HOCHSCHULE
MITTWEIDA
University of Applied Sciences

MASTERARBEIT

Frau
B.Sc. Lätizia Miedrich

Prototypische Implementierung und Evaluierung von Detektoren für digitale Bildmanipulationen

2020

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERARBEIT

Prototypische Implementierung und Evaluierung von Detektoren für digitale Bildmanipulationen

Autorin:
B.Sc. Lätizia Miedrich

Studiengang:
Cybercrime/Cybersecurity

Seminargruppe:
CY18wC-M

Erstprüfer:
Prof. Dr. rer.nat.habil. Thomas Haenselmann

Zweitprüfer:
M.Sc. Maik Benndorf

Mittweida, August 2020

Bibliografische Angaben

Miedrich, B.Sc. Lätizia: Prototypische Implementierung und Evaluierung von Detektoren für digitale Bildmanipulationen, 105 Seiten, 47 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Masterarbeit, 2020

Referat

Die folgende Arbeit behandelt die Methoden digitaler Bildmanipulationen sowie die Erkennung solcher Manipulationen anhand etablierter Detektionsverfahren. Das Hauptaugenmerk liegt dabei auf der Planung und Implementierung einer Software zur automatisierten Detektion duplizierter Regionen innerhalb eines digitalen Bildes. Anschließend an diese Implementierung folgt eine detaillierte Auswertung der Detektionsergebnisse sowie eine Bewertung der Qualität der Software gegenüber bestehenden Verfahren.

Englischer Titel der Arbeit

Prototypic Implementation and Evaluation of Detectors for Digital Image Manipulations

Abstract

The following opus addresses the methods used for manipulating digital images as well as the detection of such manipulations by established detection algorithms. The focus is thereby on planning and implementing a software for automated detection of duplicated regions within a digital image. This implementation is subsequently followed by a detailed evaluation of the detection results and an assessment of the software's quality compared to existing practices.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis und Glossar

AKAZE	Accelerated KAZE, ein Algorithmus zur Extraktion der Keypoints eines Bildes unter Einbeziehung uniformer Bildregionen, schneller als KAZE
Copy-Move	eine Methode zur Manipulation digitaler Bilder, bei welcher Regionen innerhalb eines Bildes dupliziert werden
DCT	Diskrete Kosinustransformation, eine Transformation von Signalwerten vom Orts- in den Frequenzraum unter Verwendung von Kosinusfunktionen
DWT	Diskrete Wavelettransformation, eine Transformation von Signalwerten vom Orts- in den Frequenzraum unter Verwendung von Waveletfunktionen
Image Resampling	eine Methode zur Manipulation digitaler Bilder, bei welcher geometrische Transformationen (Rotation, Skalierung etc.) auf ein Bild angewandt werden
Image Retouching	eine Methode zur Manipulation digitaler Bilder, bei welcher durch u.a. durch Farb-, Helligkeits- oder Kontraständerungen die visuelle Erscheinung eines Bildes verändert wird
Image Splicing	eine Methode zur Manipulation digitaler Bilder, bei welcher Regionen aus einem Bild ausgeschnitten und in ein zweites Bild eingefügt werden
JPEG	Joint Photographic Experts Group, ein verlustfreies Kompressionsverfahren für digitale Bilder
KAZE	das japanische Wort für Wind, ein Algorithmus zur Extraktion der Keypoints eines Bildes unter Einbeziehung uniformer Bildregionen
Keypoints	repräsentative Bildpunkte, welche aus Bildregionen mit hoher Entropie extrahiert werden
RANSAC	Random Sample Consensus, ein Resampling-Algorithmus zur Entfernung von Ausreißern aus einer Liste von Datenpunkten
RGB	Rot-Grün-Blau, ein häufig für digitale Bilder verwendetes Farbmodell
SIFT	Scale-Invariant Feature Transform, ein Algorithmus zur Extraktion der Keypoints eines Bildes
SLIC	Simple Linear Iterative Clustering, ein Algorithmus zur Berechnung von Superpixelregionen eines Bildes
Superpixel	irreguläre Bildregionen, welche semantisch zusammengehörige Bildabschnitte zusammenfassen
SURF	Speeded-Up Robust Features, ein Algorithmus zur Extraktion der Keypoints eines Bildes, schneller als SIFT
SWT	Stationäre Wavelettransformation, eine translations-invariante Version der DWT
YC_bC_r	ein Farbmodell für digitale Bilder mit Einbeziehung der Luminanz- und Chrominanzunterschiede

V. Vorwort

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank den Herren Prof. Dr. rer. nat. habil. Thomas Haenselmann und Maik Benndorf (M.Sc.), die mich vor und während der Ausarbeitung meiner Masterarbeit betreut haben und immer für Fragen und Anregungen offenstanden. Besonderer Dank geht hierbei an Herrn Benndorf, der mich bei der Themensuche unterstützt und mir hilfreiche Tipps für die Organisation und Strukturierung meiner Arbeit gegeben hat.

Zuletzt danke ich meiner Familie und meinen Freunden, die mich während der Bearbeitungsphase meiner Arbeit unterstützt, mir Rückhalt geboten und mir ein bisschen Abwechslung zum Arbeitsalltag gebracht haben.

Anmerkungen

Da ein Großteil der Literatur im Bereich digitaler Bildmanipulationen in englischer Sprache verfasst wurde, haben sich in diesem Forschungsgebiet englische Fachbegriffe etabliert, es existieren zum derzeitigen Stand keine standardmäßig gebrauchten deutschen Pendants. Aus diesem Grund werden die englischen Fachbegriffe im Folgenden einmalig übersetzt, dann jedoch als englische Begriffe weiterverwendet.

Lätizia Miedrich

Mittweida, 25.08.2020

1 Einleitung

Digitale Bilder treten in der heutigen Zeit in allen Bereichen des täglichen Lebens auf. Ob in Modezeitschriften, im Wahlkampf politischer Parteien oder als Unterhaltung im Internet, digitale Bilder begleiten auf unterschiedliche Weisen unseren Alltag. Allein auf Facebook, einem der am häufigsten verwendeten Social-Media-Kanälen [vgl. Cle20], werden täglich rund 350 Millionen Bilder hochgeladen, pro Sekunde rund 4.000 Bilder [vgl. Asl20].

Vor Beginn der Digitalisierung galten Bilder, im Gegensatz zu Texten und mündlichen Erzählungen, als Quelle authentischer Informationen. Bilder zu fälschen erforderte damals eine fachliche Expertise, heutzutage ermöglichen es Bildbearbeitungsprogramme wie Adobe Photoshop oder Gimp auch Nutzern ohne Fachwissen, Bilder nach Belieben zu gestalten und anzupassen. Während das Vertrauen in Bilder in der Vergangenheit also beständig blieb, beginnen neue Technologien dieses Vertrauen zu untergraben [vgl. Far09, S. 16]. Die Frage nach einer beweissicheren Authentifikation digitaler Bilder ist daher aktueller denn je und erfordert eine professionelle Untersuchung.

Motivation

Der steigende Vertrauensverlust in die Authentizität digitaler Bilder wirft die Frage auf, wie man in der heutigen Zeit noch zwischen originalen und manipulierten Bildern unterscheiden kann. Bildmanipulationen sind inzwischen so weit entwickelt, dass ein einfaches in Augenschein Nehmen als Grundlage zur Feststellung der Authentizität von Bildern nicht mehr ausreicht.

Eine Studie mit 707 Teilnehmern aus dem Jahr 2017 hat gezeigt, dass es Menschen im Allgemeinen schwer fällt, digitale Bildmanipulationen als solche zu erkennen und zu lokalisieren [vgl. NWW17, S. 4] . Für die Studie wurden 10 digitale Bilder von Menschen in realen Szenen aufgenommen. Anschließend wurden die Bilder auf fünf verschiedene Arten manipuliert, u.a. durch das Hinzufügen von Objekten oder durch die Verzerrung geometrischer Maße im Bild. Das Ergebnis der Manipulationen war ein Datensatz aus 40 Bildern, davon 10 originale und 30 manipulierte Bilder. Zur Durchführung der Studie untersuchten die Probanden jeweils 10 zufällig gewählte Bilder des Datensatzes und bewerteten die Authentizität der Bilder und ob sie eine mögliche Manipulation genau lokalisieren konnten. Ziel der Studie war es, die Genauigkeit der Klassifikation und die Fähigkeit der Lokalisation der Probanden zu testen. Die Ergebnisse zeigten, dass etwa 66 % der Bilder korrekt klassifiziert wurden, lediglich 45 % der Manipulationen im Bild konnten korrekt lokalisiert werden [vgl. NWW17, S. 6].

Eine weitere Studie mit 250 Probanden erforschte die menschliche Fähigkeit, zwischen Fotografien menschlicher Gesichter und computergenerierten Gesichtern zu unterschei-

den. Für die Studie wurde ein Datensatz aus 30 computergenerierten sowie 30 fotografischen Bildern erstellt, sodass je ein computergeneriertes Bild und eine Fotografie einander beinahe identische menschliche Gesichter zeigen [vgl. HF15, S. 7]. Für ein Experiment der Studie wurden den Probanden alle 60 Bilder des Datensatzes in willkürlicher Auflösung und Reihenfolge gezeigt. Die Probanden wurden nach jedem gezeigten Bild aufgefordert, zu beurteilen, ob es sich bei dem Bild um eine Fotografie oder ein computergeneriertes Bild handelt sowie ob das Bild eine männliche oder weibliche Person zeigt [vgl. HF15, S. 11]. Das Ergebnis der Studie zeigt, dass die Auflösung die Erkennung fotografischer Bilder kaum beeinflusste, die Genauigkeit der Probanden lag hier bei etwa 90 bis 92% [vgl. HF15, S. 13]. Die Genauigkeit bei der Detektion computergenerierter Bilder schwankte jedoch, je nach Auflösung, zwischen ca. 52 bis 66% [vgl. HF15, S. 14]. Im Durchschnitt wurden fotografische Bilder demnach leichter als solche erkannt als computergenerierte Bilder.

Studien wie diese beiden zeigen, dass es für den Menschen immer schwieriger wird, ohne Hilfe von speziellen Programmen und Algorithmen Bildmanipulationen als solche zu erkennen. Es gilt also, die Erkennung von Bildmanipulationen durch die Entwicklung neuer Vorgehensweisen zu verbessern und zu vereinfachen.

Zielsetzung

Ziel bisheriger Forschungen war es, effizientere Methoden zur Detektion von Bildmanipulationen zu entwickeln und diese anhand bewährter oder selbst erstellter Datensätze zu evaluieren. Ein Großteil der veröffentlichten Arbeiten konzentriert sich dabei jedoch auf eine oder wenige verschiedene Detektionsmethoden wie das Kopieren und Einfügen von Objekten sowie die Skalierung oder Kompression eines Bildes [vgl. Fon+13, S. 2]. Weiterhin enthalten nicht alle Publikationen neben der Detektion von Bildmanipulationen die Möglichkeit, gefundene Manipulationen auch genau zu lokalisieren. Für die gerichtsfeste Analyse von Bildern ist jedoch ein Werkzeug gefragt, mit welchem digitale Bilder auf verschiedene Arten von Manipulationen untersucht werden können [vgl. Fon+13, S. 2].

Diese Arbeit soll die verschiedenen Arten von digitalen Bildmanipulationen in einer aktuellen und detaillierten Gliederung zusammenfassen sowie deren Funktionsweise und Anwendungsgebiete verdeutlichen. Weiterhin sollen die bisherigen Entwicklungen im Bereich der Manipulationsdetektion digitaler Bilder zusammengefasst und miteinander verglichen werden. Nach Ermittlung der effizientesten Detektionsverfahren soll eine Softwareanwendung implementiert werden, welche die Vorteile der einzelnen Verfahren kombiniert, um so verschiedene Arten von Bildmanipulationen mit höherer Genauigkeit detektieren und lokalisieren zu können. Ziel ist ein Programm, welches Bilder nach ihrer Authentizität binär klassifiziert, manipulierte Bildregionen farbig markiert und damit die Manipulationserkennung für digitale Bilder vereinfacht.

Kapitelübersicht

Diese Arbeit ist in 6 thematische Bereiche unterteilt. Im *ersten Abschnitt* (Kapitel 2) werden Begriffe der Bildverarbeitung und Statistik erklärt, welche Voraussetzung für ein grundlegendes Verständnis der Arbeit darstellen. Weiterhin werden verschiedene Definitionen und Synonyme des Begriffs ‘Bildmanipulation voneinander abgegrenzt.

Der *zweite Abschnitt* (Kapitel 3) umfasst eine überblicksweise Kategorisierung und Erläuterung der verschiedenen Methoden zur Manipulation digitaler Bilder. Dabei wird u.a. auf die Besonderheiten der einzelnen Methoden sowie auf einige Anwendungsbeispiele eingegangen.

Im darauf *folgenden Abschnitt* (Kapitel 4) der Arbeit werden die Kategorisierung sowie die Voraussetzungen und Anwendungsbereiche bestehender Detektionsverfahren für digitale Bildmanipulationen dargelegt. Anschließend wird der aktuelle Forschungsstand des Fachgebietes erläutert.

Kapitel 5 beschreibt die Durchführung bei der Implementierung einer Software zur Detektion von digitalen Bildmanipulationen. Hierbei wird begründet, welche Detektionsverfahren implementiert werden, welche Anforderungen an die Software gestellt werden und wie Konzept sowie Design aufgebaut sind. Anschließend an die Implementierung wird in diesem Abschnitt die Auswahl geeigneter Datensätze sowie der Testdurchlauf der Softwareanwendung erläutert.

Der *nächste Abschnitt* (Kapitel 6) der Arbeit behandelt die Evaluation der Detektions- und Lokalisationsergebnisse der implementierten Software im Vergleich zu bestehenden Detektionsverfahren. Es folgt eine ausführliche Bewertung der Eignung der Software für die Verwendung im Bereich der Bildmanipulationserkennung.

Als *letzter Abschnitt* folgt in Kapitel 7 eine Zusammenfassung der Ergebnisse dieser Arbeit sowie ein Ausblick auf zukünftig wichtige Forschungen und Denkanstöße.

2 Grundlagen

Ziel dieser Arbeit ist es, bestehende Verfahren zur Detektion manipulierter Bilder zu vergleichen und darauf aufbauend eine Software zu implementieren, welche die Vorteile einzelner Verfahren kombiniert, um so die Genauigkeit bei der Klassifizierung digitaler Bildmanipulationen zu erhöhen. Um besser auf die Funktionsweise dieser Verfahren eingehen zu können, werden im nun folgenden Kapitel einige wichtige Grundlagen der Bildverarbeitung und Statistik erläutert. Anschließend wird der Begriff ‚Bildmanipulation' genauer definiert und von ähnlichen Begriffen abgegrenzt.

2.1 Grundlagen der digitalen Bildverarbeitung

Bei den folgenden Definitionen aus dem Bereich der digitalen Bildverarbeitung handelt es sich insbesondere um Transformationen und mathematische Algorithmen, mit deren Hilfe digitale Bilder auf unterschiedliche Weise dargestellt oder weiterverarbeitet werden können. Auch die Extraktion von Bildmerkmalen sowie der Vergleich einzelner Bildregionen untereinander ist durch die Anwendung solcher Algorithmen möglich.

Farbraumtransformationen

Die Darstellung digitaler Farbbilder, erfolgt i.d.R. durch die Verwendung des RGB-Farbmodells [vgl. BB15, S. 309]. Bei diesem handelt es sich um ein additives Farbmodell, welches Farben durch die Kombination von Rot-, Grün- und Blauwerten darstellt. Um neben Farbtonänderungen jedoch auch Änderungen der Farbsättigung und Helligkeit zu veranschaulichen, muss in der Bildverarbeitung häufig auf andere Farbmodelle wie das HSV-Farbschema zurückgegriffen werden [vgl. BB15, S. 322]. Da digitale Bilder häufig im RGB-Format vorliegen, muss ein Bild für die Weiterverarbeitung daher oft in einen anderen Farbraum transformiert werden.

Bekannte Detektionsverfahren für digitale Bildmanipulationen verwenden zur Extraktion von Bildmerkmalen und weiteren mathematischen Berechnungen häufig das YC_bC_r-Farbmodell. Bei diesem setzt sich das digitale Bild wie beim RGB-Farbmodell aus drei Kanälen zusammen. Der Y-Kanal speichert die Luminanz, sprich die Helligkeitsunterschiede, des Bildes, während der C_b- und der C_r-Kanal die Chrominanz, die Farbdifferenzen der Blau- bzw. Rot-Komponente von der Luminanz, enthalten [vgl. BB15, S. 337].

Die Transformation eines Bildes vom RGB- in den YC_bC_r-Farbraum lässt sich anhand Formel 2.1 durchführen. Die Vektoren (Y, C_b, C_r) und (R, G, B) beschreiben hier die Werte eines einzelnen Bildpixels, welcher anhand der Multiplikation mit der Transforma-

tionsmatrix konvertiert wird. Da die Chrominanzwerte durch diese Berechnung sowohl positive als auch negative Werte annehmen können, wird deren Wertebereich oft durch die Addition mit dem Wert 128 verschoben, um ausschließlich mit positiven Werten weiterarbeiten zu können [vgl. BB15, S. 340].

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.1}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.000 & 0.000 & 1.403 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.773 & 0.000 \end{pmatrix} \cdot \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} \tag{2.2}$$

Die Rücktransformation in den RGB-Farbraum erfolgt ähnlich zur Hintransformation, die Transformationsmatrix enthält nun andere Werte (siehe Formel 2.2). Sollte bei der ursprünglichen Transformation der Wertebereich der Chrominanzwerte verschoben worden sein, muss diese Verschiebung hier durch die Subtraktion des Wertes 128 rückgängig gemacht werden.

Diskrete Kosinustransformation (DCT)

Die diskrete Kosinustransformation ist eine Transformation, welche ausschließlich Kosinus-Funktionen unterschiedlicher Wellenzahl als Basisfunktionen“ [BB15, S. 535] verwendet. Sie wird „vor allem im Bereich der Bild- und Videokompression“ [BB15, S. 535] eingesetzt, beispielsweise bei der JPEG-Kompression von digitalen Bildern.

$$S_{uv} = \frac{1}{4} \cdot C_u \cdot C_v \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} \cdot s_{yx} \cdot \cos \frac{(2x+1)u \cdot \pi}{16} \cdot \cos \frac{(2y+1)v \cdot \pi}{16}$$

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{für } u = 0 \\ 1 & \text{für } sonst \end{cases} \qquad C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{für } v = 0 \\ 1 & \text{für } sonst \end{cases} \tag{2.3}$$

Die zweidimensionale DCT arbeitet bei der Transformation von digitalen Bildern mit einer Blockgröße von 8x8 Pixeln. Deshalb wird das zu transformierende Bild vor der Durchführung der DCT in 8x8 Pixel große Blöcke unterteilt, die anschließend nacheinander transformiert werden. Die Berechnung des DCT-Koeffizienten eines einzelnen Blockpixels erfordert dabei die Kenntnis über alle anderen Pixelwerte desselben Blocks. Die Berechnung eines DCT-Koeffizienten S_{uv} erfolgt anhand von Formel 2.3, wobei (x,y) die Koordinaten des zu transformierenden Pixels s_{xy} im Bild und (u,v) die Koordinaten des jeweiligen Pixels im Block repräsentieren. C_u und C_v sind Variablen, die für den ersten Pixel eines Blockes (obere linke Ecke) einen anderen Wert ergeben als für die

restlichen Blockpixel. Bei diesem ersten Pixel handelt es sich um den DC-Koeffizienten, alle restlichen Pixel im Block werden AC-Koeffizienten genannt. Der DC-Koeffizient enthält den Gleichanteil und ist proportional zum durchschnittlichen Pixelwert des Originalblocks. Die AC-Koeffizienten beschreiben ihre Abweichung vom DC-Koeffizienten, wobei die AC-Koeffizienten nahe des DC-Koeffizienten die niedrigen Frequenzen beschreiben. Je näher ein AC-Koeffizient an der rechten unteren Ecke eines Blocks liegt, desto eher enthält er Informationen über die hohen Frequenzanteile des Blocks wie Kanten oder Bildrauschen.

$$s_{yx} = \frac{1}{4} \cdot \sum_{u=0}^{7} \sum_{v=0}^{7} \cdot C_u \cdot C_v \cdot S_{uv} \cdot \cos \frac{(2x+1)u \cdot \pi}{16} \cdot \cos \frac{(2y+1)v \cdot \pi}{16} \tag{2.4}$$

Um ein Bild auf gleiche Weise vom Frequenzraum zurück in den Ortsraum zu überführen, kommt im Falle der DCT die Inverse Diskrete Kosinustransformation (IDCT) zum Einsatz. Die Formel zur Berechnung der IDCT unterscheidet sich nur minimal von der Formel der DCT (siehe Formel 2.4).

Wavelet-Transformationen

Wavelet-Transformationen sind Transformationen vom Orts- in den Frequenzraum, welche Wavelets als Basisfunktionen verwenden. Die bekannteste unter ihnen ist die Diskrete Wavelet-Transformation (DWT). Durch Anwendung der zweidimensionalen DWT auf ein digitales Bild, wird auf dieses zeilen- und spaltenweise ein Hoch- und ein Tiefpassfilter angewendet. Nach dieser Filterung findet in den Teilbändern ein Reduktion der Abtastrate statt, durch die jeder zweite DWT-Koeffizient entfernt wird. So entsteht nach einer einstufigen 2D-DWT ein in 4 Teilbänder unterteiltes Bild, wobei jedes Teilband nur noch ein Viertel der Größe des Originalbildes aufweist (siehe Abb. 2.1). Das Approximations-Teilband (LL) enthält die doppelt tiefpass-gefilterten DWT-Koeffizienten, das diagonale Teilband (HH) die doppelt hochpass-
gefilterten. Die anderen beiden Teilbänder (LH und HL) enthalten die DWT-Koeffizienten, die jeweils ein Mal zeilen- oder spaltenweise mit einem Tiefpass- und einem Hochpassfilter transformiert wurden.

Für die Detektion digitaler Bildmanipulationen kommt die 2D-DWT häufig zum Einsatz, um die Dimensionen des Bildes zu reduzieren. Dies verringert gleichzeitig die Rechenzeit des verwendeten Detektionsverfahrens. So werden je nach Verfahren DWTs unterschiedlicher Stufen verwendet. Während bei der einstufigen DWT das Bild in 4 Teilbänder unterteilt wird, erhöht sich die Anzahl der Teilbänder pro Level um drei, wobei in jeder Stufe jeweils das LL-Teilband weiter unterteilt wird. So ergeben sich bei der zweistufigen DWT beispielsweise sieben, bei der dreistufigen DWT zehn Teilbänder. Da das LL-Teilband die relevantesten Informationen des Bildes enthält, wird bei den meisten Detektionsverfahren auf diesem weitergearbeitet.

Die DWT ist eine translationsvariante Wavelet-Transformation. Das bedeutet, dass durch eine kleine Verschiebung im zu transformierenden Bild große Unterschiede in den entstehenden DWT-Koeffizienten zu verzeichnen sind [vgl. Mah+18, S. 205]. Viele Detektionsverfahren vergleichen Bildregionen anhand ihrer Merkmalsvektoren, um duplizierte Regionen zu finden. Werden diese Merkmalsvektoren aus DWT-Koeffizienten erstellt und wird eine duplizierte Region um wenige Pixel im Bild verschoben, können sich die erstellten Vektoren deutlich verändern. Im schlimmsten Fall kann eine leichte Verschiebung im Bild also zu einer falschen Detektion führen.

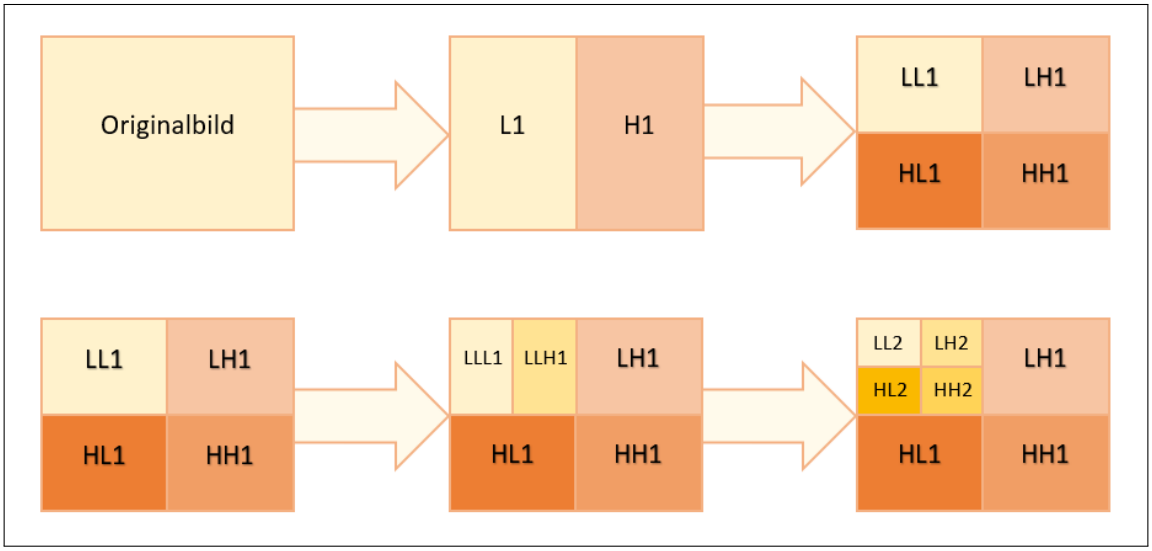


Abbildung 2.1: Ablauf der zweidimensionalen zweistufigen Diskreten Wavelet-Transformation, Eigene Darstellung

Um diese Schwäche der DWT zu umgehen, verwenden einige Detektionsverfahren alternativ die Stationäre Wavelet-Transformation (SWT). Diese ist translationsinvariant und daher besser geeignet für die Merkmalsextraktion, Muster- und Manipulationserkennung [vgl. Mah+18, S. 205]. Wie bei der DWT, wird auch bei der einstufigen 2D-SWT das Ausgangsbild durch mehrere Tief- und Hochpassfilter in vier Teilbänder unterteilt. Im Unterschied zur DWT findet hier nach der Filterung jedoch keine Reduktion der Abtastrate statt. Die entstehenden Teilbänder haben die gleiche Größe wie das Ausgangsbild. Dies erhöht zwar die Rechenzeit und den Speicheraufwand, kann jedoch zu besseren Detektionsergebnissen führen. Generell muss zwischen diesen Größen je nach Anwendungsfall immer ein Kompromiss gefunden werden.

Random Sample Consensus (RANSAC)

Der Random Sample Consensus, kurz RANSAC, ist ein iterativer Resampling-Algorithmus, mit dessen Hilfe aus einer Menge von Datenpunkten Ausreißer entfernt werden können. Bekannte Algorithmen wie die Methode der kleinsten Quadrate (MKQ) erstellen ein initiales Modell unter Verwendung möglichst vieler Datenpunkte, aus welchem durch nachfolgende Berechnungen Ausreißer entfernt werden [vgl. FB81, S. 383]. Im Gegen-

satz dazu verwendet RANSAC zur Erstellung des initialen Modells so wenig Datenpunkte wie möglich. Anhand dieses Modells wird anschließend eine Teilmenge mit Datenpunkten erstellt, welche iterativ durch weitere passende Datenpunkte ergänzt wird [vgl. FB81, S. 383]. Dabei wird das Modell immer weiter angepasst und es wird versucht, die beste Teilmenge ohne Ausreißer zu finden, das sogenannte beste Consensus Set.

Die Funktionsweise des RANSAC-Algorithmus' wird von den Autoren wie folgt beschrieben [vgl. FB81, S. 383]: Das initiale Modell benötigt mindestens n Datenpunkte zur Initialisierung seiner Parameter. Gegeben sei eine Datenmenge P mit einer Anzahl an Datenpunkten größer oder gleich n. Im ersten Schritt werden aus der Menge P willkürlich n Datenpunkte gewählt, welche die Teilmenge S1 bilden. Anhand dieser Teilmenge wird das Modell M1 initialisiert. Als nächster Schritt wird für das gerade erzeugte Modell M1 die Fehlertoleranz berechnet. Anschließend werden diejenigen Datenpunkte der Teilmenge S1 ausgewählt, die innerhalb dieser Fehlertoleranz liegen, sie bilden das Consensus Set S1* von S1.

Wenn die Anzahl der Datenpunkte des Consensus Sets S1* einen vordefinierten Schwellwert t überschreitet, wird S1* verwendet, um ein neues Modell M1* zu berechnen. Andernfalls werden die vorherigen Schritte wiederholt, es wird eine neue Teilmenge S2 gewählt. Konnte nach einer vordefinierten Anzahl an Iterationen kein Consensus Set mit t oder mehr Datenpunkten gefunden werden, so wird das Modell anhand des größten Consensus Sets berechnet oder der Algorithmus wird als fehlgeschlagen beendet.

Im Bereich der Detektion digitaler Bildmanipulationen wird RANSAC häufig bei der Erkennung duplizierter Bildregionen verwendet. Nachdem durch vorangegangene Berechnungen eine Menge aus Block- oder Punktpaaren gebildet wurde, welche potentiell kopierte Bildregionen repräsentiert, werden mithilfe von RANSAC falsch zugeordnete Blöcke oder Punkte entfernt. Dabei kann es sich beispielsweise um Blockpaare handeln, die anhand ihrer Pixeleigenschaften fälschlicherweise als duplizierte Region eingestuft wurden. So kann durch die Verwendung von RANSAC die Anzahl falsch positiver Detektionsergebnisse reduziert werden.

2.2 Statistische Gütemaße

Statistische Gütemaße, auch als Performanzmaße bezeichnet, werden u.a. bei der Klassifikation von Daten verwendet, um das Ergebnis der Klassifikation zu bewerten und einzuschätzen, wie gut der gewählte Klassifikator für ein bestimmtes Anwendungsbeispiel geeignet ist. Die Detektion von digitalen Bildmanipulationen stellt dabei einen Fall der binären Klassifikation dar, bei welcher eine Instanz einer von zwei möglichen Klassen zugeordnet wird [vgl. Faw06, S. 861]. Im Anwendungsfall der Bildmanipulationserkennung handelt es sich bei der Instanz um das zu untersuchende Bild, welches entweder als original oder als manipuliert klassifiziert wird.

Bei der binären Klassifikation gibt es für jede Instanz 4 mögliche Klassifikationsergebnisse [vgl. Faw06, S. 861]. Diese geben an, welcher der beiden Klassen die Instanz zugewiesen wurde und ob diese zugewiesene Klasse der realen Klasse der Instanz entspricht. Zur Veranschaulichung wird dafür i.d.R. eine Konfusionsmatrix, auch Kontingenztafel genannt, verwendet.

Anhand von Abbildung 2.2 soll der Aufbau einer Konfusionsmatrix am Beispiel der Bildmanipulationserkennung erläutert werden. Die gelb eingefärbten Zellen der Konfusionsmatrix bilden den Tabellenkopf. Dieser gibt an, ob es sich bei der klassifizierten Instanz um ein manipuliertes oder nicht manipuliertes Bild handelt (links) und ob das Bild durch den Klassifikator als manipuliert oder nicht manipuliert eingestuft wurde (oben). Die 4 Zellen in der Mitte der Konfusionsmatrix zeigen die 4 möglichen Klassifikationsergebnisse.

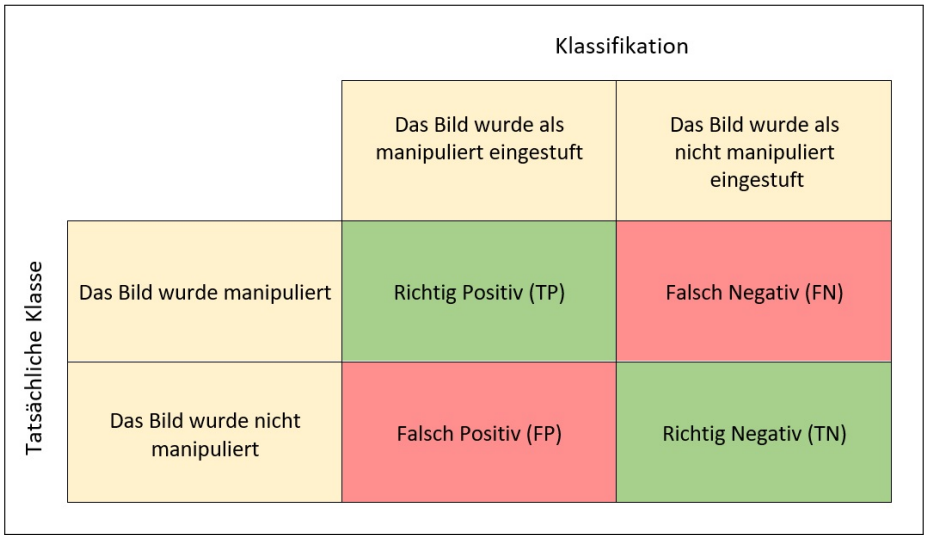


Abbildung 2.2: Konfusionsmatrix für die Klassifikation originaler und manipulierter digitaler Bilder, Eigene Darstellung[1]

Die grün eingefärbten Zellen enthalten jene Bilder, welche korrekt als manipuliertes oder als originales Bild eingestuft wurden. Diese beiden Fälle werden als *richtig positive* bzw. *richtig negative* Klassifikationen bezeichnet. Die beiden rot eingefärbten Zellen repräsentieren die Fälle der inkorrekten Klassifikation. Nicht manipulierte Bilder, welche jedoch als Manipulationen eingestuft wurden, bezeichnet man als *falsch positiv*. Manipulierte Bilder, welche als authentisch eingestuft wurden, erhalten die Bezeichnung *falsch negativ*.

Für die Bewertung von Klassifikatoren kommen, je nachdem, worauf bei der Klassifikation besonderer Wert gelegt wird, verschiedene statistische Gütemaße zum Einsatz, welche aus den Werten der Konfusionsmatrix berechnet werden.

[1] Die Abkürzungen TP, FP, TN und FN stehen für die englischen Bezeichnungen True Positive, False Positive, True Negative und False Negative.

Precision

Die Precision (deutsch: Präzision oder positiver Vorhersagewert) beschreibt den Anteil der richtig positiven Klassifikationen an der Summe der als positiv klassifizierten Instanzen. Im Fall von Bildmanipulationen ist Precision demnach die Wahrscheinlichkeit, dass ein als manipuliert klassifiziertes Bild tatsächlich manipuliert wurde [vgl. Chr+12, S. 1845]. Berechnet wird die Precision nach folgender Formel:

$$Precision = \frac{Richtig\ Positiv}{Richtig\ Positiv + Falsch\ Positiv} \tag{2.5}$$

Recall

Der Recall (deutsch: Richtig-Positiv-Rate oder Sensitivität) beschreibt das Verhältnis der richtig positiven Klassifikationen zur Summe aller tatsächlich positiven Instanzen. Er gibt an, wie viele der tatsächlich positiven Instanzen als solche klassifiziert wurden. Bezogen auf Bildmanipulationen zeigt er auf, wie viele der manipulierten Bilder durch den Klassifikator gefunden wurden. Für die Berechnung des Recalls wird demnach die folgende Formel verwendet:

$$Recall = \frac{Richtig\ Positiv}{Richtig\ Positiv + Falsch\ Negativ} \tag{2.6}$$

F-Maß

Mithilfe von Precision und Recall lassen sich zwei unterschiedliche Fälle positiv klassifizierter Instanzen berechnen. Um beide Maße in die Bewertung eines Klassifikators einzubeziehen, ist es möglich, sie mithilfe eines weiteren Gütemaßes zu kombinieren. Das F-Maß bietet hierfür die allgemeine Formel 2.7a, welche Precision und Recall je nach Größe der Variablen α gewichtet. Wählt man für α einen Wert kleiner Eins, so wird die Precision verstärkt gewichtet, bei einem Wert größer Eins wird die Gewichtung des Recalls verstärkt. Sollen Precision und Recall gleich stark in die Berechnung einbezogen werden, wird für α der Wert Eins verwendet. Die Berechnung erfolgt anhand der Formel für das entsprechend benannte F_1-Maß (siehe Formel 2.7b).

$$F_\alpha = \frac{(1+\alpha^2) \cdot Precision \cdot Recall}{\alpha^2 \cdot Precision + Recall} \qquad \alpha = \begin{cases} \text{Verstärkung der Precision, falls } \alpha < 1 \\ \text{harmonisches Mittel, falls } \alpha = 1 \\ \text{Verstärkung des Recalls, falls } \alpha > 1 \end{cases} \tag{2.7a}$$

$$\rightarrow F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.7b}$$

Das F-Maß gibt folglich Auskunft über das Verhältnis zwischen Precision und Recall. Das F_1-Maß dient als gewichtetes harmonisches Mittel der beiden Gütemaße und wird als Standard in vielen einfachen Anwendungsbeispielen verwendet.

Accuracy

Precision und Recall sowie die Kombination aus beiden mithilfe des F-Maßes konzentrieren sich weitestgehend auf die positiven Ergebnisse der Klassifikation, auf die korrekte Detektion manipulierter Bilder [vgl. Pow11, S. 38]. Zur Bewertung eines Klassifikators kann es jedoch von Vorteil sein, ebenfalls zu betrachten, wie gut die negativen Instanzen klassifiziert werden, sprich wie hoch die Anzahl der richtig und falsch negativen Instanzen ist. Die Accuracy (deutsch: Korrektklassifikationsrate) bietet hier die Möglichkeit zu berechnen, wie viele aller klassifizierten Instanzen ihren tatsächlichen Klassen zugeordnet wurden. Für die Detektion von Bildmanipulationen berechnet die Accuracy demnach die Wahrscheinlichkeit dafür, dass alle manipulierten Bilder als manipuliert und alle nicht manipulierten Bilder als authentisch klassifiziert wurden. Die Berechnung der Accuracy ergibt sich anhand folgender Formel:

$$Accuracy = \frac{Richtig\ Positiv + Richtig\ Negativ}{Richtig\ Positiv + Falsch\ Negativ + Richtig\ Negativ + Falsch\ Positiv} \tag{2.8}$$

Falsch-Positiv-Rate (FPR)

Ein weiteres statistisches Gütemaß, welches häufig Anwendung bei der Bewertung von binären Klassifikatoren findet, ist die Falsch-Positiv-Rate. Diese beschreibt das Verhältnis der falsch positiven Klassifikationen zur Summe aller tatsächlich negativen Instanzen [vgl. Pow11, S. 39]. Auf das Beispiel der Bildmanipulationserkennung bezogen gibt die Falsch-Positiv-Rate an, wie viele der authentischen Bilder fälschlicherweise als manipuliert klassifiziert wurden. Die Berechnung erfolgt anhand von Formel 2.9.

$$FPR = \frac{Falsch\ Positiv}{Richtig\ Negativ + Falsch\ Positiv} \tag{2.9}$$

Anwendung statistischer Gütemaße in der Praxis

In bestimmten Anwendungsbeispielen kann es vorkommen, dass einige Daten der Konfusionsmatrix nicht erhoben werden können. Die Folge daraus ist, dass auch die Berechnung bestimmter Gütemaße zur Beurteilung der Klassifikation nicht möglich ist. Ein Beispiel hierfür ist die Gesichtserkennung. Bei der Erkennung von menschlichen Gesichtern in Bildern ist das Ergebnis eines Klassifikators i.d.R. eine Menge aus Bildern, in welchen der Klassifikator ein oder mehrere Gesichter vermutet. Somit kann anhand dieser Ergebnismenge visuell bestimmt werden, wie viele der detektierten Bilder wirklich menschliche Gesichter enthalten und wie viele fälschlicherweise erkannt wurden. Der Klassifikator gibt in diesem Beispiel also Aufschluss auf die richtig positiven und die falsch positiven Ergebnisse der Gesichtserkennung. Da durch den Klassifikator jedoch keine Aussage getroffen wird über die falsch negativen und richtig negativen Bilder, kann u.a. nicht festgestellt werden, wie viele Bilder mit Gesicht nicht als solche erkannt wurden. In einem solchen Anwendungsfall ist es nicht möglich, Gütemaße wie Recall oder Accuracy zu berechnen, da deren Berechnung auf der Verwendung der richtig und falsch negativen Ergebnisse beruht. Hier ist die Precision oft das einzige Gütemaß, welches berechnet werden kann.

Im Forschungsgebiet der Bildmanipulationserkennung werden für die Bewertung neu implementierter Verfahren hingegen häufig Klassifikatoren verwendet, die neben den richtig und falsch positiven Ergebnissen auch Auskunft über die richtig und falsch negativen Klassifizierungen geben. Ein Grund dafür ist, dass für die Erkennung von Bildmanipulationen, besonders im strafrechtlichen Bereich, Recall und Accuracy oft eine größere Rolle spielen als die Precision. Präzise betrachtet bedeutet dies, dass es hier wichtiger ist, alle manipulierten Bilder (Recall) bzw. richtig klassifizierten Bilder (Accuracy) zu finden als festzustellen, wie viele der als manipuliert klassifizierten Bilder tatsächlich manipuliert sind (Precision). Für Letztere gilt, dass hier nicht bestimmt werden kann, wie viele manipulierte Bilder nicht als solche erkannt wurden.

Diese Beobachtung spiegelt sich in vielen Publikationen zur Detektion digitaler Bildmanipulationen wider. Diese berechnen für die Bewertung ihrer entwickelten Verfahren i.d.R. die Accuracy als Maß darüber, wie viele Bilder korrekt klassifiziert werden, oder eine Kombination aus Recall und Falsch-Positiv-Rate zur Einbeziehung der falsch positiven Ergebnisse.

In dieser Arbeit werden alle bisher genannten statistischen Gütemaße verwendet, um die Qualität der eigens implementierten Software zu bewerten. Weiterhin werden diese Maße mit den Klassifikationsergebnissen anderer relevanter Publikationen des Forschungsgebietes verglichen, um so eine Aussage darüber treffen zu können, ob die implementierte Software einen Fortschritt in der Entwicklung neuer Klassifikatoren für digitale Bildmanipulationen darstellt (siehe Kapitel 6).

2.3 Abgrenzung des Begriffs 'Bildmanipulation'

Je nach Forschungsgebiet haben sich in den vergangenen Jahrzehnten verschiedene Definitionen der Begriffe Bildmanipulation, Bildbearbeitung und Bildfälschung etabliert. Im nun folgenden Kapitel sollen die Begriffe in Bezug auf das Forschungsgebiet der Bildmanipulationserkennung genauer voneinander abgegrenzt werden. Zusätzlich werden weitere Begriffsdefinitionen für ein verbessertes Verständnis der folgenden Kapitel erläutert.

Bildmanipulation

Der Begriff „Manipulation stammt vom lateinischen Wort ‚manipulus ' [ab] und bedeutet Handhabung" [Ros09, S. 69]. Der Begriff beschreibt ein „undurchschaubares, geschicktes Vorgehen, mit dem sich jemand einen Vorteil verschafft, etwas Begehrtes gewinnt" [Dud20a].

Im Fachbereich der Bildmanipulation kursieren verschiedene Definitionen und Auffassungen des Begriffs Manipulation. Für einige Autoren umfasst der Begriff Bildmanipulation sämtliche Veränderungen am Bild, unabhängig davon, ob diese während oder nach dem Bildaufnahmeprozess stattfinden oder welche Absicht hinter der Veränderung steckt. Andere Autoren unterscheiden Begriffe wie ‚Bildbearbeitung', ‚Bildgestaltung' oder ‚Bildfälschung' anhand der Absicht des Bearbeiters. So wird den ersten beiden Begriffen zumeist keine negative Bedeutung beigebracht, während die Begriffe ‚Bildmanipulation' und ‚Bildfälschung' als Bildveränderungen mit einer eindeutigen Täuschungsabsicht des Betrachters einhergehen [vgl. Hol13, S. 4]. Holzwarth beschreibt den Begriff der Bildmanipulation wie folgt:

> *„Von Manipulation (im engeren Sinn) sollte nur dann gesprochen werden, wenn Bilder eingesetzt werden, um eine bestimmte Wirklichkeit vorzutäuschen und die Betrachter vorsätzlich und zum eigenen Vorteil zu beeinflussen, z. B. zum Kauf von Produkten, zur Erhöhung einer Attraktivitätseinschätzung, zur Übernahme einer Meinung oder zur Wahl eines Politikers.*" [Hol12, S. 7]

Für die folgende Arbeit sollen die Begriffe noch detaillierter voneinander abgegrenzt werden. Die in Abbildung 2.3 dargestellte Abgrenzung der Begriffe unterscheidet im Wesentlichen zwischen den beiden Begriffen ‚Bildbearbeitung' und ‚Bildfälschung'.

Bildbearbeitung bezeichnet hier Veränderungen am digitalen Bild, welche das Aussehen, jedoch nicht den Bildinhalt ändern. Beispiele für Verfahren der Bildbearbeitung sind Farb- und Kontrastanpassungen sowie geometrische Transformationen wie Skalierung und Rotation. Solche Veränderungen werden in der Regel ohne Täuschungsabsicht durchgeführt, das Ziel ist hingegen eine Verbesserung der Bildqualität oder Anpassung an bestimmte Anwendungsbereiche.

Der Begriff **Bildfälschung** bezieht sich hier auf alle Veränderungen am Bild, welche das Aussehen und den Inhalt des Bildes ändern, um dem Betrachter „eine bestimmte Wirklichkeit vorzutäuschen und [ihn] vorsätzlich und zum eigenen Vorteil zu beeinflussen“ [Hol12, S. 7]. Ein Beispielverfahren der Bildfälschung ist „das absichtsvolle Hinzufügen und Beseitigen von Bildelementen“ [Sch05, S. 9].

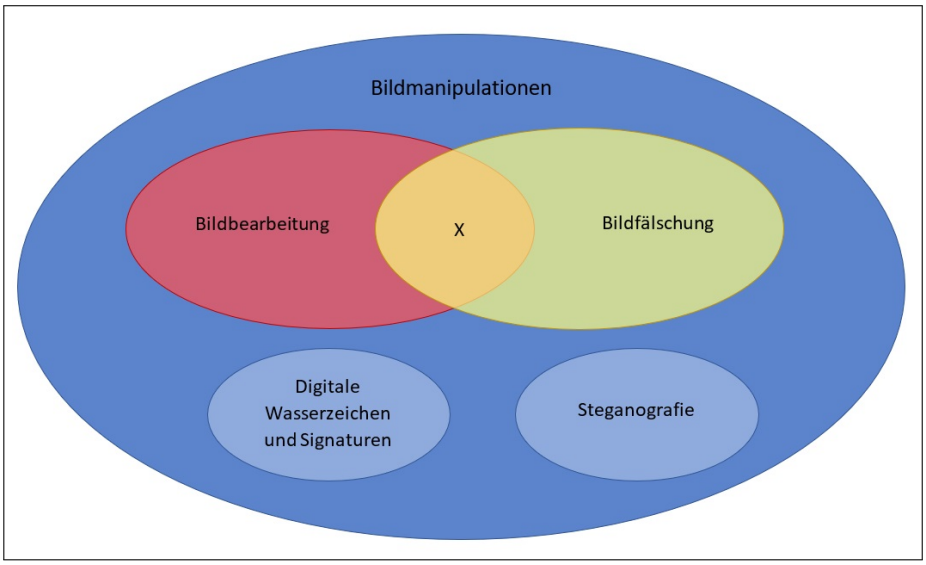


Abbildung 2.3: Abgrenzung der Begriffe Bildmanipulation, Bildbearbeitung und Bildfälschung, In Anlehnung an [ZZT18, S. 3]

Die Schnittmenge X zwischen Bildbearbeitung und Bildfälschung umfasst eine Kombination aus Verfahren beider Kategorien. Hierzu zählen beispielsweise Farb- oder Kontrastanpassungen nach dem Hinzufügen eines bildfremden Objektes, um die durchgeführte Bildfälschung durch Verfahren der Bildbearbeitung zu verschleiern.

Digitale Wasserzeichen, digitale Signaturen und Steganografie sind im Allgemeinen Verfahren, welche weder den Bildinhalt noch das Aussehen des Bildes verändern[2] [vgl. ZZT18, S. 4]. Da hier jedoch trotzdem eine Veränderung des Bildes, wenn auch auf Pixelebene und für den menschlichen Betrachter nicht sichtbar, stattfindet, zählen diese Verfahren mit zum Begriff Bildmanipulation.

Der Begriff **Bildmanipulation** dient als umfassender Oberbegriff für die oben erläuterten Begriffe. Um Verwirrung, insbesondere bei Referenzen auf andere Autoren, zu vermeiden, wird der Begriff Bildmanipulation im Folgenden jedoch im Sinne des Begriffs der Bildfälschung verwendet.

Bildforensik

Die **Forensik** ist die Wissenschaft, welche die Methoden und Techniken der Naturwissenschaften auf den Bereich des Straf- und Zivilrechts anwendet [vgl. Pag20]. Sie trägt

[2] Von Qualitätseinbuße durch die Einbettung digitaler Wasserzeichen wird hier abgesehen.

so, beispielsweise durch die Untersuchung von Fingerabdrücken, DNS[3]- und Blutspuren sowie durch die Auswertung digitaler Datenträger, zur Aufklärung von Straftaten und Zivilstreitigkeiten bei.

Die **digitale Bildforensik** stellt eines der vielen Teilgebiete der Forensik und ein direktes Untergebiet der digitalen Forensik dar. Während sich die digitale Forensik allgemein mit der Untersuchung und Wiederherstellung von Daten und Metadaten digitaler Geräte [vgl. Pag20] befasst, konzentriert sich die digitale Bildforensik auf die systematische Überprüfung der Authentizität von digitalen Bilddaten [vgl. For20]. Sie nutzt dafür sichtbare und nicht sichtbare Veränderungen in den statistischen Merkmalen von Bildern, um Bildmanipulationen zu erkennen und die Herkunft eines digitalen Bildes zu ermitteln [vgl. For20]. Man unterscheidet hier demnach zwischen den Begriffen Manipulationsdetektion und Quellenidentifikation, wobei sich Zweiteres sowohl auf den Erschaffer oder Besitzer des Bildes als auch auf das verwendete Aufnahmegerät beziehen kann.

Die **Bildmanipulationserkennung** verwendet u.a. physikalische, statistische und geometrische Merkmale [vgl. Far09, S. 16 f.] digitaler Bilder, um anhand von bestehenden Detektionsverfahren Manipulationen innerhalb eines digitalen Bildes zu erkennen und zu lokalisieren. Kenntnisse über das originale, unveränderte Bild sind dabei nicht immer notwendige Voraussetzung für die Detektion und Lokalisation von Bildmanipulationen. Anhand passiver Detektionsverfahren, welche Bildmanipulationen gänzlich ohne Informationen über das Originalbild finden können, ist die Detektion von Bildmanipulationen heutzutage auch ohne Vorkenntnisse über das vorliegende Bild möglich.

[3] DNS = Desoxyribonukleinsäure, der Träger der menschlichen Erbinformation

3 Methoden der digitalen Bildmanipulation

Um Bilder zu verändern, können je nach gewünschtem Ergebnis und Anwendungsgebiet der Manipulation verschiedene Methoden eingesetzt werden, welche das Aussehen und den Bildinhalt auf unterschiedliche Weise verändern. Diese Methoden unterscheiden sich vor allem darin, welche Eigenschaften des Bildes verändert werden sollen. Eine Vergrößerung des Bildes hat beispielsweise lediglich Einfluss auf das Aussehen oder die Qualität des Bildes, da hier keine originalen Bildinformationen entfernt werden. Das Hinzufügen oder Entfernen von Objekten beeinflusst hingegen sowohl Aussehen als auch Inhalt des Bildes. Neben sichtbaren Manipulationen gibt es ebenso Möglichkeiten, zusätzliche, für den Betrachter nicht sichtbare Daten in Bilder einzubetten.

Um manipulierte Bilder real wirken zu lassen, reicht jedoch die Verwendung einer einzelnen Methode meist nicht aus. So müssen beispielsweise Objekte, welche in ein Bild eingefügt wurden, oft skaliert oder rotiert werden, um sich dem restlichen Bildinhalt anzupassen. Auch Helligkeits- oder Kontraständerungen sowie die Anwendung von Tiefpass- und Hochpassfiltern (Weich- und Hartzeichner) werden in diesem Zusammenhang häufig benötigt, um die Übergänge zwischen originalen Bildinhalten und hinzugefügten Objekten zu verwischen.

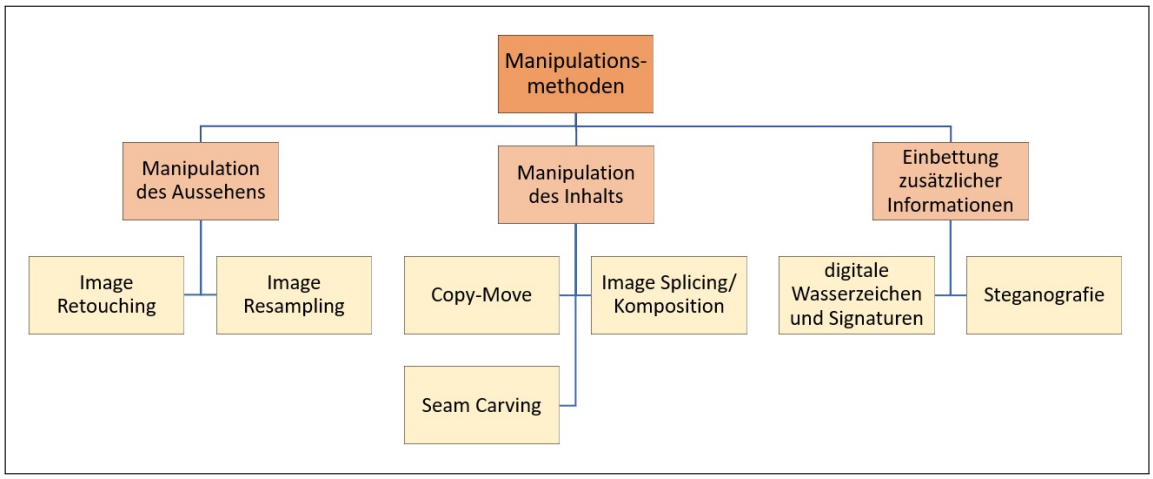


Abbildung 3.1: Einteilung der Methoden zur Manipulation digitaler Bilder, Eigene Darstellung

Im Folgenden werden die verschiedenen Methoden der digitalen Bildmanipulation erläutert. Dabei wird insbesondere auf spezifische Merkmale und Anwendungsgebiete der einzelnen Manipulationsmethoden eingegangen. Abbildung 3.1 zeigt die grundlegende Kategorisierung der Methoden in drei Untergruppen. Durch Image Resampling und Image Retouching können Merkmale eines Bildes wie die Größe, Auflösung, Farbe oder der Kontrast verändert werden. Die Methoden Copy-Move, Image Splicing und Seam Carving beeinflussen den Inhalt des Bildes u.a. durch Duplikation und Komposition einzelner oder mehrerer Bildbereiche.

Die dritte Untergruppe enthält u.a. Methoden zur Einbettung von digitalen Wasserzeichen und zur Extraktion digitaler Signaturen.

3.1 Manipulation des Aussehens

Wie bereits erwähnt, gibt es Methoden der digitalen Bildmanipulation, welche sich hauptsächlich auf die Erscheinung des Bildes, jedoch nicht auf den Bildinhalt auswirken. Diese dienen im Allgemeinen der Bildbearbeitung oder -verbesserung nach dem Bildaufnahmeprozess [vgl. WK18, S. 478], können jedoch auch als Nachbearbeitungsprozesse für andere Bildmanipulationen dienen.

3.1.1 Image Retouching

Der Begriff *Image Retouching* lässt sich etwa mit dem deutschen Begriff der Bildretusche gleichsetzen. Diese Art der Bildmanipulation dient vor allem der Betonung erwünschter sowie der Minimierung unerwünschter Bildmerkmale [vgl. QD15, S. 3]. Große Anwendung finden derartige Bildmanipulationen vor allem in der Medienindustrie [vgl. QD15, S. 3] beim Retuschieren von menschlichen Gesichtern oder Körpern.

(a) Das Original (b) Die Fälschung

Abbildung 3.2: Beispiel für die Anwendung von Image Retouching 3.2a) Ein originales Bild des Schauspielers Paul Newman 3.2b) Eine durch Image Retouching manipulierte Version des Originalbildes [Far04, S. 4]

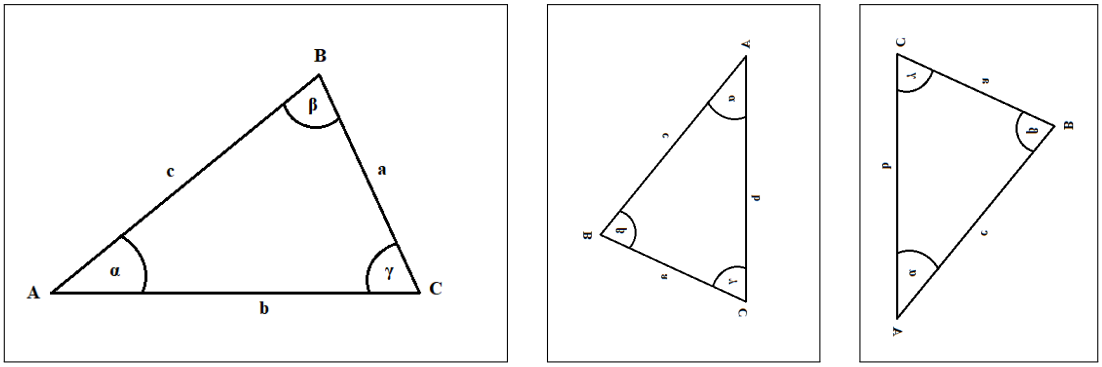
Als Beispiel für die Anwendung von Image Retouching zeigt Abbildung 3.2 zwei Bilder des US-amerikanischen Schauspielers Paul Newman (1925-2008). Im Originalfoto (siehe Abb. 3.2a) sind deutliche Altersmerkmale wie Falten, dunkle Augenringe und ein lichter Haaransatz zu sehen, es entspricht der realen damaligen Erscheinung des Schauspielers. Die manipulierte Version des Bildes (siehe Abb. 3.2b) lässt den Schauspieler um einige Jahre jünger erscheinen, hier wurden die Falten und Augenringe entfernt sowie der Haaransatz nach unten verschoben.

Image Retouching verwendet vor allem Veränderungen von Farbe, Helligkeit und Kontrast zur Qualitätsverbesserung sowie Tiefpass- und Hochpassfilter, um die Schärfe einzelner Bildbereiche anzupassen.

3.1.2 Image Resampling

Der englische Begriff *Image Resampling* (deutsch: Umtastung) umfasst die Begriffe Upsampling und Downsampling (deutsch: Erhöhung oder Reduktion der Abtastrate), welche in der Bildverarbeitung vor allem bei der Skalierung von digitalen Bildern Verwendung finden. Bezogen auf digitale Bildmanipulationen ist der Begriff jedoch weiter gefasst und bezieht sich neben der Skalierung auf weitere geometrische Transformationen wie Rotation, Spiegelung und Verzerrung von Bildern.

Die genannten Transformationen können sich sowohl auf das gesamte Bild als auch auf einzelne Bildbereiche beziehen. Hierbei ist zu beachten, dass dieser Anwendungsbereich sowie die verschiedenen Transformationen an sich eine unterschiedlich starke Wirkung auf das Aussehen eines Bildes haben können. Zudem kann sich auch die Reihenfolge der Kombination einzelner Transformationen unterschiedlich auf die Erscheinung eines Bildes auswirken.



(a) (b) (c)

Abbildung 3.3: Die Reihenfolge bei der Kombination geometrischer Transformationen beeinflusst die Erscheinung des Resultatbildes. 3a) Das Original 3b) Rotation um 90° im Uhrzeigersinn und horizontale Spiegelung 3c) Horizontale Spiegelung und Rotation um 90° im Uhrzeigersinn, Eigene Darstellung[4]

Als Beispiel hierfür zeigt Abbildung 3.3 die unterschiedliche Kombination von Rotation und Spiegelung eines rechtwinkligen Dreiecks. Während das Dreieck in Abbildung 3.3a zuerst um 90° im Uhrzeigersinn gedreht und anschließend entlang seiner horizontalen Achse gespiegelt wurde, wurden diese Transformationen bei dem Dreieck in Abbildung 3.3b in umgekehrter Reihenfolge angewendet. Die resultierende Position des Dreiecks unterscheidet sich deutlich.

[4] Die Abbildungen 3.3a) und 3.3b) wurden für eine verbesserte Darstellug herunterskaliert.

Zwar verändern die genannten Methoden nicht grundlegend den Inhalt oder die Bedeutung des Bildes, jedoch können sie bedeutenden Einfluss ausüben auf dessen Interpretation durch den Betrachter, beispielsweise können durch einfache Verbesserungen bestimmte Bildbereiche verborgen oder hervorgehoben werden [vgl. Far04, S. 4].

3.1.3 Falsche Beschriftung

Bilder stellen neben mündlicher und schriftlicher Sprache eines der Hauptmedien der menschlichen Kommunikation dar. Häufig werden verschiedene Kommunikationsmittel miteinander kombiniert, um den Inhalt einer Aussage zu betonen oder zu verändern. Ein Beispiel hierfür ist die Kombination aus Bildern und Bildbeschriftungen bzw. Untertiteln. Während ein Bild oder ein Text für sich allein gesehen bereits eine Aussage vermittelt, kann die Kombination aus Bild und Text die Interpretation durch den Betrachter in eine völlig andere Richtung lenken. Die Verwendung solcher Kombinationen erstreckt sich von humoristischen Bildmanipulationen bis hin zu solchen, welche mit negativer politischer, wirtschaftlicher oder gesellschaftlicher Absicht erstellt und verbreitet werden.

(a) Das Original, ein Stockfoto des spanischen Fotografen Antonio Guillem [Bar17]

(b) Die Fälschung, eine Variante des ‘Distracted Boyfriend Memes‘, Eigene Darstellung

Abbildung 3.4: Beispiel eines Internet Memes 3.4a) Das originale Foto ohne Text 3.4b) Das Meme als humoristische Kombination aus Bild und Text

Ein Beispiel für eine solche Kombination aus Bild und Text ist das sich immer weiter verbreitende Internetphänomen der Memes. Bei Internet Memes handelt es sich u.a. um humoristische Bilder, Videos und Texte, welche durch Internetnutzer kopiert, angepasst und verbreitet werden [vgl. Lex20]. Das bekannteste Beispiel dieses Internetphänomens sind so genannte Image Macros, bei welchen voneinander unabhängige Bilder, bspw. Auschnitte aus bekannten Filmen oder Fernsehserien, und bekannte oder originale Texte kombiniert werden, um eine humoristische Neuinterpretation des Gesamtbildes zu erzeugen. Dabei werden oft beliebte Templates (deutsch: Vorlage oder Muster) wiederverwendet und mit neuen Inhalten gefüllt, wodurch diese Templates einen Wiedererkennungswert in der Gemeinde der Internetnutzer erhalten.

Abbildung 3.4 zeigt ein Beispiel eines solchen häufig verwendeten Templates. Die Grund-

lage für das Meme bildet ein Stockfoto des spanischen Fotografen Antonio Guillem. Stockfoto ist hierbei die „[a]llgemeine Bezeichnung für Bilder, die - meist von Bild[a]genturen - vorproduziert, in Bildarchiven aufbewahrt und für aktuelle Berichterstattung, Illustration, Werbung etc. vertrieben werden“ [MS17]. Das Stockfoto (siehe Abb. 3.4a) zeigt ein Liebespaar aus Freund und Freundin, welches Händchen haltend eine Straße entlangläuft, sowie eine weitere Frau, welche das Paar passiert. Der Freund dreht sich im Laufen nach der Frau um und wirft ihr einen begehrenden Blick zu, während seine Freundin ihn dabei ertappt. Ihr Gesicht zeigt Entsetzen darüber, dass ihr Partner anderen Frauen hinterherschaut.

Das Stockfoto hat bei Internetnutzern an Beliebtheit gewonnen, welche es als Grundmaterial für die Erstellung von Memes benutzen. Dazu wird Text über den drei im Foto zu sehenden Personen eingefügt. Die Grundidee des Memes ist die konfliktbehaftete Entscheidung zwischen einem Objekt A, welches man bereits besitzt, und einem begehrten Objekt B, für welches man Objekt A eintauschen oder opfern muss. Abbildung 3.4b zeigt als Beispiel einen Mann, dessen Wille, das neueste Modell eines iPhone Smartphones zu besitzen, so stark ist, dass er dafür eine seiner Nieren verkaufen würde[5].

3.2 Manipulation des Inhalts

Neben Methoden der digitalen Bildmanipulation, welche ausschließlich das Aussehen eines Bildes verändern, gibt es auch solche, durch die sich der Bildinhalt ändert. Dies geschieht hier durch das Hinzufügen, Entfernen oder Duplizieren von Bildinformationen wie Objekten. Ziel ist es dabei, wichtige Informationen zu verstecken oder zu duplizieren [vgl. QD15, S. 3], die Dokumentation von Ereignissen durch die Anpassung des Bildinhalts zu manipulieren oder neue Bilder aus Bildbereichen mehrerer Ausgangsbilder zu erschaffen.

Der Vorgang, Bildbereiche zu kopieren oder auszuschneiden und an einer anderen Stelle des Bildes wieder einzufügen, ist die am häufigsten benutzte Methode, Bilder zu manipulieren. Dabei ist zu unterscheiden, ob Bereiche innerhalb eines Bildes oder von einem Bild in ein weiteres kopiert werden. Beide Methoden erfordern ggf. Nachbearbeitungsschritte wie Rotation, Skalierung oder Helligkeitsanpassungen des eingefügten Objekts, um das manipulierte Bild echt wirken zu lassen.

[5] Die Hintergrundinformation ist hierbei illegaler Handel mit menschlichen Organen auf dem Schwarzmarkt. Da Menschen auch mit nur einer gesunden Niere weiterleben können, wäre der Verkauf einer Niere nicht zwangsweise tödlich, würde auf dem Schwarzmarkt jedoch eine große Menge Geld einbringen

3.2.1 Copy-Move

Copy-Move, auch Copy-Paste (deutsch: Kopieren und Einfügen), beschreibt das Kopieren eines Bildbereiches innerhalb desselben Bildes, um bestimmte Regionen im Bild zu überdecken oder zu duplizieren [vgl. QD15, S. 3]. Da bei dieser Methode Objekte überlagert, jedoch nicht ausgeschnitten werden, entstehen keine Lücken im Bild, welche durch Nachbearbeitungsschritte wie Interpolation gefüllt werden müssen. Je nach Qualität der Manipulation und zusätzlicher Verwendung von Resampling-Methoden, entstehen so mehr oder weniger real wirkende manipulierte Bilder.

Es gibt verschiedene Arten, Copy-Move anzuwenden. Abbildung 3.5 zeigt eine Unterteilung der Methoden, bei welcher unterschieden wird, ob nach dem Kopieren und Einfügen eines Objektes weitere Nachbearbeitungsschritte durchgeführt werden. Die Anwendung von Copy-Move ohne nachträgliche Anpassungen unterscheidet zwischen einfachem Copy-Move, bei welchem lediglich ein einzelnes Objekt innerhalb des Bildes dupliziert wird, und multiplem Copy-Move mit der Duplikation mehrerer Objekte oder der mehrfachen Duplikation eines Objektes im Bild [vgl. WK18, S. 479].

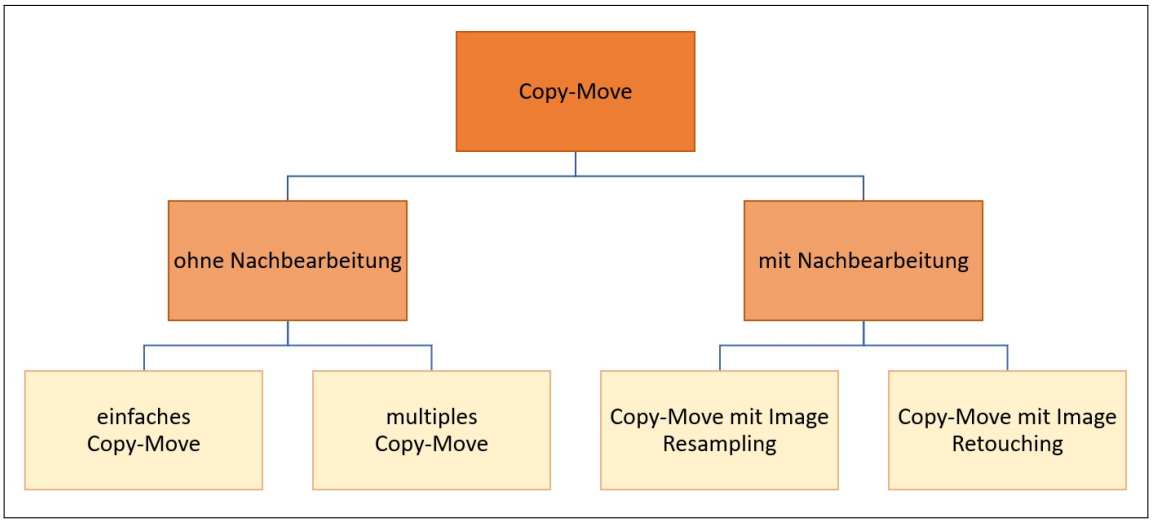


Abbildung 3.5: Unterteilung der Copy-Move-Manipulationsmethoden, Eigene Darstellung

Für die weitere Bearbeitung nach dem Kopieren und Einfügen eines oder mehrerer Bilder, kommen i.d.R. die Methoden Image Retouching und Image Resampling zum Einsatz, um das eingefügte Objekt besser an seine neue Umgebung im Bild anzupassen. Dafür wird das Objekt u.a. skaliert, rotiert oder es werden Helligkeits- und Farbanpassungen vorgenommen.

Ein Beispiel für die deutlich sichtbare Anwendung von Copy-Move stellt Abbildung 3.6 dar. Das Originalbild (siehe Abb. 3.6a) zeigt zwei nebeneinander auf der Straße sitzende Katzen. Im manipulierten Bild wurden die Katzen durch Copy-Move kopiert und neben den originalen Katzen eingefügt, sodass in diesem Bild vier statt zwei Katzen zu sehen sind. Da es sich bei den Katzen um Objekte handelt, welche sich durch ihre Textur und

Farbe deutlich vom Hintergrund abheben, ist die Manipulation leicht zu erkennen. Um mit Copy-Move real wirkende Bilder zu erstellen, sind demnach oft Nachbearbeitungsschritte nötig, damit die Manipulation dem menschlichen Auge verborgen bleibt.

(a) Das Original (b) Die Fälschung

Abbildung 3.6: Beispiel für die Verwendung von Copy-Move 3.6a) Das originale Bild mit zwei Katzen 3.6b) Das manipulierte Bild mit 4 Katzen, von denen jeweils 2 Katzen doppelt vorhanden sind. [Pyr15]

3.2.2 Image Splicing/ Komposition

Auch bei der Methode des *Image Splicings* (deutsch: Spleißen, Verbinden), auch als Komposition bezeichnet, werden Objekte eines Bildes kopiert und an anderer Stelle wieder eingefügt. Im Unterschied zur Copy-Move-Methode wird hierbei jedoch nicht nur innerhalb eines Bildes, sondern mit den Inhalten mehrerer unterschiedlicher Bilder gearbeitet [vgl. QD15, S. 3]. Auch die Kombination unterschiedlicher Bilder oder Bildteile zählt zu den Methoden des Image Splicings. Abbildung 3.7 zeigt die weitere Untertei-

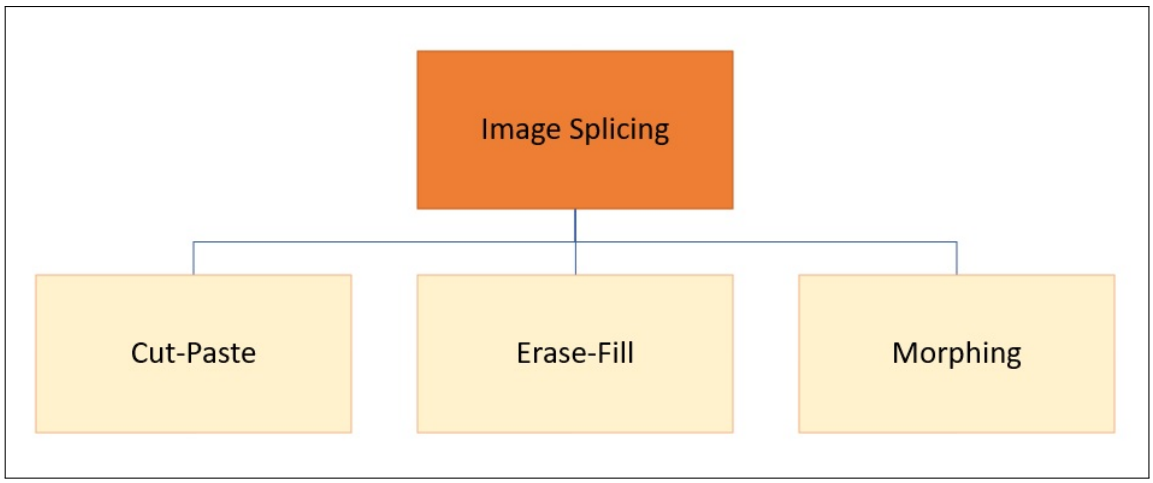


Abbildung 3.7: Unterteilung der Image-Splicing-Manipulationsmethoden, in Anlehnung an [ZZT18, S. 5]

lung des Image Splicings in die Methoden Cut-Paste, Erase-Fill und Morphing. Wie bei Copy-Move kann auch hier zwischen einfachem und multiplem Image Splicing unterschieden werden, je nachdem, ob ein oder mehrere Objekte ausgeschnitten und eingefügt werden.

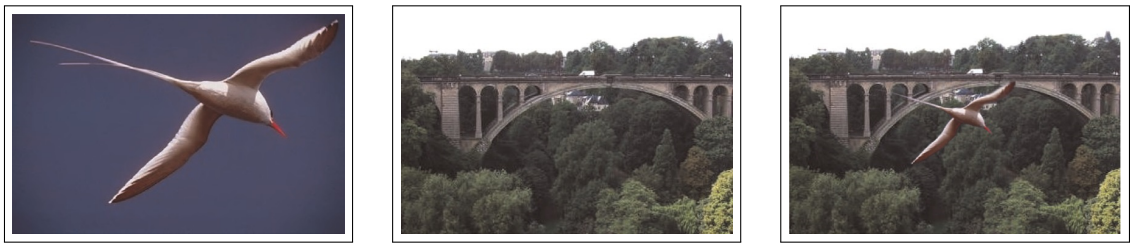
Cut-Paste

Cut-Paste (deutsch: Ausschneiden und Einfügen) ist die bekannteste Methode des Image Splicings. Hierbei wird ein Objekt eines Bildes A ausgeschnitten und in ein zweites Bild B eingefügt [vgl. ZZT18, S. 5]. Bei dieser Art der Komposition zweier Bilder wird durch das Einfügen eines Objektes ein Teil des Bildes B verdeckt. Alternativ können auch Objekte aus mehreren Bildern ausgeschnitten werden, aus welchen anschließend ein neues, manipuliertes Bild C erstellt wird. Bei beiden Methoden von Cut-Paste sind Nachbearbeitungsoperationen wie Image Resampling nötig, um die Kontur des eingefügten Objekts an seine neue Umgebung anzupassen und so den Anschein zu erwecken, es handele sich um ein originales Bild [vgl. QD15, S. 3].

Ein Beispiel für die Verwendung der Cut-Paste-Methode ist in Abbildung 3.8 zu sehen. Die Möwe aus Originalbild A ist ausgeschnitten und in das Originalbild B kopiert worden. So wurde durch die Möwe ein Teil des Bildes B verdeckt.

Erase-Fill

Die Methode *Erase-Fill* (deutsch: Löschen und Füllen) funktioniert ähnlich wie die Cut-Paste-Methode. Allerdings ist das Ziel hierbei nicht, ein Objekt des Originalbildes durch Hinzufügen eines neuen Objektes zu verdecken. Bei Erase-Fill wird zu Beginn je ein Objekt aus den Bildern A und B ausgeschnitten. Anschließend soll die Lücke, die durch das Entfernen des Objekts in Bild A entstanden ist, mit dem ausgeschnittenen Objekt des Bildes B gefüllt werden. Der Rest des Bildes B wird verworfen, da er nicht weiter benötigt wird. Auch hier sind wieder Nachbearbeitungsschritte nötig, um u.a. Form und Farbe des eingefügten Objektes an die neue Bildumgebung anzupassen.



(a) Originalbild A (b) Originalbild B (c) Die Komposition

Abbildung 3.8: Ein Beispiel für Image Splicing anhand der Cut-Paste-Methode 3.8a) Das originale Bild A 3.8b) Das originale Bild B 3.8c) Die Komposition aus Bild A und des ausgeschnittenen Objekts aus Bild B [QD15, S. 6]

Alternativ zum Einfügen eines Objektes aus Bild B kann die in Bild A entstandene Lücke

auch durch Copy-Move innerhalb des Bildes oder durch Interpolation ihrer Nachbarpixel gefüllt werden. Darauf soll hier jedoch nicht weiter eingegangen werden.

Morphing

Mit *Morphing* (deutsch: Verwandlung) wird eine Technik bezeichnet, mit welcher zwei digitale Bilder so überlagert werden, dass aus den Informationen beider Bilder ein neues Bild entsteht. Während des Morphing-Vorgangs nimmt das Ausgangsbild allmählich die Merkmale des Zielbildes an. Hierdurch entstehen mehrere Zwischenbilder, welche eine Kombination des Ausgangs- und des Zielbildes darstellen [vgl. Far04, S. 3]. Diese Zwischenbilder werden oft als Manipulationen, vor allem für humoristische Zwecke, erstellt und verbreitet. Um mit Morphing real wirkende Bilder zu erstellen, ist es wichtig, dass die Objekte im Ausgangs- und im Zielbild ähnliche Merkmale wie z.B. Größe oder Pose aufweisen [vgl. Far04, S. 3]. Um Gesichter zu morphen werden beispielsweise bestimmte Gesichtsmarker verwendet, welche in jedem menschlichen Gesicht existieren.

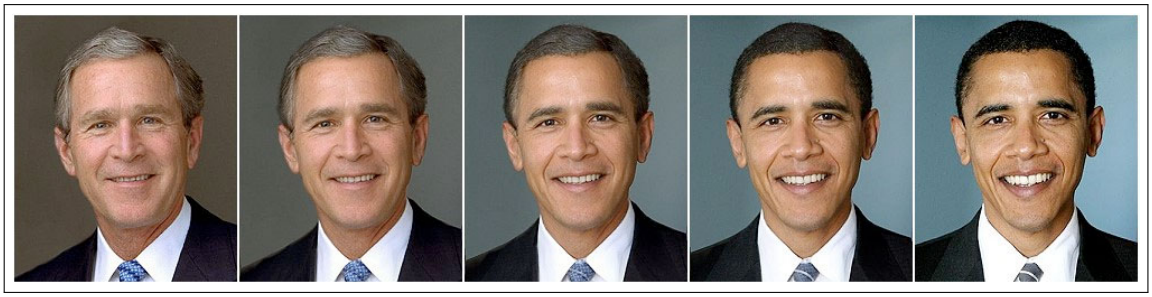


Abbildung 3.9: Morphing der Gesichter von George W. Bush und Barack Obama [Bak09]

Abbildung 3.9 zeigt ein Beispiel, bei welchem Morphing auf die Gesichter der beiden ehemaligen US-Präsidenten George W. Bush und Barack Obama angewendet wurde. In der Bildreihe zu sehen sind Bush (links), Obama (rechts) und drei Zwischenbilder, die während des Morphingprozesses entstanden sind. Das mittlere der Zwischenbilder enthält dabei zu gleichen Anteilen Merkmale beider Gesichter, die Bilder links und rechts des mittleren Bildes enthalten jeweils einen größeren Anteil eines Gesichtes.

3.2.3 Seam Carving

Seam Carving (deutsch wörtlich: Schnitzen einer Fuge/Naht) ist ein so genanntes CAIR-Verfahren[6]. Ziel eines solchen Verfahrens ist es, bei der Anwendung von Image Resampling die wichtigsten Anteile des Bildes zu erhalten [vgl. CSH13, S. 632]. Seam Carving wird speziell zur Skalierung von Bildern verwendet. Dafür werden über einen Algorithmus Pfade benachbarter Pixel (Seams) gesucht, welche eine möglichst geringe Relevanz im Bild haben. Diese Seams verlaufen in der Regel horizontal oder vertikal durchs

[6] CAIR steht für Content Aware Image Retargeting, deutsch etwa: inhaltsbewusste Neuanordnung eines Bildes

Bild, jedoch ist es auch möglich, diagonal verlaufende Seams zu verwenden. Zur Skalierung des Bilds werden die berechneten Seams nun entfernt (Herunterskalierung) oder hinzugefügt (Hochskalierung) [vgl. CSH13, S. 632].

(a) Das Original ohne Seams

(b) Das Original mit vertikalen Seams

(c) Das skalierte Bild

Abbildung 3.10: Ein Beispiel für Seam Carving mit vertikalen Seams 3.10a) Das originale Bild 3.10b) Das Original mit eingezeichneten Seams 3.10c) Das herunterskalierte Bild [Yin+15, S. 132]

Ein Beispiel für die Verkleinerung eines Bildes durch Seam Carving ist in Abbildung 3.10 zu sehen. Das Originalbild (Abb. 3.10a) zeigt einen Kirschbaum in voller Blüte. Die Abbildungen 3.10b) und 3.10c) zeigen das Originalbild mit berechneten vertikalen Seams und das skalierte Bild, bei welchem die entsprechenden Seams entfernt wurden. Das skalierte Bild wirkt trotz entfernter Seams realistisch, da nur unwichtige Bildanteile entfernt wurden.

4 Detektion digitaler Bildmanipulationen

Mit der fortschreitenden Entwicklung von Bildbearbeitungssoftwares und anderen Technologien wachsen auch die Möglichkeiten, digitale Bilder zu manipulieren. So besitzen heutige Bildmanipulationen oft eine so gute Qualität, dass die durchgeführte Manipulation durch das menschliche Auge allein kaum bis gar nicht mehr wahrnehmbar ist. Um digitale Bildmanipulationen eindeutig und beweissicher festzustellen, ist daher die Entwicklung und Verwendung sicherer Detektionsalgorithmen nötig. Welche Arten von Verfahren dabei entwickelt werden, ist abhängig von den am häufigsten auftretenden Manipulationsmethoden.

4.1 Einteilung der Detektionsverfahren

4.1.1 Aktive und passive Verfahren

Die bekannteste Einteilung von Methoden zur Detektion digitaler Bildmanipulationen ist die Unterscheidung zwischen aktiven und passiven Detektionsverfahren. Diese Einteilung basiert vor allem auf der Frage, ob der Zugriff zur Originalversion eines vorliegenden, potentiell manipulierten Bildes möglich ist.

Aktive Detektionsverfahren arbeiten i.d.R. mit digitalen Wasserzeichen oder digitalen Signaturen, welche bereits während des Aufnahmeprozesses durch das Aufnahmegerät selbst oder durch eine dafür autorisierte Person in ein Bild eingebettet werden [vgl. QD15, S. 4]. Diese eingebetteten Informationen enthalten einen eindeutigen Authentifikationscode, durch welchen zu einem späteren Zeitpunkt die Authentizität des Bildes festgestellt werden kann [vgl. BM13, S. 227]. Digitale Bildmanipulationen können die Struktur der eingebetteten Information willentlich oder unbeabsichtigt zerstören oder verändern, sodass von der Veränderung des Wasserzeichens oder der Signatur auf eine Manipulation des Bildes geschlossen werden kann.

Im Gegensatz zu aktiven Methoden benötigen **passive Detektionsverfahren**, auch als blinde Verfahren bezeichnet, keine Informationen über das Original des potentiell manipulierten Bildes, sie verwenden ausschließlich das vorhandene Bild, um dessen Authentizität und Integrität zu überprüfen [vgl. BM13, S. 228]. Die Funktionsweise passiver Detektionsverfahren basiert auf der Annahme, dass digitale Bilder feste statistische Merkmale besitzen, welche durch die Anwendung von Manipulationsmethoden oder Nachbearbeitungsoperationen verändert werden. So ist es möglich, ein Bild auf Unstimmigkeiten in seinen statistischen Eigenschaften und auf das Vorhandensein neuer, durch Manipulation entstandener Artefakte zu überprüfen [vgl. BM13, S. 228].

Da die Verwendung aktiver Detektionsverfahren den Zugriff auf das Original eines manipulierten Bildes voraussetzt, dieses in der Praxis jedoch oft nicht vorhanden ist, kommen hier hauptsächlich passive Detektionsverfahren zum Einsatz. Hinzu kommt der Nachteil aktiver Verfahren, dass die Einbettung digitaler Wasserzeichen oder Signaturen i.d.R. spezielle Hardware oder Software erfordert, welche in herkömmlichen Digitalkameras nicht grundsätzlich implementiert ist [vgl. BM13, S. 228]. Aufgrund dieser Limitierungen ist die folgende Erläuterung und Implementierung spezieller Detektionsverfahren auf den Bereich der passiven Verfahren beschränkt. Aktive Verfahren werden im Folgenden nicht weiter behandelt.

4.1.2 Low Level, Middle Level und High Level

Neben der oben erläuterten Einteilung der Detektionsverfahren in aktive und passive Methoden, besteht die Alternative, diese nach Art der im Bild vorhandenen Informationen zu kategorisieren. So können Detektionsverfahren nach [WDT09] den folgenden drei Levels zugeteilt werden:

Low Level

Während des Bildaufnahmeprozesses entstehen konsistente Korrelationen zwischen benachbarten Pixeln, welche durch Bildmanipulationen zerstört werden [vgl. WDT09, S. 309]. Low-Level-Detektionsmethoden suchen daher nach Unstimmigkeiten in diesen Korrelationen, um Rückschlüsse auf vergangene Bildmanipulationen ziehen zu können. Statt semantischen Bildinformationen nutzen diese Methoden statistische Charakteristiken des Bildes [vgl. WDT09, S. 309].

Middle Level

Methoden des Middle Levels untersuchen semantische Zusammenhänge im Bild, welche durch die Anwendung bestimmter Manipulationsmethoden oder Nachbearbeitungsoperationen entstehen. Dazu zählen beispielsweise scharfe Kanten oder Unstimmigkeiten in der Beleuchtung von Objekten desselben Bildes, die durch Image Splicing entstanden sind [vgl. WDT09, S. 309]. Für die Erkennung solcher Unstimmigkeiten kommen moderne Detektionsalgorithmen zum Einsatz.

High Level

Wie der Name vermuten lässt, bildet das High Level, auch als semantisches Level bezeichnet, das höchste und damit das anspruchsvollste der drei Levels. Es verwendet zur Detektion von Bildmanipulationen ausschließlich semantische Unstimmigkeiten im Bild. Im Gegensatz zum Middle Level beziehen sich diese Informationen jedoch nicht auf Unstimmigkeiten in der optischen Erscheinung eines Bildes, sondern auf semantische Fehler des Bildinhalts. Ein Beispiel für solch eine Unstimmigkeit ist die fotografische Darstellung zweier Personen, welche in unterschiedlichen Jahrhunderten lebten

und demnach nie gemeinsam abgelichtet werden konnten. Da es dem Menschen durch die steigende Qualität von manipulierten Bildern zunehmens schwerer fällt, solche mit bloßem Auge zu erkennen, bieten moderne Computersysteme und Detektionsalgorithmen eine gute Unterstützung zur Detektion digitaler Bildmanipulationen. Bisher zeigten sich derartige Verfahren jedoch lediglich im Bereich des Low Levels und Middle Levels erfolgreich [vgl. WDT09, S. 309]. Die Erkennung semantischer Zusammenhänge stellt sowohl im Fachbereich der Bildforensik als auch in anderen Fachgebieten wie Text Mining immer noch eine große Herausforderung dar, welche beispielsweise durch den Einsatz künstlicher neuronaler Netze zukünftig überwunden werden soll. Etablierte Detektionsalgorithmen für digitale Bildmanipulationen arbeiten demnach hauptsächlich auf dem Low oder Middle Level.

4.2 Passive Detektionsverfahren

Ein durch eine Kamera erzeugtes digitales Bild dient als Momentaufnahme einer Szene der realen Welt. Dieses Abbild speichert alle aufgenommenen Informationen in Pixeln (Rastergrafiken) oder Vektoren (Vektorgrafiken) ab. So enthalten digitale Bilder weit mehr Informationen als die radiometrischen Eigenschaften ihrer Pixel. Neben Farb- und Grauwerten werden u.a. Informationen zu den im Bild vorhandenen Objekten sowie zu den Beleuchtungsverhältnissen zum Zeitpunkt der Bildaufnahme festgehalten.

Durch die Vielfalt der durch eine Kamera aufgenommenen Informationen stellt die Detektion von digitalen Bildmanipulationen demnach ein multidimensionales Problem dar [vgl. QD15, S. 6]. Je nach Manipulationsmethode, die auf ein [originales] Bild angewandt wird, können einige Detektionsverfahren hervorragende Ergebnisse liefern, während andere Verfahren keine Wirkung zeigen [vgl. QD15, S. 6].

Auf dieser Feststellung aufbauend, nennt [Far09] fünf Kategorien, in welche passive Detektionsverfahren weiter unterteilt werden: pixelbasierte, format- bzw. kompressionsbasierte, physikbasierte, geometriebasierte und statistikbasierte Verfahren.

Abbildung 4.1 zeigt die oben erläuterte Einteilung der Detektionsverfahren in aktive und passive Verfahren sowie die weitere Unterteilung der passiven Verfahren. Die Darstellung enthält ebenfalls Beispiele für die einzelnen Bereiche der passiven Detektionsmethoden. Auf diese soll im Folgenden überblicksweise eingegangen werden. Eine detaillierte Beschreibung bestimmter Algorithmen folgt in Kapitel 5.1.

4.2.1 *Pixelbasiert*

Pixelbasierte Detektionsverfahren arbeiten, wie der Name bereits vermuten lässt, auf Pixelebene. Digitale Bilder werden hierbei auf statistische Irregularitäten in Pixeln so-

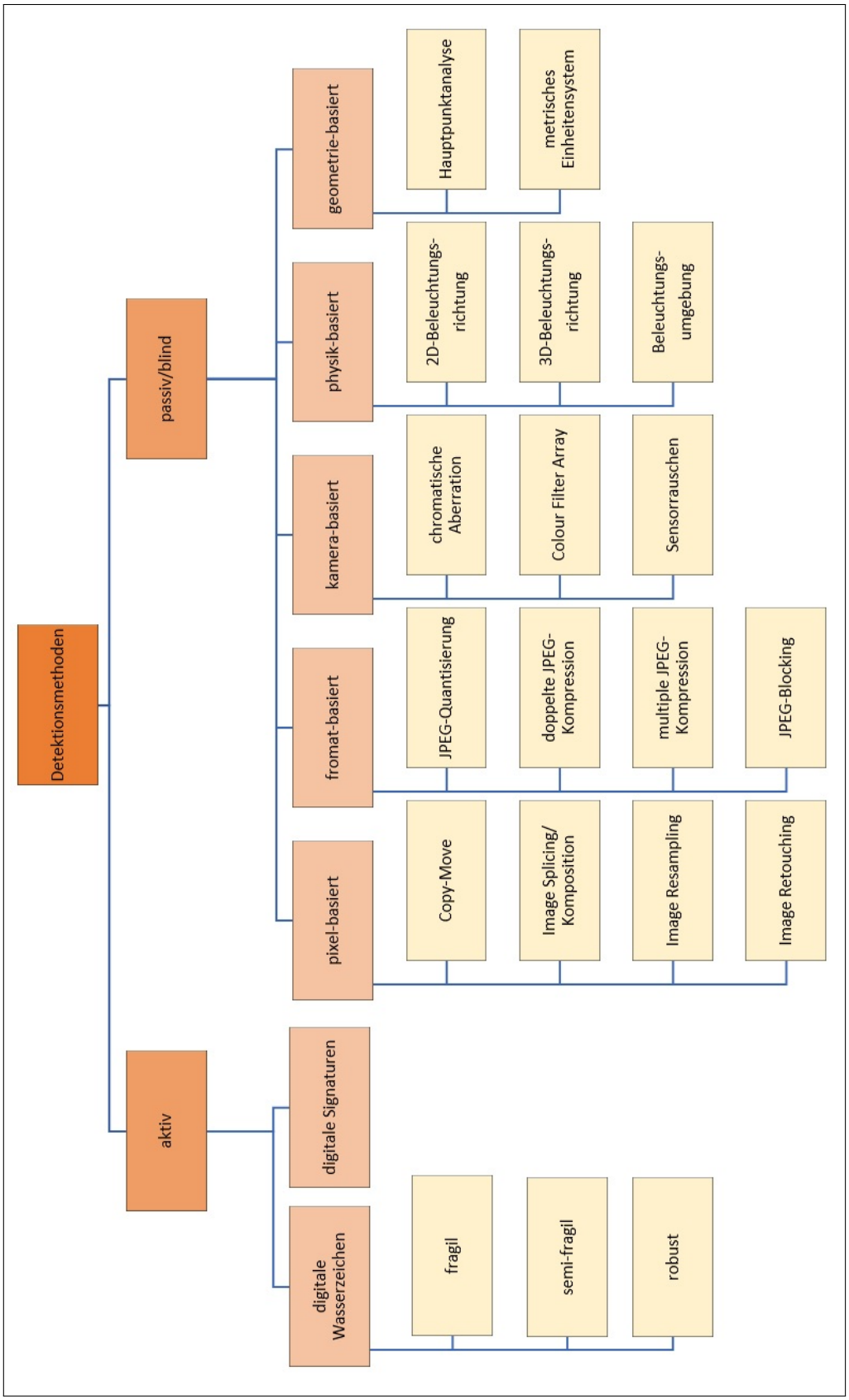


Abbildung 4.1: Einteilung der Detektionsverfahren für digitale Bildmanipulationen, in Anlehnung an [QD15, S. 6] und [NS16, S. 4]

wie auf lokale und globale Korrelationen zwischen Pixeln untersucht, welche durch die Anwendung bestimmter Manipulationsmethoden entstehen [vgl. QD15, S. 5].

Die am häufigsten verwendeten Manipulationsmethoden Copy-Move, Image Splicing, Image Resampling und Image Retouching wurden bereits in Kapitel 3 behandelt. Verbreitet sind diese Methoden der Manipulation und damit auch die entsprechenden Detektionsverfahren insbesondere dadurch, dass sie keine Informationen über den Aufnahmeprozess eines Bildes oder die durchgeführten geometrischen Transformationen benötigen.

Copy-Move

Wie bereits erwähnt, können durch Copy-Move Bildausschnitte eines Bildes kopiert und an anderer Stelle desselben Bildes wieder eingefügt werden. Durch diesen Kopiervorgang entstehen im Bild eine oder mehrere Regionen, deren Pixel nahezu dieselben Eigenschaften aufweisen. Unterschiede zwischen den duplizierten Pixeln können durch Nachbearbeitungsschritte wie Image Resampling des eingefügten Objektes oder verlustbehaftete Kompression wie JPEG entstehen. [vgl. MS08, S. 281].

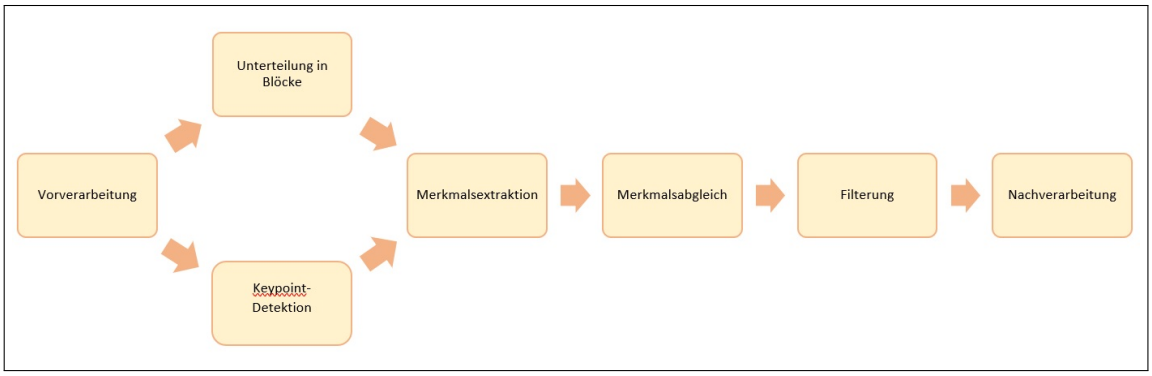


Abbildung 4.2: Allgemeiner Ablauf der Detektion von Copy-Move-Manipulationen, in Anlehnung an [Chr+12, S. 1842]

Das Ziel einiger bisheriger Detektionsverfahren war es, solche duplizierten Regionen in einem potentiell manipulierten Bild zu finden. Der generelle Ablauf eines solchen Verfahrens ist in Abbildung 4.2 zu sehen. Copy-Move-Detektionsverfahren beginnen häufig mit der **Vorverarbeitung** des Eingangsbildes. Eine Ursache hierfür ist, dass viele dieser Verfahren mit Graustufenbildern oder mit Bildern des YCbCr-Farbmodells arbeiten, der Großteil manipulierter Bilder jedoch im RGB-Format vorliegt und erst entsprechend transformiert werden muss [vgl. Chr+12, S. 1842].

Auf die Vorverarbeitung des Eingangsbildes folgt die **Extraktion von Merkmalen**, durch welche duplizierte Regionen gefunden werden können. Dieser Extraktionsschritt unterscheidet sich je nach Art des verwendeten Detektionsverfahrens. So hat es sich etabliert, die Verfahren in zwei Kategorien einzuteilen: blockbasierte und keypointbasierte Verfahren.

Blockbasierte Verfahren unterteilen das zu untersuchende Bild in Blöcke, aus welchen sie anschließend Merkmalsvektoren extrahieren [vgl. Chr+12, S. 1842]. Je nach Verfahren unterscheiden sich diese Blöcke in ihrer Größe, Form (rechteckig, kreisförmig, irregulär) und Anzahl. Etabliert hat sich hierbei die Verwendung von quadratischen, sich überlappenden Blöcken. Die Extraktion von Merkmalsvektoren erfolgt durch Anwendung verschiedener Transformationen und Rechenverfahren.

Im Gegensatz zu blockbasierten Verfahren verzichten **keypointbasierte Verfahren** auf die Unterteilung des Bildes und arbeiten stattdessen mit Regionen, welche besonders hohe Entropiewerte aufweisen, sogenannten Keypoints (deutsch: Kernpunkte oder Schlüsselpunkte). Aus diesen Punkten werden durch unterschiedliche Verfahren ebenfalls Merkmalsvektoren extrahiert. Die beiden am häufigsten verwendeten Verfahren sind hierbei SIFT[7] und SURF[8], die aufgrund ihrer jeweiligen Eigenschaften sehr robust sind gegenüber geometrischen Transformationen wie Skalierung oder Rotation, welche die Extraktion von Merkmalsvektoren erschweren können.

Auf die Extraktion aussagekräftiger Merkmale des Bildes folgt der **Abgleich** der gefundenen Merkmalsvektoren (englisch: Matching). Befinden sich an unterschiedlichen Stellen im Bild dieselben Merkmalsvektoren, ist davon auszugehen, dass es sich um eine duplizierte Region handelt. Blockbasierte Methoden verwenden häufig lexikografisches Sortieren, um die gefundenen Merkmalsvektoren miteinander zu vergleichen [vgl. Chr+12, S. 1842]. Hierbei wird aus den gefundenen Merkmalsvektoren eine Matrix gebildet, in welcher jeder Vektor eine Zeile einnimmt. Diese Matrix wird anschließend zeilenweise sortiert und benachbarte Vektoren werden auf ihre Ähnlichkeit zueinander untersucht [vgl. Chr+12, S. 1842].

Keypointbasierte Methoden verwenden für den Abgleich der Merkmalsvektoren häufig die Best-Bin-First-Suche, welche vom Algorithmus des k-dimensionalen Baums abgeleitet ist [vgl. Chr+12, S. 1842]. Beides sind effiziente Algorithmen zur Suche der nächsten Nachbarn eines Merkmalsvektors, also der einander am ähnlichsten Vektorpaare. Bekannte Detektionsverfahren verwenden hierbei oft die Euklidische Distanz als Ähnlichkeitsmaß zweier Merkmalsvektoren.

Um die Anzahl irrtümlich detektierter Vektorpaare zu reduzieren, folgen auf das Matching oft verschiedene **Filteroperationen** [vgl. Chr+12, S. 1842]. Eine Ursache für das Auftreten falscher Matches ist, dass benachbarte Pixel oft ähnliche Werte aufweisen. So kann es passieren, dass eine zusammenhängende Region ähnlicher Pixel als zwei duplizierte Regionen klassifiziert wird. Um dies zu verhindern, kommen auch hier wieder bestimmte Distanzmaße oder Schwellwerte zum Einsatz.

[7] SIFT = Scale-invariant Feature Transform
[8] SURF = peeded Up Robust Features

Als letzter Schritt des Detektionsprozesses ist das Ziel weiterer **Nachbearbeitungsschritte**, die gefundenen Vektorpaare und deren benachbarte Pixel weiter zu untersuchen, um mögliche Ausreißer zu entfernen und duplizierte Regionen hervorzuheben. Da kopierte Objekte vor dem Einfügen ins Bild oft skaliert oder rotiert werden, können hier mit Algorithmen wie RANSAC geometrische Transformationen zwischen der originalen und der duplizierten Bildregion einbezogen werden [vgl. Chr+12, S. 1842]. Durch Einfärbungen können die erkannten Regionen schließlich für den Betrachter sichtbar gemacht werden.

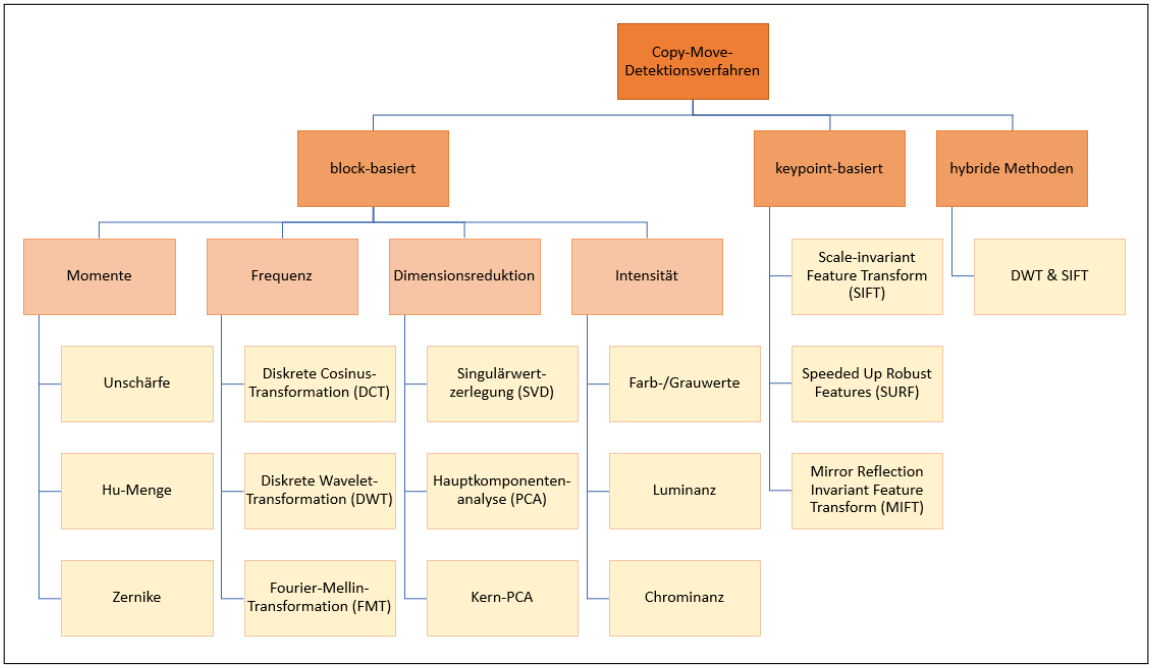


Abbildung 4.3: Einteilung der Detektionsalgorithmen für Copy-Move-Manipulationen, in Anlehnung an [vgl. Chr+12, S. 1843]

Abbildung 4.3 veranschaulicht die gerade erläuterte Einteilung der Copy-Move-Detektionsverfahren. Neben blockbasierten und keypointbasierten Verfahren, existieren mittlerweile auch hybride Methoden, welche die Vorteile beider Gruppen kombinieren, um die Detektionsgenauigkeit zu erhöhen sowie die Rechenzeit und den Speicherplatzverbrauch zu reduzieren. Während bei keypointbasierten Detektionsverfahren hauptsächlich die Algorithmen SIFT und SURF zum Einsatz kommen, können blockbasierte Verfahren weiter untergliedert werden. So unterscheiden sich die Verfahren in ihrer Verwendung spezifischer Bildmerkmale wie Farb- und Helligkeitswerte einzelner Pixel sowie bestimmter Methoden zur Transformation von Bildern. Auch statistische Momente (invariante Muster globaler und lokaler geometrischer Bildinformationen [vgl. Shu02]) und Verfahren zur Dimensionsreduktion der Merkmalsvektoren kommen dabei zum Einsatz.

Image Splicing

Wie bereits in Kapitel 3.2.2 ausführlich besprochen, handelt es sich bei dem Begriff Image Splicing um eine Manipulationsmethode, bei welcher Objekte aus einem Bild ausgeschnitten und in ein anderes Bild eingefügt werden. Ziel dieses Vorgangs ist es, Objekte oder Lücken in einem Bild zu überdecken oder aus Objekten mehrerer Bilder ein neues Bild zusammenzusetzen. Bei dieser Art der Manipulation entstehen Unstimmigkeiten zwischen verschiedenen Merkmalen des eingefügten Objektes und denen des Zielbildes. Moderne Detektionsalgorithmen nutzen solche Artefakte und Unstimmigkeiten im Zielbild zur Erkennung und Lokalisation digitaler Bildmanipulationen.

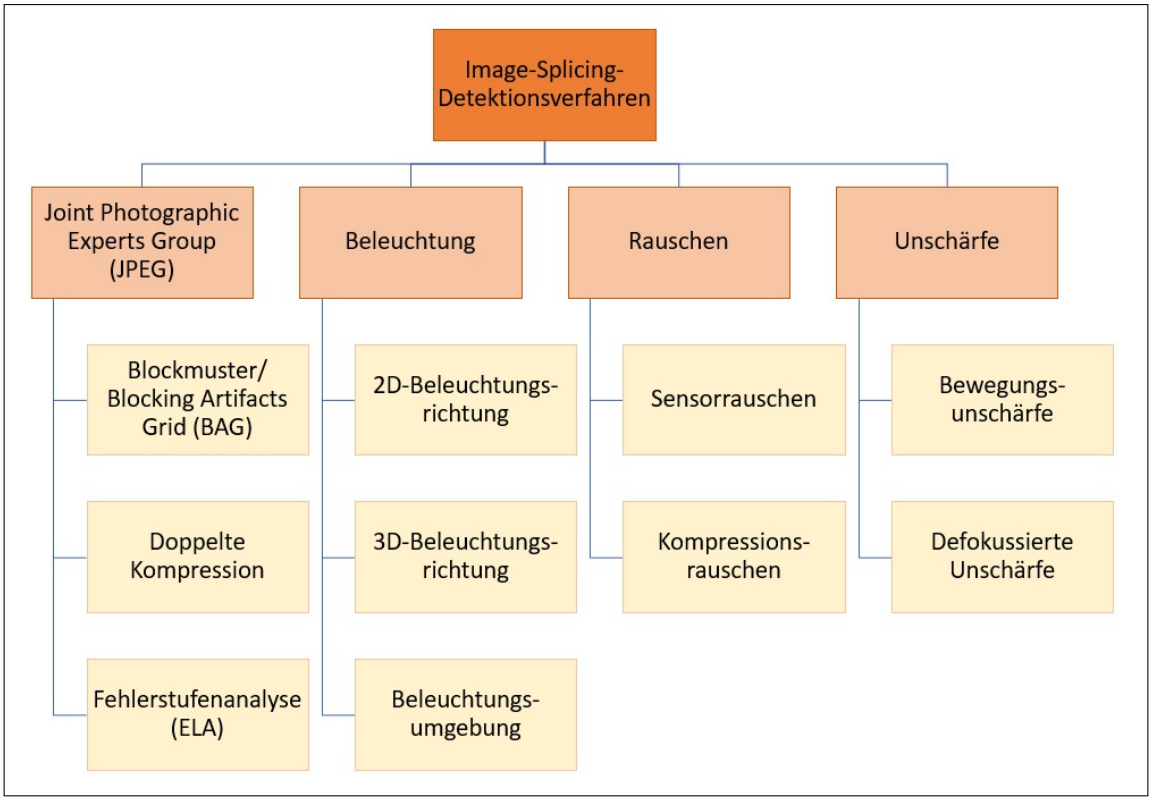


Abbildung 4.4: Einteilung der Detektionsalgorithmen für Image-Splicing-Manipulationen, Eigene Darstellung

Abbildung 4.4 zeigt eine Übersicht der hierfür am häufigsten verwendeten Bildmerkmale. Untersucht werden beispielsweise Artefakte, welche durch die verlustbehaftete Kompression mit JPEG entstehen (hierzu mehr in Kapitel 4.2.2) oder konsistentes Bildrauschen, welches durch das Einfügen eines bildfremden Objektes unterbrochen wird. Auch Beleuchtung, Schattenwurf und Unschärfe verschiedener Objekte im Bild werden auf ihre Konsistenz hin untersucht. So ist es beispielsweise möglich, dass eingefügte Objekte aus einer anderen Richtung beleuchtet wurden als der Rest des Bildes, auch dies ist anhand pixelbasierter Detektionsverfahren festzustellen.

4.2.2 *Format-/kompressionsbasiert*

Format- bzw. kompressionsbasierte Detektionsverfahren untersuchen statistische Korrelationen, welche durch die Anwendung verschiedener Kompresssionsverfahren wie JPEG[9] entstehen [vgl. Far09, S. 16]. Verlustbehaftete Kompressionsverfahren wie JPEG erschweren die Detektion von Bildmanipulationen durch pixelbasierte Verfahren, da hier Informationen während der Kompression verloren gehen oder die durch die Kompression entstehenden Muster (z.B. das JPEG-typische Blockmuster) die Korrelationen zwischen den Bildpixeln verändern. Formatbasierte Detektionsverfahren hingegen nutzen die Eigenschaften der einzelnen Kompressionsverfahren aus, um von Veränderungen kompressionstypischer Muster auf vergangene Bildmanipulationen zu schließen [vgl. Far09, S. 16].

Joint Photographic Experts Group (JPEG)

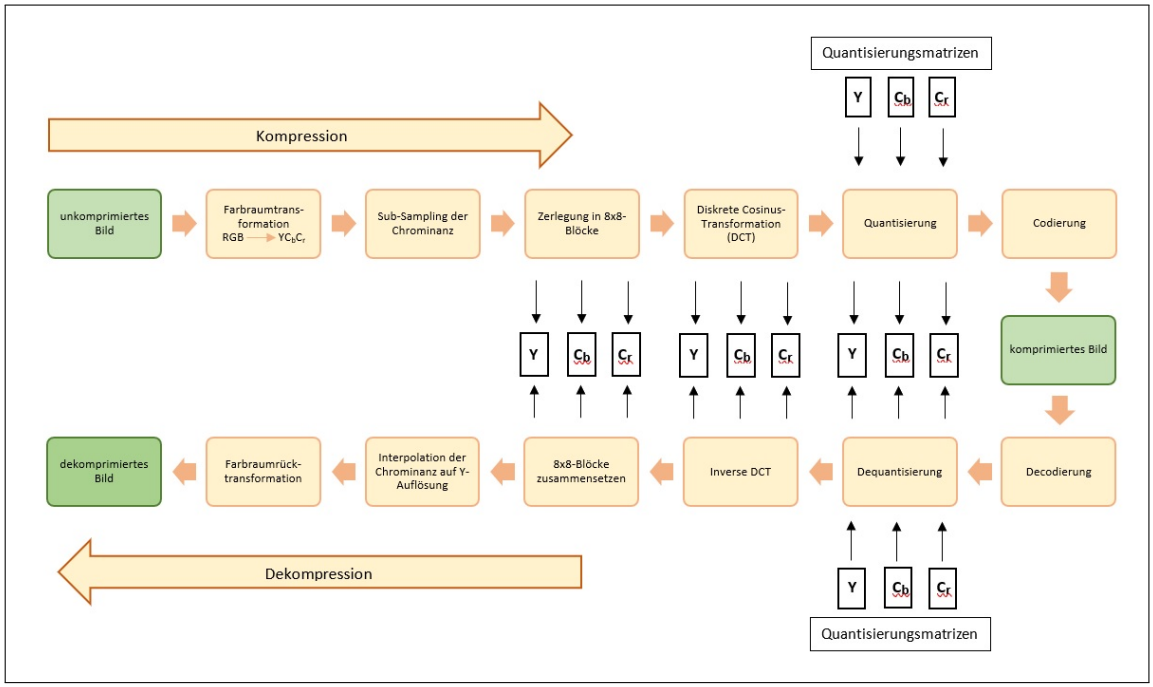


Abbildung 4.5: Allgemeine Funktionsweise des JPEG-Kompressionsverfahrens, Eigene Darstellung

Das bekannteste und am häufigsten verwendete Kompressionsverfahren ist JPEG, deswegen bauen viele der formatbasierten Detektionsverfahren auf den für JPEG typischen Eigenschaften auf. Abbildung 4.5 zeigt den allgemeinen Ablauf der JPEG-Kompression sowie der entsprechenden Dekompression. Die **Quantisierung** der DCT-Koeffizienten[10] ist bei der Kompression ausschlaggebend für den Kompressionsfaktor und damit für die Qualität des entstehenden komprimierten Bildes. Während andere Schritte des Ver-

[9] JPEG steht für das am häufigsten verwendete Bild-Kompressionsverfahren 'Joint Photographic Experts Group'

[10] DCT ist die Abkürzung der in JPEG verwendeten Diskreten Kosinustransformation

fahrens, z.B. die Transformation des Farbraums, fest vorgeschrieben sind, können die 8x8 großen Quantisierungsmatrizen an die gewünschte Kompressionsstärke angepasst werden. Hersteller von Bildaufzeichnungsgeräten verwenden für ein bestimmtes Kameramodell oft immer wieder dieselben Quantisierungsmatrizen. Bei Verdacht einer Bildmanipulation kann es demnach von Vorteil sein, die Quantisierungsmatrizen eines unbekannten komprimierten Bildes zu bestimmen, um so auf das Gerät zu schließen, mit welchem das Bild aufgenommen wurde [vgl. Far09, S. 18].

Ein weiteres Merkmal der JPEG-Kompression ist das entstehende **Blockmuster**. Dieses ist darauf zurückzuführen, dass die Diskrete Kosinustransformation sowie die Quantisierung der DCT-Koeffizienten nicht auf das Bild als Ganzes, sondern auf einzelne 8x8-Blöcke angewendet werden [vgl. Far09, S. 19], welche im Anschluss wieder zu einem Bild zusammengesetzt werden. Durch Bildmanipulationen mit Methoden wie Copy-Move kann dieses Muster unterbrochen werden, da eingefügte Objekte nicht dasselbe Blockmuster aufweisen wie das Originalbild. Um ein Bild auf solche Blockmuster zu untersuchen, werden häufig Pixelwerte innerhalb eines potentiellen Blocks sowie zwischen mehreren potentiellen Blöcken untersucht, wobei die Unterschiede der Pixelwerte innerhalb eines Blockes i.d.R. kleiner sind als die zwischen verschiedenen Blöcken [vgl. Far09, S. 19].

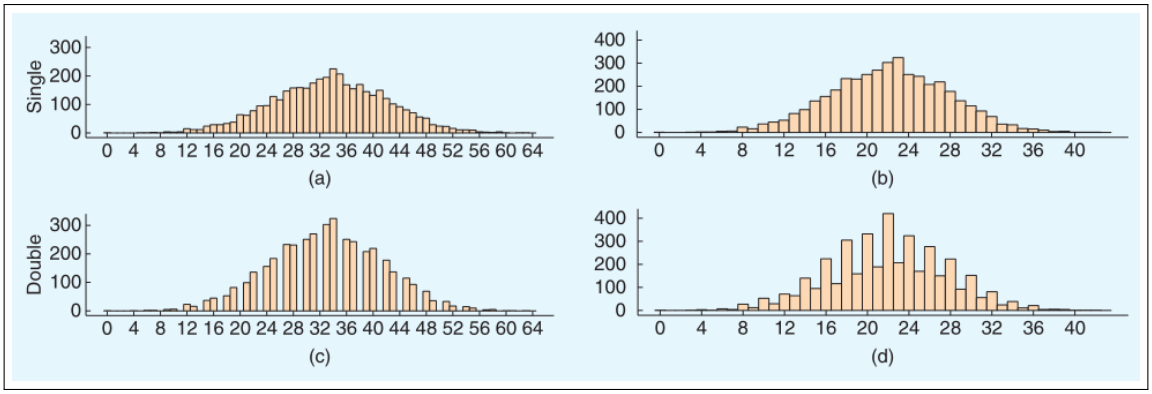


Abbildung 4.6: Einfache Quantisierung von DCT-Koeffizienten mit dem Faktor 2 (oben links) und dem Faktor 3 (oben rechts). Erneute Quantisierung mit dem Faktor 3 (unten links) und dem Faktor 2 (unten rechts) [Far09, S. 19]

Ein weiteres Phänomen, welches bei der Manipulation von JPEG-komprimierten Bildern auftaucht, ist das der **doppelten Kompression** (bzw. der doppelten Quantisierung). In der Regel ist es für die Durchführung einer digitalen Bildmanipulation notwendig, das zu manipulierende Bild in eine Bildbearbeitungssoftware zu importieren und dieses [nach der Manipulation] erneut abzuspeichern [vgl. Far09, S. 18]. Bilder, die vor der Manipulation bereits mit JPEG komprimiert wurden, werden durch das erneute Speichern einer zusätzlichen Kompression unterzogen. Da es sich bei JPEG um eine verlustbehaftete Kompression handelt, gehen umso mehr Daten verloren, je öfter die Kompression an einem Bild durchgeführt wird. Dies äußert sich insbesondere durch Artefakte in den Histogrammen der DCT-Koeffizienten nach der Quantisierung.

Abbildung 4.6 zeigt als Beispiel 4 Histogramme, welche durch unterschiedliche Quantisierung derselben DCT-Koeffizienten (ursprüngliches Intervall [0,127]) entstanden sind. Die oberen beiden Histogramme veranschaulichen die Verteilung der DCT-Koeffizienten nach einfacher Quantisierung mit dem Faktor 2 (oben links) bzw. mit dem Faktor 3 (oben rechts). Nach erneuter Quantisierung mit dem Faktor 3 (unten links) bzw. mit dem Faktor 2 (unten rechts) sind deutliche Lücken bzw. Spitzen in den unteren beiden Histogrammen zu erkennen. Lücken entstehen dann, wenn der Faktor der zweiten Quantisierung größer ist als der Faktor der ersten Quantisierung. Spitzen entstehen bei der Anwendung der Quantisierungsfaktoren in umgekehrter Reihenfolge.

4.2.3 *Kamerabasiert*

Während des Aufnahmeprozesses entstehen Artefakte im Bild, welche durch die Bauteile der Kamera (z.B. Linse oder Sensor) oder durch kamerainterne Nachbearbeitungsprozesse wie Gammakorrektur oder Weißabgleich erzeugt werden [vgl. Far09, S. 16] [vgl. QD15, S. 5]. Durch diese Artefakte ist es möglich, auf ein bestimmtes Kameramodell oder auf vergangene Bildmanipulationen zu schließen.

Chromatische Aberration

Chromatische Aberration, zusammengesetzt aus den Begriffen *chromatisch* gleich die Farbenlehre betreffend [vgl. Dud20b] und Aberration als „Abweichung der Strahlen vom idealen Bildpunkt eines optischen Instruments“ [Dud20c], bezeichnet einen Abbildungsfehler in der Optik. Durch Verwendung einer Sammellinse versuchen optische Systeme, Licht so zu brechen, dass es in einem Punkt auf einem Sensor konzentriert werden kann [vgl. Far09, S. 20]. Die elektromagnetischen Wellen des Lichts besitzen jedoch unterschiedliche Wellenlängen, welche durch die Linse unterschiedlich stark gebrochen werden und somit nicht denselben Brennpunkt besitzen.

Abbildung 4.7: Chromatische Aberration am Beispiel einer weißen Blume, deren Kontur bläulich leuchtet [Prim19]

In digitalen Bildern äußert sich diese farbliche Abweichung oft in unscharfen Konturen, welche eine andere Farbe aufweisen als die eigentliche Farbe des Objekts. Abbildung 4.7 zeigt als Beispiel eine weiße Blume, deren Rand aufgrund chromatischer Aberration bläulich leuchtet.

Während in der Optik versucht wird, die chromatische Aberration zu umgehen, beispielsweise durch die Verwendung mehrerer Linsen hintereinander, wird sie in der Filmindustrie häufig als stilistisches Mittel eingesetzt. Auch für die Detektion von Bildmanipulationen kann chromatische Aberration von Vorteil sein. Forscher fanden so u.a. heraus, dass sich die berechnete Abweichung der Farben unterschiedlicher Wellenlängen von Bild zu Bild sowie innerhalb eines Bildes unterscheidet und dass von Unterschieden in dieser Abweichung auf Manipulationsmethoden wie Copy-Move oder Image Splicing geschlossen werden kann [vgl. Far09, S. 20].

Color Filter Array (CFA)

Ein digitales RGB-Farbbild besteht i.d.R. aus drei Kanälen, wobei jeder Kanal die Helligkeitsinformationen einer der drei Farben Rot, Grün oder Blau speichert. So enthält der Rotkanal beispielsweise die Helligkeitswerte der Rotanteile des Bildes. Die meisten Digitalkameras enthalten jedoch lediglich einen CCD- oder CMOS-Sensor[11] [vgl. Far09, S. 20], welcher Helligkeitsinformationen aufnehmen, jedoch nicht zwischen den drei Farbkanälen unterscheiden kann. Um dennoch die Information aller drei Farben aufzunehmen, kommt ein Color Filter Array (CFA) zum Einsatz. Dieser Filter sorgt dafür, dass jede Sensorzelle der Kamera nur die Werte einer bestimmten Farbe aufnimmt. Durch anschließende Interpolation der Farbwerte benachbarter Pixel werden die fehlenden Farbwerte der Pixel im digitalen Bild ergänzt.

Die Interpolation der Farbwerte erzeugt Korrelationen zwischen den interpolierten Pixeln und deren Nachbarpixeln. Da Color Filter Arrays in der Anordnung ihrer Farbfilter meist periodisch aufgebaut sind[12], sind die entstehenden Korrelationen ebenfalls periodisch [vgl. Far09, S. 20]. Durch eine Manipulation des Bildes werden diese Korrelationen jedoch verändert oder zerstört. Bestehende Forschungen versuchen deshalb, Korrelationen zwischen Bildpixeln zu finden und durch Unstimmigkeiten in diesen Korrelationen Hinweise auf eventuelle Manipulationen zu finden.

4.2.4 *Physikbasiert*

Physikbasierte Detektionsverfahren untersuchen Anomalien in der dreidimensionalen Interaktion zwischen physikalischen Objekten, Lichtquellen und der Kamera [vgl. Far09, S. 16]. Solche Unstimmigkeiten entstehen durch eine fehlerhafte Abbildung der physi-

[11] CCD = Charge-coupled Device, CMOS = Complementary Metal-oxide-semiconductor

[12] z.B folgt in der Bayer-Matrix auf einen grünen immer ein blauer (ungerade Zeile) und auf einen roten immer ein grüner Pixel (gerade Zeile)

kalischen Umgebung, die während des Bildaufnahmeprozesses vorliegt.

Der Fokus bisheriger Forschungen liegt dabei insbesondere auf der Untersuchung von Unstimmigkeiten in der Beleuchtung, welche häufig nach der Verwendung von Copy-Move oder Image-Splicing auftreten. Ziel dieser Forschungen ist die Identifikation einer oder mehrerer Lichtquellen eines Bildes und die darauf folgende Untersuchung, ob die Beleuchtung bestimmter Objekte den physikalischen Zusammenhängen zu den gefundenen Lichtquellen entspricht. Zur Berechnung der Positionen der Lichtquellen verwenden verschiedene Ansätze u.a. die 2D-Normalenvektoren auf der Kontur einzelner Objekte oder 3D-Modelle des menschlichen Auges und die Lichtreflektion in diesem [vgl. Far09, S. 22 f.]. Da diese Ansätze häufig mit der Annahme einer einzelnen Lichtquelle und der isotropen Reflektion des Lichtes arbeiten, fehlt jedoch oft noch der Bezug zur Praxis.

4.2.5 *Geometriebasiert*

Geometriebasierte Detektionsverfahren untersuchen die räumlichen Zusammenhänge von Objekten sowie die Maße von Objekten in der realen Welt und deren relative Position zur Kamera [vgl. Far09, S. 17]. Zum Einsatz kommen hier insbesondere perspektivische Einschränkungen sowie kamerabedingte Parameter wie Brennweite und Seitenverhältnis [vgl. QD15, S. 6].

Beispielsweise arbeiten verschiedene Ansätze mit der Verschiebung des Hauptpunktes durch in das Bild eingefügte Objekte [vgl. Far09, S. 23] oder versuchen, reale Maße nicht frontal fotografierter Objekte zu berechnen, um abzugleichen, ob eingefügte Objekte das Verhältnis der metrischen Maße im Bild stören [vgl. Far09, S. 24].

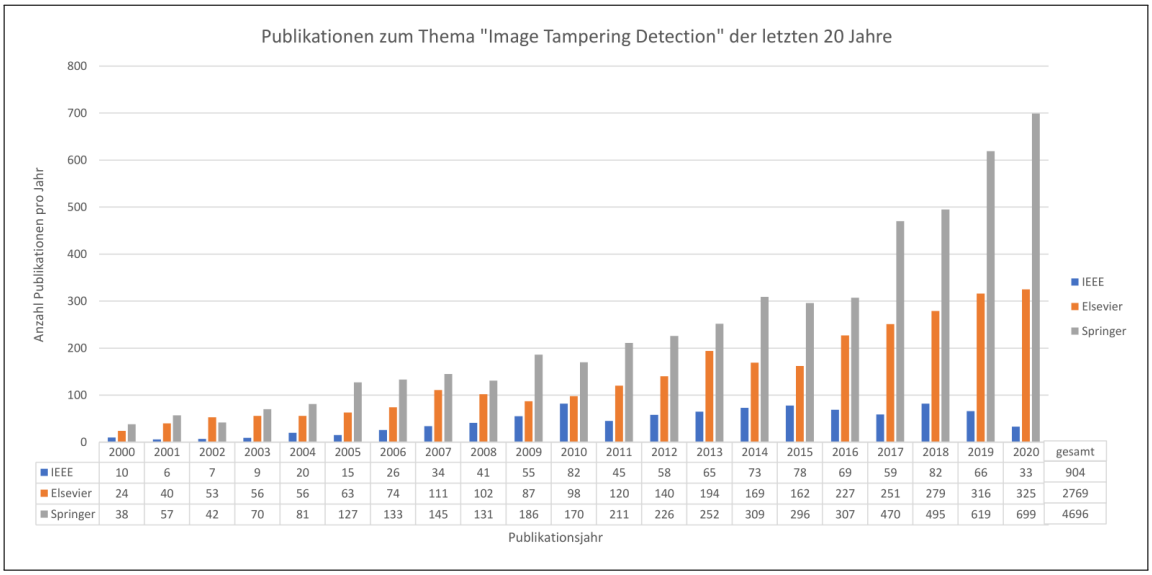
4.3 Forschungsstand

4.3.1 Vergangene Entwicklungen

Alle in Kapitel 4.2 behandelten Detektionsverfahren können als Hinweis für eine potentiell durchgeführte Bildmanipulation dienen, sie sind jedoch nicht zwingend ein eindeutiger Beweis für diese. Beispielsweise kann eine doppelte JPEG-Kompression auch versehentlich durch ein erneutes Abspeichern eines digitalen Bildes entstehen. Vergangene Forschungen haben diese Hinweise jedoch aufgegriffen und als Anhaltspunkte zur Entwicklung neuer Verfahren für die Detektion von Bildmanipulationen verwendet.

Während die Forschung auf dem Gebiet der Bildverarbeitung bereits seit der Entwicklung des ersten digitalen Bildes vorangetrieben wurde, um beispielsweise die Qualität digitaler Bilder zu verbessern oder Informationen aus diesen zu extrahieren, wurden

erste Publikationen zur Detektion digitaler Bildmanipulationen erst in den 1990er bis 2000er Jahren veröffentlicht. Grund dafür war unter anderem die fortschreitende Entwicklung von Bildverarbeitungssystemen.



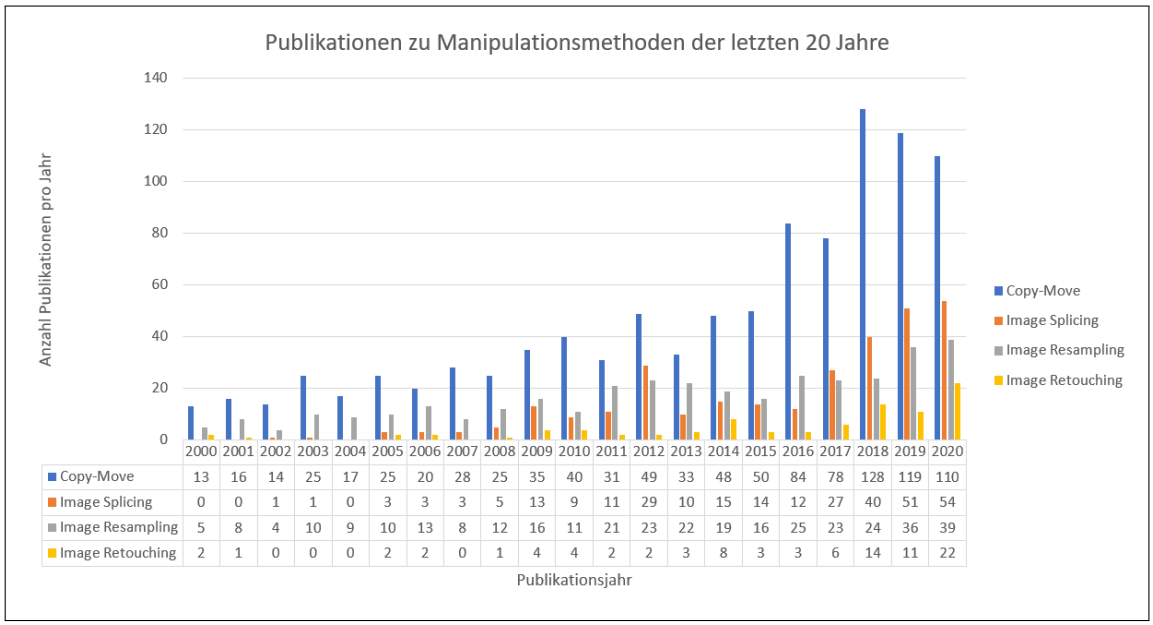
	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	gesamt
IEEE	10	6	7	9	20	15	26	34	41	55	82	45	58	65	73	78	69	59	82	66	33	904
Elsevier	24	40	53	56	56	63	74	111	102	87	98	120	140	194	169	162	227	251	279	316	325	2769
Springer	38	57	42	70	81	127	133	145	131	186	170	211	226	252	309	296	307	470	495	619	699	4696

Abbildung 4.8: Entwicklung der Anzahl wissenschaftlicher Publikationen zum Thema Bildmanipulationserkennung von 2000 bis 2020 [IEEE20; Spr20a; Els20]

Abbildung 4.8 zeigt die Entwicklung der Forschung zum Thema Bildmanipulationserkennung anhand eines Diagramms, welches die Anzahl veröffentlichter Publikationen dieses Themengebietes von 2000 bis heute darstellt. Die Erstellung des Diagramms beruht auf den Ergebnissen zur englisch-sprachigen Suchanfrage ‘Image Tampering Detection‘‘ der drei Online-Recherchedatenbanken IEEE Xplore [vgl. IEEE20], Springer Link [vgl. Spr20a] und ScienceDirect [vgl. Els20]. Anhand des Diagramms lässt sich erkennen, dass die Anzahl der Publikationen im Bereich der Bildmanipulationserkennung innerhalb der letzten 20 Jahre um ein Vielfaches angestiegen ist. So wurden beispielsweise im Jahr 2009 insgesamt 328 Publikationen auf den genannten Online-Datenbanken hochgeladen, während sich diese Anzahl bis zum Jahr 2019 bereits auf 1001 Publikationen verdreifachte.

Wie bereits erwähnt, handelt es sich bei Copy-Move, Image Splicing, Image Resampling und Image Retouching um die am häufigsten verwendeten Manipulationsmethoden, um die Qualität eines digitalen Bildes zu verbessern oder dessen Bildinhalt nach Belieben zu verändern. Bei der Entwicklung von Detektionsverfahren für digitale Bildmanipulationen stand in den vergangenen Jahren insbesondere Copy-Move im Vordergrund der Untersuchungen. Ein Grund dafür ist, dass die Lokalisation duplizierter Regionen innerhalb eines Bildes einfacher zu implementieren ist als die Detektion von durchgeführten Nachbearbeitungsoperationen wie Skalierung oder von Unstimmigkeiten in der Zusammenwirkung einzelner Bildpixel [vgl. Arm+20, S. 2]. Diese Beobachtung wird veranschaulicht durch das in Abbildung 4.9 dargestellte Diagramm. Es zeigt die Anzahl von wissenschaftlichen Publikationen der letzten 20 Jahre, welche speziell zu den ge-

nannten vier Manipulationsmethoden veröffentlicht wurden. Das Diagramm wurde auf Grundlage der Suchanfragen ‘Copy-Move‘ [vgl. Spr20b], ‘Image Splicing‘ [vgl. Spr20c], ‘Image Resampling‘ [vgl. Spr20d] und ‘Image Retouching‘ [vgl. Spr20e] in der Online-Datenbank Springer Link erstellt. Zu erkennen ist neben der deutlich gestiegenen Anzahl an Publikationen von 2000 bis 2020, dass es sich bei dem Großteil der veröffentlichten Werke um Forschungen zu Copy-Move-Bildmanipulationen handelt.



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Copy-Move	13	16	14	25	17	25	20	28	25	35	40	31	49	33	48	50	84	78	128	119	110
Image Splicing	0	0	1	1	0	3	3	3	5	13	9	11	29	10	15	14	12	27	40	51	54
Image Resampling	5	8	4	10	9	10	13	8	12	16	11	21	23	22	19	16	25	23	24	36	39
Image Retouching	2	1	0	0	0	2	2	0	1	4	4	2	2	3	8	3	3	6	14	11	22

Abbildung 4.9: Entwicklung der Anzahl wissenschaftlicher Publikationen zum Thema Bildmanipulationserkennung von 2000 bis 2020, detailliert aufgezeigt ist die Anzahl von Publikationen zu verschiedenen Bildmanipulationsmethoden [Spr20b; Spr20c; Spr20d; Spr20e]

Da die geschichtliche Entwicklung verschiedener Detektionsverfahren bereits durch andere Autoren detailliert erläutert wurde (beispielsweise durch *Birajdar und Mankar* [vgl. BM13]), soll diese im Folgenden lediglich als Überblick zusammengefasst werden.

Erste Verfahren zur Erkennung digitaler Bildmanipulationen beruhten zumeist auf der Berechnung mathematischer Zusammenhänge zwischen einzelnen Bildpixeln oder Pixelblöcken. Neben der Verwendung von Grauwerten und Farbwerten unterschiedlicher Farbräume (z.B. RGB- oder YC_bC_r-Farbraum) kamen so insbesondere verschiedene Transformationen der Signalverarbeitung zum Einsatz. Beispiele hierfür sind die Diskrete Kosinustransformation und die Diskrete Wavelet-Transformation, welche vor allem zur Reduktion der zu verarbeitenden Bildmerkmale verwendet werden. Bereits einer der ersten Versuche, duplizierte Regionen in einem Bild zu finden, berechnete die DCT-Koeffizienten innerhalb von 8x8 großen Bildblöcken und nutzte diese Koeffizienten, um einzelne Bildbereiche miteinander zu vergleichen [vgl. FSL03]. Weitere Forschungen untersuchten u.a. durch Bildverarbeitung entstandene Artefakte wie Interpolations- und JPEG-Blockmuster sowie geometrische und physikalische Beziehungen zwischen realen Objekten, die in einem Bild festgehalten wurden.

Neben blockbasierten Verfahren, welche über mehrere Jahre hinweg von verschiedenen Autoren bezüglich ihrer Genauigkeit und ihres Rechenaufwands verbessert wurden, fanden auch keypointbasierte Verfahren Verwendung bei der Detektion digitaler Bildmanipulationen. Das 1999 durch David G. Lowe für die Objekterkennung entwickelte SIFT-Verfahren [vgl. Low99] dient seit Anfang der 2000er Jahre als Grundlage verschiedener Detektionsverfahren, welche Bildregionen anhand relevanter Punkte im Bild (engl. Keypoints) vergleichen. Aufbauend auf SIFT entstanden so weitere performante Verfahren wie SURF oder das spiegelungsinvariante Verfahren MIFT[13].

Nachdem die Funktionalität der Detektionsalgorithmen gewährleistet war, galt es als neues Ziel, die bekannten Verfahren robuster zu gestalten gegen mögliche Nachbearbeitungsschritte wie verlustbehaftete Kompression, Addition von Bildrauschen, Unschärfe oder einer Kombination verschiedener Operationen [vgl. BM13, S. 230]. Auch die Verbesserung der Klassifikationsergebnisse sowie die Dimensionsreduktion der verwendeten Merkmalsvektoren zur Reduktion der Komplexität rückten als Ziele in den Vordergrund der weiteren Forschung.

4.3.2 Aktuelle Forschungen

Mithilfe der gerade vorgestellten traditionellen Herangehensweisen ist es möglich, bestimmte Manipulationsmethoden anhand der für diese Methode spezifischen Bildmerkmale zu erkennen [vgl. BG20, S. 571]. So kommen beispielsweise bei der Verwendung von blockbasierten und keypointbasierten Copy-Move-Detektionsmethoden verschiedene Algorithmen zum Einsatz, mit deren Hilfe Merkmale wie Kanten oder Keypoints eines Bildes extrahiert werden können [vgl. BG20, S. 572]. Für die Extraktion komplexerer Bildmerkmale, z.B. durch Objekterkennung oder Detektion semantischer Zusammenhänge im Bild, reichen traditionelle Verfahren jedoch oft nicht mehr aus. Durch die Weiterentwicklung von Grafikkarten und Techniken des Machinellen Lernens ist es mittlerweile jedoch möglich, solche komplexen Bildmerkmale zu finden und aufbauend auf diesen neue Detektionsverfahren für digitale Bildmanipulationen zu entwickeln.

Innerhalb der letzten fünf bis zehn Jahre sind so neue Detektionsverfahren entwickelt worden, welche sich auf den Einsatz von künstlichen neuronalen Netzen und Deep-Learning-Methoden stützen. Diese Methoden sollen insbesondere dazu dienen, Informationen aus dem Inhalt des Bildes zu extrahieren, um so die durch Manipulation entstandenen Zusammenhänge innerhalb eines Bildes zu erkennen und das daraus erlernte Wissen auf weitere Bilder anzuwenden. Das allgemeine Ziel ist hier die Verbesserung der Klassifikationsergebnisse bei der Detektion manipulierter Bilder sowie die Reduktion der für die Detektion benötigten Rechenzeit. Durch den Einsatz neuer Methoden des maschinellen Lernens ist es so auch möglich, verschiedene Arten von Bildmanipulationen zu erkennen und zu lokalisieren.

[13] MIFT steht für Mirror Reflection Invariant Feature Transform

Ein Beispiel für den Einsatz neuronaler Netze in der Bildmanipulationserkennung wurde 2017 von Amerini et al. veröffentlicht [Ame+17]. Das Verfahren beschreibt eine Methode zur Erkennung von Bildregionen, welche aus einem anderen Bild kopiert und in das zu untersuchende Bild eingefügt wurden. Für die Detektion und Lokalisation solcher Regionen verwenden die Autoren sogenannte Convolutional Neural Networks (CNN)[14], die auf die Erkennung einfach oder doppelt JPEG-komprimierter Bildregionen trainiert werden [vgl. Ame+17, S. 1865]. Hierfür werden zwei verschiedene CNNs kombiniert, sodass sowohl Informationen des Orts- als auch des Frequenzraums bei der Detektion Anwendung finden und die Schwächen rein DCT-basierter Methoden ausgeglichen werden [vgl. Ame+17, S. 1869]. Das Ergebnis des Verfahrens ist eine erfolgreiche Detektion unkomprimierter und mittels JPEG komprimierter Bildregionen sowie die zusätzliche Berechnung des verwendeten Kompressionsfaktors.

Neben den Vorteilen, die der Einsatz neuronaler Netze für die Bildmanipulationserkennung hervorbringt, existieren jedoch weiterhin Probleme, die diesen Einsatz erschweren. So erfordert beispielsweise das Trainieren eines Deep-Learning-Netzwerkes spezifisches Fachwissen sowie eine hohe Rechenzeit und einen ausreichend großen Datensatz mit unterschiedlich manipulierten Bildern [vgl. BG20, S. 574]. Nichtsdestotrotz bietet sich mit der zukünftigen Weiterentwicklung neuronaler Netze gleichfalls die Möglichkeit, die bestehenden Verfahren der Bildmanipulationserkennung zu erweitern und zu verbessern.

[14] Convolutional Neural Networks (deutsch etwa: faltende neuronale Netze) sind künstliche neuronale Netze, die aufgrund ihres Aufbaus besonders für die Verarbeitung von Bildern geeignet sind.

5 Implementierung der Software

Die vorangegangenen Kapitel boten einen Einblick in die Grundlagen und Methoden digitaler Bildmanipulationen sowie eine grobe Gliederung bestehender Verfahren zur Detektion solcher Manipulationen. In diesem Kapitel soll nun die Auswahl jener Detektionsverfahren erläutert werden, welche in die eigens implementierte Software integriert wurden. Dabei wird insbesondere auf die Funktionsweise einzelner Verfahren sowie deren Eignung für die Implementierung eingegangen. Anschließend folgt eine ausführliche Beschreibung des Aufbaus und der Funktionsweise der implementierten Software. Als letzter Punkt des Kapitels sollen die Wahl der verwendeten Testdatensätze sowie der Ablauf der Klassifierung erläutert werden.

5.1 Auswahl geeigneter Detektionsverfahren

Für die geplante Software sollen Detektoren für Bildmanipulationen implementiert werden, welche die Authentizität digitaler Bilder nach verschiedenen Algorithmen binär klassifizieren. Anschließend sollen verschiedene Ansätze miteinander kombiniert werden, um die Genauigkeit der Klassifikation zu erhöhen und die rechnerische Komplexität der Software zu reduzieren.

Wie bereits in Kapitel 4.3 ausführlich erläutert, konzentriert sich eine Vielzahl etablierter Verfahren auf die Detektion von Bildmanipulationen, welche anhand der Methode Copy-Move erstellt wurden. Ein Grund hierfür ist die einfache Anwendbarkeit der Methode, welche es auch Laien ermöglicht, digital manipulierte Bilder realistisch wirken zu lassen. Ein Großteil der sich im Umlauf befindenden Bildmanipulationen weist demnach diese Art der Manipulation auf. Hierauf aufbauend soll nun eine Software implementiert werden, welche verschiedene block- und keypointbasierte Ansätze von Copy-Move-Detektionsverfahren kombiniert.

Um die Qualität der ausgewählten Verfahren und der selbst implementierten Software vergleichen zu können, werden Erstere zunächst einzeln implementiert. Anschließend werden die verschiedenen Ansätze miteinander kombiniert. Neben der Genauigkeit bei der Klassifikation soll des Weiteren darauf geachtet werden, dass das finale Detektionsverfahren robust ist gegen Nachbearbeitungsprozesse wie Rotation oder Skalierung. Die Detektion duplizierter Regionen soll demnach auch nach Anwendung solcher Operationen möglich sein.

5.1.1 Blockbasierte Detektionsverfahren

Wie bereits erwähnt, wird das zu untersuchende Bild bei blockbasierten Detektionsverfahren in Blöcke unterteilt, welche anhand ihres Inhalts miteinander verglichen werden. Bekannte Verfahren verwenden für diese Blockzerlegung unterschiedliche Blockgrößen und -formen sowie verschiedene Iterationsmuster. Ein oft verwendetes Beispiel ist hierbei die Einteilung in quadratische, sich überlappende 8x8-Blöcke. Auch sich nicht überlappende, kreisförmige oder unregelmäßig geformte Blöcke sind denkbar. Vorteile von blockbasierten Verfahren liegen vor allem in der Genauigkeit der Detektion und Lokalisation duplizierter Regionen. Probleme treten hingegen auf, wenn auf manipulierte Regionen weitere Nachbearbeitungsschritte wie Rotation angewendet werden. Die folgenden Verfahren implementieren diese Blockzerlegung und die darauf folgende Extraktion von Blockmerkmalen auf unterschiedliche Weise. Ebenfalls werden Ansätze präsentiert, um die genannten Probleme zu beheben.

Mahmood et al. 2018

Die Autoren Toqueer Mahmood et al. veröffentlichten 2018 ein Verfahren zur Detektion und Lokalisation von Copy-Move-Bildmanipulationen [vgl. Mah+18]. Ziel der Publikation war ein translations-invariantes Programm, welches zuverlässig und unter Einsparung von Rechenzeit duplizierte Regionen innerhalb eines Bildes erkennen sollte.

Da das Luminanzbild des YC_bC_r-Farbraums mehr räumliche Informationen enthält als andere Farbräume [vgl. Mah+18, S. 204], wird das zu untersuchende Bild zu Beginn des Verfahrens vom RGB- in den YC_bC_r-Farbraum konvertiert. Die anschließende Extraktion von Bildmerkmalen findet ausschließlich auf dem Luminanzbild statt. Um die zu verarbeitenden Bildinformationen zu reduzieren, verwenden einige Detektionsverfahren die Diskrete Wavelet-Transformation (DWT), aus deren Koeffizienten Merkmalsvektoren gebildet werden können. Da es sich bei der DWT jedoch um ein translations-variantes Verfahren handelt, ändern sich die Koeffizienten und damit auch die Merkmalsvektoren schon bei leichten Verschiebungen innerhalb eines Bildes, wodurch die Detektion duplizierter Blöcke kompliziert wird. Um dieses Problem zu umgehen, arbeiten Mahmood et al. mit der translations-invarianten Stationären Wavelet-Transformation (SWT). Bei dieser findet im Gegensatz zur DWT nach der Anwendung von Hoch- und Tiefpassfiltern keine Dezimierung der Bildgröße statt, die entstehenden Teilbänder besitzen die gleichen Maße wie das Ausgangsbild. Nach Anwendung der SWT wird das Approximations-Teilband (niedrige Frequenzen) in sich überlappende 8x8-Blöcke unterteilt, wobei jeder Block nur eine Pixelzeile von seinem Vorgänger abweicht. Bei einer gegebenen Bildgröße von NxM ergibt sich so eine Anzahl von (N - 8 + 1) x (M - 8 + 1) Blöcken.

Zur Detektion duplizierter Regionen wird für jeden Block ein Merkmalsvektor gebildet, anhand dessen der Block identifiziert und mit anderen Blöcken verglichen werden kann. Alle Koeffizienten eines 8x8-Blockes ergäben jedoch einen 64-stelligen Merkmalsvektor. Um die Dimension des Merkmalsvektors zu verringern und so das spätere Sortieren

und Vergleichen der Blöcke zu erleichtern, verwendet das Verfahren von Mahmood et al. die Diskrete Kosinustransformation. Ein Merkmal der DCT ist es, dass sich die wichtigsten Informationen in den niedrig frequenten DCT-Koeffizienten konzentrieren [vgl. Mah+18, S. 205]. Nach Anwendung der DCT auf die 8x8-Blöcke werden nun die ersten 6 Koeffizienten eines Blockes anhand eines Zick-Zack-Scans extrahiert, diese bilden den 6-stelligen Merkmalsvektor des Blockes.

Der bisher beschriebene Ablauf diente lediglich der Vorverarbeitung des Bildes, um den nun folgenden Blockvergleich zu vereinfachen. Um Bildblöcke miteinander zu vergleichen und so ähnliche Blöcke im Bild zu finden, werden die erstellten Merkmalsvektoren der Blöcke zusammen in einer Liste abgespeichert. Da es viel Rechenzeit in Anspruch nehmen würde, jeden Block des Bildes mit jedem anderen Block zu vergleichen, werden die Merkmalsvektoren der Blöcke innerhalb der Liste lexikografisch sortiert, sodass einander ähnliche Blöcke nun direkt nebeneinander gespeichert sind. Dies ermöglicht es, immer nur zwei benachbarte Blöcke zu vergleichen und dennoch korrekte Ergebnisse zu erhalten.

Im letzten Schritt des Verfahrens werden nun jeweils zwei Blöcke miteinander verglichen. Eine Annahme bestehender Copy-Move-Detektionsverfahren ist, dass duplizierte Regionen im Bild nicht direkt nebeneinander liegen und sich nicht gegenseitig überschneiden [vgl. Mah+18, S. 206]. Da das Verfahren jedoch mit sich überlappenden Blöcken arbeitet, muss zuerst sichergestellt werden, dass zwei räumlich nebeneinander liegende Blöcke nicht als relevantes Blockpaar detektiert werden. Um dies zu gewährleisten, werden nur solche Blockpaare weiter untersucht, welche die Bedingung aus Formel 5.1 erfüllen. Die Blockdistanz zweier Blöcke, berechnet anhand der Koordinaten (x,y) der linken oberen Ecke der Blöcke, muss größer oder gleich dem Schwellwert B_{th} sein.

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq B_{th} \qquad B_{th} = 40 \tag{5.1}$$

Für die Blockpaare, welche die Bedingung in Formel 5.1 erfüllen, wird abschließend die Blockähnlichkeit ihrer Merkmalsvektoren $[f_i^1, f_i^2, f_i^3, f_i^4, f_i^5, f_i^6]$ und $[f_j^1, f_j^2, f_j^3, f_j^4, f_j^5, f_j^6]$ berechnet. Dies erfolgt durch die Berechnung der Euklidischen Distanz der Merkmalsvektoren nach Formel 5.2.

$$\sqrt{\sum_{k=1}^{6} (f_i^k - f_j^k)^2} \geq S_{th} \qquad S_{th} = 0.0005 \tag{5.2}$$

Ist auch die berechnete Blockähnlichkeit eines Blockpaars größer oder gleich einem vordefinierten Schwellwert S_{th}, so gilt das Paar als relevant. Diejenigen Blockpaare,

welche beide Bedingungen erfüllen, werden schließlich im Bild farbig markiert, um die duplizierten Regionen hervorzuheben.

Lin et al. 2009

Ein weiteres blockbasiertes Detektionsverfahren wurde 2009 von den Autoren Hwei-Jen Lin et al. veröffentlicht [vgl. LWK09]. Ähnlich zu Mahmood et al. wird auch hier das Bild in sich überlappende Blöcke unterteilt, allerdings beträgt die Blockgröße hier 16x16 anstatt 8x8 Pixeln. Statt mit Transformationen wie DWT oder DCT arbeiten die Autoren mit den Intensitäten der Pixelwerte einzelner Blöcke. Durch verschiedene mathematische Berechnungen wird so aus den Pixelintensitäten eines Blockes ein 9-stelliger Merkmalsvektor gebildet. Anschließend werden die Merkmalsvektoren der Blöcke miteinander verglichen, um ähnliche Blöcke innerhalb des Bildes zu finden.

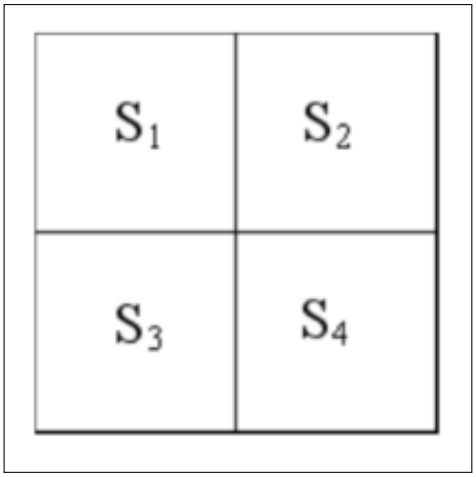


Abbildung 5.1: Unterteilung eines Bildblockes in 4 Unterblöcke nach Lin et al. 2009 [LWK09, S. 191]

Zu Beginn des Verfahrens wird das Bild in 16x16 Pixel große Blöcke unterteilt. Anschließend wird jeder einzelne Block erneut in 4 gleichgroße Unterblöcke S_1, S_2, S_3 und S_4 unterteilt (siehe Abb. 5.1). Der durchschnittliche Pixelwert des gesamten Blocks bildet die erste Stelle des 9-stelligen Merkmalsvektors. Für die nächsten 4 Stellen des Vektors wird jeweils der Anteil der Pixelwerte eines Unterblocks vom durchschnittlichen Pixelwert des gesamten Blocks berechnet. Die jeweiligen Differenzen zwischen der Durchschnittsintensität eines Unterblocks und der durchschnittlichen Intensität des gesamten Blocks bilden die letzten 4 Stellen des Merkmalsvektors. Die vollständige Berechnung des Merkmalsvektors $[f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9]$ für einen Block erfolgt durch Formel 5.3.

$$f_i = \begin{cases} Ave(B) & \text{für } i = 1 \\ Ave(S_{i-1} & \text{für } 2 \leq i \leq 5 \\ Ave(S_{i-5} - Ave(B) & \text{für } 6 \leq i \leq 9 \end{cases} \tag{5.3}$$

Nach Erstellung der Merkmalsvektoren werden diese auf ganzzahlige Werte zwischen 0 und 255 normalisiert und anschließend lexikografisch sortiert. Dies vereinfacht den darauffolgenden Vergleich der Merkmalsvektoren untereinander, durch den ähnliche Blöcke und somit duplizierte Regionen gefunden werden können. Die Normalisierung der Merkmalsvektoren erfolgt durch Formel 5.4, wobei $\lfloor . \rfloor$ eine Abrundungsfunktion symbolisiert. Die Variablen m_1 und m_2 stehen für das Maximum und das Minimum der letzten 4 Stellen des Merkmalsvektors.

$$x_i = \begin{cases} \lfloor f_i \rfloor & \text{für } i = 1 \\ \lfloor 255 \cdot f_i \rfloor & \text{für } 2 \leq i \leq 5 \\ \lfloor 255 \cdot \frac{f_i - m_2}{m_1 - m_2} \rfloor & \text{für } 6 \leq i \leq 9 \end{cases} \tag{5.4}$$

Nach dem Sortieren der Merkmalsvektoren werden diese nun paarweise miteinander verglichen. Stimmen zwei Vektoren in allen 9 Stellen überein, werden diese vorläufig als relevantes Blockpaar angesehen. Um aus der Liste der so erkannten Blockpaare falsche Zuordnungen zu entfernen, wird für jedes Paar ein Verschiebungsvektor berechnet. Dieser beschreibt den Richtungsvektor zwischen der linken oberen Ecke des ersten und der des zweiten Blockes. Da davon auszugehen ist, dass alle Blockpaare innerhalb einer duplizierten Region denselben Verschiebungsvektor besitzen, wird der am häufigsten auftretende Vektor ermittelt. Nun werden nur noch diejenigen Blockpaare als relevant betrachtet, deren Verschiebungsvektor dem gerade ermittelten Vektor entspricht. Weiterhin sollte die Länge dieses Vektors, der Abstand der Blöcke voneinander, einen gewissen Schwellwert überschreiten, um nah beieinander liegende Blöcke nicht als relevante Blockpaare zu betrachten. Die Blockpaare, welche nach diesem Aussortieren übrig bleiben, werden im Bild farblich markiert und kennzeichnen damit die duplizierten Regionen.

Da sich der Verschiebungsvektor zwischen zwei Blöcken ändert, wenn vor dem Einfügen der kopierten Region eine Rotation stattfindet, muss hier eine Anpassung des Verfahrens stattfinden. Um auch rotierte Regionen zu lokalisieren, wird deshalb bei der Suche nach Blockpaaren ebenso auf rotierten Versionen des Ausgangsbildes gearbeitet. Das Verfahren von Lin et al. beschreibt demnach einen Algorithmus zur schnellen Detektion von Copy-Move-Bildmanipulationen, welcher zudem robust ist gegen Rotationen im Bild.

Cao et al. 2012

Ein weiteres Verfahren zur Detektion von Copy-Move-Bildmanipulationen wurde 2012 von Yanjun Cao et al. veröffentlicht [vgl. Cao+12]. Ebenso wie bei anderen block-basierten Verfahren wird auch hier das Ausgangsbild (hier als Graustufenbild) in 8x8 Pixel große Blöcke unterteilt, welche miteinander verglichen werden sollen. Nach dieser Blockzerlegung des Bildes werden anschließend für jeden Block die entsprechenden DCT-Koeffizienten

berechnet. Ähnlich zu Lin et al. wird auch hier jeder Block weiter unterteilt. Während Lin et al. jedoch jeden Block in 4 gleichgroße Unterblöcke unterteilen, verwenden Cao et al. eine andere Methode, wichtige Merkmale eines Blockes zu extrahieren. So erhält jeder Block einen Kreis, welcher den Block exakt ausfüllt, der Durchmesser des Kreises entspricht hierbei der Größe des Blockes.

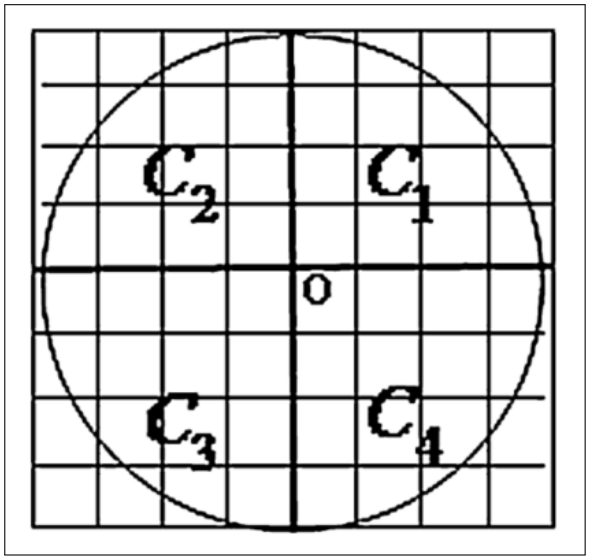


Abbildung 5.2: Unterteilung eines Bildblockes in 4 Kreisabschnitte nach Cao et al. [Cao+12, S. 35]

Das Verfahren von Cao et al. arbeitet mit 4-stelligen Merkmalsvektoren, welche die Blöcke des Bildes repräsentieren. Zur Erstellung eines Merkmalsvektors $[v_1, v_2, v_3, v_4]$ wird der eingezeichnete Kreis in 4 gleichgroße Kreissektoren C_1, C_2, C_3 und C_4 unterteilt (siehe Abb. 5.2). Jede Stelle v_i des Vektors ergibt sich aus der Summe der Pixelwerte innerhalb eines Kreissektors geteilt durch dessen Fläche. Die Berechnung erfolgt anhand der Formel 5.5:

$$v_i = \frac{\sum f(x,y)}{c\_area_i} \qquad f(x,y) \in c\_area_i, i = 1,2,3,4 \qquad c\_area_i = \frac{\Pi \cdot r^2}{4} \tag{5.5}$$

Die berechneten Merkmalsvektoren der Blöcke werden zusammen in einer Liste gespeichert und lexikografisch sortiert, sodass zwei ähnliche Blöcke nebeneinander in der Liste stehen. Um duplizierte Regionen im Bild zu finden, werden nun immer zwei in der Liste benachbarte Blöcke miteinander verglichen. Dies erfolgt durch die Berechnung der Euklidischen Distanz zwischen den 4-stelligen Merkmalsvektoren der Blöcke. Ist diese Distanz kleiner als ein vorgegebener Schwellwert $D_{similar}$ (entspricht 0.0015 bei Farb- und 0.0025 bei Graustufenbildern), so wird zusätzlich die Distanz zwischen den Mittelpunkten der Kreise berechnet, um nah beieinander liegende Blöcke auszu-

schließen. Überschreitet diese Blockdistanz einen weiteren Schwellwert N_d (entspricht 120 bei Farb-, 25 bei Graustufenbildern), so gilt das entsprechende Blockpaar als relevant. Alle so gefundenen Blockpaare werden schließlich im Bild markiert, um duplizierte Regionen hervorzuheben.

Das von Cao et al. entwickelte Verfahren zeigt eine Robustheit der Merkmalsvektoren gegenüber additivem weißem Gaußschem Rauschen (AWGN) sowie gegenüber zusätzlich hinzugefügter Gaußscher Unschärfe. Die Reduktion der Merkmalsvektoren auf vier Stellen ermöglicht zudem eine schnelle Berechnung und damit schnelle Detektion duplizierter Regionen mit gleichzeitiger Genauigkeit bei der Klassifikation der Bildblöcke.

Ustubioglu et al. 2016

Ein weiteres blockbasiertes Detektionsverfahren, welches Anwendung in der eigens implementierten Software findet, ist das Verfahren von Beste Ustubioglu et al. aus dem Jahr 2016 [vgl. Ust+16]. Die Publikation greift u.a. erneut die Idee der Blockzerlegung und der Diskreten Kosinustransformation auf.

Zu Beginn des Verfahrens wird das Ausgangsbild vom RGB- in den YC_bC_r-Farbraum konvertiert und anschließend in 8x8 Pixel große Blöcke unterteilt. Anschließend wird auf jeden Block die DCT angewandt und die entsprechenden DCT-Koeffizienten werden mit dem vordefinierten Faktor 16 quantisiert. Aus diesen quantisierten DCT-Koeffizienten soll für jeden Block ein spezifischer Merkmalsvektor erstellt werden. Um den späteren Vergleich der Merkmalsvektoren zu vereinfachen, soll deren Wertebereich weiter beschränkt werden. Hierfür wird für jeden Block ein Zeichenblock berechnet, dessen Elemente nur drei unterschiedliche Werte annehmen können. Die Berechnung der Zeichenblöcke erfolgt anhand der Regelungen in Formel 5.6.

$$\begin{cases} \forall\, B^i_{xy} > 0 \implies S^i_{xy} = 1 \\ \forall\, B^i_{xy} < 0 \implies S^i_{xy} = -1 \\ \forall\, B^i_{xy} = 0 \implies S^i_{xy} = 0 \end{cases} \tag{5.6}$$

Für jeden quantisierten DCT-Koeffizienten an der Stelle (x,y) des Blockes B^i wird geprüft, ob dessen Wert größer, kleiner oder gleich Null ist. Der Zeichenblock S^i erhält an der entsprechenden Stelle (x,y) einen der drei Werte {-1,0,1}. Da die niedrigfrequenten DCT-Koeffizienten die wichtigsten Informationen des Blockes enthalten und robuster gegenüber JPEG-Kompression sind [vgl. Ust+16, S. 4], soll aus diesen der Merkmalsvektor V^i eines Blockes erstellt werden. Hierfür wird ein Zickzack-Scan auf den eben berechneten Zeichenblock angewendet und dessen erste 16 Werte werden als Vektor abgespeichert. Die letzten drei Stellen des 19-stelligen Merkmalsvektors werden mit den durchschnittlichen Farbwerten des Y-, des C_b- und des C_r-Kanals besetzt. Die

Merkmalsvektoren aller Blöcke werden anschließend in einer gemeinsamen Liste gespeichert und lexikografisch sortiert, sodass ähnliche Blöcke nebeneinander in der Liste gespeichert sind.

Um die Merkmalsvektoren zweier Blöcke miteinander zu vergleichen, wird zuerst die Differenz der durchschnittlichen Farbwerte berechnet. Sind die Differenzen der durchschnittlichen Y-, C_b- und C_r-Werte kleiner oder gleich einem vordefinierten Schwellwert t_c = 2.8, so wird mit dem Ähnlichkeitsvergleich fortgefahren. Andernfalls wird das entsprechende Blockpaar verworfen. Die Ähnlichkeit eines Blockpaars wird nun anhand von Formel 5.7 berechnet.

$$\begin{aligned} Z_k^{ij} &= V_k^i \cdot V_k^j & k &= 1....16 & Z_k^{ij} &\in \{-1,0,1\} \\ \forall(Z_k^{ij} == 1) \vee (V_k^i == 0 \wedge V_k^j == 0) &\implies d = d+1 & & & r^{ij} &= d/16 \end{aligned} \tag{5.7}$$

Die Merkmalsvektoren zweier Blöcke werden elementenweise multipliziert. Ist das Produkt gleich 1 oder haben beide Vektoren an der entsprechenden Stelle den Wert 0, so wird der Zähler d um eins inkrementiert. Diese Berechnung erfolgt für die ersten 16 Stellen des Merkmalsvektors. Ist das Verhältnis r^{ij} gleicher Stellen zweier Merkmalsvektoren größer oder gleich einem vordefinierten Schwellwert $t_s \approx 0.81$, so wird die Blockdistanz zwischen den beiden Blöcken berechnet. Ustubioglu et al. bieten in ihrer Arbeit die Möglichkeit, den Schwellwert t_s automatisch zu bestimmen, hier reicht es jedoch, ihn auf den durchschnittlich berechneten Wert 0.81 zu setzen.

Überschreitet auch die berechnete Blockdistanz einen vordefinierten Schwellwert t_d = 40, so wird das Blockpaar als relevant eingestuft. Um falsch detektierte Blockpaare zu entfernen, wird als letzter Schritt des Verfahrens der RANSAC-Algorithmus angewandt. Anschließend werden die übrigen Blockpaare im Bild farblich markiert, sie zeigen die duplizierten Regionen.

5.1.2 Keypointbasierte Detektionsverfahren

Wie bereits erwähnt wird das Ausgangsbild bei der Anwendung keypointbasierter Verfahren nicht in Blöcke unterteilt. Stattdessen werden anhand etablierter Algorithmen sogenannte Keypoints, Bildpunkte mit besonders hoher Entropie, aus dem Bild extrahiert [vgl. Yan+17, S. 73]. Mit Hilfe dieser Keypoints und weiterer Berechnungen können so duplizierte Regionen in einem digitalen Bild gefunden werden. Die beiden am häufigsten verwendeten Algorithmen zur Extraktion von Keypoints sind SIFT und SURF, aber auch andere Extraktionsmethoden wie KAZE[15], AKAZE [vgl. Yan+17, S. 73] oder der

[15] Kaze ist das japanische Wort für Wind. AKAZE steht für accelerated (dt. beschleunigtes) KAZE

Harris-Kantendetektor finden zunehmend Anwendung. Im Bereich der Detektion manipulierter Bildregionen kommen zudem auch Kombinationen verschiedener Algorithmen zum Einsatz, um eine größere Anzahl relevanter Punkte im Bild zu finden.

Ein wesentlicher Vorteil keypointbasierter Verfahren ist die niedrige Rechenzeit. Während bei blockbasierten Verfahren häufig das komplette Bild iteriert werden muss, bieten Keypoints die Möglichkeit, nur die relevantesten Regionen eines Bildes zu untersuchen. Ein Nachteil solcher Verfahren liegt jedoch in der genauen Lokalisation manipulierter Regionen. Da anstatt mit Blöcken nur mit einzelnen Punkten des Bildes gearbeitet wird, kann es vorkommen, dass nicht alle Pixel der manipulierten Bildregion als solche detektiert werden. Um diese Bildbereiche dennoch besser hervorzuheben, verwenden einige Autoren zusätzliche morphologische Operationen wie Dilatation. Die folgenden Verfahren verwenden verschiedene Algorithmen zur Extraktion von Keypoints sowie weitere Berechnungen für die Detektion duplizierter Regionen in digitalen Bildern.

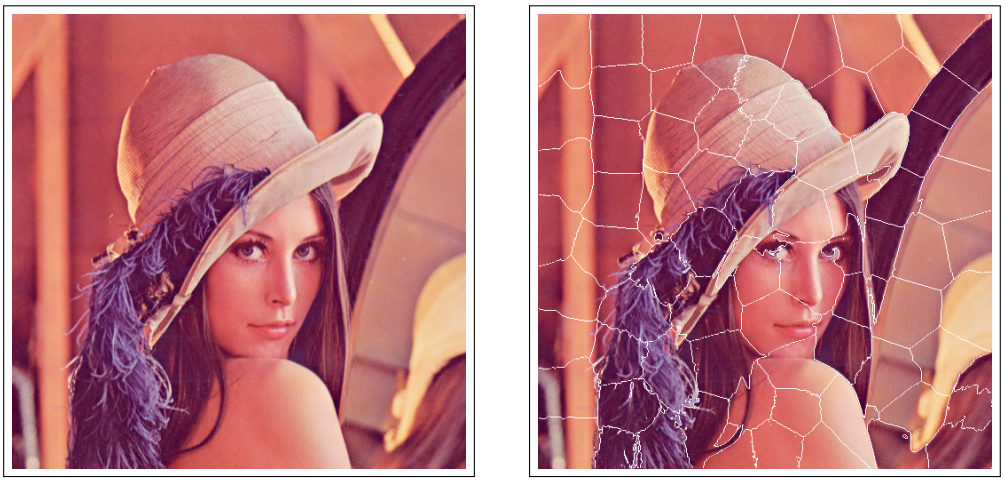
Yang et al. 2017

Das Verfahren, welches 2017 von den Autoren Fan Yang et al. veröffentlicht wurde [vgl. Yan+17], kombiniert die beiden Extraktionsalgorithmen SIFT und KAZE, um so auf einer größeren Grundmenge von Keypoints arbeiten zu können. Nach Extraktion der Keypoints beider Algorithmen wird für jeden Keypoint ein Deskriptorvektor berechnet, welcher diesen Keypoint genauer beschreibt. Anhand dieser Deskriptoren ist es möglich, Keypoints untereinander zu vergleichen. Um die Rechenzeit für den Vergleich der Keypoints zu reduzieren, werden diese in einer Liste abgespeichert und anschließend nach ihren Deskriptoren sortiert, sodass sich zwei ähnliche Keypoints nebeneinander in der sortierten Liste befinden. Dies ermöglicht es, immer zwei benachbarte Keypoints miteinander zu vergleichen anstatt für jeden Keypoint über die gesamte Liste zu iterieren.

Der Vergleich zweier Keypoints erfolgt über die Berechnung der Distanz zwischen den zwei Punkten im Bild sowie über deren Ähnlichkeit, welche anhand der Euklidischen Distanz berechnet wird. Die Berechnung der Distanz im Bild dient dabei als erster Filter bei der Auswahl relevanter Keypointpaare, um zu verhindern, dass zwei Keypoints derselben engeren Bildregion als relevantes Paar erkannt werden. Ist diese Distanz kleiner als ein vordefinierter Schwellwert $d_{spatial} = 30$, so wird das entsprechende Keypoint-Paar aus der Liste der potentiell relevanten Paare entfernt. Für die übrigen Paare wird die Euklidische Distanz berechnet. Überschreitet diese einen weiteren vordefinierten Schwellwert $d_{similar} = 0.0005$, so gilt das entsprechende Keypoint-Paar als relevant.

Der letzte Schritt des Verfahrens ist die Entfernung falscher Keypointpaare, welche sich aus zwei Schritten zusammensetzt. Der erste Schritt ist die Segmentierung des Bildes in sogenannte Superpixel. Bei diesen handelt es sich um irregulär geformte, sich nicht überlappende Bildregionen, deren Pixel gemeinsame Charakteristiken wie etwa Pixelintensitäten aufweisen [vgl. Yan+17, S. 76]. Eine häufig für diese Art der Bildsegmen-

tierung verwendete Methode ist der SLIC-Algorithmus[16]. Das Ergebnis einer solchen Segmentierung ist in Abbildung 5.3 zu sehen. Die Annahme der Autoren ist, dass korrekt erkannte Keypointpaare lediglich auf wenige Superpixelregionen verteil sind [vgl. Yan+17, S. 76]. Aus diesem Grund werden die Superpixelregionen ermittelt, in denen der Großteil der Keypoints liegt, Keypoints außerhalb dieser Regionen werden verworfen.



(a) Das Original ohne Segmentierung [Wik20]

(b) Das Bild mit eingezeichneten Superpixeln, Eigene Darstellung

Abbildung 5.3: Bildsegmentierung in Superpixelregionen anhand des SLIC-Algorithmus'

Für den zweiten Schritt zur Entfernung falscher Keypointpaare kommt der RANSAC-Algorithmus[17] zum Einsatz. Dieser sucht nach einer affinen Transformation zwischen den zwei Regionen der Keypointpaare und verwirft diejenigen Paare, welche nicht dieser Transformation entsprechen [vgl. Yan+17, S. 76]. In der Regel handelt es sich bei Copy-Move-Bildmanipulationen um einfache Verschiebungen im Bild, andere Transformationen wie Rotation oder Skalierung sind jedoch ebenfalls denkbar. Nach erfolgreicher Entfernung falscher Paare werden die übrigen Keypoints im Ausgangsbild farbig markiert, um die duplizierten Regionen hervorzuheben.

Ulutas und Muzaffer 2016

Ein großer Nachteil keypoint-basierter Methoden liegt in der Detektion duplizierter Regionen mit uniformen Flächen, beispielsweise bei der Duplikation eines wolkenbedeckten Himmels. Da Keypoints anhand hoher Entropiewerte aus einem Bild extrahiert werden und uniforme Flächen größtenteils niedrige Entropien besitzen, ist es kaum möglich, hier Keypoints zu extrahieren.

Die Autoren Ulutas und Muzaffer nehmen sich diesem Problem in ihrer Publikation von 2016 an [vgl. UM16], indem sie zur Extraktion von Keypoints den AKAZE-Algorithmus

[16] SLIC = Simple Linear Iterative Clustering
[17] RANSAC = Random Sample Consensus

verwenden. Durch diesen ist es möglich, auch Keypoints in uniformen Bildregionen zu finden. Im ersten Schritt des Verfahrens werden demnach AKAZE-Keypoints aus dem Bild extrahiert. Außerdem wird zu jedem Keypoint ein Deskriptorvektor berechnet, anhand dessen der Keypoint identifiziert und mit anderen Punkten verglichen werden kann.

Um die extrahierten Keypoints miteinander zu vergleichen, werden sie zunächst in einer Liste gespeichert und anhand ihrer Deskriptoren sortiert, sodass zueinander ähnliche Keypoints nebeneinander in der Liste abgespeichert sind. Anschließend können so je zwei nebeneinander liegende Keypoints miteinander verglichen werden. Für den Vergleich wird in diesem Verfahren nicht die Euklidische Distanz, sondern die Hamming-Distanz der beiden Deskriptoren berechnet (siehe Formel 5.8). Diese vergleicht zwei Deskriptoren A_i und A_j stellenweise und gibt die Anzahl der nicht übereinstimmenden Stellen zurück. Ist schließlich die Hamming-Distanz zweier Deskriptoren kleiner als ein vordefinierter Schwellwert δ = 50, so werden die entsprechenden beiden Keypoints als ähnlich angesehen.

$$Hamming(i,j) = \sum_{k=1}^{N} \text{XOR}\ (A_i^k, A_j^k) \qquad k = \text{Stelle des Deskriptors} \tag{5.8}$$

Nachdem alle Keypoints paarweise miteinander verglichen wurden, wird als letzter Schritt der RANSAC-Algorithmus auf die Menge der gefundenen Keypoints angewendet. Ziel ist es dabei, falsch klassifizierte Keypointpaare zu entfernen, sodass letztendlich nur die Keypoints übrig sind, die zu einer duplizierten Region im Bild gehören. Um die Manipulation zu veranschaulichen, werden die entsprechenden Keypoints im Ausgangsbild farblich markiert.

Ardizzone et al. 2015

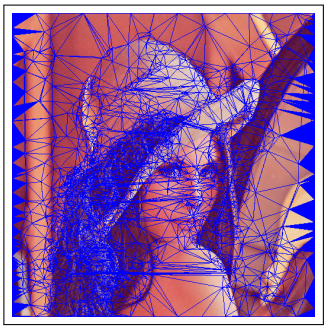
Während andere Autoren sich auf die Extraktion von Keypoints und den Vergleich derer Deskriptoren konzentrieren, entwickelten Ardizzone et al. 2015 ein mathematisch komplexeres Verfahren [vgl. ABM15]. In diesem wird das Ausgangsbild anhand von extrahierten Keypoints und eines Triangulations-Algorithmus' in unregelmäßige Dreiecke unterteilt. Die Detektion von duplizierten Bildregionen erfolgt anschließend durch den Vergleich der Eigenschaften einzelner Dreiecke.

Das Verfahren beginnt mit der Extraktion von Keypoints. Um ausreichend viele Keypoints für die anschließende Triangulation zu extrahieren, werden drei verschiedene Extraktionsalgorithmen verwendet: SIFT, SURF und der Harris-Kantendetektor. Auf diese Menge an Keypoints wird der Delaunay-Triangulationsalgorihmus angewendet. Dieser iteriert über das Ausgangsbild und zeichnet anhand mathematischer Berechnungen Linien zwischen den Keypoints ein, sodass nach erfolgreicher Triangulation das Bild in unregelmäßige Dreiecke unterteilt ist. An den Rändern eines Bildes ist es i.d.R. schwierig,

Keypoints zu extrahieren. Um diese Bildbereiche jedoch trotzdem in die Untersuchung einzubeziehen, fügen Ardizzone et al. vor der Trianglation willkürliche Punkte an den Rändern des Bildes ein. Ein Beispiel für die Ergebnisse der Keypoint-Extraktion und der Delaunay-Triangulation ist in Abbildung 5.4 zu sehen.

(a) Das Original, Quelle: [Wik20]

(b) Das Bild mit eingezeichneten Keypoints, Eigene Darstellung



(c) Das Bild mit eingezeichneten Dreiecken, Eigene Darstellung

Abbildung 5.4: Delaunay-Triangulation des Bildes „Lena“ anhand extrahierter Keypoints[18]

Ardizzone et al. stellen in ihrer Publikation zwei verschiedene Möglichkeiten vor, die gerade erzeugten Dreiecke miteinander zu vergleichen, um so duplizierte Regionen zu erkennen. Für die Implementierung der Software dieser Arbeit wird jedoch nur die erste der beiden Möglichkeiten umgesetzt.

$$||x||_1 = \sum_{k=1}^{m} |x_i| \tag{5.9}$$

Für den Vergleich zweier Dreiecke werden zuerst die jeweils 4 dominanten Rot- Grün und Blauwerte des Dreiecks berechnet. Dazu werden die Farbwerte der Pixel innerhalb des Dreiecks aufsummiert und durch die Anzahl der Pixel im Dreieck dividiert. So ergeben sich für jedes Dreieck insgesamt 12 repräsentative Farbwerte. Um die Dreiecke nun effizient vergleichen zu können, werden sie in einer Liste abgespeichert und sortiert. Das Sortieren erfolgt hier, im Gegensatz zu anderen Detektionsverfahren, welche hierfür lexikografisches Sortieren verwenden, durch die Berechnung der L_1-Norm. Die auch als Summennorm bezeichnete L_1-Norm eines Vektors ergibt sich aus der Summe der Beträge der einzelnen Vektorkomponenten, zu sehen in Formel 5.9 [BDB13, S. 84]. Der Vektor besteht hier aus den gerade berechneten 12 dominanten Farben eines Dreiecks.

Nach Sortieren der Dreiecke anhand der L_1-Norm ihrer dominanten Farben, liegen nun jeweils zwei farblich ähnliche Dreiecke in der sortierten Liste nebeneinander. Dies er-

[18] Zur besseren Veranschaulichung wurde anstatt eines Keypoints ein Kreis um den Keypoint herum eingezeichnet.

möglicht es, immer zwei benachbarte Dreiecke miteinander zu vergleichen, anstatt jedes Mal über die gesamte Liste aller Dreiecke zu iterieren. Für den Vergleich zweier Dreiecke werden als nächstes deren Flächeninhalte berechnet. Erfüllt das Verhältnis r_A ihrer Flächeninhalte die Vorgabe in Formel 5.10, so werden als nächstes die Innenwinkel der beiden Dreiecke berechnet.

$$r_A = \frac{min(A_i, A_j)}{max(A_i, A_j)} \geq 0.25 \tag{5.10}$$

Das Sortieren der Dreiecke anhand der L_1-Norm ihrer Farben und der Vergleich ihrer Flächeninhalte dienten einem groben Vergleich, durch welchen bereits erste Dreiecke ausgeschlossen werden konnten. Der detaillierte Vergleich zweier Dreiecke erfolgt nun durch die Berechnung der Summe der absoluten Differenzen (SAD) ihrer dominanten Farbwerte sowie der SAD ihrer Innenwinkel (siehe Formel 5.11). Unterschreitet die SAD der Farbwerte einen vordefinierten Schwellwert $TH^c = 0$ und unterschreitet die SAD der Innenwinkel einen weiteren Schwellwert $TH^a = 0.25$, so werden die beiden Dreiecke als dupliziertes Paar angesehen.

$$SAD = \sum_{k=1}^{m} |x_i - x_j| \tag{5.11}$$

Als Abschluss des Verfahrens werden alle Dreieckpaare, welche die vorherigen Bedingungen erfüllt haben, im Ausgangsbild farblich markiert, um so die duplizierten Regionen hervorzuheben. Für eine optimierte Markierung zusammenhängender Bildregionen, können morphologische Operationen wie Dilation angewendet werden, um die Lücken zwischen den markierten Dreiecken zu füllen.

5.2 Konzept und Design der Software

Für die Planung und Umsetzung eines Softwareprojekts ist es nötig abzuwägen, welche Architekturen und Komponenten für dieses verwendet werden sollen. Die Auswahl dieser Komponenten hängt dabei u.a. davon ab, welche Anforderungen an das Projekt gestellt werden und welche Funktionen die geplante Software erfüllen soll. Auch Überlegungen zu Aufbau und Design haben hier Auswirkung auf die Wahl der verwendeten Komponenten. Im Folgenden sollen der Aufbau und die Funktionsweise der implementierten Software erläutert werden. Dafür werden zuerst wichtige verwendete Komponenten wie externe Bibliotheken erklärt. Anschließend folgt eine detaillierte Erklärung des Klassendiagramms sowie Informationen zur Handhabung der Software.

5.2.1 Verwendete Komponenten

Zwei wichtige Komponenten, für welche für die Umsetzung eines Softwareprojekts eine Auswahl getroffen werden muss, sind die passende Programmiersprache und Entwicklungsumgebung. Für die Umsetzung dieser Software fiel die Wahl auf die Programmiersprache Java, verwendet wurde das Java Development Kit (JDK) in der Version 14.0.2. Java wurde u.a. ausgewählt, weil es sich dabei um eine robuste, plattformunabhängige Programmiersprache handelt, die zudem verschiedene hilfreiche Standardbibliotheken zur Verfügung stellt. Für die integrierte Entwicklungsumgebung fiel die Wahl auf IntelliJ IDEA des Software-Herstellers JetBrains. Diese bietet vor allem Funktionen für eine optimierte Programmierung mit Java, darunter beispielsweise intelligente Codeunterstützung sowie einfache Versionsverwaltung und die Möglichkeit zur Durchführung von Testläufen [vgl. Jet20].

Eine weitere wichtige Komponente, die bei der Umsetzung eines Softwareprojekts bedacht werden sollte, ist die Einbindung externer Bibliotheken und Quellcodes. Die in Kapitel 5.1 erläuterten Verfahren zur Detektion digitaler Bildmanipulationen verwenden teilweise komplexe Operationen der Bildverarbeitung sowie weitere mathematische Algorithmen, deren Funktionen nicht in den Java-Standardbibliotheken enthalten sind. Daher ist es nötig, hier auf externe Quellen zurückzugreifen, die genau diese Funktionen umsetzen. Im Folgenden werden die verwendeten externen Bibliotheken und Quellcodes kurz erklärt. Dabei wird insbesondere Bezug genommen zu den daraus verwendeten Funktionen.

Open Source Computer Vision Library (OpenCV)

Laut der offiziellen Webseite ist OpenCV eine Softwarebibliothek, die speziell für die Forschungsgebiete Computer Vision und Maschinelles Lernen entwickelt wurde. Anwendung finden die in der Bibliothek enthaltenen Algorithmen beispielsweise bei der Detektion von Gesichtern in Bildern und Videos sowie bei der Verfolgung von Bewegungen und der Suche nach ähnlichen Bildern innerhalb einer Datenbank. OpenCV ist als C++-Code implementiert, unterstützt jedoch auch verschiedene andere Programmiersprachen wie Java und Python sowie gängige Betriebssysteme wie Windows, Linux und Android [vgl. Open20a].

Für die in dieser Arbeit implementierte Software wird die aktuelle Version OpenCV 4.4.0 verwendet [Open20b]. Diese bietet viele verschiedene Möglichkeiten zur Verarbeitung digitaler Bilder. Beispielsweise dient die Klasse *Core* dazu, alle benötigten Systemressourcen einzubinden und ermöglicht eine OpenCV interne Verarbeitung von Bildern in Form von *Mat*-Objekten. Weitere verwendete Module sind beispielsweise *Calib3d* zur Anwendung des RANSAC-Algorithmus' sowie *Ximgproc* und *Imgcodecs* zur Berechnung von Superpixelregionen und für die Ein- und Ausgabe von Bildern. Auch die Extraktion von Keypoints und die Berechnung derer Deskriptorvektoren ist mithilfe der enthaltenen OpenCV-Module möglich.

externer Quellcode

Für die Implementierung des Quellcodes wird neben der bereits erwähnten Bibliothek OpenCV auf zwei öffentliche GitHub-Repositorys zurückgegriffen. GitHub ist hier eine Onlineplattform basierend auf der Versionsverwaltungssoftware Git, die zur privaten und öffentlichen Versionsverwaltung von Softwareprojekten verwendet wird. Das erste Repository wurde am 05.06.2018 von Johannes Diemke und Philipp Haussleiter über GitHub veröffentlicht und enthält eine Implementierung des bereits erwähnten Delaunay-Triangulations-Algorithmus', mit dessen Hilfe Bilder anhand einer Menge aus Bildpunkten in unregelmäßige Dreiecke unterteilt werden können [Die20]. Die Klassen *DelaunayTriangulator*, *Edge2D*, *EdgeDistancePack*, *NotEnoughPointsException*, *Triangle2D*, *TriangleSoup* und *Vector2D* des Repositorys werden unverändert in die Software dieser Arbeit übernommen. Die Klassen befinden sich im Repository unter dem Pfad */delaunay-triangulator/library/src/main/java/io/github/jdiemke/triangulation*. Im eigens implementierten Quellcode wird ein Objekt der Klasse *DelaunayTriangulator* erzeugt. Die anschließend aufgerufenen Methoden *triangulate()* und *getTriangles()* führen die Delaunay-
Triangulation durch, indem sie Methoden der oben genannten Klassen aufrufen, und geben schließlich eine Liste von Dreiecken zurück. Eine beispielhafte Verwendung hierfür ist in Abbildung 5.4.c) zu sehen.

Eine weitere externe Quelle, welche für die Implementierung der Software zu Rate gezogen wurde, ist ein Beitrag auf der Internetplattform Stack Overflow [Dar16]. Da es sich bei Stack Overflow um eine freie Plattform handelt, auf welcher Nutzer beinahe ohne Beschränkungen Inhalte veröffentlichen dürfen, wurde der genannte Beitrag vor der Verwendung auf seine Richtigkeit geprüft und entsprechend an die eigene Software angepasst. Stack Overflow ist zweifellos keine wissenschaftlich korrekte Quelle und soll hier nur zur Inspiration bezüglich der Implementierung dienen. Der genannte Internetbeitrag zeigt eine Möglichkeit, die Diskrete Wavelet-Transformation (DWT) in Java zu implementieren. Da der bereitgestellte Code nur auf quadratische Bilder angewandt werden kann, deren Breite und Höhe durch die Potenz 2^n teilbar ist, wurde er entsprechend für Bilder anderer Maße angepasst. Für die Implementierung der Stationären Wavelet-Transformation (SWT) wurde der Code gemäß den Anforderungen der SWT (siehe Kapitel 2.1) erneut abgeändert.

5.2.2 Aufbau und Handhabung der Software

Um ein besseres Verständnis für die Funktionsweise der implementierten Software zu erhalten, sollen im Folgenden der Aufbau sowie die Handhabung des Programms erläutert werden. Nach einem groben Überblick über die Struktur der Software folgen dazu eine Anleitung zur Ausführung sowie Anwendungshinweise in Bezug auf die Einbindung externer Bibliotheken und manueller Anpassungen, welche vor Ausführung des Programms vorgenommen werden müssen.

Struktur der Klassen und Methoden

Auf der Grundlage aufbauend, dass es sich bei Java um eine objektorientierte Programmiersprache handelt, bilden Pakete und Klassen hier das Grundgerüst der Software. Die einzelnen Klassen wiederum enthalten Attribute und Methoden, mit deren Hilfe Informationen übertragen und Funktionen ausgeführt werden können. Ein grober Aufbau der Paketstruktur ist in Abbildung 5.5 zu sehen. Die Wurzel dieser Struktur bildet das Paket *main.java*, welches fünf weitere Unterpakete sowie deren Klassen enthält. Die in Kapitel 5.1 erläuterten Detektionsverfahren werden hier als separate Klassen implementiert. Diese befinden sich in den Paketen *blockBasiert* und *keypointBasiert*, welche nach der Art der jeweils enthaltenen Detektionsmethoden benannt sind.

Das Paket *imageUtils* enthält drei Klassen, deren Methoden bei der Verarbeitung digitaler Bilder angewandt werden. In den Klassen *DWT.java* und *SWT.java* ist die jeweilige Funktionsweise der Diskreten Wavelet-Transformation bzw. der Stationären Wavelet Transformation implementiert. Die Klasse *ImageUtils.java* enthält weitere Methoden zur Verarbeitung digitaler Bilder, u.a. zur Extraktion von Keypoints oder zur Durchführung verschiedener Transformationen sowie mathematischer Operationen am Bild. Der Quellcode der Klasse *DWT.java* entstammt einer externen Quelle [Die20]. Da der originale Code lediglich für die Verarbeitung quadratischer Bilder vorgesehen ist, wurde dieser an die Transformation nicht-quadratischer Bilder angepasst. In der Klasse *SWT.java* wurde der Quellcode zusätzlich entsprechend der Funktionsweise der Stationären Wavelet-Transformation abgeändert.

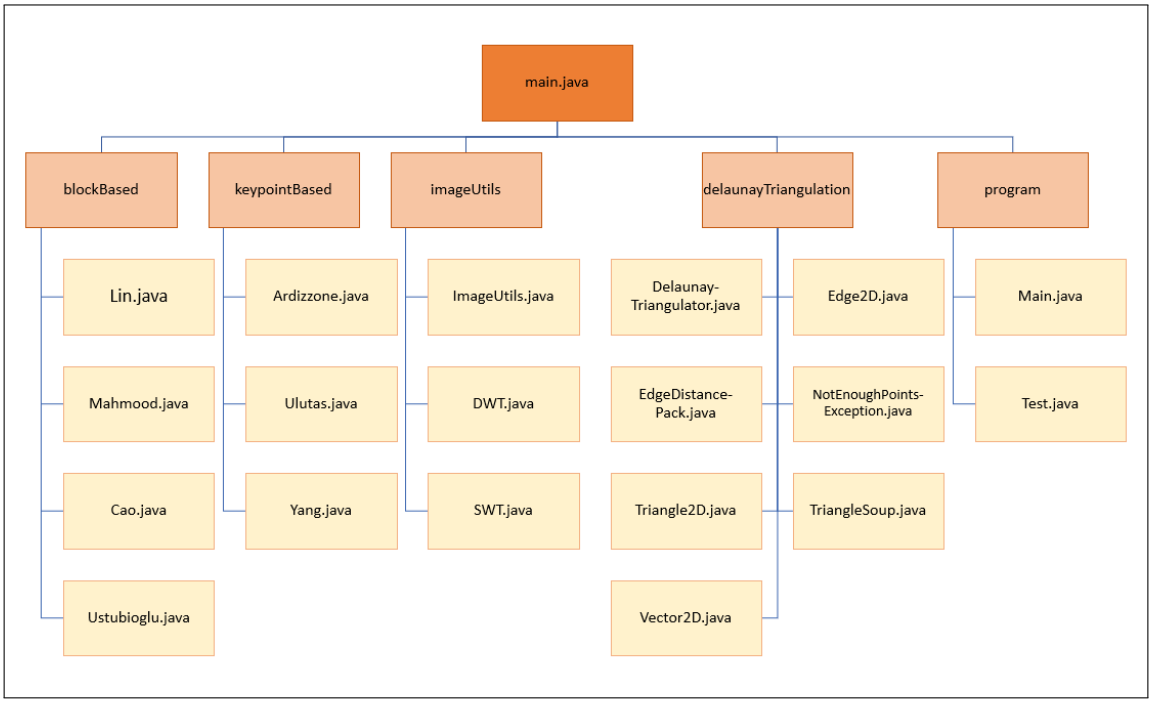


Abbildung 5.5: Übersicht der Pakete und Klassen der implementierten Software, Eigene Darstellung

Ein weiteres Paket trägt den Namen *delaunayTriangulation* und enthält mehrere Klassen, welche die Funktionsweise der Delaunay-Triangulation implementieren, durch welche ein digitales Bild anhand von extrahierten Keypoints in unregelmäßige Dreiecke unterteilt werden kann. Der Quellcode hierfür stammt von Johannes Diemke [Die20]. Die Klassenstruktur von Diemke wurde unverändert übernommen, zur Verwendung des Quellcodes wird lediglich ein Objekt der Klasse *DelaunayTriangulator.java* erzeugt, über welches alle restlichen Klassen und Methoden aufgerufen werden.

Das letzte Paket der Software, *program*, enthält die Hauptklasse, über welche das Programm ausgeführt wird, sowie eine Klasse zum Testen der Funktionsweise und Qualität der fertigen Software. Eine Erläuterung dieses Testablaufs folgt in Kapitel 5.3.2. Die Funktionsweise der Hauptklasse *Main.java* soll im Folgenden näher betrachtet werden. Abbildung 5.6 zeigt hierfür den vereinfachten Arbeitsablauf des Programms. Der Dateipfad des digitalen Bildes, welches auf Copy-Move-Manipulationen untersucht werden soll, wird als Eingabewert der Methode *main()* übergeben. Diese enthält sowohl Methodenaufrufe der eigenen Klasse als auch der externen Klassen *Yang.java, Ardizzone.java, Ulutas.java* sowie *Lin.java* und *Mahmood.java*. Ziel des Programms ist es, die Vorteile sowohl der keypointbasierten als auch der blockbasierten Detektionsverfahren zu nutzen, um genauere Ergebnisse bei der Bildmanipulationserkennung zu erzielen.

Funktionsweise des Arbeitsablaufs

Das Eingangsbild wird anhand seines Dateipfades an die *main()*-Methode der Klasse *Main.java* übergeben und dort als *BufferedImage*-Objekt abgespeichert. Im nächsten Schritt (1) werden über die Methode *getKeypoints()* die *test()*-Methoden der Klassen *Yang.java, Ardizzone.java* und *Ulutas.java* aufgerufen (2a, 2b und 2c). Diese extrahieren, unter Verwendung weiterer Methoden der genannten Klassen sowie unter Verwendung verschiedener Extraktionsalgorithmen, Keypoints aus entropiereichen Regionen des Eingangsbildes. Diese Keypoints werden anschließend miteinander verglichen, um duplizierte Regionen innerhalb des untersuchten Bildes zu erkennen. Die gefundenen Keypointpaare[19] werden als Liste an die *main()*-Methode der Klasse *Main.java* zurückgegeben.

Nach erfolgreicher Extraktion der Keypoints konnten nun duplizierte Regionen innerhalb des Eingangsbildes gefunden werden. Da keypointbasierte Detektionsverfahren jedoch nur mit einzelnen Bildpunkten anstatt mit größeren Bildblöcken arbeiten, erweist sich eine vollständige Markierung der duplizierten Regionen als schwierig. Hier sollen nun die Vorteile der blockbasierten Verfahren zum Einsatz kommen, mit deren Hilfe diese Regionen vollständig markiert werden können. Da blockbasierte Methoden nur auf größeren, zusammenhängenden Bildregionen eingesetzt werden können, besteht der nächste Schritt des Programms in der Verarbeitung der extrahierten Keypoints. Mithilfe interner Methoden der Klasse *Main.java* werden so rechteckige Regionen um die ge-

[19] Ein Paar besteht hier aus zwei Keypoints, von denen sich einer in der kopierten und einer in der neu eingefügten Bildregion befindet.

fundenen Keypoints herum ermittelt, auf welche anschließend die Methoden der blockbasierten Klassen angewendet werden können.

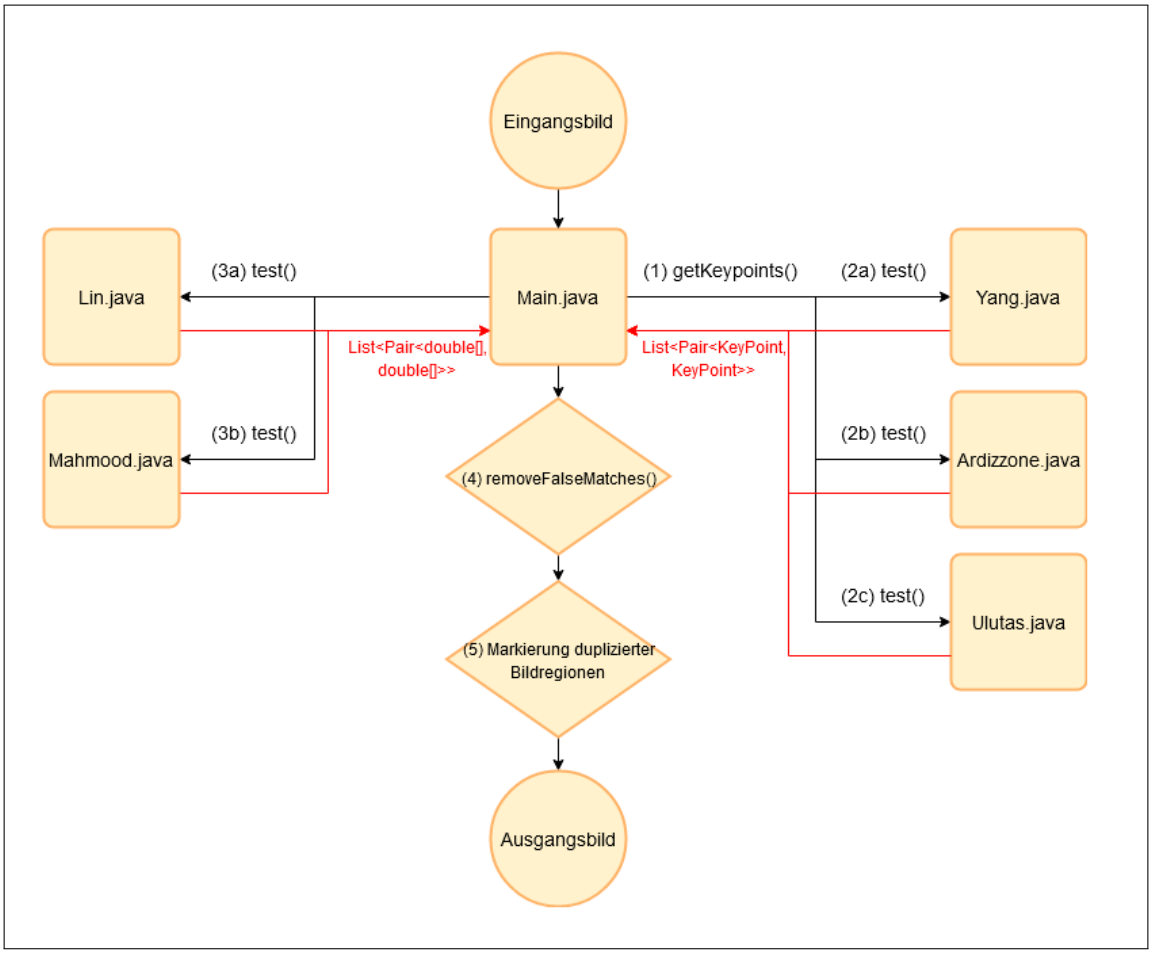


Abbildung 5.6: Arbeitsablauf der implementierten Software, Eigene Darstellung

Als nächster Schritt werden in der *main()*-Methode der Klasse *Main.java* die *test()*-Methoden der Klassen *Lin.java* und *Mahmood.java* aufgerufen (3a und 3b). Diese teilen die gerade ermittelten Bildregionen in gleichmäßige, quadratische Blöcke und vergleichen diese miteinander, um Rückschluss auf duplizierte Regionen ziehen zu können. Die so ermittelten ähnlichen Bildblöcke werden als Paare in einer Liste abgespeichert und an die *main()*-Methode der Klasse *Main.java* zurückgegeben. Da die verwendeten keypointbasierten Detektionsverfahren Keypoints, die blockbasierten Verfahren jedoch Bildblöcke in Form von Merkmalsvektoren (hier *Double-Arrays*) an die *main()*-Methode zurückgeben, folgt eine Konvertierung der beiden übergebenen Listen. Das Ergebnis ist eine kombinierte Liste aus Bildpunkten, die übergebenen Blockpaare werden hier durch ihr jeweiliges Offset im Bild (linke obere Ecke eines Blockes) repräsentiert.

Da es vorkommen kann, dass Bildblöcke oder Keypoints aufgrund ihrer ähnlichen Farb- oder Entropiewerte fälschlicherweise als Teil duplizierter Bildregionen erkannt werden, ist es bei der Detektion von digitalen Bildmanipulationen i.d.R. notwendig, die gefundenen Punkte weiter zu untersuchen. So können durch die Verwendung von Algorithmen

wie RANSAC falsche Keypoint- oder Blockpaare aus der Liste der gefundenen Paare entfernt werden. Die hier implementierte Software behält ausschließlich die Bildpunkte der Liste, welche innerhalb einer eingegrenzten Region liegen und durch beide Verfahren der Klassen *Lin.java* und *Mahmood.java* erkannt wurden. Dieser Auswahlvorgang erfolgt über den Aufruf der Methode *removeFalseMatches()* der Klasse *Main.java* (4).

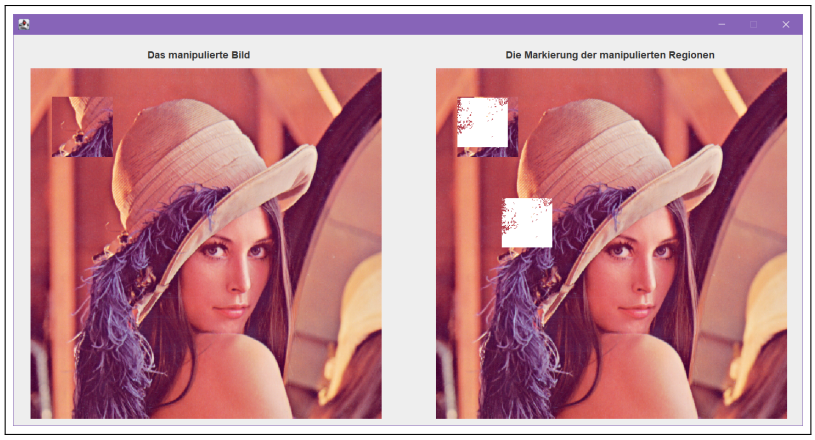


Abbildung 5.7: Beispiel der Programmausgabe nach erfolgreicher Detektion duplizierter Bildregionen, Eigene Darstellung

Im letzten Schritt des Programmablaufs werden die übrigen Punkte der Liste sowie deren zugehörige Bildblöcke im Eingangsbild eingefärbt, um die gefundenen duplizierten Regionen zu markieren (5). Konnte eine Bildmanipulation durch Copy-Move festgestellt werden, so öffnet sich ein Pop-Up-Fenster, welches das untersuchte Bild sowie eine Kopie dieses Bildes mit eingefärbten duplizierten Regionen zeigt (siehe Abb. 5.7). Wurde das Eingangsbild jedoch als authentisch klassifiziert, so öffnet sich ein ähnliches Fenster, welches das originale Bild zeigt und darauf verweist, dass es sich bei diesem nicht um eine Manipulation handelt.

Anwendungshinweise

Neben der gerade erläuterten Funktionsweise des Programms folgen nun einige Hinweise zu dessen Ausführung sowie zur Einbindung der benötigten externen Bibliotheken. Das als Git-Repository zur Verfügung gestellte Projekt kann zur Verwendung in verschiedenen Entwicklungsumgebungen geöffnet werden. Es folgt eine Erläuterung der Verwendung in der Entwicklungsumgebung IntelliJ IDEA des Software-Herstellers JetBrains.

Nachdem das Projekt in IntelliJ geöffnet wurde, müssen alle benötigten externen Bibliotheken integriert werden. Dies erfolgt über das Menü *File -> Project Structure* unter dem Reiter *Modules -> Dependencies* (siehe Abb. 5.8 oben). Die Bibliotheken *Apache Commons Math 3.6.1* und *OpenCV 4.4.0* müssen in der dargestellten Reihenfolge über das Pluszeichen (rechts) über den Reiter *JARs and directories* hinzugefügt werden. Nach Aufforderung des Betriebssystems ist hier der Pfad zu den Java-Archiven *commons-*

math3-3.6.1.jar und *opencv-440.jar* anzugeben. Weiterhin muss für OpenCV der lokale Pfad des zur Verfügung gestellten Ordners *opencv_440/Release* angegeben werden, in diesem befinden sich weitere benötigte Bibliotheken. Durch Doppelklick auf den Eintrag von OpenCV öffnet sich das in Abbildung 5.8 unten dargestellte Fenster, über welche diese Anpassungen vorgenommen werden können.

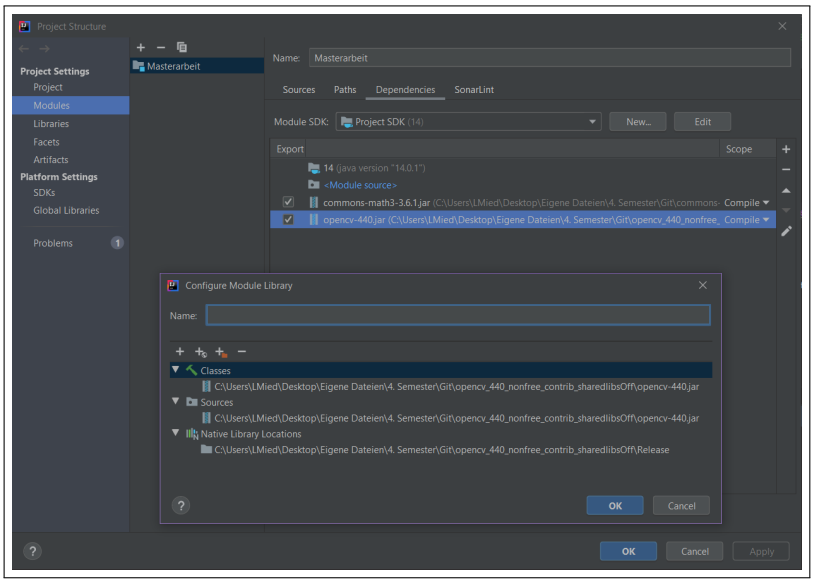


Abbildung 5.8: Einbindung der externen Bibliotheken und Anpassung der Dateipfade der Bibliotheken in IntelliJ IDEA, Eigene Darstellung

Nach korrekter Einbindung aller benötigten externen Bibliotheken und Anpassung der erwähnten Dateipfade kann das Programm nun ausgeführt werden. Um ein digitales Bild mithilfe der Software auf Copy-Move-Manipulationen zu testen, muss die *main()*-Methode der Klasse *Main.java* ausgeführt werden. Zur korrekten Ausführung muss als Übergabeparameter der absolute Dateipfad des zu testenden Bildes angegeben werden. Als Ausgabe des Programms wird ein Pop-Up-Fenster geöffnet, welches bestätigt, ob es sich bei dem untersuchten Bild um eine Manipulation handelt oder nicht. Bei einem positiven Detektionsergebnis werden die gefundenen duplizierten Regionen im Bild eingefärbt (siehe Abb. 5.7).

5.3 Test der Software

Nach der Implementierung einer Software ist es i.d.R. nötig, deren Funktionsweise genauestens zu überprüfen, um sicherzustellen, dass weder syntaktische noch semantische Fehler auftreten. Die Überprüfung der Syntax findet dabei bereits während der Kompilierung statt. Treten hier Fehler auf, lässt sich das Programm nicht kompilieren und ausführen. Nach erfolgreicher Kompilierung ist jedoch noch immer nicht sichergestellt, dass das Programm die Funktionen erfüllt, die gefordert sind. So kann es beispielsweise vorkommen, dass eine Methode falsch implementiert wurde und daher nicht

die gewünschten Ergebnisse liefert. Um auch solche semantischen Fehler zu beheben, muss das Programm weiteren Tests unterzogen werden.

Da es sich bei der Software dieser Arbeit lediglich um eine prototypische Implementierung handelt, wird hier auf das detaillierte Testen des Programms, etwa anhand von Modultests mit Frameworks wie JUnit, verzichtet. Stattdessen wird die Funktionsweise der Software anhand von Bilddatensätzen getestet, welche der Reihe nach als Eingabeparameter in das Programm geladen werden. Die jeweiligen Ausgaben werden manuell daraufhin überprüft, ob das getestete Bild als authentisch oder manipuliert klassifiziert wurde. Im Folgenden sollen nun die für den Test ausgewählten Datensätze vorgestellt werden. Dabei wird u.a. der Aufbau jedes einzelnen Datensatzes erklärt. Anschließend wird der Ablauf des Testvorgangs anhand ausgewählter Beispiele detailliert erläutert.

5.3.1 Auswahl der Testdatensätze

Im Forschungsgebiet der Detektion digitaler Bildmanipulationen haben sich über die letzten Jahrzehnte verschiedene Bilddatensätze etabliert, welche in vielen Publikationen Verwendung finden. Dabei unterscheidet sich der Aufbau dieser Datensätze je nach Anwendungsgebiet und Voraussetzungen für den Test neu entwickelter Verfahren. So existieren beispielsweise Datensätze, welche sich ausschließlich auf die Manipulationsmethode Copy-Move konzentrieren. Diese Datensätze unterscheiden sich wiederum in der Anzahl, Größe und im Dateiformat der enthaltenen Bilder sowie in der Verwendung spezifischer Nachbearbeitungsoperationen wie der Addition von Unschärfe oder Bildrauschen.

Abbildung 5.9: Das Bild ‚Lena‘, ein Ausschnitt der 1972 im Männermagazin Playboy erschienenen Aktfotografie des schwedischen Models Lena Forsén, [Wik20]

Ein Beispiel für ein häufig in der Bildverarbeitung verwendetes authentisches Bild ist ein Fotoausschnitt des Männermagazins Playboy aus dem Jahr 1972. Das Originalbild zeigt eine Aktfotografie des schwedischen Models Lena Forsén (ehemals Söderberg), der in der Bildverarbeitung verwendete Ausschnitt zeigt lediglich den Kopf des Models (siehe Abb. 5.9). Das Bild findet deshalb oft Verwendung in der Bildverarbeitung, da es sowohl homogene Flächen als auch kontrastreiche Bildregionen enthält. Dies ermöglicht es, verschiedene Bildverarbeitungstechniken anhand dieses Bildes zu testen. Das Bild ‚Lena' wird neben den im Folgenden vorgestellten Datensätzen als Beispielbild zum Testen der implementierten Software verwendet. Es dient neben weiteren authentischen Bildern als Referenzbild zur Prüfung darauf, wie viele der zu testenden, authentischen Bilder die implementierte Software fälschlicherweise als manipulierte Bilder klassifiziert.

CoMoFoD

Bei CoMoFoD[20] handelt es sich um eine Bilddatenbank, erstellt von der Abteilung 'Video Communications Laboratory (VCL)' der Universität Zagreb. Die Datenbank umfasst insgesamt 260 Bildreihen, 200 davon enthalten Bilder im Format 512x512, die Bilder der restlichen 60 Reihen haben das Format 3000x2000 [vg. Tra+13, S. 50]. Für den Test der implementierten Software werden lediglich Erstere verwendet. Diese 200 Bildreihen, bestehend aus jeweils 52 Bildern, umfassen insgesamt 5000 authentische und 5000 manipulierte Bilder sowie 400 Bildmasken, welche die duplizierte Region markieren. Die Bilder zeigen u.a. Menschen in Alltagssituationen sowie Gebäude, Tiere und Landschaften. Sowohl auf die originalen als auch auf die manipulierten Bilder wurden folgende Nachbearbeitungsoperationen angewendet [vgl. Tra+13, S. 51]:

- **JPEG-Kompression** mit einem Qualitätsfaktor
  $Q \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$
- **Addition von Bildrauschen** mit einem Filter
  $F \in \{3x3, 5x5, 7x7\}$
- **Addition von Unschärfe** mit den Parametern
  $\mu = 0$ und $\sigma \in \{0.009, 0.005, 0.0005\}$
- **Helligkeitsänderungen** mit einer unteren und oberen Grenze
  $G_H \in \{(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)\}$
- **Farbreduktion** mit einem Intensitätslevel I pro Farbkanal mit
  $I \in \{32, 64, 128\}$
- **Kontraständerungen** mit einer unteren und oberen Grenze
  $G_K \in \{(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)\}$

Durch die Anwendung verschiedener Nachbearbeitungsoperationen mit unterschiedlichen Parametern entsteht so ein umfangreicher Bilddatensatz, mit dessen Hilfe die

[20] CoMoFoD ist die englische Abkürzung für Copy-Move Forgery Detection (dt. Copy-Move-Manipulationserkennung).

Robustheit eines Verfahrens gegenüber diesen Operationen getestet werden kann. Ein Beispiel für eine der verwendeten 200 Bildreihen ist in Anhang A1 zu sehen.

COVERAGE

Ein weiterer verwendeter Datensatz mit manipulierten Bildern ist COVERAGE[21]. Er umfasst insgesamt 200 Bilder verschiedener Größen, wobei jeweils ein authentisches und ein manipuliertes Bild ein Paar bilden. So zeigt das jeweilige manipulierte Bild eine leicht abgewandelte Version des Originals, sodass sich die Objekte der Bilder ähneln, jedoch nicht vollständig identisch sind [vgl. Wen+16, S. 161]. Ein Beispiel für ein solches Bilderpaar ist in Abbildung 5.10 zu sehen.

Für die Erzeugung der manipulierten Bilder wurden eine oder mehrere der folgenden Nachbearbeitungsoperationen auf die duplizierte Bildregion angewendet [vgl. Wen+16, S. 162]:

- Translation
- Skalierung
- Rotation
- Verzerrung
- Helligkeitsänderungen

Die Bilder des Datensatzes zeigen u.a. Alltagsgegenstände wie Möbel sowie Lebensmittel und andere Supermarktprodukte.

(a) Das Original (b) Das manipulierte Bild

Abbildung 5.10: Ein authentisches Bild des COVERAGE-Datensatzes (5.6a) und sein manipuliertes Pendant mit dupliziertem Stuhl (5.6b), [Wen16]

Während bei Bildmanipulationen anderer Datensätze kopierte Objekte willkürlich in das Bild eingefügt wurden, besteht die Besonderheit der COVERAGE-Bilder darin, dass

[21] COVERAGE ist ein Akronym für die vollständige Bezeichnung **CO**py-mo**VE** forg**ER**y d**A**tabase with similar but **G**enuine obj**E**cts [vgl. Wen+16, S. 161]

durch die Duplikation eines Objektes ein zweites Objekt desselben Bildes verdeckt wird. Die beiden Objekte weisen hier bereits vor der Manipulation eine deutliche Ähnlichkeit zueinander auf. Diese Art der Manipulation dient dem Test, ob ein Algorithmus zwischen ähnlichen und duplizierten Objekten unterscheiden kann oder ob die Klassifikation manipulierter Bilder aufgrund der Ähnlichkeit zweier Objekte falsche Ergebnisse liefert. Die Anwendung der genannten Nachbearbeitungsoperationen ermöglicht eine noch detailliertere Auswertung der Klassifikation.

MICC

Ein weiterer für diese Arbeit verwendeter Bilddatensatz ist MICC[22]. Bei diesem handelt es sich genauerweise um die drei separaten Datensätze MICC-F220, MICC-F600 und MICC-F2000. Auf die Einbeziehung des Datensatzes MICC-F600 wird hier aufgrund der enormen Größen der enthaltenen Bilder verzichtet. Diese würden die Rechenzeit des Testablaufs deutlich erhöhen.

MICC-F220 umfasst insgesamt 220 Bilder, darunter befinden sich 110 authentische sowie 110 manipulierte Bilder unterschiedlicher Auflösungen [vgl. Ame+11, S. 6]. Der Datensatz ist weiter unterteilt in 11 Bildreihen, welche jeweils aus einem Original und 10 Bildmanipulationen bestehen. Die duplizierten Bildregionen in den manipulierten Bildern wurden vor dem Wiedereinfügen in das Bild verschiedenen Operationen wie Skalierung und Rotation unterzogen. Ein Beispiel für den Aufbau einer solchen Bildreihe ist in Anhang A2 zu sehen. Motive für die Bilder des Datensatzes sind u.a. weitläufige Szenen wie die Eingangshalle eines Gebäudes oder eine gut befahrene Straße.

Bei **MICC-F2000** handelt es sich um den größten der drei Datensätze. Er umfasst insgesamt 2000 Bilder im Format 2048x1536, darunter 1300 originale und 700 manipulierte Bilder. Wie MICC-F220 ist auch dieser Datensatz weiter in Bildreihen unterteilt, hier in 50 Bildreihen mit jeweils einem Originalbild und 14 auf verschiedene Weise manipulierten Bildern. Auch hier wurden vor dem Einfügen verschiedene Nachbearbeitungsoperationen wie Skalierung und Rotation auf die kopierte Bildregion angewendet. Bei den Bildmotiven handelt es sich wiederum um Alltagsszenen, Landschaften und andere Darstellungen. Ein Beispiel ist in Anhang A3 zu sehen.

Nachdem nun alle für diese Arbeit relevanten Bilddatensätze vorgestellt wurden, bietet Tabelle 5.1 hierzu einen zusammenfassenden, detaillierten Überblick. Sie enthält u.a. weitere Informationen über das Veröffentlichungsjahr sowie über verwendete Datentypen und Bildgrößen jedes einzelnen Datensatzes.

[22] MICC ist ein Akronym für das Media Integration and Communication Center der Universität Florenz. [Ame+11, S. 1]

verwendete Kombination der Datensätze

Für den Test der implementierten Software wird aus den gerade vorgestellten Datensätzen jeweils ein Teil der Daten ausgewählt. Diese Daten werden zu einem kombinierten Datensatz zusammengefügt, welcher sowohl unterschiedliche Bildformate und -größen als auch eine Vielzahl unterschiedlich manipulierter Bilder aufweist. Auf diesem Datensatz wird die Funktionsweise der Software sowie die Qualität der Klassifikationsergebnisse getestet.

Datensatz	Jahr	original	manipuliert	gesamt	Datentypen	Bildgrößen
CoMoFoD	2013	5200	5200	10400	JPEG, PNG	512x512
COVERAGE	2016	100	100	200	TIFF	verschiedene
MICC-F220	2011	110	110	220	JPEG	verschiedene
MICC-F2000	2011	1300	700	2000	JPEG	2048x1536

Tabelle 5.1: Übersicht der für die Software verwendeten Bilddatensätze, Eigene Darstellung

Aus den Datensätzen COVERAGE und MICC-F220 werden jeweils alle authentischen und manipulierten Bilder verwendet, da die Anzahl der Bilder hier mit 200 bzw. 220 gering ausfällt. Da die Anzahl der Bilder in den Datensätzen CoMoFoD und MICC-F2000 um einiges größer ist als die der anderen Datensätze, wird von diesen nur eine kleine Teilmenge der Bilder verwendet. Aus MICC-F2000 werden willkürlich 10 Bildreihen mit insgesamt 10 authentischen Bildern und den entsprechenden 140 manipulierten Versionen ausgewählt. Aus dem Datensatz CoMoFoD werden willkürlich 10 Bildreihen mit jeweils 25 authentischen und 25 manipulierten Bildern verwendet.

Nach dem Aufsummieren der gerade ausgewählten Daten entsteht so eine Gesamtanzahl von 470 authentischen und 600 manipulierten Bildern. Um falsche Klassifikationsergebnisse aufgrund der ungleichen Verteilung von manipulierten und nicht manipulierten Bildern zu vermeiden, werden aus dem Datensatz CoMoFoD weitere 130 authentische Bilder (5 Bildreihen mit jeweils 25 Bildern sowie 5 weitere einzelne Bilder) ausgewählt. So entsteht ein finaler Datensatz mit jeweils 600 authentischen und manipulierten Bildern und einer Gesamtanzahl von 1200 Bilddaten. Anhand dieser Bilder wird nun die implementierte Software getestet. Dafür wird im Folgenden der genaue Ablauf des Tests und der Klassifikation beschrieben.

5.3.2 Testablauf

Um die statistische Genauigkeit der implementierten Software zu testen, ist es notwendig, diese für jedes Bild des in Kapitel 5.3.1 erläuterten Testdatensatzes auszuführen und die Ergebnisse der automatischen Klassifikation auszuwerten. Dieser Ablauf ist in

der Klasse *Test.java* implementiert, welche über den Testdatensatz iteriert und für jedes darin enthaltene Bild die *test()*-Methode der Klasse *Main.java* aufruft. Diese untersucht das übergebene Bild auf Copy-Move-Manipulationen und liefert ein entsprechendes Klassifikationsergebnis.

Als Parameter werden der *main()*-Methode der Klasse *Test.java* zwei Pfade übergeben. Der erste der beiden Pfade verweist auf den Ordner, welcher die zu testenden Bilder enthält. Der zweite Pfad dient als Speicherort[23] für eine durch das Testprogramm erzeugte CSV-Datei, in welcher die Klassifikationsergebnisse sowie die berechneten statistischen Gütemaße gespeichert werden. Es ist zu empfehlen, für beide Anwendungen nicht denselben Pfad zu verwenden, da das Testprogramm bei der Klassifikation der Bilder sonst auch über die CSV-Datei iteriert. Dies kann möglicherweise zu einer Fehlermeldung führen. Ein Beispiel für die in der CSV-Datei hinterlegten Ergebnisse ist in Tabelle 5.2 zu sehen. Für jedes Testbild wird der Dateiname gespeichert, gefolgt von einer Aussage über die Authentizität und das Klassifikationsergebnis eines Bildes. Zuletzt folgt die berechnete Rechenzeit, welche für die Klassifikation des jeweiligen Bildes benötigt wurde. Die in der Tabelle erwähnten Bilddateien sind in Anhang B zu sehen.

Dateiname	ist manipuliert	klassifiziert als	Rechenzeit
lena.png	nein	authentisch	20,780 s
lena_fake1.png	ja	manipuliert	22,348 s
lena_fake2.png	ja	manipuliert	22,541 s
lena_fake3.png	ja	authentisch	21,178 s
lena_fake4.png	ja	manipuliert	20,434 s
kodim02.png	nein	authentisch	28,988 s
kodim04.png	nein	authentisch	35,241 s

Tabelle 5.2: Beispiel für die Ausgabe des Testprogramms, Eigene Darstellung

Nach erfolgreicher Klassifikation der Bilder des Testdatensatzes werden anhand der in der CSV-Datei gespeicherten Ergebnisse die statistischen Gütemaße Precision, Recall, F-Maß und Accuracy berechnet. Die Berechnung erfolgt anhand der Aussage über die Authentizität und das Klassifikationsergebnis des jeweiligen Bildes.

[23] Hier muss der Ordner für die zu speichernde Datei angegeben werden.

6 Evaluation

In den vorangegangenen Kapiteln wurde die implementierte Software inklusive ihrer Komponenten, Funktionsweise und Handhabung detailliert erläutert. Anschließend folgte eine Beschreibung des Testablaufs sowie der verwendeten Testdatensätze, mit deren Hilfe die Funktionsweise und Qualität der Software geprüft wurde. Im nun folgenden Kapitel werden zunächst die Ergebnisse der Detektion und Lokalisation der einzeln implementierten Detektionsverfahren sowie der selbst implementierten Software präsentiert. Anschließend werden die Ergebnisse hinsichtlich ihrer Qualität miteinander verglichen, um eine Aussage darüber zu treffen, ob die in dieser Arbeit vorgestellte Software bessere Resultate bei der Bildmanipulationserkennung liefert als die vorgestellten Detektionsverfahren.

6.1 Qualität der Lokalisationsergebnisse

Im Folgenden sollen zuerst die Ergebnisse der Lokalisation vorgestellt und ausgewertet werden. Die Lokalisation und Markierung duplizierter Regionen dient als Hilfsmittel, um erkannte Bildmanipulationen visuell nachzuvollziehen. So kann durch den Betrachter festgestellt werden, ob es sich bei den als Bildmanipulation klassifizierten Bildern tatsächlich um Manipulationen handelt oder ob der verwendete Klassifikationsalgorithmus falsch positive Ergebnisse liefert.

6.1.1 Ergebnisse der einzelnen Detektionsverfahren

Bei den folgenden Detektionsverfahren wurde versucht, die in den jeweiligen Arbeiten beschriebene Funktionsweise korrekt zu implementieren. Die Qualität der Lokalisationsergebnisse wurde anhand der Bilder in Anlage B geprüft. Es handelt sich dabei um vier selbst generierte Bildmanipulationen des bereits vorgestellten Bilds ‘Lena‘ (siehe Abb. 5.9). In allen vier Bildmanipulationen wurde eine etwa 90x90 Pixel große, quadratische oder runde Bildregion kopiert und an einer anderen Stelle des Bildes wiedereingefügt. Bei zwei Bildern wurde die kopierte Region vor dem Wiedereinfügen zusätzlich um 90° im Uhrzeigersinn rotiert. Dadurch soll geprüft werden, ob die implementierten Verfahren robust sind gegen Rotationen im Bild.

Mahmood et al. 2018

Wie bereits erwähnt, handelt es sich bei der Arbeit von Mahmood et al. um ein blockbasiertes Detektionsverfahren für Copy-Move-Manipulationen. Bei diesem Verfahren wird das zu untersuchende Bild in 8x8 Pixel große Blöcke unterteilt, auf welche anschließend die Diskrete Kosinus-Transformation angewandt wird. Aus den entstehen-

den DCT-Koeffizienten werden Merkmalsvektoren erstellt, anhand derer jeweils zwei Bildblöcke miteinander verglichen werden können. Um die Dimension der entstehenden Merkmalsvektoren zu reduzieren wird vor der Blockzerlegung die Stationäre Wavelet-Transformation (SWT) auf das Bild angewandt, alle folgenden Operationen arbeiten mit den entstehenden SWT-Koeffizienten.

(a) Das manipulierte Bild (b) Ergebnis mit SWT (c) Ergebnis ohne SWT

Abbildung 6.1: Lokalisation duplizierter Bildregionen durch das Verfahren von Mahmood et al. 2018, Eigene Darstellung

Bei der Implementierung des Verfahrens hat sich gezeigt, dass die Verwendung der SWT falsche Ergebnisse bei der Lokalisation duplizierter Blöcke liefert (siehe Abb. 6.1b). Eine mögliche Ursache hierfür ist die fehlerhafte Implementierung der Transformation aufgrund fehlender Quellen zur Ausarbeitung der Funktionsweise der SWT. Ohne die Verwendung der SWT erkennt das Verfahren sowohl rechteckige als auch runde duplizierte Bildregionen perfekt, die Rechenzeit erhöht sich lediglich um wenige Sekunden (siehe Abb. 6.1c).

Der Nachteil der Rotationsvarianz blockbasierter Detektionsverfahren zeigt sich auch in der Implementierung des Verfahrens von Mahmood et al. In den manipulierten Bildern, deren kopierte Bildregionen vor dem Einfügen rotiert wurden, konnten trotz vorliegender Manipulation keine duplizierten Blöcke gefunden werden.

Lin et al. 2009

Auch bei der Arbeit von Lin et al. handelt es sich, wie bereits erwähnt, um ein blockbasiertes Detektionsverfahren. Hier wird das Bild in 16x16 Pixel große Blöcke und jeder einzelne Block anschließend in vier 4x4 Pixel große Unterblöcke zerlegt. Der Merkmalsvektor für jeden Block berechnet sich u.a. aus den durchschnittlichen Pixelintensitäten des gesamten Blockes sowie der Unterblöcke. Auch hier werden jeweils zwei Blöcke anhand ihrer Merkmalsvektoren miteinander verglichen, um duplizierte Regionen im Bild zu ermitteln.

Die Implementierung des Verfahrens liefert ähnlich gute Ergebnisse wie der Algorithmus von Mahmood et al. Während rechteckige und runde duplizierte Bildregionen perfekt erkannt werden, scheitert das Verfahren an der Detektion rotierter Bildausschnitte.

Dies ist erneut auf den Nachteil blockbasierter Verfahren zurückzuführen, dass rotierte Versionen der Bildblöcke nicht in die Berechnung ähnlicher Blockpaare einbezogen werden. Entsprechende Verfahren sind demnach stets rotationsvariant und liefern für entsprechende Bildmanipulation falsche Ergebnisse.

(a) Das manipulierte Bild

(b) Ergebnis des Verfahrens von Mahmood et al.

(c) Ergebnis des Verfahrens von Lin et al.

Abbildung 6.2: Lokalisation runder duplizierter Bildregionen durch verschiedene Detektionsverfahren, Eigene Darstellung

Verglichen mit dem Verfahren von Mahmood et al. weist das hier beschriebene Verfahren leicht ungenauere Lokalisationsergebnisse auf, wenn es sich bei der duplizierten Bildregion um einen runden oder ovalen Bildausschnitt handelt. Während Mahmood et al. die duplizierte Region perfekt lokalisieren, zeigen sich bei Lin et al. leichte Abweichungen von der manipulierten Bildregion. Ein Beispiel hierfür ist in Abbildung 6.2 zu sehen, in welcher der durch Lin et al. markierte Bildausschnitt (6.2c) zusätzlich einzelne Blöcke außerhalb der duplizierten, runden Region aufweist.

Cao et al. 2012 und Ustubioglu et al. 2016

Die Verfahren von Cao et al. sowie Ustubioglu et al. werden hier zusammenfassend behandelt, da bei beiden bei der Implementierung des jeweiligen Algorithmus' Fehler aufgetreten sind, für welche bisher keine Lösung gefunden werden konnte.

Das Detektionsverfahren von Cao et al. ähnelt dem von Lin et al. veröffentlichten Verfahren. Der Unterschied besteht darin, dass Cao et al. nach der Zerlegung des Testbildes die entstehenden Blöcke nicht in quadratische Unterblöcke unterteilen, hier wird jedem Block stattdessen ein innerer Kreis mit vier gleichen Kreisabschnitten zugeordnet, aus deren Eigenschaften der Merkmalsvektor eines Bildblocks berechnet wird. Jeweils zwei Blöcke werden anschließend wieder anhand ihrer Merkmalsvektoren verglichen, um duplizierte Blockpaare zu finden.

Das Verfahren von Ustubioglu et al. ist etwas komplexer als die bisher erwähnten Verfahren. Es kombiniert u.a. Farbraumtransformationen und die DCT mit weiteren mathematischen Zusammenhängen sowie automatisiert berechneten Schwellwerten zur Ein-

grenzung der Blockauswahl. Auch hier werden Blockpaare anhand ihrer berechneten Merkmalsvektoren miteinander verglichen.

Bei der Implementierung beider Verfahren sind Fehler bei der Lokalisation der manipulierten Bildregionen aufgetreten (siehe Abb. 6.3). Da sowohl Cao et al. als auch Ustubioglu et al. in der schriftlichen Dokumentation ihrer Arbeiten Beispielbilder anbringen, auf welchen der jeweilige Algorithmus fehlerfreie Ergebnisse liefert, ist davon auszugehen, dass die hier festgestellten Probleme ihren Ursprung in einer fehlerhaften Implementierung haben.

(a) Das manipulierte Bild

(b) Ergebnis des Verfahrens von Cao et al.

(c) Ergebnis des Verfahrens von Ustubioglu et al.

Abbildung 6.3: Fehlgeschlagene Lokalisation duplizierter Bildregionen durch verschiedene Detektionsverfahren, Eigene Darstellung

Die Abbildungen 6.3b) und 6.3c) zeigen die Ergebnisse der Detektion durch die Verfahren von Cao et al. sowie von Ustubioglu et al. Die Offsets der gefundenen Blockpaare sowie deren Verbindungslinie wurden hier weiß markiert.

Die Informationen eines Bildes sind i.d.R. willkürlich angeordnet, detektierbare Zusammenhänge entstehen meist erst durch die Manipulation einzelner Bildregionen. Da sich bei Cao et al. ausschließlich horizontale, beinahe parallele Linien zeigen, ist davon auszugehen, dass die gefundenen Blockpaare nicht die manipulierten Bildregionen repräsentieren. Das Ergebnis von Ustubioglu et al. zeigt hingegen eine übergroße Anzahl gefundener Blockpaare in willkürlicher Anordnung. Dies ist möglicherweise auf eine fehlerhafte Selektion relevanter Blockpaare zurückzuführen.

Da die Implementierung beider Verfahren Fehler aufweist, welche bisher nicht behoben werden konnten, finden diese keine Verwendung in der selbst implementierten Software. Der Quellcode der beiden Detektionsverfahren wurde aus Dokumentationszwecken nicht entfernt, jedoch entsprechend auskommentiert.

Yang et al. 2017

Wie bereits erwähnt, handelt es sich bei dem Verfahren von Yang et al. um ein keypointbasiertes Detektionsverfahren, welches mithilfe der Algorithmen SIFT und KAZE

Keypoints aus dem zu untersuchenden Bild extrahiert und diese anhand ihrer Deskriptorvektoren miteinander vergleicht. Anschließend wird das Bild unter Verwendung des SLIC-Algorithmus' in Regionen unregelmäßiger Superpixel unterteilt, für welche die jeweilige Anzahl an Keypoints gezählt wird. So können ausschließlich diejenigen Regionen weiter betrachtet werden, welche die meisten Keypoints beinhalten. Zur Entfernung falscher Keypointpaare kommt zuletzt der RANSAC-Algorithmus zum Einsatz.

(a) Das manipulierte Bild

(b) Ergebnis des Detektionsverfahrens

Abbildung 6.4: Lokalisation duplizierter Bildregionen durch das Verfahren von Yang et al. 2017, Eigene Darstellung

Das Ergebnis der Implementierung des Verfahrens von Yang et al. ist in Abbildung 6.4b dargestellt. Es hat sich gezeigt, dass Teile der duplizierten Region durch die Extraktion der SIFT- und KAZE-Keypoints erkannt wurden. Allerdings zeigen sich neben den korrekt detektierten Keypoints auch Keypointpaare, welche nicht in der manipulierten Bildregion liegen. Durch das implementierte Verfahren ist es demnach zwar möglich, duplizierte Regionen ausfindig zu machen, jedoch müssen falsch zugeordnete Keypointpaare nachträglich visuell oder durch die Anwendung weiterer Resampling-Algorithmen entfernt werden.

Wie bei den meisten keypointbasierten Verfahren zeigt sich auch hier der Nachteil, dass nicht alle Pixel der duplizierten Region markiert werden können, da hier lediglich mit einzelnen Bildpunkten statt auf größeren Bildregionen gearbeitet wird. Insbesondere uniforme Flächen, aus welchen aufgrund fehlender Bildinformationen kaum bis keine Keypoints extrahiert werden können, bleiben bei der Verwendung solcher Verfahren oft unentdeckt. Ein Vorteil keypointbasierter Detektionsverfahren liegt jedoch in der Detektion und Lokalisation rotierter Bildregionen. Dieser Vorteil zeigt sich auch in der Implementierung des Verfahrens von Yang et al. Zwar werden auch hier einige falsche Keypointpaare markiert, die Paare der rotierten duplizierten Region werden jedoch, im Gegensatz zu blockbasierten Detektionsverfahren, korrekt erkannt.

(a) Das manipulierte Bild

(b) Ergebnis des Detektionsverfahrens

Abbildung 6.5: Lokalisation rotierter duplizierter Bildregionen durch das Verfahren von Yang et al. 2017, Eigene Darstellung

Ulutas und Muzaffer 2016

Bei dem Verfahren von Ulutas und Muzaffer handelt es sich um ein keypointbasiertes Verfahren, welches den AKAZE-Algorithmus verwendet, um Keypoints auch aus uniformen Bildregionen zu extrahieren. Auch hier werden die extrahierten Keypoints anhand ihrer Deskriptorvektoren miteinander verglichen, um duplizierte Bildregionen zu erkennen. Falsche Keypointpaare werden erneut durch den RANSAC-Algorithmus herausgefiltert.

(a) Das manipulierte Bild

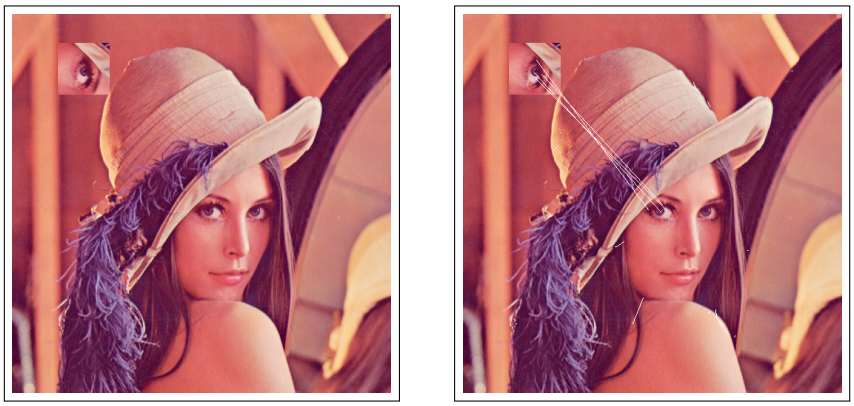
(b) Ergebnis des Detektionsverfahrens

Abbildung 6.6: Lokalisation duplizierter Bildregionen durch das Verfahren von Ulutas und Muzaffer 2016, Eigene Darstellung

Das Ergebnis der Implementierung zeigt ähnliche Ergebnisse wie der Algorithmus von Yang et al. (siehe Abb. 6.6b). Ein Teil der duplizierten Bildregionen wurde durch die Extraktion der AKAZE-Keypoints erkannt, während auch falsche Keypointpaare im Bild markiert wurden. Die Anzahl falscher Paare liegt jedoch deutlich unter der Anzahl der von Yang et al. markierten Paare. Auch die Gesamtzahl markierter Keypointpaare ist bei

dem Verfahren von Ulutas und Muzaffer wesentlich geringer als bei anderen keypointbasierten Detektionsverfahren.

Auch das implementierte Verfahren von Ulutas und Muzaffer zeigt sich robust gegenüber rotierten Bildregionen (siehe Abb. 6.7b). Der hier gewählte Bildausschnitt enthält sowohl uniforme Flächen als auch markante Informationen. Trotz des angewandten AKAZE-Algorithmus', welcher auch Keypoints von uniformen Flächen extrahieren sollte, wurden lediglich die Keypointpaare aus Regionen mit hoher Entropie erkannt. Ursache für diese Feststellung kann sowohl die Auswahl und Größe der duplizierten Region als auch die implementierte Filterung gefundener Keypointpaare sein. Ein großer Vorteil liegt hier jedoch in der geringen Anzahl falscher Keypointpaare.



(a) Das manipulierte Bild

(b) Ergebnis des Detektionsverfahrens

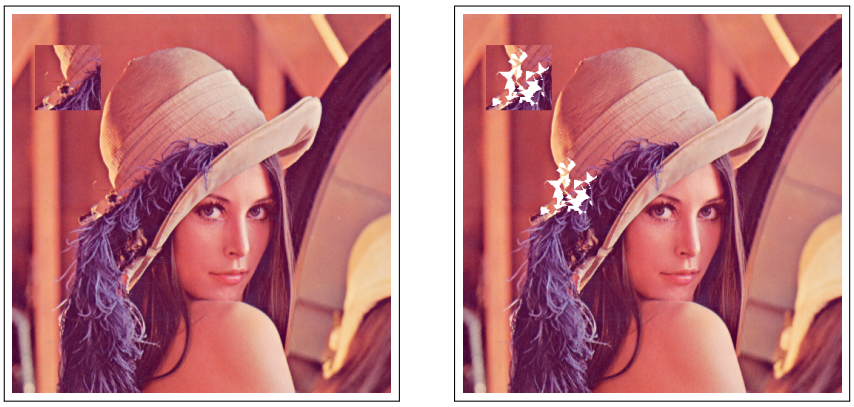
Abbildung 6.7: Lokalisation rotierter duplizierter Bildregionen durch das Verfahren von Ulutas und Muzaffer 2016, Eigene Darstellung

Ardizzone et al. 2015

Das Verfahren von Ardizzone et al. gestaltet sich etwas komplexer als die zuvor behandelten Verfahren, da es nach der Extraktion von Keypoints diese nicht anhand ihrer Deskriptorvektoren vergleicht, sondern das zu untersuchende Bild mithilfe der Delaunay-Triangulation in unregelmäßige Dreiecke unterteilt. Anschließend werden diese Dreiecke anhand ihrer Farben, Winkel und Innenflächen miteinander verglichen, um so duplizierte Regionen innerhalb des Bildes ausfindig zu machen. Detektierte Regionen werden hier demnach nicht durch Keypointpaare und deren Verbindungslinie, sondern durch unregelmäßige Dreiecke repräsentiert.

Da die bereits erwähnten Verfahren Keypoints anhand ihrer Deskriptorvektoren vergleichen und es durchaus möglich ist, dass zwei voneinander unabhängige Keypoints trotzdem ähnliche Vektoren besitzen, können hier falsch positive Lokalisationsergebnisse entstehen. Das Verfahren von Ardizzone et al. verhindert die Weiterverarbeitung falscher Keypointpaare durch die Erzeugung unregelmäßiger dreieckiger Bildregionen, welche anschließend anhand einer Vielzahl von Merkmalen miteinander verglichen wer-

den. Die Wahrscheinlichkeit, dass zwei dieser Dreiecke dieselben Merkmale besitzen, ist hier deutlich geringer als bei Verfahren, welche mit den Deskriptorvektoren einzelner Keypoints arbeiten. Das Ergebnis der Lokalisation ist in Abbildung 6.8 zu sehen. Zwar konnten nur Teile der manipulierten Bildregionen erkannt werden, das Resultatbild weist im Gegenzug dazu jedoch keine falsch markierten Pixel auf.

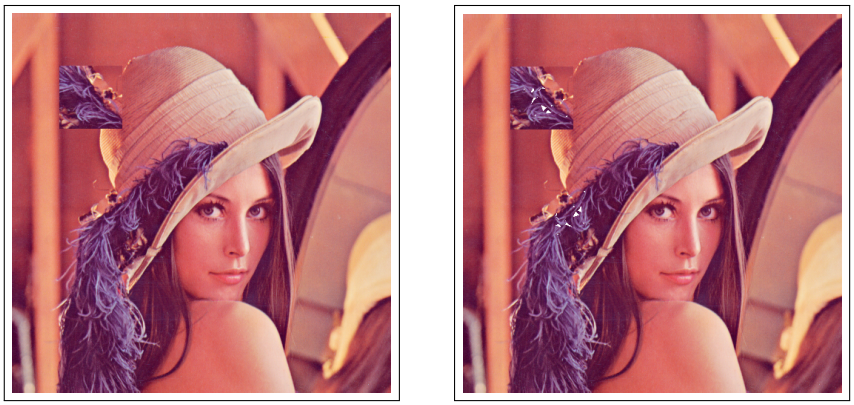


(a) Das manipulierte Bild

(b) Ergebnis des Detektionsverfahrens

Abbildung 6.8: Lokalisation duplizierter Bildregionen durch das Verfahren von Ardizzone et al. 2015, Eigene Darstellung

Die Lokalisation duplizierter Bildregionen, auf welche vor dem Wiedereinfügen zusätzlich eine Rotation angewandt wurde, zeigt ähnlich gute Ergebnisse (siehe Abb. 6.9). Zwar konnte hier nur eine geringe Anzahl an Dreieckspaaren innerhalb der manipulierten Region identifiziert werden, allerdings weist auch dieses Testergebnis keine falsch markierten Pixel auf. Es zeigt sich somit, dass auch das Verfahren von Ardizzone et al. robust ist gegenüber Rotationen der der duplizierten Bildregionen.



(a) Das manipulierte Bild

(b) Ergebnis des Detektionsverfahrens

Abbildung 6.9: Lokalisation rotierter duplizierter Bildregionen durch das Verfahren von Ardizzone et al. 2015, Eigene Darstellung

6.1.2 Ergebnisse der selbst implementierten Software

Sowohl die blockbasierten als auch die keypointbasierten Detektionsverfahren, welche im Zuge dieser Arbeit neu implementiert wurden, zeigen positive Ergebnisse bei der Lokalisation duplizierter Bildregionen[24]. Während die keypointbasierten Verfahren weniger Rechenzeit in Anspruch nehmen und robust sind gegenüber Rotationen der manipulierten Bildausschnitte, zeigen die blockbasierten Verfahren deutlich präzisere Lokalisationsergebnisse ohne falsch markierte Bildpixel. Ziel des in dieser Arbeit entwickelten Programms ist es, diese Vorteile der beiden Verfahrenstypen zu kombinieren, um die Qualität der Detektion und Lokalisation duplizierter Bildregionen zu erhöhen. Die Qualität der Lokalisation soll im Folgenden ebenfalls anhand der Testbilder aus Anhang B beurteilt werden.

Kombination der Keypoints

Der erste Ansatz für die Implementierung der geplanten Software ist die Kombination verschiedener keypointbasierter Detektionsverfahren, um möglichst viele Keypoints aus dem zu untersuchenden Bild zu extrahieren. Zu diesem Zweck werden die Keypoints, welche durch die in Kapitel 6.1.1 beschriebenen Verfahren extrahiert und miteinander verglichen wurden, in einer gemeinsamen Liste von Keypoints gespeichert. Ein Beispiel für das Ergebnis dieser Kombination ist in Abbildung 6.10b) zu sehen.

(a) Das manipulierte Bild

(b) Ergebnis des Detektionsverfahrens

(c) Ergebnis des Detektionsverfahrens

Abbildung 6.10: Extraktion von Keypoints anhand verschiedener Detektionsverfahren, Eigene Darstellung

Um falsche Keypointpaare aus der Liste zu entfernen, werden diejenigen Bildregionen ermittelt, welche die größte Anzahl an Keypoints aufweisen. Die übrigen Punkte werden aus der Liste entfernt. Das Ergebnis ist eine bereinigte Menge an Keypoints, welche die duplizierte Region genauer erfasst als die Algorithmen der einzelnen Detektionsverfahren (siehe Abb. 6.10c).

[24] Die fehlerhaft implementierten Verfahren von Cao et al. sowie von Ustubioglu et al. sind hier nicht inbegriffen.

Anwendung der blockbasierten Verfahren

Da keypointbasierte Detektionsverfahren i.d.R. weniger Rechenzeit in Anspruch nehmen als blockbasierte Verfahren, dient dieser erste Schritt des Programms der Feststellung, ob das vorliegende Bild authentisch oder manipuliert ist. Liefert diese Kombination verschiedener keypointbasierter Verfahren keine zueinander ähnlichen Keypointpaare, so ist davon auszugehen, dass es sich bei dem vorliegenden Bild nicht um eine Manipulation handelt. Andernfalls gilt es, durch die Anwendung blockbasierter Detektionsverfahren die manipulierten Bildregionen besser zu lokalisieren. Da diese nicht auf einzelnen Pixeln, sondern auf größeren Bildblöcken arbeiten, werden die Regionen um die gefundenen Keypoints herum erweitert und als separates Bild abgespeichert (siehe Abb. 6.11). Anschließend werden die in Kapitel 6.1.1 beschriebenen blockbasierten Verfahren auf diesen Bildausschnitten ausgeführt.

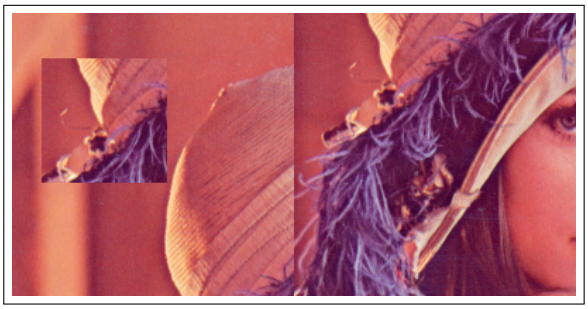


Abbildung 6.11: Beispiel der ermittelten Bildregionen für die Anwendung blockbasierter Detektionsverfahren, Eigene Darstellung

Dadurch, dass die verwendeten blockbasierten Verfahren nur noch über einen Ausschnitt des Gesamtbildes iterieren müssen, reduziert sich die Rechenzeit des Programms erheblich. Das Resultat dieser Kombination von Verfahren ist eine Liste mit einander ähnlichen Blockpaaren, welche die lokalisierte Bildmanipulation repräsentieren. Die Liste gefundener Blockpaare wird nun erneut gefiltert, um sicherzugehen, dass sie keine falschen Bildblöcke enthält. Das Ergebnis dieser Filterung ist eine um falsche Blockpaare bereinigte Liste. Als letzter Schritt des Verfahrens werden die übrigen Blockpaare im Eingangsbild farblich markiert, um die detektierte Bildmanipulation hervorzuheben.

Das Lokalisationsergebnis der implementierten Software ähnelt den Ergebnissen der blockbasierten Detektionsverfahren von Mahmood et al. sowie von Lin et al. Sowohl rechteckige als auch runde duplizierte Regionen konnten durch das Programm beinahe fehlerfrei lokalisiert werden. Zwar wird durch die Kombination von keypoint- und blockbasierten Verfahren nicht die gesamte Anzahl manipulierter Bildpixel erkannt, jedoch beschränkt sich die Markierung ausschließlich auf manipulierte Pixel, authentische Bildregionen wurden durch das Programm nicht fälschlicherweise markiert. Durch die Begrenzung der Bildregionen, über welche die blockbasierten Methoden iterieren, reduziert sich zudem die Rechenzeit des Programms.

(a) Das manipulierte Bild (b) Lokalisation rechteckiger Bildregionen (c) Lokalisation runder Bildregionen

Abbildung 6.12: Ergebnis der Lokalisation rechteckiger und runder duplizierter Bildregionen, Eigene Darstellung

Robustheit gegenüber Rotationen

Ein weiterer wichtiger Faktor bei der Implementierung eines Detektionsverfahrens für digitale Bildmanipulationen ist dessen Robustheit gegenüber verschiedenen Nachbearbeitungsoperationen. Der Fokus lag hierbei auf der Robustheit gegenüber der Rotation duplizierter Bildregionen. Während keypointbasierte Verfahren i.d.R. rotationsinvariant sind, schlagen blockbasierte Verfahren zumeist fehl, wenn eine duplizierte Bildregion vor dem Wiedereinfügen rotiert wurde. Diese Erkenntnis spiegelt sich auch im Ergebnis der in dieser Arbeit implementierten Software wider.

(a) Das manipulierte Bild (b) Extraktion der Keypoints

Abbildung 6.13: Ergebnis der Lokalisation rotierter duplizierter Bildregionen, Eigene Darstellung

Im ersten Schritt der Implementierung wurde getestet, welche Lokalisationsergebnisse die verwendete Kombination keypointbasierter Detektionsverfahren für rotierte Bildregionen liefert. Das Beispiel in Abbildung 6.13 zeigt die detektierten Keypointpaare in einer um 90° im Uhrzeigersinn rotierten Bildregion. Zwar zeigt das Ergebnis neben den korrekt identifizierten Keypointpaaren auch Keypoints außerhalb der manipulierten Regionen, das Detektionsergebnis ist jedoch als erfolgreich zu werten.

Ein ins Gegenteil ausschlagendes Ergebnis ist nach der Kombination der keypoint- und blockbasierten Verfahren zu beobachten. Hier hat sich die Vermutung bestätigt, dass rotierte Bildregionen nicht durch blockbasierte Verfahren lokalisiert werden können. Die Kombination aus keypoint- und blockbasierten Methoden stellt sich an dieser Stelle folglich als kontraproduktiv heraus, das Programm liefert im Falle von rotierten Bildregionen falsch negative Ergebnisse.

Um trotz Rotation manipulierte Bildregionen detektieren zu können, wird das Programm weiter angepasst. Wenn durch die keypointbasierten Verfahren zueinander ähnliche Keypointpaare gefunden werden konnten, wird weiter überprüft, ob auch die blockbasierten Verfahren eine Manipulation feststellen. Ist dies der Fall, so werden die gefundenen Blockpaare im getesteten Bild markiert. Sollte nur durch die keypointbasierten Methoden eine Manipulation erkannt worden sein, so werden die gefundenen Keypointpaare farblich im Bild hervorgehoben. So kann sichergestellt werden, dass auch Bildmanipulationen mit rotierten Regionen erkannt werden.

Durch diese Anpassung des Programms entsteht jedoch ein weiteres Problem. Da auch authentische Bilder Keypoints enthalten können und diese Keypoints per Zufall durch denselben Deskriptorvektor repräsentiert werden können, liefert das modifizierte Programm u.U. falsch positive Ergebnisse. Auch authentische Bilder können so fälschlicherweise als Manipulationen erkannt werden. Da für jedes authentische Bild einzeln überprüft werden müsste, wie viele Keypoints es enthält und ob es sich bei den gefundenen Keypointpaaren um zufällig detektierte Paare handelt, kann hier nicht mit einem festen Schwellwert gearbeitet werden. Durch diesen hätte andernfalls eine Anzahl an Keypointpaaren festgelegt werden können, welche in den berechneten dominanten Regionen vorhanden sein muss, um das Bild als Manipulation zu akzeptieren. Durch die Auswertung der Klassifikationsergebnisse wird sich zeigen, ob der Verdacht hoher falsch positiver Ergebnisse berechtigt ist.

6.2 Qualität der Klassifikationsergebnisse

Neben der Qualität der Lokalisationsergebnisse gilt es, auch die Ergebnisse der Klassifikation genauer zu untersuchen und anhand statistischer Gütemaße zu beurteilen, ob das implementierte Detektionsverfahren geeignet ist für das gewählte Anwendungsbeispiel. Während die Markierung manipulierter Bildregionen eine visuelle Beurteilung der Detektionsergebnisse ermöglicht, können durch die Berechnung statistischer Gütemaße verschiedene Detektionsverfahren miteinander verglichen und gegeneinander abgewogen werden. Die Beurteilung der Klassifikationsergebnisse erfolgt hier anhand des in Kapitel 5.3.1 vorgestellten Datensatzes. Dieser enthält insgesamt 1200 Bilder, davon jeweils 600 authentische und manipulierte Bilder. In den manipulierten Bildern des Datensatzes wurden jeweils eine oder mehrere Regionen dupliziert, einige Regionen wurden vor dem Wiedereinfügen zusätzlich rotiert oder skaliert.

Im Folgenden sollen die Klassifikationsergebnisse der in dieser Arbeit implementierten Software mit den Einzelergebnissen der verwendeten Detektionsverfahren verglichen werden. Anhand der in Kapitel 5.3.2 vorgestellten Testklasse wird über den Testdatensatz iteriert und für jedes darin enthaltene Bild wird die jeweilige *test()*-Methode der Klassen *Mahmood.java, Lin.java, Yang.java, Ulutas.java, Ardizzone.java* sowie *Main.java* aufgerufen. Diese überprüft die Bilder des Testdatensatzes auf duplizierte Bildregionen und berechnet die statistischen Gütemaße Precision, Recall, Falsch-Positiv-Rate, F-Maß und Accuracy sowie die für jedes Testbild benötigte Rechenzeit. Die Klassifikationsergebnisse jedes Verfahrens sowie dessen berechnete Gütemaße werden in einer Tabelle gespeichert.

Klassifikationsergebnisse

Wie bereits erwähnt, lassen sich die Klassifikationsergebnisse bei der Detektion manipulierter Bilder in vier Klassen unterteilen: richtig positiv und richtig negativ sowie falsch positiv und falsch negativ. Sie repräsentieren die Kombinationen aus der tatsächlichen Klasse eines Bildes sowie der Klasse, welche durch das Detektionsverfahren bestimmt wurde. Für jedes einzelne Detektionsverfahren sowie für die selbst implementierte Software wurde mithilfe des oben beschriebenen Testdurchlaufs die Anzahl der den vier Klassen zugeordneten Bilder bestimmt. Das Ergebnis ist in Tabelle 6.1 zu sehen.

Verfahren	Klassifikationsergebnisse			
	richtig positiv	richtig negativ	falsch positiv	falsch negativ
Mahmood et al.	420	209	390	180
Lin et al.	179	520	79	422
Yang et al.	600	5	594	1
Ulutas et al.	600	6	593	1
Ardizzone et al.	600	0	600	0
eigene Software	318	346	253	283

Tabelle 6.1: Klassifikationsergebnisse der einzelnen Detektionsverfahren und der selst implementierten Software, Eigene Darstellung

Die beiden **blockbasierten Detektionsverfahren** von Mahmood et al. sowie von Lin et al. liefern auf dem vorgegebenen Testdatensatz unterschiedliche Detektionsergebnisse. Von den insgesamt 1200 getesteten Bildern wurden hier lediglich 629 bzw. 699 Bilder richtig klassifiziert. Während bei Lin et al. ein Großteil der authentischen Bilder als solche erkannt wurden, lieferte das Verfahren von Mahmood et al. weniger richtig negative Ergebnisse. Im Gegensatz dazu weisen Lin et al. eine deutliche höhere Anzahl falsch negativer Bilder, manipulierter Bilder, welche falsch klassifiziert wurden. Wie zu erwarten, häuft sich das Vorkommen falsch negativer Ergebnisse hier insbesondere bei Bildern, deren duplizierte Bildregionen rotiert oder skaliert wurden. Grund dafür ist die Varianz blockbasierter Detektionsverfahren gegenüber solchen Nachbearbeitungsope-

rationen. Auch Bilder, auf welche JPEG-Kompression oder Rauschaddition angewandt wurden, konnten zumeist nicht richtig klassifiziert werden.

Die Klassifikationsergebnisse der **keypointbasierten Detektionsverfahren** von Yang et al., Ulutas und Muzaffer sowie von Ardizzone et al. weichen stark von denen der blockbasierten Verfahren ab. Hier wurden zwar alle manipulierten Bilder richtig klassifiziert, die Anzahl der falsch positiven Ergebnisse ist jedoch beinahe so hoch wie die der richtig positiven. Dementsprechend wurden die meisten authentischen Bilder fälschlicherweise als Manipulationen erkannt. Grund dafür ist möglicherweise die ungenaue Filterung von willkürlichen Keypointpaaren in authentischen Bildern. Da auch aus authentischen Bildern Keypoints extrahiert werden können, ist es wichtig, bei der Verwendung keypointbasierter Verfahren Schwellwerte zu definieren, durch welche die gefundenen Keypointpaare weiter gefiltert werden können. Werden diese Schwellwerte falsch gesetzt, so werden auch authentische Bilder, in welchen Keypointpaare gefunden wurden, als Bildmanipulationen eingestuft.

Ein positives Resultat ist hier hingegen der geringen Anzahl falsch negativer Ergebnisse zu entnehmen. Bei den Verfahren von Yang et al. sowie von Ulutas und Muzaffer wurde jeweils nur ein manipuliertes Bild nicht als solches erkannt, das Ergebnis von Ardizzone et al. weist sogar gar kein falsch negatives Ergebnis auf. Anzumerken ist hier jedoch, dass eine geringe Anzahl falsch negativer Ergebnisse allein nicht ausreicht, um einen Bildmanipulationsdetektor als qualitativ gut einschätzen zu können. Sowohl durch die blockbasierten als auch durch die keypointbasierten Verfahren wurde nur etwa die Hälfte der Bilder richtig klassifiziert. Da es sich bei der Detektion digitaler Bildmanipulationen um eine binäre Klassifikation handelt, liefern die hier implementierten Verfahren dementsprechend beinahe zufällige Ergebnisse. Dies zeugt von einer geringen Qualität der Klassifikationsergebnisse.

Die Detektionsergebnisse der **selbst implementierten Software** zeigen eine Kombination der Ergebnisse der einzeln getesteten Verfahren. Während die block- und keypointbasierten Verfahren eine deutliche Tendenz in Richtung der richtig und falsch negativen bzw. der richtig und falsch positiven Ergebnisse zeigten, ist die Anzahl der 1200 Testergebnisse hier in etwa gleichverteilt auf die vier Klassen. Somit werden durch die implementierte Software mehr manipulierte Bilder korrekt klassifiziert als durch die blockbasierten Verfahren. Gleichzeitig werden auch mehr authentische Bilder korrekt erkannt als durch die keypointbasierten Verfahren. Dem entgegen stehen jedoch die, im Vergleich zu den block- und keypointbasierten Verfahren, höheren Zahlen der falsch positiven bzw. falsch negativen Ergebnisse.

Insgesamt betrachtet hat die selbst implementierte Software 664 Bilder korrekt und 536 Bilder falsch klassifiziert. Damit liegt das Verhältnis der korrekt und falsch klassifizierten Bilder bei nahezu 50:50, wobei der Trend in Richtung der korrekt erkannten Bilder geht. Bei den falsch klassifizierten Ergebnissen handelt es sich insbesondere um Bilder, de-

ren duplizierte Regionen weiteren Nachbearbeitungsoperationen unterzogen wurden. Während Rotationen und Skalierungen gut detektiert wurden, scheiterte die Software an der korrekten Detektion von Bildern mit starker JPEG-Kompression oder Rauschaddition. Auch Bilder, deren Helligkeit oder Kontrast nachträglich verändert wurden, erschwerten die korrekte Klassifikation.

Will man nun anhand der implementierten Software feststellen, ob sich in einem Ordner manipulierte Bilder befinden, so liefert sie ähnlich schlechte Ergebnisse wie die einzeln implementierten Detektionsverfahren. Auch hier werden sowohl manipulierte als auch authentische Bilder beinahe zufällig klassifiziert. Grund dafür ist die Wiederverwendung der blockbasierten und keypointbasierten Verfahren, welche bereits in ihrer einzelnen Implementierung keine positiven Klassifikationsergebnisse lieferten. Zur Optimierung der selbst entwickelten Software gilt es demnach, zuerst die Funktionalität der einzelnen Verfahren zu verbessern. Hier muss ggf. die Implementierung grundlegend überarbeitet werden. Da weiterhin insbesondere nachbearbeitete Bilder falsch klassifiziert wurden, muss die Software auch in dieser Hinsicht erweitert oder angepasst werden. Das Hauptaugenmerk soll hier auf der Minimierung falsch negativer und falsch postiver Ergebnisse liegen.

Statistische Gütemaße

Aus den in Tabelle 6.1 zusammengefassten Klassifikationsergebnissen lassen sich nun verschiedene statistische Gütemaße berechnen, mithilfe derer die Qualität der implementierten Verfahren weiter bewertet und miteinander verglichen werden kann. Auch diese Berechnung wurde mithilfe der beschriebenen Testklasse durchgeführt. Die berechneten Gütemaße der einzelnen Detektionsverfahren sowie der selbst implementierten Software sind in Tabelle 6.2 zusammengefasst.

In der Praxis der Bildmanipulationserkennung muss für jedes Anwendungsbeispiel bestimmt werden, welche statistischen Gütemaße ausschlaggebend sind für die Qualität des geplanten Detektors. So kann beispielsweise bestimmt werden, wie viele der manipulierten Bilder gefunden wurden (Recall) oder wie viele der als Manipulation klassifizierten Bilder tatsächlich manipuliert sind (Precision). In anderen Anwendungsfällen kann es wiederum wichtig sein, dass auch authentische Bilder korrekt klassifiziert werden.

Für die Detektion manipulierter Bilder spielen in der Praxis insbesondere Recall und Accuracy eine wichtige Rolle. Diese geben an, wie viele der manipulierten Bilder als solche erkannt bzw. wie viele Bilder des Datensatzes korrekt klassifiziert wurden. Ziel ist es hierbei, keine manipulierten Bilder zu übersehen. Während der Recall keine Aussage über die Klassifikation der authentischen Bilder trifft, kann mithilfe der Accuracy auch die Anzahl der richtig negativen und falsch positiven Ergebnisse einbezogen werden.

Ein hoher Recall zeigt also, dass viele der manipulierten Bilder als solche erkannt wurden, gibt jedoch keine Auskunft darüber, wie viele authentische Bilder korrekt klassifiziert wurden. Eine hohe Accuracy sagt hingegen aus, dass generell viele Bilder des Datensatzes korrekt klassifiziert wurden. Viele Publikationen auf dem Gebiet der Bildmanipulationserkennung arbeiten deshalb mit der Accuracy oder beziehen bei der Auswertung des Recalls die Falsch-Positiv-Rate (FPR) mit ein, welche Auskunft darüber gibt, wie viele der authentischen Bilder fälschlicherweise als Manipulationen erkannt wurden. Ein hoher Recall ist demnach nur dann positiv zu bewerten, wenn die entsprechende Falsch-Positiv-Rate gering ausfällt. Eine hohe Accuracy geht zudem mit einem hohen Recall, einer hohen Precision sowie einer niedrigen FPR einher.

Verfahren	Precision	Recall	FPR	F_1-Maß	Accuracy	Rechenzeit
Mahmood et al.	0,519	0,700	0,651	0,596	0,525	171,127 s
Lin et al.	0,694	0,298	0,131	0,417	0,583	8,430 s
Yang et al.	0,502	0,998	0,992	0,668	0,504	35,153 s
Ulutas et al.	0,503	0,998	0,989	0,669	0,505	1,202 s
Ardizzone et al.	0,500	1,000	1,000	0,667	0,500	55,276 s
eigene Software	0,556	0,528	0,422	0,542	0,553	76,718 s

Tabelle 6.2: Statistische Gütemaße sowie durchschnittliche Rechenzeit der einzelnen Detektionsverfahren und der selbst implementierten Software, Eigene Darstellung

Die **Ergebnisse der verwendeten Verfahren** sowie der implementierten Software zeigen allesamt eine Accuracy im Bereich von 50%, durch jedes Verfahren wurde demnach in etwa die Hälfte der getesteten Bilder korrekt klassifiziert. Die Werte des Verfahrens von Lin et al. sowie der eigenen Software fielen hierbei etwas höher aus als die der restlichen Verfahren, da Letztere eine hohe Anzahl falsch positiver Detektionsergebnisse lieferten. Da die Accuracy bei allen Verfahren ähnlich ausfällt, werden diese anhand der berechneten Recall- und FPR-Werte weiter verglichen. Hier zeigen sich deutlichere Unterschiede zwischen den Verfahren als bei der Betrachtung der Accuracy.

Während der Recall aufgrund der geringen Anzahl richtig positiver Ergebnisse bei der eigenen Software und bei den blockbasierten Verfahren mittelmäßig bis gering ausfällt, erreicht dieser bei den keypointbasierten Verfahren fast durchgehend 100%, hier wurden demnach alle manipulierten Bilder als solche erkannt. Dem entgegen steht der hohe Wert der Falsch-Positiv-Rate. Auch diese liegt bei den keypointbasierten Verfahren bei fast 100%. Damit wurden zwar beinahe alle manipulierten Bilder erkannt, die Anzahl richtig negativer Ergebnisse liegt jedoch nahe des Nullbereichs. Die FPR der blockbasierten Verfahren und der eigenen Software fällt hingegen deutlich geringer aus. Abbildung 6.14 dient der Veranschaulichung des gerade beschriebenen Vergleichs.

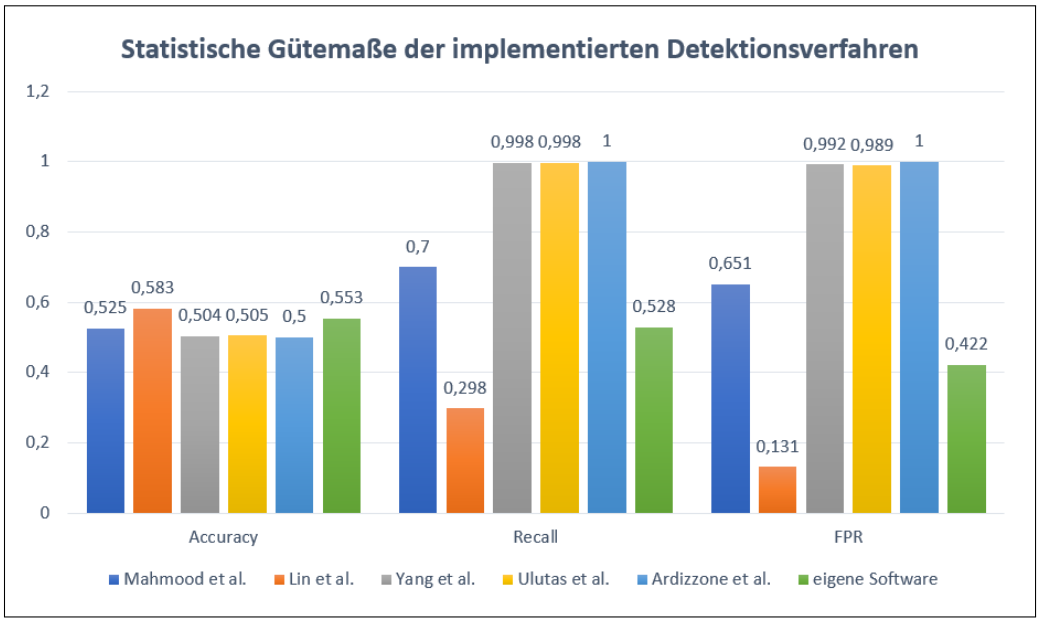


Abbildung 6.14: Vergleich der Accuracy-, Recall und FPR-Werte der implementierten Detektionsverfahren, Eigene Darstellung

6.3 Abschließende Bewertung

Nach dem Vergleich der Lokalisations- und Detektionsergebnisse der einzelnen Verfahren und der in dieser Arbeit implementierten Software folgt nun eine zusammenfassende Bewertung der Ergebnisse sowie der Eignung der Software in Bezug zu deren Verwendung in der Bildmanipulationserkennung.

Die **Lokalisationsergebnisse** zeigen, dass durch das in dieser Arbeit vorgestellte Programm duplizierte Bildregionen verschiedener Formen beinahe perfekt erkannt werden können, wenn auf diese keine weiteren Nachbearbeitungsschritte angewendet wurden. Zwar ist das Programm in der Lage, auch rotierte oder skalierte Bildregionen zu erkennen, die Lokalisation erfolgt hier jedoch über Keypoints anstatt über Bildblöcke, wodurch die manipulierten Bildregionen nicht vollständig markiert werden können. Bezüglich der Lokalisation manipulierter Bildregionen zeigt die vorgestellte Software demnach keine Verbesserungen gegenüber den verwendeten blockbasierten oder keypointbasierten Verfahren.

Verbesserungsmöglichkeiten für die implementierte Software zeigen sich insbesondere in der Anzahl falsch negativer und falsch positiver Klassifikationsergebnisse sowie in der Robustheit gegenüber weiteren Nachbearbeitungsoperationen. Während rotierte oder skalierte Bildregionen bereits teilweise erkannt werden, scheitert die Software oft an der Detektion manipulierter Bilder, auf welchen zusätzlich JPEG-Kompression, Rauschaddition, Unschärfe oder Veränderungen von Kontrast oder Helligkeit durchgeführt wurden. Auch die Lokalisation duplizierter Regionen zeigt noch keine perfekten Ergebnisse. Hier werden Teile der entsprechenden Regionen nicht korrekt erkannt. Zur

Verbesserung der Detektionsergebnisse ist es demnach zunächst erforderlich, weitere Algorithmen in die Software zu integrieren, welche eine Robustheit gegenüber derartigen Nachbearbeitungsschritten aufweisen sowie die Qualität der verwendeten Detektionsverfahren zu verbessern.

Ein weiterer optimierbarer Punkt besteht in der benötigten Rechenzeit der Software. Durch die Kombination verschiedener Algorithmen zeigt sich hier eine deutlich höhere durchschnittliche Rechenzeit als bei den einzeln implementierten Verfahren. Um die Software auch effizient auf großen Datensätze anwenden zu können, muss auch hier eine Lösung für das beschriebene Problem gefunden werden.

Zusammenfassend kann die implementierte Software in ihrer derzeitigen Form nicht als Bildmanipulationsdetektor verwendet werden, da sie kaum bessere Ergebnisse liefert als bei einer zufälligen Klassifikation. In der Praxis ist eine hohe Anzahl richtig klassifizierter Bilder jedoch ausschlaggebend für die Eignung eines Detektors. Trotz des hier gescheiterten Versuchs, blockbasierte und keypointbasierte Detektionsverfahren miteinander zu kombinieren, um die Qualität der Detektionsergebnisse zu erhöhen, sollte dieser Ansatz jedoch in zukünftigen Forschungen weiter verfolgt und optimiert werden. Die Einbindung weiterer Algorithmen, insbesondere zur Sicherstellung der Robustheit gegenüber verschiedenen Nachbearbeitungsoperationen, sollte einen weiteren Ansatzpunkt zukünftiger Forschungen darstellen.

7 Zusammenfassung und Ausblick

Im heutigen Zeitalter der Information und Digitalisierung, welches fortschreitend neue Technologien und Programme zur Manipulation digitaler Bilder bietet, wird die Authentizität digitaler Bilder immer häufiger in Frage gestellt. Da Bilder jedoch als eines der am häufigsten verwendeten Kommunikations- und Beweismittel der menschlichen Gesellschaft dienen, ist es notwendig, die Authentizität digitaler Bilder weiterhin sicherzustellen. Für die Überprüfung digitaler Bilder auf verschiedene Manipulationsmethoden wurden in der Vergangenheit deshalb zahlreiche Detektionsverfahren entwickelt.

Bei der Analyse vergangener Forschungen hat sich das Problem aufgezeigt, dass sich die Funktionalität eines Verfahrens häufig auf die Detektion einer oder weniger Bildmanipulationsmethoden beschränkt. Für einen optimalen Detektor, welcher in verschiedenen Anwendungsbereichen eingesetzt werden soll, sollte es jedoch Voraussetzung sein, ein digitales Bild auf alle bekannten Manipulationsmethoden zu untersuchen.

Ziel dieser Masterarbeit war die Planung und Implementierung eines prototypischen Detektionsverfahrens für Copy-Move-Bildmanipulationen, welches die Vorteile ausgewählter block- und keypointbasierter Verfahren miteinander kombiniert, um genauere Ergebnisse bei der Detektion digitaler Bildmanipulationen zu erzielen. Dabei wurde die Funktionalität der implementierten Software vorerst auf die am häufigsten verwendete Manipulationsmethode Copy-Move beschränkt.

Die Implementierung der Software und die ausführliche Analyse der Klassifikationsergebnisse zeigen entgegen der Erwartungen keine Verbesserungen gegenüber einzeln implementierten Detektionsverfahren. Zwar zeigen sowohl die verwendeten block- und keypointbasierten Verfahren als auch die vorgestellte Software unterschiedliche Ergebnisse bei der Klassifikation manipulierter und authentischer Bilder, die Anzahl der richtig klassifizierten Bilder liegt jedoch bei allen Verfahren kaum über 50 %. Das in dieser Arbeit entwickelte Programm liefert somit keine besseren Ergebnisse als ein zufälliger Bildmanipulationsdetektor und dementsprechend nicht als solcher in der Praxis einsetzbar.

Um in zukünftigen Forschungen auf dem vorgestellten Programm aufbauen zu können, muss dieses zuerst grundlegend überarbeitet werden. Insbesondere ist hier die Minimierung falscher Klassifikationsergebnisse sicherzustellen. Die Konzentration auf einige wenige Detektionsverfahren sowie die genauere Filterung authentischer Bilder könnten hier erste Ansatzpunkte bieten.

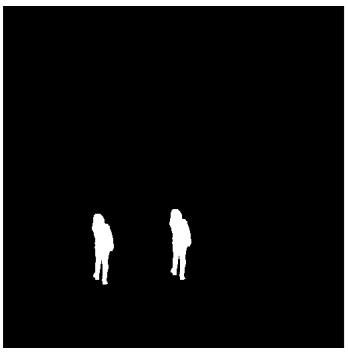
Während in dieser Arbeit insbesondere etablierte Detektionsverfahren der vergangenen 20 Jahre vorgestellt und einbezogen wurden, ist mit Blick auf zukünftige Forschungen zu betonen, dass die Verfahren der Bildmanipulationserkennung durch Einbeziehung

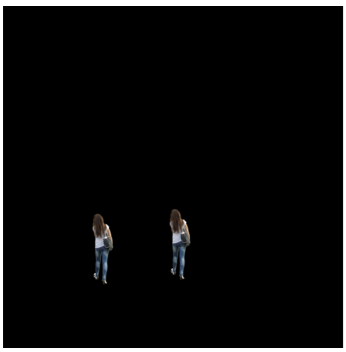
neuer Technologien, z.B. künstlicher neuronaler Netze oder weiterer Deep-Learning-Algorithmen, weiterentwickelt und optimiert werden können. Die Kombination verschiedener Ansätze sollte hier insbesondere berücksichtigt werden, da diese bei einem Großteil vergangener Forschungen vernachlässigt wurde. Trotz der größtenteils negativ ausgefallenen Ergebnisse soll diese Arbeit als Anreiz dienen, die Kombination von block- und keypointbasierten Copy-Move-Detektionsverfahren in Erwägung zu ziehen.

Anhang A: Beispiel für Bildreihen der verwendeten Datensätze

A.1 Ausgewählte Bildreihe des Datensatzes CoMoFoD

Eine Bildreihe des CoMoFoD-Datensatzes besteht aus 52 Bildern: einem Originalbild, zwei Bildmasken sowie 49 auf unterschiedliche Weise manipulierte Versionen des Originalbilds. Hier dargestellt sind einige Beispiele einer Bildreihe.







A.2 Ausgewählte Bildreihe des Datensatzes MICC-F220

Der Datensatz MICC-F220 enthält 11 Bildreihen, welche jeweils aus einem Originalbild und 10 manipulierten Versionen des Originals bestehen, sowie weitere 99 Originalbilder, zu denen keine manipulierte Version existiert. Die manipulierten Bilder zeigen eine duplizeirte Bildregion, welche zusätzlich rotiert oder skaliert wurde.







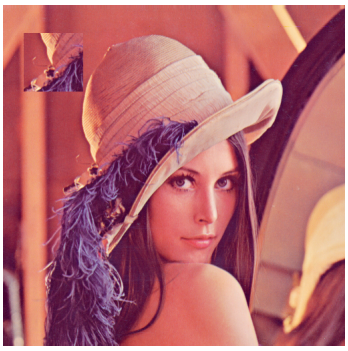



A.3 Ausgewählte Bildreihe des Datensatzes MICC-F2000

Der Datensatz MICC-F2000 enthält 50 Bildreihen mit jeweils einem Originalbild und 14 manipulierten Versionen des Originals sowie weiteren originalen Bildern, zu denen keine manipulierten Versionen existieren. Die manipulierten Bilder enthalten eine duplizierte Region, welche zusätzlich auf verschiedene Arten rotiert oder skaliert wurde.

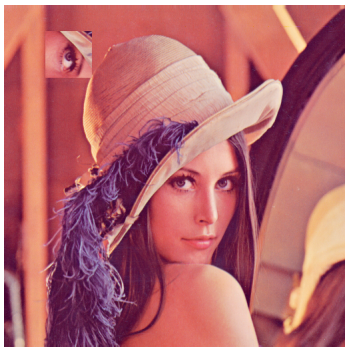
Anhang B: Bildreihe zum Testen der Lokalisationsergebnisse

(a) Das Originalbild lena_original.png



(b) lena_fake1.png

(c) lena_fake2.png



(d) lena_fake3.png

(e) lena_fake4.png

(f) Das Originalbild kodim02.png

(g) Das Originalbild kodim04.png

Abbildung B.1: Bildreihe des manipulierten Bilds 'Lena'

Literaturverzeichnis

[ABM15] Edoardo Ardizzone, Alessandro Bruno und Giuseppe Mazzola. “Copy–Move Forgery Detection by Matching Triangles of Keypoints”. In: *IEEE Transactions on Information Forensics and Security* 10.10 (2015), S. 2084–2094.

[Ame+11] Irene Amerini u. a. “A SIFT-based forensic method for copy–move attack detection and transformation recovery”. In: *IEEE Transactions on Information Forensics and Security* 6(3) (2011), S. 1099–1110.

[Ame+17] Irene Amerini u. a. “Localization of JPEG double compression through multi-domain convolutional neural networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, S. 1865–1871.

[Arm+20] Esteban Alejandro Armas Vega u. a. “Passive Image Forgery Detection Based on the Demosaicing Algorithm and JPEG Compression”. In: *IEEE Access* 8 (2020), S. 11815–11823.

[Asl20] Salman Aslam. *Facebook by the Numbers: Stats, Demographics & Fun Facts*. Jan. 2020. URL: https://www.omnicoreagency.com/facebook-statistics/ (besucht am 04. 08. 2020).

[Bak09] Paul Bakaus. *Underestimated UI techniques: Morphing*. Okt. 2009. URL: https://paulbakaus.com/?s=bush+obama&submit=Daten+absenden (besucht am 03. 08. 2020).

[Bar17] Brian Barrett. *The ’Distracted Boyfriend’ Meme’s Photographer Explains All*. Aug. 2017. URL: https://www.wired.com/story/distracted-boyfriend-meme-photographer-interview/ (besucht am 04. 08. 2020).

[BB15] Wilhelm Burger und Mark James Burge. *Digitale Bildverarbeitung - Eine algorithmische Einführung mit Java*. 3. Auflage. Berlin Heidelberg: Springer Vieweg, 2015.

[BDB13] J.P. Brooks, J.H. Dulá und E.L. Boone. “A Pure L1-norm Principal Component Analysis”. In: *Computational Statistics & Data Analysis* 61 (2013), S. 83–98.

[BG20] Zankhana J. Barad und Mukesh M. Goswami. “Image Forgery Detection using Deep Learning: A Survey”. In: *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. 2020, S. 571–576.

[BM13] Gajanan K. Birajdar und Vijay H. Mankar. “Digital image forgery detection using passive techniques: A survey”. In: *Digital Investigation* 10 (2013), S. 226–245.

[Cao+12] Yanjun Cao u. a. “A robust detection algorithm for copy-move forgery in digital images”. In: *Forensic Science International* 214 (2012), S. 33–43.

[Chr+12] Vincent Christlein u. a. “An evaluation of popular copy-move forgery detection approaches”. In: *IEEE Transactions on Information Forensics and Security* 7.6 (2012), S. 1841–1854.

[Cle20] J. Clement. *Global social networks ranked by number of users 2020*. Feb. 2020. URL: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ (besucht am 04. 08. 2020).

[CSH13] Wen-Lung Chang, Timothy K. Shih und Hui-Huang Hsu. “Detection of seam carving in JPEG images”. In: *2013 International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST 2013 & UMEDIA 2013)*. 2013, S. 632–638.

[Dar16] Darenkster. *Discrete Wavelet Transform in Java creates white Spots in the Image*. 2016. URL: https://stackoverflow.com/questions/38370691/discrete-wavelet-transform-in-java-creates-white-spots-in-the-image (besucht am 05. 08. 2020).

[Die20] Johannes Diemke. *Delaunay Triangulator*. 2020. URL: https://github.com/jdiemke/delaunay-triangulator (besucht am 05. 08. 2020).

[Dud20a] Bibliographisches Institut GmbH - Duden Verlag. *Manipulation*. 2020. URL: https://www.duden.de/rechtschreibung/Manipulation (besucht am 04. 08. 2020).

[Dud20b] Bibliographisches Institut GmbH - Duden Verlag. *chromatisch*. 2020. URL: https://www.duden.de/rechtschreibung/chromatisch (besucht am 04. 08. 2020).

[Dud20c] Bibliographisches Institut GmbH - Duden Verlag. *Aberration*. 2020. URL: https://www.duden.de/rechtschreibung/Aberration (besucht am 04. 08. 2020).

[Els20] Elsevier. *Image Tampering Detection*. 2020. URL: https://www.sciencedirect.com/search/advanced?qs=image%5C%20tampering%5C%20detection&date=2000-2020 (besucht am 04. 08. 2020).

[Far04] Hany Farid. “Creating and Detecting Doctored and Virtual Images: Implications to The Child Pornography Prevention Act”. In: *Dartmouth Computer Science Technical Report TR2004-518* (2004).

[Far09] Hany Farid. “A Survey of Image Forgery Detection”. In: *IEEE Signal Processing Magazine* 26:2 (2009), S. 16–25.

[Faw06] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern Recognition Letters* 27 (2006), S. 861–874.

[FB81] Martin A. Fischler und Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (1981), S. 381–395.

[Fon+13] Marco Fontani u. a. “A Forensic Tool for Investigating Image Forgeries”. In: *International Journal of Digital Crime and Forensics* 5(4) (2013), S. 15–33.

[For20] Digital Image Forensic. *Was ist digitale Bildforensik?* 2020. URL: https://digital-image-forensic.de/01_1.php (besucht am 05. 08. 2020).

[FSL03] Jessica Fridrich, David Soukal und Jan Lukáš. “Detection of Copy-Move Forgery in Digital Images”. In: *Proceedings of Digital Forensic Research Workshop* 3 (2003), S. 652–663.

[HF15] Olivia B. Holmes und Hany Farid. “How realistic is photorealistic?” Senior Honor Thesis. Hanover, NH: Dartmouth College, 2015.

[Hol12] Peter Holzwarth. “Menschen verändern Bilder – Bilder verändern Menschen”. In: *Medien im Kontext* Dossier Bildmanipulation (2012).

[Hol13] Peter Holzwarth. “Fotografische Wirklichkeitskonstruktion im Spannungsfeld von Bildgestaltung und Bildmanipulation”. In: *MedienPädagogik - Zeitschrift für Theorie und Praxis der Medienbildung* 23 (2013), S. 1–22.

[IEEE20] IEEE Xplore. *Image Tampering Detection*. 2020. URL: https://ieeexplore.ieee.org/search/searchresult.jsp?queryText=image%5C%20tampering%5C%20detection&highlight=true&returnType=SEARCH&matchPubs=true&returnFacets=ALL (besucht am 04. 08. 2020).

[Jet20] JetBrains. *IntelliJ IDEA: Die Java-IDE für professionelle Entwickler von JetBrains*. Aug. 2020. URL: https://www.jetbrains.com/de-de/idea/ (besucht am 10. 08. 2020).

[Lex20] Lexico Dictionaries. *Meme*. Mai 2020. URL: https://www.lexico.com/definition/meme (besucht am 04. 08. 2020).

[Low99] David G. Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Bd. 2. 1999, S. 1150–1157.

[LWK09] Hwei-Jen Lin, Chun-Wei Wang und Yang-Ta Kao. “Fast Copy-Move Forgery Detection”. In: *WSEAS Transactions on Signal Processing* 5.5 (2009), S. 188–197.

[Mah+18] Toqeer Mahmood u. a. “A robust technique for copy-move forgery detection and localization in digital images via stationary wavelet and discrete cosine transform”. In: *Journal of Communication and Image Representation* 53 (2018), S. 202–214.

[MS08] Babak Mahdian und Stanislav Saic. “Blind Methods for Detecting Image Fakery”. In: *42. Annual IEEE International Carnahan Conference on Security Technology* (2008), S. 280–286.

[MS17] Heinz-Hermann Meyer und Ansgar Schlichter. *Stockphotographie*. 2017. URL: http://filmlexikon.uni-kiel.de/index.php?action=lexikon&tag=det&id=9301 (besucht am 04. 08. 2020).

[NS16] Parameswaran Nampoothiri V und N. Sugitha. “Digital Image Forgery - A threaten to Digital Forensics”. In: *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. 2016, S. 1–6.

[NWW17] Sophie J. Nightingale, Kimberley A. Wade und Derrick G. Watson. “Can people identify original and manipulated photos of real-world scenes?” In: *Cognitive Research: Principles and Implications* 2:30 (2017).

[Open20a] OpenCV team. *About*. 2020. URL: https://opencv.org/about/ (besucht am 05. 08. 2020).

[Open20b] OpenCV team. *OpenCV 4.4.0 documentation*. 2020. URL: https://docs.opencv.org/4.4.0/ (besucht am 05. 08. 2020).

[Pag20] Pierluigi Paganini. *Digital Forensics*. Mai 2020. URL: https://www.britannica.com/topic/Digital-Forensics-2051983 (besucht am 05. 08. 2020).

[Pow11] David Martin Powers. “Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation”. In: *Journal of Machine Learning* 2(1) (2011), S. 37–63.

[Prim19] primcon GmbH. *Chromatische Aberration - Vermeiden und auf Fotos korrigieren*. Juni 2019. URL: https://www.pixolum.com/blog/fotografie/chromatische-aberration (besucht am 04. 08. 2020).

[Pyr15] Pyra Labs (Hrsg.) *Copy-Move Forgery Detection in Image - Segmentation-based Image Copy-Move Forgery Detection Scheme*. Dez. 2015. URL: http://finalyearimageprocessingprojects.blogspot.com/2015/12/copy-move-forgery-detection-in-image.html (besucht am 04. 08. 2020).

[QD15] Muhammad Ali Qureshi und Mohamed Deriche. “A bibliography of pixel-based blind image forgery detection techniques”. In: *Signal Processing: Image Communication* 39.A (2015), S. 46–74.

[Ros09] Irene Roselstorfer. “Haben Authentizität und Ethik ein Ablaufdatum? - Digitale Bildmanipulation als Herausforderung für die Medienethik”. Diplomarbeit. Wien: Universität Wien, 2009.

[Sch05] Christian Schicha. “Formen der Bildmanipulation und ihre zulässigen Grenzen”. In: *Medien-Impulse* 54 (2005), S. 9–15.

[Shu02] Jamie Shutler. *Statistical moments - An introduction*. Aug. 2002. URL: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node1.html (besucht am 05. 08. 2020).

[Spr20a] Springer Nature Switzerland AG. *Image Tampering Detection*. 2020. URL: https://link.springer.com/search?query=image+AND+tampering+AND+detection&showAll=true&date-facet-mode=between&facet-start-year=2000&previous-start-year=2020&facet-end-year=2020&previous-end-year=2020 (besucht am 04. 08. 2020).

[Spr20b] Springer Nature Switzerland AG. *Copy-Move*. 2020. URL: https://link.springer.com/search?showAll=true&query=%22copy-move%22&date-facet-mode=between&facet-start-year=2020&previous-start-year=2019&facet-end-year=2020&previous-end-year=2019 (besucht am 05. 08. 2020).

[Spr20c] Springer Nature Switzerland AG. *Image Splicing*. 2020. URL: https://link.springer.com/search?query=%22image+splicing%22&showAll=true&date-facet-mode=between&facet-start-year=2000&previous-start-year=2020&facet-end-year=2020&previous-end-year=2020 (besucht am 05. 08. 2020).

[Spr20d] Springer Nature Switzerland AG. *Image Resampling*. 2020. URL: https://link.springer.com/search?query=%22image+resampling%22&facet-start-year=2000&date-facet-mode=between&previous-end-year=2020&facet-end-year=2020&previous-start-year=2020&showAll=true (besucht am 05. 08. 2020).

[Spr20e] Springer Nature Switzerland AG. *Image Retouching*. 2020. URL: https://link.springer.com/search?showAll=true&query=%22image+retouching%22&date-facet-mode=between&facet-start-year=2000&previous-start-year=2020&facet-end-year=2020&previous-end-year=2020 (besucht am 05. 08. 2020).

[Tra+13] Dijana Tralic u. a. “CoMoFoD - New database for copy-move forgery detection”. In: *Proceedings ELMAR-2013*. 2013, S. 49–54.

[UM16] Guzin Ulutas und Gul Muzaffer. “A New Copy Move Forgery Detection Method Resistant to Object Removal with Uniform Background Forgery”. In: *Mathematical Problems in Engineering* 2016 (2016), S. 1–19.

[Ust+16] Beste Ustubioglu u. a. “A new copy move forgery detection technique with automatic threshold determination”. In: *AEU - International Journal of Electronics and Communications* 70.8 (2016), S. 1076–1087.

[WDT09] Wei Wang, Jing Dong und Tan Tieniu. “A Survey of Passive Image Tampering Detection”. In: *Digital Watermarking - 8. International Workshop (IWDW)*. Berlin Heidelberg: Springer, 2009, S. 308–322.

[Wen+16] Bihan Wen u. a. “COVERAGE — A Novel Database For Copy-Move Forgery Detection”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, S. 161–165.

[Wen16] Bihan Wen. *COVERAGE*. 2016. URL: https://onedrive.live.com/?authkey=%21ADJSupKlX%5FIj8Yc&cid=4B518F0277851508&id=4B518F0277851508%21725&parId=4B518F0277851508%21709&o=OneUp (besucht am 05. 08. 2020).

[Wik20] Wikipedia. *Lenna*. 2020. URL: https://en.wikipedia.org/w/index.php?title=Lenna&oldid=967340245 (besucht am 05. 08. 2020).

[WK18] Savita Walia und Krishan Saluja Kumar. "An Eagle-Eye View of Recent Digital Image Forgery Detection Methods". In: *Smart and Innovative Trends in Next Generation Computing Technologies (NGCT)*. Bd. 828. 2018, S. 469–487.

[Yan+17] Fan Yang u. a. "Copy-move forgery detection based on hybrid features". In: *Engineering Applications of Artificial Intelligence* 59 (2017), S. 73–83.

[Yin+15] Ting Yin u. a. "Detecting seam carving based image resizing using local binary patterns". In: *Computers & Security* 55 (2015), S. 130–141.

[ZZT18] Lilei Zheng, Ying Zhang und Vrizlynn L. L. Thing. "A Survey on Image Tampering and Its Detection in Real-world Photos". In: *Journal of Visual Communication and Image Representation* 58 (2018), S. 380–399.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 25.08.2020