Interpolacija slike uz očuvanje rubova i implementacija rješenja na realnu ADAS platformu

Kelava, Božidar

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

Sveučilišni studij

INTERPOLACIJA SLIKE UZ OČUVANJE RUBOVA I IMPLEMENTACIJA RJEŠENJA NA REALNU ADAS PLATFORMU

Diplomski rad

Božidar Kelava

Osijek, 2021.

SADRŽAJ

1. UVOD

Sigurnost vozila je osnovni segment strategije unaprjeđenja sigurnosti u automobilu, s naglaskom na dugoročne ciljeve kojima se nastoji pridonijeti smanjenju broja prometnih nesreća. Tehnologije koje pridonose zaštiti od prometnih nesreća pružaju odlične rezultate u proteklih nekoliko godina, a cilj je doprinijeti smanjenju broja žrtava. Sustav za pomoć vozaču prilikom vožnje ADAS (engl. *Advanced Driver Assistance System*) definira se kao inteligentni sigurnosni sustav za vozila, koji treba unaprijediti razinu sigurnosti na cesti u smislu smanjenja broja prometnih nesreća, ublažavanja posljedica sudara te zaštite nakon sudara. Također, ADAS se može definirati kao integrirani sustav unutar vozila koji se sastoji od senzora, procesorskih jedinica i aktuatora, a služi za pomoć vozaču te povećava sigurnost sa svojim pogodnostima i funkcijama. Danas se koristi širok spektar ADAS tehnologija, a neke su ugrađene u vozila kao standardna oprema, poput podsjetnika na sigurnosni pojas, blokade motora u slučaju pojačane koncentracije alkohola kod vozača, inteligentne prilagodbe brzine, elektroničke kontrole stabilnosti, itd. Prethodno navedene tehnologije dokaz su velikog potencijala ADAS platforme, a zasnovane su na algoritmima za obradu slike, kontroli gibanja vozila, prikupljanju podataka s dostupnih senzora, obradi dobivenih podataka i slično [1].

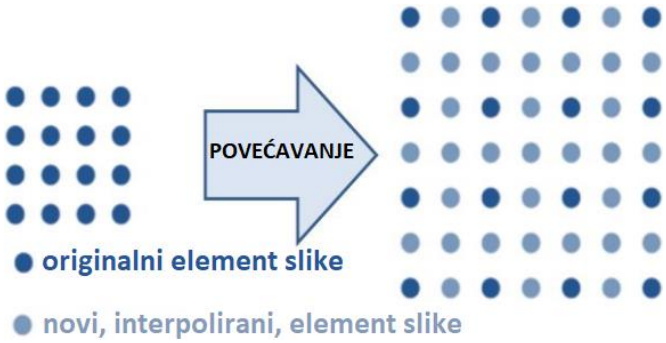
Velik broj ADAS zasnovan je na obradi slika s kamera koje se nalaze u automobilu. Budući da različiti ADAS algoritmi obrađuju slike raznih rezolucija, često je potrebno obavljati interpolaciju slike. Interpolacija slike se odnosi na promjenu broja elemenata digitalne slike upotrebom prostornih 2D interpolacijskih algoritam. Interpolacija [2] u matematici i fizici znači određivanje nepoznatih vrijednosti neke veličine uz pomoć poznatih vrijednosti u nekom intervalu, u kojem su poznate zakonitosti bitnih promjena. U različitim programskim paketima naprednih sustava za pomoć vozaču pri vožnji, zasnovanih na obradi slike s kamere unutar automobila, često je potrebno iz različitih razloga mijenjati rezoluciju slike (povećavati ju ili smanjivati). Prilikom promjene rezolucije, a pri provođenju same interpolacije, poželjno je zadržati kvalitetu slike s promijenjenom rezolucijom što je moguće većom. Promjena rezolucije nije jednoznačnica s riječju interpolacija, ali promjena rezolucije uključuje interpolaciju kao jedan od svojih koraka. Za uspješno izvođenje promjene rezolucije, potrebno je sačuvati rubove objekta u sceni što je moguće oštrijima, kako bi se što je bolje sačuvao sadržaj slike. Poznatije metode za interpolaciju slike su metoda najbližih susjeda, bilinearna interpolacija, bikubična interpolacija [3]. Više informacija o metodama interpolacije biti će dano u nastavku rada.

Zadatak ovog diplomskog rada je implementirati tri netom spomenute postojeće metode za interpolaciju slike na osobnom računalu, a zatim na realnoj ADAS platformi. Potrebno je provesti optimizaciju rada rješenja te analizu performansi implementiranih rješenja iz više pogleda: kvalitete interpolirane slike, brzine izvođenja i memorijskog otiska. Pritom je potrebno koristiti slike različitog sadržaja, s naglaskom na sadržaje kakvi su tipični za slike dobivene s kamera unutar vozila.

U drugom poglavlju bit će detaljnije analiziran problem interpolacije slike, kao i metode za interpolaciju slike, uz pregled postojećih rješenja. U trećem poglavlju bit će opisan način implementacije metoda za interpolaciju slike na osobnom računalu i ugradbenoj automotiv platformi. U četvrtom poglavlju bit će predstavljen način testiranja implementiranih rješenja na odabranom skupu slika. Posljednje, peto poglavlje sadrži zaključke rada.

2. PROBLEM INTERPOLACIJE SLIKE I METODE ZA INTERPOLACIJU SLIKE

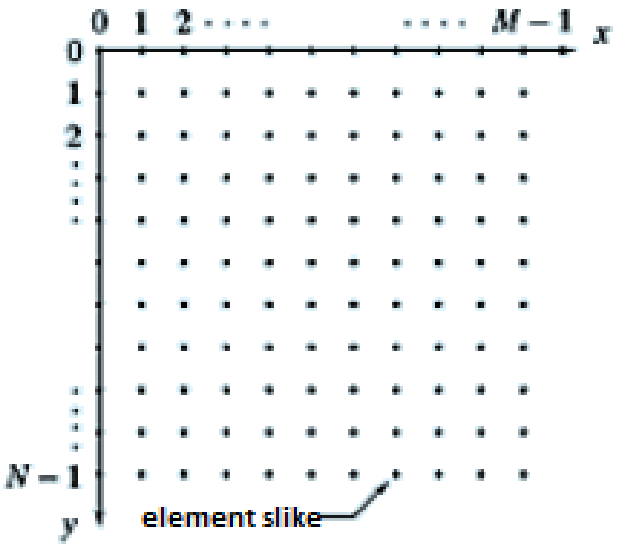
U ovom poglavlju biti će objašnjena interpolacija slike, matematički modeli nekih od metoda za interpolaciju slike te kratki pregled postojećih rješenja za interpolaciju slike. Interpolacija slike je izraz koji se koristi u digitalnoj obradi slike, ali je usko vezan i uz druge termine u literaturi, poput skaliranja slike ili ponovnog uzorkovanja slike (engl. *resampling*) [4]. Metode interpolacije slike su dostupne u mnogim alatima za obradu slike poput Photoshop-a, Photoscape-a, Corel-a. Primjena algoritama za interpolaciju slike se ogleda u postupcima povećavanja slike, smanjivanja slike, podpikselnoj registraciji slike (engl. *subpixel image registration*), dekompoziciji slike, ispravljanju prostornih iskrivljenja na slici i dr. Na slici 2.1. prikazan je osnovni koncept uvećavanja slike koristeći interpolaciju.



Slika 2.1. O*snovni koncept interpolacije kod povećavanje slike s faktorom 2 po visini i širini*

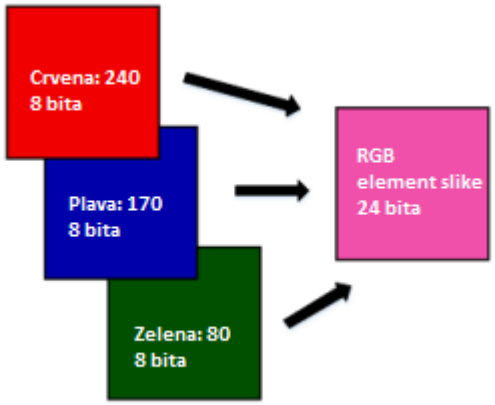
Digitalna slika je predstavljena kao signal, a prostorno varira u dvije dimenzije (visina i širina). Takav signal je dobiven uzorkovanjem i kvantizacijom, kako bi se dobile vrijednosti digitalne slike. Digitalna slika se na računalu predstavlja pomoću polja elemenata slike zapisanih s određenim brojem bita po elementu slike. Postoje dvije osnovne kategorije algoritama za interpolaciju slike: prilagodljivi (eng. *adaptive*) i neprilagodljivi (eng. *non-adaptive*). U ovom diplomskom radu korišteni su neprilagodljivi algoritmi interpolacije slike zbog jednostavnosti njihove primjene i implementacije na realnu ugradbenu platformu [5].

Digitalna slika može se prezentirati matricom veličine M stupaca i N redaka. M predstavlja širinu slike, a N visinu slike. Za određivanje pozicije pojedinog elementa slike definiraju se koordinate x i y, koje predstavljaju pozicije elemenata slike u matrici. Za gornji lijevi element slike se najčešće uzima da ima koordinate (0,0). Bitno je da koordinate elementa slike moraju biti cijeli brojevi. Na slici 2.2. je prikazana matrica koja sadrži elemente slike.



Slika 2.2. *Digitalna slika predstavljena poljem elemenata slike od M stupaca i N redaka*[6]

Rezolucija slike predstavlja veličinu koja karakterizira broj elemenata slike koji ju čine, odnosno govori o dimenzijama matrice elementa slike (M × N). Svaki element slike ima vlastitu vrijednost intenziteta svjetline i/ili vlastite vrijednosti intenziteta boje. Iako je moguć zapis intenziteta svakog elementa slike s različitim brojem bita, intenzitet digitalne slike je određen najčešće s 8 bita za pojedinu komponentu boje. RGB (engl. *Red Green Blue*) sustav boja sa zapisom uz 24 bita (8 bita po komponenti boje) dobiven je adaptivnim miješanjem intenziteta crvene, zelene i plave boje, a jedan primjer miješanja komponenata boje dan je na slici 2.3 [7].



Slika 2.3. *Primjer miješanja boja kod RGB zapisa elemenata slike*

U digitalnoj obradi slike i videa pretežno se koristi YUV model boja. YUV model boja dobiva se jednostavnom transformacijom iz RGB modela. Konverzija iz RGB sustava u YUV sustav provodi

se pomoću formula (2-1), (2-2), (2-3), a konverzija iz YUV sustava u RGB provodi se pomoću formula (2-4), (2-5), (2-6) [8].

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \tag{2-1}$$

$$U = -0{,}147 \cdot R - 0{,}289 \cdot G + 0{,}436 \cdot B \tag{2-2}$$

$$V = 0{,}615 \cdot R - 0{,}515 \cdot G - 0{,}100 \cdot B \tag{2-3}$$

$$R = Y + 1.140 \cdot V \tag{2-4}$$

$$G = Y - 0{,}395 \cdot U - 0{,}581 \cdot V \tag{2-5}$$

$$B = Y + 2{,}032 \cdot U \tag{2-6}$$

2.1. Neprilagodljivi algoritmi za interpolaciju slike

2.1.1. Matematički model interpolacije metodom najbližih susjeda

Metoda najbližih susjeda koristi postupak za izračun interpoliranih vrijednosti kako slijedi. Za razmatranje postupka, koristit će se jedna konkretna matrica **A** dimenzija 3×3:

$$A = \begin{bmatrix} 10 & 4 & 22 \\ 2 & 18 & 7 \\ 9 & 14 & 25 \end{bmatrix}.$$

Matrica promijenjenih dimenzija će u ovom primjeru biti dvostruko veća po širini i po visini, odnosno dimenzija 6×6. Uz poznate veličine redaka i stupaca originalne matrice i matrice promijenjenih dimenzija, moguće je odrediti omjer retka, odnosno omjer stupca prema formuli (2-7). Za matricu **A** omjer retka je jednak omjeru stupca i iznosi 2, a dobiven je na sljedeći način:

$$omjer_{redak/stupac} = \frac{broj\ redaka/stupaca\ matrice\ promijenjenih\ dimenzija}{broj\ redaka/stupaca\ originalne\ matrice} \tag{2-7}$$

Sljedeći korak u postupku interpolacije je normalizirati nove veličine retka i stupca. Primjer normaliziranja redka, odnosno stupca je dan formulom (2-8),

$$Redak/Stupac_{pozicija} = \frac{indeks\ elemenata\ u\frac{retku}{stupcu}matrice\ promijenjenih\ dimenzija}{omjer_{redak/stupac}} = \frac{[1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6]}{2} = [0{,}5 \quad 1 \quad 1{,}5 \quad 2 \quad 2{,}5 \quad 3].$$

(2-8)

Primjenom *ceil*[1] funkcije na rezultatima formule (2-8) dobivaju se vrijednosti dane u formuli (2-9) za redak, a formulom (2-10) za stupac:

$$ceil\left(Redak_{pozicija}\right) = [1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 3], \tag{2-9}$$

$$ceil\left(Stupac_{pozicija}\right) = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{bmatrix}. \tag{2-10}$$

Može se zaključiti da se svaki element slike prvog retka matrice **A** ponavlja dva puta. Isto vrijedi i za stupac:

$$\boldsymbol{B}_{prvi\ redak} = [10 \quad 10 \quad 4 \quad 4 \quad 22 \quad 22],$$

$$\boldsymbol{B}_{prvi\ stupac} = \begin{bmatrix} 10 \\ 10 \\ 2 \\ 2 \\ 9 \\ 9 \end{bmatrix}.$$

Konačna interpolirana matrica **B** predstavlja kompletan rezultat interpolacije matrice **A** metodom najbližih susjeda [9]:

$$B = \begin{bmatrix} 10 & 10 & 4 & 4 & 22 & 22 \\ 10 & 10 & 4 & 4 & 22 & 22 \\ 2 & 2 & 18 & 18 & 7 & 7 \\ 2 & 2 & 18 & 18 & 7 & 7 \\ 9 & 9 & 14 & 14 & 25 & 25 \\ 9 & 9 & 14 & 14 & 25 & 25 \end{bmatrix}.$$

2.1.2. Matematički model bilinearne interpolacije

Osnovna ideja bilinearne interpolacije jest da se provodi linearna interpolacija[2] u dva smjera (u smjeru osi x i u smjeru osi y). Na slici 2.4. prikazana je tehnika bilinearne interpolacije. Iz slike se može zaključiti da za izračun interpolirane vrijednosti moraju biti poznate vrijednosti 4 elementa

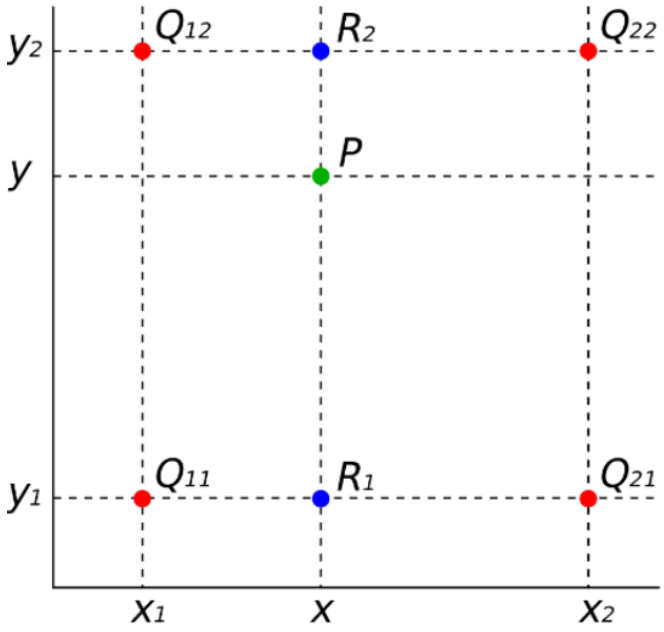
[1] Ceil() funkcija - najmanja cjelobrojna vrijednost koja nije manja od x, za ceil(π) = 4, dostupno na: https://brilliant.org/wiki/ceiling-function/

[2] Linearna interpolacija, metoda interpolacije koja koristi polinome prvog stupnja za izračun novih vrijednosti elemenata slike između poznatih elemenata slike

slike. Za izračun bilinearnom interpolacijom koristi se Lagrangeov polinom[3]. Pretpostavka je da se želi pronaći vrijednost funkcije *f* u točki s koordinatama (*x*,*y*). Prema slici 2.4., ako su poznate vrijednosti četiri točke $Q_{11} = (x_1,y_1)$, $Q_{12} = (x_1,y_2)$, $Q_{21} = (x_2,y_1)$, $Q_{22} = (x_2,y_2)$, moguće je izvršiti interpolaciju u smjeru x uz pomoć Lagrangeovog polinoma prema formulama (2-11) i (2-12).

$$f(x,y_1) \approx \frac{x_2-x}{x_2-x_1} f(Q_{11}) + \frac{x-x_1}{x_2-x_1} f(Q_{21}) \tag{2-11}$$

$$f(x,y_2) \approx \frac{x_2-x}{x_2-x_1} f(Q_{12}) + \frac{x-x_1}{x_2-x_1} f(Q_{22}) \tag{2-12}$$



Slika 2.4. *Bilinearna interpolacija*

Sljedeći korak je interpolacija u smjeru y, kako bi se izračunala tražena vrijednost, a izvod i konačni rezultat dani su u formuli (2-13) [7]:

$$f(x,y) \approx \frac{y_2-y}{y_2-y_1} f(x,y_1) + \frac{y-y_1}{y_2-y_1} f(x,y_2) = \frac{y_2-y}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1} f(Q_{11}) + \frac{x-x_1}{x_2-x_1} f(Q_{21})\right) +$$

$$\frac{y-y_1}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1} f(Q_{12}) + \frac{x-x_1}{x_2-x_1} f(Q_{22})\right) = \frac{1}{(x_2-x_1)(y_2-y_1)} (f(Q_{11})(x_2-x)(y_2-y) + f(Q_{21})(x -$$

[3] Lagrangeov interpolacijski polinom, dostupno na: http://www.grad.hr/nastava/matematika/mat3/node151.html

$$x_1)(y_2 - y_1) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1)) =$$

$$\frac{1}{(x_2 - x_1)(y_2 - y_1)} [x_2 - x \quad x - x_1] \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix} \quad (2\text{-}13)$$

Rezultat interpolacije će biti jednak ukoliko se prvo interpolira u smjeru y, zatim u smjeru x, što znači da rezultati neće ovisiti o redoslijedu smjerova interpolacije.

2.1.3. Matematički model bikubične interpolacije

Bikubična interpolacija predstavlja provođenje kubične interpolacije[4] u dvije dimenzije, odnosno dva smjera, smjeru *x* i smjeru *y*. Bikubčina interpolacija se koristi kada brzina obrade slike nije ključna, a koristi 16 elemenata slike, za razliku od bilinearne koja koristi 4 elementa slike. Poput bilinearne interpolacije, bikubična interpolacija se izračunava iz Lagrangeovih polinoma. Postupak je vrlo sličan postupku bilinearne interpolacije, a dan je formulom (2-16) [7]:

$$f(x, y_i) \approx f(x_0, y_i) \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} + f(x_1, y_i) \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} +$$

$$f(x_2, y_i) \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} + f(x_3, y_i) \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} \quad (2\text{-}16)$$

gdje je i = 0, 1, 2 i 3, indeks najbližeg elementa slike. Nakon interpolacije u smjeru x, interpolira se u smjeru y, kako je dano formulom (2-17)

$$f(x, y) \approx f(x, y_0) \frac{(y - y_1)(y - y_2)(y - y_3)}{(y_0 - y_1)(y_0 - y_2)(y_0 - y_3)} + f(x, y_1) \frac{(y - y_0)(y - y_2)(y - y_3)}{(y_1 - y_0)(y_1 - y_2)(y_1 - y_3)} +$$

$$f(x, y_2) \frac{(y - y_0)(y - y_1)(y - y_3)}{(y_2 - y_0)(y_2 - y_1)(y_2 - y_3)} + f(x, y_3) \frac{(y - y_0)(y - y_1)(y - y_2)}{(y_3 - y_0)(y_3 - y_1)(y_3 - y_2)} \quad (2\text{-}17)$$

Nije bitan smjer u kojem se prvo interpolira, kao i kod bilinearne interpolacije [10].

2.2. Pregled relevantne literature

U radu [11] autori su predstavili učinkovit način interpolacije slike uz očuvanje rubova, koji se zasniva na inverznoj težini gradijenta uz lokaciju elementa slike za interpolaciju. Predloženo rješenje u radu daje bolje rezultate u smislu kvalitete u odnosu na konvencionalne [12] algoritme,

[4] Kubična interpolacija koristi polinome trećeg stupnja za izračun vrijednosti između točaka x_1 i x_2, uz poznate vrijednosti $f(x)$ i derivacija u x_1 i x_2

a također korišteni pristup je brži (učinkovitiji na računalu – korišteno je osobno računalo niske do osrednje kvalitete specifikacija) u odnosu na napredne algoritme poput CAI (engl. *Content Adaptive Interpolation*, sadržajno prilagodljiva interpolacija), CBID (engl. *Context-Based Image Dependent*, interpolacija slike u ovisnosti o kontekstu), CBII (engl. *Context-Based Image Independent Interpolation*, interpolacija slike neovisna o kontekstu) za interpolaciju slike uz očuvanje rubova. Korištena je baza od 10 slika, slike su veličine 512×512 elemenata slike, te sve slike su u sivoj boji. Za dobivanje objektivnog mjerila slike korištena je veličina PSNR (engl. *Peak Signal-to-Noise Ratio*, odnos vršne snage signala i šuma). Uz predloženo rješenje interpolacije slike, implementirano je još pet metoda interpolacije: bilinearna interpolacija, bikubična interpolacija, CAI, CBID i CBII. PSNR se računao po izrazu (2-18), gdje je *MSE(O, I)* (engl. *Mean Squared Error*, srednja kvadratna pogreška), O predstavlja izvornu sliku, a I interpoliranu sliku koja je dobivena smanjivanjem rezolucije (engl. *downsampling*). Rezultati predloženog rješenja se ne podudaraju s bilinearnom i bikubičnom interpolacijom, ali su bolji u smislu računalne složenosti, a to znači da se štedi vrijeme i resursi. Predloženi algoritam poboljšava kvalitetu slike s očuvanjem finih rubova, a također je i računski učinkovitiji u odnosu na neke kompleksnije metode.

$$PSNR(O,I) = \frac{255^2}{MSE(O,I)} \tag{2-18}$$

U radu [13] je predstavljen algoritam za interpolaciju slike za primjenu u digitalnim kamerama, a daje prirodnije interpolirane slike u odnosu na slike interpolirane konvencionalnim metodama. Slike koje su rezultat ovog algoritma su glađe i prirodnije, ne stvaraju se artefakti blokiranja i time čuvaju oštrinu i originalnost ruba. Slike se mogu povećati do pet puta po visini i po širini uz očuvanje kvalitete i zadovoljavajuće oštrine. Rješenje se zasniva diskretnoj kosinusnoj transformaciji (DCT). DCT zahtjeva veliku količinu proračuna, dok predloženo rješenja zahtjeva samo dva od šesnaest DCT koeficijenata. Navedena dva koeficijenta mogu se dobiti korištenjem konvolucije na način da se konvolucija realizira korištenjem brze Furierove transformacije, a zatim se koristi reverzni postupak brze Furierove transformacije bez korištenja DCT-a. Rješenje ovog algoritma je testirano u programskom paketu MATLAB. Algoritam je ograničen u smislu da je potrebno odrediti optimalne parametre poput razine praga (engl. *threshold level*) za određivanje potrebnih dvaju koeficijenata DCT-a.

U radu [14] predložen je algoritam za interpolaciju prirodnih slika s tendencijom očuvanja rubova (usmjerena na očuvanje rubova). Osnovna ideja je procjena koeficijenata lokalne kovarijance na slikama manje rezolucije, zatim korištenje kovarijance za prilagodbu interpolacije

na većim rezolucijama zasnovane na geometrijskoj dualnosti između manje rezolucijske kovarijance i veće rezolucijske kovarijance. Svojstvo usmjerenosti na rubove kovarijanci služi za podešavanje koeficijenata interpolacije, kako bi odgovarali proizvoljno orijentiranom rubu. Hibridni pristup izmjena između bilinearne interpolacije i interpolacije usmjerene na rubove je predložen kako bi se smanjila računalna složenost. Dvije važne primjene predloženog algoritma su: poboljšanje razlučivosti monokromatske slike i rekonstrukcija slika u boji iz CCD uzoraka. U obje primjene interpolacija usmjerena na rubove pokazuje značajna poboljšanja u odnosu na linearnu interpolaciju u vizualnoj kvaliteti interpoliranih slika. Rješenje je testirano u programskom paketu MATLAB. Predloženi algoritam je uspoređen s bilinearnom interpolacijom i bikubičnom interpolacijom. Korišten je set od četiri slike veličine 768×512 elemenata slike kojima se smanjivala rezolucija dva puta po širini i visini dva puta (384×256 elemenata slike) direktnim smanjivanjem uzoraka (engl. *direct downsampling*) te je time u slike uveden aliasing. Dobivene slike su se interpolirane bilinearnom interpolacijom, bikubičnom interpolacijom i predloženim rješenjem na izvornu veličinu. Predloženo rješenje je uklonilo artefakte zvonjave (engl. *ringing artifacts*), a što se tiče vremenske složenosti za izvođenje zahtjeva od 5 do 10 sekundi.

Autori rada [15] su predstavili novu shemu prilagodljive interpolacije zasnovanu na neuronskim mrežama, usmjerenu na ljudski vizualni sustav (engl. *Human Visual System HVS*). Predložen je nejasan (engl. *fuzzy*) sustav odlučivanja napravljen iz HSV-a, klasificira elemente slike iz ulazne slike s obzirom na ljudsku percepciju na osjećajnu klasu i neosjećajnu klasu. Koristi se bilinearna interpolacija za interpoliranje neosjećajnih dijelova, a neuronska mreža za interpoliranje osjećajnih dijelova uz rubove. Kao referenca je korišteno 6 slika s manje i više detalja, kao mjera točnosti se koristio PSNR. PSNR u svojoj formuli sadrži MSE. MSE se računao u odnosu na originalnu sliku i interpoliranu sliku. Rezultati predloženog rješenja su uspoređeni s dvije konvencionalne metode interpolacije: bilinearna interpolacija i bikubična interpolacija; i tri moderne metode interpolacije: metoda interpolacija zasnovana na neuronskoj mreži (engl. *Neural network based interpolation method*), metoda interpolacije usmjerena na rubove (engl. *Edge directed interpolation*) i adaptivna kvadratna interpolacija (engl. *Adaptively quadratic interpolation AQua*). Digitalne slike visoke rezolucije zajedno s algoritmima za učenje korištene su za automatsko treniranje predložene neuronske mreže. Rezultati simulacije pokazali su da predloženi algoritam za poboljšanje rezolucije može proizvesti bolju vizualnu kvalitetu na interpoliranim slikama.

Autori u radu [16] daju naglasak da je glavni izazov interpolacije slike očuvanje prostornih detalja. Predlažu tehniku interpolacije zasnovanu na mekom odlučivanju, koja procjenjuje elemente slike koji nedostaju u cjelini, a ne jedan po jedan. Tehnika uči te se prilagođava različitim strukturama u sceni korištenjem 2D modela regresijske analize. Parametri modela su pretpostavljeni iz pokretnog prozora ulazne slike manje rezolucije. Struktura elementa slike je određena naučenim modelom uz proces mekog odlučivanja na bloku elemenata slike, uključuje promatrane elemente slike i procijenjene elemente slike. Rezultat je ekvivalentan rezultatu primjene adaptivnog nerazdvojivog 2D interpolacijskog filtra visokog reda. Nova interpolacija slike čuva prostornu koherenciju interpoliranih slika bolje od postojećih metoda i proizvodi najbolje rezultate na velikom spektru slika u smislu PSNR-a i subjektivne kvalitete slike. Rezultati su uspoređeni na velikom broju slika s bikubičnom interpolacijom, lokalizacijom ruba podpiksela (engl. *subpixel edge localization*), interpolacijom usmjerenom na rub (engl. *edge-directed interpolation*) i spojenom dvosmjernom interpolacijom (engl. *fused bidirectional interpolation*). Prosječni PSRN iznos 30 dB. Rubovi i teksture su dobro očuvani, a interpolacijski artefakti poput zamagljenja, nazubljenja i smanjene oštrine su uvelike smanjeni.

U razrađenim znanstvenim radovima u prethodnim odlomcima autori su nastojali predložiti nova rješenja za interpolaciju slike uz smanjivanje količine resursa potrebnih za izvođenje interpolacije te smanjivanje vremena za izvođenje interpolacije. Glavni cilj svakog od tih radova bio je što bolje očuvati rubove na slikama i postići što je moguće veću subjektivnu kvalitetu interpoliranih slika. Rješenja radova su hibridna, odnosno koriste pretežito neprilagodljive metode interpolacije slike te ih pokušavaju kombinirati na različite načine i prilagoditi korištenjem različitih metoda. Međutim, svi spomenuti radovi analizirali su implementaciju i primjenu tih rješenja u PC okruženju, koje se znatno razlikuje u pogledu dostupnih resursa u odnosu ne nekakvu namjensku ugradbenu platformu. U sklopu ovog diplomskog rada ideja je implementirati nekoliko različitih metoda za interpolaciju slike na namjensku ADAS platformu za primjenu u automotiv aplikacijama. Pritom je, naravno, potrebno voditi računa o ograničenim dostupnim resursima te o tome kako određene operacije najbolje raspodijeliti na različite procesore dostupne platforme, kako bi se zadaci što je moguće više paralelizirali te kako bi se na taj način ubrzao postupak interpolacije na dostupnoj platformi. Uvidom u znanstvene baze podataka, pronađen je rad [17] koji opisuje implementaciju bilinearne interpolacije i bikubične interpolacije na FPGA platformu koja koristi Xilinx mikročip i ImpulseC podskup C programskog jezika namijenjenu za paralelno programiranje, dok su algoritmi implementirani u VHDL programskom jeziku.

U sljedećem poglavlju bit će predstavljena implementacija triju neprilagodljivih metoda za interpolaciju slike na realnu ADAS platformu te način optimizacije rada implementiranih rješenja s ciljem postizanja što je moguće veće brzine obrade.

3. IMPELEMENTACIJA METODA ZA INTERPOLACIJU SLIKE NA UGRADBENU AUTOMOTIV PLATFORMU

U ovom poglavlju će biti detaljno pojašnjeno rješenje zadaka diplomskog rada, odnosno načini implementacije različitih algoritama interpolacije slike neprilagodljivim tehnikama na realnu ugradbenu ADAS platformu ograničenih resursa. U realnim sustavima za pomoć vozaču prilikom vožnje potrebno je iz različitih razloga ponekad mijenjati rezoluciju slike (poput detekcije objekta u dijelu slike koji je slabije vidljiv te se određenom interpolacijom može skalirati ili prikaza slike na zaslonima različitih dimenzija u automobilu). Pritom je bitno je sačuvati rubove na sceni što oštrijima kako bi slika bila što kvalitetnija za sljedeće korake obrade te kako bi objekti i dalje bili jasno vidljivi u odnosu na pozadinu, budući da ADAS nastoji detektirati objekte od interesa. Rješenje je realizirano korištenjem matematičkih modela za interpolaciju metodom najbližih susjeda, bilinearnom interpolacijom i bikubičnom interpolacijom (opisanih u drugom poglavlju). U nastavku ovog poglavlja diplomskog rada bit će detaljno objašnjen postupak izrade rješenja za svaku od triju nabrojanih metoda interpolacije.

3.1. Referentna implementacija metoda za interpolaciju na PC-u kao polazišna točka

Osnovna ideja za implementaciju metoda za interpolaciju slike je zasnovana na OpenCV biblioteci (engl. *Open Source Computer Vision Library*). OpenCV je biblioteka za programske jezike Python, C++, Java i MATLAB, koja je otvorenog izvora (engl. *open source*), a pruža programsku podršku za računalni vid i strojno učenje, izgrađena je kako bi pružila zajedničku infrastrukturu za testiranje mogućih rješenja prije komercijalizacije proizvoda [13]. OpenCV se nije koristio za usporedbu budući da izvorni kôd (engl. *source code*) navedene biblioteke za računalni vid previše složen i nezgodan za čitanje i prevođenje u programski jezik C. Umjesto OpenCV-a, za izvorni kôd interpolacije metodom najbližih susjeda se koristila implementacija u MATLAB-u prema članku [18], za bilinearnu interpolaciju prema članku [19] i za bikubičnu interpolaciju prema članku [20]. Konačno programsko rješenje ovog diplomskog rada je u cijelosti napisano u programskom jeziku C. Zašto je programski jezik C pogodan za realnu ugradbenu ADAS platformu bit će objašnjeno u sljedećem potpoglavlju.

3.2. Programski jezik C

Programski jezik C je računalni programski jezik opće namjene koji se može koristiti za aplikacije širokog spektra. Operacijski sustavi, aplikacijski softveri u rasponu od superračunala do ugradbenih sustava napisani su u programskom jeziku C. Ugradbeni sustav (engl. *embedded system*) je računalni sustav sa specifičnom funkcijom unutar većeg sustava. U današnje vrijeme ugradbeni sustavi se nalaze svugdje, od prenosivih uređaja do brzih transportnih sustava. Ugradbeni sustav praktički se može definirati kao prilagođeni softver koji radi na prilagođenom hardveru. To znači da je hardver vrlo specifičan, točnije ograničen resursima, odnosno manipulativni prostor razvojnog inženjera vrlo je ograničen. Benefiti programskog jezika C za ugradbene sustave dani su u tablici 3.1. [21].

Tablica 3.1. *Prednosti programskog jezika C za korištenje u ugradbenim računalnim sustavima*

BENEFIT	POJAŠNJENJE
NEOVISNOST O PROCESORU	C jezik nije specifičan za bilo koji mikroprocesor ili mikrokontroler, odnosno za bilo koji sustav. Radi na raznim hardverskim konfiguracijama i ne zahtjeva isti skup hardvera za pokretanje programa. Platformski je neovisan.
PRENOSIVOST	Prenosivost je važan čimbenik kada se aplikacija mora premjestiti na drugi operacijski sustav. C se može kompajlirati (engl. *compil*e) na raznim platformama s minimalnim izmjenama. C je učinkovit, jednostavan za razumijevanje, održavanje i uklanjanje eventualnih pogrešaka.
VISOKE PERFORMANSE U ODNOSU NA OSTALE JEZIKE	C programski kôd se kompajlira u binarnu izvršnu datoteku koja se izravno učitava u memoriju i izvršava. C pruža optimizirane upute za uređaj za zadani ulaz, čime se povećavaju performanse ugradbenog sustava. Većina ostalih jezika se oslanja na biblioteke za koje je potreban enormna količina memorije, što je glavni izazov u ugradbenim sustavima.
MANIPULACIJA BITOVIMA	C je strukturni jezik koji pruža manipulaciju podacima na razini bita, koristi bitne (engl. *bitwise*) operatore. Manipulacija bita je izrazito bitna u ugradbenim sustavima, budući da pruža moćne funkcionalnosti poput upravljanja registrima ili upravljanja spremnicima podataka (engl. *buffer*).

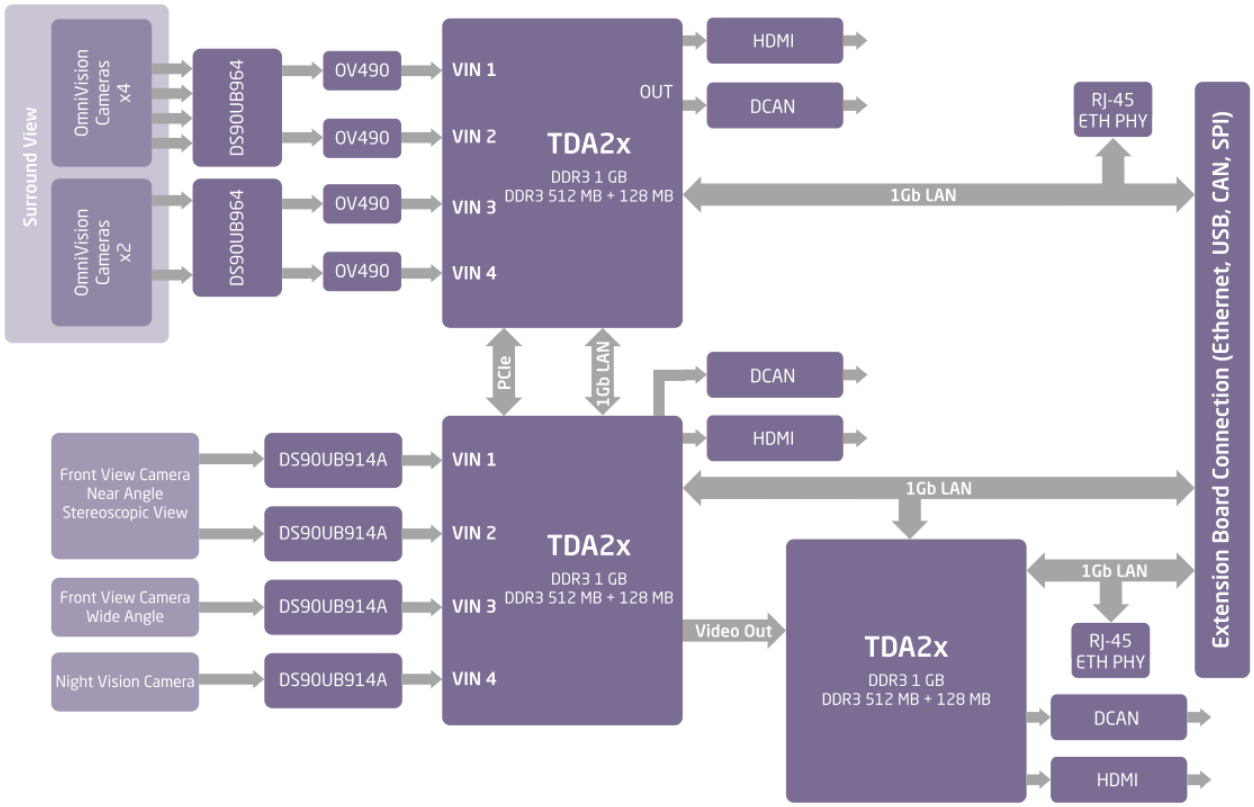
UPRAVLJANJE MEMORIJOM	C pruža izravnu kontrolu memorije koristeći pokazivače (engl. *pointers*). Pomoću pokazivača se izravno pristupa memoriji te se pomoću istih izvode različite operacije. U slučaju da zahtjev za memorijom u sustavu nije poznat, pomoću pokazivača se može memorija dinamički dodjeljivati. Ovo svojstvo programskom jezika C je glavni razlog zašto je najpoželjniji jezik za ugradbene sustave.

3.3. Opis ADAS razvojne ploče i razvojnog okruženja VisionSDK

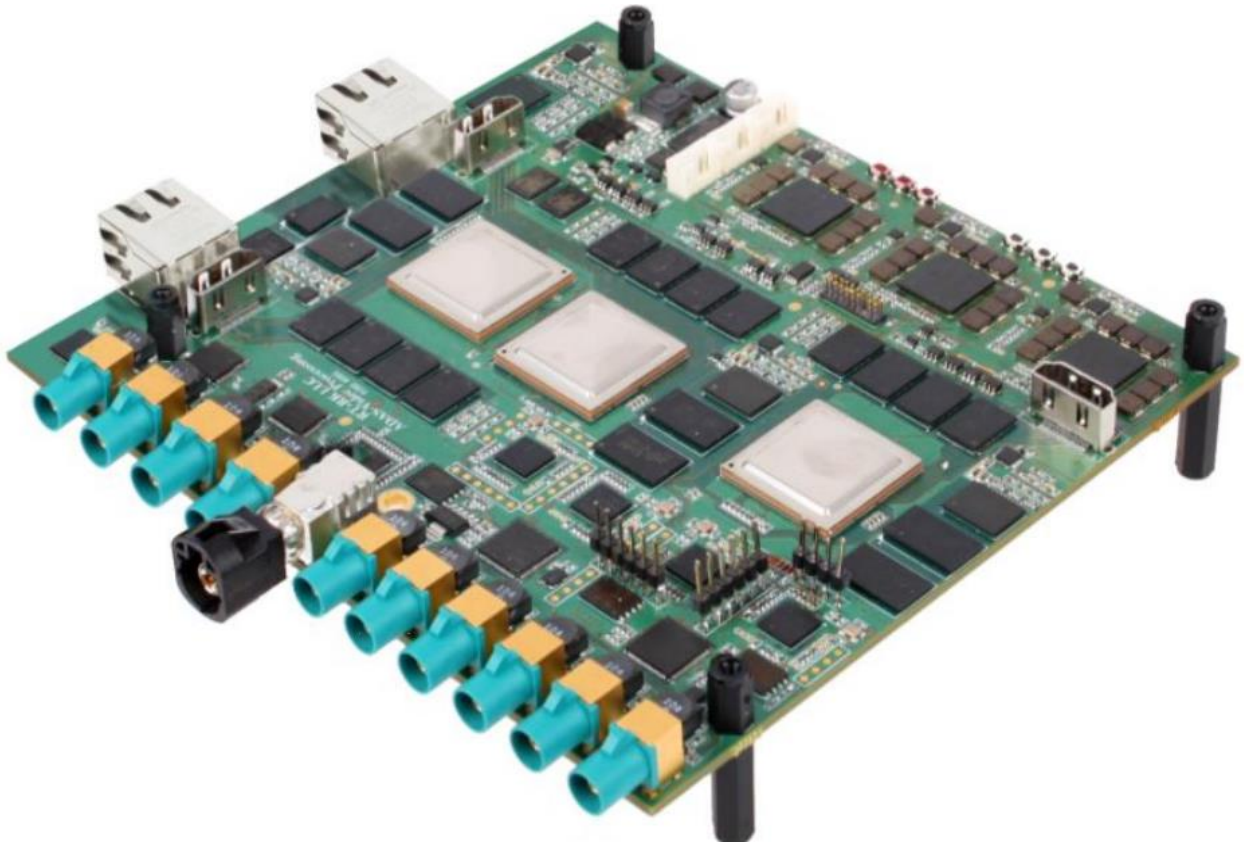
Alpha ploča (engl. *Alpha Board*) je ADAS razvojna ploča namijenjena za učenje i razvoj automotiv aplikacija, dizajnirana je od strane RT-RK Instituta. Alpha ploča podržava osnovne i napredne sustave upozorenja, poluautomatske operacije i aktivne sustave upravljanja. Skup ciljanih automotiv aplikacija pokriva različite upravljačke programe, odnosno algoritme za ciljane aplikacije u vozilu, poput: pogleda od naprijed, pogleda u okružju, zrcaljenja kamere, nadzora vozača, noćnog vida, a sve radi boljih značajki udobnosti i cjelokupnog iskustva vožnje. Hardverske specifikacije Alpha ploče nalaze se u tablici 3.1., blok dijagram arhitekture ploče dan je na slici 3.2., a fizički izgled ploče dan je na slici 3.2 [22].

Tablica 3.2. *Glavne specifikacije Alpha ploče*

R. BR.	SPECIFIKACIJA
1.	Procesori: 3 SoC-a (SCV, FFN, FUS), na svakom su dostupni procesori: 2×A15, 2×M4, 2×DSP, 4×EVE procesora
2.	Dostupno je balansiranje opterećenja algoritama između TDA2xx SoC-eva za obradu videozapisa, što omogućava optimalno radno opterećenje.
3.	Svaki SoC ima vlastita HDMI, DCAN, Ethernet, JTAG, UART sučelja te utor za Micro SD.
4.	Sustav podržava istovremeno spajanje do 10 kamera.
5.	Podržava AVB (engl. *Audio Video Bridging*) Ethernet.
6.	Sustav je moguće nadograditi putem priključka na matičnoj ploči



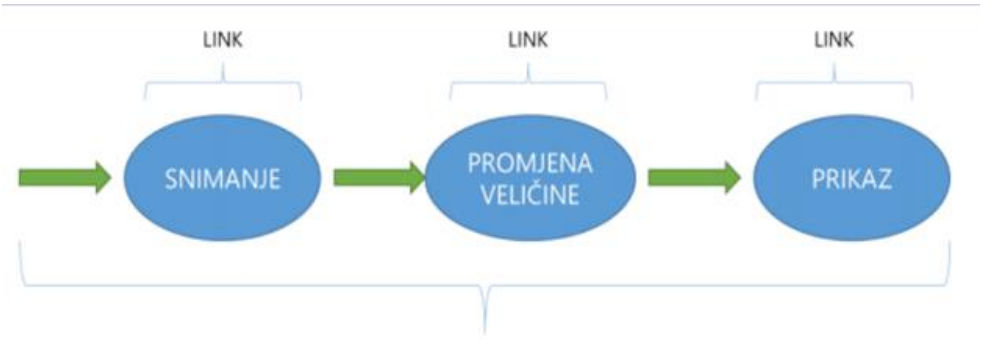
Slika 3.1. *Blok dijagram (arhitektura) Alpha ploče* [22]



Slika 3.2. *Alpha ploča korištena u ovom diplomskom radu* [22]

VisionSDK (engl. *Vision Software Development Kit*) je programska podrška za rad s Alpha pločom. Institut RT-RK je razvio dodatak (engl. *patch*) kako bi Alpha ploča bila kompatibilna s programskim paketom VisionSDK. VisionSDK je višeprocesorski programski okvir (engl. *framework*) namijenjen za TI (engl. *Texas Instruments*) porodicu ADAS SoC-ova (engl. *System on Chip*), a razvijen je također od tvrtke Texas Instruments. VisionSDK omogućava korisnicima kreiranje različitih aplikacije za ADAS ploču, a uključuje podršku za snimanje videa, predobradu videa, algoritme vezane za analizu video toka, prikaz video zapisa na ekranu, itd. Paket dostupan pri izradi ovog diplomskog rada uključuje generičke algoritme specifične za brojne aplikacije u ADAS [16]. SoC kombinira elektroničke sklopove različitih računalnih komponenata u jedan jedinstveni čip. Komponente mogu uključivati grafičku procesorsku jedinicu (GPU), središnju procesorsku jedinicu (CPU) i sistemsku memoriju (RAM). Budući da SoC uključuje i hardver i softver, sam po sebi troši manje energije te ima bolje performanse, zahtjeva manje fizičkog prostora i pouzdaniji je od sustava s više različitih čipova [23] te je zbog toga idealan za realne ugradbene ADAS platforme.

Korisnik pomoću VSDK (skraćeno od VisionSDK) stvara tokove podataka za slučaj upotrebe (engl. *Use Case*). VSDK verzija namijenjena za korištenje na Alpha ploči dolazi s različitim primjerima programskih tokova koji prezentiraju korištenje SoC-ova. VSDK je zasnovan na okviru koji se naziva „veze i lanci“ (engl. *Links and Chains*), čija je ilustracija dostupna na slici 3.3., a korisničko programsko sučelje (engl. *Application Programming Interface API*) se naziva Link API. Programski paket sadržava sve potrebne komponente za kompilaciju aplikacije (BIOS, kodeci, alati za kodiranje, jezgre algoritama, stogovi umrežavanja i slično). Nakon uspješne izgradnje rješenja, stvaraju se dvije datoteke: *AppImage* i *MLO*. Datoteke se pohranjuju na microSD karticu, koja se postavlja u za to predviđeni utor željenog SoC-a, sve u svrhu pokretanja rješenja na razvojnoj ploči. VSDK je napisan u programskom jeziku C, pa se tako i programsko rješenje realizira u programskom jeziku C.

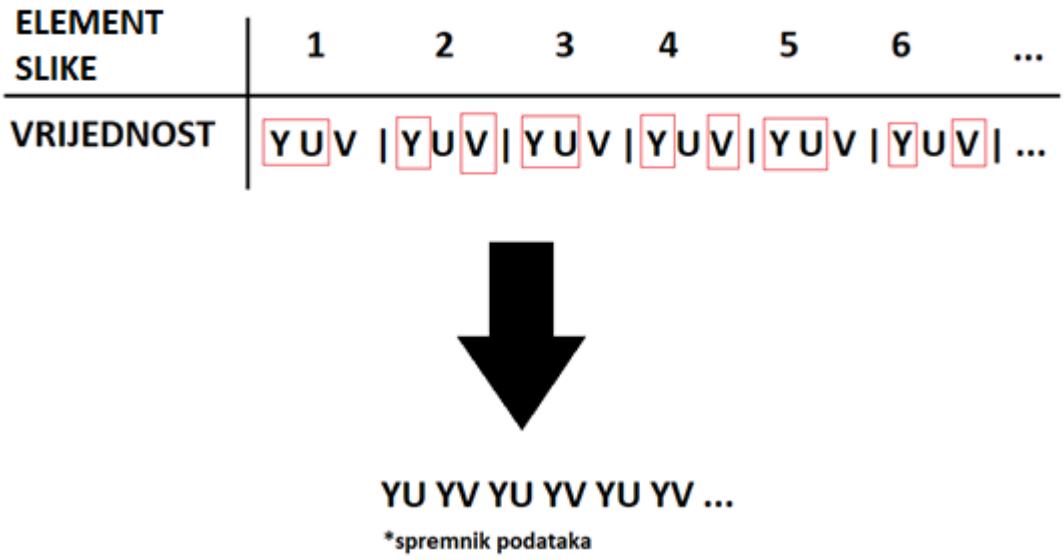


Slika 3.3. *Ilustracija okvira VSDK „Veze i lanci“* [24]

3.4. Implementacija metode najbližih susjeda na Alpha ploču

Programsko rješenje za interpolaciju slike metodom najbližih susjeda je implementirano unutar jednog postojećeg slučaja upotrebe. Ulazna slika je definirana svojom širinom i visinom (širina × visina). Budući da korišteni slučaj upotrebe koristi fiksno definirane ulazne veličine slike unutar svoje glavne funkcije, ulazne slike nužno moraju biti veličine 1280×720. Veličinu je moguće promijeniti unutar glavne funkcije, no budući da VSDK koristi isti spremnik podataka za ulaz i izlaz iz algoritma, veličina spremnika podataka je postavljena na maksimalno moguću. Zbog prethodne činjenice, može se konstatirati da je i ulazni spremnik fiksne veličine, koja ovisi o ulaznoj veličini slike i formatu poduzorkovanja. Korištena shema poduzorkovanja je YUV 4:2:2 (slika 3.4.) zapisana u formatu YUYV (slika 3.5.).

Slika 3.4. *Shema poduzorkovanja YUV 4:2:2*



Slika 3.5. *Shema poduzorkovanja 4:2:2 zapisana u formatu YUYV*

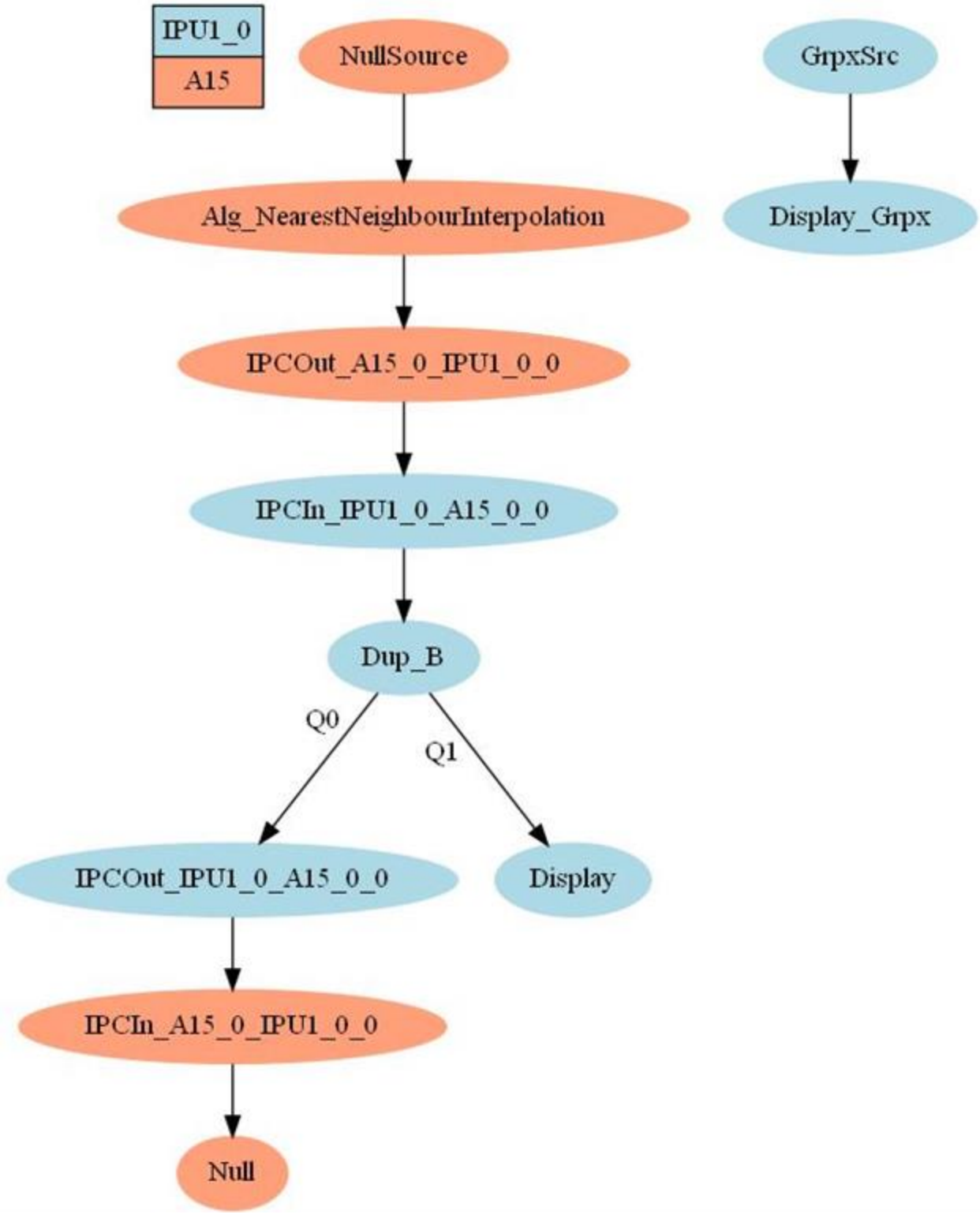
Ulazna veličina spremnika podataka je jednaka umnošku visine, širine i formata sheme poduzorkovanja i iznosi 1843200 (1280×720×2) elemenata slike. Uslijed prethodno navedenih ograničenja, a radi potreba testiranja koja će biti predstavljena u četvrtom poglavlju, ulazna slika je morala biti prilagođena unutar ulaznog spremnika podataka te je veličine 640×360 elemenata slike, a nalazi se unutar slike 1280×720. Primjer ulazne slike se nalazi na slici 3.6.

Slika 3.6. *Primjer ulazne slike u algoritme za interpolaciju*

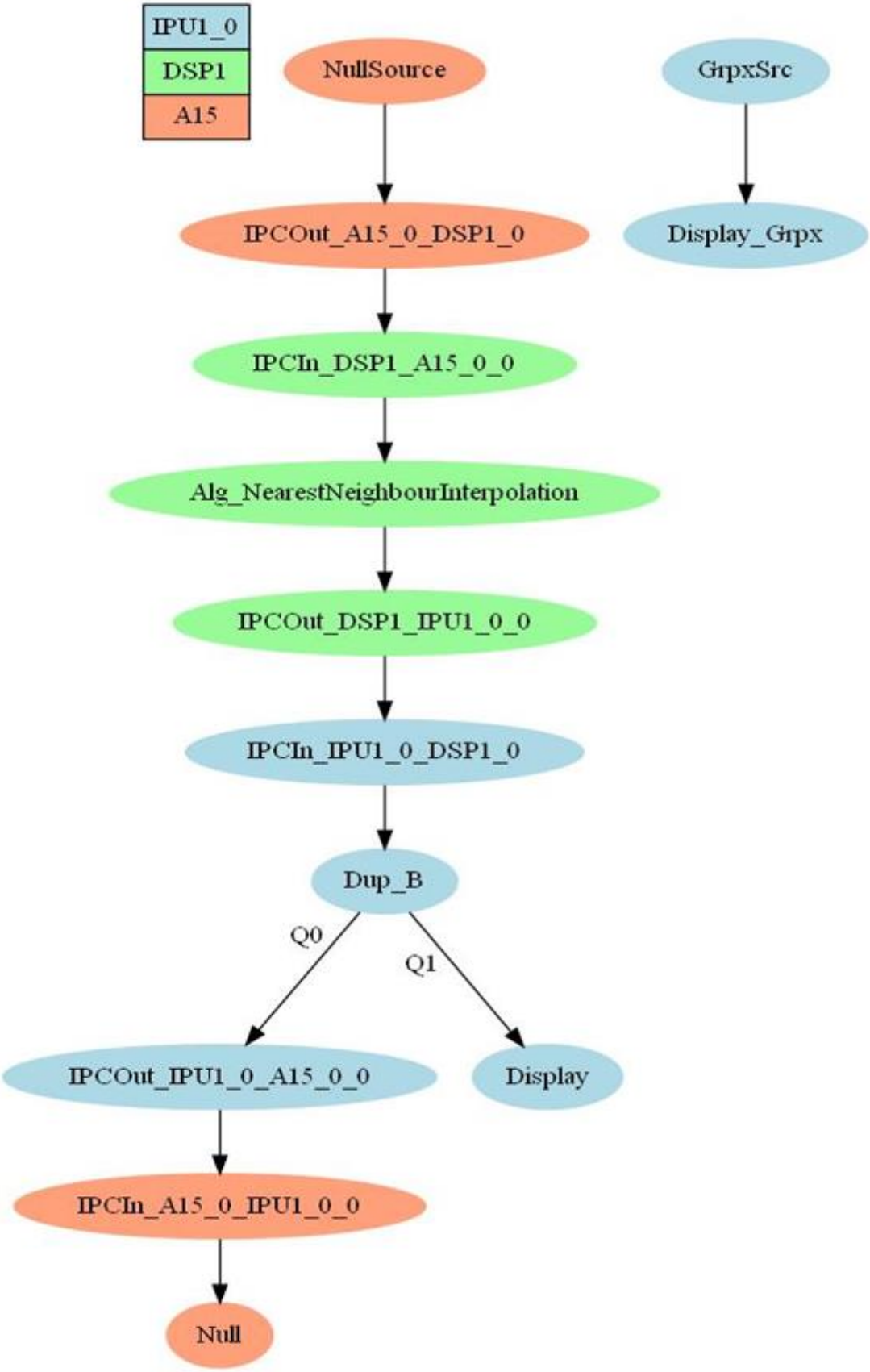
Zbog specifičnosti zadatka diplomskog rada, nužno je prilagoditi algoritam i slučaj upotrebe za svaki procesor ili istovremeno korištenje procesora pojedinačno, što će biti objašnjeno u sljedećim potpogavljima.

3.4.1. Implementacija interpolacije slike metodom najbližih susjeda koristeći zasebno procesore A15 i DSP1/DSP2

U ovom potpoglavlju biti će opisan slučaj upotrebe za implementaciju interpolacije slike metodom najbližih susjeda korištenjem procesora A15 i DSP i to zasebno. Implementacija na različite procesore je potrebna radi bržeg vremena izvođenja pojedinog algoritma. Definirani slučaj za provođenje interpolacije metodom najbližih susjeda na procesoru A15 vidljiv je na slici 3.7, a slučaj upotrebe za provođenje istog algoritma na procesoru DSP1 (postoje dva DSP-a na svakom SoC-u, tj. DSP1 i DSP2) je vidljiv na slici 3.8.



Slika 3.7. *Dijagram toka slučaja upotrebe za izvođenje algoritma za metodu najbližih susjeda na procesoru A15*



Slika 3.8. *Dijagram toka slučaja upotrebe za izvođenje algoritma za metodu najbližih susjeda na procesoru DSP1*

Slučaj upotrebe koristi nekoliko povezanih veza:

1. *NullSource (A15)* – koristi se za ulaznu vezu unutar slučaja upotrebe. Broj izlaznih veza može biti do 4. Spremnici podataka ove veze se šalju na izlazne redove na način da se prethodno čitaju iz memorije ili datoteke, ili se primaju preko mreže putem Ethernet kabla, Provodi se na procesoru A15.
2. *Null (A15)* – koristi se za izlaznu vezu iz slučaja upotrebe. Za ulaz koristi spremnik podataka prethodne veze, dok je broj ulaznih veza koje može primiti do 4. Koristi se kada se ne radi ništa bitno s ulaznim spremnikom, ili prilikom zapisivanja ulaznog spremnika podataka u datoteku, ili za kopiranje spremnika podataka u memoriju. Provodi se na procesoru A15.
3. *Dup* – koristi se za multipliciranje ulaza na 2 do 6 izlaza, može se koristiti za paralelizaciju algoritama.
4. *Display* – koristi se za prikaz izlaza na ekranu.
5. *Algoritamska veza* – predstavlja algoritam za obradu ulaznog signala, količina ulaznih i izlaznih stavki je definirana unutar algoritma.

Algoritam za metodu najbližih susjeda prati matematički model metode najbližih susjeda opisan u dijelu 2.1.1. te je jednak za izvođenje na procesorima DSP1 i A15. Za početak se alociraju dva spremnika podataka, prvi je veličine 640×360 te će u daljnjem tekstu će biti označen kao *yuv* , a drugi je veličine 1280×720 te će u daljnjem tekstu biti označen kao *matrix*. Za provođenje interpolacije nužan je format YUV 4:4:4, odnosno za svaki element slike moraju biti dostupne sve tri komponente, tj. Y, U i V. Iz ulaznog toka podataka, koji se nalazi unutar varijable *inputPtr*, učitavaju se komponente boje u spremnik podataka *matrix*, zatim se iz spremnika *matrix* učitava efektivni dio slike za interpolaciju u spremnik *yuv*. Programski kôd je dostupan na slici 3.9. Identičan postupak za učitavanje slike se koristi u svim metodama interpolacije koje će kasnije biti spomenute.

```
1.   int ct0 = 0, ct1 = 1, ct2 = 2, ct3 = 3;
2.   for (i = 0; i < h; i++)
3.   {
4.     for (j = 0; j < w; j = j + 2)
5.     {
6.       matrix[0][i * w + j] = inputPtr[ct0];
7.       matrix[1][i * w + j] = inputPtr[ct1];
8.       matrix[2][i * w + j] = inputPtr[ct3];
9.
10.           matrix[0][i * w + j + 1] = inputPtr[ct2];
11.     //...
```

```
10.     //...
11.           matrix[1][i * w + j + 1] = inputPtr[ct1];
12.           matrix[2][i * w + j + 1] = inputPtr[ct3];
13.
14.           ct0 = ct0 + 4;
15.           ct1 = ct1 + 4;
16.           ct2 = ct2 + 4;
17.           ct3 = ct3 + 4;
18.         }
19.       }
20.
21.       for (i = 0; i < inputHeight; i++)
22.       {
23.         for (j = 0; j < inputWidth; j++)
24.         {
25.           yuv[0][i * inputWidth + j] = matrix[0][i * w + j];
26.           yuv[1][i * inputWidth + j] = matrix[1][i * w  +
   j];
27.           yuv[2][i * inputWidth + j] = matrix[2][i * w  +
   j];
28.         }
29.       }
```

Slika 3.9. *Programski kôd za učitavanje ulazne slike*

Programski kôd za određivanje koordinate najbližeg susjeda i zapisivanje vrijednosti u spremnik *matrix* je dostupan na slici 3.10.

```
1.   double xScale = (inputWidth * sf) / (inputWidth - 1.00);
2.   double yScale = (inputHeight * sf) / (inputHeight - 1.00);
3.   for (i = 0; i < outputHeight; i++)
4.   {
5.     for (j = 0; j < outputWidth; j++)
6.     {
7.       iCord = (int)(round((double)i / yScale));
8.       jCord = (int)(round((double)j / xScale));
9.
10.          matrix[0][i * outputWidth + j] = yuv[0][iCord *
   inputWidth + jCord];
11.          matrix[1][i * outputWidth + j] = yuv[1][iCord *
   inputWidth + jCord];
12.          matrix[2][i * outputWidth + j] = yuv[2][iCord *
   inputWidth + jCord];
13.        }
14.      }
```

Slika 3.10. *Algoritam interpolacije metodom najbližih susjeda*

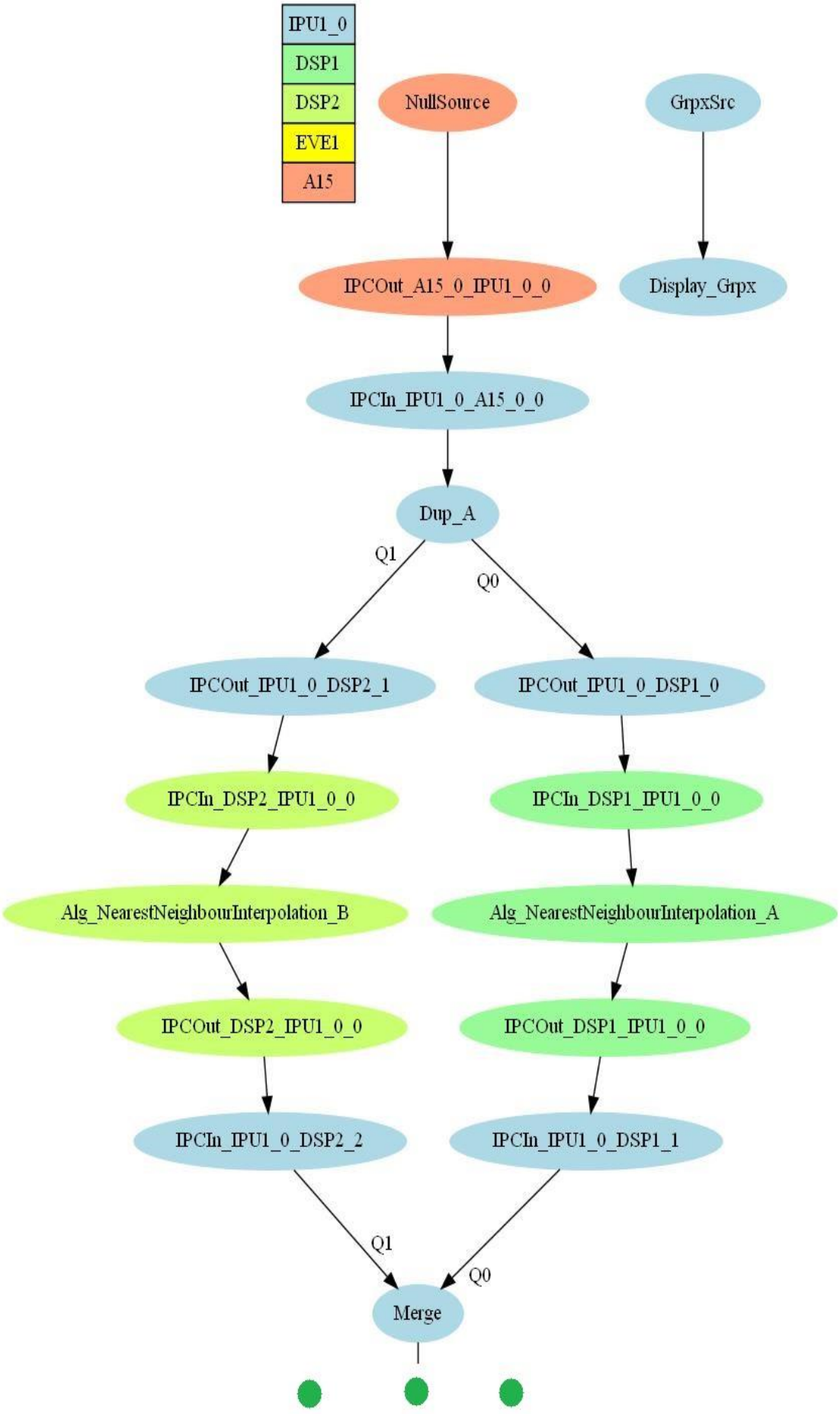
Budući da se ulazni tok koristi kao i izlazni tok, iz spremnika *matrix* je potrebno vratiti vrijednosti u tok podataka *inputPtr*, a programski kôd za to nalazi se na slici 3.11. i ovime završava algoritam za interpolaciju metodom najbližih susjeda za procesore DSP1 i A15. Identična logika se koristi kod bilinearne interpolacije i bikubične interpolacije. Elektronički prilog P.3.1. sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor DSP1, a elektronički prilog P.3.2. sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor A15. Korištenjem više različitih procesora se nastoji ispitati radi li predložena implementacija brže na procesoru A15 ili DSP.

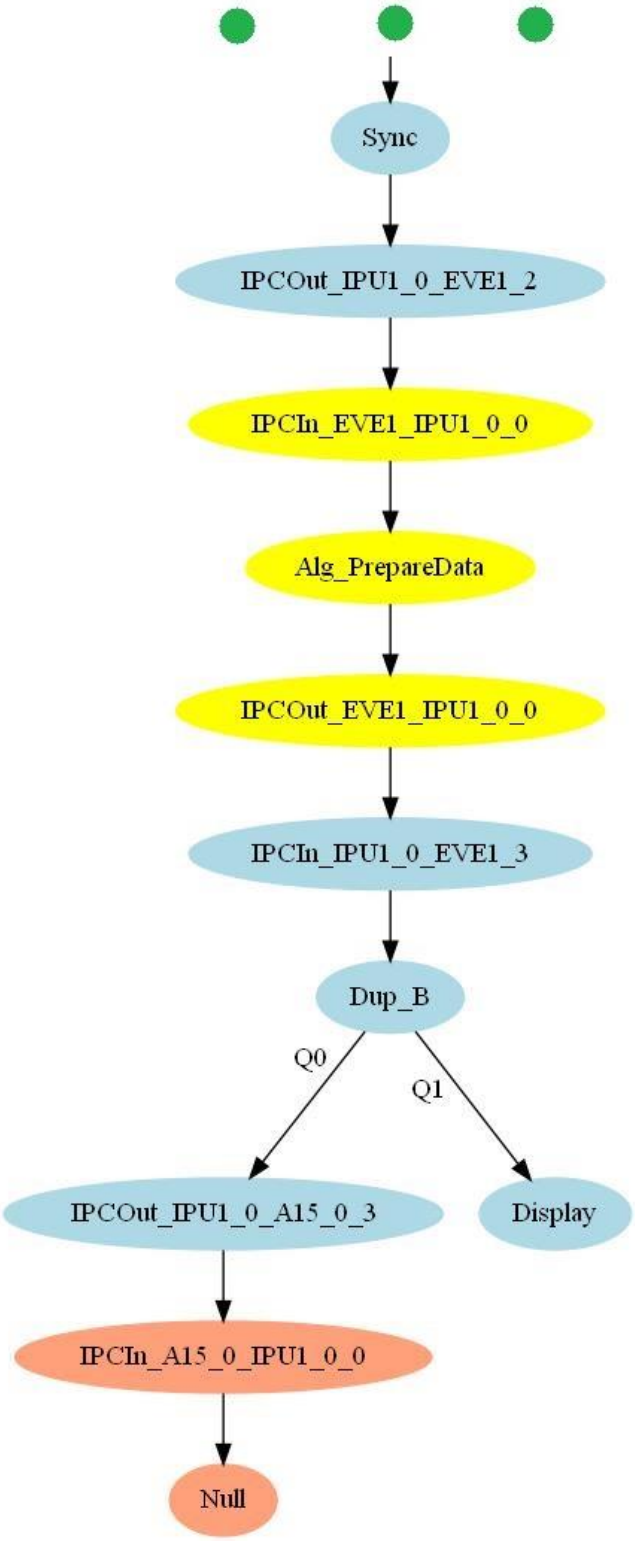
```
1.   int counter = 0;
2.   for (i = 0; i < outputHeight; i++)
3.   {
4.     for (j = 0; j < outputWidth; j = j + 2)
5.     {
6.       inputPtr[counter] = matrix[0][i * outputWidth + j];
7.       counter++;
8.       inputPtr[counter] = matrix[1][i * outputWidth + j];
9.       counter++;
10.      inputPtr[counter] = matrix[0][i * outputWidth + j + 1];
11.      counter++;
12.      inputPtr[counter] = matrix[2][i * outputWidth + j + 1];
13.      counter++;
14.         }
15.       }
```

Slika 3.11. *Programski kôd za zapisivanje rezultata u izlazni tok*

3.4.2. Implementacija interpolacije slike metodom najbližih susjeda koristeći istovremeno procesore DSP1 i DSP2

Istovremenim korištenjem procesora DSP1 i DSP2 nastoji se postići dvostruko ubrzanje vremena izvođenja algoritma u odnosu na korištenje samo jednog DSP-a. Za izvođenje algoritma koristeći istovremeno procesore DSP1 i DSP2, potrebno je prilagoditi algoritam za interpolaciju metodom najbližih susjeda i zapisivanje rješenja u izlazni tok podataka, a samim time i slučaj upotrebe. Dijagram toka slučaja upotrebe se nalazi na slici 3.12.


IPU1_0
DSP1
DSP2
EVE1
A15
NullSource
GrpxSrc
IPCOut_A15_0_IPU1_0_0
Display_Grpx
IPCIn_IPU1_0_A15_0_0
Dup_A
Q1
Q0
IPCOut_IPU1_0_DSP2_1
IPCOut_IPU1_0_DSP1_0
IPCIn_DSP2_IPU1_0_0
IPCIn_DSP1_IPU1_0_0
Alg_NearestNeighbourInterpolation_B
Alg_NearestNeighbourInterpolation_A
IPCOut_DSP2_IPU1_0_0
IPCOut_DSP1_IPU1_0_0
IPCIn_IPU1_0_DSP2_2
IPCIn_IPU1_0_DSP1_1
Q1
Q0
Merge



Slika 3.12. *Dijagram toka slučaja upotrebe algoritma za metodu najbližih susjeda uz istovremeno korištenje procesora DSP1 i DSP2*

Budući da je u slučaju upotrebe obrada slike podijeljena na procesore DSP1 i DSP2, nužno je prilagoditi algoritam, a kod za prilagođenje algoritma se nalazi na slici 3.13. Budući da je ulazna slika koju treba obraditi veličine 640×360 elemenata slike, svaki DSP interpolira polovicu slike, tj. 320×360 elemenata slike. Identična prilagodba (podjela) se koristi kod bilinearne interpolacije i bikubične interpolacije.

```
1.   if (System_getSelfProcId()==SYSTEM_PROC_DSP1)
2.   {
3.     startPointX = 0;
4.     startPointY = 0;
5.
6.     endPointX = inputWidth;
7.     endPointY = inputHeight/2;
8.     outputWidth*outputHeight;
9.     outputBufferHalfSize;
10.     x=(((h/2) -(outputHeight/2))*(w))*2; }
11.     else if (System_getSelfProcId() == SYSTEM_PROC_DSP2)
12.       {
13.         startPointX = 0;
14.         startPointY =  inputHeight/2;
15.
16.         endPointX =inputWidth;
17.         endPointY = inputHeight;
18.         x=(h/2)*(w*2); }
```

Slika 3.13. *Programski kôd za podjelu obrade slike na procesore DSP1 i DSP2*

Programski kod prilagodbe algoritma interpolacije metodom najbližih susjeda se nalazi na slici 3.14., a prilagodba zapisivanja rezultata u izlazni tok se nalazi na slici 3.15. Elektronički prilog P. 3.3. sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1 i DSP2.

```
1.   double xScale = (inputWidth * sf) / (inputWidth - 1.00);
2.   double yScale = (inputHeight * sf) / (inputHeight - 1.00);
3.   for (i = startPointY*sf; i < endPointY*sf; i++)
4.   {
5.     for (j = startPointX*sf; j < endPointX*sf; j++)
6.     { //... } }
```

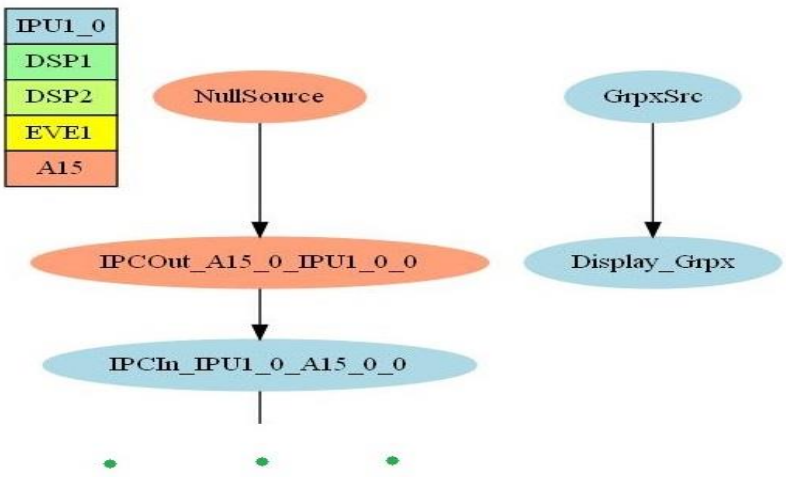
Slika 3.14. *Programski kôd za prilagodbu algoritma interpolacije metodom najbližih susjeda za istovremeno korištenje procesora DSP1 i DSP2*

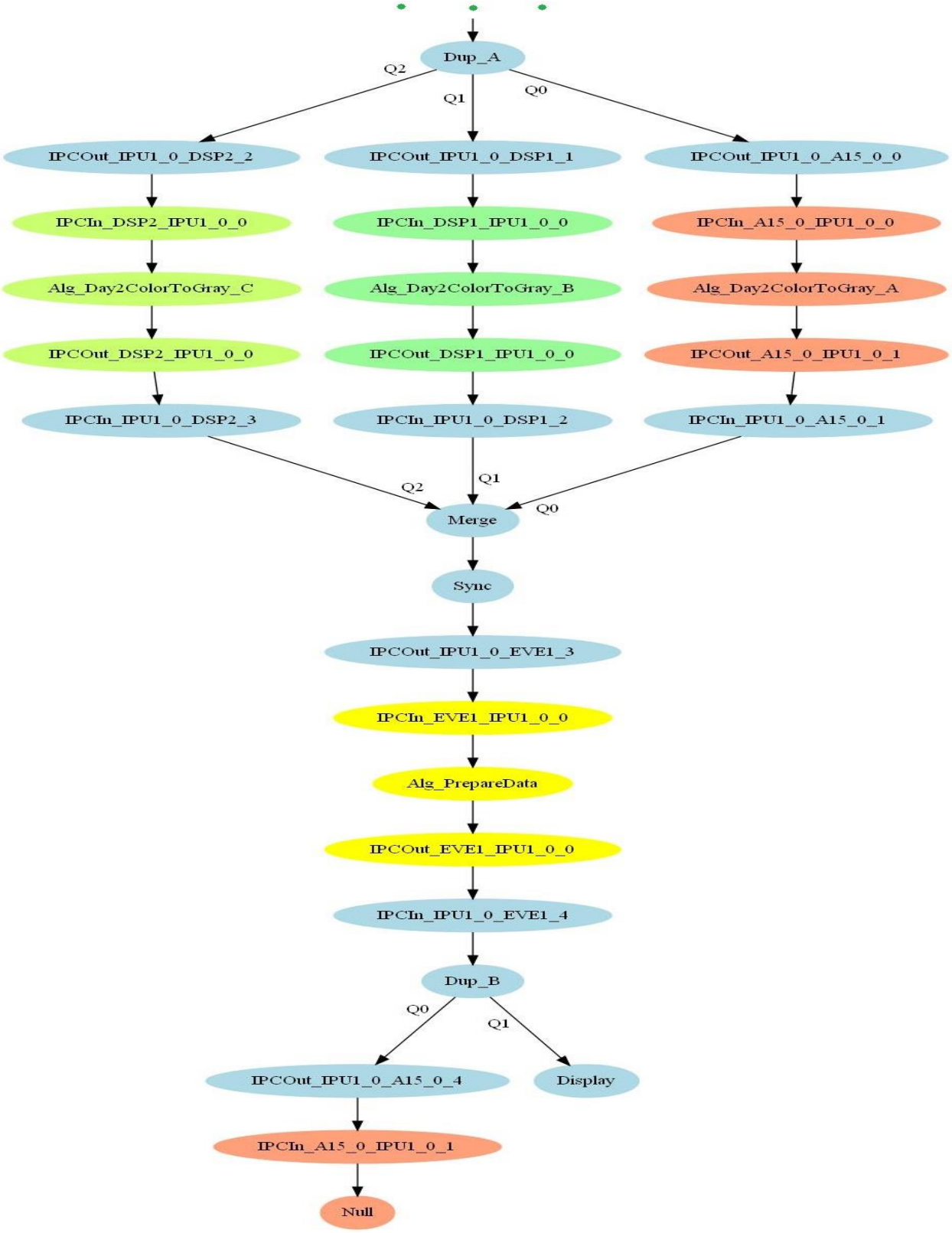
```
1.   for (i = startPointY*sf; i < endPointY*sf; i++)
2.   {
3.     counter=0;
4.     for (j = startPointX*sf; j < endPointX*sf; j = j + 2)
5.     {
6.       inputPtr[x+counter] = matrix[0][i * outputWidth + j];
7.       counter++;
8.       inputPtr[x+counter] = matrix[1][i * outputWidth + j];
9.       counter++;
10.           inputPtr[x+counter] = matrix[0][i * outputWidth +
   j + 1];
11.           counter++;
12.           inputPtr[x+counter] = matrix[2][i * outputWidth +
   j + 1];
13.           counter++;
14.         }
15.         x+=w*2;
16.       }
```

Slika 3.15. *Programski kôd za prilagodbu zapisivanja rezultata u izlazni tok za istovremeno korištenje procesora DSP1 i DSP2*

3.4.3. Implementacija interpolacije slike metodom najbližih susjeda koristeći istovremeno procesore A15, DSP1 i DSP2

Istovremenim korištenjem triju procesora, tj. A15, DSP1 i DSP2 nastoji se veće ubrzanje vremena izvođenja algoritma u odnosu na samostalno korištenje procesora A15 i istovremeno korištenje procesora DSP1 i DSP2. Za izvođenje algoritma koristeći istovremeno procesore A15, DSP1 i DSP2, potrebno je prilagoditi slučaj upotrebe za metodu najbližih susjeda. Dijagram toka slučaja upotrebe se nalazi na slici 3.16.





Slika 3.16. *Dijagram toka slučaja upotrebe algoritma za metodu najbližih susjeda uz istovremeno korištenje procesora A15, DSP1 i DSP2*

Slučaj upotrebe izvodi isti algoritam na tri različita procesora, te je zbog toga nužno prilagoditi i algoritam interpolacije metodom najbližih susjeda. Prilikom interpolacije metodom najbližih susjeda, procesor A15 interpolira 90% slike, a procesori DSP1 i DSP2 po 5% slike. Podjela je napravljena s obzirom na brzinu procesora A15 u odnosu na procesor DSP1 koja je utvrđena izvođenjem prethodno opisanih slučajeva upotrebe. Prilagođenje algoritma se nalazi na slici 3.17.

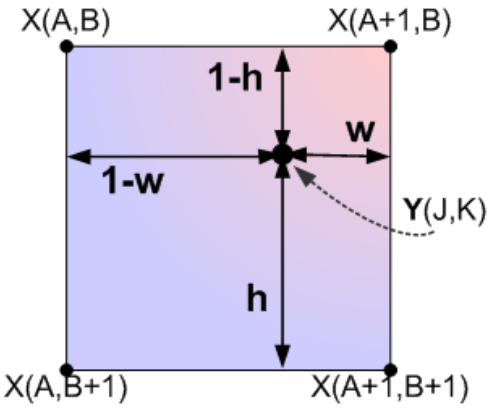
```
1.   if (System_getSelfProcId()==SYSTEM_PROC_A15_0)
2.   {
3.       startPointX = 0;
4.       startPointY = 0;
5.       endPointX = inputWidth;
6.       endPointY = inputHeight*0.9;
7.       x=0;
8.   }
9.   else if (System_getSelfProcId() == SYSTEM_PROC_DSP1)
10.       {
11.           startPointX = 0;
12.           startPointY =  inputHeight*0.9;
13.           endPointX =inputWidth;
14.           endPointY = inputHeight*0.95;
15.           x=(outputHeight*0.9)*(w*2);
16.       }
17.       else if (System_getSelfProcId() == SYSTEM_PROC_DSP2)
18.       {
19.         startPointX = 0;
20.         startPointY =  inputHeight*0.95;
21.         endPointX = inputWidth;
22.         endPointY = inputHeight;
23.         if ((int)sf == 2)
24.         {
25.           x = (h*0.95)*(w*2);
26.         }
27.         else
28.         {
29.           x = (outputHeight*0.95)*(w*2) - w*2 - w*2;
30.         }
31.       }
```

Slika 3.17. *Programski kôd za podjelu obrade slike na procesore A15, DSP1 i DSP2 prilikom interpolacije metodom najbližih susjeda*

Elektronički prilog P. 3.4. sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1, DSP2 i A15. Slična ideja se koristi i kod bilinearne interpolacije, a više o tome u sljedećem dijelu.

3.5. Implementacija bilinearne interpolacije na Alpha ploču

Algoritam bilinearne interpolacije se zasniva na matematičkom modelu bilinearne interpolacije opisanom u dijelu 2.1.2. Prilikom povećavanja uzoraka korištenjem bilinearne interpolacije stvara se novi element slike iz težinskog prosjeka četiriju najbližih susjeda u izvornoj slici. Težine koje se pritom pridaju pojedinom od ta četiri elementa slike su: *h*, *1 - h*, *w*, *1 – w* (slika 3.18.). Težine su određene relativnim položajem elementa slike u usporedbi s njegovim susjedima na originalnoj slici.



Slika 3.18. *Transformacija između koordinata izvorne slike i konačne izlazne slike* [20]

Zbog ovog svojstva, interpolirana vrijednost bilo kojeg elementa slike može se izračunati prema izrazu (3-1):

$$Y[i,j] = (1-w)\cdot(1-h)\cdot X[A,B] + w\cdot(1-h)\cdot X[A+1,B] + (1-w)\cdot h\cdot X[A,B+1] + w\cdot h\cdot X[A+1,B+1], \quad (3\text{-}1)$$

gdje su *i*, *j* koordinate elementa slike u novoj interpoliranoj slici, *A*, *B* koordinate najbližeg susjeda na izvornoj slici. Programski kôd za implementaciju bilinearne interpolacije se nalazi na slici 3.19.

```
1. double xScale = (inputWidth * sf) / (inputWidth - 1.00);
2. double yScale = (inputHeight * sf) / (inputHeight - 1.00);
3. double iCordFloor = 0, jCordFloor = 0, iCordCeil = 0,
   jCordCeil = 0;
4.     UInt8 q11, q21, q12, q22;
5.     for (i = 0; i < outputHeight; i++)
6.     { for (j = 0; j < outputWidth; j++)
7.         { double I = (double)i;
8.             double J = (double)j;
```

```
8.              //...
9.              double W = -(((I / yScale) - floor(I / yScale)) -
   1);
10.                 double H = -(((J / xScale) - floor(J /
   xScale)) - 1);
11.                 double result;
12.                 iCordFloor = floor(I / yScale);
13.                 jCordFloor = floor(J / xScale);
14.                 iCordCeil = ceil(I / yScale);
15.                 jCordCeil = ceil(J / xScale);
16.                 q11 = yuv[0][(int)(iCordFloor * inputWidth +
   jCordFloor)];
17.                 q12 = yuv[0][(int)(iCordCeil * inputWidth +
   jCordFloor)];
18.                 q21 = yuv[0][(int)(iCordFloor * inputWidth+
   jCordCeil)];
19.                 q22 = yuv[0][(int)(iCordCeil * inputWidth +
   jCordCeil)];
20.                 result = ((1 - W) * (1 - H) * q22) + (W * (1
   - H) * q21) + ((1 - W) * H * q12) + (W * H * q11);
21.                 matrix[0][i * outputWidth  + j] =
   (UInt8)result; //...
```

Slika 3.19. *Programski kôd bilinearne interpolacije*

U linijama 11 i 12 određuje se relativna pozicija novog elementa slike na novoj interpoliranoj slici u odnosu na originalnu sliku, dok izraz (3-1) odgovara liniji 22 za računanje vrijednosti novog elementa slike. Postupak se provodi za svaku komponentu boje zasebno. [20] U elektroničkom prilogu P.3.4. se nalazi slučaj upotrebe bilinearne interpolacije: u datoteci *blA15.c* za procesor A15, u datoteci *blDSP1.c* za procesor DSP1 i u datoteci *blDSP1DSP2.c* za istovremeno korištenje procesora DSP1 i DSP2.

Kako je navedeno u dijelu 3.4.2., bilinearna interpolacije za istovremeno izvođenje na procesorima A15, DSP1 i DSP2 zahtjeva određene izmjene. Za uspješno izvođenje bilinearne interpolacije na navedenim procesorima, procesor A15 interpolira 75% slike, a procesori DSP1 i DSP2 po 12,5% slike. Navedeni odnos je utvrđen provođenjem opisanog slučaja upotrebe. Na slici 3.19 se nalazi programski kôd za prilagođenje algoritma bilinearne interpolacije za navedenu raspodjelu procesora. Programski kôd prilagođenja se nalazi na slici 3.20. Elektronički prilog P.

3.5. sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesora DSP1, elektronički prilog P. 3.6. za procesor A15, elektronički prilog P. 3.7. za istovremeno korištenje procesora DSP1 i DSP2, a elektronički prilog P. 3.8. za istovremeno korištenje procesora A15, DSP1 i DSP2.

```
1.   if (System_getSelfProcId()==SYSTEM_PROC_A15_0)
2.   {
3.       startPointX = 0;
4.       startPointY = 0;
5.
6.       endPointX = inputWidth;
7.       endPointY = inputHeight*0.75;
8.       x=0;
9.
10.       }
11.       else if (System_getSelfProcId() == SYSTEM_PROC_DSP1)
12.       {
13.           startPointX = 0;
14.           startPointY =  inputHeight*0.75;
15.
16.           endPointX =inputWidth;
17.           endPointY = inputHeight*0.875;
18.
19.           x=(outputHeight*0.75)*(w*2);
20.       }
21.       else if (System_getSelfProcId() == SYSTEM_PROC_DSP2)
22.       {
23.         startPointX = 0;
24.         startPointY =  inputHeight*0.875;
25.
26.         endPointX = inputWidth;
27.         endPointY = inputHeight;
28.
29.         if ((int)sf == 2)
30.         {
31.           x = (h*0.875)*(w*2);
32.         }
33.         else
34.         {
35.           x = (outputHeight*0.875)*(w*2) - w;
36.                                 }
37.       }
```

Slika 3.20. *Programski kôd za podjelu obrade slike na procesore A15, DSP1 i DSP2 prilikom bilinearne interpolacije*

3.6. Implementacija bikubične interpolacije na Alpha ploču

Rješenje algoritma bikubične interpolacije se oslanja na temu razrađenu u dijelu 2.1.3. Bikubična interpolacija vrši interpoliranje u *x* smjeru, *y* smjeru i *xy* smjeru. Za svaki od četiri susjedna elementa slike, nužno je poznavati vrijednost intenziteta elementa slike, parcijalne derivacije u *x* (3-2) smjeru i *y* (3-3) smjeru, te dijagonalnu *xy* (3-4) derivaciju:

$$\frac{\delta f}{\delta x} = f_x \tag{3-2}$$

$$\frac{\delta f}{\delta y} = f_y \tag{3-3}$$

$$\frac{\delta^2 f}{\delta x \delta y} = f_{xy} \tag{3-4}$$

Parcijalne derivacije se mogu izračunati iz centriranih razlika, za koje se mogu pretpostaviti da su izračuni nagiba pomoću vrijednosti intenziteta iz susjednih elemenata slike. Prema tome, za f_x derivaciju vrijedi izraz (3-5), za f_y derivaciju vrijedi izraz (3-6), a za dijagonalnu derivaciju f_{xy} vrijedi izraz (3-7).

$$f_x(i,j) = \frac{f(i+1,j)-f(a-1,j)}{2} \tag{3-5}$$

$$f_y(i,j) = \frac{f(i,j+1)-f(i,j-1)}{2} \tag{3-6}$$

$$f_x(i,j) = \frac{[f(i-1,b-1)+f(i+1,j+1)]-[f(i+1,b-1)+f(i-1,b+1)]}{4} \tag{3-7}$$

Uz pomoć prethodno navedenih formula i reference na sliku 3.17., može se definirati bilo koja interpolirana vrijednost izlazne slike *p(x,y)* u obliku funkcije elementa slike na ulaznoj slici pomoću izraza (3-8), (3-9), (3-10) i (3-11) .

$$p(J,K) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{mn}(1-W)^m \cdot (1-H)^n \tag{3-8}$$

$$p_x(J,K) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{mn}(1-W)^{m-1} \cdot (1-H)^n \tag{3-9}$$

$$p_y(J,K) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{mn}(1-W)^m \cdot (1-H)^{n-1} \tag{3-10}$$

$$p_{xy}(J,K) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{mn}(1-W)^{m-1} \cdot (1-H)^{n-1} \tag{3-11}$$

Potrebno je odrediti vrijednosti koeficijenata a_{mn}. Uz pretpostavku da su *W* i *H* kontinuirani na intervalu [0, 1], redefiniraju se krajnje točke prema slici 3.17, a definiraju se na sljedeći način:

1. $f(A,B) = f(0,0)$
2. $f(A+1,B) = f(1,0)$
3. $f(A,B+1) = f(0,1)$
4. $f(A+1,B+1) = f(1,1)$

Moguće je izraziti svih 16 koeficijenata a_{mn}, za vrijednosti funkcije sljedeće četiri vrijednosti:

1. $f(0,0) = a_{00}$
2. $f(1,0) = a_{00} + a_{10} + a_{20} + a_{30}$
3. $f(0,1) = a_{00} + a_{01} + a_{02} + a_{03}$
4. $f(1,1) = a_{00} + a_{01} + a_{02} + a_{03} + a_{10} + a_{11} + a_{12} + a_{13} + a_{20} + a_{21} + a_{22} + a_{23} + a_{30} + a_{31} + a_{32} + a_{33}$,

za derivaciju po *x*, sljedeće četiri vrijednosti:

1. $f_x(0,0) = a_{10}$
2. $f_x(1,0) = a_{10} + 2a_{20} + 3a_{30}$
3. $f_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13}$
4. $f_x(1,1) = \sum_{m=1}^{3}\sum_{n=0}^{3} a_{mn}m$,

za derivaciju po *y*, sljedeće četiri vrijednosti:

1. $f_y(0,0) = a_{01}$
2. $f_y(1,0) = a_{01} + a_{11} + a_{31}$
3. $f_y(0,1) = a_{01} + 2a_{02} + 3a_{03}$
4. $f_y(1,1) = \sum_{m=0}^{3}\sum_{n=1}^{3} a_{mn}n$,

za dijagonalne (parcijalne) derivacije prvo po *x* zatim po *y*, sljedeće četiri vrijednosti:

1. $f_{xy}(0,0) = a_{11}$
2. $f_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31}$
3. $f_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13}$
4. $f_{xy}(1,1) = \sum_{m=1}^{3}\sum_{n=1}^{3} a_{mn}mn$.

U konačnici, dobije se skup linearnih jednadžbi koje tvore matričnu jednadžbu općeg oblika **A**x = b te se može zapisati u obliku:

$$M\alpha = \beta \tag{3-12}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 2 & 4 & 6 & 0 & 3 & 6 & 9 \end{bmatrix}$$

Za računanje bikubične interpolacije, potrebno je odrediti inverznu matricu matrice **M**, odnosno potrebno je odrediti $\mathbf{M^{-1}}$**:**

$$M^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\ -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\ 9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\ -6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & -2 & -2 & -2 & -1 & -1 \\ 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ -6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & -3 & -2 & -1 & -2 & -1 \\ 4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & 2 & 1 & 1 & 1 & 1 \end{bmatrix}$$

U izrazu (3-12), β je jednostupčasta matrica koja sadrži 16 vrijednosti, prve četiri vrijednosti su vrijednosti četiriju elementa slike ulazne slike, druge četiri vrijednosti dobivene su po formuli (3-5), sljedeće četiri vrijednosti su vrijednosti koje se dobiju prema izrazu (3-6), a posljednje četiri vrijednosti se izračunaju pomoću izraza (3-7). Kada se izraz (3-12) s lijeve strane pomnoži s M^{-1} dobije se izraz (3-13).

$$\alpha = M^{-1}\beta \tag{3-13}$$

Koristeći formulu (3-14), moguće je izračunati bilo koju vrijednost elementa slike nove interpolirane slike, gdje a_{mn} odgovara vrijednosti α za lokaciju *(i,j).*

$$p(i,j) = \sum_{m=0}^{3} \sum_{n=0}^{3} a_{mn}(1-w)^m(1-h)^n \tag{3-14}$$

Dio programskog koda rješenja koji odgovara prethodnom pojašnjenju se nalazi na slici 3.21.

```
1.   beta[0] = yuv[0][iCordFloor * inputWidth + jCordFloor];
2.   beta[1] = yuv[0][iCordFloor * inputWidth + jCordCeil];
3.   beta[2] = yuv[0][iCordCeil * inputWidth + jCordFloor];
4.   beta[3] = yuv[0][iCordCeil * inputWidth + jCordCeil];
5.   beta[4] = Ix[0][iCordFloor * inputWidth + jCordFloor];
6.   beta[5] = Ix[0][iCordFloor * inputWidth + jCordCeil];
7.   beta[6] = Ix[0][iCordCeil * inputWidth + jCordFloor];
8.      beta[7] = Ix[0][iCordCeil * inputWidth + jCordCeil];
9.      beta[8] = Iy[0][iCordFloor * inputWidth + jCordFloor];
10.     beta[9] = Iy[0][iCordFloor * inputWidth + jCordCeil];
11.     beta[10] = Iy[0][iCordCeil * inputWidth + jCordFloor];
12.     beta[11] = Iy[0][iCordCeil * inputWidth + jCordCeil];
13.     beta[12] = Ixy[0][iCordFloor * inputWidth + jCordFloor];
14.     beta[13] = Ixy[0][iCordFloor * inputWidth + jCordCeil];
15.     beta[14] = Ixy[0][iCordCeil * inputWidth + jCordFloor];
16.     beta[15] = Ixy[0][iCordCeil * inputWidth + jCordCeil];
17.     alpha = getAlpha(beta, alphaArr);
18.     volatile int f = 0;
19.     temp = 0;
20.     for (f = 0; f < 16; f++)
21.     {
22.         w_temp = floor((double)f / 4);
23.         h_temp = fmod((double)f, 4);
24.         temp = temp + Alpha[f] * (pow(1 - W, w_temp) * pow(1
  - H, h_temp));
25.      }
26.     temp = clip(temp);
27.      matrix[0][i * outputWidth + j] = (uint8_t)temp;
28.
```

Slika 3.21. *dio programskog kôda rješenja bikubične interpolacije*

Elektronički prilog P. 3.9. sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesora DSP1, elektronički prilog P. 3.10. za procesor A15, elektronički prilog P. 3.11. za istovremeno korištenje procesora DSP1 i DSP2, a elektronički prilog P. 3.12. za istovremeno korištenje procesora A15, DSP1 i DSP2. Mjerenja vremena izvođenja nisu moguća za istovremeno korištenje procesora DSP1, DSP2 i A15. Više o razlozima u sljedećem poglavlju.

3.7. Optimizacija rješenja

Optimizacija je način prilagodbe programskog kôda za određenu platformu s ciljem povećanja brzine izvršavanja programa, smanjenja zauzeća memorije, odnosno povećavanja efikasnosti rješenja. Prvo gotovo rješenje metode najbližih susjeda je implementirano na računalu metodom *hardcodinga*, odnosno ugrađivanja rješenja u izvorni kod programa, za razliku od dobivanja dijelova koda iz vanjskih funkcija i generiranja tijekom izvođenja. Metoda *hardcodinga* dala je za interpolaciju metodom najbližih susjeda na prvoj testnoj slici vrijeme izvođenja od 0.065 sekundi prilikom povećavanja rezolucije četiri puta (640×360 elemenata slike na 1280×720 elemenata slike). Nakon provedene optimizacije programskog koda, vrijeme izvođenja na prvoj testnoj slici bilo je 0.057 sekundi, što je ubrzanje vremena izvođenja za 12,3%. Optimizacija je uključivala uklanjanje dijelova koda koji se ponavljaju, neke operacija dijeljenja i množenja su zamijenjene pomicanjem bitova (engl. *bit shifting*), iteracijske varijable su se inicijalizirale s ključnom riječju *volatile* koja ne dopušta kompajleru da sam optimizira njezinu vrijednost. Uključena je dinamička alokacija memorije za velike spremnike podataka i njihova dealokacija nakon prestanka korištenja istih.

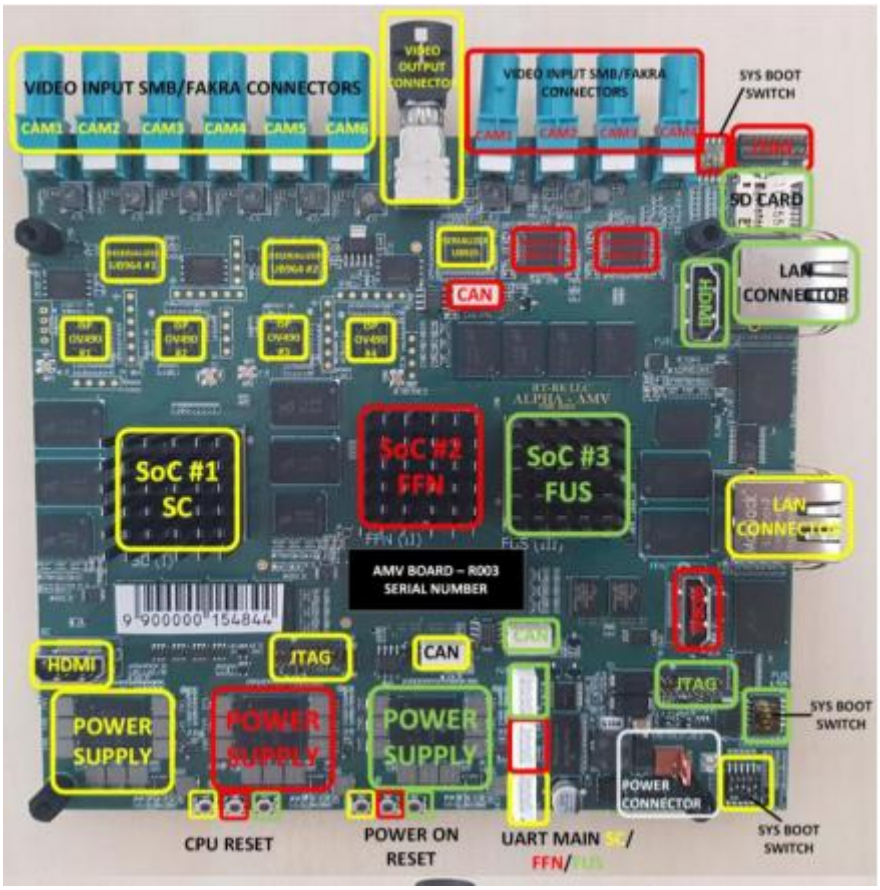
3.8. Način pokretanja rješenja na Alpha ploči

Prije pokretanja stvorenog rješenja za odabranu metodu interpolacije, potrebna je instalacija programskog paketa VSDK i terminala. Terminal je računalna konzola koja služi kao izlazni hardverski uređaj za prikaz tekstualnog sadržaja koji je izlaz s mikrokontrolera ili naprednijeg uređaja, komunikacija se odvija preko žice (UART[5]). Korišteni terminal za UART komunikaciju je TerraTerm. TerraTerm je besplatni alat koji služi za emulaciju terminala. Koraci za pokretanje rješenja nalaze se u tablici 3.2. Za pretpostavku se uzima da je instalacija VSDK i TeraTerm-a napravljena na C disku računala na kojemu je glavni operacijski sustav Windows.

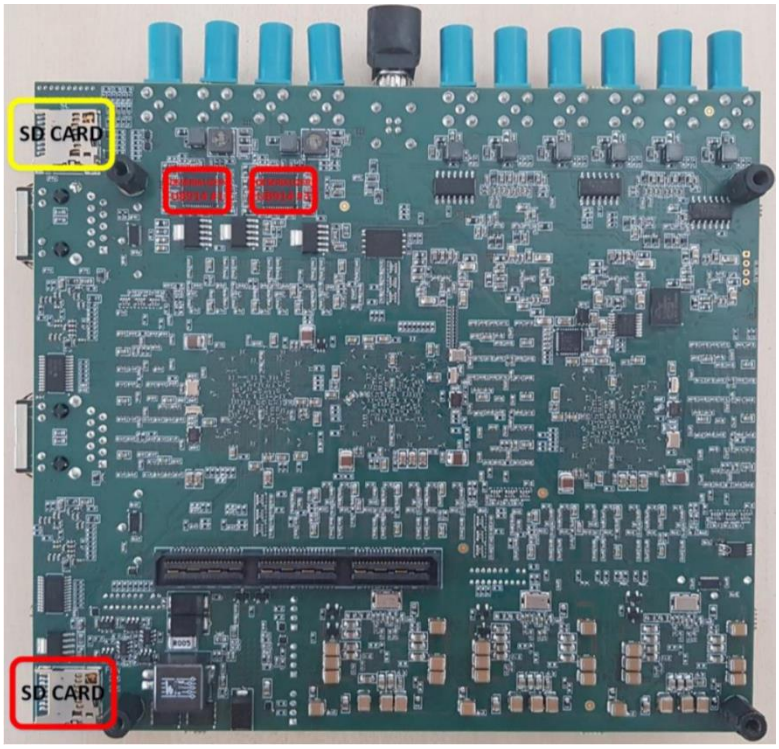
[5] UART (engl. *Universal Ansychronus Receiver-Transmitter*) predstavlja međuvezu između računala i okoline putem asinkronog prijenosa podataka

Tablica 3.3. *Upute za pokretanje rješenja na Alpha ploči*

KORAK	RADNJA
1	Otvori se datoteka koja ima putanju VSDK\vision_sdk\amv_config.h gdje se postavi SoC na SCV
2	Pokreće se Windows Command Prompt (CMD)
3	Unutar otvorenog CMD-a, pozicionira se unutar mape C:\VISION_SDK_02_12_01_00\ti_components\networking\ nsp_gmacsw_4_15_00_00
4	Unutar otvorene mape u CMD-u, pokreće se izgradnja mrežnog adaptera pomoću naredbe *xdc --xdcpath="C:/VISION_SDK_02_12_01_00/ti_components/os_tools/ bios_6_46_00_23/packages" -P packages/ti/nsp/drv/*
5	Unutar pokrenutog CMD-a, pozicionira se u mapu C:\VISION_SDK_02_12_01_00\vision_sdk
6	Unutar CMD-a, pokrenu se, redom, naredbe: 1. *gmake -s -j depend* – izgrađuje se niža razina VSDK 2. *gmake -s -j sbl_sd* – izgrađuje se *bootloader* (potreban za microSD karticu) 3. *gmake -s –j* – izgradnja VSDK rješenja 4. *gmake -s appimage* – izgradnja komprimirane datoteke VSDK rješenja
7	MicroSD kartica se formatira prema uputama koje su dio VSDK okruženja, microSD kartica služi za pokretanje rješenja na Alpha ploči.
8	Datoteka MLO unutar mape C:\VISION_SDK_02_12_01_00\vision_sdk\build\scripts\sd_tda2xx-evm\MLO te dadoteke AppImage unutar mape C:\VISION_SDK_02_12_01_00\vision_sdk\binaries\tda2xx_evm_bios_all\ vision_sdk\bin\t da2xx-evm\sbl_boot\AppImage se kopiraju na microSD karticu.
9	Nakon uspješne pohrane na microSD karticu, ona se umeće u za to predviđeni utor željenog SoC-a. Također, uključuje se kabel za UART komunikaciju i Ethernet kabel na za to predviđena mjesta određena SoC-om. Organizacija sučelja i sastavnih dijelova ploče je vidljiva na slikama 3.22 i 3.23.

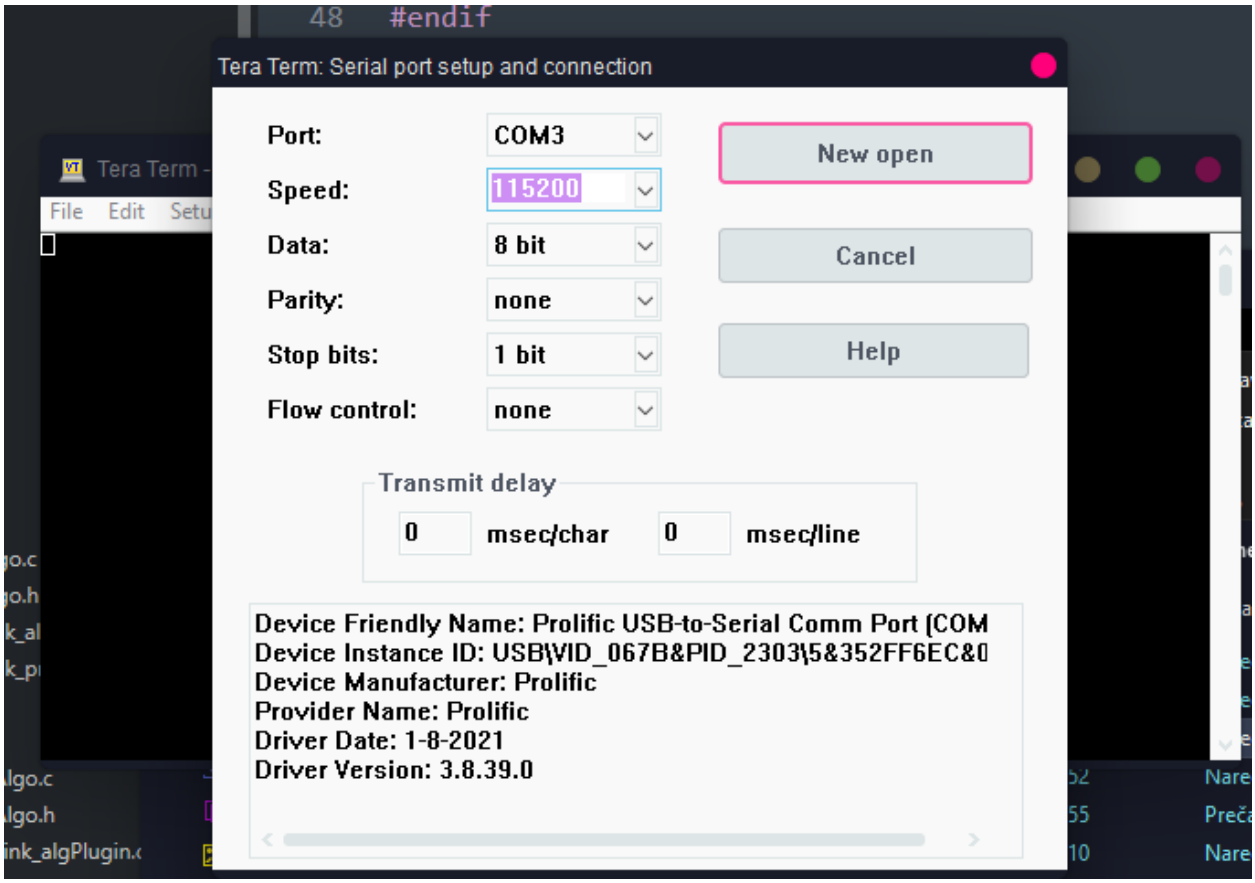


Slika 3.22. *Organizacija Alpha ploče s vidljive strane gledano od postolja*

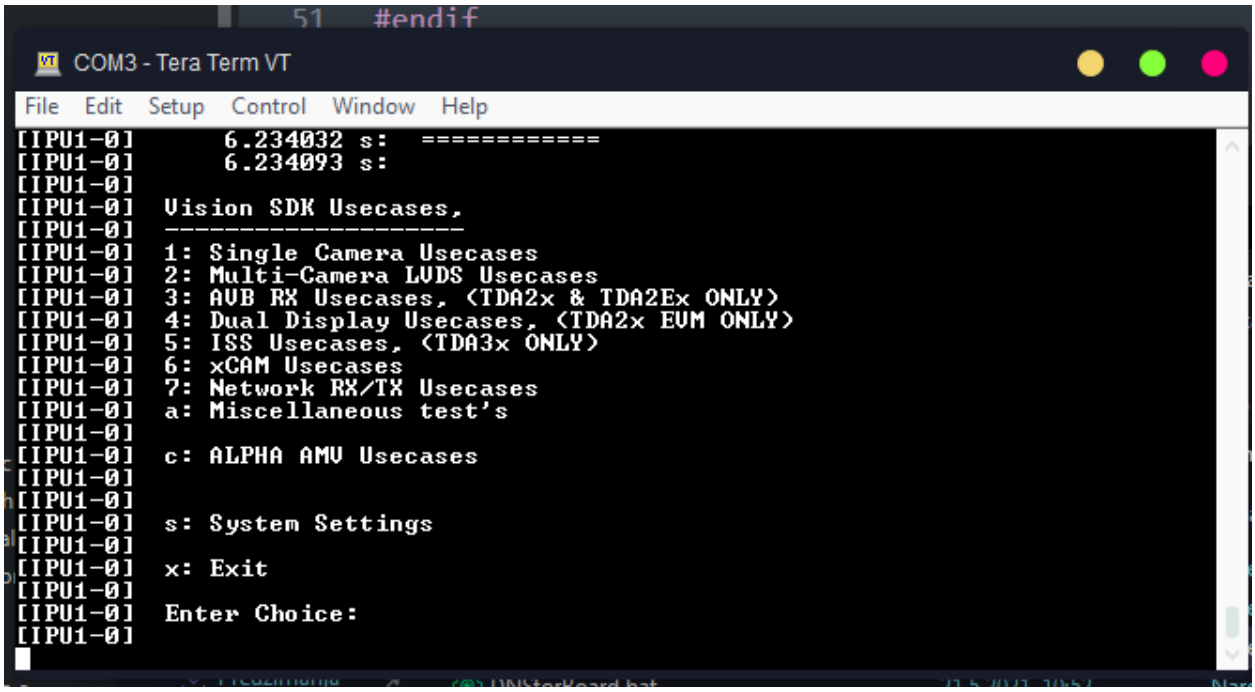


Slika 3.23. *Organizacija Alpha ploče s donje strane gledano od postolja*

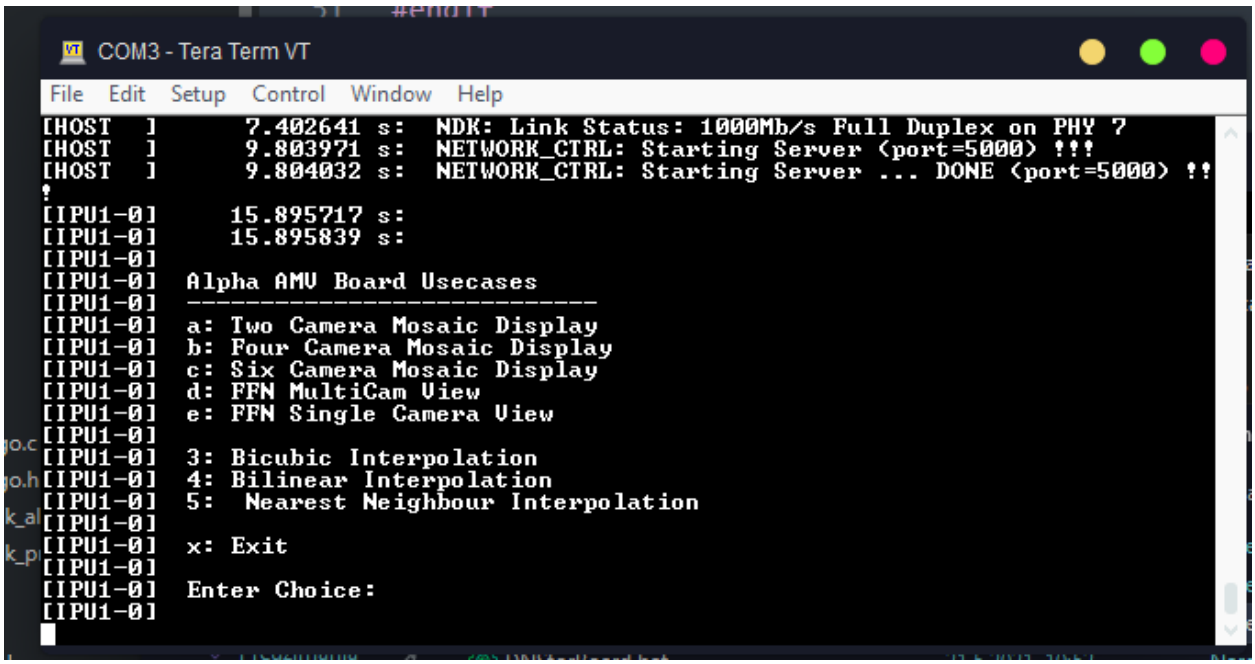
Nakon uspješno odrađenih koraka u tablici 3.3., pokreće se TerraTerm te se postavlja prema slici 3.24. Nakon postavljanja TerraTerm-a, potrebno je uključiti napajanje Alpha ploče. Nakon uključivanja napajanja, u terminalu se pokrene izbornik prikazan na slici 3.25. Pritiskom tipke „C“, pokrenu se dostupni slučajevi korištenja. Dostupni slučajevi korištenja su vidljivi na slici 3.26. Odabrana metoda interpolacije se pokreće pritiskom na broj ispred metode, što znači za interpolaciju bikubičnom interpolacijom pritisne se tipka „3“, za interpolaciju bilinearnom interpolacijom pritisne se tipka „4“, a za interpolaciju metodom najbližih susjeda pritisne se tipka „5“. Prije odabira metode interpolacije, nužno je postupiti prema uputama u tablici 3.4.



Slika 3.24. *Postavljanje TerraTerm terminala*



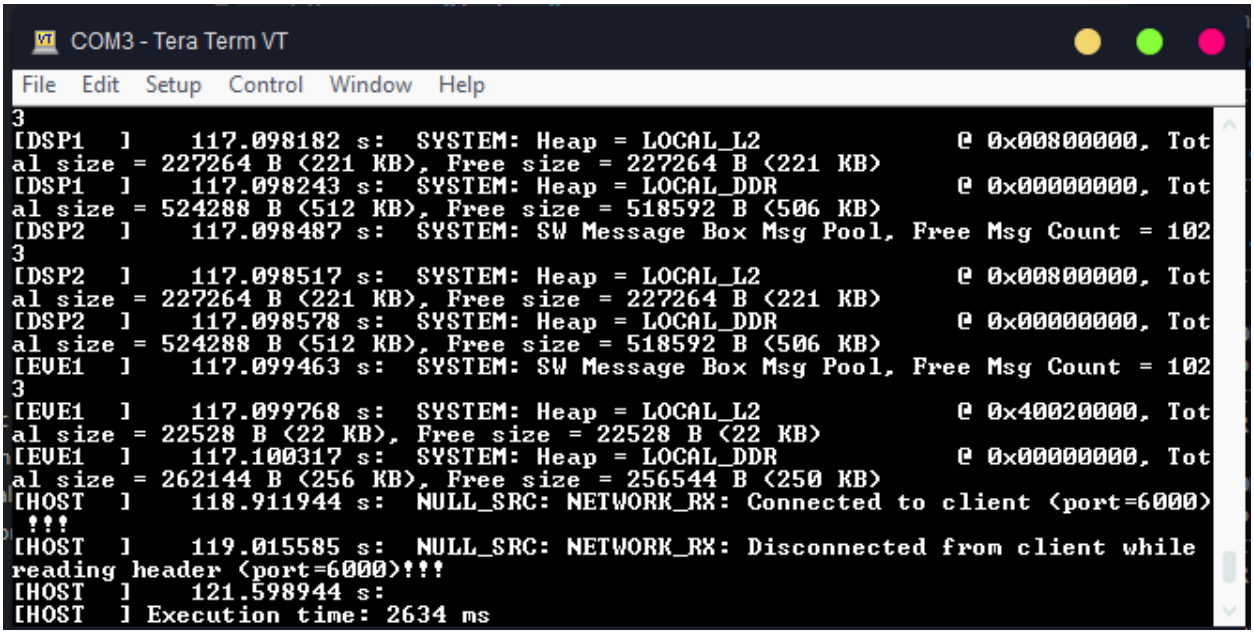
Slika 3.25. *Početni izbornik Alpha ploče u programu TerraTerm*



Slika 3.26. *Izbor metode interpolacije slike na Alpha ploči*

Tablica 3.4. *Upute za slanje slike na Alpha ploču i snimanje rješenja s Alpha ploče*

KORAK	RADNJA
1	Pokrenu se dvije odvojene instance Windows CMD-a
2	Unutar obje instance, pozicionira se u mapu C:\VISION_SDK_02_12_01_00\vision_sdk\tools\network_tools\bin
3	U jednoj od instanci se pokrene naredba *network_rx --host_ip 192.168.10.1 --target_ip 192.168.10.2 --port 7000 --files output.bin* koja čeka završetak događaja na Alpha ploči. Ova naredba se prva pokreće. Snimljena slika bit će pohranjena u datoteku *output.bin*
4	Pokrene se željena metoda interpolacije koja je prikazana na izborniku prema slici 3.24. Program čeka sliku na Ethernet portu.
5	U drugoj instanci se pokrene naredba *network_tx --host_ip 192.168.10.1 --target_ip 192.168.10.2 --no_loop –files DummyFile.bin* gdje je *DummyFile.bin* datoteke (slika) koja se šalje na ploču. Ova se naredba pokreće kao druga po redu. Naredba šalje sliku s računala putem Ethernet porta na ploču.
6	Nakon uspješne obrade slike, u terminalu se ispisuje vrijeme obrade, a primjer je prikazan na slici 3.27.
7	Mogu se zatvoriti obje instance Windows CMD-a, rješenje je dostupno unutar mape C:\VISION_SDK_02_12_01_00\vision_sdk\tools\network_tools\bin u datoteci output.bin



Slika 3.27. *Primjer završetka obrade slike odabranom metodom interpolacije*

4. TESTIRANJE RADA IMPLEMENTIRANIH METODA ZA INTERPOLACIJU SLIKE NA REALNOJ ADAS PLATFORMI

U ovom poglavlju bit će opisan način evaluacije rada implementiranih metoda za interpolaciju slike, tj. interpolacije slike metodom najbližih susjeda, bilinearnom interpolacijom i bikubičnom interpolacijom. Prvo će biti opisan skup slika korištenih prilikom testiranja, a zatim će biti objašnjena evaluacija rada pojedinih metoda interpolacije. Evaluacija se vršila na računalu specifikacija: procesor Intel® Core™ i7-9750H CPU @ 2.60GHz, RAM memorija 16GB DDR4, računalo sadrži brzi PCie NVMe SSD kapaciteta 512GB i grafičku karticu NVIDIA GeForce RTX 2070 Max-Q Design. Korištena tehnologija za izvođenje programskog jezika C je *Visual Studio 2019* i njegov ugrađeni kompajler za *C++*. Kompajler radi i s C programskim jezik uz promjenu ekstenzije izvršne datoteke iz *.cpp* u *.c*. Prvo testiranje se provodilo na zadanom skupu slika za prvu neoptimiziranu verziju rješenja. Sljedeća verzija rješenja je uz provedenu optimizaciju, vršilo se drugo testiranje na zadanom skupu slika. Na Alpha ploči se koristilo neoptimizirano rješenja. Više o ovoj temi u diskusiji o rješenjima u potpoglavlju 4.6.

4.1. Opis skupa testnih slika

U svrhu evaluacije rada različitih metoda interpolacije, korišten je skup od 50 slika. Slike su podijeljene u pet različitih kategorija na osnovu njihovog sadržaja. U svakoj kategoriji se nalazi po 10 različitih slika. Pojedine kategorije su opisane u tablici 4.1. Slike se nalaze u elektroničkom prilogu P. 4.1.

Tablica 4.1. *Kategorije slika korištenih prilikom evaluacije rješenja*

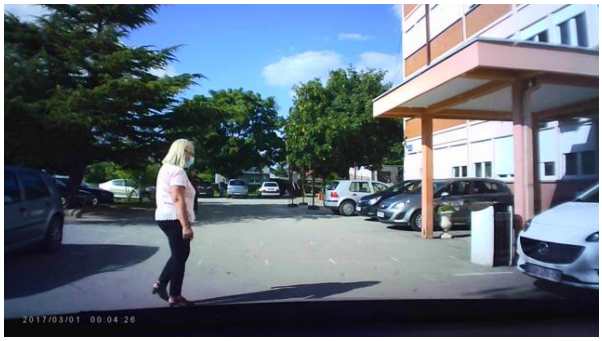
KATEGORIJA	OPIS
1	Skup slika snimljenih fiksnom kamerom (rezolucije 12 mega piksela) za nadzor vožnje montiranoj na prednjem staklu automobila. Na slikama je otisnut vremenski žig iz 2017. godine, no slike su nastale 2021. godine, a vremenski žig nije dobro postavljen. Slike iz ove kategorije su snimljene u Županijskoj ulici u Osijeku. Slike su dimenzija 640×360 elemenata slike. Motiv slika predstavlja realnu situaciju u gustom gradskom prometu.
2	Skup slika snimljenih kamerom za nadzor vožnje na Trpinjskoj cesti na ulazu u grad Vukovar. Motiv slika predstavlja cestu izvan naseljenog mjesta.

3	Skup slika snimljenih kamerom za nadzor vožnje ispred gradskog Doma zdravlja u gradu Županji. Motiv slika predstavlja realnu situaciju u gustom prometu uz prisutnost pješaka na cesti.
4	Skup slika iz prirode i okoliša različitog sadržaja, s puno i malo detalja na samoj slici. Slike su pronađene u bazi [25] slika za testiranje digitalne obrade slike.
5	Skup umjetno generiranih slika koje su izrađene u programu Bojanje (engl. *Paint*). Slike sadrže malo detalja uz nagle prijelaze intenziteta boje, što je pogodno za evaluaciju rada metoda interpolacija.

Zbog ograničenja VSDK, sve slike su zapisane kako je navedeno u potpoglavlju 3.4. (slika 3.6.) Po jedan primjer slike iz svake kategorije dan je na slici 4.1.

a)

b)



c)

d)

e)

Slika 4.1. *Primjer testne slike iz kategorije a) 1, b) 2, c) 3, d) 4, e) 5*

4.2. Način provođenja testiranja na Alpha ploči

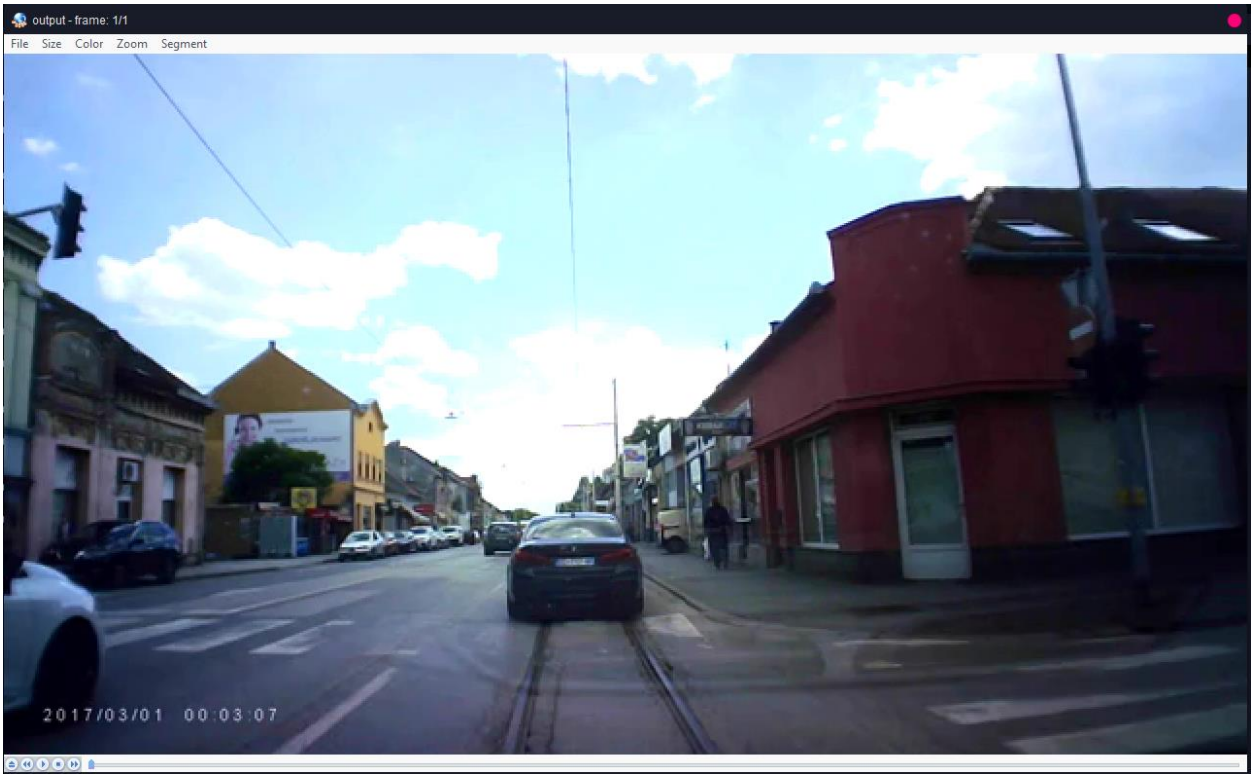
Način provođenja testiranja biti će objašnjen na primjeru interpolacije metodom najbližih susjeda. Analiza rada pojedine metode interpolacije vršila se na kompletnom skupu od 50 testnih slika i to za različite implementacije, tj. za implementacije kada se interpolacija izvršavala korištenjem samo procesora DSP1, samo procesora A15 i istovremenim korištenjem procesora DSP1 i DSP2. Procesori EVE1, EVE2, EVE3 i EVE4 su izostavljeni iz analize budući da nisu predviđeni za provođenje slučajeva korištenja i algoritama, a služe isključivo u svrhu provođenja gotovih algoritama koji dolaze u sklopu programskog paketa VSDK.

Svaka slika se šalje na Alpha ploču putem Ethernet veze, a rezultat vremena izvođenja je prikazan na emulatoru terminala. Rezultati interpolacije na ploči se uspoređuju s rezultatima izvođenja istog algoritma na računalu u programskom jeziku C, kako bi se provjerila točnost implementacije na samoj ploči, tj. kako bi se utvrdilo da rješenje obavlja identičan zadatak i pokretanjem na PC-u i pokretanjem na ploči. Ukoliko se prilikom oduzimanja rezultata (interpolirane slike) dobivenog na računalu u programskom jeziku C i rezultata (interpolirane slike) dobivenog na Alpha ploči dobije slika koja sadrži isključivo nule, odnosno zelena slika budući da se radi o YUV formatu boja, rezultati se smatraju točnima, tj. rezultati dobiveni na računalu u programskom jeziku C i na Alpha ploči se podudaraju. Korišteni alat za prikaz YUV slika je *yuvplayer* [26], a radi se o besplatnom alatu otvorenog kôda i dostupan je putem platforme github. Alat se nalazi unutar elektroničkog priloga P. 4.2.

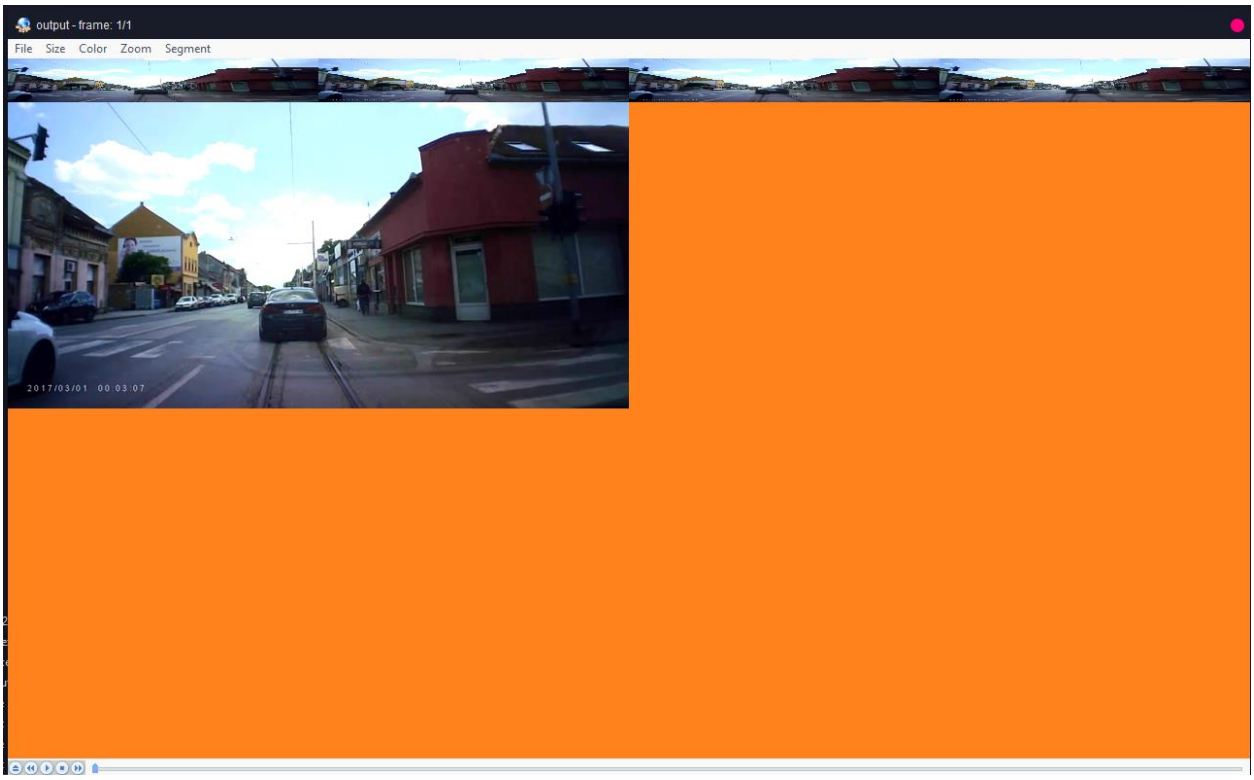
Za samu evaluaciju, potrebno je izvući informacije iz slike kako je objašnjeno u potpoglavlju 3.4. Primjer ulazne slike se nalazi na slici 3.7. Efektivni dio slike, odnosno dio slike koji je potrebno povećati ili smanjiti se nalazi unutar 640×360 elemenata slike počevši od koordinate (0, 0). Prilikom testiranja, na ploču se šalju slike dimenzija 640×360 elemenata slike na način opisan

u potpoglavlju 3.4., a rezultati se zapisuju u četiri puta većoj rezoluciji, odnosno 1280×720 elemenata slike i četiri puta manjoj rezoluciji, odnosno 320×180 elemenata slike. Primjer izlazne slike u četiri puta većoj rezoluciji se nalazi na slici 4.2. Bitno je za naglasiti da prilikom prikazivanja rezultante slike u programu *yuvplayer* dobivene na procesorima A15 i DSP1, izlaz će izgledati kako je prikazano na slici 4.3 ukoliko se postavi rezolucija u alatu *yuvplayer* na 1280×720 elemenata slike. Izlaz sam po sebi nema smisla, no smanjivanjem rezolucije na 320×180 elemenata slike unutar samog alata *yuvplayer* dobije se jasna slika, kako je prikazano na slici 4.4.

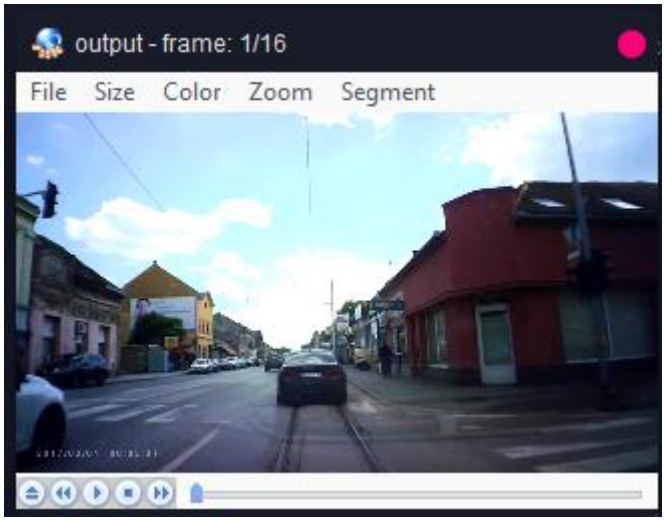
Prilikom evaluacije smanjenje slike dimenzija 320×180 elemenata slike, a unatoč ograničenjima navedenim u potpoglavlju 3.4., iz izlaznog spremnika ukupne veličine 1.843.200 bita, potrebno je pročitati prvih 460.800 bita koji sadrže smanjenu sliku. Istovremeno korištenje procesora DSP1 i DSP2 daje izlaznu sliku kako je prikazano na slici 4.5.



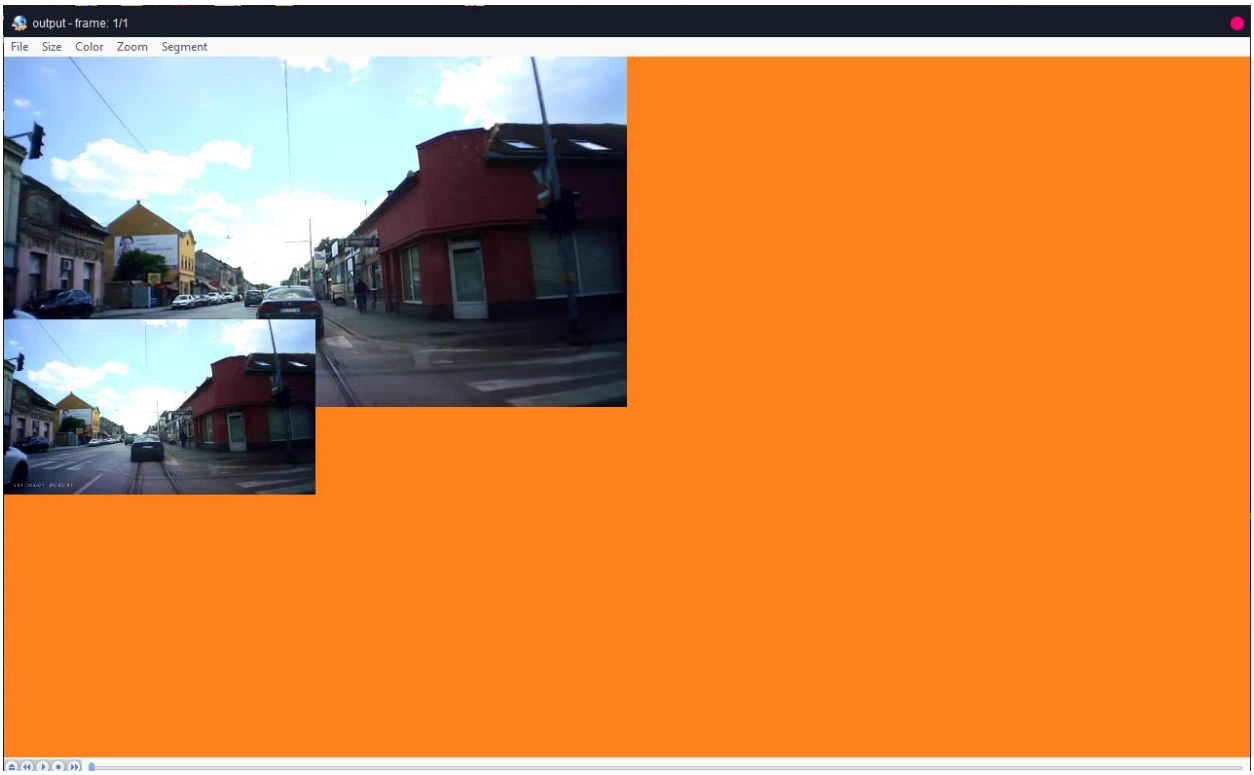
Slika 4.2. *Primjer izlazne slike prilikom skaliranja slike 640×360 → 1280×720 elemenata slike koristeći interpolaciju metodom najbližih susjeda*



Slika 4.3. *Primjer izlazne slike prilikom skaliranja slike 640×360 → 320×180 elemenata slike kada je rezolucija u alatu yuvplayer postavljena na 1280×720 elemenata slike koristeći interpolaciju metodom najbližih susjeda*



Slika 4.4. *Primjer izlazne slike prilikom skaliranja slike 640×360 → 320×180 elemenata slike kada je rezolucija u alatu yuvplayer postavljena na 320×180 elemenata slike koristeći interpolaciju metodom najbližih susjeda*



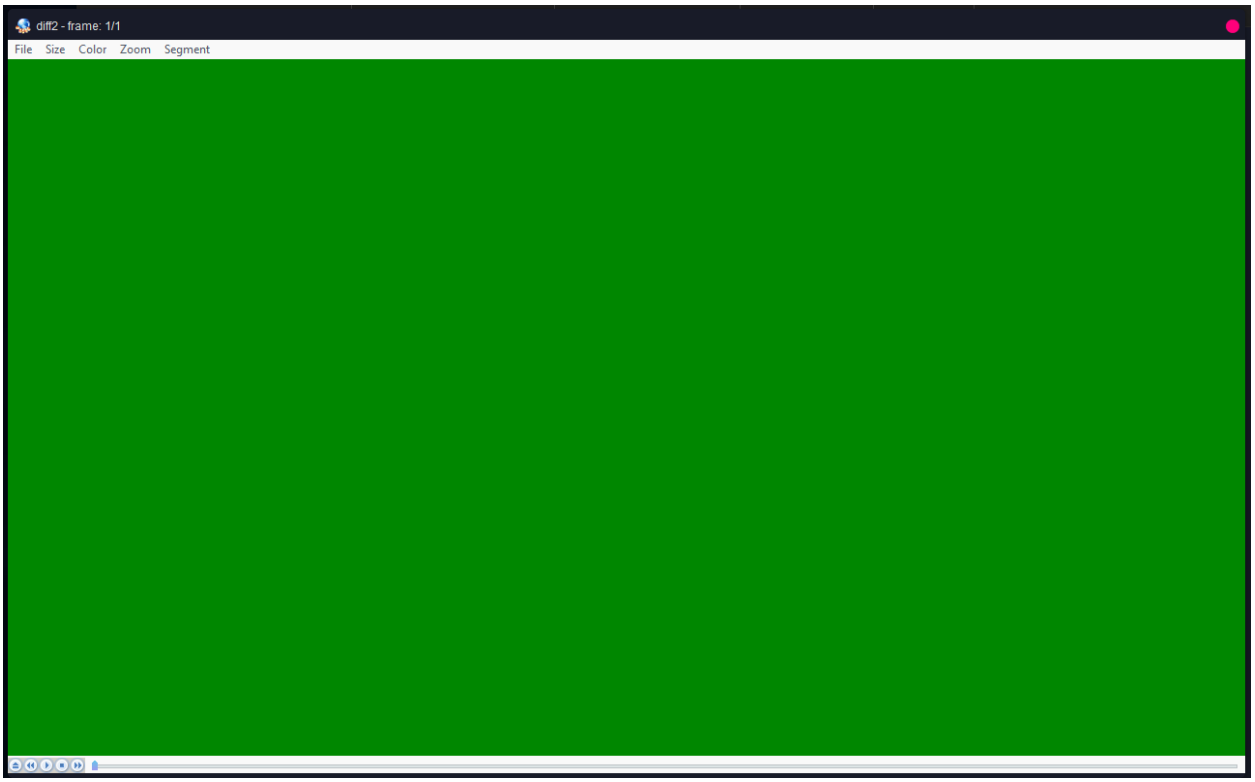
Slika 4.5. *Primjer izlazne slike prilikom skaliranja 640×360 → 320×1280 prilikom istovremenog korištenja procesora DSP1 i DSP2 za interpolaciju metodom najbližih susjeda*

Princip evaluacije rezultata rješenja prilikom smanjivanja i povećavanja rezolucije jednak je i kod bilinearne interpolacije i bikubične interpolacije. Ispravnost pojedinih rješenja se provjeravala na računalu u programskom jeziku C na način da se uzimala slika dobivena na računalu u programskom jeziku C i slika dobivena na Alpha ploči. Dio programskog koda koji je korišten prilikom provjere ispravnosti rada rješenja implementiranih na ploči nalazi se na slici 4.6.

```
1.      for (int i = 0; i < size; i++)
2.      {
3.           output[i] = bufferIn1[i] - bufferIn2[i];
4.      }
```

Slika 4.6. *Programski kod korišten za provjeru ispravnosti rada rješenja implementiranih na Alpha ploči*

U varijablu *bufferIn1* se učitava prva slika, a u *bufferIn2* se učitava druga slika. Rezultat oduzimanja se sprema u varijablu *output*. Varijabla *output* se zapisuje u binarnu datoteku koja se sprema na računalo. Ista datoteka se učitava i prikazuje unutar alata *yuvplayer*. Primjer prikaza pohranjenih vrijednosti u varijablu *output* se nalazi na slici 4.7.



Slika 4.7. *Slika dobivena oduzimanjem slike dobivene u programskom jeziku C na računalu i na Alpha ploči korištenjem interpolacije metodom najbližih susjeda – potvrda da rješenje implementirano na ploči daje rezultat identičan onome implementiranom na PC-u*

Usporedba je provedena na 50 slika, svakoj slici se četverostruko povećavala rezolucija i četverostruko smanjivala rezolucija za interpolaciju metodom najbližih susjeda, bilinearnom interpolacijom i bikubičnom interpolacijom. Ukupno je provedeno 900 iteracija (50 slika × 3 metode × 2 rezolucije × 3 tipa testiranja – jedno na računalu prije optimizacije, jedno na računalu poslije optimizacije, jedno na Alpha ploči bez optimizacije). Rezultati za sve slike su pokazali da je implementacija u programskom jeziku C na računalu identična implementaciji na Alpha ploči.

4.3. Rezultati testiranja predloženog rješenja na računalu u programskom jeziku C prije optimizacije

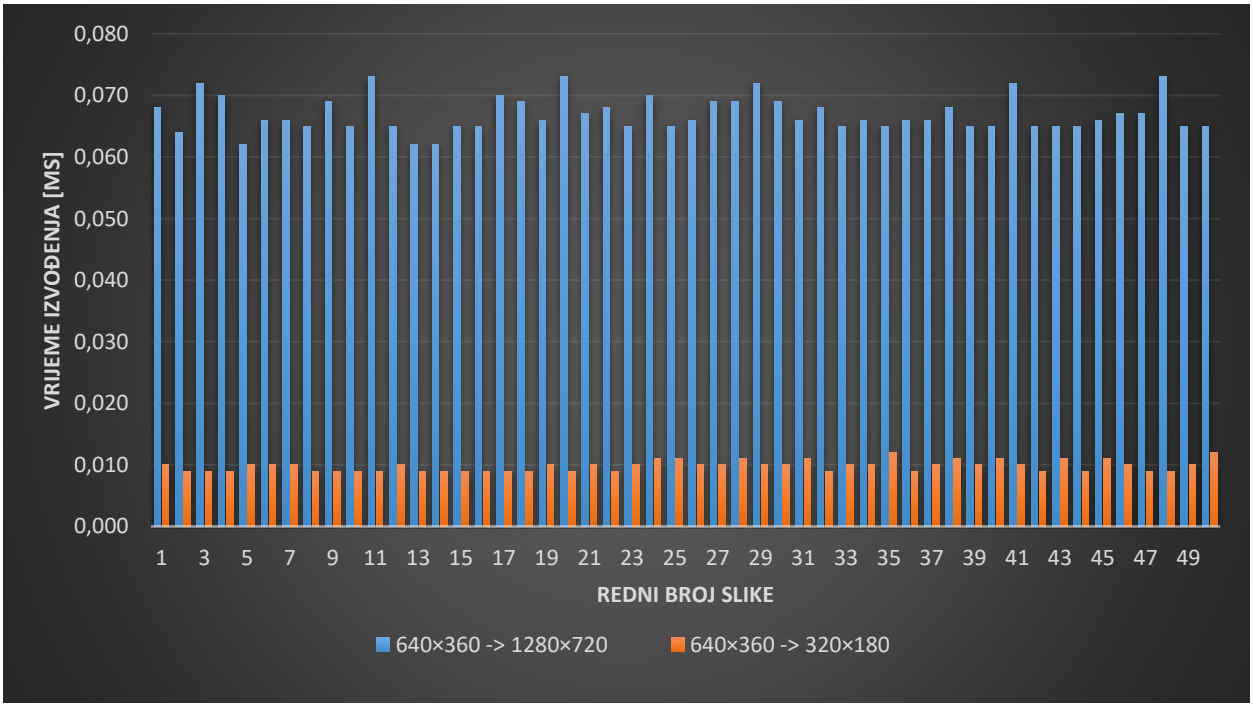
U ovom potpoglavlju bit će predstavljena analiza vremena izvođenja interpolacije na predloženim testnim slikama na računalu u programskom jeziku C prije izvedene optimizacije spomenute u potpoglavlju 3.7.

4.3.1. Interpolacija metodom najbližih susjeda

U tablici u prilogu P. 4.3. nalaze se vremena izvođenja interpolacije metodom najbližih susjeda prilikom izvođenja na računalu u programskom jeziku C bez optimizacije algoritma. Srednje vrijeme izvođenja prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike je 0,06694 sekundi, a standardna devijacija je 0,00280 sekundi. Prilikom smanjivanje rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja iznosi 0,00982 sekundi, a standardna devijacija iznosi 0,00084 sekundi. U tablici 4.2. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Podaci iz tablice iz priloga P. 4.3. se mogu prikazati grafički (slika 4.8.).

Tablica 4.2. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj interpolacije metodom najbližih susjeda na računalu bez optimizacije rješenja*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	37	74	40	80
$[\bar{x} \pm 2\sigma]$	47	94	48	96
$[\bar{x} \pm 3\sigma]$	50	100	50	100



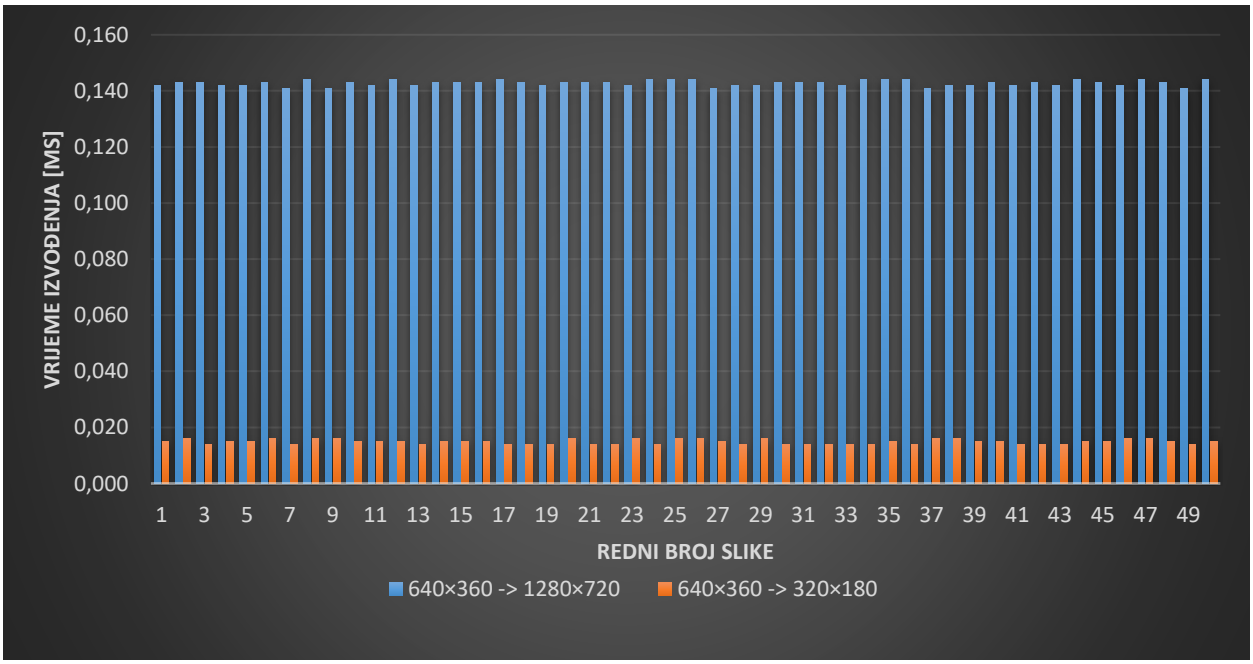
Slika 4.8. *Grafički prikaz vremena izvođenja interpolacije metodom najbližih susjeda na računalu u programskom jeziku C bez provedene optimizacije za skup od 50 testnih slika*

4.3.2. Bilinearna interpolacija

Prilog P. 4.4. sadrži tablicu u kojoj se nalaze vremena izvođenja bilinearne interpolacije prilikom izvođenja na računalu u programskom jeziku C bez optimizacije algoritma. Srednje vrijeme izvođenja je 0,14274 sekundi, a standardna devijacija je 0,00093 sekundi prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elementa slike na 1280×720 elemenata slike. Kada se rezolucija smanjuje četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja je 0,01486 sekundi, a standardna devijacija je 0,00080 sekundi. U tablici 4.3. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Grafički prikaz podataka iz tablice iz priloga P. 4.4. se nalazi na slici 4.9.

Tablica 4.3. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj bilinearne interpolacije na računalu bez optimizacije rješenja*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	33	66	17	34
$[\bar{x} \pm 2\sigma]$	50	100	50	100
$[\bar{x} \pm 3\sigma]$	50	100	50	100



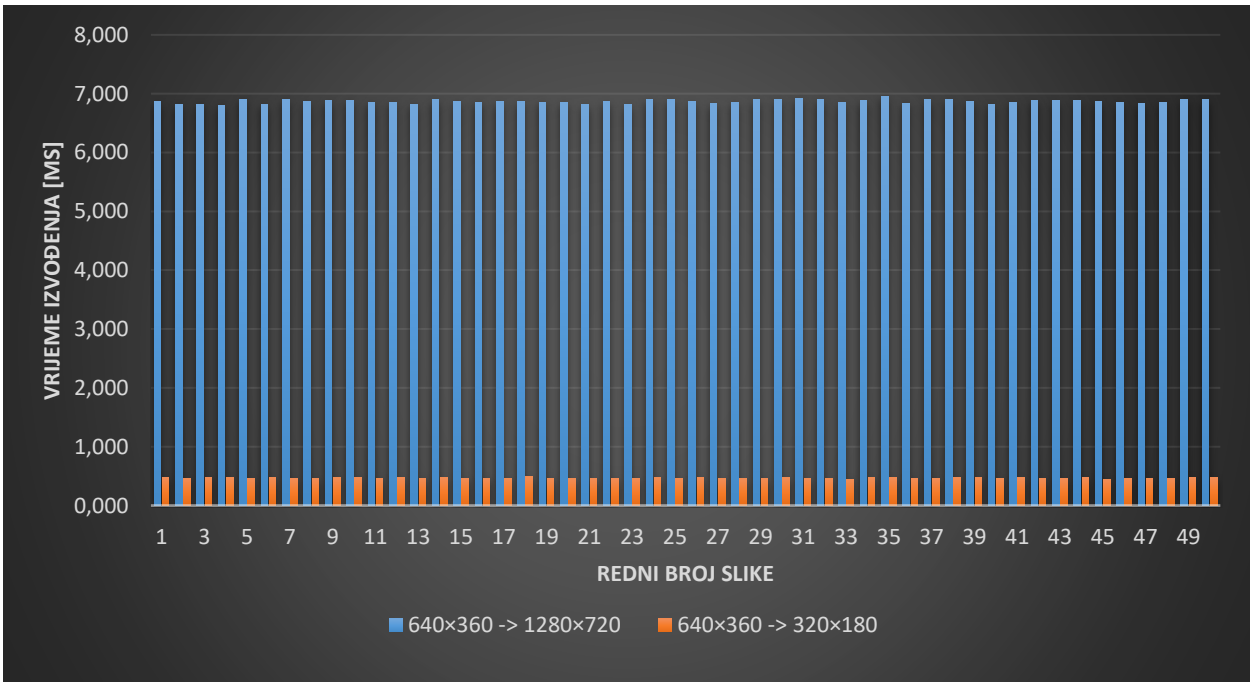
Slika 4.9. *Grafički prikaz vremena izvođenja bilinearne interpolacije na računalu u programskom jeziku C bez provedene optimizacije za skup od 50 testnih slika*

4.3.3. Bikubična interpolacija

U tablici u prilogu P. 4.5. se nalaze vremena izvođenja bikubične interpolacije prilikom izvođenja na računalu u programskom jeziku C bez optimizacije algoritma. Prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elementa slike, srednje vrijeme izvođenja je 6,86418 sekundi, a standardna devijacija je 0,03247 sekundi. Kada se rezolucija slike smanjuje četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja iznosi 0,46718 sekundi, a standardna devijacija 0,00869 sekundi. U tablici 4.4. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Podatke iz tablice iz priloga P. 4.5. je moguće prikazati grafički (slika 4.10.)

Tablica 4.4. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj bikubične interpolacije na računalu bez optimizacije rješenja*

	Povećanje rezolucije (640×360 →1280×720)		Smanjenje rezolucije (640×360 → 320×180)	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	32	64	35	70
$[\bar{x} \pm 2\sigma]$	48	96	48	96
$[\bar{x} \pm 3\sigma]$	50	100	50	100



Slika 4.10. *Grafički prikaz vremena izvođenja bikubične interpolacije na računalu u programskom jeziku C bez provedene optimizacije za skup od 50 testnih slika*

4.4. Rezultati testiranja predloženog rješenja na računalu u programskom jeziku C poslije optimizacije

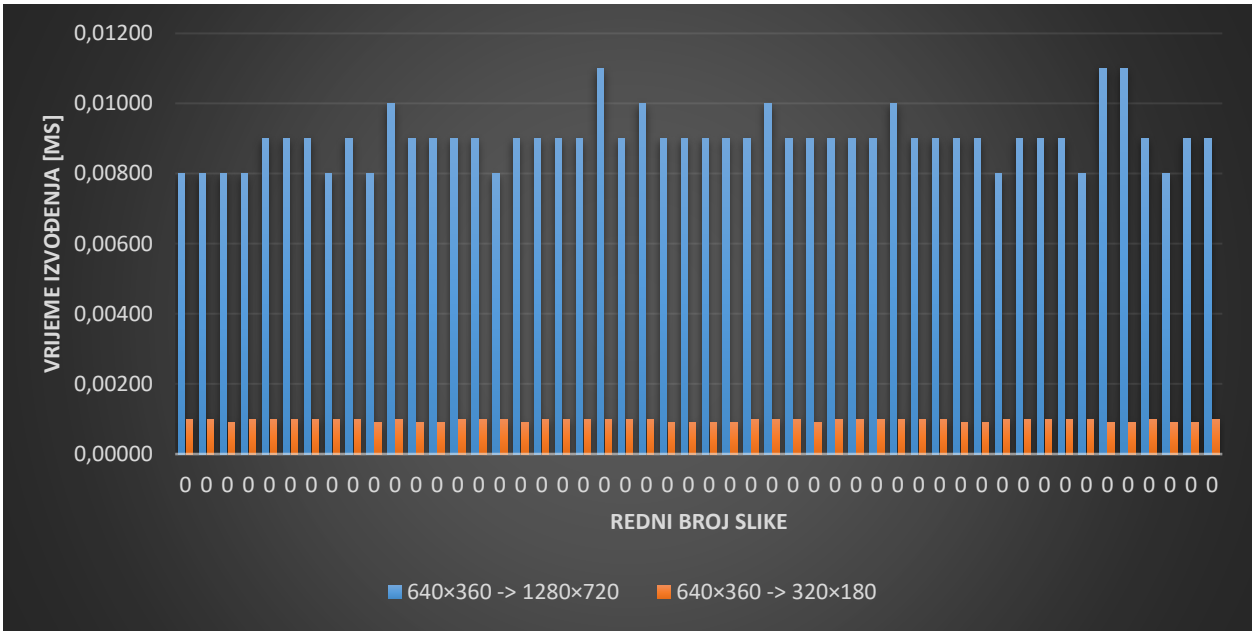
U ovom potpoglavlju bit će predstavljena analiza vremena izvođenja interpolacije na predloženim testnim slikama za metodu najbližih susjeda, bilinearnu interpolaciju i bikubičnu interpolaciju na računalu u programskom jeziku C uz provedenu optimizaciju.

4.4.1. Interpolacija metodom najbližih susjeda

U tablici u prilogu P. 4.6. se nalaze vremena izvođenja interpolacije metodom najbližih susjeda uz izvođenje na računalu u programskom jeziku C s optimiziranim algoritmom. Srednje vrijeme izvođenja prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike iznosi 0,00900 sekundi, a standardna devijacija je 0,00072 sekundi. Kada se rezolucija smanjuje četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja iznosi 0,00097 sekundi, a standardna devijacija 0,00005 sekundi. U tablici 4.5. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Podatke iz tablice iz priloga P. 4.6. je moguće prikazati grafički (slika 4.11.).

Tablica 4.5. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj interpolacije metodom najbližih susjeda na računalu uz provedenu optimizaciju rješenja*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	33	66	34	68
$[\bar{x} \pm 2\sigma]$	47	94	50	100
$[\bar{x} \pm 3\sigma]$	50	100	50	100



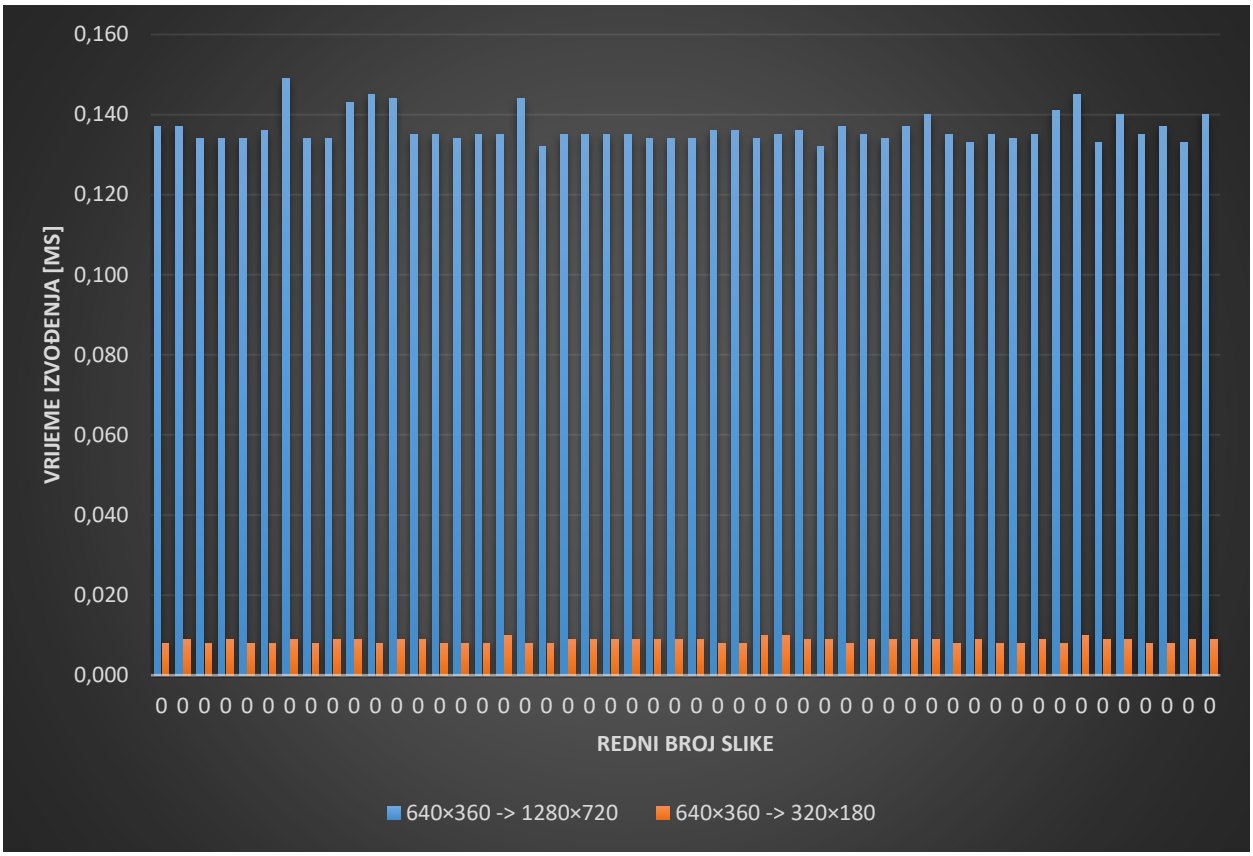
Slika 4.11. *Grafički prikaz vremena izvođenja interpolacije metodom najbližih susjeda na računalu u programskom jeziku C uz provedenu optimizaciju za skup od 50 testnih slika*

4.4.2. Bilinearna interpolacija

U tablici u prilogu P. 4.7. se nalaze vremena izvođenja bilinearne interpolacije uz izvođenje na računalu u programskom jeziku C s optimiziranim algoritmom. Srednje vrijeme izvođenja prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike iznosi 0,13642 sekundi, a standardna devijacija je 0,00374 sekundi. Kada se rezolucija smanjuje četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja iznosi 0,00868 sekundi, a standardna devijacija 0,00061 sekundi. U tablici 4.6. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Na slici 4.12. se nalazi grafički prikaz podataka tablice iz priloga P. 4.7.

Tablica 4.6. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj bilinearne interpolacije na računalu uz provedenu optimizaciju rješenja*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	41	82	26	52
$[\bar{x} \pm 2\sigma]$	45	90	46	92
$[\bar{x} \pm 3\sigma]$	49	98	50	100



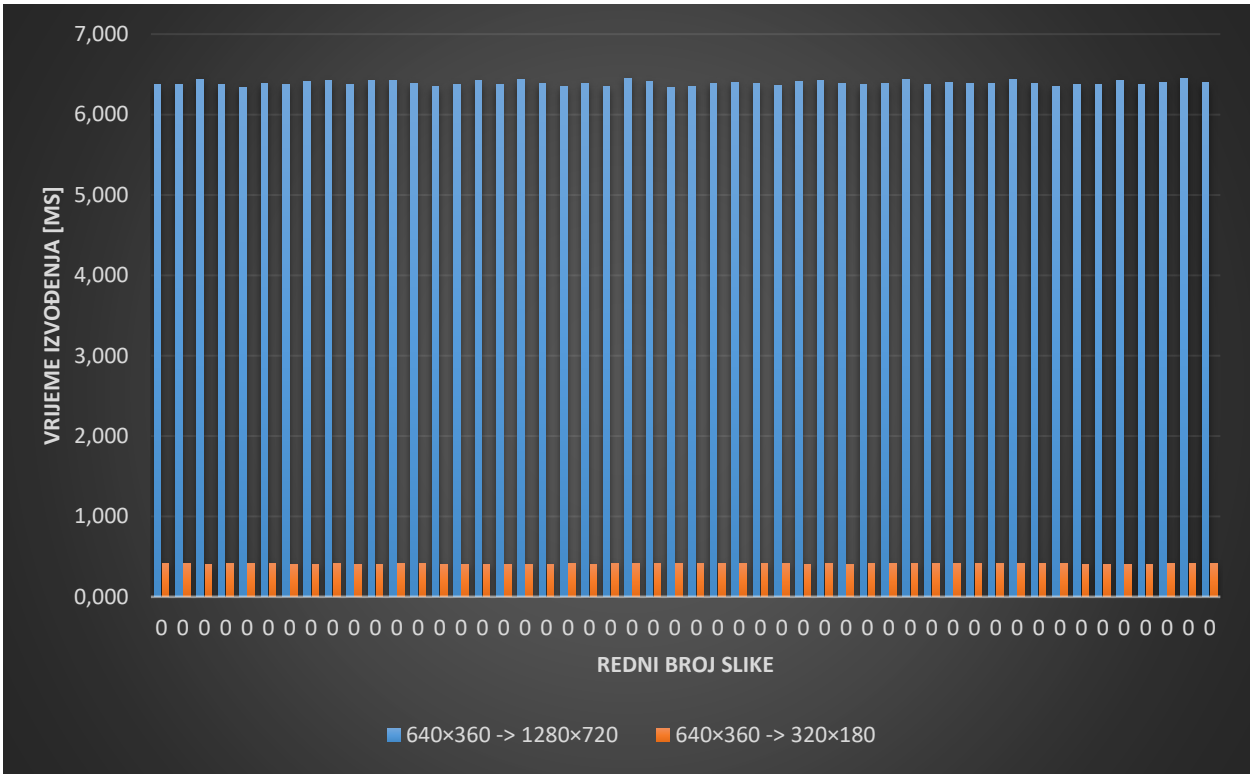
Slika 4.12. *Grafički prikaz vremena izvođenja bilinearne interpolacije na računalu u programskom jeziku C uz provedenu optimizaciju za skup od 50 testnih slika*

4.4.3. Bikubična interpolacija

U tablici u prilogu P. 4.8. se nalaze vremena izvođenja bilinearne interpolacije uz izvođenje na računalu u programskom jeziku C s optimiziranim algoritmom. Srednje vrijeme izvođenja prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike iznosi 6,39000 sekundi, a standardna devijacija je 0,02877 sekundi. Prilikom smanjivanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja iznosi 0,41000 sekundi, a standardna devijacija 0,00318 sekundi. U tablici 4.7. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Na slici 4.13. se nalazi grafički prikaz podataka tablice iz priloga P. 4.8.

Tablica 4.7. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj bikubične interpolacije na računalu uz provedenu optimizaciju rješenja*

Izvođenje na računalu u programskom jeziku C uz provedenu optimizaciju				
	Povećanje rezolucije (640×360 →1280×720)		Smanjenje rezolucije (640×360 → 320×180)	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	32	64	33	66
$[\bar{x} \pm 2\sigma]$	47	94	47	94
$[\bar{x} \pm 3\sigma]$	50	100	50	100



Slika 4.13. *Grafički prikaz vremena izvođenja bikubične interpolacije na računalu u programskom jeziku C uz provedenu optimizaciju za skup od 50 testnih slika*

4.5. Rezultati testiranja predloženog rješenja na Alpha ploči

Korištena rješenja za interpolaciju slike na Alpha ploči metodom najbližih susjeda, bilinearnom interpolacijom i bikubičnom interpolacijom su neoptimizrana rješenja s računala. Razlog nekorištenja optimiziranih verzija rješenja je što zbog nedovoljnog poznavanja izrazito složenog izvornog koda Alpha ploče nije moguće upravljati memorijom kao što je to moguće na

računalu u programskom jeziku C. Prilikom pokušaja korištenja optimiziranih rješenja na Alpha ploči, dolazi do nepoznatih logičkih pogrešaka povezanih s memorijom i pokazivačima u memoriji. Zbog navedenih razloga, u diskusiji o rješenjima koja slijedi u nastavku ovog rada bit će uspoređene neoptimizirane verzije rješenja na računalu u programskom jeziku C i rješenja na Alpha ploči.

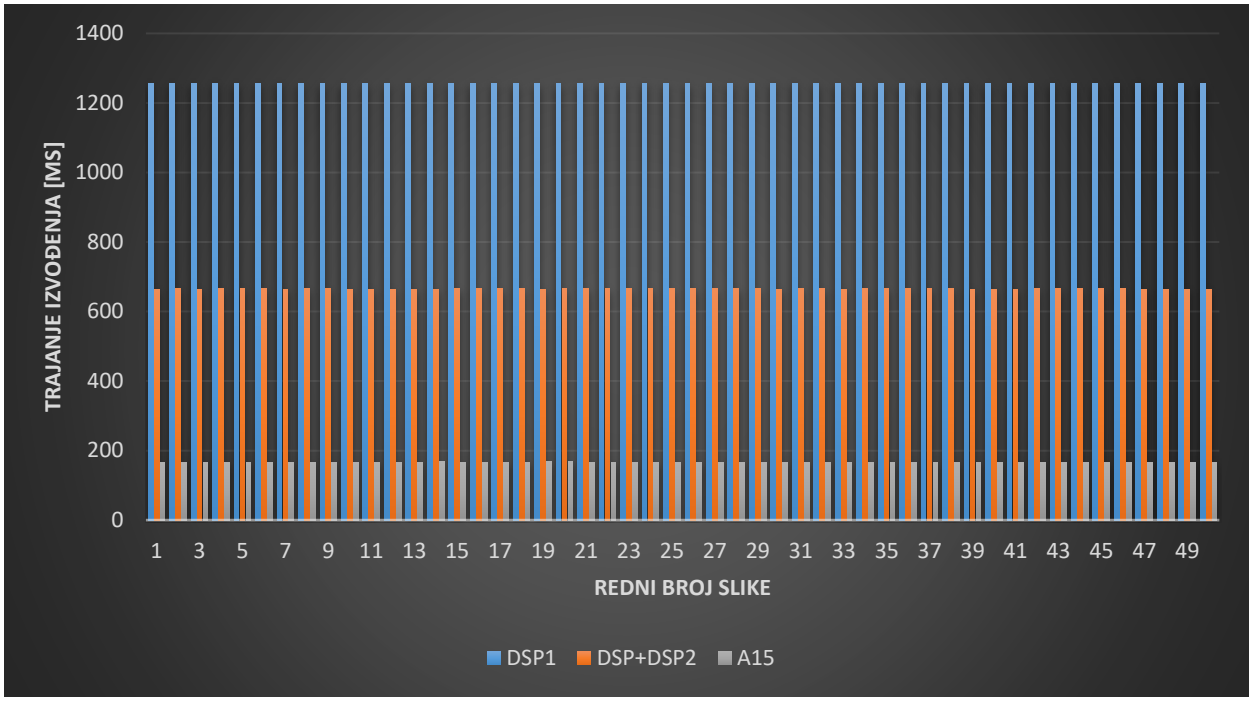
4.5.1. Interpolacija metodom najbližih susjeda

U tablici u prilogu P. 4.9. nalaze se vremena izvođenja interpolacije metodom najbližih susjeda na procesoru DSP1, na procesoru A15 i uz istovremeno korištenje procesora DSP1 i DSP2 za slučaj kada se četiri puta povećava rezolucija, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike i kada se četiri puta smanjuje rezolucija, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike. Iz tablice iz priloga P. 4.9. izračunavaju se srednje vrijednosti izvođenja te standardna devijacija uz svaku srednju vrijednost. Izračunati podaci se nalaze u tablici 4.8. za interpolaciju metodom najbližih susjeda.

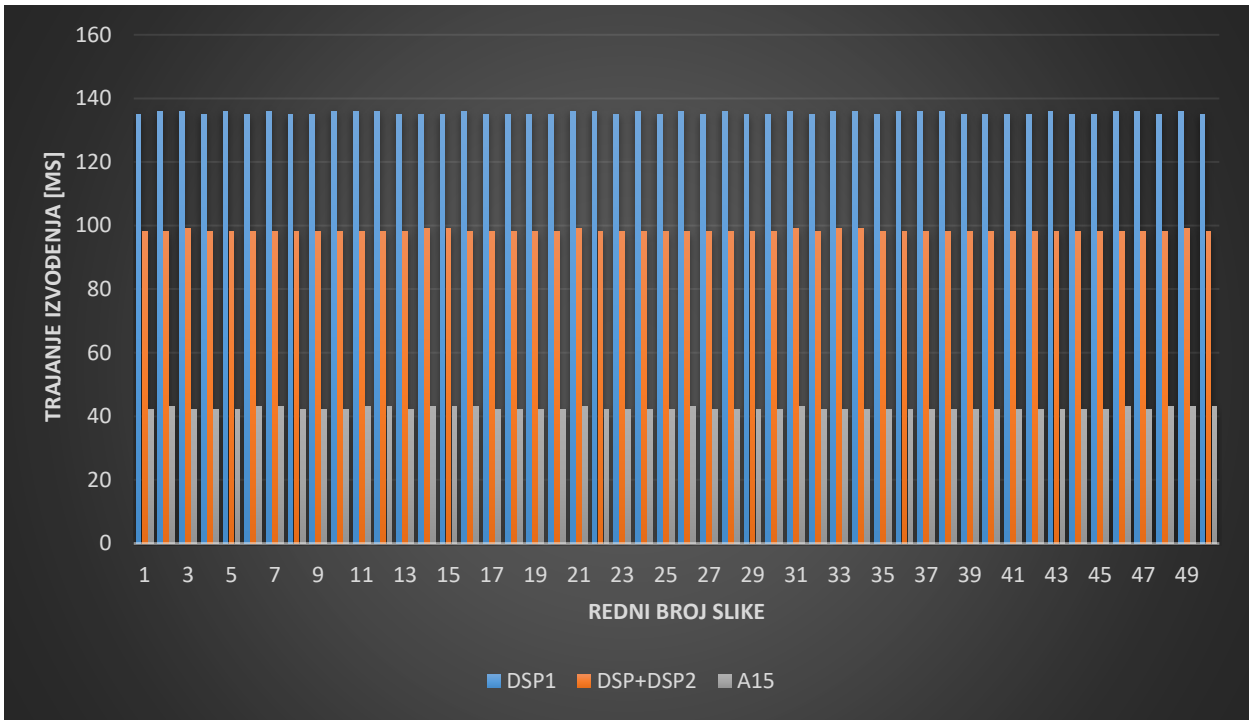
Tablica 4.8. *Srednja vrijednost i standardna devijacija vremena izvođenja interpolacije metodom najbližih susjeda koristeći različite procesore za provođenje postupka*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Procesor	**Srednja vrijednost vremena izvođenja $\bar{x}$ [ms]**	**Standardna devijacija vremena izvođenja σ [ms]**	**Srednja vrijednost vremena izvođenja $\bar{x}$ [ms]**	**Standardna devijacija vremena izvođenja σ [ms]**
DSP1	1255,22	0,41	135,46	0,50
DSP1-DSP2	664,90	0,78	98,16	0,37
A15	167,06	0,24	42,30	0,46

Podaci iz tablice iz priloga P.4.9. mogu se grafički prikazati. Grafički prikaz podataka vezanih za povećanje rezolucije četiri puta se nalaze na slici 4.14., a podaci vezani za smanjivanje rezolucije četiri puta se nalaze na slici 4.15.



Slika 4.14. *Grafički prikaz vremena izvođenja interpolacije metodom najbližih susjeda na Alpha ploči za skup od 50 testnih slika prilikom povećavanje rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i procesor A15*



Slika 4.15 *Grafički prikaz vremena izvođenja interpolacije metodom najbližih susjeda na Alpha ploči za skup od 50 testnih slika prilikom smanjivanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i procesor A15*

Pomoću tablice 4.9. moguće je odrediti broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja (tablica 4.9.).

Tablica 4.9. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja prilikom korištenja procesora DSP1, istovremeno korištenje procesora DSP1 i DSP2 i korištenje procesora A15 za povećavanje i smanjivanje rezolucije slike četiri puta*

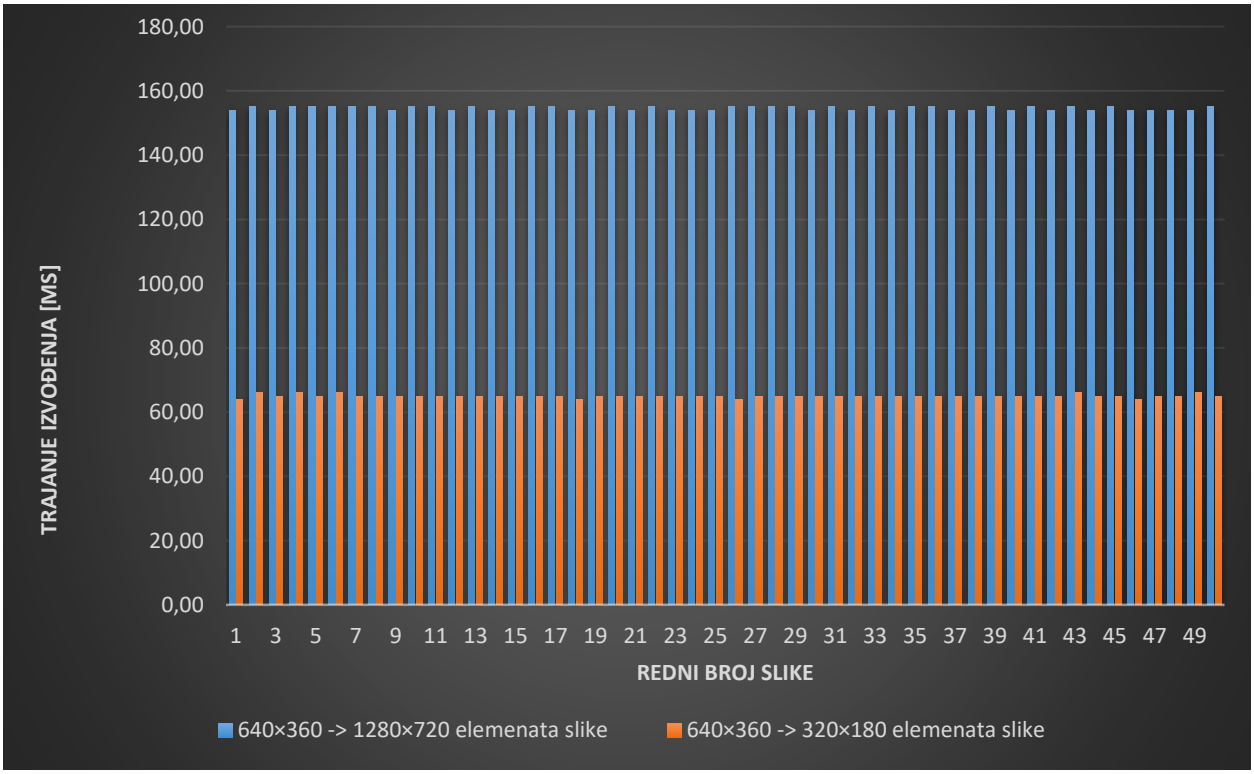
Procesor DSP1				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[$\bar{x} \pm \sigma$]	39	78	27	54
[$\bar{x} \pm 2\sigma$]	50	100	50	100
[$\bar{x} \pm 3\sigma$]	50	100	50	100
Procesori DSP1+DSP2				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[$\bar{x} \pm \sigma$]	19	38	42	84
[$\bar{x} \pm 2\sigma$]	50	100	42	84
[$\bar{x} \pm 3\sigma$]	50	100	50	100
Procesor A15				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[$\bar{x} \pm \sigma$]	47	94	35	70
[$\bar{x} \pm 2\sigma$]	47	94	50	100
[$\bar{x} \pm 3\sigma$]	47	94	50	100

Kako je naglašeno u dijelu 3.4.3., moguće je istovremeno koristiti procesore A15, DSP1 i DSP2. U tablici u prilogu P. 4.10. se nalaze vremena izvođenja interpolacije metodom najbližih susjeda prema podijeli navedenoj u dijelu 3.4.3., kada procesor A15 interpolira 90% slike, a procesori DSP1 i DSP2 po 5% slike za slučaj kada se rezolucije slike povećava četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike, te kada se četiri puta smanjuje rezolucija slike, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike. Tablica sadrži stupac *MAX(DSP1, DSP2, A15)* koji predstavlja vrijeme izvođenja interpolacije metodom najbližih susjeda prilikom istovremenog korištenja procesora DSP1, DSP2 i A15. Iz navedenog stupca može se izračunati srednje vrijeme izvođenja koje iznosi 154,52 ms prilikom navedenog povećanja

rezolucije te 65,02 ms prilikom navedenog smanjenja rezolucije. Standardna devijacija prilikom povećanja rezolucije iznosi 0,50 ms, a prilikom smanjenja rezolucije 0,42 ms. U tablici 4.10. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Podaci iz tablice iz priloga P. 4.10. se mogu prikazati grafički (slika 4.16). Vrijeme potrebno za spajanje dijelova interpolirane slike je manje od 1 ms te je zanemarivo.

Tablica 4.10. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj interpolacije metodom najbližih susjeda na Alpha ploči prilikom istovremenog korištenja procesora DSP1, DSP2 i A15*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[$\bar{x} \pm \sigma$]	26	52	41	82
[$\bar{x} \pm 2\sigma$]	50	100	41	82
[$\bar{x} \pm 3\sigma$]	50	100	50	100



Slika 4.16 *Grafički prikaz vremena izvođenja interpolacije metodom najbližih susjeda prilikom povećanja i smanjenja rezolucije slike četiri puta kada se istovremeno koriste procesori A15, DSP1 i DSP2 na skupu od 50 slika*

4.5.2. Bilinearna interpolacija

U tablici u prilogu P. 4.11. nalaze se vremena izvođenja bilinearne interpolacije za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i za procesor A15 za slučaj kada je četiri puta veća rezolucija, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike i kada je četiri puta manja rezolucija, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike. Iz tablice iz priloga P.4.10. izračunavaju se srednje vrijednosti izvođenja te standardna devijacija uz svaku srednju vrijednost. Izračunati podaci se nalaze u tablici 4.11. za bilinearnu interpolaciju. U tablici 4.12. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Podaci iz tablice iz priloga P. 4.11. mogu se prikazati grafički. Za povećavanje rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike, graf se nalazi na slici 4.17, a za smanjivanje rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, graf se nalazi na slici 4.18.

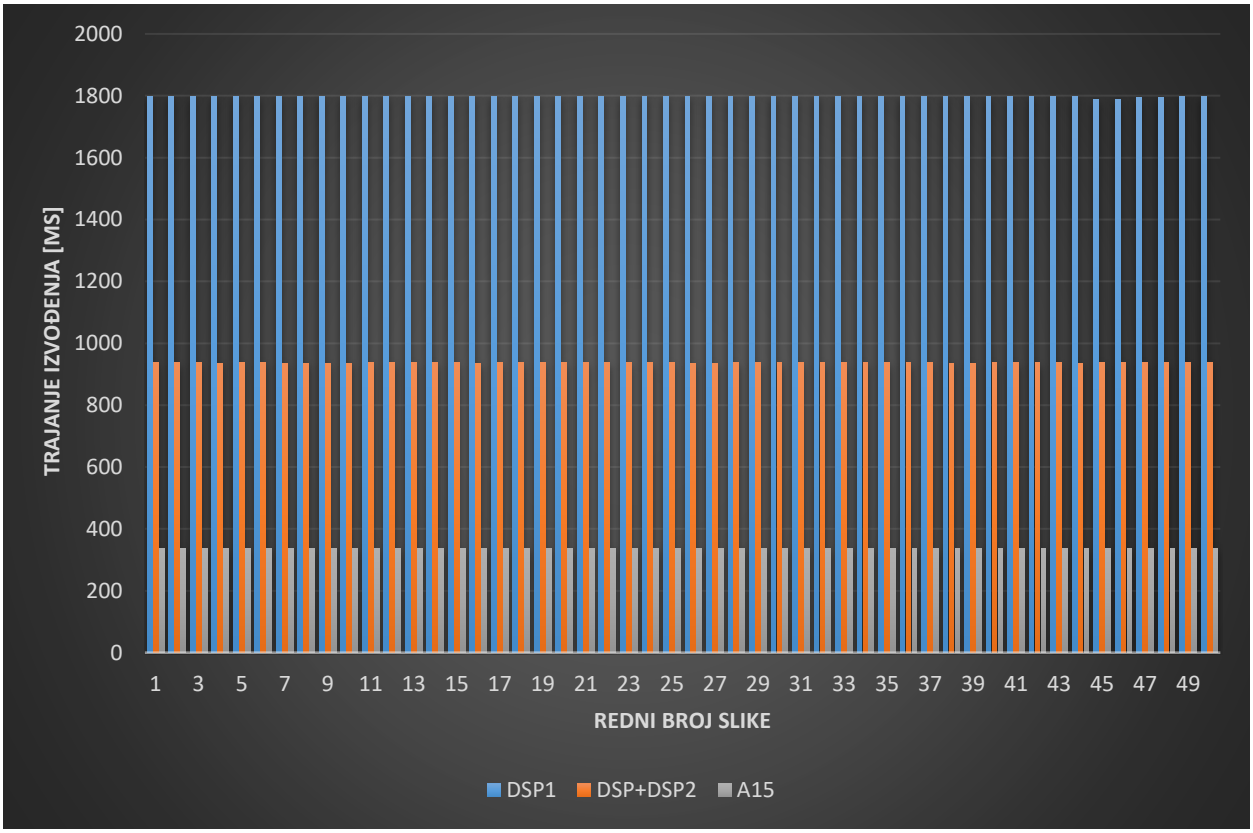
Tablica 4.11. *Srednja vrijednost i standardna devijacija vremena izvođenja bilinearne interpolacije koristeći različite procesore za provođenje postupka*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Procesor	**Srednja vrijednost vremena izvođenja $\bar{x}$ [ms]**	**Standardna devijacija vremena izvođenja σ [ms]**	**Srednja vrijednost vremena izvođenja $\bar{x}$ [ms]**	**Standardna devijacija vremena izvođenja σ [ms]**
DSP1	1797,58	2,10	169,76	0,43
DSP1-DSP2	936,78	0,41	116,78	0,41
A15	337,26	0,44	53,4	0,49

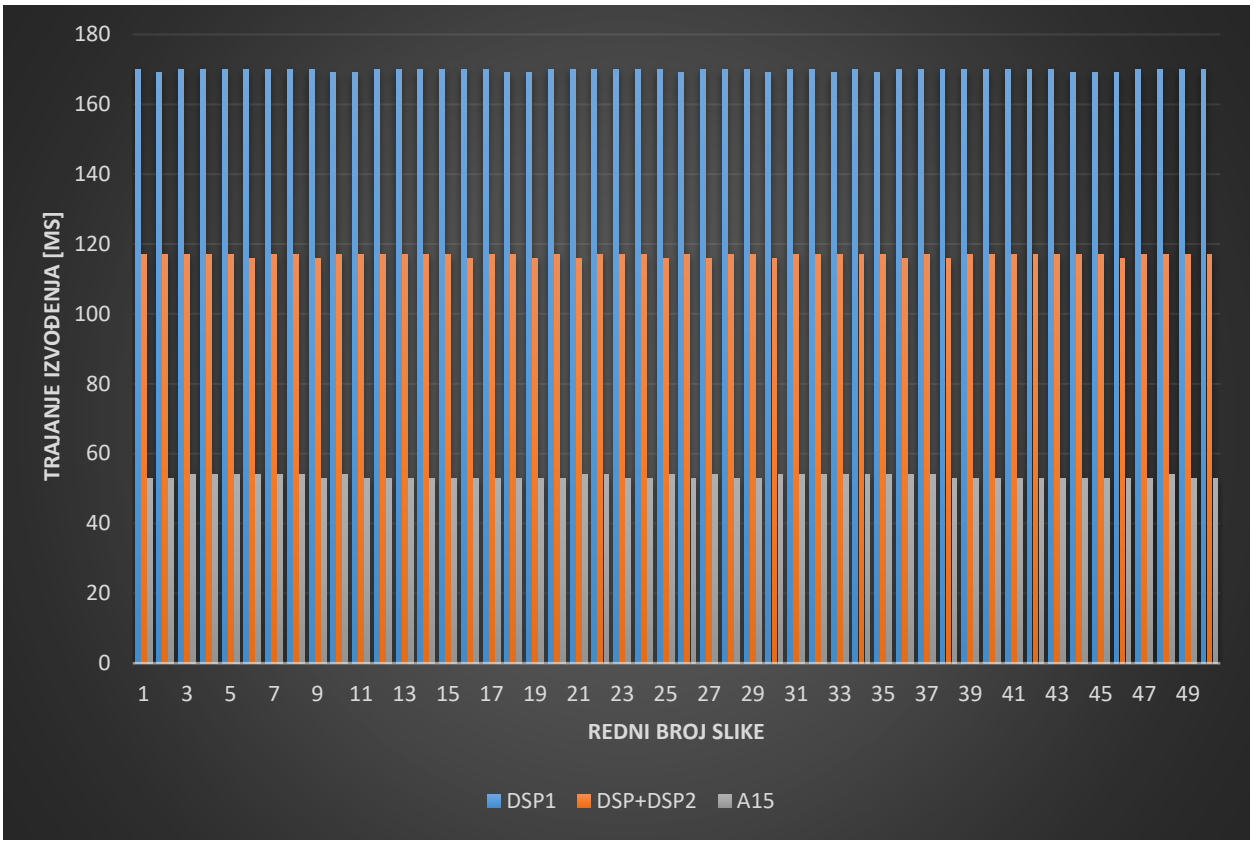
Tablica 4.12. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja prilikom korištenja procesora DSP1, istovremeno korištenje procesora DSP1 i DSP2 i korištenje procesora A15 za povećavanje i smanjivanje rezolucije slike četiri puta*

Procesor DSP1				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	47	94	38	76
$[\bar{x} \pm 2\sigma]$	48	96	50	100
$[\bar{x} \pm 3\sigma]$	48	96	50	100
Procesor DSP1+DSP2				

	Povećanje rezolucije (640×360 →1280×720)		Smanjenje rezolucije (640×360 → 320×180)	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[$\bar{x} \pm \sigma$]	39	78	39	78
[$\bar{x} \pm 2\sigma$]	50	100	50	100
[$\bar{x} \pm 3\sigma$]	50	100	50	100
Procesor A15				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[$\bar{x} \pm \sigma$]	37	74	30	60
[$\bar{x} \pm 2\sigma$]	50	100	50	100
[$\bar{x} \pm 3\sigma$]	50	100	50	100



Slika 4.17. *Trajanje izvođenja bilinearne interpolacije prilikom povećavanja rezolucije slike četiri puta, odnosno s 640×360 elemenata slike na 1280×720 elemenata slike za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i procesor A15*

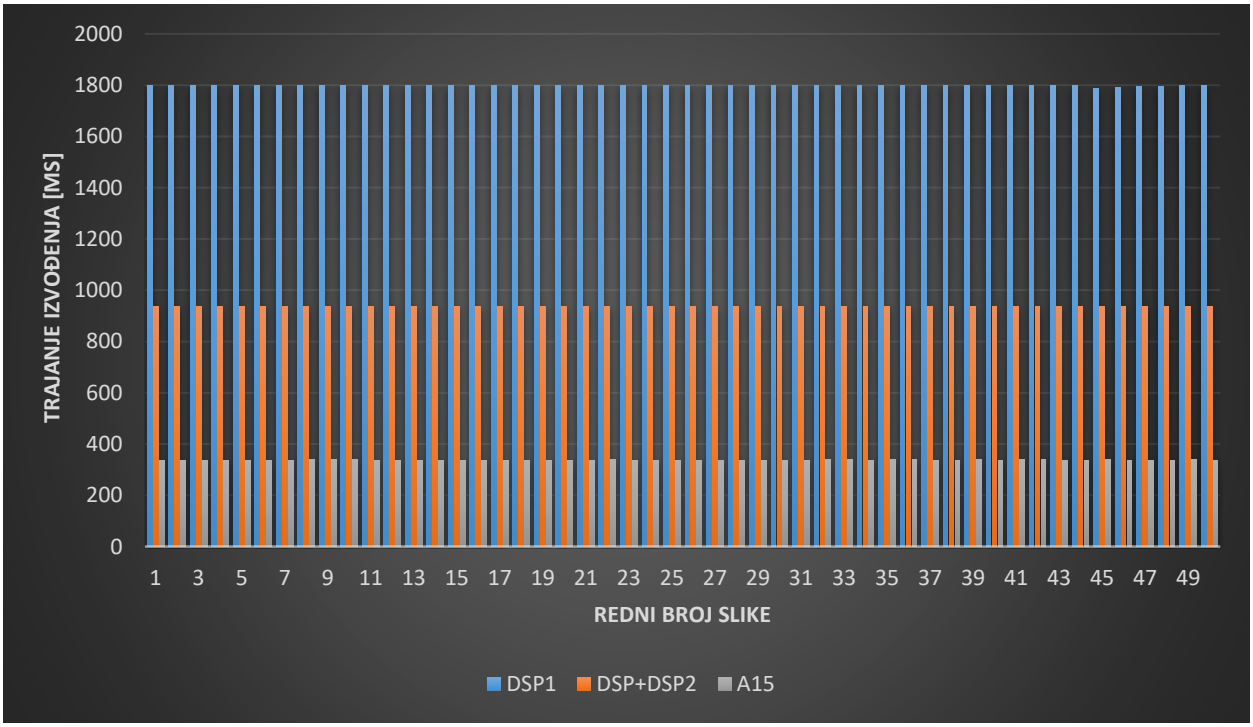


Slika 4.18. *Trajanje izvođenja bilinearne interpolacije prilikom smanjivanja rezolucije slike četiri puta, odnosno s 640×360 elemenata slike na 320×180 elemenata slike za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i procesor A15*

U tablici u prilogu P. 4.12. nalaze se vremena izvođenja bilinearne interpolacija kada se postupak interpolacije provodi istovremeno na procesorima A15, DSP1 i DSP2 (dio 3.5. ovog diplomskog rada). Izvođenje slučaja upotrebe utvrđen je optimalan postotak interpoliranja slike na pojedinom procesoru. Kako je navedeno u dijelu 4.5.1., u stupcu *MAX(A15, DSP1, DSP)* se nalazi vrijeme izvođenja bilinearne interpolacije prilikom istovremenog korištenja procesora DSP1, DSP2 i A15. Prilikom povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike, srednje vrijeme izvođenja iznosi 280,16 ms, a standardna devijacija 0,58 ms. Prilikom smanjenja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, srednje vrijeme izvođenja iznosi 75,78 ms, a standardna devijacija 0,50 ms. U tablici 4.13. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja prilikom istovremenog korištenja procesora A15, DSP1 i DSP2. Podaci iz tablice iz priloga P. 4.12. se mogu prikazati grafički (slika 4.19). Vrijeme potrebno za spajanje dijelova interpolirane slike je manje od 1 ms te je zanemarivo.

Tablica 4.13. *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja za slučaj bilinearne interpolacije na Alpha ploči prilikom istovremenog korištenja procesora DSP1, DSP2 i A15*

Istovremeno korištenje procesora A15, DSP1, DSP2				
	Povećanje rezolucije (640×360 →1280×720)		Smanjenje rezolucije (640×360 → 320×180)	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	38	76	35	70
$[\bar{x} \pm 2\sigma]$	47	94	48	96
$[\bar{x} \pm 3\sigma]$	48	96	50	100



Slika 4.19 *Grafički prikaz vremena izvođenja bilinearne interpolacije prilikom povećanja i smanjenja rezolucije slike četiri puta kada se istovremeno koriste procesori A15, DSP1 i DSP2 na skupu od 50 slika*

4.5.3. Bikubična interpolacija

U ovom potpoglavlju biti će predstavljena analiza vremena izvođenja bikubične interpolacije na predloženom skupu slika na Alpha ploči. U tablici iz priloga P. 4.13. se nalaze vremena izvođenja interpolacije metodom najbližih susjeda za procesor DSP1, za procesor A15 i

istovremeno korištenje procesora DSP1 i DSP2 za slučaj kada je četiri puta veća rezolucija, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike i kada je četiri puta manja rezolucija, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike. Iz tablice iz priloga P. 4.13. izračunavaju se srednje vrijednosti izvođenja te standardna devijacija uz svaku srednju vrijednost, a ti se podaci nalaze u tablici 4.14. Unutar tablice 4.15. se nalazi broj slika od ukupno 50 za koje se vrijeme izvođenja nalazi u intervalima od jedne, dviju ili triju standardnih devijacija oko srednjeg vremena izvođenja. Podaci iz tablice iz priloga P. 4.13. se mogu prikazati grafički. Grafički prikaz za slučaj povećavanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike se nalazi na slici 4.20, a za slučaj smanjivanja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike se nalazi na slici 4.21.

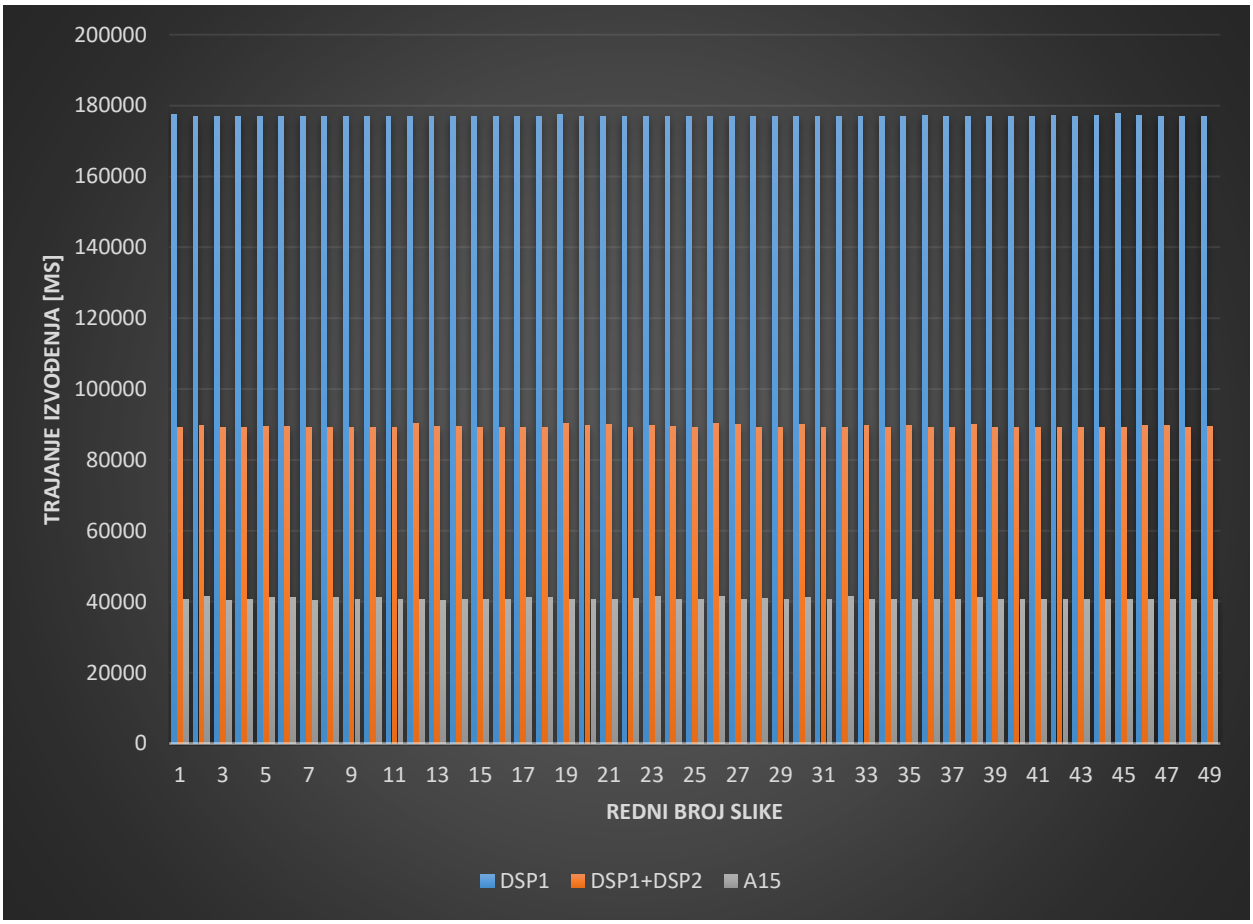
Tablica 4.14. *Srednja vrijednost i standardna devijacija vremena izvođenja bikubične interpolacije koristeći različite procesore za provođenje postupka*

	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Procesor	**Srednja vrijednost vremena izvođenja $\bar{x}$ [ms]**	**Standardna devijacija vremena izvođenja σ [ms]**	**Srednja vrijednost vremena izvođenja $\bar{x}$ [ms]**	**Standardna devijacija vremena izvođenja σ [ms]**
DSP1	176969,40	203,843	11411,80	118,71
DSP1-DSP2	89379,48	359,77	5992,22	246,54
A15	40753,10	316,43	2633,04	1,09

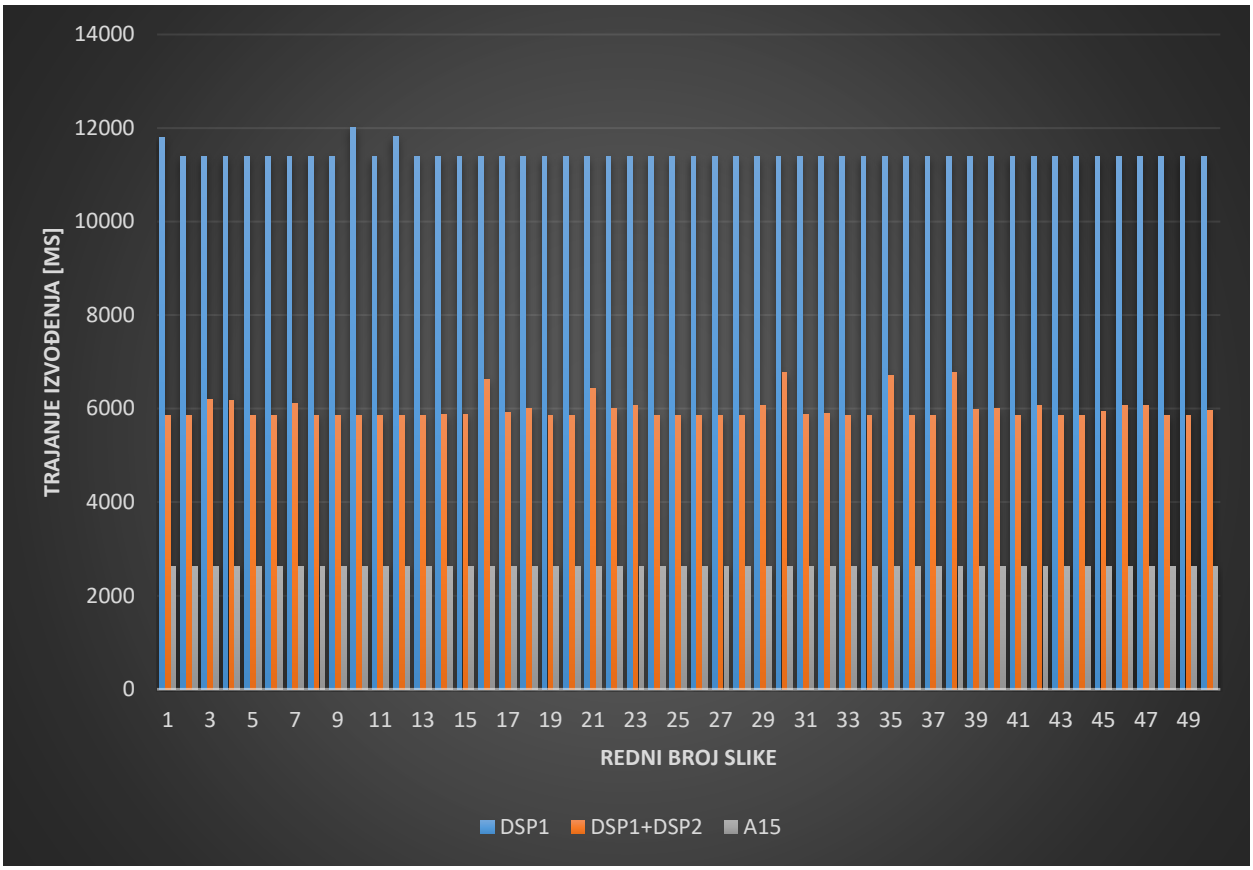
Tablica 4.15 *Broj slika (od njih ukupno 50) za koje se vrijeme izvođenja nalazi u intervalima od ± jedne, ± dviju i ± triju standardnih devijacija oko srednjeg vremena izvođenja prilikom korištenja procesora DSP1, istovremeno korištenje procesora DSP1 i DSP2 i korištenje procesora A15 za povećavanje i smanjivanje rezolucije slike četiri puta*

Procesor DSP1				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
$[\bar{x} \pm \sigma]$	44	88	47	94
$[\bar{x} \pm 2\sigma]$	47	94	47	94
$[\bar{x} \pm 3\sigma]$	48	96	47	94
Procesor DSP1+DSP2				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	

Interval	Broj	Postotak	Broj	Postotak
[x̄ ± σ]	42	84	45	90
[x̄ ± 2σ]	46	92	46	92
[x̄ ± 3σ]	50	100	48	96
Procesor A15				
	Povećanje rezolucije (640×360 →1280×720)		**Smanjenje rezolucije (640×360 → 320×180)**	
Interval	**Broj**	**Postotak**	**Broj**	**Postotak**
[x̄ ± σ]	35	70	43	86
[x̄ ± 2σ]	47	94	49	98
[x̄ ± 3σ]	50	100	50	100

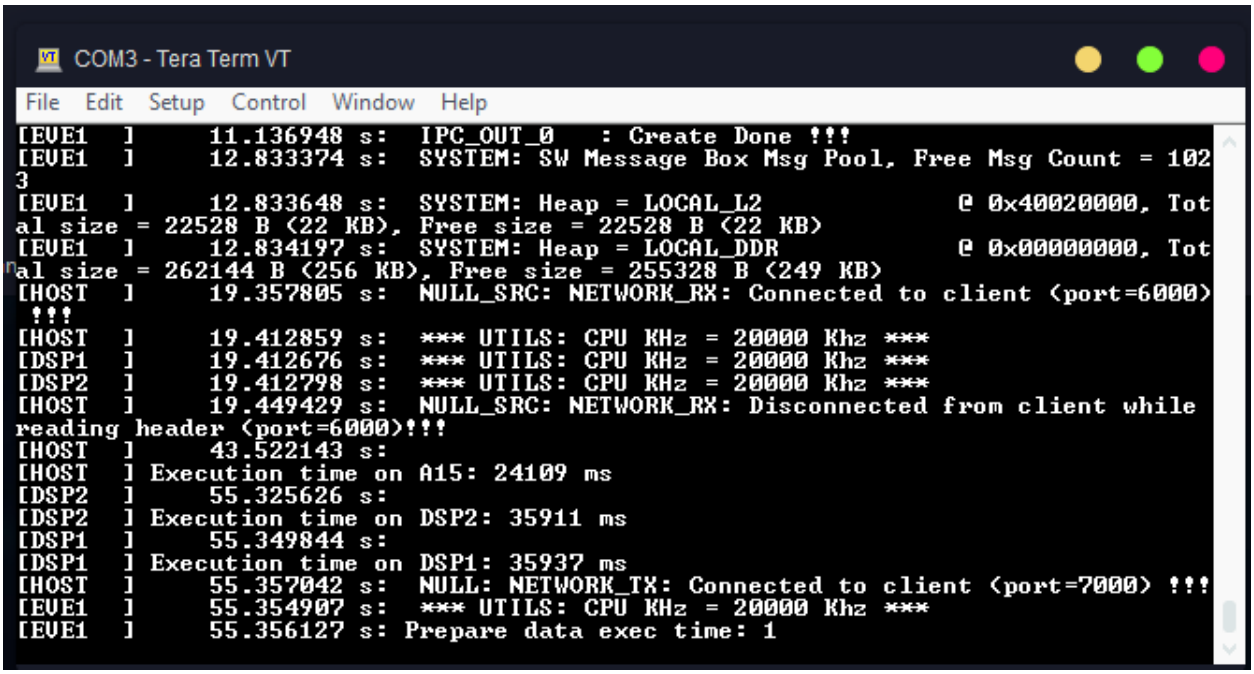


Slika 4.20. *Trajanje izvođenja bikubične interpolacije prilikom povećavanja rezolucije slike četiri puta, odnosno s 640×360 elemenata slike na 1280×720 elemenata slike za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i procesor A15*



Slika 4.21. *Trajanje izvođenja bikubične interpolacije prilikom smanjivanja rezolucije slike četiri puta, odnosno s 640×360 elemenata slike na 320×180 elemenata slike za procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 i procesor A15*

Mjerenja vremena izvođenja prilikom istovremenog korištenja procesora A15, DSP1 i DSP2 za bikubičnu interpolaciju nije bilo moguće. Razlog tomu je nedostatak memorije potrebne za alokaciju potrebnih varijabli za izvođenje istog algoritma tri puta u isto vrijeme. U programu *TerraTerm* neće se prikazivati greške nego rezultat kako je prikazan na slici 4.22., no rezultati se ne mogu uzeti u obzir budući da slika nije dobro interpolirana, odnosno sadrži slučajne brojeve. Primjer lošeg rezultata je prikazan na slici 4.23.



Slika 4.22. *Primjer ispisa vremena u programu TerraTerm za izvođenje bikubične interpolacije prilikom istovremenog korištenja procesora A15, DSP1 i DSP2 koje se ne može uzeti u obzir.*

Slika 4.23. *Primjer lošeg rezultata bikubične interpolacije prilikom istovremenog korištenja procesora A15, DSP1 i DSP2; Slika prikazuje slučajne vrijednosti iz memorije*

4.6. Mjerenje memorijskog otiska

Memorijski otisak je mjeren unutar programa *Visual Studio 2019*, pomoću alata *Diagnostic Tools* unutar samog programa. Utvrđeno je da interpolacija metodom najbližih susjeda i bilinearna interpolacija koriste ukupno na *heap*[6] memoriji 3.456.300 B memorije za tip *uint8_t* (ovaj tip je poseban tip podatka dostupan unutar biblioteke *stdint.h,* a nastao je od *unsigned char*), za tip *void* koriste 1.843.200 B, što je ukupno 5.229,23 KB memorije. Bikubična interpolacija na *heap* memoriji troši 22.118.400 B za tip *double,* 3.456.000 za tip *uint8_t i* 1.843.200 za tip *void*, što je ukupno 26.829,65 KB *heap* memorije. Kada se ukupna veličina *heap* memorije potrebna za izvođenje bikubične interpolacije pretvori iz KB u MB, ona iznosi 26,83 MB. Za istovremeno izvođenje bikubične interpolacije na procesorima A15, DSP1 i DSP2 je potrebno minimalno 80,49 MB *heap* memorije, što Alpha ploča nema na raspolaganju zbog rezerviranosti memorije, pa izvođenje bikubične interpolacije u ovom slučaju nije moguće. Čisti programski kôd interpolacije metodom najbližih susjeda spremljen u zasebnu datoteku zauzima 3.534 B prostora na disku, bilinearna interpolacija zauzima 5.125 B prostora na disku, a bikubična interpolacija zauzima 12.723 B prostora da disku, pa je pretpostavka da toliko zauzeto na *stack*[7] dijelu memorije prilikom izvršavanja programa.

4.7. Diskusija o rezultatima vezanim za vrijeme izvođenja pojedine metode za interpolaciju slike

Provedenom optimizacijom rješenja na računalu, a opisanom u potpoglavlju 3.7., dobiveno je manje vrijeme trajanja izvođenja pojedine metode interpolacije. Usporedba vremena trajanja izvođenja interpolacije pojedine metode prije optimizacije rješenja i poslije optimizacije rješenja prilikom smanjivanja, odnosno povećavanja rezolucije 640×360 elemenata slike četiri puta nalazi u tablici 4.16. Vrijednosti iz tablice 4.16. dobivene su pomoću podataka navedenih u potpoglavlju 4.3 i potpoglavlju 4.4. Iz tablice se vidi da je postignuto brže vrijeme trajanja izvođenja pojedine metode interpolacije na računalu.

[6] *Heap* memorija – memorija na kojoj se vrši dinamička alokacija memorije
[7] *Stack* memorija – privremena memorija za spremanje lokalnih varijabli

Tablica 4.16. *Usporedba vremena izvođenja pojedine metode interpolacije na računalu poslije provedene optimizacije i prije provedene optimizacije rješenja*

	Omjer vremena izvođenja poslije i prije optimizacije	
	Povećanje rezolucije (640×360 →1280×720)	**Smanjenje rezolucije (640×360 → 320×180)**
Metoda najbližih susjeda	0,13	0,10
Bilinearna interpolacija	0,96	0,58
Bikubična interpolacija	0,93	0,88

Prema [22], procesor DSP ima brzinu (radnu frekvenciju) od 600 MHz te veličinu predmemorije (engl. *cache*) od 96 KB. Procesor A15 ima brzinu od 750 MHz te nema fiksnu veličinu predmemorije te je zbog toga dosta brži od procesora DSP1 ili istovremenog korištenja procesora DSP1 i DSP2. U tablici 4.17. dan je odnos vremena izvođenja pojedinih procesora prilikom provođenja postupka interpolacije određenom metodom. Omjeri su određeni korištenjem prosječnog vremena izvođenja za pojedinu metodu interpolacije za slučajeve povećavanja i smanjivanja rezolucije.

Tablica 4.17. *Usporedba vremena izvođenja pojedine metode interpolacije na procesoru A15 u odnosu na procesor DSP1 i istovremeno korištenje procesora DSP1 i DSP2 te istovremenog korištenja procesora DSP1 i DSP2 u odnosu na DSP1 procesor*

	Povećanje rezolucije (640×360 →1280×720)		
	DSP1 / A15	**(DSP1+DSP2)/A15**	**DSP1/(DSP1+DSP2)**
Metoda najbližih susjeda	7,51	3,98	1,89
Bilinearna interpolacija	5,34	2,78	1,92
Bikubična interpolacija	4,34	2,19	1,98
	Smanjenje rezolucije (640×360 → 320×180)		
	DSP1 / A15	**(DSP1+DSP2)/A15**	**DSP1/(DSP1+DSP2)**
Metoda najbližih susjeda	3,05	2,21	1,38
Bilinearna interpolacija	3,18	2,19	1,45
Bikubična interpolacija	4,33	2,28	1,90

Generalno, iz tablice 4.17. može se zaključiti da je procesor A15 brži u odnosu na procesor DSP1 od 4 do 7 puta, ovisno o vrsti interpolacije, kada je riječ o povećavanju rezolucije slike. Prilikom smanjivanja slike, procesor A15 je od 3 do 4 puta brži, ovisno o tipu interpolacije, prilikom smanjivanja rezolucije slike. Navedenim odnosima brzine izvođenja po pojedinom procesoru se

određivao postotak slike koji je interpolirao pojedini procesor prilikom istovremenog korištenja procesora A15, DSP1 i DSP2.

U tablici 4.18 je dan odnos brzina izvođenja interpolacija na računalu u programskom jeziku C bez provedene optimizacije i na Alpha ploči po pojedinom procesoru.

Tablica 4.18. *Usporedba vremena izvođenja pojedine interpolacije na računalu u odnosu na procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 te u odnosu na procesor A15*

	Povećanje rezolucije (640×360 →1280×720)			
	(A15+DS1+DSP2) / Računalo	**A15 / Računalo**	**(DSP1+DSP2) /(Računalo**	**DSP1 / Računalo**
Metoda najbližih susjeda	1977,74	2495,67	9932,76	18751,42
Bilinearna interpolacija	1926,16	2362,76	6562,84	12593,39
Bikubična interpolacija		5937,07	13021,15	25781,58
	Smanjenje rezolucije (640×360 → 320×180)			
	A15+DS1+DSP2) / Računalo	**A15 / Računalo**	**(DSP1+DSP2) / Računalo**	**DSP1 / Računalo**
Metoda najbližih susjeda	5792,26	4307,54	9995,93	13784,11
Bilinearna interpolacija	4490,58	3593,54	7856,68	11423,96
Bikubična interpolacija		5636,03	12826,36	24426,99

U tablici je 4.19 je dan odnos srednjeg vremena izvođenja prilikom istovremenog korištenja procesora A15, DSP1 i DSP2 u odnosu na procesor A15, istovremeno korištenje procesora DSP1 i DSP2 i procesor DSP1 za interpolaciju metodom najbližih susjeda i bilinearnu interpolaciju.

Tablica 4.19. *Usporedba vremena izvođenja pojedine interpolacije prilikom istovremenog korištenja procesora A15, DSP1 i DSP2 u odnosu na procesor DSP1, istovremeno korištenje procesora DSP1 i DSP2 te u odnosu na procesor A15*

	Povećanje rezolucije (640×360 →1280×720)		
	(A15+DSP1+DSP2) / A15	**(A15+DSP1+DSP2) / (DSP1+DSP2)**	**(A15+DSP1+DSP2) / DSP1**
Metoda najbližih susjeda	0,92	0,23	0,12
Bilinearna interpolacija	0,83	0,3	0,16

	Smanjenje rezolucije (640×360 → 320×180)		
	(A15+DSP1+DSP2) / A15	(A15+DSP1+DSP2) / (DSP1+DSP2)	(A15+DSP1+DSP2) / DSP1
Metoda najbližih susjeda	1,54	0,66	0,48
Bilinearna interpolacija	1,42	0,65	0,45

Iz tablice 4.19. se može zaključiti da kombinacija triju procesora daje manje vrijeme izvođenja prilikom povećavanja rezolucije slike četiri puta u odnosu na najbrži procesor A15, dok prilikom smanjenja rezolucije četiri puta, kombinacija triju procesora ne daje kraće vrijeme izvođenja u odnosu na najbrži procesor A15.

Budući da se interpolacije rade za Alpha ploču kao realnu ADAS platformu, cilj je da vrijeme izvođenja interpolacije bude što kraće te da je kvaliteta slike što veća i da su rubovi na slici što je više moguće očuvaniji. Budući da je poznato da bilinearna interpolacije daje zadovoljavajuću kvalitetu slike uz dobro očuvane rubove, zanimljiva je za automotiv aplikacije zbog pogodnog vremena izvođenja. Uvidom u tablicu 4.19. može se konstatirati da bilinearna interpolacija ima najkraće vrijeme izvođenja prilikom istovremenog korištenja procesora A15, DSP1 i DSP2. Rubovi slike (odnosi se na automotiv slike iz kategorije 1, kategorije 2 i kategorije 3 skupa testnih slika) interpolirane bilinearnom interpolacijom su dovoljno dobro očuvani, što je bitno zbog detekcije objekata u dijelu slike, a prosječno vrijeme izvođenja interpolacije je zadovoljavajuće i iznosi približno 0,3 sekunde za povećavanje rezolucije slike četiri puta. Prilikom smanjivanja rezolucije slike četiri puta, a prema podacima iz tablice 4.19., u slučaju interpolacije metodom najbližih susjeda, omjer istovremenog korištenje procesora A15, DSP1 i DSP2 te samog procesora A15 je 1,54, što znači da je procesor A15 brži za 1,54 puta u odnosu na istovremeno korištenje A15, DSP1 i DSP2, dok za bilinearnu interpolaciju taj omjer iznosi 1,42, odnosno procesor A15 je brži za 1,42 puta u odnosu na istovremeno korištenje procesora A15, DSP i DSP2. Za navedeno smanjivanje rezolucije bi bio idealan procesor A15, koji daje najkraće vrijeme interpoliranja koje je manje od 0,10 sekundi. Ovi zaključci vrijede i za PC implementaciju budući da prema podacima iz tablice 4.18., a s obzirom na kvalitetu obrađene slike, bilinearna interpolacija daje najmanji odnos vremena izvođenja za sve metode.

Alpha ploča kao realna platforma ima i svoje nedostatke pa tako i samo rješenje pojedinih interpolacija. Glavni nedostatak rješenja svake interpolacije je optimizacija rješenja. Prilikom pokušaja optimiziranja rješenje putem skraćivanja dijelova programskog kôda u funkcije, utvrđeno je da Alpha ploča zbog ograničene količine memorije, ne može uspješno alocirati velike količine

memorije zapisane u obliku dvostrukih ili trostrukih pokazivača. Također, ne može alocirati dovoljno memorije za istovremeno korištenje procesora A15, DSP1 i DSP2. Možebitan razlog tome je niska razina upravljanja memorijom, što znači da bi razvojni inženjer trebao upravljati svakim dijelom dostupne memorije unutar okruženja, što je samo po sebi vrlo kompleksno. Korištena verzija VSDK ne može omogućiti bolju optimizaciju, a određena poboljšanja su dostupna u novijim verzijama programskog alata VSDK.

5. ZAKLJUČAK

Cilj automobilske industrije je u skoroj budućnosti ostvariti u potpunosti autonomnu vožnju. Da bi se ostvarila autonomna vožnja, funkcije u automobilu moraju se obavljati uz veliku točnost i preciznost. Često se prilikom autonomne vožnje mijenja rezolucija slike. Prilikom promjene rezolucije slike nužno je zadržati kvalitetu slike što je moguće boljom. Interpolacija se koristi kada se mijenja rezolucija slike, s ciljem očuvanja očuvanje najbitnijih vizualnih informacija. Kao realan sustav korištena je Alpha ploča razvijena od strane RT-RK Instituta. Korišteni programski paket je VisionSDK koji je izdala američka tvrtka Texas Instruments, napisan je u programskom jeziku C. Programski jezik C je pogodan za pisanje programskih rješenja za ugradbene sustave jer je neovisan i procesoru, prenosiv, daje bolje performanse u odnosu na ostale programske jezike, daje mogućnost manipulacije bitovima te kao najvažnije stavka jest moguće upravljanje memorijom.

Pregledom dostupnih postojećih znanstvenih radova, utvrđeno je da za sada nema implementacija algoritama za interpolaciju slike na realnu automotiv platformu, barem ne javno dostupnih. Za optimalnu promjenu rezolucije digitalne slike, a za realnu ADAS platformu, u ovom diplomskom radu koriste se neprilagodljive metode interpolacije. Interpolacija slike metodom najbližih susjeda zahtjeva kratko vrijeme izvođenja i implementacija algoritma je relativno laka. Slika stvorena metodom najbližih susjeda često sadrži efekt stvaranja bloka. Da bi se taj efekt smanjio, metoda najbližih susjeda se zamjenjuje bilinearnom interpolacijom koja koristi linearnu interpolaciju kao model za izračunavanje nepoznatih elemenata slike u dva smjera. Kompleksnija metoda od bilinearne interpolacije se zove bikubična interpolacija. Daje visoku kvalitetu interpolirane slike, ali su potrebni značajni računalni resursi i memorija. Na realnoj ADAS platformi utvrđeno je da korištenjem procesora s boljom frekvencijom rada i smanjivanjem ograničenja procesorske predmemorije, sve interpolacije daju bolje vremenske značajke. Testiranjem provedenim na skupu od 50 različitih ispitnih slika utvrđeno je da bilinearna interpolacija daje rezultate zadovoljavajuće kvalitete, uz konkurentno vrijeme izvođenja i potrebne resursa. Bikubična interpolacija, iako daje najbolju kvalitetu slike, traje predugo, što nije pogodno za realnu ADAS platformu, dok metoda najbližih susjeda je računalnu najmanje zahtjevna, ali daje najlošiju kvalitetu slike. Dodatnim uvidom u vlastita rješenja implementacija navedenih interpolacija, konstatirano je da optimizacija rješenja nije moguća u trenutnom verziji razvojnog okruženja VSDK, dok bi u nekim novijim verzijama vjerojatno bila moguća. Prema potpoglavlju 4.7. gdje su razrađeni rezultati, za realnu ADAS platformu, izvođenje bilinearne interpolacije daje zadovoljavajuću kvalitetu automotiv slike s dobro očuvanim rubovima te istovremenim

korištenjem procesora A15, DSP1 i DSP2 dobiva se prihvatljivo vrijeme izvođenja interpolacije od približno 0,3 sekundi za povećanje rezolucije slike četiri puta, odnosno sa 640×360 elemenata slike na 1280×720 elemenata slike. Prilikom smanjenja rezolucije četiri puta, odnosno sa 640×360 elemenata slike na 320×180 elemenata slike, bilinearna interpolacije daje najkraće vrijeme na procesoru A15.

LITERATURA

[1] European Commission, Advanced driver assistance systems, European Commission, Directorate General for Transport, February2018.

[2] Leksikografski zavod Miroslava Krleže, "Enciklopedija.hr," 2021.

[3] A. C. (Alan C. Bovik, *The essential guide to image processing*. Academic Press, 2009.

[4] Mr. P. S. Parsania and Dr. P. v. Virparia, "A Comparative Analysis of Image Interpolation Algorithms," *IJARCCE*, vol. 5, no. 1, pp. 29–34, Jan. 2016, doi: 10.17148/ijarcce.2016.5107.

[5] A. Prajapati, S. Naik, and S. Mehta, "Evaluation of Different Image Interpolation Algorithms," *International Journal of Computer Applications*, vol. 58, no. 12, Nov. 2012, doi: 10.5120/9332-3638.

[6] R. C. Gonzalez, R. E. Woods, S. L. Eddins, and G. Publishing, "Digital Image Processing Using Matlab, Second Edition," 2009. [Online]. Available: www.gatesmark.com

[7] Andra Ivanov, "Interpolacija Slike," 2018.

[8] pcmag.com, "YUV/RGB conversion formulas," 2021.

[9] imageprocessing.com, "Nearest Neighbour Interpolation".

[10] J. Juraj, "Primjena interpolacije u digitalnoj obradi slika Josipović, Iwan." [Online]. Available: https://urn.nsk.hr/urn:nbn:hr:126:999137

[11] A. K. Jha, A. Kumar, G. Schaefer, and Md. A. R. Ahad, "An efficient edge preserving image interpolation algorithm," May 2014. doi: 10.1109/ICIEV.2014.6850820.

[12] A. Savagave, A. P. Patil, E. Engg Dept, and J. J. Magdum, "Study of Image Interpolation," 2014. [Online]. Available: www.ijiset.com

[13] Kwan Pyo Hong, Joon Ki Paik, Hyo Ju Kim, and Chul Ho Lee, "An edge-preserving image interpolation system for a digital camcorder," *IEEE Transactions on Consumer Electronics*, vol. 42, no. 3, 1996, doi: 10.1109/30.536121.

[14] Xin Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing*, vol. 10, no. 10, 2001, doi: 10.1109/83.951537.

[15] C.-T. Lin, K.-W. Fan, H.-C. Pu, S.-M. Lu, and S.-F. Liang, “An HVS-Directed Neural-Network-Based Image Resolution Enhancement Scheme for Image Resizing,” *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, Aug. 2007, doi: 10.1109/TFUZZ.2006.889875.

[16] X. Zhang and X. Wu, “Image Interpolation by Adaptive 2-D Autoregressive Modeling and Soft-Decision Estimation,” *IEEE Transactions on Image Processing*, vol. 17, no. 6, Jun. 2008, doi: 10.1109/TIP.2008.924279.

[17] G. Jia and X. Peng, “Achieving the image interpolation algorithm on the FPGA platform based on ImpulseC,” Oct. 2013. doi: 10.1117/12.2031238.

[18] Matthew Giassa, “Nearest Neighbour Interpolation.”

[19] Matthew Giassa, “Bilinear Interpolation,” 2010.

[20] Matthew Giassa, “Generalized Bicubic Interpolation,” 2010.

[21] emertxe.com, “Why is C the most preferred language for embedded systems?”

[22] RT-RK Computer Based Systems, “Automotive Machine Vision ALPHA reference board on Texas Instruments SoCs,” 2021.

[23] technopedia.com, “System on a Chip (SoC),” 2017.

[24] D. Vajak, M. Vranješ, *Predlošci s laboratorijskih vježbi iz kolegija „Digitalna obrada slike i videa za autonomna vozila“, vježbe „Upoznavanje s ADAS razvojnom pločom“ i „Izrada vlastitog use-case-a za ADAS razvojnu ploču“*. 2018.

[25] University of Southern California, “The USC-SIPI Image Database,” vol. 2021.

[26] Tee0125, “yuvplayer.” 2017.

SAŽETAK

U ovom radu opisane su postojeće metode neprilagodljivih algoritama za interpolaciju slike. Objašnjen je problem interpolacije slike, opisani su matematički modeli triju metoda za interpolaciju slike (metode najbližih susjeda, bilinearne interpolacije i bikubične interpolacije) te je dan pregled te je dan pregled postojećih rješenja. Opisana je potom vlastita implementacija triju navedenih metoda interpolacije u C programskom jeziku na računalu. Nakon implementacije u C programskom jeziku na PC-u odabrane metode su implementirane na ADAS Alpha ploči, također koristeći C programski jezik. Nakon same implementacije na ploči, pažnja je posvećena raspodjeljivanju posla na dostupne procesora na Alpha ploči. Nakon optimalne raspodjele posla na pojedine procesore, uspoređivane su performase interpolacije trima spomenutim metodama prilikom izvođenja na ugradbenoj ADAS Alpha ploči. Rezultati su pokazali da na odabranom skupu od 50 testnih slika, od kojih je 30 automotiv sadržaja, 10 sadržaja motiva iz prirode s više i manje detalja i 10 umjetno generiranih slika, najbolji kompromis, ukoliko se u obzir uzmu vrijeme izvođenja i kvaliteta interpolirane slike, daje bilinearna interpolacija i to kada se zadatak interpolacije raspodijeli na tri različita procesora (DSP1, DSP2 i A15) prilikom povećavanje rezolucije slike četiri puta, odnosno s 640×360 elemenata slike na 1280×720 elemenata slike, gdje je prosječno vrijeme izvođenja 280,16 ms. Ispitivanjima je utvrđeno da za smanjivanje rezolucije slike četiri puta, odnosno s 640×360 elemenata slike na 320×180 elemenata slike, procesor A15 ima najkraće vrijeme izvođenja od 53,4 ms.

Ključne riječi: ADAS, VisionSDK, interpolacija metodom najbližih susjeda, bilinearna interpolacija, bikubična interpolacija

IMAGE INTERPOLATION WITH EDGES PRESERVING AND SOLOUTION IMPLEMENTATION ON A REAL ADAS PLATFORM

ABSTRACT

This paper describes the existing methods of non - adaptive interpolation image algorithms. The problem of interpolation images is explained, mathematical models of three methods for interpolation images (nearest neighbor methods, bilinear interpolation and bicubic interpolation) are described, and an overview is given and an overview of existing solutions is given. Then, the own implementation of the three mentioned interpolation methods in the C programming language on a computer is described. After implementation in the C programming language on a PC, the selected methods were implemented on the ADAS Alpha board, also using the C programming language. Following the same on-board implementations, attention was paid to allocating work to the available processors on the Alpha board. After optimal distributions, individual procedures were sent, and the interpolation performance of the three mentioned methods was compared when performed on a built-in ADAS Alpha board. The results showed that in the selected set of 50 test images, of which 30 automotive content, 10 nature motifs with more and less detail and 10 artificially generated images, the best compromise, if we take into account the execution time and quality are interpolated image, gives bilinear interpolation when the interpolation task is divided into three different processors (DSP1, DSP2 and A15) when increasing the image resolution four times, ie from 640 × 360 image elements to 1280 × 720 image elements, where the average execution time is 280 , 16 ms Tests have found that to reduce the image resolution four times, from 640 × 360 elements to 320 × 180 element images, the A15 processor has the shortest execution time of 53.4 ms.

Keywords: ADAS, VisionSDK, nearest neigbour interpolation, bilinear interpolation, bicubic interpolation

ŽIVOTOPIS

Božidar Kelava je rođen 3. siječnja 1998. godine u Slavonskom Brodu. Odrastao je i živi u Gunji, osnovnu školu Mate Lovraka je završio u gradu Županji. Nakon završetka osnovne škole upisuje Gimnaziju Županja, smjer Prirodoslovno matematički. U Osijeku, 2016. godine upisuje preddiplomski studdij elektrotehnike na FERIT-u, te na drugoj godini studija odabire smjer Komunikacije i informatika. 2019. godine stječe akademski naziv sveučilišni provstupnik inženjer elektrotehnike na temu završnog rada „Prilagođenje otvor antene primjenom dielektričnog plašta na otvoru – simulacija (HFSS). Iste godine upisuje diplomski sveučilišni studije elektrotehnike, smjer Komunikacije i informatika, izborni blok Mrežne tehnologije

PRILOZI

P. 3.1. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor DSP1, interpolacija metodom najbližih susjeda (elektronički prilog)

P. 3.2. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor A15, interpolacija metodom najbližih susjeda (elektronički prilog)

P. 3.3. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1 i DSP2, interpolacija metodom najbližih susjeda (elektronički prilog)

P. 3.4. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1, DSP2 i A15, interpolacija metodom najbližih susjeda (elektronički prilog)

P. 3.5. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor DSP1, interpolacija bilinearnom interpolacijom (elektronički prilog)

P. 3.6. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor A15, interpolacija bilinearnom interpolacijom (elektronički prilog)

P. 3.7. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1 i DSP2, interpolacija bilinearnom interpolacijom (elektronički prilog)

P. 3.8. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1, DSP2 i A15, interpolacija bilinearnom interpolacijom (elektronički prilog)

P. 3.9. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor DSP1, interpolacija bikubičnom interpolacijom (elektronički prilog)

P. 3.10. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za procesor A15, interpolacija bikubičnom interpolacijom (elektronički prilog)

P. 3.11. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1 i DSP2, interpolacija bikubičnom interpolacijom (elektronički prilog)

P. 3.12. Mapa koja sadrži slučaj korištenja za pokretanje rješenja uz pripadajući algoritam za istovremeno korištenje procesora DSP1, DSP2 i A15, interpolacija bikubičnom interpolacijom (elektronički prilog)

P. 4.1. Mapa koja sadrži skup ispitnih slika (elektronički prilog)

P. 4.2. Mapa koja sadrži program *yuvplayer* (elektronički prilog)

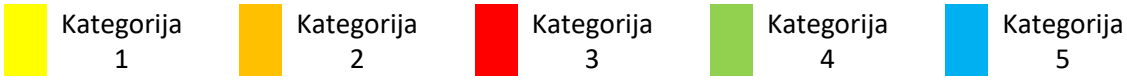
P. 4.3. Tablica vremena izvođenja interpolacije metodom najbližih susjeda na računalu u programskom jeziku C bez optimizacije rješenja

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

Slika	Povećanje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]	Smanjenje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]
1	0,0680	0,0100
2	0,0640	0,0090
3	0,0720	0,0090
4	0,0700	0,0090
5	0,0620	0,0100
6	0,0660	0,0100
7	0,0660	0,0100
8	0,0650	0,0090
9	0,0690	0,0090
10	0,0650	0,0090
11	0,0730	0,0090
12	0,0650	0,0100
13	0,0620	0,0090
14	0,0620	0,0090
15	0,0650	0,0090
16	0,0650	0,0090
17	0,0700	0,0090
18	0,0690	0,0090
19	0,0660	0,0100
20	0,0730	0,0090
21	0,0670	0,0100
22	0,0680	0,0090
23	0,0650	0,0100
24	0,0700	0,0110
25	0,0650	0,0110
26	0,0660	0,0100
27	0,0690	0,0100
28	0,0690	0,0110
29	0,0720	0,0100
30	0,0690	0,0100
31	0,0660	0,0110
32	0,0680	0,0090
33	0,0650	0,0100
34	0,0660	0,0100

35	0,0650	0,0120
36	0,0660	0,0090
37	0,0660	0,0100
38	0,0680	0,0110
39	0,0650	0,0100
40	0,0650	0,0110
41	0,0720	0,0100
42	0,0650	0,0090
43	0,0650	0,0110
44	0,0650	0,0090
45	0,0660	0,0110
46	0,0670	0,0100
47	0,0670	0,0090
48	0,0730	0,0090
49	0,0650	0,0100
50	0,0650	0,0120

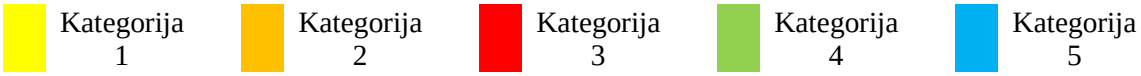
P. 4.4. Tablica vremena izvođenja bilinearne interpolacije na računalu u programskom jeziku C bez optimizacije rješenja



Slika	Povećanje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]	Smanjenje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]
1	0,1420	0,0150
2	0,1430	0,0160
3	0,1430	0,0140
4	0,1420	0,0150
5	0,1420	0,0150
6	0,1430	0,0160
7	0,1410	0,0140
8	0,1440	0,0160
9	0,1410	0,0160
10	0,1430	0,0150
11	0,1420	0,0150
12	0,1440	0,0150
13	0,1420	0,0140
14	0,1430	0,0150
15	0,1430	0,0150
16	0,1430	0,0150
17	0,1440	0,0140
18	0,1430	0,0140
19	0,1420	0,0140
20	0,1430	0,0160
21	0,1430	0,0140
22	0,1430	0,0140
23	0,1420	0,0160
24	0,1440	0,0140
25	0,1440	0,0160
26	0,1440	0,0160
27	0,1410	0,0150
28	0,1420	0,0140
29	0,1420	0,0160
30	0,1430	0,0140
31	0,1430	0,0140
32	0,1430	0,0140

33	0,1420	0,0140
34	0,1440	0,0140
35	0,1440	0,0150
36	0,1440	0,0140
37	0,1410	0,0160
38	0,1420	0,0160
39	0,1420	0,0150
40	0,1430	0,0150
41	0,1420	0,0140
42	0,1430	0,0140
43	0,1420	0,0140
44	0,1440	0,0150
45	0,1430	0,0150
46	0,1420	0,0160
47	0,1440	0,0160
48	0,1430	0,0150
49	0,1410	0,0140
50	0,1440	0,0150

P.4.5. Tablica vremena izvođenja bikubične interpolacije na računalu u programskom jeziku C bez optimizacije rješenja



Slika	Povećanje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]	Smanjenje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]
1	6,8620	0,4840
2	6,8090	0,4640
3	6,8200	0,4820
4	6,8010	0,4740
5	6,9060	0,4640
6	6,8180	0,4740
7	6,9010	0,4630
8	6,8660	0,4590
9	6,8830	0,4690
10	6,8920	0,4810
11	6,8480	0,4610
12	6,8480	0,4760
13	6,8240	0,4680
14	6,8940	0,4750
15	6,8700	0,4640
16	6,8530	0,4630
17	6,8650	0,4670
18	6,8610	0,4870
19	6,8530	0,4670
20	6,8550	0,4580
21	6,8110	0,4600
22	6,8600	0,4620
23	6,8150	0,4670
24	6,8960	0,4750
25	6,9000	0,4660
26	6,8670	0,4750
27	6,8260	0,4550
28	6,8490	0,4530
29	6,8930	0,4620
30	6,8970	0,4750
31	6,9180	0,4600
32	6,9010	0,4540

33	6,8430	0,4500
34	6,8760	0,4690
35	6,9500	0,4710
36	6,8360	0,4640
37	6,8930	0,4620
38	6,8930	0,4780
39	6,8630	0,4690
40	6,8140	0,4680
41	6,8500	0,4720
42	6,8850	0,458
43	6,8810	0,4610
44	6,8830	0,4800
45	6,8590	0,4510
46	6,8440	0,4600
47	6,8250	0,4680
48	6,8560	0,4610
49	6,8960	0,4800
50	6,9000	0,4730

P.4.6. Tablica vremena izvođenja interpolacije metodom najbližih susjeda na računalu u programskom jeziku C uz provedenu optimizaciju rješenja

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

Slika	Povećanje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]	Smanjenje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]
1	0,0080	0,0010
2	0,0080	0,0010
3	0,0080	0,0009
4	0,0080	0,0010
5	0,0090	0,0010
6	0,0090	0,0010
7	0,0090	0,0010
8	0,0080	0,0010
9	0,0090	0,0010
10	0,0080	0,0009
11	0,0100	0,0010
12	0,0090	0,0009
13	0,0090	0,0009
14	0,0090	0,0010
15	0,0090	0,0010
16	0,0080	0,0010
17	0,0090	0,0009
18	0,0090	0,0010
19	0,0090	0,0010
20	0,0090	0,0010
21	0,0110	0,0010
22	0,0090	0,0010
23	0,0100	0,0010
24	0,0090	0,0009
25	0,0090	0,0009
26	0,0090	0,0009
27	0,0090	0,0009
28	0,0090	0,0010
29	0,0100	0,0010
30	0,0090	0,0010
31	0,0090	0,0009
32	0,0090	0,0010
33	0,0090	0,0010
34	0,0090	0,0010

35	0,0100	0,0010
36	0,0090	0,0010
37	0,0090	0,0010
38	0,0090	0,0009
39	0,0090	0,0009
40	0,0080	0,0010
41	0,0090	0,0010
42	0,0090	0,0010
43	0,0090	0,0010
44	0,0080	0,0010
45	0,0110	0,0009
46	0,0110	0,0009
47	0,0090	0,0010
48	0,0080	0,0009
49	0,0090	0,0009
50	0,0090	0,0010

P.4.7. Tablica vremena izvođenja bilinearne interpolacije na računalu u programskom jeziku C uz provedenu optimizaciju rješenja

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

Slika	Povećanje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]	Smanjenje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]
1	0,1370	0,0080
2	0,1370	0,0090
3	0,1340	0,0080
4	0,1340	0,0090
5	0,1340	0,0080
6	0,1360	0,0080
7	0,1490	0,0090
8	0,1340	0,0080
9	0,1340	0,0090
10	0,1430	0,0090
11	0,1450	0,0080
12	0,1440	0,0090
13	0,1350	0,0090
14	0,1350	0,0080
15	0,1340	0,0080
16	0,1350	0,0080
17	0,1350	0,0100
18	0,1440	0,0080
19	0,1320	0,0080
20	0,1350	0,0090
21	0,1350	0,0090
22	0,1350	0,0090
23	0,1350	0,0090
24	0,1340	0,0090
25	0,1340	0,0090
26	0,1340	0,0090
27	0,1360	0,0080
28	0,1360	0,0080
29	0,1340	0,0100
30	0,1350	0,0100
31	0,1360	0,0090
32	0,1320	0,0090
33	0,1370	0,0080
34	0,1350	0,0090

35	0,1340	0,0090
36	0,1370	0,0090
37	0,1400	0,0090
38	0,1350	0,0080
39	0,1330	0,0090
40	0,1350	0,0080
41	0,1340	0,0080
42	0,1350	0,0090
43	0,1410	0,0080
44	0,1450	0,0100
45	0,1330	0,0090
46	0,1400	0,0090
47	0,1350	0,0080
48	0,1370	0,0080
49	0,1330	0,0090
50	0,1400	0,0090

P.4.8. Tablica vremena izvođenja bikubične interpolacije na računalu u programskom jeziku C uz provedenu optimizaciju rješenja

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

Slika	Povećanje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]	Smanjenje rezolucije (640×360 → 1280×720) trajanje izvođenja [ms]
1	6,3690	0,4090
2	6,3680	0,4110
3	6,4390	0,4070
4	6,3750	0,4120
5	6,3400	0,4090
6	6,3890	0,4110
7	6,3690	0,4070
8	6,4110	0,4060
9	6,4220	0,4120
10	6,3790	0,4050
11	6,4180	0,4080
12	6,4220	0,4120
13	6,3890	0,4100
14	6,3470	0,4080
15	6,3760	0,4080
16	6,4220	0,4050
17	6,3790	0,4070
18	6,4360	0,4040
19	6,3850	0,4060
20	6,3510	0,4100
21	6,3900	0,4000
22	6,3530	0,4130
23	6,4510	0,4140
24	6,4170	0,4110
25	6,3360	0,4110
26	6,3450	0,4140
27	6,3890	0,4110
28	6,4030	0,4100
29	6,3820	0,4150
30	6,3640	0,4140
31	6,4100	0,4080
32	6,4250	0,4110

33	6,3870	0,4030
34	6,3700	0,4090
35	6,3830	0,4140
36	6,4300	0,4100
37	6,3790	0,4120
38	6,3950	0,4100
39	6,3840	0,4130
40	6,3800	0,4100
41	6,4310	0,4090
42	6,3880	0,4100
43	6,3520	0,4130
44	6,3730	0,4060
45	6,3750	0,4070
46	6,4260	0,4080
47	6,3720	0,4070
48	6,3980	0,4110
49	6,4500	0,4100
50	6,4000	0,4150

P.4.9. Tablica vremena izvođenja interpolacije metodom najbližih susjeda koristeći različite procesore za provođenje postupka

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

	Povećanje rezolucije (640×360 →1280×720)			Smanjenje rezolucije (640×360 → 320×180)		
Slika	**Vrijeme izvođenja na DSP1 [ms]**	**Vrijeme izvođenja na DSP1+DSP2 [ms]**	**Vrijeme izvođenja na A15 [ms]**	**Vrijeme izvođenja na DSP1 [ms]**	**Vrijeme izvođenja na DSP1+DSP2 [ms]**	**Vrijeme izvođenja na A15 [ms]**
1	1255	664	167	135	98	42
2	1255	666	167	136	98	43
3	1255	664	167	136	99	42
4	1256	665	167	135	98	42
5	1255	666	167	136	98	42
6	1255	666	167	135	98	43
7	1256	664	167	136	98	43
8	1255	665	167	135	98	42
9	1255	665	167	135	98	42
10	1255	664	167	136	98	42
11	1255	664	167	136	98	43
12	1255	664	167	136	98	43
13	1255	664	167	135	98	42
14	1256	664	168	135	99	43
15	1255	666	167	135	99	43
16	1255	665	167	136	98	43
17	1256	665	167	135	98	42
18	1255	665	167	135	98	42
19	1255	664	168	135	98	42
20	1255	665	168	135	98	42
21	1255	666	167	136	99	43
22	1255	666	167	136	98	42
23	1255	666	167	135	98	42
24	1256	666	167	136	98	42
25	1255	665	167	135	98	42
26	1255	666	167	136	98	43
27	1255	665	167	135	98	42
28	1255	666	167	136	98	42
29	1255	665	167	135	98	42

30	1256	664	167	135	98	42
31	1256	666	167	136	99	43
32	1255	665	167	135	98	42
33	1255	664	167	136	99	42
34	1255	665	167	136	99	42
35	1255	665	167	135	98	42
36	1255	665	167	136	98	42
37	1255	665	167	136	98	42
38	1255	666	167	136	98	42
39	1256	664	167	135	98	42
40	1255	664	167	135	98	42
41	1255	664	167	135	98	42
42	1255	665	167	135	98	42
43	1256	665	167	136	98	42
44	1255	665	167	135	98	42
45	1255	665	167	135	98	42
46	1256	666	167	136	98	43
47	1256	664	167	136	98	42
48	1255	664	167	135	98	43
49	1255	664	167	136	99	43
50	1255	664	167	135	98	43

P.4.10. Tablica vremena izvođenja interpolacije metodom najbližih susjeda koristeći istovremeno procesore A15, DSP1 i DSP2, gdje *MAX(DSP1, DSP2, A15)* predstavlja srednje vrijeme izvođenja interpolacije prilikom korištenja sva tri procesora.

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

	Povećanje rezolucije 640×360 →1280×720				**Smanjenje rezolucije 640×360 → 320×180**			
SLIKA	**A15 [ms]**	**DSP1 [ms]**	**DSP2 [ms]**	**MAX(DSP1, DSP2, A15) [ms]**	**A15 [ms]**	**DSP1 [ms]**	**DSP2 [ms]**	**MAX(DSP1, DSP2, A15) [ms]**
1	154	121	121	154	41	64	64	64
2	155	122	122	155	41	66	66	66
3	154	121	121	154	41	65	65	65
4	155	122	121	155	41	66	66	66
5	155	121	121	155	41	65	65	65
6	155	122	122	155	41	66	66	66
7	155	122	122	155	41	65	65	65
8	155	122	122	155	40	65	65	65
9	154	121	121	154	40	65	65	65
10	155	121	121	155	41	65	65	65
11	155	121	121	155	40	65	65	65
12	154	121	121	154	40	65	65	65
13	155	121	121	155	40	65	65	65
14	154	121	121	154	40	65	65	65
15	154	121	121	154	41	65	64	65
16	155	122	121	155	41	65	65	65
17	155	121	121	155	40	65	64	65
18	154	121	121	154	41	64	64	64
19	154	121	121	154	41	65	64	65
20	155	122	122	155	41	64	65	65
21	154	121	121	154	41	65	64	65
22	155	122	1122	155	41	65	64	65
23	154	121	121	154	41	65	65	65
24	154	121	121	154	41	65	65	65
25	154	121	121	154	41	65	65	65
26	155	122	122	155	41	64	64	64
27	155	122	122	155	42	65	64	65
28	155	122	122	155	41	65	64	65
29	155	122	121	155	40	65	65	65

30	154	121	122	154	41	65	65	65
31	155	122	121	155	41	65	65	65
32	154	121	121	154	41	65	65	65
33	155	121	121	155	41	65	65	65
34	154	121	122	154	41	65	64	65
35	155	122	121	155	41	65	65	65
36	155	122	121	155	42	65	64	65
37	154	121	122	154	41	65	64	65
38	154	122	121	154	41	64	65	65
39	155	121	121	155	41	65	65	65
40	154	121	121	154	41	65	65	65
41	155	121	121	155	41	65	65	65
42	154	121	121	154	42	65	64	65
43	155	121	121	155	41	66	65	66
44	154	121	121	154	42	65	64	65
45	155	121	121	155	41	65	64	65
46	154	122	122	154	42	64	64	64
47	154	121	121	154	41	65	65	65
48	154	121	121	154	42	65	64	65
49	154	121	121	154	41	66	65	66
50	155	122	122	155	41	65	65	65

P.4.11. Tablica vremena izvođenja bilinearne interpolacije koristeći različite procesore za provođenje postupka

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

	Povećanje rezolucije (640×360 →1280×720)			Smanjenje rezolucije (640×360 → 320×180)		
Slika	**Vrijeme izvođenja na DSP1 [ms]**	**Vrijeme izvođenja na DSP1+DSP2 [ms]**	**Vrijeme izvođenja na A15 [ms]**	**Vrijeme izvođenja na DSP1 [ms]**	**Vrijeme izvođenja na DSP1+DSP2 [ms]**	**Vrijeme izvođenja na A15 [ms]**
1	1799	937	337	170	117	53
2	1799	937	337	169	117	53
3	1798	937	337	170	117	54
4	1798	936	337	170	117	54
5	1798	937	337	170	117	54
6	1798	937	337	170	116	54
7	1798	936	337	170	117	54
8	1798	936	338	170	117	54
9	1798	936	338	170	116	53
10	1798	936	338	169	117	54
11	1798	937	337	169	117	53
12	1798	937	337	170	117	53
13	1798	937	337	170	117	53
14	1798	937	337	170	117	53
15	1798	937	337	170	117	53
16	1799	936	337	170	116	53
17	1798	937	337	170	117	53
18	1798	937	337	169	117	53
19	1798	937	337	169	116	53
20	1798	937	337	170	117	53
21	1798	937	337	170	116	54
22	1798	937	338	170	117	54
23	1798	937	337	170	117	53
24	1798	937	337	170	117	53
25	1798	937	337	170	116	54
26	1798	936	337	169	117	53
27	1798	936	337	170	116	54
28	1799	937	337	170	117	53
29	1798	937	337	170	117	53

30	1798	937	337	169	116	54
31	1798	937	337	170	117	54
32	1798	937	338	170	117	54
33	1798	937	338	169	117	54
34	1798	937	337	170	117	54
35	1798	937	338	169	117	54
36	1799	937	338	170	116	54
37	1798	937	337	170	117	54
38	1798	936	337	170	116	53
39	1798	936	338	170	117	53
40	1798	937	337	170	117	53
41	1797	937	338	170	117	53
42	1798	937	338	170	117	53
43	1798	937	337	170	117	53
44	1798	936	337	169	117	53
45	1787	937	338	169	117	53
46	1789	937	337	169	116	53
47	1794	937	337	170	117	53
48	1796	937	337	170	117	54
49	1798	937	338	170	117	53
50	1799	937	337	170	117	53

P.4.12. Tablica vremena izvođenja bilinearne interpolacije koristeći istovremeno procesore A15, DSP1 i DSP2, gdje *MAX(DSP1, DSP2, A15)* predstavlja srednje vrijeme izvođenja interpolacije prilikom korištenja sva tri procesora.

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

	Povećanje rezolucije 640×360 →1280×720				**Smanjenje rezolucije 640×360 → 320×180**			
SLIKA	**A15 [ms]**	**DSP1 [ms]**	**DSP2 [ms]**	**MAX(DSP1, DSP2, A15) [ms]**	**A15 [ms]**	**DSP1 [ms]**	**DSP2 [ms]**	**MAX(DSP1, DSP2, A15) [ms]**
1	265	281	280	281,00	49	75	75	75,00
2	265	281	280	281,00	49	75	75	75,00
3	265	280	280	280,00	49	76	75	76,00
4	265	280	280	280,00	49	76	75	76,00
5	265	280	279	280,00	49	76	76	76,00
6	264	280	280	280,00	49	76	76	76,00
7	265	280	280	280,00	49	75	75	75,00
8	265	280	280	280,00	49	76	76	76,00
9	265	280	280	280,00	50	76	75	76,00
10	265	280	280	280,00	49	76	75	76,00
11	265	280	280	280,00	49	75	75	75,00
12	265	281	280	281,00	49	75	75	75,00
13	265	280	280	280,00	47	75	75	75,00
14	265	280	279	280,00	49	77	75	77,00
15	265	280	280	280,00	49	75	75	75,00
16	265	281	280	281,00	49	76	75	76,00
17	265	280	280	280,00	49	76	75	76,00
18	265	281	281	281,00	50	76	75	76,00
19	265	280	280	280,00	49	76	75	76,00
20	264	280	280	280,00	49	75	75	75,00
21	265	280	280	280,00	49	76	76	76,00
22	265	280	280	280,00	50	76	75	76,00
23	265	281	281	281,00	50	75	75	75,00
24	262	281	280	281,00	49	76	75	76,00
25	268	282	282	282,00	49	76	75	76,00
26	265	280	280	280,00	49	76	75	76,00
27	265	280	280	280,00	49	76	76	76,00
28	264	280	280	280,00	49	75	75	75,00
29	264	280	280	280,00	49	76	76	76,00

30	264	280	280	280,00	49	76	75	76,00
31	264	280	280	280,00	49	76	76	76,00
32	265	280	280	280,00	49	76	76	76,00
33	264	280	280	280,00	49	76	75	76,00
34	265	280	280	280,00	49	76	76	76,00
35	265	280	280	280,00	50	75	76	76,00
36	265	280	280	280,00	50	75	75	75,00
37	265	280	280	280,00	49	76	76	76,00
38	264	280	280	280,00	49	76	76	76,00
39	265	280	280	280,00	49	76	75	76,00
40	265	280	280	280,00	49	75	77	77,00
41	265	281	280	281,00	49	76	76	76,00
42	265	279	280	280,00	50	75	75	75,00
43	265	280	280	280,00	49	75	75	75,00
44	264	280	279	280,00	49	75	76	76,00
45	265	279	278	279,00	49	75	76	76,00
46	265	278	278	278,00	49	76	76	76,00
47	264	280	280	280,00	49	76	76	76,00
48	264	280	280	280,00	49	76	76	76,00
49	264	281	280	281,00	49	76	75	76,00
50	264	280	280	280,00	49	76	76	76,00

P.4.13. Tablica vremena izvođenja bikubične interpolacije koristeći različite procesore za provođenje postupka

Kategorija 1 | Kategorija 2 | Kategorija 3 | Kategorija 4 | Kategorija 5

	Povećanje rezolucije (640×360 →1280×720)			Smanjenje rezolucije (640×360 → 320×180)		
Slika	**Vrijeme izvođenja na DSP1 [ms]**	**Vrijeme izvođenja na DSP1+DSP2 [ms]**	**Vrijeme izvođenja na A15 [ms]**	**Vrijeme izvođenja na DSP1 [ms]**	**Vrijeme izvođenja na DSP1+DSP2 [ms]**	**Vrijeme izvođenja na A15 [ms]**
1	176893	89097	40291	11806	5855	2634
2	177616	89137	40576	11384	5849	2633
3	176895	89724	41366	11382	6191	2634
4	176894	89102	40296	11382	6184	2633
5	176895	89134	40596	11382	5855	2634
6	176890	89364	41194	11383	5849	2633
7	176890	89379	41241	11382	6104	2634
8	176894	89094	40286	11382	5855	2634
9	176894	89104	41145	11382	5849	2634
10	176894	89177	40629	12004	5848	2633
11	176893	89097	41286	11382	5845	2633
12	176893	89202	40616	11809	5849	2633
13	176892	90323	40627	11382	5855	2633
14	176892	89327	40397	11383	5875	2633
15	176896	89372	40615	11383	5882	2634
16	176896	89134	40573	11383	6632	2634
17	176895	89137	40621	11382	5924	2633
18	176896	89135	41064	11382	5996	2633
19	176895	89094	41210	11382	5849	2633
20	177415	90166	40610	11383	5855	2633
21	176893	89704	40618	11383	6428	2634
22	176893	89982	40613	11382	5997	2631
23	176893	89310	40776	11383	6058	2632
24	176893	89727	41582	11382	5849	2634
25	176893	89353	40605	11382	5855	2633
26	176893	89095	40568	11383	5849	2634
27	176893	90173	41441	11382	5849	2634
28	177003	89916	40624	11383	5856	2634
29	176893	89228	40904	11383	6058	2631

30	176893	89137	40616	11383	6774	2634
31	176895	90123	41183	11382	5885	2634
32	176895	89094	40608	11382	5895	2634
33	176895	89136	41431	11382	5850	2633
34	176896	89707	40618	11382	5856	2632
35	176891	89094	40623	11382	6712	2631
36	176891	89794	40570	11382	5849	2633
37	177307	89130	40617	11382	5856	2630
38	176908	89172	40560	11382	6782	2631
39	176892	90094	41135	11383	5992	2633
40	176893	89095	40614	11383	5998	2634
41	176892	89136	40622	11383	5855	2631
42	176892	89094	40616	11382	6066	2632
43	177306	89104	40620	11382	5849	2632
44	176890	89204	40730	11382	5855	2634
45	177296	89092	40560	11382	5928	2634
46	177902	89132	40568	11382	6073	2634
47	177091	89632	40624	11382	6076	2632
48	176896	89671	40573	11383	5849	2631
49	176891	89215	40573	11382	5855	2634
50	176893	89331	40624	11382	5956	2634