UNIVERSITÄT
LEIPZIG

in Kooperation mit

Competence Center for Scalable Data Services and Solutions

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

Thema:	Skalierbare bildbasierte Deduplikation
Vorgelegt von:	Christopher Rost
Geboren am: in:	06.09.1990 Querfurt
eingereicht am:	21. Dezember 2017
Universitärer Betreuer:	Dr. Eric Peukert ScaDS Dresden/Leipzig
Externer Betreuer:	Dipl.-Medieninform. Christoph Müller

Kurzfassung

Digitale Bilder, die ein und dasselbe Realweltobjekt abbilden, werden als Duplikate bezeichnet. Sie können vom Menschen in kürzester Zeit als solche identifiziert werden, unterscheiden sich jedoch in binärer Form sehr stark voneinander. Die automatisierte Erkennung dieser Duplikate anhand von Bildeigenschaften, welche ausschließlich aus den Binärdaten generiert werden, ist schon seit vielen Jahren Forschungsgegenstand. Jedoch unterstützen aktuelle Deduplikationssysteme oftmals nur textuelle Daten im gesamten Matching-Prozess. Die vorliegende Masterarbeit stellt das Konzept eines Systems vor, welches auf einer verteilten Infrastruktur eine bildbasierte Deduplikation großer Mengen von Bildern ermöglicht. Diese *Similar Image Matching Suite*, kurz *SIMaSu*, wurde zudem prototypisch unter Verwendung der Nachrichten-basierten Middleware RabbitMQ implementiert. Weiterhin gibt die Arbeit einen Überblick über die aktuell zur Verfügung stehenden Verfahren zur Berechnung von Bildähnlichkeiten. Dazu zählen *Perceptual Hash*-Technologien, *Feature*-basierte Verfahren und ein *Mean Square Error*-Ansatz. Solche Metriken stellen den Kern einer bildbasierten Deduplikation dar. Zusätzlich wurde eine Ähnlichkeitsmetrik konzipiert, welche durch Anwendung der Feature-basierten Technologien SIFT, SURF und ORB einen Ähnlichkeitswert errechnet. In einer abschließenden Evaluation werden für elf ausgewählte Implementationen verschiedener Metriken die Laufzeiten evaluiert, die Invarianzen gegen Bildtransformationen untersucht, sowie die Effektivitäten verglichen. Durch diesen fairen Vergleich werden Entscheidungshilfen für oder gegen die Verwendung einer bestimmten Metrik, sowie der Wahl eines effektiven Grenzwertes zur Klassifikation eines Bildpaares geboten.

Abstract

Digital images that refer to one and the same real-world object are called duplicates. They can be identified as such by humans very quickly, but they are very different in binary form. The automated detection of these duplicates based on image properties, which are generated exclusively from the binary data, has been the subject of research for many years. However, current deduplication systems often only support textual data throughout the matching process. This Master's thesis presents the concept of a system that enables image-based deduplication of large quantities of images on a distributed infrastructure. This *Similar Image Matching Suite*, short *SIMaSu*, was also prototypically implemented using the message broker RabbitMQ. Furthermore, the paper gives an overview of the currently available methods for the calculation of image similarities. These include *perceptual hash* technologies, *feature*-based methods, and a *mean square error* approach. Such metrics represent the core of an image-based deduplication. In addition, a similarity metric has been designed which calculates a similarity value by using the feature-based technologies SIFT, SURF, and ORB. In a final evaluation, the runtimes are evaluated for eleven selected implementations of different metrics, the invariants against image transformations are examined and the efficiencies are compared. This fair comparison provides decision support for or against the use of a particular metric, as well as the choice of an effective threshold for classifying a pair of images.

Inhaltsverzeichnis

Abbildungsverzeichnis **7**

Abkürzungsverzeichnis **8**

1. Einleitung **9**
1.1. Motivation . . . 9
1.2. Zielsetzung und Abgrenzung . . . 10
1.3. Aufbau der Arbeit . . . 11

2. Grundlagen **13**

3. Problemanalyse **17**
3.1. Anwendungsfall . . . 17
3.2. Unterscheidungsmerkmale von Bildern . . . 20
3.3. Ähnlichkeitsmetriken . . . 24
3.4. Komplexität . . . 24

4. Stand der Forschung und Technik **25**
4.1. Deduplikationssysteme . . . 25
4.2. Perceptual Hash-Verfahren . . . 26
4.2.1. Average Hash . . . 27
4.2.2. Distance Hash . . . 27
4.2.3. Perceptive Hash . . . 27
4.2.4. Resize Pixel Compare . . . 28
4.3. Feature-basierte Verfahren . . . 28
4.3.1. Scale-invariant feature transform (SIFT) . . . 29
4.3.2. Speeded Up Robust Features (SURF) . . . 30
4.3.3. Oriented FAST and rotated BRIEF (ORB) . . . 31
4.4. Mittlere quadratische Abweichung . . . 31
4.5. Evaluationen . . . 32
4.6. Übersicht . . . 32

5. SIMaSu - Similar Image Matching Suite **34**
5.1. Anforderungen . . . 34
5.2. Architekturkonzept . . . 35
5.3. Feature-basierte Ähnlichkeitsmetrik . . . 37
5.3.1. Die ideale Ähnlichkeitsmetrik . . . 37
5.3.2. Konzeption . . . 38
5.3.3. Diskussion der Parallelisierbarkeit . . . 44

6. Prototypische Implementation **45**
6.1. Technologieauswahl . . . 45
6.2. Implementierung . . . 46
6.3. Prototyp der Feature-basierten Ähnlichkeitsmetrik . . . 48

7. Evaluationen **51**
7.1. Laufzeiten . . . 52

7.2. Invarianz-Experiment 55
7.3. Effektivität 60

8. Zusammenfassung und Ausblick **64**

A. Anhang **70**
A.1. CD 70
A.2. Diagramme 71

Abbildungsverzeichnis

1. Drei Fotografien einer Stadt am Fluss aus ähnlichen Blickwinkeln [31] 9
2. Darstellung der Farbwerte eines Pixels im RGB-Farbraum anhand eines Beispielbildes aus [31] 13
3. Beispielhafte Bilder einer Kreuzfahrt-Entität [1–4, 31] 18
4. Überblick der Verfahren zur Bildähnlichkeitsermittlung 25
5. Beispielhafte Visualisierung eines Matchings mit SIFT-Desktiptoren [17] 30
6. Systemkonzept des SIMaSu-Systems 35
7. Fotografien einer Landschaft aus zwei verschiedenen Perspektiven [31] 39
8. In Graustufen transformierte Bilder der Landschaft (vgl. [31]) 39
9. Farbige Markierungen der Keypoints nach erfolgter Keypoint-Detektion (vgl. [31]) 40
10. Farbige Linien zwischen den Features zur Markierung der Treffer (vgl. [31]) . . . 40
11. Durch den Ratio-Test reduzierte Treffer (vgl. [31]) 42
12. Übersicht der Ähnlichkeitsmetrik als Datenflussdiagramm 43
13. Architekturskizze des SIMaSu-Systems 46
14. Übersicht der durchgeführten Evaluationen mit zugehörigen Fragestellungen . . . 51
15. Ergebnisse der Laufzeitmessung (in Sekunden) mit Standardabweichung 54
16. Originalbilder des Invarianz-Experiments 55
17. Ähnlichkeitswerte der Metrik *Jenssegers pHash* für Rotationen 56
18. Ähnlichkeitswerte der Feature-basierten Metrik *ORB* für Rotationen 57
19. Ähnlichkeitswerte der Feature-basierten Metrik *SIFT* für Rotationen 58
20. Beispiel für Effektivitätsmaße einer idealen Metrik 60
21. Effektivitätsmaße der Metrik *Imagick compareImages* (ICI) im Intervall $[0, 0.1]$. 61
22. Effektivitätsmaße der Feature-basierten Metrik *SIFT* mit *Bruteforce-Matching* (SIB) 62
23. Effektivitätsmaße der *ResizePixelCompare*-Metrik (RPC) 71
24. Effektivitätsmaße der *pHash*-Metrik der Anwendung *ImageMagick* (ICP) 71
25. Effektivitätsmaße der *pHash*-Metrik von *Jenssegers* (JPH) 72
26. Effektivitätsmaße der *aHash*-Metrik von *Jenssegers* (JAH) 72
27. Effektivitätsmaße der *dHash*-Metrik von *Jenssegers* (JDH) 73
28. Effektivitätsmaße der Feature-basierten Metrik *SIFT* mit *FLANN-Matching* (SIF) 73
29. Effektivitätsmaße der Feature-basierten Metrik *SURF* mit *FLANN-Matching* (SUF) 74
30. Effektivitätsmaße der Feature-basierten Metrik *SURF* mit *Bruteforce-Matching* (SUB) 74
31. Effektivitätsmaße der Feature-basierten Metrik *ORB* 75

Abkürzungsverzeichnis

ASIFT	Affine-SIFT
BRIEF	Binary Robust Independent Elementary Features
CBIR	Content Based Image Retrieval
DL	Deep Learning
DoG	Difference of Gaussians
FAST	Features from accelerated segment test
FLANN	Fast Approximate Nearest Neighbor Search
FN	False Negative
FP	False Positive
JRE	Java Runtime Enviroment
ML	Machine Learning
MSE	Mean Square Error
ORB	Oriented FAST and rotated BRIEF
px	Pixel
SIFT	Scale-invariant feature transform
SURF	Speeded Up Robust Features
TN	True Negative
TP	True Positive
URL	Uniform Resource Locator
VF-SIFT	Very Fast SIFT Feature Matching

1. Einleitung

> *[Die] Bildanalyse [...] gilt aber nach wie vor als Königsdisziplin der modernen Computerwissenschaft.* [23]

1.1. Motivation

Unstrukturierte Daten, wie Texte, Dokumente, Bilder oder Videos, machen nach wie vor die Mehrheit der im Web und Unternehmen vorhandenen Daten aus und wachsen deutlich schneller als strukturierte Daten (vgl. [14]). Laut einer in 2011 veröffentlichten IDC-Studie [20] wird im nächsten Jahrzehnt der Anteil an produzierten unstrukturierten Daten 90 Prozent ausmachen. Gerade deshalb spielen diese für den *Big Data*-Bereich eine enorme Rolle, da darin technologische Möglichkeiten geboten werden, um unter anderem sehr große Datenmengen zu verarbeiten und zu analysieren. Im gleichen Kontext wird oftmals die Untersuchung und Informationsbeschaffung aus Texten betrachtet. Damit verglichen macht die Extraktion und Analyse von Informationen aus Bildern und Videos nur einen geringen Teil aktueller Forschungen aus (vgl. [23]). Ähnliches ist bei der Deduplikation von Entitäten wahrzunehmen, welche einen wesentlichen und schwierigen Bestandteil der Datenintegration und -bereinigung darstellt. Aktuelle Frameworks aus dieser Domäne bieten eine umfangreiche Unterstützung von textuellen Daten. Multimediale Entitäten werden jedoch in den seltensten Fällen unterstützt.

Digitale Bilder, die ein und dasselbe Realweltobjekt abbilden, werden als Duplikate bezeichnet. Sie können vom Menschen in kürzester Zeit als solche identifiziert werden, unterscheiden sich jedoch in vielen Fällen in binärer Form sehr stark voneinander. Bildduplikate entstehen beispielsweise durch digitale Änderungen an einem Originalbild. Das können Rotationen, Beschneidungen, Anpassungen von Helligkeits- und Farbwerten, das Hinzufügen von Änderungen, wie Wasserzeichen oder das Retuschieren von Bildbereichen, sowie die mehrfache Aufnahme zu unterschiedlichen Zeitpunkten bzw. aus verschiedenen Perspektiven sein. Abbildung 1 zeigt beispielhaft drei Fotografien einer Stadt, welche durch diese Definition als Duplikate interpretiert werden.

Abbildung 1: Drei Fotografien einer Stadt am Fluss aus ähnlichen Blickwinkeln [31]

Unternehmen, die eine Vielzahl an Bildern zu ihren Datenbeständen zählen und Teilmengen davon ihren Kunden zur Verfügung stellen, wollen das diese gewissen Qualitätsansprüchen genügen. Ein Anspruch kann sein, dass Bilder nicht mehrfach vorkommen dürfen. Ein manuelles Aussortieren von Duplikaten bedeutet, schon bei Datenmengen in Größenordnungen von mehreren tausend Bildern, einen enormen Aufwand. Die Anwendung von geeigneten automatisierten Verfahren ist für die Selektion dieser Bilder unabdingbar.

Ein Anwendungsfall für einen solche automatisierte Suche nach Duplikaten (auch als *Image-Matching* bezeichnet) ist die Bereinigung einer Bilddatenbank, die nach einem Integrationsschritt aus mehreren Quellen entstanden ist. Als Beispiel kann ein Unternehmen betrachtet werden, welches als Anbieter von Stadtrundgängen fungiert. Dieses bezieht aus verschiedenen Online-Quellen Bilder von Sehenswürdigkeiten der angebotenen Städte. Die Wahrscheinlichkeit ist hoch, dass nach deren Zusammenführung ungewollte Duplikate existieren, welche automatisiert ermittelt und entfernt werden müssen. Auch die Suche nach Verstößen gegen das Urheberrecht (z. B. durch nicht erlaubtes Duplizieren und Modifizieren eines geschützten Bildes) oder das Finden von alternativen Versionen von gegebenen Bildern, können als Anwendungsfall angesehen werden. Rosebrock verwendet einen entsprechenden Automatismus beispielsweise zur Detektion von pornografischen Material auf einer Dating-Plattform [50]. Bevor bei dem Anbieter der Webseite ein automatisiertes Verfahren eingesetzt wurde, waren Mitarbeiter ganztägig damit beschäftigt, Profilbilder der Kunden anzuschauen und pornografische Abbildungen händisch zu blockieren. Dies verursachte Kosten von mehreren tausend Euro im Monat. Auch die Bildersuche von Google [21] nutzt bestimmte *Image-Matching*-Verfahren zum Auffinden von Bildern, die einem vom Nutzer vorgegebenen Bild ähneln.

Zur Erkennung von Duplikaten werden verschiedene, sich in Komplexität und Methodik unterscheidende, Verfahren verwendet. Sie entsprechen im Allgemeinen einer Metrik, deren resultierender Ähnlichkeitswert Aufschluss über die inhaltliche Übereinstimmung zweier Bilder gibt. Solche Verfahren sind beispielsweise Algorithmen zur Detektion interessanter Bildbereiche, sogenannter *Features*, oder die Berechnung eines digitalen Fingerabdrucks aus den Bildeigenschaften. Durch einen anschließenden Vergleich der ermittelten Werte wird das Ähnlichkeitsmaß berechnet. Je nach Anwendungsfall unterscheidet sich die Eignung spezifischer Verfahren und Systeme, die diese implementieren, deutlich. Deren Effektivität kann beispielsweise durch Einsatz eines Testdatensatzes ermittelt werden, welcher sowohl Duplikate, als auch sich unterscheidende Bilder enthält. Ein Qualitätsanspruch für ein solches System ist die korrekte Identifikation möglichst vieler Duplikate. Dabei wird eine Annäherung an die relevante Menge von semantisch äquivalenter Bilder angestrebt.

Neben der Effektivität spielt auch die Laufzeit der gesamten Deduplizierung eine Rolle. Da bei einem naiven Ansatz alle Bilder eines Datensatzes mit allen verglichen werden müssen, erhöht sich der Zeitaufwand quadratisch bei linearem Wachstum der Bildmenge. Geeignete Verfahren zur Reduktion der Vergleichspaare sowie zur parallelen Ausführung einzelner Teilprozesse auf mehreren Systemen sind zur Bewältigung großer Datensätze notwendig.

1.2. Zielsetzung und Abgrenzung

Ziel dieser Masterarbeit ist es, einen Überblick über die aktuell zur Verfügung stehenden Verfahren zur Berechnung von Bildähnlichkeiten, sowie Systeme und Bibliotheken, die diese Verfahren implementieren, zu erstellen. Diese werden dabei hinsichtlich des jeweiligen Funktionsumfangs und der zugrundeliegenden Technologien analysiert. Weiterhin wird auch ein grundsätzliches Verständnis der jeweiligen zugrundeliegenden Algorithmen verfolgt. Eigenschaften, wie Konfigurierbarkeit des Systems und Eignung für eine verteilte Ausführung auf mehreren Maschinen, werden zusätzlich erfasst. Durch Evaluationen der einzelnen Systeme wird deren Robustheit gegenüber Bildmanipulationen untersucht, die durchschnittliche Laufzeit für verschiedene Bildgrößen gemessen sowie deren Effektivität für einen ausgewählten Datensatz ermittelt.

Aus den erlangten Kenntnissen wird im Anschluss ein System konzipiert, mit dem eine bildbasierte Deduplikation durchgeführt werden kann. Die vorab ermittelten Ansätze zur Berechnung der Bildähnlichkeiten werden an dieses System angebunden. Dabei wird deren Kombination zur Verbesserung des Ergebnisses ermöglicht. Weiterhin wird großen Wert auf die Skalierbarkeit des Systems gelegt, damit auch sehr große Datenmengen mit möglichst geringem Zeitaufwand verarbeitet werden können. Zusammengefasst soll das System eine Menge an Bildern entgegennehmen, diese verteilt verarbeiten und gefundene Bildduplikate gruppiert als Ergebnis zurückgeben. Weiterhin soll eine prototypische Implementierung des Systems erfolgen, welche primär die Durchführung der Evaluationen ermöglicht.

Die Algorithmen, welche lokale *Features* aus Bildern extrahieren, wurden primär zur Erkennung von Objekten entwickelt. Deren Funktionsweise kann jedoch auch genutzt werden, um eine Aussage zur inhaltlichen Übereinstimmung zweier Bilder zu liefern. Aus diesem Grund soll eine Ähnlichkeitsmetrik konzipiert werden, um aus den übereinstimmenden *Features* zweier Bilder einen Wert zu ermitteln, welcher Aufschluss über die semantische Ähnlichkeit der Abbildungen gibt. Diese Metrik soll prototypisch umgesetzt und an das oben genannte System angebunden werden.

Innerhalb dieser Masterarbeit wird kein neues Verfahren zur Berechnung von Bildähnlichkeiten entwickelt, sondern ausschließlich auf vorhandene Technologien zurückgegriffen. Weiterhin wird von einer theoretischen Erläuterung der zugrundeliegenden Algorithmen sowie deren detaillierten Abläufen größtenteils abgesehen. Es soll lediglich ein Grundverständnis des jeweiligen Prinzips dargelegt werden. Wenn eine detailliertere Erläuterung in der Literatur zur Verfügung steht, wird auf diese verwiesen. Weiterhin wird von der Verwendung von Technologien aus den Bereichen Machine Learning (ML) und Deep Learning (DL) abgesehen, da diese spezielle Anforderungen an die Hardware besitzen und zudem einen hinreichend großen Trainingsdatensatz als Grundlage für deren Funktionsfähigkeit benötigen.

Als Bilder werden in der gesamten Arbeit ausschließlich digitale Fotografien betrachtet. Vektorgrafiken, Piktogramme, gescannte Dokumente und andere Arten von Grafiken, die in diesen Bereich fallen, werden bei den Tests und Evaluationen nicht analysiert. Ebenfalls wird die Erkennung von Gesichtern und Objekten nicht betrachtet.

1.3. Aufbau der Arbeit

Insgesamt besteht diese Masterarbeit aus acht Kapiteln, Einleitung und Zusammenfassung eingeschlossen. In Kapitel 2 wird zu Beginn auf grundlegende Begrifflichkeiten, wie das digitale Bild, Deduplikation und Effektivität, eingegangen. Anschließend wird im 3. Kapitel eine umfangreiche Problemanalyse aufgeführt, welche einen thematischen Einstieg bildet. Darin wird unter anderem ein Anwendungsfall beleuchtet, welcher grundsätzliche Problematiken bei der Deduplikation von Bildern aufzeigt. Weiterhin wird eine Vielzahl von Eigenschaften aufgeführt, anhand derer sich digitale Bilder unterscheiden können.

Im weiteren Verlauf der Arbeit wird der aktuelle Stand der Forschung und Technik, bezogen auf die hier betrachteten Themengebiete erläutert (Kapitel 4). Hauptbestandteil ist ein Überblick über bestehende Verfahren zur Ermittlung der Ähnlichkeit zweier Bilder. Mit einer Selektion von Implementationen, welche innerhalb dieser Arbeit evaluiert werden, wird dieses Kapitel abgeschlossen.

Im Kapitel 5 wird die Konzeption einer skalierbaren Anwendung zur bildbasierten Deduplikation beschrieben. Das Design einer Feature-basierten Ähnlichkeitsmetrik wird anhand schrittweiser Erläuterungen ebenfalls darin aufgeführt. Die prototypische Implementation beider Entwürfe erfolgt im anschließenden Kapitel.

Drei umfangreiche Evaluationen sind Bestandteil des 7. Kapitels. Darin werden Laufzeiten, Effektivität und Invarianz gegen Bildmanipulationen für ausgewählte Ähnlichkeitsmetriken ermittelt und diskutiert. Den letzten Teil der Masterthesis bildet eine Zusammenfassung mit abschließendem Ausblick.

2. Grundlagen

In diesem Kapitel werden die grundlegenden Begrifflichkeiten *digitales Bild*, *Duplikat*, *Deduplikation* und *Effektivität* definiert. Sie dienen dem Grundverständnis dieser Arbeit.

Das digitale Bild In dem Buch *Computer Vision* definiert Priese formal ein digitales Bild als eine Abbildung von einem Ortsbereich in einen Wertebereich, für die zu jedem Ortswert genau ein Wert existiert (vgl. [48, S. 62ff]). Im Lexikon des *Spektrum Verlages* [61] ist ein digitales Bild als geordnetes Schema diskreter Intensitätswerte[1] definiert. Ein einzelnes Element des digitalen Bildes ist in der Regel ein rechteckiges *Pixel*. Innerhalb eines Bildes sind die Pixel in senkrechten Spalten und waagerechten Zeilen in einem Raster angeordnet. Ein Pixel ist neben seiner Position im Raster durch einen diskreten Intensitätswert I definiert, welcher den Mittelwert der Helligkeiten für die Fläche des jeweiligen Pixels abbildet. Bei einem farbigen Bild im RGB-Farbraum mit einer Farbtiefe von 8 Bit ist der Intensitätswert ein dreidimensionaler Vektor, wobei standardmäßig jede Dimension den entsprechenden Farbwert für Rot, Grün und Blau als Binärwort der Länge 8 darstellt. Der Farbwert wird in der Literatur oft als Dezimalwert angegeben. Abbildung 2 verdeutlicht die einzelnen Bestandteile anhand eines Beispielbildes. (Vgl. [47, 61])

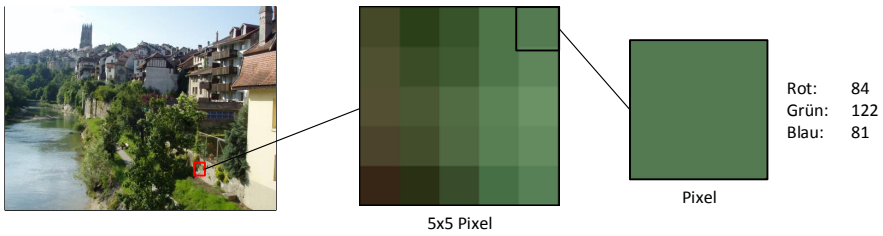


Abbildung 2: Darstellung der Farbwerte eines Pixels im RGB-Farbraum anhand eines Beispielbildes aus [31]

Innerhalb dieser Masterarbeit wird der Begriff *Bild* stellvertretend für ein zweidimensionales, digitales und farbiges Rasterbild im RGB-Farbraum verwendet.

Duplikat Ob zwei Bilder als ähnlich gelten, ist eine subjektive Wahrnehmung und hängt vom jeweiligen Betrachter oder Anwendungsfall ab. Neben dem Begriff *Ähnlichkeit* (engl. *similarity*) gilt dies auch für verwandte Begriffe, wie *Gleichheit* (engl. *equality*) oder *Duplikat* (engl. *duplicate*). Die Begriffe werden vom Menschen unterschiedlich interpretiert. Werden beispielsweise zwei Fotografien desselben Gebäudes aus verschiedenen Perspektiven für einige Personen als *gleich* oder *ähnlich* klassifiziert, sind diese jedoch für andere zwei *verschiedene* Bilder. Analog dazu kann die Interpretation auch von einem Anwendungsfall abhängen. Für das in der Einleitung beschriebene Szenario der Detektion von unrechtmäßigen pornografischen Material, gelten zwei Bilder von unterschiedlichen nackten Personen als *ähnlich*, da diese in diesem speziellen Fall herausgefiltert werden sollen. Im Gegensatz dazu steht das ebenfalls in der Einleitung erwähnte Beispiel der Suche nach Verstößen gegen das Urheberrecht. Sollen unrechtmäßige Kopien einer Aktfotografie gefunden werden, dürfen die zwei Bilder von unterschiedlichen, nackten Personen

[1] Ein Intensitätswert beschreibt die Helligkeit eines einzelnen Bildpunktes in einem festgelegten Wertebereich.

nicht als *Duplikat* identifiziert werden. Ausschließlich eine digital veränderte Version der Aktfotografie oder die Fotografie selbst gelten als *ähnlich*, *gleich* oder werden als *Duplikat* eingestuft.

Im Kontext dieser Masterarbeit sind zwei Bilder Duplikate, wenn sie eine digitale Repräsentation desselben Realwelt-Objekts sind. Unterscheiden sich beide Bilder in deren Perspektive, Größe oder wurden digital manipuliert, gelten sie trotzdem als Duplikat.

Deduplikation Das Auffinden von Duplikaten ist Bestandteil des *Entity-Matching-Problems*, welches von Köpcke und Rahm in [37] wie folgt definiert wird:

> Seien zwei Mengen an Entitäten $A \in S_A$ und $B \in S_B$ eines bestimmten semantischen Typs der Datenquellen S_A und S_B gegeben, bezeichnet das *Entity-Matching-Problem* die Identifikation aller Korrespondenzen zwischen den in $A \times B$ enthaltenen Entitäten, welche dieselben Realwelt-Objekte repräsentieren. Die Definition beinhaltet den speziellen Fall, Paare äquivalenter Entitäten einer einzigen Quelle ($A = B, S_A = S_B$) zu finden. Das Ergebnis wird typischerweise durch eine Menge von Korrespondenzen (*Mapping*) oder Cluster repräsentiert. Eine Korrespondenz $c = (e_i, e_j, s)$ verbindet zwei Entitäten e_i und e_j der Quellen S_A und S_B. Ein optionaler Ähnlichkeitswert $s \in [0, 1]$ zeigt die Ähnlichkeit oder Stärke der Korrespondenz zwischen den beiden Objekten. Bei der alternativen Ergebnis-Repräsentation beinhaltet ein Cluster Entitäten, welche als Repräsentation desselben Realwelt-Objektes angesehen werden. Idealerweise referenzieren alle Entitäten innerhalb eines Clusters dasselbe Objekt, und zwei Entitäten von zwei verschiedenen Clustern nie dasselbe Objekt. [37]

Der Prozess des Entity-Matchings wird in der Literatur auch als *Entity-Resolution*, *Object-Matching*, *Duplicate-Detection*, *Reference-Reconciliation*, *Record-Linkage* oder *Deduplikation* bezeichnet und ist ein wesentlicher Bestandteil der Datenintegration und -bereinigung (vgl. [38]). Handelt es sich bei den Objekten um Bilder und wird der Matching-Prozess ausschließlich anhand der aus dem Bild extrahierten Informationen durchgeführt, fällt dieser Prozess in den Bereich des Content Based Image Retrieval (CBIR) (vgl. [34, 56]). Innerhalb dieser Arbeit wird das Matching zur Erkennung von Duplikaten aus einer Menge von Bildern verwendet. Daher wird die Bezeichnung *bildbasierte Deduplikation* als passender Ausdruck für den Matching-Prozess ausgewählt.

Überträgt man die standardmäßigen Abfolgen einer allgemeinen Deduplikation, welche beispielsweise von Kolb et al. in [36] beschrieben wird, auf die *bildbasierte* Deduplikation, entsteht folgende Abfolge von Teilprozessen:

1. Laden der Bilder (bzw. deren Pfade)
2. Anwenden eines Blocking-Ansatzes (optional)
3. Bilden von Vergleichspaaren innerhalb eines Blocking-Clusters
4. Berechnen von n Ähnlichkeitsmaßen pro Vergleichspaar
5. Prüfen ob sich der Ähnlichkeitswert innerhalb eines Grenzwertes befindet (*Match-Decision*)
6. Bilden der Korrespondenzen oder Cluster

Der *Blocking*-Ansatz ist eine Methode, um die Menge der Vergleichspaare und somit die Komplexität der Deduplikation zu verringern. Ziel ist es, ähnliche Entitäten in Gruppen bzw. Blöcke zusammenzufassen, sodass sich mit einer hohen Wahrscheinlichkeit alle Duplikate innerhalb eines Blocks befinden. Es wird somit aus einer Menge an Entitäten, eine Menge an Blöcken erzeugt. Bei einem *Standard-Blocking* wird intern jede eingehende Entität in eine kompakte Repräsentation (z. B. ein Bitwort) überführt, welche die Eigenschaften der Entität zusammenfasst. Diese Repräsentation wird als Blocking-Key bezeichnet. Vergleichspaare werden entsprechend nur innerhalb eines Blocks bzw. mit denselben Blocking-Key gebildet, was den Suchraum einschränkt und somit die Effizienz erhöht. Die Zuordnung einer Entität zu mehr als einem Block ist dabei auch möglich. Blocking stellt bei Bild-Entitäten eine enorme Herausforderung dar, da zunächst geeignete Technologien zur Abbildung eines Bildes auf eine Eigenschaft, die zur Gruppierung genutzt werden kann, gefunden werden müssen.

Effektivität Die Effektivität einer Deduplizierung wird im Allgemeinen hinsichtlich der Maße *Precision (P)*, *Recall (R)* und *F-Measure (F)* evaluiert (vgl. [37]). Diese Maße werden standardmäßig bei Evaluationen von *Information Retrieval*-Systemen verwendet. Sie beziehen sich auf einen vorab festgelegten *Goldstandard*, welcher durch manuelle Klassifizierung erstellt werden muss. Er legt somit die perfekte Ergebnismenge fest, welche in dem hier betrachteten Fall alle Bildpaare sind, die Duplikate darstellen. Identifiziert die Deduplizierung ein Vergleichspaar als Duplikat/Treffer bzw. nicht Duplikat/kein Treffer, wird dieses Ergebnis entsprechend einer Wahrheitsmatrix (Tabelle 1) klassifiziert. Die enthaltenen farblichen Markierungen sind relevant für das nächste Kapitel. Da innerhalb der Literaturquellen ausschließlich die englischen Begriffe verwendet werden, wird dies auch in dieser Arbeit beibehalten.

	Treffer	Kein Treffer
Relevant	True Positive (TP)	False Negative (FN)
Nicht relevant	False Positive (FP)	True Negative (TN)

Tabelle 1: Wahrheitsmatrix

Wurden Duplikate durch den Matching-Prozess korrekt identifiziert, gelten diese als *True Positives*. Ein Duplikat, welches nicht als solches erkannt wurde, bezeichnet man als *False Negative*. Wurde für zwei verschiedene Bilder ein Treffer ermittelt, ist dies ein *False Positive*. *True Negatives* sind semantisch verschiedene Bilder, welche auch entsprechend korrekt identifiziert wurden. Durch diese Definition ergeben sich die Berechnungsvorschriften für *Precision (P)*, *Recall (R)* und *F-Measure (F)*, welche den Gleichungen 1, 2 und 3 zu entnehmen sind. (Vgl. [37])

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

$$R = \frac{\|TP\|}{\|TP\| + \|FN\|} \in [0,1] \tag{2}$$

$$F = \frac{2PR}{P+R} \in [0,1] \tag{3}$$

Die *Precision* gibt somit an, wie viele der ermittelten Duplikate ($\|TP\| + \|FP\|$) auch relevant sind ($\|TP\|$). Wie viele relevante Treffer ($\|TP\|$) bezogen auf die im Goldstandard definierte Gesamtmenge der relevanten Duplikate ($\|TP\| + \|FN\|$) gefunden wurden, gibt der *Recall* an. Das harmonische Mittel von *Precision* und *Recall* ist das *F-Measure*.

Abhängig von der Anzahl der zu untersuchenden Entitäten und der Anzahl der verwendeten Ähnlichkeitsmetriken ist die Deduplikation ein sehr teurer und zeitaufwändiger Prozess und somit interessant für *Big Data*-Anwendungen. Im vorherigen Abschnitt wurde schon der Blocking-Ansatz beschrieben, welcher die Effizienz durch Verminderung des Suchraums erhöht. Trotzdem bleibt die Deduplikation ein ressourcenintensiver Prozess und ist somit ein ideales Problem für eine parallele Ausführung auf verteilten Infrastrukturen. (Vgl. [35])

3. Problemanalyse

In diesem Kapitel werden die zugrundeliegenden Problemstellungen des in dieser Arbeit betrachteten Themas analysiert. Anhand eines Anwendungsfalles (Abschnitt 3.1) wird sowohl ein thematischer Einstieg geschaffen, als auch die Schwierigkeiten beim Matching-Vorgang für zweidimensionale Bilder dargelegt. Ein Überblick möglicher Transformationen und Manipulationen als Unterscheidungsmerkmale zweier Bilder wird im Abschnitt 3.2 gegeben. Weitere Probleme beim Prozess der Deduplikation von Bildern werden in den letzten beiden Abschnitten dieses Kapitels aufgezeigt.

3.1. Anwendungsfall

Der nachfolgend beschriebene Anwendungsfall stellt ein reales Problem eines in Deutschland ansässigen mittelständischen Unternehmens dar. Dabei wurden Unternehmensname und Branche fiktiv gewählt. Das beschriebene Deduplikations-Szenario nutzt zum besseren Verständnis der Problematik nur eine einfache Ähnlichkeitsmetrik.

Die Firma *mycruisetrip.eu* bietet auf ihrer Onlineplattform Angebote für eine Vielzahl verschiedener überregionaler Kreuzfahrten an. Der Kunde wählt auf der Plattform zunächst An- und Abreisetag sowie die Urlaubsregion aus. Ihm wird daraufhin eine Auflistung der für die Filterkriterien passenden Kreuzfahrten angezeigt. Nach Selektion eines Angebots, wird dem Kunden eine detaillierte Übersicht mit allen Reisedaten sowie der Reiseroute präsentiert. Außerdem wird eine Galerie mit einer Vielzahl an Bildern von dem Kreuzfahrtschiff und dessen Eigenschaften, wie Zimmerausstattung und Freizeitangeboten, sowie Bilder der angefahrenen Urlaubsorte und deren Sehenswürdigkeiten, angezeigt.

Die verfügbaren Bilder bezieht das Unternehmen von vier externen Anbietern, die sich auf die kommerzielle Bereitstellung von Bildmaterial aus dieser Branche spezialisiert haben. Die Datenintegration wird durch Mapping-Tabellen realisiert, welche von den Anbietern zur besseren Zuordnung zu einer Kreuzfahrt-Entität zur Verfügung gestellt werden. Alle Bilder werden im firmeneigenen Datacenter gespeichert und sind über eine Uniform Resource Locator (URL) abrufbar. Zusätzlich wird pro Bild eine Bild-Entität mit Attributen wie URL und Abmessungen innerhalb einer relationalen Datenbank abgelegt und mit der jeweiligen Kreuzfahrt-Entität verknüpft. Jede Bild-Entität hat zusätzlich auch ein Attribut nachdem entschieden wird, ob das Bild auf der Onlineplattform veröffentlicht wird oder nicht. Aufgrund mangelnder Datenqualität einiger Anbieter werden oftmals auch mehrere identische Bilder mit teilweise unterschiedlichen Maßen, Auflösungen und Orientierungen verteilt. Weiterhin liefert ein Anbieter alle Bilder mit einem Wasserzeichen am unteren Bildrand zur Identifikation der Datenquelle aus. Ein anderer Anbieter bietet wiederum eine hohe Datenqualität und verfügt ausschließlich über digital verbesserte Bilder mit hoher Auflösung. Kurz gesagt, weist die Menge aller einer Kreuzfahrt-Entität zugeordneten Bilder im Allgemeinen viele Duplikate auf. Verständlicherweise möchte das Unternehmen auf der Onlineplattform ausschließlich Bilder veröffentlichen, die frei von Duplikaten sind, um die Benutzererfahrung so positiv wie möglich zu gestalten. Es ist somit notwendig, alle Bild-Entitäten einer Kreuzfahrt-Entität gegeneinander abzugleichen, um Duplikate zu detektieren und zu gruppieren. Aus jeder Duplikatgruppe wird im Anschluss automatisiert jeweils ein Bild zur Veröffentlichung ausgewählt.

Zur maschinellen Detektion von Duplikaten verwendet das Unternehmen *Imagick* [25], eine native PHP-Erweiterung, welche es ermöglicht unter Verwendung der API *ImageMagick* [29] Bilder zu erzeugen, zu bearbeiten und miteinander zu vergleichen. Über die darin implementierte Funktionalität *compareImages* [30] wird aus zwei Eingabebildern ein Ähnlichkeitswert mit einem Wertebereich zwischen 0 und 1 berechnet. Dabei entspricht der Wert 0 eine sehr hohe und 1 keine Ähnlichkeit. Es erfolgt im Anschluss die Klassifikation der Vergleichs-Paare. Dabei wird über einen festgelegten Grenzwert entschieden, ob es sich um Duplikate oder semantisch verschiedene Bilder handelt. Aus den ermittelten Duplikaten wird die transitive Hülle gebildet und somit logische Gruppen erzeugt.

Abb. 3 zeigt exemplarisch sechs Bilder, die nach dem Integrationsschritt der selben Kreuzfahrt-Entität zugeordnet wurden. Abbildung 3(a) und 3(b) sind Fotografien des Kreuzfahrtschiffes *Aida Stella*, welche aus leicht verschiedenen Blickwinkeln mit unterschiedlichen Auflösungen fotografiert wurden. Die tatsächlichen Bilddimensionen (Breite x Höhe in Pixel) sind den Bildunterschriften zu entnehmen. Weiterhin sind digitale Manipulationen an 3(a) zu erkennen, wenn man den Himmel im Hintergrund betrachtet. Da es sich um das semantisch gleiche Kreuzfahrtschiff aus einem leicht verschiedenen Blickwinkel handelt, beurteilen die Mitarbeiter des Produktmanagements von *mycruisetrip.eu* beide Bilder als Duplikate. Abbildung 3(c) ist eine Abbildung des Schiffes *Aida Cara*, welches fälschlicherweise zu dieser Kreuzfahrt-Entität referenziert wurde. Die Bilder 3(d) und 3(e) zeigen eine Kleinstadt, welche ein Ausflugsziel der Kreuzfahrt darstellt. Ein weiteres Ausflugsziel ist der in 3(f) dargestellte Ort, welcher nicht mit der Kleinstadt gleichzusetzen ist.

(a) 1000 x 658 [1] (b) 4651 x 2708 [2] (c) 900 x 600 [3]

(d) 2560 x 1920 [31] (e) 2560 x 1920 [31] (f) 360 x 240 [4]

Abbildung 3: Beispielhafte Bilder einer Kreuzfahrt-Entität [1–4, 31]

Zusammenfassend gilt als Anforderung, ausschließlich die Paare {1(a),1(b)} und {1(d),1(e)} als Duplikate zu identifizieren. Die Ähnlichkeitswerte für diese beiden Paare sollten sich demnach deutlich von allen anderen Paaren abgrenzen, damit eine korrekte Klassifikation vorgenommen werden kann.

Tabelle 2 zeigt die aus der Berechnung resultierenden Ähnlichkeitswerte innerhalb einer Matrix,

welche nachfolgend als *Ähnlichkeits-Matrix* bezeichnet wird. Zur Wiederholung gibt ein kleiner Wert eine hohe Ähnlichkeit an, wohingegen ein großer Wert hohe Unterschiede der Bilder impliziert. Die farbigen Markierungen entsprechen der Zuordnung eines Vergleichspaares zu einer Kategorie der Wahrheitsmatrix (siehe Kapitel 2). Nach Tabelle 1 sind diese dabei wie folgt zu interpretieren: *True Positives*, *False Positives* und *True Negatives*.

	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)
1(a)	0	0.102	0.076	0.191	0.310	0.152
1(b)	-	0	0.210	0.131	0.118	0.229
1(c)	-	-	0	0.229	0.315	0.172
1(d)	-	-	-	0	0.122	0.208
1(e)	-	-	-	-	0	0.246
1(f)	-	-	-	-	-	0

Tabelle 2: Mit Imagick berechnete Ähnlichkeitsmaße (auf 3 Dezimalstellen gerundet) für die in Abb. 3 dargestellten sechs Bilder. Der zur Klassifikation verwendete Grenzwert beträgt 0.125.

Man erkennt, dass aus dem Vergleich eines Bildes mit sich selbst immer ein Wert von 0 resultiert. Dieses Verhalten weist auf eine reflexive Eigenschaft des verwendeten Algorithmus der *Imagick*-Bibliothek hin. Weiterhin ist die entstandene Ähnlichkeits-Matrix symmetrisch[2], was bedeutet, dass die Reihenfolge der Eingabebilder keine Auswirkungen auf das Ergebnis hat. Zur besseren Übersicht wurden die redundanten Werte der unteren Dreiecksmatrix weggelassen. Um die Vergleichsmenge möglichst gering zu halten, werden aufgrund dieser Eigenschaften für alle nachfolgenden Betrachtungen nur Vergleichspaare herangezogen, die der strikten oberen Dreiecksmatrix entsprechen. Einfacher ausgedrückt wird bei zwei Bildern $1(a)$ und $1(b)$ keine Metrik für die Paare $\{1(a), 1(a)\}$, $\{1(b), 1(a)\}$ und $\{1(b), 1(b)\}$ berechnet. Die Anzahl der Vergleiche reduziert sich bei n Bildern von n^2 auf $\frac{n^2-n}{2}$.

Für die in der Anforderung definierten Paare {1(a),1(b)} und {1(d),1(e)} wurden Ähnlichkeiten von 0.102 und 0.122 berechnet. Damit diese als Duplikat eingeordnet werden, wird ein Grenzwert von 0.125 festgelegt. Alle Vergleichspaare mit einem Ähnlichkeitswert ≤ 0.125 sind somit Duplikate. Die *False Positives* stellen dabei ein Problem dar, da deren Ähnlichkeitswerte innerhalb des Grenzwertes liegen und somit fälschlicherweise als Duplikate eingeordnet werden. Die Wahl eines kleineren Grenzwertes, beispielsweise 0.07 würde die *False Positives* verhindern. Jedoch werden die beiden *True Positives* damit auch zu *False Negatives*. Wenn die Vermeidung von *False Positives* für den hier beschriebenen Anwendungsfall von höherer Relevanz ist, als das Finden von Duplikaten, müsste sich für einen niedrigeren Grenzwert entschieden werden. Folgende Probleme können somit zusammenfassend definiert werden:

> **Problem 1: False Negatives** Der Ähnlichkeitswert einer Metrik lässt sich bei zwei semantisch *äquivalenten* Bildern nicht signifikant von dem Wert unterscheiden, der für zwei semantisch *verschiedene* Bilder erzeugt wird. Dies erschwert die Klassifikation und erhöht die Chance auf *False Negatives*.

[2] Eine reelle quadratische Matrix $\boldsymbol{A}$ heißt *symmetrisch*, wenn gilt $\boldsymbol{A} = \boldsymbol{A}^T \Leftrightarrow a_{ik} = a_{ki}$ für alle $i, k = 1, 2, ..., n$. [7, S. 180]

Problem 2: False Positives Der Ähnlichkeitswert einer Metrik lässt sich bei zwei semantisch *verschiedenen* Bildern nicht signifikant von dem Wert unterscheiden, der für zwei semantisch *äquivalente* Bilder erzeugt wird. Dies erschwert die Klassifikation und erhöht die Chance auf *False Positives*.

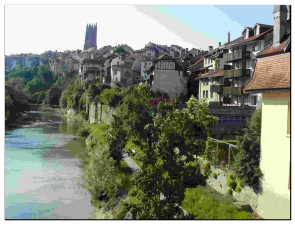
Bezogen auf die Beispielbilder des Anwendungsfalls und die verwendete Metrik besteht *Problem 1* nicht. Für die relevanten Bildpaare ergaben sich hinreichend kleine Ähnlichkeitswerte bezogen auf den Wertebereich $[0, 1]$ der Metrik. *Problem 2* ist jedoch deutlich zu vernehmen. Die Bilder des Paares $\{1(b), 1(e)\}$ unterscheiden sich sowohl semantisch als auch rein optisch sehr voneinander. Trotzdem ergab die Berechnung einen Wert von 0.118, welcher sich nicht signifikant von den Werten der korrekten Treffern abhebt. Diese Problematik kann in Unternehmen mit kritischen Anwendungsfällen zu zusätzlichen personellen Kosten führen, da die Ergebnisse von Mitarbeitern kontrolliert werden müssen.

3.2. Unterscheidungsmerkmale von Bildern

Im vorherigen Abschnitt wurde anhand von Beispielbildern (Abb. 3) gezeigt, wie sich Bilder rein optisch unterscheiden, jedoch das semantisch äquivalente Objekt abbilden können. Im Falle des Kreuzfahrtschiffes unterschieden sich die Bilder anhand der Perspektive, von der sie aufgenommen wurden, sowie von verschiedenen anderen Bildeigenschaften, wie Farbwerte, Sättigung und Helligkeit. Durch aktuell auf den Markt befindliche Bildverarbeitungssoftware (z. B. *Photoshop* von Adobe) ist die Manipulation bzw. Veränderung von Bildern oftmals mit nur sehr wenig Aufwand möglich. Gerade im kommerziellen Bereich werden Bilder oftmals mit einem Wasserzeichen (z. B. ein Firmenlogo) versehen oder entsprechend der jeweiligen Domäne „verbessert“. Für Systeme und Verfahren, die zwei Bilder anhand ihres Inhaltes als Duplikat einstufen sollen, stellen solche Transformationen ein großes Problem dar. Wie man später innerhalb dieser Arbeit erkennen kann (siehe Kapitel 7), reichen oft schon kleine Veränderungen eines Bildes, dass dieses bei Anwendung einfacher Metriken nicht mehr dem Originalbild zugeordnet werden kann. Diese Problematik zählt zu dem im vorherigen Abschnitt definierten *Problem 1*.

Bezeichnung	Beispiel	Beschreibung
Aufnahmezeitpunkt		Die Aufnahme wurde zu einem anderen Zeitpunkt erstellt. Dieser kann in der Vergangenheit oder Zukunft relativ zum Aufnahmezeitpunkt des Originalbildes liegen. Das Beispielbild zeigt eine Aufnahme der Stadt am Abend. Da eine solche Transformation nicht ohne erhöhten Aufwand digital simuliert werden kann, wird sie innerhalb dieser Ausarbeitung nicht weiter betrachtet.
Beschneidung		Ein Ausschnitt des Originalbildes. Das Beispielbild zeigt das Originalbild, von dessen Bildhöhe 40% am unteren Bildrand abgeschnitten wurden.

Bezeichnung	Beispiel	Beschreibung
Bogenförmige Deformation		Eine Krümmung des Originalbildes um einen angegebenen Winkel. Das Beispielbild wurde mit einem Winkel von 10°bogenförmig deformiert.
Vertikale Spiegelung		Eine vertikale Spiegelung des Originalbildes.
Horizontale Spiegelung		Eine horizontale Spiegelung des Originalbildes.
Farbton		Eine Veränderung des Farbtones um einen gegebenen Winkel im Farbrad, auf dem der gesamte Farbkreis abgebildet ist (vgl. [32]). Das Beispielbild entspricht einer Veränderung des Winkels um -90°.
Fassförmige Deformation		Die fassförmige Deformation kann durch die Krümmung der Kameralinse in Photos entstehen. Sie ist die Umkehrung der kissenförmigen Deformation. Das Beispielbild zeigt die nachträgliche Anwendung dieser Transformation.
Gammakorrektur		Eine Tonwertkorrektur mit einer Potenzfunktion der Form $A = E^{\gamma}$ als Korrekturfunktion. Das Beispielbild zeigt eine Gammakorrektur für $\gamma = 0.6$.
Helligkeit		Eine Veränderung des Helligkeitswertes. Das Maß der Änderung wird standardmäßig prozentual angegeben, wobei 100% dem Ursprungswert entspricht. Das Beispielbild zeigt eine Erhöhung des Helligkeitswertes auf 160%.

Bezeichnung	Beispiel	Beschreibung
Kompression		Anwendung eines verlustbehafteten Kompressionsverfahrens. Das Maß der Kompression wird im Allgemeinen prozentual zur originalen Dateigröße angegeben. 100% entsprechen dem Originalbild und somit keiner Kompression. Das Beispielbild zeigt eine JPEG-Kompression auf 5% der ursprünglichen Bildgröße.
Kontrast		Eine Veränderung des Bildkontrastes. Der Kontrast ist ein normierter Intensitätsunterschied zwischen hellen und dunklen Bildstrukturen. Das Beispielbild zeigt das Originalbild mit einem einen erhöhten Kontrastwert.
Kissenförmige Deformation		Die kissenförmige Deformation kann durch die Krümmung der Kameralinse in Photos entstehen. Sie ist die Umkehrung der fassförmigen Deformation. Das Beispielbild zeigt die nachträgliche Anwendung dieser Transformation.
Parallelogramm		Die Transformation der Bildform in ein Parallelogramm. Damit kann beispielsweise eine seitliche Bewegung während der Aufnahme des Bildes simuliert werden. Das Maß der linearen Verschiebung wird als Winkel angegeben. Das Beispielbild zeigt eine Verschiebung um 10°.
Perspektive		Die Aufnahme des Objektes aus einer anderen Perspektive.
Farbsättigung		Eine Veränderung der Farbsättigung. Das Maß der Änderung wird standardmäßig prozentual angegeben, wobei 100% dem Ursprungswert entspricht. Das Beispielbild zeigt eine Erhöhung des Sättigungswertes auf 160%.

Bezeichnung	Beispiel	Beschreibung
Skalierung		Die Änderung der Größe des Bildes. Sie wird als prozentualer Wert angegeben, wobei 100% der Originalgröße entsprechen. Das Beispielbild zeigt eine Skalierung des Originalbildes auf 50%.
Wasserzeichen		Die Platzierung eines Wasserzeichens im Bild.
Weichzeichnung		Anwendung eines Weichzeichnen-Filters, z. B. *Gaußscher Weichzeichner*. Der Grad der Weichzeichnung wird über zwei Werte festgelegt: Radius r und Standardabweichung σ. Bei gleichbleibenden Radius und Erhöhung der Standardabweichung, vergrößert sich die Unschärfe des Bildes. Das Beispielbild zeigt die Anwendung eines Weichzeichnen-Filters mit den Werten $r = 0$ und $\sigma = 10$

Tabelle 3: Auflistung von manuell und digital veränderbaren Eigenschaften anhand derer sich Duplikate unterscheiden können

Tabelle 3 versucht eine möglichst vollständige Übersicht zu geben, durch welche Eigenschaften sich Duplikate unterscheiden können. Einige Unterschiede können sowohl natürlich hervorgerufen werden, z. B. durch erhöhte Lichteinstrahlung während der Aufnahme, als auch digital mit einer Bildbearbeitungssoftware. Das Originalbild bezogen auf die in der Tabelle dargestellten Beispielbilder ist in Abbildung 3(d) gegeben. Die aufgeführten Eigenschaften werden aus Gründen des Umfangs dieser Arbeit nicht detailliert beleuchtet. Charles Poynton hat in seinem *Color FAQ* (`http://poynton.ca/ColorFAQ.html`) eine sehr Umfangreiche Erläuterung zu Helligkeit, Kontrast, Sättigung, etc. gesammelt, welche zum besseren Verständnis herangezogen werden kann. Die technische Umsetzung der Transformationen, wie beispielsweise Deformationen und Beschneidungen, wird beispielhaft im digitalen Handbuch der Software ImageMagick beschrieben, welches unter `http://www.imagemagick.org/Usage/` zu finden ist.

Wird ein Originalbild mit seinen veränderten Versionen verglichen, ist es wünschenswert, dass das errechnete Maß der Ähnlichkeit unverändert bleibt um diese als Duplikat einstufen zu können. Diese gewollte Eigenschaft wird als Invarianz bezeichnet. Liefert beispielsweise eine Metrik denselben Wert beim Vergleich eines Originalbildes mit verschiedenen rotierten Versionen dieses Bildes, ist die Metrik bzw. das angewandte Verfahren rotations-invariant. Eine perfekte

Ähnlichkeitsmetrik ist invariant gegen alle Unterscheidungsmerkmale aus Tabelle 3. In der Realität ist dies jedoch nicht immer der Fall bzw. gehört eher zur Seltenheit. Oftmals vergrößert (bzw. verkleinert) sich der Ähnlichkeitswert analog zur Quantität der Transformation, z. B. der Winkel der Rotation. Ist ein Verfahren nicht invariant gegenüber eines Großteils an Transformationen, hat dies starken negativen Einfluss auf das Ergebnis der Deduplikation und somit auf dessen Effektivität. Diese Problematik kann wie folgt zusammengefasst werden:

> **Problem 3: Transformationen** Ähnlichkeitsmetriken sind nicht invariant gegenüber bestimmter Bildtransformationen.

Es stellt sich die Frage, welches Verfahren oder System gegenüber welcher Transformation invariant ist. Eine entsprechende Evaluation dieser Beziehung könnte als Entscheidungshilfe für oder gegen eine Metrik herangezogen werden. Hat man beispielsweise innerhalb eines Anwendungsfalls Bilder, die in vielen verschiedenen Skalierungen vorliegen, ist eine skalierungs-invariante Metrik zur Deduplikation der Bilder deutlich von Vorteil.

3.3. Ähnlichkeitsmetriken

Das Unternehmen des im vorherigen Abschnitt betrachteten Anwendungsfalls hat eine bestehende Implementierung einer Ähnlichkeitsmetrik zur Deduplikation genutzt. Neben dieser existieren noch weitere Metriken und deren Implementationen, die jedoch im allgemeinen nicht umfassend evaluiert worden sind. Weiterhin findet man in aktuellen Forschungen keine Auflistung und Gegenüberstellung dieser Verfahren, welche als Entscheidungshilfe für eine spezielle Problematik herangezogen werden kann. In anderen Worten kann das Fehlen dieses Vergleichs Schwierigkeiten für Unternehmen bereiten, für einen aufkommenden Anwendungsfall eine geeignete Metrik auszuwählen.

> **Problem 4: Vergleich bestehender Metriken** In aktuellen Forschungspapieren ist keine vollständige Gegenüberstellung aktueller Verfahren und Systeme zur inhaltsbasierten Berechnung der Ähnlichkeit für zwei zu vergleichende Bilder zu finden.

3.4. Komplexität

Eine weitere Problemstellung ist die Skalierbarkeit der Deduplizierung. Da jedes zu untersuchende Bild mit allen anderen Bildern verglichen werden muss, steigt die Anzahl der Vergleichspaare quadratisch mit der Anzahl der zu untersuchenden Bilder. Das Problem besitzt somit eine quadratische Komplexität. Sollen beispielsweise eine Millionen Bilder nach Duplikaten untersucht werden, resultieren daraus rund 500 Milliarden Vergleichspaare. Folgendes Problem kann somit definiert werden:

> **Problem 5: Quadratische Komplexität** Die Anzahl der Vergleichspaare steigt quadratisch mit der Anzahl der zu untersuchenden Bilder.

Um eine solche Menge an Berechnungen zu bewältigen, müssen Strategien zur Reduktion der Vergleichspaare angewendet werden, sowie die Berechnung der Ähnlichkeit verteilt auf mehreren Systemen erfolgen. Das Problem besteht nicht nur im Kontext bildbasierten Deduplikation, sondern allgemein bei der Deduplikation von Entitäten aller Art.

4. Stand der Forschung und Technik

In den letzten Jahren sind viele Publikationen veröffentlicht und Technologien vorgestellt worden, die sich mit der Ermittlung von semantisch äquivalenten Entitäten, darunter auch Bilder, beschäftigen. Im Abschnitt 4.1 werden bestehende Deduplikationssysteme aufgeführt und deren fehlende Betrachtung von Bild-Entitäten verdeutlicht. In den anschließenden drei Abschnitten (4.2 - 4.4) werden derzeit bestehende Tools und Verfahren zur Ermittlung der Ähnlichkeit zweier Bilder vorgestellt. Dabei können diese in drei Bereiche unterteilt werden, welche in Abbildung 4 zusammenfassend dargestellt sind. Im Abschnitt 4.6 befindet sich eine Übersicht der für diese Arbeit ausgewählten Verfahren und deren Implementationen, die in das später vorgestellte Deduplikationssystem *SiMaSu* integriert werden. Für diese werden im Kapitel 7 drei Evaluationen durchgeführt. Veröffentlichungen, die sich mit dem Vergleich dieser ausgewählten Verfahren beschäftigen, werden im letzten Abschnitt aufgeführt.

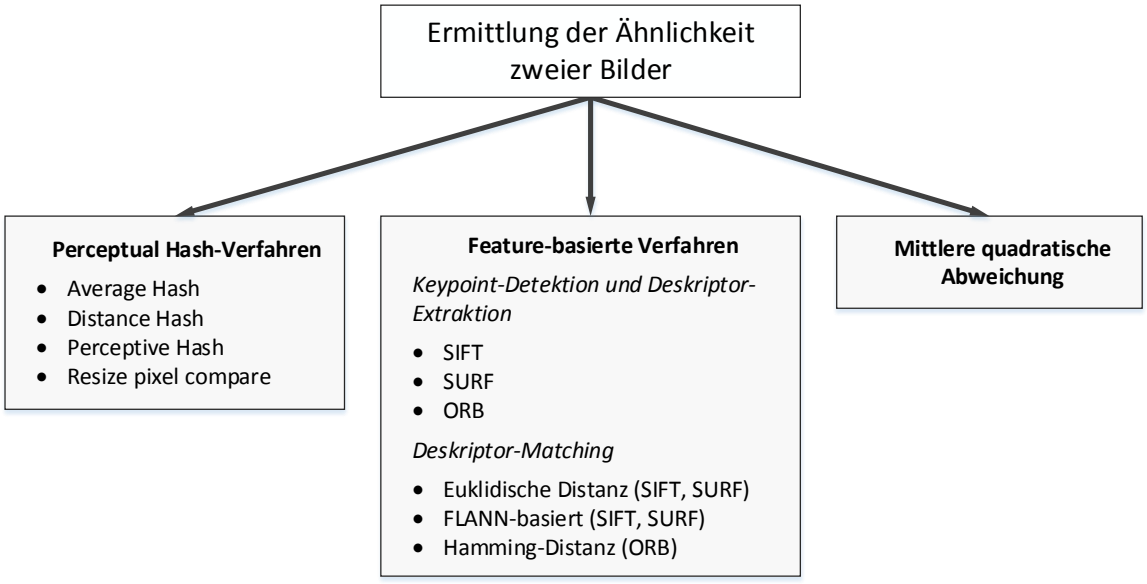


Abbildung 4: Überblick der Verfahren zur Bildähnlichkeitsermittlung

4.1. Deduplikationssysteme

Deduplikation ist ein Prozess zur Identifikation von Entitäten, die das selbe individuelle Realweltobjekt abbilden. In diesem Bereich wird schon seit vielen Jahren geforscht und es entstanden verschiedene Systeme die diesen Prozess implementieren, sowie Publikationen, welche sich mit den einzelnen Schritten dieses Prozesses beschäftigen. Deduplikationssysteme, wie FAMER [53], Dedoop [35, 36] und SERF [9], um nur einige zu nennen, ermöglichen die Kombination von verschiedenen Blocking- und Matching-Ansätzen auf zum Teil verteilten Umgebungen zur effektiven Durchführung der Matching-Schritte. Dabei wird versucht, Anforderungen wie Effektivität, Effizienz, Generalisierbarkeit und geringen manuellen Aufwand zu adressieren.

Bei den aktuell bestehenden Publikationen zu Deduplikationssystemen werden jedoch ausschließlich textuelle Repräsentationen von Entitäten und deren Beziehungen zueinander betrachtet. Eine Deduplikation von Entitäten, die als zweidimensionale Bilder repräsentiert sind, wird in der kommerziellen Software *TinEye* [58] von dem gleichnamigen Unternehmen aus Canada angeboten. Die darin enthaltene Komponente *MatchEngine* verspricht eine schnelle, skalierte und genaue Identifikation von Duplikaten aus einer großen Menge an Bildern. Die Komponente soll invariant gegen Transformationen, wie Skalierung, Cropping, Farbänderungen und Platzierungen von Wasserzeichen sein. Zur Funktionsweise wird ausschließlich die Erzeugung von Fingerprints erwähnt, aus deren Vergleich Duplikate erkannt werden können. Das System läuft auf internen Servern des Unternehmens und kann über eine REST-konforme API angesprochen werden. Da diese Software Lizenz-pflichtig ist und auch nach persönlicher Kontaktaufnahme nicht kostenfrei zu universitären Evaluationszwecken zur Verfügung steht, wird sie in diesem Umfang nicht weiter betrachtet.

Hauptbestandteil einer Deduplizierung sind Technologien, die eine Berechnung der Ähnlichkeit zweier Entitäten ermöglichen (vgl. Kapitel 2). An das in dieser Arbeit konzipierte skalierbare Deduplikationssystem *SIMaSu*, welches ausführlich in Kapitel 5 erläutert wird, sollen verschiedene Ansätze zur Ermittlung der semantischen Ähnlichkeit zweier Bilder angebunden werden. Die Gesamtheit derzeitig existierender Verfahren kann in drei Kategorien unterteilt werden: *Perceptual Hash-Verfahren*, *Feature-basierte Verfahren* und *Mittlere quadratische Abweichung*. Die erste Kategorie verwendet Ansätze, die globale Merkmale des gesamten Bildes extrahieren und miteinander vergleichen, um so eine Übereinstimmung zu ermitteln. Die Technologien der zweiten Kategorie extrahieren lokale Merkmale der Bilder und vergleichen diese zum selben Zweck miteinander. In der letzten Kategorie wird der mittlere quadratische Fehler der Pixel-Intensitätswerte beider Bilder errechnet. In den nachfolgenden Abschnitten werden die einzelnen Ansätze der Kategorien näher betrachtet.

4.2. Perceptual Hash-Verfahren

Eine Hash-Funktion $h(x)$ dient zur Abbildung von binären Daten x beliebiger Größe auf einen Wert mit fester Größe, welcher als *Hash-Wert* oder *Hash* bezeichnet wird. Perceptual Hash-Verfahren besitzen die Eigenschaft, dass aus geringen Änderungen des zugrunde liegenden Bildes ebenfalls eine geringe bis keine Änderung des Hashes resultiert. Ein solcher Hash wird in der Literatur auch als *Perceptual-Hash* oder *Fingerprint* bezeichnet (vgl. [63]). *Kollisionen*[3] sind dabei erwünscht. Diese Eigenschaft wird auch als *Kontinuität* bezeichnet (vgl. [54]) und hebt das Perceptual Hash-Verfahren von kryptografischen Hash-Verfahren ab. Bei letzteren steht die Kollisionsfreiheit im Vordergrund.

Nach der Erzeugung eines Hashes bzw. Fingerprints je Bild x und y, wird deren Distanz mittels einer Distanzfunktion $d(h(x), h(y))$ ermittelt. Eine oft verwendete ist hier die Hamming-Distanz, welche eine Aussage über die Anzahl der voneinander verschiedenen Bits der Hash-Werte trifft. Je geringer die Distanz, desto ähnlicher die Bilder. Soll eine normierte Metrik im Intervall $[0, 1]$ verwendet werden, kann die Distanz durch die Größe des Hashes dividiert werden.

[3] Als *Kollision* werden identische Hash-Werte bezeichnet, die aus unterschiedlichen Datensätzen entstanden sind.

4.2.1. Average Hash

Der sogenannte *Average Hash*-Ansatz (auch *aHash* oder *Mean Hash*) transformiert den Farbbereich des Eingabebildes in Graustufen und skaliert es auf eine Größe von 8x8 Pixel (px). Jeder Pixel entspricht einem Bit im somit 64 stelligen Hash-Wert. Ein Bit wird auf `1` gesetzt, wenn der Farbwert des entsprechenden Pixels größer als der Mittelwert aller Pixel ist. Zur Berechnung der Distanz beider entstehenden Vektoren kann beispielsweise die Hamming-Distanzfunktion genutzt werden.

Dieser Algorithmus ist sehr einfach zu implementieren und weist eine kurze Laufzeit auf. Implementiert wird er in den Bibliotheken ImageHash [26] von J. Buchner, Image::Hash [27] und ImageHash [28] von J. Segers. In dieser Ausarbeitung wird ausschließlich die zuletzt genannte Implementierung zur Evaluation verwendet.

4.2.2. Distance Hash

Analog zum *Average Hash* wird bei dem *Distance Hash*-Ansatz (auch *dHash* oder *Dynamic Hash*) das Eingabebild in Graustufen transformiert und auf eine Größe von 9x8 px skaliert. Nun wird jeder Pixel einer Zeile mit dem rechten Nachbar-Pixel verglichen. Ist der Farbwert des linken Pixels $P[x]$ kleiner als der Farbwert des rechten Pixels $P[x+1]$, wird ein entsprechendes Bit im Hash auf `1` gesetzt, andernfalls auf `0`. Diese Nachbarschaftsbeziehung betrachtet somit den Farbverlauf. Da pro 9 Pixel großer Zeile 8 Übergänge existieren und das skalierte Bild 8 Zeilen aufweist, entsteht somit ein 64 Bit großer Hash.

Dieses Verfahren kann ebenso ohne Schwierigkeiten implementiert werden und die Berechnung ist vergleichsweise schnell. Die Bibliotheken ImageHash [26] von J. Buchner, Image::Hash [27], Grafika [22] von Kosinix und ImageHash [28] von J. Segers implementieren dieses Verfahren. Für die in dieser Arbeit durchgeführten Evaluationen wird ebenfalls ausschließlich die zuletzt genannte Implementierung verwendet.

4.2.3. Perceptive Hash

Mit *Perceptive Hash* (auch *pHash*, *Perception Hash* oder *Perceptual Hash*) wird das Eingabebild analog zu den anderen beiden obigen Verfahren in Graustufen transformiert und auf 32x32 px skaliert. Auf dieser Matrix mit Helligkeitswerten wird eine diskrete Kosinustransformation (DCT) angewendet, um die Frequenzen des Bildes zu extrahieren. Aus der resultierenden Matrix werden ausschließlich die 8x8 px der linken oberen Ecke betrachtet, welche die tiefen Frequenzen des Bildes beinhalten. Der Hash wird nun berechnet, indem jeder Pixel-Wert mit dem Mittelwert aus allen 64 Pixeln verglichen wird. Ist der Pixel-Wert größer als der Mittelwert, wird das entsprechende Bit im Hash auf `1` gesetzt, andernfalls auf `0`. (Vgl. [63])

Implementiert ist dieses Verfahren in der Anwendung *compare* der *ImageMagick*-Plattform, sowie in den Bibliotheken *ImageHash* [26] von J. Buchner, *Image::Hash* [27] und *ImageHash* [28] von J. Segers. Zur Evaluation wird das Tool *compare* und die Bibliothek von J.Segers herangezogen.

4.2.4. Resize Pixel Compare

Das letzte hier vorgestellte Hash-Verfahren ist eine Entwicklung eines in Deutschland ansässigen Vergleichsportals und unterliegt dessen Lizenzbestimmungen. Im Gegensatz zu den anderen drei vorgestellten Methoden werden bei diesem Algorithmus Farbinformationen in die Bildung des Hashes einbezogen. Das Bild wird zunächst auf 20x20 px skaliert. Für jeden Farbkanal eines Pixels wird nun die Differenz des Farbwertes mit den Farbwert des Pixels an der selben Position des zu vergleichenden Bildes ermittelt. Ist die Differenz pro Farbkanal für alle Kanäle kleiner als ein vorab festgelegter Grenzwert, wird dieser Pixel als *Treffer* gezählt. Die Metrik ist das Verhältnis der *Treffer* zur Gesamtzahl der Pixel, in diesem Fall 400. Es entsteht ein Ähnlichkeitswert im Intervall $[0, 1]$. Die Werte der Farbkanäle aller Pixel können in diesem Fall als Fingerprint angesehen werden. Die Distanzfunktion ist die beschriebene Berechnung der Differenzen.

4.3. Feature-basierte Verfahren

Eine weitere Klasse von Verfahren zur Ermittlung von Bildähnlichkeiten sind Feature-basierte Algorithmen. Im Allgemeinen besteht ein solcher Algorithmus aus zwei grundlegenden Schritten: die *Keypoint-Detektion* sowie die im Anschluss folgende *Deskriptor-Extraktion*. Bei der *Keypoint-Detektion* werden lokale Operationen auf dem Eingabebild und ggf. skalierten Versionen dieses Bildes angewandt. Die daraus entstandenen Informationen über strukturelle Eigenschaften des Bildes dienen zur Detektion interessanter Bereiche, deren Zentren als *Keypoints* bezeichnet werden. Diese *Keypoints* erfordern im Anschluss eine möglichst eindeutige Beschreibung, um in einen nachfolgenden Bild wiedergefunden zu werden. Diese Beschreibung in Form eines Vektors wird als *Deskriptor* bezeichnet. Ein *Feature* ist somit die Kombination aus *Keypoint* und zugehörigem *Deskriptor*. Der Prozess zur Berechnung dieses *Deskriptors* nennt sich *Deskriptor-Extraktion* und bestimmt je nach Funktionsweise die Art und Länge des Vektors.

Wird diese Technologie auf zwei Bilder angewandt, können die jeweiligen Deskriptoren über eine Distanzfunktion miteinander verglichen werden. Die Features mit der geringsten Distanz werden zunächst als Treffer angenommen. Dieser Vorgang wird als *Deskriptor-Matching* bezeichnet. Um *False Positives* aus den Treffern zu entfernen, können Filtermechanismen angewandt werden. Der *Ratio-Test* [39] von Lowe, das *Bidirectional-Matching* [55], die RANSAC-Filterung [19] und die Ermittlung der Homografie (vgl. [18]) sind entsprechende Technologien, um nur einige zu nennen. Über die gefilterte Menge der Matches kann im Anschluss eine Aussage über die Ähnlichkeit des Inhaltes beider Bilder getroffen werden.

Nachfolgend werden drei populäre Algorithmen inklusive einer kurzen Erläuterung vorgestellt. Weiterreichende Details zu den einzelnen Algorithmen können den angegebenen Primärquellen entnommen werden. Chien et al. haben in [12] einen zusammenfassenden Vergleich herausgearbeitet, welcher für die hier betrachteten Methoden der Tabelle 4 entnommen werden kann.

Der klassische Anwendungsfall für diese Feature-basierten Verfahren ist die Erkennung eines semantisch äquivalenten Objektes aus zwei Abbildungen (vgl. [10]). Aufgrund dieser Eigenschaften können sie auch allgemein zur Ermittlung der Ähnlichkeit zweier Bilder verwendet werden. Hua et al. haben beispielsweise in [24] den SIFT-Algoritmus von Lowe zur Berechnung der Ähnlichkeit für verschieden skalierte Versionen eines Bildes verwendet. Dafür haben sie zwei einfache Metriken vorgestellt, die auf Grundlage der berechneten *Features* einen Ähnlichkeitswert in einem festgelegten Wertebereich liefern. Innerhalb dieser Arbeit wird ein weiterer Ansatz

	SIFT	SURF	ORB
Jahr und Publikation	1999 [40]	2006 [8]	2011 [52]
Invariant gegen Skalierung	Ja	Ja	Ja
Invariant gegen Rotation	Ja	Ja	Ja
Länge des Deskriptors	128-dimensional	64-dimensional	256 Bit

Tabelle 4: Vergleich von Image-Features (vgl. [12])

einer Feature-basierten Ähnlichkeitsmetrik vorgestellt, welche neben der SIFT-Technologie auch für SURF und ORB verwendet werden kann. Sie orientiert sich dabei an den von Piotrowski in [46] definierten Eigenschaften einer idealen Metrik.

4.3.1. SIFT

Eine der ersten umfassenden Arbeiten im Bereich der Feature-Detektion wurde unter dem Titel *Object recognition from local scale-invariant features* [40] von David Lowe im Jahre 1999 veröffentlicht. Darin wird ausführlich der SIFT-Algorithmus erläutert. Vom selben Autor erschien 2004 eine darauf aufbauende Publikation unter dem Titel *Distinctive image features from scale-invariant keypoints* [39]. Nachfolgend werden die beiden Kern-Funktionalitäten der Technologie kurz vorgestellt.

Keypoint-Detektion Der SIFT-Algorithmus detektiert rotations- und skaleninvariante Keypoints durch unterschiedlich starke Skalierung des Eingabebildes und dessen Faltung mit einer Difference of Gaussians (DoG)-Funktion[4]. Ein SIFT-Keypoint ist charakterisiert durch die Position im Bild und dessen Ausrichtung bzw. Orientierung.

Deskriptor-Extraktion Zu einem Keypoint wird ein 128-dimensionaler reellwertiger Deskriptor-Vektor zu dessen eindeutigen Beschreibung berechnet. Dabei werden Nachbarschaftsbeziehungen der Pixel ermittelt, indem in einem Fenster von 16x16 px um den Keypoint Helligkeitsverläufe zur Berechnung herangezogen werden.

Laut Lowe enthält ein typisches Bild 2.000 oder mehr Features, welche aus den verschiedenen abgebildeten Objekten und dem Hintergrund des Bildes resultieren. Seien A und B die zu vergleichenden Bilder, müssen zunächst die SIFT-Deskriptoren für beide berechnet werden. Die Mehrdimensionalität des Deskriptors hat einen zeitaufwändigen Matching-Prozess zur Folge. Die Distanzen zwischen den Deskriptoren von A und B kann man beispielsweise über die euklidische Distanzfunktion ermitteln (in der Bibliothek OpenCV als *Bruteforce* bezeichnet) oder die von Lowe und Muja vorgestellte Sammlung an Algorithmen zur schnellen Nachbarschafts-Suche *Fast Approximate Nearest Neighbor Search (FLANN)* [42] verwenden. Für jeden Deskriptor in A wird die geringste Distanz zu einem Deskriptor in B als Treffer angesehen.

[4] Der sogenannte Difference of Gaussians (DoG)-Filter (dt. *Differenz der Mittelwerte*) dient zur Extraktion von Kanten aus einem Originalbild unter Anwendung des Gaußschen Weichzeichners in zwei unterschiedlichen Stärken mit anschließender Berechnung deren Differenz (vgl. [16]).

Da diese Menge an Treffern im Allgemeinen von einigen ungenauen Merkmalen ausgeht, empfiehlt sich die Anwendung des von Lowe beschriebenen *Ratio-Test* [39] zur Aussortierung von uneindeutigen Treffern. Das Prinzip des Ratio-Tests ist die Gegenüberstellung der beiden geringsten, von einem Deskriptor aus A ausgehenden Distanzen d_1 und d_2. Sind sich die Distanzen sehr ähnlich, wird angenommen, dass keine eindeutige Zuordnung besteht. Ergebnisse von Lowe zeigen, dass 90% der falschen Treffer herausgefiltert werden können, wenn man alle Deskriptoren verwirft, deren Verhältnis $\frac{d_1}{d_2}$ größer oder gleich 0.8 beträgt. (Vgl. [12, 39])

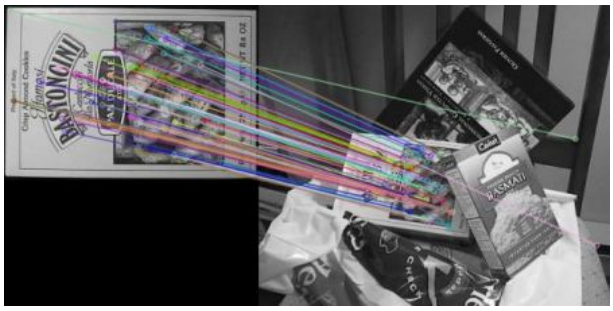


Abbildung 5: Beispielhafte Visualisierung eines Matchings mit SIFT-Desktiptoren [17]

Abbildung 5 zeigt dabei eine beispielhafte Visualisierung des Deskriptor-Matchings nach einem erfolgten Ratio-Test. Features des Buches auf dem linken Bild wurden bei dem korrekten Exemplar im rechten Bild gefunden. Zwei als Treffer identifizierte Features werden dabei durch eine farbige Linie gekennzeichnet. Diese Visualisierung ist mit der OpenCV-Bibliothek entstanden.

Da der SIFT-Algorithmus von der University of British Columbia (US) patentiert wurde, fallen Lizenzgebühren bei der kommerziellen Verwendung dieses Verfahrens an. Innerhalb der Literatur können Erweiterungen und Adaptionen von SIFT gefunden werden, wie beispielsweise *Very Fast SIFT Feature Matching (VF-SIFT)* [5] und *Affine-SIFT (ASIFT)* [41]. Jian Wu et al. haben die verschiedenen SIFT-Varianten innerhalb ihrer 2013 erschienenen Publikation *A Comparative Study of SIFT and its Variants* [62] untersucht und miteinander verglichen. Innerhalb dieser Arbeit wird ausschließlich die populärste Erweiterung namens *SURF* (siehe Abschnitt 4.3.2) betrachtet.

4.3.2. SURF

Herbert Bay et al. veröffentlichten 2006 einen als SURF [8] bekannten Algorithmus, welcher einen zu SIFT alternativen Ansatz zur Detektion von Features verfolgt und auf eine verbesserte Performanz ausgelegt ist. Er wird im Allgemeinen als schneller und stabiler als SIFT beschrieben. An Stelle der bei SIFT benutzten Gauß-Filter, werden bei SURF Mittelwertfilter verwendet. Diese können mit konstantem Zeitaufwand durch Hilfe von Integralbildern [59] berechnet werden, was einen Vorteil hinsichtlich der Laufzeit gegenüber SIFT bietet. Die Ausrichtung eines Merkmals und somit die Rotationsinvarianz wird durch Analysen der Farbverläufe im Umkreis eines Keypoints ermittelt. Zur Beschleunigung des Verfahrens, kann die Ermittlung der Orientierung optional deaktiviert werden. Der durch die Anwendung von SURF resultierende Deskriptor-Vektor ist im Vergleich zu SIFT nur 64-dimensional, wodurch ebenfalls die Distanzberechnung zur Ermittlung von Treffern beschleunigt wird. (Vgl. [8, 12])

4.3.3. ORB

Ein weiterer Ansatz zur Feature-Detektion und Deskriptor-Extraktion ist der 2011 von Ethan Rublee et al. entwickelte ORB-Algorithmus [52]. Die Abkürzung ORB bedeutet dabei *Oriented FAST and rotated BRIEF*. Wie der Name beschreibt, nutzt ORB die Technologie *Features from accelerated segment test (FAST)* [51] zur Detektion der Features und eine modifizierte Version des *Binary Robust Independent Elementary Features (BRIEF)*-Verfahrens [11] zur Berechnung der zugehörigen Deskriptoren.

Keypoint-Detektion Der FAST-Algorithmus prüft jeden Pixel im Bild, ob dieser einen Keypoint darstellt. Dazu wird ein Kreis um den zu untersuchende Pixel gespannt, welcher einen Umfang von 16 px aufweist. Diese 16 px werden anhand ihrer Intensitäten untersucht. Erfüllen sie bestimmte Kriterien, wird der zu untersuchenden Pixel als Keypoint definiert. Damit eine Invarianz gegen Skalierung erreicht wird, die der FAST-Ansatz standardmäßig nicht bietet, wird der FAST-Algorithmus mehrfach auf verschiedene Skalierungen des Eingabebildes angewendet, um skaleninvariante Features zu bestimmen. Da FAST auch keine Rotationsinvarianz besitzt, wird zusätzlich ein Verfahren namens *Intensity Centroid* verwendet, welches ein Intensitätszentrum der näheren Umgebung des Keypoints ermittelt. Ein vom Keypoint zum Intensitätszentrum gerichteter Vektor bestimmt die Orientierung des Keypoints. (Vgl. [12, 52])

Deskriptor-Extraktion Der BRIEF-Algorithmus betrachtet anhand eines vordefinierten Pixel-Musters Pixel-Paare in der nahen Umgebung des Keypoints, welche zur Erzeugung des Deskriptors verwendet werden. Da BRIEF standardmäßig keine Orientierung berücksichtigt, wird die bei der Keypoint-Detektion festgelegte Orientierung verwendet und das Muster dementsprechend rotiert. Es resultiert ein 256 Bit großer Bitvektor, wodurch die Hamming-Distanz zur Berechnung des Abstandes zweier Deskriptoren verwendet werden kann. (Vgl. [12, 52])

4.4. Mittlere quadratische Abweichung

Ein weiteres Verfahren zur Berechnung der Ähnlichkeit zwischen zwei Bildern ist die Ermittlung der mittleren quadratischen Abweichung (engl. *Mean Square Error (MSE)*) der Pixel-Intensitäten beider Bilder. Die mittlere quadratische Abweichung zweier Bilder g und h berechnet sich nach Gleichung 4, wobei $g(x, y)$ die Intensität des Pixels der Position (x, y) ist. Eine mittlere quadratische Abweichung von 0 bedeutet die absolute Gleichheit beider Bilder. Ein Wert größer als 0 kennzeichnet die Verschiedenheit der Bilder.

$$e_{MSE} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} [h(n, m) - g(n, m)]^2 \tag{4}$$

Man muss jedoch beachten, dass ein hoher Wert nicht zwingend Rückschlüsse auf große Unterschiede der Bildinhalte schließen lässt. Werden beispielsweise ausschließlich zwei identische Bilder mit unterschiedlichen Kontrastwerten miteinander verglichen, resultiert daraus ein sehr großer Wert für die Abweichung. Weiterhin können auf diese Weise nur Bilder mit gleichen Dimensionen verglichen werden. Die Software ImageMagick hat dieses Verfahren implementiert. Innerhalb dieser Arbeit wird die auf der *ImageMagick*-API aufbauende PHP-Erweiterung *Imagick* verwendet, über dessen Funktion `compareImages()` die mittleren quadratischen Abweichung

berechnet werden kann. Bei dem Anwendungsfall der Problemanalyse aus Kapitel 3 wurde diese Implementierung als Ähnlichkeitsmetrik verwendet.

4.5. Evaluationen

In dem 2017 erschienenen Paper *Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images* [33] evaluieren Karami et al. die Robustheit der Algorithmen SIFT, SURF und ORB gegenüber sechs Arten von Transformationen eines Originalbildes. Dazu zählen u.a. sechs verschiedene Rotationen, sowie jeweils eine Änderung der Skalierung, der Farbintensität, etc. Es wurde für jede der drei Technologien die Anzahl der gefundenen Features jeweils für das Originalbild und das deformierte Bild ermittelt. Weiterhin wurden auch die gefundenen Matches beider Treffer angegeben, jedoch ohne Beschreibung der verwendeten Matching-Strategie bzw. der angewandten Filter, um falsche Treffer zu entfernen. Die Evaluation wurde ausschließlich anhand eines einzelnen Eingabebildes durchgeführt. Als Ergebnis wurde ORB als schnellster und SIFT als robustester Algorithmus ermittelt.

Der Autor Fred Weinhaus hat in der Onlinequelle *Tests Of Perceptual Hash (PHASH) Compare Metric* [60] das Ergebnis seiner Evaluation der pHash-Metrik vorgestellt. Er hat sich dabei auf die Anwendung *compare* der *ImageMagick*-Plattform als Implementation dieser Metrik beschränkt. In dem Experiment wurde die Robustheit des Algorithmus gegenüber 16 verschiedenen Bildtransformationen, jeweils mit unterschiedlichen Intensitäten, evaluiert. Zu den Transformationen zählen neben Helligkeits- und Kontrastanpassungen auch das Platzieren eines Wasserzeichens. Als Bilddatensatz wurden acht gleichgroße Bilder aus der *USC SIPI open source image library* verwendet. Das Ergebnis des Experiments zeigt, dass die Implementation der Metrik u.a. invariant gegenüber Skalierungen und Spiegelungen ist. Der Wert 21 wurde als optimaler Grenzwert für eine Klassifikation ermittelt. Ebenfalls wurde diese Metrik mit verschiedensten Abwandlungen in der Masterarbeit *Implementation and Benchmarking of Perceptual Image Hash Functions* [63] von Christoph Zauner untersucht.

Neben den genannten Evaluationen existieren noch weitere, die sich mit Untersuchung der Effektivität verschiedener Ähnlichkeitsmetriken beschäftigen, sowie die Laufzeit und Invarianz gegenüber der Anwendung von Bildmanipulationen untersuchen. Dabei fällt auf, dass ein fairer Vergleich aller hier aufgeführten Technologien untereinander nicht existiert. Die Fairness ist dabei charakterisiert durch die Verwendung *eines* umfangreichen Bilddatensatzes zur Evaluation aller Verfahren innerhalb einer identischen Testumgebung. Weiterhin wird von den innerhalb der Problembeschreibung ermittelten 20 Bildtransformationen (vgl. Abschnitt 3.2), oftmals nur ein geringer Teil für Experimente herangezogen. Zudem fehlt auch in vielen Fällen die Untersuchung der Auswirkung verschiedener Intensitäten einer Transformation.

4.6. Übersicht

In den vorherigen Abschnitten wurden verschiedene Technologien und Metriken als Grundlage für die Berechnung einer semantischen Ähnlichkeit zweier Bilder erläutert. Ein Überblick dieser Technologien ist Tabelle 5 zu entnehmen. Neben dem Typ wird zu jedem Verfahren eine Bibliothek bzw. Anwendung angegeben, die eine Implementierung dieser Technologie beinhaltet oder sie ermöglicht. Die Kernfunktionen der Feature-basierten Verfahren SIFT, SURF und ORB

sind beispielsweise innerhalb der Bibliothek *OpenCV* implementiert. Die Abbildung übereinstimmender Features zweier Bilder auf einen Ähnlichkeitswert ist jedoch nicht in der Bibliothek enthalten. Daher wird im Abschnitt 5.3 eine entsprechende Metrik entworfen und prototypisch als Java-Anwendung implementiert (siehe Kapitel 6).

Technologie	Typ	Implementation
pHash	Perceptual Hash	ImageMagick [29] ImageHash [28]
aHash	Perceptual Hash	ImageHash [28]
dHash	Perceptual Hash	ImageHash [28]
Resize Pixel Compare	Perceptual Hash	Imagick [25]
SIFT	Feature-basiert	OpenCV [43]
SURF	Feature-basiert	OpenCV [43]
ORB	Feature-basiert	OpenCV [43]
MSE	Mittlere quadratische Abweichung	Imagick [25]

Tabelle 5: Überblick der Technologien mit ausgewählten Implementierungen

5. SIMaSu - Similar Image Matching Suite

In diesem Kapitel wird die **S**imilar **I**mage **Ma**tching **Su**ite **SIMaSu** vorgestellt, eine skalierbare und erweiterbare Anwendung zur Durchführung einer bildbasierten Deduplikation großer Mengen von Bildern. Die Hauptziele dieses Systems sind zum einen, die im Kapitel 4 vorgestellten Technologien der Bildähnlichkeitsberechnung zur Deduplikation zu nutzen, sowie zum anderen, die Effektivität der Methoden bei sich verändernden Grenzwerten aufzuzeigen und somit eine Entscheidungshilfe für die Wahl des Grenzwertes je Methode zu geben. Zudem wurde eine Ähnlichkeitsmetrik integriert, welche die Feature-basierten Verfahren SIFT, SURF und ORB implementiert. Neben der Skalierbarkeit, ist auch die unkomplizierte Anbindung weiterer Ähnlichkeitsmetriken, die innerhalb dieser Thesis nicht betrachtet wurden, ein wesentlicher Vorteil des Systems. Mit SIMaSu ist außerdem ein Evaluationssystem entstanden, welches einen fairen Vergleich der angebundenen Ähnlichkeitsmetriken bietet.

Im Abschnitt 5.1 werden zunächst alle Anforderungen an die Anwendung definiert. Im Anschluss wird das Architekturkonzept im Abschnitt 5.2 erläutert. Abschließend wird im Abschnitt 5.3 ausführlich die Konzeption einer Feature-basierten Ähnlichkeitsmetrik vorgestellt, welche einen wesentlichen Bestandteil von SIMaSu darstellt.

5.1. Anforderungen

Der nachfolgenden Auflistung sind alle Anforderungen an das zu entwickelnde System zu entnehmen. Diese beziehen sich auf die in der Zielsetzung genannten Features.

Eingabedaten Eine beliebige Anzahl von Bildern kann als Eingabedaten verwendet werden. Bilder werden repräsentiert durch einen absoluten Pfad oder eine URL.

Metriken Die im Abschnitt 4.6 vorgestellten Verfahren stehen zur Berechnung der Ähnlichkeit zweier Bilder zur Verfügung.

Feature-basierte Metrik Aus den Technologien SIFT, SURF und ORB soll eine Ähnlichkeitsmetrik konzipiert werden, welche im System verwendet werden kann. Sie ist weiterhin als eigenständige Implementierung verwendbar und erfüllt zudem die Eigenschaften der Reflexivität und Symmetrie.

Konfigurierbarkeit Es besteht die Möglichkeit, die für den Vergleich der Bilder zu verwendenden Verfahren auszuwählen.

Erweiterbarkeit Die Erweiterung der bestehenden Metriken um weitere Verfahren ist ohne großen Aufwand möglich.

Skalierbarkeit Die Berechnung der Metriken kann auf beliebig vielen Systemen verteilt erfolgen.

Ausgabe Ähnlichkeitsmatrix Es wird als Ausgabe pro Verfahren einer Ähnlichkeitsmatrix in einem geeigneten Format (z. B. innerhalb einer relationalen Datenbank) erzeugt.

Ausgabe Duplikatgruppen Es werden alle ermittelten Bildduplikate gruppiert in einem geeigneten Format ausgegeben.

Effektivitätsanalyse Die Effektivität (siehe Kapitel 2) einer jeden ausgewählten Metrik kann nach vorheriger Definition der erwarteten Duplikatgruppen als Precision, Recall und F-Measure für verschiedene Grenzwerte ausgegeben werden.

5.2. Architekturkonzept

Die Architekturskizze für die Kernfunktionalitäten des SIMaSu-Systems ist in Abbildung 6 dargestellt. Sie zeigt den Datenfluss von den Eingabe-Entitäten bis hin zu den gruppierten Duplikaten und dient als Grundlage der im Kapitel 6 erläuterten prototypischen Implementierung. Nachfolgend werden der Datenfluss sowie die eingezeichneten Komponenten näher erläutert.

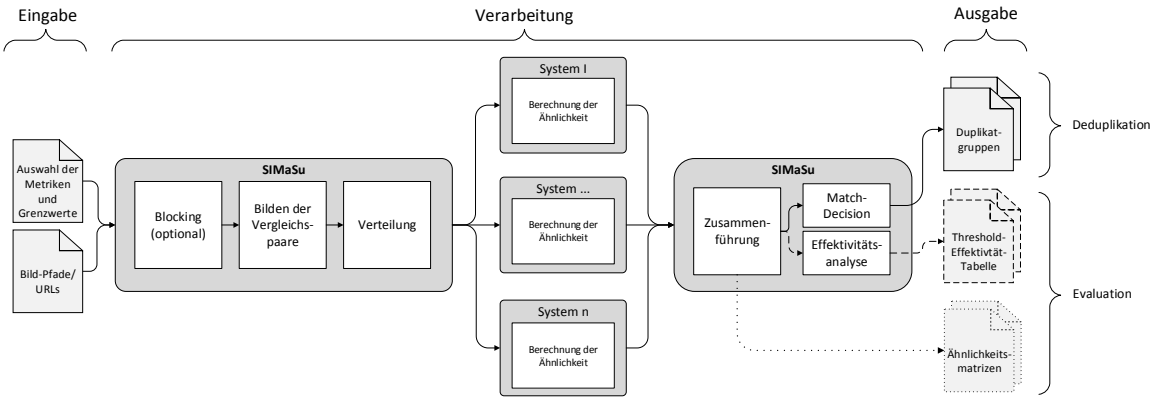


Abbildung 6: Systemkonzept des SIMaSu-Systems

Die Bilder, für die eine Deduplizierung oder Evaluation erfolgen soll, werden als Liste von Pfad- bzw. URL-Angaben der Anwendung übergeben. Soll eine Evaluation der Effektivität einzelner Metriken erfolgen, muss die Liste manuell nach Duplikaten gruppiert worden sein, damit der *Goldstandard* (siehe Kapitel 2) und somit die Grundlage der Analyse definiert wird. Weiterhin müssen die zu verwendenden Metriken inklusive eines Grenzwertes für die später stattfindende *Match-Decision* (siehe unten) angegeben werden. Soll ausschließlich die Effektivität der Metriken evaluiert werden, ist die Angabe der Grenzwerte nicht nötig. Im nächsten Schritt erfolgt die Anwendung einer Blocking-Technologie, um Bilder mit einer hohen Trefferwahrscheinlichkeit zu einem Block zuzuordnen. Dieser Schritt ist optional und dient ausschließlich der Verbesserung der Performance durch Reduktion der Vergleichspaare. Zuletzt genannte werden im nächsten Schritt durch Bildung aller Kombinationen der Bilder erstellt. Befinden sich in einem Block die Menge der Bilder $A = \{a_1, a_2, ..., a_m\}$, ist es neben dem Blocking-Ansatz notwendig, die Anzahl der Vergleichspaare so gering wie möglich zu halten. Da davon ausgegangen wird, dass Ähnlichkeitsmetriken im Allgemeinen reflexive und symmetrische Eigenschaften aufweisen, werden Bildpaare $\{a_i, a_i\}$, sowie $\{a_k, a_i\}$ verworfen, wenn $\{a_i, a_k\}$ schon als Vergleichspaar aufgenommen wurde. Wie bereits im Anwendungsfall der Problemanalyse erläutert wurde, reduziert sich damit die Anzahl der Vergleiche bei m Bildern von m^2 auf $\frac{m^2-m}{2}$.

Nachdem alle Bildpaare definiert wurden, werden diese auf eine verteilte Infrastruktur übertragen, um parallel berechnet zu werden. Die Aufgabe der Berechnung eines Ähnlichkeitsmaßes mit einer Metrik für ein Bildpaar wird dabei zur Vereinfachung als *Prozess* bezeichnet. Wurden bei der Eingabe mehrere Metriken definiert, muss für jedes Bildpaar mit jeder Metrik die Ähnlichkeit berechnet werden. Die Anzahl der zu verteilenden Prozesse errechnet sich dabei aus dem Produkt der Anzahl der Vergleichspaare mit der Anzahl der definierten Metriken. Da die Anzahl der Systeme mit hoher Wahrscheinlichkeit kleiner ist als die Anzahl der Prozesse, speichert die Komponente *Verteilung* alle Prozesse in einer Warteschlange bis diese verteilt wurden. Die Systeme, auf denen die Prozesse verteilt werden, sind in der Abbildung als *System I*, *System II*, etc. bezeichnet. Pro System können auch mehrere Berechnungen parallel ausgeführt werden. Wurde eine Berechnung fertiggestellt, wird dies der *Verteilung* bestätigt und das Ergebnis an die *Zusammenführung* übermittelt. Die nächste Berechnung kann somit erfolgen. Bleibt eine Bestätigung aus, wird der Prozess abermals in die Warteschlange eingefügt, wodurch eine garantierte Durchführung ermöglicht wird. Das gesamte System skaliert somit linear.

Nachdem die Ergebnisse zusammengeführt wurden, erfolgt die *Match-Decision*. Wurde eine einzelne Metrik festgelegt, wird jeder errechnete Wert gegen den entsprechenden festgelegten Grenzwert geprüft. Befindet sich der Wert innerhalb des Grenzwertes, zählt das Vergleichspaar als Treffer (*Match*). Wurden mehrere Metriken ausgewählt, muss eine Kombination der errechneten Werte erfolgen. Köpcke und Rahm unterscheiden dabei drei Arten von Kombinationen: *Numerisch*, *Regelbasiert* und *Workflow-basiert* (vgl. [37]). Ein numerischer Ansatz kombiniert die Ähnlichkeitswerte der Enitäts-Paare mit einer numerischen Funktion, z. B. eine gewichtete Summe oder ein gewichteter Mittelwert der Ähnlichkeitswerte. Anhand des resultierenden numerischen Wertes kann die Match-Decision erfolgen. Ein Problem ist hierbei die Verschiedenheit der Wertebereiche der Metriken. Nicht aus jeder Metrik resultieren Werte im Intervall $[0, 1]$ und nicht bei jeder Metrik spricht ein Wert, der gegen 0 läuft, für große Ähnlichkeit und ein Wert, der gegen 1 läuft, für Verschiedenheit. Aus diesem Grund ist ein regelbasierter oder Workflow-basierter Ansatz geeigneter, da dabei eine logische Kombination von Treffer-Bedingungen zur Match-Decision herangezogen wird. Eine könnte sein, dass bei k ausgewählten Metriken mindestens $\lfloor \frac{k}{2} \rfloor$ Treffer erzielt werden müssten, damit das Bildpaar global als Treffer gilt.

Aus der Menge aller ermittelten Duplikate können im Anschluss die Duplikatgruppen, bspw. durch Bildung der transitiven Hülle, erzeugt und in einer geeigneten Repräsentation (z. B. JSON, CSV oder XML) ausgegeben werden. Weiterhin besteht die Möglichkeit, die Ähnlichkeitsmatrizen direkt nach der Zusammenführung der Werte auszugeben, um weitere Evaluationen auszuführen. Dieser Datenfluss ist in der Architekturskizze mit einer gepunkteten Linie dargestellt. Die in Kapitel 7.2 durchgeführte Evaluation der Invarianzen konnte bspw. mit diesem Feature erstellt werden.

Wurde bei der Eingabe die Liste der Bildpfade manuell nach Duplikaten gruppiert, um eine Evaluation der Effektivität einzelner Metriken durchzuführen, wird die Match-Decision mehrfach mit einem schrittweise inkrementierten Grenzwert getroffen. In der Abbildung wird dieser Schritt als *Effektivitätsanalyse* bezeichnet und der Datenfluss ist mit einer gestrichelten Linie gekennzeichnet. Dabei werden pro Metrik der kleinste und größte Ähnlichkeitswert als Grenzen eines Intervalls I festgelegt. Der Grenzwert wird zu Beginn auf den kleinsten Ähnlichkeitswert und somit auf die untere Grenze von I gesetzt. Nun werden Precision, Recall und F-Measure für diesen Grenzwert berechnet. Der Grenzwert wird im Anschluss sukzessive inkrementiert, bis die obere Grenze von I erreicht ist. Für jeden Schritt werden die Maße der Effektivität neu berechnet und persistiert. Man erhält somit eine Aussage über die Effektivität der Metrik

für einen gegebenen Grenzwert, deren Qualität jedoch nur bei einer hinreichend großen Menge von Eingabebildern und einem qualitativ hochwertigen Goldstandard sichergestellt werden kann. Die Ausgabe kann pro Metrik in einer *Threshold-Effektivtät-Tabelle* ausgegeben werden, welche beispielhaft in Tabelle 6 dargestellt ist. Die Daten können zur Visualisierung in ein kartesisches Koordinatensystem übertragen werden, bei dem die Grenzwerte auf der Abszisse und der Wert der Effektivität auf der Ordinate abgebildet wird. Diese Darstellung wird innerhalb der Evaluation der Effektivität im Kapitel 7 verwendet.

Threshold	Precision (P)	Recall (R)	F-Measure
0.0	0.0	0.0	0.0
0.1	0.2	0.4	0.26
...	...	...	...
1.0	0.1	0.9	0.18

Tabelle 6: Beispiel für die Threshold-Effektivtät-Tabelle

5.3. Feature-basierte Ähnlichkeitsmetrik

Ein wesentlicher Bestandteil eines Systems zur bildbasierten Deduplikation ist die Verwendung von Ähnlichkeitsmetriken zum Vergleich zweier Bilder. Anhand einer Metrik, welche im Allgemeinen als reeller Wert im Intervall $[0, 1]$ repräsentiert wird, kann eine Aussage zur Ähnlichkeit beider Bilder getroffen werden. Über einen festgelegten Grenzwert kann das Bildpaar im Anschluss als Duplikat bzw. kein Duplikat klassifiziert werden. In diesem Abschnitt wird ein Ansatz zur Berechnung einer Metrik vorgestellt, welche auf den im Abschnitt 4.3 vorgestellten Technologien zur Feature-basierten Ermittlung von Bildähnlichkeiten aufbaut. Dabei werden zunächst die Anforderungen an eine Ähnlichkeitsmetrik formuliert, woraufhin im Anschluss das entworfene Konzept vorgestellt wird. Im letzten Abschnitt wird die Parallelisierbarkeit des Konzeptes diskutiert. Im Kapitel 7 wird diese in einem fairen Vergleich mit anderen Metriken evaluiert.

5.3.1. Die ideale Ähnlichkeitsmetrik

Zu Beginn werden Eigenschaften einer idealen Ähnlichkeitsmetrik formuliert, welche als Anforderung für das nachfolgende Konzept der Feature-basierte Ähnlichkeitsmetrik gelten sollen. Piotrowski hat diese in [46] wir folgt zusammengefasst:

Sei A eine Menge an Bilder $\{a_1, a_2, ..., a_m\}$. Möchte man eine ideales Ähnlichkeitsmaß $d(a_i, a_k)$ definieren, beschreibt man dieses mathematische als Metrik für A, welche die folgenden Bedingungen erfüllt:

$$d(a_i, a_k) = 0 \iff a_i = a_k, \tag{5}$$

$$d(a_i, a_k) = d(a_k, a_i), \tag{6}$$

$$d(a_i, a_m) \leq d(a_i, a_k) + d(a_k, a_m), \tag{7}$$

$$d(a_i, a_k) > 0 \iff a_i \neq a_k, \tag{8}$$

$$d(a_i, a_k) \leq 1. \tag{9}$$

Aus der Metrik soll sich ausschließlich bei identischen Bildern ein Wert von 0 ergeben (5). Außerdem soll die Metrik symmetrisch (6) sein und die Dreiecksungleichung erfüllen (7). Aus den ersten drei Gleichungen wird impliziert, dass alle Paare verschiedener Bilder einen positiven Wert ergeben (8), welcher maximal den Wert 1 annehmen kann (9). Man erhält somit eine Funktion $d : A \times A \rightarrow [0, 1]$[5] welche ein Bildpaar auf einen Skalar abbildet. (Vgl. [46])

Weiterhin sollte eine ideale Ähnlichkeitsmetrik nicht die in der Problemanalyse definierten Probleme 1 und 2 aufweisen (vgl. Kapitel 3). Wenn a_i und a_k Bilder semantisch gleicher Objekte sind, sollte $d(a_i, a_k)$ einem Wert entsprechen, der gegen 0 strebt und sich deutlich von $d(a_i, a_m)$ oder $d(a_k, a_m)$ unterscheiden, wenn a_m ein semantisch verschiedenes Objekt abbildet. Bestenfalls streben $d(a_i, a_m)$ und $d(a_k, a_m)$ gegen 1.

5.3.2. Konzeption

Feature-basierte Algorithmen dienen der Ermittlung interessanter und eindeutiger Bereiche eines Bildes, den sogenannten Keypoints, und deren eindeutige Beschreibung in Form eines Deskriptor-Vektors (siehe Kapitel 4.3). Ein Bild wird somit durch die Menge aller darin enthaltenen Features eindeutig beschrieben. Sollen zwei Bilder miteinander verglichen werden, kann dies über den Vergleich deren Features realisiert werden. Sind hinreichend viele übereinstimmende Features zwischen beiden Bildern zu finden, kann von einer hohen Wahrscheinlichkeit ausgegangen werden, dass die Bilder Duplikate sind.

Um aus zwei gegebenen Bildern und der Übereinstimmungen deren Features (im Allgemeinen als *Matches* bezeichnet) eine Ähnlichkeitsmetrik nach den im Abschnitt 5.3.1 definierten Anforderungen zu berechnen, kann der nachfolgend beschriebene Ansatz herangezogen werden. Zum besseren Verständnis wird dieser in sechs konzeptionelle Schritte aufgeteilt, welche im Anschluss erläutert werden.

1. Laden der Bilder (Graustufen)
2. Keypoint-Detektion
3. Descriptor-Extraktion
4. Descriptor-Matching
5. Filterung der Treffer durch Ratio-Test
6. Berechnung der Ähnlichkeit

[5] In der Quelle wird das Intervall als $[0, 1)$ angegeben, was bedeutet, dass der Wert 1 kein Bestandteil des Wertebereiches ist. Diese Festlegung hat jedoch keinen Einfluss auf die Topologie der Metrik.

Zur Veranschaulichung wird die Erläuterung der sechs Schritte anhand zweier Beispielbilder (Abb. 7 (a) und (b)) als Eingabe dargelegt. Diese stellen Fotografien einer semantisch äquivalenten Landschaft aus zwei verschiedenen Perspektiven dar. Die Abbildungen der Zwischenschritte wurden von der prototypischen Implementierung der Metrik (siehe Abschnitt 6.3) unter Verwendung der Bibliothek *OpenCV* ermöglicht und werden hier vorweg zum besseren Verständnis des Konzeptes abgebildet.

(a) (b)

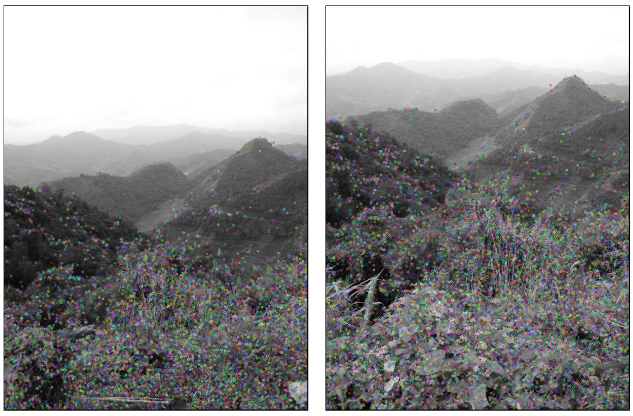
Abbildung 7: Fotografien einer Landschaft aus zwei verschiedenen Perspektiven [31]

1. Laden der Bilder (Graustufen) Zunächst werden beide Bilder als Graustufenbild geladen, da die später angewendeten Feature-basierten Verfahren keine Farbinformationen benötigen. Ein positiver Nebeneffekt ist die damit verbundene Verminderung des Speicherbedarfs. Abbildung 8 zeigt die Bilder in Graustufen.

(a) (b)

Abbildung 8: In Graustufen transformierte Bilder der Landschaft (vgl. [31])

2. Keypoint-Detektion Im Anschluss wird die Detektion der Keypoints mit einem Feature-basierten Verfahren (SIFT, SURF oder ORB) durchgeführt. Die ermittelten Keypoints sind in Abbildung 9 jeweils mit einen farbigen Kreis gekennzeichnet. Das gezeigte Beispiel ist das Resultat einer Keypoint-Detektion mit dem SIFT-Verfahren. Wurde in einem der beiden Bilder kein Keypoint detektiert, wird die Ähnlichkeitsmetrik auf 0 gesetzt und der Algorithmus mit Rückgabe dieses Wertes beendet.



(a) (b)

Abbildung 9: Farbige Markierungen der Keypoints nach erfolgter Keypoint-Detektion (vgl. [31])

3. Descriptor-Extraktion Nach der Keypoint-Detektion erfolgt die Descriptor-Extraktion, welche im Abschnitt 4.3 erläutert wurde.

4. Descriptor-Matching Wurde für jeden Keypoint der Deskriptor-Vektor berechnet, können nun übereinstimmende Features aus Bild (a) und (b) ermittelt werden. Dafür wird für jedes Feature aus (a) die Distanz zu allen Features aus (b) berechnet. Die geringste Distanz gilt als der beste Treffer und wird in Abbildung 10 als farbige Verbindung zwischen den entsprechenden Features dargestellt. Zu beachten ist, dass durch dieses Verfahren zu jedem Feature in (a) grundsätzlich ein Treffer in (b) existiert. Eine geeignete Filterung ist somit zwingend notwendig.

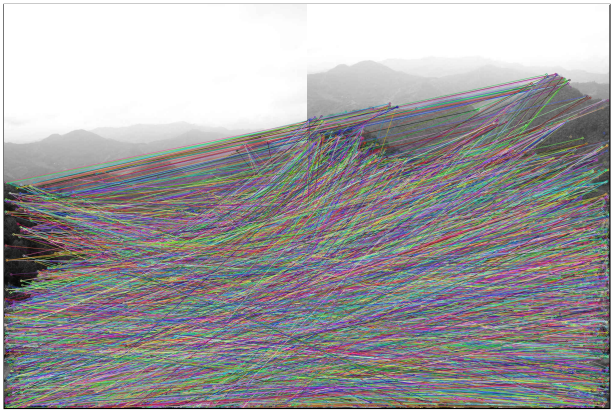


Abbildung 10: Farbige Linien zwischen den Features zur Markierung der Treffer (vgl. [31])

Ein Nachteil dieser Form des Matchings ist, dass die Ermittlung der Treffer nicht symmetrisch ist, da immer von einem Bild ausgehend die Distanzen zu den Deskriptoren des zweiten Bildes betrachtet werden. Das Ergebnis hängt somit von der Reihenfolge der Bilder ab, wodurch die Anforderung an eine symmetrische Metrik nicht gegeben ist. Um dieses Problem zu beheben bieten sich zwei Lösungsansätze an:

1. Sortierung der Bilder vor der Distanzberechnung durch Anwendung einer Hashfunktion auf den binären Rohdaten der Graustufenbilder mit anschließender aufsteigender Sortierung der Hashwerte. Die Berechnung der Distanzen wird immer ausgehend vom Bild mit „kleinerem“ Hashwert durchgeführt.
2. Berechnung der Distanzen *von beiden Bildern ausgehend* mit anschließender Mittelwertbildung der eindeutigen Trefferanzahl nach dem Ratio-Test in Schritt 5.

Beide Ansätze bewirken ein symmetrisches Verhalten der Metrik. Der erste Ansatz hat den Vorteil, dass das Deskriptor-Matching und der anschließende Ratio-Test einmalig ausgeführt werden muss. Als Hashfunktion könnte bspw. SHA-3 verwendet werden. Der Nachteil dieses Ansatzes und gleichzeitige Vorteil des zweiten Lösungsansatzes ist die Einbeziehung der Treffer von beiden Bildern ausgehend. Unterscheidet sich die Anzahl der eindeutigen Treffer vom ersten Bild ausgehend mit der Anzahl vom zweiten Bild ausgehend, ist die Bildung des Mittelwertes die bessere Wahl. Lösungsansatz 2 wird demnach für die hier entwickelte Konzeption ausgewählt. Bei einer Implementation könnte man die Möglichkeit anbieten, das Symmetrieverhalten optional zu deaktivieren, um die Laufzeit zu verringern.

5. Filterung der Treffer durch Ratio-Test Nachdem das Descriptor-Matching erfolgt ist, müssen die Treffer auf ihre Eindeutigkeit überprüft werden. Wie schon im Abschnitt 4.3.1 SIFT erläutert wurde, hat Lowe zur Filterung den Ratio-Test vorgestellt. Seien d_1 und d_2 die beiden geringsten Distanzen eines Features in Bild A zu den Features in Bild B, muss Gleichung 10 erfüllt sein, damit dieser Treffer als *eindeutig* interpretiert wird. Lowe verwendet einen Grenzwert von 0.8 als Richtwert. Zur Einhaltung der Symmetrieeigenschaft wird die Filterung der Treffer von beiden Eingabebildern ausgehend angewendet. Man erhält somit zwei Mengen an eindeutigen Treffern, von denen ausschließlich der Mittelwert für den nachfolgenden Schritt von Bedeutung ist.

$$\frac{d_1}{d_2} < 0.8 \tag{10}$$

Abbildung 11 zeigt die Ausgabe der Treffer, deren Anzahl durch den Ratio-Test deutlich reduziert wurden. Der Vergleich mit Abbildung 10 verdeutlicht dies nochmals.

6. Berechnung der Ähnlichkeit Mittels der im Schritt 5 durchgeführten Filterung kann nun aus der Anzahl der selektierten Treffer eine Ähnlichkeit im Intervall $[0, 1]$ berechnet werden. Sei x der Wert der resultierenden Metrik und n der Mittelwert der Anzahl der eindeutigen Treffer, kann der durch Gleichung 11 beschriebene Ansatz in Betracht gezogen werden. Dabei

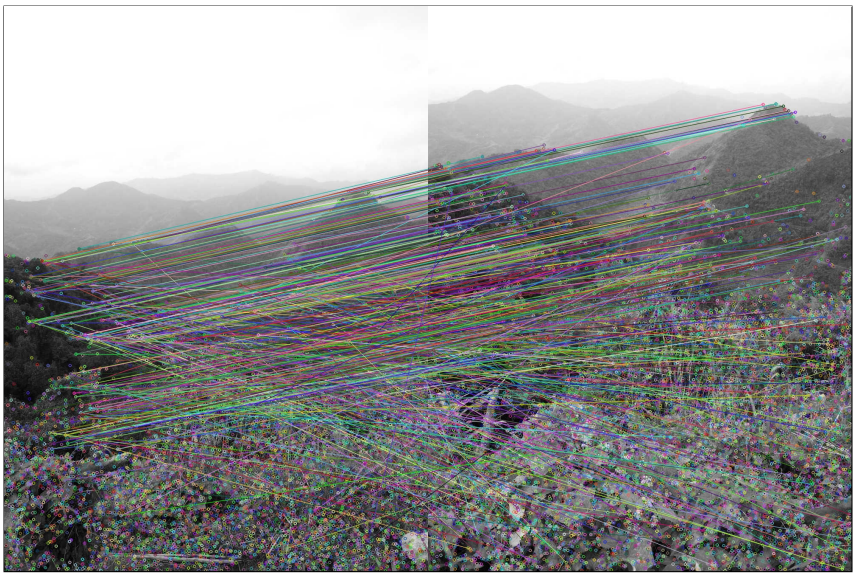


Abbildung 11: Durch den Ratio-Test reduzierte Treffer (vgl. [31])

wird angenommen, dass beim Vergleich zweier Bilder N eindeutige Treffer ausreichen, um diese als semantisch äquivalent und somit als Duplikat zu identifizieren.

$$\begin{aligned} x &= 1 \quad \text{für} \quad n \geq N \\ x &= \frac{n}{N} \quad \text{für} \quad n < N \end{aligned} \tag{11}$$

Während verschiedener Tests des hier beschriebenen Verfahrens hat sich ein Wert von $N = 200$ als geeignet herauskristallisiert. Dies ist für theoretische Betrachtungen jedoch nicht in allen Fällen sinnvoll, z. B. wenn die Anzahl der gefundenen Keypoints im Schritt 2 für Bild (a) oder (b) kleiner als 200 ist. Wurden beispielsweise im Bild (a) und (b) jeweils 50 Keypoints gefunden und nach dem Ratio-Test alle Treffer als eindeutig identifiziert, werden diese $n = 50$ Matches für die Berechnung der Metrik im Schritt 6 herangezogen. Bei einem Wert von $N = 200$ würde für dieses Beispiel ein Ähnlichkeitsmaß von $x = \frac{n}{N} = \frac{50}{200} = 0.25$ berechnet werden, welches eine widersprüchliche Aussage über die starke Übereinstimmung beider Bilder trifft. Ein Lösungsansatz ist die Festlegung von N auf die maximale Anzahl der detektierten Keypoints, wenn weniger als 200 für Bild (a) und (b) gefunden wurden. Seien k_a und k_b die Anzahlen der im Schritt 2 detektierten Keypoints der Bilder (a) und (b), kann Gleichung 12 für die Festlegung von N verwendet werden. Für das genannte Beispiel würde unter Beachtung dieser Gleichung $N = 50$ gelten und ein Ähnlichkeitsmaß von $x = \frac{n}{N} = \frac{50}{50} = 1.0$ resultieren.

$$\begin{aligned} N &= max(k_a, k_b) \quad \text{für} \quad max(k_a, k_b) < 200 \\ N &= 200 \quad \text{für} \quad max(k_a, k_b) \geq 200 \end{aligned} \tag{12}$$

Mit dieser Konzeption ist ein Algorithmus für eine Feature-basierte Ähnlichkeitsmetrik entstanden. Der gesamte Datenfluss kann als Zusammenfassung Abbildung 12 entnommen werden.

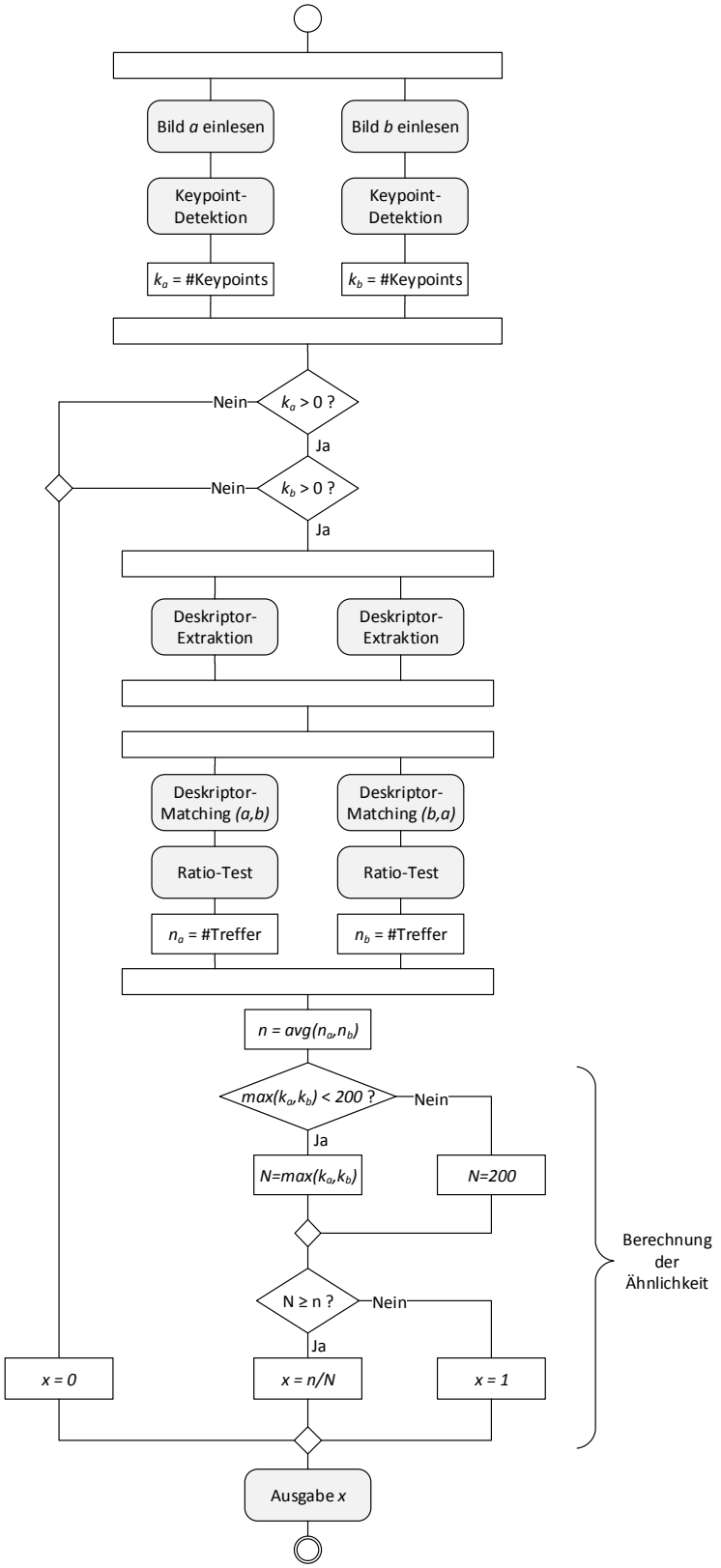


Abbildung 12: Übersicht der Ähnlichkeitsmetrik als Datenflussdiagramm

5.3.3. Diskussion der Parallelisierbarkeit

In diesem Abschnitt wird untersucht, inwieweit sich die konzipierte Anwendung für eine parallele Ausführung auf verteilten Systemen eignet. Dabei wird sich auf das Datenflussdiagramm (Abb. 12) und die im Abschnitt 5.3.2 aufgeführten sechs Schritte bezogen. Im Diagramm ist dabei durch Verzweigungsknoten (weiße Rechtecke) gekennzeichnet, wo eine parallele Verarbeitung möglich ist. Insgesamt existieren drei solche voneinander getrennte Abläufe.

Das Einlesen der Bilder und die Keypoint-Detektion kann jeweils separat in einem Prozess ausgeführt werden, da keinerlei Abhängigkeiten zwischen den Eingabebildern bestehen. Weiterhin kann die Deskriptor-Extraktion ebenso in einem separaten Prozess je Bild erfolgen. Es bietet sich an, die erzeugten Vektoren für nachfolgende Ausführungen zu persistieren, um bei erneuter Eingabe des identischen Bildes eine wiederholte Berechnung zu vermeiden. Das Deskriptor-Matching und der Ratio-Test kann ebenfalls in getrennten Prozessen erfolgen. Die Ergebnisse des Matchings müssen im Anschluss zusammengeführt werden, da diese zur Berechnung der Metrik gemäß Konzeption benötigt werden.

6. Prototypische Implementation

In diesem Kapitel wird eine prototypische Implementierung des SIMaSu-Systems, sowie der Feature-basierten Ähnlichkeitsmetrik aus Kapitel 5 vorgestellt. Die Metrik wird dabei als eigenständige Anwendung implementiert und über eine Schnittstelle an das System gekoppelt. Die Kombination ermöglicht unter anderem die Durchführung der in Kapitel 7 betrachteten Evaluationen. Der hier beschriebene Prototyp wurde primär zur Durchführung dieser entwickelt. Ein Blocking-Ansatz sowie die Kombination der Metriken und damit verbundene Erzeugung von Duplikatgruppen, wurden bisher nicht in den Prototypen integriert. Diese können jedoch leicht ergänzt werden.

6.1. Technologieauswahl

Tabelle 7 ist eine Auflistung der verwendeten Systeme und Bibliotheken, inklusive deren Version und Zweck innerhalb der Anwendung. Weiterhin ist angegeben, für welche Komponente die Technologie Verwendung findet, d.h. ist sie Bestandteil des SIMaSu-Systems, eine zu evaluierende Metrik oder wird sie für die Implementation der Feature-basierte Ähnlichkeitsmetrik verwendet.

System	Version	Komponente	Verwendung
PHP [45]	7.1	SIMaSu	Primäre Programmiersprache zur Implementation des SIMaSu-Systems
Symfony [57]	3.3	SIMaSu	PHP-Framework u. a. zum Einlesen von CLI- und Konfigurationsparametern
Composer [13]	1.5.2	SIMaSu	Paketverwaltung in PHP
MySQL [15]	5.7	SIMaSu	Speicherung der Ähnlichkeitswerte
RabbitMQ [49]	3.6.14	SIMaSu	Parallele Ausführung der Berechnungen auf mehreren Systemen
Imagick [25]	3.4.3	MSE-Metrik, Resize-PixelCompare	PHP-Bibliothek zum Einlesen und Verarbeiten von Bilddaten
ImageMagick [29]	7.0.7	pHash-Metrik	CLI-Anwendung zur Berechnung der pHash-Metrik
Jenssegers imagehash [28]	0.4.2	dHash-, aHash-Metrik	PHP-Bibliothek zur Berechnung der dHash- und aHash-Metrik
Java (JDK & JRE) [44]	8	Feature-basierte Ähnlichkeitsmetrik	Implementation der Feature-basierte Ähnlichkeitsmetrik
Apache Maven [6]	3.5.0	Feature-basierte Ähnlichkeitsmetrik	Build-Management-Tool zum Verwalten der Programmabhängigkeiten und Bauen der ausführbaren Anwendung
OpenCV [43]	2.4.9	Feature-basierte Ähnlichkeitsmetrik	Bibliothek mit Implementationen der Feature-basierten Technologien SIFT, SURF und ORB

Tabelle 7: Verwendete Systeme für die Implementation des Prototypen

Im Kontext dieser Masterarbeit wurde sich im Zuge der Implementation für die Programmiersprache *PHP* im Zusammenspiel mit der Nachrichten-basierte Middleware *RabbitMQ* entschieden. Letztere ermöglicht eine unkomplizierte Verteilung von Aufgaben an mehrere Systeme, was in diesem Kontext zur verteilten Berechnung der Ähnlichkeiten verwendet wird. Im Gegensatz dazu würden sich auch Frameworks aus dem *Big Data*-Bereich, z. B. *Apache Flink* oder *Apache Spark*, eignen. Die Feature-basierte Ähnlichkeitsmetrik wird als eigenständige Java-Anwendung implementiert und kann nach dessen Paketierung als Java-Archiv auf beliebigen Systemen ausgeführt werden, auf denen das *Java Runtime Enviroment (JRE)* und *OpenCV* installiert sind.

6.2. Implementierung

In diesem Abschnitt werden die Architektur des Prototyps und einige Implementierungsdetails näher betrachtet. Abbildung 13 zeigt die zugehörige Architekturskizze. Der gesamte ausführbare Quellcode ist auf der CD im Anhang A.1 zu finden.

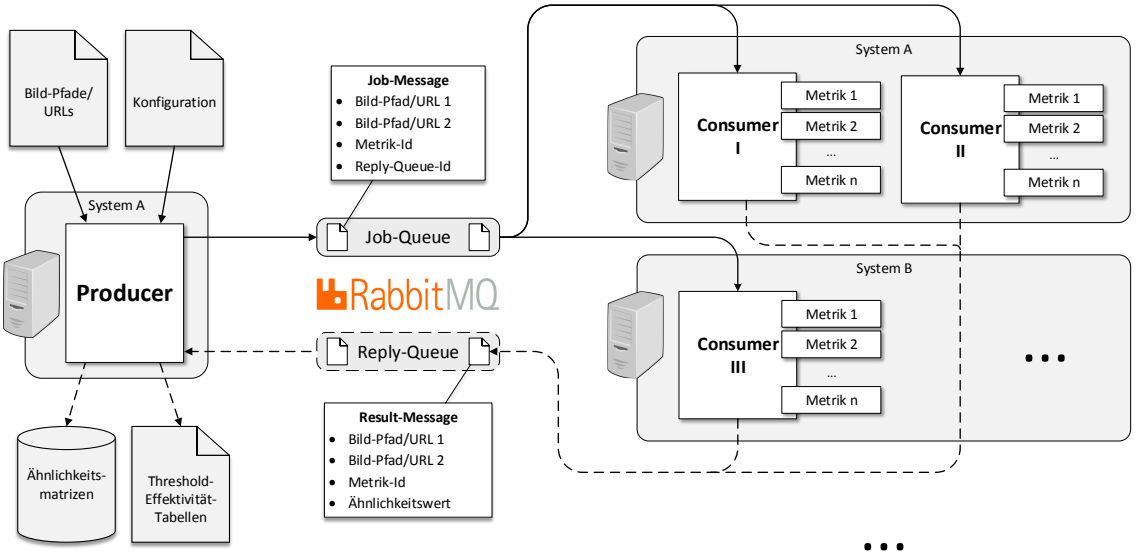


Abbildung 13: Architekturskizze des SIMaSu-Systems

RabbitMQ Queues Zur Verteilung der Berechnungen wurde sich für eine Nachrichten-basierte Middleware (auch *Message Broker*) entschieden. Neben *Apache Kafka* und *ActiveMQ* ist das Framework *RabbitMQ* eines der am weitesten verbreiteten Systeme dieser Art. RabbitMQ ist als Open-Source Software verfügbar und zeichnet sich durch Leichtgewichtigkeit, hohe Skalierbarkeit, leichte Installation und Plattform-Unabhängigkeit aus. Weiterhin existiert eine vollständige PHP-Bibliothek zur Integration der Funktionalitäten innerhalb einer PHP-Anwendung, was den Grund zur Wahl dieses Systems darstellt. Die Funktionsweise eines *Message Brokers* ist folgende: Der Broker fungiert als Server-Komponente und bietet die Möglichkeit, Warteschlangen zu erzeugen, welche mit dem Englischen Begriff *Queues* bezeichnet werden. Über eine Queue können sogenannte *Messages* in Form von Binärdaten gespeichert werden. Eine Message besteht aus einem Payload (üblicherweise im JSON-Format repräsentiert) und Metadaten. der

Broker übernimmt dabei das Entgegennehmen und Weiterleiten der Messages über die Queues. Eine Instanz, die Nachrichten an den Broker sendet, wird als *Producer* bezeichnet. Instanzen, die Nachrichten von einer Queue entgegennehmen, bezeichnet man als *Consumer*. Dieser Jargon im Kontext von Nachrichten-basierten Systemen wird zum besseren Verständnis auch in dieser Arbeit verwendet.

Producer Die *Producer*-Komponente ist der Einstiegspunkt der Anwendung. Sie nimmt die Bild-Entitäten und nötigen Metrik-Konfigurationen im JSON-Format entgegen und erzeugt nach Bildung der Vergleichspaare eine Message, welche hier zur besseren Unterscheidung als *Job-Messages* bezeichnet wird. Für jedes Bildpaar und jede Metrik wird eine solche Nachricht erzeugt und in die *Job-Queue* eingefügt. Die Nutzdaten sind demnach zwei Bildpfade und die Id einer Metrik.

Vor Erzeugung der ersten Job-Message, wird eine temporäre *Reply-Queue* mit eindeutiger Id erstellt, welche Bestandteil jeder erzeugten Job-Message ist. Die Reply-Queue dient als Rückkanal für *Result-Messages*, welche als Nutzdaten beide Bildpfade, die Id der Metrik und den berechneten Ähnlichkeitswert enthält. Nachdem alle Job-Messages versendet wurden, empfängt der Producer die eingehenden Result-Messages und fügt diese sukzessive als Ähnlichkeitsmatrizen in eine MySQL-Datenbank ein. Die Verwendung einer temporären Warteschlange hat den Vorteil, dass auf einem System mehrere Producer-Instanzen ohne gegenseitige Beeinflussung gestartet werden können.

Der Producer kann über den CLI-Befehl

```
php bin/console App:Producer -p {profile} -i {input} [-a]
```

gestartet werden. Über das Argument `-p {profile}` muss dabei der Pfad zu einer Konfigurationsdatei im JSON-Format angegeben werden, innerhalb der die zu verwendenden Metriken definiert worden sind. Eine ebenfalls im JSON-Format vorliegende Liste von Bild-Repräsentationen wird über den Parameter `-i {input}` spezifiziert. Das optionale Argument `-a` bewirkt die Durchführung der im Abschnitt 5.2 definierten Effektivitätsanalyse. Dabei wird auf Basis der berechneten Ähnlichkeiten für jede Metrik eine *Threshold-Effektivität-Tabelle* erzeugt. Die Tabellen werden als *.csv Datei in ein vorab festgelegtes Verzeichnis abgelegt. Wie in der Konzeption angegeben, setzt die Erzeugung der Tabellen voraus, dass die Bild-Entitäten bei der Eingabe manuell nach Duplikaten gruppiert worden sind.

Consumer Der *Consumer*-Prozess kann beliebig oft auf mehreren physischen Systemen über den Befehl

```
php bin/console App:Consumer
```

gestartet werden. In der Architekturskizze werden die Systeme mit *System A*, *System B*, etc. gekennzeichnet. Eine Voraussetzung dabei ist, dass eine logische Netzwerkverbindung zu dem RabbitMQ-Server besteht. Dessen Hostname sowie der Name der Job-Queue wird über eine Konfigurationsdatei spezifiziert. Auf den physischen Systemen müssen *PHP*, *ImageMagick*, das *Java JRE* und *OpenCV* (siehe Tabelle 7) installiert sein, damit der Consumer und dessen enthaltene Berechnung der verschiedenen Metriken ausgeführt werden kann. Eine Metrik ist durch Festlegung einer abstrakten Klasse (siehe Listing 1) lose an die Anwendungslogik des Consumers gekoppelt, was eine Erweiterbarkeit mit zusätzlichen Metriken erleichtert. Jede Metrik wird als

Klasse repräsentiert, die von der abstrakten abgeleitet ist. Die verpflichtende Methode `compare` nimmt zwei Bildpfade entgegen und gibt ein `CompareResult`-Objekt zurück. Wie innerhalb dieser Funktion die Berechnung der jeweiligen Metrik durchgeführt wird, ist dabei nicht festgelegt. Es können beispielsweise eine externe Anwendung per CLI-Befehl aufgerufen oder PHP-Bibliotheken verwendet werden.

```
1 <?php
2 /** File AbstractSimilarityCalculator.php **/
3 abstract class AbstractSimilarityCalculator
4 {
5     abstract function compare(string $srcImg, string $cmpImg) : CompareResult;
6
7     abstract function isMatch(float $simValue, float $threshold) : bool;
8 }
```

Listing 1: Abstrakte Klasse zur losen Kopplung der Metriken

Wird ein Consumer gestartet, verbindet sich dieser mit der Job-Queue und liest eine Job-Message ein. Für die darin enthaltenen zwei Bilder wird der Ähnlichkeitswert berechnet. Die innerhalb der Job-Message angegebene Metrik-Id bestimmt dabei die zu verwendende Klasse und somit die Berechnungslogik. Im Anschluss wird die Result-Message erzeugt und in die Reply-Queue eingefügt, deren Id ebenfalls innerhalb der Job-Message spezifiziert wurde.

Ähnlichkeitsmetriken können sich in ihrem Wertebereich und dessen Interpretation unterscheiden. Liefern zwei verschiedene Metriken A und B Werte im Intervall $[0, 1]$, so kann es sein, dass der Wert 0 für A eine hohe Übereinstimmung der Bilder bedeutet, wohingegen für B auf Verschiedenheit schließen lässt. Die Logik der Match-Decision ist somit von der jeweiligen Metrik und deren spezifischen Eigenschaften abhängig. Aus diesem Grund ist in der abstrakten Klasse ebenfalls eine abstrakte Funktion `isMatch` (siehe Listing 1, Zeile 7) implementiert. Diese nimmt einen Ähnlichkeits- und Grenzwert als Parameter entgegen. Die interne Logik der abgeleiteten Klassen führt anhand der Werte die Match-Decision durch und gibt `true` zurück, wenn der Wert einen Treffer darstellt (andernfalls `false`).

6.3. Prototyp der Feature-basierten Ähnlichkeitsmetrik

Damit das im Abschnitt 5.3 erläuterte Konzept der Feature-basierten Ähnlichkeitsmetrik prototypisch umgesetzt werden kann, wurde eine auf *Java* basierende CLI-Anwendung mit dem Build-Management-Tool *Maven* erzeugt. Die Bibliothek *OpenCV* [43] bietet Implementationen der drei Feature-basierten Technologien SIFT, SURF und ORB, was ein Alleinstellungsmerkmal der Bibliothek darstellt. Die darin enthaltene Java-API ermöglicht eine einfache Integration der Funktionalitäten in die Anwendung. Voraussetzung für die Verwendung der API ist eine bestehende OpenCV-Installation. Innerhalb dieser Ausarbeitung wird OpenCV in der Version 2.4.9 verwendet.

Zu Beginn werden beide Eingabebilder in ein OpenCV-eigenes Objekt überführt. Dazu wird die Klasse `Highgui` verwendet, welche entsprechende statische Methoden zum lesen von Bilddateien als Graustufenbilder enthält. Im Anschluss wird ein `FeatureDetector`-Objekt erzeugt, dessen `detect`-Methode die Keypoints aus den Eingabebildern detektiert und in eine Liste speichert. Bei der Erzeugung des Objektes wird über einen Parameter angegeben, welche Art von

Detektor verwendet werden soll. Zwischen den Feature-basierten Technologien SIFT, SURF und ORB bestehen Unterschiede in dem Verfahren der Detektion der Keypoints, der Extraktion der Deskriptoren und dessen Berechnung der Distanz. Abhängig von dem angegebenen 3. Argument (siehe unten) wird die entsprechende Technologie im Programmablauf verwendet. Der nächste Schritt ist die Erzeugung eines `DescriptorExtractor`-Objektes. Über dessen `compute`-Methode werden durch Übergabe der Eingabebilder und detektierten Keypoints die Deskriptoren berechnet und in Kollektionen abgelegt.

Für das Matching der Deskriptoren muss zunächst ein Objekt der Klasse `DescriptorMatcher` erstellt werden. Das zu verwendende Matching-Verfahren wird über einen Parameter definiert. Für SIFT und SURF stehen innerhalb der OpenCV-Bibliothek zwei populäre Matching-Verfahren zur Verfügung. Zum einen das sogenannte *Bruteforce*-Matching, was die Verwendung der euklidischen Distanzfunktion darstellt, zum anderen die Verwendung der von Lowe und Muja vorgestellten FLANN-Bibliothek. Für ORB wird die Hammingdistanz verwendet, da die Deskriptoren des ORB-Algorithmus als binäre Vektoren repräsentiert werden. Das eigentliche Matching übernimmt die Funktion `knnMatch`, welche auf dem `DescriptorMatcher`-Objekt aufgerufen wird. Dieser werden beide Deskriptor-Vektoren und die Anzahl der benötigten Treffer übergeben. Da für den anschließenden Ratio-Test die zwei besten Treffer benötigt werden, wird die Anzahl auf den Wert 2 gesetzt. Die Treffer werden wiederum in einer Kollektion gespeichert, welche im Listing 2 mit `matchesList` bezeichnet wird. Zeile 2 bis 7 zeigen die Iteration über alle Treffer-Paare und die Anwendung des von Lowe publizierten Ratio-Test. Treffer, die damit als eindeutig identifiziert wurden, werden in die Kollektion `goodMatchesList` übertragen. Zur Einhaltung der Symmetrie muss das Matching und die Filterung mit vertauschten Deskriptor-Vektoren angewendet werden (siehe Abschnitt 5.3.2). Abschließend wird der Ähnlichkeitswert auf Grundlage der Konzeption berechnet.

```
1 // Step 5: filter matches by ratio test
2 for (MatOfDMatch match : matchesList) {
3     DMatch[] twoMatches = match.toArray();
4     if (twoMatches[0].distance < 0.7 * twoMatches[1].distance) {
5         goodMatchesList.add(match);
6     }
7 }
```

Listing 2: Filterung der Treffer durch den *ratio test*

Bei Ausführung der Anwendung müssen als Kommandozeilenargumente die Pfade bzw. die URLs beider Bilder, sowie das zu verwendende Feature-Verfahren angegeben werden. Eine Übersicht über die Argumente ist in nachfolgender Auflistung angegeben.

1. Speicherort des ersten Bildes als Pfadangabe oder URL, z. B. `/data/img/landscape.jpg`
2. Speicherort des zu vergleichenden Bildes als Pfadangabe oder URL
3. Angabe des Feature-Verfahrens und der Matching-Strategie durch numerischen Wert:
   0 - SIFT mit Bruteforce-Matching
   1 - SIFT mit FLANN-Matching
   2 - SURF mit Bruteforce-Matching
   3 - SURF mit FLANN-Matching
   4 - ORB mit Verwendung der Hamming-Distanz

4. Ausgaben von Status- bzw. Fehlermeldungen durch Angabe von `-v`

Die Ausgabe der Metrik erfolgt als numerischer Wert auf die Standardausgabe und kann somit von anderen Applikationen interpretiert werden. Wird der Parameter `-v` angegeben, werden zusätzlich Log-Nachrichten, wie bspw. die Anzahl der Treffer vor und nach der Filterung, ausgegeben. Ein beispielhafter Aufruf der Anwendung mit vollständiger Ausgabe ist in dem Listing 3 aufgeführt.

```
1  $ java -jar feature-metric.jar /data/landscape1.jpg /data/landscape2.jpg 0 -v
2  [OpenCV-Feature-Matcher] Welcome to OpenCV 2.4.9.0
3  [OpenCV-Feature-Matcher] Use configuration: SIFT, SIFT, Bruteforce ...
4  [OpenCV-Feature-Matcher] Read images ...
5  [OpenCV-Feature-Matcher] Detect keypoints ...
6  [OpenCV-Feature-Matcher] [10720.0] keypoints detected in source image ...
7  [OpenCV-Feature-Matcher] [7972.0] keypoints detected in compare image ...
8  [OpenCV-Feature-Matcher] Compute features ...
9  [OpenCV-Feature-Matcher] Match features ...
10 [OpenCV-Feature-Matcher] [10720] matches for S->C found ...
11 [OpenCV-Feature-Matcher] Apply ratio test for S->C with threshold [0.7] ...
12 [OpenCV-Feature-Matcher] [264] unique matches for S->C after ratio test ...
13 [OpenCV-Feature-Matcher] [7972] matches for C->S found ...
14 [OpenCV-Feature-Matcher] Apply ratio test for C->S with threshold [0.7] ...
15 [OpenCV-Feature-Matcher] [250] unique matches for C->S after ratio test ...
16 [OpenCV-Feature-Matcher] Average are [257] unique matches ...
17 [OpenCV-Feature-Matcher] Done! Similarity is: 1.0
18 1.0
```

Listing 3: Beispiel für Ausführung der Anwendung mit Ausgabe

In Zeile 1 ist der Aufruf der Anwendung als Java-Archiv zu sehen. Die beiden Eingabebilder `landscape1.jpg` und `landscape2.jpg` entsprechen dabei den Landschaftsaufnahmen (a) und (b) aus Abbildung 7. Innerhalb der Implementation wird das erste angegebene Bild als *source image* und das zweite Bild als *compare image* bezeichnet. Das ausgewählte Feature-Verfahren sowie die Matching-Methode wird in Zeile 3 ausgegeben. Der angegebene Parameter `0` entspricht in dem Fall der Keypoint-Detektion mit dem SIFT-Algorithmus und dem Bruteforce-Matching der Deskriptoren über die euklidische Distanzfunktion. Im Eingabebild `landscape1.jpg` wurden beispielsweise 10720 Keypoints detektiert (Zeile 6). Dies entspricht auch der Anzahl der Treffer aus dem Matching-Vorgang (Zeile 10), wenn das *source image* mit dem *compare image* verglichen wird. Lediglich 264 Treffer (Zeile 12) erfüllten Gleichung 10 des Ratio-Tests. Beim Matching des *compare image* mit dem *source image* entstanden 250 eindeutige Treffer (Zeile 15). Der Mittelwert von 257 wird für die Berechnung der Metrik herangezogen. Mit $N = 200$ und $n = 257$ ergibt die Lösung der Gleichung 11 einen Ähnlichkeitswert von 1.0 (Zeile 18). Ohne das Kommandozeilenargument `-v` würde ausschließlich die Metrik in Zeile 18 zurückgegeben werden.

Im Prototypen des SIMaSu-Systems wurde die lose Kopplung einer Ähnlichkeitsmetrik über die Bereitstellung einer abstrakten Klasse realisiert. Damit kann nun für jedes Feature-Verfahren eine abgeleitete Klasse erzeugt werden, innerhalb der mittels `shell_exec()`[6] die hier beschriebene Anwendung aufgerufen werden kann. Der Rückgabewert der Funktion entspricht der Ausgabe der Anwendung, welche in diesem Fall der berechnete Ähnlichkeitswert ist.

[6] Die PHP-Funktion `shell_exec()` führt ein Kommando auf der Shell aus und gibt den kompletten Output als String zurück.

7. Evaluationen

Die prototypische Entwicklung des SIMaSu-Systems ermöglicht die Ausführung verschiedener Evaluationen, die in diesem Kapitel erläutert werden. Dazu gehören eine Laufzeitanalyse, ein Experiment zur Untersuchung von Invarianzen und eine Evaluation der Effektivität einzelner Metriken. Die nachfolgende Übersicht (Abb. 14) zeigt die zugrundeliegenden Fragestellungen dieser Untersuchungen auf.

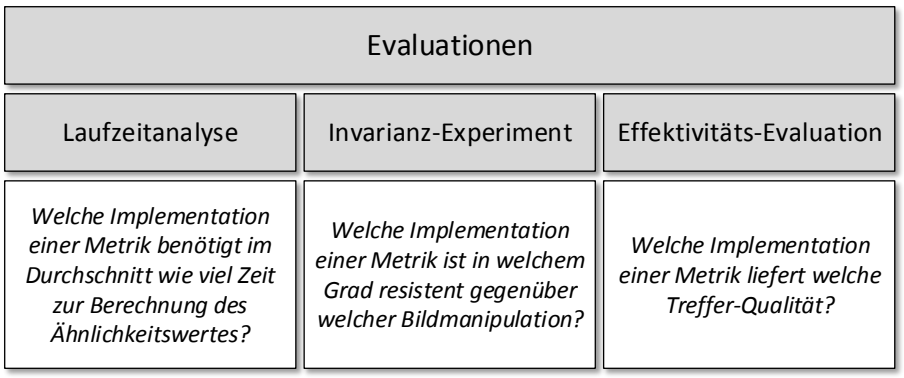


Abbildung 14: Übersicht der durchgeführten Evaluationen mit zugehörigen Fragestellungen

Die in allen drei Evaluationen untersuchten elf Implementationen von Ähnlichkeitsmetriken aus Kapitel 4 sind in Tabelle 8 aufgeführt. Jedem System wurde ein Kürzel zugeordnet, welches zur übersichtlicheren Darstellung in nachfolgenden Diagrammen verwendet wird.

System	Kürzel
ResizePixelCompare	RPC
Imagick compareImages	ICI
ImageMagick compare phash	ICP
Jenssegers pHash	JPH
Jenssegers aHash	JAH
Jenssegers dHash	JDH
SIFT mit FLANN	SIF
SIFT mit Bruteforce	SIB
SURF mit FLANN	SUF
SURF mit Bruteforce	SUB
ORB	ORB

Tabelle 8: Untersuchte Implementationen mit Kürzel

7.1. Laufzeiten

Die Laufzeit für die Berechnung eines Ähnlichkeitsmaßes spielt besonders für sehr große Datenmengen eine wichtige Rolle. In diesem Experiment wurde die durchschnittliche Laufzeit ermittelt, die eine Implementation einer Metrik benötigt, um aus zwei Bildern den Ähnlichkeitswert zu berechnen.

Setup Um ein vergleichbares Ergebnis zu erzielen, wurde für jede Messung das Bildpaar aus Abbildung 7 verwendet. Damit auch der Einfluss der Bildgröße auf die Laufzeit aufgezeigt werden kann, wurden die Messungen neben der Originaldimension von 2.448x3.264 px für zwei weitere Bildgrößen ausgeführt. Die kleineren Bilder wurden durch Ausschnitte aus den Originalen erstellt. Jede Messung wurde pro System und Bildpaar sechs mal durchgeführt. Die gemessene Zeit beinhaltet dabei neben der reinen Berechnungszeit auch das Laden der Bilder vom Sekundär- in den Hauptspeicher. Die Messungen wurden über die SIMaSu-Anwendung auf einem System durchgeführt, dessen Spezifikationen in Tabelle 9 aufgeführt sind.

CPU	Intel Core i7 4500U (1.8 GHz)
Betriebssystem	Ubuntu 16.04.3 LTS
RAM	1024 MiB DDR3
HDD	Samsung SSD 850 EVO 250GB (SATA, 3Gb/s)

Tabelle 9: Hard- und Softwareeigenschaften des für die Evaluationen benutzen Systems

Ergebnis Die Mittelwerte der Messergebnisse sowie der Standardabweichung sind innerhalb des Balkendiagrammes in Abbildung 15 dargestellt. Über jeden Balken ist zur Vollständigkeit die entsprechende Laufzeit in Sekunden abgetragen. Die Messungen ergaben einige interessante Erkenntnisse, auf die nachfolgend eingegangen wird.

Vergleicht man bei den Verfahren SIFT und SURF die Laufzeiten der beiden verschiedenen Matching-Strategien, erkennt man den Performance-Vorteil des FLANN-basierten Matchings gegenüber des Bruteforce-Matchings deutlich. Lediglich bei der kleinsten Bildgröße sind kaum Unterschiede wahrnehmbar, was voraussichtlich mit der geringen Anzahl an Keypoints zusammenhängt. Laut Literatur wird SURF verglichen mit SIFT als schnelleres Verfahren angesehen. Dieser Fakt konnte mit der hier getätigten Messung, mit Ausnahme der um 0.8 Sekunden geringeren Laufzeit von SURF gegenüber SIFT bei originaler Bildgröße, nicht bestätigt werden. SURF mit Bruteforce-Matching benötigte von allen Methoden die meiste Zeit. Die auf der ORB-Technologie basierte Metrik liefert für alle Bildgrößen eine Ausführungszeit von unter einer Sekunde, was ebenfalls für die Verfahren Resize Pixel Compare und dHash gilt. Sie bedarf bspw. für die originale Bildgröße nur ca. ein Zehntel der Zeit des SURF-Verfahrens (SUF). Für die Perceptual Hash-Verfahren wurden im Durchschnitt die geringsten Laufzeiten gemessen.

Bei Feature-basierten Verfahren ist die Gesamtlaufzeit abhängig von der Anzahl der gefundenen Keypoints. Je mehr davon detektiert wurden, desto mehr Deskriptoren und Distanzen müssen berechnet werden. Die Bildgröße spielt bei diesen Technologien voraussichtlich nur eine Rolle bei dem Laden des Bildes in den Hauptspeicher. Bei zukünftigen Messungen sollten Bilder mit verschiedenen Inhalt als Grundlage für aussagekräftigere Ergebnisse gewählt werden. Zudem ist

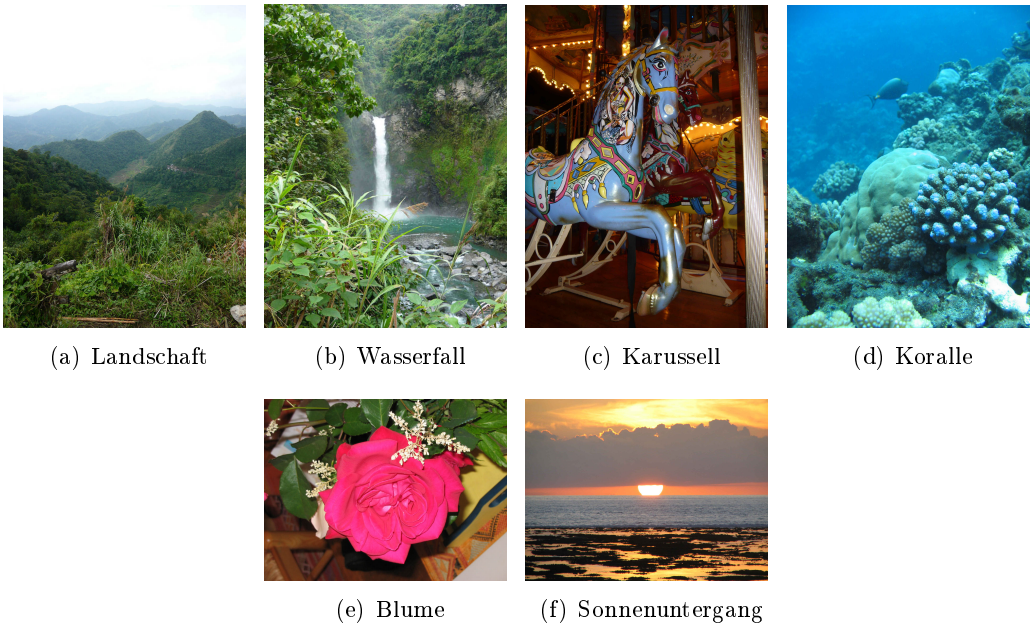
eine Unterteilung der Gesamtlaufzeit in die einzelnen Teilschritte *Laden der Bilder*, *Keypoint-Detektion*, *Deskriptor-Extraktion* und *Deskriptor-Matching*, für weiterführende Analysen hilfreich. Zudem könnte damit voraussichtlich der Grund für die hohe Laufzeit des SURF-Algorithmus bei einer Bildgröße von 1000x1000 px ermittelt werden.

Abbildung 15: Ergebnisse der Laufzeitmessung (in Sekunden) mit Standardabweichung

7.2. Invarianz-Experiment

In diesem Experiment wird untersucht, welche Ähnlichkeitswerte die hier betrachteten Implementationen von Metriken liefern, wenn ein Originalbild mit einer transformierten bzw. manipulierten Version dieses Bildes verglichen wird. Dabei werden die innerhalb der Problemdefinition erfassten digital veränderbaren Eigenschaften aus Tabelle 3 untersucht und somit das *Problem 3* aus dem Abschnitt 3.2 adressiert. Ziel der Untersuchung ist es eine Aussage zu treffen, welche Systeme gegenüber welchen Bildmanipulationen invariant sind. Damit wird eine Entscheidungshilfe geboten, die bei der Suche nach geeignete Ähnlichkeitsmetriken für einen speziellen Anwendungsfall herangezogen werden kann. Hat ein Unternehmen bspw. durch einen Integrationsschritt ausschließlich Bildduplikate mit unterschiedlichen Skalierungen erhalten, müssen Ähnlichkeitsmaße verwendet werden, welche invariant gegen Skalierungen sind.

Setup Den Ausgangszustand bilden sechs zufällig ausgewählte Fotografien (siehe Abb. 16) aus dem *Holiday-Datensatz* von *INRIA* [31]. Zur Erzeugung der manipulierten Bilder wurde eine PHP-Anwendung implementiert, welche auf einem Eingabebild 16 Transformationen mit unterschiedlichen Ausprägungen anwendet. Insgesamt resultieren aus einem Bild 172 transformierte Versionen.



(a) Landschaft (b) Wasserfall (c) Karussell (d) Koralle

(e) Blume (f) Sonnenuntergang

Abbildung 16: Originalbilder des Invarianz-Experiments

Durchführung Die Berechnung der Ähnlichkeiten für alle 11 betrachteten Implementationen von Metriken wird über den Prototypen des SIMaSu-Systems realisiert. Dieser speichert alle Werte in einer relationalen Datenbank, von der mittels SQL-Abfragen die benötigten Ähnlichkeiten extrahiert werden können. Für dieses Experiment sind ausschließlich Bildpaare aus dem Originalbild und einer transformierten Version relevant. Zur Veranschaulichung der Ergebnisse wurde für jede Kombination aus Transformation und Metrik ein Diagramm erstellt, in dem die

berechneten Werte abgetragen wurden. Aus 16 Transformationen und 11 Metriken ergeben sich somit $16 * 11 = 176$ Diagramme. Da nicht alle innerhalb dieser Masterarbeit aufgeführt werden können, wurden drei interessante selektiert und im nächsten Abschnitt erläutert.

Ergebnis Das erste Diagramm (Abb. 17) zeigt die mit der Jenssegers pHash-Metrik berechneten Ähnlichkeitswerte für unterschiedlich rotierte Versionen der Eingabebilder. Die Rotationen wurden im Intervall $[-90°, 180°]$ durchgeführt. Der Wertebereich der Metrik liegt im Intervall $[0, 1]$ wobei der Wert 0 Gleichheit und 1 Verschiedenheit bedeutet. Die Zuordnung der Linien zu den sechs Eingabebildern ist in der nebenstehenden Legende angegeben. Zusätzlich wurden aus allen Werten, die für eine bestimmte Rotation des Bildes berechnet wurden, der Mittelwert gebildet.

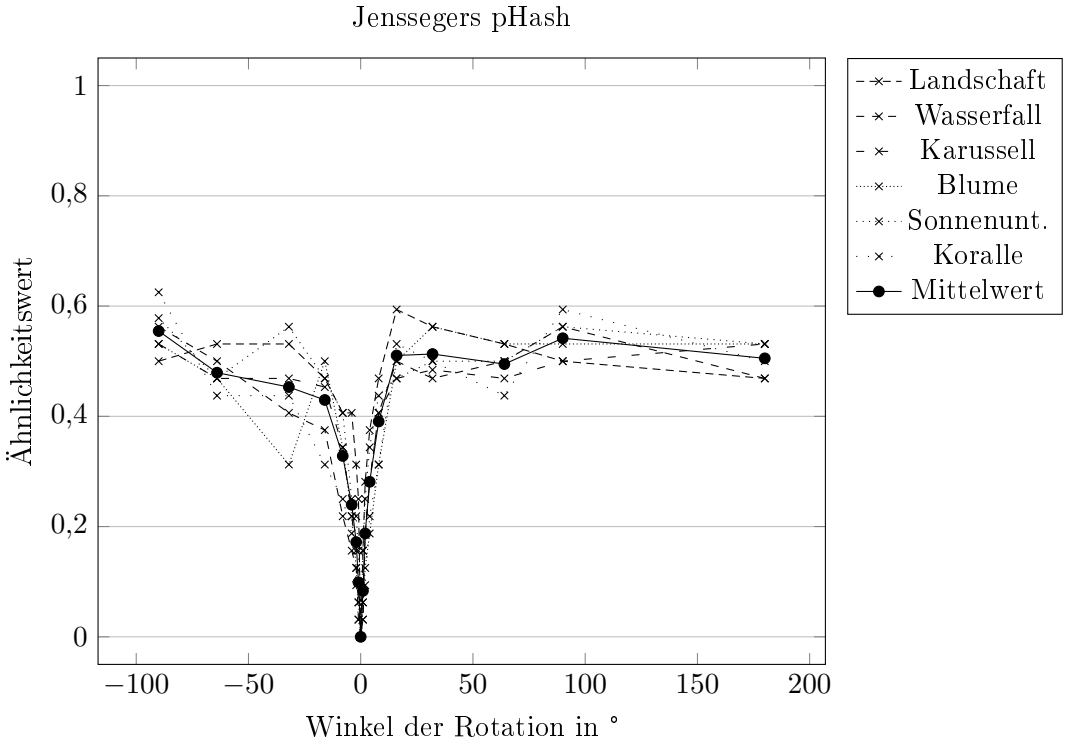


Abbildung 17: Ähnlichkeitswerte der Metrik *Jenssegers pHash* für Rotationen

Man erkennt, dass bei keiner Rotation (0°) des Bildes die Ähnlichkeit zum Originalbild 0 beträgt. Schon bei kleineren Rotationen steigt der Wert jedoch deutlich an und bleibt anschließend ab ±35° im Bereich zwischen 0.4 und 0.6. Der Anstieg zeigt, dass sich der berechnete Hash von einem leicht rotierten Bild zu dem des Originalbildes unterscheidet und somit deutlich höhere Ähnlichkeitswerte resultieren. Man kann zusammenfassen, dass die Implementierung der *pHash*-Metrik von *Jenssegers* variant gegenüber geringfügigen Rotationen des Bildes ist. Vergleicht man die Werte der verschiedenen Bilder an einer Position miteinander, unterscheiden sich diese nicht sehr stark voneinander. Die stärksten Abweichungen der Werte sind bei −32° zu erkennen. Man kann somit weiterhin die Annahme treffen, dass dieses Verhalten nicht abhängig von der Bilddomäne ist.

Das nächste ausgewählte Diagramm (Abb. 18) zeigt die Werte der innerhalb dieser Arbeit imple-

mentierten Ähnlichkeitsmetrik unter Verwendung des `ORB`-Algorithmus. Wie in der zugehörigen Konzeption (siehe Abschnitt 5.3) beschrieben, bedeutet bei dieser Metrik der Wert 1, dass die beiden miteinander verglichenen Bilder Duplikate sind. Betrachtet man die errechneten Mittelwerte in dem Diagramm, wird diese Wertigkeit ausschließlich bei einer sehr geringen Rotation von wenigen Grad (±2°), sowie einer Drehung um 180° erreicht.

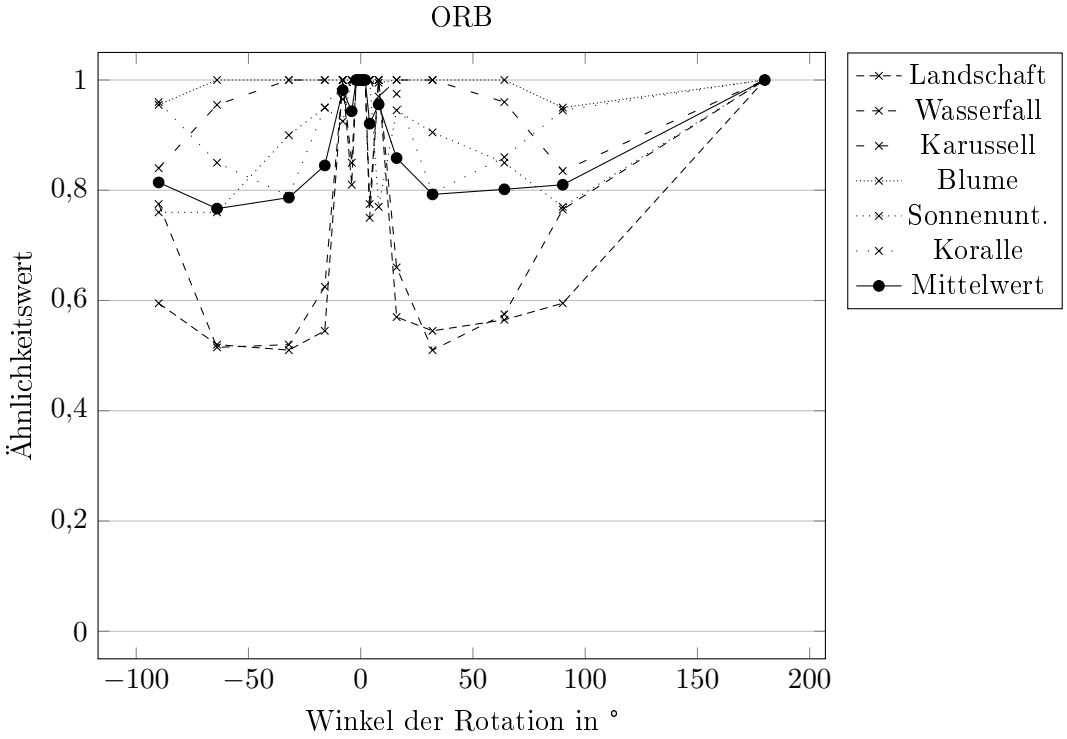


Abbildung 18: Ähnlichkeitswerte der Feature-basierten Metrik *ORB* für Rotationen

Sehr auffällig sind auch die Unterschiede der Werte bezogen auf die verschiedenen Bilder. Betrachtet man bspw. die Rotation um ±32°, ist bei den Bildern *Landschaft* und *Wasserfall* ein deutlicher Abfall der Werte zu sehen, wohingegen für die Fotografien der *Blume* und des *Karussells* der Wert 1 beträgt. Man kann hier die Annahme treffen, dass dieses Verhalten unter anderem abhängig von verschiedenen Eigenschaften, wie Farben und Strukturen innerhalb des Bildes, ist. Da bei Betrachtung des Mittelwertes trotz allem eine Verringerung des Wertes bei Erhöhung der Rotation feststellbar ist, kann diese Implementation des ORB-Algorithmus als variant gegenüber Rotationen klassifiziert werden.

Das letzte Diagramm bezieht sich ebenfalls auf rotierte Versionen der Originalbilder, jedoch wurde die Ähnlichkeit mit dem *SIFT*-Verfahren in Kombination mit dem *FLANN*-basierten Matching durchgeführt. Für alle Rotationen bleibt der Ähnlichkeitswert konstant 1. Dieses Verhalten gilt für alle sechs betrachteten Bilder und stellt den Idealfall einer Metrik dar. Das Verfahren ist somit vollständig invariant gegenüber Rotationen.

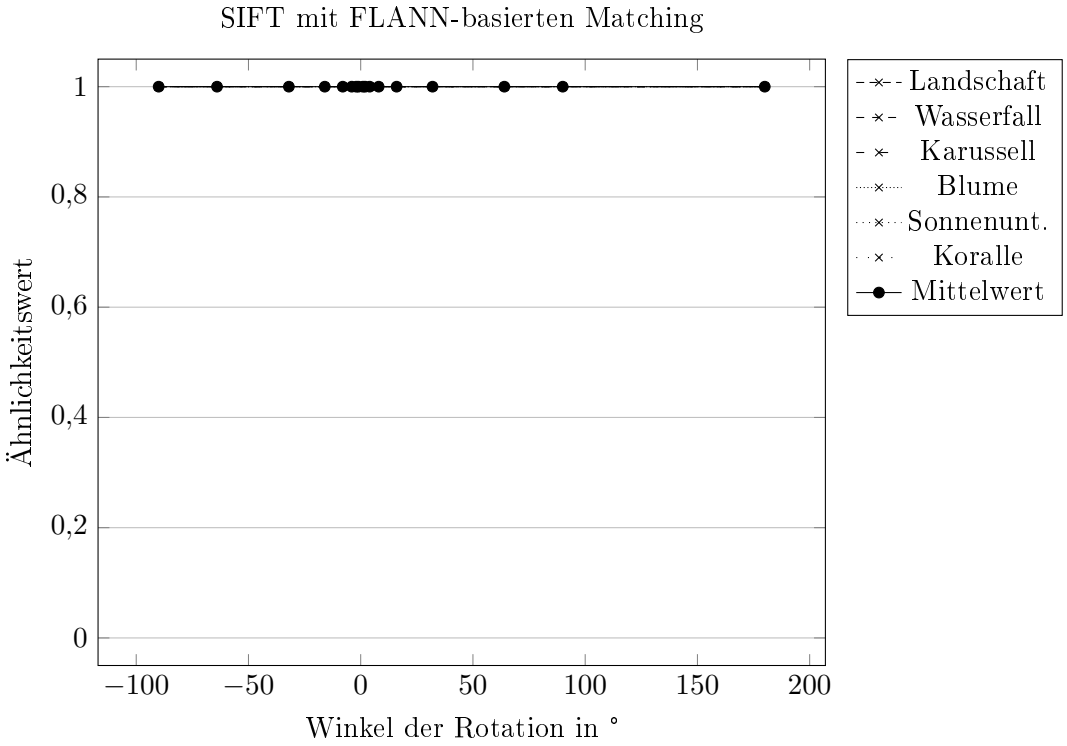


Abbildung 19: Ähnlichkeitswerte der Feature-basierten Metrik *SIFT* für Rotationen

Zusammenfassung Die drei beispielhaft betrachteten Diagramme bieten einen Einblick in die verschiedenen Verhaltensweisen der Metriken auf eine Bildtransformation. Über den Anstieg bzw. Abfall der Ähnlichkeitswerte bei quantitativer Veränderung der Transformation kann eine Aussage getroffen werden, ob eine untersuchte Implementation variant oder invariant gegenüber dieser ist. Um das Ergebnis aller Kombinationen aus Implementation und Transformation gegenüber zu stellen, bietet sich eine tabellarische Darstellung an. Die Beurteilung eines Ergebnisses wird dabei einer von vier Gruppen zugeordnet: *Invariant gegenüber Transformation*, *Invariant gegenüber geringfügiger Transformation*, *Variant gegenüber geringfügiger Transformation* sowie *Variant gegenüber Transformation*. Eine Zusammenfassung der Ergebnisse des Invarianz-Experimentes ist in Tabelle 10 aufgeführt. Die im Tabellenkopf genannten Transformationen wurden im Abschnitt 3.2 erläutert.

Betrachtet man die tabellarische Übersicht, kristallisieren sich einige interessante Zusammenhänge heraus. Beispielsweise ist keine Implementation invariant gegenüber Weichzeichnungen oder Kontraständerungen. Bei Spiegelungen weist ausschließlich das *pHash*-Verfahren von ImageMagick eine Invarianz auf. Im Gegensatz dazu sind die meisten Systeme robust gegen Kompression und Skalierung. Die Feature-basierten Metriken sind invariant gegenüber einem großen Teil der betrachteten Transformationen. Von denen schneidet jedoch der *ORB*-Algorithmus am schlechtesten ab. Die *Perceptual Hash*-Verfahren sind im Durchschnitt nicht sehr robust gegen Transformationen.

	Bogenförmig	Fassförmig	Rotation	Weichzeichnen	Helligkeit	Kontrast	Sättigung	Beschneidung	Gamma	Farbtönung	Spiegelung	Kompression	Kissenförmig	Skalierung	Trapez	Wasserzeichen
ResizePixelCompare	- -	- -	- -	- -	-	- -	-	- -	-	- -	-	+	-	-	- -	+
Imagick compareImages	- -	- -	- -	- -	-	- -	- -	- -	- -	- -	-	+	-	- -	- -	+
ImageMagick compare phash	- -	-	+	- -	- -	- -	- -	- -	- -	- -	+	-	- -	-	-	- -
Jenssegers pHash	- -	- -	- -	- -	- -	- -	-	- -	- -	- -	- -	+	- -	+	- -	- -
Jenssegers aHash	- -	- -	- -	- -	- -	- -	-	- -	- -	- -	- -	+	- -	+	-	- -
Jenssegers dHash	- -	- -	- -	- -	- -	- -	-	- -	- -	- -	- -	+	- -	+	-	- -
SIFT mit FLANN	+ +	+ +	+ +	- -	+ +	-	+ +	+	+ +	+	- -	+ +	+	+	+	+ +
SIFT mit Bruteforce	+ +	+ +	+ +	- -	+ +	-	+ +	+	+ +	+	- -	+ +	+	+	+	+ +
SURF mit FLANN	+ +	+ +	+ +	- -	+ +	-	+ +	+	+ +	+	- -	+ +	+	+	+	+ +
SURF mit Bruteforce	+ +	+ +	+ +	- -	+ +	-	+ +	+	+ +	+	- -	+ +	+	+	+	+ +
ORB	+ +	- -	-	- -	+	- -	+ +	-	+ +	-	- -	+ +	-	-	+	+

Legende:

++ Invariant gegenüber Transformation

\+ Invariant gegenüber geringfügiger Transformation

\- - Variant gegenüber geringfügiger Transformation

\- Variant gegenüber Transformation

Tabelle 10: Zusammenfassung der Ergebnisse des Invarianz-Experiments

7.3. Effektivität

Die letzte der drei Evaluationen beschäftigt sich mit der Effektivität der betrachteten elf Implementationen von bildbasierten Ähnlichkeitsmetriken. Die Qualität der resultierenden Ergebnisse kann über die Effektivitätsmaße *Precision (P)*, *Recall (R)* und *F-Measure (F)* (siehe Kapitel 2) gemessen werden. Zur Wiederholung gibt die *Precision* an, wie groß der Anteil der relevanten Duplikate innerhalb der Ergebnismenge ist. Ein Wert von 1 bedeutet, dass das Resultat ausschließlich aus erwarteten Duplikaten besteht und keine *False Negatives* beinhaltet (Idealfall). Der *Recall* liefert eine Aussage darüber, wie viele der vorab definierten Duplikate tatsächlich im Resultat vorkommen. Wurden alle gefunden, stellt das den Idealfall dar und entspricht einem *Recall* von 1. Das *F-Measure* ist das harmonische Mittel beider Maße.

Damit die genannten Maße der Effektivität berechnet werden können, muss vorab ein *Goldstandard* (siehe Kapitel 2) definiert werden. Dieser dient als Grundlage für alle zu untersuchenden Systeme. Er besteht aus einer hinreichend großen Menge von Bildern und Relevanz-Urteilen. Die Urteile kennzeichnen Bildpaare, welche von einer Person als Duplikate identifiziert wurden und somit relevante Ergebnisse darstellen. Je größer die Menge der Bilder ist, desto größer ist auch die Aussagekraft der Evaluation.

Eine Ähnlichkeitsmetrik liefert für ein gegebenes Bildpaar einen entsprechenden Wert, welcher Maß für die Ähnlichkeit der Bilder ist. Anhand eines Grenzwertes (auch *Threshold*) erfolgt die *Match-Decision*. Die Entscheidung, ob ein Bildpaar ein Duplikat darstellt, hängt somit immer vom festgelegten Grenzwert ab. Verschiebt sich dieser, ändern sich entsprechend auch die Effektivitätsmaße. Ziel dieser Evaluation ist es, für jede betrachtete Implementation einer Metrik aufzuzeigen, wie sich dessen Effektivität bei verschiedenen Thresholds verhält. Eine mögliche Darstellung dieser Beziehung ist innerhalb eines Diagrammes, bei dem der Grenzwert auf der Abszisse und die Effektivität auf der Ordinate abgetragen wird. Der Grenzwertbereich, für den bspw. das *F-Measure* oder die *Precision* am höchsten ist, kann für verschiedenste Anwendungsfälle als Richtwert genutzt werden.

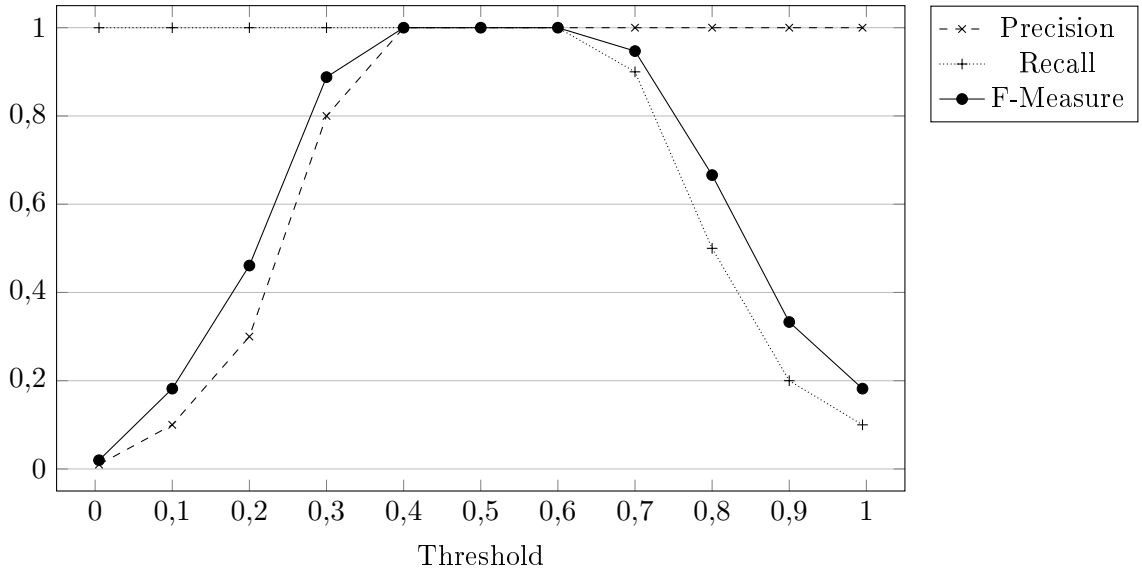


Abbildung 20: Beispiel für Effektivitätsmaße einer idealen Metrik

Abbildung 20 zeigt ein solches Diagramm, dessen Effektivitätswerte beispielhaft zur Darstellung eines idealen Falls gewählt wurden. Dem zugrunde liegt die im Abschnitt 5.3.1 beschriebene ideale Ähnlichkeitsmetrik im Intervall $[0, 1]$. Für 11 verschiedene Grenzwerte im Intervall $[0 + \varepsilon, 1 - \varepsilon]$ mit $\varepsilon > 0$ wurden *Precision*, *Recall* und *F-Measure* berechnet und abgetragen. Man erkennt, dass die Effektivität für Thresholds im Bereich 0.4 bis 0.6 den höchstmöglichen Wert erreicht. Das bedeutet, dass alle relevanten Duplikate und ausschließlich diese gefunden wurden. Bei einem sehr kleinen Grenzwert sind neben allen relevanten Duplikaten auch sehr viele falsche Treffer in der Ergebnismenge. Dadurch ist der *Recall* maximal bei einer sehr kleinen *Precision*. Wird der Grenzwert zu hoch gewählt, beinhaltet das Ergebnis im Idealfall zwar ausschließlich korrekt ermittelte Treffer, jedoch nur einen Bruchteil der relevanten Duplikate. Ein solches ideales Verhalten wird man in der Praxis mit hoher Wahrscheinlichkeit nicht finden.

Setup Zur Erzeugung des Goldstandards wurden zunächst 505 Fotografien aus dem *Holiday-Datensatz* von *INRIA* [31] ausgewählt. Im Anschluss wurden durch optische Vergleiche der Bilder die Relevanz-Urteile entschieden. Alle Duplikate wurden als Gruppierungen in eine JSON-Datei eingetragen, welche als Eingabedatensatz für das SIMaSu-System verwendet wird. Die Datei ist somit die Repräsentation des Goldstandards. Die im SIMaSu-System implementierte *Effektivitätsanalyse* berechnet für jede der elf untersuchten Metriken eine Threshold-Effektivtät-Tabelle (siehe Abschnitt 5.2). Die enthaltenen Werte werden im Anschluss als Diagramm dargestellt. Im nächsten Abschnitt werden zwei ausgewählte Diagramme abgebildet und ausgewertet. Die übrigen acht sind im Anhang A.2 innerhalb der Abbildungen 23 bis 31 aufgeführt.

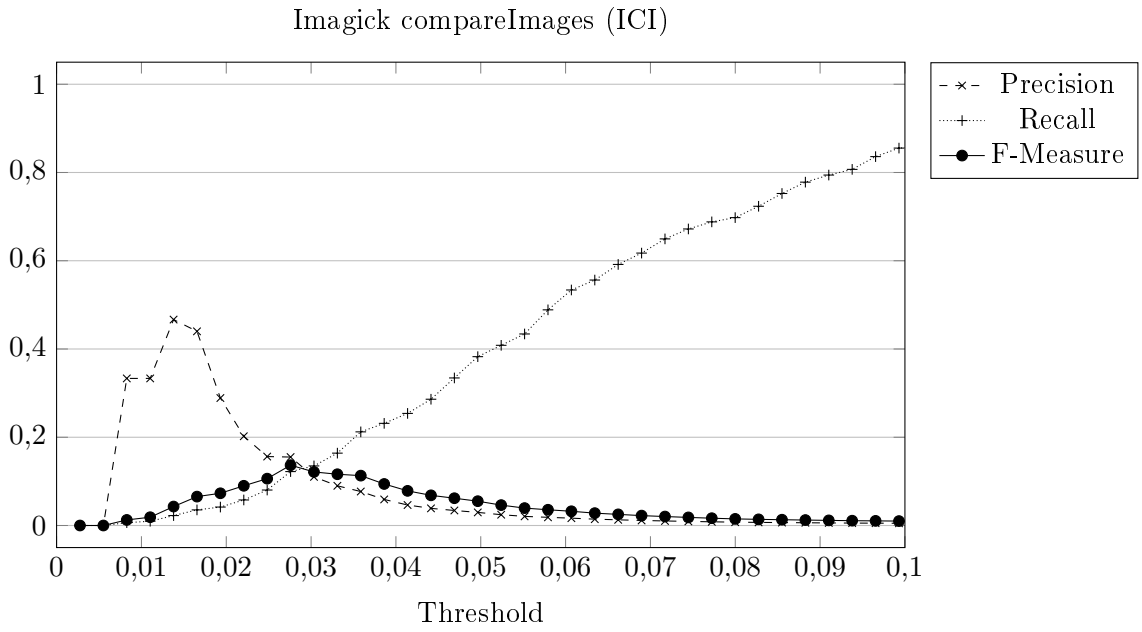


Abbildung 21: Effektivitätsmaße der Metrik *Imagick compareImages* (ICI) im Intervall [0, 0.1]

Ergebnis Abbildung 21 zeigt die Effektivitätswerte der Funktion *compareImages* der PHP-Bibliothek *Imagick*. Diese wurde auch innerhalb des Anwendungsfalls der Problemanalyse aus Kapitel 3 verwendet. Es ist zu beachten, dass ein Ähnlichkeitswert von 0 dabei einer sehr hohen Übereinstimmung der Bilder entspricht. Damit die Maximalwerte von *Precision* und *F-Measure*

besser erkennbar sind, werden ausschließlich Grenzwerte ≤ 0.1 betrachtet. Die höchste *Precision* $P_{max}(t_1) = 0.467$ wird mit einem Grenzwert $t_1 = 0.014$ erreicht. Mit einem Threshold von $t_2 = 0.028$ wurde das maximale *F-Measure* $F_{max}(t_2) = 0.137$ gemessen. Auf Grundlage dieses Ergebnisses ist es schwierig, eine qualitativ hochwertige Deduplikation mit dieser Metrik durchzuführen. Die im Abschnitt 3.1 aufgeführten Probleme *1* und *2* sind bei dieser Metrik deutlich wahrzunehmen.

Die Effektivität des SIFT-Algoritmus mit Bruteforce-Matching ist in Abbildung 22 abgebildet. Aus Übersichtlichkeitsgründen wurden die diskreten Punkte innerhalb des Diagramms entfernt.

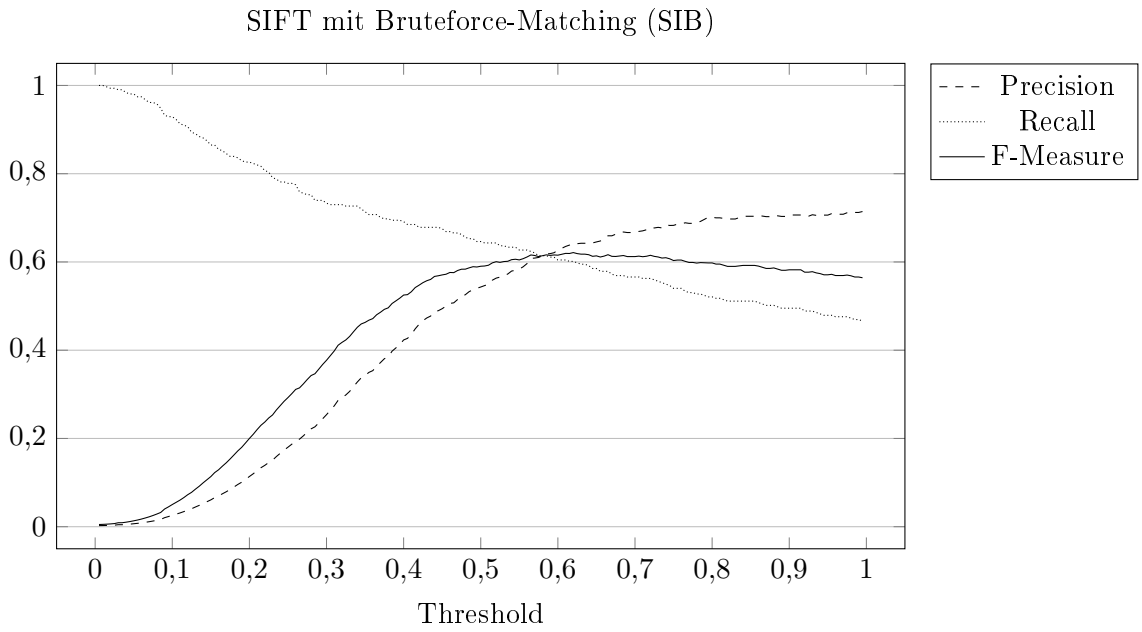


Abbildung 22: Effektivitätsmaße der Feature-basierten Metrik *SIFT* mit *Bruteforce-Matching* (SIB)

Betrachtet man den gesamten Wertebereich der Metrik, erkennt man einen kontinuierlichen Anstieg der *Precision* bei ebenfalls kontinuierlichem Abfall des *Recalls*. Verglichen mit dem Diagramm der *Imagick*-Bibliothek (Abb. 21), ähnelt das der SIFT-Metrik schon eher den idealen Effektivitätsmaßen aus der Abbildung 20. Die *Precision* $P_{max}(t_1) = 0.714$ ist maximal bei einem Grenzwert von $t_1 = 1$. Das bedeutet, dass 71.4% aller Bildpaare, für die eine Ähnlichkeit von 1.0 berechnet wurde, auch tatsächlich Duplikate sind, bzw. im Goldstandard als solche festgelegt wurden. Von den hier betrachteten elf Metriken, ist das der zweithöchste erreichte Wert. Bei einem Grenzwert von $t_2 = 0.62$ erreicht das *F-Measure* den Maximalwert von $F_{max}(t_2) = 0.621$, was den höchsten Wert aller Metriken darstellt.

Zusammenfassung Die Maximalwerte von *Precision* und *F-Measure* wurden für jede Metrik in Tabelle 11 aufgeführt. Ist ein Effektivitätsmaß über mehrere Grenzwerte konstant geblieben, ist das entsprechende Intervall angegeben. Mit einem Wert von 0.906 erreichte die Feature-basierte Metrik ORB die höchste Precision. Wie schon erwähnt, erzielte die Metrik *SIFT* mit angewandten *Bruteforce-Matching* das beste *F-Measure* mit einem Wert von 0.621. Die geringsten Maximalwerte ergaben sich für das *ResizePixelCompare*-Verfahren. Die fünf Feature-basierte Metriken erreichten im Durchschnitt die höchste Effektivität.

Metrik	t_1	$P_{max}(t_1)$	t_2	$F_{max}(t_2)$
RPC	0.16	0.015	0.15	0.029
ICI	0.014	0.467	0.028	0.137
ICP	4.5 - 5.0	0.6	10	0.135
JPH	0.188 - 0.203	0.75	0.283 - 0.294	0.133
JAH	0.064 - 0.074	0.049	0.064 - 0.074	0.071
JDH	0.097 - 0.105	0.5	0.219 - 0.232	0.149
SIF	1	0.694	0.87 - 0.875	0.600
SIB	1	0.714	0.62	**0.621**
SUF	1	0.272	0.995	0.366
SUB	1	0.332	0.995	0.411
ORB	0.345	**0.906**	0.065	0.537

Tabelle 11: Auflistung der besten Effektivitätsmaße mit jeweiliger Angabe des Thresholds

8. Zusammenfassung und Ausblick

Zu Beginn der Arbeit wurde die Notwendigkeit eines Systems zur skalierbaren bildbasierten Deduplikation anhand zweier Anwendungsfälle dargelegt und die Bedeutung dieses Themas für die Praxis aufgezeigt. Nachdem die Begriffe digitales Bild, Duplikat, Deduplikation und Effektivität theoretisch definiert wurden, erfolgte eine umfangreiche Problemanalyse anhand eines konkreten Anwendungsfalls. Darin wurde ein Szenario aufgezeigt, in dem ein Unternehmen ein automatisiertes Verfahren zur Ermittlung von Duplikaten verwendet. Es kristallisierten sich fünf Problemstellungen heraus, welche diese Arbeit motivierten und innerhalb der darauffolgenden Kapitel adressiert wurden. Dazu zählen das korrekte Auffinden von Duplikaten, das Vermeiden von fälschlicherweise als Duplikate klassifizierte Bilder, der Einfluss von Bildtransformationen auf Ähnlichkeitsmetriken, der fehlende faire Vergleich bestehender Metriken sowie die quadratische Komplexität der Deduplikation. Anhand aktueller Forschungen wurde die fehlende Berücksichtigung von Bild-Entitäten bei Deduplikationssystemen aufgezeigt. Derzeit verfügbare Systeme und Technologien zur Ermittlung von Bildähnlichkeiten wurden ebenfalls erläutert und in drei Kategorien eingeordnet. Zudem fand abschließend auch eine Betrachtung derzeit existierender Vergleiche dieser Metriken statt. Darin wurde ebenfalls die Notwendigkeit eines fairen Vergleichs aller Verfahren betont.

Im Anschluss erfolgte die Konzeption eines Systems mit dem Namen *Similar Image Matching Suite*, kurz *SIMaSu*. Das System skaliert linear und ermöglicht somit eine parallele Berechnung von Bildähnlichkeiten auf einer verteilten Hardware. Sowohl umfangreiche Evaluationen als auch eine bildbasierte Deduplikation wird mit diesem Konzept umgesetzt. Dabei wurde ebenfalls auf eine unkomplizierte Erweiterbarkeit der verwendeten Ähnlichkeitsmetriken Wert gelegt. Eine prototypische Umsetzung des Systems wurde unter Verwendung von PHP und der Nachrichtenbasierten Technologie RabbitMQ realisiert.

Um die Ähnlichkeit zweier Bilder auch mittels der Feature-basierten Technologien SIFT, SURF und ORB zu ermitteln, wurde eine Metrik konzipiert, welche die Übereinstimmung interessanter Bildregionen auf einen Ähnlichkeitswert abbildet. Bei dem Konzept wurde sich an Eigenschaften einer idealen Metrik orientiert. Eine mittels Java und OpenCV umgesetzte prototypische Implementierung dieser Metrik wurde an das *SIMaSu*-System angebunden, um weiterführende Evaluationen zu ermöglichen.

In den anschließenden drei Evaluationen wurden elf verschiedene Ähnlichkeitsmetriken hinsichtlich ihrer Laufzeit bei unterschiedlichen Bildgrößen, Invarianz gegenüber Transformationen und Effektivität untersucht. Die Feature-basierten Verfahren zeigten im Durchschnitt die höchste Effektivität und Robustheit gegen Transformationen auf, benötigen dabei jedoch deutlich mehr Zeit. Perceptual Hash-Technologien schnitten bzgl. Effektivität und Invarianz schlechter ab, benötigten jedoch nur einen Bruchteil der Berechnungszeit, verglichen mit den Feature-basierten Ansätzen. Aus der Evaluation der Effektivität resultierte je Metrik eine Entscheidungshilfe, welche die Wahl eines geeigneten Grenzwertes für die Klassifikation unterstützt. Besteht ein Anwendungsfall, in dem bestimmte Bildtransformationen (z. B. Spiegelung und platzierte Wasserzeichen) sehr häufig im Datensatz vorkommen, sollten Verfahren gewählt werden, die invariant gegen diese Transformationen sind. Das Ergebnis des Invarianz-Experiments unterstützt eine entsprechende Entscheidung für oder gegen die Wahl eines Verfahrens.

Nach Bearbeitung der Masterarbeit kristallisierten sich weitere lohnende Ansatzpunkte für zukünftige Forschungsprojekte heraus. Neben dem hier konzipierten Deduplikationssystem sollte auch

die Erweiterung einer bestehenden Anwendung, wie bspw. das Framework *FAMER* der Universität Leipzig, zur Unterstützung von Bilddaten als Eingabe-Entitäten in Betracht gezogen werden. Da eine geeignete Kombination mehrerer Ähnlichkeitsmaße in dieser Arbeit nicht evaluiert wurde, stellt dies ebenfalls ein noch ausstehendes Forschungsthema dar. Bezogen auf die Feature-basierten Technologien, existieren neben dem Ratio-Test zur Selektion eindeutiger Treffer noch weitere Ansätze, deren Auswirkungen auf die Effektivität von Interesse wären. Auch die Veränderung des für den Ratio-Test notwendigen Grenzwertes stellt einen Einflussfaktor der Effektivität dar und sollte in zukünftigen Evaluationen beachtet werden.

Eine allgemeingültige Lösung aller definierten Problemstellungen, die sich aus der Deduplikation großer Mengen von Bildern ergeben, existiert derzeit nicht. Die Wahl einer Ähnlichkeitsmetrik, bzw. der Kombination mehrerer Metriken, hängt von dem jeweiligen Anwendungsfall und dessen Anforderungen, sowie den Eigenschaften der Bilder des zu untersuchenden Datensatzes ab. In der Gesamtheit zeigt diese Masterarbeit Stärken und Schwächen einzelner Lösungsansätze und bietet neben einen Einblick in diese Thematik, auch ein Konzept für ein skalierbares bildbasiertes Deduplikationssystem.

Literatur

[1] URL: https://cdn.logitravel.de/microsites/logicms/filer_public/d9/be/d9be052a-14d5-4cde-8604-4192c328aabf/aidastella.jpg (besucht am 18. 06. 2017).

[2] URL: https://upload.wikimedia.org/wikipedia/commons/8/8b/AIDAstella%2C_6.jpg (besucht am 18. 06. 2017).

[3] URL: http://www.hamburg.de/image/4576926/4x3/690/518/6aa9b5f2311cf7b7ebcb04d1726cda78/eS/aidacara.jpg (besucht am 18. 06. 2017).

[4] URL: https://t3.ftcdn.net/jpg/00/98/26/40/240_F_98264017_0xVp4nVN5Eobds3m7dpurwWNtVF2LKkB.jpg (besucht am 18. 06. 2017).

[5] Faraj Alhwarin, Danijela Ristić–Durrant und Axel Gräser. „VF-SIFT: very fast SIFT feature matching“. In: *Pattern Recognition* (2010), S. 222–231.

[6] *Apache Maven*. URL: https://maven.apache.org/download.cgi (besucht am 13. 08. 2017).

[7] Hans-Jochen Bartsch. *Taschenbuch mathematischer Formeln für Ingenieure und Naturwissenschaftler*. Carl Hanser Verlag GmbH Co KG, 2014.

[8] Herbert Bay, Tinne Tuytelaars und Luc Van Gool. „Surf: Speeded up robust features“. In: *Computer vision–ECCV 2006* (2006), S. 404–417.

[9] Omar Benjelloun u. a. „Swoosh: a generic approach to entity resolution“. In: *The VLDB Journal—The International Journal on Very Large Data Bases* 18.1 (2009), S. 255–276.

[10] Murat Birinci u. a. „Neighborhood matching for object recognition algorithms based on local image features“. In: *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*. IEEE. 2011, S. 157–162.

[11] Michael Calonder u. a. „Brief: Binary robust independent elementary features“. In: *Computer Vision–ECCV 2010* (2010), S. 778–792.

[12] Johnny Chien u. a. „When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry“. In: (Nov. 2016), S. 1–6.

[13] *Composer*. URL: https://getcomposer.org/ (besucht am 13. 08. 2017).

[14] Intel Corporation. *Einführung in Big Data: Die Analyse unstrukturierter Daten. Ein Schnellkurs zum iT-umfeld für Big Data und neuen Techniken*. Juni 2012. URL: http://www.intel.de/content/dam/www/public/emea/de/de/pdf/unstructured-data-analytics-paper.pdf.

[15] Oracle Corporation. *MySQL - The world's most popular open source database*. URL: https://www.mysql.com/ (besucht am 07. 12. 2017).

[16] W Davidson und Mortimer Abramowitz. „Molecular expressions microscopy primer: Digital image processing-difference of gaussians edge enhancement algorithm“. In: *Olympus America Inc., and Florida State University* (2006).

[17] *Feature Matching*. URL: https://docs.opencv.org/3.3.0/dc/dc3/tutorial_py_matcher.html (besucht am 13. 08. 2017).

[18] *Feature Matching + Homography to find Objects*. URL: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html (besucht am 13. 11. 2017).

[19] Martin A Fischler und Robert C Bolles. „Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography“. In: *Communications of the ACM* 24.6 (1981), S. 381–395.

[20] John Gantz und David Reinsel. „Extracting value from chaos“. In: *IDC iview* 1142.2011 (2011), S. 1–12.

[21] *Google Bilder*. URL: `https://www.google.de/imghp` (besucht am 13. 08. 2017).

[22] *Grafika: a image processing and graphics library for PHP*. URL: `https://kosinix.github.io/grafika/` (besucht am 13. 08. 2017).

[23] Dimitri Gross. „Maschinelle Bilderkennung mit Big Data und Deep Learning“. In: *Entwickler Magazin. Spezial* 11 (2017), S. 56–61.

[24] Shungang Hua u. a. „Similarity measure for image resizing using SIFT feature“. In: *EURASIP Journal on Image and Video Processing* 2012.1 (2012), S. 6.

[25] *Image Processing (ImageMagick)*. URL: `http://php.net/manual/en/book.imagick.php` (besucht am 13. 08. 2017).

[26] *ImageHash*. URL: `https://github.com/JohannesBuchner/imagehash` (besucht am 13. 08. 2017).

[27] *Image::Hash*. URL: `http://search.cpan.org/~runarb/Image-Hash-0.06/lib/Image/Hash.pm` (besucht am 13. 08. 2017).

[28] *ImageHash*. URL: `https://github.com/jenssegers/imagehash` (besucht am 13. 08. 2017).

[29] *ImageMagick*. URL: `http://www.imagemagick.org/script/index.php` (besucht am 13. 08. 2017).

[30] *Imagick::compareImages*. URL: `http://php.net/manual/en/imagick.compareimages.php` (besucht am 13. 08. 2017).

[31] Hervé Jégou, Matthijs Douze und Cordelia Schmid. „Hamming embedding and weak geometry consistency for large scale image search-extended version“. In: (2008).

[32] George H Joblove und Donald Greenberg. „Color spaces for computer graphics“. In: *ACM siggraph computer graphics*. Bd. 12. 3. ACM. 1978, S. 20–25.

[33] Ebrahim Karami, Siva Prasad und Mohamed Shehata. „Image matching using SIFT, SURF, BRIEF and ORB: Performance comparison for distorted images“. In: *arXiv preprint arXiv:1710.02726* (2017).

[34] Mohammed Lamine Kherfi, Djemel Ziou und Alan Bernardi. „Image retrieval from the world wide web: Issues, techniques, and systems“. In: *ACM Computing Surveys (CSUR)* 36.1 (2004), S. 35–67.

[35] Lars Kolb und Erhard Rahm. „Parallel entity resolution with dedoop“. In: *Datenbank-Spektrum* 13.1 (2013), S. 23–32.

[36] Lars Kolb, Andreas Thor und Erhard Rahm. „Dedoop: Efficient deduplication with hadoop“. In: *Proceedings of the VLDB Endowment* 5.12 (2012), S. 1878–1881.

[37] Hanna Köpcke und Erhard Rahm. „Frameworks for entity matching: A comparison“. In: *Data & Knowledge Engineering* 69.2 (2010), S. 197–210.

[38] Hanna Köpcke, Andreas Thor und Erhard Rahm. „Evaluation of Entity Resolution Approaches on Real-world Match Problems“. In: *Proc. VLDB Endow.* 3.1-2 (Sep. 2010), S. 484–493. ISSN: 2150-8097. DOI: 10.14778/1920841.1920904. URL: http://dx.doi.org/10.14778/1920841.1920904.

[39] David G Lowe. „Distinctive image features from scale-invariant keypoints“. In: *International journal of computer vision* 60.2 (2004), S. 91–110.

[40] David G Lowe. „Object recognition from local scale-invariant features“. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Bd. 2. Ieee. 1999, S. 1150–1157.

[41] Jean-Michel Morel und Guoshen Yu. „ASIFT: A new framework for fully affine invariant image comparison“. In: *SIAM Journal on Imaging Sciences* 2.2 (2009), S. 438–469.

[42] Marius Muja und David G Lowe. „Fast approximate nearest neighbors with automatic algorithm configuration.“ In: *VISAPP (1)* 2.331-340 (2009), S. 2.

[43] *OpenCV*. URL: https://opencv.org/ (besucht am 13. 08. 2017).

[44] *Oracle Java*. URL: http://www.oracle.com/technetwork/java/index.html (besucht am 13. 08. 2017).

[45] *PHP*. URL: http://php.net/ (besucht am 13. 08. 2017).

[46] Jacek Piotrowski. „Top-down approach to image similarity measures“. In: *International Conference on Computer Vision and Graphics*. Springer. 2008, S. 66–69.

[47] Charles Poynton. *Color FAQ*. URL: http://poynton.ca/ColorFAQ.html (besucht am 13. 08. 2017).

[48] Lutz Priese. *Computer Vision: Einführung in Die Verarbeitung und Analyse Digitaler Bilder*. Springer-Verlag, 2015.

[49] *RabbitMQ*. URL: https://www.rabbitmq.com/ (besucht am 13. 08. 2017).

[50] Adrian Rosebrock. *Fingerprinting Images for Near-Duplicate Detection*. 2014. URL: https://realpython.com/blog/python/fingerprinting-images-for-near-duplicate-detection (besucht am 13. 08. 2017).

[51] Edward Rosten und Tom Drummond. „Machine learning for high-speed corner detection“. In: *Computer Vision–ECCV 2006* (2006), S. 430–443.

[52] Ethan Rublee u. a. „ORB: An efficient alternative to SIFT or SURF“. In: *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE. 2011, S. 2564–2571.

[53] Alieh Saeedi, Eric Peukert und Erhard Rahm. „Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution“. In: *Advances in Databases and Information Systems*. Springer. 2017, S. 278–293.

[54] Peter Schmitz. *Was ist ein Hash?* URL: https://www.security-insider.de/was-ist-ein-hash-a-635712/.

[55] Qian Sen und Zhu Jianying. „Improved SIFT-based bidirectional image matching algorithm“. In: *MECHANICAL SCIENCE AND TECHNOLOGY* 26.9 (2007), S. 1179.

[56] Arnold WM Smeulders u. a. „Content-based image retrieval at the end of the early years“. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.12 (2000), S. 1349–1380.

[57] *Symfony*. URL: https://symfony.com/ (besucht am 13. 08. 2017).

[58] *Tineye MatchEngine*. URL: https://services.tineye.com/MatchEngine (besucht am 13. 08. 2017).

[59] Paul Viola und Michael Jones. „Rapid object detection using a boosted cascade of simple features“. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Bd. 1. IEEE. 2001, S. I–I.

[60] Fred Weinhaus. *Tests Of Perceptual Hash (PHASH) Compare Metric*. URL: http://www.fmwconcepts.com/misc_tests/perceptual_hash_test_results_510/ (besucht am 07. 12. 2017).

[61] Spektrum der Wissenschaft Verlagsgesellschaft mbH. *digitales Bild*. URL: http://www.spektrum.de/lexikon/kartographie-geomatik/digitales-bild/984 (besucht am 13. 08. 2017).

[62] Jian Wu u. a. „A Comparative Study of SIFT and its Variants“. In: *Measurement Science Review* 13.3 (2013), S. 122–131.

[63] Christoph Zauner. „Implementation and benchmarking of perceptual image hash functions“. In: (2010).

A. Anhang

A.1. CD

Auf der CD befinden sich das PHP-Projekt des SIMaSu-Prototypen sowie das Java-Projekt der Feature-basierten Ähnlichkeitsmetrik als Quellcode und ausführbares Java-Archiv (jar).

A.2. Diagramme

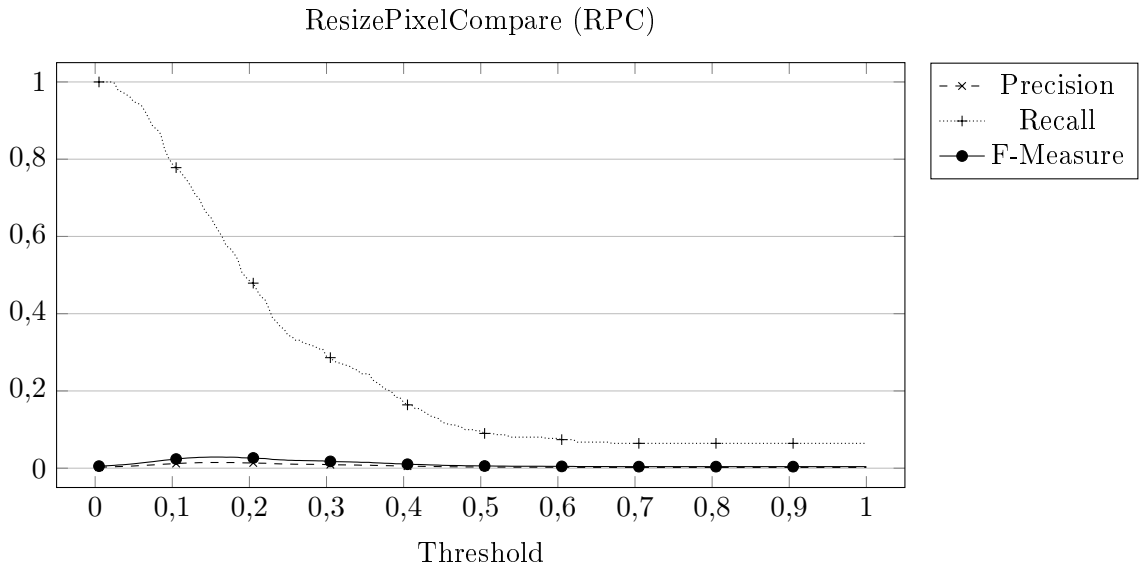


Abbildung 23: Effektivitätsmaße der *ResizePixelCompare*-Metrik (RPC)

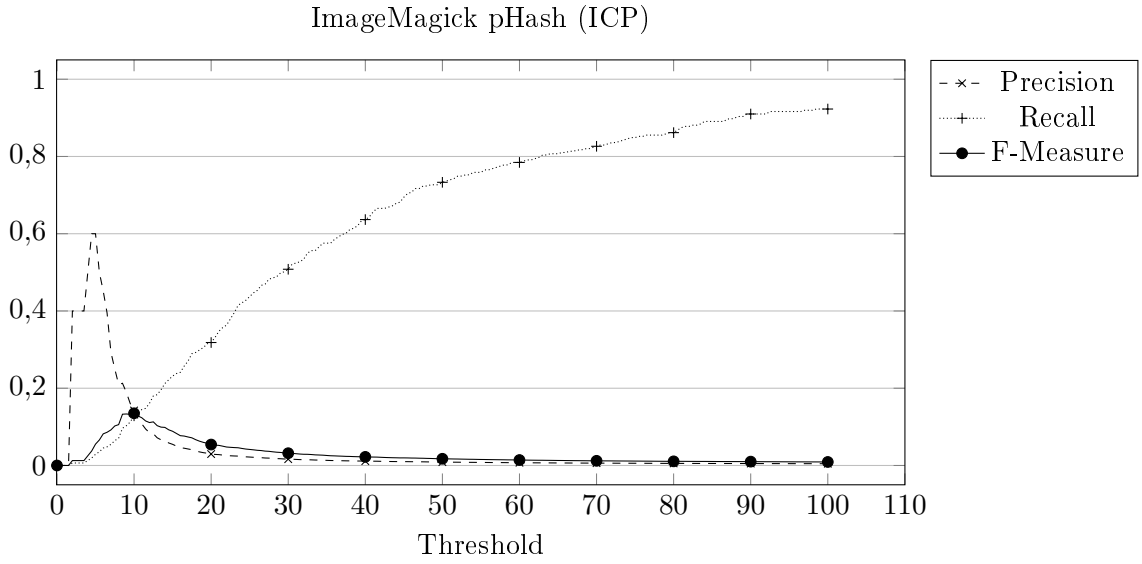


Abbildung 24: Effektivitätsmaße der *pHash*-Metrik der Anwendung *ImageMagick* (ICP)

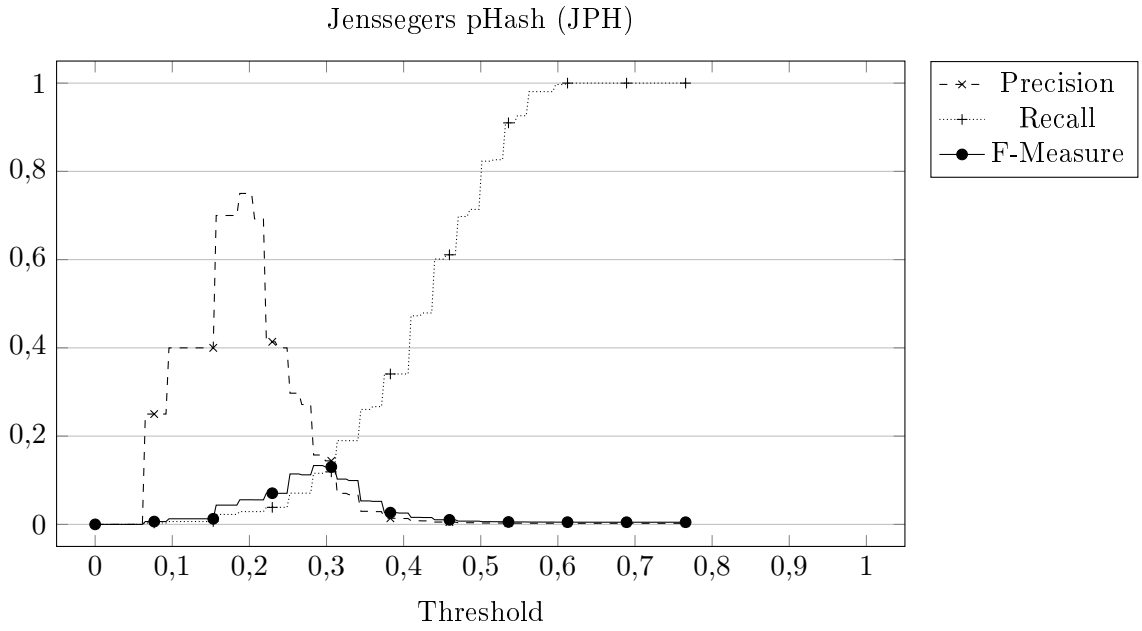


Abbildung 25: Effektivitätsmaße der *pHash*-Metrik von *Jenssegers* (JPH)

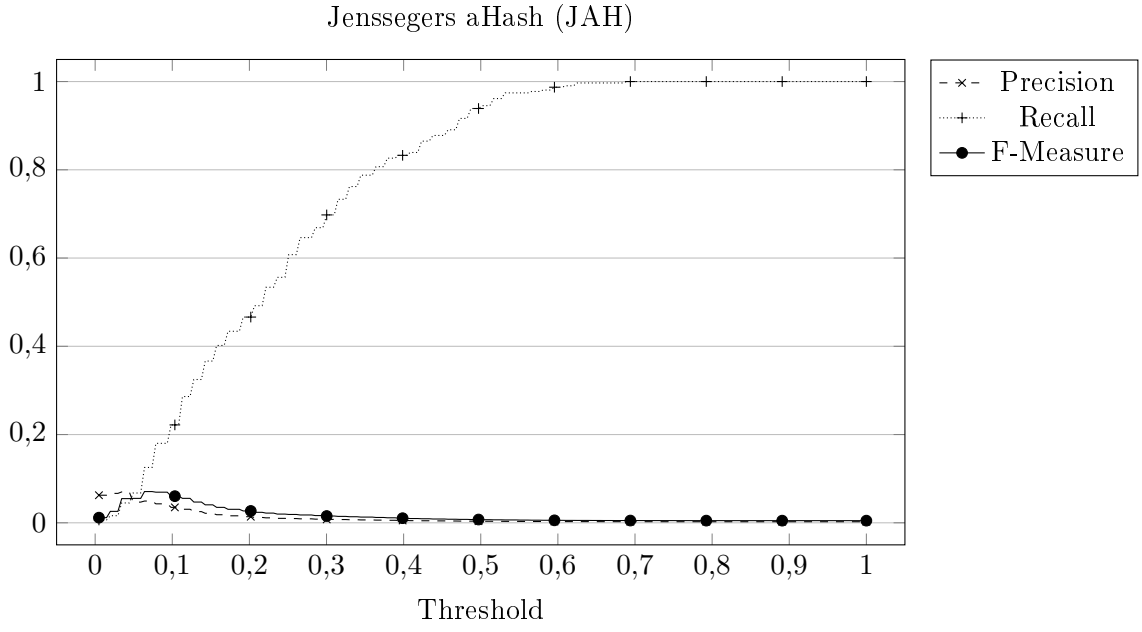


Abbildung 26: Effektivitätsmaße der *aHash*-Metrik von *Jenssegers* (JAH)

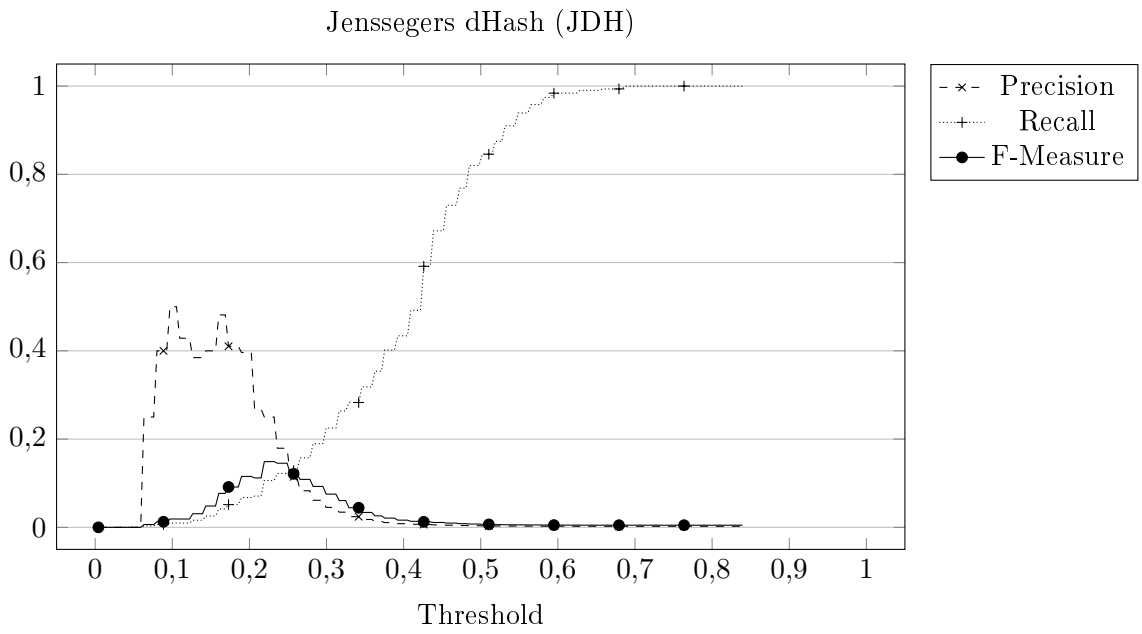


Abbildung 27: Effektivitätsmaße der *dHash*-Metrik von *Jenssegers* (JDH)

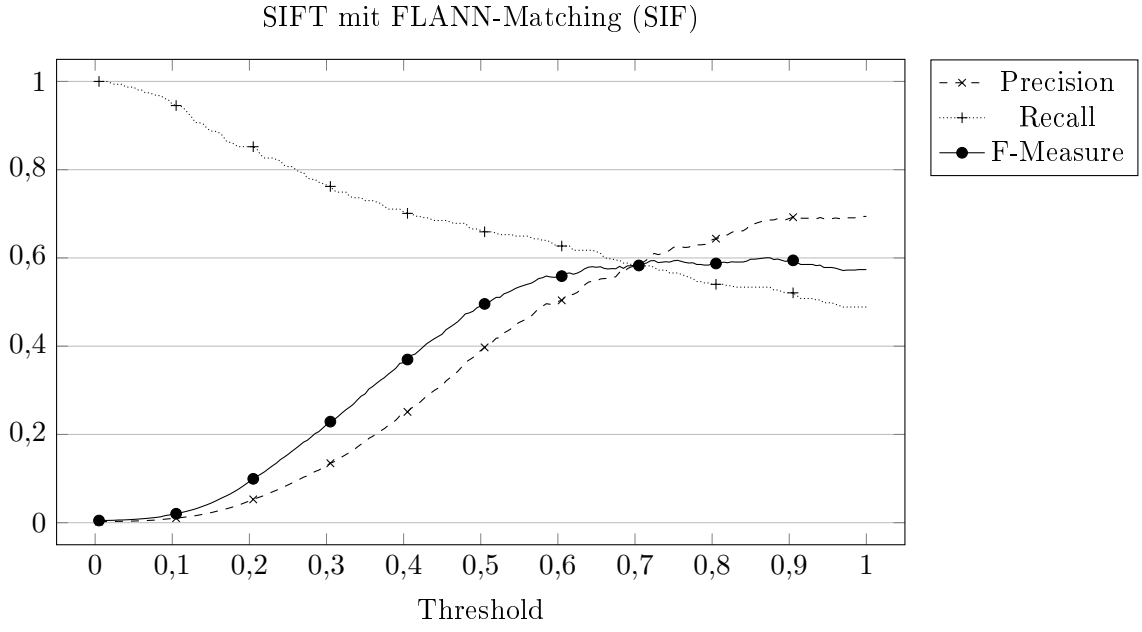


Abbildung 28: Effektivitätsmaße der Feature-basierten Metrik *SIFT* mit *FLANN-Matching* (SIF)

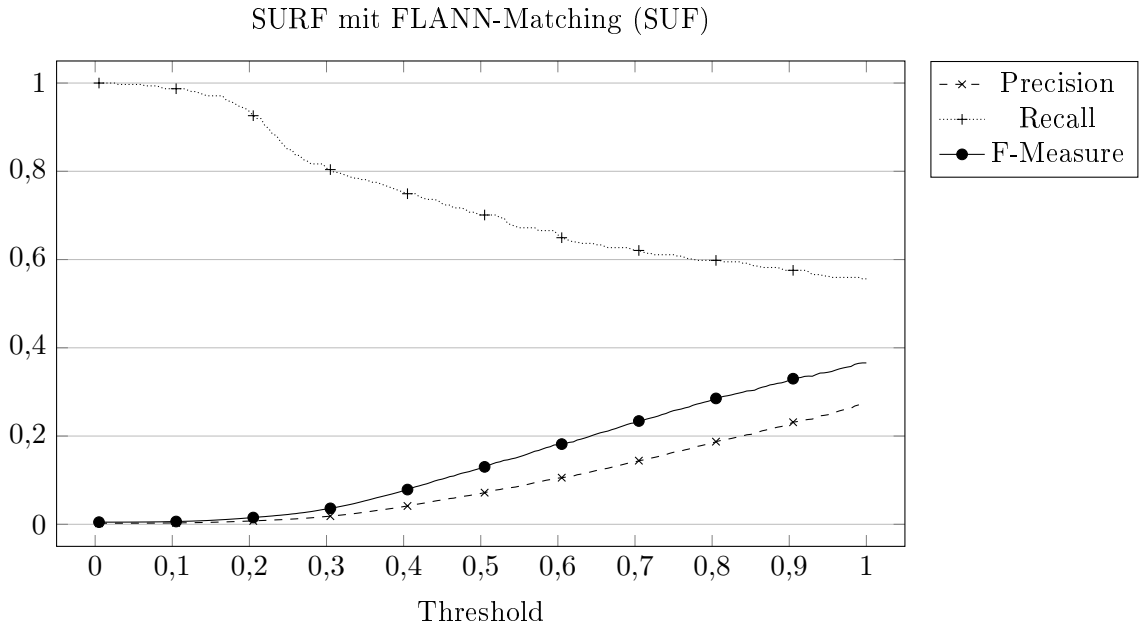


Abbildung 29: Effektivitätsmaße der Feature-basierten Metrik *SURF* mit *FLANN-Matching* (SUF)

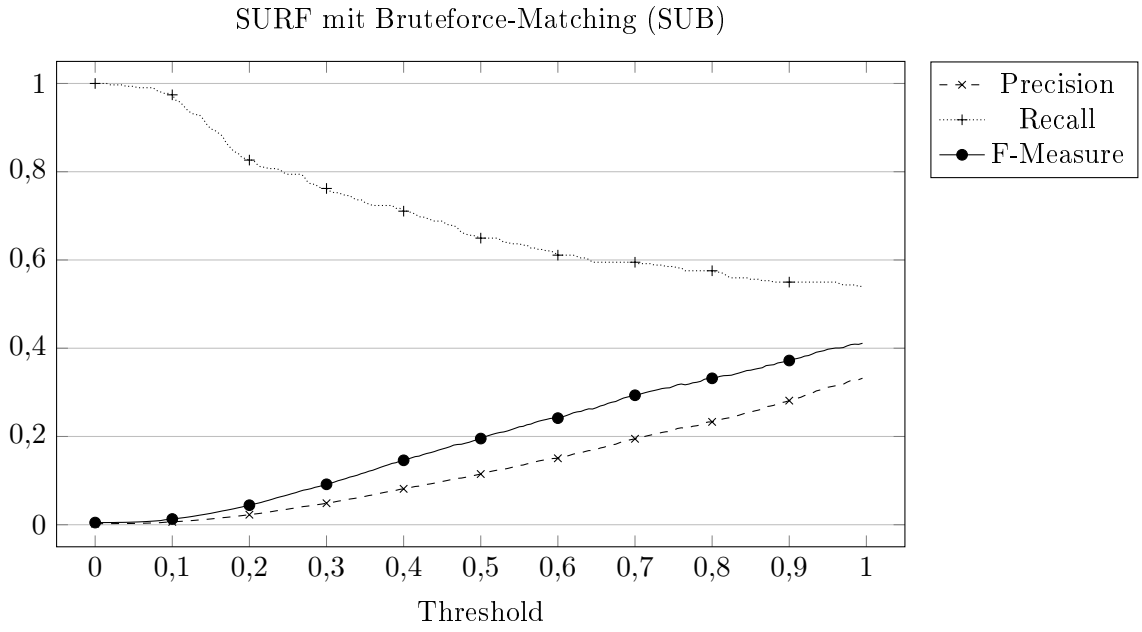


Abbildung 30: Effektivitätsmaße der Feature-basierten Metrik *SURF* mit *Bruteforce-Matching* (SUB)

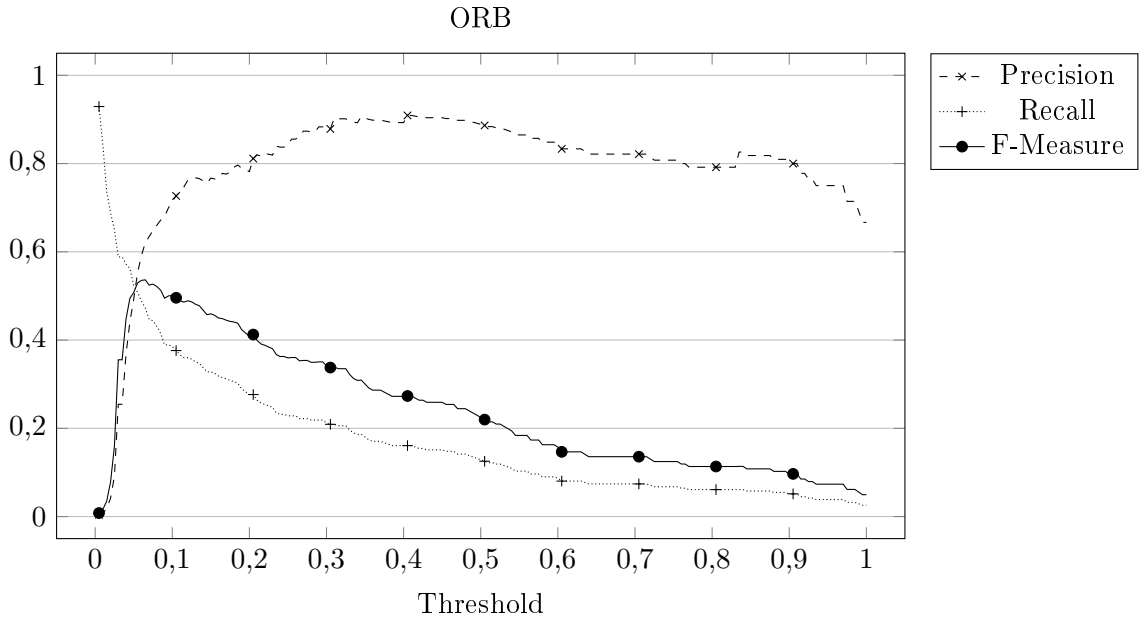


Abbildung 31: Effektivitätsmaße der Feature-basierten Metrik *ORB*

Selbstständigkeitserklärung

Hiermit erkläre ich, Christopher Rost, geboren am 06.09.1990 in Querfurt, dass ich die von mir an der Universität Leipzig eingereichte Abschlussarbeit zum Thema

Skalierbare bildbasierte Deduplikation

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Leipzig, den 21. Dezember 2017

Christopher Rost