



BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

Scalable Video Coding

Streaming Media and Development, WS 2012/13

Vorgelegt von: Sharif Hanini, Denis Oczko, Martin Maurer

Betreuer: Professor Jürgen Lohr, Professor Robert Strzebkowski

17. Januar 2013

Inhaltsverzeichnis

1. Einleitung	1
2. Aktuelle Trends und Herausforderungen	3
3. H264-SVC in a Nutshell	5
3.1. Allgemein	5
3.2. Abgrenzung zu Adaptive Streaming	7
3.3. Videokodierung mit H264/AVC	7
3.4. Videokodierung mit H264-SVC	9
3.4.1. Baselayer und Enhancementlayer	9
3.4.2. Spatiale Skalierbarkeit	10
3.4.3. Temporale Skalierbarkeit	11
3.4.4. Qualitative Skalierbarkeit	12
3.5. H264-SVC mit MPEG DASH	13
4. H264-SVC Hands-on	15
4.1. SVC-Referenzcodec	15
4.1.1. Installation	15
4.1.2. Videodateien	16
4.1.3. Enkodieren	17
4.1.4. Dekodieren	19
4.1.5. Layerstruktur anzeigen und extrahieren	20
4.1.6. Evaluation	21
4.2. Scalable Streaming Projekt	22
4.2.1. Einleitung	22
4.2.2. Projektkurzübersicht	23
4.2.3. Framework	23

4.3.	Weitere Implementierungen	26
4.3.1.	Google Talk	26
4.3.2.	MainConcept SVC	27
4.3.3.	OpenSVC Decoder	28
5.	Konklusion	29
A.	Abbildungsverzeichnis	
B.	Literatur	

1. Einleitung

In den letzten Jahren hat sich eine große Wandlung im Broadcastingumfeld ergeben. Während früher die Auslieferung von Videocontent in Standardauflösung über DVB-S und DVB-T in MPEG2 nahezu die einzige Art der Videodistribution war, ist der Distributionsprozess heute viel stärker diversifiziert und interaktiv.

Potentielle Kunden erwarten heutzutage die parallele Auslieferung des Videocontents in regulärer SD- als auch in FullHD-Auflösung. Weiterhin sind die Empfangsgeräte der Kunden sehr viel inhomogener in Bezug auf technische Anbindung als auch Dekodierfähigkeiten. Videoinhalte werden heutzutage sowohl stationär auf regulären Fernsehgeräten und modernen SmartTVs als auch mobil auf Tablets und Smartphones konsumiert. Dabei gewinnt gleichzeitig auch die Interaktivität durch zusätzliche Medieninhalten im Rahmen des hybriden Fernsehen an Bedeutung. Dadurch stehen die traditionellen Anbieter dieser Dienste vor der großen Herausforderungen, wie sie ihre Inhalte an die Kunden über verschiedene Kanäle in verschiedenen Auflösungen gleichzeitig übertragen können. Mit Scalable Video Coding (SVC) steht eine Technologie zur Verfügung, die dieses zu Leisten im Stande ist.

In dieser Ausarbeitung wird Scalable Video Coding detailliert vorgestellt. Im ersten Teil der Arbeit wird vorgestellt welche aktuellen Herausforderungen sich im Broadcastingbereich stellen. Im zweiten Teil wird die Scalable Video Coding Extension des H264/AVC-Standards vorgestellt. In diesem Abschnitt wird zuerst ein allgemeiner Ausblick auf die Notwendigkeit und Vorteile von SVC gegeben. Danach wird die Videokodierung mit H264/AVC vorgestellt und anschließend die Erweiterungen zwischen H264/AVC und H264-SVC gegenübergestellt. Im letzten Abschnitt wird die Verwendung von Scalable Video Coding im Rahmen von MPEG DASH vorgestellt.

Im vorletzten Abschnitt wird detailliert der Workflow des JSVM-Referenzcodec vorgestellt und evaluiert in wie fern sich der Codec derzeit für reale Anwendungen einsetzen lässt. Im

zweiten Teil dieses Abschnitts wird das Scalable Streaming Projekt vorgestellt und weitere SVC-Implementationen, wie der OpenSVC-Decoder vorgestellt.

Abschließend wird im letzten Abschnitt eine Zusammenfassung über den Themenbereich Scalable Video Coding gegeben.

2. Aktuelle Trends und Herausforderungen

In den vergangenen Jahren dominierten verschiedene Themen das Broadcastingumfeld. Im Jahr 2009 befand sich die Broadcastingindustrie inmitten der Vorbereitung auf die Umstellung auf HD. Daher wurde dies als wichtigster Trend der nächsten Jahre gesehen[22, S. 1]. In Deutschland begann im Jahr 2010 beispielsweise die Ausstrahlung von ARD HD im regulärem Sendebetrieb.

In den Jahren 2010 und 2011 wurde die Distribution über mehrere Plattformen, das sogenannte „Multiplatform delivery“, als wichtigster Trend gesehen[22, S. 1]. Auch im Jahr 2012 liegt dies mit weitem Abstand vor anderen Trends weiterhin vorn. Einhergehend damit wurden als weitere wichtige Trends die Optimierung des Workflows und die Übertragung via IP genannt [21, S. 1]. Die Ausrichtung auf die Bedienung mehrere Distributionskanäle und Plattformen hat mehrere Gründe.

Einerseits haben sich die Sehgewohnheiten der Konsumenten geändert. Beispielsweise sank im Jahr 2011 die Zahl an amerikanischen Haushalten, welche keinen Fernseher besaßen, zum ersten Mal in 20 Jahren[18, S. 1]. Das bedeutet nicht automatisch, dass der Fernsehkonsum rückläufig ist und durch das Internet ersetzt wird, sondern dass es zu einer Medienkonvergenz kommt. Traditionelles Fernsehen und Internet verschmelzen. Fernsehen wird über das Internet geschaut und über Fernsehgeräte bzw. SmartTVs wird im Internet gesurft und Apps werden ausgeführt [4, S. 180].

Bereits im Jahr 2011 bestand über die Hälfte des ganzen Internetverkehrs der Konsumenten aus Videodaten. Bis zum Jahr 2016 wird sich der Wert auf ca. 56% steigern. Der Anteil an Videodaten für SmartTVs lag im Jahr 2011 bei 8% und wird bis zum Jahr 2016 auf 12% steigern[2]

Die Möglichkeit Webinhalte auf dem Fernsehgerät bzw. SmartTV zu empfangen, ermöglicht einerseits neuen Konkurrenten den Einstieg in den Markt. Sogenannter Over-the-top-Content, also im Internet frei verfügbare und durch Werbung finanzierte Inhalte, wie beispielsweise Youtube oder Hulu im amerikanischen Markt, können auf lange Sicht gegenüber etablierten Anbietern im Broadcastingumfeld Marktanteile gewinnen[10, S. 2].

Andererseits bieten Fernsehgerät mit Internetzugang auch für etablierte Anbieter im Broadcastingumfeld eine neue Einnahmequelle im Rahmen von HybridTV. Zusätzlich bietet es dem Zuschauer durch Zusatzinformationen einen Mehrwert und kann sich so positiv auf die Zuschauerzufriedenheit und Marktanteile auswirken[23]

Weiterhin nimmt die Anzahl unterschiedlichen Endgeräte mit heterogenen Dekodierfähigkeiten stark zu. Während Broadcastinganbieter früher im Rahmen von Ausstrahlungen in DVB-S oder DVB-T davon ausgehen konnte, dass nahezu alle Geräte die in MPEG2 ausgestrahlten Inhalte bis zu einer bestimmten Bitrate dekodieren können, gibt es heute einerseits Geräte am unteren Ende der Skala, die nur SD-Content im H264/AVC-Baseline Profile abspielen können, wie beispielsweise ältere Android Smartphones. Andererseits gibt es wiederum Geräte, die FullHD in H264/AVC-High Profile dekodieren können, z. B. reguläre Settopboxen und SmartTVs[10, S. 3].

Ein weiterer Grund für die Forcierung auf mehrere Distributionskanäle ist der heterogene Übertragungskanal zwischen Sender und Empfänger[10, S. 4]. Die Empfänger sind sowohl mit sehr hoher Bandbreite von teilweise 100Mbit/s angebunden, als auch mit einer relativ geringen Bandbreite von unter 1Mbit/s. Durchschnittlich beträgt die Bandbreite in Deutschland derzeit beispielsweise nur 5,8 Mbit/s und wird in den nächsten Jahren nur langsam steigen[11, S. 1]. Erschwerend kommt hinzu, dass im mobilen Bereich durch die Shared-Media-Charakteristik die Bandbreite stark schwanken kann.

Um diesen sich in Zukunft weiter wachsenden Herausforderungen zu begegnen, bieten sich skalierbare Videocodecs an. Diese sollen im folgenden näher vorgestellt werden.

3. H264-SVC in a Nutshell

3.1. Allgemein

Im Bezug auf einen Videostrom versteht man unter dem Begriff „Skalierbarkeit“ die Eigenschaft, dass Teile des Videostroms entfernt werden können und der Videostrom nach Entfernen weiterhin valide und abspielbar bleibt. Dieser Teilvideostrom bildet den Originalvideostrom mit verringerter Qualität ab. Videoströme, die diese Fähigkeit nicht besitzen, nennt man Singlelayer- Videoströme [12, S. 2].

Skalierbare Videocodecs befinden sich seit nunmehr 20 Jahren in der Entwicklung. Bereits ältere Videocodecs wie das angesprochene MPEG2 boten bereits durch standardisierte Erweiterungen (z. B. MPEG2 Video) die Möglichkeiten zur Übertragung von skalierbaren Videoströmen [12, S. 1].

Allerdings wurden diese Codec-erweiterungen nur selten benutzt, da der Bedarf nach Auslieferung von skalierbaren Videoinhalten noch vergleichsweise gering war. Dies ist durch die relativ homogene technische Ausstattung der Empfänger und deren Anbindung an das Internet zu erklären. Zusätzlich verringerte sich durch den Einsatz der SVC-Erweiterungen in älteren Codecs die Effizienz der Videokodierung stark, während gleichzeitigem Rechenaufwand anstieg [13, 1f].

In den vergangenen Jahren hat in Bezug auf die technisch Anbindung und Ausstattung der Empfänger eine starke Heterogenisierung eingesetzt, sodass skalierbare Videokodierung und Distribution immer mehr in den Fokus rückt. Weiterhin hat sich mit Verabschiedung von H264/AVC im Jahr 2007 [13, S. 2] ein Codec etabliert, der sehr viel effizienter kodiert als ältere Videocodecs [14, S. 2].

Entwickelt vom JVT¹ und auf dem regulären H264/AVC-Standard aufbauend, ist H264-SVC eine Erweiterung, die als Amendment 3 in Annex G in dieses Standard aufgenommen wurde[14, S. 10]. Damit benutzt H264-SVC die effizienten Enkodierfunktionen und technischen Features des H264/AVC-Videostandards und erweitert diesen, um den zu übertragenden Videostrom skalierbar zu machen[1, S. 59]. Zusätzlich bleibt dadurch die Rückwärtskompatibilität zu herkömmlichen H264/AVC-Dekodern gewahrt. [10, S. 601]

H264-SVC bietet im Broadcastingumfeld einige Vorteile gegenüber anderen derzeit eingesetzten Verfahren, wie beispielsweise Simulcast, bei welchem jeder Videostrom unabhängig voneinander kodiert und übertragen wird. [13, S. 1] Auf H264/AVC aufbauend, ist H264-SVC vollständig abwärtskompatibel zu herkömmlichen H264-Dekodern.

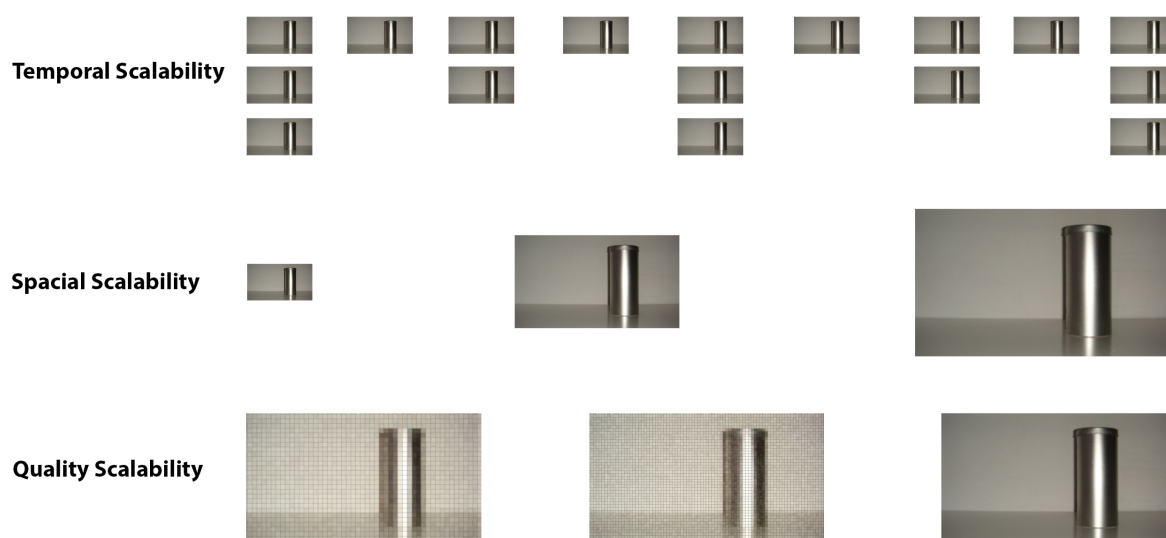


Abbildung 3.1.: Arten der Skalierbarkeit (Eigene Abbildung nach [7, S. 143])

Diese ignorieren die SVC-Erweiterung und dekodieren nur die regulären H264/AVC-Inhalt. Dies ist möglich indem neben einem regulärem H264/AVC-Videostrom, dem sogenannten Baselayer, weitere Videoströme, den sogenannten Enhancementlayers, eingebettet werden. Diese Enhancementlayers können sich gegenüber dem Baselayer in Bezug auf Framerate (Temporal), Auflösung (Spatial) und Bildqualität (Quality) unterscheiden (Siehe Abbildung 3.1) [10, S. 601]. In skalierbaren Videoströmen können diese drei Arten von Enhance-

¹JVT - Joint Video Team

mentlayern auch beliebig kombiniert werden, z. B. Spatiotemporale Enhancementlayer [12, S. 2].

Zusätzlich können mit einer in H264-SVC enkodierten Videoquelldatei gleichzeitig unterschiedliche Geräte z. B. Smartphones und FullHD-Geräte angesprochen werden. Die sonst übliche Transkodierung und das Vorhaltung vorab enkodierter Teilvideodateien entfällt. Das erleichtert die Langzeitarchivierung und verringert gleichzeitig den benötigten Speicherplatz.

Bei Verwendung dieser H264-SVC erhöht sich der Overhead nur um 10% bis 20% verglichen mit regulärem H264/AVC. Im Falle eines stark fehleranfälligen Übertragungsmediums kann der Overhead sogar geringer ausfallen, als bei Simulcastübertragung. Zusätzlich bleibt der zur Enkodierung und Dekodierung benötigte Rechenaufwand nahezu gleich [14, S. 4].

3.2. Abgrenzung zu Adaptive Streaming

H264-SVC und Adaptive Streaming unterscheiden sich in der Art der Speicherung der Videodaten. Bei der Verwendung von Adaptive Streaming werden Videodaten in einzelne Segmente variabler Länge unterteilt und getrennt gespeichert. Bei der Verwendung von H264-SVC werden die Videodaten in einzelne Layer kodiert in einer Datei gespeichert. Je nach Implementation werden bei der Auslieferung des Videocontents die Layer on the Fly extrahiert und übertragen, oder vorab extrahiert.

3.3. Videokodierung mit H264/AVC

Als Erweiterung des H264/AVC-Standards benutzt H264-SVC den gleichen Enkodierablauf und erweitert diesen in wenigen Bereichen, um Skalierbarkeit zu ermöglichen. Die Videokodierung mit H264/AVC erfolgt in vier Teilschritten. Zuerst wird der aktuelle Videoframe (F) in mehrere Slices unterteilt, die wiederum Makroblöcke enthalten [5, S. 146]. In jedem Makroblock sind die Luminanz und Chrominanzwerte gespeichert. Aufgrund des Chromasubsamplings liegen die Luminanzwerte in den meisten Fällen nur mit halbiertter Auflösung vor.

Die Größe der Makroblöcke ist auf 16x 16 Pixel festgelegt. Makroblöcke können wiederum in verschiedene Makroblockpartitionen unterteilt werden z.B 8 x 8 Pixel Blöcke. Diese Makroblockpartitionen können selbst wieder in Sub-Makroblockpartitionen von beispielsweise 4 x 4 Pixel unterteilt werden. Diese so entstehende Unterteilung wird als *Tree Structured Motion Compensation* bezeichnet [7, S. 160].

Neben der Makroblockgröße unterscheidet man auch den Typ eines Makroblocks und damit die Art der Slices. I-Makroblöcke werden von vorherigen Makroblöcken des selben Slices kodiert und dienen anderen Makroblöcke als Referenz. P-Makroblöcke referenzieren auf vorhergegangene I-Makroblöcke von vorherigen Frames und ermöglichen dadurch eine Steigerung der Kompressionrate. B-Makroblöcke referenzieren sowohl auf vorhergegangene als auch auf nachfolgende I-Makroblöcke dieser Frames und ermöglichen so eine weitere Steigerung der Kompression [7, 164f].

Analog zu den Makroblöcken werden auch die Slices nach I, P und B-Slice unterschieden. I-Slices enthalten nur I-Makroblöcke, während P-Slices sowohl I-Makroblöcke als auch P-Makroblöcke enthalten können. B-Slices können I-Makroblöcke und B-Makroblöcke enthalten [7, 159f].

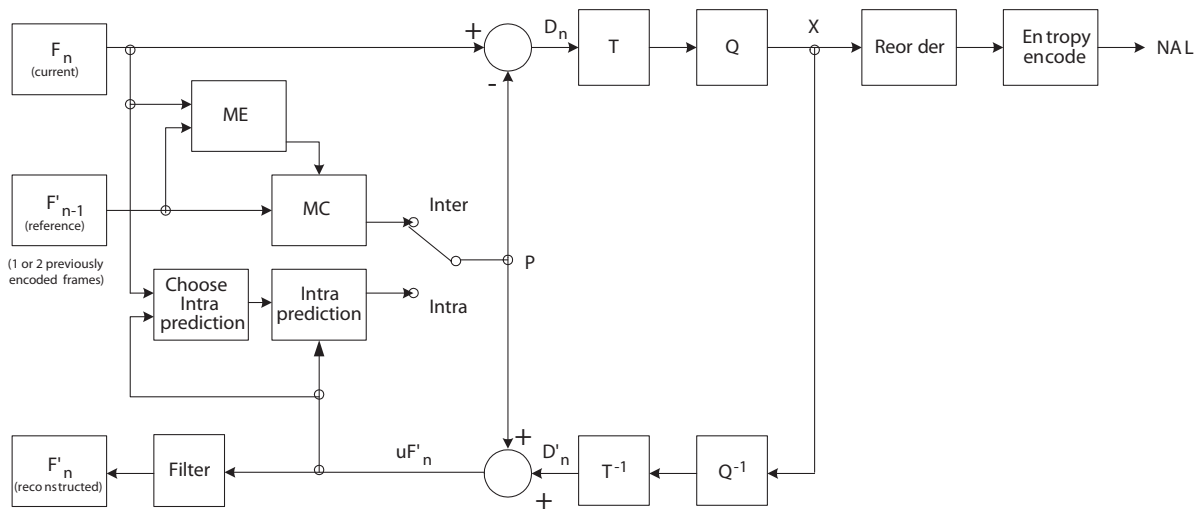


Abbildung 3.2.: Funktionsweise eines H264/AVC-Encoders [7]

Bei der Kodierung eines Frames (F) wird für jeden Makroblock eine Voraussage getroffen. Diese kann eine intra-basierte Voraussage sein, die auf der Kodierung von vorangegangenen Makroblöcken des gleichen Slice beruht. Alternativ kann es auch eine intra-

basierte Voraussage sein, die auf der Referenzierung und Kodierung von Makroblöcken aus vorausgegangenen- und nachfolgenden Frames beruht. In beiden Fällen wird die Differenz zwischen dem aktuellen Makroblock und der Voraussage gebildet und die Abweichungen zwischen beiden Makroblöcken gespeichert[7, 161f].

Im nächsten Schritt werden diese Abweichungen quantisiert und die Koeffizienten in einem Zickzack-Verfahren angeordnet[7, 198f]. Dieses Ausleseverfahren ist für die anschliessende Entropiekodierung besser geeignet, da so ähnliche Koeffizienten nacheinander stehen und diese somit einfacher zusammenfassbar sind [5, S. 30]. Bei H264/AVC kommen mehrere Verfahren zur Entropiekodierung zum Einsatz. Neben Variable-Length Codes (VLC) auch Context-Adaptive Arithmetic Coding (CABAC), Context-Adaptive Variable Coding (CAVLC) und Exp-Golomb [7, 198f].

Im letzten Schritt werden die entropiekodierten Koeffizienten zu einem Videostrom zusammengefasst und durch den Network Abstraction Layer (NAL) übertragen[7, 161f]

3.4. Videokodierung mit H264-SVC

3.4.1. Baselayer und Enhancementlayer

Zur Erzeugung von skalierbaren Videostreamen werden H264-SVC Videostreame in zwei Arten von Layer unterteilt. Einereits gibt es den sogenannten Baselayer, welcher in jedem H264-SVC Videostrom vorhanden sein muß. Zusätzlich gibt es auf diesen aufbauend mehrere Enhancementlayer, die Videostreame mit unterschiedlichen Qualitätsstufen (Auflösung, Framerate und Qualität) enthalten.[6, S. 1]

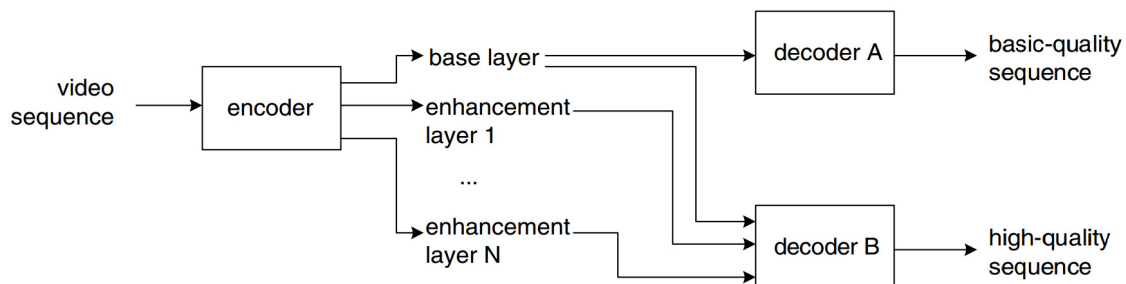


Abbildung 3.3.: Base- und Enhancementlayer [7, S. 142]

Der Baselayer wird im regulärem H264/AVC-Modus enkodiert. Die Kodierung erfolgt mit festgelegter Group of Pictures bzw. GOP und freier Refrenzierung der einzelnen Frames innerhalb jeder GOP. Dadurch kann der Baselayer auch von regulären, nicht H264-SVC-fähigen H264/AVC Videodecodern dekodiert- und angezeigt werden. Bei diesen Geräten werden die Enhancementlayer ignoriert. [7, S. 144]

H264-SVC-fähige Geräte dekodieren sowohl den Baselayer als auch den Enhancementlayer. Die einzelnen Enhancementlayer erhöhen mit jeder Iteration des Enhancementlayers die Qualität in Bezug auf Auflösung, Bitrate oder Framerate. Bei der Dekodierung wird aus dem im Enhancementlayer gespeicherten Deltawerten bzw Differenzen zum Baselayer und dem Baselayer selber das Videobild rekonstruiert [7, S. 144].

3.4.2. Spatiale Skalierbarkeit

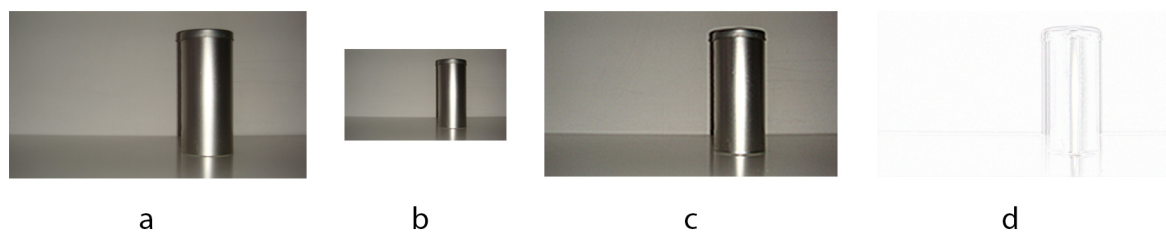


Abbildung 3.4.: Enkodierung von Baselayer und Enhancementlayer
(Eigene Abbildung nach [7, S. 143])

Die Videoenkodierung der Spatialen- bzw. Auflösungs-Enhancementlayer erfolgt, indem zuerst für jedes Frame des Videoeingangssignal (Siehe Abbildung 3.4 a) der Baselayer (Siehe Abbildung 3.4 b) mit verringerter spatialer Qualität in H264/AVC enkodiert wird.

Um die Differenzen des Baselayers zum Originalvideobild zu erhalten, wird der zuvor erzeugte Baselayer auf die ursprüngliche Größe skaliert (Siehe Abbildung 3.4 c) und die Differenz beider Bilder extrahiert (Siehe Abbildung 3.4 d). Aus diesem Differenzbild wird der erste Enhancementlayer enkodiert. Jeder weitere Enhancementlayer enkodiert die Differenz zum vorherigen, hochskalierten Enhancementlayer. [7, 142ff]

Die Dekodierung der mit H264-SVC enkodierten Videoströmen erfolgt, indem zuerst der Baselayer dekodiert wird (Siehe Abbildung 3.5 a) und auf die ursprüngliche Auflösung

hochskaliert wird (Siehe Abbildung 3.5 b). Anschließend werden die Enhancementlayer dekodiert (Siehe Abbildung 3.5 c) und deren Bildinformationen zum bereits dekodierten Baselayer hinzugefügt (Siehe Abbildung 3.5 d), um das vollständigen Videobild wiederherzustellen.[7, S. 144]



Abbildung 3.5.: Dekodierung von spatialen Enhancementlayern
(Eigene Abbildung nach [7, S. 143])

3.4.3. Temporale Skalierbarkeit

Zur Erzeugung verschiedenerer temporaler- bzw. zeitlicher- Enhancementlayer verwendet H264-SVC, die regulären in H264/AVC angewandten hierarchischen B- und P Frames[19, S. 3]. Zur Erzeugung der temporalen Enhancementlayer werden die einzelnen Frames der in höchster Framerate vorliegende Videodatei auf die einzelnen Layer aufgeteilt. Die Länge der GOP ist dabei relativ frei wählbar. Der Baselayer enthält die Qualitätsstufe mit der geringsten Framerate. In diesem wird nur ein geringer Teil der GOP gespeichert, beispielsweise I-Frame des ersten und letzten Frames der GOP und ein weiterer B-Frame.[7, S. 144]

Die Enhancementlayer kodieren sukzessive jeweils höhere Frameraten, indem sie auf die Frames der vorhergehenden Layer referenzieren. Im ersten Enhancementlayer werden weitere B-Frames des Originalvideodatei gespeichert. Diese B-Frames referenzieren die I-Frames des Baselayers. Im nachfolgendem Enhancementlayer werden wiederum weitere B-Frames des Originalvideodatei gespeichert. Damit referenziert dieser nun den Baselayer und den ersten Enhancementlayer.[8, S. 292].

Dieses Prinzip setzt sich bis zum letzten Baselayer fort. Bei der Dekodierung muß man daher alle Base- und Enhancementlayer dekodieren, die dem gewünschten Enhancementlayer vorangehen. [8, S. 292].

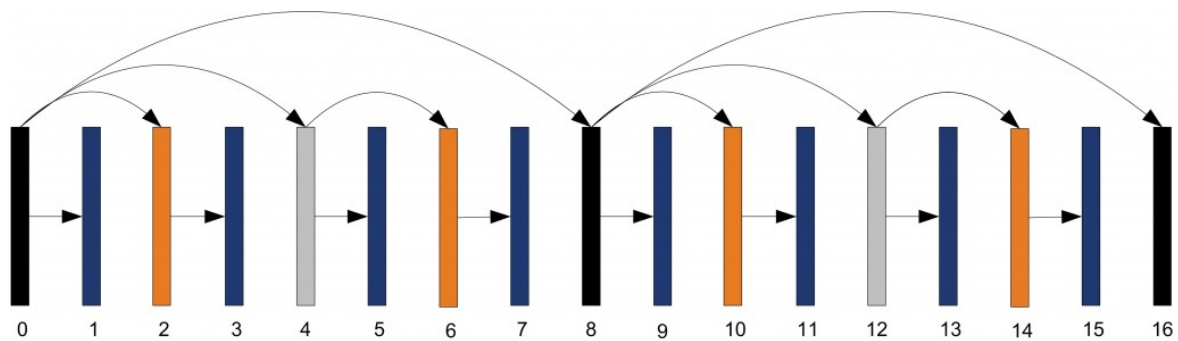


Abbildung 3.6.: Temporale Skalierbarkeit [15]

3.4.4. Qualitative Skalierbarkeit

H264-SVC enkodierte Videoströme mit qualitativer Skalierbarkeit besitzen mehrere Layer mit unterschiedlicher Bildqualitätsstufen. Diese Art der Skalierbarkeit wird auch als SNR-Skalierbarkeit bezeichnet. Im Detail unterstützt H264-SVC drei unterschiedliche Implementierung für qualitative Skalierbarkeit. [6, S. 10]

Die Basisimplementation ist das sogenannte Coarse-grain Quality Scalable Coding (CGS). Dies kann als Sonderfall der spatialen Skalierbarkeit angesehen werden. Zuerst wird aus dem Videoeingangssignal der Baselayer mit geringer Qualität (z. B. Bitrate) generiert. Anschließend werden in nachfolgenden Enhancementlayers die Differenz zwischen dem I-Frame des Originalvideos und dem I-Frame des vorherigem Layers mit festen Bitraten kodiert [19, S. 4]. Innhalb der GOP werden nur die Referenzbilder im eigenen Layer kodiert. Referenzierungen zwischen den einzelnen Layern findet nichts statt.[12, 8f]. Skalierbarkeit wird in diesem Fall durch das komplette Entfernen einzelner Enhancementlayer erreicht, wodurch allerdings die Flexibilität und Effizienz eingeschränkt ist. [9, S. 2] Die Anzahl der Layer ist auf bestimmte Bitraten beschränkt und der Wechslen zwischen verschiedenen Layern ist nur an an den GOP-Grenzen bzw. I-Frames möglich [13, S. 7].

Eine verbesserte Implementation ist das sogenannte Medium Grain Scalability (MGS), bei welchem innerhalb der GOP des Baselayers auch auf die Referenzframes des nachfolgendem Enhancementlayers zugegriffen werden kann. Dies erlaubt eine flexiblere Skalierbarkeit, erhöht aber die Fehleranfälligkeit. [6, S. 8]

Die dritte Implementation ist die sogenannte Fine Grain Scalability (FGS). Diese Art der Skalierung baut auf CGS auf und verfeinert diese Methode. Die Bitrate der einzelnen Layer

wird dynamisch an die aktuell verfügbare Bandbreite angepasst. Referenzierung einzelner Frames innerhalb der GOP wird nur im Baselayer durchgeführt. [6, S. 8]

3.5. H264-SVC mit MPEG DASH

MPEG DASH (Dynamic Adaptive Streaming over HTTP) definiert ein HTTP-basierendes Streamingverfahren. [3, S. 1]. Durch die Verwendung von HTTP gegenüber anderen Protokollen wie RTP ergeben sich mehrere Vorteile. In vielen Einsatzbereichen (z. B. Unternehmen) sind aus Sicherheitsgründen nur absolut notwendige Ports geöffnet, unter anderem HTTP auf Port 80. Zusätzlich entfällt die Auslieferung der Streaminginhalte über einen dedizierten Streamingserver. Stattdessen können reguläre Webserver zur Auslieferung benutzt werden. Dadurch sinken für Streaminganbieter die Kosten für die Auslieferung des Contents [17, S. 3].

Zur Auslieferung des Videocontents spezifiziert MPEG DASH ein XML-Dokument, das sogenannte Media Presentation Document (MPD), welches die einzelnen Segmente der Videodatei definiert und referenziert. Auf diese einzelnen Segmente kann per HTTP-GET zugegriffen werden [3, S. 1].

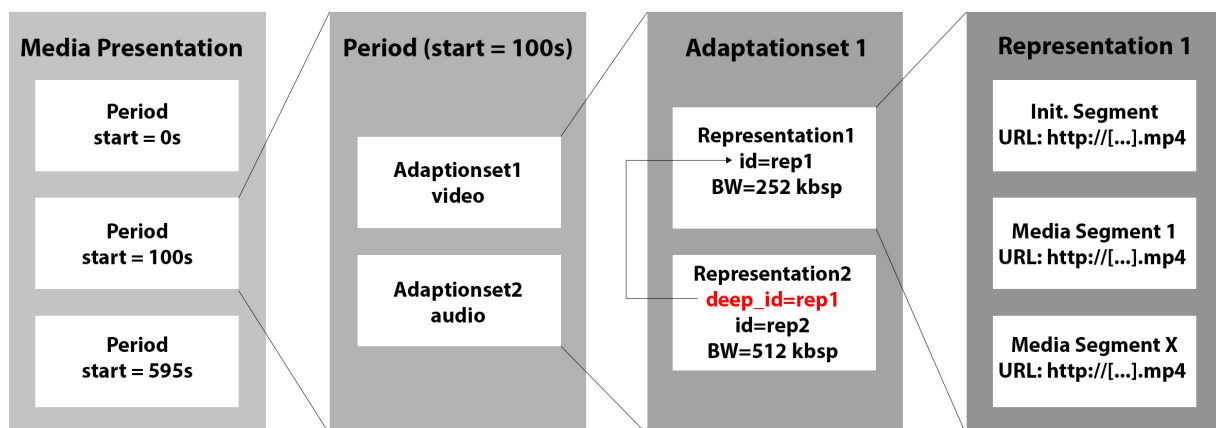


Abbildung 3.7.: Media Presentation Document und H264-SVC
(Eigene Abbildung nach [3])

Jedes in der MPD definierte Segment besteht aus einer Abfolge von Zeiteinheiten bzw. Periods, deren Intervall beliebig gewählt werden kann. In meisten Fällen beträgt dieses Intervall 2 bis 10 Sekunden. Jede Zeiteinheit besteht aus einem oder mehreren Adaptationsets z. B. jeweils für die jeweiligen Audiodaten, Videodaten oder Untertitel. Jedes Adaptionset besteht wiederum aus verschiedenen Representationen für unterschiedliche Endgeräte (z. B. Unterscheidung via Bitrate oder Auflösung). In den einzelnen Representationen werden jeweils wiederum einzelne Segmente gespeichert, die durch eine eigene URL per HTTP-GET aufrufbar sind [3, S. 1].

Durch diese modulare Struktur eignet sich MPEG DASH sehr gut zur Auslieferung von H264-SVC encodierte Videoströmen. Die einzelnen Baselayer und Enhancementlayer können jeweils in eigene Representationen gespeichert werden und innerhalb der Adaptationsets referenziert werden. [17, S. 14].

Die Zuordnung der einzelnen Layer erfolgt dabei über zusätzliche Attribute in der MPD-Datei. Jede Representation, die ein Layer definiert, erhält ein ID-Attribut und ein DependencyId -Attribut. Dieses gibt an auf welche vorherige Layer, sich dieser Layer bezieht und welche zuvor dekodiert werden müssen[17, S. 16].

4. H264-SVC Hands-on

4.1. SVC-Referenzcodec

Zur Evaluation der H264-SVC-Erweiterung wird der *JSVM-Codec* (Joint Scalable Video Model) verwendet. Dieser ist der Referenzcodec des Joint Video Team (JVT), der ISO/IEC Moving Pictures Experts Group (MPEG) und der ITU-T Video Coding Experts Group (VCEG). Neben dem Encoder und Decoder sind im JMVC-Softwarepakets auch Multiplexer und Testprogramme enthalten.

Das JMVC-Softwarepaket ist im Quelltext frei verfügbar. Der Quelltext für das Softwarepaket ist in C++ geschrieben und auf dem Webservice der Rheinisch-Westfälische Technische Hochschule Aachen (RWTH-Aachen) in einem CVS-Repository gehostet. Um dieses Abzurufen ist eine geeigneten CVS-Software wie Tortoise notwendig

Die hier verwendete Version des Encoder ist die Version 9.19.14.

Authentifizierungstyp	pserver
Hostadresse	garcon.ient.rwth-aachen.de
CVS-Pfad	/cvs/jvt
Benutzername	jvtuser
Passwort	jvt.Amd.2
Modulname	jsvm oder jsvm _{red}

Tabelle 4.1.: Zugangsdaten für das CVS-Repository der RWTH Aachen

4.1.1. Installation

Nach dem lokalen Download aus dem CVS der RWTH-Aachen befindet sich der Quellcode mit allen benötigten Klassen und Bibliotheken im Unterordner *jsvm/h264Extension/build/windows*.

Zur Vereinfachung der Installation werden bereits vorkonfigurierte Projektmappen für Microsoft Visual Studio mitgeliefert. Nach Öffnen der Projektmappe in Visual Studio wird mit der Menüoption *Create Batch/ Projektmappe erstellen* der Quellcode kompiliert. Nach erfolgreichem Kompilieren sind alle ausführbaren Programme im Unterordner *jmvc/bin* zu finden.

4.1.2. Videodateien

Der JSVM-Codec unterstützt als Eingabeformat nur unkomprimiertes YUV-Video. Zusätzlich müssen diese für die jeweiligen Layer vorab in der gewünschten Auflösung der Layer kodiert werden. Die im Test verwendeten Beispieldateien befinden sich auf dem Webservice der TU-München und können unter folgender URL heruntergeladen werden <http://nsl.cs.sfu.ca/video/library/tu-muenchen.de/>. Weitere Testvideo können unter folgender URL heruntergeladen werden <ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/>.

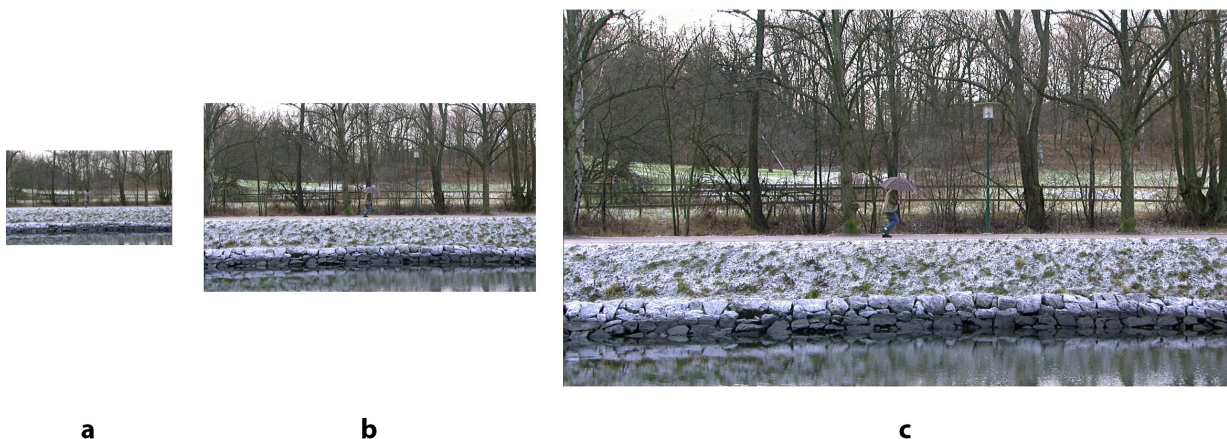


Abbildung 4.1.: Das Quellvideo „Park“ in 180p (a), 360p (b) und 720p (c) Auflösung

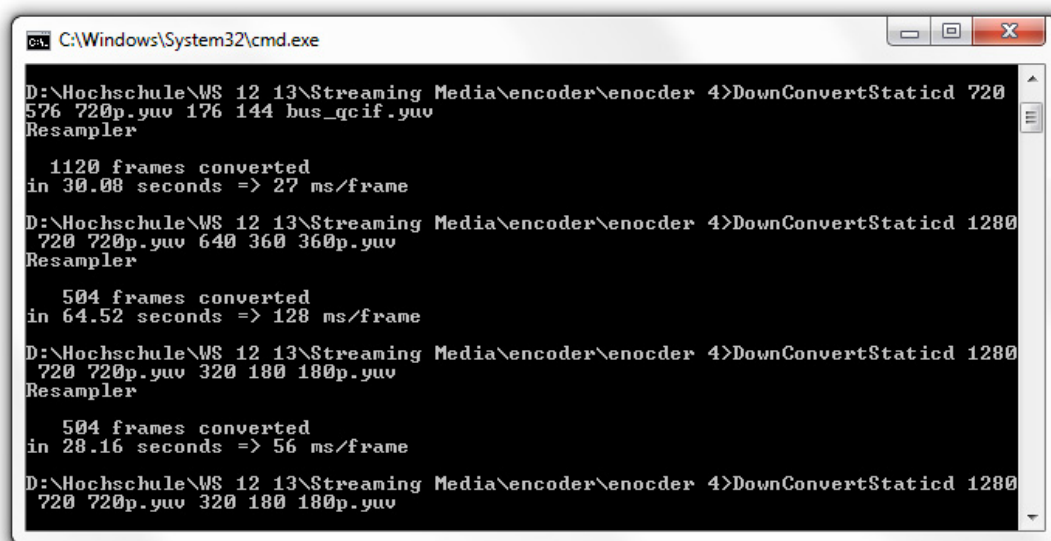
Der Referenzencoder benötigt für die Erstellung eines beliebigen skalierbaren Videostroms mehrere Versionen eines Quellvideos, das sich in Bezug auf Bildgröße, Framerate oder Bildqualität unterscheidet. Da diese unterschiedlichen Versionen meist nicht vorliegen, besitzt der Referenzencoder bereits ein Unterprogramm zur Erstellung dieser Videodateien. Das

Programm heißt „DownConvertStatic“ und wird mit folgender Syntax über die Kommandozeile aufgerufen:

```
DownConvertStaticd HAuflösung VAuflösung Input.yuv HZielAuflösung  
VeZielAuflösung Output.yuv
```

Listing 4.1: Aufruf DownConvertStatic

Anschließend befinden neben der ursprünglichen YUV-Datei weitere Videodateien mit unterschiedlicher spatialer Qualität im aufrufenden Verzeichniss.



```
C:\Windows\System32\cmd.exe  
D:\Hochschule\WS 12 13\Streaming Media\encoder\enocder 4>DownConvertStaticd 720  
576 720p.yuv 176 144 bus_qcif.yuv  
Resampler  
1120 frames converted  
in 30.08 seconds => 27 ms/frame  
D:\Hochschule\WS 12 13\Streaming Media\encoder\enocder 4>DownConvertStaticd 1280  
720 720p.yuv 640 360 360p.yuv  
Resampler  
504 frames converted  
in 64.52 seconds => 128 ms/frame  
D:\Hochschule\WS 12 13\Streaming Media\encoder\enocder 4>DownConvertStaticd 1280  
720 720p.yuv 320 180 180p.yuv  
Resampler  
504 frames converted  
in 28.16 seconds => 56 ms/frame  
D:\Hochschule\WS 12 13\Streaming Media\encoder\enocder 4>DownConvertStaticd 1280  
720 720p.yuv 320 180 180p.yuv
```

Abbildung 4.2.: Erzeugung von 360p und 180p YUV-Videodateien

4.1.3. Enkodieren

Der JSVM-Codex unterstützt zwei Betriebsarten. Einerseits die Funktionsweise als Single-layer Encoder, bei welchem er sich wie ein regulärer H264/AVC-Encoder verhält und nur H264/AVC-Videodateien erstellt. Andererseits unterstützt er auch die Erzeugung von skalierbaren Videoströmen in H264-SVC.

Zur Erzeugung eines skalierbaren Videostroms in H264-SVC werden mehrere Konfigurationsdateien benötigt. Einerseits wird eine allgemeine Konfigurationsdatei benötigt, in welcher die Videoinformationen der Quelldatei definiert werden. Dazu zählen u.a. der Dateiname, die Auflösung, die Anzahl der zu enkodierenden Frames und die Größe der Group

of Pictures. Zusätzlich werden die layerspezifischen Konfigurationsdateien in dieser Konfigurationsdatei verlinkt.

```

OutputFile          test.264    # Bitstream file
FrameRate           30.0        # Maximum frame rate [Hz]
FramesToBeEncoded   150         # Number of frames
GOPSize             16          # GOP Size (at maximum frame rate)
BaseLayerMode       2           # Base layer mode
SearchMode          4           # Search mode
SearchRange         32          # Search range (Full Pel)
NumLayers           1           # Number of layers
LayerCfg            layer0.cfg  # Layer configuration file

```

Listing 4.2: Konfigurationsdatei des Encoders

Weiterhin wird für jeden Layer eine zusätzliche Konfigurationsdatei benötigt. In dieser spezifischen Datei werden unter anderem der Dateiname der herunterskalierten Quelldatei, die Auflösung, Bitrate und die Framerate für diesen Layer definiert.

```

InputFile           BUS_CIF30.yuv # Input file
SourceWidth         352            # Input frame width
SourceHeight        288            # Input frame height
FrameRateIn         30             # Input frame rate [Hz]
FrameRateOut        30             # Output frame rate [Hz]
InterLayerPred      2              # Inter-layer Pred.

```

Listing 4.3: Konfigurationsdatei für einen Layer

Der Encoder `H264AVCEncoderLibTestStatic` wird über folgenden Befehl über die Kommandozeile aufgerufen:

```
H264AVCEncoderLibTestStaticd pf Konfigurationsdatei.cfg
```

Listing 4.4: Aufruf `H264AVCEncoderLibTestStatic`

Danach analysiert der Encoder die Konfigurationsdatei und enkodiert die einzelnen Layer. Der Baselayer wird regulär in H264/AVC enkodiert, während die Enhancementlayer in

H264-SVC enkodiert werden. Wenn die Enkodierung erfolgreich abgeschlossen ist, gibt er Encoder anschließend eine Übersicht über die enkodierten und eingebetteten Layer aus. In der Übersicht wird sowohl die spatialen Layer (Auflösung), die Framerate (temporale Auflösung) angezeigt mit Übersicht der einzelnen Bitraten und PSNR-Werten. Die entsprechende enkodierte Zielfeile befindet sich im Verzeichniss des Encoders unde trägt die Dateiendung .h264.

```

C:\Windows\System32\cmd.exe
98: P   T3 L2 Q0  QP 31  Y 28.8593  U 37.2803  U 39.5812  1257064 bit
AU 97: B   T4 L2 Q0  QP 32  Y 30.6206  U 39.7964  U 40.6233  104448 bit
AU 99: P   T4 L2 Q0  QP 32  Y 28.4747  U 39.0205  U 39.9470  173848 bit

SUMMARY :

      bitrate      Min-bitr      Y-PSNR      U-PSNR      V-PSNR
-----
320x192 @ 3.1250    115.4000    115.4000    37.7698    42.4585    45.8742
320x192 @ 6.2500    128.7538    128.7538    36.0342    41.6999    45.5373
320x192 @ 12.5000   137.6400    137.6400    35.5803    41.7155    45.5521
640x368 @ 3.1250   1710.7536   1710.7536   37.9473    41.6142    44.0565
640x368 @ 6.2500   2834.4538   2834.4538   34.6204    40.5881    43.4624
640x368 @ 12.5000  4264.9080   4264.9080   32.0474    40.0021    43.1258
640x368 @ 25.0000  4348.8360   4348.8360   32.4314    40.5190    43.1827
1280x720 @ 3.1250   9767.6929   9767.6929   38.2945    40.6886    42.3544
1280x720 @ 6.2500  16433.6115  16433.6115   35.1566    38.9723    40.8175
1280x720 @ 12.5000 27150.8320  27150.8320   33.0303    38.0651    40.0083
1280x720 @ 25.0000 42926.5080  42926.5080   31.5705    37.6268    39.7260
1280x720 @ 50.0000 45689.2160  45689.2160   31.0762    38.0571    39.8458

Encoding speed: 181741.020 ms/frame, Time:18174102.000 ms, Frames: 100
D:\Hochschule\WS 12 13\Streaming Media\encoder\encoder 4>

```

Abbildung 4.3.: Spatale und Temporale Enkodierung

4.1.4. Dekodieren

Der Referenzcodec unterstützt nur die Dekodierung des H264-SVC Videostroms zu YUV. Dabei wird aus allen eingebetteten Layern das Ausgangsvideo rekonstruiert und als unkomprimiertes YUV-Video angelegt. Die Dekodierung erfolgt mit dem Programm H264AVCDecoderLibTestStaticd mit folgendem Aufruf auf der Kommandozeile.

H264AVCDecoderLibTestStaticd InputDatei.h264 OutputDatei.yuv

Listing 4.5: Aufruf H264AVCDecoderLibTestStaticd

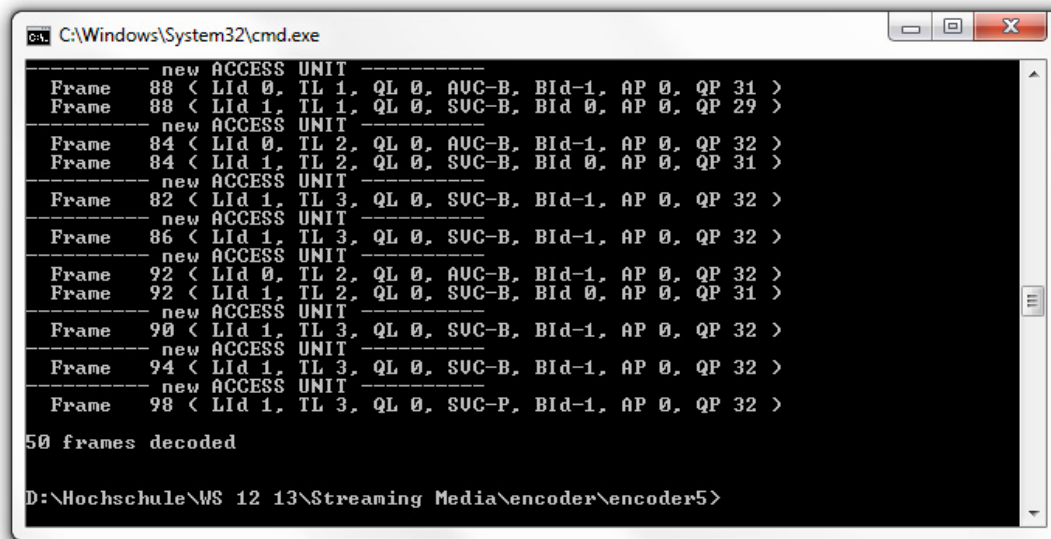


Abbildung 4.4.: Dekodierung und Rekonstruktion des Videostroms

4.1.5. Layerstruktur anzeigen und extrahieren

Der JSVM-Codec bietet mit Hilfe des Programms `BitStreamExtractorStaticd` die Möglichkeit Layer zu extrahieren und daraus eine neue H264-SVC-Datei zu erstellen, die ausschließlich die extrahierten Layern beinhaltet. Zusätzlich bietet dieses Tool die Möglichkeit die Layerstruktur zu Debugzwecken anzuzeigen. Der Aufruf erfolgt auf der Kommandozeile folgendermaßen:

```
BitStreamExtractorStaticd InputDatei.svc
```

Listing 4.6: Aufruf `BitStreamExtractorStaticd`

Anschließend erhält man eine Übersicht über alle enthaltenen Layer und deren Enkodierdetails. Zur Extraktion mehrerer Layer, z. B. in diesem Fall Layer 0 bis 5 ruft man das gleiche Programm mit folgender Syntax von der Kommandozeile auf:

```
BitStreamExtractorStaticd InputDatei.svc OutputDatei.svc sl 5
```

Listing 4.7: Aufruf `BitStreamExtractorStaticd`

Als Resultat erhält man eine neue valide H264-SVC-kodierte Videodatei, die nur die Layer 0 bis 5 enthält.

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

D:\Hochschule\WS 12 13\Streaming Media\encoder\encoder6>BitStreamExtractorStatic
d 2Layer.264
JSUM 9.19.14 BitStream Extractor

Contained Layers:
=====
      Layer  Resolution  Framerate  Bitrate  MinBitrate  DTQ
      ----  -
      0      320x192      3.1250    115.40    115.40    <0.0,0>
      1      320x192      6.2500    128.80    128.80    <0.1,0>
      2      320x192     12.5000    137.60    137.60    <0.2,0>
      3      640x368      3.1250   1711.00   1711.00    <1.0,0>
      4      640x368      6.2500   2834.00   2834.00    <1.1,0>
      5      640x368     12.5000   4265.00   4265.00    <1.2,0>
      6      640x368     25.0000   4349.00   4349.00    <1.3,0>

D:\Hochschule\WS 12 13\Streaming Media\encoder\encoder6>

```

Abbildung 4.5.: Anzeige der Layerstruktur mit Hilfe von BitStreamExtractorStatic

4.1.6. Evaluation

Der JSVM-Referenzcode ist ein sehr vielseitiges Tool und derzeit die einzige frei verfügbare Anwendung, um H264-SVC Videodateien zu erzeugen. Die Installation ist durch ein vorkonfiguriertes Visualstudioprojekt sehr einfach und funktioniert problemlos.

Die Arbeit mit dem Codec ist über die Kommandozeile und Eingabeparameter nicht sehr komfortable, vereinfacht aber dafür die Einbindung in andere Programme. Die mitgelieferte Dokumentation und Beispielprojekte machen es relativ einfach möglich selber mit dem Codec zu experimentieren. Allerdings gibt der Codec selbst nur wenig Hinweis und Fehlermeldungen, warum manche Settings nicht funktionieren und zum Ablauf des Enkodierdurchlaufs führen.

Das große Problem des JSVM-Codex sind die Geschwindigkeit der Enkodierung und die möglichen Ausgangsformate. Diese beiden Aspekte machen jegliche Verwendung in Realanwendungen derzeit unmöglich. Der Codec läuft nur auf einem Prozessorkern und nutzt dadurch die Möglichkeiten moderner PCs nicht. Dadurch ist die Enkodierung sehr langsam und zeitraubend.

Bei regulärem CIF und QCIF Videos betrug die Geschwindigkeit ca. 5 bis 6 Sekunden pro Frame. Bei Testreihen mit HD-Material dauerte die Enkodierung von 100 Frames mehr als


```

C:\Windows\System32\cmd.exe
5      640x368      12.5000      4265.00      4265.00      <1.2.0>
6      640x368      25.0000      4349.00      4349.00      <1.3.0>
Total Packet Size: 1066624

Number of input packets : 105
Number of output packets: 80

D:\Hochschule\WS 12 13\Streaming Media\encoder\encoder6>BitStreamExtractorStatic
d out.svc
JSUM 9.19.14 BitStream Extractor

Contained Layers:
=====
      Layer  Resolution  Framerate  Bitrate  MinBitrate  DTQ
      ---  -
      0      320x192      3.1250      115.40      115.40      <0.0.0>
      1      320x192      6.2500      128.80      128.80      <0.1.0>
      2      320x192      12.5000      137.60      137.60      <0.2.0>
      3      640x368      3.1250      1711.00      1711.00      <1.0.0>
      4      640x368      6.2500      2834.00      2834.00      <1.1.0>
      5      640x368      12.5000      4265.00      4265.00      <1.2.0>

D:\Hochschule\WS 12 13\Streaming Media\encoder\encoder6>

```

Abbildung 4.6.: Anzeige der geänderten Layerstruktur mit Hilfe von BitStreamExtractorStatic

fünf Stunden. Zusätzlich unterstützt der Codec derzeit als Ein- und Ausgabeformat nur YUV-Video wodurch eine sinnvolle Speicherung des Eingangsmaterials bei HD-Auflösung nicht möglich ist.

Der JSVM-Referenzcode ist daher er als Proof of Concept zu verstehen, um interessierte Anwender mit H264-SVC bekannt zu machen und diesen eine technische Basis zum Experimentieren zu geben.

4.2. Scalable Streaming Projekt

4.2.1. Einleitung

Im Folgenden wird ein Projekt von Siyuan Xiang vorgestellt, das das Konzept und die Implementierung eines Frameworks für adaptives skalierbares Video Streaming über HTTP beschreibt. Das Projekt gewann im Mai 2011 den dritten Platz beim BCNET Digital Media Challenge Studenten Wettbewerb in Vancouver [20].

Zur Zeit werden auf kommerziellen Webseiten meist zwei Techniken eingesetzt, um Video über HTTP zu streamen. Zum einen gibt es Progressive Download, wie es zum Beispiel bei den Videoplattformen YouTube und Vimeo verwendet wird, des Weiteren gibt es Adaptive Streaming, was in verschiedenen Implementierungen vorliegt, wie zum Beispiel Microsoft Smooth Streaming, Apple HTTP Live Streaming oder Adobe HTTP Dynamic Streaming.

Durch adaptives Streaming kann die Video Bitrate dynamisch angepasst werden, um verschiedene Bandbreiten mit einem passenden Videostream zu bedienen und verhindert dabei, dass zu viel Daten vorgeladen werden müssen, wenn mehr Bandbreite zur Verfügung steht. Dabei kann es jedoch vorkommen, dass Daten umsonst geladen werden. Video Server, die adaptives Streaming anbieten, müssen verschiedene Kopien des selben Videos für verschiedene Bitraten und Bandbreiten bereitstellen, um verschiedene Clients optimal bedienen zu können. Dadurch ist viel Speicherplatz auf den Servern nötig und Cachetreffer werden verringert. Durch Scalable Video Coding können diese Nachteile beseitigt werden. Laut [16] können bis 56% an Speicherplatz auf den Servern durch SVC eingespart werden.

4.2.2. Projektkurzübersicht

1. Ein Extractor wurde konzipiert und implementiert, um H.264/SVC Bitstream Abstraction Layer (NAL) Einheiten zu analysieren und um den enkodierten Bitstream in Layer-Segmente aufzuteilen.
2. Ein Rate-Adaption-Algorithmus wurde konzipiert und implementiert, um die Video Bitrate an die verschiedenen Bandbreiten und Client-Anforderungen anzupassen.
3. Ein einfacher Video-Player wurde mittels OpenSVC-Decoder und SDL (Simple DirectMedia Layer) implementiert, um die kommerzielle Nutzung darzustellen.

4.2.3. Framework

Der Video-Player besteht aus den drei Komponenten Rate-Adaption, Decoder und Display. Der Rate-Adaption-Algorithmus entscheidet, welche Video-Version beziehungsweise welcher Layer angefragt werden soll und sendet einen HTTP-Request an den Streaming-Server, welcher mit den angefragten Segmenten antwortet. Die empfangenen Segmente wer-

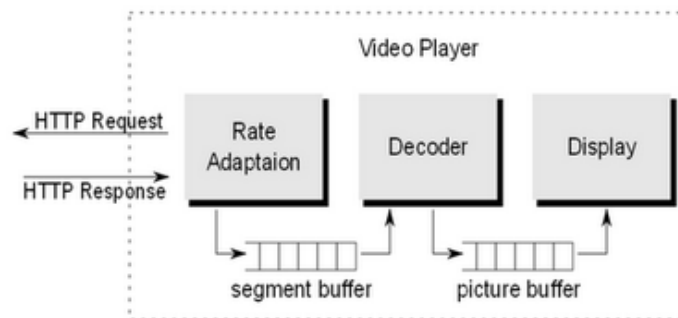


Abbildung 4.7.: Aufbau des Videoplayers

den im Segment-Buffer gespeichert. Im Decoder werden die Segmente so schnell wie möglich dekodiert und daraufhin in den Picture-Buffer abgespeichert. Die Display-Komponente stellt die Bilder aus dem Picture-Buffer in einer konstanten Bildwiederholungsrate dar.

Der komplette Prozess des adaptiven skalierbaren Video-Streaming über HTTP wird in der Projektdokumentation anhand eines spezifischen Beispiels dargestellt. Eine Übersicht der Komponenten:

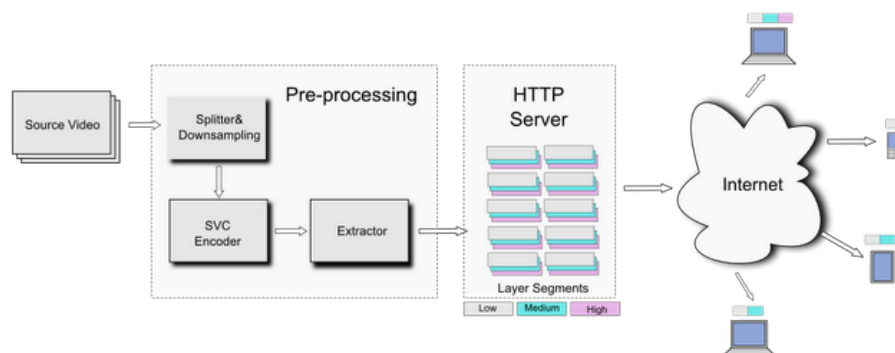


Abbildung 4.8.: Aufbau des Frameworks

Als Beispiel-Video dient Big Buck Bunny, ein Open-Movie-Projekt, in einer Ausgangsauf-
 lösung von 1080x720. Im ersten Schritt werden aus dem Video durch Downsampling zwei
 neue Versionen des Videos erstellt, welche auf die Auflösung 640x360 und 320x180 durch
 das Downsampling reduziert wurden. Dann wird jedes Video in kleinere Segmente mit je-
 weils 17 Bildern unterteilt. Da die Bildwiederholungsrate 24 Bilder pro Sekunde ist, hat
 jedes Segment eine Dauer von ca. 0,7 Sekunden, wenn die ersten 3400 Bilder genutzt wer-

Resolution	Avg. Bitrate (Kbps)	Y-PSNR	Layer Index
320x180	112.84	35.47	1
320x180	238.94	39.44	2
640x360	363.82	35.96	3
640x360	673.68	39.64	4
1080x720	955.84	36.48	5
1080x720	1997.37	40.05	6

Tabelle 4.2.: Übersicht der Layer der enkodierten Datei

den. Weiterhin wird jedes Segment mit dem SVC-Encoder (JSVM) enkodiert. Bei diesem Beispiel werden drei Auflösungen verwendet und jede Auflösung hat zwei Quality-Layer. Die folgende Tabelle fasst die Bitrate- und Video-Quality-Layer zusammen. Eine Audiospur wurde hierbei nicht berücksichtigt.

Im nächsten Schritt werden im Extractor die NAL-Units eines Layers mit dem gleichen Layer-Index aus den Segment-Frames extrahiert und zusammen in einer Layer-Segment-Datei gespeichert. Außerdem wird die Struktur der NAL-Units eines Segments analysiert und in einer Textdatei gespeichert. Ein einfacher HTTP-Server kann ähnlich wie beim Progressive Download als Streaming Server verwendet werden. Für dieses Projekt wurde `lighttpd` als Web-Server benutzt. Dieser wird auch von YouTube eingesetzt. Wichtig ist die `server.max-keep-alive-requests` Option von `lighttpd`, welche angibt, wie viele Anfragen vom Server bearbeitet werden können, bevor der Server die Verbindung schließt, da der Client Piplining benutzt, um die Layer-Segmente anzufragen. Wenn der Video-Player die Wiedergabe startet, erhält er zuerst Informationen über die Anzahl der Layer und die durchschnittliche Video-Bitrate.

Im letzten Schritt sucht der Player/Rate-Adaption-Algorithmus die passendsten Layer bei Berücksichtigung von Bandbreite, Bildschirmgröße des Geräts und Rechenleistung. Zum Beispiel ein Smartphone mit einem kleinen Bildschirm wird den low-Layer mit einer Auflösung von 320x180 anfragen. Im Gegenteil wird ein Laptop mit großem Bildschirm und mehr Rechenleistung zum Dekodieren die Layer low bis high anfragen, um die bestmögliche Qualität darzustellen.

4.3. Weitere Implementierungen

4.3.1. Google Talk

Google Talk und Google Hangout sind Instant Messaging-Dienst, die neben Textmessaging auch Videochat für eine oder mehrere Personen anbieten. Google benutzt dabei eine Videokonferenzsoftware von Vidyo. Derzeit muß zur Nutzung noch ein Plugin installiert werden, welches in Zukunft durch die Verwendung von nativen WebRTC abgelöst werden soll.

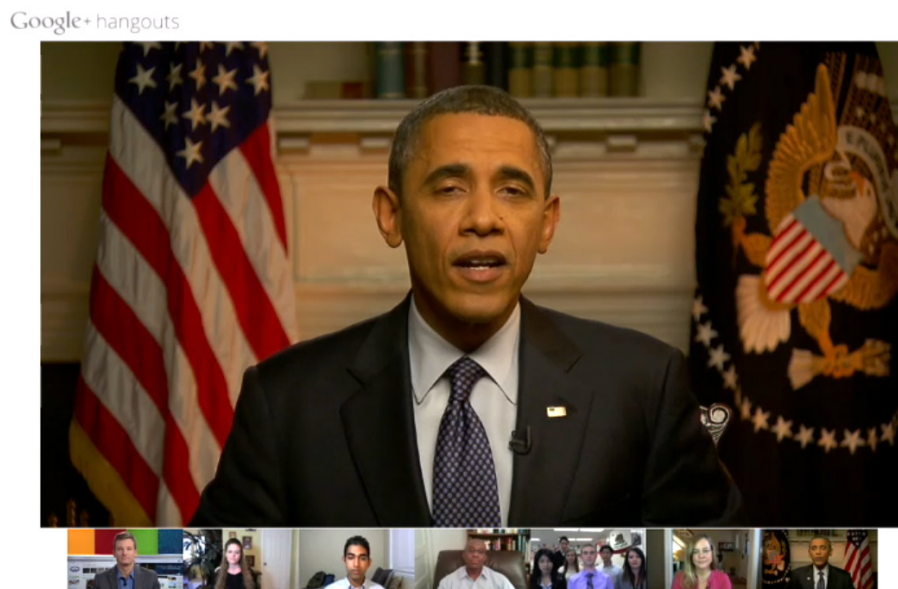


Abbildung 4.9.: Google Hangouts

Videochat in Google Talk und Google Hangout unterstützt bis jetzt den H264-SVC, H264/AVC und H263 Video-Codec, mit dem Scalable Baseline (SVC) und Baseline (AVC) Profile wie in der H.264-Spezifikation beschrieben. H264-SVC wird, wenn verfügbar, vorrangig benutzt. H263 ist nur aus Kompatibilitätsgründen angeboten.

Die folgende Abbildung zeigt die Liste der verwendeten Codecs und deren spatiale und temporale Auflösung.

```
<payload-type id="99" name="H264-SVC">  
  <parameter name="width" value="640"/>
```

```
<parameter name="height " value="400"/>
<parameter name="framerate " value="30"/>
</payload-type>
<payload-type id="96" name="H264-SVC-draft-02">
  <parameter name="width " value="640"/>
  <parameter name="height " value="400"/>
  <parameter name="framerate " value="30"/>
</payload-type>
<payload-type id="97" name="H264">
  <parameter name="width " value="640"/>
  <parameter name="height " value="400"/>
  <parameter name="framerate " value="30"/>
</payload-type>
<payload-type id="98" name="H263">
  <parameter name="width " value="640"/>
  <parameter name="height " value="400"/>
  <parameter name="framerate " value="30"/>
</payload-type>
```

4.3.2. MainConcept SVC

MainConcept bietet einen H264-SVC-Decoder innerhalb ihres kostenpflichtigen MainConcept SDKs an. In diesem sind FFDSHOW-Filter und Dekoder für verschiedene Inputformate enthalten. Zum Testen dieser Filter bietet MainConcept eine kostenlose Showcaseapplikation als Download an. Diese kann unter folgender URL bezogen werden <http://www.mainconcept.com/en/products/sdks/others/showcase.html>. Innerhalb der Showcaseapplikation wird in jedes Video ein Wasserzeichen eingeblendet.

Der SVC-Decoder bietet Möglichkeit sowohl alle Layer gleichzeitig anzuzeigen als auch selektiv nur einen bestimmten Layer anzuzeigen, z.B. nur Layer 0 mit der Framerate 7.5 Frames pro Sekunde.

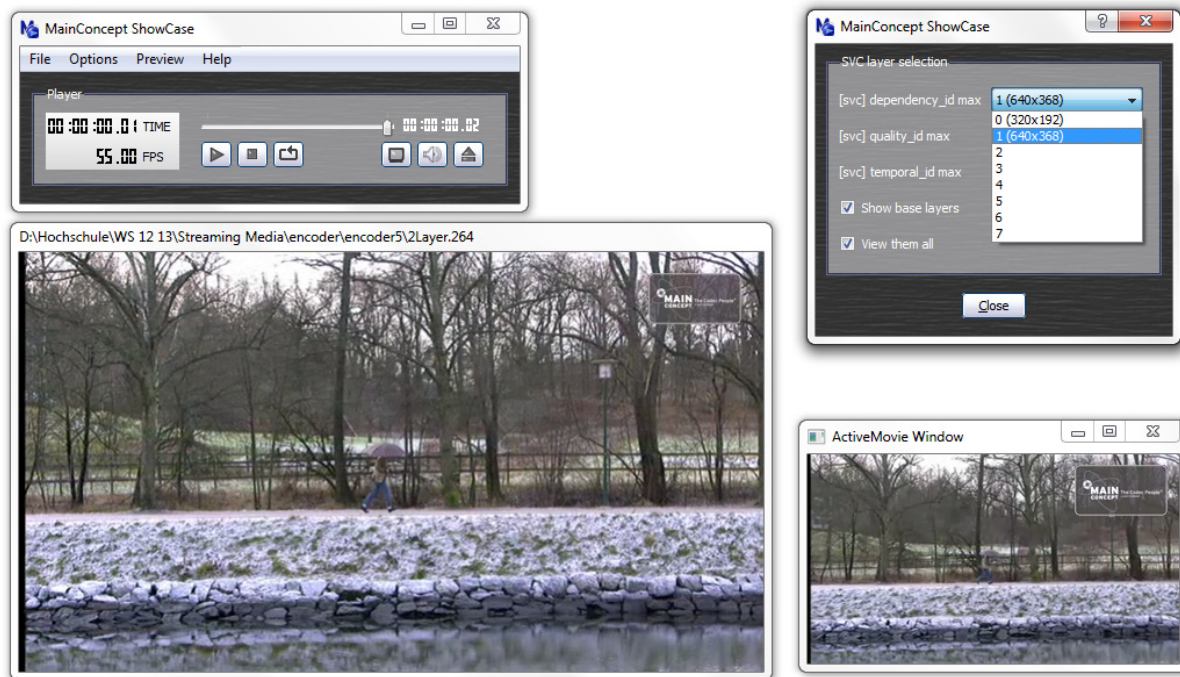


Abbildung 4.10.: MainConcept SVC-Dekoder

4.3.3. OpenSVC Decoder

Der OpenSVC Decoder kann auf folgender Website heruntergeladen werden (<http://sourceforge.net/projects/opensvcdecoder/>) Im Lieferumfang befinden sich im Verzeichnis „Mplayer“ eine speziell angepasste Version des MPplayers. Dieder lässt sich sowohl unter Windows mit Hilfe von MinGW als auch unter Linux installieren.

Die Konfiguration der Installation unter Linux erfolgt durch den Aufruf des Befehls „configure –enable-svc“ im Verzeichnis Mplayer der heruntergeladenen Datei. Die Installation wird durch die Befehle „make install“ und anschließend „install“ abgeschlossen.

Der Aufruf der Videodatei und die Auswahl des Layers erfolgt auf der Kommandozeile mit folgender Syntax:

```
mplayer -fps 25 InputVideodatei.264 setlayer X
```

Listing 4.8: Aufruf OpenSVC Decoder

5. Konklusion

Die Verwendung von H264-SVC zur Distribution von Videoformaten bringt entscheidende Vorteile gegenüber herkömmlichen Verfahren, wie das Simulcastverfahren, also der Übertragung mehrerer unabhängiger Videoströme in unterschiedlicher Qualität.

Im praktischen Anwendungsfall, dass gleichzeitig ein Videostrom an mobile Endgeräte, HD-Ready-Geräte und Geräte mit FullHD- Auflösung übertragen wird, braucht man durch die Nutzung von spatialen Enhancementlayers keine dedizierten Videodateien für bestimmte Endgeräteklassen zu enkodieren und archivieren. Stattdessen wird ein Baselayer für alle Endgeräte enkodiert, der nur den Videostrom in SD-Auflösung enthält. Mobile Endgeräte dekodieren ausschließlich diesen Baselayer. Geräte mit HD-Ready-Auflösung dekodieren zusätzlich den ersten Enhancementlayer in HD-Ready-Auflösung, der die Differenz zum Baselayer enthält. Für Geräte mit FullHD-Display werden alle Enhancementlayer dekodiert. Dadurch ist man sehr flexibel in Hinblick auf die entsprechenden Endgeräte und kann sehr einfach für weitere Geräte zusätzliche Enhancementlayer definieren. [10, S. 602]

Nachteilig an H264-SVC ist allerdings, dass es derzeit keine große Anzahl an kommerziellen Anbieter von SVC-basierten Applikationen gibt, die den kompletten Workflow von Enkodierung, Übertragung und Dekodierung abbilden. Dies ist daran zu begründen, dass in der Regel nicht SVC-basierende Lösungen einfacher zu implementieren sind, da sie auf etablierten Techniken wie reguläres H264/AVC basieren und dadurch mittlerweile einen gewissen Reifegrad erreicht haben.

Allerdings schreitet die Diversifikation der Endgeräte und Übertragungsmedien weiter voran, sodass auf lange Sicht durch Simulcast nicht alle Geräteklassen, Qualitätsstufen und Übertragungsmedien abbildbar sind und somit nicht für jedes Gerät die optimale Videoübertragung verfügbar ist.

Auf lange Sicht wird daher der Bedarf an skalierbaren Videocodecs weiter steigen und damit wird auch die Verbreitung von H264-SVC zunehmen [10, S. 603]

A. Abbildungsverzeichnis

3.1. Arten der Skalierbarkeit	6
3.2. Funktionsweise eines H264/AVC-Encoders	8
3.3. Base- und Enhancementlayer	9
3.4. Encodierung von Baslayer und Enhancementlayer	10
3.5. Dekodierung von spatialen Enhancementlayern	11
3.6. Temporale Skalierbarkeit	12
3.7. Media Presentation Document und H264-SVC	13
4.1. Das Quellvideo „Park“ in unterschiedlichen Auflösung	16
4.2. Erzeugung von 360p und 180p YUV-Videodateien	17
4.3. Spatale und Temporale Encodierung	19
4.4. Dekodierung und Rekonstruktion des Videostroms	20
4.5. Anzeige der Layerstruktur mit Hilfe von BitStreamExtractorStaticd	21
4.6. Anzeige der geänderten Layerstruktur mit Hilfe von BitStreamExtractor-Staticd	22
4.7. Aufbau des Videoplayers	24
4.8. Aufbau des Frameworks	24
4.9. Google Hangouts	26
4.10. MainConcept SVC-Dekoder	28

B. Literatur

- [1] Chuan-Yu Cho. “Overview on Scalable Video Coding - II”. In: (2010). URL: http://vc.cs.nthu.edu.tw/home/paper/codfiles/cycho/200507272019/Overview_on_Scalable_Video_Coding_-_II.ppt.
 - [2] Cisco Visual Networking Index: Forecast and Methodology. 2012. URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html.
 - [3] Heinrich Hertz Institut Fraunhofer. “Advanced MPEG Dash”. In: (2012). URL: <http://www.hhi.fraunhofer.de/de/kompetenzfelder/image-processing/research-groups/multimedia-communications/advanced-mpeg-dash.html>.
 - [4] Ralf Hohlfeld. *Crossmedia -s Wer bleibt auf der Strecke?: Beiträge aus Wissenschaft und Praxis*. 2010.
 - [5] Nikolaus Hottong. *Digitale AV-Technik WS 09/10*. 2009.
 - [6] Ronaldo Husemann Javier Del Ser Valter Roesler Aitor Rodríguez Pedro Sánchez Iraide Unanue Iñigo Urteaga. “A Tutorial on H.264/SVC Scalable Video Coding and its Tradeoff between Quality, Coding Efficiency and Performance”. In: (2011). URL: http://cdn.intechopen.com/pdfs/15138/InTech-A_tutorial_on_h_264_svc_scalable_video_coding_and_its_tradeoff_between_quality_coding_efficiency_and_performance.pdf.
 - [7] Iain E. Richardson. *H.264 and MPEG-4 Video Compression - Video Coding for Next-Generation Multimedia*. 2008.
 - [8] Iain E. Richardson. *The H.264 Advanced Video Compression Standard*. 2011.
-

- [9] Patrick Seeling Lina J. Karam Rohan Gupta Akshay Pulipaka und Martin Reisslein. “H.264 Coarse Grain Scalable (CGS) and Medium Grain Scalable (MGS) Encoded Video: A Trace Based Traffic and Quality Evaluation”. In: (2012). URL: http://mre.faculty.asu.edu/CGS_MGS_Traffic.pdf.
 - [10] Ellie Sader. “Scalable Video Coding (SVC) & Broadcast Delivery of 1080P”. In: (2008).
 - [11] Jörg Schieb. “Das Internet wird immer schneller - aber nur langsam”. In: (2012). URL: http://wdrblog.de/joergschieb/archives/2012/08/internet_tempo.html.
 - [12] Heiko Schwarz, Detlev Marpe und Thomas Wiegand. “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard”. In: *IEEE Transactions on Circuits and Systems for Video Technology* (2007). URL: http://ip.hhi.de/imagecom_G1/assets/pdfs/Overview_SVC_IEEE07.pdf.
 - [13] Puech Shahid Chaumont. “Scalable Video Coding”. In: (2010). URL: http://cdn.intechopen.com/pdfs/15732/InTech-Scalable_video_coding.pdf.
 - [14] Offer Shapiro. “H.264/SVC (Scalable Video Coding) - New Video Compression Standard”. In: (2009). URL: http://www.isccast.com/podcasts.isc365.com/powerpoint/09ISCWest_DI08.pdf.
 - [15] Stefan Slivinski. *SVC Demystified: What is Temporal Scalability?* 2011. URL: <http://www.lifesize.com/videoconferencingspot.com/?p=1406>.
 - [16] Yago Sánchez. *iDASH: improved Dynamic Adaptive Streaming over HTTP using Scalable Video Coding*. 2011. URL: http://web.cs.wpi.edu/~claypool/mmsys-2011/Day3-5_idash.pdf.
 - [17] Yago Sánchez. “Title: Scalable Video Coding based DASH for efficient usage of network resources”. In: (). URL: http://www.w3.org/2011/09/webtv/slides/w3c_presentation_v4.pdf.
 - [18] Brian Stelter. “Ownership of TV Sets Falls in U.S.” In: (2011). URL: <http://www.nytimes.com/2011/05/03/business/media/03television.html>.
-

- [19] Mathias Wien. “Performance Analysis of SVC”. In: (). URL: http://ip.hhi.de/imagecom_G1/assets/pdfs/PerformanceAnalysis_SVC_IEEE07.pdf.
- [20] Siyuan Xiang. *Scalable Streaming*. 2011. URL: <https://sites.google.com/site/svchttpstreaming/>.
- [21] Joe Zaller. “Analyzing Where is Money Being Spent in the Broadcast Industry The 2012 BBS Broadcast Industry Global Project Index”. In: (2012). URL: <http://blog.devoncroft.com/2012/08/08/analyzing-where-is-money-being-spent-in-the-broadcast-industry-the-2012-bbs-broadcast-industry-global-project-index/>.
- [22] Joe Zaller. “Tracking the Evolution of Broadcast Industry Trends 2009 2012”. In: (2012). URL: [http://blog.devoncroft.com/2012/04/09/tracking-the-evolution-of-broadcast-industry-trends-2009-2012/\(Stand11.11.2012\)](http://blog.devoncroft.com/2012/04/09/tracking-the-evolution-of-broadcast-industry-trends-2009-2012/(Stand11.11.2012)).
- [23] *Zukunft des Fernsehens: HbbTV für zusätzliche Informationen aus dem Internet*. 2011. URL: <http://www.techfacts.de/news/technik/zukunft-des-fernsehens-hbbtv-fuer-zusaetzliche-informationen-aus-dem-internet>.