Technische Universität Dresden • Fakultät Informatik

Steganalytische Untersuchung einer Erweiterung des steganographischen Algorithmus HUGO

Bachelorarbeit

zur Erlangung des ersten Hochschulgrades

Bachelor of Science (B.Sc.)

vorgelegt von: JAKOB KRUSE
geboren am: 21. August 1990 in DRESDEN

Tag der Einreichung: 03. 04. 2013
Betreuer: Dr. Elke Franz
(Institut für Datenschutz und Datensicherheit)

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Arbeit befasst sich mit der steganalytischen Untersuchung einer Erweiterung des steganographischen Algorithmus HUGO. Diese Erweiterung wurde in der Diplomarbeit *„Auswahl eines optimalen Stegobildes“* von Reduan Anton Franck am Institut für Datenschutz und Datensicherheit der Fakultät Informatik an der TU Dresden entwickelt. Sie konnte die beim Einbetten mit HUGO verursachten und für die Entdeckung einer Botschaft entscheidenden Veränderungen der Bildmerkmale in vielen Fällen deutlich reduzieren. Wie gut sich die erweiterte Methode gegen bestehende Angriffe auf HUGO bewährt, wurde jedoch noch nicht ausgewertet und soll Gegenstand dieser Arbeit sein.

Dazu wird in den folgenden Kapiteln zunächst auf die Grundlagen und Begrifflichkeiten der Steganographie und Steganalyse eingegangen. Danach wird ein mathematisches Modell namens SPAM für Bilder im Ortsbereich erläutert, welches die beim Einbetten einer versteckten Botschaft entstehenden Veränderungen herausstellt und für den darauf aufbauenden HUGO-Algorithmus von zentraler Bedeutung ist. Als nächstes wird HUGO selbst vorgestellt und beschrieben, auf welche Weise bestehende, spezialisierte Steganalyse-Werkzeuge eine Einbettung mit HUGO erkennen können. Schließlich wird genauer auf die vorgeschlagenen Erweiterungen und die ihnen zugrunde liegenden Ideen eingegangen, worauf die Auswertung mehrerer praktischer Versuche zur Entdeckbarkeit von HUGO – mit und ohne Erweiterungen – folgt. Im letzten Teil werden die Ergebnisse der Untersuchung zusammengefasst, und ein Ausblick auf denkbare Folgeuntersuchungen oder weitere Veränderungen gegeben.

2 Steganographische Grundlagen

Der Begriff *Steganographie* leitet sich aus den griechischen Wörtern *steganós*, „verdeckt“, und *gráphein*, „schreiben“, ab. Er bezeichnet also die Kunst des „verdeckten Schreibens“, das heißt die Wissenschaft der heimlichen Übermittlung vertraulicher Botschaften. Die allgemeine Zielstellung dabei lautet, dass nicht eingeweihte Dritte außerstande sein sollen, die Existenz einer übermittelten Botschaft zu entdecken. Um das zu erreichen, betten steganographische Verfahren die geheime Botschaft generell in eine größere, unverfängliche Nachricht ein, die man als *Cover* (engl. für „Deckung“, „Versteck“) bezeichnet. Das ganze Unterfangen gilt in dem Moment als gescheitert, in dem ein Dritter feststellt, dass sich hinter den Cover-Daten eine weitere, geheime Mitteilung verbirgt. Ob diese Mitteilung auch entziffert werden kann, spielt dabei keine Rolle. Die Steganographie unterscheidet sich somit deutlich von der Schwesterwissenschaft *Kryptographie*, die sich auf die Geheimhaltung des Inhalts einer Botschaft konzentriert. Auch muss man steganographische Verfahren von sogenannten *Wasserzeichen* abgrenzen, die trotz ähnlicher Konzepte gänzlich andere Ziele verfolgen. Ein *robustes* Wasserzeichen dient dazu, ein Objekt nichtwahrnehmbar, aber nachhaltig zu markieren und sollte nicht zerstörbar sein, ohne das Objekt zu beschädigen. *Fragile* Wasserzeichen zielen darauf ab, dass eine Manipulation des Objektes unweigerlich das Wasserzeichen zerstört. Steganographische Botschaften brauchen weder robust noch fragil zu sein, für sie hat Unbemerkbarkeit die höchste Priorität. Die dem gegenüber stehende Wissenschaft des Entdeckens verborgener Nachrichten nennt man *Steganalyse*.

Es gibt viele historische Beispiele für Steganographie, die bis in die Antike zurückgehen – nicht umsonst handelt es sich um ein Wort griechischen Ursprungs. Militärische Kommunikation, Spionage und geheime politische Absprachen haben seit jeher einen großen Erfindungsreichtum angeregt, wenn es um den entscheidenden Informations- und Wissensvorteil ging. So gibt es viele Rezepte für Geheimtinten, Wachstafeln mit geheimer Botschaft hinter der Wachsschicht, ausgefeilte Geheimschriften, linguistische Tricks in der Anordnung unscheinbarer Satzbausteine oder schlicht Jargonbegriffe, die alltäglichen

Äußerungen tiefere Bedeutung geben. Im zweiten Weltkrieg wurden unter dem Namen *Mikropunkt* ganze Schreibmaschinenseiten auf die Größe eines Punktes geschrumpft und so in Briefen versteckt. All diesen steganographischen Systemen ist gemeinsam, dass die Sicherheit des Verfahrens ausschließlich auf seiner Geheimhaltung beruht. Kennt ein Angreifer das vereinbarte Verfahren, ist es mehr oder minder trivial, die Existenz einer versteckten Botschaft festzustellen. Moderne Stego-Verfahren versuchen daher zumeist, in Anlehnung an die moderne Kryptographie *Kerckhoffs' Prinzip* umzusetzen und ihre Sicherheit auf den Einsatz eines geheimen Schlüssels zu stützen[1]. In seinem 1883 veröffentlichten Artikel *La cryptographie militaire* forderte Auguste Kerckhoffs sinngemäß: „Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus abhängen. Die Sicherheit gründet sich nur auf die Geheimhaltung des Schlüssels."

Heutige, rechnergestützte steganographische Verfahren sind mathematisch wesentlich komplexer als die genannten historischen Beispiele, da sie sich gegen sehr viel mächtigere und ebenfalls rechnerbasierte Angriffe behaupten müssen. Auch die Beschaffenheit der Cover-Daten hat sich mit dem Übergang ins Digitale verändert. Digitale Daten, was auch immer sie repräsentieren, liegen als diskrete Menge von Zahlenwerten bzw. Bits und Bytes vor. Damit solche Daten für steganographische Nachrichteneinbettung geeignet sind, müssen sie ein zufälliges Rauschen enthalten, das die vorgenommenen Änderungen überdeckt. Die beste Quelle für derartiges Grundrauschen sind digitalisierte Aufnahmen aus der analogen Welt, z.B. von Kameras oder Mikrophonen. Bei der Aufnahme fließen natürliche Störsignale und Ungenauigkeiten ein, die nicht vorhersagbar sind. Ob ein ohnehin zufällig entstandenes Bit nachträglich verändert wurde, ist für einen Angreifer ohne weiteres nicht festzustellen.

Die vorliegende Arbeit beschäftigt sich mit Steganographie in Bilddaten, welche im Ortsbereich vorliegen[2]. Im einfachsten Fall – von dem auch die weiteren Kapitel ausgehen werden – heißt das, dass für jedes Pixel des Bilds eine einzelne Zahl, üblicherweise 1 Byte lang, den Grauwert angibt. Da das Grundrauschen bei Digitalfotos gering ist, beschränkt sich die Einbettung geheimer Botschaften auf das jeweils niederwertigste Bit jedes Grauwerts, im Englischen *Least Significant Bit* oder kurz LSB. Eine Änderung des LSB eines Pixels um den Betrag 1 entspricht in diesem Fall einer Änderung des Farbwerts um $0,39\,\%$. Überschreibt man nun die durch den Geheimschlüssel indizierten LSBs des

[1] Zwei interessante Ausnahmen sind versteckte Schriftzeichen mittels Unicode-Tricks und Daten, die in den durch Fragmentierung ungenutzten Segmenten einer Festplatte verborgen werden. Beide sind nur sicher, solange ein Angreifer nicht weiß, wonach er zu suchen hat.

[2] Ortsbereich steht hier im Gegensatz beispielsweise zum Frequenzbereich, in dem u.a. Jpeg-Bilder vorliegen. Typische Ortsbereich-Formate sind bspw. Bitmap und Portable Graymap.

Cover-Bilds der Reihe nach mit den Bits der geheimen Botschaft, so ist die Veränderung optisch nicht wahrnehmbar – bei statistischer Auswertung des entstandenen Stego-Bildes wird sie aber sehr schnell ersichtlich [13].

Nur minimal anders, aber wesentlich weniger anfällig als LSB-Einbettung ist das so genannte LSB-Matching. In diesem Verfahren werden die LSBs im Cover-Bild nicht einfach überschrieben, sondern, falls sie dem einzubettenden Bit nicht ohnehin schon entsprechen, der gesamte Farbwert zufällig um eins erhöht oder gesenkt. Diese kleine Abweichung in der Art der Einbettung macht das Verfahren deutlich sicherer gegenüber statistischen Angriffen. Trotzdem ist es heute nicht mehr guten Gewissens einsetzbar, da ein Fehler in den oben beschriebenen Grundannahmen den Weg für neue Angriffe geöffnet hat. Diese Fehlannahme und ihre Ausnutzung werden im nächsten Kapitel erläutert.

3 Analyse mit SPAM

Im Jahr 2009 wurde von *Pevný*, *Bas* und *Fridrich* in [11] ein steganalytisches Verfahren basierend auf der so genannten *Subtractive Pixel Adjacency Matrix* (SPAM) vorgestellt, welches LSB-Matching und ähnliche Einbettungsmethoden mit großer Sicherheit erkennen kann.

Im Widerspruch zu der im vorigen Kapitel getroffenen Annahme, dass das Grundrauschen in Digitalfotos rein zufällig sei, nutzt dieses Verfahren Zusammenhänge im Rauschen benachbarter Pixel eines betrachteten Bildes aus. Zwar ist den Autoren nach das unverfälschte Sensor-Signal des Aufnahmegeräts durchaus mit zufälligem Rauschen behaftet, nachfolgende kamerainterne Verarbeitungsschritte führen allerdings neue Abhängigkeiten ein. So werden bei Farbinterpolation und -korrektur, Rauschunterdrückung und eventuell angewandten Filtern die Pixel stets unter Einbeziehung ihrer unmittelbaren Umgebung manipuliert. Auch bei einem nachträglichen Skalieren am PC findet eine Interpolation statt. Da eine Einbettung mittels LSB-Matching ein Pixel losgelöst von seiner Umgebung verändert, werden diese komplexen Abhängigkeiten an dieser Stelle verfälscht. Gelingt es, ein ausreichend genaues Modell für die „natürlich“ entstandenen Abhängigkeiten der Bilder einer Bildquelle zu entwickeln, dann können die durch steganographische Operationen verfälschten Bilder unter diesem Modell als solche identifiziert werden.

Um dieses Modell zu erhalten, wird ein Bild als erstes auf die Differenzen zwischen je zwei benachbarten Pixeln in den acht Richtungen $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$, $\nearrow$, $\searrow$, $\swarrow$ und $\nwarrow$ heruntergebrochen (Beispiel in Abbildung 3.1). Für die Richtung $\rightarrow$ ergibt sich damit eine Matrix $D^{\rightarrow}$ mit Komponenten

$$d^{\rightarrow}_{i,j} = I_{i,j} - I_{i,j+1},$$

wobei $I_{i,j}$ der Grauwert des Pixels mit den Koordinaten (i, j) ist. Die Matrizen für die anderen Richtungen entstehen analog dazu. Durch diesen Schritt werden die Eigenschaften des Bildes auf ein vereinfachtes Modell abgebildet, das für den geplanten Angriff besser beherrschbar ist. Die Bildinhalte, die für die Analyse unerheblich sind, werden

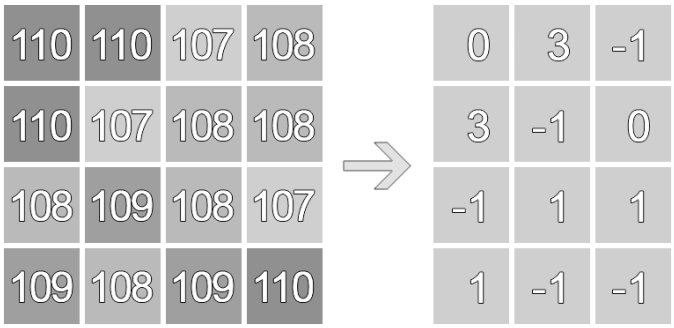


Abbildung 3.1: Pixeldifferenzen in Richtung →

unterdrückt und stattdessen die Beziehungen zwischen den Pixeln und eventuelle durch Steganographie eingebrachte Veränderungen hervorgehoben. Da benachbarte Pixel normalerweise ähnliche Farbwerte haben, hat eine Änderung des LSB einen wesentlich größeren Effekt auf die Pixeldifferenzen als auf die absoluten Farbwerte.

Bei angenommenen 256 möglichen Farbwerten pro Pixel gibt es 511 mögliche Pixeldifferenzen, von denen jedoch nur die betragskleinsten tatsächlich häufig genug auftreten, um statistisch aussagekräftig zu sein (siehe Abbildung 3.2). Deshalb beschränken die Autoren das Modell auf die Differenzen im Intervall $[-T, T]$, definiert durch den Schwellwert-Parameter T. Das hat den zusätzlichen Vorteil, die Dimension der Daten und damit die Rechenintensität aller folgenden Berechnungen zu verringern. Tatsächlich verwenden die Autoren in ihren Versuchen sehr niedrige Werte $T \in \{3, 4\}$.

Die auftretenden Pixeldifferenzen entlang je einer Richtung werden als Markov-Kette aufgefasst und die Übergangswahrscheinlichkeiten für alle Differenzen in $[-T, T]$ entlang jeder der acht Richtungen berechnet. Je nachdem wie komplex man das Modell haben möchte, kommen Markov-Ketten erster oder zweiter Ordnung zum Einsatz. Für die Richtung → ergibt sich dementsprechend je nach Ordnung eine Matrix $M^{\rightarrow}$ mit

$$M^{\rightarrow}_{u,v} = P(d^{\rightarrow}_{i,j+1} = u \mid d^{\rightarrow}_{i,j} = v)$$

beziehungsweise

$$M^{\rightarrow}_{u,v,w} = P(d^{\rightarrow}_{i,j+2} = u \mid d^{\rightarrow}_{i,j+1} = v, d^{\rightarrow}_{i,j} = w).$$

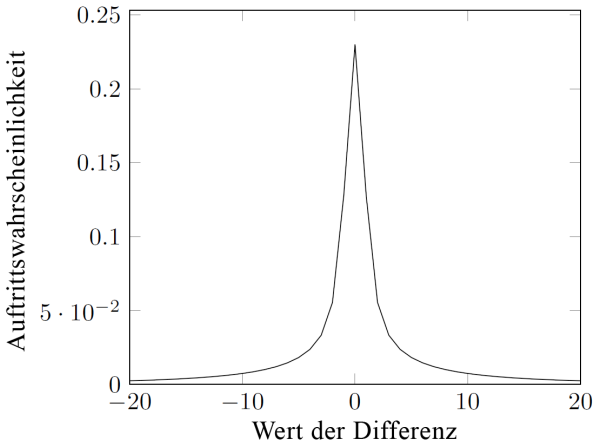


Abbildung 3.2: Durchschnittliche Verteilung der Differenzen benachbarter Pixel in einem größeren Bildbestand, entnommen aus [11].

Diese Eigenschaften eines Bildes bleiben den Autoren zufolge auch nach Spiegelung an den Hauptachsen bestehen, was den Matrizen eine gewisse Symmetrie verleiht. Daher werden anschließend die horizontalen und vertikalen Matrizen $M^{\leftarrow}$, $M^{\rightarrow}$, $M^{\uparrow}$ und $M^{\downarrow}$ gemeinsam zu F^{+} sowie die diagonalen Matrizen $M^{\nearrow}$, $M^{\searrow}$, $M^{\swarrow}$ und $M^{\nwarrow}$ zu $F^{\times}$ gemittelt. Somit liegen für ein Bild am Ende zwei Matrizen mit jeweils $(2T+1)^2$ Elementen für Markov-Ketten erster Ordnung beziehungweise $(2T+1)^3$ Elementen für Markov-Ketten zweiter Ordnung vor. Diese Matrizen heißen aufgrund ihrer Konstruktionsweise *Subtractive Pixel Adjacency Matrix*.

Die Elemente dieser beiden Matrizen dienen als *Features* bzw. Merkmale des betrachteten Bildes für Verfahren aus dem Bereich des überwachten maschinellen Lernens, die entscheiden sollen, ob es sich um ein „frisch fotografiertes“ oder ein steganographisch verändertes Bild handelt. Aneinandergereiht ergeben alle Features eines Bildes seinen *Feature-Vektor* F. In [11] wird für die Entscheidung eine *Support Vector Machine* eingesetzt, kurz SVM und übersetzbar als „Stützvektor-Methode“. Dabei handelt es sich um ein mathematisches Konstrukt, das eingegebene Vektoren auf Basis seiner angelernten Parameter einer von zwei Klassen zuordnet - im vorliegenden Fall *Cover-Bild* oder *Stego-Bild*. Um die Parameter zu lernen, benötigt die SVM eine große Trainingsmenge mit Feature-Vektoren von Cover- und Stego-Bildern, die als solche gekennzeichnet sind. Bilder aus der gleichen oder einer ähnlichen Quelle wie jener der Trainingsmenge können dann mit geringem Rechenaufwand klassifiziert werden.

Die Autoren des Artikels haben die SPAM-Features mit Markov-Ketten zweiter Ordnung und Schwellwert $T = 3$ auf einem großen Bildbestand getestet und konnten LSB-Matching bei einer Einbettungsrate von $0,25\,\text{bpp}$[1] mit einer Fehlerrate von nur $7,4\,\%$ entdecken. Die mit SPAM verglichenen steganalytischen Verfahren wiesen für die selben Daten Fehlerraten von $20,6\,\%$ bzw. $37,6\,\%$ auf.

[1] Bit pro Pixel: Verhältnis zwischen der Länge der eingebetteten Nachricht und der Größe des Cover-Bildes

4 Der HUGO-Algorithmus

Knapp ein Jahr nach der Veröffentlichung von SPAM wurde teilweise von den selben Autoren in [12] eine neue steganographische Methode mit dem Namen HUGO (kurz für *Highly Undetectable SteGO*, „äußerst schwer entdeckbar“) vorgestellt, die SPAM-Angriffen selbst bei höheren Einbettungsraten noch gut widersteht. Diese Methode ist adaptiv, d.h. sie passt sich in ihren Einbettungsentscheidungen an die Gegebenheiten des Cover-Bildes an, um die entstehende *Verzerrung* – die Abweichung des Stego-Bildes vom Cover im gewählten Bildmodell – so gering wie möglich zu halten.

Um die Verzerrung minimieren zu können, muss als erstes ein Maß für selbige eingeführt werden. Dieses Maß soll nicht-negativ und additiv sein und wird in diesem Fall wie folgt angegeben:

▸ **Definition 4.1:**
Seien $X = (x_{ij}), Y = (y_{ij}) \in \chi = \{0, \ldots, 255\}^{n_1 \times n_2}$ die Grauwerte eines Cover bzw. Stego-Bildes mit $n = n_1 n_2$ Pixeln. Dann berechnet sich das *additive Verzerrungsmaß* zwischen den beiden Bildern als

$$D(X, Y) = \sum_{k=1}^{n} \rho_k |x_k - y_k| .$$

Dabei bestimmen die Kosten-Parameter ρ_k, wie stark die gemessene Verzerrung ist, wenn das entsprechende Pixel verändert wird. Sie folgen aus dem Modell, mit dem das Bild beschrieben wird. Je höher ρ_k, desto ungünstiger ist es, dieses Pixel bei der Einbettung zu verwenden. Es ist wichtig festzustellen, dass hier die Verzerrungen einzelner Pixel-Veränderungen aufsummiert werden, als wären sie unabhängig voneinander. Wie sich herausstellen wird, ist das bei HUGO nicht der Fall, und das hier eingeführte additive Maß lässt sich nicht in dieser Form anwenden. Dies ist ein Grund für die Angreifbarkeit des Algorithmus und zum Teil auch Gegenstand der später vorgestellten Erweiterungen.

Die Betrachtung der Verzerrung als additiv hat aber einen wichtigen Vorteil. So führen die Autoren ein für additive Verzerrungsmaße geltendes Theorem aus [3] an, welches den Einfluss von Bildmodell (ρ_k) und Nachrichtenkodierung (p_k) separiert:

★ **Theorem 4.2.**
Sei $\rho = (\rho_k)_{k=1}^{n}, 0 \leq \rho_k < \infty$ *die Menge der Konstanten des additiven Verzerrungsmaßes für* $k \in \{1, \ldots, n\}$ *und* $0 \leq m \leq n$ *die Anzahl der Bits, die eingebettet werden sollen. Dann ist die minimal zu erwartende Verzerrung*

$$D_{min}(m, n, p) = \sum_{k=1}^{n} p_k \rho_k ,$$

wobei

$$p_k = \frac{e^{-\lambda \rho_k}}{1 + e^{-\lambda \rho_k}}$$

die Wahrscheinlichkeit ist, das k-te Pixel zu verändern. Den Parameter λ *erhält man durch Lösen von*

$$-\sum_{k=1}^{n} p_k \log_2 p_k + (1 - p_k) \log_2 (1 - p_k) = m.$$

Dieses Theorem erlaubt es den Entwicklern, die Performance eines gewählten Modells unabhängig von der konkreten Kodierung zu untersuchen, indem eine optimale Kodierung aufgrund der Wahrscheinlichkeiten p_k simuliert wird.

4.1 Vorgehen

Das in Kapitel 3 vorgestellte Bildmodell dient auch als Grundlage für HUGO. Statt der in der SPAM gespeicherten bedingten Wahrscheinlichkeiten werden hier aber sogenannte *Co-Occurrence*-Matrizen eingesetzt, die die Auftrittshäufigkeiten aller Grauwertfolgen in der jeweiligen Richtung beinhalten:

$$\begin{aligned} C^{\rightarrow}_{d_1,d_2} &= \left|\left\{\left(d^{\rightarrow}_{i,j}, d^{\rightarrow}_{i,j+1}\right) \mid d^{\rightarrow}_{i,j} = d_1, d^{\rightarrow}_{i,j+1} = d_2\right\}\right| \\ C^{\rightarrow}_{d_1,d_2,d_3} &= \left|\left\{\left(d^{\rightarrow}_{i,j}, d^{\rightarrow}_{i,j+1}, d^{\rightarrow}_{i,j+2}\right) \mid d^{\rightarrow}_{i,j} = d_1, d^{\rightarrow}_{i,j+1} = d_2, d^{\rightarrow}_{i,j+2} = d_3\right\}\right| \end{aligned}$$

Aus $C^{\rightarrow}_{d_1,d_2}$ und $C^{\rightarrow}_{d_1,d_2,d_3}$ lassen sich die SPAM-Features zweiter Ordnung der Richtungen $\leftarrow$ und $\rightarrow$ ableiten. Dies gilt analog auch für die anderen drei Richtungspaare. Die Co-Occurrence-Matrizen sind demnach eine Verallgemeinerung der SPAM, zudem lassen sie sich nach einem Einbettungsschritt wesentlich effizienter aktualisieren als bedingte Wahrscheinlichkeiten, die jedes Mal viele Divisionen kosten. Das Ziel bei der Einbettung ist es nun, die Co-Occurence-Matrizen so gut wie möglich zu erhalten.

Um eine Überanpassung des Modells an den Angriff mittels SPAM zu vermeiden, die potentielle Schwächen gegenüber anderen Angriffen nach sich zöge, ist eine weitere Verallgemeinerung vonnöten. Die Autoren lösen dies, indem sie den Schwellwert auf $T = 90$ anheben und argumentieren, dass die Dimensionalität des Modells damit deutlich außerhalb der Möglichkeiten moderner Steganalyse liegt. Da die Analyse mit Hilfe maschineller Lernverfahren vonstatten geht, trifft sie bei hohen Dimensionen der „Fluch der Dimensionalität“ – mit höherer Dimension der verarbeiteten Daten sind exponentiell größere Trainingsmengen nötig, um genug Exemplare jedes einzelnen Datums für eine qualifizierte Aussage über dessen Verteilung zu erhalten. Für die Steganographie stellt dies dagegen kein Problem dar, weil man hier mit den gegebenen Daten aus dem Cover-Bild arbeitet und über sichere Informationen verfügt.

Als nächstes wird eine Verzerrungs-Funktion D eingeführt, die die Unterschiede zwischen den Co-Occurrence-Matrizen C^X des Cover-Bildes und C^Y des Stego-Bildes gewichtet zusammenzählt und auf diese Weise bestimmten Merkmalen des Bildes größere Bedeutung zuspricht.

▸ **Definition 4.3:**

$$D(X,Y) = \sum_{d_1,d_2,d_3=-T}^{T} \left[w(d_1,d_2,d_3) \left| \sum_{k\in\{\rightarrow,\leftarrow,\uparrow,\downarrow\}} C^{X,k}_{d_1d_2d_3} - C^{Y,k}_{d_1d_2d_3} \right| + w(d_1,d_2,d_3) \left| \sum_{k\in\{\searrow,\nwarrow,\swarrow,\nearrow\}} C^{X,k}_{d_1d_2d_3} - C^{Y,k}_{d_1d_2d_3} \right| \right]$$

Die darin enthaltene Gewichtsfunktion w ist von der Form

$$w(d_1,d_2,d_3) = \frac{1}{\left(\sqrt{d_1^2+d_2^2+d_3^2}+\sigma\right)^{\gamma}}$$

und dient dazu, Veränderungen von Features, die besser für die Steganalyse geeignet sind, stärker in die Gesamtverzerrung einfließen zu lassen. Wenn ein Angreifer Features (d_1,d_2,d_3) aus einem Stego-Bild extrahiert, ist deren statistische Aussagekraft umso größer, je häufiger sie im Bild auftreten. Die Funktion w motiviert deshalb eine Einbettung in Bereichen mit hohen Pixeldifferenzen d, weil diese generell seltener in einem Bild vorkommen und daher für den Angreifer schlechter zu modellieren sind. Die Parameter σ und γ sind dabei standardmäßig 1 und können variiert werden, um andere Gewichtungen auszuprobieren.

Dieses Verzerrungsmaß D ist nun jedoch nicht mehr additiv wie in Definition 4.1 beschrieben. Die Autoren behelfen sich an dieser Stelle, indem sie die Einbettungskosten mit

$$\rho_{i,j} = D\left(X, Y^{i,j}\right)$$

annähern, wobei $Y^{i,j}$ das entstehende Stego-Bild bei ausschließlicher Änderung des Pixels (i,j) im Cover-Bild X ist.

Nachdem $\rho_{i,j}$ für jedes Pixel sowohl für Inkrementieren als auch Dekrementieren bestimmt wurde, werden in Abhängigkeit der einzubettenden Nachricht alle diejenigen Pixel ermittelt, die überhaupt einer Änderung bedürfen. In der einfachen Variante des Algorithmus wird daraufhin für jedes dieser Pixel die Einbettungsoperation (+1 oder −1) durchgeführt, der die geringeren Kosten zugewiesen wurden. Dieses Vorgehen führt allerdings zu erheblichen Sicherheitsmängeln, da jede Einbettung nach der nicht-additiven Verzerrungsfunktion aus Definition 4.3 Einfluss auf die Kosten aller darauf folgenden Einbettungen hat. Stattdessen bedient man sich einer Verbesserung namens *Modellkorrektur*: Die Kosten für die Inkremetierung/Dekrementierung eines Pixels werden zum Zeitpunkt der Einbettung neu berechnet, unter Einbeziehung aller vorherigen Änderungen. Dann wird wiederum die Operation gewählt, die die geringeren Kosten nach sich zieht. Die im Voraus berechneten Kosten dienen in diesem Fall nur noch dazu, die Abarbeitungsreihenfolge der Pixel festzulegen. Den Versuchen der Autoren nach hat sich dabei die Strategie bewährt, die Pixel absteigend von den höchsten Kosten $\rho_{i,j}$ zu den niedrigsten anzusteuern.

Der damit beschriebene steganographische Algorithmus erwies sich in den Testreihen der Autoren als äußerst robust gegen die bis dahin bekannten Angriffe. Bei einer Einbettungsrate von $0,3\,\text{bpp}$ konnte keine der untersuchten steganalytischen Methoden eine Fehlerrate von weniger als $40\,\%$ erreichen. Anders interpretiert entspricht diese Einbettung der siebenfachen Nachrichtengröße, die für LSB-Matching bei gleicher Fehlerrate der Angriffe möglich ist.

4.2 Erfolgreiche Angriffe auf HUGO

Während HUGO seinem Namen zur Zeit der Veröffentlichung alle Ehre machte, gab es auch in der Steganalyse einen beständigen Fortschritt. Ein Katalysator für das Aufholen der steganalytischen Seite war der Wettbewerb BOSS, *Break Our Steganographic System*,

der Fachleute aus aller Welt dazu einlud, an einem bereitgestellten Bildbestand ihre Angriffe auf HUGO zu testen [1]. Die dort vorgebrachten Ansätze, und was daraus geworden ist, sind Gegenstand der nächsten Abschnitte.

4.2.1 Boss-Challenge

Die BOSS-„Herausforderung“ lief vom 9. September 2010 bis 10. Januar 2011 und bestand darin, für 1.000 gegebene Bilder zu bestimmen, ob sie eine HUGO-Einbettung enthalten oder nicht. Dazu wurde eine Trainingsmenge aus 9.074 korrespondierenden Paaren von Cover- und Stego-Bildern mit den selben Einbettungsparametern zur Verfügung gestellt. Die Bilder stammen aus verschiedenen Kameras und wurden einheitlich auf 512×512 Pixel skaliert und auf Grauwerte heruntergebrochen.

Die höchste Punktzahl erreichte das Team *Hugobreakers* von der Binghamton University. In [5] werden die Ansätze beschrieben, die dieses Team gegen HUGO ins Feld führte und die im Folgenden vorgestellt werden sollen.

Ein erster Angriff bestand in der Suche nach einer Korrelation zwischen $\rho = \sum_i^n (y_i - x_i)\, \hat{s}_i$ (mit Cover-Pixel y_i, Stego-Pixel x_i und geschätztem Stego-Signal $\hat{s}_i$) und der Anzahl Pixel n eines Bildes. Da die verwendeten Größen nicht unabhängig voneinander sind, erwies sich die Methode als unergiebig.

Eine weitere Richtung war die Analyse in einer anderen Domäne als dem Ortsbereich, da HUGO für letzteren optimiert ist. Dazu wurden die untersuchten Bilder in den Wavelet-Raum überführt und verschiedene erweiterte Versionen der WAM-Features (s. [7]) für die Klassifikation mit einer SVM extrahiert. Das zeigte einen gewissen Erfolg, wurde aber übertroffen von der Analyse mit *Cross-Domain-Features* bzw. CDF (s. [8]), welche mit ähnlichen Ergebnissen auch von der Gruppe *BossTeam* der TU Prag angewandt wurden.

Eine noch bessere Erkennungsrate wurde wiederum im Ortsbereich erzielt, indem sogenannte *Residuen höherer Ordnung* zur Feature-Menge hinzugezogen wurden. Residuen erster Ordnung entsprechen dabei den gewohnten Pixeldifferenzen. Residuen höherer Ordnung sind Differenzen von je zwei benachbarten Residuen der nächst niedrigeren Ordnung, wie in Abbildung 4.1 dargestellt. Während HUGOs dreidimensionale Co-Occurrence-Matrizen Beziehungen zwischen je vier Pixeln wiederspiegeln, können mit Residuenfolgen höherer Ordnung Beziehungen größerer Pixelgruppen modelliert werden, ohne die Dimension der Co-Occurence-Matrizen anheben zu müssen. Damit wird der „Fluch

der Dimensionalität“ verhindert, der für vierdimensionale Co-Occurrence-Matrizen bereits Schwierigkeiten verursacht.

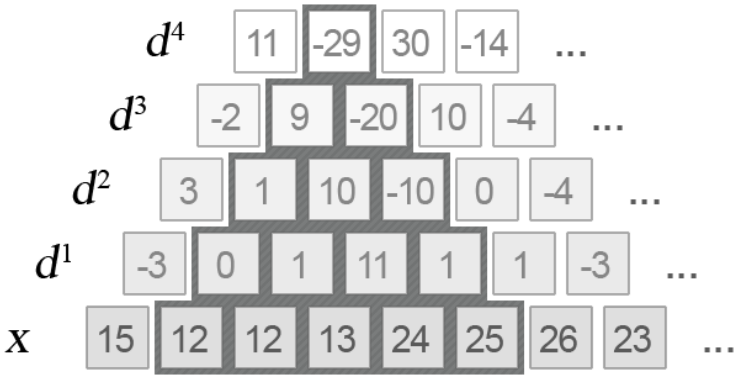


Abbildung 4.1: Bildung und Einfluss der Residuenordnungen d^1 bis d^4

Das Residuum 2. Ordnung $d^2_{i,j}$ berechnet sich somit beispielsweise als $x_{i,j-1} - 2x_{i,j} + x_{i,j+1}$. Eine alternative Interpretation dafür wird durch $d^2_{i,j} = 2\left(\hat{x}_{i,j} - x_{i,j}\right)$ angegeben, wobei $\hat{x}_{i,j} = \frac{1}{2}\left(x_{i,j-1} + x_{i,j+1}\right)$ ein Prädiktor für $x_{i,j}$ aufgrund seiner unmittelbaren Nachbarn ist.

Die Residuen 2. Ordnung wurden nun an jeder Koordinate in den Orientierungen $\leftrightarrow$, $\updownarrow$, $\nearrow$ und $\searrow$ berechnet und der kleinste und größte dieser vier Werte jeweils als $d^{\mathrm{MIN}}_{i,j}$ bzw. $d^{\mathrm{MAX}}_{i,j}$ festgehalten. Der Feature-Vektor F^{MINMAX} setzt sich dann aus den Co-Occurrence-Matrizen von horizontalen und vertikalen Tripeln von d^{MIN} und d^{MAX} zusammen, immer unter der Bedingung $d \in [-T, T]$. Um über den Schwellwert T „hinaus zu sehen“, wurde weiterhin ein *quantisierter* Feature-Vektor $F^{\mathrm{QUANT},q}$ eingeführt. Dieser ist wie F^{MINMAX} aufgebaut, verkleinert aber alle Differenzen d vorher mit dem Quantisierer $Q_q(d) = \left\lfloor \frac{d}{q} \right\rfloor$. Diese beiden Feature-Vektoren, gemeinsam von der Dimension 2.916, zeigten bereits ein gutes Erkennungsverhalten und motivierten die Suche nach weiteren Features. Um jedoch eine größere Vielfalt zu ermöglichen, war ein Ansteigen der Dimensionalität unumgänglich und die verwendete Svm deshalb nicht mehr praktikabel.

Stattdessen wurde für die wachsende Featuremenge ein neueres Konstrukt des maschinellen Lernens namens *Ensemble-Klassifikator* verwendet [10]. Ein solcher Klassifikator besteht aus einer Menge von L *Base Learnern*, denen je ein zufälliger Teilraum der Dimension d_{sub} des Feature-Raums zugeordnet wird. Die Base Learner sind einfache Fld-

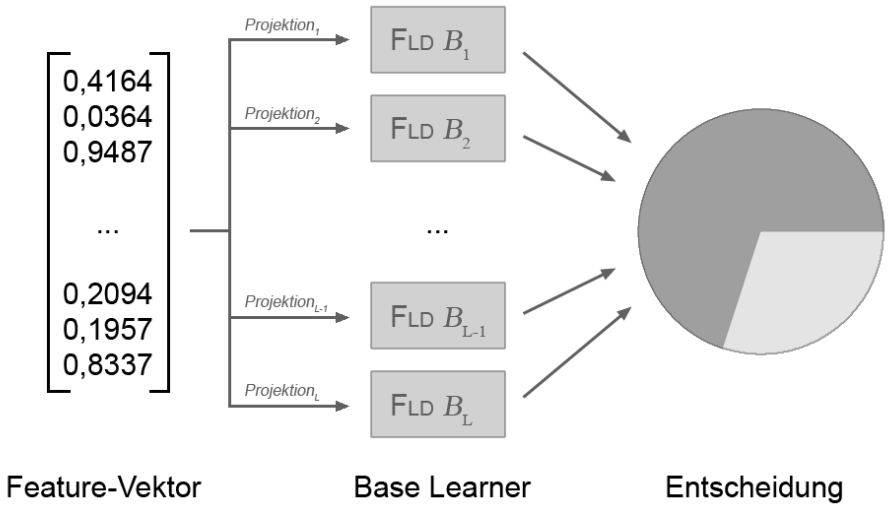


Abbildung 4.2: Entscheidungsfindung eines Ensemble-Klassifikators

Klassifikatoren[1], die von einem eingehenden Feature-Vektor nur die Projektion auf ihren jeweiligen Teilraum für ihre Entscheidung zwischen „Cover“ und „Stego“ nutzen. Die Entscheidungen aller L Base Learner ergeben per Mehrheitsabstimmung die Entscheidung des Ensembles für den Feature-Vektor (s. Abbildung 4.2). Dieser Aufbau ermöglicht ein überaus schnelles Training mit hochdimensionalen Feature-Räumen und ist in der Entscheidungsgenauigkeit den SVMs meist ebenbürtig [5]. Mit einer Reihe weiterer Features höherer Ordnungen, die am Ende einen Vektor der Dimension 24.993 bildeten, erwies sich dieser Ansatz schließlich als stärkster Beitrag zur Boss-Challenge.

Einen weiteren sehr erfolgreichen, aber etwas anderen Weg der Steganalyse wählte das Team *Guel&Kurugollu* [6]. Ihr Angriff nutzt ebenfalls eine stark hochdimensionale Feature-Menge, hier aber basierend auf multivariaten Wahrscheinlichkeitsdichtefunktionen. Im Gegensatz zum Ensemble-Angriff werden auf die Feature-Vektoren dann veschiedene Downsampling- und Feature-Selection-Verfahren angewandt, um sie für das Training mit einer SVM herunter zu skalieren.

Ein interessanter Nebenaspekt, der in [5] erwähnt wird, ist das Verhalten von Hugo bei Cover-Bildern, die größere Flächen identischer Farbwerte haben. Weil Hugo ungeeignete Pixel zwar meidet, ihnen aber dennoch eine minimale Einbettungswahrscheinlichkeit

[1] FLD steht für *Fisher's Linear Discriminant*, eine Methode der linearen Klassifikation, die durch geschickte Projektion der Feature-Vektoren versucht, die Varianz innerhalb zweier Klassen zu minimieren und zwischen den Klassen zu maximieren.

zugesteht, landen einige der Änderungen auch in diesen Flächen. Betrachtet man nur die LSBs des Stego-Bildes, so sind die betroffenen Punkte schon rein optisch sehr auffällig, zumal sie infolge der Details der Kostenberechnung gehäuft am Bildrand auftreten. Auch für die automatisierte Steganalyse sind solche Einbettungen ein leichtes Ziel, weshalb man Bilder dieser Art als Steganograph schlicht nicht als Cover nutzen sollte.

4.2.2 Angriff mit SRM

Der erfolgreichste Ansatz aus [5] wurde von den Autoren weiterentwickelt, verallgemeinert und 2012 unter dem Titel *Rich Models for Steganalysis of Digital Images*, kurz SRM, veröffentlicht [4]. Die Devise lautet weiterhin, viele möglichst unterschiedliche Teilmodelle zusammenzufügen um damit einen Ensemble-Klassifikator zu trainieren. Die Vielfalt der Modelle ist dabei wichtiger als ihre individuelle Komplexität, die weiterhin durch den Fluch der Dimensionalität beschränkt wird.

Die von oben bekannten Residuen $d_{i,j}$ werden hier für die Ordnung c angegeben mit

$$d_{i,j}^{c} = \hat{x}_{i,j}(\mathcal{N}_{i,j}) - c \cdot x_{i,j},$$

wobei $\hat{x}_{i,j}(\mathcal{N}_{i,j})$ als Prädiktor für $c \cdot x_{i,j}$ aufgrund seiner Umgebung $\mathcal{N}_{i,j}$ interpretiert wird. Die *Quantisierung* mit q und *Trunkierung* mit Schwellwert T geschehen durch

$$d_{i,j} \leftarrow \begin{cases} \left\lfloor \frac{d_{i,j}}{q} \right\rfloor, & \text{falls } \left\lfloor \frac{d_{i,j}}{q} \right\rfloor \in [-T, T] \\ T \cdot \operatorname{sgn}(d_{i,j}), & \text{sonst} \end{cases}$$

und erfüllen denselben Zweck wie bisher – die Einschränkung der Dimensionalität und gleichzeitige Einbeziehung von Features jenseits des Schwellwerts. Für SRM werden Co-Occurence-Matrizen vierter Dimension verwendet, der Schwellwert zum Ausgleich auf $T = 2$ herunter gesetzt und dafür mehrere Varianten eines Modells mit unterschiedlichen Werten für q aufgenommen. Die Vierer-Residuenfolgen werden nur entlang der Richtungen $\leftrightarrow$ und $\updownarrow$ gezählt, da sich die diagonalen Zusammenhänge als weniger nützlich erwiesen haben.

Eine weitere Änderung gibt es in der Auswahl der Residuen selbst, wie sie in Abbildung 4.3 dargestellt sind. Der schwarze Punkt entspricht dort jeweils dem zentralen Pixel $x_{i,j}$. Die um ihn herum angeordneten Symbole entsprechen den Pixeln der Nachbarschaft $\mathcal{N}_{i,j}$,

zu denen jeweils die Differenzen gebildet werden. Befinden sich unterschiedliche Symbole in der Nachbarschaft, so werden die Differenzen zu diesen Nachbarpixeln separat gebildet und je einmal mit den Operatoren min und max verknüpft. Dabei entstehen zwei Co-Occurrence-Matrizen. Diese Residuen vom Typ *minmax* bringen durch ihre nicht-lineare und asymmetrische Verteilung Abwechslung in die Feature-Menge. Residuen, die nur gleichartige Symbole enthalten, gehören dem Typ *spam* an.

Abbildung 4.3: Typen von Residuen, die in SRM verwendet werden (aus [4]).

Die Unterteilung der Residuen in sechs Klassen spiegelt die unterschiedlichen Annahmen der Prädiktoren $\hat{x}_{i,j}(\mathcal{N}_{i,j})$ wieder. Die Klasse **1st** schätzt Pixel anhand eines Nachbarn, **2nd** legt einen linearen und **3rd** (nicht abgebildet, aber analog zu **1st**) einen quadratischen Zusammenhang zugrunde. **SQUARE** bezieht alle Nachbarn mit Abstand 1 bzw. 2 ein, **EDGE3x3** und analog dazu **EDGE5x5** sind daraus abgeleitet, um besser mit Kanten im Bild umgehen zu können.

Die Namensgebung der einzelnen Residuen folgt dem Schema

$$\{\text{Typ}\}\{f\}\{\sigma\}\{scan\},$$

wobei der Typ entweder *spam* oder *minmax*, f die Anzahl unterschiedlicher Symbole und σ die Anzahl einzigartiger Orientierungen der Anordnung ist. Das Kürzel *scan* gibt an, in welcher Richtung (h für $\leftrightarrow$, v für $\updownarrow$) die Co-Occurence-Matrix für dieses Residuum gebildet werden soll. Ist die Summe der Matrizen beider Richtungen bei Rotation des Bildes um 90° konstant, so heißt das Residuum *hv-symmetrisch* und es wird nur diese Summe der Co-Occurence-Matrizen festgehalten – *scan* im Namen fällt dann weg.

Durch geschicktes Anwenden einiger dem Bild unterstellten Symmetrien lässt sich die Dimensionalität der aus den Matrizen entstehenden Featurevektoren noch einmal senken. So werden die Einträge C_{d_1,d_2,d_3,d_4} und C_{d_4,d_3,d_2,d_1} sowie C_{d_1,d_2,d_3,d_4} und $C_{-d_1,-d_2,-d_3,-d_4}$ (nur bei *spam*-Residuen) jeweils durch Addition zusammengefasst. Für *minmax*-Residuen lassen sich zudem die beiden Co-Occurrence-Matrizen wegen

$$\min(d_1, d_2, d_3, d_4) = -\max(-d_1, -d_2, -d_3, -d_4)$$

zu einer zusammenfassen. Schließlich wird jedes der so entstandenen Teilmodelle in drei Quantisierungen mit $q \in \left\{c, \frac{3}{2}c, 2c\right\}$ dem Gesamtmodell hinzugefügt, wobei c die Ordnung des zugrunde liegenden Residuums ist[2]. Nach diesem Schritt liegt ein äußerst breit gefächertes Bildmodell mit beachtlichen 34.671 Features vor, auf dem die Klassifikation mittels Ensemble aufbauen kann.

Im Gegensatz zu dem weiter oben beschriebenen Ensemble-Angriff werden die Parameter d_{sub} und L des Ensemble-Klassifikators allerdings nicht vorher festgelegt, sondern automatisiert anhand der Trainingsdaten gelernt. Dazu wird d_{sub} in größeren Schritten durchprobiert, und zu jeder betrachteten Dimension d_{sub} ein optimaler Wert L gesucht, sodass eine sogenannte *Out-Of-Bag*-Abschätzung für die Fehlerrate der Analyse minimal wird. Um das zu ermöglichen, werden die Base Learner für jedes Paar (d_{sub}, L) nur anhand eines *Bootstraps*[3] der Trainingsmenge trainiert. Danach kann für jedes Element der Trainingsmenge eine Probeentscheidung all jener Base Learner gesammelt werden, deren

[2] Ausnahme: Für Residuen der Ordnung 1 ergibt Quantisierung mit $q = \frac{3}{2}c$ und $q = 2c$ das selbe Modell, deshalb werden aus diesen Residuen nur zwei Teilmodelle erzeugt

[3] Ein *Bootstrap* ist eine spezielle Stichprobe, für die aus einer n-elementigen Menge genau n Mal mit Zurücklegen gezogen wird. Damit werden manche Elemente mehrfach gewählt und rund 37% der Menge bleiben unbetrachtet.

Training nicht von diesem Element abhing. Die OOB-Abschätzung entspricht der gemittelten Fehlerrate dieser Probeentscheidungen. Nach der Paremeter-Optimierung werden noch einmal alle Base Learner mit der gesamten Trainingsmenge trainiert und ergeben somit das fertige Ensemble.

Um Performance und Kompaktheit des Angriffs zu erhöhen, lassen sich auch beliebige Kombinationen der einzelnen Teilmodelle der vollständigen SRM-Methode bilden. In [4] werden mehrere Strategien vorgestellt, nach denen die geeignetsten Teilmodelle ausgewählt werden können. Die niedrigste Fehlerrate bei der Erkennung dreier untersuchter Stego-Verfahren weist aber das vollständige Modell auf. So vermelden die Autoren bei Einbettung mit $0,4$ bpp für LSB-*Matching* eine durchschnittliche Fehlerrate von $7,85\,\%$, für HUGO $13,55\,\%$ und für den bis dahin ungebrochenen *Edge Adaptive*-Algorithmus von Luo et al. $6,95\,\%$. Damit kann der Angriff mit SRM-Features und einem Ensemble-Klassifikator als *state-of-the-art*-Steganalyse angesehen werden, weshalb er in Kapitel 5 der vorliegenden Arbeit zur Evaluation der Stego-Verfahren herangezogen wird.

4.2.3 Weitere Aspekte

Aus dem maschinellen Lernen ist die Überanpassung an Trainingsdaten ein bekanntes Problem. Ein etwas anders situiertes Problem, das sich bei der BOSS-Challenge einstellte, ist der sogenannte *Warden's Nightmare*, der Albtraum des Angreifers: eine Diskrepanz zwischen den Quellen der Trainings- und der Test-Cover-Bilder. Die Bilder, an denen die BOSS-Teilnehmer ihre Angriffe testen sollten, stammten aus einer anderen Kamera-Zusammensetzung als die gegebenen Trainingsdaten, wodurch das Optimieren der Klassifizierer auf den Trainingsdaten die Performance im Test schlimmstenfalls sogar verschlechterte. Diesem Problem war auch durch Beifügen eigener Bilder zur Trainingsmenge nicht beizukommen, obwohl die Quelle der Testdaten nachzuempfinden versucht wurde [5]. Daraus lässt sich für die Steganalyse die wichtige Erkenntnis ziehen, dass der beste Angriff nur dann etwas nützt, wenn man die Cover-Quelle des angegriffenen Bildes kennt und für das Training hinreichend gut imitieren kann. Gerade das stellt sich aber als immer schwierigeres Problem heraus, da moderne Kameras immer komplexere interne Berechnungen mit immer stärker spezialisierter Hardware vornehmen, die nur unter großem Aufwand für große Trainingsmengen reproduzierbar sind.

Ein weiterer Punkt, der im Nachgang der BOSS-Challenge aufkam, ist HUGOs ungünstiger Umgang mit Schwellwerten T unterhalb des maximal möglichen Schwellwertes [9]. Pixel

werden stets in die Richtung um ±1 verändert, in der die Kosten dieser Operation am niedrigsten sind. Sorgt eine Operation dafür, dass die Differenz zweier Pixel den Schwellwert übersteigt, so wird die betreffende Pixelbeziehung vom Modell nicht mehr beachtet und die berechneten Gesamtkosten der Operation sinken. Umgekehrt kann das Verringern von Differenzen, die vorher nicht im Modell erfasst waren, eine Operation ungünstiger erscheinen lassen. Das führt zu einer Tendenz, Differenzen $d = T$ bzw. $d + 1 = T$ öfter zu inkrementieren als zu dekrementieren. Im Histogramm der Pixeldifferenzen zeigt sich dies umso deutlicher, je stärker texturiert das Cover-Bild ist[4]. Da eine solche Schwäche leicht auszubeuten ist, sollte HUGO stets mit maximalem Schwellwert eingesetzt werden.

4.3 Vorgeschlagene Erweiterungen

In seiner Ende 2012 eingereichten Diplomarbeit mit dem Titel *„Auswahl eines optimalen Stegobildes“* erörtert *Reduan Anton Franck* drei unterschiedliche Ansätze, den HUGO-Algorithmus zu verbessern [2]. Ziel aller dieser Ansätze ist eine Reduktion der Verzerrung, die durch die Einbettung entsteht. Die Ideen und Auswirkungen dieser Ansätze sollen im Folgenden zusammengefasst wiedergegeben werden.

4.3.1 Paralleles Einbetten mehrerer Nachrichtenbits

Der erste Vorschlag besteht darin, den Zeitpunkt der Bewertung einer Einbettung zu verändern. Der HUGO-Algorithmus mit Modellkorrektur berücksichtigt beim Einbetten zwar alle zuvor gemachten Veränderungen, zukünftige Änderungen hingegen können zu diesem Zeitpunkt noch nicht einbezogen werden. Jede Pixeländerung beeinflusst jedoch alle Einträge der Co-Occurrence-Matrizen, die zu Differenzenfolgen um dieses Pixel herum korrespondieren. Wenn sich auf diese Weise die Einträge ändern, die für die Entscheidung an einem früheren Pixel wichtig waren, wird diese Entscheidung im Nachhinein kompromittiert. So treten in jedem Bild Fälle auf, in denen der Algorithmus sich für eine Einbettungsoperation entschieden hat, diese aber rückwirkend durch andere Einbettungen doch zur schlechteren der beiden Optionen wird.

[4] Weil hier umso mehr hohe Pixeldifferenzen vorliegen, greift das Problem auch öfter – wider der Faustregel, dass stärker texturierte Bilder schlechter modellierbar und deshalb besser zum Verstecken geheimer Botschaften geeignet sind.

Dieses Phänomen lässt sich augenscheinlich nicht aus der Welt schaffen, ohne alle Einbettungen parallel zu betrachten, was äquivalent zum exponentiell aufwändigen Durchprobieren aller Kombinationen ist. Der in der Arbeit betrachtete Kompromiss besteht darin, immer eine Gruppe von n Einbettungsoperationen gleichzeitig zu bewerten – den Zeitpunkt der Bewertung aller Gruppenmitglieder also auf den Zeitpunkt der Bewertung des letzten Mitglieds zu verschieben.

Innerhalb dieser Gruppe können die beschriebenen Effekte auf diese Weise verhindert werden. Allerdings erhöht sich der Rechenaufwand enorm, da statt n Einbettungen 2^n Kombinationen von je n Einbettungen abgewogen werden müssen. Dazu kommt, dass die Betrachtung von Gruppen das Problem nur auf kleinen Abschnitten der gesamten Pixelmenge behebt und für jede bis auf die letzte Gruppe selbst wiederum auf unvollständigen Informationen beruht. Somit erbringt dieser teure Ansatz keine ausreichende Verringerung der Verzerrung und wurde daher wieder verworfen.

4.3.2 Sweeping

Einen wesentlich besseren Effekt zeigt die zweite Erweiterung, das so genannte *Sweeping*, zu Deutsch etwa „Fegen“ oder „Wischen“. Hier ist die Idee, nach Vollendung der Einbettungsroutine die veränderten Pixel noch einmal zu besuchen und zu testen, ob die dort gefällten Entscheidungen immer noch valide sind. Weil zu diesem Zeitpunkt alle notwendigen Änderungen bereits vollzogen sind, sind deren Einflüsse auf die Co-Occurrence-Matrizen bekannt und fließen in die neuen Entscheidungen ein.

Die in [2] gegebene Implementierung des Sweeping läuft wie folgt ab:

```
1  iteration_limit = nmb_of_pixels * payload / 0.4 * pcr;
2  for(iteration_limit)
3  {
4          i, j = PIXELS_TO_CHANGE[random];
5          Ysw = Y;
6          dbsw = D(X, Ysw);
7          invert_emb(Ysw(i, j));
8          dasw = D(X, Ysw);
9          if(dbsw < dasw)
10                 invert_emb(Y(i, j));
11 }
```

Zuerst wird anhand der Bildgröße, der Einbettungsrate und des neu eingeführten Pixel-Korrektur-Parameters *pcr* entschieden, wie viele Sweeping-Iterationen vorgenommen werden sollen (Zeile 1). In jeder dieser Iterationen wird dann ein zufälliges Pixel (i, j) aus der Menge aller beim Einbetten manipulierten Pixel gegriffen (Zeile 4) und die dort getroffene Einbettungsentscheidung probehalber umgedreht (Zeile 7). Eine Einbettung mittels -1 wird so zu $+1$ und umgekehrt. Wenn durch diese Änderung die Gesamtverzerrung D geringer wird, behält der Algorithmus sie bei, andernfalls wird die ursprüngliche Einbettung wieder hergestellt (Zeilen 9 & 10).

Auch die hier getroffenen Umkehrungsentscheidungen können, wie in 4.3.1 geschildert, durch spätere Iterationen und deren Einfluss auf die Co-Occurrence-Matrizen untergraben werden. Da aber jede Umkehrung optional ist und nur durchgeführt wird, wenn sie einen positiven Effekt hat, kann das Sweeping die Verzerrung nicht verschlimmern. Dennoch ist die letztendlich erzielte Verbesserung von der zufälligen Auswahl der besuchten Pixel abhängig und ihr genauer Wert ungewiss. In den dazu durchgeführten Versuchen zeigt sich eine Verringerung der Verzerrung um 5 % bis 50 %, mit umso besseren Ergebnissen, je geringer die Ausgangsverzerrung des Bildes war (Abbildung 4.4).

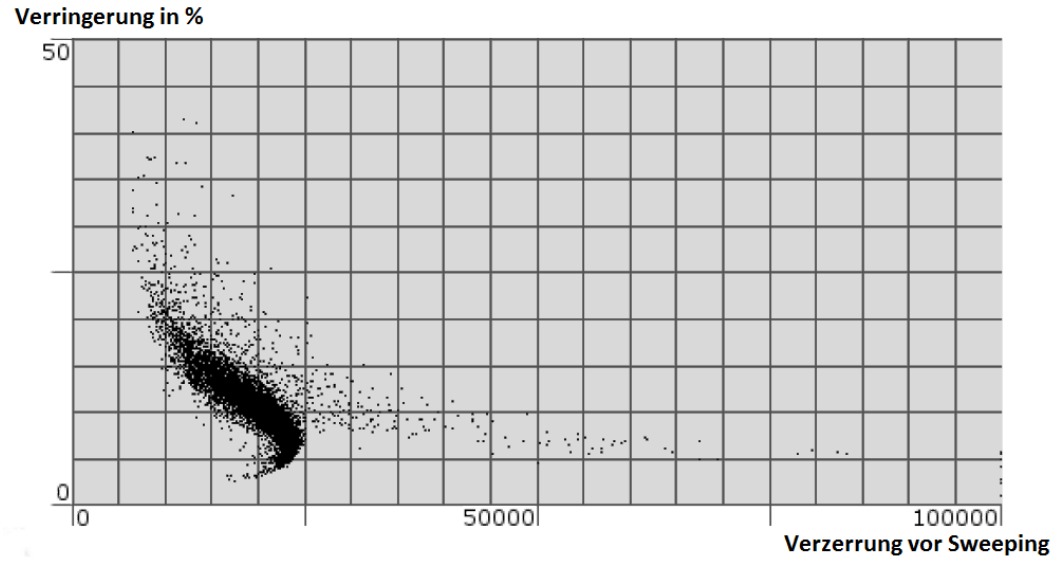


Abbildung 4.4: Verringerung der Verzerrung durch Sweeping für die 10.000 Bilder der BOSS-Datenbank [1]. Einbettung mit 0,4 bpp, $\gamma = \sigma = 1$, $T = 90$ und $pcr = 1,5$, Grafik entnommen aus [2].

4.3.3 Residuen höherer Ordnung

Als drittes wird in der Diplomarbeit eine Erweiterung des zugrunde liegenden Modells um die in Abschnitt 4.2.1 eingeführten *Residuen höherer Ordnung* vorgeschlagen. Diese werden nach [5] als Dreier-Folgen von Pixeldifferenzen höherer Ordnung (vgl. Abbildung 4.1) gebildet. Die Differenzenfolgen erster Ordnung entsprechen gerade den Einträgen der SPAM-Features bzw. der Co-Occurrence-Matrizen. Die Bildungsvorschriften für die Differenzen d_i^k der Ordnung $k \in \{1, 2, 3, 4\}$ ausgehend von einem Grauwert I_i lauten nach Umformung:

$$\begin{aligned} d_i^1 &= I_i - I_{i+1} \\ d_i^2 &= I_i - 2 \cdot I_{i+1} + I_{i+2} \\ d_i^3 &= I_i - 3 \cdot I_{i+1} + 3 \cdot I_{i+2} - I_{i+3} \\ d_i^4 &= I_i - 4 \cdot I_{i+1} + 6 \cdot I_{i+2} - 4 \cdot I_{i+3} + I_{i+4} \end{aligned}$$

Bei der Berechnung der Verzerrung D werden nun nicht mehr nur die Abweichungen im Auftreten der Features $\left(d_i^1, d_{i+1}^1, d_{i+2}^1\right)$ aller vier Richtungen gewichtet aufsummiert, sondern ebenso die Abweichungen im Auftreten der Features $\left(d_i^2, d_{i+1}^2, d_{i+2}^2\right)$, $\left(d_i^3, d_{i+1}^3, d_{i+2}^3\right)$ und $\left(d_i^4, d_{i+1}^4, d_{i+2}^4\right)$. Mit dem so erweiterten Modell erhofft man sich eine erhöhte Sicherheit durch größere Dimensionalität und eine weniger auf SPAM spezialisierte Minimierung der Verzerrung. Eine Einbeziehung der neuen Features ist schon deshalb angebracht, da sie in [5] zum Angriff auf den originalen HUGO-Algorithmus eingeführt wurden.

Die Berechnung der zusätzlichen Features bedeutet allerdings auch eine erhebliche Steigerung des Rechenaufwandes. Um dem entgegenzuwirken, wird in [2] eine Vorberechnung der Gewichte $w(d_1, d_2, d_3)$ motiviert, da deren ständige *ad hoc*-Berechnung beim Ermitteln der Verzerrung überaus rechenintensiv ist. Dieser Rechenaufwand wird einmalig an den Anfang der Bearbeitung gestellt und gegen einen konstanten Speicheraufwand eingetauscht. Bei einem Schwellwert von $T = 90$ reduziert sich der Rechenaufwand für Gewichte damit auf etwa ein Zehntel, was sich bei sequentieller Einbettung in mehrere Bilder umso mehr rentiert, da die Gewichte unabhängig vom Bildinhalt sind und gleich weiterverwendet werden können.

Der Schwellwert T selbst ist leider ein zusätzliches Problem der Residuen höherer Ordnung. Möchte man das in Abschnitt 4.2.3 beschriebene ungünstige Verhalten für Schwellwerte $T < 255$ vermeiden, indem man T auf 255 setzt, dann wächst der Speicherbedarf

der Co-Occurence-Matrizen über alle derzeit vertretbaren Grenzen. Während Residuen d_i^1 erster Ordnung $(2 \cdot T + 1) = 511$ verschiedene Werte annehmen können, steigt diese Zahl auf $(2 \cdot 510 + 1) = 1021$ für d_i^2, $(2 \cdot 1020 + 1) = 2041$ für d_i^3 und $(2 \cdot 2040 + 1) = 4081$ für d_i^4. Eine Matrix aller Dreier-Folgen solcher Residuen beinhaltet $(4081)^3 \approx 67.97 \cdot 10^9$ Elemente, was für Integer-Werte einem Speicherbedarf von $253,2$ Gigabyte entspräche. Mit allen vier Residuenordnungen lässt sich HUGO also nicht sinnvoll mit maximalem Schwellwert nutzen.

5 Evaluation der Erweiterungen

Das Ziel dieser Arbeit ist die Bewertung des Sicherheitsgewinns durch die in Kapitel 4.3 beschriebenen Erweiterungen für die steganographische Einbettung mit HUGO. Zu diesem Zweck sollten existierende steganalytische Ansätze eingesetzt und ihr Erfolg beim Entdecken von HUGO mit und ohne Erweiterungen verglichen werden.

5.1 Versuchsanordnung

Als Grundlage aller Versuche wurde die Version 1.01 der BOSS-Bilddatenbank gewählt [1]. Aufgrund der zur Verfügung stehenden Rechenkapazität beschränken sich die Versuche auf die ersten 2.500 Bilder dieses Bestands.

Aus den Cover-Bildern wurden zwei mal vier Sätze von Stego-Bildern generiert:

(a) mit der ursprünglich zur BOSS-Challenge bereit gestellten HUGO-Implementierung für den Schwellwert $T = 90$,

(b) mit dem selben Algorithmus und $T = 255$,

(c) mit der in [2] gegebenen Implementierung der Erweiterung um *Sweeping* und dem Schwellwert $T = 255$ und

(d) mit der ebenfalls dort implementierten Erweiterung um *Residuen höherer Ordnung* inklusive *Sweeping* bei $T = 90$;

je einmal mit den Einbettungsraten $a = 0,25\,\text{bpp}$ und $a = 0,4\,\text{bpp}$.[1] Die Parameter für die Gewichtsfunktion wurden bei ihren Standardwerten $\gamma = \sigma = 1$ belassen.

[1] Die Stego-Bilder des originalen Algorithmus mit $T = 90$ und $a = 0,4$ wurden direkt den Materialen zur BOSS-Challenge entnommen.

Für den Angriff wurden die vom *Digital Data Embedding Laboratory* der Universität Binghamton angebotenen MATLAB-Skripte zur Feature-Extraktion sowie die dort verfügbare Implementierung eines Ensemble-Klassifikators adaptiert.

Für alle Stego-Bildmengen wurden die 686-dimensionalen SPAM-Features mit $T = 3$ und Markov-Ketten zweiter Ordnung (siehe Kapitel 3) sowie die vollständigen, 34.671-dimensionalen SRM-Features (siehe Kapitel 4.2.2) extrahiert und für die Klassifizierung aufbereitet. Damit sind der alte, HUGO zugrunde liegende Angriff und ein starker moderner Angriff vertreten.

Zusätzlich wurden zwei Teilmengen der SRM-Features gebildet, die den in der Erweiterung um *Residuen höherer Ordnung* betrachteten Merkmalen ähneln. Einmal sind das alle in Abbildung 4.3 gezeigten SRM-Teilmodelle vom Typ *spam* (insgesamt 5.746 Features); und einmal nur die Teilmodelle vom Typ *spam* aus den Klassen **1st**, **2nd** und **3rd** (2.704 Features). Diese beiden Modelle werden im weiteren SRM5746 und SRM2704 genannt. Sie verzichten auf die Residuen vom Typ *minmax* und konzentrieren sich auf jene, deren Verzerrung in der Erweiterung minimiert wird. Die Performance des erweiterten HUGO-Algorithmus sollte unter diesen Angriffen daher besser sichtbar werden als unter dem vollständigen SRM-Angriff.

Mit dem Ensemble-Klassifikator wurde nun für jede extrahierte Feature-Menge eine Testreihe mit 25 verschiedenen zufälligen Aufteilungen in 80 % Trainings- und 20 % Testdaten durchgeführt. Die Ensemble-Parameter d_{sub} und L wurden wie in Abschnitt 4.2.2 beschrieben automatisch gelernt. Berechnet wurde stets der Anteil falsch klassifizierter Testdaten als Maß für den Erfolg oder Misserfolg der geheimen Einbettung.

Alle verwendeten Skripte und Implementierungen sowie die entstandenen Stego- und Feature-Dateien befinden sich neben den vollständigen Test-Ergebnissen auf dem der Arbeit beigelegten Datenträger.

5.2 Ergebnisse

In den Diagrammen 5.1 bis 5.4 sind die durchschnittlichen Fehlerraten der beschriebenen Testreihen für je einen Angriff und alle acht Einbettungs-Konstellationen dargestellt. Dabei gelten die Abkürzungen **SW** für die Erweiterung um *Sweeping* und **SW+R** für die Erweiterung sowohl um *Sweeping* als auch um *Residuen höherer Ordnung*. Die obere Diagramm-Grenze von 50 % entspricht der Fehlerrate einer Zufallsentscheidung.

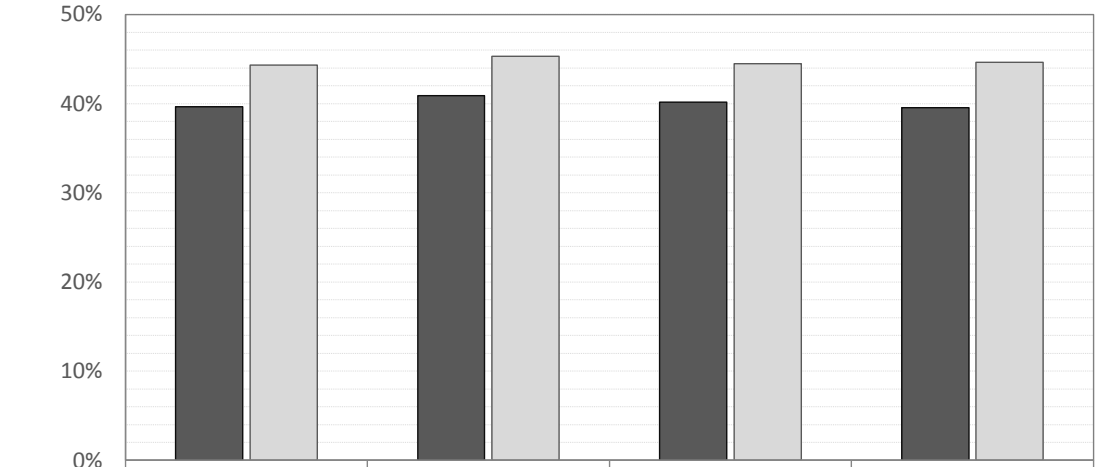
5.3 Auswertung und Interpretation

Der Anteil falsch zugeordneter Bilder ist erwartungsgemäß für alle Angriffe gringer, wenn die Einbettungsrate höher war und deshalb mehr Pixel bei der Einbettung verändert wurden. Spannender ist die Tatsache, dass sich die Fehlerraten eines Angriffs für die vier HUGO-Varianten bei fester Einbettungsrate kaum unterscheiden. Offensichtlich haben weder die Erweiterungen noch der Schwellwert T für den Erfolg der untersuchten Angriffe eine annähernd so große Bedeutung wie die Einbettungsrate. Für den Schwellwert mag das daran liegen, dass die betrachteten Angriffe das in Abschnitt 4.2.3 beschriebene Verhalten nicht explizit ausnutzen. Zielgerichtetere Analysen der $T = 90$-Schwäche würden dem Schwellwert sicher einen größeren Einfluss auf die Fehlerrate geben.

Die ausgebliebene Steigerung der Fehlerrate durch die beiden Erweiterungen zeigt, dass diese für die betrachteten Angriffe keinen Sicherheitsgewinn bringen, ist aber schwieriger zu erklären. Während die Einbettung mit *Residuen höherer Ordnung* und *Sweeping* für den SPAM-Angriff noch eine sichtbare Verbesserung bringt, hat sie bei der Analyse mit den drei SRM-Modellen sogar einen – wenn auch geringen – durchgehend negativen Effekt. Da die Verhältnisse der Fehlerraten für die reduzierten Modelle SRM5746 und SRM2704 denen des vollständigen SRM-Angriffs gleichen, kann dieser Effekt nicht von den *minmax*-Residuen herrühren.

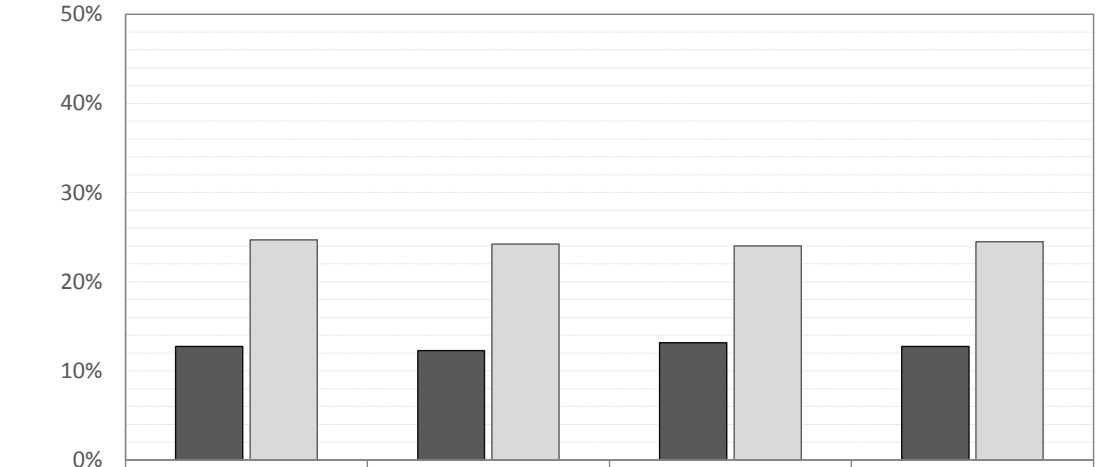
Eine Ursache für die schlechtere Performance könnte sein, dass die betrachteten Residuen in den SRM-Modellen als Viererfolgen ausgewertet werden, wohingegen die HUGO-Erweiterung die Verzerrung für Dreierfolgen minimiert. Zusammen mit den quantisierten Varianten der Residuen erlaubt das dem Angriff, Merkmalsverzerrungen zu registrieren, die bei der Einbettung nicht beachtet worden sind. Darüber hinaus spielen Residuen vierter Ordnung in den Angriffen keine Rolle, weshalb ihre Einbeziehung bei der Einbettung mehr von der Optimierung der anderen Ordnungen abträgt, als sie selbst hilft.

Die Einbettung nur mit der Erweiterung um *Sweeping* weist eine etwas bessere Sicherheit gegenüber SRM-Angriffen auf. Dieser kleine Gewinn wird jedoch, ebenso wie die zuvor besprochenen Verschlechterungen, in seiner Signifikanz noch durch die Streuung der Test-Ergebnisse relativiert, die mit der Zufallskomponente des Ensemble-Klassifikators einhergeht.



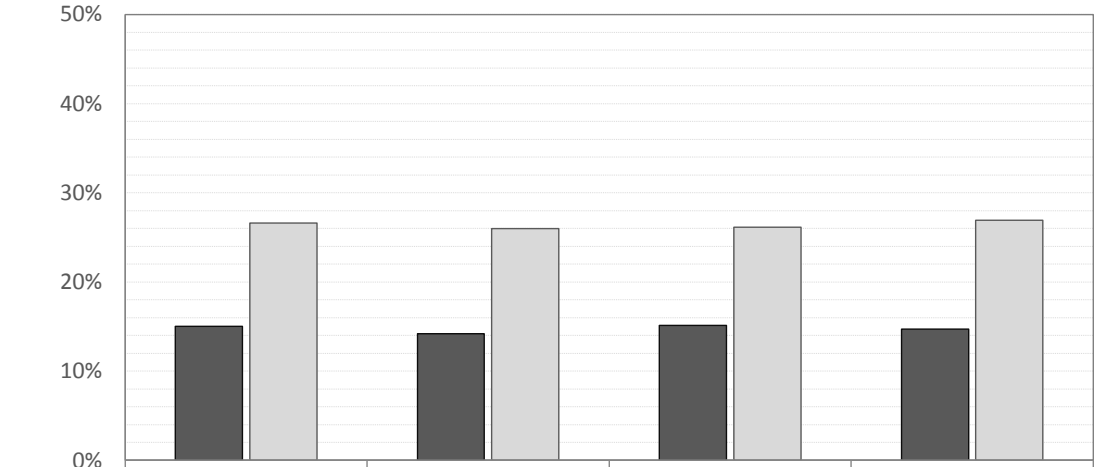
	SW | T = 255	SW+R | T = 90	T = 255	T = 90
0,40 bpp	39,66%	40,88%	40,18%	39,52%
0,25 bpp	44,30%	45,31%	44,50%	44,64%

Abbildung 5.1: Ensemble-Angriffe mit Spam-Features und 80 % Trainingsdaten



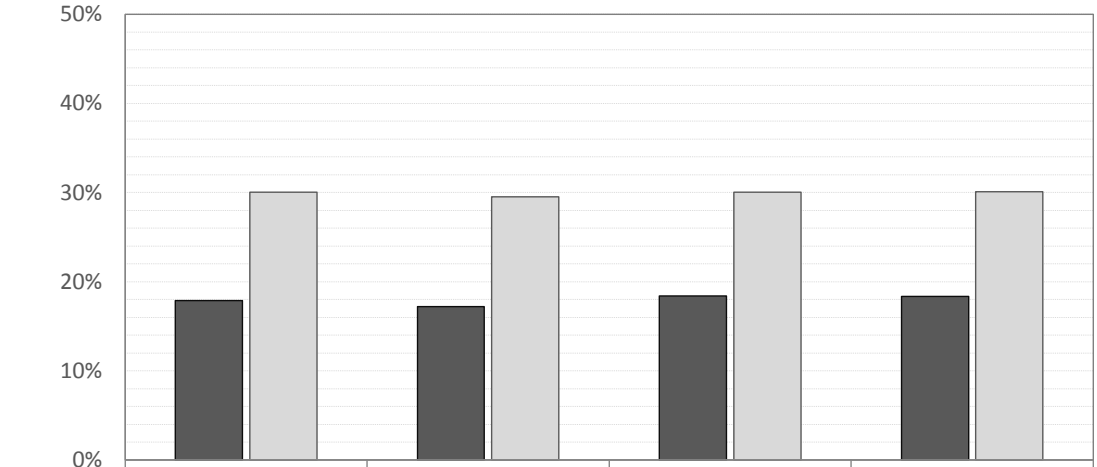
	SW | T = 255	SW+R | T = 90	T = 255	T = 90
0,40 bpp	12,72%	12,26%	13,16%	12,74%
0,25 bpp	24,67%	24,23%	24,03%	24,48%

Abbildung 5.2: Ensemble-Angriffe mit allen Srm-Features und 80 % Trainingsdaten



	SW | T = 255	SW+R | T = 90	T = 255	T = 90
0,40 bpp	15,02%	14,20%	15,12%	14,72%
0,25 bpp	26,62%	25,98%	26,12%	26,91%

Abbildung 5.3: Ensemble-Angriffe mit SRM5746-Features und 80 % Trainingsdaten



	SW | T = 255	SW+R | T = 90	T = 255	T = 90
0,40 bpp	17,88%	17,22%	18,38%	18,35%
0,25 bpp	30,03%	29,50%	30,03%	30,10%

Abbildung 5.4: Ensemble-Angriffe mit SRM2704-Features und 80 % Trainingsdaten

6 Zusammenfassung und Ausblick

Wie sich herausstellt, haben die in [2] vorgeschlagenen HUGO-Erweiterungen keinen ausgeprägten Einfluss auf die Testergebnisse der untersuchten Angriffe. In drei verschiedenen Varianten des Angriffs mit dem *Spatial Domain Rich Model* (SRM) war teilweise sogar eine kleine Verschlechterung der Sicherheit zu verzeichnen. Dieses Ergebnis ist mit hoher Wahrscheinlichkeit darauf zurück zu führen, dass die Angriffe sich auf Bildmerkmale stützen, die von den Erweiterungen nicht erfasst werden. Für den Angriff mit den veralteten SPAM-Features brachte die Erweiterung um *Residuen höherer Ordnung* eine kleine Verbesserung der Sicherheit. Da die moderne Steganalyse auf breiter gefächerte Bildmodelle (wie SRM) setzt, ist das aber von geringem Nutzen. Weiterhin hat sich gezeigt, dass auch die Steigerung des Schwellwerts auf $T = 255$ keinen nennenswerten Effekt auf die Fehlerrate der untersuchten Angriffe hat.

Um HUGO im Sinne der vorgeschlagenen Erweiterungen auch gegenüber modernen, hochdimensionalen Angriffen zu stärken, müssten bei der adaptiven Einbettung möglichst alle neuen Features dieser Angriffe einbezogen werden. Bei einer so großen Menge unterschiedlicher Features ist es allerdings fraglich, ob sich die Verzerrung zwischen Cover- und Stego-Bild überhaupt wirksam für alle optimieren lässt. Die Minimierung der Verzerrung eines Bildmerkmals wird immer auf Kosten eines anderen Merkmals geschehen, das dadurch auffälliger wird. Es ist denkbar, dass selbst die bestmögliche Lösung dieses Problems keinen signifikanten Sicherheitsgewinn mehr bringen kann. Um dies zu überprüfen, könnte man beispielsweise eine Erweiterung von HUGO um alle SRM-Residuen implementieren und gegen besagte Angriffe testen.

Eine andere Möglichkeit zur Erweiterung wird von den HUGO-Autoren selbst in [12] angedeutet. Die Kosten $\rho_{i,j}$ einer Einbettung werden dort nur approximiert, um Definition 4.1 zu genügen. Der durch diese Annäherung verursachte Fehler wird zwar durch den *Modellkorrektur*-Schritt weitestgehend ausgeglichen, eine andere Lösung für Einbettungskosten und Verzerrungsmaß könnte den Algorithmus aber möglicherweise trotzdem verbessern.

Interessanterweise haben die untersuchten Angriffe in der Praxis eine bedeutende Schwäche gemeinsam: ihre Abhängigkeit von den verfügbaren Trainingsdaten. Für die verwendeten maschinellen Lernverfahren müssen diese in großer Zahl vorliegen und die Eigenschaften des angegriffenen Bildes exakt wiederspiegeln. Speziell der zweite Punkt, der bereits in Abschnitt 4.2.3 angesprochen wurde, lässt sich für die Steganographie ausnutzen. Indem eine schwer reproduzierbare Bildquelle gewählt wird, macht man es dem Angreifer unmöglich, passende Trainingsdaten zu finden oder zu generieren und provoziert das in [5] als *Warden's Nightmare* bezeichnete Problem. Wie man am besten zu solchen Bildquellen kommt und wie groß der Effekt für die Steganalyse tatsächlich ist, scheint ein untersuchenswertes Thema zu sein.

Letztlich zeigen die hier wiedergegebenen Versuche sehr deutlich, dass die Einbettungsrate der wichtigste Faktor für die Unentdeckbarkeit HUGOs ist. Um eine Nachricht sicher zu verstecken, ist also vor allem ein bezüglich der Länge der Nachricht ausreichend großes Bild zu wählen – dann hat auch ein Angriff mit allen SRM-Features und guten Trainingsdaten keine große Aussicht auf Erfolg.

Literaturverzeichnis

[1] Bas, Patrick, Filler, Tomáš und Pevný, Tomáš: *Break Our Steganographic System — the ins and outs of organizing BOSS.* In: Filler, Tomáš (Herausgeber): *Information Hiding, 13th International Workshop*, Lecture Notes in Computer Science, Prague, Czech Republic, 18. – 20. Mai 2011. Springer-Verlag, New York.

[2] Franck, Reduan Anton: *Auswahl eines optimalen Stegobildes.* Diplomarbeit, Technische Universität Dresden, 2012.

[3] Fridrich, Jessica und Filler, Tomáš: *Practical methods for minimizing embedding impact in steganography.* In: *Proceedings SPIE, EI, Security, Steganography, and Watermarking of Multimedia Contents IX*, Seiten 2–3, San José, Kalifornien, USA, 29. Januar – 1. Februar 2007.

[4] Fridrich, Jessica und Kodovský, Jan: *Rich Models for Steganalysis of Digital Images.* IEEE Transactions on Information Forensics and Security, 7(3):868–882, 2012.

[5] Fridrich, Jessica, Kodovský, Jan, Holub, Vojtěch und Goljan, Miroslav: *Breaking HUGO – the process discovery.* In: *Information Hiding, 13th International Workshop, Lecture Notes in Computer Science*, 2011.

[6] Gül, Gökhan und Kurugöllü, Fatih: *A New Methodology in Steganalysis: Breaking Highly Undetectable Steganograpy (HUGO).* In: Filler, Tomáš, Pevný, Tomáš, Craver, Scott und Ker, Andrew (Herausgeber): *Information Hiding*, Band 6958 der Reihe *Lecture Notes in Computer Science*, Seiten 71–84. Springer Berlin Heidelberg, 2011.

[7] Goljan, Miroslav, Fridrich, Jessica und Holotyak, Taras: *New blind steganalysis and its implications.* In: *Proceedings of the SPIE, Security, Steganography, and Watermarking of Multimedia Contents VI*, Seiten 1–13, 2006.

[8] Kodovský, Jan und Fridrich, Jessica: *Calibration revisited.* In: *Proceedings of the 11th ACM workshop on Multimedia and security*, Seiten 63–74, New York, NY, USA, 2009. ACM.

[9] Kodovský, Jan, Fridrich, Jessica und Holub, Vojtěch: *On dangers of overtraining steganography to incomplete cover model.* In: *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security*, Seiten 69–76, New York, NY, USA, 2011. ACM.

[10] Kodovský, Jan, Fridrich, Jessica und Holub, Vojtěch: *Ensemble Classifiers for Steganalysis of Digital Media.* IEEE Transactions on Information Forensics and Security, 7(2):432–444, 2012.

[11] Pevný, Tomáš, Bas, Patrick und Fridrich, Jessica: *Steganalysis by Subtractive Pixel Adjacency Matrix.* In: *Proceedings of the 11th ACM Workshop on Multimedia and Security*, Seiten 75–84, New York, USA, 7. – 8. September 2009. ACM.

[12] Pevný, Tomáš, Filler, Tomáš und Bas, Patrick: *Using high-dimensional image models to perform highly undetectable steganograpy.* In: Böhme, Rainer, Fong, Philip W.L. und Safavi-Naini, Reihaneh (Herausgeber): *Information Hiding, 12th International Workshop*, Lecture Notes in Computer Science, Seiten 161–177, Calgary, Kanada, 28. – 30. Juni 2010.

[13] Westfeld, Andreas und Pfitzmann, Andreas: *Attacks on Steganographic Systems.* In: *Proceedings of the Third International Workshop on Information Hiding*, IH '99, Seiten 61–76, London, UK, 2000. Springer-Verlag.

Erklärung

Hiermit erkläre ich, dass ich die am 03. 04. 2013 eingereichte Bachelorarbeit zum Thema *Steganalytische Untersuchung einer Erweiterung des steganographischen Algorithmus* HUGO unter Betreuung von Dr. Elke Franz selbstständig erarbeitet, verfasst und Zitate kenntlich gemacht habe. Andere als die angegebenen Hilfsmittel wurden von mir nicht benutzt.

Dresden, den 03. 04. 2013 Jakob Kruse