SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5493

Strojno učenje u autonomnom upravljanju vozilom

Marko Gulan

Zagreb, lipanj 2018.

Prvenstveno se želim zahvaliti mojoj mentorici, prof. dr. sc. Željki Mihajlović, koja mi je svojim znanjem i trudom pomogla napisati ovaj završni rad.

Zahvaljujem se svim svojim prijateljima i kolegama što su bili uz mene tijekom mojeg dosadašnjeg studija i što su mi studij učinili ljepšim.

Najveću zahvalu iskazujem mojoj obitelji, a posebno roditeljima, jer su bili moja najveća moralna i financijska podrška i što su se uvijek trudili usmjeriti me na pravi put.

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

Zagreb, 13. ožujka 2018.

ZAVRŠNI ZADATAK br. 5493

Pristupnik: **Marko Gulan (0036493100)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Strojno učenje u autonomnom upravljanju vozilom**

Opis zadatka:

Proučiti osnovne tehnike strojnog učenja pogodne za upravljanje vozilom. Razraditi prikaz scene koja sadrži razne prepreke te prikaz vozila u takvom okolišu. Razraditi upravljanje vozilom u okolišu s različitim preprekama upotrebom tehnika strojnog učenja. Na različitim primjerima prikazati ostvarene rezultate. Načiniti ocjenu rezultata i implementiranih postupaka.
Izraditi odgovarajući programski proizvod. Koristiti grafički programski pogon Unity i programski jezik C#. 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: 16. ožujka 2018.
Rok za predaju rada: 15. lipnja 2018.

Mentor:
Prof. dr. sc. Željka Mihajlović

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Danas ljudi sve više vremena provode u automobilima, na motociklima, biciklima i drugim vozilima. Nažalost, razmjerno tomu, povećava se i broj prometnih nesreća. Sudeći prema istraživanjima Udruge za sigurnost međunarodnog cestovnog prometa, 1,3 milijuna ljudi godišnje umire u sudarima dok 20-30 milijuna ljudi biva ozlijeđeno [1]. Razlozi za to su mnogobrojni, a neki od najčešćih su: gubitak kontrole na vozilom, neuspjeli pokušaji zaustavljanja ili skretanja, nepažnja ili ometenost i brojni drugi [2].

Zbog navedenih razloga, pokazala se potreba za autonomnom vožnjom. Na taj će način, ljudi koji mogu osjećati umor ili izgubiti koncentraciju biti zamijenjeni strojevima kojima se to ne događa. Tako od prije nekoliko godina postoje automobili koji se mogu samostalno voziti, parkirati, izbjegavati prepreke i slično, ali uz stalan nadzor vozača te uz idealne vremenske uvjete.

Razvojem umjetne inteligencije, autonomnost postaje sve veća. 2000-ih godina se razvio sustav za automatsko parkiranje, a već za 2020. godinu, velik broj najpoznatijih proizvođača automobila najavljuju potpuno autonomno vozilo [3].

Cilj ovog rada je upoznavanje osnovnih tehnika strojnog učenja pogodnih za upravljanje vozilom. Korištenjem grafičkog programskog pogona Unity prikazana je simulacija kako se vozilo može naučiti voziti po otvorenom okolišu pomoću tih tehnika.

2. Korištene tehnologije

2.1. Grafički programski pogon Unity

Unity je grafički program pogon namijenjen za razvoj 2D i 3D video igara i simulacija. Razvijen je 2005. godine od strane tvrtke Unity Technologies. Jedna od njegovih najvažnijih karakteristika je neovisnost o platformi pa se, stoga, igre i simulacije mogu razvijati za računala , igraće konzole i mobilne uređaje.

Skripte koje upravljaju objektima mogu se pisati u 3 programska jezika, a to su C#, JavaScript i Boo. Potonji je već nekoliko godina zastario dok je proces zastare za JavaScript pokrenut 2017. godine. Zbog toga, u ovom radu, skripte su pisane u programskom jeziku C#.

2.2. Programski jezik C#

C# je jednostavan i moderan objektno-orijentirani programski jezik. Podržava raznorazne principe programskog inženjerstva poput provjera granica polja, pokušaja korištenja neinicijaliziranih varijabli, automatskog skupljanja smeća (engl. *garbage collection*) i brojnih drugih.

2.2. Format JSON

JSON (engl. *JavaScript Object Notation*) je format koji se koristi za razmjenu podataka pritom koristeći ljudima razumljiv tekstualni zapis. Izveden je iz programskog jezika JavaScript, ali je, usprkos tomu, neovisan o programskom jeziku.

3. Umjetne neuronske mreže

Od samog izuma računala, ljudi se trude predstaviti ljudsko znanje na računalu. Tako su računalni znanstvenici počeli proučavati istraživanja iz područja neurofiziologije i kognitivne znanosti. Te znanosti su dokazale da se mozak sastoji od preko 100 milijardi živčanih stanica koje se još nazivaju neuroni. Na temelju toga, 1943. godine osmišljen je model umjetnog neurona.

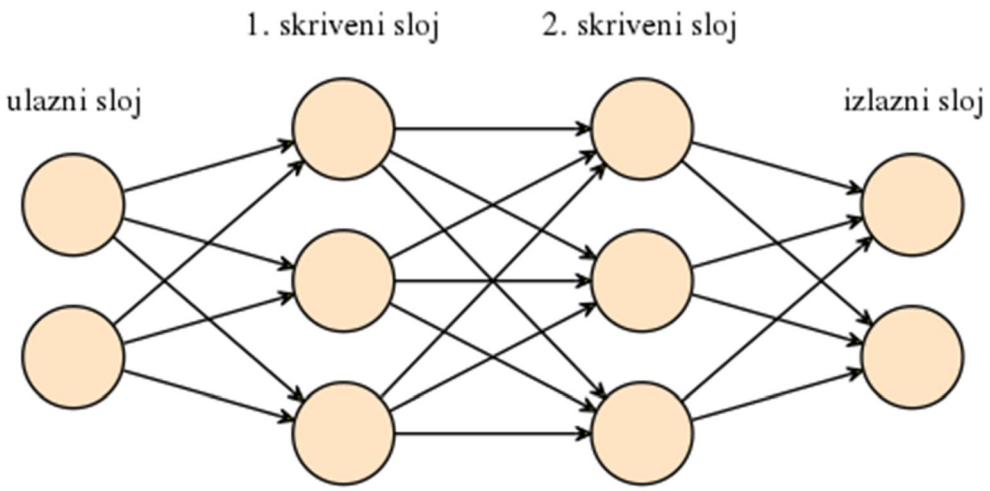
3.1. Umjetni neuron

Umjetni neuron na ulazu dobiva niz varijabli koje predstavljaju pobudu od neurona iz prethodnog sloja. Koliko će svaka pobuda utjecati na neuron definirano je težinom na način da signal koji dolazi na neki od ulaza se množi s korespodentnom težinom. Stoga je efektivno djelovanje signala definirano umnoškom tih dviju vrijednosti.

Nakon što se izračunaju efektivna djelovanja svih pobuda ista se sumiraju te se dodaje još konstantan pomak (engl. *bias*). Suma se dovodi na ulaz prijelazne funkcije (engl. *transfer function*) koja predstavlja konačan izlaz neurona. Zadaća prijelazne funkcije je ograničiti izlaz neurona na željeni interval.

3.2. Višeslojne neuronske mreže

Umjetna neuronska mreža se gradi povezivanjem više umjetnih neurona. U ovisnosti o više parametara, postoji nekoliko vrsta umjetnih neuronskih mreža, no u ovom radu je naglasak stavljen na potpuno povezane unaprijedne slojevite umjetne neuronske mreže. U takvoj mreži ne postoje ciklusi u obradi podataka. Neuroni su podijeljeni u dobro definirane slojeve te neuroni u sloju i dobivaju pobude od svih neurona iz sloja $i-1$. Primjer jedne takve neuronske mreže prikazan je na slici 3.1.



Slika 3.1 Potpuno povezana unaprijedna slojevita umjetna neuronska mreža

Svaka neuronska mreža se sastoji od 3 sloja: ulaznog, skrivenog i izlaznog sloja pri čemu skriveni sloj može sadržavati više podslojeva kako je prikazano na slici 3.1. Neuroni ulaznog sloja se jedini ne ponašaju na način objašnjen u potpoglavlju 3.1. Naime, zadatak ovih neurona je samo podatke dobivene izvana proslijediti na izlaz neurona, a ostatak mreže ih treba obraditi.

Neuroni ostalih slojeva se ponašaju na očekivan način. Pri tomu, izlazi neurona iz sloja $i-1$ predstavljaju ulaze neuronima iz sloja i. Izlazi izlaznog sloja predstavljaju obrađene podatke te se oni vraćaju korisniku.

Kako bi se postiglo da se neuronska mreža ponaša na željeni način, potrebno ju je učiti. Algoritmi učenja mreže to čine na način da modificiraju njezine parametre, a jedini parametri koje je moguće modificirati su težine.

3.3. Implementacija neuronske mreže

Za svaki sloj, osim za izlazni sloj, pamte se jednostupčana matrica konstantnih pomaka i matrica težina koje povezuju taj sloj s prethodnim. Odabran je matrični zapis zato što je jednostavan za računalo. Uz to, ako se navedeni parametri prikažu kao matrice, funkcionalnost svakog neurona može se prikazati jednostavnom formulom: $i = f(\boldsymbol{w} \cdot a + \boldsymbol{b})$. Pri tome, i označava izlaz neurona, $\boldsymbol{w}$ matricu konstantnih pomaka, a ulazne pobude, $\boldsymbol{b}$ matricu konstantnih pomaka, a f prijelaznu funkciju.

```
public float[] FeedForward(float[] a)
{
    var pairs = _biases.Zip(_weights, (b, w) => new { b, w });
    Matrix result = Matrix.FromArray(a, a.Length, 1);
    foreach (var pair in pairs)
    {
        result = pair.w.Dot(result).Add(pair.b).Map(Tanh);
    }
    return result.ToArray();
}
```

Isječak koda 3.1 Osnovni algoritam umjetne neuronske mreže

3.4. Struktura neuronske mreže korištene u radu

Neuronska mreža korištena u radu se sastoji od 4 sloja. Ulazni sloj podatke dobiva od 5 senzora udaljenosti koji vraćaju udaljenost od najbliže prepreke. Svaki senzor ima maksimalnu udaljenost koju može učitati te ukoliko se na toj udaljenosti ili bliže ne nalazi nikakva prepreka, senzor vraća unaprijed definiranu vrijednost. Kako je vidljivo na Isječku koda 3.2. ta vrijednost je postavljena na maksimalnu udaljenost koju može učitati senzor uvećana za 1.

```
float CheckSensor(GameObject sensor, Vector3 position, float angle)
{
    Vector3 direction = Quaternion.AngleAxis(angle, transform.up) * transform.forward;

    RaycastHit hit;
    if (Physics.Raycast(position, direction, out hit, sensorLength))
    {
        if (debug) DrawSensor(sensor, position, hit.point, Color.blue);

        return hit.distance;
    }

    return sensorLength + 1;
}
```

Isječak koda 3.2 Očitavanje udaljenosti sa senzora

Oba skrivena sloja sadrže po 10 neurona, a izlazni sloj se sastoji od 2 neurona. Izlaz jednog od neurona predstavlja brzinu vozila dok izlaz drugog predstavlja rotaciju vozila. S obzirom da se vozilo može kretati naprijed i nazad te rotirati lijevo i desno, za prijenosnu funkciju je odabrana funkcija tangens hiperbolni. Njezina domena je skup realnih brojeva, a kodomena skup $< -1, 1 >$ što odgovara potrebama kretanja i rotiranja u dva smjera. Naime, negativni brojevi predstavljaju kretanje unazad i rotaciju u lijevo, a pozitivni predstavljaju kretanje naprijed i rotaciju u desno.

4. Genetski algoritam

Genetski algoritam je prvenstveno inspiriran Darwinovom teorijom evolucije. Ta teorija se temelji na prirodnoj selekciji u kojoj veću šansu za preživljavanje i razmnožavanje imaju jedinke bolje prilagođene uvjetima života. Potomci tih jedinki nasljeđuju od roditelja genetski materijal pa su, stoga, vrlo slični roditeljima.

4.1. Opis algoritma

Genetski algoritam započinje stvaranjem populacije slučajno generiranih mozgova. Sve mozgove vrednujemo takozvanom funkcijom dobrote (engl. *fitness function*). U skladu s Darwinom teorijom evolucije, ona označava koliko je neka jedinka dobro prilagođena uvjetima života. Nakon toga se odabiru slučajno tri jedinke, a najbolja od njih predstavlja prvog roditelja. Isto se ponovi za drugog roditelja. Te dvije jedinke se križaju čime nastaje nova jedinka. Dodatno se nad tom jedinkom provede mutacija. Jedinku dodajemo u novu populaciju. Postupak odabira roditelja i stvaranje djeteta ponavlja se sve dok veličina nove populacije ne bude jednaka veličini trenutne populacije. Nakon što se stvori nova populacija, ona postaje trenutna populacija, vrednuje se i algoritam se može ponoviti. Algoritam se može ponavljati beskonačno, a u praksi se ponavlja sve dok trenutna populacija ne daje zadovoljavajuće rezultate.

Križanje (engl. *crossover*) je genetski operator kojim se od dviju jedinki dobiva dijete koje nasljeđuje od svakog roditelja po dio genetskog materijala. Drugi genetski operator važan za funkcioniranje genetskog algoritma je mutacija (engl. *mutation*). Mutacija se provodi na način da se definira vjerojatnost mutacije gena (engl. *mutation rate*) koja se definira kao vjerojatnost da neki gen promijeni svoju vrijednost. Potom se iterira po svim genima i sa zadanom vjerojatnošću mijenjamo njihove vrijednosti.

4.2. Implementacija genetskog algoritma

Mozgovi koje genetski algoritam koristi modelirani su neuronskim mrežama. Točnije, svako vozilo unutar populacije ima vlastitu neuronsku mrežu koja upravlja njime. Prema algoritmu, na početku je populacija slučajno generirana, a nakon toga se provodi algoritam selekcije prikazan u Isječku koda 4.1.

```
public void Selection() {
    List<Vehicle> newVehicles = new List<Vehicle>(Size);

    Vehicle bestVehicle = FindBestVehicle();
    bestVehicle.Crashed = false;
    bestVehicle.Completed = false;
    newVehicles.Add(bestVehicle);

    for (var i = 1; i < Size; i++)
    {
        var parentA = ChooseParent();
        var parentB = ChooseParent();
        var child = Vehicle.Crossover(parentA, parentB);
        Vehicle.Mutate(child, _mutationRate);
        newVehicles.Add(child);
    }
    _vehicles = newVehicles;
}
```

Isječak koda 4.1 Algoritam selekcije

Iz Isječka koda 4.1. je vidljivo je kako se u novu populaciju uvijek dodaje najbolje vozilo iz trenutne populacije. To se čini da se najbolje rješenje ne bi izgubilo, a svojstvo koje ne dopušta gubitak takvog rješenja se naziva elitizam.

Bolju funkciju dobrote imat će ona vozila koja su uspjela doći do cilja u odnosu na ona koja su sudarila s preprekom. Uz to, funkcija dobrote ovisi o vremenu koje je vozilu bilo potrebno da dođe do cilja ili se sudari s nekom preprekom. Pošto se vozilo može voziti u krug što bi dovelo do toga da vozilo nikad ne završi svoju vožnju, osmišljeno je da svako vozilo ima životni vijek te ako ga prekorači računa se kao da se sudario s nekom preprekom.

Križanje se obavlja na način da se slučajno generira pozicija gena na kojoj se roditelji cijepaju. Dijete se stvara na način da od prvog roditelja preuzme dio genetskog

materijala do točke prijeloma, a od drugog roditelja dio od točke prijeloma. Navedeni način križanja je poznat pod nazivom Križanje s jednom točkom prijeloma, a prikazan je na Isječku kodu 4.2.

```
public static Matrix Crossover(Matrix matrixA, Matrix matrixB)
{
    float[,] data = new float[matrixA.Rows, matrixB.Cols];
    int mid = random.Next(data.Length);
    for (int i = 0; i < matrixA.Rows; i++) {
        for (int j = 0; j < matrixB.Cols; j++)
        {
            if (i * matrixA.Cols + j > mid)
            {
                data[i, j] = matrixA._data[i, j];
            }
            else
            {
                data[i, j] = matrixB._data[i, j];
            }
        }
    }
    return new Matrix(matrixA.Rows, matrixB.Cols, data);
}
```

Isječak koda 4.2 Križanje dviju jedniki

Parametri koji se mogu mijenjati kod genetskog algoritma su veličina populacije i vjerojatnost mutacije gena. U radu je veličina postavljena na 50 jedinki, a vjerojatnost mutacije gena iznosi 5%.

Učenje je provođeno na računalu s dvojezgrenim procesorom frekvencije takta 1.7 GHz, 4GB radne memorije i grafičkom karticom osnovne frekvencije 1029 MHz. Za teren bez prepreka i terene s preprekama koje su daleko od ruba bilo je potrebno svega 30 minuta da se vozilo nauči doći do cilja. Pod pojmom daleko se podrazumijeva da između ruba i svake prepreke ima dovoljno prostora da prođe vozilo. Za terene sa slučajno raspoređenim preprekama bilo je potrebno oko 2 do 3 sata.

5. Struktura scene

5.1. Teren

Teren je jednostavnog kvadratnog oblika ograđen s 4 zida. Prvi pokušaj izrade terena bio je osmisliti teren za koji će se generirati visinska mapa. To je stvaralo velike probleme jer vozilo se nije uspjelo naučiti na takav teren. Uz nekoliko izmjena parametara neuronske mreže i genetskog algoritma, vozilo nije davalo zadovoljavajuće rezultate. Stoga je odluka pala na ravan teren. Kako bi se dobilo na realističnosti, na teren je dodana tekstura trave.

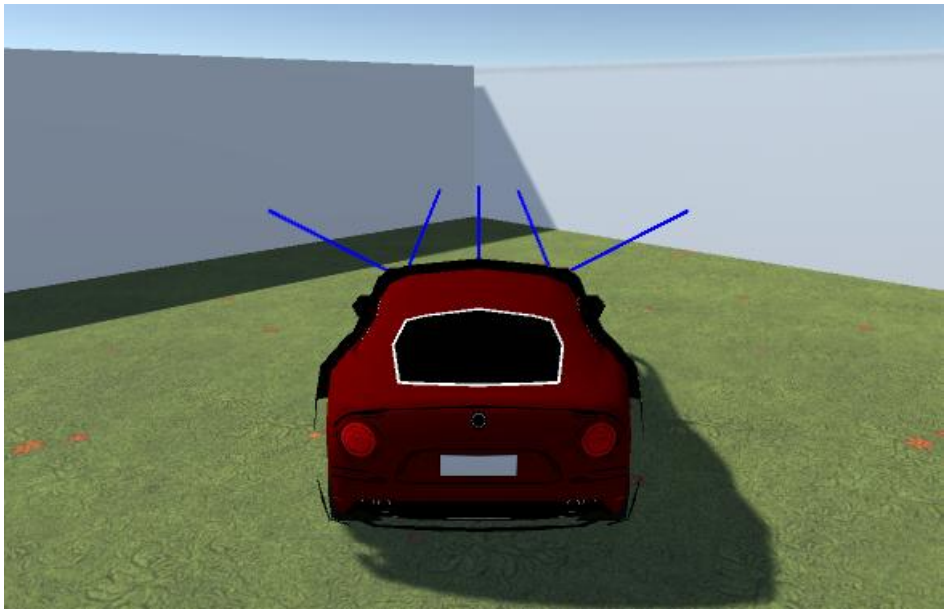
Na teren su dodane prepreke, odnosno stabla. Prepreke su implementirane na način da je na njih dodana komponenta sudarač (engl. *collider*). Ta komponenta služi za otkrivanje sudara između prepreka i vozila. Također nam pomažu pri otkrivanju najbliže prepreke što nam je potrebno za senzore na automobilu.

Slika 5.1 Teren

Svaki puta kada se simulacija pokrene, po terenu se slučajno raspoređuju prepreke. Ako se na terenu ne generira nijedna prepreka, vozilo se jako brzo naučiti voziti uz rub i tako stići uz cilj. Slično ponašanje vozilo ima ako se prepreke generiraju daleko od samog ruba.

5.2. Vozilo

Model vozila je preuzet iz standardne knjižnice modela za Unity. Na njega je postavljeno 5 senzora s jednakom maksimalnom udaljenošću koju mogu očitati. Kako bi se mogao kretati dodana je je komponenta krutog tijela (engl. rigid body component). U toj komponenti se definiraju masa vozila i sila otpora pa *Unity* može izračunati kretanje vozila. Također je dodana komponenta sudarač čija je funkcija ista kao i kod sudarača kod prepreka.



Slika 5.2 Vozilo i senzori

6. Grafičko korisničko sučelje

6.1. Glavni izbornik

Na glavnom izborniku su prikazana 3 gumba. Gumb „Učitaj mrežu“ omogućava da se iz datotečnog sustava učita datoteka u JSON formatu. Datoteka treba sadržavati definiciju neuronske mreže i polje čiji su elementi pozicije prepreka na terenu. Nakon odabira datoteke otvara se scena za testiranje postojećih mreža. Gumb „Učenje“ učitava scenu za učenje neuronske mreže, a gumb „Izađi“ služi za izlazak iz aplikacije.

Slika 6.1 Glavni izbornik

6.2. Scena za testiranje postojećih mreža

Scena za testiranje postojećih mreža prikazuje simulaciju autonomne vožnje uz pomoć mreže učitane iz datoteke. Nakon što simulacija završi, korisnik aplikacije istu može pokrenuti ponovo ili se može vratiti na glavni izbornik.

Slika 6.2 Scena za testiranje postojećih mreža

6.3. Scena za učenje neuronske mreže

Scena za učenje neuronske mreže simulira proces učenja autonomne vožnje. Ukoliko korisnik ustanovi da mreža daje zadovoljavajuće rezultate, potrebno je prvo privremeno zaustaviti scenu. Scena se ne zaustavlja u istom trenutku nego nakon što je simulirana vožnja svih jedinki iz trenutne populacije. Nakon što se scena zaustavi, korisnik može pohraniti neuronsku mrežu najboljeg vozila iz populacije. U svakom trenutku, korisnik se može vratiti na glavni izbornik.

Slika 6.3 Scena za učenje neuronske mreže

7. Zaključak

Prije strojnog učenja, za rješavanje problema bila je potrebno dobro ga definirati što je ponekad predstavljalo jednako težak problem. Ideja algoritama strojnog učenja je da se stroj sam nauči riješiti problem što je uspjelo riješiti mnoge probleme koji su ljudima zahtjevni. Stoga ne čudi da je strojno učenje danas jedno od najpopularnijih područja računarske znanosti.

U ovom radu je prikazano kako, koristeći tehnike strojnog učenja, riješiti problem autonomnog upravljanja vozilom. Implementirano je da se, prilikom vožnje, vozilom upravlja samo na temelju podataka dobivenih sa senzora. S obzirom na to, tehnike daju zadovoljavajuće rezultate. Da bi rezultati bili još bolji potrebno je razmotriti uvođenje kamera. Kamere bi raspoznavale objekte oko vozila što bi vožnju učinilo sigurnijom. Za terene sa slučajno generiranom visinskom mapom, trebalo bi dodati senzore koji su, u odnosu na smjer kretanja vozila, usmjereni prema dolje pod određenim kutom.

Literatura

[1] ≫Annual Global Crash Statistics≪, Association for Safe International Road Travel. Dostupno na: http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics (pristupljeno 3.6. 2018)

[2] ≫Herniated Disc Car Accident Statistics≪. Dostupno na: http://www.caraccidentherniateddisc.com/statistics/ (pristupljeno 3.6. 2018)

[3] ≫The Self-Driving Car Timeline – Predictions from the Top 11 Global Automakers≪, John Walker (29. 5. 2018.). Dostupno na: https://www.techemergence.com/self-driving-car-timeline-themselves-top-11-automakers/ (pristupljeno 3. 6. 2018.)

[4] Marko Čupić, *Umjetne neuronske mreže*, svibanj 2016.

[5] Marko Čupić, *Evolucijsko računarstvo*, lipanj 2016.

[6] Michael A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015

[7] Unity User Manual. Dostupno na: https://docs.unity3d.com/Manual/index.html

Strojno učenje u autonomnom upravljanju vozilom

Sažetak

U ovom radu se opisuje kako iskoristiti neuronsku mrežu za autonomno upravljanje vozilom. Objašnjeno je što su neuronske mreže i genetski algoritam i kako ih implementirati. Pokazano je kako genetski algoritam može poslužiti kao algoritam učenja neuronskih mreža. Rezultat rada je simulacija autonomnog upravljanja vozilom koja koristi tu mogućnost.

Ključne riječi: autonomna navigacija vozila, strojno učenje, neuronske mreže, genetski algoritam, Unity

Machine Learning in Autonomous Vehicle Navigation

Abstract

This thesis describes how to use neural networks for autonomous vehicle navigation. It is described what neural networks and a genetic algorithm are and how to implement them. It is shown how a genetic algorithm can be used as a neural network training algorithm. Resulting work is an autonomous vehicle navigation simulation that uses that opportunity.

Keywords: autonomous vehicle navigation, machine learning, neural networks, genetic algorithm, Unity