SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4524

SIMULACIJA AUTONOMNOG PARKIRANJA VOZILA

Hrvoje Nuić

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4524

SIMULACIJA AUTONOMNOG PARKIRANJA VOZILA

Hrvoje Nuić

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 16. ožujka 2016.

ZAVRŠNI ZADATAK br. 4524

Pristupnik: **Hrvoje Nuić (0036470074)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Simulacija autonomnog parkiranja vozila**

Opis zadatka:

Proučiti metode umjetne inteligencije prikladne za upravljanje virtualnim vozilom. Posebice proučiti i razraditi primjenu neuronskih mreža. Razraditi povezivanje upravljanja vozilom pomoću neuronskih mreža s virtualnom scenom pri postupku parkiranja vozila u svrhu ostvarivanja autonomnog parkiranja. Ostvariti implementaciju virtualnog okruženja koje omogućuje prikaz simulacije parkiranja vozila. Načiniti ocjenu i usporedbu ostvarenih rezultata.
Izraditi odgovarajući programski proizvod. Koristiti programsko okruženje Unity. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:

Prof. dr. sc. Željka Mihajlović

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Sadržaj

Uvod

Čovjek koji koristi automobil kao glavno prijevozno sredstvo dnevno provede prosječno sat vremena vozeći se u njemu[1]. Kako nije lako drastično smanjiti vrijeme provedeno na putu, javlja se potreba da to vrijeme pokušamo iskoristiti na bolji način, a jedno od mogućih rješenja je korištenje autonomnog vozila.

Proizvođači automobila su prepoznali taj zahtjev i danas možemo primijetiti rastući broj automobila s mogućnošću djelomično autonomne vožnje. Prostora za poboljšanje algoritama autonomne vožnje ima, jer i dalje postoje ograničenja poput nemogućnosti sigurne vožnje kroz kišu i snijeg te potreba za mapiranjem znakova i signalizacije na cestama[2]. Dosadašnjim trendom se prognozira da će do 2020. godine na cestama biti 10 milijuna vozila s mogućnošću autonomne vožnje[3], a do 2040. godine će ih biti čak 75%[4].

Jedna od komponenti potrebnih za ostvarenje potpuno autonomnog vozila je mogućnost pronalaska mjesta za parkiranje i izvedba manevra parkiranja.

Središte ovog rada je proučiti načine na koje umjetna inteligencija može upravljati vozilima te potom implementirati simulaciju vozila na stazi u Unity okruženju kojim upravlja neuronska mreža.

---

[1] http://nhts.ornl.gov/2009/pub/stt.pdf
[2] https://www.technologyreview.com/s/530276/hidden-obstacles-for-googles-self-driving-cars/
[3]http://www.businessinsider.com/report-10-million-self-driving-cars-will-be-on-the-road-by-2020-2015-05
[4] http://www.ieee.org/about/news/2012/5september_2_2012.html

1. Zašto koristiti autonomna vozila?

Naveo sam neke pozitivne i negativne strane korištenja autonomnih vozila.

Do vremenski učinkovitijeg i jeftinijeg parkinga dolazi kada vozač izađe na točno željenom mjestu, a vozilo potom bez pomoći vozača pronađe parking i izvede manevar parkiranja. Vozač, kad ponovno treba prijevoz, vozilu može poslati zahtjev da dođe do njega.

Povećana sigurnost i smanjen stres uzrokovan vožnjom. 94% svih prometnih nesreća nastaje ljudskom greškom[5], a korištenjem autonomnog vozila taj bi se faktor drastično smanjio, ako ne i uklonio.

Povećanje učinkovitosti goriva i smanjenje zagađenja okoliša. U odnosu na čovjeka, autonomno vozilo pomoću tablica može učinkovitije upotrebljavati gorivo[6], a implementacijom komunikacije između autonomnih vozila mogu se izbjeći nagla kočenja.

Automobil je prosječno u upotrebi samo 5% vremena[7], a ta se brojka može poboljšati dijeljenjem istog vozila, što je olakšano autonomnim parkiranjem.

Početna cijena automobila veća je zbog dodatne opreme potrebne za ispravan rad autonomnog vozila. Poslodavcima će biti isplativije koristiti autonomna vozila, zbog čega može doći do smanjenja broja zaposlenih u sektoru prijevoza robe i osoba.

---

[5] http://www-nrd.nhtsa.dot.gov/pubs/812115.pdf

[6] http://en.volkswagen.com/content/medialib/vwd4/de/Volkswagen/Nachhaltigkeit/service/download/spritspartipps/effizient_unterwegsengl/_jcr_content/renditions/rendition.file/spritspartipps_par_0008_file.pdf

[7] http://www.reinventingparking.org/2013/02/cars-are-parked-95-of-time-lets-check.html

2. Veza umjetne inteligencije i autonomnog vozila

Umjetna inteligencija je odavno široko područje te postoje mnoge metode i algoritmi kojima se mogu rješavati problemi iz svakodnevnog života. Kako postoje mnoge tehnike, važno je znati izabrati bolji način rješavanja problema. Vozilo koje želi samostalno upravljati mora moći u stvarnom vremenu tumačiti podatke iz svog okoliša i na temelju njih činiti akcije kojima dolazi bliže zadanom cilju. Budući da povezuje različita područja, izrada autonomnog vozila je složena.

Neki od primjera primjene umjetne inteligencije kod autonomnog vozila su detekcija znakova uz rub ceste, određivanje najkraćeg puta do odredišta temeljem poznatih puteva i stanja na cestama, detektiranje cestovnih traka, detekcija te izbjegavanje zapreka.

Slika 1. Primjeri korištenja umjetne inteligencije

3. Korišteni alati i tehnologije

3.1. Unity 3D

Unity omogućava jednostavnu i učinkovitu izradu igara, a glavni naglasak je na portabilnosti između sustava. Iako je praktičan za učinkovitu izradu igara, veći proizvođači igara ga ne koriste jer ima lošije performanse od programa pisanih točno za tu namjenu. Zato se koristi primarno kod dokazivanja koncepata ili manjih proizvođača igara[4].

Jedna od glavnih prednosti zbog koje se u Unity-u može brže izraditi programski proizvod je velika javno dostupna baza modela, skripti i animacija koje se vrlo lako mogu uklopiti u projekt. Iz te baze podataka sam koristio model i skriptu koja pokreće automobil te predložak za izradu cesta. Unošenjem sljedećih pojmova u tražilicu od „Asset Store“, mogu se pronaći ti modeli i skripte:

- Standard Assets objavljeno od Unity Technologies[5]
- Simple Modular Street Kit objavljeno od Jacek Jankowski[6]

3.2. Blender

Blender je besplatni program otvorenog koda s kojim se može modelirati, animirati, proizvoditi teksture i slično. Njegove mogućnosti suna razini programa za čije se licence mora odvojiti i preko 1000 kuna.

Kako predložak cesta koje sam preuzeo nije točno odgovarao mojim potrebama, koristio sam Blender kako bi uredio te modele po svojoj želji[7].

4. Funkcionalni zahtjevi

Korisnik koji koristi programski proizvod može:

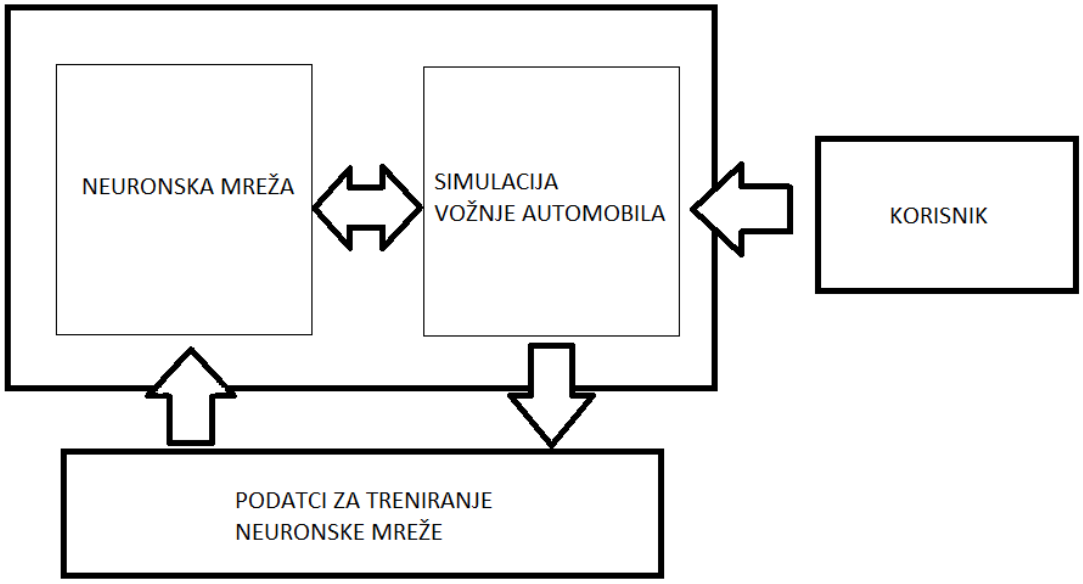
- vlastoručno voziti automobil po stazi,
- podatke o senzorima daljine, brzine automobila i kut kotača na autu spremati u datoteku
- učitati u program podatke koji su spremljeni u datoteku i metodom širenja unatrag (*engl. Backpropagation*) utrenirati neuronsku mrežu proizvoljne veličine
- mrežu nakon treniranja spremiti u datoteku
- iz datoteke učitati utreniranu mrežu proizvoljne veličine
- može pustiti utreniranu neuronsku mrežu da vozi automobil u simulaciji.
- Automobil nije vezan uz stazu po kojoj se vozi, što znači da se staza može proizvoljno prepraviti te na njoj utrenirati neuronska mreža koja će jednako dobro voziti automobil.

5. Nefunkcionalni zahtjevi:

- Implementacija neuronske mreže je napravljena kako bi se omogućila pokretnost programskog koda, što znači da se neuronska mreža može utrenirati nad podatcima koji nisu nužno vezani uz simulaciju.
- Klasa koja upravlja ulaznim i izlaznim podatcima može se lako nadograditi kako bi primao podatke iz drugih izvora kao na primjer s Internet stranice ili serijskih ulaza na računalu.

6. Arhitektura i dizajn sustava

6.1. Skica sustava



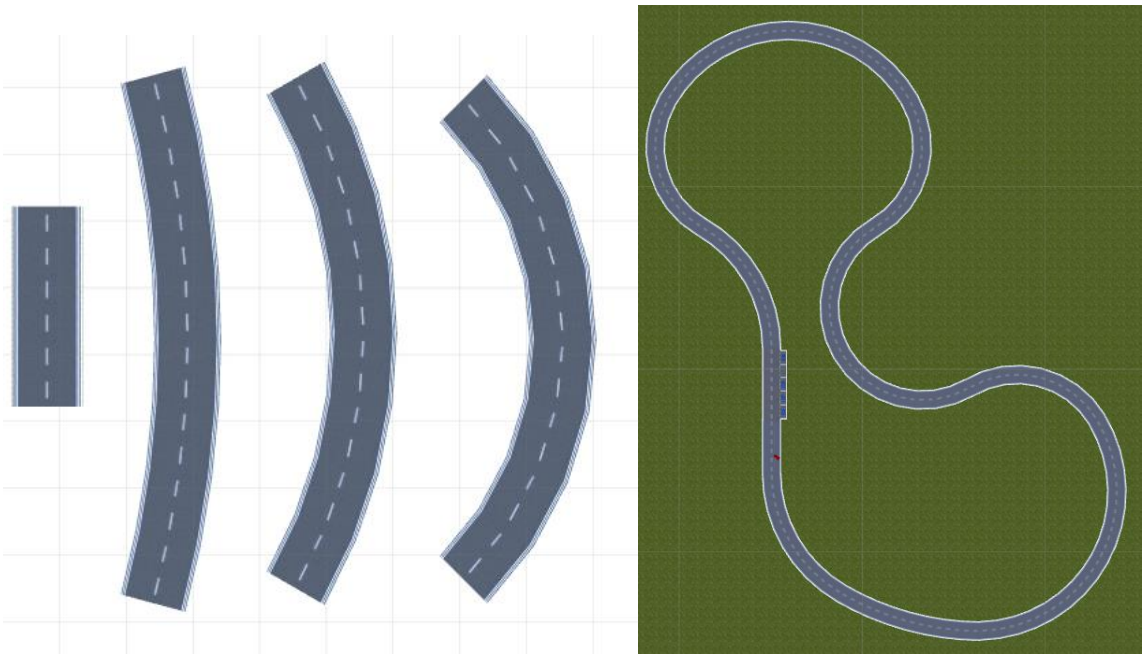
Slika 2. Skica sustava i komponenata

Skica prikazuje koji su odnosi komponenata sustava prema vanjskim aktorima. Korisnik preko standardnog ulaza, koristeći tipkovnicu, igraću palicu ili volan, upravlja modelom vozila u simulaciji. Za to vrijeme aplikacija može zapisivati na koji način je korisnik upravljao vozilom. Te podatke kasnije koristi neuronska mreža prilikom treniranja te zatim, nakon što prođe kroz sve podatke, može upravljati vozilom.

6.2. Struktura scene i simulacije

6.2.1. Staza

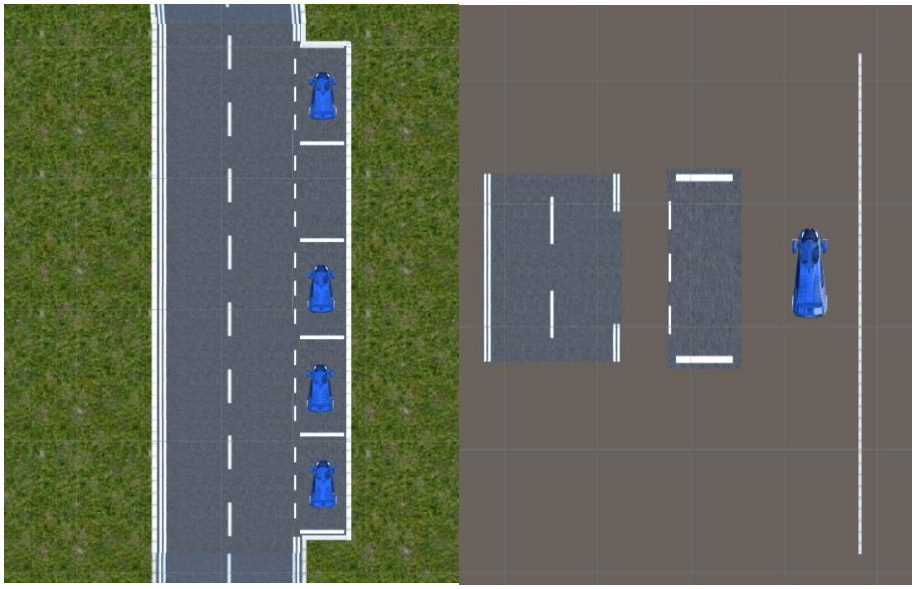
Staza je izrađena od unaprijed određenih blokova cesta. Koristio sam jedan blok ravne ceste duljine 30 metara te tri zakrivljena bloka od 30°, 60° i 90°. Cilj kod izrade staze bio je napraviti što više skretanja ulijevo kako bi se neuronska mreža mogla lakše utrenirati i za taj slučaj. Prvo sam napravio stazu koja je imala jedno blago lijevo skretanje, no automobil koji se trenirao na takvoj stazi na tim bi mjestima prelazio crtu. Problem sam mogao riješiti tako da sam stazu tijekom treniranja prošao prvo u jednom smjeru, pa zatim u drugom. Na kraju sam izradio novu stazu jer je na taj način vizualno zanimljivije.



Slika 3. Blokovi korišteni za izradu staze i izgled staze

6.2.2. Parking

Parking se sastoji od više podloga s različitim teksturama, modela automobila i rubnjaka. Različite sastavnice su spojene u Unityju, a smještene su pod zajednički objekt. Na linije od parkirališta koja su zauzeta, dodani su sudarači (*engl. Collider*) kako bi senzori daljine znali gdje se nalaze rubovi slobodnog parkirališta.

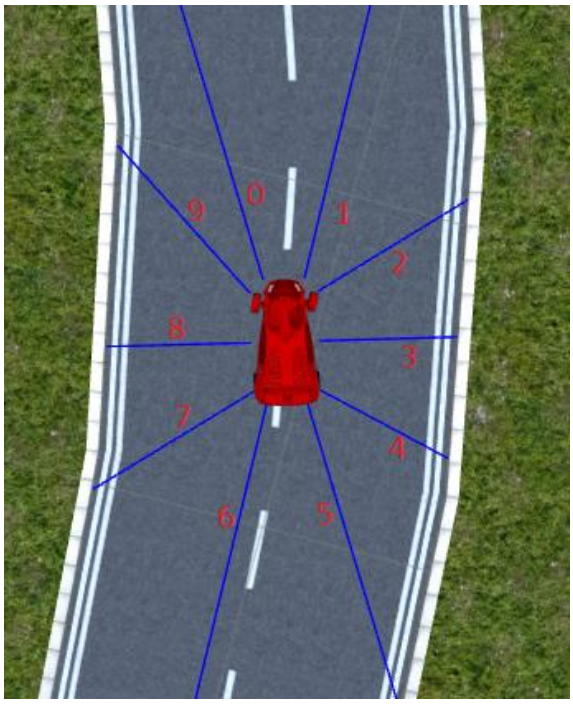


Slika 4. Blokovi korišteni za izradu parkirališta i izgled parkirališta

6.2.3. Automobil

Model automobila je preuzet iz standardnog paketa Unity trgovine. Htio sam da korisnik ima što bolju kontrolu nad automobilom, pa sam podesio parametre maksimalne brzine na 55 km/h, pojačao silu pritiska na podlogu, povećao trenje kotača u smjeru kretanja i uklonio bočno proklizavanje.

Sveukupno postoji 10 senzora, a razmještaj i orijentacija na automobilu su prikazani na slici 5. U simulaciji se automobilom može upravljati tipkovnicom ili neuronskom mrežom.

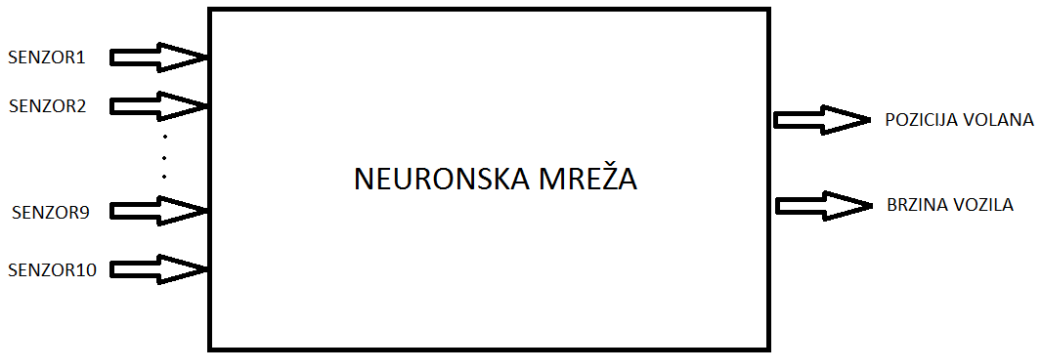


Slika 5. Pozicija i numeracija senzora na automobilu

6.3. Struktura neuronske mreže

Topologija neuronske mreže podijeljena je u tri dijela: ulazni sloj, skriveni sloj i izlazni sloj. Skriveni sloj može sadržavati više dodatnih slojeva. Isprobavao sam različite kombinacije broja skrivenih slojeva i broja neurona u njima i na kraju zaključio da što je kompleksnija mreža, to može lakše uhvatiti preciznije detalje nad ulaznim podatcima, ali isto tako traži puno više podataka i vremena za treniranje. Najbolji omjer vremena za treniranje i dobivenih rezultata davala je topologija koja ima jedan skriveni sloj s jednakim brojem neurona kao ulazni sloj.

Svaki sloj dodatno ima i neuron sa stalnim izlazom veličine 1. To omogućava da neuron u idućem sloju nakon množenja s težinom na ulazu lako može ostvariti pomak po tranzicijskoj funkciji što ubrzava i olakšava učenje neuronske mreže.



Slika 6. Opći izgled neuronske mreže za upravljanjem vozilom

Iz perspektive jednog neurona, on na ulaz prima izlaze svih neurona prethodnog sloja pomnožene težinom, a njegov izlaz primaju svi neuroni idućeg sloja. Iznimka su početni i završni sloj. Neuroni početnog sloja na svoj izlaz propuštaju ulazni signal, u ovom slučaju su to duljine primljene od senzora s automobila. Izlaze neurona završnog sloja koristimo kao izlaz neuronske mreže.

Za tranzicijsku funkciju obično se koriste funkcije $f : R \rightarrow [-1, 1]$. Zato sam odabrao tangens hiperbolni $f(x)= \tanh(x)$, a za njenu derivaciju sam uzeo dovoljno dobru aproksimaciju: $f'(x) = 1 - x^2$. Ta derivacija je potrebna kako bi se mogao ostvariti algoritam unazadnog širenja.

Kako su danas lako dostupni precizni senzori udaljenosti koji se montiraju na automobile, odlučio sam njih modelirati u simulaciji i neuronsku mrežu trenirati s tim podatcima. Senzori pomoću funkcije bacanja zrake (*engl. raycast)* očitavaju udaljenost do prve površine, a kako bi neuronska mreža mogla baratati s tim podatcima, dobivena udaljenost se skalira na segment od [0, 1].

Ako je μ(x) step funkcija, onda je funkcija kojom skaliramo pod (6-1)

$$f(x) = \frac{\mu(x)*x - \mu(x-8)*(x-8)}{8}$$

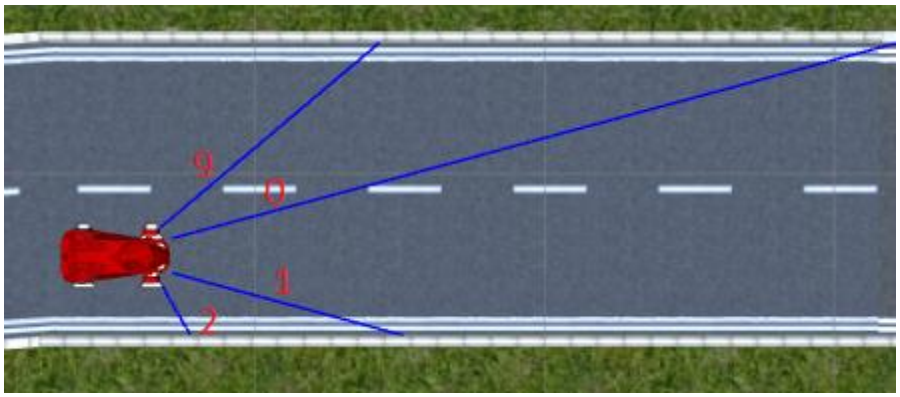
6.4. Neuronske mreže korištene u projektu

Sveukupno sam koristio 3 neuronske mreže:

1. Neuronska mreža koja upravlja automobilom tijekom vožnje
2. Neuronska mreža koja otkriva ima li u blizini automobila slobodan parking
3. Neuronska mreža koja izvodi manevar parkiranja

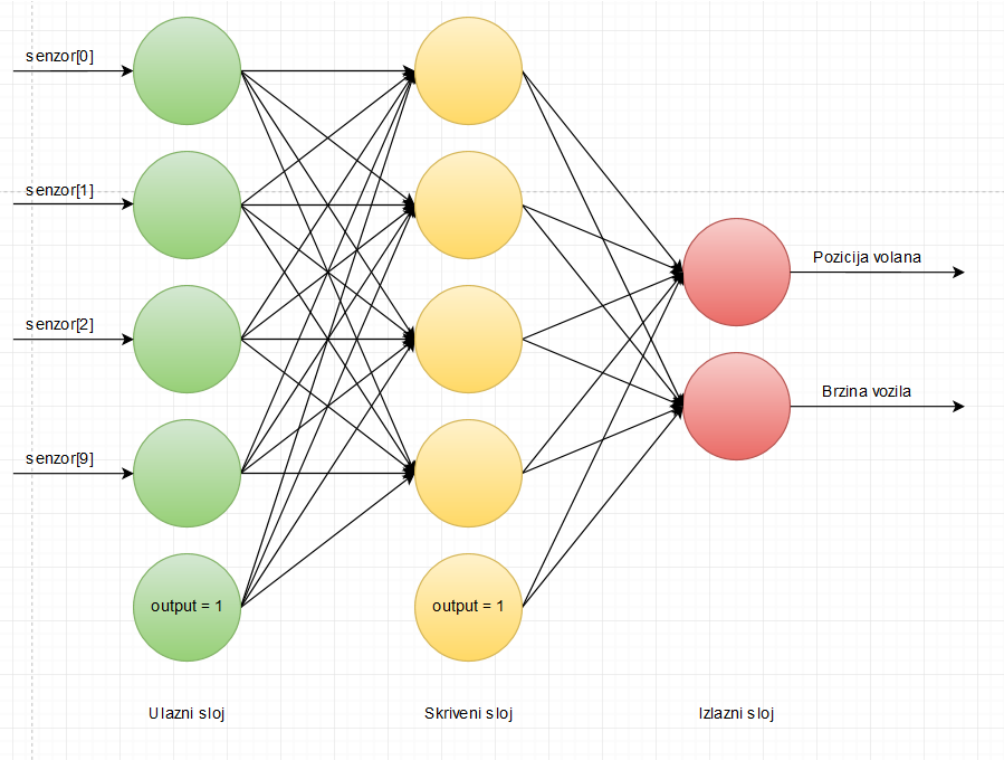
6.4.1. Neuronska mreža za vožnju

Kako se vozilo kreće samo prema naprijed, odabrao sam 4 senzora koji su tako i okrenuti kao ulaz u neuronsku mrežu, a kao izlaz, pozicija volana i brzina automobila.



Slika 7. Senzori korišteni za vožnju automobila

Prikupljanje podataka izveo sam tako što sam automobilom odvozio jedan krug po stazi, te bih svaki put kada bi se sličica na ekranu osvježila slijedno zapisao u datoteku trenutne udaljenosti dobivene od senzora, poziciju volana i brzinu kretanja automobila. Sveukupno je bilo izgenerirano 276 kB podataka.



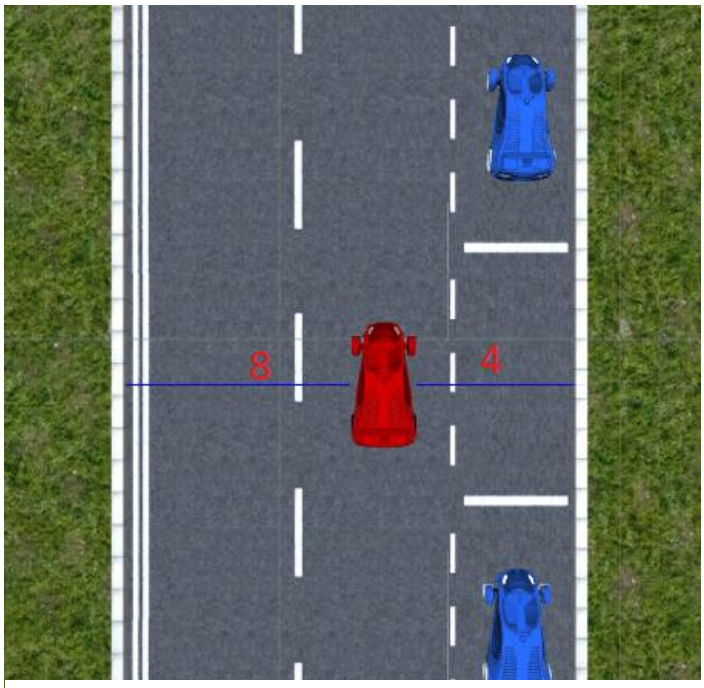
Slika 8. Neuronska mreža korištena za vožnju

Odabrao sam da neuronska mreža ima jedan skriveni sloj s četiri normalna neurona i jednim neuronom koji ima stalni izlaz. Takva mreža je na kraju utreniravanja davala najbolje rezultate.

Veoma sam zadovoljan rezultatom nakon utreniravanja jer automobil drži desnu stranu na cesti i prati liniju ceste bolje od mene dok sam vozio automobil. Neuronska mreža uspijeva ispraviti putanju automobila sve dok je kut između pravca ceste i automobila manji od 55°. Treba napomenuti da je ova neuronska mreža utrenirana na stazi koja ima konstantnu širinu, što znači da se, ako bi unijeli veće razlike u širini staze, ona se ne bi snalazila i vjerojatno bi davala loše rezultate.

6.4.2. Neuronska mreža za detektiranje parkinga

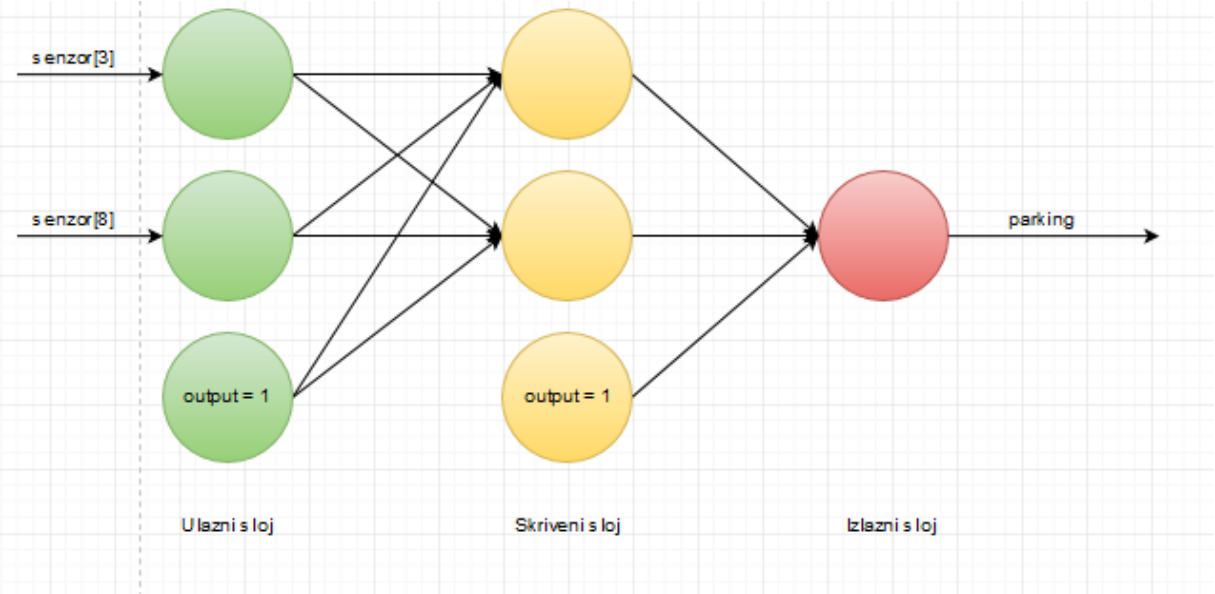
Ideja oko traženja parkinga je da neuronska mreža nađe udubine u stazi u koje automobil može stati. Ovo znači da neuronska mreža ne detektira stvarno parkirno mjesto, nego samo mjesto gdje bi se automobil mogao smjestiti, a da nije na stazi. Kako je za detektiranje potrebna širina staze, koristio sam samo jedan senzor s lijeve i jedan s desne strane, a za izlaz je odabrana vrijednost {0, 1} ovisno o tome da li se nalazi parking u blizini ili ne.



Slika 9. Senzori korišteni za detekciju parkinga

Kako je izlaz iz neuronske mreže realan broj u intervalu od [-1, 1], ispitivao sam da je li taj broj veći od 0.7 te bih, samo ako je, prihvatio da se parking nalazi desno od automobila.

Prva mreža koju sam utrenirao bila je mreža za vožnju automobila, pa sam zato prikupljanje podataka za traženje parkinga mogao automatizirati. To sam izradio tako što sam na stazi označio gdje se nalazi parking te bih, kada bi automobil bio u unutar te oznake, na izlaz proslijeđivao vrijednost 1. Kako je bitno da neuronska mreža ne dobiva puno redundantnih podataka pokretao sam simulaciju u blizini parkinga te je nakon desetak puta utrenirana mreža davala točne rezultate. Takvim postupkom sam sveukupno izgenerirao 28 kB podataka.



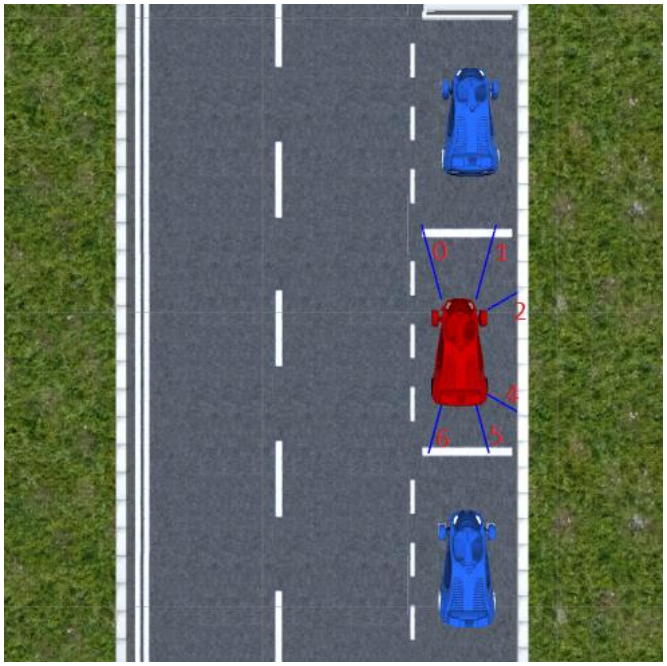
Slika 10. Neuronska mreža korištena za detekciju parkinga

Pri odabiru topologije neuronske mreže vodio sam se istom logikom te odabrao da jedan skriveni sloj ima jednak broj čvorova kao i ulazni. Nisam imao potrebe mijenjati topologiju neuronske mreže jer je odmah nakon utreniravanja davala točne rezultate. Kako je i ova neuronska mreža utrenirana na istoj širini staze, vjerojatno bi došlo do krivih rezultata ako bismo promijenili njenu širinu.

6.4.3. Neuronska mreža za parkiranje

Ova neuronska mreža je najkompliciranija do sada jer treba izvesti manevar koji ovisi o poziciji u odnosu na parking. Zato sam za ulaz u neuronsku mrežu koristio senzore koji su okrenuti prema naprijed i nazad.

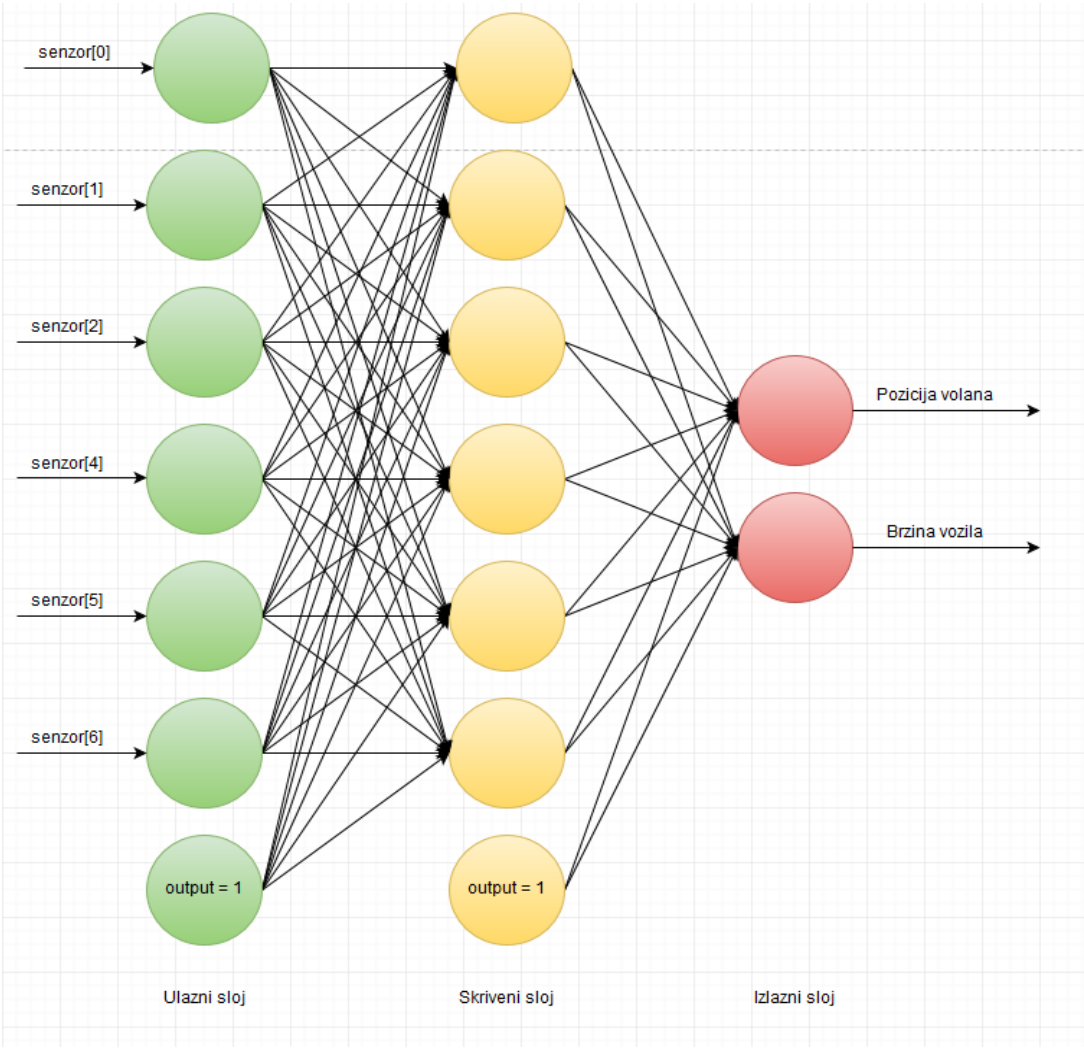
Prikupljanje podataka izveo sam tako da bih pustio prve dvije neuronske mreže da nađu parking i zaustave vozilo blizu njega, te bi onda ja ručno izvodio manevar parkiranja. Takav manevar sam morao ponoviti pedesetak puta kako bi mreža imala dovoljan broj podataka da se može uspješno utrenirati. Krajnja veličina datoteke s podatcima je 476 kB.



Slika 11. Senzori korišteni za uparkiravanje

Rezultati utreniravanja neuronske mreže isprva nisu bili zadovoljavajući jer je manevar koji je neuronska mreža obavljala redovito završavao na krivome mjestu. Zbog toga sam još dodatno prikupio podatke za parkiranje. Ovaj puta su rezultati bili bolji. Težinske vrijednosti kojima se množe izlazi neurona na početku su nasumične, tako da je rezultantna

neuronska mreža svaki put drugačija. Baš zbog te nasumičnosti sam uvijek po više puta utrenirao neuronsku mrežu, a do sad najbolju neuronsku mrežu bih spremio u datoteku.



Slika 12. Neuronska mreža korištena za uparkiravanje

Broj neurona skrivenoga sloja sam odabrao teko da bude jednak broju ulaznih čvorova. Za većinu drugih topologija neuronska mreža bi davala lošije rezultate. Zanimljivo je da za ovakvu topologiju ukupno postoji čak 56 veza, te je zbog toga utreniravanje osjetno duže trajalo.

6.5. Postupak utreniravanja neuronske mreže

Utreniravanje neuronske mreže se može rastaviti na nekoliko koraka:

1. Oblikovanje topologije neuronske mreže
2. Prikupljanje podataka
3. Propagiranje ulaznih podataka kroz mrežu
4. Usporedba rezultata na izlazu mreže s očekivanim vrijednostima
5. Utreniravanje neuronske mreže algoritmom unazadnog širenja
6. Ponavljanje koraka 2.-5. dok nismo zadovoljni s točnošću izlaza neuronske mreže

6.5.1. Propagiranje ulaznih podataka

Neuronsku mrežu možemo zamisliti kao običnu funkciju koja za svoje ulaze daje izlaz. Dapače, neuronska se mreža može i zapisati kao funkcija. Ono što jedan neuron napravi je da signal s ulaza pomnoži s težinskom funkcijom i zatim na to primjeni tranzitivnu funkciju.

Propagiranjem ulaznih podataka kroz neuronsku mrežu računamo izlaze svih neurona, sloj po sloj, da bismo na kraju saznali rezultat kojeg daje neuronska mreža.

6.5.2. Algoritam unazadnog širenja

Algoritam unazadnog širenja (*engl. Backpropagation*) služi kako bi neuronska mreža svakim novim podatkom daje sve točnije rezultate. Algoritam prvo napravi propagiranje ulaznih podataka te potom osvježi težinske funkcije svakog neurona.

Postoje 2 parametra koja utječu na učenje neuronske mreže:

- η (eta) – kojom mjerom novi podatci utječu na pomake težinskih funkcija, a zadaje se na segmentu od [0, 1].
- α (alfa) – kojom mjerom prethodni pomak težinske funkcije utječe na sadašnji, a u proizvoljnom je segmentu od [0 , η]. Dodaje momentum učenju.

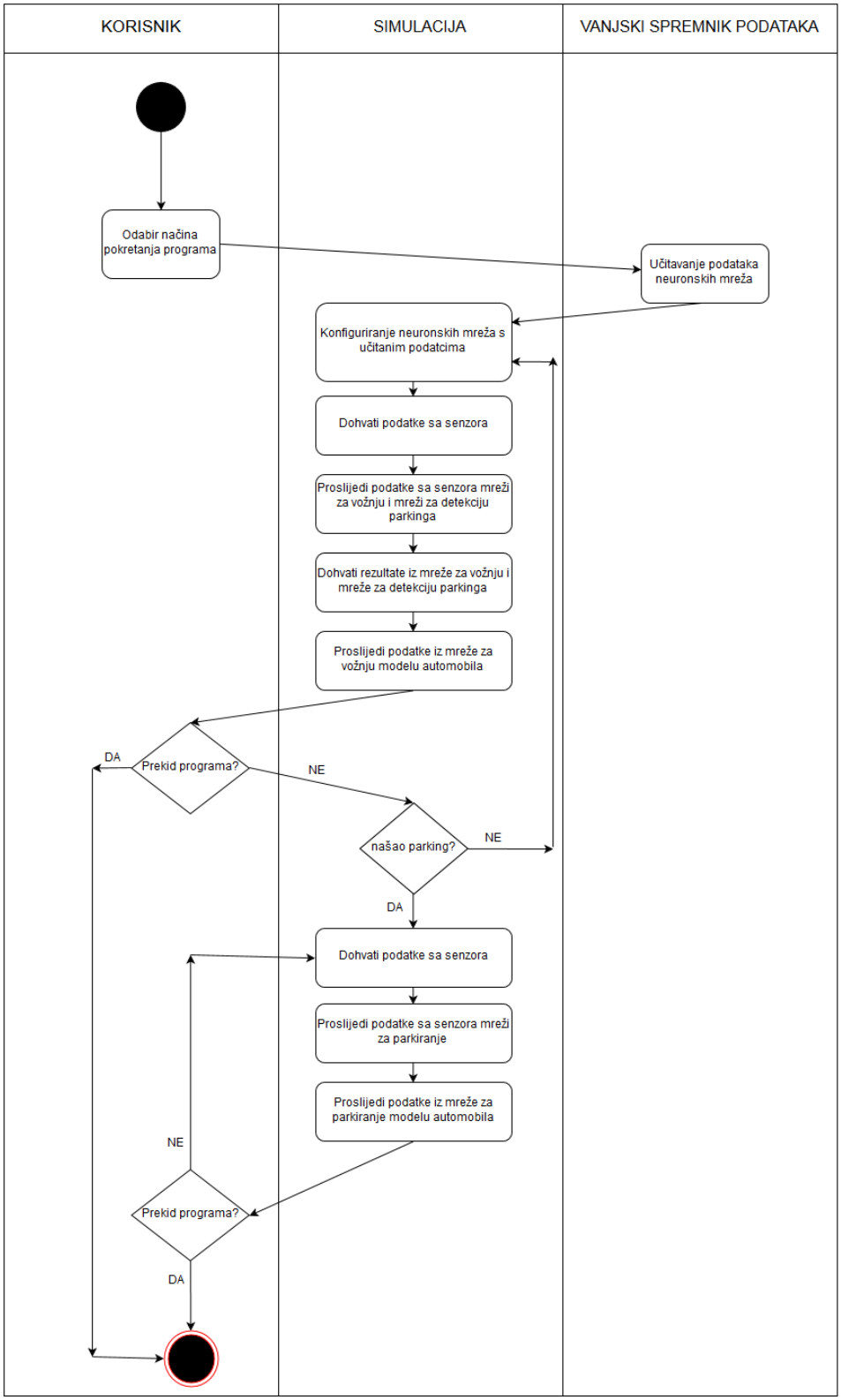
Slijedi pseudokod algoritma unazadnog širenja:

```
zaSvaki podatak = novi podatak za treniranje
        izlazMreža = mreža.propagiraj( podatak.ulaz )

        greška = (izlazMreža – podatak.izlaz) na izlaznim neuronima

        zaSvaki sloj počevši od izlaznog
                zaSvaku težinu
                     novaDeltaTežina =
                          η * neuron.trenutniIzlaz() * neuron.gradijent
                          + α * staraDeltaTežina

        zaSvaki  neuron
                osvježi težinsku funkciju
```

6.6. Dijagram aktivnosti

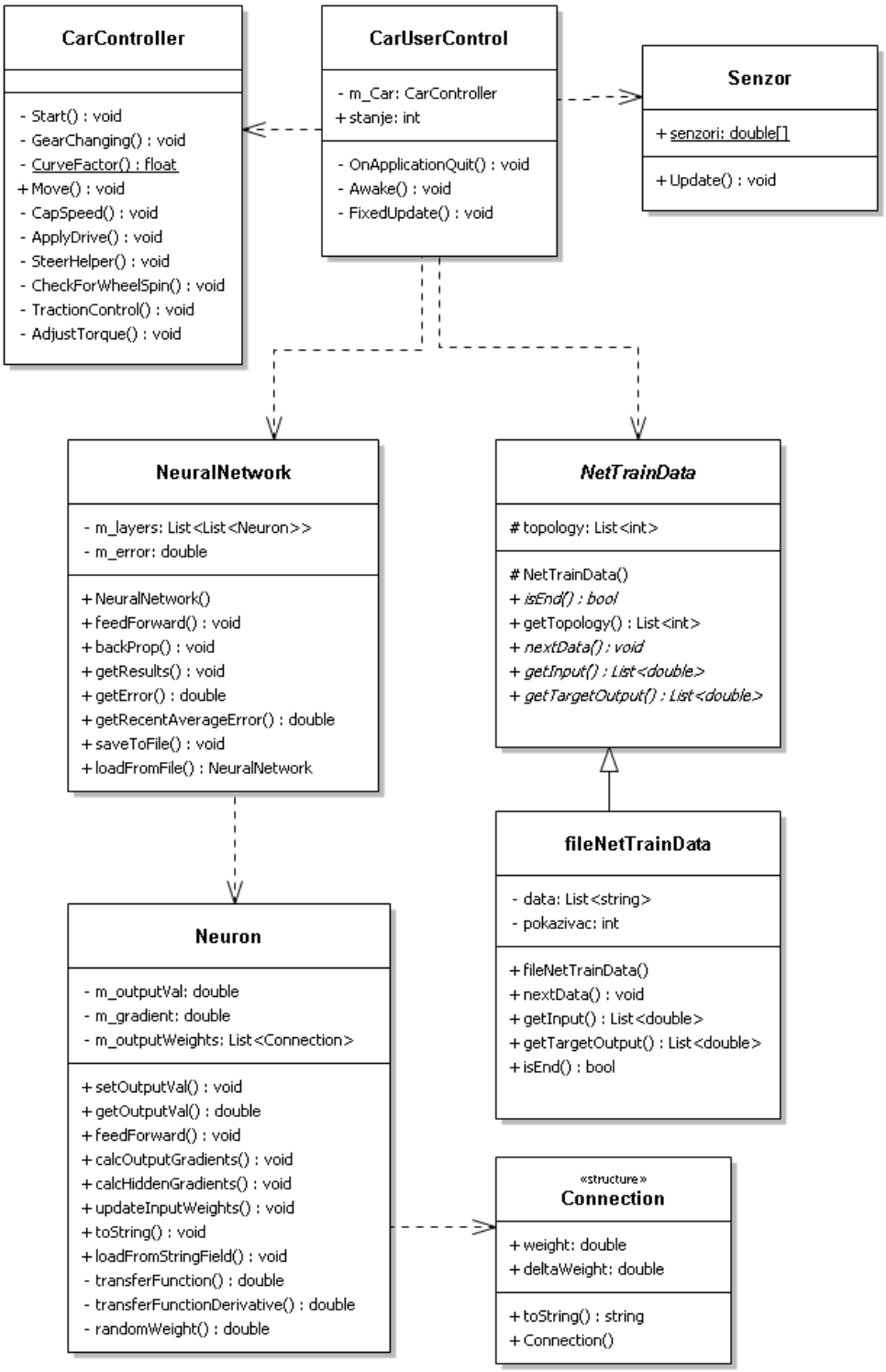
Na slici 13. prikazan je dijagram aktivnosti koji objašnjava način na koji se izvršava program u slučaju kada u datoteci imamo spremljene sve utrenirane neuronske mreže potrebne za vožnju i parking.

Program se u ovom slučaju odvija u 3 faze. Prva faza je učitavanje podataka iz datoteka i konfiguriranje neuronskih mreža. Druga faza je paralelno pokretanje neuronskih mreža za vožnju i detektiranje parkinga sve dok se ne nađe parking. Nakon što mreža za traženje parkinga signalizira da je parkiranje moguće automobil lagano usporava. Treća faza je izvođenje manevra za parkiranje pomoću posljednje neuronske mreže. Korisnik može u bilo kojem trenutku prekinuti simulaciju pritiskom tipke za izlaz.



Slika 13. Dijagram aktivnosti

6.7. Dijagram razreda s opisom



Slika 14. Dijagram razreda

Razred **CarController**

Služi za upravljanje automobilom, a Move() je jedino potrebna metoda da bi se upravljalo automobilom. Auto se pomiče tako da za svaku sličicu igre prosljeđuju podatci o gasu, poziciji volana i kočnici. Na temelju tih podataka izračunava se trenutna brzina i rotacija modela automobila u globalnom prostoru. Automobil se može konfigurirati preko javnih varijabli za maksimalnu brzinu, jačinu motora, maksimalni kut skretanja i mnogih drugih.

Razred **CarUserControl**

Glavni razred koji objedinjuje sve ostale klase. U metodi Awake() se postavljaju početna stanja podataka, učitavaju se potrebne datoteke i, ako je potrebno, utrenira se neuronska mreža s učitanim podatcima. Metoda FixedUpdate() se poziva prije svakog iscrtavanja sličice, a u njoj je ostvarena komunikacija sa standardnim ulazom i neuronskim mrežama, te se po potrebi podatci o vožnji zapisuju u datoteku.

Razred **Senzor**

Jednostavan razred koji za zadane pozicije i pravce senzora određuje koliko je udaljen najbliži objekt. Ti podatci su onda dostupni za daljnje korištenje.

Razred **NeuralNetwork**

Kod izrade objekta ovog razreda prosljeđujemo mu topologiju neuronske mreže, a on instancira potreban broj neurona i sprema ih u m_layers.

Metoda feedForward() izvršava algoritam propagiranja ulaznih podataka po neuronskoj mreži. Metoda backProp() izvršava algoritam unazadnog širenja. Metoda getResults() vraća listu izlaznih vrijednosti iz neuronske mreže. Razred ima mogućnost pretvorbe svojih podataka u string vrijednost kako bi se olakšalo zapisivanje u datoteku.

Razred **Neuron**

Konstruktor ovog razreda povezuje novoizrađeni neuron s neuronima idućeg sloja te postavlja težinske funkcije na slučajnu vrijednost.

Ovaj razred opisuje ponašanje neurona u neuronskoj mreži. Metoda feedForward() izračunava na temelju ulaza u neuron njen izlaz. Metode calcOutputGradients() i calcHiddenGradients() računaju po formuli gradijent promjene težinske funkcije. Metoda updateInputWeights() se koristi u algoritmu unazadnog širenja te računa nove težinske vrijednosti od tog neurona. Također posjeduje metode za pretvaranja u string vrijednost kako bi se olakšalo zapisivanje u datoteku.

Struktura **Connection**

Ova struktura olakšava baratanje s težinskim funkcijama i ima mogućnost pretvaranja podataka u string vrijednost za lakše zapisivanje u datoteku.

Razred **NetTrainData**

Ovo je apstraktni razred koji opisuje što treba implementirati da bi razred CarUserControl mogao koristiti podatke za utreniravanje i korištenje neuronske mreže.

Razred **fileNetTrainData**

Ovaj razred nasljeđuje razred NetTrainData i definira funkcije za učitavanje podataka iz datoteke. Konstruktor otvara datoteku za čitanje i na temelju njih konstruira topologiju za neuronsku mrežu. Metode getInput() i getTargetOutput() čitaju podatke i vraćaju ih u potrebnom formatu. Metoda isEnd() ispituje je li pročitao sve podatke iz datoteke. Metoda nextData() pomiče pokazivač na idući redak podataka.

7. Zaključak

Autonomna vozila pripadaju ne tako dalekoj budućnosti, a tehnologija potrebna za njegovo ostvarenje već postoji. Danas su prisutna djelomično autonomna vozila, poput Googleovog i Teslinog autonomnog vozila, a ona su u javnosti vrlo pozitivno prihvaćena.

U ovoj simulaciji, neuronska mreža pokazala se kao zadovoljavajući izbor za vožnju i parkiranje automobila. U realnom svijetu, uz nju bi morali implementirati niz sustava koji bi pazili na sigurnost izvršavanja akcija koje ona naredi. Samo tada bi autonomna vožnja i parkiranje bili dovoljno sigurni za upotrebu kod krajnjih korisnika.

Izrada neuronskih mreža i simulacije je novo iskustvo za mene i pokazalo se da ću znanje koje sam stekao na završnom radu moći primijeniti u mnogim drugim područjima.

8. Literatura

[1] Francis, T.S.Yu. Suganda, J. *Optical Signal Processing, Computing, and Neural Networks*. Pennsylvania: John Wiley & Sons, 1992

[2] Bašić, B. Čupić, M. Šnajder, J. Predavanja: *Umjetne neuronske mreže*. Zagreb, 2012.

[3] Miller, D. *Neural Net in C++*. *http://www.millermattson.com/dave/*.

[4] Unity Technologies *Unity3D*. *https://unity3d.com/*

[5] Unity Technologies *Standard Assets*. *https://www.assetstore.unity3d.com/en/#!/content/32351*

[6] J. Jankowski *SimpleModular Street Kit*. *https://www.assetstore.unity3d.com/en/#!/content/13811*

[7] Blender Foundation *Blender*. *https://www.blender.org/*

Sažetak

Simulacija autonomnog parkiranja vozila

Ovaj završni rad implementira simulaciju vožnje i parkiranja automobila voženu neuronskom mrežom. Obrađen je postupak izrade simulacije automobila, izrade neuronske mreže, te je objašnjen algoritam propagiranja ulaznih podataka i unazadnog širenja kod neuronskih mreža. Korisnik može tipkovnicom upravljati automobilom, odvezenom vožnjom utrenirati neuronsku mrežu, te ju pustiti da izvede naučeni manevar.

Ključne riječi: simulacija uparkiravanja vozila, virtualno okruženje, umjetna inteligencija, neuronska mreža, Unity3D, C#

Abstract

Simulation of Self-Parking Vehicles

This final work implements driving and parking simulation which is navigated with neural network. In this work it is described how to make a car simulation, how to make neural network and how backpropagatoin and feed forward algorithms work. Users can navigate a car using tje keyboard and train neural network with data they generated, after all he can let trained neural network drive the car on its own.

Keywords: driving and parking simulation, virtual environment, artificial intelligence, neural network, Unity3D, C#