TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Systemarchitektur, Lehrstuhl Rechnernetze

BELEGARBEIT

Vergleich und Übersicht von Algorithmen zur Dokumentenduplikaterkennung bei Bildern

Edgar Scherstjanoi
geboren am 17.06.1983 in Berlin

Betreuer: Dipl.-Medieninf. Klemens Muthmann
Verantwortlicher Hochschullehrer: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Eingereicht am 15.10.2010

Kurzfassung

Textdokumente sind einander umso ähnlicher, je mehr Worte und Wortstruktur zu großen Teilen übereinstimmen. Diese Formulierung gilt zwar nicht prinzipiell, dennoch finden sich in vielen Forschungsprojekten Methoden und Ideen, um Textdokumente bezüglich ihrer Übereinstimmung effizient und effektiv zu vergleichen. Da in der Informationstheorie Daten jeder Art als Menge von Wörtern verstanden werden können, liegt die Frage nahe, inwiefern andere Formate mit dieser Herangehensweise konzeptionell vereinbar sind. Einen besonderen Fall stellen Bilddokumente dar, weil die Information darin zweidimensional vorliegt und nicht ohne weiteres als Sequenz von Token interpretierbar ist. *Near Image Duplicate Detection* widmet sich der Aufgabe, den Inhalt von Bilddokumenten zu vergleichen und nutzt dabei gängige Verfahren der Bildverarbeitung. Der Begriff „Ähnlichkeit“ stellt dabei ein großes Problem dar, da er von Menschen formuliert und nur ansatzweise kontextübergreifend definiert ist. ANDREI BRODER entwickelte eine Methode zum Vergleich von Daten, die allgemein auf jedes Format anwendbar sein soll. Eine Implementierung dieses Prinzips zeigt, dass die Übereinstimmung von Texten in einer Menge von Dokumenten nicht paarweise errechnet, sondern geschätzt werden kann. Mit dieser Arbeit wird geprüft, inwiefern dieses Konzept auf Bilddaten anwendbar ist.

Inhaltsverzeichnis

1. Einleitung

Near Image Duplicate Detection ist ein fachübergreifendes Forschungsthema und findet vielseitig und praxisnah Einsatz. Es spannt sich dabei von der Erkennung identischer Kopien von Bilddatensätzen bis zur Erkennung von dem, was man mit dem schwammigen Begriff der Ähnlichkeit zweier Bilder bezeichnen könnte. Es kommen zur Verarbeitung vielerlei Medien in Betracht: Fotografien jeder Art, wissenschaftliche wie auch private, Aufnahmen spezieller Kameras, zum Beispiel im medizinischen Einsatz, oder Szenen aus Videofilmen, aber auch Schriftbilder oder gescannte Texte. Mithilfe effizienter Algorithmen kann man die Darstellung von Sammlungen solcher Daten vereinfachen, indem Gruppen von ähnlichen oder gleichen Bildern erstellt werden. Es wäre auch möglich inhaltlich zu gruppieren. Zwischen Daten verschiedener medialer Quellen werden semantische Zusammenhänge festgestellt, und somit wird eine anwenderfreundliche Struktur in unübersichtliche Datensammlungen gebracht. Eine gute Gruppierung von Dokumenten kann bei dem Einsatz von Suchmaschinen oder Datenbankanfragen sehr hilfreich sein.

Formulierungen für den Begriff der Ähnlichkeit variieren jedoch je nach Anforderung. Beispielsweise wären bei biologischen oder medizinischen Analysen Gruppen von Bildern, die gleiche Strukturen oder Muster enthalten, von Bedeutung. Wird ein Algorithmus wiederum auf eine Datenbank mit gewöhnlichen Fotos angewendet, um ähnlich thematische Abbildungen zu finden, so wird die Farbverteilung im Bild eine große Rolle spielen. Fotos eines Waldes, beispielsweise, weisen viel grün auf, Porttraitaufnahmen einen großen Anteil hautfarbenes gelb-rosa bis beige-braun. Bei den erwähnten Gruppierungen von Suchergebnissen spielen vor allem motivgleiche Bilder eine Rolle. Verfahren, die Prüfsummen oder Hashfunktionen nutzen, reichen dafür nicht aus, da Bildverarbeitungsschritte wie etwa Komprimierung von Daten, beachtet werden müssen. Oft sind solche Veränderungen am Datenmaterial für das menschliche Auge nicht deutlich erkennbar.

Grundlegender Bestandteil aller Algorithmen ist die Identifikation von Bildmerkmalen, damit die zu untersuchenden Daten in ihrer Menge an Informationen auf wesentliche Bestandteile beschränkt werden kann. Extrahierte Features werden als Repräsentanten der Bilddokumente in ihrer Ausprägung und Vorkommensweise verglichen. Beliebte Merkmale eines Bildes sind zum Beispiel Farbverteilungen, bildliche Kanten oder Punkte, die Hinweise auf Bildstrukturen geben, aber auch komplexe Informationen wie Gesichter, Menschen oder sich bewegende Objekte und deren besondere Eigenschaften. Sollen gleiche

Motive unabhängig von der Kameraposition erkannt werden, so müssen die Verfahren zur Extraktion robust gegen Translation oder perspektivische Verzerrung der Aufnahme sein. Meist ist eine Kombination der richtigen Merkmale entscheidend für die Effektivität einer Anwendung. Es geht im Wesentlichen stets um eine semantische Analyse, einen Versuch, den Bildinhalt zu beschreiben. Sind Bilder beschrieben, können Abstandsmaße oder Ähnlichkeitsoperationen definiert und angewendet werden.

In dieser Arbeit werden überwiegend Erkenntnisse aus zwei Forschungsgebieten genutzt. *Computer Vision* beschäftigt sich mit dem bildhaften Schwerpunkt, mit der Aufgabe Bilder maschinell zu sehen und zu verstehen. *Information Retrieval* will gezielt Informationen aus einer Menge von Dokumenten gewinnen, so auch Information über die Ähnlichkeit zweier Dokumente. Für die folgende Arbeit werden Erfahrungen aus beiden Forschungsgebieten genutzt. Um eine vergleichende Übersicht zu gestalten, werden 3 Programme unter gleichen Anforderungen untersucht: Das OpenSource Programm GQView, das an der Columbia University New York entwickelte INDetector [ZEY] sowie eine Umsetzung des SmH-Prinzips [BAA], das ursprünglich zum Vergleichen von Texten entwickelt wurde und nun auf Bilddokumente angewendet werden soll. Den Schwerpunkt bildet die Untersuchung, inwiefern ein Bilddokument informationstheoretisch als Sequenz verstanden werden kann. Verschiedene Überführungen der Bildinformationen in einen quasi-Text vervollständigen die Übersicht der Algorithmen zur Bildduplikaterkennung.

2. Grundlagen

In diesem Kapitel sollen Grundlagen vermittelt werden, die für das Verständnis der weiteren Arbeit notwendig sind. Um ähnliche Dokumente als Elemente einer Menge zu gruppieren, wird die Bedingung festgelegt, dass unter einem kontextbezogenen Ähnlichkeitsaspekt die Dokumente miteinander vergleichbar sind. Dazu werden einleitend Abstandsmaße von Zeichenketten vorgestellt. In großen Dokumenten oder Dokumentmengen ist es meistens notwendig, die Daten für den Vergleich vorzubereiten. Dabei kommen Verfahren zum Einsatz, die gezielt und kompakt spezielle Informationen aus Dokumenten identifizieren, extrahieren und somit eine gut vergleichbare Menge an Daten bieten. Jedes Dokument wird somit durch seine Menge an extrahierten Merkmalen repräsentiert. Da in dieser Arbeit die Analyse von Text- und Bilddokumenten eine wesentliche Rolle spielt, bilden die Grundlagen zur Merkmalsextrakion den zweiten Teil dieses Kapitels. Textdokumente benötigen meist eine statistische oder grammatikalische Analyse. Im Fall von Bildverarbeitung wiederum werden üblicherweise Dokumente zuvor einer Signalanalyse unterzogen.

2.1. Abstandsoperationen

Die vorgestellten Methoden bieten einen brauchbaren Ansatz, wie man mit dem Begriff „Ähnlichkeit“ mathematisch umgehen kann. Die Operatoren dafür gliedern sich in Distanz- und Ähnlichkeitsmaße. Sei D eine abzählbare Menge von Dokumenten sowie $sim : D \times D \rightarrow \mathbb{R}$ ein Änhlichkeitsmaß und $dis : D \times D \rightarrow \mathbb{R}$ ein Distanzmaß, dann muss gelten:

$$
\begin{aligned}
&sim(D_i, D_j) = sim(D_j, D_i) && dis(D_i, D_j) = dis(D_j, D_i) \\
&sim(D_i, D_j) \geq 0 && dis(D_i, D_j) \geq 0 \\
&sim(D_i, D_i) = 1 && dis(D_i, D_i) = 0
\end{aligned}
$$

Zum Berechnen von Ähnlichkeiten können Operatoren beider Maße verwendet werden. Es besteht ein Zusammenhang zwischen ihnen. Meist wird er definiert als

$$dis(D_i, D_j) = 1 - sim(D_i, D_j)$$

Es lässt sich allgemein festlegen: Je größer der Abstand zwischen zwei Sequenzen oder

Mengen, desto geringer die Ähnlichkeit und umgekehrt.

Zuerst sei die **Hamming-Distanz** erwähnt. Eine Anwendung findet sich in der Kodierungstheorie, wo sie den Abstand für die Unterschiedlichkeit zweier Zeichenketten bezeichnet. Ausgehend von einer festen Länge der zu untersuchenden Sequenzen wird jede Position in **A** gezählt, die ein zur korrespondierenden Position in **B** unterschiedliches Zeichen aufweist. In der Kommunikationstechnik, beispielsweise, wird damit eine Kennzahl für Übertragungsfehler eines gesendeten Wortes geboten.

$$\begin{aligned} \mathbf{A} &= \texttt{w , o , r , t} \\ \mathbf{B} &= \texttt{f , o , r , d} \\ \\ \text{Hamming}(\mathbf{A},\mathbf{B}) &= \texttt{1 + 0 + 0 + 1} \quad = \mathbf{2} \end{aligned}$$

Einen anderen Ansatz stellt die **Levensthein-Distanz**, auch einfache Editierdistanz, dar. Ausgangspunkt ist die Annahme, einzelne Zeichen seien „nur“ vertauscht, gelöscht oder hinzugefügt worden. Demnach gebe es eine Menge an Operationen, die **A** in **B** zurückführt. Die minimale Anzahl dieser Operationen, genauer: Einfüge-, Lösch- und Austauschoperationen, bestimmt die Levensthein-Distanz. Im folgenden Beispiel wird jede notwendige Operation mit einem Kostenpunkt $k = 1$ bewertet.

$$\begin{aligned} \mathbf{A} &= \texttt{f , o , r , d} \\ \mathbf{B} &= \texttt{o , r , t} \\ \\ \text{lösche}(\texttt{f},\mathbf{A}) &= \texttt{o , r , d} \\ \text{tausche}(\texttt{o},\texttt{t},\mathbf{A}) &= \texttt{o , r , t} \quad = \mathbf{B} \\ \text{Levensthein}(\mathbf{A},\mathbf{B}) &= \sum k \quad = \mathbf{2} \end{aligned}$$

Eine Möglichkeit zur Bestimmung der Ähnlichkeit von Mengen bietet der **Jaccard-Index**. Es wird dafür die relative Anzahl der Elemente einer Eigenschaft zur Gesamtzahl aller Elemente der Mengen gemessen. In diesem Beispiel sind gleiche Eigenschaften von Elementen durch Buchstaben gekennzeichnet.

$$\begin{aligned} \mathbf{A} &= \{\texttt{a , b , c , d , e}\} \\ \mathbf{B} &= \{\texttt{c , d , e , f , g, h, i}\} \\ \text{Jaccard}(\mathbf{A},\mathbf{B}) &= \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A} \cup \mathbf{B}|} = \frac{3}{9} \quad \approx \mathbf{0{,}33} \end{aligned}$$

Weiterhin seien noch Methoden erwähnt, die eine Sequenzalignierung realisieren. Vor allem in der Bioinformatik, zur Analyse von Nukleotid- oder Aminosäuresequenzen, findet dieses Verfahren Anwendung. Das Ziel ist es in diesen Forschungsgebieten, funktionelle,

strukturelle oder evolutionäre Beziehungen zu entdecken. Kernpunkt für den Vergleich ist das Verschieben oder Aufteilen der Sequenzen, um eine möglichst geringe Hamming-Distanz zu erreichen. Es wird unterschieden zwischen lokalen und globalen Alignments. Zur Illustration des Prinzips sei das globale Verfahren **Needleman-Wunsch** vorgestellt. Es beschränkt sich zur Bewertung zweier Sequenzen auf die minimalen Kosten, die entstehen, wenn Lücken, sogenannte Gaps, in die Sequenzen eingefügt werden, um eine höhere Übereinstimmung zu erreichen. Für das folgende Beispiel sei jede Übereinstimmung in einer Position mit $k = 1$, jeder Unterschied mit $k = -1$ bewertet.

$$
\begin{aligned}
\mathbf{A} &= \texttt{A ,A ,G ,C ,T} \\
\mathbf{B} &= \texttt{A ,G ,C ,T}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{A'} &= \texttt{A ,A ,G ,C ,T} \\
\mathbf{B'} &= \texttt{A ,- ,G ,C ,T} \\
\textstyle\sum k &= \texttt{1 -1 +1 +1 +1}
\end{aligned}
$$

Needleman-Wunsch(**A**,**B**) = **3**

Eine Möglichkeit, um in großen Dokumentmengen das relative Auftreten *eines* Dokumentelements festzustellen, ist mittels Berechnung der **TF/IDF - Term Frequency / Inverse Document Frequency** gegeben. Term Frequency(TF) ist eine Kennzahl für eine statistische Bedeutung eines speziellen Elements in seiner Menge, Document Frequency(DF) beschreibt wiederum die Bedeutung der Dokumente, in denen ein solches Element existiert, unter der Menge aller untersuchten Dokumente. Seien e die Elemente eines Dokumentes $d = \{e_0, e_1, ..., e_m\}$ und die Menge aller Dokumente $D = \{d_0, d_1, ..., d_n\}$ sowie $c_{x,y}$ die Anzahl der Elemente e_x im Dokument d_y

$$
tf(i,j) = \frac{c_{i,j}}{\sum_{k=1}^{m} c_{k,j}} \qquad\qquad idf(i) = \log \frac{n}{|\{d : e_i \in d\}|}
$$

dann ergibt sich TFIDF an einem Beispiel wie folgt

$$
\begin{aligned}
d_0 &= \texttt{A ,A ,A ,C ,T} \\
d_1 &= \texttt{A ,A ,C ,T} \\
d_2 &= \texttt{G ,G ,C ,T}
\end{aligned}
$$

$$
\text{TFIDF}(\texttt{A}, d_0) = \tfrac{3}{5} \cdot \log \tfrac{3}{2} = \mathbf{0{,}24}
$$

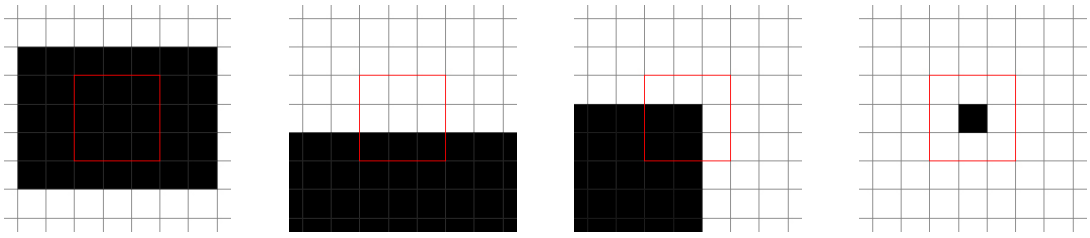
Die vorgestellten Methoden können verfeinert und auf spezielle Problemfälle hin optimiert angewendet werden. Es findet sich eine Vielzahl von Kennzahlen, die auf den genannten

Verfahren aufbauen. Zum Beispiel können bei der Levensthein-Distanz erweiterte Bewertungsschemata eingeführt oder sogar neue Operationen zur Bewertung definiert werden, so in **Damerau-Levensthein**, um vertauschte Zeichen zu identifizieren. Die richtige Auswahl und anwendungsnahe Optimierung von Methoden zum Vergleich von Zeichenketten trägt stark zur Effizienz und Effektivität eines Forschungsprojektes bei. Beispielsweise nutzt [Gm04] TF-IDF, um eine Gegenmaßnahme zur wachsenden Menge an Webspam zu entwickeln. [BS07] verwendet die Jaccard verwandte Methode **Cosinus-Ähnlichkeit**, um zwischen Vektoren von Dokumentmerkmalen den Grad der Ähnlichkeit zu messen.

2.2. Erkennung von Merkmalen in Bilddokumenten

Die Erkennung und Extraktion von Dokumentmerkmalen, sogenannten Features, verfolgt vor allem den Zweck, kompakt die Daten auf einen wesentlichen Inhalt zu beschränken und dadurch alle überflüssigen Informationen zu ignorieren. Je nach Anforderung müssen die Daten reduziert werden, damit vor allem jene Informationen extrahiert werden können, die für die Weiterverarbeitung notwendig sind. Im Falle von Duplikaterkennung ist demnach die Definition der Ähnlichkeit grundlegend für die Gestaltung der Merkmalsidentifikation. Identifizierte Informationen werden oft durch Vektoren beschrieben, so dass sie für eine bestimmte maschinelle Verarbeitung, zum Beispiel den Ähnlichkeitsvergleich, genutzt werden können.

2.2.1. Bildsignalanalyse



(a) Fläche (b) Kante (c) Ecke (d) isolierter Punkt

Abbildung 2.1.: Merkmale in einem Bild

Der Schwerpunkt der Algorithmen zur Merkmalserkennung in Bilddokumenten besteht in dem Versuch, bestimmte Bildinformationen extrahieren zu können. Vor allem die Bildbereiche, die sich in ihrem Umfeld als einzigartig herausstellen, gelten als interessant, somit auch als Kandidaten zur Extraktion. Als solche kommen meist Ecken von bildlichen Kanten, Flächen oder Formen in Frage (Abbildung 2.1). Meist muss zwischen Merkmal und Bildstörung unterschieden werden. Isolierte Pixel (in Abbildung 2.1 Bild (d)) weisen

eher auf Signalrauschen hin, da auf natürlichen Bildern selten Objekte vorkommen, die der Größe eines einzelnen Bildpunktes entsprechen.

Bildsignale werden üblicherweise als diskrete 2-dimensionale Funktion zur Weiter-

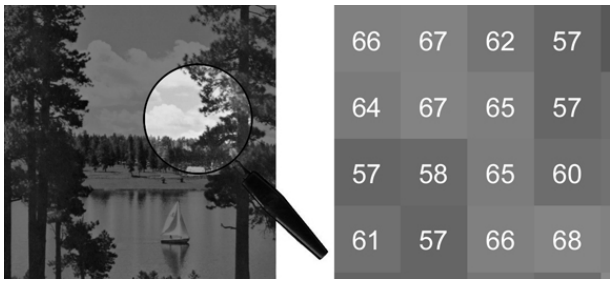
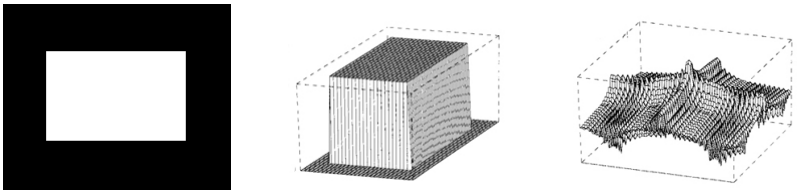


Abbildung 2.2.: Jede (x,y)-Stelle im Bild hat einen Funktionswert aus dem 8-bit Wertebereich $[0, 255]$

verarbeitung interpretiert $B : X \times Y \rightarrow [0, 255] \subseteq \mathbb{N}$ (Abbildung 2.2). Das hat vor allem technische Vorteile, da viele Methoden zur Signalanalyse verwendet werden können. Es ist möglich, beispielsweise mithilfe von Fourier-Transformationen, Bildsignale eindeutig in ihren Frequenzanteilen zu analysieren. Dadurch können für die Weiterverarbeitung wertvolle Informationen extrahiert werden. Zur Veranschaulichung zeigt Abbildung 2.3 den gemeinten Prozess [BB05]. Das Bild (a) wird als zweidimensionale Funktion interpretiert (b), dass durch Fourier-Transformation (c) mit seinen spektralen Komponenten dargestellt wird. Diese Operation ist mit jedem Bildsignal durchführbar. Ein Rechteck wird durch die Kombination von Komponenten mit in horizontale und vertikale Richtung abnehmend kleinere Koeffizienten dargestellt.



(a) Bild (b) Bildsignal (c) Signaltransformation

Abbildung 2.3.: 2D-Fouriertransformation eines Bildes, welches nur aus einem weißem Rechteck besteht.

Eine in der Bildverarbeitung oft verwendete Fouriertransformation ist die **Diskrete Kosinus-Transformation** (2D-DCT). Mit der Annahme, das Bildsignal setzt sich hinter den Bildgrenzen periodisch fort, können aus dem diskreten Signal im Ortsbereich die Koeffizienten der linear kombinierten Kosinusfunktionen im Frequenzbereich bestimmt werden. Das Ergebnis ist die Menge an spektralen Komponenten, aus denen sich das Bildsignal zusammensetzt. Die Anzahl der Koeffizienten entspricht der Größe des Bildsignals.
Jede Funktion in Abbildung 2.4 illustriert eine Kosinus-Basisfunktion als Spektralkompo-

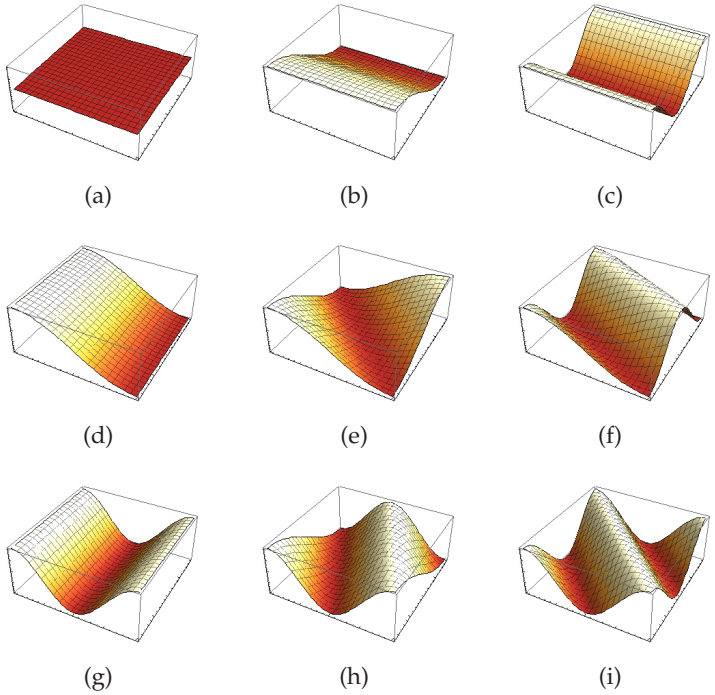


Abbildung 2.4.: Zweidimensionale Basiscosinusfunktionen.(e), (f), (h) und (i) sind jeweils Linearkombination aus der horizontalen ((d),(g)) und vertikalen ((b),(c)) Kosinusfunktion.

nente eines diskreten Bildsignals. Durch die Transformation wird mit Koeffizienten jeder Komponente das entsprechende Ausmaß im Bildsignal beschrieben. Die Größe des Bildsignals bestimmt die notwendige Menge an Basiskomponenten, deren Zusammensetzung das Bildsignal widerspiegelt. Diese Herangehensweise an Bilddaten ist Bestandteil vieler Komprimierungsverfahren, um redundante Informationen bei der Speicherung vernachlässigen zu können. Sei $B_{x,y} \in [0..255]$ das $X \times Y$ große Bild, $\tilde{B}_{\tilde{x},\tilde{y}} \in \mathbb{R}$ die Transformation, $x, \tilde{x} \in [0..X-1]$ sowie $y, \tilde{y} \in [0..Y-1]$, dann beschreibt die folgende Formel die Amplituden der $(\tilde{x}, \tilde{y})$-ten Basisfunktionen.

$$\tilde{B}_{\tilde{x},\tilde{y}} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} B_{x,y} \cdot \cos\left[\frac{\pi}{X}\left(x+\frac{1}{2}\right)\tilde{x}\right] \cos\left[\frac{\pi}{Y}\left(y+\frac{1}{2}\right)\tilde{y}\right]$$

Zur Veranschaulichung zeigt Abbildung 2.5 Beispiele von 2D-DCT transformierten Bildern. Der Koeffizient der tiefsten Frequenz ((a) in Abbildung 2.4), der Gleichanteil des Bildsignals oder DC-Komponente genannt, wird in der oberen linken Ecke des Bildes dargestellt. Jeder benachbarte Wert entspricht der horizontal beziehungsweise vertikal nächst höheren Frequenz. Negative Amplituden von Spektralkomponenten sind als schwarze Bildpunkte dargestellt, positive weiß. Grau steht für die Komponenten, die wenig oder keinen Einfluss auf das Bildsignal haben. Oft wird vor der Analyse ein Bildsignal geglättet um eventuelles Signalrauschen im Vorfeld zu eliminieren, zum Beispiel mit einem **Gauß**-Filter. Das Signal

(a) Baboon (b) Baboon DCT (c) Lena (d) Lena DCT

Abbildung 2.5.: (b) und (d) sind Kosinustransformationen von den Bildern (a) und (c)

wird mit der Gaußfunktion im Ortsbereich multipliziert, also die Helligkeit aller Punkte in ihrer Umgebung normalverteilt ausgebreitet, vergleichbar mit dem Effekt eines Tiefpassfilters im Frequenzbereich. Das hat zwangsweise auch einen Informationsverlust zur Folge, da ebenso markante Bildbereiche verloren gehen, die zur Informationsidentifikation benötigt werden. Die Verteilung des Helligkeitswertes eines Bildpunktes durch den Gaußfilter der Größe $n \times m$ und der Varianz σ^2 wird durch die Formel

$$g_{nm} = \frac{1}{2\pi\sigma^2} e^{-\frac{n^2+m^2}{2\sigma^2}}$$

beschrieben. Im Resultat der geglätteten Bilddokumente (Abbildung 2.6) wird der Zusammenhang zwischen dem Gaußfilter im Orts- und Frequenzbereich erkennbar. Es bleiben vorwiegend tiefere Frequenzen (obere linke Ecke) übrig. Scharfe Kanten im Bildsignal gelten spektral als hohe Frequenz (untere rechte Ecke) und werden durch Glättung verringert.

(a) Baboon (b) Baboon (c) Lena (d) Lena

Abbildung 2.6.: Beispielbilder und korrespondierende DCT Transformationen. (a) und (c) sind Gaußgeglättete Bilder, (b) und (d) deren korrespondierende DCT.

2.2.2. Edge Detection

Die Bildfunktion $B_{x,y}$ kann auch in ihren Extremwerten und Wendepunkten untersucht werden, um herauszufinden, an welchen Stellen, in welche Richtung und mit welcher Stärke ein Farbwechsel im Bildsignal vorliegt. Diese Operationen werden auch als Kantende-

tektoren bezeichnet, da jeder Wende- und Extrempunkt eines Signals auf die Existenz einer bildlichen Kante hinweist. Der **Sobel**-Operator ist definiert als die Kombination der horizontalen und vertikalen ersten Ableitung einer 2-dimensionalen Funktion. Wird nach der Verwendung von Sobel ein Schwellwertfilter angewendet, sind Differentialquotienten oberhalb eines Grenzwertes mit weiß, alle anderen mit schwarz gleichgesetzt. Mit dieser Methode werden vor allem solche Kanten identifiziert, die starke Farbwechsel aufweisen. Eine ausgeglichene Alternative, damit auch nicht-fokussierte oder hellere Farbwechsel als Kanten mit einbezogen werden können, bietet der **Laplace**-Filter. Der Wert der zweiten Ableitung (Laplace-Operator) ist an jedem Extrempunkt des Bilsignals gleich 0, was ebenfalls auf die Existenz einer Kante deutet. Eine Erweiterung findet sich beim **Laplacian of Gaussian**-Operator. Mit der Anwendung eines Laplace-Filters nach Gaußglättung des Bildsignals können größere Stellen starker Veränderung, also auch weichere Kanten von Objekten, erkannt werden. Ebenso häufig findet der **Differential of Gaussian**-Filter für diese Zwecke Anwendung. Die Idee besteht darin, die Differenz verschiedener Gauss-Glättungen als Indiz zur Kantendetektion zu verwenden. Abbildung 2.7 bietet eine Übersicht der erwähnten Detektoren. Sind Kanten identifiziert, können Bildregionen als wichtig markiert werden.

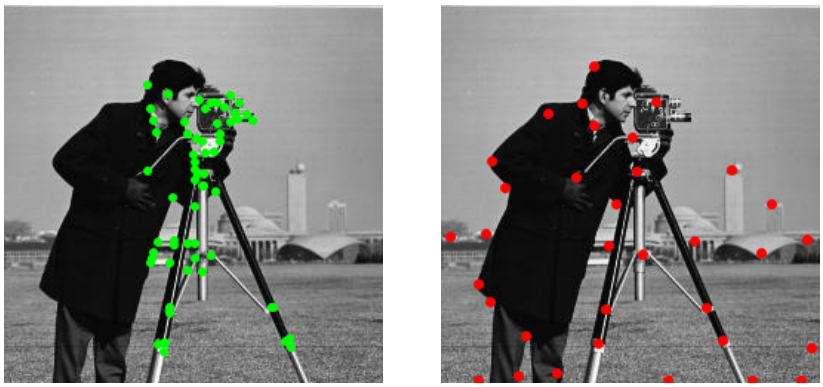
(a) (b) Sobel (c) Laplace (d) DoG (e) LoG

Abbildung 2.7.: Kantendetektoren

2.2.3. Interest Point Detection

Alternativ können auch dominante Punkte in einem Bildsignal identifiziert werden, sogenannte *Interest Points*, die sich von Kanten unterscheiden. Der **Movarec**-Operator bietet einen einfachen Ansatz zur Bestimmung von markanten Punkten eines Bildsignals. Dazu wird ausgehend von einem Bildfenster, meist $3 \times 3, 5 \times 5$ oder 7×7, die Differenz der Helligkeitswerte ermittelt, wenn das Fenster in eine der Richtungen versetzt wird. Somit kann für jede Richtung der Gradient ermittelt werden. Liegt das Minimum der Gradientensummen in einem Punkt über einem Schwellwert, so wird diese Stelle als markant bezeichnet. Mit diesem Verfahren ist es jedoch nicht möglich, Interest Points eines rotierten Bildes seinen Ursprüngen zuzuweisen. Eine rotationsinvariante Erweiterung, die Analyse der Autokorrelation des Bildsignals, wurde mit dem **Harris**-Detektor eingeführt [Der04]: Das Bildsignal wird in alle Richtungen differenziert, und eine Autokorrelation zwischen den Differentialen der Richtungen erhoben. Daraus kann für jeden Punkt ein Wert seiner „Eckigkeit“ errechnet werden. Alle Bildpunkte die entlang des Gradienten nicht zu lokalen Maxima gehören, werden aussortiert. Abbildung 2.8 zeigt Ergebnisse der *Interest Point*-Detektoren. Man erkennt

dort, dass Harris auch Extraktionskandidaten im Hintergrund auswählt, wobei *Movarec* sich nur auf starke hochfrequente Signale, den fokussierten Vordergrund im Bild, bezieht.



(a) Movarec (b) Harris

Abbildung 2.8.: Point Detection Operatoren

2.2.4. Darstellung von Bildfeatures

Features können auf verschiedene Art und Weise dargestellt werden. Der wohl einfachste Weg wäre eine Boolesche Klassifizierung, ob in einem Bildpunkt eine identifizierte Kante beziehungsweise ein Punkt vorhanden ist oder nicht. Da diese Beschreibung nicht ausreicht, um komplexe bildliche Vergleichsoperationen anzuwenden, werden oft Verfahren genutzt, die mehr Information in einem Vektor bereit stellen. Meist fällt die Auswahl auf die Ausrichtung des Merkmals im Bild, aber auch auf Farbinformationen sowie Beschreibungen einer geometrischen Region um den Merkmalspunkt.

Ein populärer Deskriptor von Bildmerkmalen ist in **SIFT - Scale Invariant Feature Transform** zu finden [Low99]. SIFT ist in der Lage transformationsunabhängig und robust gegen Beleuchtungsvarianz sowie Bildrauschen die Merkmale zu beschreiben. Der Ablauf des SIFT-Algorithmus wird zur Übersicht in vier Phasen vorgestellt.

1. Suchen nach den **Extrema im Skalenraum**. Als Skalenraum ist die Menge von verschieden geglätteten Bildern zu verstehen. Aus der Diffenrenz dieser Glättungen (DoG) können die Extrema des Skaleraumes identifiziert werden, welche die Grundmenge an Interest Points liefern, die in ihrer Position und Skalierung invariant sind.

2. **Lokalisation** der Schlüsselpunkte. Die Menge der Extrema im Skalenraum ist womöglich zu groß. Es wird mit diesem Schritt die Auswahl der Interst Points beschränkt, zum Beispiel unter dem Kriterium, in einer Skala einen festgelegten Grenzwert für Kontrast zu überschreiten. Eine weitere Kontrolle der Bildfeatures ist die Bestimmung der Punkte, die sich entlang von Kanten als sicher gegenüber Signalstörung erwiesen haben.

3. Bestimmung der **Orientierung** der verbliebenen Interest Points. In diesem Schritt lässt sich die rotationsinvariante Beschreibung zur Orientierung der Schlüsselpunkte durchführen. Der Gradient jedes Punktes bestimmt die Richtung, die dann in Relation zum benachbarten Umfeld repräsentiert wird.

4. **Beschreibung** der Merkmale. Im letzten Schritt wird dafür gesorgt, dass die Beschreibung der Interest Point durch einen *Descriptor* gestaltet wird. Dieser soll zum einen charakteristisch für das Merkmal sein und zum anderen weitere mögliche Variationen von Bilddaten tolerieren, zum Beispiel Helligkeit oder räumliche Informationen des Aufnahmepunktes.

2.3. Merkmale in Textdokumenten

In Textdokumenten bestehen die Möglichkeiten der Merkmalserkennung abhängig von der gegebenen textlichen Struktur. Sind Textabschnitte, zum Beispiel Überschrift, Autor, Aufzählungszeichen oder inhaltliche Kategorien, annotiert, so liegt der Schwerpunkt darin, die Korrektheit und Vollständigkeit dieser Notationen zu überprüfen. Vorliegende semi-Strukturen können dann zur Bestimmung eines Deskriptors oder zur Extraktion von Merkmalen des Dokumentes verwendet werden. Finden sich nur unzureichend oder gar keine Strukturen in einem Textdokument, so müssen zur Erkennung inhaltlicher Merkmale die gegebenen Wörter reduziert werden: **Lemmatisierung** versucht ein Wort zu seiner lexikalischen Basis zurückzuführen, **Stemming** beschränkt ein Wort auf kleine informationstragende Elemente, unabhängig der Lexika und daher einfach überführbar auf jede Art von Sprache.

Eine weit verbreitete Methode ist der **Porter-Stemmer**-Algorithmus [Por06]. Es gilt darin die Annahme, dass Wörter, die eine gleiche Bedeutung haben, sich auf einen Wortstamm zurückführen lassen. Eine grundlegende Herausforderung dieser Methode bestand darin, ohne die Hilfe von gr ammatikalischen Regeln, ein Reduzieren der Wörter auf ihren Grundbestandteil zu verwirklichen. Der Algorithmus funktioniert, indem innerhalb von Wörtern die Zeichenfolgen von Vokalen und Konsonanten analysiert werden. Seien C und V eine Aufeinanderfolge von Konsonanten beziehungsweise Vokalen, dann lässt sich jedes natürlichsprachige Wort als Zeichensequenz $[C]VCVC...[V]$ betrachten. Nun wird das m-fache Auftreten eines Wechsels von V nach C gezählt. Beispielsweise $m(\texttt{TREE}) = 0$ oder $m(\texttt{TREES}) = 1$. Mit dieser Bewertung von Wörtern können Kürzungsregeln definiert werden, deren Anwendung die gewünschte Reduzierung bewirkt. Zum Beispiel:

$(m > 0)$	`ATIONAL` $\rightarrow$ `ATE`	*relational* $\rightarrow$ *relate*
$(m > 0)$	`FULNESS` $\rightarrow$ `FUL`	*hopefulness* $\rightarrow$ *hopeful*
$(m > 0)$	`ENCE` $\rightarrow$	*inference* $\rightarrow$ *infer*

In [Ste98] wird **Fuzzy Fingerprinting** vorgestellt, ein Verfahren, was zur Indizierung von wichtigen Textelementen sich statistischer Messungen bedient. Die Basis dafür bilden sogenannte äquivalente Präfixklassen der natürlichen Sprache. In einem Vektor werden für ausgewählte Präfixe von Wörtern, beispielsweise `ent-`, `unter-`, `gegen-`, die Verteilungswerte innerhalb des Dokumentes gehalten. Diese Werte werden mit einem weiteren Präfixvektor mit a-priori Wahrscheinlichkeiten kombiniert, um Fingerprints des Dokumentes zu errechnen. Die Operation zwischen den beiden Vektoren wird als Fuzzifizierung bezeichnet. Unter Fuzziness versteht man grob formuliert, Definitionen, die zwischen groben Strukturen ansetzen und zur Modellierung von Unschärfe und Unsicherheiten verwendet werden. In Abbildung 2.9 ist dieser Prozess schematisiert.

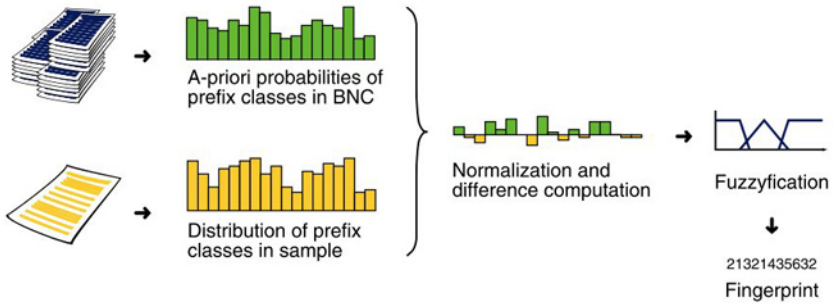


Abbildung 2.9.: Fuzzy Fingerprint Schema. Als Referenz für a-prioi Präfixwahrscheinlichkeiten wurde der British National Corpus (BNC) verwendet.

2.4. stochastische Verfahren

Die Menge an Informationen ist bei Textdokumenten eine Sequenz von Wörtern. Sie kann schnell zu groß sein. Ebenso kann es bei Bilddokumenten passieren, dass eine Menge extrahierter Features die Größe des ursprünglichen Dokumentes überschreitet. Für beide Fälle können modifizierte Hashfunktionen helfen, bestimmte Merkmale statistisch auszuwählen. Der eigentliche Zweck von Hash- oder Fingerprintfunktionen ist es, Daten eindeutige Schlüssel zuzuweisen. Dabei ist das Rückführen der Schlüssel zu ihrem ursprünglichen Datum nicht ohne weiteres möglich. Die im Zusammenhang dieser Arbeit wichtigere Eigenschaft von Hashfunktionen ist, dass die Wahrscheinlichkeit, mit der zwei verschiedene Daten den selben Hashwert erhalten, annähernd null ist. Deswegen eignen sich Hashfunktionen vorab besonders gut, um Daten effizient auf identische Inhalte zu überprüfen. Das Prinzip kann umformuliert auch für Ähnlichkeitsanfragen hilfreich sein.

Locality Sensitive Hashing (LSH) Funktionen sind Algorithmen, die sich der Problematik

des Hashings für Ähnlichkeitsbeziehung widmen. Aus einer Familie von Hashfunktionen werden zufällig und unabhängig gewählte Funktionen miteinander so verknüpft, dass sie jedem ähnlichen Datensatz mit hoher Wahrscheinlichkeit den gleichen Hashwert zuweisen. Anders formuliert: $\Pr[hash(A) = hash(B)] = sim(A, B)$. Die Wahrscheinlichkeit der Kollision zweier gehashter Dokumente A und B entspricht dem Grad ihrer Ähnlichkeit. Eine sehr einfache Hashfunktion, um beispielsweise Hamming-Distanzen zu schätzen, ist $hash(X) = X_i$ die Abbildung eines Wortes auf seine i-te Stelle [DI04]. Die Hashwerte zweier Wörter A und B

$$\begin{aligned} A &= \texttt{1, 0, 1, 0} \\ B &= \texttt{0, 1, 1, 0} \end{aligned}$$

kollidieren bei einer zufällig gewählten Hashfunktion $hash_i \quad i \in \{0, 1, ..., |A| = |B|\}$ mit der Wahrscheinlichkeit $\Pr[hash(A) = hash(B)] = 0{,}5$. Es gibt weitere Ideen zur Konstruktion solcher Familien von Hashfunktionen, um auch andere Distanzmaße (Jaccard, Kosinus,...) schätzen zu können. Das Ergebnis sind stets die auf reellen Zahlen abgebildeten Datensatzrepräsentationen.

3. Anforderungsanalyse

Damit man den Begriff Ähnlichkeit anwenden kann, muss eine Relation in Bezug auf eine oder mehrere Eigenschaften von Objekten definiert sein. Es gilt: Je geringer die Unterschiede zwischen den Eigenschaften sind, desto ähnlicher sind sich die entsprechenden Objekte unter diesem Aspekt betrachtet. Folglich weisen identische Objekte keine Unterschiede auf. Aus natürlicher Sicht ist alles in ständiger Veränderung und nichts für alle Zeit identisch. In der Mathematik wiederum sind Gleichnisse klar definiert und bilden die Grundlage dafür, dass quantitative Aussagen getroffen werden können. Insofern ist eine vom Menschen gestellte Anfrage zur Ähnlichkeit immer im Zusammenhang mit seiner Erfahrung zu sehen: alles hängt davon ab, welche Eigenschaften in dieser Anfrage gemeint worden sind. Sind diese als Voraussetzung geklärt, kann mit statistischen Methoden eine Antwort generiert werden. Die Bilder in Abbildung 3.1 illustrieren die gemeinte Problematik. Bei einer Anfrage, Bilddokumente mit gleichen Motiven zu finden, wären Bild (a) und Bild (b) mögliche Ergebnisse. Werden jedoch Dokumente der Eigenschaft „gleicher Stil“ gesucht, sind die Bilder (b) und (c) als ähnlich zu bewerten.

(a) Apfel (b) Skizze eines Apfels (c) Skizze

Abbildung 3.1.: Die Definition der Ähnlichkeit ist stets abhängig von einer Anfrage hinsichtlich bestimmter Eigenschaften.

3.1. SmH Shingle/minHash

Für den Vergleich von Textdaten werden Dokumente als Objekte und beispielsweise deren Wortstruktur als Eigenschaften, die verglichen werden sollen, gesehen. Große Dokumentmengen bringen natürlich technische Probleme mit sich, da ein paarweiser Vergleich aller Dokumente viele Operationen benötigt. Nach einer Idee von BRODER lässt sich die strukturelle Übereinstimmung einer Menge von Datensätzen effizienter abschätzen [BAA]. BRO-

DERs Ziel war es, ein System zu entwickeln, dass allgemein Textdokumente auf Ähnlichkeit hin analysiert. Textdokumente sind in erster Linie Zeichensequenzen. Es können daraus sich überschneidende Untersequenzen, sogenannte *Shingles* (k-Gramme), des Dokumentes gebildet und das Dokument kann als Menge der gebildeten Shingles betrachtet werden. Somit lässt sich das Problem der Ähnlichkeit zweier Textdokumente als mengentheoretisches Problem behandeln. Zur repräsentativen Extraktion bestimmter Shingles als Dokumentfeatures dient das *minHash*-Verfahren als statistisches Werkzeug. Die Kombination aus dem *Shingling*-Prinzip und dem *minHash*-Verfahren gestaltet diesen Ansatz effizient und effektiv. Das Konzept wird im Rahmen dieser Arbeit daher als SMH-Algorihmus(Shingle/minHash) abgekürzt. (In anderen Arbeiten wird auch die Bezeichnung DSC - Digital Syntactic Clustering - verwendet.)

3.1.1. Shingling

Mit dem SMH-Prinzip lässt sich eine Gruppierung von Textdokumenten bezüglich ihrer syntaktischen Ähnlichkeit vornehmen. So war SMH beispielsweise Bestandteil der Implementierung der Suchmaschine *AltaVista* [Bro00]. Jedes Wort eines Textdokumentes, jeder zusammenhängende Zeichensatz ohne Trennungssymbole (Leerzeichen, Trennstrich, Punkt, Komma...) ist ein Token, und daher sind Dokumente in erster Linie Sequenzen von Token. Kontinuierliche Untersequenzen in einem Dokument werden *Shingles* genannt. Die Elemente sind in Textdokumenten typischerweise Wörter der natürlichen Sprache oder Annotationen zur maschinellen Verarbeitung, wie etwa HTML-Tags. Das ω-Shingling legt die Anzahl ω der Token in einem Shingle fest. Wird das Dokument D in die Menge der Shingles $S_D(\omega)$ überführt, so würde der Text

$$D = \texttt{A rose is a rose, is a rose.}$$

bei einem $\omega = 3$-Shingling durch die Multimenge

$$\begin{aligned} S_D(3) = \{&(A, rose, is),\\ &(rose, is, a),\\ &(is, a, rose),\\ &(a, rose, is),\\ &(is, a, rose)\} \end{aligned}$$

repräsentiert. Mehrfach auftretende Shingles werden entweder mit Annotationen abzählend markiert oder man vernachlässigt solche Fälle und behält nur je ein Exemplar eines Shingles. Beide Ansätze bieten eine Voraussetzung, um die Übereinstimmung, in [BAA] *Resemblance* genannt, zweier Dokumente A und B als ω-*Shingles* $S_A(\omega)$ sowie $S_B(\omega)$ auf die Berechnung

	$\omega = 1$	$\omega = 2$	$\omega = 3$
$Resemblance_{\omega}(A, B)$	$\frac{5}{6} = 0,83$	$\frac{3}{5} = 0,6$	$\frac{2}{5} = 0,4$

Tabelle 3.1.: Übereinstimmung von A und B bei $\omega = 1, 2, 3$

der relativen Anzahl von Elementen ihrer Schnittmenge umzuformulieren:

$$Resemblance_{\omega}(A, B) = \frac{|S_A(\omega) \cap S_B(\omega)|}{|S_A(\omega) \cup S_B(\omega)|} = r$$

Dieser Term ist eine Variante des *Jaccard*-Ähnlichkeitsmaßes. Daher ist das Ergebnis $r \in [0..1] \subseteq \mathbb{Q}$ und es lässt sich schlußfolgern: je näher r an 1 ist, desto ähnlicher sind sich die Dokumente A und B, $r = 1$ deutet auf identische und $r = 0$ auf verschiedene Paare von Dokumenten. Zur Anwendung genügt es, einen Grenzwert $\tau \in]0..1]$ festzulegen. Ist $Resemblance(A, B) > \tau$, so werden die Dokumente A und B als ähnlich bezeichnet.

Zum Beispiel seien die Texte

$$A = \texttt{A rose is a rose.}$$

und

$$B = \texttt{A red rose is a rose.}$$

gegeben. Tabelle 3.1 zeigt Ergebnisse der Berechnung der *Jaccard*-Übereinstimmung von A und B für verschiedene Shinglegrößen ω. Erwähnenswert ist, dass im Falle einer Shinglegröße von $\omega = 1$ jede beliebige Permutation der Elemente von A zu einer Übereinstimmung von 1 führt. Ist $\omega = 2$, $A = (a, b, a, c, a)$ und $A' = (a, c, a, b, a)$, dann ergibt sich ebenfalls $Resemblance_{\omega=2}(A, A') = 1$. Je größer Shingles gewählt werden, desto stärker wird die Reihenfolge der Sequenz beachtet, darum sollte die Shinglegröße in der Anwendung nicht zu klein gewählt werden. Wiederum beeinflussen auch zu große Shingles die Aussagekraft zur Bestimmung der Ähnlichkeit, da jedes veränderte Token auch ω Shingles verändert und somit den errechneten Wert der Ähnlichkeit senkt.

Die Berechnung der Übereinstimmung ist nicht transitiv [BAA]. Mit anderen Worten, ist A ähnlich B, also $Resemblance(A, B) > \tau$, und B ähnlich C, analog $Resemblance(B, C) > \tau$, ist die Schlussfolgerung falsch, A sei in jedem Falle auch ähnlich C, $Resemblance(A, C) > \tau$. Das lässt sich an einem Beispiel einfach zeigen: In aufeinander folgenden Frames einer ungeschnittenen Videosequenz sind sich benachbarte Bilder ähnlicher, als das erste und das letzte Frame.

3.1.2. minHash

Herkömmliche Verfahren zum Ermessen der Ähnlichkeit benötigen einen paarweisen Vergleich. Dadurch wird in großen Dokumentenmengen der notwendige Rechenaufwand

zu hoch. In einem Datensatz von m Dokumenten wären für einen paarweisen Vergleich $\binom{m}{2}$ Operationen nötig. BRODER zeigt, dass es möglich ist mit Shingling und einer Abschätzung des Jaccard-Koeffizienten alle Dokumentenpaare in linearem Aufwand zu vergleichen.

S_D sei die Menge aller Shingles eines Dokumentes D. Jedes Shingle erhält einen eindeutigen numerischen Fingerabdruck. Somit kann S_D als eine Teilmenge der natürlichen Zahlen dargestellt werden, wobei $S_D \subseteq \{1, ..., n \in \mathbb{N}\} \stackrel{\text{def}}{=} \mathsf{N}$. Weiterhin sei $\mathfrak{F}$ die Menge aller Permutationen von $\mathfrak{N}$, sowie π eine gleichverteilte zufällige Auswahl aus $\mathfrak{F}$, dann gilt für jedes beliebige Shingle $x \in S_D$ und jedes beliebige Menge von Shingles $S_D \subseteq \mathsf{N}$:

$$\Pr[\min(\pi(S_D)) = x] = \frac{1}{|S_D|}$$

Ausformuliert bedeutet das, dass jedes zufällig gewählte Shingle die gleiche Chance hat, ein Minimum von S_D zu sein. Die Funktion $h : h(\theta) = \min(\pi(\theta))$ kann als Hashfunktion verstanden werden. Sie weist einer Menge von Shnigles eine eindeutige Kennzeichnung zu: das kleinste Element der Permutation π. BRODER ET AL. [BCFM00] bezeichnet diese Eigenschaft als *min-wise independant*. Die Wahrscheinlichkeit, dass die Hashfunktion zweier Dokumente kollidiert, soll dem Grad der Übereinstimmung der Dokumente entsprechen. Es gilt zu beweisen, dass

$$\begin{aligned}
\Pr[\min(\pi(S_A)) = \min(\pi(S_B))] &= \frac{|S_A \cap S_B|}{|S_A \cup S_B|} \\
\Pr[h(S_A \cup S_B) \in S_A \cap S_B)] &= \\
\sum_{x \in S_A \cap S_B} \Pr[h(S_A \cup S_B) = x] &= \\
\sum_{x \in S_A \cap S_B} \frac{1}{|S_A \cup S_B|} &= \frac{|S_A \cap S_B|}{|S_A \cup S_B|}
\end{aligned}$$

Die Wahrscheinlichkeit, dass ein Element x als das kleinste von $(S_A \cup S_B)$ ausgewählt wird ist $\frac{1}{(S_A \cup S_B)}$. In der Gesamtheit aller übereinstimmenden $x \in (S_A \cap S_B)$ ergibt sich die Summe als eine Abschätzung der $Jaccard$-Gleichung.

$|S_D|$ entspricht fast der Anzahl der Wörter von D. In einem Dokument D mit n Token und einer Shingle-Größe von ω gibt es $n - \omega + 1$ Shingles. Die Eigenschaft *min-wise independant* macht es möglich, für Dokumente kleine Samples aus unabhängig verschiedenen Hashfunktionen zu nutzen, um einen aussagekräftigen Wert der Übereinstimmung $Resemblance$ zu gewinnen. Für jedes Dokument D wird dazu eine Liste von k zufällig gewählten Permutationen π_k generiert, und als Sample $\tilde{S}_D$ von S_D verstanden.

$$\tilde{S}_D = (\min(\pi_1(S_D)), \min(\pi_2(S_D)), ..., \min(\pi_k(S_D)))$$

Diese Voraussetzungen machen es nun möglich, jedem Paar von m Dokumenten $(\tilde{S}_A, \tilde{S}_B)$ $A, B \in \{0, ..., m-1\}$ mit wenig Aufwand $(m \log m)$ einen repräsentativen Wert für die jeweilige Übereinstimmung zu ermitteln. Zu diesem Zweck wird eine Liste L mit den Paaren $\langle min(\pi(S_D)), D\rangle$ aller gehashter Shingles und dem Dokument, in dem sie auftreten, geführt. L wird nach Shinglehashs sortiert, so dass mehrfach vorkommende Shinglewerte nacheinander stehen. Zur Wiederholung: Jedes Shingle in S_D hat die gleiche Chance, für einen der k Repräsentanten von D gehalten zu werden. Um für jedes Dokumentenpaar die Anzahl der übereinstimmenden Shingles darzustellen, wird die Menge an 3-Tupeln $\langle A, B, c = 0\rangle$ initialisiert. Jedes gleiche Shingle in einem Dokumentenpaar erhöht den Wert c um 1. Diese Berechnung kann mittels *Mergesort*-Algorithmus der Tupel in linearem Aufwand durchgeführt werden. Im Resultat lässt sich c im Verhältnis zu k zufälligen Permutationen als den geschätzten Wert der $Resemblance$ ablesen.

Es ist anzunehmen, dass für einen Vergleich vieler Dokumente eine Gruppierung der Ergebnisse nach der berechneten Ähnlichkeit notwendig wird. Neben der Vielzahl an *Clustering*-Methoden kann einfachheitshalber festgelegt werden: überschreitet ein Paar (A, B) einen festgelegten Grenzwert, so wird eine Verbindung zwischen diesen Dokumenten angelegt. Alle miteinander verbundenen Dokumente sind Elemente eines Clusters.

Um Webdokumente syntaktisch zu clustern, wurde mit *AltaVista spider run* die Menge an 30 Mio. Webdokumenten, entsprechend 150GByte Datenmaterial, indiziert. Nach Normalisierung wurde für jedes Dokument ein Sample mit Shinglegröße $\omega = 10$ und 40-bit Fingerprint je Shingle, erhoben. Für die 600 Mio. extrahierten Shingles aller Dokumente müssen daher 3 GByte Speicherkapazität zur Verfügung stehen. Ob ein Dokumentenpaar (A, B) einander ähnlich ist, wurde bei einem Grenzwert $\tau = 0{,}5$ für $Resemblance(A, B)$ entschieden. Das Ergebnis waren 3,6 Mio. gebildete Cluster mit insgesamt 12,3 Mio. Dokumenten. Davon bestanden 2,1 Mio. Cluster aus (5,3 Mio.) ausschließlich identischen Dokumenten. Die verbleibenden 1,5 Mio. Cluster beinhalteten eine Mischung aus identischen und ähnlichen Dokumenten. Die Schlussfolgerung, dass mehr als 30% der Webdokumente sich auf rund 10% der Dokumente aufgrund ihrer Gleichheit, abbilden lassen, macht deutlich, wie stark Duplikate die Verarbeitung von Webanfragen erschweren. Dennoch dauerte der gesamte Prozess bei der Ausführung 1996 rund 10,5 CPU-Tage.

3.1.3. SmH Beispiel

Es seien die Dokumente A, B und C

$$
\begin{aligned}
A &= \texttt{A red rose is a rose.} \\
B &= \texttt{A red rose is a red rose.} \\
C &= \texttt{John Doe has a rose.}
\end{aligned}
$$

	0	2	4	6	11	15	23	27
π_1	21	10	12	3	30	7	25	1
π_2	12	5	3	8	18	16	9	26

Tabelle 3.2.: Zufällig gewählte Permuationen von $\mathsf{N} = \{0, 1, 2, ..., 31\}$

Dann ergeben sich für ein $\omega = 3$ Shingling die Menge der Shingles S_D mit einem $l = 5$ bit Fingerprint $f : S_D \rightarrow \{0, 1, \cdots, 2^l - 1\}$ aller Shingles. (Die Fingerprints wurden für dieses Beispiel frei vergeben, mehrfach vorkommende Shingles werden ignoriert.)

$$
\begin{aligned}
S_A &= \{f(A, red, rose) = 11, f(red, rose, is) = 2, f(rose, is, a) = 23, f(is, a, rose) = 4\} \\
S_B &= \{f(A, red, rose) = 11, f(red, rose, is) = 2, f(rose, is, a) = 23, f(is, a, red) = 15\} \\
S_C &= \{f(John, Doe, has) = 6, f(Doe, has, a) = 27, f(has, a, rose) = 0\}
\end{aligned}
$$

Für dieses Beispiel werden die Samples eines Dokumentes $\tilde{S_D}$ durch $k = 2$ Abbildungen repräsentiert, wobei π_1 und π_2 unabhängig gewählte Permutationen der Menge $\{0, 1, \cdots, 2^l - 1\}$ sind. In Tabelle 3.2 sind zufällig gewählte Permutationen abzulesen.

Daraus folgt schließlich

$$
\begin{aligned}
S_A = \{11, 2, 23, 4\} \Rightarrow \tilde{S}_A = (\min(\pi_1(S_A)) = 10, (\min(\pi_2(S_A)) = 3)) \\
S_B = \{11, 2, 23, 5\} \Rightarrow \tilde{S}_B = (\min(\pi_1(S_B)) = 7, (\min(\pi_2(S_B)) = 5)) \\
S_C = \{6, 27, 0\} \Rightarrow \tilde{S}_C = (\min(\pi_1(S_C)) = 1, (\min(\pi_2(S_C)) = 8))
\end{aligned}
$$

Aus der sortierten Liste L aller Shingles der Dokumente $\langle \pi_i^{-1}(\tilde{S_D}), Dokument \rangle$, lassen sich für jedes Paar gleicher Shingles die korrespondierenden Dokumente ablesen.

$$
L = (\langle 2, A\rangle, \langle 2, B\rangle, \langle 4, A\rangle, \langle 6, C\rangle, \langle 15, B\rangle, \langle 27, C\rangle) \Rightarrow \{\langle A, B, 1\rangle, \langle B, C, 0\rangle, \langle A, C, 0\rangle\}
$$

Da $k = 3$ stellt sich heraus, dass schätzungsweise Dokument A und B in $\frac{1}{2} \rightarrow 50\%$ Shingles übereinstimmen. Die Paare (A, C) und (B, C) weisen keine Übereinstimmung auf. Dieses Beispiel stellt nicht die Effizienz, sondern die prinzipielle Vorgehensweise von SMH dar. Wird die Größe l der Shingle-Identifikation zu klein gewählt, ist die Wahrscheinlichkeit höher, dass verschiedene Objekte den gleichen Fingerprint erhalten. In der Praxis soll $l = 64$ für Textanwednungen genügen [Bro00].

3.2. SmH für Bilddokumente

„[...]As techniques are developed for identifying features in other data types, there are no limits on the objects that can be compared for resemblance: images, video sequences, or databases."[BGMZ97] Mit dieser Aussage wird der Kernpunkt der Arbeit verdeutlicht. Genauer: Es geht um den Versuch, das SmH Prinzip auf eine Menge von Bilddokumenten anzuwenden. Bereits mehrere Forschungsprojekte haben den Vergleich von Bilddaten auf den Vergleich von Teilmengen extrahierter Dokumentmerkmale zurückgeführt. Einige von ihnen nutzten Methoden zur Extraktion von markanten Bildbereichen, um Bildinformationen kompakt zu repräsentieren und besser vergleichbar zu machen. In dieser Untersuchung stand die Aufgabe, Bildfeatures durch Shingling von Token zu gewinnen. Damit wird nicht den womöglich wichtigen mehr, sondern jedem Bildbestandteil die gleiche Wahrscheinlichkeit zugeordnet, als Repräsentant seiner Grundmenge extrahiert zu werden. Mit anderen Worten, es wird untersucht, ob es für einen eher allgemeinen Bildvergleich notwendig ist, dominante Bildpunkte zu identifizieren oder besser die Gesamtheit eines Bilddokumentes für den Vergleich zu betrachten. Dabei sollen Methoden für Anwendungsfälle gefunden werden, die statt mit komplexen Extraktionsverfahren, auch durch Shingling, mit deutlich geringerem Aufwand brauchbare Ergebnisse erzielen. Solch eine Anwendung wäre höher skalierbar und könnte im Zweifelsfall dennoch mit komplexeren Methoden kombiniert werden. Der Ansatz wäre einfach zu implementieren und außerdem leicht, beispielsweise mit Textanwendungen, zu verknüpfen.

Shingles werden aus Sequenzen von Wörtern gebildet [BAA]. Um dieses Prinzip für Bilddokumente zu übernehmen, muss geklärt werden, ob 2-dimensionale Bildinformationen als 1-dimensionale Sequenz so interpretiert werden können, dass wahrnehmbare Bildinformationen erhalten bleiben. Durch Shinglng bei [BAA] wird in Textdokumenten die Reihenfolge benachbarter Wörter beachtet - gewisserweise bleibt somit auch die Semantik des Textes in seiner Abbildung erhalten, da die Information in Texten üblicherweise durch die Wortfolgen formuliert wird. Bilder hingegen bestehen aus 2-dimensionalen Regionen von Farbwerten, wobei die Wahrnehmung eher sequenz- und zeitunabhängig funktioniert. Bildregionen beschreiben durch Muster, Formen und Farben den Bildinhalt. Es lässt sich daher vermuten, dass ein einzelnes statisches Bild menschlich nicht sequenziell wahrgenommen wird. Maschinell würde eine Überführung eines Bilddokumentes in eine shinglebare Sequenz genügen, jedoch ist unklar, inwiefern diese Überführung einen Informationsverlust verursacht. Diese Ungewissheit soll mit der Untersuchung verschiedener Überführungen genauer betrachtet werden.

Textduplikatlösungen verfolgen meist das Ziel, stark ähnliche Textdokumente identifizieren zu können. Die Einschätzung der Übereinstimmung soll dabei genau genutzt werden können. Da bei Bilddokumenten die Definition der allgemeinen Ähnlichkeit

ohnehin nur vage formuliert werden kann, ist ein vergleichbarer Nutzten des Ergebnisses von *Resemblance* nicht zu erwarten. Auch die Normalisierung der Wortmenge von Textdokumenten ist konzeptionell mit Bilddokumenten nicht vereinbar. Mit lexikalischen oder grammatikalische Regeln lässt sich die Menge der Worte, zum Beispiel durch Stemming oder Lemmatisierung natürlicher Wörter, mit wenig Informationsverlust komprimieren. Vergleichsweise reduzierte 2-dimensionale bildliche Informationen könnten jedoch nur schwer in sequenzieller Verarbeitung gewonnen werden.

4. Verwandte Arbeiten

4.1. Bildduplikaterkenung

4.1.1. Chum et al. 2004 & 2007

Mit dieser Arbeit der University of Oxford wurde eine bis dato neue Kombination aus Locality Sensitive Hashing und SIFT-Deskriptoren zur Erkennung von ähnlichen Bildern sowie ähnlich bildlichen Aufnahmen präsentiert [PIZ+07]. Diese sind in zwei Klassen definiert: (i) Bilder, die als identisch wahrgenommen werden (mit leichten Veränderungen in Bildrauschen, -größe und -effekten), und (ii) Bilder der selben räumlichen Szene und unterschiedlichen Aufnahmeparametern: Objektiv, Position im Raum, Belichtungszeit,.... Das Augenmerk liegt auf der Skalierbarkeit der Anwendung, da häufig sehr große Bilddatenbanken vorliegen. Zu diesem Zweck müssen Bilder kompakt dargestellt werden - hier mit zwei unterschiedlichen Verfahren: Einmal unter Verwendung von Farbhistogrammen (Abbildung 4.1), bei denen der per Local Sensitive Hashing berechnete euklidische Abstand zwischen den Deskriptoren als Maß zur Ähnlichkeit verwendet wird. Das andere Konzept basiert auf mit DoG extrahierten und per SIFT quantisierten Features, die die Szene als Bag-Of-Words repräsentieren und mittels min-Hash mehrere Dokumente in linearem Zeitaufwand effizient vergleichen können (Abbildung 4.2). Diese zweite Methode wurde in [PZ07] in der Featureextraktionsmethode, nun auch MSER und Multi-Scale Hessian Interest Points, sowie in der Vergleichsoperation um TF-IDF erweitert. Hiermit können im Dokument die extrahierten Features im Verhältnis zu der Gesamtheit aller Dokumente gemessen werden. Die Leistung des Programms wurde mit der TrecVid2006 Datenbank sowie mit Dokumenten der University of Kentucky Database untersucht.

4.1.2. Yan Ke et al. 2004

Viele Verfahren repräsentieren die zu untersuchenden Dokumente mit je einem hochdimensionalen Vektor, bestehend aus Beschreibungen der extrahierten Features. [KSH] zeigt, dass es möglich ist, die Ähnlichkeitsanfrage effizienter zu gestalten, indem zusätzlich global Tabellen gespeichert und organisiert werden. So kann eine hohe Zahl an lokalen Features identifiziert und vor allem unabhängig voneinander indiziert werden. Dazu werden Interest Points mit DoG lokalisiert und anschließend mit PCA-SIFT, einer Weiterentwicklung des vorgestellten SIFT, beschrieben. Die globalen Tabellen enthalten: eine Liste der IDs und Dateinamen der Bilddokumente, eine Liste aller Interest Points der Bilder sowie eine (per

Abbildung 4.1.: Bilder der TrecVid 2006 Datenbank. In der ersten Reihe befinden sich die Bilder, die zur Ähnlichkeitsanfrage gestellt wurden. Die benachbarten Zeilen zeigen ähnliche Bilder, mit ihrem Abstandswert nach der Histogramm-Methode

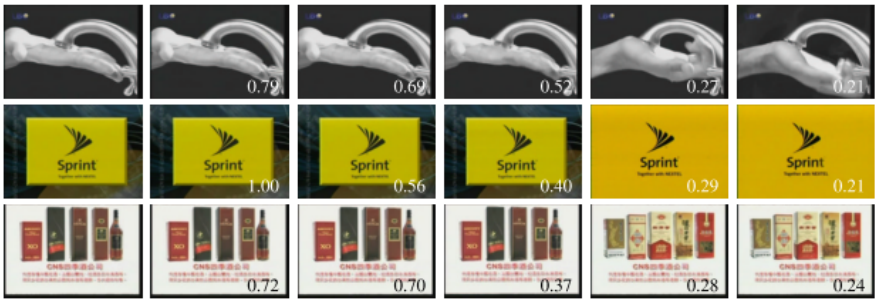


Abbildung 4.2.: In der ersten Reihe die Bilder zur Ähnlichkeitsanfrage. Die benachbarten Zeilen zeigen ähnliche Bilder, mit ihrem Abstandswert nach der SIFT-minHash-Methode

LSH gebildete) Hash-Tabelle mit Verweisen von den Bild-IDs zu den Interest Points. Eine Anfrage zur Suche nach ähnlichen Bildern in einer Datenbank beginnt mit der Analyse des Quellbildes nach Interest Points. Für jeden Punkt mit Schlüsselfunktion wird der SIFT-Deskriptor erstellt und sein LSH errechnet. Anschließend werden die Hashwerte sortiert und in der Hash-Tabelle nach dem entsprechenden ähnlichsten Deskriptor gesucht. Das Ergebnis ist die Liste aller Bilddokumente mit ihren referenzierten Interest Points. Wird auch diese Liste sortiert, so können die Ergebnisdokumente statistisch ausgewertet und abschließend das Resultat ausgegeben werden. In Anwendung dieses Verfahrens wurden aus einer Datenbank[1] mit 6261 Kunstbildern zufällig 150 Bilder ausgewählt und jedes dieser gewählten Bilder abwechselnd mit 40 Transformationen gefiltert. Somit entstand ein Testkorpus mit 12111 Bildern, von denen 6000 Bilder jeweils 150 Bildern ähnlich sind. Die verwendeten Transformationen sind unterteilt in *Standard* und *Difficult*. Für *Standard*-Transformationen wurden Bewertungen *Recall*: 99,85% - *Precision*: 100% erreicht, was fast einem perfekten Ergebniss entspricht. Auch die Transformationen *Difficult* erreichten vergleichbar gut Ergebnisse (*Recall*: 98,4% - *Precision*: 99,86%).

4.1.3. Wang et al. 2006

Diese Arbeit wurde durch die aktuelle Entwicklung angeregt, Bilder heute als eines der meistgenutzten Medienformate im Internet zu finden. Die Anzahl verfügbarer Bilddateien wächst schnell in wenigen Jahren [WJL+]. Digitale Kameras kommen öfter denn je zum Einsatz, und im Internet werden immer mehr Suchdienste und Plattformen zur Veröffentlichung angeboten. Der Algorithmus soll vor allem visuell identische, nicht nur gleiche Bilder in einer Dokumentmenge finden. Unter „visuell Identisch“ werden hier skalierte, farblich veränderte und verschieden formatierte Bilddokumente verstanden. Das Verfahren ist in vier Phasen geteilt: (i) Jedes Bild wird in einen hochdimensionalen Featurevektor, aus durchschnittlichen Grauwerten eines $N \times N$ Block des Bildes, konvertiert. (ii) Entstandene Vektoren projizieren sich durch Hauptkomponentenanalyse auf einen tiefdiemnsionalen Unterraum, dem (iii) ein K-bit Hash zugeordnet wird. Dieser Hashwert wird durch die Abfrage, ob eine Komponente einen Durchschnittswert überschreitet, generiert. Abschließend (iv) werden die gehashten Werte gruppiert, indem die Summe der Hamming-Distanzen der Hashwerte ermittelt wird. Diese Methode ist auf wenige Varianzen von Bildduplikaten beschränkt, erweist sich jedoch in besonders großen Dokumentmengen als sehr effektiv. Für 100000 Dokumente beträgt die Zeit zum Gruppieren nur $0{,}4$ Sekunden, bei einer *Precision* von $> 98\%$.

[1] http://cgfa.acropolisinc.com/ 2010

4.2. Textduplikaterkennung

4.2.1. Moses S. Charikar 2002

Im Fall von Textduplikaterkennung werden gewöhnlich Wörter oder textliche Einheiten unterteilt und somit das Textdokument als Menge von Token repräsentiert. CHARIKAR stellt ein Prinzip vor, mit welchem die Ähnlichkeit von Datenmengen geschätzt werden kann [Cha02]. In Dokumenten wird für jedes Token ein b-dimensioaler Vektor erstellt, indem die Summe aus den b-dimensionalen Projektionen aller Token berechnet wird. Nach Normalisierung auf den Bereich [0,1] hat dieser Vektor die Eigenschaft, dass die Cosinus-Ähnlichkeit, eine Variante des Jaccard-Indexes, zweier Dokumente sich proportional zu der Anzahl der Bits verhält, in denen die Projektionen der Texte übereinstimmen.

4.2.2. A. Chowdhury et al. 2002

Einen anderen Ansatz zeigt das in [CFGD02] vorgestellte Verfahren I-Match. Es existiert für jeden Term in der Dokumentmenge ein IDF-Gewicht, nachdem jedes Dokument durch Stemming so vorverarbeitet wurde, dass nur jene Terme erhalten bleiben, die sich als informationsreich erwiesen haben. Häufig auftretende Wörter wie Bindewörter, Artikel oder Pronomen gelten als vernachlässigbar. Um mit einem Hash der gefilterten Dokumente eine Vergleichsoperation durchführen zu können, müssen die Terme der Dokumente sortiert werden. Jedes $\langle DocID, Hash \rangle$-Paar wird in einer Baumstruktur gespeichert. Suchanfragen können dadurch in logarithmischen Aufwand beantwortet werden: jede Kollision der Hashwerte ist eine Repräsentation ähnlicher Dokumente, entsprechend dem Prinzip des Locality Sensitive Hashings.

4.2.3. Manber 1993

In dieser Arbeit [Man94] wird eine Methode vorgestellt, die konzipiert wurde um gleiche Dateien zu finden, die, im Kontext der Ähnlichkeit, in vielen Teilen ihres Inhaltes übereinstimmen. Die Motivation dieser Arbeit entstand durch die Problematik Daten, oder nur deren Bestandteile, die der selben Quelle entspringen auch in großen Datenmengen identifizieren zu können. Zu diesem Zweck wird ein Verfahren vorgestellt, welches an die Idee des LSH heranführt: Approximate Fingerprinting. Das Ziel ist es in Dokumenten verschiedenen Untersequenzen, unabhängig voneinander, Hashwerte zu verteilen. Um relevante Untersequenzen zu ermitteln, wird von einem bestimmten Teilwort als Fixsequenz x ausgehend, die Datenmenge auf vorkommende x untersucht. Für jede Stelle, an der x in einer Sequenz auftritt, wird eine festgelegte Anzahl folgender Zeichen zur Bildung eines Fingerprints benutzt. Die Kombination mehrerer Fixsequenzen macht es möglich, ähnlichen Texten mit hoher Wahrscheinlichkeit mehrere gleiche Hashwerte zuzuordnen. Die Schwierigkeit besteht in der Auswahl der Fixsequenzen, die statistisch erhoben werden muss.

5. Implementierung

Die Untersuchungen in dieser Arbeit nutzen eine Implemetierung von GULLI[1]. Darum soll folgendes Kapitel darstellen, wie der Algorithmus darin realisiert wurde. Der Pseudocode 1 illustriert den groben Programmablauf.

Für jedes Dokument soll ein Merkmalsvektor mit k Elementen gehalten werden. k entspricht der Anzahl von Samples die zur Repräsentation eines Textes gehalten werden. Ausgehend von einer Primzahl p werden zwei Vektoren $\overline{A} = (a_0, a_1, \cdots, a_k)$ und $\overline{B} = (b_0, b_1, \cdots, b_k)$ mit zufällig gewählten Zahlen aus $\mathbb{Z}_p$ initialisiert. Zusätzlich wird ein Vektor $\overline{m} = (m_0, m_1, \cdots, m_k)$ erstellt, dessen Elemente $\forall i \in \{0, 1, \cdots, k\} \quad m_i = p$ sind. Das Shingling verläuft erwartungsgemäß nach der Methode, die in [BAA] vorgestellt wurde. Jedes Dokument wird normalisiert und iterierbar in Token aufgeteilt, wobei jedes Token eine Identifikationsnummer $\gamma \in \mathbb{N}$ per Hashfunktion der `boost`[2]-Bibliothek erhält. Die Summe der Token-IDs in einem Shingle ergibt eine Shingle-ID $\Gamma = \sum \gamma$. Während des Programms wird daraus, nach Iteration über alle Shingles, ein Vektor $\overline{sh}$ mit k unabhängigen Hashwerten generiert.

$$\forall i \in \{0, 1, ..., k\} \quad \overline{sh}_i = ((a_i \times \Gamma) + b_i) \mod p$$

Durch diesen Term werden in der Implementierung die Hashfunktionen für die k unabhängigen Permutationen von π umgesetzt [BCFM00]. Aus der Menge der gehashten Shingles eines Dokumentes $\{\overline{sh}^0, \overline{sh}^1, \cdots, \overline{sh}^s\}$ entsteht für jedes Dokument ein repräsentativer Vektor $\overline{D}$, der durch die kleinsten (*min-wise independant*) Elemente für jedes der k Indizes gebildet wird. Jeder gebildete Hashwert ist Element aus $\mathbb{Z}_p$ und somit anfangs immer kleiner als die Elemente in $\overline{m}$, die mit p initialisiert worden sind.

$$\forall i \in \{0, 1, ..., k\} \quad \overline{D}_i = min_i(\overline{m}_i, \overline{sh}_i^0, \overline{sh}_i^1, \cdots, \overline{sh}_i^s)$$

Jedes Shingle hat somit die gleiche Wahrscheinlichkeit, für einen der k Repräsentanten eines Dokumentes in dem gebildeten Sample $\overline{D}$ gehalten zu werden. k ist die Größe des von [BAA] beschriebenen Featurevektors für jedes Dokument. Werden die Samples aller Dokumente konkateniert und sortiert, ist eine praktische Datenstruktur gegeben, um eine

[1] http://www.di.unipi.it/%7Egulli/coding/shingles_v1.0.tgz 2010
[2] http://www.boost.org/ 2010

Algorithmus 1 Gulli's SmH

```
1: init all_document_signatures // Liste aller Dokument-Hash Paare
2: init p ← 2^31 − 1
3: init number_of_signatures // die Größe des Hashvektors: k
4: init A[number_of_signatures]
5: init B[number_of_signatures]
6: init MIN[number_of_signatures]
7: for i = 1 to number_of_signatures do
8:    A[i] ← random mod p
9:    B[i] ← random mod p
10:   MIN[i] ← p + 1
11: end for
12: for all documents do
13:   tokenize(document)
14:   create_shingles(tokens)
15:   for all shingles ∈ document do
16:      shingle_ID ← sum_of_hashed_words
17:      for j = 1 to number_of_signatures do
18:         shingle_hash ← (A[j] * shingle_ID + B[j]) mod p
19:         if shingle_hash < MIN[j] then
20:            MIN[j] ← shingle_hash
21:         end if
22:      end for
23:   end for
24:   for k = 1 to number_of_signatures do
25:      pair ← makepair(MIN[k], document)
26:      all_document_signatures.add(pair)
27:   end for
28: end for
29: sort(all_document_signatures) // Sortiert den Vektor nach Hashwerten
30: for all pair ∈ all_document_signatures do
31:   pair1 ← pair
32:   for all pair ∈ all_document_signatures and pair ≠ pair1 do
33:      pair2 ← pair
34:      if pair1(min) = pair2(min) then
35:         sim(pair1(doc), pair2(doc)) + +
36:      end if
37:   end for
38:   pair1 ← pair2.next
39:   scale_similarities() // übereinstimmende Hashwerte im Verhältnis zur Größe des
      Hashvektors
40: end for
```

Schätzung der Übereinstimmung aller Dokumentenpaare zu erheben.

Bei der Übertragung der sortierten Liste aller $\langle Shingle, Dokument\rangle$-Paare in die Menge der $\langle Dokument_1, Dokument_2, |Shingles|\rangle$-Tupel wird eine Schwäche der Implementierung deutlich (Zeile 38 im Pseudocode 1) Es muss jedes Paar verschiedener Dokumente mit gleichen Shinglehashs erfasst werden, unter der Bedingung, dass jeder der k Hashwerte separat behandelt wird. Die Liste, beispielsweise, $(\langle 0, A\rangle, \langle 0, B\rangle, \langle 0, C\rangle)$ führt nach der Implementierung zu $\{\langle A, B, 1\rangle, \langle A, C, 1\rangle\}$, wobei das Tupel $\langle B, C, 1\rangle$ darin fehlt. Aber auch die Liste $(\langle 0, A\rangle, \langle 0, A\rangle, \langle 0, B\rangle, \langle 0, B\rangle)$ sollte statt $\{\langle A, B, 4\rangle\}$, die Menge $\{\langle A, B, 2\rangle\}$ ergeben. Letzterer Fall entsteht nur, wenn für gleiche Shingles verschiedene Hashfunktionen π den selben Wert ergeben. Ein weiterer Punkt, der nicht der Beschreibung in [BAA] entspricht, ist das Bilden der Summe von Worthashes, um Shingles numerisch zu identifizieren (Zeile 16 im Pseudocode 1). Addition ist kommutativ, daher wird in der gegebenen Implementierung auf die Reihenfolge der Wörter in einem Shingle verzichtet. Man kann annehmen, dass im Fall vom Textvergleich diese Tatsache vernachlässigt werden kann, da durch die Vielzahl natürlicher Wörter auch Permutationen der Elemente eines Shingles toleriert werden können. Da das im Fall von Bildverarbeitung das nicht angemessen scheint, wurde der Code dort korrigiert: Nicht eine Hashfunktion einzelner Wörter γ, sondern das konkatenierte Shingle wird zur Bildung einer Shingle-ID Γ verwendet.

Zum Überprüfen der Implementierung seien die Textdokumente $A, B,$ und C gegeben.

$$
\begin{aligned}
D_A &= \texttt{a red rose is a rose.} \\
D_B &= \texttt{a red rose is a red rose.} \\
D_C &= \texttt{john doe has a rose.}
\end{aligned}
$$

Die Parameter zur Berechnung der $Resemblance$ werden hier mit $k = 3, \quad \omega = 2$ sowie $p = 2^7 - 1 = 127$ festgelegt. Das Resultat besagt $Resemblance(A, B) = \frac{3}{3} \rightarrow 100\%$. Die Paare (A, C) und (B, C) weisen keine Übereinstimmung auf. Das richtige, zu erwartende Ergebnis wäre $Jaccard(A, B) = \frac{4}{5} \quad J(A, C) = \frac{1}{8}$ sowie $J(B, C) = \frac{0}{8}$, womit der entstandene Durchschnittsfehler bei $\epsilon = \frac{1}{6}$ liegt.

Die unternommenen Modifikationen, um Bilddokumente mit diesem Programm zu vergleichen, sind in der Dokumentvorverarbeiung umgesetzt worden. Dazu lieferte die `OpenCV`[3] Bibliothek eine umfangreiche Sammlung an üblichen Methoden zur Bildverarbeitung.

[3] http://opencv.willowgarage.com/wiki/ 2010

6. Konzept für Bildduplikaterkennung mit SmH

Zum Erzeugen von Shingles [BAA] muss das Dokument als eine Sequenz von Symbolen oder Wörtern interpretierbar sein. Für die Übersetzung eines Bilddokuments B in eine shinglebare Sequenz Q werden nun verschiedene Überführungen $c : B_{x,y} \rightarrow Q_i$ vorgestellt. Nach der Überführung aller Bilddokumente in einen Sequenz von Token folgt der SmH-Algorithmus, der im gegebenen Programm implementiert wurde (Kapitel 5). Mit dem Programm sollen vor allem solche Dokumentenpaare identifiziert werden, die wenig Unterschiede aufweisen. Dementsprechend verfolgen die Konzepte den Ansatz, Bilddokumente so zu reduzieren, dass gerade die Unterschiede in ähnlichen Bilddaten nach Überführung geringer werden.

Alle vorgestellten Konzepte beziehen sich ausschließlich auf den Luminanzkanal der Bilder. Farbinformationen werden somit vorerst ignoriert, sind aber analog dazu kanalweise berechenbar. Bilddokumente bestehen aus $X \times Y$ Bildpunkten, wobei $\forall x \in \{0, 1, ..., X\} \wedge \forall y \in \{0, 1, ..., Y\}$ gilt $B_{x,y} \in \{0, 1, 2, ..., 255\} \subseteq \mathbb{N}$, womit der Wert des Bildpunktes an der Stelle (x, y) notiert wird. Der Bildursprung $B_{0,0}$ sei auf die obere linke Ecke des Bilddokuments festgelegt.

6.1. Bildpunkte in Bildshingles

Ein naheliegender Ansatz ist es, ein Bilddokument als die Sequenz Q_i von Bildpunkten zu betrachten. Zum Beispiel zeilenweise, angefangen an der linken oberen Ecke bis zum Ende in der unteren rechten Ecke des Bildes. Da jeder Helligkeitswert eines Pixels in einem Bild pro Farbkanal mit $8bit$ kodiert wird, gibt es vorerst ein Wörterbuch mit 256 verschiedenen Token. Da in ähnlichen Bildern nicht garantiert ist, dass die Mehrheit der Werte von Bildpunkten übereinstimmt, soll ebenfalls untersucht werden, inwieweit eine Quantisierung auf eine kleinere Menge an Wörtern die Qualität der Cluster verbessert. Es gilt die Annahme: Ähnliche Bilder haben auch ähnliche Bildpunkte, demzufolge führt eine Quantisierung des Wertebereichs ähnliche Bilddokumente wieder zusammen. Sei $m_s : B \subseteq \{0, 1, ..., 255\} \rightarrow B \subseteq \{0, 1, ..., s\}$ eine Quantisierung, dann beschreibt die Skalierung $s(\cdot) = \lfloor \frac{255}{\cdot} \rfloor =$ die An-

zahl der s Quantisierungsstufen. Das hat den gleichen Effekt wie ein Schwellwertfilter der Bildpunkte. Abbildung 6.1 zeigt -quantisierte Bilder, die zur Veranschaulichung normalisiert wurden. Resultat dessen ist auch die Verkleinerung der Anzahl möglicher Shingles: Bei einer Anzahl von x verschiedenen Token gibt es x^{ω} verschieden mögliche Shingles.

(a) 129 (b) 65 (c) 33 (d) 17 (e) 9 (f) 5 (g) 3

Abbildung 6.1.: PEPPER mit verschieden quantisierten Wertebereichen. Die Bildbezeichnung entspricht der Anzahl der verschiedenen Werte für Helligkeitspunkte

Interessant ist auch die Beziehung der Bildpunkte untereinander, die sich idealerweise auch in einem Shingle widerspiegeln sollte. Bei Textdokumenten enthalten Shingles benachbarte Wörter. Analog sollten Shingles eines Bilddokuments auch benachbarte Bildpunkte enthalten. Das Prinzip vom Shingling wird zu diesem Zweck auf auf zwei Dimensionen erweitert, indem $\omega - 1$ horizontal nach rechts und $\omega - 1$ vertikal nach unten angrenzende Wörter verwendet werden. In der Sequenz Q_i der Bildpunkte sind das, mit Wissen über Bildbreite X und Bildhöhe Y,

$$
\begin{aligned}
S_B = \{(& \\
& Q_{i+(0\times X)}, Q_{i+1+(0\times X)}, Q_{i+2+(0\times X)}, ..., Q_{i+\omega+(0\times X)}, \\
& Q_{i+(1\times X)}, Q_{i+1+(1\times X)}, Q_{i+2+(1\times X)}, ..., Q_{i+\omega+(1\times X)}, \\
& Q_{i+(2\times X)}, Q_{i+1+(2\times X)}, Q_{i+2+(2\times X)}, ..., Q_{i+\omega+(2\times X)}, \\
& ... \\
& Q_{i+(\omega\times X)}, Q_{i+1+(\omega\times X)}, Q_{i+2+(\omega\times X)}, ..., Q_{i+\omega+(\omega\times X)} \\
&)\}
\end{aligned}
$$

An einem Beispiel: Sei B das 4×3 große Bilddokument mit Bildpunkten, die als Token der Sequenz Q_i interpretiert werden. (Die Werte der Bildpunkte wurden frei gewählt.)

$$
B = \begin{pmatrix} Q_0 = 211 & Q_1 = 12 & Q_2 = 53 & Q_3 = 23 \\ Q_4 = 202 & Q_5 = 102 & Q_6 = 43 & Q_7 = 1 \\ Q_8 = 121 & Q_9 = 78 & Q_{10} = 12 & Q_{11} = 11 \end{pmatrix}
$$

dann ist das Dokument bei einem $\omega = 2$ Shingling durch die Menge

$$\begin{aligned} S_B = \{ & \\ & (211, 12, 202, 102), (12, 53, 102, 43), (53, 23, 43, 1), \\ & (202, 102, 121, 78), (102, 43, 78, 12), (43, 1, 12, 11) \\ & \} \end{aligned}$$

repräsentiert.

6.2. Frequenzen des Bildsignals

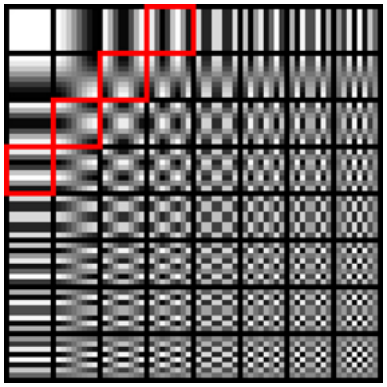
Diese Überführung bietet einen anderen Ansatz zur Wortbildung. Weil das Bilddokument als zweidimensionales diskretes Signal verstanden werden kann, sollen die spektralen Anteile der Bildfrequenz als Wörter zum Shingling genutzt werden. Zur Frequenzanalyse dienen Methoden der Fouriertransformation, in diesem Konzept die diskrete Kosinustransformation (DCT). Jeder Koeffizient eines DCT-transformierten Signals $\tilde{B}_{\tilde{x},\tilde{y}}$ beschreibt die Ausprägung einer zweidimensionalen Kosinusfunktion im Bildsignal. Die Gesamtheit aller Komponenten der Transformation setzt das Bild vollständig zusammen, wobei die Anzahl dieser spektralen Komponenten den Dimensionen des Bilddokumentes, $X \times Y$, entspricht. Fouriertransformationen ermöglichen es, nur bestimmte Frequenzanteile auszuwerten. Es kann davon ausgegangen werden, dass für einen Ähnlichkeitsvergleich wenige Repräsentative ausreichen. Bei der verwendeten DCT-Transformation wird die niedrigste Frequenz des Bildsignales, Gleichanteil oder DC-Koeffizient genannt, an der Bildstelle $(0, 0)$ dargestellt. Die horizontalen Nachbarn bilden die jeweils nächst höhere horizontale Frequenz ab, ebenso in vertikale Richtung, sowie für die Linearkombination beider Richtungen (AC-Komponenten). Es gilt folgende Annahme: Da die Kombination der $\tilde{x}$-ten horizontalen und $\tilde{y}$-ten vertikalen Basisfrequenz eines Bildsignals durch die Komponente an der Stelle $(\tilde{x}, \tilde{y})$ in der Transformation $\tilde{B}$ dargestellt wird, haben die Komponenten $\tilde{B}_{\tilde{x},\tilde{y}}$ die Eigenschaft, ein Signal mit der Frequenz vom Betrag $\tilde{x} + \tilde{y} = l$ zu repräsentieren. Sei D_l die Menge der Spektralkomponenten einer Diagonalen l (Abbildung 6.2)

$$\mathsf{D}_l = \{\tilde{B}_{(i,j)} \quad | \quad \forall i \in \{0, 1, ..., X\} \wedge \forall j \in \{0, 1, ..., Y\} \quad i + j = l\}$$

also zum Beispiel

$$\begin{aligned} \mathsf{D}_1 &= \{\tilde{B}_{0,1}, \tilde{B}_{1,0}\} \\ \mathsf{D}_2 &= \{\tilde{B}_{2,0}, \tilde{B}_{1,1}, \tilde{B}_{0,2}\} \\ \mathsf{D}_3 &= \{\tilde{B}_{3,0}, \tilde{B}_{2,1}, \tilde{B}_{1,2}, \tilde{B}_{0,3}\} \end{aligned}$$

Um eine Reihenfolge der Frequenzen beizubehalten, wird diese Eigenschaft beachtet und die Sequenz der Token aus den in diagonaler Reihenfolge abgelesenen Komponenten erstellt. Die DCT ergibt reellwertige Koeffizienten, die abhängig von der Bildgröße und der



(a) Basiskomponenten eines 8×8 Bildes

(b) $\tilde{B}_{0,3}$

(c) $\tilde{B}_{1,2}$

(d) $\tilde{B}_{2,1}$

(e) $\tilde{B}_{3,0}$

Abbildung 6.2.: Die Komponenten in D_3

Wertemenge des Bildsignals sind. Für jede Komponente wird ein Token aus zwei Zeichen gebildet, worin die Informationen enthalten sind, ob der Betrag eines Koeffizienten den Grenzwert g überschreitet (`T`, falls g überschritten wurde, `F`, wenn nicht) und welche der l-ten Linearkombination von Frequenzen durch diese Komponente beschrieben wird. Dazu wird zuvor eine Matrix W^g erstellt, welche diese Auswertung für alle Komponenten beinhaltet.

$$W^g_{i,j} = \begin{cases} l + \texttt{T} & g \geq |\tilde{B}_{i,j}| \\ l + \texttt{F} & g < |\tilde{B}_{i,j}| \end{cases}$$

Ergibt beispielsweise die Kosinustransformation eines Bilddokumentes B die Matrix $\tilde{B}$, dann entsteht bei einem Grenzwert $g = 100$ die Matrix W^{100} wie folgt

$$\tilde{B} = \begin{pmatrix} 1000 & -200 & 30 & 5 \\ 100 & -300 & 20 & 50 \\ 50 & 70 & 200 & 100 \\ -150 & 0 & 20 & 100 \end{pmatrix} \Rightarrow W^{100} = \begin{pmatrix} \texttt{0T} & \texttt{1T} & \texttt{2F} & \texttt{3F} \\ \texttt{1T} & \texttt{2T} & \texttt{3F} & \texttt{4F} \\ \texttt{2F} & \texttt{3F} & \texttt{4T} & \texttt{5T} \\ \texttt{3T} & \texttt{4F} & \texttt{5F} & \texttt{6T} \end{pmatrix}$$

Sei weiterhin s die kleinere Seite eines Bildes. $s = \min(X, Y)$. Es werden nur die Komponenten aus D_l beachtet, die $l < s$ erfüllen. Somit entsteht eine Sequenz Q aus den Wörtern

der Komponenten in W, die diagonal ausgelesen werden.

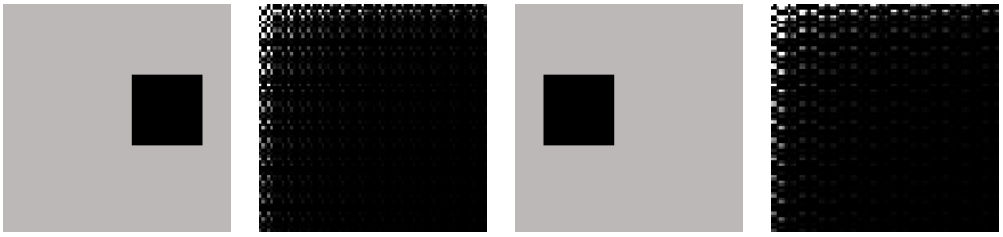
$$Q = (\texttt{0T}, \texttt{1T}, \texttt{1T}, \texttt{2F}, \texttt{2T}, \texttt{2F}, \texttt{3T}, \texttt{3F}, \texttt{3F}, \texttt{3F}, ...)$$

Das Ziel dieser Herangehensweise ist es, nur die Existenz, nicht die Amplitude einer Komponente, aber ihren Zusammenhang zu ihren benachbarten Frequenzen als Features des Dokumentes zu extrahieren.

Die Sequenz Q wird dem ursprünglichen Shingle-Prinzip unterzogen, da angenommen wird, dass eine Nachbarschaft mit dem diagonalen Auslesen bereits verwirklicht wurde und horizontal beziehungsweise vertikal angrenzende Komponenten nicht die gewollte Abhängigkeit besitzen.

6.3. Frequenzen von Bildblöcke

Hier werden die ersten beiden Ansätze miteinander kombiniert. Das Konzept aus Abschnitt 6.1 ist anfällig für Veränderung der Helligkeitswerte, das in Abschnitt 6.2 reagiert intolerant gegenüber Veränderungen im strukturellen Bildaufbau, beispielsweise durch Verschieben der Bildszene (ein Beispiel dazu ist in Abbildung 6.3 illustriert). Diese Überführung der Bildinformationen in eine Sequenz soll einen Ausgleich dazwischen auffinden. Eine örtliche Nachbarschaft der Bildpunkte $B_{x,y}$ bleibt erhalten, und für benachbarte Regionen werden aus Eigenschaften des Frequenzverhaltens Features zur Wortbildung extrahiert. Zur Ein-



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

Abbildung 6.3.: Die Verschiebung eines Objektes im Bild verursacht auch Veränderungen der Spektralkomponenten einer DCT.

teilung in Regionen wird das Bild vorerst in benachbarte Blöcke $B'_{x',y'}$, der Größe $b \times b$ aufgeteilt, so dass das Bilddokument nun als Matrix von Bildblöcken verstanden werden

kann.

$$B_{x,y} = \begin{pmatrix} B_{0,0} & B_{1,0} & B_{2,0} & ... & B_{X,0} \\ B_{0,1} & B_{1,1} & B_{2,1} & ... & B_{X,1} \\ B_{0,2} & B_{1,2} & B_{2,2} & ... & B_{X,2} \\ ... & ... & ... & ... & ... \\ B_{0,Y} & B_{1,Y} & B_{2,Y} & ... & B_{X,Y} \end{pmatrix} \Rightarrow B_{x,y} = \begin{pmatrix} B'_{0,0} & B'_{1,0} & B'_{2,0} & ... & B'_{X',0} \\ B'_{0,1} & B'_{1,1} & B'_{2,1} & ... & B'_{X',1} \\ B'_{0,2} & B'_{1,2} & B'_{2,2} & ... & B'_{X',2} \\ ... & ... & ... & ... & ... \\ B'_{0,Y'} & B'_{1,Y'} & B'_{2,Y'} & ... & B'_{X',Y'} \end{pmatrix}$$

Jeder Bildblock besteht aus den b nächsten horizontalen und vertikalen Bildpunkten. Sei $u = x' \times b$ und $v = y' \times b$, dann

$$B'_{x',y'} = \begin{pmatrix} B_{u+0,v+0} & B_{u+1,v+0} & B_{u+2,v+0} & ... & B_{u+b,v+0} \\ B_{u+0,v+1} & B_{u+1,v+1} & B_{u+2,v+1} & ... & B_{u+b,v+1} \\ B_{u+0,v+2} & B_{u+1,v+2} & B_{u+2,v+2} & ... & B_{u+b,v+2} \\ ... & ... & ... & ... & ... \\ B_{u+0,v+b} & B_{u+1,v+b} & B_{u+2,v+b} & ... & B_{u+b,v+Y} \end{pmatrix}$$

Ein Bild mit $X \times Y$ Bildpunkten ist nun ein Bild mit $X' \times Y'$ Blöcken der Größe $b \times b$, also $X' = \lfloor \frac{X}{b} \rfloor$ und $Y' = \lfloor \frac{Y}{b} \rfloor$. Nun soll über alle Blöcke eine Aussage bezüglich ihres Frequenzverhaltens ermittelt werden. Dazu sei das Verfahren aus Abschnitt 6.2 aufgegriffen: Basiskomponenten der Diagonalen D_l des kosinustransformierten Blockes bieten einen verwandten Informationsgehalt. Ziel ist es, jedem Block ein Token zuzuweisen. Dazu liefert jede Komponente aus $\{\mathsf{D}_1, \mathsf{D}_2, ...\mathsf{D}_b\}$ nicht ein einzelnen Token, sondern einen Buchstaben für das „visuelle“ Wort des Blockes. Die Koeffizienten der Kosinustransformation sind reellwertig. Diese Tatsache wird in dieser Überführung beachtet, indem nicht der Betrag der Koeffizienten, sondern ein negativer $-g$ und ein positiver $+g$ Grenzwert zur Wortbildung verwendet werden, und somit drei statt zwei Buchstaben bei der Tokenbildung zur Verfügung stehen: `P` für Komponenten über dem positiven Grenzwert, `N` für solche, die unter $-g$ liegen und `Z` für die Komponenten dazwischen. Dazu wird erneut eine Matrix W^g gebildet.

$$W^g_{i,j} = \begin{cases} \texttt{P} & +g < \tilde{B'}_{i,j} \\ \texttt{N} & -g > \tilde{B'}_{i,j} \\ \texttt{Z} & \text{sonst} \end{cases}$$

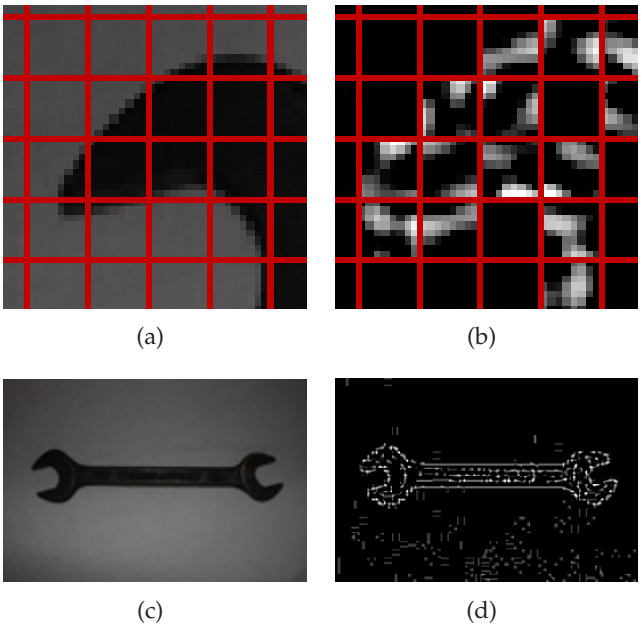
Sei zum Beispiel $\tilde{B'}$ kosinustransformierter Block der Größe 4×4 und ein Grenzwert bei $g = 50$ dann entsteht eine Matrix W^{50}

$$\tilde{B'} = \begin{pmatrix} 100 & 0 & 150 & 0 \\ -60 & 100 & 0 & -150 \\ -30 & 0 & -50 & 10 \\ -20 & 0 & 0 & 100 \end{pmatrix} \Rightarrow W^{50} = \begin{pmatrix} \texttt{P} & \texttt{Z} & \texttt{P} & \texttt{Z} \\ \texttt{N} & \texttt{P} & \texttt{Z} & \texttt{N} \\ \texttt{Z} & \texttt{Z} & \texttt{Z} & \texttt{Z} \\ \texttt{Z} & \texttt{Z} & \texttt{Z} & \texttt{P} \end{pmatrix}$$

W wird zur Wortbildung verwendet, indem in diagonaler Reihenfolge die Buchstaben ausgelesen werden. Jede Komponente aus $\{\mathsf{D}_0, \mathsf{D}_1, \mathsf{D}_2, ...\mathsf{D}_b\}$ liefert einen Buchstaben, demnach wird aus jedem Block ein Wort der Größe $\binom{b}{2}$ generiert. Im obigen Beispiel entsteht aus W das Wort für die Sequenz Q wie folgt

$$Q_i(W) = \texttt{PZNPPZZZZZ}$$

Diese Überführung kann visualisiert werden (Abbildung 6.4), indem jede Matrix W nach dem Grenzwertfilter rücktransformiert wird und vorher statt Buchstaben die Koeffizienten auf Werte $\{-1, 0, 1\}$ quantisiert werden. Durch die Aufteilung und blockweise Frequenzanalyse wird jeder Block auf möglichst aussagekräftige Frequenzen reduziert.



(a) (b)

(c) (d)

Abbildung 6.4.: Jedem Bildblock wird die aussagekräftigste Kombination an Basisfunktionen zugeteilt. (a) zeigt einen Auschnitt und Blöcke der Größe 10×10, (b) die blockweise Rücktransformation mit entsprechend zugewiesenen Komponenten. Das gesamte Bild (c) ist nach Überführung aller Blöcke wie (d) zu verstehen.

Das Verfahren lehnt sich gewisserweise an die Idee der JPEG-Komprimierung an [WEE+91]. Die Überführung zu Buchstaben ist sozusagen ein Äquivalent zur Quantisierungsmatrix. Aber es sollen nicht nur die redundanten Informationen vernachlässigt werden, die das menschliche Auge nicht wahrnehmen kann, sondern auch solche, die zum Vergleich von Bilddokumenten als überflüssig eingeschätzt werden.

Da die Wörter aus Blöcken entstanden sind, die nebeneinander liegen und somit eine örtliche Nachbarschaft im Bildsignal aufweisen, wird wieder (analog zu Abschnitt 6.1) ein zweidimensionales Shingling aus ω horizontal und ω vertikal angrenzende Wörtern verwendet.

6.4. Zusammenfassung

Die hier beschriebenen Konzepte zur Überführung der Bildinformation in eine shinglebare Sequenz aus Token sind vor allem auf Bildmengen einsetzbar, deren Ähnlichkeit auf gleichen zweidimensionalen Strukturen basiert. Neben den vielen Konfigurationsmöglichketen der SmH-Implemetierung ist es sehr wahrscheinlich, dass für ausgewählte Problemfälle die Parameter der Konzepte entsprechend genau eingestellt werden müssen, um qualitative Ergebnisse zu erzielen. Es werden hier keine räumlichen Informationen aus den Bilddaten gewonnen. Jede Rotation, Verzerrung oder Veränderung der Perspektive eines Bildes kann dem Algorithmus den Hinweis nehmen, dass eine Ähnlichkeit vorliegt, obgleich sie vom Benutzer erwartet werden würde. Die Konzepte werden im Anschluss evaluiert, um herauszufinden inwiefern diese Annahmen stimmen und wie stark ein Toleranzbereich gegenüber den genannten Schwächen der Konzepte ausgenutzt werden kann.

7. Versuchsaufbau

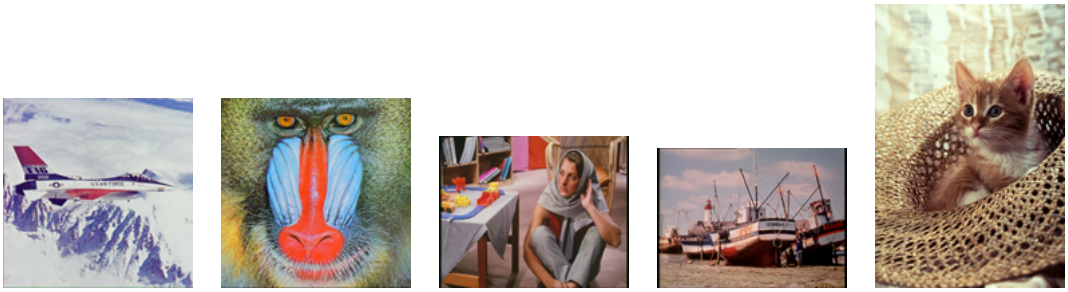
Für die Versuchsreihen wurden drei Testdatensätze erstellt, die aus jeweils 100 Dokumente bestehen. Es stehen drei Programme zur Verfügung, welche unter gleichen Bedingungen diese 100 Dokumente der richtigen Gruppe zuordnen sollen: Die OpenSource-Anwendung GQView, das Forschungsprojekt INDetector [ZEY], sowie das frei verfügbare Softwareprojekt von GULLI, welches den *SmH*-Algorithmus[BAA] implementiert und nun für die Erkennung von ähnlichen Bilddokumenten angepasst wurde. Die Qualität der gebildeten Cluster wird an den für *Information Retrieval* typischen Metriken *Precision* ,*Recall* und *F-Measure* messen. Alle Berechnungen wurden auf einem handelsüblichen Notebook (Athlon X2, 2 $\times$ 2GHz, 1GB RAM, Ubuntu 9.01) durchgeführt.

7.1. Testdokumente

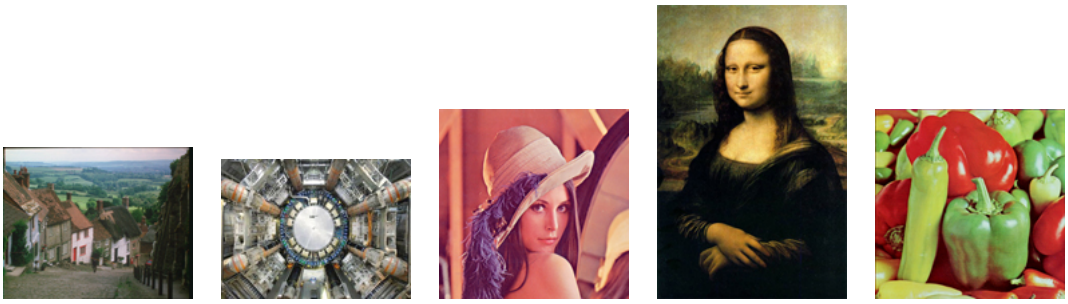
Jede Domäne eines Versuches besteht aus 10 Klassen mit je 10 Dokumenten, insgesamt 100 Dokumenten, für die die Ähnlichkeit untersucht werden soll. Dabei repräsentiert jeder Datensatz eine eigene Problematik, dessen Lösung separate Anforderungen an die Programme stellt.

FILTER wird auf der Basis von 10 Bilddokumenten gebildet. Jedes Basisdokument (Abbildung 7.1) wurde mit 9 gebrauchsüblichen Funktionen zur Modifikation von Bildinformationen verändert, darunter Punktoperatoren, die Bildpunkte separat verändern, sowie Filter, die ihre Umgebung in die Operation mit einbeziehen. Diese Dokumentenmenge soll Webanwendungen nahegebracht werden, die große Datenbanken an Bilddokumenten nutzen: Soziale Netzwerke, Foren, Suchmaschinen. Durch die Vielzahl an Nutzern, die im Internet Inhalte erstellen, werden auch viele Bilder kopiert und verändert wieder veröffentlicht, so dass sich eine Vielzahl von Bilddokumenten auf gleichen Bildquellen zurückführen lässt.

Abbildung 7.2 zeigt die Übersicht der genutzten Bildfilter in vergrößerten Ausschnitten der Bilder. Es wurden solche verwendet, die erfahrungsgemäß verbreitet Anwendung finden. Die Parameter dafür wurden so gewählt, dass der Effekt des Filters zwischen „erkennbar ersichtlich“ und „komplett verfremdet“ eingeordnet werden kann. Diese Wahl hatte das Ziel, auch starke Auswirkungen von Filtern zu untersuchen, dabei aber dennoch eine realistische Nähe zu Anwendungen zu gewährleisten.



(a) AIRPLANE (b) BABOON (c) BARBARA (d) BOATS (e) CAT

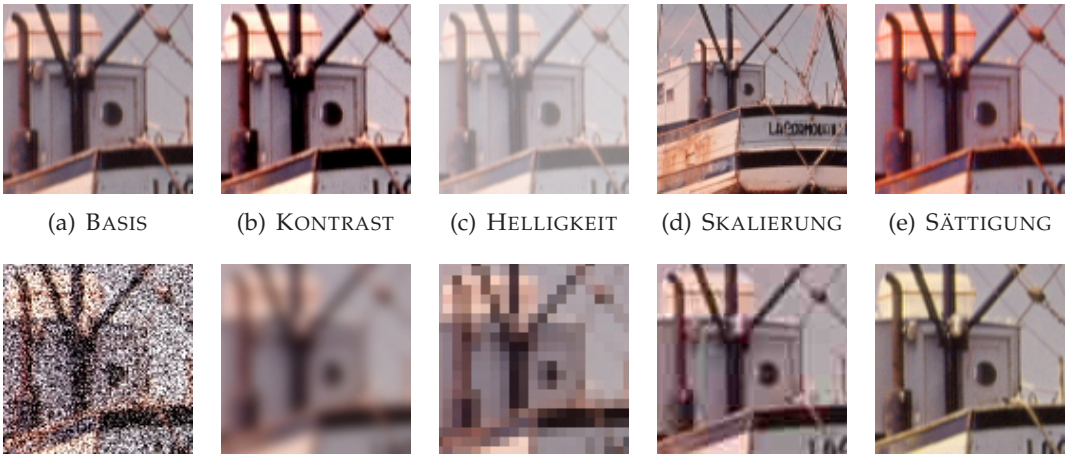


(f) GOLDHILL (g) SPEED (h) LENA (i) MONALISA (j) PEPPER

Abbildung 7.1.: FILTER Ursprungsbilder als Basisdokumente.

WEBCAM. Hiermit werden Sammlungen von Bildern, die durch öffentliche Webcams aufgenommen wurden, untersucht. Ausgangspunkt für die Analyse dieser Testdokumente ist die Annahme, verschiedene Bilder seien an gleichen Punkten und unter gleichen technischen Bedingungen aufgenommen. Wetterlagen, Lichteinfluß und Veränderungen der Szene im Aufnahmemoment variieren das entstandene Bild. Beispiele von zehn Aufnahmen einer öffentlichen Webcam über Berlin sind in Abbildung 7.4 aufgezeigt. Die statischen Bestandteile der Aufnahme, Gebäude, Straßen und Umwelt, sollen dafür sorgen, dass mit der Ähnlichkeitsanfrage jedes Bild zu seinen richtigen Korrespondenzen zugeordnet wird. Ausgehend von 10 Aufnahmen durch je 10 Webcams verschiedener Orte (Abbildung 7.3) ist dieser Testkorpus entstanden.

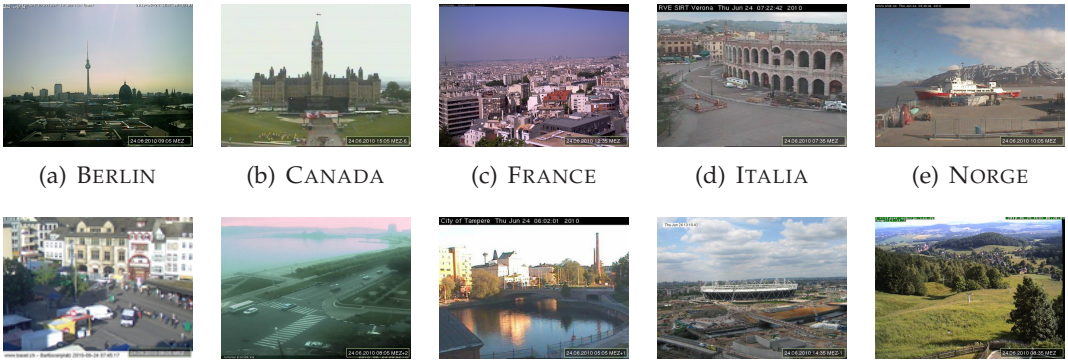
PHOTO repräsentiert einen Datensatz von Fotografien verschiedener Objekte. In diesem Fall wurden die Aufnahmen von „Hand“ gemacht, daher finden sich Variationen ähnlicher Bilddokumente vor allem in Belichtung, Fokus und Position des Motivs. Es war dabei wichtig, dass das Objekt von annähernd dem selben Ausgangspunkt geschossen, und dabei wenig Rotation, Zoom und Verschiebung der Kamera zugelassen wurde. Auf diese Weise wurde eine Menge von Bildern wiedergegeben, die vom Fotografen auch als ähnliche Bilder gemeint waren, jedoch, den natürlichen Umständen geschuldet, nicht identisch sind. Desweiteren sind alle Objekte auf einem homogenem Hintergrund fotografiert worden, haben also darin eine Gemeinsamkeit, die nach Definition nicht als ähnlich bewertet werden soll.



(a) BASIS (b) KONTRAST (c) HELLIGKEIT (d) SKALIERUNG (e) SÄTTIGUNG

(f) RAUSCHEN (g) GLÄTTEN (h) PIXEL (i) JPG LOW (j) FARBE

Abbildung 7.2.: BOATS Bilddokumente mit grafischen Filtern.



(a) BERLIN (b) CANADA (c) FRANCE (d) ITALIA (e) NORGE

(f) SUISSE (g) NOVOSIBIRSK (h) FINLAND (i) ENGLAND (j) ZITTAU

Abbildung 7.3.: WEBCAM Aufnahmen verschiedener Orte durch öffentlicher Webcams.

7.2. Qualität der Cluster

Zur Bewertung der Ergebnisse, die die Programme über den Testdatensätzen liefern, wird die Qualität der gebildeten Cluster der ähnlichen Dokumente evaluiert. Dazu finden sich in [MRS08] Verfahren, die anhand von richtigen und falschen Paarbeziehung innerhalb gebildeter Cluster die qualitative Güte misst: 1 für das perfekte beziehungsweise 0 das vollständig fehlerhafte Ergebnis. Idealerweise entspricht das Ergebnis den 10 Clustern mit den jeweils dazugehörigen 10 Dokumenten. Als passender Bewertungsmaßstab kamen die Metriken *Recall*, *Precision* und deren Verhältnis *F-Measure*in Betracht.

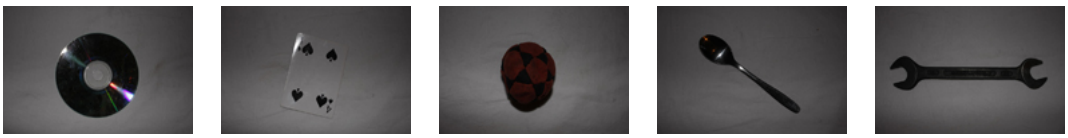
Zur Erhebung des *F-Measure* werden die Ergebnisse des Clusterings als Entscheidun-

(a) BERLIN #1 (b) BERLIN #2 (c) BERLIN #3 (d) BERLIN #4 (e) BERLIN #5

(f) BERLIN #6 (g) BERLIN #7 (h) BERLIN #8 (i) BERLIN #9 (j) BERLIN #10

Abbildung 7.4.: Aufnahmen einer öffentlichen Webcam in BERLIN.

(a) POT (b) UMBRELLA (c) BOOK (d) CABLE (e) TORUS



(f) CD (g) SPADES (h) BALL (i) SPOON (j) TOOL

Abbildung 7.5.: PHOTO: Fotografien von Objekten auf homogenem Hintergrund.

gen richtiger Paarzuweisung über alle möglichen Paare $\binom{N}{2}$ getroffen. Dazu muss ein ausgeglichenes Verhältnis aus *Precision* und *Recall* erhoben werden:

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision stellt die Genauigkeit des Clusterings dar, indem richtige Paare in den Clustern in Relation zu allen Paaren innerhalb der Cluster also $Precision = \frac{TP}{TP+FP}$ errechnet wird. *Recall* wiederum beschreibt die Trefferquote, wie viele richtige Paare im Verhältnis zur Gesamtheit aller Paare gefunden wurden $Recall = \frac{TP}{TP+FN}$. Jedes richtige Paar, was in verschiedenen Clustern vorkommt, verschlechtert den Wert für *Recall*. Als *Positiv (P)*-Paare werden alle die Paare an Dokumenten gezählt, die sich in einem Cluster befinden. Berechnet man dazu die Entscheidung *True Positive (TP)*, die Paare gleicher Klasse, die in gleichen Clustern vorkommen, ergibt die Differenz daraus $P - TP = FP$ alle *False Positive (FP)*-Paare. Analog dazu beschreibt die Anzahl an *Negative (N)*-Paaren die Paarzuweisungen von Dokumenten, die in verschiedenen Clustern auftreten, also bei n Dokumenten $N = \binom{n}{2} - P$. Als *False Negativ (FN)* gelten somit solch Paare, die zu einer Klasse gehören, sich aber in verschiedenen Clustern befinden. Die Differenz aus der Anzahl aller Paare einer Klasse zu

(a) POT #1 (b) POT #2 (c) POT #3 (d) POT #4 (e) POT #5

(f) POT #6 (g) POT #7 (h) POT #8 (i) POT #9 (j) POT #10

Abbildung 7.6.: Fotografien von POT.

den Paaren der Klasse, die sich in einem Cluster befinden, ergibt die Anzahl für FN.

Zur Verdeutlichung seien als Beispiel die Menge der Cluster $\Upsilon = \{v_1, v_2, v_3\}$ aus den Elementen der Klassen $\Phi = \{\clubsuit\heartsuit\spadesuit\diamondsuit\}$ gebildet worden, mit dem Ergebnis der $n = 12$ Elemente $v_1 = \{\clubsuit\clubsuit\heartsuit\heartsuit\heartsuit\}$ $v_2 = \{\diamondsuit\diamondsuit\clubsuit\}$ $v_3 = \{\spadesuit\spadesuit\spadesuit\diamondsuit\}$. Es ergibt sich: $P = \binom{5}{2} + \binom{3}{2} + \binom{4}{2} = 19$ sowie $TP = \binom{2}{2} + \binom{3}{2} + \binom{2}{2} + \binom{3}{2} = 8$ und somit $FP = 11$. Daraus lässt sich schließlich folgern: $FN = (FN(\clubsuit) = \binom{3}{2} - \binom{2}{2} = 2) + (FN(\heartsuit) = \binom{3}{2} - \binom{3}{2} = 0) + (FN(\diamondsuit) = \binom{3}{2} - \binom{2}{2} = 2) + (FN(\spadesuit) = \binom{3}{2} - \binom{3}{2} = 0) = 4$.

Dieses Beispiel gilt für die Annahme, dass alle Dokumente in Cluster gruppiert wurden. Allerdings besteht die Aufgabe der Programme darin, alle 100 gegebenen Dokumente zu clustern. Unter den Auswertungen der untersuchten Algorithmen ist das nicht garantiert, da nur solche Dokumente für eine Gruppierung in Betracht kommen, die einen Grenzwert τ überschreiten. Die, für die zu keinem Element eine Paarbeziehung ermittelt wurde, egal ob falsch oder richtig, sind beim Clustering nicht relevant. Für die Erfassung der Metriken muss demnach der Wert für FN so angepasst werden, dass davon ausgegangen wird, dass jede Klasse 10 Dokumente enthält.

$$\forall i \in \{1, 2..., c = 10\} \quad FN(k) = \binom{10}{2} - \binom{|v_i \cap \phi_k|}{2}$$

Daraus folgt jedoch: werden alle 100 Elemente einem Cluster zugeordnet, so wird diese Zuteilung mit *Precision*= 0,09 (Es befinden sich 10 Elemente im richtigen Cluster), *Recall*= 1 (alle Dokumente wurden geclustert; kein richtiges Paar ist in verschiedenen Clustern) und somit *F-Measure*= 0,17 bewertet. Dieser Wert gilt als Basisresulat. Alle F-Measure $<$ 0,17 sind Indiz für das Versagen einer Konfiguration.

7.3. INDetector

DONG-QING ZHANG ET AL. entwickelten ein Programm für *Near Image Duplicate Detection*, dass die Theorien aus [ZEY] umsetzt. Bilddokumente werden darin als Attribut Relational Graph (ARG) dargestellt. Mit dieser graphenbasierten Repräsentation der Szene werden Bildmerkmale, wie auch deren Beziehung zwischeneinander modelliert. Als Features des Dokumentes gelten *Harris*-identifizierte *Interest Points*. Jedes Feature wird als Knotenpunkt des Graphen behandelt und mit einem 17-dimensionalen Vektor beschrieben. Die Bewertung zur Ähnlichkeit zweier Dokumente ergibt sich aus den Kosten, die entstehen, wenn ein ARG in einen anderen transformiert wird. Diese Kosten werden stochastisch erhoben. Sei H eine zufällige boolsche Variable, entsprechend der Hypothese $H = 1$: zwei Bilder sind *Near-Duplicate* und $H = 0$: zwei Bilder sind nicht *Near-Duplicate*. Weiterhin sei Y_A die Menge aller extrahierten Features des *Attributed Relational*-Graphen G_A eines Bildes, dann ist die Ähnlichkeit zweier Graphen G_S und G_T definiert als das Verhältnis der Wahrscheinlichkeiten, ob zwei Bilder S und T als ähnlich bestimmt werden oder nicht.

$$INDsim(G_S, G_T) = \frac{\Pr(Y_T|Y_S, H = 1)}{\Pr(Y_T|Y_S, H = 0)}$$

Die Wahrscheinlichkeit der Transformation eines Graphen $Pr(Y_T|Y_S, H)$ wird als Entwicklung in zwei Schritten vorgenommen: Zuerst werden die Knotenpunkte aus G_S nach G_T kopiert und anschließend die kopierten Attribute entsprechend transformiert. Für jeden Schritt wird eine statistische Größe als Maß erhoben. Auf diese Art und Weise soll zum einen eine Möglichkeit zum automatisierten Erlernen von Ähnlichkeitsmaßen, zum anderen auch das Erkennen komplexerer Variationen zwischen zwei Szenen, wie etwa Bewegung oder Erscheinen neuer Objekte, geboten werden. ZHANG ET AL. definieren den Ähnlichkeitsbegriff wörtlich als *„[...]ein Bildpaar, von denen sich ein Bild an ein Duplikat des anderen Bildes annähert, dabei aber leichte Veränderungen während der Aufnahmeeinstellungen, Aufnahmezeit, Bearbeitung und Rendering[...]"*[ZEY] toleriert werden.

Das Programm verläuft in 4 Phasen:

1. Identifikation der *Harris-Corner*-Punkte (Kapitel 4). Das Bild wird vorerst mit einem Gauss-Filter der Stärke σ geglättet. Alle Bildpunkte oberhalb eines Grenzwertes τ und mit einem Mindestabstand r zum nächsten identifizierten Punkt gelten als *Interest Point*. Solche, die diese Bedingung nicht erfüllen, werden mit 0 überschrieben.

2. Features der *Interest Points*. Die 17 Werte des Featurevektors eines Schlüsselpunktes ergeben sich aus 2 Werten für die Position, 3 Werten für Farbeigenschaften und 12 Werten über örtliche Eigenschaften der Gabor-Transformation in diesem Punkt. Gabor-Transformation ist eine Fouriertransformation, die, ausgehend von einem Punkt, Informationen über Phase und Amplitude von Sinusfunktionen in einer Region des Si-

gnals liefert. Bei INDetector werden nur die Merkmale der Amplitude übernommen.

3. Konvertierung der Features. Dem Nutzer ist es überlassen, die Bilddokumente als *Bag of Parts*(BoP) oder *Attributed Relational Graph*(ARG) repräsentieren zu lassen. Letzterer unterscheidet sich dadurch, dass auch Kanten zwischen extrahierten Interest Points modelliert werden. Jede Kante wird mit einem 2-dimensionalen Feature versehen, welches die Differenz zwischen den Features von anliegenden *Interest Points* notiert. Zur Gewinnung dieser Daten werden Kanten zwischen allen möglichen Punkten gebildet [ZEY]. Die ARG-Methode ist deswegen im Ergebnis präziser, benötigt aber mehr Rechenaufwand. Bei BoP wird für jedes Dokument mit p Schlüsselpunkten eine $p \times 17$ große Datenrepräsentation erstellt, im Falle von ARG kommen dazu noch $\binom{p}{2} \times 2$ Werte für die Kanteninformation des Graphen.

4. Training. Es müssen manuell zwei Listen zum Training angelegt werden. Eine Liste P beinhaltet Dokumentenpaare, die als *Near Duplicate* definert werden, die andere N besteht aus Dokumenten, die keine *Near Duplicate*-Paare sind. Anhand dieser Information kann mit dem System ein Abschätzen der Wahrscheinlichkeit realisiert werden, welches auf dem stochastischen Prozess $sim(G_S, G_T)$, ob zwei Graphen *Near Duplicate* sind, basiert.

INDetector bietet zwar eine Batchfunktion, zum Überprüfen einer Liste von Ähnlichkeitsbeziehung von Bildpaaren. Es muss jedoch für einen vollständigen Vergleich aller Dokumente einer Gruppe jede Paarbeziehung separat untersucht werden. Darüberhinaus ist auch keine Clusteringmethode gegeben. Um dennoch einen Vergleich der Programme aussagekräftig zu gestalten, dient der Clusteringalgorithmus der SMH-Impementierung als Mittel zum Zweck.

7.4. GQView

GQView[1] ist ein Werkzeug zur Bildverwaltung mit der Möglichkeit, Dokumente einer Bildersammlung auf Ähnlichkeit zu überprüfen. Neben den trivialen Funktionen zum Vergleich der Dateigröße, -namen, -pfad und -prüfsumme, soll auch eine Methode genutzt werden können, die vor allem Bilder verschiedener Skalierung oder Komprimierung als „ähnlich“ identifiziert. Von den Entwicklern des Programms sind dafür auch Bilder gleichen Inhaltes vorgesehen, jedoch wird keine Analyse der Bildstruktur durchgeführt. Der Vergleich basiert auf Farbfeatures der Bilddokumente.

Jede Bilddatei wird dazu als Kombination der 3 Farbkanäle: Rot, Grün und Blau interpretiert und jeder Farbkanal wiederum in ein Raster $32 \times 32 = 1024$ Felder unterteilt. Somit ist ein Bilddokument B durch drei 1024-dimensionale Vektoren $\overline{\mathsf{R}}(B)$, $\overline{\mathsf{G}}(B)$ und $\overline{\mathsf{B}}(B)$

[1] http://gqview.sourceforge.net/ 2010

bestimmt. Jedes Element dieser Vektoren spiegelt den durchschnittlichen Helligkeitswert eines Feldes im entsprechenden Farbkanal wider. Zur Messung der Ähnlichkeit zweier Bilddateien B_1 und B_2 gilt der Betrag der Differenz aller Vektoren als Maßstab.

$$GQsim(B_1, B_2) = 1 - \frac{\sum_{i=1}^{1024} |\mathsf{R}_i(B_1) - \mathsf{R}_i(B_2)| + |\mathsf{G}_i(B_1) - \mathsf{G}_i(B_2)| + |\mathsf{B}_i(B_1) - \mathsf{B}_i(B_2)|}{1024 \times 3 \times 255}$$

Im Zähler des Quotienten steht 1024 für die Anzahl der Felder, 3 für die Anzahl der Kanäle und 255 für den Wertebereich der Bildpunkte. Es gilt die Annahme: Bilder gleichen Inhalts, lassen sich auf gleiche durchschnittliche Farbverteilung zurückführen.

GQView bietet auch einen Algorithmus zur Gruppierung von ähnlichen Bilddateien. Alle Dokumente, die verglichen werden sollen, werden paarweise dem Vergleich unterzogen. Ist ein Paar als „ähnlich“ bewertet worden, so wird dafür eine Gruppe initialisiert. Jedes weitere Dokument, dass zu dieser Gruppe auch ähnlich ist, wird der Gruppe hinzugefügt. Wichtig ist dabei, dass jede Gruppe durch eine „Eltern-Kind“ Beziehung dargestellt wird. Das Element, dass zu den meisten Partnern seiner Gruppe die höchste Ähnlichkeit hat, erlangt den Eltern-Status, alle anderen ordnen sich als „Kind“-Elemente unter. Für den Vergleich, ob ein weiteres Dokument in eine Gruppe eingeordnet werden soll, wird nur das Elternelement als Repräsentant der Gruppe verwendet. Das führt dazu, dass mit jedem neuen Element die Gruppen umstrukturiert und gegebenenfalls aufgespalten werden. Dieses Clustering unterscheidet sich methodisch vom Implementierten SMH, was auch für INDetector verwendet wird, da tendenziell mehr Cluster entstehen, obwohl untereinander eventuell eine Ähnlichkeitsbeziehung vorliegt. Dieser Unterschied sei in der Evaluation beachtet. Beide Methoden sind dennoch vergleichbar, da gleiche Bedingungen erfüllt werden: Es gibt keine einelementige Gruppe, und kein Element kommt doppelt vor.

8. Evaluation

Abschließend wird die Bewertung verglichen: Wir prüfen GQView, INDetector und die drei Konzepte für Bildduplikaterkennung mit SMH: SMH$_A$ (Abschnitt 6.1), SMH$_B$ (Abschnitt 6.2) und SMH$_C$ (Abschnitt 6.3) auf Anwendung der Testdatensätze (Kapitel 7) FILTER, WEBCAM und PHOTO und die resultierende Qualität der Cluster.

Für jeden Programmdurchlauf muss vorerst das beste Ergebnis für verschiedene Grenzwerte der Übereinstimmung τ ermittelt werden, da es nicht möglich ist, dafür eine passende Voraussage zu treffen. Wird τ zu hoch gewählt, werden womöglich ähnliche Dokumentenpaare nicht erkannt, wiederum führt ein zu niedrig gewählter Grenzwert zu vielen falsch positiven Ergebnissen. Um das beste Resultat einer Konfiguration zu ermitteln, wird eine Menge von Werten für τ erhoben, die sich zwischen dem minimalen und maximalen Ergebnissen für die Übereinstimmung von allen Dokumentenpaaren ergeben haben. Die qualitative Güte eines Versuches wird durch das höchste Resultat von F-Measure beschrieben F_{max}.

8.1. INDetector

INDetector bietet Parameter zur Anpassung der Anwendung: Kriterien für Interest Point Detection oder die Wahl, ob Bilddaten durch ARG (Attributed Relational Graph) oder BoP (Bag of Parts) repräsentiert werden. Das Ergebnis der Ähnlichkeit zweier Bilddokumente ist in diesem Programm eine Abbildung in die reellen Zahlen simIND$(B_1, B_2) \in \mathbb{R}$ und daher nicht als Metrik zu gebrauchen. Der Schwellwert, ab wann Bilder als ähnlich gelten und dadurch den Clustern zugeordnet werden, muss manuell evaluiert werden. Es gilt lediglich: je höher das Ergebnis der Übereinstimmung, desto ähnlicher sind sich die Dokumente einander. Die Ähnlichkeit zweier identischer Bilder kann nicht bewertet werden: simIND$(X, X) = \infty$.

Zum Training der Anforderung wurden Listen angelegt: P mit 5 positiven Paarbeziehungen, N mit 10 Dokumenten, die keine Ähnlichkeit im jeweiligen Testdatensatz haben. Es ist anzunehmen, dass, mit mehr in den Listen definierten Dokumenten, die Genauigkeit und Richtigkeit zur Bewertung der Ähnlichkeit steigt. In diesem Fall ist jedoch die Auswahl

begrenzt. Pro Datensatz können nicht mehr als 10 Dokumente definiert werden, die einander unähnlich sind.

Die Basis des Vergleiches bei INDetector sind die identifizierten Harris-Interest Points. Demzufolge war vor allem Varianz im Signalrauschen problematisch. Es entstanden teilweise Bildrepräsentationen, die weit mehr (teilweise > 50 MB) Speicherkapazität benötigten, als das ursprüngliche Bilddokument. Demzufolge waren auch die Vergleiche wesentlich zeitaufwändiger. Der Vergleich der ARG-Bildrepräsentation aller Dokumente aus FILTER dauerte 40CPU-Stunden. Die Problematik trat bei PHOTO und WEBCAM nicht auf, da dort die Bildeigenschaften bezüglich des Signalrauschens konstant blieben.

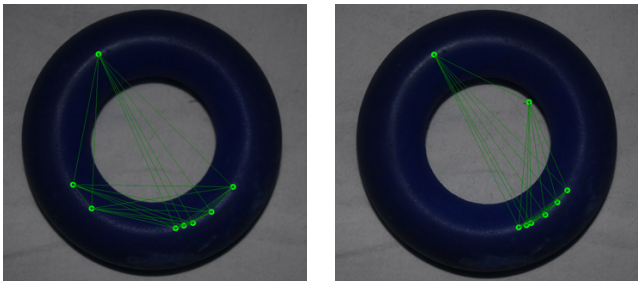
Es stellte sich heraus, dass ARG Repräsentationen auch weniger gute Ergebnisse lie-

(a) BOATS Basis (b) BOATS Gauss (c) BOATS Noise

Abbildung 8.1.: Ergbnisse der Corner-Detection von BOATS.(b) zu viele (c) zu wenig identifizierte Features, wodurch zwischen ihnen keine Ähnlichkeit ermessen wird.

ferten. Nach der Theorie [ZEY] sei INDetector vor allem für Bilder von Kameraaufnahmen entwickelt worden. Unter dieser Voraussetzung sind Bildeigenschaften voraussehbar stabil. Daher ist anzunehmen, dass die graphenbasierte Darstellung (ARG), in Abbildung 8.2 illustriert, vor allem bei komplexeren Szenen, beispielsweise Frames einer Videoaufnahme, räumlichen Bewegungen oder verschiedenen Aufnahmeparametern einer Kamera, ihre Effektivität zeigt. Für diese vergleichsweise trivialen Datensätze erwies sich die BoP-Darstellung als besser geeignet.

Abbildung 8.3 illustriert die Ergebnisse der Versuchsreihen. INDetector erreicht bei PHOTO und WEBCAM fast volle Bewertung ($F_{max} > 0{,}98$), da dort Problemfälle vorliegen, für die INDetector konzipiert wurde. Nur die Ergebnisse von FILTER bestätigen eine Schwäche im Einsatz auf Bilder, die mit grafischen Effekten bearbeitet worden sind ($F_{max} = 0{,}66$). Die Identifikation der Harris-Interest Points ist abhängig von der Signalsärke eines Bildes. Treten beispielsweise starkes Rauschen oder stark unscharfe Stelle im Bild auf, können keine Korrespondenzen zwischen ähnlichen Bildern identifiziert werden. In Abbildung 8.1 sind die extrahierten Features der gemeinten Problematik aufgezeigt. Für nur 76 Dokumente in FILTER wurde eine Paarbeziehung festgestellt. INDetector errechnete die Ähnlichkeiten darin von AIRPLANE und BOATS fast vollständig ($9/10$), jedoch GOLDHILL,



(a) TORUS #1 (b) TORUS #2

Abbildung 8.2.: ARG und Interest Points zu zwei Bilder, die einander als ähnlich bewertet werden. Es fällt auf, dass durch variierenden Fokus verschiedene Interest Points in (a) und (b) identifiziert werden

BABOON und MONALISA nur zur Hälfte richtig (je $5/10$). Alle Gaußgeglätteten und solche mit künstlichem Signalrauschen wurden nicht zugeordnet. Dennoch blieb das Programm auch in diesem Fall konstant präzise und wies wenig falsch-positive Zuordnungen auf ($Precision = 0{,}96$).

8.2. GQView

GQView vergleicht Features der Dokumente, die ausschließlich auf Farbwerten beruhen. Von daher wurde angenommen, dass der Algorithmus auf zwei Probleme treffen wird. Zum einen besteht die Möglichkeit, dass es viele ähnliche Bilder gibt, die verschiedene Features aufweisen. Beispielsweise durch Farbverschiebung in FILTER oder variierende Belichtungen der Szene in WEBCAM. Zum anderen, dass durch Bilder, die aus ähnlichen Farbkombinationen bestehen, aber unterschiedliche Inhalte darstellen, zu viele falsch-positive Cluster gebildet würden. Die Bilder in PHOTO haben sehr ähnliche Farbeigenschafen, da sie auf homogenem Hintergrund fotografiert wurden. In den Abbildungen 8.4 wird jedoch deutlich, dass das Problem nicht an der Präzision der Ergebnisse liegt. Der Clusteringalgorithmus wurde so entworfen, dass dem Nutzer stets eine hohe Richtigkeit der gebildeten Gruppen von ähnlichen Bildern geboten wird, notfalls zu Lasten der Vollständigkeit. Cluster teilen sich sehr wahrscheinlich mit jedem neuen Element was zugeordnet werden soll auf und werden gegebenenfalls neu sortiert. Die Bewertung von $Recall$ fällt darum so gering aus; es gab zu viele falsch-negativ Bewertungen. Auffällig waren Entscheidungen bezüglich der Bilder in FILTER, deren Helligkeit verändert wurde. Diese gruppierte GQView oft, unabhängig ihrer Basisdokumente, in einen eigenen Cluster (Abbildung 8.5).

8.3. Konzepte zur Übertragung der Bildinformation

Bei allen Überführungen steht die Shinglegröße ω als grundlegender Parameter zur Verfügung. Die Größe des Vektors der Merkmale ist konstant bei $k = 100$, wobei jeder Hashwert

512×512 Bildpunkte $\rightarrow 262144$ Token	≈ 40000 Token	≈ 22500 Token	≈ 10000 Token
> 20min	≈ 160sek	≈ 91sek	≈ 40sek

Tabelle 8.1.: Rechenzeit für Shingling von 100 Dokumenten

mit $l = 2^{31} - 1$ bit codiert wird. Da der Wert der Übereinstimmung durch das Verhältnis von c gleichen Shingles zu k Hashwerten errechnet wird, entspricht der in den Diagrammen abgelesene Schwellwert dem prozentual geschätzten Anteil der Übereinstimmung. Ein gleicher Shinglehash zweier Dokumente führt zu einer Übereinstimmung von $\frac{1}{100} = 1\%$ und beispielsweise $c = 50$ gleiche Shinglehashs zu 50%.

In den allen drei Fällen muss auch die Gesamtmenge der gebildeten Wörter beachtet werden, da schnell eine hohe Wortanzahl generiert wird. Ist, beispielsweise, jeder Bildpunkt $B_{x,y}$ wie ein Token behandelt, entstehen Sequenzen mit $X \times Y > 5000000$ Wörtern, was zu einer Rechenzeit von $> 30min$ für 100 Dokumente führt. Daher wurde vor der Informationsextraktion die Bilddokumentgröße verhältnismäßig verkleinert, so dass eine maximale Anzahl nicht überschritten wird. Tabelle 8.1 zeigt die Rechenzeit zum Shingling von 100 Dokumenten bei verschieden durchschnittlicher Wortanzahl und einer Größe des Featurevektors $k = 100$

SMH$_A$ nutzt direkt die Helligkeitswerte der Bildpunkte als Token - die Bildgröße wurde vorher so skaliert, dass jeweils die längere Kante 100 Bildpunkte beträgt und somit kein Bild mehr als $100 \times 100 = 10000$ Token produzieren kann. SMH$_B$ produziert bei $\min(X, Y) = 200$ maximal $\binom{200}{2} = 19900$ Token - für jede Komponente der ersten 200 Diagonalen ($\{\mathsf{D}_0, \mathsf{D}_1, \mathsf{D}_2, ..., \mathsf{D}_{200}\}$) der Kosinustransformation. Beim Konzept SMH$_C$ ist die Anzahl der Token abhängig der gewählten Blockgröße b: für jeden Block ein Token, wobei jedes Bilddokument $\frac{X \times Y}{b \times b}$ Blöcke enthält. (Zum Beispiel: ein Bild der Größe 512×512 wird skaliert auf 200×200. Bei einer Blockgröße $b = 4$ werden 2500 Wörter generiert.)

8.3.1. SMH A

SMH$_A$ wird über die Quantisierungsstufen s des Wertebereiches der Bildpunkte evaluiert. Ohne Quantisierung ist, wie zu erwarten kein brauchbares Ergebnis erzielt worden. Erst die Reduktion auf einen geringeren Wertebereich verschafft ähnlichen Dokumenten auch eine als ähnlich bewertbare Grundmenge an Shingles. Dennoch erreicht SMH$_A$ in keinem Testdatensatz ein $F_{max} > 0{,}5$, was an mangelnder Vollständigkeit liegt. Bei hohem Recall konnte keine Präzision gewährleistet werden. Es lässt sich dennoch feststellen: Je höher die Quantisierung, desto wahrscheinlicher werden ähnliche Dokumente auch den richtigen Clustern zugewiesen. Wiederum führt das auch dazu, dass mehr falsch-positive Paare in ein Cluster gruppiert werden. Außerdem ist eine Korrelation zwischen Quantisierung und

Shinglegröße festzustellen: je größer die Quantisierung, desto besser sind Ergebnisse bei größeren Shingles.

Im Ergebnis ist auffällig, dass bei PHOTO und WEBCAM die Quantisierung auf $s = 17$ Stufen und Shinglegröße $\omega = 2$ die besten Ergebnisse erzielte. Nur die Anwendung auf den Testdatensatz FILTER zeigte darin keine Übereinstimmung, sondern lieferte erst bei einer Quantisierung auf $s = 3$ Farbstufen vergleichbare Ergebnisse. Wie sich die besten Resultate von SmH_A im Verlauf mit verschieden gewählten Schwellwerten τ verhalten haben, ist in Abbildung 8.8 illustriert.

Auch hier werden Informationen des Bildinhaltes aus Farbwerten gewonnen. Darum tritt das Problem auf, dass, wie schon bei GQView, gerade die Bilder in FILTER, die mit einem Helligkeitsoperator kombiniert wurden, in einen Cluster separat gruppiert werden. Diese Dokumente sind in Abbildung 8.5 dargestellt. Im Testdatensatz WEBCAM konnten die Klassen SUISSE und ZITTAU vollständig und präzise zugeordnet werden. Probleme bereiteten die Klassen FRANCE, NORGE und NOVOSIBIRSK. Dort waren die Helligkeitswerte der Bildpunkte zu verschieden, um vom Algorithmus als ähnlich erkannt zu werden. Die Anwendung des Konzeptes auf den Testdatensatz PHOTO ergab bei $F_{max} = 0{,}44$, dass die Bilder der Klassen TORUS und CABLE vollständig richtig geclustert wurden. Jedoch konnten mit dieser Konfiguration die Klassen TOOL, SPOON, SPADES, CD, BOOK und POT gar nicht zugeordnet werden. Somit clusterte der Algorithmus darin nur 36 Elemente, was zu einem Recall von $0{,}28$ führte, obgleich die Präzision komplett erfüllt wurde.

Neben den Bewertungen wird auch festgestellt, dass qualitativ gute Entscheidungen stets bei einem Grenzwert von $0{,}3 < \tau < 0{,}4$ getroffen werden.

8.3.2. SMH B

Für SMH_B spielt der Grenzwert g, der zur Wortbildung verwendet wird, die entscheidende Rolle. g bestimmt, welcher Token aus einem Koeffizienten zum Shingling erzeugt wird. Die Koeffizienten einer DCT sind abhängig von der Bildgröße und dem Wertebereiches des Bildsignals. Da hier stets davon ausgegangen wird, herkömmliche Bilder zu untersuchen, die schwarze Bildpunkte als 0 und weiße als 255 abbilden, kann die Abhängigkeit der Koeffizienten auf die Bildgröße beschränkt werden; diese ist verhältnismäßig konstant bei $\min(X, Y) = 200$. Die DC-Komponente kann den höchsten Wert annehmen. Je weiter Komponenten davon entfernt sind, desto kleiner ist üblicherweise der Koeffizient. In einem natürlichen Bildsignal sind nur wenige hochfrequente Anteile zu erwarten, was den Informationsgehalt derer erhöht. Daraus folgt: Je höher der Grenzwert gewählt wird, desto tieffrequenter sind die Komponenten, die zur Ähnlichkeitsschätzung verwendet werden, da alle anderen Koeffizienten gleiche Token produzieren und somit weniger Einfluß auf die

Messung nehmen.

Bei Anwendung auf den Testdatensatz WEBCAM erzielte SMH_B eine Bewertung von $F_{max} = 0{,}87$ bei $\omega = 4$ und $g = 150$. Mit dieser Konfiguration wurden 95 von 100 Bilder geclustert. Alle Bilder der öffentlichen Webcams der Klassen ZITTAU, ENGLAND, FINLAND, NOVOSIBIRSK und ITALY wurden den richtigen Clustern zugewiesen. Die Cluster von BERLIN und CANADA erhielten 9 von 10 richtige Zuweisungen. Die Aufnahmen von FRANCE gruppierte SMH_B fälschlicherweise in 3 verschiedene Cluster, was an der Unstetigkeit in der Struktur der Aufnahmen lag. Abbildung 8.6 zeigt die Bilder, die nicht richtig zugeordnet werden konnten.

Desweiteren erreichte SMH_B bei der Versuchsreihe von FILTER eine Bewertung von $F_{max} = 0{,}84$, wobei 82 der 100 Elemente erfasst wurden. Der Cluster mit den Elementen der Klasse LENA wurde vollständig richtig gebildet, PEPPER, MONALISA, BOATS und AIRPLANE mit jeweils 9 von 10 richtigen Elementen. Wenn ein Bilddokument nicht eingeordnet werden konnte, dann erneut stets jenes, mit der Erhöhung der Bildhelligkeit. Dieses Ergebnis ist zu erwarten: diskrete Bildsignale haben einen endlichen Wertebereich und eine Erhöhung der Werte führt zu weniger Signalschwankung, was sich auf die Koeffizienten der Spektralkomponenten auswirkt.

Problematisch war die Anwendung dieses Konzepts auf den Testdatensatz PHOTO. Der Grund dafür liegt in Variationen der Bilddokumente, die, auch wenn nicht erkennbar, zu zu sehr verschiedenen Kosinustransformationen führten, so dass eine Ähnlichkeit vom Algorithmus nicht festgestellt werden konnte. Im Durchlauf mit dem Ergebnis $F_{max} = 0{,}47$ ($\omega = 4, g = 120$) wurden die Bilder der Klassen SPADES, CABLE und SPOON in den gleichen Cluster gruppiert, ebenso BALL und TORUS. Es könnte vermutet werden, dass die Entscheidung gefällt wurde, da diese Elemente auch ähnliche Strukturen aufweisen. Der Cluster mit Elementen aus BOOK enthielt 9 sowie TOOL 8 von 10 richtigen Elementen. 78 Dokumente wurden insgesamt geclustert, bei einem Precision-Recall Verhältnis von $0{,}43$ zu $0{,}51$.

Die Tabellen 8.2, 8.3 und 8.4 zeigen die jeweils besten Ergebnisse F_{max} im Verhältnis des gewählten Grenzwertes g zur Shinglegröße ω. Der Verlauf des Verhältnisses von Recall zu Precision in Abhängigkeit des Grenzwertes der Übereinstimmung τ ist in den Diagrammen von Abbildung 8.9 abzulesen. Unabhängig der Datensätze scheint die Wahl von $\tau \approx 0{,}7$ ein in dieser Konfiguration repräsentativer Wert der Übereinstimmung zu sein.

g \ ω	1	2	3	4	5	7	8	9
20	0,17	0,23	0,24	0,17	0,17	0,19	0,21	0,17
50	0,18	0,34	0,51	0,46	0,27	0,31	0,26	0,34
70	0,26	0,63	0,49	0,48	0,52	0,34	0,35	0,44
100	0,24	0,61	0,71	0,51	0,62	0,47	0,58	0,56
120	0,18	0,61	0,65	0,61	0,53	0,57	0,27	0,43
150	0,33	0,66	0,68	**0,87**	0,72	0,41	0,44	0,4
170	0,4	0,54	0,61	0,63	0,43	0,48	0,38	0,36
200	0,36	0,58	0,65	0,63	0,47	0,28	0,38	0,32

Tabelle 8.2.: SMH$_B$ bei Anwendung auf WEBCAM. Dargestellt sind die jeweils besten Ergebnisse von F-Measure unter Veränderung von Shinglegröße ω und Grenzwert zur Wortbildung g

g \ ω	1	2	3	4	5	7	8	9
20	0,23	0,51	0,57	0,48	0,41	0,45	0,54	0,63
50	0,19	0,51	0,49	0,57	0,6	0,63	0,67	0,61
70	0,21	0,6	0,56	0,63	0,64	0,72	0,69	0,63
100	0,25	0,61	0,69	0,72	0,68	0,68	0,63	0,7
120	0,3	0,62	0,67	0,78	0,75	0,72	0,65	0,6
150	0,28	0,68	0,67	0,75	0,78	0,72	0,7	0,71
170	0,34	0,67	**0,84**	0,8	0,82	0,74	0,73	0,68
200	0,44	0,73	0,77	0,81	0,78	0,71	0,72	0,7

Tabelle 8.3.: SMH$_B$ bei Anwendung auf FILTER. $F_{max}(\omega, g)$

g \ ω	1	2	3	4	5	7	8	9
20	0,25	0,23	0,4	0,35	0,22	0,37	0,24	0,23
50	0,24	0,34	0,27	0,3	0,3	0,33	0,33	0,28
70	0,33	0,26	0,39	0,38	0,26	0,33	0,3	0,25
100	0,29	0,24	0,36	0,4	0,46	0,26	0,24	0,23
120	0,22	0,43	0,35	0,41	0,35	0,29	0,27	0,28
150	0,22	0,3	0,42	0,39	0,41	0,35	0,23	0,24
170	0,2	0,31	0,32	0,38	0,5	0,4	0,29	0,27
200	0,17	0,25	0,31	**0,47**	0,29	0,3	0,31	0,35

Tabelle 8.4.: SMH$_B$ bei Anwendung auf PHOTO. $F_{max}(\omega, g)$

8.3.3. SMH C

Die Ergebnisse des Konzeptes SMH_C sind abhängig von der Blockgröße b und analog zu SMH_B vom Grenzwert g, der zur Bestimmung der Buchstaben von Token und somit zur Auswahl der Komponenten eines kosinustransformierten Blockes verwendet wird. Alle Blöcke B' werden, wie in SMH_A, in zweidimensionale Shingles zugeordnet, mit dem Ziel so eine Toleranz zu Verschiebungen von abgebildeten Objekten herstellen zu können. Da Bildblöcke eine quadratische Größe haben, entstehen mit größerem b auch quadratisch weniger Token, was den Rechenaufwand stark entlastet. Dennoch stellt sich heraus, dass kleine Blöcke bessere Ergebnisse liefern. Daraus ist zu schließen, dass auch die Grenzwerte g klein gewählt sein müssen, da die Koeffizienten einer DCT abhängig von der Signallänge sind. g wurde darum im Verhältnis zur Blockgröße b gewählt: $g_b = \frac{b}{g}$.

Da der Testdatensatz PHOTO mit SMH_A und SMH_B keine guten Ergebnisse erzielte ($F_{max} < 0{,}5$), wird mit SMH_C vorerst genauer beobachtet, wie sich die Qualität der Cluster in Abhängigkeit von g entwickelte. Die Bilddokumente in PHOTO stellen dem Programm in zweierlei Hinsicht eine Anforderung: zum einen sind die Hintergründe bei allen Bildern gleich, was möglicherweise zu einer falsch geschätzten Ähnlichkeit führt, zum anderen variieren sie in ihrer Position, Fokus und Belichtung. Es wurden zwar bessere Ergebnisse erzielt, jedoch nicht so deutlich, dass das Resultat von INDetector erreicht werden konnte. Genauer: Fälschlicherweise wurde für die Elemente BALL, CD und TORUS ein Cluster gebildet, in dem jedes Dokument dieser Klassen vorkommt. Diese Entscheidung ist zwar nicht gewollt, dennoch ist die strukturelle Verwandtschaft dieser Klassen nicht zu übersehen: alle fotografierten Objekte sind rund und auch im gesamten Testdatensatz, die einzig runden Objekte. In Abbildung 8.7 sind Bilder dieser Klassen dargestellt. Wäre diese Entscheidung richtig, würde die Bewertung $F_{max} = 0{,}88$ statt $F_{max} = 0{,}64$ ausfallen. Die Klassen TOOL, SPOON, SPADES und CABLE wurden vollständig richtig geclustert. Probleme bereitete UMBRELLA (2 von 10), vermutlich weil das Objekt, in einer anderen Größe vorliegt und somit gewisserweise aus dem Rahmen der Testdokumente fällt. Dieses Ergebnis wurde durch die Konfiguration $\omega = 10$, $b = 3$ und $g_3 = 120$ ermittelt. Da alle Bilder in der Vorverarbeitung mit der Größe 200×298 untersucht worden, entstanden für jedes Bild 6600 Wörter. Allerdings wurde diese Schätzung bei einem Grenzwert von $\tau = 0{,}08$ gefällt, was bedeutet, dass nur 8 von 100 gleichen Hashwerten für diese Entscheidung verantwortlich waren. (Weitere Versuche haben ergeben, dass diese Zahl in der Tat repräsentativ für die Übereinstimmung der gebildeten Wörter ist.)

Die Klassen aus dem Datensatz FILTER zeigten sich abermals problematisch mit den helleren Bildern. Das hat die selben Hintergründe, wie in Abschnitt 8.3.2 bereits erklärt. Dennoch erhielt die Mehrheit der Klassen annähernd gleiche Vollständigkeit (≈ 8 von 10) und somit bei 99 geclusterten Dokumenten eine Bewertung von $F_{max} = 0{,}64$. Sehr auffällig

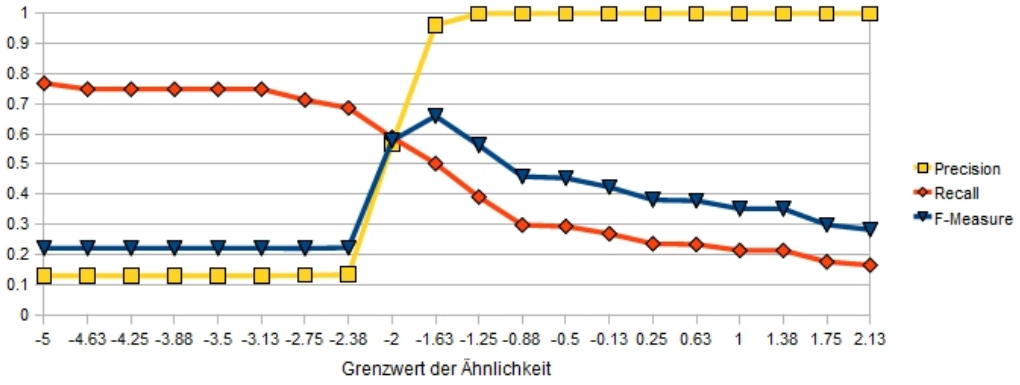
war, dass bei fast jeder Konfiguration für das Paar GOLDHILL BASIS und GOLDHILL GAUSS eine Übereinstimmung von $> 90\%$ geschätzt wurde. Daran wird erneut deutlich, dass die Wahl der Parameter den größten Teil zur Bewertung des Algorithmus ausmacht. Eventuell könnte eine differenzierte Konfiguration auch qualitativ bessere Ergebnisse erzielen. In diesem Fall wurden die Resultate durch $\omega = 3$, $b = 5$ und $g_5 = 200$ ermittelt.

Die Evaluation des Konzeptes auf Anwendung des Testdatensatzes WEBCAM erreichte $F_{max} = 0{,}7$ bei $\omega = 3$, $b = 2$ und $g_2 = 80$. Die Dokumente der Klasse NOVOSIBIRSK bildeten einen Cluster, jedoch mit weiteren Elementen aus BERLIN, sowie CANADA, welches fälschlicherweise Elemente aus NORGE enthielt. ZITTAU , FINLAND und ITALY wurden vollständig und richtig geclustert. Falsch, in mehrere Cluster sortiert, waren erneut die Bilder Klasse FRANCE. Dieser Problemfall wurde in Abschnitt 8.3.2 erwähnt und mit Abbildung 8.6 illustriert.

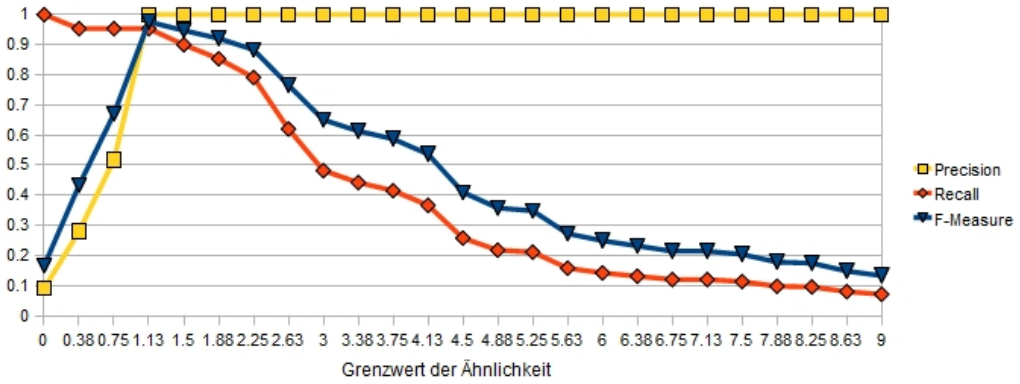
Abschließend kann für alle Datensätze, die mit diesem Konzept überführt worden sind, festgestellt werden, dass sich die besten Resultate in ungefähr gleicher Bewertung befinden: $F_{max} \approx 0{,}6$. Somit ist eine Methode gefunden worden, die, auch wenn qualitativ nicht so gut wie SMH$_B$, so dennoch unabhängig der überprüften Datensätze, gleichberechtigt anwendbar ist. In vielen Fällen war die Vollständigkeit hoch, jedoch zu Lasten der bewerteten Präzision. Werden 20 Dokumente in einen Cluster sortiert, wovon jeweils 10 richtige Paare sind, verringert das die Bewertung auch um 10 falsch-positive Paare, obwohl eine Trennung derer algorithmisch lösbar zu sein scheint.

8.4. Ergebnis der Evaluierung

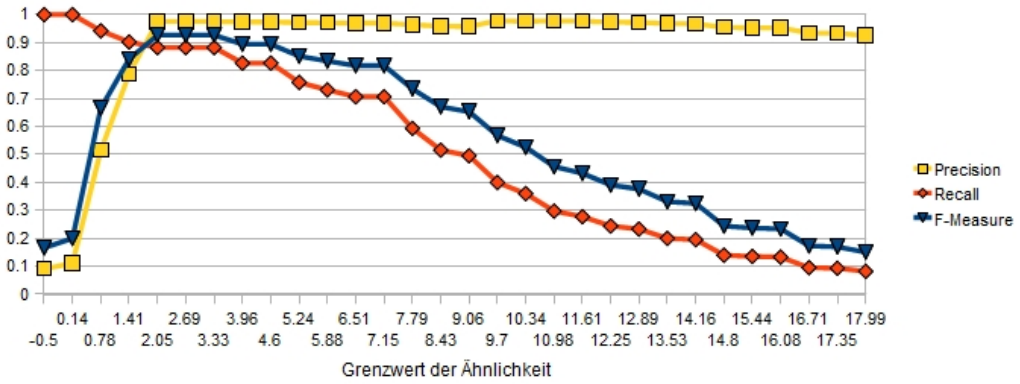
Die Konzepte konnten mit den gestellten Anforderungen in vielerlei Hinsicht richtig umgehen. Vor allem SMH$_B$ erwies sich als effektiv in Anwendung auf die Datensätze FILTER und WEBCAM. Der problematische Datensatz PHOTO konnte von SMH$_C$ besser bearbeiten werden und lieferte Ergebnisse, die zwar nicht gewollt, aber dennoch die Methodik befürworten. Obwohl alle Werte der Übereinstimmung geschätzt wurden, zeigte sich bei mehrmaligem Durchlauf, dass diese Schätzung durchaus repräsentativ, für den tatsächlichen Wert der Übereinstimmung ist. Es konnten für die Testdatensätze passende Konfigurationen gefunden werden, mit denen die Konzepte den größten Teil der Ähnlichkeitsanfragen richtig beantworteten. In jedem Fall ist mit dem Einsatz des minHash-Prinzips eine deutliche Leistungssteigerung bezüglich der Rechenzeit einhergegangen, besonders im Vergleich zum komplexen Verfahren INDetector. GQView benötigte deutlich weniger Rechenzeit (< 5 Sekunden zum Clustern von 100 Bilddokumenten). Dennoch galt dort die wahrscheinlich nutzerorientierte Forderung, so präzise wie möglich zu sein, was zu vielen falsch-negativ Entscheidungen führte und demnach von der Evaluation schlecht bewertet wurde.



(a) FILTER

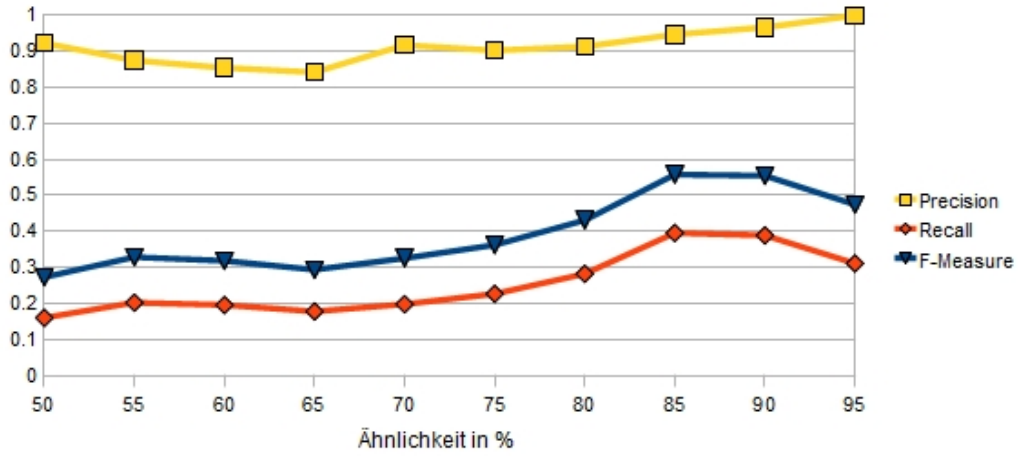


(b) WEBCAM

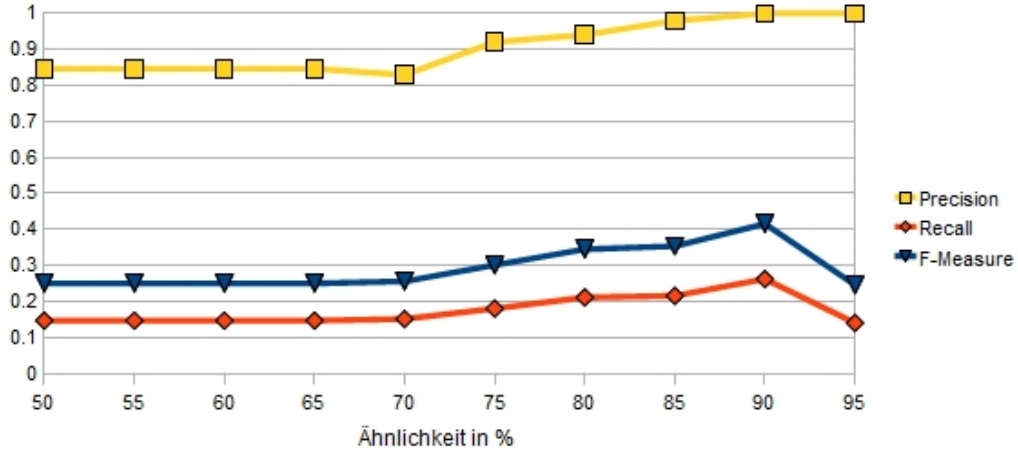


(c) PHOTO

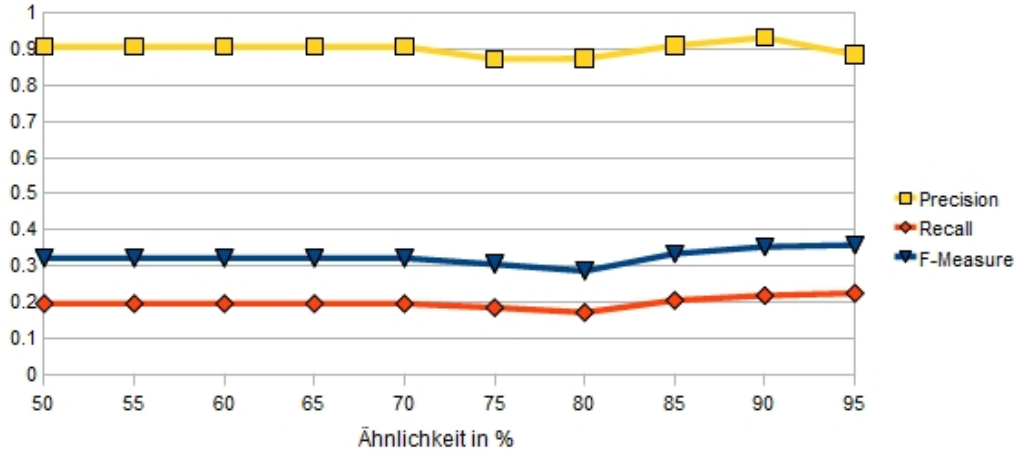
Abbildung 8.3.: Bewertung zu Precision, Recall und F-Measure von INDetector auf Anwendung der Testdatensätze. Man erkennt eine starke Steigung der Präzision, bei steigendem Grenzwert, wohingegen die Vollständigket nur langsam abnimmt.



(a) FILTER

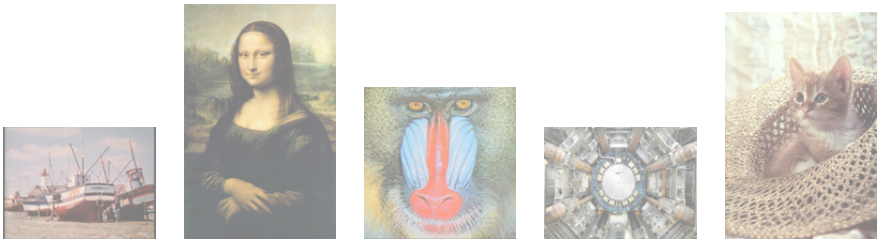


(b) WEBCAM



(c) PHOTO

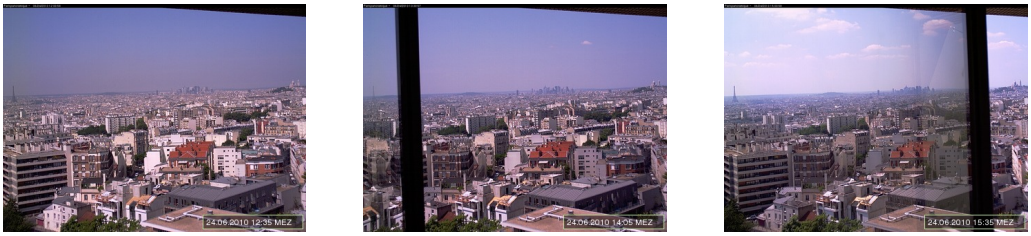
Abbildung 8.4.: Bewertung zu Precision, Recall und F-Measure von GQView auf Anwendung der Testdatensätze. Die Präzision bleibt stabil, zu Lasten der Vollständigkeit.



(a) BOATS (b) MONALISA (c) BABOON (d) SPEED (e) CAT

(f) BARBARA (g) AIRPLANE (h) PEPPER (i) GOLDHILL (j) LENA

Abbildung 8.5.: Bilder aus FILTER. Durch die Helligkeitserhöhung konnten diese Elemente nicht ihren Basisdokumenten zugeordnet werden. Diese Dokumente bildeten in vielen Fällen eigene Cluster. Nur INDetector konnte mit dieser Problematik richtig umgehen.

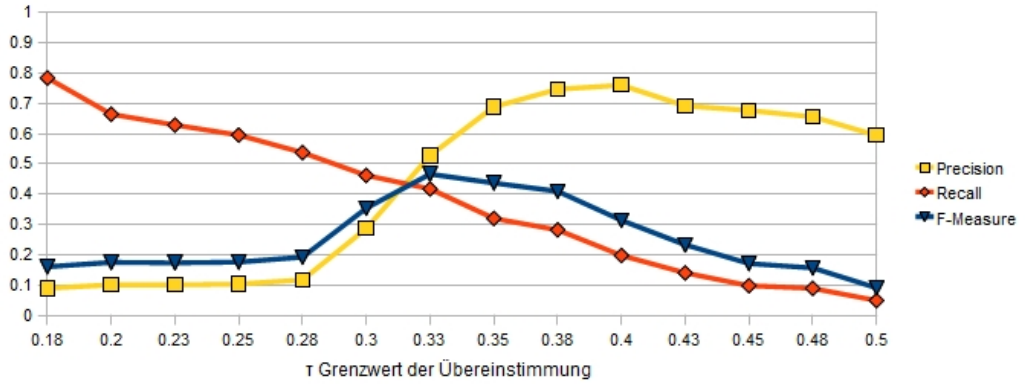


(a) FRANCE #1 (b) FRANCE #2 (c) FRANCE #3

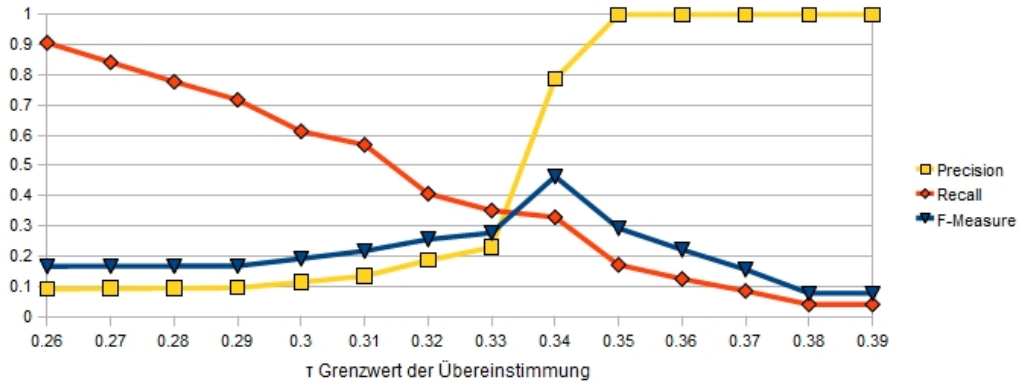
Abbildung 8.6.: Bilder von FRANCE, die mit SMH$_B$ nicht zugeordnet werden konnten, da im Aufnahmemoment sich ein Großteil der Bildstruktur verändert.

(a) TORUS (b) CD (c) BALL

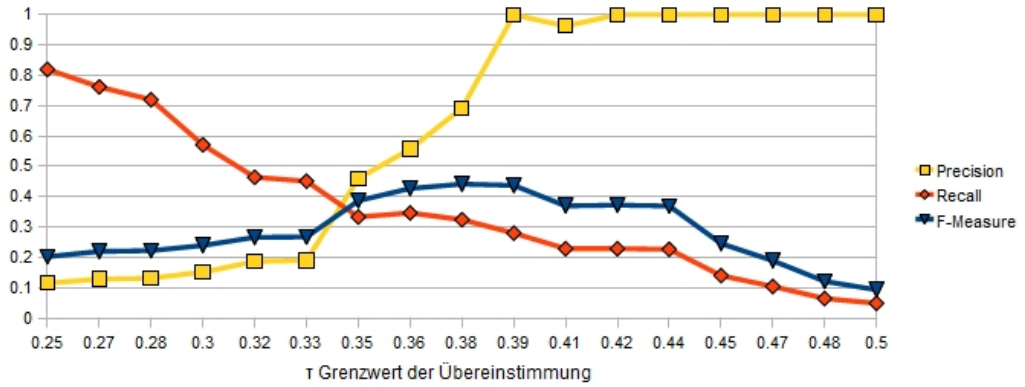
Abbildung 8.7.: Bilder von PHOTO, die mit SMH$_C$ fälschlich einen eigenen Cluster bildeten. Dennoch ist die strukturelle Verwandtschaft der abgebildeten Objekte erkennbar.



(a) FILTER $\omega = 3$ $s = 3$

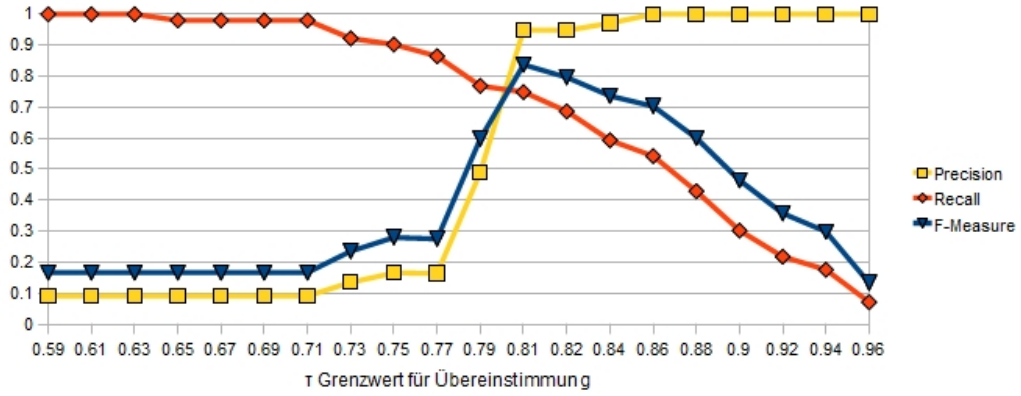


(b) WEBCAM $\omega = 2$ $s = 17$

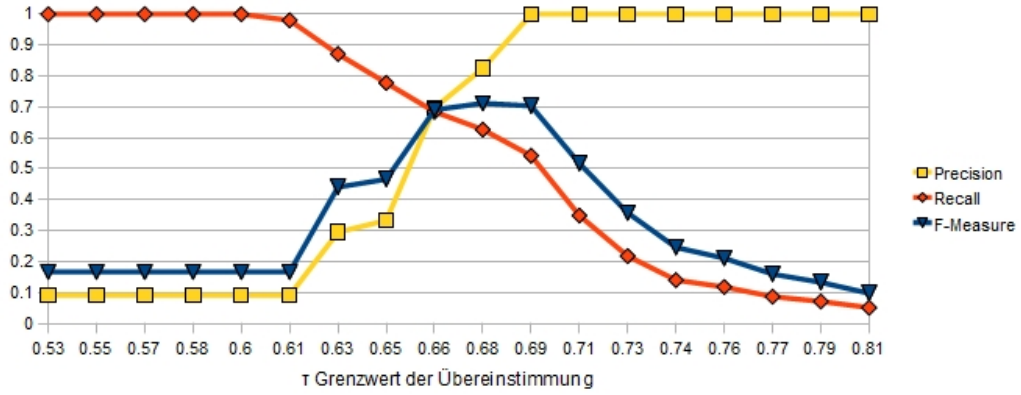


(c) PHOTO $\omega = 2$ $s = 17$

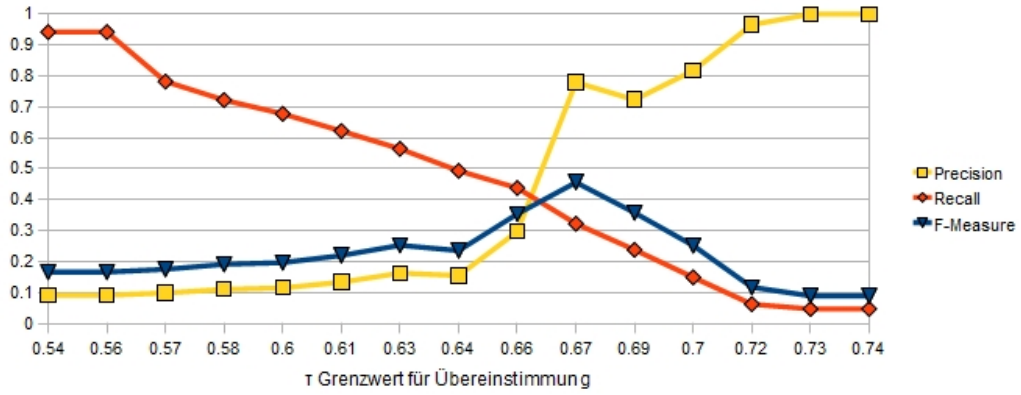
Abbildung 8.8.: Bewertung zu Precision, Recall und F-Measure von SMH_A auf Anwendung der Testdatensätze. Zur Bewertung von Entscheidungen, ob Paare als Paar gelten wurde der Grenzwert τ beobachtet.



(a) FILTER $\omega = 3$ $g = 170$

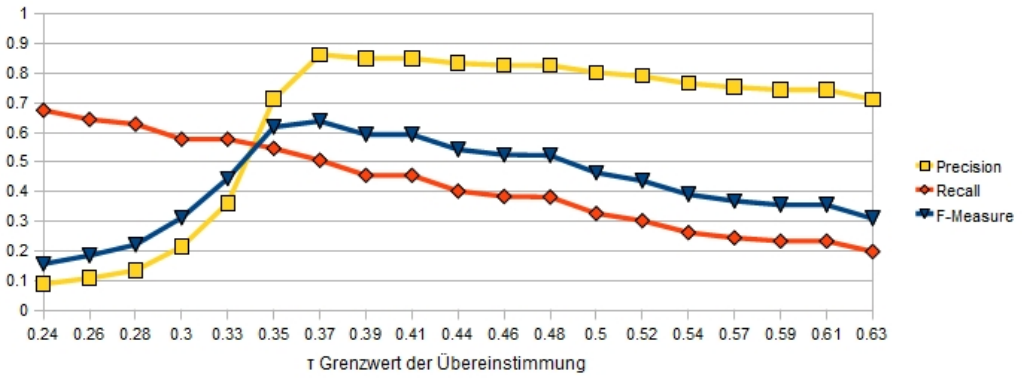


(b) WEBCAM $\omega = 4$ $g = 150$

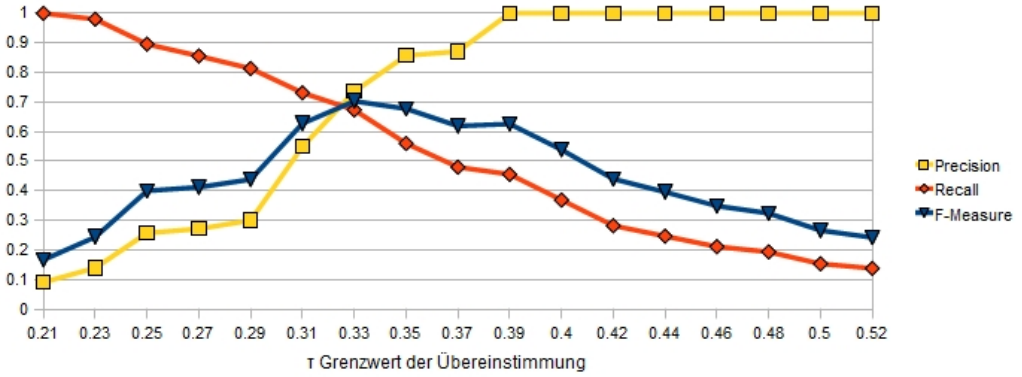


(c) PHOTO $\omega = 4$ $g = 200$

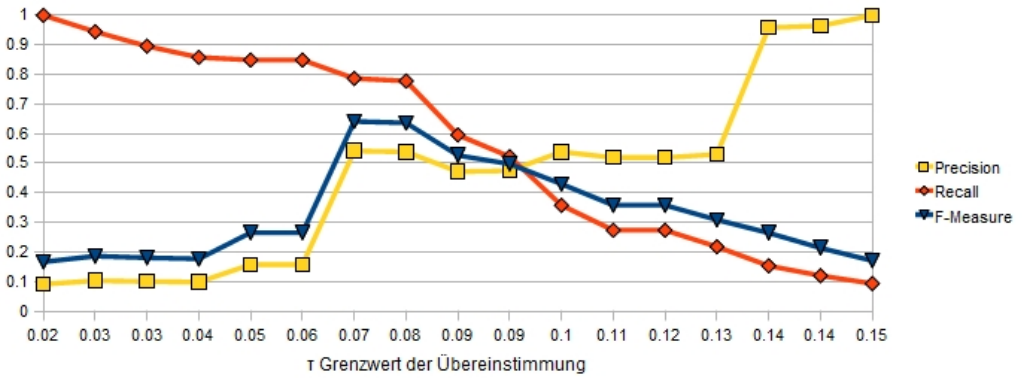
Abbildung 8.9.: Bewertung zu Precision, Recall und F-Measure von SMH$_B$ auf Anwendung der Testdatensätze in Abhängigkeit zu τ. In (a) ist eine Qualität am Verlauf des Graphen abzulesen: Der Anstieg der Präzision (gelb) ist deutlich stärker als das Sinken der Vollständigkeit (rot). In (b) und (c) ist dieses Verhältnis schwächer.



(a) FILTER $\omega = 3$ $g_5 = 200$



(b) WEBCAM $\omega = 3$ $g_2 = 80$



(c) PHOTO $\omega = 10$ $g_3 = 120$

Abbildung 8.10.: Bewertung zu Precision, Recall und F-Measure von SMH$_C$ auf Anwendung der Testdatensätze in Abhängigkeit zu τ. Im Graphen (c) musste dieser Grenzwert τ niedrig gewählt werden (<0,15). Für nur zwei Paare wurde mit SMH$_C$ eine Übereinstimmung $> 0{,}2$ geschätzt.

9. Zusammenfassung

Die Behaupung von BRODER, *„[...] there are no limits on the objects that can be compared for resemblance: images, video sequences, or databases“*[BGMZ97], trifft durchaus zu, wie sich auch schon in verwandten Arbeiten gezeigt hat. Erkenntnisreich ist darüber hinaus auch die Einschätzung, dass selbst durch Shingling identifizierte Features von Bilddokumenten einen Beitrag zu *Near Image Duplicate Detecion* leisten können. Es wurde evaluiert, dass Problemfälle existieren, bei denen auf komplexe Feature Detection in Bilddokumenten verzichten werden kann, wodurch mit deutlich weniger Aufwand brauchbare Ergebnisse erzielt werden können. Vorerst beschränkt sich dieser Vorteil auf Bilddatensätze, deren zu untersuchende Ähnlichkeit auf wenige Eigenschaften bezogen sind, vor allem solche, die in ihrer Bildstruktur gleich bleiben. Es werden Veränderungen von Farbwerten, Kontraständerungen sowie nicht erkennbare strukturelle Störungen, wie etwa Signalrauschen, Komprimierungsartefakte oder partielle Veränderungen, in kleinen Bildbereichen toleriert. Diese Toleranz spiegelt sich in einer Ähnlichkeitsanfrage wider, da solche Daten größtenteils zu ihren korrespondierenden Partnerdokumenten zurückgeführt werden konnten. Der Vorteil, der von BRODER beschrieben wurde, bleibt bestehen: der Einsatz auf große Dokumentmengen ist unproblematisch, da alle Dokumente in einem Zeitaufwand linear zur Dokumentmenge verglichen werden.

Leider erbringt das Konzept von SMH_C nicht die gewünschte Leistung, durch Kombination der Methoden SMH_A und SMH_B noch bessere Ergebnisse zu erzielen. Die Schwierigkeit lag in der Wahl der Parameter, für die kein ausgeglichenes Maß ermittelt werden konnte. Es bedarf einer detaillierteren Untersuchung, wie sich einerseits Koeffizienten der diskreten Kosinustransformation von natürlichen Bildern generell verhalten und inwiefern andererseits eine Aufteilung von Blöcken dieses Verhalten beeinflusst. Im Sinne des Shinglings könnten sich Bildblöcke auch überschneiden und somit Bildinformationen repräsentativer abgebildet werden.

Dennoch erwies sich das Konzept SMH_B als qualitativ brauchbar. Es konnte teilweise sogar bessere Ergebnisse als Vergleichsprogramme erzielen. Ein Ansatz zur Erweiterung dieser Idee wäre es, anders als im Konzept SMH_C, den gewählten Grenzwert stochastisch so zu schätzen, dass auch andere Problemfälle damit gelöst werden können. Die Konfiguration dieses Wertes war ausschlaggebend für die Qualität der Methode. Idealerweise würden sich dann Verfahren finden, die darüberhinaus auch die Entscheidung der Tokengenerierung

effektiver gestalten.

Eine mögliche Verbesserung des Ergebnisses aller SMH-Methoden könnte ein optimierter Clusteringalgorithmus sein. Der in der Implementierung verwendete und auch von BRODER so beschriebene Clusteringalgorithmus [Bro00], erstellt einen Graphen mit Kanten zwischen allen Dokumenten, die einen Grenzwert der Übereinstimmung überschreiten. Würde auch die Menge der Verbindungen bewertet werden, könnten Cluster, die fälschlicherweise durch einen Zusammenhang von nur einem falsch-positiven Paar bestimmt wurden, richtig getrennt werden. Dieses Szenario ist in Abbildung 9.1 illustriert.

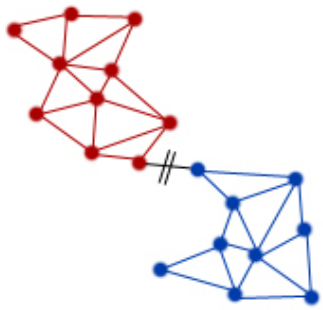


Abbildung 9.1.: Schema eines falsch gebildeten Clusters aus Elementen zweier Klassen (rot und blau). Die Kanten stellen Verbindungen dar, die ein Ähnlichkeitsverhältnis zwischen den Dokumenten (Punkte) visualisieren.

Es bleiben noch offene Fragen, mit denen eine Weiterarbeit motiviert werden könnte. Die untersuchten Konzepte zur Überführung der Bildinformationen nutzten bisher nur Features, die aus Helligkeitswerten der Bildpunkte und Fouriertransformation extrahiert werden konnten. Möglicherweise können auch komplexere Merkmale eines Bildes (SIFT/-SURF bietet eine auf viele Problemfälle der Bildverarbeitung ausgeglichene Extraktion und Beschreibung relevanter Bildfeatures. Siehe dazu Kapitel 2) in eine Reihenfolge gebracht werden, die das Shingling Prinzip konzeptionell unterstützt. Damit könnten bestehende Forschungsprojekte aufgegriffen und so umstrukturiert werden, dass die Menge gewonnener Features durch Shingling ausgewertet werden kann.

Auch eine Kombination mit Textverarbeitung wäre interessant. Da beispielsweise die Überführung SMH_B eine eindimensional verwertbare Sequenz von Token liefert, könnten prinzipiell auch andere Methoden zum Vergleich von Textdokumenten genutzt werden. Es ist denkbar, dass Anwendungen somit multimediale Quellen, Text- und Bildinhalte, inhaltlich kombiniert verglichen werden können, statt einer separaten Analyse jedes Mediums.

A. Abbildungsverzeichnis

B. Tabellenverzeichnis

C. Literaturverzeichnis

[BAA] BRODER, Andrei Z. ; AVENUE, Lytton ; ALTO, Palo: On the resemblance and containment of documents. In: *Systems Research* , S. 1–9

[BB05] BURGER, Wilhelm ; BURGE, Mark J.: *Digitale Bildverarbeitung : Eine Einführung mit Java und ImageJ*. 2., überarbeitete Auflage. Berlin, Heidelberg : Springer-Verlag, 2005

[BCFM00] BRODER, Andrei Z. ; CHARIKAR, Moses ; FRIEZE, Alan M. ; MITZENMACHER, Michael: Min-Wise Independent Permutations. In: *Journal of Computer and System Sciences* 659 (2000), S. 630–659

[BGMZ97] BRODER, Andrei Z. ; GLASSMAN, Steven C. ; MANASSE, Mark S. ; ZWEIG, Geoffrey: Syntactic clustering of the Web. In: *Computer Networks and ISDN Systems* 29 (1997), September, Nr. 8-13, S. 1157–1166. – ISSN 0169–7552

[Bro00] BRODER, Andrei Z.: Identifying and Filtering Near-Duplicate Documents. In: *World Wide Web Internet And Web Information Systems* (2000), S. 1–10

[BS07] BAYARDO, Roberto J. ; SRIKANT, Ramakrishnan: Scaling Up All Pairs Similarity Search. In: *ReCALL* (2007)

[CFGD02] CHOWDHURY, Abdur ; FRIEDER, Ophir ; GROSSMAN, David ; DATA, Introduction: Collection Statistics for Fast Duplicate Document Detection. In: *Work* 20 (2002), S. 171–191

[Cha02] CHARIKAR, Moses S.: Similarity Estimation Techniques from Rounding Algorithms. In: *Science* (2002)

[Der04] DERPANIS, Konstantinos G.: The Harris Corner Detector. In: *International Journal* (2004), S. 2–3

[DI04] DATAR, M. ; INDYK, P.: Locality-sensitive hashing scheme based on p-stable distributions. In: *In SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, ACM Press, 2004, S. 253–262

[Gm04] GARCIA-MOLINA, Hector: Web Spam Taxonomy. In: *Science* (2004), S. 1–11

[KSH] KE, Yan ; SUKTHANKAR, Rahul ; HUSTON, Larry: Ef cient Near-duplicate Detection and Sub-image Retrieval. In: *Image (Rochester, N.Y.)* , S. 869–876

[Low99] LOWE, David G.: Object Recognition from Local Scale-Invariant Features. In: *Proc. of the International Conference on Computer Vision, Corfu*, 1999

[Man94] MANBER, Udi: Finding Similar Files in a Large File System. In: *USENIX WINTER 1994 TECHNICAL CONFERENCE*, 1994, S. 1–10

[MRS08] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHTZE, Hinrich: *Introduction to Information Retrieval*. New York, NY, USA : Cambridge University Press, 2008. – ISBN 0521865719, 9780521865715

[PIZ+07] PHILBIN, James ; ISARD, Michael ; ZISSERMAN, Andrew ; SCIENCE, Engineering ; RESEARCH, Microsoft ; VALLEY, Silicon: Scalable Near Identical Image and Shot Detection. In: *Analysis* (2007)

[Por06] PORTER, M.F.: An algorithm for suffix stripping. In: *Program: Electronic Library and Information Systems* 40 (2006), Nr. 3, S. 211–218

[PZ07] PHILBIN, James ; ZISSERMAN, Andrew: Near Duplicate Image Detection: min-Hash and tf-idf Weighting. (2007)

[Ste98] STEIN, Benno: Fuzzy-Fingerprints for Text-Based Information Retrieval. In: *Knowledge Management* (1998), S. 572–579

[WEE+91] WALLACE, Gregory K. ; ENGINEERING, Multimedia ; EQUIPMENT, Digital ; MAYNARD, Corporation ; SUBMITTED, Massachusetts ; TRANSACTIONS, Ieee ; THIS, Consumer E. ; CCITT, The ; GROUP, The ; PHOTOGRAPHIC, Joint ; GROUP, Experts ; JPEG, Ccitt S. ; STANDARD, I S O. ; RECOMMENDATION, Ccitt: The JPEG Still Picture Compression Standard. In: *Entropy* (1991), S. 1–17

[WJL+] WANG, Zhe ; JOSEPHSON, William ; LV, Qin ; CHARIKAR, Moses ; STREET, Olden: Filtering Image Spam with Near-Duplicate Detection. In: *Image (Rochester, N.Y.)*

[ZEY] ZHANG, Dong-qing ; ENGINEERING, Electrical ; YORK, New: Detecting Image Near-Duplicate by Stochastic Attributed Relational Graph Matching with Learning. In: *Image (Rochester, N.Y.)* , S. 877–884

Eigenständigkeitserklärung

Hiermit erkläre ich, Edgar Scherstjanoi, die vorliegende Belegarbeit zum Thema

Vergleich und Übersicht von Algorithmen zur Dokumentenduplikaterkennung bei Bildern

eigenständig und ausschließlich unter Verwendung der im Quellenverzeichnis aufgeführten Literatur- und sonstigen Informationsquellen verfasst zu haben.

Dresden, 15. Oktober 2010