

Thema: **Identifikation von historischen
Gebäuden und Bauteilen durch
Bildklassifikation**

Bachelorarbeit

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

Verfasser: Christof Wittmann

Prüfer: Prof. Dr. Christoph Schlieder

Inhaltsverzeichnis

1. Einleitung.....	1
2. Problemstellung.....	2
3. Forschungsstand: Methoden der Bilderkennung.....	5
3.1. Grundprinzipien der Merkmalerkennung (Feature Detection)....	5
3.2. Methoden der Merkmalerkennung	6
3.3. Scale-Invariant Feature Transform (SIFT).....	7
3.3.1. Scale-Space Extrema Detection	7
3.3.2. Keypoint Localization	9
3.3.3. Orientation Assignment	9
3.3.4. Keypoint Description.....	11
3.4. Weitere Algorithmen zur Merkmalerkennung	12
3.4.1. Speeded Up Robust Features (SURF)	12
3.4.2. Binary Robust Invariant Scalable Keypoints (BRISK).....	13
3.4.3. Oriented FAST and Rotated BRIEF (ORB).....	15
3.4.4. KAZE Features und Accelerated KAZE (AKAZE)	16
3.5. Matching	17
3.6. Performancevergleiche der Algorithmen	18
3.6.1. Gebäudeklassifikation	19
3.6.2. Invarianz-Tests	20
4. Forschungsstand: Technisch	22
4.1. OpenCV	22
5. Lösungsansatz.....	24
6. Umsetzung.....	26
6.1. Architektur.....	26
6.2. Client.....	27
6.3. Server.....	28
7. Evaluierung.....	30
7.1. Matches bei abweichenden Motiven	30
7.2. Matches bei Varianz der Aufnahmebedingungen.....	33
7.2.1. Tag und Nacht	33
7.2.2. Okklusion	35
7.2.3. Perspektive - Horizontal	37

7.2.4. Perspektive - Vertikal.....	39
7.2.5. Rotation.....	41
7.2.6. Skalierung.....	43
7.3. Fazit.....	45
8. Diskussion.....	46
9. Literaturverzeichnis	47

ABBILDUNGSVERZEICHNIS

ABBILDUNG 1: BILDERZEUGUNG IM RAHMEN DER SCALE-SPACE-EXTREMA-DETECTION.	8
ABBILDUNG 2: DIFFERENZBILDER ALS ERGEBNIS DER DIFFERENCE-OF-GAUSSIAN-BERECHNUNG.	8
ABBILDUNG 3: PIXEL-NACHBARSCHAFTSVERGLEICHE BEI DER SCALE-SPACE-EXTREMA-DETECTION (LOWE, 2004).....	9
ABBILDUNG 4: HELLIGKEITSVERLÄUFE IN PIXEL-NACHBARSCHAFT EINES KEYPOINTS.	10
ABBILDUNG 5: ORIENTIERUNGSHISTOGRAMM EINES KEYPOINTS.....	10
ABBILDUNG 6: GENERIERUNG EINES 128-DIMENSIONALEN VEKTORS FÜR EINEN KEYPOINT.....	12
ABBILDUNG 7: GAUßSCHE ABLEITUNGEN (LINKS) UND DEREN APPROXIMATION DURCH BOXFILTER (RECHTS) (BAY, TUYTELAARS & VAN GOOL, 2006).	13
ABBILDUNG 8: VON SURF VERWENDETE HAAR-WAVELETS (LINKS) UND QUADRATISCHE REGIONEN UM KEYPOINTS (BAY, TUYTELAARS & VAN GOOL, 2006).	13
ABBILDUNG 9: IDENTIFIZIERUNG VON KEYPOINTS DURCH VERGLEICH MIT KREISPUNKTEN (ROSTEN, PORTER & DRUMMOND, 2008).	14
ABBILDUNG 10: VON SURF VERWENDETE HAAR-WAVELETS (LINKS) UND QUADRATISCHE REGIONEN UM KEYPOINTS (LEUTENEGGER, CHLI & SIEGWART, 2011).	15
ABBILDUNG 11: ERZEUGUNG DES SCALE SPACE MIT LINEARER DIFFUSION (OBEN) UND NON-LINEARER DIFFUSION (UNTEN) (ALCANTARILLA, BARTOLI & DAVISON, 2012).	16
ABBILDUNG 12: ARCHITEKTONISCHE BILDMOTIVE ALS BASIS FÜR PERFORMANCEMESSUNG DER MERKMALSERKENNUNGS- ALGORITHMEN. OBEN: BILDDPAAR 1 (BUILDING DATASET), UNTEN: BILDDPAAR 2 (ROOFS DATASET) (TAREEN & SALEEM, 2018).	19
ABBILDUNG 13: ARCHITEKTUR VON BIDENT BUILDING IDENTIFICATION.	26
ABBILDUNG 14: SCREENSHOTS DER APPLIKATION BIDENT BUILDING IDENTIFICATION: HAUPTSEITE (LINKS) UND DETAILSEITE (RECHTS).....	27
ABBILDUNG 15: SPALTEN DER GEBÄUDE-TABELLE BUILDING IN MYSQL-DATENBANK.	28
ABBILDUNG 16: BILDER VON GEBÄUDEN AUS NÜRNBERG FÜR DEN VERGLEICH MIT DEN AUFNAHMEN DES OXFORD BUILDINGS DATASET.	31
ABBILDUNG 17: AUSREIßER BEIM BILDVERGLEICH DES ALTEN RATHAUSES IN BAMBERG MIT AUFNAHMEN DES OXFORD BUILDINGS DATASET (PHILBIN, J., ARANDJELOVIĆ, R. UND ZISSERMAN, (N.D.)).	31
ABBILDUNG 18: BOXPLOTS MIT ANZAHL DER GUTEN MATCHES FÜR SÄMTLICHE ALGORITHMEN BEIM VERGLEICH DES ALTEN RATHAUSES MIT DEN BILDERN DES OXFORD BUILDINGS DATASETS.....	32
ABBILDUNG 19: GEBÄUDE UND BAUTEILE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH TAG- /NACHT-UNTERSCHIEDEN.	34
ABBILDUNG 20: GEBÄUDE AUS BAMBERG UND NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER OKKLUSION. OBEN IST JEWEILS DAS UNBEDECKTE GEBÄUDE ZU SEHEN.	35
ABBILDUNG 21: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER HORIZONTALEN PERSPEKTIVE.....	37
ABBILDUNG 22: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER VERTIKALEN PERSPEKTIVE.....	40
ABBILDUNG 23: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER ROTATION.	42
ABBILDUNG 24: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER SKALIERUNG.	43

TABELLENVERZEICHNIS

TABELLE 1: ÜBERSICHT ÜBER POPULÄRE ALGORITHMEN ZUR MERKMALSERKENNUNG (IN ANLEHNUNG AN ANDERSSON & MARQUEZ, 2016, TAREEN & SALEEM, 2018, ZHANG ET AL., 2019).....	4
TABELLE 2: ANZAHL MATCHES UND BERECHNUNGSZEIT BEI GEBÄUDE-BILDERN FÜR UNTERSCHIEDLICHE ALGORITHMEN (IN ANLEHNUNG AN TAREEN & SALEEM, 2018).....	20
TABELLE 3: ANTEIL KORREKTER MATCHES DER ALGORITHMEN FÜR VERSCHIEDENE INVARIANZTYPEN (IN ANLEHNUNG AN ANDERSSON & MARQUEZ, 2016).	20
TABELLE 4: ROBUSTHEIT DER ALGORITHMEN SIFT, SURF UND FAST BEZÜGLICH ROTATION, SKALIERUNG, WEICHZEICHNUNG, KOMPRESSION UND BELEUCHTUNG (ZHANG ET AL, 2019).....	21
TABELLE 5: KATEGORIEN FÜR DIE SICHERHEIT DER BILDKLASSIFIKATION MIT DEM KAZE-ALGORITHMUS (EIGENE DARSTELLUNG).	29
TABELLE 6: STATISTISCHE VERTEILUNG GUTER MATCHES UND GESCHWINDIGKEIT BEIM VERGLEICH MIT ALLEN BILDERN DES OXFORD BUILDINGS DATASETS (EIGENE DARSTELLUNG).	32
TABELLE 7: ANZAHL GUTER MATCHES FÜR ALGORITHMEN BEI TAG-/NACHT-VARIANZ (EIGENE DARSTELLUNG).	34
TABELLE 8: DAUER FÜR BERECHNUNG GUTER MATCHES FÜR ALGORITHMEN BEI TAG-/NACHT-VARIANZ (EIGENE DARSTELLUNG).	34
TABELLE 9: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ - TAG/NACHT (EIGENE DARSTELLUNG).....	35
TABELLE 10: ANZAHL GUTER MATCHES FÜR ALGORITHMEN BEI OKKLUSIONS-VARIANZ (EIGENE DARSTELLUNG).	36
TABELLE 11: GESAMTDAUER FÜR BERECHNUNG ALLER GUTER MATCHES FÜR ALGORITHMEN BEI OKKLUSIONS-VARIANZ (EIGENE DARSTELLUNG).....	36
TABELLE 12: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER OKKLUSION (EIGENE DARSTELLUNG)	37
TABELLE 13: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: ELISABETHKIRCHE, NÜRNBERG (EIGENE DARSTELLUNG).	38
TABELLE 14: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: FRAUENKIRCHE, NÜRNBERG (EIGENE DARSTELLUNG).	38
TABELLE 15: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: SCHÜRSTABHAUS, NÜRNBERG (EIGENE DARSTELLUNG).....	38
TABELLE 16: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: NASSAUER HAUS, NÜRNBERG (EIGENE DARSTELLUNG).	39
TABELLE 17: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER PERSPEKTIVÄNDERUNGEN (HORIZONTAL) (EIGENE DARSTELLUNG)	39
TABELLE 18: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL) – MOTIV: BRAUTOR, ST. SEBALD, NÜRNBERG (EIGENE DARSTELLUNG).....	40
TABELLE 19: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL) – MOTIV: ST. LORENZ, NÜRNBERG (EIGENE DARSTELLUNG).....	41
TABELLE 20: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL) – MOTIV: WEIßER TURM, NÜRNBERG (EIGENE DARSTELLUNG).....	41
TABELLE 21: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER PERSPEKTIVÄNDERUNGEN (VERTIKAL) (EIGENE DARSTELLUNG)	41
TABELLE 22: PERFORMANCE DER ALGORITHMEN FÜR ROTATIONSVARIANZ – MOTIV: ZELTNERSCLOSS, NÜRNBERG (EIGENE DARSTELLUNG).	42
TABELLE 23: PERFORMANCE DER ALGORITHMEN FÜR ROTATIONSVARIANZ – MOTIV: ST. PETER, NÜRNBERG (EIGENE DARSTELLUNG).	42

TABELLE 24: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER ROTATION (EIGENE DARSTELLUNG)	43
TABELLE 25: PERFORMANCE DER ALGORITHMEN FÜR SKALIERUNGSVARIANZ – MOTIV: ZELTNERSCLOSS, NÜRNBERG (EIGENE DARSTELLUNG).....	44
TABELLE 26: PERFORMANCE DER ALGORITHMEN FÜR SKALIERUNGSVARIANZ – MOTIV: ST. PETER, NÜRNBERG (EIGENE DARSTELLUNG).	44
TABELLE 27: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER SKALIERUNG (EIGENE DARSTELLUNG)	44
TABELLE 28: VERGLEICH DER ALGORITHMUS-PUNKTBEWERTUNGEN FÜR VERSCHIEDENE INVARIANZTYPEN – ABSOLUTE ZAHLEN (EIGENE DARSTELLUNG).	45

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, insbesondere solcher mit geringer Bekanntheit, 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 (Merkmalerkennung). 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 in Bezug auf ihre Eignung für die Klassifikation historischer Gebäude und Bauteile verglichen werden. 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 deren Bauteilen erstellt werden können. Die Fotografie wird daraufhin an einen Server übermittelt, auf dem sie unter Nutzung eines bestimmten Merkmalerkennungs-Algorithmus mit Bildern aus einer Datenbank vorhandener Gebäude verglichen wird.

Die Zielgruppe besteht dabei aus TouristInnen, die sich mit Hilfe ihres Smartphones 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 Merkmalerkennungs-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.

Die Benutzung der Anwendung soll sofort erfolgen können, ohne dass die BenutzerInnen eine Installation auf ihren Geräten durchführen müssen. Die Bilddaten, mit denen das aufgenommene Foto verglichen wird, müssen zentral auf einem Server hinterlegt werden, um den Speicherbedarf auf den mobilen Geräten zu minimieren.

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 Vergleich 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 (Hutchings & Mayol-Cuevas, 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 3.5.) 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 eines 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 berücksichtigen, dass die korrekte Identifizierung möglichst unabhängig von den äußeren Umständen der Aufnahme sein sollte.

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 1 werden diese mitsamt ihres Veröffentlichungszeitpunkts und ihrer Autoren aufgelistet (Andersson & Marquez, 2016, Tareen & Saleem, 2018, Zhang et al., 2019).

Als weitere Entscheidungsgrundlage kann dabei die Tatsache dienen, dass es sich bei diesen sechs auch um diejenigen Feature Detection-Algorithmen

handelt, die von der populären Computer-Vision-Bibliothek OpenCV zur Verfügung gestellt werden.^{1 2}

TABELLE 1: ÜBERSICHT ÜBER POPULÄRE ALGORITHMEN ZUR MERKMALSERKENNUNG (IN ANLEHNUNG AN ANDERSSON & MARQUEZ, 2016, TAREEN & SALEEM, 2018, ZHANG ET AL., 2019).

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

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)

3. Forschungsstand: Methoden der Bildererkennung

3.1. Grundprinzipien der Merkmalerkennung (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 Merkmalerkennung 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 Grauwerts 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. 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 Merkmalerkennung handelt es sich um eine der beiden Hauptrichtungen der inhaltsbasierten 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

zwar die Zugehörigkeit zweier Motive zu einer Objektkategorie gut feststellbar ist, die Gleichheit der Motive jedoch nicht. Ein weiterer Nachteil des Maschinellen Lernens ist der Bedarf an umfangreichen Mengen von Trainingsdaten. Die Merkmalerkennung 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 Merkmalerkennung

Merkmalerkennungs-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 Merkmalerkennung 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 Merkmalerkennungs-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).

3.3. Scale-Invariant-Feature-Transform (SIFT)

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 Merkmalerkennungs-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 1 zeigt exemplarisch die dabei erzeugten Bilder.

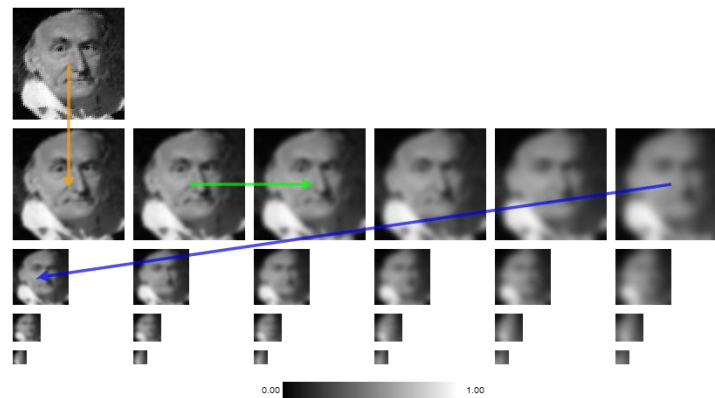


ABBILDUNG 1: BILDERZEUGUNG IM RAHMEN DER SCALE-SPACE-EXTREMA-DETECTION.³

Nun werden aus diesen Bildern mittels der Difference-of-Gaussian-Methode (DoG) Differenzbilder generiert. Hierfür werden jeweils zwei innerhalb einer Oktave nebeneinanderliegende Bilder als Ausgangsgrundlage verwendet. In Abbildung 2 sind die auf diese Weise erzeugten Differenzbilder zu sehen, wobei die Zahl der Bilder pro Oktave nun um eines verringert ist.

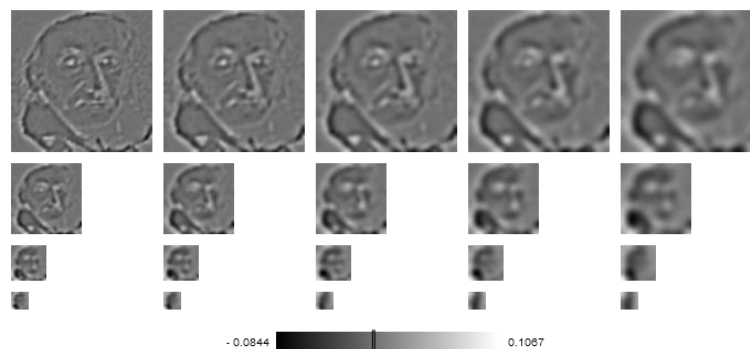


ABBILDUNG 2: DIFFERENZBILDER ALS ERGEBNIS DER DIFFERENCE-OF-GAUSSIAN-BERECHNUNG.⁴

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 3 zu sehen ist.

³ <http://weitz.de/sift> (Letzter Zugriff: 30.3.2020)

⁴ <http://weitz.de/sift> (Letzter Zugriff: 30.3.2020)

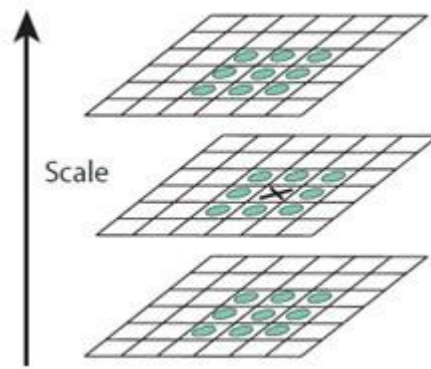


ABBILDUNG 3: PIXEL-NACHBARSCHAFTSVERGLEICHE BEI DER SCALE-SPACE-EXTREMA-DETECTION (LOWE, 2004).

Um als potenzieller Keypoint in Frage zu kommen, muss ein Pixel einen höheren bzw. niedrigeren Grauwert aufweisen als alle 26 Nachbapixel. 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 4 zu sehen sind, können sowohl Intensität als auch Richtung ermittelt werden.

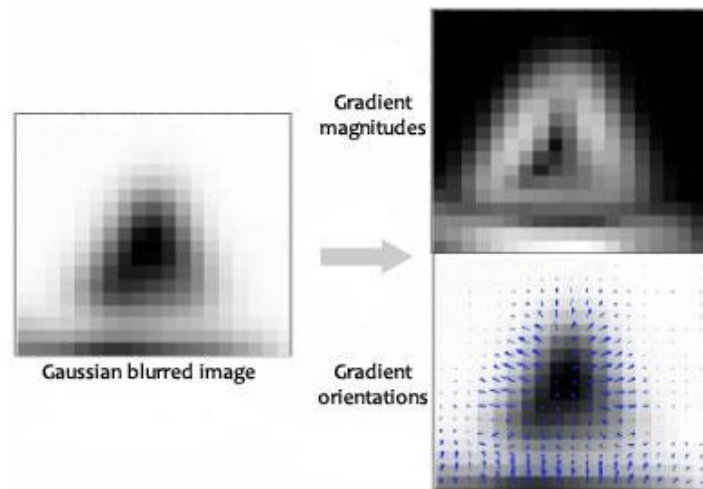


ABBILDUNG 4: HELLGKEITSVERLÄUFE IN PIXEL-NACHBARSCHAFT EINES KEYPOINTS.⁵

Es wird nun für jeden Keypoint ein Histogramm angelegt, in dem die Intensität des Verlaufs für jede Orientierung hinterlegt wird. Zuerst teilt man die 360°-Umgebung jedoch in 36 Behälter auf, die jeweils einem 10°-Abschnitt entsprechen. Ein Beispielhistogramm ist in Abbildung 5 zu sehen.

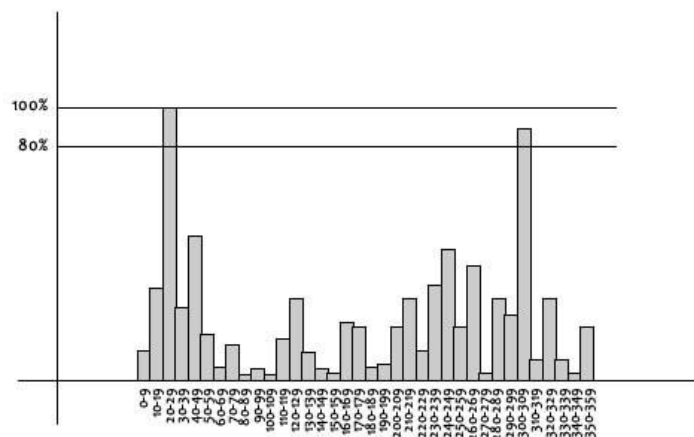


ABBILDUNG 5: ORIENTIERUNGSHISTOGRAMM EINES KEYPOINTS.⁶

In den meisten Fällen wird nun der Behälter mit dem höchsten Wert gewählt und dessen Orientierung als Orientierung des Keypoints festgelegt. Wie in Abbildung 5 zu sehen ist, können jedoch auch mehrere Orientierungen vorliegen, die eine ähnliche Intensität aufweisen. Deshalb vergleicht man die Intensität aller Behälter mit der des Behälters mit dem Maximalwert. Für Behälter, die

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)

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 6 ist auf der linken Seite die Nachbarschaft als Quadrat mit Seitenlänge 16 Pixeln zu sehen. Diese Umgebung wird nun in 16 Teilquadrate mit je 4×4 Pixeln aufgeteilt. Für jedes dieser Teilquadrate werden ähnlich wie im Schritt Orientation Assignment die Intensität der Helligkeitsverläufe und die 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 Menge von Vektoren verstanden werden. So sind denn jedem der 16 Teilbereiche 8 Vektoren zugeordnet, deren Beträge jeweils die Intensität des Helligkeitsverlaufs in diese Richtung angeben. Abbildung 6 zeigt auf der rechten Seite den so erzeugten 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).

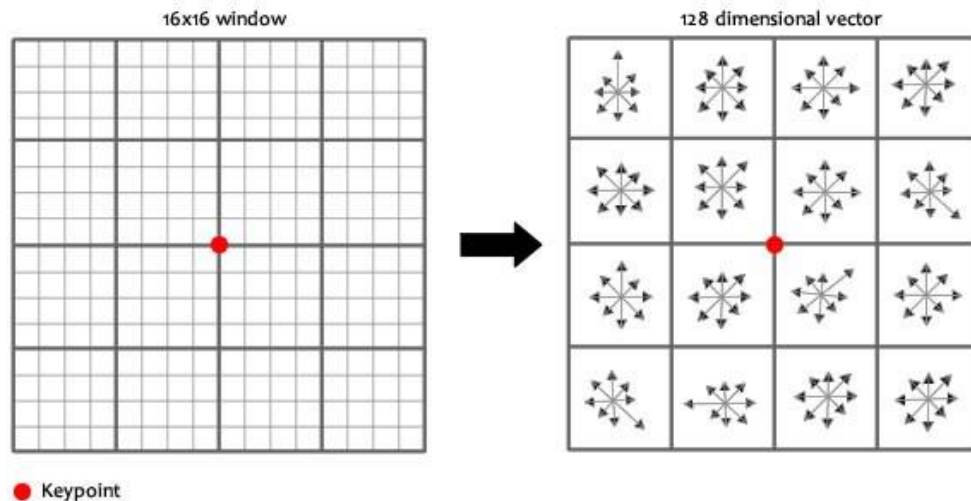


ABBILDUNG 6: GENERIERUNG EINES 128-DIMENSIONALEN VEKTORS FÜR EINEN KEYPOINT.⁷

3.4. Weitere Algorithmen zur Merkmalerkennung

Auch über 20 Jahre nach seiner ersten Veröffentlichung wird der SIFT-Algorithmus noch häufig zu Zwecken der Merkmalerkennung eingesetzt. In der Zwischenzeit haben sich jedoch zahlreiche weitere Algorithmen zu diesem hinzugesellt, deren Schöpfer den Anspruch hatten, SIFT hinsichtlich Erkennungsgenauigkeit und/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 7 zu betrachten.

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

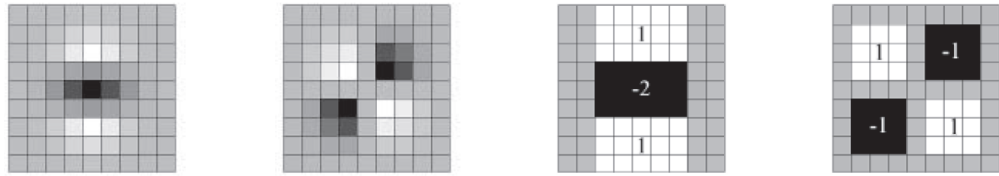


ABBILDUNG 7: GAUßSCHE ABLEITUNGEN (LINKS) UND DEREN APPROXIMATION DURCH BOXFILTER (RECHTS) (BAY, TUYTELAARS & VAN GOOL, 2006).

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 8). 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 8: 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 bauten mit ihrem 2011 veröffentlichten 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 9). 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).

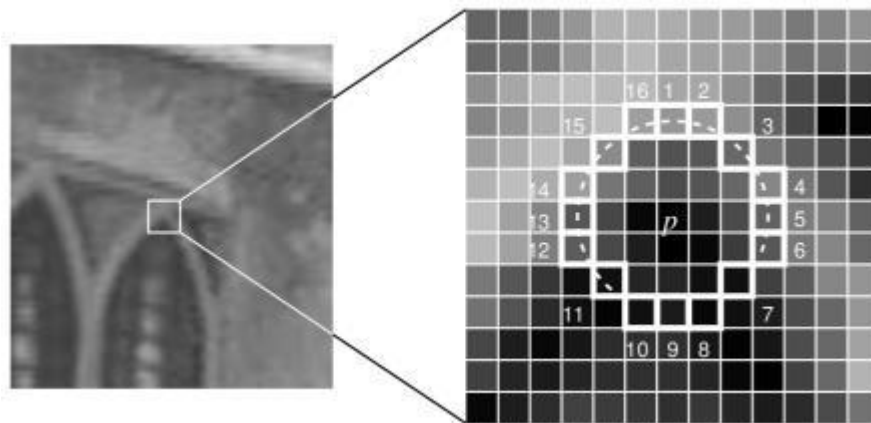


ABBILDUNG 9: IDENTIFIZIERUNG VON KEYPOINTS DURCH VERGLEICH MIT KREISPUNKTEN (ROSTEN, PORTER & DRUMMOND, 2008).

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. 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, wie es etwa bei SIFT und SURF erfolgt, wird dieser binäre Deskriptor 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 10 zeigt exemplarisch die Anordnung der Punkte im Rahmen von BRISK. Beim Vergleich wird nun untersucht, ob entweder der Keypoint oder der Vergleichspunkt einen höheren Grauwert 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).

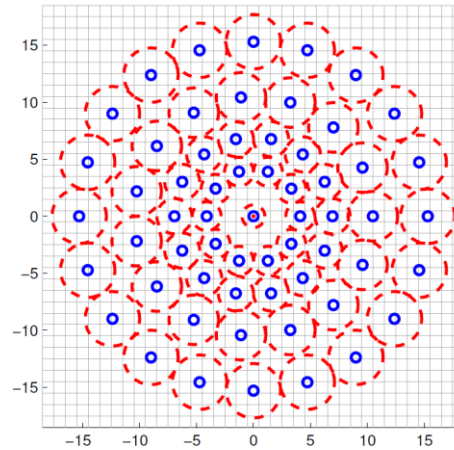


ABBILDUNG 10: ANORDNUNG VON PUNKTEN UM KEYPOINT BEIM BRISK-ALGORITHMUS (LEUTENEGER, CHLI & SIEGWART, 2011).

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 2011 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 Merkmalerkennung. 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 11 verdeutlicht wird.



ABBILDUNG 11: 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 aber immer noch 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 veröffentlichten Alcantarilla, Nuevo & Bartolli im folgenden Jahr eine überarbeitete Version des KAZE-Algorithmus namens AKAZE. Dieser bedient sich der mathematischen Technik der 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. Hierbei ergibt sich jedoch das Problem, dass auch in Bildpaaren ohne gemeinsame Inhalte derartige 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 und unterschiedlichen Anwendungskontexte kaum gerecht. Stattdessen wendet man ein Verfahren an, das erstmals von Lowe beschrieben wurde: Dabei betrachtet man das Verhältnis zwischen dem kleinsten und zweitkleinsten euklidischen 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 den Kapiteln 5 und 7 detailliert beschrieben.

3.6. Performancevergleiche der Algorithmen

In der Forschungsliteratur finden sich bereits unterschiedliche Versuche, die Performance der Merkmalerkennungs-Algorithmen miteinander zu vergleichen. Im Rahmen dieser Arbeit sind hierbei besonders diejenigen Vergleiche von Interesse, 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 unterschiedlicher 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 12 zu sehen.



ABBILDUNG 12: ARCHITEKTONISCHE BILDMOTIVE ALS BASIS FÜR PERFORMANCEMESSUNG DER MERKMALERKENNUNGS-ALGORITHMEN. OBEN: BILDPAAAR 1 (BUILDING DATASET), UNTEN: BILDPAAAR 2 (ROOFS DATASET) (TAREEN & SALEEM, 2018).

Tabelle 2 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 2: ANZAHL MATCHES UND BERECHNUNGSZEIT BEI GEBÄUDE-BILDERN FÜR UNTERSCHIEDLICHE ALGORITHMEN (IN ANLEHNUNG AN TAREEN & SALEEM, 2018).

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

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 aus dem Jahr 2016 Aufnahmen von Objekten durchgeführt und jeweils deren Rotation, Skalierung und Beleuchtung variiert. Anschließend 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 3 zu entnehmen sind, trotzdem bei der Beurteilung der Algorithmen von Nutzen sein.

TABELLE 3: ANTEIL KORREKTER MATCHES DER ALGORITHMEN FÜR VERSCHIEDENE INVARIANZTYPEN (IN ANLEHNUNG AN ANDERSSON & MARQUEZ, 2016).

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%

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).

Eine ähnliche Untersuchung von Zhang et al. beschränkt sich lediglich auf die Algorithmen SIFT, SURF und FAST, wobei bei Letzterem die Deskriptorenerstellung und das Matching ebenfalls über SURF durchgeführt wurden. Tabelle 4 zeigt das Ergebnis der Varianztests, wobei die Bewertung mit einem (schlechteste Performance) bis drei Sternen (beste Performance) vorgenommen wurde. Nur bei der Robustheit gegenüber Beleuchtungs-Varianzen erweist sich ein Algorithmus SIFT gegenüber als überlegen. Hinsichtlich der Geschwindigkeit konnte jedoch der FAST-Algorithmus die besten Ergebnisse liefern (Zhang et al., 2019).

TABELLE 4: ROBUSTHEIT DER ALGORITHMEN SIFT, SURF UND FAST BEZÜGLICH ROTATION, SKALIERUNG, WEICHZEICHNUNG, KOMPRESSION UND BELEUCHTUNG (ZHANG ET AL, 2019).

Methode	Invarianz				
	Rotation	Skalierung	Weichzeichnung	Kompression	Beleuchtung
SIFT	★★	★★★	★★	★★★	★
SURF	★	★	★★	★★	★★★
FAST	★	★★	★	★	★

4. Forschungsstand: Technisch

4.1. OpenCV

Bei der Entwicklung von Software, die sich der Merkmalerkennung bedient, kann auf unterschiedliche Weise vorgegangen werden. Eine gangbare Option ist sicherlich, einen der vorgestellten Algorithmen eigenhändig 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.

Üblicherweise wird heute jedoch zu Zwecken der Merkmalerkennung auf bestehende Softwarebibliotheken zurückgegriffen, wobei in der Regel OpenCV zum Einsatz kommt. Dadurch verringert sich nicht nur der Entwicklungsaufwand erheblich, es kann auch davon ausgegangen werden, dass die langjährige Entwicklung durch die Open-Source-Community ein Mindestmaß an Performancequalität und Stabilität gewährleistet. OpenCV kann dabei für vielfältige Anwendungszwecke im Bereich der Computer-Vision und Bildbearbeitung verwendet werden und ist auf zahlreichen Plattformen einsetzbar, etwa auf Windows, Linux, macOS, Android und iOS. Es gibt Programmbibliotheken für Programmiersprachen wie Python, Java und C++. ⁸

Mit der Bibliothek OpenCV.js steht auch eine Portierung für JavaScript zur Verfügung, die jedoch nur über einen eingeschränkten Funktionsumfang verfügt. Speziell die Algorithmen zur Merkmalerkennung stehen hierfür nur eingeschränkt zur Verfügung, jedoch sind BRISK und ORB bereits nutzbar. ⁹ Alternativ dazu bietet die Bibliothek jsfeat die Möglichkeit, FAST und ORB in einer reinen JavaScript-Anwendung zu verwenden. ¹⁰

Die bereits erwähnten Unterschiede zwischen den Algorithmen bezüglich der Lizenzrechte 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 auf diese Weise 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. ¹¹

⁸ <https://opencv.org/about/> (Letzter Zugriff: 30.3.2020)

⁹ <https://github.com/ucisysarch/opencvjs> (Letzter Zugriff: 30.3.2020)

¹⁰ <https://inspirit.github.io/jsfeat> (Letzter Zugriff: 30.3.2020)

¹¹ https://github.com/opencv/opencv_contrib (Letzter Zugriff: 30.3.2020)

Neben den genannten Merkmalerkennungs-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 beiden Kategorien Flann-Based-Matching und Brute-Force-Matching einordnen lassen:¹²

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

Wie bereits in Kapitel 2 erwähnt, unterstützt OpenCV alle Algorithmen, die im Rahmen dieser Arbeit miteinander verglichen werden sollen.

¹² https://docs.opencv.org/3.4/db/d39/classcv_1_1DescriptorMatcher.html
(Letzter Zugriff: 30.3.2020)

5. Lösungsansatz

Bei der Suche nach einer Lösung spielen vor allem zwei Aspekte eine wichtige Rolle: Die Umsetzung der Applikation, insbesondere in Hinblick auf deren Architektur, sowie die Wahl eines geeigneten Algorithmus für die Merkmalerkennung und das Matching.

Um die Applikation möglichst vielen Menschen zur Verfügung zu stellen, ist eine plattformunabhängige Umsetzung zu bevorzugen. Hierbei bietet sich die Entwicklung einer Progressive-Web-App an. Eine Installation ist für die Verwendung nicht nötig, jedoch kann diese auf den Betriebssystemen Android und iOS auf Wunsch dem Home-Screen hinzugefügt und damit ähnlich wie eine herkömmliche App genutzt werden.

Indem alle Anwendungsdaten auf einem Server vorgehalten werden, müssen auf den Geräten der BenutzerInnen keine Updates eingespielt werden. So kann auch sichergestellt werden, dass die Bilddaten, mit denen der Vergleich durchgeführt wird, immer auf dem aktuellen Stand sind. Zwar wäre es alternativ dazu möglich, eine Applikation zu entwickeln, die alle Bilddaten auf dem mobilen Gerät speichert und ggfs. auch dort die Merkmalerkennung und das Matching durchführt. Dies würde jedoch einen hohen Speicherverbrauch auf den mobilen Geräten mit sich bringen und bei Updates an den Bilddaten teilweise die Übertragung sehr großer Datenmengen erfordern.

Nachdem die wichtigsten Algorithmen zur Merkmalerkennung 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 Bildvergleiche liefern sämtliche Merkmalerkennungs-Algorithmen lediglich eine Menge von Matches. Aus diesen können mittels Lowes Distance-Ratio diejenigen Matches entnommen werden, welche mit hoher Wahrscheinlichkeit als korrekt gelten können (siehe Kapitel 3.5.). 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 jeweilige Algorithmus deshalb 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. Auch ein zu hoher Zeitverbrauch für Merkmalerkennung und Matching ist negativ zu werten, wobei jedoch die Qualität der Bildklassifikation generell eine höhere Priorität besitzt als die Geschwindigkeit.

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 13, 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 seinem Dateisystem und filtert diejenigen heraus, die sich in der Umgebung der Geräteposition befinden. Auf die verbleibenden Bilder wird der defaultmäßig festgelegte Merkmalerkennungs-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.

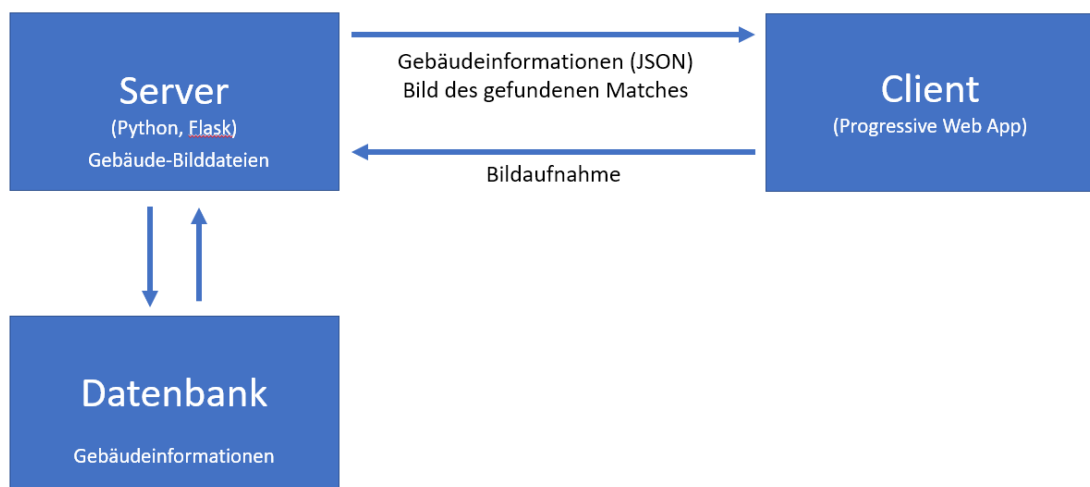


ABBILDUNG 13: ARCHITEKTUR VON BIDENT BUILDING IDENTIFICATION.

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 14 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.

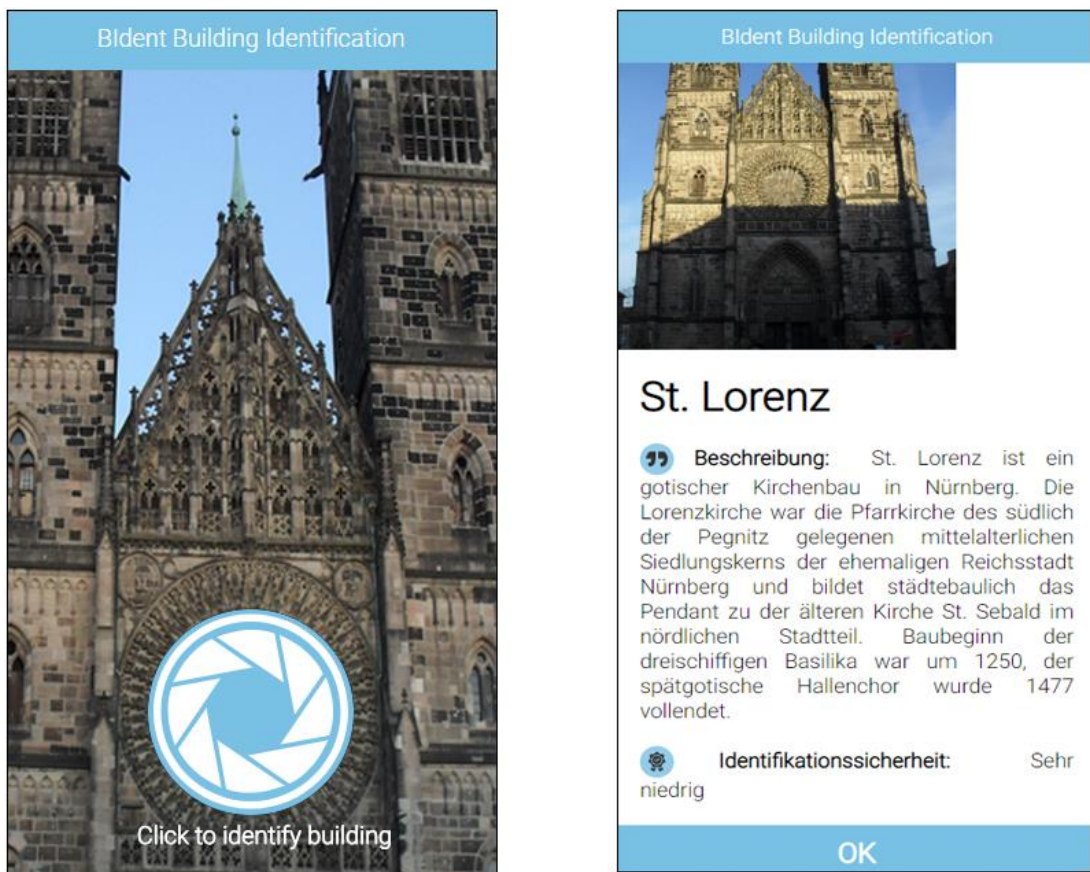
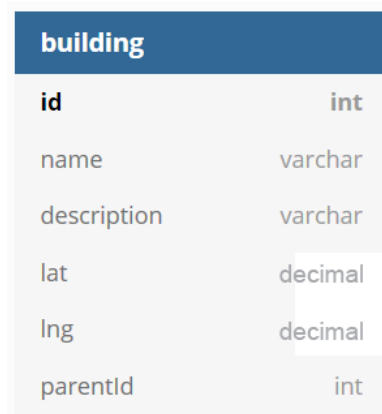


ABBILDUNG 14: SCREENSHOTS DER APPLIKATION BIDENT BUILDING IDENTIFICATION: HAUPTSEITE (LINKS) UND DETAILSEITE (RECHTS).

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. Als Bildformat wird stets 960 x 1280 Pixel gewählt, was einem Seitenverhältnis von 3:4 entspricht. Sollte die Kamera des Gerätes eine höhere oder geringere Auflösung aufweisen, wird das Bild auf diese Größe gestreckt. Da das Bild vor der Übertragung nicht komprimiert wird, ist es erforderlich, die Bildgröße zu beschränken. Andernfalls könnten Übertragungszeit und Datenverbrauch bei Geräten mit hoher Kameraauflösung inakzeptable Werte annehmen.

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 Gebäuden und Bauteilen gespeichert sind. Abbildung 15 zeigt die Spalten der Gebäude-Tabelle. Die den Gebäuden und Bauteilen zugehörigen 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 eindeutige Gebäude-Id.

The image shows a screenshot of a MySQL database table structure for a table named 'building'. The table has six columns: 'id' (int), 'name' (varchar), 'description' (varchar), 'lat' (decimal), 'lng' (decimal), and 'parentId' (int). The 'id' column is the primary key.

building	
id	int
name	varchar
description	varchar
lat	decimal
lng	decimal
parentId	int

ABBILDUNG 15: 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 eines Blob-Objekts entnommen. Das Blob-Objekt wird zuerst als Datei im JPEG-Format auf dem Server abgespeichert. Nun werden alle Gebäude aus der Datenbank entnommen und diejenigen herausgefiltert, die sich innerhalb einer Bounding Box um die Geräteposition befinden. Auf die Verwendung eines Spatial-Database wurde verzichtet, da die zu erwartenden Performancegewinne im gegebenen

Anwendungskontext den zusätzlichen Implementierungsaufwand nicht aufwiegen dürften. Es kann nun das hochgeladene Bild mit allen Bildern in der Umgebung verglichen werden, um das ähnlichste unter diesen zu ermitteln.

Dabei werden die Features und Deskriptoren der zu vergleichenden Bilder unter Benutzung des KAZE-Algorithmus berechnet. Das Matching erfolgt durch einen FLANN-based Matcher während auf die Verwendung einer Homographie verzichtet wurde. Für sämtliche Phasen der Merkmalerkennung und des Matchings wird OpenCV verwendet.

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 5 ist zu sehen, welche Anzahl Guter Matches beim KAZE-Algorithmus zu erwarten ist, wenn zwei Bilder unterschiedliche bzw. identische Motive aufweisen. Die Werte 94, 146 und 213 entsprechen dabei dem Median, dem obersten Dezilwert und dem Maximalwert für die Anzahl guter Matches bei Bildern unterschiedlicher Motive.

Auf Basis der eher stichpunkthaften Datengrundlage erscheint es nicht angebracht, konkrete Prozentwerte für die Sicherheit bzw. Wahrscheinlichkeit der korrekten Bildidentifikation zu berechnen bzw. anzugeben. Stattdessen erfolgt eine qualitative Einstufung in die Kategorien „Sehr niedrig“, „Niedrig“, „Mittel“ und „Hoch“.

TABELLE 5: KATEGORIEN FÜR DIE SICHERHEIT DER BILDKLASSIFIKATION MIT DEM KAZE-ALGORITHMUS (EIGENE DARSTELLUNG).

Sicherheitskategorie	Unterer Grenzwert	Oberer Grenzwert
Sehr niedrig	0	94
Niedrig	94	146
Mittel	146	213
Hoch	213	∞

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.

Im Rahmen dieser Arbeit werden nicht alle denkbaren Aufnahmebedingungen variiert. Ein Beispiel hierfür ist der Einfluss der Jahreszeiten auf die Bildinhalte, der von Valgren & Lilienthal für die Algorithmen SIFT und SURF untersucht wurde. Die Ergebnisse, die sich jedoch nur bedingt auf den gegebenen Anwendungsfall übertragen lassen, sprechen dabei für eine Überlegenheit von SURF hinsichtlich Matching-Qualität und Geschwindigkeit (Valgren & Lilienthal, 2007). Weitere Varianzen, wie etwa die von Zhang et al. untersuchte Weichzeichnung, spielen für den gegebenen Anwendungsfall ebenfalls keine Rolle (Zhang et al., 2019).

Alle Tests wurden auf einem Rechner durchgeführt, welcher mit einem Intel Core i5-4570-Prozessor mit 3.20 Ghz ausgestattet ist und über 8 GB RAM verfügt. Als System wurde Windows 10 in seiner 64-Bit-Version verwendet, wobei die Tests selbst in Python geschrieben wurden und für Merkmalerkennung und Matching OpenCV verwenden. Alle Aufnahmen zu Testzwecken, die nicht Teil des Oxford Buildings Dataset sind, wurden mit einer Samsung ES55-Digitalkamera erstellt, als Bildformat wurde jeweils 1280 x 960 Pixel verwendet.

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 von den Initiatoren jeweils durch eine Suche auf der Plattform Flickr nach den Namen wichtiger Gebäude in der Stadt Oxford ermittelt. Aus diesem Grund enthalten einige der Bilder nur Innenaufnahmen des Gebäudes, oder zeigen ein anderes Motiv, das lediglich von diesem Gebäude aus aufgenommen wurde. Hieraus ergibt sich jedoch der Vorteil, dass auch für das Matching von Gebäuden mit gänzlich unähnlichen Objekten nützliche Messergebnisse erzielt werden können (Philbin, J., Arandjelović, R. und Zisserman, (n.d.), Li et al., 2014).

Aus diesen Bildern wurde jedes 50ste ausgewählt und jeweils mit drei Vergleichsbildern verglichen, die in Abbildung 16 zu sehen sind. Die Vergleiche wurden jeweils mit allen sechs Merkmalerkennungs-Algorithmen durchgeführt.

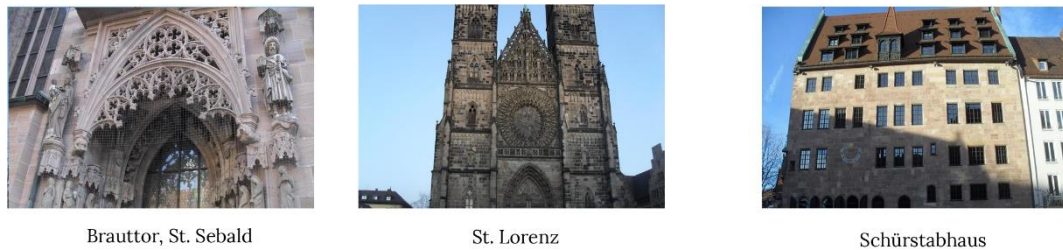


ABBILDUNG 16: BILDER VON GEBÄUDEN AUS NÜRNBERG FÜR DEN VERGLEICH MIT DEN AUFNAHMEN DES OXFORD BUILDINGS DATASET.

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.

Abbildung 17 zeigt dabei die größten Ausreißer für alle Algorithmen, also diejenigen Bilder, für die die größte Zahl Guter Matches ermittelt wurde.¹³ Dabei fällt auf, dass diese bei SIFT, BRISK, KAZE und AKAZE bei Bildern vorliegen, die gar keine Gebäude darstellen. Grundsätzlich besteht Grund zur Annahme, dass derartige Motive in der Anwendungspraxis nur in Ausnahmefällen 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.

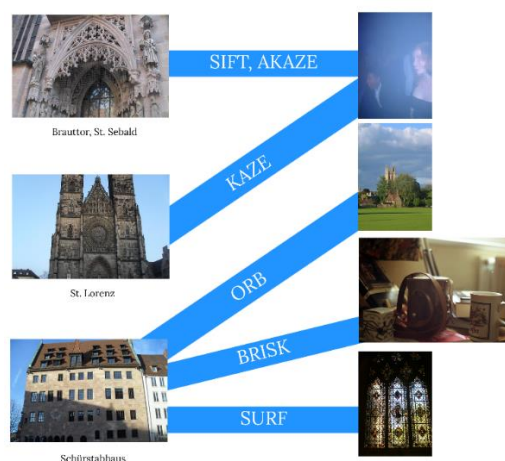


ABBILDUNG 17: AUSREIßER BEIM BILDVERGLEICH MIT AUFNAHMEN DES OXFORD BUILDINGS DATASET (PHILBIN, J., ARANDJELOVIĆ, R. UND ZISSERMAN, (N.D.)).

¹³ Für die Veröffentlichung in dieser Arbeit wurden die Gesichter der Personen auf den Bildern unkenntlich gemacht. Die Tests wurden jedoch mit den unveränderten Originalbildern durchgeführt.

Tabelle 6 sowie das zugehörige Box-Plot-Diagramm in Abbildung 18 liefern eine genauere Übersicht über die Streuung der Werte. Hierbei fällt auf, dass der SURF-Algorithmus eine besonders große Spannweite an Werten aufweist. ORB und KAZE weisen eine besonders geringe Anzahl von Ausreißern auf, während bei AKAZE und ORB die Streuung besonders gering ist. Hinsichtlich der Berechnungsdauer zeigen sich ebenfalls erhebliche Unterschiede zwischen den Algorithmen.

TABELLE 6: STATISTISCHE VERTEILUNG GUTER MATCHES UND GESCHWINDIGKEIT BEIM VERGLEICH MIT ALLEN BILDERN DES OXFORD BUILDINGS DATASETS (EIGENE DARSTELLUNG).

Name	Minimum	P ₁₀	Median	P ₉₀	Maximum	Gesamt-Berechnungs-dauer
SIFT	72	137	193	254	544	82,07 s
SURF	194	305	409	506	973	114,49 s
BRISK	40	66	94	132	297	138,78 s
ORB	0	4	9	16	27	4,14 s
KAZE	21	58	94	146	213	99,90 s
AKAZE	35	55	70	91	307	27,81 s

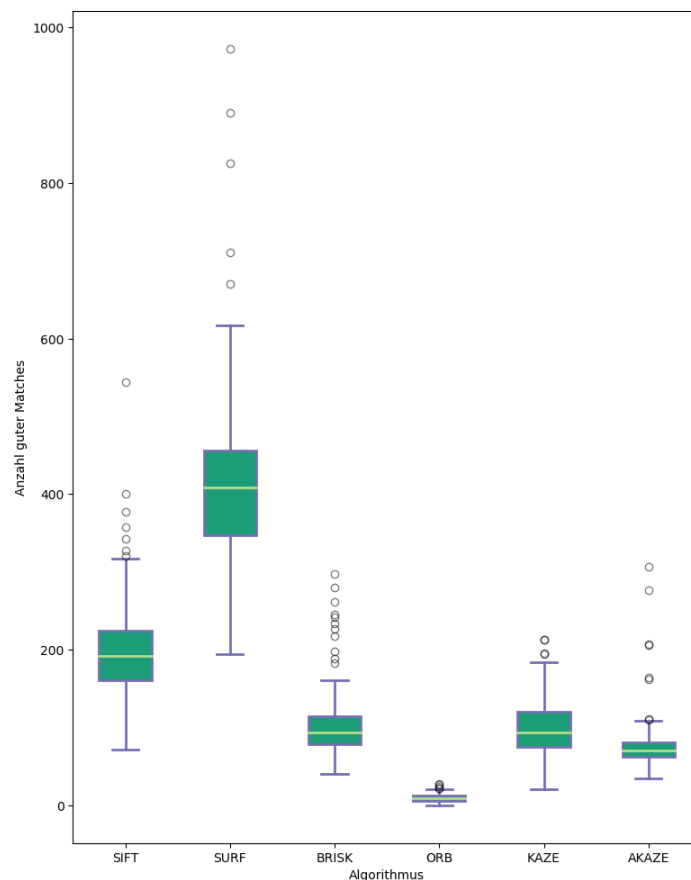


ABBILDUNG 18: BOXPLOTS MIT ANZAHL DER GUTEN MATCHES FÜR SÄMTLICHE ALGORITHMEN BEIM VERGLEICH MIT DEN BILDERN DES OXFORD BUILDINGS DATASETS.

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.

Zwar ist es kaum möglich, objektiv zu bestimmen, ab welchen Werten ein Algorithmus als geeignet eingestuft werden kann. Für die Evaluierung sollen jedoch die folgenden Einstufungen vorgenommen werden:

- Übersteigt die Anzahl Guter Matches das Maximum Guter Matches bei abweichenden Motiven, so hat die Identifikation einen hohen Sicherheitsgrad.
- Übersteigt die Anzahl Guter Matches den obersten Dezilwert Guter Matches bei abweichenden Motiven, liegt aber unterhalb des Maximums, so hat die Identifikation einen mittleren Sicherheitsgrad.
- Ist keine der beiden vorgenannten Bedingungen erfüllt, so hat die Identifikation einen niedrigen Sicherheitsgrad.

Um einen quantitativen Vergleich der Algorithmen zu ermöglichen, wird in den folgenden Kapiteln für jeden Invarianztyp die Anzahl von Bildvergleichen mit hohem, mittleren und niedrigem Sicherheitsgrad ermittelt. Anschließend werden diese im folgenden Maß mit Punkten gewichtet:

- 2 Punkte für eine Identifikation mit hohem Sicherheitsgrad
- 1 Punkt für eine Identifikation mit mittlerem Sicherheitsgrad
- 0 Punkte für eine Identifikation mit niedrigem Sicherheitsgrad

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. Wie in Abbildung 19 zu sehen, weicht auch die künstliche Beleuchtung je nach Gebäude deutlich voneinander ab.



ABBILDUNG 19: GEBÄUDE UND BAUTEILE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH TAG-/NACHT-UNTERSCHIEDEN.

Tabelle 7 zeigt die Anzahl Guter Matches für die Tag-/Nacht-Bildvergleiche bei den sechs Objekten. Die Werte von BRISK liegen dabei nur in einem einzigen Fall unter dem obersten Dezilwert für abweichende Motive aus Kapitel 7.1. Bei den übrigen Algorithmen erfolgt diese Unterschreitung in zwei oder drei Fällen.

TABELLE 7: ANZAHL GUTER MATCHES FÜR ALGORITHMEN BEI TAG-/NACHT-VARIANZ (EIGENE DARSTELLUNG).

Name	Matches Brauttor	Matches Frauenkirche	Matches Lorenzkirche	Matches Nassauer Haus	Matches Altes Rathaus	Matches Schürstabhaus
SIFT	201	357	991	90	384	165
SURF	502	508	1298	358	600	482
BRISK	159	300	581	55	156	159
ORB	11	24	8	23	22	13
KAZE	55	162	617	212	109	249
AKAZE	98	183	592	82	78	87

Die für die Berechnung dieser Guten Matches benötigte Zeit ist aus Tabelle 8 zu entnehmen. Für alle Objekte benötigt KAZE die längste und ORB die kürzeste Berechnungszeit.

TABELLE 8: DAUER FÜR BERECHNUNG GUTER MATCHES FÜR ALGORITHMEN BEI TAG-/NACHT-VARIANZ (EIGENE DARSTELLUNG).

Name	Dauer Brauttor	Dauer Frauenkirche	Dauer Lorenzkirche	Dauer Nassauer Haus	Dauer Altes Rathaus	Dauer Schürstabhaus
SIFT	0,72 s	0,78 s	0,96 s	0,57 s	0,73 s	0,64 s
SURF	0,94 s	1,10 s	1,23 s	0,62 s	1,11 s	1,08 s
BRISK	0,25 s	0,43 s	0,37 s	0,07 s	0,25 s	0,21 s
ORB	0,05 s	0,05 s	0,05 s	0,04 s	0,05 s	0,05 s
KAZE	2,15 s	2,43 s	2,26 s	2,08 s	2,43 s	2,20 s
AKAZE	0,41 s	0,45 s	0,44 s	0,39 s	0,43 s	0,43 s

Die Ergebnisse legen nahe, dass keiner der Algorithmen eine hinreichende Robustheit gegenüber Tag-/Nacht-Unterschieden aufweist, sie also alle als ungeeignet eingestuft werden müssen. Um trotzdem eine sichere Identifikation zu ermöglichen, ist ggfs. die Hinterlegung von zusätzlichen Nachtaufnahmen in der Bild-Datenbank nötig. Im Vergleich zu den anderen Algorithmen sind jedoch BRISK und KAZE robuster, wie die Punktbewertung in Tabelle 9 zeigt.

TABELLE 9: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ – TAG/NACHT (EIGENE DARSTELLUNG)

Name	Werte mit niedrigem Sicherheitsgrad	Werte mit mittlerem Sicherheitsgrad	Werte mit hohem Sicherheitsgrad	Punktbewertung
SIFT	3	2	1	4
SURF	3	2	1	4
BRISK	1	3	2	7
ORB	3	3	0	3
KAZE	2	2	2	6
AKAZE	3	2	1	4

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 20 zu sehen.

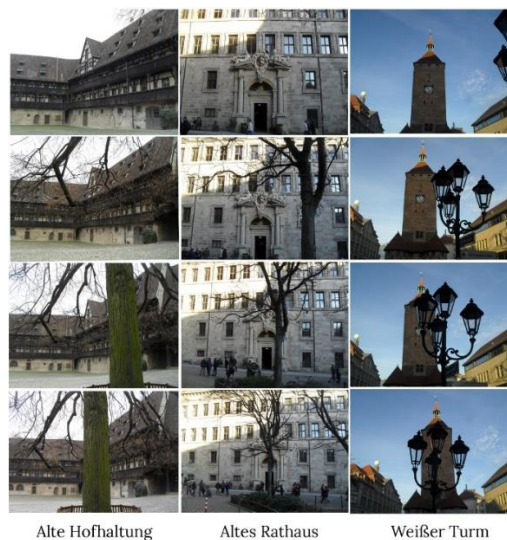


ABBILDUNG 20: GEBÄUDE AUS BAMBERG UND NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER OKKLUSION.

Es wurde nun jeweils das Bild ohne Verdeckung (in Abbildung 19 ganz oben) mit den drei verdeckten Bildern des gleichen Gebäudes verglichen. Tabelle 10 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 teilweise zu einer höheren Zahl Guter Matches führt als eine weniger zentrale Position.

KAZE liefert stets Werte, die oberhalb des Maximums bei abweichenden Motiven liegen. Die restlichen Algorithmen mit Ausnahme von ORB weisen in allen Fällen Werte auf, die den obersten Dezilwert bei abweichenden Motiven überschreiten.

TABELLE 10: ANZAHL GUTER MATCHES FÜR ALGORITHMEN BEI OKKLUSIONS-VARIANZ (EIGENE DARSTELLUNG).

Name	Matches Alte Hofhaltung			Matches Altes Rathaus			Matches Weißer Turm		
	#1	#2	#3	#1	#2	#3	#1	#2	#3
SIFT	761	461	492	1434	1068	694	382	289	265
SURF	1316	815	1007	3171	2278	1687	1039	767	712
BRISK	585	282	317	1720	795	401	466	262	367
ORB	16	10	6	59	64	44	47	54	100
KAZE	543	292	297	1259	757	559	414	329	368
AKAZE	391	205	219	892	547	397	252	180	234

Die Dauer für die Berechnung der Guten Matches kann aus Tabelle 11 entnommen werden. ORB ist dabei deutlich schneller als andere Algorithmen während KAZE als langsamster Algorithmus gelten kann. Es fällt auf, dass die Berechnungsdauer bei BRISK erheblich variiert.

TABELLE 11: GESAMTDAUER FÜR BERECHNUNG ALLER GUTER MATCHES FÜR ALGORITHMEN BEI OKKLUSIONS-VARIANZ (EIGENE DARSTELLUNG).

Name	Gesamtdauer Alte Hofhaltung	Gesamtdauer Altes Rathaus	Gesamtdauer Weißer Turm
SIFT	2,19 s	2,42 s	1,36 s
SURF	3,60 s	3,71 s	1,43 s
BRISK	4,68 s	3,30 s	0,27 s
ORB	0,12 s	0,13 s	0,09 s
KAZE	4,85 s	5,10 s	4,37 s
AKAZE	1,01 s	1,26 s	0,84 s

Wie Tabelle 12 zu entnehmen ist, erreicht KAZE stets einen hohen Sicherheitsgrad und kann deshalb als der robusteste Algorithmus gegenüber der Okklusion gelten.

TABELLE 12: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER OKKLUSION (EIGENE DARSTELLUNG)

Name	Werte mit niedrigem Sicherheitsgrad	Werte mit mittlerem Sicherheitsgrad	Werte mit hohem Sicherheitsgrad	Punktbewertung
SIFT	0	5	4	13
SURF	0	3	6	15
BRISK	0	2	7	16
ORB	2	1	6	13
KAZE	0	0	9	18
AKAZE	0	5	4	13

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 21 zu erkennen, erzeugt der Sonnenstand dabei auch stark abweichende Schattenwürfe, die den Bildvergleich beeinflussen könnten. Für jedes Gebäude wird das zentrale Bild (in Abbildung 21 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.

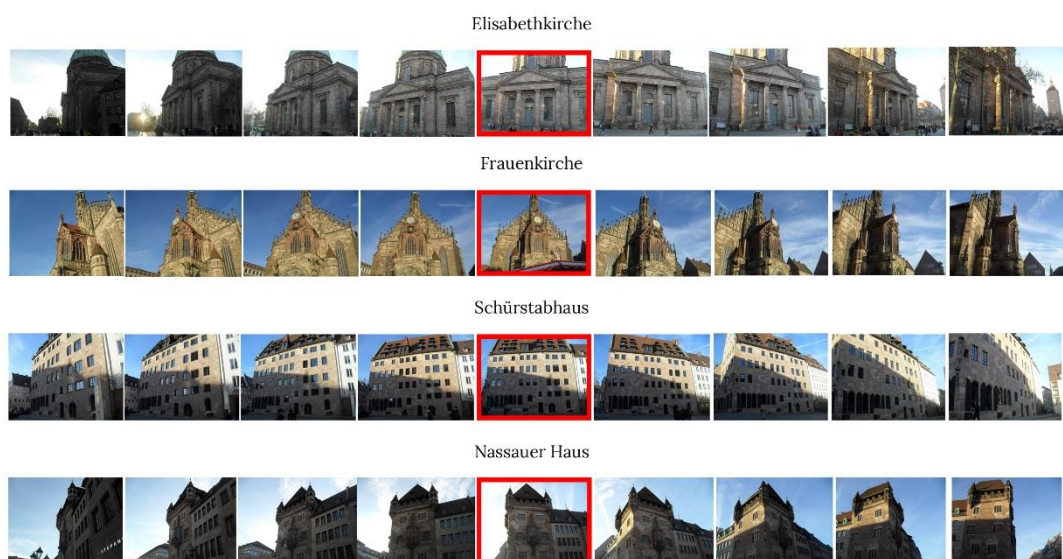


ABBILDUNG 21: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER HORIZONTALEN PERSPEKTIVE.

Generell lässt sich, wie den Tabellen 13-16 zu entnehmen ist, eine deutliche Abnahme Guter Matches beobachten, je stärker der Aufnahmewinkel sich von der Frontalansicht unterscheidet. Dabei unterschreiten die Werte bei allen Algorithmen mehrfach den obersten Dezilwert abweichender Motive. Bezüglich der Geschwindigkeit erweisen sich KAZE und ORB abermals als langsamster respektive schnellster Algorithmus.

TABELLE 13: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: ELISABETHKIRCHE, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	255	199	419	1038	891	577	251	199	4,44 s
SURF	531	541	520	1481	1178	779	462	395	7,80 s
BRISK	107	115	163	520	519	294	118	49	3,07 s
ORB	4	4	15	41	36	16	10	12	0,25 s
KAZE	56	91	168	383	332	179	96	80	11,24 s
AKAZE	51	45	85	254	230	139	71	26	2,31 s

TABELLE 14: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: FRAUENKIRCHE, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	173	274	492	1080	1057	480	286	149	5,27 s
SURF	353	375	511	985	965	542	349	299	6,54 s
BRISK	134	191	382	979	938	338	201	135	10,59 s
ORB	11	12	17	51	55	12	6	5	0,26 s
KAZE	68	125	275	741	903	272	144	66	11,34 s
AKAZE	83	91	188	501	528	148	95	70	2,94 s

TABELLE 15: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: SCHÜRSTABHAUS, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	184	381	587	1166	828	372	232	174	3,66 s
SURF	559	861	1389	2730	1961	999	622	458	6,39 s
BRISK	196	331	763	1846	1128	389	161	103	4,00 s
ORB	23	45	99	174	116	34	10	8	0,24 s
KAZE	213	301	648	1393	959	339	216	122	10,69 s
AKAZE	134	180	428	1219	725	207	123	92	2,30 s

TABELLE 16: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (HORIZONTAL) – MOTIV: NASSAUER HAUS, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	#5	#6	#7	#8	Gesamtdauer
SIFT	169	156	188	269	370	157	153	137	3,23 s
SURF	233	241	418	638	883	322	211	237	4,10 s
BRISK	48	34	104	129	216	47	45	32	1,05 s
ORB	3	18	34	28	10	19	7	10	0,20 s
KAZE	89	104	176	215	300	134	118	156	9,86 s
AKAZE	66	54	110	122	214	68	49	49	1,89 s

Hinsichtlich horizontaler Perspektivänderungen kann keiner der Algorithmen als umfassend geeignet eingestuft werden. Es erscheint aus diesem Grunde erforderlich, Aufnahmen aus weiteren Perspektiven in die Bilddatenbank aufzunehmen. Insgesamt ist die Klassifikationssicherheit bei KAZE höher als bei den restlichen Algorithmen, wie Tabelle 17 zeigt.

TABELLE 17: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER PERSPEKTIVÄNDERUNGEN (HORIZONTAL) (EIGENE DARSTELLUNG)

Name	Werte mit niedrigem Sicherheitsgrad	Werte mit mittlerem Sicherheitsgrad	Werte mit hohem Sicherheitsgrad	Punktbewertung
SIFT	14	10	8	26
SURF	13	12	7	26
BRISK	12	9	11	31
ORB	16	5	11	27
KAZE	14	3	15	33
AKAZE	7	20	5	30

7.2.4. Perspektive - Vertikal

Auch vertikale Perspektivänderungen verdienen eine genauere Betrachtung. Hierfür wurden drei Objekte aus unterschiedlicher Entfernung aufgenommen, wodurch sich auch eine Veränderung des vertikalen Blickwinkels ergab. Der Vergleich erfolgte jeweils zwischen dem Bild, das der Normalsicht am meisten entspricht (in Abbildung 22 unten und rot markiert) und den weiteren Aufnahmen des gleichen Objekts.

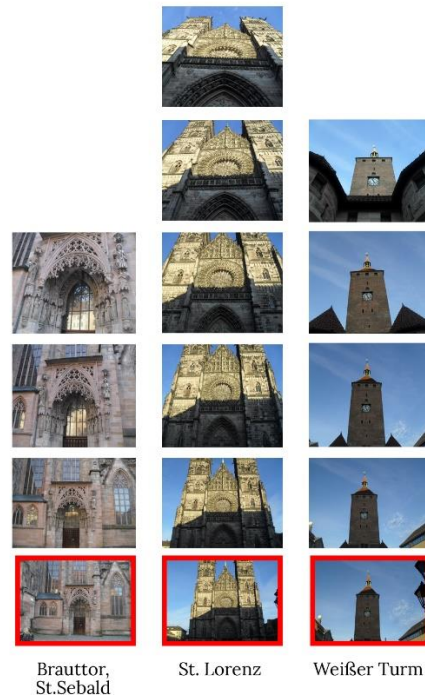


ABBILDUNG 22: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS
HINSICHTLICH DER VERTIKALEN PERSPEKTIVE.

Zwar gelingt es keinem Algorithmus, stets den obersten Dezilwert abweichender Motive zu überschreiten. Doch immerhin ist dies bei ORB bei zwei von drei Gebäuden der Fall, bei BRISK und KAZE jeweils bei einem. Die Berechnungsdauer ist bei ORB abermals am geringsten, während BRISK bei einem Gebäude und KAZE bei den sonstigen die meiste Zeit verbrauchen (siehe Tabellen 18-20).

TABELLE 18: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL)
– MOTIV: BRAUTTOR, ST. SEBALD, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	Gesamtdauer
SIFT	1854	692	252	2,32 s
SURF	2312	825	439	3,89 s
BRISK	1360	504	139	2,38 s
ORB	166	45	19	0,12 s
KAZE	891	328	147	5,03 s
AKAZE	744	287	69	1,07 s

TABELLE 19: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL)
– MOTIV: ST. LORENZ, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	#5	Gesamtdauer
SIFT	3083	1758	658	204	125	4,13 s
SURF	2859	1492	625	377	345	5,12 s
BRISK	3914	1620	500	118	91	12,35 s
ORB	154	75	17	8	16	0,22 s
KAZE	2889	1658	584	174	106	8,48 s
AKAZE	1641	916	207	103	89	2,39 s

TABELLE 20: PERFORMANCE DER ALGORITHMEN FÜR PERSPEKTIVISCHE VARIANZ (VERTIKAL)
– MOTIV: WEIßER TURM, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	Gesamtdauer
SIFT	251	173	140	101	1,43 s
SURF	742	416	216	132	1,78 s
BRISK	308	162	103	45	0,35 s
ORB	99	70	30	21	0,11 s
KAZE	308	176	93	66	5,34 s
AKAZE	192	112	69	26	0,99 s

Obwohl die Identifikationssicherheit bei allen Algorithmen teilweise niedrige Werte erreicht, zeigen die Bewertungen aus Tabelle 21, dass ORB insgesamt deutlich bessere Ergebnisse liefert als die anderen Kandidaten.

TABELLE 21: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER PERSPEKTIVÄNDERUNGEN (VERTIKAL) (EIGENE DARSTELLUNG)

Name	Werte mit niedrigem Sicherheitsgrad	Werte mit mittlerem Sicherheitsgrad	Werte mit hohem Sicherheitsgrad	Punktbewertung
SIFT	7	0	5	10
SURF	6	3	3	9
BRISK	4	2	6	14
ORB	1	4	7	18
KAZE	3	3	6	15
AKAZE	4	5	3	11

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 23). Die angegebenen Gradwerte stellen dabei jedoch nur eine Approximation an den tatsächlichen Wert dar. Der Vergleich erfolgte jeweils zwischen dem Originalbild und einem der acht Rotationsbilder.

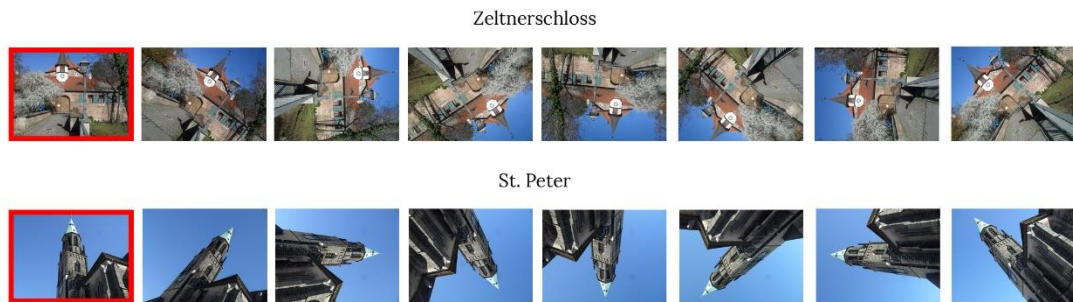


ABBILDUNG 23: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS
HINSICHTLICH DER ROTATION.

Wie den Tabellen 22 und 23 zu entnehmen ist, ist das Matching deutlich erfolgreicher, wenn der Rotationswinkel 90 Grad bzw. ein Vielfaches davon beträgt. Alle Algorithmen liegen stets deutlich über dem Maximalwert bei abweichenden Motiven. Während auch hier ORB am schnellsten und KAZE in einem Fall am langsamsten ist, fällt eine deutliche Geschwindigkeitsschwankung bei BRISK auf, der beim Objekt Zeltnerschloss weit mehr Zeit beansprucht als alle anderen Algorithmen.

TABELLE 22: PERFORMANCE DER ALGORITHMEN FÜR ROTATIONSVARIANZ – MOTIV:
ZELTNERSCLOSS, NÜRNBERG (EIGENE DARSTELLUNG).

Name	45°	90°	135°	180°	225°	270°	315°	Gesamtdauer
SIFT	4608	4121	4125	4949	5095	4400	5164	7,14 s
SURF	2537	5297	2337	5091	2565	5308	2581	7,98 s
BRISK	9764	8172	8573	10082	9731	8772	10902	39,50 s
ORB	271	220	260	261	244	289	278	0,32 s
KAZE	3973	3621	4152	4652	4323	3704	4461	10,50 s
AKAZE	3441	3061	3535	3913	3712	3207	3933	3,38 s

TABELLE 23: PERFORMANCE DER ALGORITHMEN FÜR ROTATIONSVARIANZ – MOTIV: ST.
PETER, NÜRNBERG (EIGENE DARSTELLUNG).

Name	45°	90°	135°	180°	225°	270°	315°	Gesamtdauer
SIFT	1326	1522	1477	1595	1542	1519	1539	2,84 s
SURF	1207	2886	1473	3397	1288	2746	1289	3,87 s
BRISK	2737	2896	2615	3085	2821	2926	3117	2,69 s
ORB	362	336	320	349	315	334	310	0,20 s
KAZE	2004	2592	2446	2810	2351	2415	2417	9,34 s
AKAZE	1691	1983	1976	2167	1932	1919	1981	1,91 s

Da alle Algorithmen durchgehend eine hohe Identifikationssicherheit aufweisen, erhalten sie auch allesamt die gleiche Punktbewertung (siehe Tabelle 24).

TABELLE 24: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER ROTATION (EIGENE DARSTELLUNG)

Name	Werte mit niedrigem Sicherheitsgrad	Werte mit mittlerem Sicherheitsgrad	Werte mit hohem Sicherheitsgrad	Punktbewertung
SIFT	0	0	14	28
SURF	0	0	14	28
BRISK	0	0	14	28
ORB	0	0	14	28
KAZE	0	0	14	28
AKAZE	0	0	14	28

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 Abwandlung des Zoomfaktors der Kamera, wobei die Bildqualität insbesondere bei digitalem Zoom deutlich abnimmt. Im Zuge der Evaluierung wurden beide Herangehensweisen verwendet. Abbildung 24 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.

Zeltnerschloss



St. Peter



ABBILDUNG 24: GEBÄUDE AUS NÜRNBERG ALS BASIS FÜR DIE INVARIANZ-TESTS HINSICHTLICH DER SKALIERUNG.

Allen Algorithmen mit Ausnahme von ORB gelingt es nur bei einem einzigen Gebäude, stets Werte oberhalb des obersten Dezilwerts bei abweichenden Motiven zu produzieren. Beim Objekt St. Peter verfehlen jedoch alle dieses Ziel, wobei SIFT und SURF weniger deutlich fehlschlagen. Auch für diesen Test ergibt sich wieder eine einmalige hohe Berechnungsdauer bei BRISK, während ansonsten ORB und KAZE der schnellste und langsamste Algorithmus sind (siehe Tabellen 25 und 26).

TABELLE 25: PERFORMANCE DER ALGORITHMEN FÜR SKALIERUNGSVARIANZ – MOTIV: ZELTNERSCLOSS, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	Gesamtdauer
SIFT	653	428	315	257	4,38 s
SURF	1141	710	607	549	4,64 s
BRISK	557	374	248	214	18,61 s
ORB	40	29	20	12	0,18 s
KAZE	422	269	199	180	6,23 s
AKAZE	335	232	132	109	1,78 s

TABELLE 26: PERFORMANCE DER ALGORITHMEN FÜR SKALIERUNGSVARIANZ – MOTIV: ST. PETER, NÜRNBERG (EIGENE DARSTELLUNG).

Name	#1	#2	#3	#4	Gesamtdauer
SIFT	979	572	248	214	1,97 s
SURF	2128	1083	567	446	3,16 s
BRISK	480	198	80	27	0,43 s
ORB	48	8	9	5	0,11 s
KAZE	342	154	32	11	5,59 s
AKAZE	282	121	22	21	1,04 s

Auch wenn keiner der Algorithmen durchgängig eine hohe Identifikationssicherheit aufweist, ist das Ergebnis von SURF leicht besser als das der anderen Algorithmen (siehe Tabelle 27).

TABELLE 27: BEWERTUNG DER ALGORITHMEN FÜR INVARIANZ GEGENÜBER SKALIERUNG (EIGENE DARSTELLUNG)

Name	Werte mit niedrigem Sicherheitsgrad	Werte mit mittlerem Sicherheitsgrad	Werte mit hohem Sicherheitsgrad	Punktbewertung
SIFT	2	3	3	9
SURF	1	4	3	10
BRISK	2	3	3	9
ORB	4	1	3	7
KAZE	2	3	3	9
AKAZE	2	5	1	7

7.3. Fazit

Tabelle 28 fasst die Ergebnisse der vorhergehenden Evaluierungs-Kapitel zusammen. Für jeden Invarianztyp ist der Algorithmus mit der höchsten Identifikationssicherheits-Bewertung grün markiert. KAZE erzielt dabei in zwei Kategorien die höchste Punktwertung, während dies SIFT und AKAZE in keiner gelingt. Alle anderen Algorithmen erzielen für jeweils einen Invarianztyp die höchste Wertung.

TABELLE 28: VERGLEICH DER ALGORITHMUS-PUNKTBEWERTUNGEN FÜR VERSCHIEDENE INVARIANZTYPEN – ABSOLUTE ZAHLEN (EIGENE DARSTELLUNG).

Name	Performance für Invarianztyp					
	Tag/ Nacht	Okklusion	Perspektive horizontal	Perspektive vertikal	Rotation	Skalierung
SIFT	4	13	26	10	28	9
SURF	4	15	26	9	28	10
BRISK	7	16	31	14	28	9
ORB	3	13	27	18	28	7
KAZE	6	18	33	15	28	9
AKAZE	4	13	30	11	28	7

Tabelle 29 zeigt den jeweiligen Anteil von Vergleichen mit hoher Identifikationssicherheit für jeden Invarianztyp und in der rechten Spalte einen daraus ermittelten Durchschnittswert über alle Invarianztypen hinweg. Hierbei liegt KAZE mit deutlichem Abstand vor den anderen Algorithmen.

Name	Anteil Werte mit hohem Sicherheitsgrad nach Invarianztyp						Durch- schnitt
	Tag/ Nacht	Okkl.	Persp. horiz.	Persp. vert.	Rotation	Skal.	
SIFT	16,67 %	44,44 %	25,00 %	41,67 %	100,00 %	37,50 %	44,21 %
SURF	16,67 %	66,67 %	21,88 %	25,00 %	100,00 %	37,50 %	44,62 %
BRISK	33,33 %	77,77 %	34,38 %	50,00 %	100,00 %	37,50 %	55,50 %
ORB	0,00 %	66,67 %	34,38 %	58,33 %	100,00 %	37,50 %	49,48 %
KAZE	33,33 %	100,00 %	46,88 %	50,00 %	100,00 %	37,50 %	61,29 %
AKAZE	16,67 %	44,44 %	15,63 %	25,00 %	100,00 %	12,50 %	35,71 %

Aufgrund dieser Ergebnisse erscheint es gerechtfertigt, als Default-Algorithmus der Applikation KAZE zu verwenden, auch wenn dessen hohe Identifikationssicherheit mit einer durchgehend hohen Berechnungsdauer erkauft wird.

8. Diskussion

Aufgrund der beobachteten Ergebnisse muss KAZE als der Merkmalerkennungs-Algorithmus mit der höchsten Eignung für die mobile Bildklassifikation historischer Gebäude und Bauteile angesehen werden. Zwar erweist sich insbesondere ORB als deutlich schnellere Alternative, doch wird dies mit einer unzufriedenstellenden Identifikationssicherheit erkaufte. 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 KAZE 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 muss.

Für die Weiterentwicklung der Applikation bietet sich schließlich noch eine Reihe von Möglichkeiten an. Die Geschwindigkeit der Berechnung kann etwa 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.

Des Weiteren ist es denkbar, vor der Übermittlung der Bilddaten an den Server eine Bildkompression durchzuführen, um die übertragene Datenmenge und somit auch die Übertragungsgeschwindigkeit deutlich zu verkleinern. Hierbei sind jedoch die möglichen Auswirkungen auf die Matching-Qualität zu beachten.

Für BenutzerInnen mit einem hohen Interesse an Bildklassifikations-Themen könnte man außerdem die Möglichkeit bieten, den Merkmalerkennungs-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.

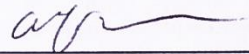
- Tareen, S.A.K., und Saleem, Z. „A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK.“ 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). IEEE, 2018. S. 1-10.
- Valgren, C. und Lilientahl, A.J. „SIFT, SURF and Seasons: Long-term Outdoor Localization using Local Features.“ ECMR 2007: Proceedings of the European Conference on Mobile Robots, 2007. S. 253-258.
- 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.

6.4.2020

(Datum)



(Unterschrift)