Planiranje kretanja autonomnog vozila na temelju dostupne globalne mape

Bajić, Jure

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

Sveučilišni studij

PLANIRANJE KRETANJA AUTONOMNOG VOZILA NA TEMELJU DOSTUPNE GLOBALNE MAPE

Diplomski rad

Jure Bajić

Osijek, 2018.

SADRŽAJ

1. UVOD

Značajnim napretkom tehnologije posljednjih godina omogućen je nagli razvoj autonomnog aspekta automobilske industrije. Pokazatelj toga jest pojava brojnih radova i implementacija ADAS (engl. *Advanced Driver-Assistance System*, ADAS) algoritama i ugradnje raznih senzora u vozila, nužnih za rad ADAS sustava, koja automatiziraju i olakšavaju vožnju. U radu je predloženo rješenje za planiranje putanje i kretanja ego vozila. Ego je ono vozilo s kojeg su očitani svi senzorski podaci i za koje se planira buduća putanja i manevri.

Planiranje kretanja autonomnog vozila predstavlja procjenu referentne putanje kojom se vozilo treba kretati kako bi stigao s trenutne pozicije na željenu. Pri tome su autonomnom vozilu na raspolaganju različiti senzori, kao npr. LIDAR (engl. *Light detection and ranging,* LIDAR) ili GPS (engl. *Global Positioning System, GPS*), ili mu je na raspolaganju globalna trodimenzionalna mapa okoline. U okviru diplomskog rada potrebno je osmisliti algoritam koji će na temelju poznatih dimenzija i dinamičkih karakteristika vozila te vlastitih koordinata procijeniti referentu putanju kojom se vozilo treba kretati. Nakon razvoja algoritma u višem programskom jeziku, potrebno je algoritam prebaciti u programski jezik C i izvršiti njegovu optimizaciju. Izvršiti implementaciju algoritma na razvojnu ploču koja sadrži napredni sustav za asistenciju vozaču njegova učinkovitost na realnoj platformi pri čemu je potrebno koristiti dostupni SDK. Rad algoritma je evaluiran kako bi se pokazala njegova uspješnost u procjeni putanje i manevriranjem vozilom, i kritički deducirale situacije u kojima vozilo griješi i time je donesen zaključak o mogućim unaprjeđenjima.

2. PREGLED POSTOJEĆIH RJEŠENJA

Već postoji mnogo odrađenih radova na istu temu, koji koriste različite metodologije i vrste pristupa rješavanja problema. U nastavku su predstavljena tri rada, izdvojena zbog svoje jedinstvenosti i načina pristupa. Svaki rad ima zajedničku karakteristiku, a to jest da je vozilo predstavljeno dinamičkim modelom, a ono što ih razlikuje je način na koji su definirani vanjski čimbenici, kreiranje putanje i set pravila za odabir najbolje.

2.1. Odabir putanje vozila korištenjem statističkog modela

U literaturi [1], predlaže se metoda dinamičkog planiranja putanje autonomnog vozila u sustavu prometa koji je modeliran kao statistički sustav. Statističke karakteristike vozila u pokretu analizirane su s modelom vozila, u kojem su ulazi definirani kao slučajne varijable s određenom razdiobom. Prema tome se može odrediti izlaz modela pomoću necentrirane transformacije (engl. *unscented transformation for probabilistic spread)*. Zatim je sveukupna vjerojatnost sudara putanja procijenjena s razinom pouzdanosti. Na kraju je metodom planiranja putanje dobiven mogući broj putanja s određenim pripadnim vjerojatnostima. Kretanje vozila je definirano sljedećom jednadžbom:

$$\dot{x} = f[x(t), u(t)], \tag{2-1}$$

gdje $x(t)$ predstavlja nepoznanicu stanja, kao npr. brzina ili ubrzanje vozila u određenom vremenu, dok su $u(t)$ predstavlja utjecaj ulaznih varijabli vozača kao npr. kočenje, upravljanje vozilom volanom, dodavanje gasa također u određenom vremenu na dinamiku vozila. Što znači da $x(t)$ predstavlja trenutno stanje vozila u kojem se ono nalazi, a $u(t)$ predstavlja onaj dio koji će promijeniti to stanje. $u(t)$ je zamišljen kao utjecaj svih vozačevih kontrola na dinamički model vozila te prema tome je f jednadžba dinamike vozila. Izraz (2-1) je diferencijalna jednadžba gdje $\dot{x}$ predstavlja kretanje vozila. Prema tome moguće je predvidjeti buduće stanje vozila u bilo kojem vremenu τ izrazom:

$$\dot{x}(\tau) = x(0) + \int_0^{\tau} f(x(t), u(t))dt. \tag{2-2}$$

Može se primijetiti kako su buduća stanja određena s $x(0)$, $u(t)$ i f, gdje je $x(0)$ početno stanje, $u(t)$ ulazni parametri u vremenu τ. Općenito, $x(0)$ se dobiva kroz senzore ego vozila, što se može smatrati poznatom informacijom. Također je i tip vozila poznat pa se parametri vozila mogu procijeniti, prema tome jedino je $u(t)$ ostaje nepoznat te je jedini nepoznat faktor koji utječe na buduće stanje vozila. Na putanju vozila puno veći utjecaj ima longitudinalno ubrzanje a_y, nego lateralno ubrzanje a_x, te je zbog toga dinamički model vozila pojednostavljen kao model s jednim ulaznim parametrom što je kut prednjih kotača δ, a brzina se uzima kao konstanta. Upravo taj kut kotača vozila modeliran je Gaussovom razdiobom. Zatim se nelinearnom transformacijom kuta δ dobivene koordinate u Kartezijevom koordinatnom sustavu. U literaturi [1], pokazano je kako se a_y može linearno modelirati i prikazati Gaussovom razdiobom. Funkcija određivanja putanje daje srednju putanju zajedno s informacijom o nesigurnosti same putanje.

Predstavljeno rješenje generalizira i pojednostavljuje problem planiranja putanje stavljajući naglasak na manevre prilikom izmjene traka. Iz seta putanja koji se dobije, svaka ima mjeru vjerojatnosti sudara, kao i mjeru pouzdanosti u putanju. Iz seta putanja odabire se najbolja tako što se u obzir uzimaju optimizirani ciljevi, efikasnost i udobnost manevra kojeg putanja izvodi.

2.2. Planiranje putanje u urbanim sredinama

U literaturi [2], predstavljen je algoritam za planiranje kretanja vozila koristeći autonomnu navigaciju u urbanim sredinama. Rad je nadogradnja postojećeg algoritama koji se temelji na A*(engl. *A star*) algoritmu već prethodno korištenim za nalaženje puta u grafu, i traženje elementa u strukturama podataka temeljenim na grafovima. Autori predlažu rješenje za autonomnu navigaciju u urbanim sredinama, budući da klasični pristupi ne uspijevaju upravljati vozilom u sredinama gdje su ceste preuske, oštri zavoji i nepregledni teren s nepredvidivim brojem vozila na cesti. Predložen je algoritam koji generira putanju vozila koja je glatka i točna za kompleksne sredine. Najpopularnija metoda, na kojoj se rad bazira, je generiranje seta putanja te odabira jedne putem funkcija koje daju težinu svakoj putanji (engl. *Cost function*). Jedan od pristupa za generiranje putanja jest rješavanje BPV (engl. *Boundary problem value*) problema korištenjem numeričke optimizacije. U svakom koraku je simuliran model vozila čiji su ulazi dani polinomom. Linearizirajući i invertirajući model vozila, parametri polinoma postavljeni su s

obzirom na odstupanje trenutnog rješenja od traženog. Drugo rješenje jest generiranje putanje s različitim bočnim odstupanjima od središta ceste, podešavanjem parametara kuta prednjih kotača vozila.

Za navigaciju u različitim sredinama koristi se A* algoritam za planiranje putanje, koji pronalazi putanje bez sudara koristeći kartu prepreka trenutne sredine koja je podijeljena na odsječke asociranim s vremenskim intervalom koji govore u kojem vremenu su blokirani. Digitalna karta korištena je za formiranje stabla čvorova koji predstavljaju konfiguraciju vozila (x, y, ψ, c_o, v), gdje (x, y, ψ) predstavljaju dvodimenzionalne koordinate i kut, c_o zakrivljenost, a v brzinu vozila. Ukoliko se vozilo kreće naprijed, jedna polovica čvorova potomaka zadržava konstantu brzinu v dok druga ima smanjenu brzinu dobivenom primjenom negativnog ubrzanja za L metara. Efekt simuliranog manevra kočenja jest da su konfiguracije za oštre zavoje prikazane u potomcima čvora, što je bitno ukoliko vozilo nailazi na oštre zavoje.

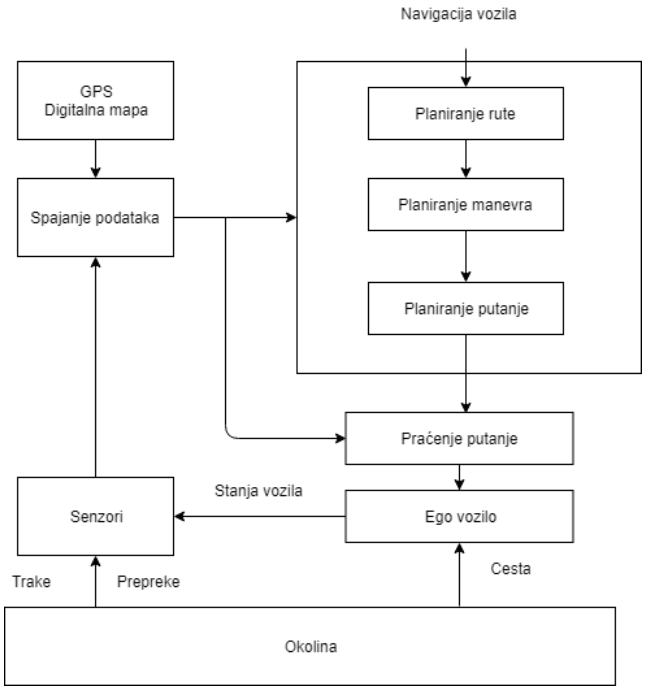
Dvodimenzionalne koordinate i zakrivljenost čvorova izračunate su deterministčkim uzorkovanjem B mjera zakrivljenosti $c_1^1, \dots, c_1^B$. Zatim su određene krajnje koordinate i zakrivljenosti svih potomaka čvora sa zadanom početnom zakrivljenošću c_0^i s $i \in \{1, \dots, B\}$, i mjerom promjene. Zakrivljenosti su odabrane tako da su dva uvjeta ispunjena: konačna zakrivljenost ne smije premašiti maksimalnu zakrivljenost koju nameću kinetička ograničenja vozila i njeno maksimalno bočno ubrzanje te nužna promjena kuta kretanja ne smije prijeći maksimalnu mjeru zakretanja vozila. Nakon izračuna svih konfiguracija čvorova, slijedi provjera putanja za sudare i izračun težina svake, s obzirom na njihove zakrivljenosti, blizini prepreka i mjerom zakretanja, te se svakoj putanje pridaje određena težina.

Tako dobivena struktura korištena je za generiranje putanje korištenjem postupaka koji autori nazivaju *one shot expansion,* čiji je cilj pronaći putanju vozila od trenutnog do odredišnog čvora u jednom koraku. To je korisno u slučajevima kada je A* ekspanzija spora u određivanju putanje. Drugi razlog korištenja iste metode jest zato što se dobiva glađa putanja do cilja, te odabir čvorova je puno precizniji. Za spajanje početnog i krajnjeg čvora koristi se Newtonova metoda za podešavanje parametara kubnog polinoma dok se ne spoje konfiguracije početnog čvora $(x^{start}, y^{start}, \psi^{start}, c_0^{start})$, s željenim čvorom konfiguracije $(x^{goal}, y^{goal}, \psi^{goal}, c_0^{goal})$, pritom imajući određenu preciznost. Iako proces namještanja polinoma može biti neisplativim, može se brzo odrediti i daje dobru početnu vrijednost koja se koristi u optimizacijskom koraku koji generira putanju. Istu tu putanju simulira kroz model vozila kako bi se odredila je li ista izvodiva.

2.3. Planiranje dinamičke putanje za autonomna vozila

U literaturi [3], objašnjena je metoda dinamičkog planiranja putanje, u svrhu brzog reagiranja na promjene u okružju, ali i kako bi se optimizirala ravnoteža između performansi i efikasnosti vožnje. Generalizacija ovog algoritma svodi se na dva manevra, zadržavanje trake i promjena trake.

Planiranje putanje podijeljeno je na tri razine koje izvršavaju različite zadatke, prva je planiranje rute na makro razini, druga jest planiranje manevra na mezo razini i zadnja je planiranje putanje na mikro razini (Sl.2.1.). Planiranje manevra generira set uputa za vozila koje definira prethodni proces planiranja rute. Svi se manevri sastoje od dva već spomenuta (zadržavanje trake i promjena trake) ili njihove kombinacije. Dok planiranje putanje definira niz točaka koje vozilo mora slijediti s vremenskom skalom reda mikro sekundi.



Sl. 2.1. *Prikaz arhitekture navigacije vozila, [2]*

Za generiranje putanje korišten je kut kretanja vozila, te se korištenjem kinetičkog modela vozila generira optimalna putanja. Zatim je provjerena dobivena putanja kako ne bi došlo do sudara na putanji s preprekama u okolini. Također se radi i procjena ostalih vozila u okolini kako bi se odredile i njihove putanje. Pozicija vozila, tj. koordinate vozila su smatrane kao dane informacije, odnosno rad se ne bavi problematikom lokalizacije. Ukoliko su poznate dinamičke

karakteristike vozila (brzina, kut prednjih kotača), moguće je predvidjeti budući položaj vozila $(x_{pre}, y_{pre}, \theta_{pre})$ koristeći procjenu unutar kratkog vremenskom perioda od τ, uz pomoć kinetičkog modela vozila koristeći izraze:

$$x_{pre} = u\tau_1 cos\theta - R_s(1 - \cos(r_s\tau_1))sin\theta, \quad (2\text{-}3)$$

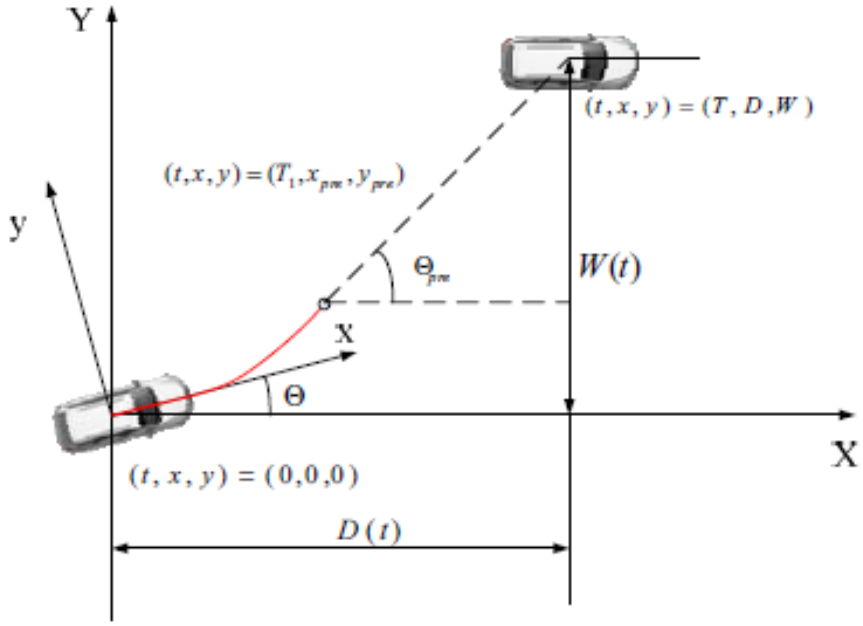
$$y_{pre} = u\tau_1 sin\theta + sgn(\delta)R_s\big(1 - cos(r_s\tau_1)\big)cos\theta, \quad (2\text{-}4)$$

$$\theta_{pre} = \theta + r_s\tau_1, \quad (2\text{-}5)$$

gdje je:

- x_{pre} - položaj vozila na apscisi,
- y_{pre} - položaj vozila na ordinati,
- θ – kut koji predstavlja orijentaciju vozila,
- r_s- mjera brzine vrtnje,
- τ- vremenski interval,
- u – brzina vozila.

Prikaz svih parametara prikazan je na slici 2.2. Iz dobivenih jednadžbi se nadalje mogu izračunati optimizirani kut upravljanja iz kojega se dobiva optimalna putanja vozila pri određenoj brzini. Iz dobivenih podataka definirane su nove koordinate vozila koje se koriste u sljedećoj iteraciji.



Sl. 2.2. *Prikaz parametara vozila bitnih za određivanje putanje, [2]*

Osim planiranja putanje ego vozila, potrebno je predvidjeti i ponašanje ostalih sudionika u prometu što je jako zahtjevno, zato što je potrebno prikupiti podatke o okolini putem različitih senzora, te ovisno o broju sudionika prometa (što može biti raznolik raspon) odrediti njihove putanje. Problem ovakvog pristupa leži upravo u nepredvidivosti proračuna zbog nepoznatog broja sudionika.

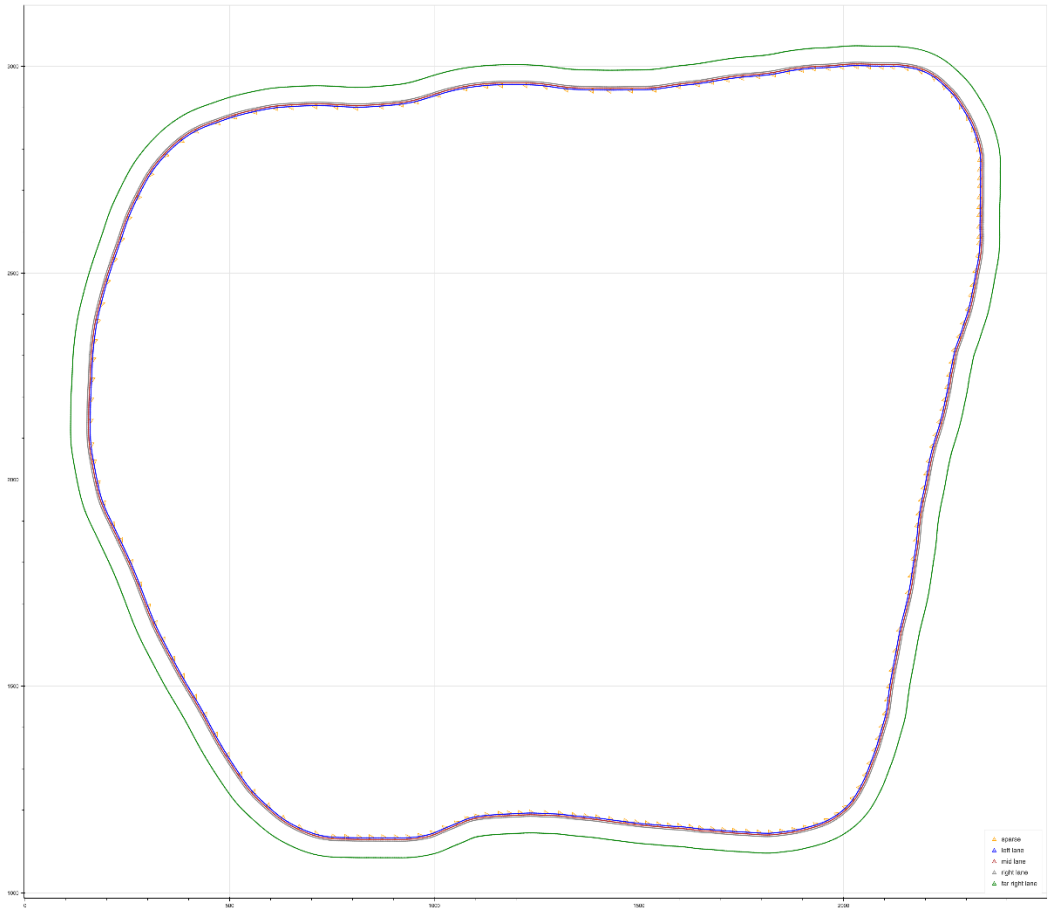
3. RAZVOJ VLASTITOG RJEŠENJA

Nakon analize gotovih radova te procjene njihovih metoda i relevantnosti rješenja, potrebno je po uzoru na ista, osmisliti vlastito rješenje. Svi radovi su zasnovani na stvarnim senzorskim vrijednostima i podacima, ili su samo teoretski pristupili rješavanju problema. Kako bi izolirali problematiku planiranja putanje vozila, nužno je pretpostaviti točnost dobivenih vrijednosti od senzora.

3.1. Razvojno okruženje

Za potrebe razvijanja i testiranja algoritma odabran je *term3_sim_windows* simulator koji se nalazi u P.3.1. Osim odabranog, postoje i drugi simulatori kao npr. *Microsoftov* simulator *AirSim* koji se temelji na *Unreal Engineu*. *Unreal Engine* je isto jedan od simulatora, ali kojeg je potrebno dodatno prilagoditi za potrebe razvijanja algoritma. Najveći nedostatak svih neodabranih simulatora jest količina resursa koje zahtijevaju kako bi simulirali vozilo u određenoj sredini, te je upravo zbog tih ograničenja je odabran *Udacityev* simulator koji ima najmanje zahtjeve.

Simulator postavlja određene uvjete kojih se vozilo mora pridržavati kako bi se vožnja smatrala uspješnom. Potrebno je pridržavati se ograničenja brzine $80{,}4672\ km/h$ (prema simulatoru $50\ mph$), apsolutna vrijednost ubrzanja ne smije prelaziti $10\ m/s^2$, a također i promjena ubrzanja po vremenu, zvana trzaj (engl. *jerk*) vozila,. Također se ne smije prelaziti vrijednost $10\ m/s^3$. Velika razlika u ubrzanju predstavlja prenaglo kočenje ili ubrzavanje vozila, a razlika trzaja predstavlja mjeru ugodnosti i laganosti vožnje. Cilj je da vozilo vozi konstantnom brzinom od $80{,}4672\ km/h$, pritom pretječući vozila koja se kreću manjom brzinom, ali je uz to uzeto u obzir da će i druga vozila mijenjati trake. Uz simulator je priložena datoteka koja predstavlja lokalizacijske podatke vozila, koja se nalazi u P.3.2. Datoteka sadrži listu $[x, y, s, dx, dy]$ gdje vrijednosti x, y predstavljaju koordinate odredišta na karti, s je udaljenost uz cestu do tog odredišta u metrima, a dx i dy su jedinični vektori u smjeru normale na stazu brze ceste. Prikaz modela brze ceste vidljiv je na slici 3.1., gdje se može primijetiti da je brza cesta eliptičnog oblika. Prema literaturi [4], duljina staze jest $6945{,}554\ m$. Za komunikaciju između simulatora i algoritma korištena je biblioteka *uWebSockets*, koja putem w*ebsocket* protokola komunicira s algoritmom, te međusobno izmjenjuju podatke.



Sl. 3.1. *Prikaz karte brze ceste iz ptičje perspektive, [5]*

Podatke koje simulator daje algoritmu predaju se kao JSON struktura prikazanim u sljedećem izrazu:

$$[x, y, s, d, yaw, speed, previous\_path\_x, previous\_path\_y,$$

$$end\_path\_s, end\_path\_d, sensor\_fusion], \tag{3-1}$$

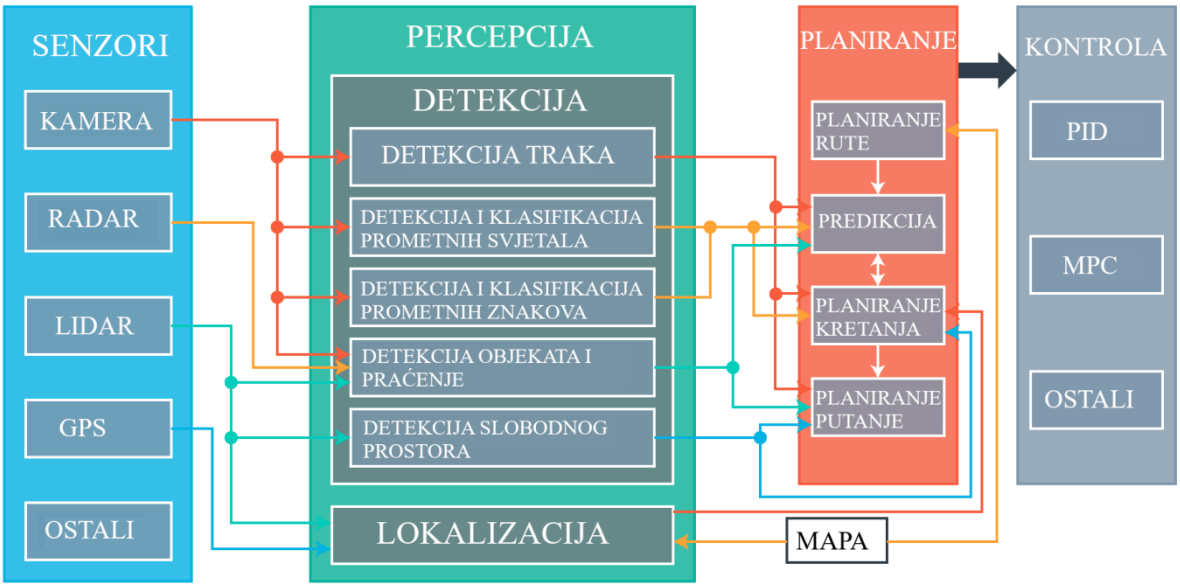
gdje je:

- x - pozicija vozila na apscisi
- y - pozicija vozila na ordinati,
- s - pozicija vozila u Frenetovim koordinatama,
- d - pozicija vozila u Frenetovim koordinatama,
- yaw - kut vozila s obzirom na apscisu na karti,
- $speed$ - brzina vozila u mph jedinici,

- $previous\_path\_x$ - prethodna lista točaka na apscisi dana simulatoru, s uklonjenim već prijeđenim točkama,
- $previous\_path\_y$ - prethodna lista točaka na ordinati dana simulatoru, s uklonjenim već prijeđenim točkama,
- end_path_s - prethodna lista s točaka dana simulatoru, s uklonjenim već prijeđenim točkama, s uklonjenim već prijeđenim točkama,
- end_path_d - prethodna lista d točaka dana simulatoru, s uklonjenim već prijeđenim točkama,
- $sensor\_fusion$ - lista seta vrijednosti $[id, x, y, dx, dy, s, d]$ za sva okolna vozilo gdje je:
    - id - identifikator vozila,
    - x - pozicija vozila na apscisi,
    - y - pozicija vozila na ordinati,
    - dx - brzina vozila u smjeru osi apscise iskazane u m/s,
    - dy - brzina vozila u smjeru osi ordinate iskazane u m/s,
    - s - pozicija vozila u Frenetovim koordinatama,
    - d - pozicija vozila u Frenetovim koordinatama.

Dobivenim rezultatima predstavljen je rezultat mjerenja brojnih senzora na vozilu koje u ovom slučaju simulator zamjenjuje. Ego vozilo u simulatoru koristi savršeni upravljač te će posjetiti sve točke (x, y) koje su mu predane u listi svakih $0{,}02$ sekundi. Koordinate točaka iskazane su u metrima te razmak između točaka određuje brzinu vozila. Vektor od jedne do druge točke određuje kut kretanja vozila. Simulator cijelo vrijeme mjeri ubrzanje u tangencijalnom i ubrzanje u smjeru normale na stazu, te promjenu sveukupnog ubrzanja, što je u biti trzaj. Simulator kao ulazne podatke traži listu točaka (x, y), koji algoritam treba proslijediti nakon proračuna.

Također treba uzeti u obzir kašnjenje zbog komunikacije između simulatora i algoritma budući da je algoritmu potrebno nezanemarivo vrijeme kako bi odredio putanju vozila. Na slici 3.2. jest vidljivo što sve prethodi dobivanju obrađenih senzorskih podataka, odnosno proces koji simulator zamjenjuje bez potreba korištenja kamera, senzora, te obrade podataka algoritmima za detekciju i klasifikaciju objekata, za detekciju brzine vozila.



Sl. 3.2. *Struktura senzora i algoritama potrebnih za rad autonomnog vozila, [5]*

Prije samog planiranja putanje vozila, nužno je imati osnovne informacije o okolini u kojoj se vozilo nalazi. Za senzore je potrebno imati kamere i radar koji služe za detekciju traka, detekciju i klasifikaciju prometnih znakova i svjetala, detekciju objekata i njihovo praćenje, LIDAR senzore i GPS za lokalizaciju. LIDAR, prema literaturi [6], jest senzor sličan sonaru i radaru, ali za razliku od njih koji koriste zvuk, odnosno radiovalove, LIDAR koristi svijetlo kako bi skenirao i prikazao svoje okružje kao trodimenzionalni model. Jedini nedostatak tog senzora jest njegova cijena, i zato se ne pojavljuje u komercijalnim vozilima. Koristeći LIDAR i GPS, odnosno lokalizacijske podatke koji se dobivaju od njih, formira se trodimenzionalna digitalna karta koja se koristi prilikom planiranja trajektorije i planiranja manevriranja vozila.

Kako bi se algoritam mogao prevesti i nakon toga pokrenuti potrebni su sljedeći programi:

- *cmake*, inačice 3.5,
- *make*, inačice 4.1,
- *gcc/g++*, inačice 5.4,
- *uWebSockets*, inačice označenom identifikatorom e94b6e1.

Za razvoj, pokretanje i testiranje algoritma korišteno je računalo sljedećih specifikacija:

- Operacijski sustav: Windows 10, Education, 64 bitni,
- Procesor: Intel(R) Core(TM) i7-47900 CPU @ 3,60 GHz,
- RAM: 16 GB,
- Grafička kartica: Intel(R) HD Graphics 4600.

Procesna snaga računala je bitna za rad algoritma, a grafička kartica za radi simulatora te njegovu obradu grafike i upravljanja vozilom.

3.2. Algoritam

Algoritam započinje učitavanjem datoteke */data/highway_map.csv* iz P.3.2., koja sadrži odredišta na koje vozilo mora stići, svako razmaknuto približno $30\,m$. Nakon toga slijedi dvosmjerna komunikacija korištenjem *webscoket* protokola. Tijekom komunikacije simulator i algoritam međusobno izmjenjuju podatke, a brzina komunikacije ovisi o procesnoj moći algoritma, odnosno o vremenu u kojem isti stigne obraditi podatke, i o raspoloživim resursima platforme na kojoj se algoritam i simulator pokreću. Nakon toga slijedi interpretacija podataka od simulatora objašnjenim u izrazu (3-1) te se postavljaju parametri ego vozila koje simulator predaje.

Zatim je određena najbliža odredišna točka s obzirom na dane koordinate vozila na karti. Udaljenost je razmatrana euklidskom vrijednošću, tj. odredište s najmanjom euklidskom vrijednošću, uzima se kao najbliže te je dalje u postupku razmatrana iduća odredišna točka. U daljnjem koraku procjenjuje se je li pronađena najbliža točka, trenutno odredište ili točka ispred nje. U izrazu (3-2) prikazano je kako se određuje kut kojim se vozilo treba kretati, gdje su map_y i map_x koordinate najbližeg odredišta, a x i y koordinate vozila. Zatim je u izrazu (3-3) određena apsolutna razlika kutova, gdje je θ kut vozila na globalnoj karti, što odgovara već prethodno spomenutoj vrijednosti yaw u izrazu (3-1), a $heading$ prethodno određen kut između trenutne pozicije vozila i iduće:

$$heading = \arctan\left(\mathrm{map_y} - \mathrm{y}, \mathrm{map_x} - \mathrm{x}\right), \tag{3-2}$$

$$angle = abs(\theta - heading). \tag{3-3}$$

Zatim se pet prethodnih i pet sljedećih odredišnih točaka ubacuje u set točaka koje se moraju nalaziti u putanji, kako bi dobivena putanja bila što glađa. Set od pet točaka odabran je

proizvoljno, broj pet jest odabran empirijski, kako ne bi došlo do zauzimanja previše procesne moći nije uzet veći broj, a zbog problematike ne dobivanja glatke putanje nije uzet manji broj.

Slijedi određivanje broja točaka koje treba interpolirati, prema izrazu (3-4), što se određuje dijeljenjem razlike zadnje i prve točke s Frenetovog koordinatnog sustava koje su prethodno ubačene u set odredišnih točaka $waypoints\_s$, s interpolacijskim parametrom koji određuje udaljenost između buduće interpoliranih točaka i iznosi 0,5 te dobiven je empirijskim putem. Te točke predstavljaju odredišta koja vozilo mora posjetiti na svom putu. Interpolacija služi kako bi se upotpunio set točaka između početne i krajnje, koristeći određenu funkciju kojom određujemo prijelaz između točaka. Interpolirane vrijednosti su s, x, y, dx i dy. Vrijednosti s se interpoliraju koristeći već izračunati parametar num_inter_points prema izrazu (3-4). Vrijednost $dist\_inc$ postavljena je proizvoljno na $0{,}5\ m$, budući da će udaljenost između svake točke biti upravo tolika.

$$num\_inter\_points = (\text{waypoints\_s[last]} - \text{waypoints\_s[first]})/\text{dist\_inc}, \qquad (3\text{-}4)$$

gdje je $waypoints\_s$ set točaka koje vozilo mora posjetiti, dok indeksi $last\ i\ first$ predstavljaju poziciju u tom setu. Interpolacija ostalih točaka računa se kubnom interpolacijom prema literaturi [7], čije rješenje je iskorišteno u ovom radu. Kubna interpolacija za dani set točaka $\{(s_1, y_1), \dots, (s_n, y_n)\}$ za koji vrijedi $s_1 < s_2 < \cdots < s_n$ definira kubni polinom $f_1, \dots, f_{n-1}$ s $f_i: [s_i, s_{i+1}] \to \mathbb{R}$, gdje su x vrijednosti Frenetovog koordinatnog sustava koje su prethodno definirane, a y predstavlja vrijednosti za koje se računaju interpolirane točke, tj. s, x, y, dx i dy. Kubni polinom prikazan je u izrazu (3-5):

$$f(s) = a_i(s - s_i)^3 + b_i(s - s_i)^2 + c_i(s - s_i) + y_i, \qquad s\epsilon[s_i, s_{i+1}]. \qquad (3\text{-}5)$$

U izrazu (3-5) a_i, b_i, c_i i y_i predstavljaju koeficijente kubnog polinoma koje se trebaju izračunati. Prilikom proračuna potrebne su derivacije prvog i drugog stupnja izraza (3-6) prikazani u sljedećim formulama:

$$f'(s) = 3a_i(s - s_i)^2 + 2b_i(s - s_i) + c_i, \qquad (3\text{-}6)$$

$$f''(s) = 6a_i(s - s_i) + 2b_i. \qquad (3\text{-}7)$$

Zatim je konstruirana matrica potrebna za izračun koeficijenata polinoma prikazana u izrazu (3-8), što predstavlja kubnu interpolacijsku matricu.

$$S(s) = \begin{cases} a_0(s-s_0)^3 + b_0(s-s_0)^2 + c_0(s-s_0) + y_0 \\ a_1(s-s_1)^3 + b_1(s-s_1)^2 + c_1(s-s_1) + y_1 \\ \vdots \\ a_n(s-s_n)^3 + b_n(s-s_n)^2 + c_n(s-s_n) + y_n \end{cases}. \quad (3\text{-}8)$$

Interpolacijska matrica S je bitna zbog izračuna koeficijenata polinoma, jer se njenim rješavanjem dobivaju koeficijenti polinoma, a prema literaturi [8] interpolacijska matrica zadovoljava sljedeća svojstva:

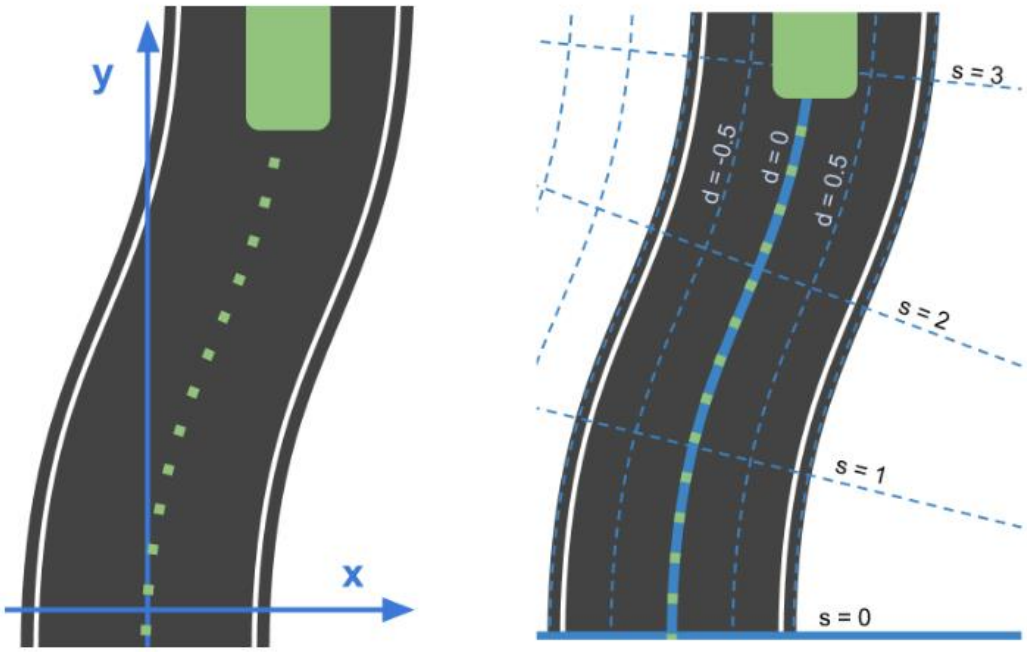
$$\begin{aligned} S(s_j) &= f(s_j)\ za\ j = 0,1,\dots,n, \\ S_j(s_{j+1}) &= S_{j+1}(s_{j+1})\ za\ j = 0,1,\dots,n-2, \\ S_j'(s_{j+1}) &= S_{j+1}'(s_{j+1})\ za\ j = 0,1,\dots,n-2, \\ S_j''(s_{j+1}) &= S_{j+1}''(s_{j+1})\ za\ j = 0,1,\dots,n-2. \end{aligned} \quad (3\text{-}8)$$

Nakon izračuna interpoliranih točaka slijedi određivanje parametara ego vozila. Struktura vozila sastoji se od pripadajućih Frenetovih koordinata (s, d) i njihovih pripadnih derivacija prvog i drugog stupnja po vremenu, što predstavlja brzinu i ubrzanje vozila. Ukoliko nema dovoljno točaka od prethodnog puta za formiranje parametara vozila, koriste se zadane vrijednosti, odnosno vozilo zadržava brzinu i kut kretanja. Ali ukoliko postoji dovoljan broj točaka, određuju se Frenetove koordinate iz već postojećih (x, y) koordinata Kartezijevog koordinatnog sustava i s interpoliranih vrijednosti. Frenetov koordinatni sustav korišten je zbog određivanja putanje vozila, zato što je po koordinatama iz Kartezijevog koordinatnog sustava teško zaključiti hoće li doći do sudara između vozila, na slici 3.3. možemo vidjeti jasnu razliku između dvaju sustava. Vidljivo je kako je sa s predstavljena udaljenost na samoj stazi, a d udaljenost od središnje linije. Prema literaturi [9], korištenjem takvog koordinatnog sustava sudar vozila se može provjeriti usporedbom d vrijednosti, budući da sva vozila voze prateći pripadnu traku.

Definiranje Frenetovog koordinatnog sustava vrši se pomoću Frenet-Serret formula prema literaturi [10], danim u izrazu (3-10). Takav sustav je intrinzičan odnosno ovisan o točki promatranja. Korišteni predložak za pisanje algoritma dan od *Udacitya* već ima definirane funkcije transformacije u Frenetove koordinate s određenim koeficijentima koji su predodređeni od strane *Udacitya*.

$$\frac{dT}{ds} = \kappa N,$$
$$\frac{dN}{ds} = -\kappa T + \tau B, \tag{3-10}$$
$$\frac{dB}{ds} = -\tau N,$$

gdje je T jedinični vektor u smjeru tangente na krivulju, N jedinični vektor normale u odnosu na krivulju, i B jedinični vektor koji je rezultat vektorskog produkta TxB, κ je mjera zakrivljenosti krivulje, τ mjera promjene zakrivljenosti krivulje, a s predstavlja duljinu puta.



Sl. 3.3. *Razlika između Kartezijevog i Frenetovog koordinatnog sustava na primjeru ceste, [9]*

Transformacijom (x, y) koordinata u Frenetove dobivamo ego parametre vozila (s, d), potom su računate brzine kao što je prikazano u izrazu (3-11).

$$v_{x1} = \frac{(pos_{x1} - pos_{x2})}{path\_dt}.$$
$$v_{y2} = \frac{(pos_{y1} - pos_{y2})}{path\_dt}. \tag{3-11}$$

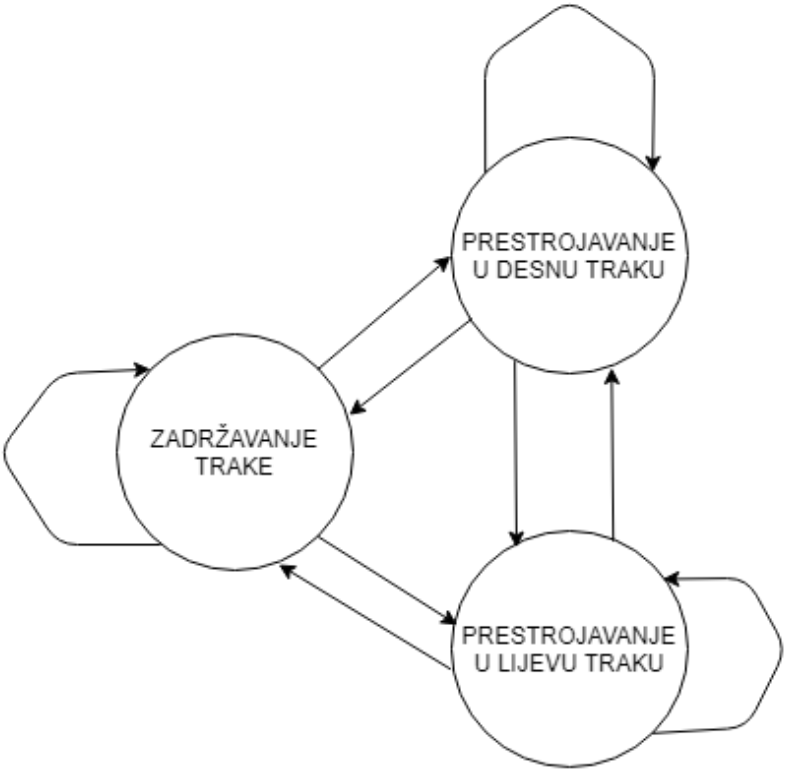
Brzine su određene u Kartezijevom koordinatnom sustavu, gdje su $pos_{x1}, pos_{x2}, pos_{y1}, pos_{y2}$ koordinate na pripadnim osima, preuzete kao posljednja i pretposljednja točka iz seta već interpoliranih točaka, a $path\_dt$ je konstanta $0{,}02\,s$ koja predstavlja vrijeme nakon kojeg simulator obrađuje svaku zadanu točku. Te iste brzine su transformirane u vektore Frenetovog sustava koristeći jedinične vektore dx, d_y koji su preuzeti kao posljednje točke interpoliranih točaka, i njima okomite vektore s_x, s_y koji zadovoljavaju sljedeću jednkakost $s_x = -d_y, s_y = d_x$ izrazom (3-12).

$$\begin{aligned} s' &= v_{x1} * s_x + v_{y1} * s_y, \\ d' &= v_{x1} * dx + v_{y1} * d_y. \end{aligned} \tag{3-12}$$

Kako bi se odredilo ubrzanje vozila u Frenetovom sustavu potrebna je još jedna interpolirana točka za koju se računa brzina. Nakon proračuna brzine određeno je ubrzanje prvo u Kartezijevom koordinatnom sustavu, a nakon toga i u Frenetovom koordinatnom sustavu prema izrazu (3-13).

$$\begin{aligned} a_x &= \frac{v_{x1} - v_{x2}}{path\_dt}, \\ a_y &= \frac{v_{y1} - v_{y2}}{path\_dt}, \\ s'' &= a_x * sx + a_y * s_{y,} \\ d'' &= a_x * dx + a_y * d_y. \end{aligned} \tag{3-13}$$

Nakon određivanja svih parametara ego vozila, generiraju se putanje za ostala vozila čiji su podaci dobiveni od simulatora. Iz dobivenih podataka određuje se brzina, te se generira lista predviđenih (s, d) koordinata uz pretpostavku da će vozilo zadržati svoju dosadašnju brzinu, te se iste predviđene putanje spremaju i uspoređuju s putanjom ego vozila. Vozilo se ponaša kao automat konačnih stanja (engl. *Finite State Machine*, FSM), te se može nalaziti u tri moguća stanja, u stanju zadržavanja trake (engl. *Keep lane,* KP), prestrojavanje u desnu traku (engl. *Lane change right,* LCR) i stanje prestrojavanja u lijevu traku (engl. *Lane change left*, LCL), što je prikazano na slici 3.4.



Sl. 3.4. *FSM stanja vozila*

Usporedbom koordinata ego vozila s koordinatama drugih, određeno je postojanje drugih vozila u neposrednoj blizini ego vozila, na dovoljnoj velikoj udaljenosti, kako bi ograničili kretanje ego vozila. Objašnjeni dio algoritma prikazan je pseudo kodom u nastavku s konkretnim vrijednostima, izraženim u metrima:

```
Za svaki detektirani auto:
razlika_s = ABS(auto.s – ego_auto.s)
Ako je razlika_s < UDALJENOST_PRAĆENJA:
     razlika_d = ABS(auto.d – ego_auto.d)
     Ako je razlika_d > 2 I razlika_d < 6:
          Auto_s_desne_strane = ISTINA
     Ili ako je razlika_d < -2 I razlika_d > -6:
          Auto_s_lijeve_strane = ISTINA
     Ili ako je razlika_d > -2 I razlika_d < 2:
          Auto_ispred = ISTINA
```

Iz prikazanog pseudo koda može se zaključiti kako se uspoređuju samo d koordinate vozila, budući da su jedino one bitne, jer kazuju u kojoj se traci vozilo nalazi. Udaljenost praćenja je jednaka vrijednosti 8,5 m i označava udaljenost na kojoj će se vozilo razmatrati kao moguća prepreka. Kao što je vidljivo iz pseudo koda, vozila se razmatraju ako su između $2\,m$ i $6\,m$ udaljenosti, što znači da razlika ta dva broja predstavlja širinu ceste, odnosno izvan $6\,m$ udaljenosti više ne pripada pod taj smjer kretanja, ili pripada pod traku iz suprotnog smjera ili van ceste. Potom su ažurirana moguća stanja vozila koja vozilo može zauzeti s obzirom na dane prepreke. Vozilo je uvijek u mogućnosti da se zadrži u traci, budući da može samo smanjiti brzinu da se prilagodi vozilu ispred, ako postoji potreba za tim. Ako nema vozila s lijeve strane onda je dostupno stanje LCL, i isto ako nema vozilo s desne strane dostupno je stanje LCR.

Sa setom mogućih stanja koje vozilo može zauzeti, odnosno manevara koje može izvesti, finalizira se putanja. Taj je postupak podijeljen u tri koraka.

Prvo se za svako od mogućih stanja koje vozilo može zauzeti računaju konačni parametri vozila $[s_{target}, s'_{target}, s''_{target}, d_{target}, d'_{target}, d''_{target}]$. Postavljanjem početnih uvjeta koje se moraju zadovoljiti zbog ograničenja koje simulator postavlja. Lateralno ubrzanje d' i torzija d'' moraju biti jednaki nuli, a longitudinalno ubrzanje s' mora biti takvo da se poveća za minimum povećanja koji je dozvoljen, i da ne prekorači granicu brzine. Torzija u longitudinalnom smjeru mora isto biti jednaka nuli, a udaljenost koju putanja određuje ne smije premašiti duljinu za koju je potrebna veća brzina od predviđene za prijeći u zadanom vremenu. Za određivanje putanje potrebno je znati parametre vozila ispred, što se dobiva od simulatora te se koristi za određivanje manevra vozila, odnosno, ako se vozilo kreće ispred ego vozila i ako je dovoljno blizu odabire se jedno od mogućih manevara za premještanja trake, LCR ili LCL.

Nakon što su izračunati parametri ego vozila za taj manevar, određuje se set koordinata za isti. Cilj je dobiti što normalniju putanju, da nema previše varijabilnosti između brzina koje vozilo postiže, odnosno cilj je dobiti što glađu putanju bez prenaglog ubrzavanja i usporavanja. I zbog toga se koristi funkcija *Jerk Minimizing Trajectory* (engl. *Jerk Minimizing trajectory*, JMT), koja prima početna stanja vozila, konačna stanja vozila i vrijeme u kojem se ta udaljenost mora prijeći. Slijedi primjer proračuna koeficijenata polinoma koji predstavlja JMT prema literaturi [12]. Matematički se definira polinom funkcije vremena koji predstavlja funkciju puta:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + \cdots \tag{3-14}$$

Derivacije tog polinoma predstavljaju redom brzinu, ubrzanje i trzaj prikazanim u izrazu (3-14):

$$\begin{aligned} v(t) = s'(t) &= a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 + \cdots, \\ a(t) = s''(t) &= 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 + \cdots, \\ j(t) = s'''(t) &= 6a_3 + 24a_4 t + 60a_5 t^2 + \cdots, \end{aligned} \tag{3-15}$$

gdje je $s(t)$ funkcija puta, $v(t)$ funkcija brzine, $a(t)$ funkcija ubrzanja i $j(t)$ funkcija trzaja po vremenu. Za proračun se uzima polinom petog stupnja zbog jednostavnosti i brzine proračuna i zbog dobivanja glađe putanje, pa se može zaključiti kako je peti stupanj najoptimalnije rješenje u ovom slučaj budući da se radi s ograničenim resursima. Već su poznati početni uvjeti vozila u vremenu $t = 0$ (s_0, s_0', s_0''), a u prethodnom koraku su određeni (s_T, s_T', s_T'') u trenutku $t = T$. Cilj je odrediti koeficijente polinoma prikazanog u izrazu (3-14), za $T = 0$ možemo izračunati prva tri koeficijenta:

$$\begin{aligned} s(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, \\ s_0 &= s(0) = a_0, \\ s_0' &= s'(0) = a_1, \\ s_0'' &= s''(0) = 2a_2. \end{aligned} \tag{3-16}$$

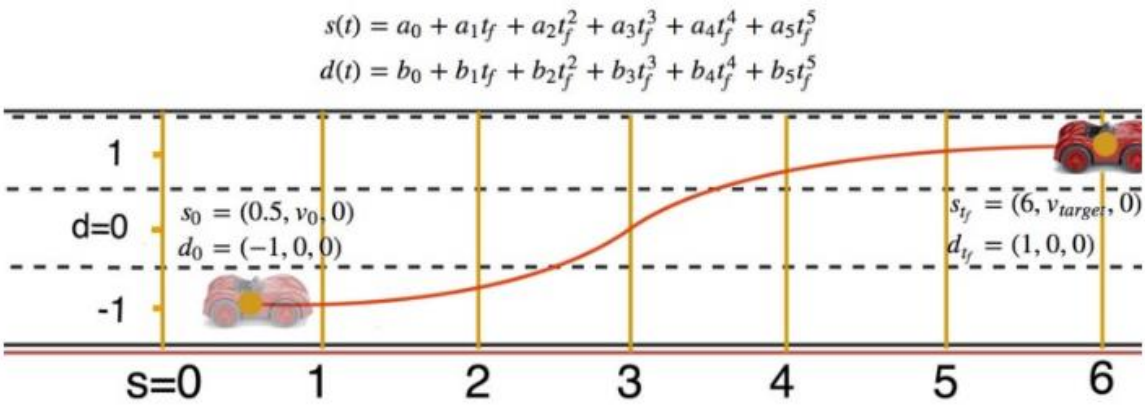
Ostala su još tri koeficijenta za odrediti, iz sustava jednadžbi prikazanog u izrazu (3-17), za konačna stanja vozila:

$$\begin{aligned} s_T &= a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4 + a_5 T^5, \\ s_T' &= a_1 + 2a_2 T + 3a_3 T^2 + 4a_4 T^3 + 5a_5 T^4, \\ s_T'' &= 2a_2 + 6a_3 T + 12a_4 T^2 + 20a_5 T^3. \end{aligned} \tag{3-17}$$

Isto se ponavlja za d koordinate:

$$d_T = b_0 + b_1 T + b_2 T^2 + b_3 T^3 + b_4 T^4 + b_5 T^5. \tag{3-18}$$

Idealno bi bilo da tijekom planiranja putanje nema ubrzanja, kao ni promjene trake, odnosno nekada je potrebno i ograničiti brzinu vozila s brzinom vozila ispred kako ne bi došlo do sudara. Također treba uzeti u obzir da se prilikom manjih brzina, ne bi smjeli raditi manevri promjena trake, odnosno obilaženja prema literaturi [11]. Slika 3.5. jasno ilustrira proces JMT-a.



Sl. 3.5. *Proces određivanja JMT prema [13]*

S dobivenim koeficijentima polinoma, upotpunjuje se putanja vozila.

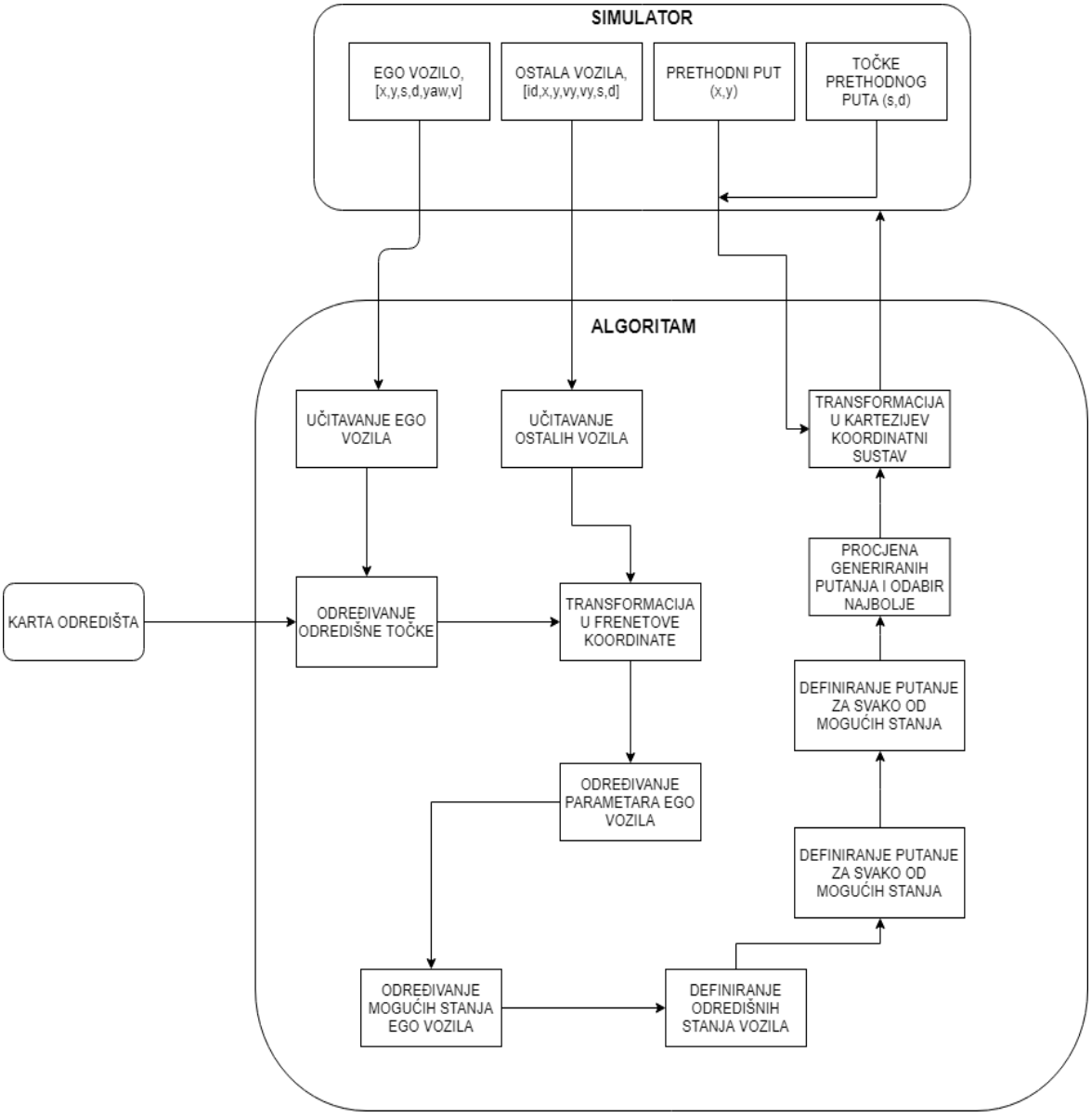
Zadnji korak je određivanje konačne putanje za što su korištene težinske funkcije koje nagrađuju, odnosno kažnjavaju određene postupke i ponašanja vozila. Za svaki od manevara izračunata je sveukupna težina izvođenja. Za dobivenu putanju, za te parametre vozila i za taj manevar računa se „cijena“ izvođenja, tj. skupom težinskih funkcija procjenjuje se najbolja putanja:

- *Collisions cost* – težinska funkcija koja uzima najbliže vozilo ego vozilu, te računa nalazi li se isto u radijusu $1{,}25\ m$ od ego vozila, vraća ili 0 ili 1, koeficijent značaja je 999999,
- *Buffer Cost* – težinska funkcija za određivanje vozila koje se najviše približava ego vozilu kroz definirane putanje ostalih vozila, vraća vrijednost između -1 i 1, koeficijent značaja je 10,
- *In lane buffer cost* – težinska funkcija koja kažnjava ego vozilo ako se previše približava drugim vozilima u traci, vraća vrijednost između -1 i 1, koeficijent značaja je 1000,
- *Efficiency cost* – težinska funkcija koja nagrađuje prosječno veliku brzinu u putanji, budući da se vozilo kreće brzom cestom, vraća ili 0 ili 1, koeficijent značaja je 10000.

Navedene težine množe se s pripadajućim koeficijentima koji označavaju njihovu cijenu. Same vrijednosti nisu bitne, već njihovi odnosi, budući da one označavaju utjecaj svake funkcije zasebno. Najvažnija težinska funkcija je *collision cost* jer ona mjeri mogućnost sudara na određenoj putanju, budući da je sudar najgori mogući scenarij, mora se apsolutno uvijek izbjeći, i

zato ima najveću cijenu. Sve su težine zbrojene te je potom odabrana ona putanja s najmanjom težinom.

Kada je odabrana putanja koja će se koristiti, na početak iste stavlja se par zadnjih točaka koja je simulator predao, budući da se želi postići glatka putanja bez prekida, zato se nastavlja na već dio puta koji je simulator predao algoritmu. Potrebno je transformirati koordinate odabrane putanje u Kartezijev koordinatni sustav, budući da algoritam radi s tim koordinatama. Transformirane točke korištene su za interpolaciju završne odabrane putanje s manevrom, koristeći već poznate vrijednosti putanje u s koordinatama i tek dobivene koordinate x ili y, i dobivaju se dva seta interpoliranih točaka (x, y) koji se predaju simulatoru. I sve se ponavlja opet kada simulator vrati podatke. Rad algoritma prikazan je na slici 3.6.



Sl. 3.6. *Dijagram toka rada algoritma i simulatora*

3.3. Pokretanje rješenja

Kako bi se rješenje uspješno pokrenulo potrebno je imati instalirane prethodno navedene programe. Programsko rješenje se nalazi u P.3.2., te je prvo potrebno izgraditi biblioteku sljedećim setom naredbi:

```
cd uWebSockets
cmake .
make
```

Nakon uspješne izgradnje biblioteke, sljedećim setom naredbi mora se izgraditi sam algoritam:

```
cd ..
mkdir build && cd build
cmake .. && make
```

Nakon toga slijedi pokretanje simulatora iz priloga 3.1., preporučano na najbrži način rada i najmanje rezolucije inače će se samo rad simulatora jako usporiti zbog prevelikih zahtjeva. Nakon što se slijede upute u simulatoru, na slici 3.7. treba odabrati *select* gumb za pokretanje simulatora.

Sl. 3.7. *Izbornik Udacity simulatora, P.1.*

Nakon toga se simulator pokreće i ulazi u simulaciju, i onda se pokreće dano rješenje algoritma i može vidjeti njegov rad u simulatoru, sljedećom naredbom:

```
./path_planning
```

3.4. Implementacija rješenja na ADAS ploču

Razvoj ADAS ploča potaknut je napretkom ADAS sustava i mogućnošću realne aplikacije istih u komercijalnim vozilima. Svrha ADAS sustava je automatizirati vožnju i time olakšati i učiniti vožnju daleko sigurniju čovjeku. Svi ADAS algoritmi se temelju na obradi podataka dobivenim od senzora, i njihovom daljnjom obradom algoritama, kao npr. su detekcija i klasifikacija objekata, praćenje objekata i procjena brzine. Razlika između potpune automatizirane vožnje i korištenje ADAS sustava leži u ljudskoj intervenciji prilikom vožnje, i time nastaje potreba za klasificiranjem stupnja autonomnosti vozila na šest kategorija prema literaturi [14]:

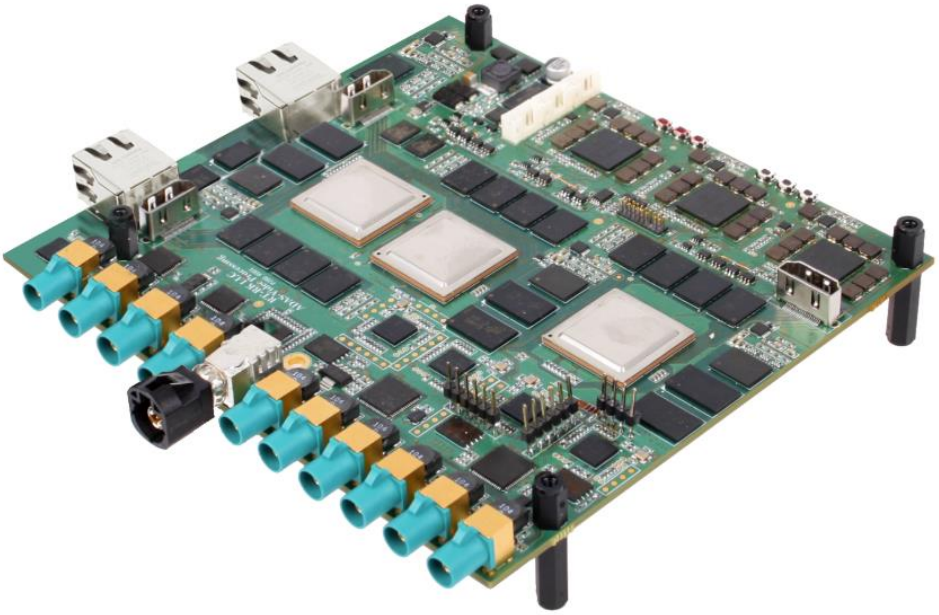
- nulta kategorija ili nulti stupanj predstavlja vozilo bez ikakvih dodatnih kontrola ili senzora,
- u prvom stupnju autonomnosti vozač ima potpunu kontrolu nad vozilom, i zadužen je za praćenje prometa, kočenje, ubrzavanje i upravljanje, vozilo jedino može prilagoditi

određene vozačeve kontrole, npr. ukoliko vozač naglo koči vozilo može pojačati intenzitet kočenja kako ne bi došlo do nesreće,

- u drugom stupnju vozilo pomaže vozaču prilikom upravljanja i ubrzavanja, ali vozač mora biti spreman u svakom trenutku preuzeti kontrolu nad vozilom, većina vozilo koja se danas proizvode imaju ovaj stupanj autonomnosti, npr. kada se vozilo kreće autocestom može se uključiti kontrola brzine ili kontrola upravljanja, kako bi vozilo zadržalo svoju trenutnu brzinu i smjer,
- najveći skok u autonomnosti je u trećem stupnju, kada vozilo je zaduženo za potpuno praćenje okoline korištenjem dostupnih senzora, vozač mora intervenirati u slučaju grešaka, vozilo je sposobno samostalno upravljati na cestama gdje je ograničenje $60\ km/h$,
- četvrti stupanj predstavlja potpuno automatizirano vozilo, s iznimkom da vozač mora uključiti takav sustav i da vozilo nije u potpunosti sposobno upravljati u dinamičnim situacijama kao npr. u slučaju prometnih gužvi, ili uključivanja na autocestu,
- zadnji stupanj predstavlja potpuno automatizirano vozilo bez ikakve vozačeve kontrole.

Postoje brojne ADAS ploče trenutno na tržištu, osim *Alpha* zanimljiva je i NVIDIA-ina *NVIDIA DRIVE AGX* koja dolazi s njihovim razvojnim okruženjem i simulatorom u kojem se mogu provjeriti i evaluirati rješenja postavljena na ploču.

Alpha ploču je prema literaturi [15], napravio Institut RT-RK Novi Sad. *Alpha* ploča je dana na slici 3.8. Ploča podržava osnovne i napredne sustave upozorenja, aktivnu kontrolu sustava i poluautomatske operacije, također je dana podrška za razvijanje algoritama za detekciju i klasifikaciju pomoću sustava kamera, te korištenje određenog skupa predefiniranih biblioteka za iste potrebe.



Sl. 3.8. *Alpha ploča, [13]*

Alpha ploča se sastoji od sljedećih komponenti:

- tri čipa TI-a (engl. *Texas Instruments*) serije TDA2x optimizirane za ADAS algoritme, s sljedećim jezgrama;
    - dvije ARM (engl. *Advanced RISC Machine*, ARM) Cortex A15,
    - dvije ARM Cortex M4,
    - dvije C66x za digitalnu obradu signala (engl. *Digital Signal Processing*, DSP),
    - četiri EVE (engl. *Embedded vision Engine*) jezgre,
- balansiranje opterećenja za algoritme za obradu video zapisa između SoC-ova (engl. *System on Chip*, SoC) je dostupan putem sabirnice (engl. *Peripheral Component Interconnect*, PCI-e),
- svaki SoC ima; HDMI (engl. *High-Definition Multimedia Interface*), DCAN (engl. *Controller Area Network*, DCAN), Ethernet, JTAG, UART (engl. *Universal Asynchronous Receiver-transmitter*, UART) sučelja i spremnik za *microSD* (engl. *Secure Digital*, SD) karticu,
- sučelja za deset kamera,
- 1,5 GB memorije za svaki od SoC-ova,
- podrške za AVB (engl. *Audio Video Bridging*, AVB) *Ethernet*.

Ploča podržava tri načina rada putem *microSD* kartice, *debug* način rada i putem *flasha*.

Izrada algoritama za *Alpha* ploču vrši se u *VisonSDK* razvojnoj okolini, koja se temelji na *Links and Chains* arhitekturi, u kojoj *link* predstavlja jednu logičku i funkcionalnu cjelinu, npr. može biti zadužena ili za slanje video podataka ili za primanje, te su te cjeline međusobno povezane kroz API *VisionSDK*-a, i time se apstrahira sklopovlje ploče, i pojednostavljuje kodiranje na različitim procesorima.

Implementacija predstavljenog rješenja na *Alpha* ploču nije uspjela. Zamišljeno je bilo kako će se sam algoritam pokretati s *Alpha* ploče, a simulator na računalu. Kako bi se to izvelo nužno ostvariti vezu između tih dviju komponenti, te se na *Alpha* ploči pokušala osposobiti veza putem *Ethernet* kabla, koristeći već dostupne alate na *Alpha* ploči za primanje i slanje podataka, ali samo primanje tekstualnih podataka na *Alpha* ploči nije ostvareno. Ostvarivanje veze nije ostvareno prvenstveno zato što *Alpha* ploča ne podržava izmjenu tekstualnih podataka. Nastojano je uhvatiti poslane podatke, ali nije točno moguće odrediti u kojem dijelu *VisionSDK*-a se točno primaju podaci. Tijekom rada izmijenjen je kod *NullSrca linka*, kako bi se odredilo u kojem se dijelu primaju podaci, ali taj dio nije otkriven. Prilikom ostvarivanja veze korišteni su sljedeći alati:

- *Wireshark* – za praćenja cjelokupnog prometa na mreži te potvrda jesu li podaci uistinu poslani,
- *Tornado* – kreirane su razne serverske i klijentske *Python* skripte koristeći Tornado biblioteku u *Pythonu*,
- *Socket* – također su kreirane klijentske skripte u *Pythonu* koristeći *socket* biblioteku,
- Mrežni alati (engl. *Network tools*) – set alata koji se nalazi uz *VisionSKD*,
- *Curl*, *libtins* – alati za formiranje mrežnog prometa.

Također jest izrađena i vlastita verzija mrežnog alata po uzoru na već dane, ali bezuspješno. Na *Alpha* ploči se pokušalo iskoristiti već postojeće rješenje za video podatke link *NullSrc*, koji služi za primanje video podataka. Na već njegove gotovo funkcije su napisane dodatne koji bi podržavale meta podatke, ali također bezuspješno.

4. EVALUACIJA RJEŠENJA

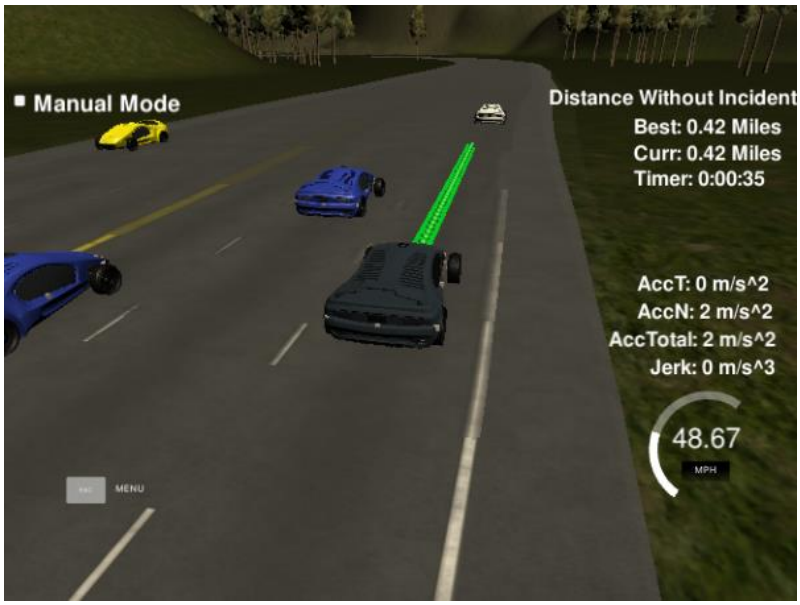
Evaluacija rješenja podijeljena je u četiri testa, tijekom kojih se zapažao rad vozila i pratila statistika kretanja vozila na stazi koja je priložena u P.4.1. Tijekom testova se pratila najduža prijeđena udaljenost vozila u kojoj nije došlo do pogreške, pratile su se sve pogreške koje i simulator prati, a to su prekoračenje brzine, ubrzanja i torzije, zadržavanje na traci i sudar. Testovi se razlikuju po svojoj duljini, a sam simulator prilikom svakog testa postavlja drugačije uvjete na cesti, odnosno simulira određen broj vozila koja voze različitom brzinom te se drugačije ponašaju.

Prvi test je trajao 5 minuta, u tablici 4.1. se mogu vidjeti rezultati zapažanja tijekom istog. U testu nije došlo do puno pogrešaka, jedino je vozilo prekoračilo brzinu, ubrzanje i torziju, što će se primijetiti i u svim naknadnim testovima. Sam simulator razvijen od *Udacitya* je napravljen u *Unityu* i izuzetno je zahtjevan prilikom rada, odnosno njegov rad jest usporen i uvelike ovisi o samoj snazi računala. Najveća prijeđena udaljenost je 3 $miles$, što je 4,82803 km.

Tab. 4.1. *Pregled pogrešaka u prvom testu*

Vrsta pogreške	Broj pogrešaka
Brzina	1
Ubrzanje	2
Torzija	2
Zadržavanje u traci	0
Sudar	0

Na slici 4.1. se može vidjeti vozilo u kretanju desnom trakom na brzoj cesti, u brzini 48,67 mph što je jednako 78,326772 km/h. Ono što uzrokuje ovakvu pogrešku su ograničeni resursi na kojima se pokreće simulator, konkretno simulator je izuzetno grafički zahtjevan budući da mora obraditi velik broj senzorskih podataka za vozila. Uvjeti koji su uzrokovali pojavu greške brzine, ubrzanja i torzije su odmah na početku algoritma, budući da se tijekom početne iteracije nema podataka od simulatora od prijašnje putanje i time se povećava proračun, odnosno ne može se nastaviti na prethodnu putanju.

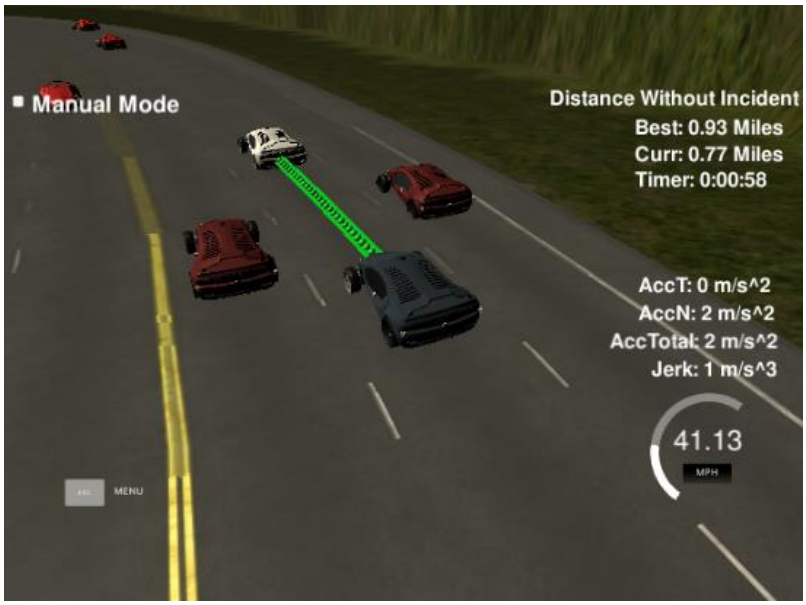


Sl. 4.1. Prikaz vozila u prvom testiranju kako se kreće desnom trakom

Drugi test je trajao 10 minuta i tu se može primijetiti veći broj pogrešaka nego u prvom, uzevši u obzir da je duže trajao. Najduži put bez pogrešaka iznosio je $2{,}22\,miles$ ili $3.54056\,km$. Vozilo je u ovom testu počinilo dvije kritične greške sudara s drugim vozilima, oba sudara su se dogodila prilikom pretjecanja. U prvom slučaju je ego vozilo mijenjalo traku iz desnu u srednju, da pretekne sporije vozilo ispred sebe, ali je vozilo u srednjoj traci naglo usporilo i ušlo u putanju ego vozila, koje nije stiglo ispraviti vlastitu putanju. Zbog istog je došlo do pogreške u ubrzanju i brzini, a u drugom slučaju je isti scenarij samo što se ego vozilo sudarilo s vozilom iz lijeve trake koje je mijenjalo traku i prebacivalo se u srednju u istom trenutku kada i ego vozilo. Do ostalih pogrešaka je došlo kada je rad računala bio usporen što se očitovalo kratkotrajnim zastojima u grafičkom sučelju računala. U tablici 4.2 su prikazane sve greške, a slika 4.2. prikazuje scenarij nakon sudara vozila, vidljivo je kako je usporilo.

Tab. 4.2. *Pregled pogrešaka u drugom testu*

Vrsta pogreške	Broj pogrešaka
Brzina	1
Ubrzanje	4
Torzija	4
Zadržavanje u traci	1
Sudar	2



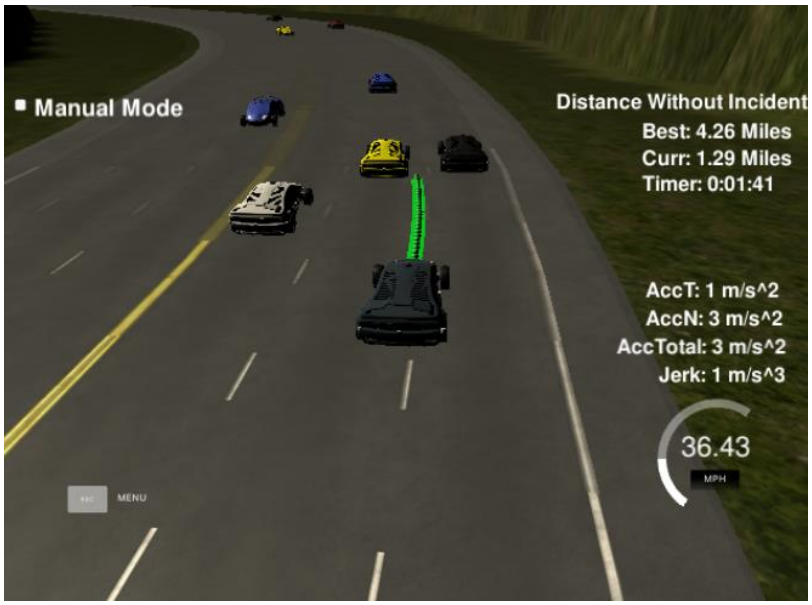
Sl. 4.2. *Prikaz vozila u drugom testiranju kako je ograđeno drugim vozilima i ne može mijenjati svoje stanje, tj. prisiljeno je voziti sporije*

Tijekom evaluacije je primijećeno kako se greške najčešće događaju kada je puno okolnih vozila koje treba analizirati zajedno s njihovim putanjama, i time se sam proračun kao i pripadno vrijeme računanja povećava.

U trećem testiranju, koje je trajalo 15 minuta, dogodio se manji broj pogrešaka nego u prethodnom testiranju, jedan sudar zbog vrlo sličnog scenarija kao i u prethodnom testiranju, a i simulator je opet kratkotrajno usporio što je uzrokovalo greške kod brzine i ubrzanja prikazanih u tablici 4.3. Na slici 4.3 se može vidjeti problem prilikom donošenja odluke vozila, zato se duže zadržava na razdjelnoj crti trake što izaziva pogrešku. Vozilo se kretalo najduže 4,26 $miles$ bez pogreške, što je 6,855805 km.

Tab. 4.3. *Pregled pogrešaka u trećem testu*

Vrsta pogreške	Broj pogrešaka
Brzina	1
Ubrzanje	3
Torzija	3
Zadržavanje u traci	3
Sudar	1



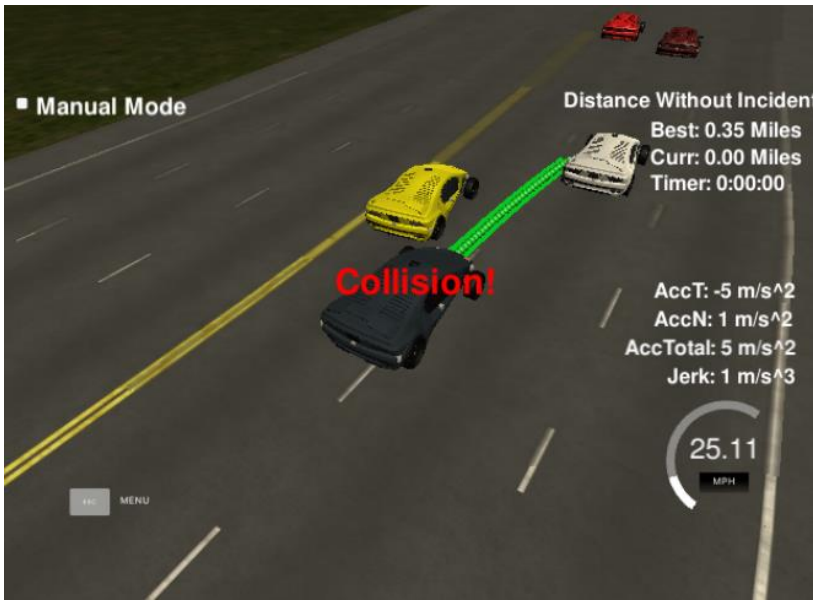
Sl. 4.3. *Prikaz vozila kada vozi između traka*

U trećem testiranju dogodile su se čak tri pogreške tijekom premještanja u drugu traku. Scenarij u kojem se to događa je kada vozilo može birati između više putanja koje uključuju manevar premještanja trake, ali se u susjednim iteracijama obrade podataka dobivaju putanje za premještanje u različite trake. I tako se generirane točke od prethodnog puta koje su vodile u jednu traku, spajaju s točkama generiranim u trenutnoj iteraciji koje vode u drugu traku, što izaziva zadržavanje na crti

Zadnje testiranje je trajalo 20 minuta, te je postignuta najveća prijeđena udaljenost vozila bez pogrešaka, što je i očekivano budući da je vožnja najduže trajala. U tablici 4.4. je vidljiv pregled svih pogrešaka koje su se dogodile tijekom zadnje vožnje. Na slici 4.4 se može primijetiti kako je došlo do sudara između vozila, u tom slučaju nije dobro izračunata putanja prilikom izvođenja manevra pretjecanja i žuto vozilo se okrznulo tijekom izvođenja.

Tab. 4.4. *Pregled pogrešaka u četvrtom testu*

Vrsta pogreške	Broj pogrešaka
Brzina	2
Ubrzanje	1
Torzija	1
Zadržavanje u traci	2
Sudar	2



Sl. 4.4. *Prikaz vozila prilikom sudara*

U P.4.1. se nalaze sve slike tijekom testiranja, i podaci svih testiranja, i posljednjeg u trajanju od 4 sata, tijekom kojeg su sakupljeni podaci o stanju vozila tijekom tog vremena. U četiri sata vozilo je najduže vozilo $14{,}10\ miles$ bez incidenata, što je $22{,}69175\ km$.

Tijekom evaluacije algoritma uočeno je kako greške nisu proporcionalne s duljinom testiranja, odnosno neke greške su se češće ponovile zbog uvjeta na cesti, tj. broja vozila i njihovih pripadnih brzina, te ih se moglo izbjeći poboljšavanjem FSM dijagrama prikazanog na slici 3.4. tako da se između stanja zadržavanja i izmjene trake, definira stanje pripreme za izmjenu trake, definiranjem raspona cijene takvog manevra, što uzrokuje nužnost poboljšanja težinskih funkcija i uvođenje novih, kao npr. funkcija težine za izmjenu traka. Budući da se treba nagraditi putanja zadržavanja trake, jer zadržavanjem trake se smanjuje vjerojatnost da dođe do nesreće. Korištenjem takvog FSM-a bi se smanjile kritične greške sudara.

Probleme prekoračenja brzine, ubrzanja i torzije moguće je ispraviti korištenjem polinom većeg stupnja za određivanje JMT-a, ali bi se time smanjila brzina izvođenja algoritma. Isti su problemi uzorkovanim nepreciznom transformaciju koordinata između Kartezijevog i Frenetovog sustava, da su podaci o obliku staze bili poznati koeficijenti za transformaciju su se mogli poboljšati.

5. ZAKLJUČAK

U sklopu diplomskog rada razvijeno je rješenje za planiranje putanje autonomnog vozila korištenjem već gotovog simulatora. Algoritam je temeljen na već postojećim rješenjima te je pojednostavljen s obzirom na korišteni simulator. Implementacija algoritma na ADAS ploču pokazala je poteškoće korištenja razvojnog okruženja za ploču. Evaluacija algoritma pokazuje kako ne zadovoljava uvjete stvarne uporabe, ne samo zbog čestih grešaka koje su se dogodile tijekom testiranja već i zbog činjenice da simulator simulira samo uvjete brze ceste, bez simulacije raznih prometnih znakova, svjetlećih znakova i ostalih uvjeta na cesti. Rad je temeljen na brojnim drugim primjerima koji su pronađeni kao predložena rješenja *Udacityevog* zadatka za izradu algoritma za planiranje putanje autonomnog vozila. Prilikom izrade algoritma korištene su već postojeće matematičke spoznaje, te su uzete u obzir i težine njihove implementacije, i zbog toga je odabran predloženi pristup rješavanju problema. Algoritam koristi dosta prethodno procesiranih podataka, odnosno za svoj rad se oslanja na rad ostalih algoritma za ADAS i time ne može samostalno funkcionirati.

Postoje brojni koraci koji su se mogli unaprijediti kako bi algoritam bolje funkcionirao, svakako prije svega potrebno je koristiti bolji simulator, kojem se mogu mijenjati uvjeti na cesti kao i sama cesta, i time bi i testovi postali pouzdaniji, također bi se i korištenjem bolje računala promijenili određeni loši aspekti rada. Problem stvaraju i već dane metode za pretvorbe koordinata od *Udacitya,* te one nisu precizne, ukoliko bi bili dani detaljniji podaci o stazi mogli su se poboljšati koeficijenti koji se koristi prilikom transformacije. A za sam FSM tijekom biranja mogućih stanja može se uvesti algoritam temeljen na strojnom učenju koji bi preciznije i bolje odabirao moguće manevre vozila.

LITERATURA

[1] H. Sun, W. Deng, S. Wang, Y. Zhang, Trajectory Planning for Vehicle Autonomous Driving with Uncertainties, IEEE, izdano: 2014.

[2] D. Fassbender, B. C. Heinrich, Hans-Joachim Wuensche, Motion Planning for Autonomous Vehicles in Highly Constrained Urban Environments, IEEE, izdano: 2016.

[3] S. Zhang, Weiwen Deng*, Q. Zhao, H. Sun, B. Litkouhi, Dynamic Trajectory Planning for Vehicle Autonomous Driving, IEEE, izdano: 2013.

[4] G. Gonzalez, *Udacity*, *Github*, https://github.com/udacity/CarND-Path-Planning-Project, izdano: 19.07.2018., datum posjete: 28.8.2018.

[5] *Udacity*, Službene stranica *Udacitya*, https://eu.udacity.com/, datum posjete: 28.8.2018.

[6] FF Team, Faraday&Future Inc., What Is Lidar & How Is It Making Self-Driving Cars Safer?, https://www.ff.com/us/futuresight/what-is-lidar/, izdano: 9.10.2016., izdano: 19.7.2018., datum posjete: 29.8.2018.

[7] T. Klage, T. Klage, Cubic Spline Interpolation in C++, http://kluge.in-chemnitz.de/opensource/spline/, izdano: 2014., datum posjete: 3.9.2018.

[8] dr. R. Buchanan, Millersville University, Department of Mathematics, Cubic Spline Interpolation, Numerical Analysis, http://banach.millersville.edu/~bob/math375/CubicSpline/main.pdf, izdano: 2013, datum posjete: 4.9.2018.

[9] A. Kastsiukavets, Medium, Highway Path Planning, https://medium.com/@kastsiukavets.alena/highway-path-planning-696215cbf062, izdano: 2017, datum posjete: 4.9.2018.

[10] A. Mate, The Frenet-Serret Sormulas, Brooklyn Collage Of The City University Of New York, izdano: 19.1.2017.

[11] M. Werling, J. Ziegler, S. Kammel, S. Thrun, Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame, IEEE, izdamo: 2010.

[12] V. Kumar, Robotics: Aerial Robotics, Coursera, University of Pennsylvania, izdano: 2016.

[13] J. Hui, Medium, Self-driving car: Path planning to maneuver the traffic, https://medium.com/@jonathan_hui/self-driving-car-path-planning-to-maneuver-the-traffic-ac63f5a620e2, izdano: 2018. datum posjete: 9.9.2018.

[14] SAE, Automated Driving – Levels of Driving Automation are Defined in New SAE International Standard J3016, izdano: 2014.

[15] RT-RK Institut, http://www.rt-rk.com/, datum posjete: 9.9.2018.

PRILOZI

Prilog P.3.1. - Simulator za vožnju, *Udacity*, nalazi se u digitalnom formatu priloženom u mapi na CD-u pod nazivom term3_sim.

Prilog P.3.2. – Programski kod za izvršavanje algoritma, nalazi se u digitalnom formatu priloženom u mapi na CD-u pod nazivom planiranjePutanje.

Prilog P.4.1. – Zapis parametara vozila tijekom vožnje za svaki od testova, priložen u digitalnom formatu u mapi evaluacija.

SAŽETAK

Tema rada je istraživanje i razvoj algoritma za planiranje putanje autonomnog vozila. Predstavljena su neka dosadašnja rješenja te je definirano razvojno okruženje koje je korišteno za razvoj vlastitog rješenja. Za razvoj algoritma korišten je *Udacityev* simulator koji simulira uvjete na brzoj cesti s tri trake. Algoritam i simulator dvosmjernom komunikacijom razmjenjuju podatke, na način da simulator predaje senzorske podatke algoritmu koji zauzvrat vraća samo koordinate vozila koje mora posjetiti u narednom vremenu. Algoritam prvo učitava dane podatke, zatim vrši transformaciju u Frenetov koordinatni sustav radi lakšeg i preciznijeg proračuna. S Frenetovim koordinatama se računaju parametri ego vozila i moguća stanja koja može zauzeti na cesti. Potom se definiraju putanje za svako od stanja i ocjenjuju s obzirom na određene kriterije i pripadne težine. Nakon definiranje konačne putanje vozila, ona se mora transformirati natrag u Kartezijev koordinatni sustav, kako bi se predala simulatoru.

Evaluacijom rješenja zaključeno je kako algoritam radi kako je i zamišljeno u uvjetima simulatora, ali to nije zadovoljavajuće za stvarne uvjete na cesti koji su nepredvidivi. Simulator nije realan prikaz uvjeta na cesti budući da simulira samo brzu cestu s ostalim vozilima, bez prometnih znakova ili promjene ograničenja. Što čini i same testove nerealnim prikazom rada algoritma.

Implementacija algoritma na *Alpha* ploču nije osposobljena zato što nije ostvarena dvosmjerna komunikacija podataka između simulatora i ploče.

Ključne riječi: ADAS, planiranje putanje, putanja, vozilo, autonoman, mapa

ABSTRACT

AUTONOMOUS VEHICLE TRAJECTORY PLANNING USING GLOBAL MAP

The theme of the paper is the research and development of an algorithm for planning the path of an autonomous vehicle. The most important solutions so far are presented, and a development environment is defined in which the algorithm is made. For the development of the algorithm Udacity's simulator is chosen as it simulates the conditions on a fast three-track road. The algorithm and simulator communicate with the two-way communication; the simulator handles the sensor data to an algorithm that in return returns only the coordinates that the vehicle will pass in its trajectory. The algorithm loads the given data, and then transforms it into a Frenets coordinate system for easier and more precise calculation. Frenets coordinates calculate the parameters of the ego vehicle and the possible manners that can be taken on the road. Then, the paths for each of the states are defined and evaluated with respect to certain criteria and multiplied with respective weights. After defining the final path of the vehicle, it must be transformed back into the Cartesian coordinate system so that the simulator can handle the data.

By evaluating the solution, it was concluded that the algorithm works as intended by the simulator, but this is not satisfactory for realistic road conditions that are unpredictable. The simulator is not a realistic view of road conditions since it only simulates a fast road with other vehicles, with no traffic signs or changes in constraints what makes the tests themselves unrealistic algorithm performance.

The algorithm implementation on the *Alpha* board failed despite numerous attempts to enable communication between the simulator located on computer and the *Alpha* board via Ethernet.

Keywords: ADAS, autonomous, car, path planning, trajectory, map

ŽIVOTOPIS

Jure Bajić rođen je 23.8.1994. u Osijeku. Osnovnu školu je pohađao u Višnjevcu. Za to vrijeme aktivno je trenirao ples i vaterpolo. 2009. godine upisuje Prirodoslovno-matematičku gimnaziju, nakon čega upisuje Elektrotehnički fakultet u Osijeku, smjera elektrotehnike. Za vrijeme ljeta 2015. godine, odrađuje praksu u *Betaweru* kao web programer. Na *Microsoftovom* natjecanju, *Software StartUp Academy* 2016. odnosi pobjedu s timom GymTeam i aplikacijom *Palester* te osvaja nastup na završnici u Poreču u sklopu *Microsoftove* konferencije *WinDays*. Iste godine počinje raditi u *UHP Digital-u* kao *backend* i android programer. Upisuje diplomski studij komunikacije i informatike smjera mrežne tehnologije, a 2017. godine postaje praktikant RT-RK Instituta Osijek.

Potpis: _____