Algoritmi za povećanje prostorne rezolucije video signala

Vukoje, Danijel

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:804021

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

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

Sveučilišni diplomski studij

ALGORITMI ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA

Diplomski rad

Danijel Vukoje

Osijek, 2016.

SADRŽAJ

1. UVOD 1

2. ALGORITMI ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA 3

2.1. Metoda najbližeg susjeda 4

2.2. Bilinearna interpolacija 5

2.3. Bikubična interpolacija 6

2.4. Triangulacijska (trokutna) interpolacija ovisna o podacima 8

2.5. Nova rubom-upravljana interpolacija 10

2.6. Iterativna interpolacija zasnovana na zakrivljenosti 10

2.7. Prostorno-vremenska interpolacija zasnovana na samo-sličnosti 12

2.8. Interpolacija super-razlučivosti zasnovana na primjeru 12

3. VLASTITO RJEŠENJE ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA 14

3.1. Prostorna aktivnost video signala 14

3.2. Vremenska aktivnost videosignala 14

3.3. Opis vlastitog rješenja za povećanje prostorne rezolucije video signala 15

3.4. Upute za pokretanje vlastitog rješenja za povećanje prostorne rezolucije video signala 17

4. VREDNOVANJE REZULTATA PREDLOŽENOG RJEŠENJA ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA 20

4.1. Korištene baze video signala 20

4.1.1. EVVQ baza podataka 20

4.1.2. LIVE baza podataka 21

4.2. Rezultati testiranja algoritma za povećanje prostorne rezolucije video signala 23

5. ZAKLJUČAK 30

LITERATURA 32

SAŽETAK 34

ŽIVOTOPIS 35

PRILOZI 36

P 3.1. Kod aktivnost_videa.m skripte 36

P 3.2. Kod prostorno_povecanje_rezolucije_videa.m skripte 39

P 4.1. Kod prostorno_smanjenje_rezolucije_videa.m skripte 48

P 7.1. DVD – ROM 52

1. UVOD

Tijekom posljednjih nekoliko desetljeća digitalna tehnologija se naglo razvija, što omogućava korištenje i produkciju video sadržaja na dnevnoj razini. Tehnologije za obradu slike i videa igraju glavnu ulogu u obradi vizualnih podataka, kako bi vizualna percepcija korisnika bila zadovoljavajuća, odnosno, na što višoj razini. U eri interneta i multimedijskih komunikacija promjena prostorne rezolucije video signala (engl. *spatial video scaling*) postala je vrlo bitna kako bi se on prilagodio za gledanje, prijenos, skidanje, dijeljenje, uređivanje i daljnju obradu. Znatnim povećanjem rezolucije digitalnih senzora povećala se i rezolucija slike, odnosno videa. Kako bi se video visoke rezolucije prikazivao, potrebni su zasloni visoke razlučivosti (engl. *HDTV - High Definition Television*) koji su postali minimum zahtjeva današnjih korisnika. Trenutni vrhunac tehnologije su video signali ultra visoke razlučivosti (engl. *UHD - Ultra High Definition*).

Međutim, nisu svi komercijalni video sadržaji dostupni u visokoj rezoluciji. Tu je i velika količina video sadržaja niske razlučivosti koji su snimljeni uređajima s niskom rezolucijom senzora, kao što su nadzorne, medicinske, web i video kamere (stari filmovi). Mnogi korisnici žele gledati taj sadržaj preko cijelog zaslona sa svojim uređajima za prikaz u visokoj razlučivosti, gdje do važnosti dolazi povećanje prostorne rezolucije video signala (engl. *spatial video upscaling*). Za povećanje prostorne rezolucije video signala koriste se različiti načini interpolacije slike. U literaturi se mogu pronaći pod raznim nazivima kao što su: promjena veličine slike, ponovno uzorkovanje slike, digitalno zumiranje, povećanje slike, poboljšanje slike (engl. *image resizing, image re-sampling, digital zooming, image magnification, image enhancement*) i drugi. Osnovni zadatak interpolacijskog algoritma je promjena rezolucije (dimenzija) slike iz manje u veću ili obrnuto, sa što manjim gubitkom vizualnog sadržaja. Od algoritma se zahtjeva predviđanje vrijednosti praznih točaka tj. elemenata slike (engl. *pixel*) koje nastaju povećanjem dimenzija slike, na osnovu manjeg broja ulaznih vrijednosti. Vrednovanje algoritama se najčešće vrši objektivnim mjerama kvalitete slike koje koriste različite matematičke algoritme za procjenu kvalitete slike na temelju određenih statističkih značajki te usporedbom kompleksnosti i računalne cijene algoritma. Kod prijenosnih uređaja kao što su pametni telefoni i tablet računala, učinkoviti interpolacijski algoritmi s niskom kompleksnošću moraju uzeti u obzir potrošnju baterije. Previše kompleksni izračuni mogu ozbiljno smanjiti trajanje baterije.

Neki od postojećih interpolacijskih algoritama prezentirani su u drugom poglavlju. U trećem poglavlju je predstavljen i objašnjen novi algoritam izrađen u sklopu ovog diplomskog rada te su dane upute za njegovo pokretanje. U četvrtom poglavlju testiran je algoritam te je proračunata

objektivna ocjena kvalitete rezultantne slike (videa) PSNR metrikom (engl. *Peak Signal to Noise Ratio*), SSIM (engl. *Structural Similarity Index*) i VQM (engl. *Video Quality Metric*) metrikom u odnosu na originalnu sliku (video). Rezultati su uspoređeni s rezultatima koje su postigli neki postojeći algoritmi.

2. ALGORITMI ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA

Interpolacijski algoritmi se mogu podijeliti u dvije kategorije, neadaptivni i adaptivni. Računalna logika neadaptivnih algoritama je fiksna te se ponavlja za svaki element slike ili skupinu lokalnih elemenata slike neovisno o ulaznim značajkama i sadržaju slike, dok se računalna logika adaptivnih algoritama prilagođava ulaznim značajkama i sadržaju slike kao što su vrijednosti intenziteta, informacije o rubovima na slici, tekstura itd. Neadaptivni algoritmi lakše se izvode i implementiraju te imaju manju računalnu složenost u odnosu na adaptivne algoritme. Iz tog razloga većina aplikacija koje se izvode u stvarnom vremenu i većina uređaja s digitalnim zaslonima koriste neadaptivne algoritme.

Neki od neadaptivnih algoritama su:

- metoda najbližeg susjeda [1]
- bilinearna interpolacija [2]
- bikubična interpolacija [2]
- *spline* interpolacija
- *sinc* interpolacija
- Lanczos interpolacija

Neki od adaptivnih algoritama su:

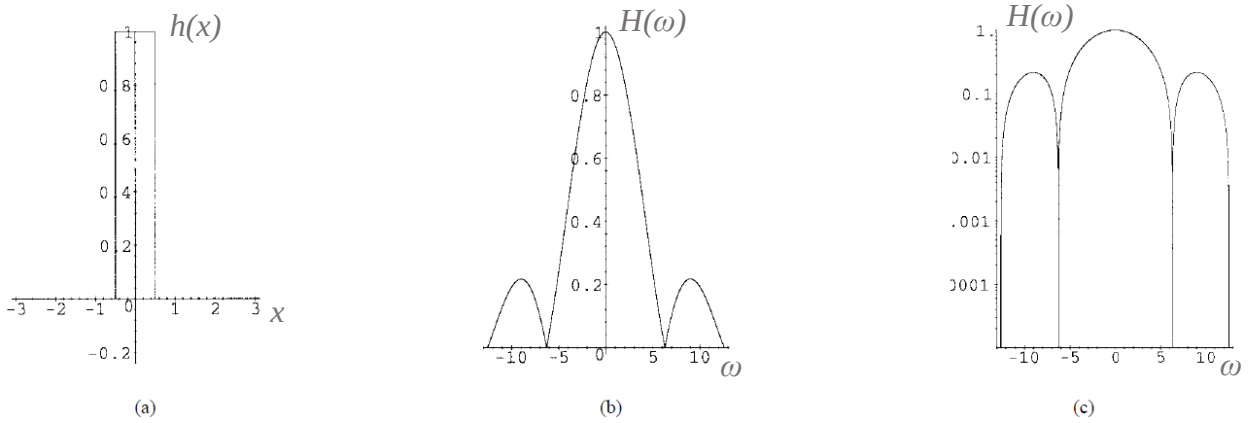
- Triangulacijska (trokutna) interpolacija ovisna o podacima (engl. *Data dependent triangulation interpolation - DDT*) [3]
- Nova rubom-upravljana interpolacija (engl. *New edge-directed interpolation - NEDI*) [4]
- Iterativna interpolacija zasnovana na zakrivljenosti (engl. *Iterative Curvature Based Interpolation - ICBI*) [5]
- Brza interpolacija zasnovana na zakrivljenosti (engl. *Fast Curvature Based Interpolation - FCBI*)
- Prostorno-vremenska interpolacija zasnovana na samo-sličnosti (engl. *Spatio-Temporal Self-Similarity*) [6]
- Interpolacija super-razlučivosti zasnovana na primjeru (engl. *Example-based super-resolution*) [7]

2.1. Metoda najbližeg susjeda

Ovo je najjednostavniji oblik interpolacije. Novi element slike određuje se na način da se za njegovu vrijednost uzme vrijednost najbližeg susjednog elementa slike (engl. *Nearest Neighbour*). Moguće je prvo vršiti horizontalnu pa vertikalnu interpolaciju, vertikalnu pa horizontalnu ili istovremeno vršiti interpolaciju u obje dimenzije. Zbog jednostavnog kopiranja elemenata slike interpolacija ima nisku računalnu cijenu te se još naziva i ponavljanje elemenata slike (engl. *pixel replication*) [1].U prostornoj domeni može se postići konvolucijom slike s interpolacijskom jezgrom. Interpolacijska jezgra prema [2] je:

$$h(x) = \begin{cases} 1, & 0 \leq |x| < 0.5 \\ 0, & 0.5 \leq |x| \end{cases}, \qquad (2\text{-}1)$$

gdje je *x* udaljenost između elementa slike kojeg je potrebno interpolirati i najbližeg originalnog elementa slike, a *h(x)* interpolacijska funkcija. Na slici 2.1. nalazi se frekvencijska analiza interpolacije.



Sl. 2.1. Metoda najbližeg susjeda: a) jezgra, b) Fourierova transformacija jezgre, c) logaritamski prikaz Fourierove transformacije [8]

Interpolacijska jezgra ima istaknute bočne komponente, a pojačanje u propusnom pojasu opada brzo, što je neželjeno svojstvo. Povećanje prostorne rezolucije slike pomoću metode najbližeg susjeda uzrokuje stepenast izgled objekata i rubova na slici. Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.2.



a) b)

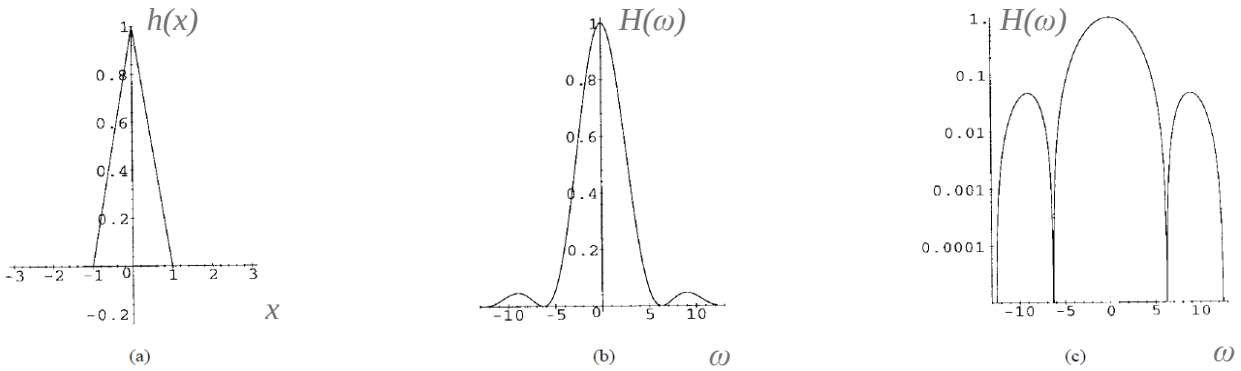
Sl. 2.2. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike metodom najbližeg susjeda: a) originalna slika, b) interpolirana slika [9]

2.2. Bilinearna interpolacija

Kao što ime govori, to je dvodimenzionalna linearna interpolacija; može se izvoditi u horizontalnom pa u vertikalnom smjeru ili obrnuto. Bilinearna interpolacija koristi težinski prosjek četiri okolna elementa slike za proračun interpoliranog elementa slike. Ima bolje performanse od metode najbližeg susjeda zbog smanjenja stepenastog efekta, što sliku vizualno čini glatkom. Međutim, pojavljuje se efekt zamućenja slike što je posljedica računanja prosjeka okolnih elemenata slike. Interpolacijska jezgra prema [2] je sljedeća:

$$h(x) = \begin{cases} 1 - |\mathrm{x}|, & 0 \leq |\mathrm{x}| < 1 \\ 0, & 1 \leq |x| \end{cases} \tag{2-2}$$

Na slici 2.3. nalazi se frekvencijska analiza interpolacije. Linearna interpolacijska jezgra ima bolje performanse od jezgre metode najbližeg susjeda. U frekvencijskoj domeni se vidi da ima manje bočne komponente, ispod 10 % vrijednosti glavne komponente što je zadovoljavajuće. Glavni nedostatci su slabljenje pojačanja u propusnom pojasu i pojava aliasinga izvan graničnih frekvencija.



Sl. 2.3. Linearna interpolacija: a) jezgra, b) Fourierova transformacija jezgre, c) logaritamski prikaz Fourierove transformacije [8]

Bilinearna interpolacija je česta u praksi i pruža prilično dobru kvalitetu s obzirom na nisku računalnu složenost. Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.4.



a) b)

Sl. 2.4. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike bilinearnom interpolacijom: a) originalna slika, b) interpolirana slika [9]

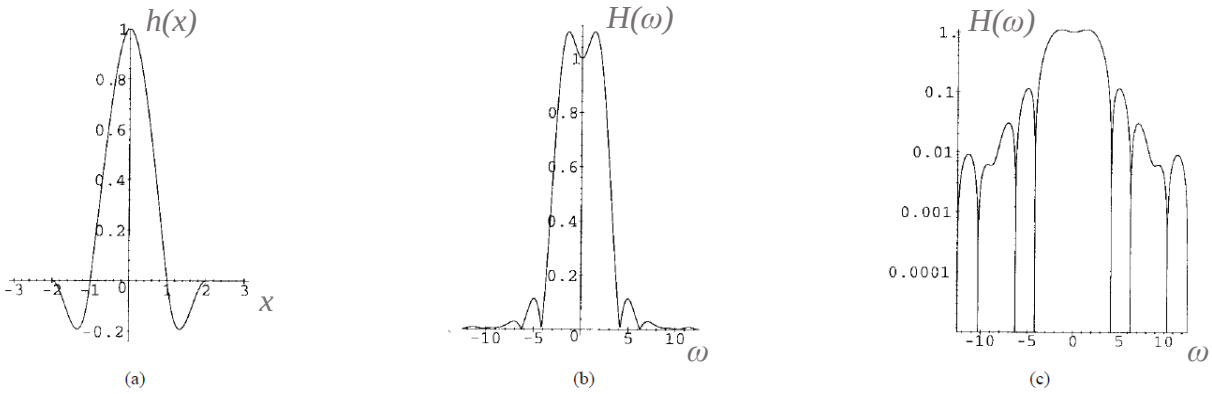
2.3. Bikubična interpolacija

Bikubična interpolacija je dvodimenzionalna kubična interpolacija. Može biti realizirana korištenjem Lagarange-ovih polinoma, kubičnih *spline*-ova itd. Interpolacijska jednadžba kubičnog *spline*-a je dobra aproksimacija idealne *sinc* funkcije, odnosno interpolacije. Slika interpolirana bikubičnom interpolacijom je puno glađa (engl. *smoother*) nego slika interpolirana metodom najbližeg susjeda ili bilinearnom interpolacijom. Bikubična interpolacija koristi težinski

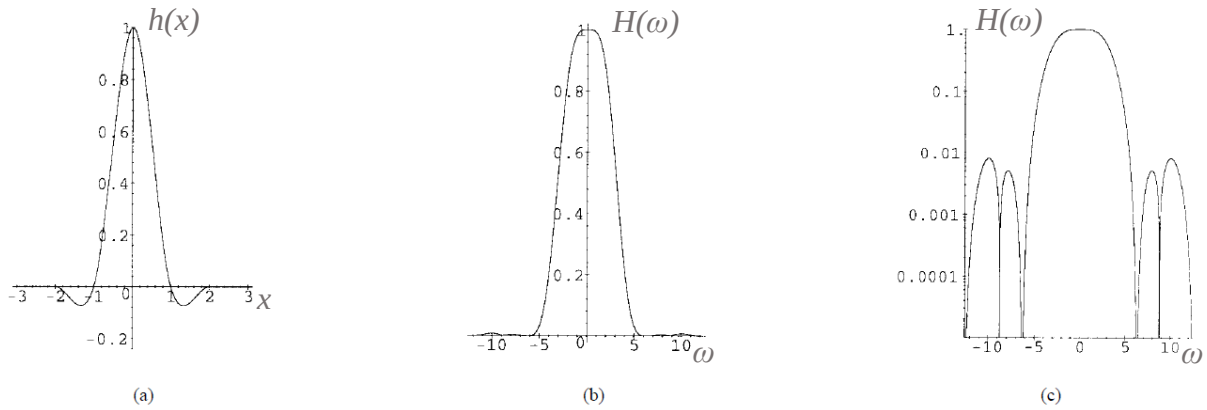
prosjek 16 okolnih elemenata slike za proračun interpoliranog elementa slike. Bikubična interpolacijska jezgra izvedena je iz interpolacijske jednadžbe kubičnog *spline*-a uz postavljanje granica. Interpolacijska jezgra prema [2] je:

$$h(x) = \begin{cases} (a+2)|x|3 - (a+3)|x|2 + 1, & 0 < |x| < 1 \\ a\,|x|3 - 5a\,|x|2 + 8a\,|x| - 4a, & 1 < |x| < 2 \\ 0, & 2 < |x| \end{cases}, \qquad (2\text{-}3)$$

gdje je *a* slobodni parametar. Vrijednost parametra *a* određuju performanse jezgre. Vrijednost između 0 i -3 aproksimira *sinc* funkciju. Vrijednost jednaka -1 će pojačati frekvencije na krajevima propusnog pojasa. To pojačanje rezultira izoštravanjem slike. Postavljanjem parametra *a* na vrijednost 0.5 dobiva se Catmull – Rom interpolacija koja daje dobru oštrinu slike. Bikubična interpolacija se najčešće koristi u komercijalnim programima, ima prihvatljivu računalnu složenost i daje zadovoljavajuće rezultate s gledišta kvalitete interpolirane slike. Na slici 2.5. nalazi se frekvencijska analiza interpolacije s parametrom a=-1, a na slici 2.6. s parametrom a=-0.5.



Sl. 2.5. Bikubična interpolacija (a=-1): a) jezgra, b) Fourierova transformacija jezgre, c) logaritamski prikaz Fourierove transformacije [8]



Sl. 2.6. Bikubična interpolacija (a=-0.5): a) jezgra, b) Fourierova transformacija jezgre, c) logaritamski prikaz Fourierove transformacije [8]

Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.7.



a) b)

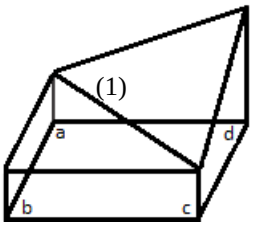
Sl. 2.7. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike bikubičnom interpolacijom: a) originalna slika, b) interpolirana slika [9]

2.4. Triangulacijska (trokutna) interpolacija ovisna o podacima

Triangulacijska (trokutna) interpolacija ovisna o podacima (engl. *Data dependent triangulation interpolation - DDT*) razvijena je u svrhu poboljšanja vizualne kvalitete slike uz smanjenje računalne kompleksnosti u odnosu na druge linearne interpolacije [3]. DDT se koristi za rješavanje nedostataka bilinearne interpolacije. Prednost nad ostalim algoritmima vidljiva je u dva segmenta. Prvi je niska računalna cijena i kompleksnost s obzirom na ostale algoritme, a drugi

je da se može koristiti proizvoljno u smislu povećanja ili smanjenja rezolucije, dok su ostali algoritmi namijenjeni samo za povećanje rezolucije.

U DDT interpolaciji prvo se povlači dijagonala (označena brojem (1) na slici 2.8.) između dva elementa slike koja čini dva trokuta koja povezuju četiri susjedna elementa slike kao što je prikazano na slici 2.8. Smjer dijagonale određuje se na osnovu ruba na slici odnosno vrijednosti četiri susjedna elementa slike. Uspoređuju se razlike | a - c | i | b - d | te se povlači dijagonala između para s manjom razlikom. Na taj način se određuje i pohranjuje smjer dijagonala (rubova) za sve kvadrate načinjene od 4 elementa slike. U drugom se koraku za svaki element slike koji treba interpolirati provjerava u kojem trokutu leži te se na osnovu vrijednosti elemenata slike koji čine taj trokut bilinearnom interpolacijom proračunava traženi element slike.



Sl. 2.8. DDT mreža [3]

Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.9.



a) b)

Sl. 2.9. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike DDT interpolacijom: a) originalna slika, b) interpolirana slika [9]

2.5. Nova rubom-upravljana interpolacija

Xin Li i ostali su u [4] predstavili novu rubom-upravljanu interpolaciju (engl. *New edge-directed interpolation - NEDI*) koja koristi geometrijsku dualnost između kovarijance slike niske rezolucije i kovarijance slike visoke rezolucije. Koristi se interpolacija četvrtog reda. Za povećanje rezolucije od 2^n puta, interpolacija se ponavlja kroz *n* iteracija. U NEDI metodi smjerovi rubova se ne uzimaju u obzir, a težine se izračunavaju uz pretpostavku da je lokalna kovarijanca slike konstantna kod slike visoke razlučivosti kao i kod slike niske razlučivosti uz određene faktore povećanja. S tim pretpostavkama dobiva se sustav jednadžbi čijim se rješavanjem dobivaju koeficijenti. Primjena ovog pravila na homogena područja (područja slike s vrlo malim promjenama vrijednosti elemenata slike gdje nema rubova) povećava računalnu cijenu izvođenja. Međutim, ova metoda zasnovana na kovarijanci ima dvostruko veću računalnu složenost nego linearne interpolacije. Kako bi se ona smanjila uz zadržavanje vizualne kvalitete slike, primjenjuje se samo na elemente slike koji se nalaze na rubovima dok se za ostale elemente slike primjenjuje bilinearna interpolacija. Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.10.



a) b)

Sl. 2.10. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike NEDI interpolacijom: a) originalna slika, b) interpolirana slika [9]

2.6. Iterativna interpolacija zasnovana na zakrivljenosti

Andrea Giachetti i Nicola Asuni predložili su interpolaciju u realnom vremenu, iterativnu interpolaciju zasnovanu na zakrivljenosti (engl. *Iterative Curvature Based Interpolation - ICBI*) [5]. Izvodi se u dva koraka, prvo se interpoliraju nove vrijednosti pomoću FCBI (engl. *Fast Curvature Based Interpolation*) algoritma nakon čega se interpolirane vrijednosti iterativnom

metodom modificiraju kako bi se smanjila funkcija „energije“ interpoliranog elementa slike. Ova funkcija dobiva se dodavanjem doprinosa za svaki interpolirani element slike, ovisno o drugoj derivaciji lokalnog kontinuiteta i drugih vrijednosti koje su minimalne kada je postignuta željena kvaliteta slike. Funkcija sumira lokalne promjene smjera druge derivacije. Prvotna vrijednost elementa slike računa se pomoću druge derivacije 8 susjednih elemenata slike u dva dijagonalna smjera; nakon toga element slike se interpolira sa srednjom vrijednošću ona dva elementa slike u kojem smjeru je manja vrijednost derivacije. Drugi korak je modifikacija interpoliranih elemenata kako bi se smanjila funkcija „energije“ interpoliranih elemenata slike, što se postiže pomoću malih promjena lokalnog kontinuiteta druge derivacije. Iterativnom metodom funkcija se prilagođava sve dok vrijednost ne pređe zadani prag.

Koraci izvođenja ICBI interpolacije [10]:

- preslikavanje originalnih elemenata slike u proširenu sliku
- proračun vrijednosti interpoliranih elemenata slike pomoću FCBI metode
- iterativna korekcija dok se ne postigne zadani prag
- proračun ostatka elemenata slike FCBI metodom
- iterativna korekcija primjenjuje se na novo dodane elemente slike
- ponavljanje procedure za daljnja poboljšanja slike

Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.11.

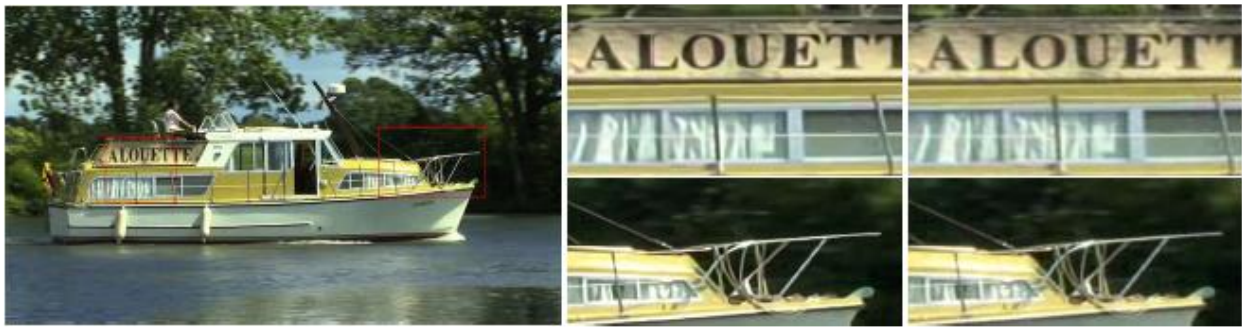


a) b)

Sl. 2.11. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike ICBI interpolacijom: a) originalna slika, b) interpolirana slika [9]

2.7. Prostorno-vremenska interpolacija zasnovana na samo-sličnosti

U [6] Ayvaci i ostali predložili su tehniku povećanja rezolucije koja se temelji na iskorištavanju sličnosti uzoraka video signala u prostornoj i vremenskoj domeni (engl. *Spatio-Temporal Self-Similarity*). Izvodi se kodiranje uzoraka slike s prethodno napravljenim rječnikom izgrađenim u lokalnom prostorno-vremenskom susjedstvu, te se uspostavlja podudarnost putem tehnike optičkog toka. Rezultati dobiveni korištenjem njihove predložene tehnike su usporedivi s rezultatima najsuvremenijih tehnika super razlučivosti (engl. *super resolution*). Računalna cijena i složenost ovog algoritma je srednja do visoka. Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.12.



a) b) c)

Sl. 2.12. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije dijela slike prostorno-vremenskom interpolacijom zasnovanom na samo-sličnosti: a) originalna slika, b) dio slike interpolirane prostorno-vremenskom interpolacijom zasnovanom na samo-sličnosti, c) dio slike interpolirane bikubičnom interpolacijom [6]

2.8. Interpolacija super-razlučivosti zasnovana na primjeru

Freeman i ostali u [7] predlažu model predviđanja koji se temelji na bazi podataka koja sadrži uzorke (engl. *Example-based super-resolution*). Uzorci su rastavljeni na frekvencijske komponente odnosno na niskofrekvencijsku komponentu (glatka slika) i visokofrekvencijsku komponentu (detalji na slici). Povećanje prostorne rezolucije postiže se analitičkom interpolacijom, zatim se visokofrekventne komponente predviđaju iz uzoraka spremljenih u bazi. Preklapanje se izvodi pomoću niskofrekventne komponente uzorka. Ovim načinom postižu se uvjerljivi detalji na cijeloj slici, na rubovima i na područjima s teksturom. Međutim, nedostatkom relevantnih uzoraka u bazi podataka može se pojaviti šum na slikama tj., nepravilnosti na zaobljenim rubovima. Upotreba većih baza podataka zahtjeva više vremena zbog dodatnih

usporedbi metodom najbližeg susjeda. Računalna cijena i vrijeme izvođenja ovog algoritma su relativno visoki zbog upotrebe vanjske baze podataka. Primjer slike dobivene pomoću ove metode uz dvostruko veći broj elemenata slike u horizontalnoj i vertikalnoj dimenziji dan je na slici 2.13.



Sl. 2.13. Primjer dvostrukog povećanja horizontalne i dvostrukog povećanja vertikalne prostorne rezolucije slike interpolacijom super-razlučivosti zasnovanom na primjeru:
a) originalna slika, b) interpolirana slika povećana interpolacijom super-razlučivosti zasnovanom na primjeru, c) interpolirana slika povećana bikubičnom interpolacijom [7]

3. VLASTITO RJEŠENJE ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA

3.1. Prostorna aktivnost video signala

Za razumijevanje vlastitog rješenja potrebno je znati što predstavljaju prostorna i vremenska aktivnost video signala. Prostorna aktivnost video signala (engl. *Spatial perceptual Information – SI*) je mjera koja prikazuje količinu detalja na slici [11]. Mjera je veća kod prostorno složenih scena. Nije namijenjena mjerenju entropije niti je povezana s teorijom komunikacija. Temelji se na Sobelovom filtru [12], svaki okvir video signala (luminantna komponenta) u vremenu *n* (F_n) prvo se filtrira Sobelovim filtrom. Nakon toga se računa standardna devijacija Sobelom filtriranog okvira (engl. *stdspace*). Ovaj postupak se ponavlja za svaki okvir video signala. Maksimalna vrijednost standardne devijacije svih okvira odabire se kao podatak koji prikazuje prostornu aktivnost video signala. Proces se prikazuje sljedećom jednadžbom:

$$SI = max_{time}\{std_{space}[Sobel(F_n)]\} \tag{3-1}$$

3.2. Vremenska aktivnost videosignala

Vremenska aktivnost video signala (engl. *Temporal perceptual Information – TI*) je mjera koja prikazuje količinu vremenskih promjena na slici. Mjera je veća kod scena s visokim intenzitetom pokreta na slici. Također, nije namijenjena mjerenju entropije niti je povezana s teorijom komunikacija [11]. Zasniva se na promjeni pokreta između dva okvira video signala $M_n(i,j)$, koja se računa kao razlika između odgovarajućih elemenata slike (luminantne komponente) na istoj lokaciji u uzastopnim okvirima. M_n je funkcija vremena i definirana je sljedećom jednadžbom:

$$M_n(i,j) = F_n(i,j) - F_{n-1}(i,j) \quad , \tag{3-2}$$

gdje je $F_n(i,j)$ element slike u *i*-tom retku i *j*-tom stupcu *n*-tog okvira video signala. Kao konačna vrijednost vremenske aktivnosti video signala uzima se maksimalna standardna devijacija svih međusobnih razlika okvira video signala, te je konačna jednadžba sljedeća:

$$TI = max_{time}\{std_{space}[M_n(i,j)]\} \quad (3\text{-}3)$$

Kod video signala koji sadrže dvije prekinute (odrezane) scene proračunava se dvije vremenske aktivnosti, prva u proračun uključuje okvir video signala na kojem je prekinuta scena, dok druga ne uključuje. Razlika između te dvije vremenske aktivnosti može biti značajna, što je bitno za daljnju obradu video signala.

3.3. Opis vlastitog rješenja za povećanje prostorne rezolucije video signala

Svaki okvir video signala se sastoji od tri komponente: *Y*, *U* i *V*. *Y* komponenta predstavlja intenzitet svijetlosti, a *U* i *V* komponente predstavljaju boju. Povećanje prostorne rezolucije vrši se na sve tri komponente, dok se za proračun *SI* i *TI* koristi samo *Y* komponenta. Program radi na principu proširivanja svakog okvira originalnog video signala praznim redcima i stupcima. Zatim se prazni redci i stupci popunjavaju s tri različita neadaptivna algoritma ovisno o prioritetu. Tri neadaptivna algoritma koje koristi ovo rješenje su:

- metoda najbližeg susjeda
- bilinearna interpolacija
- bikubična interpolacija

Prioritet se određuje ovisno o prostornoj i vremenskoj aktivnosti video signala koje su objašnjene u potpoglavlju 3.1 i 3.2. Prije prostornog povećanja rezolucije računa se prostorna i vremenska aktivnost za cjelokupan video signal, nakon čega se postupak ponavlja, ali za svaki okvir video signala posebno. Ovaj rad predstavlja dvije verzije vlastitog rješenja. Prva verzija (v1) koristi samo prostornu aktivnost video signala te odabire prioritete po sljedećim uvjetima:

- ako je vrijednost *SI* pojedinog okvira veća od 75% ukupne vrijednosti *SI* odabire se bikubična interpolacija
- ako je vrijednost *SI* pojedinog okvira između 40% i 75% ukupne vrijednosti *SI* odabire se bilinearna interpolacija
- ako je vrijednost *SI* pojedinog okvira manja od 40% ukupne vrijednosti *SI* odabire se metoda najbližeg susjeda

Druga verzija (v2) koristi i prostornu i vremensku aktivnost video signala te odabire prioritete po sljedećim uvjetima:

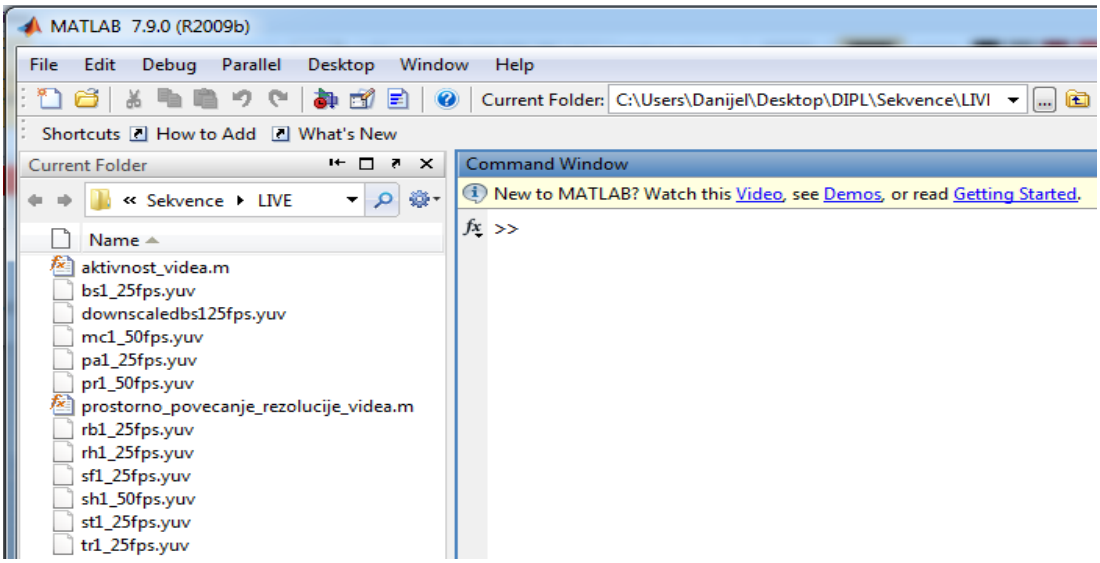
- ako je vrijednost *SI* pojedinog okvira veća od 75% ukupne vrijednosti *SI* odabire se bikubična interpolacija
- ako je vrijednost *SI* pojedinog okvira između 40% i 75% ukupne vrijednosti *SI* odabire se bilinearna interpolacija
- ako je vrijednost *SI* pojedinog okvira manja od 40% ukupne vrijednosti *SI* odabire se metoda najbližeg susjeda
- ako je istovremeno vrijednost *TI* pojedinog okvira veća od 75% ukupne vrijednosti *TI*, umjesto bikubične interpolacije koristi se bilinearna interpolacija, a umjesto bilinearne interpolacije koristi se metoda najbližeg susjeda
- ako je istovremeno vrijednost *TI* pojedinog okvira manja od 75% ukupne vrijednosti *TI* vrijede prva tri navedena uvjeta kao i kod prve verzije vlastitog rješenja (v1)

Prvi okvir video signala kod druge verzije vlastitog rješenja (v2) uvijek se interpolira bikubičnom interpolacijom jer vrijednost *TI* nije moguće izračunati za prvi okvir video signala (*TI* vrijednost se dobiva oduzimanjem trenutnog i prethodnog okvira). Na okvirima video signala s brzim pokretima i s puno detalja ljudsko oko slabije vidi izobličenja (visoka vrijednost *SI* i *TI*, visoka aktivnost video signala), za okvire takvih karakteristika trebala bi se koristiti jednostavnija interpolacija odnosno metoda najbližeg susjeda. Na okvirima video signala sa sporim pokretima i malo detalja (niska vrijednost *SI* i *TI*, niska aktivnost video signala) ljudsko oko bolje primjećuje izobličenja, za okvire takvih karakteristika trebala bi se koristiti složenija odnosno bikubična ili bilinearna interpolacija. Obzirom da se kvaliteta video signala povećane prostorne rezolucije računa objektivnom PSNR metrikom koja uspoređuje elemente slike na istim pozicijama u originalnom i video signalu povećane prostorne rezolucije, a ne subjektivnom metodom, odabrani su prethodno navedeni uvjeti za pojedini prioritet. Važnost prioriteta u odnosu na interpolacije je sljedeći:

- visoki prioritet – bikubična interpolacija
- srednji prioritet – bilinearna interpolacija
- niski prioritet – metoda najbližeg susjeda

3.4. Upute za pokretanje vlastitog rješenja za povećanje prostorne rezolucije video signala

Programsko rješenje izrađeno je u programskom paketu Matlab R2009b [13], te je isti potreban za pokretanje algoritma. Prije pokretanja algoritma nužno je pripremiti video signale u *.yuv* formatu te ih staviti u jednu datoteku zajedno sa skriptom algoritma (*prostorno_povecanje_rezolucije_videa.m*) i skriptom za proračun *SI* i *TI* (*aktivnost_videa.m*). Zatim je potrebno pokrenuti Matlab R2009b ili noviju verziju programa te odabrati datoteku s video signalima i skriptama kao što je prikazano na slici 3.1.



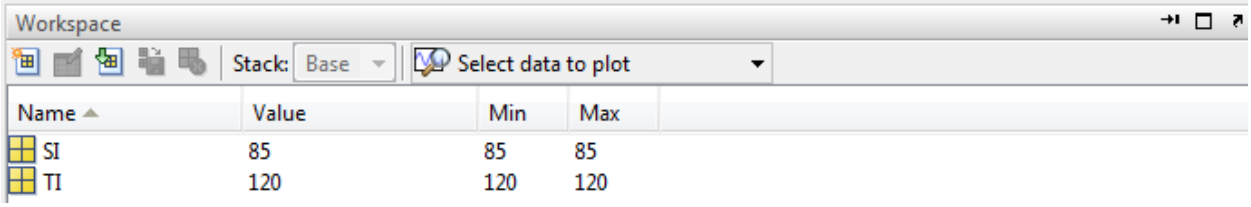
Sl. 3.1. Priprema za pokretanje vlastitog programskog rješenja u programskom paketu Matlab

Nakon pripreme potrebno je pokrenuti skriptu *aktivnost_videa.m*. U glavnoj konzoli programa Matlab, odnosno u prozoru za naredbe (engl. *Command Window*), potrebno je upisati naredbu kao što je prikazano na slici 3.2. i pričekati rezultate kao što je prikazano na slici 3.3. Rezultati se nalaze u radnom prozoru (engl. *Workspace*) u stupcu pod nazivom vrijednost (engl. *Value*).



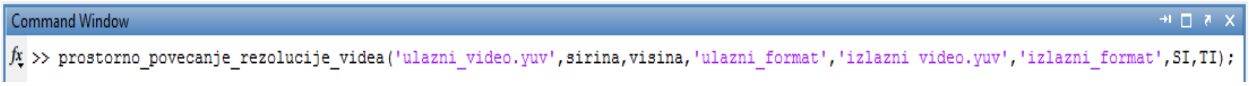
Sl. 3.2. Pokretanje skripte *aktivnost_videa.m*

Naredba se sastoji od četiri komponente koje su međusobno odvojene zarezom. Prva komponenta je naziv video signala kojem će se povećati rezolucija. Naziv mora sadržavati puno ime video signala uključujući i format (*.yuv*). Druga komponenta naredbe je širina video signala izražena u elementima slike. Treća komponenta je visina video signala, te četvrta je broj okvira koje sadrži video signal.



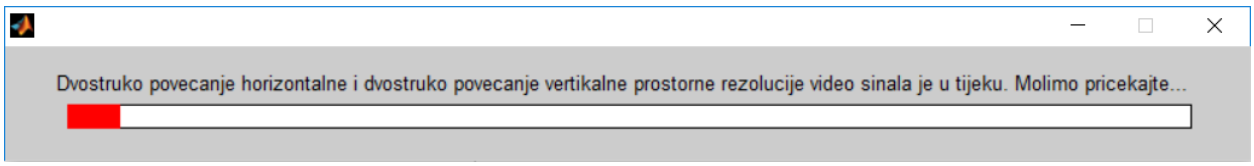
Sl. 3.3. Rezultati skripte *aktivnost_videa.m*

Nakon što je izvršen prvi dio, potrebno je pokrenuti glavnu skriptu pod nazivom *prostorno_povecanje_rezolucije_videa.m* upisivanjem naredbe u prozor za naredbe kao što je prikazano na slici 3.4.



Sl. 3.4. Pokretanje skripte *prostorno_povecanje_rezolucije_videa.m*

Naredba se sastoji od sedam komponenata koje su međusobno odvojene zarezom. Prva komponenta je naziv video signala kojem će se povećati rezolucija. Naziv mora sadržavati puno ime video signala uključujući i format (*.yuv*). Druga komponenta naredbe je širina video signala izražena u elementima slike. Treća komponenta je visina video signala. Četvrta komponenta je format uzorkovanja ulaznog video signala. Format mora biti zapisan unutar jednostrukih navodnika. Moguće vrste formata su: 400, 411, 420, 422 i 444. U praksi se najčešće pojavljuje format 420. Peta komponenta je naziv koji će biti dodijeljen novonastalom video signalu. Šesta komponenta je identična četvrtoj, ali se odnosi na izlazni videozapis. Sedma i osma komponenta su *SI* i *TI* vrijednosti koje su dobivene pokretanjem prve skripte (*aktivnost_videa.m*). Nakon unosa naredbe pojavljuje se prozor prikazan na slici 3.5. koji prikazuje trenutno stanje procesa.



Sl. 3.5. Proces obrade video signala

Nakon završetka procesa prozor sa slike 3.5 se zatvara, a video signal povećane prostorne rezolucije se nalazi u datoteci zajedno sa ostalim video signalima pod nazivom koji mu je zadan u petoj komponenti naredbe sa slike 3.4. Video signal povećane prostorne rezolucije je također u *.yuv* formatu dvostruko veće prostorne rezolucije. Dijelovi koda neophodni za učitavanje video signala u *.yuv* formatu, rastavljanje video signala na okvire te spremanje u izvorni format preuzeti su iz literature [14].

U prilogu 3.1 nalazi se kod skripte *aktivnost_videa.m*, a u prilogu 3.2 kod skripte *prostorno_povecanje_rezolucije_videa.m*.

4. VREDNOVANJE REZULTATA PREDLOŽENOG RJEŠENJA ZA POVEĆANJE PROSTORNE REZOLUCIJE VIDEO SIGNALA

4.1. Korištene baze video signala

U svrhu testiranja vlastitog rješenja u ovom diplomskom radu korištene su dvije različite javno dostupne baze podataka, odnosno video signala:

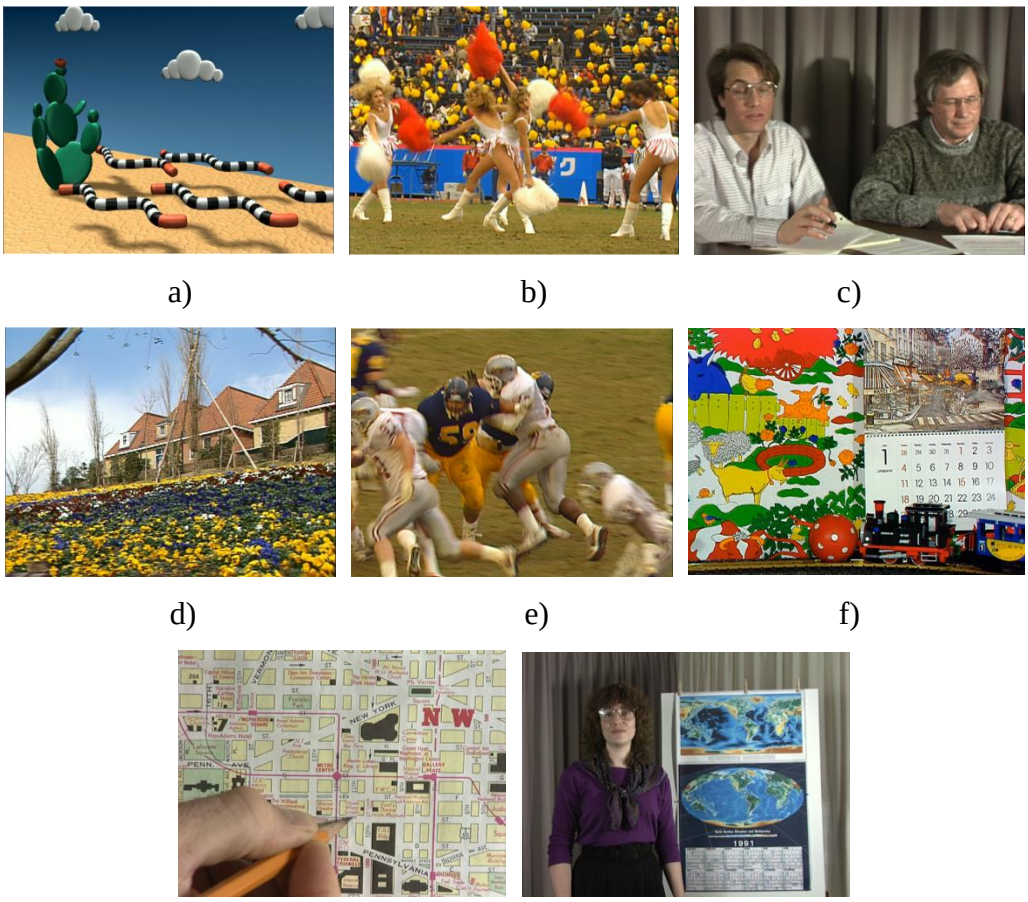
- VQG@ETFOS (engl. *Video Quality Group at Faculty of Electrical Engineering, Osijek*) ETFOS VGA Video Quality-EVVQ database [15],[16],[17]
- LIVE Video Quality Assessment Database (engl. *Laboratory for Image & Video Engineering-LIVE*) [18],[19],[20]

4.1.1. EVVQ baza podataka

EVVQ baza podataka sadrži osam nekomprimiranih progresivnih video sekvenci znatno različitog sadržaja (u smislu prostorne i vremenske aktivnosti). Referentne sekvence su u YUV 4:2:0 formatu, VGA 640x480 rezolucije s 25 okvira u sekundi, te svaka sadrži ukupno 300 okvira. U tablici 4.1. navedeni su korišteni video signali s pripadajućim *SI* i *TI* vrijednostima, a na slici 4.1. prikazan je po jedan okvir iz svakog video signala.

Tab. 4.1. Popis video signala iz EVVQ baze podataka

Broj video signala	Naziv video signala	SI	TI
1.	cartoon_vga.yuv	99.4412	78.8457
2.	cheerleaders_vga.yuv	98.6373	47.0082
3.	discussion_vga.yuv	64.0714	35.8541
4.	flower_garden_vga.yuv	140.4653	47.0957
5.	football_vga.yuv	99.3294	42.4196
6.	mobile_vga.yuv	168.2261	45.2550
7.	town_plan_vga.yuv	105.3311	23.7664
8.	weather_vga.yuv	68.8483	2.4620



Sl. 4.1. Primjer okvira pojedinog video signala iz EVVQ baze podataka:
a) cartoon_vga, b) cheerleaders_vga, c) discussion_vga, d) flower_garden_vga, e) football_vga, f) mobile_vga, g) town_plan_vga, h) weather_vga

4.1.2. LIVE baza podataka

LIVE baza podataka sadrži deset nekomprimiranih video signala visoke kvalitete sa širokim rasponom sadržaja. Referentne sekvence su u YUV 4:2:0 formatu, 768x432 rezolucije s 25 i 50 okvira u sekundi. U tablici 4.2. navedeni su korišteni video signali s pripadajućim *SI* i *TI* vrijednostima, a na slici 4.2. prikazan je po jedan okvir iz svakog video signala.

Tab. 4.2. Popis video signala iz LIVE baze podataka

Broj video signala	Naziv video signala	SI	TI
1.	bs1_25fps.yuv	156.5155	38.8717
2.	mc1_50fps.yuv	105.9724	37.4305
3.	pa1_25fps.yuv	56.4372	20.8651
4.	pr1_50fps.yuv	125.4363	48.1676
5.	rb1_25fps.yuv	62.8997	28.8131
6.	rh1_25fps.yuv	52.5208	14.3618
7.	sf1_25fps.yuv	69.9634	25.9725
8.	sh1_50fps.yuv	104.8405	43.6989
9.	st1_25fps.yuv	45.1769	11.5759
10.	tr1_25fps.yuv	72.2322	23.2732



a) b) c)



d) e) f)



g) h) i)



j)

Sl. 4.2. Primjer okvira pojedinog video signala iz LIVE baze podataka:
a) bs1_25fps, b) mc1_50fps, c) pa1_25fps, d) pr1_50fps, e) rb1_25fps, f) rh1_25fps, g) sf1_25fps, h) sh1_50fps, i) st1_25fps, j) tr1_25fps

4.2. Rezultati testiranja algoritma za povećanje prostorne rezolucije video signala

U svrhu usporedbe rezultata svim originalnim video signalima dvostruko je smanjena horizontalna i dvostruko je smanjena vertikalna prostorna rezolucija pomoću skripte *prostorno_smanjenje_rezolucije_videa.m* koja se nalazi u prilogu 4.1. Nakon toga video signalima se povećava prostorna rezolucija metodom najbližeg susjeda, bilinearnom interpolacijom, bikubičnom interpolacijom, vlastitim algoritmom v1 i v2. Usporedba rezultata kvalitete pojedinog

algoritma mjeri se PSNR, SSIM [21] i VQM [22] metrikom. Proračun objektivne ocjene rezultantnih video signala radi se u odnosu na originalne video signale. Za proračun je korišten besplatni *MSU Video Measurement Tool* računalni program [23]. U tablicama 4.3. - 4.8. prikazani su rezultati ocjena kvalitete za pojedine video signale i pojedine interpolacijske algoritme.

Tab. 4.3. Ocjene kvalitete video signala iz EVVQ baze podataka dobivene PSNR metrikom

Naziv video signala	PSNR [dB] - metoda najbližeg susjeda	PSNR [dB] - bilinearna interpolacija	PSNR [dB] - bikubična interpolacija	PSNR [dB] - vlastito rješenje v1	PSNR [dB] - vlastito rješenje v2
cartoon_vga.yuv	31.9860	30.6949	31.1220	31.0728	31.0729
cheerleaders_vga.yuv	28.7025	27.3737	27.9494	27.9494	27.8641
discussion_vga.yuv	32.6465	31.5642	31.9962	31.9962	31.9889
fl_garden_vga.yuv	25.5111	23.6689	24.2042	24.2042	24.1214
football_vga.yuv	31.3691	30.0299	30.5583	30.5009	30.4282
mobile_vga.yuv	23.1575	21.7851	22.1525	22.1525	22.1389
town_plan_vga.yuv	27.6415	26.3320	26.7648	26.7648	26.7615
weather_vga.yuv	32.2171	31.1176	31.6427	31.6427	31.5629

Tab. 4.4. Ocjena kvalitete video signala iz LIVE baze podataka dobivene PSNR metrikom

Naziv video signala	PSNR [dB] - metoda najbližeg susjeda	PSNR [dB] - bilinearna interpolacija	PSNR [dB] - bikubična interpolacija	PSNR [dB] - vlastito rješenje v1	PSNR [dB] - vlastito rješenje v2
bs1_25fps.yuv	26.2482	25.0669	25.4976	25.4369	25.3349
mc1_50fps.yuv	26.7775	25.7213	25.9778	25.9778	25.9772
pa1_25fps.yuv	33.8552	32.7798	33.1495	33.1495	32.9976
pr1_50fps.yuv	22.8087	21.8518	22.0846	22.0847	22.0842
rb1_25fps.yuv	31.3920	30.1084	30.5197	30.5197	30.1191
rh1_25fps.yuv	35.3416	34.3161	34.7609	34.7609	34.7250
sf1_25fps.yuv	33.3410	32.2135	32.6033	32.6033	32.5673
sh1_50fps.yuv	27.0368	25.9392	26.2158	26.1883	26.1877
st1_25fps.yuv	33.5604	32.4459	32.8040	32.8040	32.7746
tr1_25fps.yuv	30.8244	29.9077	30.3804	30.3804	30.1867

Tab. 4.5. Ocjene kvalitete video signala iz EVVQ baze podataka dobivene SSIM metrikom

Naziv video signala	SSIM - metoda najbližeg susjeda	SSIM - bilinearna interpolacija	SSIM - bikubična interpolacija	SSIM - vlastito rješenje v1	SSIM - vlastito rješenje v2
cartoon_vga.yuv	0.9557	0.9426	0.9486	0.9473	0.9473
cheerleaders_vga.yuv	0.9126	0.8797	0.9001	0.9001	0.8972
discussion_vga.yuv	0.9356	0.9139	0.9245	0.9245	0.9243
fl_garden_vga.yuv	0.8958	0.8320	0.8584	0.8584	0.8544
football_vga.yuv	0.9094	0.8782	0.8946	0.8916	0.8895
mobile_vga.yuv	0.8265	0.7473	0.7738	0.7738	0.7729
town_plan_vga.yuv	0.8765	0.8214	0.8439	0.8439	0.8438
weather_vga.yuv	0.9397	0.9220	0.9329	0.9329	0.9313

Tab. 4.6. Ocjene kvalitete video signala iz LIVE baze podataka dobivene SSIM metrikom

Naziv video signala	SSIM - metoda najbližeg susjeda	SSIM - bilinearna interpolacija	SSIM - bikubična interpolacija	SSIM - vlastito rješenje v1	SSIM - vlastito rješenje v2
bs1_25fps.yuv	0.9070	0.8702	0.8870	0.8848	0.8810
mc1_50fps.yuv	0.8314	0.7734	0.7934	0.7934	0.7934
pa1_25fps.yuv	0.9298	0.9107	0.9203	0.9203	0.9166
pr1_50fps.yuv	0.7793	0.7006	0.7297	0.7297	0.7297
rb1_25fps.yuv	0.9068	0.8710	0.8895	0.8895	0.8716
rh1_25fps.yuv	0.9637	0.9582	0.9641	0.9641	0.9637
sf1_25fps.yuv	0.9567	0.9507	0.9576	0.9576	0.9569
sh1_50fps.yuv	0.8516	0.7988	0.8184	0.8161	0.8161
st1_25fps.yuv	0.9001	0.8646	0.8797	0.8797	0.8786
tr1_25fps.yuv	0.9090	0.8801	0.8973	0.8973	0.8904

Tab. 4.7. Ocjene kvalitete video signala iz EVVQ baze podataka dobivene VQM metrikom

Naziv video signala	VQM - metoda najbližeg susjeda	VQM - bilinearna interpolacija	VQM - bikubična interpolacija	VQM - vlastito rješenje v1	VQM - vlastito rješenje v2
cartoon_vga.yuv	1.2347	1.7000	1.6373	1.6449	1.6429
cheerleaders_vga.yuv	2.2501	3.0169	2.8032	2.8032	2.8315
discussion_vga.yuv	1.3680	1.9363	1.8548	1.8548	1.8565
fl_garden_vga.yuv	3.1034	4.1484	3.8750	3.8750	3.9162
football_vga.yuv	1.6286	2.2197	2.1145	2.1145	2.1204
mobile_vga.yuv	4.1713	5.2798	5.0339	5.0339	5.0432
town_plan_vga.yuv	2.0496	2.6788	2.5430	2.5430	2.5441
weather_vga.yuv	1.2696	1.7749	1.6844	1.6844	1.6981

Tab. 4.8. Ocjene kvalitete video signala iz LIVE baze podataka dobivene VQM metrikom

Naziv video signala	VQM - metoda najbližeg susjeda	VQM - bilinearna interpolacija	VQM - bikubična interpolacija	VQM - vlastito rješenje v1	VQM - vlastito rješenje v2
bs1_25fps.yuv	2.9519	3.7176	3.5521	3.5790	3.6145
mc1_50fps.yuv	2.7209	3.4003	3.2744	3.2737	3.2741
pa1_25fps.yuv	1.2618	1.6771	1.6341	1.6341	1.6495
pr1_50fps.yuv	4.4273	5.4527	5.3004	5.2966	5.2971
rb1_25fps.yuv	1.8230	2.3603	2.2527	2.2527	2.3567
rh1_25fps.yuv	0.9293	1.4291	1.3559	1.3559	1.3625
sf1_25fps.yuv	1.3866	1.9116	1.8253	1.8253	1.8354
sh1_50fps.yuv	3.0478	3.8866	3.8012	3.8150	3.8151
st1_25fps.yuv	1.4504	1.8273	1.7407	1.7407	1.7478
tr1_25fps.yuv	1.8148	2.3689	2.2254	2.2254	2.2839

Provedena je usporedba brzine pojedinog interpolacijskog algoritma za jedan cijeli video signal iz EVVQ baze podataka i za jedan cijeli video signal iz LIVE baze podataka. Rezultati testiranja nalaze se u tablici 4.9.

Tab. 4.9. Usporedba brzine obrade jednog video signala pojedinom interpolacijom

Interpolacijski algoritam	Vrijeme obrade [s] - EVVQ	Vrijeme obrade [s] - LIVE
Metoda najbližeg susjeda	407.742	359.370
Bilinearna interpolacija	590.562	535.488
Bikubična interpolacija	558.120	511.236
Vlastito rješenje v1	580.326	509.244
Vlastito rješenje v2	571.758	490.986

Ako se pogledaju rezultati objektivne ocjene kvalitete video signala iz prikazanih tablica, može se vidjeti da metoda najbližeg susjeda daje najbolje rezultate i da su to pokazale sve tri korištene objektivne metrike. Takav je rezultat pomalo neočekivan. Razlog može ležati u činjenici da se kod ispitivanih materijala radi o relativno niskoj rezoluciji video signala i da ne postoji prevelika razlika između susjednih elemenata slike u originalnim slikama. Na taj način, nakon povećanja rezolucije ponavljanje elemenata slike neće dovesti do prevelike pogreške i neće znatno narušiti kvalitetu video signala. Međutim, ako se sekvence pogledaju, može se vidjeti da kvaliteta video signala nije baš takva kako sugeriraju objektivni rezultati. Naime, najveća vizualna degradacija uočava se upravo za metodu najbližeg susjeda, a najmanja za bikubičnu interpolaciju. Ova oprečnost u rezultatima kvalitete dobivenim objektivnim metrikama i samim gledanjem video signala može se objasniti činjenicom da su sve tri metrike korištene u ovom radu dizajnirane za ocjenu kvalitete mirne slike, a ne za ocjenu kvalitete video signala. One u svom proračunu kvalitete video signala ne uzimaju u obzir količinu pokreta u video signalu i učinke prostornog, a posebno vremenskog, maskiranja. Zbog toga ne mogu dati pouzdanu ocjenu kvalitete video signala kakvu daje čovjek. Također, ovi su rezultati još jednom potvrdili da je subjektivna procjena kvalitete najpouzdaniji način dobivanja ocjene kvalitete video signala.

Maksimalna razlika PSNR vrijednosti metode najbližeg susjeda u odnosu na ostale interpolacije iznosi 1.3069 dB. Većina PSNR vrijednosti vlastitog rješenja v1 nalaze se između PSNR vrijednosti bilinearne i bikubične interpolacije ili su vrijednosti jednake vrijednostima bikubične interpolacije što je očekivano. Takav rezultat dobiva se zbog toga što vlastito rješenje v1 kod odabira interpolacijskog algoritma za većinu okvira ili za sve okvire odabire bikubičnu interpolaciju. Postoji jedna iznimka gdje je PSNR vrijednost vlastitog rješenja v1 veća za 0.0001 dB od PSNR vrijednosti bikubične interpolacije. Takav rezultat dobiven je zbog toga što je za taj video signal (*pr1_50fps*) vlastito rješenje v1 za više okvira izabralo metodu najbližeg susjeda koja općenito ima najbolje PSNR rezultate kod svih video signala. Maksimalna postignuta razlika PSNR vrijednosti vlastitih rješenja u odnosu na bilinearnu interpolaciju iznosi 0.5757 dB, postignuta je kod video signala *cheerleaders_vga* u EVVQ bazi podataka. PSNR vrijednosti vlastitog rješenja v2 nalaze se između PSNR vrijednosti bilinearne i bikubične interpolacije što je očekivano. Takav rezultat dobiva se zbog toga što vlastito rješenje v2 kod odabira interpolacijskog algoritma u odnosu na vlastito rješenje v1 za odabir interpolacijskog algoritma uz vrijednost *SI* uključuje i vrijednost *TI* po uvjetima navedenim u potpoglavlju 3.3. Po tom uvjetu vlastito rješenje umjesto bikubične interpolacije koristi bilinearnu interpolaciju što smanjuje PSNR vrijednosti. Usporedbom PSNR vrijednosti bilinearne i bikubične interpolacije vidi se da bikubična

interpolacija ima veću vrijednost što je očekivano. Uvidom u tablice 4.5. - 4.8. vidi se da su SSIM i VQM metrike potvrdile rezultate PSNR metrike. Uvidom u tablicu 4.9. može se procijeniti računalna složenost pojedine interpolacije na osnovu vremena potrebnog za obradu video signala. Prema tome, metoda najbližeg susjeda ima najmanju računalnu složenost, zatim slijede bilinearna i bikubična interpolacija. Razlika u vremenu obrade između bilinearne interpolacije i metode najbližeg susjeda približno iznosi 180 sekundi za po jedan video signal iz obje baze podataka. Bikubična interpolacija izvršava se 32.442 sekunde brže od bilinearne interpolacije za jedan od video signala iz EVVQ baze podataka i 24.252 sekunde za jedan od video signala iz LIVE baze podataka. Takvo vrijeme obrade nije očekivano, a razlog je nedovoljno efikasna implementacija algoritma. Vrijeme obrade vlastitog rješenja v1 je za 22.206 sekundi sporije od vremena obrade bikubične interpolacije za jedan od video signala iz EVVQ baze podataka i 1.992 sekunde brže za jedan od video signala iz LIVE baze podataka. Obzirom da je vrijeme obrade bilinearne interpolacije duže od vremena obrade bikubične interpolacije rezultati vremena obrade vlastitog rješenja v1 u odnosu na bikubičnu interpolaciju su očekivani, no vrijeme obrade je u prosjeku 18 sekundi kraće u odnosu na bilinearnu interpolaciju. Razlika u vremenu obrade između vlastitog rješenja v1 i vlastitog rješenja v2 iznosi 8.568 sekundi za jedan od video signala iz EVVQ baze podataka i 18.258 sekundi za jedan od video signala iz LIVE baze podataka. Uvidom u tablicu 4.9. i razlike u vremenu obrade video signala zaključuje se da vlastita rješenja imaju veću brzinu obrade video signala uz teško uočljiv pad kvalitete video signala u odnosu na bilinearnu interpolaciju koja u ovom radu zahtjeva najviše vremena. Usporedbom vremena obrade vlastitih rješenja v1 i v2, PSNR vrijednosti vlastitih rješenja i neformalne subjektivne ocjene obrađenih video signala uočava se ušteda na vremenu obrade od 8 do 18 sekundi uz neznatan pad kvalitete video signala i pad PSNR vrijednosti koji je i dalje iznad PSNR vrijednosti bilinearne interpolacije kod vlastitog rješenja v2 u odnosu na vlastito rješenje v1. Usporedbom originalnih video signala i video signala interpoliranih bikubičnom interpolacijom i vlastitim rješenjima uočljiv je pad kvalitete što je očekivano. Kod video signala s visokim intenzitetom detalja pad kvalitete je uočljiviji nego kod video signala s manjim brojem detalja i visokim intenzitetom pokreta, odnosno visokom vrijednosti vremenske aktivnosti.

5. ZAKLJUČAK

Ovim radom analizirani su i teorijski objašnjeni neadaptivni i adaptivni interpolacijski algoritmi za povećanje prostorne rezolucije video signala. Glavni cilj rada bio je osmisliti i implementirati u nekom od računalnih programa novi interpolacijski algoritam, poboljšati postojeći ili iskoristiti postojeće algoritme kako bi se skratilo vrijeme obrade video signala uz istu ili približnu kvalitetu video signala. Nakon teorijske obrade područja uočena je mogućnost korištenja više neadaptivnih algoritama implementiranih u jedan algoritam, u ovisnosti o određenim parametrima. Odabrani su parametri prostorne i vremenske aktivnosti video signala. Proračun tih parametara prethodno povećanju prostorne rezolucije omogućio je postavljanje uvjeta po kojem će se birati interpolacijski algoritam posebno za svaki pojedini okvir. Korištene su metoda najbližeg susjeda, bilinearna i bikubična interpolacija. Kod okvira s visokom aktivnošću korištena je bikubična interpolacija, kod okvira sa srednjom aktivnošću korištena je bilinearna interpolacija te kod okvira s niskom aktivnošću korištena je metoda najbližeg susjeda. Takvi uvjeti odabrani su sukladno algoritmu PSNR metrike koja se uz SSIM i VQM metrike koristila za objektivno ocjenjivanje video signala. Testiranje korištenih interpolacijskih algoritama i vlastitih rješenja napravljeno je na dvije javno dostupne baze video signala, LIVE i EVVQ. Objektivno vrednovanje rezultata provedeno je proračunom PSNR, SSIM i VQM vrijednosti između originalnih video signala i signala s povećanom prostornom rezolucijom. Također je provedeno i neformalno subjektivno ocjenjivanje u svrhu boljeg određivanja granica pojedinih prioriteta kod odabira interpolacije za pojedini okvir. Pregledom i usporedbom postignutih rezultata postignut je željeni cilj. Vlastita rješenja daju približne PSNR vrijednosti kao i najsloženija bikubična interpolacija. SSIM i VQM metrike potvrdile su rezultate dobivene PSNR metrikom. Vrijeme obrade vlastitog rješenja v1 za pojedini video signal je u prosjeku 18 sekundi kraće nego vrijeme obrade bilinearne interpolacije. Vrijeme obrade vlastitog rješenja v2 za pojedini video signal je u prosjeku 32 sekunde kraće nego vrijeme obrade bilinearne interpolacije što je s obzirom na ostale dobivene rezultate zadovoljavajući rezultat. Neformalnom subjektivnom ocjenom utvrđeno je da je pad kvalitete između video signala obrađenih bikubičnom interpolacijom i vlastitim rješenjima gotovo neprimjetan. Vlastita rješenja pogodna su za primjenu kod većine današnjih video signala osim video signala koji se koriste u posebne svrhe (npr. medicinske i vojne). Ograničenje vlastitih rješenja je nemogućnost korištenja u stvarno-vremenskim aplikacijama zbog sporog vremena obrade video signala.

Daljnji razvoj vlastitih rješenja može biti usmjeren ka efikasnijoj implementaciji u smjeru poboljšanja brzine obrade video signala te hardverske implementacije algoritma. Može se poraditi i na preciznijem odabiru granica pojedinih prioriteta kod odabira interpolacije za pojedini okvir.

LITERATURA

[1] R. C. Gonzalez, R. E. Woods, Digital image processing, Prentice Hall, New Jersey, 2002.

[2] P. S. Parsania, P. V. Virparia, A Review: Image Interpolation Techniques for Image Scaling, International Journal of Innovative Research in Computer and Communication Engineering, No. 12, Vol. 2, pp. 7409-7414, December 2014.

[3] D. Su, P. Willis, Image Interpolation by Pixel Level Data-Dependent Triangulation, Computer graphics forum, No. 2, Vol. 23, pp. 189-201, June 2004.

[4] X. Li, M. T. Orchard, New edge-directed interpolation, Image Processing, IEEE Transactions, No. 10, Vol. 10, pp. 1521-1527, October 2001.

[5] A. Giachetti, N. Asuni, Real-Time ArtifactFree Image Upscaling, IEEE Transactions, Image Processing, No. 10, Vol. 20, pp. 2760-2768, October 2011.

[6] A. Ayvaci, H. Jin, Z. Lin, S. Cohen, S. Soatto, Video upscaling via spatio-temporal self similarity, International Conference on Pattern Recognition (ICPR), Vol. 21, pp. 2190-2193, Tsukuba, 2012.

[7] W. T. Freeman, T. R. Jones, E. C. Pasztor, Example-based super-resolution, IEEE Computer Graphics and Applications , No. 2, Vol. 22, pp. 56-65, March 2001.

[8] T. M. Lehmann, C. Gonner, K. Spitzer, Survey: Interpolation methods in medical image processing, IEEE Transactions on Medical Imaging, No. 11, Vol. 18, pp. 1049-1075, November 1999.

[9] V. Patel, K. Mistree, A Review on Different Image Interpolation Techniques for Image Enhancement, International Journal of Emerging Technology and Advanced Engineering, No. 3, Vol. 12, pp. 129-133 , December 2013.

[10] H. Aftab, A. B. Mansoor, M. Asim, A new single image interpolation technique for super resolution, Multitopic Conference, pp. 592-596, Karachi, December 2008.

[11] ITU-T, Subjective video quality assessment methods for multimedia applications, P.910, April 2004.

[12] O. R. Vincent, O. Folorunso, A Descriptive Algorithm for Sobel Image Edge Detection, Proceedings of Informing Science & IT Education Conference (InSITE), Sv. 613, pp. 97-107, Macon, 2009.

[13] http://www.mathworks.com/products/new_products/release2009b.html, pristup ostvaren 14.2.2016.

[14] http://freesourcecode.net/matlabprojects/69194/transform-yuv-file-in-matlab#.V-mlefkrLIU, pristup ostvaren 7.5.2016.

[15] http://www.etfos.unios.hr/vqg/, pristup ostvaren 22.5.2016.

[16] S. Rimac-Drlje, M. Vranješ, D. Žagar, Foveated mean squared error – a novel video quality metric, Multimedia Tools and Applications, No. 3, Vol. 49, pp. 425-445, 2010.

[17] M. Vranješ, S. Rimac-Drlje, K. Grgić, Review of Objective Video Quality Metrics and Performance Comparison Using Different Databases, Signal Processing-Image Communication, No. 1, Vol. 28, pp. 1-19, 2012.

[18] http://live.ece.utexas.edu/research/quality/live_video.html, pristup ostvaren 22.5.2016.

[19] K. Seshadrinathan, R. Soundararajan, A. C. Bovik and L. K. Cormack, Study of Subjective and Objective Quality Assessment of Video, IEEE Transactions on Image Processing, No. 6, Vol. 19, pp. 1427-1441, June 2010.

[20] K. Seshadrinathan, R. Soundararajan, A. C. Bovik and L. K. Cormack, A Subjective Study to Evaluate Video Quality Assessment Algorithms, SPIE Proceedings Human Vision and Electronic Imaging, No. 6, Vol. 19, pp. 1427-1441, Jan. 2010.

[21] Z. Wang, L. Lu, A.C. Bovik, Video quality assessment based on structural distortion measurement, Signal Processing: Image Communication, No. 2, Vol. 19, pp. 121-132, 2004.

[22] M.H. Pinson, S. Wolf, A new standardized method for objectively measuring video quality, IEEE Transactions on Broadcasting, Vol. 50, pp. 312-322, 2004.

[23] http://www.compression.ru/video/quality_measure/video_measurement_tool_en.html, pristup ostvaren 9.5.2016.

SAŽETAK

Prostorno povećanje rezolucije video signala ima važnu ulogu u multimedijskoj komunikaciji. U radu je dana teorijska podloga neadaptivnih i adaptivnih interpolacijskih algoritama. Implementirano je vlastito rješenje, odnosno vlastiti algoritam za prostorno povećanje rezolucije video signala u programskom paketu Matlab. Napravljena je usporedba rezultata vlastitog rješenja s rezultatima metode najbližeg susjeda, bilinearne i bikubične interpolacije. Za vrednovanje rezultata proračunata je PSNR, SSIM i VQM vrijednost kvalitete video signala te je mjereno vrijeme obrade video signala. Vlastito rješenje pokazalo se zadovoljavajućim s obzirom na kvalitetu video signala i vrijeme obrade za većinu obrađenih video signala.

KLJUČNE RIJEČI: povećanje prostorne rezolucije, video signal, interpolacijski algoritmi, prostorna aktivnost video signala, vremenska aktivnost video signala, Matlab, EVVQ baza podataka, LIVE baza podataka

ALGORITHMS FOR INCREASING OF SPATIAL RESOLUTION OF VIDEO SIGNALS

SUMMARY

Spatail upscaling of video plays an important role in the multimedia communication. The paper presents a theoretical background of non-adaptive and adaptive interpolation algorithms. Own solution has been implemented, respectively a own algorithm for spatial upscaling of video obtained using Matlab. A comparison of results were made between own solution and nearest neighbor method, bilinear and bicubic interpolation. For the evaluation of results PSNR, SSIM and VQM value of video quality is calculated and processing time of the videos is measured as well. Own solution proved to meet requirements in regards to the quality of the video and processing time for most of the processed videos.

KEY WORDS: spatial upscaling, video, interpolation algorithms, spatial perceptual information of video, temporal perceptual information of video, Matlab, EVVQ database, EVVQ database

ŽIVOTOPIS

Danijel Vukoje rođen je 5. ožujka 1992. godine. U vremenskom razdoblju 1998-2006 pohađao je osnovnu školu OŠ Darda. 2006. godine upisuje srednju Elektrotehničku i prometnu školu Osijek, smjer elektrotehničar. Maturirao je s odličnim uspjehom. 2010. godine završio je srednju školu te se upisuje na Sveučilište J. J. Strossmayera Osijek, Elektrotehnički fakultet Osijek. 2013. godine završio je preddiplomski studij elektrotehnike, smjer komunikacije i informatika. Diplomirao je s temom *Usporedba performansi različitih diskretnih modulacijskih postupaka* i prosjekom ocjena 4.125. Iste godine upisuje diplomski studij, smjer komunikacije i informatika. Izvrsno poznaje rad na računalu. Od svjetskih jezika govori i piše engleski. Ima položen vozački ispit B kategorije.

PRILOZI

P 3.1. Kod *aktivnost_videa.m* skripte

```
%Primjer:
%aktivnost_videa('st1_25fps.yuv',768,432,217);

function [SI,SI1,TI,TI1,SITI,SITI1]=aktivnost_videa(filename,width,height,
framenumber1)

tStart=tic;
yuvfile=yuvread(filename,width,height);

for cntf = 1:(framenumber1)

    y(:,:)=double(yuvfile(:,:,1,cntf));
    for i=2:height-1
        for j=2:width-1
            Gv(i,j)=-1*y(i-1,j-1)-2*y(i-1,j)-y(i-1,j+1)+y(i+1,j-
1)+2*y(i+1,j)+y(i+1,j+1);
            Gh(i,j)=-1*y(i-1,j-1)-2*y(i,j-1)-y(i+1,j-1)+y(i-
1,j+1)+2*y(i,j+1)+y(i+1,j+1);
            G(i,j)=sqrt((Gv(i,j)).^2+(Gh(i,j)).^2);
        end
    end

    G1=G(2:(height-1),2:(width-1));
    %zbog toga što i i j idu od dva on na mjesta
    %u matricama Gv i Gh (pa samim time i G) kada
    %je i ili j jednako 1 stavlja nule pa to treba
    %popraviti tako da se prije računanja
    %standardne devijacije izbace iz
    %matrice G prvi red i prvi stupac (da
    %te nule ne ulaze u proračun standardne
    %devijacije)
    stdevS(cntf)=std2(G1);
end

SI=max(stdevS)
SI1=mean(stdevS);

for cntf=2:framenumber1
    y1(:,:)=double(yuvfile(:,:,1,cntf));
    y2(:,:)=double(yuvfile(:,:,1,cntf-1));
    M(:,:)=y1(:,:)-y2(:,:);
    stdevT(cntf)=std2(M(:,:));
end

TI=max(stdevT)
TI1=mean(stdevT(2:length(stdevT)));
%stdevT za cntf=1 će po defaultu biti
%nula jer brojač cntf ide od 2 kad se računa
%TI (budući da prvi okvir nema prethodnika
%pa mu se ne može računati TI)
% zato se računa srednja vrijednost
% od drugog elementa tog vektora do
```

```
% zadnjeg (kod uzimanja maksimalne
% vrijednosti u redu prije ne smeta ta nula u
% vektoru)
SITI=TI*SI;
SITI1=TI1*SI1;
vrijeme_obrade=toc(tStart)

function YUV1 = yuvread(File1,width,height)

%set factor for UV-sampling
fwidth = 0.5;
fheight= 0.5;

%get Filesize and Framenumber
filep = dir(File1);
fileBytes = filep.bytes; %Filesize1
clear filep
framenumber1 = fileBytes/(width*height*(1+2*fheight*fwidth)); %Framenumber1
%filep = dir(File1);
%fileBytes = filep.bytes; %Filesize2
%clear filep
%framenumber2 = fileBytes/(width*height*(1+2*fheight*fwidth)); %Framenumber2
if mod(framenumber1,1) ~= 0
    %display('Error: wrong resolution, format, filesize or different video
lengths');
    k=8212;
else
    k=0;
end
%h = waitbar(0,'Please wait ... ');
for cntf = 1:(framenumber1)
    %waitbar(cntf/framenumber1,h);
    %load data of frames
    YUV1(:,:,:,cntf) = loadFileYUV(width,height,cntf,File1,fheight,fwidth,k);
end

% read YUV-data from file
function YUV = loadFileYUV(width,heigth,Frame,fileName,Teil_h,Teil_b,k)

% get size of U and V
fileId = fopen(fileName,'r');
width_h = width*Teil_b;
heigth_h = heigth*Teil_h;
% compute factor for framesize
factor = 1+(Teil_h*Teil_b)*2;
% compute framesize
framesize = width*heigth;

fseek(fileId,(Frame-1)*factor*framesize+k, 'bof');
% create Y-Matrix
YMatrix = fread(fileId, width * heigth, 'uchar');
YMatrix = int16(reshape(YMatrix,width,heigth));
%YMatrix = int16(YMatrix);
%YMatrix = int16(reshape(YMatrix,heigth,width));
%size(YMatrix);
% create U- and V- Matrix
if Teil_h == 0
    UMatrix = 0;
    VMatrix = 0;
```

```
else
    UMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    UMatrix = int16(UMatrix);
    UMatrix = reshape(UMatrix,width_h, heigth_h);
    %UMatrix = reshape(UMatrix,heigth_h, width_h);

    VMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    VMatrix = int16(VMatrix);
    VMatrix = reshape(VMatrix,width_h, heigth_h);
    %VMatrix = reshape(VMatrix,heigth_h,width_h);
end
% compose the YUV-matrix:
%size(YMatrix)
%heigth
%width
YUV(1:width,1:heigth,1) = YMatrix;
YUVt(:,:,1)=transpose(YUV(:,:,1));
%YUV(1:heigth,1:width,1) = YMatrix;
if Teil_h == 0
    YUV(:,:,2) = 127;
    YUV(:,:,3) = 127;
end
% consideration of the subsampling of U and V

UMatrix1(1:width_h,1:heigth) = int16(0);
UMatrix1(1:width_h,1:2:end) = UMatrix(:,1:1:end);
UMatrix1(1:width_h,2:2:end) = UMatrix(:,1:1:end);

VMatrix1(1:width_h,1:heigth) = int16(0);
VMatrix1(1:width_h,1:2:end) = VMatrix(:,1:1:end);
VMatrix1(1:width_h,2:2:end) = VMatrix(:,1:1:end);

YUV(1:width_h,1:heigth,2) = int16(0);
YUV(1:2:end,:,2) = UMatrix1(:,:);
YUV(2:2:end,:,2) = UMatrix1(:,:);

YUV(1:width_h,1:heigth,3) = int16(0);
YUV(1:2:end,:,3) = VMatrix1(:,:);
YUV(2:2:end,:,3) = VMatrix1(:,:);

YUVt(:,:,2)=transpose(YUV(:,:,2));
YUVt(:,:,3)=transpose(YUV(:,:,3));

YUV = uint8(YUVt);
fclose(fileId);
```

P 3.2. Kod *prostorno_povecanje_rezolucije_videa.m* skripte

```
%Primjer:
%prostorno_povecanje_rezolucije_videa('st1_25fps_downscaled.yuv',384,216,
'420','st1_25fps_2x_upscaled.yuv'','420',45.1769,11.5759);

function
prostorno_povecanje_rezolucije_videa(ulazni_video,ulazni_sirina,ulazni_visina
,ulazni_format,izlazni_video,izlazni_format,SI,TI)%za v1 umjesto TI staviti ~

tStart=tic;
provjera = dir(ulazni_video);
velicina_bitovi = provjera.bytes;
clear provjera

[ulazni_faktor_sirina, ulazni_faktor_visina]=dohvati_format(ulazni_format);
[izlazni_faktor_sirina,
izlazni_faktor_visina]=dohvati_format(izlazni_format);

broj_okvira = velicina_bitovi/(ulazni_sirina * ulazni_visina * (1 + 2 *
ulazni_faktor_sirina * ulazni_faktor_visina));

yuvfile=yuvread(ulazni_video,ulazni_sirina,ulazni_visina);
YUV=zeros(ulazni_visina,ulazni_sirina,3);

fclose(fopen(izlazni_video,'w'));
h = waitbar(0,'Dvostruko povecanje prostorne rezolucije video signala je u
tijeku. Molimo pricekajte...');
for brojac = 1:1:broj_okvira
    waitbar(brojac/broj_okvira,h);
    YUV =
loadFileYUV(ulazni_sirina,ulazni_visina,brojac,ulazni_video,ulazni_faktor_vis
ina,ulazni_faktor_sirina);
    YUV=double(YUV);

    for i=2:ulazni_visina-1
        for j=2:ulazni_sirina-1
            Gv(i,j)=-1*YUV(i-1,j-1)-2*YUV(i-1,j)-YUV(i-1,j+1)+YUV(i+1,j-
1)+2*YUV(i+1,j)+YUV(i+1,j+1);
            Gh(i,j)=-1*YUV(i-1,j-1)-2*YUV(i,j-1)-YUV(i+1,j-1)+YUV(i-
1,j+1)+2*YUV(i,j+1)+YUV(i+1,j+1);
            G(i,j)=sqrt((Gv(i,j)).^2+(Gh(i,j)).^2);
        end
    end

    G1=G(2:(ulazni_visina-1),2:(ulazni_sirina-1));
    tempSI=std2(G1);

    if brojac==1
        tempTI=0.1;
    else
        y1(:,:)=double(yuvfile(:,:,1,brojac));
        y2(:,:)=double(yuvfile(:,:,1,brojac-1));
        M(:,:)=y1(:,:)-y2(:,:);
        tempTI=std2(M(:,:));
    end
```

```
    if (tempTI<(0.75*TI))           %za v1 obrisati ovu liniju koda
        if (tempSI<=(0.40*SI))
            interpolacija=1;
        elseif (tempSI>(0.40*SI) && tempSI<(0.75*SI))
            interpolacija=2;
        else
            interpolacija=3;
        end
    else                            %za v1 obrisati ovu liniju koda
        if (tempSI<=(0.40*SI))      %za v1 obrisati ovu liniju koda
            interpolacija=1;        %za v1 obrisati ovu liniju koda
        elseif (tempSI>(0.40*SI) && tempSI<(0.75*SI)) %v1 obrisati
            interpolacija=1;        %za v1 obrisati ovu liniju koda
        else                        %za v1 obrisati ovu liniju koda
            interpolacija=2;        %za v1 obrisati ovu liniju koda
        end                         %za v1 obrisati ovu liniju koda
    end                             %za v1 obrisati ovu liniju koda

    rezultantni_YUV=obrada_live(YUV, ulazni_sirina, ulazni_visina,
interpolacija);   %za 768x432 rezoluciju
    %rezultantni_YUV=obrada_vga(YUV, ulazni_sirina, ulazni_visina,
interpolacija);   %za 640x480 rezoluciju
    rezultantni_YUV=uint8(rezultantni_YUV);

save_yuv(rezultantni_YUV,izlazni_video,768,432,izlazni_faktor_visina,izlazni_
faktor_sirina);   %za 768x432 rezoluciju

%save_yuv(rezultantni_YUV,izlazni_video,640,480,izlazni_faktor_visina,izlazni
_faktor_sirina);  %za 768x432 rezoluciju
end
close(h);
format longEng;
vrijeme_obrade=(toc(tStart))/60
end

function YUV1 = yuvread(File1,width,height)

%set factor for UV-sampling
fwidth = 0.5;
fheight= 0.5;

%get Filesize and Framenumber
filep = dir(File1);
fileBytes = filep.bytes; %Filesize1
clear filep
framenumber1 = fileBytes/(width*height*(1+2*fheight*fwidth)); %Framenumber1
%filep = dir(File1);
%fileBytes = filep.bytes; %Filesize2
%clear filep
%framenumber2 = fileBytes/(width*height*(1+2*fheight*fwidth)); %Framenumber2
if mod(framenumber1,1) ~= 0
    %display('Error: wrong resolution, format, filesize or different video
lengths');
    k=8212;
else
    k=0;
end
%h = waitbar(0,'Please wait ... ');
for cntf = 1:(framenumber1)
```

```
    %waitbar(cntf/framenumber1,h);
    %load data of frames
    YUV1(:,:,:,cntf) = loadFileYUV(width,height,cntf,File1,fheight,fwidth);

end
end

function izlaz=obrada_live(ulaz,sirina,visina,interpolacija)

prosireno=prosiri_okvir(ulaz,216,384);
matrica=zeros(440,776,3);
for k=1:3
    for i=1:(visina*2)
        for j=1:(sirina*2)
            matrica(i+3,j+3,k)=prosireno(i,j,k);
        end
    end
end
matrica=interpoliraj(matrica,768,432,interpolacija);
izlaz=zeros(432,768,3);
for k=1:3
    for i=4:(435)
        for j=4:(771)
            izlaz(i-3,j-3,k)=matrica(i,j,k);
        end
    end
end
end

function izlaz=obrada_vga(ulaz,sirina,visina,interpolacija)

prosireno=prosiri_okvir(ulaz,240,320);
matrica=zeros(488,648,3);
for k=1:3
    for i=1:(visina*2)
        for j=1:(sirina*2)
            matrica(i+3,j+3,k)=prosireno(i,j,k);
        end
    end
end
matrica=interpoliraj(matrica,640,480,interpolacija);
izlaz=zeros(480,640,3);
for k=1:3
    for i=4:(483)
        for j=4:(643)
            izlaz(i-3,j-3,k)=matrica(i,j,k);
        end
    end
end
end

function matrica=interpoliraj(matrica,sirina,visina,interpolacija)

prolaz=1;
for k=1:3
    for i=1:2:(visina)
        for j=1:2:(sirina)
            if interpolacija==1
                matrica(i+3,j+3,k)=susjed(matrica,i,j,k,prolaz);
```

```
            end
            if interpolacija==2

matrica(i+3,j+3,k)=bilinearna(matrica,i,j,k,prolaz,visina,sirina);
            end
            if interpolacija==3

matrica(i+3,j+3,k)=bikubicna(matrica,i,j,k,prolaz,visina,sirina);
            end
        end
    end
end
prolaz=2;
for k=1:3
    for i=2:2:(visina)
        for j=1:2:(sirina)
            if interpolacija==1
                matrica(i+3,j+3,k)=susjed(matrica,i,j,k,prolaz);
            end
            if interpolacija==2

matrica(i+3,j+3,k)=bilinearna(matrica,i,j,k,prolaz,visina,sirina);
            end
            if interpolacija==3

matrica(i+3,j+3,k)=bikubicna(matrica,i,j,k,prolaz,visina,sirina);
            end
        end
    end
end
prolaz=3;
for k=1:3
    for i=1:2:(visina)
        for j=2:2:(sirina)
            if interpolacija==1
                matrica(i+3,j+3,k)=susjed(matrica,i,j,k,prolaz);
            end
            if interpolacija==2

matrica(i+3,j+3,k)=bilinearna(matrica,i,j,k,prolaz,visina,sirina);
            end
            if interpolacija==3

matrica(i+3,j+3,k)=bikubicna(matrica,i,j,k,prolaz,visina,sirina);
            end
        end
    end
end
end

function izlaz=bikubicna(matrica,i,j,k,prolaz,visina,sirina)

if i>=5 && j>=5 && i<=(visina-5) && j<=(sirina-5)
    switch prolaz
        case 1 %sredina

izlaz=((((matrica(i+4,j+4,k)+matrica(i+2,j+2,k)+matrica(i+4,j+2,k)+matrica(i+
2,j+4,k)))*0.3)+((matrica(i+6,j+6,k)+matrica(i,j,k)+matrica(i+6,j,k)+matrica(
i,j+6,k))*(-0.05)));
        case 2 %donji lijevi ugao
```

```
izlaz=((matrica(i+3,j+4,k)+matrica(i+3,j+2,k)+matrica(i+2,j+3,k)+matrica(i+4,
j+3,k))*0.3+(-
0.05)*((matrica(i+3,j+6,k)+matrica(i+3,j,k)+matrica(i,j+3,k)+matrica(i+6,j+3,
k))));
        case 3 %gornji desni ugao

izlaz=((matrica(i+3,j+4,k)+matrica(i+3,j+2,k)+matrica(i+2,j+3,k)+matrica(i+4,
j+3,k))*0.3+(-
0.05)*((matrica(i+3,j+6,k)+matrica(i+3,j,k)+matrica(i,j+3,k)+matrica(i+6,j+3,
k))));
    end
else
    izlaz=susjed(matrica,i,j,k,prolaz);
end
end

function izlaz=bilinearna(matrica,i,j,k,prolaz,visina,sirina)

if i>=3 && j>=3 && i<=(visina-3) && j<=(sirina-3)
    switch prolaz
        case 1 %sredina

izlaz=(matrica(i+4,j+4,k)+matrica(i+2,j+2,k)+matrica(i+4,j+2,k)+matrica(i+2,j
+4,k))/4;
        case 2 %donji lijevi ugao

izlaz=(matrica(i+3,j+4,k)+matrica(i+3,j+2,k)+matrica(i+2,j+3,k)+matrica(i+4,j
+3,k))/4;
        case 3 %gornji desni ugao

izlaz=(matrica(i+3,j+4,k)+matrica(i+3,j+2,k)+matrica(i+2,j+3,k)+matrica(i+4,j
+3,k))/4;
    end
else
    izlaz=susjed(matrica,i,j,k,prolaz);
end
end

function izlaz=susjed(matrica,i,j,k,prolaz)

switch prolaz
    case 1 %sredina
        izlaz=matrica(i+4,j+4,k);
    case 2 %donji lijevi ugao
        izlaz=matrica(i+3,j+4,k);
    case 3 %gornji desni ugao
        izlaz=matrica(i+4,j+3,k);
end
end

function izlaz=prosiri_okvir(YUV,visina,sirina)

izlaz=zeros(visina*2,sirina*2,3);
for k=1:3
    u=2;
    for i=1:(visina)
        t=2;
        for j=1:(sirina)
```

```
            izlaz(u,t,k)=YUV(i,j,k);
            t=t+2;
        end
        u=u+2;
    end
end
end

function [faktor_sirina,faktor_visina] = dohvati_format(format)

faktor_sirina = 0.5;
faktor_visina= 0.5;
if strcmp(format,'400')
    faktor_sirina = 0;
    faktor_visina= 0;
elseif strcmp(format,'411')
    faktor_sirina = 0.25;
    faktor_visina= 1;
elseif strcmp(format,'420')
    faktor_sirina = 0.5;
    faktor_visina= 0.5;
elseif strcmp(format,'422')
    faktor_sirina = 0.5;
    faktor_visina= 1;
elseif strcmp(format,'444')
    faktor_sirina = 1;
    faktor_visina= 1;
else
    display('Pogrešan format!');
end
end

function YUV = loadFileYUV(width,heigth,Frame,fileName,Teil_h,Teil_b)

% get size of U and V
fileId = fopen(fileName,'r');
width_h = width*Teil_b;
heigth_h = heigth*Teil_h;
% compute factor for framesize
factor = 1+(Teil_h*Teil_b)*2;
% compute framesize
framesize = width*heigth;

fseek(fileId,(Frame-1)*factor*framesize, 'bof');
% create Y-Matrix
YMatrix = fread(fileId, width * heigth, 'uchar');
YMatrix = int16(reshape(YMatrix,width,heigth)');
% create U- and V- Matrix
if Teil_h == 0
    UMatrix = 0;
    VMatrix = 0;
else
    UMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    UMatrix = int16(UMatrix);
    UMatrix = reshape(UMatrix,width_h, heigth_h).';

    VMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    VMatrix = int16(VMatrix);
    VMatrix = reshape(VMatrix,width_h, heigth_h).';
```

```
end
% compose the YUV-matrix:
YUV(1:heigth,1:width,1) = YMatrix;

if Teil_h == 0
    YUV(:,:,2) = 127;
    YUV(:,:,3) = 127;
end
% consideration of the subsampling of U and V
if Teil_b == 1
    UMatrix1(:,:) = UMatrix(:,:);
    VMatrix1(:,:) = VMatrix(:,:);

elseif Teil_b == 0.5
    UMatrix1(1:heigth_h,1:width) = int16(0);
    UMatrix1(1:heigth_h,1:2:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,2:2:end) = UMatrix(:,1:1:end);

    VMatrix1(1:heigth_h,1:width) = int16(0);
    VMatrix1(1:heigth_h,1:2:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,2:2:end) = VMatrix(:,1:1:end);

elseif Teil_b == 0.25
    UMatrix1(1:heigth_h,1:width) = int16(0);
    UMatrix1(1:heigth_h,1:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,2:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,3:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,4:4:end) = UMatrix(:,1:1:end);

    VMatrix1(1:heigth_h,1:width) = int16(0);
    VMatrix1(1:heigth_h,1:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,2:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,3:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,4:4:end) = VMatrix(:,1:1:end);
end

if Teil_h == 1
    YUV(:,:,2) = UMatrix1(:,:);
    YUV(:,:,3) = VMatrix1(:,:);

elseif Teil_h == 0.5
    YUV(1:heigth,1:width,2) = int16(0);
    YUV(1:2:end,:,2) = UMatrix1(:,:);
    YUV(2:2:end,:,2) = UMatrix1(:,:);

    YUV(1:heigth,1:width,3) = int16(0);
    YUV(1:2:end,:,3) = VMatrix1(:,:);
    YUV(2:2:end,:,3) = VMatrix1(:,:);

elseif Teil_h == 0.25
    YUV(1:heigth,1:width,2) = int16(0);
    YUV(1:4:end,:,2) = UMatrix1(:,:);
    YUV(2:4:end,:,2) = UMatrix1(:,:);
    YUV(3:4:end,:,2) = UMatrix1(:,:);
    YUV(4:4:end,:,2) = UMatrix1(:,:);

    YUV(1:heigth,1:width) = int16(0);
    YUV(1:4:end,:,3) = VMatrix1(:,:);
```

```
    YUV(2:4:end,:,3) = VMatrix1(:,:);
    YUV(3:4:end,:,3) = VMatrix1(:,:);
    YUV(4:4:end,:,3) = VMatrix1(:,:);
end
YUV = uint8(YUV);
fclose(fileId);
end

function
save_yuv(data,video_file,BreiteV,HoeheV,HoehenteilerV,BreitenteilerV)

%get Resolution od Data
datasize = size(data);
datasizelength = length(datasize);

%open File
fid = fopen(video_file,'a');

%subsampling of U and V
if datasizelength == 2 || HoehenteilerV == 0
    %4:0:0
    y(1:HoeheV,1:BreiteV) = data(:,:,1);
elseif datasizelength == 3
    y(1:HoeheV,1:BreiteV,1) = double(data(:,:,1));
    u(1:HoeheV,1:BreiteV,1) = double(data(:,:,2));
    v(1:HoeheV,1:BreiteV,1) = double(data(:,:,3));
    if BreitenteilerV == 1
        %4:1:1
        u2 = u;
        v2 = v;
    elseif HoehenteilerV == 0.5
        %4:2:0
        u2(1:HoeheV/2,1:BreiteV/2) =
u(1:2:end,1:2:end)+u(2:2:end,1:2:end)+u(1:2:end,2:2:end)+u(2:2:end,2:2:end);
        u2                          = u2/4;
        v2(1:HoeheV/2,1:BreiteV/2) =
v(1:2:end,1:2:end)+v(2:2:end,1:2:end)+v(1:2:end,2:2:end)+v(2:2:end,2:2:end);
        v2                          = v2/4;
    elseif BreitenteilerV == 0.25
        %4:1:1
        u2(1:HoeheV,1:BreiteV/4) =
u(:,1:4:end)+u(:,2:4:end)+u(:,3:4:end)+u(:,4:4:end);
        u2                         = u2/4;
        v2(1:HoeheV,1:BreiteV/4) =
v(:,1:4:end)+v(:,2:4:end)+v(:,3:4:end)+v(:,4:4:end);
        v2                         = v2/4;
    elseif BreitenteilerV == 0.5 && HoehenteilerV == 1
        %4:2:2
        u2(1:HoeheV,1:BreiteV/2) = u(:,1:2:end)+u(:,2:2:end);
        u2                         = u2/2;
        v2(1:HoeheV,1:BreiteV/2) = v(:,1:2:end)+v(:,2:2:end);
        v2                         = v2/2;
    end
end

fwrite(fid,uint8(y'),'uchar'); %writes Y-Data

if HoehenteilerV ~= 0
    %writes U- and V-Data if no 4:0:0 format
```

```
    fwrite(fid,uint8(u2'),'uchar');
    fwrite(fid,uint8(v2'),'uchar');
end

fclose(fid);
end
```

P 4.1. Kod *prostorno_smanjenje_rezolucije_videa.m* skripte

```
%Primjer:
%prostorno_smanjenje_rezolucije_videa('mobile_vga.yuv',640,480,'420',
'mobile_vga_downscaled.yuv',320,240,'420');

function prostorno_smanjenje_rezolucije_videa(ulazni_video,ulazni_sirina,
ulazni_visina,ulazni_format,izlazni_video,izlazni_sirina,izlazni_visina,
izlazni_format)

tStart=tic;
[ulazni_faktor_sirina, ulazni_faktor_visina]=dohvati_format(ulazni_format);
[izlazni_faktor_sirina,
izlazni_faktor_visina]=dohvati_format(izlazni_format);

provjera = dir(ulazni_video);
velicina_bitovi = provjera.bytes;
clear provjera

broj_okvira=velicina_bitovi/(ulazni_sirina*ulazni_visina*(1+2*ulazni_faktor_v
isina*ulazni_faktor_sirina));

fclose(fopen(izlazni_video,'w'));
h = waitbar(0,'Molimo pričekajte... ');

for brojac = 1:1:broj_okvira
    waitbar(brojac/broj_okvira,h);

YUV=loadFileYUV(ulazni_sirina,ulazni_visina,brojac,ulazni_video,ulazni_faktor
_visina,ulazni_faktor_sirina);
    rezultantni_YUV = imresize(YUV,[izlazni_visina izlazni_sirina]);

save_yuv(rezultantni_YUV,izlazni_video,izlazni_sirina,izlazni_visina,izlazni_
faktor_visina,izlazni_faktor_sirina);
end
close(h);
vrijeme_obrade=toc(tStart)
end

function [faktor_sirina,faktor_visina] = dohvati_format(format)

faktor_sirina = 0.5;
faktor_visina= 0.5;
if strcmp(format,'400')
    faktor_sirina = 0;
    faktor_visina= 0;
elseif strcmp(format,'411')
    faktor_sirina = 0.25;
    faktor_visina= 1;
elseif strcmp(format,'420')
    faktor_sirina = 0.5;
    faktor_visina= 0.5;
elseif strcmp(format,'422')
    faktor_sirina = 0.5;
    faktor_visina= 1;
elseif strcmp(format,'444')
    faktor_sirina = 1;
```

```
    faktor_visina= 1;
else
    display('Pogrešan format!');
end
end

% read YUV-data from file
function YUV = loadFileYUV(width,heigth,Frame,fileName,Teil_h,Teil_b)

% get size of U and V
fileId = fopen(fileName,'r');
width_h = width*Teil_b;
heigth_h = heigth*Teil_h;
% compute factor for framesize
factor = 1+(Teil_h*Teil_b)*2;
% compute framesize
framesize = width*heigth;

fseek(fileId,(Frame-1)*factor*framesize, 'bof');
% create Y-Matrix
YMatrix = fread(fileId, width * heigth, 'uchar');
YMatrix = int16(reshape(YMatrix,width,heigth)');
% create U- and V- Matrix
if Teil_h == 0
    UMatrix = 0;
    VMatrix = 0;
else
    UMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    UMatrix = int16(UMatrix);
    UMatrix = reshape(UMatrix,width_h, heigth_h).';

    VMatrix = fread(fileId,width_h * heigth_h, 'uchar');
    VMatrix = int16(VMatrix);
    VMatrix = reshape(VMatrix,width_h, heigth_h).';
end
% compose the YUV-matrix:
YUV(1:heigth,1:width,1) = YMatrix;

if Teil_h == 0
    YUV(:,:,2) = 127;
    YUV(:,:,3) = 127;
end
% consideration of the subsampling of U and V
if Teil_b == 1
    UMatrix1(:,:) = UMatrix(:,:);
    VMatrix1(:,:) = VMatrix(:,:);

elseif Teil_b == 0.5
    UMatrix1(1:heigth_h,1:width) = int16(0);
    UMatrix1(1:heigth_h,1:2:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,2:2:end) = UMatrix(:,1:1:end);

    VMatrix1(1:heigth_h,1:width) = int16(0);
    VMatrix1(1:heigth_h,1:2:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,2:2:end) = VMatrix(:,1:1:end);

elseif Teil_b == 0.25
    UMatrix1(1:heigth_h,1:width) = int16(0);
    UMatrix1(1:heigth_h,1:4:end) = UMatrix(:,1:1:end);
```

```
    UMatrix1(1:heigth_h,2:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,3:4:end) = UMatrix(:,1:1:end);
    UMatrix1(1:heigth_h,4:4:end) = UMatrix(:,1:1:end);

    VMatrix1(1:heigth_h,1:width) = int16(0);
    VMatrix1(1:heigth_h,1:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,2:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,3:4:end) = VMatrix(:,1:1:end);
    VMatrix1(1:heigth_h,4:4:end) = VMatrix(:,1:1:end);
end

if Teil_h == 1
    YUV(:,:,2) = UMatrix1(:,:);
    YUV(:,:,3) = VMatrix1(:,:);

elseif Teil_h == 0.5
    YUV(1:heigth,1:width,2) = int16(0);
    YUV(1:2:end,:,2) = UMatrix1(:,:);
    YUV(2:2:end,:,2) = UMatrix1(:,:);

    YUV(1:heigth,1:width,3) = int16(0);
    YUV(1:2:end,:,3) = VMatrix1(:,:);
    YUV(2:2:end,:,3) = VMatrix1(:,:);

elseif Teil_h == 0.25
    YUV(1:heigth,1:width,2) = int16(0);
    YUV(1:4:end,:,2) = UMatrix1(:,:);
    YUV(2:4:end,:,2) = UMatrix1(:,:);
    YUV(3:4:end,:,2) = UMatrix1(:,:);
    YUV(4:4:end,:,2) = UMatrix1(:,:);

    YUV(1:heigth,1:width) = int16(0);
    YUV(1:4:end,:,3) = VMatrix1(:,:);
    YUV(2:4:end,:,3) = VMatrix1(:,:);
    YUV(3:4:end,:,3) = VMatrix1(:,:);
    YUV(4:4:end,:,3) = VMatrix1(:,:);
end
YUV = uint8(YUV);
fclose(fileId);
end
%Save YUV-Data to File
function
save_yuv(data,video_file,BreiteV,HoeheV,HoehenteilerV,BreitenteilerV)
%get Resolution od Data
datasize = size(data);
datasizelength = length(datasize);
%open File
fid = fopen(video_file,'a');
%subsampling of U and V
if datasizelength == 2 | HoehenteilerV == 0
    %4:0:0
    y(1:HoeheV,1:BreiteV) = data(:,:,1);
elseif datasizelength == 3
    y(1:HoeheV,1:BreiteV) = double(data(:,:,1));
    u(1:HoeheV,1:BreiteV) = double(data(:,:,2));
    v(1:HoeheV,1:BreiteV) = double(data(:,:,3));
    if BreitenteilerV == 1
        %4:1:1
        u2 = u;
```

```
        v2 = v;
    elseif HoehenteilerV == 0.5
        %4:2:0
        u2(1:HoeheV/2,1:BreiteV/2) =
u(1:2:end,1:2:end)+u(2:2:end,1:2:end)+u(1:2:end,2:2:end)+u(2:2:end,2:2:end);
        u2                         = u2/4;
        v2(1:HoeheV/2,1:BreiteV/2) =
v(1:2:end,1:2:end)+v(2:2:end,1:2:end)+v(1:2:end,2:2:end)+v(2:2:end,2:2:end);
        v2                         = v2/4;
    elseif BreitenteilerV == 0.25
        %4:1:1
        u2(1:HoeheV,1:BreiteV/4) =
u(:,1:4:end)+u(:,2:4:end)+u(:,3:4:end)+u(:,4:4:end);
        u2                       = u2/4;
        v2(1:HoeheV,1:BreiteV/4) =
v(:,1:4:end)+v(:,2:4:end)+v(:,3:4:end)+v(:,4:4:end);
        v2                       = v2/4;
    elseif BreitenteilerV == 0.5 & HoehenteilerV == 1
        %4:2:2
        u2(1:HoeheV,1:BreiteV/2) = u(:,1:2:end)+u(:,2:2:end);
        u2                       = u2/2;
        v2(1:HoeheV,1:BreiteV/2) = v(:,1:2:end)+v(:,2:2:end);
        v2                       = v2/2;
    end
end

fwrite(fid,uint8(y'),'uchar'); %writes Y-Data

if HoehenteilerV ~= 0
    %writes U- and V-Data if no 4:0:0 format
    fwrite(fid,uint8(u2'),'uchar');
    fwrite(fid,uint8(v2'),'uchar');
end
fclose(fid);
```

P 7.1. DVD – ROM

- aktivnost_videa.m
- prostorno_povecanje_rezolucije_videa.m
- prostorno_smanjenje_rezolucije_videa.m
- video signali
- Algoritmi za povećanje prostorne rezolucije video signala.doc
- Algoritmi za povećanje prostorne rezolucije video signala.docx
- Algoritmi za povećanje prostorne rezolucije video signala.pdf
- Algoritmi za povećanje prostorne rezolucije video signala.pptx
- yuvplayer