Kodiranje videa prema H.265 normi

Mekić-Delić, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

Kodiranje videa prema H.265/HEVC normi

ZAVRŠNI RAD

Luka Mekić-Delić

Osijek, 2020.

SADRŽAJ

1. UVOD

Kompresija digitalnog videa je važan dio njegove distribucije, reprodukcije i arhiviranja. Čak i danas, nekomprimirana video datoteka može zauzimati memorijski prostor i zahtijevati procesorsku moć koju uobičajeno računalno sklopovlje ne može lako pružiti. Zbog toga, razvijene su norme za kodiranje/kompresiju videa kao što su H.264, odnosno AVC (eng. *Advanced Video Coding*), i H.265, odnosno HEVC (eng. *High Efficiency Video Coding*). Cilj normi za kompresiju videa je uvesti globalni standard za kodiranje videa koji će većina korisnika koristiti tako da svi mogu dekodirati i reproducirati iste video formate. Međutim, tijekom vremena pronalaze se tehnička rješenja za učinkovitiju kompresiju, pa se uspostavljaju i nove, naprednije norme kojima je namjena zamijeniti stare. H.265 je jedna takva norma, a svrha joj je zamijeniti normu H.264.

Glavni zadatak ovog završnog rada je obraditi H.265/HEVC normu za kompresiju videa, te njezina poboljšanja u usporedbi sa njenim prethodnikom H.264/AVC. Korištenjem programa FFmpeg, napisan je kôd prema [1] u Python programskom jeziku kojim su kodirane različite video datoteke H.264 i H.265 koderima, te je napravljena usporedba kvalitete komprimiranih videosekvenci.

Drugo poglavlje postavlja teorijsku podlogu za razumijevanje načina rada kodiranja videa i objašnjava značajke i alate norme H.264 na kojoj je norma H.265 zasnovana. Objašnjena su osnovna načela postupka kompresije videa na čijoj bazi su norme H.264 i H.265 osnovane, te alati i značajke H.264 norme. Treće poglavlje opisuje značajke i alate H.265 norme. Četvrto poglavlje obrađuje objektivne mjere za ocjenu kvalitete videa. Nadalje, opisuje FFmpeg kôd koji je napisan u Python jeziku, te predstavlja rezultate ocjene kvalitete kodiranih videa dobivenih primjenom opisanih objektivnih metoda. U petom poglavlju je dan osvrt na rješenja zadatka ovog završnog rada, diskutirane su prednosti i mane H.265 norme i razmatrane prakse koje bi se trebale poticati da bi se postigao pozitivan učinak u ovom području tehnologije.

2. PRINCIPI KODIRANJA VIDEA

H.265 norma je osnovana na njenom prethodniku H.264, te nasljeđuje osnovna načela rada i alate iz njega. U ovom poglavlju će se obraditi osnovna načela kodiranja videa na bazi kojih su obje norme osnovane, te značajke i alati H.264 norme gdje će se posebno istaknuti oni koje H.265 nasljeđuje.

2.1. Osnovno načelo rada

Većina modernih digitalnih video formata, što uključuje i H.265, kodira video kroz 4 osnovna moda obrade, a to su:

1. Pred-obrada
2. Prostorna kompresija
3. Vremenska kompresija
4. Kontrola protoka

Pred-obrada predstavlja osnovnu optimizaciju videa za daljnje kodiranje. Ona obuhvaća smanjenje bitnog zapisa boje (smanjivanje dubine boje), poduzorkovanje formata boje, prostorno filtriranje slike i sl. Smanjenje broja bita po kanalu i poduzorkovanje boje smanjuju rezoluciju boje i time se ostvaruje prvi korak kompresije – uvode se gubici informacije i veličina videa se znatno smanjuje. Zbog činjenice što se koristi YCbCr format boje, zadržava se luminantna komponenta, a poduzorkuju se krominantne komponente i time se gubitak na vizualnoj kvaliteti drži minimalnim – ljudski vizualni sustav je osjetljiviji na razinu svjetline nego na razinu boje. Nadalje, mogu se izvršiti metode obrade slike kao što su prostorno filtriranje i promjena intenziteta slike, smanjivanje razine šuma i sl.

Prostorna kompresija se odnosi na kodiranje okvira videa istim metodama koje se koriste za kompresiju mirnih slika (također se zove unutarokvirna kompresija). Postupak oponaša norme kodiranja slika kao što je JPEG, a osnovni princip je iskoristiti zalihosti informacije unutar slike i kreirati tzv. *I* okvire. Ne kodiraju se svi okviri videa u ovom postupku, nego samo onoliko koliko je zadano parametrima kodiranja (najčešće *keyframe rate*). I okviri se često još nazivaju i ključni okviri (eng. *keyframes*), ali nisu svi I okviri ključni. I okviri se koriste kao referentni okviri u postupku međuokvirnog kodiranja (slijedeći korak nakon prostorne kompresije), a kako kodirani I okvir sam ne ovisi o susjednim okvirima

predstavlja mjesto u videosekvenci gdje može početi dekodiranje kada npr. korisnik promijeni televizijski kanal koji gleda. Okviri se dijele na blokove određenih dimenzija npr. 8x8 koji se DCT (eng. *Discrete Cosine Transform*) ili sličnim transformacijama prebacuju iz domene prostornih intenziteta u domenu prostornih frekvencija, te se frekvencijski koeficijenti kvantiziraju i entropijski kodiraju. Dodatak tome je spremnik na izlazu s kojim kvantizacija podataka ima povratnu vezu kako bi se uspostavila kontrola brzine protoka bitova i utvrdili pogodni kvantizacijski koraci. U pravilno kodiranom videu, I okviri su najrjeđi tip okvira i imaju najniži stupanj kompresije, ali predstavljaju neizostavnu osnovu kodiranog videa na kojoj se obavljaju sve naprednije radnje.

Vremenskom kompresijom se kreiraju *P* i *B* okviri. Ovo su okviri kojima je cilj smanjiti zalihost informacija u vremenu, tj. oni koriste metode kompenzacije pokreta kako bi se pronašle slične vrijednosti intenziteta između susjednih okvira. Tijekom reprodukcije neki dijelovi okvira se mijenjaju dok ostali ostaju isti, te se ta informacija može kodirati samo jednom, referencirati kad je potrebna i tako se smanjiti količina potrebnih bitova za kodiranje. P i B okviri su kodirani okviri koji sami po sebi nemaju kompletnu informaciju o svim svojim elementima, nego ovise o prethodnim ili budućim okvirima za proračun svojih elemenata (odnosno predviđaju se iz prethodnih ili budućih okvira). P okviri su predviđeni iz prethodnih I ili P okvira, a B okviri su predviđeni iz prethodnog i sljedećeg I ili P okvira. Dva glavna koraka koje koder mora poduzeti pri kreiranju P i B okvira je izračunati vektore pokreta i okvir razlike. Oni se tada zapisuju i kodiraju, te će ih dekoder iskoristiti za dobivanje cijele slike tih okvira. Vektori pokreta se računaju tako što se pronalaze slični makroblokovi između trenutnog i prethodnog/budućeg (referentnog) okvira i izračuna se udaljenost između njih. Na temelju informacije o vektoru pomaka za dani makroblok u okviru koji se kodira, koder u postupku kompenzacije pokreta, postvalja najsličniji makroblok referentnog okvira na poziciju makrobloka koji se kodira. Primjenom takvog postupka na sve makroblokove dobije se okvir prediktor. Naposlijetku, oduzimanjem okvira prediktora od okvira koji se kodira dobiva se okvir razlike. Okvir razlike se dalje kodira istim postupkom kao I okvir, s tim da okvir razlike sadrži manje informacija od originalnog okvira pa se može bolje komprimira nego da se originalni okvir kodira kao I okvir. B okviri zahtijevaju kompleksniji proračun od P okvira jer kao referentne okvire koriste i prethodni i sljedeći okvir pa se proračun vektora pomaka radi za obje reference, ali imaju bolji stupanj kompresije.

Kontrola protoka određuje koliku količinu informacije (bitova) će kodirana video datoteka sadržati u jedinici vremena (sekundi). Ta mjera zove se kodna brzina (eng. *bitrate*), a njena mjerna jedinica je bit/s. Prema korisničkim postavkama, koder će tijekom kodiranja držati određeno ograničenje za količinu bitova koji će se kodirati. Kodna brzina je proporcionalna s kvalitetom videa, a obrnuto proporcionalna sa stupnjem kompresije videa. Time se ostvaruje jednostavna implementacija kontrole kvalitete s kojom korisnik može odabrati proizvoljan kompromis između kvalitete i kompresije videa. Postoje dvije vrste kontrole protoka, a to su: kompresija s konstantnom brzinom protoka podataka (eng. *Constant Bit Rate* – CBR), i kompresija s varijabilnom brzinom protoka podataka (eng. *Variable Bit Rate* - VBR). Pod CBR modom rada, koder nastoji zadržati jednaku kodnu brzinu kroz cijelu video datoteku – time je kvaliteta videa promjenjiva i manje zahtijevan sadržaj će imati bolju kvalitetu, ali je lako predvidjeti veličinu kodiranog videa uzimanjem u obzir zadanu kodnu brzinu i reprodukcijsko vrijeme videa. VBR način rada podrazumijeva kodnu brzinu koja je promjenjiva u vremenu, te će koder pridružiti veću kodnu brzinu zahtjevnijim sadržajima dok će manje zahtjevni sadržaji imati manju kodnu brzinu. Ovim načinom rada korisnik umjesto kodne brzine može odabrati konstantnu kvalitetu koju koder mora održati kroz video, ali je teško predvidjeti koja će biti veličina kodiranog videa. Koderi mogu raditi u CBR ili VBR načinu rada, ali postoji i opcija VBR-a s gornjom granicom kodne brzine.

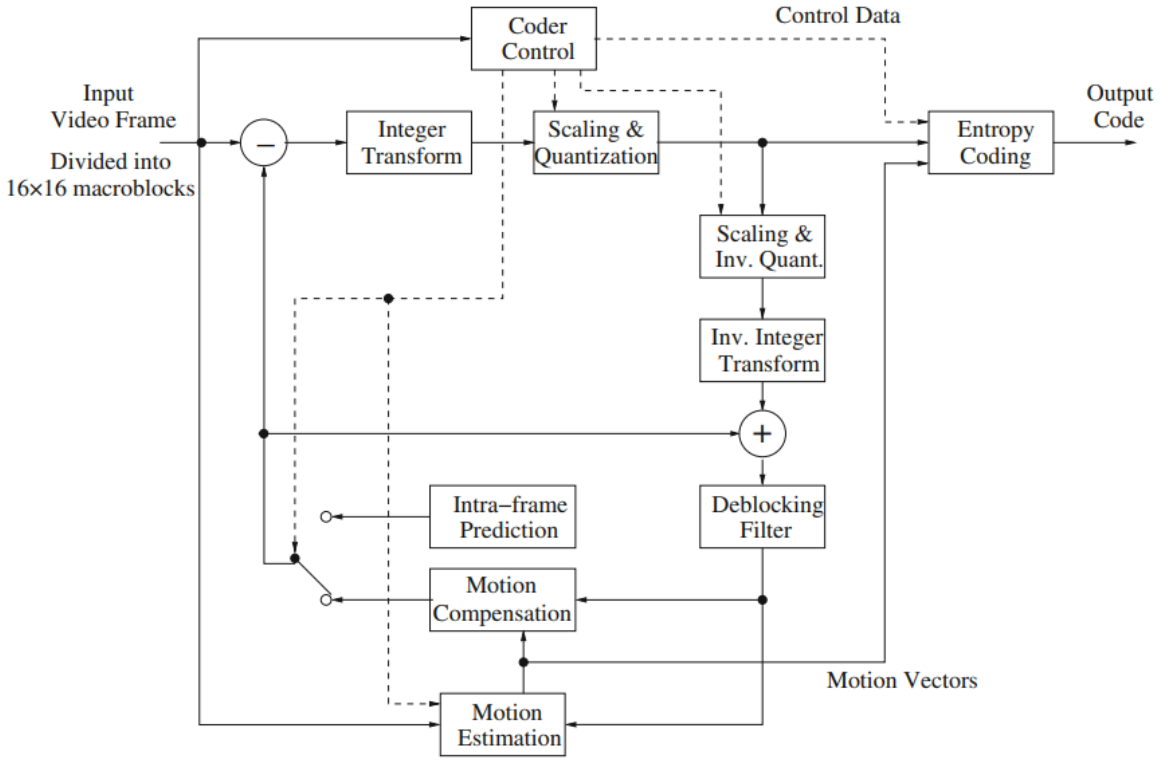
2.2. Značajke i alati norme H.264/AVC

H.264 je općeprihvaćeni, robusni standard za kodiranje videa koji podržava razne aplikacije. U odnosu na prethodne norme kao što su MPEG-2 i H.263+, ostvaruje od 30 do 50% bolju kompresiju za istu kvalitetu videa kao i poboljšanu otpornost na greške prijenosa, prema [2, str. 395]. Slično njima, H.264 kodiranje osnovano je na bazi kodiranja makroblokova dimenzija 16x16 na koje je slika podijeljena, pri tome računajući kompenzaciju pokreta i operacije transformacije. Odsječci (eng. *slices*) proizvoljnih veličina mogu grupirati više makroblokova u samostalne jedinice. Isto kao i u prethodnim normama, korištenjem različitih profila i razina pruža se jednostavno rješenje s kojom se kodiranje može prilagoditi za različite aplikacije kao što su televizija visoke definicije, mobilni uređaji, prijenos i videostrujanje preko mreže, itd.

Kada H.264 radi u VBR modu, parametar koji se koristi za odabir kvalitete zove se CRF (eng. *Constant Rate Factor*). To je logaritamska mjera kvalitete od 0 do 51 gdje niske

vrijednosti odgovaraju višoj kvaliteti, a visoke vrijednosti nižoj kvaliteti. Svakih 6 koraka označava otprilike dvostruku razliku u veličini videa, a H.264 ima zadanu vrijednost 23. Zadani format boje za H.264 je 4:2:0 s 8 bita po kanalu. Međutim, također podržava i formate 4:2:2, 4:4:4, više od 8 bita po kanalu, te monokromatsku sliku.

H.264 koristi nekoliko alata za kodiranje. Oni koje H.265 nasljeđuje uključuju cjelobrojnu transformaciju, predviđanje za unutarokvirno kodiranje, filtar za uklanjanje efekta bloka unutar petlje za predviđanje, te entropijsko kodiranje zasnovano na kodiranju varijabilne duljine prilagodljivo sadržaju (eng. *Context Adaptive Variable Length Coding* – CAVLC) i binarnom aritmetičkom kodiranju prilagodljivom sadržaju (eng. *Context Adaptive Binary Arithmetic Coding* – CABAC). Ostali alati su predikcija pokreta blokovima različitih veličina, preciznost na razini četvrtine elementa slike, te korištenje više referentnih okvira za proračun predikcije. Shema osnovnog H.264/AVC kodera je prikazana na slici 2.1.1., prema [2, str. 396]. Njegovi glavni blokovi su procjena i kompenzacija pokreta, unutarokvirna predikcija, cjelobrojna transformacija, skaliranje i kvantizacija i entropijsko dekodiranje. Za potrebe međuokvirnog kodiranja u koderu se radi dio dekodiranja već kodiranih okvira što obuhvaća inverznu kvantizaciju i transformaciju, deblokirajući filtar te pohranjivanje referentnih okvira.



Slika 2.1. *Osnovni H.264/AVC koder,* [2]

2.2.1. Cjelobrojna transformacija

H.264 koristi cjelobrojnu transformaciju za kodiranje okvira razlike dobivenih nakon proračuna predikcije. Ova transformacija se koristi umjesto DCT-a zbog nedostataka DCT transformacije. DCT u prethodnim normama za kodiranje videa uzrokuje pomak predikcije zbog greški u zaokruživanju decimalnih brojeva tijekom pripadnih operacija, te zahtijeva relativno veliku procesnu snagu jer se izvršava veliki broj operacija množenja decimalnih brojeva. H.264 blokovi su izuzetno osjetljivi na pomak predikcije zbog mnogih modova predikcije što norma omogućava, te različitih veličina blokova za proračun predikcija. Zbog mnoštva unutarokvirnih i međuokvirnih predikcija, greške zaokruživanja decimalnih brojeva se skupljaju, rezultirajući još većim greškama koje negativno utječu na svaki uzastopni blok koji se kodira.

Prostorna korelacija između elemenata slike okvira razlike je vrlo mala kod H.264 zbog moćnih implementacija predikcije, stoga se za kompresiju koristi jednostavna aproksimacija cjelobrojne preciznosti 4x4 DCT-a i IDCT-a (inverzna DCT). S njome se isključivo provode cjelobrojne 16-bitne operacije. Cjelobrojna aritmetika dopušta inverzne transformacije na svim procesorima čime se sprječava odstupanje rezultata kodera i dekodera kao što se događalo u prethodnim normama. DCT transformacija u dvodimenzionalnom prostoru se može realizirati s dvije uzastopne jednodimenzionalne transformacije – prvo u vertikalnom smjeru te u horizontalnom smjeru. To se može postići sa sljedećom formulom, iz [2, str. 400]:

$$\mathbf{F} = \mathbf{T} \cdot \mathbf{f} \cdot \mathbf{T}^{\mathbf{T}}, \tag{2-1}$$

gdje je:

- **F** – rezultirajuća matrica,
- **T** – cjelobrojna matrica koja aproksimira 4x4 DCT,
- **f** – matrica ulaznih podataka.

Da bi se dobila cjelobrojna matrica aproksimacije DCT-a, treba pomnožiti elemente DCT matrice s proizvoljnim faktorom skale i zaokružiti ih. Formula iz [2, str. 401] glasi:

$$\mathbf{H} = \text{round}(\alpha \cdot \mathbf{T}_4), \tag{2-2}$$

gdje je:

- **H** – cjelobrojna matrica koja aproksimira 4x4 DCT,
- α – faktor skale,
- $\mathbf{T_4}$ – 4x4 DCT matrica.

Matrica 4x4 DCT-a može se prikazati kao:

$$\mathbf{T_4} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

gdje je:

$a = \frac{1}{2}$,

$b = \sqrt{\frac{1}{2}} \cos \frac{\pi}{8}$,

$c = \sqrt{\frac{1}{2}} \cos \frac{3\pi}{8}$.

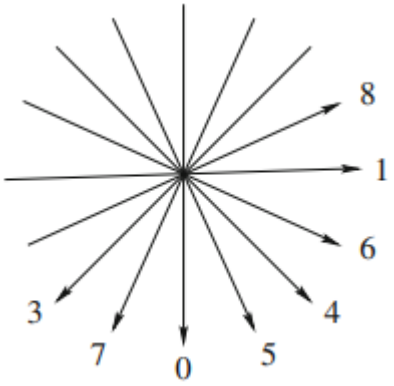
2.2.2. Predviđanje za unutarokvirno kodiranje

H.264 koristi ne samo međuokvirno predviđanje, nego i predviđanje unutar okvira za daljnju unutarokvirnu kompresiju. Unutarokvirno kodirani makroblokovi se predviđaju korištenjem susjednih elemenata slike koji su rekonstruirani ili unutarokvirnim, ili međuokvirnim predviđanjem. Postoji 4 moda za unutarokvirno kodiranje: *Intra_4x4* za luma makroblokove, *Intra_16x16* za luma makroblokove, *Intra* za kroma makroblokove, te *I_PCM*.

Intra_4x4 i *Intra_16x16* označavaju veličinu blokova za predviđanje. Postoji 9 modova predikcije za 4x4 blokove, dok za 16x16 blokove postoji samo 4, te svaki blok za predviđanje može imati vlastiti mod predikcije koji koder odabire ovisno o tome koji mod proizvodi najmanju grešku predikcije. Greška predikcije, kao i kod međuokvirnog predviđanja, je vrijednost razlike između vrijednosti trenutnog okvira koji se kodira i predviđene vrijednosti, koja se u H.264 potom kodira cjelobrojnom transformacijom.

Modovi predikcije za 4x4 blokove predviđanja su: mod 0: vertikalni, mod 1: horizontalni, mod 2: DC, mod 3: dijagonalni dolje-lijevo, mod 4: dijagonalni dolje-desno, mod 5: vertikalno-desno, mod 6: horizontalno-dolje, mod 7: vertikalno-lijevo, te mod 8: horizontalno-gore. Svi modovi osim DC moda označavaju smjer predviđanja, koji se mogu

vidjeti na slici 2.2, iz [2, str. 406]. Na primjer, za mod 0 vrijednosti elemenata slike u prvom redu će se koristiti za predikciju svih redova ispod njega, dok za mod 4 prvi red i prvi stupac će se koristiti za predikciju elemenata slike u ravnini dijagonalno dolje-desno. Modovi od 5 do 8 koriste smjerove koji su između dijagonale i horizontale/vertikale u omjeru 2:1 (krećući se 2 elemenata slike horizontalno/vertikalno, 1 element slike dijagonalno). Ovi modovi se razlikuju po tome što za neke elemente slike koji se predviđaju ne postoji jedinstveni referentni element slike, te se za referencu mora koristiti interpolacija između dva najbliža referentna elemenata slike. DC mod je posebni mod gdje se srednja vrijednost 8 prethodno kodiranih susjeda koristi za referencu predviđanja svih elemenata slike u bloku.



Slika 2.2. *Smjerovi predikcije u Intra_4x, gdje svaki broj označava pripadni mod,* [2]

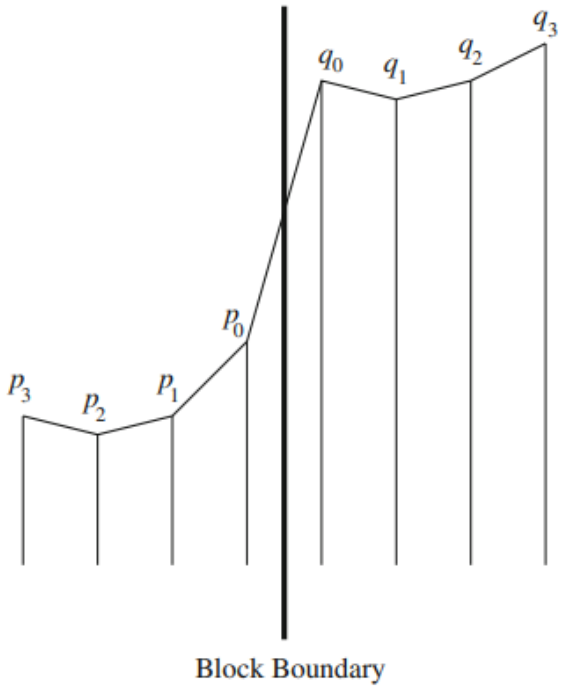
Modovi predikcije za 16x16 blokove predviđanja su: mod 0: vertikalni, mod 1: horizontalni, mod 2: DC, te mod 3: planarni. Svi modovi, osim moda 3, rade jednako kao kod 4x4 blokova, ali samo s većim dimenzijama bloka. Planarni mod je jedinstven za 16x16 blokove, a za predikciju koristi linearnu funkciju za gornje i lijeve uzorke u bloku.

Intra kodiranje za kroma makroblokove koristi iste modove predikcije kao *Intra_16x16* luma, te veličine predikcijskih blokova ovise o formatu uzorkovanja boja. Za 4:2:0 to je 8x8, za 4:2:2 je 8x16, te za 4:4:4 je 16x16.

I_PCM mod zaobilazi sve proračune predikcije i transformacijskog kodiranja, te direktno šalje luma i kroma vrijednosti koristeći PCM kodiranje (eng. *Pulse Code Modulation* – PCM). Koristi se kada ostali modovi unutarokvirnog kodiranja ne postižu nikakvu kompresiju, što se pojavljuje u rijetkim slučajevima.

2.2.3. Filtar za uklanjanje efekta bloka unutar petlje

H.264 koder koristi post-procesni filtar kako bi smanjio nuspojavu blokovnog kodiranja gdje se pojavljuje tzv. „efekt bloka.“ Dijeljenje slike na blokove i njihovo kodiranje neovisno jednih o drugima proizvodi značajno vidljive pravokutne artefakte u slici. Tijekom petlje kodiranja bloka, ne uzimaju se u obzir vrijednosti elemenata slike izvan bloka i to uzrokuje pojavljivanje znatnih razlika vrijednosti između elemenata slike na granicama susjednih blokova. Zbog toga se koristi filtar za uklanjanje efekta bloka i on je vitalan alat za poboljšanje kvalitete slike. Ovaj filtar je prilagodljiv signalu i koristi skup filtera koji se primjenjuju na 4x4 okolinama oko rubova, te se u petlji pojavljuje nakon inverzne cjelobrojne transformacije. Značajke filtra ovise o otkrivanju rubova i parametrima makroblokova, kao što su činjenica da li su unutar ili međuokvirno predviđeni, koji su kodirani koeficijenti, razlike između referentnih okvira, itd. Time se moduliraju snaga, vrsta i duljina filtra kako bi se efekt bloka što više smanjio, a kvaliteta slike održala.



Slika 2.3. *Jednodimenzionalni prikaz vrijednosti između suprotnih strana granice bloka,* [2]

Prema [2, str. 408] uzeta je slika kao primjer jednodimenzionalnog ruba bloka, gdje su p_n, q_n elementi slike, a visine njihove vrijednosti. Osnovno načelo rada deblokirajućeg filtriranja je omekšavanje rubova blokova, tj. uzimanje srednjih vrijednosti elemenata slike

u okolinama oko rubova i pridruživanje tih vrijednosti elementima slike. Međutim, pri tome se mora voditi računa da se ne omekšaju rubovi koji su dio originalnog okvira, te se moraju raspoznavati rubovi koji su rezultat blokovnog kodiranja od stvarnih rubova. Kako bi se to ostvarilo, H.264 deblokirajući filtar se primjenjuje samo ako se ispune određeni uvjeti na osnovi praga, a za p_0 i q_0 to su:

$$|p_0 - q_0| < \alpha(Q\ P),$$

$$|p_0 - p_1| < \beta(Q\ P),$$

$$|q_0 - q_1| < \beta(Q\ P),$$

dok su za p_1 i q_1:

$$|p_0 - p_2| < \beta(Q\ P),$$

$$|q_0 - q_2| < \beta(Q\ P),$$

gdje je:

p_0, q_0, p_1, q_1 – elementi slike unutar 4x4 okoline oko ruba makrobloka,

α, β – pragovi koji su funkcije $Q\ P$,

$Q\ P$ – parametar kvantizacije.

Parametar kvantizacije $Q\ P$ i pragovi α, β su međusobno proporcionalni – što je parametar kvantizacije veći, to su pragovi veći i obrnuto. Implementacija je takva jer u slučajevima gdje je $Q\ P$ malen je manje vjerojatno da će veće razlike u vrijednostima elemenata slike biti zbog efekta bloka, jer se intenziteti manje mijenjaju pod utjecajem kodiranja negoli kada je parametar kvantizacije veći.

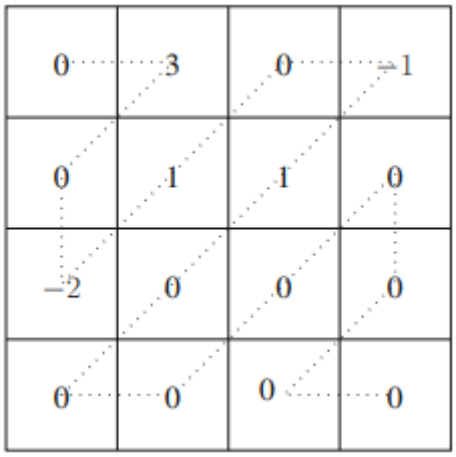
2.2.4. Entropijsko kodiranje

H.264 može koristiti 3 različitih metoda entropijskog kodiranja, a to su: eksponencijalni Golombov kôd (eng. *Exponential-Golomb – Exp-Golomb*), kodiranje varijabilne duljine prilagodljivo sadržaju (eng. *Context Adaptive Variable Length Coding* – CAVLC) i binarno aritmetičko kodiranje prilagodljivo sadržaju (eng. *Context Adaptive Binary Arithmetic Coding* – CABAC). Parametar *entropy_coding_mode* određuje koje će se metode koristiti; ako je postavljen na 0, eksponencijalni Golombov kôd 0. reda će se koristiti za zaglavlja, vektore predikcije i ostale podatke proračuna predikcije, a podaci

okvira greške predikcije će se kodirati kompleksnijim CAVLC-om. S druge strane ako je *entropy_coding_mode* postavljen na 1, koristit će se i CABAC za određene podatke.

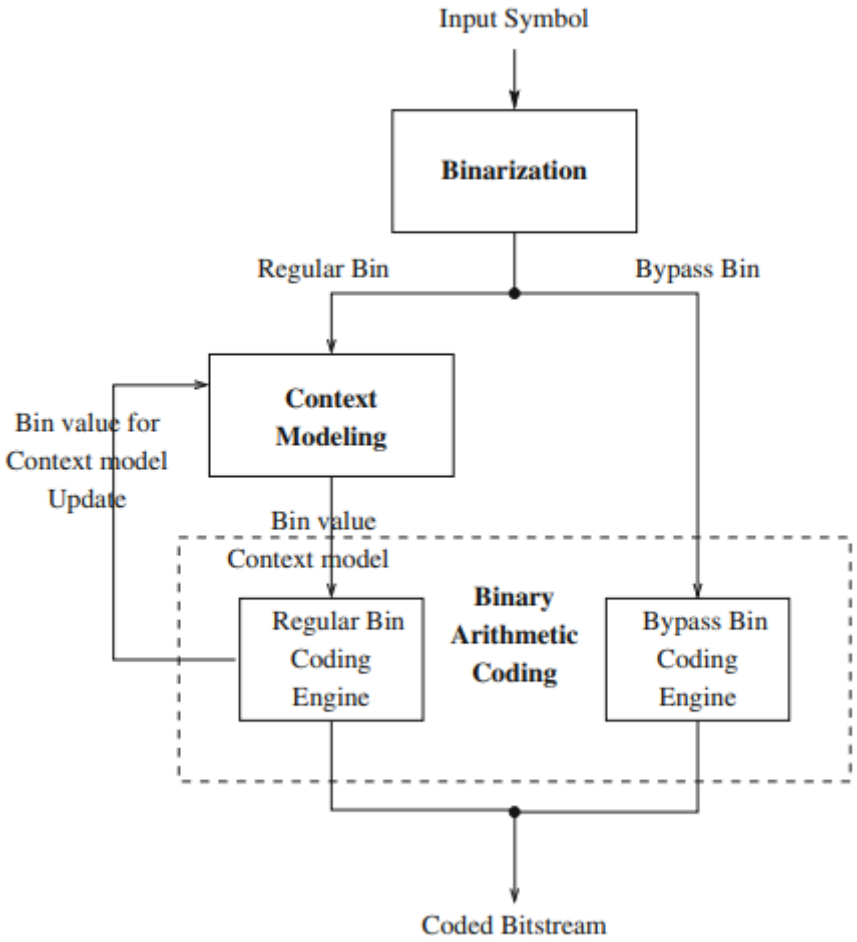
Eksponencijalni Golombov kôd može biti *n*-tog reda, a onaj koji se koristi u H.264 je 0. reda koji se također naziva i jednostavni eksponencijalni Golombov kôd. To je binarni kôd koji se sastoji od 3 dijela: prefiks, binarni 1, te sufiks. Prefiks je sekvenca koja se sastoji od l broja nula, gdje je $l = \lfloor \mathrm{ld}(N + 1) \rfloor$, a N je cijeli broj bez predznaka koji se kodira. Binarni broj 1 je jednostavno separator koji se uvijek umeće između prefiksa i sufiksa kako bi se oni mogli raspoznati. Sufiks je binarni broj $N + 1 - 2^l$ prikazan u l bitova. Ako se kodira cijeli broj s predznakom, zadnji bit rezultirajućeg sufiksa se rezervira za predznak. Brojevi bez predznaka se koriste za zapisivanje indeksa referentnog okvira, tipa makrobloka, itd., dok se brojevi s predznakom koriste za npr. razlike vektora predviđanja. Dekodiranje jednostavnog eksponencijalnog Golombovog kôda provodi se na sljedeći način: očitava se slijed nula dok se ne dođe do separatora 1, spremajući broj nula u varijablu l, preskoči se separator i očita sljedećih l bitova spremajući ih u varijablu S, te da bi se pronašao kodirani broj N izračuna se $S - 1 + 2^l$.

CAVLC je unaprijeđena metoda VLC kodiranja iz prethodnih normi video kodiranja gdje se koriste različite VLC tablice ovisno o vrsti podataka i kontekstu. Time se mogu odabrati optimizirani načini kodiranja ovisno o sekvenci nula, nivoa, prethodno dekodiranih blokova itd. CAVLC iskorištava nekoliko poznatih činjenica; matrice kvantiziranih koeficijenata greški predikcije tipično imaju veliki broj nula i jedinica sa i bez predznaka (tzv. prateće jedinice). Postupak kodiranja je sljedeći: koeficijenti matrice se slažu u jednodimenzionalni niz cik-cak metodom (kao što je prikazano na slici 2.4., iz [2, str. 411]). Zatim se uzima broj koeficijenata koji nisu nule *TotalCoeff* i broj pratećih jedinica *Trailing_1s* i njihovi predznaci. Uzimaju se nivoi koeficijenata koji nisu nule ni prateće jedinice – nivo je magnituda i predznak koeficijenta. Naposlijetku se uzimaju broj i niz nula prije zadnjeg koeficijenta koji nije nula, *zeros_left* i *run_before*.



Slika 2.4. *Primjer 4x4 matrice kvantiziranih koeficijenata,* [2]

CABAC se koristi umjesto CAVLC-a za neke podatke i koeficijente kada je varijabla *entropy_coding_mode* postavljena na 1. Cilj je ostvariti veću efikasnost kompresije u slučajevima gdje simboli imaju vjerojatnost pojavljivanja p veću od 0.5 – CAVLC mora pridružiti svakom simbolu barem 1 bit, što može biti puno veće od njegove vlastite informacije $\mathrm{ld}\frac{1}{p}$. Dijagram toka CABAC-a je prikazan na slici 2.5., prema [2, str. 414].



Slika 2.5. *Dijagram toka CABAC kodera u H.264,* [2]

CABAC se sastoji od 3 glavna dijela: binarizacija, modeliranje konteksta i binarno aritmetičko kodiranje. Svi podaci koji nisu binarni se moraju pretvoriti u binarni oblik, a to se može učiniti na pet načina: unarni, odrezani unarni, eksponencijalni Golombov kôd *k*-tog reda, konkatenacija unarnog i eksp. Golombovog kôda, te binarna shema fiksne duljine. Zatim se odabire prikladan model konteksta koji ovisi o vrsti podatka i prethodno kodiranim podacima. Za većina simbola se koristi tzv. „regularni mod kodiranja," kao što su vrsta makrobloka, podaci greške predikcije, informacija o modovima predikcije, odsječcima, itd. Kako bi se ubrzalo kodiranje, može se koristiti „mod zaobilaženja kodiranja" gdje se ne koristi nikakav model konteksta. Naposlijetku, obavlja se binarno aritmetičko kodiranje. Ono uzima dva simbola, te rekurzivno dijeli raspon između njih. Nedostatak u usporedbi s ostalim metodama entropijskog kodiranja je činjenica što je potrebna relativno velika procesna snaga kako bi se mogle obaviti brojne operacije množenja, no zbog upotrebljavanja samo dva simbola množenje je svedeno na minimum.

Ta dva simbola se uzimaju kao *LPS* (eng. *Least Probable Symbol* – najmanje vjerojatan simbol) i *MPS* (eng. *Most Probable Symbol* – najviše vjerojatan simbol), raspon između njih *R*, te donja granica od *R* zvana *L*. P_{LPS} je vjerojatnost *LPS*-a, a P_{MPS} se računa kao $P_{MPS} = 1 - P_{LPS}$. Zatim se generira novi raspon koristeći sljedeći kôd iz [2, str. 414.]:

```
PROCEDURE Računanje raspona u binarnom aritmetičkom kodiranju
BEGIN
    If S is MPS
        R = R · (1 - PLPS);
    Else //S je LPS
        L = L + R · (1 - PLPS);
        R = R · PLPS;
END
```

Izračun R = R · PLPS zahtijeva relativno veliku procesnu snagu zbog operacije množenja, te se zato umjesto njega u H.264 koristi tzv. modulo koder. Modulo koder zamjenjuje operaciju množenja s 4x64 tablicom vrijednosti koje su prethodno izračunate. Postoje 4 vrijednosti za *R* i 64 vrijednosti za P_{LPS}, te novi raspon poprima jednu od vrijednosti iz tablice ovisno o *R* i P_{LPS}. Time se povećava brzina računanja, ali smanjuje preciznost rezultata. Međutim, negativan efekt smanjene preciznosti je minimalan.

3. ZNAČAJKE I ALATI NORME H.265/HEVC

H.265 norma nasljeđuje značajke i alate iz H.264 norme, ali u dodatku tome također koristi nove alate koji služe za poboljšanje kompresije videa. Ovdje će biti opisane njezine značajke i alati.

3.1. Značajke norme H.265/HEVC

H.265 je poboljšani standard za kodiranje videa koji gradi nad osnovama H.264. Njegove glavne značajke su iskorištavanje novih sposobnosti paralelne obrade za ubrzanje kompleksnijih metoda kodiranja i dekodiranja, te unaprijeđeni stupanj kompresije za istu kvalitetu videa. U usporedbi sa H.264, H.265 prosječno ostvaruje do 50% manju veličinu videa za istu kvalitetu. Međutim, unatoč implementaciji novih algoritama paralelne obrade, brzina kodiranja i dekodiranja može biti i do 66% manja od H.264 zbog dodataka novih alata i općenito povećane kompleksnosti kodiranja. Zbog ovog razloga, kompatibilnost s postojećim uređajima je smanjena, no i dalje je omogućeno široko područje primjene putem implementacije razina i profila kao i u H.264.

H.265 također koristi CRF za VBR mod rada s vrijednošću od 0 do 51 kao i H.264, a zadana mu je vrijednost 28. Zadani format boje je također YCbCr s poduzorkovanjem 4:2:0 i 8 bita po kanalu, a po potrebi podržava poduzorkovanja 4:2:2, 4:4:4, više od 8 bita po kanalu i monokromatsku sliku.

3.2. Alati norme H.265/HEVC

H.265 koristi nekoliko novih ili poboljšanih alata za unaprjeđenje kodiranja, a to su: četverostablasta (eng. *quadtree*) hijerarhija blokova, paralelna obrada, cjelobrojna transformacija, unutarokvirno predviđanje, diskretna sinusna transformacija (eng. *Discrete Sine Transform* – DST), deblokirajući filtar, pomak prilagodljiv uzorku (eng. *Sample Adaptive Offset* – SAO), entropijsko kodiranje zasnovano na binarnom aritmetičkom kodiranju prilagodljivom sadržaju, itd.

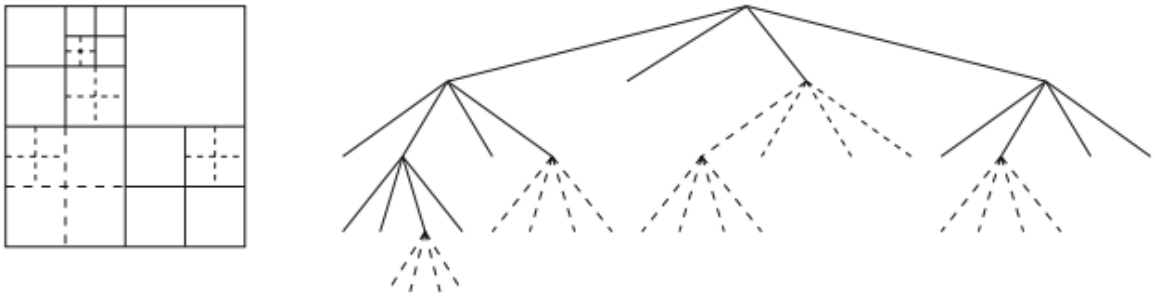
3.2.1. Kompenzacija pokreta

Za kompenzaciju pokreta, H.265 ne koristi fiksnu strukturu makroblokova kao kod H.264, nego četverostablastu hijerarhiju blokova različitih veličina. H.265, kao i H.264, za

kodiranje koristi izračun vektora pokreta i kodiranje okvira razlike (greške predikcije). Cilj ovakve strukture s više podijeljenih blokova je smanjiti greške predikcije kako bi se povećala efikasnost kodiranja. Jedinice ove hijerarhije su sljedeće, od najosnovnije (najveće) do najdetaljnije (najmanje): jedinica stabla za kodiranje (eng. *Coding Tree Unit* – CTU), blok stabla za kodiranje (eng. *Coding Tree Block* – CTB), jedinica za kodiranje (eng. *Coding Unit* – CU), blok za kodiranje (eng. *Coding Block* – CB), jedinica za predikciju (eng. *Prediction Unit* – PU), blok za predikciju (eng. *Prediction Block* – PB), jedinica za transformaciju (eng. *Transform Unit* – TU), blok za transformaciju (eng. *Transform Block* – TB).

CTU je najosnovnija jedinica hijerarhije i sadrži 1 CTB luminantne komponente i 2 CTB krominantne komponente. Luminantni CTB može biti veličina 16x16, 32x32 ili 64x64, a krominantni CTB-ovi su 4 puta manji od odabranih dimenzija (strane su pola duljine). CTB se dalje dijeli na CU-ove koji sadrže 1 luma CB dimenzije 8x8 i 2 kroma CB-a dimenzije 4x4. CB-ovi se dalje mogu dijeliti na PU-ove koji služe za unutarokvirno i međuokvirno predviđanje. Jedan PU može raditi jedan od ovih modova predikcije i sadrži pripadne luma i kroma PB-ove. Kod unutarokvirne predikcije, veličina PB-ova je ista kao i kod CB-ova s izuzetkom kada je CB dimenzije 8x8 i dopušteno je dijeljenje u 4 PB-a kako bi svaki PB mogao raditi u drugačijem modu predikcije. Za međuokvirno predviđanje, PB-ovi ne smiju biti kvadratni i moraju biti pravokutni, što znači da svaki CB može biti podijeljen na 1, 2 ili 4 PB-a. Naposlijetku, CB može biti podijeljen i na TU-ove s kojima se izvršava transformacijsko kodiranje okvira razlike. Jedan TU sadrži luma i kroma TB-ove. TB-ovi mogu biti dimenzija od 4x4 do 32x32 i mogu se rasprostirati preko granica PB-ova u međuokvirno predviđenim CU-ovima kako bi se poboljšala efikasnost kodiranja.

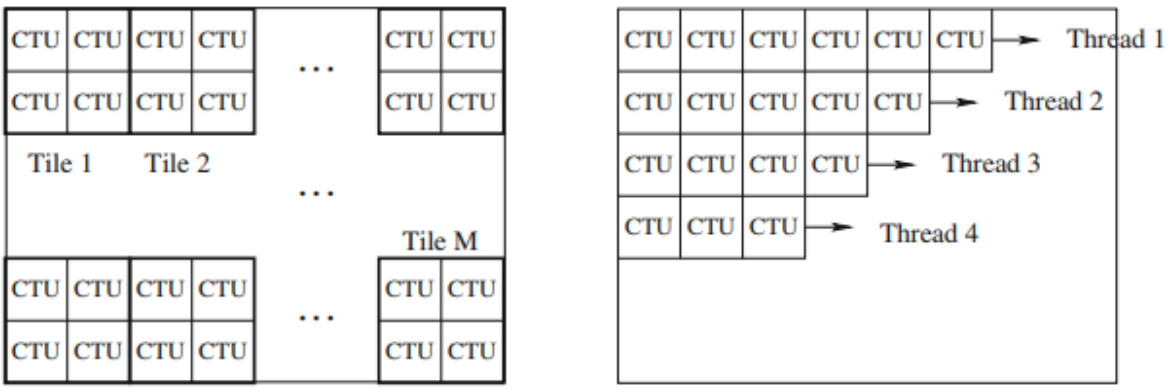
Na slici 3.1., iz [2, str. 420], može se vidjeti vizualni prikaz strukture blokova H.265. Prva slika prikazuje podjelu slike na CTB-e punim linijama i TB-e isprekidanim linijama, a druga slika pripadnu četverostablastu hijerarhiju.



Slika 3.1. *Struktura blokova u H.265,* [2]

3.2.2. Paralelna obrada

H.265 norma je posebnu pažnju posvetila primjeni paralelne obrade kako bi se iskoristila rastuća kompleksnost i mogućnosti sklopovlja i algoritama koji koriste paralelnu obradu, čime se ubrzalo kodiranje i dekodiranje kompleksnih dijelova videa. Postoje dvije metode s kojima se može postići paralelna obrada u H.265, a to su pločice (eng. *tiles*) i paralelna obrada valnog fronta (eng. *Wavefront Parallel Processing* – WPP). Trenutačno H.265 dopušta korištenje samo jedne od ovih metoda istovremeno. Pločice se sastoje od CTU-ova ili tzv. odsječaka (eng. *slices*) i više različitih pločica se može obraditi odjednom u paraleli. Odsječak je alat naslijeđen iz H.264, a to su odsječci okvira bilo kojih dimenzija koji sadrže CTU-ove i mogu biti I, P i B vrste. S druge strane, paralelna obrada valnog fronta omogućava paralelnu obradu redaka CTU-ova u obliku valnog fronta; to znači da svaka procesna nit zauzima i obrađuje jedan redak odozgo prema dolje. Slika 3.2. iz [2, str. 421] prikazuje pločice i valne frontove.



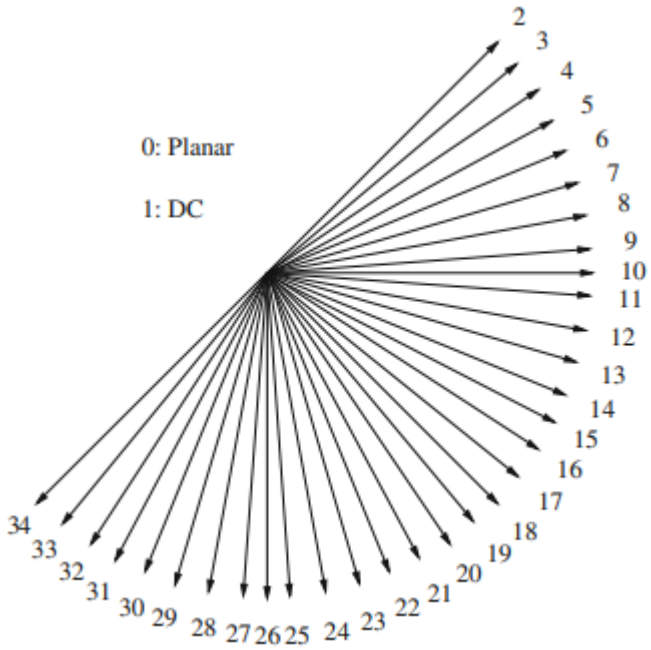
Slika 3.2. *Pločice i valni frontovi,* [2]

3.2.3. Cjelobrojna transformacija

Cjelobrojna transformacija je naslijeđena iz H.264 i koristi se za kodiranje greški predikcije. Ulazni podaci se konvoluiraju s transformacijskom matricom prvo u vertikalnom smjeru te onda horizontalnom, jednakim načinom i formulom kao u H.264. Transformacijska matrica može biti veličina 4x4, 8x8, 16x16 i 32x32, ali samo je matrica veličine 32x32 definirana u H.265. Ostale matrice su izvedene, poduzorkovane verzije definirane matrice gdje se uzimaju vrijednosti te matrice na određenim intervalima. U slučaju ako su magnitude koeficijenata veće nego što 16-bitna memorija dozvoljava, dinamički raspon rezultata se mora smanjiti pomoću 7-bitnog desnog pomaka i sječenja suvišnih bitova.

3.2.4. Unutarokvirno predviđanje

H.265 koristi predviđanje unutar okvira za prostornu kompresiju, kao i H.264. Za predikciju se koriste susjedni uzorci iz blokova gore i/ili lijevo od trenutnog bloka i greške predikcije se kodiraju transformacijskim blokovima. Transformacijski blok (TB) može biti veličina od 4x4 do 32x32. Mogući broj modova predikcije je povećan na 35 u H.265, iz 9 u H.264 jer se pokušavaju smanjiti greške predikcije i nastoji se prilagoditi dimenzijama TB-ova koje bi mogle biti puno veće nego prije. Između modova 2 i 34 su tzv. *Intra_Angular* predikcijski modovi gdje se predikcije obavljaju u 32 smjera od 45° do 225° u smjeru kazaljke na satu. Kutevi između smjerova su nejednaki, te su manji blizu vodoravnih i okomitih smjerova zbog činjenice što će većina potrebnih uzoraka za predikciju biti u području podelementa slike. Za izračune koji padaju u pozicije između elemenata slike koristi se bilinearna interpolacija dva susjedna elementa slike s preciznošću do 1/32 elementa slike. Dva posebna predikcijska moda su mod 0: *Intra_Planar* i mod 1: *Intra_DC*. Slični su kao i kod H.264, ali mod 0 koristi svih 4 kuteva za planarnu predikciju. Mod 1 koristi srednju vrijednost susjednih uzoraka za predikciju. Slika 3.3. iz [2, str. 426] prikazuje moguće smjerove predikcije, gdje brojevi označavaju pripadni mod predikcije.



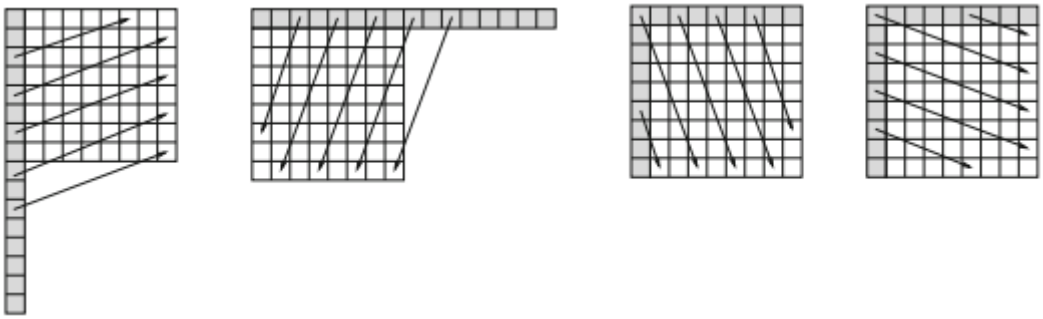
Slika 3.3. *Smjerovi predikcije,* [2]

3.2.5. Diskretna sinusna transformacija

Diskretna sinusna transformacija (DST) se koristi za cjelobrojnu transformaciju za 4x4 luminantne blokove, točnije njezina varijanta DST-VII. Razlog tome je što ima bolje rezultate nego DCT za dijelove bloka koji su udaljeniji od gornjih ili lijevih granica bloka. Zbog toga što se unutarokvirno predviđanje izvršava na susjednim uzorcima kod gornjih i lijevih granica blokova, greške predikcije su veće što su uzorci bloka udaljeniji od tih granica. Međutim, kvaliteta rezultata ovisi o smjeru moda predikcije i DCT ima bolje rezultate u jednim modovima dok DST ima bolje rezultate u drugim, te se zbog tog razloga koriste obje transformacije ovisno o smjeru.

Sukladno tome, Intra_Angular modovi se dijele na 3 kategorije. Kategorija 1 označava da su susjedni uzorci za predikciju svi ili iz gornje, ili iz lijeve strane trenuntnog bloka. U kategoriji 2 uzorci predikcije su i iz gornje, i iz lijeve strane bloka. Treća kategorija, DC, je posebna kategorija gdje je uzima srednja vrijednost susjednih uzoraka. Ovisno o kategoriji i slučaju, koristi se prikladan oblik transformacije. Za kategoriju 1 gdje su svi uzorci iz lijeve strane bloka, koristi se DST za vodoravnu i DCT za okomitu transformaciju. Za kategoriju 1 gdje su svi uzorci iz gornje strane bloka, koriste se obrnute transformacije. Kategorija 2 koristi isključivo DST, a DC kategorija isključivo DCT transformacije. Slika

3.4. iz [2, str. 427] prikazuje slučajeve kategorija smjerova predikcija koje nisu DC. Prva slika je kategorija 1 gdje su uzorci predikcije isključivo s lijeve strane, druga slika je kategorija 1 gdje su uzorci isključivo s gornje strane, a zadnje dvije slike su kategorija 2 gdje su uzorci i iz gornje i lijeve strane bloka.



Slika 3.4. *Kategorije smjerova predikcija u H.265 koje nisu DC,* [2]

3.2.6. Deblokirajući filtar

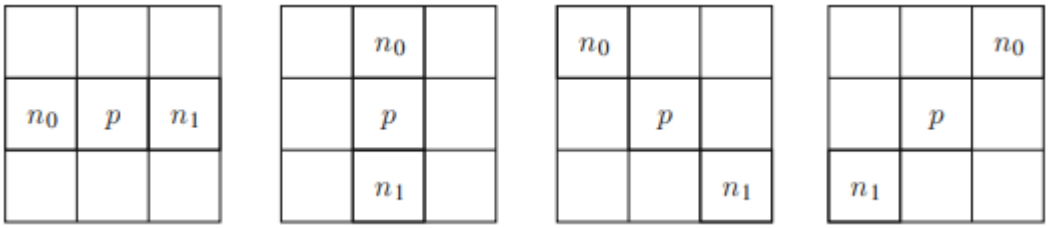
Kao i u H.264, H.265 može koristiti deblokirajući filtar unutar petlje kodiranja, ali je bolje prilagođen za paralelnu obradu. Filtriranje se primjenjuje na rubove 8x8 blokova umjesto 4x4 kao kod H.264; time se smanjuje kompleksnost i poboljšava primjerenost za paralelnu obradu jer se smanjuje šansa za kaskadnim međudjelovanjem između ostalih operacija. Vizualna kvaliteta je uvelike održana, posebice zbog uvođenja alata pomaka prilagodljivog uzorku. Najprije se filtriraju vertikalni rubovi, a onda horizontalni; kako bi se omogućila višenitna paralelna obrada. Alternativni način rada je da se filtriraju pojedinačni CTB-ovi. H.265 također dopušta 3 snage filtriranja, od 0 do 2.

3.2.7. Pomak prilagodljiv uzorku

Pomak prilagodljiv uzorku (SAO) je novi alat koji pokušava rekonstruirati amplitude originalnog signala i može se uključiti opcionalno nakon deblokirajućeg filtra. On izvršava pomak na svakom uzorku ovisno o određenim uvjetima i s time poboljšava kvalitetu slike i smanjuje artefakte. Za svaki pojedinačni CTB, SAO može biti ili isključen ili raditi u jednom od 2 moda: mod pomicanja ruba i mod pomicanja pojasa.

U modu pomicanja ruba, trenutni uzorak je uspoređen s 2 susjedna uzorka koristeći jedan od 4 smjerova prijelaza: horizontalni, vertikalni, i dva dijagonalna smjera. Slika 3.5. iz [2, str. 428] vizualno prikazuje koji se elementi slike uzimaju u obzir u svakim od tih 4

smjerova, u redoslijedu slijeva nadesno. Pet mogućih rezultata usporedbe su: uzorak je lokalni minimum, uzorak je rub koji ima manju vrijednost, uzorak je lokalni maksimum, uzorak je rub koji ima veću vrijednost, te uzorak je monoton. Kod prva dva rezultata primjenjuje se pozitivan pomak, kod sljedeća dva se primjenjuje negativan pomak, a kod zadnjeg rezultata se ne primjenjuje pomak.



Slika 3.5. *Elementi slike koji se uzimaju u obzir u modu pomicanja ruba,* [2]

Kod moda pomicanja pojasa, amplitude uzoraka u CTB-u su podijeljene na histogram od 32 pojasa i izvršava se pomak uzoraka u 4 uzastopna pojasa kako bi se smanjili „artefakti pojasa“ u glađim područjima.

3.2.8. Entropijsko kodiranje

Entropijsko kodiranje u H.265 koristi samo binarno aritmetičko kodiranje prilagodljivo sadržaju (CABAC), te se više ne koristi kodiranje varijabilne duljine prilagodljivo sadržaju u dodatku tome kao kod H.264. CABAC u H.265 je prilagođen njegovoj stablastoj strukturi kodiranja i transformacije i kao takav ima bolju efikasnost i brzinu kompresije. Kontekstno modeliranje je poboljšano kako bi bio odabran pogodan kontekst za kodiranje i broj konteksta je smanjen jer je dubina stabla strukture postala važan dio kontekstnog modeliranja. Koriste se 3 jednostavne metode skeniranja koeficijenta kako bi se maksimizirala duljina redova pronađenih nula: horizontalno, vertikalno i gore-desno. Skeniranje se provodi u 4x4 blokovima neovisno o veličini TB-a. U unutarokvirno predviđenim blokovima veličina 4x4 i 8x8, horizontalno skeniranje se koristi za smjerove predikcije koji su blizu vertikalnoj osi, vertikalno skeniranje za smjerove koji su blizu horizontalnoj osi, a skeniranje gore-desno za ostale smjerove. Za unutarokvirno i međuokvirno predviđene blokove veličina 16x16 i 32x32 uvijek se koristi skeniranje gore-desno. Kodiranje koeficijenata koji nisu nule je također poboljšano.

4. KODIRANJE VIDEA PRIMJENOM H.265/HEVC KODERA U FFMPEG PAKETU

U ovom je poglavlju razrađen kôd FFmpeg programa koji će se koristiti u svrhu uspoređivanja H.264 i H.265 videa, te su dani rezultati kodiranja. Da bi se bolje razumjele metode koje će se koristiti za ocjenu kvalitete, one su također obrađene u posebnim potpoglavljima.

4.1. Objektivne mjere za ocjenu kvalitete

4.1.1. PSNR

Omjer vršnog signala i šuma (eng. *Peak Signal-to-Noise Ratio* – PSNR) je jedna od najčešćih i najjednostavnijih korištenih mjera za objektivnu kvalitetu videa. Općenito govoreći, PSNR određuje koliki je omjer između najviše moguće snage željenog signala i prosječne snage neželjenog signala (šuma). U kontekstu mirne slike, najviša snaga signala je najviši intenzitet elementa slike kojeg omogućava format boje videa (255 za 8 bitni kanal), a prosječna snaga šuma odgovara srednjoj kvadratnoj pogreški (eng. *Mean Squared Error* – MSE) između originalne i kodirane slike. PSNR videa je jednostavno prosjek PSNR-ova svakog njegovog pojedinog okvira. Mjerna jedinica PSNR-a je decibel – dB, a formula za mirne slike iz [2, str. 430] je:

$$PSNR = 10 \log_{10} \frac{I_{max}^2}{MSE}, \tag{4-1}$$

gdje je:

- *PSNR* – omjer vršnog signala i šuma,
- I_{max} – maksimalna vrijednost intenziteta,
- *MSE* – srednja kvadrirana pogreška.

4.1.2. SSIM

Indeks mjere strukturne sličnosti (eng. *Structural Similarity Index Measure* – SSIM) ili samo indeks strukturne sličnosti (eng. *Structural Similarity Index*) je naprednija objektivna mjera kvaliteta videa koja uzima u obzir strukturu slike. Ona uspoređuje razliku između strukture originalne i kodirane slike, čime se ostvaruje realističniji pogled na degradaciju kvalitete negoli s metodama koje samo uspoređuju razlike u intenzitetima

elemenata slike neovisno jedan od drugih, kao što je PSNR. Kako bi se izračunale razlike u strukturama slika, mora se ukloniti utjecaj osvjetljenja na objekte u slici – cilj je razdvojiti promjene u osvjetljenju i kontrastu od strukturalnih izobličenja, jer ona puno više pridonose urušavanju kvalitete. Strukture slike se mogu modelirati kao vektori u slikovnom prostoru, gdje razlike između kutova vektora označavaju razliku u strukturi. Specifični SSIM indeks se računa prema formuli iz [3, str. 14]:

$$\text{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4-2}$$

gdje je:

- SSIM(**x**,**y**) – specifični SSIM indeks između prostornih vektora slika **x** i **y**,
- μ_x, μ_y – srednja vrijednost intenziteta vektora **x** i **y**,
- σ_x, σ_y – standardna devijacija intenziteta vektora **x** i **y**,
- σ_{xy} – standardna devijacija između vektora **x** i **y**,
- C_1, C_2 – konstante za stabilnost kada su $\mu_x^2 + \mu_y^2$ ili $\sigma_x^2 + \sigma_y^2$ veoma blizu nuli.

Računanje SSIM-a se najčešće provodi lokalno za pojedinačne dijelove slike, jer kod slika s velikim brojem objekata s različitim karakteristikama svaki objekt može biti urušen na drugačiji način. U tu svrhu slika se može podijeliti na blokove npr. 8x8 dimenzija, ili na omekšane dijelove koristeći Gaussovu težinsku funkciju – zadnja daje prirodnije rezultate bez efekta bloka. Rezultati SSIM-a se mogu prikazati vizualno kao slika, ili kao jedan broj ocjene kvalitete. Kada su prikazani kao slika, vrijednosti intenziteta elemenata slike poprimaju vrijednosti SSIM-a npr. na način da svijetle boje označavaju područja slike koja su najmanje narušena, a tamne boje ona koja su najviše narušena. Slika 4.1., iz [3, str. 20], prikazuje primjer vizualnog prikaza SSIM-a. S druge strane, SSIM kao jedan broj ocjene kvalitete je prosjek svih izračunatih SSIM vrijednosti, gdje se mogu koristiti različita težišta za svaki element slike.

a) b) c)

Slika 4.1. *Strukturna sličnost između originalne slike i slike kodirane JPEG2000 koderom,* [3]:

a) original;
b) kodirana slika;
c) prikaz SSIM vrijednosti

4.2. H.264/AVC i H.265/HEVC koder

Za kodiranje videa u H.264 i H.265 formate korišten je FFmpeg, prema [1]. FFmpeg je besplatan program otvorenog izvora koji koristi opširni izbor biblioteka za kodiranje i obradu multimedijskih datoteka. FFmpeg nema grafičko sučelje, nego se upravlja preko komandne linije. Tu činjenicu iskorištava Python program ovog završnog rada, te se on zasniva na predavanju argumenata FFmpeg-u u komandnoj liniji. Program za kodiranje H.264 i H.265 formata koristi dvije biblioteke iz FFmpeg-a; *libx264* i *libx265*, a svaka od njih odgovara pripadnom formatu. Ispis FFmpeg-a tijekom kodiranja se prikazuje u prozoru naredbenog retka u Windowsu, a Python program izvlači važne podatke kao PSNR i SSIM i zapisuje ih u tekstnu datoteku kako bi se mogla izvršiti daljnja analiza.

Ovdje će se obraditi glavne značajke Python programa, a cjelokupni kôd koji je bio korišten za analizu se nalazi u prilogu završnog rada pod oznakom P.4.1. Glavna funkcija u Python programu koja obavlja sve radnje vezane uz kodiranje i zapis podataka je:

encode(ffmpegpath,inputpath,outputpath,codec,profile,bitrates),

gdje je *ffmpegpath* putanja do FFmpeg programa, *inputpath* direktorij mape u kojoj se nalaze video datoteke koje će se kodirati, *outputpath* direktorij u koji će se spremiti kodirane video datoteke, *codec* biblioteka za kodiranje, *profile* korišteni profil kodera, te

bitrates red kodnih brzina s kojima će se video datoteke kodirati. Funkcija *encode()* se u kôdu poziva dva puta, gdje su svi predani argumenti isti osim *library* i *profile* koji su namješteni na *libx264* i profil *high* za H.264 kodiranje, te *libx265* i profil *main* za H.265 kodiranje.

Funkcija *encode()*, koristeći *for* petlje, uzima sve video datoteke u ulaznoj mapi, kodira svaku na sve kodne brzine koje su u varijabli reda *bitrates*, te zapisuje PSNR i SSIM vrijednosti u datoteku *psnr&ssim.log*. FFmpeg se za svako kodiranje izvršava tri puta; dva puta za kodiranje u dva prolaza i jedanput za izračun PSNR i SSIM vrijednosti. Za prva dva pozivanja se koriste sljedeće dvije linije:

> *os.system(ffmpegpath + " -y -i " + inputfile + " " + encodeparams + " -pass 1 -passlogfile " + outputfile_noext + " -f mp4 NUL")*,
>
> *os.system(ffmpegpath + " -y -i " + inputfile + " " + encodeparams + " -pass 2 passlogfile " + outputfile_noext + " " + outputfile)*.

Naredba *os.system()* je Python naredba za izvršavanje naredbi komandne linije u Windowsu. Jedini argument koji ima je niz znakova koji predstavlja takvu naredbu, a u ovom programu ona je kombinacija sljedećih dijelova: *ffmpegpath* je putanja do FFmpeg-a, *-y* naređuje FFmpeg-u da ignorira greške, *-i* i *inputfile* je putanja do ulazne datoteke, *encodeparams* su parametri kodiranja, *-pass 1* i *2*, *passlogfile*, *outputfile_noext*, *-f mp4 NUL* su vezani uz kodiranje u dva prolaza, te *outputfile* je putanja do izlazne datoteke. Varijabla *encodeparams* je značajan dio ovih naredbi, a u njoj se nalaze argumenti za postavljanje parametara kodiranja prema predloženim vrijednostima u potpoglavlju 4.3. U programu ona se postavlja na sljedeći način:

> *encodeparams = "-c:v " + codec + " -profile:v " + profile + " -level:v 4.0 -coder ac -qcomp 0.9 -qmin 3 -b:v " + bitrate + " -maxrate 20000k -bufsize 40000k"*,

gdje je *-c:v* i *codec* biblioteka za kodiranje (dakle *libx264* za H.264 i *libx265* za H.265 u ovom radu), *-profile:v* i *profile* profil videa (*high* za H.264, *main* za H.265), *-level:v 4.0* razina videa 4.0, *-coder ac* postavljanje entropijskog kodera na CABAC za H.264, *qcomp 0.9* kompresija kvantizacijske skale vrijednosti 0.9, *qmin 3* minimalna kvantizacija vrijednosti 3, *-b:v* i *bitrate* kodna brzina koja je različita za svaki prolaz *for* petlje, *-maxrate 20000k* i *-bufsize 40000k* vršna kodna brzina 20 Mb/s s veličinom međuspremnika 40 Mb/s.

U trećem izvršavanju, FFmpeg uspoređuje ulaznu i izlaznu video datoteku te izračunava PSNR i SSIM vrijednosti, koje Python program dalje obrađuje i zapisuje u tekstnu datoteku. Kôd s kojim se FFmpeg poziva glasi:

process = subprocess.Popen(ffmpegpath + " -y -i " + inputfile + " -i " + outputfile + " -lavfi \"psnr;[0:v][1:v]ssim\" -f null -",stdout=subprocess.PIPE,stderr = subprocess.PIPE,stdin=subprocess.PIPE,universal_newlines=True).

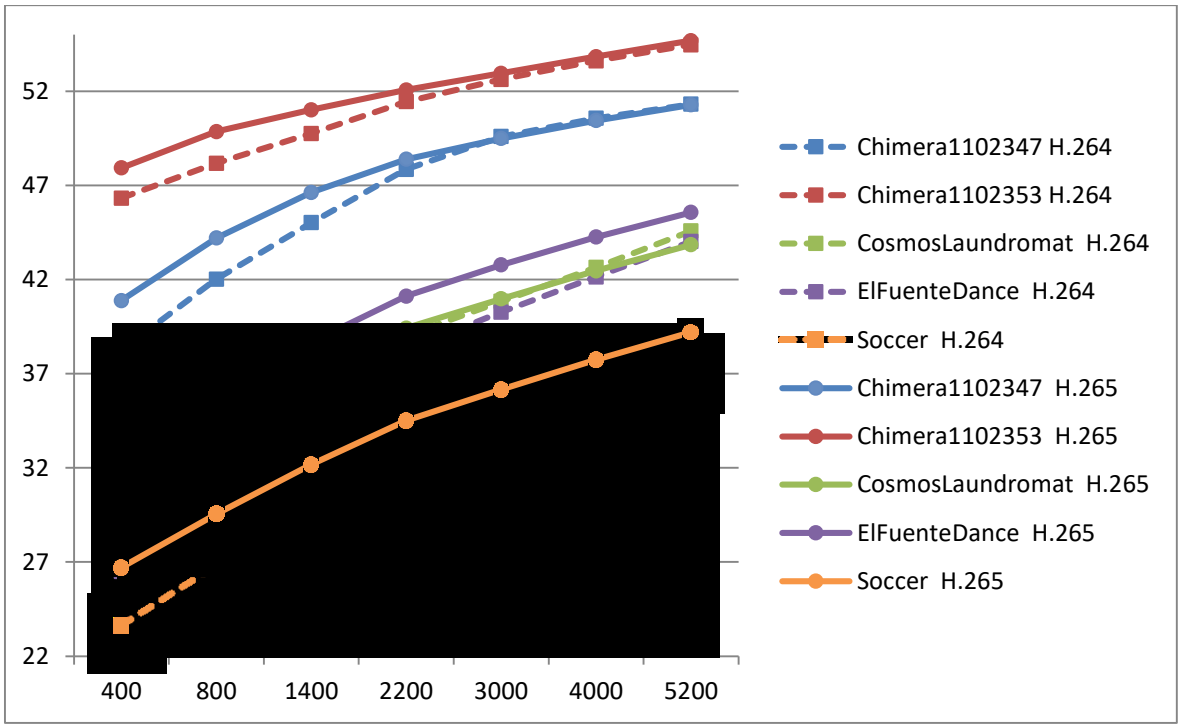
Naredba *process = subprocess.Popen()* pokreće potproces i sprema ga u varijablu *process*. Razlog zašto se FFmpeg ovaj put pokreće na ovakav način umjesto s *os.system()* je omogućavanje iščitavanja ispisa FFmpeg-a kako bi se mogle izvući vrijednosti PSNR i SSIM. Kao i u *os.system()*, prvi argument je niz znakova koji predstavlja naredbu u komandnoj liniji. U ovom slučaju sastoji se od sličnih elemenata kao i u prethodnim FFmpeg pozivanjima, s dvjema razlikama. Ovaj put postoje dvije ulazne datoteke (*-i* argument se predaje dva puta), a to su ulazna i izlazna datoteka iz prethodnog kodiranja. Zatim se predaju argumenti *-lavfi \"psnr;[0:v][1:v]ssim\" -f null -* koji naređuju FFmpeg-u da primijeni filtar usporedbe ulaznih datoteka kako bi se izračunali PSNR i SSIM. Nadalje, program izvlači te vrijednosti i sprema ih u tekstnu datoteku *psnr&ssim.log*.

4.3. Usporedba kvalitete kompresije primjenom H.264 i H.265 kodera

Postupak testiranja kvalitete videa je sljedeći: program tijekom kodiranja ispisuje vrijednosti PSNR i SSIM za svaki kodirani video, te prosjek istih za svaku odabranu kodnu brzinu. Te vrijednosti su upisane u Microsoft Excel tablicu, gdje je jedan list posvećen posebno PSNR-u, te jedan SSIM-u. Svaki list sadrži pripadne vrijednosti za H.264 i H.265 kodirane video datoteke, te su kreirani grafovi gdje je x os kodna brzina, a y os vrijednost PSNR ili SSIM. Korišteno je 5 različitih video datoteka, a postavke parametara kodera su sljedeće: kodiranje varijabilnom kodnom brzinom u dva prolaza, profil *„high“* za H.264 i *„main“* za H.265, razina 4.0, uključeno CABAC entropijsko kodiranje za H.264, kodne brzine 400 kb/s 800 kb/s 1400 kb/s 2200 kb/s 3000 kb/s 4000 kb/s 5200 kb/s, vršna kodna brzina 20 Mb/s, kompresija kvantizacijske skale 0.9, te minimalna kvantizacija 3. Slijede tablice vrijednosti i slike prikaza grafova. U tablici 4.1. dane su PSNR vrijednosti kodiranih videa i u slici 4.2. je prikazan pripadni graf, a u tablici 4.2. se nalaze SSIM vrijednosti i u slici 4.3. je prikazan graf tih vrijednosti.

Tablica 4.1. *PSNR vrijednosti kodiranih videa*

H.264						
Kodna brzina [kb/s]	Chimera1102347 H.264	Chimera1102353 H.264	CosmosLaundromat H.264	ElFuenteDance H.264	Soccer H.264	Prosječan PSNR [dB]
400	38.26	46.33	29.08	26.49	23.62	32.75
800	42.02	48.18	32.22	31.35	26.80	36.11
1400	45.02	49.75	36.28	35.29	30.06	39.28
2200	47.85	51.45	39.02	38.25	32.55	41.83
3000	49.60	52.63	40.84	40.27	34.34	43.54
4000	50.57	53.62	42.64	42.14	36.06	45.01
5200	51.31	54.47	44.58	44.02	37.70	46.41
H.265						
Kodna brzina [kb/s]	Chimera1102347 H.265	Chimera1102353 H.265	CosmosLaundromat H.265	ElFuenteDance H.265	Soccer H.265	Prosječan PSNR [dB]
400	40.88	47.93	32.14	31.91	26.71	35.92
800	44.20	49.86	34.77	35.68	29.55	38.81
1400	46.63	51.02	37.24	38.68	32.18	41.15
2200	48.39	52.07	39.43	41.12	34.51	43.10
3000	49.49	52.95	40.98	42.78	36.16	44.47
4000	50.45	53.84	42.46	44.26	37.74	45.75
5200	51.28	54.69	43.85	45.57	39.20	46.92

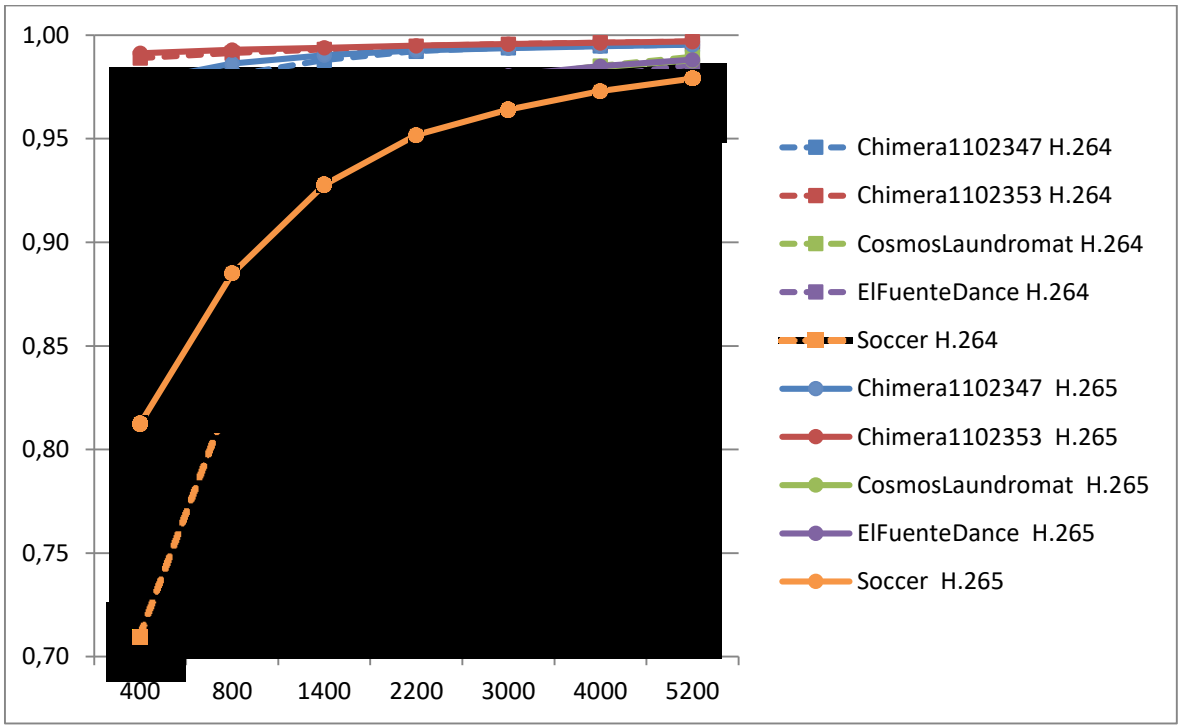


Slika 4.2. *Graf PSNR vrijednosti kodiranih videa*

Tablica 4.2. *SSIM vrijednosti kodiranih videa*

H.264						
Kodna brzina [kb/s]	Chimera1102347 H.264	Chimera1102353 H.264	CosmosLaundromat H.264	ElFuenteDance H.264	Soccer H.264	Prosječan SSIM
400	0.9664	0.9890	0.8566	0.8183	0.7095	0.8680
800	0.9807	0.9914	0.9134	0.8912	0.8276	0.9209
1400	0.9880	0.9932	0.9545	0.9380	0.9032	0.9554
2200	0.9923	0.9948	0.9716	0.9615	0.9372	0.9715
3000	0.9940	0.9957	0.9795	0.9725	0.9540	0.9791
4000	0.9948	0.9964	0.9852	0.9802	0.9661	0.9846
5200	0.9955	0.9970	0.9896	0.9859	0.9749	0.9886
H.265						
Kodna brzina [kb/s]	Chimera1102347 H.265	Chimera1102353 H.265	CosmosLaundromat H.265	ElFuenteDance H.265	Soccer H.265	Prosječan SSIM
400	0.9777	0.9913	0.9073	0.8940	0.8124	0.9165
800	0.9863	0.9929	0.9409	0.9371	0.8850	0.9484
1400	0.9904	0.9939	0.9615	0.9608	0.9278	0.9669
2200	0.9926	0.9949	0.9738	0.9740	0.9518	0.9774
3000	0.9938	0.9957	0.9800	0.9806	0.9640	0.9828

4000	0.9947	0.9964	0.9846	0.9850	0.9730	0.9867
5200	0.9954	0.9969	0.9878	0.9881	0.9791	0.9895



Slika 4.3. *Graf SSIM vrijednosti kodiranih videa*

Može se uočiti da H.265 ostvaruje značajno bolju kvalitetu videa za istu kodnu brzinu u većini slučajeva. I PSNR i SSIM prikazuju cjelokupno bolje rezultate za H.265, s malim razlikama između mjera. Zanimljivo je istaknuti da i PSNR i SSIM prikazuju bolju kvalitetu za H.264 i 5200 kb/s kodnu brzinu u slučaju datoteke „CosmosLaundromat“. Nadalje, poboljšanje u kvaliteti koje ostvaruje H.265 u odnosu na H.264 kodiranje za 3 datoteke od 5 smanjuje se kako raste kodna brzina. Sveukupno, H.265 ostvaruje prema PSNR do 3.17 dB, odnosno do 0.0486 prema SSIM rezultatima bolju kvalitetu videa od H.264 za istu kodnu brzinu. Prosječno je kodiranjem s H.265 koderom postignuto poboljšanje od 1.6 dB prema PSNR rezultatima, te 0.0143 prema SSIM rezultatima, što čini H.265 boljim za iskorištavanje ograničenog prostora za pohranu ili prijenosnog pojasa.

5. ZAKLJUČAK

Norma H.265/HEVC ostvarila je velike napretke u efikasnosti kodiranja videa u usporedbi s njenim prethodnikom H.264/AVC. H.265 ne samo ostvaruje nego i premašuje originalni cilj poboljšanja efikasnosti kompresije koji je bio 50% tijekom njegove koncepcije. Time H.265 zadržava bolju kvalitetu videa kada su količina memorijskog prostora za pohranu ili širina prijenosnog pojasa ograničeni. Međutim, za dekodiranje H.265 formata potreban je prikladan H.265 dekoder, te ne postoji podrška unatrag s postojećim H.264 dekoderima. Zbog toga je potrebno investirati resurse za uvođenje podrške za dekodiranje H.265 videa od strane web stranica, programa, uređaja, itd. Danas još uvijek prevladava podrška za H.264 i on kao vodeći standard video formata još nije u potpunosti zamijenjen. Potiče se usvajanje nove norme, jer će time i producenti sve više početi kreirati svoje video datoteke u H.265 formatu. Tako će se smanjiti gubitak kvalitete novih videa jer će kompromis na njoj zbog veličine biti manji. Gubitak podataka je nepovratan, te je od velike važnosti obraćati pažnju na metode koje se koriste pri kompresiji s gubicima.

Dodatno, u sklopu ovog završnog rada je bilo korištenje samo objektivnih mjera za ocjenu kvalitete videa. Iako su one relativno brz i jednostavan način za uspoređivanje vizualne točnosti između originalnih i kodiranih datoteka, nisu uvijek vjeran prikaz kvalitete kako ju percipira čovjek. Zbog toga su subjektivne metode ocjenjivanja važan dio tijekom procesa utvrđivanja efikasnosti novih kodera, te se ne bi smjelo oslanjati samo na objektivne metode. Također, ovaj rad nije obradio usporedbe vremena kodiranja i zahtjeva sklopovlja između H.264 i H.265 kodera, te se ovo sugerira za daljnje istraživanje u ovom području.

LITERATURA

[1] A. Haskell, Hacking FFmpeg With Python – Part One [online], Unixmen, Boston, 2017, dostupno na: https://www.unixmen.com/hacking-ffmpeg-python-one/ [5.9.2020.]

[2] Z. Li, M. S. Drew, J. Liu, Fundamentals of Multimedia Second Edition, Springer, Švicarska, 2014

[3] Z. Wang et al., Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. 13(4), 600-612, 2004

SAŽETAK

U ovom završnom radu opisana je H.265/HEVC normu za kompresiju videa, te njezina poboljšanja u usporedbi sa njenim prethodnikom H.264/AVC. Korištenjem programa FFmpeg, napisan je programski kôd u Python programskom jeziku kojim su kodirane različite video datoteke H.264 i H.265 koderima. Primjenom dvije objektivne metode ocjenjivanja kvalitete videa PSNR i SSIM napravljena usporedba kvalitete komprimiranih videa. Analizom rezultata je utvrđeno da je H.265 postigao prosječno poboljšanje kvalitete od 1.6 dB prema PSNR rezultatima i 0.0143 prema SSIM rezultatima u usporedbi s H.264 za istu kodnu brzinu, što pokazuje da je novi standard uspješno unaprijedio efikasnost kompresije.

VIDEO ENCODING ACCORDING TO H.265/HEVC STANDARD

ABSTRACT

In this final thesis the H.265/HEVC standard for video compression is described, as well as its improvements in comparison with its predecessor H.264/AVC. Using the FFmpeg application, a program code has been written in Python, with which various video files have been encoded by H.264 and H.265 encoders. Two objective methods for video quality measurement, PSNR and SSIM, have been used for comparison of compressed videos quality. Measurement results have shown that H.265 achieved the average quality improvement of 1.6 dB according to PSNR results and 0.0143 according to SSIM results in comparison with H.264 for the same bitrate, which indicates that the new standard has successfully improved compression efficiency.

ŽIVOTOPIS

Luka Mekić-Delić rođen je 1995. godine u Požegi gdje je završio OŠ Antuna Kanižlića i smjer tehničara za mehatroniku u srednjoj školi Tehnička škola Požega. Nakon toga upisao je stručni smjer informatike na fakultetu FERIT, Osijek. Bavi se programiranjem programskim jezicima poput C, C++, C#, Visual Basic, Java, Python, HTML, CSS, PHP, JavaScript, SQL, te je koristio program za kreiranje video igara GameMaker. Komponira vlastite glazbene kompozicije, svira električnu i bas gitaru, te se bavi digitalnim medijima, naročito kreiranjem i obradom zvuka.

PRILOG

P.4.1. Python programski kôd za kodiranje videa koristeći FFmpeg

```
import os
import subprocess
import sys

def deleteFile(file):
    if os.path.exists(file):
        os.remove(file)
    else:
        print("WARNING: The file " + file + " doesn't exist")

def appendToFile(file,string):
    f = open(file, 'a')
    f.write(string)
    f.close()

def insertString(string,insert,index):
        return string[:index] + insert + string[index:]

def encode(ffmpegpath,inputpath,outputpath,codec,profile,bitrates):
    standard = "NULL"
    if codec == "libx264":
        standard = "h.264"
    elif codec == "libx265":
        standard = "h.265"

    for bitrate in bitrates:
        psnr_array = []
        ssim_array = []
        psnrssimfile = outputpath + "/psnr&ssim.log"

        for inputfilename in os.listdir(inputpath):
            encodeparams = "-c:v " + codec + " -profile:v " + profile + " -level:v 4.0 -coder ac -
qcomp 0.9 -qmin 3 -b:v " + bitrate + " -maxrate 20000k -bufsize 40000k"
            if (inputfilename.endswith(".mp4")):
                inputfilename_noext = os.path.splitext(inputfilename)[0]
                inputfile = inputpath + "/" + inputfilename
                outputfilename = inputfilename_noext + "_" + standard + "_" + bitrate + ".mp4"
                outputfile = outputpath + "/" + outputfilename
                outputfile_noext = os.path.splitext(outputfile)[0]

                os.system(ffmpegpath + " -y -i " + inputfile + " " + encodeparams + " -pass 1 -
passlogfile " + outputfile_noext + " -f mp4 NUL")
                os.system(ffmpegpath + " -y -i " + inputfile + " " + encodeparams + " -pass 2 -
passlogfile " + outputfile_noext + " " + outputfile)
                deleteFile(outputfile_noext + "-0.log")
```

```
            deleteFile(outputfile_noext + "-0.log.mbtree")

            process = subprocess.Popen(ffmpegpath + " -y -i " + inputfile + " -i " +
outputfile + " -lavfi \"psnr;[0:v][1:v]ssim\" -f null -",stdout=subprocess.PIPE,stderr =
subprocess.PIPE,stdin=subprocess.PIPE,universal_newlines=True)
            out = ""
            while process.poll() is None:
               line = process.stderr.readline()
               if line != "":
                  print(line, end='')
                  out += line

            psnr_indexstart = out.find("average:",out.find("Parsed_psnr")) + len("average:")
            psnr_indexend = out.find(" ",psnr_indexstart)
            ssim_indexstart = out.find("All:",out.find("Parsed_ssim")) + len("All:")
            ssim_indexend = out.find(")",ssim_indexstart) + 1
            psnr = out[psnr_indexstart:psnr_indexend]
            ssim = out[ssim_indexstart:ssim_indexend]
            psnr_array.append(psnr)
            ssim = insertString(ssim," dB",ssim.find(")"))
            ssim_array.append(ssim)

            appendToFile(psnrssimfile, outputfilename + "\n\tPSNR: " + psnr + "
dB\n\tSSIM: " + ssim + "\n")
         else:
            continue

      psnr_avg = 0.0
      for psnr in psnr_array:
         psnr_avg += float(psnr)
      psnr_avg = round(psnr_avg / len(psnr_array), 6)
      appendToFile(psnrssimfile, "Prosjecni PSNR za " + bitrate + " kodnu brzinu: " +
str(psnr_avg) + " dB\n")

      ssim_pt_avg = 0.0
      ssim_db_avg = 0.0
      for ssim in ssim_array:
         ssim_pt = ssim[0:ssim.find(" ")]
         ssim_db = ssim[ssim.find("(")+1:ssim.find(" ",ssim.find("(")+1)]
         ssim_pt_avg += float(ssim_pt)
         ssim_db_avg += float(ssim_db)
      ssim_pt_avg = round(ssim_pt_avg / len(ssim_array), 6)
      ssim_db_avg = round(ssim_db_avg / len(ssim_array), 6)
      appendToFile(psnrssimfile, "Prosjecni SSIM za " + bitrate + " kodnu brzinu: " +
str(ssim_pt_avg) + " (" + str(ssim_db_avg) + " dB)\n\n")

ffmpegpath = "ffmpeg.exe"
inputpath = "./videoZapisi"
outputpath = "./videoZapisiKodirani"
bitrates = ["400k","800k","1400k","2200k","3000k","4000k","5200k"]
```

```
encode(ffmpegpath,inputpath,outputpath,"libx264","high",bitrates)
encode(ffmpegpath,inputpath,outputpath,"libx265","main",bitrates)

wait = input("Press Enter to continue...")
```