Razvoj skupa podataka sa stvarnim scenama iz prometa za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila

Kelemen, Matija

Master's thesis / Diplomski rad

2022

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:761390

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

RAZVOJ SKUPA PODATAKA SA STVARNIM SCENAMA IZ PROMETA ZA RAZVOJ ALGORITAMA ZA AUTONOMNU VOŽNJU ZASNOVANIH NA PROCJENI KUTA ZAKRETA UPRAVLJAČA VOZILA

Diplomski rad

Matija Kelemen

Osijek, 2022.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 16.09.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Matija Kelemen
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1131R, 15.10.2020.
OIB studenta:	94174040313
Mentor:	Izv. prof. dr. sc. Mario Vranješ
Sumentor:	,
Sumentor iz tvrtke:	David Mijić
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Ratko Grbić
Član Povjerenstva 1:	Izv. prof. dr. sc. Mario Vranješ
Član Povjerenstva 2:	Prof. dr. sc. Marijan Herceg
Naslov diplomskog rada:	Razvoj skupa podataka sa stvarnim scenama iz prometa za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U ovom radu, potrebno je kreirati skup podataka za treniranje i testiranje algoritama autonomne vožnje zasnovanih na procjeni kuta zakreta upravljača vozila (engl. steering angle prediction). Izgrađeni podatkovni skup treba se sastojati od okvira (engl. frames) izvučenih iz video sekvenci koje sadrže stvarne scene iz prometa, prikupljenih pomoću kamere montirane na vjetrobransko staklo vozila i pripadnog kuta zakreta upravljača vozila koji se mjeri odgovarajućim senzorom, npr. kombinacija akcelerometra i žiroskopa ili uređaj za mjerenje inercije (engl. inertial measurement unit, IMU). Nakon prikupljanja podataka, potrebno je ispitati korištenje tog skupa podataka na jednom od postojećih rješenja za procjenu kuta zaokreta upravljača vozila. S
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	16.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 16.09.2022.

Ime i prezime studenta:	Matija Kelemen
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1131R, 15.10.2020.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj skupa podataka sa stvarnim scenama iz prometa za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Mario Vranješ

i sumentora ,

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

Potpis studenta:

Sadržaj

1. UVOD

Umjetna inteligencija postala je dio naše svakodnevice. Kao što su veliki industrijski pogoni gotovo u potpunosti automatizirani, tako je i velik broj svakodnevnih zadataka praćen nekom vrstom umjetne inteligencije. Pokazalo se da dobro implementiran inteligentni agent može pomoći kod raznih operacija koje čovjek obavlja, pa tako i kod sigurnosno-kritičnih kao što je autonomna vožnja. Operacije poput vožnje vozila po prometnicama opasne su za čovjeka jer vozač nije otporan na umor, zbog čega može doći do gubitka koncentracije, što se pokazalo kao jedan od najčešćih uzroka prometnih nesreća. Razvojem automobilske industrije pokušava se smanjiti broj nesreća ugrađujući sustave koji će primati i obrađivati informacije iz okoline i odlučiti treba li reagirati na neko stanje iz okoline. Sustavi kao što su oni za detekciju linija voznih traka i praćenje voznih traka vrlo često se ugrađuju u današnje moderne automobile. Od 2024. godine svaki automobil morat će imati ugrađen adaptivni tempomat (engl. *Adaptive Cruise Control*), koji će automatski kočiti ako primijeti da se približava predmetu na prometnici[1].

Autonomna vožnja zasnovana na procjeni kuta zakreta upravljača vozila zamišljena je da se odvija na način da vozilo prima samo sliku s prednje kamere vozila i da obradom te slike donese ispravan zaključak o kutu zakreta upravljača vozila. Kao i svi algoritmi zasnovani na strojnom učenju, i ovaj algoritam autonomne vožnje potrebno je naučiti da ispravno reagira na situaciju u okolini, a to se postiže procesom treninga (učenja) za koji je potreban odgovarajući skup podataka. Skup podataka treba sadržavati ulazne i izlazne podatke koji služe algoritmu da nauči reagirati u pojedinim situacijama. Ulazni podaci su slike okoline ispred vozila snimljene kamerom najčešće postavljenom na vjetrobransko staklo s prednje strane, a izlazni podaci su kutovi zakreta upravljača vozila. Za prikupljanja takvog skupa podataka potrebno je montirati kameru kako bi se prikupile slike stanja ispred vozila i prikupljati podatke senzora kuta zakreta upravljača preko dijagnostičkog sučelja.

Općeniti problem prilikom razvoja rješenja zasnovanih na umjetnoj inteligenciji je nedostatak kvalitetnih skupova podataka koji su potrebni za treniranje modela. Podaci su najčešće zaštićeni od strane kompanija koje razvijaju vlastita rješenja te stoga ne omogućavaju pristup tim podacima. Sve više rješenja zasnovanih na umjetnoj inteligenciji koriste neuronske mreže kao alat za rješavanje danog problema. Kako bi osigurali da neuronska mreža što uspješnije obavlja zadani zadatak, potrebno je osigurati joj kvalitetan skup podataka za treniranje i testiranje obavljanja istog zadatka. Na primjer, za slučaj

autonomne vožnje to bi značilo prikupiti odgovarajući skup podataka u različitim uvjetima vožnje (urbano, suburbano, ruralno okruženje), različitim vremenskim uvjetima (kiša, sunce, magla, itd.), različitim dobima dana (dan, noć), itd. S obzirom na to da je velika većina skupova podataka u području procjene kuta zakreta upravljača vozila umjetno generirana (u različitim simulatorima), cilj ovog rada je osmisliti način prikupljanja podataka iz stvarnih prometnih scenarija i kreirati skup podataka koji sadrži podatke iz stvarnog svijeta.

U ovom radu predložen je pristupačan način za prikupljanja slika stanja ispred vozila i podataka o kutu zakreta upravljača vozila. Osmišljen je sustav koji se sastoji od mikroračunala koje se povezuje na dijagnostičko sučelje vozila i kamere koja se montira na prednje vjetrobransko staklo. Uz sustav, implementirano je programsko rješenje koje sprema postavke za ujednačeno snimanje, čime se omogućuje ponavljanje istih uvjeta snimanja u različitim vremenima. Dobivene podatke potrebno je vremenski sinkronizirati na način da je svaka slika vremenski uparena s kutom zakreta upravljača vozila. Napravljena je i usporedba novo-nastalog skupa podataka s poznatim postojećim skupovima podataka koji su široko korišteni za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila.

U drugom poglavlju rada predstavljena su trenutna rješenja za prikupljanje podataka za razvoj algoritama autonomne vožnje zasnovanih na procjeni kuta zakreta upravljača. Također, objašnjen je algoritam koji će se koristiti za usporedbu dvaju različitih skupova podataka (novo-nastali i postojeći). U trećem poglavlju objašnjen je način prikupljanja podataka o okolini ispred vozila koristeći Raspberry Pi kameru i podataka iz vozila koristeći OBD2 dijagnostičko sučelje. U četvrtom poglavlju ispitana je mogućnost korištenja stvorenog skupa podataka na jednom od postojećih rješenja za procjenu kuta zakreta upravljača vozila. Također, dana je usporedba performansi predviđanja modela treniranog na novo-nastalom skupu podataka i modela treniranog na postojećem skupu podataka U petom poglavlju nalazi se zaključak rada.

2. PROBLEM AUTONOMNE VOŽNJE ZASNOVANE NA KUTU ZAKRETA UPRAVLJAČA

Razvoj autonomne vožnje se pokazao kao nužan korak u povećanju sigurnosti svih sudionika u prometu jer mu je cilj smanjiti mogućnost ljudske pogreške i eliminirati pogreške koje nastaju u vožnji zbog umora, konzumiranja opijata i slično. Potpuna autonomna vožnja podrazumijeva kretanje automobila bez ikakve ljudske intervencije. To je moguće samo ako vozilo, točnije inteligentni agent, ima sposobnost prepoznavanja i klasifikacije okoline kao što to radi ljudski um. To uključuje i vozačeve interakcije s automobilom kao što su ubrzavanje, usporavanje, signaliziranje i kontrola upravljača vozila. Jedan od takvih algoritama koji se može koristiti je algoritam za autonomnu vožnju zasnovan na procjeni kuta zakreta upravljača vozila. To znači da za svaku sliku koju algoritam dobiva putem senzora, u ovom slučaju prednja kamera, mora predvidjeti koliki kut zakreta upravljača je potreban kako bi vozilo slijedilo smjer prometnice.

2.1. Problem nedostatka skupova podataka za razvoj algoritama autonomne vožnje zasnovanih na kutu zakreta upravljača

Najveći problem prilikom razvoja algoritma autonomne vožnje zasnovanog na procjeni kuta zakreta upravljača je nedostatak podataka za učenje algoritma. Kako bi algoritam mogao naučiti predvidjeti točan kut zakreta upravljača, potrebno je pripremiti veliki skup podataka na kojem će se taj algoritam učiti. Skup podataka uključuje velik broj slika i pripadnih kutova zakreta upravljača koji su bili aktualni u trenutku snimanja svake pojedine slike. Ovdje se javlja prvi problem jer nisu sve prometnice jednake. Unutar manje regije mogu se pojaviti različite prometnice koje se razlikuju u broju traka, postojanosti vanjske linije, stanju podloge i slično. Također, zbog razlike u automobilima, potrebno je utvrditi način kako iz pojedinog automobila prikupiti podatke o kutu zakreta upravljača.

Sljedeći problem leži u vremenskim uvjetima i razdoblju vožnje. Da bi skup podataka bio potpun, potrebno je prikupiti podatke iz različitih doba dana i pokriti što više vremenskih uvjeta. Radi pokrivanja što većeg broja uvjeta potrebno je osigurati ponavljanje snimanja. To uključuje spremanje postavki za ujednačeno snimanje i fiksnu pozicija kamere. Pogrešno montiranje kamere može dovesti do različitih rezultata prilikom snimanja na prethodnim relacijama pod različitim uvjetima. To je potrebno spriječiti kako bi algoritam naučio da

uvjeti ne utječu na potreban kut zakreta automobila.

Slike kamere i podatke iz automobila potrebno je sinkronizirati, odnosno potrebno je osigurati da trenutak stvaranja slike odgovara trenutku očitanja podataka o kutu zakreta upravljača. Nesinkronizirani podaci mogu navesti algoritam da krivo nauči kako obavljati zadatak, pa ga stoga ni ne može obaviti onako kako bi trebao. Nesinkroniziranost od 0.1 sekunde pri brzini od 60 km/h predstavlja pogrešan pomak automobila od 1.6 metara. Pri manjim brzinama to ne predstavlja toliki problem, no kako se u današnje vrijeme autonomna vožnja najviše primjenjuje na brzim cestama, pogreška može dovesti do teških posljedica.

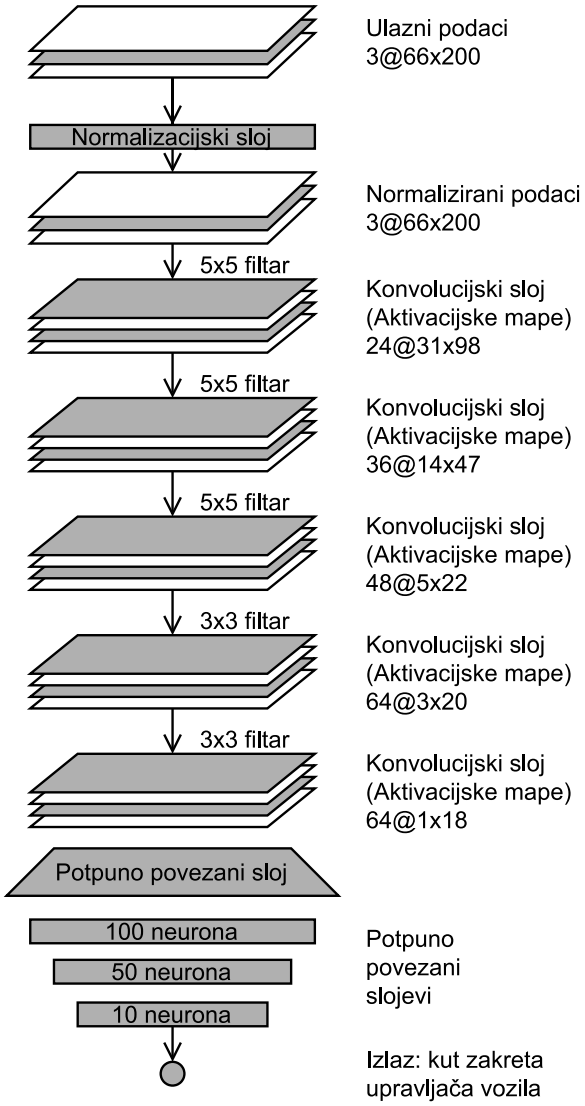
2.2. Pregled postojećih algoritama za razvoj autonomne vožnje zasnovanih na procjeni kuta zakreta upravljača

Sve više rješenja zasnovanih na umjetnoj inteligenciji koriste neuronske mreže kao alat za rješavanje danog problema. Ideja neuronskih mreža je da postoji velik broj slika za koje je poznat kut zakreta upravljača, pa se iz tih primjera treba naučiti procijeniti kut zakreta upravljača za nove slike koje će se u budućnosti dobivati. Procijenjen izlaz se zatim usporedi sa stvarnom (poznatom) vrijednošću kako bi se evaluirala mreža. Neuronska mreža općenito se sastoji od ulaznog sloja, izlaznog sloja i skrivenih slojeva. Slojevi se sastoje od neurona kojima su pridružene vrijednosti koje se nazivaju težinama neurona. Učenje neuronske mreže predstavlja iterativno podešavanje težina neurona, kako bi predviđena izlazna vrijednost neuronske mreža bila što bliža stvarnoj (izmjerenoj) vrijednosti.

Jedan od prvih algoritama autonomne vožnje zasnovanih na procjeni kuta zakreta upravljača koji je koristio neuronsku mrežu je ALVINN (engl. *Autonomous Land Vehicle In a Neural Network*)[2]. Neuronska mreža se sastojala od tri potpuno povezana sloja, čime su autori potvrdili mogućnost korištenja neuronske mreže za upravljanje vozilom. Sljedeći značajan rad koji je koristio neuronsku mrežu je DAVE (engl. *DARPA Autonomous Vehicle*)[3], gdje je korišteno vozilo navođeno daljinskim upravljačem kako bi se prikupili sati ljudske vožnje na čemu je algoritam zatim učio. U zadnjih nekoliko godina pojavljuje se sve više rješenja koja koriste konvolucijske neuronske mreže. Razlog tome je da se pojavljuju sve veći skupovi podataka i implementacija konvolucijskih slojeva na paralelnim arhitekturama, za što se koriste grafičke procesne jedinice.

Jedan od najpoznatijih algoritama za autonomnu vožnju zasnovanu na procjeni kuta zakreta upravljača vozila je PilotNet[4], koji za rješavanje problema koristi konvolucijsku

neuronsku mrežu čija je arhitektura prikazana na slici 2.1. Kao ulaz, mreža prima sliku koja je zapisana kao 3D matrica dimenzija $200 \times 66 \times 3$. Prve dvije dimenzije predstavljaju dimenziju ulazne slike (širinu i visinu), a treća dimenzija predstavlja broj kanala boje u RGB zapisu. Prvi sloj je sloj normiranja koji se ne mijenja u procesu učenja. Normizacija se provodi kako bi se izbjegli ekstremi koji se mogu pojaviti na slikama, čime se postiže slična distribucija vrijednosti elemenata slike. U nastavku se nalazi pet konvolucijskih slojeva. Prva tri konvolucijska sloja koriste filtre (engl. *kernel*) dimenzija 5×5 i korak (engl. *stride*) dimenzija 2×2. Korak je vrijednost za koliko se pomiče filtar prilikom provođenja operacije konvolucije. Četvrti i peti konvolucijski sloj koriste filtar dimenzija 3×3 i korak 1×1. Zatim slijede tri potpuno povezana sloja koji smanjuju dimenzionalnost na realnu vrijednost koja predstavlja $1/r$, gdje je r radijus okretanja vozila.



Slika 2.1: Arhitektura neuronske mreže PilotNet modela[4]

Rezultat treniranog PilotNet modela predstavljen je pomoću mjere autonomije (engl. *autonomy*) koja je definirana kao:

$$\text{autonomija} = (1 - \frac{(\text{broj intervencija}) \times 6 \text{ sekundi}}{\text{proteklo vrijeme [s]}}) \times 100 \tag{2-1}$$

Autonomija se računa kao broj ljudskih intervencija prilikom testiranja modela neuronske mreže. Nužna je intervencija ako se sredina simuliranog vozila nalazi na udaljenosti većoj od jednog metra od sredine prometne trake. Za prosječnu ljudsku interakciju potrebno je šest sekundi. Tijekom testiranja u simulatoru i stvarnim cestama postignuta je mjera autonomije od 98%[4].

2.3. Pregled postojećih skupova podataka za razvoj algoritama autonomne vožnje zasnovanih na procjeni kuta zakreta upravljača

U ovom poglavlju nalazi se pregled postojećih skupova podataka koji se mogu koristiti za razvoj algoritama autonomne vožnje zasnovanih na kutu zakreta upravljača vozila.

2.3.1. Skup podataka Nvidia PilotNet

Tvrtka Nvidia razvila je model PilotNet, a za njegov razvoji koristili su vlastiti skup podataka. Skup podataka je zaštićen te nije omogućen pristup tim podacima. Prikupljanje skupa podataka objašnjeno je u radu[4]. Za prikupljanje skupa podataka korišteno je Nvidia drive PX 2 AI računalo opremljeno s dva Tegra X1 čipa. Za snimanje slika korištene su tri kamere koje su usmjerene prema prostoru ispred vozila. Jedna kamera montirana je na sredini vozila, a druge dvije na lijevoj i desnoj strani vozila, čime je osiguran veći kut pregleda stanja okoline ispred vozila. Na slici 2.2 prikazan je položaj kamera. Dohvaćanje podataka o kutu zakreta upravljača napravljeno je na način da se računalo poveže na CAN (engl. *controller area network*) sabirnicu automobila preko dijagnostičkog sučelja. Slike se snimaju frekvencijom 10 sličica u sekundi (engl. *frames per second, FPS*), a svi podaci se spremaju na SSD (engl. *solid state disk*). Ukupan broj slika i uvjeti snimanja nisu navedeni.

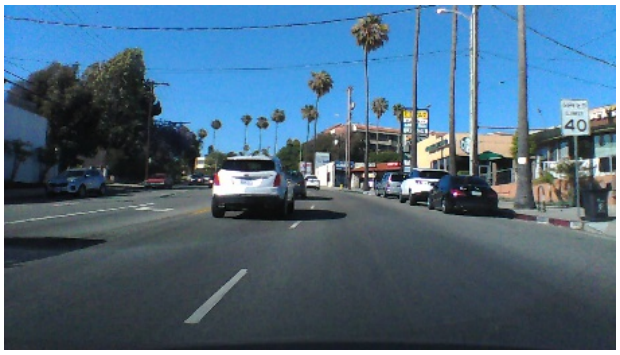
Slika 2.2: Prikaz pričvršćenih kamera korištenih u radu[5]

2.3.2. Skup podataka SullyChen

Jedan od najpopularnijih skupova podataka za učenje algoritama za razvoj autonomne vožnje zasnovanih na procjeni kuta zakreta upravljača je SullyChen[6] skup podataka, podijeljen u dva dijela. Prvi dio sadrži 45405 slika koje su snimljene vožnjom kroz Rancho Palos Verdes, Kalifornia i San Pedro, Kalifornia, 2017. godine. Drugi dio se sastoji od 63824 snimljenih slika u istoj regiji, a snimljen je 2018. godine. Svaki od dvaju skupova podataka uključuje slike u JPEG formatu dimenzije 455×256 elemenata slike i .csv datoteku kojom se povezuje ime slike s pripadnim kutom zakreta upravljača. Pozitivni kutovi predstavljaju zakret upravljača u desnu, a negativni kutovi u lijevu stranu. Na slici 2.3 dani su primjeri slika iz SullyChen skupa podataka.

(a)

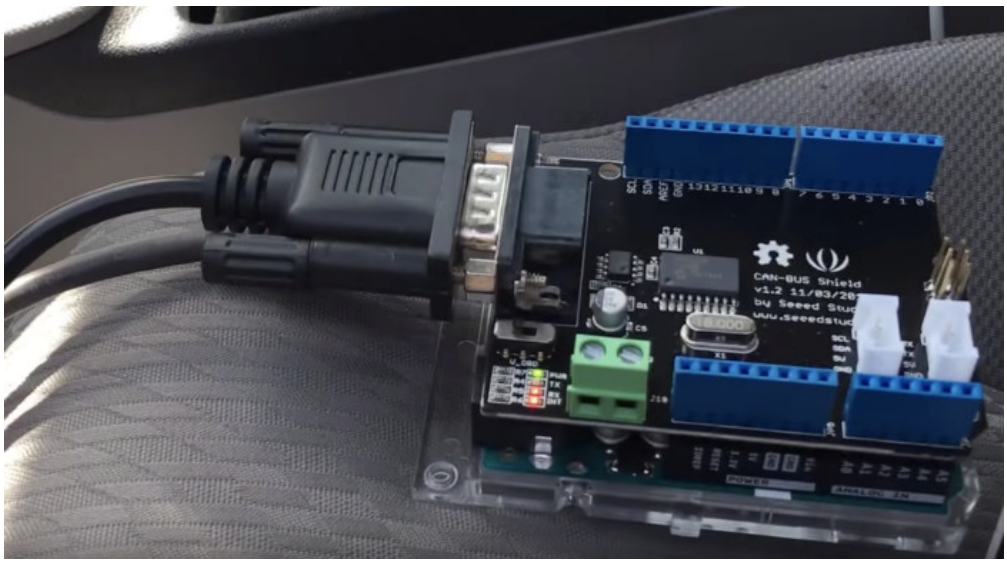
(b)




(c) (d)

Slika 2.3: Primjeri slika iz SullyChen skupa podataka (a) ruralno područje, kut zakreta upravljača 1.21°, (b) ruralno područje, kut zakreta upravljača 29.85°, (c) urbano područje, kut zakreta upravljača −2.82°, (d) urbano područje, kut zakreta upravljača −338.82°

Za prikupljanje SullyChen[6] skupa podataka korišten je Arduino s CAN pločicom za proširenje zbog povezivanje na CAN sabirnicu automobila. Kamera je montirana na prednje vjetrobransko staklo i povezana je USB sučeljem na prijenosno računalo[7]. Takav pristup omogućio je smanjenje troškova s vrlo dobrim rezultatima. Na slici 2.4 prikazano je Arduino sklopovlje za prikupljanje SullyChen skupa podataka.



Slika 2.4: Arduino UNO s CAN pločicom za proširenje korišten za prikupljanje SullyChen skupa podataka[6]

Prednost korištenja SullyChen skupa podataka je da su slike objavljene u JPEG zapisu, što omogućuje lakše pregledavanje slika nakon preuzimanja. Nedostatak je mali broj slika i nepostojanost analize skupa podataka s podacima o udjelu različitih cesta i vremenskih uvjeta.

2.3.3. Skup podataka comma2k19

Comma AI je tvrtka koja se bavi razvojem sustava koji pomažu vozaču u vožnji. Objavili su skup podataka comma2k19[8] koji se sastoji od slika dobivenih nakon tri sata vožnje autocestom u Kaliforniji. Podijeljen je u 2019 segmenata od po jednu minutu na autocesti između gradova San Jose i San Francisco. Sveukupno sadrži 931821 sliku s uparenim podacima. Podaci osim slika i kuta zakreta upravljača sadrže čitav zapis CAN poruka iz automobila, neobrađenih GNSS (engl. *Global navigation satellite system*) mjerenja, temperaturu zraka u okolini i vrijednosti akcelerometra. Podaci su zapakirani u H5 datoteku i moguće je preuzeti alate za jednostavno raspakiravanje skupa podataka s GitHub repozitorija *comma.ai* organizacije[8]. Na slici 2.5 prikazani su primjeri slika iz comma2k19 skupa podataka.

(a)

(b)

(c)

(d)

Slika 2.5: Primjeri slika iz comma2k19 skupa podataka (a) ruralno područje, kišovito vrijeme, noć, kut zakreta upravljača 0.9°, (b) ruralno područje, sunčano vrijeme, dan, kut zakreta upravljača 9.4°, (c) urbano područje, oblačno vrijeme, dan, kut zakreta upravljača 33.0°, (d) urbano područje, noć, kut zakreta upravljača 3.3°

Comma AI razvio je uređaj *Comma Three* koji omogućuje prikupljanje podataka i samostalno upravljanje automobilom, ako automobil ima mogućnost upravljanja preko dijagnostičkog sučelja. *Comma Three* montira se pomoću 3D printanog nosača na prednje vjetrobransko staklo. Uređaj u sebi sadrži Qualcomm Snapdragon 845 ARM procesor koji upravlja snimanjem. Povezivanje na dijagnostičko sučelje odvija se koristeći njihov pribor koji omogućava jednostavno korištenje. Uređaj se može povezati na internet koristeći LTE ili WiFi tehnologiju i spremati snimke na njihovu uslugu u oblaku. Unutar uređaja također se nalazi SSD disk koji omogućava snimanje u uvjetima bez povezivanja ne internet.

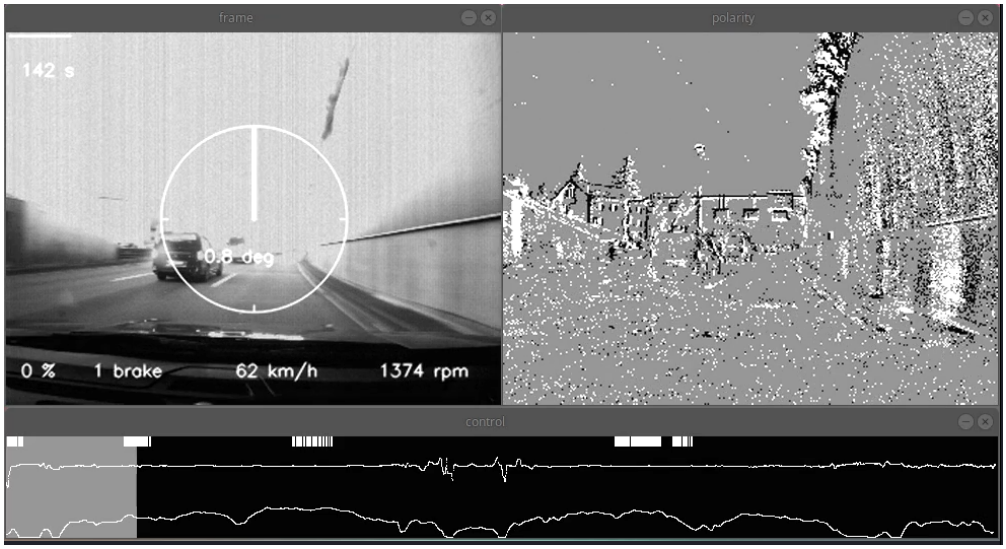
Prednost skupa podataka su velika količina podataka, budući da osim slika kamere i podataka o kutu zakreta upravljača uključuje i druge podatke koji su prethodno navedeni. Velika količina podataka uključuje različite vremenske uvjete i doba dana, što je ključno za takav skup podataka. Moguće je ponoviti njihova snimanja na vlastitom vozilu kupnjom uređaja *Comma Three*. Nedostatak je što su sva snimanja provedena na istim relacijama i što nedostaju neki detalji o skupu podataka, kao što je omjer broja slika snimljenih u različitim dobima dana, na različitim tipovima prometnicama i pri različitim vremenskim uvjetima. Na slici 2.6 nalazi se primjer montiranog *Comma Three* uređaja u automobilu.

Slika 2.6: Comma Three uređaj montiran u automobilu[8]

2.3.4. Skupovi podataka DDD17 i DDD20

Za razliku od većine skupova podataka za čije su se stvaranje koristile klasične kamere, DDD17[9] i DDD20[10] koriste kamere događaja (engl. *event camera*). Kamere događaja ne funkcioniraju kao standardne kamere, već se svaki element slike gleda samostalno.

Izlazna slika kamere događaja sastoji se od dva dijela. Prvi dio je klasična monokromatska (engl. *grayscale*) slika, a drugi dio predstavlja promjenu osvjetljenja. Promjena osvjetljenja prikazana je vrijednošću elementa slika. Ako se osvjetljenje nije promijenilo na nekoj poziciji, element slike ostaje crn. Nasuprot tome, ako se pojavila velika promjena osvjetljenja na poziciji, element slike postane bijel. Ako se radi o maloj promjeni osvjetljenja na poziciji on prima sivu boju, ovisno o razlici osvjetljenja na toj poziciji. Na slici 2.7 prikazan je isječak videa koji prikazuje slike snimljene kamerom događaja. Lijeva slika prikazuje standardnu monokromatsku sliku i podatke koji su prikupljeni iz dijagnostičkog sučelja automobila. U sredini slike prikazan je kut zakreta upravljača izražen u stupnjevima. Na dnu slike, s lijeva na desno, prikazani su položaj papučice gasa, stanje papučice kočnice, brzina vozila i broj okretaja motora vozila. Desna slika pokazuje događaje promjene elementa slike. Na dnu slike nalaze se tri grafički prikazana podatka zabilježeni kroz vrijeme snimanja. Od gore prema dolje nalazi se status prednjih svijetla, kut zakreta upravljača i brzina vozila.



Slika 2.7: Primjer jedne slike iz DDD20 skupa podataka[10]

Uz skup slika, autori skupa podataka razvili su korisne alate za pregled podataka gdje slike nisu u standardnom zapisu. DDD17 sadrži preko 12 sati snimljenih video materijala po autocestama i cestama kroz grad u raznim uvjetima, kao što su doba dana i vremenski uvjeti. Ukupan broj slika nije naveden u skupu podataka, nego je navedeno trajanje snimki i frekvencija snimanja slika, iz čega bi se dalo nazrijeti da skup sadrži nešto malo više od milijun slika. Uz slike i kut zakreta upravljača, skup podataka sadrži brzinu vozila, GPS lokaciju te pritisak papučica kočnice i gasa. DDD20 proširuje prethodni skup DDD17 i sadrži 51 sat vožnje i preko 4000 km vožnje. Na sličan se način može zaključiti da sadrži blizu tri

milijuna slika. Podaci o svim vožnjama DDD17 i DDD20 skupa prikazani su u Google Sheets tablici[11].

Prednost korištenja DDD20 skupa podataka je vrlo opširna analiza slika i uparenih kutova, čime se ostvaruje direktan pregled skupa podataka bez potrebnog preuzimanja. Nedostatak je zastarjelost alata gdje je korišten *Python2.7* koji više nije podržan i što neke od korištenih biblioteka nije moguće preuzeti sa službenih *Python* repozitorija.

2.3.5. Skup podataka A2D2

Skup podataka A2D2[12] (engl. *Audi Autonomous Driving Dataset*) sastoji se od istovremeno snimljenih slika i 3D oblaka točaka, zajedno s 3D graničnim okvirima (engl. *bounding boxes*), semantičkom segmentacijom, segmentacijom instanci i podacima izdvojenim iz dijagnostičkog sučelja automobila. Korišteno je šest kamera i pet LiDAR jedinica, pružajući punih 360 stupnjeva pokrivenosti. Sveukupno se sastoji od 392556 slika rezolucije 1920×1208 elementa slike snimljenih u tri grada na jugu Njemačke, gdje svaka slika ima precizno zabilježeno vrijeme snimanja. Slike su snimljene na različitim cestama, što uključuje urbana područja, autoceste i ruralna područja. Sve slike snimljene su tijekom dana uz različite vremenske uvjete (oblačno, kišovito i sunčano). Uz slike, zapisani su podaci iz vozila kao što su ubrzanje, brzina, GPS koordinate, pritisak kočnice, kut zakreta upravljača i druge informacije iz dijagnostičkog sučelja automobila. Na slici 2.8 dani su primjeri slika iz A2D2 skupa podataka. Pozitivne vrijednosti kutova predstavljaju kut zakreta upravljača u desnu stranu, dok su negativne vrijednosti kutovi zakreta upravljača u lijevu stranu.

(a)

(b)

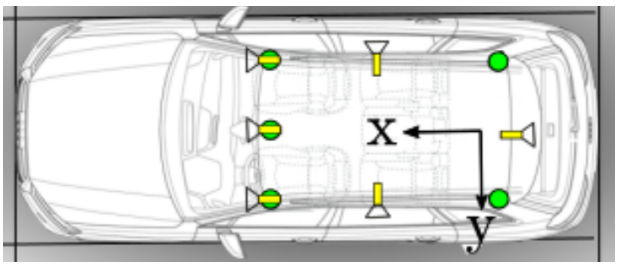
(c)

(d)

Slika 2.8: Primjeri slika prednje središnje kamere iz A2D2 skupa podataka (a) ruralno područje, oblačno vrijeme, dan, kut zakreta upravljača 4.35°, (b) urbano područje, sunčano vrijeme, dan, kut zakreta upravljača −107.4°, (c) ruralno područje, sunčano vrijeme, dan, kut zakreta upravljača −6.3°, (d) urbano područje, oblačno vrijeme, dan, kut zakreta upravljača 9.75°

Za prikupljanje skupa podataka korišten je automobil Audi Q7 e-tron s kamerama postavljenim na krovu vozila. Automobil i položaj kamera prikazani su na slici 2.9. Na slici 2.9 (b) zelenim krugovima su označene LiDAR jedinice, a žutim linijama kamere i njihova orijentacija.

(a)



(b)

Slika 2.9: (a) Vozilo s montiranim kamerama za prikupljanje slika za A2D2 skup podataka (b) položaj i orijentacija kamera na automobilu te položaj LiDAR-a[12]

Velika prednost korištenja A2D2 skupa podataka je što uključuje LiDAR podatke na kojima su označeni pojedini objekti na cesti iz čak 19 klasa. Nedostatak skupa podataka je što ne sadrži snimke po noći.

Iz ranijeg opisa pet dostupnih skupova podataka za razvoj različitih algoritama autonomne vožnje, može se vidjeti na koji način je napravljeno prikupljanje skupa podataka.

Problem korištenja navedenih skupova podataka je nedostatak analize slika i kutova zakreta upravljača, gdje bi prije odabira skupa podataka bilo vidljivo kakve uvjete skup sadrži. Upravo stoga, u ovom radu, u trećem poglavlju bit će predstavljen novo-kreirani skup podataka za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila, koji će sadržavati slike snimljene prednjom kamerom i svakoj slici vremenski uparen kut zakreta upravljača vozila. Uz sam skup podataka napravljena je detaljna analiza slika i pripadnih kutova.

3. PREDSTAVLJANJE NOVOG SKUPA PODATAKA ZA RAZVOJ ALGORITAMA ZA AUTONOMNU VOŽNJU ZASNOVANIH NA PROCJENI KUTA ZAKRETA UPRAVLJAČA VOZILA

Skup podataka za učenje algoritma mora sadržavati velik broj slika koje prikazuju stanje okoline ispred vozila te za svaku sliku vremenski uparenu vrijednost kuta zakreta upravljača vozila. Kako bi model neuronske mreže što bolje naučio procijeniti kut zakreta upravljača, potrebno ga je pripremiti za rad u različitim uvjetima. Potrebno je omogućiti višestruko snimanje vožnje istom prometnicom u različitim uvjetima, kako bi se pokrili razni uvjeti na cesti. Pod različitim uvjetima ubrajaju se vrsta ceste, doba dana i vremenski uvjeti.

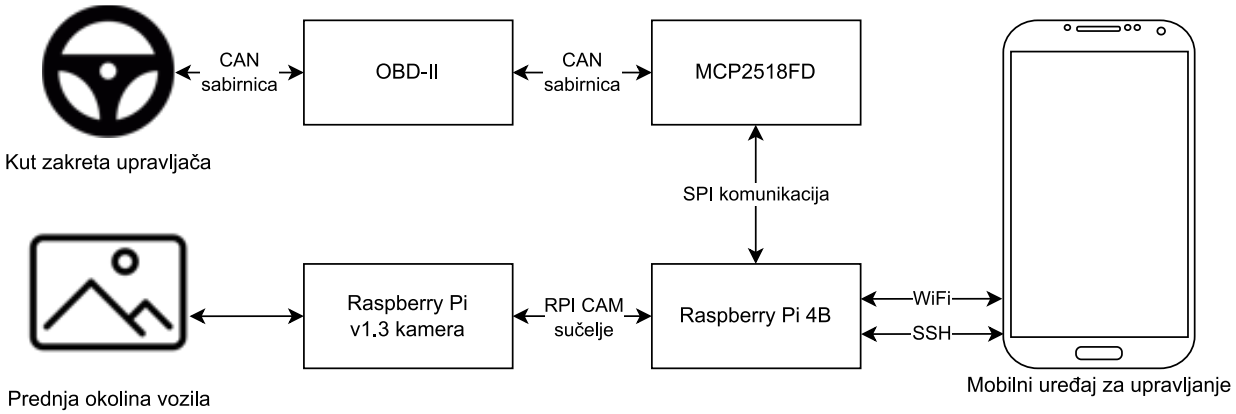
3.1. Arhitektura sustava za prikupljanje podataka

Kako bi sinkronizacija bila što preciznija, potrebno je koristiti jedan centralni uređaj koji će spremati slike kamere i dohvaćati kut zakreta upravljača vozila. Za centralni uređaj korišteno je mikro-računalo Raspberry Pi 4 model B. Model koji je bio na raspolaganju sadrži 4 GB radne memorije i SD karticu kapaciteta 64 GB. Ostale značajke mikro-računala vidljive su u specifikacijama[13].

Raspberry Pi omogućava korisniku instalaciju operacijskog sustava na SD karticu koja se zatim koristi za pohranu podatka. Instaliran je operacijski sustav Raspberry Pi OS Lite s aktualnom verzijom 11. Raspberry Pi OS je zasnovan na Debian Linux distribuciji. Korištenje operacijskog sustava omogućava izvršavanje programa u pozadini. Na taj način može se koristiti SSH (engl. *Secure Shell*) protokol koji omogućuje povezivanje na mikro-računalo s osobnog računala ili mobilnog uređaja putem lokalne mreže. Raspberry Pi nudi mogućnost povezivanja na mrežu putem bežičnih mrežnih protokola ili korištenjem Ethernet kabla.

Upravljanje mikro-računalom vrši se mobilnim uređajem koristeći *JuiceSSH*[14] aplikaciju. Kako bi se mobilni uređaj mogao povezati na Raspberry Pi, koristi se mobilna pristupna točka na koju se mikro-računalo povezuje. Tim pristupom ostvaruje se lokalna mreža te se može koristiti SSH protokol za povezivanje. Osim povezivanja s mobilnim uređajem, mikro-računalo mora se povezati na Raspberry Pi kameru i mikro-upravljač koji omogućava povezivanje s dijagnostičkim sučeljem automobila. Arhitektura sustava za

prikupljanje skupa podataka prikazana je na slici 3.1, a svaka komponenta sustava bit će objašnjena u nastavku.



Slika 3.1: Arhitektura sustava korištenog u sklopu ovog diplomskog rada za prikupljanje skupa podataka

3.2. Prikupljanje slika s prednje kamere automobila

Kako bi se napravio skup podataka sa slikama, potrebno je postaviti kameru koja će snimiti prostor ispred vozila. Prostor ispred vozila predstavlja cesta kojom se vozilo kreće. Za razvoj algoritma autonomne vožnje zasnovanog na procjeni kuta zakreta upravljača potrebno je prikupiti što više slika vožnje zavojima uz različite uvjete, kako bi model neuronske mreže koji treba kasnije samostalno upravljati vozilom naučio predvidjeti kut zakreta upravljača u raznim situacijama. Prethodno je navedeno da Raspberry Pi sprema podatke na SD karticu te je odabrana kartica sa 64 GB prostra za pohranu.

3.2.1. Raspberry Pi kamera

Za potrebe prikupljanja slika s prednje strane vozila, odabrana je Raspberry Pi kamera, verzije 1.3, prikazana na slici 3.2. Navedena je kamera odabrana zbog trenutne dostupnosti, jednostavnog povezivanja s mikro-računalom i jer pruža dovoljno visoku rezoluciju i broj slika u sekundi koje zadovoljavaju potrebe zadatka ovog rada i problema općenito. Može bilježiti slike s rezolucijom do 1080p. Za potrebe učenja neuronske mreže koriste se slike manje rezolucije te je odlučeno da će se snimati u 720p razlučivosti - 1280 elementa slike širine i 720 elementa slike visine. Većina današnjih automobila koriste upravo kamere s razlučivošću 720p ili 1080p, iako se stremi kamerama veće razlučivosti. Uz odabranu razlučivost može snimiti do 60 sličica u sekundi. Ta brzina zapisa moguća je samo u H.264 načinu zapisa.

Kamera nudi i snimanje u neobrađenom načinu zapisa, no zbog ograničene pohrane i brzine zapisa ono se neće koristiti u ovom radu.

Slika 3.2: Raspberry Pi kamera verzije 1.3 korištena prilikom prikupljanja podataka u sklopu ovog diplomskog rada

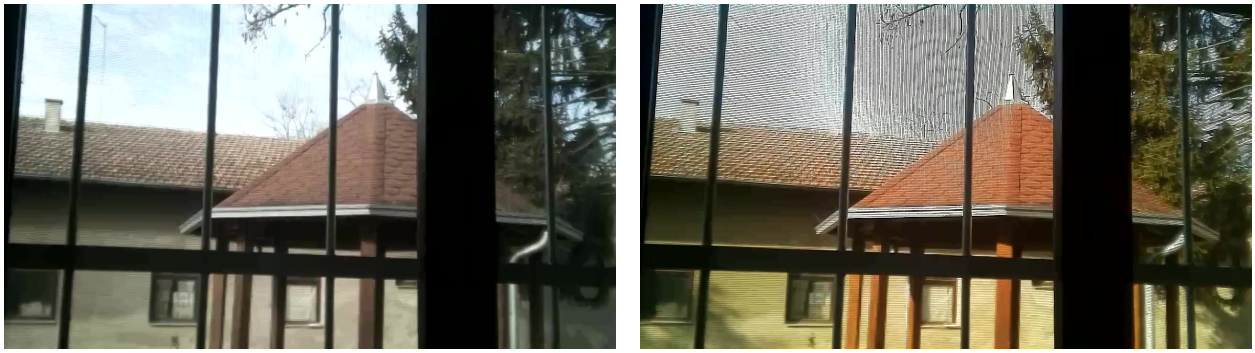
Za povezivanje s kamerom potreban je 15-pinski trakasti kabel koji se povezuje na sučelje za kameru (engl. *Camera Serial Interface*). Potrebno je pripaziti da je kabel dobro postavljen, odnosno da izvodi dodiruju kontaktnu stranu utičnice. Nakon priključenja kabla potrebno je omogućiti kameru unutar Raspberry Pi operacijskog sustava. Na novijim operacijskim sustavima (*Bullseye* i nadalje) koristi se noviji alat *libcamera*[15], gdje se instalacijom automatski detektira je li priključena kamera na mikro-računalo. U ovom radu korištena je *Bullseye* verzija. Ako se koristi starija verzija operacijskog sustava ili se želi koristiti starije okruženje *raspicam*, potrebno je omogućiti korištenje kamere u alatu naredbenog retka `raspi-config`.

Preporuča se korištenje novijeg okruženja *libcamera* zbog dugoročnih podrški i novih značajki koje će se s vremenom dodavati. Alat *libcamera* je alat naredbenog retka za manipulaciju kamerom. Omogućava fotografiranje i snimanje kamerom uz razne parametre, kako bi se kamera što ispravnije primijenila. Prilikom korištenja naredbi pokreće se automatska provjera koja detektira koja je kamera povezana, kako bi se osigurali najbolji parametri ukoliko ih korisnik nije unio samostalno. Prilikom korištenja kamere moguće je dodati konfiguracijsku datoteku koja sadrži potrebne parametre za snimanje. Tim načinom omogućeno je ponavljanja istih uvjeta snimanja u različitim vremenima. U isječku koda na slici 3.3 prikazane su postavke koje se nisu mijenjale za sva snimanja koja su obavljena.

```
nopreview=true          # bez pregleda
timeout=0               # snimanje dok se ne prekine
height=720              # visina (broj elemenata na slici)
width=1280              # širina (broj elemenata na slici)
framerate=10            # broj sličica u sekundi
exposure=sport          # ekspozicija
sharpness=2.0           # parametar za oštrinu slike
awb=auto                # automatski balans bijele boje
saturation=1.2          # zasićenje boja
```

Slika 3.3: Isječak koda za konfiguracijsku datoteku *config.txt* u kojoj su prikazane postavke kamere koje se nisu mijenjale za sva snimanja koja su obavljena

U konfiguracijskoj datoteci nalaze se parametri slike poput razlučivosti i FPS. Također, postoje parametri koji određuju ponašanje alata: ***nopreview= true*** označava da se ne radi pregled slike jer se radi na sustavu bez grafičkog sučelja. Može se odrediti trajanje snimanja s parametrom ***timeout***. Ako se parametar ***timeout*** postavi na 0, snimanje će trajati dok se ručno ne zaustavi. Moguće je podesiti i postavke senzora kao što su ekspozicija, oštrina, balans bijele boje i zasićenja boje. Na slici 3.4 prikazane su dvije slike, gdje se vidi razlika prilikom mijenjanja zadane konfiguracije snimanja.



(a) (b)

Slika 3.4: Usporedba slika dobivenih uz različite vrijednosti konfiguracijskih parametara kamere (a) slika dobivena uz exposure=normal, sharpensss=1.0, saturation=1.0 (b) slika dobivena uz exposure=sport, sharpensss=2.0, saturation=1.2

Za konfiguraciju korištenu prilikom dobivanja slike 3.4 (b) može se vidjeti da su boje izraženije i da su prijelazi između boja oštriji. Snimanje se izvodi koristeći sljedeću naredbu:

```
libcamera-jpeg --config config.txt -o image.png
```

Iz izlazne datoteke prepoznaje se format zapisa slike, koji je u ovom slučaju PNG.

Libcamera nudi brojne mogućnosti snimanja i pokretanja poslužitelja za udaljeno pregledavanje trenutne slike. Takav način povezivanja služi za podešavanje pozicije kamere u autu. Kod pokretanja skripte za snimanje, stvara se poslužitelj na kojeg se moguće povezati mobitelom koristeći aplikaciju VLC. Potrebno je da su uređaji spojeni na istu lokalnu mrežu. Nakon završetka podešavanja pozicije kamere i zatvaranjem aplikacije na mobitelu, nastavlja se prikupljanje podataka koje će biti objašnjeno u nastavku. U nastavku nalazi se naredba za pokretanje poslužitelja za udaljeni dohvat slike:

```
libcamera-vid --config config.txt --inline --listen -o tcp://0.0.0.0:5555
```

Zatim se pokreće naredba za snimanje video zapisa u H.264 formatu. Uz prethodne konfiguracijske parametre dodani su ***--save-pts*** i ***-s***. Parametar ***--save-pts*** omogućava stvaranje zasebne .txt datoteke koja sadrži vremenski odmak trenutka snimanja svake sličice od samog početka snimanja. Te će informacije kasnije služiti za sinkronizaciju sličica i podataka kuta zakreta upravljača. Parametar ***-s*** služi da se može prekinuti snimanje s prekidnim signalom ***SIGUSR2***. Tim načinom može se poslati naredba alatu da zaustavi snimanje i da ne dođe do pogreške zbog podataka koji bi ostali u radnoj memoriji. Naredba se pokreće u pozadini koristeći & operator i nalazi se u nastavku:

```
libcamera-vid --config config.txt -o output.h264 --save-pts points.txt -s &
```

3.2.2. H.264 standard za kompresiju videa

Spremanje video zapisa u neobrađenom formatu nije prikladno rješenje zbog nedostatka prostora za pohranu i brzine zapisa. Zbog toga se koriste razne tehnike kompresije videa kako bi pohrana bila brža, a utrošak memorije znatno manji. Jedan od najčešće korištenih standarda današnjice je H.264 standard kompresije. Standard je nastao 2004. godine, a 2018. godine anketa je pokazala da ga koristi 92% programera u video industriji[16].

Raspberry Pi kamera ima mogućnost snimanja u dvama formata, a to su H.264 i MJPEG. MJPEG radi kompresiju za svaku sliku zasebno, od čega je dobio ime pokretni JPEG (engl. *Motion JPEG*). H.264 koristi mnoge napredne tehnike kompresije videa i uključuje i iskorištavanje vremenske zalihosti u videu. Uz navedeno, H.264 nudi do 80% brži zapis od MJPEG formata uz jednaku kvalitetu slike[17]. Također, velika prednost korištenja H.264 formata je sklopovski ubrzano kodiranje videa (engl. *Hardware Accelerated Encoding*) pomoću jedinice za grafičku obradu (engl. *Graphics Processing Unit*).

3.2.3. Pretvaranje video toka u sličice

Kako bi se generirale sličice iz H.264 video toka, potreban je dodatni alat. Zbog složenosti kompresije korišten je alat naredbenog retka *ffmpeg* [18]. Alat u potpunosti dekodira video te ga zapisuje u obliku kojem korisnik želi. Za format slike odabran je PNG format zbog kompresije bez gubitka. Takav format zapisa slika zauzima više memorije od JPEG zapisa, ali se ne gubi na kvaliteti slike. Kreiranje slika vrši se sljedećom naredbom:

```
ffmpeg -hide_banner -i video.h264 frame%06d.png
```

Parametar *hide_banner* uklanja ispis informacije o alatu na standardnom izlazu izvođenja zbog preglednosti, *-i* predstavlja ulaznu datoteku, odnosno putanju do H.264 video toka. Zadnji parametar predstavlja naziv izlazne datoteke *frame\%06d.png*. Iz naziva definira se inkrementalni zapis slika, odnosno da svaka slika ima jedinstveni naziv i određeni format zapisa. Takav zapis *\%06d* omogućava da su nazivi svih slika jednakih duljina jer se svakom broju slike dopisuju nule kako bi broj imao 6 znamenki. Na primjer, *frame000001.png* je prva slika i prilikom sortiranja u sustavu datoteka bit će na prvom mjestu, jer se sortiranje provodi od prvog znaka imena datoteka prema zadnjem.

3.3. Prikupljanje podataka o kutu zakreta upravljača

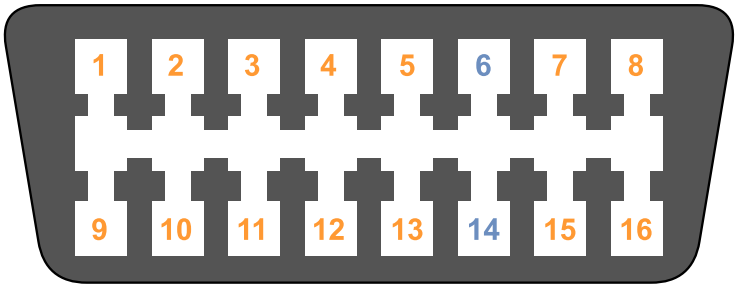
Prethodno prikupljene slike vožnje ne predstavljaju skup podataka jer su one samo ulazni podaci potrebni za razvoj algoritma autonomne vožnje zasnovanog na procjeni kuta zakreta upravljača. Izlazna veličina modela je kut zakreta upravljača vozila. Kut zakreta upravljača moguće je dohvatiti sa senzora kuta volana. Senzor se nalazi unutar vozila i bez otvaranja upravljačke ploče ne može mu se pristupiti. Proučavanjem tehničkih specifikacija (engl. *data sheet*) različitih senzora, pronađeno je da senzori koriste CAN sabirnicu za prijenos podataka do upravljačke jedinice motora.

3.3.1. CAN sabirnica

CAN je standard za komunikaciju različitih mikro-upravljača. Koristi samo dvije žice koje su suprotnih polariteta, čime se smanjuje utjecaj smetnji. Postoje CAN High i CAN Low signali, gdje ako signali nisu suprotnih vrijednosti, može se zaključiti da je došlo do smetnje. Razvijen je od strane Bosch-a za auto industriju. Teoretski je moguće povezati do 127 uređaja na CAN sabirnicu zbog broja jedinstvenih adresa koje je moguće zapisati u

7-bitnom broju u zaglavlje CAN poruke. Bitno je da svaki uređaj ima jedinstvenu adresu koja određuje taj uređaj unutar mreže. Prilikom slanja poruke svaki uređaj može vidjeti poruku koja je poslana pa i sam pošiljatelj. Prednost će imati poruka s nižom jedinstvenom oznakom ako dva ili više uređaja pokušaju poslati poruku u isto vrijeme. Svaka poruka osim jedinstvene adrese sadrži kontrolna zaglavlja, zaglavlja za provjeru kontrolnog zbroja i podatke koji u klasičnom CAN standardu mogu biti do 8 bajta veličine. Maksimalna brzina prijenosa poruka CAN sabirnicom je 1 Mbit/s. Najčešće korištene brzine prijenosa podataka u automobilima su 250 Kbit/s ili 500 Kbit/s. Postoje novije implementacije CAN sabirnice kao što je CANFD, koje omogućavaju veće pakete i brže prijenose, no još nisu često korištene u auto industriji zbog veće složenosti.

Korisnik se može povezati na CAN sabirnicu preko OBD2 sučelja. OBD2 je standard za auto dijagnostiku koja se koristi za provjeru grešaka u vozilu. Prvi automobili koji su koristili OBD2 napravljeni su 1994 godine[19]. Standard je postao obavezan u automobilima 1996. godine u Sjedinjenim Američkim Državama, a u Europi tek 2003. godine[20]. Osim CAN sabirnice, OBD2 sadrži izvode za druga komunikacijska sučelja, gdje su neka specifična za pojedine proizvođače vozila. Također je moguće i napajati uređaj preko OBD2 sučelja jer ima vezu s baterijom vozila i uzemljenjem. U ovom radu, korišteni su samo izvodi za CAN sabirnicu obojani plavom bojom na slici 3.5.

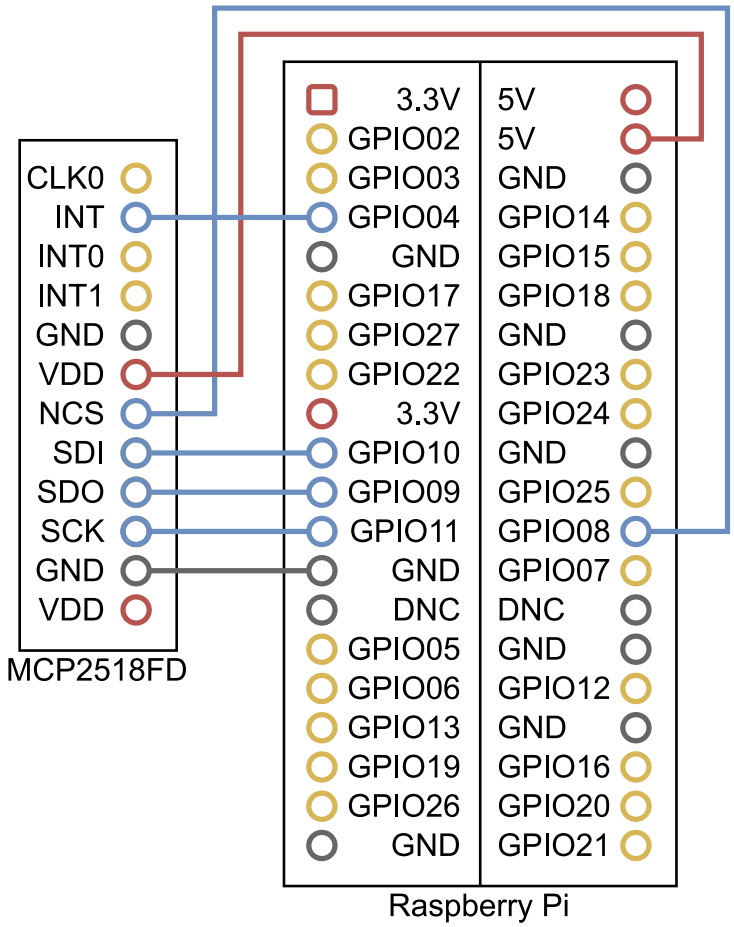


Izvod	Opis	Izvod	Opis
1	Opcija dobavljača	9	Opcija dobavljača
2	J1850 sabirnica +	10	J1850 sabirnica
3	Opcija dobavljača	11	Opcija dobavljača
4	Uzemljenje šasije	12	Uzemljenje šasije
5	Uzemljenje signala	13	Uzemljenje signala
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Opcija dobavljača	16	Napon baterije

Slika 3.5: OBD2 sučelje i opis njegovih izvoda

3.3.2. Povezivanje mikro-računala na CAN sabirnicu

Za povezivanje na sabirnicu potreban je CAN mikro-upravljač. Postoje razne verzije proizvođača kao što su Siemens, Infineon, Texas Instruments, Microchip Technology i ostali. U ovom radu korišten je MCP2518FD mikro-upravljač kojeg su razvili Microchip Technology. Kako bi se povezao s drugim uređajima, MCP2518FD mikro-upravljač koristi SPI (engl. *Serial Peripheral Interface*) komunikaciju. SPI komunikacija omogućava povezivanje više sporednih uređaja na glavni. SPI koristi četiri izvoda za komunikaciju te se odvija u oba smjera. Jedna od prednosti korištenog mikro-upravljača je da uz SPI komunikaciju koristi prekidni (engl. *interrupt*) izvod kojim signalizira mikro-računalu da je stigla poruka. Na taj način omogućava se brže primanje poruka jer se ne mora neprestano provjeravati je li stigla poruka. Na slici 3.6 prikazan je način fizičkog povezivanja mikro-upravljača na mikro-računalo Raspberry Pi putem SPI komunikacije i izvoda za prekid.



Slika 3.6: Prikaz načina fizičkog povezivanja mikro-upravljača s Raspberry Pi mikro-računalom

Nakon što su fizički povezani, potrebno je dodati sljedeću liniju koda u datoteku koja se

nalazi na SD kartici na putanji `/boot/config.txt`:

```
dtoverlay=mcp251xfd,spi0-0,interrupt=4
```

Datoteka se koristi kako bi se podesili parametri prilikom pokretanja operacijskog sustava. Potrebno je omogućiti komunikaciju na SPI0 sučelju s upravljačkim programom za `mcp251xfd`, što uključuje odabrani mikro-upravljač. Također je potrebno odabrati izvod na kojeg je povezan izvod za prekide. U datoteci je navedeno da se koristi izvod GPIO04 za prekide.

Nakon uspješnog povezivanja, potrebno je postaviti sučelje koje omogućava slanje i primanje CAN poruka unutar operacijskog sustava. Za manipulaciju CAN porukama koristi se alat naredbenog retka *can-utils*[21]. Nakon instalacije alata, moguće je dodati sučelje za komunikaciju unutar Linux mrežnog stoga (engl. *linux network stack*). Tip sučelja je *can* koji dolazi iz *can-utils* alata. Kod postavljanja sučelja potrebno je podesiti brzinu CAN sabirnice. U ovom radu prikupljanje skupa podataka provedeno je u automobilima Opel Corsa D 2010. godište i Renault Megane 2008. godište. Metodom pokušaja i pogrešaka otkriveno je da se u automobilu Opel Corsa koristi brzina 500 Kbit/s. Tu brzinu potrebno je podesiti prilikom postavljanja CAN sučelja. Kasnije je otkriveno da Renault Megane također koristi brzinu prijenosa podataka od 500 Kbit/s na CAN sabirnici. Sljedeća naredba prikazuje instalaciju alata na Ubuntu 20.04 operacijskom sustavu:

```
sudo apt update && sudo apt install can-utils
```

Zatim je potrebno inicijalizirati komunikacijsko sučelje sljedećom naredbom gdje se navodi sučelje, tip sučelja i brzina prijenosa podataka:

```
sudo ip link set can0 up type can bitrate 500000
```

Slušanje sučelja, odnosno primanje poruka dobiva se sljedećom naredbom:

```
candump can0
```

Slanje CAN poruke provodi se sljedećom naredbom, gdje je potrebno navesti komunikacijsko sučelje i poruku koja uključuje jedinstvenu adresu uređaja i sadržaj poruke:

```
cansend can0 123##DEADBEEF
```

Adresa i sadržaj odvojeni su znakom #.

Kako bi se lakše pregledavale poruke, korišteno je prijenosno računalo koje je umreženo s mikro-računalom. Računalo je povezano putem SSH protokola na Raspberry Pi, gdje je izvršena naredba: *candump can0*. Na slici 3.7 prikazan je dobiveni izlaz nakon što se pokrene naredba *candump*, gdje su prikazane sve poruke koje prolaze kroz CAN sabirnicu. Svaka poruka sadrži jedinstvenu adresu, veličinu i sadržaj poruke. Na primjer, ako se analizira poruka iz prvog retka sa slike 3.7, može se vidjeti da je poruka stigla na sučelje *can0*. Zatim slijedi adresa uređaja koji je poslao poruku. U ovoj poruci adresa je jednaka `0x1BC`. Nakon toga u uglatim zagradama zapisana je duljina poruke, ovdje je duljina 8 bajta. Na kraju poruke nalazi se N bajtova u heksadecimalnom zapisu, gdje je N duljina poruke, a bajtovi odgovaraju sadržaju poruke.

Slika 3.7: CAN poruke na zaslonu prijenosnog računala

Alat `candump` omogućava preusmjeravanje poruka sa standardnog izlaza u datoteku. Za omogućavanje zapisivanja u datoteku koristi se parametar `-l`. Zapisivanjem kreira se `.log` datoteka koja sadrži sve poruke do završetka snimanja. Svaka poruka sadrži vrijeme zapisivanja i sadržaj poruke. Vrijeme zapisivanja omogućuje sinkronizaciju kuta zakreta upravljača i prethodno navedenih slika kamere. U isječku na slici 3.8 prikazani su primjeri poruke iz datoteke.

```
(1639138587.770849) can0 1BC#0801000048FFC000
(1639138587.773821) can0 130#CE00B80381780000
(1639138587.774073) can0 1E5#46FEDAC000783AE0
(1639138587.777911) can0 0C1#1000000010000000
```

Slika 3.8: Isječak koda za zapisničku datoteku CAN poruka

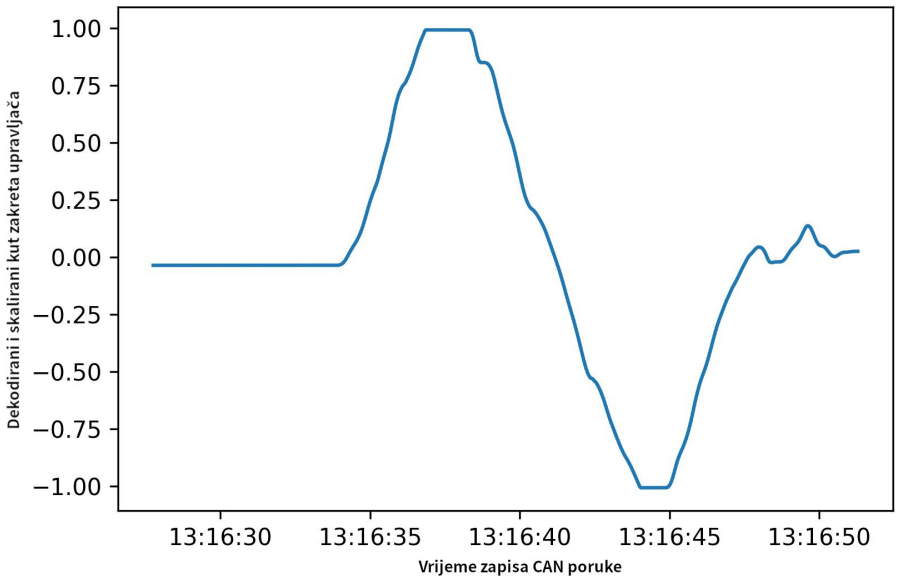
3.3.3. Dekodiranje poruka senzora kuta upravljača

CAN poruke osmišljene su kako bi prenijele što više informacija unutar jedne poruke. Svaki uređaj koristi jednu jedinstvenu adresu te se svi podaci moraju nalaziti u jednoj poruci. U trenutku kad je sustav bio spreman za korištenje, nije bila poznata CAN adresa senzora kuta zakreta upravljača, niti je bio poznat način na koji se šalju podaci. Zbog toga ih je trebalo ručno dekodirati. Odnosno, potrebno je bilo pronaći koja je adresa senzora kuta zakreta upravljača i kako su zapisani podaci u CAN poruci. Kako bi se podaci lakše dekodirali, potrebno je znati otprilike što predstavljaju. Započeto je snimanje svih CAN poruka i okrenut je upravljač u potpuno lijevu pa potom u potpuno desnu stranu. Na taj način podaci bi nakon dekodiranja trebali izgledati poput sinusoide. U projektu "Hacking a Chevy Volt Into a Mario Kart 64 Controller"[22] korišten je upravljač automobila za upravljanje vozilom u igri Mario Kart, gdje je CAN adresa senzora kuta zakreta upravljača iznosila *0x1E5*. Nakon filtriranja snimljenih poruka s adresom iz projekta, dobiven je velik broj poruka s uzorkovanjem 10 puta u sekundi. Takva frekvencija slanja CAN poruka odgovara Boschovom senzoru kuta upravljača, kao što je navedeno u njihovom priručniku[23].

Poruke s adresom *1x1E5* veličine su 8 bajta. Kut zakreta upravljača nije moguće precizno prikazati samo s jednim bajtom te je vjerojatno potrebno dva bajta kako bi promjena kuta bila fluidna. Postoje različiti načini pretvaranja sirovih bitova u broj. Potrebno je bilo provjeriti koristi li se najznačajniji bit (engl. *MSB - most significant bit*) ili najmanje značajan bit (engl. *LSB - least significant bit*), što znači nalazi li se bit s najvećom vrijednošću na lijevoj ili desnoj strani. Također, potrebno je bilo otkriti radi li se o broju s ili bez predznaka. To predstavlja 4 kombinacije za svaku poziciju, kojih ima ukupno 7 u 8 bajta podataka u poruci, što rezultira s 28 mogućih kombinacija dekodiranja poruke. Za svaku kombinaciju dekodiranja poruka nacrtan je graf podataka kroz vrijeme. Pregledane su sve slike sve dok nije prepoznata sinusoida. Pokazalo se da je duljina podatka 16 bita i započinje na 14. bitu. Podaci su zapisani s predznakom i najznačajnijim bitom. Na slici 3.9 prikazani

su dekodirani podaci sa snimke. Kako bi se skalirali u rasponu [-1, 1] potrebno je sve brojeve podijeliti s 2^{14}. Iako je duljina podatka 16 bita, dijeli se s 2^{14} jer je to najveći zapisani broj. Prvi bit odgovara predznaku broja, a drugi bit je uvijek 0 u ovom zapisu. Ostalih 14 bitova predstavljaju kut zakreta upravljača.

Automatizirano dekodiranje provedeno je pomoću *Python* programskog jezika. Manipulacija CAN poruka napravljena je pomoću biblioteke *python-can*[24]. Dekodiranje je napravljeno pomoću .dbc zapisa. DBC datoteka naziva se još CAN bazom podataka. Koristi se za pretvorbu CAN poruka u dekodirane podatke jer sadrži opise signala. Korištenjem DBC datoteke omogućava se dodavanje različitih opisa signala kako bi se dekodirale različite poruke. Uz Opel Corsu, napravljeno je i dekodiranje poruka za Renault Megane 2008. godište, ponavljanjem postupka dekodiranja poruka za Opel Corsu. Senzor kuta upravljača u Renault Megane automobilu bio je različit te koristi drugu jedinstvenu adresu. Početni bit podatka kuta zakreta unutar CAN poruke ne odgovara prethodnom, već započinje na 5. bitu. Također, zamijenjene su lijeva i desna strane pa je bilo potrebno podijeliti rezultat s negativnom vrijednošću maksimalnog kuta zakreta upravljača, kako bi odgovarali kutovima zabilježenim u automobilu Opel Corsa. U isječku koda na slici 3.10 prikazan je format za dobivanje kuta zakreta upravljača u vozilima Opel Corsa D 2010. godište i Renault Megane 2008. godište.



Slika 3.9: Dekodirani podaci kuta zakreta upravljača

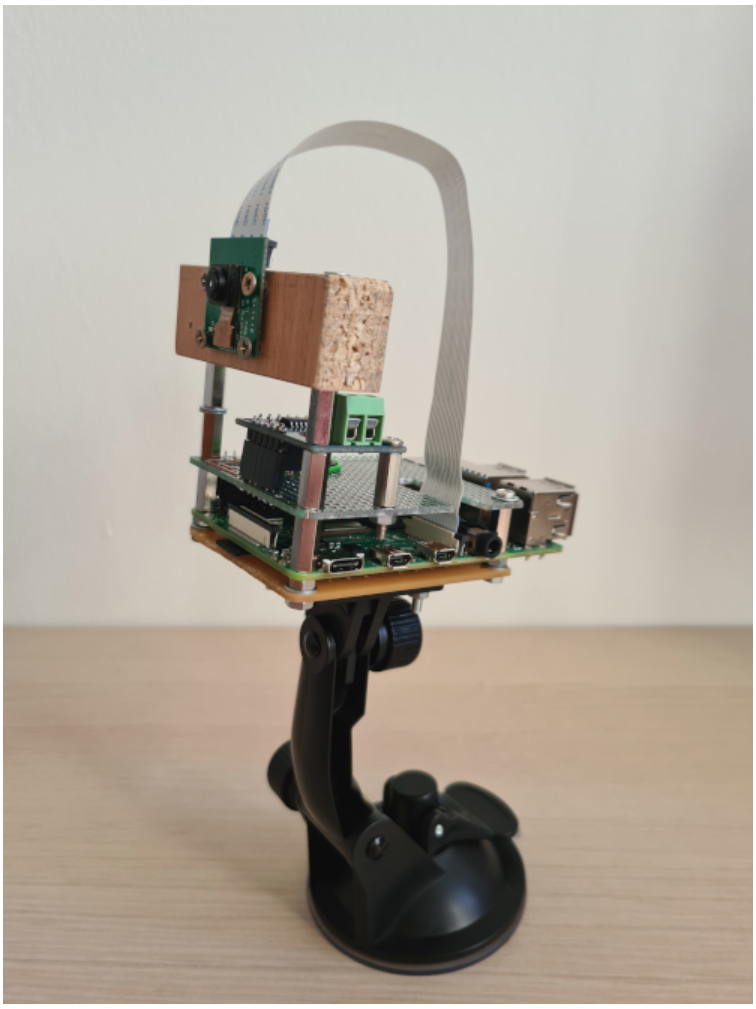
```
% Opel Corsa D 2010. godište
BO_ 485 SAS_0: 8 Vector_XXX
    SG_ angle : 14|16@0- (0.00006103515625,0) [-1|1] "degrees" Vector_XXX

% Renault Megane 2008. godište
BO_ 194 SAS_1: 6 Vector_XXX
    SG_ angle : 5|16@0- (-0.000043756016452262186,0) [-1|1] "degrees"
↪   Vector_XXX
```

Slika 3.10: Isječak koda za DBC zapis pretvorbe signala CAN poruka

3.4. Montiranje mikro-računala i kamere u automobilu

Kako bi skup podataka bio ispravno kreiran, potrebno je za sve snimke osigurati fiksnu poziciju kamere. Kamera mora biti smještena na prednjem kraju vozila i snimati prostor ispred vozila. Zbog ograničenja kratkog kabla između kamere i mikro-računala, potrebno ih je oboje montirati u područje blizu unutrašnjeg retrovizora na prednjem vjetrobranskom staklu. To je područje odabrano jer se nalazi u sredini vozila s dobrim kutom pregleda prostora ispred vozila. Mikro-računalo i kamera pričvršćeni su na nosač tableta koji se pomoću vakuumskog pričvršćivanja montira na unutrašnjost vjetrobranskog stakla.

Da bi Raspberry Pi djelovao kao cjelina zajedno s MCP2518 mikro-upravljačem, u sklopu ovog rada napravljena je razvojna pločica. Razvojna pločica uklanja smetnje na vodovima koje se pojavljuju pri korištenju žica za povezivanje mikro-računala i mikro-upravljača. Također, prednost razvojne pločice je mogućnost zamjene MCP2518 upravljača u slučaju da dođe do kvara na njemu. Na samu razvojnu pločicu okomito je montirana kamera i okrenuta prema prostoru ispred vozila. Nosač sustava mikro-računala, mikro-upravljača i kamere prikazan je na slici 3.11, a na slici 3.12 prikazan je montirani nosač sustav u automobilu Opel Corsa.



Slika 3.11: Nosač sustava Raspberry Pi, mikro-upravljača MCP2518 i kamere

Slika 3.12: Montirani nosač sustava u automobilu Opel Corsa

3.5. Sinkronizacija trenutka uzimanja slike i trenutka očitavanja kuta zakreta upravljača

Zapisivanje CAN poruka u datoteku uključuje vrijeme njihovog dolaska na računalo. Takav pristup omogućava uvid u vrijednost kuta zakreta upravljača u svakom trenutku. Za svaku sliku kamere potrebno je pronaći odgovarajući kut zakreta upravljača. Zapisivanjem vremenskih markera svake slike može se prepoznati vrijeme proteklo od početka snimanja pa do trenutka snimanja određene slike. Vrijeme početka snimanja nije jednako kao vrijeme pokretanja naredbe, nego je potrebno dohvatiti vrijeme stvaranja H264 datoteke, jer ona nastaje zapisom prve slike. Taj trenutak smatra se početkom snimke. Koristeći sljedeću naredbu može se dobiti vrijeme stvaranje H264 datoteke iz Linux okruženja. Koristeći vrijeme kreiranja datoteke i vrijeme proteklo od početka snimanja pa do snimanja svake slike, može se izračunati trenutak snimanja pojedine slike:

```
stat -c %x video.h264    # 2022-03-02 21:02:50.832160995 +0200
```

Sinkronizacija se postiže tako da se za svaku sličicu i njezin trenutak snimanja pronalazi kut zakret upravljača čiji je vremenski marker po vrijednosti najbliži trenutku snimanja slike, a da se nalazi prije tog trenutka snimanja slike. Isječak koda za određivanje kuta zakreta za svaku sličicu prikazan je na slici 3.13.

```
for frame_index in range(len(frame_times)):
    try:
        steering_data_id = next(i for i,m in enumerate(steering_data) if
        ↪  m['time'] > frame_times[frame_index]) + last_msg_id
    except StopIteration:
        break

    steering_angle = steering_data[steering_data_id]['angle']
    csv_data.append([f'frames/{frame_names[frame_index]}', steering_angle])
```

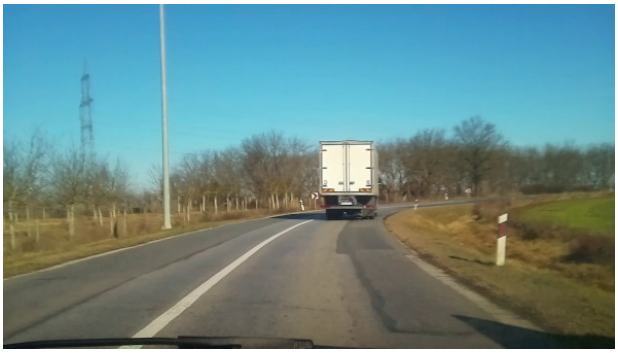
Slika 3.13: Isječak koda za sinkronizaciju vremena stvaranja slika i vremena zapisivanja kuta zakreta upravljača

3.6. Analiza novonastalog skupa podataka

Novonastali skup podataka nazvan je CanPi, zbog korištenog mikro-računala Raspberry Pi i CAN protokol za prikupljanje podataka iz dijagnostičkog sučelja vozila. Konačni CanPi

skup podataka sadrži 266933 slika kojima su pridruženi pripadni kutovi zakreta upravljača. Kao što je pisalo ranije, snimanje je provedeno koristeći 2 automobila - Opel Corsa D proizvedena 2010. godine i Renault Megane proizveden 2008. godine. Snimke vožnje nastale su na širim područjima gradova Osijeka, Požege i Iloka, što uključuje vožnje po gradu i izvan grada. Sveukupno je napravljeno je 17 snimki. Nazivi snimki generirani su na način da sadrže datum i vrijeme pokretanja snimanja. Tim načinom osigurali su se jedinstveni nazivi za različite snimke. Snimke sadrže vožnju po danu i noći, uključeni su vedri, oblačni i kišoviti vremenski uvjeti. Na slici 3.14 prikazani su primjeri slika dobivenih tijekom snimanja.

(a)



(b)

(c)

(d)

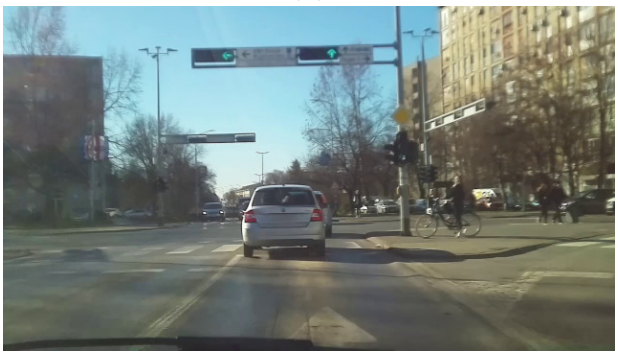
(e)

(f)

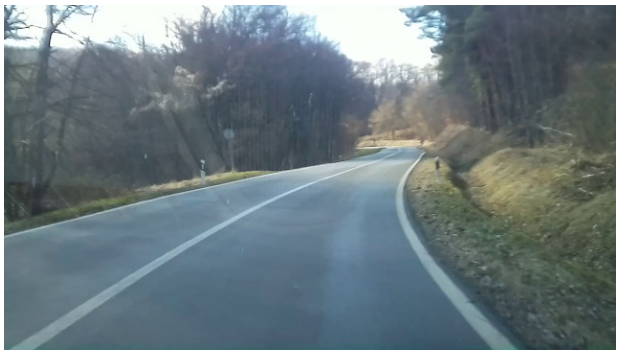
(g)

(h)

(i)



(j)



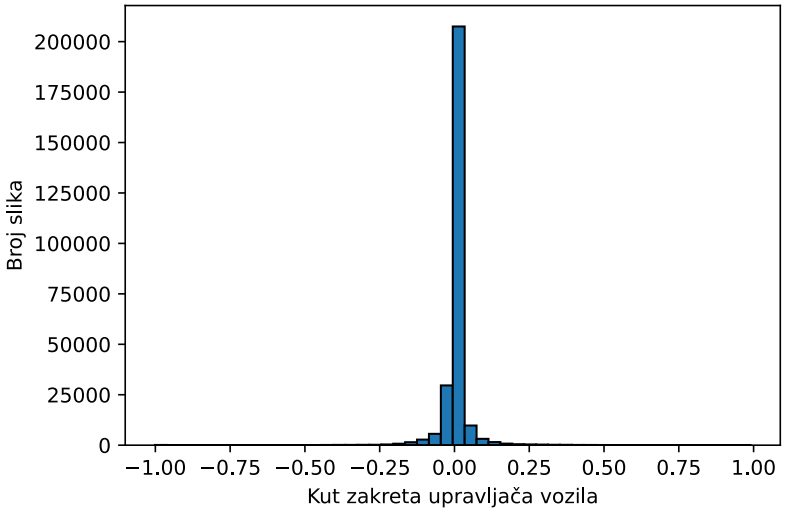
(k)

(l)

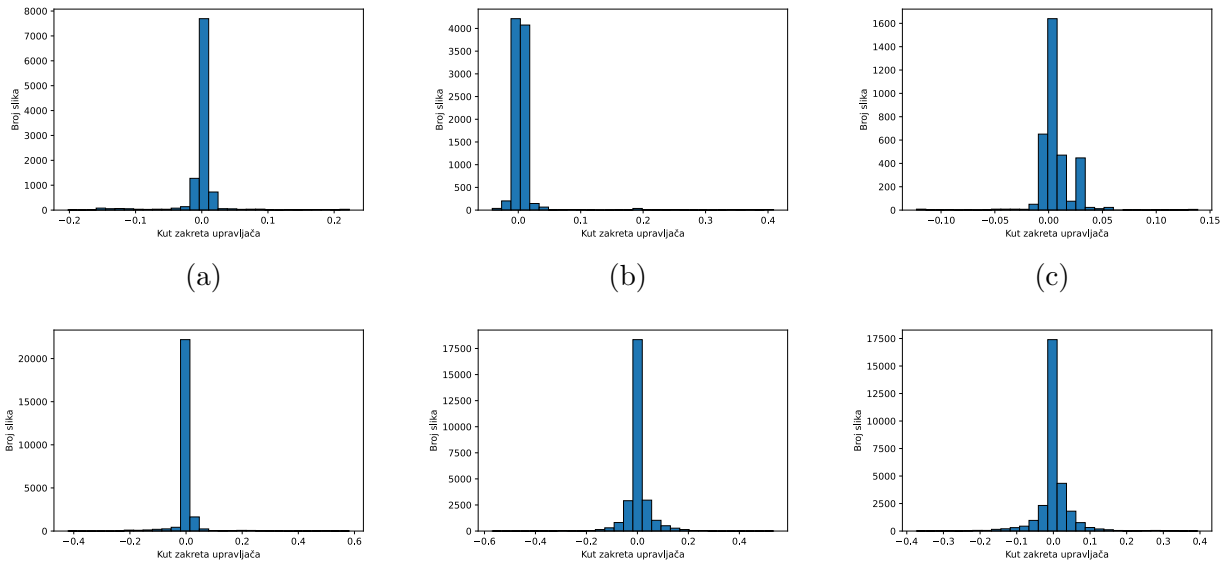
Slika 3.14: Primjer slika iz CanPi skupa podataka (a) urbano područje, sunčano vrijeme, dan, kut zakreta upravljača 2.31°, (b) ruralno područje, sunčano vrijeme, dan, kut zakreta upravljača −3.11°, (c) brza cesta, sunčano vrijeme, dan, kut zakreta upravljača 2.21°, (d) urbano područje, kišovito vrijeme, noć, kut zakreta upravljača 4.69°, (e) urbano područje, oblačno vrijeme, dan, kut zakreta upravljača 1.86°, (f) urbano područje, dan, lagano kišovito vrijeme, kut zakreta upravljača −1.11°, (g) ruralno područje, dan, sunčano vrijeme, kut zakreta upravljača 3.46°, (h) urbano područje, dan, sunčano vrijeme, kut zakreta upravljača 20.01°, (i) urbano područje, dan, oblačno vrijeme, kut zakreta upravljača 86.05°, (j) urbano područje, dan, sunčano vrijeme, kut zakreta upravljača 3.98°, (k) ruralno područje, dan, sunčano vrijeme, kut zakreta upravljača −34.04°, (l) ruralno područje, dan, oblačno vrijeme, kut zakreta upravljača 34.95°

Od ukupnog broja snimljenih slika, velik broj slika snimljen je na ravnim cestama, gdje je kut zakreta upravljača gotovo 0. To predstavlja problem jer je potrebno prikupiti i

pozamašan skup podataka koji će pomoći u učenju ponašanja vozila prilikom skretanja. Moguće je riješiti problem augmentacijom, gdje se slika umjetno (na računalu) promijeni kako bi se povećao broj slika sa zavojima kojih nedostaje i kako bi se neuronska mreža naučila ponašati u zavojima. Također, moguće je eliminirati višak slika za koje je kut zakreta 0 ili blizu 0, kako bi se napravila bolja raspodjela kutova i kako u skupu ne bi prevladavali slučajevi s kutom zakreta upravljača 0 ili vrlo blizu nule. Raspodjela kutova zakreta upravljača (tj. histogrami kutova zakreta) prikazana je na slici 3.15, a za svaku od 17 vožnji nalaze se na slici 3.16.

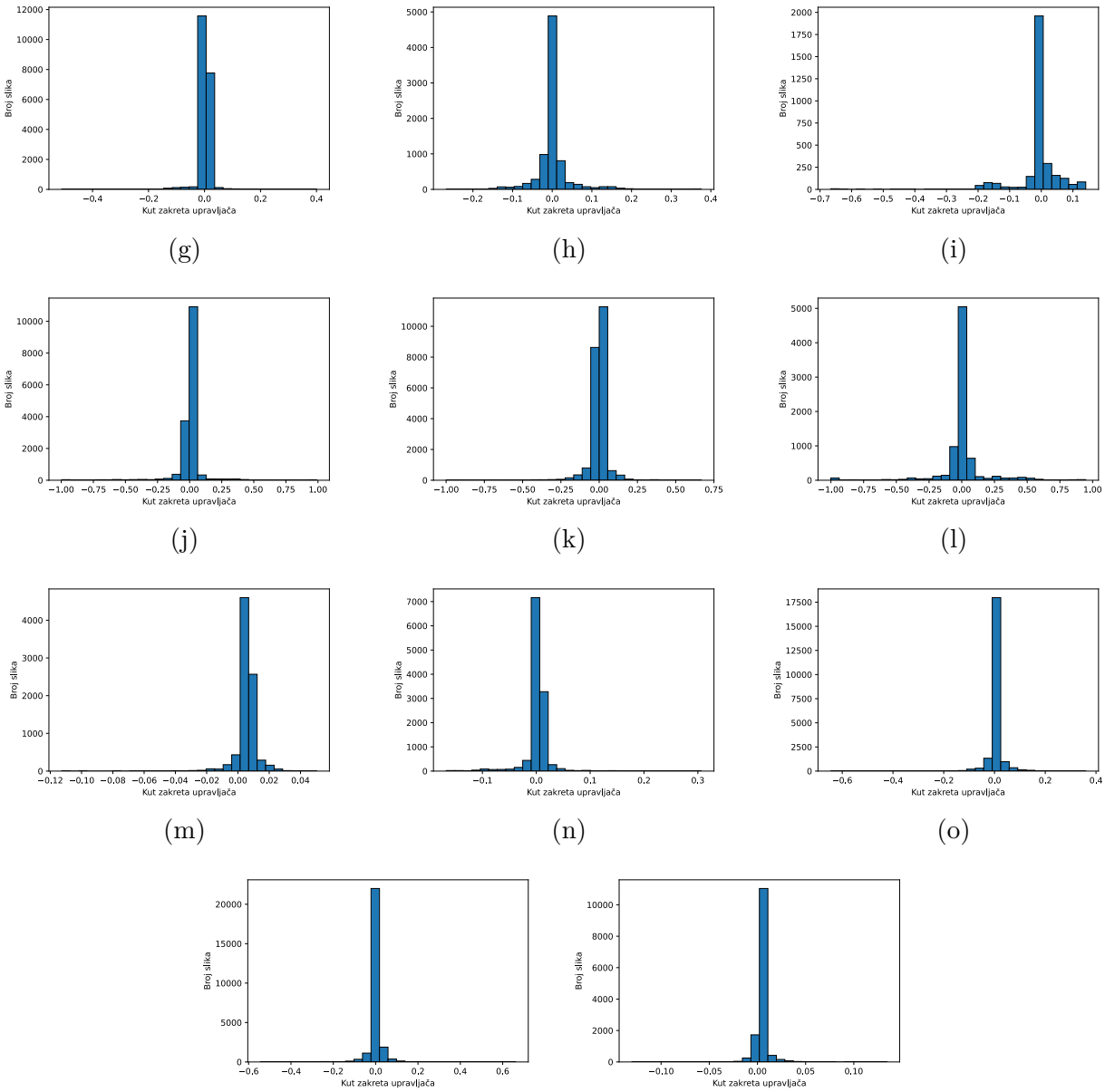


Slika 3.15: Raspodjela kutova na čitavom CanPi skupu podataka



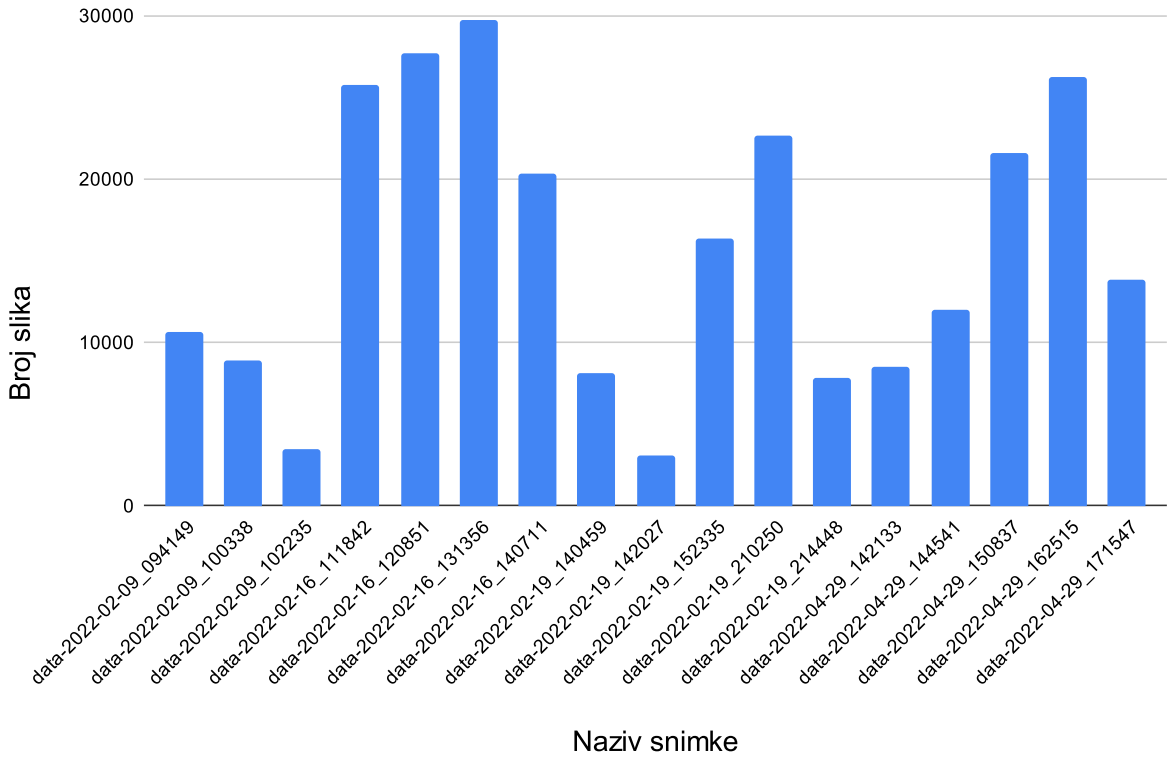
(a) (b) (c)

(d) (e) (f)



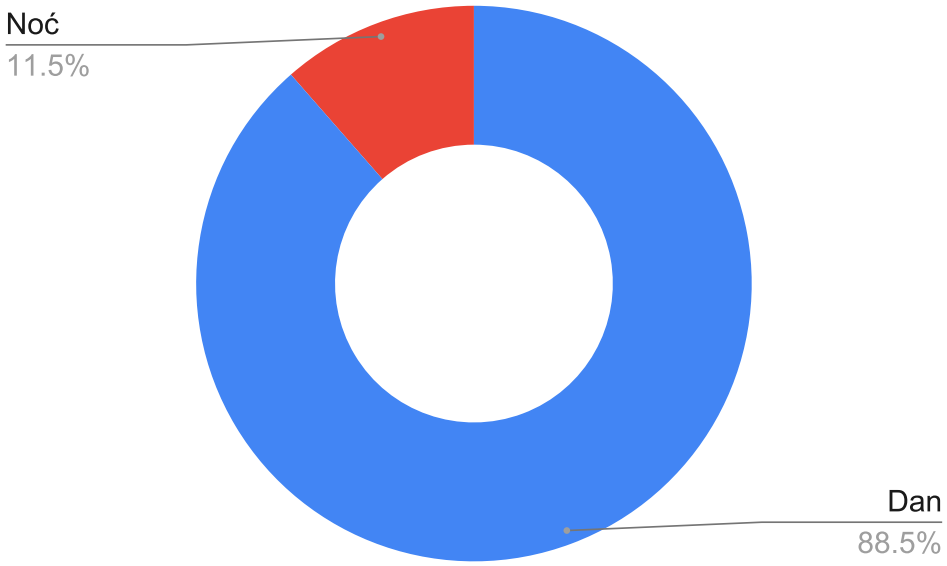
Slika 3.16: Raspodjela kutova po snimanjima (a) vožnja data-2022-02-09_094149, (b) vožnja data-2022-02-09_100338, (c) vožnja data-2022-02-09_102235, (d) vožnja data-2022-02-16_111842, (e) vožnja data-2022-02-16_120851, (f) vožnja data-2022-02-16_131356, (g) vožnja data-2022-02-16_140711, (h) vožnja data-2022-02-19_140459, (i) vožnja data-2022-02-19_142027, (j) vožnja data-2022-02-19_152335, (k) vožnja data-2022-02-19_210250, (l) vožnja data-2022-02-19_214448, (m) vožnja data-2022-04-29_142133, (n) vožnja data-2022-04-29_144541, (o) vožnja data-2022-04-29_150837, (p) vožnja data-2022-04-29_162515, (q) vožnja data-2022-04-29_171547

Na slici 3.17 prikazan je broj slika za svaku od 17 snimki. Trajanje pojedine snimke je različito. Najviše slika nalazi se u snimci *data-2022-02-16_131356*, najmanje slika sadrži snimka *data-2022-02-19_142027*.



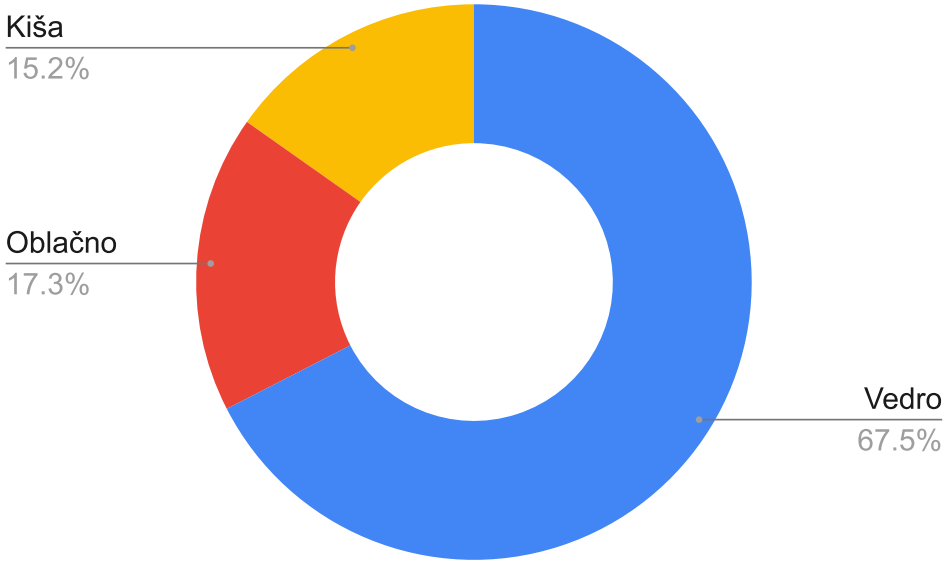
Slika 3.17: Broj slika sadržanih u svakoj od 17 snimki CanPi skupa podataka

Najviše snimki napravljeno je po danu te je evidentan problem nedostatka slika po noći. Slika snimljenih po danu ima gotovo 9 puta više od noćnih slika. Na slici 3.18 grafički je prikazan omjer slika po dobu dana u kojem su snimljene.



Slika 3.18: Omjer slika snimljenih po danu i noći

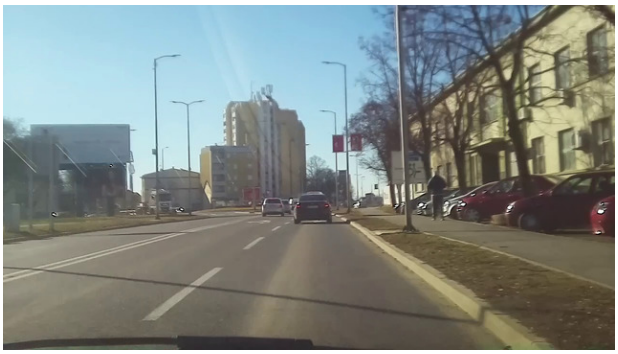
Pokušano je prikupiti snimke u što je više moguće različitih vremenskih uvjeta kao što su vedro, oblačno i kišovito, no najveći broj snimki napravljen je po vedrim vremenskim uvjetima. To ukazuje se nedostatak vremenskih uvjeta kao što su magla, snijeg. Na slici 3.19 grafički je prikazan omjer slika snimljenih po vedrom, oblačnom i kišnom vremenu.



Slika 3.19: Omjer slika snimljenih po različitim vremenskim uvjetima

Osim vremenskih uvjeta i doba dana, potrebno je bilo napraviti snimanja po različitim tipovima prometnica. Napravljena je klasifikacija prometnica na 4 kategorije: gradska cesta, seoska cesta, cesta izvan naselja i brza cesta. Kategorije prometnica prikazane su na slici 3.20. Ceste izvan naselje uključuju ceste između dvaju naselja koje nisu uređene poput brzih cesta, što znači da ponekad nemaju vanjsku kolničku crtu. U brze ceste pripadaju ceste na kojima su obavezno označene rubne kolničke crte kojima se označava granica prometnice. Na slici 3.21 grafički je prikazan omjer broja slika snimljenih na različitih tipovima cesta.

(a)

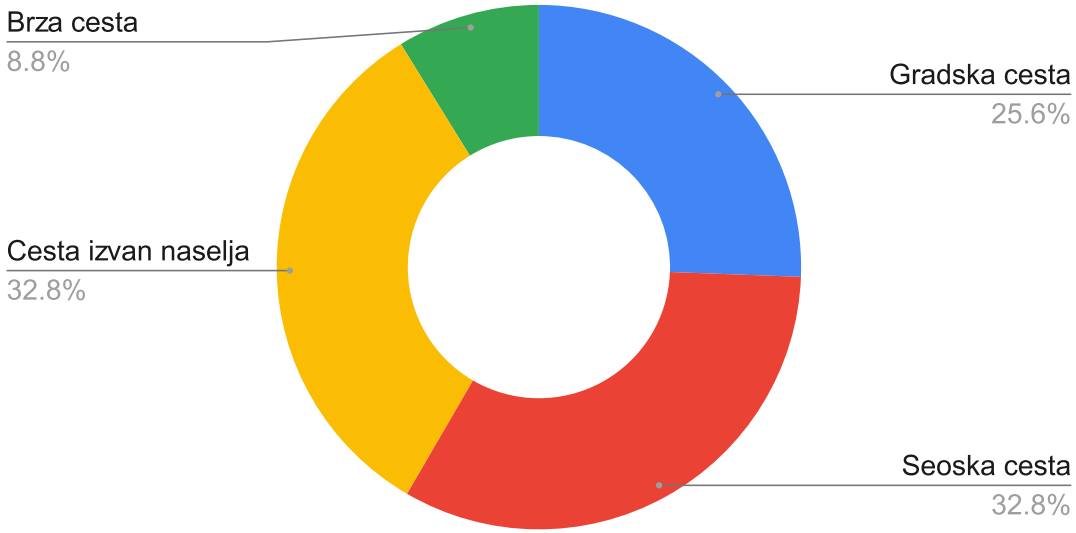


(b)

(c)

(d)

Slika 3.20: Kategorije prometnica iz skupa podataka CanPi (a) seoska cesta, (b) gradska cesta, (c) cesta izvan naselja, (d) brza cesta



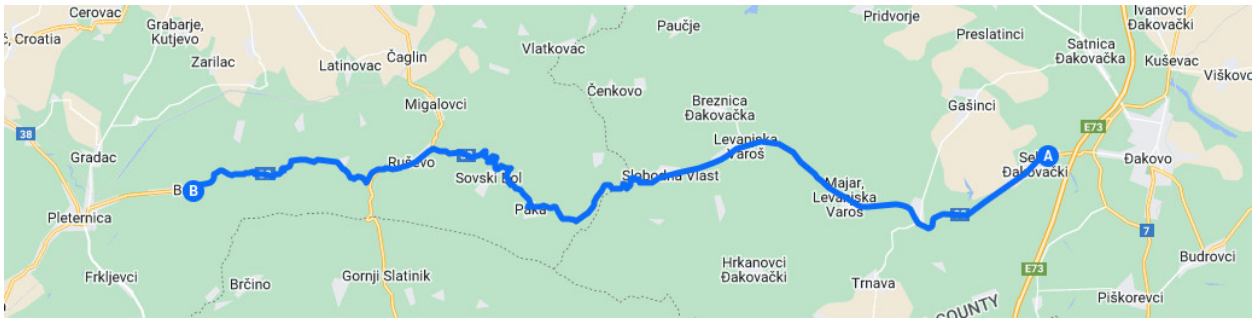
Slika 3.21: Omjer slika snimljenih na različitim tipovima cesta

Detaljan opis svakog snimanja nalazi se u tablici u elektroničkom prilogu P.3.1. Tablica sadrži podatke kao što su raspon kuta zakreta upravljača, uređaj za snimanje, rezolucija slika, broj slika, datum i vrijeme početka snimanja, putanja do karte snimanja, postavke senzora kamere, doba dana i vremenski uvjeti. Raspon kuta zakreta upravljača zabilježen je u stupnjevima kako bi se brojevi skalirali iz raspona [-1, 1] u zadani raspon kuta zakreta različitih upravljača vozila. Jedan kratki isječak tablice dane u prilogu P.3.1. dan je na slici 3.22.

		Vozilo				Oprema za snimanje		
Naziv vožnje	My Maps link	Proizvođač	Model	Godište	Raspon kuta zakreta upravljača (°)	Racunalo	Kamera	CAN
data-2022-02-09_094149	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-09_100338	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-09_102235	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-16_111842	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-16_120851	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-16_131356	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-16_140711	CanPi Dataset 1/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-19_140459	CanPi Dataset 1/2	Renault	Megane	2008. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-19_142027	CanPi Dataset 1/2	Renault	Megane	2008. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-19_152335	CanPi Dataset 1/2	Renault	Megane	2008. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-19_210250	CanPi Dataset 2/2	Renault	Megane	2008. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-02-19_214448	CanPi Dataset 2/2	Renault	Megane	2008. g	[-540, 540]	Raspberry Pi 4B (4GB)	Raspberry Pi Camera v1.3	Puni zapis
data-2022-04-29_142133	CanPi Dataset 2/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 3A (1GB)	Raspberry Pi Camera v2.0	Puni zapis
data-2022-04-29_144541	CanPi Dataset 2/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 3A (1GB)	Raspberry Pi Camera v2.0	Puni zapis
data-2022-04-29_150837	CanPi Dataset 2/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 3A (1GB)	Raspberry Pi Camera v2.0	Puni zapis
data-2022-04-29_162515	CanPi Dataset 2/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 3A (1GB)	Raspberry Pi Camera v2.0	Puni zapis
data-2022-04-29_171547	CanPi Dataset 2/2	Opel	Corsa D	2010. g	[-540, 540]	Raspberry Pi 3A (1GB)	Raspberry Pi Camera v2.0	Puni zapis

Slika 3.22: Isječak iz tablice s detaljima o pojedinim snimkama

Za svaku vožnju pohranjena je karta unutar My Maps. My Maps je usluga koja omogućuje crtanje vlastitih karti i planiranje putovanja. Unutar dviju karti napravljeni su slojevi gdje svaki sloj predstavlja jednu vožnju. Na slici 3.23 prikazan je put na kojem je nastala snimka `data-2022-02-16_120851`. Kartama svake vožnje može se pristupiti iz prethodno navedene tablice ili iz priloga P.3.2



Slika 3.23: Prikaz rute vožnje koja je služila za nastajanje snimke `data-2022-02-16_120851`

3.7. Upute za pokretanje snimanja za prikupljanja skupa podataka

Za prikupljanje skupa podataka potrebno je pokrenuti *bash* skriptu koja sadrži sve potrebne naredbe. Prije pokretanja skripte potrebno je povezati kameru i mikro-upravljač za CAN komunikaciju. Potrebne upute nalaze se u nastavku.

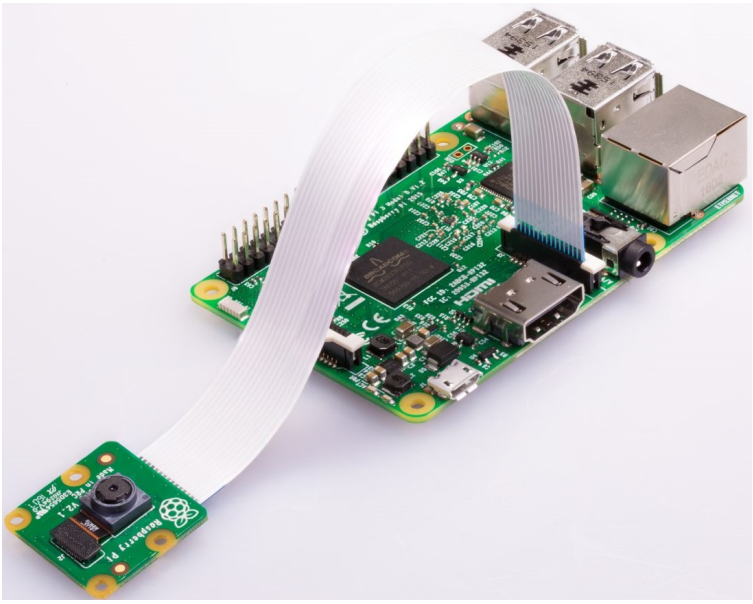
3.7.1. Instalacija operacijskog sustava Raspberry Pi OS

Mikro-računalo Raspberry Pi koristi SD karticu kao primarno područje pohrane podataka, zbog čega je potrebno pomoću drugog računala instalirati operacijski sustav

na memorijsku karticu. Operacijski sustav i alat za snimanje operacijskog sustava na memorijskoj kartici dostupni su na stranicama Raspberry Pi organizacije[25]. Nakon instalacije alata, za instalaciju sustava *Raspberry Pi Imager* potrebno je umetnuti memorijsku karticu u računalo. Odabiru se verzija operacijskog sustava i memorijska kartica na koju se želi instalirati operacijski sustav. U ovom radu korišten je operacijski sustav Raspberry Pi OS Lite (*Bullseye*). *Bullseye* verzija je bila najnovija u vrijeme izrade ovog rada te je odabrana zbog novog alat *libcamera* i automatske detekcije kamere. Nakon instalacije, memorijska kartica umeće se u Raspberry Pi mikro-računalo.

3.7.2. Povezivanje Raspberry Pi kamere na Raspberry Pi mikro-računalo

Prije uključivanja kamere potrebno je urediti datoteku `/boot/config.txt` na memorijskoj kartici gdje je instaliran Raspberry Pi OS operacijski sustav. U datoteci mora biti definirana postavka za automatsko detektiranje kamere, odnosno mora se nalaziti sljedeća linija u datoteci - *camera_ auto_ detect=1*. Nakon dodavanja linije potrebno je isključiti napajanje mikro-računala kako ne bi došlo do kvara kamere prilikom povezivanja. Povezivanje se vrši savitljivim kablom na priključku označenim tekstom CAMERA. Bitno je voditi računa o orijentaciji kabla gdje se nalaze vodljivi izvodi za povezivanje. Kod mikro-računala plava strana okrenuta je prema USB priključcima, kod kamere plava strana okrenuta je suprotno strani snimanja. Način povezivanja prikazan je na slici 3.24.



Slika 3.24: Povezivanje Raspberry Pi kamere na Raspberry Pi mikro-računalo

3.7.3. Povezivanje CAN mikro-upravljača MCP2518FD na Raspberry Pi mikro-računalo

Prije uključivanja CAN mikro-upravljača potrebno je urediti datoteku `/boot/config.txt` na memorijskoj kartici gdje je instaliran Raspberry Pi OS operacijski sustav. U datoteci mora biti definirana postavka za uspostavljanje SPI komunikacije s odabranom programskom podrškom za MCP251x grupu mikro-upravljača, odnosno mora se nalaziti sljedeća linija u datoteci - `dtoverlay=mcp251xfd,spi0-0,interrupt=4`. Nakon dodavanja linije potrebno je isključiti napajanje mikro-računala kako ne bi došlo do kvara CAN mikro-upravljača prilikom povezivanja. Potrebno je povezati izvode mikro-upravljača s izvodima mikro-računala kao što je prikazano na slici 3.6. Nakon povezivanja potrebno je uključiti mikro-računalo i instalirati alat *can-utils* koristeći sljedeću naredbu:

```
sudo apt update && sudo apt install can-utils
```

Zatim je potrebno povezati CAN High i CAN Low izvode mikro-upravljača s izvodima u vozilu koji se nalaze na dijagnostičkom sučelju OBD2.

3.7.4. Pokretanje *bash* skripte za početak snimanja

Za automatizirano snimanje implementirana je *bash* skripta koja uključuje zapisivanje CAN poruka i snimanje slika kamerom. Skripta se nalazi u prilogu P.3.3 i pokreće se naredbom:

```
source record.bash      # Direktno snimanje
source record.bash -s   # Podešavanje položaja kamere prije početka
↪  snimanja
```

Skripta nudi mogućnost pokretanja poslužitelja pomoću kojeg se pojednostavljuje montiranje kamere. Korištenjem parametra `-s` šalju se slike na drugi umreženi uređaj koji ima mogućnost grafičkog pregleda slika. Nakon završetka pregleda, skripta nastavlja s izvršavanjem. U nastavku kreira se varijabla za početak snimanja koja ujedno i predstavlja naziv snimke kako bi svako snimanje bilo jedinstveno. Zatim se pokreće zapisivanje CAN poruka u datoteku, zapisivanje slika u H264 datoteku i zapisivanje vremenskih razmaka

između svake slike u .txt datoteku. Zapisivanje se odvija u pozadini te se pritiskom tipke `q` zaustavlja zapisivanje i sve stvorene datoteke se premještaju u .tar datoteku zbog jednostavnijeg prijenosa i grupiranja datoteka. Rezultat snimanja je .tar datoteka, ostale datoteke se brišu.

3.8. Upute za pokretanje sinkronizacije trenutka uzimanje slike i trenutka očitavanja kuta zakreta

Sinkronizacija skupa podataka odvija se na računalu nakon snimanja. Zapakirane snimke potrebno je premjestiti na računalo i pratiti sljedeće korake kako bi se svakoj slici vremenski uparío zadani kut zakreta upravljača vozila.

3.8.1. Postavljanje virtualnog okruženja i instalacija potrebnih paketa

Preporuča se korištenje virtualnog okruženja kako bi se osiguralo što sličnije okruženje pri reprodukciji onoga što je bilo u razvoju. Potrebno je kreirati virtualno okruženje koristeći sljedeću naredbu:

```
python -m venv .venv
```

Nakon kreiranja potrebno je uključiti virtualno okruženje kako bi se koristio virtualni *Python* alat koji će sadržavati potrebne biblioteke. Virtualno okruženje aktivira se sljedećom naredbom:

```
./.venv/Scripts/activate    # Windows Powershell
source .venv/bin/activate   # Linux bash
```

U aktiviranom virtualnom okruženju potrebno je instalirati zadane biblioteke koje su zapisane u datoteci *requirements.txt* iz elektroničkog priloga P.3.5.

```
pip install -r requirements.txt
```

3.8.2. Instalacija *ffmpeg* alata za dekodiranje H264 datoteke snimanja

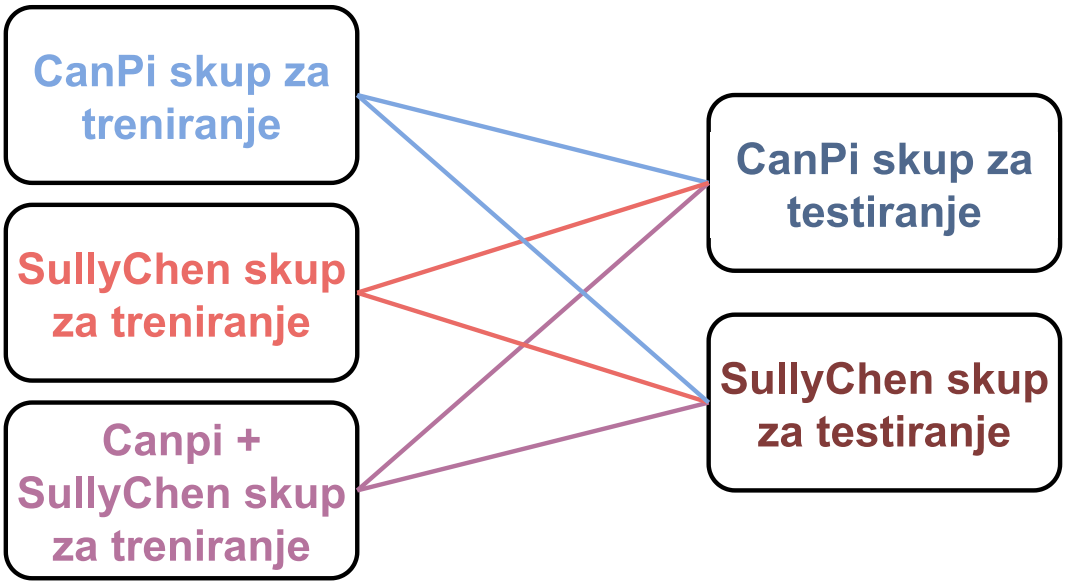
Za razdvajanje video zapisa u slike korišten je alat *ffmpeg*. Alat je dostupan na `https://ffmpeg.org/download.html`. Upute za instaliranje su također dostupne na web stranici alata.

3.8.3. Preuzimanje *Python* datoteke za kreiranje i sinkronizaciju skupa podataka

Čitav programski kod za kreiranje i sinkronizaciju nalazi se u datoteci *create_datasets.ipynb* danoj u elektroničkom prilogu P.3.4. *Python* programski kod pisan je u *Jupyter Notebook* formatu koji omogućava pisanje programskog koda i dokumentiranja u *Markdown* formatu. Dijelovi programskog koda nalaze se u zasebnim ćelijama koje se sekvencijalno izvršavaju i olakšavaju otkrivanje pogrešaka. Detaljna uputstva za pokretanje nalaze se u datoteci programskog koda. Snimljene snimke mogu se preuzeti s adrese: `https://dl.rt-rk.com/canpids/`.

4. VALIDACIJA NOVONASTALOG SKUPA PODATAKA

Nakon prikupljanja podataka, potrebno je bilo ispitati korištenje tog skupa podataka na jednom od postojećih rješenja za procjenu kuta zakreta upravljača vozila. Za postojeće rješenje odabran je PilotNet[4] model konvolucijske neuronske mreže koji će učiti prvo na CanPi skupu podataka, zatim na SullyChen skupu podataka i na kraju na kombinaciji obaju skupova podataka. Nakon svakog učenja evaluirati će se performanse modela na način da se usporede stvarne poznate vrijednosti s predviđenim vrijednostima koje daje model na testnom skupu podataka koji nije korišten za učenje. Takvim pristupom želi se provjeriti može li model naučen na jednom skupu podataka postići visoke performanse na novom (dotad neviđenom) skupu podataka te može li mu u tome pomoći učenje na dvama skupovima podataka odjednom. Slika 4.1 grafički prikazuje kakvi su sve postupci treniranja (učenja) provedeni. Na lijevoj strani slike su informacije o skupovima na kojima je rješenje trenirano, a na desnoj strani su informacije o skupovima na kojima je trenirano rješenje testirano. Može se zaključiti da je postojalo 6 različitih kombinacija "trening skup-testni skup".

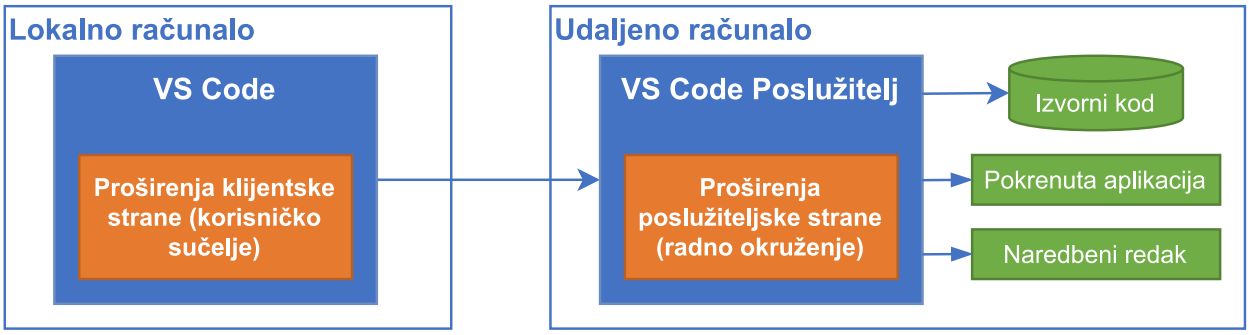


Slika 4.1: Postupak treniranja i testiranja na CanPi i SullyChen skupovima podataka - na lijevoj strani slike su informacije o skupovima na kojima je rješenje trenirano, a na desnoj strani su informacije o skupovima na kojima je trenirano rješenje testirano

4.1. Radno okruženje za treniranje modela autonomne vožnje zasnovane na procjeni kuta zakreta upravljača

Kako bi se ispitalo korištenje novonastalog skupa podataka na jednom od postojećih rješenja za procjenu kuta zakreta upravljača, odabran je model PilotNet[4]. Budući da je model autonomne vožnje koji će se koristiti u ovom poglavlju zasnovan na dubokoj neuronskoj mreži, potrebna je velika računalna moć za njegovo treniranje. Stoga je korišteno udaljeno računalo koje je opremljeno grafičkom karticom. Grafičke kartice zahvaljujući velikom brojem jezgri nude veće mogućnosti paralelnog računanja. Računalo koje je korišteno sadrži grafičku karticu Nvidia GeForce3080Ti, Intelov procesor 11. generacije i9-11900F, 32 GB radne memorije i SSD od 1 TB prostora za pohranu. Na njemu je instaliran operacijski sustav Ubuntu 20.04 i uključen SSH poslužitelj.

Za upravljanje računalom korišteno je *Remote* proširenje unutar Visual Studio Code (VS Code) programa. Proširenje omogućuje korisniku da se poveže na udaljeno računalo putem SSH protokola i pregledava datoteke kao da su na lokalnom računalu. Prilikom prvog pokretanja automatski se instalira VS Code poslužitelj, koji omogućuje manipulaciju datotekama i izvršavanja programskog koda. Tim načinom nema potrebe da poslužitelj vraća potpunu sliku kao kod tipičnih programa za udaljene radne površine, već samo datoteke koje se izvršavaju. Na slici 4.2 prikazana je arhitektura VS Remote razvojnog okruženja na udaljenom računalu. Uz *Remote* proširenje moguće je instalirati druga proširenja koja pomažu prilikom pisanja programskog koda.



Slika 4.2: Arhitektura Visual Studio Remote razvojnog okruženja

Na udaljenom računalu je zatim instaliran *Python3* i kreirano je virtualno okruženje. Virtualno okruženje omogućava instalaciju *pip* paketa koji nemaju izravne veze s globalno instaliranim paketima. Na taj način osigurava se neometan razvoj programske podrške u slučaju da drugi projekti koriste različite verzije paketa. Od paketa instalirani su *Tensorflow* i

Keras[26]. *Keras* je programsko okruženje koje pojednostavljuje kreiranje modela neuronskih mreža i učenja istih, a u pozadini za obavljanja računskih operacija koristi *Tensorflow*.

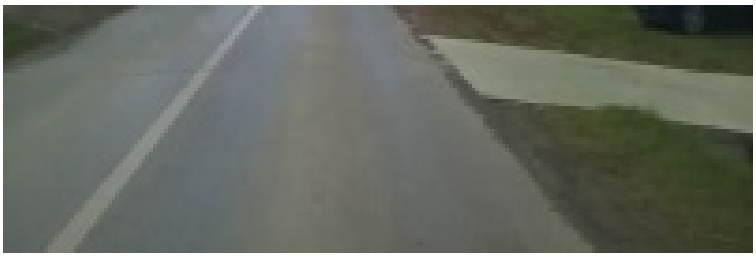
4.2. Priprema skupova podataka za treniranje neuronske mreže

Za model neuronske mreže odabran je prethodno naveden PilotNet. Ulaz u neuronsku mrežu na kojoj je model zasnovan je slika dimenzija $200 \times 66 \times 3$. Zbog definiranih dimenzija ulaznog podatka u PilotNet mrežu, potrebno je u slikama iz željenog skupa podataka izdvojiti regiju interesa i skalirati ju kako bi primila zadani oblik kojeg na ulazu očekuje neuronska mreža. Regija interesa predstavlja pravokutnik u kojem se nalazi prometnica ispred vozila. Dimenzija regije interesa su 1280×270 elemenata slike, a lokacija joj se mijenja ovisno o vidnom polju kamere. Potrebno je osigurati da se ne vidi početak vozila (na samom dnu slike) i da se regijom interesa obuhvati samo prometnica u blizini vozila. Zatim se slika regija interesa skalira na dimenzije ulaza neuronske mreže (dakle s rezolucije 1280×270 reskalira se na rezoluciju 200×66). Skaliranje je napravljeno interpolacijom dostupnom u biblioteci *OpenCV*[27]. Odabrana je ***INTER_AREA*** metoda interpolacije koja je najprikladnija za smanjenje rezolucije slike[28]. Na slici 4.3 nalazi se isječak koda za reskaliranje slike na rezoluciju 200×66 elemenata slike. Na slici 4.4 prikazana je regija interesa jedne slike iz CanPi skupa podataka.

```
cv2.resize(img, (200, 66), interpolation=cv2.INTER_AREA)
```

Slika 4.3: Isječak koda za reskaliranje regije interesa iz originalne slike na rezoluciju 200×66 elemenata slike

(a)

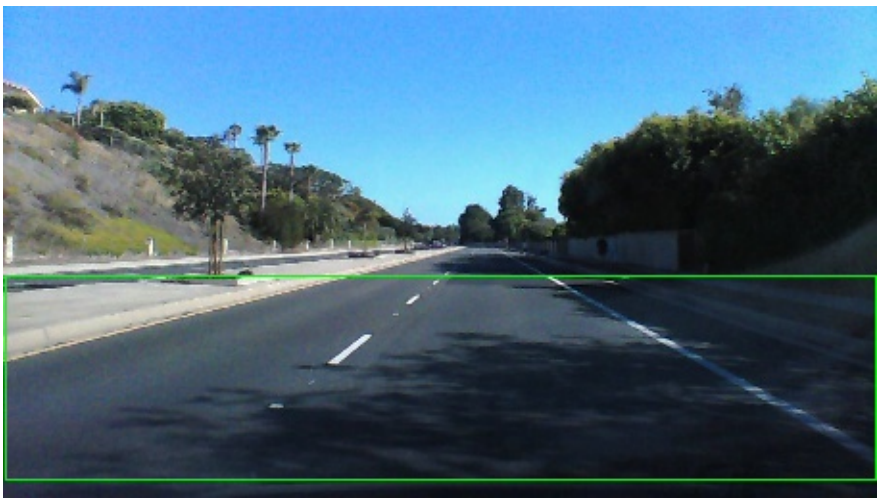


(b)

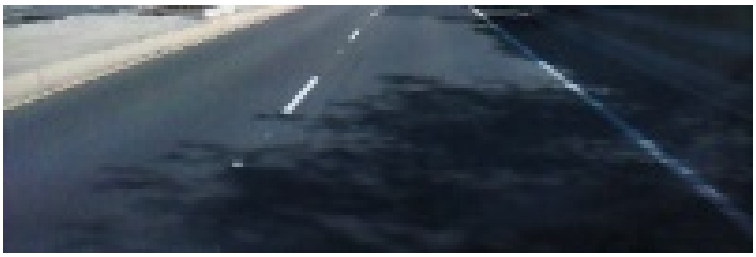
Slika 4.4: Regija interesa na slici iz CanPi skupa podataka (a) izvorna slika CanPi skupa podataka rezolucije 1280×720 elemenata slike i zelenim pravokutnikom označena regija interesa 1280×270. (b) regija interesa reskalirana na rezoluciju 200×66 elemenata slike

Čitav CanPi skup podataka koji se sastoji od 266933 slika i pripadnih kutova zakreta upravljača podijeljen je na tri manja skupa koji služe za treniranje, validaciju i testiranje modela i to u omjeru $0.7 : 0.1 : 0.2$. Na taj način u trening skupu nalazilo se 186853 slika, u validacijskom skupu 26693 slika, a u testnom skupu 53387 slika. Skupovi za treniranje, validaciju i testiranje pripremljeni su na način da su se iz svake od 17 snimki nasumično odabrale slike u zadanom omjeru. Na ovakav način svaki podskup sadrži slike iz svih snimki.

Na isti način izdvojena je regija interesa na slikama iz SullyChen skupa podataka. U slikama originalne rezolucije 455×256 elemenata slike potrebno je prvo bilo odrediti regiju interesa i svesti ju nakon toga na rezoluciju 200×66 elemenata slike. Iz slike je uklonjeno prvih 140 i zadnjih 10 redova elemenata slike, čime je dobivena regija interesa veličine 455×106. Dobivena regija interesa reskalirana je na rezoluciju 200×66 elemenata slike koristeći *INTER_AREA* metodu interpolacije iz *OpenCV* biblioteke. Na slici 4.5 prikazana je regija interesa na jednoj slici iz SullyChen skupa podataka.



(a)

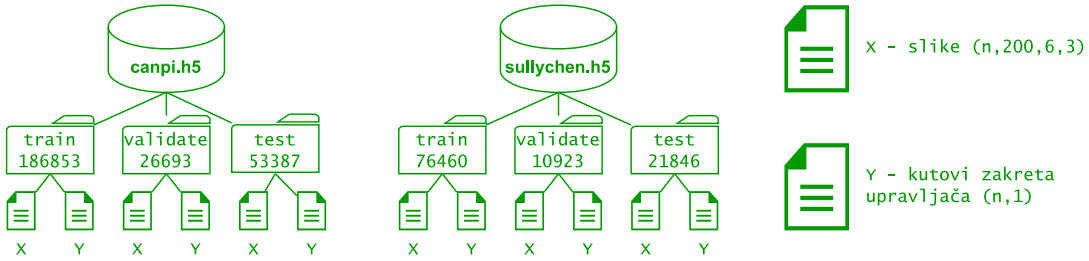


(b)

Slika 4.5: Regija interesa na slici iz SullyChen skupa podataka (a) izvorna slika SullyChen skupa podataka rezolucije 455×256 elemenata slike i zelenim pravokutnikom označena regija interesa 455×106. (b) regija interesa reskalirana na rezoluciju 200×66 elemenata slike

Potpuni SullyChen skup podataka koji se sastoji 109229 slika i pripadnih kutova zakreta upravljača podijeljen je na tri manja skupa koji služe za treniranje, validaciju i testiranje modela i to u omjeru $0.7 : 0.1 : 0.2$. Na taj način u trening skupu nalazilo se 76460 slika, u validacijskom skupu 10923 slika, a u testnom skupu 21846 slika. Skupovi za treniranje, validaciju i testiranje pripremljeni su na način da se iz svake snimke nasumično odabiru slike u zadanom omjeru. Na ovakav način svaki podskup sadrži slike iz obiju snimki.

Uobičajeno je da se skupovi za treniranje, validaciju i testiranje objedine u H5 datoteku. H5 format datoteke omogućuje sličnu hijerarhiju podataka kao kod direktorija i datoteka u upravitelju datotekama na operacijskim sustavima. Podaci se mogu spremiti kao skup podataka unutar datoteke ili unutar pojedine grupe koja može sadržavati razne parametre. Najveća prednost korištenja jedne datoteke kao mjesta pohrane skupa podataka je lakša prenosivost i jednostavnost implementacije čitanja datoteke u programskom kodu. Također, H5 datoteka nudi mogućnost kompresije pojedinih podataka s ciljem smanjenja prostora za pohranu. H5 datoteke završavaju s .h5 ekstenzijom. Mana je što se ne može koristiti pojedini dio skupa podataka bez da se preuzme čitav skup podataka. Rezultat grupiranja skupova za treniranje, validaciju i testiranje su dvije H5 datoteke (jedna za CanPi, jedna za SullyChen skup podataka), gdje svaka datoteka sadrži tri skupa podataka (*train, validate i test*). Organizacije H5 datoteka prikazane su na slici 4.6. Oznaka X predstavlja sliku regije interesa u boji rezolucije 200×66 elemenata slika, a Y predstavlja njezin pripadni kut zakreta upravljača. Datoteke su dostupne na upit.



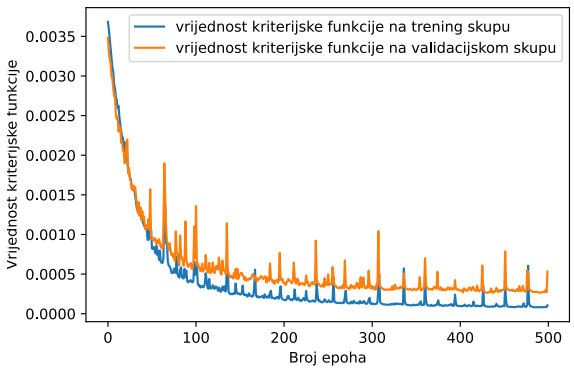
Slika 4.6: Organizacija H5 datoteka sa skupovima za treniranje, validaciju i testiranje za CanPi i SullyChen skupove podataka

4.3. Opis procesa treniranja modela na dvama različita skupa podataka

U prilogu P.4.1 nalazi se implementacija modela u *Keras* programskom okruženju. Za pojedinu iteraciju treninga odabrana je veličina 1024 (engl. *batch size*) zbog veće količine slika u skupu za treniranje. Za kriterijsku funkciju prilikom treniranja odabrana je srednja kvadratna pogreška (engl. *Mean squared error, MSE*), za optimizator koristi se *adam*[29] sa stopom učenja (engl. *learning rate*) od 0.0001. Model je učio na svakom pojedinom skupu podataka kroz 500 epoha te su nakon toga na odgovarajućem testnom skupu ispitane performanse najboljeg modela dobivenog na trening skupu.

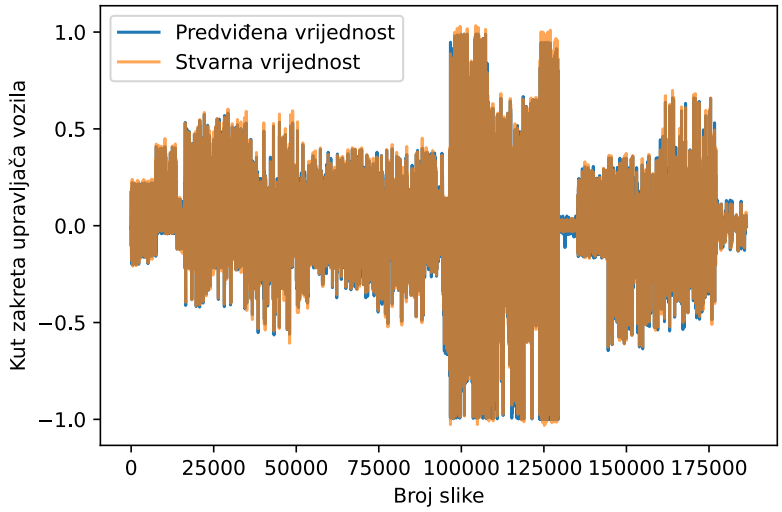
4.3.1. Treniranje PilotNet modela na CanPi trening skupu podataka

Model je treniran na 500 epoha s prethodno navedenim parametrima treniranja na CanPi trening i validacijskom skupu podataka. Na slici 4.7 prikazana je promjena kriterijske funkcije tokom treniranja PilotNet modela na CanPi trening i validacijskom skupu podataka.

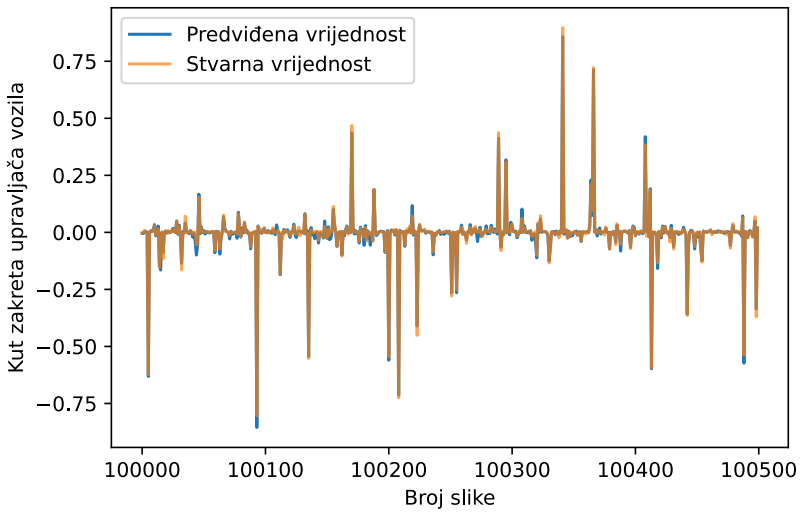


Slika 4.7: Promjena kriterijske funkcije na CanPi trening i validacijskom skupu podataka PilotNet modela treniranog na CanPi trening skupu podataka

Pad vrijednosti kriterijske funkcije ukazuje da je model naučio predviđati vrijednosti na temelju ulaznih i izlaznih vrijednosti iz CanPi trening skupa podataka. Kao najbolji model, odabrana je mreža iz 437. epohe, gdje je vrijednost kriterijske funkcije na validacijskom skupu podataka bila najniža. Na slikama 4.8 i 4.9 dani su grafovi na kojima postoji znatno podudaranje predviđenih sa stvarnim vrijednostima zakreta upravljača za podatke iz CanPi trening skupa. Na slici 4.10 dani su grafovi koji prikazuju rezultate modela treniranog i validiranog na odgovarajućim CanPi skupovima podataka, ali za CanPi testni skup.

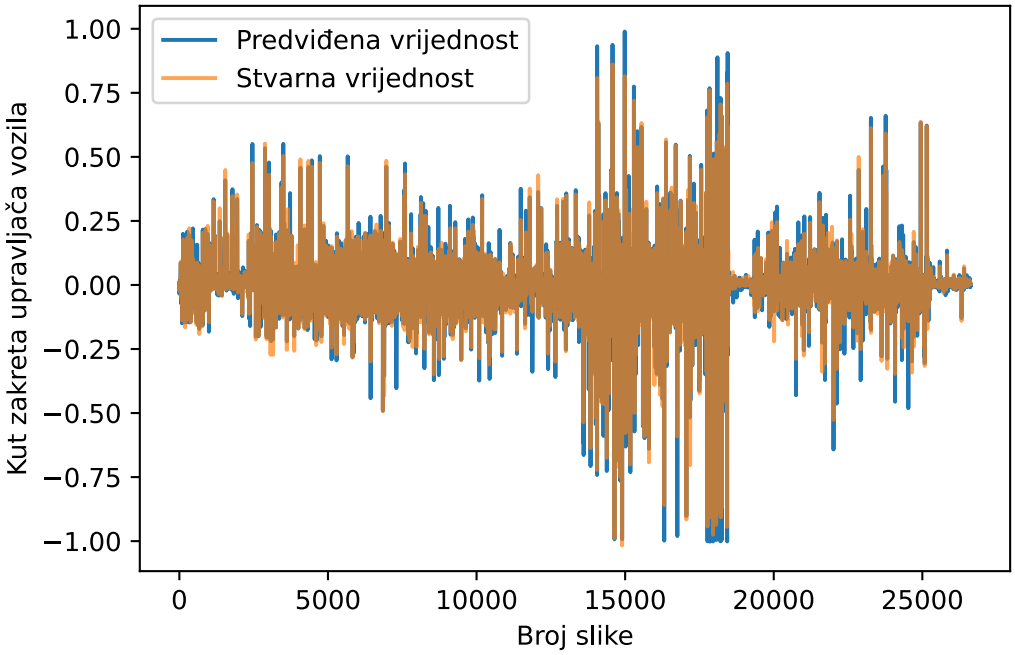


(a)

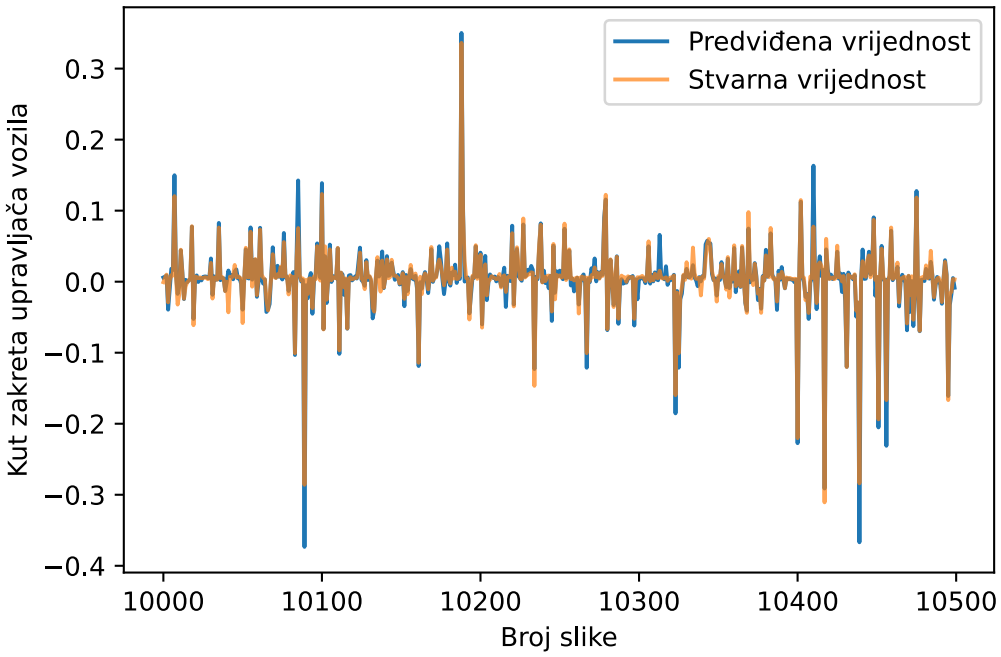


(b)

Slika 4.8: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi trening skupu podataka (a) cijeli CanPi trening skup podataka, (b) izdvojeni dio CanPi trening skupa podataka

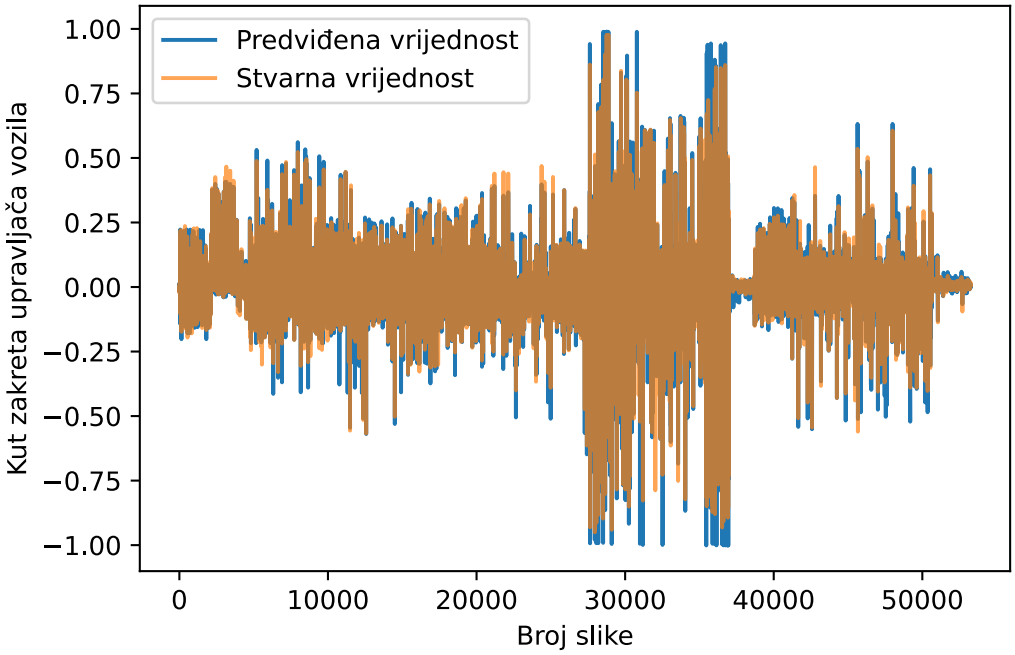


(a)

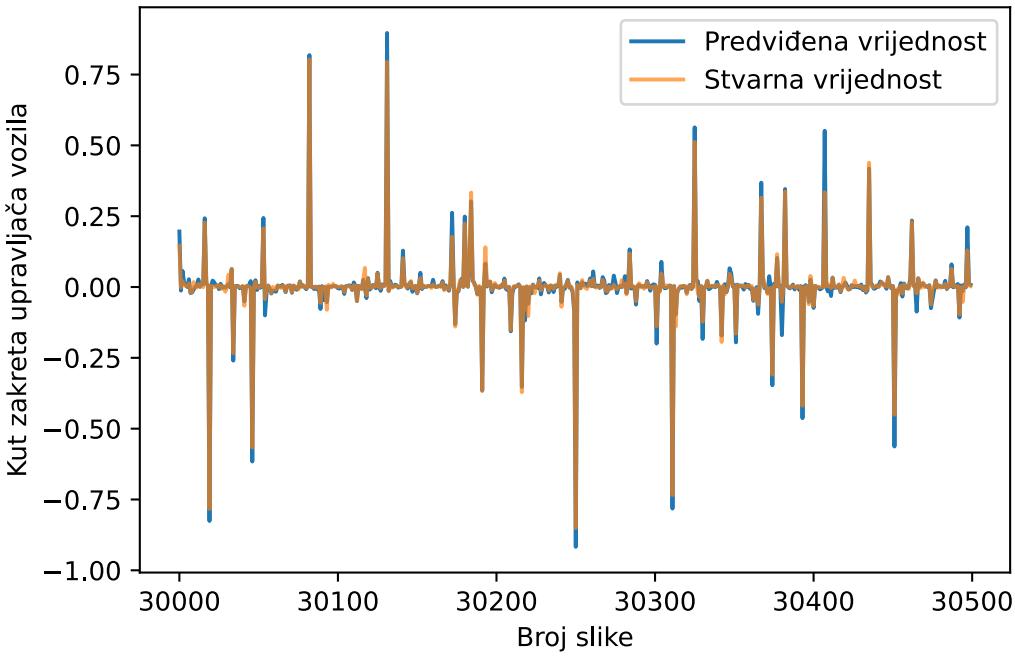


(b)

Slika 4.9: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi trening skupu podataka (a) cijeli CanPi validacijski skup podataka, (b) izdvojeni dio CanPi validacijskog skupa podataka



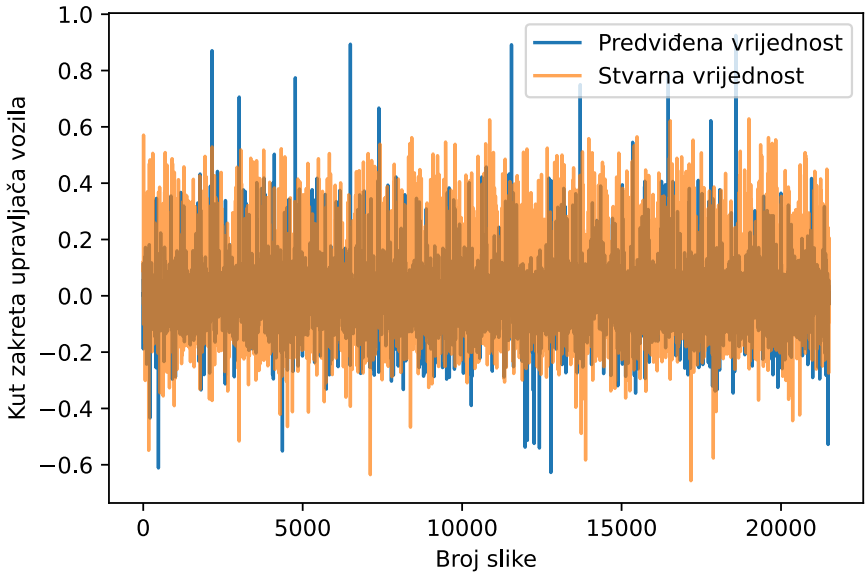
(a)



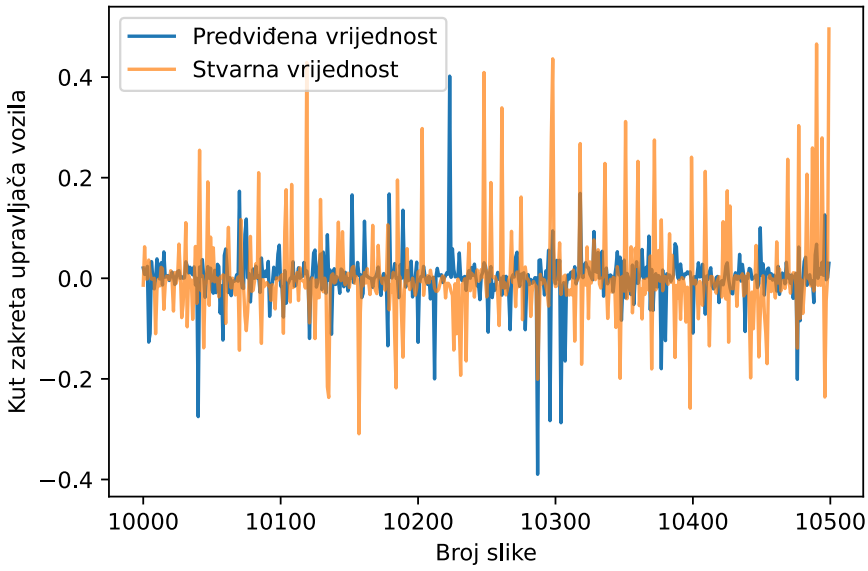
(b)

Slika 4.10: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi trening skupu podataka (a) cijeli CanPi testni skup podataka, (b) izdvojeni dio CanPi testnog skupa podataka

Nakon prikaza rezultata na CanPi skupovima podataka, slijedi prikaz rezultata modela treniranog na CanPi trening skupu podataka, ali testiranog na SullyChen testnom skupu podataka. Na slici 4.11 nalaze se stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi trening skupu podataka, testiranog na SullyChen testnom skupu podataka.



(a)



(b)

Slika 4.11: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi trening skupu podataka (a) cijeli SullyChen testni skup podataka, (b) izdvojeni dio SullyChen testnog skupa podataka

Zbog preglednije i preciznije evaluacije PilotNet modela, izračunate su MSE i srednja apsolutna pogreška (engl. *Mean absolute error, MAE*) između predviđenih i stvarnih vrijednosti kuta zakreta upravljača. Metrike se računaju prema sljedećim formulama.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4-2}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|, \tag{4-3}$$

Y_i predstavlja stvarnu vrijednost kuta zakreta upravljača, a $\hat{Y}_i$ predstavlja predviđenu vrijednost kuta zakreta upravljača za danu sliku iz korištenog skupa podataka. Parametar n predstavlja broj slika u korištenom skupu podataka. Metrike ukazuju na to koliko je dobro PilotNet model naučio predviđati vrijednosti na različitim skupovima podataka. U tablici 4.1 prikazani su dobiveni rezultati na CanPi trening, validacijskom i testnom skupu podataka te SullyChen testnom skupu podataka. Uz ukupni MSE i MAE izračunate su i medijalne vrijednosti tih veličina za svaki skup podataka zasebno.

Tablica 4.1: Prikaz rezultata PilotNet modela treniranog na CanPi skupu podataka

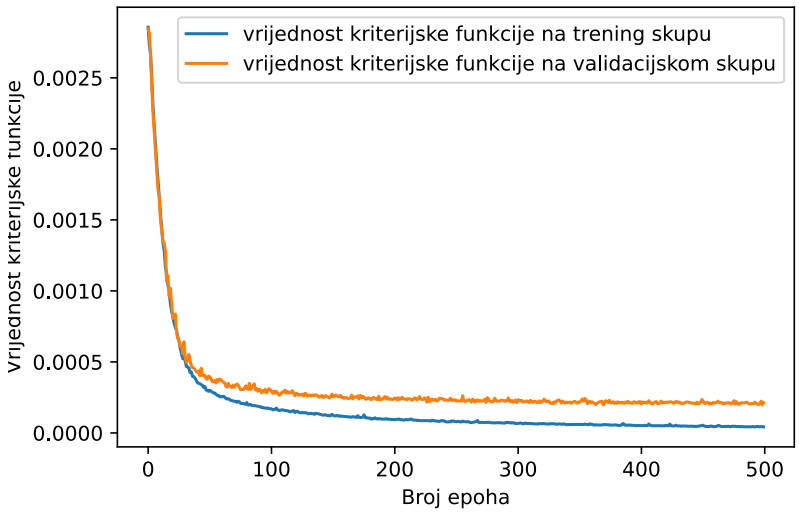
Skup podataka	**MSE**	**MAE**	**Medijan kvadratnih pogrešaka po okvirima**	**Medijan apsolutnih pogrešaka po okvirima**
CanPi - trening	0.00005746	0.00440564	0.00000664	0.00257626
CanPi - validacija	0.00026115	0.00588053	0.00000719	0.00268105
CanPi - test	0.00034988	0.00589679	0.00000699	0.00264368
SullyChen - test	0.01210961	0.06508090	0.00091386	0.03023009

Analizirajući rezultate samo na testnim skupovima, pomalo očekivano, bolje je rezultate model postigao na testnom skupu CanPi skupa podataka, budući da je i učio na trening skupu CanPi skupa podataka. Najniže vrijednosti metrika pokazale su se na CanPi trening skupu podataka, dok su nešto veće na CanPi validacijskom i testnom skupu podataka jer oni služe za validaciju i testiranje modela. Medijani kvadratnih i apsolutnih pogrešaka vrlo su slični brojčano uz mala odstupanja jer su skupovi izdvojeni iz istog CanPi skupa podataka. Srednja vrijednost (prosjek) skupa podataka dobiva se zbrajanjem svih brojeva u skupu podataka i zatim dijeljenjem s brojem vrijednosti u skupu. Medijan je vrijednost

sortiranog skupa od koje je pola elemenata skupa manje, a pola veće. Može se reći da je model dobro naučio ako predviđa kut zakreta upravljača unutar 5°, kao što je navedeno u radu[30]. U obama skupa podataka kut zakreta je skaliran na raspon [-1, 1], što znači da 5° odgovara vrijednosti 0.0092. Prema zadanom pragu tolerancije kuta zakreta upravljača, može se zaključiti da je model dobro naučio ako je srednja apsolutna pogreška manja od 0.0092. Model je naučio predviđati vrijednosti zakreta upravljača na CanPi testnom skupu, ali nije naučio na SullyChen testnom skupu podataka. Veća je razlika u medijanu i srednjoj vrijednosti kvadratnih odstupanja nego kod apsolutnih odstupanja. To ukazuje na svojstvo MSE metrike gdje je svako odstupanje kvadrirano te postoji veći raspon (najmanje i najveće vrijednosti) kvadratnih odstupanja nego kod apsolutnih odstupanja. Zbog većeg raspona dolazi do veće razlike medijana i srednje vrijednosti.

4.3.2. Treniranje PilotNet modela na SullyChen trening skupu podataka

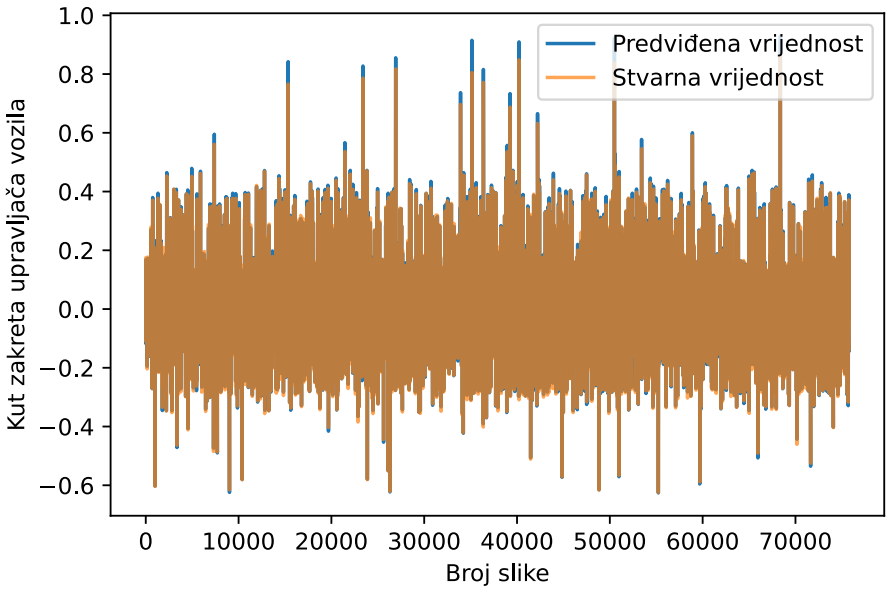
Zatim je napravljeno treniranje istog modela na SullyChen trening skupu podataka. Treniranje je započelo s identičnim parametrima kao s CanPi skupom podataka. Na slici 4.12 prikazana je promjena kriterijske funkcije tokom treniranja PilotNet modela kroz 500 epoha na SullyChen trening i validacijskom skupu podataka.



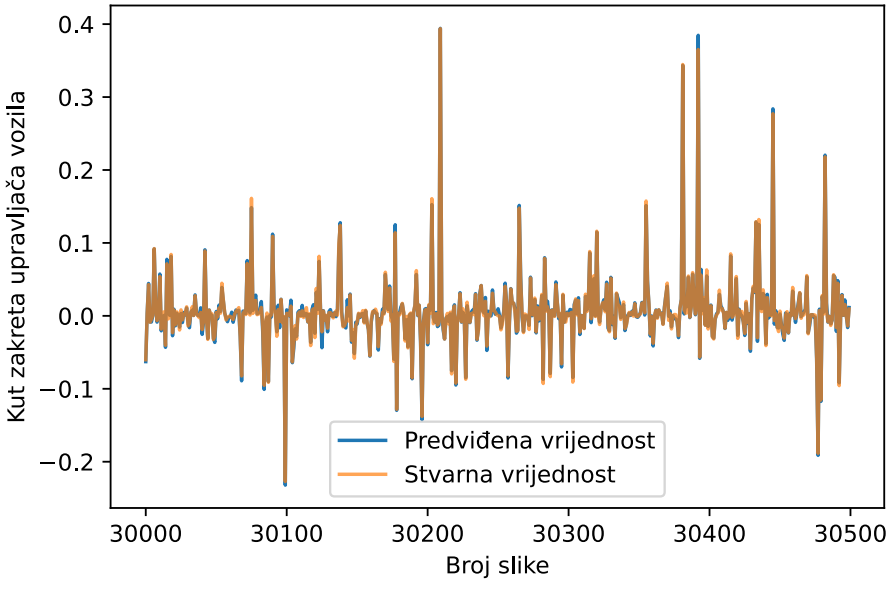
Slika 4.12: Promjena kriterijske funkcije na SullyChen trening i validacijskom skupu podataka PilotNet modela treniranog na SullyChen trening skupu podataka

Kao najbolji model, odabrana je mreža iz 498. epohe gdje je vrijednost kriterijske funkcije na validacijskom skupu podataka bila najniža. Na slikama 4.13 i 4.14 dani su grafovi na

kojima postoji znatno podudaranje predviđenih sa stvarnim vrijednosti zakreta upravljača za podatke iz SullyChen trening skupa. Na slici 4.15 dani su grafovi koji prikazuju rezultate modela treniranog i validiranog na odgovarajućim SullyChen skupovima podataka, ali za SullyChen testni skup.

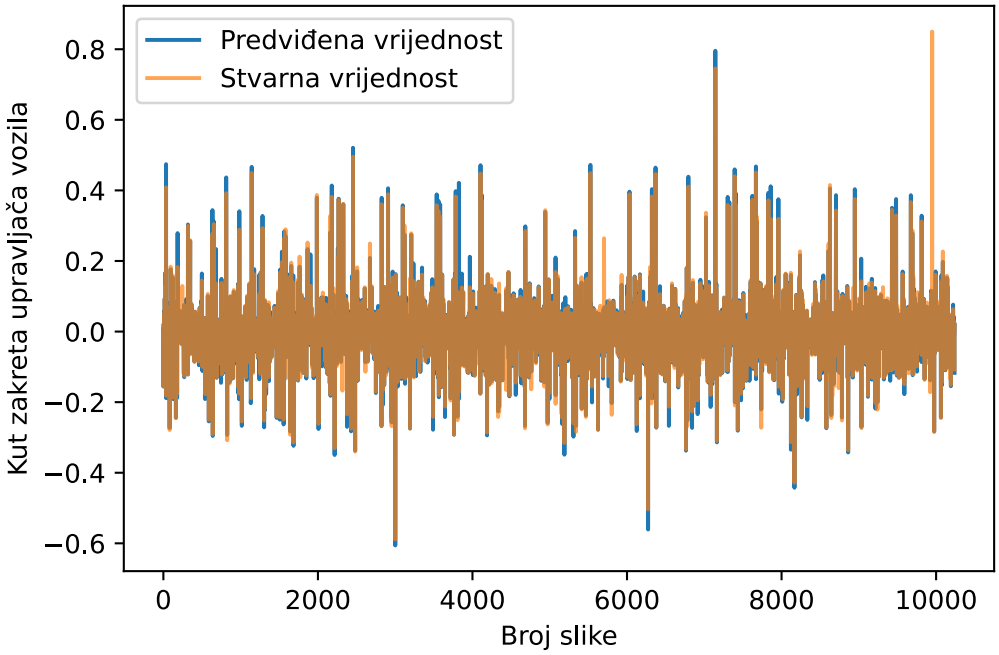


(a)

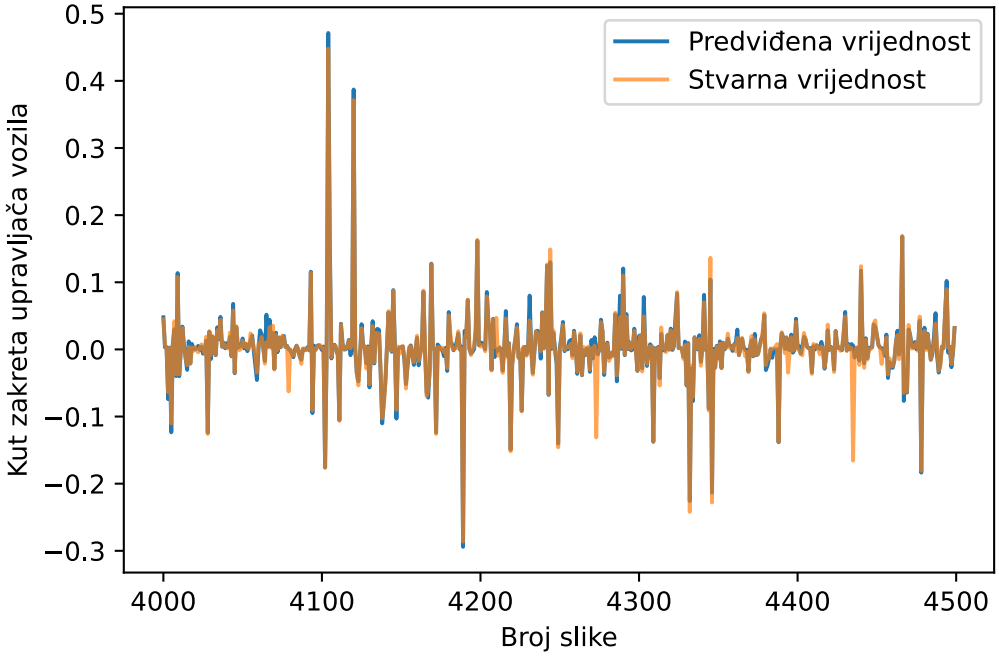


(b)

Slika 4.13: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na SullyChen trening skupu podataka (a) cijeli SullyChen trening skup podataka, (b) izdvojeni dio SullyChen trening skupa podataka

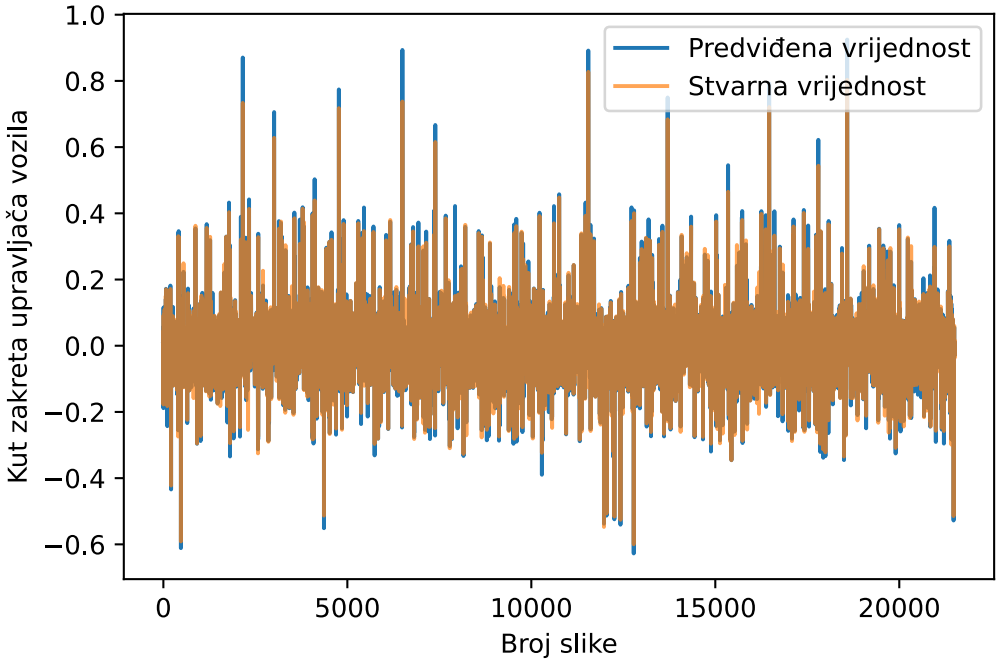


(a)

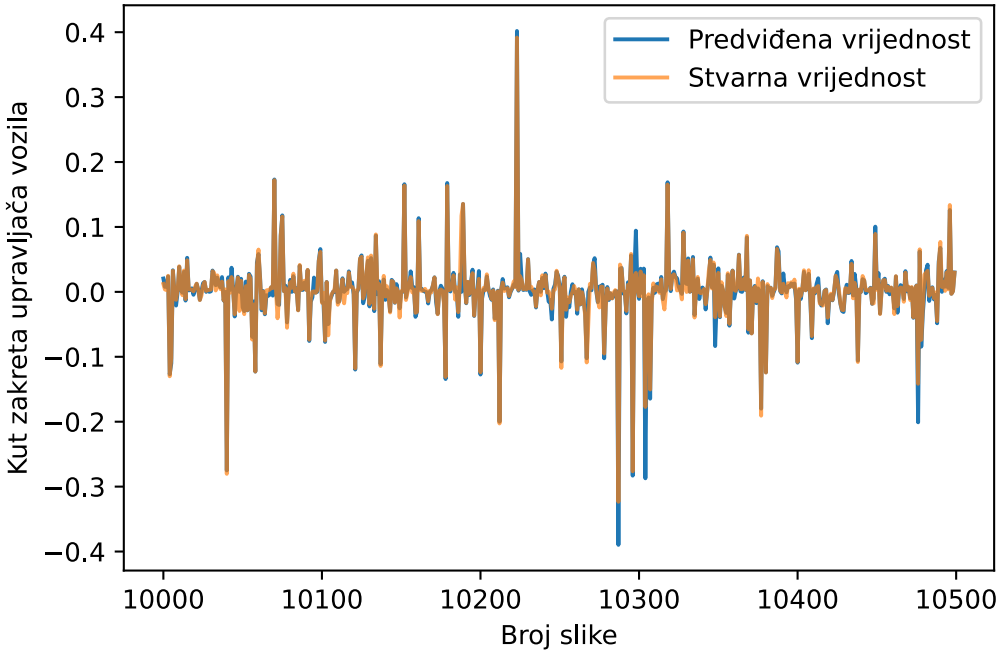


(b)

Slika 4.14: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na SullyChen trening skupu podataka (a) cijeli SullyChen validacijski skup podataka, (b) izdvojeni dio SullyChen validacijskog skupa podataka



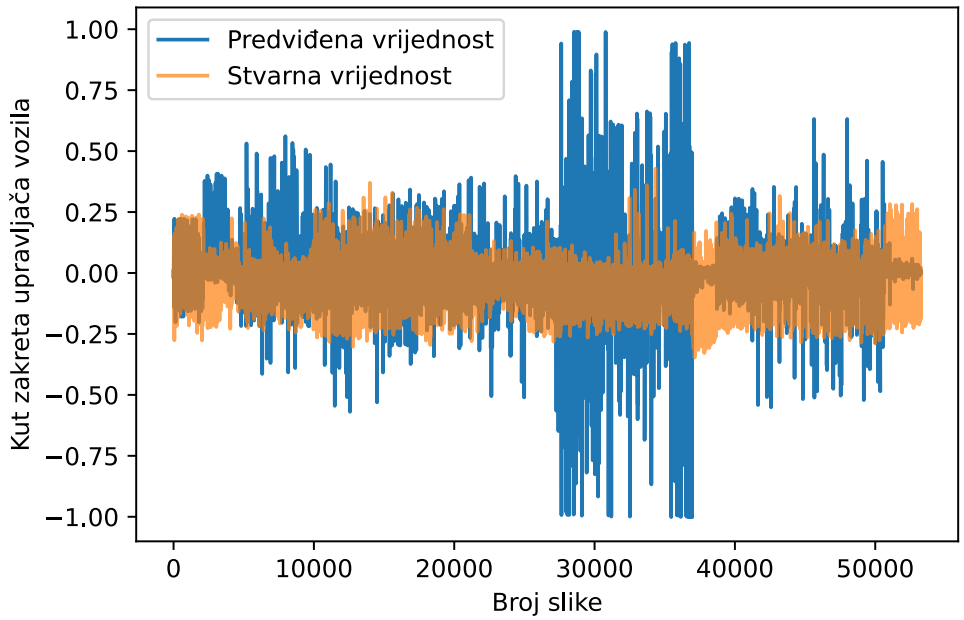
(a)



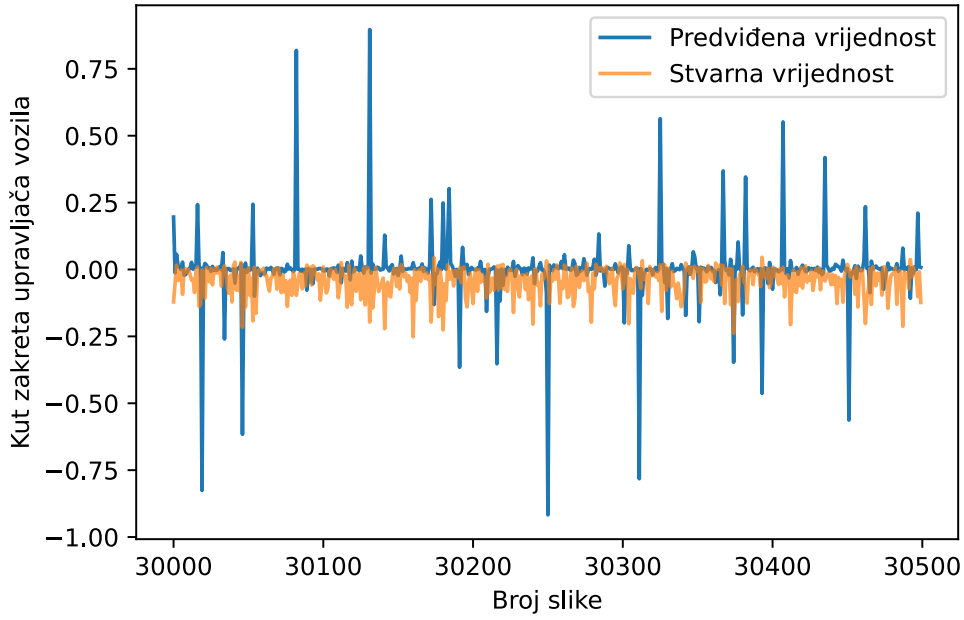
(b)

Slika 4.15: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na SullyChen trening skupu podataka (a) cijeli SullyChen testni skup podataka, (b) izdvojeni dio SullyChen testnog skupa podataka

Nakon prikaza rezultata na SullyChen skupovima podataka, na slici 4.16 nalazi se prikaz rezultata modela treniranog na SullyChen trening skupu podataka, ali testiranog na CanPi testnom skupu podataka.



(a)



(b)

Slika 4.16: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na SullyChen trening skupu podataka (a) cijeli CanPi testni skup podataka, (b) izdvojeni dio CanPi testnog skupa podataka

Kao i kod prethodnog modela treniranog na CanPi skupu podataka, izračunate su metrike modela treniranog na SullyChen skupu podataka. Metrike su prikazane u tablici 4.2.

Tablica 4.2: Prikaz rezultata PilotNet modela treniranog na SullyChen skupu podataka

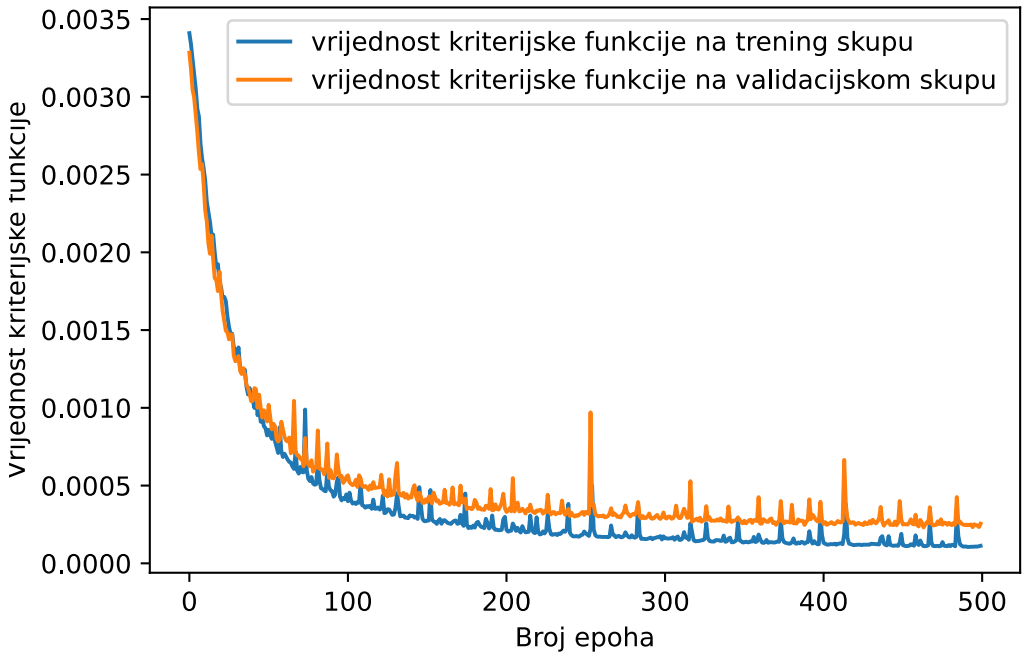
Skup podataka	**MSE**	**MAE**	**Medijan kvadratnih pogrešaka po okvirima**	**Medijan apsolutnih pogrešaka po okvirima**
SullyChen - trening	0.00002445	0.00316604	0.00000475	0.00217959
SullyChen - validacija	0.00019708	0.00497094	0.00000706	0.00265734
SullyChen - test	0.00012023	0.00486049	0.00000709	0.00266199
CanPi - test	0.01051212	0.06849131	0.00170310	0.04126859

Model treniran na SullyChen skupu podataka pokazuje bolje rezultate na SullyChen testnom skupu od modela treniranog na CanPi skupu podataka. MSE na SullyChen testnom skupu podataka za model treniran na SullyChen trening skupu je dvostruko niža nego za model treniran na CanPi trening skupu. Rezultati na testnim skupovima podataka brojčano su slični za oba modela. MSE na SullyChen testnom skupu modela treniranog na CanPi trening skupu iznosi 0.0121, a na CanPi testnom skupu modela treniranog na SullyChen skupu podataka iznosi 0.0105. Analizirajući MSE, može se zaključiti da su oba modela slično naučili predviđati kut zakreta na testnim skupovima iz nepoznatih skupova podataka. Razlika u modelima je što model naučen na CanPi trening skupu, a testiran na SullyChen testnom skupu predviđa manje vrijednosti nego što su poznate (stvarne), vidljivo na slici 4.11. Obrnuto predviđanje pojavljuje se kod modela naučenog na SullyChen trening skupu, a testiran na CanPi testnom skupu gdje su predviđene vrijednosti veće od stvarnih, vidljivo na slici 4.16. Iako na slikama izgleda da su veća odstupanja na slici 4.16, metrike ukazuju da su odstupanja vrlo slična kada se uzima apsolutno ili kvadratno odstupanje od poznate vrijednosti.

4.3.3. Treniranje PilotNet modela na CanPi i SullyChen trening skupovima podataka zajedno

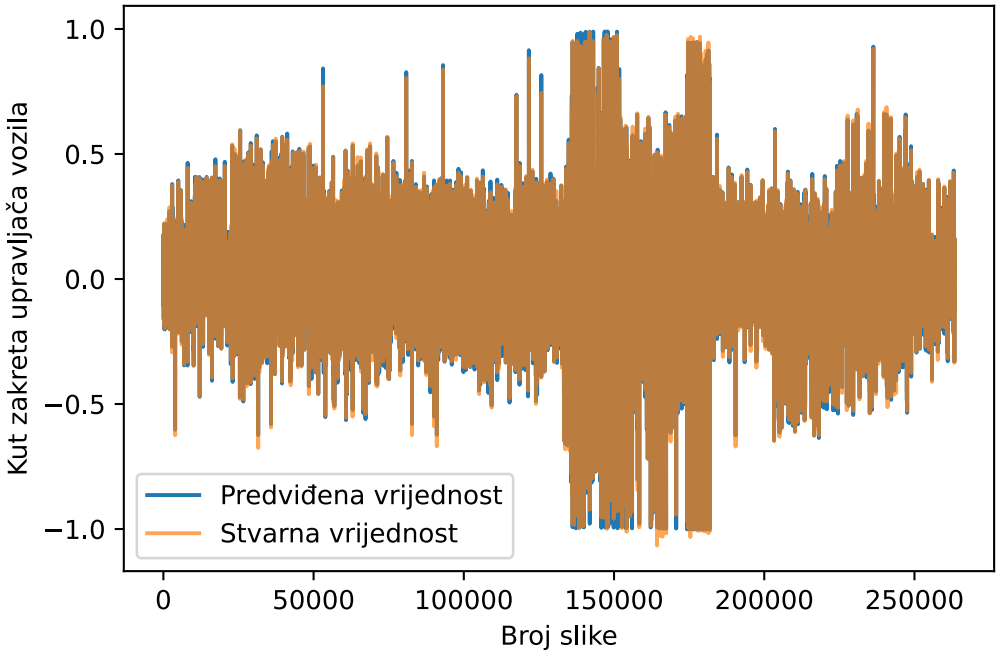
Posljednji postupak treniranja PilotNet modela proveden je na način da je treniran na obama trening skupovima podataka zajedno (CanPi i SullyChen). CanPi skup podataka sadrži više slika (186887:76461 slika u trening skupovima) nego SullyChen skup podataka

pa ih je trebalo smisleno podijeliti, kako bi svaki niz (engl. *batch*) u epohi obuhvatio slike iz obaju trening skupova podatka. Izmjerena je duljina svakog skupa podataka i izračunat omjer (2.44:1) duljina skupova. Zatim je taj omjer primijenjen i pri kreiranju svakog niza prilikom treniranja, što je rezultiralo da svaki niz duljine 1024 u epohi sadrži 727 slika iz CanPi trening skupa podataka i 297 slika iz SullyChen skupa podataka. Isti omjer je preslikan na zajednički CanPi i SullyChen validacijski skup podataka. Na slici 4.17 prikazana je promjena kriterijske funkcije tokom treniranja PilotNet modela kroz 500 epoha na CanPi i SullyChen trening i validacijskim skupovima podataka zajedno.

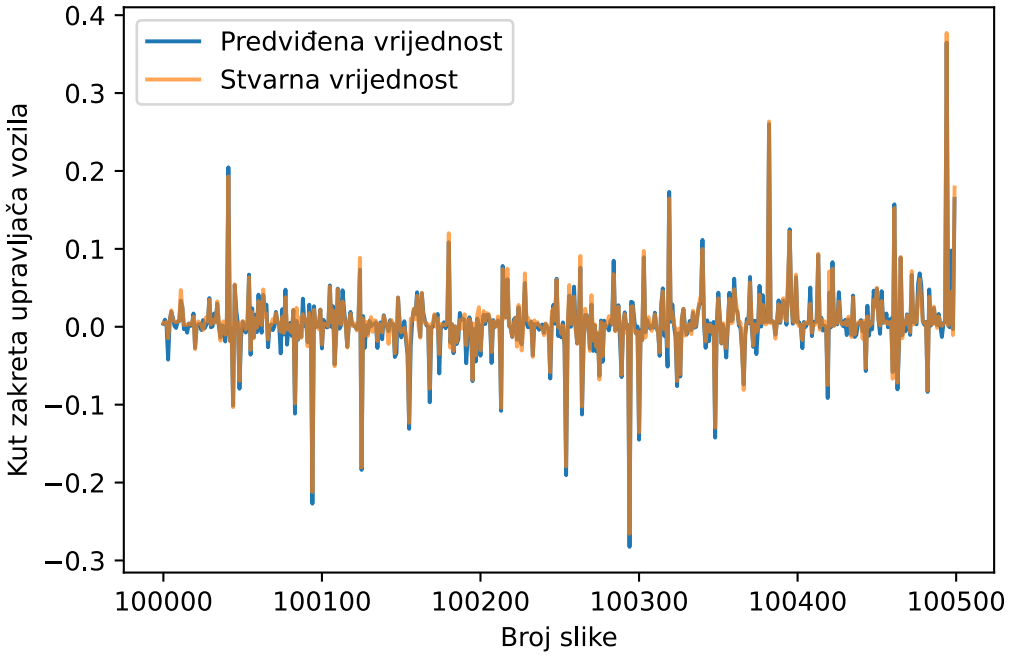


Slika 4.17: Promjena kriterijske funkcije na CanPi+SullyChen trening i validacijskom skupu PilotNet modela treniranog na CanPi+SullyChen trening skupu podataka

Kao najbolji model, odabrana je mreža iz 463. epohe gdje je vrijednost kriterijske funkcije na validacijskim skupovima podataka bila najniža. Na slikama 4.18 i 4.19 dani su grafovi na kojima postoji znatno podudaranje predviđenih sa stvarnim vrijednosti zakreta upravljača iz trening i validacijskog skupa podataka.

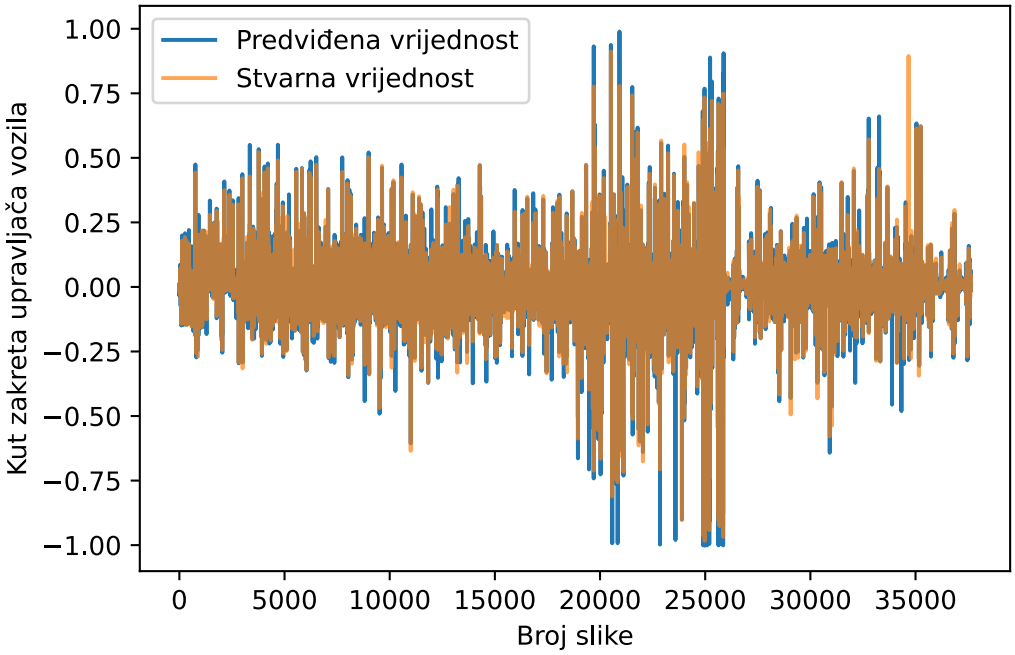


(a)

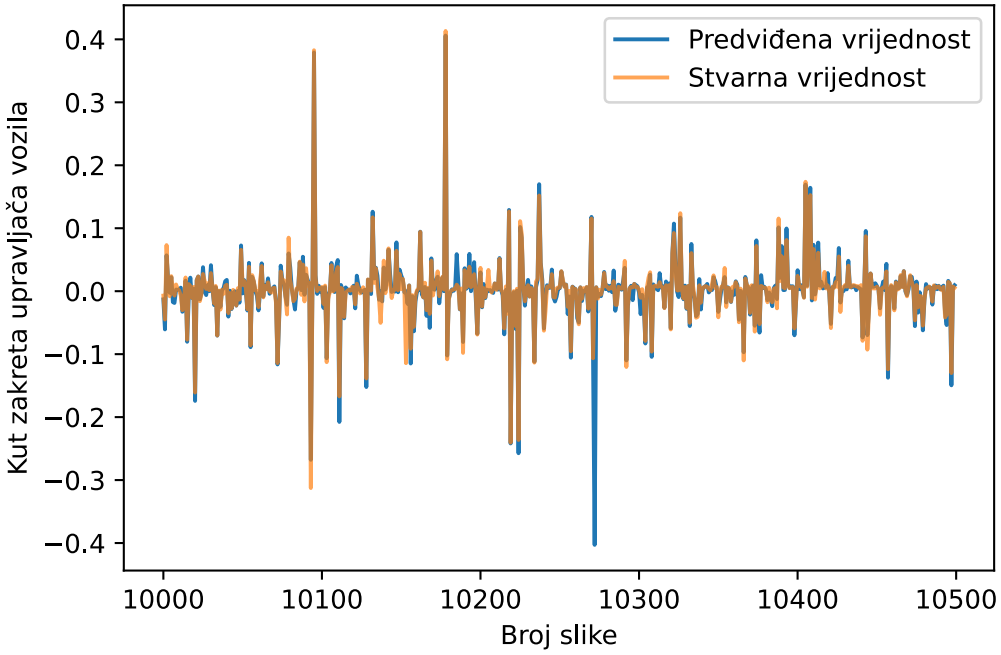


(b)

Slika 4.18: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi i SullyChen trening skupovima podataka (a) cijeli CanPi+SullyChen trening skup podataka, (b) izdvojeni dio CanPi+SullyChen trening skupa podataka



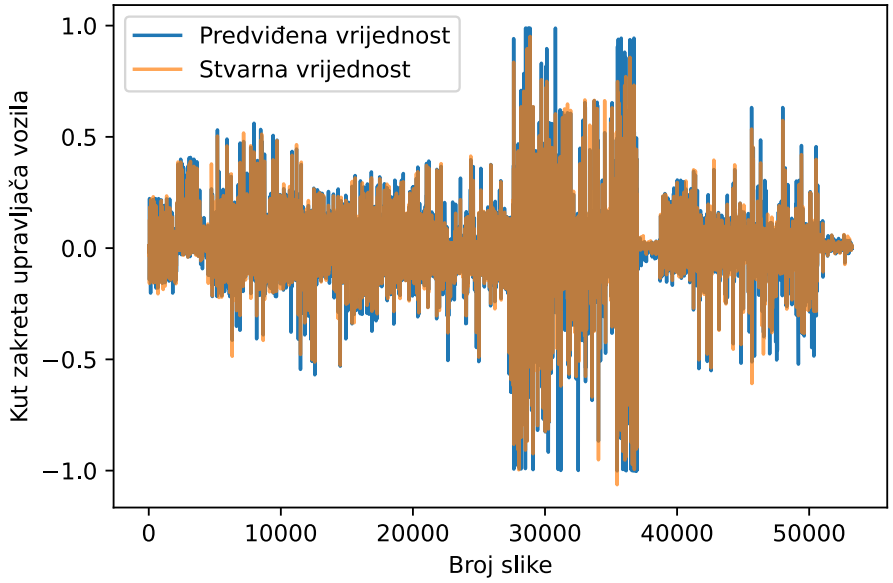
(a)



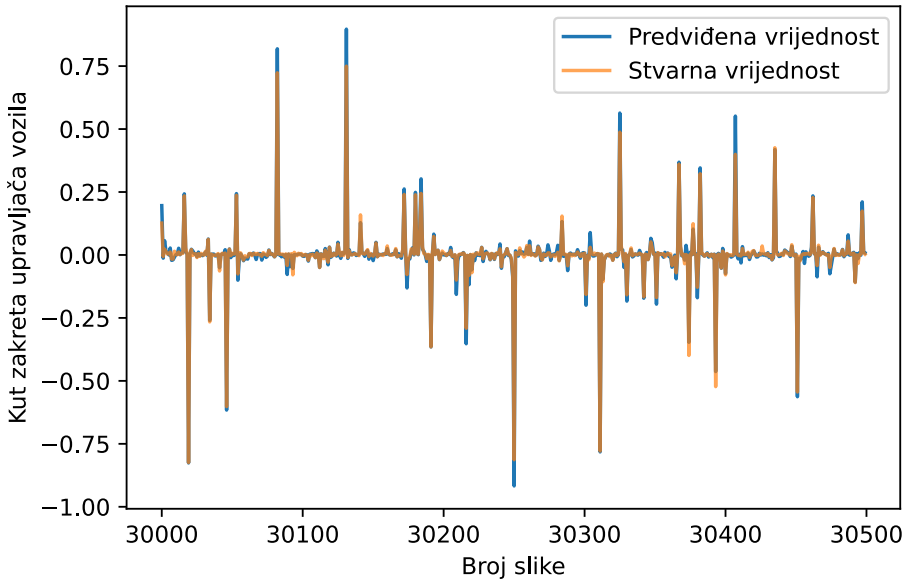
(b)

Slika 4.19: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi i SullyChen trening skupovima podataka (a) cijeli CanPi+SullyChen validacijski skup podataka, (b) izdvojeni dio CanPi+SullyChen validacijskog skupa podataka

U nastavku dani su grafovi koji prikazuju rezultate modela treniranog i validiranog na odgovarajućim CanPi i SullyChen skupovima podataka zajedno, testiranog na CanPi testnom skupu prikazanog na slici 4.20 i testiranog na SullyChen testnom skupu prikazanog na slici 4.21.

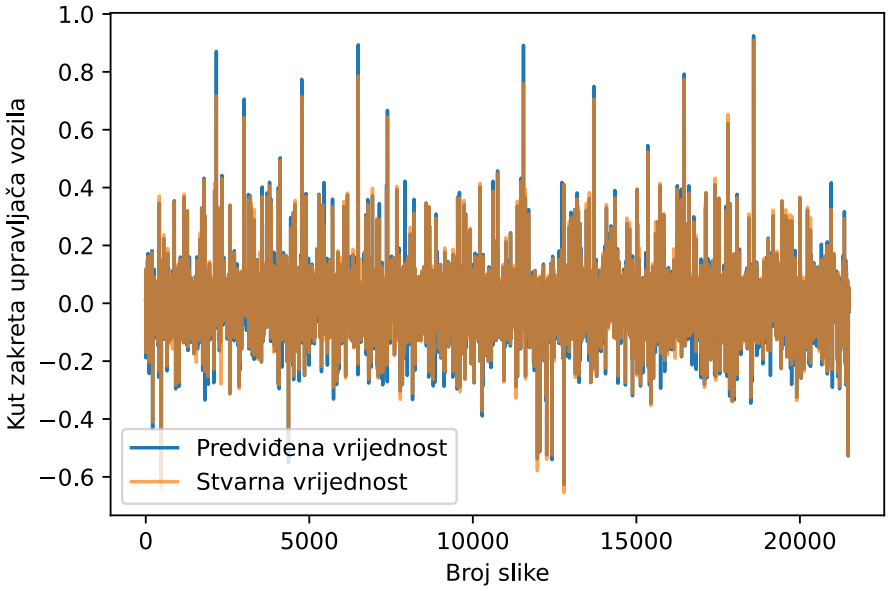


(a)

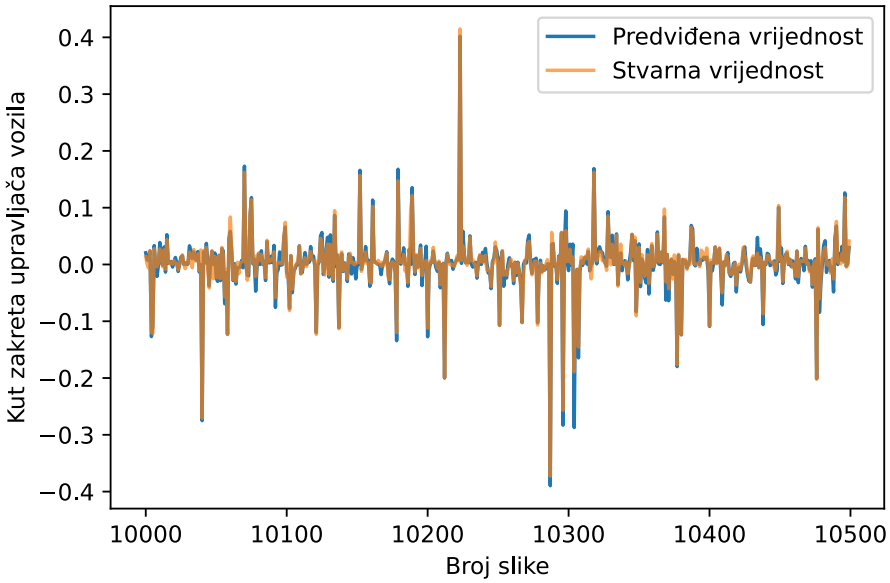


(b)

Slika 4.20: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi i SullyChen trening skupovima podataka (a) cijeli CanPi testni skup podataka, (b) izdvojeni dio CanPi testnog skupa podataka



(a)



(b)

Slika 4.21: Stvarne vrijednosti i predviđene vrijednosti kuta zakreta upravljača PilotNet modela treniranog na CanPi i SullyChen trening skupovima podataka (a) cijeli SullyChen testni skup podataka, (b) izdvojeni dio SullyChen testnog skupa podataka

Treniranje modela na obama skupovima podataka zajedno napravljeno je kako bi se ispitalo može li model treniran na dvama različitih skupova zajedno polučiti bolje rezultate na određenom testnom skupu (npr. testni SullyChen) u odnosu na model treniran na trening skupu samo jednog skupa podataka. Za lakšu usporedbu izračunate su prethodno spomenute

metrike na trening i validacijskim skupovima podataka, kao i na testnim skupovima iz obaju skupova podataka. Metrike se nalaze u tablici 4.3.

Tablica 4.3: Prikaz rezultata PilotNet modela treniranog na CanPi i SullyChen skupovima podataka

Skup podataka	**MSE**	**MAE**	**Medijan kvadratnih pogrešaka po okvirima**	**Medijan apsolutnih pogrešaka po okvirima**
CanPi+SullyChen - trening	0.00006626	0.00494712	0.00000850	0.00291570
CanPi+SullyChen - validacija	0.00023117	0.00618658	0.00000910	0.00301687
CanPi - test	0.00028086	0.00591990	0.00000729	0.00269981
SullyChen - test	0.00017560	0.00696630	0.00001625	0.00403145

MSE metrika na testnim skupovima podataka prikazuje da je model bolje predvidio kutove zakreta upravljača na SullyChen testnom skupu nego na CanPi testnom skupu podataka. MAE pokazuje obratno, manje su greške na CanPi testnom skupu. To ukazuje da na CanPi testnom skupu postoje rubna (veća) odstupanja, iako su većinom manja odstupanja što ukazuje mali medijan kvadratnih pogrešaka.

Korištenje kombinacije obaju trening skupova zajedno u procesu treninga modela dovelo je do poboljšanja performansi modela na testnim skupovima. Model treniran na obama skupovima podataka zajedno postigao je manju MSE na CanPi testnom skupu podataka nego model treniran samo na CanPi skupu podataka. Za SullyChen testni skup, MSE modela treniranog na obama skupovima podataka je malo viši nego kod modela treniranog samo na SullyChen trening skupu zbog većeg broja slika iz CanPi skupa podataka u odnosu na SullyChen skup. Promjena metrika apsolutnih odstupanja prati trend promjene kvadratnih odstupanja. Pokazalo se da treniranjem na obama skupovima podataka mogu se postići bolji rezultati i dobiti jedan model koji može predviđati rezultate na različitim skupovima podataka.

5. ZAKLJUČAK

Cilj ovog diplomskog rada bio je osmisliti sustav za prikupljanje skupa podataka sa stvarnim scenama iz prometa za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila. Skup podataka sastoji se od slika koje snimaju prostor ispred vozila s vremenski uparenima kutovima zakreta upravljača. Sustav za prikupljanje skupa podataka predložen u radu zasniva se na fotografiranju prostora ispred vozila koristeći Raspberry Pi kameru i prikupljanje podataka o kutu zakreta upravljača iz dijagnostičkog sučelja vozila koristeći CAN sabirnicu vozila. Raspberry Pi mikro-računalo povezuje se s kamerom montiranom na prednjem vjetrobranskom staklu. Također, Raspberry Pi mikro-računalo povezuje se putem OBD2 sučelja na CAN sabirnicu vozila putem koje uređaji poput senzora kuta zakreta upravljača komuniciraju unutar vozila. Ukupno je prikupljeno 266933 slika rezolucije 1280×720 elemenata slike s vremenski uparenim vrijednostima kuta zakreta upravljača. Skup podataka podijeljen je u 17 video snimki, gdje svaka snimka sadrži informacije o ruti na kojoj je snimljena, vremenskim uvjetima (sunčano, kišovito, oblačno), dobu dana (dan, noć) i tipu područja (urbano, ruralno) pri kojima je snimka nastala.

Načinjena je i validacija performansi suvremenog PilotNet[4] modela za upravljanje kutom zakreta upravljača vozila koristeći novo-kreirani i jedan dodatni postojeći skup podataka. Za usporedbu odabran je SullyChen[6] skup podataka jer sadrži slične uvjete kao i novo-nastali CanPi skup podataka. Istreniran je PilotNet model na CanPi skupu podataka, zatim na SullyChen skupu podataka i na kraju na obama skupovima podataka zajedno, kako bi se ispitalo može li model naučen na jednom skupu podataka uspješno predviđati na drugom skupu podataka. Također, željelo se ispitati kako treniranje na dvama skupovima podataka zajedno utječe na performanse modela kada se on testira na svakom od tih skupova podataka zasebno.

Pokazalo se da model treniran na CanPi trening skupu podataka daje odlične rezultate na CanPi testnom skupu podataka, čime se dokazalo da model može naučiti na novo-nastalom skupu podataka. Testiranjem prethodnog modela na SullyChen testnom skupu podataka pokazalo se da model neuspješno predviđa kutove zakreta upravljača jer je odstupanje veće od 5° od stvarne vrijednosti. Zatim je istreniran model na SullyChen trening skupu podataka, gdje je model dao dobre rezultate na SullyChen testnom skupu, ali loše na CanPi testnom skupu podataka. Na kraju, model je treniran na CanPi i SullyChen trening skupovima

podataka zajedno, gdje su se pokazali odlični rezultati kod testiranja na CanPi i SullyChen testnim skupovima podataka. Srednja kvadratna pogreška na CanPi testnom skupu niža je nego kod modela treniranog na CanPi trening skupu samostalno. Na SullyChen testnom skupu podataka, MSE kod modela treniranog na oba skupa je ponešto viši nego kod modela treniranog na SullyChen trening skupu podataka. Glavni faktor za rast MSE kod SullyChen skupa podataka je što CanPi skup podataka sadrži gotovo 2.5 puta više slika nego SullyChen skup podataka. Treniranje modela na obama skupovima podataka zajedno pokazalo je odlične rezultate, gdje je model bolje naučio predviđati kut zakreta upravljača iz testnog CanPi skupa podataka.

Dodatna umjetna oblikovanja slika (engl. *augmentation*) i izbacivanja slika s malim kutom zakreta upravljača potencijalno bi mogla doprinijeti boljim rezultatima prilikom treniranja modela. Također, proširivanje skupa podataka s više slika i različitih uvjeta snimanja potencijalno bi mogla dovesti do poboljšanja performansi modela. Skup podataka trenutno sadrži samo slike i kut zakreta upravljača, no prilikom snimanja spremljene su sve informacije iz dijagnostičkog sučelja kao što su brzina, broj okretaja motora i slično, koje je potrebno dekodirati kako bi se koristile za razvoj nekih dodatnih algoritama autonomne vožnje.

LITERATURA

[1] "Intelligent speed assistance (isa) set to become mandatory across europe," European Road Safety Charter, 7 2022. , s Interneta, https://road-safety-charter.ec.europa.eu/resources-knowledge/media-and-press/intelligent-speed-assistance-isa-set-become-mandatory-across_en

[2] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proceedings of (NeurIPS) Neural Information Processing Systems*, D. Touretzky, Ed. Morgan Kaufmann, December 1989, pp. 305 – 313.

[3] Net-Scale Technologies, Inc., "Autonomous off-road vehicle control using end-to-end learning - final technical report," 2004. , s Interneta, http://net-scale.com/doc/net-scale-dave-report.pdf

[4] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," 2017. , s Interneta, https://arxiv.org/abs/1704.07911

[5] M. Bojarski, C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. Jackel, Z. Jia, B. Lee, B. Liu, F. Liu, U. Muller, S. Payne, N. K. N. Prasad, A. Provodin, J. Roach, T. Rvachov, N. Tadimeti, J. van Engelen, H. Wen, E. Yang, and Z. Yang, "The nvidia pilotnet experiments," 2020. , s Interneta, https://arxiv.org/abs/2010.08776

[6] S. Chen, "driving-datasets," 2018. , s Interneta, https://github.com/SullyChen/driving-datasets

[7] S. Chen, "How a high school junior made a self-driving car," 12 2018. , s Interneta, https://towardsdatascience.com/how-a-high-school-junior-made-a-self-driving-car-705fa9b6e860

[8] H. Schafer, E. Santana, A. Haden, and R. Biasini, "A commute in data: The comma2k19 dataset," 2018.

[9] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd17: End-to-end davis driving dataset," 2017. , s Interneta, https://arxiv.org/abs/1711.01458

[10] Y. Hu, J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd20 end-to-end event camera driving dataset: Fusing frames and events with deep learning for improved steering prediction," 2020. , s Interneta, https://arxiv.org/abs/2005.08605

[11] T. Delbruck, "Ddd20 ford focus and mondeo davis driving dataset file descriptions." , s Interneta, https://docs.google.com/spreadsheets/d/18B9YtY_AGHkspHCmeNkMT1X_jkSYbNa-Es6Sz8VuCCg/edit?usp=sharing/

[12] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2d2: Audi autonomous driving dataset," 2020. , s Interneta, https://arxiv.org/abs/2004.06320

[13] *Raspberry Pi 4 Model B Product Brief*, Raspberry Pi Trading Ltd, 1 2021. , s Interneta, https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf

[14] T. Maddox and P. Maddox, "Juicessh - free ssh client for android," 2013. , s Interneta, https://juicessh.com

[15] *libcamera*, The libcamera documentation authors, 2020. , s Interneta, https://libcamera.org/getting-started.html

[16] "Video developer report 2018 (pdf)," Bitmovin, 9 2019. , s Interneta, https://go.bitmovin.com/hubfs/Bitmovin-Video-Developer-Report-2018.pdf

[17] "What is h.264 video encoding?" Blackbox. , s Interneta, https://www.blackbox.co.uk/gb-gb/page/38313/Resources/Technical-Resources/Black-Box-Explains/Multimedia/What-is-H264-video-encoding

[18] "ffmpeg documentation," The FFmpeg developers. , s Interneta, https://ffmpeg.org/ffmpeg.html

[19] L. Mike, "What year did obd2 start? here's a brief history lesson," 2020. , s Interneta, https://scanneranswers.com/what-year-did-obd2-start-heres-a-brief-history-lesson/

[20] "What is obdii? history of on-board diagnostics," Geotab Inc. , s Interneta, https://www.geotab.com/blog/obd-ii/

[21] linux can, "Socketcan userspace utilities and tools," 2021. , s Interneta, https://github.com/linux-can/can-utils

[22] H. Staff, "Hacking a chevy volt into a mario kart 64 controller," 2017. , s Interneta, https://www.hackster.io/news/hacking-a-chevy-volt-into-a-mario-kart-64-controller-f007bd64313

[23] *Steering Wheel Angle Sensor LWS*, Bosch Engineering GmbH, 2 2021. , s Interneta, https://www.bosch-motorsport.com/content/downloads/Raceparts/Resources/pdf/Data%20Sheet_191153675_Steering_Wheel_Angle_Sensor_LWS.pdf

[24] B. Thorne, "python-can 4.0.0 documentation." , s Interneta, https://python-can.readthedocs.io/en/master

[25] "Raspberry pi os," Raspberry Pi Trading Ltd, 2022. , s Interneta, https://www.raspberrypi.com/software/

[26] F. Chollet *et al.* (2015) Keras. , s Interneta, https://github.com/fchollet/keras

[27] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[28] "Image resizing with opencv," BIG VISION LLC, 2022. , s Interneta, https://learnopencv.com/image-resizing-with-opencv/

[29] F. Chollet *et al.*, "Adam," 2015. , s Interneta, https://keras.io/api/optimizers/adam/

[30] P. Wan, Z. Song, and J. Lu, "Incorporating orientations into end-to-end driving model for steering control," 2021. , s Interneta, https://arxiv.org/abs/2103.05846

SAŽETAK

U ovom radu dan je pregled postojećih skupova podataka sa stvarnim scenama iz prometa za razvoj algoritama za autonomnu vožnju zasnovanih na procjeni kuta zakreta upravljača vozila. Izrađen je novi sustav za prikupljanje stvarnih scena iz prometa. Glavno računalo sustava za prikupljanje podataka je Raspberry Pi, na koje je povezana Raspberry Pi kamera za snimanje scene ispred vozila i mikro-upravljač za CAN komunikaciju kojim se povezuje na dijagnostičko sučelje vozila. Rezultat prikupljanja skupa podataka su 266933 slike s vremenski uparenima kutovima zakreta upravljača vozila. Novi skup podataka nazvan je CanPi. Slike su snimljene tijekom vožnje u različitim tipovima okruženja (urbano, suburbano, ruralno okruženje), pri različitim vremenskim uvjetima (kiša, sunce, oblačno) i u različitim dobima dana (dan, noć). Skup podataka se sastoji od slika iz ukupno 17 različitih video snimki. Dodatno, u sklopu rada ispitano je korištenje stvorenog skupa podataka za dobivanje kvalitetnog modela koji predviđa kut zakreta upravljača. Dani su i komentirani dobiveni rezultati te su rezultati korištenja novog skupa podataka uspoređeni s rezultatima korištenja jednog postojećeg skupa podataka. Pokazalo se da treniranjem na obama skupovima podataka zajedno mogu se postići bolji rezultati i dobiti jedan model koji može predviđati rezultate na različitim skupovima podataka.

Ključne riječi **— autonomna vožnja, kut zakreta upravljača, skup podataka, strojno učenje, CanPi skup podataka**

DEVELOPMENT OF THE NEW DATASET WITH REAL TRAFFIC SCENES FOR THE PURPOSE OF DEVELOPING THE AUTONOMOUS DRIVING ALGORITHMS BASED ON VEHICLE STEERING ANGLE PREDICTION

ABSTRACT

This paper provides an overview of existing datasets with real traffic scenes for the purpose of developing the autonomous driving algorithms based on vehicle steering angle prediction. A new system for collecting real traffic scenes has been developed. The main computer of the data collection system is a Raspberry Pi, which is connected to a Raspberry Pi camera for recording the scene in front of the vehicle and a micro-controller for CAN communication, which is connected to the diagnostic interface of the vehicle. The result of collecting the dataset is 266933 images with time-matched vehicle steering angles. The new dataset is called CanPi. The pictures were taken while driving in different types of environments (urban, suburban, rural), under different weather conditions (rain, sun, cloudy) and at different times of the day (day, night). The dataset consists of images from a total of 17 different video recordings. Additionally, as part of the work, the use of the created dataset to obtain a high-quality model that predicts the angle of the steering wheel was tested. The obtained results are given and commented on, and the results of using a new dataset are compared with the results of using an existing dataset. It has been shown that training on both datasets together can achieve better results and yield a single model that can predict results on different datasets.

Keywords — **autonomous driving, steering angle, dataset, machine learning, CanPi dataset**

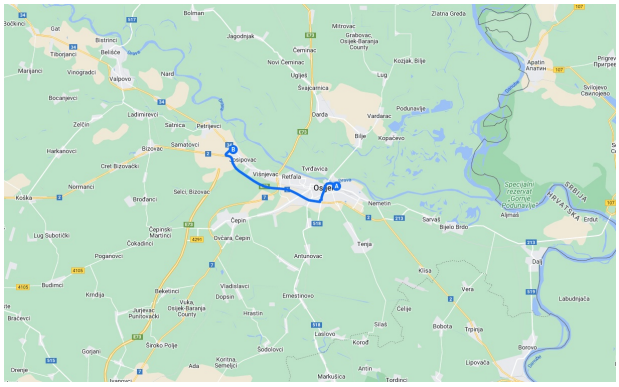
ŽIVOTOPIS

Matija Kelemen rođen je 16.8.1997 u Varaždinu. Završio je prirodoslovno-matematičku gimnaziju u Prvoj gimnaziji Varaždin s vrlo dobrim uspjehom. Zatim upisuje računarstvo na Tehničkom fakultetu u Rijeci. Godine 2020. stječe naziv sveučilišni prvostupnik (lat. *baccalaureus*) inženjer računarstva. Iste godine upisuje diplomski studij računarstva, smjer robotika i umjetna inteligencija na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U objema godinama studiranja na diplomskom studiju stipendiran je od strane Instituta RT-RK Osijek koji krajem druge godine postaje TTTech Auto Hrvatska.

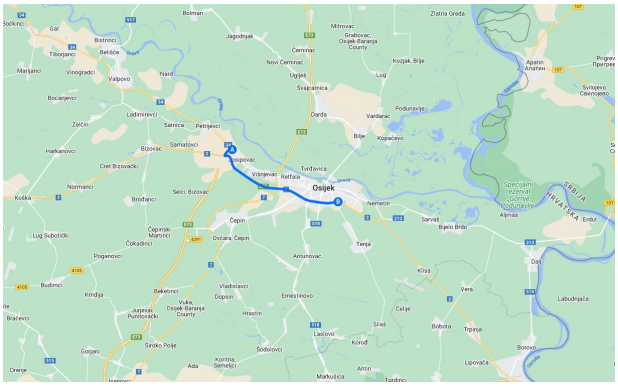
PRILOZI

P.3.1. Tablica s detaljima o pojedinim snimkama (elektronički prilog)

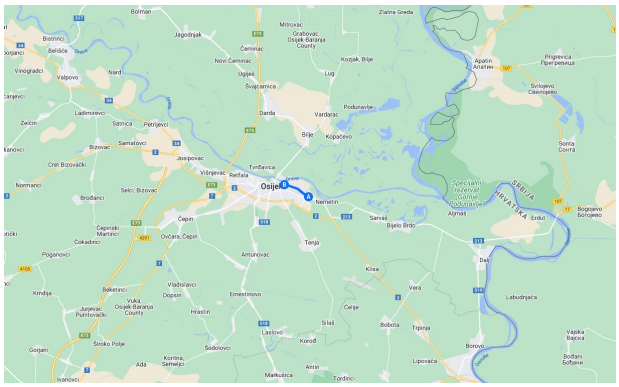
P.3.2. Prikaz ruti vožnji koje su služile za nastajanje 17 video snimki CanPi skupa podataka



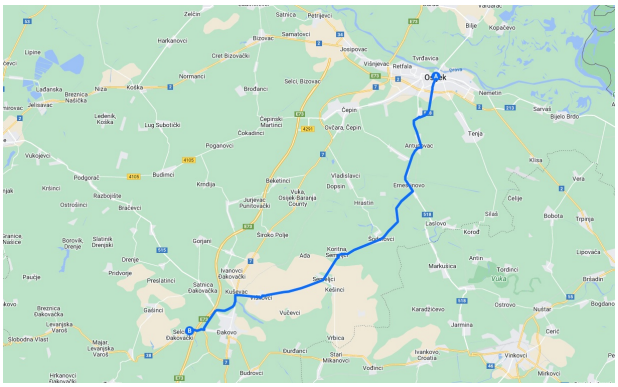
(a) Karta vožnje data-2022-02-09_094149



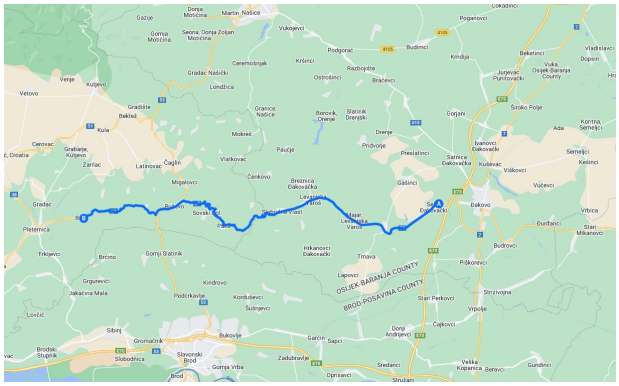
(b) Karta vožnje data-2022-02-09_100338



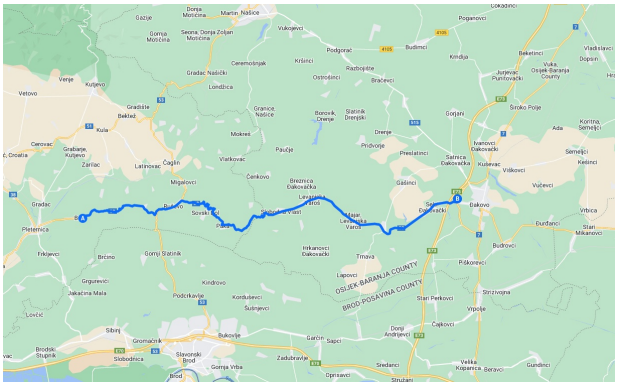
(c) Karta vožnje data-2022-02-09_102235



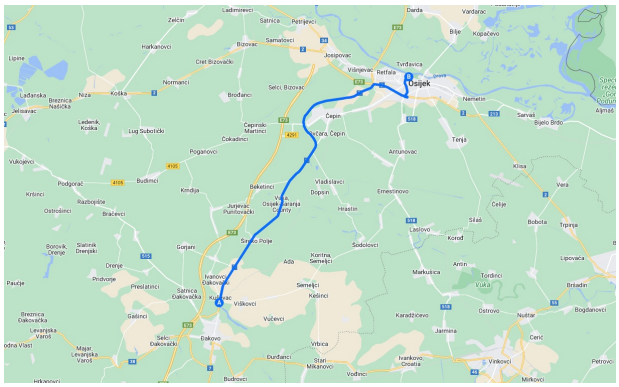
(d) Karta vožnje data-2022-02-16_111842



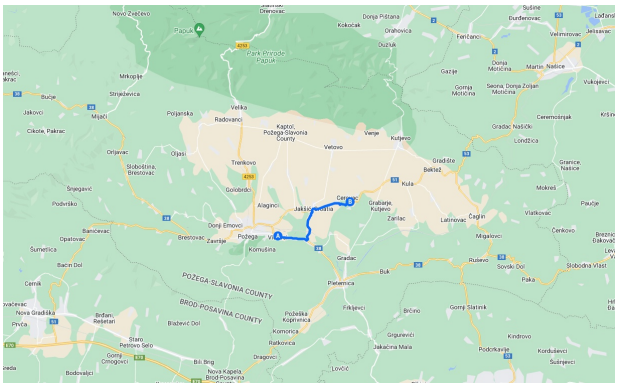
(e) Karta vožnje data-2022-02-16_120851



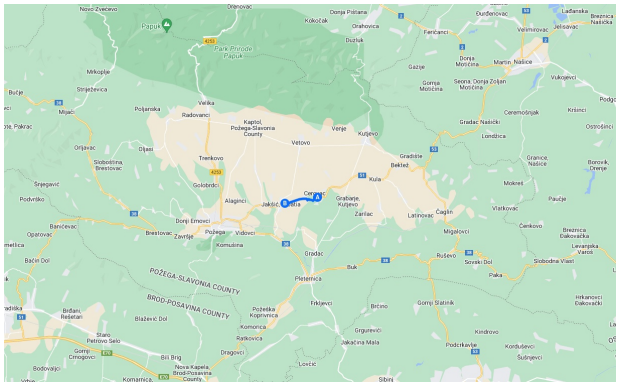
(f) Karta vožnje data-2022-02-16_131356



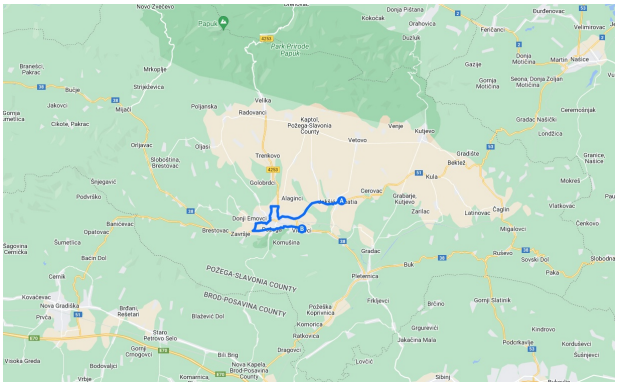
(g) Karta vožnje data-2022-02-16_140711



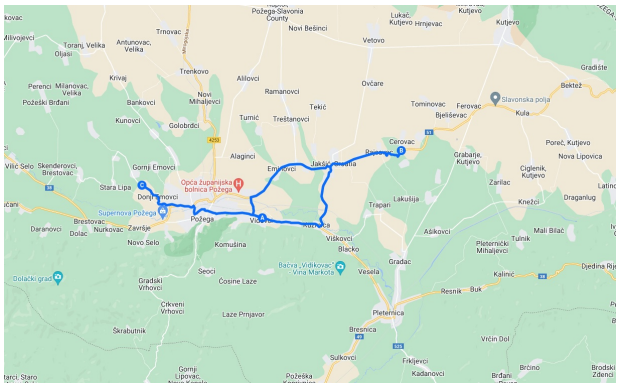
(h) Karta vožnje data-2022-02-19_140459



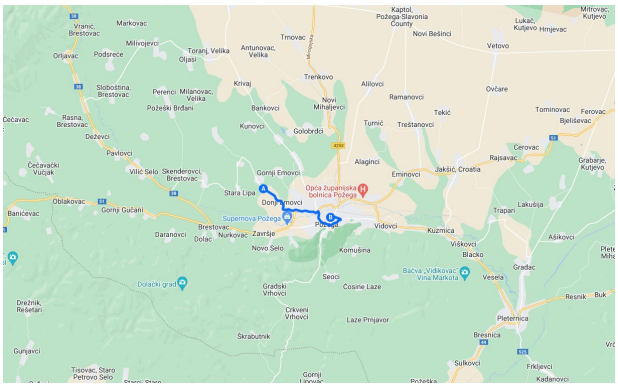
(i) Karta vožnje data-2022-02-19_142027



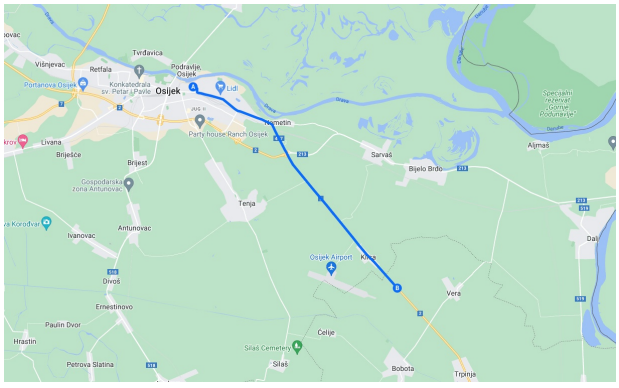
(j) Karta vožnje data-2022-02-19_152335



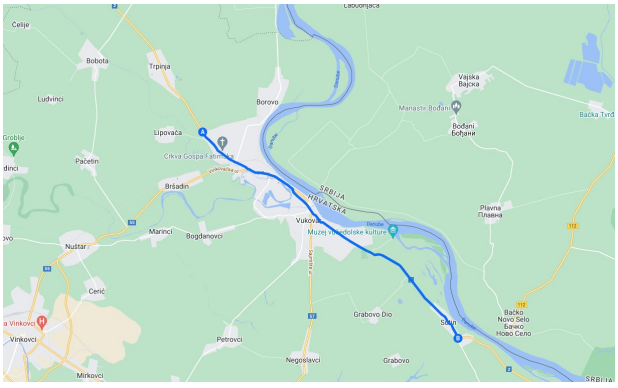
(k) Karta vožnje data-2022-02-19_210250



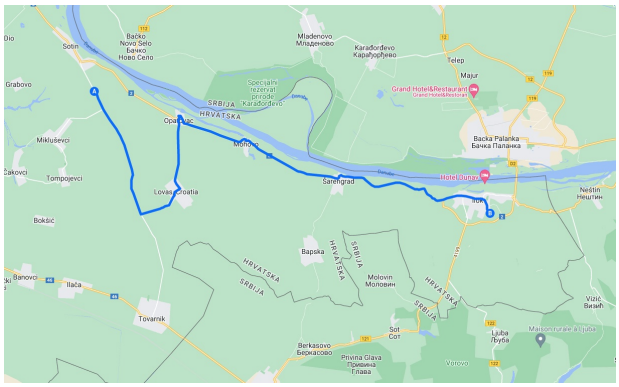
(l) Karta vožnje data-2022-02-19_214448



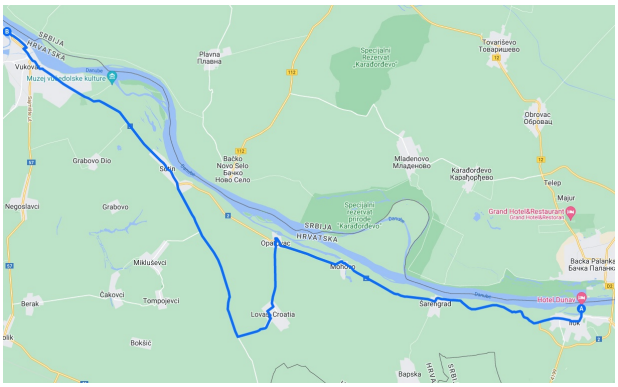
(m) Karta vožnje data-2022-04-29_142133



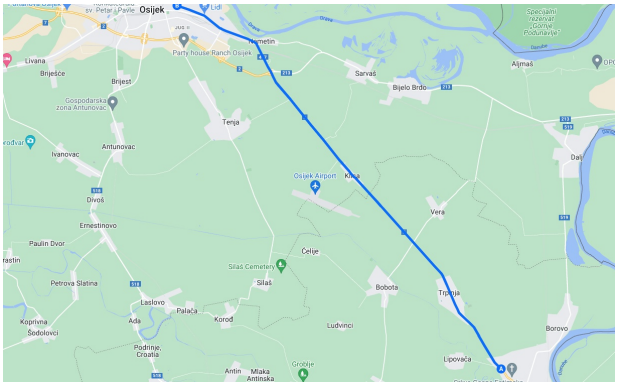
(n) Karta vožnje data-2022-04-29_144541



(o) Karta vožnje data-2022-04-29_150837



(p) Karta vožnje data-2022-04-29_162515



(q) Karta vožnje data-2022-04-29_171547

P.3.3. Bash skripta za pokretanje naredbi za prikupljanje skupa podataka

```
#!/bin/bash

# Check for parameters
STREAM=0
CONFIG_FILE=config.txt
for i; do
  if [[ $i == "-s" ]]; then STREAM=1; fi
done

# Start stream for
if [[ $STREAM -eq 1 ]]; then
  libcamera-vid --config $CONFIG_FILE --inline --listen -o
  ↪  tcp://0.0.0.0:5555
fi

# Create the interface
sudo ip link set can0 up type can bitrate 500000

# Start recording both
START=$(date +%Y-%m-%d_%H%M%S)
candump can0 -l &
CANBUS_PID=$!
libcamera-vid --config $CONFIG_FILE -o video-$START.h264 --save-pts
↪  points-$START.txt -s &
CAMERA_PID=$!

# Wait for q
GO=0
while [[ $GO -eq 0 ]]; do
```

```
  read -s -n 1 key
  if [[ $key = 'q' ]]; then GO=1; fi
done

# SIGUSR2 causes libcamera-vid to quit.
kill -SIGUSR2 $CAMERA_PID
kill -9 $CANBUS_PID

# Sleep for 5 sec
echo 'Waiting 5 seconds.'
sleep 5

# Write info file after recording
INFO_FILE=info-$START.txt
echo 'FRAME_START='$(stat -c %x video-$START.h264) > $INFO_FILE
cat $CONFIG_FILE >> $INFO_FILE

# Pack in tar
tar -czvf data-$START.tar.gz candump-$START.log info-$START.txt
↪  points-$START.txt video-$START.h264
rm candump-$START.log info-$START.txt points-$START.txt video-$START.h264
```

P.3.4. Programski kod za kreiranje i sinkronizaciju skupa podataka - *create_dataset.ipynb* (elektronički prilog)

P.3.5. Tekstualna datoteka s potrebnim bibliotekama za pokretanje programskog koda za kreiranje skupa podataka - *requirements.txt* (elektronički prilog)

P.3.6. DBC datoteka s opisima signala za dekodiranje CAN poruka - *canpi.dbc* (elektronički prilog)

P.4.1. Programski kod za stvaranje PilotNet modela u Keras programskom okruženju

```
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Input(shape=(66, 200, 3)),
    keras.layers.Conv2D(filters=24, kernel_size=(5, 5), strides=(2, 2),
    ↪  activation='relu'),
    keras.layers.Conv2D(filters=36, kernel_size=(5, 5), strides=(2, 2),
    ↪  activation='relu'),
    keras.layers.Conv2D(filters=48, kernel_size=(5, 5), strides=(2, 2),
    ↪  activation='relu'),
    keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Flatten(),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(50, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(1),
])

model.compile(loss=loss, optimizer=optimizer)
```