TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAMI AHOVAINIO

HEVC-VIDEON HAJAUTETTU PAKKAAMINEN

Kandidaatintyö

Tarkastaja: Sakari Lahti
16. tammikuuta 2018

TIIVISTELMÄ

SAMI AHOVAINIO: HEVC-videon hajautettu pakkaaminen
Tampereen teknillinen yliopisto
Kandidaatintyö, 16 sivua
Tammikuu 2018
Tietotekniikan kandidaatin tutkinto-ohjelma
Pääaine: Ohjelmistotekniikka
Tarkastaja: Sakari Lahti

Avainsanat: HEVC, video, videon pakkaaminen, hajautus

Videodataa täytyy pakata, jotta sitä voidaan siirtää ja tallentaa järkevästi. Videonpakkaamisalgoritmit muuttuvat jatkuvasti tehokkaammiksi ja kompleksisemmiksi. Jotta videota pystyttäisiin pakkaamaan tehokkaammin samassa ajassa, vaaditaan enemmän konetehoja. Tässä työssä tutkitaan mahdollisuutta hyödyntää internet yhteyttä, ja tämän avulla useampia koneita, hajautetussa videon pakkaamisessa.

SISÄLLYSLUETTELO

1. JOHDANTO 1
2. TEORIAA VIDEON PAKKAAMISESTA 2
2.1 Raakadata ja sen käsittely 2
2.2 Videon pakkaamien 2
2.3 Pakattu data ja sen käsittely 3
2.4 Hajautettu pakkaaminen 4
3. MITTAUSJÄRJESTELYT 7
3.1 Käytössä olevat ohjelmisto ja laitteisto 7
3.2 Saadut mittaustulokset 8
4. TULOKSIEN TULKINTA 14
5. YHTEENVETO 15
LÄHTEET 16

LYHENTEET JA MERKINNÄT

CTU	Coding Tree Unit
FHD	Full High Definition eli 1920 x 1080 -resoluutio
FPS	Frames Per Second
HEVC	High Efficiency Video Coding eli H.265 standardi
NAL	Network Abstraction Layer
PSNR	Peak Signal-to-Noise Ratio
QP	Quantization Parameter
RGB	Red Green Blue -videoformaatti
UHD	Ultra High Definition eli 3840 x 2160 -resoluutio
WZ	Wyner–Ziv

1. JOHDANTO

Videokuvaa siirrellään internetin yli jatkuvasti ja sitä on tallennettuna valtavia määriä erinäisiin toistopalveluihin. Jotta tällaisia määriä videota pysytään siirtelemään ja tallentamaan järkevästi, pitää videota pakata vähemmän tilaa vievään muotoon. Pakkaamatonta videota kutsutaan raakavideoksi ja sen koko on suoraan verrannollinen videon resoluutioon. Datan koosta puhuttaessa perusyksikkönä tässä työssä on tavu, eli 8 bittiä. Tavallisesti raakavideo on kooltaan 1,5-3 kertaa pikselien kokonaismäärä, eli jokaista pikseliä kohden datassa on 1,5-3 tavua formaatista riippuen. Tämä tarkoittaa, että pienimmillään Full HD tarkkuudella (1920 x 1080) raakavideossa jokainen kuva vie noin 3,1 MB tilaa. Jos videota halutaan toistaa 30 kuvaa sekunnissa, tarkoittaa tämä sitä, että videon kooksi tulee 93,3 MB/s. Tällaisen datamäärän siirtäminen ei ole järkevää, eikä se monissa tapauksissa ole edes mahdollista. Tästä syystä videodataa pakataan vähemmän tilaa vievään muotoon.

Videon laatu- ja resoluutiovaatimusten kasvaessa videon tehokas pakkaaminen tulee jatkuvasti tärkeämmäksi ja myös kompleksisemmaksi. Videonpakkausalgoritmit vaativat yhä enemmän laskentatehoa, jotta niillä pystyttäisiin pakkaamaan videota tehokkaammin videon laadun kuitenkaan kärsimättä.

Laskentatehon lisäämiseksi on kehitelty muiden muassa erilaisia hajautetun pakkaamisen menetelmiä. Tässä tutkimuksessa selvitetään internet yhteyden hyödyntämistä laskentatehon lisäämiseksi. Käytännössä osa raakadatasta lähetetään internetin yli toiselle tietokoneelle pakattavaksi. Näin saadaan lisää laskentatehoa, mutta oletettavasti pullonkaulaksi muodostuu raakadatan siirtäminen lähteestä pakattavaksi. Tutkimuksessa selvitetään hajautuksella saavutettavaa nopeutusta ja suhteutetaan se videon laadun ja pakkaustehon laatuun.

Luvussa 2 käydään läpi teoriaa raakadatasta, videon pakkaamisesta, pakatusta videosta sekä hajautetusta pakkaamisesta. Luvussa 3 esitellään mittausjärjestelyt sekä mittausten tulokset. Luvussa 4 on syvällisempää pohdintaa saaduista tuloksista ja luku 5 on yhteenveto koko tutkimuksesta.

2. TEORIAA VIDEON PAKKAAMISESTA

Videon pakkaamisella videodata saadaan tiivistettyä selvästi alkuperäistä pienempään kokoon. Näin mahdollistetaan muun muassa videodatan lähettäminen internetin yli, joka pakkaamattomalla videolla olisi lähes mahdotonta. Videon pakkauksessa hyödynnetään muun muassa tietoa edellisistä kuvista, jolloin seuraavan kuvan datasta voidaan ottaa talteen vain muuttuvat osuudet ja jättää samana pysyvät osat pois.

2.1 Raakadata ja sen käsittely

Raakadataksi kutsutaan pakkaamatonta videodataa. Tässä työssä käytettävä raakadataformaatti on YUV420p, missä kuva koostuu yhdestä luma-kanavasta ja kahdesta chroma-kanavasta. Datan perusyksikkönä on tavu, jonka koko on 8 bittiä. Luma-kanavassa on kuvan kirkkausdata, ja se on kooltaan yhtä suuri kuin videon resoluutio, eli jokaista pikseliä kohden on olemassa yksi luma-arvo. Chroma-kanavissa on kuvan väridata. Värikanavat ovat punainen ja sininen.[1] Vihreän arvot lasketaan YUV-formaatissa punaisen ja sinisen kanavan avulla. Jokaista 2 x 2 luma-aluetta kohden on yksi punainen ja yksi sininen arvo. Näin ollen värikanavien koot ovat neljännes kirkkauskanavan koosta. Yhteenlaskettu koko tällä videoformaatilla on siis 1,5 kertaa resoluutio.

YUV420p-formaatin raakadata koostuu niin, että kuvat on kirjoitettu peräkkäin ja joka kuvassa aluksi on koko kirkkausdata ja sen jälkeen kaikki väridata kanavittain. Data luetellaan kuvan vasemmasta ylänurkasta alkaen vasemmalta oikealle rivi kerrallaan. Tästä syystä helpoin tapa jakaa kuva osiin on jakaa se vaakasuunnassa. Tällöin täytyy vain laskea syntyneen viipaleen koko ja lukea se data kuvasta, hypätä eteenpäin värikanavan alkukohtaan, joka on sama kuin resoluutio, lukea neljännes viipaleen koosta, siirtyä kohtaan 1,25 kertaa resoluutio ja lukea jälleen neljännes viipaleen koosta. Näin saadussa kuvaviipaleessa on mukana kuvan kirkkauskanava sekä molemmat värikanavat. Raakakuvaa pystytään siis helposti viipaloimaan, kunhan sen formaatti ja resoluutio ovat tiedossa.

2.2 Videon pakkaamien

Videon pakkaamisessa pyritään pienentämään videon tiedostokoko mahdollisimman pieneksi pitäen videon laatu mahdollisimman lähellä alkuperäistä. Videon kokoa pystytään pienentämään muun muassa kirjoittamalla kuvadataan vain muutokset edelliseen kuvaan

verrattuna, jolloin voidaan hyödyntää osia edellisestä kuvasta. Tässä työssä video pakataan High Efficiency Video Coding (HEVC) -standardin mukaisesti. HEVC-standardi tunnetaan myös H.265-standardina.

HEVC-video jaetaan 64 x 64 pikselin kokoisiin alueisiin, joita kutsutaan CTU-alueiksi (Coding Tree Unit). Kuva pakataan CTU kerrallaan.[2] Pakattaessa voidaan tehdä viittauksia pakattavaa CTU-aluetta ympäröiviin alueisiin, joista voidaan kopioida jokin osa käytettäväksi uudelleen nykyisessä CTU:ssa. Tällä tavalla pystytään vähentämään kirjoitettavan datan määrää.

Pakkaamisprosessia pystytään nopeuttamaan huomattavasti käyttämällä pakattaessa hyväksi rinnakkaisuutta. Kun ylimmän rivin kaksi ensimmäistä CTU:ta on käsitelty, voidaan aloittaa seuraavan rivin ensimmäisen CTU:n käsitteleminen.[3] Tällöin käsiteltävällä CTU:lla on tarpeeksi ympäröiviä käsiteltyjä alueita, joihin se voi viitata. Prosessi jatkuu edelleen samaan tapaan, kunnes koko kuva on käsitelty ja voidaan alkaa käsitellä seuraavaa kuvaa. Samanaikaisten rivien määrä pakattaessa riippuu pakkaajalle annetusta säiemääräargumentista sekä käytettävän koneen säiemäärästä.

Pakkaamisessa käytetään myös häviöllistä menetelmää, jota kutsutaan kvantisoinniksi. Kvantisoinnissa kuvadataa jaetaan vakiolla, jota kutsutaan kvantisointiarvoksi (QP), ja tulokset pyöristetään niin, että lopputuloksessa esimerkiksi lähellä toisiaan olevat värit tulkitaan samaksi väriksi. [2, 4] Näin pystytään hyödyntämään paremmin viitteitä viereisistä ja edellisistä kuvista, sillä pienimmät muutokset karsiutuvat kvantisoinnissa pois[4]. Pakkausohjelmalle annetaan jokin kvantisointiarvo, jota se käyttää koko prosessin ajan. Mitä suurempi annettu kvantisointiarvo on, sitä suurempi häviö kuvanlaadussa havaitaan, mutta samalla videon koko pienenee ja pakkausprosessi nopeutuu.

2.3 Pakattu data ja sen käsittely

Pakattu videodata voidaan jakaa kahteen pääryhmään: intra-kuvat, joissa ei ole hyödynnetty dataa edeltävistä kuvista ja inter-kuvat, joissa datamäärää on pystytty vähentämään myös vertaamalla nykyistä kuvaa edelliseen kuvaan, ja kirjaamalla dataan vain kuvan muutokset edelliseen verrattuna [5]. Jokainen kuva kirjoitetaan pakattaessa NAL-yksikköön (Network Abstraction Layer), jonka otsikkoon kirjoitetaan tieto muun muassa siitä, mihin kohtaan kuvaa kyseinen pakattu palanen kuuluu. Pakatun videon alussa on pääotsikko, johon on kirjoitettu muun muassa koko videon resoluutio. [2] Jotta videota voidaan pakata osissa ja myöhemmin yhdistää takaisin kokonaiseksi videoksi, täytyy näitä otsikoita hieman muokata. Pääotsikkoon kirjoitetaan pakattavan videoviipaleen resoluution sijasta koko videon resoluutio ja jokaisen viipaleen NAL-yksikköön kirjoitetaan aloituskohta suhteutettuna koottuun kuvaan. Tällöin pakattu video voidaan kirjoittaa tiedostoon

viipaleet peräkkäin oikeassa järjestyksessä ja saadaan tuloksena alkuperäinen video pakattuna.

HEVC-video koostuu 64 x 64 palasista, mikä rajoittaa sitä, miten kuva voidaan jakaa viipaleiksi. Se, että kuva jää reunasta vajaaksi, eli esimerkiksi leveys ei ole 64:llä jaollinen, ei haittaa. Sen sijaan, jos kuvan pitäisi jatkua vielä vajaan palikan jälkeen, eli jos viipaleiden rajalla olevat palikat eivät ole kokonaisia, rikkoutuu vajaan palikan alapuolinen osa videosta. Tällöin viipaleiden väliin jää tyhjää tilaa, sillä NAL-yksikköön kirjoitettu aloituskohta on aina suhteutettuna 64 x 64 osioihin. Jos siis jossakin keskellä kuvaa oleva viipale ei ole 64:llä jaollinen, siirtyy seuraavan viipaleen aloituskohta lähimpään 64 jaolliseen lohkoon. Viipaleen kuva pysyy kuitenkin muuttumattomana, jolloin kuvassa havaitaan vain rako viipaleiden välissä. Tästä syystä viipaleiden korkeuksien tulee olla 64:llä jaollisia, lukuun ottamatta viimeistä viipaletta, johon jää loput datasta. Pakkausohjelma asettaa rajoitteeksi resoluution jaollisuuden 8:lla.

2.4 Hajautettu pakkaaminen

Hajautettu videon pakkaaminen, englanniksi Distributed Video Coding (DVC), ei ole aiheena uusi. Aiheesta on jo olemassa joitakin tutkimuksia. Esimerkiksi K. S. Geetha et.al. tutkimuksessaan lähestyivät ongelmaa vähentämällä videon pakkauspuolen kompleksisuutta ja siirtämällä sitä videon purkupäähän. Näin onnistuttiin nopeuttamaan pakkaamisprosessia ja vähentämään pakkaajan kompleksisuutta. [6] Artikkelissa Low complexity distributed video coding V. K. Kodavalla et.al. lähestyivät ongelmaa myös vähentämällä pakkauspuolen kompleksisuutta siirtämällä liikkeenarviointialgoritmin videon purkupäähän [7]. Mengyao Sun et.at. esittelivät tutkimuksessaan, kuinka vastaanotin, joka saa paremman laatuista videota, voi tehdä yhteistyötä toisen vastaanottimen kanssa ja parantaa toisenkin vastaanottaman videon laatua. [8] Myös B. Girod et.al. artikkelissaan lähestyivät hajautettua pakkaamista siirtämällä kompleksisuutta pakkauspäästä purkupäähän [9].

Hajautetusta videon pakkaamisesta on kirjoitettu myös yhteenveto, jossa selostetaan sen suurimmat ongelmat ja kompastuskivet. Myös tässä artikkelissa hajautettua videon pakkaamista lähestyttiin näkökulmasta, jossa pakkauspuolen prosessia pyrittiin keventää siirtämällä joitakin prosesseja pakkaamispäästä purkupäähän. [10]

Tässä työssä hajautettua pakkaamista lähestytään eri näkökulmasta. Sen sijaan, että jaettaisiin pakkaajan ja purkajan tehtäviä jotenkin eri lailla, hyödynnetään tässä työssä koneen ulkopuolisia resursseja jakamalla pakkausprosessi usealle tietokoneelle samanaikaisesti, muuntelematta varsinaisesti pakkaamisprosessia mitenkään ja täysin koskematta purkupään toimintaan. Käytännössä raakavideo viipaloidaan vaakasuunnassa, lähetetään

pakattavaksi eri tietokoneille, vastaanotetaan pakatut viipaleet ja kirjoitetaan viipaleet peräkkäin oikeassa järjestyksessä lopulliseen ulostuloon. Pakkausohjelmistoa ei ole muutettu muuten, kuin lisäämällä toiminto, joka kirjoittaa NAL-yksiköiden otsikot uudelleen vastaamaan koko videota sen sijaan, että otsikkoon tulisi viipaleen tiedot. Esimerkiksi pääotsikkoon tulee resoluutioksi kootun videon resoluutio, eikä viipaleen resoluutio ja jokaiseen otsikkoon korjataan aloituskohta vastaamaan kyseisen palasen paikkaa kootussa videossa.

Aiemmissa hajautetun pakkauksen menetelmissä ongelmiksi on muodostunut muun muassa liikkeenarviointialgoritmin siirtäminen purkupäähän sen tarkkuuden kärsimättä ja epätasainen PSNR-arvo (Peak Signal-to-Noise Ratio) avainkuvien ja Wyner–Ziv-kuvien välillä, joka aiheuttaa kuvan välkkymistä. [6, 10] Tämän työn lähestymistavan kannalta yllämainitut ongelmat voidaan välttää, mutta sen sijaan ongelmaksi muodostuu viipaleiden reunakohtien pakkaaminen. Viipaleiden reunoja pakattaessa ei voida viitata näiden reunojen yli, vaikka koko kuva jatkuisikin kyseisestä reunasta. Koska muualla kuvassa pystytään hyödyntämään viittauksia viereisistä CTU-palikoista, tulee kuvasta laadultaan epätasainen. Tämä ei kuitenkaan näy, jos käytettävä kvantisointiarvo on tarpeeksi pieni. Suurella kvantisointiarvolla viipaleiden rajat tulevat selvästi näkyviin, sillä reunat pakataan jatkuvasta intra-kuvina. Toisin sanoin reunoille ei tehdä mitään viittauksia viereisistä tai edellisistä kuvista, jolloin kvantisoinnin vaikutus ei näy niillä niin selvästi.

Pakkausprosessin nopeudessa voidaan havaita kaksi selvästi muita suurempaa tekijää. Ensinnäkin videota pakattaessa viipaleina, on sanomatakin selvää, että prosessi on välttämättä juuri niin hidas kuin hitain viipaleen pakkaus. Tämän lisäksi, koska videokuva täytyy koota uudelleen pakatuista viipaleista kirjoittamalla viipaleet tiedostoon oikeassa järjestyksessä, joudutaan joidenkin kirjoitusten kohdalla odottamaan edellisen kirjoituksen valmistumista. Näiden odotusten varalta on hajautetun pakkaamisen ohjelmaan lisätty puskureita, jolloin voidaan ottaa vastaan jo seuraaviakin kuvia ennen kuin kaikki edelliset on kirjoitettu ja kirjoittaa nämä heti ilman vastaanottamisaikoja, kun edelliset kirjoitukset ovat valmiit. Viipaleina pakkaamiseen liittyy myös se ongelma, että viipaleiden pakkaamisen kompleksisuus, ja täten aika joka käytetään jokaista viipaletta kohti, pitäisi olla suurin piirtein tasaisesti jakautunut. Näin ollen saadaan suurin mahdollinen hyöty pakkauksen hajauttamisesta.

Hajauttaessa tulee myös ottaa huomioon sekä käytettävien koneiden laskentatehot, että käytettävissä olevan internetyhteyden nopeus. Näin päästäänkin toiseen suureen tekijään, joka muodostuu pullonkaulaksi hajautuksessa, nimittäin internetyhteyden nopeuteen. Koska raakadata on kooltaan varsin valtavaa, vaaditaan sen siirtämiseen erittäin nopeaa yhteyttä. Esimerkiksi FHD videon lähettäminen raakana 30 kuvaa sekunnissa vaatii 1920*1080*1,5*30 ≈ 93MB, eli vähintään 100MB/s yhteyttä. Samaa yhteyttä myöten kulkee myös pakattu data takaisinpäin, joka myös vaatii osan kaistasta vaikkakin selvästi pienemmän osan kuin raakadatan siirtäminen.

Hajautetussa pakkaamisessa tulee siis ennen kaikkea huomioida kuvan tasalaatuisuus, käytettävissä olevat laskentatehot sekä tiedonsiirtoyhteys näiden välillä. Lisäksi tulee huolehtia viipaleiden jakaminen tasaisesti laskentayksiköiden välille niin, että viipaleiden pakkausprosessit valmistuisivat mahdollisimman samanaikaisesti.

3. MITTAUSJÄRJESTELYT

Mittaukset on toteutettu käyttäen 1-3 suurin piirtein saman tehoista tietokonetta. Tuloksissa esitetty suhdeluku x : y : z kertoo, montako viipaletta kullakin koneella on pakattu. Viipaleiden kokonaismäärä on siis x + y + z.

3.1 Käytössä olevat ohjelmisto ja laitteisto

Pakkausohjelmistona käytetään Kvazaar-ohjelmistoa [2]. Kvazaarista on tehty hieman muunneltu versio, joka kirjoittaa NAL-yksiköiden otsikot oikein viipaleiden takaisinkokoamista varten. Hajautetun pakkaamisen tehokkuutta ja kuvanlaatua verrataan samoilla parametreilla toimivaan muuttamattomaan Kvazaar-ohjelmistoon.

Hajautetun pakkaamisen ohjelmisto on edelleen kehityksessä, joten se ei ole täysin optimoitu. Ohjelmisto on kuitenkin siinä kunnossa, että sillä pystyy jo toteamaan mahdolliset hajautuksen hyödyt. Ohjelmisto koostuu yhdestä isäntä-prosessista ja n kappaleesta orja-prosesseja. Isäntä-prosessi viipaloi raakadatan osiin ja lähettää sitten nämä osat orja-prosesseille määritetyn suhdeluvun mukaisesti. Orja-prosessit vastaanottavat raakadataviipaleen, pakkaavat sen ja lähettävät pakatun kuvan takaisin isäntä-prosessille. Isäntä-prosessi vastaanottaa pakatut viipaleet orja-prosesseilta ja kirjoittaa ne sitten oikeaan järjestykseen lopulliseen tiedostoon.

Mittauksissa on käytetty kahta pöytäkonetta ja yhtä kannettavaa tietokonetta. Pöytäkoneissa käyttöjärjestelmänä on Windows 7 ja kannettavassa Windows 8.

Taulukko 1. *Mittauksissa käytettyjen tietokoneiden tiedot*

	Käyttöjärjestelmä	Prosessori	RAM (GB)
Pöytäkone 1	Windows 7 Enterprise (64 bit)	Intel(R) Core(TM) i7-4770 CPU @ 3.40 GHz	16,0 GB
Pöytäkone 2	Windows 7 Enterprise (64 bit)	Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz	32,0 GB
Kannettava	Windows 8.1 Pro (64 bit)	Intel(R) Core(TM) i7-4800MQ CPU @ 2.70 GHz	16,0 GB

Pakkausprosessissa ei käytetä GPU:ta mihinkään. Merkittävät tiedot käytettävistä koneista ovat koneiden prosessorit, käytettävissä oleva muistimäärä sekä käyttöjärjestelmä. Käyttöjärjestelmä on merkittävä, koska datan siirtämiseen käytettävät socketit toimivat hieman eritavoin käyttöjärjestelmästä riippuen. Muistilla ei sinänsä ole suurtakaan merkitystä ohjelmistolle muuten, mutta se voi rajoittaa puskurien kokoa ja määrää.

3.2 Saadut mittaustulokset

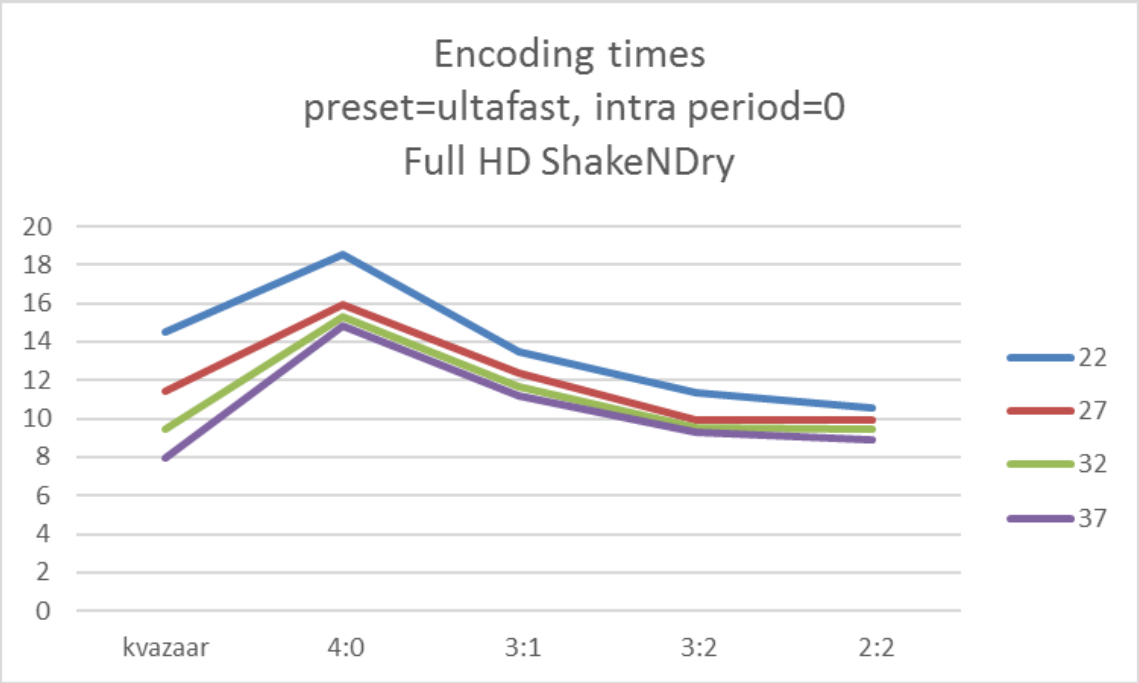
Mittaukset on suoritettu 10 testin sarjoina ja saadut tulokset ovat keskiarvoja 10 otoksesta. Taulukossa 2 on mittaustulokset testivideosta ShakeNDry 1920x1080 8bit YUV RAW [11]. Käytetyt asetukset ensimmäisessä testissä ovat: preset = ultrafast ja intra period = 0. Taulukko on jaettu QP-arvojen mukaan, jotta tuloksia on helpompi verrata keskenään. Tässä mittauksessa on käytetty kahta pöytäkonetta.

Taulukko 2. *Mittaustulokset Full HD videosta, jossa koira ravistelee itsensä kuivaksi.*

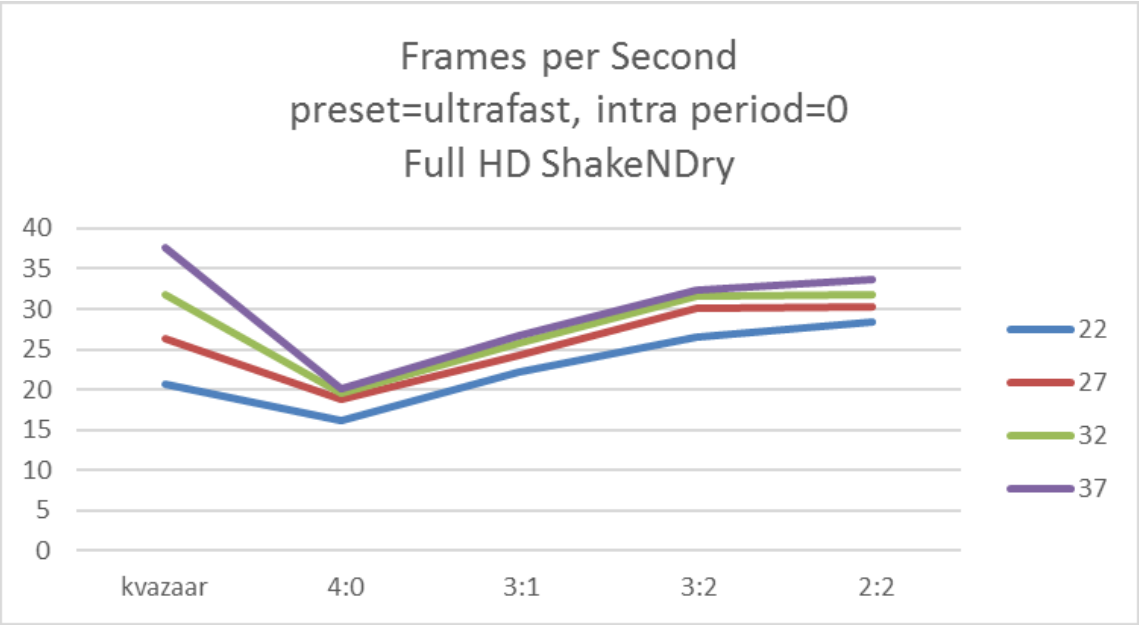
		Encoding time	Frames/second	PSNR	Size (b)
QP=22	Kvazaar	14.547	20.623	41.8	14335101
	4:0	18.534	16.188	41.74	13880893
	3:1	13.521	22.197	41.74	13880893
	3:2	11.341	26.455	41.74	13900861
	2:2	10.534	28.48	41.74	13880893
QP=27	Kvazaar	11.415	26.265	39.8	6633082
	4:0	15.967	18.791	39.72	6356878
	3:1	12.341	24.309	39.72	6356878
	3:2	9.944	30.175	39.71	6369296
	2:2	9.903	30.3	39.72	6356878
QP=32	Kvazaar	9.435	31.798	37.6	2932389
	4:0	15.316	19.588	37.45	2780395
	3:1	11.65	25.752	37.45	2780395
	3:2	9.51	31.548	37.44	2787763

	2:2	9.432	31.826	37.45	2780395
QP=37	Kvazaar	7.98	37.595	35.3	1207790
	4:0	14.8	20.166	35.15	1120708
	3:1	11.207	26.772	35.15	1120708
	3:2	9.258	32.408	35.13	1128532
	2:2	8.892	33.738	35.15	1120708

Kun taulukoidut tulokset pakkaamiseen kuluvasta ajasta ja pakkaamisen nopeudesta mitattuna kuvina per sekunti, huomataan raskaampaa laskentaa vaativan pakkauksen hyötyvän hajautuksesta selvästi enemmän. Mitä korkeampi QP arvo on, sitä kevyempää pakkauksessa oleva laskenta on.

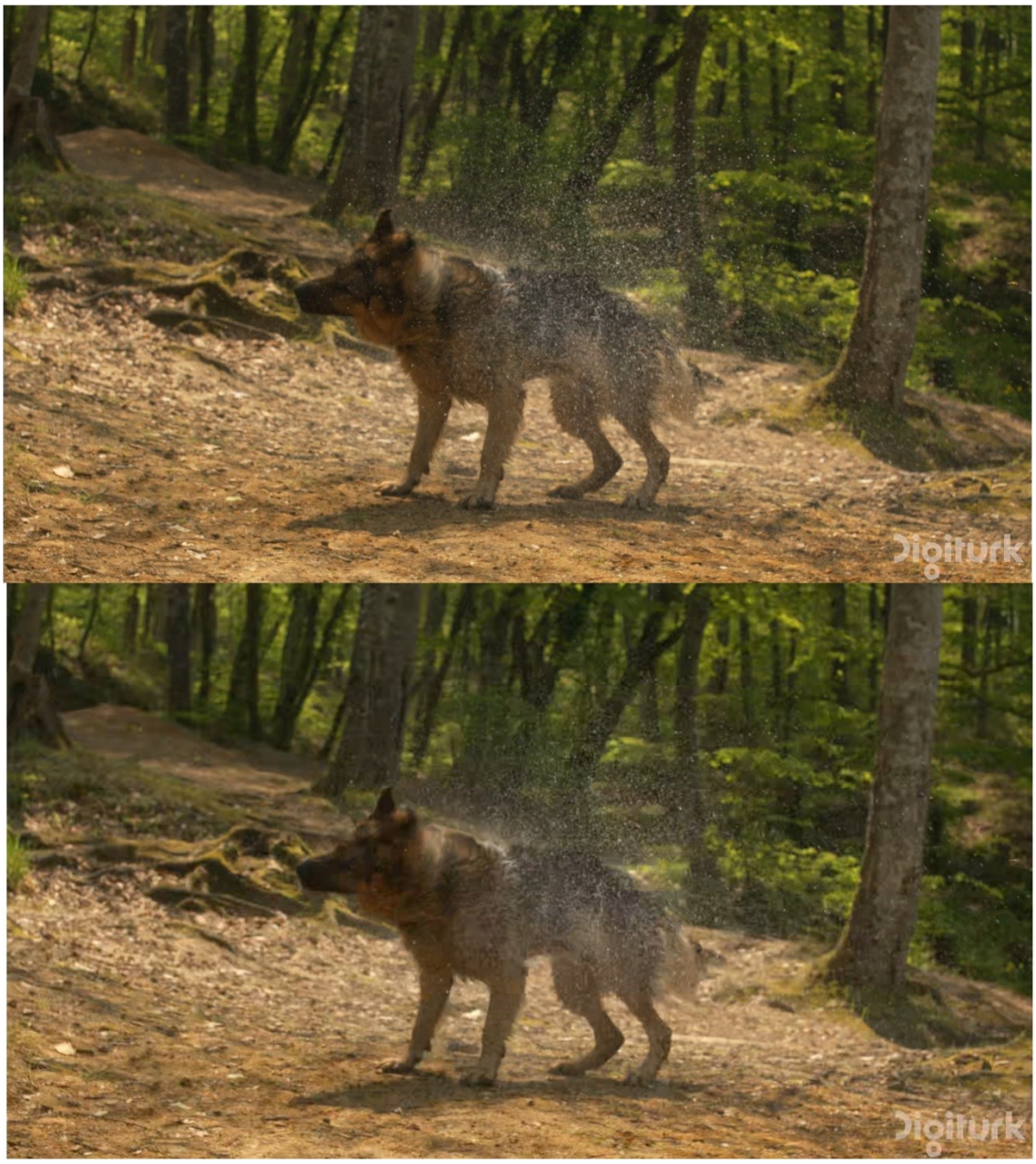


Kuva 1. *Taulukon 2 pakkausajoista tehty kaavio.*



Kuva 2. *Taulukon 2 kuvaa sekuntia kohden -arvoista tehty kaavio*

Kuten ylläolevista kuvista ja taulukosta voidaan huomata, on yhdellä koneella ajettu hajautetun pakkaamisen ohjelmisto selvästi Kvazaaria hitaampi. Optimaalisesti nämä olisivat yhtä nopeat, mutta hajautetun pakkaamisen ohjelmistoa ei ole vielä optimoitu käytännössä lainkaan.

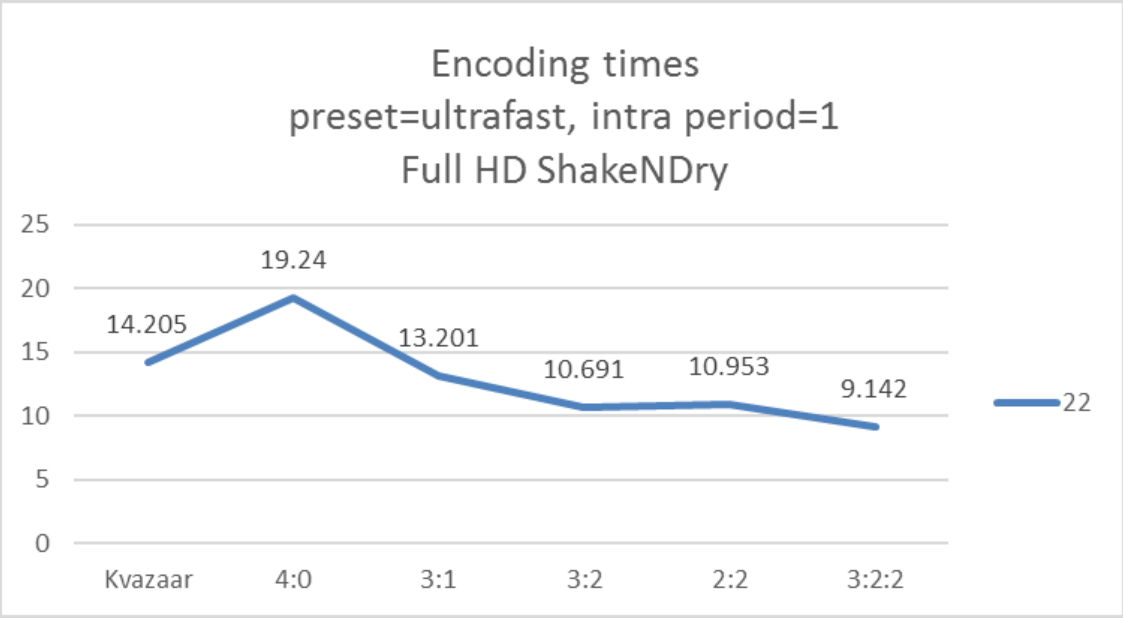


Kuva 3. *Yllä QP-arvolla 22 ja alla QP-arvolla 37 pakattu video.*

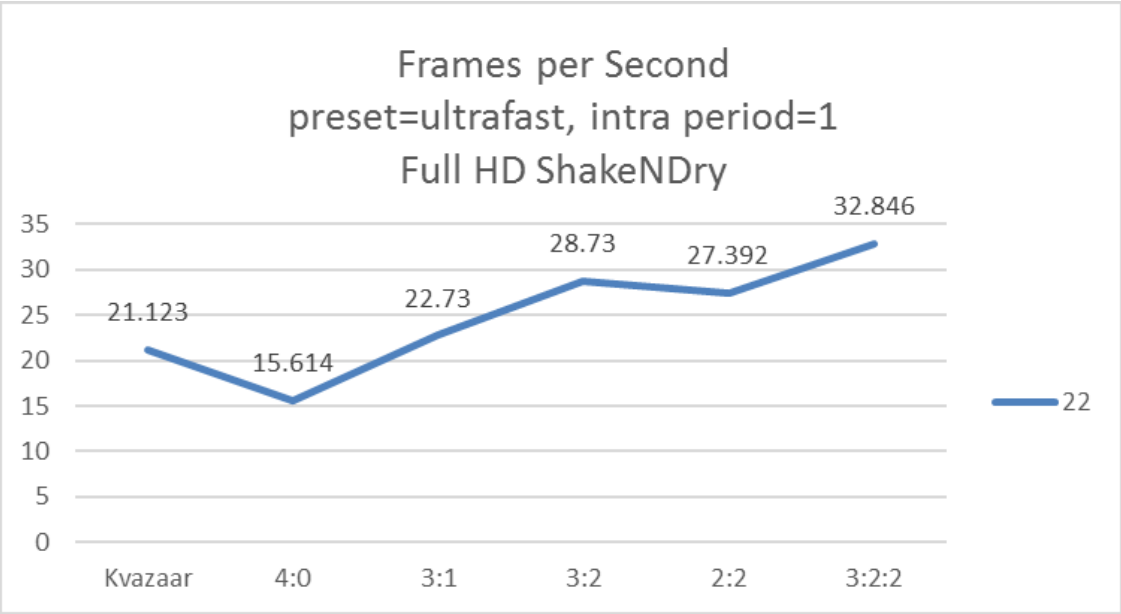
Kuten kuvasta 3 havaitaan, tulee suurella QP-arvolla viipaleiden väliin vääristymä. Nämä vääristymät kiinnittävät enemmän huomiota videossa, sillä vääristymät pysyvät jatkuvasti samassa kohtaa, kun taas muu kuva muuttuu tasaisesti. QP-arvolla 27 raja on vielä niin himmeä, ettei se pistä silmään. QP-arvolla 22 saadaan parhaat tulokset hajautuksesta ja samalla kuvanlaatu on paras valittavissa olevista. Tästä syystä tulevat mittaukset on tehty vain QP-arvolla 22.

Taulukko 3. *FHD ShakeNDry, preset = ultrafast, full intra (intra period = 1)*

	Encoding time	Frames/second	PSNR	Size (b)
Kvazaar	14.205	21.123	43.28	57872844
4:0	19.240	15.614	43.28	58043335
3:1	13.201	22.730	43.28	58043335
3:2	10.619	28.265	43.27	58093921
2:2	10.953	27.392	43.28	58043335
3:2:2	9.142	32.846	43.27	58188216



Kuva 3. *Kaavio taulukon 3 pakkausajoista.*



Kuva 4. *Kaavio taulukon 3 FPS arvoista*

Tuloksista huomataan, että parhaat tulokset saadaan antamalla paikalliseen pakkaamiseen hieman enemmän dataa, kuin muille koneille. Tämä johtuu siitä, että paikallisesti pakattavaa dataa ei tarvitse siirrellä internetin yli. Kuitenkin datamäärät kannattaa jakaa mahdollisimman tasaisesti koneiden välille. Esimerkiksi 3:2-jaossa paikalliseen pakkaamiseen jää 1920 x 576 viipale ja toiselle koneelle lähetetään 1920 x 504 viipale. Viipaleet eivät siis suoraan noudata jaon suhteita, sillä viipaleiden tulee olla 64:llä jaollisia. Tällöin viimeiseen viipaleeseen jää jonkin verran ylimääräistä, sillä siihen sisältyy aina myös videon korkeudesta 64:n jakojäännös.

4. TULOKSIEN TULKINTA

Saaduista mittaustuloksista huomataan, että hajautetun pakkaamisen ohjelmistossa on selviä parantamisen kohteita yhdellä koneella pakkaamisessa, sekä suuren QP-arvon videoiden pakkaustehokkuudessa. Näistä heikkouksista huolimatta voidaan tuloksista kuitenkin huomata selvä potentiaali hajautettujen resurssien käytössä. Useammalla koneella pakattaessa saadaan selvä nopeutus pakkaamiseen videon laadun kuitenkaan juurikaan kärsimättä. Ohjelmistoa optimoimalla voidaan jo saavutettua nopeutusta saada vieläkin suuremmaksi pitäen laatu kuitenkin samalla tasolla mitä se näissä mittauksissa on ollut.

Paikallisen, eli yhdellä koneella suoritettavan pakkaamisen nopeutta voidaan pitää hyvänä mittarina ohjelmiston nopeudelle, sillä siihen ei vaikuta käytössä olevan internetyhteyden nopeus. Todennäköisimmin pullonkaulaksi ohjelmistossa on muodostunut datan siirto, sillä datan siirtoon käytettävät keinot vaikuttavat olevan huonosti rinnakkaistuvia. Ohjelmistoa voisi nopeuttaa myös ottamalla käyttöön Kvazaarista useamman kuvan samanaikaisen käsittelyt, mutta se vaatisi myös muokkauksia datansiirtoon.

Suuren QP-arvon pakkaamisen nopeuttaminen ei ole kovin tärkeää, sillä kuvasta tulee nopeasti hyvin sekava, johtuen viipaleiden reuna-alueiden vääristymistä. Hajautetulla pakkaamisella voidaan siis sanoa olevan käyttöä korkealaatuisen videon pakkausprosessin nopeuttamisessa. Kolmella koneella pakattaessa Full HD videossa saatiin full intra pakkauksessa nopeutusta yli 10 FPS.

Saatujen tuloksien perusteella hajautetun pakkaamisen ohjelmistolla voidaan saavuttaa merkittävä nopeutus laadun juurikaan kärsimättä. Näin ollen ohjelmiston kehittämistä kannattaa jatkaa edelleen, jotta saadaan suurimmat pullonkaulat optimoitua pois. Tämän lisäksi tulee myös tutkia, pystyykö viipaleiden jakoa koneiden välille jotenkin automatisoimaan niin, että isäntäprosessi pystyy suoraan valitsemaan optimaalisen viipalejaon koneiden välille. Tätä varten täytyisi ohjelmistoon saada jokin tapa mitata käytössä olevan internetyhteyden nopeutta koneiden välillä sekä koneiden laskentatehoa.

5. YHTEENVETO

Työn mittaustulosten perusteella internet-yhteyttä voidaan hyödyntää videonpakkauksessa hajautettujen resurssien hyödyntämiseen. Full HD videota pakattaessa internet-yhteyden nopeus ei vielä muodostunut pullonkaulaksi, vaan suuremman ongelman tuotti lähetystavan huono rinnakkaistuminen. Ultra HD videossa myös internet-yhteydeltä vaaditaan selvästi enemmän, kuten myös koneiden suorittimilta. Kuitenkin Ultra HD videossakin pystytään hyödyntämään hajautettuja resursseja, kunhan hajautus suhteutetaan oikein käytettävien laskentatehojen ja internet-yhteyden nopeuden mukaan. Tulevaisuudessa videon resoluution kasvaessa voidaan olettaa myös internet-yhteyksien parantuvan sekä laskentatehon lisääntyvän. Näin ollen hajautuksella voidaan olettaa olevan kysyntää myös tulevaisuudessa.

Hajautetun pakkaamisen ohjelmistoa voidaan kehittää vielä selvästi eteenpäin. Tämä vaatii vain laajempaa tutkimusta siitä, mikä prosessissa vie eniten aikaa. Kun pullonkaulat on löydetty ja varmistettu, voidaan niitä alkaa optimoida. Sokeasti oletusten perusteella ei kuitenkaan kannata lähteä optimoimaan mitään, sillä pahimmat hidastukset ovat yleensä alueilla, joita ei tule edes ajatelleeksi.Kun hajautetun pakkaamisen ohjelmisto on saatu optimoitua riittävän hyvin, voidaan alkaa kehittää siihen tekoälyä, jolla ohjelmisto osaisi itse päätellä parhaat viipaleiden jaot.

LÄHTEET

[1] Zhan Jun Si, Fei Yuan, Yong Guang Zhao, Design of Android-Based YUV Video Player, Applied Mechanics and Materials, Vol. 731, 2015, pp. 248. https://search.proquest.com/docview/1649227687.

[2] M. Viitanen, A. Koivula, A. Lemmetti, J. Vanne, T. D. Hmlinen, Kvazaar HEVC encoder for efficient intra coding, 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1662-1665.

[3] G.J. Sullivan, J.R. Ohm, W.J. Han, T. Wiegand, Overview of the High Efficiency Video Coding (HEVC) Standard, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, Iss. 12, 2012, pp. 1649-1668.

[4] R. M. Gray, D. L. Neuhoff, Quantization, IEEE Transactions on Information Theory, Vol. 44, Iss. 6, 1998, pp. 2325-2383. Available (accessed ID: 1): .

[5] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand, Comparison of the Coding Efficiency of Video Coding Standards﷒Including High Efficiency Video Coding (HEVC), IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, Iss. 12, 2012, pp. 1669-1684. Available (accessed ID: 1): .

[6] K. S. Geetha, S. B. Hegde, H. P. Vishwas, Design and implementation of distributed video coding architecture, 2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS), pp. 39-43.

[7] Low complexity distributed video coding, in: Journal of Visual Communication and Image Representation, 2014, pp. 361-372.

[8] Mengyao Sun, Yumei Wang, Hao Yu, Yu Liu, Distributed cooperative video coding for wireless video broadcast system, 2015 IEEE International Conference on Multimedia and Expo (ICME), pp. 1-6.

[9] B. Girod, A. M. Aaron, S. Rane, D. Rebollo-Monedero, Distributed Video Coding, Proceedings of the IEEE, Vol. 93, Iss. 1, 2005, pp. 71-83.

[10] V. K. Kodavalla, Challenges in practical deployment of distributed video coding, 2016 International Conference on Microelectronics, Computing and Communications (MicroCom), pp. 1-6.

[11] Testivideot http://ultravideo.cs.tut.fi/#testsequences