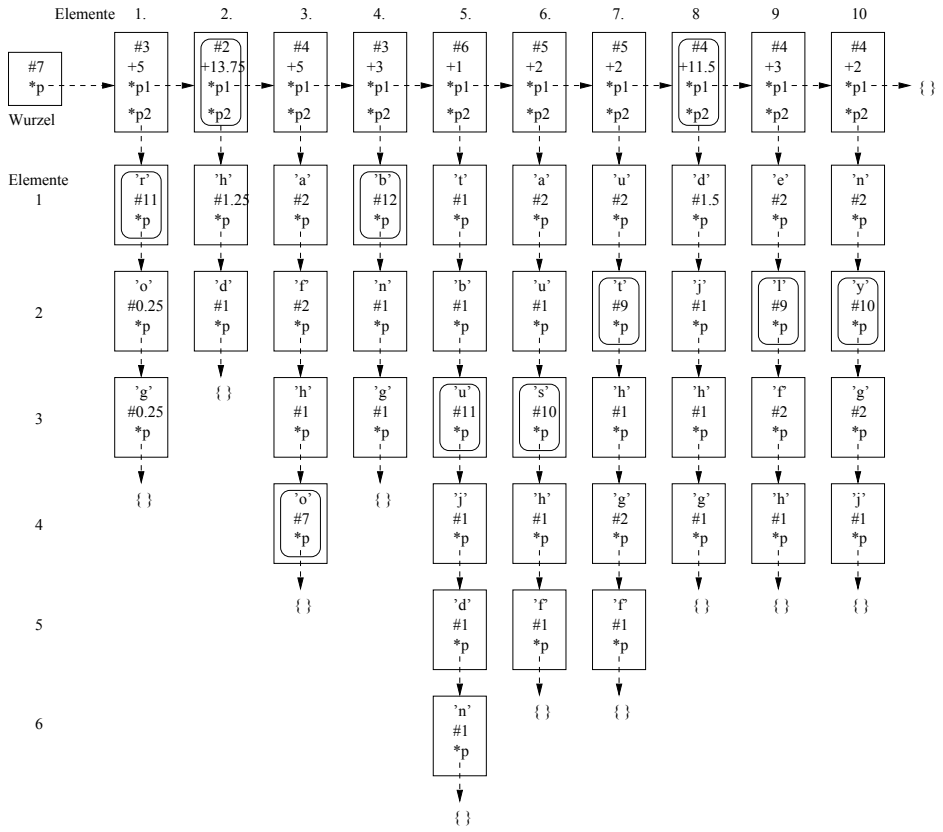


Abbildung 5.4: *Durch die Levenshtein Matrix, die Wegmatrix und den 4 Operatoren wurden alle Ergebnisse der Texterkennung aus der Tabelle 5.1 automatisch in die Datenstruktur eingefügt. In den abgerundeten Rechtecken sind die Elemente mit der höchsten Wahrscheinlichkeit je Spalte markiert.*

ändernden Bedingungen benötigt. Entscheidend sind dabei die Wortlänge, die durchschnittlichen Fehler je erkanntem Wort der Texterkennung und die Menge der verschiedenen möglichen Schriftzeichen.

Der Versuchsaufbau

Prinzipiell wird am Ende einer zeitlichen Filterung von vielen Ergebnissen der Texterkennung von dem einen Wort von zeitlich aufeinanderfolgenden Bildern ein Ergebnis errechnet. Dieses Ergebnis ist ein Wort, wobei bei diesem jedes Schriftzeichen mit der höchsten Wahrscheinlichkeit an dieser Position ermittelt wurde. Das Ergebnis ist korrekt, wenn das gefundene Wort mit jedem Schriftzeichen identisch zu dem zu suchenden Wort ist. Natürlich kann auch eine Aussage über ein nicht korrektes Ergebnis getroffen werden in wieweit es dem gewünschten Resultat ähnlich ist. In den folgenden Berechnungen werden aber nur

vollständige Ergebnisse bzw. falsche Ergebnisse gewertet. Demnach wird am Ende das Ergebnis immer mit dem gewünschten Wort verglichen und eine Aussage getroffen, ob es identisch ist oder nicht und damit wird ein Ja oder Nein für dieses vermerkt. Durch eine beliebig große Streuung der möglichen Ergebnis der Texterkennung kann das Ergebnisse bei gleichen Bedingungen zu einem Ja und bei einer weiteren Berechnung mit unterschiedlichen Wörtern zu einem Nein führen. Daher gilt es hier, eine Aussage über die Trefferrate für die einzelnen Bedingungen zu finden Prozentual gesehen können dann Ergebnisse der Erkennungsrate zwischen 0% und 100% erwartet werden. Es gilt dabei, dass je mehr Berechnungen durchgeführt werden, die Genauigkeit der Aussage über die Erkennungsrate zunimmt.

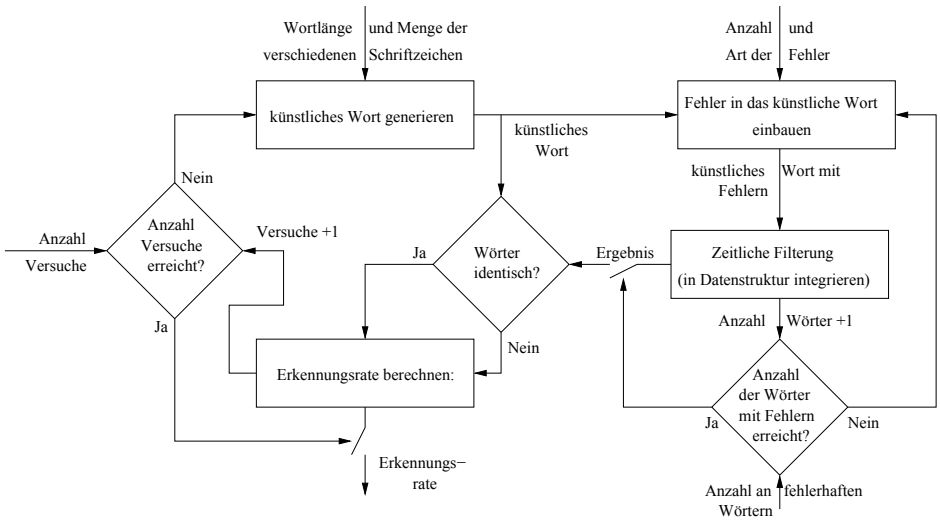
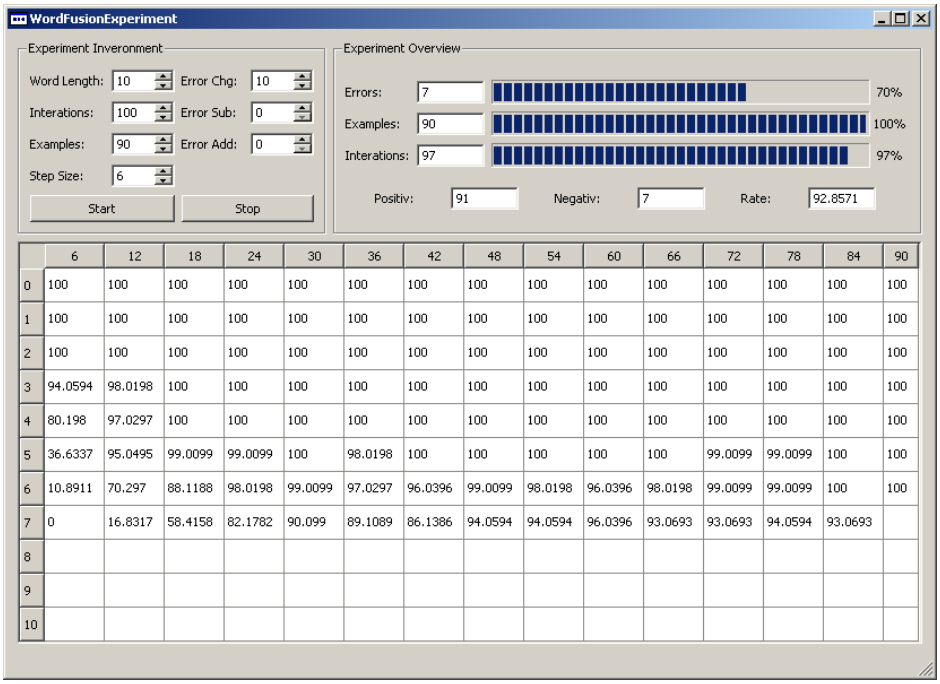


Abbildung 5.5: *Zur Bestimmung der Erkennungsrate werden verschiedene Eingaben benötigt. Die Anzahl der Versuche bestimmt, wieviel Berechnungen durchgeführt werden um eine Erkennungsrate zu bestimmen. Die Wortlänge gibt an, wie viel Schriftzeichen das verwendete künstlich generierte Wort besitzen muss. Die Anzahl und Art der Fehler bestimmt, welche Fehler in dem künstlich generierten Wort für einen Eingang in die zeitliche Filterung erzeugt werden sollen. Die Anzahl der fehlerhaften Wörter gibt an, wieviele fehlerhafte Wörter zur zeitlichen Filterung benutzt werden sollen.*

Für die einzelnen sich ändernden Bedingungen müssen nun Versuchsbeispiele gefunden werden. Da der Wortschatz für jeden Fall limitiert ist, kann die Genauigkeit der Erkennungsrate eventuell nicht gut genug bestimmt werden. Es liegt nahe, hier künstliche Wörter zu erzeugen. Das Verfahren achtet nicht auf den Sinn von Silben oder ähnlichen Zusammenhängen von Schriftzeichen in den Wörtern. Daher können hier Wörter mit beliebig kombinierten Schriftzeichen erstellt werden. Nachdem ein solches Wort gefunden wurde wäre es möglich, dieses mit einem Video zu filmen eine Detektion und Erkennung von Text auf den Bildern des Videos durchzuführen und die einzelnen Ergebnisse für das Verfahren zu nutzen. Am Ende könnte dann das Ergebnis des Verfahrens mit dem künstliche gefundenen Wort wieder

verglichen werden. Diese Variante birgt einige Nachteile. Zum einen ist es zeitlich enorm aufwendig und zum anderen wird das Ergebnis der Erkennungsrate durch die Detektion und Erkennung des Texts und deren Fehler beeinträchtigt. Eine Lösung hierfür wäre, aus dem künstlich gefundenen Wort ebenfalls künstlich gefundene Fehlerwörter zu erzeugen. Hierfür können zufällig Schriftzeichen ersetzt, gelöscht oder zusätzliche Schriftzeichen eingefügt werden. Dazu wird eine Gleichverteilung der Fehler über alle Schriftzeichen und deren Positionen angenommen. Auf diese Weise ist es möglich, für gleiche Bedingungen fast beliebig viele künstliche Wörter mit zusätzlich beliebig vielen Wörtern mit festgelegter Fehlerzahl zu erzeugen. Die Wörter mit den Fehlern können dann mit dem Verfahren ausgewertet und mit dem künstlichen Wort ohne Fehler verglichen werden. Ist das Ergebnis mit dem künstlichen Wort identisch, wird ein Ja festgehalten, andernfalls ein Nein. Diese Methode kann solange wiederholt werden, bis die Genauigkeit der Erkennungsrate erreicht wurde. Als Resultat wird nun für bestimmte Bedingungen, für das das Verfahren getestet wurde, eine Erkennungsrate mit gewünschter Genauigkeit gefunden. Ein solcher Versuch ist in Abbildung 5.5 definiert Das Resultat von einem solchen Versuch ist genau eine Erkennungsrate für die in dem Versuch genutzten Parameter.

Interessant für die Auswertung der einzelnen Parameter ist ein Vergleich der Ergebnisse, wenn wenige bzw. nur ein Parameter sich ändert und die anderen konstant gehalten werden. Auf diese Weise können einzelne Aussagen zu den verschiedenen Parametern getroffen werden. Dazu ist es notwendig, viele solche Versuche zur Bestimmung der Erkennungsrate in Folge zu berechnen und in einer Grafi darzustellen. Das in Abbildung 5.6 dargestellte Programm erfüllt diesen Zweck. Hier werden bestimmte Parameter festgelegt und je 2 Parameter werden variiert. Dadurch entsteht eine zweidimensionale Matrix von Erkennungsraten, die später in einer Grafi dargestellt und analysiert werden kann. Dazu sind am Anfang die Wortlänge, die Anzahl der Wiederholungen pro Erkennungsrate und die Art der Fehler festgelegt. In diesem Fall wird die Anzahl der benutzen Wörter für jede Erkennungsrate in Schritten von 6 bis 90 berechnet. Diese Einstellung wurde durch die eingegebenen Werte 90 im Feld „Examples“ und 6 im Feld „Step Size“ definiert Außerdem werden die Fehler von 0 bis 10 berechnet. Die Anzahl und Art der Fehler ergibt sich aus dem eingegebenen Wert 10 im Feld „Error Chg“. Hier werden nur Schriftzeichen als Fehler ausgetauscht. „Error Sub“ steht für gelöschte Schriftzeichen und „Error Add“ für hinzugefügte Schriftzeichen. Mit diesen 3 Fehlerquellen sind die 3 Operationen zum Umformen eines Wortes in ein anderes abgedeckt. Insgesamt besitzt jedes künstlich erzeugte Wort die Länge 10, welches durch das Feld „Word Length“ eingestellt wurde. Für die Berechnung der Erkennungsrate werden 100 Versuche durchgeführt, welches im Feld „Iterations“ definier ist. Rechts neben den Einstellungen sind der aktuell zu berechnende Parametersatz und die entsprechenden Fortschrittsbalken für die gesamte Berechnung dargestellt. Das entsprechende Zwischenergebnis ist durch die Felder „Positiv“, „Negativ“ und „Rate“ jeder Zeit abzulesen. Das Endergebnis jeder Berechnung der Erkennungsrate für einen Parametersatz wird dann in das entsprechende Feld in der Matrix unten eingetragen. In der Waagerechten werden dann die in der zeitlichen Filterung der Texterkennung benutzen Wörter aufgetragen. Hierfür kann eine Schrittgröße eingegeben werden, die bestimmt, wie viel mehr Wörter in jedem Versuch benutzt werden sollen. In der Senkrechten werden dann die Anzahl der durchschnittlichen Fehler pro Wort, beginnend bei null Fehlern bis zur eingestellten Größe, aufgetragen. So wird die Anzahl an durchschnittlichen Fehler pro Wort im Verhältnis zu deren Anzahl aufgelistet.



	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90
0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
1	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
3	94.0594	98.0198	100	100	100	100	100	100	100	100	100	100	100	100	100
4	80.198	97.0297	100	100	100	100	100	100	100	100	100	100	100	100	100
5	36.6337	95.0495	99.0099	99.0099	100	98.0198	100	100	100	100	100	99.0099	99.0099	100	100
6	10.8911	70.297	88.1188	98.0198	99.0099	97.0297	96.0396	99.0099	98.0198	96.0396	98.0198	99.0099	99.0099	100	100
7	0	16.8317	58.4158	82.1782	90.099	89.1089	86.1386	94.0594	94.0594	96.0396	93.0693	93.0693	94.0594	93.0693	
8															
9															
10															

Abbildung 5.6: *In dieser Abbildung ist die Oberf äche des Programms zur Bestimmung der Erkennungsraten für Versuchsreihen zur Analyse der zeitlichen Filterung der Ergebnisse der Texterkennung zu erkennen. Für jeden Versuch werden einige Parameter festgelegt. Damit eine Aussage für den entsprechenden Parametersatz getroffen werden kann, werden die Anzahl der durchschnittlichen Fehler in der Senkrechten in jedem Wort und die Anzahl der benutzten Wörter in der Waagerechten für die Berechnung der Erkennungsraten variiert.*

Erstellen eines künstlichen Wortes und Erzeugung von Fehlern

Für die Erzeugung eines künstlichen Wortes sind die Menge an zu verwendenden Schriftzeichen und die Länge des zu erzeugenden Wortes entscheidend. Für die folgenden Experimente zur Bestimmung der Zuverlässigkeit der zeitlichen Filterung werden hier zwei verschiedene Mengen an Schriftzeichen betrachtet. Hierbei ist das Ziel die Menge zu variieren um eine Aussage bei den Experimenten über den Einflus von unterschiedlichen Schriftzeichenmengen zu gewinnen. Hier wird die Menge von 25 bzw. 61 Schriftzeichen für die Erstellung der künstlichen Wörter verwendet. In Tabelle 5.14 sind die beiden Schriftzeichenmengen dargestellt.

Die Auswahl der Parameter für die Versuche

Für die Messung der Zuverlässigkeit des Verfahrens müssen die Parameter, die das Ergebnis beeinflusse können, variiert und gemessen werden. Als Basis dient für jede Untersuchung

Schriftzeichenmenge 1	Schriftzeichenmenge 2
’a’, ’b’, ’c’, ’d’, ’e’, ’d’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’	’a’, ’b’, ’c’, ’d’, ’e’, ’d’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’G’, ’H’, ’I’, ’J’, ’K’, ’L’, ’M’, ’N’, ’O’, ’P’, ’Q’, ’R’, ’S’, ’T’, ’U’, ’V’, ’W’, ’X’, ’Y’, ’Z’, ’ ’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’0’

Tabelle 5.14: *In der Tabelle sind zwei unterschiedlich große Mengen an Schriftzeichen aufgezählt. Jedes Schriftzeichen ist in Anführungsstrichen ’ ’ dargestellt. In der rechten Spalte befinde sich 61 Schriftzeichen. Es sind alle Schriftzeichen von a bis z in großer und kleiner Schreibweise vorhanden. Zusätzlich befinde sich noch in dieser Menge die Zahlen 0 bis 9 und das Leerzeichen, welches ebenfalls als Schriftzeichen gewertet wird. In der linken Spalte befinde sich lediglich 25 Schriftzeichen mit a bis z in kleiner Schreibweise.*

eines Parameters das Durchlaufen von einer vordefinierte Menge an benutzten Wörtern für jede Messung, welche kontinuierlich von 3 Wörtern bis 150 Wörtern gesteigert wird und eine vordefiniert Menge an Fehlern je Wort, die ebenfalls kontinuierlich von 0 Fehlern bis zur maximalen Anzahl von Fehlern gesteigert wird. Jede Kombination ergibt ein Messresultat. So entsteht eine zweidimensionale Matrix mit Messwerten. Die Messwerte können dann in die dritte Dimension je nach Höhe aufgetragen werden und bilden mit allen Messungen eine dreidimensionale Messkurve. Die minimale Anzahl von 3 Wörtern ist notwendig, um eine erste Aussage über die Verteilung der Schriftzeichen an den Positionen treffen zu können. Auf der anderen Seite werden maximal 150 Wörter in einer Messung benutzt. In einem Video werden jede Sekunde 25 Bilder gezeigt. So entsprechen die 150 Wörter 150 Einzelbildern und damit 6 Sekunden, in denen das Wort in dem Video zu sehen ist. Die Anzahl der möglichen Fehler je Wort ist, bis auf zusätzliche Schriftzeichen, limitiert auf die Länge des Wortes. In den Experimenten ist daher eine Fehlerzahl von 0 bis zu der Länge des Wortes gewählt. Aus der Tatsache heraus, dass sich fehlende, zu viele oder falsche Schriftzeichen in dem Verfahren anders verhalten können, muss jede der Fälle einzeln gemessen werden. Auf diese Weise kommt es zu einer Dreiteilung der Experimente in Messreihen für fehlende, für zu viele und für falsche Schriftzeichen. In jeder dieser drei Teile müssen die 2 verschiedenen Schriftzeichenmengen gemessen werden. Zusätzlich ist die Wortlänge entscheidend. Hierfür werden weiterführend 3 Wortlängen mit der Länge 5, 10 und 20 verwendet. So ergeben sich daraus 18 Messreihen mit 2 Schriftzeichenmengen x 3 Wortlängen x 3 Fehlertypen.

- Messung falscher Schriftzeichen, Wortlänge = 5 und 61 verschiede Schriftzeichen
- Messung falscher Schriftzeichen, Wortlänge = 5 und 25 verschiede Schriftzeichen
- Messung falscher Schriftzeichen, Wortlänge = 10 und 61 verschiede Schriftzeichen
- Messung falscher Schriftzeichen, Wortlänge = 10 und 25 verschiede Schriftzeichen
- Messung falscher Schriftzeichen, Wortlänge = 20 und 61 verschiede Schriftzeichen

- Messung falscher Schriftzeichen, Wortlänge = 20 und 25 verschiede Schriftzeichen
- Messung fehlender Schriftzeichen, Wortlänge = 5 und 61 verschiede Schriftzeichen
- Messung fehlender Schriftzeichen, Wortlänge = 5 und 25 verschiede Schriftzeichen
- Messung fehlender Schriftzeichen, Wortlänge = 10 und 61 verschiede Schriftzeichen
- Messung fehlender Schriftzeichen, Wortlänge = 10 und 25 verschiede Schriftzeichen
- Messung fehlender Schriftzeichen, Wortlänge = 20 und 61 verschiede Schriftzeichen
- Messung fehlender Schriftzeichen, Wortlänge = 20 und 25 verschiede Schriftzeichen
- Messung zu vieler Schriftzeichen, Wortlänge = 5 und 61 verschiede Schriftzeichen
- Messung zu vieler Schriftzeichen, Wortlänge = 5 und 25 verschiede Schriftzeichen
- Messung zu vieler Schriftzeichen, Wortlänge = 10 und 61 verschiede Schriftzeichen
- Messung zu vieler Schriftzeichen, Wortlänge = 10 und 25 verschiede Schriftzeichen
- Messung zu vieler Schriftzeichen, Wortlänge = 20 und 61 verschiede Schriftzeichen
- Messung zu vieler Schriftzeichen, Wortlänge = 20 und 25 verschiede Schriftzeichen

Da jeder Versuch ein Ergebnis mit erkannt oder nicht erkannt ergibt, kann daraus nicht direkt die Erkennungsrate ermittelt werden. Deshalb muss jede Einstellung mehrfach gemessen und ausgewertet werden. Erst mit einer großen Zahl von Ergebnissen von einer Einstellung ist eine Aussage über die Menge der richtigen zu den falsch erkannten Fällen möglich. Diese Prozentzahl ist dann die Erkennungsrate. Für alle Messkurven werden im Folgenden 1000 Versuche je Erkennungsrate durchgeführt.

Für alle Messkurven gilt, dass in der Achse „Integrierte Wörter“ die Anzahl benutzter Wörter in Schrittlängen mit 3 von 0 bis 150 erhöht wird. In der Achse „Fehler je Wort“ werden die Fehler in Schrittlängen mit 1 von 0 bis 5, 10 bzw. 20 erhöht. In der Achse „Erkennungsrate“ der dreidimensionalen Abbildung werden dann die Werte der Erkennungsrate aufgetragen. Die Fehleranzahl von 0 dient hier als Referenz und Prüfung der Messkurve. Das Verfahren muss bei fehlerfreien Wörtern ebenfalls fehlerfreie Ergebnisse erzeugen und damit eine Erkennungsrate von 100% aufweisen. Die maximale Fehlerzahl 5, 10 bzw. 20 dient ebenfalls als Referenz und Prüfung der Messkurve. In diesem Fall ist jedes Zeichen in jedem integrierten Wort falsch und damit ist es nahezu unmöglich ein richtiges Ergebnis zu erzielen. Es gibt eine extrem geringe Chance, dass dennoch ein Wort identisch zu dem gesuchten Wort ist. Bei 61 Schriftzeichen und einer Wortlänge von 5 ist diese Chance für ein integriertes Wort bei 1 zu 844.596.301. Mit der Anzahl der integrierten Wörter wird diese Chance noch deutlich weiter reduziert. Daher ist bei einer maximalen Fehlerzahl die Erkennungsrate von 0% zu erwarten. Je Messkurve werden zwei verschiedene Abbildungen für die bessere Lesbarkeit dargestellt. In der jeweils linken Abbildung ist die Messkurve dreidimensional dargestellt. In der jeweils rechten Abbildung ist dasselbe Ergebnis zweidimensional dargestellt. In beiden Grafi en geben die Farben Aufschluss über die Erkennungsrate. Rot mit 100% ist hierbei eine hohe und blau mit 0% eine niedrige Erkennungsrate.

Messung von falschen Schriftzeichen

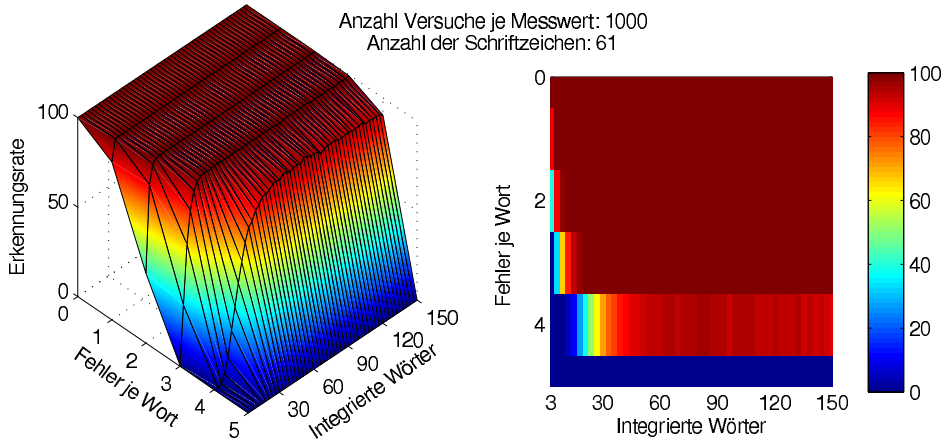


Abbildung 5.7: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

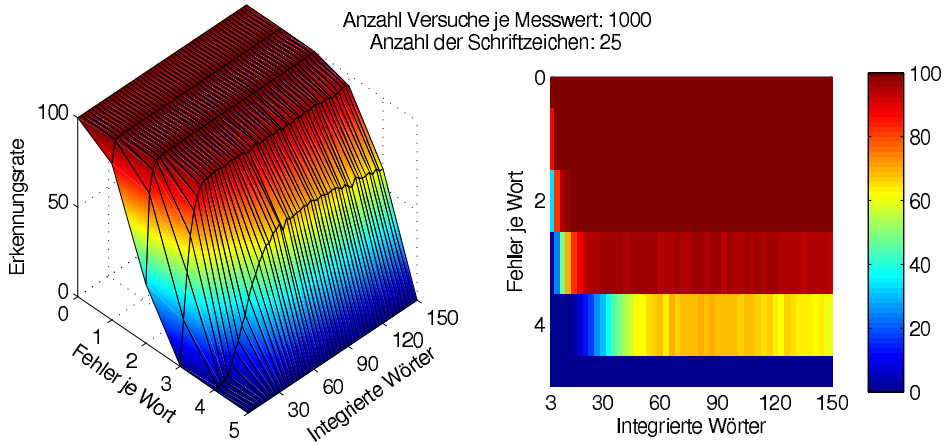


Abbildung 5.8: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

In der Messkurve in Abbildung 5.7 befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der

integriertem Wörter. Bis 80% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 50,5%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 80% und bei 150 integrierten Wörtern auf 90,3%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt.

In der Messkurve in Abbildung 5.8 befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 80% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 18,2%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 44,5% und bei 150 integrierten Wörtern auf 60,4%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt. Im Gegensatz zu der Messkurve in Abbildung 5.7 sind alle Erkennungsraten im direkten Vergleich deutlich geringer.

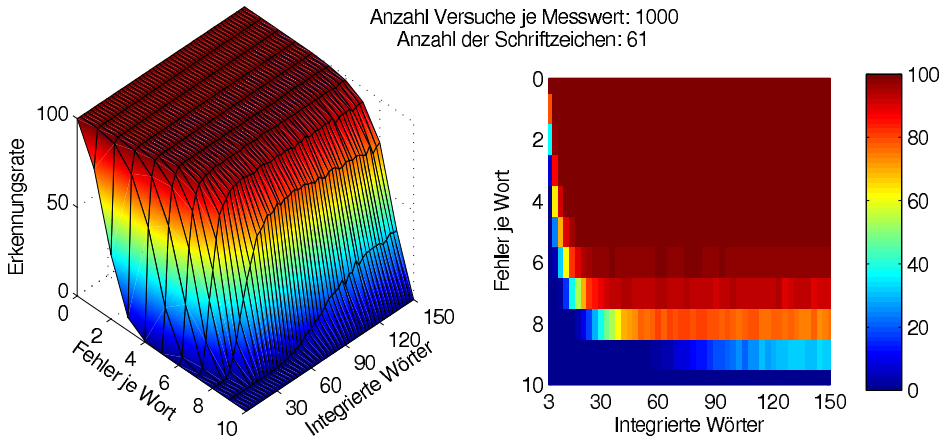


Abbildung 5.9: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

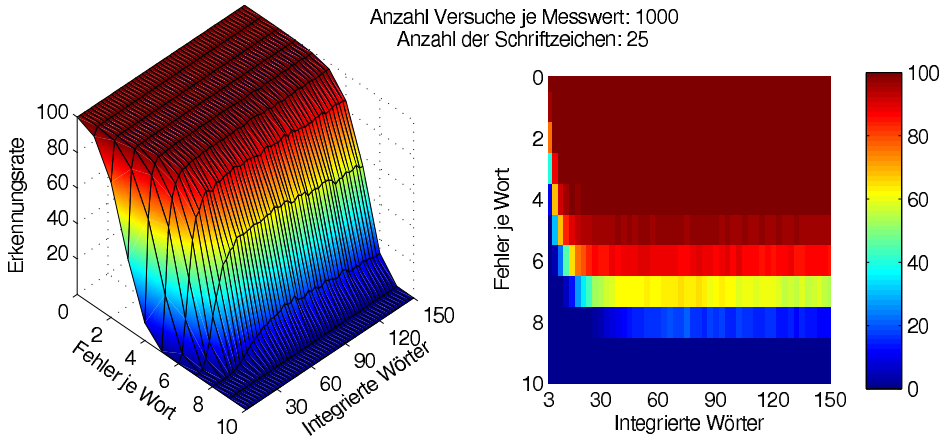


Abbildung 5.10: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

In der Messkurve in Abbildung 5.9 befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.

Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 80% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 17,6%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 54,6% und bei 150

integrierten Wörtern auf 75,4%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt.

In der Messkurve in Abbildung 5.10 befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.

Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 80% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 1,3%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 8,7% und bei 150 integrierten Wörtern auf 12,2%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt. Im Gegensatz zu der Messkurve in Abbildung 5.9 sind alle Erkennungsraten im direkten Vergleich deutlich geringer.

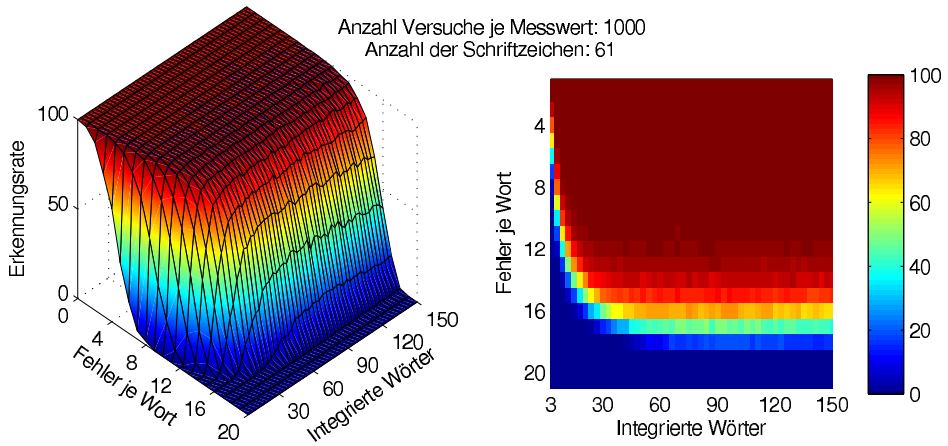


Abbildung 5.11: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

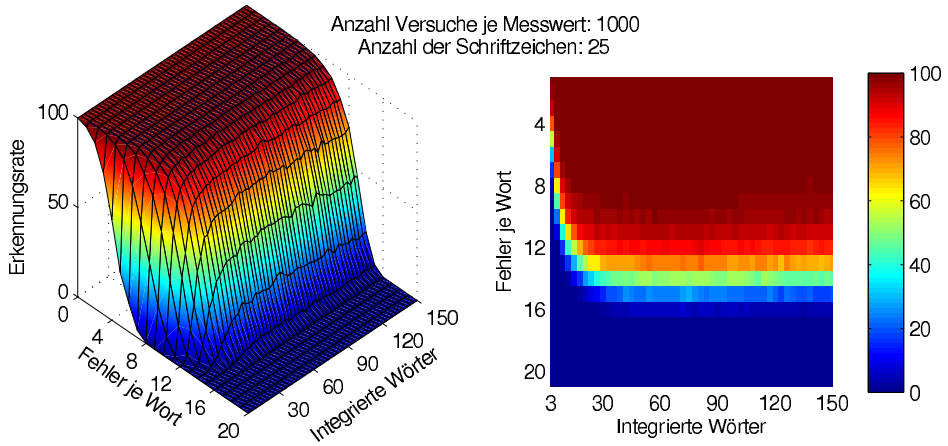


Abbildung 5.12: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

In der Messkurve in Abbildung 5.11 befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.

Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 80% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 0,0%.

Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 24,6% und bei 150 integrierten Wörtern auf 39,6%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt.

In der Messkurve in Abbildung 5.12 befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.

Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 80% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 0%. Mit über 36 integrierten Wörtern liegt die Erkennungsrate noch bei 0% und bei 150 integrierten Wörtern bei 0,1%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt. Im Gegensatz zu der Messkurve in Abbildung 5.9 sind alle Erkennungsraten im direkten Vergleich deutlich geringer.

Schrift-zeichen	Wort-länge	24 Wörter integriert 80% Fehler	36 Wörter integriert 80% Fehler	150 Wörter integriert 80% Fehler
61	5	50,5%	80,0%	90,3%
	10	17,6%	54,6%	75,4%
	20	0,0%	24,6%	39,6%
25	5	18,2%	44,5%	60,4%
	10	1,3%	8,7	12,2%
	20	0,0%	0%	0,1%

Tabelle 5.15: *Vergleich charakteristischer Messwerte aus allen Messkurven für falsche Schriftzeichen.*

In Tabelle 5.15 sind einige charakteristische Messwerte aus den einzelnen Messkurven für falsche Schriftzeichen aufgeführt. Hierbei war die Fehlermenge von 80% je integriertes Wort und die Anzahl von 24, 36 und 150 integrierten Wörtern entscheidend. Die Erkennungsraten sind bei allen Messungen mit einer kleinen Schriftzeichenmenge geringer als bei einer größeren Schriftzeichenmenge. Daraus lässt sich schließen, dass mit steigender Schriftzeichenmenge die Erkennungsrate steigt. Der Grund hierfür ist in der Wahrscheinlichkeit für ein bestimmtes Schriftzeichen zu suchen. Dagegen sinkt die Erkennungsrate mit der Länge der Wörter.

Messung von fehlenden Schriftzeichen

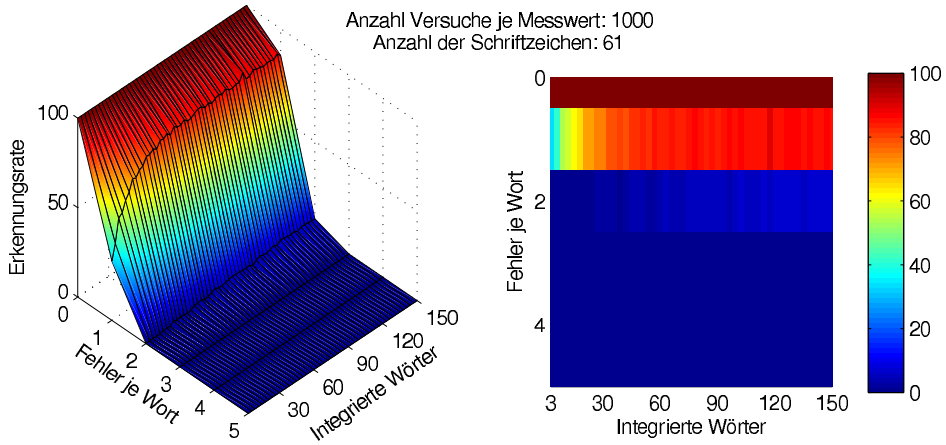


Abbildung 5.13: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gelöscht.*

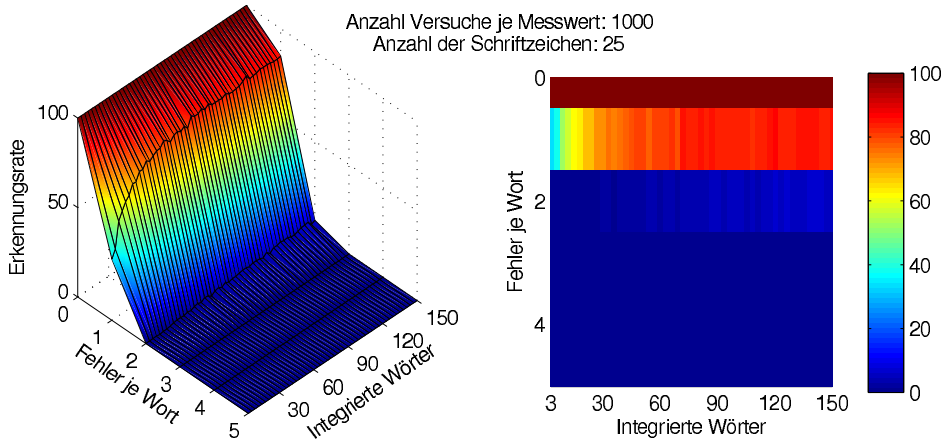


Abbildung 5.14: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gelöscht.*

In der Messkurve in Abbildung 5.13 befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gelöscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler

je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 70,6%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 78,6% und bei 150 integrierten Wörtern auf 84,9%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt.

In der Messkurve in Abbildung 5.14 befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gelöscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 68,3%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 74,6% und bei 150 integrierten Wörtern auf 84,6%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt. Im Gegensatz zu der Messkurve in Abbildung 5.13 sind alle Erkennungsraten im direkten Vergleich geringer.

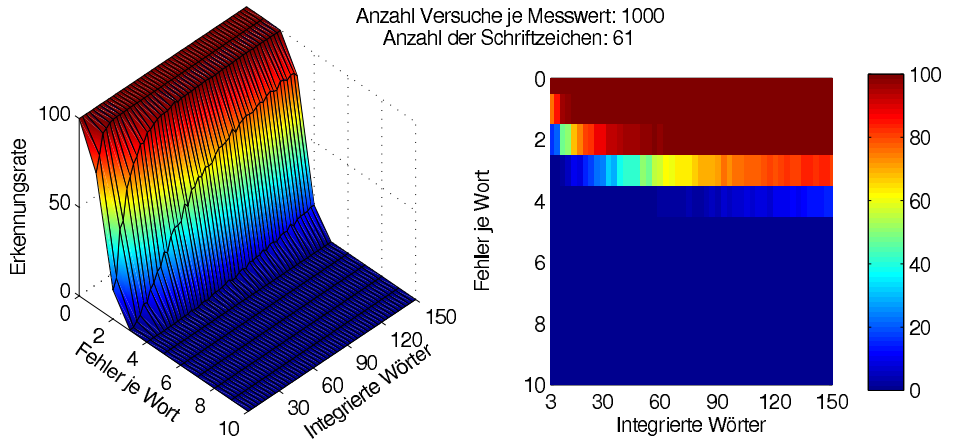


Abbildung 5.15: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gelöscht.*

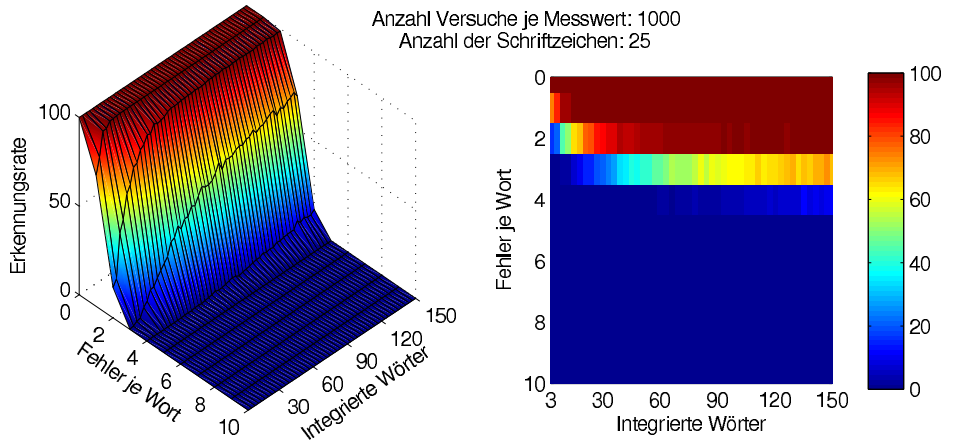


Abbildung 5.16: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gelöscht.*

In der Messkurve in Abbildung 5.15 befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gelöscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 82,7%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 92,8% und bei 150 integrierten

Wörtern auf 99,7%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt.

In der Messkurve in Abbildung 5.16 befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gelöscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 78,4%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 89,7% und bei 150 integrierten Wörtern auf 98,6%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt. Im Gegensatz zu der Messkurve in Abbildung 5.15 sind alle Erkennungsraten im direkten Vergleich geringer.

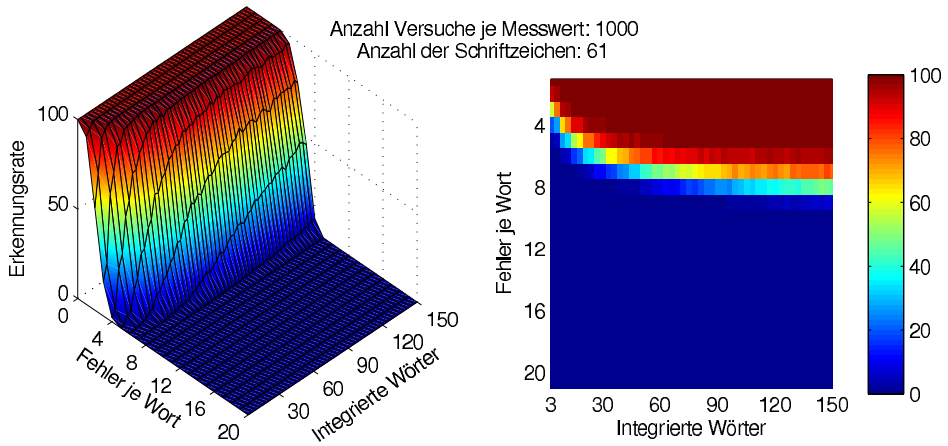


Abbildung 5.17: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gelöscht.*

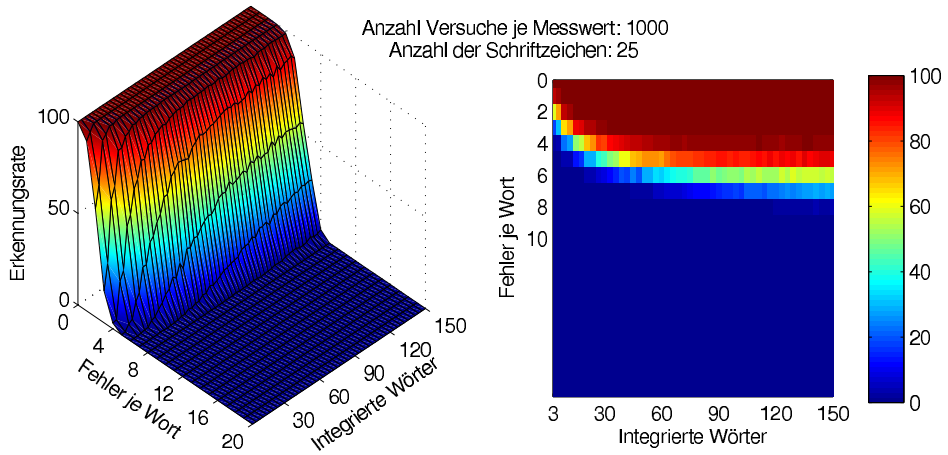


Abbildung 5.18: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gelöscht.*

In der Messkurve in Abbildung 5.17 befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen gelöscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 76,0%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 93,4% und bei 150 integrierten

Wörtern auf 99,8%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt.

In der Messkurve in Abbildung 5.18 befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen gelöscht. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 72,6%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 87,5% und bei 150 integrierten Wörtern auf 98,2%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%, genauso wie bei maximalen Fehlern je Wort die Erkennungsrate bei 0% liegt. Im Gegensatz zu der Messkurve in Abbildung 5.17 sind alle Erkennungsraten im direkten Vergleich geringer.

Schrift-zeichen	Wort-länge	24 Wörter integriert 20% Fehler	36 Wörter integriert 20% Fehler	150 Wörter integriert 20% Fehler
61	5	70,6%	78,6%	84,9%
	10	82,7%	92,8%	99,7%
	20	76,0%	93,4%	99,8%
25	5	68,3%	74,6%	84,6%
	10	78,4%	89,7	98,6%
	20	72,6%	87,5%	98,2%

Tabelle 5.16: *Vergleich charakteristischer Messwerte aus allen Messkurven für fehlenden Schriftzeichen.*

In Tabelle 5.16 sind einige charakteristische Messwerte aus den einzelnen Messkurven für fehlende Schriftzeichen aufgeführt. Hierbei war die Fehlermenge von 20% je integriertes Wort und die Anzahl von 24, 36 und 150 integrierten Wörtern entscheidend. Die Erkennungsraten sind bei allen Messungen mit einer kleinen Schriftzeichenmenge geringer als bei einer größeren Schriftzeichenmenge. Daraus lässt sich schließen, dass mit steigender Schriftzeichenmenge die Erkennungsrate steigt. Der Grund hierfür ist in der Wahrscheinlichkeit für ein bestimmtes Schriftzeichen zu suchen. Im Gegensatz zu den Messungen für falsche Schriftzeichen werden die Erkennungsraten bei fehlenden Schriftzeichen mit steigender Wortlänge im Schnitt höher.

Messung von zu vielen Schriftzeichen

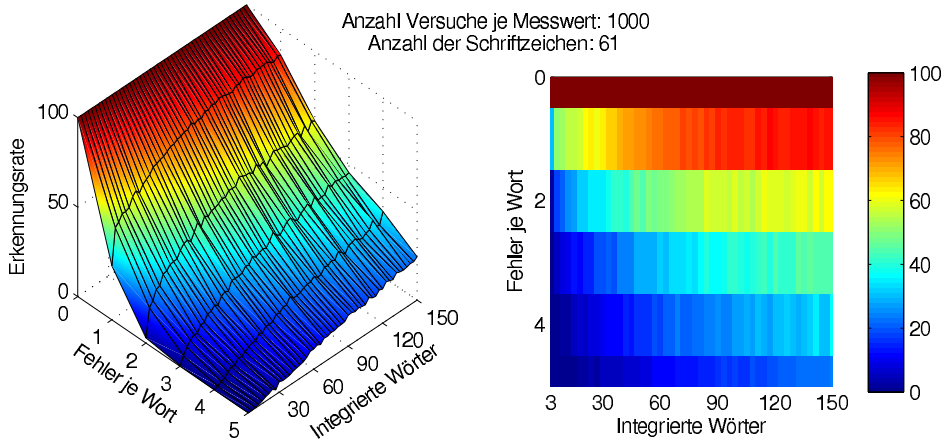


Abbildung 5.19: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen hinzugefügt.*

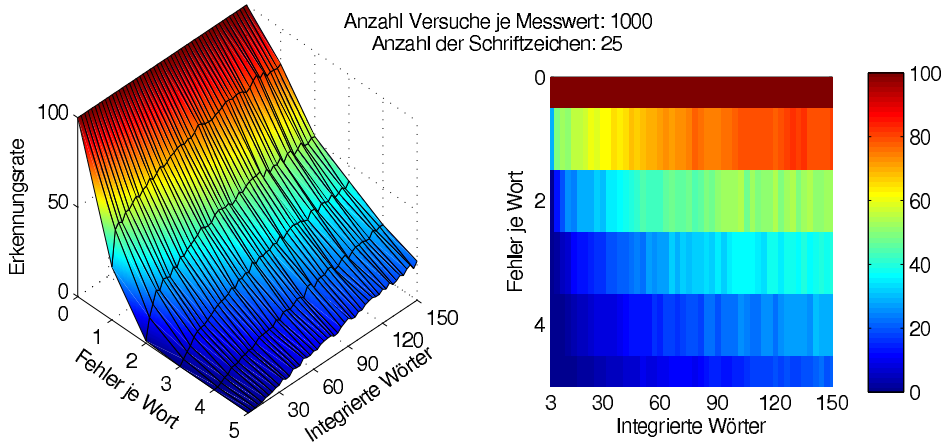


Abbildung 5.20: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen hinzugefügt.*

In der Messkurve in Abbildung 5.19 befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen hinzugefügt. In diesem Fall ist die Fehleranzahl je Wort beliebig groß, da beliebig viele Schriftzeichen hinzugefügt werden können. Als maximale Fehlerzahl wird hier die gleiche Menge wie die

originale Wortlänge mit 5 verwendet. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 63,1%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 67% und bei 150 integrierten Wörtern auf 85%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%.

In der Messkurve in Abbildung 5.20 befinde sich die Erkennungsraten für die Wortlänge 5 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen hinzugefügt. In diesem Fall ist die Fehleranzahl je Wort beliebig groß, da beliebig viele Schriftzeichen hinzugefügt werden können. Als maximale Fehlerzahl wird hier die gleiche Menge wie die originale Wortlänge mit 5 verwendet. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 59,9%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 65% und bei 150 integrierten Wörtern auf 79,9%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%. Im Gegensatz zu der Messkurve in Abbildung 5.19 sind alle Erkennungsraten im direkten Vergleich geringer.

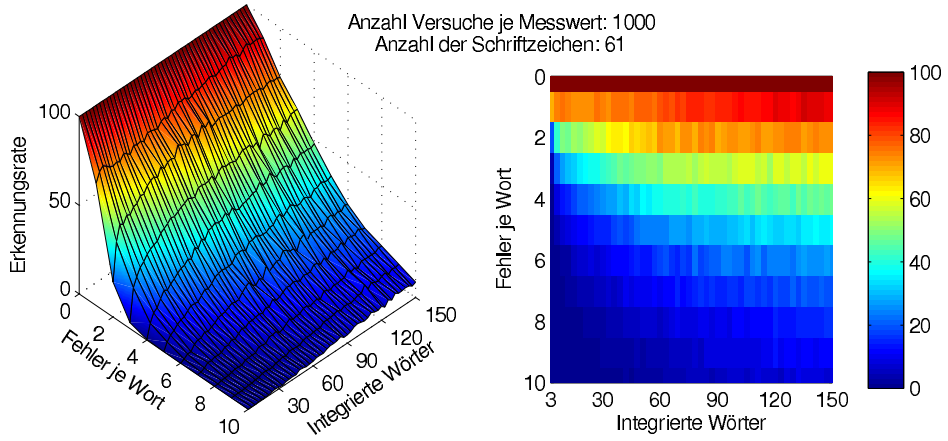


Abbildung 5.21: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen hinzugefügt.*

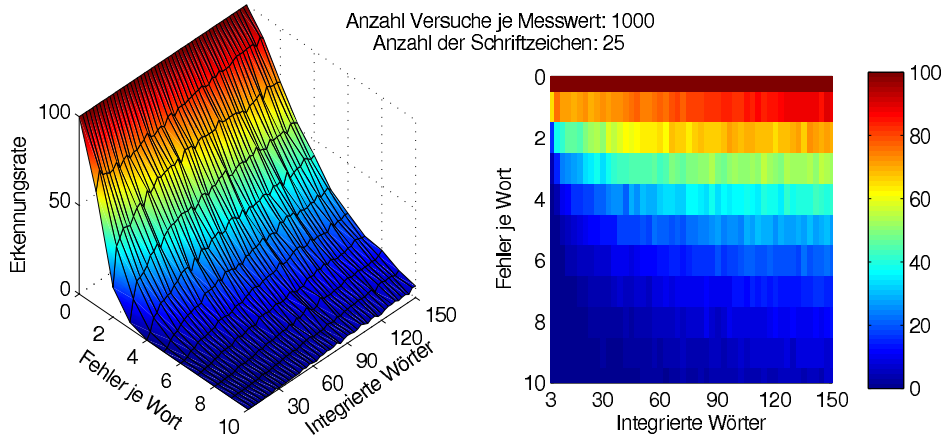


Abbildung 5.22: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen hinzugefügt.*

In der Messkurve in Abbildung 5.21 befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen hinzugefügt. In diesem Fall ist die Fehleranzahl je Wort beliebig groß, da beliebig viele Schriftzeichen hinzugefügt werden können. Als maximale Fehlerzahl wird hier die gleiche Menge wie die originale Wortlänge mit 10 verwendet. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die

Erkennungsrate noch bei 54,9%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 62,7% und bei 150 integrierten Wörtern auf 71,9%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%.

In der Messkurve in Abbildung 5.22 befinde sich die Erkennungsraten für die Wortlänge 10 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen hinzugefügt. In diesem Fall ist die Fehleranzahl je Wort beliebig groß, da beliebig viele Schriftzeichen hinzugefügt werden können. Als maximale Fehlerzahl wird hier die gleiche Menge wie die originale Wortlänge mit 10 verwendet. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 52,5%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 55,9% und bei 150 integrierten Wörtern auf 69,4%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%. Im Gegensatz zu der Messkurve in Abbildung 5.21 sind alle Erkennungsraten im direkten Vergleich geringer.

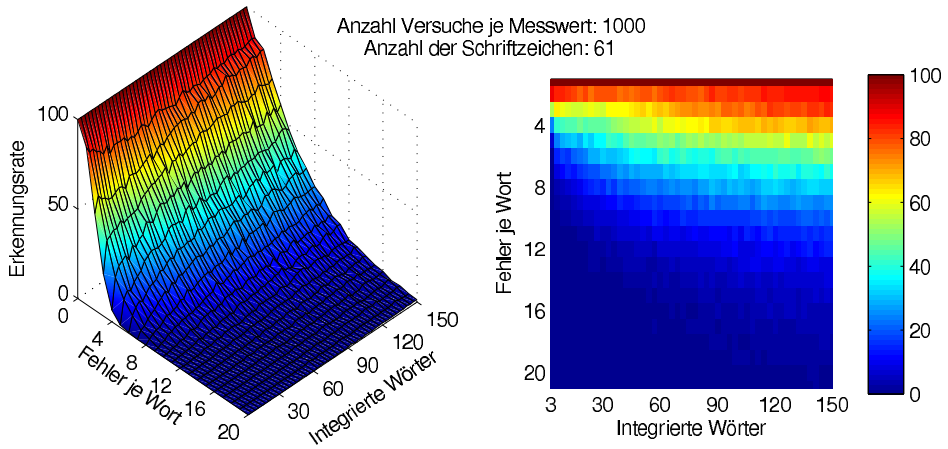


Abbildung 5.23: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen hinzugefügt.*

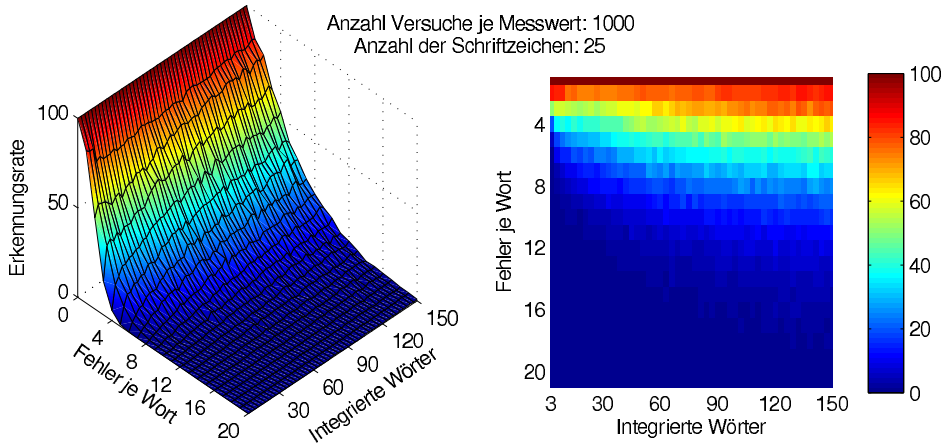


Abbildung 5.24: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen hinzugefügt.*

In der Messkurve in Abbildung 5.23 befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 61. Als Fehler werden hier Schriftzeichen hinzugefügt. In diesem Fall ist die Fehleranzahl je Wort beliebig groß, da beliebig viele Schriftzeichen hinzugefügt werden können. Als maximale Fehlerzahl wird hier die gleiche Menge wie die originale Wortlänge mit 20 verwendet. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der

integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate noch bei 32,8%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 41,6% und bei 150 integrierten Wörtern auf 58,3%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%.

In der Messkurve in Abbildung 5.24 befinde sich die Erkennungsraten für die Wortlänge 20 mit einer Schriftzeichenanzahl von 25. Als Fehler werden hier Schriftzeichen hinzugefügt. In diesem Fall ist die Fehleranzahl je Wort beliebig groß, da beliebig viele Schriftzeichen hinzugefügt werden können. Als maximale Fehlerzahl wird hier die gleiche Menge wie die originale Wortlänge mit 20 verwendet. Es ist zu erkennen, dass, je mehr Fehler in den Wörtern vorkommen, die Erkennungsrate sinkt. Dagegen steigt die Erkennungsrate mit der Anzahl der integrierten Wörter. Bis 20% Fehler je integriertem Wort und 24 integrierten Wörtern ist die Erkennungsrate bei 30,8%. Mit über 36 integrierten Wörtern steigt die Erkennungsrate auf 36% und bei 150 integrierten Wörtern auf 53,5%. Wie erwartet ist bei 0 Fehlern je integriertem Wort die Erkennungsrate immer 100%. Im Gegensatz zu der Messkurve in Abbildung 5.23 sind alle Erkennungsraten im direkten Vergleich geringer.

Schrift-zeichen	Wort-länge	24 Wörter integriert 20% Fehler	36 Wörter integriert 20% Fehler	150 Wörter integriert 20% Fehler
61	5	63,1%	67,0%	85,0%
	10	54,9%	62,7%	71,9%
	20	32,8%	41,6%	58,3%
25	5	59,9%	65,0%	79,9%
	10	52,5%	55,9	69,4%
	20	30,8%	36,0%	53,5%

Tabelle 5.17: *Vergleich charakteristischer Messwerte aus allen Messkurven für hinzugefügte Schriftzeichen.*

In Tabelle 5.17 sind einige charakteristische Messwerte aus den einzelnen Messkurven für hinzugefügte Schriftzeichen aufgeführt. Hierbei war die Fehlermenge von 20% je integriertes Wort und die Anzahl von 24, 36 und 150 integrierten Wörtern entscheidend. Die Erkennungsraten sind bei allen Messungen mit einer kleinen Schriftzeichenmenge geringer als bei einer größeren Schriftzeichenmenge. Daraus lässt sich schließen, dass mit steigender Schriftzeichenmenge die Erkennungsrate steigt. Der Grund hierfür ist in der Wahrscheinlichkeit für ein bestimmtes Schriftzeichen zu suchen. Dagegen sinkt die Erkennungsrate mit der Länge der Wörter.

Aussagen über die Zuverlässigkeit des Verfahrens

Im Falle einer richtigen Erkennung wurden alle Fehler, die durch die Detektion und Erkennung von Text entstanden sind, korrigiert. Jedes vollständig richtig erkannte Wort, welches von dem Verfahren gefunden wurde, ist eine Verbesserung der Detektion und Erkennung von Text. Aus den bisherigen Messungen hat sich ergeben, dass das Verfahren bei steigender Anzahl von integrierten Wörtern in der Datenstruktur höhere Erkennungsraten erreicht. Mit anderen Worten: Um so mehr Bilder von einem Text in einem Video vorhanden sind, umso höhere Trefferraten können erzielt werden. Wie zu erwarten war, sinkt die Erkennungsrate mit steigenden Fehlern in den integrierten Wörtern. Ab einem bestimmten Punkt treten so viele Fehler auf, dass eine Korrektur durch das hier vorgestellte Verfahren nicht mehr möglich ist. Weiterhin hat sich gezeigt, dass die Erkennungsrate im Schnitt mit steigender Wortlänge sinkt. Der Grund hierfür liegt in den Wahrscheinlichkeiten von Fehlern an jeder Position. Jede Position hat die gleiche Wahrscheinlichkeit für ein falsches Schriftzeichen. Für eine richtige Erkennung müssen alle Zeichen korrekt sein. Daher reicht schon ein Fehler für eine Falscherkennung aus. Da mit der Anzahl der Positionen immer wieder eine weitere Chance für ein falsches Zeichen hinzukommt, ist eine schlechtere Erkennungsrate für lange Wörter zu erwarten gewesen. Einen weiteren Einflus auf die Erkennungsrate haben die Schriftzeichenmengen. Die Schriftzeichenmenge bezieht sich auf die möglichen Schriftzeichen an jeder Position in den Wörtern. Hier hat sich gezeigt: Je mehr Schriftzeichen möglich sind, umso höher wird die Erkennungsrate. Der Grund hierfür liegt in der Variation der Fehler. Wenn es zu einem Fehler kommt, können bei mehr möglichen Schriftzeichen die Fehler mehr verteilt werden. Wird zum Beispiel je Position eine Fehlerwahrscheinlichkeit von 10% angenommen und es gibt 4 verschiedene mögliche Schriftzeichen, dann besitzt jedes falsches Schriftzeichen eine Wahrscheinlichkeit von 33,3%. Steigt die Schriftzeichenmenge auf 61 Schrifteichen, dann besitzt jedes falsche Schriftzeichen nur noch eine Wahrscheinlichkeit von 1,67%. Die Chance ist dann deutlich geringer, dass ein immer wieder identisches falsches Schriftzeichen an einer Position auftritt und damit das richtige Schriftzeichen überwiegt. Werden die 3 verschiedenen Fehlerquellen „Ersetzen“, „Löschen“ und „Hinzufügen“ auf Erkennungsraten verglichen, zeigt sich, dass die Erkennungsraten beim „Vertauschen“ deutlich höher liegen. Für ersetzte Schriftzeichen liegt die Grenze für die Korrektur durch das Verfahren ca. bei 80-90% Fehler je integriertem Wort. Hier spielt de Länge der integrierten Wörter eine große Rolle. Haben die integrierten Wörter eine andere Länge zum Originalwort können ab einer bestimmten Länge durch die Levenshtein Matrix die richtigen Positionen für die Schriftzeichen nicht mehr ermittelt werden. Sind zu viele Schriftzeichen vorhanden, kann es zu mehrfach identisch vorkommenden Strukturen in der Datenstruktur kommen, bei denen jedes weitere integrierte Wort, je nach eigenen Fehlern, in die eine oder andere Struktur integriert wird. Sind dagegen zu wenig Schriftzeichen in den zu integrierenden Wörtern vorhanden, fehlen ab einer bestimmten Länge die nötigen Informationen, um diese sinnvoll zu integrieren. Für zu viele bzw. zu wenige Schriftzeichen liegt die Grenze für die Korrektur durch das Verfahren ca. bei 20-30% Fehler je integriertem Wort. Aus den gefundenen Eigenschaften des Verfahrens sollten Messungen über eine kurze Wortlänge und bei einer sehr großen Schriftzeichenmenge die besten Erkennungsraten erzielt werden. Als letztes Experiment für die Zuverlässigkeit des Verfahrens soll hier die Wortlänge 5 dienen. Als Schriftzeichenmenge wird der komplette ASCII-Zeichensatz von 256 Zeichen ohne die 32 Steuerzeichen genutzt. Das entspricht 224 verschiedenen möglichen Schriftzeichen je Position. Es müssen wieder die 3 verschiedenen

Fehlerquellen „Ersetzen“, „Löschen“ und „Hinzufügen“ geprüft werden.

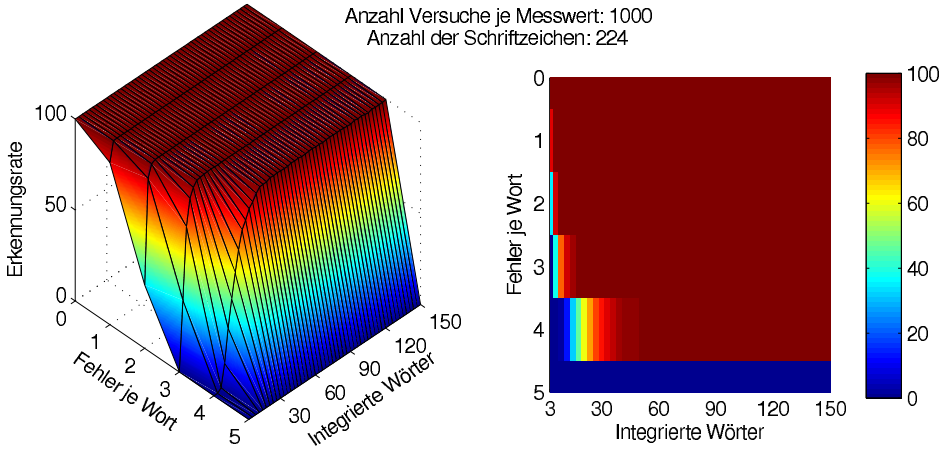


Abbildung 5.25: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 und mit einer theoretischen Schriftzeichenanzahl von 224. Als Fehler werden hier Schriftzeichen gegen falsche Schriftzeichen ausgetauscht.*

Schrift-zeichen	Wort-länge	24 Wörter integriert 80% Fehler	36 Wörter integriert 80% Fehler	150 Wörter integriert 80% Fehler
25	5	18,2%	44,5%	60,4%
61	5	50,5%	80,0%	90,3%
224	5	71,2%	93,4%	99,7%

Tabelle 5.18: *Vergleich der Ergebnisse von Messungen unterschiedlicher Schriftzeichenmengen. Die Fehlerquelle sind vertauschte Schriftzeichen.*

Aus Abbildung 5.25 und dem Vergleich mit Tabelle 5.18 ergibt sich das vorausgesagte Ergebnis. Die Erkennungsraten sind mit der theoretischen Schriftzeichenmenge von 224 deutlich über den Erkennungsraten der anderen Messungen bei vertauschten Schriftzeichen.

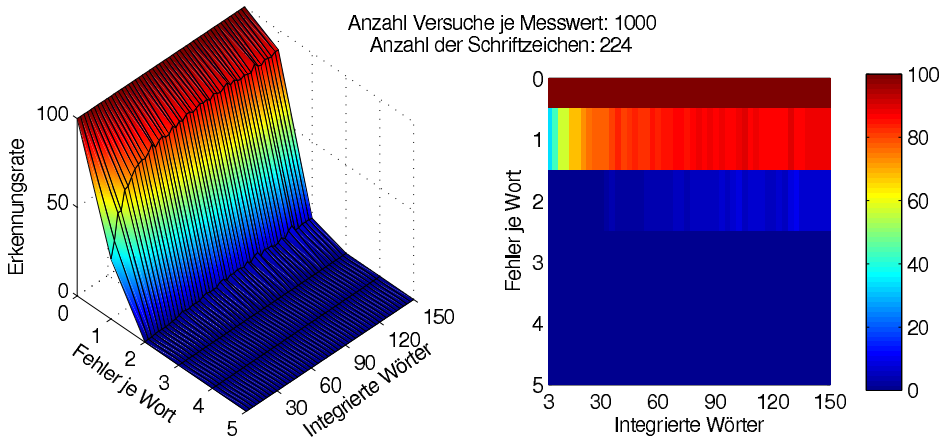


Abbildung 5.26: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 und mit einer theoretischen Schriftzeichenanzahl von 224. Als Fehler werden hier Schriftzeichen gelöscht.*

Schrift-zeichen	Wort-länge	24 Wörter integriert 20% Fehler	36 Wörter integriert 20% Fehler	150 Wörter integriert 20% Fehler
25	5	68,3%	74,6%	84,6%
61	5	70,6%	78,6%	84,9%
224	5	75,5%	79,8%	88,7%

Tabelle 5.19: *Vergleich der Ergebnisse von Messungen unterschiedlicher Schriftzeichenmengen. Die Fehlerquelle sind gelöschte Schriftzeichen.*

Aus Abbildung 5.26 und dem Vergleich mit Tabelle 5.19 ergibt sich das vorausgesagte Ergebnis. Die Erkennungsraten sind mit der theoretischen Schriftzeichenmenge von 224 deutlich über den Erkennungsraten der anderen Messungen bei gelöschten Schriftzeichen.

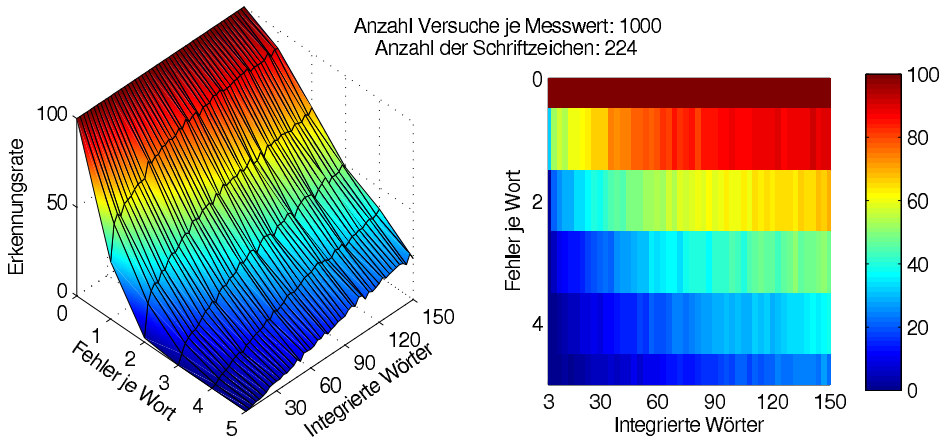


Abbildung 5.27: *In dieser Messkurve befinde sich die Erkennungsraten für die Wortlänge 5 und mit einer theoretischen Schriftzeichenanzahl von 224. Als Fehler werden hier Schriftzeichen hinzugefügt.*

Schrift-zeichen	Wort-länge	24 Wörter integriert 20% Fehler	36 Wörter 20% Fehler	150 Wörter integriert 20% Fehler
25	5	59,9%	65,0%	79,9%
61	5	63,1%	67,0%	85,0%
224	5	63,9%	74,9%	89,5%

Tabelle 5.20: *Vergleich der Ergebnisse von Messungen unterschiedlicher Schriftzeichenmengen. Die Fehlerquelle sind hinzugefügte Schriftzeichen.*

Aus Abbildung 5.27 und dem Vergleich mit Tabelle 5.20 ergibt sich das vorausgesagte Ergebnis. Die Erkennungsraten sind mit der theoretischen Schriftzeichenmenge von 224 deutlich über den Erkennungsraten der anderen Messungen bei hinzugefügten Schriftzeichen. Damit sind bei allen 3 möglichen Fehlerquellen „Ersetzen“, „Löschen“ und „Hinzufügen“ die Erkennungsraten mit steigender Schriftzeichenmenge besser.

5.5.7 Messung der zeitlichen Filterung der Texterkennung in einem realen Video

Nachdem die Zuverlässigkeit der zeitlichen Filterung anhand von theoretischen Messungen ausgewertet wurde ist nun die Analyse von Text in einem realen Video notwendig. Für die Bestimmung der Zuverlässigkeit wird wieder das Video, beschrieben in Abschnitt 3.5, verwendet. Aus der Detektion von Text sind die Textbereiche der einzelnen Wörter bekannt. Über die Verfolgung der Textbereiche können diese über viele Bilder im Video einander zugeordnet werden. So entsteht für jedes zu erkennende Wort eine Liste an Ergebnissen der Texterkennung (siehe Abschnitt 4.5). Für eine bessere Analyse der Zuverlässigkeit der zeit-

lichen Filterung werden hier die gleichen Ausschnitte zur Auswertung der Texterkennung verwendet. Die Erkennungsrate der Schriftzeichen lag hier bei ca. 39,8%. Das entspricht ca. 3 Fehlern bei 5 Schriftzeichen. Die gefundenen Wörter aus der Texterkennung können nun durch die zeitliche Filterung analysiert werden. Als Resultat wird aus der zeitlichen Filterung für jede Liste von Wörtern ein Wort erzeugt. Dieses Wort wird anschließend mit dem manuell gefundenen Wort verglichen. Da für die Liste an Wörtern nun nur noch ein repräsentatives Wort als Ergebnis vorhanden ist, kann nun das Ergebnis für die ganze Liste nur richtig oder falsch sein. Jedes Wort aus der Liste der Erkennung von Text wird dann mit dem Wort aus der zeitlichen Filterung gleichgestellt. Auf diese Weise steht das Ergebnis der zeitlichen Filterung für die gesamte Liste und kann damit direkt verglichen werden.

Die Erkennungsrate (R, engl. Recall) ergibt sich dann aus:

$$R = \frac{N_{TP}}{N_{TP} + N_{FN}},\ P\ \in [0,1] \tag{5.14}$$

Ist das Wort richtig, wird von einer positiv richtigen Erkennung (TP, engl. True Positive) gesprochen. Die falsch erkannten Wörter werden als negative falsche Erkennung (FN, engl. False Negative) bezeichnet. Die unterschiedlichen Fälle können dann in N_{TP} für TP und N_{FN} für FN gezählt werden.

N_{TP}	N_{FP}	R_z
638	285	0,6912

Tabelle 5.21: *Aus den Zählungen von N_{TP} und N_{FP} lässt sich die Erkennungsrate R_z für die zeitliche Filterung bestimmen.*

Die Erkennungsrate R_z der Wörter aus Tabelle 5.21 liegt bei 0,6732 bzw. 67,32%. Die zeitliche Filterung hat damit die Erkennungsrate der Wörter von knapp 39,8% auf über 69,1% verbessert.

Die in dem Experiment zu erkennende durchschnittliche Wortlänge beträgt 6,05. Von den automatisch erkannten Wörter hatten 66,9% die gleiche Länge zu den manuell erkannten Wörtern (W_c). 22,2% der Wörter haben zusätzlich eine zu geringe Wortlänge (W_s) mit durchschnittlich 2,07 zu wenig Schriftzeichen und 10,9% haben dagegen zusätzlich eine zu große Wortlänge (W_a) mit durchschnittlichen 1,95 zu vielen Schriftzeichen. Außerdem kam durchschnittlich jedes zu erkennende Wort in 185,38 Bildern vor. Damit konnte für die zeitliche Filterung ebenfalls durchschnittlich 185 einzelne Ergebnisse der Texterkennung verwendet werden. Damit sind mehr Wörter für jede Filterung vorhanden als in Abschnitt 5.5.6 für falsche Schriftzeichen, 5.5.6 für fehlende Schriftzeichen und 5.5.6 für zu viele Schriftzeichen notwendig waren um eine Sättigung der Verbesserung der Erkennungsrate zu erreichen. Die Messung der Erkennungsrate in einem realen Video kann mit den Messungen in Abbildung 5.7, 5.13 und 5.19 verglichen werden. Die durchschnittliche Wortlänge in den Experimenten auf dem realen Video liegt mit ca. 6 nahe bei der Wortlänge 5 von den 3 Messungen. Zu erkennen waren in allen Messungen 61 Schriftzeichen mit „A“-„Z“„a“-„z“ und „0“-„9“. In der Messung Abbildung 5.7 für falsche Schriftzeichen wurden bei 150 integrierten Wörtern mit einem durchschnittlichen Fehler von 80% je Schriftzeichen eine Erkennungsrate R_c von über 0,9 bzw. 90% erreicht. In der Messung in dem realen Video kommt es aber

in den integrierten Wörtern nicht nur zu falschen Schriftzeichen, sondern auch zu fehlenden und zu vielen Schriftzeichen. Für die 22,2% der Wörter mit durchschnittliche 2 zu wenigen Schriftzeichen kann aus der Abbildung 5.13 eine Erkennungsrate R_s von ca. 0,07 bzw. 7% abgelesen werden. Für die 10,9% der Wörter mit durchschnittliche 2 zu vielen Schriftzeichen kann aus der Abbildung 5.19 eine Erkennungsrate R_a von ca. 0,61 bzw. 61% abgelesen werden. Werden die unterschiedlichen Erkennungsraten gewichtet gemittelt ergibt sich eine theoretische Erkennung R_T von 0,675 bzw 67,5%.

$$
\begin{aligned}
R_T &= W_c * R_c + W_s * R_c * R_s + W_a * R_c * R_a && (5.15)\\
R_T &= 0,669 * 0,9 && (5.16)\\
&\quad +0,22 * 0,9 * 0,07 && (5.17)\\
&\quad +0,109 * 0,9 * 0,61 && (5.18)\\
R_T &= 0,675 && (5.19)
\end{aligned}
$$

Die zu vielen und zu wenigen Wortlängen beeinflusse also negativ die Erkennungsrate der zeitlichen Filterung für falsche Schriftzeichen. Aus der Messungen der Erkennungsrate aus dem realen Video mit 69,12% können damit die Messung der künstlichen Wörter in Abschnitt 5.5.6, 5.5.6 und 5.5.6 mit 67,5% bestätigen.

Der Vergleich zu Verfahren zum Mitteln der Bilder über die Zeit und anschließender Texterkennung ist schwierig. Für einen direkten Vergleich sind Experimente auf den gleichen Videodaten nötig. Da für diese Arbeit die exakten Verfahren und Vergleichsvideos nicht zur Verfügung stehen, sind nur indirekte Vergleiche möglich. In dem Verfahren, veröffentlicht in [51], wurde auf einzelnen Textblöcken eine Erkennungsrate von 53% erreicht. Das entspricht einer Fehlerrate von 47%. Durch Mitteln der Bilder der Textblöcke über die Zeit wurde eine Steigerung der Erkennung auf 88% möglich. Aus der Übereinstimmung der Messung der zeitlichen Filterung von Text auf künstlich generierten Wörtern und in einem realen Video kann aus den Kurven in Abschnitt 5.5.6 für falsche Schriftzeichen, 5.5.6 für fehlende Schriftzeichen und 5.5.6 für zu viele Schriftzeichen ein Vergleich abgelesen werden. Der Fehler muss dazu proportional aufgeteilt werden. Für R_c kann 0,99, für R_s kann 0,99 und für R_a kann 0,78 abgelesen werden. Für eine Erkennungsrate von 53% bzw. eine Fehlerrate von 47% ist eine Steigerung auf ca. 0,964 bzw. 96,4% zu erwarten.

$$
\begin{aligned}
R &= W_c * R_c + W_s * R_c * R_s + W_a * R_c * R_a && (5.20)\\
R &= 0,669 * 0,99 + 0,222 * 0,99 * 0,99 + 0,109 * 0,99 * 0,78 && (5.21)\\
R &= 0,964 && (5.22)
\end{aligned}
$$

5.5.8 Messung der Rechenzeit für die Filterung

Sind die Wörter durch die Erkennung von Text und durch die Zugehörigkeit der Wörter über mehrere Bilder in dem Video durch die Verfolgung von Text vorhanden, wird die zeitliche Filterung ausgeführt. Als Testprocessor diente hier eine Intel Quad Core mit 3 GHz Taktfrequenz. Durch das Multitasking Betriebssystem Windows 7 und deren Nebenläufig eit der Prozesse kann für die Rechenzeit nur eine Schätzung angegeben werden. In allen Zeitmessungen über die von der Erkennung von Text gefunden 68.134 Wörter mit 923 zu fusionierenden

Wörtern wurde jeweils eine Rechenzeit für die zeitliche Filterung von unter einer Minute gemessen.

5.5.9 Messung der zeitlichen Filterung der Texterkennung in einem realen Video unterschiedlicher Bitraten

Die Erkennung von Text in Videos ist stark abhängig von der Qualität des Videomaterials. Sinkt die Anzahl der zur Verfügung stehenden Daten für einzelne Bilder in einem Video, auch Bitrate genannt, veringert sich auch die Möglichkeit den Text in diesen Bildern zu erkennen. Die in dieser Arbeit entwickelte zeitliche Filterung zur Verbesserung der Texterkennung wird dadurch ebenfalls beeinflußt Für eine Auswertung des Verhaltens der Erkennung von Text und der anschließenden zeitlichen Filterung bei sinkender Bitrate wird das Video in dieser Arbeit mit unterschiedlichen Bitraten neu kodiert. Für jedes Video wird das Experiment für die Texterkennung und die zeitliche Filterung wiederholt.

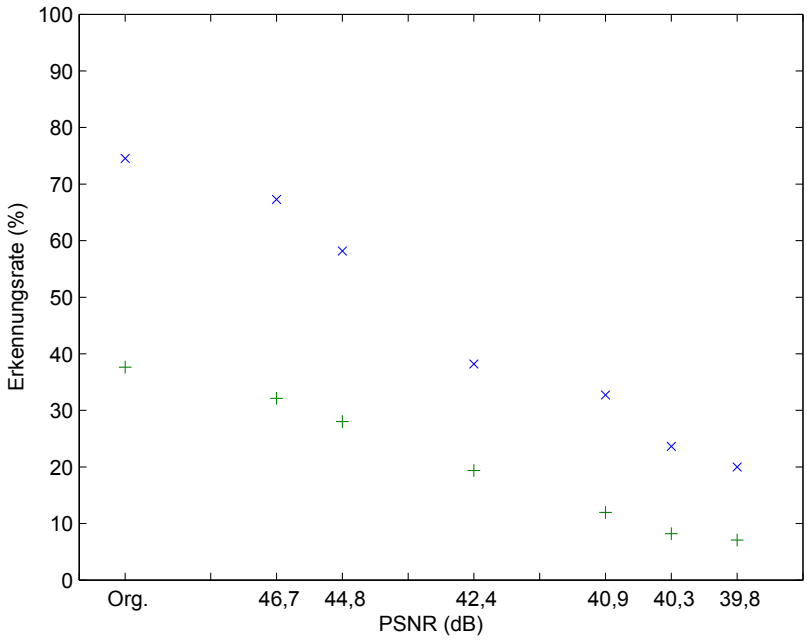


Abbildung 5.28: *In dem Diagramm sind die Ergebnisse der Texterkennung (grüne Kreuze) und die Ergebnisse der zeitlichen Filterung (blaue X) für die selben Bitraten übereinander aufgetragen.*

In Abbildung 5.28 sind die Ergebnisse der Texterkennung (grüne +) und die Ergebnisse der zeitlichen Filterung (blaue x) für die selben Bitraten des Videos übereinander aufgetragen. Auf der rechten Seite ist mit 203 kBit/s das Experiment mit dem Orginalvideo durchgeführt. Von rechts nach links sind die Bitraten für das Video Schritt für Schritt weiter reduziert. Die jeweiligen Experimente für die verschiedenen Bitraten für die Erkennung von Text und die zeitliche Filterung zeigen, dass die zeitliche Filterung in jedem Fall eine deutlich höhere Erkennung besitzt. Mit einer Bitrate von nur noch 110 kBit/s wird bei der Erkennung von

Text eine Erkennungsrate von knapp über 7% erreicht. Die zeitliche Filterung erreicht bei dieser Bitrate noch eine Erkennung von 20%.

6. Zusammenfassung und Ausblick

In dieser Arbeit ist ein komplettes Verfahren zum Detektieren und Erkennen von Text beschrieben. Prinzipiell kann hier zwischen nachträglich eingefügtem Text in Bildern und schon in der Szene enthaltenem Text unterschieden werden. Beide Fälle werden hier erkannt. Hierbei spielen der Kontrast, die Helligkeit, die Rotation, die Skalierung und die Deformationen von Text eine Rolle und haben Einflus auf die Genauigkeit der Detektion und Erkennung. Neben den verschiedenen Arten von Text in Bildern spielt vor allem auch die Qualität der Bilder oder Videos eine große Rolle. Durch die begrenzte Datenmenge zum Speichern von Videos wird hier oft eine Komprimierung der Daten vorgenommen, welche zu Einbußen in der Qualität führt. Auf diese Weise entsteht ein Fehler in den Ergebnissen der Erkennung von Test. Aus der Analyse der Fehlerquellen in der Erkennung von Text wird in dieser Arbeit eine Methode zur Reduktion von Fehlern entwickelt.

Im Ersten Schritt des kompletten Systems wird versucht die Textposition in einem Video zu finden Dabei wird ein Wort, welches in vielen aufeinanderfolgenden Bildern in dem Video vorhanden ist, in jedem einzelnen Bild detektiert. Durch die Bewegung der Kamera bewegen sich die Wörter in den zeitlich aufeinanderfolgenden Bildern, wodurch eine Verfolgung der Position von den Wörtern notwendig wird. Hierbei entsteht in dem System eine Reihe von Fehlern. Zum einen kann in einem Bild unter Umständen ein Wort nicht gefunden werden. Zum anderen kann die Verfolgung in einem folgenden Bild scheitern. Die Gründe hierfür sind vielfältig.

Die Rotation von Text wird in dieser Arbeit nicht berücksichtigt. Für eine Verbesserung der Detektion von rotiertem Text muss hier ein entsprechendes Verfahren entwickelt werden. In Abschnitt 2.5 und 2.4 sind Fälle von rotiertem Text beschrieben. Die in Abschnitt 2.6 beschriebenen Eigenschaften von Schriftzeichen könnten hierfür eine Hilfe darstellen. Durch die Gewichtung der Bildpunkte der Schriftzeichen kann unter Umständen eine Orientierung ermittelt werden.

Für die Auswertung von automatischen Verfahren zum Detektieren und Erkennen von Text müssen die gefundenen Schriftzeichen und Wörter manuell auf Richtigkeit geprüft werden. Ein Video manuell in jedem einzelnen Bild auf Text zu durchsuchen und mit den gefunden Schriftzeichen und Wörtern zu vergleichen ist sehr zeitaufwendig. Im Rahmen dieser Arbeit ist ein Programm mit dem Namen Annotation Tool entstanden diesen Vorgang zu beschleunigen. In diesem Programm kann ein Video geladen und mit einer Reihe von Werkzeugen Objekte markiert und beschrieben werden. Die so entstandenen Markierungen, auch Annotationen genannt, können anschließend gespeichert werden und dienen in den Experimenten als Grundwahrheit. Das Programm kann für die Markierung beliebig verschiedene Objekte neben Text verwenden. Mit Hilfe dieses Programms war es möglich, in angemessener Zeit, Testvideos für die Experimente vorzubereiten.

Die Detektion von Text wird hauptsächlich durch einen hohen Frequenzanteil in dem Bild gefunden. Hierfür wird entweder die Kantenerkennung nach Canny, die Wavelet Transformation oder MPEG Verfahren genutzt. Befinde sich aber gleichzeitig Objekte mit ähnlichen Frequenzanteilen, ähnlicher Größe und Form in dem Bild ist eine Differenzierung

über diese Verfahren nicht möglich. Das Experiment zur Detektion in dieser Arbeit erreicht in einem Testvideo eine Erkennungsrate R_d von 35,09% und besitzt dabei eine Präzision P_d von 86,25%. Die gefundenen Bereiche können von den per Hand markierten Bereichen abweichen, obwohl derselbe Text enthalten ist. Hierfür wurde eine minimale Überdeckung von 70% angenommen. Wird die minimale Überdeckung auf 50% reduziert ergeben sich die Erkennungsrate R_d von 37,62% mit einer Präzision P_d von 96,42%. Die Ergebnisse zeigen, dass die eingestellten Parameter für die Detektion von Text viele Textstellen übersehen, dagegen aber wenig falsche Stellen als Text markieren. Werden die Parameter dahin verändert, dass mehr Text gefunden wird, werden automatisch auch mehr falsche Stellen als Text gefunden. Um hier eine genauere Detektion der Textstellen mit gleich bleibender Präzision zu erreichen müssten die anderen Objekte ebenfalls interpretiert werden können. Neben den Schriftzeichen müssten dann zum Beispiel Autos, Gesichter und vieles mehr als solcher erkannt werden. Der Mensch kann hier als Vorbild dienen. Er erkennt solche Objekte in einem Bild und sortiert sie vor dem Lesen aus. Dagegen werden heute in den Verfahren solche Objekte als Schrift interpretiert, was zu einem Fehler führt.

Die Verfolgung von Textblöcken in den zeitlich aufeinanderfolgenden Bildern wird in dem hier vorgestellten System durch die Ähnlichkeit der Bildpunkte und der örtlichen Nähe vollzogen. Weichen die Daten weiter ab als die voreingestellten Parameter zulassen, versagt die Verfolgung. Die Folge aus diesem Fehler sind mehrere Listen statt einer Liste von Textmarkierungen. Dadurch reduziert sich die Anzahl der Ergebnisse der Texterkennung für eine Liste. Für die zeitliche Filterung, die den Fehler in der Texterkennung reduzieren kann, ist die Anzahl der Ergebnisse der Texterkennung entscheidend. Umso mehr Ergebnisse vorliegen, umso besser kann der Fehler reduziert werden. Durch die Reduzierung der Ergebnisse der Texterkennung für eine Liste wird damit auch das Ergebnis der zeitlichen Filterung schlechter.

In dieser Arbeit wurde ebenfalls ein Teilsystem zum Erkennen von Text betrachtet. Hierfür dienten Hidden Markov Modelle als Klassifikations erfahren der Schriftzeichen. Für jedes zu erkennende Schriftzeichen ist ein Modell vorhanden. Das Training jedes Modells bedarf einer großen Menge an ausgewählten Beispielen an Bildern für das entsprechende Schriftzeichen. Entscheidend dabei ist die Schriftzeichenart. Es gibt sehr viele verschiedene Formen für ein Schriftzeichen. Ist die zu erkennende Form nicht trainiert worden, kann eine Erkennung fehlschlagen. Da ein vollständiges Training aller bekannten Schriftzeichenarten den zeitlichen Rahmen dieser Arbeit übersteigt, wird hier im Weiteren ein kommerziell erhältliches Programm mit dem Namen LEADTOOLS OCR SDK zum Erkennen von Text verwendet.

Zur Bestimmung der Zuverlässigkeit des Programms LEADTOOLS OCR SDK wurde dasselbe Video verwendet, mit dem die Experimente zur Bestimmung der Zuverlässigkeit der Textdetektion bestimmt wurden. Als Basis für die Texterkennung dienten an dieser Stelle statt der automatisch detektierten Textmarkierungen die manuell erstellten Textmarkierungen. Für die Auswertung der Zuverlässigkeit ist als erstes die Erkennungsrate der einzelnen Schriftzeichen und als zweites die Erkennungsrate der gesamten Wörter zu berücksichtigen. Während ein Schriftzeichen richtig oder falsch erkannt werden kann, müssen in einem Wort für eine korrekte Erkennung alle Schriftzeichen richtig sein. Die Erkennungsrate R_s der Schriftzeichen liegt in dem Experiment bei 19,05%. In einigen Fällen kann es bei der Erkennung von einem Schriftzeichen zu einer Teilung in zwei Schriftzeichen kommen, welches dann zu einem falschen Ergebnis führt. Genauso kann es zu einer Erkennung von einem Schriftzeichen

aus zwei verschiedenen zu erkennenden Schriftzeichen kommen. Beispiele wären hierfür das Teilen des Schriftzeichen „m“ in „n“ und „i“ oder das Kombinieren der Schriftzeichen „l“ und „i“ zu „h“. Daher muss bei diesem Experiment eine Präzision P_s angegeben werden, die hier bei 90,42% liegt. Die Erkennungsrate R_w der Wörter liegt mit 39,82% deutlich über der Erkennungsrate der einzelnen Schriftzeichen. Daraus lässt sich schließen, dass die Fehler bei der Erkennung von den Schriftzeichen verstärkt in den falsch erkannten Wörtern enthalten sind. Mit knapp 40% der zu erkennenden Wörter ist nur $2/5$ vom Text gefunden worden. Die 40% der erkannten Wörter müssen nun durch eine Nachbearbeitung erhöht werden.

Insgesamt ist die Texterkennung mit knapp 40% erkannten Schriftzeichen in diesem Video sehr niedrig. Ein Experiment auf künstlich generierten Bildern ohne Datenverluste mit verschiedenen Schriftzeichenarten zeigt, dass verschiedene Schriftzeichenarten wie „Arial“ oder „MS Sans Serif“ in den Modellen des Programms LEADTOOLS OCR SDK gut trainiert wurden. Dagegen sind andere Schriftzeichenarten wie „Courier“ oder „Segoe Ul“ schlechter trainiert. Daraus lässt sich schließen, dass für eine automatische Texterkennung durch immer neue Schriftzeichenarten Grenzen vorhanden sind. In der Literatur gibt es neben dem Ansatz des Modells zur Erkennung von Schriftzeichen auch Ansätze diese als Graphen zu interpretieren. In den Publikationen [53–56] wird diese Thematik aufgegriffen.

Eine Möglichkeit die Erkennungsrate von knapp 40% zu verbessern ist ein Vergleich mit einem Wörterbuch. Ist ein Wort nicht vorhanden, kann dieses entweder gelöscht oder durch ein enthaltenes Wort mit Ähnlichkeit ersetzt werden. Hierbei besteht das Risiko, dass ein nicht enthaltenes Wort, welches dennoch korrekt ist, verloren geht oder ein falsch erkanntes Wort durch ein anderes falsches Wort ersetzt wird. Viele Verfahren versuchen daher die Bilder für die Texterkennung zu verbessern. Dazu wird meist ein Bild von einem Wort über viele aufeinanderfolgende Bilder gemittelt. Aber auch hier ist, selbst auf perfekten Bildern, der Erfolg durch den Fehler in der anschließenden Texterkennung limitiert. Grundlage für die Nachbearbeitung in dieser Arbeit sind die einzelnen Ergebnisse der Texterkennung für immer dasselbe Wort in den zeitlich aufeinanderfolgenden Bildern. In dieser Liste der Wörter können korrekte Lösungen enthalten sein. Dennoch ist nicht immer klar, welches das gewünschte Ergebnis ist. In dieser Arbeit wird angenommen, dass die entstandenen Fehler gleich verteilt auf alle Schriftzeichen sind. Damit ist in jedem Ergebnis ein Teil der Lösung enthalten. Durch Mitteln der Schriftzeichen könnte damit die Lösung gefunden werden. Die Schwierigkeit hierbei liegt in der unterschiedlichen Wortlänge der Ergebnisse aus der Texterkennung für immer dasselbe Wort. Aus diesem Grund können die Positionen in den Wörtern nicht direkt untereinander verglichen werden. An dieser Stelle wird in dieser Arbeit mittels der Levenshtein Matrix ermittelt, wie die Wörter ineinander überführt werden können. Dadurch ist es möglich, eine Aussage über die Position der Schriftzeichen der einzelnen Ergebnisse zu allen anderen Schriftzeichen der anderen Ergebnisse zu gewinnen. Durch kontinuierliches Integrieren der Wörter in eine Datenstruktur können die Wahrscheinlichkeiten für ein Schriftzeichen an jeder Position ermittelt und ein repräsentatives Wort für die Liste gefunden werden. Dieses Verfahren kann damit auch als zeitliche Filterung angesehen werden.

Auf Grundlage der Annahme, dass der Fehler gleich verteilt ist, wurden im Rahmen dieser Arbeit umfangreiche Messungen für die Zuverlässigkeit der zeitlichen Filterung für fest definiert Rahmenbedingungen vorgenommen. Der Fehler ist hier in die 3 Kategorien falsche Schriftzeichen, fehlende Schriftzeichen und zu viele Schriftzeichen gegliedert. Für jede einzelne Messung wird ein künstliches Wort mit zufälligen Schriftzeichen generiert. Ausgehend von diesem Wort werden je nach Messung eine Reihe von Wörtern mit entsprechender Feh-

lermenge generiert, die zur zeitlichen Filterung verwendet werden. Das Ergebnis der zeitlichen Filterung wird dann mit dem anfänglich generierten Wort verglichen. Durch eine große Menge von Wiederholungen dieses Vorgangs kann eine prozentuale Erkennungsrate für diesen Fall ermittelt werden. Entscheidend sind hier die Untersuchungen zu der Trefferrate bei zunehmenden Fehler bzw. zunehmender Länge der Liste für ein Wort. Durch Variation dieser zwei Parameter kann eine dreidimensionale Messkurve gezeichnet werden. Ein weiterer Faktor für die Bestimmung der Zuverlässigkeit ist in den Experimenten die Anzahl der unterschiedlichen zu erkennenden Schriftzeichen. Für eine Schriftzeichenmenge von 61 ergibt sich bei einer Wortlänge von 5 und nach 150 integrierten Wörtern mit einer Fehlerrate der Schriftzeichen von 80% falschen Schriftzeichen eine Erkennungsrate von 90,3%. Bei gleichen Bedingungen ergibt sich bei einer Fehlerrate von 20% zu wenigen Schriftzeichen eine Erkennungsrate von 84,9% und von 20% zu vielen Schriftzeichen eine Erkennungsrate von 85,0%. Die jeweilige Fehlerrat der Schriftzeichen bezieht sich dabei auf jedes integrierte Wort. Das bedeutet bei einer Fehlerrate von 80%, dass in jedem integrierten Wort 4 von 5 Schriftzeichen falsch sind. Auf diese Weise wird durch die ermittelten Erkennungsraten zu welcher Chance aus nur falschen Wörtern das richtig ermittelt werden kann.

Die Annahme der Gleichverteilung und der daraus resultierenden Ergebnisse wird in dieser Arbeit durch die Anwendung der zeitlichen Filterung auf ein reales Video getestet. Hierfür wird wieder das gleiche Testvideo aus der Detektion und Erkennung von Text verwendet. Durch die Texterkennung in diesem Video werden auf den einzelnen Bildern 39,8% der Wörter gefunden. Nach der Anwendung der zeitlichen Filterung ergibt sich eine Erkennungsrate R_z von 69,1%.

Die in dem Experiment zu erkennende durchschnittliche Wortlänge beträgt 6,05. Von den automatisch erkannten Wörtern haben 22,2% eine zu geringe Wortlänge mit durchschnittlich 2,07 zu wenig Schriftzeichen. 10,9% haben dagegen eine zu große Wortlänge mit durchschnittlichen 1,95 zu vielen Schriftzeichen. 66,9% der erkannten Wörter hatten die gleiche Länge. Außerdem kam durchschnittlich jedes zu erkennende Wort in 185,38 Bildern vor. Damit konnte für die zeitliche Filterung ebenfalls durchschnittlich 185 einzelne Ergebnisse der Texterkennung verwendet werden. Die Messung der Erkennungsrate in einem realen Video kann mit den Messungen in Abbildung 5.7, 5.13 und 5.19 verglichen werden. Die durchschnittliche Wortlänge in den Experimenten auf dem realen Video liegt mit ca. 6 nahe bei der Wortlänge 5 von den 3 Messungen. Zu erkennen waren in allen Messungen 61 Schriftzeichen mit „A“-„Z“„a“-„z“ und „0“-„9“. In der Messung 5.7 für falsche Schriftzeichen wurden bei 150 integrierten Wörtern mit einem durchschnittlichen Fehler von 80% je Schriftzeichen eine Erkennungsrate von über 90% erreicht. In der Messung in dem realen Video kommt es aber in den integrierten Wörtern nicht nur zu falschen Schriftzeichen, sondern auch zu fehlenden und zu vielen Schriftzeichen. Für die 22,2% der Wörter mit durchschnittlicher 2 zu wenigen Schriftzeichen kann aus der Abbildung 5.13 eine Erkennungsrate von ca. 50% abgelesen werden. Für die 10,9% der Wörter mit durchschnittlicher 2 zu vielen Schriftzeichen kann aus der Abbildung 5.19 eine Erkennungsrate von ca. 65% abgelesen werden. Die zu vielen und zu wenigen Wortlängen beeinflusse also negativ die Erkennungsrate der zeitlichen Filterung für falsche Schriftzeichen. Werden die unterschiedlichen Erkennungsraten gewichtet gemittelt ergibt sich eine theoretische Erkennung R_T von 0,675 bzw 67,5%. Die Messungen der Erkennungsrate aus dem realen Video mit 69,1% können damit die Messung der künstlichen Wörter in Abschnitt 5.5.6 für falsche Schriftzeichen, 5.5.6 für fehlende Schriftzeichen und 5.5.6 für zu viele Schriftzeichen mit 67,5% bestätigen.

Für einen direkten Vergleich müssten beide Verfahren auf den selben Videos angewendet werden. Es liegt nahe für diese beiden Verfahren und weitere in der Zukunft eine Datenbank mit annotierten Videos zu erstellen. Damit könnte jedes Verfahren auf die Videos angewendet werden und die Ergebnisse direkt gegenübergestellt werden. Das im Rahmen dieser Arbeit entwickelte Annotation Tool kann für eine Erstellung einer annotierten Videodatenbank verwendet werden.

In Klassifikations erfahren wird das Modell mit der höchsten Übereinstimmung als Treffer gewählt. Werden alle Ergebnisse der Modelle miteinander verglichen und normiert, kann eine Aussage über die Präzision für ein Schriftzeichen angegeben werden. Ist dieser Wert für jedes Schriftzeichen bekannt, ist eine Verbesserung der zeitlichen Filterung möglich. Auf diese Weise können die Schriftzeichen in die Datenstruktur mit entsprechender Gewichtung integriert werden.

A. Beispiel-Sequenzen aus dem realen Video

In diesem Appendix sind aus dem Video verschiedene charakteristische Sequenzen ausgewählt

- Keine Text im Hintergrund
- Die Kamera wechselt von dem Sprecher zu der Präsentation
- Die Kamera wechselt von der Präsentation zum Sprecher
- Wechsel des Textes im Hintergrund ohne Bewegung der Kamera
- Zeitlich verdeckter Text im Hintergrund durch eine Person
- Abspann mit Bewegung in der Vertikalen
- Text begleitet durch Formeln und Bilder

Frame 6890

Frame 6900

Frame 6910

Frame 6920

Abbildung A.1: *Teil 1: In den Frames sind bis auf einen eingeblendeten Vordergrundtext, keine Texte vorhanden. Ein Großteil des Videos hat diese Form.*

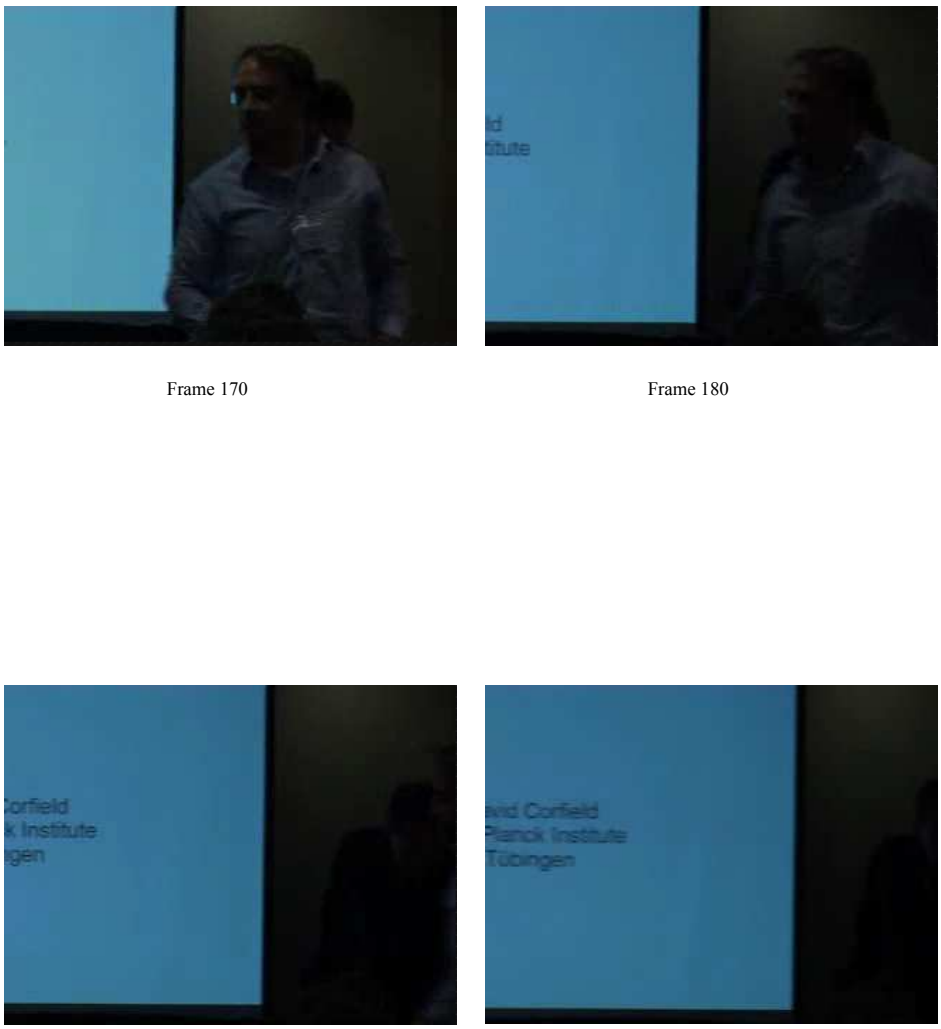
Frame 6930

Frame 6940

Frame 6960

Frame 6970

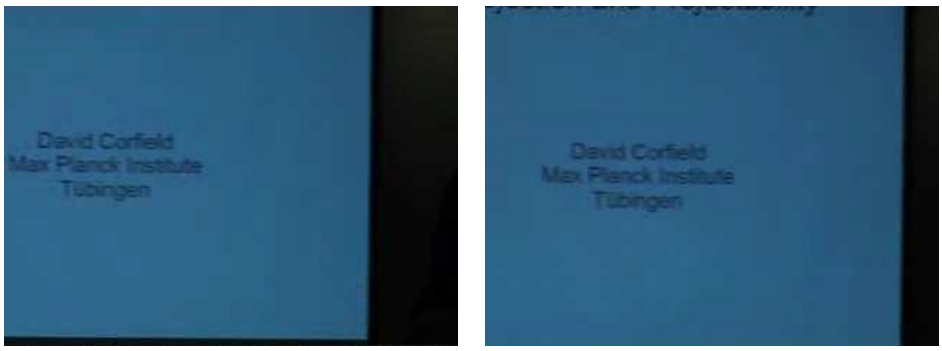
Abbildung A.2: *Teil 2: In den Frames sind bis auf einen eingeblendeten Vordergrundtext, keine Texte vorhanden. Ein Großteil des Videos hat diese Form.*



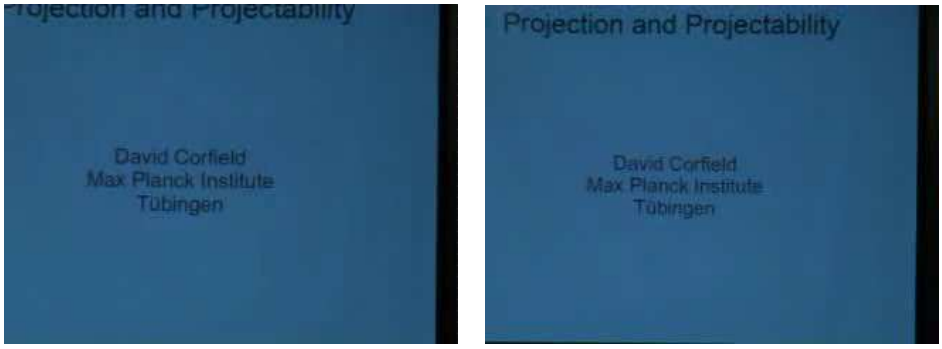
Frame 170 Frame 180

Frame 190 Frame 200

Abbildung A.3: *Teil 1: Die Kamera wechselt von dem Sprecher zu der Präsentation. Wärend der Bewegung wird das Bild unscharf und die Erkennung von Text wird erschwert.*



Frame 210 Frame 220

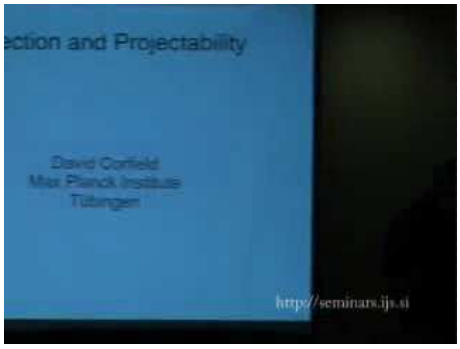


Frame 230 Frame 240

Abbildung A.4: *Teil 2: Die Kamera wechselt von dem Sprecher zu der Präsentation. Wärend der Bewegung wird das Bild unscharf und die Erkennung von Text wird erschwert.*

Frame 2740

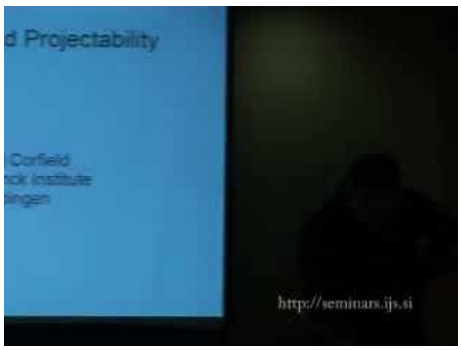
Frame 2750



Frame 2760

Frame 2770

Abbildung A.5: *Teil 1: Die Kamera wechselt von der Präsentation zum Sprecher. Wärend der Bewegung wird das Bild unscharf und die Erkennung von Text wird erschwert.*



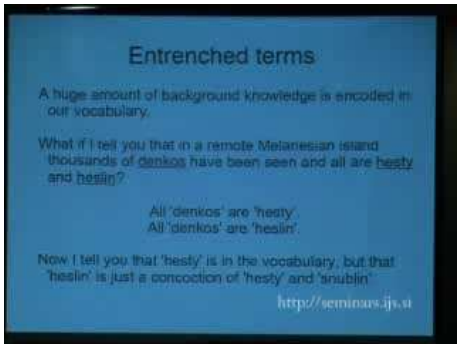
Frame 2780

Frame 2790

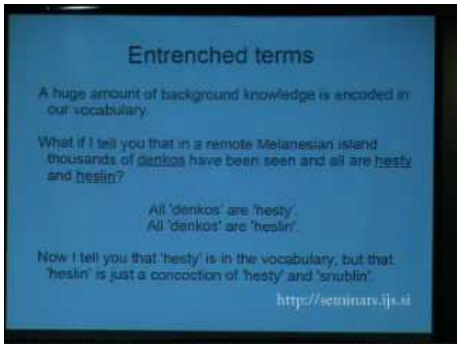
Frame 2800

Frame 2810

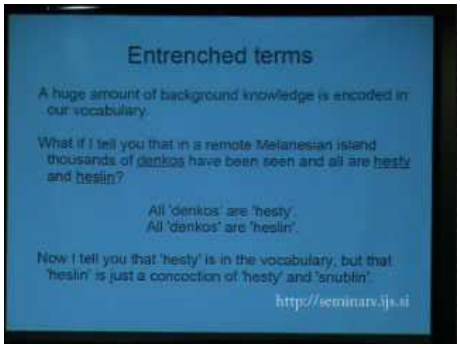
Abbildung A.6: *Teil 2: Die Kamera wechselt von der Präsentation zum Sprecher. Wärend der Bewegung wird das Bild unscharf und die Erkennung von Text wird erschwert.*



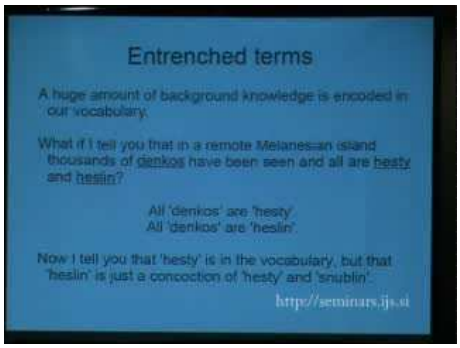
Frame 18250



Frame 18260

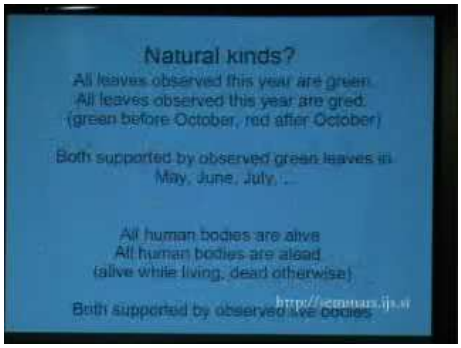


Frame 18270



Frame 18280

Abbildung A.7: *Teil 1: Wärend die Kamera die Präsentation filmt wechselt die gezeigte Folie, was zu Problemen in der Detektion von Text führen kann.*



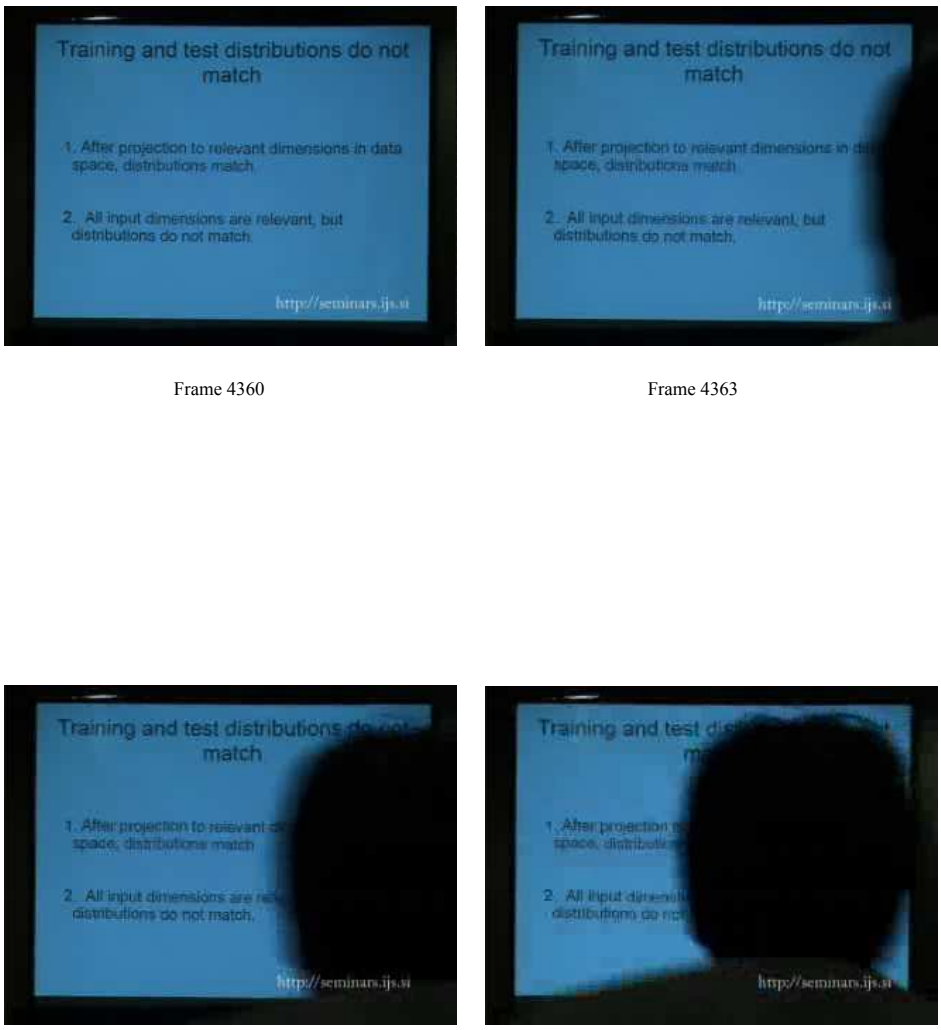
Frame 18290

Frame 18300

Frame 18310

Frame 18320

Abbildung A.8: *Teil 2: Wärend die Kamera die Präsentation filmt wechselt die gezeigte Folie, was zu Problemen in der Detektion von Text führen kann.*



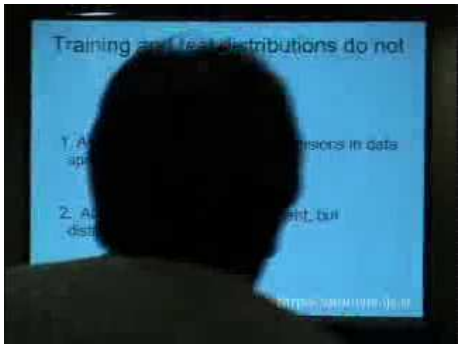
Frame 4360 Frame 4363

Frame 4366 Frame 4369

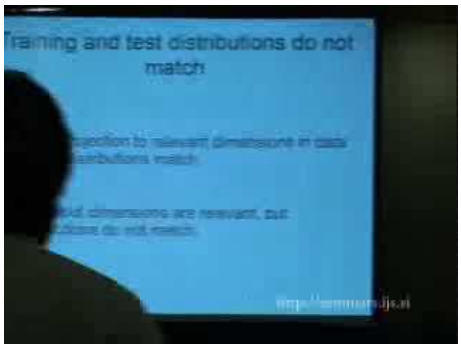
Abbildung A.9: *Teil 1: Wärend die Kamera die Präsentation filmt läuft eine Person durch das Bild. Durch die temporäre Verdeckung kann es zu Fehlern in der Detektion und Erkennung des Textes kommen.*

Frame 4372

Frame 4375



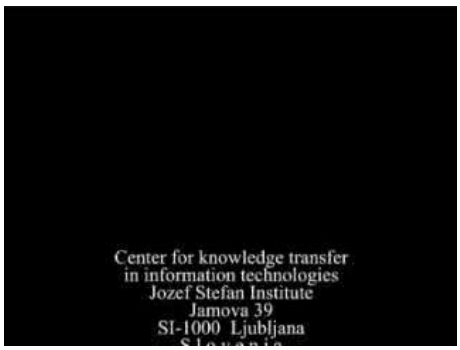
Frame 4378



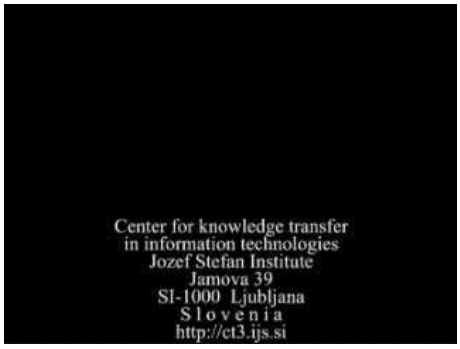
Frame 4400

Abbildung A.10: *Teil 2: Wärend die Kamera die Präsentation filmt läuft eine Person durch das Bild. Durch die temporäre Verdeckung kann es zu Fehlern in der Detektion und Erkennung des Textes kommen.*

Frame 45550



Frame 45560



Frame 45570

Frame 45580

Abbildung A.11: *Teil 1: Am Ende des Videos wird ein Abspann mit vielen Namen gezeigt. Die Namen haben viele Sonderzeichen, die nicht in der Englischen Sprache vorhanden sind und somit zu Fehlern in der Erkennung führen können.*

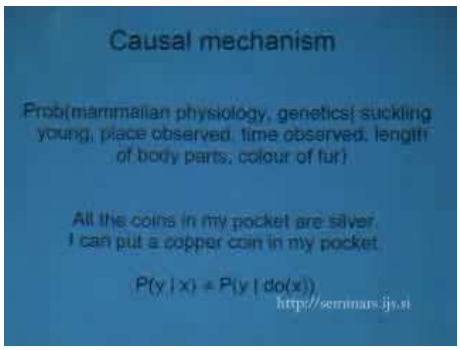
Frame 45590

Frame 45600

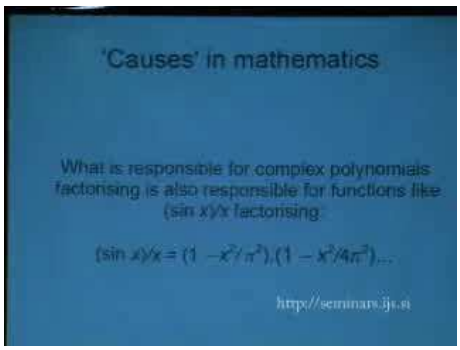
Frame 45610

Frame 45620

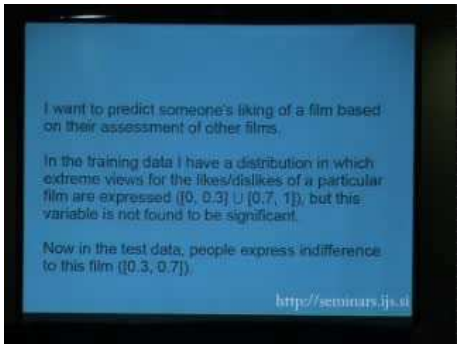
Abbildung A.12: *Teil 2: Am Ende des Videos wird ein Abspann mit vielen Namen gezeigt. Die Namen haben viele Sonderzeichen, die nicht in der Englischen Sprache vorhanden sind und somit zu Fehlern in der Erkennung führen können.*



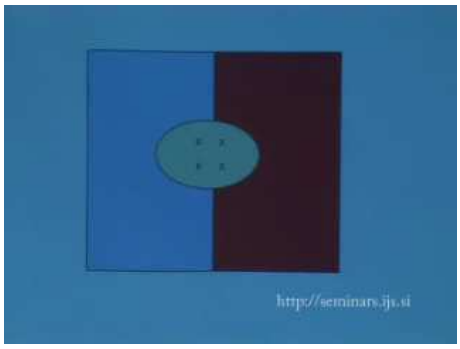
Frame 22830



Frame 23940



Frame 29380



Frame 34140

Abbildung A.13: *In einigen Folien in der Präsentation kommen Formeln vor. Diese können nur schwer als Text interprätiert werden und führen oft zu Fehlern in der Erkennung.*

B. Liste aller Wörter im realen Video

Im Folgendem sind alle in dem Video enthaltenen Wörter aufgelistet. Neben dem ersten und dem letzten Vorkommen im Video wurden für die Messung der Zuverlässigkeit des Verfahren ebenfalls die Koordinaten der Wörter in jedem Bild gespeichert. Die Menge der Koordinatendaten lässt eine Auflistun dieser Informationen in dieser Tabelle nicht zu.

Nummer	Wort	Start-Frame	End-Frame
1	Institute	189	417
2	Corfiel	196	414
3	Tübingen	203	410
4	Planck	203	408
5	David	207	406
6	Max	212	402
7	Projection	236	387
8	and	236	408
9	Projectability	236	415
10	http://seminars.ijs.si	369	518
11	DAVID	519	668
12	CORFIELD	519	668
13	NIPS	519	668
14	06	519	668
15	WHISTLER	519	688
16	CANADA	519	688
17	http://seminars.ijs.si	669	45467
18	Institut	2457	2789
19	Corfiel	2462	2783
20	Tübingen	2469	2777
21	Planck	2470	2778
22	David	2475	2775
23	Max	2492	2769
24	Projection	2535	2749
25	and	2535	2777
26	Projectability	2535	2789
27	data	3252	3728
28	in	3258	3723
29	but	3268	3710
30	dimension	3281	3695
31	relevant	3284	3692
32	are	3289	3687
33	do	3292	3693
34	not	3292	3693
35	distributions	3292	3686

Tabelle B.1: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
36	match	3293	3683
37	match	3293	3683
38	match	3295	3683
39	relevant	3293	3683
40	to	3297	3679
41	test	3299	3677
42	not	3300	3676
43	do	3305	3672
44	dimension	3307	3670
45	and	3309	3668
46	distributions	3315	3665
47	projection	3315	3665
48	input	3319	3663
49	All	3326	3660
50	After	3328	3659
51	distributions	3334	3657
52	space	3337	3657
53	1	3339	3655
54	2	3339	3655
55	Training	3346	3656
56	data	4064	4117
57	in	4070	4363
58	but	4079	4364
59	dimensions	4094	4364
60	relevant	4097	4366
61	are	4104	4366
62	match	4109	3667
63	match	4109	4366
64	relevant	4109	4365
65	to	4116	4368
66	not	4119	4369
67	do	4133	4372
68	dimensions	4139	4367
69	distributions	4155	4368
70	projection	4162	4369
71	input	4174	4374
72	All	4187	4376
73	After	4190	4375
74	distributions	4199	4368
75	space	4202	4375

Tabelle B.2: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
76	match	4202	4367
77	1	4206	4385
78	2	4206	4385
79	data	4217	4362
80	Training	4231	4385
81	and	4231	4372
82	test	4231	4370
83	distributions	4231	4372
84	do	4231	4366
85	not	4231	4364
86	not	4371	4448
87	do	4373	4448
88	data	4374	4468
89	in	4375	4464
90	but	4376	4457
91	distributions	4380	
92	test	4380	4426
93	and	4380	4387
94	dimensions	4384	4446
95	relevant	4384	4443
96	are	4388	4437
97	match	4389	4432
98	match	4389	4434
99	and	4392	4417
100	match	4492	4434
101	relevant	4392	4433
102	not	4392	4426
103	Training	4393	4400
104	to	4395	4429
105	dimensions	4398	4415
106	distributions	4403	4409
107	all	5103	5622
108	extrem	5110	5607
109	I	5105	5109
110	enough	5113	5593
111	an	5113	5593
112	matters	5116	5583
113	in	5118	5580
114	it	5119	5578

Tabelle B.3: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
115	Canada	5121	5461
116	to	5125	5572
117	broadly	5126	5570
118	of	5130	5577
119	place	5129	5568
120	problem	5131	5566
121	whether	5132	5565
122	been	5133	5563
123	before	5133	5456
124	observation	5134	5258
125	taken	5135	5561
126	form	5135	5561
127	I	5136	5454
128	this	5136	5559
129	observation	5138	5274
130	match	5139	5557
131	idea	5139	5557
132	are	5140	5555
133	features	5141	5553
134	never	5141	5553
135	time	5144	5549
136	good	5145	5547
137	I’ve	5146	5546
138	a	5148	5545
139	confronts	5149	5543
140	object	5150	5543
141	and	5150	5543
142	test	5150	5553
143	distributions	5150	5563
144	do	5150	6511
145	not	5150	5621
146	I	5151	5454
147	variables	5152	5540
148	have	5154	5536
149	though	5156	5535
150	input	5159	5531
151	learning	5161	5530
152	P—class	5162	5529
153	if	5162	5528
154	Even	5163	5527
155	Training	5166	5524

Tabelle B.4: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
156	observation	5274	5496
157	observation	5383	5584
158	Canada	5467	5498
159	Canada	5525	
160	observation	5528	5561
161	before	5533	5610
162	I	5533	5627
163	many	10376	10881
164	evidance	10382	10881
165	generalization	10382	10881
166	green	10385	10879
167	the	10386	10880
168	green	10387	10879
169	green	10387	10879
170	have	10388	10871
171	have	10388	10878
172	observed	10390	10877
173	is	10390	10877
174	is	10390	10877
175	are	10390	10877
176	a	10391	10875
177	b	10391	10875
178	thus	10393	10873
179	etc	10393	10873
180	support	10394	10871
181	We	10396	10869
182	we	10397	10867
183	t	10399	10866
184	emeralds	10399	10866
185	emerald	10400	10865
186	emerald	10400	10865
187	All	10402	10863
188	time	10404	10860
189	green	10404	10861
190	statements	10406	10858
191	at	10408	10856
192	be	10408	10856
193	Induction	10410	10864
194	to	10412	10852
195	that	10414	10849

Tabelle B.5: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
196	these	10415	10849
197	and	10423	10842
198	emeralds	10431	10833
199	statements	10432	10833
200	Suppose	10435	10830
201	Nelson	10442	10831
202	Goodman's	10442	10854
203	New	10442	10878
204	Riddle	10442	10881
205	of	10442	10881
206	to	10882	10886
207	apply	10882	10889
208	to	10882	10900
209	all	10882	10905
210	case	10882	10883
211	they	10882	10890
212	are	10882	10899
213	case	10882	10884
214	they	10882	10891
215	are	10882	10900
216	statements	10882	10889
217	support	10882	10886
218	the	10882	10899
219	are	11560	12374
220	the	11561	11728
221	they	11567	12366
222	statements	11569	12365
223	support	11572	11732
224	case	11574	12359
225	in	11577	12356
226	grue	11578	12356
227	grue	11578	12356
228	evidence	11581	12353
229	is	11581	12353
230	is	11581	12353
231	just	11583	12353
232	a	11584	12352
233	b	11584	12352
234	the	11586	12350
235	etc	11586	12350

Tabelle B.6: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
236	statements	11587	12350
237	things	11592	12347
238	also	11593	12356
239	hypothesis	11595	12345
240	emerald	11596	12344
241	emerald	11596	12344
242	other	11600	12342
243	have	11601	12341
244	evidence	11601	12341
245	to	11605	12339
246	we	11611	12338
247	and	11631	12335
248	these	11631	12335
249	observed	11636	12335
250	before	11636	12345
251	t	11636	12351
252	just	11636	12352
253	in	11636	12356
254	case	11636	12359
255	Then	11647	12331
256	and	11654	12330
257	now	11659	12329
258	defin	11661	12336
259	the	11661	12342
260	predicate	11661	12345
261	grue	11661	12354
262	to	11661	12361
263	things	11665	12326
264	they	11665	12365
265	green	11666	12325
266	But	11672	12322
267	blue	11675	12321
268	All	11675	12342
269	emeralds	11675	12345
270	are	11675	12353
271	grue	11675	12357
272	apply	11676	12365
273	to	11694	12374
274	are	11701	12373
275	all	11712	12377

Tabelle B.7: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
276	support	11744	12362
277	the	11750	12373
278	if	12855	12895
279	grue	12857	13039
280	and	12862	13561
281	t	12864	13559
282	t	12864	13045
283	t	12875	13550
284	to	12877	13547
285	thereafter	12877	13557
286	up	12881	13544
287	blue	12886	13552
288	thereafter	12866	13557
289	before	12872	13552
290	before	12872	13552
291	t	12877	13548
292	thereafter	12877	13548
293	to	12880	13545
294	up	12883	13541
295	observed	12883	
296	observed	12884	13541
297	green	12886	13550
298	if	12886	13538
299	if	12886	13040
300	green	12888	13536
301	observed	12888	13541
302	blue	12889	13536
303	observed	12889	13540
304	thereafter	12891	13556
305	and	12891	13043
306	grue	12892	13533
308	bleen	12896	12898
309	grue	12899	13527
310	bleen	12900	13525
311	blue	12911	13520
312	green	12913	13519
313	bleen	12914	13532
314	if	12921	13038
315	if	12925	13538

Tabelle B.8: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
316	Syntactic	12932	13531
317	complexity	12932	13543
318	if	13040	13229
319	t	13048	13559
320	and	13050	13561
321	bleen	13052	13070
322	grue	13067	13229
323	if	13070	13573
324	bleen	13075	13220
325	bleen	13226	13241
326	grue	13238	13248
327	if	13238	13571
328	bleen	13246	13566
329	grue	13253	13566
330	vocabulary	14033	14458
331	background	14037	14471
332	amount	14037	14461
333	of	14037	14468
334	Huge	14038	14455
335	our	14039	14453
336	A	14041	14451
337	knowledge	14106	14485
338	Entrenched	14115	14466
339	terms	14115	14487
340	is	14115	14499
341	encoded	14160	14502
342	in	14171	14515
343	is	15768	16088
344	encoded	15768	16095
345	in	15768	16153
346	knowledge	15770	16065
347	background	15790	16049
348	of	15797	16046
349	amount	15832	16031
350	Entrenched	15835	16044
351	terms	15835	16068
352	vocabulary	15845	16026
353	huge	15855	16019
354	our	15858	16015
355	A	15864	16003

Tabelle B.9: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
356	vocabulary	16926	18285
357	our	16935	18285
358	in	16847	18285
359	hesty	16854	17471
360	are	16859	17481
361	encoded	16862	18285
362	island	16862	17493
363	all	16863	18285
364	is	16865	18285
365	and	16869	17501
366	seen	16877	17512
367	Melanesian	16879	17529
368	hesty	16879	18285
369	heslin	16879	18285
370	knowledge	16881	18285
371	been	16885	18285
372	are	16885	18285
373	are	16885	18285
374	remote	16890	18285
375	a	16893	18285
376	have	16894	18285
377	in	16896	18285
378	denkos	16898	18285
379	denkos	16898	18285
380	background	16900	18285
381	that	16903	18285
382	All	16903	18285
383	All	16903	18285
384	of	16904	18285
385	denkos	16906	18285
386	you	16910	18285
387	of	16910	18285
388	tell	16919	18285
389	amount	16921	18285
390	I	16922	18285
391	Entrenched	16923	18285
392	terms	16923	18285
393	if	16926	18285
394	heslin	16926	18285
395	huge	16933	18285

Tabelle B.10: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
396	and	16935	18285
397	thousands	16935	18285
398	A	16938	18285
399	What	16938	18285
400	Now	17396	18285
401	I	17396	18285
402	tell	17396	18285
403	you	17396	18285
404	that	17396	18285
405	hestly	17396	18285
406	is	17396	18285
407	in	17396	18285
408	the	17396	18285
409	vocabulary	17396	18285
410	but	17396	17720
411	that	17396	17702
412	heslin	17396	18285
413	is	17396	18285
414	just	17396	18285
415	a	17396	18285
416	concoction	17396	18285
417	of	17396	18285
418	hesty	17396	18285
419	and	17396	18285
420	snublin	17396	18285
421	seen	17531	18285
422	and	17541	18285
423	Melanesian	17542	18285
424	all	17543	18285
425	are	17543	18285
426	hesty	17544	18285
427	but	17733	18285
428	that	17737	17747
429	that	17759	18285
430	Natural	18286	18494
431	kinds	18286	18508
432	All	18286	18483
433	leaves	18286	18486
434	observed	18286	18495
435	this	18286	18508

Tabelle B.11: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
436	year	18286	18515
437	are	18286	18529
438	green	18286	18549
439	All	18286	18483
440	leaves	18286	18486
441	observed	18286	18495
442	this	18286	18508
443	year	18286	18515
444	are	18286	18529
445	gred	18286	18550
446	green	18286	18482
447	before	18286	18490
448	October	18286	18499
449	red	18286	18511
450	after	18286	18519
451	October	18286	18542
452	Both	18286	18480
453	supported	18286	18486
454	by	18286	18499
455	observed	18286	18503
456	green	18286	18519
457	leaves	18286	18548
458	in	18286	18565
459	May	18286	18496
460	June	18286	18503
461	July	18286	18511
462	All	18286	18490
463	human	18286	18494
464	bodies	18286	18503
465	are	18286	18514
466	alive	18286	18522
467	All	18286	18489
468	human	18286	18493
469	bodies	18286	18502
470	are	18286	18513
471	alead	18286	18521
472	alive	18286	18486
473	while	18286	18493
474	living	18286	18500
475	dead	18286	18508

Tabelle B.12: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
476	otherwise	18286	18518
477	both	18286	18482
478	supported	18286	18488
479	by	18286	18502
480	observed	18286	18506
481	live	18286	18523
482	bodies	18286	18547
483	in	19340	20620
484	leaves	19353	19825
485	October	19355	20620
486	alive	19362	20620
487	alead	19363	20620
488	after	19365	20620
489	green	19365	20620
490	otherwise	19365	20620
491	are	19370	20620
492	are	19371	20620
493	red	19373	20620
494	July	19373	20620
495	dead	19377	20620
496	observed	19384	20620
497	June	19384	20620
498	bodies	19384	20620
499	bodies	19385	20620
500	living	19388	20620
501	by	19389	20620
502	October	19390	20620
503	May	19395	20620
504	human	19398	20620
505	human	19399	20620
506	while	19400	20620
507	before	19406	20620
508	All	19406	20620
509	All	19407	20620
510	observed	19408	20620
511	this	19408	20620
512	year	19408	20620
513	are	19408	20620
514	gred	19408	20620
515	leaves	19415	20620

Tabelle B.13: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
516	alive	19415	20620
517	supported	19417	20620
518	supported	19422	20620
519	by	19422	20620
520	observed	19422	20620
521	live	19422	20620
522	bodies	19422	20620
523	all	19425	20620
524	Both	19426	20620
525	green	19428	20620
526	all	19434	20620
527	leaves	19434	20620
528	observed	19434	20620
529	this	19434	20620
530	year	19434	20620
531	are	19434	20620
532	green	19434	20620
533	Both	19441	20620
534	Natural	19463	20620
535	kinds	19463	20620
536	in	19839	20620
537	Causal	20621	20966
538	mechanism	20621	20980
539	Prob	20621	20949
540	mammalian	20621	20960
541	physiology	20621	20976
542	genetics	20621	20991
543	suckling	20621	21002
544	young	20621	20952
545	place	20621	20965
546	observed	20621	20972
547	time	20621	20986
548	observed	20621	20972
549	length	20621	21004
550	of	20621	20967
551	body	20621	20970
552	parts	20621	20977
553	colour	20621	20986
554	of	20621	20993
555	fur	20621	20996

Tabelle B.14: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
556	length	21658	22150
557	fur	21664	22143
558	of	21668	22113
559	observed	21671	22140
560	time	21679	22134
561	colour	21679	22133
562	parts	21692	22126
563	observed	21699	22120
564	body	21703	22117
565	of	21708	22141
566	place	21713	22110
567	mammalian	21734	22104
568	suckling	21653	22148
569	genetics	21672	22139
570	young	21774	22098
571	Causal	21787	22112
572	mechanism	21787	22128
573	Prob	21783	22094
574	physiology	21693	22124
575	length	22687	22727
576	suckling	22689	22726
577	silver	22694	22789
578	fur	22696	23573
579	pocket	22696	22789
580	of	22700	23570
581	are	22700	23571
582	my	22702	23570
583	observed	22703	23551
584	genetics	22704	23568
585	in	22706	23567
586	time	22712	23563
587	colour	22712	23563
588	pocket	22713	23563
589	coin	22714	23562
590	my	22720	23559
591	parts	22723	23557
592	in	22724	23556
593	physiology	22725	23555
594	copper	22727	23553
595	observed	22729	23569

Tabelle B.15: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
596	a	22730	23551
597	body	22732	23550
598	coins	22733	23548
599	of	22736	23546
600	put	22737	23546
601	place	22742	23544
602	the	22742	23544
603	$P(y\|x)$	22744	23549
604	$P(y\|do(x))$	22744	23561
605	can	22753	23541
606	All	22756	23539
607	I	22757	23538
608	mammalian	22757	23539
609	young	22783	23462
610	Causal	22789	23451
611	mechanism	22789	23451
612	Prob	22794	23455
613	length	22790	23580
614	suckling	22792	23578
615	silver	22800	23574
616	pocker	22801	23573
617	like	23802	23856
618	polynomials	23814	23860
619	for	23818	24984
620	$(1 - x^2/4\pi^2)$	23823	24889
621	complex	23826	24601
622	functions	23829	23878
623	for	23830	24596
624	responsible	23835	24592
625	$(1 - x^2/\pi^2)$	23839	24589
626	also	23843	24586
627	$(sinx)/x$	23843	24575
628	is	23846	24583
629	reponsible	23850	24582
630	is	23854	24576
631	$(sinx)/x$	23858	24586
632	What	23878	2471
633	factorising	23884	24568
634	factorising	23885	24906
635	Causes	23909	24576

Tabelle B.16: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
636	in	23909	24593
637	mathematics	23909	24597
638	functions	23897	24667
639	polynomials	23913	24438
640	like	23920	24435
641	polynomials	24442	24635
642	like	24443	24998
643	complex	24630	24668
644	for	24637	24667
645	responsible	24644	24666
646	$(1 - x^2/\pi^2)$	24649	24969
647	$(sinx)/x$	24653	24966
648	also	24655	24966
649	is	24662	24963
650	responsible	24667	24960
651	polynomials	24659	24669
652	like	24670	24671
653	functions	24673	24988
654	polynomials	24675	24988
655	like	24675	24998
656	$(sinx)/x$	24679	24955
657	is	24674	24957
658	What	24808	24948
659	factorising	24810	24947
660	factorising	24946	25353
661	$(1 - x^2/4\pi^2)$	24950	25364
662	like	26840	27116
663	polynomials	26860	27086
664	functions	26860	27086
665	for	26867	27079
666	$(1 - x^2/4\pi^2)$	26873	27074
667	for	26882	27067
668	complex	26877	27071
669	factorsing	26881	27067
670	responsible	26886	27062
671	responsible	26887	27049
672	$(1 - x^2/\pi^2)$	26891	27058
673	also	26894	27055
674	$(sinx)/x$	26895	27055
675	is	26898	27052

Tabelle B.17: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
676	is	26907	27045
677	$(sinx)/x$	26911	27042
678	What	26923	27034
679	factorsing	26931	27031
680	pixel	27714	27770
681	while	27718	27775
682	left	27722	27783
683	projected	27724	27771
684	top	27730	27795
685	pixel	27731	27793
686	pixel	27731	27792
687	the	27737	28276
688	dark	27740	28274
689	dark	27741	28273
690	matches	27742	27796
691	classificatio	27744	27791
692	fin	27745	28270
693	I	27747	28268
694	has	27748	28268
695	has	27749	28267
696	the	27751	28265
697	to	27756	28262
698	task	27757	28262
699	data	27758	28261
700	data	27759	28260
701	distribution	27759	28260
702	distribution	27764	
703	test	27767	28256
704	test	27767	28256
705	matter	27769	28254
706	training	27774	28251
707	of	27779	28248
708	of	27780	28252
709	training	27781	28247
710	recognition	27781	28247
711	20%	27784	28245
712	doens's	27789	28242
713	digit	27794	28240
714	80%	27794	28240
715	On	27890	28230

Tabelle B.18: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
716	a	27809	28237
717	Yet	27842	28235
718	Projected	27891	28230
719	Projection	27917	
720	top	27842	28284
721	pixel	27848	28282
722	pixel	27850	28283
723	classificatio	27857	28270
724	left	27869	28292
725	while	27883	28297
726	pixel	27895	28306
727	projected	27895	28290
728	pixel	28352	28674
729	while	28356	28359
730	left	28358	28674
731	projected	28361	28395
732	while	28364	28389
733	pixel	28379	28387
734	while	28393	28674
735	top	28432	28674
736	pixel	28436	28674
737	pixel	28437	28674
738	the	28450	28674
739	dark	28454	28674
740	dark	28455	28674
741	matches	28457	28674
742	fin	28459	28674
743	classificatio	28460	28674
744	I	28462	28674
745	has	28463	28674
746	has	28464	28673
747	the	28467	28670
748	task	28473	28666
749	to	28473	28666
750	data	28474	28666
751	data	28475	28665
752	distribution	28475	28664
753	distribution	28481	28661
754	test	28484	28658
755	test	28484	28658

Tabelle B.19: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
756	matter	28487	28656
757	of	28488	28655
758	training	28489	28653
759	recognition	28494	28649
760	of	28494	28650
761	training	28495	28649
762	20%	28497	28648
763	doesn't	28501	28645
764	digit	28503	28643
765	80%	28504	28643
766	a	28508	28640
767	Yet	28511	28637
768	Projection	28513	28651
769	Projected	28514	28634
770	On	28513	28634
771	of	28675	28682
772	a	28675	28687
773	fil	28675	28691
774	based	28675	28703
775	film	28675	28676
776	in	28675	28693
777	which	28675	28700
778	of	28675	28684
779	a	28675	28689
780	particular	28675	28693
781	but	28675	28691
782	this	28675	28701
783	indifference	28675	28688
784	based	29152	29562
785	which	29154	29561
786	in	29159	29555
787	particular	29159	29556
788	fil	29161	29554
789	a	29162	29553
790	indifference	29164	29447
791	a	29165	29552
792	of	29168	29550
793	of	29171	29549
794	film	29182	29542
795	express	29183	29541

Tabelle B.20: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
796	liking	29184	29542
797	distribution	29185	29541
798	significan	29188	29193
799	a	29189	29539
800	other	29195	29536
801	be	29195	29536
802	likes/dislikes	29198	29535
803	significan	29199	29540
804	of	29200	29533
805	people	29200	29534
806	have	29201	29534
807	to	29201	29534
808	I	29204	29532
809	([0, 0.3] U (0.7, 1])	29205	29532
810	but	29205	29555
811	the	29206	29531
812	this	29208	29561
813	someone's	29209	29529
814	data	29212	29528
815	for	29213	29527
816	found	29213	29527
817	data	29214	29526
818	test	29219	29522
819	not	29220	29522
820	([0.3, 0.7])	29220	29521
821	predict	29223	29519
822	is	29225	29518
823	assessment	29225	29517
824	views	29225	29519
825	expressed	29226	29517
826	the	29228	29516
827	to	29229	29515
828	fil	29230	29515
829	training	29231	29514
830	in	29234	29512
831	are	29236	29510
832	their	29238	29508
833	the	29240	29507
834	this	29240	29507
835	want	29245	29505

Tabelle B.21: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
836	extreme	29254	29502
837	In	29254	29502
838	I	29254	29502
839	on	29254	29502
840	variable	29254	29502
841	fil	29254	29502
842	Now	29254	29502
843	to	29254	29502
844	indifference	29481	29552
845	data	33160	33222
846	in	33165	33585
847	but	33176	33574
848	dimensions	33189	33546
849	relevant	33192	33543
850	are	33198	33538
851	relevant	33201	33533
852	match	33201	33534
853	match	33201	33533
854	to	33204	33530
855	not	33206	33528
856	do	33209	33524
857	dimensions	33212	33520
858	projection	33217	33513
859	distributions	33217	33514
860	input	33219	33509
861	All	33223	33502
862	After	33224	33501
863	space	33225	33498
864	distributions	33226	33499
865	1	33227	33497
866	2	33227	33497
867	match	33245	33532
868	data	33247	33590
869	Training	33257	33495
870	and	33257	33519
871	test	33257	33528
872	distributions	33257	33537
873	do	33257	33553
874	not	33257	33553
875	Center	45482	45597

Tabelle B.22: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
876	for	45482	45597
877	knowledge	45482	45597
878	tranfer	45482	45597
879	in	45487	45604
880	information	45487	45604
881	technologies	45487	45604
882	Jozef	45493	45610
883	Stefan	45493	45610
884	Institute	45493	45610
885	Jamova	45500	45616
886	39	45500	45616
887	SI-1000	45507	45622
888	Ljubljana	45507	45622
889	Slovenia	45513	45629
890	http://ct3.ijs.si	45520	45635
891	Head	45531	45649
892	Mitja	45538	45654
893	Jermol	45538	45654
894	Multimedia	45549	45666
895	team	45549	45666
896	Sebastian	45555	45672
897	Mislej	45555	45672
898	Spela	45562	45678
899	Sitar	45562	45678
900	Tina	45568	45684
901	Anzic	45568	45684
902	Davor	45573	45690
903	Orlic	45573	45690
904	Darko	45580	45696
905	Ignjatovic	45580	45696
906	Nina	45585	45701
907	Rancic	45585	45701
908	Sebastian	45592	45707
909	Smrkolj	45592	45707
910	Samuel	45597	45713
911	Kranjc	45597	45713
912	Jure	45602	45719
913	Ferlez	45602	45719
914	Drago	45609	45725
915	Treveznjk	45609	45725

Tabelle B.23: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Nummer	Wort	Start-Frame	End-Frame
916	Marjana	45615	45732
917	Plukavec	45615	45732
918	Joze	45621	45740
919	Mislej	45621	45740
920	http://seminars.ijs.si	45680	45820
921	Contacts	45692	45820
922	sebastian.mislej@ijs.si	45698	45820
923	spela.sitar@ijs.si	45704	45820

Tabelle B.24: *Auflistun aller Wörter in dem Testvideo. Einige Wörter sind in dem Video mehrfach vorhanden.*

Literaturverzeichnis

[1] C. Hurley, *At f ve years, two billion views per day and counting*, http://youtube-global.blogspot.com/2010/05/at-f ve-years-two-billion-views-per-day.html.

[2] C. Hurley, *Y,000,000,000uTube*, http://youtube-global.blogspot.com/2009/10/y000000000utube.html.

[3] N. Jurran, *YouTube: Über 1 Milliarde Videoabrufe pro Tag*, http://www.heise.de/newsticker/meldung/YouTube-Ueber-1-Milliarde-Videoabrufe-pro-Tag-821259.html.

[4] L. Goldmann, U. Moenich, and T. Sikora, *Robust Face Detection Based on Components and Their Topology*, in *Visual Communications and Image Processing (VCIP), IS&T/SPIE's Electronic Imaging 2006*, San Jose, CA, USA, 2006.

[5] R. Glasberg, S. Schmiedeke, M. Mocigemba, and T. Sikora, *New Real-Time Approaches for Video-Genre-Classificatio Using High-Level Descriptors and a Set of Classifier* , in *ICSC '08: Proceedings of the 2008 IEEE International Conference on Semantic Computing*, pages 120–127, Washington, DC, USA, 2008, IEEE Computer Society.

[6] H. Kaufmann, *Strabismus*, in *Strabismus*, page 218, Stuttgart: Enke, Stuttgart, 1986.

[7] T. Declerck and A. Cobet, *Towards a cross-media Analysis of spatially co-located Image and Text Regions in TV-News*, in *Proceedings of the 2nd International Conference on Semantics And digital Media Technologies (SAMT'07)*, pages 188–191, 2007.

[8] J. Nemrava, P. Buitelaar, N. Simou, D. Sadlier, V. Svatek, T. Declerck, A. Cobet, T. Sikora, N. O'Connor, V. Tzouvaras, H. Zeiner, and J. Petrak, *An Architecture for Mining Resources Complementary to Audio-Visual Streams*, in *International Workshop on Knowledge Acquisition from Multimedia Content (KAMC '07)*, 2007.

[9] C. M. University, *The CAPTCHA Project*, http://www.captcha.net/.

[10] T. Sato, T. Kanade, E. K. Hughes, and M. A. Smith, *Video OCR for Digital News Archive*, in *CAIVD '98: Proceedings of the 1998 International Workshop on Content-Based Access of Image and Video Databases (CAIVD '98)*, page 52, Washington, DC, USA, 1998, IEEE Computer Society.

[11] R. Lienhart, *Comparison of Automatic Shot Boundary Detection Algorithms*, *Proceedings of SPIE Conference on Storage and Retrival for Image and Video Databases* , 290 (1998).

[12] H. B. Aradhye, *Agenericmethod for determining up/down orientation of text in roman and non-roman scripts*, in *Pattern Recognition 38*, pages 2114–2131, 2005.

[13] R. Lienhart, *Automatic text recognition for video indexing*, *Proc. ACM Multimedia 96* , 11 (2002).

[14] R. Lienhart, C. Kuhmunch, and W. Effelsberg, *On the detection and recognition of television commercials*, in *Multimedia Computing and Systems '97. Proceedings., IEEE International Conference on*, pages 509 –516, 1997.

[15] O. Hori, *A video text extraction method for character recognition*, in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 25 –28, 1999.

[16] R. Lienhart and W. Effelsberg, *Automatic text segmentation and text recognition for video indexing*, *Multimedia Syst.* **8**, 69 (2000).

[17] H. Li, D. Doermann, and O. Kia, *Automatic text detection and tracking in digital video*, *Image Processing, IEEE Transactions on* **9**, 147 (2000).

[18] R. Lienhart and A. Wernicke, *Localizing and segmenting text in images and videos*, *Circuits and Systems for Video Technology, IEEE Transactions on* **12**, 256 (2002).

[19] J. Ohya, A. Shio, and S. Akamatsu, *Recognizing Characters in Scene Images*, *IEEE Trans. Pattern Anal. Mach. Intell.* **16**, 214 (1994).

[20] Wu, V. Manmatha, R. Riseman, and E.M., *Textfinder an automatic system to detect and recognize text in images*, *IEEE Trans. Pattern Anal. Mach. Intell.* (1999).

[21] X. C., J. Y., J. Z., and A. Waibel, *Automatic detection and recognition of signs from natural scenes*, *IEEE Transactions on Image Processing* (2004).

[22] J. Canny, *A computational approach to edge detection*, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 679 (1986).

[23] H. Jeong and C. Kim, *Adaptive determination of filte scales for edge detection*, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **14**, 579 (1992).

[24] P. Bao, L. Zhang, and X. Wu, *Canny Edge Detection Enhancement by Scale Multiplication*, *IEEE Trans. Pattern Anal. Mach. Intell.* **27**, 1485 (2005).

[25] P.J.Burt and E. Adelson, *The Laplacian Pyramid as a Compact Image Code*, *Communications, IEEE Transactions on* **31**, 532 (1983).

[26] H.Erik and W. Meijering, *Spline Interpolation in Medical Imaging: Comparison with other convolution-based approaches*, *EUSIPC* **4**, 1989 (2000).

[27] C. Lanczos, *An Interativ Methode For The Solution Of The Eigenvalue Problem Of Linear Differential And Integral Operators*, *Journal of Research of the National Bureau of Standards* **45**, 255 (1950).

[28] J. Jain and A. Jain, *Displacement Measurement and Its Application in Interframe Image Coding*, *ommunications, IEEE Transactions on* **29**, 1799 (1981).

[29] A. Tourapis, O. Au, and M. Liou, *Highly efficien predictive zonal algorithms for fast block-matching motion estimation*, *Circuits and Systems for Video Technology, IEEE Transactions on* **12**, 934 (2002).

[30] M. Chan, Y. Yu, and A. Constantinides, *Variable size block matching motion compensation with applications to video coding*, *Communications, Speech and Vision, IEE Proceedings I* **137**, 205 (1990).

[31] L. Lee, J. Wang, J. Lee, and J.-D. Shie, *Dynamic search-window adjustment and interlaced search for block-matching algorithm*, *Circuits and Systems for Video Technology, IEEE Transactions on* **3**, 85 (1993).

[32] Y. Chen, Y. Hung, and C. Fuh, *Fast block matching algorithm based on the winner-update strategy*, *Image Processing, IEEE Transactions on* **10**, 1212 (2001).

[33] V. E. Seferidis and M. Ghanbari, *General approach to block-matching motion estimation*, *Optical Engineering* **32**, 1464 (1993).

[34] A. Haubold, *Selection and ranking of text from highly imperfect transcripts for retrieval of video content*, in *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 791–792, New York, NY, USA, 2007, ACM.

[35] Nokia, *QT - Cross-platform application and UI framework*, http://qt.nokia.com/.

[36] public, *fast forward moving pictures expert group - FFmpeg*, http://www.ffmpeg.org/index.html.

[37] Microsoft, *Microsoft Visual Studio*, http://www.microsoft.com/germany/visualstudio/.

[38] E. R. Harold and W. S. Means, *A Desktop Quick Reference*, in *XML in a Nutshell*, page 498, O'REILLY, 2001.

[39] D. Corfield *Projection and Projectability*, http://videolectures.net/different06_corfiel_pp/.

[40] C. Gopalan and D. Manjula, *Sliding window approach based Text Binarisation from Complex Textual images*, *IEEE Trans. International Journal on Computer Science and Engineering* **2** (2010).

[41] T. Lelore and F.Bouchara, *Document Image Binarisation Using Markov Field Model*, in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 551 –555, 2009.

[42] A. Khashman, *Document Image Binarisation Using A Supervised Neural Network*, *International Journal of Neural Systems (IJNS)* **18**, 405 (2008).

[43] L. TECHNOLOGIES, *LEADTOOLS OCR SDK*, http://www.leadtools.com/sdk/ocr/default.htm.

[44] L. R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Apllications in Speech Recognition*, in *Readings in Speech Recognition*, edited by A. Waibel and K.-F. Lee, pages 267–296, Kaufmann, San Mateo, CA, 1990.

[45] D. Chen, H. Bourlard, and J.-P. Thiran, *Text identificatio in complex background using SVM*, in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages 621 – 626, 2001.

[46] G. Kim, V. Govindaraju, and S. N. Srihari, *An architecture for handwritten text recognition systems*, *International Journal on Document Analysis and Recognition* **2**, 37 (1999).

[47] A. J. Viterbi, *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*, *IEEE Trans. Information Theory* **IT-13**, 260 (1967).

[48] A. E. GmbH, *ABBYY Homepage - Software Lösung für OCR*, http://www.abbyy.de/.

[49] G. P. License, *OpenSource OCR*, http://www.gocr.de/.

[50] H. Mittelbach, X. Mueller, and P. Schlegel, *Method For Automatic Character Recognition Employing A Lexicon Having Updated Character Strings*, December 18, 1990, Patent: 4979227.

[51] H. Li and D. S. Doermann, *Text enhancement in digital video using multiple frame integration*, in *ACM Multimedia (1)*, pages 19–22, 1999.

[52] V. Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals*, *Soviet Physics Doklady* **12**, 707 (1966).

[53] H. Bunke, *Error correcting graph matching: on the influenc of the underlying cost function*, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **21**, 917 (1999).

[54] H. Bunke, *Graph matching for visual object recognition*, *Spatial Vision* **13**, 335 (2000).

[55] H. Bunke and S. Günter, *Weighted Mean of a Pair of Graphs*, *Computing* **67**, 209 (2001).

[56] H. Bunke, *Recent Advances in Structural Pattern Recognition with Applications to Visual Form Analysis*, in *Visual Form 2001*, edited by C. Arcelli, L. Cordella, and G. di Baja, volume 2059 of *Lecture Notes in Computer Science*, pages 11–23, Springer Berlin / Heidelberg, 2001.

Abbildungsverzeichnis

Tabellenverzeichnis