Multimedia(technieken)

Practicum Video 1: Audio/Video en containers

Deadline: donderdag 22 maart 14u

1 Inleiding

Een multimediabestand is meer dan enkel video. Zo een bestand bevat ook (één of meerdere) audiostromen en kan bijvoorbeeld ook ondertitels bevatten. Al deze elementen worden dan samen gebundeld in een container.

In dit practicum gaan we na hoe een multimediabestand opgebouwd wordt uit programma's, ondertitels, audio- en videostromen. Verder worden methoden aangeboden om zelf een kwaliteitsvolle montage te maken zodat het automatisch toevoegen van reclame, het toevoegen van een intro, en zelfs het toevoegen van een logo mogelijkheden worden van onze videodienst.

2 Elementaire videostromen

Een van de belangrijkste elementen van een multimediabestand is een videostroom. In zo een videostroom wordt een opeenvolging van tweedimensionale pixelstructuren omgezet naar een gecomprimeerd bestand van I-, P- en B-beelden. Hierbij is een I-beeld een beeld dat intra-gecodeerd is, wat erop neerkomt dat het voor encoderen en decoderen niet afhankelijk is van andere beelden. P- en B-beelden zijn daarentegen inter-gecodeerd en kunnen enkel gecodeerd en gedecodeerd worden als dit ook al gebeurd is voor de beelden waarvan ze afhankelijk zijn. De vergelijkingen op het gebied van compressie-efficiëntie die we in het volgende practicum gaan maken, zijn relevant voor dit deel van een multimediabestand. Compressiestandaarden die gebruikt kunnen worden om video te comprimeren zijn onder andere:

- Motion JPEG: Deze compressiestandaard definieert dat elk beeld als een afzonderlijke JPEG afbeelding geëncodeerd moet worden. Tot vandaag wordt deze standaard vaak ondersteund door digitale camera's, IP beveiligingscamera's, en webcams.
- Motion JPEG2000: Het idee in deze standaard is gelijkaardig aan dat van Motion JPEG, maar dan gebruik makende van de waveletgebaseerde JPEG2000 beeldcompressiestandaard.
- MPEG-2 Part-2 (Video) of ISO/IEC 13818-2 of ITU-T Rec. H.262: Dit is de eerste videostandaard (gefinaliseerd in november 1994) die wereldwijd op zo een grote schaal werd uitgerold dat er nog steeds toepassingen gebruik van maken tot op de dag van vandaag. Deze blokgebaseerde compressiestandaard heeft het mogelijk gemaakt om digitale video aan te leveren via de kabel, satelliet, terrestriële antennes, IPTV en optische media (DVD). Vandaag mag MPEG-2 Video zelfs nog steeds toegepast worden op Blu-Ray schijfjes.
- MPEG-4 Part-2 (Advanced Simple Profile ASP): Deze standaard is zeer bekend geworden in de illegale filmdistributiewereld als DivX en Xvid. Dit zijn 2 softwarepakketten die MPEG-4 Part-2 (ASP) compatibele videostromen genereerden.
- H.264/MPEG-4 AVC: Dit is de opvolger van MPEG-4 ASP en kent een even grote marktpenetratie als MPEG-2. De meeste HD digitale uitzendingen zijn gecomprimeerd gebruikmakende van deze standaard en op Blu-Ray schijfjes wordt deze standaard aanzien als de geprefereerde keuze. Verder wordt deze standaard uitermate veel gebruikt in een internetomgeving zoals bij Vimeo, YouTube en Netflix.
- H.265/HEVC: Deze compressiestandaard is gefinaliseerd in januari 2013 en wordt aanzien als de meest efficiënte compressiestandaard als het gaat over het verkleinen van de videostroom.
- Theora: In tegenstelling tot alle voorgaande compressiestandaarden is de Theora specificatie vrij van patenten. Bijgevolg moeten er geen licenties betaald worden voor hardwarematige implementaties die deze specificatie gebruiken. Ook al zijn de compressie-eigenschappen minderwaardig vergeleken met commerciële hedendaagse concurrenten, in de gemeenschap van open informatie is deze compressie nog steeds populair.
- Daala: Daala is de mogelijke opvolger van Theora en nog steeds in ontwikkeling. Met deze standaard probeert de open source community een competitieve standaard te ontwikkelen waar geen licentiekosten aan verbonden zijn. Uiteindelijk zijn bepaalde elementen van Daala opgenomen in AV1.

- VP8: Dit is een compressietechnologie die oorspronkelijk ontwikkeld werd door On2, maar na de overname door Google opging in de portfolio van Google. Na de overname zijn alle patenten die Google bezit en die verbonden zijn aan VP8 gedoneerd aan het publieke domein. VP8 wordt als de publiek beschikbare concurrent van H.264/AVC gezien.

- VP9: VP9, de door Google ontwikkelde opvolger van VP8, is eveneens open en zonder licentiekost. VP9 moet de concurrentie aangaan met de HEVC compressiestandaard.

- VP10: Sinds 12 september 2014 is Google begonnen aan VP10. Uiteindelijk is deze standaard nooit in gebruikgenomen, aangezien Google besloten heeft om deze op te nemen in AV1.

- AOMedia Video 1 of AV1: deze standaard is ontstaan uit een project van bedrijven zoals Amazon, Apple, ARM, Cisco, Facebook, Google, IBM, Intel, Microsoft, Mozilla, Netflix en NVIDIA die zich samengevoegd hebben onder de koepel genaamd 'Alliance for Open Media'. Hun doel met AV1 is om een compressiestandaard te hebben voor videotransmissie over het internet, die in tegenstelling tot HEVC open is en vrij van licentiekosten.

Een videocompressiestandaard houdt zich enkel bezig met het verkleinen van beelden. In een elementaire videostroom zit geen enkele informatie over wanneer deze afgebeeld moeten worden. Deze informatie zal pas in een hogere laag, zoals in de container of in het transportprotocol toegevoegd worden.

3 Elementaire audiostromen

Voor een audiostroom kan op een gelijkaardige manier een elementaire compressie gedefinieerd worden, namelijk audiosamples worden aangeboden en een gecomprimeerde audiostroom wordt gegenereerd.

- MPEG-2 Part-3 (Audio): Deze standaard wordt ook wel MP3 genoemd en is een wijdverspreide audiocompressiestandaard waarvan de eerste editie gefinaliseerd werd in 1993. Ondertussen zijn alle licenties op deze compressiestandaard verlopen en is deze dus vrij te gebruiken.

- Advanced Audio Coding (AAC): Deze standaard is binnen MPEG gestandaardiseerd als de opvolger van MP3. De High-Efficiency Advanced Audio Coding (HE-AAC) versie van AAC wordt gebruikt in DAB+ radiotransmissie.
- Free Lossless Audio Codec (FLAC): Een open en enkel verliesloze compressietechnologie. Het meestgebruikte formaat voor verliesloze compressie van audio.
- Vorbis: Een vrij beschikbare en open-source compressiestandaard. Vorbis geluidsstromen worden vaak in een Ogg-container geplaatst waardoor men vaak spreekt over Ogg Vorbis.
- Opus: Een vrij beschikbare en open-source compressiestandaard en opvolger van Vorbis. Deze standaard kan zowel audio als spraak efficiënt comprimeren en is hiervoor momenteel de meest efficiënte standaard.

4 Compressiestandaard tegenover compressiebibliotheek

Een compressiestandaard is niet hetzelfde als een compressiealgoritme of bibliotheek. Een standaard beschrijft enkel hoe een videostroom opgebouwd moet worden en hoe deze te decoderen. De standaard beschrijft niets in verband met het (aanbevolen) gebruik van deze compressiestandaard. Het is dus perfect mogelijk om een encoder te maken die door een inefficiënte implementatie een slechte prestatie neerzet voor een potentiële hoogperformante compressiestandaard. Bij vergelijkingen tussen verschillende standaarden moet er dus altijd rekening gehouden worden met de gebruikte algoritmen.

Een voorbeeld van beschikbare bibliotheken of softwarepakketten die video of audio kunnen genereren die voldoet aan een bepaalde standaard is gegeven in de volgende tabel:

Standaard	compressiebibliotheek
MPEG-4 Part-2	Xvid/libxvid , DivX, ffmpeg/mpeg4
H.264/MPEG-4 AVC	x264, JM
H.265/HEVC	x265, HM
Theora	libtheora
VP8, VP9, VP10	libvpx
MP3	lame mp3 encoder
AAC	ffmpeg/aac, ffmpeg/libfaac
FLAC	flac

Daarbij komt dat bibliotheken zoals ffmpeg, andere encoders zoals x264 en x265 hebben geëncapsuleerd zodat deze compressiealgoritmen binnen eenzelfde raamwerk aangesproken kunnen worden. De ffmpeg-uitvoering van de HEVC encoder is dus exact dezelfde als x265 omdat deze encoder als bibliotheek aangesproken wordt binnen ffmpeg. De standaardinstellingen van beide implementaties kunnen wel verschillen waardoor er toch een licht andere compressieëfficientie, kwaliteit, of complexiteit ondervonden kan worden.

5 Containerformaten

Wanneer een videostroom op een gesynchroniseerde manier samengevoegd moet worden met een audiostroom dan is een containerformaat noodzakelijk. Dit concept is verder uitgebreid zodat een videostroom ook meerdere audiostromen kan bevatten bijvoorbeeld om meerdere talen te ondersteunen. Ook ondertitelingsinformatie, een programmagids of zelfs teletekst-informatie kan meegestuurd worden in sommige formaten.

- Audio Video Interleave (.avi): deze container werd door Microsoft geïntroduceerd in 1992 en werd de defacto-standaard doorheen de jaren. Door de technische beperkingen (aspect ratio, tijdscodes, B-beelden, VBR codering) van dit formaat wordt deze amper nog gebruikt.

- QuickTime File Format(.mov or .qt): In het QuickTime formaat is de basis gelegd voor het MPEG-4 bestandsformaat (.mp4) en later het meer algemene ISO Base Media File Format dat gezien kan worden als een allesomvattend formaat dat al deze varianten zoals MP4 kan bevatten. Dit ISO Base Media Format vormde later ook de basis voor 3GP en het Motion JPEG2000 bestandsformaat.

- Matroska (MKV): MKV is een nieuwe-generatie open containerformaat dat geen beperkingen oplegt naar het audio- en videoformaat dat het kan bevatten. Bijgevolg kan bijna elk audio en videoformaat in een MKV-container geplaatst worden.

- MPEG-2 Transport Stream (MPEG-TS): In tegenstelling tot voorgaande formaten is het MPEG-TS formaat vooral bedoeld om aan live video streaming te doen. MPEG-TS wordt veelvuldig gebruikt in audio en video broadcasting systemen zoals bij distributie over satelliet, kabel, antenne, IPTV en optische dragers zoals DVD en Blu-Ray. Dit formaat kan meerdere programma's bevatten en ook verdere informatie zoals een programmagids, teletext en dergelijke. Ondersteuning voor meerdere

programma's werd toegevoegd omdat systemen die bijvoorbeeld via satelliet of kabel werken een bepaalde bandbreedte ter beschikking stellen per kanaal. Op deze analoge kanalen geraakten meerdere digitale kanalen waardoor er meerdere digitale uitzendingen in 1 enkele transportstroom gebundeld moesten worden. Bij een broadcasting systeem is het belangrijk dat de gebruiker op eender welk moment kan inschakelen in de videostroom zodat er genoeg herhaling moet zijn van informatie zoals de hoogte en de breedte van het beeld, de framerate en dergelijke. Dit is in tegenstelling tot bestandsformaten waarbij deze informatie slechts 1 keer in het begin van het bestand wordt aangegeven.

- Ogg: Dit is een open containerformaat van de Xiph.org Foundation waarin voornamelijk Vorbis audio en Theora video geëncapsuleerd wordt.

- Material Exchange Format (MXF): Dit is een door SMPTE gestandaardiseerd formaat dat je vooral zal tegenkomen in een professionele video-omgeving zoals in een montageruimte. Voor professioneel gebruik vertoonden andere formaten tekortkomingen waardoor er een nieuw containerformaat werd ontwikkeld.

- WebM: WebM is een open containerformaat ontwikkeld door Google om video in HTML5 te ondersteunen. Door het streven naar open formaten is dit formaat ontwikkeld om vooral VP8-en VP9-video te bevatten gecombineerd met Vorbis audio.

6 Opgaven

In de volgende secties worden een aantal opgaven en vragen geformuleerd. Het is de bedoeling dat de antwoorden verwerkt worden in een document waarin duidelijke verwijzingen worden opgenomen naar de nummering van de opgave. De vormgeving van het document is vrij, maar het formaat van het ingediende document moet een MS Word- of PDF-document zijn. Dit laatste kan gecreëerd worden met LaTeX 2_ε, CutePDF of een recente versie van MS-Word.

Als algemene richtlijn geldt dat de antwoorden best zo bondig mogelijk gehouden worden (zonder afbreuk te doen aan de correctheid en volledigheid ervan, uiteraard). Zorg verder voor een duidelijk gestructureerd document. Geef bij de beschrijving van de uitgevoerde testen ook steeds het volledige commando dat gebruikt werd.

Vermeld uw naam en groepsnummer in het verslag.

6.1 Inhoud van een container

We hebben een systeem waarop we multimediabestanden kunnen afspelen, maar we weten niet welke compressiealgoritmen er allemaal ondersteund worden door het toestel. Je kan niet afgaan op de specificaties die meegeleverd worden. Na wat experimenteren, zien we dat er 4 types bestanden zijn die we perfect kunnen afspelen. Van deze bestanden weten we dat er voldoende functionaliteit in ons toestel aanwezig is om deze af te spelen. Dit probleem doet zich al eens voor bij het afspelen van media op smart-TVs, spelconsoles, smartphones, draagbare mediaspelers en professionele playback toestellen uit de broadcast wereld.
Deze 4 mediabestanden kunnen afgespeeld worden:
- ElFuente2b.avi
- ElFuente2b.webm
- ElFuente2b.mp4
- ElFuente2b.mkv

- Maak een zeer beknopte tabel met voor elk multimediabestand het containerformaat en de compressiestandaard die gebruikt werd voor audio en video? (voor hulp, zie Appendix in sectie 6.6.4)

Bekijk nu satellite.ts. Dit is een transportstroom geëxtraheerd van een HOT BIRD satelliet op 13° Oost. Meer specifiek hebben we onze satellietontvanger ingesteld met volgende gegevens:
- HOT BIRD (13° East)
- Frequency: 10727 MHz
- Polarization: Horizontal
- Symbol rate: 30 Mbaud/s
- DVB-S2, 8PSK, FEC 3/4

- Vat de gegevens van de verschillende programma's samen in een tabel. Vermeld voor elk programma, de verschillende stromen en voor elke stroom zeker audio en video standaard, bitsnelheid, video resolutie en audio taal indien dit af te leiden is uit de probe informatie.
- Probeer elk programma in de transportstroom te bekijken en elk audiokanaal te beluisteren aan de hand van de meegeleverde mediaplayers. Wat is volgens jou de functie van de verschillende geluidskanalen?
- Kan je elk videoprogramma bekijken? Geef een mogelijke verklaring.
- extra - Bedenk een situatie waarin een programma meerdere videokanalen zou bevatten en slechts 1 audiokanaal.

6.2 Een container samenstellen

In dit deel van de opgave gaan we multimediabestanden maken die compatibel zijn met de geanalyseerde afspeelapparatuur uit de vorige opgave. Zet dus aan de hand van ffmpeg het videobestand ElFuente.mkv om naar een formaat dat zeker afgespeeld kan worden (dus zowel het containerformaat als de compressiestandaarden moeten dezelfde zijn als in de vorige opgave). Hou rekening met videostandaard, audiostandaard, een videobitrate van 4Mbps en een audiobitrate van 128kbps.

```
ffmpeg -i input -c:v videocompressiebibliotheek
-c:a audiocompressiebibliotheek
-b:v videobitrate -b:a audiobitrate output
```

Rapporteer de commandolijnen zodat wij zelf de volgende output kunnen genereren:
- ElFuenteb.avi
- ElFuenteb.webm
- ElFuenteb.mp4
- ElFuenteb.mkv
Lever enkel de commandolijnen aan in het verslag.

De volgende twee commando's resulteren beiden in een container met daarin een videostroom volgens de MPEG-4 Part-2 standaard. Wat doen de volgende twee commando's dan verschillend?

```
ffmpeg -i input -c:v mpeg4 -vtag xvid output
ffmpeg -i input -c:v libxvid output
```

6.3 Hercoderen tegenover stream copy

In de vorige opgave zal je misschien gemerkt hebben dat het aangeleverde bestand El-Fuente.mkv gebruik maakte van H.264/AVC videocompressie en van FLAC audiocompressie. Bij het genereren van de 4 resultaatbestanden zal je ook gezien hebben dat 1 van hen H.264/AVC gebruikt. Als dit het geval is, dan zou het handig zijn moest de video niet volledig gehercodeerd (H.264/AVC → decoderen → ruwe video → encoderen → H.264/AVC) worden, maar gewoonweg overgezet worden van de ene container in de andere. Dit overzetten zonder te decoderen en te encoderen heet "stream copy". Probeer aan de hand van een internet zoekmachine informatie te vinden om deze optie in te schakelen in ffmpeg.

- Geef dus nu de 4 commando's zoals in voorgaande opgave, maar nu aan de hand van stream copy van ofwel het video gedeelte of het audiogedeelte indien mogelijk. In de andere gevallen kopieer je gewoon het resultaat uit vorige opgave.

- Wat zijn de voordelen van stream copy tegenover hercoderen?

1) Met betrekking tot computationele complexiteit. Waarom?

2) Met betrekking tot videokwaliteit. Waarom?

6.4 Ondertitels

In deze opgave gaan we ondertitels toevoegen aan een mediafragment, namelijk het ElFuenteTalk.mp4 media fragment.

6.4.1 Aanmaken van een ondertitelbestand

Maak eerst een ondertitelbestand aan gebruik makende van het onderstaand ondertitelformaat (ElFuenteTalk.srt; SubRip text bestandsformaat):

1. Begin elke ondertitel met zijn nummer (een teller).
2. Plaats de tijdsduur van het weergeven en het verwijderen van de ondertitel.
3. Plaats hier de tekst die weergegeven moet worden.
4. Eindig alle tekst met een nieuwe lijn.

Voorbeeld:

```
1
00:00:05,440 --> 00:00:07,375
Senator, we're making
our final approach into Coruscant.
```

```
2
00:00:09,476 --> 00:00:15,501
Very good, Lieutenant.
```

Merk op dat niet alle gesproken tekst als ondertitel aanwezig moet zijn, ongeveer drie zinnen zijn voldoende.

Speel nu het bestand ElFuenteTalk.mp4 af met MPC-HC.
- Worden de ondertitels weergegeven? Hoe komt dit en wat is het verschil met de hierop volgende methoden?

6.4.2 Ondertitels toevoegen aan het mediabestand

Er zijn 2 mogelijkheden om ondertitels toe te voegen aan een media bestand, namelijk:

```
ffmpeg -i ElFuenteTalk.mp4 -vf subtitles=ElFuenteTalk.srt
ElFuenteTalk_sub.mp4
```

```
ffmpeg -i ElFuenteTalk.mp4 -sub_charenc CP1252
-i ElFuenteTalk.srt -map 0:v -map 0:a -c copy -map 1
-c:s:0 mov_text -metadata:s:s:0 language=eng
ElFuenteTalk_sub2.mp4
```

Bekijk beide bekomen bestanden met ffprobe.
- Beschrijf kort het verschil dat je opmerkt?

Bekijk beide bestanden nu met MPC-HC.
- Kan je bij beide bestanden de ondertitels afzetten?
- In welke mediastroom zitten de ondertitels bij zowel ElFuenteTalk_sub.mp4 als ElFuenteTalk_sub2.mp4?
- Welke methode is het best met betrekking tot videokwaliteitsverlies en waarom?
- Welke methode is het snelst en hoe komt dit?
- Wat wordt er gebruikt op DVD en Blu-Ray als er meerdere talen ondersteund moeten worden?

6.5 Een selectie uit een videobestand maken

Bestanden nodig: ElFuente5b.mp4

In dit deel zullen we 3 methodes vergelijken om een selectie te maken uit een videobestand. Voor elk van deze methoden zullen we de eerste 2,5 seconden van het filmpje verwijderen en de daaropvolgende 5 seconden selecteren. We zullen dus het bereik 2,5s-7,5s selecteren. De originele video (ElFuente5b.mp4) is geëncodeerd met H.264/AVC met een Intra-periode van 64 beelden. Dit wil zeggen dat er elke 64 beelden een I-beeld wordt toegevoegd. De beeldsnelheid van deze video is 59,94 fps.

1. Maak met behulp van ffmpeg een MP4-bestand dat het bereik 2,5s-7,5s uit de originele video opnieuw encodeert. Je mag hierbij de libx264 encoder gebruiken. Geef hierbij ook de bitrate van de originele sequentie mee als parameter zodat die bitrate benaderd wordt. Het geluid mag gewoon gekopieerd worden. Om het juiste bereik te selecteren kan je de -ss en -t parameters gebruiken. Let op dat de -ss parameter zeker na de [-i input] parameter gebruikt wordt.

   - Welk type beeld is het eerste beeld van de output?
   - Het hoeveelste beeld is dit uit de originele video? Hoe heb je dit gevonden?

2. Maak met behulp van ffmpeg en stream copy een mp4-bestand dat het bereik 2,5s-7,5s uit de originele video bevat.
   - Van welk type is het eerste beeld?
   - Het hoeveelste beeld is dit uit de originele video? Hoe heb je dit gevonden?

3. Copy heeft een extra parameter -copyinkf.
   - Zoek op wat er gebeurt wanneer je deze parameter gebruikt?

Maak met behulp van ffmpeg, stream copy en de copyinkf parameter een mp4-bestand dat het bereik 2.5s-7.5s uit de originele video bevat.
- Van welk type is het eerste beeld van de output?
- Het hoeveelste beeld is dit uit de originele video? Hoe heb je dit gevonden?

4. Vergelijk nu de 3 methoden door een tabel te maken met een kolom voor elke methode.
- Zeg voor elke methode (re-encode, stream-copy en copyinkf) in 1 zin wat de methode doet.
- Wat zijn de voor- en nadelen van elk van deze methoden?
- Wanneer zou je welke methode gebruiken? Geef een voorbeeld.

6.6 TEDxGhent:Automatische videomontage

6.6.1 Logo inbranden

Bestanden nodig: talk.mp4, logo_alpha.png

In een eerste oefening willen we een logo inbranden in de linker bovenhoek van het beeld. Dit 15 pixels rechts en 15 pixels onder de linker bovenhoek. Het resultaat is in volgende afbeelding weergegeven:

De grootte van het logo mag 1:1 behouden worden en hoeft dus niet herschaald te worden. De transparantie die in het logo zit moet behouden blijven in het eindresultaat.

Tip: gebruik filter_complex in combinatie met een overlay filter.

Zorg dat het commando het resultaat opslaat als solution_logo.mp4.

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

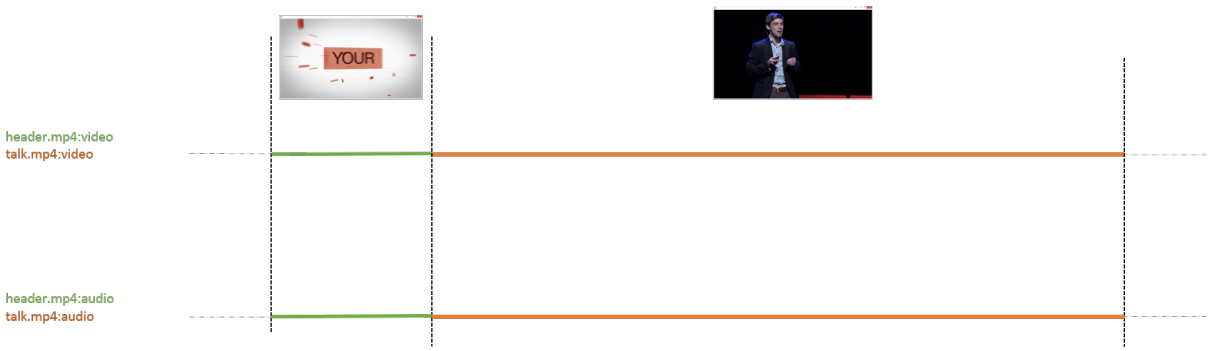
6.6.2 Toevoegen van een introductievideootje

Bestanden nodig: talk.mp4, header.mp4

In een tweede oefening moeten 2 verschillende videobestanden gecombineerd worden tot 1. Hierbij wordt er een extra introfilmpje voor de presentatie van de gebruiker gezet. Dit als aankondiging van de spreker.

Voor deze oefening moeten beide filmpjes volledig samengevoegd worden tot 1 bestand.

In volgende afbeelding zie je hoe dit eruit moet zien:



Tip: Gebruik filter_complex in combinatie met de concat filter.

Zorg dat het commando het resultaat opslaat als solution_merge.mp4.

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

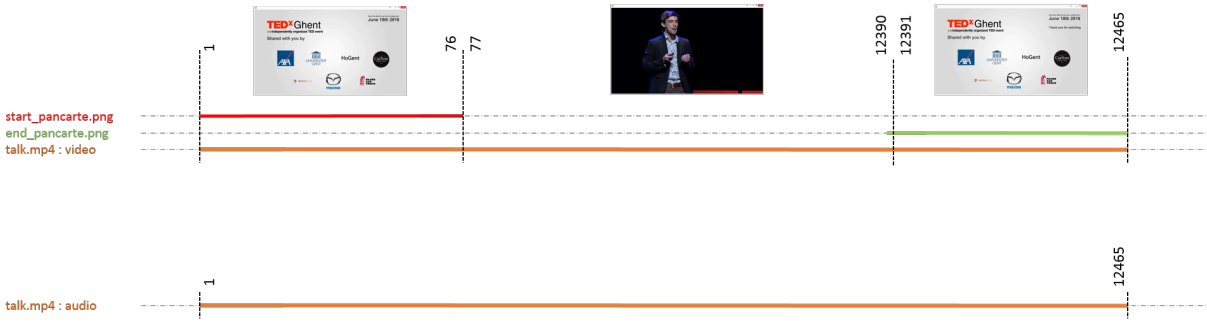
6.6.3 Delen video vervangen door stilstaande beelden

Bestanden nodig: talk.mp4, start_pancarte.png, end_pancarte.png

In een derde oefening willen we onze sponsors vermelden in de video. Hiervoor moet vlak voor het begin van de presentatie en vlak erna een pancarte (vast beeld) getoond worden. Hierbij mag de audio van het bronbestand echter niet vervangen worden.

Vertrek voor deze oefening terug van het originele talk.mp4 bestand. In dit bestand zijn vooraan en achteraan vaste placeholders voorzien om aan te tonen op welke frames de uiteindelijke pancartes moeten terecht komen. (Frame 1 t.e.m. 76 en frame 12391 t.e.m. 12465)

In volgende afbeelding vind je de exacte framenummers tot waar de pancartes zichtbaar moeten zijn.



Tips:

- Begin met het samenstellen van de video en laat audio buiten beschouwing tot het einde van deze oefening.
- Alle content heeft een vaste framerate van 25 beelden per seconde. Dit kan nodig zijn om frameaccurate berekeningen te maken. Het eindresultaat moet ook 25 frames per seconde zijn.
- Indien ffmpeg fouten geeft i.v.m. sar en/of dar (aspect ratios), bekijk dan de setsar en/of setdar filters en voeg deze toe.
- Het geluid kan makkelijkst toegevoegd worden door de talk.mp4 stream nogmaals als input mee te geven en het geluid daaruit rechtstreeks te mappen op de output.

Zorg dat het commando het resultaat opslaat als solution_pancarte.mp4

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.

6.6.4 Integratie

Bestanden nodig: header.mp4, start_pancarte.png, talk.mp4, logo_alpha.png, end_pancarte.png

In voorgaande oefeningen werd steeds een nieuwe encodering gedaan wat kwaliteitsverlies veroorzaakt. In deze oefening willen we bovenstaande oefeningen integreren in 1 enkele commandolijn.
Vereisten:

1. Video start met een header (header.mp4) (zowel audio als video)
2. Video gaat verder zoals beschreven in opgave 3
3. Tussen de 2 pancartes (dus niet tijdens de header en de pancartes) willen we een ingebrand logo zien.

Sla dit resultaat op als solution_total.mp4

- Geef de volledige commandolijn (1 lijn) zodat we dit commando zelf kunnen evalueren.
- Geef een tabel met voor elke parameter een beknopte uitleg waarom je die hebt toegevoegd en waarom op die positie. Dit zou overeen moeten komen met de uitleg die je zou geven aan iemand die verder zou moeten werken op dit commando.

Appendix A: Analyse van video

Soms willen we te weten komen wat de eigenschappen zijn van mediabestanden. Om een globaal overzicht te krijgen van wat er in een multimediabestand zit, kan je ffprobe gebruiken:

```
ffprobe input
```

Willen we meer details, dan kan er informatie over de verschillende programma's gevraagd worden:

```
ffprobe -show_programs input > output
```

Om deze informatie gemakkelijk te analyseren kan je de output van dit commando best uitschrijven naar een tekstbestand (output).

Voor meer details over de mediastromen in de programma's kan het volgende commando helpen:

```
ffprobe -show_streams input > output
```

Een mediastroom wordt opgebouwd uit paketten:

```
ffprobe -show_packets input > output
```

```
[PACKET]
codec_type=video
stream_index=0
pts=45183
pts_time=0.753050
dts=42180
dts_time=0.703000
duration=1001
duration_time=0.016683
convergence_duration=N/A
convergence_duration_time=N/A
size=275
pos=50623
flags=_
[/PACKET]
```

Interessante bevindingen kunnen gedaan worden aan de hand van de "Presentation timestamp"(pts_time) en de grootte van elk pakket (size).

Van elk beeld kan je ook het beeldtype (I, P, B) te weten komen. Deze detailinformatie kan je verkrijgen met het volgende commando:

```
ffprobe -show_frames input > output
```

```
[FRAME]
media_type=video
stream_index=0
key_frame=0
pkt_pts=46184
pkt_pts_time=0.769733
pkt_dts=46184
pkt_dts_time=0.769733
best_effort_timestamp=46184
best_effort_timestamp_time=0.769733
pkt_duration=1001
pkt_duration_time=0.016683
pkt_pos=52219
pkt_size=264
width=1024
height=540
pix_fmt=yuv420p
sample_aspect_ratio=1:1
pict_type=B
coded_picture_number=5
display_picture_number=0
interlaced_frame=0
top_field_first=0
repeat_pict=0
[/FRAME]
```

Bij het inspecteren van de beelden kunnen we pakketinformatie terugvinden (pkt_pts_time en pkt_size). Interessanter is natuurlijk het type beeld (pict_type) en het hoeveelste beeld dit is in de videostroom (coded_picture_number). Bij het opvragen van informatie per beeld zal ffprobe enkel informatie geven vanaf dat de videostroom decodeerbaar is, dus vanaf het eerste I-beeld.

Willen we de individuele beelden visueel analyseren, dan zijn er technieken om deze te extraheren. Het eerste beeld van een mediabestand kan eenvoudig geëxtraheerd worden met het volgende commando:

```
ffmpeg.exe -flags2 +showall -i bestand.mp4 -vframes 1
methodex-beeld1.bmp
```

Willen we de eerste 100 beelden als foto bekijken dan kan het volgende commando helpen:

```
ffmpeg.exe -flags2 +showall -i bestand.mp4 -vframes 100
methodex-beeld%d.bmp
```

Appendix B: Transport protocollen

In de praktijk wordt video slechts op 2 manieren verspreid over het Internet:Hypertext Transport Protocol (HTTP) of Real-time Transport Protocol (RTP). De keuze van 1 van deze technieken is volledig gebaseerd op de vertraging die de video mag oplopen wanneer deze naar de gebruiker wordt gestuurd.

Als de vertraging van minder belang is zoals in een video-on-demand dienst, dan zal er eerder geopteerd worden voor een op TCP gebaseerd transport protocol zoals HTTP. Een videodienst zoals degene die we hier aan het ontwikkelen zijn, zal dus van het HTTP protocol gebruik maken. In de praktijk komt dit neer op het opslaan van een videobestand op een webserver.

Als er directe communicatie nodig is en dus de vertraging van belang is, dan schakelt men meestal over op UDP gespecialiseerde protocollen zoals RTP. Door gebruik te maken van UDP kan de vertraging tussen zender en ontvanger laag gehouden worden (omdat er niet gewacht moet worden op hertranmissies), maar dit ten koste van de betrouwbaarheid van de verbinding.

Appendix C: Applicaties

Om wat meer context te geven rond de configuraties gebruikt voor het verspreiden van video, worden er 2 applicaties toegelicht, namelijk DASH en WebRTC. Beide technologieën komen veelvuldig voor op het net en verdienen daarom wat meer aandacht.

C.1 Dynamic Adaptive Streaming over HTTP (DASH)

Videodiensten zoals Netflix en Youtube maken gebruik van adaptive streaming van video. Dit wil zeggen dat er verschillende bandbreedteversies van dezelfde videocontent worden aangeboden waaruit de toepassing van de eindgebruiker kan kiezen. Van eenzelfde video

wordt bijvoorbeeld een versie gemaakt van 5Mbps, 2Mbps, 1Mbps en 500kbps. Afhankelijk van de snelheid van de verbinding van de eindgebruiker zal dus een zo hoog mogelijke bandbreedte geselecteerd worden.

Moest 1 van deze versies gekozen worden afhankelijk van de capaciteit op dat moment, dan zou bij eender welke verandering van bandbreedte een slechte gebruikservaring aangeboden worden. Bij een daling in bandbreedte zal er constant gebufferd moeten worden en bij een stijging kijkt de eindgebruiker naar een lagere kwaliteit dan wat er mogelijk is.

Om dit probleem op te lossen is adaptive streaming uitgevonden. Met deze techniek wordt het multimediabestand opgesplitst in segmenten, meestal van rond de 10 seconden. Elk segment mag geen gebruik maken van informatie uit het vorige segment en moet dus starten met een I-beeld. Door dit principe toe te passen is de toepassing in staat om elke 10 seconden aan te passen aan de variërende beschikbare bandbreedte. Door een goede buffermanagement toe te passen kan de toepassing dus een soepele vertoning van het multimediabestand garanderen zonder eindeloos te moeten bufferen bij een foute keuze in het begin.

Om de toepassing van de eindgebruiker in staat te stellen om op een automatische manier een overzicht te krijgen van de verschillende keuzemogelijkheden is een bestand noodzakelijk waarin een overzicht wordt gegeven van de beschikbare versies. Dit is een manifest bestand en een voorbeeld wordt hier gegeven:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:dash:schema:mpd:2011"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
  type="static"
  mediaPresentationDuration="PT654S"
  minBufferTime="PT2S"
  profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">

  <BaseURL>http://example.com/ondemand/</BaseURL>
  <Period>
    <!-- English Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40.5" lang="en"
    subsegmentAlignment="true" subsegmentStartsWithSAP="1">
      <Representation id="1" bandwidth="64000">
        <BaseURL>ElephantsDream_AAC48K_064.mp4.dash</BaseURL>
      </Representation>
    </AdaptationSet>
    <!-- Video -->
    <AdaptationSet mimeType="video/mp4" codecs="avc1.42401E"
    subsegmentAlignment="true" subsegmentStartsWithSAP="1">
      <Representation id="2" bandwidth="5000000" width="1920" height="1080">
        <BaseURL>ElephantsDream_H264BPL30_5000.264.dash</BaseURL>
      </Representation>
      <Representation id="3" bandwidth="2000000" width="1280" height="720">
        <BaseURL>ElephantsDream_H264BPL30_2000.264.dash</BaseURL>
      </Representation>
      <Representation id="4" bandwidth="1000000" width="720" height="576">
        <BaseURL>ElephantsDream_H264BPL30_1000.264.dash</BaseURL>
      </Representation>
      <Representation id="5" bandwidth="500000" width="720" height="576">
        <BaseURL>ElephantsDream_H264BPL30_0500.264.dash</BaseURL>
      </Representation>
    </AdaptationSet>
  </Period>
```

</MPD>

In deze versie van de ElephantsDream sequentie is 1 audioversie aanwezig en 4 videoversies waaruit gekozen kan worden.

C.2 WebRTC

WebRTC is een verzameling protocollen die gebruikt worden om ondersteuning in de browser te voorzien voor directe videocommunicatie (zonder communicatieserver) tussen mensen onderling.

Omdat WebRTC is ontwikkeld met communicatie in het achterhoofd is het logisch dat er op UDP gebaseerde technologieën gebruikt worden en dus ook RTP. Dit door Google samengesteld pakket aan bibliotheken bevat technologieën om veel voorkomende technische problemen rond rechtstreekse UDP-communicatie tussen mensen onderling te vereenvoudigen. Het grootste probleem met zo een communicatie is de aanwezigheid van lokale LAN netwerken die achter een Network Address Translation (NAT) zitten. UDP poorten worden vaak geblokkeerd en de vertaling naar een lokaal IP-adres verloopt vaak met moeilijkheden. Daarom zijn in WebRTC de nodige protocollen toegevoegd om deze problemen zo goed mogelijk op te lossen.