Gottfried Wilhelm Leibniz Universität Hannover
Institut für Informationsverarbeitung

Bachelorarbeit

Videosignaturen für die Detektion von Videoinhalten gleichen Ursprungs

Gerion Entrup

lizensiert unter CC-BY-NC-SA 4.0

Betreuer:	Dipl.-Ing. Marco Munderloh
Erstprüfer:	Prof. Dr.-Ing. Jörn Ostermann
Zweitprüfer:	Prof. Dr.-Ing. Bodo Rosenhahn

Hannover, den 02. Oktober 2014

Institut für
Informationsverarbeitung
Leibniz Universität Hannover

Appelstr. 9A - 30167 Hannover
Sekretariat: +49 511 762 - 5316
Telefax: +49 511 762 - 5333
E-mail: office@tnt.uni-hannover.de
URL: http://www.tnt.uni-hannover.de

2014-06-01

Bachelorarbeit

für

Gerion Entrup

2790480

Videosignaturen für die Detektion von Videoinhalten gleichen Ursprungs

Die MPEG-7 Video Signature Tools for Content Identification sind ein von der MPEG standardisiertes Verfahren zur Erstellung von Signaturen von Videosequenzen. Signaturen stellen eine Art Hashwert dar, der den Inhalt des Materials eindeutig identifiziert. Die Signaturen sollen dabei für gleiche Inhalte auch gleiche Signaturen generieren, unabhängig von dem Format oder der verwendeten Codierung. Gleichzeitig sollen die Signaturen robust sein gegenüber Manipulationen wie z.B. Rauschen, Bilddeformation oder Codierartefakte. Dies ermöglicht die Wiedererkennung von Videoinhalten unabhängig von der Verfügbarkeit von Metainformationen.

Herr Entrup erhält die Aufgabe, den MPEG-7 Standard für Video Signaturen als einen Filter in das FFmpeg-Videoprocessingframework zu integrieren. Die Korrektheit der Implementierung soll anhand von Referenzdaten verifiziert und die Klassifikationsleistung des Verfahrens bei verschiedenen Manipulationen untersucht werden. Dabei sind Schwächen für bestimmte Angriffe zu identifizieren und Lösungsvorschläge zur Erhöhung der Robustheit gegen solche Angriffe zu evaluieren.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

_____

Gerion Entrup

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Eventuell kennen Sie das Problem: Sie durchstöbern Ihre Sammlung von TV-Aufnahmen auf Ihrer Festplatte und finden die Datei „ZDF-2014-04-21-19:57.mkv“ oder noch schlimmer „Aufnahme5.mkv“. Oder Sie schauen sich ein YouTube-Video an, klicken auf den nächsten Link und dasselbe Video erscheint erneut (und das in besserer Qualität). Das würde nicht passieren, wenn ein Computer dazu in der Lage wäre, Filme, bzw. allgemeiner ausgedrückt, Videosequenzen automatisch zu erkennen.

Glücklicherweise gibt es bereits Lösungen für diese Probleme, die unter dem Begriff *Videosignaturen* zu finden sind. Der allgemeine Ansatz dieser Signaturen ist es, eine bestehende Sammlung mit bekannten Videosequenzen zu katagolisieren und dann bei neuen, unbekannten Videosequenzen mit dieser Sammlung zu vergleichen.

Der Sinn einer Videosignatur liegt dabei hauptsächlich in der Extraktion von Merkmalen in einer Videosequenz mit z. B. bekannten Metadaten und dem anschließenden Vergleich mit einer anderen unbekannten Sequenz, um auf Gleichheit zu prüfen.

Die Anwendungen für Videosignaturen liegen in diesen Bereichen [1, 2]:

- Automatische Klassifizierung von Videosequenzen mit Metadaten
- Automatische Erkennung von urheberrechtlich geschütztem Material in Filmdatenbanken wie YouTube
- Automatische Erkennung von Dubletten im Sinne einer Deduplikation

Eng verwandt mit Videosignaturen sind Audiofingerprinting-Techniken, die ebenfalls dieselben Anwendungsbereiche in Bezug auf Audiodaten ermöglichen [3].

Videosignaturen sind nicht das einzige Mittel, besagte Aufgabenstellung zu lösen. Grundsätzlich existieren zwei Ansätze für die automatische Klassifikation von Videos [4]:

1. Im Vorhinein markieren
2. Im Nachhinein erkennen

Der erste Ansatz markiert das Video bereits bei der Erzeugung mit Metadaten oder einem Wasserzeichen. Das setzt aber entsprechende Hardware bzw. Aufnahmesoftware voraussetzt und kann nicht zwangsläufig gewährleistet werden.

Der zweite Ansatz analysiert das Video und bildet daraus eine Videosignatur, die dann mit einer anderen Signatur (die man soeben gebildet hat oder die in einer Datenbank gespeichert ist) verglichen werden kann. Videosignaturen kann man sich also wie Fingerabdrücke vorstellen.

1.2 Sinn und Ziel dieser Arbeit

In dieser Arbeit werde ich mich mit den bestehenden Videosignaturtechniken auseinandersetzen (Abschnitt 2) und den daraus entstandenen Standard der MPEG[1] eingehend untersuchen (Abschnitt 3), sowie in FFmpeg[2] implementieren (Abschnitt 4). Außerdem werde ich mit eigenen Messungen Schwachstellen des bestehenden Standards aufdecken und eine vom Standard abgeleitete eigene verbesserte Signatur entwickeln (Abschnitt 5) und umfassend evaluieren (Abschnitt 6).

[1] „Moving Picture Experts Group“, eine Gruppe von Fachleuten, die sich mit der Sammlung und Standardisierung von Verfahren in Bezug auf Audio und Video befasst [5]

[2] Eine Software, die Verfahren im Bereich Multimedia implementiert.

2 Videosignaturen

Videosignaturen bezeichnen Verfahren, die in der Lage sind, ein Video eindeutig zu kennzeichnen und zwar auf Basis der inhaltlichen Daten. Allen Verfahren liegen dabei zwei Teile zugrunde:

1. Extraktion der Signatur aus dem Inhalt des Videos.
2. Vergleichen mehrerer Signaturen um die Gleichheit der Videos zu prüfen.

Das Extraktionsverfahren versucht immer eindeutige Merkmale aus dem Inhalt des Videos zu gewinnen und möglichst kompakt zu speichern. Generell gibt es verschiedene Ansätze, die später noch detailliert dargestellt werden.

Das Vergleichen von mehreren Videosignaturen (auch *Lookup* genannt) muss nicht unbedingt lokal erfolgen, sondern kann auch mit einer Datenbank, die im Netzwerk liegt, passieren. Außerdem muss u.U. gegen eine sehr große Anzahl anderer Signaturen geprüft werden. Darum ist es hier wichtig, möglichst wenig Datenrate und Rechenleistung zum Prüfen zu verbrauchen. Teilweise bieten sich auch schrittweise Verfahren an, die zuerst mit einem groben Verfahren Kandidaten suchen, um dann detailliert nur noch die kleine Menge der Kandidaten zu prüfen.

2.1 Anforderungen an die Signatur

Die MPEG hat für den Standardisierungsprozess Anforderungen erarbeitet, die hier übernommen werden sollen [6]. Im Folgenden sind diese einzeln aufgelistet:

Einzigartigkeit/Reproduzierbarkeit Die Videosignatur soll einen einzigartigen Fingerabdruck für jede gleiche Sequenz darstellen. Wenn man die Signatur mehrfach extrahiert, soll sie immer identisch sein.

Robustheit Die Videosignatur soll verschiedenen Modifikationen und Verfälschungen der Videosequenz standhalten. So sollte man ohne Probleme z. B. Untertitel überlagern oder eine Skalierung durchführen können.

Verschiedenartigkeit Inhaltlich verschiedene Videosequenzen sollen verschiedene Videosignaturen ergeben. So sollen beispielsweise zwei Videosequenzen, in denen die Sahara vorkommt, die aber sonst verschieden sind, nicht als gleich erkannt werden.

Schneller Lookup Das Verfahren, um zwei Videosignaturen auf Gleichheit zu überprüfen, soll eine geringe Komplexität aufweisen und somit eine schnelle Durchführung erlauben.

Schnelle Extraktion Das Verfahren, um die Videosignatur zu extrahieren, soll eine geringe Komplexität aufweisen und somit eine schnelle Durchführung erlauben.

Kompaktheit Die Videosignaturen sollen möglichst klein sein.

Unversehrtheit der Quelle Die Quelldateien sollen für die Extraktion in keiner Weise verändert werden müssen.

Eigenständigkeit Eine Videosignatur soll die komplette Information beinhalten, die zum Vergleich benötigt wird. D.h. zwei Signaturen können ohne die originale Videosequenz verglichen werden.

Unabhängigkeit Die Videosignatur soll unabhängig von der Videocodierung oder dem Format sein.

Diese Anforderungen sind unmöglich alle zur Gänze zu erfüllen, sodass immer ein Kompromiss eingegangen werden muss. So gilt generell: Je besser die Features im Video erkannt werden (höhere Robustheit), desto länger ist die Rechenzeit (langsame Extraktion). Je besser die Signatur die Sequenz beschreibt (Einzigartigkeit und Verschiedenartigkeit), desto länger wird sie (Kompaktheit und langsamer Lookup).

2.2 Klassifikation der Extraktionsverfahren

Videosignaturen können nach zwei Merkmalen unterschieden werden, der Merkmalsart und der Dimension [7].

Allen Videosignaturen ist gemein, bestimmte Merkmale auf Framebasis zu extrahieren. Diese lassen sich aber in blockbasierte, keypointbasierte oder Verfahren auf globaler Ebene unterteilen.

Weiterhin spielt bei Videosignaturen die Dimension eine Rolle. So können zeitliche oder räumliche Merkmale beachtet werden.

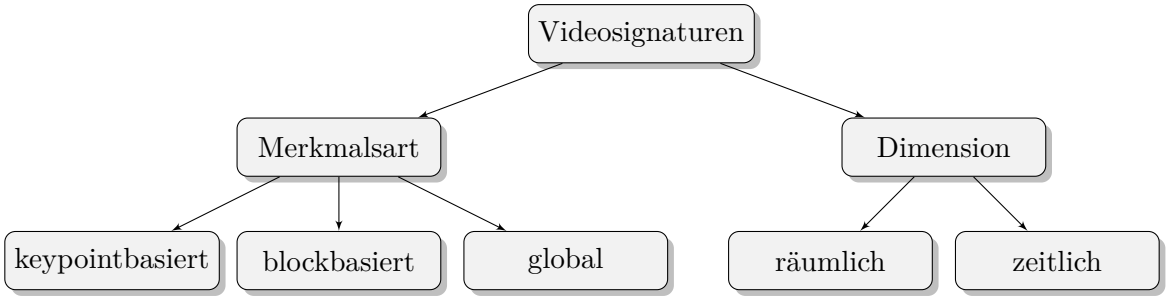


Abbildung 1: Klassifikation der Extraktionsverfahren

Verfahren auf globaler Ebene Für die Merkmalsextraktion auf Frameebene können globale Merkmale beachtet werden. So wird beispielsweise in [8] die Featureextraktion auf Basis der CLD (Color Layout Descriptor) durchgeführt. Dazu wird im für das gesamte Bild pro Kanal (bei einem Y/CB/Cr-Farblayout) eine DCT durchgeführt und davon die per Zickzack-Scan ersten 6 Werte genommen. Diese so insgesamt 18 Werte bilden die Basis für die weitere Signatur. Eine ähnliche Methode benutzt [9], die ebenfalls den CLD verwenden, zusätzlich aber noch mit dem

EHD (Edge Histogram Descriptor) und dem SCD (Scalable Color Descriptor) kombinieren, zwei weiteren globalen Verfahren[3].

Weiteres Beispiele für ein globales Verfahren sind [10], bei dem eine NMF[4] als Transformationsmethode genutzt wird oder [12], bei dem eine Haar-Wavelet-Transformation zum Einsatz kommt. Auch [4] kann man zu den globalen Verfahren zählen. Dort werden bei jedem Frame (nach Skalierung) die Pixel, die auf sechs Diagonalen liegen, extrahiert.

Globale Verfahren können zu kompakten Signaturen führen und sind je nach Verfahren auch rechentechnisch effizient. Allerdings sind sie nur begrenzt robust gegen globale Angriffe wie eine Farbänderung und haben zumeist weniger Aussagekraft.

Keypointbasierte Verfahren Wenn bei einem Signaturverfahren Punkte im Bild erkannt werden, die besonders auffällig sind, beispielsweise scharfkantige Ecken, so werden diese den keypointbasierten Verfahren zugeordnet. SIFT-Features[5], bezeichnen beispielsweise eine Methode, die auffällige Punkte findet. Sie werden in [15] oder [16] verwendet. Eine sehr ähnliche Methode benutzt [17]. Dort werden auf einem DOG-Bild auffällige Extrema als Merkmalspunkte verwendet[6]. Eine alternative Methode sind Harris-Interest-Points[7], die in [19] (in einer modifizierten Variante und nur bei Keyframes) verwendet werden.

Keypointbasierte Verfahren sind zumeist sehr rechenintensiv. Dafür sind sie aber weitgehend immun gegen geometrische Transformationen wie Rotation oder Translation und in begrenztem Maß sogar gegen perspektivische Verzerrungen.

Blockbasierte Verfahren Basiert die Extraktion auf mathematischen Operationen auf Blöcken, zusammenhängenden rechteckigen Pixelmengen innerhalb des Bildes, so ist das Verfahren blockbasiert. Diese werden u.a. in [20] oder [2] verwendet. Bei letzterem wird jeder Frame in einen 4×4 Blöcke zerlegt und dann die Signatur auf Basis der Korrelation der Elemente jedes Unterblocks berechnet. Auch der im Weiteren verwendete und detailliert beschriebene MPEG-7 Standard ist blockbasiert.

Blockbasierte Verfahren zeichnen sich hauptsächlich durch eine sehr schnelle Berechenbarkeit aus, können aber dennoch auf lokale Bildinformationen eingehen, was den globalen Verfahren nicht möglich ist. Sie sind außerdem nur begrenzt robust gegen geometrische Operationen wie einer Rotation oder einer Skalierung.

Räumliche Verfahren Räumliche Verfahren zeichnen sich durch die ausschließliche Extraktion räumlicher Merkmale aus bzw. die Nichtbeachtung zeitlicher Veränderungen. Für gewöhnlich werden die aus dem Frame extrahierten Merkmale aneinandergehängt und erst im Lookup als zeitliche Abfolge betrachtet.

Beispiele für räumliche Verfahren sind [8] oder [2].

[3] Sowohl CLD, wie auch EHD und SCD gehören zum MPEG-7 Standard [8, 9]

[4] „Nonnegative Matrix Factorization“, eine Methode um eine Matrix in zwei nichtnegative Komponenten zu zerlegen [11].

[5] Ein patentgeschützer Algorithmus von David G. Lowe. SIFT steht für „Scale-invariant feature transform“ [13, 14].

[6] „Difference of Gaussians“, ein Algorithmus, bei dem die Differenz des Originalbildes mit dem gaußgefilterten Bild verglichen wird. SIFT basiert auf der gleichen Methode, ist aber umfangreicher [13, 18].

[7] Ein Algorithmus, der Ecken detektiert, auch unter dem Namen „Harris Corners“ zu finden [14].

Zeitliche Verfahren Bei zeitlichen Verfahren werden die Bildänderungen im Bezug auf die Zeit verwendet. So wird im eben erwähnten [20] die Merkmalsextraktion nur auf der Differenz von zwei aufeinanderfolgenden Bildern vorgenommen. Auf dem Differenzbild wird dann ein einfaches Blockverfahren angewandt, das das Bild in 4 Teile teilt. Andere Verfahren operieren nur auf Keyframes (die basierend auf zeitlichen Änderungen gewählt werden), wie [19].

Neben den hier vorgestellten Verfahren gibt es noch weitere, die mehrere Kategorien kombinieren. Dist gilt z. B. für den MPEG-7 Standard selbst, der zusätzlich zu einer Extraktion auf räumlicher Basis noch eine zeitliche Komponente beinhaltet. Auch zu nennen wären hier [21], das auf zeitlicher Ebene Keyframes mittels eines Binärbaums extrahiert. Aus diesen wird mithilfe eines MRF (Markov Random Field), einer globalen Operation, der Vordergrund auswählt. Das Bild wird dazu in einen Graphen modelliert und versucht die Knoten des Graphen bzw. die Bildpunkte optimal in Vordergrund und Hintergrund aufzuteilen. Die Entscheidungsregel basiert dabei auch der Helligkeitsänderung in Bezug auf die Entfernung der Pixel. Von der Verteilung der Helligkeiten der Vordergrundpixel wird schließlich die Kumulante vierter Ordnung gebildet, ein statistisches Verfahren, das die Abweichung von einer Normalverteilung ausdrückt. Die Kumulante wird dann als Fingerabdruck benutzt.

Weiterhin existiert mit [22] auch eine Signatur, die den Zusammenhang zwischen Ton und Bild ausnutzt und aus einer kombinierten einfachen Videosignatur auf globaler Basis (jedes Frame wird geviertelt und der Schwerpunkt der Grauwerte berechnet) und einer Audiosignatur besteht.

2.3 Geschichte und aktuelle Lage

Die ersten Arbeiten zu Videosignaturen gab es Anfang des Jahrtausends. 8 Jahre später wurde dann von der MPEG beschlossen, ein entsprechendes Verfahren zu spezifizieren und ein Call for Proposals im Juli 2008 gestartet [23]. Dieser wurde im Oktober desselben Jahres überarbeitet und endete am 26. Januar 2009 [6]. Die Einsendungen wurden von der MPEG unter Leitung von Miroslaw Bober, Paul Brasnett und Ryoma Oami evaluiert und in überarbeiteter Form als MPEG-7 Videosignatur im Juli 2012 standardisiert. Unter dem Stichwort „Video Signature“ sind seit der Verabschiedung des Standards keine Veröffentlichungen mehr zu finden. Ansätze mit gleicher Zielsetzung finden sich jedoch u.a. in [10, 12, 21], die 2013 und 2014 veröffentlicht wurden. Sie beruhen allerdings – wie oben dargestellt – auf anderen Techniken als der MPEG-7 Standard.

Es ist bekannt, dass Videosignaturtechnik von YouTube genutzt wird. Sie firmiert dort unter dem Namen ContentID [24]. Es ist nicht bekannt, wie diese Technik im Detail arbeitet. Da in [25] aber „Spiegeln“ als wirksame Umgehungsmaßnahme genannt wird, scheinen die Videodaten analysiert zu werden. Mit Vimeo benutzt eine weitere Videoplattform eine automatisierte Technik um Copyright geschützte Inhalte zu finden. Diese basiert aber alleine auf der Erkennung der Tonspur [26].

Auf technischer Ebene sind Videosignaturen mit Audiofingerprinting-Techniken vergleichbar. Diese haben bereits weite Verbreitung in Musiksuchmaschinen auf Smartphones und Tablets gefunden. Als Beispiele sind SoundHound [27] oder Shazam [28] zu nennen. Auch Musicbrainz (eine Datenbank für Musikmetadaten) benutzt mit AcousticID [3] solch ein Verfahren.

Bei entsprechenden Filmdatenbanken wie der IMDb [29], OFDb [30] oder TMDb [31] finden sich allerdings keine Hinweise auf den Gebrauch einer solchen Technik.

Es fällt zudem auf, dass die Arbeit aus dem MPEG-7 Standard nur noch eingeschränkt zugänglich ist. Das Paper, das den Standard beschreibt [7] ist nach kurzer Recherche zwar auffindbar, da aber einige Techniken und vor allem Schwellwerte dort nicht im Detail erklärt werden, benötigt man zum vollen Verständnis, oder um die Technik neu zu implementieren und anzuwenden, zwangsläufig die Referenzsoftware. Diese ist nicht öffentlich auffindbar und wurde mir dankenswerterweise in großen Teilen von meinem Betreuer zur Verfügung gestellt [32]. Sie ist allerdings ausschließlich auf den Signaturteil beschränkt, sodass sie nicht ohne weiteres kompiliert[8] und somit auch nicht getestet werden kann. Das Testset sowie zugehörige Signaturen sind ebenso nicht mehr vollständig auffindebar, in [33] (das mir ebenso von meinem Betreuer zur Verfügung gestellt wurde) finden sich allerdings mehrere Videosequenzen mit den dazugehörigen Videosignaturen, sodass eine Prüfung auf Korrektheit möglich war.

[8] Es fehlen z. B. diverse Konstanten und Funktionen, die in anderen – nicht mitgelieferten – Dateien definiert sind.

3 Der MPEG-7 Standard

Der von der MPEG verabschiedete Standard „MPEG-7 Video Signature Tools for Content Identification“ ist ein Teil von MPEG-7, einer Klasse von Verfahren zur Bilddatenanalyse. Er ist als ISO/IEC 15938-3:2002/AMD 4:2010 standardisiert [34].

3.1 Geschichte

Um einen Standard zu spezifizieren, wurde von der MPEG im Juli 2008 ein Call for Proposals gestartet und im Januar 2009 beendet. Die Einsendungen wurden anschließend in mehreren Meetings der MPEG evaluiert. Dabei entschied sich die MPEG für das kombinierte Verfahren von NEC und Mitsubishi Electric [35, 36]. Dieses beschreibt die in Abschnitt 3.2.1 erläuterte Finesignature fast vollständig. Sie wurde allerdings aufgrund experimenteller Resultate um 40 Elemente von 340 auf 380 erweitert. Zudem wurden die Bag-of-Words und darauf aufbauend die Coursesignature (Abschnitt 3.2.2) hinzugefügt, die den Lookup maßgeblich beschleunigt [36]. Die Komprimierung wurde ebenfalls in der Methode von [35] nicht beschrieben, sondern wird in [37] detailliert vorgestellt und auch so in den Standard übernommen [7].

3.2 Technik

Der Standard basiert auf einer frame- und blockbasierten Technik, beinhaltet aber auch globale Merkmale. Die komplette Signatur selbst besteht dabei aus zwei Teilen: Aus jedem Frame wird eine *Finesignature* extrahiert und anschließend auf deren Basis eine *Coursesignature* berechnet. Die Finesignature ist dabei den lokalen Signaturen zuzuordnen, die Coursesignature einer globalen Signatur [7]. Dieses kombinierte Verfahren führt im Wesentlichen zu einem schnelleren Lookup.

Abbildung 2 stellt eine Übersicht über die vollständige Videosignatur dar.

3.2.1 Finesignature

Eine Finesignature besteht aus einer *Framesignature*, einem *Confidencewert* und fünf 5-dimensionalen Wörtern. Zuallererst werden dazu pro Frame ausschließlich die Helligkeitswerte betrachtet (8 Bit Genauigkeit). Außerdem wird jedes Frame auf 32×32 Pixel skaliert. Die Skalierung passiert durch Unterteilung des ursprünglichen Bildes in 32×32 Blöcke und anschließender Mittelwertberechnung innerhalb jedes Blocks [38].

Framesignature Die Basis der Finesignature und somit auch der gesamten Videosignatur bildet die Framesignature, welche aus 380 ternären[9] Werten besteht. Diese Werte werden durch Schwellwertbildung aus Blocksummen gewonnen.

Dazu werden zunächst 10 Kategorien gebildet. Die ersten beiden Kategorien bilden dabei die *Average-Blocks* (A1 und A2), die 8 weiteren Kategorien die *Difference-Blocks* (D1 bis D8). Die Average-Blocks enthalten insgesamt 32 Blöcke, die Difference-Blocks 348 Blöcke:

[9] ternäre Werte besitzen 3 Zustände, ähnlich zu den binären Werten mit 2 Zuständen oder dezimalen Werten mit 10 Zuständen

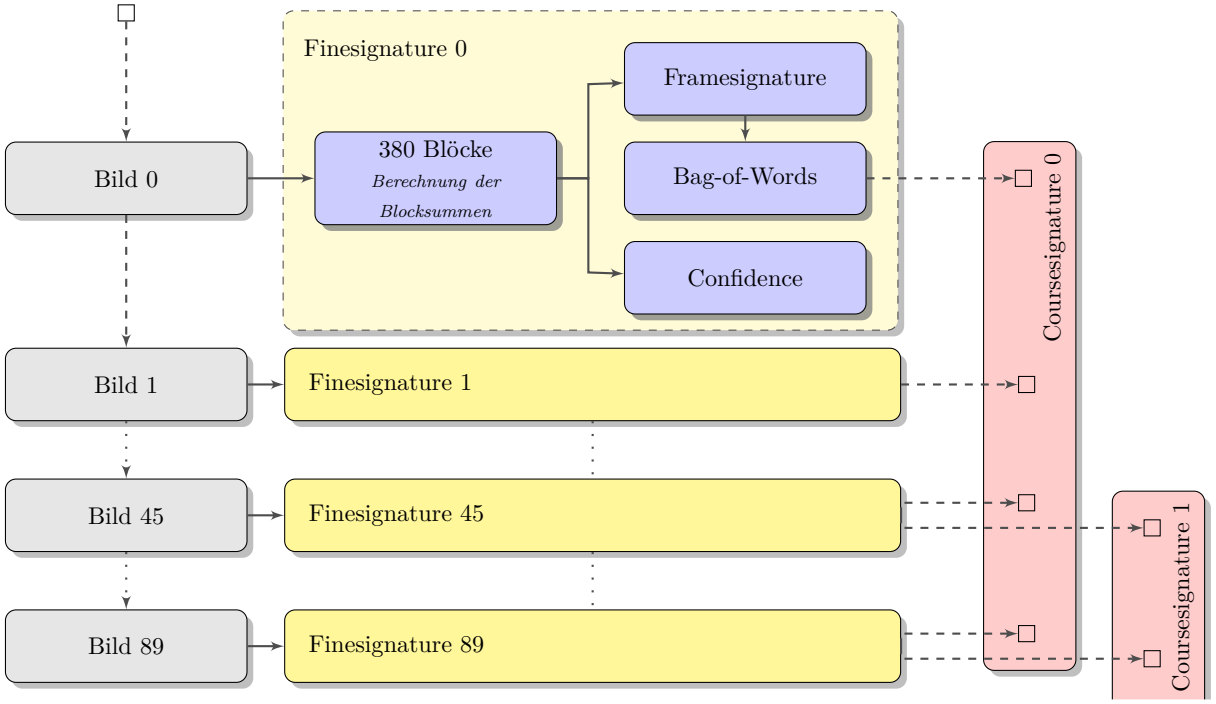


Abbildung 2: Aufbau der vollständigen MPEG-7 Videosignatur

- Ein Average-Blockelement besteht aus einem Block innerhalb der 32×32 Pixel, über dessen Fläche der Mittelwert gebildet wird.
- Ein Difference-Blockelement besteht aus zwei Blöcken, über deren Flächen der Mittelwert gebildet wird. Anschließend wird der zweite Mittelwert vom ersten subtrahiert.

Die Blöcke müssen dabei nicht unbedingt rechteckig sein. Jede Kategorie zeichnet sich aber durch dieselbe Art von Blockmuster, zumeist auch durch die gleiche Blockgröße, aus. Einige Beispiele jeder Kategorie sind in Abbildung 3 zu sehen.

Der Mittelwert eines Block liegt, da 8-Bit-Helligkeitswerte gemittelt werden, zwischen 0 und 255. Der Wert eines Difference-Blockelements liegt somit zwischen $0-255=-255$ und $255-0=255$. Die Werte der Average-Blöcke, denen das Difference-Element fehlt, werden durch die Subtraktion von 128 auf diesen Wertebereich angeglichen und liegen somit zwischen $0-128=-128$ und $255-128=127$.

Anschließend werden die Werte ternarisiert, also in 0, 1 oder 2 codiert[10]. Eine Zuteilung erfolgt dabei anhand des Schwellwertes th [7]:

$$t(x)=\begin{cases}0 & x<-th\\ 1 & -th\leq x<th\\ 2 & x\geq th\end{cases}$$

[10] In [39] findet man auch -1, 0, 1 als Zahlenwerte, die die Zugehörigkeit zu einem Wertebereich um 0 herum besser symbolisieren, die Darstellung als 0, 1 und 2 entspricht aber der Implementierung und wird auch in [7] verwendet.

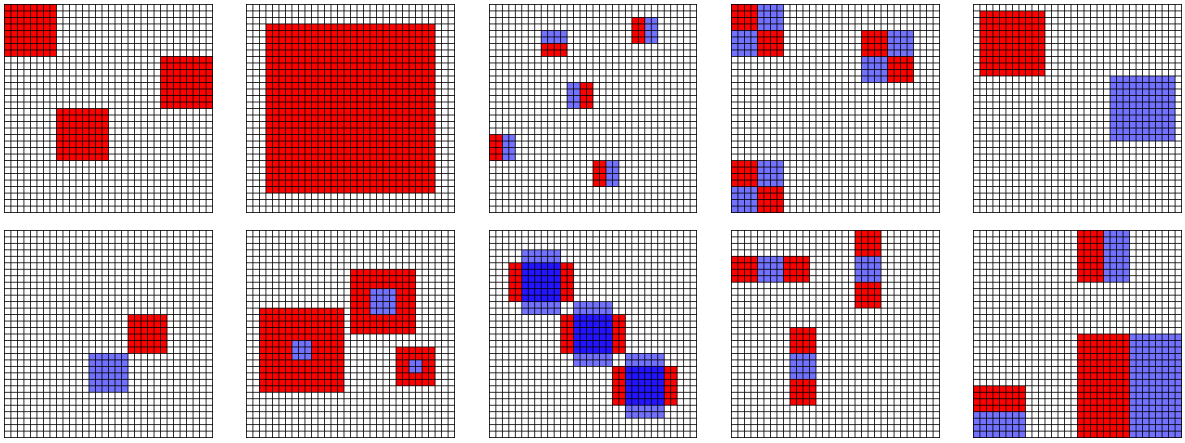


Abbildung 3: Grafische Darstellung der Blöcke von allen Kategorien in der Reihenfolge A1, A2, D1, D2, . . . , D8. Es sind jeweils mehrere Elemente einer Kategorie (außer von A2, D3 und D4) zu sehen. Die blauen Flächen werden von den roten Flächen abgezogen. Man beachte die Überlappung in Kategorie D6.

Der Schwellwert wird dabei für jede Kategorie neu bestimmt. Dazu wird von allen Elementen der Kategorie der Betrag gebildet, die so entstehende Liste sortiert und anschließend der Wert des Elements gewählt, das nach $\frac{1}{3}$ der Anzahl der Listenelemente[11] auftritt.

Alle 380 ternarisierten Werte im Gesamten bilden schließlich die *Framesignature* [7].

Confidencewert Der Confidencewert gibt an, wieviel Information ein Frame zur Signaturbestimmung beisteuern kann [7]. Er wird gebildet, indem man den Median m über alle nicht ternarisierten Absolutwerte der Difference-Blocks bestimmt und anschließender Berechnung von:

$$\min(\lfloor m \cdot 8 \rfloor, 255)$$

Durch die Beschränkung auf 0 bis 255 sind 8 Bit zur Speicherung ausreichend. Starke Helligkeitsschwankungen innerhalb des Frames resultieren in einem hohen Median, schwache Schwankungen in einem niedrigen. Die Multiplikation mit 8 erhöht die Feinheit der Messung. Ein hoher Confidencewert steht also für starke Helligkeitsschwankungen bzw. kontrastreiche Bilder [7].

Bag-of-Words Die fünf 5-dimensionalen Wörter werden auch Bag-of-Words („Sack/Ansammlung von Wörtern“) genannt. Sie werden aus der Framesignature extrahiert. Dazu werden an folgenden Positionen die ternären Werte entnommen [32]:

$$\begin{array}{lll} \{ & 210, 217, 219, 274, 334, & 44, 175, 233, 270, 273, \quad 57, 70, 103, 237, 269, \\ & 100, 285, 295, 337, 354, & 101, 102, 111, 275, 296 \quad \} \end{array}$$

Jeweils fünf der extrahierten Ternärwerte entsprechen einem Wort.

[11] Nach Referenzcode eigentlich: Wenn n die Anzahl der Elemente ist, das Element, das an Position $\lfloor n \cdot 0{,}33333 \rfloor$ in der Liste auftritt [32]. Das Paper benutzt die Werte 0,333 und 0,3333 [7].

3.2.2 Coursesignature

Hat man die Finesignature aus jedem Frame extrahiert, kann man daraus die Menge der *Coursesignatures* berechnen. Die Coursesignature ist somit komplett redundant und dient hauptsächlich dazu den Lookup durch eine vorab auswählende Suche deutlich zu beschleunigen.

Bedingt durch die Redundanz ist es prinzipiell nicht zwangsläufig notwendig, die Coursesignatures mit in der Signatur zu übertragen. Sie kann genauso gut erst im Lookup berechnet werden. Der Standard schreibt aber die Berechnung und Übertragung vor[12] [7].

Es gibt zu jeder Zeit (außer den ersten 45 Frames) immer zwei Coursesignatures. Sie werden jeweils über 90 Frames berechnet und laufen um 45 Frames versetzt. Die erste Coursesignature läuft also von Frame 0-89, die zweite von Frame 45-134, die dritte von Frame 90-179 usw..

Um die Coursesignature zu berechnen, werden fünf binäre Histogramme auf Basis der Wörter gebildet. Dazu wird aus jeder Finesignature der 90 Frames das erste 5-dimensionale Wort genommen und als Zahl zur Basis drei interpretiert. Wenn das Wort also $(w_0, w_1, w_2, w_3, w_4)$ ist, dann lautet die resultierende Zahl r [32]:

$$r = w_0 \cdot 3^4 + w_1 \cdot 3^3 + w_2 \cdot 3^2 + w_3 \cdot 3^1 + w_4 \cdot 3^0$$

r kann dabei zwischen 0 und $3^5 - 1 = 242$ liegen.

Die sich so ergebenden 90 Zahlen werden in ein Histogramm einsortiert, welches danach mit dieser Entscheidungsregel binarisiert wird:

$$t(x) = \begin{cases} 0 & x = 0 \\ 1 & \text{sonst} \end{cases}$$

Anschließend wiederholt man den Prozess für die zweiten (dritten, ...) Wörter. So entstehen fünf binäre Histogramme, die aneinandergereiht die Coursesignature [7] bilden.

3.2.3 Komprimierung

Um Speicher bzw. Übertragungsrate einzusparen, beschreibt der Standard eine verlustlose Komprimierung, die die Ähnlichkeiten der aufeinanderfolgenden Finesignatures beachtet und somit einer Interframekodierung gleicht.

Die Komprimierung wird auf Gruppen von 45 folgenden Frames angewandt, die auch jeweils die Startpunkte einer neuen Coursesignature darstellen. Sie betrifft ausschließlich die Framesignature [7].

Für die Komprimierung wird zwischen *Key Pictures* (KP) und *Predicted Pictures* (PP)[13] unterschieden. KPs werden ohne Komprimierung gespeichert. Die erste der 45 Framesignatures ist immer ein KP. Eine Gruppe von Bildern zwischen zwei KPs (inklusive des ersten, exklusive des zweiten) wird als *Group of Pictures* (GOP) bezeichnet.

[12] Die gleichen Überlegungen gelten übrigens auch schon für die 5 Wörter.
[13] vgl. I-Frames und P-Frames

Außerhalb der festgelegten KPs wird eine Framesignature als KP codiert, wenn die Anzahl der gleichen ternären Werte zur vorhergehenden Framesignature eine gewisse Schwelle unterschreitet[14].

Die Komprimierung zwischen einem KP und PP bzw. PP und PP erfolgt dann durch eine Modulo-3 Substraktion:

$$\widetilde{x}_{i,m} = ((x_{i,m-1} - x_{i,m}) \mod 3)$$

i ist dabei der Index des Ternärwertes innerhalb der Framesignature und m der Index der Framesignature.

Der Zweck dieser Substraktion ist es, den Ternärwert mit dem vorherigen Frame zu prädizieren und möglichst eine Null zu erzeugen. Dadurch wird die Entropie reduziert und die Nullen können mit einer Lauflängencodierung effizient komprimiert werden. Die Modulo-3 Operation normiert auch im Negativen auf 0 bis 2, so gilt z. B.:

$$1 = (0 - 2) \mod 3)$$

Die Rücktransformation erfolgt dann per:

$$x_{i,m} = ((x_{i,m-1} - \widetilde{x}_{i,m}) \mod 3)$$

Die so prädizierten KPs kann man als Matrix, wie in Abbildung 4 gezeigt, darstellen.

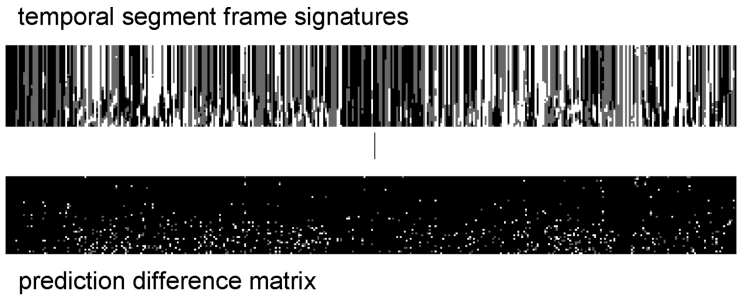


Abbildung 4: Grafische Darstellung der GOP und der prädizierten GOP. Von links nach rechts sind die 380 ternären Elemente aufgetragen, von oben nach unten die Framesignatures der GOP [7].

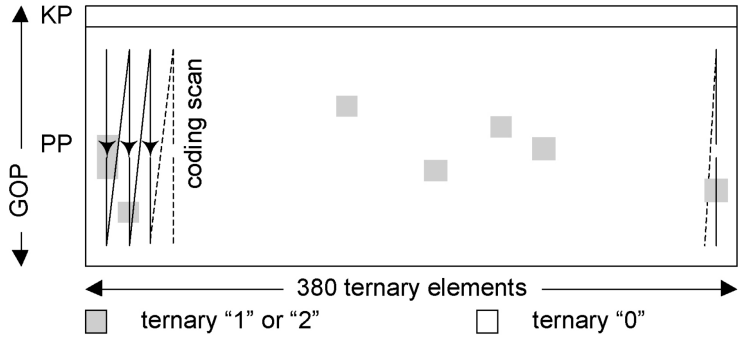


Abbildung 5: Grafische Darstellung des 1D-Scanning Verfahrens der GOP [7].

[14] Die Anzahl der gleichen ternären Werte kann man als Maß für die Korrelation zwischen den Frames betrachten. Somit wird ein neuer KP erzeugt, wenn die Korrelation zum vorhergehenden Frame nicht mehr groß genug ist.

Es fällt auf, dass die ternären Elemente über die gesamte GOP ähnlich oder sogar gleich bleiben (an den schwarzen Flächen der prädizierten GOP gut zu erkennen, auf Zahlenebene zeigt sich dies an langen Folgen von Nullen). Diese Beobachtung wird für die weitere Codierung ausgenutzt, indem die Matrix spaltenweise in einen 1D-Vektor gescannt wird (siehe dazu auch Abbildung 5).

Durch die Modulo-3 Operation kommen nur 0, 1 und 2 als Werte in diesem Vektor vor, die verschieden codiert werden. Eine 1 wird mit einem „0“-Bit codiert, eine 2 mit einem „1“-Bit. Nach jeder 1 oder 2 wird dann ein Lauflängencodewort für die Anzahl der nachfolgenden Nullen eingefügt (falls keine Null nach der 1 oder 2 existiert, wird ein Codewort für die Länge „Null“ eingefügt[15]).

Die Zuordnung des Codewortes zu der Lauflänge entspricht einer Entropiecodierung. In [37] wurden basierend auf Videosequenzen des Internet Archives Statistiken der vorkommenden Lauflängen erstellt (Abb. 6). Dort ist eine exponentielle Verteilung nach folgendem Muster zu sehen:

$$p_\omega \sim \beta\omega^{-\alpha}$$

ω entspricht der Lauflänge, p_ω der Auftrittswahrscheinlichkeit. β und α sind Parameter mit $\beta > 0$ und $\alpha > 0$. In Abbildung 6 erkennt man weiterhin ein sehr langsam auslaufendes Ende, anders gesagt nimmt die Wahrscheinlich für große Lauflängen immer weniger ab.

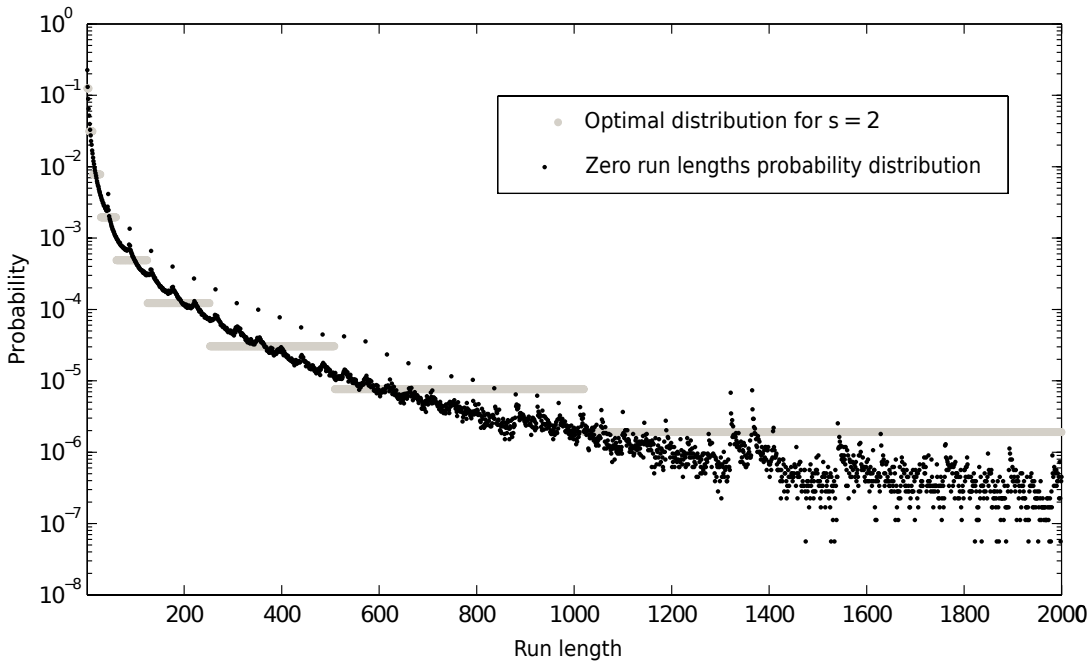


Abbildung 6: Wahrscheinlichkeit verschiedener Lauflängen [37]. In grau ist die optimale Verteilung für eine Exp-Golomb-Code mit Parameter 2 aufgetragen, in schwarz die Wahrscheinlichkeitsverteilung der gemessenen Lauflängen.

Für diese Art der Verteilung eignet sich der Exponential-Golomb-Code (auch Exp-Golomb), der sich zudem durch eine geringe Komplexität auszeichnet. Dieser Code ist parametrisiert über

[15] Das hier verwendete Codewort für die Länge „Null“ ist *0*, resultierend aus dem weiter unten beschriebenen Exp-Golomb-Code.

einen Parameter k und ideal für eine bestimmte Wahrscheinlichkeitsverteilung der zu codierenden Zahlen. Die entsprechende Verteilung für $k = 2$ ist ebenfalls in Abbildung 6 zu sehen. Die Codewortlänge eines Exp-Golumb-Code errechnet sich zu:

$$l_\omega = 1 + 2\lfloor \log_2(\omega + 2^k) \rfloor - k$$

Der Code setzt sich aus einem Präfix, gefolgt von einem Trennzeichen, gefolgt von einer Binärzahl zusammen [40, 41]. Das hier vorgestellte Verfahren benutzt Einsen als Präfix und eine Null als Trennzeichen[16] [32]. Die Länge des Präfixes zusammen mit der dann folgenden Binärzahl macht den Code eindeutig identifizierbar [42].

Zum Dekodieren wird aus dem Exp-Golumb-Code die Lauflänge ermittelt und in ternäre Nullen dekodiert. Anschließend werden die Bits, die nicht zum Exp-Golumb-Code gehören in ternäre Einsen und Zweien übersetzt, die Matrix wiederhergestellt und dann mit der oben erwähnten Formel die Prädiktion rückgängig gemacht [7].

Wie in MPEG-2 und MPEG-4 AVC ist nur der Decoder spezifiziert. Der Encoder kann beliebig umgestaltet werden, solange er einen Bitstrom liefert, den der Decoder lesen kann [7].

3.3 Bitstrom

Um einen platzsparenden Bitstrom zu schaffen, muss die Umwandlung von Ternärwerten (in denen sowohl die Framesignature als auch die Wörter gespeichert sind) in Binärwerte effizient erfolgen.

Die Confidence kann unverändert übernommen werden. Das gleiche gilt für die bereits binarisierten Coursesignatures.

Ternärwerte werden in Binärwerte codiert, indem jeweils fünf ternäre Werte als Zahl zur Basis drei interpretiert (vergleiche die Coursesignature-Bildung) und dann in einer 8-Bit Zahl gespeichert werden. Ein Überlauf ist dabei nicht möglich, da gilt:

$$3^5 - 1 = 242 < 2^8 - 1 = 255$$

Der genaue Aufbau des Bitstroms ist in [38] beschrieben. Hier soll nur die grundlegende Speicherung erläutert werden. In einen Bitstrom können mehrere Signaturen eingebettet werden, dazu unterteilt er sich in mehrere Regionen. Jede Region besitzt eine Größe und einen Startpixel und definiert einen Teilausschnitt des Bildes. So kann nur auf einem Teil des Bildes die Videosignatur gebildet werden, was für die Bild-in-Bild-Erkennung wichtig ist [7]. Jede Region beinhaltet dann, neben diversen Headerdaten wie Anzahl der Frames, die Signatur aus Coursesignatures und Finesignatures.

Der generelle Aufbau der unkomprimierten Finesignature ist in Abb. 7 ersichtlich. Für die Framesignature werden $\frac{380}{5} = 76$ Byte benötigt. Die Finesignatures werden dann in einer Kette hintereinander gehängt. Falls die Signatur in komprimierter Form vorliegt, werden die Framesignatures weggelassen und hinter die Kette von Finesignatures in komprimierter Form gehängt.

[16] Die Zuordnung von Eins und Null muss nicht so sein. Insbesondere der MPEG-4 AVC/H264-Standard, der die bekannteste Anwendung für Exp-Golomb-Codes darstellt, benutzt als Präfix Nullen und als Trennzeichen Einsen [40, 41].

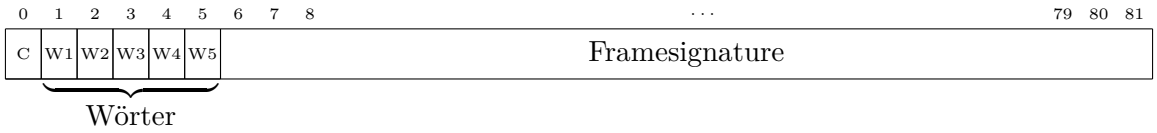


Abbildung 7: Binärer Aufbau der Finesignature, C: Confidence, Angaben in Byte

Die Coursesignatures werden vor die Finesignatures gehängt[17], sodass sich für die unkomprimierte Form der Bitstrom in Abb. 8 ergibt. Die ebenfalls abgespeicherten Größen von n und m wurden in der Abbildung nicht berücksichtigt. Außerdem werden bei jeder Coursesignature noch Nummer und optional Zeit des Start- und Endframes hinterlegt.

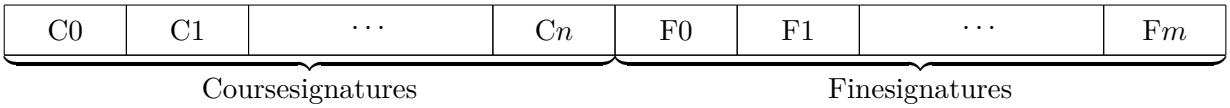


Abbildung 8: Videosignatur in unkomprimierter Form

In der komprimierten Form ergibt sich der Aufbau in Abb. 9. Der Finesignatureteil besteht hier nur noch aus Confidence und Bag-of-Words. CS bezeichnet ein *Compressed Segment*, das immer 45 Frames beinhaltet. Durch die KP-Verteilung bei der Kompression ergibt sich so mindestens eine GOP pro CS. Man beachte außerdem die doppelte Verwendung von n, die daraus resultiert, dass ein CS zusammen mit einer neuen Coursesignature beginnt und so von beiden dieselbe Anzahl existiert.

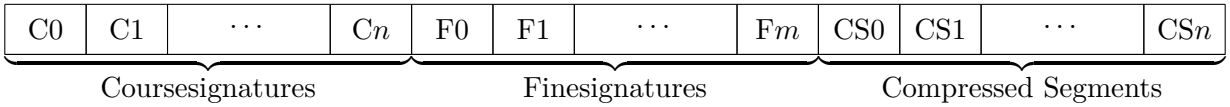


Abbildung 9: Videosignatur in komprimierter Form

3.4 Speicherplatzbetrachtung

Ein Kriterium der MPEG war die Kompaktheit der Signatur, die die Länge des Bitstroms bezeichnet. Diese unterscheidet sich in der unkomprimierten und komprimierten Form.

3.4.1 Unkomprimierte Speicherung

Der Speicher für die unkomprimierte Videosignature ergibt sich aus der Summe der Fine- und Coursesignatures.

Mit dem oben beschriebenen Bitstromformat ergibt sich der Speicherbedarf für eine Finesignature von 8 Bit für den Confidencewert, 5 mal 8 Bit für die Wörter und $\frac{380}{5}$ mal 8 Bit für die Framesignature. Da pro Frame eine Finesignature gebildet wird erhält man [7]:

$$8 + 5 \cdot 8 + \frac{380}{5} \cdot 8 = 82 \cdot 8 = 656\,\text{Bit/Frame}$$

[17] Dieses Verhalten verhindert im Übrigen eine Ausgabe während der Berechnung.

Für eine Coursesignature wird ein Speicher von 5 mal 243 Bit benötigt, dies resultiert auf Framebasis in:

$$\frac{5 \cdot 243}{90} = 13{,}5\,\text{Bit/Frame}$$

Da immer zwei Coursesignature gleichzeitig auftreten, ergibt sich insgesamt für die Videosignature:

$$13{,}5 \cdot 2 + 656 = 683\,\text{Bit/Frame}$$

Bei einer Bildrate von 30 Bildern/s führt das zu:

$$683\,\text{Bit/Frame} \cdot 30\,\text{Frame/s} = 20490\,\text{Bit/s} \approx 20\,\text{kBit/s}$$

3.4.2 Komprimierte Speicherung

Bei Anwendung der Komprimierung auf die Videosignatur erhält man eine mittlere Kompressionsrate der Framesignature auf etwa 23 % der Originalrate. Das resultiert in einer mittleren Kompressionsrate auf etwa 27 % für die komplette Signatur. In Zahlen ausgedrückt sind das im Mittel 184 Bit/Frame, was bei 30 Bildern/s eine Bitrate von 5532 bit/s ergibt [7].

Experimente haben weiterhin ergeben, dass die beschriebene Komprimierung bessere Resultate liefert als generische Komprimierungsalgorithmen. Getestet wurden „bzip2“, „ppm+arithmetic coding“ und „LZMA“ [7, 37].

3.5 Lookup

Der Lookup gehört nicht offiziell zum Standard. Nichtsdestotrotz musste die MPEG ihre Ergebnisse evaluieren und hat dazu einen Lookupmechanismus implementiert und auch beschrieben. Dieser wird hier übernommen.

Der Lookup besteht dabei aus 3 Schritten, die zusammen eine hierarchische Suche darstellen und so eine deutliche Geschwindigkeitssteigerung beim Vergleichen erzielen. Es werden zuerst in einem Schritt einige wenige Kandidaten auf Basis der Coursesignature ausgewählt und anschließend in zwei Schritten mit der Finesignature detailliert auf Übereinstimmung geprüft [7].

Außerdem ist die Komprimierung so gestaltet, dass der erste Schritt ohne Dekomprimierung stattfinden kann (da er nur die nicht komprimierte Coursesignature nutzt) und erst bei der detaillierten Überprüfung eine Dekomprimierung erfolgen muss.

3.5.1 Schritt 1: Coursesignature-Lookup

In diesem ersten Schritt, wird die gesamte Menge der Paare aller Coursesignatures beider Videos miteinander verglichen. Der Vergleich basiert dabei auf dem Jaccard-Index [7, 43]:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Es wird für jedes der fünf Histogrammpaare ein Index berechnet. Die Schnittmenge zweier Histogramme ist dabei gleichgesetzt mit einer Veroderung, die Vereinigungsmenge als Verundung. Seien beispielhaft die beiden Histogramme gegeben:

$$H_1^1 = \{1,1,1,0,0,1,0,0\} \qquad H_1^2 = \{0,1,1,1,0,1,1,0\}$$

Dann entspräche der Jaccard-Index:

$$J\left(H_1^1, H_1^2\right) = \frac{3}{6} = \frac{1}{2}$$

Nach der Berechnung wird jeder der fünf Jaccard-Indizes mit einem Schwellwert Thd verglichen. Falls die Hälfte der Indizes größer als der Schwellwert ist, wird das Coursesignature-Paar verworfen. Abschließend wird die Summe der Indizes mit einem weiteren Schwellwert Thd_C verglichen [7]:

$$\sum_i J\left(H_i^1, H_i^2\right) < Thd_C$$

Ist dieser größer, wird das Paar ebenfalls verworfen. Alle gültigen Paare durchlaufen dann den nächsten Schritt.

3.5.2 Schritt 2: Berechnung von Framerate und Offset

Im nächsten Schritt werden die jeweiligen 90 zugehörigen Finesignatures des Coursesignature-Paars genommen und ebenfalls paarweise geprüft. Sei das Coursesignature-Paar c_1 und c_2, dann sind die entsprechenden Finesignatures $f_{c_1}^0$-$f_{c_1}^{89}$ und $f_{c_2}^0$-$f_{c_2}^{89}$.

Für jedes Paar $f_{c_1}^i$ und $f_{c_2}^j$ wird nun die L_1-Distanzbildung vorgenommen, also der Abstand der 380 ternären Werte aufaddiert (fr_f^i bezeichnet die Framesignature von der Finesignature f an der Position i:

$$\sum_{k=0}^{380} \left| fr_{f_{c_1}^i}^k - fr_{f_{c_2}^j}^k \right|$$

Wenn dieser einen Schwellwert Thx_H überschreitet, wird das Paar verworfen [7].

Anschließend werden jeweils zwei der Paare, deren L_1-Distanz kleiner als Thx_H ist, dazu genutzt mithilfe einer linearen Regression mögliche Parameter für das Frameratеverhältnis r und Offset o zu erhalten. Das Frameratеverhältnis gibt dabei das Verhältnis der beiden Videosequenzen an, der Offset die zeitliche Verschiebung [32]:

$$r = \frac{\text{Framerate(zu bestimmende Sequenz)}}{\text{Framerate(Originalsequenz)}}$$

$$o = \text{Zeitstempel(zu bestimmende Sequenz)} - \text{Zeitstempel(Originalsequenz)}$$

Seien die beiden Paare $f_{c_1}^{i_1}$,$f_{c_2}^{j_1}$ und $f_{c_1}^{i_2}$,$f_{c_2}^{j_2}$. j kann als lineare Funktion von i interpretiert werden:

$$j = g(i) \qquad \text{mit } g(i) = r \cdot i + o$$

r und o berechnen sich dann zu:

$$r = \frac{j_2 - j_1}{i_2 - i_1} \qquad o = j_1 - r \cdot i_1 = \frac{j_2 - j_1}{i_2 - i_1} \cdot i_1$$

Diese Relation kann als eine $\mathbb{N},\mathbb{N} \to \mathbb{Q},\mathbb{Q}$-Funktion aufgefasst werden:

$$(r, o) = R_{\text{lin}}(f_1, f_2)$$

Mit dieser Funktion kann über die Gesamtheit der Paare eine Hough-Transformation durchgeführt werden:

$$(f_1, f_2) = R_{\text{hough}}(r, o)$$

Der am dichtesten besetzte Punkt im Hough-Raum (also die Framerateverhältnis- und Offsetwerte, die am häufigsten auftreten) wird anschließend als temporärer Parameter gewählt und zusammen mit dem Framesignature-Paar mit der geringsten Distanz in diesem Punkt an Schritt 3 weitergereicht[18] [32].

3.5.3 Schritt 3: Evaluierung

Im letzten Schritt wird, ausgehend von dem in Schritt 2 berechneten besten Paar, in beide zeitlichen Richtungen evaluiert [7].

Dazu werden, ausgehend von dem Framerateverhältnis r, die nächsten beiden und die vorherigen beiden Frames berechnet. Sei dazu die Originalsequenz S_o und die zu bestimmende Sequenz S_b und das beste Paar dann $f_{S_o}i$ und $f^j_{S_b}$. Ausgehend von r werden dann die folgenden Frame-Paare ausgewählt [32]:

$$\begin{array}{lllll} \text{für } r \geq 1: & f^{i+k}_{S_o} & \text{und} & f^{j+\lfloor r \cdot k \rceil}_{S_b} \\ \text{für } r < 1: & f^{j+\lfloor \frac{1}{r} \cdot k \rceil}_{S_o} & \text{und} & f^{i+k}_{S_b} \end{array}$$

Von diesen Paaren wird dann wieder die L_1-Distanz der Framesignature berechnet. Falls die L_1-Distanz einen Schwellwert Thx_H übersteigt, werden die Confidencewerte der Finesignature betrachtet. Wenn der Frame zu „unwichtig“ ist, d.h. wenn die Confidencewerte der Frames unter einem Schwellwert Thc_2 liegen, wird das Paar ignoriert und weitergemacht. Erst wenn 3 Paare hintereinander eine zu niedrige L_1-Distanz aufweisen und einen Confidencewert haben der über Thc_2 liegt, wird die Suche beendet. Falls die zeitliche Menge der so gefundenen Paare eine gewisse Zeit überschreitet, wird die Evaluierung als erfolgreich beendet und die beiden Sequenzen als gleich angesehen, ansonsten werden die Sequenzen verworfen. Im Falle eines Erfolgs wird außerdem das größte Intervall zurückgegeben, in dem die L_1-Distanzen unter Thx_H und die Confidence über Thc_2[19] liegt [7].

Dieses Verhalten führt dazu, dass auch ein teilweises Matching[20] möglich ist. Dies bedeutet das Auffinden von einzelnen gleichen Szenen in ansonsten verschiedenen Sequenzen. Das Gegenstück dazu ist das direkte Matching, das überprüft, ob die komplette Videosequenz identisch ist.

[18] In der Implementierung ist dabei der eigentlich unendliche Hough-Raum auf Framerates von 0 bis 60 beschränkt (d.h. ein Framerateverhältnis von 0 bis 2 bei einer Originalframerate von 30fps) und Offsets von -90 bis 90 [32].

[19] Die Werte der Schwellwerte sind ausschließlich in der Referenzsoftware definiert und liegen bei: $Thd = 9000$, $Thd_C = 60000$, $Thx_H = 116$ und $Thc_2 = 1$ [32]

[20] als übereinstimmend erkennen

3.6 Core-Experiments

Die MPEG hat das vorgestellte Videosignaturverfahren in Core-Experiments ausgiebig getestet. Dabei wurden zwei Testläufe durchgeführt. Die *Unabhängigkeitstests* und die *Tests auf Robustheit* [7].

Für die Tests wurden aus einer Videosequenz (Originalsequenz) Teile von 2, 5 und 10 Sekunden entnommen (Suchsequenzen), die in einem ersten Schritt direkt gegen die Originalsequenz getestet und in einem zweiten Schritt mit weiterem Videomaterial davor und danach versehen und dann gegen die Originalsequenz getestet wurden (Abb. 10). Dadurch wurden immer 6 *Query-Clips* gegen eine Originalsequenz getestet.

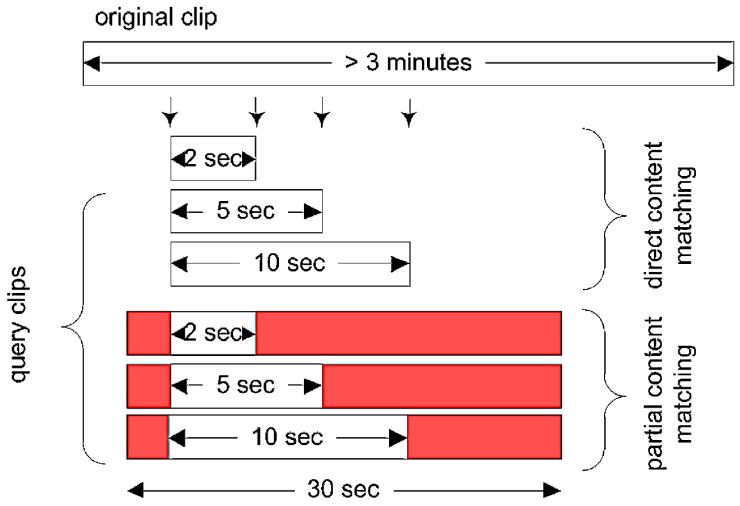


Abbildung 10: Darstellung des Testverfahrens für die Core-Experiments [7]

3.6.1 Unabhängigkeitstest

Die verwendeten Testvideos für den Unabhängigkeitstest bestanden aus 1900 Videosequenzen verschiedener Typen wie z. B. Film, Dokumentation, News, Zeichentrick, Sport, Amateur Video usw. mit einer Länge von jeweils 3 Minuten. Für die Tests wurden diese in jeweils sechs 30 Sekunden Stücke geteilt, aus denen dann wiederum die sechs Query-Clips erstellt wurden. So ergaben sich etwa 70000 Query-Clips. Jeder der Query-Clips wurde dann gegen jede der 1900 Videosequenzen getestet, was etwa 120 Mio. Tests ergab, die zur Bestimmung der Ausfallrate[21] verwendet wurden [7].

3.6.2 Tests auf Robustheit

Für die Tests auf Robustheit wurden 545 Videosequenzen verschiedenen Typs mit 3 Minuten Länge verwendet. Aus diesen wurden wiederum die sechs Query-Clips erstellt. Jeder davon wurde dann einer der Kategorien in Tabelle 1 zugeteilt, entsprechend modifiziert und gegen die Originalsequenz getestet.

[21] auch „false positive rate“, siehe Abschnitt 4.5.3 für Details

Name	Variationen S	Variationen M	Variationen H[22]	Testverfahren
TL	10 %	20 %	30 %	Überlagerung mit Text/Logo
KOMP	512	256	64	Komprimierung mit AVC mit n kBit/s bei der Auflösung CIF
RA	CIF[23]	QCIF[24]		Reduktion der Auflösung ausgehend von SD
RB	15	5	4	Bildratenreduktion auf n Bilder pro Sekunde (jeweils von 25 und 30 fps)
KAM	0 %	5 %	10 %	Aufnahme des Bilds mit einer Kamera mit SD Auflösung mit Rand von n %
VCR	1	2	3	n-maliges analoges Aufnehmen per Videokassette und anschließende digitale Abtastung
GRAU				Konvertierung von RGB in Grauwerte mit $I = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B$ (entspricht dem ITU-R BT 601 Standard der RGB-YCbCr Farbraumkonvertierung)
BR	+9	-18	+36	Helligkeitsänderung um n (jeweils auf die Werte 0-255 begrenzt)
IP				Interlaced/Progressive-Konvertierung (progressives Bildmaterial wird in zeilenversetztes Bildmaterial umgewandelt und anschließend mit einem Deinterlacer – BoB – wieder zurück in progressives Bildmaterial umgewandelt)

Tabelle 1: Auflistung der Modifikationen für die Tests auf Robustheit [6, 7]

3.6.3 Evaluierungskriterien

Die Bedingungen, damit ein gematchtes Video auch als richtig gematcht galt, waren wie folgt definiert:

Direktes Matching Sei der originale Startzeitpunkt der Testsequenz in der Originalsequenz OSP. Sei weiterhin der durch die Signatur ermittelte Startzeitpunkt SSP, dann muss gelten [6]:

$$|OSP - SSP| < 1s$$

Teilweises Matching Sei der originale Startzeitpunkt der Matchingsequenz in der Originalsequenz OSP_o, der entsprechende Endzeitpunkt OEP_o. Sei weiterhin der durch die Signatur ermittelte Startzeitpunkt SSP_o und der entsprechende Endzeitpunkt SEP_t. Seien analog die Start- und Endzeitpunkte der Matchingsequenz in der Testsequenz als OSP_t, OEP_t, SSP_t und SEP_t definiert, dann muss gelten [6]:

$$\begin{aligned} |(SEP_t - SSP_t) - (OEP_o - OSP_o)| &< 2s \\ |(OSP_o - SSP_o)| &< 1s \\ |(OSP_t - SSP_t)| &< 1s \end{aligned}$$

[22] **S**chwach, **M**ittel, **H**och, gibt den Grad der Modifizierung an
[23] Common Intermediate Format, entspricht 352×288 Pixel
[24] Quarter CIF, entspricht 176×144 Pixel

Test	Direktes Matching								
	$D = 2s$			$D = 5s$			$D = 10s$		
	H	M	S	H	M	S	H	M	S
TL	79,63	89,54	98,90	87,16	93,03	99,45	88,62	93,21	100,00
KOMP	99,27	99,63	99,63	99,82	100,00	99,82	99,53	100,00	100,00
RA	-	99,27	99,82	-	99,63	99,82	-	99,82	100,00
RB	99,63	99,27	99,63	99,63	99,82	99,82	100,00	100,00	100,00
KAM	79,27	92,27	92,29	83,67	93,55	95,05	91,01	95,21	95,60
VCR	94,42	94,81	95,78	96,45	96,85	96,88	96,59	97,04	97,24
GRAU	-	-	99,82	-	-	100,00	-	-	99,82
BR	100,00	98,53	99,63	100,00	99,45	99,82	100,00	100,00	99,82
IP	-	-	99,82	-	-	100,00	-	-	99,63

Test	Teilweises Matching								
	$D = 2s$			$D = 5s$			$D = 10s$		
	H	M	S	H	M	S	H	M	S
TL	66,97	71,93	78,35	65,50	75,05	78,35	65,50	73,76	97,95
KOMP	96,88	99,44	98,53	97,30	99,08	98,53	97,98	99,03	98,90
RA	-	98,90	98,90	-	98,90	98,90	-	94,50	98,53
RB	81,18	86,11	96,34	81,89	86,51	99,44	93,73	94,62	99,72
KAM	49,17	90,61	91,01	77,25	92,82	93,94	86,42	93,55	94,50
VCR	93,63	94,07	95,96	95,68	96,11	96,70	96,25	96,48	96,69
GRAU	-	-	99,08	-	-	99,27	-	-	99,63
BR	98,72	98,35	98,90	98,72	98,72	98,72	98,72	98,72	99,45
IP	-	-	98,90	-	-	99,27	-	-	99,27

Tabelle 2: Ergebnisse der Tests auf Robustheit. Die Zahlen entsprechen der Sensitivität in % [7].

3.6.4 Ergebnisse

Für die Unabhängigkeitstest wurde eine Ausfallrate von < 5PPM ermittelt. Die Ergebnisse der Tests auf Robustheit sind in Tabelle 2 dargestellt. Daraus ergibt sich eine Erfolgsrate von 95,49 %. Man erkennt weiterhin, dass direktes Matching höhere Erfolgsquoten aufweist und somit ein leichteres Problem ist als teilweises Matching.

Schwierigkeiten scheint der Algorithmus bei Überlagerung mit Text zu haben, dort liegen die Erkennungsraten gerade beim teilweisen Matching unter 80 %. Bei längeren Matchingsequenzen und nicht allzu starker Überlagerung steigen die Raten aber wieder auf über 95 %.

Außerdem ergeben sich bei Erkennungsprobleme, wenn eine Sequenz erneut abgefilmt wird. Bei einer Matchinglänge von 2 Sekunden, teilweisem Matching und einem breiten Rand von 10 % des Bildes liegt die Rate bei inakzeptablen 49 %. Sobald allerdings direktes Matching, weniger Rand oder längere Sequenzen verwendet werden, steigen die Raten rapide auf über 90 %.

Es muss außerdem gesagt werden, dass die Erfolgsrate generell bei längeren Sequenzen zunimmt und in realen Bedingungen für gewöhnlich mehr als zwei oder fünf Sekunden zum Vergleichen vorliegen, sodass man von höheren Erfolgsraten ausgehen kann [7].

4 Implementierung in FFmpeg

Ein Ziel dieser Arbeit war es, den Standard als Filter in FFmpeg zu implementieren. Hier sollen die Anbindung an das Framework und einige Implementierungsdetails vorgestellt werden.

FFmpeg ist ein freies Multimediaframework, dass initial von Fabrice Bellard [44] gegründet wurde. Anfangs war es hauptsächlich zur Video- und Audiodekodierung (insbesondere zur MPEG2-Dekodierung) geeignet, hat sich heutzutage aber zu einem komplexen Framework mit einer aktiven Community weiterentwickelt. Es besteht aus mehreren Komponenten [45]:

libavdevice Eine Sammlung von Input- und Outputdevices, z. B. um HTTP-Streams direkt zu öffnen oder den Ton direkt der Soundkarte zu übergeben.

libavcodec Die frühere Kernbibliothek. Eine Sammlung von Decodern und Encodern zu allen erdenklichen Codecs.

libavformat Eine Sammlung von Demuxern- und Muxern für diverse Containerformate.

libavfilter Eine umfangreiche Filterbibliothek, um diverse Filter auf Video- und Audiostreams anzuwenden.

libswscale Eine Bibliothek, die Skalierung, Farbraum- und Pixelformatänderungen implementiert.

libswresample Eine Bibliothek, die Audioresampling und Sampleformatänderungen ermöglicht.

Tools Mehrere Kommandozeilentools, um die Bibliotheken direkt über die Kommandozeile benutzen zu können.

Die Extraktion der Videosignaturen und ein entsprechender Lookupmechanismus ist dabei den Analysefiltern zuzuordnen und wurde dementsprechend als Filter in `libavfilter` implementiert.

4.1 Designentscheidungen

Der erste Ansatz bestand darin, ausschließlich die Extraktion als simplen V→V-Filter zu implementieren, d.h. der Videodatenstrom wird durch den Filter geleitet und dabei die Videosignatur extrahiert. Dieser Ansatz erwies sich als nicht ausreichend. Um die Güte der Signatur testen zu können, war es ebenso notwendig, den Lookupmechanismus zu implementieren. Dazu gab es prinzipiell 3 Ansätze:

1. Ein externes Programm schreiben, das die von FFmpeg errechneten Signaturen vergleicht.
2. Einen weiteren Filter schreiben, der die Daten mehrerer Signaturfilter entgegennimmt und den Vergleich durchführt.
3. Den Vergleichsmechanismus direkt in den Filter implementieren.

Nach Rücksprache mit dem Projekt [46] habe ich mich für den dritten Ansatz entschieden. Die Gründe dafür waren, dass Ansatz 1 den Nachteil hat, dass andere Programme in diesem Fall den Vergleichsalgorithmus nicht nativ benutzen können. Ansatz 2 ist komplizierter und schlechter benutzbar als Ansatz 3 und außerdem müsste der Vergleichsfilter entweder die Signaturfilter automatisch davorschalten (was mit der momentanen Filter-API nicht möglich ist) oder die Arbeit dem Nutzer überlassen und entsprechende Fehlermeldungen ausgeben. Ansatz 3 war das Mittel der Wahl, da die Filter-API das Konzept von dynamischen Inputs kennt. Somit kann man per Parameter aktivieren, ob man nur die Signatur ausgerechnet haben oder auch noch den Vergleich durchführen will und kann außerdem noch beliebig viele Inputvideoströme in den Filter geben. Außerdem konnten die verwendeten Datenstrukturen weiterverwendet werden und der Parsingschritt, den Ansatz 1 benötigen würde entfällt.

Aus diesen Gründen besteht der implementierte Filter aus zwei Teilen:

1. Der Extraktionsteil, der vom Code her in `filter_frame()` stattfindet.
2. Der Lookupteil, der in `lookup_signatures()` passiert und in eine separate Datei ausgelagert wurde.

Auf Dateiebene ist der Filter in 3 Dateien aufgeteilt:

signature.h Sammlung der verwendeten Datenstrukturen.

vf_signature.c Die Hauptdatei, die die Filterlogik und die Extraktion implementiert.

signature_lookup.c Alle Funktion, die zum Lookup-Mechanismus gehören, finden sich in dieser Datei.

Außerdem ist der Filter auf Pixelformate beschränkt, die einen separaten Lumakanal haben. Somit muss keine Farbraumwandlung ausgeführt werden[25].

4.2 Anbindung an das Framework

Die FFmpeg-Filterlogik ist an mehreren Stellen implementiert. Hauptsächlich passiert die Anbindung aber hier:

```
AVFilter ff_vf_signature = {
    .name          = "signature",
    .description   = NULL_IF_CONFIG_SMALL("Calculate the MPEG-7 video signature"),
    .priv_size     = sizeof(SignatureContext),
    .priv_class    = &signature_class,
    .init          = init,
    .uninit        = uninit,
    .query_formats = query_formats,
    .inputs        = NULL,
    .outputs       = signature_outputs,
    .flags         = AVFILTER_FLAG_DYNAMIC_INPUTS,
};
```

[25] Aufgrund dieser Beschränkung führt FFmpeg durch einen seperaten, vorgeschalteten Filter automatisch eine Farbraumumwandlung durch, falls RGB-Input o.ä. vorliegt.

Dort werden insbesondere die `init`, `uninit` und `query_formats` Funktionen zugewiesen, die später vom Framework aufgerufen werden. Zu beachten ist dabei auch der Nullpointer `inputs`, der die dynamische Anzahl von Inputs, also eingehenden Datenströmen, signalisiert.

Die tatsächliche Verknüpfung der Inputs passiert dann in der `init`-Funktion, die für jeden Input ein neues *AVFilterPad* erstellt und mit dem Input verbindet:

```
    for (i = 0; i < sic->nb_inputs; i++) {
        AVFilterPad pad = { 0 };

        pad.type = AVMEDIA_TYPE_VIDEO;
        pad.name = av_asprintf("in%d", i);
        if (!pad.name)
            return AVERROR(ENOMEM);
        pad.config_props = config_input;
        pad.filter_frame = filter_frame;
        ...
        if ((ret = ff_insert_inpad(ctx, i, &pad)) < 0){
            av_freep(&pad.name);
            return ret;
        }
    }
```

Für jedes der Inputs bzw. *AVFilterPads* wird dann die `filter_frame`-Funktion aufgerufen und die Signatur berechnet. Zusammengeführt wird das ganze in der `uinit`-Funktion, die neben der abschließenden Speicherfreigabe vorher für jedes Inputpaar den Lookup in Auftrag gibt und als Ergebnis darstellt.

4.3 Datenstrukturen

Die Länge des Videodatenstroms ist in FFmpeg nicht zwangsläufig vorher bekannt (z. B. bei Bildschirmaufnahmen oder HTTP-Livestreams). Daher war es notwenig, für die Videosignatur eine dynamische Struktur zu verwenden, sodass für die Finesignature eine doppelt verkettete und für die Coursesignature eine einfach verkettete Liste verwendet wird. Jede Coursesignature speichert außerdem noch einen Verweis auf die erste zu ihr gehörende Finesignature:

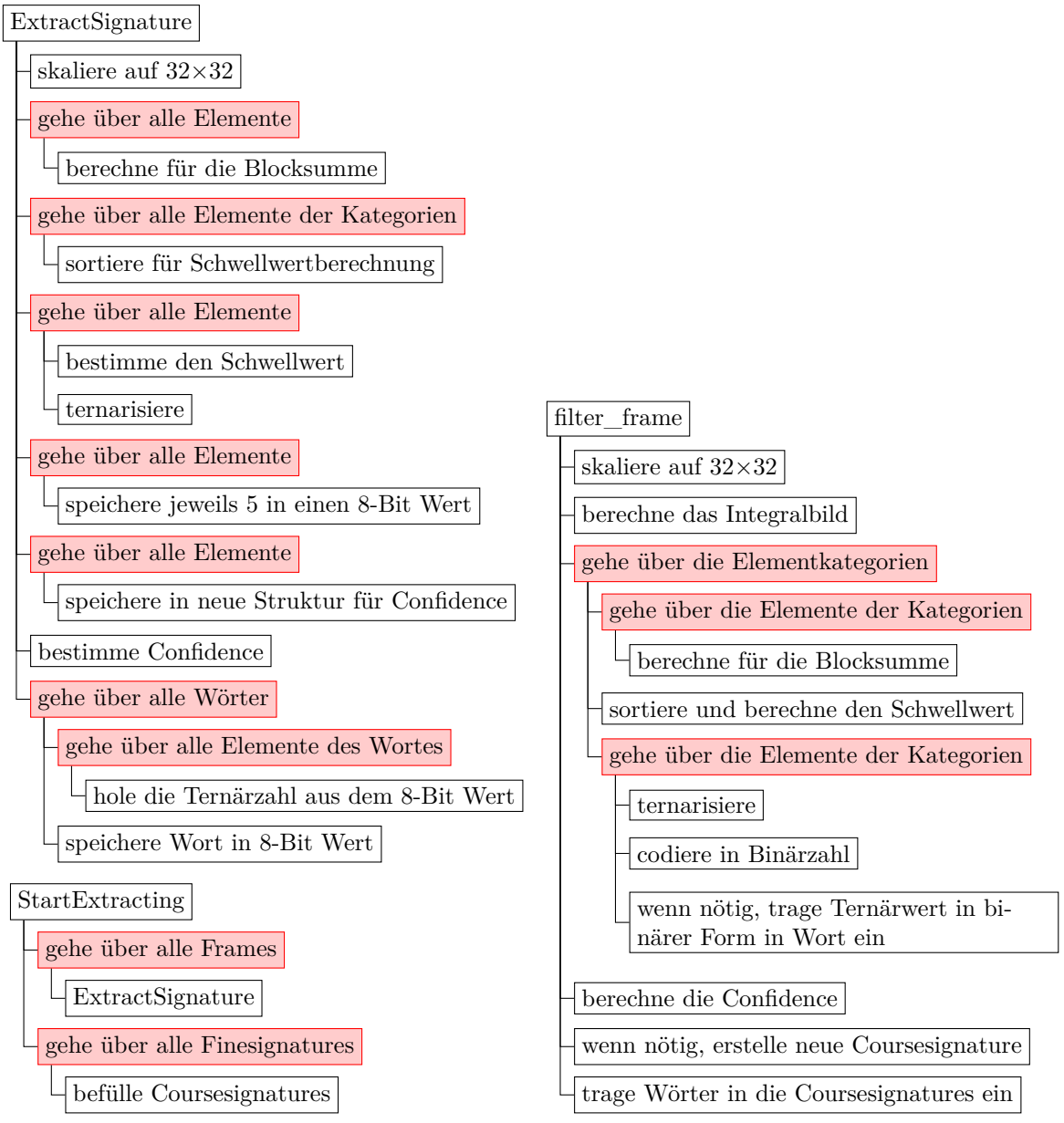
```
typedef struct FineSignature{
    struct FineSignature* next;
    struct FineSignature* prev;
    uint8_t confidence;
    uint8_t words[5];
    uint8_t framesig[SIGELEM_SIZE/5];
} FineSignature;

typedef struct CourseSignature{
    uint8_t data[5][31]; //5 words with min. 243 bit
    struct FineSignature* first; //associated Finesignatures
    struct CourseSignature* next;
} CourseSignature;
```

4.4 Implementierungsdetails

4.4.1 Extraktion

Die Extraktion passiert in der Methode `filter_frame`. `filter_frame` enthält als Parameter hauptsächlich das aktuelle Bild `AVFrame *picref`.

Die hier erstellte Implementierung besitzt eine andere Ablaufstruktur gegenüber der Referenzimplementierung. Beide Strukturen sind in Abbildung 11 dargestellt.



(a) Referenzimplementierung [32] (b) eigene Implementierung

Abbildung 11: Codeflussdiagramm der Extraktion des Referenzcodes (a) und des hier erarbeiteten Codes (b)

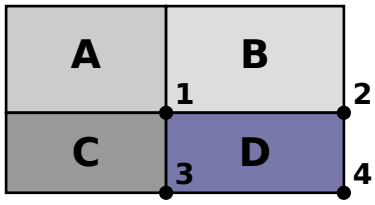


Abbildung 12: Block im Integralbild [47]

Man erkennt im Gegensatz zur Referenzimplementierung die Verwendung eines Integralbildes und die Konzentration auf möglichst wenige Schleifen. Die Schleife „gehe über alle Frames“ ist dabei in der eigenen Implementierung implizit in `filter_frame` enthalten, da diese Funktion für jeden Frame aufgerufen wird.

Integralbild Falls man viele Blocksummen auf denselben Pixeln bzw. auf derselben Matrix berechnen muss, kann man die Anzahl der Berechnungen durch ein Integralbild deutlich reduzieren.

Dazu berechnet man in jedem Bildpunkt die Summe des bis hierhin aufgespannten Blocks [47]:

$$I_\Sigma(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j)$$

In der Implementierung wird dazu der Wert des Pixels oberhalb des aktuellen Pixels auf die Summe der aktuellen Zeile bis zum aktuellen Pixel addiert. So erreicht man die komplette Integralbildberechnung in einer Iteration über die Matrix:

```
1    for (i=1; i<33; i++){
2        rowcount = 0;
         for(j=1; j<33; j++){
             rowcount += intpic[i][j];
             intpic[i][j] = intpic[i-1][j] + rowcount;
         }
7    }
```

Um anschließend beliebige Blöcke zu berechnen genügt es, die vier Eckpunkte aufzuaddieren (Abb. 12) [47]:

$$I_\Sigma(\text{Fläche}_D) = I_\Sigma(P_1) + I_\Sigma(P_4) - I_\Sigma(P_2) - I_\Sigma(P_3)$$

Für die anschließende Mittelwertberechnung muss noch durch die Fläche geteilt werden, die aber wegen der bekannten 4 Eckpunkte durch einfache Multiplikation der entsprechenden Koordinaten berechnet werden kann.

Ternärwert in Binärzahlumwandlung Der Referenzcode bietet zwei Methoden, die die Umwandlung von fünf Ternärwerten in eine 8-Bit Binärzahl ermöglichen:

```
1 static unsigned int Pack5TernaryValuesToA32bitInt(unsigned char * p_ternary_array);
unsigned char EncodeTo8BitBinary(unsigned int packed_ternary_value) const
```

Der Rückgabewert der ersten Methode ist dabei der Input für die zweite, die im wesentlichen alles bis auf die letzen 8 Bit abschneidet.

Die Implementierung in FFmpeg verzichtet auf die Sammlung von 5 Ternärwerten und codiert direkt als Binärzahl. Das ermöglicht die effizientere Implementierung in einer Schleife[26]:

```
1       for(j=0; j<elemcat->elem_count; j++) {
            ternary = get_ternary(elemsignature[j], th);
3           fs->framesig[f/5] += ternary * pot3[f%5];
            ...
            f++;
        }
```

Zuerst wird über die Elemente der jeweiligen Kategorie in der Framesignature iteriert und der Ternärwert entsprechend des Schwellwertes bestimmt. Anschließend wird dieser an der entsprechenden Position in `fs->framesig` (ein Array auf 8 Bit Binärzahlen) einsortiert. f ist dabei ein Zähler von 0 bis 380. Da immer 5 Ternärzahlen in eine 8 Bit Binärzahl codiert werden können, muss f durch 5 dividiert werden, um die Postition im Array zu erhalten. Auf die Zahl an dieser Position wird anschließend die Ternärzahl, multipliziert mit der jweiligen Potenz, addiert. pot3 enthält die benötigten Dreierpotenzen:

```
1 unsigned int pot3[5] = { 3*3*3*3, 3*3*3, 3*3, 3, 1 };
```

Auf ähnliche Art und Weise findet die Umwandlung der Ternärwerte in Binärzahlen für die Wörter statt.

Dieses Vorgehen erspart einem das „Sammeln“ von jeweils 5 Werten – der Ternärwert kann direkt verrechnet werden.

Wortextraktion Im Referenzcode wird am Ende der Blockberechnung über den kompletten Vektor iteriert und die Wörter an den entsprechenden Positionen extrahiert. Die Implementierung in FFmpeg verzichtet auf die zusätzliche Iteration und extrahiert die Wörter direkt nach Berechnung des jeweiligen Wertes. An derselben Stelle findet gleichzeitig auch die Ternär- in Binärzahlumwandlung statt.

```
1       for(j=0; j<elemcat->elem_count; j++) {
            ternary = get_ternary(elemsignature[j], th); //
            fs->framesig[f/5] += ternary * pot3[f%5];    //bereits von oben bekannt
4           if(f==wordvec[w]){
                fs->words[s2usw[w]] += ternary * pot3[wordt2b[s2usw[w]]++];
                if (w < 24) //solange es noch weitere Woerter gibt
                    w++;
            }
9           f++;
        }
```

[26] Die Methode `get_ternary` existiert nicht im Code, sie wurde hier nur als Äquivalent zum besseren Verständnis eingefügt.

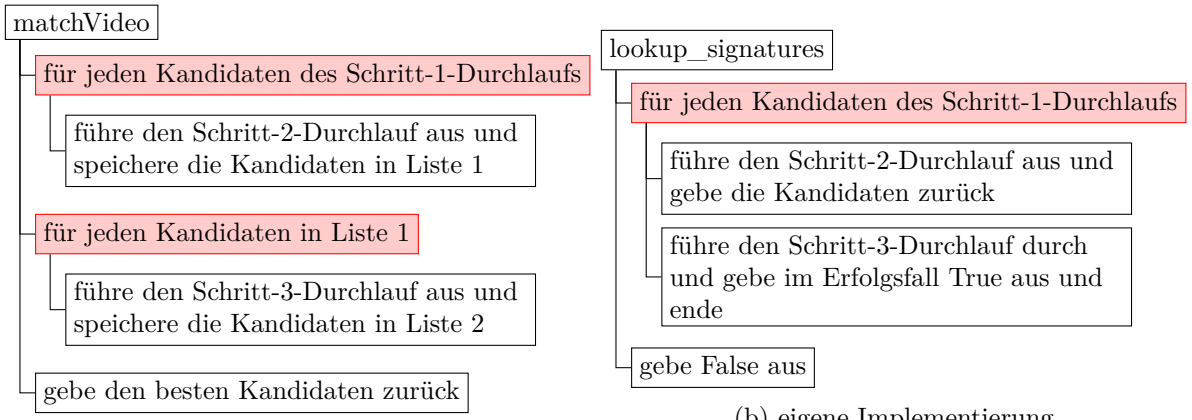
Der neue Code in diesem Abschnitt füllt die Wörter. Dazu existiert ein Vektor `wordvec`, der die Positionen in sortierter Reihenfolge enthält. Wenn f einer solchen Position entspricht (d.h. ein Wort muss befüllt werden), dann wird der Ternärwert `ternary` mit der entsprechenden Dreierpotenz malgenommen und auf das passende Word aufaddiert. Dieses erhält man, indem man im Vektor `s2usw` ebenfalls an Position w nachschaut, an der der Index des Wortes gespeichert ist, das zu w gehört.

Die richtige Dreierpotenz erhält man, indem man im Array `wordt2b` nachschaut, in dem wiederum die Position für den schon bekannten Vektor `pot3` enthalten ist. Da die Indizes innerhalb eines Wortes sortiert vorliegen, wird diese Position in jedem Schritt um eins erhöht, um die nächste Dreierpotenz zu erhalten.

4.4.2 Lookup

Der Lookup soll hier nur grob beschrieben werden. Der generelle Aufbau der Unterfunktionen des Referenzcodes und der Neuimplementierung ist ähnlich, die Funktion dieselbe, selbst wenn einige Codeteile umgangen werden mussten. So benutzt beispielsweise die Referenzimplementierung in der Hough-Transformationen einen C++-Vektor als dynamische Liste (die in C ohne weiteres nicht implementierbar ist). Es hat sich herausgestellt, das die Umsortierung in dieses Element unnötig ist, sodass es weggelassen werden konnte.

Der generelle Codefluss von beiden Implementierungen ist in Abbildung 13 dargestellt. In meiner Implementierung wird dabei im Gegensatz zum Referenzcode immer der erste Kandidat zurückgegeben. Dies ist ausreichend, da meine Implementierung bislang nur das direkte Matching unterstützt.



(a) Referenzimplementierung [32]

(b) eigene Implementierung

Abbildung 13: Codeflussdiagramm des Lookups des Referenzcodes (a) und des hier erarbeiteten Codes (b)

Die Zeitstempelberechnung erfolgt allerdings in beiden Implementierungen im Schritt 3. Somit wäre das partielle Matching in der Implementierung in FFmpeg ohne allzu großen Aufwand nachrüstbar. Weiterhin wurde der Zeitraum, bis zu dem ein gematchte Sequenz verworfen wird, auf 60 Frames (über einen anpassbaren Schwellwert festgelegt). Im Referenzcode wird dieser Zeitraum je nach Matchingverfahren verschieden errechnet. Die 60 Frames wurden gewählt, da dies bei 30 Frames/s zwei Sekunden entspricht, der kürzesten Matchingsequenz der Core-Experiments. Da

dieser Wert nicht fest im Code verankert ist, kann bei Bedarf die Mindestanzahl der gematchten Frames angepasst werden.

4.5 Eigene Messungen

4.5.1 Aufbau

Die Qualität der Videosignatur wurde umfangreich gemessen. Dabei wurden zusätzlich zu den Modifikationen in den Tests der Core-Experiments noch weitere durchgeführt, um Schwächen aufzudecken, die bislang nicht aufgetreten sind. Die detaillierten Modifikationen sind in Tabelle 3 dargestellt.

Name	Variationen	FFmpeg-Filter	Kommentar
FDUP		framestep, fps	Eliminierung jedes zweiten Bildes und anschließende Ersetzung durch das Vorgängerbild
CROP	0, 5, 10	crop	Wegschneiden von jeweils n % jeder Kante in jeder Kombination (also z. B. 0 % links 5 % oben, 5 % rechts und 10 % unten)
ACROP	0, 5	crop	Wegschneiden von jeweils n % jeder Kante in jeder Kombination von beiden zu testenden Videoseqenzen
BOX	10, 20, 30	drawbox	Überlagerung des Bildes mit einer Box mit den Seitenlängen von n % der Bildseitenlängen
GITT	5, 10, 20	drawgrid	Überlagerung des Bildes mit einem Gitter der Dicke 2 Pixel und der Gitterbreite und -höhe von n Pixeln
VS		vflip	Vertikales Spiegeln an der senkrechten Achse
HS		hflip	Horizontales Spiegeln an der senkrechten Achse
R180		vflip, hflip	Drehung um 180°
SC	10, 20, 30	scale	Skalierung auf n % der Seitenlängen des Originalbildes
HELL	36, -18, 9	lutyuv	Helligkeitsänderung um n, jeweils in den Grenzen von 0-255
NOISE	10, 20, 30, 40	noise	Hinzufügen von computergenerierten Rauschen der Stärke n
ROT	1, 2, . . . , 10	rotate	Rotation des Bildes um $n°$
GAMMA	0,5, 0,8, 1,25, 2	lutyuv	Modifikation mithilfe einer Gammakorrektur[27]
IL		interlace	Prüfen nach einer Umwandlung in ein Interlacedbildes
PERS	0, 5, 10	perspective	Anpassen der vertikalen Lotgraden um n der Breite in allen Kombinationen
DENOI	2, 3, . . . , 10	hqdn3d	Anwendung des hqdn3d-Filter zum Entrauschen der Luma-Spatial-Stärke n
BIT	10^{-4}, 10^{-3}, 0,001, 0,1	-	Verfälschen des Bildes mit Bitfehlern um n %. Die Sequenzen wurden vorher mit n % zufällig verteilten Bitfehlern verfälscht.
BR	35, 40, 45	-	Recodierung mit x264 und dem CRF[28] n. Die Videosequenzen wurden vorher recodiert.

Tabelle 3: Überblick über die Modifikationen der eigenen Tests

[27] Im Wesentlichen $y_{neu} = y_{alt}^{\gamma}$ mit y_{neu} bzw y_{alt} als neuem bzw. altem Helligkeitswert und γ als Gammawert (etwaige Überläufe werden abgefangen).

[28] „Constant Rate Factor“, Parameter um möglichst ein ähnliches Qualitätslevel über die gesamte Sequenz beizubehalten.

Name	Frames	Breite	Höhe	Name	Frames	Breite	Höhe
BasketballDrive.mp4	501	1920	1080	RaceHorses.mp4	300	832	480
BlowingBubbles.mp4	501	416	240	SlideShow.mp4	500	1280	720
BQMall.mp4	601	832	480	soccer30.mp4	300	704	576
SteamLocomotiveTrain.mp4	300	2560	1600	soccer.mp4	600	704	576
BQSquare.mp4	601	416	240	sunflower.mp4	500	1920	1080
BQTerrace.mp4	601	1920	1080	Tennis.mp4	240	1920	1080
ChinaSpeed.mp4	500	1024	768	tractor.mp4	690	1920	1024
KristenAndSara.mp4	600	1280	720	Traffic.mp4	300	2560	1600
Flowervase.mp4	301	832	480	tree.mp4	250	720	576
PeopleOnStreet.mp4	150	2560	1600	vidyo1.mp4	600	1280	720
FourPeople.mp4	600	1280	720	vidyo3.mp4	600	1280	720
Johnny.mp4	600	1280	720	vidyo4.mp4	600	1280	720
Kimono1.mp4	240	1920	1080	waterfall.mp4	260	720	480
ParkScene.mp4	240	1920	1080				

Tabelle 4: Das verwendete Testset

4.5.2 Testset

Für die Tests wurden 27 Videos in Auflösungen von 416×240 bis 2560×1600 verwendet. Jede der Sequenzen läuft durch jeden Test in jeder Kategorie und wird gegen alle anderen getestet, was eine Gesamtzahl von etwa 300 000 Tests ergibt. Die Videosequenzen sind dabei offiziell von der MPEG zur Verfügung gestellte Testsequenzen (u.a. [48]). Sie werden in Tabelle 4 vollständig aufgeführt.

4.5.3 Ergebnisse

Da der Lookup mithilfe einer Signatur ein Test mit einer „Ja/Nein“-Entscheidung ist, kann man diesen nach dem Prinzip der Klassifikation evaluieren, als Klassifikator dient hierbei die Signatur [49].

Es gelten folgende Abkürzungen:

	Test positiv	Test negativ
Videossequenzen gleich	richtig positiv (tp)	falsch negativ (fn)
Videossequenzen ungleich	falsch positiv (fp)	richtig negativ (tn)

Daraus ergeben sich folgende Formeln:

$$\text{Sensitivität (tprate)} = \frac{tp}{tp+fn} \qquad \text{Spezifität (tnrate)} = \frac{tn}{fp+tn}$$

$$\text{Falsch-Negativ-Rate (fnrate)} = \frac{fn}{tp+fn} \qquad \text{Ausfallrate (fprate)} = \frac{tp}{fp+tn}$$

Generell gewünscht sind dabei eine gute Sensitivität und Spezifität und eine geringe Falsch-Negativ- und Ausfallrate.

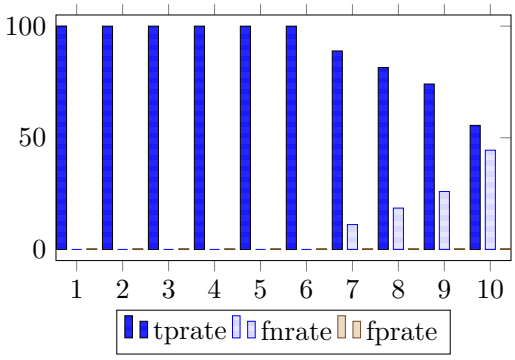
Test	tprate	fnrate	tnrate	fprate	Test	tprate	fnrate	tnrate	fprate
CROP	94,81	5,19	99,71	0,29	SC	96,29	3,71	99,71	0,29
VS	3,70	96,30	100,00	0,00	IL	100,00	0,00	99,71	0,29
BOX	100,00	0,00	99,71	0,29	DENOI	98,35	1,65	99,71	0,29
HS	22,22	77,78	99,71	0,29	GITT	100,00	0,00	99,71	0,29
GAMMA	100,00	0,00	99,71	0,29	BIT	26,92	73,08	99,92	0,08
HELL	100,00	0,00	99,71	0,29	ROT	90,00	10,00	99,71	0,29
R180	3,70	96,30	100,00	0,00	NOISE	98,14	1,86	99,71	0,29
ACROP	96,72	3,28	99,71	0,29	FDUP	100,00	0,00	99,71	0,29
BR	94,80	5,20	99,83	0,17	PERS	99,44	0,56	99,71	0,29

Tabelle 5: Ergebnisse der Tests für die Implementierung nach dem Standard

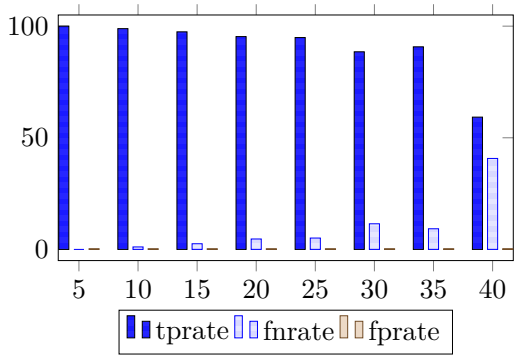
Die Ergebnisse der Tests für die neue Implementierung des Standards sind in Tabelle 5 zu finden. Gut zu erkennen sind die hohe Sensitivität und geringe Ausfallrate der meisten Tests. Eine vergleichsweise niedrige fprate ist beim Bitfehlertest zu erkennen. Bitfehler müssen vom Dekoder behandelt werden, sodass die Signaturbildung und Erkennung in diesen Fällen stark von diesem abhängig ist. Die Fehlerraten sind außerdem im Vergleich zu realen Anwendungen eher hoch, sodass eine Erkennungsrate von etwa 27 % durchaus einen guten Wert darstellt.

Ähnliches gilt für die Bitratentest, der bei einer Sensitivität von knapp unter 95 % liegt. Dort wurden CRF-Werte von 40 und 45 verwendet, die eine sehr geringe Qualität liefern und starke Artefakte hervorrufen. Somit sind knapp unter 95 % erstaunlich hoch.

Weitere Tests, die unter 95 % liegen, sind Rotation und Cropping. Für diese Tests sind die Raten für die jeweiligen Kategorien in Abb. 14 zu finden. Man erkennt bei der Rotation eine stetige



(a) Rotation in Grad



(b) Cropping; Kategorien aus Summen der Prozentwerte, die abgeschnitten wurden, z. B. oben 5 %, links 10 %, sonst 0 % ergibt Kategorie 15

Abbildung 14: Erkennungsrate bei Rotation und Cropping. Raten in %

Abnahme der Sensitivität ab 7°. Davor lagen die Werte bei 100 %. Da 7° bereits eine starke Rotation darstellt, kann man auch hier von einer guten Erkennungsrate sprechen. Analoges gilt für das Cropping, das zwar insgesamt schlechtere Werte als die Rotation liefert, einen stärkeren Abfall der Sensitivität aber erst aufweist, wenn von allen Kanten jeweils 10 % der Höhe bzw. Breite des Bildes abgeschnitten werden.

Vergleicht man die Daten mit denen der Core-Experiments, so muss man vor allem die Tests BOX, HELL, BR, SC, IL und FDUP betrachten. Generell sind die Resultate hier besser, allerdings sind die Modifikationen hier auch schwächer. Somit sind die Ergebnisse am besten mit denen der schwachen Modifikationen in den Core-Experiments zu vergleichen. Dort werden mit beiden Verfahren Raten von zumeist über 98 % erzielt.

Was signifikant fehlschlägt ist die vertikale Spiegelung und die Rotation um 180° mit jeweils nur 3,7 %, sowie die horizontale Spiegelung mit einer Sensitivität von 22,22 %. Dies wurde zum Anlass genommen, die Signatur genau auf diese Missstände hin zu optimieren.

4.6 Korrektheit

In [33] werden Testsequenzen mit dazugehörigen Signaturen gestellt. Diese wurden zum Vergleich herangezogen. Es konnte nachgewiesen werden, dass die Signatur bis auf wenige Ternärwerte in der Framesignature (die um eins verschieden waren) identisch ist. Die Anzahl dieser Ternärwerte liegt deutlich unter den 15, die in [33] noch als akzeptabel erklärt werden.

5 Spiegelsignatur

Die Ergebnisse des letzten Abschnittes haben eine Anfälligkeit gegen jegliche Art der Spiegelung gezeigt (auch eine Rotation um 180° kann man als Spiegelungsoperation betrachten, da es die Hintereinanderausführung einer vertikalen und horizontalen Spiegelung ist). In diesem Abschnitt soll eine Verbesserung der Signatur erarbeitet werden, die den Filter robust gegen diese drei Modifikationen macht, ähnlich robust gegen die anderen Modifikationen ist, eine ähnliche Performance zeigt und sich in die bestehende Technik gut integriert.

5.1 Grundidee

Der offensichtliche Ansatz, gedrehte, gespiegelte oder auf andere Art geometrisch transformierte Videos zu erkennen, ist, die entsprechende Rücktransformation anzuwenden und danach den Signaturalgorithmus anzuwenden. Um so vertikal, horizontal und um 180° rotierte Videosequenzen zu erkennen, sähe die Implementierung etwa so aus:

```
for video in zu_erkennende_videos:
    pruefe_ob_video_erkannt_wird
    if(pruefung_erfolgreich)
        return erkannt
    else
        spiegele_video_horizontal
        pruefe_ob_video_erkannt_wird
        if(pruefung_erfolgreich)
            return erkannt
        else
            ...
```

Mit diesem Ansatz müssten im schlechtesten Fall alle Operationen nacheinander angewendet werden und somit vier Mal die Signatur berechnet und verglichen werden.

Der hier vorgestellte Ansatz unterscheidet sich davon signifikant, da er die Signatur so verändert, dass die geometrische Information gewahrt bleibt und die Coursesignature sogar für alle 3 Fälle gleich ist. Somit muss in jedem Fall nur einmal die Signatur berechnet werden und die geometrische Erkennung kann auf die späteren Lookup-Schritte beschränkt werden. Es ergibt sich also dieses Muster:

```
for video in zu_erkennende_videos:
    pruefe_ob_video_erkannt_wird
    if(pruefung_erfolgreich)
        return erkannt
    else
        return nicht erkannt
```

Der Lookup ist dabei so gestaltet, dass er maximal effizient arbeitet und insbesondere nicht oder erst am Ende die Signatur spiegelt und dreht. Im Vergleich zu dem trivialen Ansatz führt dies zu einer deutlichen Zeitersparnis und reduziert auch noch die Menge der übertragenden Daten, da nur noch eine Signatur berechnet und übertragen werden muss.

Die Signatur besteht im Wesentlichen aus den 380 Blöcken. Der Rest wird auf Basis dieser Blöcke berechnet. Betrachten wir jetzt ein achsensymmetrisches Bild (siehe Abb. 15), dann fällt auf, dass weder eine horizontale, noch eine vertikale Spiegelung, noch eine Drehung um 180° das Bild ändern.

Abbildung 15: Beispiele für achsensymmetrische Bilder (bei der Uhr ohne Zeiger), Lizenz: CC0

Der hier verwendete Ansatz, um die Spiegelungen mit in der Signatur zu „speichern“, ist es, die 380 Elemente entsprechend achsensymmetrisch zu gestalten. Das garantiert, dass ein Block im Originalbild, der eventuell zu einer 2 in der Framesignature führt, auch nach einer Spiegelung zu der gleichen 2, wenn auch an anderer Stelle in der Signatur, führt.

5.2 Spiegelungsinvariante Signatur

5.2.1 Framesignature

Die Framesignature bildet sich aus den 380 Ternärelementen, die wiederum durch eine Schwellwertberechnung, basierend auf der jeweiligen Kategorie, entstehen. Somit ist es notwendig, die Achsensymmetrie innerhalb der Kategorien zu erhalten. Sei dazu beispielhaft die Kategorie A gegeben (siehe Abb. 16), die aus 8 Elementen besteht. Nach Berechnung ergeben diese die Werte:

$$-129, -101, 71, -34, -50, -233, 142, 66$$

Der Threshold th ist jetzt das Element nach einem Drittel der sortierten Liste:

$$-233, -129, \underbrace{-101}_{th}, -50, -34, 66, 71, 142$$

Damit ergeben sich als Ternärwerte:

$$0, 1, 1, 1, 1, 0, 2, 1$$

Wenn das Bild jetzt vorher vertikal gespiegelt wird, ergeben sich durch die besondere Anordnung der Elemente in A immer noch die gleichen 8 Zahlen, nur in der Reihenfolge:

$$71, -101, -129, -50, -34, 66, 142, -233$$

Da sich die Elemente der Liste nicht geändert haben, bleibt auch th gleich und es ergeben sich als Ternärwerte:

$$1, 1, 0, 1, 1, 1, 2, 0$$

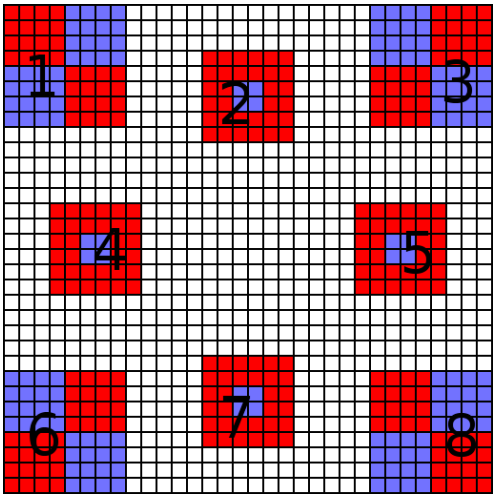


Abbildung 16: Grafische Darstellung der Blöcke von Beispielkategorie A. Die blauen Flächen werden dabei von den roten Flächen abgezogen.

Durch die symmetrische Anordnung hat sich also nur die Reihenfolge der Ternärwerte geändert, sie selbst sind gleich geblieben.

Der erste Schritt zu einer Videosignatur, die robust gegenüber Spiegelungen ist, war also die Symmetrisierung der Kategorien. A1, A2, D5, D6 und D7 waren schon symmetrisch, dementsprechend mussten D1, D2, D3, D4 und D8 angepasst werden:

D1 In D1 mussten Gruppen von jeweils vier Elementen angepasst werden. Abbildung 17 stellt so eine Gruppe dar.

D2 In D2 mussten von einigen Elementen die Richtung der Differenzbildung angepasst werden. In Abbildung 17 macht sich dies durch eine Vertauschung der Farben bemerkbar.

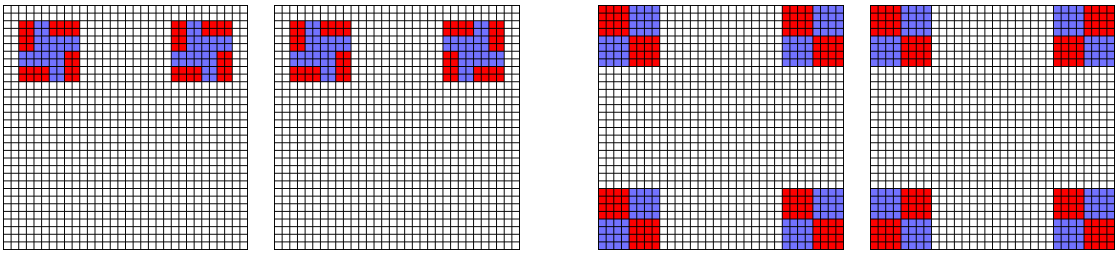


Abbildung 17: Grafische Gegenüberstellung von jeweils 8 Elementen aus der D1 Kategorie (links) und jeweils 4 Elementen aus der D2 Kategorie. Jeweils links ist die Signatur nach dem Standard, rechts die symmetrisierte Version dargestellt.

D3 D3 bildet in einem 9×9-Raster sämtliche Paare. Diese waren aber in Gänze asymmetrisch. Bei einigen Paaren war es möglich, durch Vertauschung wie in D2 zu symmetrisieren. Einige Elemente musste ich aber verwerfen und durch andere ersetzen. Abbildung 18 stellt dazu einen Fall dar.

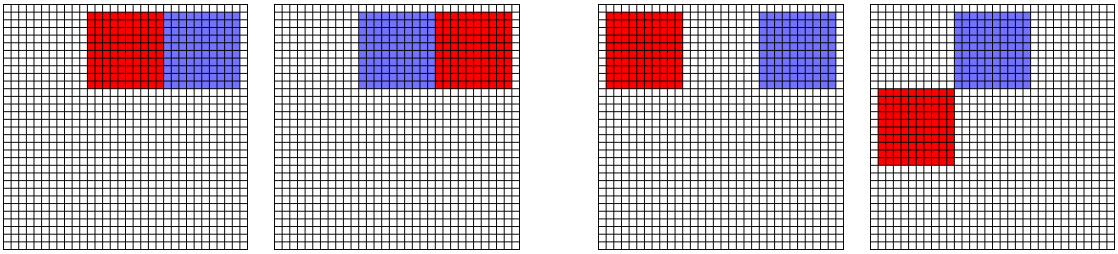


Abbildung 18: Grafische Gegenüberstellung von 2 Elementen aus der D3 Kategorie. Bei dem ersten Element (die beiden linken Bilder) hat eine Vertauschung ausgereicht. Bei dem zweiten Element musste das Element komplett durch ein anderes ersetzt werden. Die bearbeitete Version ist jeweils rechts.

D4 Für D4 gilt in etwa dasselbe wie für D3. Bis auf 4 Elemente war keines symmetrisch und konnte auch nicht auf triviale Weise umgewandelt werden. Dennoch habe ich das Muster beibehalten. Für einige Beispiele siehe Abbildung 19. Einige Elemente musste ich aber komplett verwerfen und durch andere ersetzen.

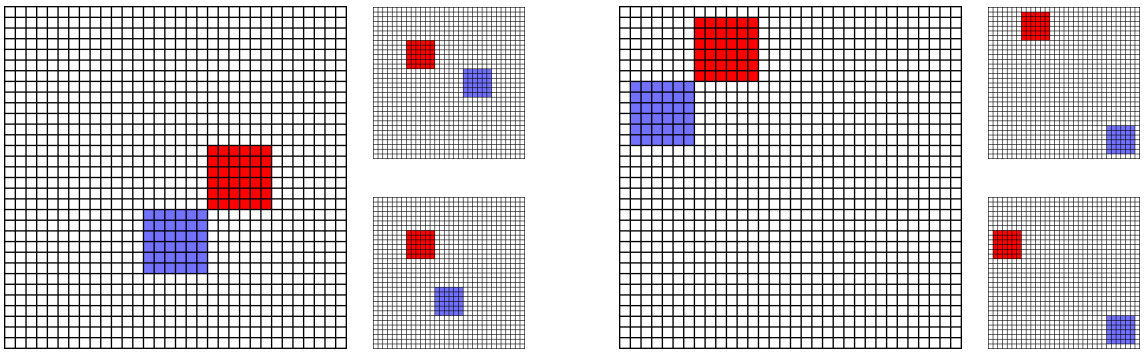


Abbildung 19: Grafische Gegenüberstellung von 2 Elementen aus der D4 Kategorie. Das Originalelement kann dabei immer durch zwei Elemente ersetzt werden (die kleinen Bilder), sodass die Elemente der Kategorie wieder symmetrisch sind (es gibt für jede Ecke so ein Element). Durch die Ersetzung mit jeweils 2 Bildern müssen andere aber verworfen werden.

D8 D8 ist ähnlich aufgebaut wie D1 nur mit größeren Blöcken. Deswegen gelten für diese Kategorie auch die gleichen Aussagen wie für D1.

Confidence Der Confidence-Wert wird immer noch auf die gleiche Weise gebildet und kann genauso weiterverwendet werden.

5.2.2 Bag-of-Words

Der Sinn hinter der Änderung der Wörter ist es, die Information über die Spiegelung in der Coursesignature beizubehalten oder sie anzugleichen. Mein erster Ansatz war, für jedes der fünf Wörter vier symmetrische Elemente zu nehmen und ein Element, dass in der Mitte liegt und somit in sich selbst symmetrisch ist. Dieser Weg hat sich als nicht zielführend erwiesen.

Sei beispielhaft das erste Wort: $2, 2, 1, 0, 1$. Dann ergibt das die Zahl 226, die in der Coursesignature in das entsprechende Diagramm eingetragen wird. Nach einer Spiegelung seien die Elemente

jetzt so angeordnet: $1, 2, 0, 2, 1$ (die letzte 1 bleibt immer gleich, wegen des in sich symmetrischen Elements). Dann ergibt das die Zahl 142, die dann in das Histogramm eingetragen wird. Man könnte jetzt theoretisch die Elemente aus der 142 rekonstruieren, allerdings weiß man im Histogramm (bzw. in der Coursesignature) nicht mehr, wann die 142 auftrat und nach der Binarisierung auch nicht mehr wie oft.

Um dieses Problem zu umgehen, habe ich die Wortvektoren geändert und um 4 Wörter erweitert. Es ergeben sich damit das *symmetrische Wort* und 4 *Doppelwörter*.

Symmetrisches Wort Das symmetrische Wort besteht wie im Original aus 5 Ternärwerten, allerdings sind alle 5 Elemente in sich selbst symmetrisch, siehe dazu auch Abbildung 20. Die Elemente sind im Einzelnen:

$$\{24, 31, 168, 290, 305\}$$

Durch diesen Aufbau ist das Wort, gleichgültig welche Spiegelung angewendet wurde, immer gleich.

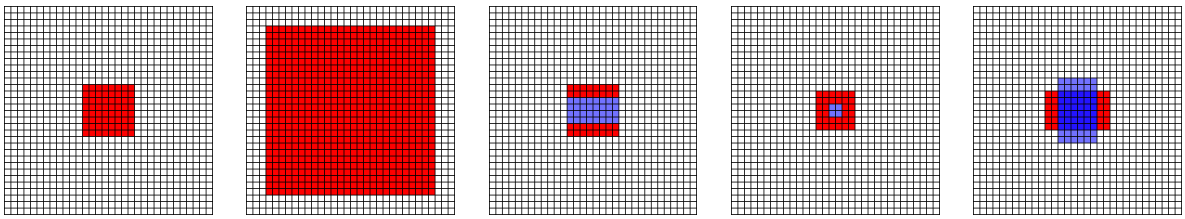


Abbildung 20: Grafische Darstellung der Elemente des symmetrischen Wortes.

Doppelwörter Die Doppelwörter haben keine direkte Entsprechung in der Framesignature, sondern müssen berechnet werden. Jedes Doppelwort hat dazu 5 Vektoren mit jeweils 4 Elementen der Framesignature. Diese 4 Elemente sind symmetrisch angeordnet (Abbildung 21). Die genaue

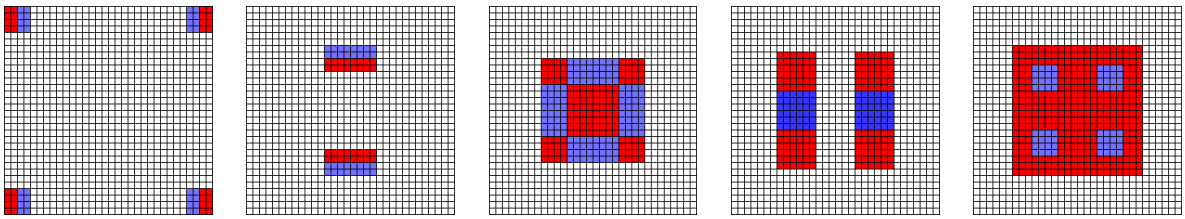


Abbildung 21: Grafische Darstellung der Elemente des Doppelwortes 2. Pro Bild sind die jeweils 4 Elemente des Vektors abgebildet. Die Elemente im vierten Bild überlappen sich.

Auflistung der Vektoren ist:

$$\underbrace{\begin{pmatrix}44\\57\\70\\83\end{pmatrix}\begin{pmatrix}175\\190\\196\\205\end{pmatrix}\begin{pmatrix}225\\228\\270\\303\end{pmatrix}\begin{pmatrix}269\\270\\273\\274\end{pmatrix}\begin{pmatrix}301\\303\\307\\309\end{pmatrix}}_{\text{Doppelwort 1}} \qquad \underbrace{\begin{pmatrix}32\\53\\74\\95\end{pmatrix}\begin{pmatrix}102\\103\\124\\125\end{pmatrix}\begin{pmatrix}153\\154\\157\\158\end{pmatrix}\begin{pmatrix}213\\216\\217\\220\end{pmatrix}\begin{pmatrix}248\\249\\250\\251\end{pmatrix}}_{\text{Doppelwort 2}}$$

$$\underbrace{\begin{pmatrix}100\\101\\126\\127\end{pmatrix}\begin{pmatrix}254\\256\\259\\261\end{pmatrix}\begin{pmatrix}280\\282\\298\\300\end{pmatrix}\begin{pmatrix}320\\328\\340\\348\end{pmatrix}\begin{pmatrix}363\\366\\369\\372\end{pmatrix}}_{\text{Doppelwort 3}} \qquad \underbrace{\begin{pmatrix}109\\111\\116\\118\end{pmatrix}\begin{pmatrix}165\\167\\169\\171\end{pmatrix}\begin{pmatrix}268\\271\\272\\275\end{pmatrix}\begin{pmatrix}284\\286\\294\\296\end{pmatrix}\begin{pmatrix}313\\317\\353\\357\end{pmatrix}}_{\text{Doppelwort 4}}$$

Um daraus das Element des Doppelwortes zu berechnen, werden die 4 Elemente summiert und die Summe als Ternärzahl codiert. Die Summe von vier Ternärelementen liegt zwischen 0 und 8. Damit ist immer eine eindeutige Codierung in zwei Ziffern einer Ternärzahl gewährleistet. Die erste Ziffer wird dann im ersten Wort des Doppelwortes gespeichert, die zweite Ziffer im zweiten Wort.

Durch die Summierung wird erreicht, dass egal welche Spiegelungen vorher ausgeführt werden, die Wörter immer gleich sind. Seien dazu diese Ternärzahlen gegeben:

$$\begin{pmatrix}0\\2\\0\\0\end{pmatrix}\begin{pmatrix}2\\0\\1\\1\end{pmatrix}\begin{pmatrix}1\\2\\2\\0\end{pmatrix}\begin{pmatrix}2\\1\\0\\0\end{pmatrix}\begin{pmatrix}1\\1\\2\\2\end{pmatrix}$$

Das entspräche den Summen $\{2,4,5,3,6\}$ und damit den Ternärzahlen $\{(02)_3,(11)_3,(12)_3,(10)_3,(20)_3\}$ und ergäbe dieses Doppelwort:

$$\underbrace{\{0,1,1,1,2\}}_{\text{Wort 1}} = 41 \qquad \underbrace{\{2,1,2,0,0\}}_{\text{Wort 2}} = 207$$

Wenn der Vektor durch eine vorherige Spiegelung jetzt so aussähe, blieben die Wörter gleich:

$$\begin{pmatrix}0\\0\\0\\2\end{pmatrix}\begin{pmatrix}1\\1\\2\\0\end{pmatrix}\begin{pmatrix}2\\0\\1\\2\end{pmatrix}\begin{pmatrix}0\\0\\2\\1\end{pmatrix}\begin{pmatrix}2\\2\\1\\1\end{pmatrix}$$

Prinzipiell hätte auch eine Mittelwertbildung über die 4 Ternärwerte oder die bloße Speicherung der ersten bzw. zweiten Ziffer genügt. Die Speicherung in 2 Wörtern erlaubt aber die Codierung von 9 Werten im Gegensatz zu 3 und erhält so die volle Information der Summe, die bereits die spezifische Information über die Lage verwirft.

5.2.3 Coursesignature

Die Coursesignature muss jetzt noch auf die anderen Wörter angepasst werden. Dazu wird für das symmetrische Wort, wie gehabt, ein binäres Histogramm gebildet. Bei den Doppelwörtern werden

für jede Finesignature immer beide Wörter des Doppelwortes in das Histogramm eingetragen und anschließend binarisiert.

Durch diese Technik ist die Coursesignature von gespiegelten Videosequenzen identisch zu der von ungespiegelten Videosequenzen, d.h. insbesondere, dass der Lookup-Schritt 1, der ausschließlich auf den Coursesignatures basiert, nicht angepasst werden muss.

5.2.4 Symmetrische L_1-Distanz

Um die Spiegelsignatur sinnvoll einzusetzen, müssen auch die Lookup-Schritte 2 und 3 angepasst werden. Diese basieren maßgeblich auf der L_1-Distanzbildung, um die Ähnlichkeit zweier Framesignatures zu bestimmen. Die L_1-Distanz versagt allerdings bei der Erkennung von Spiegelungen, darum muss ein anderes Verfahren gefunden werden.

Der Ansatz hierzu kombiniert dabei die L_1-Distanz und die Summenbildung, die in ähnlicher Form schon in der Wortbildung auftritt.

Zunächst werden alle 380 Elemente untersucht und in drei Kategorien unterteilt.

1. Die Kategorie K_1 besteht aus den Elementen, die von alleine symmetrisch sind. Einige Beispiele sind im symmetrischen Wort zu sehen. Sie besteht aus 12 Elementen.
2. Die Kategorie K_2 besteht aus Difference-Elementen, bei der beide Blöcke auf einer der Spiegelachsen liegen und die paarweise auftreten. In dieser Kategorie gibt es 22 Elemente mit jeweils 2 Unterelementen.
3. Die Kategorie K_3 besteht aus den Elementen, die auch in den Doppelwörtern auftreten und bei denen immer 4 Elemente zueinder symmetrisch sind. Von dieser Kategorie gibt es 81 Elemente mit jeweils 4 Unterelementen.

Um nun daraus die symmetrische L_1-Distanz zu bilden, wird folgende Rechnung ausgeführt:

$$\sum_{i}^{K_1} \left| E^i_{K_1,2} - E^i_{K_1,1} \right| + \sum_{i}^{K_2} \left| \left(\sum_{k=0}^{2} E^{i,k}_{K_2,2} \right) - \left(\sum_{k=0}^{2} E^{i,k}_{K_2,1} \right) \right| + \sum_{i}^{K_3} \left| \left(\sum_{k=0}^{4} E^{i,k}_{K_3,2} \right) - \left(\sum_{k=0}^{4} E^{i,k}_{K_3,1} \right) \right|$$

$E^{i,j}_{K_x,y}$ ist das j-te Unterelement des i-ten Elementes in der Kategorie x aus der Framesignature y des Paares, das verglichen werden soll. Im Grunde entspricht obige Rechnung der L_1-Distanzbildung, nur dass vorher noch die Summe der symmetrischen Elemente gebildet wird, falls es mehrere gibt.

5.2.5 Lookup-Schritte 2 und 3

In den Lookup-Schritten 2 und 3 wird die L_1-Distanz beide Male im Vergleich zum Schwellwert Thx_H eingesetzt. Da die symmetrische L_1-Distanz auf dem gleichen Verfahren basiert, kann diese Technik übernommen werden. Es muss nur gegen einen anderen Schwellwert $Thx_{H,sym}$ geprüft werden. Die Höhe dieses Schwellwertes wurde experimentell ermittelt (siehe Abschnitt 6).

Zur Abstandsbestimmung werden deshalb sämtliche Vorkommen der L_1-Distanz mit der symmetrischen L_1-Distanz ersetzt und gegen $Thx_{H,sym}$ geprüft anstatt Thx_H.

5.3 Qualitätsbetrachtung

Der neue bzw. angepasste Signaturalgorithmus hat diverse Vor- und Nachteile. Als Nachteile seien zu nennen:

- Die längere Laufzeit durch erhöhten Rechenaufwand
- Die Erhöhung des Speicherbedarfs, da 4 Wörter pro Frame zusätzlich gespeichert werden müssen.

Dem gegenüber stehen folgende Vorteile:

- Erkennung von horizontalen und vertikalen Spiegelungen (auch hintereinander ausgeführt) mit einer Erkennungsrate von 100 %.
- Durch Integration in die Signatur deutlich weniger erhöhter Rechen- und Speicheraufwand als mit dem trivialen Ansatz.
- Durch Integration kompatibel mit dem alten Lookupmechanismus (einzig der Parser für den Signaturbitstrom muss angepasst werden).

Auch für die geänderte Signatur wurden Tests durchgeführt. Diese sind identisch zu den in Abschnitt 4.5.1 beschriebenen, benutzen also insbesondere auch die gleichen Testsequenzen und garantieren so die Vergleichbarkeit. Sie werden detailliert im nächsten Abschnitt 6 ausgewertet.

6 Auswertung

6.1 Ablauf

Zusätzlich zu dem bereits vorgestellten Test der Originalsignatur wurden insgesamt 3 Testläufe mit der Spiegelsignatur gestartet. Diese hatten folgende Zielsetzungen:

Test der Spiegelsignatur mit altem Lookup Dieser Test benutzt für die Extraktion zwar die symmetrischen Elemente, verwendet für den Lookup aber ausschließlich das standardisierte Verfahren. Der Test soll sicherstellen, dass die Qualität der Signaturen selbst gleichwertig ist.

Test mit verschiedenen Schwellwerten Dieser Test benutzt die Spiegelsignatur mit neuem Lookup und varriiert $Thx_{H,sym}$ um den bestmöglichsten Wert herauszufinden.

Test der Spiegelsignatur mit neuem Lookup Dieser Test benutzt den im letzten Test ermittelten Schwellwert $Thx_{H,sym}$ und dient zum Evaluieren der Spiegelsignatur und zum Vergleich mit der alten Signatur. Er benutzt den neuen Lookup.

6.2 Test der Spiegelsignatur mit altem Lookup

Für diesen Test wurden dieselben Videosequenzen und dasselbe Testverfahren angewendet wie das in Abschnitt 4.5.1 beschriebene. In Tabelle 6 sind die Ergebnisse dieses Tests zu finden. Dort sind zu Vergleichszwecken ebenfalls die Ergebnisse aus Abschnitt 4.5.1 dargestellt.

Test	Signatur nach Standard				Signatur nach Anpassung			
	tprate	fnrate	tnrate	fprate	tprate	fnrate	tnrate	fprate
HS	22,22	77,78	99,71	0,29	22,22	77,78	99,71	0,29
ACROP	96,72	3,28	99,71	0,29	96,85	3,15	99,71	0,29
SC	96,29	3,71	99,71	0,29	96,29	3,71	99,71	0,29
ROT	90,00	10,00	99,71	0,29	91,48	8,52	99,71	0,29
GAMMA	100,00	0,00	99,71	0,29	100,00	0,00	99,71	0,29
HELL	100,00	0,00	99,71	0,29	98,76	1,24	99,71	0,29
BIT	26,92	73,08	99,92	0,08	47,66	52,34	99,82	0,18
DENOI	98,35	1,65	99,71	0,29	99,17	0,83	99,71	0,29
BR	94,80	5,20	99,83	0,17	93,50	6,50	99,83	0,17
GITT	100,00	0,00	99,71	0,29	100,00	0,00	99,71	0,29
PERS	99,44	0,56	99,71	0,29	99,67	0,33	99,71	0,29
BOX	100,00	0,00	99,71	0,29	100,00	0,00	99,71	0,29
IL	100,00	0,00	99,71	0,29	100,00	0,00	99,71	0,29
NOISE	98,14	1,86	99,71	0,29	99,07	0,93	99,71	0,29
FDUP	100,00	0,00	99,71	0,29	96,29	3,71	99,71	0,29
VS	3,70	96,30	100,00	0,00	3,70	96,30	100,00	0,00
R180	3,70	96,30	100,00	0,00	3,70	96,30	100,00	0,00
CROP	94,81	5,19	99,71	0,29	95,18	4,82	99,71	0,29

Tabelle 6: Ergebnisse der Tests mit altem Lookup für jede Kategorie

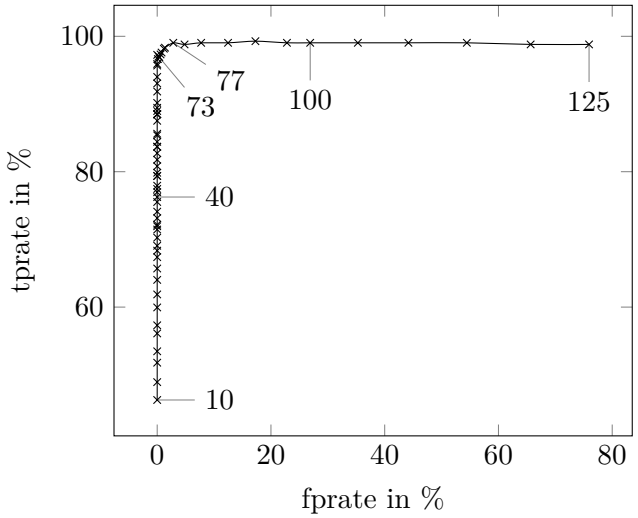


Abbildung 22: Sensitivität des Schwellwerttests in Abhängigkeit von der Ausfallrate, Verzeichnet ist außerdem der Schwellwert von ausgewählten Punkten.

Es ist zu erkennen, dass die Ergebnisse zwar abweichen, sich im Mittel aber wieder ausgleichen. Dies zeigt eine gleichwertige Extraktionsqualität der symmetrischen Signatur.

6.3 Test mit verschiedenen Schwellwerten

Dieser Test dient zur experimentellen Bestimmung des besten Schwellwertes. Die Beobachtung einer Stichprobe zeigte (bei zwei nicht zueinanderpassenden Videosequenzen) symmetrische L_1-Distanzen bis etwa 150.

Ausgehend von dieser Beobachtung wurde ein Test mit einem variierten Schwellwert über den Bereich von 10 bis 150 durchgeführt (zu den Randbereichen hin mit größerer Schrittweite) und zudem mit einem reduzierten Testset (da die Rechenzeit für Durchführung über das volle Testset den Zeitraum der Bearbeitungszeit dieser Arbeit gesprengt hätte) gearbeitet.

Das Ergebnis ist in einer „Receiver Operation Characteristic“ (ROC) auftragbar (Abbildung 22). Man erkennt hier sehr gut die stetige Verbesserung der tprate, deren Steigung aber bei einem Schwellwert von etwa 70 deutlich nachlässt und knapp unter 100 % liegt. Bei einem Wert von 74 macht allerdings der Wert der fprate erstmals einen Sprung auf 0,13 %, der dann bei 79 bereits auf 1,18 % gestiegen ist. Davor lag er immer bei nahezu 0 %. Da beim MPEG-7 Standard sehr auf eine geringe fprate geachtet wurde, wird hier 73 als Schwellwert gewählt.

Mit 73 ist $Thx_{H,sym}$ deutlich kleiner als Thx_H mit 115, was aber aufgrund der verminderten Anzahl von Differenzen (nur 115 statt 380) auch erwartet wurde.

6.4 Test der Spiegelsignatur mit neuem Lookup

Dieser Test verwendet den im Abschnitt 4.5.1 vorgestellten Testablauf, benutzt allerdings die Spiegelsignatur und einen Lookup unter ausschließlicher Verwendung der symmetrischen L_1-

Distanz mit dem oben ermittelten Schwellwert $Thx_{H,sym} = 73$. Die Ergebnisse sind in Tabelle 7 dargestellt (ebenfalls im Vergleich zu der Signatur nach Standard).

Test	Signatur nach Standard				Signatur nach Anpassung			
	tprate	fnrate	tnrate	fprate	tprate	fnrate	tnrate	fprate
ACROP	96,72	3,28	99,71	0,29	98,79	1,21	99,61	0,39
BIT	26,92	73,08	99,92	0,08	46,29	53,71	99,85	0,15
HELL	100,00	0,00	99,71	0,29	100,00	0,00	99,62	0,38
CROP	94,81	5,19	99,71	0,29	96,48	3,52	99,59	0,41
DENOI	98,35	1,65	99,71	0,29	99,58	0,42	99,71	0,29
FDUP	100,00	0,00	99,71	0,29	100,00	0,00	99,43	0,57
GAMMA	100,00	0,00	99,71	0,29	100,00	0,00	99,46	0,54
GITT	100,00	0,00	99,71	0,29	100,00	0,00	99,62	0,38
HS	22,22	77,78	99,71	0,29	100,00	0,00	99,71	0,29
IL	100,00	0,00	99,71	0,29	100,00	0,00	99,57	0,43
NOISE	98,14	1,86	99,71	0,29	99,07	0,93	99,71	0,29
BR	94,80	5,20	99,83	0,17	92,85	7,15	99,60	0,40
BOX	100,00	0,00	99,71	0,29	100,00	0,00	99,66	0,34
PERS	99,44	0,56	99,71	0,29	99,02	0,98	99,59	0,41
ROT	90,00	10,00	99,71	0,29	91,85	8,15	99,64	0,36
R180	3,70	96,30	100,00	0,00	100,00	0,00	99,71	0,29
SC	96,29	3,71	99,71	0,29	98,76	1,24	99,66	0,34
VS	3,70	96,30	100,00	0,00	100,00	0,00	99,71	0,29

Tabelle 7: Ergebnisse der Tests mit neuem Lookup mit ausschließlicher Verwendung der symmetrischen L_1-Distanz für jede Kategorie.

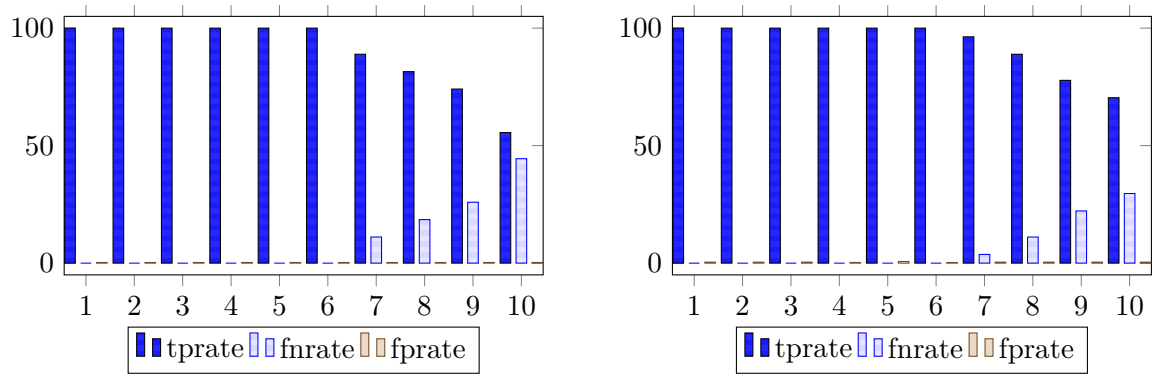
Man erkennt sehr gut die deutliche Verbesserung der Erkennungsraten der vertikalen und horizontalen Spiegelungen sowie der Rotation um 180°. Die sonstigen Raten bleiben nahezu identisch oder verbessern sich sogar.

Auch wenn man die Raten aufgeteilt nach den Kategorien der einzelnen Tests betrachtet, erhält man nahezu identische Ergebnisse. Als Beispiel sei hier die Rotation aufgeführt (Abbildung 23). So fällt in beiden Tests die Erkennungsrate erst bei Drehungen um 7°, liefert aber immer noch brauchbare Ergebnisse. Die Erkennungsrate mit dem neuen Verfahren vermindert sich sogar weniger als die des alten Verfahrens.

Die fprate vergrößert sich mit den neuen Lookup, bleibt allerdings dauerhaft unter 0,6 %. Dies stellt immer noch einen akzeptablen Wert dar, kann aber alternativ durch die Verringerung des Schwellwertes $Thx_{H,sym}$ angepasst werden.

6.5 Abschließende Betrachtungen

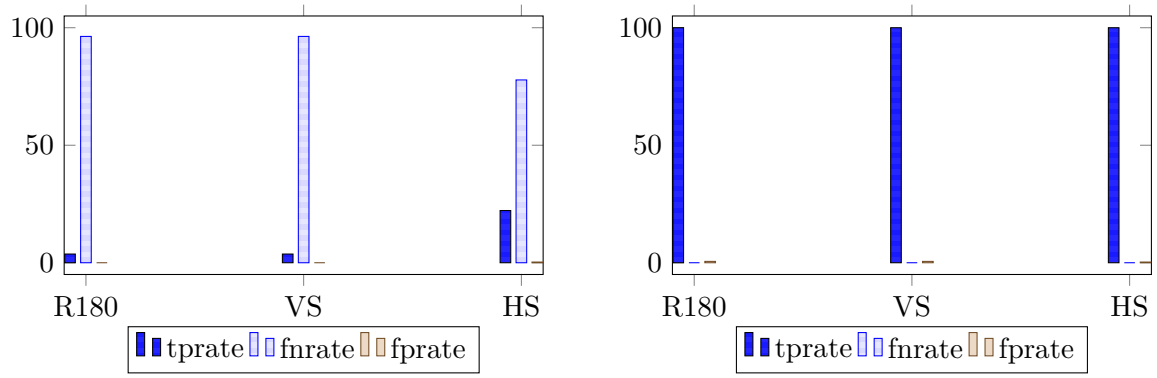
Die Messungen zeigen eine gleichbleibende Erkennungsrate der Spiegelsignatur zur Signatur nach Standard, auch wenn die symmetrische L_1-Distanz verwendet wird.



(a) MPEG-7 Signatur (b) Spiegelsignatur

Abbildung 23: Vergleich der Rotation bei Signatur nach Standard und Spiegelsignatur. Aufgetragen sind die Raten in % abhängig von der Rotation in Grad.

Die gleichbeidende Qualität der Signatur selbst wird durch die Messung der Spiegelsignatur mit dem alten Lookup gezeigt. Zusammen mit dem neuen Lookup werden aber zusätzlich sämtliche Arten von Spiegelungen zuverlässig erkannt (Abb. 24).



(a) MPEG-7 Signatur (b) Spiegelsignatur

Abbildung 24: Darstellung der Erkennungsrate bei Spiegelungen. Auf der X-Achse sind die Spiegelungen aufgetragen, auf der Y-Achse die Raten in %.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Diese Arbeit hatte zum Ziel nach Untersuchung des MPEG-7 Standard für Videosignaturen und alternativen Verfahren eine Erweiterung oder neue Signatur zu schaffen, die aufgedeckte Missstände behebt.

Dazu wurde nach einer Prüfung bestehender Verfahren der MPEG-7 Standard analysiert und in FFmpeg implementiert. Er konnte infolgedessen zusätzlich zu den Tests aus den Core-Experiments auch noch anderen Tests unterzogen werden.

Die MPEG-7 Signatur ist ein blockbasiertes Verfahren. Sie berechnet Durchschnittshelligkeitswerte von Blöcken und verknüpft diese in codierter Form zu einer Signatur auf Framebasis. Aus diesen wird in Form eines Histogramms über 90 Bildern eine weiterer Teil gebildet, der zusätzlich zu den Signaturen auf Framebasis die vollständige Videosignatur bildet. Aufgrund dieser Trennung ist eine deutliche Geschwindigkeitssteigerung beim Vergleichen zweier Signaturen zu erreichen.

Bereits in den Core-Experiments wurde der Standard auf Textüberlagerung, Helligkeitsänderungen, erneutes Abfilmen mit Rand, Recodierung, Skalierung und Bildratenänderungen getestet. In den neuen Tests kamen noch Rotation, Beschneiden des Bildrands, vertikale und horizontale Spiegelungen, Bildverzerrungen, Rauschen und Entrauschen dazu. Der Standard hat sich dabei mit über 95 % Erkennungsrate als robust gegen die meisten Modifikationen erwiesen. Stärkere Rotationen, ein zu starkes Abschneiden der Seitenränder, Textüberlagerung in massivem Ausmaß und erneutes Abfilmen führten allerdings zu schlechteren Erkennungsraten. Die blieben zumeist aber noch über 75 %. Einzig bei Spiegelungen wurden Erkennungsraten von unter 25 %, z.T. sogar nur 4 %, festgestellt. Die Ausfallrate blieb dabei bei allen Modifikationen durchgehend gering.

Die fehlende Erkennung der Spiegelung wurde zum Anlass genommen, auf Basis des Standards die Spiegelsignatur zu entwickeln, die diesen Missstand beheben sollte. Dem Verfahren lag dabei die Symmetrisierung der Signatur zugrunde, sodass gespiegelte Bilder auch noch in der Signatur als solche erkannt werden können. Im Gegensatz zu anderen Signaturen, die nach dem Standard entstanden sind und eine vollständig andere Technik aufweisen, bietet die Spiegelsignatur eine vollständige Integration in das bestehende Verfahren. So bleibt der grundlegende Aufbau der Signatur vollständig erhalten, der Speicher- bzw. Bandbreitenbedarf erhöht sich nur um 4 Byte pro Frame und im Lookup muss lediglich die L_1-Distanzbildung ersetzt werden.

In den Tests konnte eine akzeptable Erhöhung der Ausfallrate, aber auch eine leichte Verbesserung der Erkennungsraten festgestellt werden. Spiegelungen hingegen werden zuverlässigt und vollständig erkannt. Die Spiegelsignatur ist außerdem mit dem alten Lookup kompatibel (es muss nur der Parser für den Bitstrom angepasst werden) und liefert mit diesem ähnliche Raten wie die MPEG-7 Signatur. Sie kann somit als vollständiger verbesserter Ersatz zur Signatur nach dem Standard fungieren.

Ein weiterer Aspekt dieser Arbeit ist meine neu gewonnene, inzwischen gute bis sehr gute Kenntnis des FFmpeg-Filterframeworks und eine Reimplementierung des verschollenen Referenzcodes.

7.2 Ausblick

Die Spiegelsignatur sorgt bereits für eine Erkennung von Spiegelungen, kann aber noch nicht erkennen, welche Spiegelung vorgenommen wurde. Eine entsprechende Anpassung könnte im Lookup-Schritt 3 vorgenommen werden, indem die L_1-Distanz aller Elemente verglichen wird, bei einem zu niedrigen Schwellwert die vertikal gespiegelte L_1-Distanz aller Elemente verglichen wird usw.. Dies würde allerdings eine verlängerte Rechenzeit zur Folge haben.

Generell ist zu sagen, dass die Signatur nach dem MPEG-7 Standard wie auch die Modifikation anfällig gegen geometrische Transformationen ist, was an ihrem blockbasierten Charakter liegt. Da Performance ein ausschlaggebendes Argument für diese Art Signatur ist, werden noch neue Verfahren entwickelt werden müssen, die keypointbasierte Techniken mit hoher Performance vereinen, um das Problem der geometrischen Transformationen zu lösen.

Ich bin bestrebt, den von mir geschriebenen Filter dem FFmpeg-Projekt zugänglich zu machen und ihn somit allen zur Verfügung zu stellen. Dazu muss aber noch ein standardkonformer Bitstrom ausgegeben werden, da die Signatur in binärer Form bislang nur in den internen Datenstrukturen liegt und mit hexadezimalen Zeichen ausgegeben wird. Außerdem wäre eine vollständige Implementierung des partiellen und direkten Matchings wünschenswert. Weiterhin muss noch der umfangreiche Review-Prozess innerhalb des Projektes durchlaufen werden, weswegen in dieser Arbeit aus zeitlichen Gründen auch davon abgesehen wurde.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] X. Wu, A. G. Hauptmann, and C.-W. Ngo, "Practical elimination of near-duplicates from web video search." in *ACM Multimedia*, R. Lienhart, A. R. Prasad, A. Hanjalic, S. Choi, B. P. Bailey, and N. Sebe, Eds. ACM, 2007, pp. 218–227. [Online]. Available: http://dblp.uni-trier.de/db/conf/mm/mm2007.html#WuHN07a;http://doi.acm.org/10.1145/1291233.1291280;http://www.bibsonomy.org/bibtex/25712195131f43cae7beec4a3316bc14d/dblp

[2] M.-C. Yeh and K.-T. Cheng, "A compact, effective descriptor for video copy detection." in *ACM Multimedia*, W. Gao, Y. Rui, A. Hanjalic, C. Xu, E. G. Steinbach, A. El-Saddik, and M. X. Zhou, Eds. ACM, 2009, pp. 633–636. [Online]. Available: http://dblp.uni-trier.de/db/conf/mm/mm2009.html#YehC09;http://doi.acm.org/10.1145/1631272.1631375;http://www.bibsonomy.org/bibtex/229397b1815985fc3330a9daf0a276582/dblp

[3] Verschiedene, "AcoustID," 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://musicbrainz.org/doc/AcoustID

[4] S. Possos, A. Garcia, M. Mendolla, J. Schwartz, and H. Kalva, "An analysis of independence of video signatures based on tomography," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, June 2009, pp. 698–701.

[5] MPEG, "MPEG | The Moving Picture Experts Group website," 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://mpeg.chiariglione.org

[6] MPEG Video Sub-Group, "Updated Call for Proposals on Video Signature Tools," p. 1381, Oct 2008.

[7] S. Paschalakis, K. Iwamoto, P. Brasnett, N. Sprljan, R. Oami, T. Nomura, A. Yamada, and M. Bober, "The MPEG-7 Video Signature Tools for Content Identification," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 7, pp. 1050–1063, July 2012.

[8] A. Sarkar, V. Singh, P. Ghosh, B. Manjunath, and A. Singh, "Efficient and Robust Detection of Duplicate Videos in a Large Database," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 6, pp. 870–885, June 2010.

[9] M. Bertini, A. D. Bimbo, and W. Nunziati, "Video Clip Matching Using MPEG-7 Descriptors and Edit Distance." in *CIVR*, ser. Lecture Notes in Computer Science, H. Sundaram, M. R. Naphade, J. R. Smith, and Y. Rui, Eds., vol. 4071. Springer, 2006, pp. 133–142. [Online]. Available: http://dblp.uni-trier.de/db/conf/civr/civr2006.html#BertiniBN06;http://dx.doi.org/10.1007/11788034_14;http://www.bibsonomy.org/bibtex/20893546ca9d4b64dda879d70f4603380/dblp

[10] O. Cirakman, B. Gunsel, N. S. Sengor, and S. Kutluk, "Content-based copy detection by a subspace learning based video fingerprinting scheme," *Multimedia Tools and Applications*, vol. 71, no. 3, p. 1381, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11042-012-1269-8

[11] Diverse, "Non-negative matrix factorization," 2014, [Online; abgerufen am 19-September-2014]. [Online]. Available: https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

[12] D. Dutta, S. K. Saha, and B. Chanda, "An attack invariant scheme for content-based video copy detection," *Signal, Image and Video Processing*, p. 1, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11760-013-0482-x

[13] Diverse, "Scale-invariant feature transform," 2014, [Online; abgerufen am 19-September-2014]. [Online]. Available: https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[14] P. Azad, T. Asfour, and R. Dillmann, "Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition." in *IROS*. IEEE, 2009, pp. 4275–4280. [Online]. Available: http://dblp.uni-trier.de/db/conf/iros/iros2009.html#AzadAD09a;http://dx.doi.org/10.1109/IROS.2009.5354611;http://www.bibsonomy.org/bibtex/20f580f5c6120f410c5a0eb34be0b6075/dblp

[15] L. Zhu, L. Tao, and B. Shahraray, "AT&T Research at TRECVID 2009 Content-based Copy Detection," in *TRECVID 2009*, 2009.

[16] C.-Y. Chiu, C.-C. Yang, and C.-S. Chen, "Efficient and Effective Video Copy Detection Based on Spatiotemporal Analysis," in *Multimedia, 2007. ISM 2007. Ninth IEEE International Symposium on*, Dec 2007, pp. 202–209.

[17] A. Massoudi, F. Lefebvre, C. Demarty, L. Oisel, and B. Chupeau, "A Video Fingerprint Based on Visual Digest and Local Fingerprints," in *Image Processing, 2006 IEEE International Conference on*, Oct 2006, pp. 2297–2300.

[18] Diverse, "Difference of Gaussians," 2014, [Online; abgerufen am 19-September-2014]. [Online]. Available: https://en.wikipedia.org/wiki/Difference_of_Gaussians

[19] A. Joly, O. Buisson, and C. Frelicot, "Content-Based Copy Retrieval Using Distortion-Based Probabilistic Similarity Search," *Multimedia, IEEE Transactions on*, vol. 9, no. 2, pp. 293–306, Feb 2007.

[20] R. Radhakrishnan and C. Bauer, "Content-based Video Signatures based on Projections of Difference Images," in *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*, Oct 2007, pp. 341–344.

[21] X. Nie, J. Sun, Z. Xing, and X. Liu, "Video fingerprinting based on graph model," *Multimedia Tools and Applications*, vol. 69, no. 2, p. 429, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11042-012-1341-4

[22] B. Senechal, D. Pellerin, L. Besacier, I. Simand, and S. Bres, "Audio, video and audio-visual signatures for short video clip detection: experiments on Trecvid2003," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, July 2005, pp. 4 pp.–.

[23] MPEG, "MPEG Multiplies Views of Video," July 2008, 85th MPEG meeting.

[24] Google Inc., "How Content ID works," 2014, [Online; aufgerufen am 13-September-2014]. [Online]. Available: https://support.google.com/youtube/answer/2797370?hl=en

[25] Grepus, "TIL You can flip/mirror a YouTube video to circumvent the copyright infringement algo/scanner!" 2010, [Online; aufgerufen am 13-September-2014]. [Online]. Available: http://www.reddit.com/r/todayilearned/comments/hqq6k/til_you_can_flipmirror_a_youtube_video_to/

[26] D. Witt, "Copyright Match on Vimeo," 2014, [Online; aufgerufen am 13-September-2014]. [Online]. Available: http://vimeo.com/blog/post:626

[27] SoundHound Inc., "SoundHound - The Most Immersive Music Search, Discovery and Play Experience on Mobile." 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://www.soundhound.com/

[28] Shazam Entertainment Limited, "Shazam Company," 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://www.shazam.com/company

[29] IMDb.com, Inc., "IMDb App for Android Phones & Tablets," 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://www.imdb.com/apps/android/

[30] OFDb.de - Die Online-Filmdatenbank, "Erweiterte Suchmöglichkeiten und besondere Funktionen," 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://www.ofdb.de/view.php?page=erwsuche

[31] Unbekannt, "Movie & TV API Overview," 2014, [Online; aufgerufen am 14-September-2014]. [Online]. Available: http://www.themoviedb.org/documentation/api

[32] Video Subgroup, "Text of ISO/IEC 15938-6:2003/FPDAM 4," 2010.

[33] K. Iwamoto, P. Brasnett, S. Paschalakis, and M. Bober, "Study Text of ISO/IEC 15938-7:2003/PDAM 6 Conformance Testing for Video Signature Tools," July 2010.

[34] MPEG, "Visual | MPEG," 2014, [Online; aufgerufen am 23-September-2014]. [Online]. Available: http://mpeg.chiariglione.org/standards/mpeg-7/visual

[35] K. Iwamoto, T. Sato, R. Oami, and T. Nomura, "Visual duplicate based topic linking using a robust video signature," in *Consumer Electronics (ICCE), 2013 IEEE International Conference on*, Jan 2013, pp. 280–283.

[36] K. Iwamoto, P. Brasnett, S. Paschalakis, R. Oami, and M. Bober, "CE Results on Video Signature Tools (VCE7)," NEC Corporation and Mitsubishi Electric R&D Centre Europe, Oct 2009.

[37] N. Sprljan, P. Brasnett, and S. Paschalakis, "Compressed signature for video identification," in *Picture Coding Symposium (PCS), 2010*, Dec 2010, pp. 454–457.

[38] K. Iwamoto, P. Brasnett, S. Paschalakis, and M. Bober, "Study Text of ISO/IEC 15938-3:2002/FPDAM4 Video Signature Tools," Jan 2010.

[39] K. Iwamoto, P. Brasnett, S. Paschalakis, M. Bober, and R. Oami, "Study of Working Draft 15938-3:2002/Amd. 4 Video Signature Tools," July 2009.

[40] J. Garrett-Glaser, "Variable length coding and you," Sept 2008, [Online; aufgerufen am 17-September-2014]. [Online]. Available: http://x264dev.multimedia.cx/archives/category/exponential-golomb-codes

[41] Diverse, "Exponential-Golomb coding," 2014, [Online; abgerufen am 23-September-2014]. [Online]. Available: https://en.wikipedia.org/wiki/Exponential-Golomb_coding

[42] L. Li and K. Chakrabarty, "On Using Exponential-Golomb Codes and Sub-exponential Codes for System-on-a-Chip Test Data Compression." *J. Electronic Testing*, vol. 20, no. 6, pp. 667–670, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/et/et20.html#LiC04;http://dx.doi.org/10.1007/s10677-004-4254-0;http://www.bibsonomy.org/bibtex/26d0cc099505260e64b2e85059c0ebb8b/dblp

[43] Diverse, "Jaccard-Koeffizient," 2014, [Online; abgerufen am 19-September-2014]. [Online]. Available: https://de.wikipedia.org/wiki/Jaccard-Koeffizient

[44] F. Bellard, "Fabrice Bellard's Home Page," 2014, [Online; aufgerufen am 15-September-2014]. [Online]. Available: http://bellard.org/

[45] Unbekannt, "About FFmpeg," 2014, [Online; aufgerufen am 15-September-2014]. [Online]. Available: http://ffmpeg.org/about.html

[46] C. Bœsch, "A few filter questions," 2014, [E-Mail; geschrieben am 17-Juli-2014]. [Online]. Available: https://ffmpeg.org/pipermail/ffmpeg-devel/2014-July/159977.html

[47] Diverse, "Integralbild," 2014, [Online; abgerufen am 15-September-2014]. [Online]. Available: https://de.wikipedia.org/wiki/Integralbild

[48] F. Bossen, "Common test conditions and software reference configurations," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Oct 2012.

[49] Verschiedene, "Beurteilung eines Klassifikators," 2014, [Online; abgerufen am 19-September-2014]. [Online]. Available: https://de.wikipedia.org/wiki/Beurteilung_eines_Klassifikators

[50] O. Tange, "GNU Parallel - The Command-Line Power Tool," *;login: The USENIX Magazine*, vol. 36, no. 1, pp. 42–47, Feb 2011. [Online]. Available: http://www.gnu.org/s/parallel