Authentizitätsnachweis bei Multimediadaten

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Waldemar Berchtold aus Winogradnoje
Tag der Einreichung: 1. Oktober 2021, Tag der Prüfung: 18. Januar 2022

1. Gutachten: Prof. Dr. Michael Waidner
2. Gutachten: Prof. Dr. Jana Dittmann
3. Gutachten: Prof. Dr. Martin Steinebach
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Lehrstuhl für Sicherheit in der Informationstechnik

Authentizitätsnachweis bei Multimediadaten

Genehmigte Dissertation von Waldemar Berchtold

1. Gutachten: Prof. Dr. Michael Waidner
2. Gutachten: Prof. Dr. Jana Dittmann
3. Gutachten: Prof. Dr. Martin Steinebach

Tag der Einreichung: 1. Oktober 2021
Tag der Prüfung: 18. Januar 2022

Darmstadt
Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2022

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-210922
URL: https://tuprints.ulb.tu-darmstadt.de/21092

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 1. Oktober 2021

W. Berchtold

Zusammenfassung

In einigen Anwendungsszenarien ist das Zuordnen von Objekten, z.B. Daten und Produkten, zu ihrem Ursprung entscheidend. Beispiele dafür sind der Nachweis der Urheberschaft von Daten oder das Prüfen auf Fälschung von Produkten.

In dieser Arbeit werden Verfahren zum Authentizitätsnachweis für Mulitmediadaten vorgestellt. Es werden verschiedene Szenarien aufgezeigt, in denen die Urheberschaft oder ein Käufer nachzuweisen ist, aber auch solche, bei denen ein Originalausdruck von seiner Kopie unterschieden werden muss. Die Konzepte umfassen sowohl digitale Wasserzeichen für 3D-Modelle, Texturen und Videos als auch ein Fingerprintverfahren für bedrucktes und blankes Papier. Ein robuster Hash, entwickelt für 3D-Modelle, unterstützt bei der schnellen Suche von bekannten Modellen. Zur Bewertung der visuellen Wahrnehmbarkeit von Veränderungen bei texturierten 3D-Modellen, verursacht durch die Wasserzeichenalgorithmen für 3D-Modelle und Texturen, wurde eine Metrik entwickelt.

Das Fingerprintverfahren zeigt eine gute Robustheit bei gleichzeitig hoher Sicherheit, obwohl das Verfahren eine öffentliche Verifikation voraussetzt. Beispiele für Einflussfaktoren, bei denen das Verfahren robust ist, sind unterschiedliche Umgebungsbeleuchtung, Lichttemperatur, Smartphonemodelle, Papiertypen, Drucker und Druckverfahren. Die Sicherheit des Verfahrens wird mit einem Klonangriff demonstriert, der auf dem Ausnutzen des öffentlichen Wissens basiert und erfolgreich erkannt werden kann. Bei einer öffentlichen Verifikation stehen alle algorithmischen Schritte öffentliche zur Verfügung und können entsprechend von Angreifern genutzt werden. Das Verfahren beruht auf der Auswertung der Fleckigkeit und Körnigkeit einer einheitlichen Region.

Das Wasserzeichenverfahren für 3D-Modelle ist robust gegenüber typischen Nachverarbeitungsschritten und sicher gegen einige der getesteten Angriffe, jedoch stellen sowohl das ungleichmäßige Skalieren als auch die Modellvereinfachung Herausforderungen für das Verfahren dar. Das Verfahren verändert die Knotenverteilung eines 3D-Modells bei der Einbettung des Wasserzeichens. Die Veränderungen erfolgen auf dem spektral komprimierten 3D-Modell. Ein evolutionärer Optimierer bettet iterativ die Nachricht ein.

Das Texturwasserzeichen ist sicher gegen Angriffe und Nachverarbeitungsschritte bei gleichzeitiger Nicht-Wahrnehmbarkeit. Der Algorithmus zum Einbetten der Wasserzeichen in Texturen arbeitet auf den komprimierten Daten und ist ein blockbasierter Ansatz. Die

Veränderungen werden auf der Ebene der Indizes innerhalb eines Blockes erlaubt. Gibt es nach dem Verändern von Indizes keinen kleinsten oder größten Wert für einen Block, wird eine Aktualisierung durchgeführt, um so die Robustheit des Wasserzeichens zu sichern.

Das Videowasserzeichenverfahren ist invariant gegen verschiedene Angriffe und Nachverarbeitungsschritte bei gleichzeitig nicht-wahrnehmbaren Veränderungen. Das Wasserzeichen kann selbst nach dem Abfilmen mit Smartphones gelesen werden. Es basiert auf Maximal Stabilen Extrem Regionen zum synchronisieren zwischen dem Einbettungs- und Auslesealgorithmus. Um diese extrahierte Region wird die kleinste umschließende Ellipse berechnet, um einen festen Faktor vergrößert und ein vorher kreiertes Muster auf die Größe der Ellipse skaliert und addiert. Der Detektionsalgorithmus bestimmt über die Berechnung der Korrelation zwischen dem verwendeten Muster und der Region des Frames, welches Bit eingebettet wurde. Zum Sichern der visuellen Qualität nutzt der Algorithmus einen Laplace-Filter und eine Szenenerkennung.

Der robuste Hash für 3D-Modelle ist invariant gegen die drei Nachverarbeitungsschritte Rotation, gleichmäßige Skalierung und Translation. Außerdem stellen das Addieren von zufälligem Rauschen, Vereinfachen, Verschieben und Tessellieren keine große Herausforderung für das Verfahren dar. Die Fehlerraten des Verfahrens liegen in sehr niedrigen Bereichen, sodass sie sich zur Voranalyse eignen, bevor die Wasserzeichendetektion durchgeführt wird. In der Vorgehensweise des robusten Hashverfahrens wird zuerst ein robuster Bezugspunkt ermittelt und mithilfe dessen werden die Knoten in ihre sphärische Koordinatenrepräsentation transformiert. Es werden dann Sphären definiert und daraus Bins erzeugt. Jedes Bin wird zu seinem übergeordneten Bin ins Verhältnis gesetzt und daraus ein Hashbit extrahiert.

Die Metrik zur Bewertung der Nicht-Wahrnehmbarkeit des Wasserzeichens bei texturierten 3D-Modellen berücksichtigt die subjektive Wahrnehmung eines menschlichen Betrachters. Sie erweitert und kombiniert jeweils eine Metrik aus dem 2D und 3D-Bereich und bezieht verschiedene messbare Größen wie die Rauheit, Flächenänderung und Verzerrung bei 3D-Modellen in ihre Berechnungen mit ein.

Abstract

In some application scenarios, assigning objects, e.g. data and products, to their origin is crucial. Examples include proving the authorship of data or checking for counterfeit products.

In this work, methods for proving authenticity for multimedia data are presented. Different scenarios are shown, in which a copyright holder or a consumer has to be proven, but also in which an original print has to be distinguished from its copies. Concepts include digital watermarking for 3D models, textures and videos, as well as a fingerprinting method for printed and blank paper. A robust hash, developed for 3D models, assists in the rapid retrieval of known models. A metric was developed to evaluate the visual perceptibility of changes in textured 3D models caused by the watermarking algorithms for 3D models and textures.

The fingerprinting method shows good robustness while maintaining high security, although the method requires public verification. Examples of influencing factors for which the method is robust include varying ambient lighting, light temperature, smartphone, paper, printer, and printing method. The security of the procedure is demonstrated with a cloning attack based on exploiting public knowledge that can be successfully detected. In a public verification, all algorithmic steps are publicly available and can be exploited accordingly by attackers. The method is based on evaluating the patchiness and granularity of a uniform region.

The watermarking method for 3D models is robust to typical post-processing steps and secure against some of the attacks tested, but non-uniform scaling as well as model simplification present challenges for the method. The method changes the nodal distribution of a 3D model when embedding the watermark. The changes occur on the spectrally compressed 3D model. An evolutionary optimizer iteratively embeds the message.

The texture watermark is secure against attacks and post-processing steps while remaining imperceptible. The texture watermark embedding algorithm works on the compressed data and is a block-based approach. The changes are allowed at the level of indices within a block. If there is no smallest or largest value for a block after changing indexes, an update is performed, thus ensuring the robustness of the watermark.

The video watermarking method is invariant to various attacks and post-processing steps while maintaining non-perceptible changes. The watermark can be read even after filming with smartphones. It is based on Maximal Stable Extreme Regions to synchronize between the embedding and readout algorithms. Around this extracted region, the smallest enclosing ellipse is calculated, enlarged by a fixed factor and a previously created pattern is scaled to the size of the ellipse and added. The detection algorithm determines which bit was embedded by calculating the correlation between the pattern used and the region of the frame. In order to ensure visual quality, the algorithm uses a Laplace filter and scene detection.

The robust hash for 3D models is invariant to the three post-processing steps of rotation, uniform scaling and translation. Moreover, adding random noise, simplifying, shifting, and tessellating do not pose much of a challenge to the method. The error rates of the method are in very low ranges, making it suitable for pre-analysis of its own before watermark detection is performed. In the robust hashing procedure, a robust reference point is first determined and used to transform the nodes into their spherical coordinate representation. Spheres are then defined and bins are generated from them. Each bin is related to its parent bin and a hash bit is extracted from it.

The metric for evaluating the transparency of the watermark in textured 3D models takes into account the subjective perception of a human viewer. It extends and combines one metric from each of the 2D and 3D domains and incorporates various measurable quantities, such as the roughness, surface change and distortion in 3D models into its calculations.

Danksagung

Danken möchte ich verschiedenen Personen, die einen Beitrag zu dieser Arbeit geleistet haben. Zunächst möchte ich meinem Doktorvater Prof. Dr. Michael Waidner und meinen Korreferenten Prof. Dr. Jana Dittmann und Prof. Dr. Martin Steinebach für die Betreuung meiner Forschungsarbeit meinen Dank aussprechen. Ein besonderer Dank gilt meinem Mentor Prof. Dr. Martin Steinebach für die Zeit der Betreuung, die inspirierenden Diskussionen und konstruktive Ratschläge.

Meinen Kollegen der Abteilung für Multimediasicherheit und IT-Forensik am Fraunhofer SIT danke ich für die guten Diskussionen und das Feedback. Besonders möchte ich den Kollegen Dr. Oren Halvani und Prof. Dr. Rüdiger Grimm für die Korrekturen des Manuskripts und die wertvollen Ratschläge danken. Darüber hinaus haben mich während der gesamten Zeit viele Studenten im Rahmen von Diplom-, Bachelor- und Masterarbeiten sowie studentische Hilfskräfte unterstützt, die in den entsprechenden Kapiteln namentlich genannt und deren Arbeiten referenziert werden.

Meiner Frau Laura und meinen Kindern Lennart, Linnea und Linett danke ich für die Liebe, Ermutigungen und Geduld während der gesamten Zeit der Arbeit.

Inhaltsverzeichnis

Danksagung **ix**

1 Einleitung **1**
1.1 Motivation 2
1.2 Historie 3
1.3 Ziele 4
1.4 Methodik und Gliederung 5

2 Grundlagen **7**
2.1 Authentizität 7
2.2 Eigenschaften von Verfahren zum Authentizitätsnachweis 8
2.3 Grundkonzepte 9
2.3.1 Digitale Wasserzeichen 9
2.3.2 Robuste Hashverfahren 10
2.3.3 Fingerprintverfahren 11
2.4 Wavelet-Transformation 11
2.5 3D-Modelle 12
2.6 Bezugspunkt in 3D-Modellen 12
2.6.1 Graviationszentrum 13
2.6.2 Kleinster umschließender Kreis 13
2.7 Rauheit eines 3D-Modells 13
2.8 Sonstiges 14
2.8.1 Konfusionsmatrix 14
2.8.2 Drucktechniken 15
2.8.3 Farbraum 16

3 Anforderungen **17**
3.1 Fingerprint für Produkt- und Dokumentenauthentifizierung 17
3.2 3D-Modell-Wasserzeichen 18
3.3 Texturwasserzeichen 19

3.4 Metrik zur Bewertung der Nicht-Wahrnehmbarkeit der Wasserzeichen texturierter 3D-Modelle . . . 20
3.5 Robuster Hash für 3D-Modelle . . . 21
3.6 Videowasserzeichen . . . 22

4 Fingerprint für Produkt- und Dokumentauthentifizierung 23
4.1 Fingerprintalgorithmus . . . 24
4.1.1 Variationen in homogenen Flächen . . . 24
4.1.2 Bildaufnahme und Fingerabdruck-Extraktion . . . 25
4.1.3 Waveletzerlegung . . . 26
4.1.4 Fingerabdruckvergleich . . . 26
4.1.5 Fingerabdruck Homogenität . . . 27
4.2 Evaluierung . . . 28
4.2.1 Testaufbau . . . 28
4.2.2 Parameterwahl . . . 29
4.2.3 Substrateinfluss . . . 33
4.2.4 Robustheit . . . 35
4.2.5 Sicherheit . . . 39
4.3 Zusammenfassung . . . 41

5 3D-Modell-Wasserzeichen 43
5.1 3D-Wasserzeichenalgorithmus . . . 44
5.1.1 Vorverarbeitungsschritt . . . 44
5.1.2 Histogramm-Modifikation . . . 49
5.1.3 Evolutionärer Optimierer . . . 50
5.1.4 Rekonstruktion . . . 58
5.1.5 Detektieren des 3D-Modell Wasserzeichens . . . 58
5.2 Evaluierung . . . 59
5.2.1 Testaufbau . . . 59
5.2.2 Evaluierung der Nicht-Wahrnehmbarkeit . . . 60
5.2.3 Evaluierung der Sicherheit und Robustheit . . . 60
5.2.4 Performanz Analyse . . . 65
5.3 Zusammenfassung . . . 66

6 Texturwasserzeichen 69
6.1 DDS Texturwasserzeichenalgorithmus . . . 70
6.1.1 Grundlagen des Texturwasserzeichenalgorithmus . . . 70
6.1.2 Blockauswahl . . . 71

6.1.3 Einbettungsregeln 72
6.1.4 Pixelauswahl und Flippen der Pixel 73
6.1.5 Blockaktualisierung 74
6.1.6 Auslesen des Wasserzeichens 75
6.2 Tests und Ergebnisse 77
6.2.1 Testaufbau 77
6.2.2 Visuelle Qualität 77
6.2.3 Evaluierung der Robustheit und Sicherheit 78
6.3 Zusammenfassung 83

7 Metrik zur Bewertung der Nicht-Wahrnehmbarkeit der Wasserzeichen bei texturierten 3D-Modellen 87
7.1 Metrik zur Bewertung der Nicht-Wahrnehmbarkeit 88
7.1.1 Konzept 88
7.1.2 Anpassung und Optimierung der 2D-Metrik 89
7.1.3 Anpassung der 3D-Metrik 90
7.1.4 Zusammenführen der 2D und 3D-Metrik 92
7.2 Evaluierung 94
7.2.1 Vergleich zu MSSIM 94
7.2.2 Evaluierung der 3D-Einflussgrößen 95
7.3 Zusammenfassung 99

8 Robustes Hashverfahren für 3D-Modelle 101
8.1 Robustes Hashverfahren 102
8.1.1 Erzeugen von Bins 103
8.1.2 Hash Extraktion 105
8.2 Evaluierung 105
8.2.1 Hash Authentifizierung 106
8.2.2 Testaufbau 106
8.2.3 Angriffe 107
8.2.4 Evaluierung der Sicherheit, Performanz und Eindeutigkeit 108
8.2.5 Fehlerraten 114
8.3 Zusammenfassung 115

9 Videowasserzeichen 117
9.1 Videowasserzeichen Algorithmus 118
9.1.1 Kreisförmiges Wasserzeichenmuster 119
9.1.2 Analyse der Frames 121

9.1.3 Angepasste MSER Detektion 121
9.1.4 Auswahl der Ellipsen 124
9.1.5 Verstärken der MSER 125
9.1.6 Einbetten der Wasserzeichennachricht 126
9.1.7 Synchronisation 126
9.1.8 Wasserzeichen Detektion 127
9.2 Evaluierung 127
9.2.1 Testaufbau 128
9.2.2 Sicherheit und Robusheit 128
9.2.3 Nicht-Wahrnehmbarkeit 129
9.2.4 Diskussion und Bewertung der Evaluationsergebnisse 130
9.3 Zusammenfassung 131

10 Zusammenfassung und Ausblick 133

Abbildungsverzeichnis

2.1 Beispiel eines zu bewertenden Knotens. Dabei ist v_l der zu bewertende Knoten, $M_{v_l} = 4$ Dreiecke, die den Knoten gemeinsam haben und $N^{(k)}$ sind die Normalenvektoren der Dreiecke 14

4.1 Testbild als homogene Cyan-Region mit vier Suchmustern in den Ecken. . 25
4.2 Binarisierte Fingerabdrücke: Links Canon IP, Mittig HP Laserjet und rechts MAN Roland. 28
4.3 Korrelationen der AR und RR Aufnahmen der Cyan-Blöcke. 30
4.4 Reflexionsbild von Cyan-Blöcken, die mit den evaluierten Druckern und Druckmaschinen gedruckt und bei der Aufnahme mit dem Blitz des Smartphones ausgeleuchtet wurden. 33
4.5 Level 3 und 4 aus den Aufnahmen aus Abildung 4.4. Das Histogramm ist zur Visualisierung gestreckt worden. 34
4.6 Korrelationen eines Fingerabdrucks eines Substratblocks mit dem Fingerabdruck nach dem Überdrucken mit einer Farbe. YC steht für die Druckreihenfolge Yellow dann Cyan und bei YCM wurde noch Magenta über Yellow und Cyan gedruckt. 35
4.7 Die Korrelationen der Fingerabdrücke werden hier in Abhängigkeit von verschiedenen Druckern und unter verschiedenen Umgebungsbeleuchtungen abgetragen. Low-Light gibt das Ergebnis bei ausgeschaltetem Umgebungslicht in einem dunklen Raum wider, während “Office“ die Aufnahmen unter Neonbeleuchtung eines Büros zeigt. Die Ergebnisse für die Lichttemperatur von 3000, 4000, 5000 und 6500 Kelvin wurden mit zwei Schreibtischlampen des Typs TaoTronics TT-DL27 durchgeführt. 36
4.8 Die Korrelationen der Fingerabdrücke werden hier in 15° Schritten durchgeführt und als blaue Punkte an dem entsprechenden Winkel abgetragen. Links werden neben den Korrelationen der Median in Grün und der Schwellwert in Rot für einen Laserdruck und rechts für einen Tintenstrahldruck gezeigt. 37

4.9 Die drei Heatmaps zeigen links die Korrelationen der Fingerabdrücke für einen Tintenstrahldrucker, mittig bei einem Laserdrucker und rechts für einen Offsetdrucker. Auf beiden Achsen wird die Abweichung der Ausrichtung vom Zentrum der Region abgetragen, wobei Blau eine niedrige Korrelation darstellt und Orange und Rot hohe Korrelationen. 38
4.10 Korrelation der aufgenommenen Drucke in Abhängigkeit der verschiedenen Drucker und Abstände zu den Regionen. Bei den Drucken mit den Druckmaschinen ist ein stärkerer Abfall der Korrelationen abzulesen als bei den Druckern. 40

5.1 Überblick des Wasserzeichenalgorithmus für 3D-Modelle 44
5.2 Berechnung des Bezugspunktes für das “Dog“ Modell mithilfe des Iterativen SEB 47
5.3 x, y und z-Koordinaten des stabilen Bezugspunkt in Abhängigkeit der Anzahl der Iterationen nach dem Gewichten 48
5.4 Beispielhafte Zuordnung der Bins zu den Gruppen eines bestimmten Wasserzeichenbits 51
5.5 Einbettungsstrategie 52
5.6 Performanzanalyse des 3D-Modell Wasserzeichenembedders 66
5.7 Performanzanalyse des 3D-Modell Wasserzeichendetektors 67
5.8 Tiefgreifendere Performanzanalyse des 3D-Modell Wasserzeichenembedders, die den Rechenaufwand des Optimierers weiter unterteilt 68

6.1 Überblick der Indexsequenz entsprechend der Helligkeitswerte gemeinsam mit den Möglichkeiten, die Indizes zu flippen. 72
6.2 Übersicht, wie die Bereiche für jedes der 16 Pixel gewählt werden, wobei das schwarze Pixel das potentiell zu flippende Pixel ist. Seine Nachbarn sind grau eingefärbt. 73
6.3 Flippen der Indizes basierend auf den Variationswerten der beiden Möglichkeiten, in die der Algorithmus flippen darf. 74
6.4 Dichtefunktion der Relation zwischen den beiden Gruppen γ'_{α_i} und γ'_{β_i} aus unmarkiertem Content, die das Verhältnis der Indizes ’0’ und ’3’ der beiden Gruppen zueinander darstellen. Die Ordinate zeigt die relative Häufigkeit der entsprechenden Gruppenverhältnisse, die auf der Abszissenachse abgetragen sind. Die gestrichelte Linie entspricht links $-\tau$ und rechts τ. . . 76
6.5 Die Detektionsrate ist nach verschiedenen JPEG-Kompressionen und unterschiedlichen Einbettungsstärken w_{int} auf der Abzisse in Abhängigkeit der Anzahl der Nachrichtenbit abgetragen. 79

6.6 Links die originale Textur aus dem Videospiel SKYRIM und rechts die Textur nach einem Hintergrund-Rausch Angriff. 80
6.7 Eine Szene aus dem Videospiel SKYRIM , die die Textur aus Abbildung 6.6 beinhaltet. Im oberen Spielausschnitt wurde die originale Textur verwendet; unten die Textur nach einem Hintergrund-Rausch Angriff 84
6.8 Die Textur (bonecrown.dds) aus SKYRIM mit den sechs unterschiedlichen Mip-Map-Level, die auf $512x512$ hoch skaliert wurden. Eine höheres Mip-Map Level bedeutet die halbe Kantenlänge des unteren Levels. 85

7.1 Überblick des Konzepts zur Qualitätsbewertung 89
7.2 Textur “iron battle axe“ aus SKYRIM , wobei links die mit 10% JPEG komprimierte und rechts die entsprechende originale Textur abgebildet ist. 94
7.3 Textur “Hairlong“ aus SKYRIM , wobei links die Texturen und rechts die zugehörigen Spielszenen sind. Oben befindet sich die originale Textur, unten die modifizierte, jeweils als Graustufenbild. 96

8.1 Überblick des robusten Hashverfahrens für 3D-Modelle 103
8.2 Bestimmen von fünf Sphären, um daraus vier Bins b_i zu erhalten, die eine Breite von d_r haben. 105
8.3 Links ist die normalisierte Hammingdistanz abhängig von der Stärke des addierten zufälligen Rauschens abgebildet. Rechts ist die normalisierte Hammingdistanz nach dem Vereinfachungsangriff abgebildet. 109
8.4 Links ist die normalisierte Hammingdistanz abhängig der prozentualen Krümmung abgebildet und rechts die normalisierte Hammingdistanz nach dem ungleichmäßigen Skalieren des 3D-Modells. 110
8.5 Links ist die normalisierte Hammingdistanz abhängig der Streckung in einer Dimension abgebildet und rechts die normalisierte Hammingdistanz nach der Tessellierung, wobei sich die prozentuale Angabe auf die Anzahl der Knoten des originalen Modells bezieht. 111
8.6 Links, das originale 3D-Modell *Smbehemoth* des Videospiels Fallout III und rechts, das mit zufälligem Rauschen der Stärke 3.0 angegriffene Modell. . 111
8.7 Links, das originale 3D-Modell *Smbehemoth* des Videospiels Fallout III und rechts, das mit der Intensität 60 gekrümmte 3D-Modell. 112
8.8 Links, das originale 3D-Modell *Smbehemoth* des Videospiels Fallout III und rechts, das um den Faktor 0.5 gestreckte 3D-Modell. 112

9.1 Überblick des vorgestellten Videowasserzeichenalgorithmus 119

9.2 Beispielhafte Muster, die zum Einbetten eines Wasserzeichens in eine MSER eines Videoframes genutzt werden. Links das Urmuster, in der Mitte das daraus abgeleitete Muster mit dem Durchmesser von 256Pixel und rechts das aus dem Urmuster abgeleitete Muster mit dem Durchmesser von 128 Pixel 120
9.3 Beispielhafte Analyse eines Frames anhand dem Bild "Lena". 122
9.4 Muster zum Verstärken der MSER eines Videoframes 125

Tabellenverzeichnis

4.1 Skalierungslevel der Wavelettransformation mit den entsprechend geforderten Größen, Abtastraten und Frequenzbändern 27
4.2 Score der Blöcke für jedes Wavelet Level für Konsumerdrucker 31
4.3 Score nach Gleichung 4.8 der Cyan- und Substratblöcke für jedes Wavelet-Level von Ausdrucken der fünf berücksichtigten Druckmaschinen 32
4.4 Korrelation und Homogenität von Substrat- und Cyanblöcken für zwei Drucker. Die Korrelation und die Homogenität übersteigen die jeweiligen Schwellwerte. Somit sind die Klone nicht authentifiziert und als Fälschung erkannt. 41
4.5 Korrelation und Homogenität von Substrat- und Cyanblöcken für zwei Drucker. Die Korrelation und die Homogenität übersteigen die jeweiligen Schwellwerte. Somit sind die Klone nicht authentifiziert und als Fälschung erkannt. 41

5.1 Überblick der in der Evaluierung verwendeten Parameter 60
5.2 Evaluierung der Nicht-Wahrnehmbarkeit der wasserzeichenmarkierten 3D-Modelle im Vergleich zu ihren originalen 3D-Modellen 60
5.3 Evaluationsergebnisse nach Angriffen der Kategorie geometrischer Transformationen 62
5.4 Evaluationsergebnisse nach dem Addieren von zufälligem Rauschen 64
5.5 Evaluationsergebnisse nach dem Vereinfachungsangriff mit unterschiedlichen Faktoren 65

6.1 Durchschnittliche PSNR Werte über die gesamte Testbasis. Die PSNR Werte sind in (db) angegeben 78

6.2 Evaluationsergebnisse nach dem Mip-Map basierten Angriff. Mit M_1 ist das Hochskalieren aus dem Mip-Map Level 1 gemeint, mit M_2 das Hochskalieren von Level 2 und mit M_3 von Level 3. “Korrekt“ bedeutet das richtige Wiedererkennen des eingebetteten Bits. “?“ bedeutet, dass das Bit nicht zuweisbar ist, und somit nicht klar ist, ob das Bit eine “0“ oder “1“ sein soll. “Falsch“ bedeutet, dass das Wasserzeichenbit falsch detektiert wurde. 82

7.1 Evaluationsergebnisse $MSSIM$ vs. $MSSIM_\alpha$ 95
7.2 Ergebnisse der 2D-Metriken $MSSIM$ und $MSSIM_\alpha$, nachdem die transparenten Pixel auf den Farbwert weiß gesetzt wurden. 95
7.3 Evaluierungsergebnisse zum Einfluss der Rauheit auf die Gesamtmetrik. Diese zeigt bei Veränderungen von Texturen, die zu rauen 3D-Modellen gehören, einen höheren Wert und damit einen geringeren Einfluss auf die visuelle Qualität. 97
7.4 Evaluierungsergebnisse zum Einfluss der Flächenänderung und Verzerrung auf die Gesamtmetrik. Diese zeigt bei einer JPEG Kompression von Texturen, die auf verzerrte 3D-Modell gemapped wurde, einen höheren Wert und damit einen geringeren Einfluss auf die visuelle Qualität haben. 98
7.5 Evaluierungsergebnisse zum Einfluss der Wasserzeichenmarkierung von 3D-Modellen und DDS Texturen. 99

8.1 Übersicht zur Anzahl der Knoten der im Testset verwendeten 3D-Modelle aus den unterschiedlichen Bereichen 107

9.1 Evaluationsergebnisse der BER nach den entsprechenden Angriffen 129
9.2 Evaluationsergebnisse der SSI Metrik nach Trick und Thiemert [98] 129

1 Einleitung

Heutzutage ist es einfach Multimediainhalte zu erzeugen, kopieren, verändern und verteilen. Software- und Hardwareunterstützt können sehr einfach auch nicht digital vorliegende Inhalte kopiert und verbreitet werden. Der Schutz von Multimediainhalten führt zur Entwicklung des Urheberrechtsschutzes und der Verwaltung digitaler Rechte (DRM). Während DRM die Nutzung digitaler Medien kontrolliert, werden digitale Wasserzeichen genutzt um das Eigentum oder das Urheberrecht an Medieninhalten zu identifizieren und zu schützen, indem Daten für einen Menschen nicht wahrnehmbar in den Originalinhalt eingebettet werden. Digitale Wasserzeichen können als fragil und robust kategorisiert werden. Fragile Wasserzeichen eignen sich zur Feststellung der Integrität der Daten, weil kleinste Veränderungen erkannt werden. Robuste Wasserzeichen eignen sich zur Authentifizierung, da selbst nach gewissen Änderungen das Wasserzeichen korrekt gelesen werden kann. Einigen Anwendungen erfordern die Klärung der Frage nach dem Ursprung, z.B. wer der Urheber eines Werkes ist oder ob eine Eintrittskarte original ist. Durch das unrechtmäßige Verbreiten von Multimediainhalten entsteht ein Schaden für Rechteinhaber und Endverbraucher. Daraus ergibt sich die Notwendigkeit zum Nachweis und Prüfung der Authentizität von Multimediainhalten.

Die vorliegende Arbeit behandelt Verfahren zum Authentizitätsnachweis auf Basis von digitalen Wasserzeichen zum Urheberrechtsschutz für 3D-Modelle, Texturen und Videos und einem robusten Hashverfahren für 3D-Modelle sowie einem Fingerprintverfahren für Papier. Zum Messen und Sicherstellen der Nicht-Wahrnehmbarkeit der Wasserzeichenmarkierung von texturierten 3D-Modellen, nachdem sowohl die Textur als auch das zugehörige 3D-Modell unabhängig voneinander mit einem Wasserzeichen markiert wurden, wird zusätzlich eine Qualitätsmetrik eingeführt. Es werden für ausgewählte Szenarien Algorithmen etabliert, die den Anforderungen in diesen Szenarien genügen. Im Folgenden wird die Notwendigkeit der Entwicklung neuer Verfahren aufgezeigt und der geschichtliche Hintergrund dazu gegeben, bevor die Ziele der Arbeit folgen. Im Anschluss wird die Vorgehensweise beim wissenschaftlichen Arbeiten und die Gliederung der Arbeit erläutert.

1.1 Motivation

Digitale Wasserzeichen, robuste Hashverfahren und Fingerprintverfahren sind etablierte Technologien, die unter anderem zur Authentifikation bei Multimediadatenmaterial eingesetzt werden. Digitale Wasserzeichen werden neben steganographischen Verfahren zum Verstecken von Informationen bei Multimediainhalten im Kontext der Kryptologie eingesetzt. Während bei der Steganographie die Zielsetzung das Sichern der Vertraulichkeit ist, ist es bei digitalen Wasserzeichen der Schutz der Integrität und Authentizität.

Digitale Wasserzeichen werden von Rechteinhabern genutzt, um Informationen einzubetten und so Kunden bei einem Datenleck zu identifizieren oder die Urheberschaft nachzuweisen. Fingerprintverfahren und robuste Hashverfahren eignen sich ebenfalls zur Authentifizierung von Datenmaterial, wobei sie im Gegensatz zu digitalen Wasserzeichen vorhandene identifizierende Merkmale des Mediums oder Trägermaterials extrahieren, um es wiederzuerkennen. Robuste Hashverfahren finden ihren Einsatz beim schnellen Wiederfinden von Multimediadatenmaterial, selbst wenn typische Nachverarbeitungsschritte durchgeführt wurden. An Fingerprintverfahren wird typischerweise nicht der Robustheitsanspruch gestellt wie bei robusten Hashverfahren. Es wird erwartet, dass z.B. bei einer Kopie eines Ausdrucks, der Fingerabdruck nicht mehr identisch ist.

Während digitale Wasserzeichenverfahren für Audio, Bild, eBooks und Dokumente, die die Anforderungen an sie zum Authentizitätsnachweis erfüllen, in der Literatur vorhanden sind, gibt es Forschungsbedarf bei Videowasserzeichen und 3D-Modellen mit deren Texturen. Die Robustheit und Sicherheit der Verfahren gegenüber typischen Nachverarbeitungsschritten ist im Stand der Technik nicht ausreichend gegeben. 3D-Modelle werden in der digitalen Welt immer wichtiger. Mit 3D-Druckern und Scannern ist es möglich, Bauteile und Konstruktionen schnell und kostengünstig zu erstellen und zu kopieren. Mit dem Aufstreben von Plattformen zum Vertrieb von 3D-Modellen für Videospiele, Konstruktionen und Gimmicks, erleben Rechteinhaber die gleichen Herausforderungen beim Urheberschutz und der Kundenidentifizierung, die bei Multimediainhalten bereits seit langem bekannt sind. Durch das Streamen von Videos nach Hause wird das Abfilmen des Videos von einem Screen immer einfacher und stellt gleichzeitig eine der größten Herausforderungen bei Videowasserzeichenverfahren dar.

Digitale Wasserzeichen sollen das Multimediamaterial nicht wahrnehmbar verändern, sodass menschlicher Betrachter im Durchschnitt die Veränderungen nicht wahrnimmt. Zum automatisierten Bewerten dieser Eigenschaft führt die vorliegende Arbeit eine Qualitätsmetrik für texturierte 3D-Modelle ein, denn in die Textur als auch in das 3D-Modell werden die digitalen Wasserzeichen getrennt voneinander eingebettet, wobei der Betrachter aber das texturierte 3D-Modell angezeigt bekommt. Somit können die Veränderungen verstärkt werden oder gar Artefakte auftreten. Mittels der Metrik lässt sich die Qualitätsänderung

eines texturierten 3D-Modells nach einer durchgeführten Wasserzeichenmarkierung ausdrücken, ohne dabei Bewertungen durch Testpersonen durchführen lassen zu müssen. Der zeitliche und finanzielle Aufwand kann damit gering gehalten werden und es ist eine objektive Vergleichbarkeit gegeben. Auch das Finden von optimalen Einbettungsparametern zum Erreichen einer gewünschten Nicht-Wahrnehmbarkeit der Veränderungen lässt sich mit einer entsprechenden Metrik wesentlich schneller realisieren.

Digitale Wasserzeichen sind ein passiver Schutz, der das Kopieren von digitalen Dateien nicht verhindert, sondern entdeckt und das Verfolgen ermöglicht. Dazu ist es notwendig, dass das Datenmaterial online gefunden, heruntergeladen und das digitale Wasserzeichen detektiert wird. Das automatisierte Finden und Identifizieren des Datenmaterials kann mithilfe kryptografischer Hashfunktionen, robuster Hashfunktionen oder inhaltsbasierter Erkennung erfolgen. Robuste Hashverfahren eignen sich dazu am besten, da sie selbst nach gängigen Nachbearbeitungsschritten Datenmaterial wiederfinden. Aus diesem Grund wird eine robuste Hashfunktion für 3D-Modelle eingeführt.

Die vorliegende Arbeit stellt ein Fingerprintverfahren vor, das Merkmale aus blankem und bedrucktem Papier extrahiert, sodass sowohl Ausdrucke von digital identischen Dateien als auch Kopien von Ausdrucken jeweils voneinander unterschieden werden können. Ein solches Verfahren ist immer dann sinnvoll, wenn sichergestellt werden muss, dass es sich um ein Original und keine Kopie oder eine Fälschung handelt. Anwendungsbeispiele dazu sind Eintrittskarten, Tickets, Gutscheine, Arztrezepte und Produktverpackungen. Bei den genannten Anwendungsbeispielen wendet das frühe Erkennen von Fälschungen finanziellen Schaden ab. In einigen Beispielen bedeutet das, dass ein Nutzer in der Lage sein muss, das Original von einer Fälschung zu unterscheiden. So kann ein Nutzer, der z.B. ein Ticket aus zweiter Hand kauft, direkt prüfen, ob es gültig ist, bevor die Transaktion stattfindet. Naheliegend ist, diese Prüfung durch den Nutzer mit seinem Smartphone durchführen zu lassen. Aus Datenschutz- und Sicherheitsgründen eignet sich für die genannten Anwendungsszenarien das Extrahieren der Merkmale aus Ungenauigkeiten im Substrat, der Tintenformulierung und dem Druckprozess. Damit kann die Extraktion auf dem Endgerät des Nutzers stattfinden, ohne einen Datenaustausch mit einem Server.

1.2 Historie

Wasserzeichen wurden nach Cox et al. [32] erstmals um 300 in China genutzt. Im 13. Jahrhundert wurden sie zum ersten Mal in Europa eingesetzt und dienten der Identifizierung des Papierherstellers. Wasserzeichen sind der Allgemeinheit aus dem Alltag bei Geldscheinen bekannt und authentifizieren ihre ausgebende Institution. Die Idee, Videodaten mit einem Wasserzeichen zu versehen zum Realisieren von interaktiven Anwendungen

im TV-Bereich, wurde 1975 von Ralph H. Baer von der Lockheed Sanders Inc. patentiert [5]. Anfang 1990 gab es die ersten Applikationen von Nielson und VEIL Interactive Inc. im Bereich der digitalen Videowasserzeichen. Es wurden verschiedene Verfahren entwickelt, deren Einbettung entweder im Frequenzraum oder der räumlichen Domäne erfolgt. Zum Einbetten im Frequenzraum dienen hauptsächlich die drei bekannten Transformationen, nämlich diskrete Cosinus Transformation (DCT)[33], Fourier Transformation (FT)[73] und Wavelettransformation (WT)[108]. Eines der bekanntesten Verfahren zum Einbetten in der räumlichen Domäne wurde von Kalker et al. [50] eingeführt, das als Grundlage für viele Videowasserzeichen-Verfahren dient.

Ein Fingerabdruck, wie er in der vorliegenden Arbeit verstanden wird, ist eine binäre Repräsentation von einzigartigen Merkmalen des Multimediadatenmaterials. In der Literatur werden Fingerabdruckverfahren für Mediendaten je nach Robustheit auch als robuster [42, 74] oder semi-fragiler [66] Hash bezeichnet.

Das Verfahren, einen Fingerabdruck von Papier zu extrahieren, wurde erstmals im Jahr 2001 von Pappu [75] publiziert. Arbeiten, die darauf aufbauen, nutzen einen Laserstrahl, der auf das Substrat trifft und einzigartige Muster erzeugt [79]. Toreini et al. [96] nutzten bei ihrer Arbeit eine Lampe als Lichtquelle, mit der sie durch das Substrat leuchteten und nahmen das Ergebnis mit einer Kamera auf. Die Arbeiten zeigen die Möglichkeit der Unterscheidung von Ausdrucken, wobei das digitale Bild immer identisch bleibt.

Robuste Hashverfahren für 3D Objekte wurden erstmals 2010 von Lee et al. [60] publiziert. Spätere Arbeiten gehen in den Ansätzen auf die verschiedenen Repräsentationen und Dateitypen für 3D Modelle ein und entwickeln speziell dafür geeignete Verfahren [58, 19].

1.3 Ziele

Der Einsatz von digitalen Wasserzeichen zur Verfolgung von Urheberrechtsverletzungen ist das primär betrachtete Szenario sowohl bei wissenschaftlichen Arbeiten als auch bei kommerziellen Anbietern. Für diesen Einsatzzweck werden eine hohe Robustheit und Sicherheit bei gleichzeitiger Nicht-Wahrnehmbarkeit beansprucht. Ziel dieser Arbeit ist es, die Robustheit und Sicherheit der bekannten Algorithmen bei Video-, Textur- und 3D-Modell-Wasserzeichen zu verbessern, bei weiterhin hoher visueller Qualität. Zur objektiven Bewertung der visuellen Qualität eines texturierten 3D-Modells, bei dem die Textur und das 3D-Modell unabhängig voneinander markiert wurden, ist ein entsprechender Algorithmus erforderlich, da in der Literatur kein Verfahren dazu bekannt ist.

Zur Unterstützung der automatisierten Suche von Urheberrechtsverletzungen auf Tauschbörsen und Marktplätzen eigenen sich robuste Hashverfahren, um schnell und

selbst nach Nachverarbeitungsschritten das Multimediamaterial mit geringen Fehlerraten wiederzuerkennen. Damit ist ein weiteres Ziel dieser Arbeit, ein Hashverfahren für 3D-Modelle zu etablieren, dessen Sicherheit den Anforderungen für dieses Anwendungsgebiet erfüllt.

Das Authentifizieren von Produkten und damit von bedrucktem und unbedrucktem Papier ist nach Stand der Technik nicht ohne weiteres mit Smartdevices möglich. Auch wird das nutzbare Papier eingeschränkt. Ein weiteres Ziel der Arbeit wird das Extrahieren von Merkmalen als Fingerabdruck aus bedrucktem und unbedrucktem Druckerpapier nur mit Hilfe eines Smartphones, ohne weiteres Hilfsequipment oder Hardware. Damit wird einem Nutzer die Möglichkeit gegeben, mit einem Smartphone eine originale Faltschachtel oder ein Dokument von der jeweiligen Kopie zu unterscheiden. Das Verfahren soll eindeutige Merkmale aus einem vordefinierten Bereich extrahieren. Die Merkmale sollen sicher sein, also nicht nachgestellt werden können, und möglichst robust gegen typischen äußere Einflussfaktoren sein. Wird eine Kopie erzeugt oder das digitale Dokument nochmals ausgedruckt, so unterscheiden sich die extrahierten Merkmale und damit die Fingerabdrücke voneinander.

1.4 Methodik und Gliederung

Das Forschungsdesign für diese Arbeit basiert auf der von Vaishnavi & Kuechler [100] eingeführten Design Science Methode. Das Ziel von Design Science Research (DSR) ist es, im Prozess der Entwicklung zu lernen. Die Arbeit führt die Problemstellungen in Kapitel 1 ein. Kapitel 2 legt die Grundlagen zu der Thematik dar. Die Anforderungen an die Lösungen folgen in Kapitel 3. Im Anschluss wird der Stand des Wissens aufgezeigt, gefolgt von den verwandten Arbeiten, bevor die eigenen Lösungsvorschläge in den Kapiteln 4 - 9 präsentiert und evaluiert werden. In Kapitel 4 wird die Entwicklung eines Fingerprintverfahrens eingeführt, gefolgt von einem 3D-Modell Wasserzeichen in Kapitel 5 und einem Texturwasserzeichen in Kapitel 6. Zur Bewertung der Nicht-Wahrnehmbarkeit wasserzeichenmarkierter texturierter 3D-Modelle, wird in Kapitel 7 eine entsprechende Metrik vorgeschlagen. In Kapitel 8 folgt ein robustes Hashverfahren für 3D-Modelle, das sich zum schnellen und robusten Auffinden von bekannten 3D-Modellen eignet. Kapitel 9 zeigt ein Videowasserzeichen, das vor allem robust gegenüber Nachverarbeitungsschritten und sicher gegen Angriffe ist. Jedes der Kapitel mit den Lösungsvorschlägen enthält eine eigene Diskussion der Ergebnisse, eine Zusammenfassung und einen Ausblick, wobei die gesamte Arbeit in Kapitel 10 zusammengefasst wird.

2 Grundlagen

Multimediadaten sind unterschiedlichen Risiken ausgesetzt. Neben der Authentizität, die ausschließlich in der vorliegenden Arbeit betrachtet wird, gibt es nach [36] weitere Schutzziele, nämlich Integrität, Verfügbarkeit, Vertraulichkeit, Nicht-Abstreitbarkeit und Revisionsfähigkeit. Neben Kryptografischen Verfahren bietet die Kryptologie auch Data Hiding Verfahren, wie die Steganographie und digitale Wasserzeichen, zum Durchsetzten der Schutzziele an. Während mit steganographischen Verfahren die Vertraulichkeit adressiert wird, kann mit digitalen Wasserzeichen die Integrität und Authentizität von Multimediadaten geprüft werden. Das Erfüllen der Schutzziele ist bei digitalen Wasserzeichen lediglich prüfbar, da sie passive Verfahren sind. Im Gegensatz dazu können mit aktiven Verfahren die Schutzziele durchgesetzt und das Verletzen der Schutzziele verhindert werden. Ein Beispiel für aktive Verfahren ist das Digital Rights Management (DRM), das auf Basis des Betriebssystems Rechte durchsetzt. Aktive Verfahren werden in dieser Arbeit nicht berücksichtigt, da sie am sichersten und robustesten auf Betriebssystemebene von den Herstellern integriert werden können und gut erforscht sind. In diesem Kapitel wird zuerst die Definition der Authentizität gegeben und danach werden die Eigenschaften erläutert, die die zu entwickelten Verfahren aufweisen müssen zum Erreichen der Ziele. Im Anschluss werden die Grundkonzepte der Verfahren und die Grundlagen der aus der Literatur genutzten Algorithmen oder Techniken eingeführt.

2.1 Authentizität

Die Authentizität beschreibt die Echtheit eines Objektes, wobei es beliebige materielle und immaterielle Objekte sein können. Zum Authentizitätsnachweis von Objekten wird die Authentifikation durchgeführt und damit geprüft, ob der Nachweis der Identität des Objekts erbracht werden kann.

Die vorliegende Arbeit unterscheidet zwischen der Authentizität von Trägermaterial, im speziellen Papier, zum Nachweis der Originalität und der Authentizität einer Person, im speziellen des Urhebers oder Kunden, zum Nachweis der Identität.

Zum Realisieren der Kundenauthentizität erhält jeder Kunde eine individualisierte Kopie des Mediums. So lässt sich durch die Kundenauthentifikation bei Verletzung der Urheberrechte durch einen Kunden selbiger mit einem gekauften Medium in Verbindung bringen. Bei der Urheberauthentizität wird hingegen jede Kopie des Mediums mit der identischen Kennzeichnung versehen, sodass der Rechteinhaber des Mediums nachgewiesen werden kann.

2.2 Eigenschaften von Verfahren zum Authentizitätsnachweis

Die im Umfang der Arbeit entwickelten Verfahren fokussieren den Authentizitätsnachweis. Die wichtigsten Eigenschaften nach [35] sind Robustheit, Sicherheit, Nicht-Wahrnehmbarkeit, geheime oder öffentliche Verifikation, Invertierbarkeit und Komplexität. Die beiden letztgenannten stehen jedoch nicht im Vordergrund für das Anwendungsszenario des Authentizitätsnachweises für robuste Hashverfahren, digitale Wasserzeichen und Fingerprints.

Robustheit Robust sind Verfahren, die invariant gegen typische Nachverarbeitungsschritte sind. Die Information kann also selbst nach Modifikationen, die nicht mit Kenntnis des Verfahrens durchgeführt wurden, ausgelesen werden. Unter typische Nachverarbeitungsschritte fallen z.B. Kompression, Format-Konvertierung und Skalierung, die meist angewandt werden, um den Speicherbedarf des Mediums zu reduzieren.

Nicht-Wahrnehmbarkeit Dieser Punkt beschreibt, ob vorgenommene Modifikationen des Datenmaterials für einen Menschen optisch oder akustisch nicht wahrnehmbar sind. Fügt ein Verfahren nicht wahrnehmbare Veränderungen hinzu, so spricht man auch von einer transparenten Modifikation.

Sicherheit Ein Verfahren gilt als sicher, wenn die Informationen nicht aufgespürt, verändert, gelöscht oder kopiert werden können. Als Voraussetzung wird angenommen, dass ein Angreifer das zugrundeliegende Verfahren kennt, aber weder Kenntnis über den kryptografischen Schlüssel hat noch ein kryptografisches Verfahren brechen kann. Damit beinhaltet die Sicherheit im Gegensatz zur Robustheit alle gezielten Angriffe unter Ausnutzung des Wissens über das Verfahren.

Kapazität Die Kapazität gibt an, wie viele Informationen in das Datenmaterial mittels des Verfahrens integriert werden können. Die Angaben zur Kapazität sind abhängig vom

Medienformat und werden in Bit pro Bezugsgröße gemessen. Die Bezugsgröße wird bei Bildern z.B. in Bit pro Pixelgröße, bei Videomaterial in Bit pro Sekunde und bei 3D-Modellen in Bit pro Knotenanzahl angegeben.

Geheime oder Öffentliche Verifikation Die Algorithmen zur Authentifikation werden anhand der Verifikation unterschieden, je nachdem ob es sich um eine geheime oder öffentliche Verifikation handelt. Die digitalen Wasserzeichen sind symmetrische Verfahren und sind der geheimen Verifikation zugeordnet, da sie nur durch Personen, die Kenntnis über den symmetrischen Schlüssel haben, verifiziert werden können. Das robuste Hashverfahren und das Fingerabdruckverfahren sind hingegen öffentlich verifizierbar. Ein Nutzer muss den robusten Hash oder Fingerabdruck selbst extrahieren können. Dazu muss er in der vollen Kenntnis der Verfahren und genutzter Parameter sein.

2.3 Grundkonzepte

Im Folgenden werden die Grundkonzepte von digitalen Wasserzeichen, robusten Hash- und Fingerprintverfahren eingeführt und am Ende der jeweiligen Unterkapitel deren Eigenschaften genannt, die zum Erreichen des Ziels, dem Nachweis der Authentizität, benötigt werden.

2.3.1 Digitale Wasserzeichen

Unter Wasserzeichen wird generell das Verstecken von zusätzlichen Informationen in einem Trägermedium verstanden. Damit soll im allgemeinen die Integrität oder Authentizität nachweisbar gemacht werden. Sehr ähnlich zu digitalen Wasserzeichen ist die Steganographie, deren Zielsetzung jedoch das Sichern der Vertraulichkeit ist. Bei der Steganographie wird eine Information in das Medium eingebettet, um darüber versteckt zu kommunizieren. Bei Wasserzeichen dagegen soll eine geheime Verifikation möglich sein, die verlässlich und damit robust und sicher ist.

Ein digitales wasserzeichenmarkiertes Medium durchläuft die folgenden Schritte [35]:

- Eine Wasserzeichennachricht wird mittels eines Einbettungsalgorithmus E in ein Trägermedium C nicht wahrnehmbar, mit Hilfe eines psychovisuellen oder psychoakustischen Modells [77, 45] eingebettet. Unter Verwendung eines geheimen Schlüssels K bei der Einbettung W_J wird das Wasserzeichen gegen unerwünschten Zugriff oder gegen Veränderungen abgesichert. Man erhält eine wasserzeichenmarkierte Kopie $C_W = E(C, W_J, K)$ als Ausgabe.

- Im Verteilungsprozess kann ein Nutzer eine Kopie des Mediums erstellen und weiterverteilen. Diese Kopie kann potentiell ein manipuliertes Wasserzeichen W_I enthalten, wenn z.B. der Nutzer das Medium manipuliert oder angegriffen hat.
- Sobald eine Kopie im Netz gefunden wird, kann die Wasserzeichennachricht W_I aus dem Medium ausgelesen werden unter der Zuhilfenahme des Detektors R und des geheimen Schlüssels K. Die detektierte Wasserzeichennachricht $W_I = R(C_W, K)$ wird mit allen eingebetteten Wasserzeichen verglichen, um festzustellen von wem die Kopie stammt.

Die betrachteten Wasserzeichenverfahren haben die Eigenschaften der Robustheit, Nicht-Wahrnehmbarkeit, Kapazität, Sicherheit und geheimen Verifikation.

2.3.2 Robuste Hashverfahren

Ein robuster Hash repräsentiert durch eine binäre Zeichenkette das Datenmaterial. Ein Hash ändert sich selbst bei typischen Nachverarbeitungsschritten und Angriffen nicht und daher gilt er als robust. Ausgehend vom robusten Hash kann nicht auf das Datenmaterial geschlossen werden. Als Alternative zu den robusten Hashverfahren existiert die inhaltsbasierte Erkennung. Diese funktioniert auch auf verändertem Datenmaterial, wobei robuste Hashfunktionen eine geringere Komplexität aufweisen und sich damit zum schnelleren Auffinden von ähnlichem Datenmaterial in großen Datenbeständen eignen. Robuste Hashverfahren weisen jedoch im Gegensatz zu kryptografischen Hashfunktionen keine starke Kollisionsresistenz auf. Kryptografische Hashfunktionen werden auf der Bytesequenz berechnet und verfügen so über eine starke Kollisionsresistenz und sind wesentlich performanter in ihrer Berechnung, verglichen mit robusten Hashverfahren. In der Literatur werden robuste Hashverfahren auch als Fingerabdruckverfahren oder wahrnehmungsbasierte Hash bezeichnet.

Robuste Hashverfahren durchlaufen folgende Schritte [25]:

- Das Datenmaterial wird in Subgruppen T unterteilt und für sie eine feste Reihenfoge definiert anhand des Schlüssels K. Eine robuste globale Bezugsgröße B wird auf Basis des Datenmaterials errechnet. Ein Merkmal M von T wird in Bezug zu B gesetzt und anhand eines Schwellwerts der binären “0“ oder “1“ zugeordnet. Aus den einzelnen Bits setzt sich der robuste Hash $H = E(M_T, B, K)$ unter Berücksichtigung der vordefinierten Reihenfolge zusammen.
- Ein Nutzer eines Mediums kann eine Kopie erstellen. Diese Kopie kann potentiell einen modifizierten Hash H_I enthalten, wenn z.B. der Nutzer das Medium manipuliert oder angegriffen hat.

- Soll ein Medium automatisiert auf bekanntes Material analysiert werden, kann der robuste Hash $H_I = E(M_T, B, K)$ aus dem Medium ausgelesen werden. Der robuste Hash H_I wird mit den Hashes in der Datenbank verglichen. Ist die Distanz zu einem Hash in der Datenbank kleiner als ein Schwellwert $Th > HD(H, H_I)$, wird das Medium als bekannt angesehen.

Der in der Arbeit vorgestellte robuste Hash erfüllt die die Eigenschaft der Robustheit, Sicherheit und öffentlichen Verifikation.

2.3.3 Fingerprintverfahren

Ein Fingerprint, oder auch Fingerabdruck genannt, repräsentiert identifizierende Merkmale eines physikalischen Trägermediums. Damit der Fingerprint eindeutig für jedes physikalische Trägermedium ist, müssen ausreichend viele Merkmale vorhanden sein. Sie sollen robust gegenüber fest definierten äußeren Einflüssen sein, jedoch fragil gegenüber Angriffen. Dies unterscheidet ein Fingerprint von robusten Hash. Im Gegensatz zu digitalen Wasserzeichen werden die Merkmale nicht erzeugt, sondern vorhandene werden extrahiert. Die Stabilität der Merkmale bei noch unbekannten physikalischen Trägermedien ist schwer abzuschätzen.

Ein Fingerprint, der aus einem Medium zum Authentizitätsnachweis extrahiert wird, durchläuft typischerweise folgende Schritte [79]:

- Ein physikalisches Medium S wird analysiert. Merkmale M werden anhand eines Auslesealgorithmus E extrahiert und anhand einer Bezugsgröße B als Fingerprint durch “0“ oder “1“ repräsentiert $F(S) = E(S, B)$.

- Das physikalische Medium wird verteilt. Soll die Authentizität des verteilten physikalischen Mediums verifiziert werden, wird der Fingerprint mit dem Auslesealgorithmus aus vorhergehendem Schritt extrahiert $F(S') = E(S, B)$. Der Fingerprint $F(S')$ wird mit allen Fingerprints in der Datenbank verglichen. Befindet sich ein Fingerprint $F(S)$ in der Datenbank, dessen Distanz zu $F(S')$ geringer ist als ein gewählter Schwellwert $Th > HD(F(S), F(S'))$, so ist das physikalische Medium S als das Original authentifiziert, sonst nicht.

2.4 Wavelet-Transformation

Die Wavelet-Transformation eignet sich, ähnlich der Fourier-Transformation, zur Analyse der Frequenzkomponenten eines Signals. Im Unterschied zur Fourier-Transformation

enthält die Wavelet-Transformation Informationen über die Raum- oder Zeitdomäne. In dieser Arbeit werden Graustufenbilder des Helligkeitskanals Y aus dem Farbraum CIE XYZ mit einer Größe von $600x600$ Pixel mittels der Familie der Daubechier-Wavelet-Transformation [48] zerlegt.

2.5 3D-Modelle

Ein 3D-Modell ist eine mathematische Repräsentation eines dreidimensionalen Objekts. Eine Repräsentation eines solchen Objekts wird in vielen verschiedenen Bereichen genutzt, wie z.B. bei Computer-Aided Design (CAD), wissenschaftlicher und medizinischer Bildgebung, Objekterkennung und Computergrafik [29].

Ein Drahtgittermodell, auch Polygonnetz genannt, ist ein Objekt bestehend aus vielen Dreiecksflächen, die an ihren Kanten zusammengefasst sind. Ein Drahtgittermodell $\mathbf{M} = (\mathbf{V}, \mathbf{E}, \rho)$ kann durch ein Set $\mathbf{V} = v_1, ..., v_n$ von n Knoten, ein Set von Kanten $\mathbf{E} \in \mathbf{V}^2$, die die Knoten verbinden und eine Funktion $\rho : \mathbf{V} \rightarrow \mathbf{R}^3$, die jedem Knoten v_i eine feste Position $\vec{v}_i = \rho(v_i)$ im kartesischen Raum vergibt, repräsentiert werden. Das Set von Kanten trägt die Information der Topologie des 3D-Modells. Damit sind zwei Knoten v_i und v_j, wobei $i, j \in \{1, ..., n\}$ und $i \neq j$, nur dann miteinander verbunden oder benachbart sind, wenn gilt $(v_i, v_j) \in \mathbf{E}$. Die Nachbarschaftsmatrix A mit den Einträgen $\{0, 1\}^{\mathbf{V} \times \mathbf{V}}$ hat überall den Eintrag 0, wenn nicht $(v_i, v_j) \in \mathbf{E}$ erfüllt ist. Die Anzahl der benachbarten Knoten eines spezifischen Knotens v_i wird *Grad* genannt und mit d_i bezeichnet. Die zugehörige Diagonalmatrix $D^{\mathbf{V} \times \mathbf{V}}$ ist vom Grad $D(i, i) = d_i$.

Die Drahtgittermodelle bestehen aus Dreiecken, da Dreiecke die einfachsten Polygone sind und am effizientesten gerendert werden können. Außerdem lassen sich mit Dreiecken beliebige Oberflächen nachbilden. Dementsprechend sind Algorithmen, die das Rendern übernehmen, auf Dreiecke optimiert. OpenGl ist z.B. eine API, die zum Rendern von 3D-Modellen genutzt wird. Sie unterstützt zwar Rechtecke, aber bricht sie intern auf Dreiecke herunter.

2.6 Bezugspunkt in 3D-Modellen

Zum Durchführen von reproduzierbaren Operationen auf einem 3D-Modell, wird ein Bezugspunkt benötigt. Typischerweise exisitert bei einem 3D-Modell ein Bezugspunkt, der jedoch nicht robust ist, sobald leichte Manipulationen am 3D-Modell vorgenommen werden.

Es werden zum einen die kleinste umschließende Kugel und das Gravitationszentrum eingeführt.

2.6.1 Graviationszentrum

Die Koordinaten des Gravitationszentrums $C_g = (x_g, y_g, z_g)$, auch Massezentrum genannt, werden mittels Durchschnittsbildung der einzelnen Koordinaten x_i, y_i und z_i aller Knoten $\{v_1, ..., v_{n'}\}$, mit $i = 1, ..., n'$ des Kernmodells M' bestimmt:

$$C_g = (x_g, y_g, z_g) = \left(\frac{1}{n'} \sum_{i=1}^{n'} x_i \ , \ \frac{1}{n'} \sum_{i=1}^{n'} y_i \ , \ \frac{1}{n'} \sum_{i=1}^{n'} z_i \right)$$

2.6.2 Kleinster umschließender Kreis

Das Finden des smallest enclosing ball (SEB) von einer Punktwolke ist ein klassisches Problem der Geometrie [37]. Sei $P \subset \mathbb{R}^d$ ein Set von Punkten in einem d-dimensionalen Raum, $\vec{c}_{min} \in \mathbb{R}^d$ das Zentrum des SEB und $r_{min} \in \mathbb{R}^{\geq 0}$ der Radius des SEB, dann lässt sich der SEB $\langle \vec{c}_{min}, r_{min} \rangle$ wie folgt berechnen: $\forall \vec{p}_i \in P : \|\vec{c} - \vec{p}_i\|_2 \leq r$. Der SEB ist definiert als der kleinstmöglichste Ball, der den Radius r_{min} minimiert und deshalb existiert kein anderer gültiger Miniball $\langle \vec{c}, \tilde{r}_{min} \rangle$ mit einem kleineren Radius $\tilde{r} < r_{min}$. Kälber führt in [49] den Beweis dazu.

Zur Bestimmung des SEB wird ein Set von Hilfspunkten $S \subseteq P$ zur Unterstützung herangezogen, sodass der SEB vom originalen Modell P und den Hilfspunkten S identisch ist. Dabei liegen die Punkte $\vec{s} \in S$ auf der Oberfläche des SEB $\forall \vec{s} \in S : \|\vec{c}_{min} - \vec{s}\|_2 \leq r_{min}$. Die Anzahl der Punkte, die zur Unterstützung dienen, ist im d-dimensionalen Raum maximal $d + 1$.

2.7 Rauheit eines 3D-Modells

Die Rauheit eines 3D-Modells lässt sich über die Normalen angrenzender Dreiecke berechnen [107]. Abbildung 2.1 zeigt ein Beispiel, bei dem die Rauheit des Dreiecks T mit seinen Eckpunkten v_1, v_2, v_3 bewertet werden soll. Zuerst bewertet der Algorithmus die Eckpunkte, indem er die Normalen der Dreiecke, die einen gemeinsamen Eckpunkt haben, verrechnet. Daraus errechnet der Algorithmus das arithmetische Mittel und die Varianz. Seien v_l die Eckpunkte des zu bewertenden Dreiecks, $\vec{N}^{(k)}$ der Normalenvektor des k-ten Dreiecks und $\vec{N}^{(k+1)}$ der Normalenvektor des k+1-ten Dreiecks, die beide v_l als gemeinsamen Eckpunkt haben. Außerdem sei M_{v_l} die Anzahl der Dreiecke, die den Eckpunkt v_l gemeinsam haben, $\mu(v_l)$ das arithmetische Mittel und $\sigma(v_l)$ die Varianz des

l-ten Eckpunkts des zu bewertenden Dreiecks, dann lässt sich die lokale Bewertung $R^{(k)}$ des k-ten Dreiecks wie folgt errechnen:

$$
\begin{aligned}
p_d &= 1 - \vec{N}^{(k)} \cdot \vec{N}^{(k+1)}, \\
\mu^{(k)}(v_l) &= \frac{1}{M_{v_l}} \sum_{d=1}^{M_{v_l}} p_d, \\
\sigma^{(k)}(v_l) &= \frac{1}{M_{v_l}} \sum_{d=1}^{M_{v_l}} (p_d - \mu^{(k)}(v_l))^2 \\
R^{(k)} &= \frac{\sum_{i=1}^{3} \mu^{(k)}(v_i) \cdot \sigma^{(k)}(v_i)}{\sum_{i=1}^{3} \sigma^{(k)}(v_i)}
\end{aligned}
\tag{2.1}
$$

Es macht im Allgemeinen keinen Unterschied, ob die Rauheit über die Kanten oder Knoten der Dreiecke bestimmt wird. Sowohl der Rechenaufwand als auch das Ergebnis beider Herangehensweisen ist identisch.

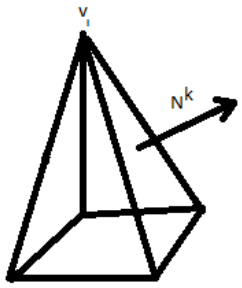


Abbildung 2.1: Beispiel eines zu bewertenden Knotens. Dabei ist v_l der zu bewertende Knoten, $M_{v_l} = 4$ Dreiecke, die den Knoten gemeinsam haben und $N^{(k)}$ sind die Normalenvektoren der Dreiecke

2.8 Sonstiges

2.8.1 Konfusionsmatrix

Zur Bewertung der Ergebnisse der vorgestellten Klassifikationsverfahren wird die Konfusionsmatrix herangezogen. Bei korrekten Klassifikationen unterscheidet die Konfusionsma-

trix zwischen der Akzeptanzrate (AR) und Rückweisungsrate (RR), bei den Fehlern in der Klassifikation wird zwischen Falschakzeptanz- (FAR) und Falschrückweisungsrate (FRR) unterschieden.

Die Falschakzeptanz beschreibt die Wahrscheinlichkeit, dass ein unbekanntes Objekt fälschlicherweise als bekannt identifiziert wird. Mit dieser Rate wird auch die Kollisionswahrscheinlichkeit angegeben.

Die Falschrückweisung beschreibt die Wahrscheinlichkeit, dass ein bereits bekanntes Objekt als unbekannt identifiziert wird. Die Grenzen der Fehler sind die FAR und FRR. Beide Fehler variieren abhängig von der Parametrisierung und dem Anwendungsszenario.

2.8.2 Drucktechniken

Drucken ist das Aufbringen eines Druckbildes auf ein Substrat. Als Substrat dient jede Art von Papier, Verpackung, Textil und Kunststoff. Ziel eines Druckes ist die Informationsweitergabe. Es wird eine bildbeschreibende digitale Datei an einen Drucker gesendet, die er auf ein Substrat abbildet. Die aktuellen Drucktechniken lassen sich in zwei Kategorien einteilen, nämlich Anschlagdrucker und berührungslose Drucker. Berührungslose Druckverfahren, wie z.B. Tintenstrahl- oder Laserdrucker, benötigen keine Druckform, wobei Anschlagdrucker, wie z.B. Offsetdrucker, eine Druckform benötigen. In diesem Kapitel werden Druckerzeugnisse von Offset-, Tintenstrahl- und Laserdruckern verwendet.

Beim Offsetdruck werden die Druckplatten zum Auftragen der Farbe auf das Substrat verwendet, wobei jede eine Farbe aufträgt. Die Druckplatte besteht aus zwei Bereichen, dem bedruckten Bereich, der die Farbe aufnimmt und dem nicht-bedruckten Bereich, der keine Farbe aufnimmt.

Beim Tintenstrahldruck wird Tinte über Düsen auf das Substrat gesprüht, die anschließend schnell trocknet. Das zu druckende Bild wird gerastert und beim Druck in jede Punktposition werden kleine Tröpfchen gesprüht. Mischfarben werden durch die Kombination der vier Druckfarben und deren Menge gesteuert.

Laserdrucker nutzen elektrisch geladene Drucktrommeln. Über einen Laser wird das Negativ eines Druckbildes auf die Trommel projiziert. Der Toner ist entgegengesetzt zur Trommel geladen und an den Stellen, die vom Laser nicht ausgeleuchtet wurden, wird der Toner angezogen. Hinter dem Substrat wird eine stärkere Ladung erzeugt, sodass der Toner auf das Substrat gezogen wird und durch Erwärmung dauerhaft auf dem Substart verbleibt. Über Bürsten werden Farbreste entfernt und die Trommel gereinigt.

Im Allgemeinen sind Druckplatten teurer und sinnvoller bei großen Auflagen zu nutzen, wobei bei einer kleinen Stückzahl Tintenstrahl oder Laserdrucker zu bevorzugen sind.

2.8.3 Farbraum

Während Drucker den CMYK-Farbraum nutzen, nutzen Bildschirme und digitale Kamerasensoren den RGB-Farbraum. Zur qualitativen Bewertung der gedruckten Farbe wird die menschliche Wahrnehmung herangezogen. Diese ist in dem CIEXYZ-Farbraum modelliert. Er beschreibt Farbe als eine Mischung von Reizwerten für jeden der drei Zapfen im menschlichen Auge. Die Zapfen reagieren unterschiedlich auf kurz-, mittel- und langwelliges Licht. In dem Farbraum bildet X langwelliges Licht (rot), Y mittelwelliges Licht (grün) und Z kurzwelliges Licht (blau) ab. Da das menschliche Auge im mittleren Wellenbereich besonders empfindlich ist, wird Y auch als Helligkeitswert verwendet.

Digitale Bildsensoren in Kameras erfassen die Intensität des Lichtes von bestimmten Wellenlängen an einer Position. In Smartphones sind typischerweise CCD oder CMOS Sensoren verbaut, deren Funktionsweise zueinander ähnlich ist. Beim Auftreffen von Photonen auf dem kristallinen Silizium der Sensoren wird eine Spannung erzeugt, die gemessen und in ein digitales Signal umgewandelt wird. Jedem Sensor ist ein Filter vorgeschaltet, sodass nur ein bestimmtes Wellenlängenband durchgelassen wird. Vier solcher Sensoren, nämlich ein blauer, ein roter und zwei grüne Filter, werden gemeinsmam im Bayer-Pattern zusammengefasst. So wird ähnlich wie im CIE XYZ-Farbraum die Empfindlichkeit des menschlichen Auges nachgebildet. Um aus den Sensor-Rohdaten, die für jedes Pixel einen Wert enthalten, ein Bild zu erzeugen, werden aus den Umgebungsfarben für jedes Pixel die anderen beiden Farbwerte interpoliert.

Zum Transformieren der Farbkanäle R, G und B in den CIEXYZ Farbraum werden folgende Berechnungen durchgeführt:

$$\begin{aligned} X(x,y) &= 0,4125 \cdot R(x,y) + 0,3576 \cdot G(x,y) + 0,1804 \cdot B(x,y), \\ Y(x,y) &= 0,2126 \cdot R(x,y) + 0,7152 \cdot G(x,y) + 0,0722 \cdot B(x,y), \\ Z(x,y) &= 0,0193 \cdot R(x,y) + 0,1192 \cdot G(x,y) + 0,9503 \cdot B(x,y) \end{aligned} \tag{2.2}$$

3 Anforderungen

In diesem Kapitel werden die Anforderungen an die zu entwickelnden Verfahren gestellt und mit den Eigenschaften von Verfahren zum Authentizitätsnachweis aus Abschnitt 2.2 zusammengeführt.

3.1 Fingerprint für Produkt- und Dokumentenauthentifizierung

Der Fingerprint soll durch jeden Nutzer, unter Zuhilfenahme des Smartphones, extrahiert werden können und er soll vergleichbar gut bei verschiedenen gängigen Papiersorten unter verschiedenen Umgebungseinflüssen funktionieren. Daraus leiten sich die folgenden Anforderungen an den Fingerprint ab, gegen die er invariant sein muss:

Smartphonekamera Der Markt der Smartphones ist sehr groß, mit einigen Herstellern und vielen Modellen. Ebenso ist die Liste der Zulieferer für Kameras, Sensoren, Optik, Bildverarbeitungssoftware und Betriebssysteme für die Hersteller und deren Modelle heterogen. Dieses heterogene Umfeld bei den Smartphonekameras resultiert, aufgrund von Parametern, wie z.B. Auflösung, Gamma Korrektur, Sensor Sensitivität, Fokus und Belichtung, in unterschiedliche Bilder. Der Fingerabdruck muss robust gegen diese Einflüsse sein, die ihre Ausprägung in Rauschen, Unschärfe und der Auflösung haben. Hier ist vor allem zu berücksichtigen, dass nicht der kameraspezifische Fingerabdruck mit extrahiert wird. Diese Anforderung berücksichtigt damit die Eigenschaft der Robustheit.

Substrat und Druckverfahren In Kapitel 2.8.2 sind die drei relevanten Druckverfahren eingeführt worden. Der Fingerabdruck muss ungeachtet der Druckverfahren, Drucker, Veredelung und des Substrates robust sein. Unter das Substrat fallen die verschiedenen Papiertypen, -eigenschaften und -stärken. Bei der Veredelung werden die Druckerzeugnisse mit einer Beschichtung versehen, die z.B. glänzt oder matt ist. Diese Anforderung berücksichtigt damit die Eigenschaft der Robustheit.

Umgebungseinfluss Macht ein Endnutzer eine Aufnahme mit seinem Smartphone, so ist das Umfeld mit seinen Eigenschaften nicht beeinflussbar. Das Umgebungslicht mit der Temperatur und Beleuchtungsstärke variiert von etwa 2000 bis über 6500 Kelvin und 50 bis 130.000lx. Diese Anforderung berücksichtigt damit die Eigenschaft der Robustheit.

Positionierung Wenn der Benutzer ein Bild einer vordefinierten Region aufnimmt, kann die Kamera gedreht oder nicht perfekt im Bild zentriert sein. Die Kamera könnte sich auch in unterschiedlichen Höhen über dem Bild befinden, was je nach Auflösung der Kamera auch zu unterschiedlichen Abtastraten führen kann. Diese Anforderung berücksichtigt damit die Eigenschaft der Robustheit.

Kollisionssicherheit Es soll ausgeschlossen werden, dass für unterschiedliche Regionen zweimal der gleiche Fingerprint extrahiert wird. Dazu muss der Fingerprint eine ausreichende Größe haben. Diese Anforderung berücksichtigt damit die Eigenschaften der Kapazität und Sicherheit.

Angriffssicherheit Ein Angreifer soll nicht nur bei einem trivialen Angriff wie dem Kopieren der Region scheitern. Selbst bei der Kenntnis des Fingerprintverfahrens soll er nicht in der Lage sein, einen existierenden Fingerprint zu erzeugen. Diese Anforderung berücksichtigt damit die Eigenschaften der öffentlichen Verifikation und Sicherheit.

3.2 3D-Modell-Wasserzeichen

Der Vetrieb und Austausch von 3D-Modelle spielt mit immer präziser werdenden 3D-Drucker und -Scannern, als auch 3D-Drucker für verschiedenste Materialien, eine immer wichtigere Rolle. An das digitale Wasserzeichen für 3D-Modelle, das zum erbringen des Authentizitätsnachweis genutzt werden kann, werden die folgenden Anforderungen gestellt:

Nicht-Wahrnehmbarkeit Die Modifikationen eines 3D-Modells durch einen Einbettungsalgorithmus sollen möglichst für das durchschnittliche menschliche Auge nicht wahrnehmbar sein. Die Nicht-Wahrnehmbarkeit evaluiert jedoch nicht das Einhalten von Maßen bei maßgenauen Bauteilen.

Robustheit bei Nachverarbeitungsschritten Ein 3D-Modell kann zur Reduktion des Speicherbedarfs vereinfacht werden. Veränderungen der Ruhestellung durch Rotationen um die Hauptachsen, ebenso wie die Lage des 3D-Modells bezogen auf sein Bezugspunkt, können nach belieben durchgeführt werden. Durch Skalierung kann ein Betrachter das Modell in der Größe entsprechend anpassen. Diese Nachverarbeitungsschritte sollen für ein Wasserzeichen für 3D-Modelle keine Herausforderung darstellen.

Angriffe Ist bekannt, dass ein Wasserzeichen als Schutzmechanismus für 3D-Modelle genutzt wurde, ist davon auszugehen, dass Angreifer gezielt versuchen werden, das Wasserzeichen unleserlich zu machen oder zu löschen. Als Angriff kann das hinzufügen von zufälligem Rauschen oder eine geometrische Transformation gesehen werden. Dieser Aspekt wird in der Eigenschaft der Sicherheit adressiert.

Performanz Das Verändern eines 3D-Modells unter Berücksichtigung der vorher genannten Anforderungen, mündet in ein Optimierungsproblem. Das Lösen davon ist typischerweise zeitintensiv. Für Anwendungen zum Urhebernachweis sind lange Einbettungszeiten kein Problem, jedoch muss für Transaktionswasserzeichen eine Einbettung in wenigen Sekunden durchgeführt worden sein.

3.3 Texturwasserzeichen

Die Anforderungen für ein Wasserzeichenverfahren zum Markieren von Texturen entsprechen denen von Bildern nur begrenzt, da es einige Besonderheiten bei Texturen zu berücksichtigen gilt.

Angriffe: Angriffe zielen darauf ab, das Wasserzeichen zu löschen, zu schwächen oder zu ändern. Bei Texturen fallen darunter auch typsiche Bildnachverarbeitungsschritte, wie z.B. die Kompression. Denn eine Textur liegt bereits komprimiert in verschiedenen angepassten Größen skaliert zum 3D-Modell vor. Bei Texturen existieren für das Texturieren nicht genutzte Bereiche, die aber auf der 2D-Textur zu sehen sind. Diese könnte ein Angreifer einfach manipulieren, ohne dass es eine visuelle Auswirkung auf das texturierte 3D-Modell hat. Außerdem liegen die Texturen in unterschiedlichen Auflösungen in einer Textur vor und können durch einen Angreifer beliebig hoch und herunter skaliert werden. Beim Entwickeln des Wasserzeichenalgorithmus ist zusätzlich auf die beiden Punkte, die zur Eigenschaft der Sicherheit zu zählen sind, zu achten.

Nicht-Wahrnehmbarkeit: Veränderungen an einer Textur wirken sich sowohl an dem 2D als auch texturierten 3D-Modell aus. Für die visuelle Qualität für einen Betrachter ist aber letzteres in den meisten Fällen relevant und muss vom Wasserzeichenalgorithmus berücksichtigt werden.

3.4 Metrik zur Bewertung der Nicht-Wahrnehmbarkeit der Wasserzeichen texturierter 3D-Modelle

Die Metrik soll texturierte statische und animierte 3D-Modelle unter Berücksichtigung psychovisueller Eigenschaften von Menschen in der Wahrnehmbarkeit der Wasserzeichenmarkierung bewerten. Die folgenden Anforderungen adressieren die Eigenschaft der Nicht-Wahrnehmbarkeit (siehe Kapitel 2.2).

Geometrische Eigenschaften Durch das Mapping entstehen Helligkeitsunterschiede, Verzerrungen, Skalierungen und Strukturinformationen, die bestmöglich in der Bewertung der Metrik mit berücksichtigt werden sollten.

Vor- und Hintergrund Texturen beinhalten Bereiche, die nach dem Mapping sichtbar bzw. unsichtbar sind. Die Metrik soll bei ihrer Bewertung nur solche Bereiche der Textur bewerten, die sichtbar sind, und die nicht genutzten Bereiche im Hintergrund unberücksichtigt lassen.

Menschliche Wahrnehmung Ein menschliches Auge nimmt verschiedene Änderungen unterschiedlich wahr. Diese Eigenschaft wird in einem psychovisuellen Modell widergespiegelt. Das Verfahren sollte das menschliche Sehen berücksichtigen.

Flächenänderungen und Verzerrungen Ein Netz, das ein 3D-Modell bildet, besteht ausschließlich aus Dreiecken. Beim Markieren aber auch bei der Animation der 3D-Modelle werden die Flächen der Dreiecke geändert und Verzerrungen entstehen. Diese Flächenänderungen und Verzerrungen müssen mit berücksichtigt werden.

Position und Stellung Ein 3D-Modell ist typischerweise nur in seiner Ruhestellung abgespeichert. Bei Animationen befindet es sich nicht mehr in dieser Ruhestellung und aus diesem Grund muss diese das texturierte 3D-Modell nicht in der Ruhestellung bewerten, sondern alle Stellungen berücksichtigen.

Effizienz Die Herausforderungen bei der Stellung und Position münden in ein Optimierungsproblem, das möglichst effizient gelöst werden soll.

3.5 Robuster Hash für 3D-Modelle

Im Folgenden werden die Anforderungen an den robusten Hashalgorithmus eingeführt.

Effizienz: Damit ein robuster Hash praxistauglich ist, muss er effizient in der Komplexität und Geschwindigkeit sein. Das optimale Ziel ist es, möglichst nahe an die Effizienz von kryptografischen Hashes heranzukommen.

Sicherheit: Der robuste Hash muss sicher gegen typische Nachbearbeitungsschritte und verschiedene Angriffe sein. Typische Nachbearbeitungsschritte sind z.B. die Formatkonvertierung, Kompression, Vereinfachung oder das Neuordnen des Drahtgittermodells. Außerdem soll der Algorithmus sicher gegen Angriffe sein, die darauf abzielen, das Wasserzeichen im 3D-Modell zu zerstören und unlesbar zu machen. Im Umfang der Arbeit wird gefordert, dass der robuste Hash so lange gleich bleibt, wie das Wasserzeichen detektierbar ist. Ist das Wasserzeichen zerstört, so besteht für diese Anwendung kein Mehrwert, wenn der robuste Hash noch richtig extrahiert werden kann.

Hashlänge: Die Hashlänge soll variabel wählbar sein. Damit bleibt dem Anwender entsprechend seinem Bedarf die Möglichkeit, unterschiedliche Hashlängen zu verwenden. Die Hashlänge hat Einfluss auf die Anzahl der unterscheidbaren Objekte. Außerdem kann im Allgemeinen gesagt werden: Je kürzer der Hash, desto sicherer, aber auch fehlerträchtiger wird er.

Eindeutigkeit: Es sollen keine Kollisionen auftreten, sodass jedes Objekt eindeutig durch einen Hash repräsentiert wird. Eine Kollision tritt dann auf, wenn zwei oder mehr optisch unterschiedliche 3D-Modelle den gleichen Hash haben. In der Praxis ist die Wahrscheinlichkeit für eine Kollision bei robusten Hashverfahren größer verglichen zu kryptografischen Hashverfahren. Sie soll aber möglichst gering gehalten werden.

Fehlerraten: Robuste Hashverfahren werden anhand der *Falschakzeptanzrate* und *Falschrückweisungsrate* evaluiert (siehe Kapitel 2.8.1).

3.6 Videowasserzeichen

Der Fokus der Anforderungen an ein Videowasserzeichen hat sich gerade aufgrund des Streamings von Videocontent verändert:

Nachverarbeitungsschritte: Ein Video, das digital vorliegt, kann z.B. zur Reduktion des Speicherbedarfs in der Bitrate reduziert werden, auf eine geringere Auflösung skaliert werden, oder die Framerate reduziert werden. Eine weitere Möglichkeit besteht in einer verlustbehafteten Kompression, die bei Videos typischerweise das h.264 Format ist. Ein Videowasserzeichenalgorithmus muss mit diesen Nachverarbeitungsschritten umgehen können, sodass das Wasserzeichen auch nach einem der Schritte korrekt auffindbar ist. Dieser Aspekt fällt unter die Eigenschaft der Robustheit.

Angriffe: Neben den Nachverarbeitungsschritten kann ein Video gezielt so manipuliert werden, dass ein Wasserzeichen möglichst nicht mehr korrekt auffindbar ist. Darunter fallen nicht-lineares Skalieren, Zuschneiden in der Größe, leichtes Rotieren, horizontale Spiegelung und das Abfilmen eines Videos von einem Bildschirm oder einer Leinwand. Dieser Aspekt wird in den Eigenschaften unter der Sicherheit adressiert.

Nicht-Wahrnehmbarkeit: Veränderungen, die durch den Videowasserzeichenalgorithmus durchgeführt werden, dürfen nicht störend für den menschlichen Betrachter wirken. Dazu zählen bei Videos nicht nur die Änderungen innerhalb eines Frames, sondern auch solche, die an hintereinanderliegende Frames durchgeführt werden. Werden z.B. unterschiedliche Veränderungen in aufeinanderfolgenden Frames durchgeführt, so entsteht der Flickering Effekt (Flackern). Somit muss das Durchführen von Veränderungen die vorliegenden Frames berücksichtigen.

4 Fingerprint für Produkt- und Dokumentauthentifizierung

Dokumente werden meist mit einem Stempel und einer Unterschrift des Ausstellenden versehen. Damit ist weder der Ausstellende noch der Inhalt für einen Externen sicher überprüfbar. Im Bereich des Produktschutzes nutzen etablierte Verfahren z.B. Hologramme, Siegelsysteme, irisierende und fluoreszierende Sicherheitsfarben. Die Herausforderung liegt in der Prüfbarkeit dieser Sicherheitsmechanismen für z.B. Groß- und Einzelhändler, Zoll und Endkunden, Behörden und Unternehmen. Es besteht dabei der Konflikt, dass die Sicherheit der etablierten Verfahren auf der Geheimhaltung basiert, die Verfahren aber offengelegt werden müssten, um von jedermann verifizierbar zu sein. Handelt es sich um eine Privatperson, ist eine Offenlegung alleine nicht ausreichend, sondern eine technische Unterstützung zur Verifikation ist notwendig.

In der Literatur existieren verschiedene Ansätze, ein Fingerprint aus einem bedruckten Papier zu extrahieren. Chong et al. schlägt in [27] eine Methode vor, bei der ein Bereich des Bildes mit Phosphorpartikeln besprüht wird. Dieser behandelte Bereich der Verpackung wird mit einer Smartphone-Kamera aufgenommen, muss dazu aber auch mit einer UV-Lichtquelle ausgeleuchtet werden. Die Autoren behaupten, dass das Phosphor-Streumuster nicht klonbar ist, gehen aber nicht näher auf die Sicherheitseigenschaften der Methode ein. Andere Verfahren in der Literatur nutzen entweder ein Mikroskop zum Extrahieren des Merkmals [40, 92], Laser [24], oder Flachbrettscanner [28]. Ein Verfahren, ähnlich dem in diesem Kapitel vorgestellten, extrahiert Merkmale aus dem durch das Papier scheinenden Licht [96]. Die genannten Verfahren genügen den Anforderungen aus 3.1 nicht.

In diesem Kapitel wird ein Fingerprintverfahren für bedrucktes und unbedrucktes Papier vorgestellt. Es kann zur Dokument- und Produktauthentifizierung genutzt werden. Dazu ist es notwendig, dass der Fingerabdruck digital signiert und die Signatur in einem Barcode encodiert wird. Das Signieren und Encodieren der Daten in einem Barcode wurde in [105] ausgeführt und wird in der Arbeit nicht weiter berücksichtigt. Das im Folgenden vorgestellte Verfahren wurde im Umfang mehrerer betreuter Abschlussarbeiten entwickelt, umgesetzt, evaluiert [95, 76, 72] und publiziert [14].

4.1 Fingerprintalgorithmus

Zuerst wird das Fingerprintverfahren formal eingeführt, bevor auf die technische Umsetzung eingegangen wird. Vorausgesetzt wird in dieser Arbeit ein Bereich eines Substrats, der wahlweise unbedruckt oder homogen mit einer Farbe bedruckt ist. Der Bereich des Substrats S wird mit einer Kamera aufgenommen. In dieser Arbeit werden Smartphone-Kameras verwendet. Aus der Aufnahme wird der Fingerabdruck $F(S)$ extrahiert. Bei jeder Aufnahme S_x sollte das System in irgendeiner Weise zum gleichen Ergebnis kommen, ungeachtet des genutzten Smartphones. Zum Ausgleichen der Variationen des Fingerabdrucks aus den Aufnahmen wird eine Ähnlichkeitsfunktion benötigt. Ist der Unterschied zweier Fingerabdrücke geringer als ein Schwellwert, so wird angenommen, dass beide vom gleichen Substrat stammen. Ansonsten wird von unterschiedlichen Substraten ausgegangen. Für die Ähnlichkeit zweier Fingerabdrücke wird der Operator $\approx$ verwendet. Sind zwei Fingerabdrücke unterschiedlich, wird der Operator $\not\approx$ verwendet:

$$F(S) \approx F(S'), \text{wenn } S = S'$$

$$F(S) \not\approx F(S'), \text{wenn } S \neq S'.$$

Es wird ein vordefinierter Bereich A des Substrates S aufgenommen und daraus der Fingerabdruck $F(S_A)$ berechnet. Wir definieren:

$$F(S_A) \approx F(S'_{A'}), \text{wenn } A = A'S = S' \tag{4.1}$$

$$F(S_A) \not\approx F(S'_{A'}), \text{wenn } A \neq A'S = S' \tag{4.2}$$

$$F(S_A) \not\approx F(S'_{A'}), \text{wenn } A \neq A'S \neq S' \tag{4.3}$$

$$F(S_A) \not\approx F(S'_{A'}), \text{wenn } A = A'S \neq S' \tag{4.4}$$

4.1.1 Variationen in homogenen Flächen

Als Voraussetzung für einen Fingerabdruck ist es notwendig, dass mindestens ein Produktionsprozess zugrundeliegt, der unkontollierbare Variabilitäten aufweist, die mit den genutzten Messgeräten erfassbar sind [79]. Das Messen von Variabilitäten, die ein Druckprozess mit sich bringt, wird in zwei Druckqualitätsstandards beschreiben, die hier als Grundlage genutzt werden [47, 46]. Im Folgenden wird aufgezeigt, welche Merkmale der

Variabilitäten ausreichend Entropie bei Aufnahmen mit Smartphones liefern, damit sie als Fingerabdruck genutzt werden können.

Die Uniformitätsmetrik bewertet Variationen in homogenen Flächen. Nach der Aufnahme der Region findet eine Wavelet-Transformation statt, auf deren Basis die folgende Auswertung erfolgt. Zur Auswertung werden jedoch nur die Frequenzbänder genutzt mit weniger als 1,5 Zyklen pro Millimeter und damit drei Abtastwerten pro Millimeter. Bei ausreichender Ausleuchtung und einem nicht zu großen Abstand vom Objekt erreichen diese Abtastrate aktuelle Smartphones. Angelehnt an die Bewertungsmetrik für Fleckig- und Körnigkeit aus ISO/IEC 24790 ist das Fingerabdruckverfahren aufgebaut. Es wird eine Region definiert mit einer homogenen Fläche, wie in Abbildung 4.1 gezeigt. Die Suchmuster in den Ecken sind zum schnellen Auffinden und geometrischen Synchronisieren platziert worden. Für die gesamte Auswertung wird diese Region herangezogen und die digitale Datei bleibt unverändert. Die Suchmuster und die Region sind in ihrer Größe aufeinander abgestimmt, sodass die Möglichkeit zur Korrektur besteht, wenn z.B. Skalierung oder eine perspektivische Verzerrung vorliegt.

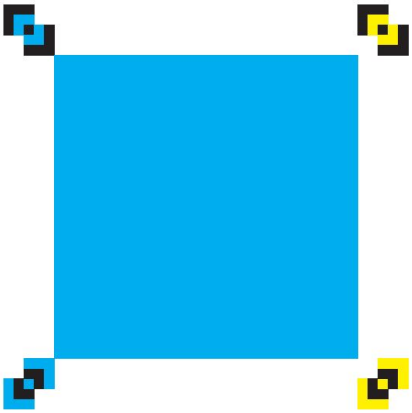


Abbildung 4.1: Testbild als homogene Cyan-Region mit vier Suchmustern in den Ecken.

4.1.2 Bildaufnahme und Fingerabdruck-Extraktion

Die Bildaufnahme wird über Android Smartphones durchgeführt. Eine App steuert die Belichtung, Autofokus und das Schreiben der RAW-Bilddaten. Der LED-Blitz ist bei der Aufnahme immer aktiv. Die Bilddaten werden ohne das Demosaicing, den Weißabgleich und Nachverarbeitungsschritte in voller Auflösung in einer Datenbank abgespeichert. Bei den Aufnahmen wird eine minimale Abtastrate von 20 Pixel pro Millimeter erwartet.

Die RAW-Bilddaten werden über das Demosaiken in ein RGB Bild gewandelt, um die Suchmuster zu lokalisieren und deren Größe zu bestimmen. Mit Hilfe des Zentrums der Suchmuster und deren Größe wird die Position der Region bestimmt und extrahiert. Es wird die Region um eine halbe Modulgröße am Rand verkleinert. So wird sichergestellt, dass im Falle einer nicht perfekten Synchronisierung keine Störeinflüsse durch die Kante der Region oder Kanten von außerhalb der Region mit in die Fingerabdruckberechnung einfließen.

Mit der Größe der Suchmuster von der Aufnahme (in Pixel) und der Größe vom digitalen Testbild (in mm) kann die Größe der Region (in pixel/mm) berechnet werden. Mittels einer Transformationsmatrix werden perspektivische Verzerrungen herausgerechnet, sodass aus der Region ein Quadrat der Größe 30 x 30 mm und 20 px/mm entsteht. Ist eine Aufnahme einer Region kleiner oder größer als 600 x 600 px, wird entsprechend hoch oder herunter skaliert und anschließend in den CIE XYZ Farbraum transformiert.

4.1.3 Waveletzerlegung

Zur Waveletzerlegung wird jeder Kanal getrennt herangezogen. Zur Transformation vom RGB Farbraum nach CIE XYZ wird die Formel 2.2 genutzt. Mittels der Daubechier Waveletzerlegung der Ordnung 16 wird der jeweilige Kanal in 8 Level zerlegt und analysiert. Ein Level wird auf Null gesetzt, wenn es nicht zur weiteren Analyse benötigt wird. Die verbleibenden Level werden mit der inversen Wavelettransformation in ein Bild zurücktransformiert, das nur die entsprechenden Frequenzen beinhaltet. Dieses resultierende Bild wird für den Fingerabdruck genutzt. In Tabelle 4.1 werden die Abtastrate (in Pixel pro mm), das Frequenzband (in Zyklen pro mm) und die Größe der Region (in Pixel) dem Level zugeordnet.

4.1.4 Fingerabdruckvergleich

Ob ein Fingerprint bereits bekannt ist, wird durch einen Abgleich mit der Datenbank beantwortet. Das Errechnen eines Fingerabdrucks aus einer Aufnahme I' und der Vergleich zum Template Fingerabdruck aus der Datenbank T' über den Korrelationskoeffizienten KC mit $KC \in [-1, 1]$ wird im Folgenden eingeführt. Sei w die Breite und h die Höhe der aufgenommenen Region in Pixel. Die Breite und Höhe beider Regionen I und T müssen übereinstimmen. $I'(x, y)$ und $T'(x, y)$ geben den Pixelwert der Fingerabdrücke an den Koordinaten (x, y), jeweils um den Mittelwert korrigiert:

$$T'(x, y) = T(x, y) - \frac{1}{w \cdot h} \sum_{x', y'} T(x', y') \tag{4.5}$$

Level	Frequenzband (cy/mm)	Abtastrate (px/mm)	Größe der Region
8	10 bis 5	20 bis 10	600 bis 300
7	5 bis 2,5	10 bis 5	300 bis 150
6	2,5 bis 1,25	5 bis 2,5	150 bis 75
5	1,25 bis 0,75	2,5 bis 1,25	75 bis 37,5
4	0,75 bis 0,375	1,25 bis 0,75	37,5 bis 18,75
3	0,375 bis 0,1875	0,75 bis 0,375	18,75 bis 9,375
2	0,1875 bis 0,09375	0,375 bis 0,1875	9,375 bis 4,6875
1	0,09375 bis 0,046875	0,1875 bis 0,09375	4,6875 bis 2,3437

Tabelle 4.1: Skalierungslevel der Wavelettransformation mit den entsprechend geforderten Größen, Abtastraten und Frequenzbändern

$$I'(x,y) = I(x,y) - \frac{1}{w \cdot h} \sum_{x',y'} I(x',y') \tag{4.6}$$

$$KC = \frac{\sum_{x,y} (T'(x,y) \cdot I'(x,y))^2}{\sqrt{\sum_{x,y} T'(x,y)^2 \cdot \sum_{x,y} I'(x,y)^2}} \tag{4.7}$$

Zuerst wird CIE Y für den Fingerabdruckvergleich genutzt. Gilt ein Fingerabdruck als bekannt, wird im V-Kanal des HSV-Farbraums die aufgenommene Region auf Homogenität geprüft. Nur wenn beide Bedingungen erfüllt sind, nämlich in CIE Y der Fingerabdruck als bekannt gilt und V aus HSV homogen ist, ist der Fingerabdruck erfolgreich authentifiziert.

4.1.5 Fingerabdruck Homogenität

Die Prüfung auf Homogenität des Fingerabdrucks im Farbkanal V des HSV-Farbraums erfolgt über das Bilden zweier Histogramme, die voneinander subtrahiert werden. Das eine Histogramm H_h beinhaltet alle Pixel aus Regionen, die zu den hellen Regionen des binarisierten Fingerabdrucks gehören. Das zweite Histogramm H_d beinhaltet alle Pixel aus Regionen, die zu den dunklen Regionen des binarisierten Fingerabdrucks gehören.

Der binarisierter Fingerabdruck wurde extrahiert, indem die Level drei und vier der Aufnahme der originalen Region mit der inversen Wavelet-Transformation auf [0, 1] normiert und mit dem Schwellwert von 0,5 binarisiert wurden. Das so erhaltene Schwarz-Weiß-Bild gilt als der binarisierte Fingerabdruck. Drei Beispiele von unterschiedlichen Druckern sind in Abbildung 4.2 gezeigt.

Abbildung 4.2: Binarisierte Fingerabdrücke: Links Canon IP, Mittig HP Laserjet und rechts MAN Roland.

Die beiden Histogramme wurden binweise B zueinander verglichen mit: $|H_d - H_h|$. Ein Fingerabdruck gilt als homogen wenn gilt $\sum_B |H_d - H_h| < \tau$, sonst gilt der Fingerabdruck als heterogen. Ein geeigneter Wert für τ wird im Folgenden bestimmt.

4.2 Evaluierung

Im Folgenden wird das beschriebene Verfahren hinsichtlich der Anforderungen aus Kapitel 3.1 evaluiert.

4.2.1 Testaufbau

Zur Evaluierung wurde eine Testseite entworfen mit Blöcken, die 30 mm × 30 mm groß waren. Diese Größe wurde gewählt, um die im Standard [46] festgelegte digitale Bildgröße zu erreichen. Bei einer Aufnahme von 20 Pixeln pro Millimeter entspricht es einer Größe von 600 × 600 Pixel. Die Blöcke enthalten jeweils eine der Grundfarben (CMYK) und einer der Blöcke wurde nicht bedruckt, um die Eigenschaft des Substrats selbst zu untersuchen. Ein Beispiel wird in Abbildung 4.1 gezeigt. Damit sichergestellt ist, dass alle Drucker und Treiber die gleichen Farbinformationen erhalten, wurden die Blöcke im PDF/X-3:2002-Format exportiert.

Zur Evaluierung wurden acht verschiedene Drucker berücksichtigt. Auf jedem Drucker wurde die gleiche PDF-Datei mehrfach gedruckt. Zwei davon (HP OfficeJet 5220 und Canon Pixma IP 4000) sind handelsübliche Tintenstrahldrucker, einer (HP Color LaserJet M276nw) ist ein handelsüblicher Laserdrucker und einer (BizHub C458) ein Büro-Laserdrucker. Zur Evaluierung verschiedener Substrate wurde auf zwei verschiedenen Papiersorten von Bürokopierpapier gedruckt, wobei beide ein Gewicht von $80g/m^2$

aufweisen. Außerdem wurden eine Reihe von Verpackungen, auf denen dieselben Blöcke gedruckt waren, bei der Evaluierung mit berücksichtigt. Diese Drucke wurden mit professionellen, großformatigen Offset- und Digitaldruckmaschinen erstellt. Zum Einsatz kamen die vier Druckmaschinen Heidelberg Speedmaster XL105-8P, MAN Roland R704+L, HP Indigo 30000 und HP Indigo 12000. Das Verpackungsmaterial wird normalerweise nach dem Druck mit einem Schutzlack zur Veredelung überzogen. Dieser Lack wurde auf alle Offset- und Digitaldrucke aufgetragen, mit Ausnahme eines der Offsetdrucke, der unveredelt blieb.

Die Aufnahmen dieser gedruckten Blöcke wurden mit zwei verschiedenen Smartphones (Huawei Mate 9 und Nokia 5) mit deren Rückkamera gemacht, wobei die Bilder mit dem LED-Blitz beleuchtet wurden. Ein weiteres Smartphone (Samsung Galaxy S8) wurde später verwendet, um die Funktionalität des Verfahrens mit einem neuen Gerät zu überprüfen. Die Smartphones waren bei den Aufnahmen in einem Stativ befestigt, das 14 cm im Abstand zum Substrat eingestellt war und eine Verschiebung in x- und y-Richtung zuließ. Es wurde darauf geachtet, dass bei der Aufnahme der hellste Teil der LED-Leuchte oder des Displays immer so nah wie möglich an der Mitte des Bildes lag.

Zur Evaluierung von Umgebungsbeleuchtungseinfluss wurde mit drei LED-Tischlampen (Taotronics TT-DL27) getestet. Sie lassen vier Farbtemperatureinstellungen zu. Weitere Raumbeleuchtung wurde abgeschaltet, sodass ein Einfluss von weiterem Licht ausgeschlossen werden kann.

Somit werden in der Evaluierung acht Drucker, vier Druckverfahren, fünf Papiertypen, drei Smartphones, vier Lichttemperaturen bei einem festen Abstand der Smartphones zur Region berücksichtigt. Es wurden insgesamt 576 Regionen aufgenommen und dienen als Testset. Das Testset wurde zufällig in zwei Gruppen geteilt mit 476 Aufnahmen in einer Gruppe zum Extrahieren der optimalen Parameter und die verbleibenden 100 Aufnahmen zum Evaluieren der Parameter. Bei der Gruppe zum Extrahieren der optimalen Parameter ergeben sich insgesamt 113.050 Vergleiche. Das Verhältnis zwischen den Bildern in der Gruppe der AR mit 476 und RR mit 112.574 ist nicht ausgewogen. Das Ungleichgewicht wird bei der Evaluierung mit berücksichtigt, z.B. bei der Abschätzung von Kollisionen durch eine Approximation zweier Verteilungen.

4.2.2 Parameterwahl

In diesem Abschnitt wird ausgewertet, welche der Wavelet-Level für die Evaluierung zu nutzen sind. Es werden die Korrelationen für die acht Level untersucht, die mit den beiden verschiedenen Kameras für die unterschiedlichen Blöcke, jeden Abdruck mit und ohne Blitz, aufgenommen wurden. Außerdem wollen wir die Ergebnisse für die unterschiedlichen Farben und das Substrat analysieren, um festzustellen, welche Konfiguration die besten

Ergebnisse bringt. Das bedeutet eine höchstmögliche Korrelation aus Gleichung 4.7 für ein und dasselbe gedruckte Testbild und gleichzeitig die geringstmögliche Korrelation für jedes der anderen Testbilder des Satzes.

Um für das Szenario die beste Kombination von Wavelet-Level zu finden, wurde eine Scoring-Funktion benötigt. Wie bei den Anforderungen in Kapitel 3.1 eingeführt, ist eine hohe Kollisionssicherheit gefordert. Zur Berechnung des Scores wurden der Interquartilsbereich IQR und sowohl das erste als auch dritte Quartil verwendet. Diese Werte werden zur Erkennung von Ausreißern verwendet. Zunächst müssen das erste Q_1 und dritte Quartil Q_3 für die RR und AR in einem Trainingsdatensatz berechnet werden.

Der Interquartilsbereich (IQR) lässt sich aus den beiden Quartilen berechnen mit $IQR = Q_3 - Q_1$. Daraus können weiterhin die oberen und unteren Grenzen (OG und UG) in den AR t und RR f ermittelt werden mit $OG_f = Q_3(f) + 1,5 \cdot IQR(f)$ und $UG_t = Q_1(t) - 1,5 \cdot IQR(t)$. Mit OG und UG errechnet sich der Score aus der Differenz der unteren und oberen Grenze mit

$$Score = UG_t - OG_f. \tag{4.8}$$

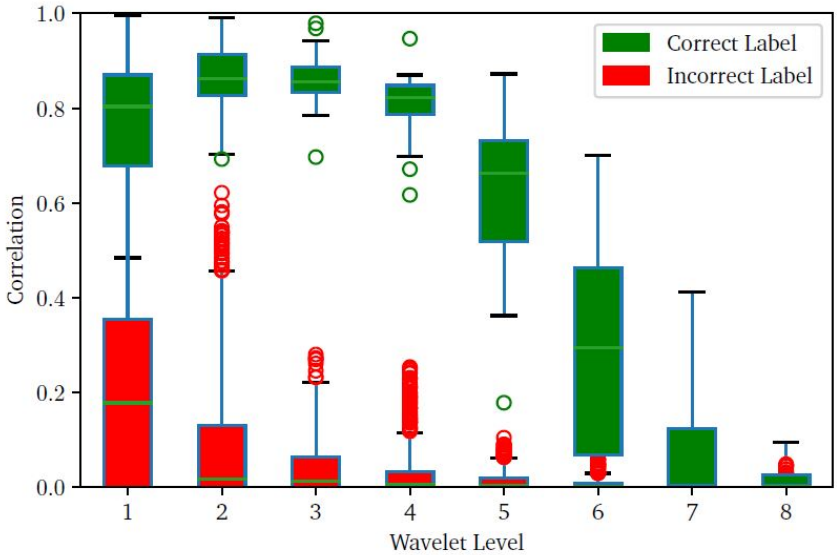


Abbildung 4.3: Korrelationen der AR und RR Aufnahmen der Cyan-Blöcke.

Der genutzte Datensatz wurde für jeden Block separat unter den im vorherigen Abschnitt 4.2.1 definierten Bedingungen mit der Rückkamera des Nokia 5 und des Huawei

Level	Substrat	Magenta	Cyan	Schwarz	Yellow
1	-0,287	-0,836	-0,381	-1,516	-0,496
2	0,102	0,182	0,245	-0,616	0,028
3	0,571	0,516	0,557	0,459	0,539
4	0,578	0,508	0,587	0,457	0,547
5	0,346	0,292	0,301	0,211	0,405
6	-0,104	-0,174	-0,166	-0,216	-0,099
7	-0,123	-0,265	-0,275	-0,270	-0,185
8	-0,042	-0,174	-0,135	-0,078	-0,150

Tabelle 4.2: Score der Blöcke für jedes Wavelet Level für Konsumerdrucker

Mate 9 fotografiert. Abbildung 4.3 zeigt ein Boxplot der Korrelationen der Cyan-Blöcke ohne Berücksichtigung der Drucke der Druckmaschinen. Die grünen Boxen zeigen die Korrelationen für die AR derselben Blöcke an, während die roten Boxen die Korrelationen zwischen allen anderen Cyan-Blöcken darstellen. Jede grüne Box besteht aus 24 Vergleichen, während jede rote Box aus 1104 Vergleichen besteht. Die abgebildeten oberen und unteren Whisker werden beim Berechnen des Scores verwendet. In Tabelle 4.2 wird die Punktzahl für jede Wavelet-Stufe für jeden der Blöcke berechnet. In der Tabelle werden alle Daten der hinteren Kamera miteinander korreliert. Die Level 3,4 und 5 trennen zwischen den AR und RR, wobei Level 3 und 4 vergleichbar sind und Level 5 daran gemessen eine geringere Trennung aufweist. Die Level 6-8 sind zu verrauscht, um zuverlässig zu trennen, und Level 1 enthält wahrscheinlich zu wenig Informationen. Level 2 trennt, außer bei Schwarz.

Die Offset- und Digitaldrucke wurden auf die gleiche Weise fotografiert, jedoch variierte die Beleuchtung, da die Bildaufnahme aufgrund der Größe der Drucke etwas komplexer war. Wie aber in Abschnitt 4.2.4.1 gezeigt wird, ist das Verfahren invariant gegen unterschiedliche Lichttemperatur, so dass das Ergebnis nicht wesentlich beeinflusst sein sollte. Hierfür wurden allerdings nur Cyan- und Substratblöcke ausgewertet. Insgesamt wurden alle 48 Cyan-Blöcke der beiden Speedmaster-Offsetdrucke fotografiert, von jedem Substratblock wurden jedoch nur zehn Bilder aufgenommen. Da die beiden Indigo-Drucker das gleiche Substrat verwendeten, wurden nur die Substratblöcke auf den Indigo 30-Materialien fotografiert. Aus demselben Grund wurde das gestrichene Speedmaster-Substrat nicht fotografiert, da der Roland-Offsetdrucker dasselbe Substrat verwendete. Von allen anderen Offset- und Digitaldrucken wurden zehn Cyan- und Substratblöcke fotografiert.

Level	Substrat	Cyan	Indigo12	Indigo30	Roland	HSM-U	HSM-C
1	-0,722	-0,488	-0,395	-0,321	-0,101	-1,085	-0,189
2	-1,102	-0,758	0,311	0,225	-0,525	-1,649	-0,340
3	0,311	0,278	0,466	0,547	0,301	0,163	0,274
4	0,386	0,331	0,541	0,533	0,642	0,216	0,556
5	0,211	0,096	0,359	0,413	0,506	0,090	0,394
6	-0,148	-0,160	-0,081	0,160	0,157	-0,165	0,041
7	-0,324	-0,223	-0,186	-0,034	-0,160	-0,168	-0,232
8	-0,033	-0,165	-0,286	-0,255	-0,136	-0,074	-0,244

Tabelle 4.3: Score nach Gleichung 4.8 der Cyan- und Substratblöcke für jedes Wavelet-Level von Ausdrucken der fünf berücksichtigten Druckmaschinen

Tabelle 4.3 fasst die Ergebnisse für Offset- und Digitaldrucke zusammen. In der zweiten und dritten Spalte von links werden die Ergebnisse der Druckmaschinen im Gesamten für Aufnahmen von Substrat- und Cyan-Blöcken angegeben, während rechts die Ergebnisse pro Druckmaschine nur für die Cyan-Blöcke aufgeschlüsselt sind, da die Ergebnisse für die Substrat-Blöcke zu denen der Cyan-Blöcke sehr ähnlich sind und keine weiteren Erkenntnisse liefern. Die beste Trennung über alle Farben und Drucker hinweg wurde wiederum in den Wavelet-Level drei und vier erzielt. Nur der unveredelte Offsetdruck erzielte bei den Cyan-Blöcken niedrigere Werte als alle anderen getesteten Druckmaschinen.

Die Daten zeigen bei den gleichen Wavelet-Level für jeden Druckertyp vergleichbare Ergebnisse. Die Intensitäten der Reflexionen sowie die Form der Muster variieren jedoch stark. Abbildung 4.4 gibt zu den verschiedenen Druckern das Graustufenbild der Cyan-Blöcke. Die Reflexion der Tintenstrahldrucke ist am geringsten, wobei auch da Helligkeitsunterschiede gut erkennbar sind. Die Laserdrucke zeigen ähnliche Reflexionsmuster wie die Digitaldrucke. Das zentrierte Licht der Smartphones wird über das Bild reflektiert. Die veredelten Offsetdrucke liefern ein fast spiegelähnliches, kreisförmiges Reflexionsmuster. Dennoch ist ein komplexes Muster erkennbar, wie Abbildung 4.5, was auf die unebene Oberfläche der Verpackung zurückzuführen ist.

Im letzten Schritt wurde mit Hilfe des eingeführten Scores aus Gleichung 4.8 ausgewertet, welche Kombination der Wavelet-Level die besten Trenneigenschaften liefert und welcher Schwellwert genutzt werden soll zur Trennung zwischen dem Annehmen einer Region als bekannt und dem Ablehnen der Region als unbekannt.

Dazu wurden 576 Bilder genutzt, mit denen insgesamt 165.600 Vergleiche durchgeführt wurden. Zwischen den AR und RR liegt ein Ungleichgewicht vor. Es liegen 468 Bilder in der Gruppe AR entgegen 165.132 in RR zu Grunde. Als Resultat konnte mit einem Score

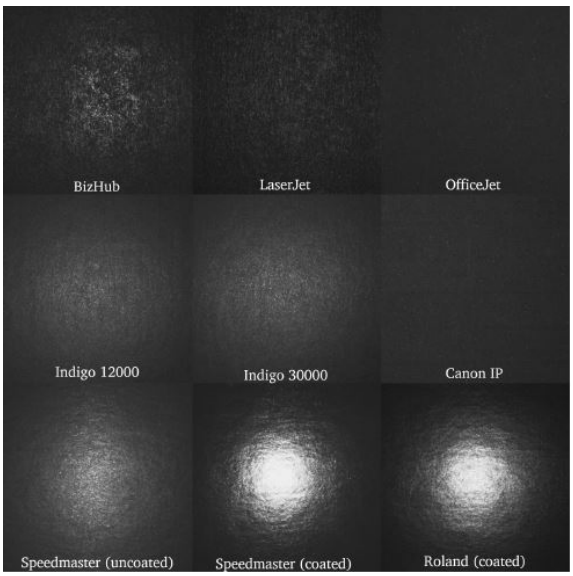


Abbildung 4.4: Reflexionsbild von Cyan-Blöcken, die mit den evaluierten Druckern und Druckmaschinen gedruckt und bei der Aufnahme mit dem Blitz des Smartphones ausgeleuchtet wurden.

von $0,457$ die Levelkombination 3 und 4 identifiziert werden, mit der im Folgenden alle weiteren Evaluierungen durchgeführt wurden.

Zum Finden des Schwellwerts wurde der Datensatz in einen Trainings- und einen Testdatensatz unterteilt, wobei der Testdatensatz $1/3$ des Datensatzes betrug. Die Daten aus dem Trainingsdatensatz wurden dann verwendet, um einen Schwellwert zu bestimmen. Dazu wurden wieder UG_t und OG_f aus der Scoreberechnung verwendet. Der Schwellwert wurde als der Wert zwischen den beiden festgelegt mit $threshold = UG_t - \frac{UG_t - OG_f}{2}$, womit ein maximaler Abstand von beiden UG_t und OG_f erreicht wird. Für die Daten ergab sich der Schwellwert als $threshold = 0,335$.

Für die weitere Auswertung wurde mit diesem Schwellwert evaluiert. Für die Trainings- und Testdaten konnte mit diesem Schwellwert die Trennung ohne FAR oder FRR erreicht werden.

4.2.3 Substrateinfluss

Die Ergebnisse in Tabelle 4.2 und 4.3 zeigen für das Substrat vergleichbare Ergebnisse wie die farbigen Blöcke. In diesem Abschnitt soll geklärt werden, welchen Einfluss die

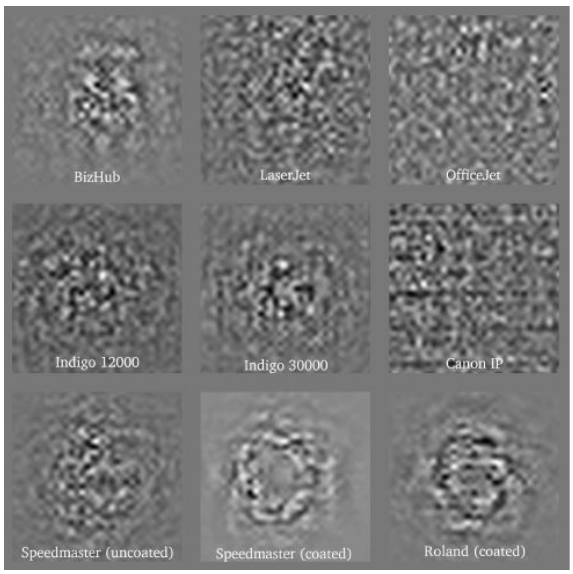


Abbildung 4.5: Level 3 und 4 aus den Aufnahmen aus Abildung 4.4. Das Histogramm ist zur Visualisierung gestreckt worden.

Oberflächenstruktur des Substrats auf den endgültigen Fingerabdruck hat und wie die auf das Substrat gedruckte Tinte den extrahierten Fingerabdruck verändert. Dazu wurde mit dem Canon-Tintenstrahldrucker zunächst nur Finderpattern auf sechs Papiere gedruckt. Damit lag jeweils ein Substrat-Block vor. Davon wurde jeweils ein Bild aufgenommen und die Wavelet-Level drei und vier extrahiert. Anschließend wurden jeweils Farben auf den Substrat-Block mit Yellow, Magenta, Cyan und Key aufgedruckt und nach jedem Druck jeweils wieder ein Fingerabdruck extrahiert. Schließlich wurden die vier Fingerabdrücke mit dem des Substrats korreliert, um zu sehen, inwieweit der Substrat-Fingerabdruck noch durchscheint. Außerdem wurden zwei Experimente durchgeführt, bei denen mehrere Farben hintereinander gedruckt wurden. Zum einen wurde auf den Substratblock erst Yellow und anschließend Cyan gedruckt und zum anderen zuerst Yellow, dann Cyan und zuletzt Magenta. Abbildung 4.6 zeigt die Ergebnisse. Die Ergebnisse veranschaulichen, in welchem Maße die Textur des Substrats bei Tintenstrahldruckern noch durchscheint. Insbesondere bei Yellow sind die extrahierten Merkmale sehr ähnlich denen des Substrats, wohingegen bei Key die Merkmale des Substrats am wenigsten durchscheinen.

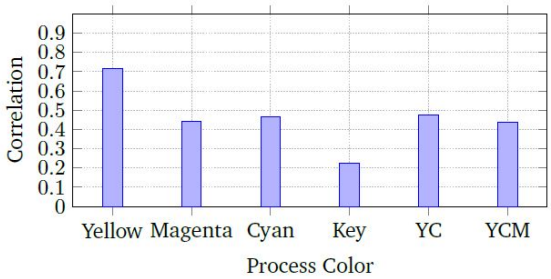


Abbildung 4.6: Korrelationen eines Fingerabdrucks eines Substratblocks mit dem Fingerabdruck nach dem Überdrucken mit einer Farbe. YC steht für die Druckreihenfolge Yellow dann Cyan und bei YCM wurde noch Magenta über Yellow und Cyan gedruckt.

4.2.4 Robustheit

In diesem Abschnitt wird die Robustheit des Fingerabdrucks in verschiedenen Szenarien evaluiert. Der Datensatz dient bei der Evaluierung der Robustheit als Referenz. Es wurde in den Unterkapiteln jeweils ein Umgebungseinflussfaktor variiert, ein neues Bild aufgenommen und zum Datensatz korreliert. Der Schwellwert aus vorhergehendem Abschnitt wird über die gesamte Evaluierung verwendet. Wie die Ergebnisse zeigen, haben der Abstand und die Lichttemperatur keinen signifikanten Einfluss auf die Ergebnisse, somit nutzen wir als Parameter ein Abstand von 14 cm und zwei Lichtquellen mit 6000 Kelvin. Die Kamera wurde mittig über der aufzunehmenden Region ausgerichtet.

4.2.4.1 Beleuchtung

Der Einfluss der Beleuchtung auf den Fingerabdruck wurde mit fünf verschiedenen Lichtverhältnissen aufgenommen und die Veränderungen der Korrelation bei den Lichtverhältnissen analysiert. Dazu wurden zwei TaoTronics TT-DL27 Schreibtischlampen mit einstellbarer Lichttemperatur genutzt. Die Aufnahmen wurden mit dem Huawei und Samsung Smartphone aufgenommen. Die anderen Parameter waren die gleichen wie im vorhergehenden Abschnitt eingeführt. Die Schreibtischlampen wurden so positioniert, dass sie das Testbild von zwei Seiten aus einem Abstand von etwa 30 cm beleuchteten. Die Aufnahmen wurden bei Lichttemperaturen von 3000, 4000, 5000, 6500 Kelvin, bei Neonlicht und bei ausgeschaltetem Licht aufgenommen. Jede Aufnahme wurde mit dem entsprechenden Referenzblock korreliert. Die Ergebnisse in Abbildung 4.7 zeigen nahezu identische Korrelationen unabhängig von der externen Lichtquelle. Die Korrelation ist bei

den unterschiedlichen Druckern insgesamt weit über dem Schwellwert. Zusammenfassend kann gesagt werden, dass wechselnde Lichtverhältnisse der Umgebung keinen signifikanten Einfluss auf die Korrelationen der Fingerprints haben, da die Differenz der minimalen und maximalen Korrelation für einen Druck bei unterschiedlichen Lichttemperaturen geringer ist als $0,15$ und die Korrelation oberhalb des Schwellwerts liegt.

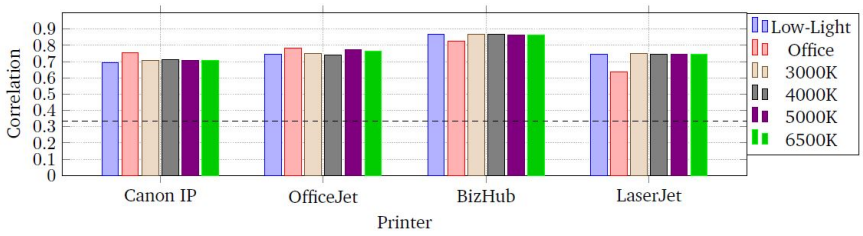


Abbildung 4.7: Die Korrelationen der Fingerabdrücke werden hier in Abhängigkeit von verschiedenen Druckern und unter verschiedenen Umgebungsbeleuchtungen abgetragen. Low-Light gibt das Ergebnis bei ausgeschaltetem Umgebungslicht in einem dunklen Raum wider, während “Office“ die Aufnahmen unter Neonbeleuchtung eines Büros zeigt. Die Ergebnisse für die Lichttemperatur von 3000, 4000, 5000 und 6500 Kelvin wurden mit zwei Schreibtischlampen des Typs TaoTronics TT-DL27 durchgeführt.

4.2.4.2 Rotation

Zur Evaluierung des Einflusses der Rotation auf die Robustheit wurden die Aufnahmen in unterschiedlichen Winkeln durchgeführt. Dazu wurde das Bild unter der Kamera in 15° Schritten gedreht, der Fingerabdruck extrahiert und zum Referenzfingerabdruck aus dem Datensatz korreliert. Es wurden zwei Testdrucke ausgewählt, ein cyanfarbener Block aus dem Set der Laserdrucke und ein cyanfarbener Block aus dem Set der Tintenstrahldrucke. Bei der Auswertung wurde die Rückseitenkamera des Huawei als Bildaufnahmegerät verwendet. Die Ergebnisse sind in Abbildung 4.8 dargestellt. Die Korrelation für jeden Fingerabdruck ist als Punkt in Blau in dem Winkel aufgetragen, in dem die aufgenommene Region unter der Kamera gedreht wurde. Der Median der Korrelation für alle Drehungen des Laserdrucks betrug 0,64, während der Median für den Tintenstrahldruck 0,7 betrug. Der Median ist als grüner Kreis in der Abbildung 4.8 eingezeichnet. Aufnahmen ohne einen Winkel weisen eine Korrelation für den Laserdruck von 0,74 auf, während bei dem Tintenstrahldrucker eine Korrelation von 0,76 erreicht wird. Die niedrigste Korrelation lag bei 0,47 für den Laserdruck bei einem Winkel von etwa $210°$. Die Rotation hat einen

geringen Einfluss auf die Robustheit des Fingerabdrucks. Die niedrigste Korrelation liegt oberhalb des Schwellwerts, der in Rot in der Abbildung eingezeichnet ist.

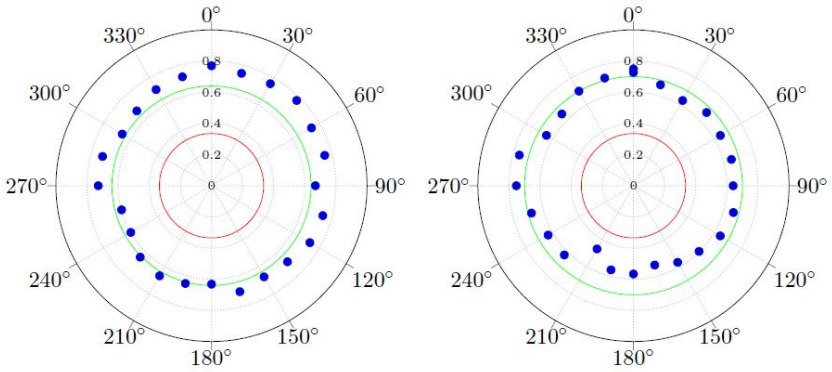


Abbildung 4.8: Die Korrelationen der Fingerabdrücke werden hier in 15° Schritten durchgeführt und als blaue Punkte an dem entsprechenden Winkel abgetragen. Links werden neben den Korrelationen der Median in Grün und der Schwellwert in Rot für einen Laserdruck und rechts für einen Tintenstrahldruck gezeigt.

4.2.4.3 Ausrichtung

In diesem Abschnitt wird der Einfluss der Ausrichtung der Kamera auf die aufzunehmende Region evaluiert. Es ist davon auszugehen, dass die Smartphone-Kamera nicht zentrisch auf die Region ausgerichtet sein wird. Dazu wurde mit Millimeterpapier ein Raster mit 11 Schritten von 0,5 cm erstellt und das Testbild in jeder Richtung vom Zentrum der Kamera wegbewegt. Für jede der drei Drucktechnologien, Offset-, Laser- und Tintenstrahldruck, wurde jeweils ein Cyan-Block zufällig ausgewählt. Für die Bewertung beim Laserdruck stammt der Cyan-Block vom HP Laserjet, beim Tintenstrahldruck vom HP Officejet und beim Offsetdruck vom Heidelberg Speedmaster. Für diese Evaluierung wurde die Huawei-Rückseitenkamera verwendet. Das LED-Licht strahlte relativ stark in die Mitte jeder Region. Als Umgebungslicht wurde für die Tests mit den Tinten- und Laserstrahldruckern eine Lichttemperatur von 6500 K verwendet, während für den Offsetdruck die Neonbeleuchtung genutzt wurde. Die Ergebnisse sind als Heatmap in Abbildung 4.9 dargestellt.

In der Abbildung wurden negative Korrelationen auf 0 gesetzt. Jede Aufnahme wird an ihrer Position relativ zur Mitte des Testdrucks in x- und y-Richtung aufgetragen. Die Aufnahme, bei der die Kamera mittig über der Region ausgerichtet war, wurde als Referenz

genutzt, zu der die umliegenden Aufnahmen korreliert wurden. Das schwarze Rechteck bezeichnet die äußeren Ränder der gedruckten Region im Abstand von 1,5 cm vom Zentrum in alle Richtungen. Außerhalb des schwarzen Rechtecks traf das LED-Licht nicht mehr direkt auf die Region.

Der Tintenstrahldruck zeigt sich invariant gegenüber der Ausrichtung. Auch wenn das Licht nicht direkt auf den Druck traf, blieben die Korrelationen hoch. Der Laserdruck zeigt, dass sobald kein direktes Licht mehr auf die Region trifft, die Korrelationen rapide fallen. Der veredelte Offsetdruck reagiert sehr empfindlich, sobald die Kamera nicht exakt auf das Zentrum der Region ausgerichtet war. Selbst wenn der Mittelpunkt des Lichts nur etwa 1 cm vom Zentrum der Region entfernt war, konnte der Fingerabdruck nicht erfolgreich zugeordnet werden.

Zusammenfassend kann gesagt werden, dass die Technik nur bei Tintenstrahl- und Laserdruck robust gegenüber Translationen ist. Bei Anwendungen, die diesen Fingerabdruck nutzen, sollte daher z.B. eine Methode genutzt werden, die misst, wann der Mittelpunkt der Region mit der maximalen Lichtstärke ausgeleuchtet ist, und nur dann auszulösen. Als Grund für diese Beobachtung wird die Vermutung angestellt, dass dieses Verhalten durch die Reflexion der Oberfläche herbeigeführt wird. In Abbildung 4.5 wurde gezeigt, dass gerade beim Offsetdruck die Fingerabdrücke sehr stark auf dem Kreis um das Zentrum herum extrahiert werden. Wenn sich das Zentrum verschiebt, weil die Kamera nicht mehr mittig auf die Region ausgerichtet ist, verändert sich der Fingerabdruck entsprechend.

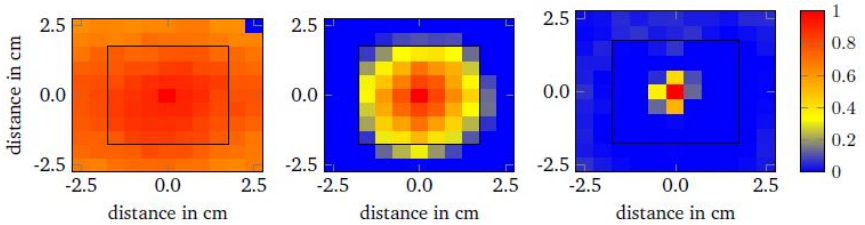


Abbildung 4.9: Die drei Heatmaps zeigen links die Korrelationen der Fingerabdrücke für einen Tintenstrahldrucker, mittig bei einem Laserdrucker und rechts für einen Offsetdrucker. Auf beiden Achsen wird die Abweichung der Ausrichtung vom Zentrum der Region abgetragen, wobei Blau eine niedrige Korrelation darstellt und Orange und Rot hohe Korrelationen.

4.2.4.4 Abstand

Zur Evaluierung des Einflusses des Abstands der Kamera zur aufzunehmenden Region wurde ein Kamerastativ mit drei verschiedenen Höheneinstellungen verwendet. Dieses Stativ ermöglichte es, Aufnahmen in einer Höhe von 14 cm, 21,5 cm und 26,5 cm zu machen. Die Aufnahmen wurden nur mit dem Huawei-Smartphone und mit einer Lichttemperatur von 6500 Kelvin durchgeführt. Die Pixel pro Millimeter der Kamera sanken auf durchschnittlich 14,9 bei 21,5 cm und auf 11,9 bei 26,5 cm. Bei jeder Änderung der Höhe wurde die Kamera so ausgerichtet, dass sie auf die Mitte der Region zeigte. Der extrahierte Fingerabdruck in jeder Höhe wurde mit dem entsprechenden Referenzfingerabdruck aus dem Datensatz verglichen.

Obwohl sich die Abtastrate des Bildes für verschiedene Höhen änderte, war sie bei einem Abstand von 26,5 cm immer noch hoch genug. Da die Form und Größe des Lichts, das auf die gedruckten Bilder fällt, bei unterschiedlichen Höhen variiert, wurde erwartet, dass die Höhe entsprechend einen Einfluss auf die Korrelation hat. Die Ergebnisse in Abbildung 4.10 widerspiegeln die Erwartung.

Der Einfluss des Abstands der Kamera zur aufgenommenen Region auf die Korrelation ist bei den professionellen Druckmaschinen stärker zu beobachten im Vergleich zu den Druckern. Bei den Druckmaschinen nimmt die Korrelation stärker ab, je größer der Abstand wird. Der Schwellwert wurde zwar in keinem Fall unterschritten, aber beim veredelten Speedmaster-Druck lag die Korrelation nur leicht darüber.

Auch hier können die Ergebnisse mit dem Reflexionsmuster zusammenhängen. Bei den Offset Drucken ist es zentriert und kreisförmig, siehe Abbildung 4.5, was sich bei veränderter Höhe mit ändert. Zusammenfassend kann gesagt werden, dass der Fingerabdruck robust gegenüber kleinen Höhenänderungen ist. In der Praxis ist es empfehlenswert, den Nutzer bei der Aufnahme zu unterstützen, sodass er einen geeigneten Abstand findet.

4.2.5 Sicherheit

In diesem Abschnitt wird die Sicherheit des Fingerabdrucks bewertet. Dazu wird zunächst die Kollisionssicherheit betrachtet und die Wahrscheinlichkeit geschätzt, dass ein Hersteller mehrere Regionen mit demselben Fingerabdruck erzeugt. Abschließend wird gezeigt, wie ein Klonangriff erfolgreich erkannt wird.

4.2.5.1 Kollisionssicherheit

Die Kollisionssicherheit betrachtet die Wahrscheinlichkeit für ein Szenario, in dem ein Hersteller unabsichtlich zwei Produkte herstellt, die von der Methode als identisch einge-

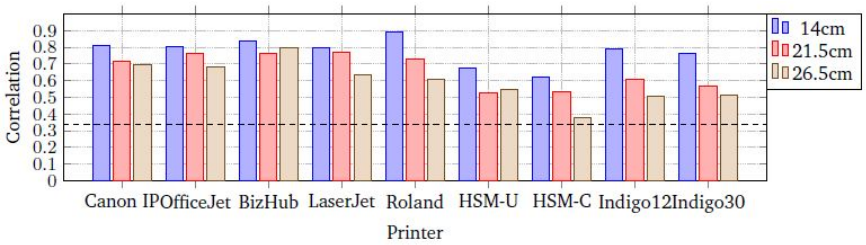


Abbildung 4.10: Korrelation der aufgenommenen Drucke in Abhängigkeit der verschiedenen Drucker und Abstände zu den Regionen. Bei den Drucken mit den Druckmaschinen ist ein stärkerer Abfall der Korrelationen abzulesen als bei den Druckern.

stuft werden. Die Modellierung erfolgt, indem eine Verteilung auf die empirischen Daten gelegt wird und deren kumulative Verteilungsfunktion die Wahrscheinlichkeit schätzt, ab wann ein bestimmter Schwellenwert überschritten wird. Eine Kollision tritt ein, wenn die Korrelation zwischen zwei Fingerabdrücken den Schwellwert überschreitet. Eine Kollision ist hier ein FAR Ereignis. Eine fehlgeschlagene Erkennung ist ein FRR Ereignis und tritt ein, wenn die Korrelation von zwei Fingerabdrücken derselben Region unter den Treshold fällt.

Für die Schätzung der FAR wurde derselbe Datensatz verwendet wie zum Finden des Schwellwerts. Zum Bestimmen guter Approximationen der AR und RR der empirischen Daten wurde der Kolmogorov-Smirnov-Test sowie der Chi-Quadrat-Test verwendet, um die Anpassung einer Reihe verschiedener Verteilungen zu schätzen. Die Student's t-Verteilung hatte den höchsten p-Wert in den Chi-Quadrat-Tests und Kolmogorov-Smirnov-Tests für die RR Werte. Die Beta-Verteilung approximierte die AR am besten. Anhand der kumulativen Verteilungsfunktion (CDF) konnte die Wahrscheinlichkeit von FRR und FAR geschätzt werden. Bei dem hier gewählten Schwellwert von $0,335$ ergibt sich so eine Wahrscheinlichkeit für FRR von $6,09 \cdot 10^{-4}$ und eine FAR von $5,7 \cdot 10^{-6}$.

4.2.5.2 Klonangriff

Beim Klonangriff wurde der extrahierte Fingerabdruck eines Originals auf eine Kopie übertragen, indem der binarisierte Fingerabdruck (siehe Abschnitt 4.1.5) mittig auf die kopierte Region abgeschwächt addiert und anschließend gedruckt wurde.

Die Tests wurden mit Cyan- und Substratblöcken durchgeführt. Die Angriffe auf die cyanfarbenen Blöcke wurden mit dem Canon-Tintenstrahldrucker gedruckt und mit dem Huawei-Smartphone fotografiert. Die Angriffe auf Substratblöcke hingegen wurden mit

Drucker	Korr. Substrat	Korr. Cyan	Homo. Substrat	Homo. Cyan
Canon IP	0,609	0,721	6,9	7,1
LaserJet	0,610	0,738	6,5	6,7

Tabelle 4.4: Korrelation und Homogenität von Substrat- und Cyanblöcken für zwei Drucker. Die Korrelation und die Homogenität übersteigen die jeweiligen Schwellwerte. Somit sind die Klone nicht authentifiziert und als Fälschung erkannt.

Drucker	Deckkraft in %	Correlation	Homogenität
Canon IP	30	0,721	7,1
Canon IP	10	0,42	3,6
Canon IP	5	0,293	2,0
Canon IP	0	0,0811	0,3

Tabelle 4.5: Korrelation und Homogenität von Substrat- und Cyanblöcken für zwei Drucker. Die Korrelation und die Homogenität übersteigen die jeweiligen Schwellwerte. Somit sind die Klone nicht authentifiziert und als Fälschung erkannt.

dem LaserJet gedruckt und ebenfalls mit dem Huawei-Smartphone fotografiert. Alle Bilder wurden unter normalen Bürolichtbedingungen aufgenommen. Die Fingerabdrücke wurden von den Aufnahmen der Klone auf die gleiche Weise berechnet wie von denen im Datensatz.

Als Schwellwert für die Korrelation wurde auch hier $0,335$ genutzt und $\tau = 2,1$. In Tabelle 4.5 sind die Korrelationen für die beiden Drucker und Substrate sowie deren Homogenität aufgelistet.

Die Korrelationen sind zwar über dem Schwellwert, jedoch gelten sie auch als heterogen, da sie τ übesteigen und damit kann der Klonangriff erfolgreich erkannt werden.

4.3 Zusammenfassung

In diesem Abschnitt wurde ein Fingerprintverfahren vorgestellt, das es ermöglicht, Druckerzeugnisse mit Smartphones zu authentifizieren. Dabei wurden die aus Kapitel 3.1 gestellten Anforderungen erreicht. Das Verfahren wurde so konzipiert, dass es unter verschiedenen Umgebungseinflüssen und der Verwendung unterschiedlicher Substrate und Druckverfah-

ren robust mit Smartphones den Fingerabdruck extrahieren lässt. Im Falle von Einschränkungen bei Offset Drucken wurden Lösungsvorschläge zur Verbesserung der Robustheit für die praktische Umsetzung gegeben.

Als Grundlage für den Fingerabdruck wurden zwei ISO Standards genutzt, die eine Qualitätsbewertung von Druckerzeugnissen definieren. Es wurde eine dort beschriebene Methode modifiziert und umgesetzt. Die eindeutige Identifizierung eines Druckerzeugnisses wird auf der Grundlage der Gleichmäßigkeit von monochromen Blöcken oder der Oberfläche des Substrats durchgeführt. Das Verfahren wurde hinsichtlich der Anforderungen an die Robustheit, Sicherheit und öffentliche Verifikation bewertet.

Die Ergebnisse der Evaluierung zeigen einzigartige, wieder identifizierbare Informationen im Rauschen bei mehreren Ausdrucken des gleichen Druckers und der gleichen Eingabedatei. Untersucht wurden eine Reihe verschiedener Drucker, Substrate und Smartphones unter sich ändernden Umgebungsbedingungen. Die Druckerzeugnisse von professionellen Offset- und Digitaldrucken zeigten Optimierungspotential auf, wobei der Reflexion dabei vermutlich die entscheidende Rolle zukommt. Neben einer guten Kollisionssicherheit für Fingerprints konnte auch der Klonangriff erkannt werden.

Zur Erhöhung der Robustheit bei Offset Druckerzeugnissen kann in zukünftigen Arbeiten geprüft werden, ob entweder die Überbelichtung bei der Aufnahme herausgerechnet werden kann, sodass das Reflexionsmuster nicht mehr den überproportionalen Beitrag zum Verfahren leistet, oder bei der Veredelung auf einen nicht so stark reflektierenden Lack bei der Region zurückgegriffen werden sollte.

Die Robustheit wurde nicht mit verschiedenen Nutzern getestet. Dies kann bei weiteren Forschungsarbeiten nachgeholt werden, sodass auch deren Verhalten mit einfließen kann. Um die Sicherheit des gesamten Systems zu verbessern, könnten weitere Identifikationsmerkmale berücksichtigt werden, die z.B. durch ein komplexeres Testbild erreichbar sind.

5 3D-Modell-Wasserzeichen

Im Bereich von digitalen Wasserzeichen für 3D-Modelle existieren verschiedene Ansätze basierend auf volumetrischen Modellen [41] und Drahtgittermodellen [109, 110, 101, 55]. Weitere Verfahren Modifizieren die Geometrie [54, 34], Toplogie [21, 67] oder ein 3D-Modell im Spektrum [43, 65]. Alle genannten Verfahren erfüllen jedoch nicht die Anforderungen aus Kapitel 3.2 an die Robustheit und Transparenz. 3D-Modell-Wasserzeichen sind im Vergleich zu anderen Medientypen, wie z.B. Audio, Video oder Bild, ein relativ junges Forschungsgebiet. Die verschiedenen Dateiformate bei 3D-Modellen unterscheiden sich neben der Verarbeitungsgeschwindigkeit durch Grafikkarten auch in der Repräsentation des Modells.

In diesem Kapitel wird der eigens entwickelte 3D-Wasserzeichenalgorithmus präsentiert. Die Evaluierung wird auf 3D-Modellen aus Videospielen durchgeführt. Der Algorithmus erwartet 3D-Modelle, die als *Polygonnetze*, oder auch *Drahtgittermodelle* genannt, vorliegen. Sie sind unter anderem bei Videospielen verbreitet. Durch Konvertierungen können die Modelle zu Polygonnetzen und wieder zurück konvertiert werden.

Der Wasserzeichenalgorithmus modifiziert die Knotenverteilung in der räumlichen Domäne und lässt nur solche Änderungen zu, die auf einem spektral komprimierten 3D-Modell vorgenommen wurden. Damit sind nur Änderungen in niedrigen Frequenzen, also dem *Kernmodell*, zulässig. Änderungen an hohen Frequenzen werden unterbunden. Damit das bestmögliche Einbettungsergebnis in kürzester Zeit erreicht wird, basiert der Wasserzeichenalgorithmus auf einem evolutionären Algorithmus [3].

Der Wasserzeichenalgorithmus eignet sich nicht für das Markieren von Konstruktionsteilen, bei denen aufgrund von Passgenauigkeit keine Änderungen vorgenommen werden dürfen.

Der im Folgenden vorgestellte Algorithmus wurde im Rahmen einer betreuten Masterarbeit [97] entwickelt und in [99] veröffentlicht.

5.1 3D-Wasserzeichenalgorithmus

Der hier vorgestellte Wasserzeichenalgorithmus für 3D-Modelle ist für Drahtgittermodelle konzipiert. Sie sind hauptsächlich in den drei Dateiformaten Polygon File Format (PLY), Wavefront Object Format (OBJ) und dem NetImmerse File Format (NIF) zu finden. Die Dateiformate, ebenso wie der Aufbau der Drahtgittermodelle, sind in [97] näher beschrieben.

Der Algorithmus benötigt neben dem 3D-Modell einen geheimen Schlüssel, die einzubettende Wasserzeichennachricht und die Parameter zur Steuerung der Robustheit, Transparenz und Datendichte.

In diesem Abschnitt wird zuerst der Einbettungsalgorithmus beschrieben, der in drei Teile untergliedert ist. Anschließend wird der Detektionsalgorithmus eingeführt. Der Ablauf des Einbettungsalgorithmus ist in Abbildung 5.1 dargestellt. Er untergliedert sich in den SChritten der *Vorverarbeitung*, *Histogramm-Modifikation* und *Rekonstruktion*. Der evolutionäre Algorithmus verbessert das Einbettungsergebnis, indem er iterativ die Modifikationen des Histogramms verstärkt. Zum Erreichen einer hohen Robustheit bei gleichzeitig guter Transparenz modifiziert der Algorithmus nur das Kernmodell des Drahtgitters.

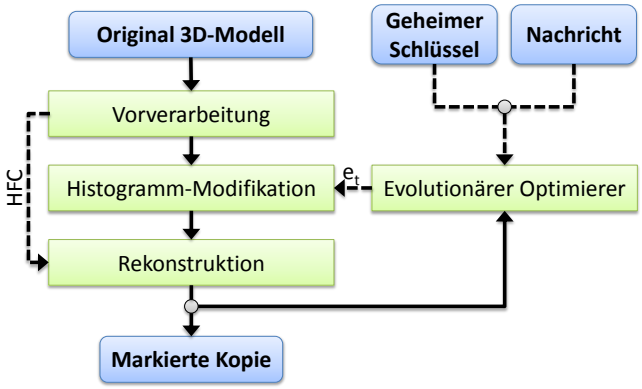


Abbildung 5.1: Überblick des Wasserzeichenalgorithmus für 3D-Modelle

5.1.1 Vorverarbeitungsschritt

Im Vorverarbeitungsschritt wird das 3D-Modell in die Frequenzdomäne überführt. Anschließend wird eine Kompression durchgeführt, um das Kernmodell zu erhalten. Auf Basis des Kernmodells wird in einem nächsten Schritt ein stabiler Bezugspunkt mittels des kleins-

ten umschließenden Balls bestimmt. Mithilfe des Bezugspunktes führt der Algorithmus eine Koordinatentransformation durch.

Frequenzraum Zum Transformieren der Knotenkoordinaten des Drahtgitters in eine Frequenzrepräsentation werden die Eigenvektoren E des originalen 3D-Modells entsprechend Karni et al. [51] berechnet. Mithilfe der Eigenvektoren und der Knoten des 3D-Modells M kann das Modell in den Frequenzraum C durch Berechnen von $C = E^T \cdot M$ gewandelt werden.

Spektrale Kompression Anschließend werden die Frequenzkoeffizienten, die die hohen Frequenzen (HFC) repräsentieren, separat gespeichert und dann auf 0 gesetzt. Durch diese Modifikation repräsentiert C das Kernmodell des ursprünglichen 3D-Modells. Sei C^c die Frequenzrepräsentation des spektral komprimierten 3D-Modells, τ ein gewählter Schwellwert, N die Anzahl der Knoten und $t \in \{1, ..., N\}$, dann gilt:

$$C_t^{\mathbf{c}} := \begin{cases} 0, & \forall t > \tau \\ C_t, & \text{sonst} \end{cases} \tag{5.1}$$

Der Algorithmus berechnet $(M^c)^T = (C^c)^T \cdot E^T$, um vom Frequenzraum zurück in die räumliche Domäne zu gelangen. Mit Θ^T ist die transponierte der Matrix Θ gemeint, und (M^c) bezeichnet das spektral komprimierte 3D-Modell in der räumlichen Domäne.

Robuster Bezugspunkt durch die iterative Berechnung des Miniball-Zentrums Nachdem das spektralkomprimierte 3D-Modell M^c berechnet wurde, wird ein robuster Bezugswert bestimmt, auf dem die Wasserzeicheneinbettung aufbaut. Dazu wird das Zentrum des Miniballs bestimmt. Der Miniball ist der kleinstmöglichste Ball (*smallest enclosing ball* SEB), der alle Knoten umschließt [37], wie in Kapitel 2.6.2 eingeführt. Alternativ könnte auch das Gravitationszentrum (Center of Gravity) [26] verwendet werden, jedoch ist beim Gravitationszentrum die Robustheit etwas schlechter als beim SEB.

Die Berechnungen werden aus Gründen der Performanz auf den Kugelkoordinaten durchgeführt und entsprechend ist die Umwandlung von kartesischen Koordinaten in Kugelkoordinaten notwendig.

Es existieren mehrere Algorithmen zur Berechnung des SEB und dessen Zentrum. Das Wasserzeichenverfahren verwendet die beiden Implementierungen von Gärtner und Badoiu-Clarkson. Der Algorithmus von Gärtner performed besser und wird zuerst verwendet. Kommt er zu keinem gültigen Resultat, wird der Algorithmus von Badoiu-Clarkson

zum Bestimmen des Zentrum des SEBs genutzt. Die Performanz des Algorithmus von Gärtner wurde auf Kosten der Genauigkeit optimiert. Er enthält zwei Werte, die *accuracy*, die die maximale Distanz beschreibt zwischen dem errechneten SEB und den Knoten, die außerhalb vom SEB liegen. Der zweite Wert, *slack* genannt, beschreibt die Abweichung zum angenommenen Optimum. Gärtner empfiehlt das Ergebnis zu akzeptieren, wenn $slack = 0$ und $accuracy < e^{-15}$ gilt. Weichen die Werte von diesen Schranken ab, wird auf den Algorithmus von Badoiu-Clarkson zurückgegriffen.

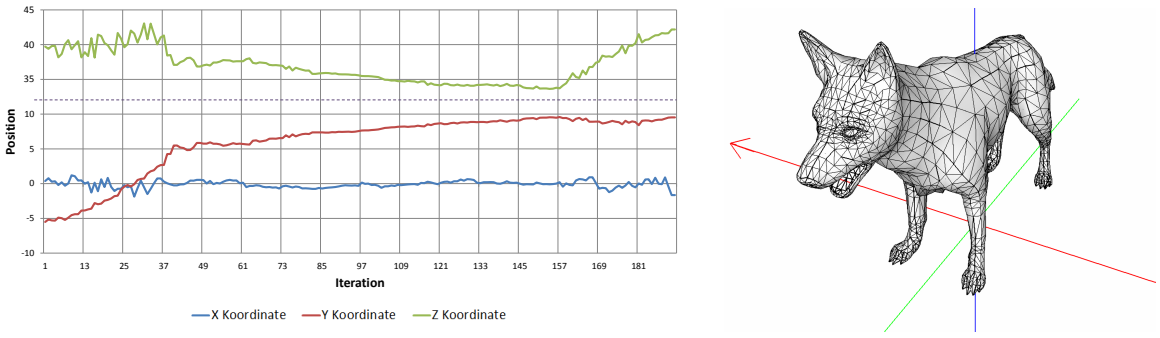
Der Algorithmus berechnet den Bezugspunkt als Zentrum der Silhouette des Kernmodells. Damit der Bezugspunkt in der Robustheit weiter gestärkt wird, berechnet der Algorithmus iterativ das Zentrum des SEB und lässt in jeder Iteration die Hilfsknoten des Kernmodells außen vor. Dieser Schritt macht den Bezugspunkt weniger sensibel gegen das Löschen von Knoten. Die Iterationen werden so lange durchgeführt, bis sich das Zentrum des SEB nicht mehr als ein vorgegebener Schwellwert verändert. Solange die Proportionen des 3D-Modells beibehalten werden, ist der so berechnete Bezugspunkt stabil. Schlussendlich wird das berechnete Zentrum des SEB als Bezugspunkt genutzt, um das Modell in die Kugelkoordinaten umzuwandeln.

Der SEB Algorithmus wird in der ersten Iteration mit dem Kernmodel $\mathbb{V}$ und dem entsprechenden Knotenset P_1 initialisiert. Nach jedem Iterationsschritt i wird das Knotenset P_i um die Hilfsknoten S_{i-1} aus dem vorhergehenden Schritt verringert, sodass gilt:

$$\langle \vec{c}, r, S \rangle_i := SEB(P_i) \quad mit \quad P_i = \begin{cases} \mathbb{V}, & \text{wenn} \quad i = 1 \\ P_{i-1} \setminus S_{i-1}, & \text{sonst} \end{cases}$$

Als Ergebnis liefert der Algorithmus $\langle \vec{c}, r, S \rangle$, wobei $\vec{c}$ das Zentrum des SEB ist und damit der Bezugspunkt für die weiteren Berechnungen, r der Radius und S die neuen Hilfsknoten.

Abbildung 5.2 zeigt das Zentrum des SEB in seinen x, y und z-Koordinaten, abhängig von der Anzahl der Iterationen. Nach etwa 50 Iterationen wird die Position des Zentrums stabiler, bis 150 Iterationen, danach ändert sich die Position des Zentrums wieder stärker. Für dieses Modell ist das Zentrum des SEB zwischen 50 und 150 Iterationen zu wählen. Die Koordinaten des Zentrums zeigen ein Driften in der stabilen Region. In der Praxis ist die genaue Anzahl an notwendigen Iterationen unbekannt. Damit der Detektor in der Lage ist den gleichen Bezugspunkt zu finden, wird ein Abbruchkriterium eingeführt. Der Algorithmus bricht mit dem Iterieren ab, sobald $\lfloor \frac{1}{3}\mathbb{V} \rfloor$ überschritten wird, wobei mit $\mathbb{V}$ die Anzahl der Knoten gemeint ist. Damit innerhalb dieser Anzahl an Iterationen ein Minimum erreicht wird, wird mit einem gleitenden Fenster gearbeitet.



Verlauf der x, y und z-Koordinaten

“Dog“ 3D-Modell

Abbildung 5.2: Berechnung des Bezugspunktes für das “Dog“ Modell mithilfe des Iterativen SEB

Gleitendes Fenster Das gleitende Fenster gewichtet die Position des Zentrums nach jeder Iteration. Dabei werden niederfrequente Bereiche höher bewertet als hochfrequente Bereiche. Dafür wird ein gewichteter Mittelwert über die Positionen des Zentrums der letzten N_{win} Iterationen berechnet. Die verwendeten Gewichte stammen aus der *Hann*-Fensterfunktion, die auch als *raised cosine window function* bekannt ist.

Sei $\overline{\vec{c_i}}$ der Mittelwert des i-ten Zentrums mit $i \geq N_{win}$, $u \in x(\overline{\vec{c_i}}), y(\overline{\vec{c_i}}), z(\overline{\vec{c_i}})$ eine der drei Koordinaten, $\tau = \frac{1}{2}N_{win}$ und W die Summe über alle Gewichte w_k, dann lässt sich das gewichtete Zentrum des SEB wie folgt berechnen:

$$u(\overline{\vec{c_i}}) := \frac{1}{W}\sum_{k=1}^{N_{win}} w_k \cdot u(\vec{c}_{(i-k)}) \quad \text{mit} \quad w_k = \frac{1}{2}(1 + cos(\frac{2\pi(k-\tau)}{N_{win}}))$$

Mit der Hann-Fensterfunktion werden Werte in der Mitte des Fensters höher gewichtet als Werte, die an den Rändern liegen. In einem letzten Schritt wird abschließend jedes Fenster gewichtet. Sei $v \in x(\widetilde{\vec{c_i}}), y(\widetilde{\vec{c_i}}), z(\widetilde{\vec{c_i}})$ eine der drei Koordinaten der gewichteten Fenster des Zentrums, $F_s > 0$ die fade-out Stärke und R die Anzahl an Iterationen, die der Algorithmus durchlaufen hat, dann gilt:

$$v(\widetilde{\vec{c}_i}) := \frac{1}{W'} \sum_{k=1}^{R} \overline{w_i} \cdot v(\vec{\bar{c}}_{(i)}) \quad \text{mit} \quad \overline{w_i} = \widetilde{w_i} \frac{1}{v(\sigma_i^2) + \epsilon}, \tag{5.2}$$

$$\widetilde{w_i} := 1 - \frac{1}{1 + e^{q(i)}} \quad \text{und} \quad q(i) = 2\pi - \frac{i - N_{win}}{F_s} \tag{5.3}$$

Als fade-out Funktion wird die Sigmoidalfunktion genutzt. Abbildung 5.3 zeigt das Resultat der Glättung mithilfe der Gleichung 5.3 für das “Dog“ 3D-Modell. Das Zentrum ist mit der Zunahme der Iterationen stabil und zeigt kein Driften oder Erhöhen von Schwankungen auf. Mit diesem gewichteten Zentrum des SEB lässt sich der Bezugspunkt für alle weiteren Rechenschritte bestimmen.

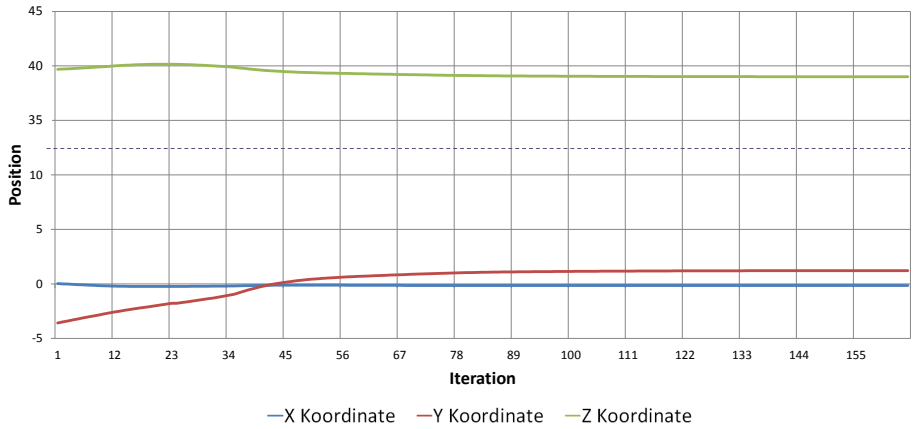


Abbildung 5.3: x, y und z-Koordinaten des stabilen Bezugspunkt in Abhängigkeit der Anzahl der Iterationen nach dem Gewichten

Umrechnen der Kartesischen- in Kugelkordinaten Die komprimierte Repräsentation des 3D-Modells $M^c(x, y, z)$ wird von den kartesischen in die Kugelkoordinaten $M^\circ(r, \phi, \theta)$ transformiert. Die Kugelkoordinaten werden durch den Abstand r_i zwischen den Knoten $v_i \in \mathbb{V}$ und dem berechneten Bezugspunkt $\vec{c}$ und zwei Winkeln repräsentiert:

$$r_i := \sqrt{(x(\vec{v}_i) - x(\vec{c}))^2 + (y(\vec{v}_i) - y(\vec{c}))^2 + (z(\vec{v}_i) - z(\vec{c}))^2}$$
$$\phi_i := \arctan(\frac{x(\vec{v}_i) - x(\vec{c})}{x(\vec{v}_i) - x(\vec{c})}$$
$$\theta_i := \arccos((z(\vec{v}_i) - z(\vec{c}))\frac{1}{r_i})$$

5.1.2 Histogramm-Modifikation

Der Algorithmus erstellt anschließend ein Histogramm der sortierten Radien. Der Wasserzeichenalgorithmus modifiziert beim Einbetten der Nachricht den Abstand der Knoten zum Bezugspunkt und belässt dabei die beiden Winkel, Azimut- und Polarwinkel. Dazu werden Knoten des originalen Kernmodells in Gruppen, die im Folgenden *Bin* genannt werden, zusammengefasst. Abhängig vom einzubettenden Nachrichtenbit wird der Abstand der Knoten eines Bins zum Bezugspunkt verringert oder vergrößert.

Erzeugen der Bins Ein geringer prozentualer Anteil Φ_{skip} der größten und kleinsten Radien wird aus der Bin-Erzeugung ausgeschlossen, um das Risiko, dass eine zu stark variierende Anzahl an Knoten pro Bin erzeugt wird, zu minimieren. Die verbleibenden Radien werden in N_{Bin} viele und gleich große Bins b_j, mit $j \in 1, ..., N_{Bin}$, aufgeteilt. Zwischen zwei disjunkten Bins wird ein kleiner Abstand Φ_{gap}, im Folgenden auch *Gap* genannt, eingehalten. Die beiden Parameter Φ_{gap} und Φ_{skip} werden empirisch gewählt.

Jeder Radius wird dem entsprechenden Bin, mit seiner Ober- und Untergrenze b_j^{max}, b_j^{min}, zugeordnet. Liegt ein Radius in einem Gap, so wird er dem nächst liegenden Bin zugeordnet. Nachdem alle Radien einem Bin zugeordnet wurden, normalisiert der Algorithmus die Radien r_i eines Bins b_j auf das Intervall $[-1, 1]$:

$$\widehat{r}_i := 2 \cdot \frac{r_i - b_j^{min}}{b_j^{max} - b_j^{min}} - 1. \tag{5.4}$$

Bin-Modifikation Durch das Modifizieren der Radien verringert oder erhöht der Wasserzeicheneinbettungsalgorithmus die Varianz der Radien innerhalb der Bins. Dazu potenziert der Algorithmus jeden Radius $\widehat{r}_i$ mit einem gewählten Exponenten e_i. Das Resultat ist ein modifizierter und normalisierter Radius $\widehat{r}_i^*$:

$$\widehat{r}_i^* := sig(\widehat{r}_i) \cdot |\widehat{r}_i|^{e_i} \quad \text{mit} \quad sig(x) = \begin{cases} +1 & \text{wenn } x \geq 0 \\ -1 & \text{sonst} \end{cases} \quad \text{und } e_i \in \mathbb{R}.$$

Durch die Normalisierung wird sichergestellt, dass die resultierenden Radien $\widehat{r}_i^*$ innerhalb der Grenzen ihres Bins liegen und nur auf dem Intervall $[-1, 1]$ modifiziert werden. Das Wählen von geeigneten Exponenten e_i für jeden Radius ist der aufwendigste Teil der Einbettung. Es wird von einfachen Ansätzen, die z.B. alle Exponenten gleich groß wählen, abgesehen, weil so kein robustes, sicheres und transparentes Wasserzeichen erzeugt werden kann. Für das Finden von geeigneten Exponenten, die nur das Kernmodell modifizieren, wird ein evolutionärer Optimierer in Kapitel 5.1.3 vorgestellt.

Schlüsselabhängiges Zuordnen der Bins Ein vom Rechteinhaber geheimgehaltener Schlüssel ist bei digitalen Wasserzeichen essentiell. Der Schlüssel schützt die Wasserzeichennachricht gegenüber unautorisiertem Auslesen und vor Angriffen, die darauf abzielen, das Wasserzeichen zu löschen oder zu ändern. Ein Wasserzeichenbit wird in mehrere Bins eingebettet und der Schlüssel bestimmt die Zusammengehörigkeit der Bins.

Zwei Gruppen α_i und β_i aus disjunkten Bins werden für jedes Wasserzeichenbit m_i mit $i \in 1, ..., N_{bits}$ benötigt, wobei N_{bits} die Länge des Wasserzeichens ist. Die Anzahl der Bins muss deshalb mindestens doppelt so groß sein wie die Länge des Wasserzeichens, damit $N_{Bin} \geq 2 \cdot N_{bits}$ erfüllt ist. Dazu initialisiert der Algorithmus einen Zufallsgenerator mit dem Schlüssel, um die Bins zufällig den Gruppen α_i oder β_i für jedes Wasserzeichenbit m_i zuordnen zu können. Die Anzahl der Möglichkeiten $\mathbb{B}$, die dem Algorithmus für die Zuordnung zur Verfügung stehen, können wie folgt berechnet werden:

$$\mathbb{B} = \prod_{i=1}^{2 \cdot N_{bits}} \binom{N_{Bin} - (i-1)\frac{N_{Bin}}{2 \cdot N_{bits}}}{\frac{N_{Bin}}{2 \cdot N_{bits}}}.$$

Durch Einsetzten der verwendeten Parameter $N_{Bin} = 16$ und $N_{bits} = 4$, ergeben sich $\mathbb{B} \approx 81$ Milliarden Zuordnungsmöglichkeiten. Das entspricht einer Schlüssellänge von etwa 36 Bits. In Abbildung 5.4 wird beispielhaft die Zuordnung der Bins zu ihren Gruppen eines bestimmten Wasserzeichenbits dargestellt.

5.1.3 Evolutionärer Optimierer

Zu einem gegebenen Drahtgittermodell und einer Wasserzeichennachricht soll ein passendes Set von Exponenten gefunden werden. Werden alle Exponenten gleich oder zufällig

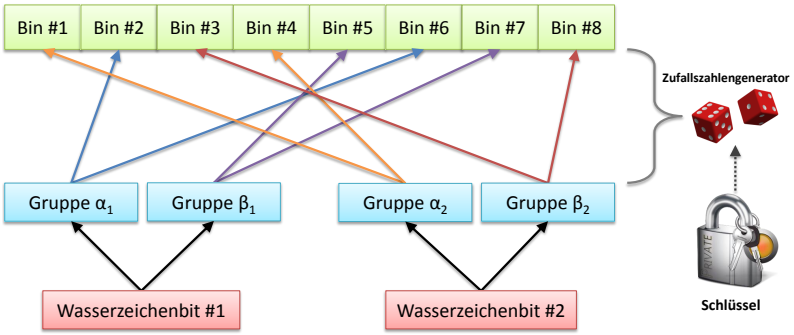


Abbildung 5.4: Beispielhafte Zuordnung der Bins zu den Gruppen eines bestimmten Wasserzeichenbits

gewählt, dann ist die Wasserzeichennachricht nicht robust, denn die meisten Änderungen werden schon durch die spektrale Kompression ignoriert. Optimale Exponenten zu finden ist ein klassisches Optimierungsproblem. Der Wasserzeichenalgorithmus für 3D-Modelle bestimmt das Set von Exponenten mithilfe eines evolutionären Optimierers. Alternativen, die bei Optimierungsproblemen existieren, wie z.B. das Gradientenverfahren, führen hier im Allgemeinen nicht zum gewünschten Optimum, da kein konvexes Problem vorliegt.

Abhängig vom einzubettenden Wasserzeichenbit m_i muss die Varianz der Gruppen modifiziert werden. Soll das Bit "1" eingebettet werden, dann muss die Ungleichung $Var(\alpha_i) > Var(\beta_i)$ erfüllt werden. Soll hingegen das Bit "0" eingebettet werden, dann muss die Varianz des Histogramms der entsprechenden Bins so modifiziert werden, dass die Ungleichung $Var(\alpha_i) < Var(\beta_i)$ erfüllt ist. Diese Modifikationen lassen sich über die Exponenten e_i für jeden Knoten v_j steuern. Für das Vergrößern der Varianz sind die Exponenten mit $e_j > 1$ zu wählen, soll hingegen die Varianz verringert werden, so sind die Exponenten mit $e_j < 1$ zu wählen.

Da das Wasserzeichen komplett und eindeutig nach einer Kompression auslesbar sein soll, muss die Varianz der Bins so modifiziert werden, dass Veränderungen nur in den niedrigen Frequenzen vorkommen und keine in den hohen Frequenzen. Es müssen also Exponenten gefunden werden, die die Radien der Knoten so verändern, dass das 3D-Modell nur in den niedrigen Frequenzen Änderungen erfährt. Da aber ein Zusammenhang zwischen den Änderungen der Radien in der räumlichen Domäne und den Auswirkungen auf die niedrigen Frequenzen sehr komplex ist, können auch keine Abschätzungen getroffen werden. Nur ein iterativer Einbettungsalgorithmus kann diese Herausforderung lösen. In jedem Schritt sind die Radien kontrolliert zu ändern und deren Auswirkung auf die

Frequenzen zu verifizieren. In Abbildung 5.5 wird der Ablauf des Einbettungsprozesses gezeigt.

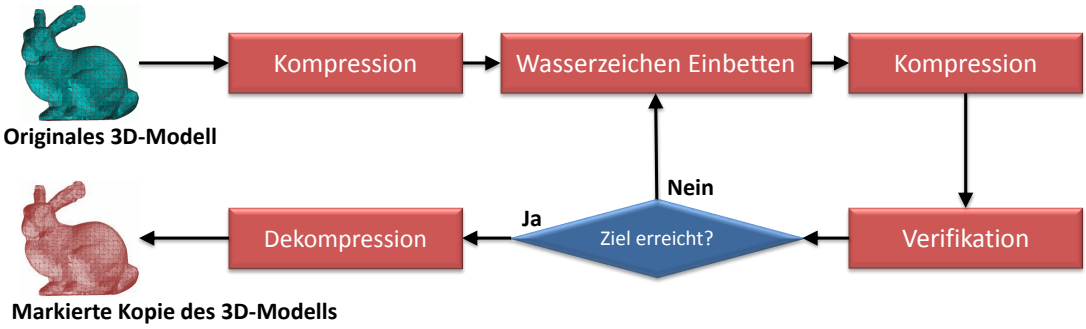


Abbildung 5.5: Einbettungsstrategie

Die Einbettungsstrategie sieht das Modifizieren der Knoten des 3D-Modells auf dem komprimierten 3D-Modell vor. Anschließend wird das modifizierte Modell wieder komprimiert und das Modell mit den verbleibenden Änderungen verifiziert. Bei der Verifikation überprüft der Algorithmus wie robust das Wasserzeichen mit den verbleibenden Änderungen ist. Dazu vergleicht er das detektierte mit dem einzubettenden Wasserzeichen. Weichen beide Wasserzeichen nicht voneinander ab und ist das detektierte Wasserzeichen ausreichend stark, hört der Algorithmus auf und fügt die hohen Frequenzen beim Dekompressionsschritt hinzu. Sind diese beiden Kriterien noch nicht erreicht, werden die Knoten weiter modifiziert, das Modell anschließend komprimiert und das neue Ergebnis verifiziert. Der Algorithmus durchläuft diese Schritte solange, bis beide Kriterien erreicht sind.

Damit das Wasserzeichen in jedem Iterationsschritt robuster wird, aber trotzdem transparent bleibt, wird ein entsprechendes Minimierungsproblem definiert. Ziel ist es, den Vektor aus Exponenten $(e_1, ..., e_N) \in \mathbb{R}^N$ so zu wählen, dass die falsch detektierten Wasserzeichenbits minimiert werden $f_g : \mathbb{R}^N \to \mathbb{N}_0$. Das Wählen der Exponenten wird mit einem evolutionären Algorithmus durchgeführt.

Evolutionärer Algorithmus Die Grundlagen zum evolutionären Algorithmus werden in [97] eingeführt. Dieser Abschnitt beinhaltet die Notationen und die Parametrisierung des evolutionären Optimierers des 3D-Wasserzeichenalgorithmus.

Der evolutionäre Algorithmus wird mit einer gültigen Lösung $X_k \in X$ initialisiert, wobei X das Set aller gültigen Lösungen ist. Eine spezielle Lösung wird durch das Tupel $X_k = (\vec{x}_k, \vec{\sigma}_k)$ repräsentiert. Der Parameter $\vec{x}_k = (x_{k,1}, ..., x_{k,n})$ beschreibt die Charakteristiken

des Individuums, die den Exponenten e_j entsprechen. Der Parameter $\vec{\sigma}_k = (\sigma_{k,1}, ..., \sigma_{k,n})$ beschreibt die Strategie des Optimierers, wobei n die Schrittweite ist.

Bei einem evolutionären Optimierer liefern die Nachkommen, im Vergleich zu ihren Eltern, nicht unbedingt eine bessere Lösung. Deshalb werden für den folgenden Iterationsschritt beide Populationen $|P_t| = \mu + \lambda$ berücksichtigt, wobei $|P_t|$ die Populationsgröße ist, die über alle Iterationen konstant bleibt. μ ist die Anzahl der Eltern und λ die Anzahl der Nachkommen. Damit kann sichergestellt werden, dass keine Verschlechterung des Ergebnisses eintritt. Diese Strategie wird als Elitismus bezeichnet.

Der Algorithmus wird zu Beginn mit $|P_t| = \mu + \lambda$ zufällig gewählten Individuen initialisiert, die die erste Population P_1 bilden. Die Individuen $x_{k,i}$ genügen einer Gauss-Normalverteilung $\mathcal{N}(\mu_{init}, \sigma^2_{init})$, mit $\mu_{init} = 0$ und σ^2_{init}. Damit sind Werte nahe 0 wahrscheinlicher als solche, die weiter weg liegen. Der Parameter $\vec{\sigma}$ wird mit einer konstanten Schrittweite σ_{step} initialisiert.

Evaluierung der Individuen Zur Evaluierung der Individuen jeder Generation P_t wird die *Fitness Funktion* $f : X \to \mathbb{R}^{\geq 0}$, die auch *Zielfunktion* genannt wird, definiert. Das Resultat dieser Funktion wird als Güte interpretiert und je größer der Wert ist, desto besser eignet sich ein Individuum. Im vorliegenden Szenario evaluiert der Algorithmus, ob und wie gut ein Nachrichtenbit mit den gegebenen Individuen detektierbar ist. In jeder Iteration ist ein Evaluierungsschritt notwendig, um die aktuelle Stärke des Nachrichtenbit ausdrücken zu können.

Abbruchkriterium Der evolutionäre Algorithmus arbeitet mit zwei Abbruchkriterien. Zum einen bricht der Algorithmus ab, wenn eine maximale Anzahl an Iterationen G_{max} erreicht wird und zum anderen, wenn in den letzten Generationen τ_ϵ die Verbesserung der Fitness $\epsilon(t)$ unter dem Schwellwert ϵ_{min} bleibt.

$$\epsilon(t) = \begin{cases} \frac{X^{(t)}_{best} - X^{(t-\tau_\epsilon)}_{best}}{X_{best}}, & \text{wenn} \quad t > \tau_\epsilon \\ \infty, & \text{sonst} \end{cases}$$

Auswahl der Eltern Solange die beiden Abbruchkriterien nicht erreicht sind, werden für jede Generation Elternpaare benötigt. Für die Auswahl der Eltern wird das deterministische Verfahren gewählt. Entsprechend ihrer Fitnesswerte werden die $\mu + \lambda$ Individuen sortiert und die besten μ Individuen als Eltern für die nächste Iteration gewählt. Für den Quotienten λ/μ, der die Optimierungsgeschwindigkeit angibt, wird ein Wert aus dem Intervall $\lambda/\mu \in [4/3, 100]$ gewählt.

Rekombination Der Rekombinationsschritt ist optional, aber im vorliegenden Optimierungsproblem verbessert er die Lösung und wird deshalb durchgeführt. Es werden zwei Individuen gewählt, um ein neues Individuum daraus zu generieren. Für die Rekombination wird eine diskrete und eine Zwischenwertmethode genutzt. Die diskrete Methode wird für die Vererbung der Objektparameter eingesetzt und die Zwischenwertmethode zur Vererbung des Strategieparameters.

Seien X_{P_1} und X_{P_2} die zufällig ausgewählten Eltern aus dem Set und X_{new} ihr Nachkomme. Sei weiterhin r eine gleichverteilte Zufallsvariable aus dem Intervall $[0, 1)$, dann können mittels der diskreten Methode die Objektparameter des Nachkommens $x_{new,i}$ mit gleicher Wahrscheinlichkeit von einem Elternteil $x_{P_1,i}$ oder $x_{P_2,i}$ vererbt werden:

$$x_{new,i} := \begin{cases} x_{P_1,i} & \text{wenn} \quad r_i < 0.5 \\ x_{P_2,i} & \text{sonst} \end{cases} \qquad mit \quad i \in \{1, ..., |\mathbb{V}|\},$$

wobei $|\mathbb{V}|$ die Anzahl der Knoten des Modells sind.

Zum Bestimmen des Strategieparameters σ_{new}, wird mit der Zwischenwertmethode die Strategie der Eltern vererbt. Seien dazu $\sigma_{P_1,i}$ und $\sigma_{P_2,i}$ die Strategieparameter der Eltern und α ein Gewicht aus dem Intervall $\alpha \in [0, 1]$. Dann lassen sich die Strategieparameter $\sigma_{new,i}$ wie folgt berechnen:

$$\sigma_{new,i} := \alpha \cdot \sigma_{P_1,i} + (1 - \alpha) \cdot \sigma_{P_2,i}, \quad \text{mit } i \in \{1, ..., |\mathbb{V}|\}.$$

Der Parameter α wird auf 0.5 gesetzt. Die beschriebenen Rekombinationsschritte werden solange wiederholt, bis λ Nachkommen daraus kreiert wurden.

Mutation Der wichtigste Teil des Optimierers ist die Mutation mit der neuen Population als Ergebnis. Der Algorithmus addiert bei der Mutation eine Zufallszahl zu den Objekt- und Strategieparametern hinzu. Die Zufallszahlen sind normalverteilt $\mathcal{N}(0, \sigma^2)$ mit σ als fester Schrittgröße. Damit sind die Zufallszahlen um null zentriert und extreme Ausreißer unwahrscheinlich. Der Parameter σ kontrolliert die Mutation und beeinflusst damit die Konvergenz der Evolution. Deswegen ist die Wahl für σ sehr vorsichtig zu treffen.

Eine große Schrittweite führt dazu, dass der Algorithmus sich schneller einem Optimum nähert. Wird die Schrittweite zu groß gewählt, führt dies zu einer chaotischen Suche. Wird hingegen der Parameter zu klein gewählt, ist der gegenteilige Effekt zu beobachten und das Risiko zu stagnieren steigt. Eine große Schrittweite ist dann zu wählen, wenn die Lösung weit vom Optimum entfernt liegt. Liegt die Lösung nahe am Optimum, eignet sich eine kleinere Schrittweite besser. Deshalb ist eine adaptive Wahl der Schrittweite sinnvoll.

Der Algorithmus wählt für jeden Objektparameter und jedes Individuum eine adaptive Schrittweite. Damit lässt sich das Erreichen des Optimums feingranular steuern. Denn für Parameter, die vom Optimum weit weg liegen, kann eine große Schrittweite gewählt werden und gleichzeitig für nahe am Optimum liegende Parameter eine kleine Schrittweite. Bei einer großen Anzahl an Objektparametern oder bei einer geringen Population können individuelle Schrittweiten problematisch werden. Für das vorliegende Optimierungsproblem verbessert die individuelle Schrittweitenwahl die Lösung und wird deshalb im Algorithmus mit aufgenommen.

Seien $\tau_1, \tau_2 \in [0, 2]$ die Kontrollparameter der Schrittweite und $\sigma_{k,i}$ die Schrittweite eines Individuums, dann lässt sich die resultierende Schrittweite wie folgt berechnen:

$$\widetilde{\sigma}_{k,i} := \sigma_{k,i} \cdot e^{\tau_1 \cdot r_k} \cdot e^{\tau_2 \cdot s_{k,i}} \quad \textit{mit} \quad r_k \sim \mathcal{N}(0,1) \quad \textit{und} \quad s_{k,i} \sim \mathcal{N}(0,1)$$

Nachdem die Schrittweiten neu berechnet wurden, lassen sich die Objektparameter $x_{k,i}$ wie folgt bestimmen:

$$\widetilde{x}_{k,i} := x_{k,i} + z_{k,i} \quad \textit{mit} \quad z_{k,i} \sim \mathcal{N}(0, \widetilde{\sigma}).$$

Mit der Mutation ist die Optimierung vollständig. Die Population für die nächste Generation P_{t+1} ergibt sich durch die Eltern der Population P_t gemeinsam mit ihren Nachkommen.

Fitnessfunktion Die Fitnessfunktion bestimmt für die Objektparameter $\vec{x}_k$ einen Fitnesswert. Die Fitnessfunktion wertet aus, wie gut ein Wasserzeichenbit nach einem Iterationsschritt eingebettet werden konnte. Der Wertebereich der Fitnessfunktion ist: $F_{\vec{x}_k} \in [0, 1]$, wobei der Wert $F_{\vec{x}_k} = 1$ nur dann erreicht wird, wenn alle Wasserzeichenbits korrekt detektiert werden konnten. Der Wert $F_{\vec{x}_k} = 0$ sagt aus, dass die inverse Wasserzeichennachricht ausgelesen wurde. Der Fitnesswert $F_{\vec{x}_k} \approx 0.5$ sagt aus, dass das Medium als unmarkiert vom Algorithmus angesehen wird.

Sei B die schlüsselabhängige Zuordnung der Bins, $\mathbb{M}$ das originale Drahtgittermodell und $\overline{m} \in \{0, 1\}^{N_{bits}}$ die angestrebte Nachricht, dann ist die Fitnessfunktion wie folgt definiert:

$$F_{\vec{x}_k} := f_B(\mathbb{M}, \overline{m}, \vec{x}_k) \quad \textit{mit} \quad 0 \leq F_{\vec{x}_k} \leq 1$$

Die vollständige Definition der Fitnessfunktion erfolgt mit Gleichung 5.8.

Bestimmen der Exponenten Damit die wasserzeichenmarkierte Kopie des 3D-Modells erzeugt und der evolutionäre Algorithmus evaluiert werden kann, müssen aus den Objektparametern die Exponenten e_i bestimmt werden. Aufgrund der asymmetrischen Natur der

Exponenten ist das “1:1“ Abbilden nicht sinnvoll. Der Wertebereich der Exponenten, die beim Einbetten die Radien vergrößern sollen, ist $e_i \in (1, \infty)$, wohingegen der Wertebereich der Exponenten, die die Radien verringern sollen, $e_i \in (0, 1)$ ist. Aus diesem Grund wird die Funktion ζ, die zur Bestimmung der Exponenten aus den Objektparametern $x_{k,i}$ dient, so gewählt, dass die Exponenten symmetrisch sind:

$$e_i = \zeta(x_{k,i}) := \begin{cases} \sqrt{\pi/2} \cdot \ln(1 + x_{k,i}) + 1 & \text{wenn} \quad x_{k,i} \geq 0 \\ 1/(1 - x_{k,i}) & \text{sonst} \end{cases} \quad \textit{mit} \quad i \in \{1, ..., |\mathbb{V}|\}.$$

Mit der Funktion $\zeta(x_{k,i})$ werden die ursprünglichen Radien $\widehat{r}_i$ modifiziert, um so die Radien zu erhalten $\widehat{r}^* = \widehat{r}_i^{\zeta(x_{k,i})}$.

Gütewert und Fitnessfunktion Mithilfe der Exponenten e_i wird das Wasserzeichen in das originale komprimierte 3D-Modell $\widetilde{M}$ eingebettet. Anschließend wird die daraus entstandene markierte Kopie des 3D-Modells wieder komprimiert. Durch die Komprimierung werde potentiell hinzugekommene hohen Frequenzen gefiltert und das Modell wieder geglättet. Das Wasserzeichen wird aus dem gefilterten Modell detektiert. Dazu sind der Schlüssel und die einzubettende Nachricht notwendig. Es wird geprüft, ob das ausgelesene Wasserzeichen mit dem angestrebten Wasserzeichen $\overline{m_i}$ übereinstimmt. Die Fitnessfunktion vergibt über diesen Vergleich einen Gütewert $\overline{\Delta}$ für jedes Bit:

$$\overline{\Delta}_j := \begin{cases} Var_B(\alpha_j) - Var_B(\beta_j) & \text{wenn} \quad \overline{m}_j = 1 \\ Var_B(\beta_j) - Var_B(\alpha_j) & \text{wenn} \quad \overline{m}_j = 0 \end{cases} \tag{5.5}$$

Die Varianz $Var_B(h_j)$ mit $h_j \in \{\alpha_j, \beta_j\}$ lässt sich folgendermaßen bestimmen:

$$Var_B(h_j) := \frac{1}{||h_j||} \cdot \sum_{i=1}^{N_{bins}} (\delta_{h_j,B_i} \cdot \sum_{k=1}^{|\mathbb{V}|} (\Phi_{i,k} \cdot (r_k^*)^2))$$

$$mit \quad \delta_{h_j,B_i} = \begin{cases} 1 & \text{wenn } B_i = h_j \\ 0 & \text{sonst} \end{cases} \quad \text{und} \quad \Phi_{i,k} = \begin{cases} 1 & \text{wenn } b_i^{min} \leq r_k^* \leq b_i^{max} \\ 0 & \text{sonst} \end{cases}$$

In Gleichung 5.5 ist die Güte $\overline{\Delta}_j$ so definiert, dass sie positiv ist, wenn das detektierte Wasserzeichen mit dem angestrebten Wasserzeichen identisch ist, ansonsten ist sie negativ. Je größer der Gütewert $\overline{\Delta}_j$ ist, desto verlässlicher ist das Wasserzeichenbit detektierbar und umgekehrt.

Das vorliegende Optimierungsproblem berücksichtigt miteinander in Konflikt stehende Kriterien bei der Optimierung. Es gibt daher im Allgemeinen nicht eine Lösung, die für alle Parameter optimal ist, sondern sogenannte *paretooptimale Lösungen*. Vom Set der paretooptimalen Lösungen wird der beste Kompromiss ausgewählt.

Eine Lösung X ist paretooptimal, wenn es keine andere Lösung X' gibt, die besser ist als die Lösung X und zusätzlich kein Kriterium aus X schlechter performt im Vergleich zu den Kriterien in X'. Unter dieser Definition wird der beste Kompromiss aus dem Set der paretooptimalen Lösungen ausgewählt.

Ein gängiger Ansatz, um eine Lösung für solche komplexen und miteinander in Konflikt stehenden Kriterien zu finden, ist das Zusammenfassen der individuellen Kriterien in einer Fitnessfunktion. Es werden die Kriterien additiv zusammengefasst und individuell gewichtet. Wird keine individuelle Gewichtung gewählt, tendiert der Optimierer dazu, die Wasserzeichenbits, die ohnehin eine hohe Güte haben, weiter zu verbessern. Die Gewichtung dient daher der robusten Einbettung der Wasserzeichenbits.

Es wird ein intuitiver Ansatz für die Gewichtung gewählt. Das Wasserzeichenbit mit geringer Güte erhält die höchste Gewichtung, während das mit der höchsten Güte die geringste Gewichtung erhält. Für die dazwischenliegenden Gütewerte wird das Gewicht linear gewählt. Damit erreicht der evolutionäre Algorithmus die besten Fitnesswerte, indem er die schlechtesten Gütewerte der Wasserzeichenbits zuerst verbessert. Verbesserungen von bereits guten Bits werden nur adäquat honoriert, wenn die Möglichkeit, Wasserzeichenbits mit schlechten Gütewerten weiter zu optimieren, nicht mehr gegeben ist.

Sei $\widetilde{\overline{\Delta}}_j$ der angepasste Wert an der Stelle j aus der aufsteigend sortierten Liste der Gütewerte, Δ_{thr} ein geeignet gewählter Schwellwert und seien ω_j die normalisierten Gewichte. Dann lässt sich die auf das Intervall $[0,1]$ normierte Fitness $f(\mathbb{M}, \overline{m}, \vec{x}_k)$ folgendermaßen berechnen:

$$f(\mathbb{M}, \overline{m}, \vec{x}_k) := \frac{1}{2} + \Gamma(-\Delta_{thr}, \overline{\Delta}_{total}, \Delta_{thr}) \frac{1}{2 \cdot \Delta_{thr}} \quad \text{mit} \quad \Gamma(a, x, b) = max(a, min(b, x)), \tag{5.6}$$

$$\overline{\Delta}_{total} := \sum_{j=1}^{N_{bits}} \omega_j \cdot \widetilde{\overline{\Delta}}_j \quad \textit{mit} \quad \omega_q = \begin{cases} 0.5 & \text{wenn} \quad q = 1 \\ 0.5 \cdot \omega_{q-1} & \text{wenn} \quad 1 < q < N_{bits} \\ \omega_{q-1} & \text{wenn} \quad q = N_{bits}, \end{cases} \tag{5.7, 5.8}$$

5.1.4 Rekonstruktion

Nachdem das 3D-Modell in einem Vorverarbeitungsschritt in die Frequenzdarstellung transformiert, das Kernmodell durch das Löschen der hohen Frequenzen errechnet, der Bezugspunkt zur Transformation der Knoten in Kugelkoordinaten errechnet wurde, die Bineinteilung erfolgt ist und das Histogramm durch die Exponenten modifiziert wurde, muss der Algorithmus das Modell zurück in die kartesischen Koordinaten transformieren. Dazu werden die modifizierten Radien wieder in den Bereich $[b_j^{min}, b_j^{max}]$ gebracht, indem in Gleichung 5.4 r_i mit r_i^* und $\widehat{r}_i$ mit $\widehat{r}_i^*$ ersetzt werden. Es ist anzumerken, dass die Radien in diesem Schritt die mittleren und hohen Frequenzen (HFC) schon beinhalten. Zusammen mit dem Zentrum des SEB $\vec{c}$, den Radien r_i^* und den Winkeln ϕ_i und θ_i, kann das Modell in seine kartesischen Koordinaten, mithilfe der folgenden Berechnung, transformiert werden:

$$
\begin{aligned}
x(\vec{v}_i^*) &:= (r_i^* \cdot \sin(\theta_i) \cdot \cos(\phi_i)) + x(\vec{c}) \\
y(\vec{v}_i^*) &:= (r_i^* \cdot \sin(\theta_i) \cdot \cos(\phi_i)) + y(\vec{c}) \\
z(\vec{v}_i^*) &:= (r_i^* \cdot \cos(\theta_i)) + z(\vec{c}),
\end{aligned}
$$

5.1.5 Detektieren des 3D-Modell Wasserzeichens

In diesem Abschnitt wird der Detektionsalgorithmus eingeführt. Der 3D-Modell-Wasserzeichenalgorithmus erhält als Eingabe ein 3D-Modell gemeinsam mit dem Schlüssel und dem Parameterset, das zum Einbetten genutzt wurde. Es handelt sich um ein blindes Verfahren und damit wird das Originale 3D-Modell nicht benötigt. Als Ausgabe liefert der Algorithmus das detektierte Wasserzeichen.

Beim Detektieren des Wasserzeichens aus dem 3D-Modell durchläuft der Algorithmus verschiedene Schritte. Zuerst wird das 3D-Modell mithilfe der Eigenvektoren in den Frequenzraum transformiert. Mithilfe der Gleichung 5.1 werden die hohen Frequenzen herausgefiltert und anschließend mit dem Schwellwert τ das Kernmodell erzeugt. Das Kernmodell wird von den Kartesischen- in die Kugelkoordinaten umgewandelt, nachdem das Zentrum des SEB errechnet wurde. Darauffolgend kann das Modell in Bins, entsprechend 5.1.2, eingeteilt werden.

Die verbleibenden Radien werden in N_{Bin} Bins gleicher Breite eingeteilt. Jeder Radius wird einem Bin zugeordnet und nach Gleichung 5.4 auf das Intervall $[-1, 1]$ normalisiert. Danach werden mithilfe des Schlüssels die Bins den Gruppen α oder β eines Wasserzeichenbits zugeordnet. Zum Extrahieren des Wasserzeichenbits m_i^{detect} wird im letzten Schritt die Varianz der Radien der beiden Gruppen $Var(\alpha_i)$ und $Var(\beta_i)$ berechnet und zueinander

ins Verhältnis gesetzt. Übersteigt das Verhältnis betragsmäßig einen Schwellwert, so kann daraus das Wasserzeichenbit wie folgt interpretiert werden:

$$m_i^{detect} := \begin{cases} 1 & \text{wenn } Var(\alpha_i) - Var(\beta_i) > \tau^{detect} \\ 0 & \text{wenn } Var(\alpha_i) - Var(\beta_i) < -\tau^{detect} \\ ? & \text{sonst} \end{cases}$$

Mit dem Schwellwert τ^{detect} wird sichergestellt, dass kein Wasserzeichen aus unmarkierten 3D-Modellen detektiert wird. Dieser Wert ist empirisch bestimmt worden, indem der beschriebene Detektionsalgorithmus auf unmarkiertem Material mit unterschiedlichen Schlüsseln ausgeführt wurde. Die detektierten Werte ergaben eine Gaussverteilung, von der das 1. Quantil als Schwellwert τ^{detect} genutzt wird. Dadurch wird die Wahrscheinlichkeit stark reduziert, dass bei unmarkiertem Material ein Wasserzeichen detektierbar ist, denn 67% der Wasserzeichenbits sind im Schnitt nicht interpretierbar und daraus lässt sich selbst mit Hilfe von Fehlerkorrekturalgorithmen keine gültige Nachricht erzeugen.

Auch beim Detektionsalgorithmus gilt: Je größer der Abstand der Gruppenvarianzen $Var(\alpha_i)$ zu $Var(\beta_i)$ ist, desto verlässlicher ist das detektierte Wasserzeichenbit. In dieser Arbeit werden keine Vorwärtsfehlerkorrekturverfahren (FEC) oder Cycle Redundancy Checks (CRC) behandelt, da es etablierte Verfahren gibt, die genutzt werden können.

5.2 Evaluierung

In diesem Kapitel werden die Resultate der Evaluierung des 3D-Modell-Wasserzeichenalgorithmus zusammengetragen und diskutiert. Zuerst wird der Aufbau der Testbasis beschrieben und neben der Sicherheit des Wasserzeichens gegen verschiedene Angriffe, werden auch die Performanz und die Nicht-Wahrnehmbarkeit evaluiert und diskutiert. Es wird darauf hingewiesen, dass in dieser Arbeit die Evaluierung der Sicherheit die Evaluierung der Robustheit mit abdeckt. Die Gründe dafür sind in Kapitel 3.2 diskutiert.

5.2.1 Testaufbau

Es wird eine prototypische Implementierung des in Kapitel 5.1 vorgestellten 3D-Modell-Wasserzeichenalgorithmuses evaluiert. Der Algorithmus ist für drei verschiedene Formate ausgelegt, nämlich NIF (NetImmerse File Format), PLY (Polygon File Format) und OBJ (Wavefront Object File Format). Die Implementierung wurde in C++ und Python mithilfe von einigen externen Bibliotheken umgesetzt.

Parameter	Wert	Parameter	Wert	Parameter	Wert
N_{Bin}	16	$\mu + \lambda$	$25 + 75$	τ_ϵ	50
N_{bits}	4	σ_{step}	5	ϵ_{min}	1/1000
τ_1	1	τ_2	0	Z *und* N_{key}	128
T_{IMCS}	3	Δ	2/3	G_{max}	1024

Tabelle 5.1: Überblick der in der Evaluierung verwendeten Parameter

Nicht-Wahrnehmbarkeit	BER
Original	4.11%
bla $n_s = 0.5$	7.30%
bli $n_s = 1.0$	12.00%
blub $n_s = 3.0$	23.00%

Tabelle 5.2: Evaluierung der Nicht-Wahrnehmbarkeit der wasserzeichenmarkierten 3D-Modelle im Vergleich zu ihren originalen 3D-Modellen

Als Testbasis dienen 76 NIF 3D-Modelle des Videospiels FALLOUT III von Bethesda und die bekannten PLY 3D-Modelle des Standford Ropositories, "Bunny", "Happy Buddah" und "Dragon". Die in der Evaluierung des vorgestellten 3D-Wasserzeichenverfahrens verwendeten Parameter sind in Tabelle 5.1 gelistet.

5.2.2 Evaluierung der Nicht-Wahrnehmbarkeit

Die Auswertung der subjektiv wahrgenommenen Wasserzeichenmarkierung wird mit Hilfe der in Kapitel 7 vorgestellten Metrik durchgeführt. Dazu werden die zu den 3D-Modellen gehörenden Texturen im Originalzustand belassen, da es in diesem Kapitel ausschließlich um die Wahrnehmbarkeit der Änderungen durch das 3D-Wasserzeichenverfahren geht. Die Metrik bewertet die Rauheit, Verzerrung und Flächenänderung der 3D-Komponente. Die 3D-Modelle wurden mit zwei zueinander inversen Wasserzeichen markiert und der Wert der Metrik für jedes Modell gemittelt. In Tabelle 5.2 sind die Ergebnisse aufgeführt.

5.2.3 Evaluierung der Sicherheit und Robustheit

In diesem Abschnitt werden verschiedene Nachverarbeitungsschritte evaluiert. Darunter fallen die Kompression, geometrische Transformation, das Vereinfachen des Drahtgittermodells und das Hinzufügen von zufälligem Rauschen.

In den folgenden Paragraphen werden die unterschiedlichen Angriffe durchgeführt und auf der Testbasis evaluiert, die Ergebnisse präsentiert und diskutiert. Es wurden zwei markierte Kopien jedes 3D-Modells erzeugt, wobei die eine Kopie das inverse Wasserzeichen der anderen Kopie trägt. Das Einbetten beider Bits an jeder Stelle soll die Unabhängigkeit des Einbettungserfolges vom Wasserzeichen zeigen. Zur Beurteilung des Einbettungserfolges wird die Bitfehlerrate (BER) berechnet. Dazu wird die Hammingdistanz $H(m_i, m_i^*)$ zwischen dem einzubettenden Wasserzeichen m_i mit dem detektierten Wasserzeichen m_i^* berechnet, sodass sich die Bitfehlerrate wie folgt berechnen lässt: $BER = \frac{1}{n}\sum_{i=1}^{n} H(m_i, m_i^*)$.

Geometrische Transformationen Geometrische Transformationen sind einfach anzuwendende Angriffe, mit dem Ziel das Wasserzeichen zu zerstören. Als Vertreter der unterschiedlichen geometrischen Transformationen werden hier die Translation, Rotation und Skalierung herausgegriffen, da sie einfach zu implementieren, aber gleichzeitig sehr effektiv sind. Aus diesem Grund muss auch der Wasserzeichenalgorithmus diesen Angriffen standhalten. Zur Evaluierung werden die beiden wasserzeichenmarkierten Kopien der 3D-Modelle herangezogen und nach jedem der unterschiedlichen Angriffe die BER zwischen dem detektierten und eingebetteten Wasserzeichen berechnet. Die Parameter für die unterschiedlichen Angriffe werden zufällig gewählt.

Eine geometrische Transformation ist wie folgt definiert:

$$(x(\vec{v}_i^{'})), y(\vec{v}_i^{'}), z(\vec{v}_i^{'}), \Xi)^T = (x(\vec{v}_i), y(\vec{v}_i), z(\vec{v}_i), 1)^T \cdot T_{\{T,S,R_1,R_2,TSR_2\}} \quad \forall v_i \in \mathbb{V},$$

wobei $T_{\{T,S,R_1,R_2,TSR_2\}}$ die Transformationsmatrizen sind mit der homogenen Translation T_T, homogenen Skalierung T_S, homogenen Rotation $T_{R_{\{1,2\}}}$ und einer Kombination aus allen drei Transformationen T_{TSR_2}, indem sie hintereinander ausgeführt werden:

$$\begin{aligned}
T_T &= Trans(r_1 \cdot s_1 \cdot c_1 \cdot R(M), r_2 \cdot s_2 \cdot c_1 \cdot R(M), r_3 \cdot s_3 \cdot c_1 \cdot R(M)) \\
T_S &= \begin{cases} Scal(1/2 + s_1, 1/2 + s_1, 1/2 + s_1) & \text{if } s_1 \leq 1/2 \\ Scal(2 + s_1, 2 + s_1, 2 + s_1) & \text{if } s_1 > 1/2 \end{cases} \\
T_{R_1} &= Rot(r_1 \cdot s_1 \cdot \pi, 0, 0) \\
T_{R_2} &= Rot(r_1 \cdot s_1 \cdot \pi, r_2 \cdot s_2 \cdot \pi, r_3 \cdot s_3 \cdot \pi) \\
T_{TSR_2} &= T_T \cdot T_S \cdot T_{R_2},
\end{aligned}$$

wobei $Trans(\delta_x, \delta_y, \delta_z)$ das Modell entlang der Achsen um die Werte δ_x, δ_y und δ_z verschiebt. Mit $Scal(\lambda_x, \lambda_y, \lambda_z)$ ist eine Skalierung entlang der Achsen um den Faktor λ und mit $Rot(\alpha, \beta, \gamma)$ ist die Rotation des 3D-Modells entsprechend der drei Winkel α, β und γ

	Transformation	BER
N	Keine Transformation	4.11%
T	Zufällige Translation	1.97%
S	Zufälliges aber gleichmäßiges Skalieren	2.63%
R_1	Zufällige Rotation an einer Achse	6.09%
R_2	Zufällige Rotation an drei Achsen	1.64%
TSR_2	Kombination von T,S und R_2	2.30%

Tabelle 5.3: Evaluationsergebnisse nach Angriffen der Kategorie geometrischer Transformationen

gemeint. Die Zufallsvariablen $r_{\{1,2,3\}} \in \{-1, 1\}$ und $s_{\{1,2,3\}} \in [0, 1]$ sind gleichverteilt. Die Konstante wurde mit $c_1 = \frac{1}{2}$ gewählt. R(M) spiegelt die mittlere Größe oder Ausdehnung des 3D-Modells M in den x, y, z Dimensionen wieder.

Die Ergebnisse der 152 markierten und angegriffenen Modelle sind in Tabelle 5.3 gelistet. Der Tabelle kann entnommen werden, dass die BER bei 6% liegt und damit in etwa 94% der Bits korrekt detektiert wurden. Damit kann die Aussage getroffen werden, dass der vorgestellte Algorithmus invariant in Bezug auf geometrische Transformationen ist. Das Verschieben oder Rotieren der Knoten ändert die Koordinaten bis auf Rundungsfehler nicht. Ebenso bleibt die Position des Zentrums des SEB stabil. Die Sicherheit gegen Rotation wird erreicht, da auf einem normalisierten radialen Abstand der Knoten zum SEB eingebettet wird.

Die etwas widersprüchliche Beobachtung, dass die BER nach den meisten der Angriffe besser ist im Vergleich zu den nicht angegriffenen 3D-Modellen, ist einigen schwach eingebetteten Wasserzeichenbits geschuldet. Durch Rundungsfehler ist bei einigen Wasserzeichenbits das Verhältnis der Gruppenvarianz zufällig in die richtige Richtung verschoben worden. An der Stelle muss gesagt werden, dass sich die BER durch die schwach eingebetteten Wasserzeichenbits in den meisten Fällen vergrößert.

Ungleichmäßiges Skalieren Neben den geometrischen Transformationen wird auch das ungleichmäßige Skalieren als Angriffsszenario evaluiert. Es ist definiert als:

$$T = Scal(f(s_1), f(s_2), f(s_3)) \quad \text{mit} \quad s_{\{1,2,3\}} \in [0, 1] \quad f(x) = \begin{cases} 1/2 + x & \text{wenn } x \leq 1/2 \\ 2x & \text{wenn } x > 1/2 \end{cases}$$

Es wurde entlang der drei Hauptachsen um die Faktoren $0.6, 0.93$ und 1.37 skaliert. Die BER beläuft sich auf 52.14% und zeigt, dass das Wasserzeichen komplett unlesbar

wurde. Die Proportionen des Modells wurden ungleichmäßig skaliert und damit wurde der radiale Abstand der Knoten zum SEB ungleichmäßig verändert. Sehr viele Radien werden so zu anderen Bins verschoben und die Varianz in jedem Bin ändert sich mit. Das Wiederherstellen des ursprünglich eingebetteten Wasserzeichens ist nach einem ungleichmäßigen Skalieren unmöglich.

Zufälliges Rauschen In diesem Abschnitt wird die BER nach dem Addieren von zufälligem Rauschen evaluiert. Zufälliges Rauschen taucht typischerweise bei der Übertragung von analogen Daten auf oder bei Skalierung und Kompression. Es wird jeder Knoten um einen zufälligen Wert manipuliert. Dabei sind die Zufallswerte unabhängig voneinander.

Seien $\vec{v}_i$ die Knoten, n_s die Stärke des zufälligen Rauschens und ζ die mittlere Boxgröße, die ein Modell umgibt. Dann lässt sich der zufällige Rauschangriff wie folgt definieren:

$$\vec{v}_i^{\,\prime} := \vec{v}_i + \vec{o}_i \quad \text{mit } \vec{o}_i = (o_{i_x}, o_{i_y}, o_{i_z}) \quad \text{und } o_{i_{\{x,y,z\}}} \in [-\frac{n_s \cdot \zeta}{100}, \frac{n_s \cdot \zeta}{100}] \text{ zufällig gewählt}$$

In Tabelle 5.4 sind die Ergebnisse des Angriffs gelistet. Der Parameter zur Steuerung der Stärke des zufälligen Rauschens wurde mit $n_s = \{0.5, 1.0, 3.0\}$ gewählt. Die Angaben in der Tabelle sind die Mittelwerte der BER über 152 3D-Modelle. Wie der Tabelle zu entnehmen ist, resultiert das Addieren von zufälligem Rauschen mit der Stärke $n_s = 0.5$ in einer akzeptablen BER von 7.30%. Erhöht man die Stärke des Rauschens auf ein mittleres Niveau $n_s = 1.0$, so steigt die BER auf 12%. Bei einer Stärke von $n_s = 3.0$ steigt die BER auf 23%. Bei beiden Stärken $n_s = 1.0$ und $n_s = 3.0$ sind Änderungen und Artefakte nach dem Angriff am 3D-Modell sichtbar.

Der Algorithmus ist invariant gegenüber hinzugefügtem Rauschen, da die spektrale Kompression genutzt wird. Das Addieren von zufälligem Rauschen ändert die Radien der Knoten und verschiebt damit die Knoten zu benachbarten Bins oder manipuliert die Varianz innerhalb der Bins. Je höher die Stärke des Rauschens, desto stärker tritt der Effekt auf. Die Topologie des 3D-Modells wird dadurch jedoch nicht zerstört und somit bleibt das Wasserzeichen auslesbar.

Mit dem Parameter τ aus Abschnitt 5.1.1 lässt sich die Robustheit des Wasserzeichens gegen zufälliges Rauschen steuern. Es kann ein gröberes Kernmodell verwendet werden zur Steigerung der Robustheit. Der Parameter wurde für die Tests mit $\tau = 128$ gewählt.

Vereinfachen des Drahtgittermodells Das Vereinfachen des Drahtgittermodells kann ein typischer Nachverarbeitungsschritt sein, um dadurch Rechenzeit oder Speicherplatz bei großen 3D-Modellen zu sparen. Der Hauptunterschied zwischen der Vereinfachung und Nachverarbeitungsschritten aus den vorhergehenden Abschnitten ist, dass dieser die

Hinzufügen von zufälligem Rauschen	BER
Original	4.11%
$n_s = 0.5$	7.30%
$n_s = 1.0$	12.00%
$n_s = 3.0$	23.00%

Tabelle 5.4: Evaluationsergebnisse nach dem Addieren von zufälligem Rauschen

Topologie verändert. Durch Verringern der Anzahl der Knoten und Kanten wird das Modell vereinfacht, wobei gleichzeitig die Nicht-Wahrnehmbarkeit dieser Modifikation möglichst hoch sein soll. Das kann durch Minimieren der Artefakte, die durch die Vereinfachung entstehen, erreicht werden. Es existieren verschiedene Ansätze, um ein Drahtgittermodell zu vereinfachen, die alle das Ziel verfolgen, die Artefakte zu minimieren. Die Ansätze messen auf unterschiedliche Art und Weise die Artefakte. Hier wird ein einfacher Algorithmus zur Evaluierung verwendet, der von Trick [97] eingeführt wurde.

Der Algorithmus wählt ein Paar von benachbarten Knoten (v_1, v_2), die eine minimale Distanz d zueinander haben und verrechnet sie zu einem Knoten v_{new}. Der neue Knoten ersetzt die beiden bestehenden Knoten und die Kanten werden entsprechend aktualisiert. Dazu wird die Kante, die beide Knoten verbindet, gelöscht und alle Kanten, die mit v_1 oder v_2 verbunden waren, mit v_{new} verbunden. Das Verrechnen zweier Knoten wird solange durchgeführt, bis ein vorgegebener Faktor m erreicht wurde.

Zur Evaluierung wird der Faktor m variiert und die mittlere BER zum entsprechenden Faktor errechnet. Die Ergebnisse sind in Tabelle 5.5 gelistet. Nachdem 1% der Knoten miteinander verrechnet wurden, wächst die BER von 4.11% auf 12.5%. Werden 5% der Knoten miteinander verrechnet, dann wächst die BER auf 28.5% an. Die Nicht-Wahrnehmbarkeit der Wasserzeichenmarkierung ist nach dem Vereinfachungsangriff mit beiden Faktoren relativ gut, sodass nahezu keine Artefakte bei den 3D-Modellen sichtbar sind. Die Sichtbarkeit der Artefakte ist jedoch stark abhängig von der Struktur der 3D-Modelle.

Die Vereinfachungen des Drahtgittermodells schwächen das Wasserzeichen signifikant mit steigendem m. Als Hauptgrund dafür wird die Manipulation der Topologie durch den Angriff gesehen. Der Angriff reduziert die Anzahl der Knoten in glatten Regionen. Damit wird die genutzte Eigenschaft beim Einbetten des Wasserzeichens modifiziert.

Ein weiterer Grund, der verglichen zum vorher genannten einen nicht so starken Einfluss auf die BER hat, ist, dass durch den Angriff die Knoten verschoben werden und damit die Bins und die Varianz der Bins manipuliert wird.

Vereinfachungsangriff	BER
Original	4.11%
$m = 1.0$	12.50%
$m = 5.0$	28.50%

Tabelle 5.5: Evaluationsergebnisse nach dem Vereinfachungsangriff mit unterschiedlichen Faktoren

Wird das Kernmodell nicht mit einem festen Schwellwert $\tau = 128$, sondern einem variablen Schwellwert erstellt, kann das zur Invarianz des Verfahrens gegenüber der Vereinfachung führen.

5.2.4 Performanz Analyse

In diesem Abschnitt werden die Ergebnisse der Performanzevaluation diskutiert. Es werden anhand eines ausgewählten 3D-Modells die Zeitmessungen für die unterschiedlichen Schritte des Algorithmus angegeben.

Zur detaillierteren Performanzanalyse wird das Modell "Giant Ant"gewählt. Es hat 3076 Knoten und 5966 Dreiecke. Die Analyse beinhaltet Zeitmessungen für den Embedder und Detektor. Bei der Evaluierung wurde die Anzahl der Generationen des evolutionären Algorithmus auf $G_{max} = 16$ festgesetzt, anstatt $G_{max} = 1024$. Damit benötigte der Embedder 298.4 Sekunden und der Detektor 18.6 Sekunden. In Abbildung 5.6 wird ein Überblick zu den prozentualen Laufzeiten der einzelnen Teile des Einbettungsalgorithmus gegeben.

Abbildung 5.6 zeigt, dass 92% der CPU-Zeit auf den Optimierer entfallen, wobei nur 4% der Zeit für das Bestimmen der Eigenvektoren des 3D-Modells entfällt. Die verbleibende Zeit entfällt auf das Importieren, Exportieren und weitere Berechnungen.

Im Gegensatz zum Embedder braucht der Detektor 70% der Zeit für die Berechnung der Eigenvektoren, wie der Abbildung 5.7 zu entnehmen ist. Absolut gesehen brauchen der Embedder und der Detektor die gleiche Zeit für die Berechnung der Eigenvektoren. Im Detektor entfällt der Optimierer und das Exportieren des 3D-Modells. Deshalb ergibt sich eine andere Aufteilung der Rechenzeit und die verbleibenden Schritte fallen unterschiedlich ins Gewicht.

Abbildung 5.8 gibt einen tieferen Einblick in den Embedder. Dafür wird die Zeit, die der Embedder im Optimierer benötigt, weiter analysiert. Die Hälfte der Zeit entfällt auf die Berechnung des Zentrums des SEBs. Weitere 40% der Rechenzeit werden von der Komprimierung und anschließenden Reparatur in Anspruch genommen. Das Berechnen

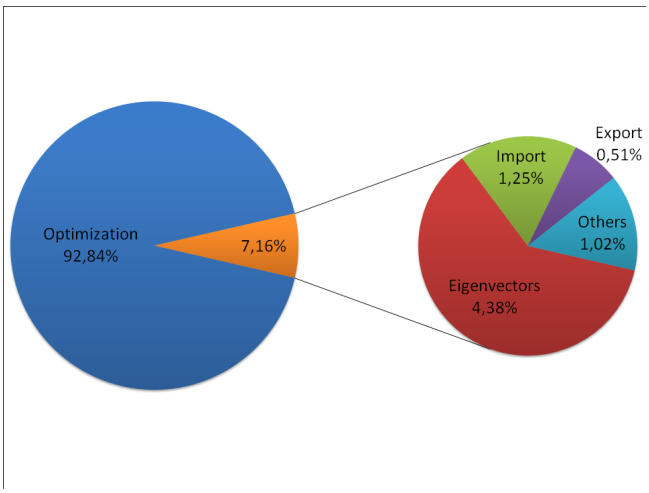


Abbildung 5.6: Performanzanalyse des 3D-Modell Wasserzeichenembedders

des SEB Zentrums, die Kompression und der Reparaturschritt müssen für jedes Individuum in jeder Generation neu berechnet werden. Die Detektion wird nach jeder Iteration des Embedders ausgeführt und benötigt in etwa 10% der Zeit.

Die Performanzanalyse zeigt auf, dass die Implementierung noch Optimierungsbedarf hat, um praktikabel im Aufwand zu sein. Dabei ist das Berechnen des SEB-Zentrums sowie das Komprimieren und Reparieren des Modells am aufwendigsten. Der Algorithmus bietet sehr viel Möglichkeit zum Parallelisieren von Prozessen. Außerdem können einige Rechenschritte auf die GPU portiert werden.

5.3 Zusammenfassung

3D-Modell-Wasserzeichen sind ein relativ neues Feld im Vergleich zu anderen Multimediawasserzeichen, z.B. für Audio, Bild oder Video. Das entwickelte Verfahren zum Markieren von 3D-Modellen mit einem Wasserzeichen genügt den Anforderungen aus Kapitel 3.2. Die Evaluierungsergebnisse zeigen eine gute Robustheit, wobei die Vereinfachung des Drahtgittermodells und das ungleichmäßige Skalieren Herausforderungen darstellen, aber die Angriffe wirken sich auch sehr stark auf die visuelle Wahrnehmbarkeit aus. Es wurden Vorschläge diskutiert, die weiter verfolgt werden können, für den Fall, dass die beiden Angriffe in Anwendungsszenarien eine Rolle spielen.

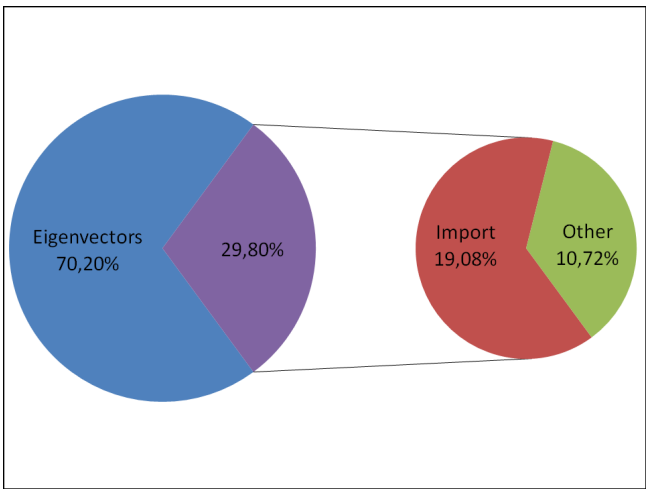


Abbildung 5.7: Performanzanalyse des 3D-Modell Wasserzeichendetektors

Der vorgestellte Algorithmus modifiziert die Knotenverteilung eines Kernmodells zur Einbettung des Wasserzeichens. Das Kernmodell erhält der Algorithmus durch eine spektrale Kompression. Ein evolutionärer Optimierer übernimmt die iterative Einbettung, um das bestmögliche Ergebnis zu erreichen. Die Wahl der Parameter kann nach Bedarf angepasst werden.

Der Algorithmus kann in der Performanz optimiert werden. Dazu können viele Komponenten des Algorithmus parallelisiert und auf die GPU portiert werden. Bei diesem Optimierungsschritt sollte mit den zeitintensivsten Komponenten des Algorithmus begonnen werden. Das Verfahren kann leicht so modifiziert werden, dass für jedes Modell jedes Wasserzeichenbit nur einmal vorberechnet werden müsste. Beim Erzeugen einer Wasserzeichenmarkierten Kopie wird dann das entsprechende Set an Knoten kopiert.

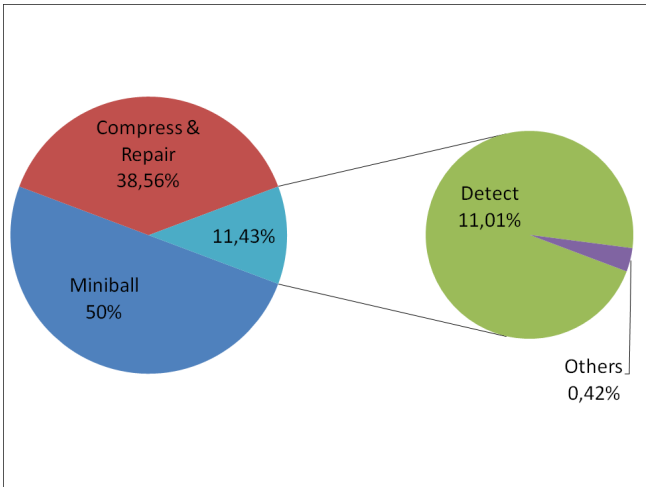


Abbildung 5.8: Tiefgreifendere Performanzanalyse des 3D-Modell Wasserzeichenembedders, die den Rechenaufwand des Optimierers weiter unterteilt

6 Texturwasserzeichen

Texturen sind die Hülle der 3D-Modelle. Es gibt verschiedene Dateiformate für die Texturen. Im Folgenden wird ein Wasserzeichenalgorithmus für Texturen vorgestellt, der für das von Microsoft entwickelte Direct Draw Surface (DDS) Format geeignet ist. Dieses Format zeichnet sich durch die Vielfalt an bereitgestellten Eigenschaften aus, wie z.B. die effektive Kompression und die hohe Verarbeitungsgeschwindigkeit. Das angezeigte Bild wird durch Texturen, die auf die entsprechenden 3D-Modelle gelegt werden, dargestellt. Bei diesem Prozess wird vom Texturieren der 3D-Modelle gesprochen.

Typischerweise existieren Texturen für 3D-Modelle. Im Folgenden nutzen wir das DDS-Format, das sich vor allem im Bereich von Videospielen und im Rendering durchgesetzt hat. Es wurde von Microsoft entwickelt und basiert auf der S3 Texturkompression S3TC, die auch als DXTn [23, 44] bekannt ist. Die Anforderungen an die Wasserzeichenverfahren aus Kapitel 3.3 können mit gängigen Bildwasserzeichenverfahren nicht erfüllt werden. Die Gründe dafür liegen an der Kompression, die beim DDS-Format genutzt wird und den stark variierenden Größen der Texturen.

Ein Texturwasserzeichen in Kombination mit einem 3D-Modell-Wasserzeichen ergibt für das Gesamtmodell eine höre Datenkapazität. Die Robustheit der Wasserzeichen und Sicherheit gegen mögliche Angriffe können ebenfalls gesteigert werden. Ein neuer Angriff im Kontext von Texturen im DDS-Format ist ein Mip-Map-Angriff. Beim Mip-Map Angriff skaliert der Angreifer die Textur von einem niedrigen Level hoch und löscht damit die Wasserzeicheninformation, die ursprünglich in dem gelöschten Level eingebettet war.

Der Kernalgorithmus wurde in [64] publiziert und in den betreuten Bachelorarbeiten [62] und [22] weiterentwickelt. Die Anforderungen aus Kapitel 3.3 wurden bei der Konzeptionierung berücksichtigt. Es wird bei den Texturen nicht auf ein bereits bestehendes Wasserezichenverfahren zurückgegriffen, da im DDS-Format ein spezielles Kompressionsverfahren genutzt wird und darauf die Einbettung angepasst wurde, sodass die Performance, Robustheit und Sicherheit optimiert werden konnten.

6.1 DDS Texturwasserzeichenalgorithmus

Der Wasserzeichenalgorithmus für Texturen, der auch in [64] vorgestellt wurde, bettet das Wasserzeichen direkt in die komprimierte DDS Textur ein. Dazu werden die Grundlagen des DDS Formats vorausgesetzt [62]. Das Einbetten der Wasserzeichennachricht erreicht der Algorithmus durch Flippen von Indizes innerhalb der Farbpalette eines Blockes. Dazu kann jeder Block verwendet werden und somit können alle Texturen genutzt werden, auch solche, die aus einem Block bestehen. Als Basis für das Einbetten dient das Patchworkverfahren aus [7]. Dazu werden zuerst zwei gleich große Gruppen aus zufällig gewählten Blöcken generiert. Beim Einbetten der Wasserzeichennachricht mittels des Patchworkverfahrens wird ein Ungleichgewicht zwischen der Anzahl der Indizes in diesen beiden Gruppen erzeugt. Dadurch wird das Einbetten der “0“ oder “1“ erreicht.

6.1.1 Grundlagen des Texturwasserzeichenalgorithmus

Die DDS Texturen sind in Blöcke aus $4x4$ Pixel B_j gruppiert, wobei $j \in \{1, ..., B_G\}$. Der Wasserzeichenalgorithmus wählt einige Blöcke schlüsselabhängig aus und ordnet sie einer der beiden Gruppen α_i oder β_i zu, um ein Nachrichtenbit m_i einbetten zu können, wobei $i \in \{1, ..., N_{bits}\}$. Die Anzahl der Möglichkeiten, $\mathbb{B}$ Blöcke den Gruppen zuzuordnen, kann wie folgt errechnet werden. Sei N_{bits} die Anzahl an Nachrichtenbits, die in die Texturen eingebettet werden sollen und $|B|$ die Anzahl an DDS Blöcken des gesamten Videospiels, dann gilt:

$$\mathbb{B} = \prod_{k=1}^{2 \cdot N_{bits}} \binom{|B| - (k-1)\frac{|B|}{2 \cdot N_{bits}}}{\frac{|B|}{2 \cdot N_{bits}}}.$$

Beim Einbetten der Wasserzeichennachricht werden die Relationen zweier Gruppen α_i und β_i entsprechend der Ungleichung 6.1 und 6.3 modifiziert. Der Algorithmus zählt dafür die Häufigkeit der Indizes 0 und 3 in den Blöcken B_j, die jeweils eine der beiden Gruppen α_i oder β_i zugeordnet ist und speichert diese entsprechend als γ_{α_i} und γ_{β_i}. Es werden nur solche Blöcke berücksichtigt, die mindestens zwei Farbwerte und damit auch mindestens zwei Indizes besitzen. Die Variablen $count_{\alpha_i}$ und $count_{\beta_i}$ repräsentieren die Anzahl der Indizes 0 und 3 der jeweiligen Gruppen α_i und β_i. Sei m_i das einzubettende Wasserzeichenbit, w_{int} die Wasserzeichenintensität und damit das mindestens zu erreichende Verhältnis der beiden Gruppen γ_{α_i} und γ_{β_i} zueinander, dann gilt:

$$\log_{10}(\frac{\gamma_{\alpha_i}}{\gamma_{\beta_i}}) > w_{int} \quad \text{wenn } m_i = 1 \quad und \quad \log_{10}(\frac{\gamma_{\alpha_i}}{\gamma_{\beta_i}}) < -w_{int} \quad \text{wenn } m_i = 0. \tag{6.1}$$

Je größer w_{int} gewählt wird, desto stärker ist das Wasserzeichen eingebettet. Gleichzeitig verringert sich damit aber auch die Transparenz. Je kleiner w_{int} gewählt wird, desto geringer ist die Sicherheit, aber gleichzeitig wird das Wasserzeichen transparenter. Die Wasserzeichenintensität kann im Bereich von $w_{int} \in (0, 1)$ gewählt werden. Es ist empfehlenswert, nicht weniger als $B_G = 32$ Blöcke pro Gruppe für den Wasserzeichenalgorithmus zu verwenden. Die Variablen γ_{α_i} und γ_{β_i} sind wie folgt definiert:

$$\gamma_{x_i} := \sum_{j=1}^{|B_G|} count_{x_j} \quad mit \quad count_{x_j} = \begin{cases} |\{0,3\}| & \text{wenn mehr als zwei Indizes in } B_j \\ 0 & \text{sonst} \end{cases} \tag{6.2}$$

Damit die Ungleichungen (6.1) erfüllt sind, muss nach dem Einbetten für jedes Bit gelten:

$$\begin{aligned} \widetilde{\gamma_{\alpha_i}} &\geq (1 + w_{int}) \cdot \frac{\gamma_{\alpha_i} + \gamma_{\beta_i}}{2} \quad \text{und} \quad \widetilde{\gamma_{\beta_i}} \leq (1 - w_{int}) \cdot \frac{\gamma_{\alpha_i} + \gamma_{\beta_i}}{2} \quad wennm_i = 1 \\ \widetilde{\gamma_{\alpha_i}} &\leq (1 - w_{int}) \cdot \frac{\gamma_{\alpha_i} + \gamma_{\beta_i}}{2} \quad \text{und} \quad \widetilde{\gamma_{\beta_i}} \geq (1 + w_{int}) \cdot \frac{\gamma_{\alpha_i} + \gamma_{\beta_i}}{2} \quad wennm_i = 0, \end{aligned} \tag{6.3}$$

6.1.2 Blockauswahl

Nachdem festgelegt ist, wie die beiden Gruppen γ_{α_i} und γ_{β_i} in Abhängigkeit der Nachrichtenbit zu modifizieren sind, um das Wasserzeichen einzubetten, geht es in diesem Abschnitt um die Auswahl der Blöcke. In den nächsten Abschnitten wird darauf aufbauend die Auswahl der zu flippenden Pixelindizes beschrieben. Zuerst werden jedoch geeignete Blöcke B_j gesucht und dann wird in diesen Blöcken nach flipbaren Pixel gesucht.

Der Algorithmus wählt Blöcke mit hoher Inhomogenität aus. Damit lässt sich gewährleisten, dass die Transparenz möglichst hoch ist. Inhomogene Regionen sind Bereiche, in denen große Unterschiede unter benachbarten Pixel eines Blockes vorliegen. Um die inhomogenen Regionen zu finden, errechnet der Algorithmus einen Variationswert. Der Variationswert σ_{V_l} repräsentiert die Varianz des Histogramms der Pixelverteilung und wird nach Gleichung (6.4) berechnet. Dieser Wert wird für alle Blöcke, die mindestens drei unterschiedliche Indizes haben, berechnet.

Sei $|\{u\}|$ die Häufigkeit des entsprechenden Indizes u in dem Block B_j, dann lassen sich sowohl der Mittelwert μ_{V_l} als auch der Variationswert σ_{V_l} wie folgt berechnen:

$$\mu_{V_l} := \frac{1}{4} \sum_{u=1}^{4} |\{u\}|, \quad \sigma_{V_l} := \frac{1}{4} \sum_{u=1}^{4} (|\{u\}| - \mu_{V_l})^2. \tag{6.4}$$

Anschließend sortiert der Algorithmus die Blöcke B_j in aufsteigender Reihenfolge, in Abhängigkeit des Variationswertes σ_{V_l}. Je geringer der Variationswert, desto weniger unterscheiden sich die Indizes in einem Block und desto mehr gleichen sich die Häufigkeiten der unterschiedlichen Indizes. Damit entsprechen kleine Variationswerte inhomogenen Blöcken und große Variationswerte homogenen Blöcken.

6.1.3 Einbettungsregeln

Das Flippen der Pixel wird entsprechend den Einbettungsregeln vorgenommen, die in diesem Abschnitt eingeführt werden. Die Einbettungsregeln sind an die Blockkompression angelehnt. Nachdem der Algorithmus die zur Einbettung geeigneten Pixel ausgewählt hat, flippt er den Index des Pixels zu einem seiner Nachbarindizes. Dadurch wird erreicht, dass die Transparenz so hoch wie möglich gehalten werden kann. Abbildung 6.1 zeigt die Anordnung der Indizes nach ihren Helligkeitswerten und auch die Möglichkeiten, in die ein Index geflippt werden darf. Index 0 repräsentiert den hellsten Wert des Blockes, Index 2 den Zweithellsten, Index 3 den Dritthellsten und Index 1 den dunkelsten Wert des Blockes.

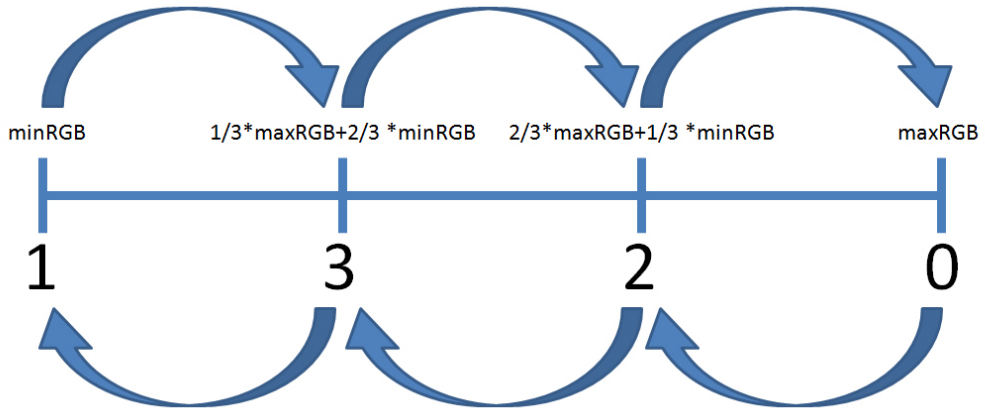


Abbildung 6.1: Überblick der Indexsequenz entsprechend der Helligkeitswerte gemeinsam mit den Möglichkeiten, die Indizes zu flippen.

Nach Abbildung 6.1 können Pixel mit dem Index 1 zum Index 3 und Pixel mit dem Index 0 zum Index 2 geflippt werden. Der Index 2 kann nach Bedarf zum Index 3 oder 0 geflippt werden und das Pixel mit dem Index 3 kann zum Index 1 oder 2 geflippt

werden. Mit diesem Ansatz werden die kleinstmöglichen Änderungen zugelassen, um die höchstmögliche Transparenz zu erreichen.

6.1.4 Pixelauswahl und Flippen der Pixel

Bei der Auswahl der zum Flippen geeigneten Pixel soll eine möglichst hohe Robustheit des Wasserzeichens erreicht werden, bei gleichzeitig hoher Transparenz. Die Pixelauswahl hängt vom einzubettenden Nachrichtenbit m_i und der entsprechenden Gruppen γ_{α_i} oder γ_{β_i} ab. Das Verhältnis der Gruppen $log(\frac{\gamma_{\alpha_i}}{\gamma_{\beta_i}})$ muss positiv sein, wenn das einzubettende Wasserzeichennachrichtenbit $m_i = 1$ ist. Damit müssen die Indizes 0 und 3 in den Blöcken der Gruppe γ_{α_i} erhöht und gleichzeitig in der Gruppe γ_{β_i} verringert werden. Hierfür werden die Pixel mit den Indizes 1 und 2 der Gruppe γ_{α_i} und Pixel mit den Indizes 0 und 3 der Gruppe γ_{β_i} ausgewählt.

Das Einbetten der Wasserzeichennachricht geschieht durch das Flippen der Indizes der ausgewählten Pixel. Für jedes Pixel wird ein Bereich, bestehend aus seinen Nachbarn, ausgewählt, wie in Abbildung 6.2 dargestellt. Das dort schwarz markierte Pixel ist das potenziell zu flippende Pixel und die grauen Pixel sind als dessen Nachbarn gekennzeichnet. Für jedes der 16 Pixel wird ein Variationswert σ_{V_s} nach Gleichung (6.4) errechnet. Das Pixel, dessen zugehöriger Variationswert σ_{V_s} am größten ist, wird ausgewählt und für jede der beiden Möglichkeiten, in die geflippt werden kann, wird der Variationswert σ_{V_s} erneut berechnet. Damit das Wasserzeichen robust gegen z.B. Bildkompression eingebettet werden kann, flippt der Algorithmus den Index des Pixels nur dann, wenn der Variationswert anschließend höher als der bisherige ist. Ein höherer Variationswert entspricht einem homogeneren Bereich.

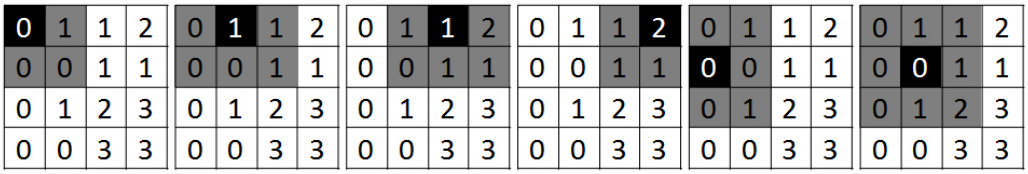


Abbildung 6.2: Übersicht, wie die Bereiche für jedes der 16 Pixel gewählt werden, wobei das schwarze Pixel das potentiell zu flippende Pixel ist. Seine Nachbarn sind grau eingefärbt.

Die beschriebene Vorgehensweise entspricht der Arbeitsweise der Kompressionsalgorithmen, die Bilder homogenisieren. Gleichzeitig besteht die Gefahr, dass homogenere Flächen bevorzugt weiter homogenisiert werden bei einem Kompressionschritt und das

Wasserzeichen schwächen würden. Angenommen, das aktuell einzubettende Nachrichtenbit ist $m_i = 1$ und der ausgewählte Block B_j ist der Gruppe γ_{α_i} zugeordnet. Dann soll der Algorithmus die Anzahl der Pixel mit den Indizes 0 und 3 erhöhen. Dafür wählt der Algorithmus die Pixel mit den Indizes 1 und 2 und errechnet für jeden dieser Indizes den Variationswert. Abbildung 6.3 zeigt das Verfahren des Algorithmus. In dem dort gewählten Beispiel ist das schwarz markierte Pixel das einzige, das geflippt wird, denn der Variationswert bleibt gleich oder sinkt bei all den anderen Möglichkeiten. Außerdem ist der in Abbildung 6.3 betrachtete Bereich der mit dem kleinsten Variationswert und damit der zuletzt betrachtete.

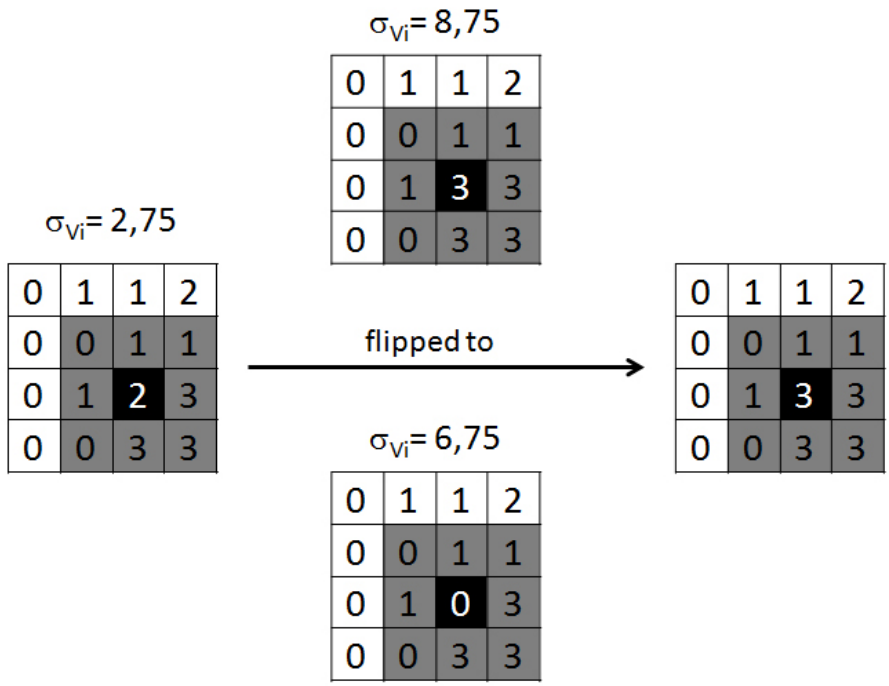


Abbildung 6.3: Flippen der Indizes basierend auf den Variationswerten der beiden Möglichkeiten, in die der Algorithmus flippen darf.

6.1.5 Blockaktualisierung

Sobald beim Einbetten der letzte vorhandene Pixel Index '1' oder '0' eines Blockes geflippt wird, führt der Algorithmus die Blockaktualisierung durch. Denn es ist notwendig, dass

jeder Block einen minimalen und maximalen Farbwert hat. Eine Ausnahme stellen die Blöcke dar, bei denen alle Pixel den gleichen Farbwert oder zwei bzw. drei benachbarte Farbwerte haben. Diese bleiben jedoch vom Einbettungsalgorithmus unberücksichtigt. Würde die Blockaktualisierung nicht durchgeführt werden, so würde durch ein wiederholtes Komprimieren der Textur die Einbettung zerstört werden.

Der Algorithmus unterscheidet zwischen zwei Fällen:

1. Es existiert kein Pixel mehr mit dem Index '0': Das heißt, dass kein Pixel im aktuellen Block vorhanden ist, das dem maximalen RGB Farbwert $color_0$ entspricht. Stattdessen repräsentiert $color_2$ den maximalen Farbwert des Blocks. Damit das eingebettete Bit nicht zerstört wird, flippt der Algorithmus alle Pixel mit dem Index '2' zu '0' und alle Pixel mit Index '3' erhalten den Index '2'. Es verbleibt kein Pixel mit Index '3'.

2. Es existiert kein Pixel mehr mit dem Index '1': Somit ist kein Pixel in dem Block vorhanden, das den minimalen Farbwert $color_1$ repräsentiert. Die Blockaktualisierung flippt alle Pixel mit dem Index '3' zum Index '1'. Außerdem sollen die statistischen Eigenschaften, die bis dahin durch das Einbetten erzwungen wurden, nicht verloren gehen und deshalb flippt der Algorithmus die Pixel mit dem Index '2' zum Index '3'. Es verbleiben damit keine Pixel mit dem Index '2'.

Sobald die Blockaktualisierung durchlaufen wurde, führt der Algorithmus das Flippen der Pixel fort, denn es könnten Möglichkeiten entstanden sein, um das Wasserzeichenbit sicherer einzubetten.

6.1.6 Auslesen des Wasserzeichens

Der Detektionsalgorithmus ist blind und erhält zum Auslesen der Wasserzeichennachricht den geheimen Schlüssel und die Parameter, die auch beim Embedder genutzt wurden. Die ausgewählten Blöcke ordnet der Algorithmus den Gruppen α_i oder β_i ebenfalls schlüsselabhängig zu, um daraus das Nachrichtenbit m_i zu extrahieren. Anschließend zählt der Algorithmus die Indizes '0' und '3' in jeder der Gruppen γ_{α_i} und γ_{α_i} entsprechend der Definition (6.2). Danach betrachtet der Algorithmus das Verhältnis der beiden Gruppen, um das Nachrichtenbit auszulesen und trifft dabei folgende Entscheidung:

$$m_i = \begin{cases} 1, & \text{wenn } \log_{10} \frac{\gamma'_{\alpha_i}}{\gamma'_{\beta_i}} > \tau \\ 0, & \text{wenn } \log_{10} \frac{\gamma'_{\alpha_i}}{\gamma'_{\beta_i}} < -\tau \end{cases}$$

Mithilfe des Parameters τ wird gesichert, dass bei einer nicht markierten digitalen Datei auch keine Wasserzeichennachricht auslesbar ist. Liegt die Relation der beiden Gruppen zwischen $-\tau$ und τ, so ist kein Nachrichtenbit auslesbar. Je größer der Betrag des Gruppenverhältnisses ist, desto robuster ist das ausgelesene Wasserzeichenbit.

Der Parameter τ wird hier mit $\tau = 1.39 \cdot 10^{-2}$ gewählt. Der Wert entspricht der Standardabweichung des analysierten und unmarkierten Materials. Mit dieser Wahl können im Durchschnitt $66,7\%$ der Nachrichtenbit des unmarkierten Materials nicht ausgelesen werden und damit lässt sich keine Nachricht aus unmarkiertem Material auslesen. Abbildung 6.4 zeigt die normierte Wahrscheinlichkeitsdichtefunktion der Gruppenverhältnisse $\log_{10} \frac{\gamma'_{\alpha_i}}{\gamma'_{\beta_i}}$ zueinander. Die gestrichelten vertikalen Linien sind das 1-Quantil, das $-\tau$ und τ entspricht. Gruppenverhältnisse links von der linken gestrichelten Linie interpretiert der Algorithmus als Nachrichtenbit '0' und rechts der rechten gestrichelten Linie als '1'.

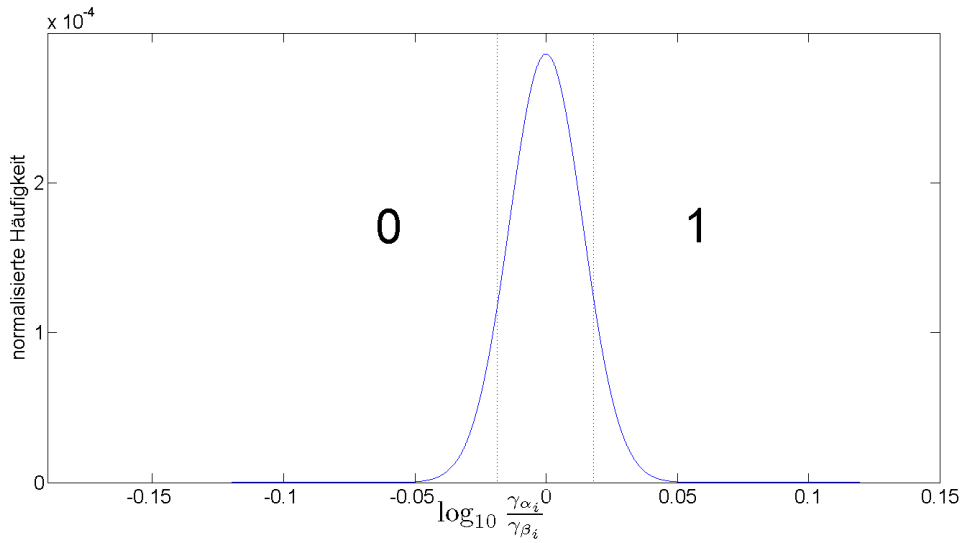


Abbildung 6.4: Dichtefunktion der Relation zwischen den beiden Gruppen γ'_{α_i} und γ'_{β_i} aus unmarkiertem Content, die das Verhältnis der Indizes '0' und '3' der beiden Gruppen zueinander darstellen. Die Ordinate zeigt die relative Häufigkeit der entsprechenden Gruppenverhältnisse, die auf der Abszissenachse abgetragen sind. Die gestrichelte Linie entspricht links $-\tau$ und rechts τ.

6.2 Tests und Ergebnisse

Im Folgenden wird der vorgestellte Algorithmus zum Markieren von DDS komprimierten Texturen evaluiert. Nach dem Einbetten des Wasserzeichens werden verschiedene Manipulationen durchgeführt und anschließend die Robustheit und Transparenz bewertet.

6.2.1 Testaufbau

Als Testbasis wurden Texturen aus den drei Videospielen ausgewählt, nämlich *The Elder Scrolls V:Skyrim*, *Fallout 3* und *Dragon Age:Origins*. Insgesamt besteht die Testbasis aus 47.000 Texturen, wobei auf SKYRIM mehr als 18.000, *Fallout 3* mehr als 12.000 und *Dragon Age* mehr als 16.000 Texturen entfallen.

Blöcke mit nur einem Index werden vom Algorithmus nicht berücksichtigt, da sie einheitliche Flächen repräsentieren. Zum Beispiel beinhalten 21% der insgesamt 160 Millionen Blöcke aus SKYRIM nur einen Index. Diese Blöcke bilden hauptsächlich den transparenten Hintergrund der Texturen, deren Alphakanal '0' und damit unsichtbar ist. In der Evaluierung werden nur Pixel verwendet, deren Alphakanal größer als $0,5$ ist.

Die Gruppen γ'_{α_i} und γ'_{β_i} ergeben sich jeweils aus 1.200 schlüsselabhängig ausgewählten Blöcken. Die Einbettungsstärke w_{int} wird zwischen $0,35$ und $0,55$ gewählt.

6.2.2 Visuelle Qualität

Die visuelle Qualität der markierten Texturen wird mit dem PSNR (Peak Signal-to-Noise Ratio) als Metrik evaluiert. Damit werden die durch das Einbetten des Wasserzeichens entstandenen absoluten Fehler gemessen und bewertet. Die in Kapitel 7 eingeführte Metrik, die ebenfalls die Qualität objektiv bewertet, wird hier nicht herangezogen. Denn hier bleibt das 3D-Modell unberücksichtigt. Die Texturen sind unabhängig von ihren zugehörigen 3D-Modellen markiert worden und somit können Artefakte nach dem Mapping der Textur auf das 3D-Modell auftreten.

Der PSNR ist von der wahrgenommenen Qualität eines menschlichen Betrachters lösgelöst. Er wird aber als Maß verwendet, um die durch die Wasserzeichenmarkierung vorgenommenen Änderungen auszudrücken. Der PSNR ist wie folgt definiert. Sei I die originale Textur und K die wasserzeichenmarkierte Textur mit der Größe $m \times n$, dann gilt:

$$PSNR := 10 \cdot \log_{10}(\frac{255^2}{\sqrt{MSE}}), \quad MSE := \frac{1}{m \cdot n} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} [I(x,y) - K(x,y)]^2.$$

$w_{int} = 0.35$	$w_{int} = 0.45$	$w_{int} = 0.55$	$w_{int} = 0.55$
38.76	35.88	34.21	32.8

Tabelle 6.1: Durchschnittliche PSNR Werte über die gesamte Testbasis. Die PSNR Werte sind in (db) angegeben

Abhängig der unterschiedlichen Einbettungsstärken wird der Mittelwert über alle Texturen aus der Textbasis in Tabelle 6.1 angegeben. Mit steigender Einbettungsstärke w_{int} nimmt der Wasserzeichenalgorithmus entsprechend mehr Änderungen beim Einbetten vor. Die PSNR Werte sind mit gängigen Bildwasserzeichenverfahren vergleichbar.

6.2.3 Evaluierung der Robustheit und Sicherheit

Zur Auswertung der Robustheit und Sicherheit des Wasserzeichens wurde

1. die Textur verlustbehaftet nach JPEG komprimiert,
2. ein Hintergrund-Rausch Angriff,
3. ebenso wie ein Mip-Map Angriff durchgeführt.

6.2.3.1 JPEG-Kompression

JPEG ist einer der populärsten Bildkompressionsstandards und kann problemlos auf Texturen angewandt werden. Das Wasserzeichen muss die JPEG-Kompression bis zu einem gewissen Grad überstehen, da die Qualität nach einer geringen Kompressionsrate akzeptabel bleibt. Starke Kompressionsraten können jedoch Artefakte hervorrufen.

Bei den Tests wurden die DDS-Texturen mit einer unterschiedlichen Anzahl an Wasserzeichenbit markiert und zwar mit einem, zwei, vier oder acht Bit. Danach wurden sie in das Bitmap Format konvertiert und anschließend in das verlustbehaftete JPEG-Format mit dem Faktor 75 und 50 komprimiert. Beim Zurückführen wurden die Texturen vom JPEG-Format wieder über das Bitmap-Format in das DDS-Format konvertiert.

Abbildung 6.5 zeigt die mittlere Detektionsrate nach der JPEG-Kompression mit unterschiedlichen Kompressionsraten und Einbettungsstärken. Die Detektionsrate D_r ist definiert als das Verhältnis der Anzahl der korrekt ausgelesenen Wasserzeichenbit W^* zu der Anzahl der eingebetteten Nachrichtenbit $|W|$: $D_r := 1 - \frac{HD(W^*,W)}{|W|}$, wobei mit $HD(x, y)$ die Hammingdistanz zwischen x und y gemeint ist. In der Abbildung ist die Detektionsrate abhängig von der Anzahl der eingebetteten Nachrichtenbit dargestellt.

Nach einer JPEG-Kompression von 75 und einer Wasserzeichenintensität von $w_{int} = 0.55$ kann der Algorithmus 98.4% der Wasserzeichenbit richtig auslesen. Bei geringeren Einbettungsstärken liegt der Algorithmus bei gleicher JPEG-Kompressionsrate das Wasserzeichen immer noch sehr verlässlich aus. Damit zeigt der Algorithmus eine hohe Robustheit gegenüber einer JPEG-Kompression.

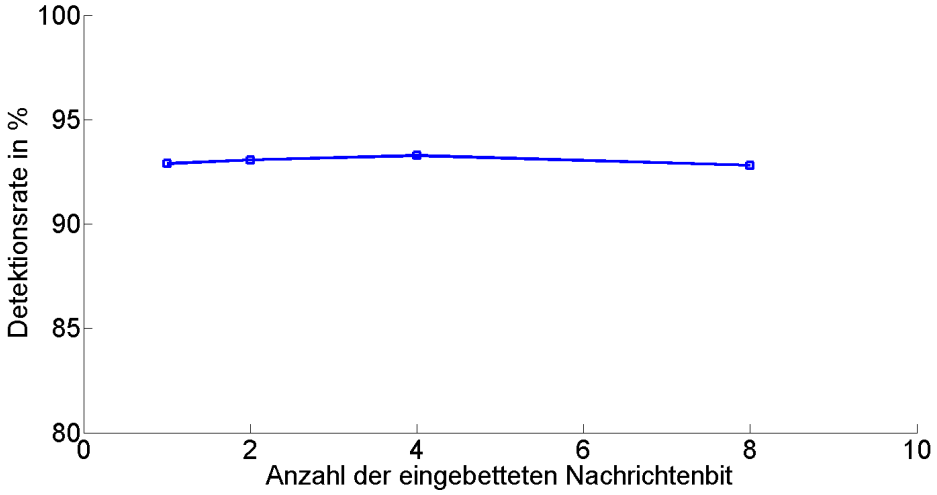


Abbildung 6.5: Die Detektionsrate ist nach verschiedenen JPEG-Kompressionen und unterschiedlichen Einbettungsstärken w_{int} auf der Abzisse in Abhängigkeit der Anzahl der Nachrichtenbit abgetragen.

6.2.3.2 Hintergrund-Rausch Angriff

Für Texturen sind verschiedene Angriffe denkbar, die nur eine geringe Auswirkung auf die visuelle Qualität des texturierten 3D-Modells haben. Ein solcher Angriff ist der Hintergrund-Rausch Angriff. Dabei werden die Pixel manipuliert von Bereichen, die nicht direkt gesehen werden. Es sind die Bereiche, die z.B. Zwischenräume zwischen den Objekten abdecken. Diese Pixel werden mit zufällig gewählten Farbwerten überschrieben. Der PSNR wird zwar kleiner, aber die wahrnehmbare Qualität für den menschlichen Betrachter bleibt unverändert hoch.

Abbildung 6.6 zeigt die originale Textur auf der linken Seite und auf der rechten Seite die Textur nach dem Hintergrund-Rausch Angriff. Dabei wurden Bereiche, deren Alphakanal unter 0.1 lag, mit zufälligen Farbwerten gefüllt. In Abbildung 6.7 wird eine Szene aus

dem Videospiel SKYRIM gezeigt, die die Textur aus Abbildung 6.6 beinhaltet. Dabei ist oben die Szene aus dem originalen Videospiel und unten die gleiche Szene nach dem Hintergrund-Rausch Angriff.

Nach diesem Angriff können 99.1% aller Wasserzeichenbit richtig ausgelesen werden. Der DDS Wasserzeichenalgorithmus nutzt nur solche Bereiche, deren Alphakanal größer als 0.5 ist.

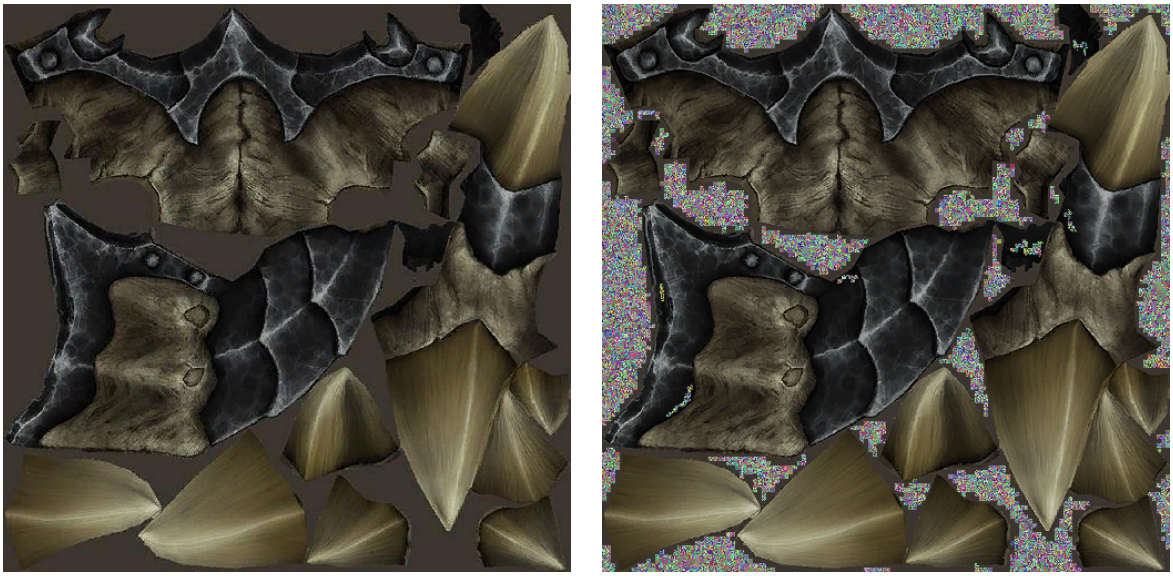


Abbildung 6.6: Links die originale Textur aus dem Videospiel SKYRIM und rechts die Textur nach einem Hintergrund-Rausch Angriff.

6.2.3.3 Mip-Map-Angriff

Das DDS-Format sieht Mip-Maps vor, in denen die Textur in unterschiedlichen Auflösungen vorgehalten wird. Das minimiert die Grafikkartenauslastung und beschleunigt die Anzeige einer Szene. Dabei wird bei Objekten, die weiter hinten in einer Szene sind, ein höheres Mip-Map-Level der Textur genutzt und damit die niedrig aufgelöstere Textur geladen. In jedem Mip-Map-Level ist die Textur mit einer Auflösung von einem viertel zum vorhergehenden Level abgelegt. Da die Texturen in den betrachteten Videospielen gleich lange Seiten haben und ein DDS Block die Größe $4x4$ Pixel hat, sind die Seitenverhältnisse der Texturen immer eine Zweierpotenz und immer quadratisch. Die Texturen liegen teilweise

bis zu dem Mip-Map-Level 10 vor. Abbildung 6.8 zeigt die Textur *Bonecrown* mit ihren verschiedenen Mip-Map-Level. Die Textur hat die Größe $512x512$ und die Auflösung im Mip-Map-Level 6 ist $16x16$.

Beim Mip-Map-Angriff skaliert der Angreifer die Textur in der Auflösung hoch. Damit erhält er abhängig des verwendeten Skalierungsfaktors eine akzeptable Qualität. Wurde z.B. nur das Mip-Map-Level 0 mit der Wasserzeichennachricht markiert, so kann der Angreifer vom Mip-Map-Level 1 die Textur hochskalieren und erhält eine Textur ohne Wasserzeichen. Wird das Wasserzeichen über alle Level verteilt eingebettet, so kann mit der beschriebenen Vorgehensweise ein Teil der Wasserzeichennachricht gelöscht werden.

Die gleiche Wasserzeichennachricht wird auf allen Mip-Map-Level eingebettet, um eine möglichst hohe Sicherheit zu erreichen. Die Wasserzeicheninformation wird dadurch auf den höheren Mip-Map Level weniger sicher eingebettet im Vergleich zu den niedrigen, aber der beschriebene Angriff ist immer noch ausreichend sicher, um Angriffe zu überstehen.

Testaufbau des Mip-Map-Angriffs: Es werden drei verschiedene wasserzeichenmarkierte Kopien des Videospiels *The Elder Scrolls V:Skyrim* mit dem vorgestellten DDS-Wasserzeichenverfahren bis einschließlich Mip-Map-Level 3 erzeugt. Die Wasserzeichenlänge beträgt dabei 100 Bit. Die Anzahl der verwendeten Blöcke, um ein Wasserzeichenbit einzubetten, ist im Mip-Map-Level 3 mit $\gamma'_{\beta_i} = \gamma'_{\alpha_i} = 20.000$ gewählt und der Intensitätsparameter auf $w_{int} = 0.45$ gesetzt. Der Detektionsschwellwert wird mit $\tau = 0,013967$ gewählt.

Testergebnis des Mip-Map-Angriffs: Sei T eine markierte Textur, M_i eine markierte Mip-Map aus Level i vor einem Angriff auf die DDS-Datei, T' die Textur und M'_i die Mip-Map aus Level i der angegriffenen DDS-Datei. Die Detektionsraten nach den Angriffen können der Tabelle 6.2 entnommen werden. Die Tabelle zeigt, dass ein erfolgreiches Detektieren immer nur dann möglich ist, wenn die hochskalierte Textur in das entsprechende Mip-Map Level zurückskaliert wird. Damit das Detektionsverfahren weiterhin blind bleibt, ist nur der Weg über das Ausprobieren möglich. Nur dann, wenn ein geringer prozentualer Anteil an "?" zurückgegeben wird, kann vom richtigen Level ausgegangen werden. Ein "?" wird immer dann vom Detektionsalgorithmus zurückgegeben, wenn kein Bit detektiert werden kann. Tabelle 6.2 zeigt, dass der Wert bei maximal 1% liegt, wenn das Detektieren im korrekten Mip-Map Level stattfindet, ansonsten ist der Wert bei mindestens 21%. Damit wird das Verfahren auch sicher gegen einen Mip-Map-Angriff.

	T'	M_1'	M_2'	M_3'
M_1				
Korrekt	41%	99%	25%	25%
?	32%	1%	37%	47%
Falsch	27%	0%	38%	28%
M_2				
Korrekt	42%	20%	99%	28%
?	21%	32%	1%	41%
Falsch	37%	48%	0%	31%
M_3				
Korrekt	41%	24%	27%	98%
?	44%	62%	60%	1%
Falsch	15%	14%	13%	1%

Tabelle 6.2: Evaluationsergebnisse nach dem Mip-Map basierten Angriff. Mit M_1 ist das Hochskalieren aus dem Mip-Map Level 1 gemeint, mit M_2 das Hochskalieren von Level 2 und mit M_3 von Level 3. “Korrekt“ bedeutet das richtige Wiedererkennen des eingebetteten Bits. “?“ bedeutet, dass das Bit nicht zuweisbar ist, und somit nicht klar ist, ob das Bit eine “0“ oder “1“ sein soll. “Falsch“ bedeutet, dass das Wasserzeichenbit falsch detektiert wurde.

6.3 Zusammenfassung

Texturen mit herkömmlichen Bildwasserzeichenalgorithmen zu markieren ist ungeeignet. Die Anwendung von Bildwasserzeichenverfahren auf Texturen genügen nicht den Anforderungen der Sicherheit gegen Angriffe. Entsprechend den Anforderungen aus Kapitel 3.3 wurde ein DDS Wasserzeichenalgorithmus entworfen. Bei der Konzeptionierung wurden Angriffe berücksichtigt, die speziell bei Texturen möglich sind, wie z.B. der Hintergrund-Rausch-Angriff und Mip-Map-Angriff.

Das Verfahren bettet die Wasserzeicheninformation auf den bereits komprimierten Daten ein, womit die Dekomprimierung und erneute Kompremierung entfallen. Es ist ein blockbasierter Ansatz, der die Koeffizienten innerhalb eines Blocks durch Flippen auf einen im Block enthaltenen Koeffizienten erlaubt, um so auch die Nicht-Wahrnehmbarkeit der Wasserzeichenmarkierung zu erreichen.

Eine Bewertungsmethode zur Nicht-Wahrnehmbarkeit wasserzeichenmarkierter, texturierter 3D-Modelle wird in Kapitel 7 eingeführt und genutzt.

Abbildung 6.7: Eine Szene aus dem Videospiel Skyrim , die die Textur aus Abbildung 6.6 beinhaltet. Im oberen Spielausschnitt wurde die originale Textur verwendet; unten die Textur nach einem Hintergrund-Rausch Angriff

Abbildung 6.8: Die Textur (bonecrown.dds) aus SKYRIM mit den sechs unterschiedlichen Mip-Map-Level, die auf $512x512$ hoch skaliert wurden. Eine höheres Mip-Map Level bedeutet die halbe Kantenlänge des unteren Levels.

7 Metrik zur Bewertung der Nicht-Wahrnehmbarkeit der Wasserzeichen bei texturierten 3D-Modellen

In diesem Kapitel wird eine Metrik eingeführt zur Bewertung der visuellen Qualität von wasserzeichenmarkierten Texturen, die auf wasserzeichenmarkierte 3D-Modelle gemappt wurden. Die vorgestellte Metrik ist eine Kombination aus einer 2D und einer 3D-Metrik. Das Ziel der Metrik ist es, die Nicht-Wahrnehmbarkeit und damit visuelle Qualität des texturierten 3D-Modells nach der Wasserzeichenmarkierung so zu bewerten, wie sie von einem menschlichen Betrachter auf einem 2D-Bildschirm wahrgenommen wird. Diese Metrik ist im Umfang einer betreuten Abschlussarbeit [106] entwickelt und publiziert [17] worden.

Die Nicht-Wahrnehmbarkeit ist, wie in Kapitel 2.2 eingeführt, eine der relevanten Eigenschaften eines digitalen Wasserzeichens. Es gibt bereits Metriken zur Bewertung der visuellen Qualität eines modifizierten Bildes [102, 104] oder eines 3D-Modells [52, 30, 31]. Es sind aktuell keine Arbeiten bekannt, die eine Kombination aus einer Metrik für Texturen und 3D-Modellen betrachten. Zur Visualisierung oder dem Rendern von Szenen werden Texturen auf ihr zugehöriges 3D-Modell gemappt. Durch das Mapping erfährt die Textur eine dreidimensionale Verzerrung. Wird eine Textur im 2D-Bereich verändert, werden diese Veränderungen nach dem Mapping auf das zugehörige 3D-Modell von einem Betrachter anders wahrgenommen.

Bei der Umsetzung der Metrik wird eine 2D-Metrik angepasst und mit einer lokalen 3D-Bewertung kombiniert. Damit lässt sich die visuelle Qualität der Textur bei Animationen messen und die Parametrisierung der Wasserzeichenalgorithmen vornehmen.

Unter Berücksichtigung der Anforderungen werden im Folgenden bestehende Ansätze analysiert.

Zur Bewertung der Qualitätsunterschiede bei Bildern nach durchgeführten Änderungen existieren unterschiedliche Metriken. Einige sind bei der Bewertung an ein psychovisuelles Modell angelehnt andere sind davon lösgelöst.

Die beiden Messgrößen des Mean Square Error (MSE) [80] und des Peak Signal-to-Noise Ratio (PSNR) sind performant, jedoch bewerten sie nur die Pixelunterschiede, ohne die menschliche Wahrnehmung zu berücksichtigen. Im Gegensatz dazu nutzt sowohl die Mean Structural Similarity Metrik (MSSIM) [104] als auch die Multi-Scale Structural Similarity Metrik (MS-SSIM)[102] jeweils ein psychovisuelles Modell. Die MS-SSIM baut auf der MSSIM auf und ist präziser in ihrer Bewertung, da sie unterschiedlich skalierte Versionen mit berücksichtigt. Somit werden niedrig aufgelöste Texturen, im Vergleich zu hoch aufgelösten, entsprechend fair bewertet. Die Mindestanforderung der MS-SSIM ist eine $16x16$ Pixel große Textur. Dies ist bei Texturen nicht immer gegeben und so können niedriger aufgelöste Texturen nicht mittels der MS-SSIM bewertet werden. Aus diesem Grund wird im Folgenden die MSSIM genutzt.

Im Folgenden wird die MSSIM so erweitert, dass sie transparente Bereiche nicht in der Bewertung berücksichtigt. Anschließend wird die Erweiterung mit einer 3D-Modell Metrik so kombiniert, dass die resultierende Metrik texturierte 3D-Modelle bewerten kann.

7.1 Metrik zur Bewertung der Nicht-Wahrnehmbarkeit

Die folgende Metrik eignet sich zum Bewerten von texturierten 3D-Modellen nach Veränderungen durch die Wasserzeicheneinbettung. Dazu werden zwei aus der Literatur bekannte Algorithmen angepasst und zusammengeführt. Anschließend wird die Erweiterung des Algorithmus vorgestellt, die die Qualität der texturierten 3D-Modelle bei Animationen bewertet.

7.1.1 Konzept

Als Grundlage der Qualitätsmetrik dient eine modifizierte Version der Mean Structural Similarity Metric (MSSIM) [104] gemeinsam mit einer lokalen flächenbasierten 3D-Modell Metrik [107]. Berücksichtigt werden in der Metrik von 3D-Modellen die Struktur- und Beleuchtungseinflüsse, sowie die Verzerrung. Bei der Berechnung der Gesamtmetrik wird der Einfluss des 2D-Bereichs mit der 3D-Komponente potenziert. Bei Animationen wird der niedrigste Wert der Gesamtmetrik als Resultat ausgegeben. Um den niedrigsten Wert zu finden, enthält das Konzept eine Iterationsschleife über die unterschiedlichen Möglichkeiten der Gelenkstellungen des 3D-Modells. Der Wertebereich der Metrik liegt im

Bereich von $W \in (0, 1]$, wobei 1 nur dann erreicht wird, wenn die beiden zu vergleichenden Texturen identisch sind. Abbildung 7.1 zeigt den Ablauf des Algorithmus im Überblick.

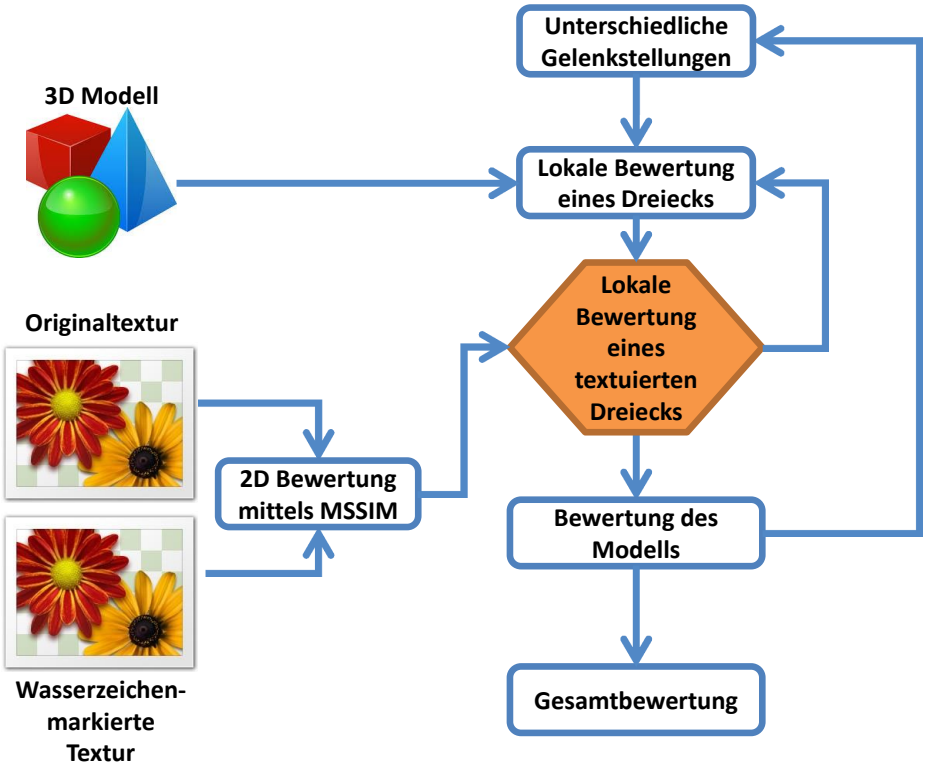


Abbildung 7.1: Überblick des Konzepts zur Qualitätsbewertung

7.1.2 Anpassung und Optimierung der 2D-Metrik

Dieser Abschnitt beschreibt die Erweiterung der MSSIM Metrik von Wang et al. [103]. Die Metrik wird so erweitert, dass sie die Besonderheit bei Texturen berücksichtigt. Es existieren Bereiche in der Textur, die nicht verwendet werden. Solche Bereiche sollen auch nicht in die Bewertung mit einfließen.

Hierzu wird der Alphakanal, der die Information der Transparenz eines Pixels trägt, herangezogen. Der Algorithmus multipliziert dafür den auf den Wertebereich $[0, 1]$ normalisierten Alphakanal mit dem zugehörigen Pixelwert, um so den gewichteten Grauwert

des jeweiligen Pixels für die Bewertung zu erhalten. Sei x_i der Grauwert des Pixels an der Position i, x'_i der neue gewichtete Grauwert des Pixels an der Stelle i, α_i der Alphawert des gleichen Pixels und $\alpha_{max} = 255$ der maximal mögliche Alpha-Wert, dann gilt:

$$x'_i := x_i \cdot \frac{\alpha_i}{\alpha_{max}}. \tag{7.1}$$

Gleichung (7.1) sichert, dass transparente Bereiche bei denen $\alpha_i = 0$ gilt, der gewichtete Grauwert ebenfalls $x'_i = 0$ wird. Für Bereiche mit $\alpha_i = 1$ gilt $x'_i = x_i$.

Die MSSIM ist definiert als der Mittelwert der SSIM Werte über alle lokalen Fenster j:

$$MSSIM = \frac{1}{M} \sum_{j=1}^{M} SSIM_j, \tag{7.2}$$

wobei M die Anzahl der verwendeten lokalen Fenster ist. Beim Bilden der MSSIM besteht die Möglichkeit, die lokalen Fenster unterschiedlich zu gewichten. Dazu wird ein Gaussian Fenster $G(x, y)$ der Größe 11×11 Pixel verwendet mit der Standardabweichung $\sigma_{x,y} = 1.5$ in beiden Dimensionen x und y. Lokale Fenster am Rand der Textur werden im Vergleich zu denen in der Mitte weniger stark gewichtet. Die MSSIM aus Gleichung 7.2 wird um den Alphakanal erweitert zu $M_{2D,\alpha}$ und definiert als:

$$M_{2D,\alpha} := \frac{1}{\frac{1}{M}\sum_{j=1}^{M}(\sum_{i,k}^{11} \alpha_{x_i,y_k} \cdot G(x_i, y_k))} \sum_{j=1}^{M} (\sum_{i,k}^{11} \alpha_{x_i,y_k} \cdot G(x_i, y_k))_j \cdot SSIM_j. \tag{7.3}$$

Die Normierung des Wertebereichs der Metrik wird durch die Division der Summe des gewichteten Alphakanals erreicht.

7.1.3 Anpassung der 3D-Metrik

Dieser Abschnitt beschreibt die notwendigen Anpassungen der 3D-Metrik nach Wue et al. [107], die in Kapitel 2.7 eingeführt wurde.

Der flächenbasierte Ansatz errechnet das Skalarprodukt der Normalen $\vec{N}^{(k)}, \vec{N}^{(k+1)}$ benachbarter Dreiecksflächen, die eine gemeinsame Kante haben und daraus mit Hilfe der Mittelwerte und Standardabweichungen die Rauheit. Damit die 3D- mit der 2D-Metrik zusammengefasst werden kann, muss vorher eine Normalisierung für p_d aus Gleichung (2.1) erfolgen.

Es wird dazu die Annahme getroffen, dass raue Oberflächen Änderungen an der Textur weniger sichtbar erscheinen lassen. Dazu wird der Winkel ϕ ausgewertet, der von zwei benachbarten Dreiecken mit gemeinsamer Kante eingeschlossen ist. Der Winkel liegt

zwischen $\phi \in [0, \pi]$ und errechnet sich über die Normalen der Dreiecke. Wird die erste Gleichung aus (2.1) mit π normiert, so ergibt sich die angepasste Berechnung für p_d wie folgt:

$$p_d := \frac{1}{\pi} \cdot \vec{N}^{(k)} \cdot \vec{N}^{(k+1)} \tag{7.4}$$

Der Wertebereich ist $p_d \in [0, 1]$. Der Wert "1"bedeutet die beiden Dreiecke liegen auf einer Ebene und schließen einen Winkel von $180°$ ein. Der Wert "0"widerspiegelt damit sehr raue Flächen, bei denen die Dreiecke einen Winkel von $360°$ einschließen.

Das Mappen der Textur auf ihr zugehöriges 3D-Modell bringt außerdem Verzerrungen und Flächenänderungen der 2D-Textur mit sich. Dadurch ändert sich die Wahrnehmung für ein menschliches Auge im Vergleich zur Betrachtung der 2D-Metrik. Im Folgenden werden diese zwei weiteren Komponenten eingeführt.

Einfluss der Verzerrung Verzerrungen bedeuten im Allgemeinen eine Qualitätsminderung. Sie wirken sich beim Mapping sowohl für die originale als auch die wasserzeichenmarkierte Textur gleichermaßen aus. Ist zusätzlich das 3D-Modell unabhängig der Textur markiert worden, müssen die durch das Wasserzeichen vorgenommenen Änderungen am 3D-Modell berücksichtigt werden.

Zur Bewertung des Einflusses der Verzerrung wird die UV-Map herangezogen. Sie ordnet jeder Texturposition einen Punkt des 3D-Modells zu. Verzerrungen lassen sich am einfachsten aus Winkeländerungen der einzelnen Dreiecke des 3D-Modells beschreiben. Wird ein Winkel des Dreiecks vergrößert, so werden mehr Pixel in diesem Bereich benötigt, die durch eine Interpolation berechnet werden. Beim Verkleinern eines Winkels werden Pixel in diesem Bereich zusammengefasst.

Der Einfluss der Verzerrung auf die visuelle Wahrnehmung wird durch einen Winkelvergleich errechnet. Seien $\alpha_T^{(k)}, \beta_T^{(k)}$ und $\gamma_T^{(k)}$ die drei Winkel eines Dreiecks k der UV-Map und seien $\alpha_M^{(k)}, \beta_M^{(k)}$ und $\gamma_M^{(k)}$ die Winkel des entsprechenden Dreiecks k des 3D-Modells, dann kann die Verzerrung $V^{(k)}$ wie folgt errechnet werden:

$$V^{(k)} = 1 - \frac{|\alpha_T^{(k)} - \alpha_M^{(k)}| + |\beta_T^{(k)} - \beta_M^{(k)}| + |\gamma_T^{(k)} - \gamma_M^{(k)}|}{360} \tag{7.5}$$

Der Wertebereich der Verzerrung $V^{(k)}$ liegt bei $[0, 1]$, wobei 1 keiner Änderung entspricht und 0 ist nur erreichbar, wenn eine starke Veränderung vorliegt.

Einfluss von Flächenänderungen Das Mapping verursacht eine Flächenänderung, die dem Zoomen entspricht. Wird das Dreieck der UV-Map durch das Mapping kleiner darge-

stellt, dann sind Texturunterschiede zwischen der originalen und der wasserzeichenmarkierten Textur weniger sichtbar und andersherum.

Damit der Einfluss der Flächenänderung fair bewertet werden kann, ist eine Normierung notwendig, die es erlaubt, einen Bezug zwischen der Mantelfläche und der Texturfläche herzustellen.

Sei A_{3D} der Flächeninhalt der Mantelfläche des 3D-Modells, A_{2D} der Flächeninhalt der verwendeten Flächen aus der Textur und N ein Normierungsfaktor. Sei ferner $A_{2D}^{(k)}$ der Flächeninhalt des Dreiecks k der Textur und $A_{3D}^{(k)}$ der Flächeninhalt des entsprechenden Dreiecks k des 3D-Modells, dann lässt sich der Einfluss der Flächenänderung eines Dreiecks $F^{(k)}$ wie folgt berechnen:

$$F^{(k)} = N \cdot \frac{A_{2D}^{(k)}}{A_{3D}^{(k)}} \qquad mit \qquad N = \frac{A_{3D}}{A_{2D}} \tag{7.6}$$

7.1.4 Zusammenführen der 2D und 3D-Metrik

Nachdem die einzelnen Einflussgrößen eingeführt wurden, wird in diesem Abschnitt die Zusammenführung der 2D und 3D-Komponente zu einer Metrik beschrieben. Zur fairen Bewertung der visuellen Wahrnehmung texturierter Modelle bei Animationen wird am Ende des Abschnitts die Metrik im Zusammenspiel mit einem evolutionären Algorithmus vorgestellt.

Zur Gesamtbewertung wird der mit dem Alphakanal gewichtete 2D-Anteil aus Kapitel 7.1.2 mit den in Kapitel 7.1.3 vorgestellten 3D-Komponenten verknüpft. Die 3D-Komponenten aus den Gleichungen 2.1, mit der Anpassung aus 7.4, 7.5 und 7.6, werden gewichtet und additiv verknüpft. Sie werden additiv und nicht multiplikativ verknüpft, da die einzelnen Komponenten sonst wechselseitigen Einfluss aufeinander ausüben würden. Zusammen ergeben sie die 3D-Einflussgröße für ein einzelnes Dreieck k des 3D-Modells $M_{3D}^{(k)}$:

$$M_{3D}^{(k)} := w_1 \cdot F^{(k)} + w_2 \cdot V^{(k)} + w_3 \cdot R^{(k)} \qquad \text{mit} \qquad w_1 + w_2 + w_3 = 1 \tag{7.7}$$

Die Gewichte w_i wurden mit $w_1 = w_2 = w_3 = \frac{1}{3}$ gewählt. Damit wird der Einfluss der Rauheit, Verzerrung und Flächenänderung gleich bewertet.

Für den Wertebereich der Metrik soll gelten $W_M \in (0, 1]$. Das bedeutet, dass die Metrik nur dann 1 sein soll, wenn beide Texturen und 3D-Modelle identisch sind. Um das zu erreichen, wird der 2D-Anteil eines Dreiecks mit dem 3D-Anteil desselben Dreiecks potenziert und über alle Dreiecke aufsummiert. Sei dazu $M_{2D,\alpha}$ der mit dem Alphakanal

normierte 2D-Anteil aus Gleichung 7.3, $M_{3D}^{(k)}$ der 3D-Anteil aus Gleichung 7.7 und N die Anzahl der Dreiecke des 3D-Modells, dann wird die Metrik M wie folgt definiert:

$$M := \frac{1}{n} \sum_{k=1}^{N} {M_{2D,\alpha}}^{M_{3D}^{(k)}} \tag{7.8}$$

Verwendung der Metrik zur Bewertung von Animationen Beim Bewerten animierter 3D-Modelle soll die Metrik das Modell in der Position bewerten, in der die Modifikationen durch das Wasserzeichen am meisten auffallen. Die Metrik bewertet diese Stellung entsprechend am schlechtesten. Um die Position zu finden, in der das texturierte Modell am schlechtesten bewertet wird, gibt es mehrere Möglichkeiten.

Erstens kann die Metrik für alle möglichen Gelenkstellungen berechnet werden und dann der kleinste Wert als Ergebnis zurückgegeben werden. Das ist jedoch sehr rechen- und zeitintensiv. Der Rechenaufwand, alle Bewertungen zu errechnen, ist bei einer Anzahl von N Gelenken $\mathcal{O}(2^N)$. Zweitens lässt sich ein Minimum der Metrik durch ein Optimierungsverfahren, wie z.B. das Gradientenverfahren, finden. Das dadurch gefundene Minimum ist meistens nicht das globale Minimum, sondern ein lokales. Um sicherzustellen, dass der Algorithmus immer das globale Minimum findet, muss das zu optimierende Problem konvex sein. Da die Metrik nicht ohne Weiteres in ein konvexes Problem überführt werden kann, wird ein evolutionärer Algorithmus zum Finden des Minimums vorgeschlagen. Dazu werden die Grundlagen evolutionärer Algorithmen vorausgesetzt [39].

Die zu optimierende Funktion ist die Metrik aus Gleichung 7.8. Die Gelenke sind die änderbaren Größen. Für die Startpopulation werden zufällig Kandidaten erwürfelt, da die Position, in der die Metrik am schlechtesten bewertet, nicht abschätzbar ist. Die Wettkampfselektion wird zur Bestimmung der nächsten Generation gewählt. Da wir ein Minimierungsproblem haben, gewinnt der Kandidat mit dem geringsten Wert. Bei der Mutation wird immer zufällig ein Gelenk verändert. Von dem Gelenk abhängige Gelenke werden dadurch mit verändert. Eine 2-Punkt-Rekombination wird zur Rekombination verwendet, denn dadurch lassen sich die Änderungen am Gesamtmodell, verursacht durch die Gelenkabhängigkeiten, verringern. Von einer lokalen Suche in der Nähe der aktuellen Lösungskandidaten wird abgesehen, da es für das vorliegende Optimierungsproblem zu rechenintensiv ist.

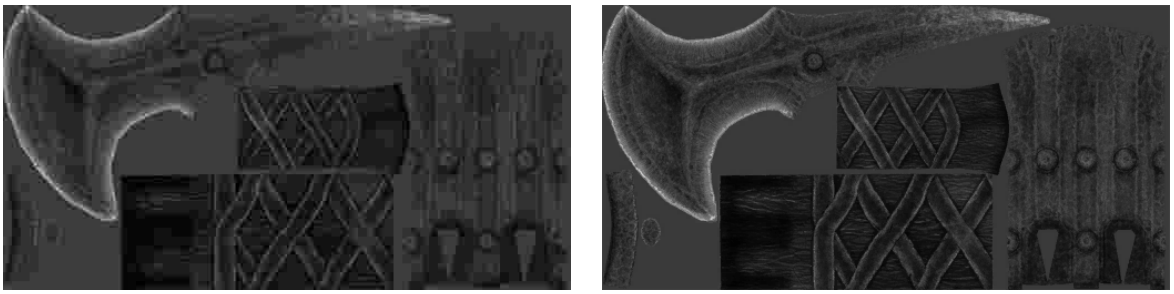
7.2 Evaluierung

In diesem Abschnitt wird die Metrik evaluiert. Dabei werden die Komponenten aus dem 2D- und 3D-Bereich im einzelnen auf ihren Einfluss zur Gesamtmetrik hin evaluiert.

7.2.1 Vergleich zu MSSIM

Für den Vergleich der Metrik MSSIM aus [103], mit der Erweiterung aus Kapitel 7.1.2, werden Texturen aus dem Videospiel THE ELDER SCROLLS V: SKYRIM als Testbasis gewählt. Die Texturen aus diesem Videospiel eigenen sich gut als Testbasis, da sie sehr unterschiedlich in ihrer Größe, der Helligkeit, dem Kontrast und dem Inhalt sind. Weitere Vorteile sind, dass der Hintergrund einfarbig und der Alphakanal null ist. Für diesen Evaluierungsschritt werden die Texturen nach JPEG komprimiert, wobei die komprimierte Textur nur noch 10% der Qualität vom Original hat. Anschließend werden die Texturen von beiden 2D-Metriken bewertet.

Erwartet wird, dass die Ergebnisse sich nicht stark unterscheiden, da der Hintergrund nicht weiter komprimiert werden kann, und somit bei der Bewertung der originalen und komprimierten Textur in diesem Bereich nahezu keine Unterschiede vorhanden sind. Lediglich an den Rändern zwischen den Objekten und dem Hintergrund entstehen Artefakte, die sich in einer unterschiedlichen Bewertung niederschlagen. Abbildung 7.2 zeigt beispielhaft eine mit 10% JPEG komprimierte Textur aus der Testbasis links und die zugehörige originale Textur rechts.



Komprimiert mit 10% JPEG | Original

Abbildung 7.2: Textur “iron battle axe“ aus SKYRIM, wobei links die mit 10% JPEG komprimierte und rechts die entsprechende originale Textur abgebildet ist.

$w_{int} = 0.35$	$w_{int} = 0.45$	$w_{int} = 0.55$	$w_{int} = 0.55$
38.76	35.88	34.21	32.8

Tabelle 7.1: Evaluationsergebnisse $MSSIM$ vs. $MSSIM_\alpha$

$w_{int} = 0.35$	$w_{int} = 0.45$	$w_{int} = 0.55$	$w_{int} = 0.55$
38.76	35.88	34.21	32.8

Tabelle 7.2: Ergebnisse der 2D-Metriken $MSSIM$ und $MSSIM_\alpha$, nachdem die transparenten Pixel auf den Farbwert weiß gesetzt wurden.

Die Testergebnisse aus Tabelle 7.1 zeigen eine ähnliche Bewertung der Texturen durch beide Metriken. Damit behält die in Kapitel 7.1.2 eingeführte Metrik die Eigenschaften der MSSI-Metrik bei.

Einfluss des Alphakanals In diesem Abschnitt wird der Einfluss des Alphakanals auf die Gewichtung evaluiert. Dazu werden die Texturen so modifiziert, dass alle Pixel, die einen Alphawert von unter 5 haben, auf den Farbwert weiß gesetzt werden, und ihr Alphakanal erhält den Wert 0. Damit ändert sich im Videospiel nichts für einen Nutzer, aber die Texturen im 2D-Bereich sehen unterschiedlich aus, wie auch in Abbildung 7.3 zu sehen ist. Erwartet wird, dass die hier vorgestellte Metrik die modifizierte Textur im Vergleich zur originalen Textur nahezu identisch bewertet, also einen Wert nahe 1 hat.

In Abbildung 7.3 ist die originale und die modifizierte Textur mit einem Screenshot aus einer Szene, in der die Textur vorkommt, zu sehen. Die Bereiche der Textur, die einen Alphawert nahe 0 haben, sind vollständig durchsichtig.

Die Evaluierungsergebnisse in Tabelle 7.2 zeigen, dass die MSSIM aus [103] die Textur schlechter bewertet als die $MSSIM_\alpha$. Die Werte der $MSSIM_\alpha$ sind nahe bei 1 und spiegeln das erwartete Ergebnis wider, denn beim texturierten 3D-Modell sind durch die vorgenommenen Modifikationen keine Änderungen sichtbar.

7.2.2 Evaluierung der 3D-Einflussgrößen

In diesem Abschnitt werden die Rauheit, Verzerrung und Flächenänderung als Einflussgrößen auf die Metrik evaluiert. Dabei wird gezeigt, dass das Konzept mit den zu erwartenden Ergebnissen übereinstimmt. Nachdem die drei Einflussgrößen im einzelnen evaluiert wurden, wird im nächsten Abschnitt die zusammengeführte Metrik aus Abschnitt 7.1.4 ausgewertet.

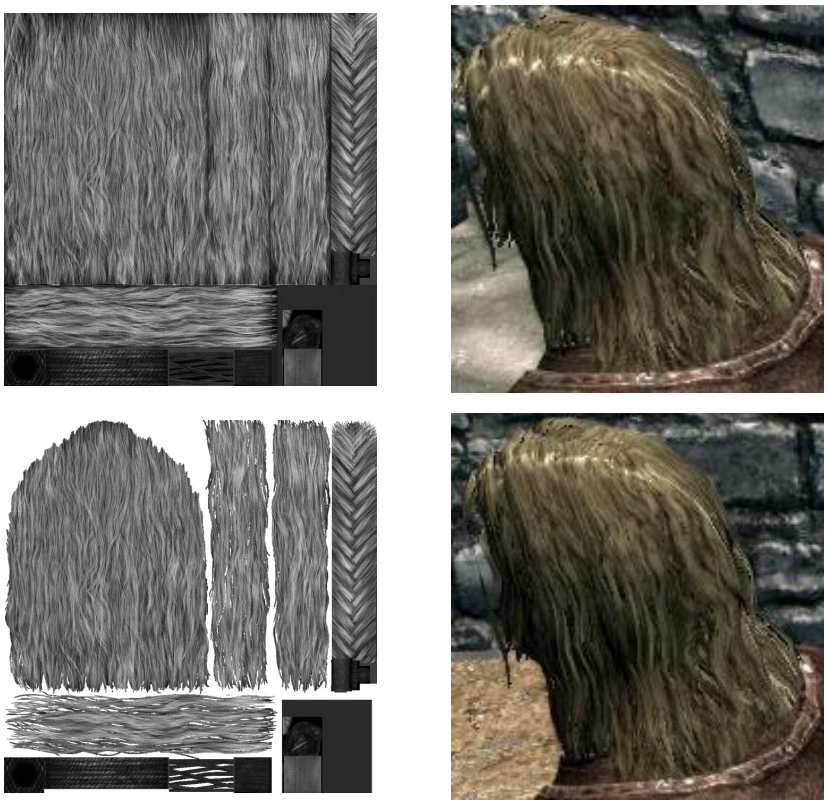


Abbildung 7.3: Textur “Hairlong“ aus SKYRIM , wobei links die Texturen und rechts die zugehörigen Spielszenen sind. Oben befindet sich die originale Textur, unten die modifizierte, jeweils als Graustufenbild.

Evaluierung der Rauheit Der Einfluss der Rauheit auf die Metrik wurde in Kapitel 7.1.3 erläutert. Die Evaluierung des Einflusses der Rauheit auf die visuelle Wahrnehmung erfolgt durch das Mappen der Texturen auf plane Flächen und das anschließende Bewerten mit sowohl der $MSSIM_\alpha$ als auch mit der Gesamtmetrik M. Es wird erwartet, dass beide Bewertungen sich nicht unterscheiden. Zu erwarten ist, dass die Bewertungen M zu 1 tendieren, je rauer das 3D-Modell ist und M tendiert zu dem Wert von $MSSIM_\alpha$ je planer ein Modell ist. Im Allgemeinen sind aber die Werte der Gesamtmetrik M größer im Vergleich zur $MSSIM_\alpha$. Es findet weder eine Verzerrung noch eine Flächenänderung statt und deshalb gehen wir davon aus, dass bei der Gesamtmetrik M nur die Rauheit eine Rolle spielt.

M_P	D_{M_P}	Var_{M_P}	M_R	D_{M_R}	Var_{M_R}
0.6592	0.0076	0.00038	0.6748	-0.0323	0.0088

Tabelle 7.3: Evaluierungsergebnisse zum Einfluss der Rauheit auf die Gesamtmetrik. Diese zeigt bei Veränderungen von Texturen, die zu rauen 3D-Modellen gehören, einen höheren Wert und damit einen geringeren Einfluss auf die visuelle Qualität.

Für die Evaluierung der Rauheit wurden händisch Hauswände und Mauern gesucht, die als plane Flächen dienen. Insgesamt konnten so aus der Testbasis 60 3D-Modelle ausgewählt werden. Es wurden weitere 60 Modelle aus der Testbasis ausgewählt, deren Summe über die Sprektralkoeffizienten ab dem Index 129 betragsmäßig am größten waren. Dies spiegelt die Modelle wider, die am rausten sind, siehe Kapitel 5.1. Die zugehörigen Texturen werden mit JPEG Stufe 10 komprimiert und auf die Modelle gemappt. Die originalen Texturen werden ebenfalls auf die 3D-Modelle gemappt. Die texturierten Modelle werden im der Bewertung durch die Metrik miteinander verglichen und daraus die qualitativen Unterschiede ausgegeben. Die Evaluierung wird über den Mittlwert der Differenzen D_M zwischen $MSSIM_\alpha$ und M und die Varianz Var_M der Differenzen zusammengefasst. Dabei sind D_{M_P} und Var_{M_P} die mittlere Differenz und Varianz der Differenzen bei planen Modellen und D_{M_R} und Var_{M_P} die Differenz und Varianz der Differenzen von rauen Modellen. Die Ergebnisse in Tabelle 7.3 zeigen, dass Veränderungen bei Texturen, die zu rauen 3D-Modellen gehören, einen höheren Wert und damit einen geringeren Einfluss auf die visuelle Qualität haben. Außerdem ist über die mittlere Differenz zu erkennen, dass die 2D-Metriken dieser Texturen, die zu rauen 3D-Modellen gehören, wesentlich schlechter bewertet wurden. Hieraus lässt sich erkennen, dass die Metrik die visuelle Wahrnehmung gut widerspiegelt.

Evaluierung der Flächenänderung und Verzerrung Zur Evaluierung des Einflusses der Flächenänderung auf die Gesamtmetrik werden die 3D-Modelle gestreckt. Das Strecken wird mit *3ds Max* von *Autodesk* [1] mit der Funktion *Stretch Modifier* auf der x-Achse durchgeführt. Dazu wurde "Stretch" zwischen $0,1$ und $0,4$ mit einer Schrittweite $0,05$ und "Amplify" mit $0,1$ gewählt und so die Größe der Mantelfläche verändert. Es wird erwartet, dass nach dem Vergrößern des 3D-Modells die Bewertung schlechter ausfällt, also kleiner ist im Vergleich zur 2D-Metrik. Für die Tests dienten alle 3D-Modelle aus der Testbasis. Die 3D-Modelle sind mit 3DS Max modifziert und mit Texturen texturiert worden, die mit einer JPEG Stufe 10 komprimiert und unter M_K zusammengefasst wurden. Die Evaluierung fand auf der gesamten Testbasis statt. Die Ergebnisse in Tabelle 7.4 bestätigen

$Stretch$	M_K	D_{M_K}	Var_{M_K}
0.1	0.7836	-0.0261	0.0066
0.15	0.7422	-0.0281	0.0084
0.2	0.7292	-0.0226	0.0058
0.25	0.7009	-0.034	0.0078
0.3	0.68	-0.0291	0.0053
0.35	0.6697	-0.0316	0.0069
0.4	0.757	-0.0309	0.0074

Tabelle 7.4: Evaluierungsergebnisse zum Einfluss der Flächenänderung und Verzerrung auf die Gesamtmetrik. Diese zeigt bei einer JPEG Kompression von Texturen, die auf verzerrte 3D-Modell gemapped wurde, einen höheren Wert und damit einen geringeren Einfluss auf die visuelle Qualität haben.

die Erwartung. Die Auswertung zeigt den Mittelwert der Metrik M_K, den Mittlwert der Differenzen D_{M_K} zwischen $MSSIM_\alpha$ und M_K und die Varianz Var_{M_K} der Differenzen.

Evaluierung der Gesamtmetrik Um die Gesamtmetrik aus Gleichung 7.8 als Ganzes zu bewerten, werden die 120 3D-Modelle aus dem vorherigen Abschnitt genutzt und zweimal mit dem in Kapitel 5 vorgestellten Wasserzeichen-Algorithmus versehen. Dabei wird die Parametrisierung bis auf G_{max} beibehalten. Die erste Markierung wird mit $G_{max} = 10$ und die zweite mit $G_{max} = 100$ durchgeführt, womit die Einbettung nach 10 bzw 100 Iterationen beendet wird. Bei geringerer Iterationszahl sind weniger Änderungen am Modell vorgenommen worden und somit ist eine bessere Bewertung zu erwarten als bei der Markierung mit einem größeren G_{max}. Bei der Markierung von 3D-Modellen kann es zu einer Zunahme oder Abnahme der Rauheit kommen, was aber weniger wahrscheinlich ist, da der Wasserzeichenprozess nur Änderungen im niedrigen Frequenzbereich vornimmt. Flächenänderungen und Verzerrungen werden durch den Wasserzeichenalgorithmus verursacht und haben daher einen dominierenden Einfluss in diesem Auswertungsschritt. Zur Markierung der Texturen wurde der in Kapitel 6 vorgestellte DDS-Wasserzeichenalgorithmus verwendet. Die Metrik M_{10} zeigt den Mittelwert der Bewertungen der texturierten 3D-Modelle, wobei $G_{max} = 10$ genutzt wurde und M_{100} umfasst den Mittelwert der Gesamtmetrik, nachdem bei der Einbettung nach 100 Iterationen die Einbettung beendet wurde. Tabelle 7.5 zeigt bessere Ergebnisse, wenn weniger Iterationen beim Einbetten genutzt wurden. Das entspricht der Erwartung, da mit weniger Iterationen auch weniger Änderungen durchgeführt wurden.

M_{10}	M_{100}
0.911	0.896

Tabelle 7.5: Evaluierungsergebnisse zum Einfluss der Wasserzeichenmarkierung von 3D-Modellen und DDS Texturen.

7.3 Zusammenfassung

Mithilfe der vorgestellten Metrik soll die Parametrisierung der Wasserzeichenalgorithmen für 3D-Modelle aus Kapitel 5 und Texturen aus Kapitel 6 vereinfacht werden, um die Nicht-Wahrnehmbarkeit der Wasserzeicheneinbettung zu bewerten, ohne dass aufwändige ABX-Tests notwendig werden. Die Metrik ist mit der Absicht entwickelt worden, dass sie die subjektive Wahrnehmung eines menschlichen Betrachters berücksichtigt. Sie erweitert und kombiniert eine Metrik aus dem 2D und 3D-Bereich und bezieht verschiedene messbare Größen wie die Rauheit, Flächenänderung und Verzerrung bei 3D-Modellen in ihre Berechnungen mit ein. Die als Basis genutzte 3D-Metrik von Wu et al. [107] berücksichtigt die Rauheit der 3D-Modelle zwar, aber sie wurde angepasst, indem die Struktureigenschaften mittels der Ausrichtung der benachbarten Dreiecke mit einbezogen wurden. Außerdem wurden Verzerrungen und Flächenänderungen, die durch das Markieren der Texturen und 3D-Modelle auftreten können, in die Bewertung mit einbezogen. Bei der Verzerrung wurden die Winkeländerungen zwischen der UV-Map und dem 3D-Modell herangezogen. Die Flächenänderung wurde durch die Beziehung der Mantelfläche des 3D-Modells zu den verwendeten Bereichen der Textur in der Metrik berücksichtigt.

Im Gegensatz zu bestehenden 2D-Metriken werden die nicht benötigten Bereiche einer Textur durch das Mapping bei der Bewertung auch nicht berücksichtigt. Dafür wurde die 2D-Metrik aus [103] als Basis verwendet und so erweitert, dass die Struktur-, Helligkeits- und Kontrastunterschiede berücksichtigt wurden. Außerdem wurde eine Gewichtung für den Alphakanal eingeführt. Damit lassen sich die transparenten Bereiche einer Textur herausrechnen. Die Evaluierungsergebnisse zeigen, dass die Anforderungen an die Metrik erfüllt wurden. Zur Feinabstimmung der Metrik auf die menschliche Wahrnehmung müsste eine breite Evaluierung mit Personen durchgeführt werden. Optimierungen sind durch Anpassen der Gewichtung der Einflussgrößen im 3D-Raum möglich.

8 Robustes Hashverfahren für 3D-Modelle

3D-Modelle werden immer wichtiger im Bereich der Fertigung und Erbringung von Dienstleistungen basierend auf 3D-Druckern. Urheber sind daran interessiert, einen Missbrauch ihres geistigen Eigentums verfolgen zu können.

Robuste Hashverfahren können zum schnellen Wiedererkennen von gleichem und ähnlichem Content verwendet werden. Selbst nach Veränderungen, z.B. durch Kompression oder andere typische Nachverarbeitungsschritte, ist Content wiedererkennbar. Im Gegensatz zu kryptographischen Hashverfahren basiert der robuste Hash nicht auf den einzelnen Bits des Contents, sondern auf dessen markanten Eigenschaften. Diese Eigenschaften werden in einer Bitfolge abgebildet.

In dieser Arbeit werden robuste 3D-Hashverfahren zum schnellen Auffinden eigener 3D-Modelle genutzt. Es ist wesentlich performanter, wenn ein robustes Hashverfahren zur Suche von eigenen 3D-Modellen im Internet genutzt wird anstatt des Detektors des 3D-Modell Wasserzeichens aus Kapitel 5.

Ein robuster Hash wird einem kryptografischen Hash vorgezogen, um selbst nach Veränderungen des 3D-Modells, z.B. durch Kompression oder Angriffe, es wiederzuerkennen. Metadaten werden als leicht manipulierbar angesehen und eigenen sich nicht zum verlässlichen Wiedererkennen.

In der Literatur finden sich verschiedene robuste Hashverfahren für 3D-Modelle. Ein erstes Verfahren wurde 2010 von Lee et al. [60] publiziert. Die Autoren legen mit ihrem Ansatz besonderen Fokus auf die Robustheit bei Skalierung und Euklidische Transformationen. Als Bezugsgrößen für den robusten Hash werden die Verteilung der Krümmung und die Blockintensität des Formindex des Modells herangezogen. Das Verfahren zeigt beim allen getesteten Angriffen schwächen, besonders wenn Rauschen hinzugefügt ist des Hash fragil. In [58] nutzen Lee und Kwon als Bezugsgröße normalisierte Feature-Objects, die nach ihren Distanzen gruppiert werden. Die Gruppen werden Schlüsselabhängig permutiert und daraus Feature Vektoren berechnet und binarisiert, um so zum robusten Hash zu gelangen. Mit diesem Ansatz wird zwar die Robustheit zur deren vorherigen Arbeit erreicht, aber beim hinzufügen von Rauschen ist bei leichtem Rauschen der Hash bereits zu stark beschädigt. Lee et al. [57, 61, 59] stellen einen weiteren Ansatz für einen robusten Hash vor. Das Verfahren basiert auf einer sogenannten Heat-Kernel-Signature, womit die

Autoren eine Multiskale Formkurve beschreiben und somit Robustheit bei Isometrischen Angriffen erreichen. Das Verfahren stellt eine Verbesserung zu den vorherigen Arbeiten dar, zeigt aber auch weiterhin schwächen nach hinzufügen von Rauschen. Salvador et al. [69, 68] basieren ihr robustes Hashverfahren auf der Fragmentierung des Modells, wobei die Fragmente sich überschneiden, um so ein ortsabhängiges Hashing zu erreichen. Das Verfahren zeigt gute Robustheit bei einigen Angriffen, hat jedoch schwächen nach hinzufügen von Rauschen und bei geringen Verzerrungen.

Im Folgenden wird ein robustes Hashverfahren für 3D-Modelle vorgestellt und evaluiert. Die Herausforderung bei einem robusten Hashverfahren liegt im Finden einer Balance zwischen einer hohen Wiedererkennungsrate, z.B. nach einem Nachverarbeitungsschritt oder Angriff, und gleichzeitiger Fragilität, um unterschiedliche 3D-Modelle voneinander abzugrenzen. Gemessen wird diese Eigenschaft mit der *Falschakzeptanzrate (FAR)* und der *Falschrückweisungrate (FRR)*. Anhand dieser beiden Raten wird der Algorithmus auf einem Testbed evaluiert, um eine geeignete Parametrisierung zu finden.

Das robuste Hashverfahren wurde im Umfang einer betreuten Bachelorarbeit von Rettig [78] umgesetzt.

8.1 Robustes Hashverfahren

Der folgende Algorithmus zum Extrahieren eines robusten Hash von 3D-Modellen, die als Polygonnetze repräsentiert werden, ist in vier Schritte untergliedert. Zuerst findet die spektrale Kompression statt, anschließend wird der Bezugspunkt bestimmt, worauf aufbauend die Bins kreiert werden, um im vierten Schritt den Hash zu extrahieren.

Mithilfe der spektralen Kompression wird ein Kernmodell erzeugt. Dazu werden die Eigenvektoren berechnet und anschließend die hohen Frequenzen entfernt (siehe Kapitel 5.1.1). Die Knoten des Kernmodells nutzt der Algorithmus zum Berechnen des Massezentrums C_g als stabilen Bezugspunkt. Ausgehend vom Bezugspunkt werden Bins erzeugt, indem Sphären mit unterschiedlichen Radien um das Massezentrum herum gebildet werden. Die Hashextraktion basiert auf dem Verhältnis der Anzahl der Knoten einzelner Bins zueinander. Dabei bezieht sich jedes Bit auf benachbarten Bins. Ein Überblick des Algorithmus ist in Abbildung 8.1 gegeben.

Der extrahierte robuste Hash wird mit den Metadaten des 3D-Modells in einer Datenbank hinterlegt.

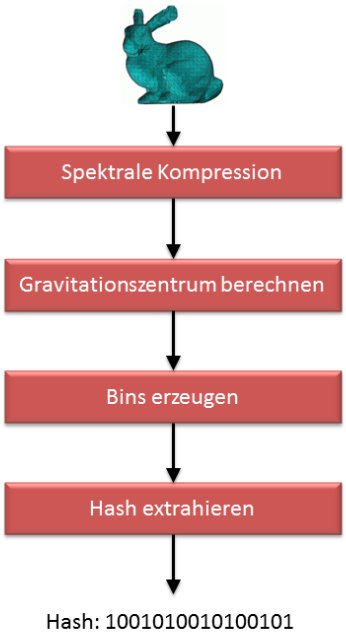


Abbildung 8.1: Überblick des robusten Hashverfahrens für 3D-Modelle

8.1.1 Erzeugen von Bins

Bevor die Bins erzeugt werden, ist es notwendig, die Knoten $v_i = (x_i, y_i, z_i) mit i \in \{1, ..., n'\}$ des Kernmodells M' von der kartesischen Koordinatenrepräsentation in die sphärische Koordinatenrepräsentation zu überführen. Damit werden die Knoten nicht mehr durch ihre x, y und z Koordinaten repräsentiert, sondern durch den Polarwinkel, den Azimut und den radialen Abstand zum Bezugspunkt. Für die Hashextraktion wird nur der radiale Abstand der Knoten zum Bezugspunkt verwendet und im Folgenden als Radius bezeichnet.

Der Radius $r(v_i)$ der Knoten $v_i = (x_i, y_i, z_i)$ zum Bezugspunkt $C = (x_c, y_c, z_c)$ wird durch die Euklidische Norm bestimmt:

$$r(v_i) = |v_i - C| = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2}$$

Anschließend werden alle Knoten in aufsteigender Reihenfolge entsprechend ihrem Abstand zum Bezugspunkt sortiert. Damit wird der Knoten, der den kleinsten Abstand

zum Bezugspunkt hat, mit v_1 bezeichnet und der mit dem größten Abstand wird mit $v_{n'}$ bezeichnet.

Es finden zwei unterschiedliche Bineinteilungen statt:

1. Die sortierten Radien werden in $N+1$ Sphären eingeteilt, wobei N die Hashlänge ist. Dazu werden $N+1$ Radien $S_1, ..., S_{N+1}$ errechnet. Der innerste Radius S_1 und der äußerste Radius S_{N+1} lassen sich wie folgt berechnen:

$$r(S_1) = \left\lfloor \frac{n'}{N+2} \right\rfloor \quad , \quad r(S_{N+1}) = \left\lceil N \cdot \frac{n'}{N+2} \right\rceil ,$$

   wobei n' die Anzahl der Knoten des Kernmodells ist, $\lfloor s \rfloor$ rundet zur nächsten kleineren ganzen Zahl und $\lceil s \rceil$ rundet zur nächsten größeren ganzen Zahl.

   Die verbleibenden $N-1$ Sphären $S_2, ..., S_N$ werden wie folgt definiert:

$$r(S_i) := r(S_1) + (i-1) \cdot d_r, mit i = \{2, ..., N\}$$

   wobei $d_r := \frac{r(S_{N+1}) - r(S_1)}{N}$ die Distanz zweier benachbarter Sphären ist. Damit lässt sich jeder Bin b_i durch die Sphären S_i und S_{i+1} mit $i \in \{1, ..., N\}$ beschreiben. Die Anzahl der Knoten, die in Bin b_i fallen, werden mit $|b_i|$ bezeichnet. Der Mittelwert eines Bins $\bar{b}_i$ wird wie folgt definiert:

$$\bar{b}_i := \frac{1}{|b_i|} \sum_{j=1}^{|b_i|} r(v_{i_j}) \quad \forall v_{i_j} \in b_i,$$

   wobei $\{v_{i_j}\}_{j=1}^{|b_i|}$ das Set der Knoten repräsentiert, die in b_i liegen. Das bedeutet, dass $\bar{b}_i$ der mittlere Abstand der Knoten eines Bins b_i zum Bezugspunkt C ist.

2. Die sortierten Radien werden in fünf Sphären $\hat{S}_1, ..., \hat{S}_5$ eingeteilt. Diese fünf Sphären werden durch die Radien $r(\hat{S}_{s+1}) := r(\hat{S}_s) + d_s$ bestimmt, wobei $r(\hat{S}_1) = r(S_1)$ gilt. Mithilfe dieser fünf Sphären werden vier übergeordnete Bins $\hat{b}_s$ erzeugt, mit $s = \{1, ..., 4\}$ und einer Breite $d_s := \frac{1}{4}(r(S_5) - r(S_1))$. Damit wird das übergeordnete Bin $\hat{b}_s$ durch die beiden Sphären $\hat{S}_s$ und $\hat{S}_{s+1}$ eingeschlossen. Die Anzahl der Knoten, die in einem übergeordneten Bin liegen, ist $|\hat{b}_s|$. Die Indizes i und s stehen wie folgt in Beziehung: $s = \lceil \frac{4 \cdot i}{N} \rceil$.

Abbildung 8.2 zeigt schematisch das Erzeugen von fünf Bins, die um den Bezugspunkt C liegen. Die Knoten, die innerhalb der inneren Sphäre und außerhalb der äußeren Sphäre liegen, werden beim Erzeugen der Bins nicht berücksichtigt. Diese Knoten sind meist zu instabil und eigenen sich nicht für die Generierung eines robusten Hashs.

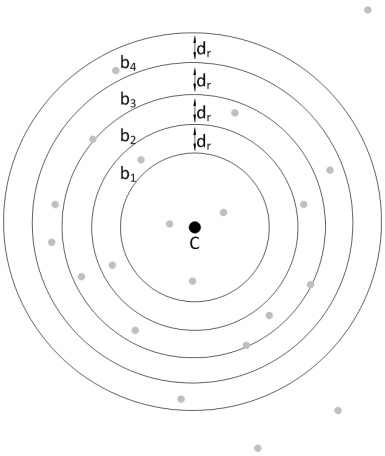


Abbildung 8.2: Bestimmen von fünf Sphären, um daraus vier Bins b_i zu erhalten, die eine Breite von d_r haben.

8.1.2 Hash Extraktion

Nachdem sowohl die N Bins und die vier übergeordnete Bins erstellt wurden, kann der Hash extrahiert werden. Ein binäres Hashbit H_i wird wie folgt bestimmt:

$$H_i = \begin{cases} 0 & \text{wenn } \bar{b}_i \leq \frac{4}{N} \cdot |\hat{b}_s| \\ 1 & \text{sonst} \end{cases}$$

8.2 Evaluierung

In diesem Abschnitt wird der vorgestellte robuste Hash für 3D-Modelle evaluiert. Dazu wird zuerst der Testaufbau beschrieben. Anschließend werden die Angriffe eingeführt, mit Hilfe derer die Evaluierung durchgeführt wird. Die Ergebnisse der Evaluierung werden so weit wie möglich mit den Ergebnissen von Lee et al. [60] verglichen und diskutiert. Die Komplexität, die Fehlerraten und die Eindeutigkeit des Verfahrens werden außerdem betrachtet. Abschließend werden die potentiellen Schwächen diskutiert und Verbesserungsvorschläge gegeben.

8.2.1 Hash Authentifizierung

Das Extrahieren des robusten Hash erfolgt wie in Kapitel 8.1 beschrieben. Die Hammingdistanz $d(H, H')$ wird zwischen dem extrahierten Hash H' und jedem Hash H in der Datenbank berechnet. Gilt $d(H, H') < \tau$, so unterscheiden sich zwei binäre Hashes um weniger als τ Bits und die zugehörigen 3D-Modelle werden als identisch gesehen bzw. das Modell mit dem Hash H' wird als bekannt angesehen. Diese Toleranz ermöglicht eine erhöhte Robustheit des Verfahrens bei gleichzeitig steigenden Fehlerraten.

Die Wahl von τ ist abhängig vom Anwendungsszenario und der Hashlänge. In dieser Arbeit wird der Schwellwert bei einer Hashlänge von 256 auf $\tau = 8$ gesetzt, bei der Hashlänge 128 auf $\tau = 4$ und bei der Hashlänge 16 auf $\tau = 1$.

8.2.2 Testaufbau

Das Testset besteht aus zwei Klassen: 1. bekannte und 2. unbekannte 3D-Modelle. Als bekannt wird ein 3D-Modell bezeichnet, dessen Hash in der Datenbank abgelegt wurde. Als unbekannt wird ein 3D-Modell bezeichnet, wenn sein robuster Hash nicht in der Datenbank abgelegt wurde.

In die Klasse der bekannten 3D-Modelle wurden 50 3D-Modelle aus dem Videospiel *Fallout III*, weitere 50 3D-Modelle aus dem 3D-Modell Archive *archive3D*[1] und die Stanford Modelle [2] aus dem *Stanford Computer Graphics Laboratory* gewählt. In die Klasse der unbekannten 3D-Modelle wurden ebenfalls 50 3D-Modelle aus den drei Bereichen gewählt, wobei sich die beiden Klassen der bekannten und unbekannten 3D-Modelle keine Modelle teilen.

Das Testset wurde aus zwei Gründen so gewählt. Zum einen variiert die Größe der 3D-Modelle mit der Anzahl der Knoten und Kanten stark, wie auch in Tabelle 8.1 zu sehen ist. Zum anderen ähneln einige der 3D-Modelle einem weiteren Modell im Testset. Einige der Modelle unterscheiden sich nur in wenigen Knoten und Kanten. Aus diesen beiden Gründen ist trotz der geringen Größe von jeweils 150 3D-Modellen das Testset praxisnah aufgebaut und die Ergebnisse belastbar.

Das Hashverfahren wird mit drei unterschiedlichen Hashlängen evaluiert: 16, 128 und 256 Bits. Es werden zu Beginn von jedem aus der Klasse der bekannten 3D-Modelle die robusten Hashes mit unterschiedlicher Länge extrahiert und in einer Datenbank abgelegt. Anschließend werden auf allen, sowohl bekannten als auch unbekannten 3D-Modellen, die Angriffe durchgeführt. Von den angegriffenen 3D-Modellen wird ebenfalls der robuste Hash extrahiert und die Hammingdistanz zu jedem in der Datenbank abgelegten Hash

[1] www.archive3d.net

3D-Modelle	Kleinstes	Größtes	Durchschnitt
Fallout III	120	8.309	4.180
Standford	34.834	466.597	250.715
Archive3D	32	163.826	28.424

Tabelle 8.1: Übersicht zur Anzahl der Knoten der im Testset verwendeten 3D-Modelle aus den unterschiedlichen Bereichen

berechnet. Unterschreitet die Hammingdistanz einen Schwellwert τ, so gilt das 3D-Modell als bekannt.

8.2.3 Angriffe

Damit die Evaluierung des vorgestellten Verfahrens mit der aus [60] vergleichbar ist, wird *3DS-Max* für die Durchführung der Angriffe verwendet. Die evaluierten Angriffe sind: Vereinfachung des Drahtgittermodells, Krümmen, Addieren von zufälligem Rauschen, Strecken, Verschieben und Tessellierung. Im Folgenden werden die Angriffe beschrieben, aber auf die Implementierung wird hier nicht näher eingegangen, da sie in [1] nachzulesen ist.

8.2.3.1 Vereinfachung des Drahtgittermodells

Zur Vereinfachung des Drahtgittermodells stellt 3ds Max die Funktion *MultiRes Modifier* bereit. Dabei wird das Drahtgitter vereinfacht, aber die globale Form des Modells beibehalten. Die Funktion wird über die folgenden Parameter gesteuert: *Vert Percent, Vertex Merging, Within Mesh: on, Threshold: 0.25*. Wie stark ein 3D-Modell vereinfacht werden soll, wird bei der Evaluierung prozentual festgelegt.

8.2.3.2 Krümmung

Zum Krümmen eines 3D-Modells wird die Funktion *Bend Modifier* bereitgestellt. Die zu wählenden Parameter sind *Angle* und *Direction*. Damit wird der Winkel gewählt, mit dem von der vertikalen Achse relativ zur horizontalen Achse das 3D-Modell gekrümmt wird.

8.2.3.3 Addieren von zufälligem Rauschen

Mit der Funktion *Noise Modifier* kann zufälliges Rauschen dem Modell hinzugefügt werden. Dabei lässt sich eine der drei Achsen x, y und z wählen, entlang der das Rauschen dem

Modell hinzugefügt wird. Die Funktion wird durch die Parameter *'Fractal: on'*, *Roughness* und *Strength* gesteuert, um die Intensität des Rauschens zu steuern.

8.2.3.4 Verschieben von Knoten

Die Funktion *Push Modifier* ermöglicht das Verschieben von Knoten und Kanten nach innen oder außen, ausgehend von der originalen Form des Modells. Dabei wird durch den Parameter *Push Value* die Distanz zum Zentrum des 3D-Modells bestimmt, die durch den Angriff erreicht werden soll.

8.2.3.5 Ungeleichmäßiges Skalieren

Die Funktion *Stretch Modifier* erlaubt das Strecken und Stauchen des 3D-Modells entlang einer Achse, deswegen wird dieser Angriff auch im Folgenden als *nicht-lineares Strecken* bezeichnet. Es wird entlang der beiden verbleibenden Achsen in die entgegengesetzte Richtung gestaucht oder gestreckt. Durch den Paramter *Stretch* wird das Strecken gesteuert. Der Angriff wirkt sich im Zentrum des Modells am stärksten aus, da mit einem festen Wert gestreckt oder gestaucht wird. Je weiter man vom Bezugspunkt weg geht, umso geringer ist der Effekt des Angriffs.

8.2.3.6 Tessellierung

Beim Tessellieren fügt 3ds max Knoten in die Mitte einer Kante ein und kreiert neue Kanten, um den neuen Knoten mit den umliegenden Knoten zu verbinden. Damit wird das 3D-Modell feiner aufgelöst. Die Funktion enthält den Parameter *Tension*, mit dem bestimmt werden kann, ob der neu hinzugefügte Knoten innerhalb oder außerhalb der originalen Form des Modells liegen soll. Somit kann die Oberflächenstruktur verändert werden.

8.2.3.7 Rotation, gleichmäßiges Skalieren und Translation

Diese Angriffe werden unter den gleichen Bedingungen wie in Kapitel 5.2.3 unter dem Abschnitt *Geometrische Transformationen* durchgeführt.

8.2.4 Evaluierung der Sicherheit, Performanz und Eindeutigkeit

In diesem Abschnitt wird das entwickelte robuste Hashverfahren gegen die im vorhergehenden Abschnitt beschriebenen Angriffe evaluiert. Außerdem wird neben der Wahrscheinlichkeit für Hashkollisionen auch die Verarbeitungsgeschwindigkeit pro 3D-Modell

evaluiert. Für alle Angriffe gilt, dass der robuste Hash etwa genauso sicher sein muss wie das 3D-Modell-Wasserzeichenverfahren. Denn das Ziel ist es, die geringe Kapazität des Wasserzeichenverfahrens durch den Hash dahingehend zu erweitern, dass der mögliche Nachrichtenraum für jedes Modell gilt. Damit ist das Wasserzeichen nur mit Hilfe des Hashes verwendbar.

In den Abbildungen 8.3, 8.4 und 8.5 werden die Evaluationsergebnisse für einen Teil der Angriffe gezeigt. Die Evaluierung der verbleibenden drei Angriffe, Rotation, gleichmäßiges Skalieren und Translation zeigt, dass die Hashes nach den Angriffen keine Fehler aufweisen und damit ist die normalisierte Hammingdistanz für diese drei Angriffe bei allen Hashlängen und Angriffsstärken Null. Das Verfahren ist damit sicher gegen diese Angriffe.

Bei der Evaluierung wurde die Hammingdistanz zwischen dem Hash aus dem originalen 3D-Modell mit dem Hash aus dem angegriffenen 3D-Modell berechnet und mit der Hashlänge normiert.

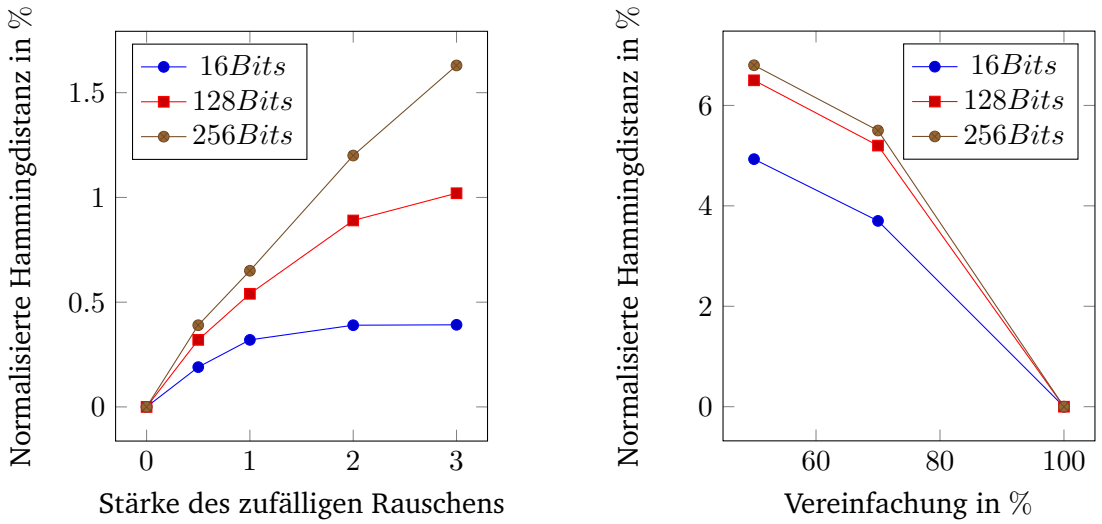


Abbildung 8.3: Links ist die normalisierte Hammingdistanz abhängig von der Stärke des addierten zufälligen Rauschens abgebildet. Rechts ist die normalisierte Hammingdistanz nach dem Vereinfachungsangriff abgebildet.

Die Ergebnisse zeigen Schwächen des vorgestellten robusten Hashverfahrens beim Krümmen und ungleichmäßigen Skalieren des 3D-Modells. Wie die Ergebnisse in Kapitel 5.2.3 zeigen, ist das Wasserzeichen nach diesen Angriffen ebenfalls nicht auslesbar. Im

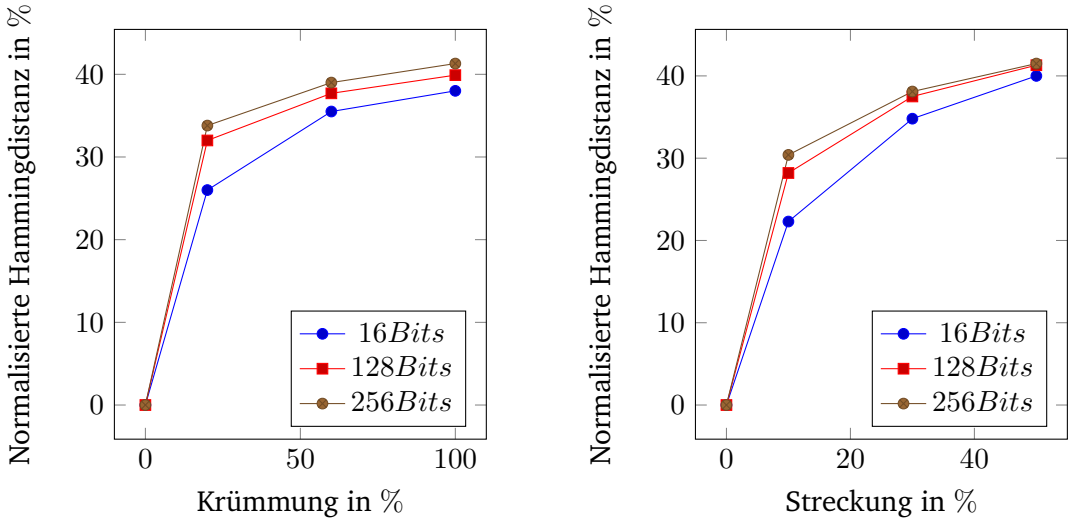


Abbildung 8.4: Links ist die normalisierte Hammingdistanz abhängig der prozentualen Krümmung abgebildet und rechts die normalisierte Hammingdistanz nach dem ungleichmäßigen Skalieren des 3D-Modells.

Folgenden werden die Auswirkungen der Angriffe auf das Modell aufgezeigt und diskutiert. Allgemein kann gesagt werden, dass der robuste Hash im Vergleich zum 3D-Modell-Wasserzeichenverfahren resistenter gegen die ausgeführten Angriffe ist. Ein Angriff, der z.B. zufälliges Rauschen der Stärke 3.0 auf das 3D-Modell addiert, stellt für das Hashverfahren keine Herausforderung dar.

Zur Veranschaulichung der Auswirkung des Rauschangriffs auf ein 3D-Modell zeigt Abbildung 8.6 das mit der Stärke 3.0 angegriffene *Smbehemoth* Modell aus dem Videospiel *Fallout III* im Vergleich zum ursprünglichen Modell. Die Form ist wiederzuerkennen, aber der Angriff bringt wahrnehmbare und störende Artefakte mit sich. Außerdem werden die Ausmaße des Krümmens in Abbildung 8.7 und die des ungleichmäßigen Skalierens in Abbildung 8.8 gezeigt. Die Krümmung wurde mit der Stärke 60 durchgeführt. Das 3D-Modell wurde beim nicht-linearen Strecken vertikal mit dem Faktor 0.5 gestreckt. Beide Angriffe bringen wahrnehmbare und störende Artefakte mit sich.

Im Umfang dieser Arbeit wird nicht weiter darauf eingegangen, bis zu welcher Stärke die beiden Angriffe vom Wasserzeichen und damit auch vom robusten Hash überstanden werden müssen. Das hängt individuell von den Anforderungen der Anwendung und den Auswirkungen für den Nutzer ab.

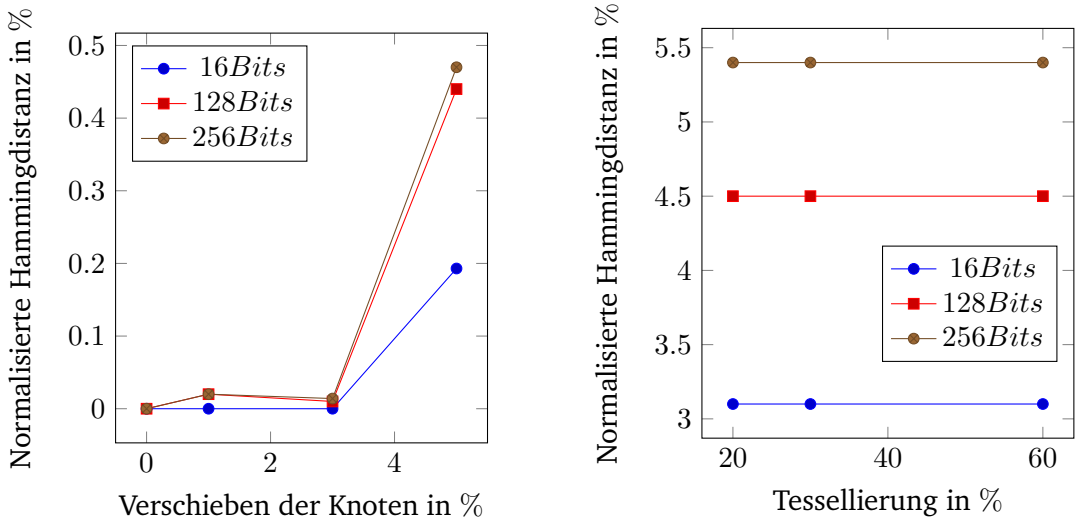


Abbildung 8.5: Links ist die normalisierte Hammingdistanz abhängig der Streckung in einer Dimension abgebildet und rechts die normalisierte Hammingdistanz nach der Tessellierung, wobei sich die prozentuale Angabe auf die Anzahl der Knoten des originalen Modells bezieht.

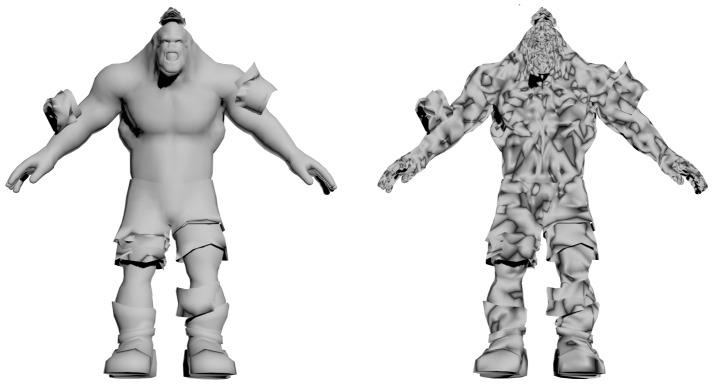


Abbildung 8.6: Links, das originale 3D-Modell *Smbehemoth* des Videospiels Fallout III und rechts, das mit zufälligem Rauschen der Stärke 3.0 angegriffene Modell.

Wird das Modell nur gering vereinfacht und fällt dabei nur ein kleiner Anteil der Knoten weg, so sind keine Artefakte sichtbar und das Hashverfahren ist resistent dagegen.

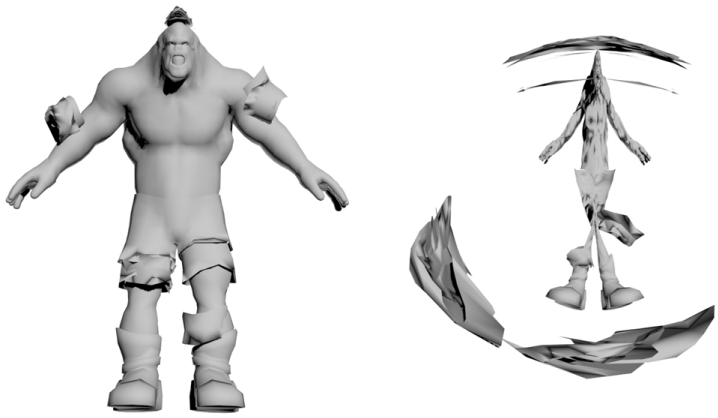


Abbildung 8.7: Links, das originale 3D-Modell *Smbehemoth* des Videospiels Fallout III und rechts, das mit der Intensität 60 gekrümmte 3D-Modell.

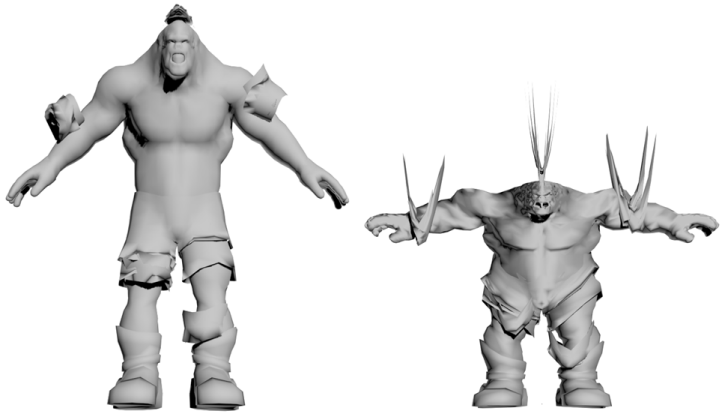


Abbildung 8.8: Links, das originale 3D-Modell *Smbehemoth* des Videospiels Fallout III und rechts, das um den Faktor 0.5 gestreckte 3D-Modell.

Beim Vergleichen der Sicherheit des 3D-Modell-Wasserzeichenverfahrens mit dem des robusten Hashs nach dem Vereinfachungsangriff ist hervorzuheben, dass beim Wasserzeichenverfahren über 28% der Wasserzeichenbits fehlerhaft waren, nachdem 5% der Knoten wegfielen, wobei beim robusten Hash dagegen weniger als 2% der Hashbits fehlerhaft waren. Das vorgestellte robuste Hashverfahren zeigt eine hohe Sicherheit gegen den Vereinfachungsangriff.

Das vorgestellte robuste Hashverfahren konnte nach keinem Krümmungsangriff ein Modell über den robusten Hash wiedererkennen. Abbildung 8.7 zeigt die Auswirkung eines mit der Intensität 60 gekrümmten 3D-Modells. Dieser Angriff bringt sehr starke und störende Artefakte für die visuelle Qualität mit sich. In dieser Arbeit wird nicht weiter verfolgt, bis zu welcher Intensität des Angriffs die 3D-Modelle in ihrer Qualität akzeptable sind, da bei jeder getesteten Intensität die visuelle Qualität störend wahrnehmbar war und davon auszugehen ist, dass ein gekrümmtes 3D-Modell sich nicht zur Weiterverbreitung eignet.

Die gleiche Argumentation wie beim Krümmungsangriff gilt auch für das nicht-lineare Strecken und Stauchen der 3D-Modelle. Auch bei diesem Angriff konnte nach dem Strecken kein Modell über seinen robusten Hash wiedererkannt werden. Auch das 3D-Modell-Wasserzeichen ist gegen diesen Angriff nicht sicher, siehe Kapitel 5.2.3. Wie jedoch Abbildung 8.8 zeigt, bringt der Angriff wahrnehmbare und störende Artefakte mit sich und ist mit dieser Parametrisierung nicht für einen Angriff praktikabel. Würden die Artefakte händisch eliminiert, würde der vorgestellte robuste Hash nicht robust gegen diesen Angriff sein, da die Knoten nicht linear verschoben und damit die Hashbits kippen werden.

Den Evaluierungsergebnissen nach stellt das Verschieben der Knoten keine Herausforderung für den robusten Hash dar, siehe Abbildung 8.5. Der Angriff verändert die Grundform des Modells nicht und es entstehen auch keine wahrnehmbare Artefakte.

Bei der Tessellierung ist die Sicherheit negativ korreliert mit der Hashlänge, nämlich je kürzer desto sicherer. Der Angriff zerstört weder die Grundform des Modells, noch erzeugt er wahrnehmbare Artefakte. Deshalb ist es notwendig, dass der robuste Hash resistent gegen diesen Angriff ist. Daraus kann abgeleitet werden, dass z.B. bei einer Hashlänge von 256 Bits der Schwellwert τ den Fehler von etwa 5.5% berücksichtigt und dementsprechend $\tau = 15$ gewählt werden sollte.

Allgemein lässt sich der Zusammenhang erkennen, dass kürzere Hashlängen eine höhere Sicherheit mit sich bringen als längere Hashes.

8.2.4.1 Performanz

Nachdem die Sicherheit des robusten Hashs evaluiert und diskutiert wurde, ist in diesem Abschnitt die Geschwindigkeit des vorgestellten Verfahrens zu evaluieren. Für die Eva-

luierung der Laufzeit wurde der Code auf einem PC mit Windows 7 (SP1, 64-Bit), 3.00 GHz Intel Core2 Duo Prozessor und 4GB Ram ausgeführt. Dabei benötigte das robuste Hashverfahren für ein durchschnittliches 3D-Modell mit ≈ 24.000 Knoten etwa 2.8 Sekunden. Der Code lässt noch einige Vereinfachungen und Parallelisierungen zu, wodurch die Laufzeit verkürzt werden kann.

8.2.4.2 Eindeutigkeit

Zur Auswertung der Eindeutigkeit eines robusten Hashes, die auch die Gegenwahrscheinlichkeit zur Kollision darstellt, wird die Unabhängigkeit der Bits entsprechend ihrer Positionen ausgewertet. Diese Methode wird gewählt, da es hier aufgrund des kleinen Testsets an 3D-Modellen nicht möglich ist die Kollisionen direkt zu ermitteln. Damit die Wahrscheinlichkeit für eine Kollision gering wird, ist es notwendig, dass die Wahrscheinlichkeit für eine "0" oder "1" für jede Bitposition gleich ist, also im optimalen Fall 50% entspricht. Das vorgestellte Verfahren erreicht 47% und zeigt damit, dass die Wahrscheinlichkeit für Kollisionen sehr gering ist.

8.2.5 Fehlerraten

Da der robuste Hash tolerant gegen Veränderungen sein soll, die durch die Transformationen oder Angriffe verursacht werden, können zwei unterschiedliche Arten von Fehlern auftreten, nämlich die Falschakzeptanz- (FAR) und Falschrückweisungsrate (FRR). Die FAR wird gemessen, indem die von unbekannten 3D-Modellen extrahierten, robusten Hashes gegen die aus bekannten 3D-Modellen extrahierten, robusten Hashes verglichen werden. Die FAR repräsentiert dabei die Wahrscheinlichkeit, dass ein Hash von einem unbekannten 3D-Modell einem Hash eines bekannten 3D-Modells entspricht. Die FRR hingegen misst die Wahrscheinlichkeit, dass ein aus einem bekannten 3D-Modell extrahierter Hash, verglichen mit den bereits bekannten Hashes, als unbekannt identifiziert wird. Zum Evaluieren der Fehlerraten werden die bekannten 3D-Modelle durch Angriffe manipuliert.

Für die Tests wurden die 3D-Modelle in zwei Gruppen geteilt, wobei 100 als bekannt und 50 als unbekannt galten. Die bekannten 3D-Modelle wurden außerdem mit den in 8.2.3 beschriebenen Angriffen manipuliert. Anschließend wurden die Hashes aus den Modellen extrahiert und miteinander verglichen, um daraus die mittleren normierten Fehlerraten zu berechnen.

Beide Fehlerraten sollen möglichst klein sein, wobei die FRR am besten "0" sein sollte. Bei den Tests zur Bestimmung der Fehlerrate wurde bei der Hashlänge von $N = 256$ der Schwellwert mit $\tau = 17$ gewählt. Ist die Hammingdistanz zweier Hashes kleiner als τ,

so gelten sie als identisch. Damit kann eine FAR von 5.7% und eine FRR von 0% erreicht werden.

8.3 Zusammenfassung

Das in diesem Kapitel vorgestellte robuste Hashverfahren eignet sich, um eine Einschätzung zu erhalten, ob ein 3D-Modell bereits bekannt ist. Der robuste Hash zeigt bei allen getesteten Angriffen eine höhere Sicherheit als das Wasserzeichenverfahren.

Das Hashverfahren ermittelt einen robusten Bezugspunkt und transformiert mithilfe dessen die Knoten in ihre sphärische Koordinatenrepräsentation. Anschließend werden Sphären definiert und daraus Bins erzeugt. Jedes Bin wird zu seinem übergeordneten Bin ins Verhältnis gesetzt und daraus ein Hashbit extrahiert. Es wurden drei unterschiedliche Hashlängen evaluiert: 16, 128 und 256 Bits.

Die Evaluierungsergebnisse zeigen, dass das Verfahren invariant gegen die drei Angriffe Rotation, gleichmäßige Skalierung und Translation ist. Außerdem kann bei einer Hashlänge von 256 Bits mit einem Schwellwert von $\tau = 17$ die Sicherheit gegen das Addieren von zufälligem Rauschen, Vereinfachen, Verschieben und Tessellieren erreicht werden. Dabei ist die $FRR = 0$ und die $FAR = 5.7\%$. Die Hashbits sind unabhängig voneinander und die Wahrscheinlichkeit für eine “0“ oder “1“ für eine Bitposition ist 47%. Damit sind Hashkollisionen eher unwahrscheinlich.

Inwieweit die Schwächen des Verfahrens bei der Krümmung und ungleichmäßigen Skalierung relevant sind, muss abhängig vom Einfluss auf die Nicht-Wahrnehmbarkeit bei der jeweiligen Applikation evaluiert werden. Die beiden Angriffe verändern die Grundform der 3D-Modelle und hinterlassen wahrnehmbare Artefakte. Der Vergleich der Evaluationsergebnisse aus diesem Kapitel mit jenen von Lee et al. [60] zeigt, dass der hier vorgestellte Algorithmus eine wesentlich höhere Sicherheit gegen die getesteten Angriffe hat. Bei den beiden Angriffen Krümmung und ungleichmäßiges Skalieren schneiden jedoch beide Algorithmen schlecht ab.

9 Videowasserzeichen

In diesem Kapitel wird das entwickelte Videowasserzeichen vorgestellt, welches auf den nicht komprimierten Videodaten Modifikationen durchführt. In der Literatur existieren verschiedene Videowasserzeichenverfahren, die in der räumlichen Domain einbetten [50, 56, 53], oder das Frequenzspektrum modifizieren und dabei unterschiedliche transformationen nutzen [4, 93, 81]. Die in Kapitel 3.6 aufgezeigten Anforderungen werden durch die existierenden Verfahren nicht vollständig erfüllt. Das bekannteste Videowasserzeichenverfahren wurde von Kalker et al. [50] publiziert und weist eine Sicherheit gegen Abfilmen auf. Jedoch ist es nicht sehr robust wenn das Video skaliert, rotiert oder desynchronisiert wurde. Es gibt ein kommerzielles Verfahren von Civolution[1], ehemals CineFence, von Philips[2], das nach Angaben des Unternehmens in der Filmindustrie eingesetzt wird, aber dessen Funktionsweise und damit auch die Sicherheit, Robustheit und Nicht-Wahrnehmbarkeit nicht weiter untersucht werden konnten. Zwei weitere Verfahren, die sicher gegen das Abfilmen sein sollen, sind Coded Anti-Piracy CAP von Kodak und das gleichnamige Produkt von Deluxe Laboratories. Bei beiden Verfahren wird kritisiert, dass das Wasserzeichen wahrnehmbar ist. Die Wasserzeichen von Deluxe wurden sogar teilweise als störend wahrgenommen und werden nicht mehr eingesetzt. Die Sicherheit von CAP ist nicht sehr hoch und damit ist das Wasserzeichen leicht zu löschen.

Damit sind die Anforderungen aus Kapitel 3.6 nicht vollständig adressiert. Das hier vorgestellte Verfahren legt den Fokus auf die Nicht-Wahrnehmbarkeit bei gleichzeitiger Sicherheit und Robustheit des Wasserzeichens.

Der vorgestellte Algorithmus bettet das Wasserzeichen in der räumlichen Domain ein. Er sucht zuerst die Maximal Stabile Extrem-Regionen (MSER) [70] in jedem Bild des Videos und modifiziert die Regionen schlüsselabhängig mithilfe von vordefinierten Mustern. Am Ende des Kapitels erfolgt die Evaluierung vor dem Hintergrund verschiedener Angriffe und eine Diskussion der Ergebnisse. Zur Evaluierung der Sicherheit des Wasserzeichens gegen Abfilmen wird neben Smartphones auch CamMark [82] genutzt. Es simuliert das Abfilmen der Testvideos unter Berücksichtigung unterschiedlicher Eigenschaften.

[1] http://www.civolution.com/about-us/audio-video-watermarking-and-fingerprinting/video-watermarking-products/

[2] http://www.business-sites.philips.com/shared/assets/global/Downloadablefile/CineFence-13275.pdf

Die Grundlagen des Videowasserzeichenverfahres wurden im Umfang zweier betreuter Abschlussarbeiten von Senker [91] und Schildknecht [89] entwickelt. Das hier vorgestellte Verfahren ist eine Kombination und Erweiterung der beiden Verfahren, um bessere Eigenschaften in der Robustheit, Sicherheit und Transparenz zu erhalten und wurde publiziert in [12]. Zur Einbettung des Wasserzeichens benötigt der Algorithmus das zu schützende Video als nicht komprimierten Datenstrom, die Wasserzeicheninformation als binäre Sequenz, den Schlüssel zum sicheren Verstecken der Nachricht und verschiedene weitere Parameter zum Steuern der Transparenz und Datenrate.

9.1 Videowasserzeichen Algorithmus

In diesem Abschnitt wird zuerst der Algorithmus im Groben skizziert, bevor die Schritte im Detail beschrieben werden.

Der Videowasserzeichenalgorithmus teilt sich in zwei Schritte auf, die Einbettung und die Detektion. Bei der Einbettung wird das Wasserzeichen in das Video eingebettet und bei der Detektion wird das Wasserzeichen aus dem Video extrahiert. Die beiden Teile des Algorithmus ähneln sich. Zuerst wird die Einbettung beschrieben, danach die Detektion.

Der Algorithmus erzeugt schlüsselabhängig kreisförmige Wasserzeichenmuster in unterschiedlichen Größen, die er für die Einbettung benötigt. Die Videos werden im Xvid-Codec erwartet. Das Video wird im ersten Schritt dekomprimiert, sodass der Algorithmus auf jedes Frame zugreifen kann. Die Frames des Videos werden einzeln analysiert und zum Einbetten geeignete Bereiche extrahiert. Als geeignete Bereiche werden maximal stabile Extrem-Regionen (MSER) angesehen. MSER besitzen die Eigenschaft, dass sie selbst nach Veränderungen des Videos immer wieder gefunden werden können. Jede gefundene Region wird mit der kleinstmöglichen umschließenden Ellipse approximiert. Die Ellipse wird entlang ihrer Hauptachsen um einen festen Faktor skaliert. Überschneiden sich Ellipsen, wird nur die Ellipse gewählt, die die stabilere MSER approximiert.

In die ausgewählten Ellipsen bettet der Algorithmus das Wasserzeichen ein, indem die schlüsselabhängigen kreisförmigen Wasserzeichenmuster addiert werden. Das Wasserzeichenmuster, das von der Größe am besten zu der Region passt, wird entsprechend auf diese Ellipse skaliert. Abhängig der einzubettenden Wasserzeichenbit wird das Wasserzeichenmuster addiert. Soll eine binäre “1“ eingebettet werden, so wird das skalierte Wasserzeichenmuster addiert, bei einer “0“ wird das inverse Muster auf die Region addiert. Addiert wird nur in den Bereichen außerhalb der MSER und innerhalb der Ellipse. Somit werden die Helligkeitswerte der Regionen modifiziert. Zur Wahrung der Transparenz wird ein Laplace Hochpassfilter in Kombination mit einer Szeneerkennung verwendet. Der Laplace Hochpassfilter steuert die Intensität jeder Änderung innerhalb einer Region.

Die Szeneerkennung verhindert, dass innerhalb einer Szene des Videos die Regionen unterschiedlich in ihrer Intensität verändert werden, um den Flickering-Effekt [6] zu unterbinden. Abbildung 9.1 skizziert den groben Ablauf des Algorithmus.

Beim Detektieren extrahiert der Algorithmus die MSER der Frames, approximiert die MSER mit der kleinstmöglichen Ellipse und skaliert die Ellipse entlang ihrer Hauptachsen. Bei sich überschneidenden Ellipsen wird wieder nur die Ellipse ausgewählt, die die stabilste MSER approximiert, um so die bei der Einbettung modifizierten Regionen zu erhalten. Dann werden die einzelnen Regionen analysiert. Bei der Analyse bestimmt der Detektionsalgorithmus die Korrelation zwischen dem Muster, das zur Einbettung genutzt wurde, und der Ellipse. Dazu werden die Helligkeitswerte der Regionen der Ellipse, die im Muster positiv sind, ebenso wie die Regionen, die im Muster negativ sind, getrennt voneinander aufsummiert und jeweils durch ihre Anzahl dividiert. Abhängig vom Verhältnis der beiden normalisierten Werte entscheidet der Algorithmus, ob es sich um eine “0“ oder “1“ handelt.

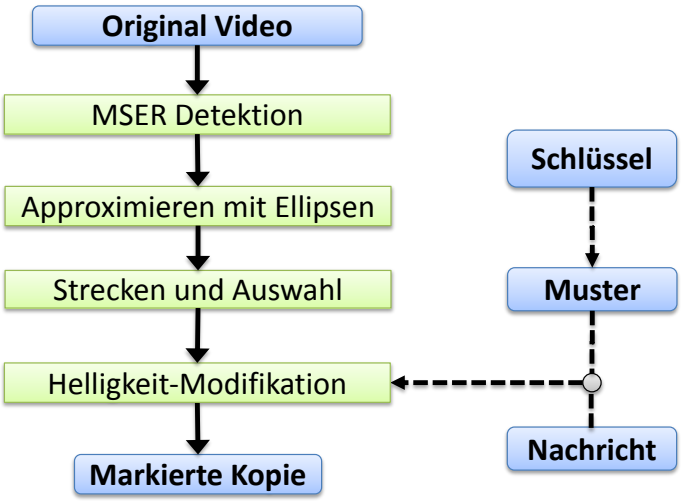


Abbildung 9.1: Überblick des vorgestellten Videowasserzeichenalgorithmus

9.1.1 Kreisförmiges Wasserzeichenmuster

Der Algorithmus erstellt zu Beginn ein kreisförmiges Muster. Ein Kreis ist eine Spezialform einer Ellipse, bei der beide Hauptachsen gleich lang sind und daraus werden beliebige Ellipsen abgeleitet. Das Kreismuster wird abhängig von einem zu Beginn gewählten

Schlüssel zufällig mit den Werten $-1, 1$ erzeugt. Abbildung 9.2 zeigt beispielhaft ein erzeugtes Urmuster, wobei die Werte verstärkt und auf die Skala von $[0, 255]$ abgebildet wurden. Zum Erhöhen der Robustheit werden die Muster in unterschiedliche Größen mit dem Durchmesser, $16, 32, 64, 128, 256$ Pixel, skaliert und die hohen Frequenzen im Muster eliminiert. Dazu wird das Muster mit der Fouriertransformation in den Frequenzraum transformiert und die Frequenzen, die größer als der Schwellwert τ_m sind, auf $'0'$ gesetzt und anschließend mit der inversen Fouriertransformation zurück in den Bildbereich transformiert und auf $-1, 1$ gesetzt. In Abbildung 9.2 sind beispielhaft ein Urmuster und die abgeleiteten Muster mit einem Durchmesser von 128 und 256 dargestellt, wobei zur Visualisierung die Werte für -1 auf 0 und für 1 auf 128 gesetzt wurden.

Das Muster, das vom Durchmesser am nächsten an der größeren Hauptachse der Ellipse liegt, wird für das Einbetten auf die entsprechende Größe skaliert. Bei der Einbettung einer "1" wird das Muster auf die Intensität der Region addiert, beim Einbetten einer "0" wird das Inverse des Musters addiert.

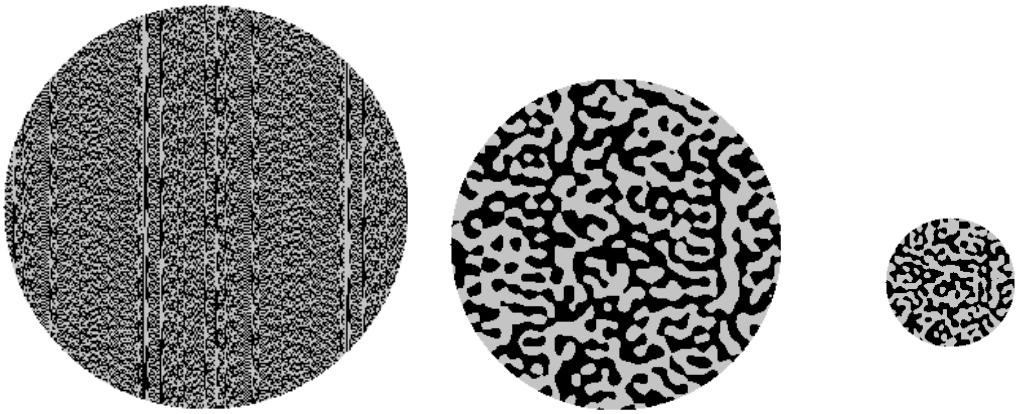


Abbildung 9.2: Beispielhafte Muster, die zum Einbetten eines Wasserzeichens in eine MSER eines Videoframes genutzt werden. Links das Urmuster, in der Mitte das daraus abgeleitete Muster mit dem Durchmesser von 256Pixel und rechts das aus dem Urmuster abgeleitete Muster mit dem Durchmesser von 128 Pixel

9.1.2 Analyse der Frames

Das Ziel der Analyse ist es, in den Frames Bereiche zu finden, die sich zur Einbettung eignen. In diese Bereiche wird dann das Wasserzeichen eingebettet. Im ersten Schritt der Analyse werden die im Frame enthaltenen MSER detektiert. Anschließend wird um jede MSER die kleinstmögliche Ellipse bestimmt, die die MSER vollständig umschließt. Das Bestimmen der kleinstmöglichen Ellipse und ihrer Ausrichtung wird im Zusammenhang mit dem Modifizieren der MSER Detektion in Kapitel 9.1.3 beschrieben. Die Ausrichtung der Ellipse wird im Wesentlichen anhand einer speziellen Regressionsgeraden durch die Punkte der MSER bestimmt. Die Ausrichtung der Ellipse führt zur Invarianz des Wasserzeichens gegenüber Rotationen. Durch die Ellipsenform ist der Algorithmus invariant gegenüber nicht linearer Skalierung.

Es werden nur die Ellipsen genutzt, die vollständig im Frame enthalten sind. Die nicht vollständig im Frame liegenden Ellipsen, die an den Rändern des Frames liegen, werden nicht weiter berücksichtigt. Grundsätzlich ist die Annahme, dass Bereiche, die an den Rändern des Frames liegen, wenig bis keine Relevanz für das Video haben. Außerdem sind die Bildränder nach dem Abfilmen auch nicht identisch zum Original.

Treten Überschneidungen der Ellipsen auf, so wird nur die Ellipse genutzt, die die stabilste MSER approximiert. Die stabilste MSER ist die, die die geringste Varianz besitzt. Würden alle genutzt, bedeutet das für das Wasserzeichen, dass mehrere Muster übereinander addiert werden und sich verstärken oder auslöschen würden. Die Transparenz und Robustheit des Wasserzeichens würden dadurch leiden. In Kapitel 9.1.4 ist die Auswahl der stabileren MSER aus dem Set der detektierten MSER im Detail beschrieben.

In Abbildung 9.3 sind die einzelnen Schritte an einem Beispiel veranschaulicht. Dafür wurde das Bild "Lena"verwendet (a). Es werden die MSER detektiert (b) und die Ellipsen um die einzelnen MSER gelegt (c). Anschließend filtert der Algorithmus nur solche Ellipsen, die vollständig im Bild vorhanden sind. Bei den überlappenden Ellipsen werden nur solche ausgewählt, die in diesen Bereichen am stabilsten sind (d). Bevor das Muster addiert werden kann, wird die Ausrichtung bestimmt (e).

9.1.3 Angepasste MSER Detektion

Maximal stabile Extrem-Regionen (MSER) wurden von Matas et al. [70] eingeführt. Die Regionen sind invariant gegenüber kontinuierlichen Transformationen der Bildkoordinaten oder monotonen Transformationen der Bildintensität, wie z.B. Rotation, Skalierung, Verzerrung, Helligkeitsänderung. Damit lassen sich die MSER nach diesen Transformationen wieder finden; z.B. nach einer Rotation wird die rotierte MSER wieder gefunden. Matas et al. nutzen die Eigenschaften der MSER als Hilfsmittel, um dasselbe Objekt in

(a) Beispielbild (b) MSER Detektion (c) Einprägungsbereiche bilden

(d) Einige Bereiche auswählen (e) Ausrichtungen bestimmen

Abbildung 9.3: Beispielhafte Analyse eines Frames anhand dem Bild "Lena".

Bildern aus unterschiedlichen Perspektiven wiederzuerkennen. Die Invarianz der MSER gegenüber einer Vielzahl an Transformationen macht sie für weitaus mehr als diese Anwendungen interessant. In dem hier beschriebenen Videowasserzeichenverfahren bilden MSER die Grundlage. Das Verfahren zum Finden der MSER wurde von Nister und Stewenius [71] genau beschrieben. Die Bezeichnung maximal stabile Extremregion erweckt den Eindruck, dass jede gefundene MSER immer eine hohe Stabilität aufweist. Das muss nicht der Fall sein, denn mit dem Algorithmus wird lediglich sichergestellt, dass die als MSER detektierten Regionen stabiler sind als die nächst größere oder kleinere Extremregion.

Zur Detektion der MSER wurde auf die OpenCV[3] Bibliothek zurückgegriffen und eigene Anpassungen durchgeführt. Die Detektion wurde so modifiziert, dass sie der ursprüngli-

[3] OpenCV (Open Source Computer Vision), http://opencv.org

chen Definition aus [70] entspricht. Findet der Algorithmus eine MSER, die vollständig von einer größeren MSER umschlossen ist, wobei die umschließende MSER eine größere Varianz hat als die darin liegende, dann enthält die Ergebnismenge die umschließende MSER nicht. Hat jedoch die umschließende MSER eine geringere Varianz, so werden beide MSER in der Ergebnismenge aufgenommen. Eine geringe Varianz impliziert eine höhere Stabilität. Die zuletzt genannte Erweiterung wurde vorgenommen, um den Rechenaufwand zu verringern.

Parameterwahl für die MSER Extraktion Zur Extraktion der MSERs wurden die Parameter wie folgt gewählt:

- $\delta = 4$
- minimale Größe: 0.1%
- maximale Größe: 10%
- maximale Varianz: 0.4
- minimale Verschiedenheit: 0.5

Mit diesen Parametern wird konfiguriert, welche der detektierten MSER in die Ergebnismenge mit aufgenommen werden. Die Parameter beschreiben folgende Eigenschaften:

- δ: Der Parameter dient zur Klassifizierung in stabile und nicht stablie Extrem-Region. Sei dazu $Q_1, Q_2, ..., Q_n$ eine stetig wachsende Folge von ineinander verschachtelten minimalen oder maximalen Extrem-Regionen mit $Q_1 \subset Q_2 \subset ... \subset Q_n$, dann ist Q_i eine maximal stabile Extrem-Region, wenn gilt: $q'(i) = 0$ und $q''(i) > 0$ mit $q(i) = \frac{|\frac{Q_{i+\delta}}{Q_{i-\delta}}|}{|Q_i|}$.
- minimale Größe: MSER, die kleiner als dieser Schwellwert sind, werden nicht in die Ergebnismenge aufgenommen. Der Parameter bezieht sich in der vorliegenden Implementierung prozentual auf die Größe des Bildes.
- maximale Größe: MSER, die größer als dieser Schwellwert sind, werden nicht in die Ergebnismenge aufgenommen. Der Parameter bezieht sich ebenfalls prozentual auf die Größe des Bildes.
- maximale Varianz: Ist die Varianz einer MSER größer als dieser Schwellwert, wird die MSER nicht in die Ergebnismenge aufgenommen. Dieser Parameter sichert, dass nur solche MSERs aufgenommen werden, die stabil genug sind.

- minimale Verschiedenheit: Mit diesem Parameter wird verhindert, dass in der Ergebnismenge zu ähnliche MSER aufgenommen werden. Es gibt Bilder, die ineinander verschachtelte MSER enthalten. Einige davon unterscheiden sich nur um einige Pixel. Nur dann, wenn sich zwei ineinander verschachtelte MSER um mindestens die Größe des Parameters unterscheiden, werden beide in die Ergebnismenge aufgenommen.

9.1.4 Auswahl der Ellipsen

Nachdem die MSER extrahiert wurden, wird jeweils die kleinstmögliche Ellipse um die Region bestimmt und ihre Hauptachsen werden mit einem festen Faktor gestreckt. Es gibt zwei Gründe, weshalb der Radius vergrößert wird. Zum einen erreicht der Algorithmus eine bessere Ausnutzung des Trägermaterials. Zum anderen wird dadurch ermöglicht, dass die MSER bei der Einbettung verstärkt werden kann und nur der Bereich außerhalb der MSER zum Einbetten des Wasserzeichens genutzt wird. Wird die MSER bei der Einbettung verstärkt, so ist die Wahrscheinlichkeit größer, sie beim Detektieren des Wasserzeichens genau so wieder zu finden. Bei der Wahl des Vergrößerungsfaktors ist zu beachten, dass die Bereiche nicht zu groß werden, da sich sonst die Wahrscheinlichkeit erhöht, dass sich mehr und mehr Bereiche überschneiden oder gar das ganze Frame zu einem Einbettungsbereich wird. In dieser Arbeit wird ein Vergrößerungsfaktor von 1.4 gewählt, da er sich als gut geeignet herausgestellt hat.

Für die Bestimmung der kleinstmöglich umschließenden Ellipse, in der Literatur als *smallest enclosing ellipsoid* bekannt, existieren zwei Klassen von Algorithmen. Zum einen kann die Lösung über Verfahren erfolgen, die exakt rechnen oder solche, die die Lösung über Heuristiken bestimmen und damit von der tatsächlichen Lösung abweichen können. Verfahren, die auf Heuristiken basieren, haben eine geringere Laufzeit als solche, die exakt rechnen. Abhängig der im Anwendungsszenario erforderlichen Genauigkeit kann ein entsprechendes Verfahren gewählt werden. In der Arbeit wird eine existierende Implementierung von Gärtner und Schönherr [38] verwendet, die auf Heuristiken basiert und meist sehr genau die kleinste umschließende Ellipse findet. Der Algorithmus stellt zu jeder Approximation auch zwei Werte bereit, wobei der eine, accuracy genannt, die maximale Distanz der approximierten Ellipse zur Region liefert und der andere, slack genannt, die Abweichung vom angenommenen Optimum. Es wird von den Autoren empfohlen, die approximierte Ellipse zu akzeptieren, wenn $slack = 0$ und $accuracy \leq e^{-15}$.

Nachdem der Algorithmus die detektierten MSER durch Ellipsen approximiert hat, werden die Hauptachsen der Ellipsen durch einen festen Faktor gestreckt und die Ellipsen wie in Abschnitt 9.1.2 beschrieben ausgewählt und im Folgenden als *Einprägungsbereich* bezeichnet.

9.1.5 Verstärken der MSER

Zur Steigerung der Robustheit und der Wahrscheinlichkeit, dass der Detektor auch die gleichen MSER wiederfindet, wird der Rand innerhalb und außerhalb der MSER verstärkt. Bei einer Maximalregion erhöht der Algorithmus dazu die drei äußersten Pixelreihen einer MSER in ihrer Helligkeit, da sie heller ist als ihre Umgebung, und verringert die drei Pixelreihen außerhalb der MSER. Bei einer Minimalregion ist die MSER dunkler als ihre Umgebung und deshalb wird ihr Rand in die entgegengesetzte Richtung verändert. Dazu werden die beiden Pixelreihen, die die Grenze der MSER einschließen, um drei Helligkeitsstufen verändert. Die daran angrenzenden Pixelreihen werden um zwei Helligkeitsstufen verändert. Die verbleibenden zwei Pixelreihen werden nur um eine Helligkeitsstufe verändert.

Die MSER selbst wird verstärkt, indem bei einer Minimalregion das Muster in Abbildung 9.4 subtrahiert und bei einer Maximalregion addiert wird. Dazu wird das Muster auf die Größe der kleinsten umschließenden Ellipse der MSER skaliert und über die Hauptachsen der Ellipsen ausgerichtet, bevor das Muster addiert oder subtrahiert wird. Die Verstärkung fügt niedrige Frequenzen im Bereich der MSER hinzu. Die Robustheit wird auf Kosten der Transparenz erhöht.

Abbildung 9.4: Muster zum Verstärken der MSER eines Videoframes

9.1.6 Einbetten der Wasserzeichennachricht

Für das Einbetten der Wasserzeichennachricht benötigt der Algorithmus die Einprägungsbereiche eines Frames, das Muster und die Nachricht. In jedes Frame wird ein Bit des Wasserzeichens eingebettet. In jeden Einprägungsbereich wird abhängig dem einzubettenden Bit das Muster auf den Einprägungsbereich addiert oder subtrahiert. Damit das addierte Muster nicht wahrnehmbar ist, nutzt der Algorithmus die in Kalker [50] eingeführte Laplace-Filter Matrix:

$$L = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{9.1}$$

Die Matrix wird nur auf die Einprägungsbereiche angewandt, um die Stärke der möglichen Änderungen zu bestimmen. Die Matrix L wird mit $L* = min(20, 0.05 \cdot L)$ verwendet. Damit wird das Muster schwächer im Vergleich zu L eingebettet, was die Robustheit reduziert, jedoch die Nicht-Wahrnehmbarkeit wahrt. Weiterhin wird mit dem Anwenden der Szenenerkennung von Trick und Thiemert [98] die Transparenz der Muster im Video weiter verbessert. Der Laplace-Filter reduziert die Wahrnehmbarkeit der Änderungen innerhalb eines Frames und die Szenenerkennung reduziert das Flickering und damit die Wahrnehmbarkeit der Änderungen innerhalb einer Szene. Der Flickering-Effekt tritt auf, wenn z.B. in aufeinanderfolgenden Frames innerhalb einer Szene die eingeprägten Muster an der gleichen Stelle invertiert eingebettet werden.

9.1.7 Synchronisation

Die Synchronisation ist eine fest vordefinierte Abfolge von Bits, die dem Wasserzeichen vorangestellt wird. Sie ermöglicht es, die Startposition der Wassereichennachricht zu finden. Beim Detektieren des Wasserzeichens wird entsprechend nach dieser Synchronisation gesucht. Nachdem sie gefunden wurde, beginnt das Detektieren des Wasserzeichens.

Die Abfolge der Bits darf beliebig gewählt werden. In dieser Arbeit wird das Synchronisationmuster mit 111000111000111000 gewählt. Bevor das Synchronisationsmuster gestartet wird, wird geprüft, ob es ohne Unterbrechung und vollständig in eine Sequenz eingebettet werden kann. Es gibt Sequenzen am Anfang der Videos, die einen Vorspann, einfarbige Flächen, Schriftzüge oder sich schnell überblendende Bereiche beinhalten und damit nur vereinzelt verwendbare MSER bieten. Die Synchronisation wird nach dem gleichen Verfahren wie das Wasserzeichen eingebettet.

9.1.8 Wasserzeichen Detektion

Zuerst wird nach dem Synchronisationsmuster gesucht. Hat der Algorithmus es gefunden, wird das Wasserzeichen detektiert. Sowohl beim Detektieren des Synchronisationsmusters als auch der Wasserzeichennachricht werden die gleichen Schritte durchlaufen. Zuerst werden die MSER detektiert und mit der kleinsten umschließenden Ellipse approximiert. Die Hauptachsen der Ellipse werden mit einem festen Faktor gestreckt. Gestreckte Ellipsen, die nicht vollständig innerhalb des Frames liegen, werden eliminiert. Bei sich überschneidenden Ellipsen wird nur die Ellipse verwendet, die eine stabilere MSER approximiert. Aus den resultierenden Ellipsen wird das Bit ausgelesen. Dazu berechnet der Algorithmus die Korrelation zwischen der Mustervorlage und der Region zwischen der MSER und der Ellipse. Vorher muss die Mustervorlage entsprechend auf die Ellipse skaliert und an den Hauptachsen ausgerichtet werden.

Um zu bestimmen, ob das Frame das Bit “1“ oder “0“ trägt, werden jeweils die Helligkeitswerte der Pixel, die im Muster den Wert “1“ und “-1“ haben, getrennt voneinander aufaddiert und jeweils durch ihre Anzahl an Pixel dividiert. Der so erhaltene Durchschnittswert der Helligkeiten beider Gruppen wird für den Wert “1“ der Variable A zugeordnet und die, die den Wert “-1“ haben, der Variablen B. Anschließend werden beide Werte ins Verhältnis gesetzt und daraus der Logarithmus bestimmt. Das daraus resultierende Ergebnis wird zum Bestimmen des Bits b_i verwendet:

$$m_i = \begin{cases} 1, & \text{wenn } \log_{10} \frac{A}{B} > \tau \\ 0, & \text{wenn } \log_{10} \frac{A}{B} < -\tau \\ ?, & \text{sonst} \end{cases}$$

Findet der Algorithmus beim Auslesen fälschlicherweise andere MSER als beim Einbetten, so wird durch den Schwellwert τ sichergestellt, dass daraus kein Wasserzeichenbit detektiert wird. Bei der Analyse von unmarkiertem Material hat sich gezeigt, dass der Logarithmus der Relationen beider Gruppen zueinander $'0'$ oder sehr nahe $'0'$ ist und damit kleiner τ.

9.2 Evaluierung

Zum Evaluieren der Sicherheit des vorgestellten Videowasserzeichenverfahrens werden wasserzeichenmarkierte Videos mit unterschiedlichem Inhalt angegriffen.

9.2.1 Testaufbau

Die Testvideos bestehen aus unterschiedlichen Inhalten und werden mit 4 Wasserzeichenbit pro Frame markiert. Die Testbasis besteht aus Trailer, Landschaftsvideos, Animationen, Zeichentrick, Serien und Videos aus Videospielen. Die Trailer enthalten viele und schnelle Bewegungen und häufige Szenenwechsel und einfarbige Frames. Die Landschaftsfilme enthalten dagegen sehr wenig Bewegung mit wenig Änderungen. In die Kategorie der Animationen wurden computeranimierte Musikvideos aufgenommen. Zu den Zeichentrickfilmen zählen alle Videos, die typischerweise durchweg große einfarbige Regionen haben. Das Set der Serien enthält Spielfilme und TV Shows. In das Set der Videospiele wurden alle Videosequenzen aus Videospielen aufgenommen.

Die Testvideos haben alle eine Auflösung von 1280x720, 30 Frames pro Sekunde und einer Bitrate von 10.000kBit/s. Jedes Video wurde mit der Nachricht 100110010 und der inversen Nachricht 011001101 markiert. Die Videos wurden vor dem Markieren auf eine Länge von 30 Sekunden geschnitten. Ohne die Aktivierung der Szenenerkennung bilden 12 Frames eine Sequenz zur Einbettung eines Wasserzeichenbits. Damit sind 3,4 Sekunden Spieldauer eines Videos nötig, um einmal die Wasserzeichennachricht einzubetten. Mit Aktivierung der Szenenerkennung variiert die benötigte Spieldauer abhängig der Dauer der Szenen. Bei der Evaluierung der Robustheit wurde der Faktor zum Einbetten des Wasserzeichenmusters mit 2 gewählt. Damit wurden alle Helligkeitswerte beim Einbetten mit $-2, 2$ modifiziert.

9.2.2 Sicherheit und Robusheit

Die Sicherheit und Robustheit des Wasserzeichens gegen typische Videowasserzeichenangriffe und Nachverarbeitungsschritte wurde anhand der folgenden Angriffe und Nachverarbeitungsschritte evaluiert: Abfilmen, lineares und nicht lineares Skalieren, Zuschneiden, Rotieren, Spiegeln, Formatkonvertierung, Ändern der Framerate und Ändern der Bitrate. Die letztgenannten drei sind der Evaluierung der Robustheit zuzuordnen, während die verbleibenden der Evaluierung zur Sicherheit zuzuordnen sind.

Nach jedem der Evaluierungschritte wurde das Wasserzeichen detektiert und die Bitfehlerrate (BER) zu der vorher eingebetteten Wasserzeichennachricht berechnet. Als Referenz wird für jedes Video nach dem Einbetten, ohne dass ein Angriff oder Nachverarbeitung stattgefunden hat, das Wasserzeichen detektiert und die BER bestimmt. Die Ergebnisse der BER nach den Angriffen und Nachverarbeitungsschritten sind in Tabelle 9.1 gelistet.

Nr.	Angriffsszenario	BER
1	Ohne Angriff	0.89%
2	Reduzieren der Bitrate auf 2000kBit/s	5.91%
3	Reduzieren der Bitrate auf 1000kBit/s	24.66%
4	Reduzieren der Bitrate auf 500kBit/s	30.64%
5	Lineares Skalieren auf 640x360	13.35%
6	Lineares Skalieren auf 1920x1080	1.68%
7	Nicht-Lineares Skalieren auf 960x720	4.77%
8	Zuschneiden auf 960x720	3.55%
9	Rotieren um 10°	2.71%
10	Horizontale Spiegelung	45.10%
11	Formatkonvertierung nach h.264	0.87%
12	Ändern der Framerate auf 25 Frames/s	3.35%
13	Ändern der Framerate auf 20 Frames/s	5.75%
14	Abfilmen mit Smartphone Kamera (iPhone 4, Samsung Galaxy S4)	27.69%
15	Abfilmen mit CamMark	12.27%

Tabelle 9.1: Evaluationsergebnisse der BER nach den entsprechenden Angriffen

Faktor	2	3	4	5	6
SSIM	0.03	0.05	0.08	0.13	0.16

Tabelle 9.2: Evaluationsergebnisse der SSI Metrik nach Trick und Thiemert [98]

9.2.3 Nicht-Wahrnehmbarkeit

Die Transparenz des vorgestellten Videowasserzeichenverfahrens wurde mit der SSI Metrik, die in [98] für Videos erweitert wurde, bewertet. Dazu wurden fünf verschiedene Tests mit den Videos des Testsets durchgeführt, wobei jedes Pixel des Wasserzeichenmusters mit einem festen Faktor multipliziert wurde, bevor es im Einbettungsprozess auf den Ellipsenbereich addiert wurde. Folgende Faktoren wurden bei den Tests gewählt: 2, 3, 4, 5 und 6.

Zur Bewertung der Qualität und damit der Transparenz mit der SSIM wird sowohl das originale als auch das markierte Video benötigt. Tabelle 9.2 sind die Ergebnisse der Bewertung zu entnehmen. Werte nahe an $'0'$ lassen auf eine gute Qualität der Videos nach dem Markieren schließen. Je größer die Werte, desto schlechter die Qualität, wobei $'1'$ der höchste erreichbare Wert in der existierenden Implementierung ist.

9.2.4 Diskussion und Bewertung der Evaluationsergebnisse

Die Evaluationsergebnisse zeigen, dass ein Wasserzeichenbit mit hoher Wahrscheinlichkeit nach einer Vielzahl an Angriffen korrekt wiedererkannt wird. Je mehr die Qualität des markierten Videos reduziert wird, desto mehr steigt die BER. Das entspricht den Erwartungen, wobei das Wasserzeichen so lange auslesbar sein sollte, wie die Qualität des Videos akzeptabel ist. Ab wann die Qualität als inakzeptable gilt, ist abhängig vom Videokontent und Anwendungsszenario. Darauf wird hier nicht weiter eingegangen.

Beim Reduzieren der Bitrate in den Angriffen 2-4 ist eine steigende BER abzulesen bei Verringern der Bitrate. Der Grund dafür ist eine Kombination aus geschwächten MSER und geschwächtem Wasserzeichenmuster. Wird beim Generieren der Muster darauf geachtet, dass weniger hohe Frequenzen enthalten sind und die Werte des Musters nach dem Filtern der hohen Frequenzen nicht wieder auf $-1, 1$ gesetzt, so könnte sich die BER verbessern. Die geschwächten MSER führen dazu, dass nicht alle bzw. die MSER nicht in ihrer Ursprungsform und Ausrichtung erkannt werden. Damit ist die Berechnung der Korrelation zwischen dem Einbettungsmuster und dem markierten Video fehlerhaft. Als Lösung können die MSER weiter verstärkt oder fehlerkorrigierende Codes eingesetzt werden.

Das Verändern der Auflösung stellt für das Videowasserzeichen keine große Herausforderung dar. Es ist jedoch zu erkennen, dass beim Verändern der Auflösung die BER zunimmt. Der Grund liegt bei den kleiner werdenden MSER, die durch das Skalieren entstehen. Bei der Ausrichtung und Größe der gefundenen Region gibt es Abweichungen im Vergleich zur ursprünglichen Region. Als Lösung kann ein kleinerer Schwellwert τ_m beim Erstellen des Musters dienen.

Die drei Angriffe in 7,8 und 9 haben keine signifikanten Auswirkungen auf die BER. Vereinzelt wird zwar das eingebettete Muster stark geschwächt, z.B. durch Abschneiden von Bereichen des Frames, in denen das Muster eingebettet wurde, aber durch die redundante Einbettung bleibt die BER gering.

Die horizontale Spiegelung ist für das Wasserzeichenverfahren nicht ohne weiteres erkennbar. Damit das Wasserzeichen nach einer Spiegelung automatisiert erkannt werden kann, soll beim Auslesen des Wasserzeichens die Korrelation mit dem gespiegelten Wasserzeichenmuster berechnet werden, wenn die Korrelation mit dem ursprünglichen Muster sehr klein ist und der Wasserzeichendetektor das Video als unmarkiert einstuft. Das gleiche gilt für die vertikale Spiegelung.

Da das vorgestellte Verfahren unabhängig von einem Codec entwickelt wurde, ist es auch invariant gegen Formatkonvertierung. Wird bei der Konvertierung die Bitrate mit reduziert, so steigt die BER, wie auch bei den Angriffen 2-4 zu sehen.

Das Verändern der Framerate hat keinen signifikanten Einfluss auf das Wasserzeichenverfahren. Es wird über eine feste Sequenz 4 Bit eingebettet und somit hat die Framerate keinen Einfluss. Die Verschlechterung der BER kommt durch das Überblenden der Frames, wobei dadurch MSER oder die eingebetteten Muster geschwächt werden. Damit ist in einigen Fällen die MSER nicht mehr in ihrer ursprünglichen Form zu finden, oder die Korrelation ist zu gering.

Wie die Ergebnisse der BER nach dem Abfilmen mit den Smartphone Kameras und dem Nutzen von CamMark zeigen, ist das Verfahren sicher gegen dieses Angriffsszenario. Die Ergebnisse mit CamMark haben eine geringere BER, weil sie eine wesentlich bessere Auflösung und Qualität haben als die mit den Smartphones abgefilmten Videos. Die Aufnahmen wurden von älteren Smartphones gemacht. Es wurde ein 24-Zoll Monitor mit einer Frequenz von 80Hz genutzt. Es ist bei den Aufnahmen mit den Smartphones ein Rauschmuster zu erkennen, das sich über das ganze Bild erstreckt.

Durch das Erhöhen der Redundanz lässt sich die BER weiter reduzieren. Das Verfahren ist in der aktuellen Implementierung nicht echtzeitfähig, kann aber durch Parallelisierung beschleunigt werden, um echtzeitfähig zu werden. Die Transparenz des Videowasserzeichenverfahrens ist hoch mit Werten nahe $'0'$. Die eingebetteten Muster sind nicht störend. In wenigen Fällen können sie im Video erkannt werden.

Der Vergleich der Ergebnisse mit dem Stand der Technik zeigt, dass das vorgestellte Verfahren robust gegen Abfilmen, Rotation, nichtlineare Skalierung und Zuschneiden der Frames ist.

9.3 Zusammenfassung

Bei dem vorgestellten Videowasserzeichenverfahren wurde der Schwerpunkt auf die Sicherheit gegen bekannte Angriffe mit besonderem Fokus auf die Sicherheit gegen das Abfilmen gelegt. Das Abfilmen ist heute einfacher denn je durch das Streamingangebot für zu Hause umzusetzen und stellt einen lang bekannten Angriff dar, gegen den die bestehenden und getesteten Videowasserzeichenalgorithmen nicht sicher sind.

Das Videowasserzeichenverfahren basiert auf MSER, die aus jedem einzelnen Frame extrahiert werden. Um diese extrahierte Region wird die kleinste umschließende Ellipse berechnet und um einen festen Faktor vergrößert. Bei sich überschneidenden Ellipsen werden nur solche weiter verwendet, die die stabilste MSER approximieren. Beim Einbettungsprozess wird ein vorher kreiertes Muster herangezogen und auf die Größe der Ellipse skaliert. Abhängig vom einzubettenden Bit wird das Muster oder sein inverses Muster zu den Helligkeitswerten der Ellipsenregion außerhalb der MSER, aber innerhalb der Ellipse addiert. Die MSER selbst wird verstärkt, sodass die Wahrscheinlichkeit steigt, sie nach

starken Angriffen unbeschadet wiederzufinden. Der Detektionsalgorithmus bestimmt über die Berechnung der Korrelation zwischen dem verwendeten Muster und der Region des Frames, welches Bit eingebettet wurde. Zum Sichern der visuellen Qualität nutzt der Algorithmus einen Laplace-Filter und eine Szenenerkennung.

Die Evaluationsergebnisse zeigen eine Invarianz des Verfahrens gegen verschiedene Angriffe. Bei Angriffen mit einer BER höher als 10% wurden Optimierungsvorschläge diskutiert. Kommerzielle Verfahren, die den Anspruch erheben, sicher gegen das Abfilmen zu sein, konnten nicht evaluiert werden. Das Verfahren ist sicher gegen das Abfilmen und weitere Angriffe, wie Rotation, nichtlineares Skalieren und Zuschneiden.

10 Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Forschungsarbeit wurde das Sicherheitsziel der Authentizität für Multimediainhalte betrachtet. Für den Nachweis der Authentizität eines Kunden, Urhebers und Trägermediums wurden verschiedene Anforderungen für unterschiedliche Multimediatypen gestellt, bei denen im Stand der Technik nicht alle Eigenschaften berücksichtigt wurden. Als Lösungen werden digitale Wasserzeichenverfahren für 3D-Modelle, Texturen und Videos vorgeschlagen. Ein robustes Hashverfahren wurde für 3D-Modelle vorgestellt sowie eine Metrik zur Bewertung der Nicht-Wahrnehmbarkeit wasserzeichenmarkierter, texturierter 3D-Modelle eingeführt. Ein Fingerprintverfahren wurde sowohl für bedrucktes und unbedrucktes Papier als auch für Verpackungen vorgestellt.

Das entwickelte Fingerprintverfahren ermöglicht es, Druckerzeugnisse mit Smartphones zu authentifizieren, wobei die gestellten Anforderungen erreicht wurden. Das Verfahren ist robust bei verschiedenen Umgebungseinflüssen, Substraten, Druckverfahren und Smartphones. Der extrahierte Merkmalsraum besteht aus einzigartigen, wieder identifizierbaren Informationen im Rauschen bei mehreren Ausdrucken des gleichen Druckers und der gleichen Eingabedatei. Als Grundlage für den Fingerabdruck wurden die Fleckigkeit und Körnigkeit von monochromen Blöcken betrachtet und daraus die Merkmale extrahiert. Außerdem wurde die Homogenität jedes Blocks betrachtet, um so automatisiert Klone zu erkennen. Das Verfahren wurde hinsichtlich der Anforderungen an die Robustheit, Sicherheit und öffentliche Verifikation bewertet. Die Druckerzeugnisse von professionellen Offset- und Digitaldrucken zeigten Optimierungspotential auf. Bei Offset Druckerzeugnissen kann die Robustheit weiter optimiert werden, indem entweder die Überbelichtung bei der Aufnahme herausgerechnet wird oder bei der Veredelung auf einen nicht reflektierenden Lack zurückgegriffen wird.

Das entwickelte 3D-Modell-Wasserzeichen ist für Drahtgittermodelle einsetzbar und entspricht den gestellten Anforderungen. Die Evaluierungsergebnisse zeigen eine gute Robustheit, wobei die Vereinfachung des Drahtgittermodells und das ungleichmäßige Skalieren Herausforderungen darstellen. Beide Angriffe wirken sich visuell auf die Wahrnehmbarkeit aus. Die diskutierten Vorschläge können verfolgt werden, wenn die beiden Angriffe in Anwendungsszenarien eine Rolle spielen. Der Algorithmus modifiziert beim Einbetten die Knotenverteilung eines Kernmodells. Das Kernmodell erhält der Algorithmus

durch eine spektrale Kompression. Ein evolutionärer Optimierer übernimmt die iterative Einbettung, um das bestmögliche Ergebnis zu erreichen. Der Algorithmus kann in der Performanz optimiert werden. Dazu könnte das Verfahren für jedes Modell und jedes Wasserzeichenbit nur einmal Vorberechnungen durchführen. Beim Erzeugen einer wasserzeichenmarkierten Kopie müsste dann nur noch das entsprechende Set an Knoten kopiert werden.

Da das Markieren von Texturen mit herkömmlichen Bildwasserzeichenalgorithmen nicht robust möglich ist, wurde ein DDS Wasserzeichenalgorithmus entworfen, wobei die Anforderungen an die Robustheit und Nicht-Wahrnehmbarkeit erfüllt werden. Bei der Konzeptionierung wurden Angriffe berücksichtigt, die speziell bei Texturen möglich sind, wie z.B. der Hintergrund-Rausch-Angriff und Mip-Map-Angriff. Der Algorithmus bettet die Wasserzeicheninformation auf den komprimierten Daten ein. Als blockbasierter Ansatz werden die Indizes eines Blocks nach festgelegten Regeln geflippt.

Damit die getrennt voneinander markierten Texturen und 3D-Modelle nach dem Texturieren in der Nicht-Wahrnehmbarkeit bewertet werden können, bedarf es einer entsprechenden Metrik, die im Rahmen der Arbeit entwickelt wurde. Mithilfe der Metrik kann die Parametrisierung der Wasserzeichenalgorithmen vereinfacht werden, ohne dass aufwändige ABX-Tests notwendig werden. Die Metrik berücksichtigt die subjektive Wahrnehmung eines menschlichen Betrachters und überführt sie in einen objektiven Wert. Als Ansatz wurden Metriken aus dem 2D und 3D-Bereich modifiziert, indem auch z.B. Rauheit, Flächenänderung und Verzerrung mit in der Metrik berücksichtigt werden. Bei der Metrik werden die nicht benötigten Bereiche einer Textur durch das Mapping bei der Bewertung auch nicht berücksichtigt. Zur Feinabstimmung der Metrik auf die menschliche Wahrnehmung müsste eine breite Evaluierung mit Personen durchgeführt werden. Optimierungen sind durch Anpassen der Gewichtung der Einflussgrößen im 3D-Raum möglich.

Das robuste Hashverfahren für 3D-Modelle wurde entwickelt, um eine Einschätzung zu erhalten, ob es sich um ein bekanntes 3D-Modell handelt. Der robuste Hash zeigt bei allen getesteten Angriffen eine höhere Sicherheit als das Wasserzeichenverfahren. Bei der Vorgehensweise ermittelt das Verfahren einen robusten Bezugspunkt und transformiert mithilfe dessen die Knoten in ihre sphärischen Koordinaten. Anschließend werden Bins erzeugt und jedes Bin wird zu seinem übergeordneten Bin ins Verhältnis gesetzt. Aus dem Verhältnis lässt sich ein Hashbit extrahieren. Die unterschiedlichen Hashlängen von 16, 128 und 256 Bits wurden evaluiert. Die Ergebnisse zeigen, dass das Verfahren bei den drei Angriffen Rotation, gleichmäßige Skalierung und Translation sicher ist. Außerdem kann bei einer Hashlänge von 256 Bits mit einem Schwellwert von $\tau = 17$ die Sicherheit gegen das Addieren von zufälligem Rauschen, Vereinfachen, Verschieben und Tessellieren bei geringen FRR und FAR erreicht werden. Inwieweit die Schwächen des Verfahrens

bei der Krümmung und ungleichmäßigen Skalierung relevant sind, muss abhängig vom Einfluss auf die Nicht-Wahrnehmbarkeit bei der jeweiligen Applikation evaluiert werden. Die beiden Angriffe verändern das 3D-Modell in deren Grundform.

Beim entwickelten Videowasserzeichenverfahren wurde der Schwerpunkt auf die Sicherheit gegen das Abfilmen gelegt. Das Abfilmen ist heute einfacher denn je durch das Streamingangebot für zu Hause umzusetzen und stellt einen lang bekannten Angriff dar, gegen den die bestehenden und getesteten Videowasserzeichenalgorithmen nicht sicher sind. Die Evaluationsergebnisse zeigen eine Invarianz des Verfahrens gegen verschiedene Angriffe. Bei Angriffen mit einer BER höher als 10% wurden Optimierungsvorschläge diskutiert. Kommerzielle Verfahren, die den Anspruch erheben, sicher gegen das Abfilmen zu sein, konnten nicht evaluiert werden. Das Verfahren ist sicher gegen das Abfilmen und weitere Angriffe, wie Rotation, nichtlineares Skalieren und Zuschneiden. Das implementierte Verfahren basiert auf MSER, die aus jedem einzelnen Frame extrahiert werden. Um diese extrahierte Region wird die kleinste umschließende Ellipse berechnet und um einen festen Faktor vergrößert. Die Grenzen der Region werden gestärkt, sodass die Möglichkeit, die identische MSER wiederzufinden, gegeben ist. Zum Einbetten der Wasserzeicheninformation wird ein vorher kreiertes Muster herangezogen und auf die Größe der Ellipse skaliert sowie in Abhängigkeit des einzubettenden Bits addiert oder subtrahiert.

Publikationsliste des Autors der Dissertation

Im Folgenden werden alle meine Veröffentlichungen in zeitlich absteigender Reihenfolge aufgelistet.

[13] Waldemar Berchtold, Dani El-Soufi und Martin Steinebach. Smartphone-supported integrity verification of printed documents. In: Electronic Imaging, Media Watermarking, Security, and Forensics 2022

[63] Huajian Liu, Simon Bugert, Waldemar Berchtold, und Martin Steinebach. Cultural assets identification using transfer learning. In: Electronic Imaging, Imaging and Multimedia Analytics at the Edge 2022

[18] Waldemar Berchtold, Huajian Liu, Simon Bugert, York Yannikos und Martin Steinebach. Recognition of objects from looted excavations with the help of a smartphone. In: Electronic Imaging, Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2022

[90] Thomas Schnattinger, Nikolas Thenée, Waldemar Berchtold und Huajian Liu. Mehr Sicherheit durch Farbe – Digitale Siegel und der neue internationale JAB-Code Standard. In: 18. Deutscher IT-Sicherheitskongress 2022

[14] Waldemar Berchtold, Markus Sütter und Martin Steinebach. Fingerprinting Blank Paper and Printed Material by Smartphones. In: Electronic Imaging, Media Watermarking, Security, and Forensics 2021

[16] Waldemar Berchtold, Huajian Liu, Martin Steinebach, Dominik Klein, Tobias Senger und Nicolas Thenée. JAB Code - A Versatile Polychrome 2D Barcode. In: Electronic Imaging, Mobile Devices and Multimedia: Technologies, Algorithms & Applications 2020

[105] Christian Winter, Waldemar Berchtold, Jan Niklas Hollenbeck. Securing physical documents with digital signatures. In: IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), 2019

[94] Martin Steinebach und Waldemar Berchtold. MP3 partial encryption for DRM. In: Electronic Imaging, Media Watermarking, Security, and Forensics 2017

[8] Waldemar Berchtold, Patrick Lieb und Martin Steinebach. Secure communication protocol for a low-bandwidth audio channel, In: 25th European Signal Processing Conference (EUSIPCO), 2017

[12] Waldemar Berchtold, Marcel Schäfer und Martin Steinebach. Maximal stable extremal regions for robust video watermarking, In: Electronic Imaging, Media Watermarking, Security, and Forensics 2016

[85] Marcel Schäfer, Waldemar Berchtold, Teetje Stark, Nils Reimers und Martin Steinebach. A novel attack model for collusion secure fingerprinting codes, In: Electronic Imaging, Media Watermarking, Security, and Forensics 2016

[17] Waldemar Berchtold, Marcel Schäfer, Sascha Wombacher und Martin Steinebach. Quality metric for 2D textures on 3D objects, In: Electronic Imaging, Media Watermarking, Security, and Forensics 2016

[87] Marcel Schäfer, Sebastian Mair, Waldemar Berchtold und Martin Steinebach. 2015. Universal threshold calculation for fingerprinting decoders using mixture models, In Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '15). Association for Computing Machinery, New York, NY, USA, 109–114.

[19] Waldemar Berchtold, Marcel Schäfer, Michael Rettig und Martin Steinebach. Robust hashing for 3D models, In: Proceedings of SPIE - The International Society of Optical Engineering, Media Watermarking, Security, and Forensics 2014

[64] Huajian Liu, Waldemar Berchtold, Marcel Schäfer, Patrick Lieb und Martin Steinebach. Watermarking textures in video games, In: Proceedings of SPIE - The International Society of Optical Engineering, Media Watermarking, Security, and Forensics 2014

[99] Daniel Trick, Waldemar Berchtold, Marcel Schäfer und Martin Steinebach. 3D watermarking in the context of video games, 2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP), 2013

[11] Waldemar Berchtold, Marcel Schäfer und Martin Steinebach. 2013. Leakage detection and tracing for databases, In Proceedings of the first ACM workshop on Information hiding and multimedia security (IH&MMSec '13). Association for Computing Machinery, New York, NY, USA

[20] Waldemar Berchtold, Marcel Schäfer, Huajian Liu, Fabian Touceira Takahashi, Andre SChmitz, Sascha Zmudzinski, Martin Steinebach und Jonas Wieneke. Video game watermarking, In: Proceedings of SPIE - The International Society of Optical Engineering, Media Watermarking, Security, and Forensics 2013

[9] Waldemar Berchtold und Marcel Schäfer. 2012. Performance and code length optimization of joint decoding tardos fingerprinting. In Proceedings of the on Multimedia and security (MM&Sec '12). Association for Computing Machinery, New York, NY, USA

[84] Marcel Schäfer, Waldemar Berchtold und Martin Steinebach. Ranking search for probabilistic fingerprinting codes, Proceedings of SPIE - The International Society of Optical Engineering 2012

[10] Waldemar Berchtold und Marcel Schäfer. Rebound on symmetric tardos codes, Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2012

[15] Waldemar Berchtold, Sascha Zmudzinski, Marcel Schäfer und Martin Steinebach. Collusion-secure patchwork embedding for transaction watermarking. In: Proceedings of SPIE - The International Society of Optical Engineering, Media Watermarking, Security, and Forensics III, 2011

[83] Marcel Schäfer, Waldemar Berchtold, und Martin Steinebach. 2011. Fast and adaptive tracing strategies for 3-secure fingerprint watermarking codes. In Proceedings of the 11th annual ACM workshop on Digital rights management (DRM '11). Association for Computing Machinery, New York, NY, USA

[86] Marcel Schäfer, Waldemar Berchtold, Margareta Heilmann, Sascha Zmudzinski, Martin Steinebach und Stafan Katzenbeisser. Collusion Secure Fingerprint Watermarking for Real World Applications. In: Gesellschaft für Informatik, Proc. of GI-Sicherheit 2010

[88] Marcel Schäfer, Waldemar Berchtold, Sascha Zmudzinski und Martin Steinebach. 2010. Zero false positive 2-secure fingerprinting watermarking based on combining hamming distance conditions and parent pair search. In Proceedings of the 12th ACM workshop on Multimedia and security (MM&Sec '10). Association for Computing Machinery, New York, NY, USA

Patent Waldemar Berchtold, Marcel Schäfer, Sascha Zmudzinski und Martin Steinebach. Verfahren zur Auswertung von mit Transaktionswasserzeichen markiertem Datenmaterial zwecks Kundenrückverfolgung. DPMA German Patent DE102010044228 A1, März 2012. (Einreichung: Sep. 2010, Erteilung: Okt. 2012)

Literatur

[1] URL: http://www.autodesk.com/products/autodesk-3ds-max/overview.

[2] URL: http://www-graphics.stanford.edu/data/3Dscanrep/.

[3] Jürgen Adamy. *Fuzzy Logik, Neuronale Netze und Evolutionäre Algorithmen*. Berichte aus der Steuerungs– und Regelungstechnik. Aachen: Shaker, 2005. ISBN: 3832244611.

[4] Afroja Akter, Nur-E-Tajnina und Muhammad Ahsan Ullah. „Digital image watermarking based on DWT-DCT: Evaluate for a new embedding algorithm“. In: *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*. 2014, S. 1–6.

[5] Ralph H. Baer. „Digital video modulation and demodulation system“. US 1975/3993861A. 1975.

[6] S. Baudry u. a. „Modeling the flicker effect in camcorded videos to improve watermark robustness“. In: *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*. 2014, S. 42–47.

[7] W. Bender u. a. „Techniques for Data Hiding“. In: *IBM Systems Journal, MIT Media Lab* 35.3,4 (1996), S. 313–336.

[8] Waldemar Berchtold, Patrick Lieb und Martin Steinebach. „Secure communication protocol for a low-bandwidth audio channel“. In: *2017 25th European Signal Processing Conference (EUSIPCO)*. 2017, S. 2206–2210.

[9] Waldemar Berchtold und Marcel Schäfer. „Performance and code length optimization of joint decoding Tardos fingerprinting“. In: *Proceedings of the on Multimedia and security*. MM&Sec ’12. Coventry, United Kingdom: ACM, 2012, S. 27–32. ISBN: 978-1-4503-1417-6.

[10] Waldemar Berchtold und Marcel Schäfer. „Rebound on Symmetric Tardos Codes“. In: *Intelligent Information Hiding and Multimedia Signal Processing, 2012. IIHMSP ’12 International Conference on*. 2012.

[11] Waldemar Berchtold, Marcel Schäfer und Martin Steinebach. „Leakage Detection and Tracing for Databases“. In: *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*. IH&MMSec ’13. Montpellier, France: Association for Computing Machinery, 2013, S. 29–34. ISBN: 9781450320818.

[12] Waldemar Berchtold, Marcel Schäfer und Martin Steinebach. „Maximal stable extremal regions for robust video watermarking“. In: *Society for Imaging Science and Technology -IS&T, International Symposium on Electronic Imaging 2016. Media Watermarking, Security, and Forensics. 2016, San Francisco, California, USA.* IS&T SPIE. Jan. 2016.

[13] Waldemar Berchtold, Dani El-Soufi und Martin Steinebach. „Smartphone-supported integrity verification of printed documents“. In: *International Symposium on Electronic Imaging Science and Technology (IS&T)*. 2022.

[14] Waldemar Berchtold, Markus Sütter und Martin Steinebach. „Fingerprinting Blank Paper and Printed Material by Smartphones“. In: *International Symposium on Electronic Imaging Science and Technology (IS&T)*. 2021.

[15] Waldemar Berchtold u. a. „Collusion-secure patchwork embedding for transaction watermarking“. In: *Proceeding of Electronic Imaging 2011 - Media Watermarking, Security, and Forensics XIII*. IS&T SPIE. Jan. 2011.

[16] Waldemar Berchtold u. a. „JAB Code - A Versatile Polychrome 2D Barcode“. In: *Electronic Imaging, Mobile Devices and Multimedia: Technologies, Algorithms & Applications*. 2020.

[17] Waldemar Berchtold u. a. „Quality metric for 2D textures on 3D objects“. In: *Society for Imaging Science and Technology -IS&T, International Symposium on Electronic Imaging 2016. Media Watermarking, Security, and Forensics. 2016, San Francisco, California, USA.* IS&T SPIE. Jan. 2016.

[18] Waldemar Berchtold u. a. „Recognition of objects from looted excavations with the help of a smart-phone“. In: *International Symposium on Electronic Imaging Science and Technology (IS&T)*. 2022.

[19] Waldemar Berchtold u. a. „Robust Hashing for 3D Models“. In: *Proceeding of Electronic Imaging 2014 - Media Watermarking, Security, and Forensics 2014*. IS&T SPIE. Jan. 2014.

[20] Waldemar Berchtold u. a. *Video game watermarking*. 2013.

[21] A. G. Bors und M. Luo. „Optimized 3D watermarking for minimal surface distortion“. In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*. 2012, S. 1822–1835.

[22] Marco Bräuning. „Optimierung von DDS-Watermarking gegen Mipmap-basierte Angriffe“. Magisterarb. TU-Darmstadt, 2014.

[23] Darren Brooker. „Essential CG lighting techniques with 3ds max“. In: *Focal Press* 2nd edition (2006).

[24] J. Buchanan u. a. „Forgery: ‘Fingerprinting’ documents and packaging“. In: *Nature* 436 (2005), S. 475–475.

[25] P. Cano u. a. „A review of algorithms for audio fingerprinting“. In: *2002 IEEE Workshop on Multimedia Signal Processing.* 2002, S. 169–173.

[26] J.-W. Cho, R. Prost und H.-Y. Jung. „An Oblivious Watermarking for 3-D Polygonal Meshes Using Distribution of Vertex Norms“. In: *Signal Processing, IEEE Transactions on* 55.1 (2007), S. 142–155. ISSN: 1053-587X.

[27] Cheun Ngen Chong u. a. „Anti-counterfeiting with a Random Pattern“. In: *2008 Second International Conference on Emerging Security Information, Systems and Technologies*. 2008, S. 146–153.

[28] William Clarkson u. a. „Fingerprinting Blank Paper Using Commodity Scanners“. In: *2009 30th IEEE Symposium on Security and Privacy*. 2009, S. 301–314.

[29] *Computational Representations of Geometry*. 1996.

[30] M. Corsini u. a. „Watermarked 3-D Mesh Quality Assessment“. In: *IEEE Transactions on Multimedia* 9.2 (2007), S. 247–256.

[31] Massimiliano Corsini, Elisa Drelie Gelasca und Touradj Ebrahimi. „A multi-scale roughness metric for 3d watermarking quality assessment“. In: *in Workshop on Image Analysis for Multimedia Interactive Services 2005*. 2005.

[32] I. J. Cox, M. L. Miller und J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2002.

[33] I. J. Cox u. a. „Secure spread spectrum watermarking for multimedia“. In: *IEEE Transactions on Image Processing* 6.12 (1997), S. 1673–1687.

[34] Rony Darazi, Roland Hu und BenoÎt Macq. „Applying Spread Transform Dither Modulation for 3D-mesh watermarking by using perceptual models“. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, S. 1742–1745.

[35] Jana Dittmann. *Digitale Wasserzeichen*. Springer-Verlag Berlin Heidelberg, 2000. ISBN: 978-3-540-66661-5.

[36] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. De Gruyter Oldenbourg, 2018. ISBN: 9783110563900.

[37] Bernd Gärtner. „Fast and Robust Smallest Enclosing Balls“. In: *Proceedings of the 7th Annual European Symposium on Algorithms*. ESA ’99. London, UK, UK: Springer-Verlag, 1999, S. 325–338.

[38] Bernd Gärtner und Sven Schönherr. *Smallest Enclosing Ellipses – Fast and Exact*. 1997.

[39] Ingrid Gerdes, Frank Klawonn und Rudolf Kruse. *Evolutionäre Algorithmen: genetische Algorithmen, Strategien und Optimierungsverfahren, Beispielanwendungen*. Vieweg, 2004. ISBN: 3-528-05570-7.

[40] T. Haist und H.J. Tiziani. *Optical Detection of Random Features for High Security Applications*. Universitätsbibliothek der Universität Stuttgart, 2007.

[41] T. Harte und A. G. Bors. „Watermarking 3D models“. In: *Proceedings. International Conference on Image Processing*. Bd. 3. 2002, 661–664 vol.3.

[42] J. Herre, E. Allamanche und O. Hellmuth. „Robust matching of audio signals using spectral flatness features“. In: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*. 2001, S. 127–130.

[43] Roland Hu, Patrice Rondao-Alface und Benoit Macq. „Constrained optimisation of 3D polygonal mesh watermarking by quadratic programming“. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2009, S. 1501–1504.

[44] Konstantine I. Iourcha, Krishna S. Nayak und Zhou Hong. „System and method for fixed-rate block-based image compression with inferred pixel values“. US 5956431. 1999.

[45] ISO/IEC 11172-3. „MPEG-1 - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 MBit/s – Part 3: Audio“. In: *ISO/IEC JTC 1/SC* (Mai 1993).

[46] *ISO/IEC 24790:2017 information technology — office equipment — measurement of image quality attributes for hardcopy output — monochrome text and graphic images.*

[47] *ISO/TS 15311-1:2019 graphic technology – print quality requirements for printed matter – part 1: Measurement methods and reporting schema.*

[48] Gerald Kaiser. *A Friendly Guide to Wavelets*. USA: Birkhauser Boston Inc., 1994. ISBN: 0817637117.

[49] Linus Kälber. „Efficient Computation of Smallest Enclosing Balls in Three Dimensions“. In: *Miniconference on Interesting Results in Computer Science and Engineering, IRCSE ’12,* (2013).

[50] Ton Kalker u. a. „A video watermarking system for broadcast monitoring“. In: *Proceedings of IS&T/SPIE/EI25, Security and Watermarking of Multimedia Content*. 1999, S. 103–112.

[51] Zachi Karni und Craig Gotsman. „Spectral Compression of Mesh Geometry“. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000).

[52] Zoran Kotevski und Pece Mitrevski. „Experimental Comparison of PSNR and SSIM Metrics for Video Quality Estimation“. In: *ICT Innovations 2009*. Hrsg. von Danco Davcev und Jorge Marx Gómez. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 357–366. ISBN: 978-3-642-10781-8.

[53] Ashish M. Kothari, Vedvyas Dwivedi und Rohit M. Thanki. „Video Watermarking in Spatial Domain“. In: *Watermarking Techniques for Copyright Protection of Videos*. Cham: Springer International Publishing, 2019, S. 19–35.

[54] Ki-Ryong Kwon u. a. „Watermarking for 3D polygonal meshes using normal vector distributions of each patch“. In: *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*. Bd. 2. 2003, S. II–499.

[55] Muna M Laftah. „3D Model Watermarking based on Wavelet Transform“. In: *Iraqi Journal of Science* 62.12 (2021), S. 4999–5007.

[56] R. Lancini, F. Mapelli und S. Tubaro. „A robust video watermarking technique in the spatial domain“. In: *International Symposium on VIPromCom Video/Image Processing and Multimedia Communications*. 2002, S. 251–256. DOI: 10.1109/VIPROM.2002.1026664.

[57] Suk Hwan Lee u. a. „Key-dependent 3D model hashing for authentication using heat kernel signature“. In: *Digit. Signal Process.* 23.5 (2013), S. 1505–1522.

[58] Suk-Hwan Lee und Ki-Ryong Kwon. „Robust 3D mesh model hashing based on feature object“. In: *Digit. Signal Process.* 22.5 (Sep. 2012), S. 744–759. ISSN: 1051-2004.

[59] Suk-Hwan Lee, Eung-Joo Lee und Ki-Ryong Kwon. „Multi-scale Curvature-Based Robust Hashing for Vector Model Retrieval and Authentication“. In: *Arabian Journal for Science and Engineering*. 2019.

[60] Suk-Hwan Lee, Eung-Joo Lee und Seong-Geun Kwon. „Robust 3D mesh hashing based on shape features“. In: *2010 IEEE International Conference on Multimedia and Expo*. 2010, S. 1040–1043.

[61] Suk-Hwan Lee u. a. „Hash function for 3D mesh model authentication“. In: *2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*. 2015, S. 1–5.

[62] Patrick Lieb. „Optimization of Texture Watermarking for Video Games“. Magisterarb. Technische Universität Darmstadt, 2013.

[63] Huajian Liu u. a. „Cultural assets identification using transfer learning“. In: *International Symposium on Electronic Imaging Science and Technology (IS&T), Imaging and Multimedia Analytics at the Edge*. 2022.

[64] Huajian Liu u. a. *Watermarking Textures in Video Games*. 2014.

[65] Yang Liu, Balakrishnan Prabhakaran und Xiaohu Guo. „A Robust Spectral Approach for Blind Watermarking of Manifold Surfaces“. In: MM&Sec ’08. Oxford, United Kingdom: Association for Computing Machinery, 2008, S. 43–52.

[66] P. D. S. K. Malarchelvi. „A Semi-Fragile Image Content Authentication Technique based on Secure Hash in Frequency Domain“. In: *Int. J. Netw. Secur.* 15 (2013), S. 365–372.

[67] Xiaoyang Mao, Makoto Shiba und Atsumi Imamiya. „Watermarking 3D geometric models through triangle subdivision“. In: *Security and Watermarking of Multimedia Contents III*. Hrsg. von Ping Wah Wong und Edward J. Delp III. Bd. 4314. International Society for Optics und Photonics. SPIE, 2001, S. 253–260.

[68] Salvador Marti’nez, Se’bastien Ge’rard und Jordi Cabot. „Efficient model similarity estimation with robust hashing“. In: *Software and Systems Modeling*. Association for Computing Machinery, 2022.

[69] Salvador Marti’nez, Se’bastien Ge’rard und Jordi Cabot. „Robust Hashing for Models“. In: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. MODELS ’18. Copenhagen, Denmark: Association for Computing Machinery, 2018, S. 312–322. ISBN: 9781450349499.

[70] J. Matas u. a. „Robust Wide Baseline Stereo from Maximally Stable Extremal Regions“. In: *Proc. BMVC*. 2002, S. 36.1–36.10. ISBN: 1-901725-19-7.

[71] David Nistér und Henrik Stewénius. „Linear Time Maximally Stable Extremal Regions“. In: *Proceedings of the 10th European Conference on Computer Vision: Part II*. ECCV ’08. Marseille, France: Springer-Verlag, 2008, S. 183–196. ISBN: 978-3-540-88685-3.

[72] Matthias Peter Nowak. „Identifizierung von Druckungenauigkeiten für einen Fingerabdruck“. Magisterarb. Technische Universität Darmstadt, 2021.

[73] J.J.K. O'Ruanaidh und T. Pun. „Rotation, scale and translation invariant digital image watermarking“. In: *Proceedings of International Conference on Image Processing*. Bd. 1. 1997, 536–539 vol.1.

[74] Hamza Özer, Bülent Sankur und Nasir Memon. „Robust audio hashing for audio identification“. In: *2004 12th European Signal Processing Conference*. 2004, S. 2091–2094.

[75] Ravikanth Pappu. „Physical One-Way Functions“. In: *Science (New York, N.Y.)* 297 (Okt. 2002).

[76] Steffen Pochanke. „Verbesserung des Fälschungsschutzes eines auf Smartphone-Kameras basierenden Fingerabdruckverfahrens für Drucke“. Magisterarb. Technische Universität Darmstadt, 2020.

[77] C. Podilchuk und W. Zeng. „Perceptual watermarking of still images“. In: *Multimedia Signal Processing, 1997., IEEE First Workshop on*. Juni 1997, S. 363–368.

[78] Michael Rettig. „Robust Hashing Algorithm for 3D Models“. Magisterarb. Hochschule Darmstadt, 2013.

[79] Ulrich Rührmair u. a. „Optical PUFs Reloaded“. In: *Report* (2013).

[80] „Mean Squared Error“. In: *Encyclopedia of Machine Learning*. Hrsg. von Claude Sammut und Geoffrey I. Webb. Boston, MA: Springer US, 2010, S. 653–653. ISBN: 978-0-387-30164-8.

[81] Jie Sang, Qi Liu und Chun-Lin Song. „Robust video watermarking using a hybrid DCT-DWT approach“. In: *Journal of Electronic Science and Technology* 18.2 (2020), S. 100052.

[82] Philipp Schaber u. a. „CamMark: A Camcorder Copy Simulation As Watermarking Benchmark for Digital Video“. In: *Proceedings of the 5th ACM Multimedia Systems Conference*. MMSys '14. Singapore, Singapore: ACM, 2014, S. 91–102. ISBN: 978-1-4503-2705-3.

[83] Marcel Schäfer, Waldemar Berchtold und Martin Steinebach. „Fast and adaptive tracing strategies for 3-secure fingerprint watermarking codes“. In: *Proceedings of the 11th annual ACM workshop on Digital rights management*. DRM '11. Chicago, Illinois, USA: ACM, 2011, S. 41–50. ISBN: 978-1-4503-1005-5.

[84] Marcel Schäfer, Waldemar Berchtold und Martin Steinebach. „Ranking search for probabilistic fingerprinting codes“. In: *Proceedings of Electronic Imaging 2012 - Media Watermarking, Security, and Forensics 2012*. Proceedings of SPIE. IS&T/SPIE. Jan. 2012.

[85] Marcel Schäfer u. a. „A novel attack model for collusion secure fingerprinting codes“. In: *Society for Imaging Science and Technology -IS&T, International Symposium on Electronic Imaging 2016. Media Watermarking, Security, and Forensics. 2016, San Francisco, California, USA*. IS&T SPIE. Jan. 2016.

[86] Marcel Schäfer u. a. „Collusion Secure Fingerprint Watermarking for Real World Applications“. In: *Proc. of GI-Sicherheit 2010*. Gesellschaft für Informatik. Okt 2010.

[87] Marcel Schäfer u. a. „Universal Threshold Calculation for Fingerprinting Decoders Using Mixture Models“. In: *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*. IH&MMSec ’15. Portland, Oregon, USA: Association for Computing Machinery, 2015, S. 109–114. ISBN: 9781450335874.

[88] Marcel Schäfer u. a. „Zero False Positive 2-secure Fingerprinting Watermarking based on Combining Hamming Distance Conditions and Parent Pair Search“. In: *Proceeding of The 12th ACM Workshop on Multimedia and Security 2010 (MMSEC 2010)*. Sep. 2010.

[89] Jannik Schildknecht. „Semi-blindes Videowasserzeichenverfahren für verbesserte Robustheit gegen Abfilmen“. Magisterarb. Technische Universität Darmstadt, 2014.

[90] Thomas Schnattinger u. a. „Mehr Sicherheit durch Farbe – Digitale Siegel und der neue internationaleJAB-Code Standard“. In: *18. Deutscher IT-Sicherheitskongress*. 2022.

[91] Matthias Senker. „Neues Verfahren für digitale Videowasserzeichen für verbesserte Robustheit gegen Abfilmen“. Magisterarb. Technische Universität Darmstadt, 2014.

[92] Ashlesh Sharma, Lakshminarayanan Subramanian und Eric Brewer. „PaperSpeckle: Microscopic fingerprinting of paper“. In: *CCS’11 - Proceedings of the 18th ACM Conference on Computer and Communications Security*. 2011, S. 99–109. ISBN: 9781450310758.

[93] B. Sridhar und C. Arun. „An Interlacing Technique-Based Blind Video Watermarking Using Wavelet“. In: *International Journal of Computer and Information Engineering* 9.6 (2015), S. 1514–1517.

[94] Martin Steinebach und Waldemar Berchtold. „MP3 Partial Encryption for DRM“. In: *Electronic Imaging, Media Watermarking, Security, and Forensics 2017*. 2017.

[95] Markus Sütter. „Evaluating the feasibility of printed material fingerprinting using smartphone cameras“. Magisterarb. Technische Universität Darmstadt, 2019.

[96] Ehsan Toreini, Siamak F. Shahandashti und Feng Hao. *Texture to the Rescue: Practical Paper Fingerprinting based on Texture Patterns*. 2019. arXiv: `1705.02510 [cs.CR]`.

[97] Daniel Trick. „Digital Watermarking for 3D-Models in Video Games“. Magisterarb. Technische Universität Darmstadt, 2013.

[98] Daniel Trick und Stefan Thiemert. *A new metric for measuring the visual quality of video watermarks*. 2011.

[99] Daniel Trick u. a. „3D Watermarking in the Context of Video Games“. In: *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*. 2013.

[100] Vijay K. Vaishnavi und William Kuechler Jr. *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*. 1st. Boston, MA, USA: Auerbach Publications, 2007. ISBN: 1420059327, 9781420059328.

[101] Yinghui Wang u. a. „3D model watermarking algorithm robust to geometric attacks“. In: *IET Image Processing* 11.10 (2017), S. 822–832.

[102] Z. Wang, E.P. Simoncelli und A.C. Bovik. „Multiscale structural similarity for image quality assessment“. In: *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*. Bd. 2. 2003, 1398–1402 Vol.2.

[103] Zhou Wang u. a. „Image Quality Assessment: From Error Visibility to Structural Similarity“. In: *IEEE TRANSACTIONS ON IMAGE PROCESSING* 13.4 (2004), S. 600–612.

[104] Zhou Wang u. a. „Image quality assessment: from error visibility to structural similarity“. In: *IEEE Transactions on Image Processing* 13.4 (2004), S. 600–612.

[105] Christian Winter, Waldemar Berchtold und Jan Niklas Hollenbeck. „Securing physical documents with digital signatures“. In: *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*. 2019, S. 1–6.

[106] Sascha Wombacher. „Bewertung von Texturunterschieden zweier Texturen von 3D Modellen“. Magisterarb. Hochschule Darmstadt, 2014.

[107] Jian-Hua Wu u. a. „An effective feature-preserving mesh simplification scheme based on face constriction“. In: *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*. 2001, S. 12–21.

[108] Xiang-Gen Xia, C.G. Boncelet und G.R. Arce. „A multiresolution watermark for digital images“. In: *Proceedings of International Conference on Image Processing*. Bd. 1. 1997, 548–551 vol.1.

[109] Faxin Yu u. a. „3D Model Watermarking“. In: *Three-Dimensional Model Analysis and Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 305–370. ISBN: 978-3-642-12651-2.

[110] Yong-zhao Zhan u. a. „A blind watermarking algorithm for 3D mesh models based on vertex curvature“. In: *Journal of Zhejiang University SCIENCE C* 15.5 (2014), S. 351–362.