Procjena i korekcija lateralnog gibanja vozila

Turina, Marko

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 673

PROCJENA I KOREKCIJA LATERALNOG GIBANJA VOZILA

Marko Turina

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 673

PROCJENA I KOREKCIJA LATERALNOG GIBANJA VOZILA

Marko Turina

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 673

Pristupnik:	**Marko Turina (0036507194)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentorica:	prof. dr. sc. Marija Seder

Zadatak: **Procjena i korekcija lateralnog gibanja vozila**

Opis zadatka:

Nadziranje cestovnog prometa bitno je za određivanje gibanja suputnih vozila u prometu. Kombinacijom niza slika te qps lokacija vozila moguće je odrediti putanje vozila. Međutim, zbog zašumljenjh i nepotpunih senzorskih podataka putanje mogu biti pogrešno određene. U diplomskom je radu potrebno razviti algoritam lokalizacije suputnih vozila unutar cestovne mreže s naglaskom na procjenu i korekciju lateralnog pomaka vozila unutar trake koristeći skup snimljenih slika i gps podataka. Algoritam je potrebno provjeriti u simulacijama te na stvarnim podatcima.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se roditeljima i sestri Sari koji su uvijek bili uz mene i pružali mi podršku u svakom mojem pothvatu. Njihovo vjerovanje u mene mi je davalo snagu u najtežim trenutcima i bez njihove bezuvjetne pomoći ne bih uspio u završetku fakulteta.

Zahvaljujem se mentorici prof. dr. sc. Mariji Seder na ukazanom povjerenju i pomoći tokom cijelog studija.

Zahvaljujem se bliskim prijateljima na podršci i pomoći tijekom studiranja bez kojih ništa ne bi bilo moguće.

Sadržaj

1. Uvod

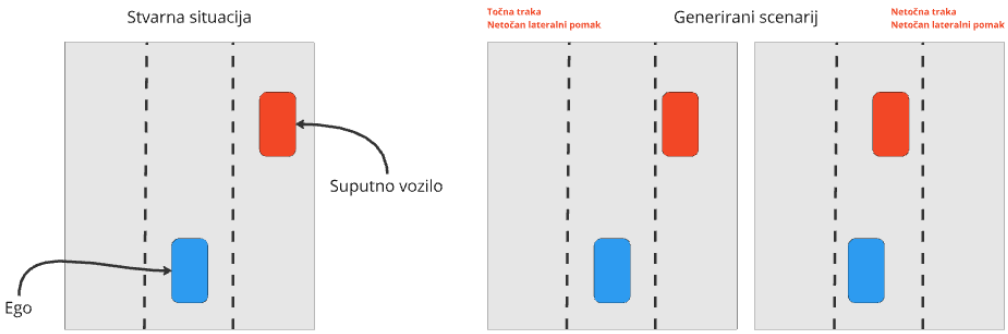
Koncept autonomnih vozila kroz dugi je niz godina u očima sudionika u prometu doimao se poput pothvata kojeg će čovječanstvo u cijelosti ostvariti u dalekoj budućnosti, nimalo sličnu i istovjetnu s 21. stoljećem u kakvom živimo danas. Međutim, realnost u kojoj živimo 2024. godine je takva da se pojava autonomnih vozila u cestovnom prometu današnjice počinje činiti kao izgledna u razdoblju koje predstoji te industrija proizvodnje i razvitka autonomnih vozila uzima sve više maha. Imajući u vidu prethodno iznesenu činjenicu kako je implementacija autonomnih vozila u sustav prometne svakodnevice, u skladu s prometnim normama ustaljenim u cijelome svijetu neizbježna u budućnosti, jasno proizlazi da je postizanje maksimalne razine sigurnosti u prometnom sustavu nužan preduvjet za ostvarenje okruženja u kojem će autonomna vozila postati pravilo, a ne visokotehnološka iznimka.

Sukladno podacima prikupljenim na globalnoj razini, postotak ljudskog faktora odgovornim za ukupan broj prometnih nesreća iznosi 90%. Usprkos činjenici da čak i u svijetu u kojem bi autonomna vozila u potpunosti preuzela prioritet nad ljudski upravljanim vozilima tj. u kojem bi sva vozila koja sudjeluju u prometu bila autonomna, postotak od 0% prometnih nesreća predstavlja veliki izazov. Jasno je da bi se postupnom implementacijom autonomnih vozila u prometna okruženja postotak prometnih nesreća značajno smanjio, a samim time i sustav u kojem bi se prometna pravila dosljedno poštovala i broj stradalih u prometu sveo na apsolutni minimum, postao stvarnost.

Cilj i svrha ovoga rada jest kroz metodologiju iza tehnologije detekcije linija prometnih traka i detekciju objekta iskristalizirati utjecaj implementacije iste na stvarnost te približiti se daljnjem razvoju sigurnosti u prometu, na praktičnim primjerima razložiti sve načine na koje obrada podataka doprinosi ostvarivanju maksimalne razine sigurnosti vožnje.

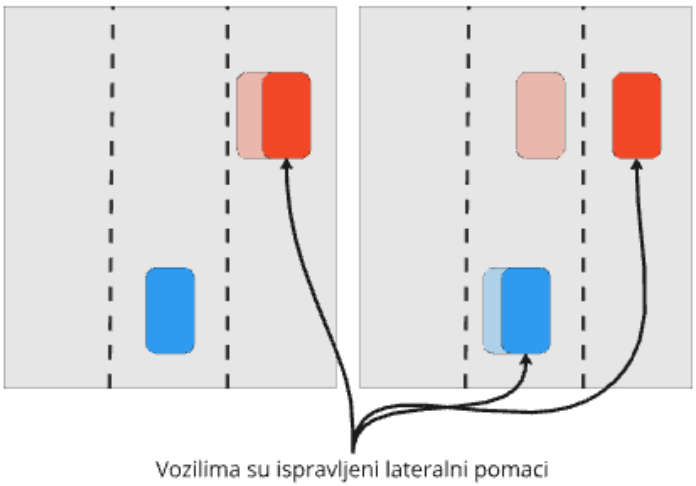
2. Opis problema

Osnovni problem čije otklanjanje u svrhu postizanja maksimalne razine ostvarivosti sastoji se u činjenici kako u sklopu tehnologije, lateralna pozicija Ego vozila (vozilo koje je glavni predmet scenarija) i suputnih vozila (engl. *Fellow vehicles*) (ostala vozila u scenariju) uslijed pojave pogrešaka u primarnom označavanju ili mjerenjima vozila može biti netočna. Ulazne podatke nam čine: videozapis vožnje sniman od strane Ego vozila, generirana lista trajektorija svih vozila u cestovnom scenariju i podaci o cesti. Generirani podaci često znaju biti netočni zbog pogrešaka u labeliranju trake vozila ili greškama u mjerenju. Skica problema vidljiva je na slici 2.1.



Slika 2.1. Prikaz situacije u stvarnosti (lijevo) i prikaz situacije u podacima (desno)

Cilj je, putem algoritma detekcije linija prometnih traka, zajedno s algoritmom detekcije objekata, odrediti i maksimalno točno ispraviti trajektorije na način da budu istovjetne stvarnim podacima u naravi mreže cestovnog prometa, slijedom čega konačni cilj predstavlja reduciranje nedosljednosti između zabilježenih trajektorija i stvarne situacije cestovnog prometa, slika 2.2. Tijek u kojem bi se putem konsolidiranja veze između problema i cilja koji se želi postići, došlo do pune implementacije tehnologije na način da se nedosljednosti u primjeni otklone, obrazložen je na način kako slijedi u daljnjem tijeku poglavlja.



Slika 2.2. Prikaz ciljnog rješenja

Prvenstveno, nužno je koristeći se algoritmom detekcije linija prometnih traka, detektirati predstojeće prometne trake u neposrednoj blizini vozila. Nadalje, potrebno je locirati suputna vozila koristeći se algoritmom detekcije objekata, a iz slike 2.1. razvidno je da se iz situacije prikazane istom, valja prvenstveno izvršiti ekstrakciju bočnog pomaka unutar prometne trake i mapirati detektirano suputno vozilo putem odgovarajućeg unosa. Korak kojeg je dalje nužno poduzeti sastoji se u korištenju prethodno prikupljenih i gore opisanih podataka i konfiguracije cestovne mreže u svrhu identifikacije detektiranih vozila i vozila u listi trajektorija.

Izazov u kontekstu iznesenog prethodno prvenstveno se sastoji u određivanju središta navedenog obvezujućeg obujma i izračunu omjera bočnog pomaka, posebice u kontekstu bočnih prometnih traka (iz perspektive Ego vozila). Nužno je nadalje provesti pridruživanje suputnih vozila s onima prethodno dodanim i zabilježenim unutar popisa oznaka/objekata.

Posljednji korak unutar procesa opisanog prethodnim stavcima i pripadajućim slikovnim primjerima, sastojao bi se u procjeni pogrešaka unutar popisa objekata i korekciji istih. Navedeni bi se proces sastojao u izdvajanju omjera bočnog pomaka i njihovoj usporedbi s njihovim usklađenim pandanima u popisu objekata.

U sklopu svakog datog kadra (engl. *frame*) videozapisa postoji mogućnost izračuna lateralne pozicije ega i suputnih vozila te unosa promjene u popisu objekata koji se realno ogleda u izračunatom stanju. Izazovi za koje se realno može zaključiti da postoje

prilikom praktične provedbe gore opisanog procesa nadasve se ogledaju u nužnosti posjedovanja konkretnih informacija o dubini za pravilno mapiranje objekata iz trajektorija u video i obrnuto te nadalje, a budući da na raspolaganju imamo samo ne-orijentacijski inkluzivne 2D obvezujuće obujme, teško je procijeniti bočni pomak suputnih vozila unutar prometne trake u odnosu na bočne trake, i nužno je istaknuti da je opisani pristup ograničen na primjenu u odnosu na slučajeve u kojima su vidljive sve linije prometnih traka u sklopu slike.

3. Korištene tehnologije, alati i skopovi podataka

3.1. Python

Jedan od najpopularnijih programskih jezika je Python. Njegova zastupljenost raste iz godine u godinu. Prednost je i njegova neovisnost o platformi, te mogućnost rada na svim operacijskim sustavima. Svestran je i popularan jezik za strojno učenje zbog svoje jednostavnosti i bogatog izbora biblioteka. Ključne biblioteke uključuju NumPy za numeričke proračune, Pandas za manipulaciju podacima i Matplotlib za vizualizaciju. Za modele strojnog učenja, biblioteka Scikit-learn pruža učinkovite alate za klasifikaciju, regresiju i grupiranje, dok se biblioteke TensorFlow i PyTorch koriste za duboko učenje. Jednostavna sintaksa Pythona, zajedno s ovim bibliotekama, čini ga idealnim za ovaj rad. Cijela implementacija ovog rada napisana je u Python-u.

3.2. OpenCV

OpenCV (engl. *Open Source Computer Vision Library*) [1] je biblioteka otvorenog koda koja se prvenstveno koristi za zadatke računalnog vida u stvarnom vremenu. Sadržava više od 2500 algoritama. Podržava alate za obradu slike i videa, otkrivanje objekata, prepoznavanje lica, izdvajanje značajki itd. OpenCV podržava širok raspon programskih jezika, uključujući Python, C++ i Java te je usklađen s različitim operativnim sustavima. Njegova obimna zbirka algoritama čini ga idealnim za primjene u robotici, strojnom učenju, proširenoj stvarnosti i autonomnoj vožnji. OpenCV je široko prihvaćen zbog svoje djelotvornosti i svestranosti. U ovome radu OpenCV je korišten za pripremu i obradu videozapisa za interakciju s YOLO modelima.

3.3. SimplePreviewer

SimplePreviewer je alat koji sam osobno razvio i koristan je za jednostavan prikaz trajektorija vozila i ceste. Alat služi jednostavnom i brzom učitavanju scenarija, liste trajektorija vozila i pripadnog cestovnog opisnika te lako pregledavanje tih trajektorija iz ptičjeg pogleda. Alat je detaljnije objašnjen u poglavlju 9.

3.4. COCO skup podataka

COCO (engl. *Common Objects in Context*) [2] skup podataka koristi se u istraživanju računalnog vida za zadatke poput otkrivanja objekata, segmentacije i opisivanja slika. Sadrži preko 330.000 slika, označenih s više od 80 kategorija objekata i detaljnim kontekstualnim informacijama. Slike sadrže složene svakodnevne scene s uobičajenim objektima u njihovoj prirodnoj okolini. COCO se vrlo često koristi za obuku i evaluaciju modela strojnog učenja, naročito u zadacima koji uključuju prepoznavanje objekata. U ovome radu skup podataka koristi se za treniranje modela za detekciju objekata u poglavlju 6.

3.5. TuSimple skup podataka

TuSimple [3] je skup podataka specijaliziran za istraživanje autonomne vožnje, posebno usmjeren na otkrivanje linija traka. Sadrži videozapise snimljene kamerom postavljenom na kamionu, uključujući 6408 označenih okvira. Svaki okvir označen je oznakama voznih traka u scenarijima vožnje u stvarnom svijetu, često u izazovnim uvjetima kao što su sjene, zavoji i gužve u prometu. Veliki dio okvira snimljen je po lošim vremenskim uvjetima sa slabijom vidljivošću. Skup podataka koristi se za obuku i usporednu analizu algoritama za otkrivanje traka, pomažući u poboljšanju točnosti i sigurnosti vožnje. U ovom radu je TuSimple skup podataka korišten za treniranje modela za detekciju linija traka.

3.6. Scenariji

Za algoritam lateralne korekcije pomaka vozila koristi se skup scenarija vožnje. Svaki scenarij sastoji se od videozapisa vožnje te popratnih podataka koji su naknadno izvedeni različitim metodama. Osim videozapisa imamo podatke o cesti, točne podatke o tra-

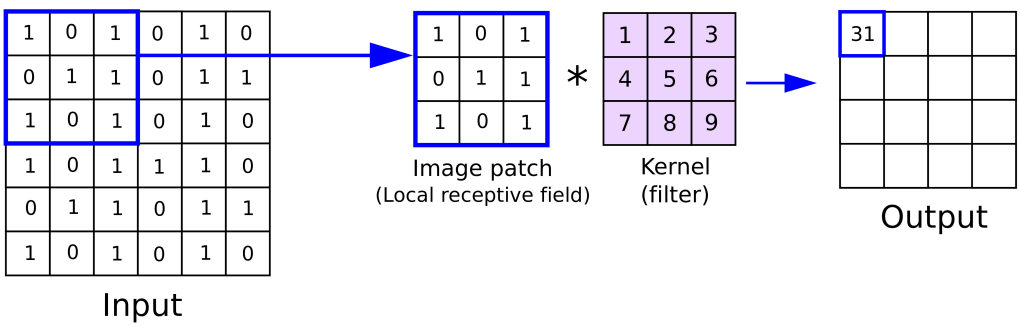
jektorijama vozila u scenariju (ground truth) i podatke o trajektorijama dobivene samo GPS-om koje treba popraviti algoritmom.

4. Konvolucijske neuronske mreže (CNN)

Konvolucijske neuronske mreže CNN (engl. *Convolutional Neural Network*) simboliziraju tip neuronskih mreža koje se sastoje od minimalno jednog konvolucijskog sloja te su izrazito korisne prilikom vršenja obrade podataka strukturiranih u obliku matrice poput npr. fotografija te se iz tog razloga rabe prilikom detekcije objekata. U svojoj suštini konvolucijske neuronske mreže predstavljaju vrstu neuronskih mreža koje su specijalizirane za područje obrade podataka koje se sastoje od prostorne strukture, a i prethodno navedene fotografije predstavljaju upravo takav primjer relevantan u kontekstu ovoga rada. Bazična struktura konvolucijskih neuronskih mreža sadržana je u njezinim osnovnim slojevima.

4.1. Konvolucijski slojevi

Konvolucijski slojevi koji čine temelj istoimenih neuronskih mreža i koji koriste primjenu procesa konvolucije na ulaznu fotografiju, koristeći se detekcijom lokaliziranih uzoraka poput rubova ili oblika, slijedom čega svaki pojedini nastali filter (jezgra) kre-

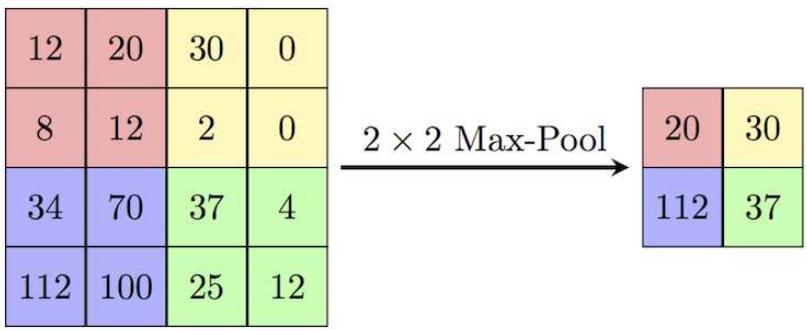


Slika 4.1. Primjer rada konvolucijskog sloja

ira novu mapu čimbenika koja specificira podatke o različitim značajkama sadržanim unutar svake pojedine fotografije. Definiran je ulaznim podatcima i filterom. Filter se pomiče po ulaznim podatcima i provodi operaciju konvolucije te time služi za detektiranje lokalnih uzoraka poput rubova ili oblika. Primjer rada konvolucijskog sloja je na slici 4.1. [4] gdje je prikazana operacija konvolucije između ulaznih podataka i filtera.

4.2. Sloj sažimanja

Slojevi sažimanja (engl. *pooling*) postoje u svrhu redukcije veličine filtrirane slike s izostankom znatnog utjecanja na podatke sadržane u istoj tj. smanjuju se dimenzije mape značajki na način da se relocira filter koji sadrži definiranu veličinu i vrši se pomak preko cjelokupnog ulaznog podatka, radi smanjenja senzibiliteta podataka na ulaznu poziciju. Usprkos posljedici slojeva sažimanja koja se prvenstveno ogleda u gubitku većeg broja informacija, prednosti sloja sažimanja primjenjive su najviše u redukciji složenosti i maksimiziranju učinkovitosti. Djeluje tako da ulaznu sliku ili mapu značajki dijeli na područja koja se ne preklapaju i odabirom maksimalne vrijednosti iz svakog područja. Obično se koristi sažimanje maksimalne vrijednosti (engl. *max pooling*) ili prosječno sažimanje. Primjer rada maksimalnog sažimanja je na slici 4.2. [5].

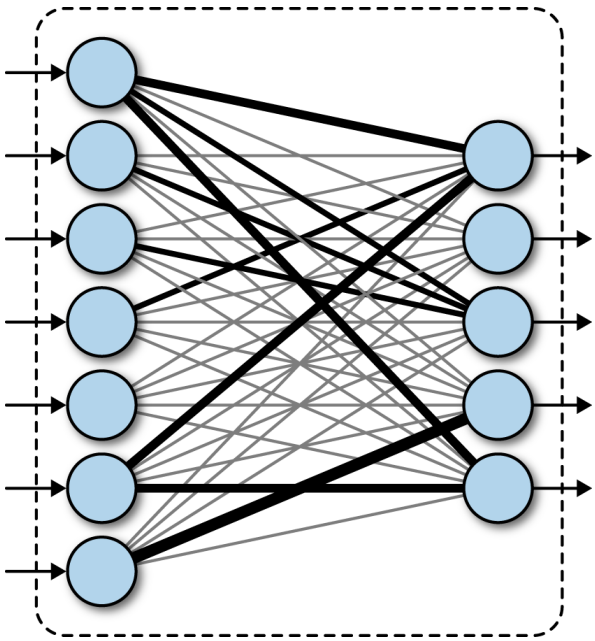


Slika 4.2. Primjer sloja sažimanja

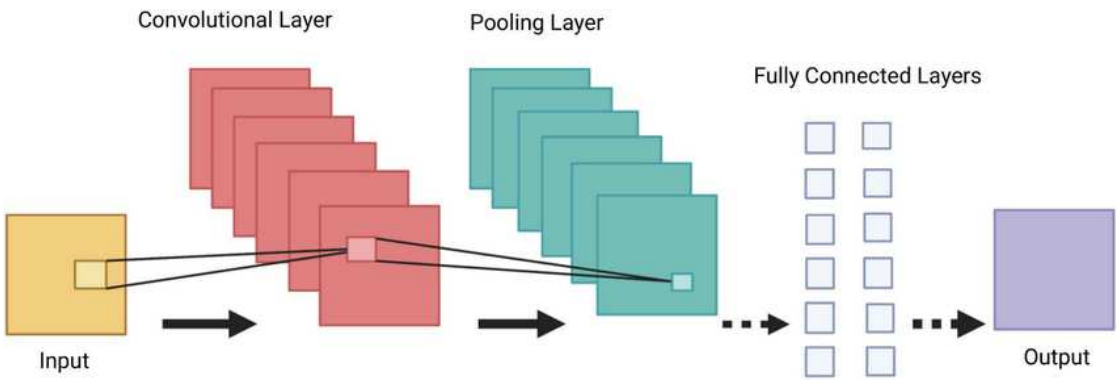
4.3. Potpuno povezani slojevi

Potpuno povezani slojevi (engl. *fully connected layers*) rabe se u svrhu klasificiranja. Svi neuroni sadržani unutar jednoga sloja povezani su s cjelokupnim brojem neurona unutar sljedećeg sloja. Izlaz iz konvolucijskih slojeva transformira se u ulaz u potpuno povezani sloj (engl. također obilježen i kao tzv. *dense layer*) tj. mapa karakteristika pretvara se u

listu (vektor) neurona koji simboliziraju ulazni sloj neuronske mreže. Slično kao što su konvolucijski slojevi podložni nizanju, tako također postoji mogućnost spajanja i više potpuno povezanih slojeva što rezultira kreacijom umjetne neuronske mreže. Na slici 4.3. [6] vidimo primjer potpuno povezanog sloja.



Slika 4.3. Prikaz potpuno povezanog sloja



Slika 4.4. Potpuna struktura CNN mreže

Potpuna struktura CNN mreže koja se sastoji od konvolucijskih slojeva, slojeva sažimanja i potpuno povezanih slojeva vidljiva je na slici 4.4. [7].

4.4. YOLO model

YOLO (engl. *You Only Look Once*) [8] pripada obitelji jednostupanjskih detektora koji ulazne podatke (sliku) analizira samo jednom. YOLO je model detekcije objekata u stvarnom vremenu temeljen na dubokom učenju dizajniran za brzinu i učinkovitost. Duboko

učenje je grana strojnog učenja zasnovana na kompleksnim podatkovnim reprezentacijama. Tradicionalne metode dubokog učenja primjenjuju klizne prozore ili prijedloge regija. Nasuprot tome, YOLO obrađuje cijelu sliku u jednom prolazu kroz konvolucijsku neuronsku mrežu (CNN). Dijeli sliku u rešetku, a svaka ćelija rešetke mreže predviđa granične okvire, klase objekata i mjeru pouzdanosti. YOLO-ova jednoprolazna arhitektura omogućuje pronalaženje objekata u stvarnom vremenu. Time je YOLO vrlo učinkovit za zadatke koji zahtijevaju rezultate u stvarnom vremenu poput autonomne vožnje i videonadzora.

YOLO-ova arhitektura sastoji se od višestrukih konvolucijskih slojeva koji izvlače značajke iz ulazne slike. Model daje fiksni broj graničnih okvira i za svaki okvir predviđa vjerojatnosti klase i mjeru pouzdanosti koja ukazuje na vjerojatnost da se objekt nalazi u okviru. Jednostavnost YOLO dizajna dovodi do ravnoteže između brzine i točnosti. YOLO se može nositi s malim ili preklapajućim objektima.

U ovome radu korištene su razne inačice YOLO modela, YOLinO model za detekciju linija traka koji je bolje objašnjen u poglavlju 5. i YOLOv4 [9] za detekciju vozila u poglavlju 6.

5. Detekcija linija trake

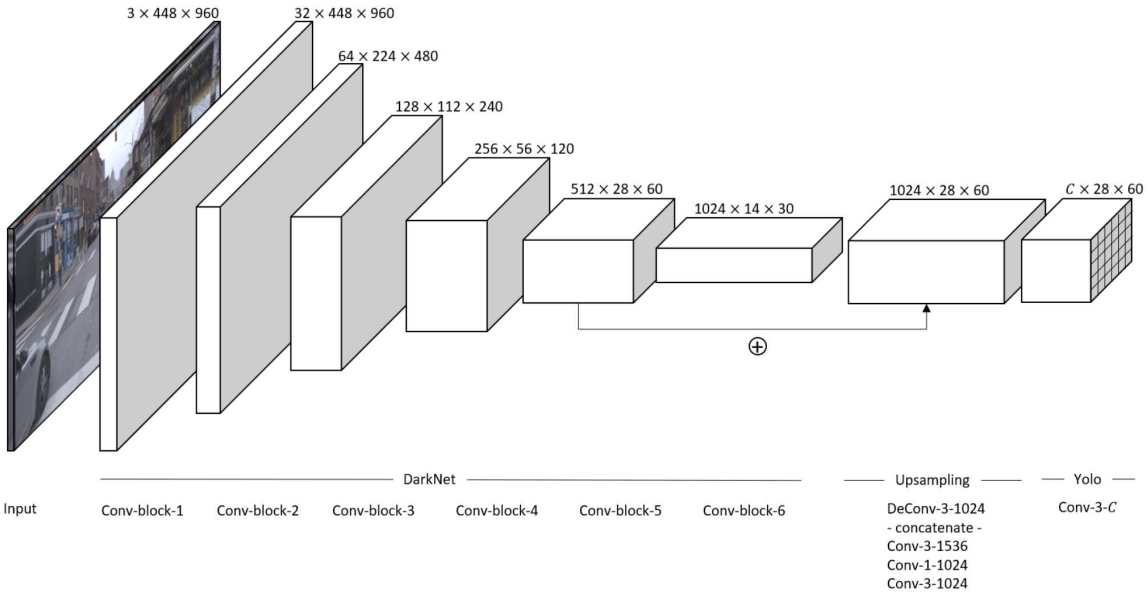
Prvenstveni problem čije je rješavanje nužno jest detekcija traka u slici. Postoji više pristupa kojima to možemo postići. Tradicionalnim računalnim vidom otklanjanje spomenutog problema moglo bi biti izvedeno na način da prvo otkrijemo rubove u slici s Cannyjevim detektorom rubova pa potom prepoznamo ravne trake s Houghovom linijskom transformacijom. Taj pristup ima puno ograničenja i nedostataka pa je korištenje dubokih neuronskih mreža u posljednje vrijeme potpuno zamijenilo tradicionalni pristup u brojnim sličnim problemima.

Klasične metode su često osjetljive na buku (šum) u podacima, uvjete osvjetljenja i loše označene trake. Podaci sa šumom su podaci koji su oštećeni, iskrivljeni ili imaju nizak omjer signala i šuma. Klasične metode rade dobro na ravnim i jednostavnim krivuljama, ali imaju problema kada su trake zakrivljene, isprekidane ili nepravilne. Također, uopće ne koriste sve dostupne podatke kako bi se poboljšao njihov izlaz. Na sve ove probleme odgovor daju duboke neuronske mreže.

YOLinO [10] koristi jedan prolaz za otkrivanje polilinija. Algoritam je inspiriran YOLO (You Only Look Once) arhitekturom. YOLO algoritam je osmišljen da u jednom prolazu kroz podatke, time je dobio i sam naziv, odredi detekciju linija i time je sposoban raditi u sustavima u stvarnom vremenu gdje je brzina bitna.

5.1. Arhitektura YOLinO-a

Za okosnicu koristi se DarkNet-19 [11] koja se sastoji od 19 konvolucija i pet slojeva maksimalnog sažimanja za izlaz mape značajki fiksne veličine koja predstavlja rešetku ćelija. Okosnica je dodatno proširena sa dodatnom konvolucijom za predikciju parametara za određeni broj segmenata linija unutar svake ćelije.



Slika 5.1. Arhitektura DarkNet-19

Svaki prediktor je konstruiran kao P = (l, k, c), gdje je l geometrijsko obilježje linije, k je pouzdanost za klasu, a c pouzdanost predikcije. Broj prediktora po ćeliji stavlja gornju među na broj linija segmenata koja mogu biti predviđena po ćeliji rešetke. Klase k označavaju tip linije koju model predviđa, npr. puna linija, iscrtkana linija itd.

Svaki slika I je podjeljena na SxS rešetku, gdje je svaka ćelija rešetke odgovorna za otkrivanje segmenata polilinije unutar svoje regije. Za svaku ćeliju C_{ij} model predviđa skup segmenata linije, parametriziranih njihovim krajnjim točkama (x_1,y_1) i (x_2,y_2) u prostoru slike, njihove oznake klasa k i pouzdanost predikcije c. Svaki segment linije je geometrijski određen sa:

$$L_{ij} = (x_1^{ij}, y_1^{ij}), (x_2^{ij}, y_2^{ij}) \tag{5.1}$$

Gdje L_{ij} predstavlja segment linije predviđen pomoću i,j-te ćelije rešetke. Ovi segmenti su potom pročišćeni i kombinirani potiskivanjem nemaksimalnih odziva (Non maximal suppression - NMS) kako bi se uklonila dvostruka ili preklapajuća predviđanja, čime se osigurava čista konačna polilinija.

5.2. Potiskivanje nemaksimalnih odziva

Potiskivanje nemaksimalnih odziva (engl. *Non Maximal Suppresion*) [12] primjenjuje se na skup predviđenih segmenata kako bi se uklonile nepotrebne detekcije. Ovaj proces funkcionira odabirom segmenta s najvećom ocjenom pouzdanosti i potiskivanjem svih ostalih segmenata koji se preklapaju s najvećim odabranim segmentom iznad određenog praga. S obzirom na skup predviđanja $S = L_1, L_2, \ldots, L_n$ s odgovarajućim ocjenama pouzdanosti $C = c_1, c_2, \ldots, c_n$, potiskivanje nemaksimalnih odziva algoritam iterativno odabire segment s najvećom ocjenom pouzdanosti c_i i odbacuje ostale koji imaju veći IoU (Intersection over Union) s odabranim segmentom od unaprijed zadanog praga.

Za linijske segmente L_i i L_j, IoU je definirana kao omjer njihovog presjeka i njihove unije:

$$IoU(L_i, L_j) = \frac{|L_i \cap L_j|}{|L_i \cup L_j|} \tag{5.2}$$

IoU osigurava da je konačni skup polilinija malen i bez duplikata.

5.3. Funkcija gubitka i točnost

Postupak optimizacije parametara određenih strukturom modela uključuje prilagođenu funkciju gubitka dizajniranu za usklađivanje točnosti otkrivanja kratkih i dugih segmenata linija. Oni su ključni za rekonstrukciju kontinuiranih polilinija poput granica traka. Funkcija gubitka koja se koristi u YOLinO sastoji se od nekoliko komponenti koje koriste težinsku sumu za ukupni gubitak.

$$\mathcal{L} = \lambda_{conf} * \mathcal{L}_{conf} + \lambda_{coord} * \mathcal{L}_{coord} + \lambda_{smooth} * \mathcal{L}_{smooth} \tag{5.3}$$

- $\mathcal{L}_{conf}$ predstavlja gubitak pouzdanosti, kažnjava netočne predikcije linijskih segmenata
- $\mathcal{L}_{coord}$ predstavlja gubitak koordinata linijskog segmenta, što osigurava točnost krajnjih točaka (x_1,y_1) i (x_2,y_2)

- $\mathcal{L}_{smooth}$ je ograničenje glatkoće, koje osigurava da je prijelaz između uzastopnih segmenata linije gladak i kontinuiran, oponašajući izgled stvarnih granica voznih traka

Komponente λ_{conf}, λ_{coord} i λ_{smooth} su hiperparametri koju uravnotežuju ove komponente i određuju se unutar validacije modela.

Jedna od ključnih prednosti YOLinO-a je njegova sposobnost rada u stvarnom vremenu, koja radi pri 187 sličica u sekundi (FPS). To ga čini prikladnim za aplikacije u stvarnom vremenu u autonomnoj vožnji gdje je niska latencija kritična.

Točnost modela detekcije traka može se ocijeniti koristeći metrike preciznosti (engl. *precision*), odziva (engl. *recall*) i mjere F_1. Preciznost mjeri udio ispravno klasificiranih primjera u skupu pozitivno označenih primjera. Odziv je mjera koja pokazuje udio pozitivno klasificiranih primjera u skupu svih pozitivnih primjera. F_1 mjera je harmonijska sredina preciznosti i odziva. Kombinira preciznost i odziv modela i maksimizaciju mjere F_1 istovremeno maksimizira preciznost i odziv.

$$Tocnost = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.4}$$

$$Preciznost = \frac{TP}{TP + FP} \tag{5.5}$$

$$Odziv = \frac{TP}{TP + FN} \tag{5.6}$$

$$F_1 = \frac{2 * Preciznost * Odziv}{Preciznost + Odziv} \tag{5.7}$$

gdje je TP broj točno pozitivnih (engl. *True positives*) primjera, FP je broj netočno pozitivnih (engl. *False positives*) primjera, FN je broj netočno negativnih (engl. *False negatives*) primjera te TN je broj točno negativnih primjera (engl. *True negatives*).

Model je pretreniran na skupu podataka TuSimple i evaluiran sa TuSimple mjerilom

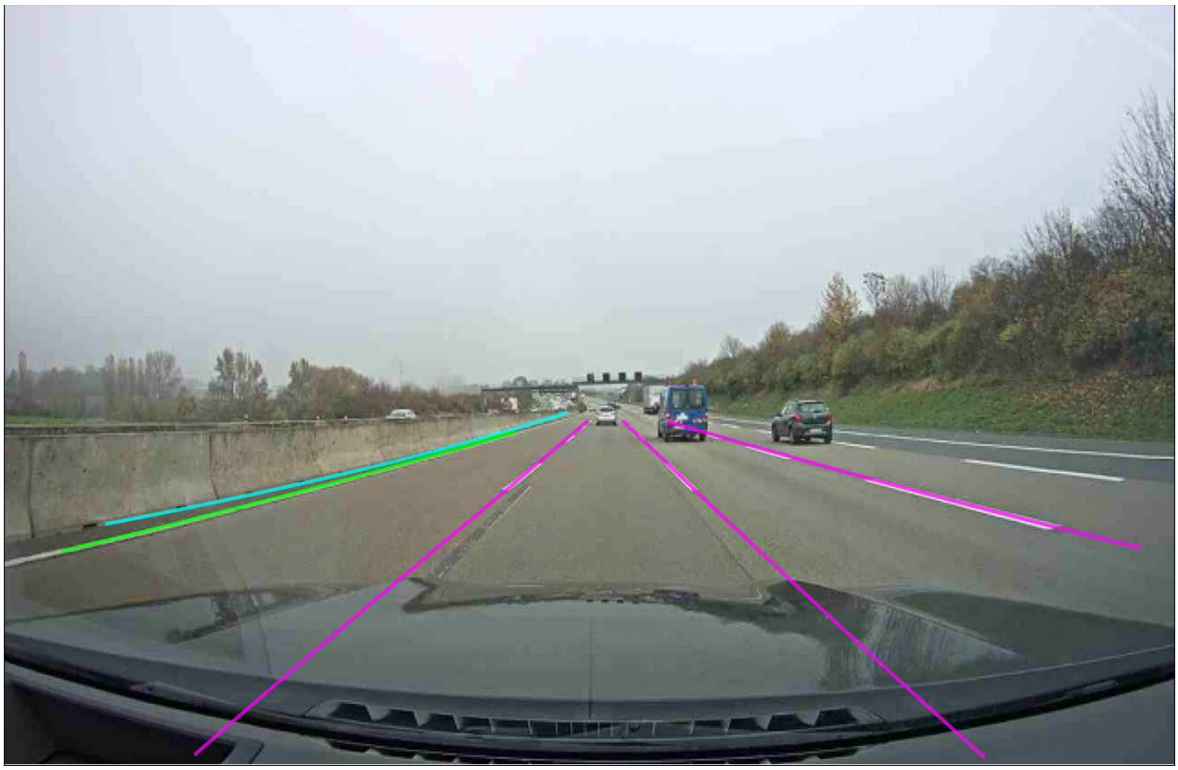
(engl. *benchmark*). Dodatni hiperparametar modela je broj prediktora po ćeliji rešetke, odnosno koliko prediktora se nalazi na svakom podijeljenom dijelu ulazne slike. Rezultati modela vidljivi su na tablici 5.1. S tablice je vidljivo da je točnost najbolja s 4 prediktora dok je F_1 mjera najbolja za 12 prediktora. Pošto ne radimo iznimno kompleksan zadatak nego samo tražimo detekciju linija traka, 8 prediktora nam se čini kao najbolje rješenje.

Tablica 5.1. Tablica mjera modela detekcija traka

Broj prediktora	**Točnost**	**Preciznost**	**Odziv**	**F_1**
4	0.911	0.520	0.920	0.664
8	0.905	0.589	0.932	0.722
12	0.901	0.602	0.918	0.727

Na priloženoj slici možemo vidjeti rad modela. Na ulaz dostavljamo neobrađenu sliku 5.2.a a na izlazu modela dobivamo sliku 5.2.b, točnije na izlazu modela dobivamo skup segmenata koji sadrže klasu linije. Na drugoj slici možemo vidjeti različite boje klasa, npr. plava boja označava rubnik ceste, zelena boja označava vanjsku punu liniju dok su ljubičastom bojom označene isprekidane linije te skup koordinata (x,y) koje točno označuju liniju.

(a) Ulazna slika



(b) Označene trake

Slika 5.2. Primjer detekcije traka

6. Detekcija objekata

U sustavima autonomne vožnje su otkrivanje, klasifikacija i praćenje vozila u stvarnom vremenu ključni su za omogućavanje sigurne navigacije. YOLOv4, naprednija verzija obitelji YOLO (You Only Look Once), nudi značajna poboljšanja na brzini i točnosti za otkrivanje objekata u stvarnom vremenu, što ga čini vrlo prikladnim za otkrivanje i klasificiranje vozila u dinamičnim okruženjima. Ovo poglavlje govori o tome kako se YOLOv4 koristi za prepoznavanje vozila, njihovo klasificiranje i praćenje njihovih pozicija kroz uzastopne okvire u videozapisu.

6.1. Arhitektura

YOLOv4, nadovezuje se na arhitekturu svojih prethodnika dok uključuje nove tehnike optimizacije za poboljšanje ravnoteže između brzine i točnosti.

YOLOv4 se sastoji od tri ključne komponente:

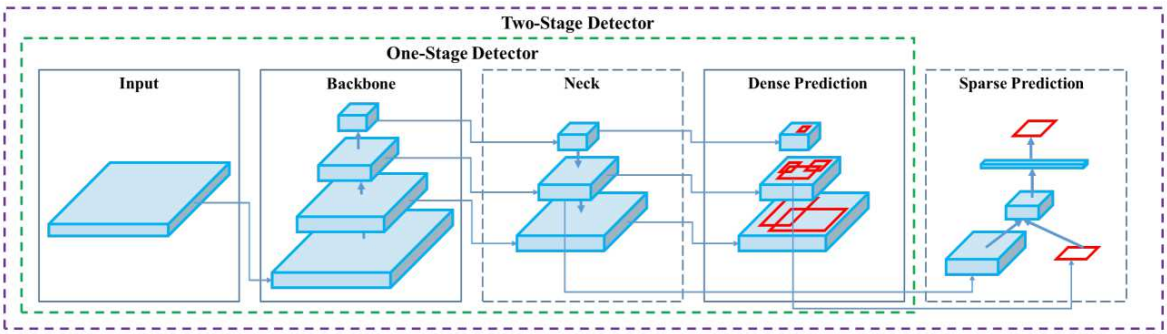
1. **Okosnica**: YOLOv4 koristi CSPDarknet53 kao okosnicu za ekstrakciju značajki. CSPDarknet53 poboljšava učinkovitost računanja dok istovremeno čuva bogate informacije o značajkama iz ulazne slike. Okosnica izvlači značajke u više razmjera pomoću konvolucijskih slojeva, grupne normalizacije i funkcija aktivacije

   CSP (engl. *Cross Spatial Partial Connections*) označava dijeljenje mape značajki na dva dijela, jedan dio prolazi kroz konvolucijske slojeve dok drugi dio se preskače konvolucijsku mrežu nakon čega se oba rezultata spajaju.

2. **Vrat**: Vrat služi kao spojnica između slojeva okosnice i glave modela. YOLOv4 koristi PANet (engl. *Path Aggregation Network*) [13] i SPP (engl. *Spatial Pyramid Pooling*) [14] kao komponente vrata za poboljšanje ekstrakcije značajki iz različi-

tih slojeva. PANet poboljšava protok značajki u mreži agregiranjem značajki niže i više razine, osiguravajući da se u detekciji koriste i precizne i kontekstualne informacije.

3. **Glava**: Glava YOLOv4 odgovorna je za stvaranje graničnih okvira, vjerojatnosti klasa i mjera pouzdanosti. YOLOv4 koristi sidrene okvire za predviđanje lokacija objekata u različitim mjerilima, što ga čini sposobnim za otkrivanje objekata različitih veličina.



Slika 6.1. Arhitektura YOLOv4

Slično svojim prethodnicima, YOLOv4 dijeli ulaznu sliku u SxS ćelija. Svaka ćelija mreže odgovorna je za otkrivanje objekata čije središte pada unutar ćelije. Za svaku ćeliju mreže, YOLOv4 predviđa više graničnih okvira, pri čemu se svaki granični okvir parametrizira pomoću (x, y, w, h) gdje su (x, y) koordinate okvira a (w, h) su širina odnosno visina okvira. Svaki okvir je također parametriziran s mjerom pouzdanosti C da okvir sadrži objekt.

Model je sposoban ne samo prepoznati objekte nego ih i klasificirati. Svakom okviru je određena vjerojatnost pripadnosti svim klasama te je na kraju izabrana klasa koja ima najveću vjerojatnost.

$$c_{class} = \arg\min_{k \in classes} P(c_k) \tag{6.1}$$

Potpuni izlaz parametara okvira za pojedinu ćeliju (i, j) rešetke izgleda:

$$Izlaz_{i,j} = \{(x, y, w, h, C), P(c_1), P(c_2), ..., P(c_k)\} \tag{6.2}$$

6.2. Funkcija gubitka

Slično kao i kod detekcije linije traka, naša funkcija gubitka sastoji se od više djelova kako bi balansirali izlaz za pojedine komponente modela.

Funkcija gubitka sastoji se od tri komponente:

$$\mathcal{L} = \mathcal{L}_{CIoU} + \mathcal{L}_{conf} + \mathcal{L}_{class} \tag{6.3}$$

- **Gubitak CIoU** $\mathcal{L}_{CIoU}$: CIoU (engl. *Complete Intersection over Union*) gubitak je sličan MSE (engl. *Mean Squared Error*) gubitku. Predlaže malo zanimljiviju usporedbu širine i visine (dosljednost između omjera širine i visine), ali zadržava MSE za usporedbu između središta graničnih okvira. Više detalja možete pronaći na slici **??**
- **Gubitak pouzdanosti** $\mathcal{L}_{conf}$: gubitak kažnjava pogrešku mjere pozdanosti
- **Gubitak klasifikacije klase** $\mathcal{L}_{class}$: gubitak unakrsne entropije za vjerojatnosti klase u odnosu na oznaku točne klase

6.3. SORT

Praćenje objekta je proces lociranja objekta u videozapisu i praćenje njegovog kretanja tijekom vremena. Glavni cilj praćenja označenog okvira je održavanje dosljedne identifikacije za svako detektirano vozilo dok se kreće scenom.

Algoritam SORT (engl. *Simple Online and Real-time Tracking*) [15] jednostavan je i efikasan algoritam koji može pratiti više objekata istovremeno i asocirati detekciju okvira objekata kroz frame-ove. SORT je kombinacija Kalmanovog filtera i IoU metrike. SORT koristi Kalmanov filter [16] za predikciju lokacije objekta u sljedećem frame-u i asocira detekciju sa predviđenom lokacijom koristeći IoU. Nakon svakog frame-a stanje Kalmanovog filtra ažurira se.

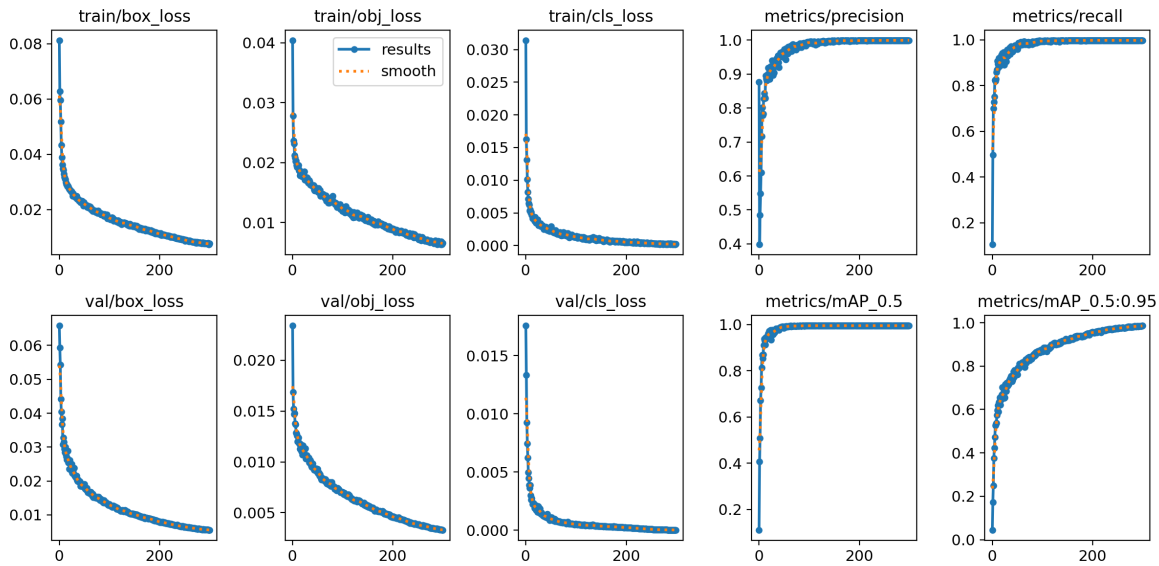
Prednosti SORT algoritma jest brzina izvođenja što ga čini dobrim kandidatom za korištenje u aplikacijama u stvarnom vremenu i visokoj preciznosti.

6.4. Treniranje i evaluacija

Model YOLOv4 je pretreniran na skupu podataka COCO te radi dobro na širokom skupu primjena, ali ukoliko želimo bolju točnost potrebno ga je dodatno trenirati (engl. *fine-tuning*) na skupu podataka koji bolje opisuju naš problem. Ovim pristupom omogućujemo našem modelu da bolje pronalazi tražene objekte i s većom preciznošću. Dodatno treniranje provodim na skupu podataka od 2000 slika prometa s označenim okvirima vozila i njihovom klasom.

Tako treniranom modelu potrebno je ocjeniti preformanse i pouzdanost na validacijskom dijelu skupa podataka. Ocjenjivanje modela je ključan korak u razumijevanju koliko dobro model radi na određenom zadatku i koliko je učinkovit na neviđenim podacima.

Treniranje modela izvođeno je na NVIDIA A100 grafičkoj kartici kroz 300 epoha sa serijom veličine 16 (engl. *batch size*).



Slika 6.2. Metrike tijekom treniranja modela

Na slici 6.2. vidljivi su gore navedeni gubici, preciznost, odziv i metrika mAP [17] tj. srednja vrijednost prosječne preciznosti (engl. *Mean Average Precision*) kroz svaku epohu treniranja modela. Srednje vrijednosti prosječne preciznosti (engl. *Average precision*) izračunavaju se preko vrijednosti odziva od 0 do 1.

- **mAP50**: Srednja vrijednost prosječne preciznosti za IoU prag od 0.5

- **mAP50-95**: Srednja vrijednost prosječne preciznosti za IoU prag od 0.5 do 0.95 gdje se prag IoU postepeno povećava, ova metrika pruža strožu procjenu nego mAP50

Na slici 6.3. vidimo rad prethodno treniranog modela. Na ulaz dovodimo videozapis iz scenarija i detektiramo okvire objekata s modelom. Na ulaz dovodimo originalni videozapis, a ne izlaz iz modela detekcije traka jer označene trake utječu na točnost predikcije modela. Trake često prolaze kroz vozila te time onemogućuju modelu da napravi točnu predikciju. Ionako nas samo zanimaju parametri okvira objekata tj. kooridinate pronađenog objekta (x, y) na slici te širina i visina objekta. Te parametre možemo naknadno spojiti s izlazom detekcije traka te koristiti u daljnjem računanju.

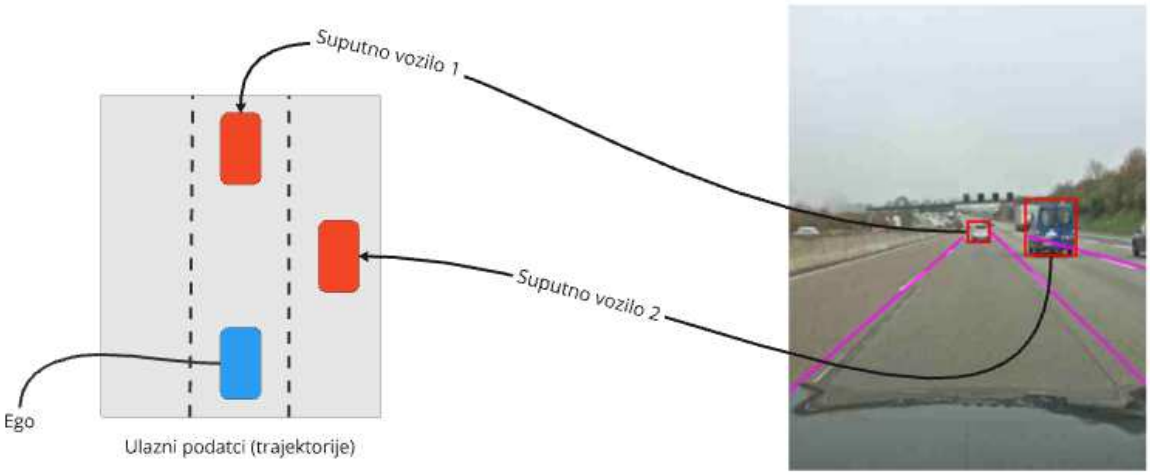
(a) Ulazna slika

(b) Označena vozila

Slika 6.3. Primjer detekcije vozila

7. Pronalaženje podudarajućih objekata u listi

Nakon što smo pronašli okvire objekata potrebno ih je upariti s listom objekata koja opisuje scenarij 7.1. Svako prepoznato vozilo potrebno je pronaći u listi vozila u danom frame-u i upariti ga s točnim vozilom u listi podataka.



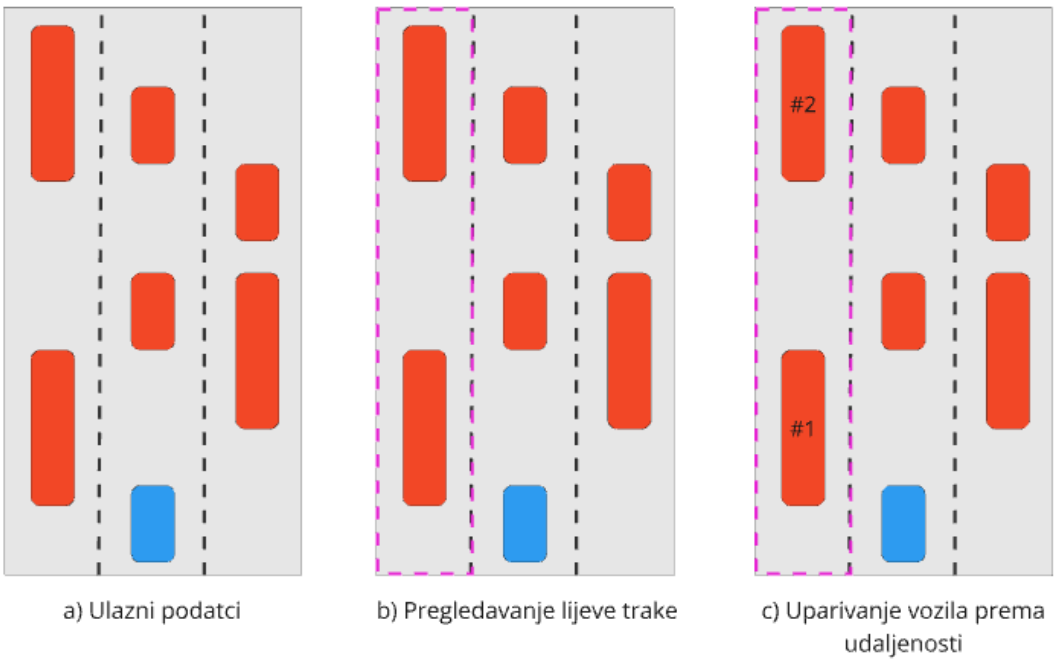
Slika 7.1. Pronalaženje vozila u listi podataka

Način na koji ostvarujem označavanje traka je detektiranjem linija traka oko ego vozila i numeriranjem istih. Traku čine lijeva i desna polilinija trake prepoznata s modelom iz poglavlja 5. Nakon što su trake označene uparujem ih s trakama iz podataka ceste. To činim na način da pronađem ego u listi objekata i pronađem njegovu i dvije susjedne trake, lijevu i desnu.

Svakom prepoznatom objektu (vozilu) potrebno je odrediti traku kojoj pripada tako da se izračuna sredina doljnjeg brida prepoznatog okvira (C_x, C_y) i izračuna u kojoj traci se nalazi vozilo. Izračunatom središtu donjeg brida vozila se određuje najbliža lijeva i desna polilinija trake te se svrstava u tu traku te mu se time jednožnačno određuje traka

u kojoj se nalazi. Ukoliko je više objekata prepoznato u istoj traci oni su poredani po udaljenosti od ega. Udaljenost od ega je zapravo iznos C_y jer nas zanima samo longitudalna udaljenost od ega jer dva objekta ne mogu imati preklapajući okvir zbog pretpostavke da se u stvarnosti nalaze na nekoj udaljenosti. Time je svaki prepoznati objekt svrstan u svoju traku te je još potrebno pronaći podudarajući objekt iz liste vozila.

U listi vozila potrebno je pronaći objekte u lijevoj, središnjoj (traka u kojoj je ego) i desnoj traci u odnosu na ego vozilo te ih svrstati po udaljenosti od ego vozila po longitudinalnoj koordinati, tj. iznosu y pixela, lateralna udaljenost, tj. iznos x koordinate nam nije potrebna.



Slika 7.2. Primjer uparivanja vozila u lijevoj traci

Posljednji korak je odrediti podudarnost između okvira objekata i vozila koji se nalaze u listi. Iteriramo kroz tri trake (lijeva, središnja i desna) te ukoliko se u traci nalazi prepoznati okvir onda pronalazimo objekt u listi trajektorija kojemu odgovara ta traka i udaljenost od ega. Uzimamo u obzir samo objekte koji se nalaze ispred ego vozila u listi objekata jer prepoznati objekt ne može biti iza vozila pod pretpostavkom da je kamera orijentirana unaprijed. Ukoliko se u traci nalazi više prepoznatih okvira objekata onda ih poredamo po iznosu y koordinate okvira te time su nam prepoznati okviri sortirani po udaljenosti od ego vozila jer što je objekt dalje u stvarnosti to će biti i "više" na slici. Potom te okvire uparujemo s vozilima iz te trake u listi trajektorija, isto svrstani po uda-

ljenosti od ego vozila. Na slici 7.2. možemo vidjeti proces uparivanja prepoznatih okvira u lijevoj traci s vozilima iz liste objekata sortiranih prema udaljenosti.

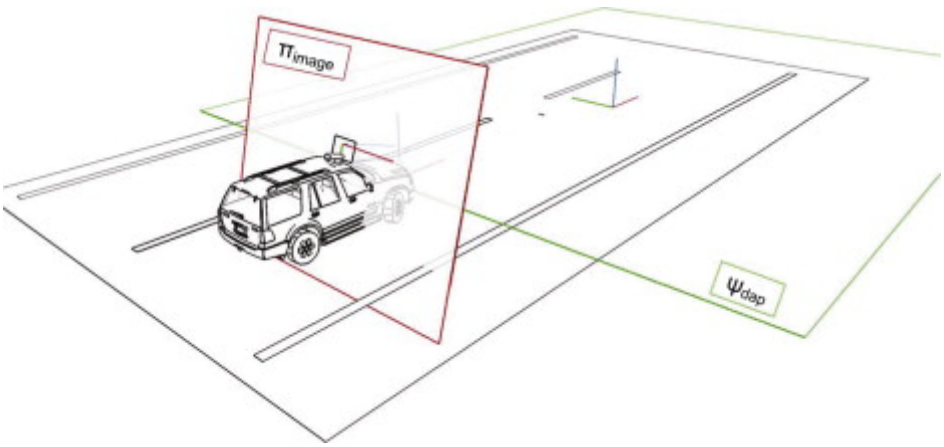
Ukoliko ne pronađemo vozilo u istoj traci u listi vozila na razumnoj udaljenosti potom pregledavamo susjednu traku u listi vozila tražeći podudarajući objekt. Time rješavamo situaciju u kojoj je vozilo u listi trajektorija u pogrešnoj traci. U slučaju da okviru nije pronađen podudarajući objekt u list vozila taj okvir se za tu sliku odbacuje i ne vršimo nikakav lateralni pomak.

8. Lateralna korekcija objekata

8.1. Inverzno perspektivno mapiranje

Inverzno perspektivno mapiranje IPM (engl. *Inverse Perspective Mapping*) [18] je tehnika računalnog vida koja se koristi za transformaciju perspekcije slike (obično s kamere vozila) u pogled iz ptičje perspektive (odozgo prema dolje). Ova transformacija uklanja efekte izobličenja perspektive, olakšavajući mjerenje udaljenosti i preciznu detekciju objekata ili obilježja ceste. U autonomnoj vožnji, IPM je ključan za zadatke kao što su izbjegavanje prepreka i određivanje bočnog pomaka ego vozila u odnosu na cestu ili linije traka.

Kada kamera snimi sliku ceste, objekti na slici prolaze kroz perspektivnu projekciju, gdje objekti koji su bliže kameri izgledaju veći, a objekti koji su dalje manji. Ovo iskrivljenje predstavlja izazov za točno tumačenje položaja i veličine objekata u stvarnom svijetu.



Slika 8.1. Inverzno perspektivno mapiranje

Odnos između točke na cesti i njezine odgovarajuće točke slike je nelinearan zbog ove perspektivne projekcije. Cilj IPM-a je preslikati točke na slici iz ove iskrivljene perspek-

tive u ravan pogled odozgo prema dolje, kao da se scena promatra izravno odozgo. Na slici 8.1. možemo vidjeti primjer transformacije slike. π_{image} je ravina koja je uhvaćena kamerom a ψ_{dap} je originalna ravnina na koju želimo vratiti projekciju uhvaćenu kamerom. Pošto je projekcija slike dobivena određenom projekcijom to znači i da je moguće napraviti inverznu projekciju.

Točka $P_w = (X_w, Y_w, Z_w)$ u trodimenzionalnom stvarnom svijetu je projicirana na kamerinu dvodimenzionalnu sliku na koordinate $P_i = (u, v)$. Projekcija je vođena Pinhole camera modelom, koji je opisan sljedećom jednadžbom:

$$s * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K * \begin{bmatrix} R * T \end{bmatrix} * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{8.1}$$

Gdje je:

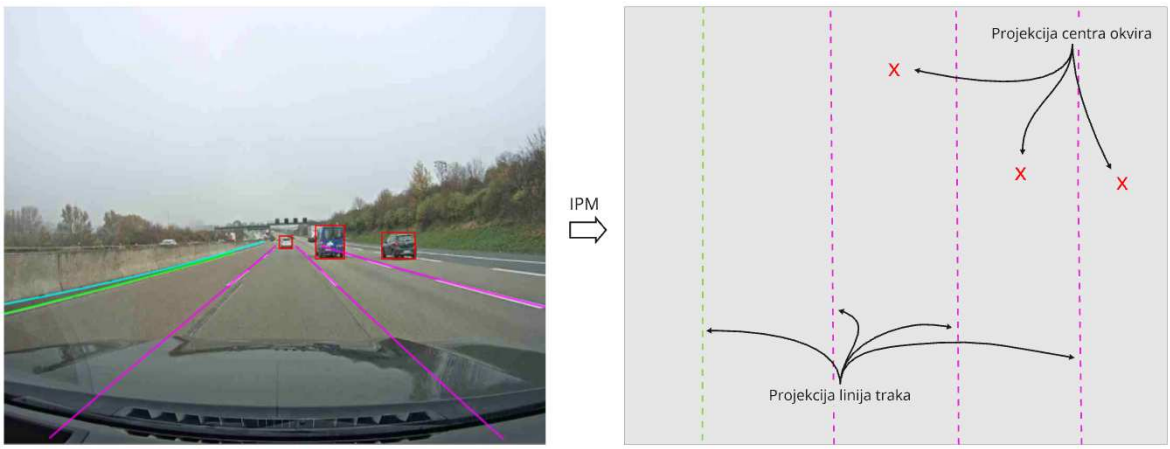
- s je faktor skaliranja
- K je intrinzična matrica kamere, koja definira žarišnu duljinu i središnu točku kamere
- R je je matrica rotacije, koja predstavlja orijentaciju kamere
- T je vektor translacije, koji predstavlja položaj kamere u odnosu na koordinatni sustav stvarnosti
- u i v su koordinate piksela u slici

Kako bi uklonilo izobličenje perspektive, IPM okreće ovaj proces projekcije. S obzirom na koordinate piksela (u,v) na slici, cilje je pronaći odgovarajuće koordinate u stvarnom svijetu (X_w, Y_w). To se postiže rješavanjem inverza jednadžbe projekcije. Srećom mi našu jednadžbu pojednostavljujemo jer nas zanima projekcija na ($Z_w = 0$) odnosno projekcija na tlo u stvarnome svijetu. Transformacija nam onda izgleda ovako:

$$\begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix} = H^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{8.2}$$

Ovdje je matrica H homografska matrica koja povezuje koordinate slike s osnovnom ravninom. Ova matrica sadrži unutarnje parametre kamere i vanjske parametre (rotaciju i translaciju) koji opisuju njezinu orijentaciju i položaj u odnosu na cestu.

Inverzna transformacija H^{-1} primjenjuje se na svaki piksel na slici, učinkovito preslikavajući sliku kamere u pogled odozgo prema dolje. Ova transformacija ispravlja izobličenje perspektive, dopuštajući udaljenostima i kutovima na slici da odražavaju svoje prave vrijednosti u stvarnom svijetu.



Slika 8.2. Preslikavanje okvira i linija traka

Veliko računalno olakšanje nam dolazi iz činjenice da ne trebamo svaki piksel preslikavati i za njega računati rješenje matrice nego samo za grupu piksela koje nam označavaju prepoznate linije traka i središte donjeg brida okvira pronađenog s modelom prepoznavanja objekata. Na slici 8.2. možemo vidjeti željeno preslikavanje centra donjeg brida okvira objekta i projekciju linija traka. Kako bi smo mogli izračunati lateralni pomak ego vozila i suputnih vozila u scenariju potrebno je inverzno projicirati te objekte, u ovome slučaju središte donjeg brida prepoznatog okvira objekta te odrediti omjer pomaka vozila u traci.

8.2. Lateralna korekcija Ego vozila

U kontekstu autonomne vožnje, točno određivanje bočnog položaja ego vozila u odnosu na oznake trake ključno je za sigurnu navigaciju. Bočni položaj ega vozila odnosi se na njegov relativni pomak od središta trake. Izdvajanje tih informacija moguće je analizom linija trake otkrivenih pomoću algoritma detekcija traka YOLinO koja je detaljno objašnjena u poglavlju 5. U nastavku je detaljno objašnjenje kako je to moguće napraviti, uključujući potrebne korake i formule.

Detekciju lateralnog pomaka ego vozila računamo s pretpostavkom da je kamera postavljena u lateralnoj sredini vozila, odnosno ukoliko se omjer x koordinati pixela predstavi od [0,1] onda će naša kamera ležati na 0.5, tj. centru vozila. Bez informacije o položaju kamere unutar vozila je teško izračunati pomak u traci te se zbog toga kao standard uzima da je vrijednost x koordinate jednaka polovici slike. Također to znači da je i nakon transformacije naša točka gledišta u sredini.

Potrebno je izračunati omjer položaja ego vozila s obzirom na lijevu i desnu liniju trake. Pronalaskom najbliže dvije linije traka središtu slike moguće je izračunati te omjere. Za reprezantivnu točku svake od tih dviju traka uzimamo prvu vrijednost u listi koordinata koje označuju te polilinije. Drugim rječima, uzimamo najbližu točku točci gledišta koja je točka linije trake.

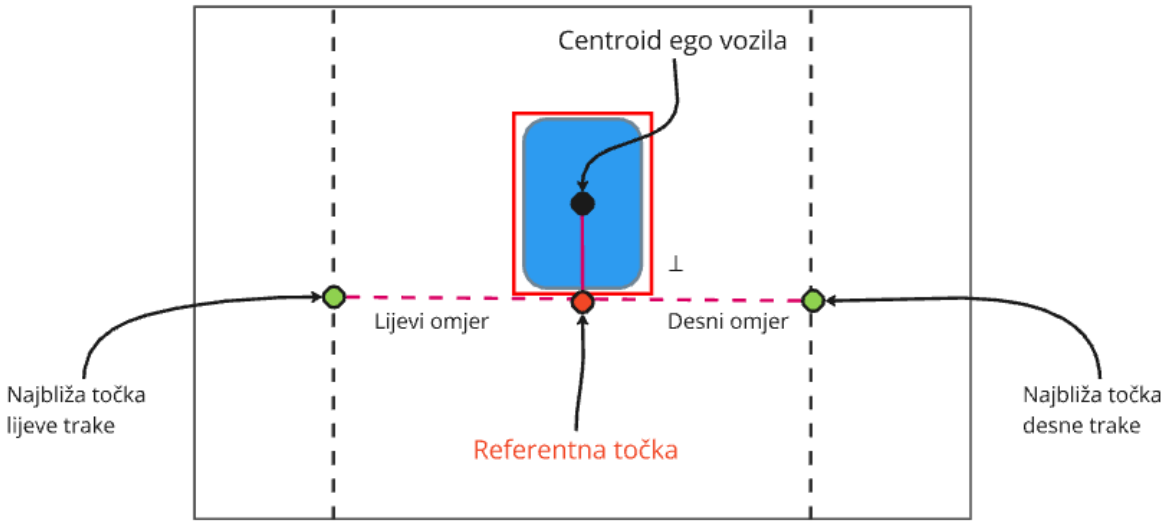
Nakon pronalaska dviju najbližih točaka linija traka potrebno je izračunati udaljenost od položaja ego vozila do njegove lijeve i desne linije trake. Nakon izračuna udaljenosti do lijeve linije trake d_{lijevo} i udaljenosti do desne linije trake d_{desno} trebamo izračunati omjer udaljenosti koji ćemo dalje koristiti.

$$L_{omjer} = \frac{d_{lijevo}}{d_{lijevo} + d_{desno}} \tag{8.3}$$

Gdje L_{omjer} označava omjer udaljenosti vozila između lijeve i desne trake, tj. poziciju središta vozila unutar trake. Potrebno je izračunati samo lijevi ili desni omjer jer njihov zbroj mora iznositi 1.

Nakon izračunatog omjera uzimamo varijable ego vozila iz liste trajektorija (x, y, w,

l, α) u istom kadru. Koordinate (x, y) predstavljaju poziciju lijevog donjeg kuta vozila, w i l su širina odnosno dužina vozila te α je kut kretanja vozila. Računamo centroid vozila (C_x, C_y) koristeći širinu, visinu i kut vozila. Koristeći taj izračunati centroid pronalazimo centroidu ega najbliže točke lijeve i desne trake u opisniku linija traka u listi trajektorija. Točka desne trake (x_1, y_1) i točka lijeve trake (x_2, y_2) opisuju pravac koji je okomit na traku te prolazi kroz centroid vozila.



Slika 8.3. Računanje omjera pomaka ega

Koristeći te dvije točke te prethodni omjer računamo točku u koju moramo pomaknuti ego vozilo kako bismo korigirali lateralni pomak.

$$x_c = x_2 + L_{omjer} * (x_1 - x_2) \tag{8.4}$$

$$y_c = y_2 - L_{omjer} * (y_2 - y_1) \tag{8.5}$$

x_c označava izračunatu vrijednost x koordinate pozicije centroida vozila, a y_c označava izračunatu vrijednost y koordinate. (x_c , y_c) označava centroid nove pozicije ego vozila u listi trajektorija. Slično kao i prije računamo lijevi donji kut vozila koristeći kut vozila i te podatke zapisujemo u listu trajektorija. Time smo ispravili lateralni pomak ego vozila.

8.2.1. Rezultati korekcije ego vozila

Evaluaciju rezultata ispavljenog pomaka računam euklidskom udaljenošću centroida vozila izračunatog kroekcijom i centroida vozila u ispravnim podacima (engl. *ground truth*) gdje su (x_c, y_c) koordinate korigiranog centroida vozila, (x_u, y_u) su koordinate centroida vozila ulaznih podataka te su (x_{gt}, y_{gt}) koordinate centroida vozila u ispravnim podacima.

$$\Delta d_{cor} = \sqrt{(x_{gt} - x_c)^2 + (y_{gt} - y_c)^2} \tag{8.6}$$

gdje je Δd_{cor} iznos pomaka korigiranog centroida i centroida vozila u ispravnim podacima.

$$\Delta d_{ulaz} = \sqrt{(x_{gt} - x_u)^2 + (y_{gt} - y_u)^2} \tag{8.7}$$

gdje je Δd_{ulaz} iznos pomaka centroida u ulaznim podatcima i centroida vozila u ispravnim podacima.

Tablica 8.1. Iznos greške lateralnog pomaka

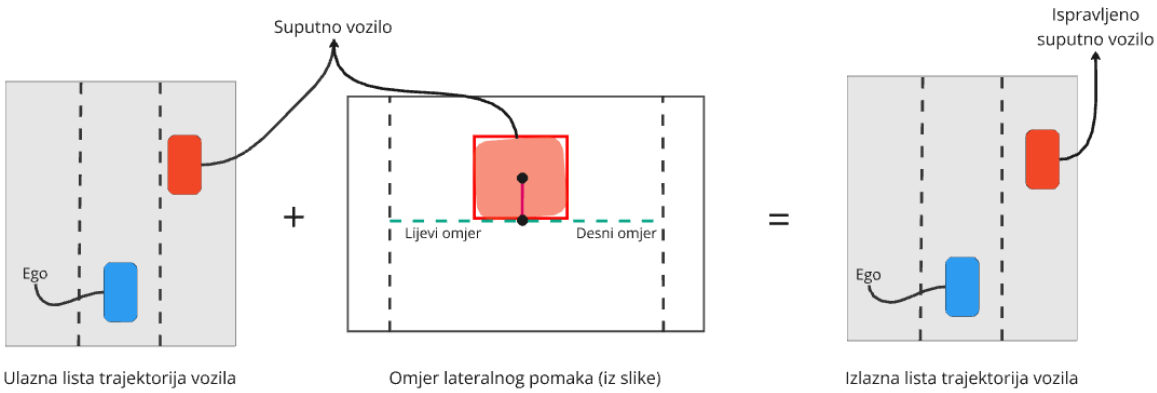
	Ulazni podatci	**Ispravljenji podatci**
Δd [cm]	20.72	7.97

U tablici 8.1. vidimo da korekcijom lateralnog pomaka dobivamo točnije podatke nego što smo imali u ulaznoj listi trajektorija vozila. U ovom dijelu rada najveću grešku u odnosu na stvarne vrijednosti donosi pogreška u detekciji linija traka. Unaprijeđenja modela detekcije linije traka će se pojaviti te time i poboljšanja performansi računanja pomaka.

8.3. Lateralna korekcija suputnih vozila

Korekcija pomaka suputnih vozila (engl. *fellow vehicles*) je slična kao i korekciji ega opisanog u prethodnom poglavlju.

Koristimo središte donjeg brida prepoznatog okvira koji je mapiran inverznom perspekcijom te koristimo linije traka kojima je također promjenjena perspektiva. Novoizračunatom centru vozila nalazimo dvije najbliže točke linija traka, jednu točku za lijevu liniju trake i jednu točku za desnu liniju trake. Računamo omjer udaljenosti do lijeve trake d_{lijevo} i udaljenosti do desne linije trake d_{desno} te koristimo formulu 8.3 za izračun omjera udaljenosti L_{omjer}. Potom koristimo podatke o vozilu u listi trajektorija za izračun najbližih točaka lijeve i desne trake te centroida vozila u podatcima te time dobijemo pravac koji je okomit na traku i prolazi kroz centroid vozila koristeći formule 8.4 i 8.5 za proračun korigiranih koordinata centroida suputnog vozila.



Slika 8.4. Proces ispravljanja lateralnog pomaka

8.3.1. Rezultati korekcije suputnih vozila

Evaluacija rezultata pomaka suputnih vozila se računa na isti način kao i ego vozilu. Koristimo euklidsku udaljenost za izračun udaljenosti pomaka ulaznih trajektorija i korigiranih trajektorija u odnosu na ispravne podatke.

Δd_{cor} je iznos pomaka korigiranog centroida od centroida u ispravnim podatcima koji je izračunat s formulom 8.6 Δd_{ulaz} je iznos pomaka centroida u ulaznim podatcima od centroida u ispravnim podatcima koji je izračunat s formulom 8.7

U tablici 8.2. možemo vidjeti izračunate lateralne pomake izražene u centimetrima [cm]. Lijevi stupac označava udaljenost suputnog vozila od ego vozila te klasificiramo suputna vozila u tri kategorije s obzirom na udaljnost od ega. Vozila koja su na udaljenosti manjoj od 10 metara, vozila koja su na udaljenosti od 10 do 30 metara i vozila koja su na udaljenosti većoj od 30 metara. U tablici je vidljivo da algoritam radi dobro

na vozilima u srednjoj traci i vozilima s velikom udaljenošću od ega jer im prepoznati okvir detekcijom objekata vjerodostojno opisuje stražnju siluetu pomoću koje računamo centar vozila. Na bliskim vozilima u susjednim trakama od ego vozila algoritam ne postiže prihvatljivu točnost, dapače čak rezultira većom greškom nego što je to u ulaznim podatcima. Razlog tomu je nemogućnost izračunavanja koordinate centra vozila.

Tablica 8.2. Rezultati korekcije suputnih vozila

	Lijeva traka		**Srednja traka**		**Desna traka**	
Udaljenost od ega	Δd_{ulaz}	Δd_{cor}	Δd_{ulaz}	Δd_{cor}	Δd_{ulaz}	Δd_{cor}
<10m	22.37	25.32	20.22	8.02	19.42	27.42
10-30m	21.32	14.52	22.31	9.24	21.34	17.23
>30m	23.12	10.34	19.52	10.25	20.72	11.03

Najveće unaprijeđenje točnosti bi donijela metoda točnog izračunavanja centra vozila iz prepoznatog okvira vozila. Detekcija linija traka kao i kod ego vozila unosi pogrešku s nekom mjerom. Preporučeno korištenje metoda ovoga rada je korekcija dalekih suputnih vozila ili vozila u istoj traci dok je bliska vozila u susjednim trakama najbolje ne korigirati.

9. Prikaz trajektorija i ceste

Prikaz trajektorija je iznimno važan zadatak za sve stranke koje sudjeluju u procesu istraživanja, testiranja sigurnosnih i optimizacijskih zahtjeva modernih vozila. Postoji potreba prikazati složene zapise trajektorija i cesti u kratkom vremenu. Brojne tvrtke imaju svoje alate za prikaz scenarija koji su složeni i sposobni prikazati scenarije na vjerodostojan način podacima. Takvi alati su sposobni zadovoljiti sve kriterije ispitivača npr. vjerodostojan prikaz kretanja vozila u trodimenzijskom prostoru. Međutim, ti alati su često računalno i vremenski zahtjevni. Postojanje alata koji može na prihvatljivi način prikazati scenarije, tj. listu trajektorija vozila i pripadajuću cestu je veoma bitno ispitivačima trajektorija koji nemaju potrebu koristiti sva svojstva složenijih alata. Ponekad je potrebno samo pregledati izgled trajektorija i kretanja vozila u scenariju bez ispitivanja složenijih svojstva. Umjesto korištenja alata koji trebaju nekoliko minuta da učitaju jedan scenarij s pripadnom cestom moguće je postići sličnu primjenu korištenjem SimplePreviewer-a.

SimplePreviewer sam razvio s namjenom pružanja mogućnosti brzog i računalno nezahtjevnog pregleda trajektorija korisnicima.

Alat je razvijen u Python-u te koristi biblioteku tkinter za GUI (engl. *Graphical User Interface*) te biblioteku matplotlib za crtanje trajektorija i ceste.

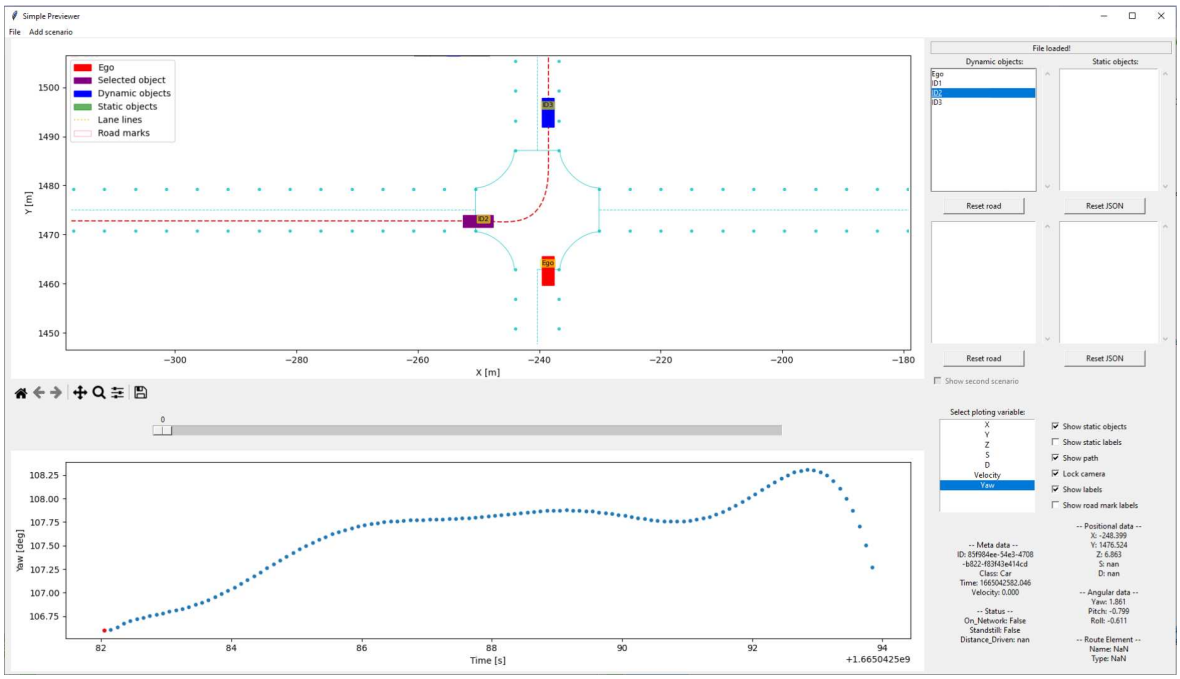
Alat pruža mogućnosti crtanja svih vozila i statičnih objekata iz liste objekata te crtanje pripadajuće ceste. Ponekad naša lista objekata ima statične objekte u prometu poput znakova, zgrada itd., sve to je potrebno vjerodostojno prikazati te pružiti mogućnost prestanka crtanja objekata u scenariju ukoliko je potrebno izdvojiti pojedine nama zanimljive objekte i prikazati samo njih.

Alat je sposoban prikazati sve kadrove pojedinog scenarija te pruža korisniku mo-

gućnost kretanja kroz kadrove scenarija koristeći klizajuću ljestvicu (engl. *sliding Scale*). Brzo pregledavanje scenarija je zahtjevno matplotlib biblioteci koja je inače namjenjena za statičke prikaze te prikaz puno objekata u kratkom vremenu predstavlja opterećenje. Ovaj problem je rješen korištenjem blit funkcije matplotlib koje pruža mogućnost bržeg osvježavanja prikaza te crtanje samo promjenjenih vrijednosti. Ovime se postiže brzina prikazivanja od 175 FPS-a (engl. *Frames per second*) koja je zadovoljavajuća za pregled scenarija u kratkom roku.

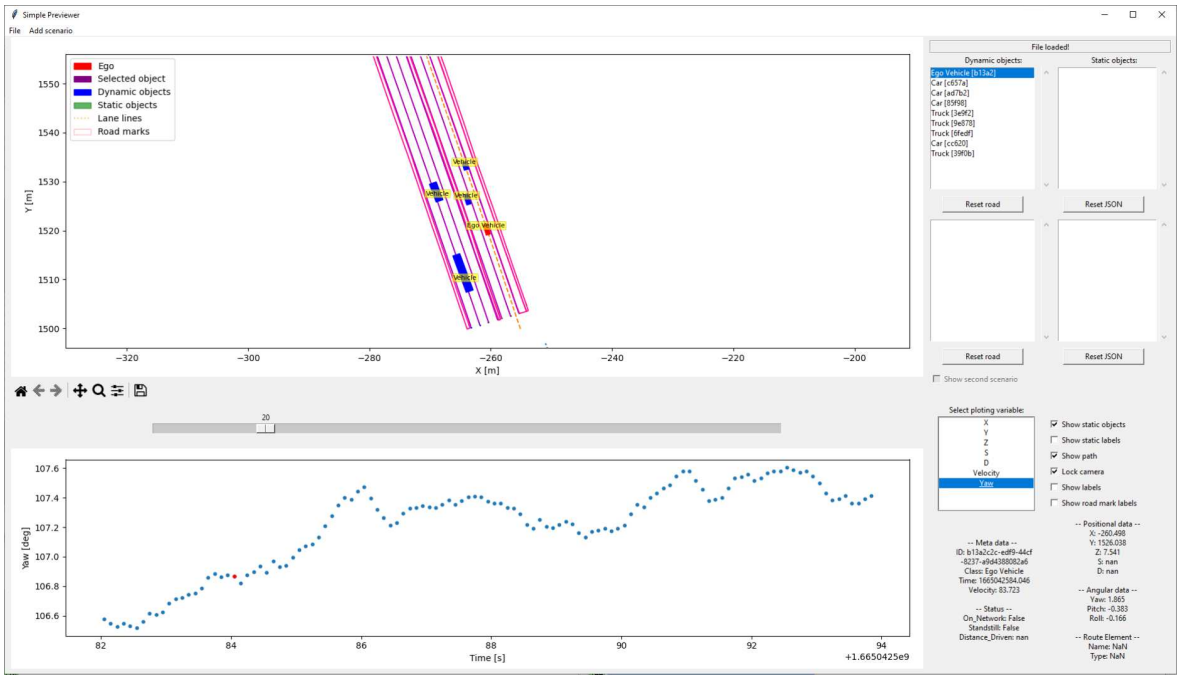
Alat ima nekoliko vrsti učitavanja koje isčitavaju samo najpotrebnije podatke iz scenarija poput listi (x, y, z, w, h, α) koje označavaju sve parametre vozila iz različitih vrsta spremljenih trajektorija. Moguće je pregledavati izglede trajektorija svih vozila te mijenjati perspektivu pogleda iz vozila u vozila klikom na vozilo u listi objekata.

Dodatni parametri vozila kroz vrijeme su prikazani u sekundarnom prikazu gdje je moguće pregledati kretanje parametara kroz svaki kadar u kojemu se pojavljuje vozilo. Moguće je pregledavati promjenu vrijednosti (x, y, z) koordinata, brzine vozila, kut orijentacije itd. kroz vrijeme trajanja scenarija.



Slika 9.1. Primjer prikaza trajektorije suputnog vozila

Na slici 9.1. moguće je vidjeti prikaz trajektorije suputnog vozila u scenariju, u donjem prikazu vidimo promjenu vrijednosti kuta kroz vrijeme. Svakom vozilu je moguće uključiti prikaz indentifikacijske oznake te time raspoznavati prikazana vozila.



Slika 9.2. Primjer scenarija nakon korekcije lateralnog pomaka

Zaključno, na slici 9.2. je prikaz jednog scenarija čijim su vozilima ispravljeni lateralni pomaci prethodno opisanim poglavljima u ovome radu. Prikaz ovim alatom omogućuje brzu provjeru valjanosti scenarija te skraćuje proces pregledavanja brojnih scenarija.

Daljnji razvoj alata uključuju cjelovitu implementaciju algoritma opisanog u ovome radu, odnosno učitavanje ulazne liste trajektorije vozila i pripadnog videozapisa scenarija te istovremenog računanja korigiranog scenarija te prikaz istog.

10. Zaključak

U sklopu ovog diplomskog rada istražena je mogućnost procjene i korekcije lateralnog pomaka vozila. Korištenjem videozapisa snimanog ego vozilom moguće je korigirati lateralni pomak u listi objekata koje opisuju trajektorije vozila u scenariju. Prvotno korištenjem modela YOLinO za detekciju linija traka otkrivamo polilinije koje opisuju linije traka za svaki kadar ulaznog videozapisa. Potom koristimo model YOLOv4 za detekciju objekata koji dodatno treniramo na skupu podataka koji sadrži slike iz prometa te tako povećavamo uspješnost modela za detekciju okvira vozila u svakom kadru vidozapisa. Objekti koje su prepoznati u videozapisu uparujemo s podudarajuće vozilo koje opisuje njihovo kretanje kroz listu trajektorija. Korištenjem parametara kamere koja je snimala scenarij moguće je inverzno transfromirati točke linija traka i pronađenih okvira objekata da se vjerodostojnije preslikavaju u prostoru. Potom računamo omjere pomaka ego vozila i suputnih vozila te ih korektiramo sukladno prethodno izračunatim omjerima. Algoritam postiže dobru korekciju za ego vozila jer jedini gubitak preciznosti se događa u modelu prepoznavanja traka. Korekcija suputnih vozila radi dobro za objekte koji su u istoj traci kao i ego te suputna vozila koja su u susjednim trakama ali na velikoj udaljenosti od ega pa njihovi okviri prepoznati detekcijom objekata odgovaraju njihovoj stražnjoj silueti vozila te time izračunata sredina okvira je odgovarajuća stvarnoj. Kod vozila koja su u susjednim trakama na bliskoj udaljenosti izračun korekcije ne daje dobre rezultate jer je teško procijeniti središnju točku vozila iz prepoznatog okvira.

Daljnja unaprijeđenja i optimizacije modela za detekciju traka i modela za detekciju objekata pružaju značajnu priliku u vidu mogućnosti poboljšanja performansi algoritma. Najveći doprinos točnosti bi donio izračun centroida objekta iz prepoznatog okvira detekcijom objekata. Cijelokupna implementacija sustava u postojeće alate poput SimplePreviewer-a bi olakšala korištenje algoritma korisnicima, budući da je on alat

koji omogućava vizualizaciju i analizu podataka, i njegova implementacija omogućuje jednostavniji pristup i interakciju s algoritmom, čime se poboljšava korisničko iskustvo.

Literatura

[1] G. Bradski, “The OpenCV Library”, *Dr. Dobb’s Journal of Software Tools*, 2000.

[2] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll’a r, i C. L. Zitnick, “Microsoft COCO: common objects in context”, *CoRR*, sv. abs/1405.0312, 2014. [Mrežno]. Adresa: http://arxiv.org/abs/1405.0312

[3] S. Yoo, H. Lee, H. Myeong, S. Yun, H. Park, J. Cho, i D. H. Kim, “End-to-end lane marker detection via row-wise classification”, 2020. [Mrežno]. Adresa: https://arxiv.org/abs/2005.08630

[4] D. J. Medium.com, “The structure of convolutional neural networks (cnns)”, https://medium.com/@dylanj07/the-structure-of-convolutional-neural-networks-cnn-9d8c5c1c8aec, 2022.

[5] R. Patel, “Comprehensive study of applying convolutional neural network for computer vision”, https://www.researchgate.net/figure/An-Example-of-Max-pooling-16_fig3_344121826, 2022., 2024.

[6] K. Senthil i V. Thulasiraman, “Ovarian cancer diagnosis using pretrained mask cnn-based segmentation with vgg-19 architecture”, *Bio-Algorithms and Med-Systems*, 09 2021. https://doi.org/10.1515/bams-2021-0098

[7] L. Karatzia, N. Aung, i D. Aksentijevic, “Artificial intelligence in cardiology: Hope for the future and power for the present”, *Frontiers in Cardiovascular Medicine*, sv. 9, 10 2022. https://doi.org/10.3389/fcvm.2022.945726

[8] J. Redmon, S. K. Divvala, R. B. Girshick, i A. Farhadi, “You only look once: Unified, real-time object detection”, *CoRR*, sv. abs/1506.02640, 2015. [Mrežno]. Adresa: http://arxiv.org/abs/1506.02640

[9] A. Bochkovskiy, C. Wang, i H. M. Liao, “Yolov4: Optimal speed and accuracy of object detection”, *CoRR*, sv. abs/2004.10934, 2020. [Mrežno]. Adresa: https://arxiv.org/abs/2004.10934

[10] A. Meyer, P. Skudlik, J. Pauls, i C. Stiller, “Yolino: Generic single shot polyline detection in real time”, *CoRR*, sv. abs/2103.14420, 2021. [Mrežno]. Adresa: https://arxiv.org/abs/2103.14420

[11] J. Redmon i A. Farhadi, “YOLO9000: better, faster, stronger”, *CoRR*, sv. abs/1612.08242, 2016. [Mrežno]. Adresa: http://arxiv.org/abs/1612.08242

[12] J. H. Hosang, R. Benenson, i B. Schiele, “Learning non-maximum suppression”, *CoRR*, sv. abs/1705.02950, 2017. [Mrežno]. Adresa: http://arxiv.org/abs/1705.02950

[13] S. Liu, L. Qi, H. Qin, J. Shi, i J. Jia, “Path aggregation network for instance segmentation”, *CoRR*, sv. abs/1803.01534, 2018. [Mrežno]. Adresa: http://arxiv.org/abs/1803.01534

[14] K. He, X. Zhang, S. Ren, i J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, *CoRR*, sv. abs/1406.4729, 2014. [Mrežno]. Adresa: http://arxiv.org/abs/1406.4729

[15] A. Bewley, Z. Ge, L. Ott, F. Ramos, i B. Upcroft, “Simple online and realtime tracking”, *CoRR*, sv. abs/1602.00763, 2016. [Mrežno]. Adresa: http://arxiv.org/abs/1602.00763

[16] Y. Pei, S. Biswas, D. S. Fussell, i K. Pingali, “An elementary introduction to kalman filtering”, 2019. [Mrežno]. Adresa: https://arxiv.org/abs/1710.04055

[17] J. Hui, “map (mean average precision) for object detection”, https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, 2018., 2024.

[18] S. Tuohy, D. O Cualain, E. Jones, i M. Glavin, “Distance determination for an automobile environment using inverse perspective mapping in opencv”, sv. 2010, 07 2010., str. 100 – 105. https://doi.org/10.1049/cp.2010.0495

Sažetak

Procjena i korekcija lateralnog gibanja vozila

Marko Turina

Ovaj diplomski rad bavi se procijenom i korekcijom lateralnog gibanja vozila u svrhu točnijih podataka. Precizna detekcija položaja unutar prometnih traka nužna je za procjene položaja i gibanja svih sudionika u prometu na određenom segmentu ceste. Pomoću kamera, prvenstveno prednje, dobiju se videozapisi vožnje. Zadatak je, analizom dobivenih slika putem algoritma detekcije linija prometnih traka i algoritmom za detekciju objekata odrediti i maksimalno točno ispraviti trajektorije na način da budu što sličniji stvarnim podacima. Opisani su načini i tehnologije koje omogućavaju u stvarnom vremenu transformirati sliku iz kamere u pogled odozgo te time otkloniti efekte izobličenja perspektive. Time je omogućeno precizno otkrivanje objekata i obilježja ceste. Rezultati pokazuju da sustav može učinkovito procijeniti i korigirati lateralni položaj vozila u scenariju te se time približiti cilju točnosti podataka.

Ključne riječi: YOLO, YOLinO, detekcija objekata, detekcija linija traka, CNN, Python

Abstract

Vehicles' lateral motion estimation and correction

Marko Turina

This master thesis deals with the estimation and correction of the lateral offset of the vehicle for the purpose of more accurate data. Precise detection of the position within traffic lanes is necessary to assess the position and movement of all road users on a certain road segment. Driving videos are obtained using the cameras, primarily the front ones. The task is to determine and correct the trajectories as closely as possible to the real data by analyzing the obtained images using the lane line detection algorithm and the object detection algorithm. Methods and technologies are described that make it possible to transform the image from the camera into a view from above in real time, thereby removing the effects of perspective distortion. This enables precise detection of objects and road features. The results show that the system can effectively estimate and correct the lateral position of the vehicles in scenario, thereby approaching the goal of data accuracy.

Keywords: YOLO, YOLinO, object detection, lane line detection, CNN, Python

A Skraćenice

OpenCV - Open Source Computer Vision Library

COCO - Common Objects in Context

CNN - Convolutional Neural Network

YOLO - You Only Look Once

IoU - Intersection over Union

CSP - Cross Spatial Partial Connections

PANet - Path Aggregation Network

SPP - Spatial Pyramid Pooling

CIoU - Complete Intersection over Union

MSE - Mean Squared Error

SORT - Simple Online and Real-time Tracking

IPM - Inverse Perspective Mapping

FPS - Frames per second

TP - True positives

TN - True negatives

FP - False positives

FN - False negatives