Identifikation von historischen Gebäuden und Bauteilen durch Bildklassifikation

Christof Wittmann

Bachelorarbeit

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

4.4.2019

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

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

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

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

In dieser Arbeit sollen nun die wichtigsten dieser Algorithmen miteinander verglichen werden, insbesondere in Bezug auf ihre Eignung für die Klassifikation historischer Gebäude und Bauteile. Dabei sollen zuerst die verfügbaren Algorithmen und ihre Beziehung zueinander dargestellt werden. 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 identifizierte Objekt anzuzeigen. In Hinblick auf diesen Anwendungsfall wird schließlich die Performance der verfügbaren Algorithmen getestet, um zu ermitteln, welche(r) von ihnen am Besten geeignet ist.

2. Problemstellung

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. Hierbei ist selbstverständlich auch zu berücksichtigten, dass die korrekte Identifizierung möglichst unabhängig von den äußeren Umständen der Aufnahme sein sollte.

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

Als weitere Entscheidungsgrundlage kann dabei die Tatsache dienen, dass es sich dabei auch um die *Feature Detection*-Algorithmen handelt, die von der populären *Computer Vision*-Bibliothek OpenCV zur Verfügung gestellt werden. [https://docs.opencv.org/master/d5/d51/group__features2d__main.html]

TABELLE X. ALGORITHMEN ZUR MERKMALSERKENNUNG

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

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

Alle Algorithmen und Unterschiede kurz vorstellen, gerade historisch. Auf spätere Kapitel verweisen.

3. Forschungsstand: Methoden der Bilderkennung

3.1. Grundprinzipien der Merkmalserkennung (Feature Detection)

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

Die fotographische Aufnahme eines Objekts kann auf sehr unterschiedliche Weise erfolgen, 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 von Kamera und Person beeinflusst werden, wobei die Möglichkeiten ggfs. durch die Umgebungssituation des Objekts eingeschränkt werden. Mittels der Kameraeinstellungen ist etwa die Helligkeit oder Farbbalance der Aufnahme wählbar, ebenso das Format des erzeugten Bildes. Weniger Einfluss hat die fotografierende Person auf die Lichtverhältnisse, insbesondere im Freien. Selbst die Wahl einer geeigneten Tageszeit und der Einsatz künstlicher Beleuchtung können nicht verhindern, dass örtliche Lichtverhältnisse stark durch örtliche Wetterverhältnisse 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, auch wenn die Aufnahmen in vielerlei Hinsicht erheblich voneinander abweichen (Andersson & Marquez, 2016).

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

den Inhalt des Bildes auf der höchstmöglichen Ebene zu beschreiben. Entsprechend trainierte neuronale Netzwerke können somit bestimmte Bildbestandteile erkennen und klassifizieren, wobei diese Verallgemeinerung jedoch auch mit einem Informationsverlust verbunden ist, der dazu führt, dass die Gleichheit von Objekten einer gemeinsamen Kategorie auf diese Weise schwer festzustellen ist. Ein weiterer Nachteil des Maschinellen Lernens ist der Bedarf an umfangreichen Mengen von Trainingsdaten. Die Merkmalserkennung nimmt dagegen keine Generalisierung oder Klassifikation vor. Sie ist nicht nur für den direkten Ähnlichkeitsvergleich von Bildern einsetzbar, sondern etwa auch beim Videotracking, dem *Image Stitching* oder bei der dreidimensionalen Rekonstruktion von Objekten auf Basis photographischer Aufnahmen. [Scherer, S. 2]

3.2. Methoden der Merkmalserkennung

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

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

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

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

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

Beschreibungen aus KAZE Features für folgende Kapitel verwenden.

3.3. Scale-Invariant Feature Transform (SIFT)

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

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

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

3.3.1. Scale-Space Extrema Detection

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

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

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

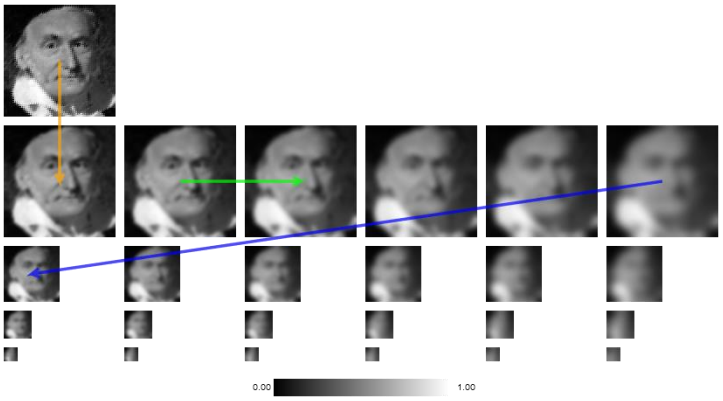

Abbildung x: Bilderzeugung im Rahmen der Scale-Space Extrema Detection (Quelle)

Nun werden aus diesen Bildern mittels der Difference of Gaussian (DoG)-Methode Differenzbilder generiert. Hierfür werden jeweils zwei innerhalb einer Oktave nebeneinanderliegend Bilder verwendet. In Abbildung x ist das Ergebnis zu sehen, wobei die Zahl der Bilder pro Oktave nun um eins verringert ist.

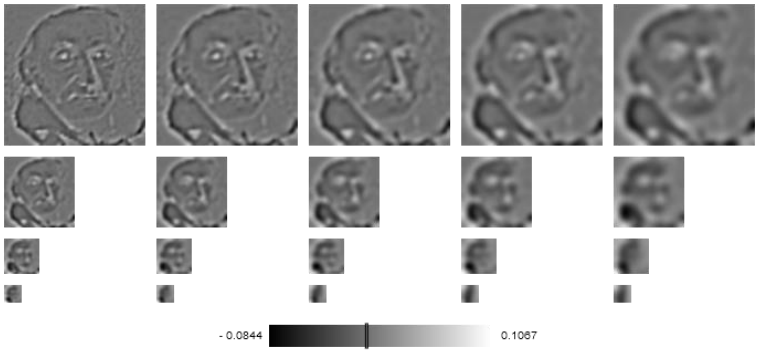

Abbildung x: Differenzbilder durch Difference of Gaussian-Berechnung (Quelle)

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 die angrenzenden neun Pixel in den Differenzbildern der nächstoberen und nächstunteren Oktaven.

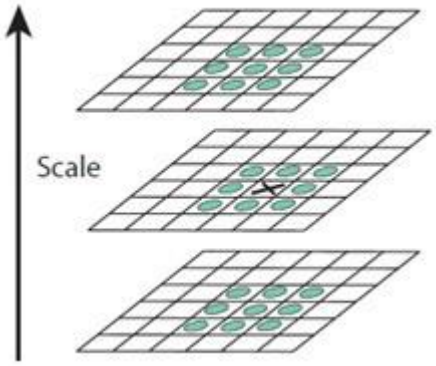

Abbildung x: Pixel-Nachbarschaftsvergleiche bei der Scale-Space Extrema Detection (Quelle)

Um als potentieller Keypoint in Frage zu kommen, muss ein Pixel einen höheren bzw. niedrigeren Wert aufweisen als alle 26 Nachbarpixel. Hierdurch wird die Skalierungsinvarianz gewährleistet (Warum? Sinn erklären.) (Lowe, S. 94-97, Andersson & Marquez, 2016).

3.3.2. Keypoint Localization

Die im letzten Schritt ermittelten Keypoints müssen 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, S. 97-99, 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 alle Keypoints Pixel in einem weichgezeichneten Bild sind, besteht ihre Umgebung aus Helligkeitsverläufen (Gradients). Für diese Verläufe, welche in Abbildung x zu sehen sind, können sowohl Intensität als auch Richtung ermittelt werden.

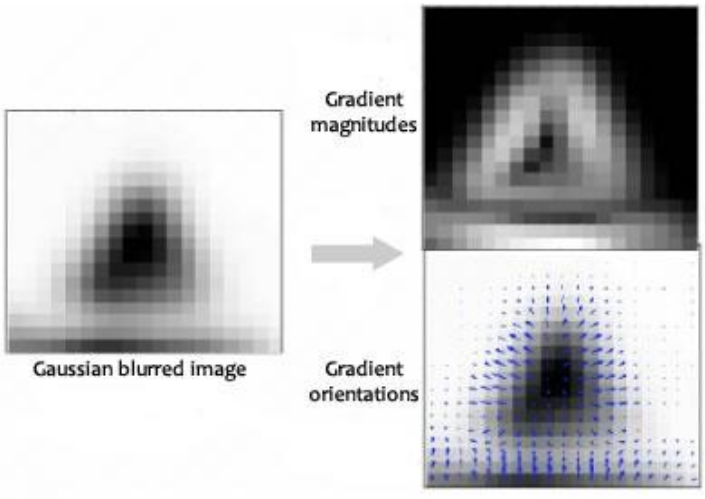

Abbildung x: Helligkeitsverläufe in Pixel-Nachbarschaft eines Keypoints (Quelle)

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

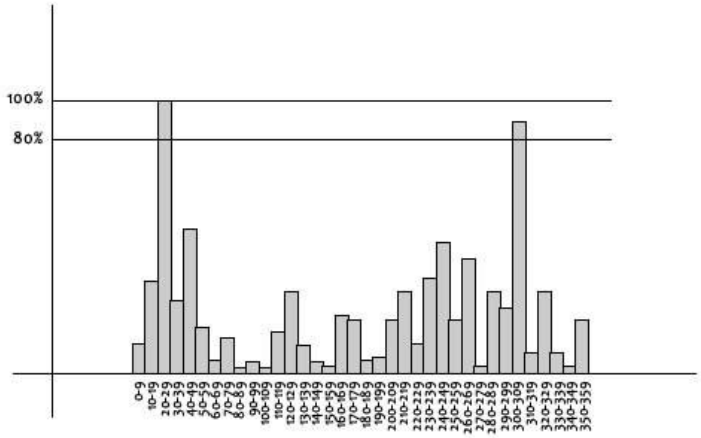

Abbildung x: Orientierungshistogramm eines Keypoints (https://medium.com/analytics-vidhya/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40)

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

ähnliche Intensität aufweisen. Deshalb vergleicht man die Intensität aller Behälter mit der des Behälters mit dem Maximalwert. Für Behälter, die mindestens 80% von dessen Intensität erreichen, wird jeweils ein weiterer Keypoint mit der jeweiligen Orientierung des Behälters erstellt. Somit kann die endgültige Menge an Keypoints auch solche enthalten, deren Lage und Skalierung identische ist, und die sich lediglich hinsichtlich der Orientierung unterscheiden (Lowe, S. 99-100, Andersson & Marquez, 2016).

3.3.4. Keypoint Description

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

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

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

Um die Invarianz bzgl. der Rotation herzustellen, wird jeweils die Orientierung des Keypoints von den ermittelten Orientierungen subtrahiert. Die Helligkeitsinvarianz wird dagegen durch eine Normalisierung gewährt, bei der man einen oberen Schwellenwert für die auftretenden Vektoren festlegt (Lowe, S. 100-103, Andersson & Marquez, 2016).

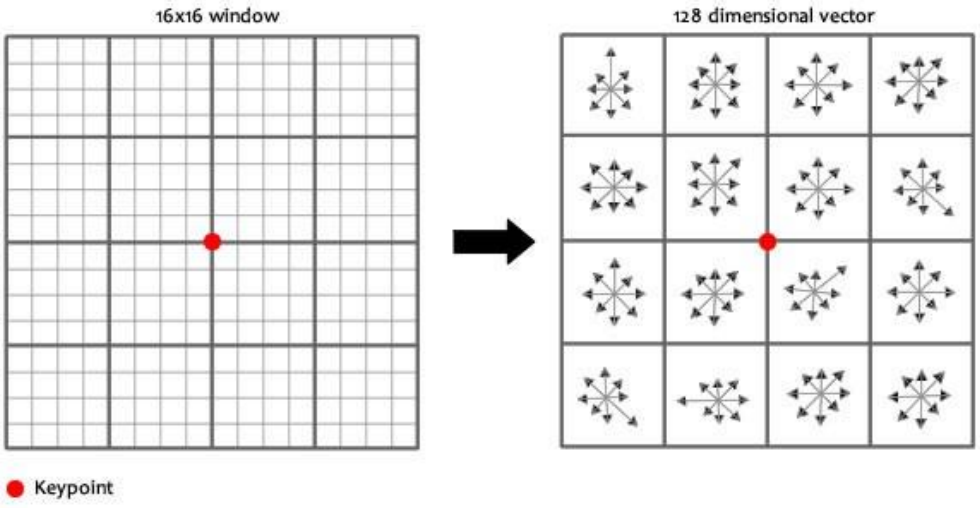

Abbildung x: Generierung eines 128-dimensionalen Vektors für einen Keypoint (https://medium.com/analytics-vidhya/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40)

3.4. Weitere Algorithmen zur Merkmalserkennung

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

3.4.1. Speeded Up Robust Features (SURF)

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

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

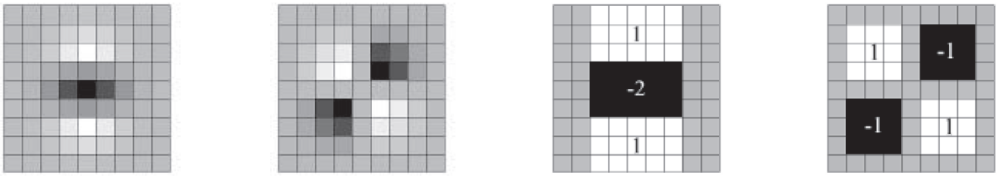

Abbildung x: Gaußsche Ableitungen (links) und deren Approximation durch Boxfilter (rechts) [Bay et al.]

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

Abbildung x: Von SURF verwendete Haar-Wavelets (links) und quadratische Regionen um Keypoints [Bay et al.]

3.4.2. Binary Robust Invariant Scalable Keypoints (BRISK)

Leutenegger et al. bauen mit ihrem Binary Robust Invariant Scalable Keypoints-Algorithmus auf SIFT und SURF auf. Auch sie sind primär an einer Erhöhung der Berechnungsgeschwindigkeit interessiert, während bezüglich der Treffergenauigkeit lediglich eine Äquivalenz zu SIFT und SURF beabsichtigt 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 et al. ist im Bereich der Eckendetektion anzusiedeln. Wird ein potenzieller Keypoint auf seine Eignung hin untersucht, erfolgt ein Vergleich mit 16 Pixeln, die alle auf einem Kreis um diesen Punkt liegen (siehe Abbildung x). Liegen auf diesem Kreis eine Mindestzahl von zusammenhängenden Pixeln, die allesamt heller oder niedriger als der Mittelpunkt sind, so kann der Punkt als geeignet gelten [Rosten et al.].

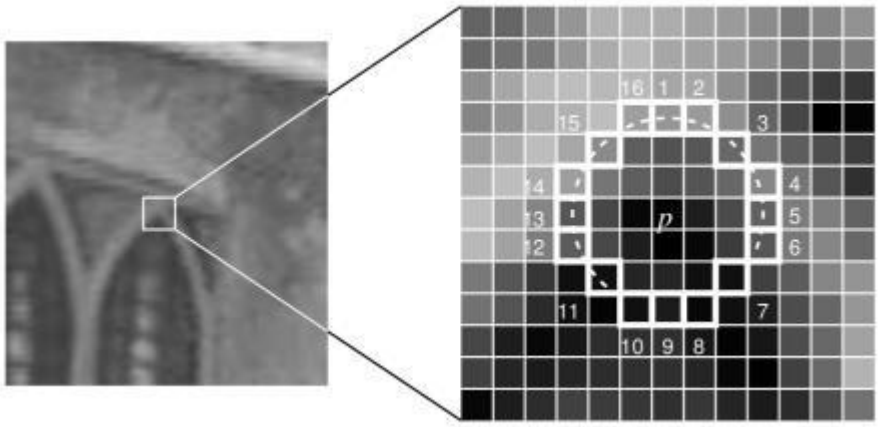

Abbildung x: Identifizierung von Keypoints durch Vergleich mit Kreispunkten [Rosten et al.]

BRISK verwendet nun eine Modifizierung 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 sogenannten Interoktaven zur Anwendung kommen. [Leutenegger et al.]

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 deutlich Zeiteinsparungen. Statt die Deskriptoren zuerst in herkömmlicher Form zu generieren und anschließend in Binärcode umzuwandeln, wird dieser bei BRIEF direkt erzeugt. Dabei wird der Keypoint mit einer festen Zahl von Punkten verglichen, die in festen Abständen voneinander auf konzentrischen Kreisen liegen, welche den Keypoint umgeben. Abbildung x zeigt exemplarisch die Anordnung der Punkte im Rahmen von BRISK. Beim Vergleich wird nun untersucht, ob entweder der Keypoint oder der Vergleichspunkt einen höheren Helligkeitswert haben. Ist der Wert des Vergleichspunkts höher, wird im Deskriptor 1 eingetragen, ansonsten 0. Insgesamt hat die binäre Zeichenkette des BRIEF64-Deskriptor 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 allein keine rotationsinvarianten Deskriptoren erzeugt. Die Invarianz muss deshalb vom verwendenden Algorithmus hergestellt werden, was dann auch bei BRISK der Fall ist. [Leutenegger et al., Calonder et al.]

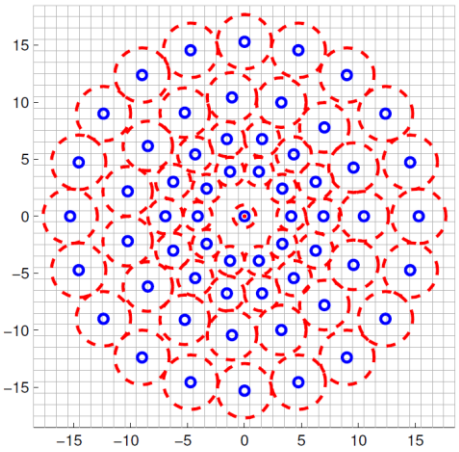

Abbildung x: Von SURF verwendete Haar-Wavelets (links) und quadratische Regionen um Keypoints [Leutenegger et al.]

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., Andersson & Marquez, 2016).

3.4.4. KAZE Features und Accelerated KAZE (AKAZE)

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

Analog zu SIFT wird der Scale Space auch hier durch die Erzeugung von Oktaven erzeugt, deren Bestandteile Bilder mit zunehmender Weichzeichnung sind. Die Berechnung der linearen Diffusionsgleichungen wird dabei nur approximiert. Dies erfolgt mittels Additive Operator Splitting (AOS), wobei der Berechnungsaufwand weiterhin erheblich ist.

Wie in Abbildung x zu sehen ist, werden Details und Rauschen bei der non-linearen Diffusion in stark unterschiedlichem Ausmaß weichgezeichnet.

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

Für die weitere Berechnung der Keypoints wird auch hier die Hesse-Matrix verwendet, während als Deskriptor eine Variation von SURF (m-SURF) gewählt wird. (Alcantarilla, Bartoli & Davison, 2012)

Aufgrund der hohen Berechnungsaufwandes veröffentlichte Alcantarilla im folgenden Jahr eine überarbeitete Version des KAZE-Algorithmus namens AKAZE. Dieser bedient sich der mathematischen Technik des Fast Explicit

Diffusion anstelle von AOS, um die Berechnung des Scale Space in deutlich kürzerer Zeit zu ermöglichen. (Alcantarilla, Nuevo & Bartoli, 2011)

3.5. Matching

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

- Zwei Bilder werden direkt miteinander verglichen, um die Gleichheit ihrer Motive anhand einer bestimmten Vergleichsmetrik zu bestimmen
- Ein Bild wird mit einer größeren Zahl von Bildern in einer Datenbank verglichen. Hierbei wird 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]

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 potentieller 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-Bäum. Die damit verbundenen deutlichen Geschwindigkeitszugewinne werden jedoch generell durch eine geringere Qualität der Ergebnisse erkauft (Minichino & Howse, 2015, „Feature Matching“, n.d.).

Wurden die zu vergleichenden Keypoints mit einem dieser Verfahren identifiziert, so 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 falsch positive Übereinstimmungen auftreten. Zwar könnte man versuchen, dies durch die Festlegung eines globalen Schwellenwerts für den euklidischen Abstand zu verhindern, doch wird dieser Ansatz der heterogenen Natur der Deskriptoren kaum gerecht. Stattdessen wendet man ein Verfahren an, das erstmals von Lowe

beschrieben wurde: Dabei betrachtet man das Verhältnis zwischen dem kleinsten und zweitkleinsten Abstand und entfernt Matches, bei denen dieses Verhältnis zu groß ist. Matches, deren Distanzverhältnis (Distance Ratio) einen bestimmten Schwellenwert – Lowe empfiehlt hier den Wert 0,8 - unterschreitet, können als Gute Matches gelten (Lowe, 2004, Dawson-Howe, 2015, „Feature Matching with FLANN“, (n.d.)).

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 Distanzverhältnis hier zu bevorzugen sind. Es gilt deshalb, für den jeweiligen Anwendungsfall zu untersuchen, welche Abstände zwischen den Keypoints bei identischen Bildern zu erwarten sind und wie diese durch die Umstände der Bildkomposition beeinflusst werden. Das genaue Vorgehen für das in dieser Arbeit besprochene Anwendungsbeispiel wird in Kapitel x detailliert beschrieben.

3.6. Performancevergleiche der Algorithmen

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

3.6.1. Gebäudeklassifikation

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

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

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

Die folgende Tabelle x gibt einige der wichtigsten Messwerte an, die von den Autoren ermittelt wurden. Die Varianten 128D und 64D von SURF verwenden jeweils unterschiede Deskriptorenlängen. Bei BRISK(1000) und ORB(1000) wurden die Algorithmen mit einer

TABELLE X. PERFORMANCE VON MERKMALSERKENNUNGS-ALGORITHMEN

Algorithmus	**Anzahl Matches Bildpaar 1 (Building)**	**Anzahl Matches Bildpaar 2 (Roofs)**	**Gesamtzeit Matching in Sekunden (Bildpaar 1)**	**Gesamtzeit Matching in Sekunden (Bildpaar 2)**
SIFT	384	423	0,5186	0,7186
SURF(128D)	319	171	0,8940	0,6129
SURF(64D)				
BRISK	481	436	0,2390	0,5905
BRISK(1000)				

ORB	854	498	0,2086	0,4899
ORB(1000)				
KAZE	465	172	0,4924	0,5162
AKAZE	475	175	0,1772	0,1839

Tab. 2. Übersicht über Messwerte, die beim Performancevergleich von Merkmalserkennungs-Algorithmen ermittelt wurden. [Tareen & Saleem]

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, dass SIFT die höchste Treffergenauigkeit aufweist, obwohl die absolute Anzahl gefundener Matches dies auf den ersten Blick nicht nahelegen würde.

3.6.2. Invarianz-Tests

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

TABELLE X. ANTEIL KORREKTER MATCHES FÜR INVARIANZTYPEN

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

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

Es zeigt sich, dass keiner der Algorithmen den anderen in jeder Hinsicht als überlegen gelten kann. Hingegen erscheint die Fähigkeit des ORB-Algorithmus, korrekte Matches zu ermitteln, generell geringer ausgeprägt zu sein. Für ORB spricht hingegen, dass dieser im Rahmen der Tests im Mittel mehr als zehn Mal

schneller war als SIFT und AKAZE und mehr als hundert Mal so schnell wie KAZE (Andersson & Marquez, 2016).

4. Forschungsstand: Technisch

4.1. OpenCV

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

Üblicherweise wird jedoch zu Zwecken der Merkmalserkennung auf bestehende Softwarebibliotheken zurückgegriffen, wobei in der Regel OpenCV zum Einsatz kommt. Diese quelloffene Bibliothek kann für vielfältige Anwendungszwecke im Bereich der Computer Vision und Bildbearbeitung verwendet werden.

OpenCV ist auf zahlreiche Plattformen einsetzbar, etwa auf Windows, Linux, macOS, Android und iOS. Es kann darüber hinaus mit Programmiersprachen wie Python, Java und C++ genutzt werden. Mit der Bibliothek OpenCV.js steht auch einer Portierung für JavaScript zur Verfügung, die jedoch nur über eine eingeschränkte Zahl von Funktionen verfügt. Speziell die Algorithmen zur Merkmalserkennung stehen hierfür nur eingeschränkt zur Verfügung, jedoch sind BRISK und ORB bereits nutzbar [https://github.com/ucisysarch/opencvjs]. Alternativ dazu bietet auch die Bibliothek jsfeat die Möglichkeit, FAST und ORB in einer JavaScript-Anwendung zu nutzen. [opencv.js Seite, https://inspirit.github.io/jsfeat/]

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

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

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

[Tazehkandi]

5. Forschungsstand: Gebäudeidentifikation

5.1. Bisherige Ansätze

Hutchings und Mayol-Cuevas: Ähnlich wie hier, aber es sind Bilder mit unterschiedlicher Skalierung gespeichert. Geeignetstes anhand GPS-Position ausgewählt. Mehr

Trinh et al.
Weitere Forschung mit Gebäuden! Gebäuderkennungs-Apps siehe Literaturverzeichnis.

6. Lösungsansatz

7. Umsetzung

BIdent Building Identification ist eine Progressive Web App, mit der BenutzerInnen plattformübergreifend photographische Aufnahmen von historischen Gebäuden und deren Bauteilen machen und diese automatisch identifizieren lassen können.

7.1. Architektur

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

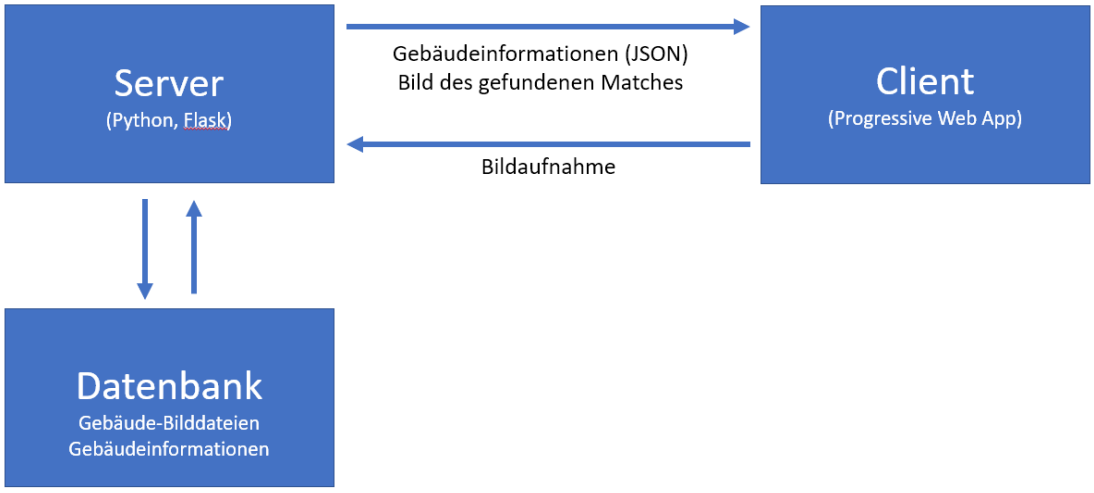

Abbildung x: Architektur von *BIdent Building Identification*.

7.2. 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 gespeichert. Um trotzdem die Offline-Fähigkeit zu gewährleisten, wird deshalb ein *Service Worker* für das Caching der für die Ausführung notwendigen Dateien eingesetzt.

Screenshot

Die graphische Benutzeroberfläche des Clients wurde mit HTML, CSS und JavaScript umgesetzt, wobei die Prinzipien des *Responsive Design* zur Anwendung kamen. Auf der Hauptseite wird lediglich eine Überschrift und ein Kamera-Button angezeigt, die beide den Input der Gerätekamera überlagern. Nach dem Betätigen des Kamera-Buttons wird nach einer gewissen Wartezeit eine Seite mit Details über das Gebäude bzw. Bauteile angezeigt.

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.

Upload-Seite zu Client oder Server?

7.3. Server

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

MySQL-DB-Schemabild

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

Nun erfolgt das Matching.

Matching (Algorithmus wählen – Strategy Pattern)

Hamming oder Flann

Keine Homographie

Wie Sicherheit bestiimmen

Das Ergebnis wird nun in ein JSON-Objekt umgewandelt, das neben einer Id auch den Namen und eine Beschreibung des Objekts enthält. Außerdem wird noch ein Prozentwert für die Sicherheit der korrekten Identifizierung übermittelt. Dieses JSON-Objekt wird abschließend als Response auf dessen ursprüngliche Anfrage an den Client zurückgegeben.

Daneben bietet der Client noch einige weitere Funktionen, darunter eine simple CRUD-Oberfläche für die Verwaltung der Gebäude- bzw. Bilddateien und eine Möglichkeit, Bilddateien anhand ihrer Id bereitzustellen.

8. Evaluierung

Einer der wichtigsten Aspekte bei der Umsetzung der Anwendung ist die Frage, ab wann zwei Bilder als Repräsentation des gleichen Objekts gelten können. Als Ergebnis der Bildvergleichs liefern sämtliche Merkmalserkennungs-Algorithmen lediglich eine Menge von Matches. Aus diesen können mittels x diejenigen Matches entnommen werden, welche mit hoher Wahrscheinlichkeit als korrekt gelten können (siehe Kapitel x). Doch aus der Anzahl dieser guten Matches allein lässt sich noch keine Aussage über die Richtigkeit der Bildklassifikation liefern.

Hierfür ist es stattdessen erforderlich, für den jeweiligen Anwendungskontext zu ermitteln, welche Anzahl an guten Matches erwartet werden kann. Insbesondere sind hier die folgenden drei Situationen zu prüfen, mit denen sich die folgenden Kapitel beschäftigen werden:

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

In Kapitel 8.3. jeweils Invarianz-Testergebnisse mit untersten 90% etc. vergleichen.

8.1. Matches bei identischen Bildern

Fehlt. Einfach einige hundert Bilder mit sich selbst vergleichen, für jeden Algorithmus.

8.2. Matches bei unterschiedlichen Motiven

Aus den vorhergehenden Invarianz-Tests lässt sich bereits mit einiger Sicherheit ableiten, wie viel gute Matches zu erwarten sind, wenn das Motiv auf beiden

Bildern übereinstimmt. Für eine deutlich sicherere Einordnung ist es jedoch ebenso erforderlich, einen Vergleichswert für die Anzahl guter Matches bei unterschiedlichen Bildmotiven zu ermitteln.

Als Grundlage wurde hierfür das *Oxford Buildings Dataset* gewählt. Die 5062 enthaltenen Bilder wurden jeweils durch eine Suche auf der Plattform *Flickr* nach den Namen bestimmter Gebäude in der Stadt Oxford ermittelt. Aus diesem Grund enthalten viele der Bilder nur Innenaufnahmen des Gebäudes, oder zeigen ein anderes Motiv, das lediglich von diesem Gebäude aus aufgenommen wurde. Hieraus ergibt sich jedoch der Vorteil, dass auch für das Matching mit gänzlich unähnlichen Objekten nützliche Messergebnisse erzielt werden können.

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

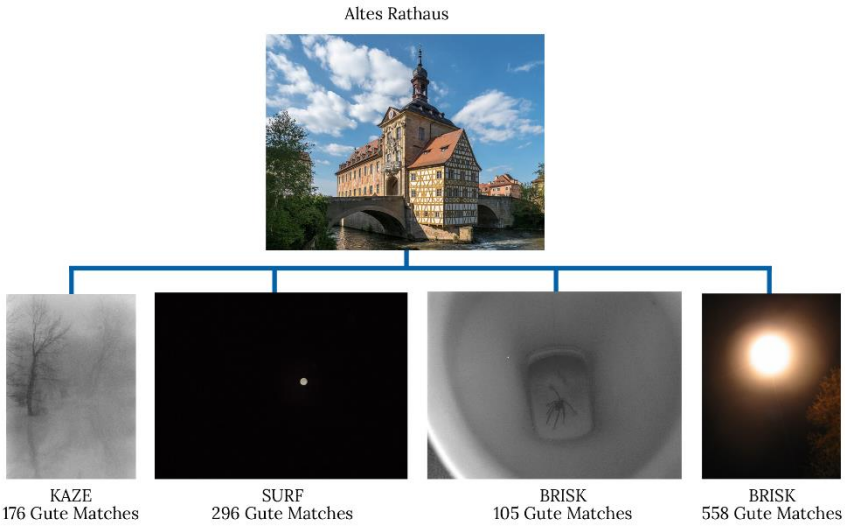

Abbildung x: Ausreißer beim Bildvergleich des Alten Rathauses in Bamberg mit Aufnahmen des Oxford Buildings Dataset. Neue Werte!

Tabelle X sowie das zugehörige Box-Plot-Diagramm (siehe Abbildung x) liefern eine genauere Übersicht über die Verteilung der Werte. Hierbei fällt auf, dass der BRISK-Algorithmus eine besonders große Spannweite aufweist. Auch bei SURF,

KAZE und AKAZE finden sich mehrere Ausreißer, die von den üblichen Werten erheblich abweichen.

Zu Zwecken der Evaluierung ist es sicherlich möglich, derartige Bilder nicht in die Berechnung einfließen zu lassen. Es ist jedoch zu erwarten, dass bei der praktischen Anwendung der Identifikations-Applikation auch Aufnahmen getätigt werden, die ebenfalls zu ähnlichen Matching-Ergebnissen führen können. Die Wahl eines Algorithmus, der vergleichsweise robust gegenüber solchen Ausreißern ist, erscheint deshalb geeignet, die Ergebnissicherheit zu erhöhen.

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

Name	**Minimum**	**Unterer Quartilswert**	**Median**	**Oberer Quartilswert**	**Maximum**
SIFT	16	44	50	57	109
SURF	20	79	88	96	595
BRISK	6	25	30	36	1392
ORB	0	5	7	10	48
KAZE	0	25	30	36	543
AKAZE	3	19	22	26	327

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

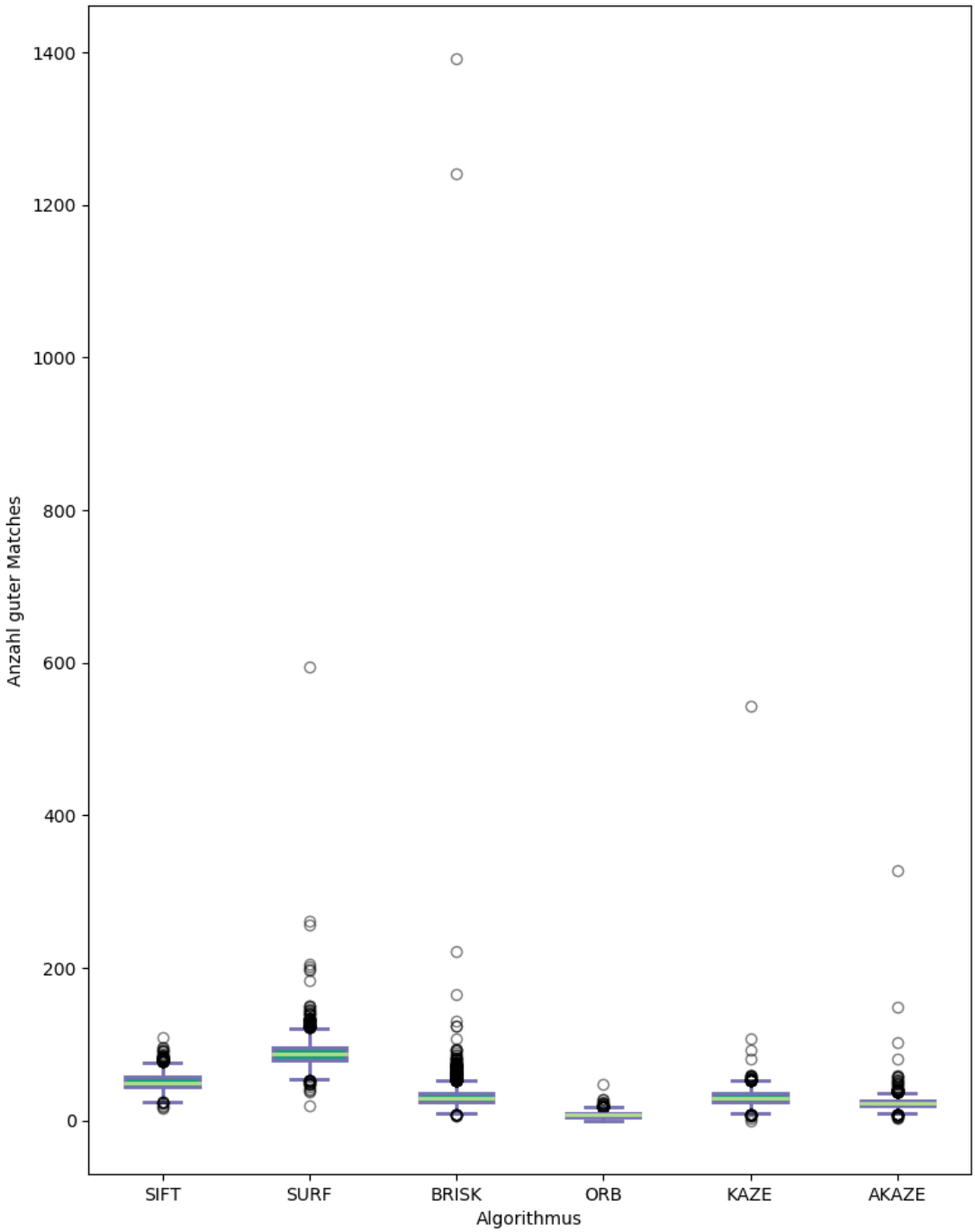

Abbildung x: Boxplots mit Anzahl der guten Matches für sämtliche Algorithmen beim Vergleich mit allen Bildern des *Oxford Buildings Datasets*.

Weiterer Vergleich aller Nbg/Ba-Bilder untereinander. Nur reguläre Frontansicht.

8.3. Matches bei Varianz der Aufnahmebedingungen

Für den Bildvergleich ist es unabdingbar, sich für einen der Merkmalserkennungs-Algorithmen, die in Kapitel x vorgestellt wurden, zu entscheiden. Hierfür sollen im folgenden exemplarische Bildvergleiche historische Gebäude und Bauteile aus der Stadt Nürnberg die Performance der Algorithmen im Rahmen des Anwendungszwecks dieser Arbeit ermitteln.

Als untersuchte Parameter wurden zum einen die Anzahl guter Matches (vgl. x) und zum anderen die für deren Berechnung benötigte Zeit gewählt, da sowohl die Qualität der Bilderkennung als auch die Geschwindigkeit der Anwendung als wichtigste Faktoren für die Benutzerzufriedenheit anzusehen sind.

8.3.1. 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 bei der Bewertung berücksichtigt werden muss. In Abbildung x sind die verwendeten Bilder zu sehen.

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

Tabelle x zeigt die Anzahl guter Matches für die Tag-/Nacht-Bildvergleiche bei den sechs Objekten. SURF erkennt dabei in allen Fällen die meisten guten Matches während ORB stets die wenigsten aufweist.

TABELLE X. ANZAHL GUTER MATCHES – TAG/NACHT

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

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

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

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

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

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

8.3.2. Okklusion

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

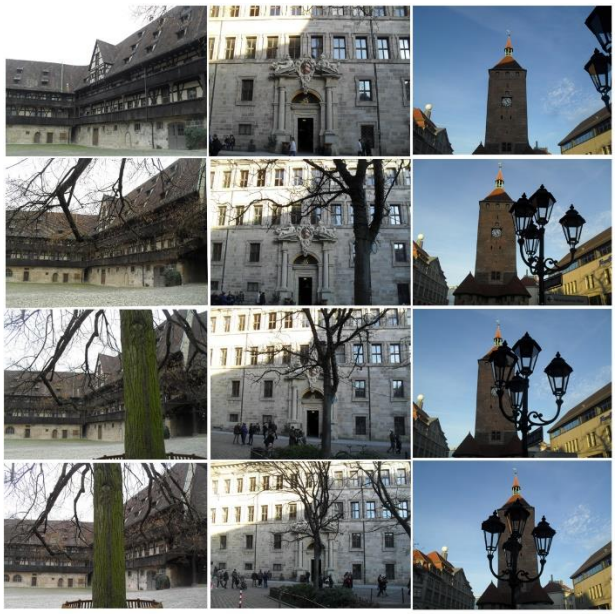

Abbildung x: Gebäude aus Bamberg und Nürnberg als Basis für die Invarianz-Tests hinsichtlich der Okklusion. Oben ist jeweils das unbedeckte Gebäude zu sehen.

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

Generell ist zu beobachten, dass die Match-Anzahl bei den stärker verdeckten Objekten geringer ausfällt. Bei den Bildern der Alten Hofhaltung in Bamberg und des Weißen Turms in Nürnberg fällt jedoch auf, dass die Positionierung des verdeckenden Objekts in der Bildmitte zu einer höheren Zahl guter Matches führt als eine weniger zentrale Position. Auch hier liefert ORB nur eine deutlich geringere Anzahl guter Matches als die anderen Algorithmen während SURF stets die meisten guten Matches erkennt.

TABELLE X. ANZAHL GUTER MATCHES – OKKLUSION

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

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

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

TABELLE X. GESAMTDAUER FÜR BERECHNUNG GUTER MATCHES – OKKLUSION

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

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

8.3.3. 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.
Nicht genau.
Schatten.

Für jedes Gebäude wird das zentrale Bild (in Abbildung x mittig und rot markiert) mit den anderen Bildern des gleichen Objekts verglichen. In den folgenden Tabellen werden diese Vergleichsbilder jeweils von links nach rechts mit den Nummern 1-8 identifiziert.

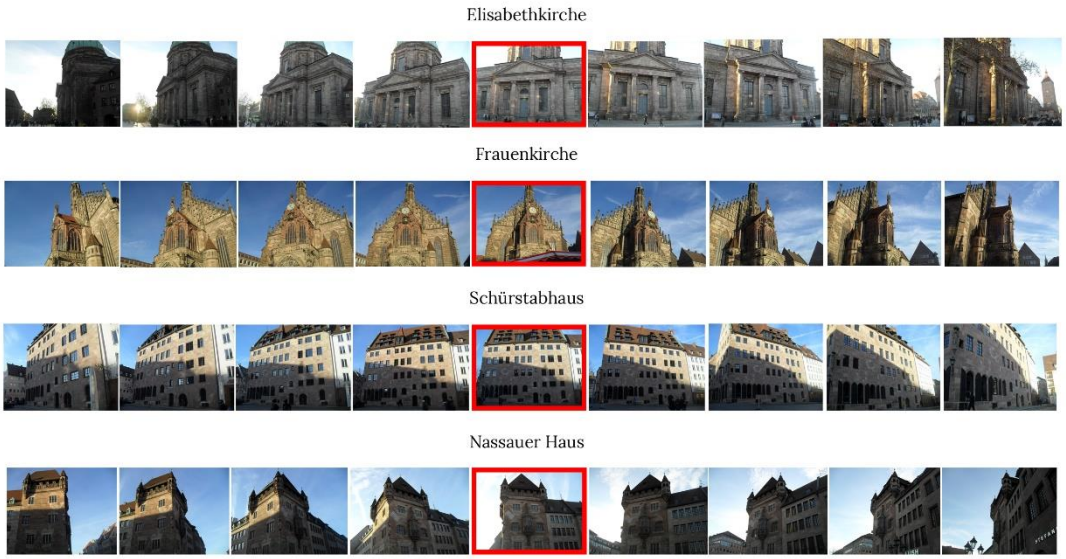

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der horizontalen Perspektive.

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

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

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

Generell lässt sich eine deutliche Abnahme guter Matches beobachten, je stärker der Aufnahmewinkel sich von der Frontalansicht unterscheidet.

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

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

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

Text

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

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

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

Text

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

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

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

Text

8.3.4. 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.

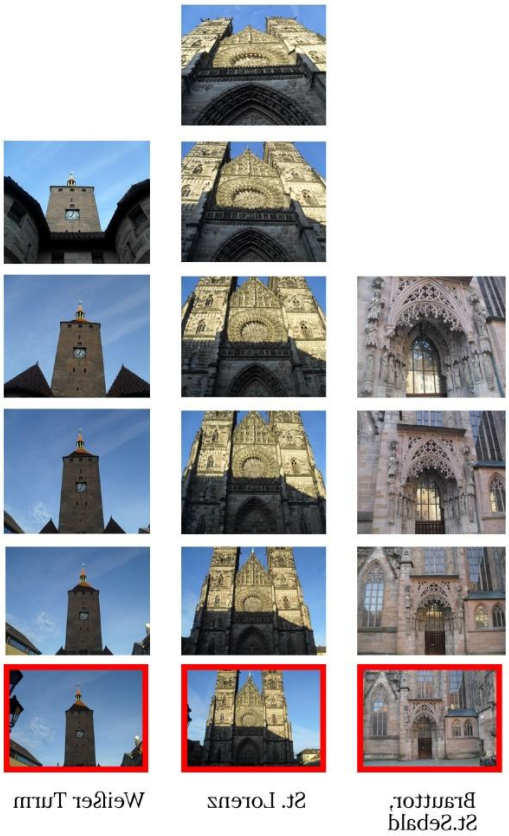

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der vertikalen Perspektive.

Text

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

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

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

Text

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

Name	#1	#2	#3	#4	#5	Gesamtdauer

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

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

Text

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

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

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

Text

8.3.5. 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 45 Grad vorgenommen wurde (siehe Abbildung X). Der Vergleich erfolgte jeweils zwischen dem Originalbild und einer der acht Rotationsbilder.

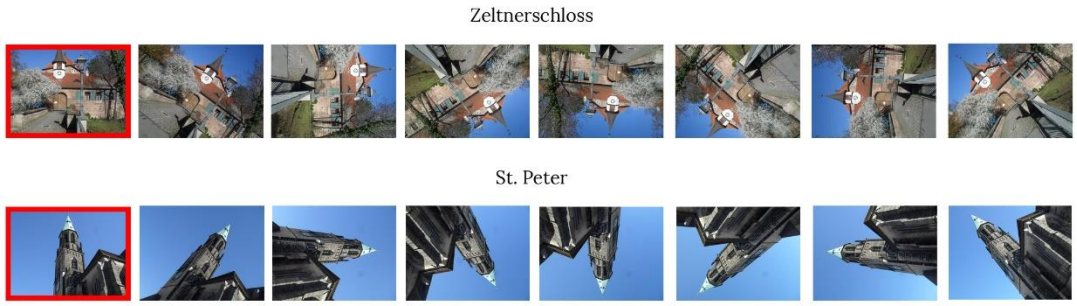

Abbildung x: Gebäude aus Nürnberg als Basis für die Invarianz-Tests hinsichtlich der Rotation.

TABELLE X. PERFORMANCE FÜR ROTATIONSVARIANZ – ZELTNERSCHLOSS

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

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

Text

TABELLE X. PERFORMANCE FÜR ROTATIONSVARIANZ – ST. PETER

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

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

Text

8.3.6. Skalierung

Die Skalierung kann bei Aufnahmen von (immobilen) Gebäuden auf zweierlei Art variiert werden: Durch das Erstellen von Bildern in unterschiedlicher Entfernung sowie über die Abwandelung des Zoomfaktors der Kamera. Im Zuge der Evaluierung wurden beide Herangehensweisen verwendet. Abbildung x zeigt die verwendeten Aufnahmen, wobei beim Objekt Zeltnerschloss die Erstere zum

Einsatz kam, die Letztere hingegen beim Bauteil von St. Peter. In beiden Fällen wurde jeweils die Aufnahme mit der größten Nähe zum Objekt bzw. dem größten Zoomfaktor mit den anderen Aufnahmen verglichen.

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

TABELLE X. PERFORMANCE FÜR SKALIERUNGSVARIANZ – ZELTNERSCHLOSS

Name	**#1**	**#2**	**#3**	**#4**	**Gesamtdauer**
SIFT	1913	1327	1013	1056	20,56 s
SURF	3309	2404	1989	1865	19,88 s
BRISK	1317	885	600	511	229,61 s
ORB	11	11	13	9	0,64 s
KAZE	1261	746	653	550	29,99 s
AKAZE	1043	708	510	435	18,93 s

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

Text

TABELLE X. PERFORMANCE FÜR SKALIERUNGSVARIANZ – ST. PETER

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

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

Text

8.4. Fazit

Aus den vorhergehenden Invarianz-Tests lässt sich bereits mit einiger Sicherheit ableiten, wie

Geschwindigkeit / Matches abhängig von Architekturstil ?

9. Diskussion

Zusammenfassung der Vorteile und Wirksamkeit der Lösung. Auch Nachteile erwähnen.

Verbesserungen der App selbst:

- Deskriptoren in DB speichern, statt jedes Mal aus Bild selbst berechnen zu müssen. Aber nur für manche Algorithmen möglich(?).
- Verwenden von Machine Learning
- Stitching der Bilder aus verschiedenen Perspektiven, um Bildzahl in DB zu verringern
- Sicherheitsberechnung nach statistischen Methoden.

Andere Möglichkeiten zur Orientierung:

- Nutzer können selbst Bauteile durch Fotos hinzufügen statt diese selbst in Karte (z.B. OSM) hinzufügen zu müssen.
- Identifizierung durch Position, Richtung und Visibility-Algorithmus.

10. Anhang

11.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)

„Feature Matching with FLANN“ (n.d.) https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_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.

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.

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“. 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.

Tazehkandi, A.A. „Hands-On Algorithms for Computer Vision.“ Packt Publishing, Birmingham, Vereinigtes Königreich, 2018.

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.

Ort, Datum	Unterschrift