Proračun vektora pokreta na razini blokova i njihovo grupiranje prema sličnosti uz implementaciju na realnu ADAS razvojnu platformu

Radičević, Valentin

Master's thesis / Diplomski rad

2021

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:098658

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

Sveučilišni studij

PRORAČUN VEKTORA POKRETA NA RAZINI MAKROBLOKOVA I NJIHOVO GRUPIRANJE PREMA SLIČNOSTI UZ IMPLEMENTACIJU NA REALNU ADAS RAZVOJNU PLATFORMU

Diplomski rad

Valentin Radičević

Osijek, 2021.

SADRŽAJ

1. UVOD

Razvoj tehnologije u autoindustriji oduvijek je vođen idejom povećanja performansi automobila. Automobili postaju sve brži, troše sve manje energije (gorive mase), sve su sigurniji za vožnju, imaju bolje sigurnosne sustave u slučaju sudara, bolje sustave kočenja, itd. U sve navedene komponente autoindustrije s vremenom sve više ulazi i računalna tehnologija. Računalo izračunava kako rasporediti gorivu masu po cilindrima te kada započinje aktivacija kojeg cilindra ne bi li automobil postigao veću brzinu, kako rasporediti silu kočenja po kotačima ne bi li kočioni put bio kraći, kojim rasporedom aktivirati zračne jastuke u slučaju sudara i sl. Tako razvijani sustavi postali su plodno tlo za trenutni smjer razvoja autoindustrije, a to je računalno upravljana vožnja s krajnjim ciljem potpuno autonomne vožnje. Sve to može se staviti pod jedno ime – ADAS (engl. *Advanced Driver Assistance System*). Ideja ADAS je da zamijeni što više ljudskih radnji u tijeku vožnje. Ovisno o razini autonomije vožnje, može se raditi o zamijeni svih radnji, no ono što je danas aktualno i realno jesu vozila s implementiranim razinama autonomije od prve do četvrte razine. Sustavi koji su trenutno najdostupniji su oni prve, druge i treće razine koji traže da je vozač i dalje sudionik u vožnji, ali omogućavaju centriranje u voznim trakama, upozoravanje na objekte u okolini poput znakova, automobila i pješaka te drugih sudionika u prometu, automatsko kočenje u slučaju nužde (ukoliko automobil nailazi na prepreku na koju čovjek nije reagirao) [1] i sl.

Za realizaciju prethodno navedenih značajki ADAS koriste se između ostalog kamere, LIDAR i radar s čijih se senzora prikupljaju podaci te se obrađuju koristeći procesore namjenskih ugradbenih računalnih sustava u automobilu. Tako npr. računalo iz video okvira izdvaja značajke na osnovu kojih potom nastoji odrediti što je objekt, a što ne, gdje se nalaze linije koje ograničavaju cestu i sl.

Za detekciju objekata u video okviru koriste se napredne metode kao što su različite metode računalnog vida i strojnog učenja (npr. konvolucijske mreže). No ponekad je potrebno prepoznavati samo postoji li nešto što nije okolina koja miruje i postoji li nešto što se u okolini kreće te kojom brzinom i gdje se kreće. Takav podatak također je unaprjeđenje u vožnji jer omogućava da nešto što vozač nije uopće uočio ili što nije uočio na vrijeme, računalo uoči u puno kraćem vremenu i iz takvog podatka pokuša predvidjeti brzinu i smjer kretanja objekta u nekoliko budućih trenutaka te o tome obavijesti vozača. Ovakve metode imaju visok stupanj točnosti, ali zahtijevaju određeno vrijeme, veliku grupu signala za treniranje u svrhu detekcije i klasifikacije objekata te vrlo velik memorijski otisak kako bi se ostvarila ta točnost.

Uz navedene metode postoje jednostavnije metode kojima se može pronaći postoji li kretanje u video okviru. Takve metode proizlaze iz tehnika za kompresiju video signala i zasnivaju se na izračunima vektora pokreta (u nastavku rada VP) između dvaju susjednih okvira u slijedu video okvira.

Zadatak ovoga rada je implementacija algoritma za proračun VP između dvaju susjednih video okvira te njihovo grupiranje prema sličnosti, s ciljem određivanja smjera i brzine kretanja objekata u sceni. Metoda se zasniva na usporedbi makroblokova fiksne veličine koji čine video okvir s makroblokovima sljedećeg ili prethodnog video okvira. Takve metode računalno su zahtjevne, ali ih je zbog prirode izračuna moguće lako raspodijeliti na izračune po dijelovima okvira te ih tako rasporediti na nekolicinu procesora, odnosno paralelizirati te mnogostruko povećati brzinu izračuna, što je također zadatak ovog rada. Nakon izračuna VP, vektore je potrebno grupirati prema karakteristikama snage (duljine), kutu, odnosno smjeru i položaju unutar okvira, kako bi se odredilo što je objekt, u kojem dijelu scene se nalazi te u kojem se smjeru kreće. U ovome radu implementirat će se 3 metode za proračun VP, gdje će se svaka najprije implementirati na osobnom računalu, a onda na realnoj ADAS platformi uz korištenje dostupnih procesora. Potrebno je napraviti analizu performansi implementiranog rješenja iz nekoliko pogleda: točnosti određivanja smjera kretanja objekata, brzine izvođenja rješenja. Pomoću rezultata analize performansi procijenit će se mogućnost korištenja predloženih metoda na ADAS platformi s ograničenim resursima.

U sljedećem poglavlju navest će se za što se vektori pokreta inače koriste te će se obraditi postojeće metode za pronalazak istih i navesti odabrane metode koje su implementirane u ovome radu. U trećem poglavlju opisat će se predložena rješenja za proračun vektora pokreta i njihovo grupiranje, opisati ADAS platforma koja se koristi, programska podrška za tu platformu te proces izrade algoritma i implementacije na platformu. U četvrtom poglavlju predstavit će se i protumačiti rezultati testiranja odabranih metoda odnosno implementiranih rješenja na ADAS platformi. Zadnje, peto poglavlje donosi zaključke rada.

2. METODE ZA PRORAČUN VEKTORA POKRETA U VIDEO SIGNALU

Povećanjem korištenja video prijenosa u svijetu i povećanjem rezolucije video signali su postali sve veći, a širina pojasa prijenosa takvoga signala ostala je ograničena stoga je trend kompresije video signala porastao. Kompresija video signala dijeli se na unutar-okvirnu te među-okvirnu. U unutar-okvirnoj kompresiji iskorištava se prostorna sličnost elemenata slike kako bi se smanjila veličina jednog okvira. U među-okvirnoj koristi se vremenska zalihost, odnosno velika sličnost između dvaju susjednih okvira u video signalu, ne bi li se smanjila veličina video signala. U navedenoj među-okvirnoj kompresiji stvaraju se 2 tipa okvira - P i B, dok se I okviri stvaraju u unutar-okvirnoj kompresiji. I okvir se isključivo komprimira unutar-okvirno, dok P i B okviri koriste vremensku zalihost i informacije iz I okvira. Među-okvirna kompresija zasniva se na procjeni pokreta (engl. *motion estimation* - ME). Okvir koji se komprimira dijeli se na makroblokove određene veličine (fiksne ili sadržaju prilagodljive), prikazano na slici 2.1.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Slika 2.1. *Okvir koji se komprimira podijeljen na makroblokove fiksne veličine*

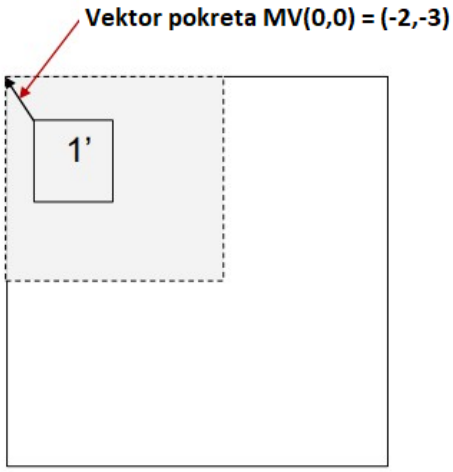
Nakon podjele svaki se makroblok uspoređuje s makroblokovima referentne slike i pronalazi se njemu najsličniji izračunavanjem određene mjere sličnosti. Mjera sličnosti dvaju makroblokova je brojčani podatak za dva makrobloka matematički najčešće izračunat prema formuli (2-1), koja predstavlja sumu apsolutnih razlika (engl. *Sum of absolute error* - SAE) elemenata slike makroblokova ili prema formuli (2-2), koja predstavlja srednju vrijednost sume kvadrata razlike (engl. *Mean squared error* – MSE) elemenata slike makroblokova:

$$SAE_{b_x,b_y}(d_x, d_y) = \sum_{j=0}^{15}\sum_{i=0}^{15} |F_t(16b_x + i, 16b_y + j) - F_{t-1}(16b_x + d_x + i, 16b_y + d_y + j)| \quad (2\text{-}1)$$

$$MSE_{b_x,b_y}(d_x, d_y) = \frac{1}{16 * 16}\sum_{j=0}^{15}\sum_{i=0}^{15} (F_t(16b_x + i, 16b_y + j) - F_{t-1}(16b_x + d_x + i, 16b_y + d_y + j))^2 \quad (2\text{-}2)$$

Pritom b_x i b_y označavaju koordinatu gornjeg lijevog ruba makrobloka u slici koja se komprimira za koji se traži najsličniji makroblok u referentnoj slici, a d_x i d_y pomak makrobloka u referentnoj slici po x osi (d_x) i y osi (d_y) u odnosu na lokaciju makrobloka na slici koja se komprimira. F_t označava vrijednost elementa slike u trenutnom okviru, a F_{t-1} vrijednost elementa slike u referentnom (prethodnom) okviru. Formule vrijede za makroblokove 16x16 elemenata slike, no iste se mogu proširiti na bilo koju veličinu makrobloka te sukladno tome promijeniti.

Dva makrobloka, za koja je izračunati SAE ili MSE (ovisno koji se kriterij uzima u obzir) minimalan, smatraju se najsličnijim makroblokovima i zapisuje se VP koji ukazuje na razliku njihova položaja u dvama susjednim okvirima kojima oni pripadaju. VP se zapisuje u obliku MV(b_x,b_y) = (u,v) gdje u označava pomak makrobloka po x osi, a v označava pomak makrobloka po y osi, što je prikazano na slici 2.2.



Slika 2.2. *Referentna slika s označenim najsličnijim makroblokom 1 iz slike koja se komprimira i njegovim vektorom pokreta MV*

Slika prikazuje vektor pokreta MV(0,0) = (-2,-3) koji govori da se makroblok 1 na koordinati

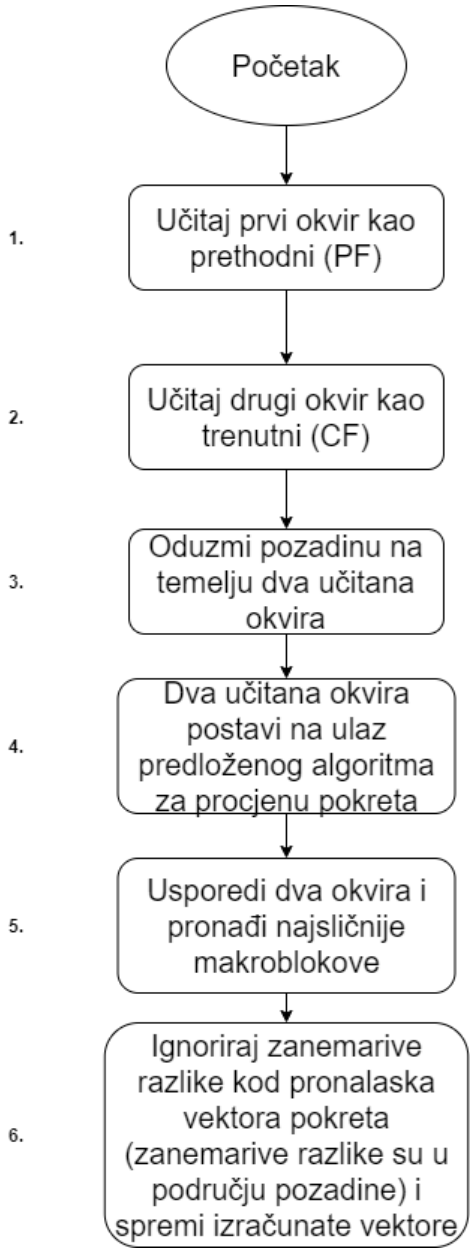
(0,0) (gornja lijeva koordinata makrobloka) iz okvira koji se kompirmira (prikazano na slici 2.1.) na referentnom okviru nalazi na mjestu udaljenom za 2 elementa slike desno i za 3 elementa slike dolje, makroblok 1'.

Nakon izračuna VP za sve makroblokove stvara se novi okvir načinjen od makroblokova referentne slike, pomaknutih za pripadne VP, čime nastaje slika prediktor ili trenutni predviđeni okvir. Prediktor se potom oduzima od okvira koji se trenutno komprimira i nastaje okvir razlike, koji se zajedno s VP u koderu entropijski kodira i šalje dekoderu kako bi on mogao što vjernije rekonstruirati trenutni okvir [2].

Metoda pronalaska VP je izrazito računalno zahtjevna. Stoga se u slučaju video signala jednom izračunati VP dalje koriste samo za dekompresiju, što je puno manje računalno zahtjevno. U ovome radu VP se želi iskoristiti kako bi se uvidjelo koji dijelovi scene se kreću te tako saznati postoji li objekt koji se kreće. Ograničenje je to što se u ovome slučaju pronalazak VP treba izvoditi u stvarnom vremenu. Stoga će se u nastavku ovog poglavlja razmotriti 5 znanstvenih radova u kojima se predlažu metode za pronalazak vektora pokreta te se daje osvrt na točnost procjene i brzinu izvođenja. Osim metoda za pronalazak VP razmotrit će se jedna metoda za grupiranje VP.

2.1. Unaprijeđena metoda pune pretrage

Puna pretraga (engl. *Full search*) predstavlja metodu u kojoj se makroblok trenutnog okvira uspoređuje sa svakim makroblokom referentnog okvira. Takva pretraga je računalno zahtjevna i povećava vrijeme izvođenja. Autori rada [3] predlažu poboljšanje izračuna VP s osvrtom na problematiku velike energetske potrošnje procesnih jedinica za procjenu pokreta u postojećim računalima. Prva pretpostavka rada je da nije potrebno tražiti najsličniji makroblok na cijelome okviru već samo na dijelu okvira koji je određen najvećim mogućim pomakom *P*. Druga pretpostavka rada je da se kao kriterij sličnosti koristi SAE. SAE je suma apsolutnih razlika odgovarajućih elemenata slike opisana formulom (2-1). Ovaj kriterij je izrazito manje računalno zahtjevan u usporedbi s drugim kriterijima kao što je MSE, jer uključuje operaciju kvadriranja razlike - unošenje dodatne operacije množenja. Kao osnovni prijedlog poboljšanja pune pretrage predstavlja se metoda oduzimanja pozadine [3]. Navedena metoda izdvaja objekte prednjeg plana od pozadine. U radu [3] je za oduzimanje pozadine korištena metoda aproksimacije medijana. Dijagram toka metode predložene u radu [3] prikazan je slikom 2.3.



Slika 2.3. *Dijagram toka metode za pronalazak VP korištene u predloženom radu [3]*.

U koraku izračuna VP, označenim s 5. na slici 2.3., dolazi do mnogostrukog smanjenja brzine jer zbog oduzete pozadine smanjuje se područje pretrage za količinu oduzetog područja. Za područje koje je određeno kao pozadina ne traže se VP već se odmah postavljaju u vrijednost (0,0). Korištenjem metode oduzimanja pozadine ne dolazi do smanjenja kvalitete izračunatih vektora (kvaliteta vektora iskazana je u najmanjoj pronađenoj vrijednosti mjere SAE tijekom pretrage za dva određena makrobloka), što proizlazi iz činjenice da se za cijelo područje okvira koje nije pozadina i dalje koristi metoda pune pretrage koja uvijek daje najmanje moguće vrijednosti mjere SAE.

Predložena metoda testirana je na osobnom računalu nepoznatih karakteristika i uspoređivana je s osnovnom metodom pune pretrage koja traži najsličniji makroblok na cijelom području okvira. Veličina makrobloka i kriterij najvećeg pomaka *P* nisu dani u radu [3]. Kao testne sekvence korištena su dva video signala, jedan rezolucije 176x144 elemenata slike s 300 okvira i drugi rezolucije 320x240 elememata slike od 480 okvira. Rezultati metode predstavljeni su kroz usporedbu s osnovnom metodom pune pretrage kroz kriterije vremena izvođenja i srednje vrijednosti izračunate SAE mjere za najsličnije makroblokove. Rezultati su prikazani tablicom 2.1. za prvi video signal i tablicom 2.2. za drugi video signal, u kojima su dane vrijednosti vremena za cijeli video signal.

Tablica 2.1. *Rezultati predložene unaprijeđene metode pune pretrage [3] i metode pune pretrage na cijelom okviru za video signal rezolucije 176x144 elemenata slike s 300 okvira (FS - puna pretraga, IFS – predložena unaprijeđena metoda pune pretrage, SAE – srednja vrijednosti izračunatih mjera razlike svih pronađenih najsličnijih makroblokova), vrijeme izvođenja odnosi se na cijeli video signal*.

Metoda pretrage vektora pokreta	Vrijeme izvođenja [s]	SAE
IFS	20.9532	31.4767
FS	215.4866	9.0001

Tablica 2.2. *Rezultati predložene metode pune pretrage i metode pune pretrage na cijelom okviru za video signal rezolucije 320x240 elemenata slike s 480 okvira (FS - puna pretraga, IFS – predložena unaprijeđena metoda pune pretrage, SAE – srednja vrijednosti izračunatih mjera razlike svih pronađenih najsličnijih makroblokova)), vrijeme izvođenja odnosi se na cijeli video signal*.

Metoda pretrage vektora pokreta	Vrijeme izvođenja [s]	SAE
IFS	32.3894	9.5961
FS	969.6491	1.6849

Iako dolazi do smanjenja brzine izvođenja do 30 puta, ovakva metoda primjenjiva je u ograničenom broju slučajeva. Slučajevi na koje je metoda primjenjiva odnose se na video signale u kojima je dinamičnost pozadine minimalna i u kojim uopće postoji pozadina, jer u slučajevima dinamične pozadine ne može doći do oduzimanja iste te se metoda ponovno svodi na običnu metodu brze pretrage, stoga je metoda većinom primjenjiva za fiksne kamere. Ovakve metode dovode do nestabilnog vremena izvođenja, jer vrijeme izvođenja ovisi o sadržaju okvira. Ukoliko većina okvira sadrži objekt ili objekte koji se kreću, manja je količina pozadine, proporcionalno je veće područje pretrage i vrijeme izvođenja se povećava.

2.2. Metoda tri koraka s predikcijom

U radu [4] koristi se metoda tri koraka za pronalazak VP. Metoda tri koraka (engl. *Three step search* - TSS) zasniva se na pretrazi makroblokova na referentnom okviru u bliskoj okolini (određena korakom pretrage *k*) makrobloka trenutnog okvira. Kao prvi makroblok u referentnom okviru za kojeg se provjerava sličnost s makroblokom iz trenutnog okvira odabire se makroblok koji se nalazi na istoj lokaciji kao onaj u okviru koji se komprimira. Oko početnog odabranog makrobloka određuje se 8 okolnih makroblokova udaljenih za korak pretrage *k*. Kada se izračuna mjera razlike za svih 9 makroblokova, odabire se onaj koji ima najmanju mjeru razlike i on se sada postavlja kao središnji. Korak pretrage *k* postaje *k*/2 i odabire se novih 8 makroblokova na udaljenosti novog koraka pretrage za koje se računa mjera razlike. Makroblok s najmanjom mjerom ponovno se postavlja kao središnji makroblok. Korak pretrage postavlja se na $k = 1$ te se odabire novih 8 makroblokova na udaljenosti *k*. Nakon izračuna mjere razlike za 8 novoodabranih makroblokova kao najsličniji makroblok odabire se onaj s najmanjopm mjerom razlike koji se koristi za izračun VP. Autori rada [4] predlažu poboljšanje ove metode korištenjem predikcije.

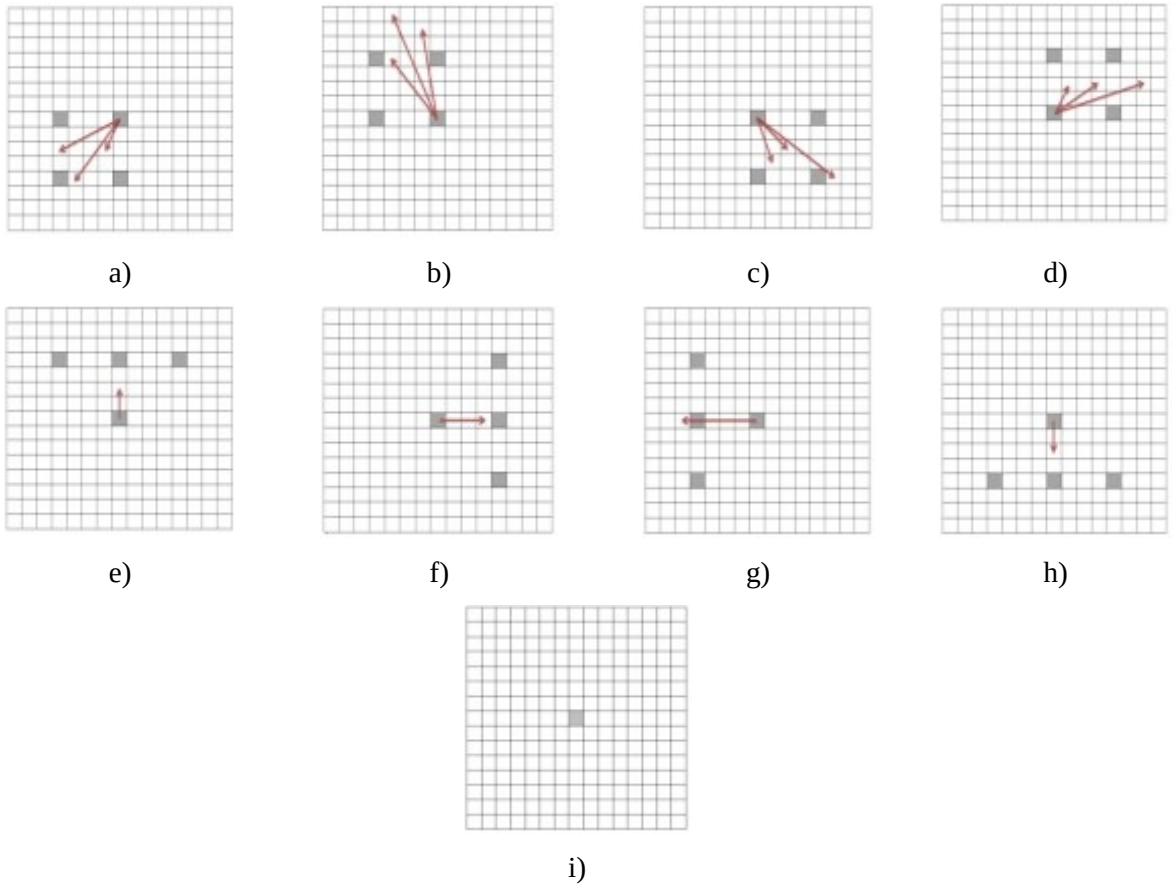
Osnovna pretpostavka rada [4] je da su VP u bliskoj okolini istog ili sličnog smjera, jer se na razini makroblokova smjer susjednih makroblokova ne može mijenjati s jako velikom amplitudom. Iz navedenog razloga početni makroblok pretrage odabire se na osnovu rezultata pretrage VP dobivenih za prethodni lijevi makroblok okvira koji se komprimira. Metoda je u osnovi ista kao i TSS, s razlikom u prvom koraku gdje se zbog predviđenog pomaka makrobloka pretražuje samo 1-4 bloka u usporedbi s 8 u metodi tri koraka.

Metoda 3 koraka s predviđanjem (engl. *Predictive three step search* – PTSS):

1) Korak pretrage postavlja se na S=4, određuje se oblik pretrage u ovisnosti o VP prvog lijevog makrobloka prema slici 2.4. Na primjer, ukoliko je VP u trećem kvadrantu, odabiru se 4 makrobloka pretrage prema podslici a) slike 2.4. (ishodište koordinatnog sustava je makroblok za koji se tražio njemu najsličniji) te analogno tome za sve ostale smjerove izračunatog VP lijevog susjednog makrobloka. Za izabrane makroblokove izračunava se mjera razlike SAE.

2) Korak pretrage se umanjuje za pola, na S=2, te se izračunava SAE za 8 makroblokova u okolini prethodnog dobivenog najsličnijeg makrobloka s najmanjom vrijednosti SAE iz koraka 1, udaljeni za korak pretrage.

3) Korak pretrage se postavlja na *S*=1, te se ponavlja prethodni korak izračuna. Makroblok s najmanjim SAE proglašava se najsličnijim makroblokom.



Slika 2.4. *Oblik pretrage za 1. korak PTSS metode u ovisnosti o izračunatom VP (označen crvenom strelicom) za lijevi susjedni makroblok, translatiran na koordinate trenutnog makrobloka (onoga za koji se u trenutnom koraku traži VP) [4].*

Kao dodatan element metode omogućeno je postavljanje granice minimalne vrijednosti SAE koja, ukoliko je postignuta, zaustavlja daljnju pretragu. Zbog predviđanja početnog makrobloka i odabira oblika pretrage umanjeno je vrijeme izvođenja te je u usporedbi s običnim TSS algoritmom povećana točnost pronađenih vektora pokreta.

Testiranje je provedeno na osobnom računalu nepoznatih performansi. Kao testni signali odabrana su 4 video signala. Dva video signala ("foreman" i "garden") rezolucije 352x240 elemenata slike te dva video signala ("tennis" i "Miss america") rezolucije 360x288 elemenata slike. Video signali su snimljeni u 30 okvira u sekundi, a za testiranje je korišteno prvih 100

okvira svake sekvence. Testiranje je provedeno uz dvije veličine makroblokova, 5x5 elemenata slike i 16x16 elemenata slike. Rezultati su predstavljeni pomoću omjera vršne snage signala i snage šuma (engl. *Peak signal to noise ratio* - PSNR) gdje je vršna snaga signala kvadrat najveće vrijednost elementa slike, a šum je izračunata mjera razlike MSE između okvira dobivenog iz izračunatih VP i odgovarajućeg okvira iz video signala. Osim PSNR-a, rezultati su predstavljeni i pomoću broja makroblokova s kojima se trenutni makroblok uspoređivao u ovoj metodi te u metodi pretrage u tri koraka. Granica SAE kod makroblokova 5x5 elemenata slike postavljena je na 50, a kod makroblokova 16x16 elemenata slike na 512. Rezultati su prikazani tablicama 2.3., 2.4., 2.5. i 2.6.

Tablica 2.3. *Srednja vrijednost PSNR svih okvira video signala za pronađene najsličnije makroblokove 5x5 elemenata slike pomoću metoda TSS, PTSS i PTSS s granicom [4].*

	TSS	PTSS	PTSS s granicom
foreman	34.30	34.42	34.40
garden	20.59	21.67	21.67
tennis	26.66	26.99	26.99
Miss america	38.26	38.34	38.09

Tablica 2.4. *Srednja vrijednost broja makroblokova s kojima se uspoređivao trenutni makroblok za makroblok 5x5 elemenata slike za sve makroblokove na svim okvirima pojedine testne sekvence pomoću metoda TSS, PTSS i PTSS s granicom [4].*

	TSS	PTSS	PTSS s granicom
foreman	24.47	19.08	13.10
garden	24.44	19.50	16.89
tennis	24.44	18.66	16.68
Miss america	24.51	19.37	11.75

Tablica 2.5. *Srednja vrijednost PSNR svih okvira za pronađene najsličnije makroblokove 16x16 elemenata slike pomoću metoda TSS, PTSS i PTSS s granicom [4].*

	TSS	PTSS	PTSS s granicom
foreman	32.57	32.61	32.58
garden	22.20	22.37	22.37
tennis	25.05	24.98	24.98
Miss america	37.24	37.26	37.21

Tablica 2.4. *Srednja vrijednost broja makroblokova s kojima se uspoređivao trenutni makroblok za makroblok 16x16 elemenata slike za sve makroblokove na svim okvirima pojedine testne sekvence pomoću metoda TSS, PTSS i PTSS s granicom [4].*

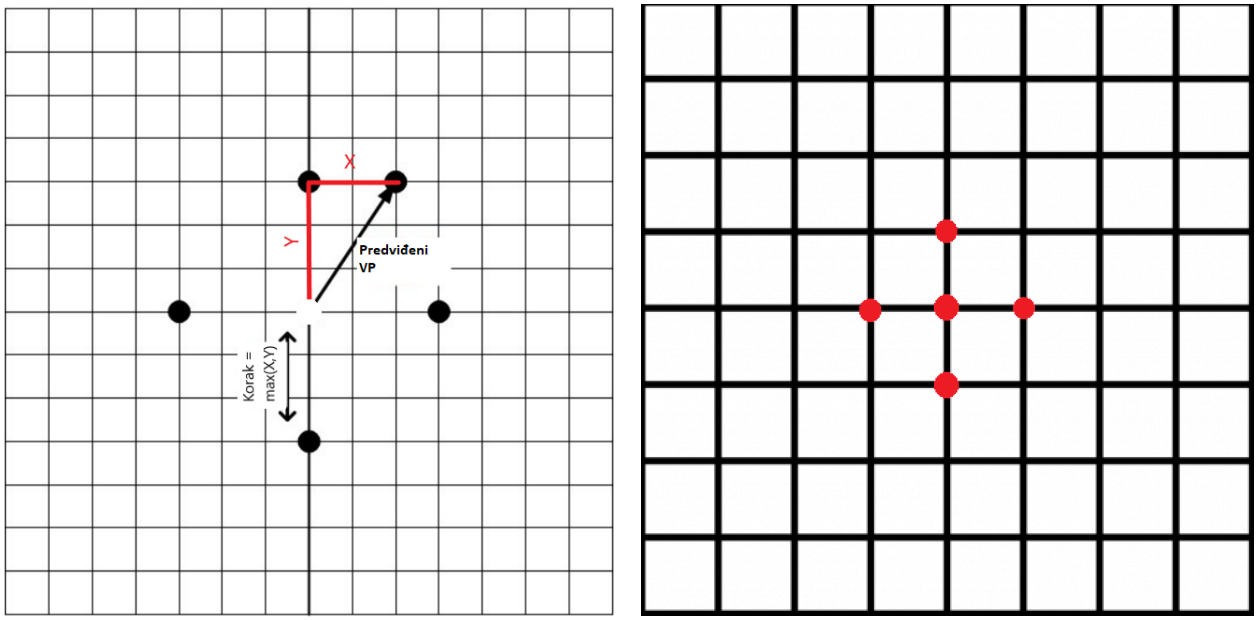
	TSS	PTSS	PTSS s granicom
foreman	23.29	18.20	15.04
garden	23.24	18.60	16.99
tennis	23.24	17.76	16.53
Miss america	23.45	18.48	12.33

Ovakvom metodom povećava se vjerojatnost pronalaska makroblokova najmanje mjere razlike (u nastavku točni makroblokovi) te se ostvaruje poboljšanje brzine izvođenja do 20% u usporedbi s TSS metodom. U slučaju kompresije video signala mjera PSNR predstavlja koliko su okvir dobiven iz vektora pokreta i stvarni okvir slični, no ne znači nužno da pronađeni vektori pokreta predstavljaju stvarni pomak makrobloka. Zbog ignoriranja određenog broja makroblokova u prvom koraku ove pretrage, postoji mogućnost da se stvarni najsličniji makroblok nalazi u tome području te će biti izostavljen od usporedbe. Takav slučaj pojavljuje se u području okvira na kojima se nalaze rubovi objekata. Autori rada [4] su takvu grešku umanjili upotrebom granice mjere razlike SAE.

2.3. Unaprijeđena metoda adaptivnog uzorka pretrage u obliku križa

Autori rada [5] svoj algoritam zasnivaju na metodi adaptivnog uzorka pretrage u obliku križa (engl. *Adaptive rood pattern search* - ARPS). ARPS metoda zasniva se na pretpostavki da se pokreti unutar okvira odvijaju koherentno, tj. ukoliko postoji pokret jednog makrobloka i njegova okolina se također kreće, odnosno u lokalnim minimumima (bliskoj okolini) pokreti su slični ili isti. Iz navedenog razloga se na osnovu prethodno izračunatog VP lijevog makrobloka određuje mogući VP trenutnog makrobloka te se kao jedna od 5 početnih točaka pretrage odabire predviđena točka pomoću VP lijevog makrobloka (VP lijevog makrobloka translatiran na mjesto trenutnog makrobloka), 2 točke lijevo-desno od trenutnog makrobloka i 2 točke gore-dolje udaljene od središta trenutnog makrobloka za duljinu dulje katete predviđenog VP (jedna kateta je horizontalni pomak makrobloka, a druga vertikalni pomak makrobloka), prikazano na podslici a) slike 2.5. Nakon pronalaska makrobloka s najmanjom mjerom razlike prelazi se na oblik pretrage malog dijamanta (oblik dijamanta gdje su vrhovi makrobloka u pretrazi udaljeni za 1 element slike od središta makrobloka te središte tog oblika za koji se traži najsličniji – kao na

podslici b) slike 2.5. s korakom od 1 elementa slike). Takav oblik pretrage se ponavlja dok najmanju mjeru razlike ne poprimi središnji makroblok u obliku pretrage malog dijamanta.



a) b)

Slika 2.5. *Oblik pretrage ARPS metode a) u prvom koraku ARPS metode s označenom duljinom koraka dobivenom iz veće katete predviđenog VP (crne točke označavaju lokaciju pretrage), b) u drugom koraku ARPS metode (oblik malog dijamanta, crvene točke označavaju lokaciju pretrage).*

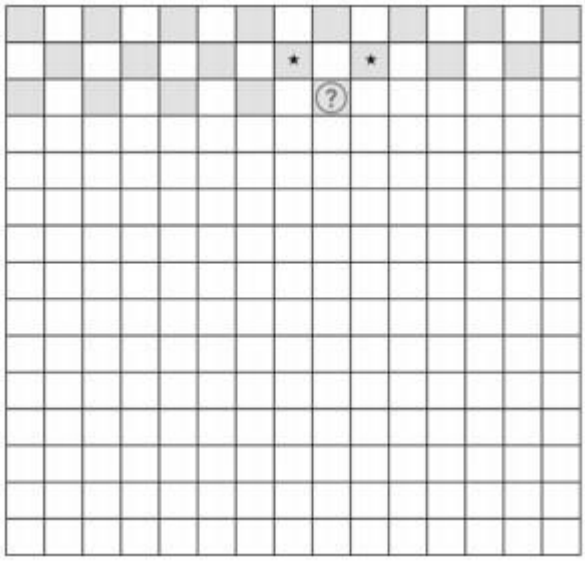
VP se može predstaviti kao suma dvaju vektora, horizontalnog i vertikalnog. Autori rada [5] su iz istraživanja prirode okvira uočili kako se pokreti pretežito odvijaju u horizontalnom smjeru, i stoga horizontalni vektor ima veću amplitudu. Na osnovu te pretpostavke početni oblik ARPS metode su izmijenili tako da umjesto 4 točke vrhova križa uzimaju 4 točke oblika slova *x* (gore lijevo, dolje lijevo, gore desno i dolje desno) gdje se za horizontalnu duljinu kraka slova *x* uzima se veći krak predviđenog VP, a vertikalna duljina kraka je 1 element slike. Metodu su nazvali ARPS_F (engl. *Adaptive rood pattern search F*). U slučaju da se makroblok s najmanjom mjerom razlike nalazi u sredini, gore lijevo ili dolje desno, za sljedeći oblik pretrage, s upola umanjenim korakom, ponovno se odabire oblik križa, u suprotnom se odabire oblik malog dijamanta.

Metoda koja je predložena u radu [5] je testirana u usporedbi s ARPS metodom te nekolicinom drugih metoda pretrage. Testiranje je provedeno na 5 video signala QCIF i CIF rezolucije na osobnom računalu nepoznatih karakteristika. Rezultati su predstavljeni srednjim brojem točaka u

kojima se tražio najsličniji makroblok te mjerom PSNR. Srednji broj točaka pretrage je umanjen, no PSNR predložene metode ARPS_F je u manji u usporedbi s ARPS metodom. Iako metoda ostvaruje manje vrijeme pretrage, manji PSNR ukazuje na to da pronađeni VP nisu u potpunosti točni, odnosno da pronađeni najsličniji makroblok nije dovoljno sličan. Izrazito poboljšanje uočeno je u usporedbi s metodom pune pretrage i metodom 3 koraka, no vrlo mala razlika je uočena u usporedbi s postojećom metodom ARPS. Metoda ARPS_F iziskuje složeniji pristup kod izrade rješenja za implementaciju u odnosu na ARPS metodu te zbog lošijeg PSNR-a nije prihvatljiva za implementaciju ukoliko se bira između ARPS i ARPS_F metode.

2.4. Procjena pokreta strategijom predvidivog šahovskog uzorka

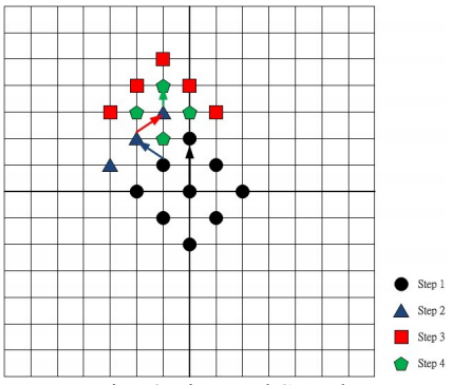
Rješenje koje je predstavljeno radom [6], kao i u većini *state-of-the-art* metoda, zasniva se na pretpostavci korelacije susjednih vektora pokreta te vremenski koreliranih VP između dvaju susjednih okvira. Autori rada [6] predlažu pristup u kojemu se koriste dva različita algoritma pretrage korištenjem predikcije inicijalnog VP pomoću VP izračunatih za okolne makroblokove, a odabir koji algoritam će biti primijenjen je određen podjelom svih makroblokova na crne i bijele, raspoređene kao na šahovskoj ploči, prikazano na slici 2.6.



Slika 2.6. *Prikaz crnih i bijelih makroblokova kod pretrage strategijom šahovske ploče [6]*.

Metoda je opisana u 3 koraka. Najprije se traže privremeni VP za crne makroblokove. Uzorak pretrage su makroblokovi koji su određeni predviđenim VP iz gornjeg lijevog, gornjeg desnog te vremenski koreliranog VP za taj makroblok (VP makrobloka na istoj lokaciji iz prethodnog okvira). Makroblok s kojim je usporedba dala najmanju mjeru razlike određuje se kao

privremeni VP. Ukoliko je mjera razlike ispod neke zadane granice, proces pretrage se zaustavlja, u suprotnom se pretraga nastavlja. Oko dobivenog najsličnijeg makrobloka pokreće se pretraga u oblika dijamanta (prikazano slikom 2.7.), nakon čega se odabire makroblok s najmanjom mjerom sličnosti od uspoređenih.



Slika 2.7. *Pretraga u obliku dijamanta (crna boja odnosi se na prvi korak, plava na drugi, crvena na treći i zelena na četvrti) [5].*

Nakon pronalaska vektora pokreta za crne makroblokove, računaju se vektori pokreta za bijele makroblokove. Za uzorak pretrage uzimaju se makroblokovi izračunati pomoću 8 susjednih - crnih makroblokova, središnji makroblok (0,0) te makroblok čija je lokacija izračunata pomoću vektora pokreta za makroblok na istoj lokaciji u prethodnom okviru. Ukoliko je pronađen najsličniji makroblok s mjerom razlike manjom od zadane granice, pretraga se zaustavlja, u suprotnom se započinje pretraga u obliku dijamanta. Kada je pronađeni najsličniji makroblok onaj u središtu oblika dijamanta ili je mjera sličnosti manja od zadane granice, pretraga se zaustavlja, u suprotnom se nastavlja pretraga u obliku dijamanta.

Kao zadnji korak ove metode primjenjuje se ispravak VP crnih makroblokova. Izračunava se lokacija najsličnijeg makrobloka na osnovu VP 4 susjedna makrobloka - gornjeg, donjeg, lijevog i desnog te se izračunava mjera razlike za isti. Ukoliko je mjera razlike manja od zadane granice pretraga se zaustavlja, a u suprotnom se započinje s pretragom oblika dijamanta, s ishodištem u prethodno pretpostavljenom najsličnijem makrobloku, sve dok najsličniji makroblok nema mjeru razlike manju od granice ili nije u središtu pretrage u obliku dijamanta.

Ovakav pristup omogućava ubrzani pronalazak vektora pokreta u usporedbi s metodom pune pretrage, veću vrijednost PSNR mjere za okvire dobivene pomoću pronađenih VP nego u metodama kao što su ARPS ili TSS. Testiranja je provedeno na 3 grupe video signala. Prva grupa je rezolucije 176x144 elemenata slike s 300 okvira, druga grupa je rezolucije 352x288

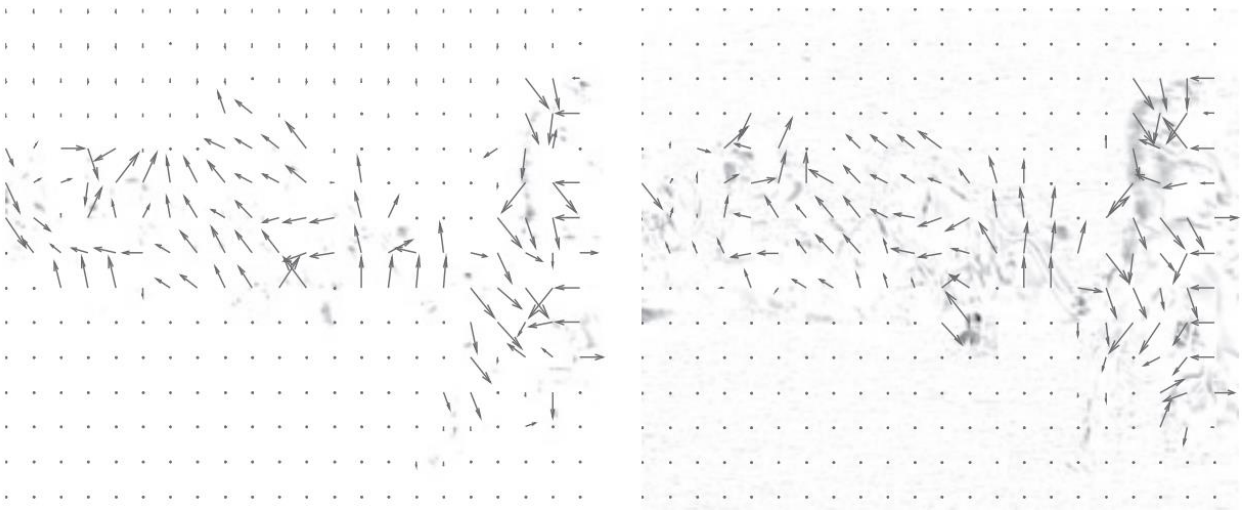
elemenata slike s 50-300 okvira, a treća rezolucije 1920x1080 elemenata slike sa 100-300 okvira, dok je zadana veličina makrobloka 16x16 elemenata slike. Karakteristike osobnog računala na kojemu je metoda testirana nisu zadane. Preporučena metoda dala je rezultate kvalitativno najsličnije onima pune pretrage - koje je u kvalitativnom smislu najbolje rješenje (najveći PSNR), te brzine pretrage najmanje od svih uspoređenih metoda.

Rješenje uzima u obzir i pretragu područja uniformnog dijela slike, koje je moguće preskočiti u određenim slučajevima jer su u takvim dijelovima ne očekuje pomak. Ovakva metoda nailazi na problem kod implementacije zbog složene arhitekture rješenja u višekoračnosti kao i zahtjevu za paralelizacijom - vektori pokreta za bijele makroblokove se počinju pretraživati paralelno sa crnim makroblokovima nakon što je određeni broj crnih makroblokova pretražen.

2.5. Brzi algoritam za usporedbu makroblokova zasnovan na korelaciji vektora pokreta i integralnoj projekciji

Metoda koju autori rada [7] predstavljaju zasniva se na promatranju VP dobivenih metodom pune pretrage. Na osnovu rezultantnih VP metode pune pretrage donesene su dvije pretpostavke. VP dobivenih za neki manji dio okvira su sličnih karakteristika i VP u kratkom vremenskom razmaku su također sličnih ili istih karakteristika. Prva pretpostavka omogućava predviđanje VP trenutnog makrobloka na osnovu okolnih VP, a druga pretpostavka omogućava da se na osnovu VP makrobloka na istoj lokaciji iz prethodnog okvira video signala predvidi VP trenutnog makrobloka. Vremenska i prostorna sličnost u radu [7] je iskazana slikom 2.8. Određivanje početne lokacije pretrage u radu nazvano ISC (engl. Initial search center), odnosno inicijalni centar pretrage. Za određivanje ISC-a koristi se skup VP u okolini makrobloka trenutnog okvira te makrobloka i okoline makrobloka, na istoj lokaciji kao i trenutni makroblok, iz prethodnog okvira video signala za širinu pretrage w=7 elemenata slike (7 u sva 4 smjera (gore, dolje, lijevo, desno) od lokacije središnjeg makrobloka) i veličinu makrobloka 16x16 elemenata slike. Navodeći drugu pretpostavku o vremenskoj korelaciji, autori rada [7] naglašavaju da su vremenski korelirani makroblokovi s manjom mjerom razlike važniji u procjeni ISC-a. VP koji su izračunati uz manju mjeru razlike makroblokova daju točniju informaciju o tome gdje se makroblok pomaknuo i stoga su i relevantniji u predviđanju pomaka trenutnog makrobloka. Na osnovu te dvije pretpostavke i proširenja druge pretpostavke, odabrane su težine skupa za određivanje ISC-a. Uz navedene parametre, metoda nalaže izračun granice najveće i najmanje mjere razlike prema integralnoj projekciji objašnjenoj u radu [7, str.4]. Ukoliko je mjera razlike za makroblok na ISC lokaciji manja od donje zadane granice, VP je pronađen, ukoliko je

vrijednost mjera razlike između gornje i donje granice nastavlja se pretraga korištenjem metode 2 stupnja pretrage (engl. *Dual stage search* – DSS) opisane u radu [7], te ukoliko je vrijednost mjere razlike veća od gornje granice primjenjuje se puna pretraga u okolini makrobloka ISC na zadanoj udaljenosti (nije navedena u radu). Ovakva metoda omogućava veliku preciznost i izrazito smanjenje vremena trajanja proračuna kod okvira s velikom količinom pokreta.



Slika 2.8. *Prikaz pronađenih VP na dvama uzastopnim okvirima video signala iz rada [7].*

Testiranje je provedeno na 100 okvira video signala različitih količina i brzina pokreta. Rezultati su predstavljeni u usporedbi s drugim jednostavnijim metodama usporedbe makroblokova. Za video signale u kojima postoji velika količina pokreta, predložena metoda je višestruko brža i ima izrazito veću vrijednost PSNR-a. S druge strane, za slike s manjom količinom pokreta ostvaruje bolje rezultate samo u usporedbi s metodom pune pretrage dok u usporedbi s metodom pretrage u obliku dijamanta i metodom tri koraka ima bolje, ali ne znatno bolje rezultate. Ovakva metoda iziskuje postojanje povratne sprege i velikog memorijskog otiska zbog iskorištavanja vremenske zalihosti VP. Osim navedenog, u radu postoji velika količina metrike dobivene na prethodno analiziranim video slijedovima koja je korištena kod izračuna određenih koeficijenata, što u sumnju dovodi adaptivnost implementacije u generalne svrhe.

2.6. Višenitno određivanje pokreta objekata - pristup grupiranja vektora pokreta i modeliranja pozadine

Metode za pronalazak VP koje su opisivane do sada kao rezultat daju grupu VP za okvir. Sama grupa VP ne predstavlja informaciju o tome postoji li jedan objekt koji se kreće ili više njih,

odnosno grupa dobivenih VP ne ukazuje ni na što drugo osim na to da su se makroblokovi pomaknuli između dvaju susjednih okvira. Kako bi se dobila informacija o broju objekata koji se kreću i smjeru te jačini njihova pokreta, iste je potrebno grupirati. Grupiranju se može pristupiti iz pogleda sličnosti karakteristika VP kao što su kut VP, duljina i položaj VP unutar okvira. Zbog nesavršenosti dobivenih VP (nesavršeni jer mogu ukazivati na najsličniji makroblok, ali to ne mora značiti da sadržaj makrobloka predstavlja objekt koji se kretao u tome smjeru) nije moguće primjerice izračunati broj različitih vrijednosti kutova VP i iz toga zaključiti koliko je različitih objekata ili grupirati VP unutar okvira na nekoliko grupa prema lokaciji, jer je za to potrebna početna informacija o tome koliko se objekata nalazi na okviru. Kako bi se što ispravnije grupiralo VP i naposlijetku odredilo što je objekt i gdje se kreće, potrebno je koristiti što više karakterističnih veličina VP te ih na smislen način analizirati. Jedno od mogućih rješenja navedenog problema grupiranja VP predstavljeno je u nastavku ovoga rada.

Autori rada [8] predlažu rješenje za određivanja smjera kretanja objekata u slučaju suprotnih smjerova kretanja u istome okviru, koristeći se paralelizacijom, smanjivanjem šuma u okviru uklanjanjem pozadine i sumiranjem toka vektora.

Osnovna metoda u radu je praćenje pokreta na razini ključnih točaka i praćenja ključnih točaka dobivenih na rubovima objekata koji se kreću. Pronalazak su podijelili na 4 niti gdje svaka nit obrađuje jednu četvrtinu slike (označeno na slici 2.9. slovima A, B, C i D) uz područja preklapanja između njih (označeno na slici 2.9. brojevima 1, 2, 3, 4 i 5). Za grupiranje VP autori predlažu korištenje izračuna centroida i grupiranju vektora oko centroida koristeći se grupiranjem algoritmom k srednjih vrijednosti (engl. K-means clustering) [9]. Svaka grupa nakon toga predstavlja jedan objekt. Rješenje je testirano na video signalu nadzorne kamere gdje su objekti koji se kreću ljudi. Testiranje je provedeno na dva računala, jednojezgrenom i dvojezgrenom računalu. Rezultati su dali točnosti od 70%+ za metodu bez uklanjanja pozadine te točnost od 90%+ za metode s uklanjanjem pozadine. Ovakvo rješenje omogućava brzo grupiranje vektora pokreta zbog dobre raspodjele na niti. Ograničenje ovog rada je pronalazak VP na razini ključnih točaka zbog kojih je potrebno koristiti složene metode za izdvajanje značajki rubova. Uz to, uklanjanje pozadine predstavlja problem za sebe koji može unijeti i određenu pogrešku kod izračuna pokreta.

Slika 2.9. *Prikaz rasporeda dijelova okvira na niti [8]*.

3. IMPLEMENTACIJA METODA ZA PRORAČUN VEKTORA POKRETA NA REALNU ADAS PLATFORMU I NJIHOVO GRUPIRANJE PREMA SLIČNOSTI

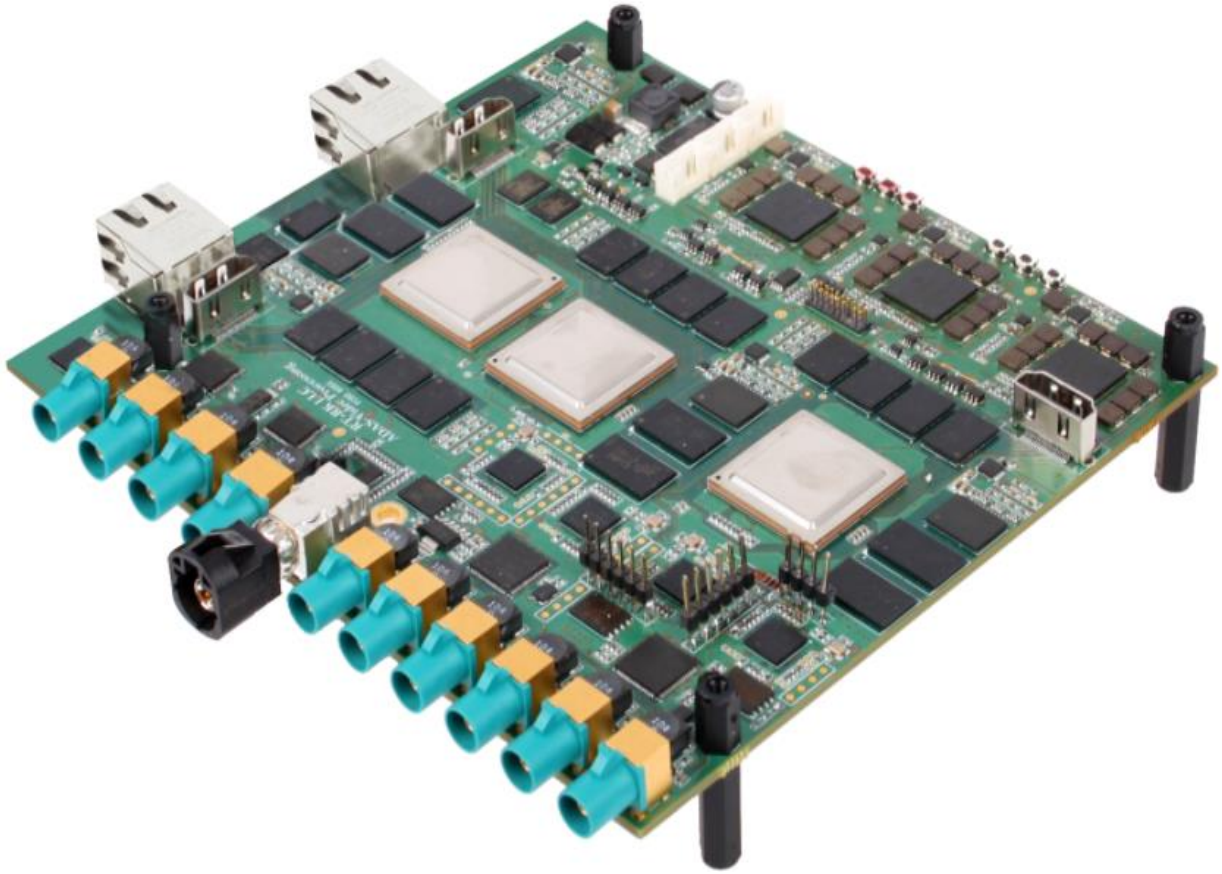
U ovome poglavlju će se detaljno predstaviti koncept rješenja za pronalazak VP i njihovo grupiranje, platforma na kojoj je rješenje realizirano te cjelokupna arhitektura implementiranog rješenja na realnu ADAS platformu. Rješenje će se najprije opisati konceptualno kroz dijagrame toka i zatim će se objasniti svaka od funkcionalnih cjelina rješenja. Poseban osvrt bit će dan za realnu ADAS platformu, odnosno njen hardver (engl. *Hardware*) koji je korišten u realizaciji predloženog rješenja, kao što su procesori i sučelja koja posjeduje te razvojno okruženje Vision SDK (engl. *software development kit* - SDK). Predloženo rješenje ima zadaću pronalaska VP u okvirima video signala koji se dostavljaju sa slikovnog senzora ADAS platforme u realnom vremenu. Koristeći se nekim pretpostavkama iz prethodnog poglavlja i potrebama korištenja ovog rješenja, predložit će se 3 metode za pronalazak VP, kojima se kombiniraju dvije osnovne zadaće takvih metoda - brzina pronalaska i točnost pronađenih VP. Završni korak rješenja je grupiranje pronađenih VP u smislene cjeline, kako bi se odredili objekti u pokretu i iscrtali na samim ulaznim okvirima koji izlaze iz procesnog dijela rješenja. Funkcionalni dio rješenja realiziran je koristeći okruženje Vision SDK, gotove module za iscrtavanje grafike, upravljanje memorijom izravnim pristupom, upravljanje tokovima video okvira, modulima za primanje i slanje video okvira, modulima za primanje i slanje preko mrežnog prilagodnika, gdje je sve realizirano u C programskom jeziku i dostupnim standardnim bibliotekama za C jezik. Detaljniji opis rješenja, platforme, korištenih alata i implementacije rješenja dan je u nastavku ovog poglavlja.

3.1. Realna ADAS platforma

Kod razvijanja ADAS rješenja za automobilsku industriju potrebno je posjedovati namjensko sklopovlje na kojem će se to rješenje izvoditi. Za realizaciju rješenja ovoga rada korištena je razvojna ugradbena ADAS platforma naziva Alpha razvojna ploča [10], proizvođača RT-RK, a njezin je prikaz na slici 3.1. Napravljena za primarno za potrebe testiranja i razvoja ADAS rješenja.

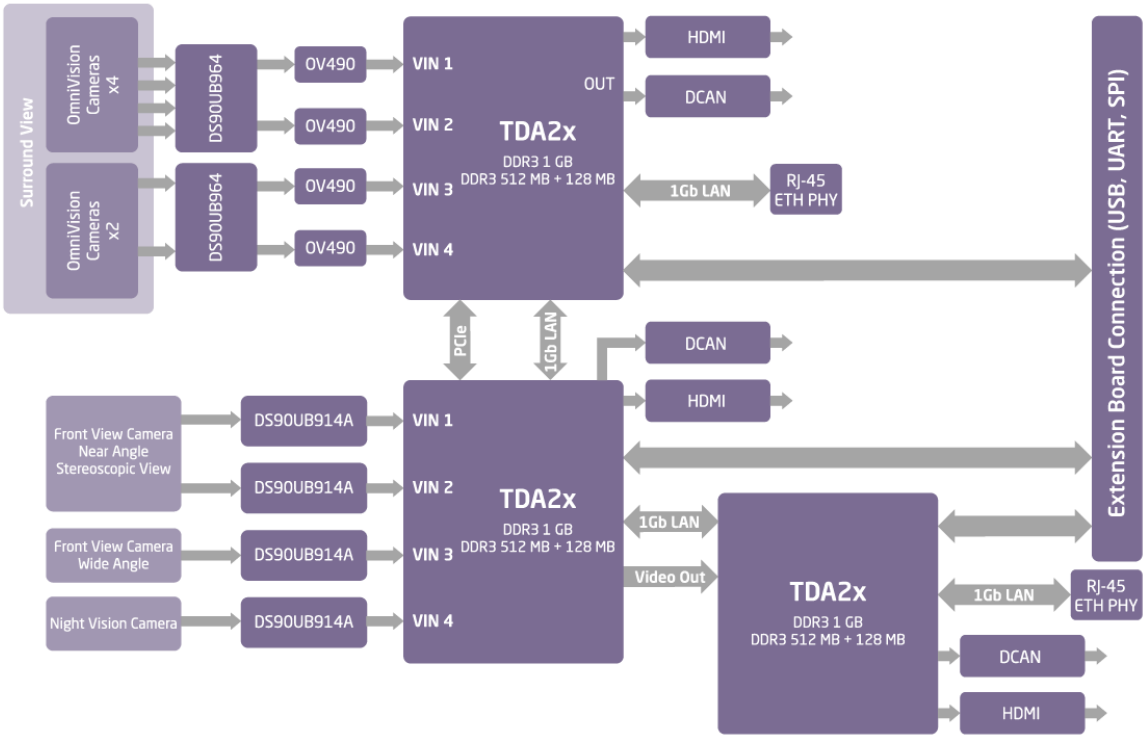
Alpha ploča, kao i svaka ADAS platforma, sadrži priključke za spajanje senzora široke namjene poput video senzora, radara, LiDAR-a, kao i priključke za programsko upravljanje kao što su UART, JTAG, Ethernet i sl. Osim navedenih, Alpha ploča na sebi sadrži HDMI priključke koji s

koriste za prikaz izlaznog video signala. Senzore, aktuatore i generalno sve komponente ulaza i izlaza ploče kontroliraju mikročipovi upakirani u nešto što se zove sustavi na čipu (engl. *System on a Chip* - SoC). SoC-evi sastoje se od 2 ili više središnjih procesnih jedinica (engl. *Central proccesing unit* - CPU).



Slika 3.1. *Alpha ploča korištena u sklopu diplomskog rada [10].*

Alpha ploča koristi SoC-eve tvrtke Texas Instruments, točnije tri TDA2x SoC-a. Svaki SoC sadrži 2 ARM Cortex A15 CPU-a, 2 ARM Cortex M4 CPU-a, 2 C66x Digital Signal Proccesor CPU-a te 4 Embedded vision Engine CPU-a. Uz navedene resurse, SoC-evi imaju pristup po 1.5GB DDR3 RAM memorije te 32MB *flash* memorije. Svaki SoC ima pristup barem jednom izlaznom HDMI, DCAN, Ethernet, JTAG, UART i MicroSD sučelju izravno priključenom na SoC ili preko PCIe sučelja [11]. Raspored SoC-eva te način vezanja komponenata prikazan je na slici 3.2. Učitavanje programskih rješenja na ploču izvršava se korištenjem JTAG priključka ili MicroSD utora pripadajućeg SoC-a.



Slika 3.2. *Arhitektura ADAS Alpha ploče [11].*

Zbog zadatka rada koji nalaže postizanje velikih brzina obrade, nužno je navesti karakteristike 2 CPU-a na kojima će se pokretati stvoreni algoritmi.

- ARM Cortex A15 - do 750 MHz, L2 Cache do 2MB,
- C66x Digital Signal Proccesor - do 750 MHz, L2 Cache do 200kB.

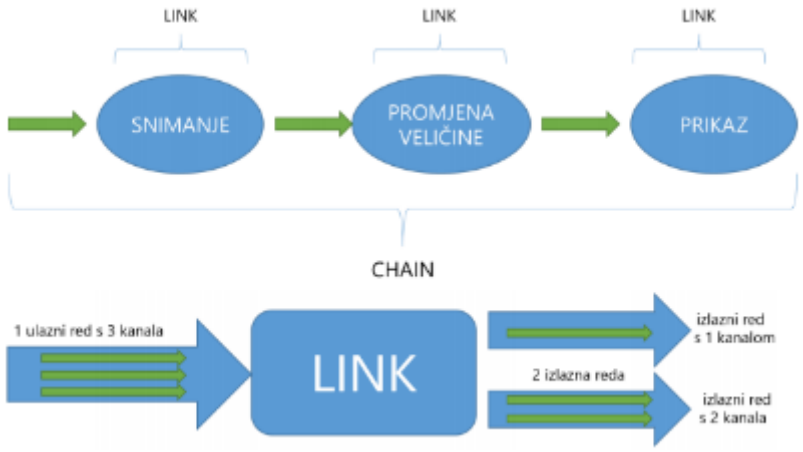
Upravljanje sklopovljem Alpha ploče postignuto je na razini operacijskog sustava Linux. Konkretna verzija koja se koristi na ploči je SysBios i pokreće se na procesoru domaćinu koji je u slučaju Alpha ploče ARM Cortex A15. Sama kontrola programskih rješenja koja se implementiraju na ploči izvršava se na drugom ARM procesoru, ARM Cortex M4 [11].

Za postavljanje programskog rješenja na sklopovlje Alpha ploče koristi se razvojno okruženje Vision SDK koje će biti opisano u sljedećem podpoglavlju.

3.2. Razvojno okruženje Vision SDK za Texas Instruments porodicu SoC-ova

Rješenja koja se koriste u ADAS razvoju potrebno je s konceptualne razine prenijeti u oblik programskog rješenja. S obzirom da se u ovome radu koristi Alpha ploča koja sadrži SoC-ove

tvrtke Texas Instruments, u ovome radu koristit će se razvojno okruženje Vision SDK. Vision SDK je razvojno okruženje napravljeno na konceptu "Veza i lanaca" (engl. *Links and Chains*), kojemu je programski pristup omogućen korisničkim programskim sučeljem (engl. *Application Programming Interface - API*) "Link API". Link API je sučelje oko kojega se stvaraju algoritmi koji vrše određene zadaće, upravljaju i obrađuju video tok. Vision SDK dolazi s gotovim rješenjima veza izgrađenim oko Link API-a za neke karakteteristične zadatke. Veze se slažu u seriju ili paralelu i tako nastaje lanac (engl. *Chain*) veza (engl. *Links*), koji predstavlja gotovo korisničko rješenje prikazano na slici 3.3.



Slika 3.3. *Prikaz koncepta „Veze i lanci“ [12].*

Osnovni oblik podatka koji je ulaz i izlaz svake veze je sistemski spremnik. Ovisno o definiranoj vezi, veza može imati jedan ili više ulaza, koji se u Vision SDK naziva red (engl. *Que*). Svaki *Que* je predstavljen jednim sistemskim spremnikom (engl. *System Buffer*). Sistemski spremnik može sadržavati jedan ili više spremnika podataka koji mogu biti tipa spremnika video okvira, toka bitova ili metapodataka. Svaki spremnik u sebi nosi karakteristični popis varijabli koji opisuju ono što se u njemu prenosi.

Osnovni *linkovi* u razvojnom okruženju Vision SDK:

- *NullSource* - veza koja dohvaća podatke s Ethernet sučelja na platformi, sprema ju u spremnik, dodjeljuje informacije o spremniku i obavještava sljedeću vezu o spremnosti spremnika za preuzimanje.
- *Null* - veza za preuzimanje spremnika podataka i slanje na Ethernet sučelje.
- *Capture* - veza koja predstavlja senzor platforme, omogućava odabir tipa podatka koji dolazi sa senzora i određen je dimenzijama podatka senzora.

- *Display* - veza za slanje video spremnika na video izlaz platforme, HDMI.
- *VPE* - veza kojom se izmjenjuje veličina ili format boje video spremnika, omogućava i povećanje i smanjivanje video okvira koristeći bilinearnu interpolaciju.
- *Dup* - veza koja kao ulaz ima jedan red, a na izlazu ima 2 reda u kojima predaje reference istog sistemskog spremnika.
- *Merge* - veza koja spaja 2 ulazna reda u jedan izlazni.
- *Sync* - veza koja uspoređuje vremenske informacije kanala reda koji ulaze u njega te ovisno o postavljenim parametrima *Sync* veze odlučuje hoće li propustiti taj red dalje ili ne, koristi se za sinkronizaciju kanala u jednom kompozitnom spremniku.
- *Alg_NameOfAlgo* - veza koja omogućava kreiranje korisnički definiranog *linka*, broj i vrste ulaza i izlaza, obrada podataka.

Cijelo okruženje napisano je u C programskom jeziku, ali omogućava i pisanje u C++-u jer je operacijski sustav kojeg Vision SDK generira zasnovan na Linux-u. Operacijski sustav i sva korisnička programska rješenja generiraju se korištenjem "*make*" alata, koji prema uputama napisanim u datotekama s ekstenzijom *.mk* stvara binarne datoteke za pokretanja rješenja na platformi. Dvije binarne datoteke su "*AppImage*", koji sadrži korisničko programsko rješenje, te "*MLO*", koji sadrži operacijski sustav za platformu.

3.3. Koncepti rješenja za pronalazak vektora pokreta

Koncepti rješenja za pronalazak VP koji su implementirani na realnu ADAS platformu u sklopu ovog rada jednostavnije su prirode od onih naprednijih koji se koriste pri kompresiji video signala, te su nadograđeni s nekoliko pretpostavki vezanih za svrhu zadatka. Cjelokupno rješenje implementirano na Alpha ploči treba riješiti problem pronalaska objekta (ili više njih) koji se kreće. U svrhu pronalaska VP implementirane su tri različite metode: metoda iscrpne pretrage (u nastavku označena s EBMA (engl. Exhaustive blockmatching algorithm, algoritam iscrpne pretrage blokova)), metoda 3 koraka s mogućnošću povećanja broja koraka (u nastavku označena s TSS-UL (engl. *Three step search – updateable length*, pretraga u tri koraka – promjenjiva duljina)) i metoda tri koraka s uključenim predviđanjem i prilagodbom na odabir oblika pretrage (u nastavku označena s PR&AD-TSS (engl. *Predictive and adaptive – three step search*, predvidljiv i prilagodljiv – pretraga u tri koraka)). Sve metode koriste komponentu svjetline video signala zapisanog u memoriju koristeći YUV model boja kao 2D područje na kojemu se traži VP. Svaka od metoda za pronalazak VP uzima makroblok iz trenutnog okvira i traži njemu najsličniji makroblok u prethodnom okviru.

Korištene su 2 pretpostavke kod svih metoda. Prva pretpostavka je da uniformni makroblok (slične ili iste vrijednosti svjetline svih elemenata makrobloka) ne predstavlja objekt koji se kreće. Iako se uniformni makroblokovi kreću između dvaju susjednih okvira, pronalazak najsličnijih makroblokova korištenjem mjere razlike dovodi do pronalaska netočnih VP (VP koji ne pokazuju stvarni pomak makrobloka) te doprinose pogrešci detekcije objekta kod koraka grupiranja VP. Razlog krivog izračuna VP je što je mjera razlike između takvih makroblokova vrlo bliska po vrijednosti. Primjer uniformnog područja okvira je prikazan na slici 3.4 te je na slici 3.5. za primjer dana vrijednost elemenata makroblokova.

Slika 3.4. *Uniformno područje jednog okvira.*

Kada bi se prema formuli (2-1) računala mjera razlike za bilo koja dva bloka u ovom području, svi parovi bi imali istu vrijednost mjere razlike, čime bi bilo vrlo teško odrediti stvarni VP. Zbog toga se takvi makroblokovi preskaču u proračunu VP jer ne pridonose informaciji o kretanju objekta te mogu pridonijeti donošenju loše odluke o smjeru kretanja, jer su vjerojatno netočnog smjera.

8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8	8	8

Slika 3.5. *Vrijednosti elemenata svjetline uniformnog područja sa slike 3.4.*

Druga pretpostavka rješenja je da, ukoliko je pronađen VP s velikom vrijednošću mjere razlike, donosi se odluka o tome da je VP za taj makroblok (0,0), što označava da se makroblok nije pomaknuo i njegova duljina je 0, zbog čega će biti izbačen u koraku grupiranja VP. Velika mjera razlike ukazuje da pronađeni VP nije stvarni pomak makrobloka, stoga se njegovim izbacivanjem želi postići izbjegavanje netočnih VP koji mogu otežati ispravno grupiranje vektora.

Koncept prve metode (EBMA) za pronalazak VP je pretraga za sličnim makroblokom metodom pune pretrage, gdje je područje pretrage na referentnom okviru odabrano kao područje u okolini makrobloka iste lokacije kao i na trenutnom okviru. Veličina područja pretrage i detaljan opis metode nalaze se u podpoglavlju s opisom programskog rješenja.

Druga metoda koja je korištena je metoda 3 koraka s mogućnošću povećanja broja koraka (TSS-UL), gdje je veličina koraka pretrage određena u ovisnosti o veličini bloka pretrage – S=7 u slučaju makrobloka veličine 8x8 i 16x16 elemenata slike, te S=15 u slučaju bloka pretrage veličine 32x32 elemenata slike.

Kao treća predložena metoda koristi se metoda tri koraka s uključenim predviđanjem i prilagodbom na odabir oblika pretrage (PR&AD-TSS) s odabranim početnim korakom S=4 u slučaju makrobloka veličine 8x8 i 16x16 elemenata slike te S=7 u slučaju makrobloka veličine 32x32 elemenata slike. U ovoj se metodi korištenjem prethodno dobivenog VP prvog lijevog susjednog makrobloka određuje početna lokacija središnjeg makrobloka u pretrazi, koji se potom uspoređuje s makroblokovima prethodnog okvira. Kako bi se smanjio broj makroblokova s kojima se radi usporedba, definiran je odabir oblika pretrage što će biti opisan u opisu rješenja u podpoglavlju s opisom programskog rješenja. Ovakav pristup smanjenja broja makroblokova za koje se obavlja pretraga odabran je prema radu [5].

3.4. Koncept rješenja za grupiranje vektora pokreta

Kada se pronađu VP za zadani okvir jednom od prethodno spomenutih triju metoda, najprije se uklanjaju vektori koji imaju nezadovoljavajuće procijenjene duljine, manji su od duljine 2 i veći su od duljine 50. Pomak makrobloka od 2 elementa slike ne smatra se pomicanjem objekta. Tijekom razvoja metoda pronalaska VP uočeno je kako su makroblokovi u područjima okvira niske prostorne frekvencije (niske, ali veće od granične vrijednosti mjere uniformnosti koja će biti opisana u podpoglavlju s opisom programskog rješenja) najsličniji na udaljenostima od jednog ili dva elementa slike. Zbog prisutnosti navedenog problema, VP duljine 2 ili manje se izbacuju prilikom grupiranja. Osim navedenog problema, tijekom razvoja metoda za pronalazak

VP uočeno je da VP duljine 50 ili više imaju veću mjeru razlike (ali manju od zadane najveće granice mjere razlike). Kada je pronađen VP duljine 50 u svakoj testnoj sekvenci, VP susjednih makroblokova uvijek su bili manje duljine i drugačijeg smjera u odnosu na VP duljine 50. Drugi razlog izbacivanja VP većih od 50 je što bi takvi VP predstavljali pomake objekta (odnosno brzinu kretanja) koji uglavnom nisu realni između dvaju susjedna okvira u video signalu s izmjenom 30 okvira u sekundi.

Svaki vektor opisan je sa 6 karakterističnih veličina - koordinate izvorišne točke VP (x_I i y_I), koordinate završne točke VP (x_Z i y_Z) te veličine izračunate iz navedenih koordinata - duljina VP i kut VP u odnosu na smjer horizontalno desno. Izračunavanje duljine i kuta izvršava se nakon pronalaska VP kako bi se taj korak izbjegao u koraku grupiranja VP te kako bi se informacija koju VP predstavlja preoblikovala u domenu koja će olakšati grupiranje VP. Karakteristične veličine se uspoređuju i grupiraju algoritmom k srednjih vrijednosti, gdje *k* predstavlja broj centroida oko kojih se pomoću Euklidske udaljenosti karakterističnih veličina elementa, u slučaju ovog rada VP, određuje pripada li element (tj. VP) grupi oko tog centroida ili ne. Za svaki centroid od njih *k* najprije se određuju početne vrijednosti svake karakteristične veličine. Te vrijednosti se najčešće određuju kao slučajni broj iz skupa opsega najmanja do najveća vrijednost za karakterističnu veličinu. Element se najprije svrsta u centroid kojem je euklidski najbliži po karakterističnim veličinama, a potom se centroid za tu grupu ponovno izračunava s dodanom vrijednošću elementa koji se dodao grupi. Postupak se ponavlja sve dok elementi ne prestanu mijenjati grupe, ili dok se ne postigne unaprijed zadani najveći broj iteracija razvrstavanja. U rješenju ovoga rada koristit će se sustavno razvrstavanje VP u nekoliko iteracija s različitim brojem grupa *k*. Osnovna strategija je razvrstavanje VP gdje je centroid predstavljen karakterističnim veličinama koordinate izvorišne točke VP, duljine i kuta VP te smanjivanjem broja grupa *k* procjenom sličnosti grupa prema srednjoj vrijednosti karakterističnih veličina svake od grupa. Konkretna implementacija ovog rješenja predstavljena je u poglavlju s opisom rješenja.

3.5. Opis programskog rješenja na osobnom računalu

Rješenje za pronalazak VP i njihovo grupiranje napisano je prvotno u obliku koda u C programskom jeziku na osobnom računalu. Razvoj rješenja na osobnom računalu omogućava brži proces otklanjanja grešaka u kodu u odnosu na razvoj rješenja izravno na ADAS platformi zbog dugotrajnog postupka izgradnje izvršnih datoteka nakon svake promjene u kodu. Razvijeno rješenje na osobnom računalu predstavlja početni oblik rješenja s kojim je zatim moguće

usporediti rješenje implementirano na ADAS platformu prema brzini izvođenja te je moguće pronaći potencijalne nepravilnosti izazvane ograničenjima ADAS platforme, kao što su pojava pogreške zbog tipa podatka ili nedovoljne količine memorije za podatke na ploči.

Za pisanje i izvođenje programskog rješenja korišteno je okruženje Microsoft Visual Studio (u nastavku MVS). Neki dijelovi rješenja su isti, neovisno o tome koja se metoda koristila za pretragu VP. Za učitane okvire u YUYV formatu boje ili YUV420SP_UV, rezolucija 1280x720 elemenata slike, najprije je potrebno izdvojiti komponentu svjetline. Za svaki makroblok u trenutnom okviru, prije nego se započne s pronalaženjem najsličnijeg, izračunava se mjera uniformnosti prema kodu prikazanom na slici 3.7. Varijabla „*upperLeftMacroBlockCoo*“ je struktura podataka koja sadrži vrijednost koordinate gornjeg lijevog elementa makrobloka (*x* i *y*). U varijabla „*frame*“ pohranjeni su podaci za vrijednost svjetline svakog elementa slike trenutnog okvira. „*WIDTH*“ i „*HEIGHT*“ su globalno definirane veličine okvira u memoriji odnosno broj elemenata slike po širini i visini.

```
int blockValueDeviation(uint8_t* frame, Point upperLeftMacroblockCoo) {
    int sum = 0;

    int y, x;
    for (y = 0; y < BLOCK_SIZE; y++) {
        for (x = 0; x < BLOCK_SIZE; x++) {
            sum = sum + frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x +
upperLeftMacroblockCoo.x)];

        }
    }
    int mean = sum / (BLOCK_SIZE * BLOCK_SIZE);

    sum = 0;
    for (y = 0; y < BLOCK_SIZE; y++) {
        for (x = 0; x < BLOCK_SIZE; x++) {
            sum += abs(mean - frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x +
upperLeftMacroblockCoo.x)]);
        }
    }

    return sum;
}
```

Slika 3.7. *Prikaz koda za izračun uniformnosti makrobloka.*

Funkcija izračunava sumu razlike pojedinog elementa slike makrobloka od srednje vrijednosti elemenata slike istog makrobloka. Nakon toga se dobivena vrijednost dijeli s brojem elemenata slike makrobloka i uspoređuje se s vrijednosti granice uniformnosti. Za granicu uniformnosti u razvijenom rješenju odabrana je vrijednost 5.0. Vrijednost je empirijski određena iterativnim postupkom na nekoliko parova okvira različitog sadržaja - promet, čovjek u pokretu, objekt u

pokretu s uniformnom pozadinom. Ukoliko je izračunata mjera uniformnosti veća od zadane granice, za makroblok se izračunava VP odnosno traži se njemu najsličniji makroblok na referentnoj slici. Ako je izračunata mjera uniformnosti manja od zadane granice, za blok se ne traži VP nego se kao VP trenutnog makrobloka dodjeljuje vrijednost koordinate gornjeg lijevog elementa slike trenutnog makrobloka čija je uniformnost prethodno izračunata – to označava da makroblok nema pomaka. Kada je određen makroblok s najmanjom mjerom razlike, provjerava se je li mjera razlike manja ili jednaka zadanoj granici za najveću mjeru razlike. Ako je vrijednost manja, nema promjene, no ukoliko je vrijednost veća, kao pronađeni makroblok se odabire koordinata trenutnog makrobloka (onoga sa koji se traži najsličini) i to se pohranjuje kao VP za trenutni makroblok (kao i u slučaju nezadovoljavanja granice mjere uniformnosti). Kao granična vrijednost mjere razlike odabrana je vrijednost „*9 × broj elemenata makrobloka*“ u slučaju izračuna SAE, te 9.0 u slučaju MSE mjere, jer je rješenje, ovisno o stupnju optimizacije koda, testirano s jednom od navedenih mjera. Kada je određen VP, za njega se izračunavaju karakteristične veličine duljine i kuta te se dodjeljuju opisnoj strukturi za taj VP koja sadrži svih 6 karakterističnih veličina VP spomenutih u podpoglavlju koncepta rješenja, prikazano na slici 3.8. Ovo su identični dijelovi koda za sve tri metode pronalaska VP opisane u nastavku ovog podpoglavlja. Funkcija „*CAST_ANGLE*“ postavlja vrijednost kuta u područje [0,360] jer funkcija „*calculateAngle2Points*“ vraća vrijednosti iz područja [-180, 180]. Funkcija „*calculateLength2points*“ izračunava Euklidsku udaljenost VP.

```
vectors[0][index] = (int16_t)privFrom.x;
vectors[1][index] = (int16_t)privFrom.y;
vectors[2][index] = (int16_t)privTo.x;
vectors[3][index] = (int16_t)privTo.y;
vectors[4][index] = (int16_t)calculateLength2Points(privFrom, privTo);
vectors[5][index] = (int16_t)
                    CAST_ANGLE(calculateAngle2Points(privTo,privFrom));
```

Slika 3.8. *Prikaz koda za dodjelu vrijednosti karakteristika vektoru.*

Programski kod rješenja na osobnom računalu za sve 3 metode pretrage VP i njihovo grupiranje nalaze se u elektroničkom prilogu P.3.1.

3.5.1. Implementacija metoda za pronalazak vektora pokreta u C programskom jeziku

Prva implementirana metoda je EBMA. Pomoću koordinata gornje lijeve točke trenutnog makrobloka (na slici 3.9. označene kao „*prevMacroblockCoo*“) određuju se koordinate koje definiraju područje pretrage na prethodnom okviru koje će biti pohranjene u polje strukture

točaka označene s „*pointsSA*“ i koje predstavljaju koordinate gornje lijeve i donje desne točke područja pretrage. Granice područja pretrage određuju se funkcijom „*getSearchArea*“ sa slike 3.9. Svaka granica udaljena je od rubova makrobloka pretrage za fiksno određenu udaljenost – 25 u slučaju makrobloka veličine 32x32 elementa slike, odnosno 15 u slučaju makrobloka veličine 8x8 ili 16x16 elemenata slike. Vrijednost udaljenosti određena je prema najvećem željenom dosegu pretrage, odnosno najvećem pomaku koji je trenutni makroblok mogao ostvariti. Odabrane udaljenosti pokazale su se dovoljnima na većini testnih video signala, što će biti prikazano u poglavlju testiranja rješenja. Koordinate gornje lijeve točke područja pretrage nalaze se za odabranu udaljenost elemenata slike, gore te lijevo, od koordinata gornje lijeve točke trenutnog makrobloka (po x i y osi), pazeći pri tome da koordinate ne poprime negativne vrijednosti. Koordinate donje lijeve točke područja pretrage nalaze se dolje i desno od koordinata donje desne točke trenutnog makrobloka (po x i y osi), pazeći pri tome da su vrijednosti koordinata veće od „*širina - veličina makrobloka*“ u slučaju x koordinate i „*visina - veličina makrobloka*“ u slučaju y koordinate.

```
void getSearchArea(Point prevMacroblockCoo, int searchArea, Point pointsSA[2]) {

    int x = prevMacroblockCoo.x - searchArea;
    if (x < 0) { x = 0; }
    int y = prevMacroblockCoo.y - searchArea;
    if (y < 0) { y = 0; }
    pointsSA[0].x = x;
    pointsSA[0].y = y;
    x = prevMacroblockCoo.x + BLOCK_SIZE + searchArea;
    if (x > WIDTH) { x = WIDTH - BLOCK_SIZE; }
    y = prevMacroblockCoo.y + BLOCK_SIZE + searchArea;
    if (y > HEIGHT) { y = HEIGHT - BLOCK_SIZE; }
    pointsSA[1].x = x;
    pointsSA[1].y = y;

}
```

Slika 3.9. *Prikaz koda za određivanje područja pretrage pri implementaciji metode iscrpne pretrage (EBMA) za pronalazak VP.*

Svaka lokacija unutar područja pretrage na prethodnom okviru predstavlja gornju lijevu točku makrobloka koji se uspoređuje s makroblokom trenutnog okvira. Ukoliko je mjera razlike manja od prethodno sačuvane najmanje (inicijalno je sačuvana vrijednost 9999), pohranjuje ju kao novu najmanju i pohranjuje lokaciju makrobloka, onaj s kojim se uspoređivao trenutni makroblok, kao lokaciju najsličnijeg makrobloka, ukoliko nije, nema promjene. Algoritam ponavlja postupak sve dok ne dođe do kraja područja pretrage odnosno donje desne točke područja pretrage.

Druga metoda (TSS-UL) napisana je po uzoru na metodu pronalaska VP u 3 koraka. Koordinata gornjeg lijevog ruba trenutnog makrobloka za koji se treba izvršiti pretraga najprije se pomiče dolje i desno za pola veličine makrobloka (16 u slučaju makrobloka veličine 32x32 elementa slike, 8 u slučaju makrobloka veličine 16x16 elemenata slike i 4 u slučaju makrobloka veličine 8x8 elemenata slike), ne bi li se dobila središnja točka trenutnog makrobloka. Kada je određeno središte pretrage, ulazi se u petlju koja je ograničena veličinom koraka koji se u svakoj iteraciji petlje umanjuje za pola, sve dok ne postigne vrijednost manja od 2 (jer je korak cjelobrojni tip podatka zbog čega se kod dijeljenja zaokružuje na manji cijeli broj). Zbog navedenog je moguće da se pretraga odvija u više od 3 koraka u usporedbi s metodom 3 koraka, što omogućava iscrpniju pretragu okoline trenutnog makrobloka (više makroblokova prethodnog okvira s kojima će se usporediti trenutni makroblok s trenutnog okvira). Početni korak pretrage za veličine makrobloka 8x8 i 16x16 elemenata slike je 7, a za veličinu makrobloka 32x32 elementa slike je 15. U svakoj iteraciji petlje najprije se određuje 9 točaka pretrage prema slici 3.10. Točka na adresi polja točaka „*points[0]*“ je početna lokacija pretrage, odnosno središnja točka makrobloka s prethodnog okvira, koja je odabrana u nekoj od prethodnih iteracija kao lokacija makrobloka s najmanjom mjerom razlike.

```
points[1].x = points[0].x;              //gore
points[1].y = points[0].y - step;
points[2].x = points[0].x;              //dolje
points[2].y = points[0].y + step;
points[3].x = points[0].x - step;       //lijevo
points[3].y = points[0].y;
points[4].x = points[0].x + step;       //desno
points[4].y = points[0].y;
points[5].x = points[0].x - step;       //gore lijevo
points[5].y = points[0].y - step;
points[6].x = points[0].x + step;       //gore desno
points[6].y = points[0].y - step;
points[7].x = points[0].x - step;       //dolje lijevo
points[7].y = points[0].y + step;
points[8].x = points[0].x + step;       //dolje desno
points[8].y = points[0].y + step;
```

Slika 3.10. *Prikaz koda za određivanje točaka pretrage pomoću vrijednosti koraka (step) metode TSS-UL.*

Za svaku od 9 točaka izvršava se sljedeći postupak:

1. S obzirom da točka predstavlja središte makrobloka, određuje se gornja lijeva točka makrobloka,
2. izračunava se mjera razlike makrobloka trenutnog okvira i makrobloka referentnog okvira na lokaciji trenutne točke,

3. provjerava se je li izračunata mjera razlike najmanja do sada; ako je, sprema se kao najmanja i točka se sprema kao lokacija najsličnijeg makrobloka.

Kada se završi usporedba s makroblokovima na lokaciji svake odabrane točke sačuvane u polju „*points*“, korak pretrage se umanjuje za pola. Nakon toga slijedi provjera granice mjere razlike (prema postupku opisanom na početku ovog podpoglavlja) i dodjeljuje se VP za makroblok trenutnog okvira za koji je izvršavana pretraga.

Treća metoda (PR&AD-TSS) sastoji se od dijelova metode 3 koraka te određivanju lokacija na kojima se ne izvršava pretraga. Uz navedeno, lokacija središta pretrage odabire se predikcijom, točnije dobivenim VP za prethodni lijevi makroblok trenutnog okvira. Pomak koji je ostvaren po x i y osima, za prethodni lijevi makroblok trenutnog okvira, dodaje se koordinati trenutnog makrobloka. Za tako dobivenu koordinatu makrobloka određuju se njegova središnje koordinata te algoritam ulazi u petlju čiji je kraj uvjetovan veličinom koraka, analogno metodi 3 koraka koja je prethodno opisana. Kao i kod metode 3 koraka, određuje se 9 točaka pretrage. U prvoj iteraciji prolazi se kroz svih 9 točaka pretrage za koje se računa mjera razlike. Ovisno o tome u kojoj od točaka je pronađena najmanja mjera razlike, odabire se hoće li se umanjiti korak za pola i koje će se lokacije od 9 točaka u sljedećoj iteraciji izostaviti iz usporedbe, s iznimkom da se središnja točka („*points[0]*“) u svakoj sljedećoj iteraciji izostavlja od usporedbe. Odluka o tome koje će točke biti izostavljanje, prikazana je na slici 3.11.

```
if (idBest == 0) {//središnja točka

            skips[1] = 0;//gornja
            skips[2] = 0;//donja
            skips[3] = 0;//lijeva
            skips[4] = 0;//desna
            skips[5] = 0;//gornja lijeva
            skips[6] = 0;//gornja desna
            skips[7] = 0;//donja lijeva
            skips[8] = 0;//donja desna
            step = step / 2;

        }
        else if (idBest == 1) { // gornja točka
            skips[1] = 0;
            skips[2] = 1;
            skips[3] = 1;
            skips[4] = 1;
            skips[5] = 0;
            skips[6] = 0;
            skips[7] = 1;
            skips[8] = 1;
        }
        else if (idBest == 2) { //donja točka
            skips[1] = 1;
            skips[2] = 0;
```

```
        skips[3] = 1;
        skips[4] = 1;
        skips[5] = 1;
        skips[6] = 1;
        skips[7] = 0;
        skips[8] = 0;



    }
    else if (idBest == 3) {//lijeva točka
        skips[1] = 1;
        skips[2] = 1;
        skips[3] = 0;
        skips[4] = 1;
        skips[5] = 0;
        skips[6] = 1;
        skips[7] = 0;
        skips[8] = 1;


    }
    else if (idBest == 4) {//desna točka
        skips[1] = 1;
        skips[2] = 1;
        skips[3] = 1;
        skips[4] = 0;
        skips[5] = 1;
        skips[6] = 0;
        skips[7] = 1;
        skips[8] = 0;

    }
    else if (idBest == 5) {//gornja lijeva
        skips[1] = 0;
        skips[2] = 1;
        skips[3] = 0;
        skips[4] = 1;
        skips[5] = 0;
        skips[6] = 0;
        skips[7] = 0;
        skips[8] = 1;

    }
    else if (idBest == 6) {//gornja desna
        skips[1] = 0;
        skips[2] = 1;
        skips[3] = 1;
        skips[4] = 0;
        skips[5] = 0;
        skips[6] = 0;
        skips[7] = 1;
        skips[8] = 0;

    }
    else if (idBest == 7) {//gornja lijeva
        skips[1] = 1;
        skips[2] = 0;
        skips[3] = 0;
        skips[4] = 1;
        skips[5] = 0;
        skips[6] = 1;
```

```
        skips[7] = 0;
        skips[8] = 0;

    }
    else if (idBest == 8) {//gornja desna
        skips[1] = 1;
        skips[2] = 0;
        skips[3] = 1;
        skips[4] = 0;
        skips[5] = 1;
        skips[6] = 0;
        skips[7] = 0;
        skips[8] = 0;

    }
```

Slika 3.11. *Kod za određivanje veličine sljedećeg koraka pretrage i odlučivanje o tome koje će se točke preskočiti kod sljedeće iteracije pretrage PR&AD-TSS metode za pronalazak VP.*

3.5.2. Implementacija metode za grupiranje vektora pokreta u C programskom jeziku

Grupiranje vektora odvija se korištenjem metode k srednjih vrijednosti opisanoj u prethodnom podpoglavlju. Najprije se odvoje svi VP s duljinom manjom od 2 i većom od 50, zbog smanjenja greške grupiranja koja je opisana u prethodnom podpoglavlju. Centroidi se u ovom algoritmu inicijaliziraju tako da se odrede najveća i najmanja vrijednost za svaku od 6 karakterističnih veličina VP te se tada za prvih 6 grupa određuje vrijednosti svake karakteristične veličine centroida prema slici 3.12.

```
    means[i][ftr_i] = getRandom(&minima[ftr_i], &maxima[ftr_i]);
    switch (i) {
    case 0: {
        inter = minima[ftr_i];
        means[i][ftr_i] = inter;
        break;
    case 1: {
        inter = maxima[ftr_i];
        means[i][ftr_i] = inter;
        break;            }
    case 2: {
        inter = abs(maxima[ftr_i] - minima[ftr_i]);
        inter = minima[ftr_i] + inter;
        means[i][ftr_i] = inter;
        break; }
    case 3: {
        inter = abs(maxima[ftr_i] - minima[ftr_i]);
        means[i][ftr_i] = minima[ftr_i] + (inter / 2);
        break;            }
    case 4: {
        inter = abs(maxima[ftr_i] - minima[ftr_i]);
        means[i][ftr_i] = maxima[ftr_i] - (inter / 2);
        break;            }
    case 5: {
```

```
                inter = abs(maxima[ftr_i] - minima[ftr_i]);
                means[i][ftr_i] = inter;
                break;}
            }
```

Slika 3.12. *Prikaz koda za određivanje vrijednosti centroida u metodi grupiranja algoritmom k srednjih vrijednosti za grupiranje VP.*

Za karakteristične veličine centroida, za koje će se izvršavati algoritam grupiranja k srednjih vrijednosti, odabiru se koordinate izvorišta VP te duljina i kut VP. Broj grupa u koje će se VP razvsrtati je k=3. Broj grupa je odabran kao najveći broj objekata u pokretu koji se inicijalno može detektirati. Tako dobivene grupe se potom međusobno uspoređuju po svakoj od četiriju karakterističnih veličina. Usporedba se izvršava tako da se za svaku grupu izračunava srednja vrijednost svake karakteristične veličine korištene u algoritmu grupiranja pomoću k srednjih vrijednosti te se provjerava postoji li dovoljna sličnost (empirijski određena veličina razlike za svaku karakterističnu veličinu na grupi testnih video signala) između srednjih vrijednosti karakterističnih veličina. Provjera sličnosti se izvršava kako bi se zaključilo je li rezultat algoritma grupiranja k srednjih vrijednosti dao točan rezultat, jer zbog pogrešno dobivenih VP u metodi pretrage VP postoji mogućnost da zbog manje udaljenosti vrijednosti centroida koji su inicijalno određeni funkcijom sa slike 3.12, dođe do stvaranja dvaju grupa centroida s bliskim vrijednostima karakterističnih veličina. Usporedba se izvršava za 3 para grupa: prva i druga, prva i treća te druga i treća grupa, prema kodu sa slike 3.13. Bitno je naglasiti da se kod usporedbe srednjih vrijednosti karakteristične veličine koordinata izvorišne točke VP koristi dvodimenzionalna Euklidska udaljenost.

```
uint8_t checkSimilarity(float* meanFirst, float* meanSecond) {//mean je u ovome UC-u
složen kao [0] - x ishodista, [1] - y ishodissta, [2] - duljina, [3] - kut

	uint8_t angleSim = 0, lengthSim = 0, locationSim = 0;
	int angleDis = CAST_ANGLE_DIS(abs(meanFirst[3] - meanSecond[3]) % 360);
	float lengthDis = abs(meanFirst[2] - meanSecond[2]);
	float locationDis = helperEuclidDis(meanFirst[0], meanFirst[1], meanSecond[0],
meanSecond[1]);
	if (angleDis < 40) {
		angleSim = 1;
	}
	if (lengthDis < 10) {
		lengthSim = 1;
	}
	if (locationDis <0.25*WIDTH) {
		locationSim = 1;
	}
	return (uint8_t)(angleSim * lengthSim* locationSim);
}
```

Slika 3.13. *Kod za usporedbu sličnosti srednjih vrijednosti grupa u metodi grupiranja VP.*

Granice sličnosti koje se nalaze u argumentima „*if*“ izjava predstavljaju određena ograničenja predložene metode. Kao najveća razlika između srednje vrijednosti karakteristične veličine kuta grupa koje se uspoređuju, odabrano je 40°, za razliku karakteristične veličine duljine 10, a za dvodimenzionalnu prostornu udaljenost srednje vrijednosti izvorišne točke grupe četvrtina širine okvira. Granice su odabrane empirijski te iako razlika kuta od 40° može predstavljati veliku razliku, zbog raspršenih vrijednosti kutova VP unutar grupe (raspršenih zbog pogreške u metodi pretrage VP) pokazalo se da je zadana razlika zadovoljavajuća. Isto vrijedi i za srednju vrijednost karakteristične veličine duljine. Najveće ograničenje je postavljena granica za dvodimenzionalnu udaljenost srednje vrijednosti izvorišne točke jer u slučaju da se na okviru kreće objekt koji zauzima područje okvira veće od četvrtine širine okvira, te ukoliko su pronađeni VP za takav objekt razvrstani u dvije grupe, neće se otkriti sličnost. Kada se odredi postoji li sličnost između grupa, slične grupe postaju ista grupa i ukupan broj grupa se umanjuje. Nakon provjere sličnosti i potencijalne promjene broja grupa, provjerava se raspršenost karakteristične veličine kuta VP u svakoj grupi pomoću mjere raspršenosti koja se izračunava prema kodu sa slike 3.14. U izračunu mjere raspršenosti karakteristične veličine kuta, zbraja se udaljenost između srednje vrijednosti karakteristične veličine kuta grupe i kuta pojedinog VP u grupi. Dobivena suma razlika dijeli se s maksimalnom mogućom vrijednošću razlika za tu grupu i množi se sa 100 kako bi se raspršenost izrazila u postotku. Ukoliko je raspršenost veća od 35% za tu grupu, ponovno se provodi algoritam grupiranja pomoću k srednjih vrijednosti za k=2 grupe. Dobivena granica je također određena empirijski na testnom skupu video signala. Svaka grupa za koju se ponovno provodi grupiranje algoritmom k srednjih vrijednosti, povećava ukupan broj grupa za 1. Pripadnost VP grupi tako ostaje zapisana kao konačna u jednodimenzionalnom polju cijelih brojeva čije su vrijednosti elemenata u području [0, kukupno-1].

```
float getAngleDeviation(float mean, int16_t* items, int clusterSize)
{
	int i;
	float sum = 0;
	for (i = 0; i < clusterSize; i++) {
		sum += CAST_ANGLE_DIS(abs(mean - items[i]) % 360);
	}
	return (sum/(clusterSize*180))*100;
}
```

Slika 3.14. *Kod za izračun mjere raspršenosti karakteristične veličine kuta unutar grupe u algoritmu grupiranja VP.*

3.5.3. Iscrtavanje grupiranih vektora pokreta na okvir video signala

Crtanje VP na osobnom računalu realizirano je korištenjem Python programskog jezika te biblioteka „*cv2*“ [14] i „*numpy*“ [15]. Python skripta najprije se učitava dvije binarne datoteke dobivene iz rješenja u MVS-u. Prva datoteka sadrži vektore pokreta, a druga informacije o grupama u koje su vektori raspodijeljeni. Uz navedeno učitava se okvir na koji će se iscrtavati vektori pokreta u trodimenzionalno „numpy“ polje podataka (svaka dimenzija jedan kanal RGB boje). Kada su sve datoteke učitane, iterira se kroz polje vektora pokreta te se prema podatcima iz datoteke s informacijama o grupama određuje kojom će bojom vektor biti iscrtan. Svaka boja predstavlja jednu grupu. Funkciji iz biblioteke cv2 za crtanje po „numpy“ polju podataka , predaju se vrhovi vektora i boja kao što je prikazano na slici 3.15.

```
cv2.arrowedLine(png,(pointFrom[0],pointFrom[1]), (pointTo[0],pointTo[1]),boje[bel
ongsTo[bcount]], 2)
```

Slika 3.15. *Kod za crtanje VP na polje „png“ koje predstavlja okvir.*

Rezultat koji generira skripta prikazan je na slici 3.16., a kod skripte nalazi se u elektroničkom prilogu P.3.2.

Slika 3.16. *Prikaz iscrtanih grupiranih VP rješenja implementiranog na osobnom računalu.*

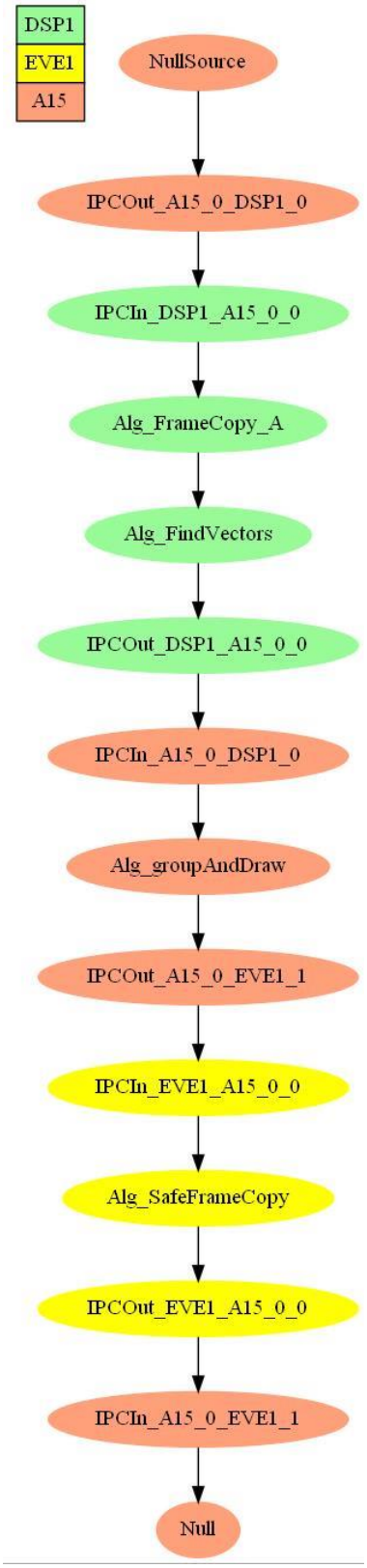
3.6. Implementacija rješenja na ADAS realnu platformu

Kao što je već ranije spomenuto, rješenje za pronalazak i grupiranje VP najprije je kreirano na osobnom računalu, a tek ona implementirano na realnu ADAS platformu, kako bi se olakšao razvoj rješenja. Razvoj rješenja na realnoj ploči je zahtjevan zbog njenih ograničenja, kao što su ponovna izrada izvršnih datoteka nakon svake promjene koda. Koncept rješenja, kao i sve metode korištene za proračune kod pronalaska VP, ostaju isti kao i kod rješenja na osobnom računalu. Pod konceptom rješenja smatra se osnovni tok izvođenja programa za određivanje VP i njihovo grupiranje.

3.6.1. Opis implementacije rješenja na realnoj ADAS platformi

Rješenje je realizirano u Vision SDK okruženju za rad na ADAS platformama, konkretno za rad na Alpha ploči. Dijagram toka implementiranog osnovnog rješenja na Alpha ploči složeno je u lanac (engl. *chain*, lanac) veza prikazan na slici 3.17. Narančasta boja na dijagramu pokazuje da se veza izvršava na procesoru A15, zelena da se izvršava na procesoru DSP1, a žuta da se izvršava na procesoru EVE1. Osnovno rješenje predstavlja korisnički slučaj u kojemu se video signal rezolucije 1280x720 elemenata slike, čiji su okviri u memoriji zapisani u formatu boje YUV420SP_UV, šalje na Alpha ploču s osobnog računala putem Ethernet sučelja te se za svaki okvir izvršava pronalazak (na jednome procesoru), grupiranje i iscrtavanje grupiranih VP na okvir, a potom se iscrtani okvir šalje na osobno računalo putem Ethernet sučelja.

Okvir dolazi s Ethernet sučelja koristeći vezu *NullSrc*. Za navedenu vezu definirano je da mora primati okvir formata YUV420SP_UV i da je rezolucije 1280x720 elemenata slike. Sljedeća veza je *Alg_FrameCopy_A* algoritam koji kopira ulazni spremnik video okvira u izlazni te alocira memoriju za metapodatke. Tako prošireni sistemski spremnik video okvira prosljeđuje se dalje. Nakon toga slijedi korisnički definirana veza predstavljena algoritmom *Alg_FindVectors*. U *create* metodi ove veze nalazi se poziv funkcija za alociranje memorije za sve potrebne varijable za pronalazak VP. Najveći spremnici koji se alociraju su spremnici za komponente svjetline trenutnog i prethodnog okvira. U *process* metodi ove veze nalaze se pozivi metoda za izdvajanje komponente svjetline iz okvira i spremanje u spremnik trenutnog okvira. Nakon toga slijedi poziv funkcije za pronalazak VP. VP potom se spremaju u spremnik metapodataka ulazno-izlaznog sistemskog spremnika. Metoda *control* omogućava izbor algoritma za pretragu: EBMA, TSS-UL i PR&AD-TSS.



Slika 3.17. *Prikaz korisnički definiranog lanca za osnovno rješenje pronalaska i grupiranja VP na ADAS Alpha ploči.*

Nakon veze *Alg_FindVectors* slijedi druga korisnički definirana veza *Alg_groupAndDraw*, koja služi za svrstavanje VP u grupe te iscrtavanje VP na trenutni okvir. U *create* metodi zauzimaju se spremnici za rezultate i međurezultate algoritma za grupiranje k srednjim vrijednostima. Veza se sastoji od dva poziva *proces* metode. U prvoj *process* metodi poziva se funkcija za svrstavanje VP po grupama, čiji je tijek opisan u podpoglavlju s rješenjem na osobnom računalu. Kada su VP razvrstani, informacija o grupi kojoj VP pripada te broj preostalih VP, sprema se u lokalni spremnik *belongsTo* koji će se nadalje koristiti za iscrtavanje vektora na trenutni video okvir. Potom slijedi druga metoda *process* koja koristi biblioteku *draw.h* napisanu za Vision SDK okruženje, koja se inicijalizira informacijama iz sistemskog spremnika podataka koristeći format boje, visinu, širinu okvira te širinu jednog reda okvira slike koja se razlikuje ovisno o formatu boje. Ukoliko je okvir zapisan u formatu boje YUV422_YUYV, dva elementa slike zapisana su u memoriji kao: komponenta svjetline Y_1 prvog elementa slike, komponenta boje U koju dijele dva elementa slike, komponenta svjetline Y_2 drugog elementa slike te komponenta boje V koju dijele oba elementa slike. Prema navedenom, jedan red video okvira u memoriji se zapisuje na *„(širina okvira × 1) + (širina okvira × suma broja U i V komponenti)*“ bajtova. U slučaju YUV420SP_UV formata boje, posložen u memoriji na način da se prvo zapisuju sve komponente svjetline Y za svaki elemenata slike te zatim naizmjence U i V komponente boje, od kojih svaki par dviju komponenti opisuje boju za 4 susjedna elementa slike, jedan red u memoriji zauzima „*pola širine okvira*“ bajtova jer će na taj način jedan red komponenti boje odgovarati jednom redu komponente svjetline.

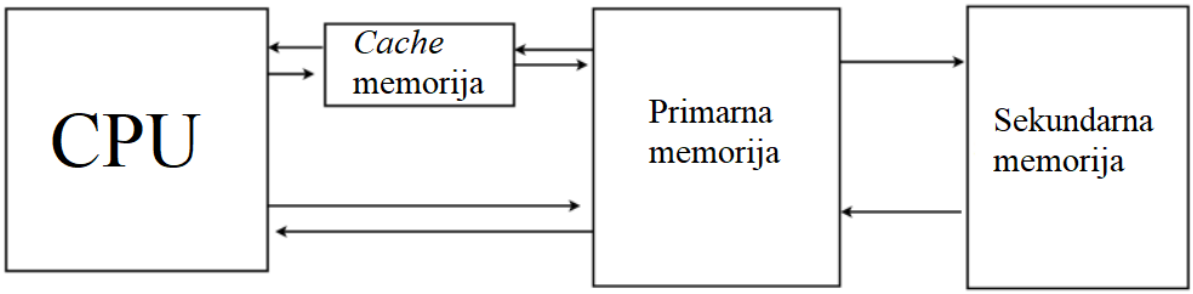
Nakon inicijalizacije slijedi analiza VP u kojima se za svaki VP poziva funkcija iz *draw.h* biblioteke, koja na trenutni video okvir iscrtava VP prema parametrima iz spremnika za taj VP i odabire boju vektora prema pripadnosti grupi.

Na kraju lanca dolazi veza *Null* koja šalje video okvir s iscrtanim VP na Ethernet sučelje. Primjer podataka koji dolazi Ethernet sučeljem na računalo prikazan je na slici 3.18.

Slika 3.18. *Prikaz rezultata koji se s Ethernet sučelja Alpha ploče prima na računalu (svaka boja predstavlja jednu grupu VP grupiranih u vezi Alg_groupAndDraw).*

3.6.2. Algoritamska optimizacija za predloženo rješenje

Nakon implementacije rješenja na Alpha ploču i postizanja funkcionalnosti, pristupilo se algoritamskoj optimizaciji, odnosno optimizaciji samog koda, kako bi se ubrzalo njegovo izvođenje. Algoritamskom optimizacijom mijenjaju se dijelovi koda, gdje se kod preoblikuje na osnovu arhitekture procesora i memorije, s ciljem poboljšanja brzine izvođenja. Naime, funkcije tijekom svoga izvođenja koriste lokalne, globalne i dinamički alocirane varijable. Kad funkcija pozove čitanje vrijednosti varijable, iste se čitaju iz primarne i sekundarne memorije sustava. Takve memorije su fizički udaljenije od procesora i potrebno je određeno vrijeme da im se pristupi. Kako bi se ubrzao proces čitanja, uz sami procesor dodaje se memorija niskog kapaciteta i velike brzine (engl. *cache*) ostvarene malom udaljenošću od procesora, prikazano na slici 3.19. Svakim zahtjevom za čitanjem vrijednosti s neke adrese iz memorije sustava, vrijednost sa zatražene adrese te vrijednosti s adresa u okolici zatražene adrese, učitavaju se u *cache* memoriju [16].



Slika 3.19. *Prikaz arhitekture procesorskog pristupa različitim tipovima memorija u računalnom sustavu (udaljenost memorije od CPU-a proporcionalna je brzini pristupa – „bliže CPU“ = „brži pristup memoriji“).*

S obzirom da je kapacitet *cache* memorije ograničen, broj učitanih vrijednosti sa susjednih adresa ovisi o tipu podatka na toj adresi, odnosno njegovoj veličini u memoriji. U osnovnom rješenju kao mjera sličnosti korištena je MSE mjera, dana formulom (2-2). MSE mjera za makroblok veličine *M*x*M* radi *M*x*M* kvadriranja i sprema ih u podatak tipa *float*. Kvadriranje je složeniji postupak od običnog zbrajanja ili oduzimanja i stoga se mjera MSE u optimiziranom rješenju zamjenjuje mjerom razlike SAE, koja je dana formulom (2-1) te se postiže optimizacija zbog promjene tipa podatka *float* u *integer*. Promjena načina izračunavanja mjere razlike ne uzrokuje promjenu rezultata pretrage VP jer je granica najveće mjere razlike, za koju se pronađeni VP pohranjuje, promijenjena ovisno o mjeri koja se koristi (9.0 za MSE odgovara broju „*9 × broj elemenata slike unutar makrobloka*“ za SAE).

Nakon navedene optimizacije korištena je metoda razgradnje petlje [17]. Metoda nalaže da se u jednoj iteraciji petlje iščitava što više susjednih elemenata polja podataka. Osim navedenog unutar jedne petlje, gdje god je bilo moguće, dodani su akumulatori suma. Zbog sposobnosti procesora da istovremeno izvršava više od jedne operacije zbrajanja ili oduzimanja [17], dodavanje susjednih vrijednosti u 4 različita akumulatora sume u jednoj iteraciji petlje, omogućeno je višestruko ubrzanje izvođenja metoda prikazanih na slikama 3.20. i 3.21.

```
int calculateSAE(uint8_t* currentFrame, uint8_t* prevFrame, Point currentMacroblockCoo,
Point prevMacroblockCoo) {
	int sum1=0, sum2=0, sum3=0, sum4=0;
	int x, y;
	for (y = 0; y < BLOCK_SIZE; y++) {
		for (x = 0; x < BLOCK_SIZE; x+=4) {
			sum1 = sum1 + (abs(prevFrame[((y + prevMacroblockCoo.y) * WIDTH) + (x
+ prevMacroblockCoo.x)] - currentFrame[((y + currentMacroblockCoo.y) * WIDTH) + (x +
currentMacroblockCoo.x)]));
```

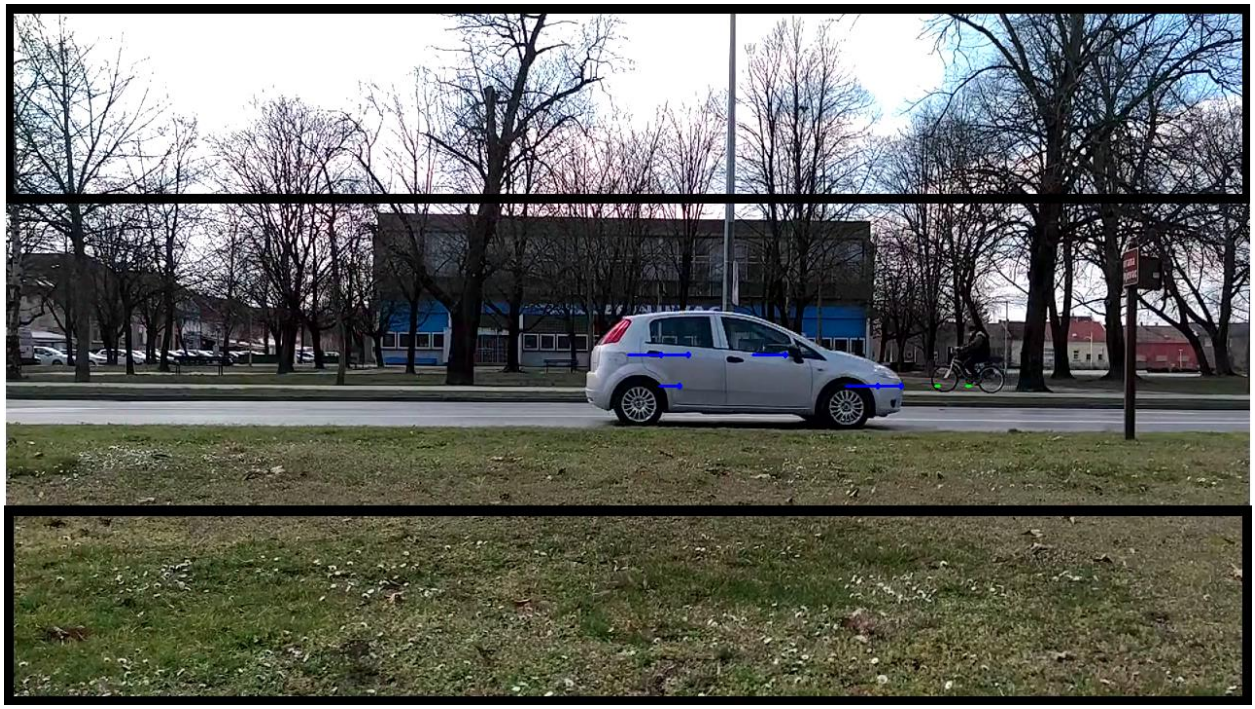
```
                        sum2 = sum2 + (abs(prevFrame[((y + prevMacroblockCoo.y) * WIDTH) + (x
+ prevMacroblockCoo.x+1)] - currentFrame[((y + currentMacroblockCoo.y) * WIDTH) + (x +
currentMacroblockCoo.x+1)]));
                        sum3 = sum3 + (abs(prevFrame[((y + prevMacroblockCoo.y) * WIDTH) + (x
+ prevMacroblockCoo.x+2)] - currentFrame[((y + currentMacroblockCoo.y) * WIDTH) + (x +
currentMacroblockCoo.x+2)]));
                        sum4 = sum4 + (abs(prevFrame[((y + prevMacroblockCoo.y) * WIDTH) + (x
+ prevMacroblockCoo.x+3)] - currentFrame[((y + currentMacroblockCoo.y) * WIDTH) + (x +
currentMacroblockCoo.x+3)]));
                }
        }

        return (sum1+sum2)+(sum3+sum4);
                                                }
```

Slika 3.20. *Kod funkcije za izračun mjere razlike dvaju makroblokova optimizan korištenjem metode razgradnje petlje (dodani akumulatori sum2, sum3 i sum4).*

```
int blockValueDeviation(UInt8* frame, Point upperLeftMacroblockCoo) {
        int sum = 0;
        int sum1 = 0;
        int sum2 = 0;
        int sum3 = 0;
        int y, x;
        for (y = 0; y < BLOCK_SIZE; y++) {
                for (x = 0; x < BLOCK_SIZE; x += 4) {
                        sum = sum + frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x +
upperLeftMacroblockCoo.x)];
                        sum1 = sum1 + frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x +
upperLeftMacroblockCoo.x + 1)];
                        sum2 = sum2 + frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x +
upperLeftMacroblockCoo.x + 2)];
                        sum3 = sum3 + frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x +
upperLeftMacroblockCoo.x + 3)];
                }
        }
        sum = (sum + sum1) + (sum2 + sum3);
        int mean = sum / (BLOCK_SIZE * BLOCK_SIZE);
        sum = 0;
        sum1 = 0;
        sum2 = 0;
        sum3 = 0;
        for (y = 0; y < BLOCK_SIZE; y++) {
                for (x = 0; x < BLOCK_SIZE; x += 4) {
                        sum += abs(mean - frame[((y + upperLeftMacroblockCoo.y) * WIDTH) + (x
+ upperLeftMacroblockCoo.x)]);
                        sum1 += abs(mean - frame[((y + upperLeftMacroblockCoo.y) * WIDTH) +
(x + upperLeftMacroblockCoo.x + 1)]);
                        sum2 += abs(mean - frame[((y + upperLeftMacroblockCoo.y) * WIDTH) +
(x + upperLeftMacroblockCoo.x + 2)]);
                        sum3 += abs(mean - frame[((y + upperLeftMacroblockCoo.y) * WIDTH) +
(x + upperLeftMacroblockCoo.x + 3)]);
                }
        }
        sum = (sum + sum1) + (sum2 + sum3);
        return sum;
                                                }
```

Slika 3.21. *Kod funkcije za izračuna uniformnosti makrobloka optimiziran metodom razgradnje petlje (dodani akumulatori sum1, sum2 i sum3).*

Kao završni element optimizacije iskorištena je priroda sadržaja video okvira. Naime sadržaj okvira u gornjoj i donjoj četvrtini svakog okvira kod video signala kamera iz automobila, smještenih na radnu ploču automobila, najčešće obuhvaća dio u kojem se objekti od interesa ne kreću (npr. nebo ili tlo, cesta), što je prikazano na slici 3.22. Učinci provedene optimizacije bit će vrednovani u četvrtom poglavlju rada.



Slika 3.22. *Primjer okvira s označenim (crni pravokutnici) područjem u kojemu nema pomaka relevantnih pokretnih objekata.*

3.6.3. Raspodjela rješenja na više procesora Alpha ploče (paralelizacija)

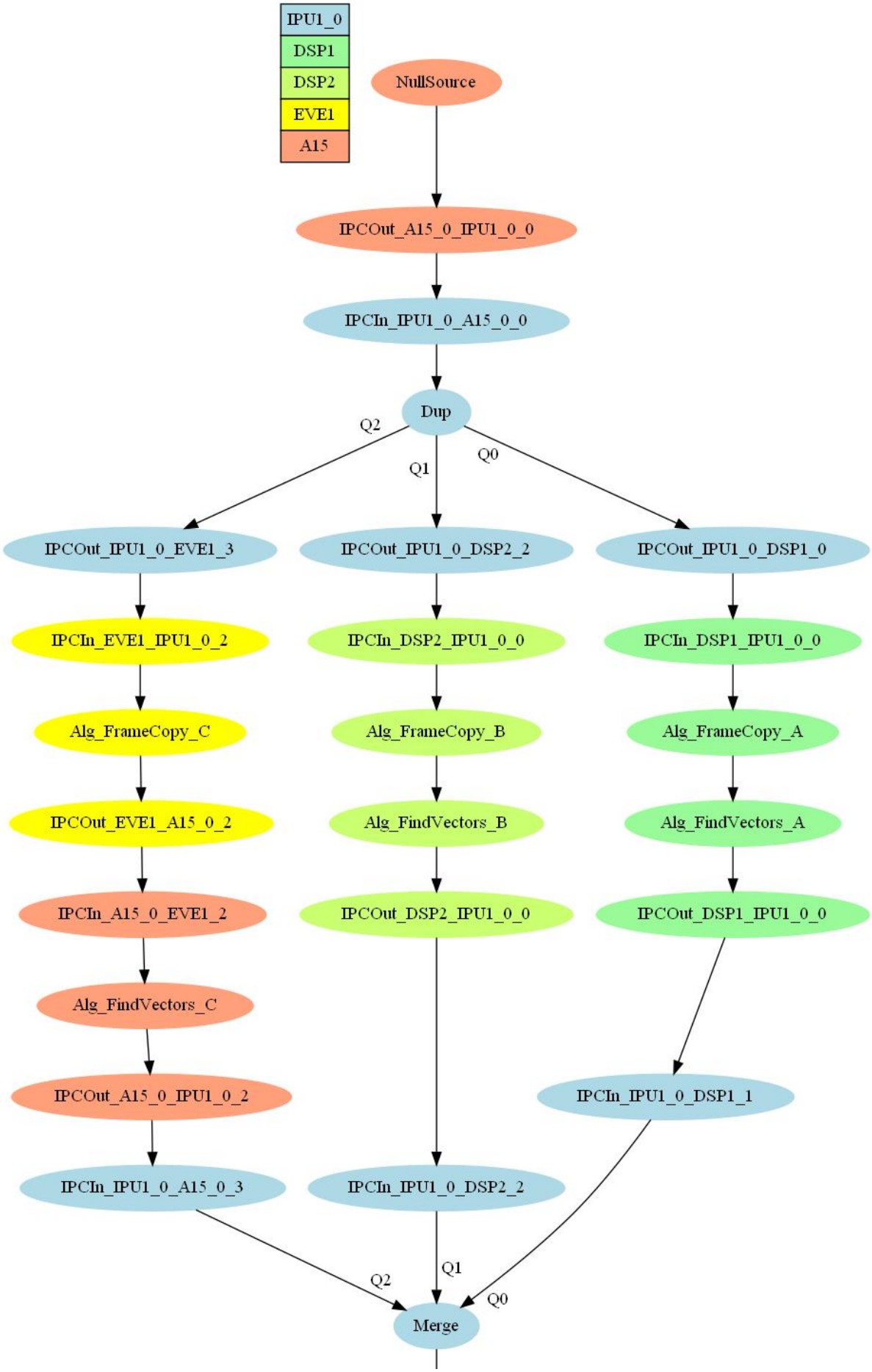
Proučavanjem dobivenog rezultata brzine izvođenja osnovnog rješenja na testnom skupu video signala, kod kojeg se pretraga VP izvršava u vezi *Alg_FindVectors* na procesoru DSP1, zaključeno je da će vrijeme izvršavanja u slučaju korištenja dva DSP procesora biti dvostruko kraće. Prema uzoru na osnovno rješenje, izrađeno je rješenje koje se izvodi na A15 procesoru. Jedina razlika u lancu rješenja, u usporedbi s osnovnim rješenjem na DSP1 procesoru, je oznaka procesora pored veze *Alg_FindVectors*, gdje u slučaju izvršavanja na procesoru A15 stoji oznaka „(A15)“ umjesto „(DSP1)“. Preliminarnim rezultatima brzine izvođenja utvrđeno je da procesor A15 kôd rješenja izvršava dvostruko sporije od DSP procesora. Razlog takvog vremena izvođenja je što optimizirano rješenje veze *Alg_FindVectors* ne sadrži velike tipove podataka (tipovi podataka veći od 4 bajta), stoga A15 procesor ne ostvaruje očekivano bolje rezultate koje bi u usporedbi s DSP procesorom imao u slučaju velikih tipova podataka korištenih u funkcijama

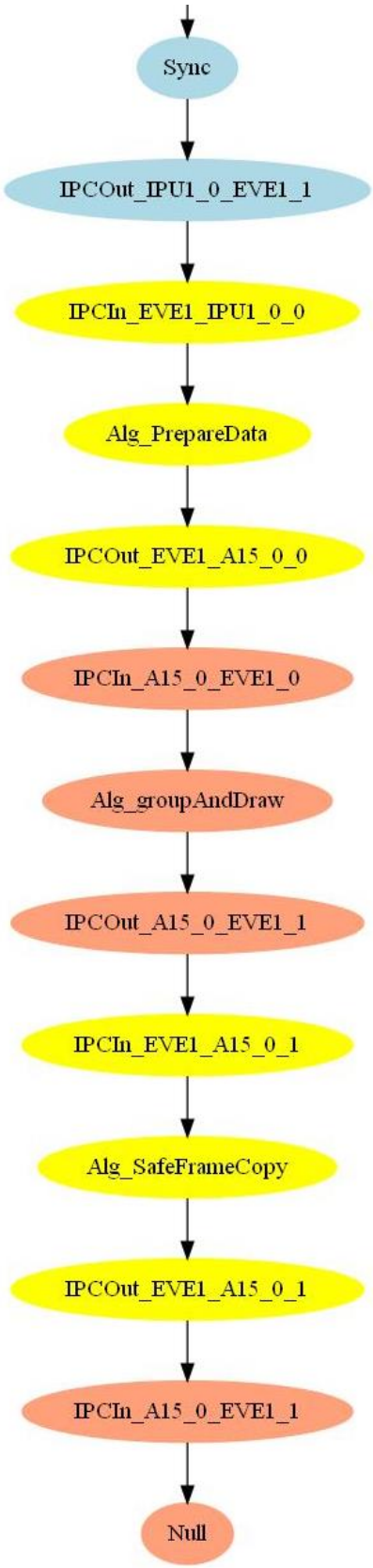
veze zbog puno veće *cache* memorije. Kako bi se iskoristile mogućnosti sklopovlja dostupnog na Alpha ploči i na taj način postiglo ubrzanje rada rješenja, potrebno je na smislen način raspodijeliti određene zadatke na više različitih procesora. Da bi se to učinilo, potrebno je dodati karakteristične veze u lanac rješenja za Alpha ploču. Nakon primanja video okvira potrebno je koristiti vezu *Dup* onoliko puta koliko se procesora paralelno koristi u rješenju. U završnom rješenju ovoga zadatka istovremenose koriste 3 procesora – 2xDSP (DSP1 i DSP2) te 1xA15. Sve algoritamski optimizirane veze iz osnovnog rješenja koriste se i u završnom rješenju. Unutar veze *Alg_FindVectors_X*, ovisno o procesoru koji je korišten, određuje se koje je područje pretrage za taj procesor. Primjer dodjeljivanja vrijednosti područja pretrage prikazano je slikom 3.23.

```
chains_bmaNetwork_SetFindVectorsAlgPrms(&pUcObj->Alg_FindVectors_APrm,
       0,
       200,
       512,
       520,
       pObj->chainsCfg);
chains_bmaNetwork_SetFindVectorsAlgPrms(&pUcObj->Alg_FindVectors_BPrm,
       512,
       200,
       1024,
       520,
       pObj->chainsCfg);
chains_bmaNetwork_SetFindVectorsAlgPrms(&pUcObj->Alg_FindVectors_CPrm,
       1024,
       200,
       1280,
       520,
                                        pObj->chainsCfg);
```

Slika 3.23. *Primjer koda koji definira područje pretrage za pojedine procesore kod paralelnog korištenja dvaju procesora DSP i jednog procesora A15.*

Izlaze korisnički definiranih veza *Alg_FindVectors_A, B i C* potrebno je povezati koristeći vezu *Merge*. *Merge* prima *n* ulaznih redova i spaja ih u 1 ulazni red s *n* kanala. Nakon veze *Merge* dodaje se *Sync* veza koja omogućava da redovi, koji dolaze u *Merge* vezu, pripadaju vremenski istome okviru. Iz *Sync* veze izlazi sistemski spremnik tipa kompozitnog spremnika. Potrebno je definirati novu vezu koja će iz jednog od kanala kompozitnog spremnika iščitati video okvir, a i iz svih sistemskih spremnika iščitati spremnik metapodataka i pohranit ga u metapodatak vezan za izlazni sistemski spremnik. Za navedeno je definirana veza Alg_*PrepareData* koja se pokreće na EVE1 procesoru. Navedeni procesor se koristi zbog pristupa biblioteci za izravan pristup memoriji kojom je omogućeno kopiranje video okvira u milisekundi iz ulaznog u izlazni spremnik.

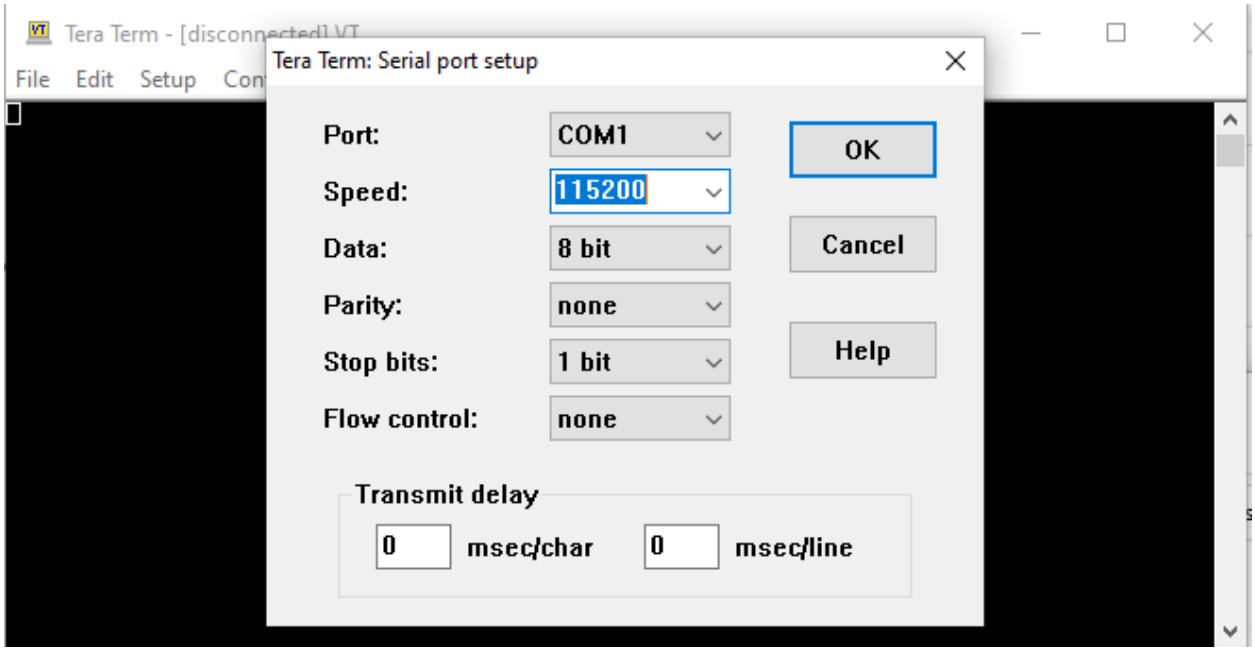

IPU1_0
DSP1
DSP2
EVE1
A15
NullSource
IPCOut_A15_0_IPU1_0_0
IPCIn_IPU1_0_A15_0_0
Dup
Q2
Q1
Q0
IPCOut_IPU1_0_EVE1_3
IPCOut_IPU1_0_DSP2_2
IPCOut_IPU1_0_DSP1_0
IPCIn_EVE1_IPU1_0_2
IPCIn_DSP2_IPU1_0_0
IPCIn_DSP1_IPU1_0_0
Alg_FrameCopy_C
Alg_FrameCopy_B
Alg_FrameCopy_A
IPCOut_EVE1_A15_0_2
Alg_FindVectors_B
Alg_FindVectors_A
IPCIn_A15_0_EVE1_2
IPCOut_DSP2_IPU1_0_0
IPCOut_DSP1_IPU1_0_0
Alg_FindVectors_C
IPCOut_A15_0_IPU1_0_2
IPCIn_IPU1_0_DSP1_1
IPCIn_IPU1_0_A15_0_3
IPCIn_IPU1_0_DSP2_2
Q2
Q1
Q0
Merge



Slika 3.24. *Prikaz korisnički definiranog lanca za paraleliziranu implementaciju završnog rješenja (završno rješenje) kod kojeg se paralelno koriste DSP1, DSP2 i jedan procesor A15.*

3.7. Upute za korištenje programskog rješenja na realnoj ADAS platformi

Za potrebe pokretanja i korištenja implementiranog rješenja na Alpha ploči, služi se alatima *network_tx*, *network_rx* za slanje i primanje video okvira na i sa ploče, te alat *Tera Term* [18]. USB kabel priključuje se iz alpha ploče u bilo koji od USB utora računala. Nakon priključivanja otvara se aplikacija „Tera Term“ te se USB utor postavlja na postavke prikazane slikom 3.25.



Slika 3.25. *Prikaz postavki sučelja USB-a za komunikaciju računala s Alpha pločom.*

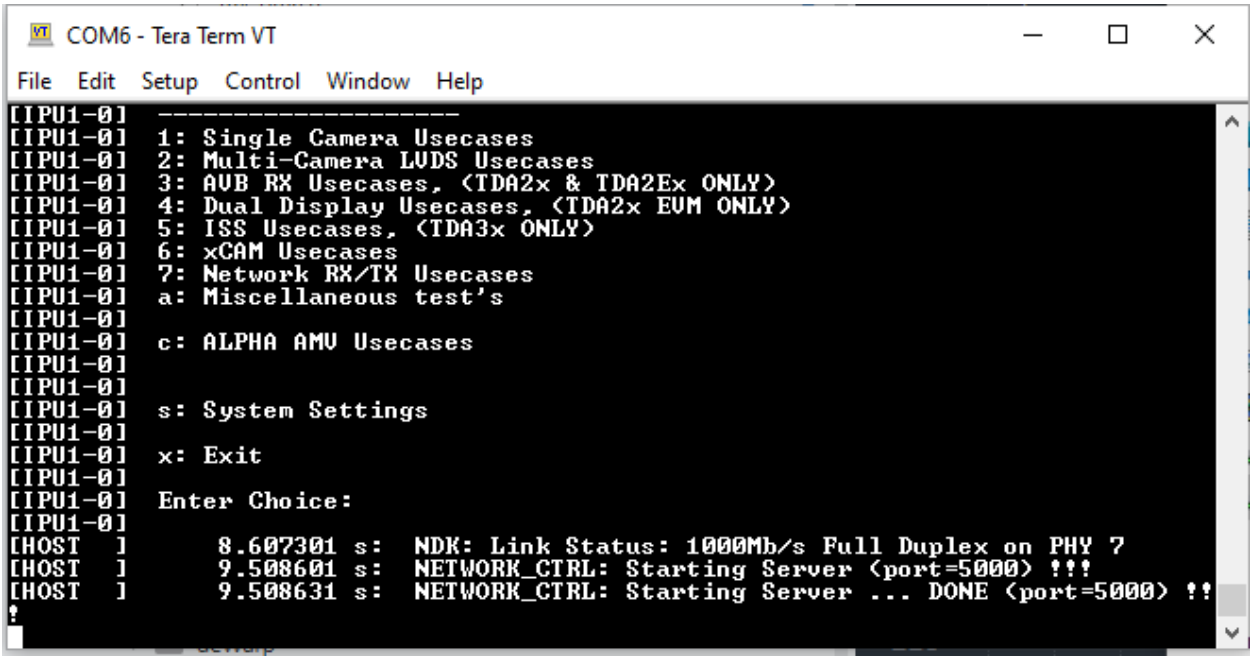
Kada se uključi napajanje ploče, u terminalu aplikacije *Tera term* otvara se izbornik prikazan na slici 3.26. u kojem je potrebno odabrati opciju „c“. Potom se otvara novi izbornik (slika 3.27.) za odabir korisničkog rješenja. Na ovome mjestu odabire se rješenje s nazivom „BMA over network“ (završno rješenje) tipkom 1. Kada se rješenje u potpunosti pokrene, korisnik može birati metodu kojom želi pretraživati VP, prikazano slikom 3.28. Na raspolaganju su metode EBMA, PR&AD-TSS i TSS-UL. Kako bi se podatci mogli primati s ploče, u „tools/network_tools/bin“ mapi Vision SDK okruženja, potrebno je pokrenuti terminal i naredbu za spremanje dolaznih video okvira. Naredba za spremanje podataka s ploče je:

„*network_rx --host_ip 192.168.10.1 --target_ip 192.168.10.2 --port 7000 --files putanja\output.yuv*“.

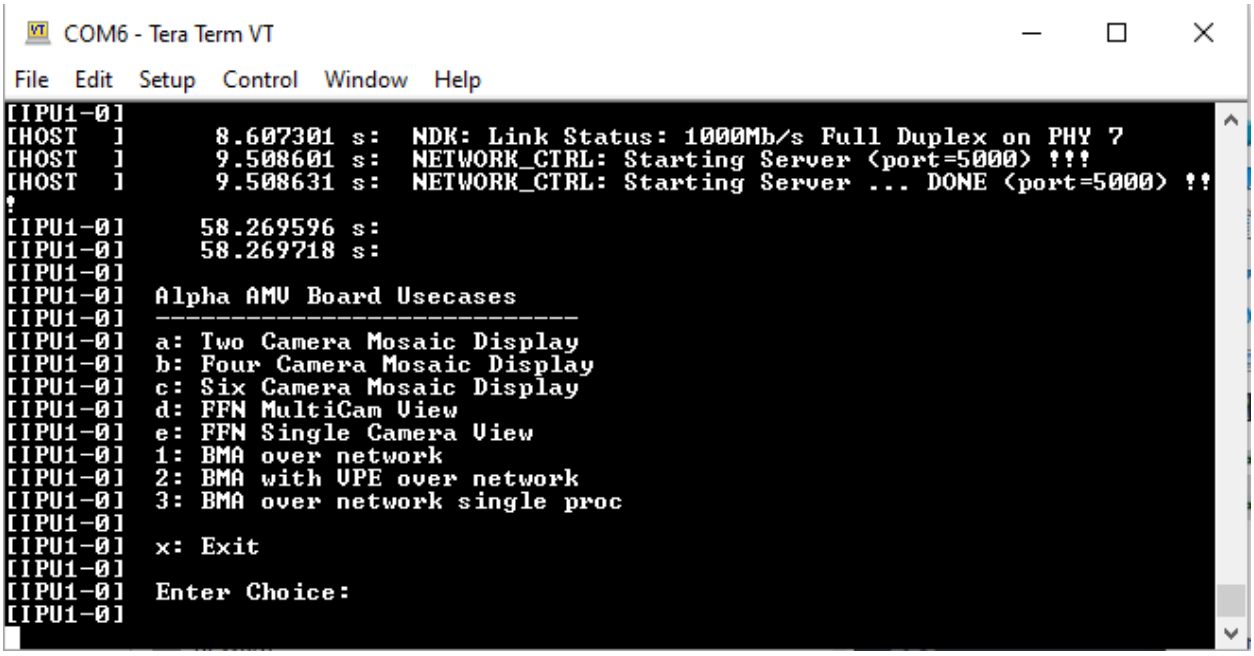
Za obradu podataka na ploči potrebno je prirediti datoteku s video okvirima u sirovom formatu s poduzorkovanom shemom YUV420SP_UV. Datoteka se na ploču šalje pokretanjem naredbe

„*network_tx --host_ip 192.168.10.1 --target_ip 192.168.10.2 --no_loop --files naziv.bin*“.

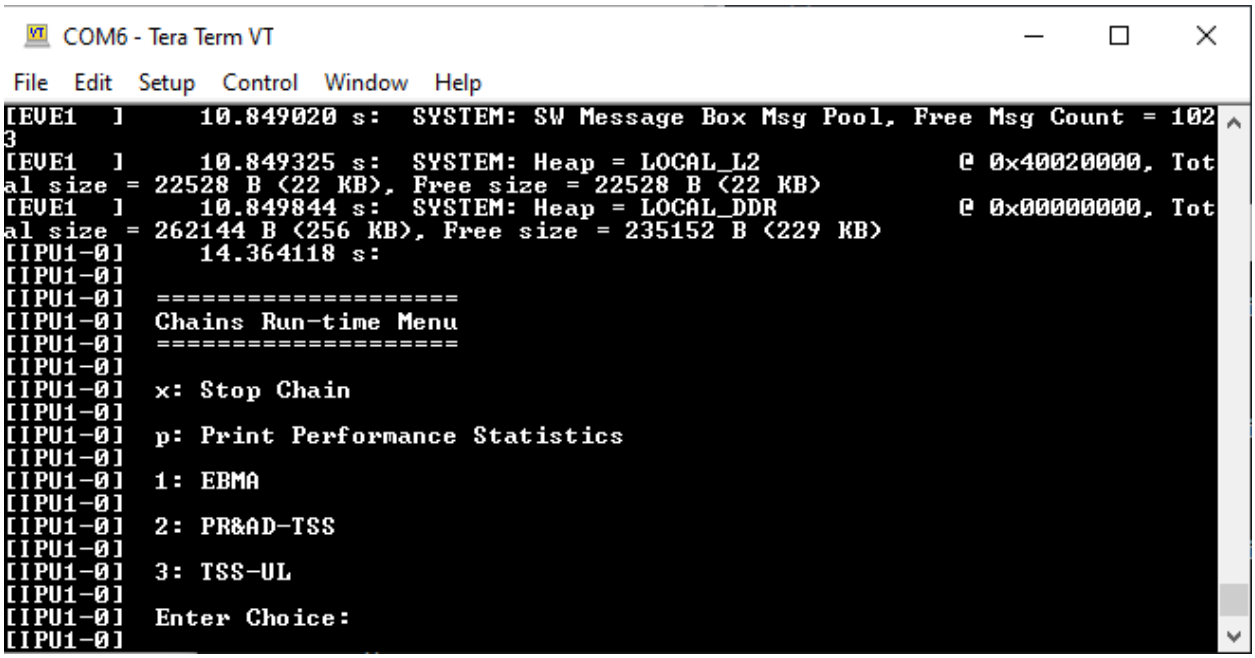
Rezultat koji dolazi s ploče vidljiv je na slici 3.18.



Slika 3.26. *Prikaz početnog sučelja nakon priključivanja Alpha ploče na napajanje.*



Slika 3.27. *Prikaz sučelja za izbor korisničkih rješenja na Alpha ploči.*



Slika 3.28. *Prikaz sučelja za izbor metode pronalaska VP za korisničkog rješenje pronalaska i grupiranja VP na Alpha ploči.*

4. TESTIRANJE RADA PREDLOŽENOG RJEŠENJA ZA PRORAČUN VEKTORA POKRETA I NJIHOVO GRUPIRANJE

Nakon implementacije rješenja potrebno je ispitati njegove performanse, odnosno vrednovati rezultate koje rješenje postiže u pogledu vremena izvođenja rješenja i točnosti (pronađenih VP te njihovog grupiranja). Tijekom kreiranja i implementacije rješenja korišten je manji broj parova okvira, no završno vrednovanje potrebno je provesti na većem skupu testnih okvira različitog sadržaja. U podpoglavlju 4.1. predstavit će se baza video signala korištenih za vrednovanje implementiranog rješenja. Opisat će se uvjeti u kojima su video signali prikupljeni, jesu li nastali video kamerom ili računalnim generiranjem, koji tip kamere je korišten, je li kamera u mirnom položaju ili se kreće, te će se predstaviti sadržaj okvira. Nakon predstavljanja baze signala, u podpoglavlju 4.2., opisat će se način provođenja testiranja točnosti metoda za pretragu VP te implementirane metode za grupiranje VP. Potom će se u podpoglavlju 4.3. predstaviti način provođenja i rezultati testiranja na cjelokupnom skupu testnih nizova okvira, uz vrednovanje brzine izvođenja pojedine metode za pronalazak VP na PC-u s osvrtom na algoritamsku optimizaciju, brzine izvođenja rješenja implementiranog na Alpha ploči na pojedinom procesoru (DSP1 ili A15) te završnom rješenju s 3 procesora korištena u paraleli. Provedeno testiranje predložene metode za grupiranje VP bit će vrednovano u podpoglavlju 4.4. U posljednjem podpoglavlju (4.5.) dat će se kritički osvrt na rezultate točnosti i brzinu izvođenja implementiranog rješenja.

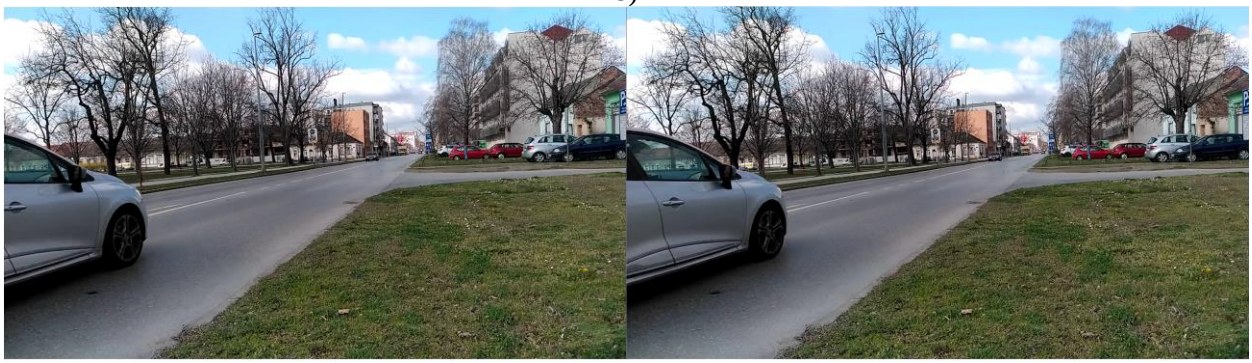
4.1. Opis baze signala korištenih za testiranje implementiranog rješenja

Testne sekvence se mogu podijeliti u dvije osnovne grupe – one prikupljene video kamerom te računalno generirane sekvence. Za prikupljanje testnih sekvenci korišten je pametni telefon Xiaomi Mi A3. Sekvence su snimljene prednjom kamerom od 48 MP, u rezoluciji 1920x1080 elemenata slike i uz 60 okvira po sekundi (engl. *Frames per second* - FPS). Videozapisi su spremljeni u MP4 formatu videa, koji je zatim poduzorkovan sa 60 na 30 FPS. Nakon poduzorkovanja, video signal je skaliran na rezoluciju 1280x720 elemenata slike uz korištenje bilinearne interpolacijom pomoću alata FFmpeg [19] te su svi okviri video signala pretvoreni u sirovi format slike YUV420SP_UV. Snimljene su 4 različite sekvence. Prve tri sekvence snimljene su na način da je kamera bila postavljena na stativ, neposredno uz prometnu cestu, iz triju različitih kutova - s pogledom na lijevo, s pogledom okomito na cestu te s pogledom na desno. Iz 3 prethodno navedene snimljene sekvence izdvojeno je ukupno 20 parova okvira (jedan par su dva susjedna okvira iz sekvence uz 30 FPS, 40 okvira) koji sadrže karakteristične situacije

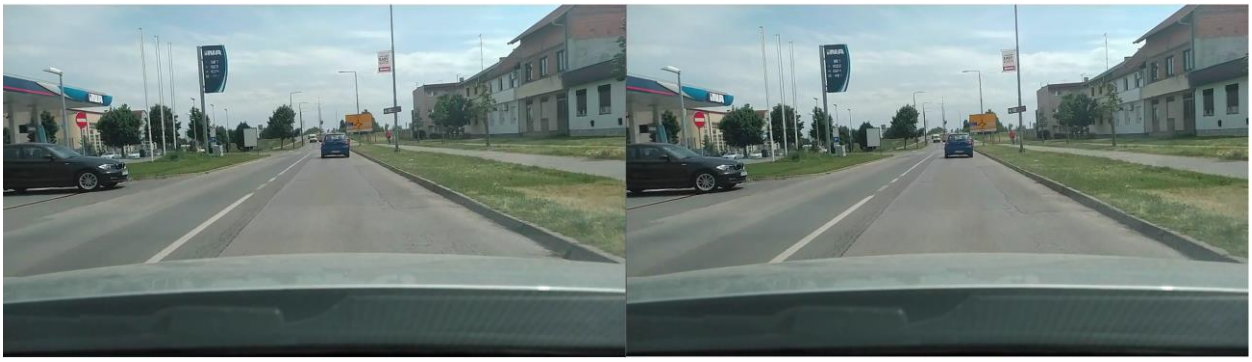
u prometu, poput jednog automobila u kadru na cesti, dva automobila u kadru na cesti, koji se mogu kretati jedan iza drugoga ili jedan prema drugom i sl. Izdvojeni skup okvira predstavlja prvi testni niz (u nastavku nazvan *mirna kamera*). Četvrta snimljena sekvenca je nastala u vožnji automobilom, a kamera je bila postavljena na upravljačku ploču s pogledom prema naprijed. Iz tako snimljene sekvence, također je izdvojeno 20 parova okvira karakterističnih situacija u prometu, koji predstavljaju drugi testni niz (u nastavku nazvan *pokretna kamera*). Primjeri po jednog para okvira iz svake od četiriju snimljenih sekvenci prikazani su na slici 4.1. Kompletne video sekvence, kao i svi parovi okvira izdvojeni iz njih za potrebe testiranja nalaze se u elektroničkom prilogu P.4.1. priloženom uz ovaj rad.

a)

b)



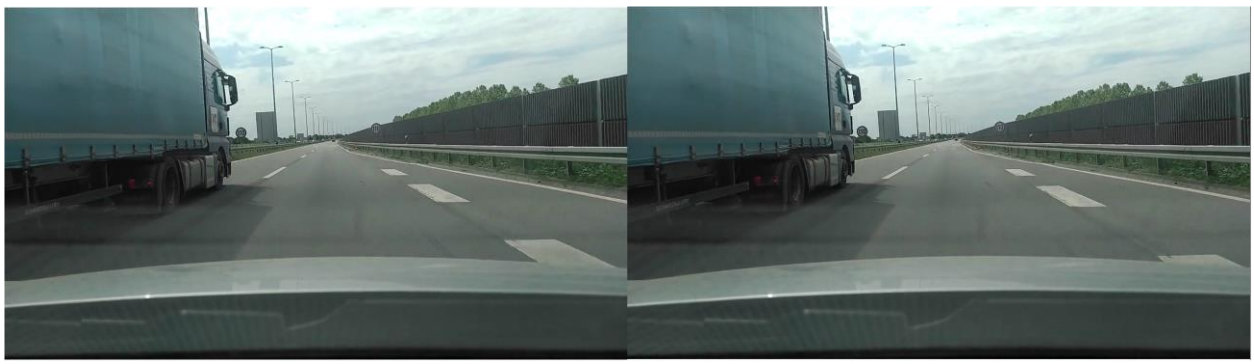
c)



d)

Slika 4.1. *Primjeri parova okvira iz snimljenih video sekvenci rezolucije 1280x720 elemenata slike uz 30 FPS, snimljenih a) pored ceste s pogledom na lijevo (kamera na stativu), b) pored ceste s pogledom okomito na cestu (kamera postavljena na stativ), c) pored ceste s pogledom na desno (kamera postavljena na stativ), d) u vožnji automobilom, gdje je kamera postavljena na upravljačku ploču.*

Nadalje, za potrebe testiranja slijeda okvira (tj. kada se želi testirati algoritam na više uzastopnih okvira neke scene, a ne samo njih 2), izdvojena su 2 skupa od 40 uzastopnih okvira iz videozapisa snimljenog sa upravljačke ploče automobila. Prvi niz sadrži kamion koji ulazi u kadar te se udaljava od kamere, a drugi niz od 40 okvira sadrži automobil u istoj situaciji. Izdvojeni nizovi predstavljaju treći i četvrti testni niz s nazivima *slijedni kamion* i *slijedni auto.* Oba izdvojena niza okvira također se nalaze u elektroničkom prilogu P.4.1. Primjer po jednog para okvira iz niza *slijedni kamion* i *slijedni auto* prikazani su na slici 4.2., kako bi se dobio dojam o sadržaju niza okvira.

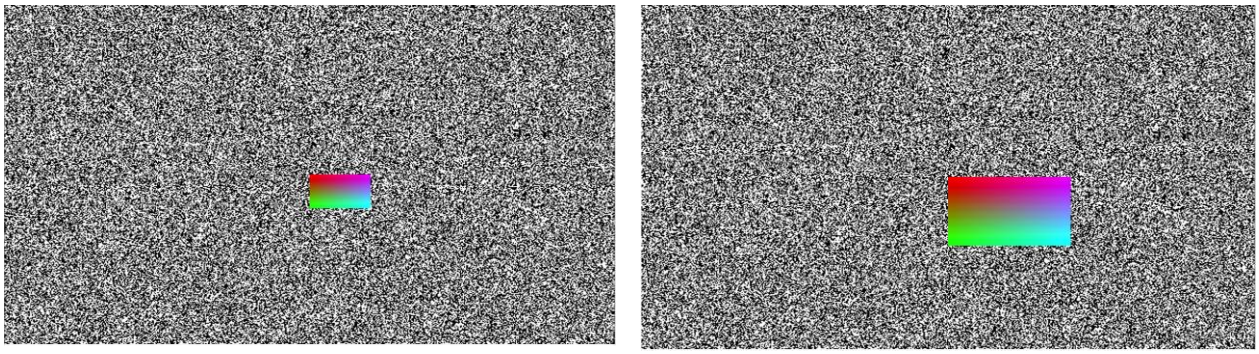


a)

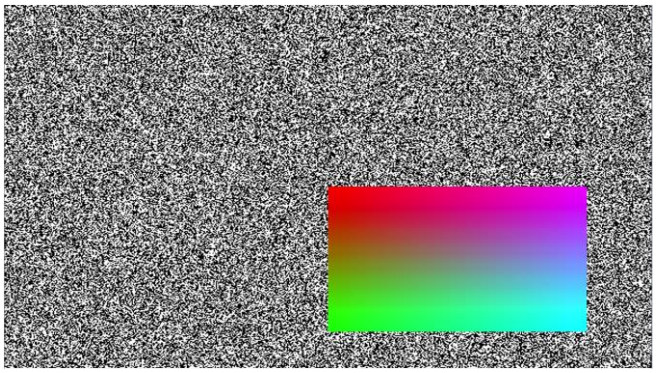
b)

Slika 4.2. *Primjer parova okvira rezolucije 1280x720 elemenata slike uz 30 FPS iz testnog niza a) slijedni auto i b) slijedni kamion*

Osim svih dosad nabrojanih sekvenci stvarnog sadržaja iz prometa snimljenih kamerom, za potrebe testiranja ispravnosti, tj. točnosti rada rješenja, stvorena je i tzv. umjetna sekvenca okvira koristeći Python programski jezik. Na biplanarno platno rezolucije 1280x720 elemenata slike, s nasumičnom raspodjelom crnih i bijelih elemenata slike, postavljen je objekt određene veličine. Objekt je predstavljen pravokutnikom podijeljenim na 4 manja pravokutnika (crvene, ljubičaste, zelene i svijetlo plave boje) s gradijentnim prijelazima između susjednih manjih pravokutnika. Kako bi se provjerio utjecaj veličine uniformnog područja na okviru tijekom pronalaska VP, navedena sekvenca sastoji se od 3 objekta različite veličine - 128x72, 256x144 i 512x288 elemenata slike, prikazanih na slici 4.3.



a) b)

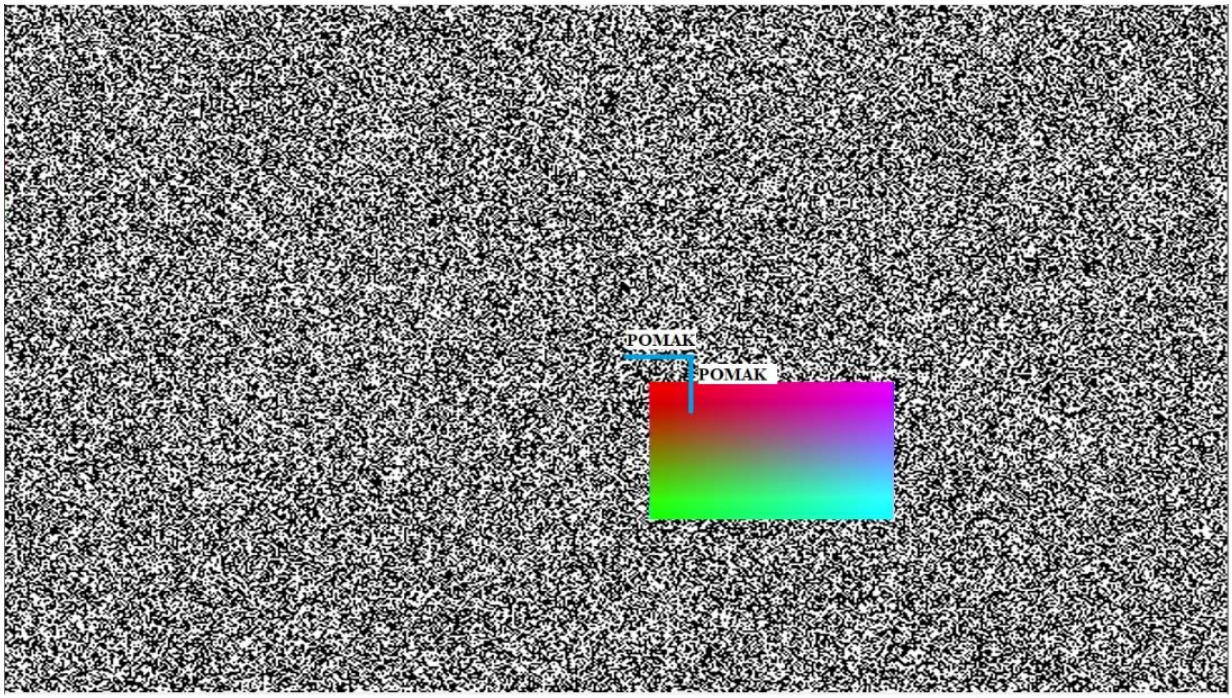


c)

Slika 4.3. *Primjer objekta na biplanarnom platnu rezolucije 1280x720 elemenata slike. Objekt veličine a) 128x72 elemenata slike, b) 256x144 elemenata slike i c) 512x288 elemenata slike.*

Za svaku veličinu objekta generirano je „$8 \times 3 = 24$“ para okvira (8 smjerova kretanja – gore, dolje, lijevo, desno, gore desno, gore lijevo, dolje desno, dolje lijevo – te 3 duljine pomaka po svakoj koordinati – 15, 25 i 35 elemenata slike) što je sveukupno „$3 \times 3 \times 8 = 72$“ (za svaku veličinu objekta – njih 3, svaku duljinu pomaka – njih 3, svaki smjer kretanja – njih 8) para okvira u jednom nizu. Parovi okvira, za objekt pojedine veličine i pojedinu duljinu pomaka, posloženi su tako da se u slijedu od 8 parova objekt iste veličine kreće između dvaju susjednih okvira za isti pomak, ali u 8 definiranih smjerova, zatim se povećava pomak i ponavlja kretanje u 8 smjerova. Nakon svih definiranih pomaka objekt se povećava te se proces opisan u prethodnoj rečenici ponavlja te tako za sve tri veličine objekta. U prvom okviru iz para okvira, gornja lijeva točka objekta nalazi se u središtu okvira na koordinatama (640,360), a u drugom okviru nalazi se na lokaciji definiranoj trenutnim pomakom i smjerom kretanja. Pomicanje u više smjerova omogućava testiranje osjetljivosti za pronalazak VP na promjenu smjera kretanja objekta u okviru. Ovakva sekvenca, nazvana *dummy*, omogućava testiranje točnosti pronađenog VP, jer se zbog definiranih pomaka tijekom stvaranja niza okvira u Python-u točno zna za koliko se elemenata slike objekt pomaknuo i u kojem smjeru se pomaknuo. Primjer pomaka između okvira prikazan je na slici 4.3.

Svih 5 nizova (*dummy, mirna kamera, pokretna kamera, slijedni kamion, slijedni auto*) okvira umanjeni su i na rezoluciju 640x360 elemenata slike koristeći FFmpeg uz bilineranu interpolaciju, kako bi se vrednovale performanse predloženih metoda na manjim okvirima i manjim makroblokovima (8x8 i 16x16 elemenata slike), odnosno kako bi se provjerila točnost uz očekivano četverostruko umanjenje brzine izvođenja (jer je broj elemenata slike umanjen dvaput po horizontali i dvaput po vertikali).



Slika 4.3. *Biplanarno platno rezolucije 1280x720 elemenata slike na kojemu se objekt veličine 264x144 pomakne za 30 elemenata slike desno i 30 elemenata slike dolje.*

U nastavku poglavlja, u podpoglavlju 4.2., najprije će se vrednovati točnost svakog od 3 predložena algoritma za pretragu VP na *dummy* nizu, nakon čega će se vrednovati subjektivno procijenjena točnost algoritma za grupiranje VP na nizovima okvira iz prometa u podpoglavlju 4.3., a potom će se vrednovati brzina izvođenja implementiranih rješenja u podpoglavlju 4.4. Na kraju poglavlja dat će se kritički osvrt na rezultate testiranja implementiranog rješenja. Na sve testne sekvence iscrtani su pronađeni i grupirani VP te su priloženi u elektroničkom prilogu P.4.2.

4.2. Testiranje točnosti različitih metoda za pretragu vektora pokreta

Za svaku predloženu metodu za pretragu VP testirana je točnost koja se njome postiže. Kao ulaz u pojedinu metodu postavljen je *dummy* testni niz okvira koji je prethodno opisan. Po završetku pretrage VP su filtrirani po duljini na način da su izbačeni VP koji su kraći od 2 i dulji od 50 elemenata slike. Tako filtrirani VP spremaju se u binarnu datoteku za daljnju analizu. S obzirom da su video okviri kreirani programskim kodom, poznate su duljine te vrijednosti kutova VP za svaki pomak objekta. Duljina i kut svakog izračunatog VP uspoređena je s referentnim vrijednostima za taj okvir. Rezultati su predstavljeni tablicama točnosti za pojedinu veličinu objekta gdje se točnost izračunava formulom (4-1). Rezultat je postotak u rasponu [0, 100], gdje 0% označava da nema točno pronađenih VP, a 100% da su svi pronađeni VP točni.

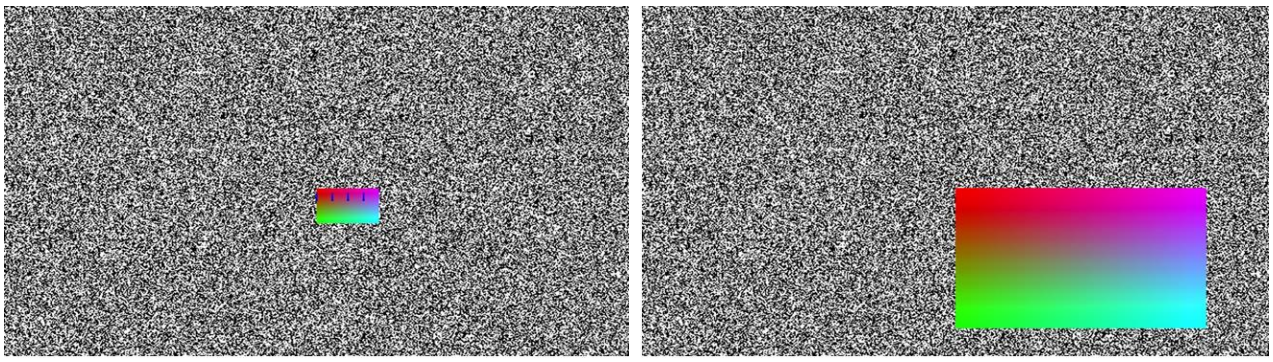
$$Točnost = \frac{Broj\ točnih\ VP}{Ukupan\ broj\ VP} * 100\% \quad (4\text{-}1)$$

Osim numerički točnih VP, u ovome će se podpoglavlju komentirati i subjektivno točni VP. Subjektivno točni VP su oni koji pokazuju ispravan smjer, ali s manjom greškom u kutu usmjerenosti VP i njegovoj duljini.

Točnost pronađenih VP testirana je na obje verzije *dummy* niza okvira, rezolucije 1280x720 elemenata slike uz makroblokove veličine 32x32 elementa slike te rezolucije 640x720 elemenata slike uz makroblokove 8x8 i 16x16 elemenata slike.

4.2.1. Točnost metode iscrpne pretrage (EBMA)

Rezultati testiranja *dummy* niza okvira rezolucije 1280x720 elemenata slike i makroblokovima veličine 32x32 elemenata slike pokazuju da broj pronađenih VP ovisi o veličini objekta koji se kreće. Veći objekt sadrži više makroblokova. Broj točno pronađenih VP prestaje rasti te se počinje smanjivati kod najvećeg objekta jer takav objekt sadrži više uniformnih makroblokova koji se izostavljaju tijekom pretrage, zbog izračunate mjere uniformnosti koja je za takve makroblokove veća od postavljene granice od 5.0. Primjer usporedbe uniformnih područja najmanjeg i najvećeg objekta prikazana je slikom 4.4. Na slici je vidljivo kako veći objekt veličine 512x288 elemenata slike na biplanarnom okviru ima veće uniformno područje unutar kojeg metoda pune pretrage ne pronalazi niti jedan VP i ima točnost od 0%, dok je za objekt veličine pretrage 128x72 točnost 100%. Kako je već i navedeno, do ovakvoga rezultata dolazi zbog izostavljanja područja makroblokova koji imaju mjeru uniformnosti veću od 5.0, što je u slučaju većega objekta prisutno unutar cijelog područja kojeg on zauzima na okviru.



a) b)

Slika 4.4. *Primjeri okvira s iscrtanim VP dobivenim pomoću EBMA metode na dummy nizu okvira rezolucije 1280x720 elemenata slike, uz makroblok 32x32 elementa slike i pomak od 15 elemenata slike a) gdje objekt zauzima prostor okvira od 128x72 elemenata slike, b) gdje objekt zauzima prostor okvira od 512x288 elemenata slike.*

Osim spomenutog problema uniformnog područja, zbog specifičnog položaja objekta na okviru (koordinate 640,360) makroblok od 32x32 elementa slike s trenutnog okvira nikada ne zahvaća sadržaj okvira u kojemu je dio makrobloka sadržaj biplanarnog platna, a da je drugi dio makrobloka sadržaj objekta. Kada bi postojao takav slučaj, metoda za pronalazak VP bi za takav makroblok izračunala mjeru uniformnosti veću od zadane granice te bi se za njega započela pretraga VP, gdje bi dobiveni VP bili raspoređeni na rubovima objekta.

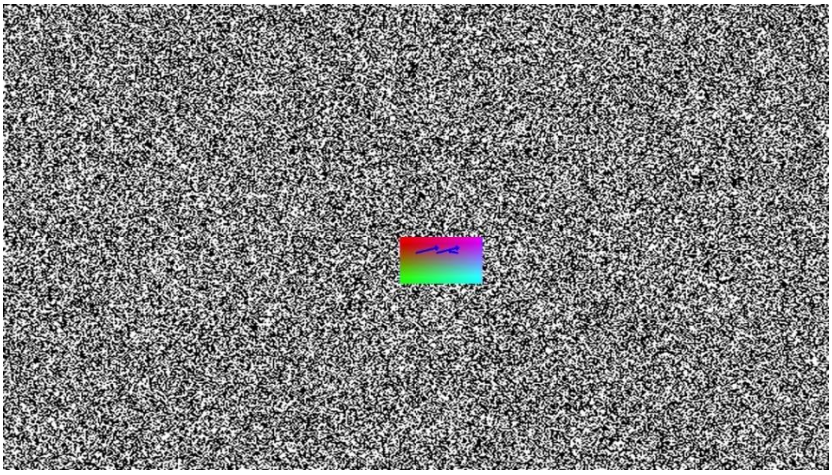
Metoda EBMA daje potpunu točnost za pomake objekta koji su u dosegu područja pretrage za rezoluciju okvira 1280x720 elemenata slike i makroblok 32x32 elementa slike. Rezultati su prikazani tablicom 4.1. koja je dobivena na 72 para okvira iz *dummy* niza okvira.

Tablica 4.1. *Točnost metode EBMA za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	100	100	100	Desno	100	100	100
Dolje	100	100	100	Dolje	100	100	100
Dolje desno	100	100	100	Dolje desno	100	100	100
Lijevo	100	0	0	Lijevo	100	0	0
Gore	100	0	0	Gore	100	0	0
Gore lijevo	100	0	0	Gore lijevo	100	0	0
Gore desno	100	0	0	Gore desno	100	0	0
Dolje lijevo	100	0	0	Dolje lijevo	100	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	100	100	100
Dolje	0	100	100
Dolje desno	100	100	100
Lijevo	100	0	0
Gore	100	0	0
Gore lijevo	100	0	0
Gore desno	100	0	0
Dolje lijevo	100	0	0

Dobiveni rezultati predstavljaju karakteristiku metode pune pretrage u kojoj se makroblokovi referentnog okvira uspoređuju sa svakim od makroblokova trenutnog okvira u području pretrage. Iako numerički rezultati točnosti pokazuju da nema točno pronađenih VP za pomake koji su djelomično van područja pretrage, subjektivnom analizom rezultata utvrđuje se da postoji pdređena točnost i kod takvih, prikazano na slici 4.5.

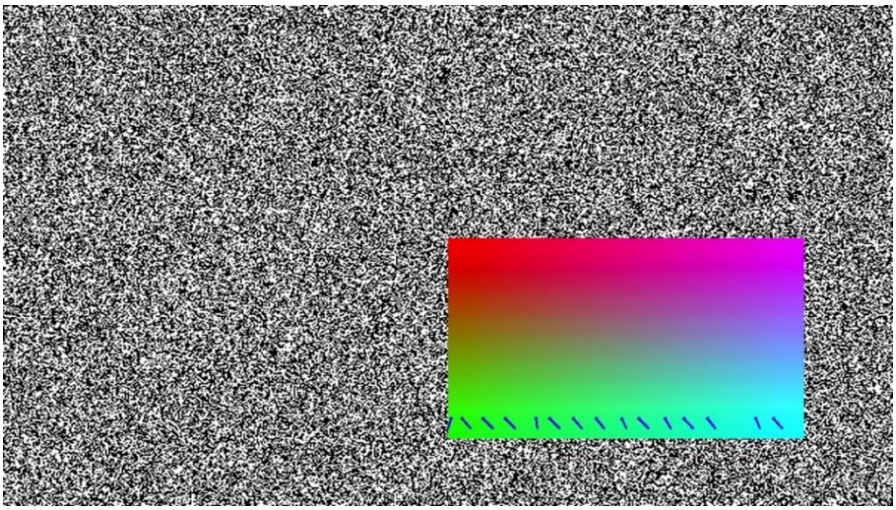


Slika 4.5. *Subjektivno točno pronađeni VP za koje je numerički dobivena točnost od 0%.*

Rezultat koji je vidljiv na slici 4.5. je smatran netočnim u tablici 4.1. jer je dobivena duljina netočna zbog ograničenja područja pretrage, a kut je djelomično točan s blažom devijacijom. Iako objekt nije u potpunosti u području pretrage, već se samo dijelovi najsličnijih makrobloka nalaze u području pretrage, mjera razlike daje dovoljno malu vrijednosti zbog koje makroblok bude odabran kao najsličniji.

Povećanje pomaka objekta uzrokuje pronalazak potpuno netočnih VP, prikazano slikom 4.6. jer se najsličniji makroblokovi koji tvore objekt ne nalaze unutar područja pretrage. Takvi VP dobiveni su uz vrijednost mjere razlike manje od postavljene granice. Izračunata mjera razlike za udaljene makroblokove koji tvore objekt (npr. makroblok na krajnjem lijevom rubu objekta i makroblok na krajnjem desnom rubu objekta) vrijednošću je manja jer je sadržaj sličan i nema velike razlike u svjetlini takva dva makrobloka.

Smanjenjem veličine okvira za dva puta po visini i dva po širini, dolazi do smanjenja točnosti za makroblokve veličine 8x8 i 16x16. Točnosti za prethodno navedeni slučaj prikazane tablicama 4.2. i 4.3.



Slika 4.6. *Primjer netočno pronađenih VP za smjer gore, pomak 25 elemenata slike, objekt veličine 512x256 elemenata slike na okviru rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike.*

Tablica 4.2. *Točnost metode EBMA za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 640x360 elemenata slike uz makroblok 8x8 elementa slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	34	31	34	Desno	38	69	63
Dolje	0	0	0	Dolje	0	0	0
Dolje desno	0	0	0	Dolje desno	0	0	0
Lijevo	53	38	0	Lijevo	63	63	0
Gore	0	3	0	Gore	0	0	0
Gore lijevo	3	3	0	Gore lijevo	0	0	0
Gore desno	0	0	0	Gore desno	4	0	0
Dolje lijevo	0	0	0	Dolje lijevo	0	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	12	21	33
Dolje	19	8	8
Dolje desno	0	0	1

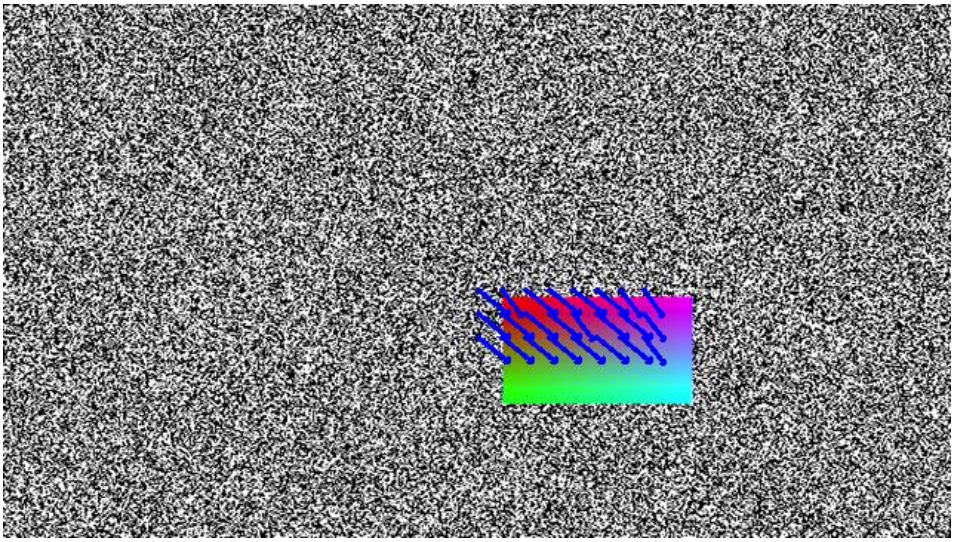
Lijevo	12	50	0
Gore	50	26	0
Gore lijevo	17	10	0
Gore desno	19	10	0
Dolje lijevo	15	11	0

Tablica 4.3. *Točnost metode EBMA za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 640x360 elemenata slike uz makroblok 16x16 elemenata slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	50	50	25	Desno	54	54	54
Dolje	0	0	0	Dolje	0	0	0
Dolje desno	0	0	0	Dolje desno	0	0	0
Lijevo	25	25	0	Lijevo	46	46	0
Gore	0	0	0	Gore	0	0	0
Gore lijevo	0	0	0	Gore lijevo	0	0	0
Gore desno	0	0	0	Gore desno	0	0	0
Dolje lijevo	0	0	0	Dolje lijevo	0	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	13	50	44
Dolje	0	0	0
Dolje desno	0	0	0
Lijevo	38	25	0
Gore	13	13	0
Gore lijevo	0	0	0
Gore desno	0	0	0
Dolje lijevo	0	0	0

Iako je numerička točnost umanjena, povećao se broj subjektivno točnih VP što je prikazano na slici 4.7.



Slika 4.7. *Primjer subjektivno točnog rezultata na okviru veličine 640x360 uz makroblok 16x16 elemenata slike i objekt veličine 256x144 elementa slike za pomak dolje desno.*

Povećanje subjektivne točnosti proizlazi iz načina stvaranja okvira manje veličine. Zbog korištenja interpolacije kod umanjivanja okvira, unosi se greška koja se očituje u manjoj, ali subjektivno prihvatljivoj, devijaciji kuta i duljina pronađenih VP na takvim okvirima te stoga postoji više subjektivno točnih od numerički točnih VP.

4.2.2. Točnost metode 3 koraka s mogućnošću povećanja broja koraka (TSS-UL)

Metoda TSS-UL za veličine okvira 1280x720 elemenata slike uz makroblok 32x32 daje visoku točnost za veličine objekata 128x72 i 256x144 koji se kreće, prikazano tablicom 4.4. Ostvarena točnost proizlazi iz iscrpne pretrage kroz tri koraka s mogućnošću povećanja broja koraka, koja se svakim korakom približava najsličnijem makrobloku, ponajviše u zadnjem koraku pretrage gdje se najsličniji makroblok pretražuje u okolici makrobloka na udaljenosti jednog elementa slike po x i y osi. Iz navedenog razloga metoda skokovito prilazi objektu koji je udaljen i za više od inicijalnog koraka pretrage.

Ostvarena točnost proizlazi iz iscrpne pretrage kroz tri koraka s mogućnošću povećanja broja koraka, koja se svakim korakom približava najsličnijem makrobloku, ponajviše u zadnjem koraku pretrage gdje se najsličniji makroblok pretražuje u okolici makrobloka na udaljenosti jednog elementa slike po x i y osi. Iz navedenog razloga metoda skokovito prilazi objektu koji je udaljen i za više od inicijalnog koraka pretrage.

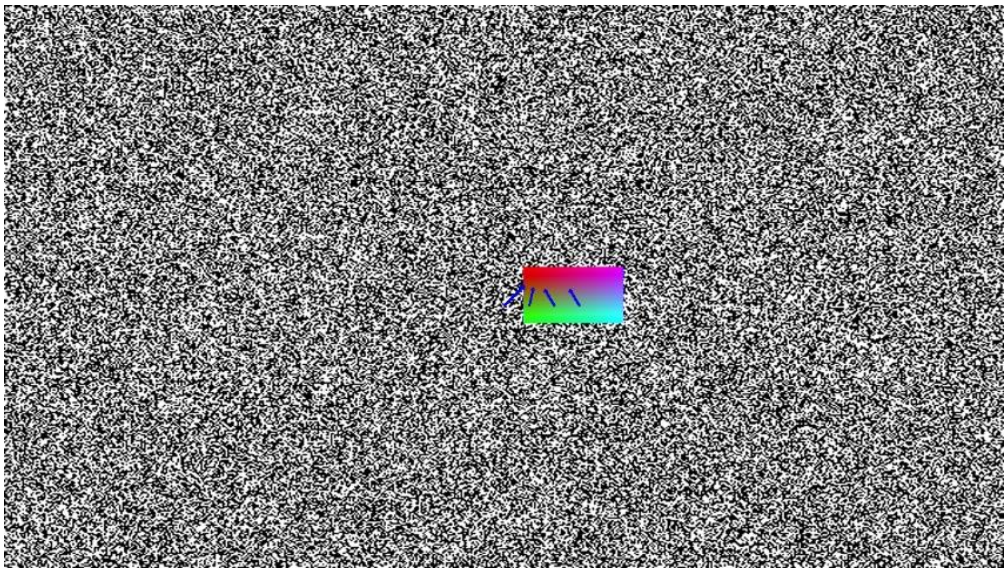
Tablica 4.4. *Točnost metode TSS-UL za dummy niz okvira koji sadrži 72 para okvira rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina koraka:	15	25	35	Veličina koraka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	100	100	0	Desno	100	100	0
Dolje	100	25	0	Dolje	100	13	0
Dolje desno	100	100	0	Dolje desno	100	100	0
Lijevo	100	100	0	Lijevo	100	100	0
Gore	100	25	0	Gore	100	13	0
Gore lijevo	100	100	0	Gore lijevo	100	100	0
Gore desno	100	25	0	Gore desno	100	13	0
Dolje lijevo	100	25	0	Dolje lijevo	100	13	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	100	100	0
Dolje	0	0	0
Dolje desno	100	100	0
Lijevo	100	100	0
Gore	100	6	0
Gore lijevo	100	87	0
Gore desno	100	35	0
Dolje lijevo	100	44	0

Broj pronađenih VP i netočnih pronađenih VP zbog uniformnosti objekata, mijenja se kao i kod metode EBMA. Na slici 4.8. prikazan je primjer netočno pronađenih vektora pokreta.

Na okvirima rezolucije 640x360 elemenata slike i makroblokovima 8x8 i 16x16 elemenata slike smanjuje se točnost, prikazano tablicama 4.5. i 4.6.



Slika 4.8. *Primjer netočno pronađenih vektora pokreta uz metodu TSS-UL za pomak gore desno.*

Tablica 4.5. *Točnost metode TSS-UL za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 640x360 elemenata slike uz makroblok 8x8 elementa slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	31	0	0	Desno	0	0	0
Dolje	0	0	0	Dolje	0	0	0
Dolje desno	0	0	0	Dolje desno	0	0	0
Lijevo	34	0	0	Lijevo	0	0	0
Gore	3	0	0	Gore	0	0	0
Gore lijevo	0	0	0	Gore lijevo	0	0	0
Gore desno	3	0	0	Gore desno	0	0	0
Dolje lijevo	0	0	0	Dolje lijevo	0	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	0	0	0
Dolje	25	0	0
Dolje desno	0	0	0
Lijevo	0	0	0

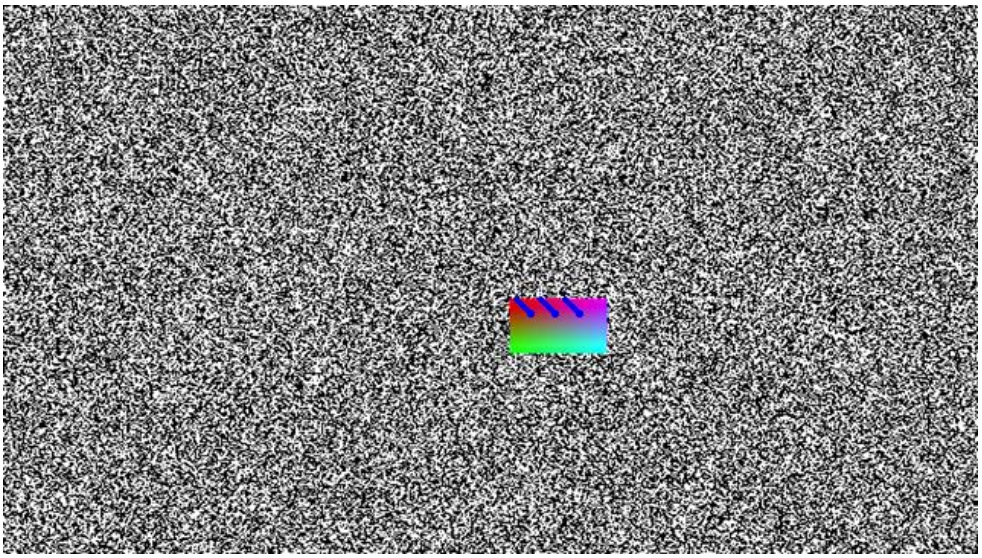
Gore	38	0	0
Gore lijevo	4	0	0
Gore desno	13	0	0
Dolje lijevo	8	0	0

Tablica 4.6. *Točnost metode TSS-UL za proanalazk VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 640x360 elemenata slike uz makroblok 16x16 elementa slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	25	0	0	Desno	42	0	0
Dolje	0	0	0	Dolje	0	0	0
Dolje desno	0	0	0	Dolje desno	0	0	0
Lijevo	25	0	0	Lijevo	33	0	0
Gore	0	0	0	Gore	0	0	0
Gore lijevo	0	0	0	Gore lijevo	0	0	0
Gore desno	0	0	0	Gore desno	0	0	0
Dolje lijevo	0	0	0	Dolje lijevo	0	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	0	0	0
Dolje	0	0	0
Dolje desno	0	0	0
Lijevo	0	0	0
Gore	6	0	0
Gore lijevo	0	0	0
Gore desno	0	0	0
Dolje lijevo	0	0	0

Točnost je manja zbog pogreške unesene umanjivanjem okvira. Primjer numerički netočnih, a subjektivno točno pronađenih VP prikazan je slikom 4.9.



Slika 4.9. *Subjektivno točni vektori pokreta na umanjenom okviru uz makroblok 16x16 elemenata slike.*

4.2.3. Točnost metode tri koraka s uključenim predviđanjem i prilagodbom na odabir oblika pretrage (PR&AD-TSS)

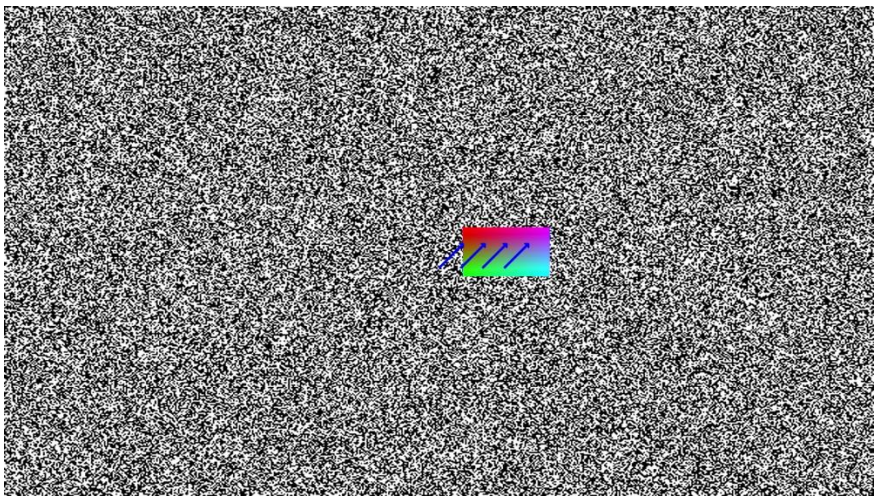
Točnost metode tri koraka s uključenim predviđanjem i prilagodbom na odabir oblika pretrage (PR&AD-TSS) daje očekivane rezultate. Točnost za okvire veličina 1280x720 uz makroblok veličine 32x32 je prikazana tablicom 4.7. Kod najmanjeg pomaka ostvarena je najveća točnost, ali nema drastičnog umanjivanja i za ostale pomake. Ova metoda ima uvjetovano umanjivanje koraka, zbog toga pretraga može pronaći najsličniji blok i kod velikog pomaka, prikazano slikom 4.10.

Tablica 4.7. *Točnost metode PR&AD-TSS za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	100	100	100	Desno	100	100	100
Dolje	25	0	0	Dolje	13	0	0
Dolje desno	25	0	100	Dolje desno	13	0	100
Lijevo	100	100	0	Lijevo	100	100	0
Gore	100	0	100	Gore	100	8	100

Gore lijevo	100	0	100	Gore lijevo	100	9	100
Gore desno	0	0	100	Gore desno	0	67	100
Dolje lijevo	25	0	0	Dolje lijevo	13	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	100	100	100
Dolje	0	0	100
Dolje desno	100	40	100
Lijevo	100	100	100
Gore	100	100	100
Gore lijevo	100	100	100
Gore desno	100	100	100
Dolje lijevo	100	9	50

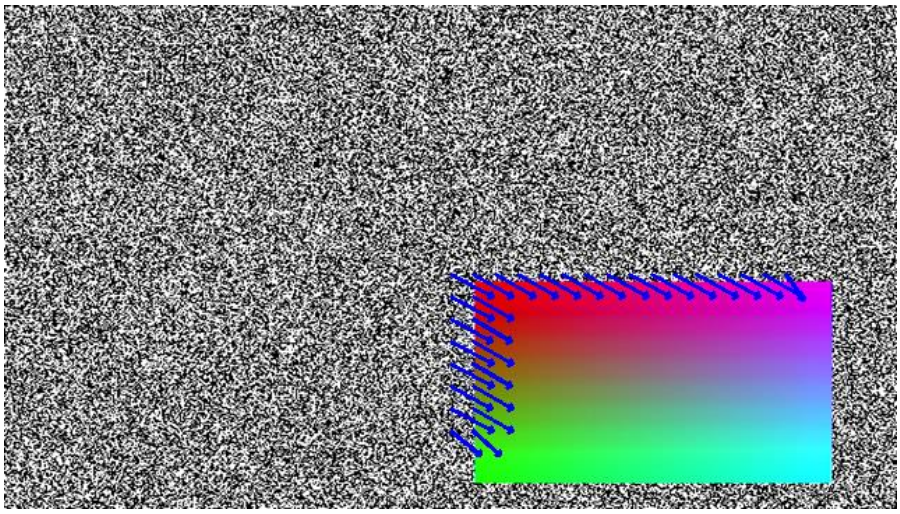


Slika 4.10. *Pronađeni VP kod pomaka 35, objekta veličine 128x72 na okviru 1280x720 elemenata slike uz makroblok 32x32 elementa slike.*

Ovakva metoda ima više netočno pronađenih VP u nekim dijelovima okvira zbog predviđanja koje koristi. Ukoliko je početak pretrage krivo usmjeren predviđanjem, iako iscrpna, pretraga teško dolazi do stvarnog najsličnijeg makrobloka jer se možda dogodilo da je najsličniji makroblok u potpuno suprotnom smjeru od predviđenog. I pri korištenju u ove metode nema pronađenih VP na uniformnim područjima najvećeg objekta.

Kod okvira rezolucije 640x360 elemenata slike i veličinama makrobloka 8x8 i 16x16 elemenata slike, numerička točnost se drastično smanjuje zbog greške unesene umanjivanjem okvira korištenjem bilinearne interpolacije. Kod objekta najveće veličine, ostvarena je veća točnost zbog povećanog dosega pretrage ove metode i zbog predikcije koja omogućava loše VP na početku pretrage, ali poboljšava numeričku točnost svakog sljedećeg pronađenog VP, prikazano slikom 4.11. Točnost umanjenih okvira je prikazana tablicama 4.8. i 4.9.

s



Slika 4.11. *Prikaz točno pronađenih VP kod najvećeg objekta na okvirima veličine 640x360 elemenata slike uz makroblok 16x16.*

Tablica 4.8. *Točnost metode PR&AD-TSS za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 640x360 elemenata slike uz makroblok 8x8 elemenata slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	47	47	45	Desno	38	71	69
Dolje	0	0	0	Dolje	0	0	0
Dolje desno	0	0	0	Dolje desno	0	0	0
Lijevo	50	44	13	Lijevo	75	56	50
Gore	0	3	0	Gore	0	0	3
Gore lijevo	3	3	3	Gore lijevo	0	0	0
Gore desno	0	0	0	Gore desno	4	0	0
Dolje lijevo	0	0	0	Dolje lijevo	0	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	12	26	39
Dolje	25	11	11
Dolje desno	2	1	1
Lijevo	12	44	59
Gore	41	21	19
Gore lijevo	15	5	6
Gore desno	15	9	9
Dolje lijevo	13	10	6

Tablica 4.9. *Točnost metode PR&AD-TSS za pronalazak VP na dummy nizu okvira koji sadrži 72 para okvira rezolucije 640x360 elemenata slike uz makroblok 16x16 elemenata slike za objekte veličina 128x72, 256x144 i 512x288 elemenata slike.*

Objekt veličine 128x72				Objekt veličine 256x144			
Veličina pomaka:	15	25	35	Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %	Smjer:	Točnost %	Točnost %	Točnost %
Desno	50	50	50	Desno	54	54	54
Dolje	0	0	0	Dolje	0	0	0
Dolje desno	0	0	0	Dolje desno	0	0	0
Lijevo	50	50	25	Lijevo	46	46	13
Gore	0	0	0	Gore	0	0	0
Gore lijevo	0	0	0	Gore lijevo	0	0	0
Gore desno	0	0	0	Gore desno	0	0	0
Dolje lijevo	0	0	0	Dolje lijevo	0	0	0

Objekt veličine 512x288			
Veličina pomaka:	15	25	35
Smjer:	Točnost %	Točnost %	Točnost %
Desno	13	38	44
Dolje	0	0	0
Dolje desno	0	0	0

Lijevo	38	25	56
Gore	13	19	0
Gore lijevo	0	0	0
Gore desno	0	0	0
Dolje lijevo	0	0	0

4.3. Testiranje točnosti predložene metode za grupiranje vektora pokreta

VP dobiveni predloženim metodama govore postoji li kretanje u sceni na kojoj se traže VP te u kojim dijelovima scene. Kako bi se odredilo postoji li više od jednog objekta koji se kreće u sceni te u kojim smjerovima se objekt ili objekti kreću između dvaju susjednih okvira, u podpoglavlju 3.4. predloženo je rješenje za njihovo grupiranje prema sličnosti karakterističnih veličina VP. Predloženo rješenje testirano je na 4 skupa nizova okvira, dva niza rezolucije 1280x720 elemenata slike, gdje jedan predstavlja slučaj u kojemu kamera miruje (*mirna kamera*), a drugi predstavlja slučaj u kojemu se kamera kreće (*pokretna kamera*). Iz dva navedena niza okvira dobivena su druga dva smanjenjem rezolucije okvira na 640x360 elemenata slike. Na prva dva niza provedena je pretraga VP uz makroblok veličine 32x32 elementa slike, a na druga dva provedena je pretraga VP uz makroblokove veličine 8x8 i 16x16 elemenata slike. Pronađeni VP, za svaki par iz niza okvira, grupirani su predloženom metodom za grupiranje s ciljem određivanja lokacije, smjera te broja objekata koji se kreću između dvaju susjednih okvira. Predmet ovog poglavlja je analiza točnosti grupiranja VP rješenja implementiranog na Alpha ploču. Analiza je izvršena subjektivnom procjenom, budući da točni iznos i smjer pokreta nisu poznati kao kod *dummy* niza okvira, jer se radi o stvarnom sadržaju iz prometa.

Najprije se na svakom paru okvira unaprijed ručno odredio broj objekata koji se kreću između dvaju susjednih okvira kako bi bila poznata referentna informacija. Primjer odabranog objekta koji se kreće između dvaju susjednih okvira označen je narančastim graničnim okvirom na slici 4.12. Objekti koji su subjektivno daleko od kamere (otprilike 20 m i više) ne smatraju se objektima koji se kreću pa se tako na slici 4.12. za odabrani okvir odredilo da je broj objekata koji se kreće jednak 1, iako u pozadini postoji nekolicina objekata koji se kreću, ali su udaljeniji od kamere. Do navedenog načina odabira objekata se došlo jer metode pronalaska VP na takvoj udaljenosti ne pronalaze VP zbog male promjene u vrijednostima elemenata slike, tj. jako malog pomaka.

Slika 4.12. *Primjer okvira iz niza okvira mirna kamera rezolucije 1280x720 elemenata slike s označenim objektom (označen narančastom bojom) koji se kreće između prikazanog i sljedećeg okvira.*

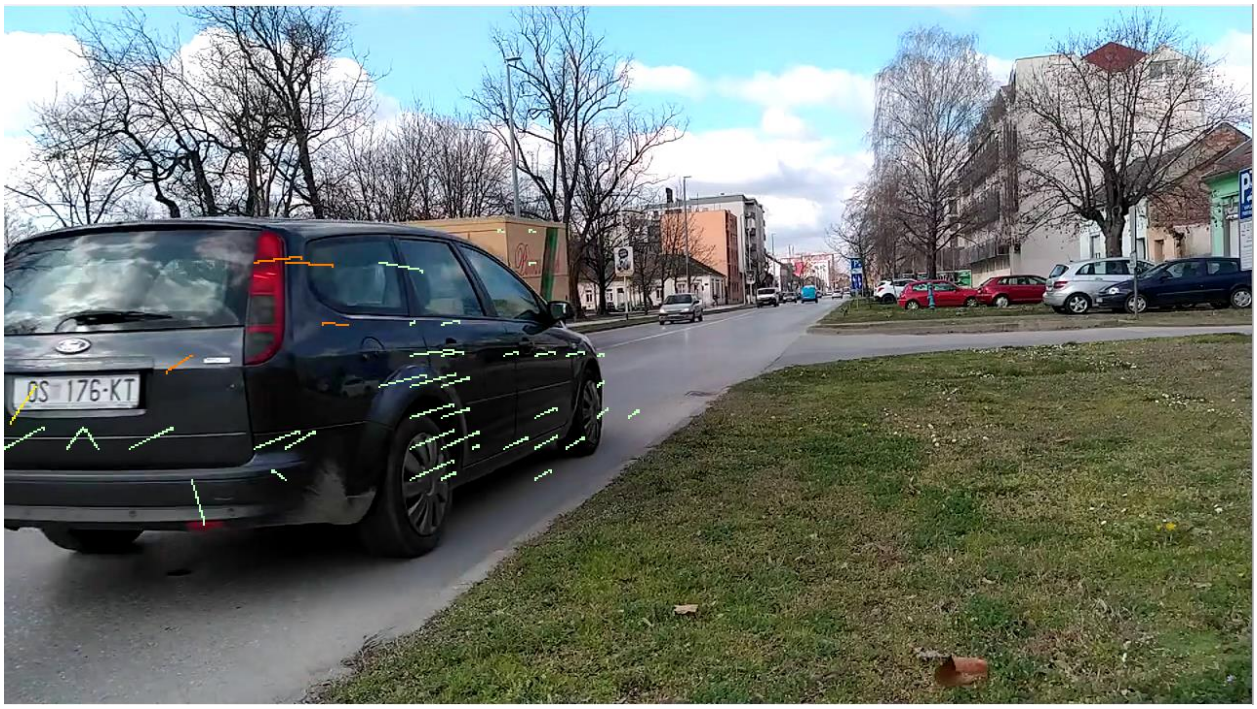
Točno pronađenim objektom smatra se svaka grupa VP (ista grupa su svi VP iscrtani na okvir istom bojom) koja se nalazi na području okvira na kojemu se nalazi ijedan od referentnih pokretnih objekata koji se kreću, te ukoliko smjer te grupe VP odgovara stvarnom smjeru kretanja objekta. Netočno pronađenom grupom smatra se svaka grupa VP koja se ne nalazi na području okvira gdje se nalazi i objekt koji se kreće. Dodatno, ukoliko postoji grupa VP na području objekta koji se kreće, no istoj grupi pripada i neki VP na udaljenosti od približno jedne četvrtine širine okvira od objekta (zbog mogućnosti postojanja sjene objekta u okolini objekta koja se također kreće), navedena grupa se ne smatra točno pronađenim objektom koji se kreće. Primjeri točno i netočno pronađenih grupa VP prikazani su na slici 4.13.

Na slici 4.13. podslici a), iscrtani VP nalaze se na području okvira na kojemu se nalazi objekt koji se kreće te su svi iscrtani VP iste boje, što znači da je objekt ispravno detektiran. Predložena metoda za grupiranje VP ponekad grupira VP istog objekta u dvije grupe (slika 4.13. b)). Iako su VP smjerom slični, duljine VP unutar grupa su različite te zbog toga ne dolazi do spajanja tih dviju grupa u dijelu provjere sličnosti grupa VP u predloženoj metodi za grupiranje. Tako pronađene dvije grupe istoga objekta smatraju se netočno pronađenim objektom, jer iako smjer koji imaju obje grupe odgovara smjeru u kojemu se objekt kreće između dvaju susjednih okvira, točan broj objekata koji se kreću nije točan. Iz navedenog razloga, kada bi metoda trebala upozoravati gdje se u području okvira nalaze objekti koji se kreću, a ne bi se pitalo koliko ih je, ovo bi bio točan rezultat. Primjer sa podslike 4.13. c) prikazuje netočno određene grupe VP. U navedenom primjeru se na području okvira na kojemu se nalazi objekt koji se kreće između prethodnog i trenutnog okvira, nalaze dvije grupe suprotnih smjerova te postoji grupa koja u sebi sadrži VP suprotnih smjerova. Na podslici d) VP jednog objekta su točno grupirani (narančasti VP), a drugi se smatraju netočno određenom grupom jer postoji VP koji se ne nalazi na objektu

kojem pripadaju i ostali VP iz grupe, već je na okviru udaljen od ostatka grupe VP za jednu četvrtinu širine okvira. Točnost je predstavljena brojem objekata koji se kreću u paru okvira te brojem objekata koji su točno pronađeni na paru okvira, prikazano u tablicama u nastavku podpoglavlja.

a)

b)



c)

d)

Slika 4.13. *Iscrtane grupe VP na okvire niza mirna kamera rezolucije 1280x720 elemenata slike i dobiveni uz makroblok 32x32 uz EBMA metodu pronalaska VP implementiranu završnim rješenjem s 3 procesora na Alpha ploči, a) točno određene grupe, b), c) i d) netočno određene grupe.*

Implementirano rješenje s tri procesora (završno rješenje) na Alpha ploči dalo je rezultate točnosti grupiranja za okvire rezolucije 1280x720 elemenata slike i makroblokove 32x32

elementa slike, za 20 parova okvira iz niza *mirna kamera*, prikazane na tablici 4.10. za sve tri predložene metode za pronalazak VP.

Tablica 4.10. *Prikaz rezultata grupiranja VP za 20 parova okvira niza mirna kamera rezolucije 1280x720 elemenata slike uz makroblok 32x32 elemenata slike za sve tri predložene metode za pronalazak VP.*

mirna kamera

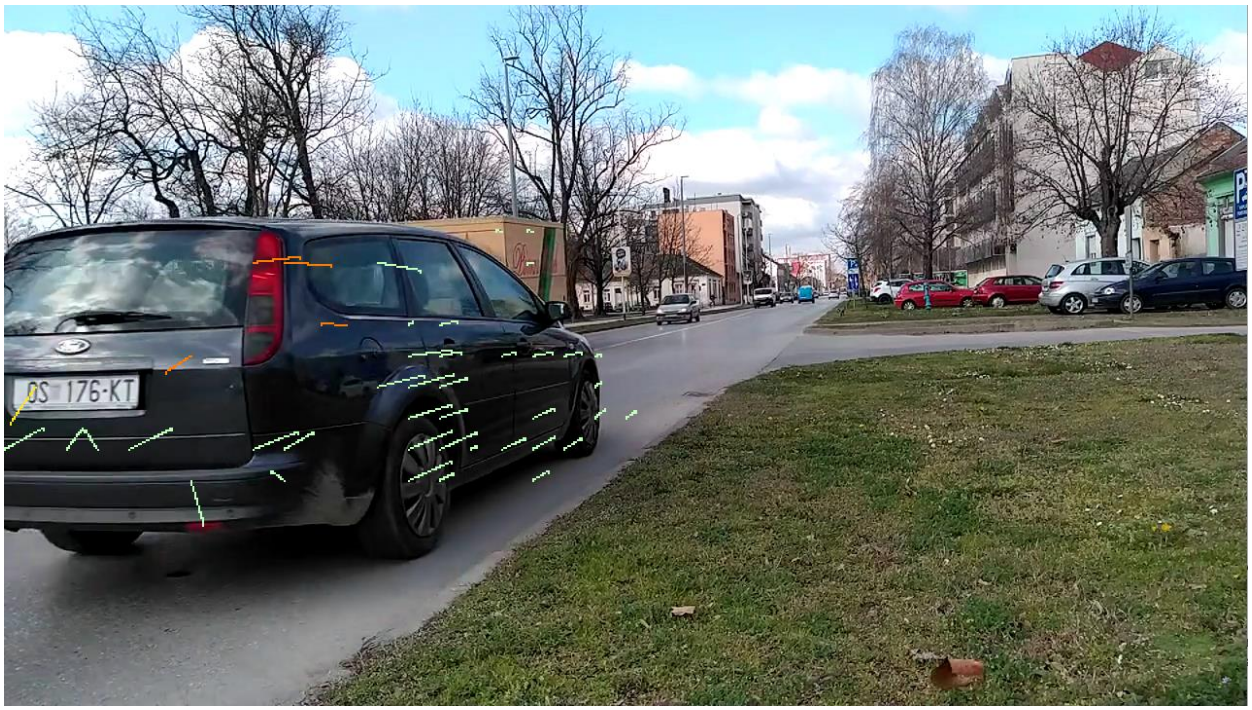
EBMA				PR&AD-TSS				TSS-UL			
Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %
1	1	1	100	1	1	1	100	1	1	1	100
2	1	1	100	2	1	1	100	2	1	1	100
3	1	1	100	3	1	1	100	3	1	1	100
4	1	1	100	4	1	1	100	4	1	1	100
5	1	1	100	5	1	1	100	5	1	1	100
6	1	0	0	6	1	0	0	6	1	0	0
7	1	1	100	7	1	1	100	7	1	1	100
8	2	2	100	8	2	2	100	8	2	2	100
9	3	3	100	9	3	1	33	9	3	1	33
10	2	2	100	10	2	2	100	10	2	2	100
11	2	2	100	11	2	1	50	11	2	1	50
12	1	1	100	12	1	1	100	12	1	0	0
13	1	1	100	13	1	1	100	13	1	1	100
14	1	1	100	14	1	1	100	14	1	1	100
15	2	0	0	15	2	0	0	15	2	0	0
16	2	2	100	16	2	1	50	16	2	0	0
17	2	2	100	17	2	1	50	17	2	1	50
18	2	2	100	18	2	1	50	18	2	1	50
19	3	2	67	19	3	2	67	19	3	1	33
20	1	1	100	20	1	1	100	20	1	1	100
Ukupno:	31	27	87	Ukupno:	31	21	68	Ukupno:	31	18	58

Analogno rezultatima testiranja za točnost pojedine metode za pretragu VP, iz tablice 4.10. uočava se da je i točnost grupiranja VP najveća za metodu EBMA. Ovakvi rezultati točnosti grupiranja VP govore kako je za ispravno grupiranje VP korištenjem predložene metode

potrebno imati što točnije pronađene VP. Svaki VP koji je netočno pronađen tijekom pretrage VP otežava ispravno grupiranje VP jer utječe na srednje vrijednosti karakterističnih veličina, zbog čega algoritam grupiranja k srednjih vrijednosti neispravno svrstava druge VP zbog krivo određenog centroida. Ukoliko bi broj grupa u koju se svrstavaju VP bio veći od broja objekata koji se nalaze na okviru, takvi VP bili bi izdvojeni u posebne grupe, no za isto je potrebno imati onoliko dodatnih grupa koliko je i krivo pronađenih VP, što je nemoguće odrediti pojedinačno za svaki par okvira. Za 6. i 15. par okvira u nizu *mirna kamera*, metode za pretragu VP pronalaze previše netočnih VP ili nije pronađen niti jedan VP, zbog čega je provedeno grupiranje na njima dalo netočne rezultate koji su vidljivi na slici 4.14.

Predložena metoda za grupiranje VP daje zadovoljavajuće rezultate i kada postoji više od jednog objekta. Točnost grupiranja u slučaju više objekata također ovisi o broju i točnosti pronađenih VP. Stoga je za slučaj u kojemu na paru okvira postoji više objekata najbolji rezultat grupiranja dala metoda pronalaska VP EBMA, što je vidljivo na slici 4.15.

a)



b)

Slika 4.14. *Primjer iscrtanih, netočno grupiranih, VP iz niza mirna kamera rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike metode EBMA a) zbog nepronađenih VP, b) zbog previše netočno pronađenih VP.*

U slučaju niza okvira *pokretna kamera* rezolucije 1280x720 elemenata slike provjera točnosti na način na koji je ona do sada rađena nije smislena jer se cijela slika, uključujući i mirnu pozadinu, pomiče zbog pokreta kamere kojom se snima. Ipak se željela ispitati i primjenjivost rješenja u takvim situacijama pa je točnost ipak testirana. Kao što je i bilo očekivano, rješenje se pokazalo nepraktično za situacije u kojima se kamere kreće. Točno grupiranje ostvareno je samo za okvire u kojima objekt koji se kreće zauzima većinu okvira, ili u slučaju kada je pokret kamere iznimno mali (npr. za 2 elementa slike), jer tada nema velikog pomicanja pozadine (VP manji od 2 elementa slike se izbacuju), čiji bi VP inače uzrokovali netočno grupiranje. Objekt koji zauzima većinu područja okvira omogućava da zbog brojnosti VP bude predstavljen jednim centroidom, a pozadina koja se kreće drugim i/ili trećim (prikazano na slici 4.16.). Prema tablici 4.11. vidljivo je da na grupiranje VP nije utjecala točnost metoda za pronalazak VP jer je za sve metode ostvarena niska točnost.

a)

b)

c)

Slika 4.15. *Primjer iscrtanih grupiranih VP iz niza mirna kamera rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike u slučaju više objekata na okviru za svaku od metoda pretrage VP, a) EBMA, 3 objekta, b) PR&AD-TSS, 2 objekta, c) TSS-UL, 2 objekta.*

Slika 4.16. *Iscrtani grupirani VP za 9. par okvira iz niza pokretna kamera, rezolucije okvira 1280x720 elemenata slike uz makroblok 32x32 elementa slike, korištenjem metode EBMA za pronalazak VP.*

Tablica 4.11. *Točnost grupiranja VP za 20 parova okvira iz niza pokretna kamera rezolucije okvira 1280x720 elemenata slike uz makroblok 32x32 elementa slike za sve tri predložene metode pronalaska VP.*

pokretna kamera

EBMA				PR&AD-TSS				TSS-UL			
Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %
1	2	0	0	1	2	0	0	1	2	0	0
2	1	1	100	2	1	1	100	2	1	1	100
3	1	0	0	3	1	0	0	3	1	0	0
4	2	0	0	4	2	0	0	4	2	0	0
5	2	0	0	5	2	0	0	5	2	0	0
6	4	0	0	6	4	0	0	6	4	0	0
7	0	0	0	7	0	0	0	7	0	0	0
8	1	0	0	8	1	0	0	8	1	0	0
9	1	1	100	9	1	1	100	9	1	1	100
10	0	0	0	10	0	0	0	10	0	0	0
11	1	0	0	11	1	0	0	11	1	0	0
12	2	0	0	12	2	0	0	12	2	0	0
13	2	0	0	13	2	0	0	13	2	0	0
14	2	0	0	14	2	0	0	14	2	0	0
15	0	0	0	15	0	0	0	15	0	0	0
16	1	0	0	16	1	0	0	16	1	0	0
17	2	0	0	17	2	0	0	17	2	0	0
18	2	0	0	18	2	0	0	18	2	0	0
19	0	0	0	19	0	0	0	19	0	0	0
20	1	0	0	20	1	0	0	20	1	0	0
Ukupno:	27	2	7	Ukupno:	27	2	7	Ukupno:	27	2	7

Nizovi okvira *mirna kamera* i *pokretna kamera* smanjeni na rezoluciju 640x360 elemenata slike dali su približno jednaku točnost grupiranja VP za sve metode pretrage VP, u usporedbi s okvirima rezolucije 1280x720 elemenata slike (uz makroblok 32x32 elementa slike), uz obje veličine makrobloka za nižu rezoluciju (8x8 i 16x16 elemenata slike). Rezultati točnosti grupiranja VP za rezoluciju okvira 640x360 elemenata slike prikazani su uz veličinu makrobloka 8x8 elemenata slike u tablici 4.12. i veličinu makrobloka 16x16 elemenata slike u tablici 4.13.

Tablica 4.13. *Točnost grupiranja VP 20 parova okvira za nizove okvira a) mirna kamera i b) pokretna kamera za sve tri predložene metode pretrage VP, rezolucije 640x360 elemenata slike uz makroblok 8x8 elemenata slike.*

a)

mirna kamera

EBMA				PR&AD-TSS				TSS-UL			
Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %
1	1	1	100	1	1	1	100	1	1	1	100
2	1	1	100	2	1	1	100	2	1	1	100
3	1	1	100	3	1	1	100	3	1	1	100
4	1	1	100	4	1	1	100	4	1	1	100
5	1	1	100	5	1	1	100	5	1	1	100
6	1	0	0	6	1	0	0	6	1	1	100
7	1	1	100	7	1	1	100	7	1	1	100
8	2	1	50	8	2	1	50	8	2	1	50
9	3	2	67	9	3	1	33	9	3	1	33
10	2	1	50	10	2	1	50	10	2	1	50
11	2	2	100	11	2	2	100	11	2	2	100
12	1	1	100	12	1	1	100	12	1	1	100
13	1	0	0	13	1	1	100	13	1	1	100
14	1	1	100	14	1	0	0	14	1	0	0
15	2	0	0	15	2	0	0	15	2	0	0
16	2	1	50	16	2	1	50	16	2	1	50
17	2	1	50	17	2	1	50	17	2	1	50
18	2	0	0	18	2	0	0	18	2	0	0
19	3	1	33	19	3	0	0	19	3	0	0
20	1	0	0	20	1	1	100	20	1	0	0
Ukupno:	31	17	55	Ukupno:	31	16	52	Ukupno:	31	16	52

b)

pokretna kamera

EBMA				PR&AD-TSS				TSS-UL			
Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %
1	2	0	0	1	2	0	0	1	2	0	0
2	1	0	0	2	1	0	0	2	1	0	0
3	1	0	0	3	1	0	0	3	1	0	0
4	2	0	0	4	2	0	0	4	2	0	0
5	2	0	0	5	2	0	0	5	2	0	0
6	4	0	0	6	4	0	0	6	4	0	0
7	0	0	0	7	0	0	0	7	0	0	0
8	1	0	0	8	1	0	0	8	1	0	0
9	1	1	100	9	1	1	100	9	1	1	100
10	0	0	0	10	0	0	0	10	0	0	0
11	1	0	0	11	1	0	0	11	1	0	0
12	2	0	0	12	2	0	0	12	2	0	0
13	2	0	0	13	2	0	0	13	2	0	0
14	2	0	0	14	2	0	0	14	2	0	0
15	0	0	0	15	0	0	0	15	0	0	0
16	1	0	0	16	1	0	0	16	1	0	0
17	2	0	0	17	2	0	0	17	2	0	0
18	2	0	0	18	2	0	0	18	2	0	0
19	0	0	0	19	0	0	0	19	0	0	0
20	1	0	0	20	1	0	0	20	1	0	0
Ukupno:	27	1	4	Ukupno:	27	1	4	Ukupno:	27	1	4

Tablica 4.14. *Točnost grupiranja VP 20 parova okvira za nizove okvira a) mirna kamera i b) pokretna kamera za sve tri predložene metode pretrage VP, rezolucije 640x360 elemenata slike uz makroblok 16x16 elemenata slike.*

a)

mirna kamera

EBMA				PR&AD-TSS				TSS-UL			
Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %
1	1	1	100	1	1	1	100	1	1	1	100
2	1	1	100	2	1	1	100	2	1	1	100
3	1	0	0	3	1	0	0	3	1	1	100
4	1	1	100	4	1	1	100	4	1	1	100
5	1	1	100	5	1	1	100	5	1	1	100
6	1	0	0	6	1	1	100	6	1	1	100
7	1	1	100	7	1	1	100	7	1	0	0
8	2	1	50	8	2	1	50	8	2	1	50
9	3	2	67	9	3	2	67	9	3	2	67
10	2	1	50	10	2	1	50	10	2	1	50
11	2	2	100	11	2	2	100	11	2	0	0
12	1	1	100	12	1	1	100	12	1	1	100
13	1	1	100	13	1	1	100	13	1	1	100
14	1	1	100	14	1	1	100	14	1	0	0
15	2	1	50	15	2	0	0	15	2	0	0
16	2	2	100	16	2	2	100	16	2	2	100
17	2	2	100	17	2	2	100	17	2	2	100
18	2	1	50	18	2	0	0	18	2	0	0
19	3	2	67	19	3	2	67	19	3	2	67
20	1	1	100	20	1	1	100	20	1	1	100
Ukupno:	31	23	74	Ukupno:	31	22	71	Ukupno:	31	19	61

b)

pokretna kamera

EBMA				PR&AD-TSS				TSS-UL			
Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %	Redni broj para okvira	Broj objekata	Broj pronađenih objekata	Točnost %
1	2	0	0	1	2	0	0	1	2	0	0
2	1	0	0	2	1	0	0	2	1	0	0
3	1	0	0	3	1	0	0	3	1	0	0
4	2	0	0	4	2	0	0	4	2	0	0
5	2	0	0	5	2	0	0	5	2	0	0
6	4	0	0	6	4	0	0	6	4	0	0
7	0	0	0	7	0	0	0	7	0	0	0
8	1	0	0	8	1	0	0	8	1	0	0
9	1	1	100	9	1	1	100	9	1	1	100
10	0	0	0	10	0	0	0	10	0	0	0
11	1	0	0	11	1	0	0	11	1	0	0
12	2	0	0	12	2	0	0	12	2	0	0
13	2	0	0	13	2	0	0	13	2	0	0
14	2	0	0	14	2	0	0	14	2	0	0
15	0	0	0	15	0	0	0	15	0	0	0
16	1	0	0	16	1	0	0	16	1	0	0
17	2	0	0	17	2	0	0	17	2	0	0
18	2	0	0	18	2	0	0	18	2	0	0
19	0	0	0	19	0	0	0	19	0	0	0
20	1	0	0	20	1	0	0	20	1	0	0
Ukupno:	27	1	4	Ukupno:	27	1	4	Ukupno:	27	1	4

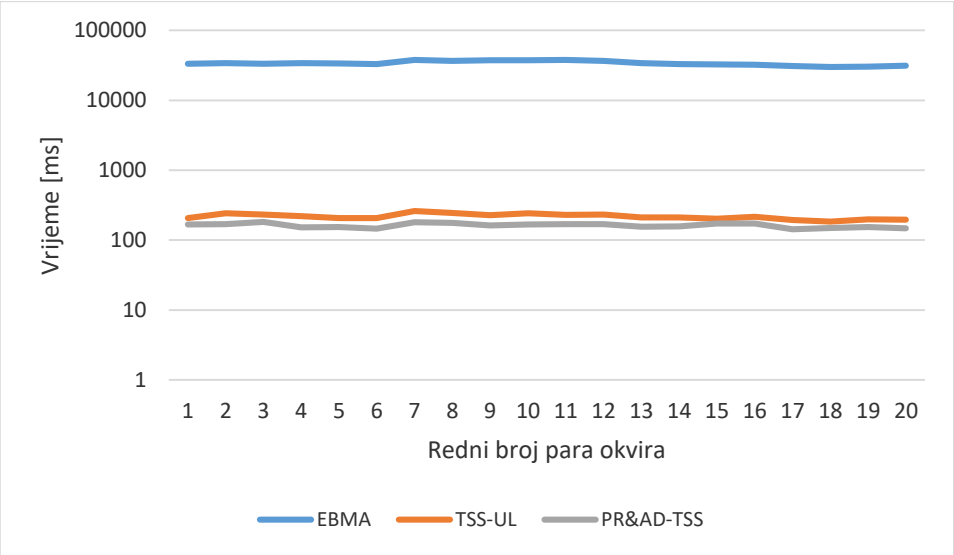
S obzirom da je uz manje makroblokove (8x8 i 16x16 elemenata slike) kod rezolucije okvira 640x360 elemenata slike broj pronađenih VP za svaki par okvira puno veći od onog kod veće rezolucije i većeg makrobloka, veći broj VP znači da je i veća vjerojatnost da ima više netočno pronađenih VP, no predložena metoda za grupiranje VP i tu zadržava velik postotak točnosti.

4.4. Testiranje predloženog rješenja na kompletnom skupu nizova okvira uz mjerenje brzine izvođenja rješenja implementiranih na PC-u i na ADAS ploči

Testiranje je provedeno na svim testnim nizovima koji su opisani na početku ovoga poglavlja. Na ulaze rješenja na PC-u, prije i poslije algoritamske optimizacije, postavljani su testni nizovi za koje je mjereno i pohranjivano vrijeme izvođenja za svaki par okvira. Testiranje rješenja implementiranog na ADAS platformi provedeno je kroz 3 korisnički definirana rješenja – na jednom DSP procesoru, na jednom A15 procesoru te na dva DSP procesora i jednom A15 procesoru na koje su paralelno raspodijeljeni zadaci pronalaska VP. Vrijeme izvođenja pojedine veze u korisnički definiranom rješenju, serijskim sučeljem slalo se na osobno računalo. Za svaki niz okvira i svaku metodu, rezultati su spremani u datoteku ekstenzije *.log* te su potom Python skriptom izdvojeni u tablice.

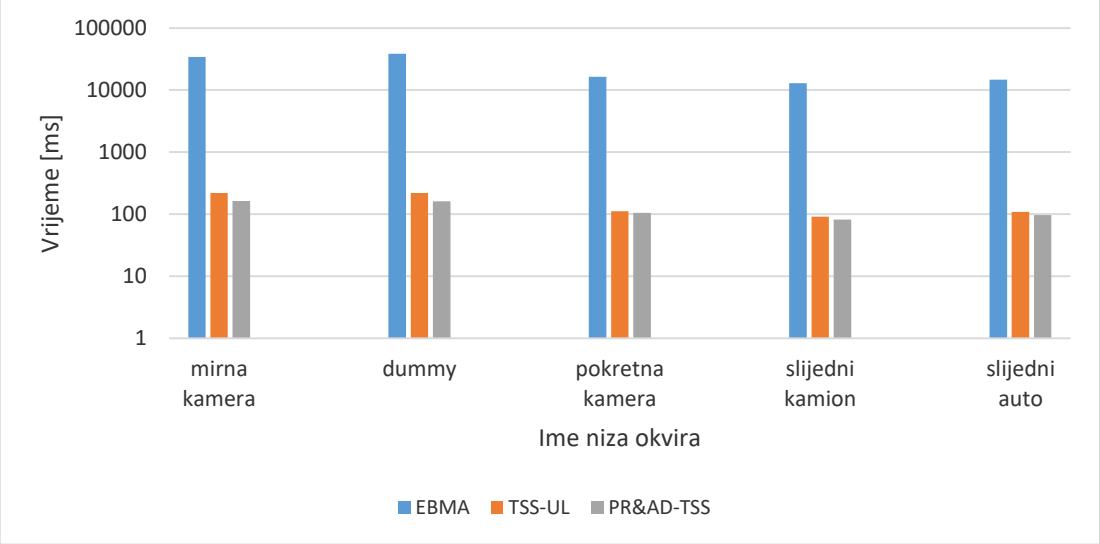
4.4.1. Neoptimizirano rješenje implementirano na PC-u

Prvo rješenje stvoreno je za implementaciju na PC-u za kojeg je dobiven očekivani odnos vremena izvođenja svih triju metoda. EBMA ostvaruje najsporije vrijeme izvođenja za svaki testni niz na kojemu je testiran, dok TSS-UL i PR&AD-TSS ostvaruju približno iste rezultate. Ovakav rezultat ostvaren je zbog osobitosti PR&AD-TSS metode da zbog uvjetovanog smanjenja koraka brže dolazi u područje najsličnijeg makrobloka. Osim navedenog, PR&AD-TSS metoda kumulativno radi manji broj pretraga od svih ostalih jer ovisno o lokaciji pronađenog najsličnijeg makrobloka u trenutnom koraku, uvjetno odabire makroblove za pretragu u sljedećem koraku samo za one makroblokove za koje još nije napravljena usporedba, tj. npr. ako je pronađeni najsličniji makroblok lijevi onda se donji lijevi i donji preskaču u sljedećem koraku pretrage jer je s njima već napravljena usporedba. Graf s vremenom izvođenja neoptimiziranog rješenja na PC-u opisanog u podpoglavlju 3.5., svake od triju metoda za pronalazak VP, za niz okvira *mirna kamera* rezolucije 1280x720 elemenata slike i makroblok 32x32 elementa slike, prikazan je na slici 4.17.



Slika 4.17. *Graf vremena izvođenja svih metoda za pronalazak VP neoptimiziranog rješenja na PC-u na nizu okvira mirna kamera za okvir rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike.*

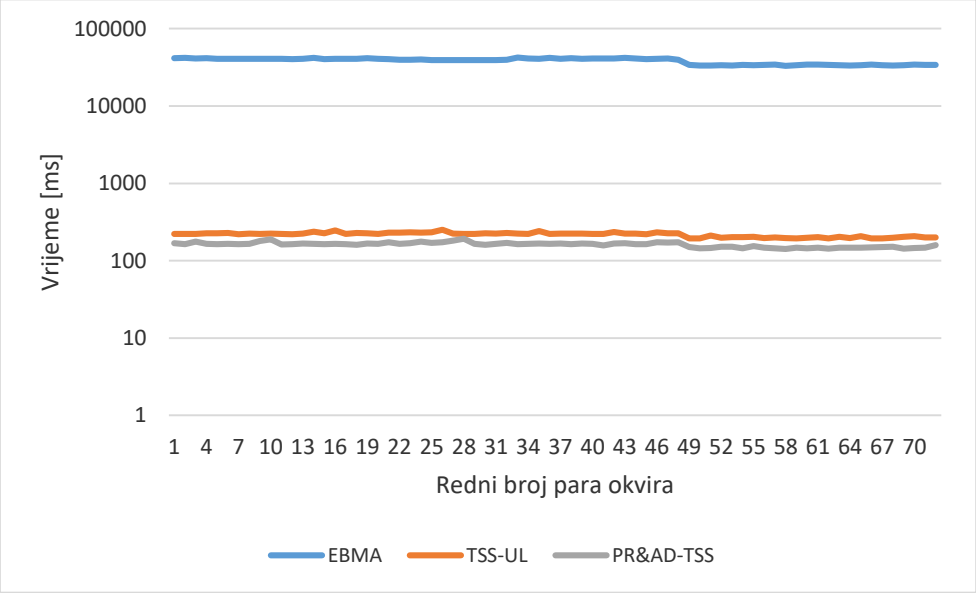
Usporedbom rezultata vremena izvođenja neoptimiziranog rješenja na PC-u, za sve metode na cjelokupnom testnom skupu nizova okvira, uočava se razlika u vremenu izvođenja za različite nizove, prikazano grafom sa slike 4.18. Takav rezultat je pokazatelj utjecaja sadržaja okvira na vrijeme izvođenja pronalaska VP.



Slika 4.18. *Graf srednjih vremena izvođenja svih metoda za pronalazak VP neoptimiziranog rješenja na PC-u na svakom od testnih nizova okvira uz rezoluciju 1280x720 elemenata slike uz makroblok 32x32 elementa slike.*

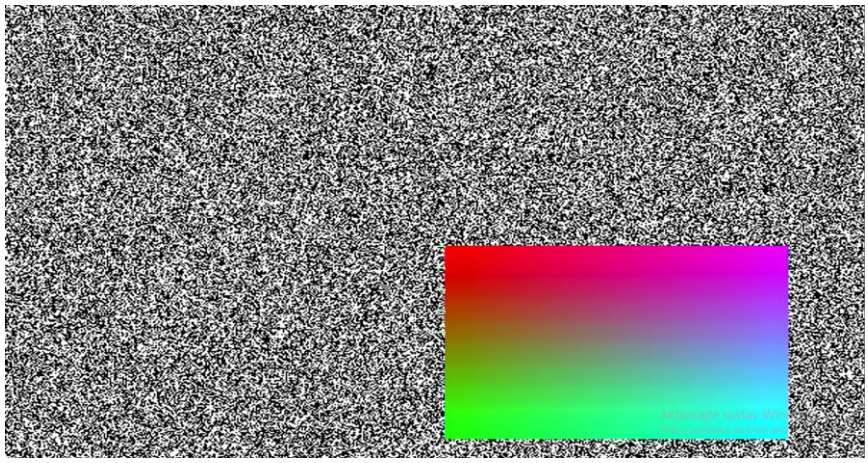
S obzirom da je na svakom okviru *dummy* niza sadržaj pretežito neuniforman zbog biplanarnog platna, za takav niz je vrijeme izvođenja najveće. Svi nizovi okvira koji su dobiveni iz video sekvence snimljene pokretnom kamerom posjeduju veća područja okvira uniformnog sadržaja u usporedbi s mirnom kamerom, jer se na okvirima nalazi više sadržaja kao što su cesta, pokrov kamiona i bočni zid uz cestu (vidljivo na slici 4.1. podslikama a) i d)).

Nastavno na prethodni zaključak o utjecaju sadržaja okvira na vrijeme izvođenja, kod vremena izvođenja predloženih metoda na *dummy* nizu okvira veličine 1280x720 elemenata slike i makrobloku 32x32 elementa slike, primjećuje se trend smanjenja vremena izvođenja za sve predložene metode nakon 49. okvira *dummy* niza, prikazano slikom 4.19.



Slika 4.19. *Graf vremena izvođenja svih metoda za pronalazak VP neoptimiziranog rješenja na PC-u na dummy nizu okvira uz rezoluciju 1280x720 elemenata slike i makroblok 32x32 elementa slike.*

Takav rezultat je ostvaren zbog korištenja mjere uniformnosti makrobloka. Ukoliko je mjera uniformnosti iznad neke granice, makroblok se preskače kod pretrage. U nizu okvira, nakon 49. okvira, objekt koji se kreće poprima veličinu 512x256 elemenata slike te se uniformna područja povećavaju, prikazano slikom 4.20.



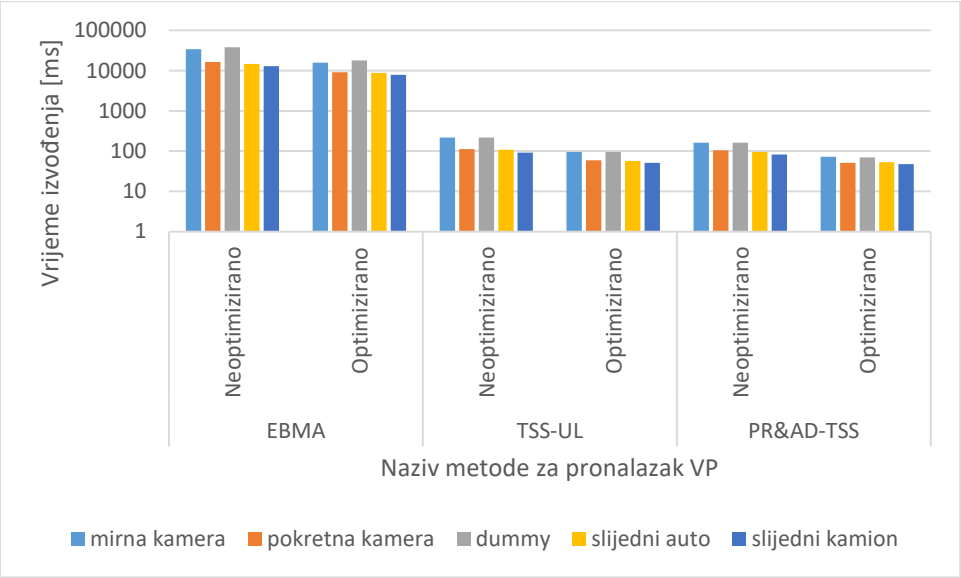
Slika 4.20. *Prikaz uniformnosti objekta veličine 512x256 elemenata slike koji se pojavljuje nakon 49. ovkira u dummy nizu okvira.*

4.4.2. Optimizirano rješenje implementirano na PC-u

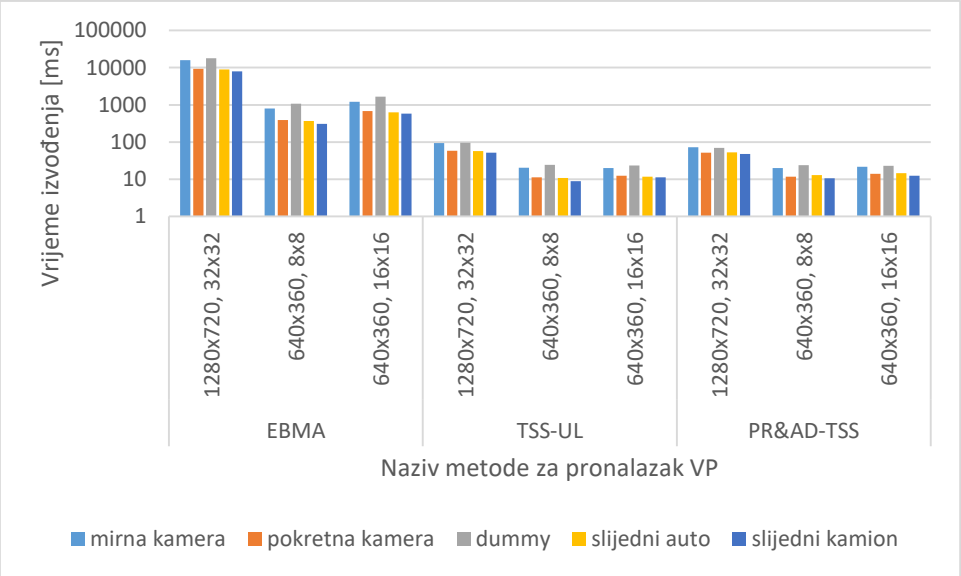
Drugo rješenje na kojemu je provedeno mjerenje vremena izvođenja za sve metode je algoritamski optimizirano rješenje na PC-u. Međusobni odnosi vremena izvođenja svih triju metoda ostali su isti kao i kod neoptimiziranog rješenja. Slika 4.21. prikazuje da je za svaku od metoda postignuto smanjenje vremena izvođenja. Takav rezultat proizlazi iz optimizacije opisane u dijelu 3.6.2.

Smanjivanje testnih nizova na rezoluciju okvira rezolucije 640x360 elemenata slike uz makroblokove 16x16 i 8x8 elemenata slike, ostvareno je znatno smanjenje vremena izvođenja kod algoritamski optimiziranog rješenja na PC-u. To je rezultat manjih makroblokova, jer je tako manji broj iteracija petlje kod izračuna mjere razlike, kao i 4 puta manjeg područja pretrage na pojedinom okviru. Usporedba vremena izvođenja umanjenog i neumanjenog niza, korištenjem algoritamski optimiziranog rješenja, prikazan je grafom sa slike 4.22.

Rezultati optimizacije prikazani u tablici 4.15. prikazuju da je ostvareno približno dvostruko smanjenje vremena izvođenja za svaku predloženu metodu za pronalazak VP uz rezoluciju 1280x720 elemenata slike i makroblok 32x32 elementa slike, a tablica 4.16. uz rezoluciju 640x360 elemenata slike i makroblokove 8x8 i 16x16 elemenata slike.



Slika 4.21. *Graf usporedbe srednjeg vremena izvođenja metoda za pronalazak VP optimiziranog i neoptimiziranog rješenja na PC-u za svaki testni niz okvira i sve metode uz okvire rezolucije 1280x720 elemenata slike uz makroblok 32x32 elemenata slike.*



Slika 4.22. *Graf usporedbe vremena izvođenja metoda za pronalazak VP algoritamski optimiziranog rješenja za cijeli skup testnih nizova okvira uz rezoluciju 1280x720 elemenata slike i uz makroblok 32x32 elementa slike te uz rezoluciju 640x360 elemenata slike i uz makroblokove 16x16 i 8x8 elemenata slike.*

Tablica 4.15. *Srednje vrijeme izvođenja optimiziranog i neoptimiziranog programskog rješenja za pronalazak VP implementiranih na osobnom računalu na kompletnom skupu testnih nizova uz rezoluciju 1280x720 elemenata slike i uz makroblok 32x32 elementa slike.*

Ime metode:	EBMA		TSS-UL		PR&AD-TSS	
Optimizacija:	NE	DA	NE	DA	NE	DA
Ime niza	Srednje vrijeme izvođenja [ms]					
mirna kamera	34021,95	15728,15	218,30	94,45	162,40	72,15
pokretna kamera	16445,60	9220,35	111,35	58,75	105,10	51,40
dummy	38379,76	18004,44	217,96	94,92	161,31	70,17
slijedni auto	14681,13	8866,80	108,70	57,33	96,03	52,53
slijedni kamion	12939,35	7946,68	91,48	51,33	81,95	47,45

Tablica 4.16. *Srednje vrijeme izvođenja optimiziranog i neoptimiziranog programskog rješenja za pronalazak VP implementiranih na osobnom računalu na kompletnom skupu testnih nizova uz rezoluciju 640x360 elemenata slike i uz makroblokove a) 8x8 i b) 16x16 elementa slike.*

a)

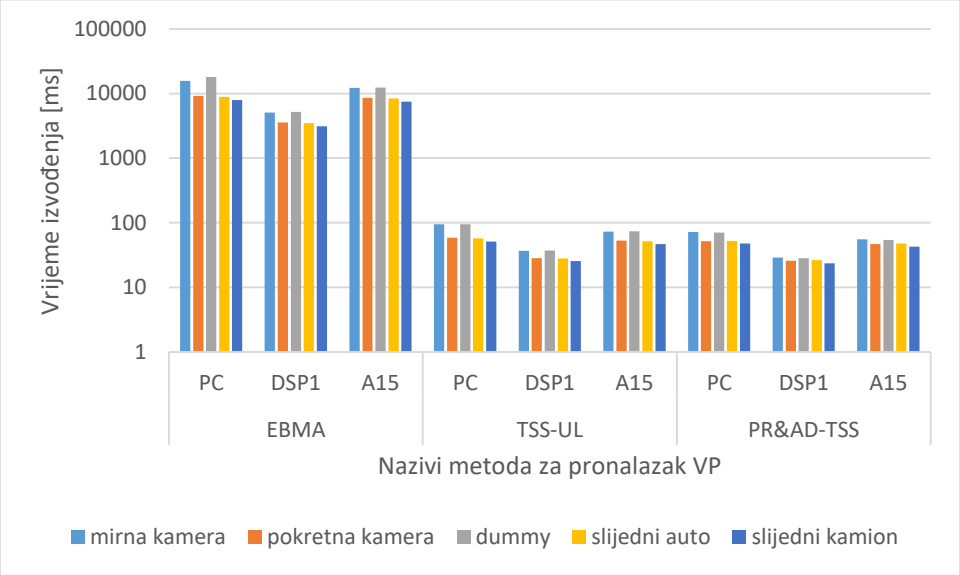
Ime metode:	EBMA		TSS-UL		PR&AD-TSS	
Optimizacija:	NE	DA	NE	DA	NE	DA
Ime niza	Srednje vrijeme izvođenja [ms]					
mirna kamera	2021,85	802,45	50,15	20,35	48,95	20,00
pokretna kamera	690,10	387,05	20,65	11,25	24,75	11,85
dummy	2679,39	1082,78	59,63	24,63	57,07	23,74
slijedni auto	570,05	365,68	17,64	10,90	19,82	12,88
slijedni kamion	494,33	310,10	16,03	8,85	18,67	10,73

b)

Ime metode:	EBMA		TSS-UL		PR&AD-TSS	
Optimizacija:	NE	DA	NE	DA	NE	DA
Ime niza	Srednje vrijeme izvođenja [ms]					
mirna kamera	2741,05	1196,20	47,75	20,20	46,05	21,60
pokretna kamera	1270,30	683,45	23,40	12,45	25,30	14,00
dummy	3674,31	1644,42	52,07	23,56	50,17	23,00
slijedni auto	1096,33	632,43	21,00	11,65	23,74	14,50
slijedni kamion	985,08	575,08	19,03	11,25	22,90	12,38

4.4.3. Optimizirana rješenja implementirana na jedan DSP procesor odnosno na jedan A15 procesor Alpha ploče

Nakon algoritamske optimizacije rješenja na PC-u isto je implementirano i na ADAS Alpha realnu platformu. Kako bi se usporedilo vrijeme izvođenja pojedinog procesora Alpha ploče stvorena su dva korisnička rješenja, jedno s implementacijom na DSP1 procesor te jedno s implementacijom na A15 procesor. Sve tri predložene metode za pronalazak VP imaju međusobno jednak odnos vremena izvođenja na oba procesora, kao i kod rješenja na PC-u. Usporedba vremena izvođenja za navedena tri slučaja (PC, DSP1 i A15), na svim testnim nizovima okvira rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike, prikazan je grafom sa slike 4.23.

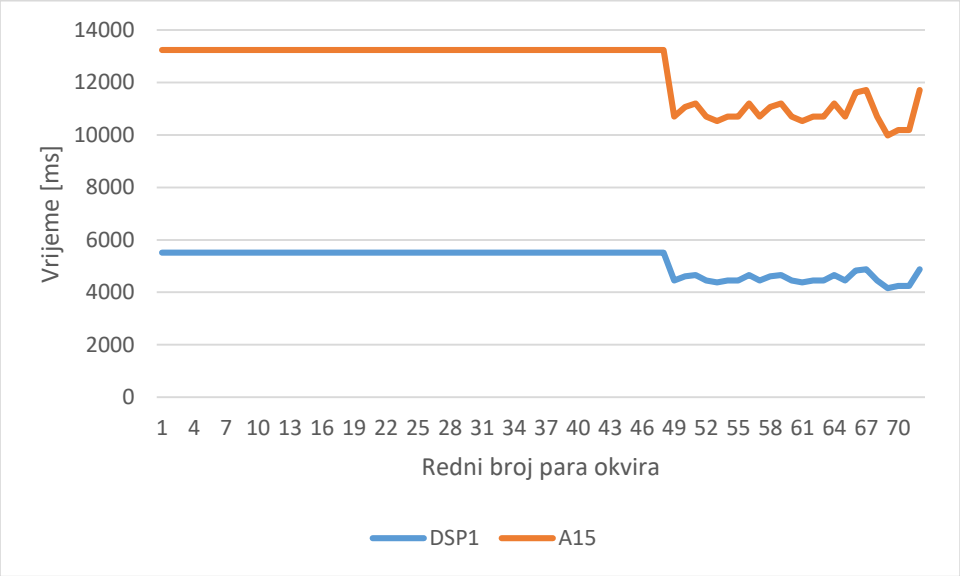


Slika 4.23. *Graf usporedbe vremena izvođenja metoda za pronalazak VP algoritamski optimiziranog rješenja na PC-u, procesoru DSP1 i procesoru A15, uz nizove okvira rezolucije 1280x720 elemenata slike i uz makroblok 32x32 elementa slike.*

Usporedba rješenja na 3 prethodno navedena slučaja pokazuje bolje rezultate izvođenja svake od metoda na svakom testnom nizu kod implementacije na ADAS platformi u usporedbi s implementacijom na PC-u. Očekivano bi bilo da je PC brži u izvođenju predloženih metoda zbog veće frekvencije procesora, međutim tijekom mjerenja vremena izvođenja uvjeti na računalu (pozadinski zadaci na računalu, sporiji rad zbog dotrajalosti komponenti unutar računala i sl.) su vjerojatno ograničili frekvenciju procesora, koja je za predviđeni zadatak od iznimne važnosti zbog velikog broja računanja koje je potrebno provesti za svaki makroblok, te su znatno usporili brzinu izvođenja na PC-u u usporedbi s onom na ADAS platformi. Još jedno od mogućih

objašnjenja ovakvog rezultata je udaljenost *cache* memorije od samog CPU-a koja bi omogućila brže izvođenje prilikom izračuna mjere razlike, ali ne postoji dokumentacija koja bi to potvrdila.

Testiranjem brzine izvođenja korištenjem dvaju DSP procesora dobiveno je približno dvostruko kraće vrijeme izvođenja, što je bilo očekivano, stoga nije potrebno prikazati rezultate za slučaj korištenja dvaju DSP procesora. DSP1 procesor dvostruko je brži od A15 procesora za predviđeni zadatak, što se vidi uspoređivanjem izmjerenog vremena izvođenja za DSP1 procesor i A15 procesor na primjeru metode EBMA i *dummy* nizu okvira prikazanom grafom sa slike 4.24.

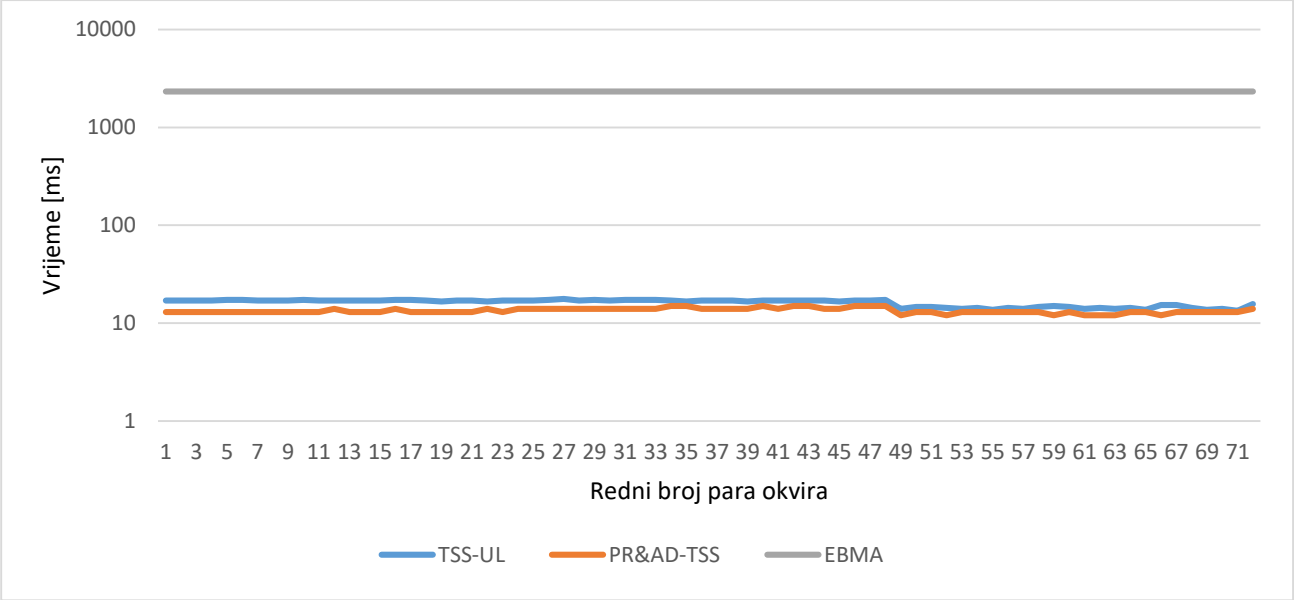


Slika 4.24. *Rezultati mjerenja vremena izvođenja algoritamski optimiziranog rješenja (EBMA metode) za pronalazak VP, implementiranog na Alpha ploču korištenjem DSP1 te A15 procesora, na 72 para okvira dummy niza, rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike.*

4.4.4. Optimizirano rješenje raspodijeljeno na procesore Alpha ploče (DSP1, DSP2 i A15)

Kako je i navedeno u opisu implementiranog rješenja za Alpha ploču, procesori DSP1, DSP2 i A15 koriste se povezani u paralelu. DSP1 i DSP2 rade pretragu VP na tri osmine okvira pojedinačno – zajedno tri četvrtine okvira, dok A15 procesor VP traži na jednoj četvrtini okvira. Ovakav raspored zadataka proizlazi iz izmjerenog vremena izvođenja preliminarnog niza okvira. Raspored zadataka nemoguće je idealno raspodijeliti na procesore jer frekvencija rada procesora varira, ali i zbog ovisnosti vremena izvođenja pronalaska VP o sadržaju okvira (sadržaj nikada nije isti na cijelom području okvira). S obzirom da je mjerenje vremena izvođenja provedeno na

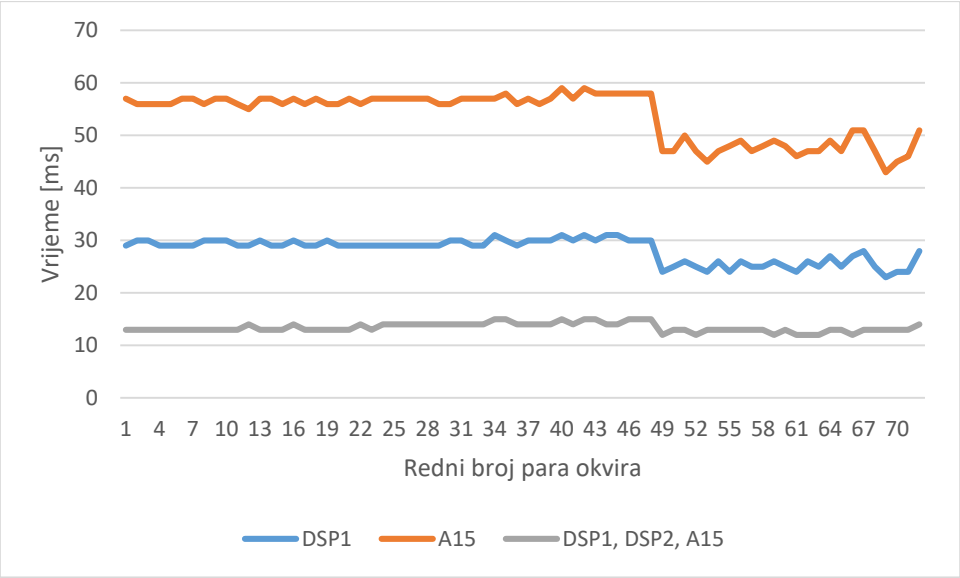
svakom procesoru koji je povezan u paralelu, kod prikaza rezultata mjerenja, koristit će se najveće vrijeme trajanja jer toliko će se ukupno čekati na dovršetak pretrage VP. Primjer izmjerenog vremena izvođenja implementiranog rješenja s raspodijeljenim zadacima na tri procesora Alpha ploče, na okvirima rezolucije 1280x720 elemenata slike i makroblokom 32x32 elementa slike, prikazan je slikom 4.25. za sve 3 predložene metode za pronalazak VP.



Slika 4.25. *Prikaz vremena izvođenja rješenja za pronalazak VP s raspodjelom zadataka na tri procesora Alpha ploče, na 72 para okvira dummy niza rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike.*

Zbog sadržaja srednje četvrtine okvira iz *dummy* niza (područje okvira u kojemu se nalazi objekt koji se kreće), DSP2 procesor ima najviše izračunavanja VP, stoga nema oscilacije vremena ni nakon 49. okvira. Odnos vremena izvođenja pojedine metode ostaje isti kao i kod svakog implementiranog rješenja, što je bilo očekivano. Vremena izvođenja koja su ostvarena raspodjelom zadatak na više procesora, omogućilo je vrijeme izvođenja implementiranih metoda PR&AD-TSS i TSS-UL ispod 20 ms, što omogućava pronalazak VP na video sekvencama uz preko 50 FPS.

Slika 4.26. prikazuje graf vremena izvođenja PR&AD-TSS metode na *dummy* nizu okvira rezolucije 1280x720 elemenata slike i makroblokom 32x32 elementa slike, kojim se želi predstaviti odnos vremena izvođenja metoda za pronalazak VP na procesoru DSP1, na procesoru A15 te na 3 procesora Alpha ploče (DSP1, DSP2 i A15) na koje su raspoređeni zadaci.

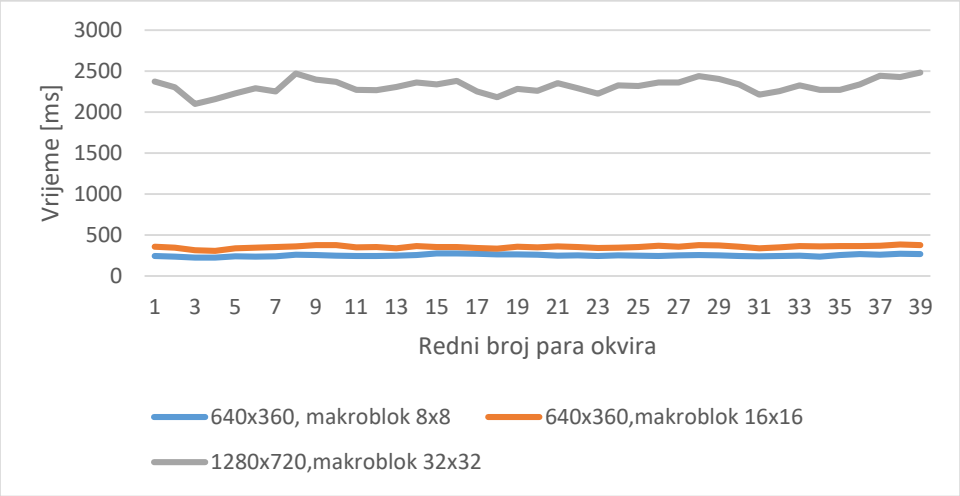


Slika 4.26. *Graf s rezultatima mjerenja vremena izvođenja PR&AD-TSS metode za pronalazak VP na 72 para okvira dummy niza rezolucije 1280x720 elemenata slike uz makroblok 32x32 elementa slike, na tri rješenja implementirana na Alpha ploču (samo DSP1, samo A15, DSP1+DSP2+A15).*

Rasporedom zadataka na tri procesora ostvareno je dvostruko (u usporedbi s rješenjem na DSP1 procesoru) odnosno peterostruko smanjenje ukupnog vremena (u usporedbi s rješenjem na A15 procesoru) što je vidljivo u tablici 4.17. s prikazom srednjih vremena izvođenja na *dummy* testnom nizu uz rezoluciju 1280x720 elemenata slike i makroblokom 32x32 elementa slike. Najbolje poboljšanje rasporeda zadataka na više procesora ostvaruje se na okvirima kod kojih svaka četvrtina područja okvira sadrži objekte koji se kreću jer je na taj način svaki procesor u potpunosti opterećen.

Tablica 4.17. *Srednje vrijeme izvođenja metoda za pronalazak VP implementiranih na Alpha ploču (DSP1, A15 i DSP1+DSP2+A15 na 72 para okvira dummy testnog niza uz rezoluciju 1280x720 elemenata slike i uz makroblok 32x32 elementa slike.*

Ime niza:	dummy niz		
Ime metode:	EBMA	TSS-UL	PR&AD-TSS
Procesor	Srednje vrijeme izvođenja [ms]		
DSP1	5178,58	37,17	28,17
A15	12216,35	72,70	55,35
DSP1+DSP2+A15	2444,50	16,90	13,75



Slika 4.27. *Graf s usporedbom vremena izvođenja metode EBMA za pronalazak VP za niz slijedni auto između niza okvira rezolucije 1280x720 elemenata slike, makroblok 32x32 elementa slika, i niz okvira rezolucije 640x360 elemenata slike, makroblokovi 8x8 i 16x16 elemenata slike..*

Tablica 4.18. *Srednje vrijeme izvođenja metoda za pronalazak VP implementiranih na Alpha ploču (DSP1, A15 i DSP1+DSP2+A15 na 72 para okvira dummy testnog niza uz rezoluciju 1280x720 elemenata slike i makroblokove a) 8x8 i b) 16x16 elementa slike.*

a)

Ime niza:	dummy niz		
Ime metode:	EBMA	TSS-UL	PR&AD-TSS
Makroblok	Srednje vrijeme izvođenja [ms]		
16x16	426,93	4,44	4,67
8x8	561,60	7,85	8,42

b)

Ime niza:	dummy niz		
Ime metode:	EBMA	TSS-UL	PR&AD-TSS
Makroblok	Srednje vrijeme izvođenja [ms]		
16x16	426,93	4,44	4,67
8x8	561,60	7,85	8,42

Kod smanjenja rezolucije okvira s 1280x720 elemenata slike i makroblokom 32x32 elementa slike na rezoluciju 640x360 elemenata slike i makroblokovima 16x16 i 8x8 elemenata slike, ostvareno je smanjenje vremena izvođenja analogno rješenju na računalu ili na bilo kojem pojedinačnom procesoru. Usporedba vremena izvođenja za obje rezolucije i sve tri veličine makroblokova vidljiva je na grafu sa slike 4.27.

Pretraga VP na okvirima manje rezolucije je očekivano brža jer je manji broj elemenata u memoriji s kojim se nešto treba izračunavati. Kod rezolucije 640x360 elemenata slike ostvareno

je niže vrijeme izvođenja u slučaju manjeg makrobloka (8x8 elemenata slike) jer je svaka usporedba dva makrobloka imala manji broj iteracija tijekom izračuna mjere razlike, kao i kod mjere uniformnosti što je vidljivo u tablici 4.18.

4.5. Osvrt na dobivene rezultate

Nakon provedenog testiranja na cjelokupnom skupu testnih nizova okvira na Alpha ploči ostvareni su zadovoljavajući rezultati za pronalazak VP te djelomično zadovoljavajući za njihovo grupiranje. U ovome podpoglavlju kritički će se komentirati dobiveni rezultati s osvrtom na moguća poboljšanja implementiranog rješenja.

4.5.1. Osvrt na rezultate testiranja točnosti metoda za pronalazak VP

Za metodu EBMA i metodu TSS-UL, primjećuje se smanjenje točnosti kod pomaka objekta koji izlaze iz njihovog dosega pretrage. Takav problem moguće je umanjiti korištenjem većeg područja pretrage kod metode EBMA te većeg inicijalnog koraka pretrage kod metode TSS-UL. Kod navedenih metoda točnost opada kod okvira manje rezolucije zbog prirode nastanka takvih okvira – nastali su korištenjem bilinearne interpolacije koja je numerička metoda i unosi pogrešku u vrijednostima elemenata slike. Problem numerički netočnih VP kod takvih okvira može se smanjiti izravnom izradom okvira takve veličine u programskom jeziku Python. Sve metode u slučaju velikog objekta pronalaze mal broj VP. Kako je spomenuto u dijelu 4.2.1., testiranja točnosti metode EBMA, navedeni problem proizlazi iz velikih područja uniformnosti na takvim objektima. Smanjenje granice mjere uniformnosti bi povećalo broj pronađenih VP, no ovakav rezultat je zadovoljavajući jer se generalno i dalje pronalazi objekt koji se kreće te nije potrebno mijenjati granice.

4.5.2. Osvrt na rezultate testiranje točnosti metode grupiranja VP

Dana implementacija grupiranja VP dala je zadovoljavajuće rezultate na nizovima snimljenim korištenjem mirne kamere. Za sekvence snimljene u pokretu, ostvarena je nezadovoljavajuća točnost grupiranja vektora pokreta na danom rješenju. Takav rezultat proizlazi iz povećanja raspršenosti karakteristika kuta vektora pokreta kod takvih okvira. Zbog kretanja kamere, prostorno susjedni vektori imaju veće razlike u veličini kuta, zbog čega metoda grupiranja k srednjim vrijednostima u kombinaciji s predloženim tokom grupiranja, daje nepouzdane rezultate. Osim navedenih razloga, nepouzdanost rješenja grupiranja VP je velika i zbog prirode nastanka testnih sekvenci kamere koja se kreće. Kamera koja se koristila sprema videozapise u

MP4 formatu. Taj format koristi razne metode za kompresiju videozapisa, zbog čega se stvaran sadržaj okvira gubi, odnosno dobivene vrijednosti elemenata slike su izračunate, a ne one stvarne (nisu izravno dobivene sa senzora kamere). Metode za pronalazak VP zbog toga daju manje točne VP jer ovise o točnim vrijednostima elemenata slike.

4.5.3. Osvrt na rezultate testiranja brzine izvođenja implementiranog rješenja na Alpha ploču

Optimiziranjem rješenja i raspodjelom na 3 procesora Alpha ploče, ostvareno je vrijeme izvođenja koje za PR&AD-TSS metodu i metodu TSS-UL omogućava obradu video okvira od preko 50 okvira u sekundi. Potencijalno je moguće ostvariti bolje rezultate daljnjom raspodjelom pretrage na više SOC-ova Alpha ploče, što je u implementaciji ovog rada izostavljeno zbog manjka dokumentacije o PCIe komunikaciji SOC-ova Alpha ploče. Komunikaciju je moguće ostvariti i Ethernet sučeljem na ploči, međutim ono je ograničeno ukupnom mogućom količinom video okvira u memoriji koji mogu biti obrađeni u sekundi te ne bi pospješilo rezultate testiranja brzine izvođenja.

5. ZAKLJUČAK

Povećanje opremljenosti vozila računalima vodi prema sve većoj rezini autonomnosti vozila. Za takav oblik vožnje potrebno je prikupljanje i obrada velikih količina podataka iz okolice vozila korištenjem senzora. ADAS platforme trebaju obavljati zadatak obrade takvih informacija s najvećom preciznošću i velikom brzinom ne bi li se postigla autonomnost. Osnovno shvaćanje okoline iz perspektive računala zasnovano je na detekciji objekata u okolini i njihovom praćenju. Rad se bavi metodama za detekciju i praćenje objekata korištenjem metode pronalaska vektora pokreta između dvaju susjednih okvira te grupiranjem istih prema sličnosti. U okviru ovoga rada predložena su tri rješenja koja omogućuju odabir između točnosti i brzine izvođenja. EBMA omogućava veću točnost, dok metode TSS-UL i PR&AD-TSS omogućavaju povećanje brzine izvođenja uz nešto nižu točnost pronađenih vektora pokreta. Najveće povećanje brzine izvođenja metoda je postignuto optimiziranjem koda korištenjem znanja o *cache* memoriji te smanjivanjem područja pretrage na jednom okviru, zasnovano na prirodi sadržaja prikupljenih okvira. Navedenom optimizacijom programskog rješenja ostvarena je obrada od preko 50 FPS. Testiranje programskog rješenja dalo je zadovoljavajuću točnost svih metoda za obje grupe sekvenci, rezolucije 1280x720 i 640x360 elemenata slike. Grupiranje vektora pokreta dalo je pouzdane rezultate za video sekvence u kojima kamera miruje, ali nepouzdane za sekvence s kamerom koja se kreće. Takvi rezultati proizlaze iz prirode nastanka video sekvenci korištenih kod testiranja i nepouzdanosti predloženog toka programskog rješenja za grupiranje vektora pokreta na takvim sekvencama. Moguće je ostvariti poboljšanje koristeći neke naprednije tehnike grupiranja vektora pokreta te korištenjem kamere koja sprema video okvire u sirovom formatu. Alpha ploča nije mogla biti korištena kod prikupljanja testnih sekvenci jer je protok podataka na Ethernet sučelju ograničen te ne može ostvariti potrebnu količinu okvira u sekundi (maksimalno moguće je 10 okvira u sekundi).

LITERATURA

[1] Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, SAE International, 2021., dostupno na: https://www.sae.org/standards/content/j3016_202104

[2] Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems, ISO/IEC , 2015., dostupno na: https://www.iso.org/standard/67331.html

[3] S. S. M. AlDabbagh, A. H. Mabrouk, I. F. T. Al Shaikhli, An improved full search block matching algorithm for imaging applications, IEEE, 20.9.2012.god. dostupno na: https://ieeexplore.ieee.org/document/6271172

[4] Predictive Three Step Search (PTSS) algorithm for motion estimation, Research gate, 2013, dostupno na:https://www.researchgate.net/publication/255950649_Predictive_Three_Step_Search_PTSS_algorithm_for_motion_estimation

[5] C. M. Wu, J. Y. Huang, A new block matching algorithm for motion estimation, Trans Tech Publications Ltd, Switzerland, 2016, dostupno na: https://www.scientific.net/AMM.855.178

[6] H. Amirpour, M. Ghanbari, A. Pinheiro, M. Pereira, Motion estimation with chessboard pattern prediction strategy, Springer Science, 2019, dostupno na: https://www.researchgate.net/publication/332197238_Motion_estimation_with_chessboard_pattern_prediction_strategy

[7] M. Ghoneim, N. Tsumura, T. Nakaguchi, T. Yahagi, Y. Miyake, A fast block matching algorithm based on motion vector correlation and integral projections, IEICE Transactions on Information and Systems, 2009, dostupno na: https://www.semanticscholar.org/paper/A-Fast-Block-Matching-Algorithm-Based-on-Motion-and-Ghoneim-Tsumura/04ffe96b009b13e938a3d382999fa92eb4052a37

[8] M. R. M. Assis, B. Muller, Multithread identification of objects flow, Research Gate, 2015., dostupno na: https://www.researchgate.net/publication/38115189_MULTITHREAD_INDETIFICATION_OF_OBJECTS_FLOW_APPROACH_BY_GROUPING_MOTION_VECTORS_AND_MODELING_BACKGROUND

[9] S. Na, L. Xumin, G. Yong Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm, IEEE, 2010., dostupno na: https://ieeexplore.ieee.org/document/5453745

[10] „Automotive Machine Vision ALPHA reference board on Texas Instruments SoCs" Dostupno na: https://www.rt-rk.com/download/rt-rk_ALPHA_ADAS_board.pdf (pristupljeno 28. kolovoza 2021.)

[11] “RT-RK - Automotive.” https://www.rt-rk.com/services/automotive (pristupljeno 5. srpnja 2021).

[12] S. Rimas Drlje, M. Vranješ, “Upoznavanje s ADAS razvojnom pločom.” Dostupno na: https://loomen.carnet.hr/pluginfile.php/1583196/mod_resource/content/1/Vjezba_1.pdf.

[13] M. Vranješ, D. Vajak, »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“«.

[14] About OpenCV, [Na internetu]. Dostupno na: https://opencv.org/about/ (pristupljeno srpanj 10., 2021.).

[15] NumPY, [Na internetu] Dostupno na: https://numpy.org/doc/ (pristupljeno srpanj 2021.)

[16] B. Franke, C Compilers and Code Optimization for DSPs, Research Gate, 2013. dostupno na: https://www.researchgate.net/publication/251096215_C_Compilers_and_Code_Optimization_for_DSPs

[17] A.Fog, Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms, [na internetu] dostupno na: https://www.agner.org/optimize/optimizing_cpp.pdf (pristupljeno srpanj 10., 2021.)

[18] “Tera Term - Terminal Emulator for Windows.” [na internetu] dostupno na: https://ttssh2.osdn.jp/ (pristupljeno srpanj 10., 2021).

[19] “FFmpeg.” https://ffmpeg.org/ (pristupljeno srpanj 12., 2021).

SAŽETAK

Ovaj rad se bavi metodama za pronalazak vektora pokreta i njihovim grupiranjem s ciljem detekcije broja i smjera kretanja pokretnih objekata u sceni te implementacijom metoda na ugradbenu ADAS Alpha ploču. Nakon implementacije i optimizacije rješenja u C programskom jeziku na osobnom računalu, odabrane metode su implementirane na ADAS ploču, također korištenjem C programskog jezika. S dovršetkom prve implementacije na Alpha ploču, pažnja je posvećena na raspodjelu poslova na dostupne procesore Alpha ploče. Nakon prikladne raspodjele posla, uspoređene su performanse triju metoda za pretragu vektora pokreta (metoda iscrpne pretrage, metoda tri koraka s mogućnošću povećanja broja koraka te metoda tri koraka s uključenim predviđanjem i prilagodbom na odabir oblika pretrage), prilikom izvođenja na osobnom računalu i na ugradbenoj ADAS Alpha ploči. Rezultati su pokazali da je predložena metoda iscrpne pretrage najtočnija metoda za pronalazak VP, a ostale metode imaju prihvatljivu točnost. Testiranjem rješenja na video signalima iz prirode (s prometnica) uočeno je da je predloženo rješenje za grupiranje VP primjenjivo u slučajevima gdje kamera miruje, dok je za slučaj video signala u kojima se kamera kreće dalo neprihvatljive rezultate točnosti. Rezultati testiranja brzine izvođenja optimiziranog rješenja pokazali su da je moguća obrada uz više od 50 okvira u sekundi za ulazne slike rezolucije 1280x720 elemenata slike.

Ključne riječi: *vektori pokreta, grupiranje vektora pokreta, detekcija objekata, ADAS, VisionSDK*

CALCULATION OF MOTION VECTORS AT THE BLOCK LEVEL AND THEIR GROUPING ACCORDING TO SIMILARITY WITH IMPLEMENTATION ON A REAL ADAS DEVELOPMENT PLATFORM

ABSTRACT

This paper presents methods for the discovery of motion vectors (MV) and their grouping with the goal of detection of number and direction of movable objects contained in the scene as well as the implementation of those methods onto an ADAS Alpha board. After implementing and optimizing the solutions in C programming language on a personal computer, chosen methods were implemented onto an ADAS board using the same programming language. With the completion of the first implementation on the Alpha board, attention was given to the allocation of tasks to the available Alpha board processors. After appropriate task allocation, the performance of three motion vector search methods (exhaustive search method, three-step search method with the possibility of increasing the number of steps, and three-step method with prediction and adaptation of search form selection) was compared both on a personal computer as well as on the ADAS Alpha board. The results have shown that the exhaustive search method is the most accurate method for finding MV with other methods having acceptable accuracy. Testing the solutions on real-life (taken from roads) video signals have shown that that the proposed solution for grouping MV is applicable in cases where the camera is stationary while it gave unacceptable accuracy results in cases of video signals where the camera is moving. The results of testing the execution speed of an optimized solution have shown that processing is possible with more than 50 frames per second for input images with a resolution of 1280x720 pixels.

Key words: *motion vectors, motion vectors grouping, object detection, ADAS, VisionSDK*

ŽIVOTOPIS

Valentin Radičević rođen je u Slavonskom Brodu 7.travnja 1997. Pohađao je Osnovnu školu „Stjepan Radić“ u Oprisavcima. Nakon završene osnovne škole upisuje Gimnaziju „Matija Mesić“ u Slavonskome Brodu, smjer Opća gimnazija. Gimnaziju završava 2016. godine, nakon čega iste godine upisuje Preddiplomski sveučilišni studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, tadašnji Elektrotehnički fakultet, na Sveučilištu J.J. Strossmayera. Nakon završetka preddiplomskog studija elektrotehnike, 2019. godine stječe akademski naziv sveučilišni prvostupnik (lat. baccalaureus) inženjer elektrotehnike. Iste godine upisuje diplomski sveučilišni studij Elektrotehnike, smjer komunikacije i informatika, izborni blok Mrežne tehnologije u Osijeku.

_____

Potpis autora

PRILOZI

P.3.1. Programski kod rješenja na osobnom računalu za sve 3 metode pretrage VP i njihovo grupiranje (elektronički prilog)

P.3.2. Kod skripte za iscrtavanje vektora na sliku korištenjem Python programskog jezika (elektronički prilog)

P.4.1. Kompletne video sekvence i svi parovi okvira izdvojeni iz njih za potrebe testiranja (elektronički prilog)

P.4.2. Testni nizovi okvira s iscrtanim rezultatima pretrage i grupiranja (elektronički prilog)