Analiza videozapisa primjenom tehnika računarstva visokih performansi

Rubinić, Ivan

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski studij računarstva

Završni rad

Analiza videozapisa primjenom tehnika računarstva visokih performansi

Rijeka, rujan 2021.

Ivan Rubinić
0069085793

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski studij računarstva

Završni rad

Analiza videozapisa primjenom tehnika računarstva visokih performansi

Mentor: doc. dr. sc. Goran Mauša

Rijeka, rujan 2021.

Ivan Rubinić
0069085793

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 12. ožujka 2021.

Zavod: **Zavod za računarstvo**
Predmet: **Uvod u objektno orijentirano programiranje**
Grana: **2.09.02 informacijski sustavi**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Ivan Rubinić (0069085793)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Analiza videozapisa primjenom tehnika računarstva visokih performansi / Video-analysis using high-performance computing techniques**

Opis zadatka:

Proučiti mogućnosti i način rada knjižnice programskih funkcija OpenCV. Osmisliti i predložiti vlastito rješenje za nadzor prostorija i analizu dostupnih nadzornih snimki korištenjem tehnika računarstva visokih performansi. Usporediti pojedine inačice rješenja i objasniti prednosti računarstva visokih performansi na osmišljenom rješenju. Navesti i objasniti parametre po kojima su pojedine inačice uspoređene i ocijenjene.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 15. ožujka 2021.

Mentor:

Doc. Goran Mauša, dipl. ing.

Predsjednik povjerenstva za završni ispit:

Izv. prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2021.

_____

Ivan Rubinić

Zahvala

Zahvaljujem svojem mentoru doc. dr. sc. Goranu Mauši na korisnim uputama i savjetima, brojnim sastancima i pruženoj podršci tijekom pisanja ovoga rada. Također zahvaljujem mag. ing. mech. Anti Sikirici na savjetima prilikom konfiguracije i pokretanja programa na superračunalu BURA.

Želim se zahvaliti Dinu i Kristijanu na velikom trudu, pruženoj pomoći, znanju i savjetima kroz sve naše godine prijateljstva.

Posebno sam zahvalan svojim roditeljima i bratu na bezuvjetnoj podršci, razumijevanju i ljubavi.

Sadržaj

Popis slika **viii**

Popis tablica **xi**

1 Uvod **1**

1.1 Računarstvo visokih performansi 1

1.1.1 Usporedba: osobna računala i superračunala 2

1.1.2 Svrha računarstva visokih performansi 3

1.2 Alati 7

1.2.1 OpenCV 7

1.2.2 CUDA 8

1.2.3 Python 10

1.2.4 Arduino 11

1.3 Cilj završnoga rada 12

2 Aplikacija za detekciju osoba **14**

2.1 Odabir algoritma za analizu videozapisa 14

2.1.1 Konačni odabir - histogram orijentiranih gradijenata 18

2.2 Različiti pristupi 25

2.2.1 Prva inačica aplikacije 25

2.2.2 Druga inačica aplikacije - ubrzanje grafičkom karticom 32

3 Aplikacija za regresijsku analizu koordinata detekcija 40

3.1 Regresijska analiza 40

3.1.1 Potreba i cilj regresijske analize kod detekcije osoba 41

3.2 Programska podrška za analizu koordinata 44

3.2.1 Opis rada programa 45

3.2.2 Input i output 45

3.3 Primjeri ulazne datoteke, izlazne datoteke i animacije 46

4 Analiza inačica softverskih rješenja 49

4.1 Korištenje dostupnih resursa 49

4.1.1 Procesor i grafička kartica 49

4.1.2 Radna memorija 51

4.2 Brzina obrade kadrova 55

4.3 Energetska učinkovitost 59

4.3.1 Usporedba provedenih mjerenja 59

5 Pokretanje detekcije na superračunalu BURA 68

5.1 Pokretanje programske podrške na superračunalu 68

5.2 Usporedba superračunala i osobnog računala 71

6 Zaključak 75

Bibliografija 76

Sažetak 79

A Programski kôd klase Args druge inačice rješenja 80

Popis slika

1.1 Ubrzanje analize koristeći podršku grafičke kartice. Dijagram prikazuje ubrzanja za različite ulazne vrijednosti *Re* uz implementaciju putem 400, 635 i 900 dretvi na grafičkoj kartici. 10

1.2 Prikaz regresijske funkcije koja predstavlja nejučestalije kretnje detektiranih osoba (detekcije predstavljene plavim točkama). 11

1.3 Prikaz čitanja i vizualiziranja informacija o potrošnji energije s vanjskog sklopovlja (ordinata predstavlja potrošnju u Watt-ima, a apscisa vrijeme u sekundama od početka čitanja podataka). 12

2.1 Reprezentacija izračunatih magnituda i smjerova gradijenata. 19

2.2 Primjer pripreme kadra. Slike nisu prikazane u točnoj veličini zbog rasporeda na stranici, ali su točne dimenzije navedene u opisima. . . 21

2.3 Prikaz magnituda gradijenata. Možemo primjetiti da horizontalni gradijent ističe vertikalne rubove, a vertikalni horizontalne rubove. . 22

2.4 Prikaz podjele sekcije na ćelije dimenzija 8x8 piksela 23

2.5 Prikaz izvođenja prve inačice aplikacije. 30

2.6 Prikaz problema prilikom prikazivanja putanja u prvoj inačici aplikacije. 31

2.7 Prikaz iskorištavanja procesorskih resursa u prvoj inačici aplikacije. . 32

2.8 Primjer dopuštenih (zeleni pravokutnici) i nedopuštenih (crveni pravokutnici) detekcija na temelju njihovog položaja. 39

Popis slika

3.1 Prikaz zavisnosti parametara. Plave točke predstavljaju položaje svih detekcija unutar prozora visine y-osi i širine x-osi. Crvena krivulja predstavlja dobivenu regresijsku funkciju koja predstavlja položaje svih detektiranih osoba. 42

3.2 Prikaz stupnja pripadnosti položaja prosječnoj putanji detekcija. . . 44

3.3 Prikaz primjera ulazne datoteke aplikacije za regresijsku analizu koordinata detekcija. 47

3.4 Prikaz primjera izlazne datoteke aplikacije za regresijsku analizu koordinata detekcija. 47

3.5 Prikaz animacije za regresijsku analizu na prvih 175, 350 i 525 koordinata položaja detekcija osoba. 48

4.1 Postotak korištenja kapaciteta procesora računala kroz vrijeme od strane aplikacija za detekciju osoba. 52

4.2 Utrošak vremena na procese različitih kategorija prilikom izvađanja aplikacija za detekciju osoba. 53

4.3 Količina korištene radne memorije računala kroz vrijeme od strane aplikacija za detekciju osoba. 54

4.4 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba. 57

4.5 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba (smanjena veličina kadrova tijekom procesa pripreme). 58

4.6 Prikaz rasporeda komponenata alata za mjerenje potrošnje električne energije. 60

4.7 Prikaz alata za mjerenje potrošnje električne energije. 62

4.8 Potrebna snaga od strane računala prije i tijekom rada pojedinih inačica aplikacije za detekciju osoba. 63

4.9 Prikaz potrebe za snagom kroz učestalost pojedinih vrijednosti za različite inačice aplikacije za detekciju osoba. 64

4.10 Prikaz odnosa kumulativne potrebne snage i kumulativnog izvršenog posla kroz vrijeme za različite inačice aplikacije. 65

4.11 Prikaz odnosa potrošnje energije pomoću integrala snage kroz vrijeme za prvu inačicu aplikacije. 66

4.12 Prikaz odnosa potrošnje energije pomoću integrala snage kroz vrijeme za drugu inačicu aplikacije. 67

5.1 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba na superračunalu i osobnom računalu. 73

5.2 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba na superračunalu i osobnom računalu (smanjena veličina kadrova tijekom procesa pripreme). 74

Popis tablica

1.1 Prikaz najznačajnijih karakteristika superračunala, preuzeto iz [1] . . 4

1.2 Prikaz karakteristika osobnog računala na kojem je napisan ovaj završni rad. 4

1.3 Prikaz problema iz akademske zajednice, industrije i državne vlasti koji primjenjuju računarstvo visokih performansi, prilagođeno iz [2]. 6

Popis prikaza programskog kôda

2.1 Klasa Detector 27
2.2 Uvodne radnje unutar main funkcije 28
5.1 SLURM skripta za pokretanje programske podrške. 70
A.1 Programski kôd klase Args druge inačice rješenja 80

Poglavlje 1

Uvod

Zadatak završnog rada je osmisliti, izraditi i analizirati vlastito rješenje za nadzor prostorija i analizu videozapisa koristeći tehnike računarstva visokih performansi. Najveći izazov pri razvoju spomenutog rješenja je detekcija osoba s dostupnih videozapisa. Problemi s kojima se susrećemo prilikom analize kadrova videozapisa su međusobno različiti izgledi osoba, pregršt različitih ljudskih poza, šum pozadine, nedostatak svjetla, pozicija kamere i slično [3]. Pri detekciji također treba obratiti pozornost na korištenje tehnika računarstva visokih performansi, a ne smije se zanemariti niti važnost provođenja kvalitetne analize razvijene programske podrške.

U sljedećih nekoliko potpoglavlja bit će objašnjen pojam računarstva visokih performansi, cilj završnoga rada kao i alati koji su korišteni prilikom izrade istog. Opis razvoja zadatka završnog rada, njegova dodatna programska podrška, analiza razvijenog rješenja te implementacija istog na superračunalu opisani su u sljedećim poglavljima.

1.1 Računarstvo visokih performansi

Razvojem računarstva pojavljuju se sve kompliciraniji izazovi čija se rješenja temelje na algoritmima visoke složenosti koji iziskuju sve veću računsku snagu i količinu vremena kako bi se izvršili. Sama povijest razvoja računarske znanosti nam to prikazuje na jednostavan način – dok su računala danas u stanju izvršiti impresivan spektar

zadataka, u početku su služila isključivo kao pomoć pri izračunu osnovnih aritmetičkih operacija. Upravo takva, jednostavna računala su zaslužna za pokretanje razvoja računarstva i pojavu sve složenijih problema čijim rješavanjem dolazimo do sve korisnijeg sklopovlja i programske podrške koja nam danas pomaže u rješavanju nekada nezamislivih zadataka.

Naravno, neka pitanja današnjice je vrlo teško riješiti bez primjene računarstva visokih performansi (engl. *High Performance Computing*) koje nam omogućuje korištenje velike računalne moći za rješavanje zahtjevnih problema. Tipični zadaci računarstva visokih performansi najčešće podrazumijevaju analizu velike količine podataka i izradu raznih simulacija u najkraćem mogućem vremenskom razdoblju. Pri tome se koriste superračunala čiji su resursi raspodijeljeni u više zasebnih računalnih odjeljenja što sustavu daje mogućnost paralelizacije, fleksibilnosti i preciznog dodjeljivanja resursa među svojim korisnicima.

1.1.1 Usporedba: osobna računala i superračunala

Prilikom primjene tehnika računarstva visokih performansi koriste se superračunala. Iako se razlikuju po svojim zapanjujućim performansama, veličini, težini i troškovima održavanja, superračunala ipak dijele priličan broj karakteristika s osobnim računalima.

Superračunalni sustavi s osobnim računalima, radnim stanicama i poslovnim poslužiteljima dijele uobičajene značajke koje karakteriziraju svako računalo [2]:

- vrijednosti ulaznih podataka se transformiraju u izlazne rezultate,
- unutarnja memorija pohranjuje podatke pomoću kojih i s kojima sustav radi,
- sustav sadrži komunikacijske kanale putem kojih se prenose informacije između njegovih dijelova i podsustava tijekom izvađanja aplikacija,
- sustav sadrži kontrolno sklopovlje koje koordinira rad njegovih sastavnih dijelova i podsustava,
- sustav sadrži sklopovlje za pohranu trajnih podataka i programske podrške,
- korisnici upravljaju sustavom pomoću ulazno-izlaznih kanala i sučelja (npr.

tipkovnica, ekran...),

- operacijski sustav nadzire rad svih dijelova računala,
- kompajleri prevode programski kôd u binarni kôd,
- datotečni sustav na apstraktan način reprezentiraja podatke dostupne računalu te organizira podatke na svim dostupnim medijima.

Bez obzira na popriličan broj sličnosti, potpuno očekivano, postoji velika razlika između osobnih računala i superračunala [2]. Kao najvažnija razlika ističe se organizacija, međupovezanost i razmjer računarskih resursa superračunala u odnosu na osobna računala, ali i odgovarajuća programska podrška koja na što učinkovitiji način treba upravljati dostupnim resursima. Učinkovito upravljanje uključuje koordinaciju paralelizacije među svim komponentama superračunala prilikom istovremenog izvršavanja svih zadanih zadataka. Iako se koordinacija mora odvijati na svakom modernom višejezgrenom računalu, superračunala tu zadaću obavljaju između mnogo više dostupnih resursa i zadanih zadataka te upravo u toj činjenici uočavamo znatnu razliku. Usporedba dostupnih resursa kod superračunala i osobnog računala na kojem je pisan ovaj završni rad je dana u tablicama 1.1 (stranica 4) i 1.2 (stranica 4). Nadalje, razlika se može uočiti i sa stanovišta programera koji osim razmišljanja o paralelizaciji (prisutno i na osobnim računalima) mora razmišljati i o distribuciji. Dakle, programer tijekom razvijanja programa mora u obzir uzeti činjenicu da se na superračunalu instrukcije mogu odvijati na značajnim fizičkim udaljenostima koja je kod osobnih računala zanemariva. Zbog navedenog, programer mora posjedovati znanje i vještine koje će mu omogućiti iskorištenje cjelokupnog potencijala superračunala kako bi se izračuni proveli na optimalan način.

1.1.2 Svrha računarstva visokih performansi

Prava važnost računarstva visokih performansi primjećuje se tek kada se odgovori na pitanja ne mogu odgonetnuti eksperimentima (empirijskim postupcima), proračunima (teorijskim postupcima) ili osobnim računalom [2].

Primjerice, tok vode kroz riječno korito u slučaju poplave je vrlo teško predvidjeti eksperimentom, "ručnim" proračunom ili osobnim računalom. Ukoliko bismo

Tablica 1.1 Prikaz najznačajnijih karakteristika superračunala, preuzeto iz [1]

Rang u svijetu (11/2021)	**Superračunalni sustav**	**Broj procesorskih jezgri**	**Maksimalne teoretske performanse (TFlop/s)**	**Radna memorija (GB)**	**Potrošnja električne energije (kW)**
1	Fugaku (Japan)	7 630 848	537212.0	5 460 620	29 899
2	Summit (SAD)	2 414 592	200794.9	2 848 520	10 096
3	Sierra (SAD)	1 572 480	125712.0	2 702 160	7 438
4	Sunway TaihuLight (Kina)	10 649 600	125435.9	1 320 000	15 371
5	Selene (SAD)	555 520	79215.0	615 720	2 646
...	...	...	...	...	...
?	Bura (Hrvatska)	6912	287.5	12 000	240

Tablica 1.2 Prikaz karakteristika osobnog računala na kojem je napisan ovaj završni rad.

Broj procesorskih jezgri	**Maksimalne teoretske performanse (TFlop/s)**	**Radna memorija (GB)**	**Potrošnja električne energije (kW)**
6	0.0179	16	0.3

poplavu željeli simulirati eksperimentom, ovisno o veličini poplave i riječnog korita, bila bi potrebna ogromna količina novčanih i ljudskih resursa za provedbu pokusa. Uz to, postojala bi mogućnost stvarne štete nastale kao posljedica poplave. Nadalje, stvaranje predviđanja u slučaju poplave temeljem "ručnog" proračuna bi se zasigurno pokazalo izuzetno zahtjevnim i vjerojatno nemogućim. U postupak izračuna morali bismo uvrstiti količinu vode, širinu, duljinu, dubinu i oblik riječnog korita, moguće vanjske utjecaje i sl. Sve parametre bismo tada morali podložiti fizikalnim zakonima na kojima bi se temeljili eventualni rezultati i predikcije. Naposlijetku, unos svih navedenih parametara i fizikalnih zakona u simulacijski program pokrenut na osobnom računalu poboljšava situaciju i olakšava izračun jedino ukoliko se možemo zadovoljiti približnim, nepreciznim zaključcima.

S druge strane, primjenom računarstva visokih performansi, odnosno korištenjem superračunala, simulacija poplave je u potpunosti realan i izvediv zadatak. Super-

računalo je zbog svoje ogromne računalne moći u stanju uzeti u obzir sve dane parametre (količina vode, širina, duljina, dubina i oblik riječnog korita...) i njihovo međudjelovanje, modelirati ih fizikalnim zakonima te proizvesti rezultate koji se mogu prikazati na čovjeku razumljiv način (dijagrami, slikovne reprezentacije, videozapisi i sl.).

Naravno, primjer simulacije poplave je samo jedan specifičan problem rješiv primjenom superračunala na kojem sam htio ilustrirati svrhu i važnost računarstva visokih performansi. Lista rješivih problema je iznimno duga, a može se prikazati na raznim problemima iz akademske zajednice, industrije i državne vlasti za čije je rješavanje zaslužno upravo računarstvo visokih performansi [2]. Prikaz je dan u tablici 1.3 na stranici 6.

Tablica 1.3 Prikaz problema iz akademske zajednice, industrije i državne vlasti koji primjenjuju računarstvo visokih performansi, prilagođeno iz [2].

Izračun koji se provodi korištenjem računarstva visokih performansi	**Akademska zajednica**	**Industrija**	**Državna vlast**
Rješavanje parcijalnih diferencijalnih jednadžbi	Navier-Stokes jednadžbe (gibanje viskoznih fluida), Einsteinove jednadžbe (dinamika prostor-vremena), Maxwellove jednadžbe (elektromagnetizam)	Black-Scholesov model (dinamika tržišta), Navier-Stokes jednadžbe (npr. modeliranje spremnika za ulje)	Vremenska prognoza, predviđanje kretanja uragana
Rješavanje velikih sustava s međudjelovanjima sila	Kozmologija, simulacija međumolekularne interakcije	Pridonosi napredcima u medicini, biomolekularna interakcija	Modeliranje plazme
Rješavanje problema linearne algebre	Služi kao podrška za rješavanje parcijalnih diferencijalnih jednadžbi i mjerenje performansi sustava	PageRank (alogiram iza Google pretraga)	Modeliranje klime, evaluacija superračunala
Rješavanje problema zadanih grafovima	Strojno učenje	Otkrivanje prevare	Sigurnosni servisi, analiza podataka
Rješavanje stohastičkih sustava	Fizika čestica	Analiza rizika u polju financija, dizajn nuklearnog reaktora	Javno zdravstvo - modeliranje širenja bolesti

1.2 Alati

1.2.1 OpenCV

OpenCV (engl. *Open Computer Vision*) je korišten pri rješavanju problema danog u tekstu zadatka završnog rada. Služila je za detektiranje osoba na slici prema danim parametrima te za prikaz osoba koje se nalaze izvan zadane tolerancije najčešće kretnje ljudi.

Inače, OpenCV je knjižnica programskih funkcija otvorenog koda koja pruža rješenja iz područja računalnog vida i strojnog učenja. OpenCV knjižnica sadržava više od 2500 optimiziranih algoritama – od onih klasičnih do onih najnovijih i najmodernijih [4]. Algoritmi uključeni u knjižnicu korisnicima služe kao olakšanje prilikom razvoja programa za detekciju i prepoznavanje lica, objekata, ljudskih gesta, kombiniranja različitih slika iste scene, [5] provjeru industrijskih proizvoda, medicinske slike, sigurnost, robotiku itd. Tijekom dizajniranja OpenCV-a, velika pažnja pridana je njegovoj računskoj efikasnosti. Gotovo svi podržani algoritmi podrazumijevaju višejezgreni rad te neka omogućuju i ubrzanje izvođenja korištenjem grafičke kartice što je moj posao prilikom izrade rješenja zadatka završnog rada uvelike olakšalo.

Raširenoj upotrebi OpenCV-a zasigurno pridonosi činjenica da je kompatibilan s Windows, Linux, Mac OS i Android operacijskim sustavima. Također, moguće ga je koristiti u kombinaciji s nekoliko različitih programskih jezika.

Za što se koristi OpenCV?

Većina ljudi nije svjesna važnosti računalnog vida, a samim time i OpenCV-a u modernoj svakodnevici [5]. Uz standardne primjene kod nadzora prostorija, prikazivanja slika i videozapisa na internetu, autonomne vožnje ili biomedicinske analize, OpenCV se koristi na puno širem spektru problema vezanih za računalni vid.

Primjerice algoritmi za spajanje pogleda i kalibraciju kamere omogućuju virtualne šetnje ulicama i gradovima cijelog svijeta [5]. Također, gotovo svaki proizvod nastao procesom masovne indurstrijske proizvodnje se u nekoliko točaka svog nastajanja automatski provjerava primjenom računalnog vida, odnosno primjenom algoritama

koje OpenCV implementira. Treba imati na umu da su ovo samo jedni u nizu od ogromnog broja primjena računalnog vida i OpenCV-a.

Zanimljivo je navesti i neke utjecajne, dobro poznate tvrtke koje koriste mogućnosti koje im nudi spomenuta knjižnica programskih fukcija. OpenCV spominje Microsoft, IBM, Yahoo, Google, Intel i Hondu kao jedne od najvećih korisnika svojeg proizvoda [4].

1.2.2 CUDA

CUDA (engl. *Compute Unified Device Architecture*) je platforma za paralelizaciju i model programiranja za opće izračune koristeći grafičku karticu koju je razvila NVIDIA [6]. Glavna prednost kod korištenja CUDA-e je mogućnost izvršavanja računski zahtjevnih dijelova izračuna koji se mogu paralelizirati na grafičkoj kartici. Na taj način možemo iskoristiti tisuće raspoloživih jezgri grafičke kartice i uvelike ubrzati izvođenje svog programa. Naravno, standardni, slijedni dio programa se još uvijek izvršava na glavnom procesoru računala koji upravlja prijenosom informacija te zadavanjem naredbi prema grafičkoj kartici.

Treba napomenuti – iako CUDA može biti od velike pomoći, treba obratiti pozornost na isplativost korištenja CUDA-e na specifičnom problemu. Algoritam se mora moći paralelizirati na način da se izvodi kroz nekoliko stotina, čak i tisuća dretvi kako bismo uspjeli iskoristiti mogućnosti CUDA-e [7]. S druge strane, ukoliko je algoritam uglavnom slijedne prirode, vjerojatno je da CUDA neće donijeti značajna ubrzanja. Također, potrebno je obratiti pažnju i na učestalost te opseg slanja i primanja podataka na i s grafičke kartice. Svaki prijenos podataka predstavlja izostanak korisnih izračuna dok se čeka na izvršenje operacije (engl. *overhead*).

Za što se koristi CUDA?

Primjenom CUDA-e mnogobrojna postojeća programska rješenja koja se već smatraju iznimno dobro implementiranima otkrivaju ogromna poboljšanja [8]. Iako se najčešće koncentriramo isključivo na performanse, odnosno brzinu programskog rješenja, poboljšanja prilikom primjene ubrzanja grafičkom karticom uključuju i sma-

njenje potrošnje energije bez kompromitiranja brzine. U nastavku su dane neke situacije u kojima je moguće primjeniti CUDA-u i ostvariti značajna unapređenja [8].

Prvi se primjer tiče primjene u medicini. Opće je poznato da, iako je potrebno za određene medicinske preglede, rendgensko zračenje šteti ljudskom organizmu. U pokušaju eliminacije rendgenskog zračenja kod otkrivanja nekih vrsta raka i zamjene istoga ultrazvukom, CUDA je odigrala važnu ulogu. Naime, pregled ultrazvukom koristeći trodimenzionalnu tehniku na početku svog razvoja pokazao se prezahtjevnim za osobna računala u smislu izvođenja svih potrebnih izračuna u razumnom vremenu prilikom stvaranja trodimenzionalnih vizualizacija na temelju prikupljenih podataka [8]. Prilagodba programskog rješenja na način da se koriste prednosti CUDA-e omogućila je pretvorbu 35GB podataka s 15-minutnog pregleda u potpuno prihvatljivom vremenskom razdoblju od 20 minuta. Naravno, treba napomenuti da je navedeni primjer opisan 2010. godine te da su danas poboljšanja zasigurno veća.

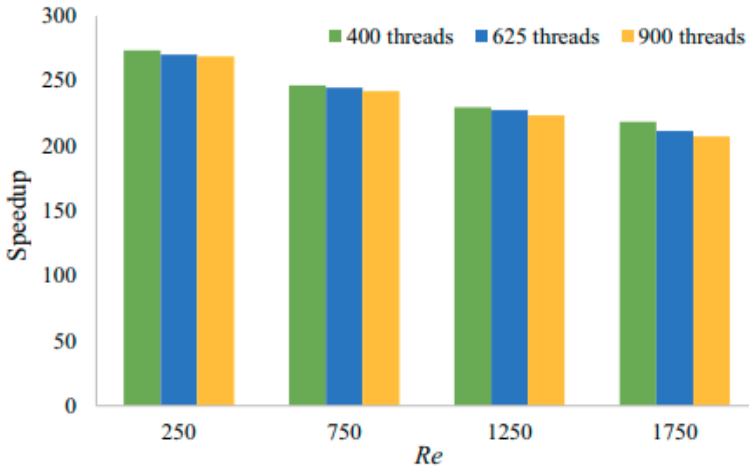
CUDA donosi poboljšanja i u području računalne dinamike fluida. Budući da je za precizne simulacije kretanja fluida najčešće potrebno analizirati zapanjujuće veliku količinu podataka, mnoga pitanja iz područja simulacije dinamike fluida prije pojave CUDA-e mogla su se osloniti isključivo na superračunala [8]. Međutim, pojavom grafičkog ubrzanja i prilagodbom simulacijskih algoritama na način da u najvećoj mogućoj mjeri iskorištavaju prednosti grafičke kartice, danas su čak i osobna računala u mogućnosti simulirati kompleksnu dinamiku fluida.

Poboljšanje performansi - primjer

Stupanj poboljšanja računalnih performansi koristeći CUDA-u može biti opisan na različite načine. Jedan od znanstvenih članaka koji se bavi tematikom opisuje problem nastao prilikom analize toka fluida i prijenosa topline među različitim sustavima [9]. Autori navode kako je primjena numeričkih metoda u cilju predviđanja toka fluida i topline postala vrlo česta, a budući da izračuni numeričkim metodama uglavnom podrazumijevaju velike zahtjeve u smislu računalnih resursa, analize se nerijetko vrše neprihvatljivo sporo.

Potaknuti sporim izvađanjem analiza, autori članka su svoj analitički algoritam

prilagodili za izvađanje na grafičkoj kartici. Time su vrijeme potrebno za vršenje analiza uvelike skratili, a poboljšanje je prikazano dijagramom 1.1 na stranici 10 [9]. Dijagram jasno pokazuje zapanjujuće ubrzanje – koristeći spomenutu implementaciju, analiza se ovisno o ulaznim podacima i broju dretvi skraćuje za faktore veće od 200.



Slika 1.1 Ubrzanje analize koristeći podršku grafičke kartice. Dijagram prikazuje ubrzanja za različite ulazne vrijednosti *Re* uz implementaciju putem 400, 635 i 900 dretvi na grafičkoj kartici.

Naravno, računalna dinamika fluida samo je jedno od područja u kojem je moguće iskoristiti grafičke kartice i CUDA-u za ubrzanje izvođenja izračuna. Također, treba napomenuti da neke implementacije algoritama koristeći prednosti CUDA-e ne nose ubrzanja reda veličine onih s dijagrama 1.1 te da neke čak nisu niti isplative. U sljedećim poglavljima bit će opisan pokušaj i rezultat ubrzanja detekcije osoba koristeći grafičku karticu.

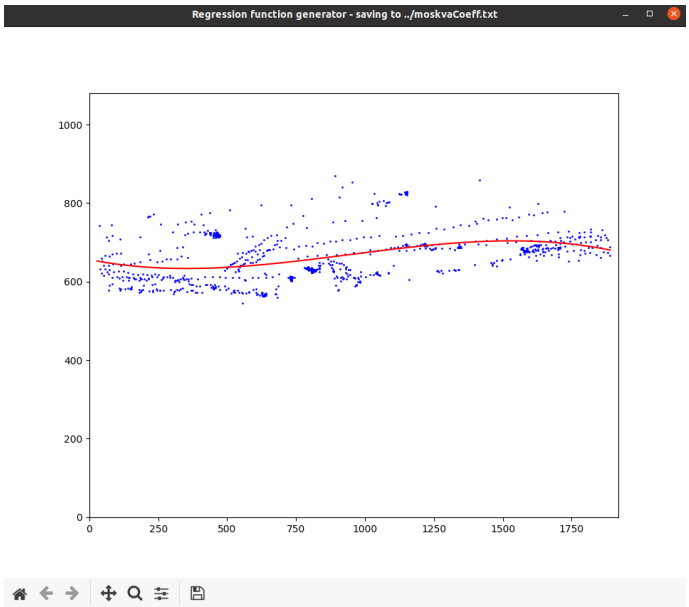
1.2.3 Python

Python je objektno-orijentirani programski jezik koji je korišten za stvaranje programa koji služe kao podrška glavnoj aplikaciji. Razlozi koji su u najvećoj mjeri pridonjeli korištenju Python-a su činjenice da je pomoću istoga vrlo jednostavno

stvarati pregledne i informativne dijagrame, komunicirati s vanjskim sklopovljem i analizirati podatke prikupljene glavnom aplikacijom.

Programi podrške glavnoj aplikaciji napravljeni u Pythonu:

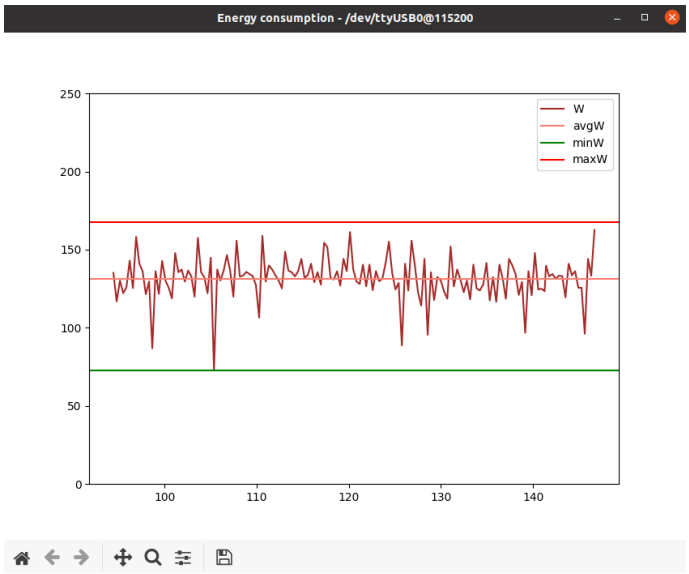
1. program za izračun regresijske funkcije koja predstavlja opće kretanje detektiranih osoba – sučelje prikazano na slici 1.2 na stranici 11,
2. program za dohvaćanje i vizualizaciju informacija s vanjskog sklopovlja za mjerenje potrošnje energije – sučelje prikazano na slici 1.3 na stranici 12.



Slika 1.2 Prikaz regresijske funkcije koja predstavlja nejučestalije kretnje detektiranih osoba (detekcije predstavljene plavim točkama).

1.2.4 Arduino

Arduino je tvrtka koja proizvodi istoimene mikrokontrolere i istoimeno okruženje (engl. *Integrated Developement Environment - IDE*) koje nam omogućuje programiranje spomenutih mikrokontrolera. U sklopu završnog rada, Arduino je korišten za prikupljanje podataka o potrošnji električne energije. U sljedećim poglavljima



Slika 1.3 Prikaz čitanja i vizualiziranja informacija o potrošnji energije s vanjskog sklopovlja (ordinata predstavlja potrošnju u Watt-ima, a apscisa vrijeme u sekundama od početka čitanja podataka).

mjerenje potrošnje električne energije bit će detaljno objašnjeno – dio kroz prikaz korištenih komponenti i njihov raspored, a dio kroz prikaz programa pokrenutog na mikrokontroleru.

1.3 Cilj završnoga rada

Računarstvo visokih performansi potrebno je prikazati na zadatku završnog rada – analizi videozapisa koja se primarno posvećuje detekciji i praćenju osoba te njihovog kretanja. Pokazni primjer bi mogao biti od koristi svakom pojedincu koji posjeduje nadzornu kameru i ne želi pregledavati višesatne snimke, ali i raznim institucijama kojima je bitno brzo i efikasno analizirati podatke sa svojih nadzornih kamera. Izabrani primjer ulazi u područje računarstva visokih performansi zbog velike količine podataka koja može biti dostupna za analizu te zbog činjenice da svaki kadar videozapisa predstavlja zasebnu cjelinu što omogućuje paralelizaciju detekcije osoba kroz različite kadrove. Uz to, korištenjem nekog od algoritama za prepoznavanje osoba na

videozapisu u kombinaciji s ubrzanjem programa korištenjem grafičke kartice otvaraju se mnoge mogućnosti za povećanje efikasnosti. Za rješavanje glavnog zadatka završnog rada korišteni su programski jezik C++ s jednom od najpoznatijih knjižnica programskih funkcija koje pružaju rješenja za računalni vid (opisana u poglavlju 1.2.1 na stranici 7).

Poglavlje 2

Aplikacija za detekciju osoba

U ovom je poglavlju opisan razvoj glavne aplikacije završnog rada – od biranja algoritama do konkretne implementacije izabranih algoritama. Aplikacija služi detektiranju osoba na videozapisima te isticanju osoba koje su skrenule s najčešće ili dopuštene putanje. Treba napometuti da se glavna aplikacija oslanja na aplikaciju opisanu u poglavlju 3 na stranici 40 koja nudi podršku u vidu izračunavanja koeficijenata regresijske funkcije. Regresijska funkcija se određuje analizom detektiranih osoba, a opisuje najčešće kretnje osoba na promatranom videozapisu. Glavna aplikacija je izrađena kroz dva različita pristupa koji su detaljnije objašnjeni u sekciji 2.2 na stranici 25.

2.1 Odabir algoritma za analizu videozapisa

Prije početka razvoja glavne aplikacije, potrebno je odlučiti na koji način ćemo pristupiti samom razvoju. To podrazumijeva proučavanje algoritama za detekciju osoba te odabir jednog od njih.

Jedan od radova koji se primarno bavi opisom nove tehnike identifikacije i praćenja osoba na slici ipak u svojem uvodu navodi neke od starijih, tradicionalnijih pristupa detekciji i praćenju objekata na slici [10]:

- Algoritmi temeljeni na razlikama kadrova (engl. *frame differences*)

[11] Vrlo jednostavan pristup koji detektira pomicanje objekata na videozapisu. Način otkrivanja kretnje se bazira na oduzimanju susjednih, slijednih kadrova. Time se dobiva nova matrica (koja predstavlja razliku kadrova) na kojoj vrijednost posjeduju samo šumovi i objekti koji su se u promatranom intervalu pomicali. Isticanjem takvih lokacija na izvornom kadru ukazujemo na kretnje objekata između dva uzastopna kadra.

- Algoritmi temeljeni na oduzimanju pozadina (engl. *background subtraction*)

  Oduzimanje pozadine funkcionira na gotovo isti način kao prethodno opisana vrsta algoritama. Detaljni opis različitih pristupa koristeći oduzimanje pozadine dostupan je u [12].

- Algoritam temeljen na Haarovom sustavu valova (engl. *Haar Wavelets*)

  [13] Detekcija se provodi u dva dijela – stvaranje klasifikatora i provedba kadrova kroz isti. Spomenuti klasifikator nastaje procesom treniranja korištenjem skupa kadrova koji sadrže traženi objekt i skupa kadrova koji ga ne sadrže. Provedba kadrova kroz klasifikator odvija se u zasebnim sekcijama (sekcije se odnose na dio kadra koji trenutno promatramo), a podrazumijeva izlučivanje karakteristika sekcija kadra te unos istih u klasifikator. Na temelju dobivenih izlaznih rezultata utvrđuje se prisutnost traženog objekta u kadru. U cijeli proces se Haarov sustav valova uklapa u vidu pružanja rješenja za kvalitetnu reprezentaciju slikovnih sekcija, a detaljnije je objašnjen u znanstvenom radu [13].

- Algoritam temeljen na histogramu orijentiranih gradijenata (engl. *Histogram of Oriented Gradients - HOG*)

  [14] Proces detekcije se odvija u dva koraka, a provodi se za svaku sekciju kadra. Prvi korak podrazumijeva određivanje magnituda i smjerova gradijenata te stvaranje histograma prikupljenim vrijednostima. Nakon toga se dobiveni podaci provode kroz istrenirani klasifikator koji odgovara na pitanje prisutnosti određenog objekta u kadru.

- Algoritam temeljen na metodi parcijalnih najmanjih kvadrata (engl. *Partial Least Squares - PLS*)

Ideja ove vrste algoritama se temelji na smanjenju složenosti pristupa opisanog kod histograma orijentiranih gradijenata. Algoritam za svaku promatranu sekciju unutar kadra izlučuje matrice prema karakteristikama rubova (kao i HOG), boje (engl. *color frequency matrix*) i teksture (engl. *co-occurrence matrix*) [15]. Zatim se matrice spajaju i primjenjuje se metoda parcijalnih najmanjih kvadrata u cilju smanjenja dimenzionalnosti rezultantne matrice. Na taj način se omogućuje korištenje jednostavnijeg klasifikatora te ubrzanje cjelokupnog procesa detekcije. Metoda parcijalnih najmanjih kvadrata je detaljnije objašnjena u [15].

- Algoritam temeljen na Houghovoj transformaciji (engl. *Hough transform*)

  Postupak se primjenjuje kod detektiranja ljudskog lica na videozapisu [16]. Sve započinje stvaranjem reprezentacije pomicanja čovjeka iz dva uzastopna kadra temeljem razlike kadrova. Na dobivenoj reprezentaciji se nakon odstranjivanja šumova pronalazi piksel najbliži vrhu kadra (podrazumijeva se da je to vrh glave). Kako ljudsko lice možemo aproksimirati kružnicom za koju nam je poznato da prolazi kroz prethodno spomenuti piksel, smisleno je pretražiti područje u okolini lociranog piksela u cilju pronalaska kružnog objekta (lica). Različiti oblici (između ostalog i kružnica) se mogu na slici utvrditi Houghovom transformacijom koja je detaljno objašnjena u [17].

- Algoritam temeljen na detekciji točaka

  [18] Ova vrsta algoritama služi praćenju objekata, a odvija se implementacijom SIFT (engl. *Scale Invariant Feature Transform*) algoritma. SIFT se provodi u pet glavnih faza:

  1. detekcija ekstrema u promatranom području (engl. *Scale-Space Extrema Detection*),
  2. lokalizacija glavnih točaka,
  3. dodjeljivanje orijentacije,
  4. stvaranje opisnika (engl. *descriptor*) glavnih točaka iz prikupljenih podataka,
  5. ocjenjivanje podudarnosti glavnih točaka.

Dakle, na prvom, referentnom kadru stvaramo prve, referentne opisnike (prva četiri koraka algoritma). U sljedećim kadrovima uspoređujemo novoizračunate opisnike kadra s referentnim opisnicima te na taj način utvrđujemo promjenu položaja promatranog objekta.

- Algoritam temeljen na aktivnom konturnom modelu (engl. *Active Contour Model*)

  [19] Na početku procesa formira se krivulja u inicijalnoj poziciji. Energija krivulje (unutarnja – zaglađuje krivulju; vanjska – privlači krivulju prema promatranom objektu) je predstavljena sustavom parametarskih jednadžbi. Rješavanjem sustava do konvergencije krivulja se pripaja uz rub promatranog objekta i tvori njegovu konturu. Ovim algoritmom se ne izračunava podudarnost s nekim objektom, već algoritam služi izdvajanju karakteristika iz kadra. Izdvojene karakteristike se tada mogu usporediti s predefiniranim modelima traženih objekata.

S druge strane, nužno je spomenuti i pristupe detekciji osoba baziranih na dubokom učenju. Tako možemo opisati nekoliko algoritama [20]:

- R-CNN (engl. *Region-based Convolutional Neural Networks*)

  Algoritam detekcije objekata se provodi u nekoliko glavnih koraka:

  1. Prikupljanje regija od interesa (dijelovi kadra za koje se pretpostavlja da sadrže neke objekte, engl. *candidate regions*) na temelju selektivne pretrage (engl. *selective search*) iz izvornog kadra
  2. Prilagođavanje veličine (visine i širine) dobivenih regija od interesa (uglavnom 1000-2000 regija) tako da ista bude u skladu s fiksnom veličinom ulaza konvolucijske neuronske mreže
  3. Izlučivanje skupa karakteristika svake regije od interesa koristeći konvolucijsku neuronsku mrežu
  4. Klasifikacija dobivenih skupova karakteristika

- Fast R-CNN

  Fast R-CNN uklanja najveću manu prethodno opisanog R-CNN-a. Prila-

gođavanje veličine svake regije od interesa (drugi korak prethodnog algoritma) uvelike usporava cijeli proces detekcije. Fast R-CNN umjesto sporim prilagođavanjem veličine problemu pristupa projekcijom regja od interesa na područje fiksne veličine. Nakon toga se izlučuju skupovi karakteristika pomoću konvolucijske neuronske mreže te se vrši klasifikacija i izračun dimenzija detektiranog objekta.

Ovaj pristup u odnosu na R-CNN rezultira većom brzinom i točnosti prilikom detekcije objekata. Detaljan opis se nalazi u [21].

- YOLO (engl. *You Only Look Once*)

  Prepoznavanje objekata se realizira na prilično izravan način – na ulazne podatke (kadar) se gleda samo jednom, kako i samo ime pristupa kaže. Umjesto prilagođavanja veličine ili projekcije regija od interesa, YOLO u svom radu u potpunosti odbacuje koncept regija od interesa. Umjesto toga, uzima čitav kadar i pretvara ga u potrebne dimenzije (448x448 piksela) [22]. Potom konvolucijska neuronska mreža istovremeno ističe pozicije detekcija različtih objekata i njihove razrede (čovjek, auto, pas...).

  Autori ovog pristupa navode da je najveća prednost YOLO-a u odnosu na Fast R-CNN mogućnost razlikovanja pozadine od objekata pri čemu kod Fast R-CNN-a češće dolazi do pogrešaka [22]. Temeljita usporedba s ostalim pristupima i detaljno objašnjenje YOLO-a se nalazi u [22].

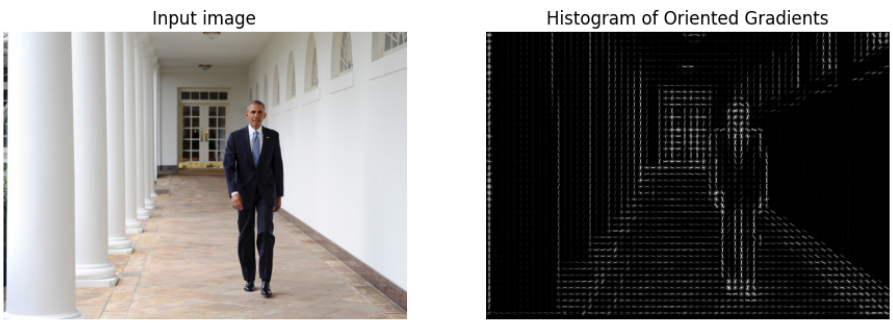
- SSD (engl. *Single Shot Detector*)

  Umjesto detekcije algoritmom na temelju analize sekcija kadra, SSD se bazira na podjeli kadra u mrežu (engl. *grid*). Svaka ćelija unutar mreže je tada odgovorna za detekciju objekata u svojem području. SSD u obzir uzima i situacije kod kojih objekti prelaze granice ćelija. Ovaj je pristup detaljno objašnjen u [23].

2.1.1 Konačni odabir - histogram orijentiranih gradijenata

Na temelju proučenih pristupa za detekciju ljudi na videozapisima i alata s kojima se spomenuti pristupi mogu implementirati, izabran je pristup detekcije putem his-

tograma orijentiranih gradijenata. Razlog odabira upravo tog pristupa je jednostavnost njegove implementacije te dostupnost velikog broja izvora koji opisuju proces implementacije. Također, unutar OpenCV knjižnice programskih funkcija (podsekcija 1.2.1 na stranici 7) već je dostupna gotovo finalna implementacija što cijeli proces uvelike olakšava te ostavlja više vremena za testiranje inačica i pokušaj implementacije inačica na superračunalu. U ovoj će podsekciji biti detaljnije objašnjen princip detekcije osoba putem histograma orijentiranih gradijenata temeljem [14] i [24]. Reprezentacija izračunatih magnituda i smjerova gradijenata je prikazana na slici 2.1 na stranici 19.



Slika 2.1 Reprezentacija izračunatih magnituda i smjerova gradijenata.

Osnovnu ideju detekcije pristupom histograma orijentiranih gradijenata autori istog opisuju kao: [14, izgled i oblik lokalnih objekata mogu često biti vrlo dobro okarakterizirani distribucijom lokalnih gradijenata intenziteta i smjerova rubova]. Dakle, njihov opis pristupa možemo shvatiti na način da se magnituda i smjer gradijenata koriste kao opis pojedinih dijelova slike (kadra) te da se na taj način komplicirana struktura podataka (kao što je kadar) prevodi u strukturu s kojom je lakše provoditi izračune.

Prije objašnjenja procesa detekcije, potrebno je definirati parametre kojima se vode [14]:

- Slika je zapisana u RGB prostoru boja

- Filteri kojima se izlučuju horizontalne i vertikalne magnitude gradijenata glase (na temelju kojih se računaju i smjerovi gradijenata): $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ i $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$
- Histogrami orijentiranih gradijenata se formiraju koristeći devet spremnika prema smjeru (0° – 180°; nije potrebno uključiti smjerove 0° – 360° zbog toga što se gradijent u smjeru od, primjerice, 90° na isti način predstavlja i u smjeru od 270°, osnosno djeluje duž pravca bez usmjerenja)
- Ćelije su veličine 8x8 piksela
- Blokovi pri normalizaciji se sastoje od četiri ćelije (odnosno veličine su 16x16 piksela), a pomiču se u koracima od 8 piksela (odnosno jedne ćelije)
- Sekcije u kojima se vrši detekcija su dimenzija 64x128 piksela
- Za klasifikaciju se koristi linearni stroj potpornih vektora (engl. *Support Vector Machine - SVM*)

Priprema kadra

Prije početka provedbe same detekcije, svaki je kadar potrebno pripremiti kako bi bio se algoritam mogao provesti u sukladnosti s iznad navedenim parametrima. S izvornog kadra bilo kojih dimenzija se izlučuju sekcije s $1 : 2$ omjerom širine i visine (točne dimenzije nisu relevantne) [24]. Izlučene sekcije se tada prilagođavaju veličini od 64x128 piksela (zbog toga odnosi širine i visine sekcija moraju biti u navedenom omjeru). Iako se u sljedećem koraku vrši normalizacija *gamme* i boje [14], također se navodi da [14, normalizacije imaju skroman utjecaj na performanse, vjerojatno zbog toga što naknadna normalizacija HOG opisnika (engl. *HOG Descriptor*) postiže sličan učinak]. Navedena činjenica je potvrđena i od strane drugih izvora te zaključujemo da normalizacije *gamme* i boje možemo preskočiti zbog njihovog nezamjetnog utjecaja [24].

Na primjer: ukoliko analiziramo kadar dimenzija 1000x800 piksela, detekciju vršimo nad bilo kojom sekcijom istoga te je jedini uvjet pri odabiru sekcije odnos njezine širine i visine u prethodno spomenutom omjeru $1 : 2$. Izabranu sekciju tada prila-

gođavamo dimenzijama od 64x128 piksela te započinjemo sljedeći korak u procesu detekcije. Primjer je ilustriran na slici 2.2 na stranici 21.

(a) Originalni kadar dimenzija 1024x728 piksela. (b) Sekcija za detekciju, 256x512 piksela. (c) Prilagodba sekcije, 64x128 piksela.

Slika 2.2 Primjer pripreme kadra. Slike nisu prikazane u točnoj veličini zbog rasporeda na stranici, ali su točne dimenzije navedene u opisima.

Izračun magnituda i smjera gradijenata

U ovom koraku opisan je izračun magnituda i smjera gradijenata koji su nužni za stvaranje histograma orijentiranih gradijenata (HOG opisnik). Magnitude gradijenata se računaju temeljem prethodno spomenutih filtera, a smjerovi temeljem izračunatih magnituda.

Filtere za izračun magnituda možemo pojednostaviti na formule za horizontalnu i vertikalnu magnitudu gradijenta nekog piksela – M_H i M_V, respektivno:

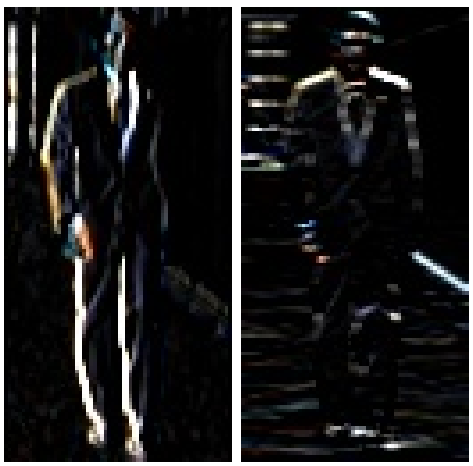
- $M_H = \begin{bmatrix} I_L & I & I_D \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix} = (-1) * I_L + 0 * I + 1 * I_D = -I_L + I_D$, gdje su I_L, I i I_D intenziteti lijevog piksela, piksela za kojeg se računa magnituda i desnog piksela respektivno

- Analogno dobivamo: $M_V = -I_{IZ} + I_{IS}$, gdje su I_{IZ} i I_{IS} vrijednosti intenziteta piksela iznad i ispod promatranog piksela za kojeg se računa magnituda

Rezultantna magnituda M se tada računa kao korijen zbroja kvadrata njezinih komponenata – $M = \sqrt{M_H^2 + M_V^2}$

Smjer gradijenta je tada jednako trivijalno za izračunati – koristi se formula $\Theta = \arctan \frac{M_V}{M_H}$, pri čemu je Θ rezultantni smjer gradijenta.

Prikaz svrhe računanja magnituda gradijenata je dan na slici 2.3 na stranici 22.



(a) Prikaz horizontalne magnitude gradijenta prilagođene sekcije.

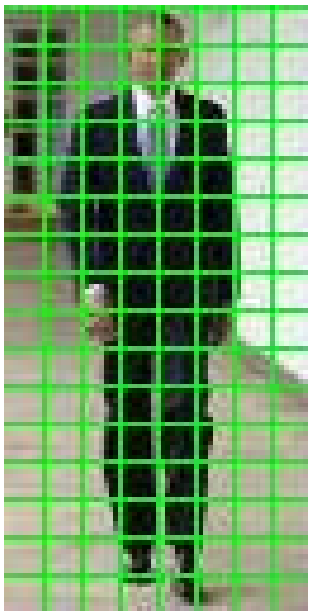
(b) Prikaz vertikalne magnitude gradijenta prilagođene sekcije.

Slika 2.3 Prikaz magnituda gradijenata. Možemo primjetiti da horizontalni gradijent ističe vertikalne rubove, a vertikalni horizontalne rubove.

Stvaranje histograma orijentiranih gradijenata

Kada znamo kako izračunati magnitudu i smjer gradijenata, sliku dijelimo u ćelije (prikazano na slici 2.4 na stranici 23) prema prethodno opisanim parametrima (ćelije

dimenzija 8x8 piksela) za koje računamo magnitude i smjer gradijenata. Prema dobivenim podacima tada stvaramo histograme orijentiranih gradijenata.



Slika 2.4 Prikaz podjele sekcije na ćelije dimenzija 8x8 piksela

Odjeljivanje se vrši kako bi se ćelije, odnosno pojedini dijelovi sekcija mogli opisati na računalu prigodan način [24]. Nadalje, odjeljivanje u ćelije je dobar način za komprimiranje informacija – ćelija inicijalno sadrži $8 * 8 * 3$ vrijednosti (64 RGB piksela), a zatim reprezentacijom kroz magnitudu i smjer gradijenata piksela sadržaj ćelija smanjujemo na $8 * 8 * 2$ vrijednosti (64 gradijenta s vrijednostima magnitude i smjera) [24]. Preostalih 128 vrijednosti ćelije se stvaranjem histograma orijentiranih gradijenata pretvara u tek 9 vrijednosti.

Kod ovog je koraka još ostalo za opisati njegovu najvažniju točku – stvaranje histograma orijentiranih gradijenata. Histogrami se stvaraju po principu odjeljivanja magnituda gradijenata u spremnike prema njihovim smjerovima (prethodno spomenuti raspon od 0° – 180°). Histogrami su ustvari vektori (ili nizovi) od 9 elemenata koji odgovaraju smjerovima gradijenata (redom smjerovi 0°, 20°, 40°, ..., 160°, 180°) [24]. Gradijent pridonosi vrijednosti (ili vrijednostima) nekog (ili nekih) spremnika na način da se temeljem njegovog smjera njegova magnituda pridoda postojećoj vri-

jednosti odgovarajućeg (ili odgovarajućih) spremnika.

Primjerice, ukoliko gradijent ima magnitudu $M = 7$ i smjer $\Theta = 20°$, spremniku histograma koji predstavlja smjer od 20° ćemo na postojeću dodati vrijednost magnitude – 7. S druge strane, ako gradijent ima magnitudu $M = 10$ i smjer $\Theta = 70°$, ne možemo ga jednoznačno svrstati niti u jedan spremnik (budući da ne postoji spremnik koji predstavlja smjer od 70°). U toj situaciji činimo logičnu stvar – budući da je 70° točno na polovici između 60° i 80°, polovicu vrijednosti magnitude pridodajemo spremnicima koji predstavljaju smjerove od 60° i 80°. Naravno, kada smjer gradijenta ne bi bio točno između vrijednosti nekih spremnika, magnituda bi se rasporedila proporcionalno (recimo, za smjer gradijenta od 75°, 25% vrijednosti magnitude $(1 - \frac{|60-75|}{20})$ se pridodaje spremniku koji predstavlja smjer od 60°, a 75% vrijednosti magnitude $(1 - \frac{|80-75|}{20})$ spremniku koji predstavlja smjer od 80°).

Stvar o kojoj možemo razmišljati su dimenzije ćelija – zašto baš 8x8 piksela? Budući da je čovjek predstavljen sekcijom dimenzija 64x128 piksela, njegovi dijelovi se najbolje obuhvaćaju ćelijom dimenzija 8x8 piksela [24].

Normalizacija

Korak normalizacije je potreban zbog minimiziranja utjecaja osvjetljenja na histograme orijentiranih gradijenata [24].

Primjerice, uzmimo da je histogram reprezentiran vektorom s dva elementa – $V_1 = \begin{bmatrix} 16 & 8 \end{bmatrix}$. Ukoliko se razina svjetlosti poveća za faktor 2, histogram će poprimiti dvostruku vrijednost, odnosno biti će reprezentiran vektorom $V_2 = \begin{bmatrix} 32 & 16 \end{bmatrix}$. Budući da bi to moglo imati negativan učinak na cijeli proces detekcije, provodimo normalizaciju.

Normalizacija se provodi dijeljenjem svake vrijednosti vektora s njegovom apsolutnom vrijednosti. Apsolutne vrijednosti prethodno spomenutih vektora V_1 i V_2 bi tako iznosile 17.88 i 35.77, respektivno. Nakon provedene normalizacije nad vektorima, vrijednosti bi se ponovno izjednačile unatoč dvostrukom osvjetljenju prilikom stvaranja vektora (zamišljenog histograma s dvije vrijednosti) V_2. Tako bi rezultantna situacija izgledala ovako: $V_1 = V_2 = \begin{bmatrix} 0.894 & 0.447 \end{bmatrix}$.

Kod HOG pristupa normalizacija se odvija nad blokovima koji se sastoje od 4 ćelije (dimenzije 16x16 piksela) te se iterativno pomiču za jednu ćeliju udesno kroz cijelu sekciju dok se ne normaliziraju svi mogući blokovi sekcije [24]. Naravno, dimenzije i korak pomicanja blokova nisu definirani proizvoljno, već prema prije spomenutim parametrima navedenima u [14].

Završetkom procesa normalizacije za svaki se blok dobiva vektor s 36 vrijednosti (blok predstavljen s 4 ćelije od kojih je svaka predstavljena vektorom od 9 vrijednosti).

Klasifikacija

Dobiveni normalizirani vektori s 36 vrijednosti se spajaju u jedan zajednički vektor s 3780 vrijednosti (105 blokova s 36 vrijednosti) [24]. Finalni vektor koji predstavlja cijelu sekciju dimenzija 64x128 piksela se unosi u linearni stroj potpornih vektora čiji izlaz naznačuje o kojem se objektu radi.

2.2 Različiti pristupi

Prilikom razvoja softverskog rješenja za implementaciju HOG detekcije osoba, stvorene su dvije inačice rješenja. Obje se oslanjaju na podršku OpenCV knjižnice programskih funkcija. Prva inačica pri detekciji osoba na videozapisima koristi isključivo procesorsku snagu dok druga inačica, iako temeljena na prvoj, uz procesorsku snagu koristi i prednosti grafičkog ubrzanja opisanih u podsekciji 1.2.2.

U sljedeće dvije podsekcije ukratko je opisan razvoj pojedinih inačica, a usporedba istih je dana u poglavlju 4 na stranici 49.

2.2.1 Prva inačica aplikacije

Prva verzija aplikacije zamišljena je kao inicijalni pokušaj namjenjen shvaćanju principa rada knjižnice programskih funkcija OpenCV (1.2.1, stranica 7) – njezinih osnovnih struktura podataka i njihovih operacija te algoritma detekcije na principu histograma orijentiranih gradijenata čiju implementaciju knjižnica pruža.

Kako tema završnog rada navodi, svrha aplikacije je analiza videozapisa u smislu detekcije prisutnih osoba i njihovog kretanja. Uz to, potrebno je koristiti tehnike računarstva visokih performansi koje se u ovoj inačici reprezentiraju višedretvenim radom koji je ugrađen u implementaciju algoritma detekcije [25]. U ovom se programu ne koriste prednosti grafičkog ubrzanja te se oslanjamo isključivo na podršku glavnog procesora računala.

Opis rada programa

Program naravno započinje uključivanjem potrebnih knjižnica:

- iostream i iomanip - omogućuje korištenje standardnog ulaza i izlaza te manipulaciju istim (koristi se naredba setprecision kako bismo definirali broj decimalnih mjesta prilikom prikazivanja brzine obrade kadrova po sekundi)
- opencv2/objdetect.hpp - omogućuje korištenje klase koja reprezentira HOG opisnik i provodi proces detekcije (HOGDescriptor)
- opencv2/highgui.hpp - omogućuje korištenje naredbe imshow (prikazivanje kadrova u prozoru operacijskog sustava) i waitKey (reagiranje sukladno korisničkim uputama s tipkovnice)
- opencv2/imgproc.hpp - omogućuje modifikaciju Mat objekata koji reprezentiraju kadar (označavanje detekcija, putanja, brzine izvršavanja...)
- opencv2/videoio.hpp - omogućuje čitanje datoteke videozapisa

Nakon što smo uključili svu potrebnu pomoć, stvaramo iznimno jednostavnu klasu koja će nam služiti kao detektor osoba na videozapisima, a zamišljena je kao temelj za buduću nadogradnju. Klasa je nazvana Detector, a u sebi sadrži isključivo objekt prethodno istaknute klase HOGDescriptor. Prilikom instanciranja objekta, objektu klase HOGDescriptor se postavlja predefinirani, istrenirani linearni stroj potpornih vektora. Uz konstruktor, klasa sadrži i dvije funkcije namijenjene detektiranju i prilagodbi (temeljenu na iskustvu kroz korištenje programa) detektiranih područja. Navedene funkcije su također zamišljene kao temelji sljedeće inačice kod koje će se detekcija i praćenje osoba provoditi na nešto drukčiji način.

Bitno je napomenuti naredbu koja se poziva nad objektom hog – detectMultiScale (linija 13 u prikazu programskog kôda 2.1 na stranici 27). Naime, naredba služi provođenju detekcije objekata različite veličine (engl. *detect multi scale*) putem danih ulaznih informacija (objekt img) te spremnika za detektirane lokacije (referenca na vektor found). Uz navedene glavne argumente, detectMultiScale može primiti nekolicinu drugih argumenata (istaknuti argumenti se redoslijedom kako su navedeni koriste u prikazu programskog kôda 2.1) [26]:

- **img** – obavezan argument u obliku matrice koja se sastoji od jednog ili tri kanala 8-bitnih cijelih brojeva bez predznaka (jednostavnije rečeno – slike) na kojima se provodi detekcija,
- **foundLocations** – predstavlja vektor pravokutnika unutar kojih se nalaze detektirani objekti,
- foundWeights – predstavlja vektor decimalnih vrijednosti unutar kojih se nalaze vrijednosti pouzdanosti za detekciju svakog pojedinog objekta (dakle, veličina vektora foundLocations je jednaka veličini vektora foundWeights),
- **hitThreshold** – definira prag za označavanje potencijalnih detekcija valjanim (njegovom izmjenom se mijenja broj elemenata vektora foundLocations),
- **winStride** – označava pomak blokova prilikom normalizacije (veličina blokova mora biti dijeljiva veličinom ćelija kako bi cijeli proces bio valjan – normalizacija objašnjena u podsekciji 2.1.1; što je pomak veći, to je izvođenje algoritma brže, ali i nepreciznije),
- **padding** – definira broj piksela u horizontalnom i vertikalnom smjeru koji se podstavljaju (engl. *padding*) iz ćelija prije stvaranja HOG opisnika [27],
- **scale** – određuje koeficijent kojim se promatranom dijelu mijenja veličina (budući da želimo detektirati objekte različitih veličina) prilikom detekcije (što je koeficijent veći, nad manje različith veličina vršimo detekciju) [27],
- finalThreshold – OpenCV dokumentacija ne nudi dodatan opis,
- **useMeanshiftGrouping** – određuje korištenje Mean shift algoritma za grupiranje detekcija.

Prikaz programskog kôda 2.1 Klasa `Detector`

```
1  class Detector
2  {
3      HOGDescriptor hog;
4
5  public:
6      Detector() : hog()
7      {
8          hog.setSVMDetector(HOGDescriptor::
               getDefaultPeopleDetector());
9      }
10
11     void detect(InputArray img, vector<Rect>& found)
12     {
13         hog.detectMultiScale(img, found, 0, Size(8, 8), Size(),
               1.05, true);
14     }
15
16     void adjustRect(Rect& r) const
17     {
18         r.x += cvRound(r.width * 0.1);
19         r.width = cvRound(r.width * 0.8);
20         r.y += cvRound(r.height * 0.07);
21         r.height = cvRound(r.height * 0.8);
22     }
23 };
```

Nakon definiranja klase za detekciju osoba, dolazimo do glavne funkcije programa. U prvih nekoliko linija prikazanih u prikazu programskog kôda 2.2 na stranici 28 se instancira objekt koji koristimo za dohvaćanje kadrova iz videozapisa, objekt za pohranu dohvaćenog kadra, vektor točaka u koji se spremaju koordinate koje služe prilikom prikazivanja putanja i objekt prethodno opisane klase.

Prikaz programskog kôda 2.2 Uvodne radnje unutar `main` funkcije

```
1      string video_location = "cpuhog_input.mov";
```

```
2
3       VideoCapture cap(video_location);
4
5       if (!cap.isOpened())
6       {
7           cout << "Can␣not␣open␣video␣stream." << endl;
8           return -1;
9       }
10
11      Mat frame;
12      vector<Point> track;
13      Detector detector;
```

Program zatim ulazi u petlju koja se izvodi dok se ne procesuiraju svi kadrovi ili dok korisnik ne označi želju za zaustavljanjem analize.

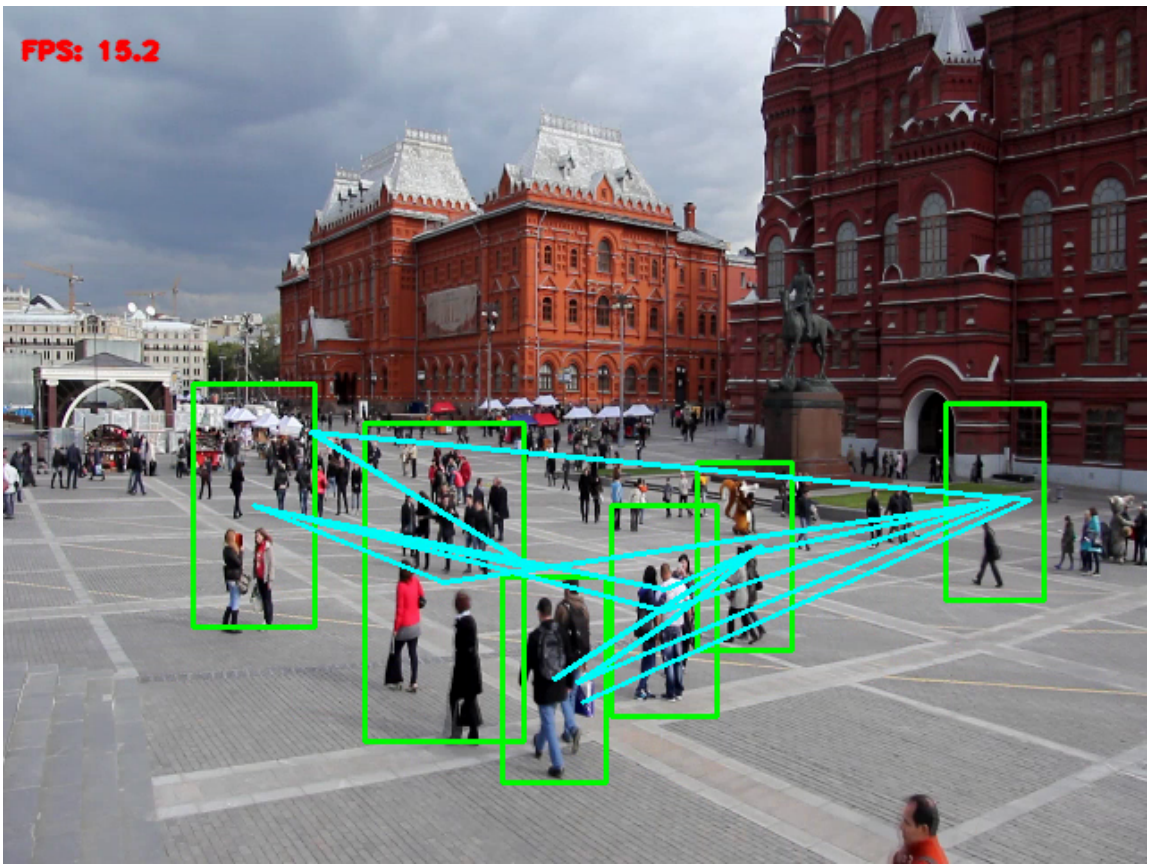
Unutar petlje, nakon inicijalne provjere jesmo li analizirali sve kadrove krećemo s postupkom prilagodbe veličine pročitanog kadra te pretvaranjem kadra u boji u crno-bijelu varijantu. Navedene postupke provodimo s ciljem ubrzanja cijelog procesa detekcije prilikom čega su gubitci na preciznosti nezamjetni. Sljedeći korak je detekcija koristeći objekt detector, odnosno njegovu funkciju detect. Zatim prikazujemo kadar uz isticanje detektiranih osoba, njihove putanje i količinu analiziranih kadrova po sekundi. Iteracija petlje završava provjerom eventualne korisnikove želje za izlaskom iz programa.

Prikaz dobivenih rezultata pokretanjem programa na slici 2.5 na stranici 30. Provedena mjerenja i usporedba s drugom inačicom aplikacije su prikazani u poglavlju 4 na stranici 49.

Zapažanja prilikom rada prve inačice aplikacije

Već pri kratkotrajnom korištenju prve inačice aplikacije uočava se nekoliko problema, ali i ohrabrujućih znakova.

Prvi od njih se tiče odabira koeficijenata prilikom pozivanja funkcije detectMultiScale. Naime, mnogo je argumenata vrlo osjetljivo te čak i minimalne promjene istih uzro-

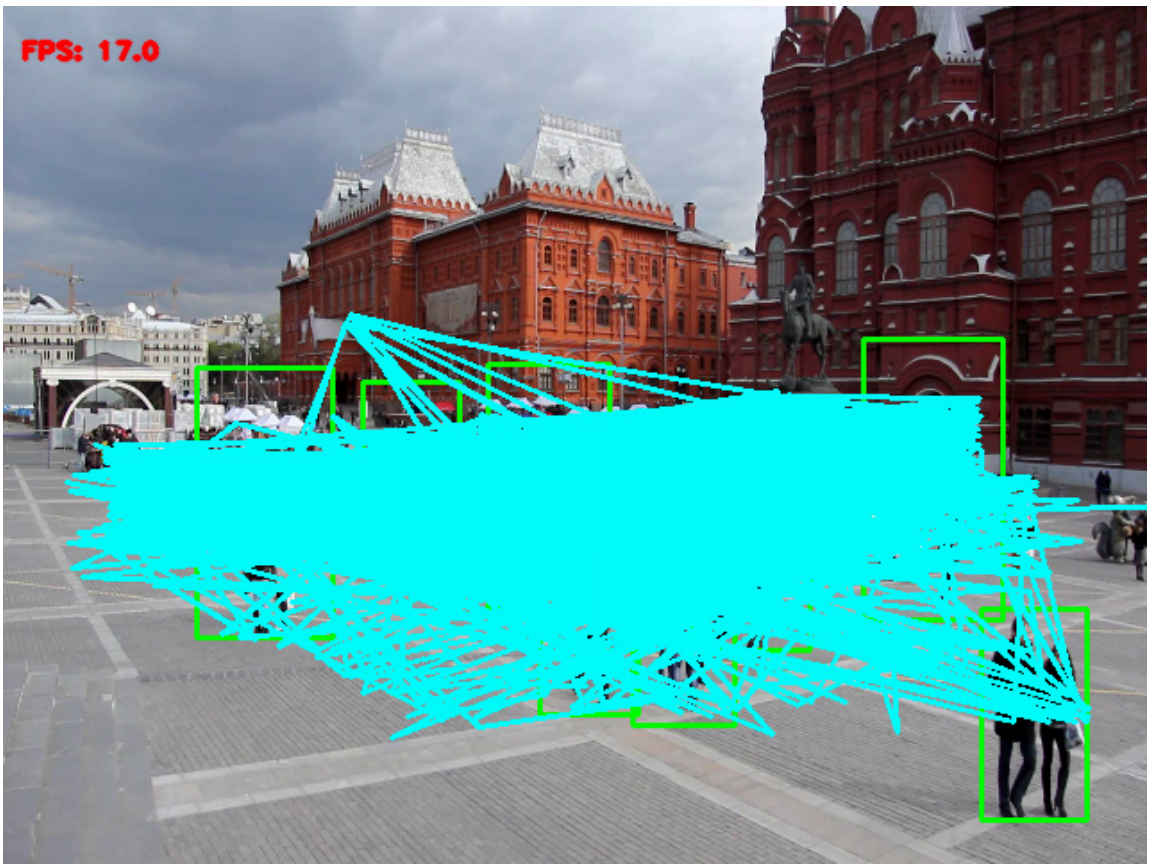


Slika 2.5 Prikaz izvođenja prve inačice aplikacije.

kuju velike razlike u broju detektiranih osoba te brzini izvađanja analize. Također, za svaki je videozapis potrebno dodatno izvršiti precizno podešavanje argumenata kako bi algoritam što bolje radio. Ovaj je problem u drugoj inačici aplikacije riješen uvođenjem mogućnosti potpune kontrole nad najutjecajnijim argumentima spomenute funkcije za vrijeme izvađanja programa.

Nadalje, javlja se problem prilikom prikaza putanje detektiranih osoba. Kod videozapisa koji sadržavaju velik broj detekcija, istaknute putanje su iznimno nepregledne (prikazano na slici 2.6 na stranici 31) i umjesto otkrivanja novih informacija, otežavaju nam promatranje detektiranih osoba. U svrhu minimiziranja negativnog utjecaja navedenog problema, prikazuje se najviše 15 najnovijih putanja. Time se naravno gube neke informacije, ali se preglednost značajno poboljšava (prikazano

na slici 2.5 na stranici 30). Inače, problem je prouzrokovan činjenicom da prilikom implementacije u obzir nije uzeta činjenica da će videozapisi sadržavati više različitih osoba (iako je to naravno bilo očito i potpuno očekivano). Implementacija koja ne bi imala navedeni problem bila bi mnogo složenija zbog toga što bi zahtijevala i pouzdan algoritam prepoznavanja osoba koji bi putanje spajao s pojedinim osobama. Budući da bi to zahtjevalo puno više vremena te da prepoznavanje osoba nije eksplicitno uključeno u zadatak završnog rada, druga inačica aplikacije problem je rješava na nešto drukčiji, ali istovremeno prilično zadovoljavajuć način.

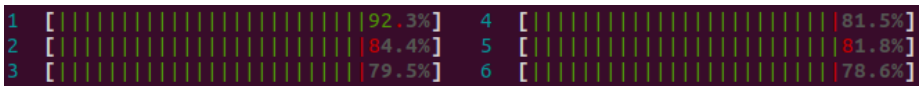


Slika 2.6 Prikaz problema prilikom prikazivanja putanja u prvoj inačici aplikacije.

Posljednji problem tiče se efikasnosti. Trenutna brzina analize videozapisa je između 15 i 20 kadrova u jednoj sekundi. Budući da je poželjno da je programsko rješenje primjenjivo u stvarnosti te da može raditi u realnom vremenu, to nije do-

voljno. Naime, većina današnjih nadzornih kamera snima prostorije brzinom od 15 kadrova u sekundi, ali prilikom detektiranja kretanja brzinu snimanja povećavaju na 20-30 kadrova po sekundi [28]. Poboljšanje brzine značajno je u drugoj inačici aplikacije.

Unatoč navedenim problemima, uspjeh je da ovo jednostavno rješenje gotovo prihvatljivom brzinom analizira videozapise te da pritom koristi sve dostupne procesorske resurse. To nam jasno daje do znanja da višedretveni rad algoritma funkcionira te da je program vjerojatno primjenjiv na superračunalima u cilju bržeg izvršavanja. Postotak iskorištavanja procesorskih resursa jasno je prikazan slikom 2.7 na stranici 32.



Slika 2.7 Prikaz iskorištavanja procesorskih resursa u prvoj inačici aplikacije.

2.2.2 Druga inačica aplikacije - ubrzanje grafičkom karticom

Iako je prva inačica aplikacije u poprilično mjeri ispunila svoj cilj i očekivanje (upoznavanje OpenCV knjižnice, rad s njezinim osnovnim strukturama, operacijama i algoritmima), također je istaknula nekoliko problema koje je potrebno riješiti. Cilj prilikom razvoja druge inačice aplikacije je omogućavanje dinamičke kontrole parametara, rješavanje pitanja isticanja detekcije čija su kretanja izvan prosječne putanje te poboljšanje efikasnosti.

Dinamička kontrola parametara i isticanje detekcija

Ova inačica aplikacije omogućava dinamičku kontrolu parametara detekcije (opisani u podsekciji 2.2.1 na stranici 27), ali i mnogih drugih parametara koji se tiču rada same aplikacije (zapisivanje detekcija, učitavanje najčešće putanje i sl.). Nadalje, problem isticanja osoba izvan prosječne putanje riješen je podrškom jednostavne, zasebno

razvijene aplikacije (detaljnije opisana u poglavlju 3 na stranici 40), a kontrolira se putem dodjeljivanja vrijednosti odgovarajućim parametrima.

Parametri i općenite postavke koje je moguće postaviti prilikom pozivanja (pokretanja) programa su sljedeći:

- <image_name> – specificiranje imena i lokacije slike koju se analizira
- ––video <video_name> —specificiranje imena i lokacije videozapisa kojeg se analizira
- ––camera <camera_id> – identifikator kamere čiji se podaci analiziraju
- ––folder <folder_path> – mapa koja sadrži slike za analizu
- ––svm <svm_file_path> – put do linearnog stroja potpornih vektora
- ––make_gray <true/false> – želimo li da se kadar pretvara u crno-bijelu boju prije analize radi povećanja brzine
- ––resize_src <true/false> – želimo li da se prilagođava veličina kadra radi povećanja brzine izvođenja
- ––width <width> – širina kadra koja se koristi prilikom prilagodbe veličine
- ––height <height> – visina kadra koja se koristi prilikom prilagodbe veličine
- Definiranje parametara koji se koriste prilikom detekcije, a većina je opisana u 2.2.1 na stranici 27:
  - ––hit_threshold <hit_threshold>
  - ––scale <scale>
  - ––nlevels <number_of_levels> – broj različitih veličina prozora prilikom detekcije
  - ––win_width <window_width> – određuje veličinu prozora prilikom detekcije (visina se računa kao dvostruka vrijednost širine; zadana vrijednost: 64)
  - ––win_stride_width <window_stride_width> – definira horizontalni pomak prozora prilikom detekcije

  - ––win_stride_height <window_stride_height> – definira vertikalni pomak prozora prilikom detekcije
  - ––block_width <block_width> – određuje veličinu blokova koji se koriste prilikom normalizacije (visina jednaka širini; zadana vrijednost: 16)
  - ––block_stride_width <block_stride_width> – definira horizontalni pomak blokova prilikom normalizacije
  - ––block_stride_height <block_stride_height> – definira vertikalni pomak blokova prilikom normalizacije
  - ––cell_width <cell_width> – određuje veličinu ćelije (visina jednaka širini)
  - ––nbins <number_of_bins> – broj spremnika histograma
  - ––gr_threshold <group_threshold> – određuje osjetljivost funkcije grupiranja susjednih detekcija
  - ––gamma_correct <true/false> – želimo li koristiti gamma normalizaciju ili ne

- Definiranje parametara koji se koriste prilikom spremanja videozapisa detekcije:
  - ––write_video <true/false> – želimo li da se videozapis detekcije sprema
  - ––dst_video <path> – lokacija na koju se videozapis detekcije sprema ukoliko je ––write_video postavljen na true
  - ––dst_video_fps <fps> – broj kadrova u sekundi izlaznog videozapisa detekcije ukoliko je ––write_video postavljen na true

- ––save_coordinates <coordinates_file_name> – lokacija na koju se spremaju koordinate detektiranih osoba
- ––load_reg <reg_file_name> – lokacija s koje se čitaju regresijski koeficijenti
- ––tolerance <tolerance> – tolerancija skretanja s putanje regresijske funkcije u postocima (u odnosu na visinu kadra) prilikom isticanja osoba

Od svih navedenih parametara, određen broj se može izmjenjivati tijekom izvo-

đenja aplikacije različitim unosima putem tipkovnice računala:

- pritiskom na tipku M ili m se uključuje ili isključuje podrška grafičke kartice,
- pritiskom na tipku G ili g se uključuje ili isključuje pretvorba kadra u njegovu crno-bijelu varijantu prije procesa detekcije,
- pritiskom na tipke 1, 2, 3 i 4, respektivno, se prethodno opisani parametri *scale*, *nlevels*, *gr_threshold* i *hit_threshold* povećavaju,
- pritiskom na tipke Q ili q, W ili w, E ili e i R ili r, respektivno, se prethodno opisani parametri *scale*, *nlevels*, *gr_threshold* i *hit_threshold* smanjuju,
- pritiskom na tipku 5 se postotak tolerancije u slučaju dostupnih podataka o najčešćoj putanji povećava,
- pritiskom na tipku T ili t se postotak tolerancije u slučaju dostupnih podataka o najčešćoj putanji smanjuje,
- pritiskom na tipku C ili c se uključuje ili isključuje gamma korekcija.

Efikasnost detekcije

Budući da su performanse analize također jedan od problema prethodne inačice (posebice kod eventualnog korištenja u realnom vremenu), u ovoj je verziji aplikacije razmotreno korištenje ubrzanja izvođenja programa pomoću grafičke kartice. Nakon proučavanja znanstvenih članaka i dostupnih internetskih izvora koji se bave tom tematikom, odlučeno je da je pokušaj ubrzanja grafičkom karticom vrijedan istraživanja i implementacije. Naime, određeni članci izvještavaju o poboljšanju ubrzanja od 6700% (67 puta) kada se umjesto sekvencijalnog izvođenja programa koristi paralelno izvođenje uz podršku grafičke kartice [29]. Naravno, pogrešno je zanositi se tolikim, zapanjujućim ubrzanjem i očekivati upravo takav rezultat. Razlozi zbog kojih se očekuje manje poboljšanje su:

- u prethodnoj implementaciji se nije koristilo sekvencijalno izvođenje – implementacija je već podrazumijevala višedretveni rad koji također u velikoj mjeri pridonosi poboljšanju performansi,
- izvor koji spominje navedeno poboljšanje se oslanja na vlastitu implementaciju

HOG algoritma koja podrazumijeva direktno upravljanje grafičkom karticom – ovakav pristup zahtijeva znatno više vremena i iskustva.

Opis rada programa

Sukladno činjenici da ova inačica aplikacije sadrži nešto veći broj funkcionalnosti, broj knjižnica koje je potrebno učitati je također veći. Prije početka izvođenja programa, učitavaju se sve knjižnice kao kod prethodne inačice uz dodatak sljedećih:

- fstream – služi za rad s datotekama (pisanje i čitanje podataka vezanih za najčešću putanju na kojoj se detektiraju osobe),
- string – služi za rad sa znakovnim varijablama (engl. *string*),
- sstream – služi za za rad sa znakovnim varijablama,
- stdexcept – služi za rad s iznimkama koje se pojavljuju prilikom rada programa,
- opencv2/core/utility.hpp – služi za određene funkcije knjižnice programskih funkcija OpenCV (primjerice, za poziv funkcija koje računaju broj obrade kadrova u sekundi),
- opencv2/cudaobjdetect.hpp – služi za implementaciju algoritma histograma orijentiranih gradijenata uz ubrzanje grafičkom karticom.

Nakon učitavanja svih potrebnih knjižnica, započinje konkretniji rad aplikacije. Možemo ga podijeliti u dva dijela temeljena na programskim klasama unutar aplikacije – Args i App kojima se upravlja iz korijenske funkcije main.

Aplikacija započinje čitanjem argumenata i inicijalizacijom postavki svoga rada. Za to se koristi klasa Args čija je svrha da na organiziran, jednostavan i pregledan način pohrani sve parametre kojima se određuje cjelokupan proces učitavanja ulaznih vrijednosti, detekcije i interpretacije detekcija. Proces započinje stvaranjem objekta spomenute klase putem njenog konstruktora koji u članske varijable koje predstavljaju argumente učitava zadane, predefinirane vrijednosti. Tada se metodom read koja pripada istoj klasi nalaze svi eksplicitno navedeni parametri pri pozivu aplikacije te se isčitanim vrijednostima zamjenjuju odgovarajuće vrijednosti unutar instanciranog objekta zadane prilikom poziva konstruktora. Klasa Args je dana u prilogu

A.1 na stranici 80.

Drugi dio rada aplikacije koristi klasu App koja nudi rješenja za sâm proces detekcije, isticanje iznimaka u kretanju, komunikaciju s korisnikom i dinamičku izmjenu parametara te izračun brzine obrade kadrova. Naravno, prije korištenja ponuđenih rješenja na osnovu prethodno dobivenog objekta koji reprezentira ulazne argumente se instancira objekt klase App čime stvaramo podlogu za izvođenje pruženih mogućnosti. Sljedeći korak je nad stvorenim objektom izvršiti metodu run koja, kako joj i ime kaže, predstavlja cjelokupno izvođenje aplikacije.

Metoda run se u svojem djelovanju oslanja na nekoliko drugih metoda svoje klase koje joj pomažu u računaju brzine obrade kadrova, ocjeni udaljenosti detekcija od najčešće putanje te uvažavanju unosa s tipkovnice, a odvija se u nekoliko koraka:

1. inicijalizacija prethodno opisanih veličina (win_size, block_size, win_stride, block_stride, cell_size) koje će se uz odabrani broj spremnika histograma koristiti za stvaranje dvaju objekata koji predstavljaju opisnik histograma orijentiranih gradijenata od kojih jedan podržava grafučko ubrzanje,
2. učitavanje stroja potpornih vektora (ukoliko ne posjedujemo vlastiti, učitava se predefinirani stroj potpornih vektora koji pruža OpenCV),
3. formiranje sučelja za dohvaćanje kadrova (možemo ih dohvaćati iz raznih izvora poput kamere ili videozapisa pa se implementacije razlikuju),
4. zapis dimenzija kadrova u izlaznu datoteku u slučaju korištenja mogućnosti izračuna najčešće putanje kretanja,
5. dohvaćanje kadra i završetak metode ukoliko je kadar prazan,
6. provođenje detekcije pristupom histograma orijentiranih gradijenata s podrškom grafičke kartice ukoliko je tako parametrima određeno (prilikom korištenja podrške grafičke kartice, kadar se prvo treba pohraniti na istoj kako bi prilikom provođenja izračuna ulazni podaci bili dostupni, a koriste se strukture podataka i metode iz knjižnice opencv2/cudaobjdetect.hpp koje predstavljaju ekvivalente struktura podataka i metoda korištenih u prvoj inačici aplikacije uz uvođenje funkcionalnosti koje omogućuje CUDA),
7. isticanje dobivenih detekcija (u slučaju dostupnih podataka o najčešćoj puta-

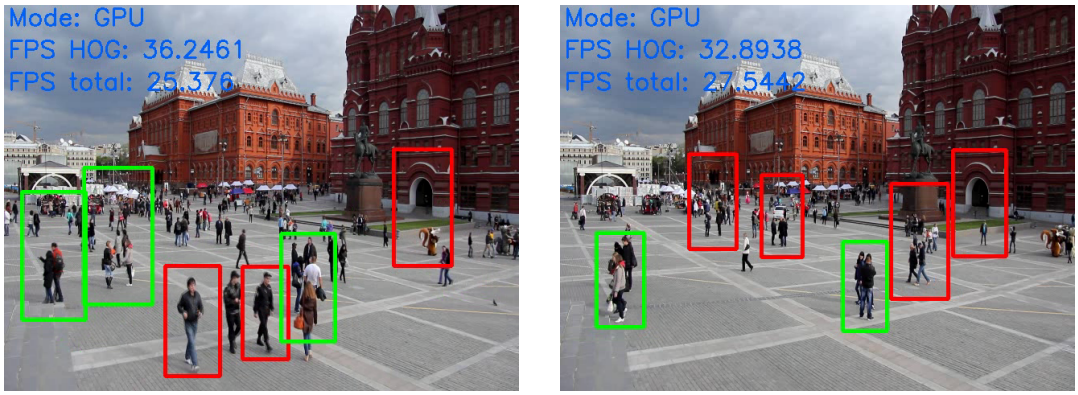
nji, detekcije unutar zadanog postotka tolerancije se iscrtavaju zelenom, a izvan zadanog postotka tolerancije crvenom bojom) te pohranjivanje koordinata detekcija (u slučaju da je argument save_coord istinit),

8. prikazivanje broja obrađenih kadrova po sekundi (engl. *FPS - frames per second*) i spremanje izlaznog videozapisa (ukoliko je argument write_video istinit),
9. procesiranje korisničkog ulaza koje omogućava dinamičku kontrolu parametara,
10. povratak na peti korak.

Završetkom funkcije run završava i izvođenje aplikacije. Kao izlazne podatke, ukoliko smo na taj način konfigurirali aplikaciju putem argumenata, aplikacija stvara datoteku s podacima koji se koriste prilikom računanja regresijske funkcije i/ili videozapis na kojem su istaknute sve detekcije pronađene prilikom rada aplikacije.

Riješeni problemi prve inačice aplikacije

Kako je opisano u dijelu 2.2.1 na stranici 29, prva inačica aplikacije je podrazumijevala nekoliko zamjetnih problema. Prvi od njih, spomenut već nekoliko puta, ticao se dinamičke promjene parametara detekcije kako bi se cijeli proces na jednostavan način mogao prilagoditi različitim vrstama videozapisa. Druga inačica aplikacije problem rješava uvođenjem metode koja procesira korisnički ulaz (sedmi korak metode run) te se na parametre u pitanju može utjecati jednostavnim pritiskom tipke. Uvođenjem ove mogućnosti se uvelike smanjuje vrijeme potrebno za pronalazak optimalnih parametara pojedinih videozapisa koji se među sobom razlikuju u broju, veličini i položaju osoba koje je potrebno detektirati. Nadalje, problem detekcije osoba izvan najčešće putanje sada je reprezentiran na mnogo pregledniji i intuitivniji način. Primjer razlikovanja "dopuštenog" i "nedopuštenog" položaja unutar kadra prikazan je na slici 2.8 na stranici 39. Kao najznačajnije poboljšanje možemo napomenuti performanse koje su poboljšane uvođenjem podrške grafičke kartice prilikom vršenja izračuna, a detaljnije su analizirane u poglavlju 4 na stranici 49.



(a) Primjer 1. (b) Primjer 2.

Slika 2.8 Primjer dopuštenih (zeleni pravokutnici) i nedopuštenih (crveni pravokutnici) detekcija na temelju njihovog položaja.

Poglavlje 3

Aplikacija za regresijsku analizu koordinata detekcija

U ovom će poglavlju biti opisan nastanak potrebe, postupak razvoja i cilj aplikacije koja provodi regresijsku analizu koordinata detekcija i prethodno je spomenuta prilikom opisa druge inačice aplikacije za detekciju osoba u dijelu 2.2.2 sa stranice 32. Ukratko, aplikacija koristi postupak regresijske analize podataka dobivenih provođenjem detekcije osoba na videozapisima za otkrivanje najčešće putanje svih detektiranih osoba.

3.1 Regresijska analiza

Regresijsku analizu koristimo prilikom promatranja određenih parametara između kojih očekujemo neku zavisnost, a ona nam nije poznata (ne posjedujemo matematički ili neki drugi izraz koji ju opisuje) [30]. Primjerice, možemo imati dva parametra, x i y, između kojih postoji veza $y = f(x)$ (slika 3.1 na stranici 42 prikazuje funkcijsku vezu koja predstavlja ovisnost koordinata položaja detektiranih osoba). Točke naravno ne moraju biti u potpunosti točne zbog mogućih pogrešaka prilikom prikupljanja podataka ili zbog toga što nam u analizi nedostaje utjecaj trećih parametara o kojima nemamo podataka [30]. [30, U tom slučaju, regresijska analiza podrazumijeva određivanje funkcijske veze $y = f(x)$ koja najbolje opisuje zavisnost

u podacima.] Postupak određivanja funkcijske veze i različiti regresijski modeli su detaljno opisani u [30].

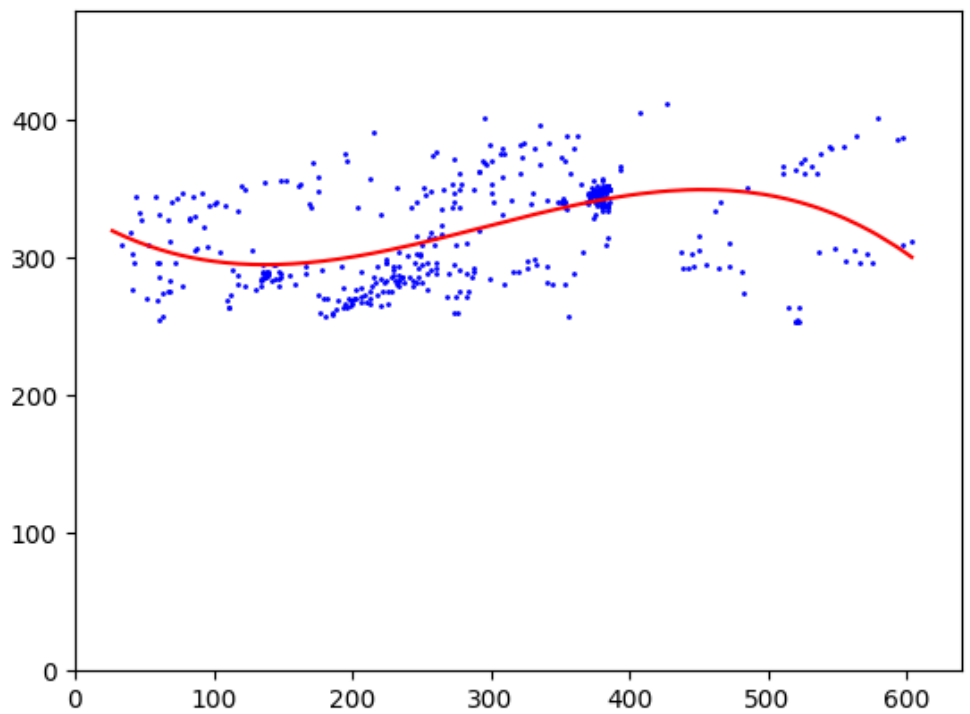
Izračunavanjem spomenute funkcijske veze dobivamo mogućnost izračuna svakog mogućeg uređenog para (x, y) što nam olakšava analizu i korištenje podataka na različite načine:

1. ne moramo se ograničiti isključivo na prikupljene podatke, već iste možemo poopćiti (primjerice, promatrajući sliku 3.1 možemo doći do zaključka da temeljem izračunate funkcijske veze možemo "stvoriti" nove detekcije – one su predstavljene crvenom krivuljom, a istovremeno predstavljaju prosječne položaje detekcija promatranog videozapisa obzirom na x koordinatu položaja detekcije),
2. na jednostavan način se ispituje pripadnost određenog podatka nekom skupu (podatak možemo shvatiti kao pojedinu detekciju – neka točka unutar koordinatnog sustava na slici 3.1, skup podataka kao sve prikupljene detekcije – plave točke slike 3.1, a pripadnost podatka skupu kao razinu podudarnosti podatka i skupa podataka – udaljenost neke točke unutar koordinatnog sustava slike 3.1 od crvene krivulje slike 3.1).

3.1.1 Potreba i cilj regresijske analize kod detekcije osoba

U dijelu 2.2.1 na stranici 29 spomenut je problem isticanja putanja pojedinih detektiranih osoba unutar promatranog videozapisa. Taj problem je riješen, iako s nešto drukčijom funkcionalnošću, u drugoj inačici aplikacije za detekciju osoba (opisano u dijelu 2.2.2 na stranici 32) upravo uz podršku aplikacije za regresijsku analizu položaja detekcija.

Potreba za provođenjem regresijske analize nad skupom položaja detekcija se pojavljuje iz želje da položaje svih detekcija prikažemo jedinstvenom funkcijskom zavisnosti koja bi ih mogla opisati na optimalan način. Funkcijska zavisnost nastala procesom regresijske analize pruža upravo to – kada je prikazana kroz cijelu domenu (širinu kadra), na optimalan način reprezentira skup položaja svih detektiranih osoba. Do te činjenice može se doći kroz nekoliko kratkih zaključaka:



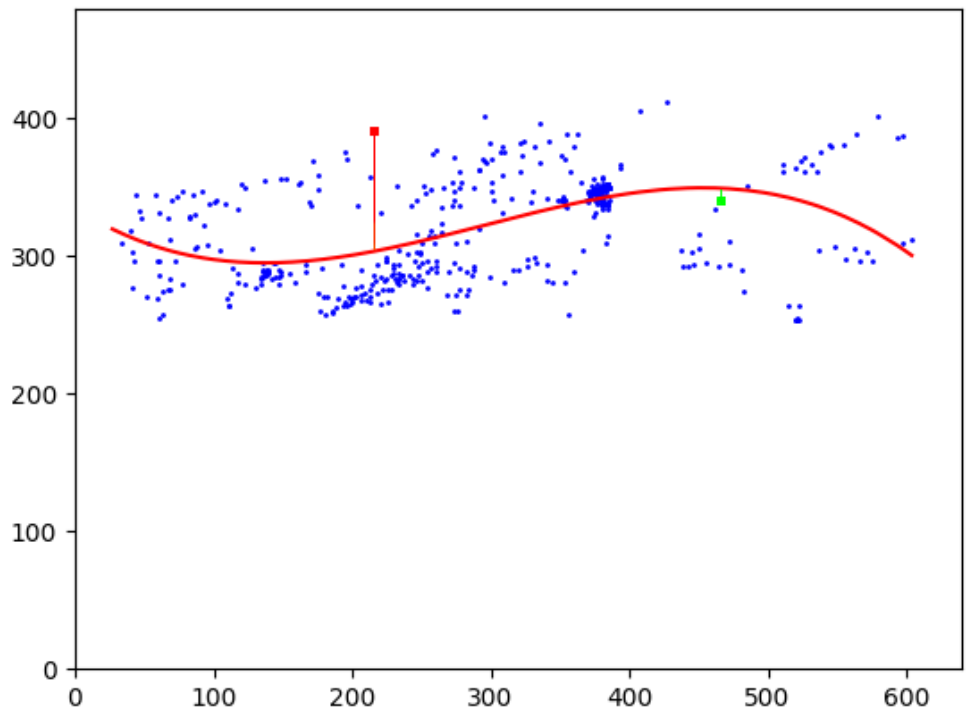
Slika 3.1 Prikaz zavisnosti parametara. Plave točke predstavljaju položaje svih detekcija unutar prozora visine y-osi i širine x-osi. Crvena krivulja predstavlja dobivenu regresijsku funkciju koja predstavlja položaje svih detektiranih osoba.

- položaji svih detekcija čine skup položaja,
- skup položaja čini putanju,
- regresijskom analizom svih detekcija dobivamo funkcijsku zavisnost koja optimalno predstavlja skup položaja, odnosno putanju svih detektiranih osoba.

Posjedovanjem takve funkcijske zavisnosti, kako je prethodno spomenuto, na jednostavan način možemo odrediti stupanj pripadnosti nekog položaja putanji (reprezentirana funkcijskom zavisnosti) koja oslikava sve detekcije. Na temelju stupnja pripadnosti možemo različitim načinima (u aplikaciji za detekciju osoba, konkretno,

crvenim i zelenim pravokutnicima) istaknuti detektirane osobe izvan i unutar dozvoljenog odstupanja od izračunate putanje. Stupanj pripadnosti se može kvantificirati kao odnos ukupne visine kadra i okomitog odstupanja određenog položaja od krivulje pri čemu je ukupna visina kadra jednolika i poznata, a okomito odstupanje se računa kao razlika y-koordinate određenog položaja i y-koordinate položaja dobivenog uvrštavanjem x-koordinate određenog položaja u regresijsku funkciju $y = f(x)$.

Primjerice, za visinu kadra H, određeni položaj (x_P, y_P) i funkcijsku zavisnot $y = f(x)$ bi stupanj podudarnosti u postocima visine kadra iznosio $\frac{|y_P - f(x_P)|}{H} * 100$. Sa slike 3.2 na stranici 44 se jasno vidi da s obzirom na ukupnu visinu kadra položaj označen crvenim kvadratom u značajno većoj mjeri odstupa od prosječne putanje detekcija nego položaj označen zelenim kvadratom. Ukoliko bi parametar tolerance opisan u prethodnom poglavlju (postotak visine kadra koji označava dozvoljeno odstupanje od prosječne putanje) bio postavljen na odgovarajući način, detekcije položaja sa slike bile bi istaknute istim bojama (detekcija predstavljena crvenim kvadratom bi bila nedopuštena, odnosno istaknuta crvenim pravokutnikom, a zelenim kvadratom dopuštena, odnosno istaknuta zelenim pravokutnikom).



Slika 3.2 Prikaz stupnja pripadnosti položaja prosječnoj putanji detekcija.

3.2 Programska podrška za analizu koordinata

U svrhu jednostavnog provođenja postupka regresijske analize nad dostupnim položajima detektiranih osoba, razvijena je jednostavna aplikacija u programskom jeziku Python. Ona na temelju datoteke stvorene od strane aplikacije za detekciju osoba provodi regresijsku analizu trećeg stupnja pri čemu stvara novu datoteku na temelju koje aplikacija za detekciju osoba može razlikovati dopuštene i nedopuštene položaje osoba. Uz to, moguće je pokrenuti i animaciju koja prikazuje evoluciju krivulje s povećanjem broja položaja koji se uzimaju u obzir prilikom regresijske analize.

3.2.1 Opis rada programa

Program započinje jednostavnom komunikacijom s korisnikom kroz komandnu ljusku prilikom koje se naznačuju lokacije ulazne i izlazne datoteke te se naznačuje želi li se pokrenuti animacija dobivanja regresijske krivulje (u tom se slučaju definira i brzina povećanja broja položaja uzetih u obzir prilikom pojedinih regresijskih analiza).

Tada se iz ulazne datoteke čitaju sve dostupne koordinate, provodi se regresijska analiza (način opisan u [30]) te se regresijski koeficijenti zapisuju u datoteku na prethodno definiranoj lokaciji. Izlazna datoteka tada može biti korištena u aplikaciji za detekciju osoba kao vrijednost parametra load_reg te će se detekcije razlikovati na temelju njihvog položaja prema boji pravokutnika koji ih ističu.

Ukoliko korisnik želi pokrenuti i animaciju, program na temelju definirane brzine povećanja broja položaja uzetih u obzir prilikom regresijske analize ulazi u petlju. Svaka iteracija petlje podrazumijeva povećanje broja promatranih položaja za definirani iznos od strane korisnika i zasebno prikazivanje regresijske krivulje koja reprezentira položaje uzete u obzir u toj određenoj iteraciji petlje. Na taj način korisnik dobiva uvid u izmjenu regresijske krivulje s pojavljivanjem novih detekcija. Primjerice, ukoliko se petlja izvršava s korakom 50, u prvoj će se iteraciji prikazati regresijska krivulja nastala na temelju prvih 50 koordinata. U sljedećoj će se iteraciji prikazati krivulja nastala na temelju prvih 100 koordinata. Tako će se krivulja uzastopno izmjenjivati sve dok se u obzir ne uzmu sve dostupne koordinate (što se događa u posljednjoj iteraciji).

3.2.2 Input i output

Aplikacije za detekciju osoba i regresijsku analizu koordinata detekcija se oslanjaju na određeni format svojih ulaznih i izlaznih datoteka kako bi ih obje mogle koristiti (ulazna datoteka jedne aplikacije je izlazna datoteka druge i obrnuto).

Ulazna datoteka aplikacije za regresijsku analizu koordinata detekcije nastaje postavljanjem parametra save_coordinates prilikom korištenja aplikacije za detekciju osoba koja prilikom svojeg rada u datoteku u pitanju sprema dimenzije kadra (prethodno spomenuto, koriste se za granice grafa animacije) i koordinate položaja

svake detektirane osobe. Dakle, format datoteke je:

- širina i visina kadra,
- x i y koordinata položaja svake detektirane osobe.

Što se tiče izlazne datoteke aplikacije za regresijsku analizu koordinata detekcije, njen je format još jednostavniji. Datoteka u pitanju se sastoji od četiri zasebna retka od kojih svaki predstavlja jedan koeficijent funkcijske zavisnosti $y = f(x)$ (uvijek sadrži četiri retka jer se uvijek koristi kubna regresijska analiza) dobivene procesom regresijske analize. Aplikacija za detekciju osoba spomenutu datoteku koristi kao ulaznu datoteku za iščitavanje koeficijenata regresijske funkcijske zavisnosti i mora biti postavljena korištenjem parametra load_reg koji predstavlja njenu lokaciju unutar datotečnog sustava računala.

Primjeri ulazne i izlazne datoteke su dani u sljedećem potpoglavlju na slikama 3.3 i 3.4.

3.3 Primjeri ulazne datoteke, izlazne datoteke i animacije

Slika 3.3 prikazuje primjer ulazne datoteke aplikacije za regresijsku analizu koordinata detekcija. Iz prvog reda možemo iščitati da se radi o kadru dimenzija $1920x1080$ što određuje veličinu grafa koji se animira. Nadalje, možemo primjetiti da su detektirane osobe na položajima $(255, 578)$, $(111, 578)$, $(810, 625)$, $(51, 642)$ itd. koje će na animaciji biti predstavljene plavim točkama. Zadnji red služi isključivo kao oznaka da u pravoj ulaznoj datoteci u sljedećim redovima mogu stajati dodatne koordinate položaja detekcija osoba.

Slika 3.4 prikazuje primjer izlazne datoteke aplikacije za regresijsku analizu koordinata detekcija. Ona jednostavno služi tome kako bi aplikacija za detekciju osoba mogla iščitati koeficijente regresijske funkcijske zavisnosti. U primjeru prikazanom slikom 3.4 regresijska funkcija $y = f(x)$ u konkretnom obliku glasi:

$$y = -7.61\text{e}{-08} * x^3 + 2.25\text{e}{-04} * x^2 - 1.44\text{e}{-01} * x + 6.64\text{e}{+02}$$

```
1920 1080
255 578
111 578
810 625
51 642
... ...
```

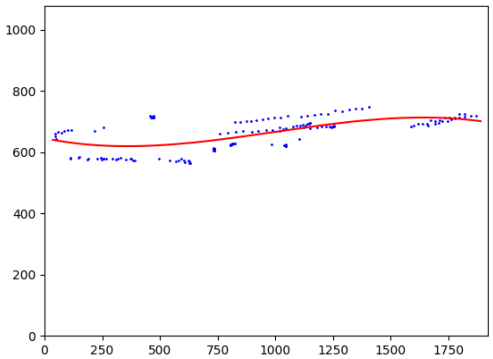
Slika 3.3 Prikaz primjera ulazne datoteke aplikacije za regresijsku analizu koordinata detekcija.

.

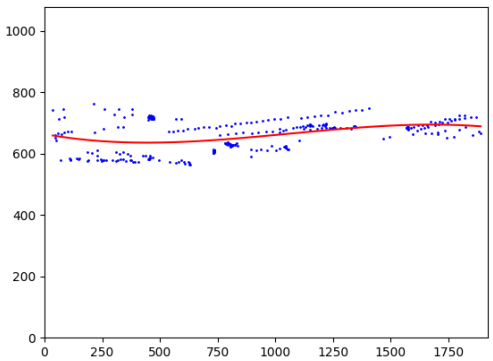
```
-7.60056785937703167 6e-08
2.25059103144088350 8e-04
-1.43738145617301110 7e-01
6.63882186597161990 0e+02
```

Slika 3.4 Prikaz primjera izlazne datoteke aplikacije za regresijsku analizu koordinata detekcija.

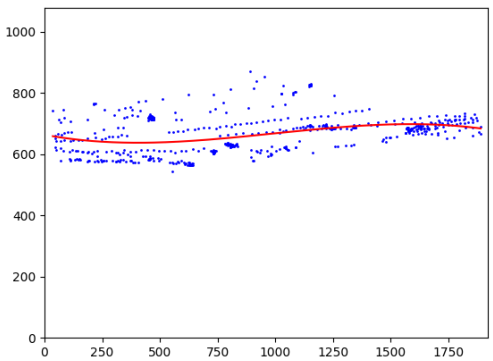
Animacija prikazana od strane aplikacije za regresijsku analizu koordinata detekcija se stvara na način opisan u prethodnom potpoglavlju, a njene četiri uzastopne iteracije prikazane su na slici 3.5.



(a) Prvih 175 točaka.



(b) Prvih 350 točaka.



(c) Prvih 525 točaka.

Slika 3.5 Prikaz animacije za regresijsku analizu na prvih 175, 350 i 525 koordinata položaja detekcija osoba.

Poglavlje 4

Analiza inačica softverskih rješenja

Svrha analize dviju prethodno opisanih inačica softverskih rješenja je vrlo jednostavna – na objektivan i jednoznačan način odlučiti koja od uspoređenih inačica na bolji način koristi dostupne resurse. Boljim korištenjem resursa podrazumijeva se učinkovitije korištenje računarske moći računala (primjenom boljih pristupa detekciji osoba), ali i iskorištavanje svih dostupnih resursa u cilju bržeg izvađanja programskog rješenja. Kroz ovo poglavlje inačice programskih rješenja uspoređene su temeljem opterećenja procesora i radne memorije računala, brzine izvođenja analize videozapisa i potrošnje električne energije tokom rada programskih rješenja.

Sva mjerenja i usporedbe izvršena su s istim ulaznim podacima na istom računalu i operacijskom sustavu s raspoloživim resursima prikazanim u tablici 1.2 na stranici 4.

4.1 Korištenje dostupnih resursa

4.1.1 Procesor i grafička kartica

Najočitija i najveća razlika u korištenju procesora i grafičke kartice računala prilikom rada prve i druge inačice aplikacije za detekciju osoba očituje se u činjenici da prva inačica aplikacije uopće ne koristi grafičku karticu (osim, naravno, u nezamjetnoj mjeri prilikom prikazivanja rezultantnih kadrova detekcije).

Unatoč primjeni istog algoritma za detekciju osoba, a zbog različite implementacije (izostanak korištenja podrške grafičke kartice kod prve inačice aplikacije), javljaju se znatne razlike u korištenju kapaciteta procesora računala prilikom rada dviju inačica aplikacije. Razlika je prikazana na slici 4.1 na stranici 52. Kao što je i očekivano, kod rada prve inačice aplikacije postotak korištenja kapaciteta procesora je mnogo veći (u prosjeku 76.08% u prvih 30 sekundi rada). Razlog tome je činjenica da procesor u prvoj inačici aplikacije sudjeluje u svim koracima analize videozapisa:

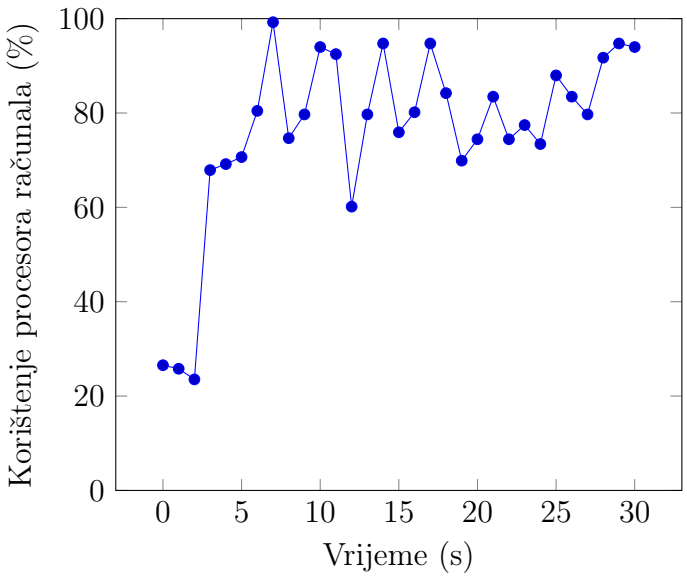
- čitanje kadrova iz ulaznih videozapisa,
- prilagodba pročitanih kadrova prije pokretanja detekcije (pretvaranje u crnobijele kadrove, smanjenje veličine...),
- detekcija osoba,
- prikazivanje rezultata i rezultantnih kadrova.

S druge strane, druga inačica aplikacije se služi podrškom grafičke kartice što dovodi do značajnih promjena u korištenju kapaciteta procesora računala. Izostanak izvršavanja algoritma histograma orijentiranih gradijenata na procesoru dovodi do velikog rasterećenja istog (u prosjeku 15.31% u prvih 30 sekundi rada).

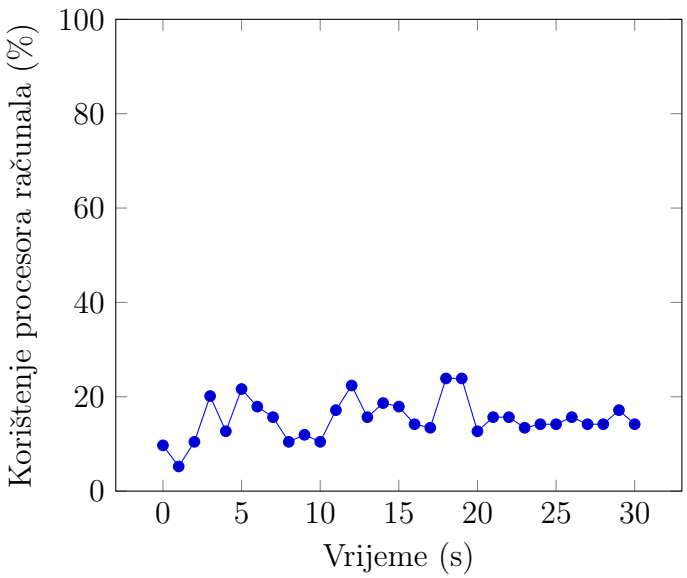
Slika 4.2 na stranci 53 prikazuje utrošak vremena aplikacije na procese različite prirode. Budući da se prva inačica aplikacije (prikaz 4.2a) oslanja na procesor kao sredstvo izvođenja algoritma, uočavamo vrlo veliku količinu vremena utrošenu na ulazno-izlazne operacije – najveći čimbenik u tome je konstantna komunikacija procesora i radne memorije prilikom provođenja detekcije. Nasuprot tome, druga inačica aplikacije (prikaz 4.2b) ulazno-izlazne operacije umanjuje za 62% budući da radna memorija u navedenoj inačici služi kao međuspremnik prilikom komunikacije procesora i grafičke kartice računala. Shodno tome, operacije kroz jezgru računala se povećavaju za 32.9% (posljedica međudjelovanja i komunikacije procesora i grafičke kartice računala). Posljednja činjenica koju možemo uočiti usporedbom dvaju grafova sa slike 4.2 je porast utroška vremena druge inačice aplikacije na operacije koje se odvijaju na grafičkoj kartici za 24.6%. Razlog tome je trivijalan i naveden na početku odlomka 4.1.1 sa stranice 49.

4.1.2 Radna memorija

Analizom količine korištene radne memorije prilikom izvođenja pojedinih inačica aplikacija ne utvrđujemo bitne razlike. Podaci o količinu korištene radne memorije su prikazani na slici 4.3 na stranici 54. Prva inačica aplikacije kroz prvih 30 sekundi rada prosječno koristi 1702.58 MB radne memorije dok druga inačica aplikacije u istom vremenskom razdoblju koristi nezamjetno viših (4.06%) 1771.77 MB radne memorije.

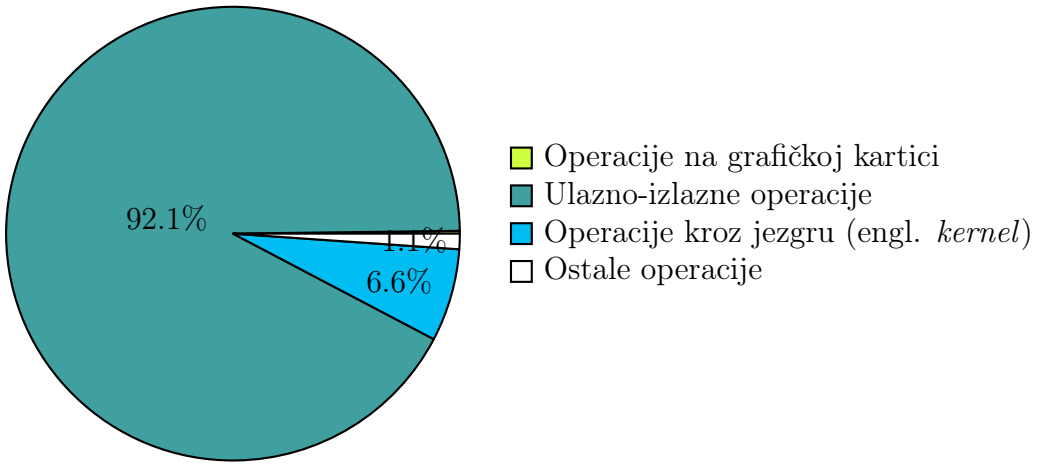


(a) Prikaz postotka korištenja procesora računala tijekom rada prve inačice aplikacije.

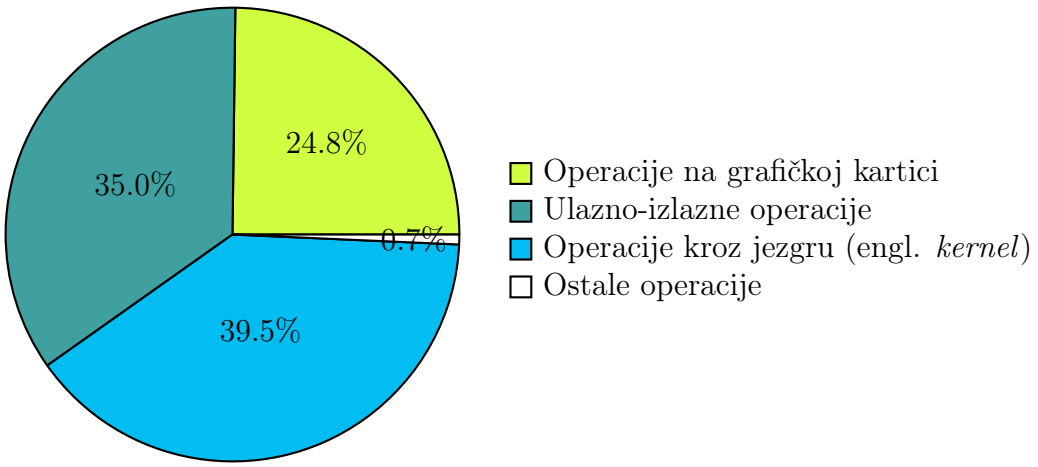


(b) Prikaz postotka korištenja procesora računala tijekom rada druge inačice aplikacije.

Slika 4.1 Postotak korištenja kapaciteta procesora računala kroz vrijeme od strane aplikacija za detekciju osoba.

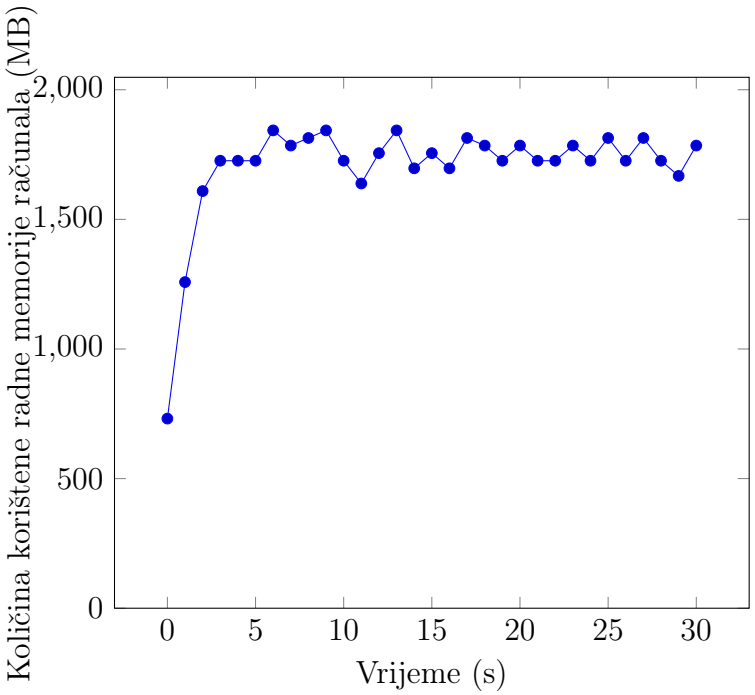


(a) Prikaz utroška vremena prve inačice aplikacije.

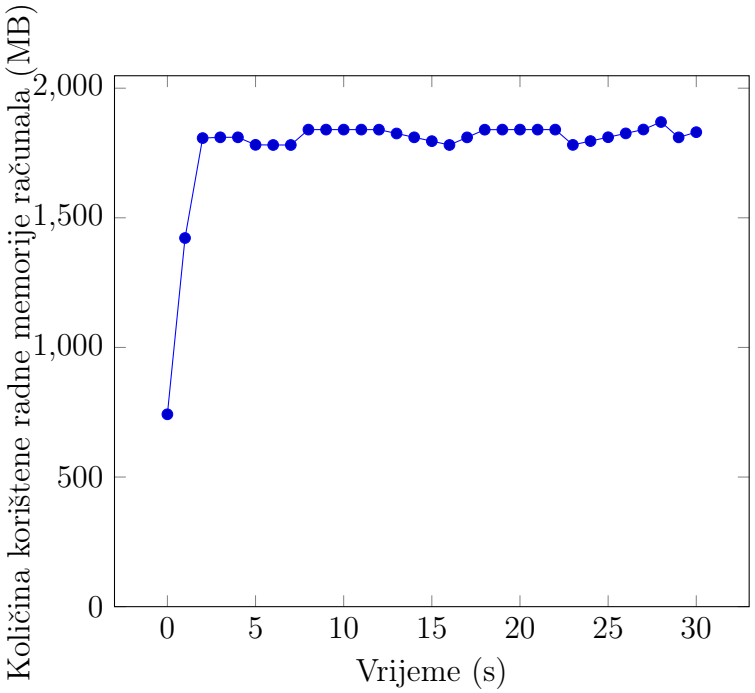


(b) Prikaz utroška vremena druge inačice aplikacije.

Slika 4.2 Utrošak vremena na procese različitih kategorija prilikom izvađanja aplikacija za detekciju osoba.



(a) Prikaz količine korištene radne memorije računala tijekom rada prve inačice aplikacije.



(b) Prikaz količine korištene radne memorije računala tijekom rada druge inačice aplikacije.

Slika 4.3 Količina korištene radne memorije računala kroz vrijeme od strane aplikacija za detekciju osoba.

4.2 Brzina obrade kadrova

Brzina obrade kadrova u najvećoj mjeri utječe na brzinu izvođenja programa i izražava se u kadrovima po sekundi (KPS, engl. *FPS - Frames Per Second*). Podaci o brzini obrade kadrova su prikupljeni na način da su zapisani u tekstualnu datoteku od strane modificiranih aplikacija za detekciju osoba. U svrhu boljeg shvaćanja prikupljenih podataka, razlikujemo brzinu obrade kadra pod koju spada cijeli proces obrade (čitanje i priprema kadra te detekcija osoba) i brzinu obrade kadra pod koju spada samo proces detekcije (algoritam histograma orijentiranih gradijenata).

Na slici 4.4 sa stranice 57 prikazani su prikupljeni podaci o brzini obrade kadrova. Možemo primjetiti da druga inačica aplikacije koja koristi podršku grafičke kartice kadrove u oba slučaja obrađuje mnogo brže (cjelokupan proces s prosječnih 8.09KPS, a proces detekcije s prosječnih 8.80KPS) od prve inačice aplikacije (cjelokupan proces s prosječnih 3.38KPS, a proces detekcije s prosječnih 3.48KPS). Treba napomenuti da je brzina obrade kadrova prilično mala zbog toga što su podaci prikupljeni prilikom obrade kadrova širine 1920 piksela i visine 1080 piksela (tipično se kadrovi tako velikih dimenzija smanjuju radi povećanja brzine što je prikazano u sljedećem primjeru). Iako se na prvi pogled razlika brzine obrade kadrova od tek nešto više od 5KPS čini mala, druga inačica aplikacije je zapravo cjelokupan proces obrade kadrova ubrzala za 139.36% što je izuzetno veliko ubrzanje. Još jedna stvar koju možemo primjetiti je veće odstupanje brzine cjelokupne obrade kadrova od brzine procesa detekcije nad kadrovima u slučaju druge inačice aplikacije. Cjelokupna obrada kadrova je od procesa detekcije nad kadrovima u slučaju prve inačice aplikacije sporija za 2.72% nasuprot 8.01% kod druge inačice aplikacije. Takvo razlikovanje usporavanja možemo pripisati troškovima prijenosa kadrova i komunikaciji između procesora i grafičke kartice u slučaju druge inačice aplikacije.

Slika 4.5 sa stranice 58 prikazuje istu vrstu podataka kao prethodno spomenuta slika 4.4, ali uz uključenu opciju smanjenja kadrova prilikom procesa pripreme kadra za detekciju na kadrove širine 640 piksela i visine 480 piksela. Trivijalno je za shvatiti da je upravo smanjenje veličine kadrova pridonjelo iznimno velikom povećanju brzine – 501.68% u slučaju prve inačice aplikacije i 319.64% u slučaju druge inačice aplikacije. Dobitak na brzini druge inačice u odnosu na prvu inačicu aplikacije

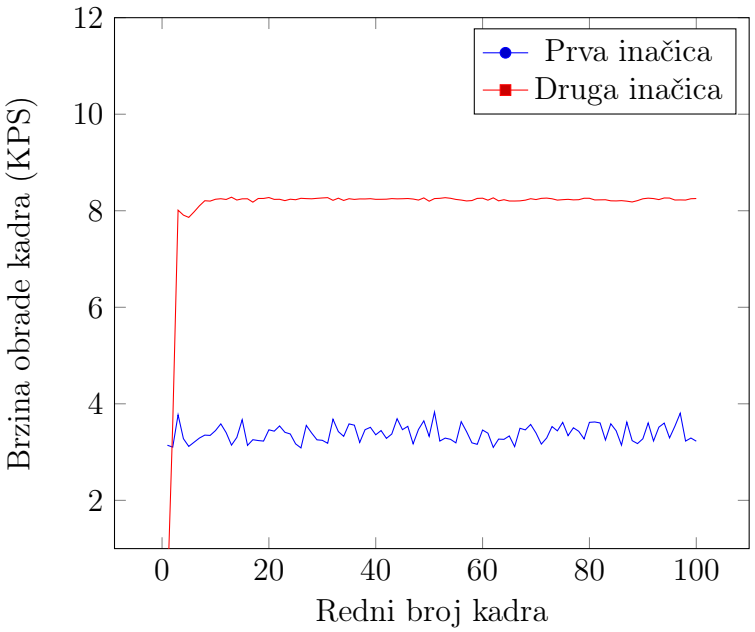
sada iznosi 66.94% u odnosu na prijašnjih 139.36%. Neproporcionalno povećanje brzine i manji dobitak na brzini obrade kadrova ponovno možemo pripisati troškovima prijenosa kadrova i komunikaciji između procesora i grafičke kartice u slučaju druge inačice aplikacije. Sada je spomenuti trošak puno utjecajniji zbog toga što su kadrovi manjih dimenzija pa je dobitak na brzini uslijed provođenja algoritma histograma orijentiranih gradijenata na grafičkoj kartici računala manji (brzina procesa detekcije uz podršku grafičke kartice je sada 92.43% brža u odnosu na prijašnjih 153.13%).

Dakle, podrška grafičke kartice prilikom cjelokupnog procesa, a pogotovo prilikom procesa detekcije nad kadrovima donosi velika ubrzanja. Unatoč tome, trošak komunikacije i prijenosa datoteka između procesora i grafičke kartice računala koji ovisi o dimenzijama kadrova videozapisa nije nezamjetan te na njega svakako treba obratiti pozornost.

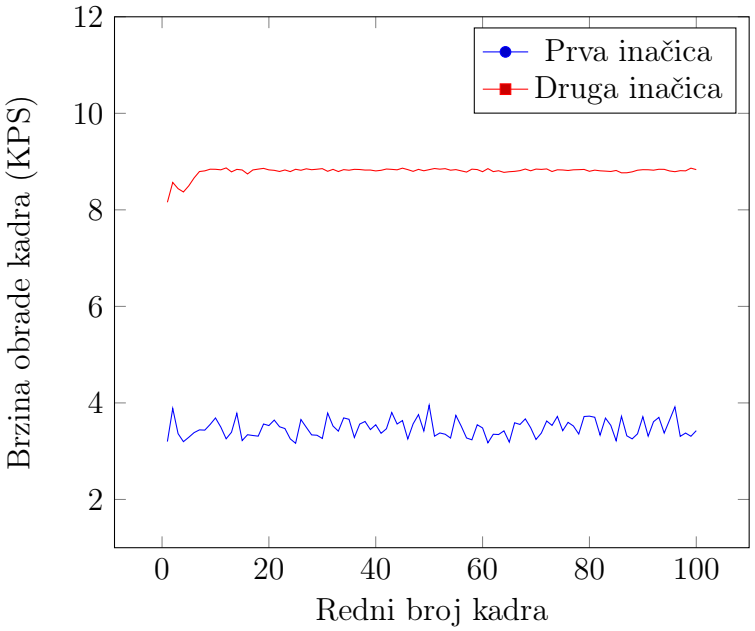
Za postizanje najvećeg dobitka pomoću ubrzanja grafičkom karticom dimenzije kadra promatranog videozapisa moraju biti postavljene na način da se parametri koji utječu na ubrzanje nalaze u optimalnom odnosu:

- postotak ubrzanja detekcije na grafičkoj kartici u odnosu na detekciju prve inačice aplikacije mora biti što veći (što je kadar veći, postotak ubrzanja je veći),
- utrošak prilikom prijenosa kadra te komunikacije sklopovlja mora biti što manji (što je kadar manji, utrošak je manji).

Potrebno je pronaći dimenzije kadra promatranog videozapisa za koje spomenuti parametri donose najveću prednost u odnosu na prvu inačicu aplikacije.

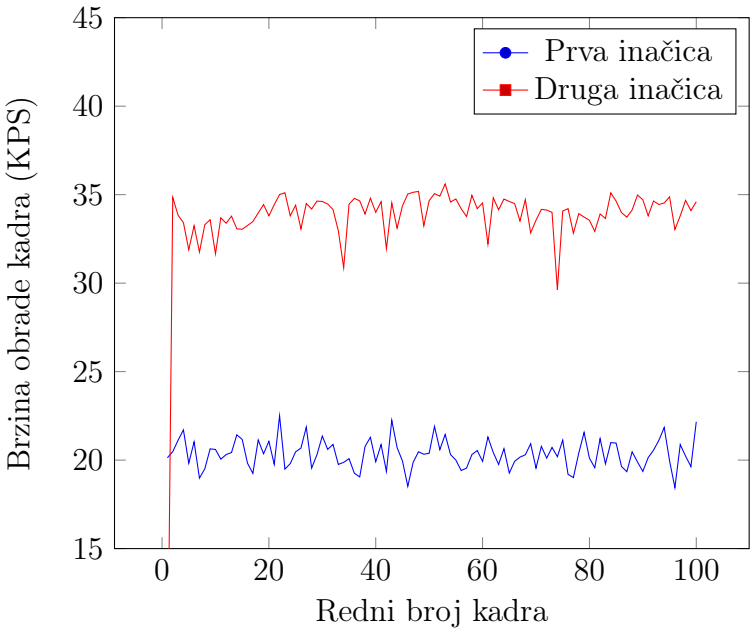


(a) Usporedba brzine cjelokupne obrade kadrova različitih inačica aplikacije.

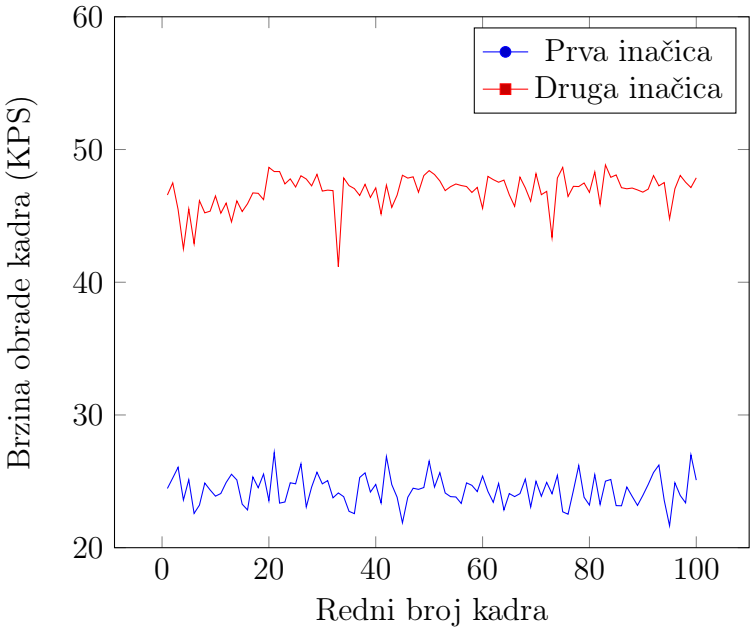


(b) Usporedba brzine obrade (uključen samo proces detekcije) kadrova različitih inačica aplikacije.

Slika 4.4 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba.



(a) Usporedba brzine cjelokupne obrade kadrova različitih inačica aplikacije.



(b) Usporedba brzine obrade (uključen samo proces detekcije) kadrova različitih inačica aplikacije.

Slika 4.5 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba (smanjena veličina kadrova tijekom procesa pripreme).

4.3 Energetska učinkovitost

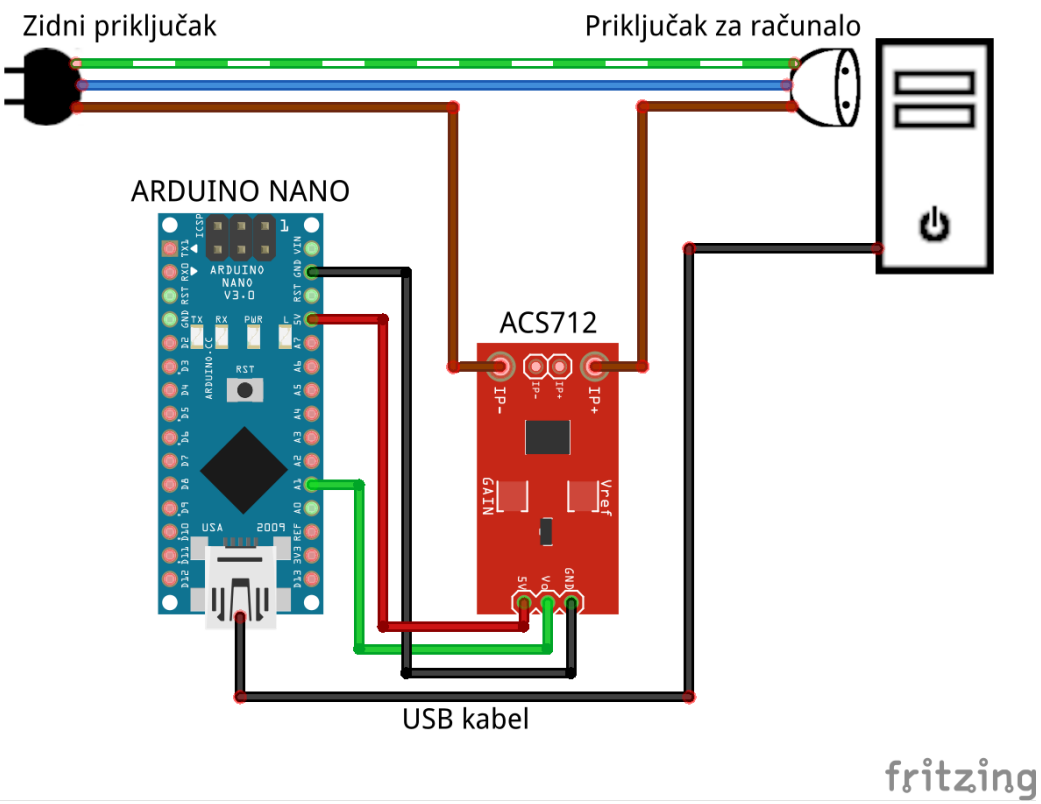
Budući da nije bilo moguće naći valjanu programsku podršku za mjerenje potrošnje električne energije, odlučeno je implementirati svoje programsko rješenje s podrškom pripadajućeg sklopovlja. Vrlo jednostavno sklopovlje se sastoji od mikrokontrolera Arduino Nano, senzora ACS712 koji mjeri jakost električne struje i još nekoliko dodatnih dijelova (žice, utikač i slično). Mikrokontroler je konfiguriran na način da vrijednosti senzora interpretira kao električnu energiju i zatim ih šalje na računalo njihovog zajedničkog USB (engl. *Universal Serial Bus*) priključka. Program napravljen u Pythonu tada prikazuje primljene podatke unutar dijagrama u ovisnosti o vremenu.

Proces mjerenja se pokreće na jednostavan način – jedna strana alata za mjerenje se spaja na priključak za električnu energiju, a druga strana alata služi kao utikač za napajanje računala. Nakon toga, preostaje povezati računalo i alat za mjerenje putem USB kabela te pokrenuti Python program. Prikaz alata za mjerenje potrošnje električne energije te način spajanja na računalo su dani na slikama 4.6 sa stranice 60 i 4.7 sa stranice 62.

4.3.1 Usporedba provedenih mjerenja

Prilikom provođenja mjerenja potrošnje električne energije, operacijski sustav je pokrenut bez dodatnih korisničkih aplikacija. Cilj toga je pokušaj minimiziranja utjecaja aplikacija koje se odvijaju bez našeg utjecaja. Također, aplikacija za provođenje mjerenja je pokrenuta na drugom računalu kako njeno izvođenje i napajanje sklopovlja alata za mjerenje potrošnje električne energije ne bi utjecalo na točnost mjerenja.

S usporedbe prikazane na slici 4.8 sa stranice 63 lako se iščita da druga inačica aplikacije troši nešto više snage s obzirom na vrijeme prilikom rada u odnosu na prvu inačicu aplikacije. Računalo u mirovanju u prosjeku troši 114.36W, tijekom rada prve inačice aplikacije 146.67W, a tijekom rada druge inačice aplikacije za detekciju osoba 154.76W. Dakle, prilično je jasno da prva inačica aplikacije kroz vrijeme prosječno troši 32.30W, a druga inačica 40.40W. Prema tome, zahtjevi za snagom aplikacije ubrzane podrškom grafičke kartice su u prosjeku viši za 25.08% (kada se



Slika 4.6 Prikaz rasporeda komponenata alata za mjerenje potrošnje električne energije.

potrošnja promatra u ovisnosti o vremenu). Treba napomenuti da potrebna snaga uvelike ovisi o konfiguraciji računala (npr. energetska efikasnost procesora i grafičke kartice) i da bismo mjerenjem potrošnje električne energije iste aplikacije na drukčije konfiguriranom sklopovlju zasigurno dobili različite rezultate.

Uz prikaz izmjerenih podataka u ovisnosti o vremenu, iznimno je korisno spomenute podatke prikazati i u ovisnosti o rednom broju kadra koji se obrađuje. Na taj način ne promatramo apsolutnu snagu koju programska podrška zahtijeva, već zahtjevanu snagu uz vrijeme dovodimo i u vezu s odrađenim poslom. Drugim riječima, tek takvom usporedbom dobivamo ispravnu predodžbu o efikasnosti i štedljivosti inačica programske podrške.

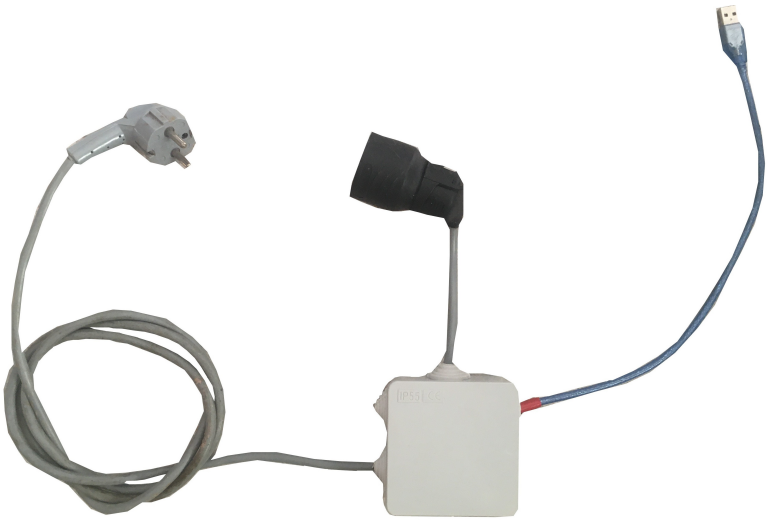
Primjerice, sa slike 4.8 na stranici 63 se dobiva dojam da prva inačica svoj posao

odvija na efikasniji način u osnosu na drugu inačicu. Sličan dojam steći će se i promatranjem dijagrama sa slike 4.9 na stranici 64. Naravno, to je ispravan zaključak jedino ukoliko se pod efikasnošću podrazumijeva apsolutna potrošnja energije kroz neki vremenski interval $[t_1, t_2]$, a količina odrađenog posla zanemaruje. Ukoliko u izračun efikasnosti želimo uračunati i količinu obavljenog posla, potrebu za snagom, osim s vremenom, moramo dovesti i u vezu s brojem obrađenih kadrova (izvršenim poslom).

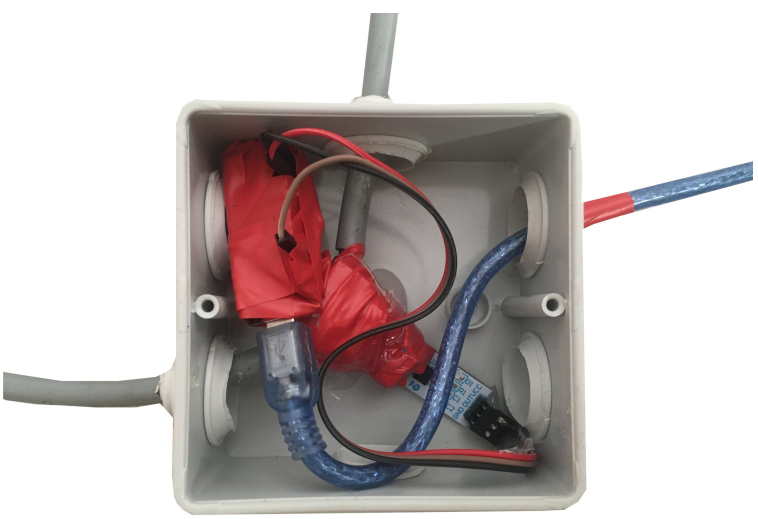
Slika 4.10 sa stranice 65 prikazuje opisanu relaciju kumulativne potrebne snage i kumulativnog izvršenog posla (obrađenih kadrova) kroz vrijeme. Efikasnost pojedine inačice se vrednuje na jednostavan način – što je potreba za snagom po jedinici posla (obrađenom kadru) manja, to je inačica efikasnija (štedljivija). Na temelju toga se može zaključiti da je efikasnost druge inačice aplikacije za detekciju osoba znatno veća.

Precizniji način kojim se razumljivije može ilustrirati veća efikasnost druge inačice rješenja je prikazan na slikama 4.11 sa stranice 66 i 4.12 sa stranice 67. Krivulje navedenih slika prikazuju interpolaciju prikupljenih diskretnih podataka o snazi potrebnoj prilikom izvođenja aplikacije u promatranom vremenskom intervalu od 30 sekundi. Na spomenutim slikama mogu se uočiti i reprezentacije integrala snage po vremenu koji podrazumijevaju količinu energije iskorištene za obavljanje posla u promatranom vremenskom intervalu. Integral prve inačice aplikacije za detekciju osoba iznosi 1098.43Ws (0.31Wh), dok integral druge inačice iznosi 1162.59Ws (0.32Wh). Uz integrale je iz prikupljenih podataka poznat i broj obrađenih kadrova unutar promatranog vremenskog intervala – prva inačica aplikacije je obradila 97 kadrova, a druga inačica je obradila 234 kadra. Jednostavnim izračunom saznajemo da je prva inačica utrošila 11.32Ws po obradi jednog kadra, dok je druga inačica za isti posao utrošila 4.97Ws.

Lako se zaključuje da je druga inačica aplikacije za detekciju osoba efikasnija (štedljivija) za 127.77% u odnosu na prvu inačicu aplikacije. Osim toga, druga inačica aplikacije za detekciju osoba je prosječno brža za 139.36% u odnosu na prvu inačicu (detaljno opisano u dijelu 4.2 na stranici 55).

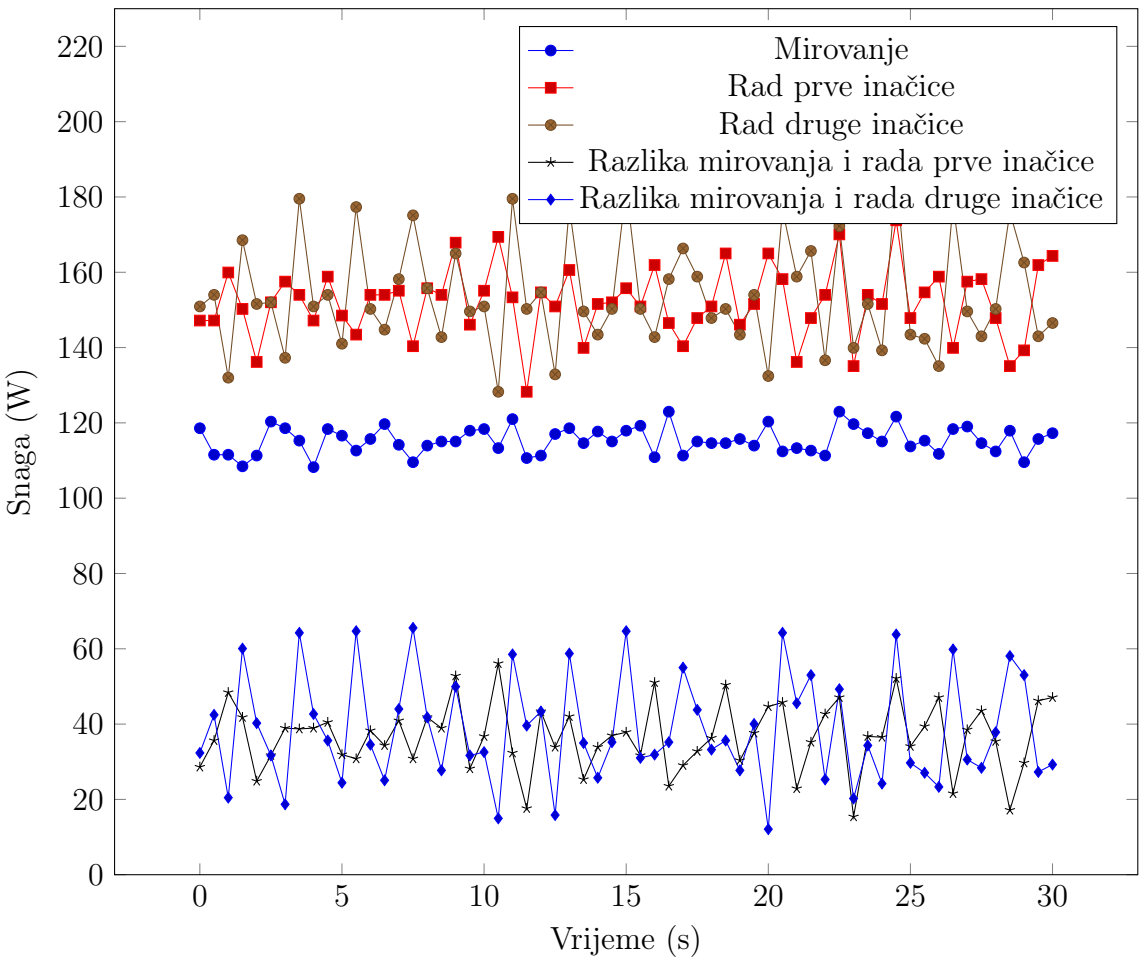


(a) Vanjski izgled alata.



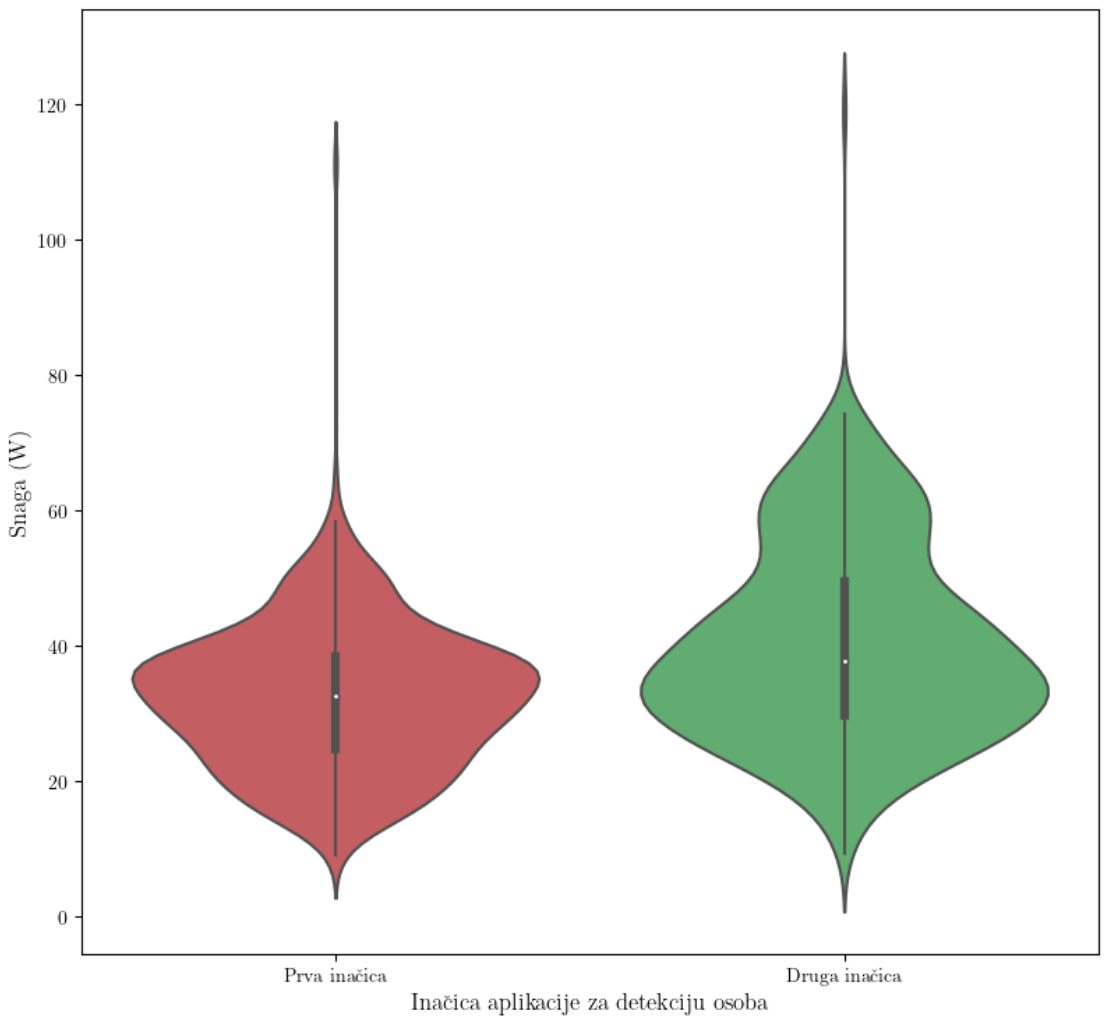
(b) Unutarnji raspored komponenata alata.

Slika 4.7 Prikaz alata za mjerenje potrošnje električne energije.

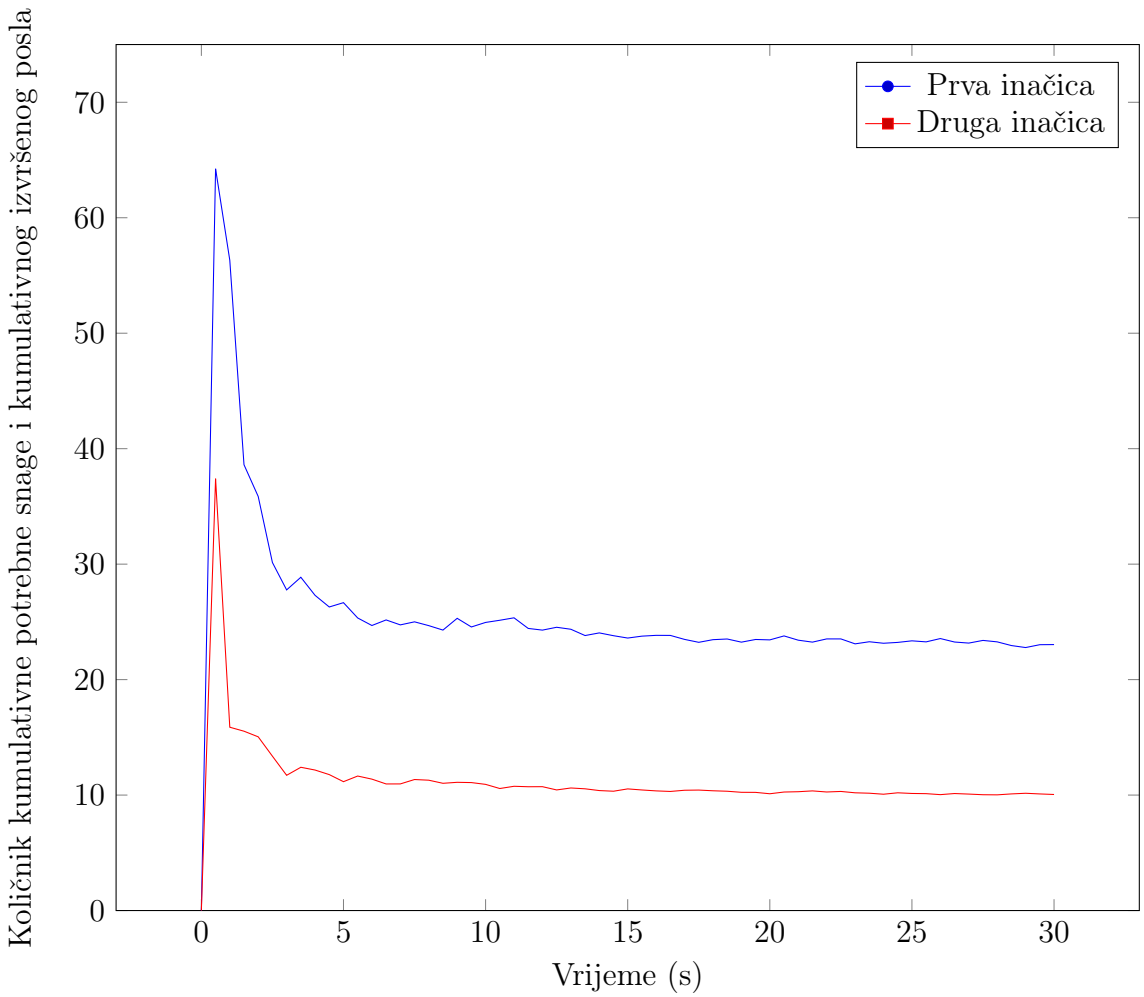


(a) Usporedba potrebne snage u različitim situacijama.

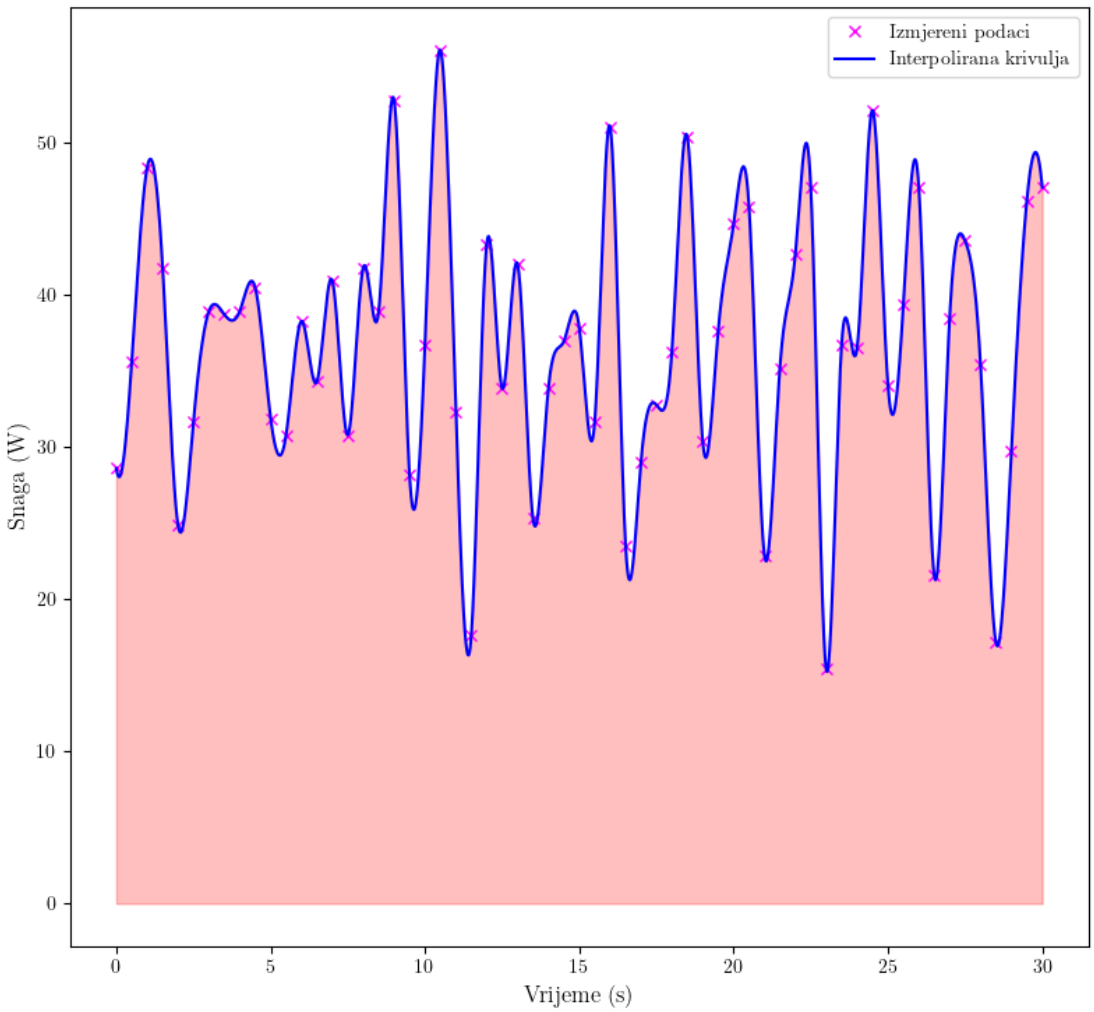
Slika 4.8 Potrebna snaga od strane računala prije i tijekom rada pojedinih inačica aplikacije za detekciju osoba.



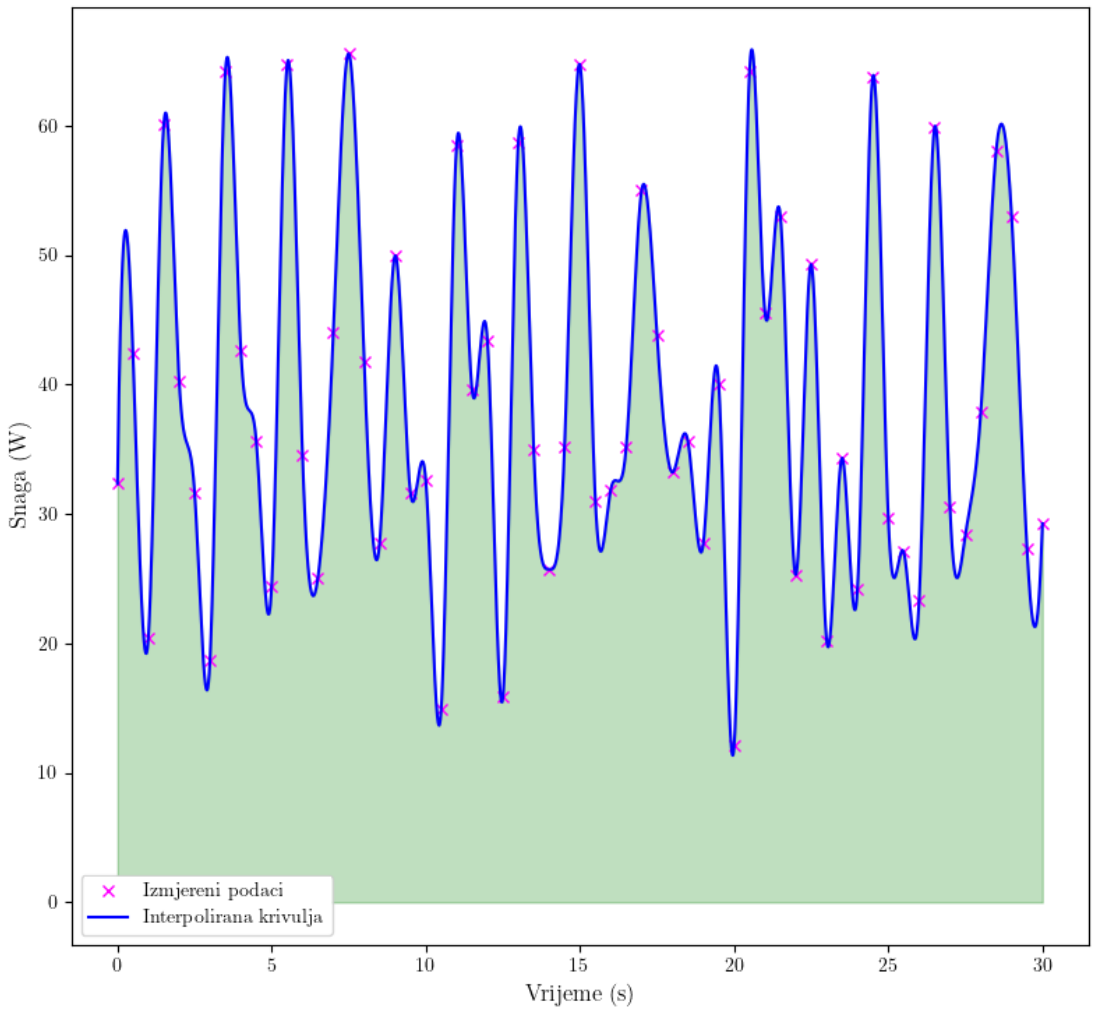
Slika 4.9 Prikaz potrebe za snagom kroz učestalost pojedinih vrijednosti za različite inačice aplikacije za detekciju osoba.



Slika 4.10 Prikaz odnosa kumulativne potrebne snage i kumulativnog izvršenog posla kroz vrijeme za različite inačice aplikacije.



Slika 4.11 Prikaz odnosa potrošnje energije pomoću integrala snage kroz vrijeme za prvu inačicu aplikacije.



Slika 4.12 Prikaz odnosa potrošnje energije pomoću integrala snage kroz vrijeme za drugu inačicu aplikacije.

Poglavlje 5

Pokretanje detekcije na superračunalu BURA

Budući da je tema završnoga rada usko povezana s računarstvom visokih performansi, razvijeno programsko rješenje je pokrenuto na superračunalu BURA. U sljedeća dva potpoglavlja opisani su procesi koji prethode pokretanju programske podrške na superračunalu te usporedba rada iste na superračunalu i osobnom računalu.

5.1 Pokretanje programske podrške na superračunalu

Prije pokretanja sâme programske podrške, bilo je potrebno osigurati da su u okruženju za pokretanje iste dostupne sve programske knjižnice navedene u poglavlju 2 s početkom na stranici 14. U svrhu ispunjenja toga cilja, uz već dostupnu Nvidia CUDA-u na superračunalu, još je bilo potrebno instalirati knjižnicu programskih funkcija OpenCV.

Iako se to čini kao jednostavan zadatak, na superračunalu se svi paketi specifični pojedinom korisniku instaliraju isključivo u njegovu vlastitu, korisničku mapu bez administratorskih privilegija. Dakle, svu potrebnu programsku podršku je potrebno preuzeti u obliku izvornog kôda i tada ju na osnovi preuzetoga instalirati.

Problemi se pojavljuju prilikom ovisnosti pojedinih programskih podrški za drugim paketima koje treba instalirati na isti način. Tako se stvara rekurzivan proces instalacije koji, makar u ovom slučaju, prilično dugo traje. Cijeli proces završava tek kada se instalira sva programska podrška koja predstavlja svojevrstan preduvjet za instalaciju OpenCV-a (OpenCV je moguće instalirati i bez dodataka, ali u tom slučaju nedostaju u ovom slučaju potrebne funkcionalnosti). Nakon uspješne instalacije knjižnice programskih funkcija OpenCV i učitavanja već dostupnog Nvidia CUDA modula, možemo sastaviti (engl. *compile*) razvijenu programsku podršku i krenuti na sljedeće korake koji prethode pokretanju.

Pokretanje programa na superračunalu je, naravno, nešto složenije nego kod osobnog računala. Na umu treba imati da se superračunalom koristi velik broj ljudi od kojih svaki želi pokrenuti vlastitu programsku podršku. Superračunalo BURA u svrhu učinkovitog i pravednog upravljanja resursima koristi SLURM (engl. *Simple Linux Utility for Resource Management* – jednostavan Linux alat za upravljanje resursima). SLURM je sustav za upravljanje resursima na velikim i malim Linux klasterima (niz računala koja su povezana i funkcioniraju kao cjelina) [31]. Autori SLURM-a ističu njegove tri glavne funkcionalnosti [31]:

1. korisnicima rezervira prava na korištenje određenih resursa za izvršavanje programa u nekom vremenskom periodu,
2. pruža okvir za pokretanje, izvršavanje i nadzor poslova na rezerviranim resursima,
3. rješava konflikte nastale prilikom podnošenja zahtjeva za resursima od strane različitih korisnika na način da stvara i upravlja redom poslova koje treba izvršiti.

SLURM-om se koristimo kroz sučelje naredbenog retka (engl. *CLI – Command Line Interface*) uz pomoć nekoliko prilično intuitivnih naredbi od kojih svaka započinje prefiksom 's' koji označava da se radi o SLURM naredbi. Postoji nekoliko osnovnih naredbi [31]:

- scontrol – služi za izvršavanje administrativnih naredbi poput isključivanja čvora unutar superračunala; mnoge se funkcije mogu izvršiti isključivo od

strane privilegiranih korisnika,

- sinfo – služi za prikaz informacija o particijama i čvorovima superračunala; podržava razne oblike ispisa i filtriranje prema različitim kategorijama,
- squeue – služi za prikaz informacija o poslovima koji se trenutno izvađaju i o poslovima koji čekaju na izvršavanje; podržava razne oblike ispisa te filtriranje i sortiranje prema različitim kategorijama,
- srun – služi za alokaciju resursa i podnošenje poslova,
- scancel – služi za otkazivanje određenog posla.

Program na superračunalu najlakše je pokrenuti koristeći SLURM skriptu. Ona u sebi sadrži sve parametre koji su SLURM-u potrebni za pokretanje nekog zadatka. Definiranju svakog parametra prethodi oznaka #SBATCH koja označava da se radi o SLURM parametru. SLURM skripta korištena za pokretanje programske podrške detekcije osoba dana je u prikazu 5.1 na stranici 70. Skripta u sebi definira nekoliko stvari koristeći prilično intuitivne parametre:

- ––job–name – ime posla koji zadajemo na izvršavanje (biti će prikazano u ispisu squeue naredbe),
- ––time – označava maksimalnu količinu vremena na koju rezerviramo resurse,
- ––nodes – označava broj čvorova superračunala koje želimo rezervirati,
- ––mem–per–cpu – označava količinu radne memorije po procesoru koju želimo rezervirati,
- ––partition – označava particiju superračunala na kojoj želimo rezervirati resurse,
- –o – označava ime izlazne datoteke.

Treba napomenuti da prikaz 5.1 na stranici 70 prikazuje samo jednu od mogućnosti stvaranja SLURM skripte. Može se, primjerice, povećati broj čvorova korištenih za izvađanje posla, ali je u slučaju aplikacije za detekciju osoba brzina izvođenja ostala gotovo identična. Nakon stvaranja SLURM skripte, jedino što preostaje je pokrenuti istu pomoću naredbe sbatch IME_SKRIPTE.sh.

Prikaz programskog kôda 5.1 SLURM skripta za pokretanje programske podrške.

```
1  #!/bin/bash
2
3  #SBATCH --job-name=gpuhog
4  #SBATCH --time=00:05:00
5  #SBATCH --nodes=1
6  #SBATCH --mem-per-cpu=16348MB
7  #SBATCH --partition=comp_gpu
8  #SBATCH -o gpuhog.out
9
10 ./gpuhog --video gpuhog_input.mov
```

5.2 Usporedba superračunala i osobnog računala

Budući da podaci o korištenju sklopovlja i utrošku energije superračunala nisu dostupni, ova se usporedba koncentrira na brzinu obrade kadrova čije je značenje i način prikupljanja opisano u potpoglavlju 4.2 na stranici 55.

Prije prezentacije rezultata usporedbe treba navesti konfiguraciju čvora superračunala na kojem je pokrenuto izvođenje razvijene programske podrške za detekciju osoba na videozapisu:

- čvor sadrži dva procesora Intel Xeon E5-2650 v2,
- čvor sadrži dvije grafičke kartice Nvidia TESLA K40.

Također treba napomenuti da je prilikom prikupljanja podataka sa slike 5.1 na stranici 73 korišten ulazni videozapis širine kadra 3840 piksela i visine kadra 2160 piksela, a prilikom prikupljanja podataka sa slike 5.2 na stranici 74 videozapis širine kadra 640 piksela i visine kadra 480 piksela.

Slika 5.1 na stranici 73 prikazuje odnos brzine obrade kadrova između:

- osobnog računala,
- superračunala,

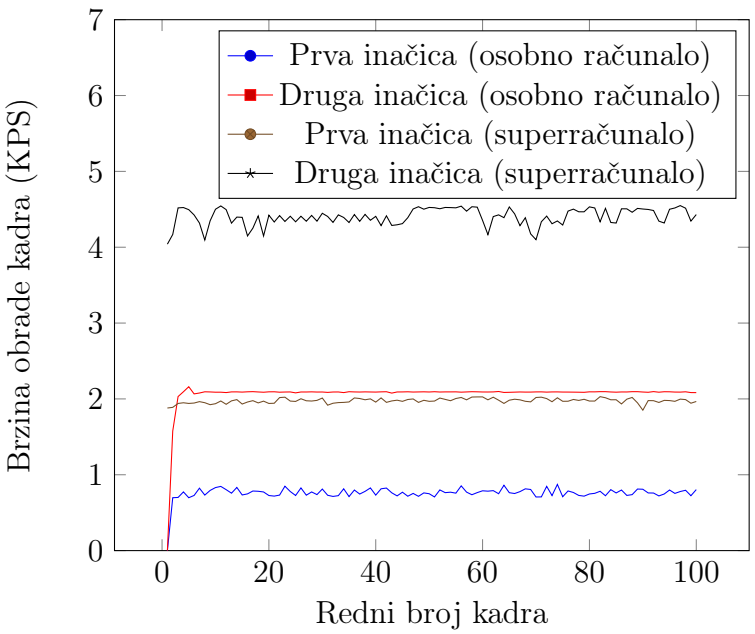
- osobnog računala uz podršku grafičke kartice,
- superračunala uz podršku grafičke kartice.

Rezultati se do neke mjere mogu okarakterizirati kao očekivani – superračunalo je u oba prikaza sa slike 5.1 na stranici 73 prikazalo svoju nadmoć nad osobnim računalom. Prilikom izvođenja prve inačice programske podrške za detekciju osoba, superračunalo proces detekcije nad kadrovima izvađa 160,09% brže od osobnog računala, a prilikom izvođenja druge inačice programske podrške superračunalo proces detekcije nad kadrovima izvađa 108,66% brže od osobnog računala. Faktori ubrzanja superračunala nad osobnim računalom prilikom promatranja cjelokupnog procesa obrade kadrova (razlika opisana u potpoglavlju 4.2 na stranici 55) su veoma slični faktorima ubrzanja tijekom procesa detekcije – 160.22% za prvu inačicu aplikacije i 113.16% za drugu inačicu aplikacije.

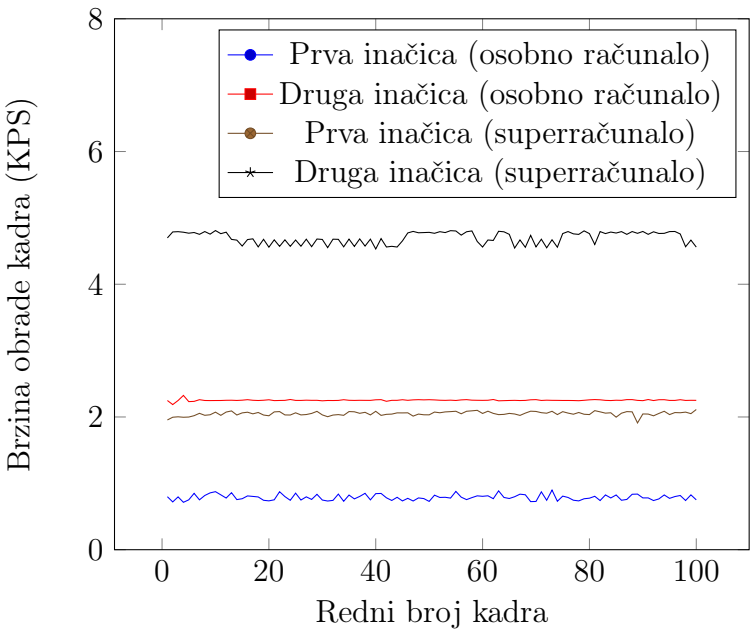
Promatranjem slike 5.2 sa stranice 74 još jednom se uviđa prednost superračunala u brzini obrade kadrova. Sukladno vrsti mjerenog procesa (cjelokupan ili isključivo detekcija) i pokrenutoj inačici aplikacije faktori ubrzanja superračunala u odnosu na osobno računalo iznose:

- 135,52% za cjelokupan proces obrade kadrova prve inačice aplikacije,
- 91,98% za cjelokupan proces obrade kadrova druge inačice aplikacije,
- 167,50% za proces detekcije prve inačice aplikacije,
- 41,65% za proces detekcije druge inačice aplikacije.

Iako superračunalo nedvojbeno postiže bolje rezultate u odnosu na osobno računalo, bilo je za očekivati da će brzina obrade kadrova biti višestruko, a ne tek dvostruko veća. Izostanak opravdanja takvog očekivanja možemo pripisati neoptimiziranosti aplikacije za detekciju osoba prilikom korištenja više računalnih čvorova ili pak donekle zastarjelom sklopovlju superračunala. Procesor i grafička kartica čvora na kojem su prikupljeni rezultati su proizvedeni 2013. godine i njihove performanse nisu višestruko bolje od sklopovlja osobnog računala. Za donošenje pouzdanog zaključka o izostanku višestruko veće brzine obrade kadrova potrebno je provesti detaljne analize implementacije histograma orijentiranih gradijenata i bolje se upoznati s mogućnostima superračunala.

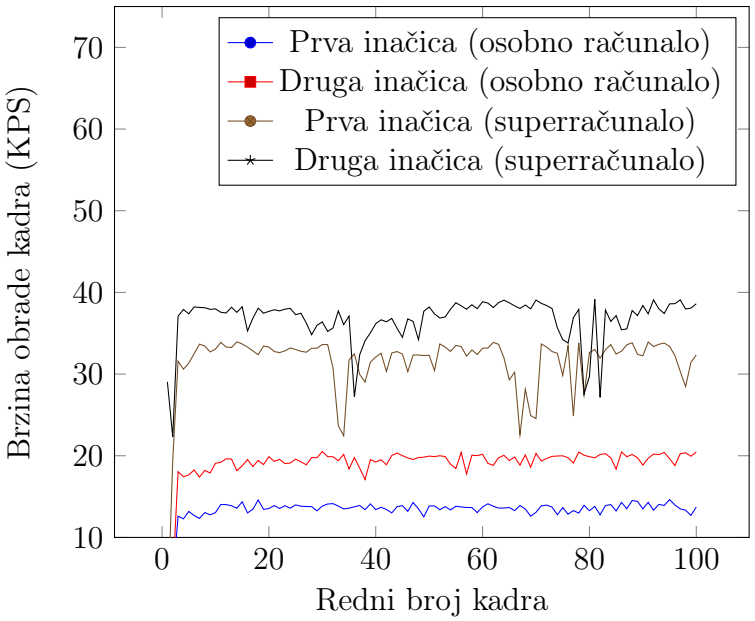


(a) Usporedba brzine cjelokupne obrade kadrova različitih inačica aplikacije na superračunalu i osobnom računalu.

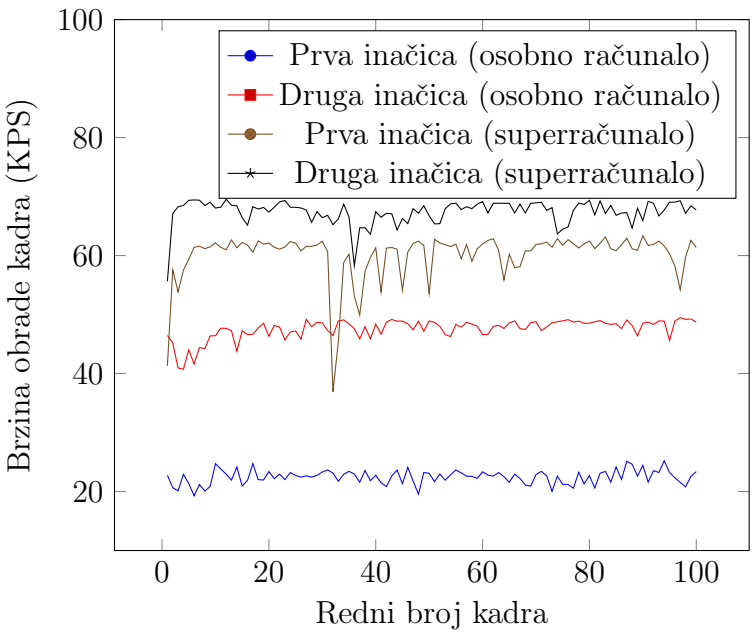


(b) Usporedba brzine obrade (uključen samo proces detekcije) kadrova različitih inačica aplikacije na superračunalu i osobnom računalu.

Slika 5.1 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba na superračunalu i osobnom računalu.



(a) Usporedba brzine cjelokupne obrade kadrova različitih inačica aplikacije na superračunalu i osobnom računalu.



(b) Usporedba brzine obrade (uključen samo proces detekcije) kadrova različitih inačica aplikacije na superračunalu i osobnom računalu.

Slika 5.2 Brzina obrade kadrova videozapisa od strane aplikacija za detekciju osoba na superračunalu i osobnom računalu (smanjena veličina kadrova tijekom procesa pripreme).

Poglavlje 6

Zaključak

Superračunala i tehnike računarstva visokih performansi rješavaju čitav niz problema koji nisu rješivi pomoću osobnih računala. Osim na njih, ovaj se rad istovremeno oslanja i na korištenje tehnika i pristupa računalnog vida kako bi rješavanje zadanog zadatka bilo moguće. Upravo su tehnike računarstva visokih performansi i tehnike računalnog vida u velikoj mjeri zaslužni za mnoga nova znanstvena otkrića i svakodnevicu kakvom ju danas poznamo.

Kroz ovaj rad su tehnike računarstva visokih performansi i tehnike računalnog vida prikazane putem aplikacije za analizu videozapisa koja podrazumijeva detekciju i interpretaciju položaja osoba koje se na njima nalaze. Razvijene inačice rješenja se zasebno razlikuju po svojim mogućnostima, ali i načinu korištenja dostupnih računalnih resursa. Pojedine su inačice zasebno i usporedno vrednovane te opisane kroz nekoliko različitih pokazatelja kvalitete. Različite inačice rješenja su pokrenute i na superračunalu BURA Sveučilišta u Rijeci te su rezultati uspoređeni s onima prikupljenima na osobnom računalu.

Završni rad pruža uvid u važnost maksimalnog iskorištenja računalnih resursa i na prikladan način kvantificira prednosti koje takav pristup donosi.

Bibliografija

[1] TOP500. Home - | TOP500. , s Interneta, https://www.top500.org/

[2] T. Sterling, M. Anderson, and M. Brodowicz, "Chapter 1 - introduction," in *High Performance Computing*, T. Sterling, M. Anderson, and M. Brodowicz, Eds. Morgan Kaufmann, 2018, pp. 1–42. , s Interneta, https://www.sciencedirect.com/science/article/pii/B9780124201583000010

[3] V. Gajjar, Y. Khandhediya, and A. Gurnani, "Human detection and tracking for video surveillance: A cognitive science approach," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 2017.

[4] OpenCV. About - OpenCV. , s Interneta, https://opencv.org/about/

[5] G. B. Adrian Kaehler, *Learning OpenCV 3*. O'Reilly UK Ltd., 2017. , s Interneta, https://www.ebook.de/de/product/24777502/adrian_kaehler_gary_bradski_learning_opencv_3.html

[6] NVIDIA. CUDA zone. , s Interneta, https://developer.nvidia.com/cuda-zone

[7] N. Zlatanov, "Cuda and gpu acceleration of image processing."

[8] E. K. Jason Sanders, *Cuda by Example: An Introduction to General-Purpose Gpu Programming*. ADDISON WESLEY PUB CO INC, 2010. , s Interneta, https://www.ebook.de/de/product/10554805/jason_sanders_edward_kandrot_cuda_by_example_an_introduction_to_general_purpose_gpu_programming.html

[9] A. Afzal, Z. Ansari, and M. K. Ramis, "Parallel performance analysis of coupled heat and fluid flow in parallel plate channel using CUDA," *Computational and Applied Mathematics*, vol. 39, no. 3, 2020.

[10] T. Barbu, "Pedestrian detection and tracking using temporal differencing and HOG features," *Computers & Electrical Engineering*, vol. 40, no. 4, pp. 1072–1079, 2014.

[11] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Fourth International Conference on Image and Graphics (ICIG 2007)*. IEEE, 2007.

[12] N. Ogale, "A survey of techniques for human detection," 2008.

[13] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, 2005.

[15] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis, "Human detection using partial least squares analysis," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009.

[16] S. Ghidary, Y. Nakata, T. Takamori, and M. Hattori, "Human detection and localization at indoor environment by home robot," in *SMC 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions' (Cat. No.00CH37166)*. IEEE, 2001.

[17] M. S. Nixon and A. S. Aguado, "High-level feature extraction: fixed shape matching," in *Feature Extraction and Image Processing for Computer Vision*. Elsevier, 2020, pp. 223–290.

[18] X. Hu, Y. Tang, and Z. Zhang, "Video object matching based on SIFT algorithm," in *2008 International Conference on Neural Networks and Signal Processing*. IEEE, 2008.

[19] S. Menet, P. Saint-Marc, and G. Medioni, "Active contour models: overview, implementation and applications," in *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*. IEEE, 1990.

[20] B. Wang, "Research on pedestrian detection algorithm based on image," *Journal of Physics: Conference Series*, vol. 1345, p. 062023, 2019.

[21] R. Girshick, "Fast r-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37.

[24] S. Mallick. Histogram of Oriented Gradients explained using OpenCV. , s Interneta, https://learnopencv.com/histogram-of-oriented-gradients/

[25] OpenCV. opencv/hog.cpp at master opencv/opencv. , s Interneta, https://github.com/opencv/opencv/blob/master/modules/objdetect/src/hog.cpp

[26] ——. cv::hogdescriptor struct reference. , s Interneta, https://docs.opencv.org/4.5.2/d5/d33/structcv_1_1HOGDescriptor.html

[27] A. Rosebrock. (2015, Nov) Hog detectmultiscale parameters explained. , s Interneta, https://www.pyimagesearch.com/2015/11/16/hog-detectmultiscale-parameters-explained/

[28] Z. Segal. (2021, Jan) Average frame rate video surveillance statistics 2021. , s Interneta, https://ipvm.com/reports/average-frame-rate-video-surveillance-2021

[29] V. Prisacariu and I. Reid, "fasthog - a real-time gpu implementation of hog," Department of Engineering Science, Oxford University, Tech. Rep. 2310/09, 2009.

[30] S. Ivić, J. Škifić, S. Družeta, B. Crnković, M. Tuhtan, L. Grbčić, I. Lučin, and M. Čavrak, *Računarsko inženjerstvo uz programski jezik Python*. Zavod za mehaniku fluida i računarsko inženjerstvo, Tehnički fakultet Sveučilišta u Rijeci, 2019.

[31] M. Jette and M. Grondona, "Slurm: Simple linux utility for resource management," *ClusterWorld Conference and Expo*, 2003.

Sažetak

Važnost superračunala, računarstva visokih performansi i račnunalnog vida u svakodnevici postaje sve veća. Završni rad se bavi temom superračunala i tehnika računarstva visokih performansi uz primjenu tehnika računalnog vida na primjeru stvaranja aplikacije za analizu videozapisa koja podrazumijeva detekciju i interpretaciju položaja osoba koje se na njima nalaze. Osim toga, dvije stvorene inačice aplikacije za detekciju osoba se vrednuju i uspoređuju kroz razne parametre kvalitete programskog rješenja. Na samom kraju, govori se o razlikama pokretanja i izvođenja aplikacije za detekciju osoba na superračunalu BURA i osobnom računalu.

Ključne riječi **— superračunalo, računarstvo visokih performansi, računalni vid**

Abstract

Importance of supercomputers, high-performance computing and computer vision in everyday life is constantly growing. This thesis covers the topic of supercomputers and high-performance computing techniques through the application of computer vision techniques on the example of creating an application for video analysis that incorporates detection and interpretation of people's positions. Furthermore, the two versions of the application are evaluated and compared considering various software quality parameters. Lastly, the differences of running and executing the application on BURA supercomputer as opposed to personal computers are described.

Keywords **— supercomputer, high-performance computing, computer vision**

Dodatak A

Programski kôd klase Args druge inačice rješenja

Prikaz programskog kôda A.1 Programski kôd klase `Args` druge inačice rješenja

```
1  class Args
2          // Klasa u koju spremamo sve parametre koje je
3          // korisnik unio prilikom pokretanja programa
4  {
5  public:
6          Args();
7          static Args read(int argc, char** argv);
8
9          // Dimenzije kadra
10         int width;
11         int height;
12
13         // Ako se u parametrima naznaci da se cita iz
14         // mape ili datoteke videozapisa, onda se cita s
15         // lokacije src, a inace se cita s kamere s
16         // ID-om camera_id
17         string src;
18         bool src_is_folder;
19         bool src_is_video;
20         bool src_is_camera;
21         int camera_id;
22
23         // Ako imamo vlastiti SVM se moze ucitati sa svm_src
24         bool svm_load;
25         string svm_src;
26
27         // Ako zelimo spremati video detekcije, write_video je
```

```
28          // true i sprema se na lokaciju dst_video s FPS-om dst_video_fps
29          bool write_video;
30          string dst_video;
31          double dst_video_fps;
32
33          // Ako zelimo crno-bijelu sliku, make_gray je true
34          bool make_gray;
35
36          // Ako zelimo promijeniti velicinu ulaznog kadra (w x h)
37          // za brze rezultate, resize_src je true
38          bool resize_src;
39          int resize_width;
40          int resize_height;
41
42          // Parametri za HOG
43          double scale;
44          int nlevels;
45          int gr_threshold;
46
47          bool hit_threshold_auto;
48          double hit_threshold;
49
50          int win_width;
51          int win_stride_width;
52          int win_stride_height;
53          int block_width;
54          int block_stride_width;
55          int block_stride_height;
56          int cell_width;
57          int nbins;
58
59          // Gamma korekcija
60          bool gamma_corr;
61
62          // Ako zelimo spremati koordinate detekcija, save_to_file je true
63          // i sprema se u file_name.txt
64          bool save_coord;
65          string file_name_coord;
66
67          // Ako zelimo ucitati koeficijente regresije iz file_name_reg
68          bool load_reg;
69          string file_name_reg;
70          double a3, a2, a1, a0; // Regresijski koeficijenti (reg. treceg reda)
71          double tolerance; // Tolerancija u postocima
72  };
```