Identifikation von historischen Gebäuden und Bauteilen durch Bildklassifikation

Christof Wittmann

Bachelorarbeit

im Studiengang Angewandte Informatik der
Fakultät Wirtschaftsinformatik und Angewandte Informatik
Otto-Friedrich-Universität Bamberg

4.4.2019

Wissenschaftliche Betreuung: Prof. Dr. Christoph Schlieder
Softwaretechnischer Ansprechpartner: Thomas Heinz
Lehrstuhl für Angewandte Informatik
in den Kultur-, Geschichts- und Geowissenschaften

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

In den letzten Jahren hat das Thema Bildklassifikation das Interesse einer breiten Öffentlichkeit auf sich gezogen. Mit Hilfe von *Machine Learning* und Neuronalen Netzwerken ist es nunmehr möglich, Objekte auf Bildern mit bisher ungekannter Sicherheit zu klassifizieren und identifizieren. Ein Nachteil dieses Ansatzes ist jedoch der Bedarf an einer großen Menge an verfügbaren Trainingsdaten, in diesem Fall also Bildern der zu identifizierenden Objekte. Während es somit relativ leicht möglich ist, auf einer Aufnahme etwa Gebäude als Objekte des Typs „Gebäude“ zu erkennen, so stellt die Identifizierung individueller Bauwerke weiterhin eine kaum zu überwindende Hürde dar.

Für Anwendungsfälle im Bereich der Bildklassifikation, bei denen die Verfügbarkeit von Trainingsbildern deutlich eingeschränkt ist, muss deshalb bis auf Weiteres auf alternative Methoden zurückgegriffen werden. Ein vielversprechender Ansatz ist dabei die *Feature Detection* (Merkmalserkennung). Hierbei extrahiert ein Algorithmus aus einem Bild eine Menge von Punkten, sog. *Keypoints*, die als besonders geeignet gelten können, dieses Bild zu beschreiben. Die Ähnlichkeit zweier Bilder kann nun durch den Vergleich dieser *Keypoints* ermittelt werden.

In dieser Arbeit sollen die wichtigsten dieser Algorithmen miteinander verglichen werden, insbesondere in Bezug auf ihre Eignung für die Klassifikation historischer Gebäude und Bauteile. Dabei werden zuerst die verfügbaren Algorithmen und ihre Beziehung zueinander dargestellt. Als konkretes Anwendungsbeispiel dient schließlich eine plattformunabhängige Applikation, mit der eine fotografische Aufnahme eines Gebäudes oder Bauteils mit Aufnahmen in einer Datenbank verglichen wird, um den BenutzerInnen anschließend Informationen über das identifizierte Objekt anzuzeigen. In Hinblick auf diesen Anwendungsfall wird schließlich die Performance der verfügbaren Algorithmen getestet, um zu ermitteln, welcher von ihnen am Besten geeignet ist, um im Rahmen der Applikation eingesetzt zu werden.

2. Problemstellung

BIdent Building Identification ist eine mobile Applikation, mit der fotografische Aufnahmen von (historischen) Gebäuden und Bauteilen erstellt werden können. Die Fotografie wird daraufhin an einen Server übermittelt, auf dem sie unter Nutzung eines bestimmten Merkmalserkennungs-Algorithmus mit Bildern aus einer Datenbank vorhandener Gebäude verglichen wird.

Die Zielgruppe besteht dabei aus TouristInnen, die sich näher über historische Gebäude in der jeweiligen Stadt informieren möchten. Dabei kann kein Hintergrundwissen über Fragen der Bildklassifikation vorausgesetzt werden, weshalb die Möglichkeit zur Auswahl oder Konfiguration eines Merkmalserkennungs-Algorithmus seitens der BenutzerInnen kein Erfordernis für die Benutzung der Anwendung sein soll. Es ist deshalb ein Default-Algorithmus zu bestimmen und verwenden, der für unterschiedliche Motive und Aufnahmebedingungen stets zufriedenstellende Ergebnisse liefert. Für die Qualität der Umsetzung spielt es eine besondere Rolle, welcher Algorithmus für den Bildvergleich eingesetzt wird. Diese Wahl beeinflusst nicht nur die Geschwindigkeit der Anwendung und damit die Zufriedenheit der BenutzerInnen, sondern auch die Qualität des Bildvergleichs, also die Wahrscheinlichkeit, dass das fotografierte Objekt korrekt identifiziert wird.

Für die Bildübertragung zwischen Client und Server ist abhängig von der Netzwerk-Konnektivität mit einer zeitlichen Dauer von mehreren Sekunden zu rechnen. Dabei wird nicht nur die Fotoaufnahme von Client zu Server übertragen, sondern auch ein in der Datenbank hinterlegtes Bild des identifizierten Gebäudes oder Bauteils zurück an den Client. Aus Gründen der Benutzerfreundlichkeit ist es deshalb erforderlich, die Berechnungsdauer der Algorithmen am Server möglichst gering zu halten.

Da mit einer größeren Zahl von Bildern in der Datenbank zu rechnen ist, ist eine geographische Eingrenzung der Vergleichsobjekte zwingend vorzunehmen. Da das mobile Gerät, die Erlaubnis der BenutzerInnen vorausgesetzt, in der Lage ist, seine momentane Position über GPS zu bestimmen, können die so gewonnenen Informationen verwendet werden, um den Verglich nur mit solchen Bildern durchzuführen, deren hinterlegte geographische Position sich in der Nähe des mobilen Geräts befindet. Dieser Ansatz wurde bereits von anderen Autoren umgesetzt, etwa bei Hutchings & Mayol-Cuelvas (Hutchings & Mayol-Cuelvas, 2005, zitiert nach Li et al., 2014).

Neben der Berechnungsdauer ist auch die Qualität des Bildvergleichs von entscheidender Bedeutung für die Akzeptanz seitens der BenutzerInnen. Nachdem ein Bildvergleich abgeschlossen ist, muss anhand der vorliegenden Kennzahlen mit möglichst hoher Genauigkeit bestimmt werden, ob die

verglichenen Bilder ein identisches Motiv enthalten. Die bloße Anzahl gefundener Matches kann hier jedoch nicht als Kriterium gewählt werden, da diese nur einen geringen Aussagewert besitzt. Der Vergleich der Menge an Guten Matches gemäß Lowes *Distance Ratio* (vgl. Kapitel x) ist hingegen deutlich aussagekräftiger. Sofern die Aufnahmebedingungen der Fotografie nicht zu stark von denen des Vergleichsbilds in der Datenbank abweichen, sollte eine korrekte Identifizierung generell möglich sein.

Hierbei ist jedoch zu beachten, dass die Bilddatenbank unmöglich alle existierenden Gebäude enthalten kann. Es ist demnach auch der Fall zu berücksichtigen, dass BenutzerInnen Aufnahmen von Gebäuden oder Bauteilen machen, die nicht in der Datenbank enthalten sind. Würde der Server nun lediglich Informationen über das Gebäude mit den meisten Guten Matches zurückgeben, so wäre in diesem Fall mit einer inkorrekten Identifikation zu rechnen. Aus diesem Grund ist es unbedingt erforderlich, zu prüfen, wie viele Gute Matches ein bestimmter Algorithmus sowohl bei Vorliegen als auch Nichtvorliegen eins identischen Motivs zurückgibt und auf dieser Basis Wertebereiche festzulegen, anhand derer die Applikation die Sicherheit der Identifikation feststellen kann. Ein Algorithmus, bei dem die Anzahl Guter Matches sich bei Gleichheit und Ungleichheit des Motivs möglichst stark voneinander unterscheidet, kann deshalb als besonders geeignet gelten. Hierbei ist selbstverständlich auch zu berücksichtigten, dass die korrekte Identifizierung möglichst unabhängig von den äußeren Umständen der Aufnahme sein sollte.

Eine kommerzielle Nutzung der Applikation ist nicht vorgesehen, weshalb die lizenzrechtlichen Einschränkungen der Algorithmen SIFT und SURF nicht berücksichtigt werden müssen.

Im Rahmen dieser Arbeit ist es zuerst erforderlich, sich auf eine kleinere Anzahl von Algorithmen zu beschränken, unter denen dann im Rahmen der Evaluierung der geeignetste ermittelt werden kann. Bei der Auswahl der zu vergleichenden Algorithmen kann etwa deren Erwähnung in der bestehenden Forschungsliteratur als Kriterium verwendet werden. Die Auswertung mehrerer Vergleichsstudien liefert dabei eine Liste von sechs Algorithmen, die mindestens in einer Arbeit untersucht wurden. In Tabelle X werden diese mitsamt ihrem Veröffentlichungszeitpunkt und ihren Autoren aufgelistet. (Andersson & Marquez, 2016, Tareen & Saleem, 2018, Zhang et al., 2019).

Als weitere Entscheidungsgrundlage kann dabei die Tatsache dienen, dass es sich dabei auch um die *Feature Detection*-Algorithmen handelt, die von der populären *Computer Vision*-Bibliothek *OpenCV* zur Verfügung gestellt werden.[1] [2]

[1] https://docs.opencv.org/master/d5/d51/group__features2d__main.html (Letzter Zugriff: 30.3.2020)

[2] https://docs.opencv.org/2.4/modules/nonfree/doc/feature_detection.html (Letzter Zugriff: 30.3.2020)

TABELLE X. ALGORITHMEN ZUR MERKMALSERKENNUNG

Name	**Jahr der Veröffentlichung**	**Autor(en)**
SIFT	1999	Lowe
SURF	2006	Bay, Tuytelaars, Van Gool
BRISK	2011	Leutenegger, Chli, Siegwart
ORB	2011	Rublee, Rabaud, Konolige, Bradski
KAZE	2012	Alcantarilla, Bartoli, Davison
AKAZE	2013	Alcantarilla, Nuevo, Bartoli

Tab. X. Übersicht über populäre Algorithmen zur Merkmalserkennung (Andersson & Marquez, 2016, Tareen & Saleem, 2018, Zhang et al., 2019).

3. Forschungsstand: Methoden der Bilderkennung

3.1. Grundprinzipien der Merkmalserkennung (Feature Detection)

Um Bilder informatisch miteinander vergleichen und ihre Ähnlichkeit ermitteln zu können, ist es erforderlich, sich auf bestimmte Attribute dieser Bilder zu konzentrieren. Bei Verfahren der Merkmalserkennung werden deshalb interessante Punkte ermittelt, die besonders für den Bildvergleich geeignet sind. Als interessant kann dabei ein Punkt gelten, der in Bezug auf seine Nachbarschaft eine signifikante Veränderung aufweist, etwa hinsichtlich seiner Farbe, seines Helligkeitswertes oder seiner Richtung. Die solchen Verfahren zugrundeliegende Annahme ist, dass derart interessante Punkte mit hoher Wahrscheinlichkeit auf allen Bildern zu finden sind, die ein identisches Objekt abbilden (Andersson & Marquez, 2016).

Die fotografische Aufnahme eines Objekts kann auf sehr unterschiedliche Weise erstellt werden, wobei die fotografierende Person eine Vielzahl von Faktoren variieren kann, um zum gewünschten Ergebnis zu kommen. Eigenschaften wie Perspektive, Entfernung und Richtung können direkt durch Positionsänderung der Kamera beeinflusst werden, wobei die Möglichkeiten ggfs. durch die Umgebungssituation des Objekts eingeschränkt werden. Mittels der Kameraeinstellungen ist etwa die Helligkeit oder Farbbalance der Aufnahme bis zu einem gewissen Grad beeinflussbar, ebenso das Format des erzeugten Bildes. Weniger Einfluss hat die fotografierende Person auf die Lichtverhältnisse, insbesondere bei Aufnahmen im Freien. Selbst die Wahl einer geeigneten Tageszeit und der Einsatz künstlicher Beleuchtung können nicht verhindern, dass örtliche Lichtverhältnisse stark durch die vorliegenden Wetterbedingungen beeinflusst werden. Beim Bildvergleich ist es deshalb von herausragender Bedeutung, dass bezüglich der genannten Faktoren eine größtmögliche Invarianz gegeben ist. Dies bedeutet, dass bei identischen Objekten idealerweise auch die gleichen Features identifiziert werden, selbst wenn die Aufnahmen in vielerlei Hinsicht erheblich voneinander abweichen (Andersson & Marquez, 2016).

Bei der Merkmalserkennung handelt es sich um eine der beiden Hauptrichtungen der inhaltbasieren Bildsuche und -klassifikation. Alternativ dazu existieren auch Methoden, die sich des Maschinellen Lernens bedienen, um

den Inhalt des Bildes auf der höchstmöglichen Ebene zu beschreiben. Entsprechend trainierte neuronale Netzwerke können somit bestimmte Bildbestandteile erkennen und klassifizieren, wobei diese Verallgemeinerung jedoch auch mit einem Informationsverlust verbunden ist, der dazu führt, dass die Gleichheit von Objekten einer gemeinsamen Kategorie auf diese Weise schwer festzustellen ist. Ein weiterer Nachteil des Maschinellen Lernens ist der Bedarf an umfangreichen Mengen von Trainingsdaten. Die Merkmalserkennung nimmt dagegen keine Generalisierung oder Klassifikation vor. Sie ist nicht nur für den direkten Ähnlichkeitsvergleich von Bildern einsetzbar, sondern etwa auch beim Videotracking, dem *Image Stitching* oder bei der dreidimensionalen Rekonstruktion von Objekten auf Basis photographischer Aufnahmen (Scherer, 2020).

3.2. Methoden der Merkmalserkennung

Merkmalserkennungs-Algorithmen können generell einer der drei folgenden Kategorien zugeordnet werden:

- Kantendetektion (Edge Detection)
- Eckendetektion (Corner Detection)
- Blobdetektion (Blob Detection)

Die Kantendetektion identifiziert Bildpunkte, die entlang einer Linie liegen, die auffallende Unterschiede bzgl. der vorliegenden Helligkeits- bzw. Farbwerte aufweist. Für sich genommen ist die Kantendetektion jedoch ungeeignet für die Merkmalserkennung und ist für diese somit nur von historischer Bedeutung.

Die Eckendetektion, für die etwa die *Harris Corner Detection* als bekanntes Beispiel genannt werden kann, bedient sich der Kantendetektion und ermittelt auf deren Basis Schnittpunkte zwischen zwei oder mehreren Kanten. Die so identifizierten Ecken sind als Features deutlich besser geeignet als Kanten. Nichtsdestotrotz ist die Eckendetektion nicht in der Lage, eine Invarianz bezüglich der Skalierung zu gewährleisten. Deshalb wird die Eckendetektion in heutigen Merkmalserkennungs-Algorithmen entweder gar nicht oder nur in Verbindung mit der Blobdetektion verwendet.

Ein entscheidender Vorteil der Blobdetektion ist die Invarianz gegenüber Perspektive, Entfernung und Rotation, womit die entsprechenden Algorithmen für viele übliche Anwendungszwecke als Mittel der Wahl gelten können. Ein bekanntes Beispiel hierfür ist der SIFT-Algorithmus, der im Folgenden vorgestellt werden soll (Andersson & Marquez, 2016).

Beschreibungen aus KAZE Features für folgende Kapitel verwenden.

3.3. Scale-Invariant Feature Transform (SIFT)

Compare with:
https://www.inf.fu-berlin.de/lehre/SS09/CV/uebungen/uebung09/SIFT.pdf

Der Scale-Invariant Feature Transform-Algorithmus (im Folgenden als SIFT abgekürzt) wurde 1999 von David Lowe entwickelt. Anhand des Namens ist bereits erkennbar, dass die grundlegende Verbesserung gegenüber bisherigen Merkmalserkennungs-Verfahren in der Invarianz bezüglich der Skalierung besteht. Der SIFT-Algorithmus wird in die folgenden vier Schritte aufgeteilt, die in den folgenden Kapiteln detailliert vorgestellt werden sollen:

1. Scale-Space Extrema Detection
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint Description

3.3.1. Scale-Space Extrema Detection

Das Ziel des ersten Verarbeitungsschritts besteht darin, eine Vielzahl interessanter Punkte innerhalb des gewählten Bildes zu identifizieren. Diese werden im Rahmen des SIFT-Algorithmus als *Keypoints* bezeichnet.

Zu Beginn werden aus dem Ursprungsbild weitere Bilder erzeugt, die sich bezüglich Skalierung und Weichzeichnungsgrad voneinander unterscheiden. Dabei wird das Bild in der Ausgangsgröße zuerst stufenweise immer stärker weichgezeichnet, wobei der *Gaussian Scale-Space Kernel* zur Anwendung kommt. Alle Bilder der gleichen Größe werden als Oktave bezeichnet.

Anschließend wird das Bild mit dem größten Weichzeichnungsgrad auf die Hälfte seiner Größe verkleinert und erneut stufenweise weichgezeichnet. Dieser Prozess wiederholt sich für weitere Oktaven, bis die Bildgröße einen unteren Schwellenwert erreicht. Abbildung x zeigt exemplarisch die dabei erzeugten Bilder.

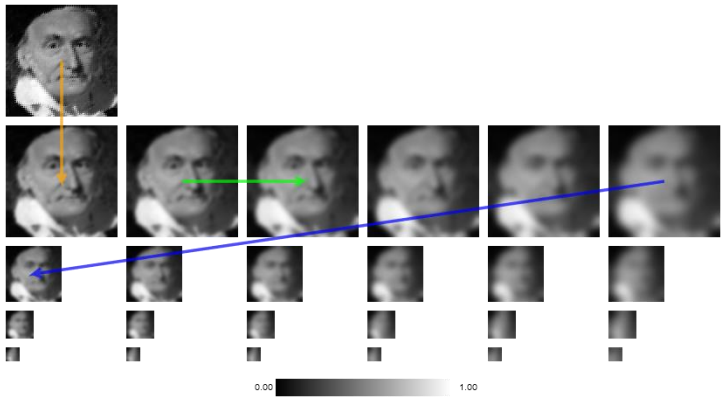

Abbildung x: Bilderzeugung im Rahmen der *Scale-Space Extrema Detection*.[3]

Nun werden aus diesen Bildern mittels der *Difference of Gaussian* (DoG)-Methode Differenzbilder generiert. Hierfür werden jeweils zwei innerhalb einer Oktave nebeneinanderliegende Bilder als Ausgangsgrundlage verwendet. In Abbildung x sind die auf diese Weise erzeugten Differenzbilder zu sehen, wobei die Zahl der Bilder pro Oktave nun um eines verringert ist.

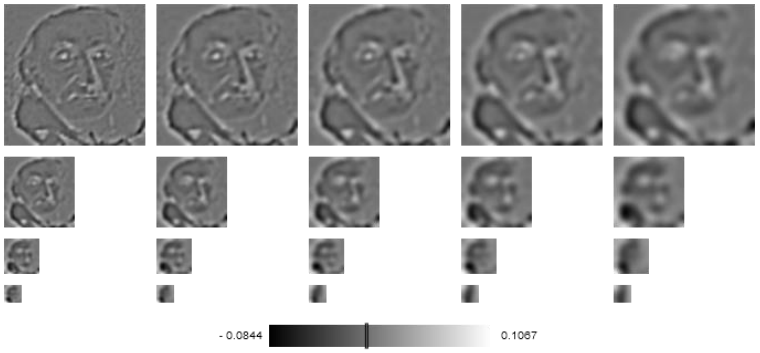

Abbildung x: Differenzbilder als Ergebnis der *Difference of Gaussian*-Berechnung. [4]

Schließlich werden die Pixel in diesen Differenzbildern anhand von Nachbarschaftsvergleichen auf ihre Eignung als interessante Punkte geprüft. Dabei werden nicht nur die umliegenden acht Pixel als Vergleichspunkte gewählt, sondern auch jeweils die angrenzenden neun Pixel in den Differenzbildern der nächstoberen und nächstunteren Oktaven, wie in Abbildung X zu sehen ist.

[3] http://weitz.de/sift (Letzter Zugriff: 30.3.2020)
[4] http://weitz.de/sift (Letzter Zugriff: 30.3.2020)

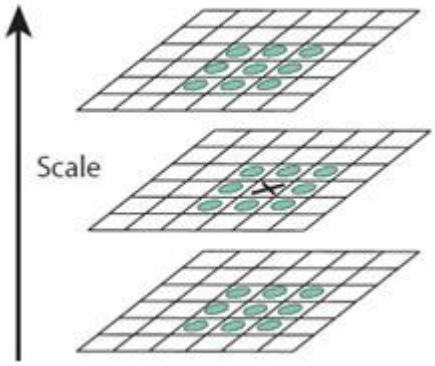

Abbildung x: Pixel-Nachbarschaftsvergleiche bei der *Scale-Space Extrema Detection* (Lowe, 2004).

Um als potenzieller Keypoint in Frage zu kommen, muss ein Pixel einen höheren bzw. niedrigeren Wert aufweisen als alle 26 Nachbarpixel. Hierdurch wird die Skalierungsinvarianz gewährleistet (Lowe, 2004, Andersson & Marquez, 2016).

3.3.2. Keypoint Localization

Die Menge der im letzten Schritt ermittelten *Keypoints* muss nun weiter eingegrenzt werden, da nicht alle von ihnen als Merkmale für die Bildidentifikation geeignet sind. Gründe für die fehlende Eignung sind entweder ein zu niedriger Kontrast oder die Lage entlang einer Kante. Um Punkte mit niedrigem Kontrast zu identifizieren, wird zuerst mittels Taylorentwicklung die genaue Position lokaler Extrema bestimmt. Aus den so ermittelten Extrempunkten werden solche herausgefiltert, deren Wert einen gegebenen Schwellenwert von 0,03 unterschreiten. Zur Entfernung von Kantenpunkten bedient man sich einem Verfahren, das der *Harris Corner Detection* verwandt ist. Um die beiden Hauptkrümmungen für alle Keypoints zu berechnen, wird die Hesse-Matrix verwendet. Anschließend wird das Verhältnis dieser Hauptkrümmungen ermittelt. Liegt dieses oberhalb des Schwellenwerts 10, so wird davon ausgegangen, dass der Punkt sich auf einer Kante befindet, weshalb er verworfen wird (Lowe, 2004, Andersson & Marquez, 2016).

3.3.3. Orientation Assignment

Um die Invarianz gegenüber der Rotation sicherzustellen, wird nun jedem *Keypoint* eine Orientierung zugewiesen. Zuerst betrachtet man hierfür die Nachbarschaft des Punktes. Da es sich bei allen *Keypoints* um Pixel in einem weichgezeichneten Bild handelt, besteht ihre Umgebung aus Helligkeitsverläufen

(*Gradients*). Für diese Verläufe, welche in Abbildung x zu sehen sind, können sowohl Intensität als auch Richtung ermittelt werden.

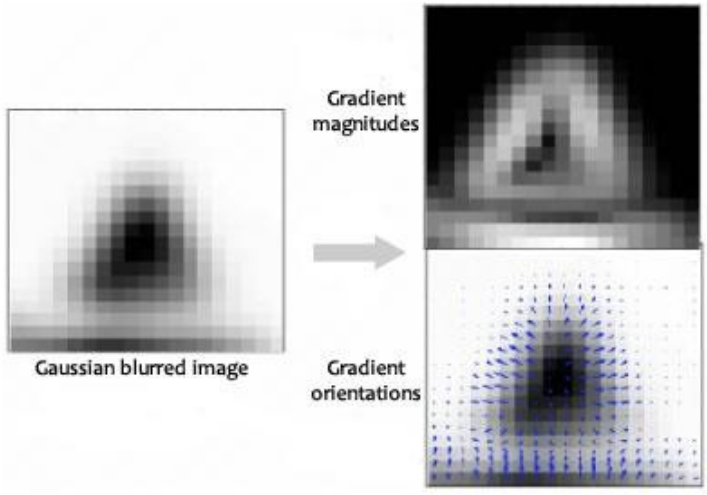

Abbildung x: Helligkeitsverläufe in Pixel-Nachbarschaft eines *Keypoints*.[5]

Es wird nun für jeden *Keypoint* ein Histogramm angelegt, in dem die Intensität des Verlaufs für jede Orientierung hinterlegt wird. Aus Performancegründen teilt man die 360°-Umgebung jedoch in 36 Behälter auf, die jeweils einem 10°-Abschnitt entsprechen. Ein Beispielhistogramm ist in Abbildung x zu sehen.

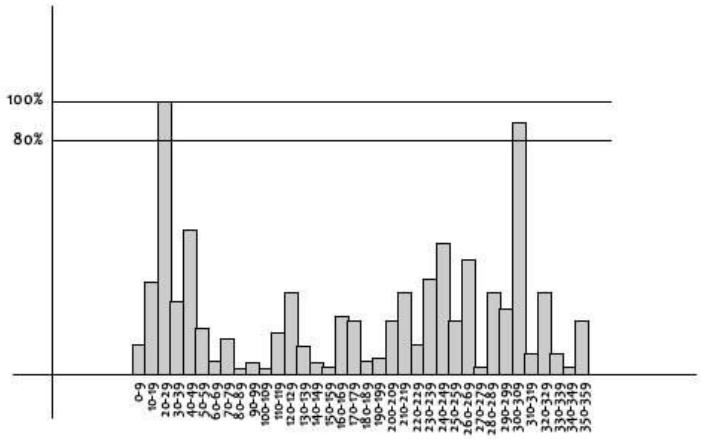

Abbildung x: Orientierungshistogramm eines *Keypoints*.[6]

In den meisten Fällen wird nun der Behälter mit dem höchsten Wert gewählt und dessen Orientierung als Orientierung des *Keypoints* festgelegt. Wie in Abbildung x zu sehen ist, können jedoch auch mehrere Orientierungen vorliegen, die eine

[5] https://aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation (Letzter Zugriff: 30.3.2020)

[6] https://medium.com/analytics-vidhya/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40 (Letzter Zugriff: 30.3.2020)

ähnliche Intensität aufweisen. Deshalb vergleicht man die Intensität aller Behälter mit der des Behälters mit dem Maximalwert. Für Behälter, die mindestens 80% von dessen Intensität erreichen, wird jeweils ein weiterer zusätzlicher *Keypoint* mit der Orientierung dieses Behälters erstellt. Somit kann die endgültige Menge an *Keypoints* auch solche enthalten, deren Lage und Skalierung identisch sind und die sich lediglich hinsichtlich der Orientierung unterscheiden (Lowe, 2004, Andersson & Marquez, 2016).

3.3.4. Keypoint Description

Nachdem jeder Keypoint bereits über eine Position, eine Skalierung sowie eine Orientierung verfügt, wird nun abschließend eine Beschreibung der Keypoint-Umgebung hinzugefügt. Diese dient dazu, den Keypoint eindeutig zu identifizieren und somit den Ähnlichkeitsvergleich von Bildern zu ermöglichen.

Zu diesem Zweck werden die Pixel in der Umgebung des Keypoints betrachtet. In Abbildung x ist auf der linken Seite die Nachbarschaft als Quadrat mit Seitenlänge 16 Pixeln zu sehen. Diese Umgebung wird nun in 16 Teilquadrate mit je 4 x 4 Pixeln aufgeteilt. Für jedes dieser Teilquadrate werden nun ähnlich wie im Schritt *Orientation Assignment* die Intensität der Helligkeitsverläufe und der Orientierung berechnet.

Die Ergebnisse dieser Berechnung werden für jedes Teilquadrat in einem Histogramm mit 8 Behältern gespeichert. Diese Behälter teilen die 360°-Umgebung in Bereiche von jeweils 45°. Hierbei ist noch zu bemerken, dass Pixel, die vom betrachteten *Keypoint* weiter entfernt sind, schwächer gewichtet werden als solche, die diesem näher sind. Das Histogramm kann auch als Vektor verstanden werden, bei dem jedem der 16 Teilbereiche 8 Vektoren zugeordnet sind, deren Länge jeweils die Intensität des Helligkeitsverlauf in diese Richtung angeben. Abbildung x zeigt auf der rechten Seite den so entstandenen 128-dimensionalen Merkmalsvektor.

Um die Invarianz bzgl. der Rotation herzustellen, wird jeweils die Orientierung des *Keypoints* von den ermittelten Orientierungen subtrahiert. Die Helligkeitsinvarianz wird dagegen durch eine Normalisierung gewährt, bei der man einen oberen Schwellenwert für die auftretenden Vektoren festlegt (Lowe, 2004, Andersson & Marquez, 2016).

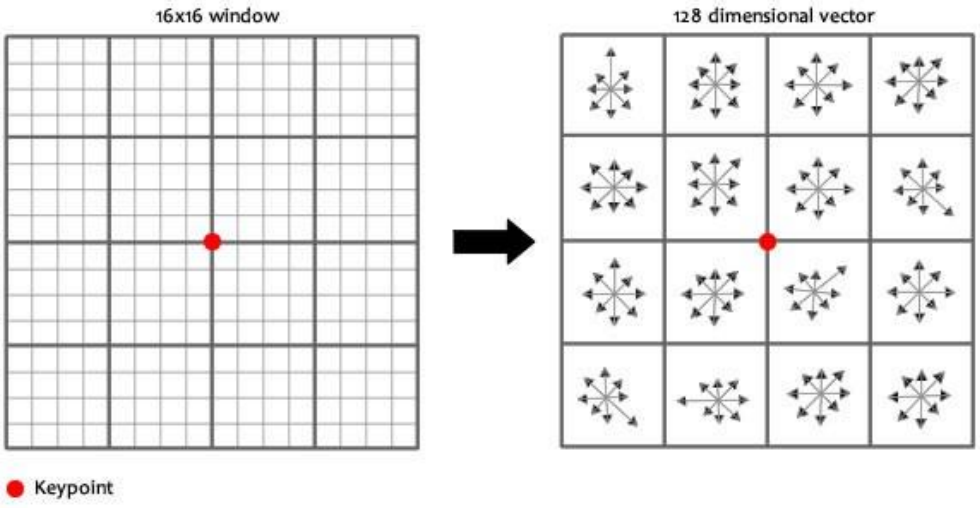

Abbildung x: Generierung eines 128-dimensionalen Vektors für einen Keypoint.[7]

3.4. Weitere Algorithmen zur Merkmalserkennung

Auch über 20 Jahre nach seiner ersten Veröffentlichung wird der SIFT-Algorithmus noch häufig zu Zwecken der Merkmalserkennung eingesetzt. In der Zwischenzeit haben sich jedoch zahlreiche weitere Algorithmen zu diesem hinzugesellt, deren Schöpfer den Anspruch haben, SIFT hinsichtlich Erkennungsgenauigkeit oder Geschwindigkeit zu übertreffen. Diese sollen im Folgenden näher beschrieben werden.

3.4.1. Speeded Up Robust Features (SURF)

Der *Speeded Up Robust Features*-Algorithmus kann als eine Weiterentwicklung von SIFT verstanden werden. Das Hauptziel bei der Entwicklung von SURF war dabei die Erhöhung der Berechnungsgeschwindigkeit gegenüber SIFT bei gleichzeitiger Beibehaltung von dessen hoher Erkennungsrate.

Eine der beiden Hauptneuerungen stellt der *Fast-Hessian-Detector* dar, der die genaue Berechnung der zweiten Gaußschen Ableitung durch eine Approximation ersetzt. Hierbei bedient man sich Boxfiltern und Integralbildern, um zum gewünschten Ergebnis zu kommen. Das Ergebnis der Approximation ist in Abbildung x zu betrachten.

[7] https://medium.com/analytics-vidhya/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40 (Letzter Zugriff: 30.3.2020)

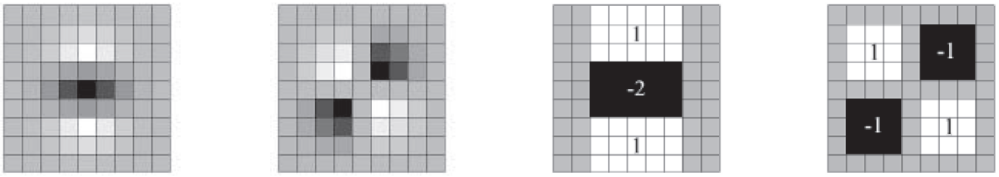

Abbildung x: Gaußsche Ableitungen (links) und deren Approximation durch Boxfilter (rechts) (Bay, Tuytelaars & Van Gool, 2006).

Als weiterer wichtiger Unterschied zu SIFT kann die Verwendung des neuen SURF-Deskriptors gelten. Um die Komplexität der Berechnung, und damit deren Dauer, zu verringern, wurde die Dimensionalität des Deskriptors verringert. Als erster Schritt werden dabei in einem kreisförmigen Nachbarschaftsbereich um den *Keypoint* Filterantworten anhand von *Haar-Wavelets* berechnet. Nachdem man auf diese Weise eine Orientierung ermittelt hat, wird nun eine quadratische Region um den Keypoint festgelegt, die um den Wert der Orientierung rotiert ist (siehe Abbildung x). Diese Region wird nun in Unterregionen mit je 4 x 4 Pixeln geteilt. Für diese wird, ebenfalls unter Verwendung von *Haar-Wavelets*, ein vierdimensionaler Beschreibungsvektor berechnet. Jeder *Keypoint* verfügt somit lediglich über einen 64-dimensionalen Deskriptor, während die Dimensionalität von SIFT bei 128 liegt (Bay, Tuytelaars & Van Gool, 2006).

Abbildung x: Von SURF verwendete Haar-Wavelets (links) und quadratische Regionen um Keypoints (Bay, Tuytelaars & Van Gool, 2006).

3.4.2. Binary Robust Invariant Scalable Keypoints (BRISK)

Leutenegger, Chli und Siegwart bauen mit ihrem *Binary Robust Invariant Scalable Keypoints*-Algorithmus auf SIFT und SURF auf. Auch sie sind primär an einer Erhöhung der Berechnungsgeschwindigkeit interessiert, während bezüglich der Treffergenauigkeit lediglich eine Äquivalenz zu SIFT und SURF angestrebt wird. Zwar ist bezüglich des Ablaufs des Algorithmus eine signifikante Ähnlichkeit zu SIFT und SURF zu beobachten, es existieren jedoch auch nennenswerte Unterschiede, etwa die Verwendung von FAST zur Ermittlung von Keypoints sowie des binären Deskriptors BRIEF.

Der FAST-Algorithmus (*Features from Accelerated Segment Test*) von Rosten, Porter & Drummond ist im Bereich der Eckendetektion anzusiedeln. Wird ein potenzieller Keypoint auf seine Eignung hin untersucht, erfolgt ein Vergleich mit 16 Pixeln, die alle auf einem Kreis um diesen Punkt liegen (siehe Abbildung x). Liegen auf diesem Kreis eine Mindestzahl von zusammenhängenden Pixeln, die allesamt heller oder niedriger als der Mittelpunkt sind, so kann der Punkt als geeignet gelten (Rosten, Porter & Drummond, 2008).

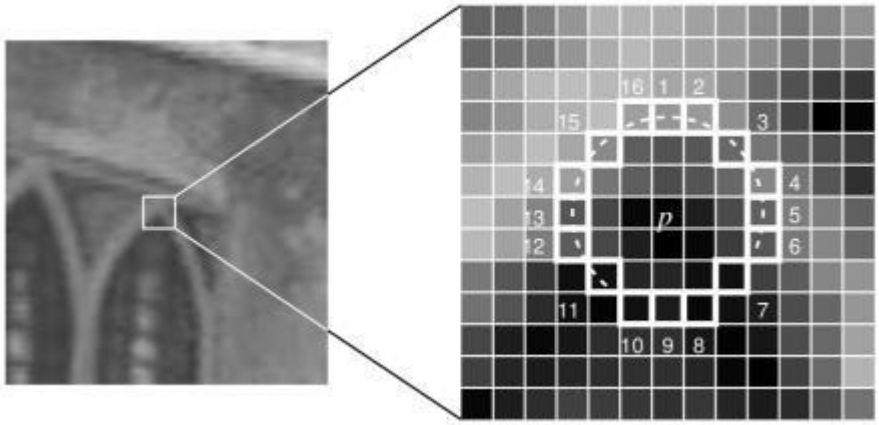

Abbildung x: Identifizierung von Keypoints durch Vergleich mit Kreispunkten (Rosten, Porter & Drummond, 2008).

BRISK verwendet nun eine Modifikation des FAST-Algorithmus namens AGAST, die eine noch weiter erhöhte Geschwindigkeit verspricht. Um die Invarianz gegenüber der Skalierung sicherzustellen, erfolgen die Vergleiche mit den auf dem Kreis liegenden Punkten nicht nur innerhalb eines einzigen Bildes, sondern, analog zu SIFT, zusätzlich mit Bildern anderer Oktaven, wobei hier außerdem sogenannte Interoktaven zur Anwendung kommen (Leutenegger, Chli & Siegwart, 2011).

Der Deskriptor BRIEF zeichnet sich dadurch aus, dass er Informationen über die zu beschreibenden Merkmale in Form binärer Zeichenketten abspeichert. Dadurch ergeben sich sowohl bei der Generierung als auch beim *Matching* deutliche Zeiteinsparungen. Statt die Deskriptoren zuerst in herkömmlicher Form zu generieren und anschließend in Binärcode umzuwandeln, wird dieser bei BRIEF direkt erzeugt. Dabei wird der Keypoint mit einer festen Zahl von Punkten verglichen, die in ebenfalls festen Abständen voneinander auf konzentrischen Kreisen liegen, welche den Keypoint umgeben. Abbildung x zeigt exemplarisch die Anordnung der Punkte im Rahmen von BRISK. Beim Vergleich wird nun untersucht, ob entweder der Keypoint oder der Vergleichspunkt einen höheren Helligkeitswert haben. Ist der Wert des Vergleichspunkts höher, wird im Deskriptor an dieser Stelle 1 eingetragen, ansonsten 0. Insgesamt hat die binäre Zeichenkette des BRIEF64-Deskriptors eine Länge von nur 512 Bit, was einer

achtfachen Verkleinerung gegenüber SIFT und einer vierfachen gegenüber SURF gleichkommt. Abschließend ist noch zu erwähnen, dass BRIEF alleine keine rotationsinvarianten Deskriptoren erzeugt. Die Invarianz muss deshalb vom verwendenden Algorithmus hergestellt werden, was dann auch bei BRISK der Fall ist (Leutenegger, Chli & Siegwart, 2011, Calonder, 2010).

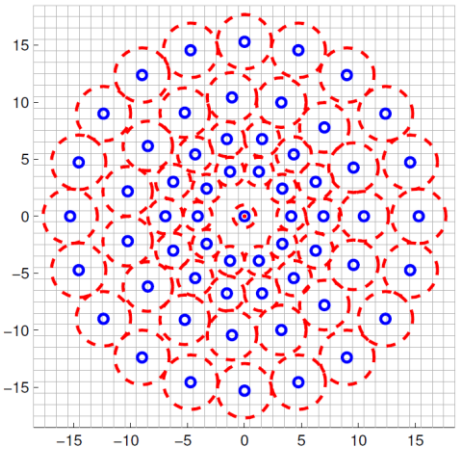

Abbildung x: Von SURF verwendete Haar-Wavelets (links) und quadratische Regionen um Keypoints (Leutenegger, Chli & Siegwart, 2011).

3.4.3. Oriented FAST and Rotated BRIEF (ORB)

Ein weiterer Algorithmus, der sich der Kombination aus FAST und BRIEF bedient, ist der im gleichen Jahr wie BRISK publizierte ORB von Rublee et al. Auch hier steht wieder die Geschwindigkeitserhöhung im Vordergrund.

Da FAST keine Orientierung für *Keypoints* ermittelt, haben die Autoren den Algorithmus modifiziert und oFAST (*Oriented* FAST) benannt. Hierbei bedienen sie sich zuerst der *Harris Corner Detection*, um Punkte, die keine Ecken sind, auszuschließen. Anschließend wird eine Bildpyramide erzeugt, um die Invarianz bezüglich der Skalierung herzustellen. Nun wird zwischen dem Mittelpunkt einer Ecke und dem geometrischen Schwerpunkt ihrer Helligkeitsintensität unterschieden. Aus der Richtung des Vektors zwischen Mittelpunkt und Schwerpunkt ergibt sich schließlich die Orientierung des Keypoints.

Auch bei der verwendeten Variante von BRIEF (rBRIEF) spielt die Rotationsinvarianz eine entscheidende Rolle. Statt den Wert des Keypoints mit zufällig generierten Punkten auf umliegenden Kreisen zu vergleichen, arbeitet nBRIEF mit vorgegebenen Punktmustern. In einem ersten Schritt werden durch Matrixmultiplikation orientierte Deskriptoren (sBRIEF bzw. steered BRIEF) erzeugt. Dies gewährleistet zwar eine höhere Invarianz gegenüber der Rotation, verringert jedoch auch die Streuung der Deskriptorenwerte. Mittels eines Greedy-Algorithmus wählt man deshalb solche Keypoints aus, die sich durch

Vielfalt und fehlende Korrelationen untereinander auszeichnen (Rublee et al., 2011, Andersson & Marquez, 2016).

3.4.4. KAZE Features und Accelerated KAZE (AKAZE)

Der von Alcantarilla, Nuevo und Bartolli entwickelte KAZE Features-Algorithmus unterscheidet sich in mehrfacher Hinsicht von den vorgenannten Ansätzen zur Verbesserung der Merkmalsextraktion. Einerseits steht hier nicht die Geschwindigkeit im Vordergrund, sondern primär die Qualität der Merkmalserkennung. Andererseits setzen die Autoren bei der Optimierung auch an einer bislang wenig beachteten Stelle an: Während die bisher genannten Algorithmen sich der Gaußschen Weichzeichnung bedienen, um den *Scale Space* zu erzeugen, wird bei (A)KAZE hierfür die non-lineare Diffusion verwendet. Als Grund geben die Autoren an, dass die Gaußsche Weichzeichnung die natürlichen Grenzen von Objekten nicht respektiert und somit Details und Rauschen in gleichem Maße weichzeichnet, was in Abbildung X verdeutlicht wird.

Abbildung x: Erzeugung des *Scale Space* mit Linearer Diffusion (oben) und Non-linearer Diffusion (unten). (Alcantarilla, Bartoli & Davison, 2012)

Analog zu SIFT wird der *Scale Space* auch hier durch die Erzeugung von Oktaven konstruiert, deren Bestandteile Bilder mit zunehmender Weichzeichnung sind. Die Berechnung der linearen Diffusionsgleichungen wird hier aber nur approximiert. Dies erfolgt mittels *Additive Operator Splitting* (AOS), wobei der Berechnungsaufwand weiterhin erheblich ist. Für die weitere Berechnung der Keypoints wird auch hier die Hesse-Matrix verwendet, während als Deskriptor eine Variation von SURF (m-SURF) gewählt wird. (Alcantarilla, Bartoli & Davison, 2012)

Aufgrund der hohen Berechnungsaufwandes veröffentlichte Alcantarilla im folgenden Jahr eine überarbeitete Version des KAZE-Algorithmus namens AKAZE. Dieser bedient sich der mathematischen Technik des *Fast Explicit Diffusion* anstelle von AOS, um die Berechnung des *Scale Space* in deutlich kürzerer Zeit zu ermöglichen. (Alcantarilla, Nuevo & Bartoli, 2011)

3.5. Matching

Nach Abschluss der Merkmalsextraktion liegt unabhängig vom verwendeten Algorithmus für das Ausgangsbild eine Menge von Keypoints vor. Es bieten sich nun drei Arten von Bildvergleichen an:

- Zwei Bilder werden direkt miteinander verglichen, um die Gleichheit ihrer Motive anhand einer bestimmten Vergleichsmetrik zu bestimmen.
- Ein Bild wird mit einer größeren Zahl von Bildern in einer Datenbank verglichen. Hierbei wird idealerweise nicht für jedes einzelne Bild eine Neuberechnung der Merkmalsvektoren durchgeführt. Stattdessen werden diese Merkmalsvektoren selbst in der Datenbank gespeichert.
- Es wird nur ein kleiner Ausschnitt eines Bildes als sog. Template definiert, etwa wenn das gesuchte Objekt nur einen Teil des Bildes ausfüllt und der Rest der Aufnahme für den Vergleich als irrelevant eingestuft wird. Andere Bilder werden lediglich mit diesem Template verglichen, wobei diese weiterhin in voller Größe verwendet werden (Gollapudi, 2019).

Möchte man zwei Bilder auf ihre Ähnlichkeit hin überprüfen, erfolgt dies durch den Vergleich ihrer *Keypoints*. Dabei können zwei unterschiedliche Strategien zum Einsatz kommen: Das *Brute-Force*-Matching sowie das FLANN-Matching. Ersteres bedeutet, dass jeder *Keypoint* mit jedem *Keypoint* des anderen Bildes verglichen wird. Dies ist mit einem hohen Rechenaufwand verbunden, garantiert jedoch, dass unter allen potenziellen Matches tatsächlich die besten gefunden werden. Alternativ dazu wird beim FLANN-Verfahren (*Fast Library for Approximate Nearest Neighbours*) eine Auswahl an Matches getroffen, wobei *Nearest-Neighbour*-Suchtechniken ebenso zur Anwendung kommen wie k-d-Bäume. Die damit verbundenen deutlichen Geschwindigkeitszugewinne werden jedoch generell durch eine geringere Qualität der Ergebnisse erkauft (Minichino & Howse, 2015, „Feature Matching“, n.d.).

Beim direkten Vergleich zweier *Keypoints* wird jeweils der euklidische Abstand zwischen diesen Punkten ermittelt. Als Match kann derjenige *Keypoint* des anderen Bildes gelten, zu dem der euklidische Abstand am geringsten ist.

Wirklich? Ist das schon alles?

Hierbei ergibt sich jedoch das Problem, dass auch in Bildpaaren ohne gemeinsame Inhalte derartige falsch positive Übereinstimmungen auftreten. Zwar könnte man versuchen, dies durch die Festlegung eines globalen Schwellenwerts für den euklidischen Abstand zu verhindern, doch wird dieser Ansatz der heterogenen Natur der Deskriptoren kaum gerecht. Stattdessen wendet man ein Verfahren an, das erstmals von Lowe beschrieben wurde: Dabei

betrachtet man das Verhältnis zwischen dem kleinsten und zweitkleinsten Abstand und entfernt Matches, bei denen dieses Verhältnis zu groß ist. Matches, deren Distanzverhältnis (*Distance Ratio*) einen bestimmten Schwellenwert – Lowe empfiehlt hier den Wert 0,8 - unterschreitet, können als Gute Matches gelten (Lowe, 2004, Dawson-Howe, 2015).

Für Algorithmen, die sich binärer Deskriptoren bedienen - etwa AKAZE, ORB und BRISK - hat sich stattdessen die Berechnung des Hamming-Abstands zwischen den Deskriptoren bewährt (Tareen & Saleem, 2018).

Doch selbst durch diese Maßnahmen kann keine endgültige Gewissheit bestehen, dass beim Vorliegen eines Guten Matches tatsächlich ein identisches Objekt bzw. ein Bestandteil desselben auf beiden Bildern zu erkennen ist. Zwar können auf Basis der gefundenen Matches Bilder generiert werden, die die Übereinstimmungen etwa durch Verbindungslinien zwischen den korrespondierenden Punkten darstellen. Ebenfalls können aus den Matches sogenannte Homographie-Matrizen generiert werden, mit denen die Bilder perspektivisch so transformiert werden können, dass die Matches auf beiden Bildern an der gleichen Stelle liegen. Auf diese Weise ist es mit einiger Sicherheit möglich, vorliegende Matches per Hand auf ihre Richtigkeit zu überprüfen (Tareen & Saleem, 2018).

Für den Umgang mit größeren Datenmengen, insbesondere für deren statistische Auswertung, erscheint dieses manuelle Vorgehen jedoch ungeeignet, so dass die Guten Matches im Sinne von Lowes *Distance Ratio* hier zu bevorzugen sind. Es gilt deshalb, für den jeweiligen Anwendungsfall zu untersuchen, welche Abstände zwischen den *Keypoints* bei abweichenden sowie identischen Motiven zu erwarten sind und wie diese durch die Umstände der Bildkomposition beeinflusst werden. Das genaue Vorgehen für das in dieser Arbeit untersuchte Anwendungsbeispiel wird in Kapitel x detailliert beschrieben.

3.6. Performancevergleiche der Algorithmen

In der Forschungsliteratur finden sich bereits unterschiedliche Versuche, die Performance der Merkmalserkennungs-Algorithmen miteinander zu vergleichen. Im Rahmen dieser Arbeit sind hierbei besonders diejenigen Vergleiche von Bedeutung, bei denen Gebäude als Vergleichsobjekte gewählt wurden. Darüber hinaus können jedoch auch generelle Untersuchungen der Robustheit der Algorithmen hinsichtlich der Invarianz der Aufnahmebedingungen hilfreich sein.

3.6.1. Gebäudeklassifikation

Grundsätzlich sind fast alle Bildmotive für den Bildvergleich geeignet, sofern die Aufnahmen ein Mindestmaß an Eckpunkten bzw. Helligkeitsunterschieden aufweisen. Es liegt jedoch nahe, anzunehmen, dass die vorgestellten Algorithmen nicht für alle Motive im gleichen Umfang geeignet sind. Um dieser Frage weiter nachzugehen, haben Tareen & Saleem die Algorithmen SIFT, SURF, KAZE, AKAZE, ORB und BRISK für einer Reihe unterschiedliche Motive getestet und dabei Quantität sowie Qualität der ermittelten Merkmale und Matches sowie die Geschwindigkeit ermittelt.

Von besonderer Relevanz ist hierbei die Tatsache, dass von den elf ausgewählten Bildmotiven zwei aus dem Bereich der Architektur gewählt wurden. Die beiden Bildpaare sind in Abbildung x zu sehen.

Abbildung x: Architektonische Bildmotive als Basis für Performancemessung der Merkmalserkennungs-Algorithmen. Oben: Bildpaar 1 (Building Dataset), unten: Bildpaar 2 (Roofs Dataset) (Tareen & Saleem, 2018).

Die folgende Tabelle x gibt einige der wichtigsten Messwerte an, die von den Autoren ermittelt wurden. Die Varianten 128D und 64D von SURF verwenden jeweils unterschiede Deskriptorenlängen. Bei BRISK(1000) und ORB(1000) wurden die Algorithmen mit einer Beschränkung für die maximale Zahl an berechneten Features versehen, was zwar deutliche Geschwindigkeitsverbesserungen bringt, jedoch auch die Zahl gefundener Matches signifikant verringert.

TABELLE X. PERFORMANCE VON MERKMALSERKENNUNGS-ALGORITHMEN

Algorithmus	Anzahl Matches Bildpaar 1 (Building)	Anzahl Matches Bildpaar 2 (Roofs)	Gesamtzeit Matching (Bildpaar 1)	Gesamtzeit Matching (Bildpaar 2)
SIFT	384	423	0,5186 s	0,7186 s
SURF(128D)	319	171	0,8940 s	0,6129 s
SURF(64D)	612	247	0,6367 s	0,4606 s
BRISK	481	436	0,2390 s	0,5905 s
BRISK(1000)	190	90	0,0586 s	0,0613 s
ORB	854	498	0,2086 s	0,4899 s
ORB(1000)	237	91	0,0391 s	0,0385 s
KAZE	465	172	0,4924 s	0,5162 s
AKAZE	475	175	0,1772 s	0,1839 s

Tab. 2. Übersicht über Messwerte, die beim Performancevergleich von Merkmalserkennungs-Algorithmen ermittelt wurden (Tareen & Saleem, 2018).

Während die Werte für die Gesamtzeit des Matchings bei der Auswahl eines geeigneten Algorithmus sicherlich behilflich sein können, ist bei der Betrachtung der Anzahl gefundener Matches jedoch Vorsicht geboten. Deren Zahl enthält nämlich auch falsch positive Funde. So bescheinigen denn auch die Autoren nach manueller Prüfung der Ergebnisse, dass SIFT die höchste Treffergenauigkeit aufweist, obwohl die absolute Anzahl gefundener Matches dies auf den ersten Blick nicht nahelegen würde (Tareen & Saleem, 2018).

3.6.2. Invarianz-Tests

Andersson und Marquez haben für ihre Studie Aufnahmen von Objekten durchgeführt und jeweils deren Rotation, Skalierung und Beleuchtung variiert. Anschließen ermittelten sie, mit welcher Sicherheit die Algorithmen SIFT, KAZE, AKAZE und ORB in der Lage sind, diese Objekte auf den abweichenden Aufnahmen wiederzuerkennen. Auch wenn es sich bei keinem der Motive um ein Gebäude handelte, so können die Ergebnisse, welche Tabelle X zu entnehmen sind, trotzdem bei der Beurteilung der Algorithmen von Nutzen sein.

TABELLE X. ANTEIL KORREKTER MATCHES FÜR INVARIANZTYPEN

Name	Anteil korrekter Matches - Rotation	Anteil korrekter Matches - Skalierung	Anteil korrekter Matches - Beleuchtung
SIFT	96%	93%	90%
KAZE	85%	78%	95%
AKAZE	88%	84%	100%
ORB	62%	40%	75%

Tab. x. Anteil korrekter Matches der Algorithmen für verschiedenen Invarianztypen (Andersson & Marquez, 2016).

Es zeigt sich, dass keiner der Algorithmen den anderen gegenüber in jeder Hinsicht als überlegen gelten kann. Hingegen erscheint die Fähigkeit des ORB-Algorithmus, korrekte Matches zu ermitteln, generell geringer ausgeprägt zu sein. Für ORB spricht hingegen, dass dieser im Rahmen der Tests im Mittel mehr als zehn Mal schneller war als SIFT und AKAZE und mehr als hundert Mal so schnell wie KAZE (Andersson & Marquez, 2016).

4. Forschungsstand: Technisch

4.1. OpenCV

Bei der Entwicklung von Software, die sich der Merkmalserkennung bedient, kann auf unterschiedliche Weise vorgegangen werden. Eine gangbare Option ist sicherlich, einen der vorgestellten Algorithmen eigenständig zu implementieren. Ein Beispiel für diese Vorgehensweise ist die mobile Bildklassifikations-Applikation, die Groeneweg et al. im Jahr 2006 vorgestellt haben, die mit einer modifizierten Version von SIFT die Performance-Beschränkungen damaliger Mobiltelefone zu umgehen sucht (Groeneweg et al., 2006).

Üblicherweise wird heute jedoch zu Zwecken der Merkmalserkennung auf bestehende Softwarebibliotheken zurückgegriffen, wobei in der Regel *OpenCV* zum Einsatz kommt. Diese quelloffene Bibliothek kann für vielfältige Anwendungszwecke im Bereich der *Computer Vision* und Bildbearbeitung verwendet werden. OpenCV ist auf zahlreiche Plattformen einsetzbar, etwa auf *Windows*, *Linux*, *macOS*, *Android und iOS*. Es kann darüber hinaus mit Programmiersprachen wie *Python*, *Java* und C++ genutzt werden.

Mit der Bibliothek *OpenCV.js* steht auch einer Portierung für *JavaScript* zur Verfügung, die jedoch nur über eine eingeschränkte Zahl von Funktionen verfügt. Speziell die Algorithmen zur Merkmalserkennung stehen hierfür nur eingeschränkt zur Verfügung, jedoch sind BRISK und ORB bereits nutzbar.[8] Alternativ dazu bietet die Bibliothek *jsfeat* die Möglichkeit, FAST und ORB in einer reinen *JavaScript*-Anwendung zu verwenden.[9]

Die bereits erwähnten Unterschiede zwischen den Algorithmen bezüglich des Lizenzrechts spielen auch bei der Arbeit mit *OpenCV* eine wichtige Rolle. Während die Copyright-geschützten SIFT und SURF in früheren Versionen der Bibliothek noch ohne Mehraufwand einsetzbar waren, ist deren Verwendung seit Version 3 nicht mehr möglich. Die als „non-free" gekennzeichneten Algorithmen können seitdem nur noch in der Bibliothek *opencv_contrib* verwendet werden, welche Module enthält, die nicht Teil der offiziellen Distribution sind.[10]

Neben den genannten Merkmalserkennungs-Algorithmen bietet *OpenCV* auch eine Reihe von unterschiedlichen Matching-Verfahren an. Dabei kann zwischen den folgenden *Descriptor-Matcher*-Algorithmen gewählt werden, die sich in die

[8] https://github.com/ucisysarch/opencvjs (Letzter Zugriff: 30.3.2020)
[9] https://inspirit.github.io/jsfeat (Letzter Zugriff: 30.3.2020)
[10] https://github.com/opencv/opencv_contrib (Letzter Zugriff: 30.3.2020)

beiden Kategorien *Flann-Based-Matching* und *Brute-Force-Matching* einordnen lassen:[11]

- FLANNBASED
- BRUTEFORCE
- BRUTEFORCE_L1
- BRUTEFORCE_HAMMING
- BRUTEFORCE_HAMMINGLUT
- BRUTEFORCE_SL2

11 https://docs.opencv.org/3.4/db/d39/classcv_1_1DescriptorMatcher.html (Letzter Zugriff: 30.3.2020)

5. Lösungsansatz

Bereinigen. Bisher nur Copy & Paste aus anderen Bereichen!

Nachdem die wichtigsten Algorithmen zur Merkmalserkennung detailliert vorgestellt und die Anforderungen an eine Lösung im Rahmen des gegebenen Anwendungsbeispiels definiert wurden, muss nun ein Vorgehen gefunden werden, mit dem man den geeignetsten Algorithmus bestimmen kann. Einer der wichtigsten Aspekte bei der Umsetzung der Anwendung ist die Frage, ab wann zwei Bilder als Repräsentation des gleichen Objekts gelten können. Als Ergebnis der Bildvergleichs liefern sämtliche Merkmalserkennungs-Algorithmen lediglich eine Menge von Matches. Aus diesen können mittels x diejenigen Matches entnommen werden, welche mit hoher Wahrscheinlichkeit als korrekt gelten können (siehe Kapitel x). Doch aus der Anzahl dieser guten Matches allein lässt sich noch keine Aussage über die Richtigkeit der Bildklassifikation liefern.

Hierfür ist es stattdessen erforderlich, für den jeweiligen Anwendungskontext zu ermitteln, welche Anzahl an guten Matches erwartet werden kann. Insbesondere sind hier die folgenden zwei Situationen zu prüfen, die im Kapitel zur Evaluierung intensiv betrachtet werden sollen:

- Vergleich von zwei Bildern mit unterschiedlichen Motiven
- Vergleich von zwei Bildern des gleichen Motivs bei Varianz der Aufnahmebedingungen

Falls die Anzahl Guter Matches in den beiden Kategorien generell zu nahe beieinander liegt, muss der Algorithmus dabei als ungeeignet eingestuft werden. Es ist ebenfalls damit zu rechnen, dass es bei der Messung zu Ausreißern kommt. So kann es einerseits zu einer großen Zahl Guter Matches bei Bildern unterschiedlicher Motive kommen, während andererseits auch bei Bildern des gleichen Motivs nur eine kleine Zahl Guter Matches auftreten kann. Ist bei einem Algorithmus eine größere Zahl solcher Ausreißer zu beobachten, ist dies bei der Bewertung ebenfalls negativ zu berücksichtigen.

Algorithmen mit Vor- und Nachteilen vorstellen auf Basis von Evaluation.

Gründe für SIFT. Hier oder Fazit?

6. Umsetzung

BIdent Building Identification ist eine plattformunabhängige Web-Applikation, mit der BenutzerInnen plattformübergreifend photographische Aufnahmen von historischen Gebäuden und deren Bauteilen machen und diese automatisch identifizieren lassen können. Als möglicher Auftraggeber kommen etwa Tourismusbehörden in Frage, die mittels der Applikation die Popularität lokaler Sehenswürdigkeiten vergrößern möchten.

6.1. Architektur

Der architektonische Aufbau, schematisch dargestellt in Abbildung X, folgt dabei dem Client-Server-Modell. Über den Client bzw. Web-Browser werden mit der Kamera des Geräts Aufnahmen erstellt und mittels eines HTTP-POST-Requests an den Server übertragen. Dabei wird auch die momentane geographische Position des Geräts übermittelt. Der Server bezieht nun sämtliche Bilder aus seiner Datenbank und filtert diejenigen heraus, die sich in der Umgebung der Geräteposition befinden. Auf die verbleibenden Bilder wird der gewählte Merkmalserkennungs-Algorithmus angewandt. Für das Objekt mit den besten Matching-Ergebnissen wird anschließend eine HTTP-Response im JSON-Format an den Client übermittelt. Diese enthält nicht nur Textinformationen über das Gebäude bzw. Bauteil sondern auch eine Identifikationsnummer, anhand derer der Client die zugehörige Bilddatei vom Server bezieht.

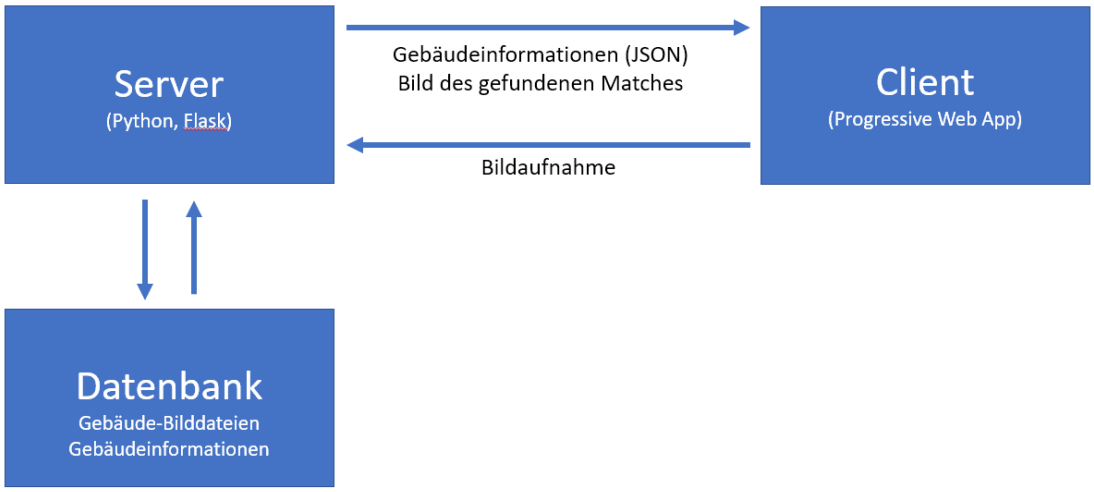

Abbildung x: Architektur von *BIdent Building Identification*.

6.2. Client

Um die Verwendung auf möglichst vielen Geräten zu ermöglichen, wurde die Client-Applikation als *Progressive Web App* verwirklicht. Sie kann also sowohl innerhalb eines Web-Browsers ausgeführt werden als auch als eigene App auf mobilen Betriebssystemen wie Android und iOS installiert werden. In jedem Fall wird der Code der Anwendung jedoch auf einem Web-Server vorgehalten. Um trotzdem die Offline-Fähigkeit zu gewährleisten, wird deshalb ein *Service Worker* für das Caching der für die Ausführung notwendigen Dateien eingesetzt.

Die graphische Benutzeroberfläche des Clients wurde mit HTML5, CSS und JavaScript umgesetzt, wobei die Prinzipien des *Responsive Design* zur Anwendung kamen. Auf der Hauptseite, welche in Abbildung X zu sehen ist, wird als interaktives UI-Element lediglich ein Aufnahme-Button angezeigt, welcher den Input der Gerätekamera überlagert. Nach dem Betätigen dieses Buttons wird nach einer gewissen Wartezeit eine Seite mit Details über das Gebäude bzw. Bauteil angezeigt.

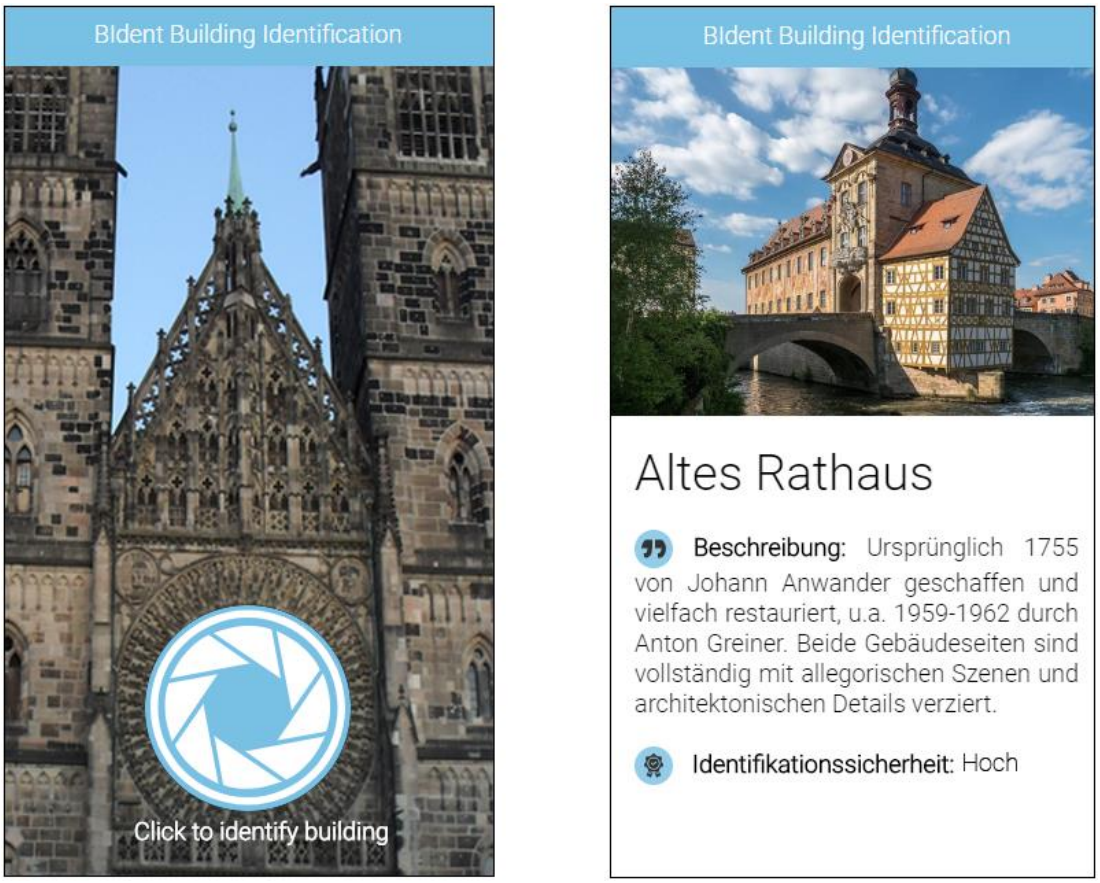

Abbildung x: Screenshots der Applikation *BIdent Building Identification*: Hauptseite (links) und Detailseite (rechts).

Sowohl die Anzeige des Kamera-Inputs als auch die Aufnahme werden dabei über die *MediaDevices* API gewährleistet, die Teil von HTML5 ist. Vor der Übermittlung an den Server ist jedoch noch eine Umwandlung des Bildes erforderlich. Da anfangs nur die zugehörige *data URI* verfügbar ist, müssen deren Daten zuerst extrahiert und dann in ein Blob-Objekt umgewandelt werden.

6.3. Server

Der Server besteht in erster Linie aus einer *Flask*-Applikation, die mit *Python* umgesetzt wurde. Diese greift auf eine *MySQL*-Datenbank zu, in der weiterführende Informationen zu den jeweiligen Bildern gespeichert sind. Die Bilddateien werden hingegen nicht in der Datenbank gespeichert, sondern in einem Upload-Verzeichnis auf dem Server. Der Abruf ergibt sich über die im Dateinamen enthaltene Gebäude-Id.

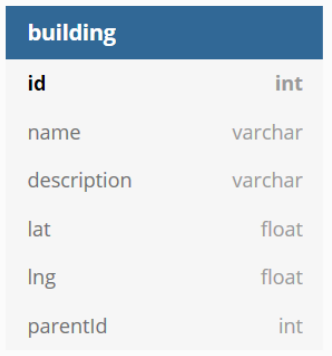

Abbildung x: Spalten der Gebäude-Tabelle *building* in MySQL-Datenbank.

Nachdem der Server eine HTTP-POST-Anfrage entgegengenommen hat, werden daraus die Geodaten des anfragenden Geräts sowie das Bild in Form eine Blob-Objekts entnommen. Nun werden alle Bilder aus der Datenbank entnommen und diejenigen herausgefiltert, die sich innerhalb einer *Bounding Box* um die Geräteposition befinden. Auf die Verwendung eines *Spatial Database* wurde im ersten Schritt verzichtet, da die zu erwartenden Performancegewinne im gegebenen Anwendungskontext den zusätzlichen Implementierungsaufwand nicht aufwiegen dürften.

Nun werden die Deskriptoren der zu vergleichenden Bilder unter Benutzung des SIFT-Algorithmus berechnet. Das Matching erfolgt durch einen *FLANN-based Matcher* während auf die Verwendung einer Homographie verzichtet wurde.

Neben der Rückgabe des wahrscheinlichsten Gebäude-Objekts ist es auch erforderlich, den BenutzerInnen mitzuteilen, mit welcher Sicherheit das Programm die Richtigkeit der Bildklassifikation einstuft. In Tabelle X ist zu sehen, welche Anzahl Guter Matches beim SIFT-Algorithmus zu erwarten ist, wenn zwei

Bilder unterschiedliche bzw. identische Motive aufweisen. Die Werte 62 und 109 entsprechen dabei dem obersten Dezilwert und Maximalwert für die Anzahl guter Matches bei Bildern unterschiedlicher Motive. Der Wert 192 ist der Minimalwert für gute Matches bei Bildern mit identischen Motiven.

Auf Basis der eher stichpunkthaften Datengrundlage erscheint es nicht angebracht, konkrete Prozentwerte für die Wahrscheinlichkeit der korrekten Bildidentifikation zu berechnen bzw. anzugeben. Stattdessen erfolgt eine qualitative Einstufung in die Kategorien „Niedrige“, „Mittel“ und „Hoch“. Wird kein Objekt gefunden, für das die Anzahl der Guten Matches den Wert 62 übersteigt, so wird davon ausgegangen, dass das fotografierte Gebäude bzw. Bauteil keine Entsprechung in der Bilddatenbank hat.

TABELLE X. KATEGORIEN FÜR SICHERHEIT DER BILDKLASSIFIKATION – SIFT-ALGORITHMUS

Sicherheitskategorie	**Unterer Grenzwert**	**Oberer Grenzwert**
Kein Objekt gefunden	0	62
Niedrig	62	109
Mittel	109	192
Hoch	192	∞

Tab. x. Kategorien für die Sicherheit der Bildklassifikation mit dem SIFT-Algorithmus.

Wurde ein Objekt ermittelt und eine Sicherheitseinstufung vorgenommen, werden die gewonnen Werte nun in ein JSON-Objekt umgewandelt, welches abschließend als Response an den Client zurückgegeben wird.

Auf dem Server existiert des Weiteren noch eine simple CRUD-Oberfläche für die Verwaltung der Gebäude- bzw. Bilddateien und eine Möglichkeit, Bilddateien anhand ihrer Id bereitzustellen. Die Kommunikation mit der Datenbank bzw. dem Dateisystem des Servers erfolgt dabei ebenfalls über die Flask-Applikation.

7. Evaluierung

Ziel dieses Kapitels ist es, die Anzahl Guter Matches für den Bildvergleich zu ermitteln – einerseits bei Bildern mit abweichenden Motiven, andererseits bei Bildern identischer Motive mit abweichenden Aufnahmebedingungen. Durch den Vergleich der Menge der ermittelten Guten Matches lassen sich Wertebereiche für jeden Algorithmus definieren, anhand derer die Sicherheit eines konkreten Identifikationsergebnisses bestimmt werden kann. Darüber hinaus soll auch die Geschwindigkeit der Berechnung Guter Matches zwischen den Algorithmen verglichen werden. Abschließend erfolgt eine Bewertung, inwiefern die Algorithmen für den Anwendungsfall geeignet sind.

7.1. Matches bei abweichenden Motiven

Im Folgenden soll versucht werden, einen Vergleichswert für die Anzahl guter Matches bei abweichenden Bildmotiven zu ermitteln. Als Grundlage wurde das *Oxford Buildings Dataset* gewählt. Die 5062 enthaltenen Bilder wurden jeweils durch eine Suche auf der Plattform *Flickr* nach den Namen wichtiger Gebäude in der Stadt Oxford ermittelt. Aus diesem Grund enthalten viele der Bilder nur Innenaufnahmen des Gebäudes, oder zeigen ein anderes Motiv, das lediglich von diesem Gebäude aus aufgenommen wurde (Philbin, J., Arandjelović, R. und Zisserman, (n.d.), Li et al., 2014). Hieraus ergibt sich jedoch der Vorteil, dass auch für das Matching von Gebäuden mit gänzlich unähnlichen Objekten nützliche Messergebnisse erzielt werden können.

Sämtliche Fotografien des *Oxford Buildings Dataset* wurden jeweils für jeden der sechs Merkmalserkennungs-Algorithmen mit einem Vergleichsbild des Alten Rathauses in Bamberg verglichen. Ein bemerkenswertes Ergebnis der Berechnung ist, dass bei einigen wenigen Bildern eine auffallend hohe Zahl Guter Matches ermittelt wurde, obwohl mit dem bloßen Auge weder eine Übereinstimmung noch eine Ähnlichkeit der Motive erkennbar ist. Diese Ausreißer traten jeweils nur für einen einzigen Algorithmus auf. Aus Abbildung x lässt sich jedoch entnehmen, dass derartige Motive in der Anwendungspraxis nicht unbedingt zu erwarten sind. Nichtsdestotrotz weist ihre Existenz darauf hin, dass auch eine hohe Anzahl Guter Matches keine Garantie für eine tatsächlich vorliegende Übereinstimmung sein kann.

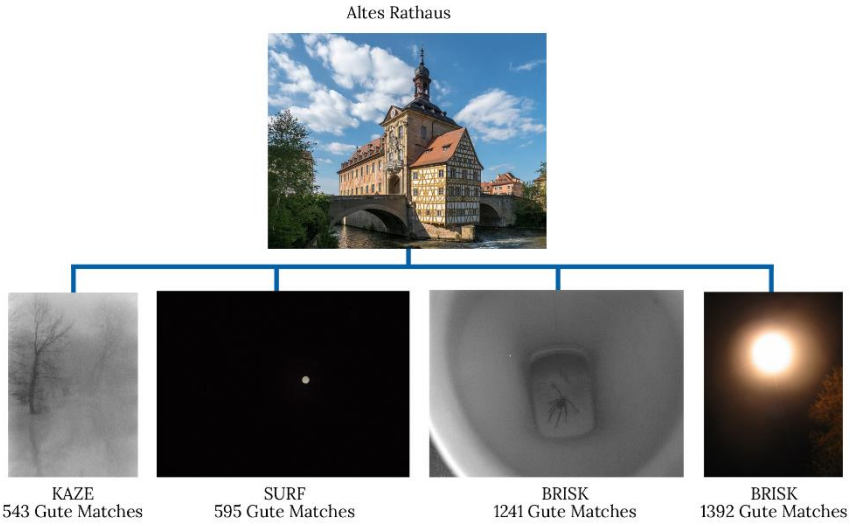

Abbildung x: Ausreißer beim Bildvergleich des Alten Rathauses in Bamberg mit Aufnahmen des *Oxford Buildings Dataset*.

Tabelle X sowie das zugehörige Box-Plot-Diagramm (siehe Abbildung x) liefern eine genauere Übersicht über die Verteilung der Werte. Hierbei fällt auf, dass der BRISK-Algorithmus eine besonders große Spannweite aufweist. Auch bei SURF, KAZE und AKAZE finden sich mehrere Ausreißer, die von den üblichen Werten erheblich abweichen. Zu Zwecken der Evaluierung ist es sicherlich möglich, derartige Bilder nicht in die Berechnung einfließen zu lassen. Es ist jedoch zu erwarten, dass bei der praktischen Anwendung der Identifikations-Applikation auch Aufnahmen erstellt werden, die ebenfalls zu ähnlichen Matching-Ergebnissen führen könnten. Die Wahl eines Algorithmus, der vergleichsweise robust gegenüber solchen Ausreißern ist, erscheint deshalb geeignet, die Ergebnissicherheit zu erhöhen.

Betrachtet man nur diejenigen Werte, die in den unteren neun Dezilen der Merkmalsverteilung liegen (aufgeführt in Spalte P_{90} von Tabelle X), so fällt auf, dass diese für jeden Algorithmus einen deutlichen Abstand zu den Maximalwerten aufweisen.

TABELLE X. STATISTISCHE VERTEILUNG GUTER MATCHES BEIM VERGLEICH MIT BILDERN DES OXFORD BUILDINGS DATASET

Name	**Minimum**	**P_{10}**	**Median**	**P_{90}**	**Maximum**
SIFT	16	39	50	62	109
SURF	20	73	88	104	595
BRISK	6	21	30	43	1392
ORB	0	4	7	12	48
KAZE	0	20	30	41	543
AKAZE	3	16	22	30	327

Tab. x. Statistische Verteilung Guter Matches beim Vergleich mit allen Bildern des *Oxford Buildings Datasets*.

In Abbildung X werden die Ergebnisse der Auswertung in Form eines Boxplots dargestellt, wobei zwei Werte des BRISK-Algorithmus (erstes und zweites Bild von rechts in Abbildung X) nicht angezeigt werden. Es ist zu erkennen, dass die überwiegende Zahl der Ausreißer in der Nähe der Whisker liegen.

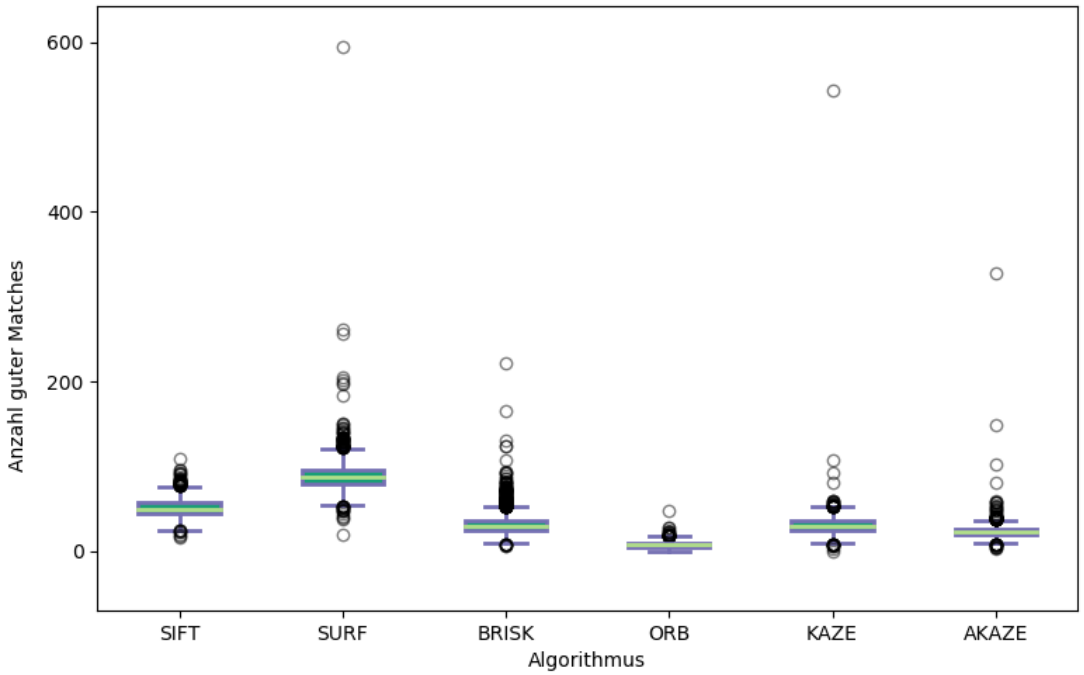

Abbildung x: Boxplots mit Anzahl der Guten Matches für sämtliche Algorithmen beim Vergleich des Alten Rathauses mit allen Bildern des *Oxford Buildings Datasets*.

7.2. Matches bei Varianz der Aufnahmebedingungen

Im nächsten Schritt werden Aufnahmen des gleichen Motivs unter variierenden Aufnahmebedingungen miteinander verglichen. Die auf diese Weise ermittelten Werte müssen zur Bewertung den im vorhergehenden Kapitel ermittelten Werten für abweichend Motive gegenübergestellt werden, um dadurch besonders geeignete Algorithmen zu bestimmen.

7.2.1. Tag und Nacht

Von sechs Objekten wurden am gleichen Tag Aufnahmen erstellt, wobei jeweils eine Fotografie etwa eine Stunde vor und die andere etwa eine Stunde nach Sonnenuntergang aufgenommen wurde. Bezüglich Perspektive bzw. Rotation und Abstand sind die Aufnahmen nicht komplett identisch, was evtl. einen

Einfluss auf die Ergebnisse haben könnte. In Abbildung x sind die verwendeten Bilder zu sehen.

Abbildung x: Gebäude und Bauteile aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich Tag-/Nacht-Unterschieden.

Tabelle x zeigt die Anzahl Guter Matches für die Tag-/Nacht-Bildvergleiche bei den sechs Objekten. Nur beim SIFT-Algorithmus liegen alle Ergebnisse über denen bei abweichenden Motiven aus Kapitel X. Die Algorithmen SURF, BRISK, KAZE und AKAZE liefern stets Werte, die den obersten Dezilwert bei abweichenden Motiven überschreiten. Bei ORB hingegen wurden teilweise Werte ermittelt, die nicht einmal oberhalb des Medians bei abweichenden Motiven liegen.

TABELLE X. ANZAHL GUTER MATCHES – TAG/NACHT

Name	**Matches Brauttor**	**Matches Frauen-kirche**	**Matches Lorenz-kirche**	**Matches Nassauer Haus**	**Matches Altes Rathaus**	**Matches Schür-stabhaus**
SIFT	748	1102	1476	351	964	603
SURF	1590	1442	2304	585	1663	1527
BRISK	475	648	793	78	273	310
ORB	5	15	6	14	17	5
KAZE	163	527	1346	426	312	478
AKAZE	271	493	1408	108	251	331

Tab. x. Anzahl guter Matches für Algorithmen bei Tag-/Nacht-Varianz.

Die für die Berechnung dieser guten Matches benötigte Zeit ist aus Tabelle x zu entnehmen. Für alle Objekte benötigt KAZE die längste und ORB die kürzeste Berechnungszeit.

TABELLE X. DAUER FÜR BERECHNUNG GUTER MATCHES – TAG/NACHT

Name	**Dauer Brauttor**	**Dauer Frauen-kirche**	**Dauer Lorenz-kirche**	**Dauer Nassauer Haus**	**Dauer Altes Rathaus**	**Dauer Schür-stabhaus**
SIFT	2,89 s	3,18 s	2,99 s	2,40 s	2,86 s	2,81 s
SURF	3,45 s	4,35 s	4,17 s	1,90 s	4,05 s	3,68 s
BRISK	0,65 s	1,27 s	0,81 s	1,69 s	0,67 s	0,75 s
ORB	0,14 s	0,16 s	0,14 s	0,12 s	0,15 s	0,14 s
KAZE	8,85 s	9,77 s	9,68 s	8,84 s	9,40 s	9,07 s
AKAZE	1,69 s	2,17 s	2,00 s	1,75 s	1,76 s	1,87 s

Tab. x. Dauer für Berechnung guter Matches für Algorithmen bei Tag-/Nacht-Varianz.

Es zeigt sich, dass SIFT die stärkste Invarianz gegenüber Tag-/Nacht-Unterschieden aufweist, während ORB aufgrund seiner niedrigen Matching-Genauigkeit und KAZE aufgrund seiner langen Berechnungsdauer als ungeeignet erscheinen.

7.2.2. Okklusion

Für die Beurteilung der Okklusions-Performance wurden Aufnahmen von drei Gebäuden erstellt, die in unterschiedlichem Ausmaß von davor befindlichen Objekten verdeckt wurden. Diese Aufnahmen sind auf Abbildung x zu sehen.

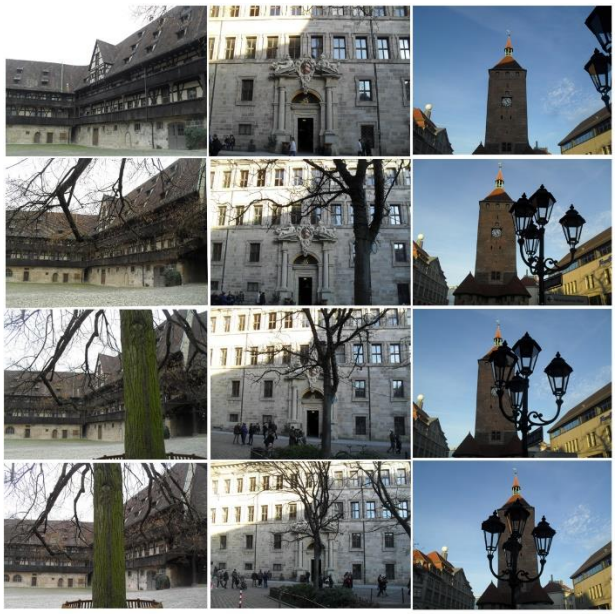

Abbildung x: Gebäude aus Bamberg und Nürnberg als Basis für die Invarianz-Tests hinsichtlich der Okklusion. Oben ist jeweils das unbedeckte Gebäude zu sehen.

Es wurde nun jeweils das Bild ohne Verdeckung (in Abbildung x ganz oben) mit den drei verdeckten Bildern des gleichen Gebäudes verglichen. Die Tabelle x gibt für jedes Gebäude jeweils die Anzahl Guter Matches für diese drei Vergleichsbilder an, wobei die Reihenfolge der Nummerierung in der Tabelle der vertikal absteigenden Reihenfolge in der Abbildung entspricht.

Generell ist zu beobachten, dass die Anzahl Guter Matches bei den stärker verdeckten Objekten geringer ausfällt. Bei den Bildern der Alten Hofhaltung in Bamberg und des Weißen Turms in Nürnberg fällt jedoch auf, dass die Positionierung des verdeckenden Objekts in der Bildmitte zu einer höheren Zahl Guter Matches führt als eine weniger zentrale Position.

Es zeigt sich ebenfalls, dass alle Algorithmen mit Ausnahme von ORB und BRISK stets Werte aufweisen, die höher liegen als die Maximalwerte bei abweichenden Motiven. Selbst diese beiden Algorithmen liefern jedoch Werte, die oberhalb des obersten Dezilswerts bei abweichenden Motiven liegen.

TABELLE X. ANZAHL GUTER MATCHES – OKKLUSION

Name	Matches Alte Hofhaltung			Matches Altes Rathaus			Matches Weißer Turm		
	#1	#2	#3	#1	#2	#3	#1	#2	#3
SIFT	2000	1500	1554	3420	2346	1559	1390	578	1064
SURF	3692	2527	2841	8711	6489	4806	2753	1610	1918
BRISK	1482	896	1050	3652	1557	790	1543	445	961
ORB	24	14	15	42	44	21	61	54	114
KAZE	1010	642	724	3166	1804	1335	1051	776	905
AKAZE	785	472	543	2410	1450	1101	832	542	666

Tab. x. Anzahl guter Matches für Algorithmen bei Okklusions-Varianz.

Die Dauer für die Berechnung der Guten Matches kann aus Tabelle x entnommen werden. ORB ist dabei deutlich schneller als andere Algorithmen während KAZE in zwei Fällen die längste Zeit beansprucht und BRISK in einem. Es fällt auf, dass die Berechnungsdauer bei BRISK erheblich variiert.

TABELLE X. GESAMTDAUER FÜR BERECHNUNG GUTER MATCHES – OKKLUSION

Name	Gesamtdauer Alte Hofhaltung	Gesamtdauer Altes Rathaus	Gesamtdauer Weißer Turm
SIFT	12,31 s	9,34 s	5,53 s
SURF	15,95 s	14,13 s	5,20 s
BRISK	59,09 s	16,21 s	1,05 s
ORB	0,45 s	0,38 s	0,27 s
KAZE	20,73 s	21,17 s	17,99 s
AKAZE	5,94 s	7,00 s	3,42 s

Tab. x. Gesamtdauer für Berechnung aller guter Matches für Algorithmen bei Okklusions-Varianz.

Alle Algorithmen erweisen sich im Bereich der Okklusions-Invarianz als hinreichend robust, wobei BRISK und ORB etwas schlechtere Werte liefern. Bezüglich der Geschwindigkeit erweisen sich KAZE und BRISK als am wenigsten geeignet.

7.2.3. Perspektive - Horizontal

Für den folgenden Performance-Test wurde bei der Aufnahme der Gebäude die Perspektive fortlaufend verändert, indem die Aufnahmeposition um das Objekt als Mittelpunkt rotiert wurde. Wie in Abbildung X zu erkennen, erzeugt der Sonnenstand dabei auch stark abweichende Schattenwürfe, die den Bildvergleich ebenfalls beeinflussen können. Für jedes Gebäude wird das zentrale Bild (in

Abbildung x mittig und rot markiert) mit den anderen Bildern des gleichen Objekts verglichen. In den folgenden Tabellen werden diese Vergleichsbilder jeweils von links nach rechts mit den Nummern 1-8 identifiziert.

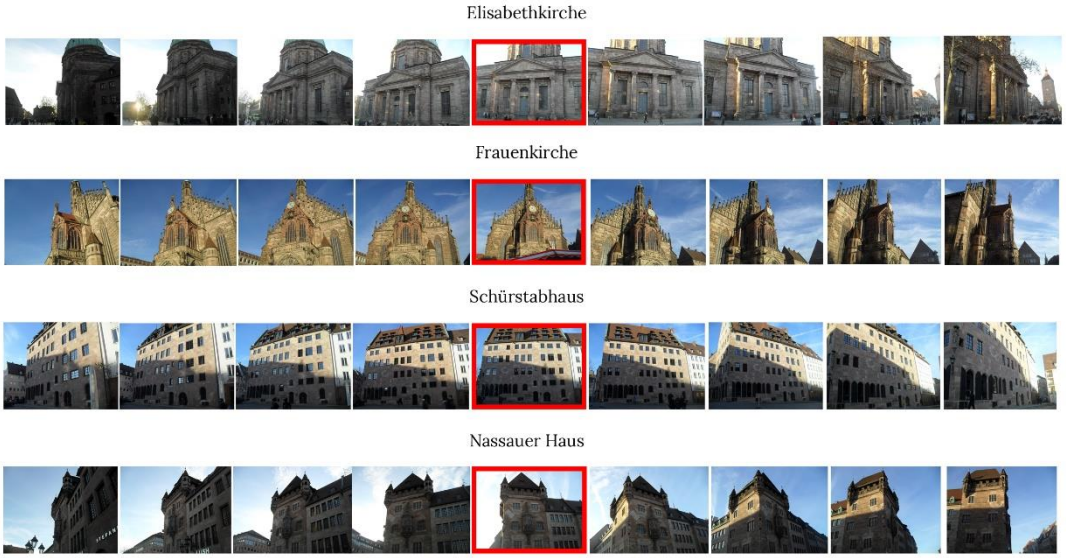

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der horizontalen Perspektive.

Generell lässt sich, wie den Tabellen X-X zu entnehmen ist, eine deutliche Abnahme Guter Matches beobachten, je stärker der Aufnahmewinkel sich von der Frontalansicht unterscheidet. Nur bei SIFT liegen die niedrigsten Werte über den Maximalwerten bei abweichenden Motiven, während SURF diese lediglich ein Mal unterschreitet. KAZE und AKAZE gelingt es, stets oberhalb des obersten Dezilswerts bei abweichenden Motiven zu bleiben, während ORB und BRISK auch daran scheitern. Bezüglich der Geschwindigkeit liegt KAZE meist an letzter Stelle, wobei BRISK auch hier in einem Fall die längste Berechnungsdauer aufweist.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) - ELISABETHKIRCHE

Name	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**Gesamtdauer**
SIFT	687	616	838	2088	1750	1153	619	653	19,23 s
SURF	1473	1553	1825	4046	3214	2063	1541	1492	27,31 s
BRISK	285	247	439	1458	1073	640	219	165	17,27 s
ORB	12	18	14	29	29	10	20	32	0,73 s
KAZE	209	198	439	1089	838	531	286	201	41,71 s
AKAZE	135	123	251	800	661	332	119	107	9,77 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Horizontal) für Elisabethkirche.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – FRAUENKIRCHE

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	692	955	1465	2920	2945	1442	936	738	22,38 s
SURF	1184	1234	1861	3690	3044	1651	1318	1218	25,94 s
BRISK	336	518	977	2468	2310	846	501	370	69,16 s
ORB	9	11	10	40	32	13	4	8	0,84 s
KAZE	314	470	1069	2527	2751	939	536	283	49,04 s
AKAZE	241	292	703	1855	1603	526	308	233	20,94 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Horizontal) für Frauenkirche.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – SCHÜRSTABHAUS

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	830	1072	1796	3303	2309	1249	887	766	17,42 s
SURF	1614	2084	3553	7222	5196	2789	1843	1426	25,43 s
BRISK	465	686	1604	3982	2273	887	330	311	28,56 s
ORB	50	60	81	170	78	25	15	8	0,71 s
KAZE	578	778	1531	3527	2344	829	470	467	44,07 s
AKAZE	351	506	957	2829	1582	445	279	235	12,03 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Horizontal) für Schürstabhaus.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – NASSAUER HAUS

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	289	281	264	390	508	265	269	256	12,57 s
SURF	614	548	788	1141	1471	653	571	551	14,29 s
BRISK	62	99	151	171	241	70	41	56	3,23 s
ORB	6	15	34	26	28	13	12	18	0,63 s
KAZE	270	298	378	410	600	342	264	251	40,96 s
AKAZE	107	120	175	201	358	138	128	79	8,21 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Horizontal) für Nassauer Haus.

SIFT und SURF gelingt es demnach, den Bildvergleich mit der größten Sicherheit durchzuführen. Bei ORB und BRISK legen die ermittelten Werte hingegen nahe, dass diese nicht für eine sichere Identifikation geeignet sind. Aufgrund seiner Geschwindigkeit trifft dies ebenfalls auf KAZE zu.

7.2.4. Perspektive - Vertikal

Auch vertikale Perspektivänderungen verdienen eine genauere Betrachtung. Hierfür wurden drei Objekte aus unterschiedlicher Entfernungen aufgenommen, wodurch sich auch eine Veränderung des vertikalen Blickwinkels ergab.

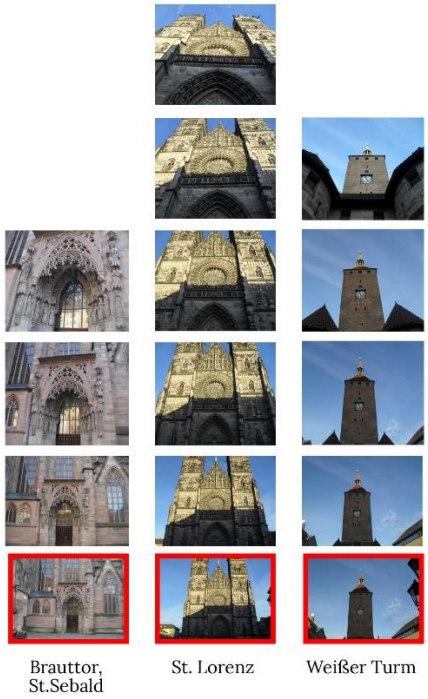

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der vertikalen Perspektive.

Abermals liegen alle Werte, welche Tabelle X zu entnehmen sind, von SIFT über den Maximalwerten bei abweichenden Motiven, während alle sonstigen Algorithmen mit Ausnahme von ORB über dem obersten Dezilswert liegen. Die längste Berechnungsdauer findet sich bei KAZE und BRISK.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL) – BRAUTTOR

Name	#1	#2	#3	Gesamtdauer
SIFT	4202	1759	828	9,07 s
SURF	6417	2499	1359	13,441 s
BRISK	2665	960	293	8,75 s
ORB	110	27	15	0,36 s
KAZE	2709	929	377	19,38 s
AKAZE	2331	850	227	5,26 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Vertikal) des Brauttors der Sebalduskirche.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL) – LORENZKIRCHE

Name	#1	#2	#3	#4	#5	Gesamtdauer
SIFT	8728	4210	1627	765	664	16,99 s
SURF	8905	4586	2044	1151	957	21,66 s
BRISK	9844	4180	1193	360	246	82,93 s
ORB	133	82	18	9	12	0,65 s
KAZE	9316	5312	1854	524	365	35, 93 s
AKAZE	5374	3029	606	281	270	19,04 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Vertikal) der Lorenzkirche.

TABELLE X. PERFORMANCE FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL) – WEISSER TURM

Name	#1	#2	#3	#4	Gesamtdauer
SIFT	1600	1196	926	443	8,35 s
SURF	1964	1135	712	574	6,73 s
BRISK	1167	575	347	115	2,76 s
ORB	137	76	41	34	0,35 s
KAZE	685	321	188	149	21,47 s
AKAZE	512	273	176	80	4,20 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Perspektivische Varianz (Vertikal) des Weißen Turms.

Ähnlich wie in den bisherigen Kapiteln erweist sich auch hier SIFT als der Algorithmus mit der höchsten Treffergenauigkeit, während für ORB das Gegenteil gilt. Aufgrund der Berechnungsdauer können BRISK und KAZE auch hier als ungeeignet eingestuft werden.

7.2.5. Rotation

Die Variierung der Bildrichtung kann sowohl manuell beim Tätigen der Aufnahme als auch nachträglich unter Verwendung von Bildbearbeitungs-Tools vorgenommen werden. Um die tatsächliche Verwendung der Applikation möglichst realistisch zu simulieren, wurde der erste der beiden Ansätze gewählt, auch wenn die Drehung dabei nicht mit dem gleichen Ausmaß an Exaktheit vorgenommen werden konnte. Neben dem nicht rotierten Originalbild wurden noch acht weitere Aufnahmen erstellt, wobei vor jeder Aufnahme eine Rotation um etwa 45 Grad vorgenommen wurde (siehe Abbildung X). Der Vergleich erfolgte jeweils zwischen dem Originalbild und einem der acht Rotationsbilder.

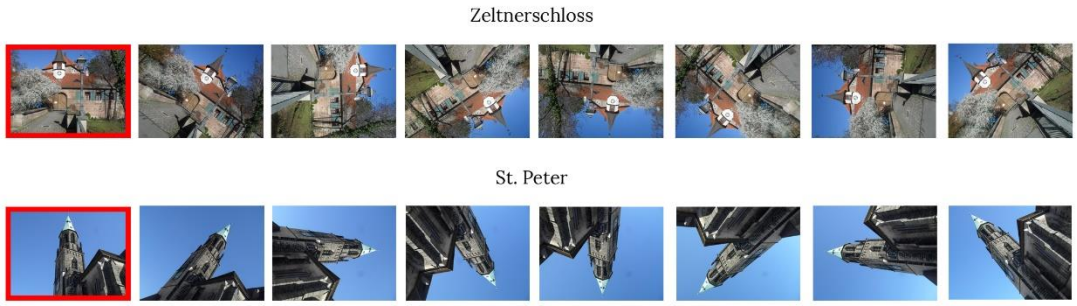

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der Rotation.

Wie Tabelle X zu entnehmen, liegen in diesem Fall alle Algorithmen über den Maximalwerten für abweichende Motive. Hinsichtlich der Berechnungsdauer liegt BRISK bei einem Objekt weit oberhalb der anderen Algorithmen, während KAZE beim anderen Objekt höher liegt.

TABELLE X. PERFORMANCE FÜR ROTATIONSVARIANZ – ZELTNERSCHLOSS

Name	**45°**	**90°**	**135°**	**180°**	**225°**	**270°**	**315°**	**Gesamtdauer**
SIFT	15845	12410	13057	17507	18246	16219	19766	43,90 s
SURF	6722	14321	5927	14657	6927	15807	7199	38,27 s
BRISK	27464	21524	24482	31172	31067	27891	33382	435,51 s
ORB	218	137	171	180	177	239	228	1,13 s
KAZE	15529	13206	14883	17742	17112	15004	17941	50,57 s
AKAZE	12512	10765	11816	14209	13880	12375	14924	39,92 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Rotationsvarianz für Zeltnerschloss, Nürnberg.

TABELLE X. PERFORMANCE FÜR ROTATIONSVARIANZ – ST. PETER

Name	**45°**	**90°**	**135°**	**180°**	**225°**	**270°**	**315°**	**Gesamtdauer**
SIFT	4077	4657	3467	4979	4824	4532	4773	13,45 s
SURF	3349	9012	3704	10407	3742	8560	3565	14,62 s
BRISK	6637	7583	5899	8011	7627	7627	7741	13,53 s
ORB	349	329	316	350	311	334	286	0,61 s
KAZE	6395	8305	7222	8962	7306	7898	7490	39,10 s
AKAZE	5064	6400	5855	6898	5978	6216	6088	9,95 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Rotationsvarianz für St. Peter, Nürnberg.

BRISK und KAZE müssen hier aufgrund der Berechnungsdauer als ungeeignet eingestuft werden. Eine hinreichende Rotations-Invarianz ist jedoch für alle Algorithmen gegeben.

7.2.6. Skalierung

Die Skalierung kann bei Aufnahmen von immobilen Objekten auf zweierlei Art variiert werden: Durch das Erstellen von Bildern in unterschiedlicher Entfernung sowie über die Abwandelung des Zoomfaktors der Kamera. Im Zuge der Evaluierung wurden beide Herangehensweisen verwendet. Abbildung x zeigt die verwendeten Aufnahmen, wobei beim Objekt Zeltnerschloss die Erstere zum Einsatz kam, die Letztere hingegen beim Bauteil von St. Peter. In beiden Fällen wurde jeweils die Aufnahme mit der größten Nähe zum Objekt bzw. dem größten Zoomfaktor mit den anderen Aufnahmen verglichen.

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der Skalierung.

SIFT und SURF liegen hier stets über den Maximalwerten für abweichende Motive. KAZE und AKAZE übersteigen stets den obersten Dezilwert, was BRISK und ORB jedoch nicht gelingt. Bei BRISK zeigt sich wieder eine deutlich unterschiedliche Berechnungsdauer.

TABELLE X. PERFORMANCE FÜR SKALIERUNGSVARIANZ – ZELTNERSCHLOSS

Name	#1	#2	#3	#4	Gesamtdauer
SIFT	1913	1327	1013	1056	20,56 s
SURF	3309	2404	1989	1865	19,88 s
BRISK	1317	885	600	511	229,61 s

ORB	11	11	13	9	0,64 s
KAZE	1261	746	653	550	29,99 s
AKAZE	1043	708	510	435	18,93 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Skalierungsvarianz für Zeltnerschloss, Nürnberg.

TABELLE X. PERFORMANCE FÜR SKALIERUNGSVARIANZ – ST. PETER

Name	**#1**	**#2**	**#3**	**#4**	**Gesamtdauer**
SIFT	1453	916	375	192	7,27 s
SURF	4152	2667	1224	974	10,24 s
BRISK	609	361	95	34	0,90 s
ORB	79	8	15	11	0,34 s
KAZE	806	350	142	64	21,94 s
AKAZE	835	401	129	62	4,35 s

Tab. x. Performance (Anzahl guter Matches und Gesamtberechnungsdauer) der Algorithmen für Skalierungsvarianz für St. Peter, Nürnberg.

Nur SIFT und SURF können hier bezüglich der Treffergenauigkeit als geeignet eingestuft werden, während die Geschwindigkeit von BRISK und KAZE nicht als ausreichend gewertet werden kann.

7.3. Fazit

Aus den vorliegenden Kapiteln lassen sich nun Werte für die zu erwartende Zahl Guter Matches unter den vorgegebenen Bedingungen nennen. Tabelle X stellt dabei die Maximalwerte (einschließlich der Ausreißer) sowie die obersten Dezilwerte für Bilder mit abweichenden Motiven den Minimalwerten bei identischen Motiven gegenüber. So lässt sich eine Aussage darüber treffen, ob die Algorithmen eine zuverlässige Differenzierung der Ergebnisse anhand der ermittelten Zahl Guter Matches ermöglichen.

TABELLE X. ZUSAMMENFASSUNG – GUTE MATCHES BEI UNTERSCHIEDLICHEN UND GLEICHEN MOTIVEN

Name	**Gute Matches - Maximalwert Abweichende Motive**	**Gute Matches - Oberster Dezilwert Abweichende Motive**	**Gute Matches - Minimalwert Identische Motive**
SIFT	109	62	192

SURF	595	104	551
BRISK	1392	43	34
ORB	48	12	4
KAZE	543	41	64
AKAZE	327	30	62

Tab. x. Maximalwerte sowie der obere Dezilwert für unterschiedliche Motive sowie der Minimalwert für identische Motive. Angegeben ist jeweils die Anzahl guter Matches.

Aus diesen Ergebnissen lässt sich ableiten, dass SIFT generell die höchste Erkennungsgenauigkeit liefert. Ist ein Objekt auf den Vergleichsbildern enthalten, so ist die Anzahl Guter Matches bei der Verwendung von SIFT stets deutlich höher als beim Vergleich von Bildern mit abweichenden Motiven. Auch die Ergebnisse von SURF sind weitestgehend zufriedenstellend, auch wenn diese für manche Testfälle nicht die optimale Sicherheit bieten konnten, was insbesondere an SURFs stärkerer Anfälligkeit für Ausreißer liegt. Bei den weiteren Algorithmen wurde eine deutlich schlechtere Identifikationssicherheit festgestellt, wobei besonders ORB negativ auffällt.

Hinsichtlich der Geschwindigkeit erweisen sich KAZE und BRISK als die langsamsten Algorithmen, wobei bei BRISK je nach Bild sehr starke Abweichungen auftreten. In der Regel ist bei ORB die höchste Geschwindigkeit zu beobachten, was jedoch dessen geringe Treffergenauigkeit nicht auszugleichen vermag.

Bisher wurde nur das Verhalten beim Vorliegen einzelner Varianzen überprüft. Durch deren Kombination – etwa durch eine nächtliche Aufnahme mit starker perspektivischer Verzerrung - könnte es ggfs. möglich sein, dass auch die Anzahl Guter Matches bei SIFT so niedrig ausfällt, dass man vom Vorliegen unterschiedlicher Motive ausgehen muss. Dessen ungeachtet hat sich SIFT als der geeignetste Algorithmus erwiesen.

8. Diskussion

Aufgrund der beobachteten Ergebnisse muss SIFT als der Merkmalserkennungs-Algorithmus mit der höchsten Eignung für die mobile Bildklassifikation historischer Gebäude und Bauteile angesehen werden. Er überzeugt nicht nur hinsichtlich der Sicherheit der Identifikation, sondern führt die Berechnungen auch schneller durch als mehrere der anderen Algorithmen. Zwar erweist sich insbesondere ORB als deutlich schnellere Alternative, doch wird dies mit einer unzufriedenstellenden Identifikationssicherheit erkauft. Die sonstigen untersuchten Algorithmen können weder bei der Identifikationssicherheit noch bei der Geschwindigkeit überzeugen.

Auch wenn das Ergebnis für den gegebenen Anwendungsfall für die Verwendung von SIFT spricht, ist es vorstellbar, dass für andere Anwendungsfälle – ggfs. auch im Bereich der Bildklassifikation historischer Gebäude – ein anderer Algorithmus zu bevorzugen ist. Dies könnte etwa der Fall sein, wenn eine besonders hohe Zahl von Vergleichsbildern in möglichst kurzer Zeit mit der aufgenommenen Fotografie verglichen werden müssen.

Für die Weiterentwicklung der Applikation bietet sich schließlich noch eine Reihe von Möglichkeiten an. So kann die Qualität der Identifikation beispielsweise durch den Einsatz von Machine Learning verbessert werden. Die Geschwindigkeit der Berechnung kann hingegen deutlich verringert werden, falls man die Deskriptoren der Bilder bereits vorberechnet und diese in der Datenbank abspeichert. So müssen beim Matching lediglich die Deskriptoren der Fotoaufnahme neu berechnet werden, was eine deutliche Zeitersparnis verspricht.

Für BenutzerInnen mit einem hohen Interesse an Bildklassifikations-Themen könnte man außerdem die Möglichkeit bieten, den Merkmalserkennungs-Algorithmus selbst auszuwählen und so deren Erkennungsqualität je nach Motiv zu untersuchen und miteinander zu vergleichen.

9. Literaturverzeichnis

Alcantarilla, P.F., Bartoli, A. und Davison, A.J. "KAZE features." Computer Vision – ECCV 12. Springer-Verlag, Berlin, Deutschland, 2012. S. 214-227.

Alcantarilla, P.F., Nuevo, J. und Bartoli, A.J. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces." IEEE Transaction on Pattern Analysis and Machine Intelligence, 34.7 (2011). S. 1281-1298.

Andersson, O. und Marquez, S.R. „A comparison of object detection algorithms using unmanipulated testing images: Comparing SIFT, KAZE, AKAZE and ORB." 2016. https://pdfs.semanticscholar.org/f054/dfbfc8208304b298b849a8befec3f348dc9b.pdf (Letzter Zugriff: 31.03.2020)

Bay, H., Tuytelaars, T. und Van Gool, L. „Surf: Speeded up robust features." Computer Vision – ECCV 2006. Springer-Verlag, Berlin, Deutschland, 2006. S. 404-417.

Calonder, M. et al. „BRIEF: Binary robust independent elementary features." Computer Vision – ECCV 2010. Springer-Verlag, Berlin, Deutschland, 2010. S. 778-792.

Dawson-Howe, K. „A practical introduction to computer vision with OpenCV." John Wiley & Sons, Hoboken, Vereinigte Staaten, 2014.

„Feature Matching" (n.d.) https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html (Letzter Zugriff: 31.03.2020)

Gollapudi, S. „Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs." New York City, Vereinigte Staaten, 2019.

Groeneweg, N.J.C. et al. „A Fast Offline Building Recognition Application on a Mobile Telephone." International Conference on Advanced Concepts for Intelligent Vision Systems. Springer-Verlag, Berlin, Deutschland, 2006. S. 1122-1132.

Leutenegger, S., Chli, M. und Siegwart, R.Y. „BRISK: Binary robust invariant scalable keypoints." Proceedings of the IEEE International Conference on Computer Vision, 2011. S. 2548-2555.

Li, J. et al. „Building Recognition in Urban Environments: A Survey of State-of-the-Art and Future Challenges." Information Sciences, 277 (2014). S. 406-420.

Lowe, D.G. „Distinctive image features from scale-invariant keypoints." International Journal of Computer Vision, 60.2 (2004). S. 91-110.

Minichino, J. und Howse, J. „Learning OpenCV 3 Computer Vision with Python." Packt Publishing, Birmingham, Vereinigtes Königreich, 2015.

Philbin, J., Arandjelović, R. und Zisserman, A. „The Oxford Buildings Dataset". (n.d.) https://www.robots.ox.ac.uk/~vgg/data/oxbuildings/ (Letzter Zugriff: 31.03.2020).

Rosten, E., Porter, R. und Drummond, T. "Faster and better: A machine learning approach to corner detection." IEEE Transactions on Pattern Analysis and Machine Intelligence, 32.1 (2008). S. 105-119.

Rublee, E. et al. „ORB: An efficient alternative to SIFT or SURF." 2011 International conference on computer vision, Barcelona, Spanien, 2011. S. 2564-2571.

Scherer, R. „Computer vision methods for fast image classification and retrieval." Springer International Publishing, Basel, Schweiz, 2020.

Tareen, S.A.K., und Saleem, Z. „A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK.“ 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). IEEE, 2018. S. 1-10.

Zhang, Y., Yu, F., Wang, Y. und Wang, K. „Performance Evaluation of Feature Detection Methods for Visual Measurements.“ Engineering Letters, 27.2 (2019). http://www.engineeringletters.com/issues_v27/issue_2/EL_27_2_08.pdf (Letzter Zugriff: 31.03.2020)

A. Eidesstattliche Erklärung

Ich erkläre hiermit gemäß §17 Abs. 2 APO, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

(Datum)	(Unterschrift)