Detektiranje artefakta stvaranja blokova u videu

Jurić, Tomislav

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

Sveučilišni diplomski studij

DETEKTIRANJE ARTEFAKTA STVARANJA BLOKOVA U VIDEU

Diplomski rad

Tomislav Jurić

Osijek, 2016.

Sadržaj

1. UVOD

Zbog velike količine resursa koje zauzima pri pohrani na medij i pri prijenosu mrežom, digitalni video signal potrebno je komprimirati u skladu s određenom normom za kompresiju videa. Zbog samog načina kodiranja, rastavljanja okvira videa (engl. *frame*) na blokove koji se zasebno kodiraju, najčešći artefakt koji se pojavljuje u komprimiranom videu je artefakt stvaranja blokova (engl. *blocking*). U radu je potrebno dati pregled postojećih algoritama za detekciju artefakta stvaranja blokova u videu. Algoritmi koji će se razmatrati nemaju pristup nekomprimiranom video signalu niti ikakvim drugim podacima vezanim za njega.

Cilj ovog rada je napraviti algoritam koji treba detektirati je li se u određenom okviru videa pojavio artefakt stvaranja bloka ili nije te pohraniti lokaciju samog artefakta. Za svaki okvir videa potrebno je izračunati kvantitativnu mjeru pojavljivanja blokova. Potrebno je potvrditi dosljednost algoritma provjerom rezultata za video datoteke istog sadržaja ali različitog stupnja kompresije. Algoritam treba funkcionirati za video signale komprimirane u skladu s MPEG-2 i H.264 normama za razne rezolucije i brzine izmjene okvira.

U drugom poglavlju dan je kratak pregled najčešće korištenih normi za kompresiju digitalnog video signala, objašnjeno je nastajanje artefakta stvaranja blokova te je prezentirana osnovna teorija koja je nužna za potpuno razumijevanje diplomskog rada. U trećem poglavlju prezentiran je rad novo-predloženog algoritma za detekciju artefakta stvaranja blokova u videu. Pojašnjeni su postupci koji su pridonijeli smanjenju vremena izvršavanja algoritma, te je objašnjen postupak skaliranja rezultata algoritma. U četvrtom poglavlju su objašnjene baze testnih video signala, te su prezentirani rezultati novo-predloženog algoritma. Dobiveni rezultati uspoređeni su s rezultatima komercijalnog algoritma koji je u vlasništvu tvrtke Institut RT-RK Osijek d.o.o [1].

2. KOMPRESIJA VIDEA I ARTEFAKT STVARANJA BLOKOVA U VIDEU

U ovom poglavlju dan je kratak pregled najčešće korištenih normi za kompresiju digitalnog video signala, objašnjeno je nastajanje artefakta stvaranja blokova u videu te je dana osnovna teorija koja je nužna za potpuno razumijevanje diplomskog rada. Navedena su neka od postojećih rješenja detekcije i mjerenja artefakta stvaranja blokova u videu.

2.1. Digitalni video signal

Video se sastoji od slijeda okvira koji kada se dovoljno brzo izmjenjuju, gledatelju pružaju osjećaj pokreta. Da bi se kreirala iluzija pokreta potrebno je prikazati najmanje 24 okvira u sekundi (engl. *frames per second – fps*). Prema [2], da bi se spremila 2 sata nekomprimiranog video materijala snimljenog u rezoluciji visoke kvalitete s 1080 horizontalnih i 1920 vertikalnih linija (engl. *Full HD resolution*) koji izmjenjuje 25 okvira u sekundi, a pri tome je svaki element slike kodiran s 12 bita potrebno je 559.9 GB memorije. Ta količina memorije znatno premašuje današnje kapacitete prijenosa i spremanja podataka. Stoga da bi bilo moguće spremiti i prenijeti video datoteke, potrebno ih je komprimirati u skladu s nekom od normi za kompresiju videa.

Kodek (engl. *codec*) je softverska implementacija koja je sposobna kodirati i dekodirati digitalni signal. Ukoliko se promatra kvaliteta kompresije i rekonstrukcija podataka, postoje dvije vrste kodeka – kodeci s gubitcima (engl. *lossy*) i kodeci bez gubitaka (engl. *lossless*). U praksi se koriste obje vrste kodeka ovisno o zahtjevima na sustav, odnosno ciljevima koje je potrebno postići.

Kodeci s gubitcima provode nepovratnu (engl. *irreversible*) kompresiju, što znači da dekompresija predstavlja aproksimaciju originalnog podatka. Na taj se način smanjuje kvaliteta kako bi se postigla kompresija i smanjila veličina podataka, ali su originalni podaci izgubljeni. Ovakva vrsta kompresije koristi se obično za kompresiju video i audio sadržaja pri prijenosu mrežom. Iako je sadržaj izgubio određenu količinu informacija to ne znači da će korisnik uvijek primijetiti gubitke.

Kodeci bez gubitaka provode reverzibilnu kompresiju. Na taj način su originalni podaci sačuvani bez gubitaka i oni se dobivaju dekodiranjem. Ovakva vrsta kompresije koristi se kada je naglasak na čuvanju originalnih podataka, a ne na smanjenju veličine podataka. Najčešće se primjenjuje na tekstualne datoteke i proračunske tablice.

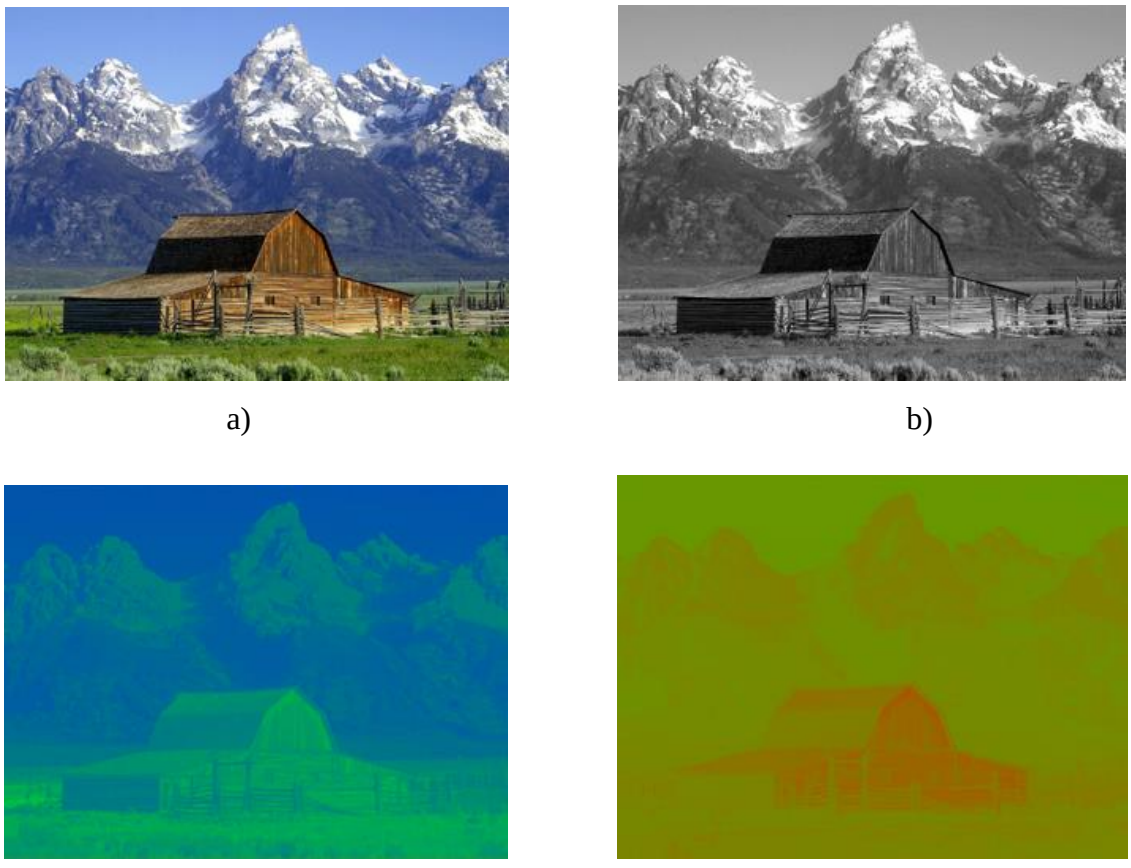
Algoritmi za obradu videa i slike najčešće obrađuju samo Y komponentu YUV prostora boja koja predstavlja svjetlinu (engl. *luminance*). Razlog tomu je što je ljudsko oko puno osjetljivije na promjenu intenziteta odnosno svjetline, nego na promjenu boje. Gledajući crno-bijelu sliku umjesto iste slike u boji, ljudi su u stanju prepoznati sve objekte sa slike. U i V su krominantne komponente i one prenose boju. YUV signali najčešće nastaju iz RGB (engl. *red, green, blue*) signala upotrebom linearnih transformacija. Formule (3-1), (3-2) i (3-3) su izrazi za izračun komponenti YUV prostora boja iz komponenti RGB modela boja.

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \quad (3\text{-}1)$$

$$U = 0.492 * (B - Y) \quad (3\text{-}2)$$

$$V = 0.877 * (R - Y) \quad (3\text{-}3)$$

Slika 2.1. sastoji se od 4 prikaza istog sadržaja gdje su prikazani originalni sadržaj, Y komponenta, U komponenta i V komponenta.



a) b)

c) d)

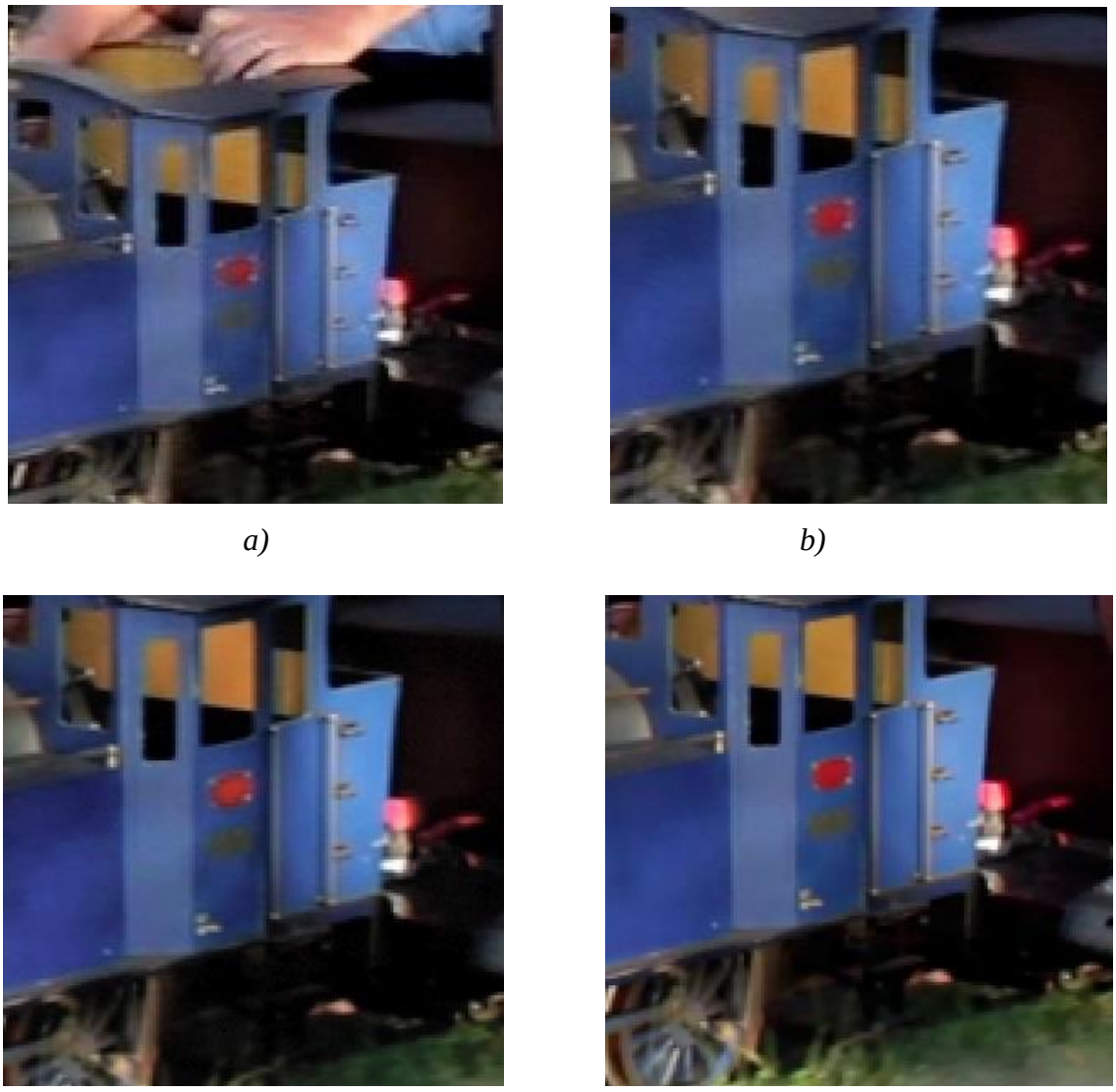
Sl. 2.1. *Prikaz sadržaja upotrebom:*
(a) RGB modela (b) Y komponente
(c) U komponente (d) V komponente YUV prostora boja [3].

Zbog činjenice kako je ljudsko oko puno osjetljivije na svjetlinu nego na boju, uvedena je praksa poduzorkovanja krominantnih komponenti (engl. *color subsampling*). Provođenjem poduzorkovanja smanjuje se količina podataka koju je potrebno poslati. Umjesto vrijednosti boja svih elemenata slike (engl. *pixel*), šalju se samo određene vrijednosti, ovisno o tipu poduzorkovanja. Glavni tipovi poduzorkovanja boja su:

- 4:4:4 – sve tri komponente se prenose u punoj rezoluciji, što znači da zapravo i nema poduzorkovanja; prenose se sve dostupne informacije
- 4:2:2 – obje krominantne komponente su poduzorkovane tako da se prenosi polovica rezolucije; poduzorkovanje se vrši u horizontalnom smjeru; ovaj tip poduzorkovanja smanjuje količinu informacija za 33.33%.
- 4:2:0 – obje krominantne komponente su uzorkovane na taj način da se prenosi četvrtina rezolucije, poduzorkovanje se vrši i horizontalno i vertikalno; ovo je najkorišteniji tip poduzorkovanja, smanjuje količinu informacija za 50%
- 4:1:1 – kao i kod 4:2:0, prenosi se četvrtina rezolucije za krominantne komponente, količina informacija je smanjena za 50%, ali se poduzorkovanje vrši samo horizontalno

Slika 2.2. prikazuje princip funkcioniranja navedenih tipova poduzorkovanja. Slika 2.3. se sastoji od 4 prikaza istog sadržaja nakon primjene poduzorkovanja boja.

Sl. 2.2. *Princip funkcioniranja glavnih tipova poduzorkovanja krominantnih komponenti* [4].



a) *b)*

c) *d)*

Sl. 2.3. *Izgled sadržaja ovisno o tipu poduzorkovanja:*
(a) poduzorkovanje 4:1:1 (b) poduzorkovanje 4:2:0
(c) poduzorkovanje 4:2:2 (d) bez poduzorkovanja [5].

2.2. Norme za kompresiju videa

Kompresija je postupak smanjenja veličine datoteke. Kako bi se video datoteka komprimirala, norme za kompresiju videa iskorištavaju vremensku i prostornu zalihost. Prostorna zalihost predstavlja činjenicu da elementi slike koji su prostorno blizu imaju slične vrijednosti, stoga ukoliko postoji crveni element slike u nekom od okvira, za očekivati je da su elementi u njegovoj okolini ili svijetlo crveni ili tamno crveni. Vremenska zalihost predstavlja činjenicu da su uzastopni okviri vrlo slični, često su gotovi i identični, osim što su pomaknuti zbog pomjeranja objekata. Vremenska zalihost se iskorištava tako što se sprema razlika vrijednosti elemenata slike u uzastopnim okvirima. Spremljena razlika je ispravljena za onu količinu pokreta koja se dogodila.

Budući da su spremljene razlike male, moguće ih je kodirati s manjim brojem bita. Ovaj proces je poznat pod nazivom kompenzacija pokreta (engl. *motion compensation*), dok je iznos pokreta poznat kao vektor pokreta (engl. *motion vector*).

Makroblok je osnovna jedinica za predikciju pokreta kompenzacijom, u MPEG-2 je njegova veličina 16x16 elemenata slike. Sastoji se od 16x16 polja za svjetlinu a za boju s 4:2:0 poduzorkovanjem sadrži 2 bloka od 8x8 elemenata. Iako prvenstveno ovisi o sadržaju video datoteke, općenito vrijedi pravilo što je veći broj okvira u sekundi, veća je vremenska zalihost, budući da su okviri snimljeni u manjem vremenskom razmaku. Iskorištavanjem prostorne i vremenske zalihosti, moguće je smanjiti broj bitova koji su potrebni za kodiranje elementa slike (engl. *bits per pixel – bpp*), što rezultira manjom veličinom video datoteke.

S obzirom na stupanj kompresije okvira, postoje tri tipa okvira: I okvir (engl. *intracoded frame*), P okvir (engl. *predicted frame*) i B okvir (engl. *bidirectional frame*). I okvir, koji služi za sinkronizaciju ima najniži stupanj kompresije i on se komprimira samo na temelju podataka koje on sadrži. Stoga se prilikom njegova kodiranja ne može iskoristiti vremenska zalihost podataka. P i B okviri nastaju međuokvirnim kodiranjem i iskorištavaju i prostornu i vremensku zalihost podataka. Razlika između i P i B okvira je što se P okvir kodira iz prethodnih I ili P okvira, dok se B okvir uz to može kodirati i iz sljedećih I i P okvira. B okvir osigurava dobru predikciju dijelova okvira koji ne postoje u prethodnim okvirima i ima najviši stupanj kompresije. Iako korištenje B okvira povećava količinu proračuna, znatno povećava kvalitetu videa.

Svaka od normi za kompresiju videa na jezgru algoritma nadovezuje profile (engl. *profiles*) i razine (engl. *levels*). Profili definiraju funkcionalnosti, a razine definiraju vrijednosti parametara, poput maksimalne rezolucije ili brzine izmjene okvira koje je moguće postaviti korištenjem te razine prilikom kodiranja sadržaja.

MPEG-2, AVC/H.264 (engl. *Advanced Video Coding*) i HEVC/H.265 (engl. *High Efficient Video Coding*) su norme za kompresiju videa koje su nastale suradnjom VCEG (engl. *Video Coding Experts Group*) i MPEG (engl. *Motion Pictures Experts Group*) grupa. VCEG grupa je ekspertna skupina unutar ITU-T sektora koji koordinira standarde za telekomunikacije, a dio je ITU organizacije (engl. *International Telecommunication* Union) [6]. Partnerstvo VCEG i MPEG grupi koje je razvilo H.264 normu poznato je pod nazivom JVT (engl. *Joint Video Team*), što je ujedno i manje poznato ime za H.264 normu, a koristi se kako bi se referenciralo na grupu koja ga je razvila [7].

U siječnju 2010. godine osnovana je nova grupa, JCT-VC (engl. *Joint Collaborative Team on Video Coding*) koja započinje rad na H.265 normi, a sastoji se od MPEG i VCEG stručnjaka. Prema [6], H.265 norma se razvija s namjerom smanjenja pritiska na globalne mreže, te kako bi uvela svijet u eru digitalne televizije ultra visoke kvalitete (engl. *Ultra High Definition Television - UHD*).

Za korištenje MPEG-2 i H.264 patenata potrebno je platiti pristojbu za licencu (engl. *license fee*) i naknadu za autorsko pravo (engl. *royalty fee*). Za izdavanje licence zadužena je tvrtka MPEG LA [8]. U dokumentu navedenom na [9] mogu se pronaći određene brojke koje definiraju brojnost uređaja i cijenu koji se odnose na H.264 normu. Na [10] se nalaze isti podaci, ali za MPEG-2 normu.

U budućnosti će neki od glavnih suparnika gore navedenim normama biti proizvod Alliance for Open Media organizacije. Organizaciju su osnovali: Google, Cisco, Amazon, Microsoft, Intel, Mozilla i Netflix. Popis svih članica moguće je vidjeti na [11]. Cilj ove organizacije je izrada novih normi otvorenog koda (engl. *open source*) za kompresiju video sadržaja za koji nije potrebno plaćati licencu (engl. *royalty-free*). Današnji suparnici su norme VP8 i VP9 koje je razvio Google koristeći sestrinsku tvrtku WebM Project [12]. Obje norme su otvorenog koda i potpuno su besplatne. Prije formiranja Alliance for Open Media organizacije, Cisco i Mozilla su krenuli u izradu svojih potpuno besplatnih normi otvorenog koda. Ciscov projekt se naziva Thor, a prema [13] glavni razlozi pokretanja su: i do 16 puta skuplje licenciranje za H.265 nego za H.264 (po jedinici), i nepostojanje gornje granice za troškove godišnjeg licenciranja za H.265, dok H.264 ima granicu. Mozilla je svoj projekt nazvala Daala.

Iako VP9, koji se natječe s H.265 i VP8, koji se natječe s H.264, još nisu u rangu svojih suparnika, s obzirom na znanja i dostignuća članica organizacije Alliance for Open Media, za očekivati je da će AOMedia Video 1 (AV1) norma koju razvijaju promijeniti odnos snaga nakon izlaska na tržište.

2.2.1. ISO/IEC 13818-2

ISO/IEC 13818-2 je norma za generičko kodiranje videa koja podržava širok spektar primjena, razne brzine prijenosa (engl. *bit rate*), rezolucije okvira i kvalitete. Ova norma je poznatija pod imenom MPEG-2 i prva verzija je izdana 1996. godine.

MPEG-2 norma vrši podjelu okvira u 8x8 blokove, gdje svaki element slike pripada točno jednom bloku. Na svaki od tih blokova se zatim zasebno primjenjuje DCT transformacija (engl.

Discrete Cosine Transform) čiji se rezultati kvantiziraju. Dobiveni koeficijenti se organiziraju korištenjem zig-zag uzorka kako bi se dobio dugi niz nula što omogućuje entropijskom kodiranju učinkovitiju kompresiju.

2.2.2. ISO/IEC 14496-10

ISO/IEC 14496-10 je norma za napredno video kodiranje za općenite audiovizualne usluge, a poznatija je pod nazivima H.264 i AVC. Prva verzija je odobrena 2003. godine.

H.264, nasljednik MPEG-2 norme uvodi značajke koje omogućavaju efikasniju kompresiju video sadržaja. Neke od najbitnijih značajki su:

- kodiranje u odnosu na više referentnih okvira – norma dozvoljava da se prilikom kodiranja predviđenih okvira na raspolaganju nalazi do 16 referentnih okvira u progresivnom kodiranju (engl. *progressive*) ili 32 polja ukoliko se radi o isprepletenom kodiranju (engl. *interlaced*). U prethodnim normama ta vrijednost obično iznosi 1, osim kodiranja B okvira gdje iznosi 2.
- varijabilna veličina bloka – veličine bloka za kodiranje svjetline mogu poprimiti neku od sljedećih vrijednosti: 4x4, 8x8, 16x16, 4x8, 8x4, 8x16 i 16x8. Ovime je poboljšana kvaliteta slike i povećan stupanj kompresije. Na velikoj uniformnoj površini koristiti će se blokovi koji se sastoje od više elemenata slike, dok će se na dijelovima okvira gdje se nalazi dosta detalja, poput prirodnih bridova, koristiti blokovi koji sadrže manje elemenata slike čime se postiže jasnija slika s manje degradacije.
- mogućnost korištenja višestrukih vektora pokreta za makroblok – vektori pokreta za 8x8 i veće regije mogu pokazivati na različite referentne okvire.
- filtar za smanjenje artefakta stvaranja blokova (engl. *deblocking filter*) – je video filtar koji se primjenjuje na komprimirani video signal kako bi se povećala kvaliteta i sposobnost predviđanja zaglađivanjem oštrih rubova koji se mogu pojavljivati na granicama makroblokova. Filtar je primijenjen na okvir prije nego se okvir koristi kao referentni okvir i moguće ga je primijeniti na sve tri komponente YUV prostora boja.
- nova entropijska kodiranja: CABAC (engl. *Context-adaptive binary arithmetic coding*), CAVLC (engl. *Context-adaptive variable-length coding*) i Exp-Golomb (engl. *Exponential Golomb coding*)
- uporaba cjelobrojne transformacije umjesto DCT transformacije

2.2.3. ISO/IEC 23008-2

ISO/IEC 23008-2 je norma za visoko efikasno kodiranje video sadržaja. Norma je poznatija pod nazivima H.265, MPEG-H Part 2 i HEVC. Prva verzija je odobrena 2013 godine.

Najbitnija poboljšanja i promjene u odnosu na H.264 normu su:

- podržana rezolucija – H.265 podržava rezolucije do 8K UHD sa 8192x4320 elemenata slike
- veličina makroblokova – H.265 uvodi veličinu makrobloka do 64x64 elementa slike
- 35 modova predikcije prilikom unutarokvirnog kodiranja naspram 9 modova H.264 norme
- SAO filtar – H.265 uvodi SAO filtar (engl. *Simple Adaptive Offset*) – proces koji neutralizira vrijednosti elementa slike na temelju vrste brida artefakta i vrijednosti elementa slike
- preciznije kodiranje vektora pokreta
- poboljšani filtar za smanjenje artefakta stvaranja blokova
- podržan je maksimalan iznos od 300 okvira u sekundi za kodiranje 4K DCI video sadržaja (engl. *Digital Cinema Initiatives*) rezolucije 4096x2160 elemenata slike

2.3. Nastajanje artefakta stvaranja blokova u videu

Zbog načina kodiranja, rastavljanja okvira videa na blokove koji se zatim zasebno kodiraju, artefakt stvaranja blokova u videu je najčešći artefakt. Artefakt stvaranja blokova nastaje usred prevelike kompresije video sadržaja. Stupanj kompresije koji se može primijeniti na video datoteku prije pojave artefakta stvaranja blokova ovisi o više faktora od kojih se izdvajaju kompleksnost sadržaja i način snimanja. Budući da se blokovi kodiraju zasebno, ukoliko je na njihovim granicama vidljiv diskontinuitet u svjetlini, kao što je prikazano na slikama 2.4. i 2.5., dolazi do pojave artefakta stvaranja blokova. Osim artefakta stvaranja blokova postoje i artefakti gubitka paketa (engl. *packet loss*), zamrzavanja (engl. *freezing*), kidanja (engl. *tearing*), zamućenja (engl. *blurring*). Primjere i detaljniji opis tih artefakata moguće je vidjeti na [14].

Sl. 2.4. *Okvir video datoteke kodirane u skladu s MPEG-2 normom s vrlo visokim stupnjem kompresije.*

Sl. 2.5. *Okvir video datoteke kodirane u skladu s H.264 normom s vrlo visokim stupnjem kompresije.*

2.4. Pregled postojećih rješenja za detekciju artefakta stvaranja blokova

Svrha ovog algoritma je određivanje dijela okvira koji je zahvaćen artefaktom stvaranja blokova. Budući da algoritam vrši objektivno mjerenje, potrebno ga je usporediti s algoritmima koji funkcioniraju na sličan način. Većina algoritama, poput [15] i [16] vrše ocjenu kvalitete videa, odnosno računaju utjecaj artefakta stvaranja blokova na kvalitetu slike. Takvi algoritmi se često temelje na posebnoj strukturi informacija o artefaktima koju je moguće povezati s ljudskim vizualnim sustavom (engl. *human visual system – HVS*).

U radu [17] prezentiran je algoritam koji vrši objektivno mjerenje artefakta stvaranja blokova. Temeljna ideja algoritma je promatrati komprimirani okvir videa kao spoj nekomprimiranog okvira i signala pogreške. Primjenom signala pogreške na nekomprimirani video okvir pojavljuju se artefakti stvaranja blokova. Zadatak algoritma je detektirati signal pogreške i izmjeriti njegov iznos. Utjecaj signala se zasebno mjeri u horizontalnom i vertikalnom smjeru. Pretpostavljena je veličina bloka, B=8. Na vrijednost signala pogreške utječe slučajna varijabla V. Vrijednost idealnog signala pogreške ne bi trebala ovisiti o V, tako da je prvi korak računanje aposlutnih razlika duž signala pogreške. Tim izračunom dobivene su vertikalna i horizontalna razlika okvira. Primjenom Fourierove transformacije i prelaskom u frekvencijsku domenu procjenjuje se spektar snage (engl. *power spectrum*) koji se rastavlja na L segmenata čija je duljina potencija broja 2. Kako bi se procijenila ukupna snaga spektra primijenjen je median filtar čija je svrha izgladiti krivulju snage spektra prije konačnog izračuna. Pretpostavka je da su vertikalni i horizontalni učinci artefakta stvaranja blokova jednake važnosti.

Komercijalni alat *MSU Video Quality Measurement Tool* sadrži MSU Blocking metriku koja mjeri kvalitetu videa mjereći količinu artefakta stvaranja blokova [18]. Navedena metrika ne vrednuje sve blokove istom vrijednošću. Bitno je nalazi li se blok na uniformnom dijelu okvira ili ne. Metrika se koristi podacima iz prethodnih okvira kako bi postigla veću preciznost.

3. NOVI ALGORITAM ZA DETEKCIJU ARTEFAKTA STVARANJA BLOKOVA U VIDEU

Ovisno o dostupnim podacima, algoritmi kojima se obrađuju slike ili video sadržaj dijele se u tri glavne skupine [19]. Tako se razlikuju algoritmi koji koriste:

- Obradu s potpunom referencom (engl. *Full-reference processing*)
- Obradu s djelomičnom referencom (engl. *Reduced-reference processing*)
- Obradu bez reference (engl. *No-reference processing*)

Obrada s potpunom referencom je način obrade u kojem algoritam na raspolaganju ima izvorni odnosno nekomprimirani okvir videa ili sliku. U praksi se podrazumijeva da izvorni podaci imaju savršenu kvalitetu. Zbog najveće količine informacija koja im je dostupna, ovakve metode na raspolaganju imaju najširi raspon alata i općenito daju najbolje rezultate. Problem kod takvih metoda je što algoritam na raspolaganju mora imati i originalni i komprimirani sadržaj, a to u nekim slučajevima nije moguće. Takva vrsta obrade zahtjeva veću količinu memorije na raspolaganju u usporedbi s preostala dva načina obrade.

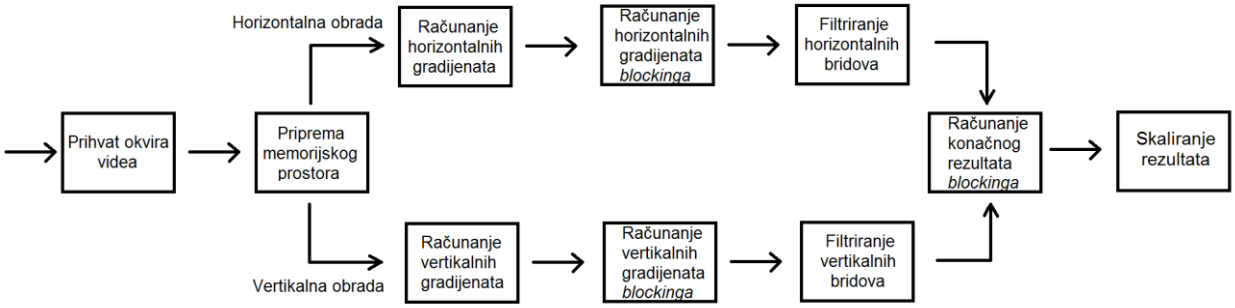
Obrada s djelomičnom referencom je dizajnirana na taj način da može predvidjeti ili izmjeriti kvalitetu komprimiranih okvira ili slika imajući na raspolaganju djelomične podatke o referentnom okviru ili slici i komprimirani sadržaj.

Obrada bez reference je način obrade u kojem algoritam ima pristup isključivo komprimiranim podacima. Ukoliko bi na raspolaganju bila dva algoritma koji vrše istu obradu, algoritam A1 koji vrši obradu s potpunom referencom i algoritam A2 koji vrši obradu bez reference, da bi ta dva algoritma davali vrlo slične rezultate, kompleksnost algoritma A2 je višestruko veća od kompleksnosti algoritma A1. Razlog tomu je ograničenost sa strane dostupnih podataka, i često je gotovo nemoguće napraviti da algoritam A2 daje vrlo slične rezultate kao algoritam A1.

Algoritam za detekciju artefakta stvaranja blokova izrađen u sklopu ovog diplomskog rada pripada u skupinu algoritama koji vrše obradu bez reference. Obrada se vrši na luminantnoj komponenti koja odgovara Y komponenti iz YUV prostora boja, a algoritam ne koristi nikakve pomoćne biblioteke ili slična programska rješenja. Zbog samog načina kodiranja i sastavljanja blokova u okvir videa, kako bi se dobila potpuna informacija o prisustvu artefakta stvaranja blokova, okvir je potrebno obraditi i horizontalno i vertikalno.

3.1. Način rada novo-predloženog algoritma za detekciju artefakta stvaranja blokova

Na slici 3.1. vidljiv je blok dijagram algoritma na kojemu su prikazane etape izvršavanja istoga. Budući da se izračuni vrše u dva smjera – horizontalno i vertikalno, programsko rješenje u potpunosti odgovara slici 3.1., što znači da se obrada u jednom smjeru izvršava u glavnoj niti, a obrada u drugom smjeru u pomoćnoj niti. Prije samog algoritma potrebno je provesti određene radnje kao što su primitak okvira videa i podešavanja memorijskog prostora, koje jamče ispravnost konačnog rezultata. Prije računanja konačnog rezultata, potrebno je osigurati da algoritam ima sve podatke na raspolaganju, odnosno potrebno je pričekati da se završe obrade u obje niti, te je prije izvršavanja zadnje funkcije programski implementirana sinkronizacija.



Sl. 3.1. *Blok dijagram algoritma.*

3.1.1. Računanje gradijenata

U sklopu ovog dijela algoritma izvode se dvije funkcije koje računaju gradijente između dva susjedna elementa slike. Budući da postoje dva smjera obrade, postoje i dvije matrice, *M1* i *M2*, u koje se zapisuju dobiveni rezultati. Formula (3-1) prikazuje izračun horizontalnih gradijenata.

$$M1_{i,j} = P_{i,j-1} - P_{i,j} \qquad \text{i = 1,..H, j = 1,..W} \qquad (3\text{-}1)$$

gdje je:

- *M1* – vrijednost horizontalnog gradijenta na lokaciji (i,j)
- *P* – vrijednost elementa slike na naznačenoj lokaciji
- *H* – broj horizontalnih linija okvira videa
- *W* – broj vertikalnih linija okvira videa
- *i* – redni broj horizontalne linije okvira koja se trenutačno obrađuje, poprima vrijednost od 1 do H

– j – redni broj vertikalne linije okvira koja se trenutačno obrađuje, poprima vrijednost od 1 do W

Izraz za izračun vertikalnih gradijenata je prikazan je formulom (3-2).

$$M2_{i,j} = P_{i-1,j} - P_{i,j} \tag{3-2}$$

gdje je:

– *M2* – vrijednost vertikalnog gradijenta trenutno obrađivanog elementa slike

3.1.2. Računanje gradijenata za nastanak artefakta stvaranja blokova

Ova funkcija koristi prethodno dobivene matrice *M1* i *M2*, te za svaki element slike računa gradijent za nastajanje artefakta stvaranja blokova. Gradijent za artefakt se računa za svaki element slike i to tako da se od gradijenta trenutno obrađivanog elementa oduzmu lijevi i desni susjedni gradijenti ukoliko se radi o horizontalnoj obradi, odnosno gornji i donji susjedni gradijenti ukoliko se radi o vertikalnoj obradi. Formula (3-3) prikazuje izračun horizontalnog gradijenta za artefakt stvaranja blokova, a formula (3-4) prikazuje izračun vertikalnog gradijenta za artefakt stvaranja blokova.

$$HBG = M1_{i,j} - M1_{i,j-1} - M1_{i,j+1} \tag{3-3}$$

gdje je:

– *HBG* – vrijednost horizontalnog gradijenta za stvaranje blokova na naznačenoj lokaciji

$$VBG = M2_{i,j} - M2_{i-1,j} - M2_{i+1,j} \tag{3-4}$$

gdje je:

– *VBG* – vrijednost vertikalnog gradijenta za stvaranje blokova na naznačenoj lokaciji

Da bi se element slike smatrao dijelom brida bloka koji je nastao kao artefakt, gradijent za artefakte (*HBG*, *VBG*) mora biti veći od postavljenog praga *K1*, te istovremeno gradijent tog elementa zapisan u matrici *M1* ili *M2* – ovisno o kojem smjeru obrade se radi, mora biti manji od praga *K2*. Prag *K1* ima ulogu filtriranja šuma, dok prag *K2* ima ulogu filtriranja prirodnog brida.

Rezultati i vertikalne i horizontalne obrade zapisuju se u matricu *M3*, na način da je osigurano da prepisivanje ne može prouzročiti gubitak ili krivo tumačenje podataka. Algoritam je dizajniran

na taj način kako bi se ubrzala obrada podataka. U etapi „Priprema memorijskog prostora“ prikazanoj na slici 3.1. napravljen je bitan preduvjet kako bi se ovakav način zapisivanja rezultata mogao provesti.

Primarni razlog zapisivanja u istu matricu je smanjenje vremena obrade. Ukoliko bi se koristile dvije matrice, jedna za horizontalnu obradu i jedna za vertikalnu obradu, tada je potrebno napraviti uniju tih matrica što znači da se s paralelne obrade prelazi na serijsku, što povećava vrijeme obrade okvira. Još jedan nedostatak ovakve obrade je memorijski prostor. Ukoliko se obrađuje video ultra visoke kvalitete, rezolucije 4096x2160 elemenata slike, a to je definirani maksimum obrade ovog algoritma, onda je potrebno 8.4375 MB više raspoložive memorije u početku rada algoritma. Iako to u današnje vrijeme nije velika količina, to nije pravilna programerska praksa ni sa strane memorijskih zahtjeva ni sa strane vremena obrade.

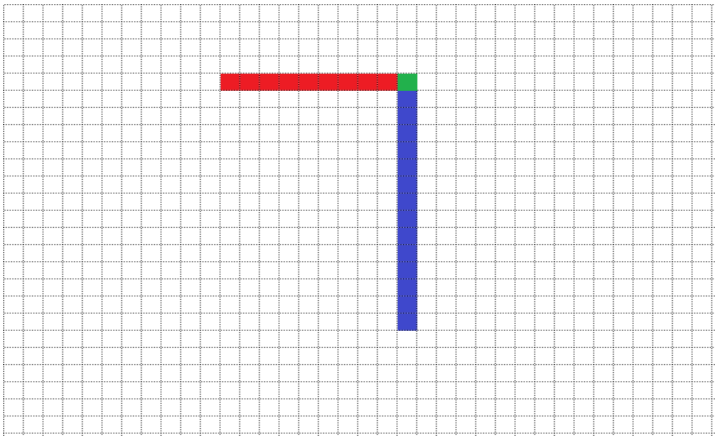
Sekundarni razlog ovakve obrade su rubni slučajevi detekcije koji će biti objašnjeni u sljedećem potpoglavlju nakon uvođenja potrebnih dodatnih oznaka.

3.1.3. Filtriranje bridova

Ova funkcija ima za zadaću filtrirati bridove koji su pronađeni u prethodnoj funkciji. Horizontalna funkcija filtriranja iterira kroz matricu *M3* tako što obrađuje red po red okvira videa i broji na koliko je uzastopno mjesta zapisana informacija o položaju brida artefakta. Prilikom daljnje obrade koriste se pragovi *K3* i *K4*. Ukoliko je pronađeno više od *K3* a manje od *K4* uzastopnih položaja brida artefakta tada se u matricu *M4* postavljaju jedinice na detektirane lokacije. Jedinice se nalaze na položajima elemenata slike koji se nalaze na bridu artefakta stvaranja bloka. Analogno horizontalnoj funkciji radi i vertikalna funkcija koja također zapisuje u matricu *M4*, ali okvir videa obrađuje stupac po stupac.

Kao što je navedeno u prethodnom potpoglavlju ovdje će biti objašnjen sekundarni razlog korištenja samo jedne matrice u prethodnoj funkciji. Jedan od primjera je slijedeći: ukoliko je u vertikalnoj obradi pronađeno *K3* – 1 uzastopnih bridova, da se ne koristi jedna matrica ta detekcija bi se odbacila. Gore objašnjeno rješenje ostavlja prostor kako bi se taj zadnji brid detekcije dobio iz druge vrste obrade, u ovom slučaju horizontalne. Ovakvim rješenjem se poboljšava detekcija, više kvalitativno nego kvantitativno. Ukoliko su pronađeni i horizontalni i vertikalni bridovi koji su povezani preko tog jednog brida, takvi bridovi imaju veći utjecaj na ljudsku percepciju budući da su vezani i više utječu na kvalitetu slike od pojedinačnih detekcija. Sve četiri kombinacije su pokrivene ovakvom implementacijom, a to su: vertikalnom obradom je pronađeno K3 – 1 bridova a horizontalnom obradom je pronađen brid ili s lijeve ili s desne strane detektiranih bridova. Vrijedi

analogno pravilo u slučaju zamjene smjera obrade. Ova situacija je prikazana na slici 3.2., gdje svako polje predstavlja jedan element slike. U ovom slučaju prag *K3* iznosi 10, dok vrijednost praga *K4* iznosi 20. Na slici 3.2., crveno označeni elementi prikazuju vertikalno detektirane bridove dok plavo označeni elementi predstavljaju horizontalno detektirane bridove. Element označen zelenom bojom također je detektiran horizontalnom obradom. Sa slike se uočava da broj vertikalno uzastopno detektiranih bridova iznosi 9, odnosno *K3* – 1, dok horizontalnih ima 15 što je više od *K3* i manje od *K4* za ovaj slučaj. Zbog ovakve obrade i horizontalne detekcije elementa obojanog zelenom bojom, rezultati vertikalne detekcije neće bit odbačeni.



Sl. 3.2. *Primjer detekcije artefakta stvaranja blokova koja koristi implementiranu logiku i ne odbacuje stvarni artefakt.*

Budući da se rezultati ove funkcije zapisuju u istu matricu, matricu *M4*, rubni bridovi kao što je zeleno područje (Sl. 3.2.) će se dva puta zapisati, odnosno njegova vrijednost će se prepisati, ali je to zanemarivo i sa stajališta vremena obrade s obzirom na gore navedene prednosti koje ovakva vrsta obrade donosi. Ova činjenica nema utjecaja na točnost rezultata.

3.1.4. Računanje konačnog rezultata

U ovoj funkciji, zadnjoj koju algoritam izvršava, koristi se prag *K5* koji definira veličinu bloka za taj okvir. Prvo se izvršava izračun veličine rešetke (engl. *grid*). Veličina rešetke izražena je brojem elemenata slike i predstavlja teorijski maksimum broja elemenata koji mogu činiti artefakte stvaranja blokova za taj okvir videa. Izračun veličine rešetke vrši se na temelju rezolucije okvira i praga *K5*.

Prvo je potrebno izračunati broj horizontalnih linija čiji elementi mogu tvoriti brid bloka po formuli (3-5). Analogno tome se računa broj vertikalnih linija po formuli (3-6).

$$HL = \frac{W}{K5} - 1 \tag{3-5}$$

gdje je:

- *HL* – broj horizontalnih linija

$$VL = \frac{H}{K5} - 1 \tag{3-6}$$

gdje je:

- *VL* – broj vertikalnih linija

Prilikom izračuna veličine rešetke, potrebno je prebrojati broj elemenata slike koji se nalaze na horizontalnim i vertikalnim linijama, te oduzeti elemente koji su dva puta prebrojani, a to su oni elementi koji se nalaze na sjecištima linija. Za izračun veličine rešetke koristi se formula (3-7).

$$GS = HL * W + VL * H - HL * VL \tag{3-7}$$

gdje je:

- *GS* – veličina rešetke

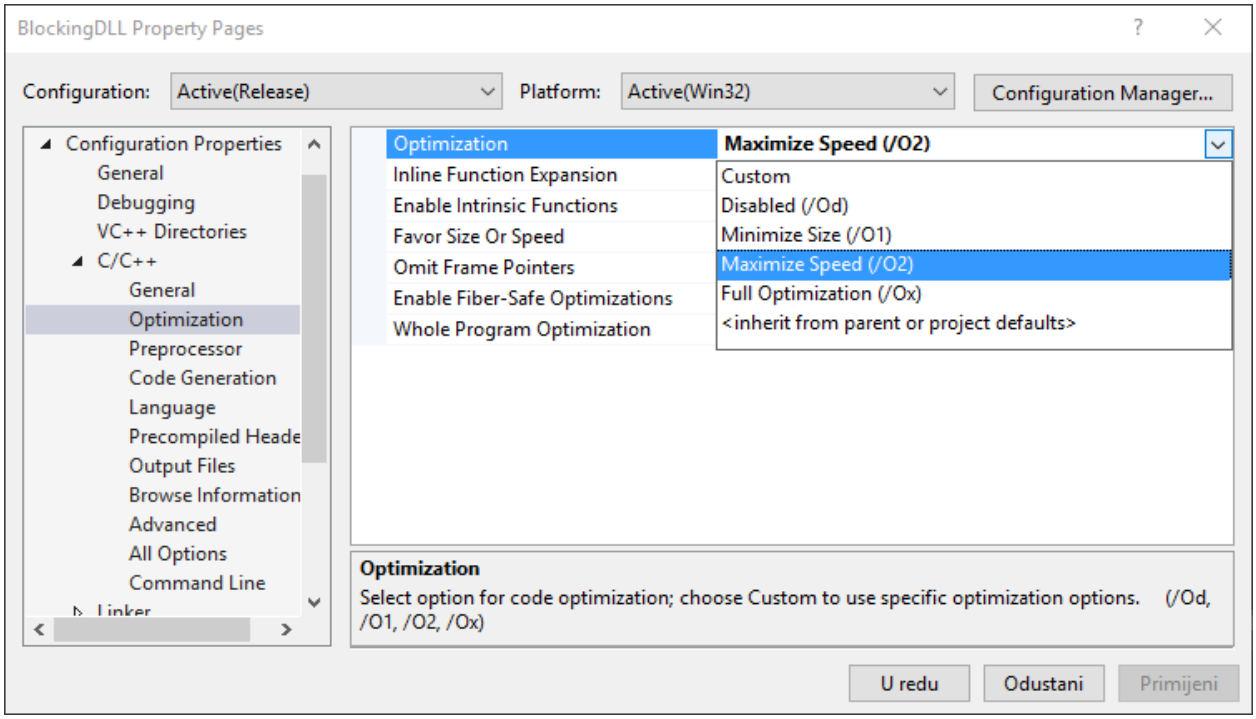
Nakon izračuna veličine rešetke, iterira se kroz matricu *M4* koja se sastoji od *W*x*H* elemenata. U toj su matrici zapisani elementi slike koji čine artefakte stvaranja blokova, te ih je potrebno prebrojiti. Vrijednost dobivena izračunom dijela rešetke koju rubovi blokova zauzimaju u odnosu na maksimalnu veličinu rešetke skalirana je koristeći jedan od tri matematička modela koja su objašnjena u podpoglavlju 3.3. Programski je osigurano da povratna vrijednost ne može iznositi više od 100.00.

Bitno je napomenuti da je prije ove funkcije implementirana sinkronizacija kako bi se osigurao završetak prethodnih obrada i u glavnoj i u pomoćnoj niti. Ukoliko sinkronizacija nije implementirana, za višestruke obrade istog okvira videa, dobivali bi se različiti rezultati, odnosno nema garancije konzistentnosti rezultata. Razlog zbog kojeg bi se to događalo je dodjeljivanje procesorskog vremena svim procesima na računalu, uključujući i sam algoritam. To bi rezultiralo nekada bržom, nekada sporijom obradom jedne ili obje od niti, što u konačnici rezultira nepotpunim rezultatima koji su potrebni ovoj funkciji za izračune.

3.2. Optimizacija vremena izvođenja algoritma

Prilikom razvoja algoritma korišten je MATLAB [20] jer je nužno vizualno verificirati rad algoritma. Nakon što je napravljena konačna verzija algoritma, kod je prepisan u C programski jezik. Zbog drugačijih zaokruživanja MATLAB-a i C-a te rada s matricama zbog kojih se MATLAB i koristi, postojala su odstupanja te kodovi nisu vraćali identične rezultate. Nakon što je to popravljeno nastala je prva verzija C koda koja radi identično kao i MATLAB verzija.

Kako bi algoritam bio primjenjiv u praksi, bilo je nužno smanjiti vrijeme obrade okvira videa. Zbog gore navedenih razloga, algoritam u C-u je pisan u *Debug* verziji kojom je olakšano pronalaženje pogrešaka. Prvi korak u optimizaciji je bilo korištenje *Release* verzije, verzije koja je načelno spremna za izlazak na tržište. Glavna razlika između tih verzija je što su u *Release* verziji uključene optimizacije prevoditelja (engl. *compiler*). Budući da je C kod pisan u Visual Studio 2013 [21] integriranom razvojnom okruženju (engl. *Integrated Development Environment - IDE*), proizvodu tvrtke Microsoft, korišteni prevoditelj je izuzetno kvalitetan, kompleksan i moćan. Opcije koje su ponuđene prilikom optimizacije prevoditelja prikazane su na slici 3.3.



Sl. 3.3. *Ponuđene optimizacije prevoditelja u razvojnom okruženju Visual Studio 2013.*

Prilikom odabira optimizacije prevoditelja, dva su glavna načina na koji je moguće optimizirati kod, odnosno datoteke koje se generiraju nakon što prevoditelj odradi svoju zadaću. Prva opcija

je usmjerena na smanjenje veličine datoteka koje nastaju nakon prevođenja, dok je druga opcija usmjerena na ubrzanje izvođenja koda, upravo ono što je nužno postići da bi algoritam bio primjenjiv u praksi.

Prelazak na *Release* verziju bitno je ubrzao obradu okvira videa. Iako postoji još razlika između verzija, one ne utječu na vrijeme obrade. Jedna od tih razlika je da u *Release* verziji za vrijeme izvršavanja programa (engl. *runtime*) nije moguće izvršavati naredbu po naredbu (engl. *Step by step*) dok je to u *Debug* verzije moguće.

Sljedeći korak optimizacije uključivao je provjeravanje i podešavanje tipa podataka svih varijabli algoritma kako lokalnih tako i globalnih. Postojali su slučajevi gdje su bili korišteni tipovi podataka koji su nepotrebni, jer ta varijabla nikada ne bi iskoristila cijeli raspon vrijednosti koju taj tip nudi. Što se tiče vremena obrade, provedeni postupak nije značajno pridonio smanjenju vremena obrade, ali nije potrebno postavljati vrijednosti na 32 bita, ako se u konačnici isti rezultat može dobiti postavljanjem vrijednosti na 8 ili 16 bita. Ovakvi slučajevi se ne događaju jednom ili dva puta u obradi jednog okvira, događaju se znatno više puta jer je potrebno iterirati kroz sve gotovo sve elemente slike jednog video okvira kako bi se taj okvir u potpunosti obradio. Promjenom tipa podataka određenih varijabli smanjeni su zahtjevi na memorijski prostor. Iako prednosti koje su dobivene ovim postupkom nisu od velike važnosti, one su ipak mjerljive i pokazuju dobru programersku praksu. Vrhunsko poznavanje sustava te briga o detaljima ključni su preduvjeti za maksimalno iskorištavanje resursa i pogodnosti koje taj sustav nudi.

Zadnji korak optimizacije uključuje upotrebu predefiniranih vrijednosti te upotrebu pokazivača u svrhu iteracija kroz matrice prilikom dohvaćanja i spremanja podataka. Korištenje predefiniranih vrijednosti umjesto brojki u kodu je dobra programerska praksa, kako sa strane izvršavanja koda, tako i sa strane dokumentacije i nasljeđivanja koda. Na slici 3.4 može se vidjeti primjer predefiniranih vrijednosti.

```
#define WIDTH_INDEX                 0
#define HEIGHT_INDEX                1
#define FRAME_DATA_INDEX            2
#define FRAME_COUNTER_INDEX         3
#define SAVE_ARTIFACTS_INDEX        4
#define ARTIFACTS_TYPE_INDEX        5
#define ARTIFACTS_COUNTER_INDEX     6
#define ARTIFACTS_ANALYSIS_INDEX    7
#define ARTIFACTS_LOCATION_INDEX    8
#define COLOR_SPACE_INDEX           9
#define PREVIOUS_FRAMES_KEPT_INDEX  10
#define START_PROCESSING_INDEX      11
#define RESULT_INDEX                12
#define BLOCKING_THRESHOLD_INDEX    13
#define NATURAL_EDGE_THRESHOLD_INDEX 14
#define BLOCK_SIZE_INDEX            15
#define MIN_BLOCK_SIZE_INDEX        16
#define MAX_BLOCK_SIZE_INDEX        17
```

Sl. 3.4. *Primjer predefiniranih vrijednosti u programskom jeziku C.*

Na početku izrade diplomskog rada proučeni su trenutačni televizijski i filmski standardi i njihovi trendovi. Da bi algoritam mogao biti koristan u praksi, potrebno je u stvarnom vremenu (engl. r*ealtime*) moći obraditi sadržaj pune visoke rezolucije (engl. *Full High Definition*) koji u sekundi prikaže 50 okvira. Sadržaji te rezolucije sastoje se od 1080 horizontalnih linija, a pritom se svaka linija sastoji od 1920 elemenata slike. Kako bi se zadovoljili ovi kriteriji, vrijeme obrade ne smije prelaziti 20 ms. Iz prethodno navedenih podataka dolazi se do zaključka da bi algoritam u sekundi trebao u potpunosti obraditi 103 680 000 elemenata slike.

Da bi se takvi rezultati postigli, prilikom razvoja i dizajna algoritma moralo se voditi računa o tome da je algoritam moguće paralelizirati. Ciljana arhitektura uređaja na kojem bi se algoritam izvršavao nije u potpunosti definirana, ali se zasigurno radi o uređaju koji posjeduje procesor s više jezgara. U tablici 3.1. napisane su komponente računala na kojemu su se vršila testiranja vezana za vrijeme obrade video okvira ciljane rezolucije.

Tab. 3.1. *Konfiguracija računala korištenog za mjerenje vremena obrade novim algoritmom.*

Procesor	Intel(R) Core(TM) i7-4790 CPU @3.60GHz, 4 jezgre, 8 logičkih procesora
Matična ploča	MSI H81M-E33 (MS-7817)
RAM memorija	Kingston 16.0 GB DDR3, 1600Mhz
Grafička kartica	Intel(R) HD Graphics 4600
Tvrdi disk	Toshiba DT01ACA100
Operacijski sustav	Windows 8.1 Enterprise x64

Mjerenja su izvršena nad 72 176 okvira ciljane rezolucije od 1920x1080 elemenata slike. Prosječno vrijeme obrade jednog okvira iznosilo je 12.665 ms. Dobiveni rezultat implicira da je algoritam, na računalu korištenom za testiranje, u mogućnosti obraditi 78.959 video okvira ciljane rezolucije u jednoj sekundi. Ugrubo rečeno, algoritam može obraditi između 163 000 000 i 164 000 000 elemenata slike u sekundi, na računalu korištenom za testiranje. Ovi rezultati su znatno bolji od onih koji su definirani na početku izrade diplomskog rada i u potpunosti zadovoljavaju postavljene zahtjeve.

Zbog velike količine obrađenih podataka, na slici 3.5. vidljiv je manji dio rezultata mjerenja vremena obrade video okvira ciljane rezolucije.

4287	12.678	12.8799	13.0133	12.6837	12.8944	13.4923	12.1412	11.9683	12.1981	12.1614	12.3429	12.0422	12.2175	12.1984
4288	12.5517	12.5386	13.0005	13.4784	12.6467	12.9203	11.9586	12.292	11.9885	12.1156	12.3034	12.1535	12.0442	12.0872
4289	12.7937	12.7249	12.8284	12.8833	13.0748	13.0301	12.4979	12.2772	12.1279	12.2482	12.1279	12.0974	12.1606	12.1429
4290	12.852	13.1058	12.8876	13.0742	13.1294	13.0967	12.0121	12.2032	11.9654	11.9805	12.1813	12.2721	11.9441	12.1705
4291	12.7078	12.4857	12.4709	12.7596	12.3455	12.5978	12.4769	12.0806	12.1114	12.1401	12.1924	12.1973	12.1452	12.3423
4292	14.0914	12.7818	12.8495	13.2915	13.048	13.1766	12.1455	11.9114	12.924	12.1796	12.1372	12.2129	12.0132	11.997
4293	13.4164	13.1285	13.3882	13.4568	13.3379	13.6422	12.284	12.3312	12.1401	12.2453	12.1728	11.995	12.0977	12.0792
4294	13.481	13.3055	13.3914	13.0947	13.1749	13.605	12.7861	12.3446	12.0405	11.9387	12.1648	12.1489	11.9299	11.972
4295	13.2747	12.8998	13.025	12.9311	13.225	13.2062	12.5892	12.0226	12.104	12.5787	12.119	12.0801	12.2516	12.0582
4296	12.7468	12.0664	12.0511	11.8909	11.7049	11.6921	11.9487	12.1694	12.0866	12.3241	12.1003	12.2217	11.9987	12.0587
4297	11.3106	11.2444	11.2276	11.1764	10.9784	11.1531	12.5144	12.5326	12.1847	12.1404	12.1478	12.0798	12.0806	11.9825
4298	10.781	10.7298	11.1021	10.6701	10.8575	10.5802	12.0761	12.1472	12.117	12.0863	12.1506	12.2396	12.0186	12.2237
4299	10.6795	10.5816	10.7139	10.8228	10.7509	10.8715	12.6032	12.0297	12.1506	12.2587	12.2914	12.061	12.0701	12.1634
4300	10.4513	10.5572	10.5694	10.5142	10.4849	10.5537	12.1114	12.342	12.0408	12.1503	12.3176	12.0749	12.0951	12.5192
4301	10.5287	10.9101	10.7884	11.0117	10.4496	11.237	12.7596	12.4717	12.1663	12.2203	12.1921	12.0067	12.329	12.1466
4302	10.5765	10.5737	10.4926	10.5207	10.5045	10.4633	12.102	12.3062	11.9262	11.9936	12.1936	12.2456	12.0038	12.2169
4303	10.5796	10.4988	10.7278	10.5779	10.9346	10.8837	12.131	12.3617	12.1449	12.129	12.2567	11.9671	12.554	12.3876
4304	10.521	10.4681	10.523	10.6052	10.4314	10.5492	11.9285	12.3127	12.158	12.0724	12.1071	12.2192	12.0428	12.007
4305	10.6055	10.6453	10.7961	10.5773	10.7537	10.9596	12.418	12.164	11.9242	12.1293	12.1099	12.0889	12.1901	12.1711
4306	10.5108	10.4889	11.1155	10.566	10.601	10.494	12.296	11.9723	12.1887	11.9014	12.1207	12.2829	12.1136	12.2089
4307	10.6442	10.8496	10.7958	10.8715	10.5927	11.072	12.4032	11.966	12.1552	12.2581	12.1281	12.1256	12.2206	12.1589
4308	10.6783	10.591	11.9691	10.473	10.7153	10.6479	12.2792	11.9415	12.0843	12.199	12.1802	12.2732	12.0977	12.0587
4309	10.7176	11.1986	10.9719	10.5973	10.6177	10.7264	12.3611	12.4658	12.4006	12.5591	12.1188	11.9882	12.1688	12.1626
4310	10.6846	10.4786	10.5734	11.074	10.5512	10.535	12.5181	12.4578	12.0633	12.1549	12.1975	12.1663	12.0164	11.9819
4311	11.1095	10.5688	10.8302	10.4795	10.8439	10.6852	12.1839	12.8657	12.5474	12.1068	12.2621	12.1023	12.1296	12.1074
4312	10.5597	10.5444	10.6365	10.5324	10.5845	10.4351	12.1222	12.4555	12.2254	11.9563	12.6131	12.1779	12.2078	11.9976
4313	10.7693	10.8553	10.9884	11.0094	10.6943	10.6857	12.1222	12.4382	12.1671	12.1984	12.1094	11.9057	12.1919	12.0613
4314	10.6214	10.9503	10.7511	10.7179	10.5239	10.6146	11.8875	12.3244	12.0428	11.9711	12.3893	12.1603	12.042	12.0033
4315	10.8547	10.581	10.7398	10.6194	10.6544	10.7352	12.1253	12.0451	12.1623	12.098	12.0548	11.9657	12.1864	12.1586
4316	10.7017	10.55	10.5768	10.6163	10.4818	10.5213	11.9205	12.1643	12.0118	12.104	12.0934	12.1731	12.0858	12.2888
4317	10.9585	10.6064	10.8169	10.9602	10.8385	10.7457	12.3443	12.3199	12.2166	12.2806	11.9572	12.2015	12.193	12.119
4318	10.6044	10.5628	10.7338	10.632	10.9047	10.504	11.964	12.3207	12.1224	12.1856	12.1259	12.1981	12.0431	12.1765
4319	10.5825	10.6331	10.9107	10.4932	10.6169	10.7762	12.4427	12.0892	12.1938	12.0459	12.0869	11.8969	12.3002	12.0707
4320	10.7136	10.5793	10.7193	10.5867	10.8388	10.7031	12.003	12.0579	12.0462	12.0004	12.1526	12.2138	12.1859	11.9617
4321	10.8245	10.5552	11.0089	10.9127	11.2677	10.7571	12.3796	12.4089	12.2735	12.1754	12.197	11.9819	12.1458	12.1589
4322	10.5739	10.5296	10.618	10.8937	10.6277	10.4983	12.3583	11.9916	12.2795	12.0118	12.2368	12.3827	12.1569	12.288
4323	10.6072	10.4337	10.8041	10.5993	10.517	10.7355	12.1896	12.4069	12.1	12.1921	12.1367	12.1404	12.1933	12.1899
4324	10.723	10.5745	11.1007	10.6598	10.6866	10.5973	12.191	11.976	12.0115	12.0513	12.1458	12.125	12.042	12.0829

Sl. 3.5. *Dio rezultata mjerenja vremena obrade video okvira rezolucije 1920x1080 elemenata slike.*

Za izračun vremena obrade okvira korištene su *QueryPerformanceFrequency* i *QueryPerformanceCounter* funkcije čija se dokumentacija nalazi na [22] i [23].

3.3. Skaliranje rezultata

Algoritam bez konkretne metrike koja evaluira obrađeni okvir nije moguće koristiti u praksi. Budući da povratna vrijednost algoritma ovisi o veličini rešetke koja predstavlja teorijski maksimum pojavljivanja artefakta stvaranja blokova za taj okvir, a teorijski maksimum je gotovo nemoguće dosegnuti, provedeno je skaliranje rezultata kako bi rezultat algoritma vjerodostojnije predstavljao dio okvira zahvaćen artefaktom stvaranja blokova.

Sadržaji čiji su rezultati prije skaliranja iznosili barem 40%, sadržavali su neprimjereno visok broj artefakata stvaranja blokova, a vrijednost od 40% to ne sugerira zbog upotrebe teorijskog maksimuma prilikom izračuna vrijednosti algoritma.

Iz baza testnih signala, koje su prezentirane u sljedećem poglavlju, izdvojeno je 180 okvira. 100 okvira pripada video sadržaju kodiranom MPEG-2 normom, a preostalih 80 su okviri kodirani H.264 normom. Izdvojeni okviri su dani grupi od 16 ljudi na procjenu, s naputkom da procjene koliki dio okvira sadrži artefakte stvaranja blokova. Testiranje je provedeno u dva uzastopna dana kako zamor ocjenjivača ne bi utjecao na rezultat. Iako je u oba dana bilo potrebno ocijeniti 90

okvira, oni nisu razdvojeni po normi, već su u oba dana ocjenjivači ocjenjivali okvire kodirane i MPEG-2 i H.264 normom.

Proces izračuna funkcija za prilagodbu (engl. *fitting function*) započeo je definiranjem tri skupa podataka. Prvi skup podataka činile su vrijednosti ocjenjivača za okvire kodirane MPEG-2 normom, drugi skup su sačinjavale vrijednosti ocjenjivača za okvire kodirane H.264 normom. Treći skup je načinjen unijom prvog i drugog skupa podataka. Za svaki od skupova nasumičnim odabirom su definirana dva podskupa, podskup za treniranje i podskup za testiranje. Podskupovi sadržavaju jednak broj elemenata.

Uloga podskupa za treniranje je pronalazak matematičkog modela koji najbolje prilagođava rezultate algoritma rezultatima gledatelja. Matematički model dobiven na podskupu za treniranje primjenjuje se na podskup za testiranje. Budući da podaci u testnom podskupu podataka sadrže vrijednosti koje se žele predvidjeti, čija je stvarna vrijednost poznata, moguće je odrediti ispravnost matematičkog modela. Ovaj postupak je iterativan. Za svaki od tri definirana skupa podataka pronađen je odgovarajući matematički model.

4. VREDNOVANJE REZULTATA NOVOG ALGORITMA ZA DETEKCIJU ARTEFAKTA STVARANJA BLOKOVA

U ovom poglavlju prezentirane su informacije o bazama testnih video signala, rezultatima testiranja te mogućnosti buduće nadogradnje. Prikazani su primjeri detekcije za MPEG-2 i H.264 norme.

4.1. Baze testnih video signala

Kako bi bilo moguće vrednovati ispravan rad algoritma, potrebni su komprimirani digitalni video signali. Budući da je baza testnih signala ključni dio razvoja algoritma za obradu videa, nužno je na raspolaganju imati kvalitetnu bazu. Pod izrazom kvalitetna baza podrazumijeva se da se u bazi nalaze i sporo i brzo promjenjivi video sadržaji, s različitim stupnjevima kompresije.

Algoritam treba zadovoljavati uvjet detekcije postupnog povećanja artefakta stvaranja blokova, što znači da povratna vrijednost algoritma mora biti veća što je veća razina kompresije. Razinu kompresije moguće je kvalitativno izmjeriti uspoređivanjem veličina video datoteka, ali to nije u potpunosti pouzdan način. Kako bi se dobila kvantitivna vrijednost kompresije video datoteke korišten je program MSU Video Quality Measurement Tool kojim su izmjerene određene metrike između nekomprimiranog i svakog od komprimiranih videa, čime se potvrđuje stupanj kompresije videa. Prilikom testiranja rada algoritma korištene su dvije baze testnih signala.

4.1.1. Live Video baza signala

Ova baza video signala preuzeta je s internetske stranice navedene na [24] i sastoji se od 90 video datoteka relevantnih za ovaj diplomski rad, koje su podijeljene na 10 različitih sadržaja. Svaki od sadržaja prezentiran je s 9 relevantnih video datoteka. Za svaki sadržaj postoji originalna YUV video datoteka koja se u praksi smatra „savršenom kvalitetom“, 4 YUV video datoteke različitog stupnja kompresije označene brojevima od 9 do 12 koje su kodirane H.264 normom i 4 YUV video datoteke različitog stupnja kompresije označene brojevima od 13 do 16 koje su kodirane MPEG-2 normom. Stupanj kompresije se povećava kako se povećava brojčana oznaka video datoteke. Tako oznaka 12 označava najkomprimiraniju video datoteku kodiranu H.264 normom dok oznaka 13 predstavlja video datoteku s najnižim stupnjem kompresije kodiranu MPEG-2 normom. Video datoteke svih sadržaja koje su kodirane MPEG-2 normom sačinjavaju Live Video MPEG-2 bazu signala, a video datoteke svih sadržaja koje su kodirane H.264 normom sačinjavaju Live Video H.264 bazu signala.

Kako bi ih algoritam mogao obraditi, video datoteke su prekodirane iz .yuv spremnika u video datoteke s .mp4 spremnikom koristeći alat *FFmpeg*.

Budući da sve video datoteke istog sadržaja imaju iste vrijednosti u tablici 4.1., radi efikasnosti prikaza, predstavljene su imenom sadržaja bez brojčanih oznaka koje specificiraju normu i stupanj kompresije. Rezolucija svih sadržaja je 768x432.

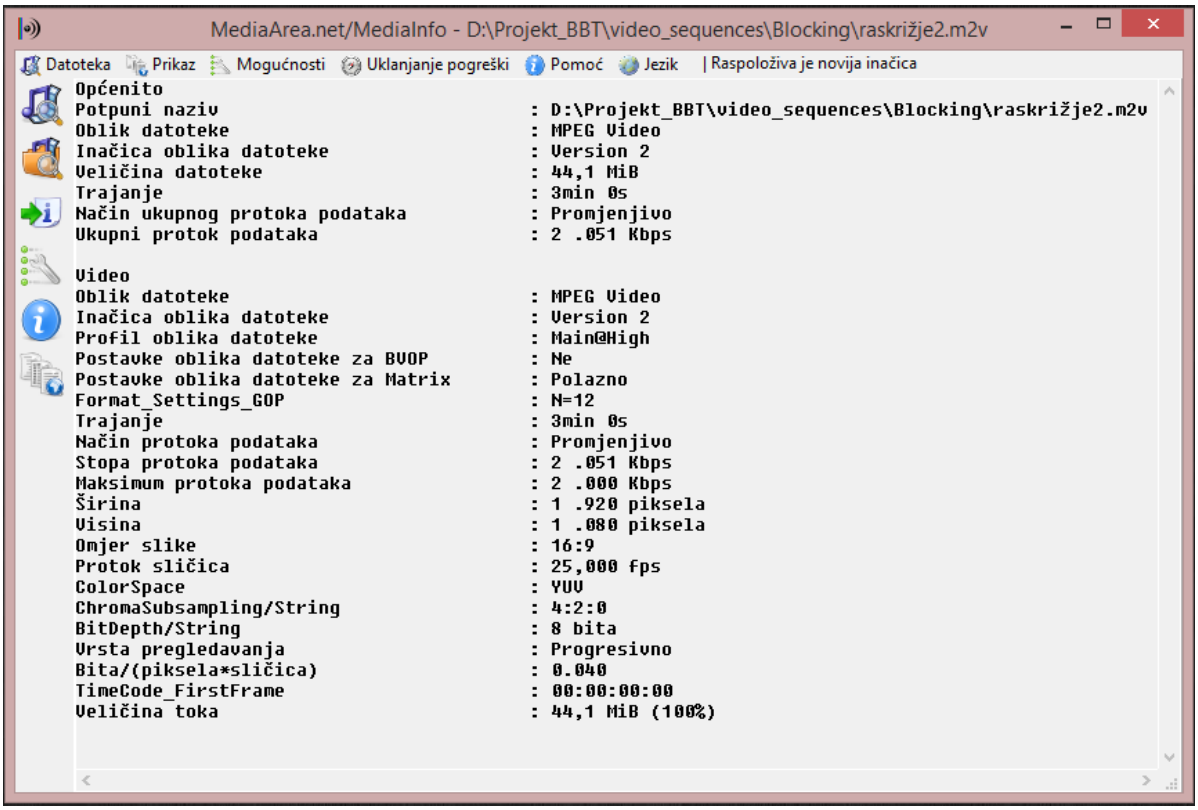
Tab. 4.1. *Opći podaci o video sadržajima Live Video baze signala.*

Ime sadržaja	Broj okvira	Broj okvira u sekundi (FPS)	Trajanje sadržaja (s)
bs	217	25	8.68
mc	500	50	10.00
pa	250	25	10.00
pr	500	50	10.00
rb	250	25	10.00
rh	250	25	10.00
sf	250	25	10.00
sh	500	50	10.00
st	250	25	10.00
tr	250	25	10.00

4.1.2. RT-RK baza signala

FFmpeg je besplatan alat koji sadržava biblioteke i programe za manipuliranje multimedijskim podacima [25]. Najbitniji program je ffmpeg - program komandne linije (engl. *command line tool*) koji omogućava kodiranje multimedijskih datoteka. Trenutačna stabilna verzija 3.1.2 sadrži više od 400 kodeka za zvuk, video i prijevode (engl. *subtitle*). Osim *ffmpeg* programa, alat sačinjavaju programi *ffplay*, *ffprobe* i *ffserver*. Sastavni dio *ffmpeg* programa je i libx264 koder H.264 norme.

Ova baza video signala nastala je u suradnji s Institutom RT-RK Osijek d.o.o. Baza je kreirana korištenjem *ffmpeg* programa i sastoji se od 10 relevantnih video datoteka koje su podijeljene u 3 različita sadržaja. Svaki sadržaj se sastoji od originalne datoteke i tri ili četiri komprimirane datoteke. Za svaku video datoteku postavljena je ciljana brzina prijenosa i sadržaj je kodiran MPEG-2 normom. Alatom *MediaInfo* [26] moguće je iščitati podatke o glazbenim, slikovnim i video datotekama. Na slici 4.1. prikazan je ispis *MediaInfo* alata za jednu video datoteku iz ove baze signala.



Sl. 4.1. *MediaInfo prikaz testne video datoteke.*

U većini slučajeva, ciljana brzina prijenosa je varijabilna i odstupa za minimalnu vrijednost od postavljene. Kao referentna brzina prijenosa koristit će se „Ukupni protok podataka“ iz zaglavlja sa slike 4.1.. U tablici 4.2. prikazane su relevantne video datoteke i opći podaci o njima. Sve datoteke izmjenjuju 25 okvira u sekundi i imaju rezoluciju od 1920x1080 elemenata slike.

Tab. 4.2. *Opći podaci o video datotekama RT-RK baze signala.*

Ime video datoteke	Veličina datoteke (KB)	Ciljani protok podataka (Kb/s)	Ukupni protok podataka (Kb/s)	Broj okvira	Trajanje sadržaja (s)
oblaci1	65 996	3 000	3 004	4500	180.00
oblaci2	32 991	1 500	1 501	4500	180.00
oblaci3	18 360	750	836	4500	180.00
raskrižje1	66 114	3 000	3 006	4505	180.20
raskrižje2	45 125	2 000	2 051	4505	180.20
raskrižje3	40 272	1 000	1 831	4505	180.20
starcraft1	330 896	15 000	15 000	4522	188.88
starcraft2	224 302	10 000	10 200	4522	188.88
starcraft3	134 198	3 000	6 078	4522	188.88
starcraft4	129 951	2 000	5 885	4522	188.88

4.2. Rezultati testiranja

Kako bi se dokazali ispravnost i dosljednost algoritma, potrebno je provesti određena testiranja na razini video datoteke umjesto okvira videa. Budući da su u tablici 4.2. predstavljene video datoteke istog sadržaja ali različitog stupnja kompresije, algoritam tu činjenicu mora potvrditi svojim povratnim vrijednostima. Na razini sadržaja, video datoteke manje veličine imaju viši stupanj kompresije, a budući da se artefakt stvaranja blokova pojavljuje usred višeg stupnja kompresije, na razini video datoteke povratna vrijednost algoritma mora biti veća za komprimiranije video datoteke. Institut RT-RK Osijek d.o.o ustupio je svoj komercijalni RT-RK-BM (engl. *RT-RK-Blocking Measurement*) algoritam za detekciju artefakta stvaranja blokova kako bi se mogli usporediti rezultati. Algoritam realiziran u ovom diplomskom radu nazvan je BDA (engl. *Blocking Detection Algorithm*). Budući da RT-RK-BM algoritam procjenjuje artefakte stvaranja blokova u rasponu od 0 do 10 000, a BDA od 0.00 do 100.00, u tablicama 4.3., 4.4. i 4.5., koje prikazuju rezultate oba algoritma za svaku od baza podataka dodan je stupac u kojem se vrijednost RT-RK-BM algoritma skalira kako bi raspon vrijednosti algoritma odgovarao rasponu BDA algoritma. Skalirana vrijednost RT-RK-BM algoritma dobivena je tako što se originalna vrijednost tog algoritma podijelila sa 100.

4.2.1. Rezultati za Live Video MPEG-2 bazu signala

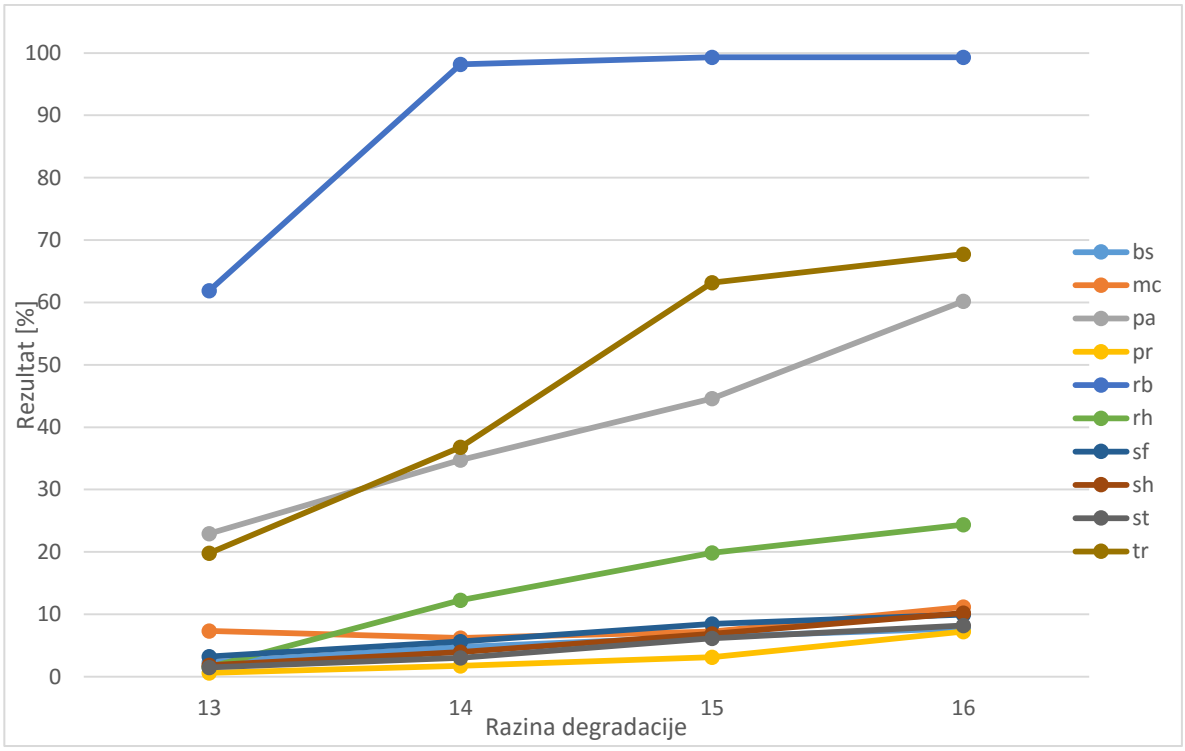
U ovom potpoglavlju prezentirani su rezultati dobiveni obradom video sadržaja Live Video baze signala koji su kodirani MPEG-2 normom. Obrađeno je 40 video datoteka. Dosljednost RT-RK-BM algoritma za ovu bazu iznosi 95%, budući da od svih obrađenih datoteka samo dvije ne zadovoljavaju uvjet postupnog povećanja artefakta stvaranja blokova. Konzistentnost BDA algoritma iznosi 100%. Nedosljedni rezultati su u sljedećim tablicama prikazani crvenom bojom.

Tab. 4.3. *Rezultati BDA i RT-RK-BM algoritama za Live Video MPEG-2 bazu signala.*

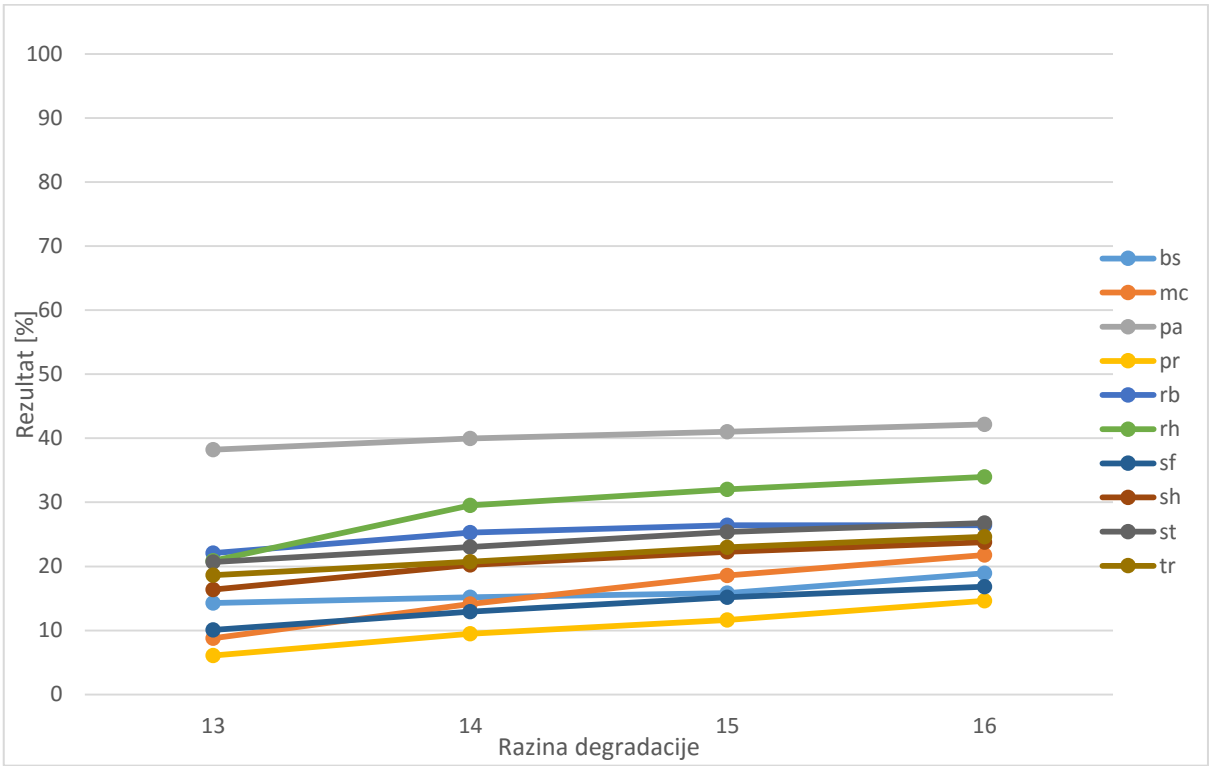
Ime sadržaja	Rezultat BDA algoritma [%]	Rezultat RT-RK-BM algoritma	Skalirana vrijednost RT-RK-BM algoritma [%]
bs_13	14.27	231.21659	2.31
bs_14	15.16	476.709677	4.77
bs_15	15.83	641.230415	6.41
bs_16	18.95	774.207373	7.74
mc_13	8.77	731.466	7.31
mc_14	14.14	620.494	6.20
mc_15	18.59	721.8	7.22
mc_16	21.79	1119.678	11.20

Ime sadržaja	Rezultat BDA algoritma [%]	Rezultat RT-RK-BM algoritma	Skalirana vrijednost RT-RK-BM algoritma [%]
pa_13	38.22	2294.568	22.95
pa_14	39.99	3474.924	34.75
pa_15	41.02	4459.804	44.60
pa_16	42.17	6018.524	60.19
pr_13	6.09	63.4	0.63
pr_14	9.46	174.652	1.75
pr_15	11.63	313.246	3.13
pr_16	14.61	723.212	7.23
rb_13	22.07	6189.532	61.90
rb_14	25.26	9819.452	98.19
rb_15	26.41	9933.156	99.33
rb_16	26.41	9933.156	99.33
rh_13	20.88	154.744	1.55
rh_14	29.54	1225.304	12.25
rh_15	32.04	1983.964	19.84
rh_16	33.96	2437.38	24.37
sf_13	10.06	325.068	3.25
sf_14	12.94	563.212	5.63
sf_15	15.16	845.048	8.45
sf_16	16.81	995.328	9.95
sh_13	16.38	182.864	1.83
sh_14	20.22	395.282	3.95
sh_15	22.29	687.47	6.87
sh_16	23.76	1019.586	10.20
st_13	20.66	146.772	1.47
st_14	23.04	304.788	3.05
st_15	25.38	617.532	6.18
st_16	26.79	821.792	8.22
tr_13	18.65	1982.088	19.82
tr_14	20.71	3679.444	36.79
tr_15	22.97	6316.216	63.16
tr_16	24.64	6776.504	67.77

Slika 4.2. prikazuje rezultate RT-RK-BM algoritma za Live Video MPEG-2 bazu signala, a slika 4.3. prikazuje rezultate BDA algoritma za istu bazu signala.



Sl. 4.2. *Rezultati RT-RK-BM algoritma za Live Video MPEG-2 bazu signala.*



Sl. 4.3. *Rezultati BDA algoritma za Live Video MPEG-2 bazu signala.*

Primjer detekcije dijelova okvira zahvaćenih artefaktom stvaranja blokova za jedan okvir odabrane MPEG-2 sekvence dan je na slici 4.4.

Sl. 4.4. *Rezultati BDA algoritma na okviru prikazanom na slici 2.4.*

4.2.2. Rezultati za Live Video H.264 bazu signala

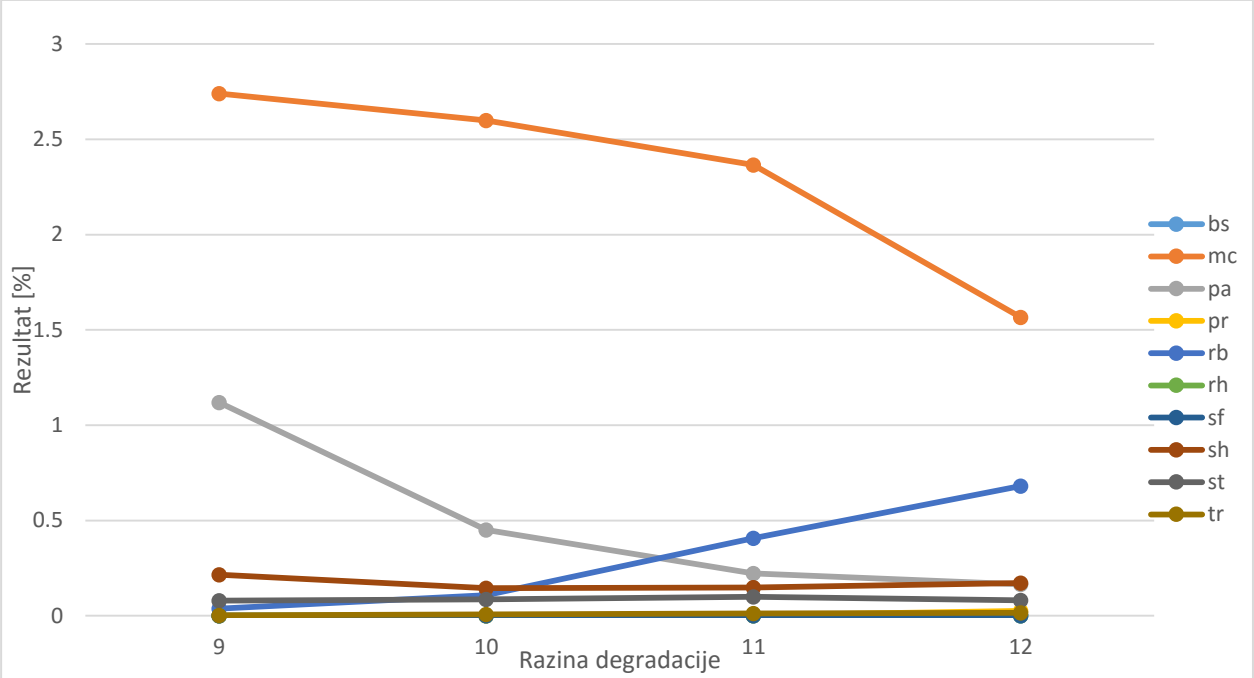
U ovom potpoglavlju prezentirani su rezultati dobiveni obradom video sadržaja Live Video baze signala koji su kodirani H.264 normom. Obrađeno je 40 video datoteka. Dosljednost RT-RK-BM algoritma za ovu bazu podataka iznosi 25%, dok dosljednost BDA algoritma iznosi 90%.

Tab. 4.4. *Rezultati BDA i RT-RK-BM algoritama za Live Video H.264 bazu signala.*

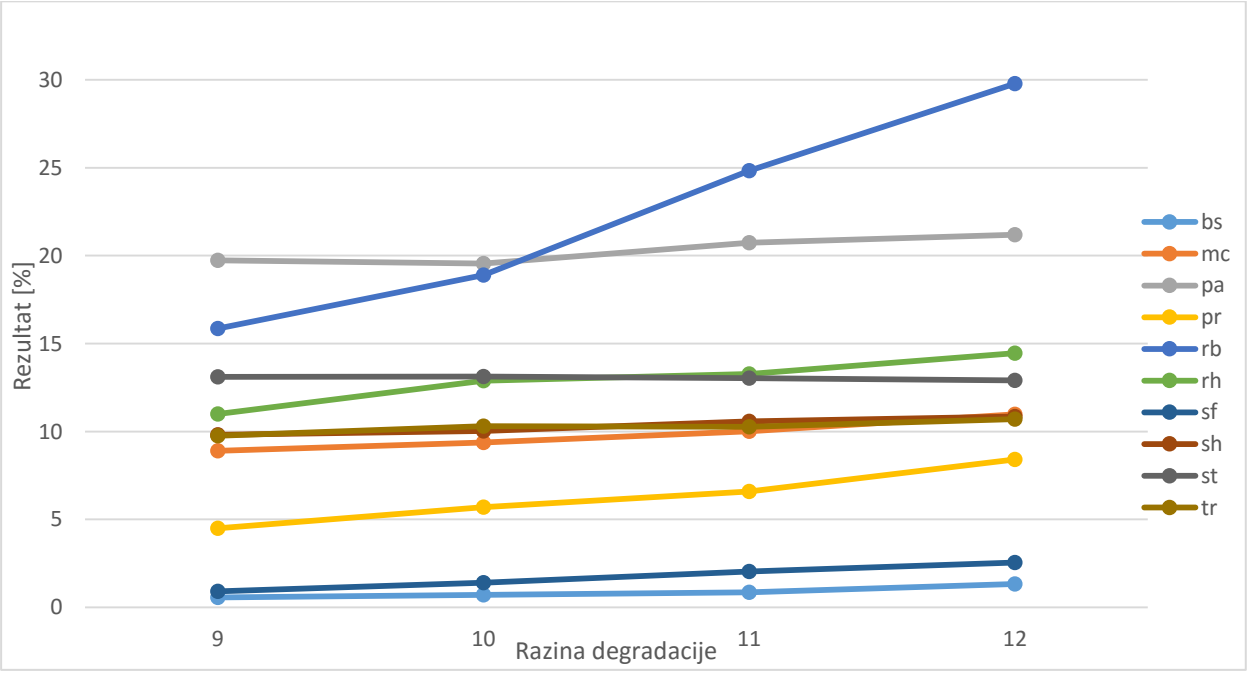
Ime sadržaja	Rezultat BDA algoritma [%]	Rezultat RT-RK-BM algoritma	Skalirana vrijednost RT-RK-BM algoritma [%]
bs_9	0.56	0.004608	0.00
bs_10	0.71	0.004608	0.00
bs_11	0.85	0.004608	0.00
bs_12	1.32	0.009217	0.00
mc_9	8.91	273.994	2.74
mc_10	9.37	259.882	2.60
mc_11	10.02	236.538	2.37
mc_12	10.97	156.672	1.57

Ime sadržaja	Rezultat BDA algoritma [%]	Rezultat RT-RK-BM algoritma	Skalirana vrijednost RT-RK-BM algoritma [%]
pa_9	19.73	115.952	1.12
pa_10	19.55	45.116	0.45
pa_11	20.74	22.276	0.22
pa_12	21.19	16.54	0.17
pr_9	4.49	0	0.00
pr_10	5.69	0	0.00
pr_11	6.59	0.308	0.00
pr_12	8.41	2.528	0.03
rb_9	15.86	3.828	0.04
rb_10	18.89	10.868	0.11
rb_11	24.83	40.748	0.41
rb_12	29.79	68.072	0.68
rh_9	10.99	0.024	0.00
rh_10	12.89	0.056	0.00
rh_11	13.28	0.024	0.00
rh_12	14.46	0.092	0.00
sf_9	0.91	0.12	0.00
sf_10	1.41	0.12	0.00
sf_11	2.03	0.12	0.00
sf_12	2.54	0.12	0.00
sh_9	9.82	21.564	0.22
sh_10	10.04	14.524	0.15
sh_11	10.58	14.84	0.15
sh_12	10.85	17.228	0.17
st_9	13.10	7.948	0.08
st_10	13.12	8.624	0.09
st_11	13.03	10	0.10
st_12	12.90	8.072	0.08
tr_9	9.76	0.24	0.00
tr_10	10.30	0.676	0.01
tr_11	10.27	1.284	0.01
tr_12	10.70	1.596	0.02

Na slici 4.5. prikazani su rezultati RT-RK-BM algoritma za Live Video H.264 bazu signala. Slika 4.6. prikazuje rezultate BDA algoritma za istu bazu signala.

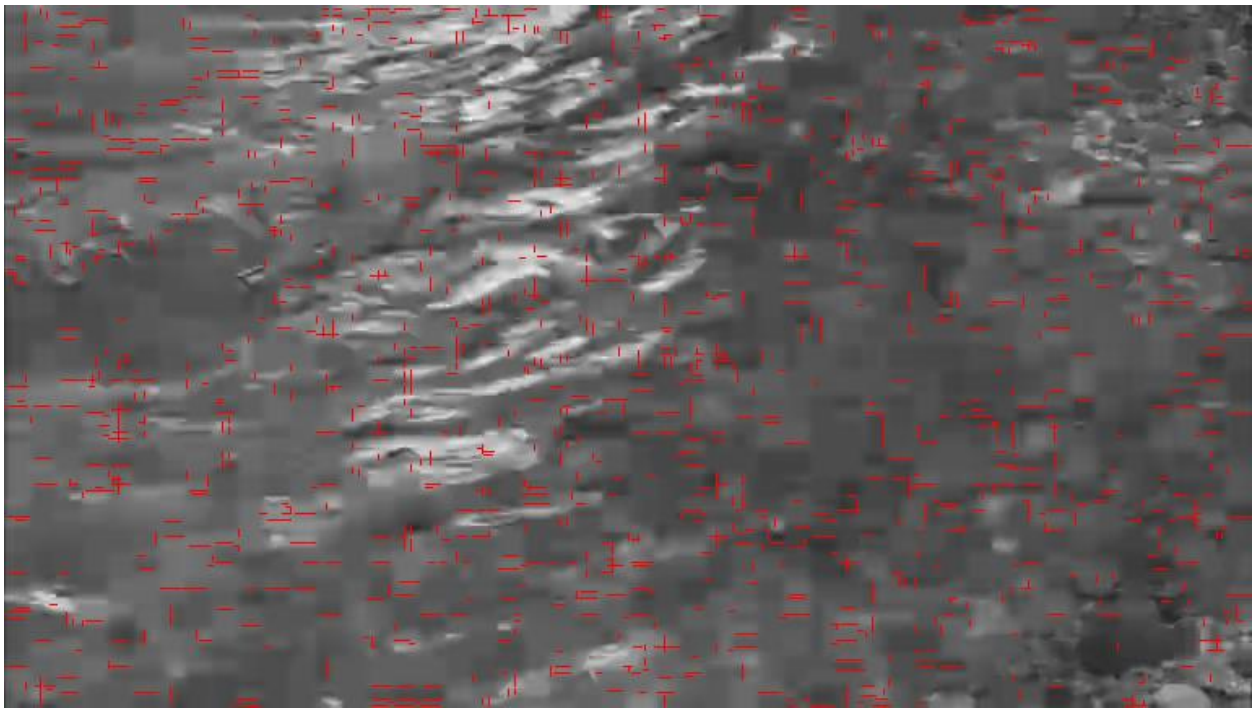


Sl. 4.5. *Rezultati RT-RK-BM algoritma za Live Video H.264 bazu signala.*



Sl. 4.6. *Rezultati BDA algoritma za Live Video H.264 bazu signala.*

Primjer detekcije dijelova okvira zahvaćenih artefaktom stvaranja blokova za jedan okvir odabrane H.264 sekvence dan je na slici 4.7.



Sl. 4.7. *Rezultati BDA algoritma na okviru prikazanom na slici 2.5.*

Budući da H.264 norma posjeduje filtar za uklanjanje artefakta stvaranja blokova, povećanjem stupnja kompresije ne stvaraju se vidno izraženi blokovi kao kod MPEG-2 norme, već dolazi do zamućivanja okvira. Ukoliko se radi o vrlo visokom stupnju kompresije, ponekada je moguće prepoznati granice blokova, ali to prvenstveno ovisi o sadržaju okvira.

Iz tablice 4.4 vidljivo je da RT-RK-BM algoritam ne uspijeva detektirati artefakt stvaranja blokova u H.264 sekvencama. Iznosi koje on daje kao krajnji rezultat ni približno ne odgovaraju stvarnim iznosima koji se mogu procijeniti gledanjem istih sekvenci. Uza sve to, algoritam je i poprilično nedosljedan.

Za razliku od njega, BDA algoritam puno bolje procjenjuje koliki je dio okvira zahvaćen artefaktom i rezultati su mu puno bliži stvarnima. Uza sve to, BDA algoritam je i puno dosljedniji od RT-RK-BM algoritma.

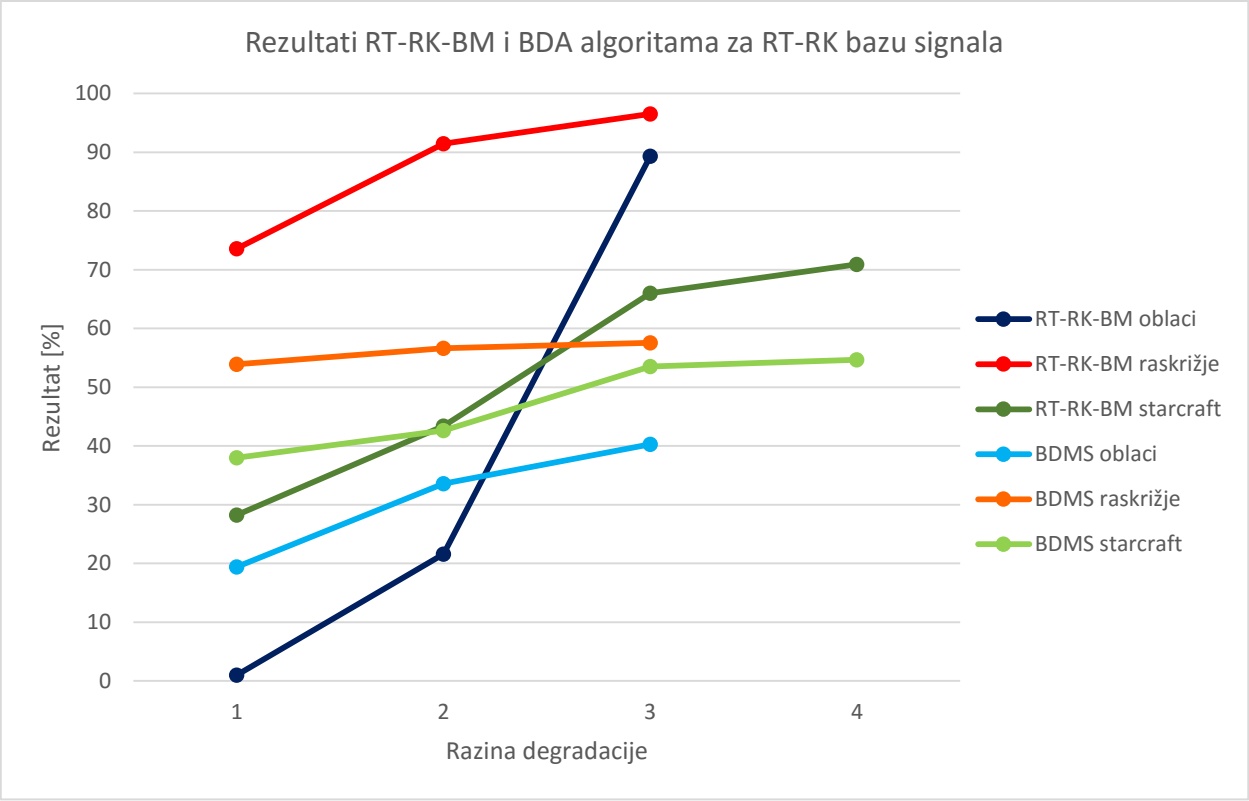
4.2.3. Rezultati za RT-RK bazu signala

U ovom potpoglavlju prezentirani su rezultati dobiveni obradom video sadržaja RT-RK baze signala. obrađeno je 10 video datoteka Sve video datoteke su kodirane u skladu s MPEG-2 normom. Dosljednost oba algoritma za ovu bazu iznosi 100%.

Tab. 4.5. *Rezultati BDA i RT-RK-BM algoritama za RT-RK bazu signala.*

Ime sadržaja	Rezultat BDA algoritma [%]	Rezultat RT-RK-BM algoritma	Skalirana vrijednost RT-RK-BM algoritma [%]
oblaci1	19.43	98.070682	0.98
oblaci2	33.58	2161.713714	21.62
oblaci3	40.30	8934.512336	89.35
raskrižje1	53.90	7359.451377	73.59
raskrižje2	56.64	9144.203597	91.44
raskrižje3	57.56	9652.994449	96.53
starcraft1	37.97	2822.169432	28.22
starcraft2	42.64	4339.818403	43.40
starcraft3	53.52	6600.609157	66.00
starcraft4	54.69	7089.386861	70.89

Na slici 4.8. nalaze se rezultati oba algoritma za RT-RK bazu signala.



Sl. 4.8. *Rezultati RT-RK-BM i BDA algoritama za RT-RK bazu signala.*

Uzimajući u obzir sve obrađene video datoteke, njih 90, dosljednost RT-RK-BM algoritma iznosi 64.44% budući da je pravilo dosljednosti zadovoljeno za 58 video datoteka. Kako je BDA algoritam zadovoljio isto pravilo za 86 video datoteka, njegova dosljednost iznosi 95.56%, što predstavlja bitno poboljšanje u odnosu na RT-RK-BM algoritam.

Ako se usporede izlazne vrijednosti BDA i RT-RK-BM algoritama za MPEG-2 sekvence u tablicama 4.3. i 4.5. može se vidjeti da RT-RK-BM daje puno veće vrijednosti. U nekoliko slučajeva one su jako blizu maksimalno moguće vrijednosti (10000, odnosno 100). Pregledavanjem predmetnih sekvenci ustanovljeno je da RT-RK-BM algoritam u tim slučajevima značajno precjenjuje stvarnu vrijednost dijela okvira zahvaćenog artefaktom stvaranja blokova, dok je BDA algoritam puno bliže toj vrijednosti.

4.3. Buduće nadogradnje

Budući da je ovo prva verzija algoritma jasno je da prostora za napredak ima. Razmatrajući područje primjene algoritma, budući rad može obuhvatiti sljedeća poboljšanja:

- naprednije optimizacije za izvođenje u stvarnom vremenu
- efikasnija detekcija
- detektiranje norme i parametrizacija

Ukoliko bi bila potrebna naprednija optimizacija algoritma, taj posao morao bi se povjeriti osobi koja ima višegodišnje iskustvo programiranja u C programskom jeziku ili asembleru. Budući da postoji video sadržaj ultra visoke kvalitete i televizijska i filmska industrija se kreću u smjeru izrade sadržaja u sve većim rezolucijama, kvalitetnija optimizacija bi mogla omogućiti obradu takvih sadržaja u realnom vremenu. Za obradu sadržaja koji reproducira 50 okvira u sekundi, 4K DCI rezolucije, algoritam u jednoj sekundi treba obraditi 442 368 000 elemenata slike. Uspoređujući s podacima o ciljanoj rezolucijom i brzini izmjene okvira prezentiranim u potpoglavlju 3.2., radi se o 426.67% povećanju količine informacija koju je potrebno obraditi.

Izrada algoritma je kompleksan i iscrpan posao i takav algoritam je vrlo teško napraviti ukoliko se problematika i dostupni alati ne poznaju dovoljno dobro. Problem kod algoritama je što nema garancije da se u jednom trenutku neće pojaviti video datoteka čiji rezultati neće biti ispravni. Navedena situacija će zahtijevati dorade postojećeg algoritma, s time da nova verzija algoritma ne smije utjecati na ispravnost rezultata video datoteka ili okvira koji su ispravno vrednovani postojećom verzijom algoritma. Kompleksniji izračuni bi polučili kvalitetniju detekciju, ali bi

ovisno o stupnju kompleksnosti povećali vrijeme potrebno za obradu okvira i time dodatno ograničili algoritam sa strane obrade u stvarnom vremenu.

Ukoliko bi se napravio programski modul za detektiranje norme ili da se norma prepoznaje upotrebom neke vanjske aplikacije, moguće je podešavati pragove algoritma prema normi. Rezultat toga su kvalitetnija detekcija, konzistentniji rezultati i mogućnost obrade određenih specifičnih slučajeva.

5. ZAKLJUČAK

U diplomskom radu osmišljen je i realiziran algoritam za detekciju i mjerenje artefakta stvaranja blokova u videu. Algoritam funkcionira za video signale komprimirane u skladu s MPEG-2 i H.264 normama za razne rezolucije i brzine izmjene okvira. Testiranja su dokazala da je u implementiranoj verziji algoritma u stvarnom vremenu moguće obraditi oko 1.57 puta više podataka od željene količine podataka koja bi algoritam učinila primjenjivim u praksi. Prva faza razvoja odvijala se u MATLAB okruženju, a konačna verzija algoritma je napisana u programskom jeziku C i ne koristi pomoćne biblioteke za obradu. Prilikom obrade algoritam ima pristup samo komprimiranim video okvirima.

Algoritam može biti modul veće aplikacije koja uključuje skup algoritama za analizu kvalitete i kompresije, mogu ga koristiti telekom operateri u svrhe dijagnostike i popravaka, poput provjere stupnja kompresije sadržaja i usporedbe utjecaja parametara kodera na pojavljivanje artefakta stvaranja blokova.

Ostvareni algoritam moguće je ugraditi kao dio kodera koji bi vršio naknadnu obradu te svoje rezultate prosljeđivao koderu, koji bi na temelju tih informacija prilagođavao svoje parametre.

LITERATURA

[1] Institut RT-RK http://www.rt-rk.com/, pristup ostvaren 20.8.2016.

[2] Understanding the Application: An Overview of the H.264 Standard http://www.springer.com/cda/content/document/cda_downloaddocument/9781461422297-c1.pdf?SGWID=0-0-45-1338306-p174267072, pristup ostvaren 20.7.2016.

[3] RGB, Y, U, V prikaz slike https://upload.wikimedia.org/wikipedia/commons/thumb/2/29/Barn-yuv.png/320px-Barn-yuv.png, pristup ostvaren 1.9.2016.

[4] Poduzorkovanje boja http://calendar.perfplanet.com/wp-content/uploads/2015/12/chroma/chroma-subsampling-basics.jpg, pristup ostvaren 1.9.2016.

[5] Primjena poduzorkovanja boja https://upload.wikimedia.org/wikipedia/commons/0/06/Colorcomp.jpg, pristup ostvaren 1.9.2016.

[6] VCEG http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/vceg.aspx, pristup ostvaren 15.9.2016.

[7] JVT http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/jvt.aspx, pristup ostvaren 15.9.2016.

[8] MPEG LA http://www.mpegla.com/main/Pages/About.aspx, pristup ostvaren 15.9.2016.

[9] MPEG LA – H.264 http://www.mpegla.com/main/programs/AVC/Documents/avcweb.pdf, pristup ostvaren 15.9.2016.

[10] MPEG LA – MPEG-2 http://www.mpegla.com/main/programs/M2/Documents/m2web.pdf, pristup ostvaren 15.9.2016.

[11] Alliance for Open Media http://aomedia.org/about-us/, pristup ostvaren 15.9.2016.

[12] WebM Project http://www.webmproject.org/about/faq/, pristup ostvaren 10.9.2016.

[13] Cisco Thor http://blogs.cisco.com/collaboration/world-meet-thor-a-project-to-hammer-out-a-royalty-free-video-codec , pristup ostvaren 10.7.2016.

[14] RT-RK BBT http://bbt.rt-rk.com/audiovideo-analysis/, pristup ostvaren 5.9.2016.

[15] G. Qiu et al., Advances in Multimedia Information Processing - - PCM 2010, Part I, Springer, Šangaj Kina, 2010.

[16] H. Liu, I. Heynderickx, A no-reference perceptual Blockiness metric, Acoustics, Speech and Signal Processing, 865 – 868, Las Vegas Nevada, 2008.

[17] Z. Wang, A.C. Bovik, B.L. Evan, Blind measurement of blocking artifacts in images, Proceedings 2000 International Conference on Image Processing, 981 – 984 Vol 3, Vancouver Kanada, 2000.

[18] MSU Blocking
http://www.compression.ru/video/quality_measure/info_en.html#blockingmeasure, pristup ostvaren 5.9.2016.

[19] Tipovi algoritama ovisno o dostupnim podacima
http://live.ece.utexas.edu/publications/2012/tip%20brisque.pdf, pristup ostvaren 6.9.2016.

[20] MATLAB http://www.mathworks.com/products/matlab/, pristup ostvaren 30.6.2016.

[21] Visual Studio https://www.visualstudio.com/, pristup ostvaren 10.9.2016.

[22] QueryPerformanceFrequency
https://msdn.microsoft.com/en-us/library/windows/desktop/ms644905(v=vs.85).aspx, pristup ostvaren 25.8.2016.

[23] QueryPerformanceCounter
https://msdn.microsoft.com/en-us/library/windows/desktop/ms644904(v=vs.85).aspx, pristup ostvaren 25.8.2016.

[24] Live Video Quality Assessment Database
http://live.ece.utexas.edu/research/quality/live_video.html, pristup ostvaren 20.6.2016.

[25] FFmpeg https://ffmpeg.org/, pristup ostvaren 15.8.2016.

[26] MediaInfo https://mediaarea.net/hr/MediaInfo, pristup ostvaren 15.8.2016.

SAŽETAK

U diplomskom radu osmišljen je algoritam za detekciju i mjerenje artefakta stvaranja blokova u videu. Algoritam, napisan u programskoj jeziku C, uspješno radi za MPEG-2 i H.264 kodirane video signale za razne rezolucije i brzine izmjene okvira u sekundi. Algoritam ne zahtjeva nikakve podatke o nekomprimiranom video signalu, odnosno radi obradu bez reference i ne koristi nikakve vanjske biblioteke za obradu okvira. Testiranjima je dokazano da je u stvarnom vremenu moguće obraditi sadržaj sa 1920x1080 elemenata slike gdje se u sekundi izmjeni do 78 okvira. Dosljednost algoritma iznosi vrlo visokih 95.56%. Algoritam je u potpunosti primjenjiv u praksi.

Ključne riječi: obrada videa, detekcija artefakta stvaranja blokova, obrada bez reference, artefakti kompresije, obrada u stvarnom vremenu

DETECTION OF BLOCKING ARTIFACT IN VIDEO

ABSTRACT

The objective of this Master Thesis was to design a blocking artefact detection and measurement algorithm for video reproduction. The algorithm, written in C programming language, works for video signals coded in MPEG-2 and H.264, in different resolutions and frame rates. The algorithm requires no data about uncompressed video signal, which means that it processes signal without reference and uses no external libraries for frame processing. By testing it is proven that it is possible to process up to 78 frames per second for the video resolution of 1920x1080 pixels in real-time. The algorithm consistency is very high 95.56%. The algorithm is completely applicable in practical tasks.

Key words: video processing, blocking artifacts detection, no-reference processing, compression artifacts, realtime processing

ŽIVOTOPIS

Tomislav Jurić rođen je 26.05.1992. u Splitu. Osnovnu i srednju školu pohađa u Našicama. Tijekom srednjoškolskog obrazovanja redoviti je sudionik natjecanja iz matematike, fizike i geografije. 2011. godine, nakon završene Prirodoslovno-matematičke gimnazije, upisuje preddiplomski studij Računarstva na Elektrotehničkom fakultetu u Osijeku, kao redovan student. Od 2014. godine član je MENSE. Na 3. godini preddiplomskog studija započinje s proučavanjem izrade web aplikacija fokusirajući se na serverski dio koristeći PHP i SQL baze podataka, te iz tog područja radi završni rad. Početkom druge godine diplomskog studija zaključuje ugovor o stipendiranju s tvrtkom „Institut RT-RK Osijek d.o.o“. Za vrijeme trajanja stipendije sluša tečaj naprednog C programiranja te dodatni izborni kolegij „Digitalna videotehnika“ kojim steče dodatna znanja iz područja digitalne televizije. U svibnju iste godine pristupa EWoB Business Hackatonu – studentskom natjecanju u kojem se testiraju logika, strategija, analitika i vodstvo, gdje se plasira među 6 finalista od 170 prijavljenih studenata Osječkog sveučilišta.

PRILOZI

Na CD-u priloženom uz diplomski rad nalaze se:

Dokumenti:

Diplomski rad – Tomislav Jurić.doc

Diplomski rad – Tomislav Jurić.docx

Diplomski rad – Tomislav Jurić.pdf