Practicum Multimedia

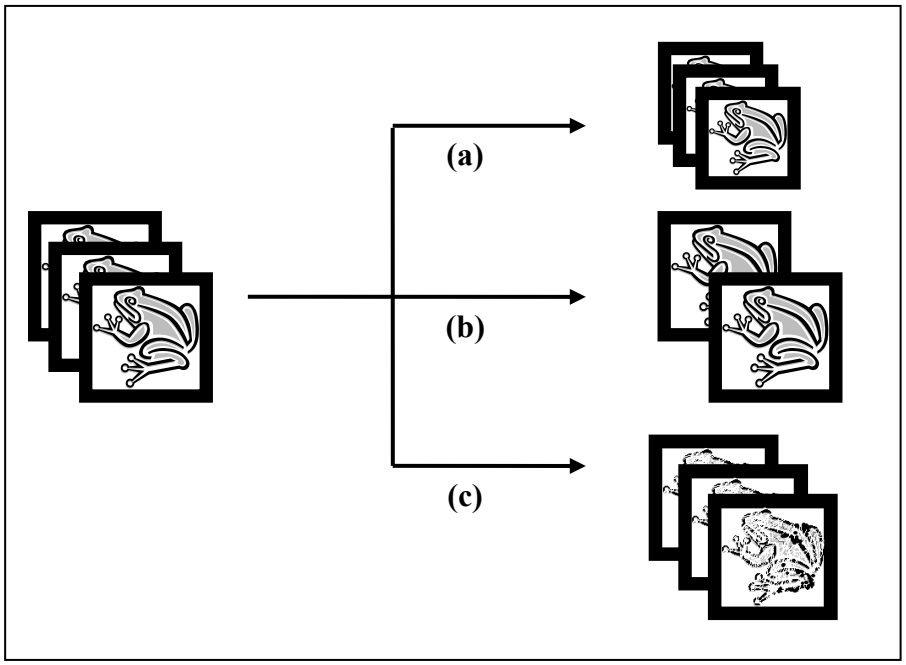
Videoadaptatie & schaalbare videocodering

1. Inleiding: schaalbare videocodering

Wanneer men spreekt over **schaalbare videocodering**, heeft men het over codeeralgoritmes die *schaalbare* of *gelaagde* bitstromen afleveren. Dit zijn bitstromen waaruit gemakkelijk, met behulp van eenvoudige editeerbewerkingen, aangepaste versies kunnen afgeleid worden. Dergelijke aangepaste versies zullen op één of andere manier een lagere kwaliteit afleveren, maar anderzijds ook minder eisen opleggen aan de decoder (hardware en software) en/of het netwerk. Conceptueel kan men de bitstroom dus beschouwen als een basislaag, die een minimale kwaliteit aanbiedt, en één of meerdere verbeteringslagen, die telkens de kwaliteit verbeteren, maar ook hogere eisen stellen.
Die lagere kwaliteit kan verschillende vormen aannemen. Over het algemeen worden drie vormen van schaalbaarheid onderscheiden, zoals ook weergegeven in
Figuur 1:

- spatiale schaalbaarheid, waarbij een lagere kwaliteit een vermindering in resolutie (aantal pixels) betekent;
- temporele schaalbaarheid, waarbij een lagere kwaliteit overeenkomt met een lagere beeldsnelheid (het aantal beelden per seconde), wat als minder vloeiend of schokkerig ervaren wordt;
- SNR-schaalbaarheid of kwaliteitsschaalbaarheid, waarbij een lagere kwaliteit zal betekenen dat er meer artefacten (codeerfouten) zichtbaar zullen zijn in de gedecodeerde beelden.

Nogal wat multimediatoepassingen kunnen de voordelen van schaalbare videocodering gebruiken. Zo kan een aanbieder van schaalbare videostromen zeer veel verschillende soorten klanten tegelijk bedienen, omdat de schaalbare data gemakkelijk kan aangepast worden aan beperkingen opgelegd door het netwerk, de hardware of de software van de ontvanger. In het geval van videostroming over een foutgevoelig kanaal kan men de basislaag, die een minimale kwaliteit biedt, trachten beter te beschermen dan de andere data, bijvoorbeeld door de data te dupliceren. Ook kan men heel snel reageren op veranderingen in de beschikbare bandbreedte door het aantal verbeteringslagen systematisch aan te passen.
De eerste standaard voor schaalbare videocodering was een onderdeel van de MPEG-2-standaard voor videocodering. Hierin was het toegelaten om een bitstroom te genereren die uit twee lagen bestond, een basislaag en een verbeteringslaag. In de MPEG-4 Visual standaard kon een encoder ook meer dan twee lagen aanbieden en werd de zogenaamde fijnkorrelige schaalbaarheid geïntroduceerd. Hierbij kon men binnen de verbeteringslaag de data van een beeld op om het even welke plaats afbreken, zodat de uiteindelijke bandbreedte heel gemakkelijk en heel nauwkeurig aanpasbaar was.
Ondanks de voordelen voor verschillende multimediatoepassingen werden deze technieken nooit op grote schaal gecommercialiseerd. Hiervoor waren verschillende redenen, maar één ervan was dat deze methodes voor een significant verlies aan compressie-efficiëntie zorgden: om dezelfde kwaliteit te bekomen waren er veel meer bits nodig bij gebruik van een schaalbaar formaat dan bij een niet-schaalbaar formaat.



Figuur 1. Overzicht van de verschillende vormen van schaalbaarheid in videocodering: (a) spatiale schaalbaarheid, (b) temporele schaalbaarheid en (c) SNR-schaalbaarheid.

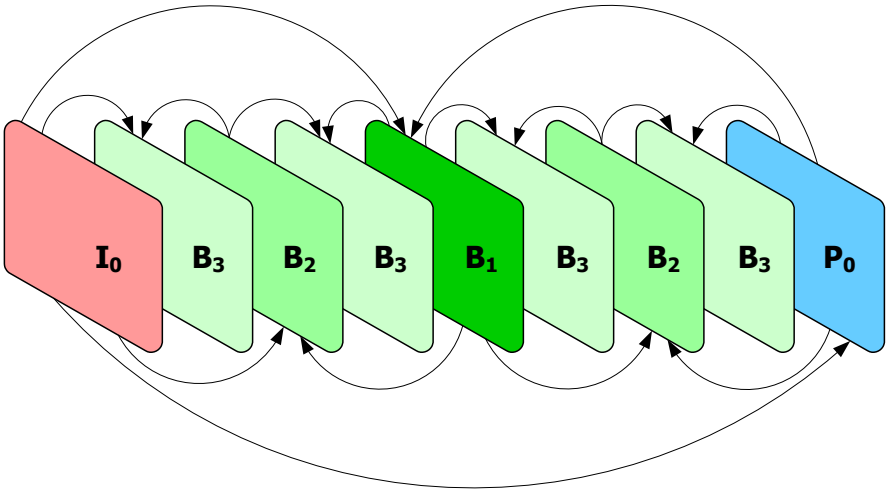
Sinds enkele jaren is de interesse voor schaalbare videocodering aanzienlijk toegenomen. De belangrijkste oorzaak daarvan is ongetwijfeld de groeiende heterogeniteit die men aantreft op het internet: de hoeveelheid toestellen die toegang hebben tot het internet en digitale video neemt steeds maar toe, zowel in termen van hardware en software, als in termen van netwerkverbinding.
Hier kan schaalbare videocodering een oplossing bieden: er moet slechts één keer een hogekwaliteitsversie aangemaakt worden om tientallen verschillende versies, elk met hun eigen kwaliteitsniveau, en elk met hun eigen vereisten, te kunnen aanbieden. Die aanpassingen kunnen gebeuren aan de hand van vrij eenvoudige editeerbewerkingen, zoals het weglaten van de juiste blokken van data.
In 2007 werd de schaalbare extensie van H.264/AVC gefinaliseerd, kortweg bekend als Scalable Video Coding (SVC). Waar H.264/AVC enkel temporele schaalbaarheid ondersteunt, geeft SVC de mogelijkheid om zowel spatiale, temporele als SNR-schaalbaarheid uit te buiten.

2. Kennismaking met SVC

De eerste opgave van dit practicum is een korte kennismaking met de schaalbare extensie van H.264/AVC. In het bijzonder zal aandacht worden besteed aan de werking van de referentiesoftware en de verschillende schaalbaarheidsassen.

De oorspronkelijke H.264/AVC-standaard (zonder SVC-uitbreiding) biedt reeds de mogelijkheid om *temporeel* schaalbare videostromen te creëren. Dit is te danken aan de hoge flexibiliteit die H.264/AVC toelaat bij het ordenen van referentiebeelden. Door de codering van de beelden op een hiërarchische manier te organiseren kan een temporeel schaalbare structuur verkregen worden. Een dergelijke structuur wordt weergegeven in Figuur 2 voor

vier temporele niveaus. Niveau 0 bevat hierbij beelden I_0 en P_0; niveau 1 bevat een enkel beeld B_1; enz. In deze structuur worden afhankelijkheden van *hogere* temporele niveaus vermeden. Als gevolg hiervan kunnen achtereenvolgens niveaus geschrapt worden (van hoog naar laag), zonder de resterende bitstroom te schaden. In dit voorbeeld kan een H.264/AVC-stroom met een beeldsnelheid van 24 Hz op die manier eenvoudig gereduceerd worden tot 12 Hz, 6 Hz of 3 Hz.



Figuur 2. Temporele schaalbaarheid bij H.264/AVC a.d.h.v. een hiërarchische codeerstructuur.

De definitie van **GOP-lengte** bij SVC heeft een andere betekenis dan bij H.264/AVC. Deze waarde komt bij SVC namelijk overeen met het aantal beelden aanwezig tussen twee opeenvolgende beelden van temporeel niveau 0. In het voorbeeld van Figuur 2 is de GOP-lengte met andere woorden de afstand tussen de beelden I_0 en P_0, i.e., een GOP-lengte van acht. De **intraperiode** wordt gedefinieerd als de afstand tussen twee I-beelden (en is dus $\geq$ de GOP-lengte).

Typisch wordt in de context van SVC steeds een dergelijke hiërarchische codeerstructuur gebruikt. SVC zal de mogelijkheden van H.264/AVC uitbreiden door meerdere kwaliteits- en/of spatiale lagen toe te laten. Op die manier kunnen, door het verwijderen van één of meerdere lagen, uit een enkele stroom sub-stromen worden geëxtraheerd met een lagere kwaliteit en/of bitsnelheid. De gelaagde structuur zorgt ervoor dat deze sub-stromen zonder problemen kunnen gedecodeerd worden en dat dergelijke aanpassingen op zeer eenvoudige wijze kunnen uitgevoerd worden (bv. in netwerknodes). Uiteraard bevat de basislaag (laag 0) de minimale kwaliteit en/of resolutie, en wordt de maximale kwaliteit en/of resolutie bereikt wanneer alle lagen nog aanwezig zijn aan de decoderzijde.

Bij de ontwikkeling van SVC werd in het bijzonder aandacht besteed aan het minimaliseren van het verlies aan codeerefficiëntie in vergelijking met éénlaagscodering (*single-layer* codering) met H.264/AVC. Verschillende predictietechnieken werden voorzien in SVC om een reductie van de bitsnelheid te verkrijgen in vergelijking met simulcast-oplossingen, door predictie toe te laten tussen de verschillende lagen (*inter-layer* predictie). Dergelijke simulcast-oplossingen worden vaak gebruikt in de praktijk door eenvoudigweg verschillende versies van dezelfde videocontent (verschillende resoluties en/of kwaliteit) ter beschikking te stellen op de server.

In dit practicum maken we gebruik van de Joint Scalable Video Model (JSVM) referentiesoftware, ontwikkeld door het Joint Video Team (JVT). Dit softwarepakket bevat onder andere een SVC-encoder, een SVC-decoder, en een tool die onderbemonstering mogelijk maakt voor het geval van spatiale schaalbaarheid. De parameters die de encoder moet gebruiken bij het encoderen worden meegegeven via configuratiebestanden. Naast een algemeen configuratiebestand (dat o.a. aangeeft hoeveel kwaliteits- of spatiale lagen er gebruikt worden), dient er bijkomend per laag een configuratiebestand meegegeven te worden (dat o.a. de QP en de beeldsnelheid voor de respectieve lagen aangeeft).
Voor dit practicum wordt gevraagd dat jullie de gegeven configuratiebestanden aanvullen volgens de hieronder gevraagde configuraties. Op basis van de gegenereerde bitstromen worden vervolgens RD-grafieken opgesteld, die een idee geven van de impact van schaalbaarheid op de codeerefficiëntie.

Opmerkingen:
- **Alle** gevraagde stromen voorzien sowieso temporele schaalbaarheid wanneer ze met JSVM geëncodeerd worden. Met 'single-layer' bedoelen we m.a.w. H.264/AVC-stromen met één enkele kwaliteits- of spatiale laag.
- Het is niet nodig om de volledige sequentie te encoderen; enkel de eerste **50 beelden** volstaan.
- Om de RD-punten te bepalen kunnen jullie ofwel (i) de informatie gebruiken die voorzien wordt door de JSVM-encoder ofwel (ii) een eigen meting van bitsnelheid en PSNR uitvoeren na extractie van de gevraagde punten met de BitStreamExtractor. Deze laatste methode maakt jullie vertrouwd met het extraheren van lagen, ter voorbereiding op het tweede deel van dit practicum.

2.1. Opgave: Kwaliteitsschaalbaarheid

Maak een vergelijking van single-layer codering en kwaliteitsschaalbaarheid met SVC voor de sequentie "BBB_640x360_24.yuv" (Big Buck Bunny) met volgende configuraties:
- Single-layer codering
  - Genereer verschillende bitstromen met telkens één laag (wat neerkomt op een simulcast-oplossing) met vaste QP-waarden: 30, 35, 40. Gebruik de JSVM-software (met temporele schaalbaarheid) gebruikmakend van de codeerstructuur weergegeven in Figuur 2. **Verbind de resulterende RD-punten in een RD-curve**.
- Kwaliteitsschaalbaarheid met SVC
  - JSVM-software
  - Genereer een SVC-stroom met drie **MGS**-kwaliteitslagen: QP-waarden 30, 35, 40. Gebruik de JSVM-software. Doe dit met én zonder inter-layer predictie, en **zet de twee bijhorende RD-curves uit**.

Maak een RD-grafiek die de verschillende configuraties vergelijkt en bespreek.

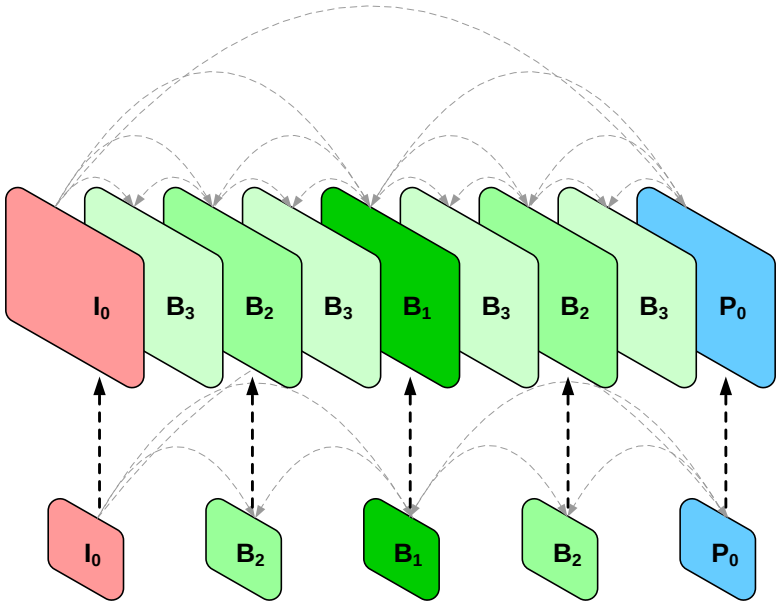
2.2. Opgave: Spatiale schaalbaarheid

Maak een vergelijking van single-layer codering, simulcast en spatiale schaalbaarheid voor de sequentie "BBB_640x360_24.yuv" met volgende configuraties:

- Single-layer codering
  - Genereer een bitstroom (één laag) met 640x360-resolutie aan 24 Hz en een tweede bitstroom (één laag) met 320x180-resolutie aan 12 Hz. Gebruik de JSVM-software en de codeerstructuur weergegeven in Figuur 3 (vaste QP-waarde: 30).
- Spatiale schaalbaarheid met SVC
  - Genereer een SVC-bitstroom met twee lagen (basislaag met 320x180-resolutie aan 12 Hz en uitbreidingslaag met 640x360-resolutie aan 24 Hz) zoals weergegeven in Figuur 3. Gebruik de JSVM-software (vaste QP-waarde: 30). Doe dit met én zonder inter-layer predictie, en **bepaal de bijhorende RD-punten**.

Voor deze opgave is het nodig een geschaalde versie van BBB_640x360_24.yuv aan te maken voor de lageresolutieversie. Hiervoor dien je gebruik te maken van de "non-normative" downscaling methode uit de resampling tool.
Besteed bij het genereren van de configuratiebestanden voldoende aandacht aan de beeldsnelheden van de basis- en de uitbreidingslaag.

Zet de resulterende RD-punten uit in een grafiek en bespreek.



Figuur 3. Voorbeeld van temporele en spatiale schaalbaarheid.

2.3. In te dienen bestanden

Volgende bestanden moeten ingediend worden voor de eerste opgave van dit practicum:

1. Een kort verslag met de vergelijkingen en interpretatie. Vermeld eveneens de gebruikte commandolijn-instructies voor het encoderen van de verschillende stromen (SVC.pdf of SVC.doc(x)).
2. De gebruikte configuratiebestanden voor de kwaliteits- en de spatiale schaalbaarheid. Gebruik de gepaste meegeleverde configuratiebestanden als vertrekpunt.

2.4. Software

Voor deze eerste opgave wordt gebruikgemaakt van de laatste versie van de JSVM-referentiesoftware (versie 9.19) van SVC. Deze bevat ondermeer volgende tools.

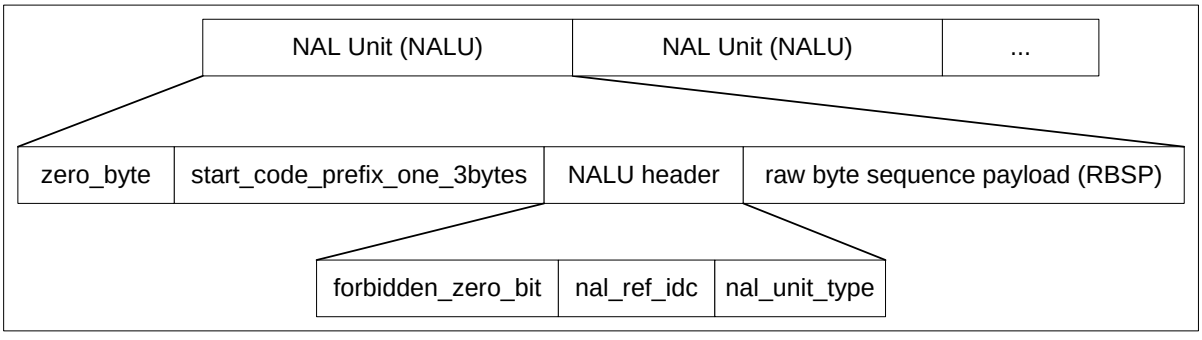
- Resampling tool voor het herschalen van video (DownConvertStatic.exe).
- JSVM-encoder (H264AVCEncoderLibTestStatic.exe).
- JSVM-decoder (H264AVCDecoderLibTestStatic.exe).
- Bitstroomextractie tool (BitStreamExtractorStatic.exe).

Voor meer informatie over de verschillende parameters wordt verwezen naar de bijhorende presentatie.

3. SVC-adaptatie

3.1. De H.264/AVC-specificatie

De H.264/AVC-standaard of "MPEG-4 Part 10, Advanced Video Coding" is momenteel de meest courante videocoderingsstandaard. Naast de verbeterde compressie-efficiëntie ten opzichte van zijn voorgangers maakt H.264/AVC gebruik van een netwerkvriendelijke representatie. De H.264/AVC-standaard maakt een onderscheid tussen de Videocoderingslaag (VCL) en de Netwerkabstractielaag (NAL). Het resultaat van het codeerproces is VCL-data; dit zijn een opeenvolging van bits die de gecodeerde videosequentie voorstellen. Deze worden omgevormd tot NAL-eenheden die gebruikt worden voor opslag en transport. Hierdoor wordt er een duidelijk onderscheid gemaakt tussen het feitelijke coderen en het transport-specifieke formaat.
Een H.264/AVC-bitstroom bestaat uit een opeenvolging van Network Abstraction Layer Units (NALUs) die elk een NALU-header (1 byte) en een Raw Byte Sequence Payload (RBSP) bevatten (Figuur 4). Een RBSP bevat de eigenlijke data van de gecodeerde videosequentie of informatie over de videosequentie (afmetingen, parameters voor de decoder, ...). Het begin van een NALU wordt aangegeven door een startcode van 3 bytes: 0x000001, die eventueel kan voorafgegaan worden door één of meerdere zerobytes.



Figuur 4. Structuur van een H.264/AVC-bitstroom.

Hieronder is een vereenvoudigde vorm van een klein gedeelte van de H.264/AVC-specificatie gegeven. Met deze informatie heb je in principe genoeg om een H.264/AVC-bitstroom te beschrijven tot op hoogte van de NALU-header. Voor de geïnteresseerden kan de volledige versie van de H.264/AVC-specificatie hier gedownload worden.

H.264/AVC-bitstroom syntax

H.264/AVC-bitstroom(){	Aantal bits	Waarde
while(!atEndOfBitstream())		
nal_unit()		
}		

NAL Unit syntax

nal_unit(){	Aantal bits	Waarde
while(nextbits(24) != 0x000001)		
zero_byte	8	0x00
start_code_prefix_one_3bytes	24	0x000001
nal_unit_header()		
raw_byte_sequence_payload()		
}		

NAL Unit header syntax

nal_unit_header(){	Aantal bits	Waarde
forbidden_zero_bit	1	0
nal_ref_idc	2	
nal_unit_type	5	
}		

Het **nal_ref_idc** syntaxelement geeft de prioriteit aan van het betreffende NALU-pakket. Pakketten die noodzakelijk zijn voor verdere decodering (zoals parametersets en referentiebeelden) krijgen typisch de waarde 3, terwijl niet-referentiebeelden en optionele pakketten zoals SEI-berichten[1] de waarde 0 krijgen. Het volgende onderscheid wordt bv. gemaakt in de referentiesoftware[2]:

prioriteit	nal_ref_idc
NALU_PRIORITY_HIGHEST	3
NALU_PRIORITY_HIGH	2
NALU_PRIORITY_LOW	1
NALU_PRIORITY_DISPOSABLE	0

Tabel 1. NALU-prioriteit (aangegeven door nal_ref_idc).

Bij de specificatie van H.264/AVC werd voorzien in een onderscheid tussen 32 mogelijke NALU-types (aangegeven door **nal_unit_type**), waarvan er bij de eerste versie van de standaard 12 werden ingevuld.
SVC-bitstromen voldoen bij uitbreiding aan bovenstaande syntax, maar voegen een aantal elementen en NALU's toe (momenteel worden 17 NALU-types gebruikt). De voornaamste

[1] SEI = supplemental enhancement information. Deze berichten geven bijkomende informatie die kan gebruikt worden door de decoder (andere decoders kunnen deze negeren zonder impact op het decodeerproces). Deze pakketten bevatten bv. informatie over timing, buffering, schaalbaarheid, 3D-informatie, of bijkomende gebruikersdata.
[2] Het onderscheid tussen de waarden 1-3 kan vrij gekozen worden afhankelijk van de toepassing (en de hiervoor gebruikte encoder en decoder).

NALU-types die voorkomen in een H.264/AVC- of SVC-bitstroom worden opgelijst in Tabel 2.

NALU-types 1 en 5 bevatten de eigenlijke (gecomprimeerde) beeldinhoud, met onderliggende syntax-elementen zoals bewegingsvectoren, data i.v.m. predictiemodes, en residuele coëfficiënten (i.e., na predictie, transformatie, quantisatie en entropiecodering). Deze NALU-types bevatten dan ook de hoofdmoot van de totale bitstroom.
Het onderscheid tussen deze twee NALU-types wordt gemaakt op basis van het beeldtype in het NALU-pakket. Een NALU met **type 5** kan enkel een 'random access' I-beeld bevatten (in de specificatie een *instantaneous decoding refresh* beeld genoemd). Een dergelijk beeld zorgt voor een 'reset' van de decoder, zodat de decoder gegarandeerd het decodeerproces kan starten op dit punt. Dit is bv. interessant bij het omschakelen/zappen tussen twee kanalen. Alle P- en B-beelden, alsook de overige I-beelden worden verpakt in een NALU met **type 1**.

nal_unit_type	inhoud
1	niet-IDR-slice/beeld
5	IDR-slice/beeld (random access)
6	SEI-bericht
7	sequence parameter set
8	picture parameter set
14	prefix NAL unit
15	subset sequence parameter set
20	slice/beeld in (SVC-)extensielaag

Tabel 2. Courante NALU-types in SVC-stromen.

In een SVC-stroom worden de beelden uit de uitbreidingslaag verpakt in NALU's met **type 20**. Dit type pakketten bevat een hoofding met een identificatie van de huidige laag (temporeel, spatiaal, en kwaliteit). Deze additionele hoofding bestaat uit drie extra bytes, zoals weergegeven in Tabel 3. Ook de AVC-(basislaag)pakketten in een SVC-stroom worden vergezeld van deze drie bytes, zij het dan in een afzonderlijk prefix-NALU-pakket (met **type 14**).

Byte #	0	1							2							3							
Bit #	AVC NALU	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
purpose	header	S	I	PRID						N	DID			QID				TID			U	D	O	PR	

Tabel 3. Opbouw van de eerste vier bytes van NALU-types 14 en 20. De eerste byte is identiek aan de H.264/AVC-NALU-header (cf. hierboven).

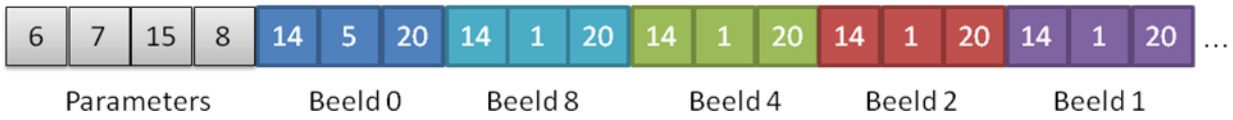
Hierbij wordt een onderscheid gemaakt tussen volgende elementen:

- S (1 bit): **svc_extension_flag**
  - Deze vlag geeft aan of bytes 1-3 gebruikt worden voor SVC- of MVC/3D-informatie. Voor dit practicum moet deze vlag gelijk zijn aan 1.
- I (1 bit): **idr_flag**
  - Deze bit specificeert of het betreffende beeld al dan niet een IDR-beeld is (random access).

- PRID (6 bits): **priority_id**
  - Deze vlag geeft een indicatie van de prioriteit van de NALU. Een lagere waarde geeft een hogere prioriteit aan.
- N (1 bit): **no_inter_layer_pred_flag**
  - Deze vlag geeft aan of inter-layer predictie al dan niet mag gebruikt worden voor het huidige beeld.
- DID (3 bits): **dependency_id**
  - Deze waarde geeft de index aan van de huidige laag bij spatiale schaalbaarheid of CGS-kwaliteitsschaalbaarheid.
- QID (4 bits): **quality_id**
  - Deze waarde geeft de index aan van de huidige laag bij MGS-kwaliteitsschaalbaarheid.
- TID (3 bits): **temporal_id**
  - Deze waarde geeft de index aan van het huidige temporele niveau in de predictiehiërarchie.
- U (1 bit): **use_ref_base_pic_flag**
  - Een waarde van 1 voor dit element geeft aan dat de basislaag wordt gebruikt tijdens inter-predictie (wordt gebruikt bij MGS-kwaliteitsschaalbaarheid).
- D (1 bit): **discardable_flag**
  - Een waarde van 1 geeft aan dat de huidige NALU niet wordt gebruikt om NALU's met een hogere waarde van dependency_id te decoderen. Deze NALU's kunnen m.a.w. verwijderd worden zonder de integriteit van hogere lagen aan te tasten.
- O (1 bit): **output_flag**
  - Deze waarde geeft aan dat het gedecodeerde beeld door de decoder als uitvoer wordt gebruikt.
- RR (2 bits): **reserved_three_2bits**
  - Gereserveerd voor toekomstige uitbreidingen (gelijk aan 3).

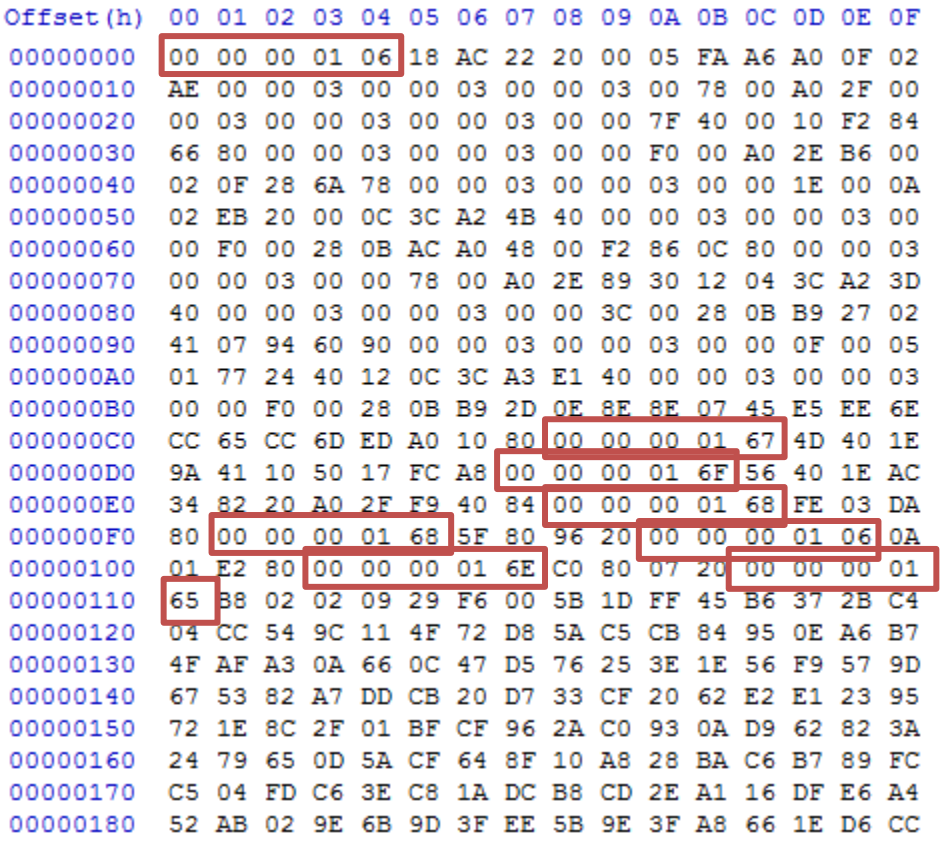
Voor het vervolg van dit practicum zijn vooral de waarden van de dependency_id, quality_id en temporal_id van belang.

Een SVC-stroom zal dan de volgende opeenvolging van NALU-pakketten krijgen (mits abstractie van een aantal bijkomende parameter- en SEI-NALU's):



Figuur 5. Typische NALU-opbouw van een SVC-stroom.

In een hex-editor kan het begin van de SVC-stroom er als volgt uitzien (met aanduiding van de start-code en de eerste byte van de eerste NALU's):



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	00	00	00	01	06	18	AC	22	20	00	05	FA	A6	A0	0F	02
00000010	AE	00	00	03	00	00	03	00	00	03	00	78	00	A0	2F	00
00000020	00	03	00	00	03	00	00	03	00	00	7F	40	00	10	F2	84
00000030	66	80	00	00	03	00	00	03	00	00	F0	00	A0	2E	B6	00
00000040	02	0F	28	6A	78	00	00	03	00	00	03	00	00	1E	00	0A
00000050	02	EB	20	00	0C	3C	A2	4B	40	00	00	03	00	00	03	00
00000060	00	F0	00	28	0B	AC	A0	48	00	F2	86	0C	80	00	00	03
00000070	00	00	03	00	00	78	00	A0	2E	89	30	12	04	3C	A2	3D
00000080	40	00	00	03	00	00	03	00	00	3C	00	28	0B	B9	27	02
00000090	41	07	94	60	90	00	00	03	00	00	03	00	00	0F	00	05
000000A0	01	77	24	40	12	0C	3C	A3	E1	40	00	00	03	00	00	03
000000B0	00	00	F0	00	28	0B	B9	2D	0E	8E	8E	07	45	E5	EE	6E
000000C0	CC	65	CC	6D	ED	A0	10	80	00	00	00	01	67	4D	40	1E
000000D0	9A	41	10	50	17	FC	A8	00	00	00	01	6F	56	40	1E	AC
000000E0	34	82	20	A0	2F	F9	40	84	00	00	00	01	68	FE	03	DA
000000F0	80	00	00	00	01	68	5F	80	96	20	00	00	00	01	06	0A
00000100	01	E2	80	00	00	00	01	6E	C0	80	07	20	00	00	00	01
00000110	65	B8	02	02	09	29	F6	00	5B	1D	FF	45	B6	37	2B	C4
00000120	04	CC	54	9C	11	4F	72	D8	5A	C5	CB	84	95	0E	A6	B7
00000130	4F	AF	A3	0A	66	0C	47	D5	76	25	3E	1E	56	F9	57	9D
00000140	67	53	82	A7	DD	CB	20	D7	33	CF	20	62	E2	E1	23	95
00000150	72	1E	8C	2F	01	BF	CF	96	2A	C0	93	0A	D9	62	82	3A
00000160	24	79	65	0D	5A	CF	64	8F	10	A8	28	BA	C6	B7	89	FC
00000170	C5	04	FD	C6	3E	C8	1A	DC	B8	CD	2E	A1	16	DF	E6	A4
00000180	52	AB	02	9E	6B	9D	3F	EE	5B	9E	3F	A8	66	1E	D6	CC

Figuur 6. Voorbeeld van een hexadecimale voorstelling van het begin van een SVC-stroom.

3.2. Opgave: SVC-adaptatie

Schrijf een SVC-parser (in C of C++) met volgende eigenschappen:

- De parser herkent de NAL-eenheden van een SVC-stroom, en kan de stroom aanpassen (NALU's verwijderen) volgens de drie schaalbaarheidsassen:
  - kwaliteit;
  - spatiale resolutie;
  - beeldsnelheid;
  - of een combinatie van de drie.
- Het resultaat moet een compatibele SVC- of H.264/AVC-bitstroom zijn, die kan gedecodeerd worden door de JSVM-decoder.
- Geef de benodigde argumenten door via de commandolijn.
- Schrijf de NALU-types van de herkende pakketten uit in een log-bestand, met indicatie of ze al dan niet verwijderd worden.
- Pas de parser toe op de gecodeerde stromen uit het eerste deel van dit practicum.
- Zorg voor een minimum aan code (onze 'modeloplossing' telt <250 regels in C++), maar hou de code leesbaar, en voeg commentaar toe.
- Tip: vergelijk jullie resultaat-stromen met deze van de SVC-BitstreamExtractor.

3.3. In te dienen bestanden

De volgende bestanden moeten ingediend worden voor de tweede opgave van dit practicum:

- Een volledige, compileerbare en uitvoerbare solution met C/C++-broncode (versie Visual Studio: vrij te kiezen). Tussentijds mogen jullie werken op een platform/editor/IDE naar keuze, maar dien uiteindelijk een VS-solution in die zonder fouten compileert.
- Een script (batch/Python) dat jullie executable (volgens jullie gekozen commandolijn-parameters) uitvoert op de twee gecreëerde **schaalbare** stromen uit secties 2.1 en 2.2 van dit practicum, en dat de bijhorende substromen extraheert uit de SVC-stromen.
- Geef in dit script tevens een voorbeeld van hoe de beeldsnelheid kan gereduceerd worden voor de single-layer H.264/AVC-bitstromen (temporele schaalbaarheid).
- De stromen zelf dienen niet meegestuurd te worden.