HTWK

Hochschule für Technik,
Wirtschaft und Kultur Leipzig

Bachelorarbeit
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

im Bachelorstudiengang Medieninformatik
der Fakultät Informatik und Medien

Entwicklung eines Bilderkennungssystems zum finden von Duplikaten in einer Bilddatenbank bei der Registrierung neuer Bilder

vorgelegt von
Martino Thomann

Leipzig, den 26. September 2023

Erstprüfer:	Prof. Dr. Sibylle Schwarz
Zweitprüfer:	B.Sc. Marc Bellmann

Zusammenfassung

In dieser Arbeit wird der Entwurf eines Bilderkennungssystems beschrieben. Das System ist in der Lage zu erkennen, ob es sich bei einem neuen Bild um Duplikat handelt, das bereits in einer Datenbank mit registrierten Bildern vorhanden ist. Um Duplikate zu erkennen, deren Motiv im Vergleich zum Original modifiziert, aber immer noch als Kopie erkennbar sind, wird ein feature-based Bilderkennungs-Algorithmus eingesetzt. Innerhalb der wurden SIFT, ORB und BRISK miteinander im Anwendungsfall von T-Shirt Designs bei Spreadshirt innerhalb von verschiedenen Szenarien getestet und miteinander verglichen.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Spread Group bietet Nutzern über ihre Plattformen die Möglichkeit personalisierte Produkte zu kaufen. Dabei können Nutzer auch eigene Designs hochladen, um diese beispielsweise auf selbst erstellten Produkten zu verkaufen. Täglich werden tausende solcher selbst erstellter Designs hochgeladen. Dabei muss seitens der Spread Group sichergestellt werden, dass die hochgeladenen Designs nicht gegen das Gesetz oder die unternehmensinternen Richtlinien verstoßen. Die manuelle Überprüfung von so vielen Designs ist sehr aufwändig und häufig kommt es vor, dass die gleichen Designs mehrmals hochgeladen werden. Eine automatische Sperrung, von bereits abgelehnten Designs, die erneut hochgeladen werden, kann den Überprüfungsprozess entlasten. Daher ist ein System notwendig, dass bei neu hochgeladenen Designs Duplikate in der Datenbank der abgelehnten Designs erkennen kann.

1.2 Aufgabenstellung

Ziel dieser Bachelorarbeit ist die Entwicklung eines Systems zur Identifikation von Duplikaten in einer Datenbank. Das System soll dazu in der Lage sein, neue Bilder mit Bildern in der Datenbank zu vergleichen und festzustellen, ob diese identisch oder ähnlich sind. Ein neues Bild soll auch dann als Duplikat erkannt werden, wenn der Bildinhalt im Vergleich zum Original transformiert, also rotiert, skaliert, verschoben oder gespiegelt wurde. Fälle in denen das Bild anders gefärbt ist oder als Teil eines größeren Bildes auftaucht, sollen ebenfalls berücksichtigt werden.

Zur Implementierung des Systems soll ein feature-based (dt. merkmalbasierter) Algorithmus verwendet werden. Es gibt eine Vielzahl an feature-based Algorithmen, die jeweils ihre eigenen Stärken und Schwächen haben. Eine Auswahl dieser Algorithmen sollen innerhalb der Arbeit für den Einsatz in der Bildakkreditierung bei Spread Group getestet und miteinander verglichen werden.

1.3 Erfolgs- und Qualitätskriterien

Die Güte des Bilderkennungssystems soll anhand von einem Testdatensatz ermittelt werden. Der Testdatensatz ist dabei in zwei Teilsätze unterteilt. Der erste Teilsatz an Bildern stellt die Menge an gespeicherten Datenbankbildern dar. Der zweite Teilsatz enthält eine Untermenge an Duplikaten aus dem Datenbanksatz und eine Menge an neuen Bildern, die nicht im Datenbanksatz auftauchen. Getestet wird in verschiedenen Szenarien, die die Robustheit des Systems gegenüber bestimmter Sonderfälle testen sollen. Je nach Szenario sind die Duplikate auf unterschiedliche Weise im Vergleich zum Original verändert. Als Vergleich dient der pHash-Algorithmus, der momentan in der Sprad Group zur Duplikatensuche verwendet wird.

Die verwendeten Metriken werden in den Grundlagen 2.3 erklärt. Am wichtigsten ist dabei ein hoher Recall, sodass möglichst viele Duplikate durch das System abgefangen werden. Da automatisch gesperrte Designs nochmal manuell geprüft werden, fällt eine niedrigere Spezifizität bei der Auswertung nicht so sehr ins Gewicht. Laufzeit- und Speicherkosten sollen ebenfalls innerhalb der Arbeit abgeschätzt werden.

Um für die Verwendung in Frage zu kommen, muss das neue System Duplikate zuverlässiger erkennen können, als die momentan eingesetzte pHash-Implementation. Dafür wird ein höherer Recall angestrebt. Um sicherzustellen, dass Spezifizität nicht zu sehr absinkt, wird auf eine höhere oder zumindest gleichbleibende Balancierte-Genauigkeit abgezielt.

2 Grundlagen

Im folgenden Kapitel werden die Grundlagen erklärt, die für das Verständnis der Arbeit nötig sind. Zuerst werden in Abschnitt 2.1 die feature-based Algorithmen erklärt, die in der Arbeit verglichen werden sollen. Danach wird im Abschnitt 2.2 darauf eingegangen, wie die Deskriptoren, die aus den feature-based Algorithmen hervorgehen, verglichen werden können. Zuletzt beschäftigt sich Abschnitt 2.3 mit den Metriken, die für die Evaluation der Algorithmen eingesetzt werden.

2.1 feature-based Algorithmen

Ziel der feature-based (dt. Merkmalbasierten) Algorithmen ist es markante Punkte innerhalb von Bildern zu finden und zu beschreiben. Die Algorithmen bestehen dabei im Grunde aus zwei Schritten:

1. Die Schlüsselpunktsuche, bei der nach Koordinaten innerhalb eines Bildes gesucht wird, an denen sich markante Punkte befinden.

2. Die Erstellung von Deskriptoren, die den Bereich um die Schlüsselpunkte beschreiben. Diese sollen später mit Deskriptoren aus anderen Bildern verglichen werden, um gemeinsame Merkmale zu finden.

Wie genau diese beiden Schritte implementiert sind ist je nach Algorithmus unterschiedlich.

Meistens handelt es sich bei den gefundenen Merkmalen um Rand- und Eckpunkte oder Details auf einer Fläche.

2.1.1 SIFT: Scale-Invariant Feature Transform

Ziel des SIFT-Algorithmus ist es Merkmale in einem Bild zu finden und mit Deskriptoren zu beschreiben, die Robust gegenüber Rotation, Verschiebung und Skalierung sind. Auch der Einfluss durch Bildrauschen und Verzerrung soll möglichst gering sein. Dabei durchläuft der Algorithmus 4 Schritte. Schritt 1 bis 3 umfassen die Suche und Beschreibung der Schlüsselpunkte, während Schritt 4 sich mit Erstellung der Deskriptoren befasst. [Low99, S. 2]

1. Suche nach Extrempunkten SIFT sucht nach Extrempunkten innerhalb von Bildern. Extrempunkte sind dabei Punkte, an denen lokale Maxima oder Minima in der Helligkeit auftreten. Also helle Punkte innerhalb von dunklen Bereichen und dunkle Punkte innerhalb von hellen Bereichen.

Um Extrempunkte zu finden wird der Difference of Gaussian (kurz DOG) genutzt. Zuerst wird eine bestimmte Anzahl an Bildern, durch das Weichzeichnen des Originalbilds mit dem Gaußschen Weichzeichner, erstellt. Der Grad der Unschärfe steigt dabei bei jedem Bild um einen konstanten Faktor. Ein Satz dieser weichgezeichneten Bilder wird als Oktave bezeichnet. Für jedes paar an aneinander grenzenden weichgezeichneten Bildern, wird die Differenz berechnet. Bei dieser Differenz handelt es sich um den DOG. Durch den DOG werden Rand- und Eckpunkte, sowie andere Details im Bild hervorgehoben. Bei der Implementation von SIFT werden pro Oktave meist fünf weichgezeichnete Bilder verwendet, aus denen vier DOG Bilder entstehen. [Low04, S. 94]

Jeder Pixel der mittleren DOG-Bilder wird mit seinen acht Nachbarn innerhalb desselben Bildes und seinen je neun Nachbarn im DOG-Bild darüber und darunter verglichen. Ein Pixel ist dann ein Schlüsselpunkt, wenn sein Helligkeitswert größer oder kleiner als der Wert all seiner Nachbarn ist. [Low04, S. 95]

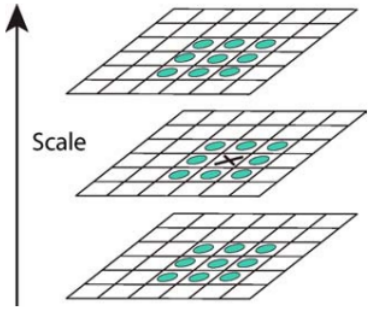

Abbildung 2.1: Vergleich des Schlüsselpunkt-Kandidats mit seinen 26 Nachbarpixeln aus [Low04, S. 95]

Diese Extrempunktsuche wird dabei für mehrere Oktaven durchgeführt. Dabei wird für jede Oktave das Originalbild um den Faktor 2 herunter skaliert. Das ist notwendig, da manche Extrempunkte nur bei bestimmten Auflösungen auffindbar sind. [Low04, S. 94f] Durch die Sammlung von Schlüsselpunkten auf verschiedenen Auflösungen, sind diese bei Bildern in verschiedenen Auflösungen vergleichbar. Der Aufbau von DOG-Bilder innerhalb von mehreren Oktaven wird als scale-space (dt. Bildpyramide) bezeichnet (Siehe Abbildung 2.2).

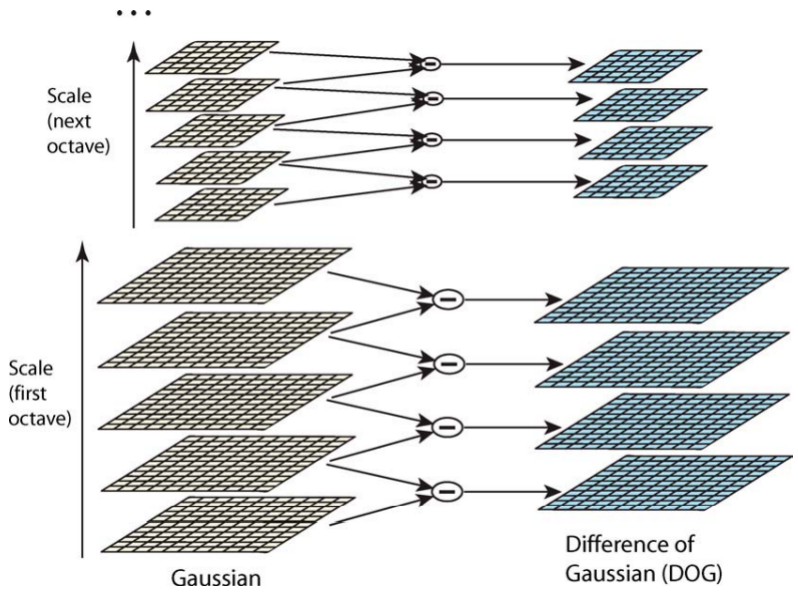

Abbildung 2.2: Bildpyramide aus [Low04, S. 95]

2. Akkurate Ortsbestimmung der Schlüsselpunkte Die Positionen der Schlüsselpunkte, die auf niedrigen Auflösungen gefunden wurden, entspricht gegebenenfalls nicht genau der Position in der Originalauflösung. Deshalb wird die Taylorreihe verwendet, um die tatsächliche Position zu interpolieren. [Low04, S. 97f]

Außerdem werden Schlüsselpunkte an Rändern und Schlüsselpunkte mit niedrigem Kontrast zu ihrer Umgebung herausgefiltert. Das ist nötig, da solche Schlüsselpunkte Deskriptoren hervorbringen, die nicht zuverlässig verglichen werden können. [Low04, 98f]

3. Bestimmung der Schlüsselpunkt Ausrichtung Um Schlüsselpunkte und die aus ihnen resultierenden Deskriptoren robust gegenüber Bildrotationen zu machen, wird jedem Schlüsselpunkt eine Ausrichtung zugewiesen. Damit diese Ausrichtung konsistent ist, wird diese anhand der lokalen Bildeigenschaften um den Schlüsselpunkt herum bestimmt. [Low04, S. 99]

Um das zu erreichen, wird ein 360 Grad umfassendes Histogramm mit 36 bins aufgebaut. Anhand der Skalierung des Schlüsselpunkts wird ein weichgezeichnetes Bild mit passender Skalierung gewählt. Innerhalb dieses Bildes werden Abtastpunkte um den Schlüsselpunkt herum genommen. Für jeden Abtastpunkt wird die Orientierung und Stärke des Gradienten bestimmt. Der Gradient beschreibt dabei die Änderung der Pixelhelligkeit innerhalb eines Bildes. Sowohl Stärke als auch Orientierung des Gradienten sind für alle Ebenen der Bildpyramide vorberechnet worden. [Low04, S. 99]

Jede Gradient-Orientierung wird in das Histogramm eingetragen und anhand der Gradient-Stärke gewichtet. Da das Histogramm 36 bins hat, werden Orientierungen in 10-Grad-Schritten zusammengefasst. Aus dem Histogramm wird die Orientierung des globalen Maximums, sowie die Orientierungen aller lokalen Maxima die mindestens 80 Prozent des globalen Maximums betragen, entnommen. Für jede entnommene Orientierung wird ein Schlüsselpunkt erstellt. Dadurch kann es mehrere Schlüsselpunkte mit der gleicher Position und Skalierung aber mit unterschiedlichen Orientierungen geben. [Low04, S. 100]

4. Deskriptoren erstellen In dem Bereich um den Schlüsselpunkt werden wie im Schritt 3 (Abschnitt 2.1.1) für jeden Abtastpunkt die Gradienten-Stärke und -Orientierung bestimmt. Der Bereich wird dabei in 4x4 große Unterbereiche aufgeteilt, für die ein Histogramm erstellt wird. Dabei werden für das Histogramm nur 8 bins verwendet, in denen Gradienten-Stärken mit ähnlicher Orientierung aufsummiert werden. Aus diesen Werten wird ein Deskriptor erstellt. Üblicherweise werden Abtastbereiche mit einer Größe von 16x16 verwendet. Daraus ergeben sich insgesamt 16 Unterbereiche der Größe 4x4 mit je 8 Orientierungs-bins. Ein typischer Deskriptor ist ein Vektor mit 16 * 8 = 128 Werten. [Low04, S. 101]

2.1.2 ORB: Oriented FAST and Rotated BRIEF

ORB basiert auf dem FAST-Algorithmus für die Schlüsselpunktsuche und dem BRIEF-Algorithmus für die Generierung von Deskriptoren

Schüsselpunktsuche mit FAST Der FAST-Algorithmus sucht nach Eckpunkten in einem Bild. Dazu wird um jeden Pixel im Bild ein Kreis mit 16 Randpixeln gezogen. Der betrachtete Kandidat-Pixel ist genau dann ein Eckpunkt, wenn eine bestimmte Anzahl an aneinander grenzenden Randpixel heller oder dunkler sind als der Kandidat-Pixel plus oder minus einem bestimmten Schwellenwert. [RD06, S. 4]

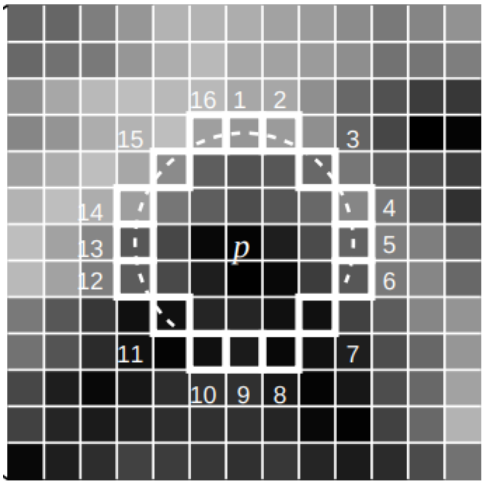

Abbildung 2.3: Randpixel um Kandidat-Pixel aus [RD06, S. 4]

Bei der Implementation in ORB müssen 9 der 16 Randpunkte heller oder dunkler als der Kandidat-Pixel sein, damit dieser sich als Randpunkt qualifiziert. Zudem wird die "Harris corner measure"genutzt, um Randpunkte herauszufiltern. Dazu wird jedem Schlüsselpunkt ein Wert zugeteilt, der aussagt, wie wahrscheinlich es ist, dass es sich bei dem Schlüsselpunkt um einen Eckpunkt handelt. Der Schwellenwert, um sich als Eckpunkt zu qualifizieren, wird niedrig genug gesetzt um eine bestimmte Mindestanzahl an Schlüsselpunkten zu erreichen. Die Mindestanzahl an Schlüsselpunkten, kann dabei je nach Anwendungsfall variieren. Es muss genügend Schlüsselpunkte geben, um das Bild zu beschreiben. Gleichzeitig möchte man aber Schlüsselpunkte vermeiden deren Deskriptoren nicht zuverlässig verglichen werden können. [RRKB11, S. 2565]

Die durch FAST gefundenen Schlüsselpunkte sind anders als bei SIFT noch nicht Robust gegenüber Rotations- und Auflösungsunterschieden. Um Invarianz gegenüber Auflösungsunterschieden entgegenzuwirken, generiert ORB mehrere niedriger aufgelöste Varianten des Originalbilds. Für jede Auflösung wird die Schlüsselpunktsuche mit FAST und Randpunktfilterung mit "Harris corner measure"durchgeführt. Um die Orientierung des Schlüsselpunkts zu ermitteln, wird in dessen Umgebung nach dem Pixel mit dem höchsten Helligkeitswert gesucht. Die Orientierung ergibt sich dann aus dem Vektor zwischen diesen beiden Punkten. [RRKB11, S. 2565]

Erstellen der Deskriptoren mit BRIEF Der BRIEF-Algorithmus definiert um den Schlüsselpunkt einen Bereich p, der weichgezeichnet wird. Aus diesem Bereich werden n Pixelpaare (x, y) genommen und einem Binärtest τ unterzogen. Für jedes Pixelpaar werden die Helligkeitswerte $p(x)$ und $p(y)$ verglichen. Je nachdem ob der Helligkeitswert $p(x)$ größer als $p(y)$ ist, ergibt der Binärtest τ 0 oder 1. In der Implementation bei

ORB werden 256 Pixelpaare genutzt. Der Deskriptor ist eine Aneinanderreihung der Testergebnisse aller 256 Pixelpaare. Somit ergibt sich ein 256 Bit Deskriptor pro gefundenen Schlüsselpunkt. [RRKB11, S. 2566]

$$\tau(p; x, y) := \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases}$$

$$f_n(p) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i, y_i)$$

Damit die durch BRIEF erstellten Deskriptoren robust gegenüber Rotationen sind, nutzt ORB die Orientierung, die für die Schlüsselpunkte ermittelt wurden. Dazu wird eine Matrix S mit den Pixelpaaren für den Binärtest erstellt. Gemäß der Orientierung θ des Schlüsselpunkts wird eine Rotationsmatrix R_θ erstellt. Matrix S wird mit der Rotationsmatrix R_θ rotiert und ergibt S_θ. Der Deskriptor für den Schlüsselpunkt wird anhand der rotierten Pixelpaare aus S_θ erstellt. [RRKB11, S. 2566]

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S_\theta$$

2.1.3 BRISK: Binary Robust Invariant Scalable Keypoints

Schlüsselpunktsuche Ähnlich wie in SIFT wird bei BRISK eine Bildpyramide mit Oktaven aufgebaut. Auch hier wird die Auflösung jeder folgenden Oktave halbiert. Allerdings werden, anders als bei SIFT, keine weichgezeichneten Varianten zur Berechnung des DOG in den einzelnen Oktaven erstellt. Stattdessen besteht jede Oktave aus einen einzelnen Bild und es werden Intra-Oktaven zwischen den Oktaven berechnet. Üblicherweise werden 4 Oktaven und 4 Intra-Oktaven genutzt. [LCS11, S.2549]

Die erste Intra-Oktave wird erstellt, indem die Auflösung des Originalbilds um den Faktor 1.5 reduziert wird. Die Auflösung der ersten Intra-Oktave beträgt dann also 2/3 der Auflösung des Originalbilds. Alle weiteren Intra-Oktaven leiten sich aus der ersten ab, indem die Auflösung für alle weiteren Intra-Oktaven halbiert wird. Ziel der Bildpyramide ist es, genau wie bei SIFT, durch Suchen auf verschiedenen Auflösungsebenen Schlüsselpunkte zu finden und Deskriptoren zu berechnen, die Robust gegenüber Änderungen in der Auflösung sind. [LCS11, S.2549f]

BRISK nutzt AGAST für die Implementierung der Schlüsselpunktsuche. [LCS11, S. 2552] AGAST ist ein Algorithmus zum finden von Eckpunkten, der ähnlich wie FAST (siehe

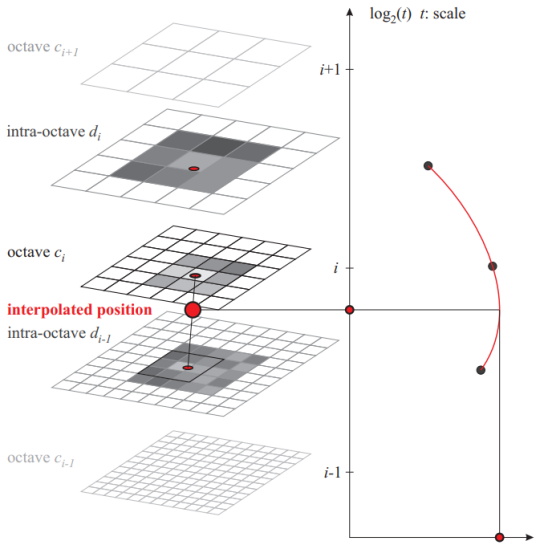

ABBILDUNG 2.4: Interpolation der Schlüsselpunktauflösung in der Bildpyramide aus [LCS11, S. 2550]

Abschnitt 2.1.2) funktioniert. Für jede Oktave und Intra-Oktave wird AGAST angewendet um potenzielle Schlüsselpunkte zu finden. Danach wird für den Schlüsselpunktpixel, sowie allen 8 benachbarten Pixeln der *saliency-score* berechnet. Dieser score gibt an für welchen maximalen Schwellenwert sich der Pixel noch als Eckpunkt qualifiziert. Der *saliency-scores* muss für den Schlüsselpunkt-Kandidaten größer sein als der all seiner Nachbarn. Danach werden auch die scores in dem gleichen Bereich für die beiden angrenzenden (Intra-)Oktaven berechnet. Anhand der drei *saliency-scores*, die in der Regel unterschiedliche Werte haben, wird eine Parabel gebaut. Das Ziel ist, mithilfe der Parabel, die Auflösungsebene zu interpolieren, in der der Schlüsselpunkt den höchsten *saliency-score* erreicht. Diese Auflösungsebene, kann sich dabei zwischen den Oktaven befinden. Dadurch erhält man, die Position und die Skalierung, des Schlüsselpunkts. [LCS11, S. 2550]

Erstellen des Deskriptors BRISK nutzt ein vordefiniertes Muster, um Positionen für Abtastpunkte zu definieren. Die Abtastpunkte sind in konzentrischen Kreisen um den Schlüsselpunkt herum positioniert. Auf die Bereiche um die Abtastpunkte herum wird der Gaußsche Weichzeichner angewendet. Außerdem wird das Muster an die Position und Skalierung des Schlüsselpunkts angepasst. Insgesamt werden 60 Abtastpunkte definiert, die miteinander gepaart werden.[LCS11, S. 2551]

Abhängig von der Skalierung des Schlüsselpunkts werden Schwellenwert-Distanzen bestimmt, mit denen die Paare in Langdistanzpaare L und Kurzdistanzpaare S unterschiedenen werden. Um die Orientierung des Schlüsselpunkts zu ermitteln wird folgende Formel auf die Langdistanzpaare angewendet:

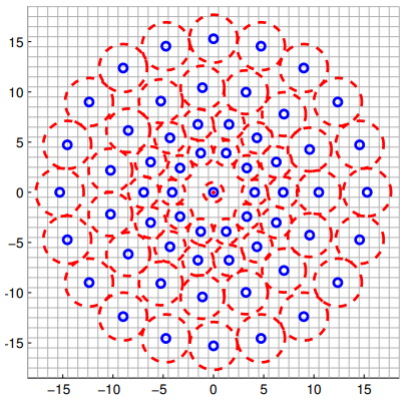

Abbildung 2.5: Vordefiniertes Abtastmuster mit Abtastpunkten in blau und weichgezeichnete Bereiche in rot aus [LCS11, S. 2551]

$$\begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{|L|} * \sum_{(p_i,p_j)\in L} \mathbf{g}(p_i, p_j)$$

$\mathbf{g}(p_i, p_j)$ ist dabei der Gradient der Helligkeitswerte zwischen der Abtastpunkten p_i und p_j einer Paarung. Der Winkel ergibt sich dann aus $\alpha = arctan2(g_y, g_x)$. Es werden nur die Langdistanzpaarungen für die Berechnung der Orientierung verwendet, da davon ausgegangen wird, dass sich die Gradienten der Kurzdistanzpaarungen gegenseitig aufheben würden. [LCS11, S. 2551]

Damit der Deskriptor Robust gegenüber Rotationen ist, wird das Abtastmuster mit den Winkel α um den Schlüsselpunkt rotiert. Um den Deskriptor zu erstellen werden für alle Kurdistanzpaarungen auf dem rotierten Abtastmuster die Helligkeitswerte der beiden Abtastpunkte miteinander verglichen. Je nachdem welcher Helligkeitswert größer ist, wird ein Bit in einer Bitfolge auf 0 oder 1 gesetzt. Diese Bitfolge stellt den Deskriptor für BRISK dar. Insgesamt hat die Bitfolge 512 Stellen, woraus sich ein 512 Bit Deskriptor für jeden Schlüsselpunkt ergibt. [LCS11, S. 2551]

2.2 Match-Suche mit Deskriptoren

Im Abschnitt 2.1 wurden die feature-based Algorithmen erklärt, mit denen Deskriptoren für Merkmale in einem Bild generiert werden können. In diesem Abschnitt soll es darum gehen, wie diese Deskriptoren genutzt werden können, um die Merkmale aus zwei Bildern zu vergleichen und Übereinstimmungen zu finden. Übereinstimmungen werden auch als Match bezeichnet.

2.2.1 Deskriptoren Vergleichen

Um zwei Deskriptoren zu vergleichen, wird die Distanz zwischen den beiden berechnet. Die Distanz beschreibt in diesem Fall wie ähnlich sich Deskriptoren sind. Je kleiner die Distanz desto ähnlicher die Deskriptoren. Abhängig davon welche Art Deskriptor verglichen wird, wird ein anderer Distanzmaßstab verwendet.

Euklidische-Distanz Bei Deskriptoren, die aus dem SIFT-Algorithmus resultieren, wird die Euklidische Distanz genutzt. [Low04, S. 104] Die Deskriptoren sind hierbei eine Folge von Gradient-Werten. Stelle für Stelle wird die Euklidische Distanz für alle Gradient-Werte der Deskriptoren berechnet und aufsummiert.

$$distance(desc1, desc2) := \sum_{i=1}^{128} \sqrt{(desc1[i] - desc2[i])^2}$$

SIFT-Deskriptoren bestehen in der Regel aus 128 Gradient-Werten.

Hamming-Distanz Die Hamming-Distanz kann verwendet werden, um Bitfolgen zu vergleichen. Dazu werden beide Bitfolgen Stelle für Stelle verglichen. Für jede Stelle, die nicht übereinstimmt, steigt die Distanz um 1.

$$distance(desc1, desc2) := |\{i \in \{1, ..., 256\} \mid desc1_i \neq desc2_i\}|$$

Die Hamming-Distanz eignet sich um Deskriptoren, die durch den ORB- oder BRISK-Algorithmus errechnet werden, zu vergleichen. [RRKB11, S. 2569] Bei diesen handelt es sich um Bitfolgen mit 256 Stellen bei ORB und 512 Stellen bei BRISK. Auch bei dem Vergleich von pHash-Werten kommt die Hamming-Distanz zum Einsatz.

Durch bitweises XOR kann die Hamming-Distanz deutlich effizienter berechnet werden, als die Euklidische Distanz. Binär-Deskriptoren haben dadurch beim Vergleichen einen Geschwindigkeitsvorteil.

2.2.2 Matching Strategien

Brute-Force Matcher Um Übereinstimmungen von Merkmalen in zwei Bildern zu finden, werden alle Deskriptoren des ersten Bilds mit allen Deskriptoren des zweiten Bilds verglichen. [ope]

Der Vorteil dieser simplen Herangehensweise ist, dass garantiert die besten Matches gefunden werden. Allerdings ist diese Strategie mit $\mathcal{O}(n^2)$ auch sehr ineffizient, wodurch sie gerade bei Datensets mit einer großen Menge an Deskriptoren ins Gewicht fällt.

FLANN-Based Matcher FLANN ist eine Bibliothek für fast approximate nearest neighbor search. Abhängig vom verwendeten Datensatz wählt die Bibliothek automatisch aus einer Reihe von Algorithmen, die für approximate nearest neighbor Suche optimiert sind, den besten aus. FLANN kann für den schnellen Vergleich von Deskriptoren eingesetzt werden.

Für größere Datensets arbeitet FLANN schneller als der Brute-Force Matcher. Allerdings kann durch die approximate nearest neighbor search nicht garantiert werden, dass FLANN die besten Matches findet. [ope]

2.2.3 Filterung der Matching-Ergebnisse

Für jeden Deskriptor aus dem ersten Bild wird der erst- und zweitbeste Match aus dem zweiten Bild gespeichert. Ein Match besteht dabei aus den beiden Deskriptoren, die verglichen werden, und die resultierende Distanz (Siehe Abschnit 2.2.1). Daraus ergibt sich eine 2xN Matrix, wobei N die Anzahl der Deskriptoren im ersten Bild darstellt.

Es können Deskriptoren vorkommen, für die keine zuverlässiger Match gefunden werden kann. Das kann unter anderen passieren, wenn Schlüsselpunkte an Rändern von Objekten oder wirren Bereichen im Bild gefunden werden. Weil mit solchen Deskriptoren keine gute Aussage über die Ähnlichkeit von zwei Bildern getroffen werden kann, müssen diese herausgefiltert werden.

Dazu schlägt Lowe den sogenannten "ratio test"vor. Dabei wird die Distanz des erst- und zweitbesten Matchs verglichen. Wenn das Verhältnis zwischen den beiden Distanzen kleiner als 0.8 ist, wird der erstbeste Match behalten. Ansonsten wird der Match verworfen. Dieses Vorgehen basiert auf der Annahme, dass die Distanz beim erstbesten Match deutlich niedriger als die Distanz beim zweitbesten Match sein muss, damit der erstbeste

Match als zuverlässig gilt. Sind die Distanzen zu ähnlich, kann kein eindeutiger Match für den Deskriptor gefunden werden. [Low04, S.104]

2.3 Test-Metriken

Das Bilderkennungssystem soll die Bilder in zwei Klassen einordnet: Duplikate und Unikate. Da es in diesem Fall bei der Klassifizierung nur zwei mögliche Klassen gibt, kann man das Bilderkennungssystem als binären Klassifikator bezeichnen.

Die Performance von binären Klassifikatoren kann durch die Werte einer Wahrheitsmatrix quantifiziert werden. Die Wahrheitsmatrix gibt dabei an, wie viele richtige und falsche Entscheidungen das System bei der Klassifikation getroffen hat. [Tha20, S. 170]

System Klass.:	Duplikat	Unikat
Duplikat	Anzahl richtige Duplikate (TP)	Anzahl falsche Duplikate (FP)
Unikat	Anzahl falsche Unikate (FN)	Anzahl richtige Unikate (TN)

Tabelle 2.1: Wahrheitsmatrix für Duplikaten-Suche

Aus den Werten der Wahrheitsmatrix lassen sich weitere Metriken ableiten, die für die Bewertung eines binären Klassifikators nützlich sein können. Interessant für diese Arbeit sind Recall, Spezifizität und Balancierte-Genauigkeit.

Der Recall, oder auch true positive rate, gibt das Verhältnis zwischen den Duplikaten, die das System korrekt klassifiziert hat, und allen Duplikaten, die sich in dem Suchdatensatz befinden, an. [Tha20, S. 172]

$$Recall = \frac{TP}{TP + FN}$$

Die Spezifizität, oder auch true negative rate, gibt das Verhältnis zwischen den Unikaten, die das System korrekt klassifiziert hat, und allen Unikaten, die sich im Suchdatensatz befinden an. [Tha20, S. 172]

$$Specificity = \frac{TN}{FP + TN}$$

Die Accuracy (dt. Genauigkeit), gibt das Verhältnis zwischen den Bildern, die das System korrekt klassifiziert hat, und allen Bildern im Suchdatensatz an. Sie spiegelt die Fähigkeit des Systems Bilder richtig zu Klassifizieren wieder. [Tha20, S. 171]

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Recall * P + Specificity * N}{P + N}$$

Allerdings ist die Genauigkeit kein zuverlässiger Wert, wenn man mit unausgeglichenen Datensätzen arbeitet. Ein Datensatz gilt dann als unausgeglichen, wenn einer Klassifikation mehr Elemente angehören, als der anderen Klassifikation. [Tha20, S. 171] In den Testdatensätzen, die für diese Arbeit verwendet werden, befinden sich mehr Unikate als Duplikate. Dieses Ungleichgewicht kann die Genauigkeit stark beeinflussen. So kann zum Beispiel eine gute Spezifizität einen schlechten Recall ausgleichen, wenn der Datensatz zum größten Teil aus Unikaten besteht.

Daher kommt bei unbalancierten Datensätzen die balanced-accuracy (dt. Balancierte-Genauigkeit) zum Einsatz. [Tha20, S. 175]

$$Balanced - Accuracy = \frac{Recall + Specificity}{2}$$

Da sie den Durchschnitt aus Recall und Spezifizität bildet, werden Unterschiede zwischen den beiden Werten ausgeglichen. Dadurch ist die Balancierte-Genauigkeit robust gegenüber unausgeglichenen Datensätzen.

3 Analyse

In der Analyse wird Problemstellung untersucht, die durch diese Arbeit gelöst werden soll. Dafür wird in Abschnitt 3.1 erläutert, wie der Begriff Duplikat im Kontext dieser Arbeit interpretiert wird. Abschnitt 3.2 beschäftigt sich mit dem Baseline-Algorithmus der momentan in der SpreadGroup eingesetzt wird. In Abschnitt 5.2 werden die Daten besprochen, die für die Test eingesetzt werden sollen. Zuletzt wird in Abschnitt 3.4 ein Plan für das weitere Vorgehen in der Arbeit aus der Analyse abgeleitet

3.1 Duplikate

Definition Im Kontext dieser Arbeit sind mit Duplikaten Bilder gemeint, die den gleichen Inhalt wie ein bereits bekanntes Bild haben. Der Inhalt ist in diesem Fall im optischen und nicht im semantischen Sinne gemeint. Ein Duplikat muss optische gleiche, oder zumindest sehr ähnliche Merkmale im Vergleich zum Original aufweisen. Wenn zwei Bilder ein Haus zeigen, und somit im semantischen Sinne den gleichen Inhalt haben, ist das erstmal nicht interessant. Nur wenn die Häuser ähnliche oder gleiche Merkmale aufweisen, indem sie zum Beispiel den gleichen Aufbau haben, kommt eine Klassifizierung als Duplikat in Frage.

Da der Vergleich von Merkmalen hierbei eine zentrale Rolle spielt, muss ein Bild keine pixelgenaue Kopie sein um als Duplikat zu gelten. Ein Bild kann auch dann ein Duplikat sein, wenn sein Inhalt im Vergleich zum Original rotiert, skaliert, verschoben oder gespiegelt wurde. Bilder mit anderer Hintergrundfarbe und anders gefärbten Motiv können ebenfalls als Duplikat gelten. Wichtig ist, die Form de Merkmale. Auch wenn das Motiv des Originalbilds als Teil des Motivs, des neuen Bilds auftaucht, soll es im Sinne der Arbeit als Duplikat erkannt werden.

Diese sehr weit gefasste Definition von Duplikaten ist nötig, damit die Bildakkreditierung bei Spreadshirt nicht so einfach überlistet werden kann. Der Nutzer soll nicht in der Lage sein, auf Spreadshirts Plattform ein verbotenes Bild hochzuladen, wenn er es über die

genannten Methoden verändert. Ein Bild wird aufgrund seines Inhalts verboten. Solange der verbotene Inhalt erkennbar ist, spielt es keine Rolle, wie er transformiert wurde.

Duplikate mit Merkmalen erkennen Die in den Grundlagen in Abschnitt 2 vorgestellten merkmalbasierten Algorithmen liefern Deskriptoren, mit denen bei Deskriptoren von einem anderen Bild nach Übereinstimmungen, auch Matches genannt, gesucht werden können.

Die Menge an gefundenen Matches an sich gibt allerdings noch keinen Aufschluss darüber, ob die beiden Bilder tatsächlich übereinstimmen. Es braucht eine Formel, die anhand der Matches eine Aussage darüber treffen kann, wie ähnlich sich zwei Bilder sind. Bei meiner Recherche habe ich keine konkrete Formel gefunden, die das bewerkstelligt. Daher muss eine eigene Formel entwickelt werden.

3.2 Baseline-Algorithmus (pHash)

Spreadshirt nutzt momentan einen Perceptual-Hashing-Algorithmus (kurz pHash) um neu hochgeladene Bilder zu erkennen, die bereits verboten wurden. Die Idee von pHash ist, dass für ähnliche Bilder ein ähnlicher pHash-Wert ermittelt wird. Die pHash Implementierung bei Spreadshirt stützt sich auf den Blockeintrag [pHa] und lässt sich grob in drei Phasen unterteilen.

Phase 1: Vorverarbeitungsphase Zuerst kommt das Bild in eine Vorverarbeitungsphase. Falls das Bild einen transparenten Hintergrund hat, wird es so zugeschnitten, dass der Bildrand direkt Rand des Motivs liegt. Danach wird das Bild auf eine Auflösung von 32x32 herunter skaliert und in ein Graustufenbild umgewandelt.

Phase 2: Umwandlung in den Frequenzraum In dieser Phase wird das Bild vom Ortsraum in den Frequenzraum überführt. Dazu wird die diskrete Kosinustransformation (kurz DCT) genutzt. Es entsteht eine 32x32 DCT-Matrix bestehend aus Koeffizienten, mit denen, in Kombination mit der Kosinusfunktion, das Originalbild rekonstruiert werden kann.

Für pHash wird der obere linke 8x8 Bereich der DCT-Matrix genutzt. Dieser Bereich enthält die Niederfrequenzkomponenten und wird als reduzierte DCT-Matrix bezeichnet. Die Niederfrequenzkomponenten sind für pHash interessant, weil sie die groben Strukturen

und Muster des Bildes beschreiben. Zwei Bilder mit ähnlichen Strukturen und Mustern werden ähnliche Niederfrequenzkomponenten haben und damit zu ähnlichen Hash-Werten führen.

Phase 3: Hash-Berechnung Für die Hash-Berechnung wird der Median in der reduzierten DCT-Matrix ermittelt. Die reduzierte DCT-Matrix der Größe 8x8 beinhaltet insgesamt 64 Werte. Für jeden Wert wird geprüft, ob er über oder unter dem Median liegt. Wenn der Wert größer ist, wird eine 1 und ansonsten eine 0 an den Hash-Wert angehängt. Damit ergibt sich ein 64-bit Hash-Wert für ein Bild.

pHash-Werte vergleichen Zum vergleichen von pHash-Werten, wird die Hamming-Distanz (Siehe Abschnitt 2.2.1) genutzt. Bei Spreadshirt gilt ein Bild als Duplikat, wenn beim Hash-Vergleich mit einem Bild aus der Datenbank eine Hamming-Distanz kleiner oder gleich 4 ermittelt wird.

3.3 Test-Daten

Um die Algorithmen auf ihre Leistung im Anwendungsfall bei Spreadshirt testen zu können, sind Bilder notwendig, die repräsentativ für die Designs, die bei Spreadshirt hochgeladen werden, sind.

Für die Tests wird ein **Datenbank-Set**, mit Bildern die Design-Datenbank bei Spreadshirt darstellen sollen, erstellt.

Zudem wird für jedes Test-Szenario (siehe Abschnitt 5.1) ein **Such-Set** erstellt. Dieses Set enthält Suchbilder, die mit den Bildern aus der Datenbank verglichen werden sollen. Das Set enthält dabei Duplikate aus dem Datenbank-Set, die entsprechend des Szenarios modifiziert sind. Es sind auch Bilder enthalten, die keine Duplikate aus dem Datenbank-Set sind und auch nicht als solche erkannt werden sollen.

3.4 Vorgehen

Aus der Betrachtung der Problemstellung ergeben sich die folgenden Teilaufgaben:

1. Definition einer Formel zur Bestimmung der Ähnlichkeit von zwei Bildern

2. Evaluation des Baseline-Algorithmus und der merkmalbasierten Algorithmen
    a) Definition von Test-Szenarien in denen die Algorithmen verglichen werden
    b) Erstellung von Suchbildern für die Test-Szenarien
    c) Entwicklung eines Test-Systems, das die Algorithmen durch die Test-Szenarien laufen lässt und Statistiken festhält
    d) Auswertung der Tests zur Evaluation des Baseline-Algorithmus und zur Bestimmung welcher Feature-Based Algorithmus am besten für das neue Bilderkennungssystem geeignet ist
3. Entwicklung und prototypische Implementierung eines neuen Bilderkennungssystems
4. Vergleich des neuen Bilderkennungssystems mit dem Baseline-Algorithmus

4 Formel zur Bestimmung der Bildähnlichkeit

In diesem Kapitel soll eine Formel definiert werden, mit der ein Ähnlichkeits-Score errechnet wird. Mit diesem Score soll eine Aussage darüber getroffen werden, wie Ähnlich sich zwei Bilder sind. Als Eingabe erhält die Formel Matches, die durch eine der Matching-Strategien (Abschnitt 2.2.2) gefunden und durch Lowes ratio test (Abschnitt 2.2.3) gefiltert wurden.

4.1 Vorüberlegung

Je niedriger die Distanz eines Matches desto ähnlicher sind sich die, in dem Match verglichenen, Deskriptoren. (Abschnitt 2.2.1) Und je ähnlicher die Deskriptoren desto ähnlicher sind sich die verglichenen Bilder. Daher sollte die durchschnittliche Distanz aller Matches Aufschluss darüber geben, wie ähnlich sich zwei Bilder sind.

Die Werte des Ähnlichkeits-Score sollten außerdem auf einen festgelegten Bereich abgebildet werden. Dadurch ist der Ähnlichkeits-Score einer Bildpaarung gut mit dem Ähnlichkeits-Score einer anderen Bildpaarung vergleichbar.

4.2 Berechnung

$$score = 1 - \frac{\sum_{d \in D} \frac{d}{max(D)}}{|D|}$$

Es werden alle Distanzen aller gefilterten Matches normalisiert und aufsummiert. Um die Werte auf den Bereich zwischen 0 und 1 abzubilden, wird die Distanz durch die Maximaldistanz geteilt. Bei der Normalisierung sind Werte kleiner Null sind nicht möglich, da alle Distanzen mindestens 0 betragen.

Ursprünglich war für die Formel die herkömmliche Min-Max-Normalisierung mit $\frac{d-min(D)}{max(D)-min(D)}$ vorgesehen. Allerdings hat sich diese bei weiterer Überlegung als ungeeignet herausgestellt. Für hohe Werte bei $min(D)$, die für große unterscheide in den verglichenen Bildern sprechen, kommen bei der Min-Max-Normalisierung niedrige Werte heraus, wenn $min(D)$ ausreichend nah an $max(D)$ liegt. Dadurch kann es in solchen Fällen zu false-positives, also Bildern die fälschlicherweise als Duplikat erkannt werden, kommen. Durch das auslassen von $min(D)$ in der Formel werden die Distanz-Werte nicht durch ein hohes $min(D)$ beschnitten.

Die Summe der normalisierten Werte wird durch die Anzahl der gefilterten Matches geteilt, um einen normalisierten Durchschnittswert zu erhalten. Der Ähnlichkeits-Score ergibt sich aus der Differenz von 1 und dem normalisierten Durchschnittswert und liegt im Bereich zwischen 0 und 1. Je höher der Score desto ähnlicher sind sich die verglichenen Bilder. Ein Score von 1 würde bedeuten, dass beide Bilder identisch sind.

Der Mindest-Score, der erreicht werden muss, damit ein Bild als Duplikat gilt, soll im Verlauf der Tests ermittelt werden.

5 Test-System

5.1 Test-Szenarien

Aus der Analyse hat sich ergeben, dass ein Bild auf unterschiedliche Weisen verändert werden und trotzdem als Duplikat gelten kann. Daher sollen Test-Szenarien definiert werden, in denen die Algorithmen getestet werden. Jedes Szenario bildet dabei eine Modifikation ab, die auf ein Bild angewendet werden kann, sodass es im Sinne der Arbeit immer noch als Duplikat gilt.

In den folgenden Szenarien soll getestet werden:

- **identische Kopie** Im Suchset sind pixelgenaue Kopien aus dem Datenbank-Set enthalten. Dieses Szenario stellt den Trivialfall dar, bei eine Balancierte-Genauigkeit von 100% und ein Ähnlichkeits-Score von 1 bei Duplikaten erwartet wird. Eine niedrigere Balancierte-Genauigkeit oder ein niedrigerer Ähnlichkeits-Score lassen darauf schließen, dass der getestete Algorithmus nicht richtig implementiert und die Bildähnlichkeits-Formel fehlerhaft ist.
- **Geänderte Auflösung** Es werden für die duplizierten Bilder niedriger aufgelöste Varianten erstellt. Dabei werden die 3 Skalierungsfaktoren 2, 4, 10 und angewendet. Für jeden Skalierungsfaktor wird eine Variation erstellt, deren Seitenlängen um den Skalierungsfaktor gekürzt werden.
- **Verschobenes Motiv** Das Motiv des Duplikats ist im Vergleich zum Original verschoben.
- **Rotiertes Motiv** Die Duplikate sind in diesem Fall im Vergleich zum Original rotiert. Es werden die Rotationswinkel 5°, 10°, 45°, 90° und 180° verwendet. Für jedes duplizierte Bild wird für jeden Winkel eine Variation erstellt.
- **Gespiegeltes Motiv** Für jedes duplizierte Bild existiert in diesem Szenario zwei Varianten. Die eine Variante ist an der X-Achse gespiegelt und die andere Variante ist an der Y-Achse gespiegelt.

- **Andere Hintergrundfarbe** Die Duplikate haben in diesem Szenario im Vergleich zum Originalbild eine andere Hintergrundfarbe. Die neue Hintergrundfarbe muss genügend Kontrast zum Motiv haben, sodass das Motiv noch erkennbar ist.
- **Motiv in anderer Farbe** In diesem Szenario haben die Duplikate eine anders gefärbtes Motiv. Dafür werden Originalbilder mit flachen Farbstil verwendet. Diese lassen sich leichter neu einfärben.
- **Originalbild als teil eines größeren Bildes** Für Duplikate in diesem Szenario wird das Originalbild auf einem größeren Bild platziert.
- **Misch-Szenario** Alle anderen Szenarien kommen in diesem Szenario vor. Ziel ist es eine realistische Anwendungssituation darzustellen. Dazu werden auch Duplikate mit mehreren Modifikationen erstellt. Beispielsweise können Duplikate, die sowohl rotiert als auch eine niedriger Auflösung haben, vorkommen.

Für alle Szenarien werden die, in den Grundlagen definierten Metriken (Abschnitt 2.3), festgehalten. Dadurch kann jedes Szenario einzeln ausgewertet werden und Aufschluss über die individuellen Stärken und Schwächen der getesteten Algorithmen geben.

In den Tests der feature-based Algorithmen soll in jedem Szenario mit verschiedenen Mindestwerten für den Bildähnlichkeits-Score experimentiert werden. Ziel ist es für jeden Algorithmus einen Mindestwert zu finden, der für alle Szenarien eine möglichst hohen Recall und eine gute Balancierte-Genauigkeit erreicht.

Zudem werden im Misch-Szenario Laufzeit und Speicherkosten der einzelnen Algorithmen dokumentiert. Da nur in diesem Szenario, die Geschwindigkeit eine Rolle spielt, werden hier der Brute-Force und FLANN-Based Matcher verglichen. Da es bei allen anderen Szenarien um die Genauigkeit geht, wird dort nur der Brute-Force Matcher eingesetzt.

5.2 Erstellung der Suchbilder

Für die Erstellung der Suchbilder wurden für jedes Szenario zufällige Bilder aus dem Datenbanksatz genommen. Zum Erstellen der Variationen wurden für einige Szenarien ein Skript verwendet. Das Ändern der Auflösung und das spiegeln an den Hauptachsen sind einfache Transformationen, die sich leicht automatisieren lassen. Da alle Bilder aus dem Datenbankset einen transparenten Hintergrund haben, lassen sich diese leicht auf die Dimensionen des Motivs zuschneiden. Dadurch können die Motive verschoben und rotiert werden, ohne das sich der Hintergrund mit bewegt. Auch das Ändern der Hintergrundfarbe und das Platzieren des Motivs auf einen größeren Bild sind so leicht möglich. Einzig das

Ändern der Färbung von Motiven lässt sich nicht einfach in einem Skript realisieren und wurde daher von Hand gemacht.

Das Skript ist in Go [go] geschrieben und alle Bildtransformationen ließen sich über die Standartbibliotheken implementieren. Für die Suchsets wurden pro Szenario 100 bis 150 Variationen erstellt.

5.3 Entwurf und Umsetzung

6 Versuche und Auswertung

6.1 Evaluation des Baseline-Algorithmus

6.2 Evaluation der feature-based Algorithmen

7 Neues Bilderkennungs-System

7.1 Vorüberlegung

Die Auswertungen der Tests haben gezeigt, das die Deskriptoren einen sehr hohen Speicherbedarf haben und sich daher nicht für die Persistierung in einer Datenbank eignen. Auch die Laufzeit von feature-based Algorithmen ist deutlich höher als die Laufzeit des pHash Algorithmus, wenn alle Deskriptoren in der Datenbank verglichen werden sollen.

Außerdem würde der Umstieg vom momentan genutzten pHash-System auf ein feature-based-System bedeuten, das die bereits durch Spreadshirt gesammelten Hash-Werte nicht mehr brauchbar sind. Für das neue System müssten für mehrere Millionen abgelehnte Designs die Deskriptoren berechnet und gespeichert werden, was ein großer, wenn auch einmaliger, Aufwand ist.

Daher wird für das neue Bilderkennungs-System ein Ansatz vorgesehen, der sich die niedrigen Speicherkosten und den schnellen Vergleich von Bildern mit pHash zunutze macht. Gleichzeitig soll es aber auch von der Robustheit der feature-based Algorithmen profitieren.

Die Idee ist, dass pHash genutzt wird, um aus der Datenbank eine Vorauswahl mit potenziellen Duplikaten zu treffen. Danach wird ein feature-based Algorithmus genutzt, um den besten Match zu finden. Die Schlüsselpunkte und Deskriptoren des Suchbildes und der Kandidaten aus der Vorauswahl werden dabei zur Laufzeit extrahiert.

Mit diesem Hybrid-Ansatz können die bereits bei Spreadshirt gesammelten Hash-Werte weiterhin genutzt werden. Daher muss das bisherige pHash-System nicht ersetzt werden und wird lediglich erweitert. Außerdem bleiben auch die Persistierungskosten für die abgelehnten Designs gleich, da weiterhin nur ein einzelner 64 Bit pHash-Wert pro Bild gespeichert werden muss.

Die Zuverlässigkeit des Systems wird aber dennoch weiterhin stark durch die Zuverlässigkeit des pHash-Algorithmus beschränkt. Der feature-based Algorithmus hat in diesem

Entwurf nur Zugriff auf die Bilder, die in der Vorauswahl durch den pHash Algorithmus gesammelt wurden. Daher können nur die Duplikate erkannt werden, die auch durch pHash erkannt wurden.

Die Tests haben gezeigt, das der pHash Algorithmus Schwierigkeiten hat Duplikate zu erkennen die gespiegelt oder rotiert sind. Besonders problematisch sind aber auch Duplikate, bei denen das Originalbild als Teil des Motivs auftauchen. Daher wird es nötig sein das System für die Vorauswahl so anzupassen, sodass pHash mit den diesen Fällen umgehen kann. Das Ziel ist für die Vorauswahl einen möglichst hohen Recall zu erreichen, damit möglichst alle potenziellen Duplikate durch den feature-based Algorithmus geprüft werden können.

7.2 Entwurf und Umsetzung

7.3 Auswertung

8 Schluss

8.1 Fazit

8.2 Diskussion

8.3 Ausblick

Literaturverzeichnis

[go] Go dokumentation.

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, 2011.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[ope] Opencv dokumentation.

[pHa] Blockeintrag phash.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 430–443. Springer, 2006.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[Tha20] Alaa Tharwat. Classification assessment methods. *Applied computing and informatics*, 17(1):168–192, 2020.

Abbildungsverzeichnis

Tabellenverzeichnis

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Graduierungsarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommene Stellen sind als solche einzeln kenntlich gemacht.

Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Leipzig, 26. September 2023 Unterschrift