Procjena dubinske mape na temelju slike dobivene s jedne kamere u okviru autonomne vožnje

Pocrnja, Josip

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

Sveučilišni studij računarstva

PROCJENA DUBINSKE MAPE NA TEMELJU SLIKE DOBIVENE S JEDNE KAMERE U OKVIRU AUTONOMNE VOŽNJE

Diplomski rad

Josip Pocrnja

Osijek, 2023. godina.

SADRŽAJ

1. UVOD

Svaki element slike u dubinskoj mapi predstavlja informaciju o udaljenosti od kamere (ili senzora) do objekata ili površina u sceni. Dubinsku mapu moguće je dobiti iz monokularnih (pojedinačnih) ili stereo (više prikaza scene) slika. Tradicionalne metode koriste geometriju s više pogleda za pronalaženje odnosa između slika. Ovaj zadatak danas ima široku primjenu u različitim područjima kao što su robotika, autonomna vožnja, razumijevanju scene i rekonstrukciji 3D prostora. Ove primjene obično koriste višestruke primjere iste scene, poput parova stereo slika [1], više kadrova s pokretnom kamerom ili statičkih snimaka pod različitim uvjetima osvjetljenja kako bi se obavilo mapiranje dubine.

S obzirom na impresivan napredak postignut u mapiranju dubine iz više promatranja, prirodno je došlo do razvoja mapiranja dubine iz jedne slike jer zahtijeva manje troškova i ograničenja. Međutim, precizna procjena dubine iz jedne slike je izazovna, čak i za ljudsko oko, jer predstavlja loše postavljen problem s beskonačno mnogo 3D scena koje se mogu projicirati na istu 2D sliku. Kako bi razumjeli geometrijsku konfiguracije iz jedne slike, ljudi uzimaju u obzir ne samo lokalne značajke, poput izgleda tekstura u različitim uvjetima osvjetljenja i zaklanjanja, perspektive ili relativne skale prema poznatim objektima, već i globalni kontekst, kao što je cjelokupni oblik ili raspored scene [2].

Cilj procjene dubine iz monokularne slike je poboljšanje sposobnosti računalnih sustava u razumijevanju 3D prostora na temelju jedne slike, bez potrebe za dodatnim senzorima ili skupnim sustavima. Kroz daljnji razvoj arhitektura neuronskih mreža, njihove optimizacije i korištenjem velikih skupova podataka, očekuje se daljnji napredak u preciznosti procjene dubine iz monokularnih slika, što će imati široku primjenu u mnogim područjima računalnog vida i pomoći u razvoju naprednih autonomnih sustava.

Razvoj procjene dubine iz monokularne slike temeljene na konvolucijskim neuronskim mrežama (engl. *Convolutional Neural Network* - CNN) omogućio je značajan napredak u ovom području. Konvolucijske mreže mogu naučiti složene prostorne obrasce iz slike i koriste se za izvlačenje značajki koje su relevantne za procjenu dubine. Arhitekture temeljene na kodiranju i dekodiranju pokazale su se

korisnima jer istovremeno zahtijevaju globalni kontekst i rezoluciju na razini elementa slike. Nedavno su se pojavile i arhitekture temeljene na pozornosti (engl. *attention-based*) koje koriste prednosti pozornosti u raznim zadacima računalnog vida. Ove arhitekture omogućuju bolje modeliranje veza između mapa značajki enkodera i dekodera te bolju integraciju globalnog konteksta i lokalne rezolucije [3].

U okviru ovog rada je izrađen algoritam za procjenu dubinske mape pomoću neuronske mreže na osnovu slike dobivene s prednje kamere vozila. Predloženi algoritam implementiran je u programskom jeziku Python uz pomoć Pytorch biblioteke za strojno učenje. Algoritam se temelji na neuronskoj mreži *PixelFormer* za procjenu dubinske mape. Ova mreža je prilagođena CARLA (engl. *Car Learning to Act*) simulatoru [4] na način da je mreža dotrenirana slikama prikupljenim u simulatoru. Na temelju algoritma za procjenu dubinske mape izrađen je algoritam za automatsko kočenje vozila u simulatoru u slučaju nailaska na prepreku. Predloženi algoritam detaljno je evaluiran u CARLA simulatoru.

U drugom poglavlju rada objašnjen je problem procjene dubinskih mapa na temelju monokularne slike i detaljnije su opisane postojeće metode procjene dubinskih mapa. U trećem poglavlju objašnjen je način funkcioniranja odabranog algoritma za procjenu dubinske mape na temelju monokularne slike i predstavljen je algoritam za automatsko kočenje vozila u simulatoru. Opisane su korištene biblioteke i skup podataka za treniranje neuronske mreže. U četvrtom poglavlju opisan je način evaluacije predloženog algoritma i predstavljeni su postignuti rezultati na podacima u CARLA simulatoru. Na kraju je dan zaključak rada.

2. PREGLED POSTOJEĆIH METODA ZA PROCJENU DUBINSKE MAPE NA TEMELJU SLIKE DOBIVENE S JEDNE KAMERE U OKVIRU AUTONOMNE VOŽNJE

2.1. Problem procjene dubinske mape

Procjena dubinske mape (engl. *Depth Estimation* - DE) podrazumijeva određivanje udaljenosti ili dubine objekata u sceni. To se može postići primjenom pasivnih i aktivnih pristupa [5]. U aktivnim pristupima koriste se tehnologije poput LIDAR (engl. *Laser Imaging, Detection, And Ranging*) senzora i RGB-D (engl. *Red, Green, Blue – Depth*) kamera kako bi se dobila dubinska informacija.

LIDAR je metoda za određivanje dometa ciljanjem objekta ili površine laserom i mjerenjem vremena potrebnog za povratak reflektirane svjetlosti do prijemnika. LIDAR može raditi u fiksnom smjeru ili može skenirati više smjerova. Međutim, LIDAR može biti neučinkovit u lošim vremenskim uvjetima i često je oprema skupa.

Princip rada RGB-D kamere je da snimaju RGB slike i pripadajuće dubinske mape, što ih čini pogodnima za primjenu u pametnim telefonima i bespilotnim letjelicama zbog niske cijene i potrošnje energije. Međutim, RGB-D kamere imaju ograničen raspon dubine i mogu biti osjetljive na odsjaj i apsorbirajuće objekte koje su prisutne u sceni.

U pasivnim pristupima, procjena dubine se često postiže korištenjem dvije osnovne metodologije: dubina iz stereo slika i dubina iz monokularnih slika. Glavni cilj oba pristupa je pomoć u izgradnji prostorne strukture okoliša, koja predstavlja 3D prikaz scene. Stereo vid je postupak ekstrakcije 3D informacija iz para slika koje se dobiju pomoću dvije kamere na fiksnoj udaljenosti. Usporedbom informacija o sceni s dvije različite točke gledišta, moguće je izdvojiti 3D informacije ispitivanjem relativnih položaja objekata na tim slikama.

Ponekad primjena algoritama za izračunavanje dubinske mape stereo pristupom može stvoriti različite izazove. Na primjer, funkcija za usklađivanje troškova koja se koristi u algoritmu može generirati lažno pozitivne signale, što rezultira stvaranjem dubinskih mapa niske točnosti. Stoga je korištenje postupaka naknadne obrade (npr.

median filter, bilateral filter i interpolacija) od velike važnosti u aplikacijama stereo vida kako bi se uklonio šum i poboljšala procjena dubinske mape.

S druge strane, monokularna procjena dubine ne zahtijeva podudaranje stereo slika budući da modeli monokularne procjene dubine rade s nizom slika izdvojenih iz jedne kamere. Ova jednostavnost i lak pristup jedna su od glavnih prednosti monokularne procjene dubine u usporedbi sa stereo pristupima, koji zahtijevaju dodatne kompliciranе dijelove opreme. Metode monokularne procjene dubine usredotočene su na procjenu udaljenosti između objekata i kamere s jedne točke gledišta. Cilj tih metoda je rekonstruirati dubinske mape koje prikazuju 3D strukturu scene. Procjena dubinske mape se danas rješava pristupima temeljenim na dubokim neuronskim mrežama (engl. *Deep Neural Networks* - DNN). Većina modela monokularne procjene dubine ima osnovnu arhitekturu koja se sastoji od dvije glavne komponente: dubinske mreže i mreže pozicije (engl. *Pose network*). Dubinska mreža predviđa dubinske mape, dok mreža pozicija radi procjenu pokreta kamere između uzastopnih slika [5].

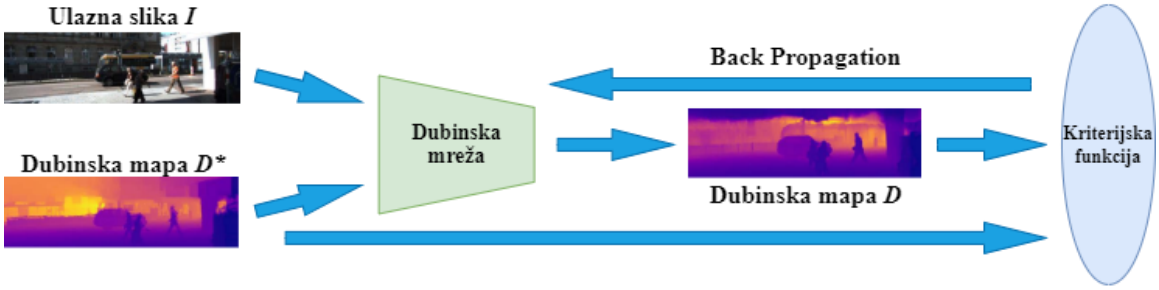
Zbog gubitka 3D informacija u procesu snimanja slike dobivene monokularnom kamerom, nije jednostavno procijeniti dubinsku mapu iz slike koja je snimljena iz jednog kuta. Za razliku od stereo metoda koje koriste par slika dobivenih pomoću dviju kamera za procjenu dubine, ranije metode računalnog vida s monokularnim kamerama koriste različite vizualne karakteristike kao znakove za predviđanje dubinske mape, kao što su tekstura, granice, defokus, boja, okluzija, magla, površinske značajke i veličina poznatih objekata [6].

2.2. Pregled postojećih metoda za procjenu dubinske mape na temelju slike dobivene s jedne kamere

Uspjeh dubokog učenja u klasifikaciji slika potakao je razvoj i primjenu metoda za procjenu dubine pomoću monokularnih kamera. U ovom potpoglavlju dan je pregled metoda temeljenih na dubokom učenju za monokularnu procjenu dubine. Postoje tri vrste pristupa učenju koja se razlikuju po količini označenih podataka, to jest dubinskih mapa: nadzirano, nenadzirano i polunadzirano učenje [6].

2.2.1. Procjena dubine pomoću metoda temeljenih na nadziranom učenju

Tok rada metoda procjene dubine temeljenih na nadziranom učenju prikazan je na slici 2.1 i može se opisati na sljedeći način: mreža za procjenu dubine koristi RGB sliku *I* i odgovarajuću označenu dubinsku mapu *D** (engl. *ground-truth*) kako bi naučila informacije o strukturi scene za procjenu dubinske mape *D*. Zatim se parametri mreže ažuriraju minimiziranjem funkcije gubitka *L*(*D**, *D*), koja mjeri razliku između *D* i *D**.



Slika 2.1: Princip treniranja mreže nadziranim učenjem, ulaz u mrežu je RGB slika i označena dubinska mapa [6]

Primjer metode za procjenu dubinske mape iz jedne RGB slike koja se temelji na nadziranom učenju je *PixelFormer* [7] koji je korišten u ovom radu. *PixelFormer* predstavlja novu strategiju enkodera i dekodera za monokularnu procjenu dubine koja postavlja problem kao zadatak za doradu upita elemenata slike. Elementi slike se prvo obrađuju za procjenu dubinske mape na globalnoj razini, a onda se dorađuju na višu razlučivost putem modula za preskakanje pozornosti (engl. *Skip Attention Module* - SAM).

Drugi primjer mreže za nadzirano učenje je *AdaBins* [8]. Mreža započinje procjenu s osnovnom arhitekturom konvolucijske neuronske mreže enkoder-dekoder i postavlja pitanje kako globalna obrada informacija može pomoći u poboljšanju ukupne procjene dubine. U tu svrhu predložen je blok arhitekture temeljen na transformeru koji dijeli raspon dubine u spremnike (engl. *bins*) čija se središnja vrijednost procjenjuje

adaptivno po slici. Konačne vrijednosti dubinske mape procjenjuju se kao linearne kombinacije središta spremnika.

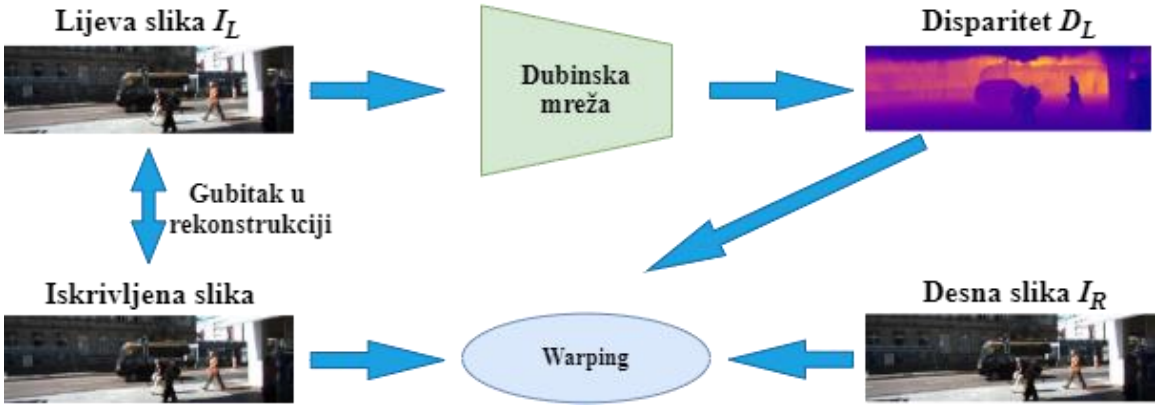
2.2.2. Procjena dubine pomoću metoda temeljenih na nenadziranom učenju

Metode nenadziranog učenja koriste stereo slike ili video sekvence s malim promjenama položaja kamere između kadrova jer se dva uzastopna kadra mogu tretirati kao stereo slike. Ove metode formuliraju procjenu dubine kao problem rekonstrukcije slike, pri čemu su dubinske mape međuproizvod koji se integrira u gubitak rekonstrukcije slike. Tok rada metoda temeljenih na nenadziranom učenju je prikazan na slici 2.2 i može se opisati na sljedeći način: mreža koristi dvije slike (nazovimo ih I_L i I_R) iste scene, ali s različitim perspektivama. Naknadno se procjenjuje dubinska mapa za I_L, a dobivena dubinska mapa označena je s D_L. S D_L i pokretom kamere između slika, I_R se može transformirati u sliku koja je slična I_L jednadžbom:

$$I_R(D_L) \rightarrow \tilde{I}_L \qquad (2-1)$$

gdje je $\tilde{I}_L$ transformirana slika. Mreža se može trenirati s odgovarajućim gubitkom rekonstrukcije:

$$\text{gubitak} = L(\tilde{I}_L, I_L) \qquad (2-2)$$



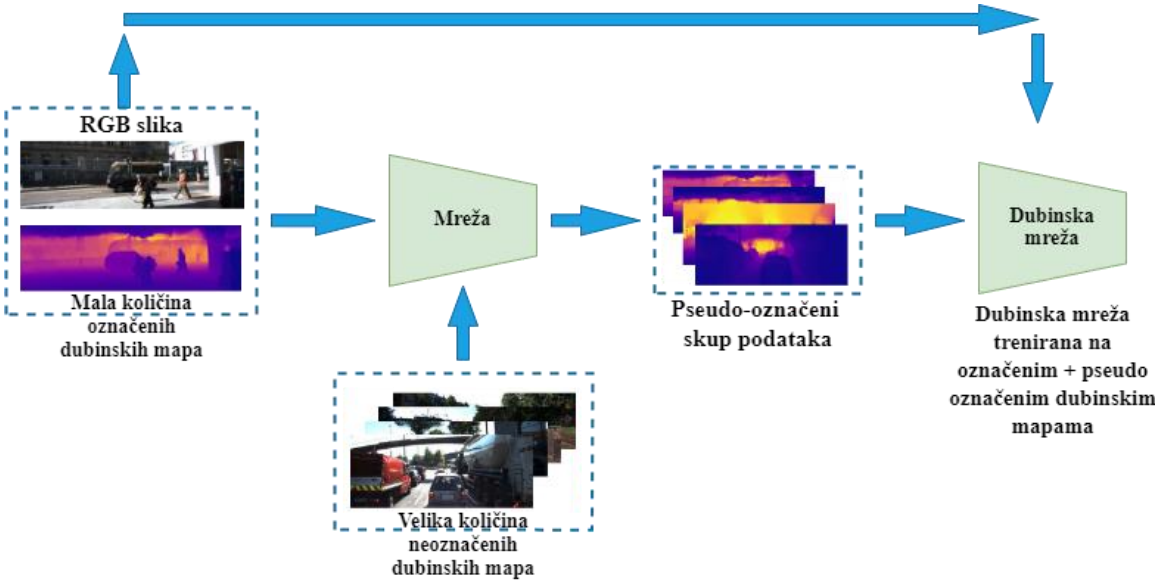
Slika 2.2: Princip treniranja mreže nenadziranim učenjem, ulaz u mrežu je par stereo slika [6]

Primjer metode za procjenu dubinske mape iz jedne RGB slike koja se temelji na nenadziranom učenju predstavljen je u radu [9] koji predviđa dubinsku mape sintetizirajući mape značajki s različite točke gledišta. Dizajnirana mreža uključuje tri dijela: ekstraktor značajki s više skala, početni procjenitelj dispariteta i modul za dotjerivanje dispariteta. Izdvojene značajke prosljeđuju se početnom procjenitelju dispariteta za predviđanje mapa dispariteta koje su usklađene s ulaznom i sintetiziranom slikom desnog prikaza. Modul za pročišćavanje pročišćava početnu nejednakost izvođenjem stereo usklađivanja između stvarnih i sintetiziranih prikaza mapa značajki.

2.2.3. Procjena dubine pomoću metoda temeljenih na polunadziranom učenju

Metode nenadziranog učenja eliminiraju potrebu za označenim dubinskim mapama, čije dobivanje je često vremenski zahtjevno i skupo. Međutim, njihova točnost je ograničena stereoskopskom rekonstrukcijom slike. Iz tog razloga, metode polunadziranog učenja koriste malu količinu RGB slika koje imaju pripadne dubinske mape i veliku količinu neoznačenih RGB slika kako bi poboljšale točnost procjene dubine.

Princip treniranja opće mreže za procjenu dubinske mape polunadziranim pristupom na temelju monokularnih slika prikazana je na slici 2.3. Najprije se neuronska mreža trenira s malom količinom parova RGB slika i pripadnih dubinskih mapa sve dok ne postigne zadovoljavajuće rezultate. Zatim se trenirana mreža koristi s neoznačenim podacima kako bi generirala izlaze koji se nazivaju pseudo-označene dubinske mape. Kao posljednji korak, neuronska mreža se ponovno trenira s parovima RGB slika i pripadnih dubinskih mapa zajedno s generiranim pseudo-označenim dubinskim mapama.



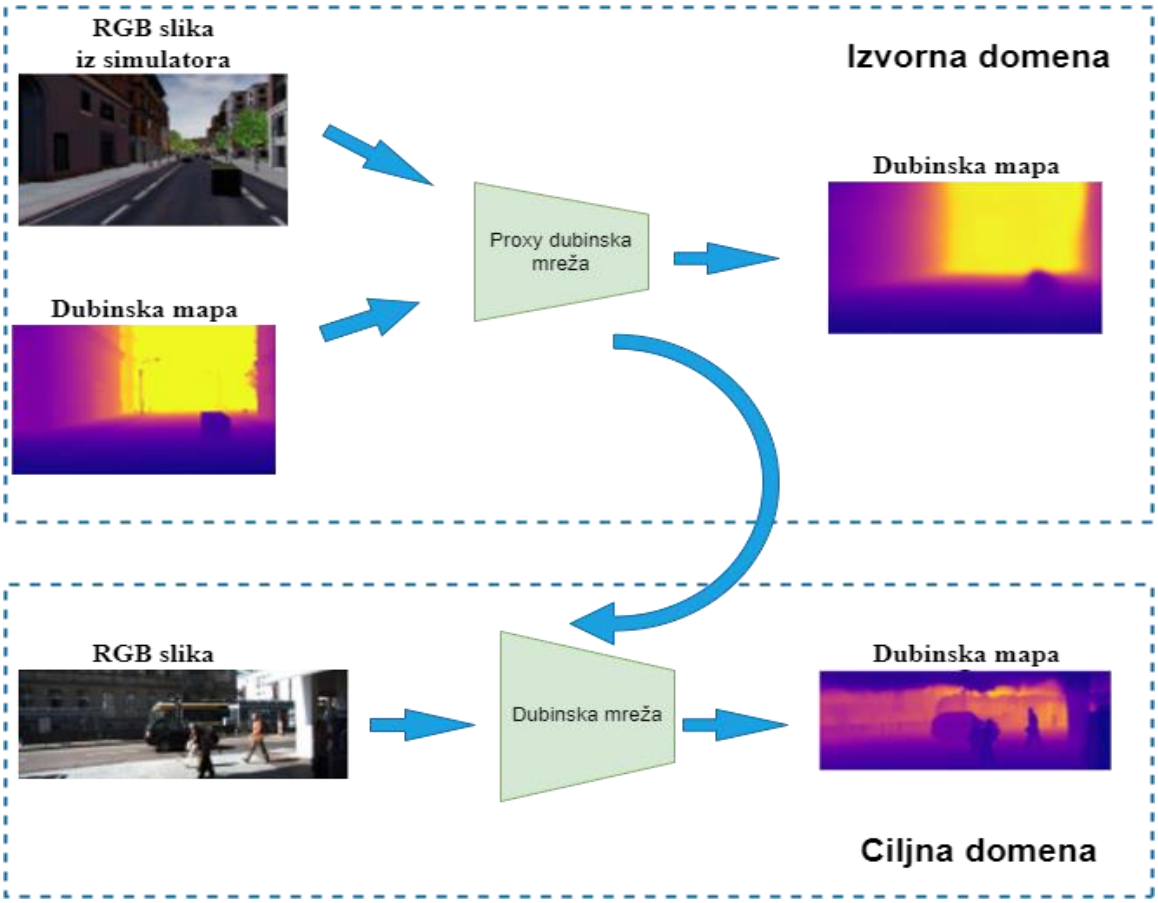
Slika 2.3: Princip treniranja mreže polunadziranim pristupom, ulaz u mrežu je mala količina parova RGB slika i dubinskih slika i velika količina neoznačenih RGB slika [6]

Primjer metode za procjenu dubinske mape iz jedne RGB slike koja se temelji na polunadziranom učenju može se pronaći u radu [10]. Autori uvode polu-nadziranu kontradiktornu mrežu učenja koja se trenira na malom broju parova dubine slike i velikom broju neoznačenih monokularnih slika. Predloženi okvir sastoji se od generatorske mreže za procjenu dubine i dvije diskriminatorske mreže za mjerenje kvalitete procijenjene dubinske mape.

2.2.4. Procjena dubine pomoću metoda temeljenih na prilagodbi domene

U prethodnim točkama dan je pregled metoda temeljenih na dubokim neuronskim mrežama za monokularno procjenjivanje dubine koje su trenirane na podacima prikupljenim u stvarnom svijetu. Nedavni napredci u računalnoj grafici i modernim platformama visoke razine kao što su grafički programi za razvoj i upravljanje videoigrama omogućuju generiranje velikog skupa sintetičkih 3D scena. S izgrađenim scenama, znanstvenici mogu lako i brzo stvoriti veliku količinu sintetičkih RGB slika s pripadnim dubinskim mapama koje se mogu koristiti u svrhu nadziranog učenja mreža za procjenu dubinskih mapa. Takav pristup treniranju modela značajno smanjuje troškove vezane uz prikupljanje stvarnih skupova podataka koji sadrže veliki broj parova RGB slika i dubinskih slika. Međutim, modeli trenirani na sintetičkim podacima obično ne generaliziraju dobro na stvarne scene zbog inherentne razlike u domeni. Kako bi se riješio taj problem, predložene su metode temeljene na prilagodbi domene koje najprije treniranju modele monokularne procjene dubine na sintetičkim podacima kako bi ublažile učinke razlika u karakteristikama između sintetičkih i stvarnih podataka. Cilj je istrenirati model na sintetičkim podacima, a onda još dodatno podesiti na manjem broju parova stvarnih RGB slika i dubinskih mapa kao što je prikazano na slici 2.4.

Ovakvi pristupi najprije treniraju mrežu na slikama iz određene domene kao što su sintetički podaci, a zatim je fino podešavaju na slikama iz ciljne domene. Primjer takve mreže je *DispNet* [11] koji primjenjuje fino podešavanje za procjenu dubinske mape. *DispNet* se prvo trenira na velikom sintetičkom skupu podataka, a zatim se fino podešava na manjem skupu podataka koji ima označene dubinske mape. Metode temeljene na finom podešavanju obično zahtijevaju određenu količinu označenih podataka iz ciljne domene.



Slika 2.4: Princip rada metode prilagodbe domene [6]

3. PREDLOŽENI ALGORITAM ZA PROCJENU DUBINSKE MAPE NA TEMELJU SLIKE DOBIVENE S JEDNE KAMERE I AUTOMATSKO KOČENJE U OKVIRU AUTONOMNE VOŽNJE

U ovom poglavlju opisan je predloženi algoritam za monokularnu procjenu dubinske mape i automatsko kočenje u okviru CARLA simulatora. Procjena dubinske mape se temelji na neuronskoj mreži *PixelFormer* koja je zasnovana na *Swin* transformeru (engl. *Swin transformer*) [12]. Predložena mreža najprije je trenirana i testirana na KITTI skupu podataka [13], a nakon toga je prilagođena za rad u simulacijskim uvjetima dotreniravanjem na prikupljenim sintetičkim podacima u CARLA simulatoru. Izlaz iz mreže je zatim korišten u algoritmu za automatsko kočenje vozilom kako bi se vozilo zaustavilo na određenoj udaljenosti u slučaju nailaska na prepreku ili drugo vozilo u simulatoru.

3.1. Biblioteke i alati korišteni za razvoj algoritma

U ovom potpoglavlju dan je kratak pregled najznačajnijih biblioteka i alata koji su korišteni za razvoj rješenja. U prilogu P3.1. moguće je pronaći kompletan popis korištenih biblioteka i njihovih verzija.

3.1.1. Pytorch

Pytorch [14] je biblioteka otvorenog koda za strojno učenje razvijena u Facebookovom laboratoriju za istraživanje umjetne inteligencije. Tehnologija strojnog učenja razvijena je koristeći Python i C++ programski jezik te CUDA API. Prvi put je objavljena u rujnu 2016. godine i od tad se koristi za mnoge primjene strojnog učenja. Kao što ime sugerira, Pytorch je prvenstveno dizajniran za upotrebu u pythonu, iako Pytorch ima C++ sučelje. S druge strane, Torch je također otvorena biblioteka za strojno učenje na kojoj je Pytorch temeljen i zajedno s Pythonom čini naziv Pytorch.

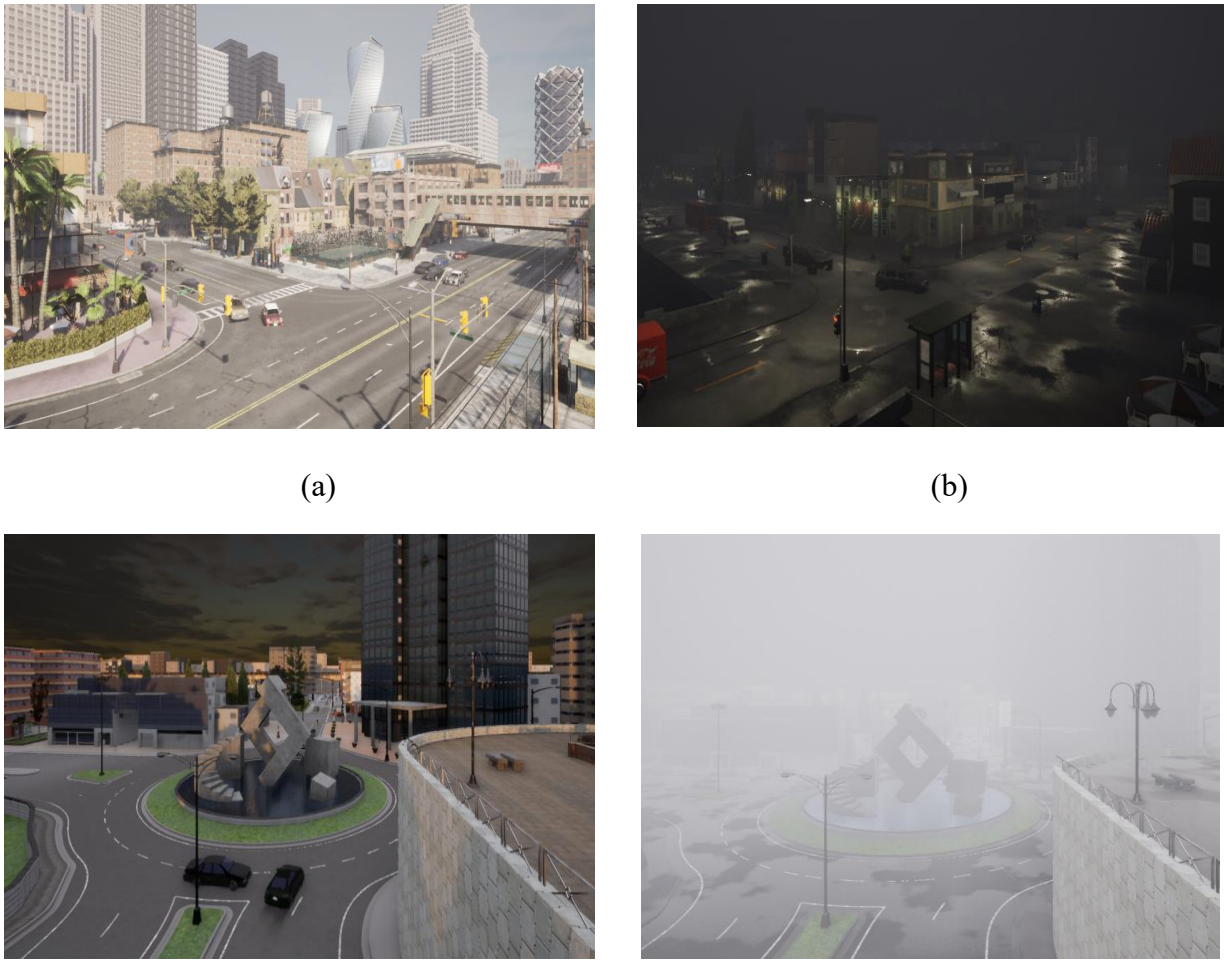
3.1.2. CARLA simulator

CARLA je besplatni simulator otvorenog koda za razvoj i istraživanje autonomne vožnje [4]. Simulator pruža okolinu u kojoj je moguće treniranje i testiranje novih modela za autonomnu vožnju. Okolina se sastoji od 3D modela statičnih objekata kao što su zgrade, vegetacija, prometni znakovi i infrastruktura, kao i dinamičnih objekata kao što su vozila i pješaci. Simulator ima mogućnost stvaranja velikog broja senzora na vozilu kao što su dinamički vizualni senzori, RGB i dubinske kamere, kamere optičkog toka, senzori kolizije, LIDAR, radar, navigacijski sustav i akcelerometar. Osim senzora simulator pruža niz mjerenja povezanih sa stanjem vozila uključujući lokaciju i orijentaciju vozila u odnosu na svjetski koordinatni sustav i brzinu. Omogućuje pristup točnim lokacijama i graničnim okvirima svih dinamičkih objekata u okruženju.

U okviru ovog rada u simulatoru se promatra vozilo koje je u interakciji s okolinom u diskretnim vremenskim koracima. Odgovarajući programski kod koji upravlja vozilom u simulatoru obično se naziva agent. U svakom vremenskom koraku, agent dobiva zapažanje o_t i mora proizvesti akciju a_t. Akcija je trodimenzionalni vektor koji predstavlja zakret volana, vrijednost gasa i vrijednost kočnice. Opservacija o_t je niz senzorskih ulaza. To može uključivati visokodimenzionalna senzorska opažanja, kao što su slike u boji i dubinske mape, i niže dimenzionalna mjerenja, kao što su brzina i GPS očitanja.

CARLA simulira dinamičan svijet i pruža jednostavno sučelje između svijeta i agenta koji je u interakciji sa svijetom. Kako bi podržala ovu funkcionalnost, CARLA je dizajnirana kao poslužitelj-klijent sustav, gdje poslužitelj pokreće simulaciju pomoću izvršne datoteke i prikazuje scenu. Klijentsko programsko sučelje (engl. *Application Programming Interface* - API) implementirano je u Pythonu i odgovorno je za interakciju između autonomnog vozila i servera putem utičnica (engl. *Sockets*). Klijent šalje naredbe poslužitelju i prima odgovore senzorskih očitanja. Naredbama kojima se kontrolira vozilo uključuju zakret volana, ubrzanje i kočenje. Klijent je u mogućnosti upravljati i poslužiteljem i moguće je ponovno postavljanje simulacije, promjenu svojstava okoline i modificiranje skupa senzora. Svojstva okoline uključuju vremenske uvjete, osvjetljenje te gustoću vozila i pješaka. Kada se poslužitelj resetira, vozilo se ponovno inicijalizira na

novoj lokaciji koju odredi klijent [4]. Okolina simulatora u različitim vremenskim uvjetima i na različitim lokacijama prikazana je na slici 3.1.



(a) (b)

(c) (d)

Slika 3.1: Primjeri okruženja CARLA simulatora u različitim vremenskim uvjetima: (a) vedro vrijeme, (b) kišno, noćno vrijeme, (c) izlazak sunca i (d) maglovito vrijeme

3.2. Opis skupova podataka

U ovom potpoglavlju opisana su dva skupa podataka koja su korištena prilikom razvoja rješenja. Prvi skup podataka je KITTI *Vision Benchmark Suite* koji sadrži RGB slike i pripadne dubinske mape iz stvarnog svijeta dok je drugi skup podataka vlastito prikupljen sintetički skup podataka dobiven u CARLA simulatoru.

3.2.1 KITTI skup podataka

Odabrani algoritam monokularne procjene dubine za treniranje pripadne mreže koristi KITTI *Vision Benchmark Suite* [13] skup podataka (u nastavku teksta ovaj podatkovni skup se skraćeno naziva KITTI). To je skup podataka u kojem se nalaze različite vanjske scene u više prometnih situacija i upravo je zato pogodan za izgradnju predloženog algoritma. Ovaj skup podataka pruža stereo slike i pripadajuća 3D laserska skeniranja vanjskih scena snimljenih pomoću opreme montirane na vozilo. Skup podataka sadrži ulazne RGB slike od 1241x375 razlučivosti elemenata slike (slika 3.2) i pripadne dubinske mape koje su dobivene iz LIDAR skeniranja. Mjerenja dubine pomoću LIDAR-a provode se za diskretne točke ili elemente slike u okruženju. Te točke odgovaraju objektima ili površinama koje reflektiraju laserske impulse. Budući da LIDAR impulsi putuju u ravnim linijama, oni mogu mjeriti samo udaljenosti do objekata koji su izravno na njihovom putu. Kao rezultat toga, vrijednosti elemenata slike kod LIDAR dubinske mape često su rijetke (slika 3.3), što znači da imaju praznine ili područja bez informacija o dubini. Skup podataka je podijeljen na dva: skup podataka za treniranje koji se sastoji od 26000 podataka za treniranje i skup podataka za testiranje koji se sastoji od 697 slika.

Slika 3.2: Primjer RGB slike iz KITTI podatkovnog skupa

Slika 3.3: Primjer dubinske mape dobivene LIDAR skeniranjem iz KITTI podatkovnog skupa

3.2.2. Izgradnja vlastitog sintetičkog skupa podataka

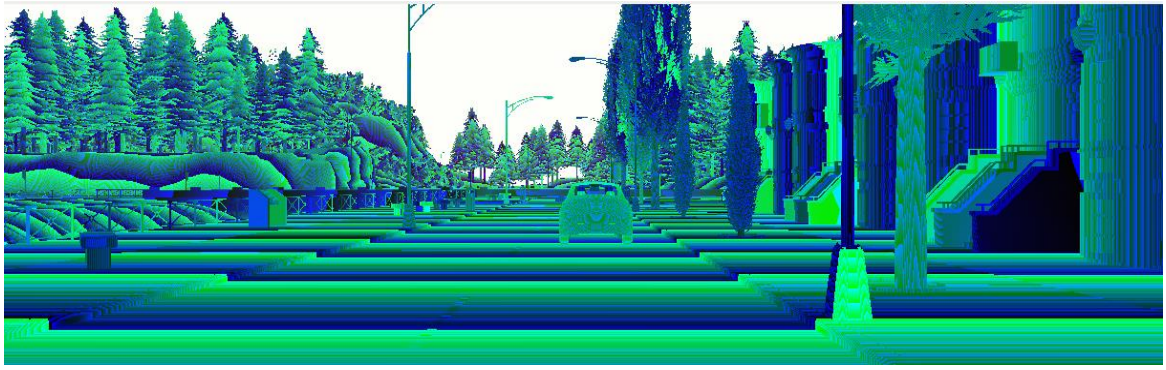
Za izradu vlastitog skupa podataka u okviru CARLA simulatora bilo je potrebno prikupiti označene podatke koji su slični označenim podacima u KITTI podatkovnom skupu. Za stvaranje pripadajućih RGB slika i označenih dubinskih mapa, napravljene su instance RGB i dubinske kamere i postavljene se na prednju stranu vozila. Za izgradnju vlastitog skupa podataka korištena je ista razlučivost RGB slike kao i kod KITTI skupa od 1241x375 elemenata slike. Primjer RGB slike koja se dobije na ovaj način prikazana je na slici 3.4.

Slika 3.4: Primjer RGB slike dobivene s prednje kamere vozila u CARLA simulatoru

Dubinska kamera pruža neobrađene podatke o sceni kodirajući udaljenost svakog elementa slike od kamere (također poznat kao dubinski međuspremnik ili z-međuspremnik) kako bi se stvorila dubinska mapa elemenata. Slika kodira vrijednost dubine po elementu slike pomoću tri kanala RGB prostora boja kao što je prikazano na slici 3.5. Stvarna udaljenost za svaki element slike u metrima $d_{i,j}$ za neki element slike na poziciji (i,j) može se izračunati pomoću izraza:

$$d_{ij} = \frac{R_{ij} + G_{ij} * 256 + B_{ij} * 256 * 256}{256 * 256 * 256 - 1} * 1000 \qquad (3-1)$$

Rezultantna dubinska mapa prikazana je na slici 3.6. u sivim tonovima, gdje svaka vrijednost elementa slike odgovara udaljenosti u metrima, npr. ako element slike ima vrijednost 10 znači da je objekt na mjestu tog elementa slike udaljen 10 metara od kamere. Što je objekt bliže kameri biti će prikazan tamnijom nijansom, ako je objekt dalje od kamere bit će prikazan svjetlijom nijansom, najveća moguća vrijednost udaljenosti od kamere je 255 metra. Kako bi ovaj skup bio što sličiniji KITTI skupu podataka, dubinski senzor je ograničen na računanje udaljenosti do 80 metara, a vrijednosti koje premaše tu udaljenost automatski su postavljene na 80 metara.



Slika 3.5: Primjer izlaza iz prednje dubinske kamere

Slika 3.6: Primjer dobivene dubinske mape na temelju slike 3.5.

Za stvaranje vlastitog skupa podataka izmijenjena je skripta *lidar_to_camera.py* dostupna unutar API-a simulatora. Budući da je potrebno uskladiti rad dva različita senzora, korišten je sinkroni način rada simulacije. U sinkronom načinu rada poslužitelj čeka klijenta prije nego što ažurira simulaciju na sljedeći korak. Sinkroni način rada postaje važan sa sporim klijentskim aplikacijama i kada je potrebna sinkronizacija između različitih elemenata, kao što su senzori. Ako klijent radi presporo, a poslužitelj ne čeka, može doći do preljeva informacija. To znači da klijent neće moći učinkovito upravljati svim podacima, što može rezultirati gubitkom ili miješanjem podataka. Na sličan način, ako se koristi mnogo senzora u asinkronom načinu rada, postaje izazovno osigurati da svi senzori koriste podatke iz istog trenutka u simulaciji.

Nakon što se simulacija uskladi sinkronim načinom rada za prikupljanje podataka vozilo se autonomno vozi po različitim predefiniranim mapama u CARLA simulatoru uz različite vremenske uvjete s montiranom RGB kamerom i dubinskom kamerom. Svaka

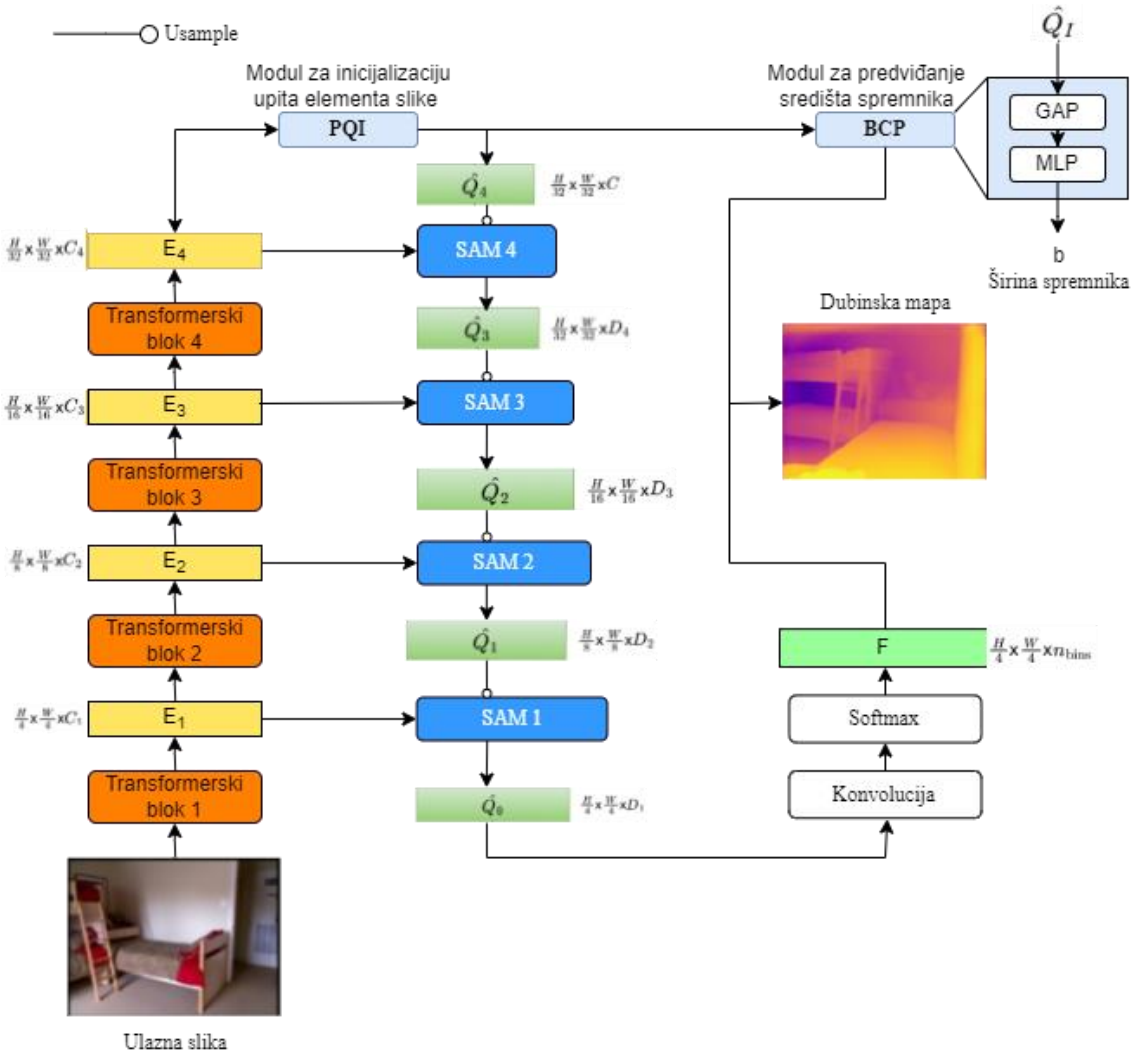
kamera pohranjuje podatke u svoj unaprijed određeni direktorij. Podaci su skupljeni tako da svaki direktorij sadrži po 24000 parova RGB slika i označenih dubinskih mapa. Prikupljanje podataka je izvršeno na pet različitih mapa definirane u API-u simulatora od *Town01* do *Town05*.

3.3. Opis arhitekture neuronske mreže za monokularnu procjenu dubinske mape

U ovom poglavlju predstavljen je neuronska mreža za procjenu monokularne dubinske mape koja se temelji na *PixelFormer* arhitekturi [7]. Ovaj pristup se temelji na nadziranom učenju putem enkoder dekoder arhitekture koji se sastoji od više komponenti kao što je prikazano na slici 3.7, uključujući transformerski blok, modul za inicijalizaciju upita elementa slike (engl. *Pixel Query Initialiser* - PQI), modul za predviđanje središta spremnika (engl. *Bin Center Predictor* - BCP) i modul za preskakanje pozornosti. PQI agregira globalne informacije iz ulazne slike kako bi inicijalizirao upite elementa slike, dok BCP prilagodljivo predviđa središta spremnika za diskretizaciju raspona dubine.

U kontekstu monokularnog određivanja dubine, spremnici (engl. *bins*) predstavljaju diskretne intervale na koje se dijeli kontunirani raspon dubina. Svaki spremnik obuhvaća određeni raspon dubina koje se koriste za predviđanje dubine pojedinih elemenata slike. Umjesto predviđanja dubinske mape za svaki element slike, model predviđa vjerojatnosti za svaki spremnik, a zatim koristi te vjerojatnosti za određivanje točnih dubinskih vrijednosti.

Konvolucijske jezgre imaju intrinzičnu lokalnu prirodu, što znači da su mape značajki u ranijim fazama konvolucijske mreže preciznije, ali im nedostaje globalno recepcijsko polje, zato ne mogu u potpunosti obuhvatiti cijelu scenu i kontekst. Zato se koristi mape značajki piramidalnog dekodera koja kombinira semantički bogate, nisko razlučive mape značajki dekodera s višom razlučivošću, ali semantički slabijim mapama značajki enkodera. To se postiže korištenjem veza za preskakanje (engl. *Skip connections*) koje povezuju mape značajki između enkodera i dekodera.

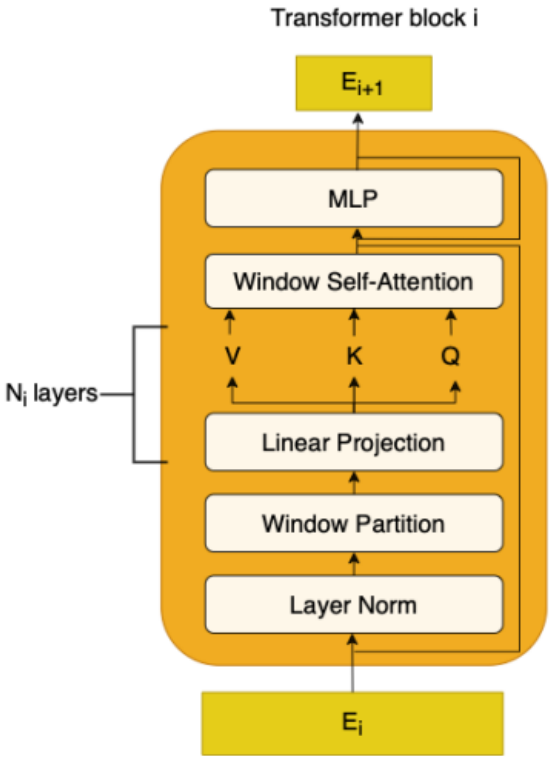


Slika 3.7: Detaljan prikaz PixelFormer arhitekture [7]

S obzirom na ulaznu sliku, enkoder koji se temelji na *swin* transformeru prvo izdvaja mape značajki. Mapa značajki s najgrubljom rezolucijom (E4) daje se kao ulaz u PQI modul. PQI modul proizvodi početne upite o elementima slike koji se daju kao ulaz BCP modulu koji proizvodi širine spremnika. Početni upiti o elementima slike zatim se pročišćavaju na više rezolucije pomoću SAM modula. Konačno, primjenjuje se operacija konvolucije praćena *softmax* funkcijom kako bi se dobila distribucija vjerojatnosti po elementu slike preko središta spremnika.

Mnogi noviji radovi za procjenu dubinske mape koristili su arhitekture temeljene na samopozornosti (engl. *self-attention*) zbog nedavnog uspjeha transformera u drugim područjima. Samopozornost omogućuje modelima da povećaju receptivno polje i

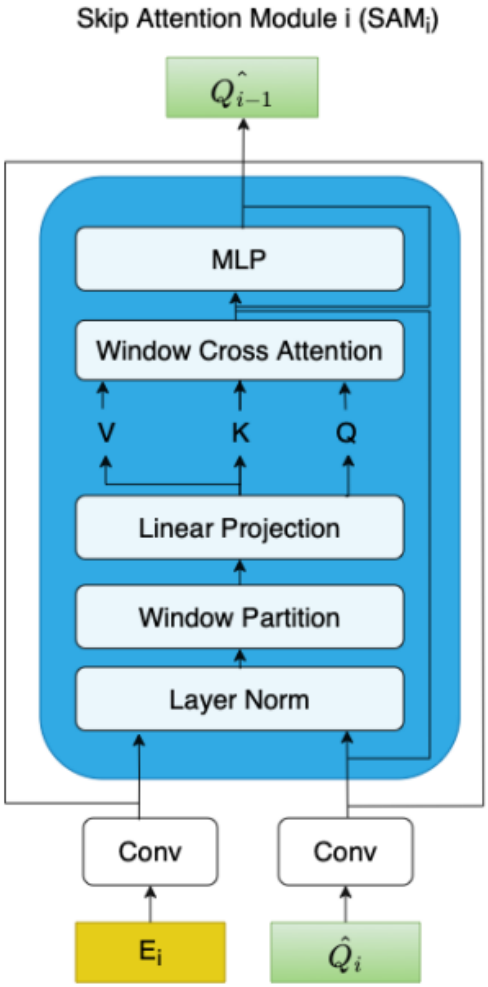
omogućava računanje ovisnosti u mapama značajki na velikim udaljenostima, što je korisno za razumijevanje globalnog konteksta. Međutim, praktična primjena samopozornosti na visokorazlučive mape značajki može biti izazovna zbog visokih memorijskih i računalnih zahtjeva. Trenutni najnapredniji pristup rješava ovaj izazov primjenom samopozornosti temeljene na prozorima koristeći temeljnu strukturu *swin* transformera prikazanu na slici 3.8.



Slika 3.8: Blok *swin* transformera [7]

Jedan ključni aspekt monokularne procjene dubinske mape je precizno poravnavanje granica dubine s granicama objekata na slici. Iako su postojeće metode pokazale visoku točnost u tom poravnanju, još uvijek se suočavaju s izazovima u ispravnom dodjeljivanju dubinskih oznaka pojedinačnim elementima slike. Ova zabuna u označavanju dubine pripisuje se nemogućnosti trenutnih metoda da učinkovito spajaju lokalne mape značajki visoke rezolucije iz enkodera i globalne kontekstualne mape značajke iz dekodera [7].

Kako bi se prevladala ova ograničenost, uvodi se novi modul za preskakanje pozornosti kako bi se poboljšala fuzija informacija kao što je prikazano na slici 3.7. SAM izračunava sličnost između upita elementa slike na temelju mapa značajki dekodera i njihovih odgovarajućih susjeda iz mapa značajki enkodera unutar predefiniranog prozora kako bi pratio i agregirao informacije na većem rasponu. SAM omogućuje modelu da zabilježi dugotrajne ovisnosti i korelacije na velikim udaljenostima, što poboljšava sposobnost modela da predvidi ispravne dubinske oznake za svaki element slike. Detaljan prikaz arhitekture modula za preskakanje pozornosti može se vidjeti na slici 3.9.



Slika 3.9: Arhitektura modula za preskakanje pozornosti [7]

Nedavne metode procjene dubine [8] definiraju problem kao kombinaciju klasifikacije i regresije, u kojem je dubina predviđena linearnom kombinacijom središta

spremnika diskretiziranih u dubinskom rasponu. Središta spremnika se adaptivno predviđaju za svaku sliku, što omogućuje mreži da se koncentrira na područja raspona dubine koja su vjerojatnija za scenu ulazne slike. Za generiranje središta spremnika obično se koristi *swin* transformer koji agregira globalne informacije iz izlaza drugog modela transformera temeljenog na enkoder dekoder modelu.

Na temelju radova [8, 15], monokularnu procjenu dubinske mape modeliramo kao kombinaciju klasifikacije i regresije. Na temelju ulazne slike *I*, mreža predviđa širine spremnika *b*, koji diskretiziraju kontinuirani raspon dubine u broj intervala *nbins.* Spremnici se prilagodljivo predviđaju za svaku sliku. Konačni vektor vjerojatnosti dimenzija tretira se kao vektor težina, a dubina d_i, za element slike *i*, računa se kao linearna kombinacija vjerojatnosti na elementu slike s predviđenim središtima spremnika za pojedinu sliku.

Pregled arhitekture: ulazna slika *I* prvo se šalje *swin* transformeru [12], koji koristi više slojeva pozornosti temeljene na prozorima za ekstrakciju mapa značajki koje predstavljaju sliku u odnosu na skaliranje razlučivosti $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ u odnosu na *I* (Slika 3.7). Mape značajki imaju globalno receptivno polje zbog prirode *swin* transformera. Mape značajki na razlučivosti $\frac{1}{32}$ se zatim šalju u predloženi modul za inicijalizaciju upita elemenata slike. Modul PQI agregira informacije o cijeloj sceni koristeći višeskalno globalno prosječno grupiranje kako bi inicijalizirao upite elemenata slike. Upiti elemenata slike se hijerarhijski usavršavaju sa mapama značajki enkodera koristeći predloženi modul za preskakanje pozornosti implementiran u različitim fazama kako bi se predvidjela vjerojatnosna distribucija po središtima spremnika za svaki element slike [7]. Inicijalizirani upiti elemenata slike se također šalju modulu za predviđanje središta spremnika. On adaptivno predviđa središta spremnika po slici koristeći globalno prosječno grupiranje, nakon čega slijedi višeslojni perceptron (engl. *multi-layer perceptron* - MLP) kao što je vidljivo na slici 3.7.

Modul za inicijalizaciju upita elemenata slike agregira globalne informacije scene u svaki ugrađeni element slike. Mape značajki slike s najgrubljom razlučivošću, koje sadrže najvažnije detalje u sceni, šalju se kao ulaz u modul za inicijalizaciju upita elemenata slike. Podaci ulazne mape značajki veličine $\frac{H}{32}$ x $\frac{W}{32}$ x C4, koriste piramidalno

prostorno grupiranje (engl. *Pyramid spatial pooling* - PSP) [15] s prilagodljivim globalnim grupiranjem u mjerilima 1, 2, 3 i 6. Mape značajki se zatim uzorkuju do razlučivosti $\frac{1}{32}^{th}$ i ulančavaju se. Tada se provodi konvolucijska operacija kako bi se učinkovito integrirale globalne informacije, kako bi se dobili početni upiti elemenata slike Q_I veličine $\frac{H}{32}$ x $\frac{W}{32}$ x C, gdje je C = 512.

Prethodni radovi [8] koristili su *swin* transformer za predviđanje središta spremnika koji diskretiziraju dubinsku mapu u fiksni broj intervala. *Swin* transformer dijeli mapu značajki slike na dijelove od 16x16 elementa slike i koristi slojeve samopozornosti kako bi razmijenio informacije među tim dijelovima. Prvo ugrađivanje prolazi kroz glavu višeslojnog *perceptrona* kako bi predvidio središta spremnika. Umjesto dekodiranja mape značajki slike na visoku razlučivost i zatim korištenja *swin* transformera, iskoristit će početne upite elemenata slike za predviđanje središta spremnika. Osim što je učinkovitije, predloženi dizajn pomaže u ugradnji informacija o dubini u upite elemenata slike putem izravnog nadgledanja stvarnih vrijednosti dubine. Modul za predviđanje spremnika sastoji se od jednostavnog globalnog prosječnog grupiranja (engl. *Global Average Pooling* - GAP), a zatim slijedi višeslojni *perceptron* za predviđanje širine spremnika b dimenzije n_{bins}. Ovdje n_{bins} označava broj prilagodljivih spremnika po slici. U modelu se koristi n_{bins} = 256 kako je predloženo u [8]. Danim upitima elemenata slike Q veličine $\frac{H}{32}$ x $\frac{W}{32}$ x C, predviđa se:

$$b = \mathrm{MLP}\big(\mathrm{GAP}(Q)\big) \qquad (3-2)$$

na kraju, središta spremnika za ulaznu sliku se izračunavaju po (3-2) za $i \in \{1, ..., n_{bins}\}$:

$$c(b_i) = d_{min} + (d_{max} - d_{min})(\frac{b_i}{2} + \sum_{j=1}^{i-1} b_j) \qquad (3-3)$$

Za stvaranje dubinske mape, semantičke mape značajki grube razine i detalji fine razine slike ključni su za točnu procjenu dubinske mape. Stoga, slično kao i u prethodnim radovima [16, 17], koristi se pristup odozdo prema gore koji počinje sa mapom značajki

najniže razlučivosti, povećava je i umetne fini detalj iz mape značajke enkodera na određenoj razini koristeći modul za preskakanje. Za razliku od operacije konvolucije za spajanje mape značajki enkodera i dekodera gdje težine jezgre nisu prilagodljive prema lokaciji elemenata slike, koristi se modul preskakanje pažnje koji koristi sličnost između upita elemenata slike i odgovarajuće mape značajki enkodera kako bi učinkovito spojio globalne i lokalne značajke [7].

Izvedba SAM modula uključuje davanje mape upita elemenata slike $\hat{Q}_i$ odgovarajuće mape značajki enkodera E_i za određenu razinu i, prvo se izvodi 3x3 konvolucija E_i s D_i kanala i na E_i i $\hat{Q}_i$ kako bi se broj kanala upita elemenata slike generiranih iz mape značajki dekodera podudarao s brojem kanala u mapama značajki enkodera. Nakon operacije konvolucije za izlaz se dobiva matrica upita Q iz $\hat{Q}_i$, a matrice ključeva K i vrijednosti V dobivaju se iz E_i koristeći težine W_q, W_k i W_v implementirane pomoću MLP slojeva. Budući da nije računalno izvedivo da upit q_i koji odgovara upitu elemenata slike na lokaciji i pristupa svim ključevima matrici K, pažnju se ograničava na prozor, kao što je predloženo za *swin* transformer [8]. Matrice Q, K i V se prvo dijele u prozore veličine WxW. Kao što je navedeno u [8], koristi se W=7. Neka su Q_w, K_w i V_w upit, ključ i vrijednost koji odgovaraju elementima slike u prozoru w. Izlaz se računa na sljedeći način:

$$\text{Attention}(Q, K, V) = \text{Rearrange}(\text{Softmax}(Q_w K_w^T + B)V_w) \qquad (3-4)$$

ovdje B označava relativni položajni odmak. B je matrica veličine $w^2 \; x \; w^2$ koja predstavlja ugrađivanje relativnog položaja za svaki par upita i ključeva. Pažnja se računa za svaki prozor w, nakon čega operacija preuređivanja raspoređuje prozore prema njihovim odgovarajućim prostornim lokacijama u Q.

Kako bi se ugradile informacije o različitim rasponima dubina, svaki upit elementa slike je podijeljen u H_i glava, a operacija pažnje se primjenjuje za svaku glavu. Nakon pažnje, ugrađivanja po dubini elementa slike agregiraju se pomoću MLP slojeva. Preostale veze nakon pažnje i MLP slojeva dodaju se za glatke gradijente, ako su zadani $\hat{Q}_i$ i E_i za upit elemenata slike i mape značajki enkodera na razini i [7]:

$$\bar{Q}_i = \text{LayerNorm}(\hat{Q}_i)$$
$$\bar{E}_i = \text{LayerNorm}(E_i)$$
$$Q = W_Q \bar{Q}_i, K = W_K \bar{E}_i, V = W_V \bar{E}_i$$
$$\hat{Q}_{i-1} = \text{MultiheadAttention}(Q, K, V) + \hat{Q}_i \qquad (3-5)$$
$$\hat{Q}_{i-1} = \text{MLP}(\hat{Q}_{i-1}) + \hat{Q}_{i-1}$$
$$\hat{Q}_{i-1} = \text{MLP}(\hat{Q}_{i-1}) + \hat{Q}_{i-1} + E_i$$

Koristi se D_1, D_2, D_3, D_4 = {128, 256, 512, 1024} gdje D_i odgovara broju kanala u konvolucijskoj jezgri koja se primjenjuje prije pažnje temeljene na prozorima za fazu *i*. Broj glava H_1, H_2, H_3, H_4 = {4, 8, 16, 32}, gdje H_i predstavlja broj glava pažnje korištenih u modulu za preskakanje pažnje na razini *i*. Navedeno je prikazano na slici 3.7.

Arhitektura dekodera prikazana na slici 3.7, započinje prvim upitom elemenata slike $\hat{Q}_i$ dobivenih izlazom iz modula za inicijalizaciju upita elementa slike. $\hat{Q}_i$ se uzorkuje na dvostruku veličinu razlučivosti pomoću *Pixel Shuffle* [18] i predaje se kao ulaz modulu za preskakanje pažnje zajedno s odgovarajućim mapama značajki enkodera E_4. Početni upiti elemenata slike se fino usavršavaju do finih razlučivosti prisustvom mrežnih mapa značajki enkodera na različitim razlučivostima putem modula za preskakanje pažnje. Za dani upit elemenata slike na razini $\hat{Q}_i$ i odgovarajuće mape značajki enkodera E_i,

$$\hat{Q}_i = \text{SAM}(\text{Upsample}(\hat{Q}_{i+1}, E_{i+1}), \qquad i \in \{0, 1, 2, 3\} \qquad (3-6)$$

provedena je i konvolucijska operacija na $\hat{Q}_0$ kako bi se dobila konačna ugrađena dubinska mapa značajki *F* veličine $\frac{H}{32}$ x $\frac{W}{32}$ x n_{bins}. Na kraju se primjenjuje operacija *softmax* elemenata slike kako bi se dobila vjerojatnosna distribucija po spremnicima p_{bins}:

$$p_{bins} = \text{Softmax}\left(\text{Conv}(\hat{Q}_0)\right) \qquad (3-7)$$

Konačna dubina se predviđa linearnom kombinacijom središta spremnika sažetim vrijednostima vjerojatnosti:

$$d_i = \sum_{k=1}^{n_{bins}} c(b_k) p_{ik} \qquad (3-8)$$

gdje je d_i predviđena dubina na elemenata slike i, $c(b_k)$ je središte spremnika na k-tom mjestu, n_{bins} je broj spremnika, a p_{ik} je vjerojatnost za središte spremnika k za elemenat slike i [7].

3.4. Treniranje neuronske mreže za procjenu dubinske mape na prikupljenim podacima iz simulatora

Trenirane su dvije neuronske mreže. Prva mreža (KITTI) je mreža za procjenu dubinske mape koja je trenirana na KITTI skupu podataka. Druga mreža (KITTI_TL) je mreža dobivena prijenosnim učenjem, to jest KITTI mreža koja je dotrenirana na prikupljenim simulacijskim podacima. Za treniranje mreže KITTI_TL na prikupljenim simulacijskim podacima potrebno je učitati podatke pomoću učitavača podataka (engl. *Dataloader*) iz skripte *dataloader.py* [19] koja se nalazi unutar koda *pixelformer* arhitekture. Skripta učitava podatke za RGB slike i odgovarajuće dubinske mape iz odgovarajućeg skupa podataka. Kod koristi biblioteke PyTorch i torchvision za rukovanje podacima i transformacije slika, te podržava tri različita načina rada: *train*, *test* i *online* evaluaciju. Prilikom treninga učitava podatke iz označene datoteke i primjenjuje metode augmentacije podataka poput nasumičnog okretanja, povećavanje svjetline i boje podataka. Kod implementira prilagođenu *ToTensor()* funkciju koja pretvara RGB sliku i dubinsku mapu u PyTorch tenzore i na njih primjenjuje normalizaciju.

Treniranje predloženog algoritma napravljeno je u skripti *train.py* pomoću unaprijed definirane instance modela *PixelFormera* kojoj se predaju RGB slike spremljene u *image* varijablu kao što je prikazano na tablici 3.1.

Tablica 3.1: Isječak koda iz *train.py* skripte za petlju treniranja modela dubokog učenja

```
while epoch < args.num_epochs:
    for step, sample_batched in enumerate(dataloader.data):
        optimizer.zero_grad()
        before_op_time = time.time()
        image = torch.autograd.Variable(sample_batched['image'].cuda(args.gpu,
non_blocking=True))
        depth_gt = torch.autograd.Variable(sample_batched['depth'].cuda(args.gpu,
non_blocking=True))
        depth_est = model(image)
```

Varijabla depth_est predstavlja predviđenu dubinsku mapu dobivenu pomoću neuronske mreže.

Korištena funkcija gubitka za trening proces je funkcija koja je invarijantna na skaliranje uz logaritamsku transformaciju (engl. *Scale Invariant log loss* – SIlog). Za svaki element slike *i*, izračunava se prvo logaritamska udaljenost g_i između predviđene dubinske mape d_i i stvarne dubinske mape d^*_i kao:

$$g_i = \log(\hat{d}_i) - \log(d_i^*) \qquad (3-9)$$

Zatim se *SIlog* funkcija gubitka izračunava na sljedeći način:

$$L_{SILog} = \alpha \sqrt{\frac{1}{n}\sum_i g_i^2 - \frac{\lambda}{n^2}\left(\sum_i g_i\right)^2} \qquad (3-10)$$

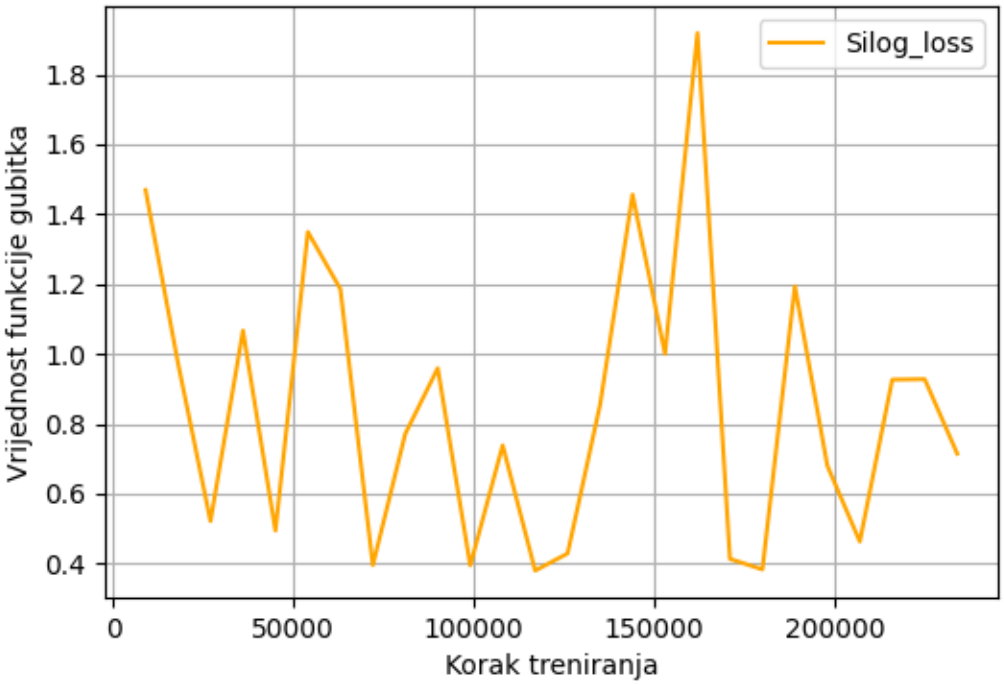
Gdje, n označava broj elemenata RGB slike koji imaju odgovarajuće vrijednosti u pripadnoj dubinskoj mapi. Sljedeći [17], za sve eksperimente korišten je $\lambda = 0.85$ i $\alpha = 10$.

U tablici 3.2. prikazan je dio koda *train.py* skripte koji je implementacija funkcije gubitka koja uspoređuje izlaz modela i označenu dubinsku mapu.

Tablica 3.2: Isječak koda iz *train.py* skripte za izračunavanje funkcije gubitka

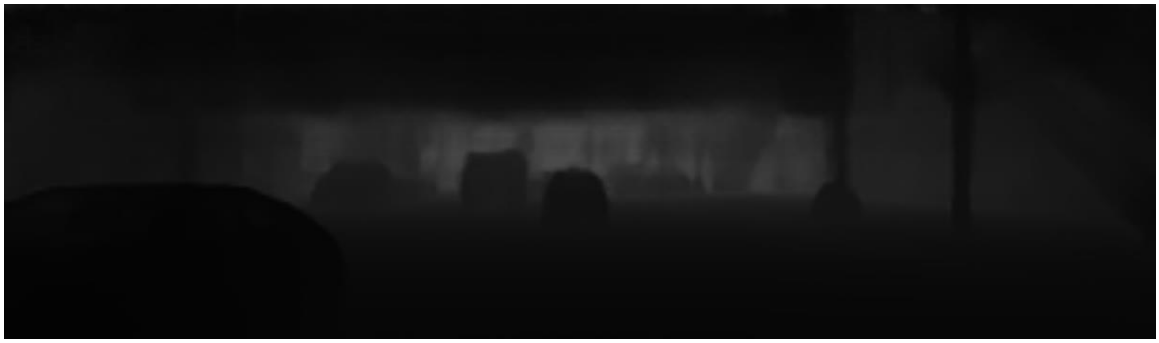
```
mask = depth_gt > 1.0
loss = silog_criterion.forward(depth_est, depth_gt, mask.to(torch.bool))
loss.backward()
for param_group in optimizer.param_groups:
    current_lr = (args.learning_rate - end_learning_rate) * (1 - global_step / num_total_steps) ** 0.9 + end_learning_rate
    param_group['lr'] = current_lr
optimizer.step()
```

Mreža je istrenirana na sveukupno 24000 parova RGB slika i dubinskih mapa koje su prikupljene u simulatoru kako je objašnjeno u potpoglavlju 3.2.2. Trenirana je na 35 epoha sa duljinom 4 niza podatka (engl. *batch size*). Na slici 3.10. je tijek funkcije gubitka tijekom treniranja neuronske mreže. Model dobiven u zadnjoj epohi treniranja odabran je kao finalni model.



Slika 3.10: Tijek funkcije gubitka tijekom treniranja mreže

Na slikama 3.11 i 3.12 dani su primjeri predviđenih dubinskih mapa pomoću neuronske mreže KITTI i KITTI_TL na temelju jedne RGB slike iz sintetičkog skupa podataka koji je korišten u procesu prijenosnog učenja.


Slika 3.11: Predviđena dubinska mapa istrenirane mreže KITTI

Slika 3.12: Predviđena dubinska mapa istrenirane mreže KITTI_TL

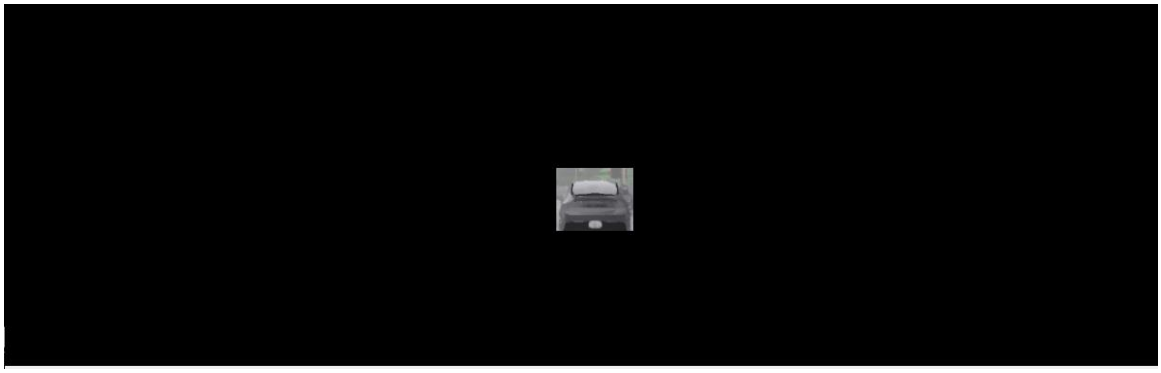
3.5. Izrada algoritma za automatsko kočenje na temelju detekcije prepreke iz dubinske mape

U ovom potpoglavlju opisan je algoritam za automatsko kočenje vozilom koji svoj rad temelji na procjeni dubinske mape pomoću mreže koja je opisana u potpoglavlju 3.4. Nadalje, ideja je detektirati prepreku (drugo vozilo, pješak i sl.) u dubinskoj mapi te u pravom trenutku započeti proces kočenja kako bi se vozilo zaustavilo na sigurnoj udaljenosti od prepreke.

Za potrebe izrade algoritma za kočenje napravljena je python skripta *driving_depth_pred.py* u kojoj se stvara novi klijent za CARLA simulator. U ovoj skripti učitava se istrenirana neuronska mreža koja tijekom izvođenja simulacije predviđa dubinsku mapu na temelju RGB slike dobivene s prednje strane vozila na osnovu koje će se razviti algoritam za kočenje.

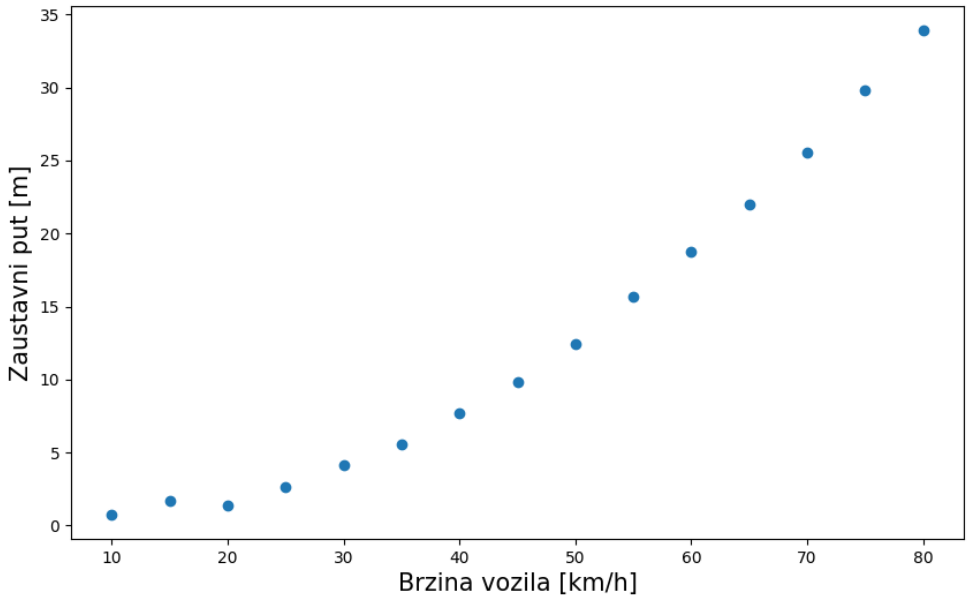
Algoritam za automatsko kočenje kao izvor informacija koristi procijenjenu dubinsku mapu. Budući da je potrebno zaustaviti vozilo kada se prepreka nađe na putu vozila, iz procijenjene dubinske mape izdvaja se regija od interesa (engl. *Region of Interest* - RoI) što je zapravo dio slike koji odgovara prostoru ispred vozila. Primjer RGB slike dobivene pomoću RGB kamere montirane na prednjoj strani vozila prikazana je na slici 3.13 dok je odgovarajući RoI prikazan na slici 3.14. Analizom RoI-a u odgovarajućoj dubinskoj mapi moguće je utvrditi nalazi li se prepreka u pravcu kretanja vozila.

Slika 3.13: Primjer RGB slike dobivene pomoću kamere montirane na prednjoj strani vozila u simulatoru



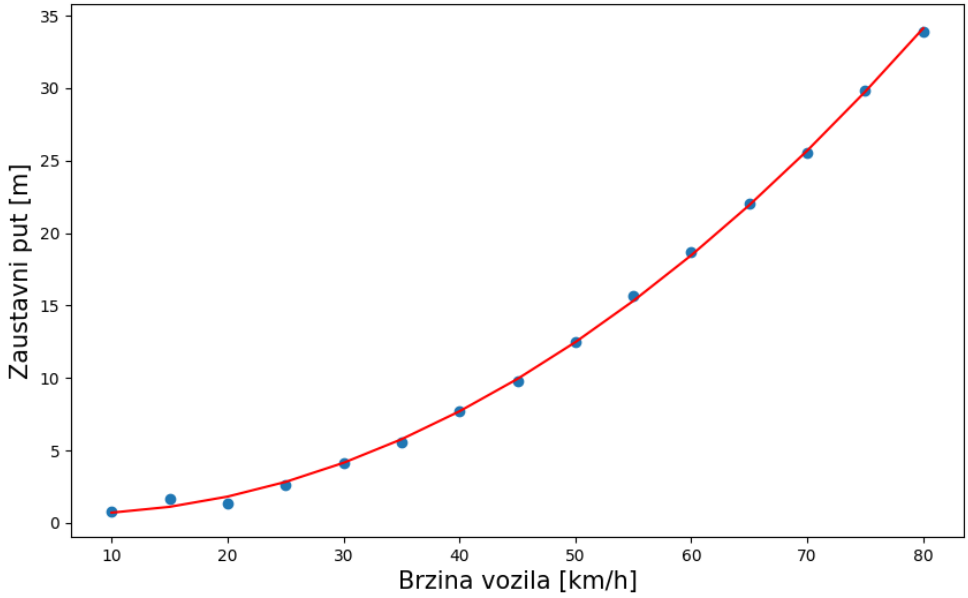
Slika 3.14: Izdvojena regija interesa za sliku 3.13.

Za izrađivanje algoritma automatskog kočenja potrebno je izračunati zaustavni put vozila pri različitim brzinama. Iz tog razloga su u okviru CARLA simulatora napravljeni eksperimenti kojima je izmjeren zaustavni put vozila i to za brzine od 10km/h do 80 km/h u koracima po 5 km/h. Pri tome je korištena jačina pritiska na papučicu kočnice u iznosu od 90%. Na slici 3.15. prikazana su dobivena mjerenja.



Slika 3.15: Mjerenja zaustavnog puta u ovisnosti o brzini vozila dobivena u CARLA simulatoru

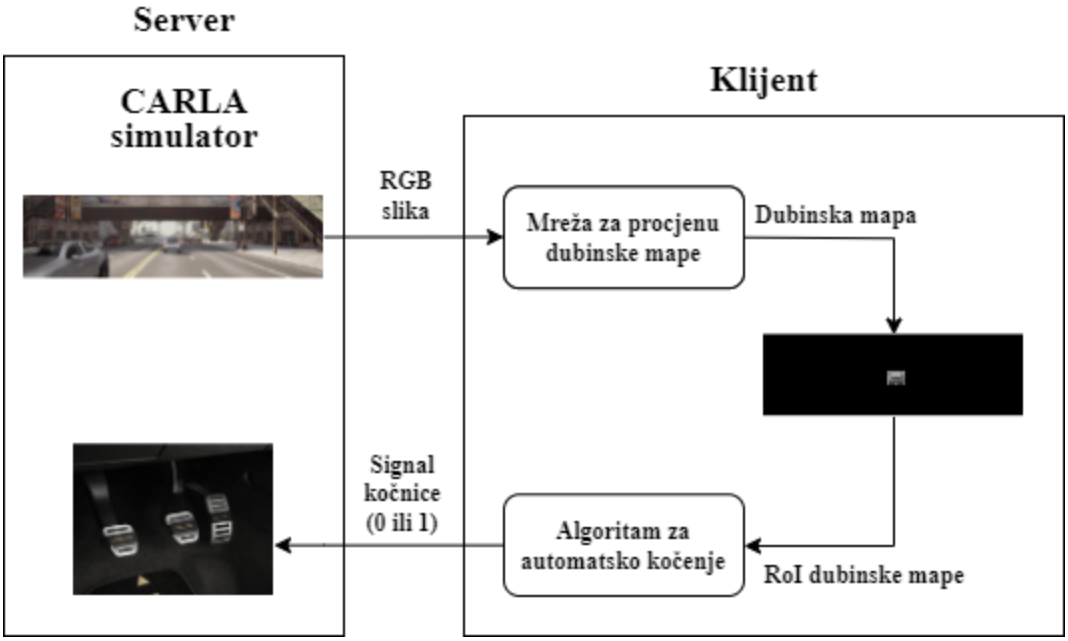
Na podatke za zaustavni put primijenjena je polinomska regresija kako bi se procijenila funkcijska ovisnost između brzine kretanja vozila i zaustavnog puta. Polinomska regresija provedena je korištenjem klase *PolynomialFeatures* iz popularne biblioteke za strojno učenje scikit-learn [20] za stvaranje polinomskih mapa značajki iz izvornih mapa značajki. Potrebno je odabrati odgovarajući red polinoma, koji je u ovom slučaju kvadratni, a koji se može odrediti na temelju kompleksnosti odnosa u podacima. Na temelju polinomskih značajki procijenjen je model linearne regresije pomoću ugrađene funkcije *fit()*. Dobivena funkcionalna ovisnost može se koristiti za procjenu zaustavnog puta na temelju trenutne brzine vozila pomoću ugrađene funkcije *predict()*. Dobivena funkcija ovisnost prikazana je na slici 3.16. zajedno s izmjerenim podacima na temelju kojih je izvršena procjena funkcionalne ovisnosti.



Slika 3.16: Dobivena funkcionalna ovisnost između brzine vozila i zaustavnog puta u CARLA simulatoru

Algoritam za automatsko kočenje mora zaustaviti vozilo na sigurnoj udaljenosti od prepreke. U ovom radu željena udaljenost između (zaustavljenog) vozila i prepreke je postavljena na 3 metra. Ako se uzme u obzir mjesto montaže RGB kamere na vozilu (približno 1 metar od prednjeg kraja vozila), tada se u slučaju detekcije prepreke u dubinskoj mapi mora uzeti u obzir dodatnih 4 metra na duljinu zaustavnog puta prilikom izračuna kada započeti kočenje vozilom. Na primjer, iz slike 3.16. vidi se da je pri brzini od 75 km/h duljina zaustavnog puta približno 30 metra. To znači da kočenje mora započeti ako se detektira prepreka u dubinskoj mapi na udaljenosti jednakoj (ili manjoj) od 34 metra. Pri tome je ključno obrađivati sliku po sliku iz RGB kamere montirane na vozilo kako bi se na vrijeme detektirala prepreka i započelo kočenje.

Primijenjeni princip detekcije prepreke je prilično jednostavan. Ako barem jedan element slike u procijenjenoj dubinskoj mapi u području regije od interesa poprimi vrijednost koja je manja ili jednake od definirane, vozilo će započeti s automatskim kočenjem. Na slici 3.17 prikazan je tok rada algoritma za automatsko kočenje.



Slika 3.17: Tok rada algoritma za procjenu dubinske mape na temelju slike dobivene s jedne kamere i automatsko kočenje u okviru CARLA simulatora

4. EVALUACIJA IZRAĐENOG ALGORITMA ZA PROCJENU DUBINSKE MAPE I AUTOMATSKO KOČENJE U CARLA SIMULATORU

Predloženi algoritam za procjenu dubinske mape na temelju monokularne slike i automatsko kočenje evaluiran je na dva načina. Najprije je evaluirana efikasnost procjene dubinske mape iz RGB kamere montirane na vozilu u simulatoru, a zatim je evaluirano automatsko kočenje vozila mjerenjem udaljenosti na kojoj se vozilo zaustavilo u različitim mapama s obzirom na različite prepreke.

4.1. Evaluacija izgrađene neuronske mreže za procjenu dubinske mape

Evaluacija izgrađene neuronske mreže izvršena je usporedbom procijenjenih dubinskih mapa sa stvarnim vrijednostima dubine (engl. *ground truth*) pomoću nekoliko različitih metrika na testnim podacima dobivenim u simulatoru.

Neka je procijenjena dubinska mapa D, a odgovarajuća stvarna dubinska mapa označena s D^*. Neka D_i i D_i^* predstavljaju procijenjene i stvarne dubinske vrijednosti na elementu slike redom indeksirane s i, a N predstavlja ukupan broj elemenata slike za koje postoje i valjani elementi slike stvarne i procijenjene dubine. Što se tiče kvantitativne usporedbe procijenjene dubinske mape i stvarnih vrijednosti, uobičajene metrike evaluacije su sljedeće:

- Apsolutna relativna razlika (*Abs Rel*) definirana je kao prosječna vrijednost L_1 udaljenosti između stvarne i procijenjene dubine, skalirane procijenjenom dubinom, za sve elemente slike:

$$Abs\ Rel = \frac{1}{N}\sum_{N}\frac{|D_i^* - D_i|}{D_i} \qquad (4-1)$$

- Kvadrat relativne pogreške (*Sq Rel*) definirana je kao prosječna vrijednost L_2 udaljenosti između stvarne i procijenjene dubine, skalirane procijenjenom dubinom, za sve elemente slike:

$$Sq\ Rel = \frac{1}{N}\sum_{N}\frac{|D_i^* - D_i|^2}{D_i} \qquad (4-2)$$

- Srednja kvadratna pogreška (*RMSE*) definirano je kao:

$$RMSE = \sqrt{\frac{1}{N}\sum_{N}|D_i^* - D_i|^2} \qquad (4-3)$$

- Logaritamsko srednje kvadratno odstupanje (*RMSE log*) definirano je kao:

$$RMSE\ log = \sqrt{\frac{1}{N}\sum_{N}|log\ D_i^* - log\ D_i|^2} \qquad (4-4)$$

- Točnost unutar praga je postotak predviđenih elemenata slike u kojima je relativna pogreška unutar zadanog praga. Formula je prikazana kao:

$$max\left(\frac{D_i}{D_i^*}, \frac{D_i^*}{D_i}\right) = \delta < prag \qquad (4-5)$$

Vrijednosti praga obično su postavljene na 1.25, 1.25^2 i 1.25^3 i označava se s δ.

- Također, *Eigen* [21] dizajniraju mjeru za skalabilnu grešku kako bi mjerili odnose između točaka u sceni, neovisno o apsolutnoj globalnoj skali. Srednja kvadratna pogreška skalabilnosti u logaritamskom prostoru definirana je jednadžbom:

$$E(D, D^*) = \frac{1}{2N}\sum_{i=1}^{N}(\log\ D_i - \log\ D^*{}_i + a(D, D^*))^2 \qquad (4-6)$$

gdje je:

$$a(D, D^*) = \frac{1}{N}\sum_i (log\ D^*{}_i + log\ D_i) \qquad (4-7)$$

vrijednost α koja minimizira razliku za određeni *(D, D^*)*. Za svaku procjenu *D*, err^a je skala koja ju najbolje poravna sa stvarnim vrijednostima, a err je razlika između *D* i D^* u logaritamskom prostoru. S obzirom na to da svi skalarni umnošci od D^* imaju istu grešku, skala je invarijantna [6].

U CARLA simulatoru prikupljeno je 1000 RGB slika i pripadnih dubinskih mapa dobivenih pomoću dubinske kamere za potrebe testiranja mreže za procjenu dubinske mape. Slike su prikupljene unutar pet različitih mapa definirane u API-u simulatora od *Town01* do *Town05*. Testirane su dvije mreže. Prva mreža (KITTI) je mreža za procjenu dubinske mape koja je trenirana na KITTI skupu podataka. Druga mreža (KITTI_TL) je mreža dobivena prijenosnim učenjem, to jest KITTI mreža koja je dotrenirana na prikupljenim simulacijskim podacima. U tablici 4.1. dani su postignuti rezultati obje mreže na 1000 testnih podataka iz simulatora.

Evaluacija je izvršena na način da je uspoređena vrijednost označene dubinske mape s vrijednosti predviđene dubinske mape. Prije obavljene evaluacije parovi RGB slika i dubinskih mapa su izrezane po uzoru na *Garg crop* [22] gdje su uklonjeni manje važni dijelovi slike za koje nije bitna vrijednost dubinske mape, kao npr. nebo koje onda neće ulaziti u rezultat evaluacije.

Tablica 4.1: Evaluacija izgrađenih mreža na 1000 podataka iz CARLA simulatora

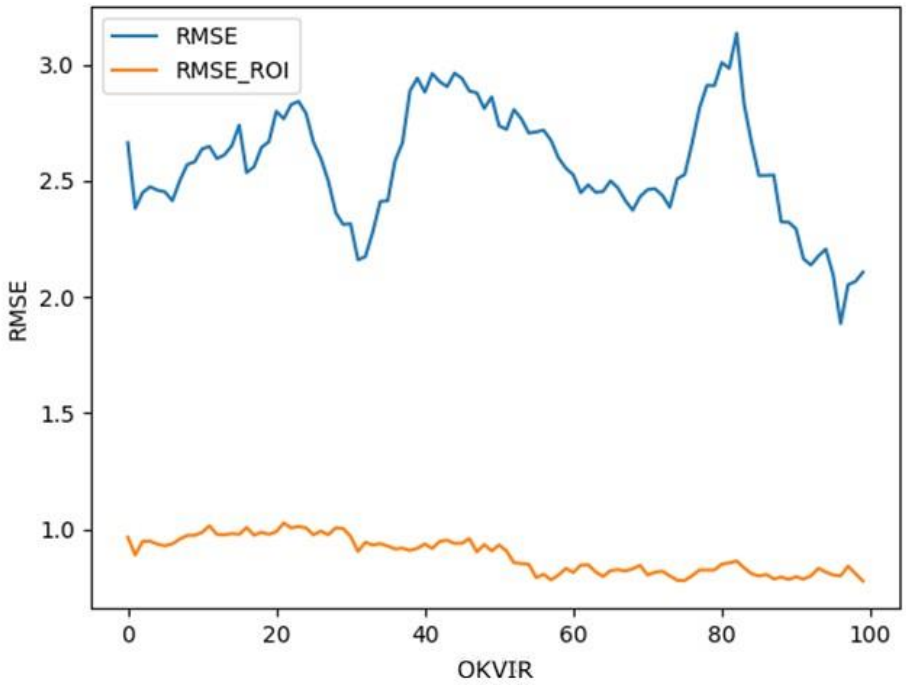
Mreža	δ 1	δ 2	δ 3	AbsRel	SqRel	RMSE	RMSElog	SILog	log10
KITTI	0.531	0.857	0.928	0.277	2.237	9.016	0.335	19.629	0.120
KITTI_TL	0.956	0.988	0.993	0.072	0.304	3.326	0.112	9.913	0.029

Vrijednosti δ1, δ2, δ3 su bolje što su veće inače su preko 0.9 dobre vrijednosti, a ostale vrijednosti su bolje sa što manjom vrijednosti. Rezultati koji se dobiju mrežom koja je dotrenirana na simulacijskim podacima su značajno bolji prema tablici 4.1. Npr. korijen iz srednje kvadratne pogreške RMSE je gotovo 3 puta manji.

Postignuti rezultati u tablici 4.1. sugeriraju kako je korijen iz srednje kvadratne pogreške u procjeni dubine na temelju RGB slike oko 3 metra za model koji je dotreniran

na simulacijskim podacima. Ovakve performanse u procjeni dubine nisu dovoljno dobre za primjenu algoritma za automatsko kočenje. Iz tog razloga se pristupilo dodatnoj analizi procjeni dubine u simulatoru na razini RGB slike.

Na sekvenci od 100 uzastopnih slika koje su dobivene vožnjom vozila u simulatoru izračunata je RMSE za područje koje se dobiva *Garg Cropom*, ali i za područje regije od interesa (RoI) kako je definirano u potpoglavlju 3.5. Dobivena RMSE na razini svake slike prikazana je na slici 4.1. Uočava se kako se RMSE za ovu konkretnu sekvencu u slučaju *Garg cropa* kreće oko približno 2.5 metra dok se RMSE za ovu sekvencu u slučaju primjene RoI-a kreće gotovo ispod 1 metar. Ovo sugerira da mreža prilično točno procjenjuje dubinu u području ispred vozila (RoI).



Slika 4.1: Prikaz RMSE na razini jedne slike u sekvenci od 100 slika

Navedeni izračun RMSE je pokrenut nekoliko puta na različitim mapama i u različitim vremenskim uvjetima i dobiveni grafovi uvijek su bili vrlo slični grafu prikazanom na slici 4.1. Stoga je zaključeno kako je korijen iz prosječne kvadratne pogreške u procjeni dubine za RoI gotovo uvijek ispod 1 metar što je dovoljno dobro za razvoj algoritma za automatsko kočenje uz predefinirani željeni razmak između zaustavljenog vozila i prepreke u iznosu od 3 metra.

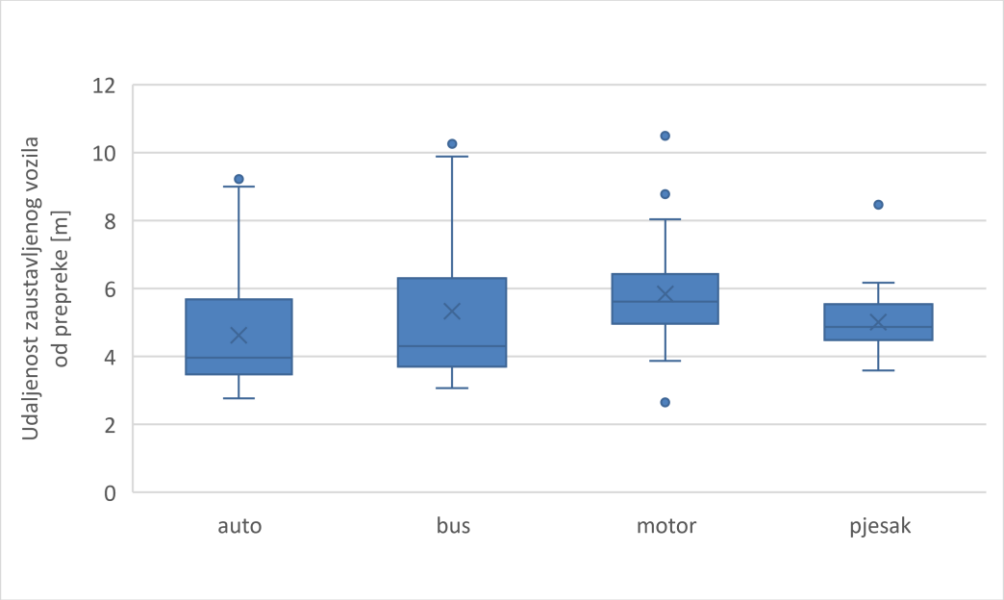
4.2. Evaluacija algoritma za automatsko kočenje vozila

Za potrebe testiranja algoritma za automatsko kočenje vozila izgrađena je vlastitita skripta *driving_depth_pred.py*. U ovoj skripti vozilo se autonomno vozi po određenoj mapi u CARLA simulatoru uz određene vremenske uvjete pri čemu je na određenoj lokaciji prisutna prepreka na kolniku. Skripta je pokrenuta ukupno 1000 puta, na pet različitih mapa, gdje je svaka mapa obrađena 200 puta. Na svakoj mapi algoritam za automatsko kočenje je testiran na četiri različite prepreke: automobil, motocikl, autobus i pješak, gdje se svaka prepreka pojavila 50 puta. Sve prepreke su postavljene stacionarno na cesti ispred vozila. Također, algoritam za automatsko kočenje testiran je na različitim brzinama vozila: 20, 40, 50, 60 i 70 km/h, gdje se svaka brzina pojavila 20 puta. Na svakoj mapi algoritam je testiran na različitim vremenskim uvjetima koji uključuju vedro, oblačno i kišovito vrijeme. Od toga dvije mape su pokrenute na vedrim vremenskim uvjetima, druge dvije mape su pokrenute na kišovitim vremenskim uvjetima, a posljednja mapa na oblačnim vremenskim uvjetima.

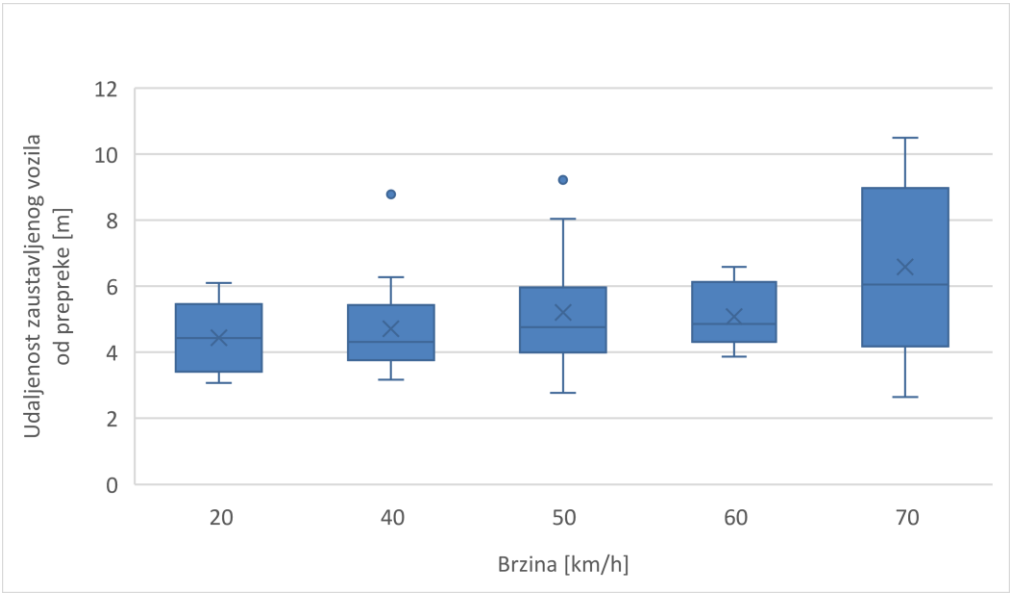
Na slici 4.2 prikazana je distribucija dobivenih mjerenja odnosno udaljenosti zaustavljenog vozila do prepreke s obzirom na tip prepreke, na brzinama od 20 do 70 km/h. Za prikaz distribucije korišten je kutijasti dijagram (engl. *boxplot*). Vidljivo je kako vrijednosti variraju od 3 do 10 metara, iako je početno zadana udaljenost na kojoj vozilo treba stati 3 metra. Ovakva razlika nastaje najviše zbog pogrešne procijene udaljenost prepreke te se algoritam za kočenje u mnogim situacijama aktivira prerano. Kutijasti dijagram pokazuje medijan gdje je većina podataka u rasponu od 25%-75%. Dijagram pokazuje ekstreme (engl. *Outliers*) kod svake vrste prepreke, pa se u nekim situacijama dogodi da se vozilo zaustavi čak 10 metara ispred prepreke. Ni u jednom slučaju nije došlo do sudara između vozila i prepreke.

Na slici 4.3 prikazana je distribucija dobivenih mjerenja odnosno udaljenosti zaustavljenog vozila do prepreke s obzirom na brzinu vozila od 20 do 70 km/h. Za prikaz distribucije također je korišten kutijasti dijagram. Kutijasti dijagram pokazuje vrijednosti slično kao i kod grafa za prepreke od 3 do 10 metara. Pokazuje medijan, gdje je većina podataka u rasponu od 25%-75%, a manje je rasipanje rezultata na brzinama manjim od 60 km/h. Povećanje brzine vozila uzrokuje ranije aktiviranje kočenja i zaustavljanje na

većoj udaljenosti od prepreke, što je posljedica lošije procjene udaljenosti kad je prepreka udaljenija od vozila. Ovo može rezultirati duljim zaustavnim putem.



Slika 4.2: Distribucija udaljenosti između zaustavljenog vozila i prepreke s obzirom na tip prepreke



Slika 4.3: Distribucija udaljenosti između zaustavljenog vozila i prepreke s obzirom na brzinu kretanja vozila

5. ZAKLJUČAK

Monokularna procjena dubine ima ključnu ulogu u razvoju autonomnih vozila i robota. Uobičajeno se procjena dubine zasniva na senzorima kao što su stereo kamera, radar i LIDAR. Međutim, u novije vrijeme razvijaju se algoritmi koji procjenjuju dubinsku mapu samo na temelju slike dobivene s jedne kamere. Osim niže cijene, prednost ovakvog sustava su i dimenzije, a uz to nije potrebna kalibracija kao u slučaju stereo sustava.

U okviru ovog rada predložen je algoritam za procjenu dubine iz RGB slika i automatsko kočenje u CARLA simulatoru. Procjena dubine temelji se na neuronskoj mreži *PixelFormer* koja je zasnovana na swin transformeru koja je naučena pomoću metoda temeljenih na nadziranom učenju i metoda temeljenih na prilagodbi domene.

Algoritam za automatsko kočenje se temelji na detekciji prepreke u procijenjenoj dubinskoj mapi i eksperimentalno utvrđenom zaustavnom putu vozila za različite brzine. Evaluacija je pokazala da algoritam uspješno detektira prepreke i pravilno izračunava zaustavni put. Međutim, postoje određene varijacije u rezultatima zaustavljanja što je posljedica točnosti procjene dubinske mape.

Algoritam za automatsko kočenje na temelju detekcije prepreke pokazuje obećavajuće rezultate u simuliranom okruženju. Međutim, prije nego što bi se ovakav algoritam primijenio u stvarnom okruženju, potrebno je daljnje testiranje i optimizacija kako bi se osigurala pouzdanost i sigurnost u različitim situacijama na cesti. Implementacija i testiranje u stvarnom okruženju omogućilo bi validaciju performansi i moguće prilagodbe algoritama kako bi se osigurala optimalna sigurnost autonomnog vozila.

LITERATURA

[1] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International journal of computer vision, 47(1-3):7–42, 2002.

[2] I. P. Howard. Perceiving in depth, volume 1: basic mechanisms. Oxford University Press, 2012.

[3] D. Kim, W. Ka, p. Ahn, D. Joo, S.Chun, and J. Kim. „Global-Local Path Networks for Monocular Depth Estimation with Vertical CutDepth“ dostupno na: https://arxiv.org/pdf/2201.07436.pdf

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1–16.

[5] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and d. Puig. „Monocular Depth Estimation Using Deep Learning: A Review„ dostupno na: https://www.mdpi.com/1424-8220/22/14/5353

[6] Dong, X.; Garratt, M.A.; Anavatti, S.G.; Abbass, H.A. „Towards real-time monocular depth estimation for robotics: A survey“. arXiv 2021, dostupno na: arXiv:2111.08600. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9750985

[7] Agarwal, Ashutosh and Arora, Chetan, „Attention Attention Everywhere: Monocular Depth Prediction With Skip Attention“ Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), January 2023, pages 5861-5870. dostupno na: https://arxiv.org/pdf/2210.09071.pdf

[8] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4009–4018, June 2021.

[9] F. Tosi, F. Aleotti, M. Poggi and S. Mattoccia, "Learning monocular depth estimation infusing traditional stereo knowledge", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 9799-9809, Jun. 2019.

[10] R. Ji et al., "Semi-supervised adversarial monocular depth estimation", IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 10, pp. 2410-2422, Oct. 2020.

[11] https://openaccess.thecvf.com/content_cvpr_2016/papers/Mayer_A_Large_Dataset_CVPR_2016_paper.pdf

[12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021., dostupno na: https://arxiv.org/abs/2103.14030

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR), 2013.

[14] P. Böllhoff „PyTorch vs TensorFlow: The Right Machine Learning Software" dostupno na: https://kruschecompany.com/pytorch-vs-tensorflow/

[15] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[16] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[17] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3916–3925, June 2022.

[18] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[19] Link na github repozitorij korištene arhitekture „PixelFormer" https://github.com/ashutosh1807/PixelFormer/tree/main/pixelformer

[20] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. And Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. „Scikit-learn: Machine Learning in Python"

[21] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in Proc. NIPS, 2014, pp. 2366–2374.

[22] Ravi Garg, Vijay Kumar B.G., Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision – ECCV 2016, pages 740–756, Cham, 2016. Springer International Publishing.

SAŽETAK

U ovom radu opisane su postojeće metode procjene dubinske mape na temelju slike dobivene s jedne kamere u okviru autonomne vožnje. Objašnjena je korištena neuronska mreža *PixelFormer* koja je zasnovana na *swin* transformeru. Dobivena mreža trenirana je i testirana na prikupljenim skupovima podataka iz CARLA simulatora. Način prikupljanja novog skupa podataka u simulaciji izveden je po uzoru na KITTI skup podataka koristeći kamere postavljene na prednjem kraju vozila. Dobiveni rezultat je neuronska mreža koja radi procjenu dubinske mape u simulacijskim uvjetima. Nadalje, predložen je algoritam za automatsko kočenje u slučaju nailaska vozila na prepreku koji koristi procijenjenu dubinsku mapu na temelju RGB slike. Cjelokupni algoritam izrađen je u Python programskom okruženju s dodatkom PyTorch biblioteke. Predloženi algoritam evaluiran je na dva načina. Najprije je evaluirana efikasnost procjene dubinske mape iz RGB kamere montirane na vozilu u simulatoru, a zatim je evaluirano automatsko kočenje vozila mjerenjem udaljenosti na kojoj se vozilo zaustavilo na različitim mapama s obzirom na različite prepreke.

Ključne riječi: procjena dubinske mape, simulator, neuronska mreža, autonomna vožnja, algoritam automatskog kočenja

DEPTH ESTIMATION USING MONOCULAR IMAGE IN AUTONOMOUS DRIVING

ABSTRACT

This paper describes existing methods for estimating depth maps based on images obtained from a single camera within the context of autonomous driving. It explains the use of the PixelFormer neural network, which is based on the Swin Transformer architecture. The obtained network was trained and tested on datasets collected from the CARLA simulator. The data collection approach in the simulation was inspired by the KITTI dataset, using cameras mounted at the front of the vehicle. The result is a neural network that can estimate depth maps in simulated conditions. Furthermore, an algorithm for automatic braking in case of encountering obstacles is proposed, which utilizes the estimated depth map based on the RGB image. The algorithm is implemented in a Python programming environment with the addition of the PyTorch library. The proposed algorithm was evaluated in two ways. First, the efficiency of estimating the depth map from an RGB camera mounted on the vehicle in the simulator was evaluated. Then, automatic braking of the vehicle was evaluated by measuring the distance at which the vehicle came to a stop in different scenarios with various obstacles.

Keywords: depth map estimation, simulator, neural network, autonomous driving, automatic breaking algorithm

ŽIVOTOPIS

Josip Pocrnja rođen je 22.2.1999. godine u Tomislavgradu, Bosna i Hercegovina. Opću gimnaziju je završio u Srednja škola "Uskoplje" u Bugojnu, Bosna i Hercegovina. Nakon srednje škole upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Akademski naziv sveučilišni prvostupnik (baccalaureus) inženjer računarstva stječe 2021. godine. Iste je godine upisao diplomski sveučilišni studij računarstva, smjer informacijske i podatkovne znanosti te nakon položenog testiranja postaje stipendist TTTech Auto CEE u Osijeku (tadašnji institut RT-RK).

Potpis:

_____

PRILOZI

Prilog P.3.1: Kompletan popis korištenih biblioteka i njihovih verzija.

addict==2.4.0
carla==0.9.14
certifi==2023.7.22
charset-normalizer==3.2.0
colorama==0.4.6
contourpy==1.0.7
cycler==0.11.0
filelock==3.12.4
fonttools==4.39.4
fsspec==2023.9.2
huggingface-hub==0.17.3
idna==3.4
importlib-metadata==6.8.0
Jinja2==3.1.2
joblib==1.3.1
kiwisolver==1.4.4
markdown-it-py==3.0.0
MarkupSafe==2.1.3
matplotlib==3.7.1
mdurl==0.1.2
mmengine==0.8.5
mpmath==1.3.0
networkx==3.1
numpy==1.24.3
opencv-python==4.7.0.72
packaging==23.1
Pillow==9.5.0

platformdirs==3.10.0
protobuf==4.24.3
Pygments==2.16.1
pyparsing==3.0.9
python-dateutil==2.8.2
PyYAML==6.0.1
regex==2023.8.8
requests==2.31.0
rich==13.5.3
safetensors==0.3.3
scikit-learn==1.3.0
scipy==1.11.1
six==1.16.0
sklearn==0.0.post5
sympy==1.12
tensorboardX==2.6.2.2
termcolor==2.3.0
threadpoolctl==3.1.0
timm==0.9.7
tomli==2.0.1
torch==2.0.1
torchaudio==2.0.2+cu117
torchvision==0.15.2
tqdm==4.66.1
typing_extensions==4.8.0
urllib3==2.0.5
yapf==0.40.2
zipp==3.17.0