Interpolacija video okvira primjenom bilateralne procjene pokreta

Kišpećo, Igor

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

VREMENSKA INTERPOLACIJA VIDEO OKVIRA PRIMJENOM BILATERALNE PROCJENE POKRETA

Završni rad

Igor Kišpećo

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 23.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Ime i prezime studenta:	Igor Kišpečo
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	a4052, 27.09.2018.
OIB studenta:	69332961938
Mentor:	Prof.dr.sc. Snježana Rimac-Drlje
Sumentor:	Dr.sc. Denis Vranješ
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc. dr. sc. Višnja Križanović
Član Povjerenstva:	Dr.sc. Denis Vranješ
Naslov završnog rada:	Interpolacija video okvira primjenom bilateralne procjene pokreta
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak završnog rada	Današnje video aplikacije često uključuju postupke za povećanje vremenske rezolucije, što se naziva i vremensko naduzorkovanje. Povećanje vremenske rezolucije znači povećanje brzine izmjene okvira, a to znači da se neki okviri trebaju interpolirati. Kvaliteta videa značajno ovisi o algoritmu koji se koristi za ovu interpolaciju. U radu je potrebno opisati način proračuna vektora pokreta te primjenu vektora pokreta za interpolaciju video-okvira. U izabranom programskom jeziku student treba napisati program za interpolaciju video-okvira primjenom bilateralne procjene pokreta. treba napraviti usporedbu kvalitete videa primjenom Usporedbu različitih metoda vremenske interpolacije treba napraviti za nekoliko video sekvenci. Sumentor: Denis Vranješ
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Dobar (3)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 1 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 1 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	23.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.10.2019.

Ime i prezime studenta:	Igor Kišpećo
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	a4052, 27.09.2018.
Ephorus podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom: **Interpolacija video okvira primjenom bilateralne procjene pokreta**

izrađen pod vodstvom mentora Prof.dr.sc. Snježana Rimac-Drlje

i sumentora Dr.sc. Denis Vranješ

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

S razvojem računalne tehnologije paralelno se razvija i audio/video tehnika. Sukladno tome danas se snimaju video zapisi iznimno visoke kvalitete i rezolucije. Međutim visoka kvaliteta video i audio zapisa ne dolazi bez nedostataka. Veličine video zapisa postale su relativno velike i ako se pri tome video zapis mora prebaciti preko internet mreže od točke A do točke B u što kraćem vremenskom periodu ili se prikazuje u realnom vremenu, onda veličina datoteke počinje predstavljati problem. Kako bi se riješio taj problem zadnjih tridesetak godina intenzivno se razvijaju napredni programi i algoritmi za kodiranje audio/video zapisa kako bi informaciju koju prenosi video zapis uspjeli sažeti u što manji memorijski prostor. Kod prijenosa videa mrežama manjeg ili promjenjivog kapaciteta, u nekim se slučajevima može dobiti bolja kvaliteta komprimiranog videa ako se prije postupka kompresije smanji prostorna ili vremenska rezolucija. Na prijemnoj strani se nakon postupka dekodiranja provodi i postupak povećanja rezolucije na originalnu. Pri tome se kod povećanja vremenske rezolucije provodi postupak povećanja brzine izmjene slika što znači da se slike (okviri) koji su na predajnoj strani uklonjeni zbog smanjenja vremenske rezolucije, na prijemnoj strani moraju nadomjestiti.

Tema ovog rada je postupak povećanja brzine izmjene slika u videu interpolacijom slika primjenom bilateralne procjene pokreta. To je jedan od mogućih postupaka povećanja vremenske rezolucije, a u radu je uspješnost ovog postupka uspoređena s uspješnosti povećanja vremenske rezolucije primjenom usrednjavanja i primjenom bilateralne procjene pokreta s preklapajućim blokovima.

Rad je koncipiran u pet poglavlja. Nakon uvoda, drugo poglavlje opisuje osnovne parametre vezane za digitalnu sliku i video, treće poglavlje obrađuje interpolaciju video okvira primjenom bilateralne procjene pokreta, a u četvrtom poglavlju su dani rezultati testiranja različitih algoritama za vremensku interpolaciju slika te njihova usporedba. Zaključci su dani u petom poglavlju.

2. PARAMETRI DIGITALNE SLIKE I VIDEO SEKVENCE

Osnovni i elementarni dio svake video sekvence je digitalna slika. U ovom su poglavlju opisani glavni parametri koji karakteriziraju digitalnu sliku i video sekvencu.

2.1. Digitalna slika

Monokromatska mirna slika predstavlja se dvodimenzionalnom matricom veličine **MxN** kao što je prikazano izrazom (2-1).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ f(M-1,0) & f(M-1,1) & & f(M-1,N-1) \end{bmatrix} \tag{2-1}$$

Prostorna rezolucija slike sadrži podatak koliko se koristi elemenata slike (piksela) za njen prikaz i ovisi o kvaliteti uzorkovanja slike. Druga bitna karakteristika slike je dubina boje koju će slika posjedovati. Ona ovisi o modelu boja koji za video mogu biti YUV, YCrCb, YIQ…itd. (Y označava luminantnu komponentu, U i V su krominantne komponente). Model boja koji će se koristiti ovisi o sadržaju, prijemniku, tipu signala, kanalu i drugim paramterima. Dubina boje također ovisi kvantizaciji i broju kvantizacijskih razina L koje će se koristiti.

$$L = 2^{k} \tag{2-2}$$

Gdje je:

-L = broj kvantizacijskih razina

-k = broj korištenih bita

Za monokromatsku sliku se najčešće koristi 8 bit-a po elementu slike, no moguće je koristiti i 10,12,16 bit-a po elementu čime se dobije vjerniji prikaz sadržaja slike, ali raste joj i veličina.

Pri ovome treba imati na umu da se kod kromatskih slika koristi 8 bit-a samo za jednu komponentu, dok npr. RGB model boja ima tri komponente, pa se za svaku od njih koristi 8 bit-a, što je u konačnici 24 bit-a samo za krominantnu komponentu. S 8 bit-a može se prikazati 256

razina (nijansi) crvene, 256 razina zelene i 256 razina plave, što u kombinaciji daje preko 16 milijuna boja.

2.2. Video sekvenca

Ljudski vizualni sustav prilično je kompleksan, a početak obrade slike počinje u ljudskom oku. Ljudsko oko je najosjetljivije na srednjim frekvencijama (3-5 perioda po 1° vizualnog kuta) te na luminantnu komponentu slike. Nadalje, za percepciju kontinuiranog pokreta, potrebna je izmjena najmanje 24 slike u sekundi (engl. *fps – frames per second*). Naime, zbog pojave tzv. „tromosti oka“, kod 24 i više izmjena u sekundi, čovjek vidi niz slika kao kontinuirani pokret, odnosno video sekvencu.

Osjetljivost oka u odnosu na broj izmjena sličica:

<10 – vidi se izmjena slika

10-16 – isprekidani pokreti

24fps – filmska traka

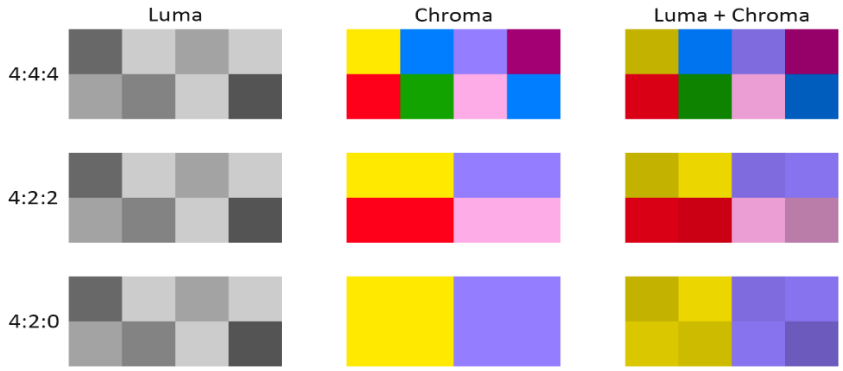
25fps – TV PAL

30fps – TV NTSC

60fps – HDTV

Što je veći broj izmjena u sekundi to je pokret glađi, ali i veličina datoteke je veća te je duljina kodiranja i dekodiranja signala dulja. Stoga se, ovisno o potrebama, video sekvenca kodira u različitim standardima i različitim algoritmima. Jedna od karakteristika pri kodiranju je gore spomenuta osjetljivost oka na luminaciju i krominaciju. Ukoliko se smanji količina krominantnih informacija, slika će zauzimati memorije bez značajnog gubitka na kvaliteti. Stoga se često koriste tipovi kompresije (poduzorkovanje boje) poput 4:2:2 ili čak 4:2:0. Prvi broj označava veličinu uzorka, dva sljedeća broja odnose se na boje. Oba su relativna u odnosu na prvi broj i definiraju horizontalno i vertikalno poduzorkovanje. Signal s formatom poduzorkovanja 4:4:4 nema kompresije, informacije o krominaciji i luminaciji prenose se u cijelosti. Kod 4:2:2 formata poduzorkovanja se prenosi samo pola informacija o boji u odnosu na 4:4:4 format poduzorkovanja,

dok se kod signala s formatom poduzorkovanja 4:2:0 prenosi samo četvrtina. Za signal s formatom poduzorkovanja 4:2:2 uzorkovat će se svaki drugi uzorak horizontalno, a vertikalno u svakoj liniji videa. Za format poduzorkovanja 4:2:0 analizirati će se svaki drugi uzorak horizontalno, a vertikalno u svakoj drugoj liniji videa, što se može vidjeti na slici 2.1.



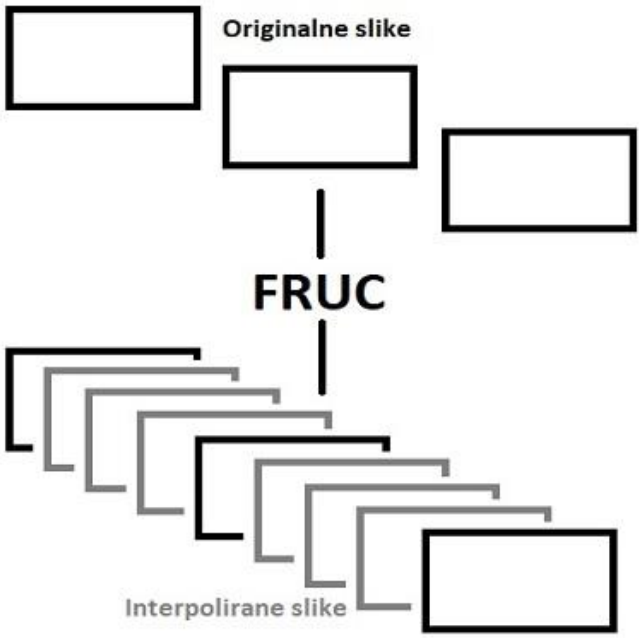
Slika 2.1. Poduzorkovanje boje, [9]

Neki od najčešće korištenih standarda kodiranja videa su MPEG-2 i H.264 (MPEG4 part10 AVC), [1]. Problem današnjice kad je u pitanju prikaz videa je taj što postoji veliki broj različitih predočnika kao i različitih izvora videa (24Hz film, 60Hz video, 120Hz LCD display…). Kako bi se umanjila razlika koja bi utjecala na subjektivnu ocjenu videa kod gledatelja, koriste se tehnike poput konverzije brzine izmjene slike (*eng. Frame rate conversion*). To su tehnike koje pomoću procjene pokreta između slika mogu interpolirati nove slike te time povećati broj izmjena slika u jedinici vremena. Veća brzina izmjene slika generira realniji i glađi pokret unutar video sekvence, pri čemu se ne opterećuje prijenosni kanal i propusnost kanala jer se sama konverzija vrši unutar dekodera.

3. VREMENSKA INTERPOLACIJA

Da bi se smanjile veličine podataka za prijenos video zapisa koriste se tehnike kompresija video sekvence. Tehnike video kompresije koje postižu visoki stupanj kompresije u današnje su vrijeme iznimno bitne kod aplikacija koje služe za višekorisničke video konferencije. Kad se internetom prenosi video sadržaj uživo, često se koristi reduciranje vremenske rezolucije video sekvence (*eng. temporal sub-sampling*) i/ili prostorno smanjivanje rezolucije videa kako ne bi bilo zahtjeva za visokom propusnosti na mreži. Reduciranje vremenske rezolucije znači smanjenje brzine izmjene slika u sekundu, a postiže se izbacivanjem nekih slika iz video sekvence prije kodiranja (kompresije). Da bi se rekonstruirao originalni broj slika u sekundi te poboljšala kvaliteta zapisa, u dekoderu se izbačeni slike rekonstruiraju primjenom interpolacije. Glavni problem u ovom postupku je kako što vjernije rekonstruirati slike koje nedostaju, a da se pritom koristi što manje računalnih resursa, odnosno kako postići da računanje i predviđanje bude što točnije i brže. [2]

Povećanje brzine izmjene video slika (eng. *Frame Rate Up Conversion* – FRUC) je tehnika konverzije niže brzine izmjene slika u višu brzinu izmjene, pri čemu se video sekvenci dodaju slike metodom interpolacije, (slika 3.1.).



Sl. 3.1. Povećanje brzine izmjene video okvira (FRUC)

Osim kod prijenosa videa malim brzinama, FRUC tehnika se koristi i za smanjenje zamućenja slike kod ekrana LCD (eng. *Liquid Crystal Display*) tipa, gdje se zbog tromosti kod rotacije polarizacije kristala pojavljuje zamućenje pokreta (eng. *Motion blur*), te se time degradira kvaliteta slike. Najčešće rješenje je konverzija ulaznog videa iz 50 fps-a u 100 fps-a, što se pokazalo kao efikasno sredstvo u održavanju kvalitete i oštrine izlazne slike.[4]

Često se koriste jednostavne interpolacijske metode poput ponavljanja slika i nadomještanje slikom nastalom kao srednja vrijednost postojećih susjednih slika, (eng. *Frame Averaging -FA*), metode čija je kompleksnost mala. Ti algoritmi se uspješno primjenjuju u sekvencama u kojima nema puno pokreta, dok u intenzivnijim scenama izazivaju pojavljivanje zamućivanja i trzaje u sceni. S toga su se razvile nove tehnike interpolacije koje koriste kompenzaciju pokreta u video sekvencama (eng. *Motion Compensated Frame Interpolation* – MCFI), koja predviđa kretanja između slika te time poboljšava interpolaciju novih slika. [2]

Povećanje brzine izmjene slika estimacijom pokreta (eng. *Motion Compensated Frame Rate Up Conversion* – MC FRUC). MC FRUC se sastoji od dvije komponente: procjena (estimacija) pokreta (eng. *Motion Estimation* - ME) i interpolacija kompenzacijom pokreta (eng. *Motion Compensated Interpolation* - MCI). Interpolacija slika kompenzacijom pokreta bazira se na pretpostavci da je pokret u video sekvenci gladak, fluidan i translacijski. Ova je pretpostavka često točna za većinu video sekvenci, posebno za sekvence u kojima ima relativno malo pokreta. Ideja je da se uradi procjena pokreta između prethodne i trenutne slike kako bi se dobili vektori pokreta, te se zatim uzima srednja vrijednost dobivenih vektora pokreta čime se predviđa pomak između prethodne slike i one koju treba interpolirati ili trenutne i one koju treba interpolirati (vremenski postavljene između trenutne i prethodne slike). Na temelju te procjene pokreta između postojećih slika i one koju treba interpolirati, određuju se vrijednosti elemenata slike u novoj interpoliranoj video slici. Pri tome je bitno dobiti što točnije vektore pokreta, te je to ujedno i jedan od glavnih istraživačkih ciljeva: što bolja estimacija pokreta. Dvije glavne kategorije metoda za procjenu pokreta su: procjena na osnovu blokova (eng. *block based*) i procjena na osnovu elemenata slike (eng. *pixel- wise*). Procjena na osnovu elemenata slike dati će točnije rezultate, ali zato zahtijeva iznimno puno resursa i računalnih operacija, zbog čega se više i češće koristi kod off-line interpolacija nego kod video sadržaja koji zahtijeva interpolaciju u stvarnom vremenu. Za razliku od toga, algoritmi koji rade provjeru podudarnosti blokova elemenata slike (eng. *Block Matching Algorithm* – BMA) mogu se relativno lako i efikasno implementirati, a omogućuju zadovoljavajuće rezultate. BMA nije idealna tehnika jer ne pronalazi istinski pokret, već traži vektore pokreta koji imaju najmanju rezidualnu energiju, ali se najviše koristi kod interpolacija

video okvira zbog svoje relativno jednostavne računice. Također, BMA ima tendenciju stvaranja blok artefakata, poput preklapanja područja kompenzacije, te dijelova koji ne mogu biti kompenzirani. Postoje algoritmi koji bolje predviđaju i otkrivaju pravi pokret (eng. True motion) od klasičnih BMA algoritama, algoritmi poput: 3D recursive search, Bayesian motion estimation, phase plane correlation...itd.[4, 5]

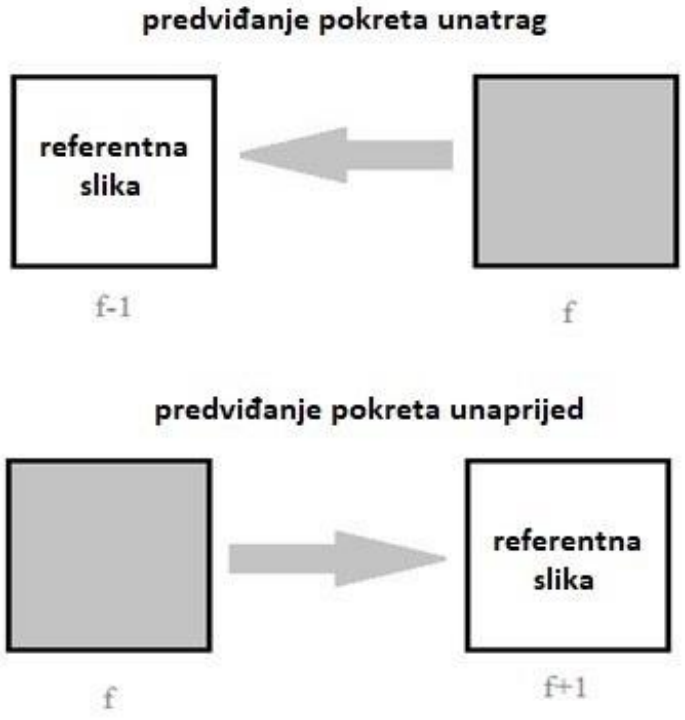
Video kodiranje koje koristi manje blokove pri estimaciji pokreta dati će točnije rezultate, manju energiju kod rezidualnih slika te će samu kompresiju učiniti efikasnijom. Iako se manji blokovi poput 8x8 ili 4x4 često koriste kod video standarda poput H.264, ipak u konačnici kod interpolacije okvira procjenom pokreta najviše se koriste blokovi veličina 16x16 piksela. [2, 3]

3.1. Procjena (estimacija) pokreta

Procjena pokreta i proračun vektora pokreta provodi se za promatranu sliku u odnosu na neku referentnu sliku, koja može biti ili prethodna ili naredna slika u odnosu na promatranu, kako je prikazano na slici 3.2. Ovisno o položaju referentne slike postoji:

- predikcija pokreta unatrag (engl. *backward motion estimation*) ako se koristi prethodna slika za referentnu;

- predikcija pokreta unaprijed (engl. *forward motion estimation*) ukoliko se koristi naredna slika kao referentna. [5]



Sl. 3.2. Predviđanje pokreta

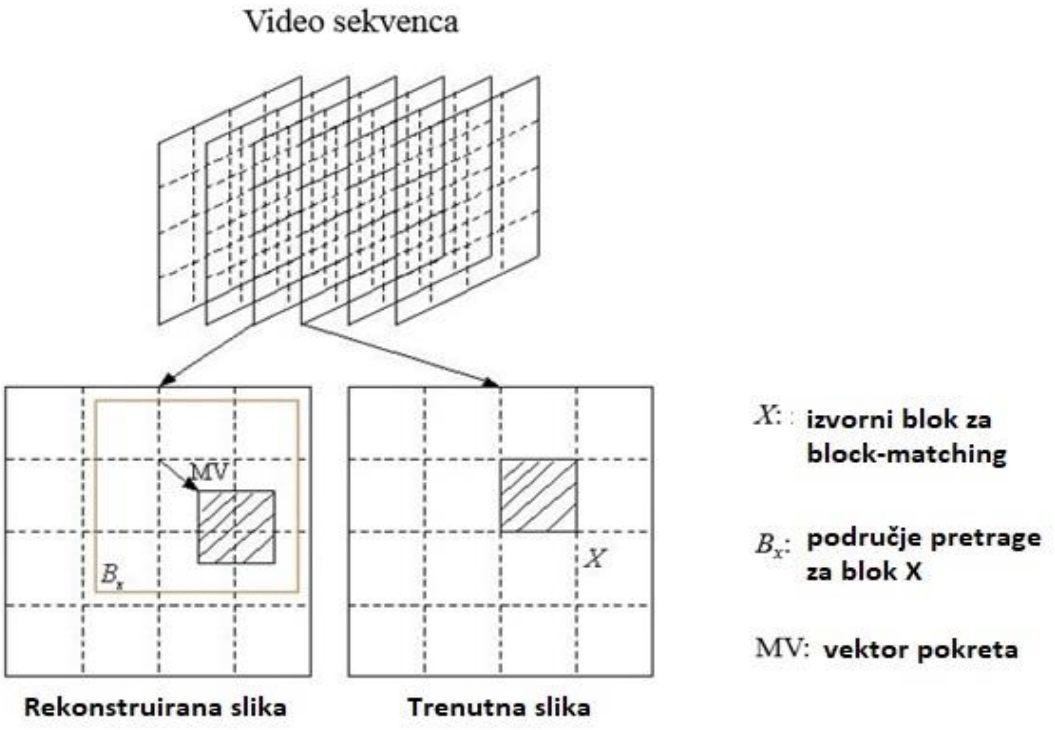
Računalno efikasan način procjene i kompenzacije pokreta je *block-matching*, odnosno uspoređivanje makro blokova unutar dviju slika u svrhu dobivanja vektora pomaka.

Block-matching algoritam je način lociranja podudarajućih makroblokova unutar video slika u svrhu otkrivanja pomaka u video sekvenci. Temeljna pretpostavka kod predviđanja pokreta je da se određeni objekti ili pozadine pojavljuju skoro identično ili slično u narednim slikama, uz mogući pomak objekata ili dijelova objekata iz jedne u drugu sliku. U postupku kompresije to se koristi kako bi se otkrila temporalna redundancija u video sekvenci, koja u konačnici, definiranjem sadržaja makro-bloka pomoću referentnog makro-bloka povećava efektivnost međuslikovne video kompresije.[3]. U postupku procjene pokreta za potrebe interpolacije nedostajuće slike (tj. povećanja brzine izmjene slika), procjena pokreta između postojećih slika je prvi korak u procjeni pokreta između postojećih i slike koju treba interpolirati.

Block-matching metoda (Slika 3.5.) radi na principu da se trenutna slika u videu podijeli u makroblokove veličine MxM elemenata slike, najčešće 16x16, te uspoređuje s odgovarajućim blokom iz susjednih slika (prethodne, ili u slučaju bidirekcijskog predviđanja i buduće).

Pronalaženjem istih ili približno istih blokova u susjednim slikama kreiraju se vektori pokreta. Vektor pokreta govori dekoderu koliko je i u kojem smjeru određeni blok, a time i element slike unutar bloka pomaknut. Drugim riječima vektori pokreta tvore estimaciju pokreta u video okviru.

Zona pretrage za odgovarajućim makro-blokom u referentnoj slici je definirana „područjem pretrage“(*p)*, gdje je *p* broj elemenata slike u sva četiri smjera u odnosu na promatrani makro-blok. Područje pretrage je mjera pokreta: što je veća vrijednost ovog parametra, veća je potencijalna kretnja i vjerojatnost pronalaska odgovarajućeg sličnog ili istog makro-bloka u susjednim slikama. Pretraživanje cjelokupne slike bilo bi idealno, ali i iznimno računski zahtjevno, pa se za makro-blok od 16x16 elemenata slike najčešće uzima područje pretrage od 7 elemenata slike u svaku stranu.



Sl. 3.5. Grafički prikaz block-matching algoritma [4]

Block-matching algoritam traži podudarnost blokova na osnovu matematičkih funkcija, i najčešće su to funkcija srednje apsolutne vrijednosti (3-1) (*eng. Mean absolute difference – MAD*) i funkcija srednje kvadratne pogreške (3-2) (*eng. Mean squared error – MSE*).

Mean Absolute Difference (MAD):

$$\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij}-R_{ij}\right| \tag{3-1}$$

Mean Squared Error (MSE) =

$$\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(C_{ij}-R_{ij}\right)^2 \tag{3-2}$$

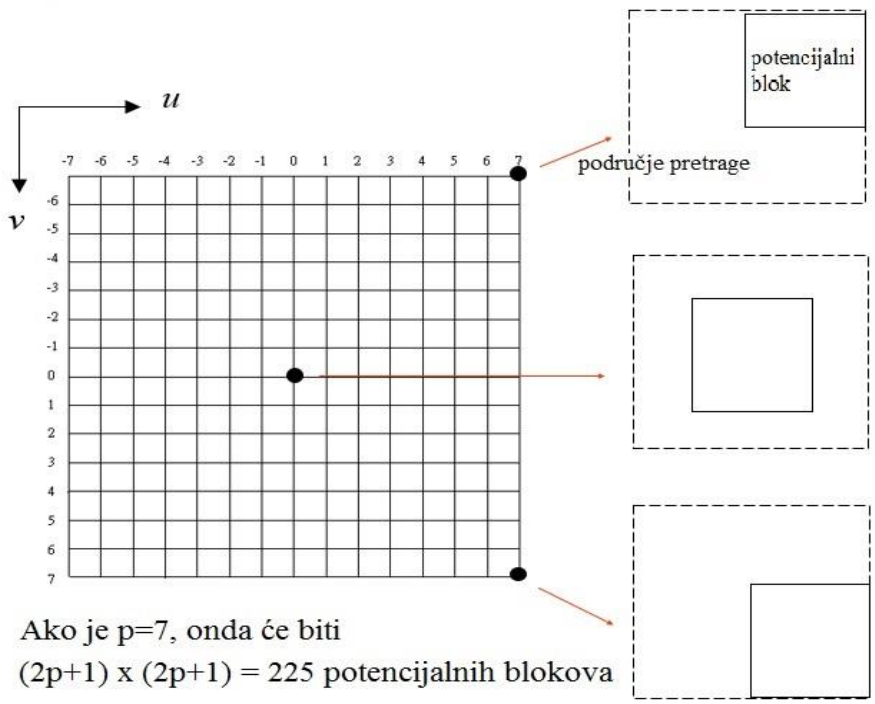
U formulama za računanje sličnosti između makroblokova MAD i MSE, *N* predstavlja veličinu makro-bloka , C_{ij} i R_{ij} predstavljaju elemente slike koji se uspoređuju u trenutnom makro-bloku i referentnom (izvornom) makrobloku. [3, 4]

3.2. Algoritmi pretrage makro-blokova

Kako bi se pretraga makro-blokova smanjila na što manje koraka te time podigla brzina i efikasnost pretrage, koriste se razni algoritmi za pretragu makro-blokova. Idealna pretraga značila bi pretraživanje područja cijele slike, no to bi bilo računski i vremenski zahtjevno. Zato su se s vremenom razvile metode koje su dovele do brže i efikasnije pretrage makro-blokova. [3, 6]

Postoji velik broj algoritama pretrage, no neki od poznatijih i češće korištenih su algoritam pune pretrage i algoritam tri koraka.

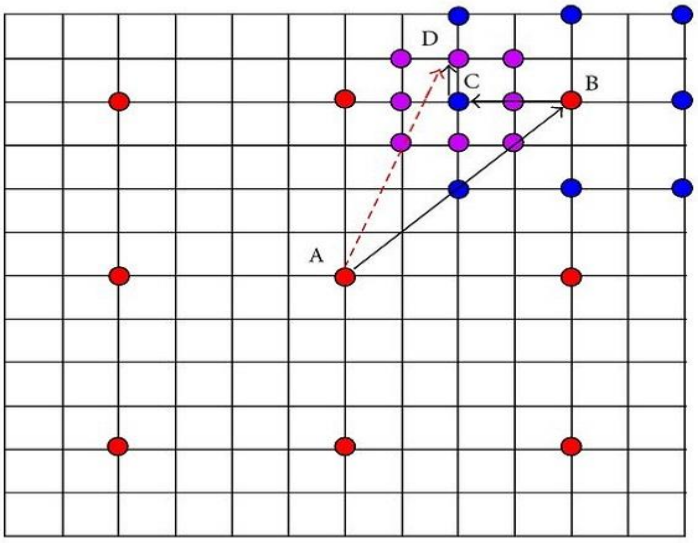
Algoritam pune pretrage (*eng. Full search algorithm/exhaustive search***)** prikazan je na slici 3.6.



Sl. 3.6. Algoritam pune pretrage [6]

Ovaj algoritam vrši pretragu za potencijalnim makroblok kandidatima na svakom elementu slike unutar područja pretrage zbog čega daje najtočniji rezultat. Slika koja se dobije kao rezultat pretrage ima najviši PSNR (*eng. Peak signal-to-noise ratio*) u usporedbi s bilo kojim drugim algoritmom. No cijena te točnosti je visoka zahtjevnost za računskim i vremenskim resursima.

Algoritam tri koraka (*eng. Three step search algorithm TSS*) prikazan je na slici 3.7.



Sl. 3.7. Algoritam tri koraka [8]

Algoritam tri koraka (TSS) (Slika 3.7.) jedan je od prvih brzih algoritama za usporedbu makro blokova. Princip njegovog rada je sljedeći:

1. Započinje pretragu u centru (lokacija 0, 0) područja pretrage.
2. Postavlja korak pretrage „S“, S = 4, i postavlja područje pretrage „p“, p = 7.
3. Pretražuje osam lokacija +/- S elemenata slike oko lokacije (0, 0) uključujući i centar
4. Odabire jednu od devet lokacija, onu s najmanjom vrijednosti MAD funkcije.
5. Postavlja novu centralnu lokaciju u odabranu točku.
6. Postavlja novi korak pretrage kao S = S / 2.
7. Ponavlja proceduru dok S = 1.

Rezultat lokacije kad je S = 1 je onaj koji ima minimalnu MAD funkciju i makro blok na toj lokaciji je najbolji kandidat podudaranja. Kompleksnost proračuna je smanjena za faktor 9 jer algoritam pune pretrage treba napraviti proračun MAD funkcije za 225 potencijalnih kandidatskih makroblokova za područje pretrage *p*=7, dok TSS algoritam taj proračun treba napraviti za samo 25 kandidatskih makroblokova [6].

Navedena su samo dva algoritma za komparaciju makro blokova (ta dva su korištena u praktičnom primjeru interpolacije), ali postoje još mnogi drugi algoritmi i svi imaju svoje pozitivne i negativne strane u specifičnim situacijama. Neki od njih su: *New Three Step Search* algoritam, *Simple and Efficient Search*, *Four Step Search* algoritam, *Diamond Search*, *Adaptive Rood Pattern Search* algoritam [6].

Svi navedni algoritmi mogu se koristiti kod predviđanja pokreta unaprijed, unatrag ili prilikom bilateralne procjene pokreta, odnosno pri jednosmjernom i dvosmjernom predviđanju. Kod unidirekcijskog predviđanja, ovisno o tome je li predviđanje unaprijed ili unatrag, na blokove se dijeli prethodna ili trenutna slika, te definira područje pretrage i traži se odgovarajući blok u prethodnoj, odnosno trenutnoj slici. Tu se najčešće javlja problem preklapajućih blokova te „rupe“ između blokova nakon primjena vektora pokreta. Ti problemi se mogu ublažiti ili riješiti primjenom bilateralne procjene pokreta pri kojoj se potencijalna interpolirana slika dijeli na mrežu nepreklapajućih blokova te se traži blok u budućoj slici koji odgovara bloku iz prethodne slike oko te lokacije [3, 4, 5].

3.3. Dvosmjerna (bilateralna) procjena vektora pokreta

Dvosmjerna ili bilateralna procjena pokreta je dizajnirana tako da eliminira rupe između blokova i preklapanje blokova koji se događaju kod unidirekcijske procjene te se kasnije moraju primijeniti druge metode kako bi se uklonili ti nedostatci. Za razliku od unidirekcijske procjene pokreta kod koje se na blokove dijeli buduća ili prethodna slika, bilateralna predikcija koristi sliku koja će tek biti interpolirana te se ta slika dijeli na nepreklapajuće blokove. Oni još uvijek ne sadrže informaciju o vrijednostima elemenata slike no služe kao referentna točka lokacije. Procjenom pokreta traži se blok iz prethodne slike koji odgovara bloku u budućoj slici, pri čemu vektor pomaka prolazi kroz referentnu lokaciju bloka u slici koja se interpolira, (slika 3.8.). Odgovarajući blokovi u prethodnoj i trenutnoj slici imaju relativne pozicije (*dx*, *dy*) i (*-dx*, *-dy*) u odnosu na interpoliranu sliku.

Slika 3.8. opisuje bilateralnu procjenu pokreta. U interpoliranoj slici f_n pozicija elementa slike je označen sa **s**, **v** označava vektor pokreta bloka kandidata. Pozicija elementa slike u interpoliranoj slici je povezan s pozicijom elementom slike u prethodnoj slici f_{n-1} preko formule 3-3.

$$s_b = s - v \qquad (3\text{-}3)$$

Vektor pokreta unaprijed je povezan s vektorom pokreta u slici unatrag, tako da pozicija odgovarajućeg elementa u narednoj slici f_{n+1} odgovara formuli 3-4.

$$s_f = s + v \qquad (3\text{-}4)$$

U ovom slučaju s_b i s_f nalaze se nasuprot jedan drugog u odnosu na **s**. Bilateralna procjena pokreta koristi model pokreta bloka, $B_{i.j}$ označava blok u interpoliranoj slici.

Za svaki kandidat vektora pokreta **v** računa se suma apsolutne bilateralne razlike (3-5) (SBAD engl. *Sum of bilateral absolute difference*):

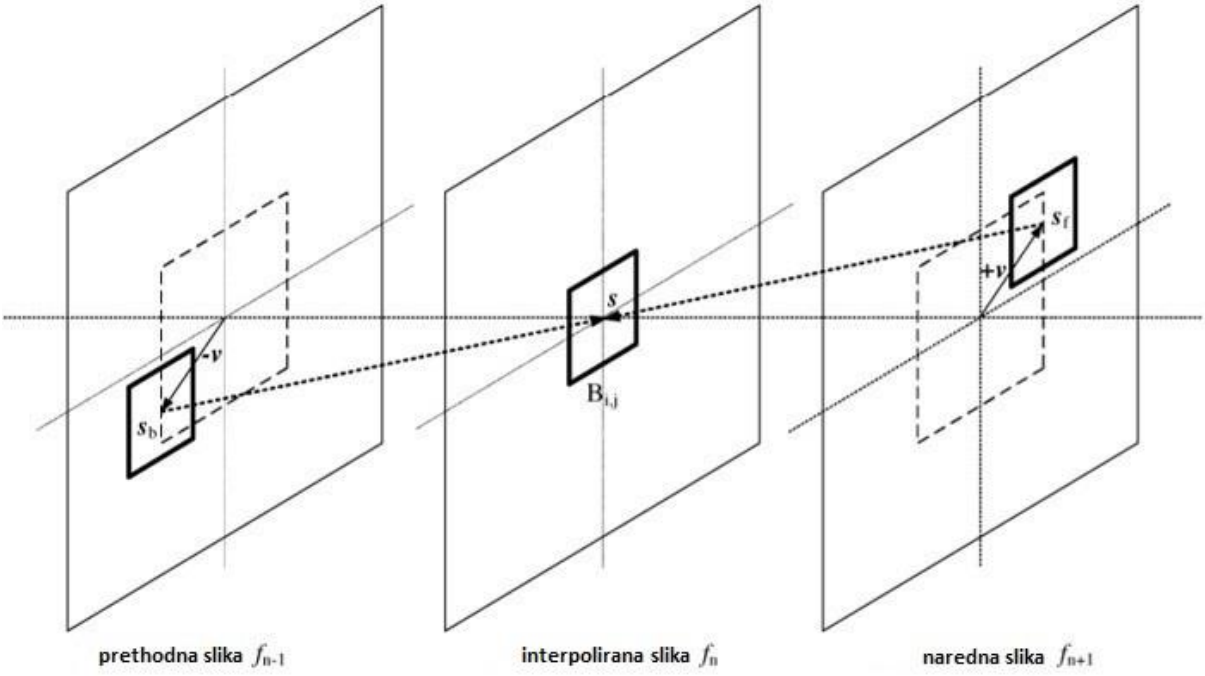
$$SBAD(B_{ij}, v) = \sum_{s \in B_{ij}} | f_{n-1}(s - v) - f_{n+1}(s + v)|$$

(3-5)

zatim se pronalazi vektor pomaka koji minimizira SBAD (3-4) pomoću:

$$v_{ij} = \arg\min_{v} \{SBAD\ (B_{ij}, v)\}$$

(3-6)



Sl. 3.8. Bilateralna procjena pokreta [2]

Najčešći problem kod bilateralne procjene pokreta je detekcija malih objekata koji se brzo kreću pa ih se detektira i obrađuje kao dva zasebna objekta, jedan koji postoji u prethodnoj slici a nema ga u trenutnoj, i jedan koji ne postoji u prethodnoj slici a pojavljuje se u trenutnoj. U procesu kompenzacije pokreta pojavljuje se sličan vizualni efekt kao kod interpolacije usrednjavanjem ili se objekt pojavi samo na kratko prije nego nestane u potpunosti. Razne metode se mogu koristiti kako bi se riješio taj problem te kako bi estimacija pokreta bila što preciznija. Choi u svom radu [2] predlaže SMD (*eng. Side match distortion*), odnosno da se ograničavanjem prostornog zaglađivanja poboljša robusnost bilateralne procjene pokreta na način da se izmjeri glatkoća na rubovima bloka, kako je prikazano na slici 3.9., SMD računa prosječnu vrijednost apsolutne razlike između piksela predviđenog bloka i susjednih blokova uzduž sve četiri strane ruba, prema (3-7):

$$SMD(B_{ij}, v) = \frac{1}{N}\sum_{k=0}^{N-1} \left|\widehat{f_n}(g_k, v) - \widehat{f_n}(h_k)\right|$$

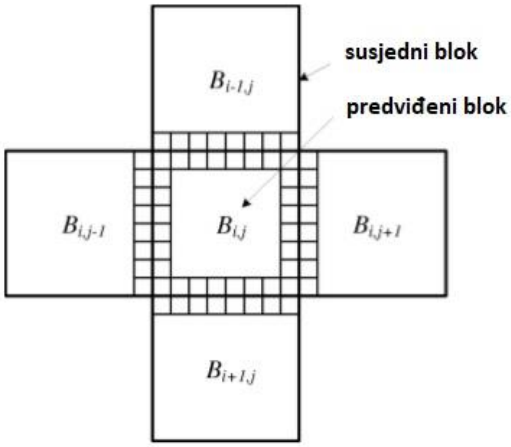
(3-7)

gdje g_k i h_k označavaju poziciju *k*-tog piksela na rubovima između predviđenog bloka i susjednih blokova, a *N* označava broj rubnih piksela. Nadalje, $\widehat{f_n}(g_k, v)$ označava vrijednost interpoliranog piksela u predviđenom bloku, kad mu je vrijednost vektora pokreta jednaka *v*. Treba uzeti u obzir da vrijednost interpoliranih piksela $\widehat{f_n}(h_k)$ u susjednim blokovima mora biti dostupna prije računanja SMD-a, stoga se procjena pokreta aplicira iterativno. U konačnici najbolji rezultat za dobijanje točnijeg vektora pokreta v_{ij} za predviđeni blok B_{ij} računa se prema (3-8):

$$v_{ij} = \arg\min_{v} \{\mu \cdot SBAD\left(B_{ij}, v\right) + (1-\mu) \cdot SMD\left(B_{ij}, v\right)\}$$

(3-8)

gdje je μ težinski koeficijent. S obzirom da računanje SMD-a zahtjeva vrijednost piksela u svim okolnim blokovima, estimacija vektora pokreta ovisi o vrijednostima ostalih blokova koje se mijenjaju sa zaglađivanjem. Stoga se minimizacija funkcije (3-8) ponavlja za sve blokove u interpoliranoj slici, dok se predviđeni vektori pokreta ne prestanu mijenjati [2].

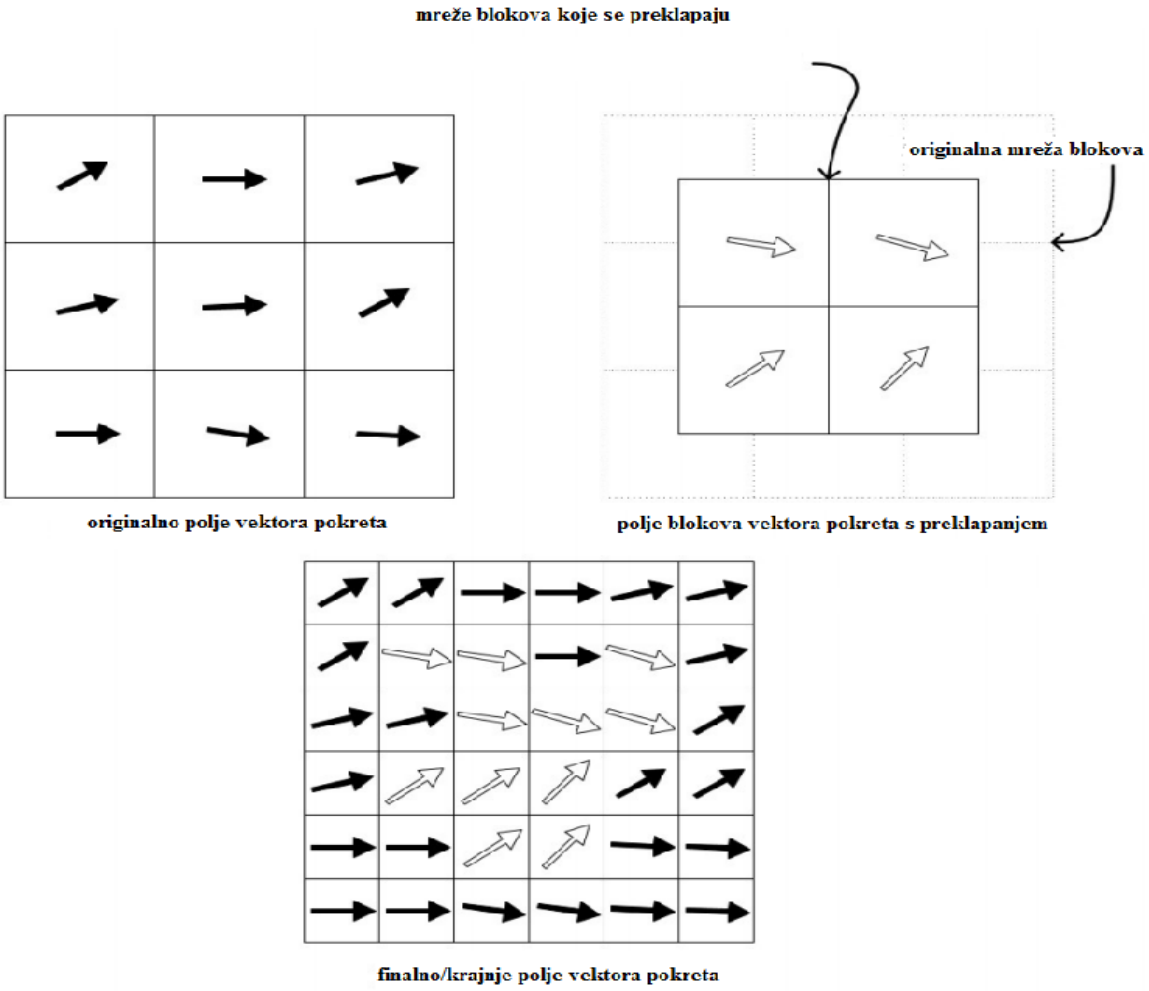


Sl. 3.9. Poručje zaglađivanja susjednih blokova [2]

3.3.1. Proširena bilateralna procjena pokreta

Proračun kod bilateralne procjene pokreta je puno jednostavniji nego kod jednosmjernih algoritama blok predikcije zato što je okvir pretrage kod bilateralne procjene pokreta jedna

četvrtina okvira za blok predikciju. No ako putanja pokreta nije simterična iz aspekta srednjeg okvira neće se moći dobiti pravi vektor pokreta. Proširena bilateralna predikcija (EBME engl. *extended bilateral motion estimation*) [7] izvodi bilateralnu procjenu pokreta i za preklapajuće blokove kako bi našla što preciznije vektore pokreta. Slika 3.10. prikazuje kako EBME modificira polje vektora pokreta kako bi bilo preciznije od originalnih polja vektora pokreta. Uspoređujući sumu bilateralnih razlika originalne mreže blokova s preklapajućom mrežom blokova određuje se krajnje polje vektora pokreta .

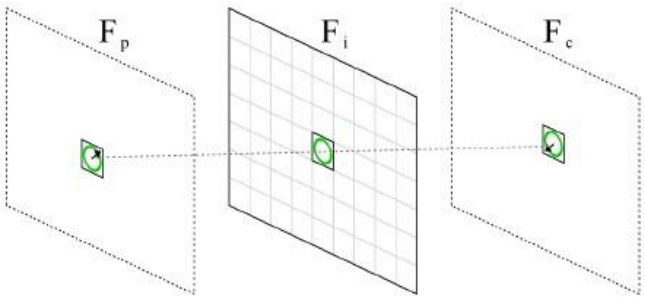


Sl. 3.10. Proširena bilateralna estimacija pokreta (EBME) [7]

3.4. Kompenzacija pokreta

Kompenzacija pokreta se koristi kako bi se aproksimirala slika F_i iz predviđenog polja vektora pokreta te iz referentnih slika F_p i F_c (slika 3.11.). Kod kompenzacije pokreta makro blokovima

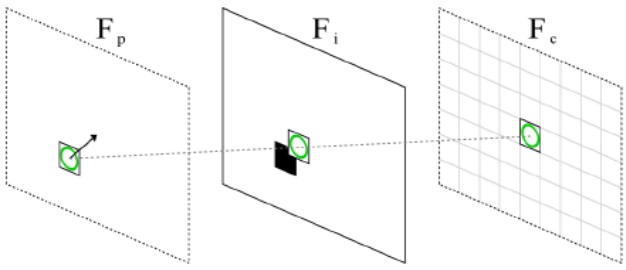
(*eng. Block motion compensation – BMC*) vektori pokreta predstavljaju translacijski pomak blokova između dva okvira te se zato metoda zove kompenzacija pokreta makroblokovima. Kada se za *BMC* koriste vektori pokreta dobijeni primjenom bilateralne procjene pokreta, F_i okvir može biti interpoliran direktno iz vrijednosti elemenata iz slika F_p i F_c na lokaciji izračunatoj pomoću vektora pokreta (slika 3.11.).



Sl. 3.11. Kompenzacija pokreta pomicanjem makrobloka [5]

Najjednostavniji način određivanja elemenata interpoliranog bloka je srednja vrijednost elemenata odgovarajućih blokova u F_p i F_c.

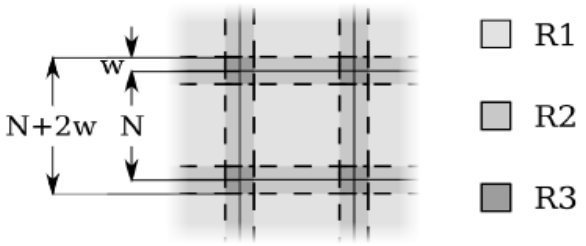
Kod unilateralne procjene i kompenzacije pokreta određivanje elemenata bloka koji se interpolira, F_i , zasniva se na direktnom pomicanju blokova tako da se primjene vektori pokreta na blokove u slici F_p na pola duljine vektora pokreta prema odgovarajućoj poziciji u slici F_c, ili obratno ukoliko se radi o kompenzaciji pokreta unatrag, odnosno unaprijed. U tom slučaju blokovi će se najčešće preklopiti te nakon samog pomjeranja ostat će rupe u generiranoj slici, kao što se vidi na slici 3.12.[5].



Sl. 3.12. Direktno pomicanje bloka unutar referentnih okvira [5]

Prednost bilateralne procjene pokreta je što se ne pojavljuju rupe koje nastaju pomicanjem blokova nakon primjene vektora pokreta. Kako bi se popunile praznine potrebno je konvertirati staro polje vektora pokreta u bilateralno polje vektora pokreta, a to zahtjeva da se prati putanja i da se dodaju kandidatski vektori pokreta. Najbolje odgovarajući vektori pokreta se zatim odabiru za popunjavanje rupa.

Algoritmi za kompenzaciju pokreta makroblokovima uzrokuju pojavljivanje blok artefakta, ali se ti artefakti mogu ublažiti koristeći procjenu pokreta pomoću preklapajućih blokova (*eng. Overlapping block motion compensation – OBMC*). Blokovi se proširuju tako da se sam blok centrira u njegovoj originalnoj lokaciji, a veličina bloka se poveća. Tako se generira polje preklapajućih blokova te se njihovi rubni dijelovi, mogu interpolirati sa rubnim dijelovima susjednih blokova. Choi (2000.) u svom radu predlaže da se uvećani blokovi podijele u tri skupine (po regijama bloka), prva regija je sam originalni blok, odnosno dijelovi koji se ne preklapaju (R1), zatim regija koja se preklapa između dva bloka (R2) i regija koja se preklapa između četiri bloka (R3), što se vidi na slici 3.13.. Vrijednosti elemenata regije R1 se dobiju koristeći dobijene vektore pokreta, dok se vrijednosti elemenata za R2 i R3 računaju na temelju vrijednosti elemenata u blokovima koji se preklapaju na toj lokaciji [5].



Sl. 3.13. Kompenzacija pokreta pomoću preklapajućih blokova – *OBMC* [5]

4. ZADATAK I REZULTATI PRAKTIČNOG DIJELA RADA

U okviru završnog rada napravljen je kôd u programskom paketu Matlab u kojem je implementiran postupak povećanja brzine izmjena slika u video sekvenci uporabom vremenske interpolacije video okvira primjenom bilateralne procjene pokreta. Za analizu kvalitete interpolacije primjenom izrađenog koda izabrane su 4 video sekvence sa značajno različitim sadržajima: *news*, *football*, *mobile* i *foreman*. Izbacivanjem svake druge slike, izabranim video sekvencama promijenjena je brzina izmjena slika sa 25 slika u sekundi na 12.5 slika u sekundi. Primjenom napravljenog koda brzina izmjena slike je vraćena na prvotnu vrijednost te je izračunata vrijednost PSNR (eng. *Peak Signal to Noise Ratio*) kao mjera odstupanja video sekvence s interpoliranim slikama od originalne sekvence. PSNR se računa primjenom (4-1)

$$PSNR = 10\ \log_{10}\left(\frac{R^2}{MSE}\right) \tag{4-1}$$

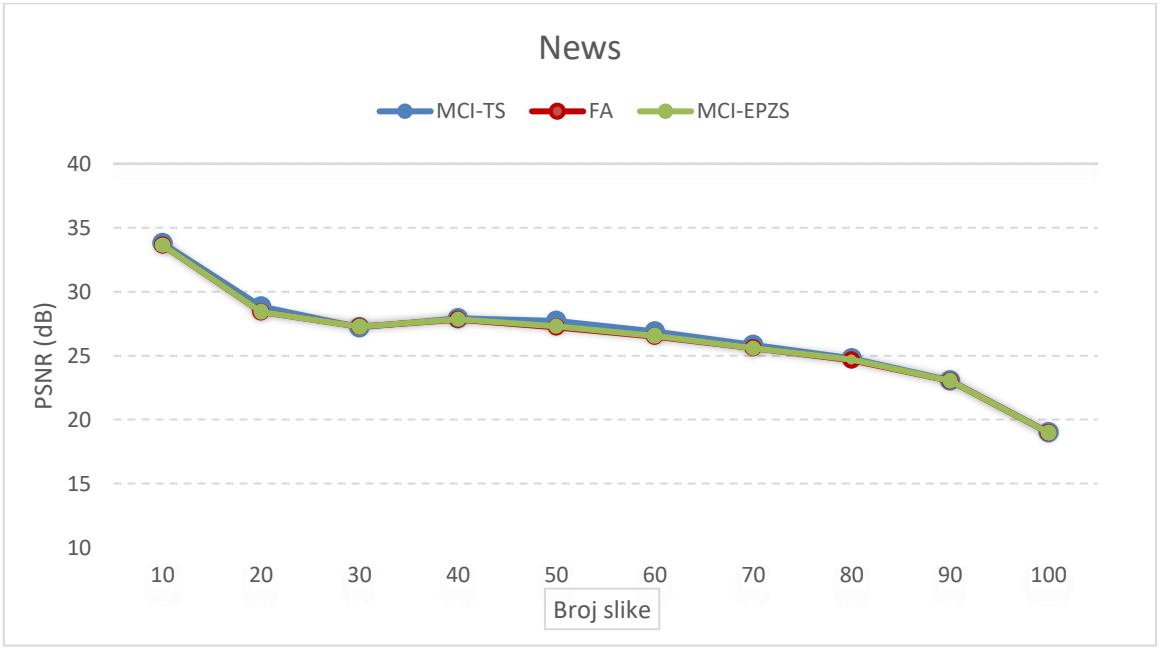
Kvaliteta interpolacije primijenjenom metodom uspoređena je s kvalitetom interpolacije s druge dvije metode. Metode korištene za interpolaciju su: metoda iz rada (opisana u teorijskom dijelu, dalje u tekstu MCI-TS), metoda usrednjavanjem (*eng. Frame Averaging – FA)*, metoda vremenske interpolacije primjenom drugog algoritma (u rezultatima i daljnjem radu predstavljena kraticom MCI-EPZS). Metoda MCI-EPZS ima iduće karakteristike: bilateralna procjena pokreta, algoritam pretrage pokreta je *Enhanced Predictive Zonal Search Algorithm* , te je kompenzacija pokreta tipa OBMC (*eng. Overlapped Block Motion Compensation*). Metoda usrednjavanja koristi srednje vrijednosti između dvije slike i pomoću njih generira novu sliku.

Konverzija vremenske rezolucije primjenom FA i MCI-EPZS metoda i proračun PSNR izvedeni su pomoću alata FFmpeg (u prilogu se nalaze kôdovi korišteni za izvođenje zadatka u FFmpeg-u). Kvaliteta sekvenci s interpoliranim slikama je izmjerena PSNR metrikom, a dobiveni rezultati su predstavljeni u tablicama (tablice 4.1.,4.2.,4.3. i 4.4.) i grafičkim prikazom (slike 4.1.,4.2.,4.3. i 4.4.), za svaku video sekvencu zasebno. U tablicama su dane PSNR vrijednosti za prvih sto slika u video sekvenci, i to kao srednje vrijednosti za deset po deset uzastopnih slika. U tablici 4.5. dane su prosječne PSNR vrijednosti (prosjek po svim slikama u pojedinoj sekvenci) za sve tri metode, za sve sekvence.

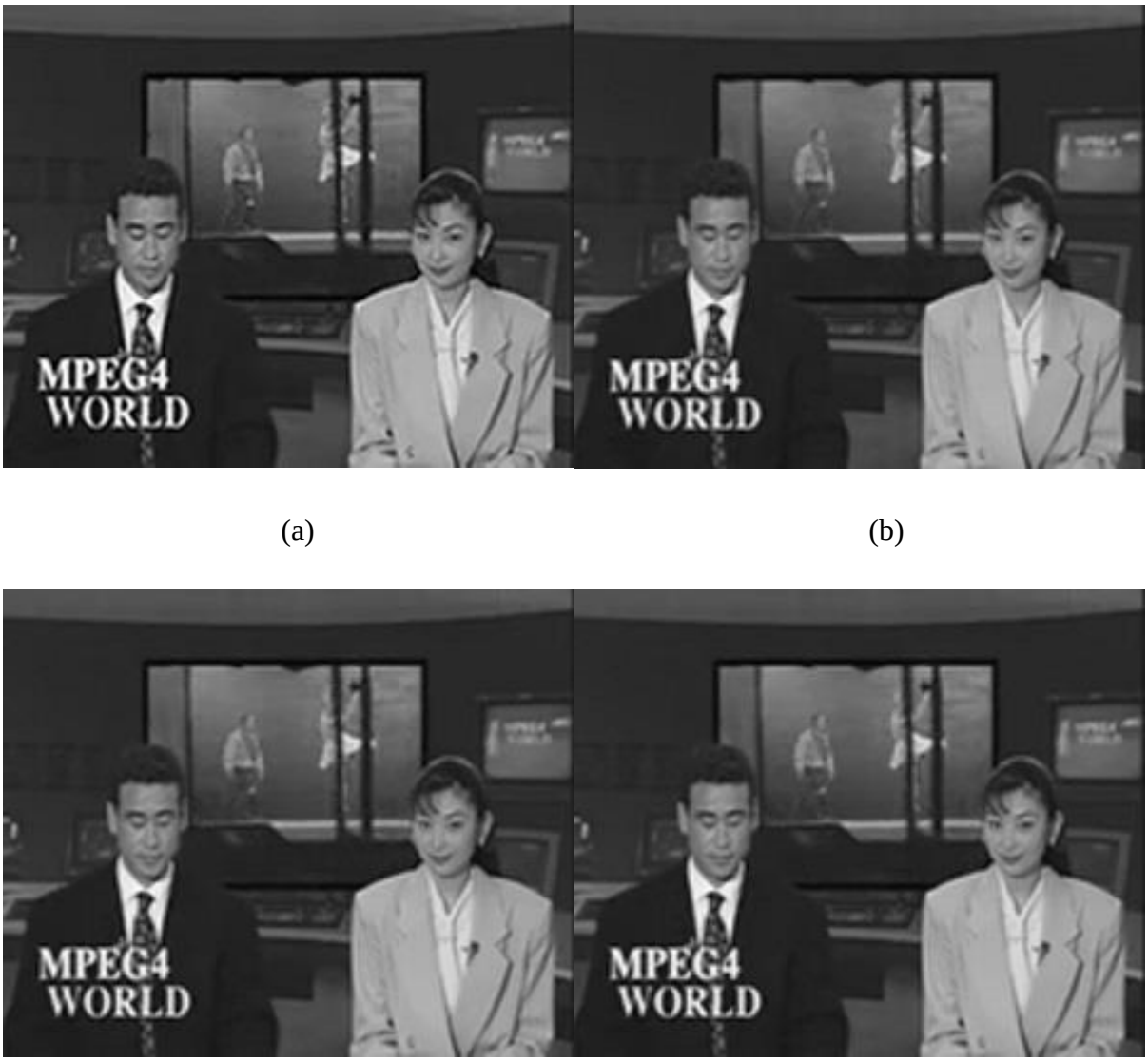
U tablici 4.1 i na slici 4.1 dani su rezultati za sekvencu *News.* Sekvenca *News* sastoji se od 300 slika, scena je statična i bez brzih pokreta s dva objekta koja se polako pomiču. U sredini scene pojavljuje se prozor sa statičnom scenom i dva objekta brzih pokreta, a iza svih objekata je pozadina relativno visokog kontrasta. Kao što se vidi iz podataka (tablica 4.1.) za sve tri metode interpolirane sekvence imaju slične PSNR vrijednosti, s tim da je metoda MCI-TS postigla nešto veće vrijednosti nego druge dvije metode.

Tab. 4.1. PSNR vrijednosti iskazane u decibelima(dB) za sekvencu *News*

metoda\br. slika	10	20	30	40	50	60	70	80	90	100
MCI-TS	33.8	28.84	27.23	27.92	27.72	26.89	25.84	24.79	23.07	19
FA	33.63	28.41	27.28	27.83	27.23	26.5	25.6	24.65	23.03	18.98
MCI-EPZS	33.63	28.41	27.28	27.84	27.3	26.53	25.6	24.69	23.04	18.98



Sl. 4.1. Grafički prikaz PSNR vrijednosti za sekvencu *News*



(a) (b)

(c) (d)

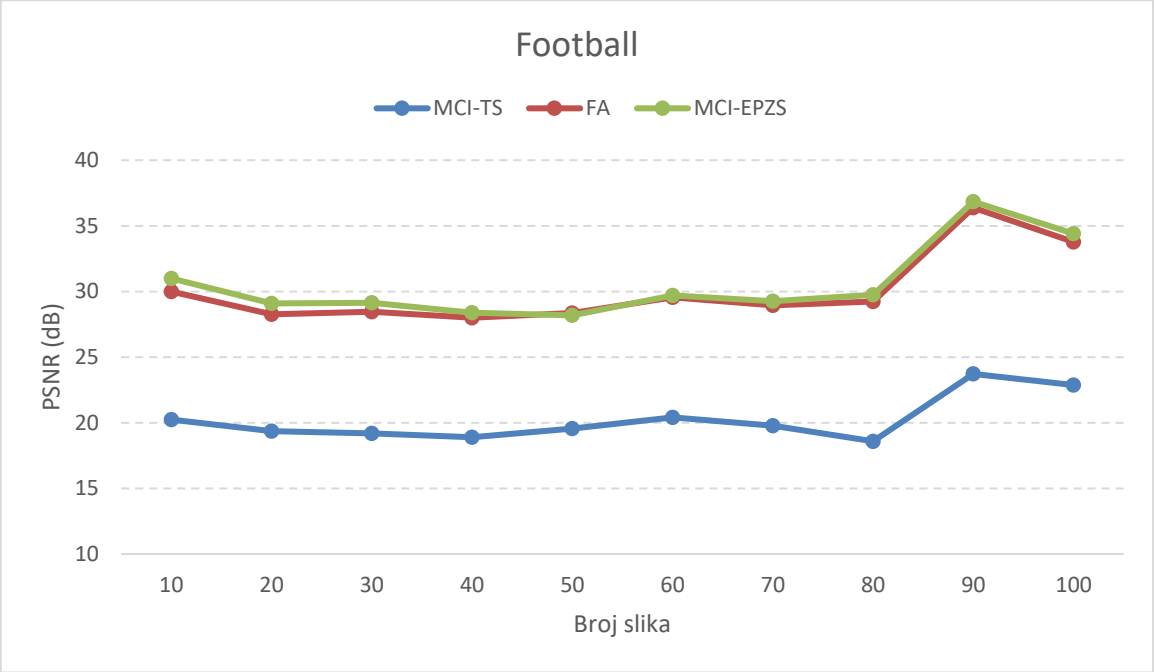
Sl. 4.2. Originalni i interpolirani pedeseti okvir sekvence *News*: (a) originalni okvir, (b) metoda iz rada MCI-TS, (c) metoda FA, (d) metoda MCI-EPZS

Subjektivno gledajući interpolaciju sekvence *News* (slika 4.2.) može se zaključiti da nema velike razlike u kvaliteti između interpoliranih slika, bez obzira na metodu interpolacije. Bitno je napomenuti kako PSNR nije najidealnija metoda komparacije kvalitete, iz razloga što je PSNR omjer jačine signala i jačine šuma i vrijednost može ukazivati da je razlika velika, dok subjektivno razlika može biti mala ili neprimjetna. U ovom se slučaju rezultati PSNR-a poklapaju sa subjektivnim dojmom dobivenih rezultata.

U tablici 4.2. i na slici 4.3. dani su rezultati za sekvencu *Football*.

Tab. 4.2. PSNR vrijednosti iskazane u decibelima(dB) za sekvencu *Football*

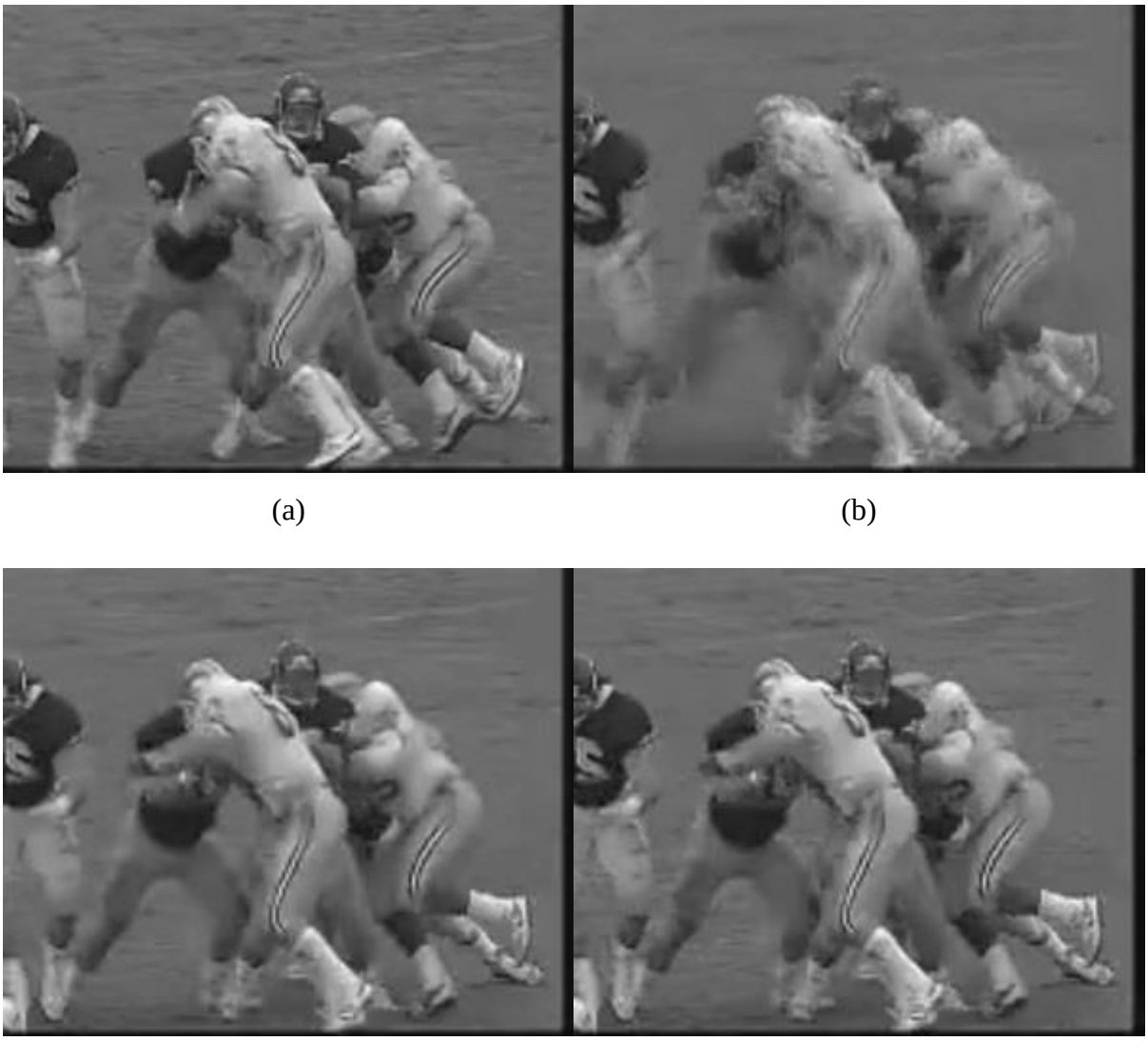
metoda\br. slika	10	20	30	40	50	60	70	80	90	100
MCI-TS	20.24	19.36	19.2	18.9	19.56	20.41	19.78	18.59	23.72	22.87
FA	30	28.25	28.45	28	28.35	29.55	28.94	29.24	36.38	33.76
MCI-EPZS	31	29.1	29.13	28.38	28.18	29.71	29.25	29.75	36.84	34.4



Sl. 4.3. Grafički prikaz PSNR vrijednosti za sekvencu *Football*

Sekvenca *Football* sastoji se od 300 slika, sekvenca sadrži više objekata s različtim brzinama kretnji, cjelokupna scena je dinamična i cijelo vrijeme se kamera pomiče. Kontrast između objekata i pozadine je relativno visok u većini scene. Ovakve scene su dosta zahtjevne za interpolaciju jer sadrže puno objekata koji se brzo kreću, scena se pomjera i rubovi objekta se često poklope pa kontrast skoro da i ne postoji. Metoda kompenzacije pokreta MCI-EPZS je dala najbolje PSNR vrijednosti što se vidi iz tablice 4.2., dok je metoda MCI-TS ostvarila između 9 i 13 dB manje vrijednosti PSNR u odnosu na MCI-EPZS.

Na slici 4.4. prikazan je primjer interpolirana slike sa sve tri primijenjene metode. Vidljivo je da metoda MCI-TS ostvaruje sliku lošije kvalitete te je očita pojava artefakta na samoj slici kao i pojava određene razine zamućenja pokreta. Razlog tome je što kôd korišten za interpolaciju MCI-TS nema dodatnu obradu slike nakon interpolacije, niti filtere za uklanjanje mutnih djelova, dok *ffmpeg* alat koristi razne tehnike poboljšanja kvalitete slike nakon same interpolacije.



(a) (b)

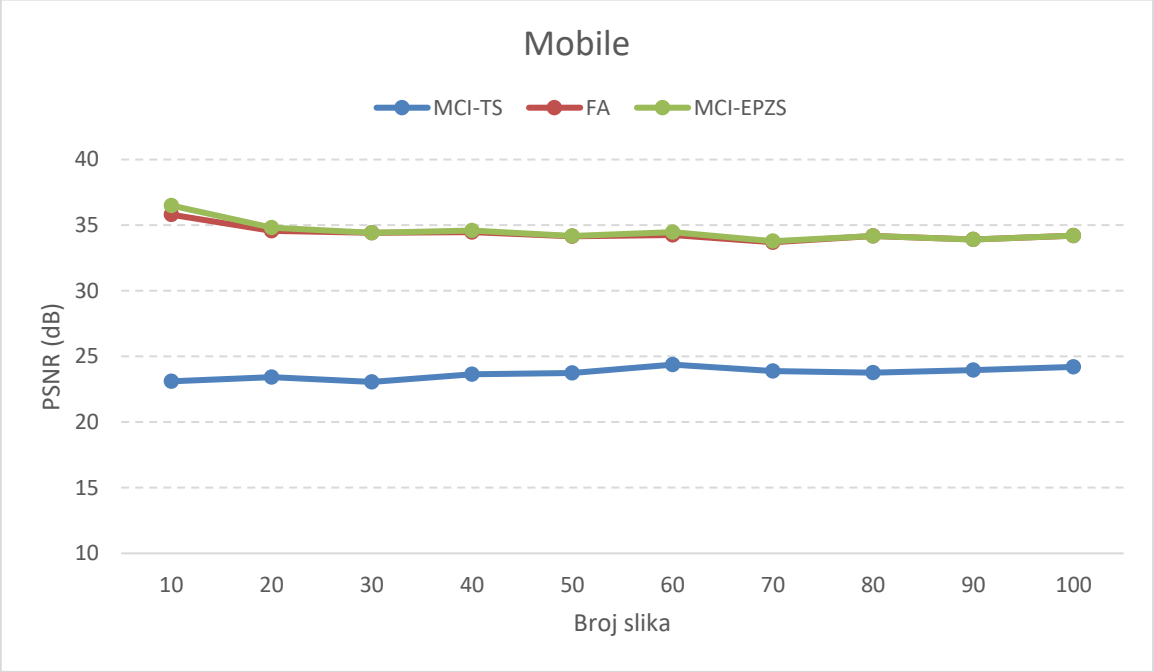
(c) (d)

Sl. 4.4. Originalni i interpolirani pedeseti okvir sekvence *Football*: (a) originalni okvir, (b) metoda iz rada MCI-TS, (c) metoda FA, (d) metoda MCI-EPZS

U tablici 4.3. i na slici 4.5. dani su rezultati za sekvencu *Mobile:*

Tab. 4.3. PSNR vrijednosti iskazane u decibelima(dB) za sekvencu *Mobile*

metoda\br. slika	10	20	30	40	50	60	70	80	90	100
MCI-TS	23.11	23.42	23.06	23.65	23.75	24.38	23.89	23.77	23.97	24.2
FA	35.8	34.56	34.41	34.48	34.16	34.25	33.7	34.19	33.91	34.21
MCI-EPZS	36.5	34.82	34.42	34.6	34.18	34.46	33.78	34.17	33.91	34.21

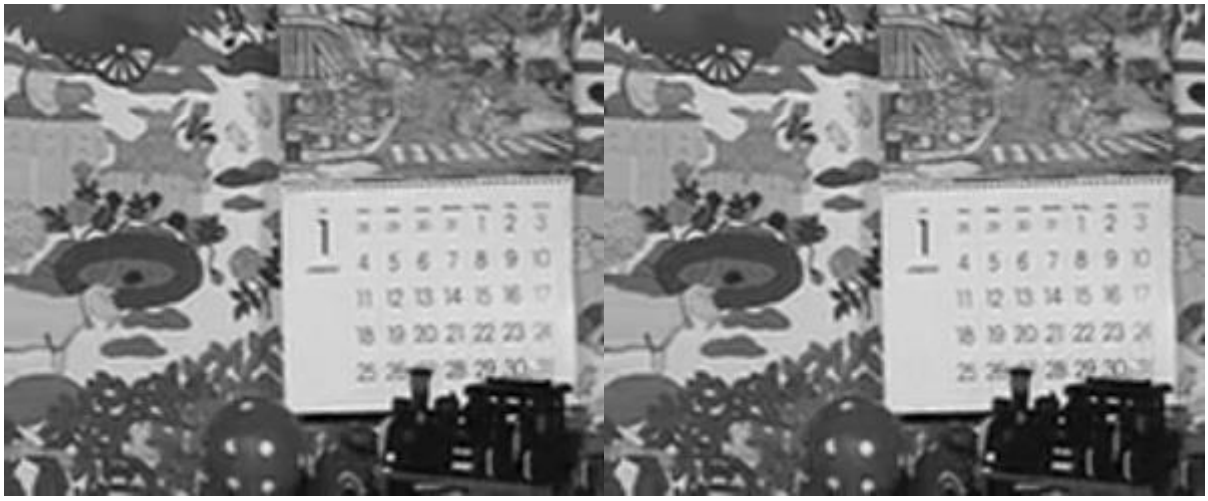


Sl. 4.5. Grafički prikaz PSNR vrijednosti za sekvencu *Mobile*

Sekvenca *Mobile* sadrži 300 slika, cjelokupna sekvenca sadrži spore pokrete, s četiri objekta koji se pomiču. U sekvenci ima dosta detalja i rubova, različitih kontrasta od kojih se neki preklapaju i slični su. Scena također sadrži sitne detalje koji u nekim dijelovima sekvence ili budu potpuno prekriveni ili izlaze iz scene. PSNR vrijednosti (tablica 4.3.) najbolje su kod MCI-EPZS metode, jako slične vrijednosti imala je i FA metoda, dok metoda iz rada MCI-TS ima slabiji rezultat, što je također vidljivo u grafičkom prikazu (slika 4.5.)

Iako metoda MCI-TS ima znatno manje PSNR vrijednosti, subjektivnim pregledom ne primjećuje se velika razlika (slika 4.6.). Subjektivno sve tri interpolacije ne odstupaju kvalitetom od originala. Vidljivo na slikama je da nema zamućenja pokreta, niti stvaranja artefakta na slici. Razlog tomu je dinamčki spora scena, iako ima dosta detalja i različitih kontrasta ipak kretnje objekata u sceni su usporene.

(a) (b)



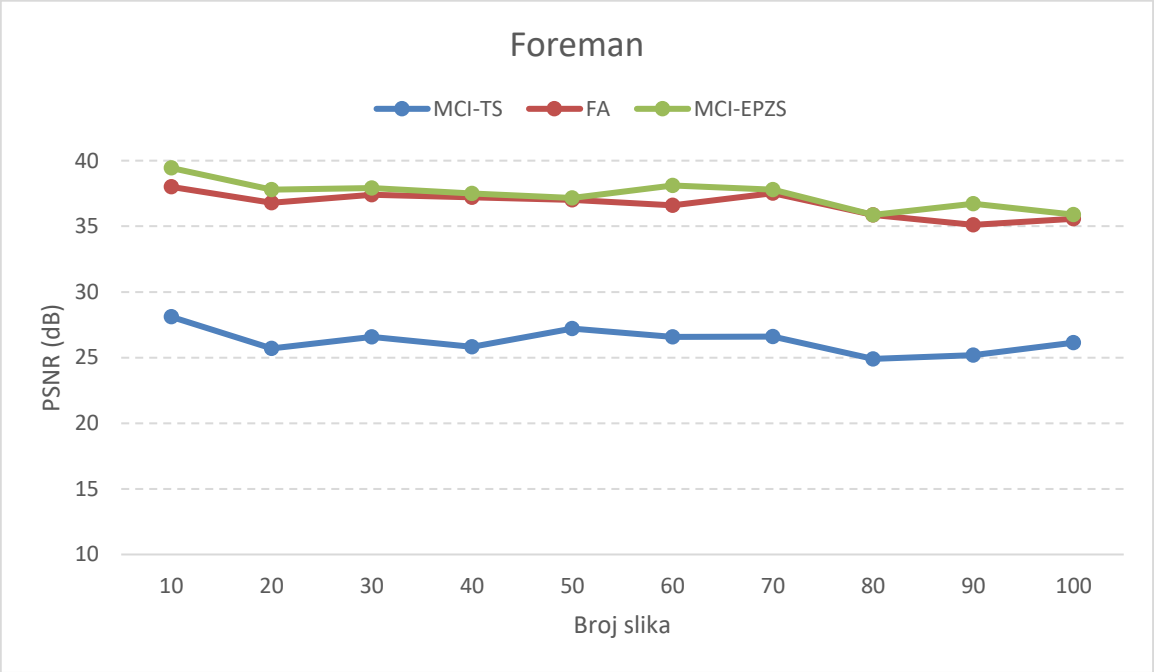
(c) (d)

Sl. 4.6. Originalni i interpolirani pedeseti okvir sekvence *Mobile*: (a) originalni okvir, (b) metoda iz rada MCI-TS, (c) metoda FA, (d) metoda MCI-EPZS

U tablici 4.4. i na slici 4.7. dani su rezultati za sekvencu *Foreman:*

Tab. 4.4. PSNR vrijednosti iskazane u decibelima(dB) za sekvencu *Foreman*

metoda/br. slika	10	20	30	40	50	60	70	80	90	100
MCI-TS	28.1	25.7	26.57	25.82	27.2	26.57	26.6	24.9	25.19	26.13
FA	38	36.79	37.41	37.21	37	36.59	37.53	35.86	35.11	35.58
MCI-EPZS	39.44	37.8	37.92	37.51	37.16	38.11	37.78	35.87	36.73	35.9



Sl. 4.7. Grafički prikaz PSNR vrijednosti za sekvencu *Foreman*

Sekvenca *Foreman* sadrži 300 slika, na sceni je jedan objekt koji se pomjera srednjom brzinom. Scena ima dinamičnu pozadinu veći dio videa, te se zatim pomjeranjem kamere cjelokupna scena pomjera nakon čega nemaniti jedan pomjerajući objekt, te sama scena postaje statična. U prvom dijelu kontrast je relativno mali, dok je u drugom dijelu sekvence znatno veći. PSNR vrijednosti (tablica 4.4.) su najbolje kod MCI-EPZS metode, slično kao i kod sekvence *Mobile* metoda FA dala je približne rezultate kao i najbolja metoda (slika 4.7.).

Iako podaci iz tablice 4.4. pokazuju veliku razliku u kvaliteti, subjektivna razlika u kvaliteti na slikama (slika 4.8.) je mala. S obzirom da scena ne sadrži previše brzih pokreta te je cijela dinamika scene sporija, a rubovi objekata su lako primjetni ne dolazi do stvaranja artefakta kao

kod scene *Foobtall*. Zamućenje pokreta je primjetno po rubovima objekta, no pozadina same scene je adekvatno interpolirana.

(a) (b)

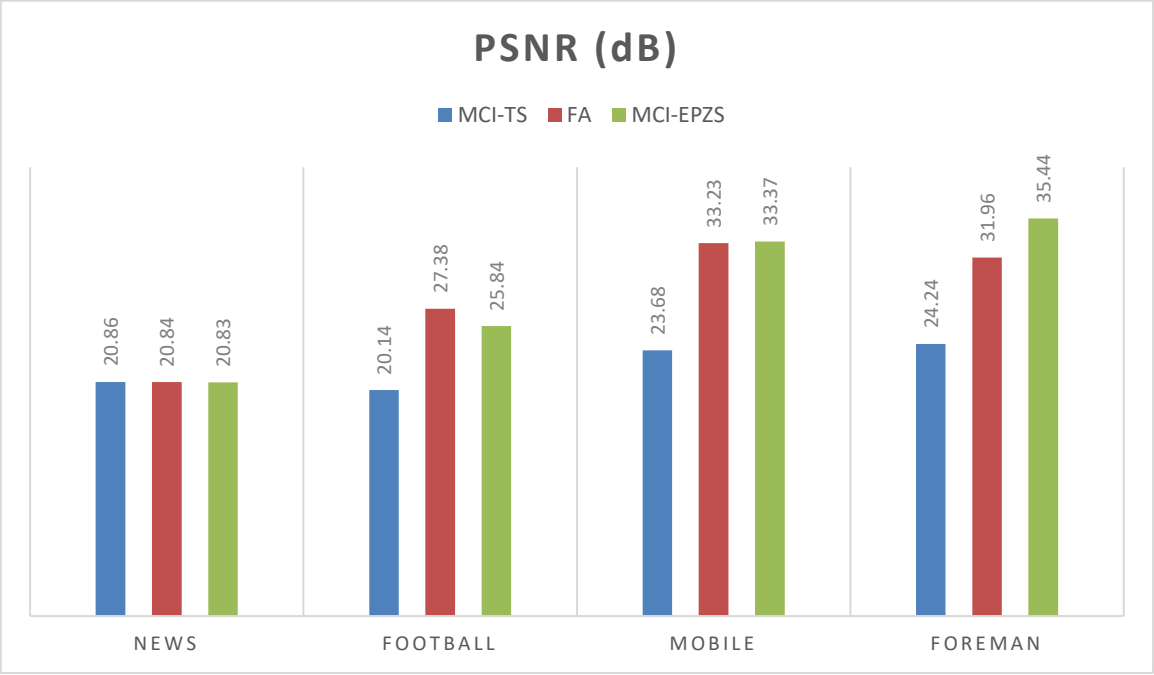
(c) (d)

Sl. 4.8. Originalni i interpolirani pedeseti okvir sekvence *Foreman*: (a) originalni okvir, (b) metoda iz rada MCI-TS, (c) metoda FA, (d) metoda MCI-EPZS

Prosječne vrijednosti PSNR za korištene metode u svim sekvencama dane su u tablici 4.5 i na slici 4.9. :

Tab. 4.5. Prosječne PSNR iskazane u decibelima(dB) vrijednosti za sve tri metode

metoda\sekv.	*News*	*Football*	*Mobile*	*Foreman*
MCI-TS	20.86	20.14	23.68	24.24
FA	20.84	27.38	33.23	31.96
MCI-EPZS	20.83	25.84	33.37	35.44



Sl. 4.9. Grafički prikaz prosječnih vrijednosti PSNR za sve tri metode

Ovisno od sekvence, odnosno što se na sekvenci nalazi, svaka od tri interpolacije se pokazala boljom ili lošijom u određenim okolnostima. Metoda iz rada se pokazala kao najbolja u sekvenci *News*, metoda usrednjavanjem u sekvenci *Football*, dok se interpolacija procjenom pokreta sa EPZS algoritmom pokazala najboljom u sekvencama *Mobile* i *Foreman*. U konačnici PSNR rezultati metoda FA i MCI-EPZS su dosta različite od MCI-TS metode iz razloga što su kod njih primijenjene razne tehnike poboljšavanja kvalitete slike nakon interpolacije.

5. ZAKLJUČAK

Interpolacija video okvira ima široku primjenu u video kodiranju u današnje vrijeme kad je bitno da se sadržaj visoke kvalitete prenese kroz mrežu u stvarnom vremenu. Razne tehnike se koriste i ne postoji jedinstvena metoda koja će odgovarati svakoj situaciji. Puno je parametara koje se uzima u obzir: kvaliteta videa, format videa, resursi koji se mogu izdvojiti za kompleksne računske operacije kodiranja i dekodiranja, a najveći je problem što točnije predviđanje pokreta u najkraćem mogućem vremenu.

Najčešće se koriste dvije vrste interpolacije. Interpolacija metodom usrednjavanja te interpolacija metodom procjene pokreta. Metoda procjene pokreta sastoji se iz dva ključna koraka, predikcija pokreta i kompenzacija pokreta, te je znatno računski zahtjevnija od metode usrednjavanja koja interpolira video okvir na osnovu srednjih vrijednosti piksela između slika. Kod interpolacije procjenom pokreta, predikcija pokreta znatno utječe hoće li interpolacija biti uspješna i koliko će vremena trebati za nju. Iako i unilateralno i bilateralno predviđanje pokreta može koristiti iste algoritme pretrage, bilateralna procjena pokreta ipak daje točnije rezultate pri samoj interpolaciji, s manje artefakta u interpoliranoj sekvenci.

U okviru praktičnog dijela napisan je program za interpolaciju primjenom bilateralne procjene pokreta. Rezultati interpolacije ovim programom uspoređeni su s rezultatima interpolacije primjenom programa FFmpeg i to s dvije različite metode interpolacije. Jedna je metoda usrednjavanja, a druga je metoda s bilateralnom procjenom pokreta, algoritam pretrage pokreta eng. *Enhanced Predictive Zonal Search Algorithm*, te je kompenzacija pokreta tipa OBMC (eng. *Overlapped Block Motion Compensation*). Kvaliteta interpoliranih slika je izmjerena PSNR metrikom, a dobivene vrijednosti pokazuju kako kvaliteta interpolacije značajno ovisi o sadržaju video sekvence za koju se radi interpolacija. No osim same interpolacije dobiveni rezultati također ukazuju da na kvalitetu interpoliranog videa ne utječe samo interpolacija i vrsta interpolacije nego i razne tehnike koje se primjenjuju nakon interpolacije slike. To su tehnike poput ublažavanja zamućenja pokreta, kao i tehnike koje uklanjaju artefakte koji se pojavljuju prilikom preklapanja blokova.

LITERATURA

[1] M. Vranješ, Digitalni video 1. dio, Predavanje s kolegija Multimedijska tehnika, FERIT Osijek, 2013

[2] B.D. Choi, K. Chang-Su, J.W. Han, S.J. Ko, Motion-Compensated Frame Interpolation Using

Bilateral Motion Estimation and Adaptive Overlapped Block Motion Compensation, IEEE Transaction on circuits and systems for video technology, vol. 17, no. 4, April, 2007

[3] B.K.V. Reddy, Fast Block Matching Algorithms for Video Compresion, Master's of Tehcnology in Electronics and Instrumentation, Department of Electronics and Communication Engineering, NIT Rourkela, 2013

[4] R. Han and A. Men, Frame Rate Up-Conversion for High-Definition Video Applications, IEEE Transactions on Consumer Electronics, br. 1, Vol. 59, str.:229-236, March 25, 2013

[5] F. Vestermark, Implementation of a frame rate up conversion filter, Master's Thesis in Computing Science, UMEA University Sweden, May 24, 2013

[6] A. Barjatya, Block Matching Algorithms For Motion Estimation, ECE dept at Utah State University, April 26, 2004

[7] S.J. Kang, K.R. Cho, Y.H. Kim, Motion Compensated Frame Rate Up-Conversion Using Extended Bilateral Motion Estimation, IEEE Transactions on Consumer Electronics, br. 4, vol. 53, str.:1759-1767, November 15, 2007

[8] T. Miyoshi, Three Step Search, ResearchGate, 2012., dostupno na: http://www.researchgate.net/figure/An-example-of-three-step-search-algorithm_fig17_258384327 (28.08.2019.god.)

[9] A. Babcock, Chroma Subsampling, Rtings.com, 2019., dostupno na: http://www.rtings.com/tv/learn/chroma-subsampling (29.08.2019.god.)

SAŽETAK

Tema završnog rada je interpolacija slika primjenom bilateralne procjene pokreta u postupku povećanja brzine izmjene slika video sekvenci. U radu su opisane metode interpolacije primjenom estimacije i kompenzacije pokreta te načini pretrage u postupku procjene vektora pokreta. Objašnjene su razlike između bilateralne i unilateralne interpolacije. U praktičnom dijelu zadatka napravljen je kod u programskom paketu Matlab za interpolaciju slika primjenom bilateralne procjene pokreta. Kvaliteta slika interpoliranih implementiranom metodom izmjerena je PSNR metrikom za 5 sekvenci različitog sadržaja. Ti su rezultati uspoređeni s rezultatima interpolacije napravljene programom ffmpeg i to metodom usrednjavanja i metodom MCI-EPZS. Pokazano je da uspješnost metoda interpolacije ovisi o sadržajima video sekvenci, ali i o metodama poboljšanja kvalitete videa koje se primjenjuju nakon interpolacije.

Ključne riječi: interpolacija video okvira, procjena pokreta, kompenzacija pokreta, bilateralna estimacija, povećanje brzine izmjene slika, ffmpeg

ABSTRACT

Video Frame Interpolation Based on Bilateral Motion Estimation

The topic of this final paper is video frame interpolation based on bilateral motion estimation in the process of frame rate upscaling. This paper describes methods of motion interpolation by using motion estimation and motion compensation, also various algorithms that are used for adequate motion detection and obtaining motion vectors. It explains the key differences between bilateral and unilateral interpolation. As a practical part of this final thesis, a code for frame interpolation with bilateral motion estimation was made in Matlab. The quality of the interpolated pictures that are obtained by the implemented method is measured with PSNR metrics on 5 video sequences with different content. These results are compared to the results of interpolation made with ffmpeg software by using two methods, frame averaging and MCI-EPZS. The results show that successful motion interpolation depends on the content of the video sequences, but also on the methods for image improvement that are applied after the interpolation itself.

Key words: video frame interpolation, motion estimation, motion compensation, bilateral estimation, frame rate up conversion, ffmpeg

PRILOG

Prilog sadrži kodove za dvije različite aplikacije, prva aplikacija je MATLAB, i u njoj je napisan programski kod za vremensku interpolaciju video okvira primjenom bilateralne procjene pokreta. Programski kod koristi *„Three step search“* algoritam za detekciju pokreta. Druga aplikacija je FFmpeg i korištena je u svrhu izvršenja zadatka rada, učitavanje video sekvenci, interpolacija usrednjavanjem i interpolacija kompenzacijom pokreta, te za računanje PSNR vrijednosti.

MATLAB:

```
clc
clear all
close all
tic
% CREATE VIDEO READER AND WRITER OBJECTS AND SEPCIFY FILENAMES
    filename = 'foreman_lowfps.mp4';                    % INPUT FILE Name
    newfilename   =   strcat('interp_',filename);       %   OUTPUT   FILE   Name
(INTERPOLATED)

    vidReader = VideoReader(filename);      % INPUT FILE Reader
    vidWriter = VideoWriter(newfilename,'MPEG-4');       % OUTPUT FILE Writer
(INTERPOLATED)
    vidWriter.FrameRate = vidReader.FrameRate*2;     %Upconverting the frame
rate of the output file
    open(vidWriter);
% USER DEFINED PARAMETERS
    r = 35;         % Block Size Rows
    c = 35;         % Block Size Columns

    maxmov = 25;     % Maximum motion possible in consecutive frames (in terms
of pixels)

    mysearchmethod = 'Three-step';  % The other option is 'Exhaustive' which is
more accurate but more time consuming
    %mysearchmethod = 'Exhaustive';  % The other option is 'Three-step' which
is less accurate but also less time consuming

% CREATE VISION OBJECT (BLOCK MATCHER) with User Defined paramters
    hbm     = vision.BlockMatcher('ReferenceFrameSource',...
              'Input                                        port','BlockSize',[r
c],'SearchMethod',mysearchmethod,'MaximumDisplacement',[maxmov maxmov]);
    hbm.OutputValue = 'Horizontal and vertical components in complex form';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        aForward = 0.5; %Forward compensation weight
        aBackward = 0.5;    %Backward compensation weight
```

```
ctr=0; % intialize ctr. This variable will be used to mark the second frame (
because we can start our interpolation from second frame onwards )
while hasFrame(vidReader)   % For Each Frame of the Input Video
    ctr=ctr+1;
    nextframe = readFrame(vidReader);
    %nextframe = nextframe(200:500,300:1000,:);   % Crop the image for better
view ( you may skip this)

    if ctr>2        % Second Frame onwards
        % STEP 1: Compute motion vector between the two images.
        img1=rgb2gray(prevframe);
        img2=rgb2gray(nextframe);

        % motion = hbm(img1,img2);  % This is the motion vector in complex form
(real part is motion along x-axis and imaginary part is motion along y-axis)
        motionForward = step(hbm,img1,img2);   % This is the forward motion
vector in complex form (real part is motion along x-axis and imaginary part is
motion along y-axis)
        motionBackward = step(hbm,img2,img1);   % This is the backward motion
vector in complex form (real part is motion along x-axis and imaginary part is
motion along y-axis)

        % STEP 2: Filter out slower motions

        %% Filtering forward motion vectors
        threshForward = median(real(motionForward(:)));   % Slower motions
threshold are the median of motion vector
        idxForward = find(real(motionForward)<=threshForward);
        motionForward(idxForward)              =              0              +
double((imag(motionForward(idxForward))))*i ;
        threshForward = median(imag(motionForward(:)));
        idxForward = find(imag(motionForward)<=threshForward);
        motionForward(idxForward) = double((real(motionForward(idxForward))))
+ 0*i;

        %% Filtering backward motion vectors
        threshBackward = median(real(motionBackward(:)));   % Slower motions
threshold are the median of motion vector
        idxBackward = find(real(motionBackward)<=threshBackward);
        motionBackward(idxBackward)              =              0              +
double((imag(motionBackward(idxBackward))))*i ;
        threshBackward = median(imag(motionBackward(:)));
        idxBackward = find(imag(motionBackward)<=threshBackward);
        motionBackward(idxBackward)                                           =
double((real(motionBackward(idxBackward)))) + 0*i;

        % STEP 3: Copy Motion value to Each Pixel of the Block
        totForward = zeros(size(img1));             % Initialize as all zero
motions. This variable will contain the motion
        totBackward = zeros(size(img1));            % Initialize as all zero
motions. This variable will contain the motion
        for lp1 = 1:size(motionForward,1)  % rows
            for lp2 = 1:size(motionForward,2) % cols
                blkForward = repmat(motionForward(lp1,lp2),[r c]);
                blkBackward = repmat(motionBackward(lp1,lp2),[r c]);
                sr=(lp1-1)*r+1;
                sc=(lp2-1)*c+1;
                totForward(sr:sr+r-1,sc:sc+c-1)=blkForward;
                totBackward(sr:sr+r-1,sc:sc+c-1)=blkBackward;
            end
```

```
        end

        % STEP 4: Make the Interpolted Frame, basing on motion vectors.
        xmotionForward = real(totForward);     % Forward motion along x-axis
(rows in image)
        ymotionForward = imag(totForward);     % Forward motion along y-axis
(columns in image)

        xmotionBackward = real(totBackward);    % Backward motion along x-axis
(rows in image)
        ymotionBackward = imag(totBackward);    % Backward motion along y-axis
(columns in image)

        ctr2 = 0;  % to record the old/new values of pixel coordinates.
        interpI =uint8(zeros(size(img1)));  % Interpolated Image
        data=zeros(1,4);
        for lp1 = 1:size(img1,1)      % rows and ymotion
            for lp2 = 1:size(img1,2) % cols and x motion
                newyForward = lp1 + round(ymotionForward(lp1,lp2) /2); % new y-
coordinate value for forward interpolation (row number)
                newxForward = lp2 + round(xmotionForward(lp1,lp2) /2); % new x-
coordinate value for forward interpolation (col number)     % move/relocate by
half the motion between the two frames
                if newxForward > size(img1,2) newxForward = size(img1,2);end  %
check to not to exceed image size for new x-coordinate value (col number)
                if newyForward > size(img1,1) newyForward = size(img1,1);end  %
check to not to exceed image size for new y-coordinate value (row number)
                if newxForward < 1             newxForward = 1            ;end
% check to not to exceed image size for new x-coordinate value (col number)
                if newyForward < 1             newyForward = 1            ;end
% check to not to exceed image size for new y-coordinate value (row number)

                newyBackward = lp1 + round(ymotionBackward(lp1,lp2) /2); % new
y-coordinate value for backward interpolation (row number)
                newxBackward = lp2 + round(xmotionBackward(lp1,lp2) /2); % new
x-coordinate value for backward interpolation (col number)     % move/relocate
by half the motion between the two frames
                if newxBackward > size(img2,2) newxBackward = size(img2,2);end
% check to not to exceed image size for new x-coordinate value (col number)
                if newyBackward > size(img2,1) newyBackward = size(img2,1);end
% check to not to exceed image size for new y-coordinate value (row number)
                if newxBackward < 1             newxBackward = 1            ;end
% check to not to exceed image size for new x-coordinate value (col number)
                if newyBackward < 1             newyBackward = 1            ;end
% check to not to exceed image size for new y-coordinate value (row number)

                interpI(lp1,lp2) = aForward*img2(newyForward,newxForward)...
                    +   aBackward*img1(newyBackward,   newxBackward);          %
Interpolated Image has values of image2 at new Coordinates location
                % Interpolated image has a linear combination of weighted
                % intensity values of img1 (previous frame) and img2 (next
                % frame) at the respective new coordinates

                ctr2 = ctr2 + 1;
                %data(ctr2,:) = [lp1 lp2 newy newx];
            end
        end

        interpI = mat2gray(interpI);
```

```
        % Write Video Frames in File
            writeVideo(vidWriter,img1);
            writeVideo(vidWriter,interpI);

            %disp(['Iteration ' num2str(ctr) ' complete']);
    end
    prevframe = nextframe;
end

writeVideo(vidWriter, img2);
pause(1)
close(vidWriter);

toc/60
```

Potrebno je kopirati kod u Matlab, te spremiti dokument (*eng. File*) u mapu u kojoj se nalazi video na kojem želimo izvršiti interpolaciju. Nakon toga u šestoj liniji koda treba promijeniti ime ulaznog videa da bude točno kao ime videa u mapi.

Kod je napisan pomoću mnogih instrukcijskih prijedložaka sa raznih stranica, uključujući najviše službenu stranicu Matlab-a. S obzirom da niti jedan tutorijal nije kopiran u sam program, nisu navedene reference tih stranica, jer su služile samo za učenje. Jedina web lokacija s koje je kopiran dio koda je Matlab stranica za Vision Tool Box:

https://www.mathworks.com/help/vision/referencelist.html?type=function&s_tid=CRUX_gn_function

Drugi korišteni alat je FFmpeg, multimedijalni *framework* koji je ima mogućnosti obrađivati veliki broj audio i video formata. FFmpeg ima na raspolaganje niz mogućnosti obrade audio video materijala. U samom radu od FFmpeg-a je korišten samo manji broj komandnih linija u cmd-u (navodni znaci se ne pišu samo kod imena ulazne i izlazne video sekvence):

-konverzija datoteke:

ffmpeg -i „input_videoname.format“ „output_videoname.format“

-spuštanje *frame rate-a* sa 25 na 12.5 :

ffmpeg -i „input_videoname.format“ -filter:v fps=fps=12.5 „output_videoname.format“

-interpolacija video sekvence primjenom interpolacije usrednjavanjem, vraćanje na 25 slika u sekundi:

ffmpeg -i „input_videoname.format“ -filter „minterpolate='mi_mode=blend:fps=25'“ „output_videoname.format“

-interpolacija video sekvence primjenom interpolacije procjenom pokreta:

ffmpeg -i „input_videoname.format“ -filter „minterpolate='mi_mode=mci:fps=25'“ „output_videoname.format“

(opcija „mci“ još otključava opcije za odabir algoritma pretrage, vrstu kompenzacije pokretom, međutim te opcije nisu upisane u liniju koda jer defaultni parametri su točno oni koje smo trebali u zadatku)

-PSNR evaluacija između dva videa (u ovom radu između originalne sekvence i interpolirane), te zapis u eksternu txt datoteku

ffmpeg -i „input_1st_videoname.format“ -i „input_2nd_video.format“ -lavfi psnr=“stat_file=psnr.log“ „output_videoname.format“