Duboko učenje pojačavanjem primijenjeno na robota pokretanog nogama

Piškorić, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Ivan Piškorić

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Izv.prof. dr. sc. Petar Ćurković, dipl. ing.

Student:

Ivan Piškorić

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru Dr. sc. Petru Ćurkoviću, dipl. ing. na pomoći i prenesenom znanju. Također zahvaljujem se ostalim profesorima, roditeljima, prijateljima i svima drugima koji su mi pomagali i podupirali me u toku pohađanja preddiplomskog studija Fakulteta strojarstva i brodogradnje.

Ivan Piškorić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Ivan Piškorić** JMBAG: **0035209042**

Naslov rada na hrvatskom jeziku: **Duboko učenje pojačavanjem primijenjeno na robota pokretanog nogama**

Naslov rada na engleskom jeziku: **Deep reinforcement learning applied on a legged mobile robot**

Opis zadatka:

Učenje pojačavanjem u kombinaciji s metodama dubokog učenja omogućuje tehničkim sustavima izvođenje iznimno složenih zadataka. Pojačavanje se odnosi na učenje onoga što je učinjeno dobro, odnosno preslikavanja scenarija u akcije s ciljem maksimizacije (pojačanja) numeričkog nagradnog signala.

U kontekstu bioničkog sustava mobilnoga robota pokretanog nogama, ulazni skup podataka čini prostor stanja robota (zakreti svih zglobova, brzine i ubrzanja svih članova, ulazi svih senzora) dok je izvršni prostor sačinjen od momenata na pojedinim zglobovima. Nagradni signal potrebno je oblikovati tako da se dvonožnom robotu omogući učenje efikasnog kretanja prostorom kontinuiranim preslikavanjem prostora stanja u izvršne momente.

U okviru rada potrebno je napraviti sljedeće:

- Upoznati se s MathWorks Reinforcement Learning i Deep Learning toolboxima
- Osmisliti model dvonožnog robota sa šest stupnjeva slobode gibanja i ugraditi ga u SimScape
- Oblikovati nagradni signal tako da robot nauči kretati se prostorom
- Analizirati doprinos svake komponente nagradnog signala učinkovitosti kretanja

U radu je potrebno navesti literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
30. 11. 2021.	**1. rok:** 24. 2. 2022. **2. rok (izvanredni):** 6. 7. 2022. **3. rok:** 22. 9. 2022.	**1. rok:** 28. 2. – 4. 3. 2022. **2. rok (izvanredni):** 8. 7. 2022. **3. rok:** 26. 9. – 30. 9. 2022.

Zadatak zadao:

Doc. dr. sc. Petar Ćurković

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
POPIS OZNAKA IV
SAŽETAK V
SUMMARY VI
1. UVOD 1
2. UMJETNA INTELIGENCIJA I STROJNO UČENJE 2
2.1. Umjetna inteligencija 2
2.1.1. Primjena umjetne inteligencije 2
2.2. Strojno učenje 3
2.2.1. Nadzirano učenje 3
2.2.2. Nenadzirano učenje 4
2.2.3. Učenje pojačavanjem 5
2.2.3.1. Duboko učenje pojačavanjem 6
3. STROJNO UČENJE U PROGRAMSKOM PAKETU MATLAB 8
3.1. Matlab 8
3.2. Reinforcement Learning toolbox 8
3.3. Deep Learning Toolbox 10
4. MODEL DVONOŽNOG ROBOTA 12
4.1. Izrada modela u programskom paketu CatiaV5 12
4.2. SimsCape Multibody 19
5. ANALIZA GOTOVOG MODELA HODAJUĆEG ROBOTA 21
5.1. Tijelo robota 21
5.2. Neuronska mreža 22
5.3. Zapažanja i uvjeti završetka simulacije 25
5.4. Nagradni signal 27
5.4.1. Što je nagradna funkcija? 27
5.4.2. Kako osmisliti nagradnu funkciju? 27
5.4.3. Nagradni signal dvonožnog hodajućeg robota 29
5.5. Pokretanje simulacije 31
5.6. Rezultati simulacije 32
6. ZAKLJUČAK 34
LITERATURA 35
PRILOZI 37

POPIS SLIKA

Slika 1.	Shema nadziranog učenja	4
Slika 2.	Shema nenadziranog učenja	5
Slika 3.	Shema učenja pojačavanjem	6
Slika 4.	Neuronska mreža	6
Slika 5.	Shema dubokog učenja pojačavanjem	7
Slika 6.	Eulerov dijagram obrađenih pojmova	7
Slika 7.	Stvaranje agenta unutar aplikacije	9
Slika 8.	Ponuda prethodno definiranih okruženja	9
Slika 9.	Postavke procesa učenja	10
Slika 10.	Grafičko praćenje procesa učenja	10
Slika 11.	Sučelje *Deep Network Designer* aplikacije	11
Slika 12.	Revolutni zglob	12
Slika 13.	Sketch	13
Slika 14.	Stvaranje novee ravnine	13
Slika 15.	Naredba *Multi-Sections Surface*	14
Slika 16.	Ispuna nacrtanih kontura	14
Slika 17.	Odabir modula *Part Design*	15
Slika 18.	Naredba *Close Surface*	15
Slika 19.	Dodavanje materijala naredbom *Pad*	16
Slika 20.	Zaobljenje oštrih rubova	16
Slika 21.	Model potkoljenice	17
Slika 22.	Model natkoljenice	17
Slika 23.	Model trupa	18
Slika 24.	Model stopala	18
Slika 25.	CAD model dvonožnog robota	19
Slika 26.	Stopalo robota	20
Slika 27.	Model dubokog učenja pojačavanjem hodajućeg robota u Simulinku	21
Slika 28.	Model desne noge robota	22
Slika 29.	Shema DDPG algoritma	23
Slika 30.	Actor neuronska mreža	24
Slika 31.	Critic neuronska mreža	25
Slika 32.	Izgled bloka Zapažanja	26
Slika 33.	Check if Done Box	27
Slika 34.	Ulazni i izlazni podaci okruženja	29
Slika 35.	Nagradni signal u obliku blokovskog dijagrama	31
Slika 36.	Simulacija kretanja robota	32
Slika 37.	Graf nagradnog signala	33
Slika 38.	Graf momenata zglobova	33

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
i	-	indeks određenog zgloba
R	-	vrijednost nagradne funkcije
t	-	Aktualni broj koraka ponavljanja
Tf	s	Vrijeme konačne simulacije
Ts	s	vrijeme uzorka
u	Nm	Pokretni moment zglobova
v_x	m/s	translacijska brzina u smjeru osi x
y	m	translacija robota u negativnom ili pozitivnom smjeru osi y u odnosu na pravac osi x
z	m	vertikalna translacija

SAŽETAK

Hodajući robot svojom definicijom spaja hodanje kao izrazito kompleksni naučeni neuro-fizički proces i pojam robota koji kao strojarsko dostignuće i alat svakim danom postaje sve potrebniji i raširenij u gotovo svim poljima čovjekova djelovanja. Odavno u robotici postoji težnja za izradom hodajućeg robota, a uz pojavu umjetne inteligencije i strojnog učenja to postaje moguće. U ovom radu predstavit će se dijelovi procesa dubokog učenja pojačavanjem hodajućeg robota te analiza projekta.

Ključne riječi: hodanje, robotika, umjetna inteligencija, strojno učenje

SUMMARY

Walking robot by its definititon connects very complex and learned neuro-physical process of walking and a robot, engineering achievement and tool which is becoming more and more required nad spreaded in every field of human's acting. For a long time there is a desire to built a walking robot in robotics. With artificial intelligence and machine learning this has become very possible. In this paper, some parts of deep reinforcement learning of a walking robot will be presented.

Key words: walking, robotics, artificial intelligence, machine learning

1. UVOD

Robotika je interdisciplinarno znanstveno područje koje značajno pridonosi razvoju društva. Važan smjer dugotrajnog istraživanja u području robotike zadnjih godina su mobilni roboti pokretani nogama, odnosno njihovi problemi, uključujući hardware, analizu stabilnosti i algoritme upravljanja. Kretanje nogama u prostoru zahtijeva veći broj stupnjeva slobode za prenošenje sile u više različitih smjerova, što omogućuje prilagodbu i manevriranje na teškom i neravnom terenu. To je glavna prednost mehanizma s nogama nad mehanizmom s kotačima, no dodavanje stupnjeva slobode povlači i veću mehaničku složenost, veću težinu robota te veću potrošnju. Dvonožni roboti najpopularnija su vrsta mobilnih robota pokretanih nogama koja u svom djelovanju tehnički reproducira kretanje čovjeka. Kod njih je potrebno istaknuti problem stabilnosti jer robot mora održavati ravnotežu i kada samo stoji kako se ne bi prevrnuo.

Mobilni roboti pokretani nogama koriste metode umjetne inteligencije kako bi mogli kreativno rješavati probleme. Strojno učenje, kao podskup umjetne inteligencije, omogućuje učenje iz iskustva bez prethodnog programiranja. Duboko učenje s pojačavanjem popularna je vrsta strojnog učenja koja tehničkim sustavima omogućuje izvođenje iznimno složenih zadataka. Kod njega se učenje odvija autonomno, kroz interakciju s okolišem, tako što agent (robot) uči preslikati scenarije u akcije kako bi dobio maksimalni numerički nagradni signal. Mogućnost primjene dubokog učenja pojačavanjem na robota pokretanog dvjema nogama fokus je ovog završnog rada.

U narednim poglavljima predstavljeni su osnovni pojmovi vezani za strojno učenje, programski paketi *Matlab* i *CatiaV5* te je prikazano modelirane nekih dijelova kompleksnog sustava hodajućeg dvonožnog robota

2. UMJETNA INTELIGENCIJA I STROJNO UČENJE

2.1. Umjetna inteligencija

Umjetna inteligencija (eng. *Artificial Intelligence – AI*) simulacija je ljudske inteligencije, a provode ju računala. Računala se treniraju to jest uče rješavati probleme na što bolji i učinkovitiji način. Cilj procesa je da računalo počne razmišljati kao ljudsko biće. Računalo oponaša ljudske radnje, ali isto tako i uči koji je najracionalniji put da se dođe do željenog cilja. Učenje se ostvaruje putem zadanih podataka koje računalo isto tako korigira i osvježava na putu do cilja.

U kratkom roku, razvoj umjetne inteligencije je brzo napredovao i mnogi vjeruju kako će se uskoro razviti sustavi, alati i algoritmi kojima će računala nadmašiti ljudske radnje. Iako je to jedan od baznih ciljeva umjetne inteligencije, mnogi se pribojavaju da će računala postati opasnost za ljudsku vrstu.

2.1.1. *Primjena umjetne inteligencije*

Područja primjene umjetne inteligencije su sve šira i šira, a neka od najvažnijih su medicina, robotika, promet, prehrambena industrija, informatika, ekonomija i gospodarstvo. U sljedećem paragrafu navest će se neke primjene.

Medicina je jedna od vodećih znanosti u primjeni umjetne inteligencije. Ona se koristi za detekciju bolesti (npr. prepoznavanje stanica karcinoma). Računalo koristi ogromnu količinu podataka (simptomi, stanje pacijenta kroz izmjerene fizičke veličine, povijesni podaci itd.). Umjetna inteligencija je vrlo učinkovita jer može prepoznati puno manje promjene u tijelu nego čovjek. Otkrivanje novih lijekova i pomoć pri operaciji također su važne primjene u ovom području.

Koristeći svoj mobilni uređaj, svakodnevno smo objekt proučavanja i korisnik umjetne inteligencije, a da toga nismo niti svjesni. Sugeriranje sadržaja koji nam se nudi na internetskim stranicama, predviđanje vremena, prijevod sa stranog jezika, put do odredišta koristeći navigaciju i prepoznavanje lica za otključavanje telefona su samo neki od „proizvoda“ algoritama kojima se računalo uči.

Može se reći da se AI koristi svugdje gdje je uključen rad robota. Od pametnog usisivača do robota u operacijskim salama, svi su naučeni dolaziti do zadanog cilja izbjegavajući prepreke na koje nailaze. Pri tome im pomažu razni senzori koji slanjem podataka komuniciraju s

računalom te ono na temelju tih informacija osvježava sliku o svojoj okolini koja se nerijetko mijenja.

2.2. **Strojno učenje**

Strojno učenje (eng. *Machine Learning – ML*) jedna je od osnovnih tehnika za dubinsku analizu podataka i čini podskupinu umjetne inteligencije. Ono tvori osnovnu razliku između standardne analize podataka i dubinske analize podataka kojom se koristi umjetna inteligencija. Dubinska analiza podataka koristi podatke kako bi otkrila nova učinkovita znanja putem povezivanja velikog broja dobivenih empirijskih podataka. Strojnim učenjem izgrađujemo model čiju učinkovitost provjeravamo na novim podacima koji nisu prethodno dani računalu.

Na primjer, cilj učenja je da model na slici prepozna psa. Računalu se daje veliki broj podataka u obliku slika (vizualizacija) na kojima se nalazi pas te opisa osnovnih fizičkih karakteristika psa (podaci). Nakon izgradnje, model se testira prikazom slika koje još nije vidio [1-3].

Primjena strojnog učenja dobrodošla je u rješavanju svakog problema za koji standardno programiranje nije dostatno. Neke primjene su raspoznavanje uzoraka, raspoznavanje govora, predviđanje trendova i bio informatika.

Postoje tri osnovna tipa strojnog učenja.

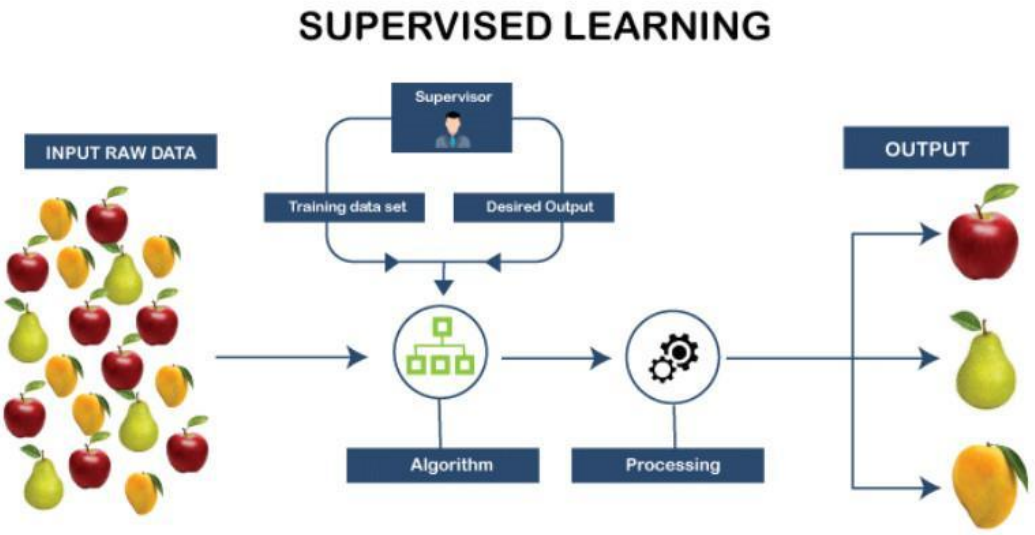
1) Nadzirano učenje (eng. *Supervised learning*)
2) Nenadzirano učenje (eng. *Unsupervised learning*)
3) Učenje pojačavanjem (eng. *Reinforcement learning*)

2.2.1. ***Nadzirano učenje***

Kod ove vrste strojnog učenja algoritam generira određenu funkciju koja povezuje ulazne i izlazne podatke. Model proučava određeni broj označenih setova izlaznih i ulaznih podataka te pomoću toga uči spomenutu funkciju kako bi predvidio izlazni rezultat „testnog“ ulaznog podatka s kojim se do sada nije susreo. Nadzirano učenje smatra se pouzdanijom vrstom učenja u odnosu na nenadzirano, ali zahtjeva intervenciju čovjeka u obliku preciznog označavanja setova podataka.

Nadzirano učenje često se koristi kada govorimo o problemu predviđanja baziranog na poznatim označenim podacima.

Shema nadziranog učenja [Slika 1]



[4]

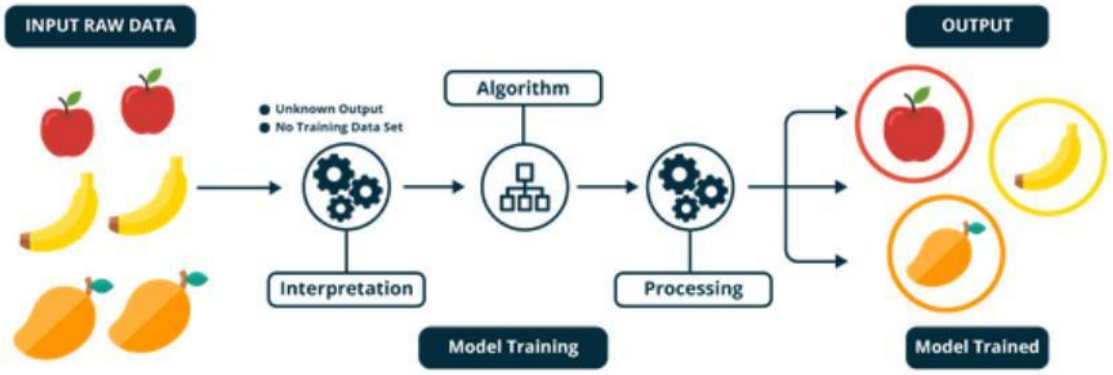
Slika 1. Shema nadziranog učenja

2.2.2. *Nenadzirano učenje*

Osnovna razlika nenadziranog učenja u odnosu na nadzirano je u tome što ovdje dani setovi ulaznih i izlaznih podataka nisu obilježeni. Model sam mora naučiti svojstvenu strukturu neoznačenih podataka bez ljudske intervencije. Model generira uzorke po kojima će grupirati podatke. Cilj ovog tipa učenja nije da računalo prepoznaje ulazni podatak i dodjeljuje ga poznatom označenom izlaznom podatku kao kod nadziranog učenja, nego da grupira različite objekte u skupine vlastitim algoritmom. Radi se o jednostavnijem tipu strojnog učenja, ali je ujedno i nepouzdaniji od nadziranog.

Nenadzirano učenje često se koristi za probleme grupiranja i detekcije anomalija. Grupiranje se odvija na temelju sličnosti podataka koje model već poznaje i novih podataka koje treba procesuirati pomoću vlastitog algoritma. Moguća je i hibridizacija s drugim područjima umjetne inteligencije, poput evolucijskih algoritama, algoritmima roj čestica i sl. [4,7, 8, 10].

Ovakav pristup omogućuje robusno učenje robota kombinirajući metode optimiranja s metodama strojnoga učenja. Primjeri primjene su razni, od planiranja kretanja robota, planiranja kretanja robota koji surađuju, dijeleći radnu okolinu i predstavljajući prepreku jedan drugome, do planiranja i optimiranja kretanja robotske ruke u prostoru.



[5]

Slika 2. Shema nenadziranog učenja

2.2.3. *Učenje pojačavanjem*

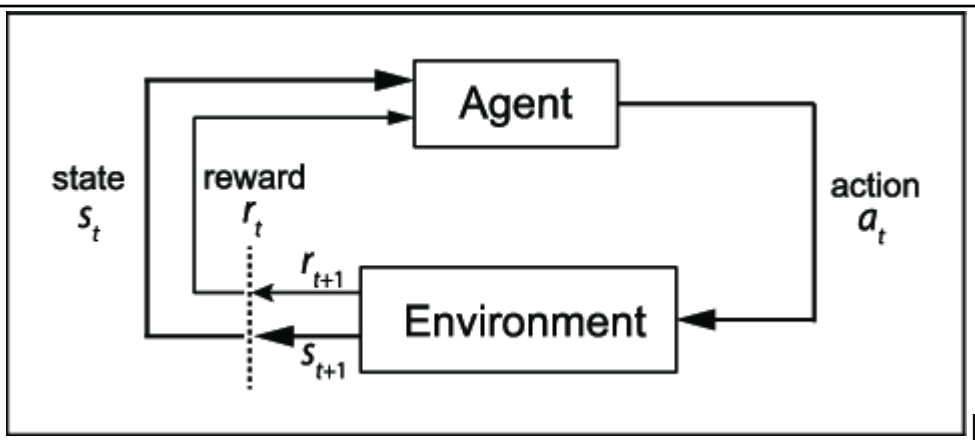
Učenje pojačavanjem je tip strojnog učenja koji je primijenjen kod modela hodajućeg robota koji je objekt promatranja ovog rada.

Učenje pojačavanjem koristi takozvanog agenta koji je zapravo dio software-a i njegovo okruženje (eng. *Environment*) kako bi model izvršio potrebne akcije koje dovode do cilja i prikupio što više takozvanih nagrada kojima se usmjerava njegovo djelovanje.

Agent istražuje okoliš te je s njim u međudjelovanju i od njega uči kako bi u sljedećoj iteraciji akcije bolje djelovao. Svojim akcijama agent može mijenjati stanje svog okruženja, a okruženje mu dodjeljuje „nagradu“ ili „kaznu“ za poduzetu radnju. U slučaju dvonožnog hodajućeg robota okruženje je i sami *hardware* robota to jest njegove fizičke komponente.

Politika (eng. *Policy*) djelovanja agenta mijenja se s obzirom na poduzete akcije, dobivene nagrade i promjene okruženja. Cilj je maksimizirati nagradni signal.

Rad u dinamičkom okruženju jedna je od glavnih razlika u odnosu na prethodna dva tipa strojnog učenja. Može se reći da učenje pojačavanjem koristi metodu pokušaja i pogrešaka kako bi predvidjelo rezultate i unaprijedilo sljedeći pokušaj.

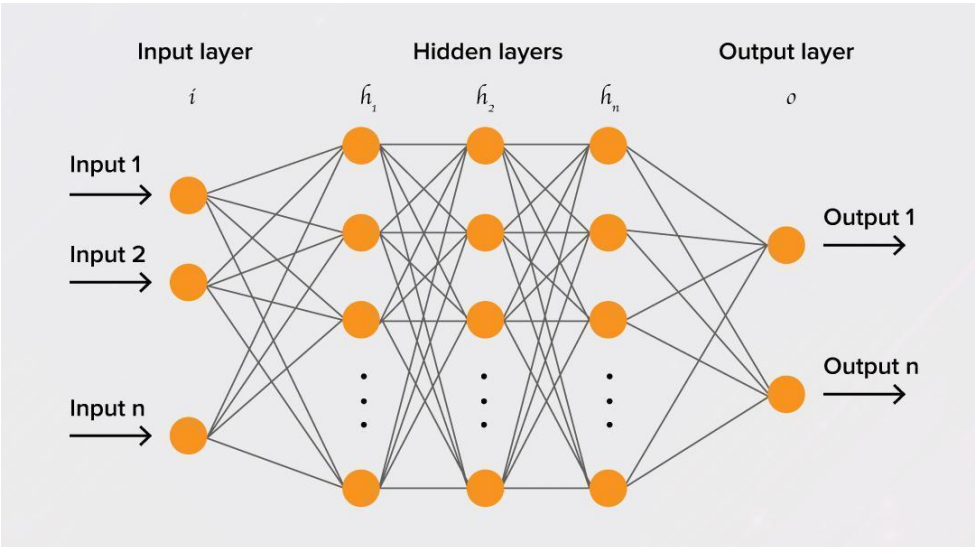


[9]

Slika 3. Shema učenja pojačavanjem

2.2.3.1. *Duboko učenje pojačavanjem*

Duboko učenje pojačavanjem (eng. *Deep Reinforcement Learning*) je vrsta strojnog učenja koja stupa na snagu kada donošenje odluka postaje preteško za učenje pojačavanjem. „Duboko“ u nazivu označava upotrebu neuronske mreže. Ulazni podaci šalju se u prethodno oblikovanu neuronsku mrežu. U neuronskoj mreži [Slika 4.] ulazni podaci prolaze kroz nelinearne transformacije čime problem postaje apstraktniji.

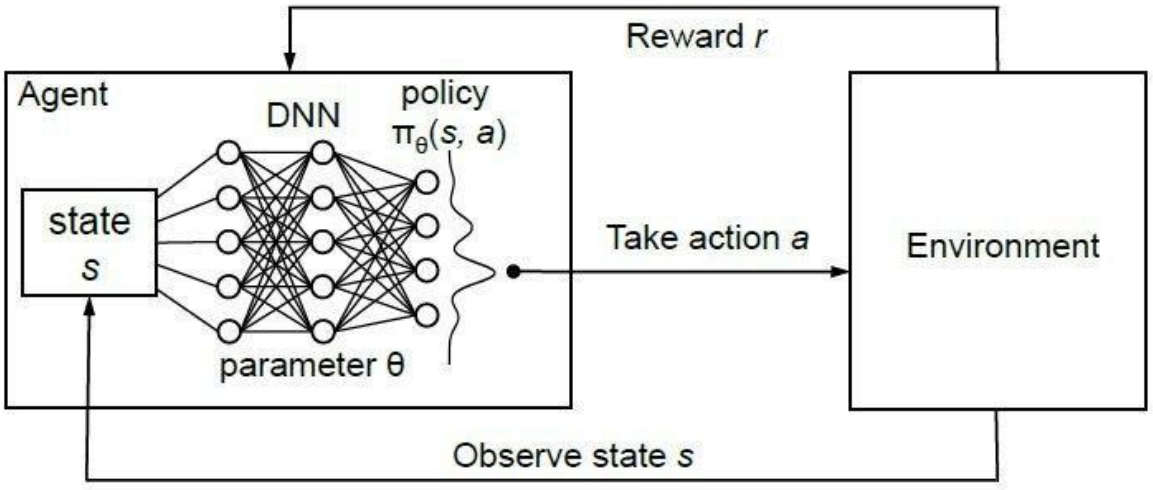


[18]

Slika 4. Neuronska mreža

Neuronska mreža procjenjuje svako novo stanje okruženja te se time eliminira potreba za spremanjem i učenjem svake moguće kombinacije ulaznih podataka (stanja okruženja) i izlaznih podataka (poduzete akcije). Može se reći da ova metoda zamjenjuje tablični zapis

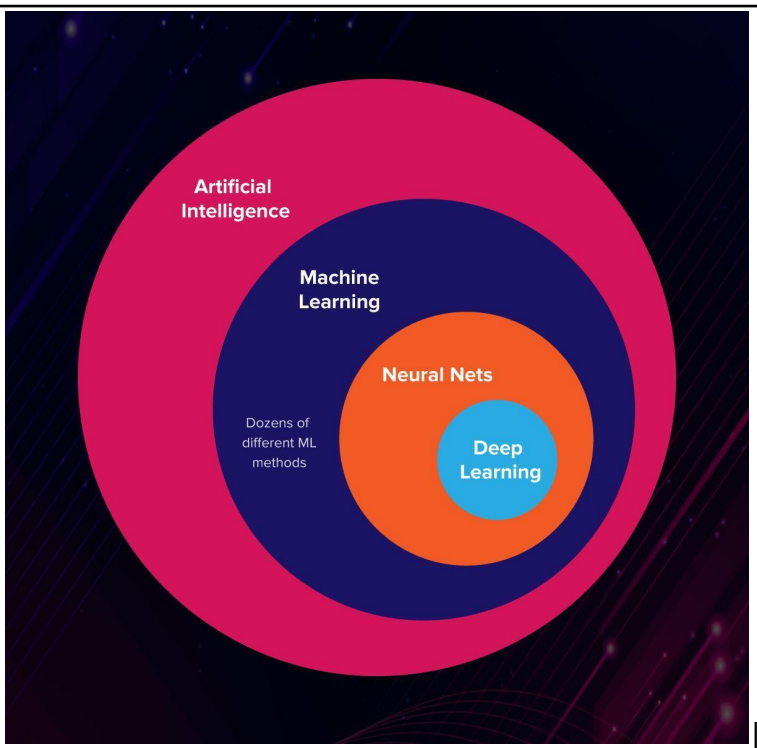
setova podataka sa funkcijom aproksimacije. Dodatna prednost neuronske mreže je u tome što agent može generalizirati postupak djelovanja za neko stanje okruženja s kojim do tada uopće nije bio upoznat. Prikazana je shema dubokog učenja pojačavanjem [Slika 5.] te se odmah iz usporedbe dvaju shema prepoznaje razlika između ove dvije vrste strojnog učenja koje se uspješno upotpunjuju.



[16]

Slika 5. Shema dubokog učenja pojačavanjem

Veza do sada obrađenih pojmova jednostavno je prikazana je na Slici 6. pomoću Eulerovog dijagrama.



[11]

Slika 6. Eulerov dijagram obrađenih pojmova

3. STROJNO UČENJE U PROGRAMSKOM PAKETU MATLAB

3.1. **Matlab**

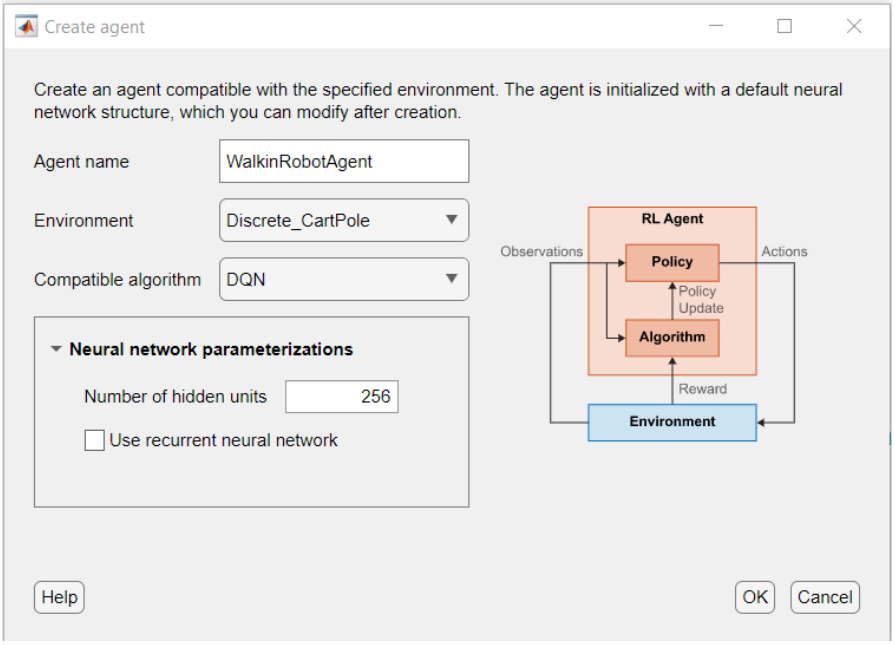
Matlab se može smatrati programskim jezikom, ali zapravo je puno više od toga. Radi se o platformi za programiranje namijenjenoj znanstvenicima i inženjerima u svrhu programiranja, modeliranje i analize raznih sustava. Omogućuje vizualizaciju sustava i rezultata simulacija, implementaciju algoritama, stvaranje sučelja, praćenje sustava, korištenje metoda umjetne inteligencije i strojnog učenja i mnoge druge aktivnosti koje svakodnevno pomažu velikom broju korisnika.

U širokoj paleti ugrađenih „dodataka“ nalaze se i 2 takozvana *toolboxa* koja će biti opisana u ovom radu.

3.2. **Reinforcement Learning toolbox**

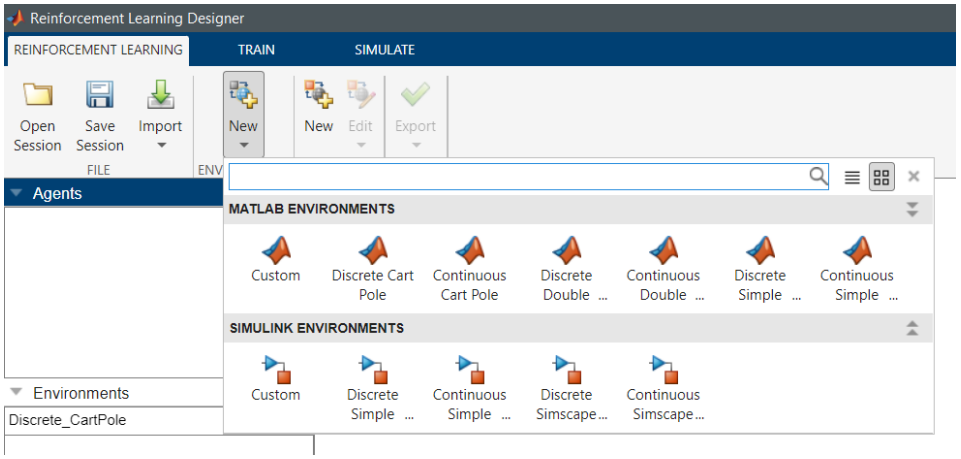
U prethodnom poglavlju opisan je pojam „Učenje pojačavanjem“, no sada će se prikazati kako pomoću spomenutog programskog alata izgraditi sustav učenja. Ovaj *toolbox* sadrži funkcije, blokove i algoritme za primjenu učenja pojačavanjem. Omogućuje se rad kroz sve korake rada učenja pojačavanjem uključujući kreiranje okoline i agenta do treniranja algoritma koji mijenja svoju politiku rada.

Definiranje agenta provodi se biranje ponuđenih algoritama (DQN, DDPG, TD3, SAC, SARSA, Q-Learning itd.) ili implementacijom vlastitog prethodno definiranog algoritma. Kreiranje agenta može se provesti interaktivno putem *Reinforcement Learning Designera* [Slika 7] ili definiranjem programskog koda. Za slučaj dubokog učenja aplikacija automatski generira neuronsku mrežu. Neuronska mreža može se također konstruirati programskim kodom ili putem ugrađenih funkcija.



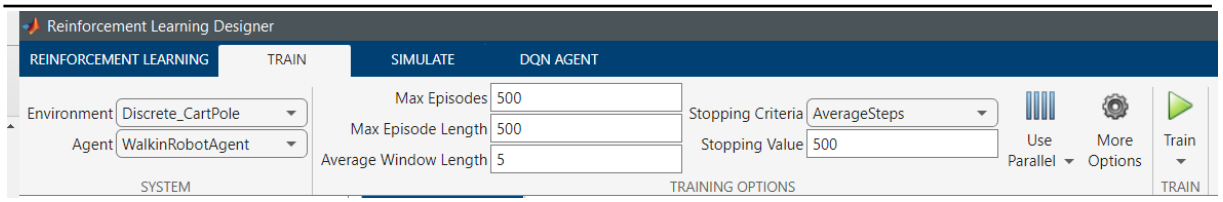
Slika 7. Stvaranje agenta unutar aplikacije

Okolina se može konstruirati u *Matlab*-u ili *Simulink*-u, a potrebno je stvoriti oko istu s dinamičkim svojstvima, zapažanjima i nagradnim signalom. Program nudi nekoliko već definiranih okolina koje se mogu koristiti u modelu učenja.



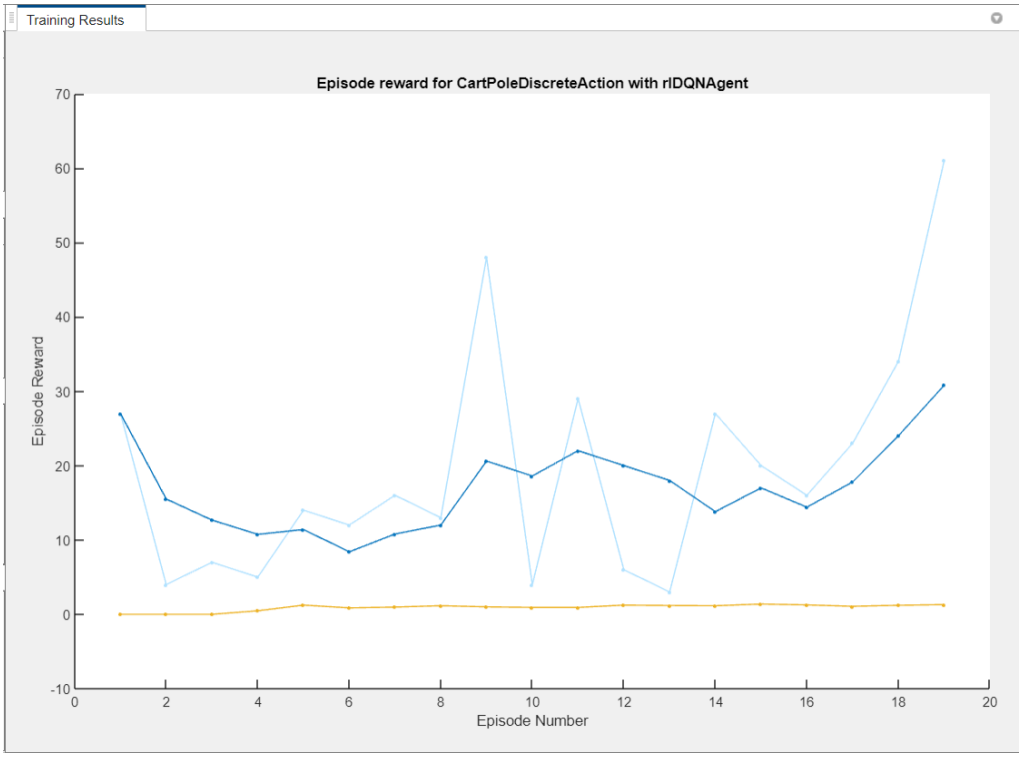
Slika 8. Ponuda prethodno definiranih okruženja

Prije početka učenja potrebno je još samo definirati opcije učenja to jest treniranja kao što su broj ponavljanja, trajanje jednog ponavljanja i kriterij zaustavljanja.[Slika 9.] Učenje se također može provoditi i putem sučelja aplikacije i programskim kodom. *Matlab*-ov paralelni server i *Parallel computing Toolbox* također omogućava ubrzanje učenja paralelnim učenjem.



Slika 9. Postavke procesa učenja

Tokom učenja *Episode Manager* pruža vizualno i grafičko praćenje učenja [Slika 10.]. Nakon završetka učenja može se pokrenuti animacija i izvršiti potrebne promjene.



Slika 10. Grafičko praćenje procesa učenja

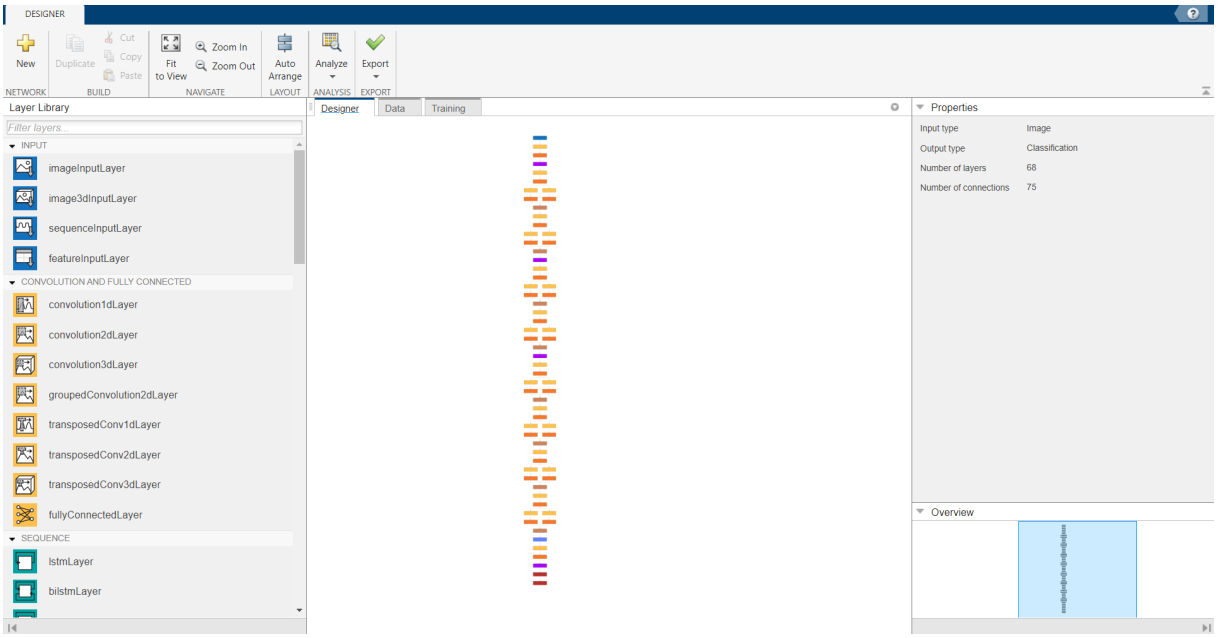
3.3. Deep Learning Toolbox

Pojam dubokog učenja pojačavanjem također je opisan u prethodnom poglavlju. Za tu vrstu strojnog učenja *Matlab* nudi posebnu aplikaciju sa algoritmima i alatima za kreiranje, učenje i analizu dubokih neuronskih mreža. Alat dolazi s brojnim već izgrađenim primjerima koji se mogu koristiti.

Pomoću *Experiment Manager* sučelja omogućeno je upravljanje više eksperimenata dubokog učenja. Koristi se za praćenje parametara, analizu rezultata, a moguća je i grafička usporedba s algoritmima iz drugih eksperimenata.

Uz korištenje provjerenih modela neuronskih mreža, omogućeno je i slaganje vlastite mreže uporabom ponuđenih slojeva. U tom slučaju korisnik ima mogućnost modificirati parametre slojeva. Isto tako, mogu se uvesti mreže kreirane u aplikacijama s kojima aplikacija ima podržanu vezu razmjene podataka

Nakon konstruiranja mreže, potreban je unos podataka za učenje koji se mogu naknadno pregledati. Ostaje još samo postaviti opcije učenja i treniranje unutar aplikacije može započeti. Nakon treninga omogućen je izvoz podataka o mreži i rezultatima. Podaci o mreži mogu se izvesti i u obliku *Matlab* programskog koda [5 - 6].

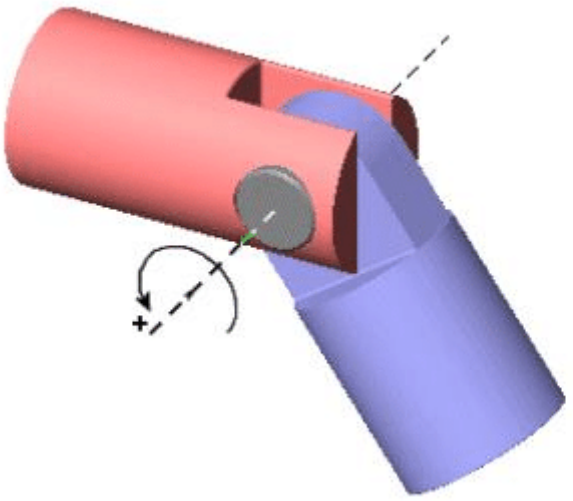


Slika 11. Sučelje *Deep Network Designer* aplikacije

4. MODEL DVONOŽNOG ROBOTA

U ovom poglavlju prikazana je izrada modela dvonožnog robota sa šest stupnjeva slobode gibanja.

Stupnjevi slobode gibanja predstavljaju broj neovisnih gibanja u tijelu. U ovom slučaju gibanja su relativna tj. radi se o gibanjima između dijelova modela robota. Dijelovi robota sklopljeni su sa šest revolutnih zglobova [Slika 12.]. Revolutni zglob omogućava isključivo rotaciju oko jedne osi. Vrlo jednostavnim računom dolazi se do rezultata da je potrebno kreirati šest zglobova kako bi se modelirao robot sa šest stupnjeva slobode gibanja.



[12]

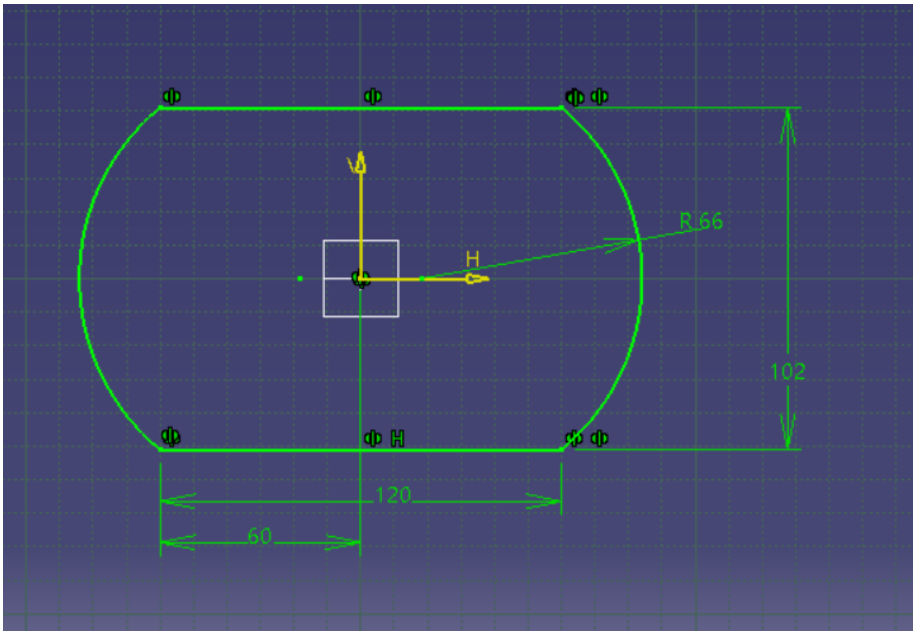
Slika 12. Revolutni zglob

Model robota prikazan u ovom poglavlju izrađen je u programskom paketu CatiaV5, a sastoji se od trupa, dvije natkoljenice, dvije potkoljenice i dva stopala.

4.1. Izrada modela u programskom paketu CatiaV5

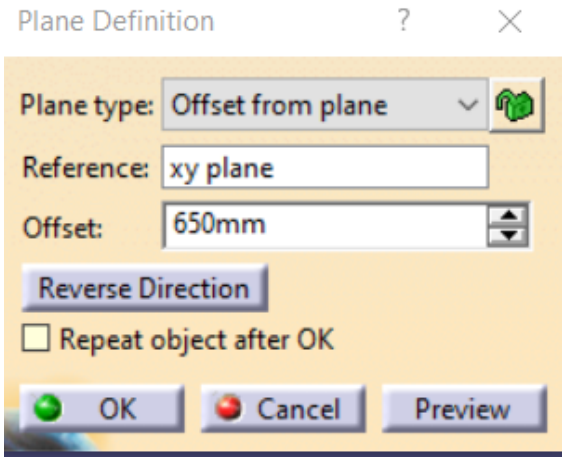
Prikazan je detaljan proces oblikovanje jedne potkoljenice robota.

U početnom izborniku odabire se modul *Generative Shape Design* u skupini *Shape*. U prostoru odabire se željena ravnina te klikom na naredbu *Sketch* počinje crtanje. Koriste se naredbe *Line*, *Three point arc* i *Mirror* kako bi se došlo do konture crteža, a naredbama *Constraint* i *Constraints defined in Dialog Box* dimenzionira se željeni profil [Slika 13.]



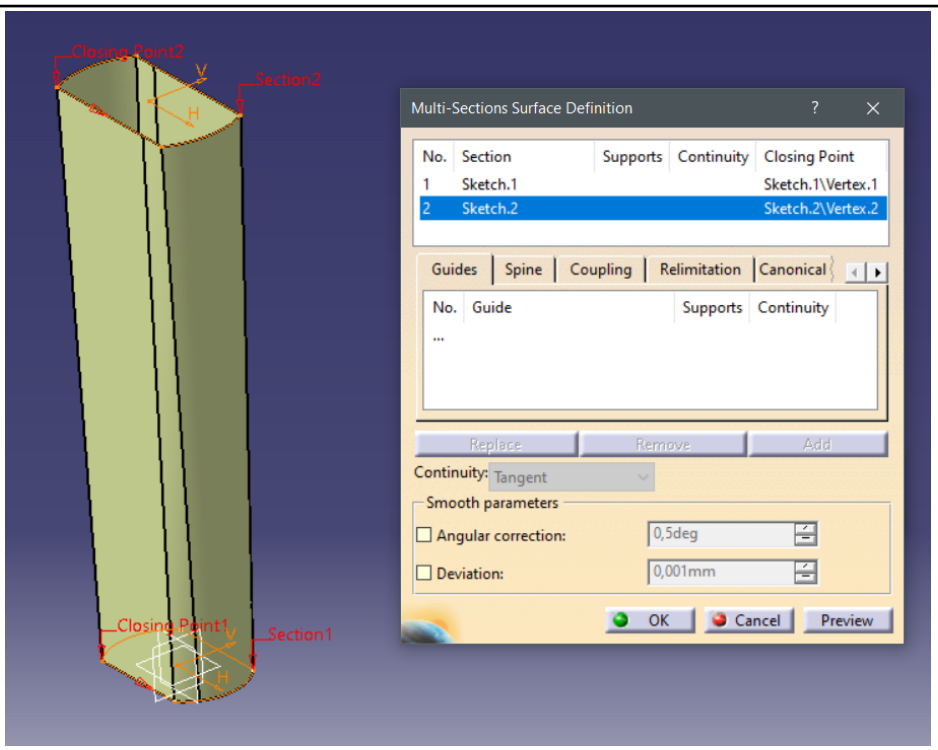
Slika 13. Sketch

Naredbom *Exit workbench* izlazi se iz *Sketch*-a. Odabire se naredba *Plane* kojom se generira nova ravnina u kojoj će se stvarati novi *Sketch* sličnog izgleda kao i prvi. [Slika 14.]



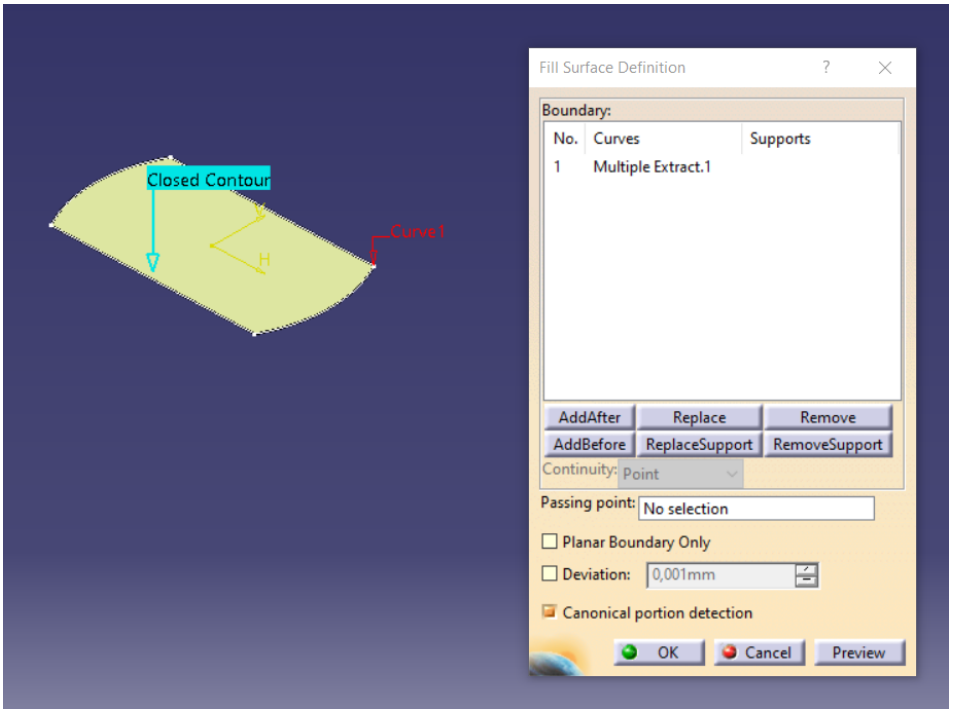
Slika 14. Stvaranje novee ravnine

Spajanjem kontura dviju nastalih skica dobit će se površina koja je zapravo plašt željenog modela potkoljenice. U tu svrhu koristi se naredba *Multi-Sections Surface*. [Slika 15.]



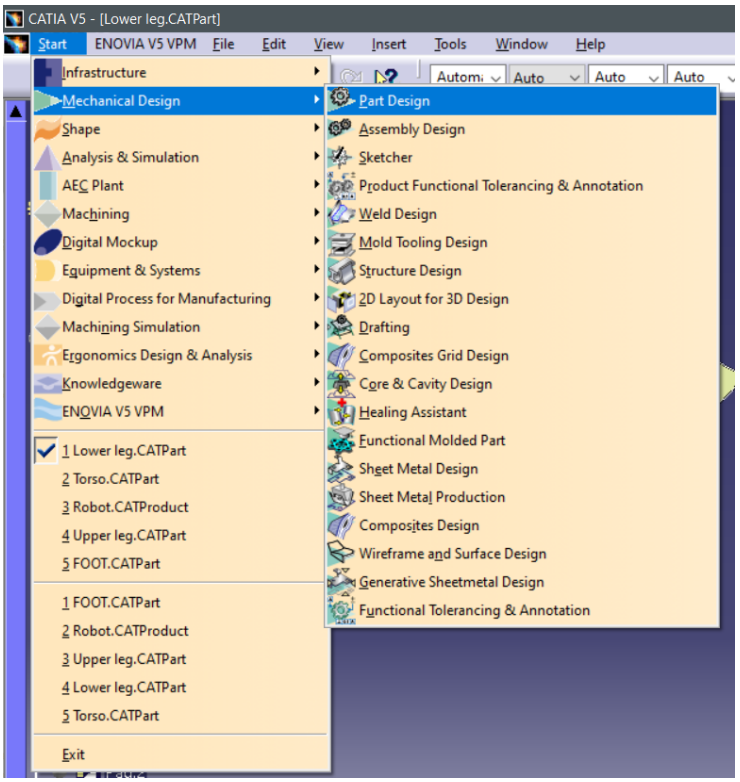
Slika 15. Naredba *Multi-Sections Surface*

Kako bi se dobile dvije površine koje će zatvoriti plašt modela koriste se funkcije *Fill* i *Multiple Extract Definition*. Time su se ispunile konture nacrtane na početku. [Slika 16.]



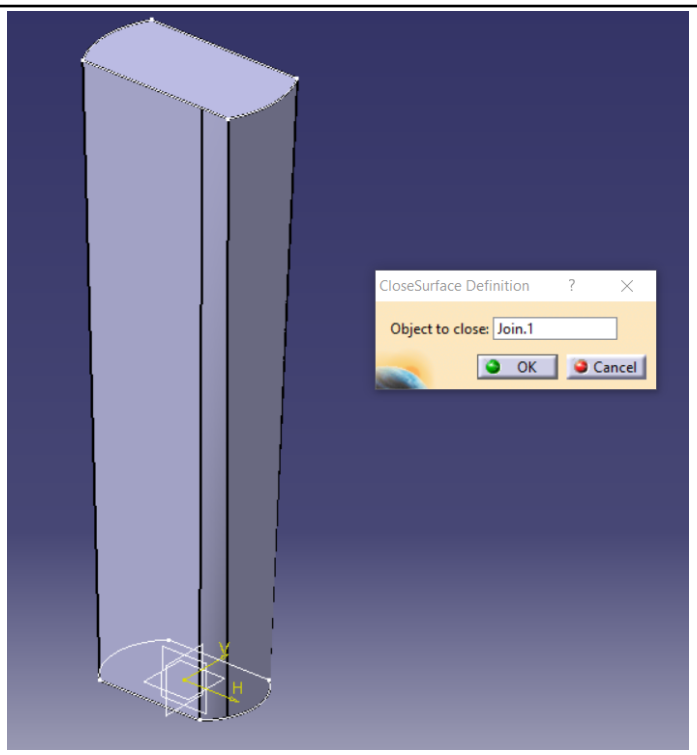
Slika 16. Ispuna nacrtanih kontura

Naredbom *Join* spojit će se tri dobivene površine. Nakon spajanja prelazi se u modul *Part Design* koji se nalazi u skupini *Mechanical Design*. [Slika 17.]



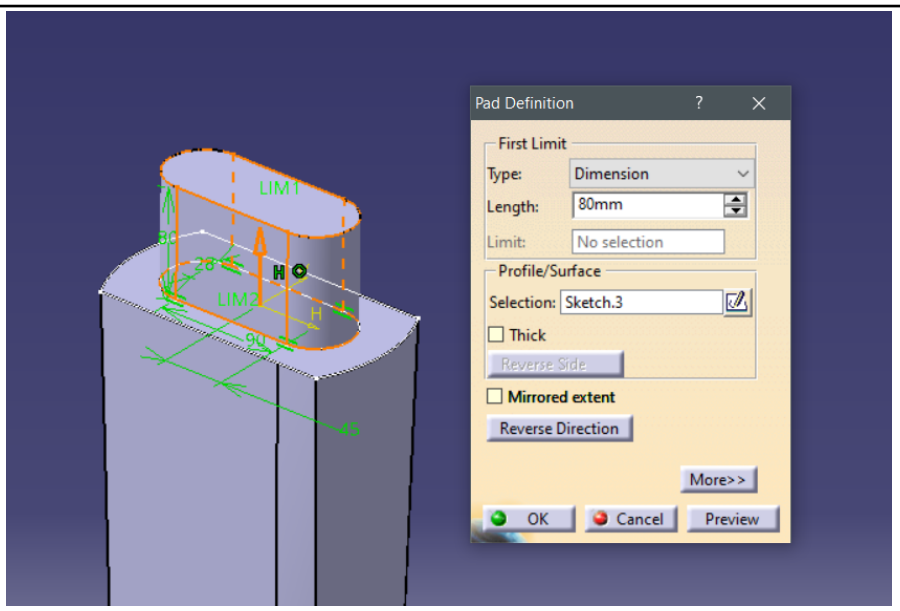
Slika 17. Odabir modula *Part Design*

Kako bi se nastavilo raditi na oblikovanju modela, stvorenu grupu površina potrebno je „zatvoriti“ što znači da će se od njih stvoriti volumen to jest dobit će se popunjeni model oblikovan tim površinama. Tome služi naredba *Close Surface*. [Slika 18.]



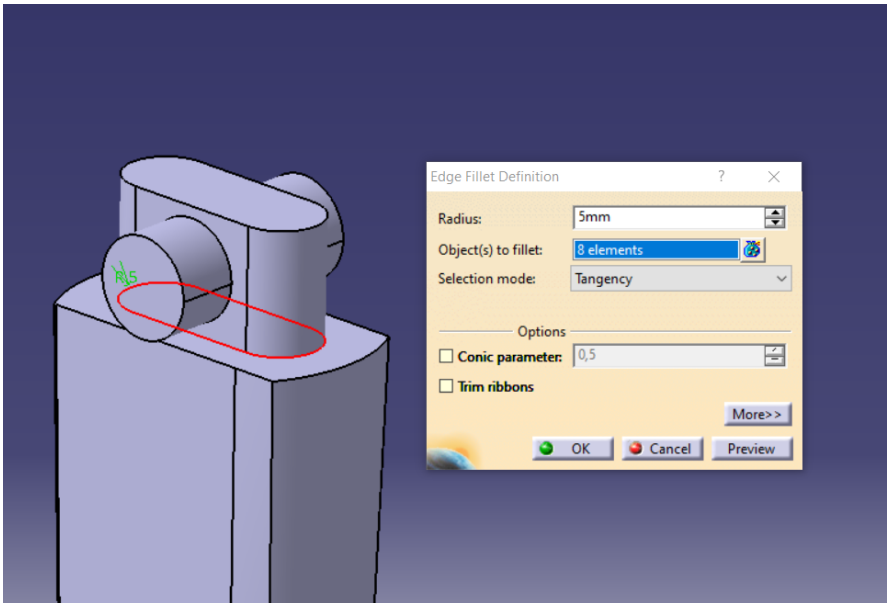
Slika 18. Naredba *Close Surface*

Tijelo potkoljenice time je oblikovano. Kako bi se Potkoljenica mogla spojiti s ostalim dijelovima robota. Potrebno je oblikovanje prihvatnih dijelova (osovinica i provrta). U tu svrhu opet se koristi naredba *Sketch* te naredbe *Pad* i *Pocket*. Naredba *Pad* dodaje materijal u obliku nacrtanog profila, dok naredba *Pocket* materijal oduzima. [Slika 19.] Kod definiranja naredbi, potrebno je odabrati dobro oblikovan i zatvoren profil to jest *Sketch*, a potom odrediti duljinu dodavanja ili oduzimanja materijala.



Slika 19. Dodavanje materijala naredbom *Pad*

Na samome kraju vrši se zaobljivanje oštrih bridova naredbom *Edge Fillet*. [Slika 20.] Za skošenja bridova kod provrta koristi se naredba *Chamfer*. Opisanim postupcima dobije se Potkoljenica hodajućeg robota. [Slika 21.]



Slika 20. Zaobljenje oštrih rubova

Slika 21. Model potkoljenice

Na sljedećih par slika prikazani su gotovi modlei ostalih dijelova robota. Svaki od njih dobiven je postupcima sličnim onima nabrojanima u postupku izrade Potkoljenice. Nakon što su izrađeni svi dijelovi odabire se modul *Assembly design* koji generira novi sklop. Prvi korak je dodati sve prethodno modelirane dijelove u novonastali sklop. Dijelovi se mogu dodati svi odjednom ili jedan po jedan.

Slika 22. Model natkoljenice

Slika 23. Model trupa

Slika 24. Model stopala

Kako bi se dobio smisleni sklop robota [Slika 25], potrebno je dijelove pomaknuti u prostoru i postaviti određene poveznice i ograničenja između njih. Potrebne su naredbe *Fix*, *Coincedence*, *Offset* i *Contact*. Tim naredbama odredilo se u kojim osima će se dijelovi spajati u cjelinu te koliko su neke njihove stranice udaljene.

Slika 25. CAD model dvonožnog robota

Naredbom *Fix* jednom dijelu se ograničava kretanje u prostoru, no to je puno bitnije kada je riječ o izradi mehanizma. Za to se koristi modul *DMU Kinematics*. Unutar modula dijelovi se spajaju određenim mehaničkim vezama (eng. *Joint*). Također je potrebno odrediti koje veze su pokretačke ili aktivne to jest one kojima kontrolima kinematički sklop, a koje su pasivne. Ako cijelu sustav ima kinematičkog smisla tj. ako broj stupnjeva slobode gibanja odgovara broju naredbi aktivnih veza, program će nam dati do znanja da je moguće simulirati kretanje.

Kako je opisano u prethodnom tekstu, koristit ćemo se revolutnim zglobovima koji omogućuju relativno gibanje između elemenata u obliku rotacije oko osi zgloba. Prilikom definiranja veze potrebno je izabrati zajedničku os rotacije te udaljenost između dvije paralelne ravnine elemenata.

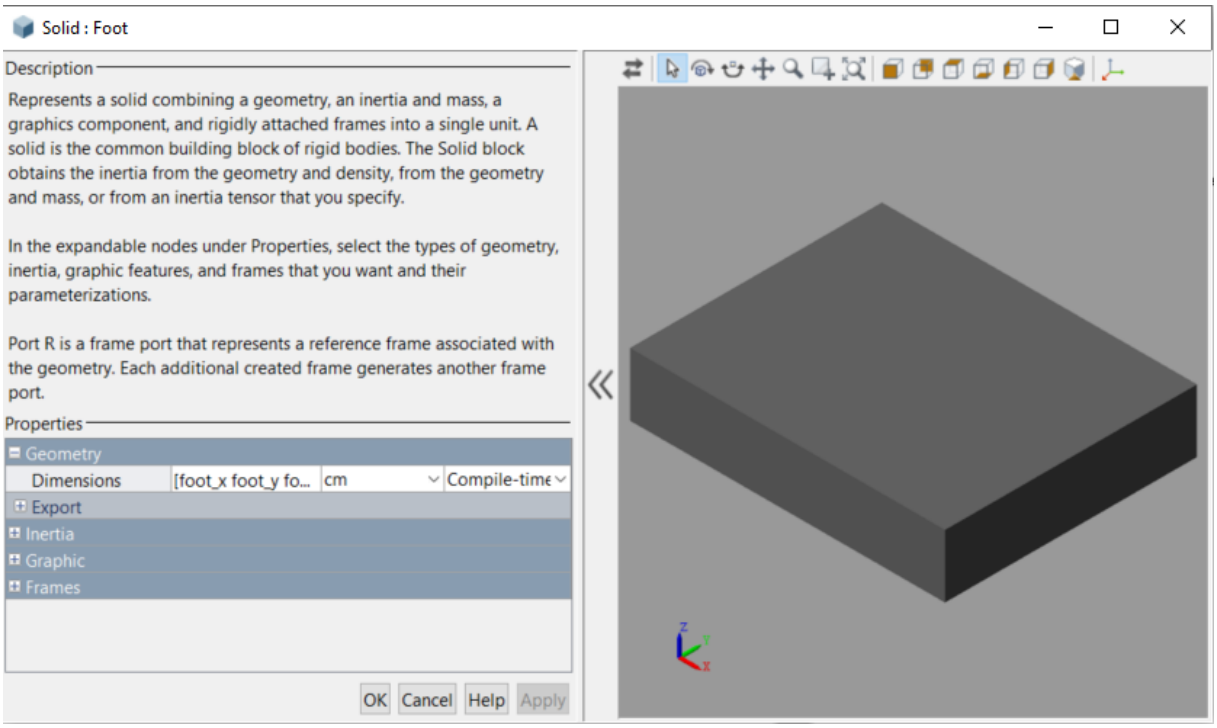
4.2. **SimsCape Multibody**

Matlab kao programski alat u svojem paketu dolazi sa *Simulink* programskim sučeljem. Na *Simulink* platformi pomoću blok dijagrama omogućeno simuliranje sustava. Sustave je moguće oblikovati, razvijati i analizirati kako bismo spoznali valjanost i efikasnost sustava bez izgradnje fizičkih komponenti. Time se štedi mnogo vremena i novca. u sklopu platforme dano je mnogo prethodno definiranih blokova kojima gradimo sustav. Blokovi mogu predstavljati funkcije, CAD modele, fizičke veze, sile, matematičke operacije svih vrsta, vrijeme itd. Omogućena je izgradnja i analiza sustava bez prethodnog programiranja, ali i ugradnja definiranog programskog koda za rad sustava.

Simulink primjenu nalazi u autoindustriji, svemirskoj industriji, kontroli sustava, biotehnologiji, farmaciji, medicini i mnogim drugim područjima.

Simscape je dio *Simulink-a* koji omogućava brzo modeliranje fizičkih sustava temeljeno na njihovim fizičkim vezama. U svrhu konstruiranja modela robota koristit će se njegovim modulom *SimsCape Multibody*. Odabrani modul pruža okolinu za analizu i simulaciju sustava sastavljenih od više 3D modela. Blokovi kojime se koristi u radu u *SimsCape Multibody* sučelju predstavljaju tijela, kinetičke veze, sile, momente, fizička ograničenja, senzore te koordinatne sustave.

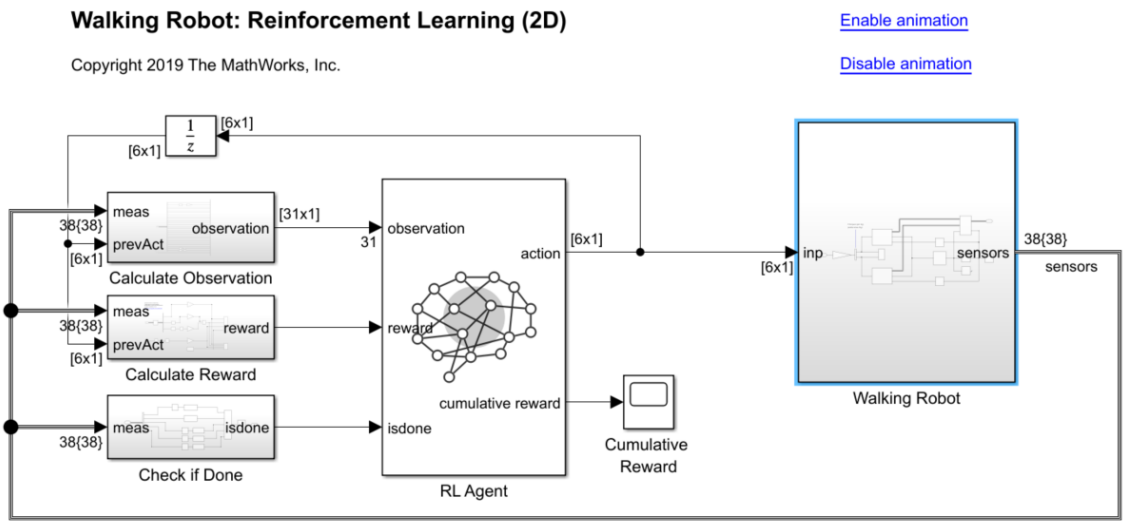
Jednostavna 3D tijela moguće je modelirati unutar modula te kreirate kinematičke veze između njih, no za složenija tijela i sklopove omogućen je prijenos geometrije iz CAD alata. Model koji je u narednom poglavlju predmet analize sadrži dijelove tijela robota izrađene u *Simscape*-u. [Slika 26.]



Slika 26. Stopalo robota

5. ANALIZA GOTOVOG MODELA HODAJUĆEG ROBOTA

U ovom poglavlju analiziran je gotovi model dvonožnog hodajućeg robota. Model je napravljen u potpunosti unutar *Matlab* programske platforme koja je u ranijem tekstu kratko opisana. Model je preuzet unutar programa u obliku *Matlab*-ovog *Add-on*-a to jest dodatka. Model se pokreće pomoću par skripti i jednog kompleksne blokovske sheme unutar *Simulink*-a. [Slika 27.]

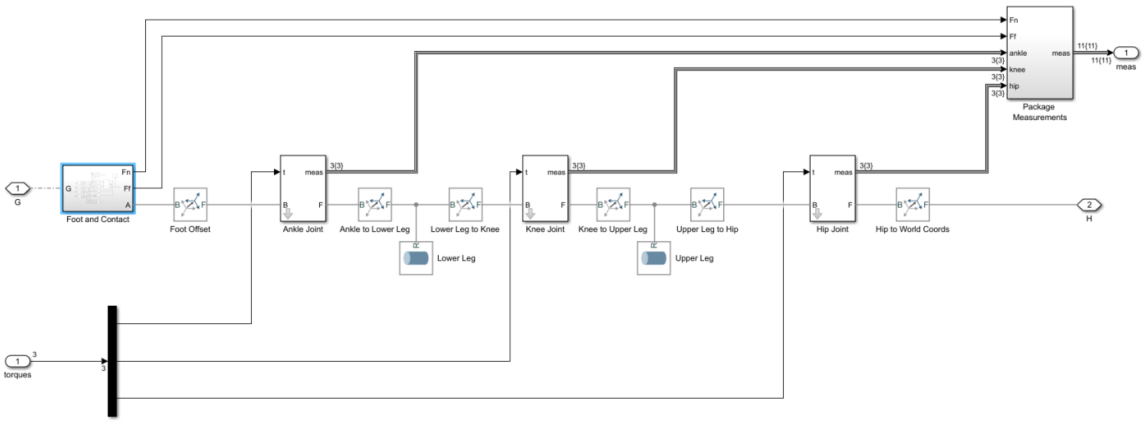


Slika 27. Model dubokog učenja pojačavanjem hodajućeg robota u Simulinku

U analizi je model rastavljen na svoje glavne dijelove. U odvojenim poglavljima opisat će se tijelo robota koje predstavlja okruženje, istrenirana neuronska mreža u ulozi agenta, signali opažanja i na kraju nagradna simbol.

5.1. Tijelo robota

Tijelo robota koje u sklopu dubokog učenja pojačavanjem vrši ulogu okruženja je na Slici 27. prikazano desnim blokom naziva „Walking Robot“. Unutar bloka, robot je podijeljen na dva veća bloka koji predstavljaju svaki jednu nogu (uključujući sve zglobove), blok trupa tijela, blok „World and Ground“, blok senzora te još nekoliko blokova i funkcija različitih uloga.



Slika 28. Model desne noge robota

Na slici 28. prikazan je model desne noge robota. Sastoji se od stopala koje je u posebnom bloku povezano sa površinom, tri zgloba, te natkoljenice i potkoljenice. Stopala i podloga povezani su pomoću „*Spatial Contact Force*“ blokova koji silu opisuju kao krutu oprugu s faktorom prigušenja. Modelirane su i četiri kuglice na vrhovima stopala kao dijeli stopala koji su u interakciji s podlogom. U modelu nogu najzanimljiviji su blokovi zglobova. Zglob je definiran vrijednosti čvrstoće, prigušenja, početnog kuta kao i krajnja dva kuta zakreta. Ulazni signal u blok zglobova tvore vrijednosti momenata pojedinog zgloba, a izmjerene pozicije i momente kao izlazni signal blok šalje u mjerni paket. U mjerni paket se također šalje sila reakcije podloge izmjerena na stopalu. U istom obliku modeliran je i blok lijeve noge.

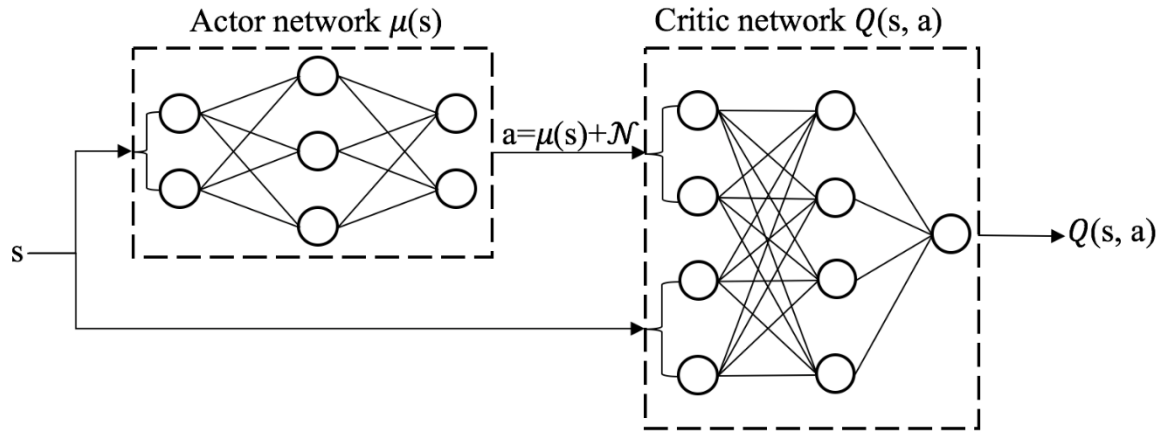
Blok „World and Ground“ nalazi se u funkciji zadavanja globalnog koordinatnog sustava modela te površine po kojoj se robot kreće. Spojem sa 6 stupnjeva slobode gibanja povezan je sa modelom trupa. Pomoću tog bloka definiran je i pravac kretanja odnosno trajektorija dvonožnog robota.

5.2. Neuronska mreža

U mapi s potrebnim datotekama za pokretanje simulacije, dane su i prethodno istrenirane neuronske mreže koje se implementiraju u model kako bi predstavljale ulogu agenta u procesu dubokog učenja pojačavanjem. Moguće je i samostalno konstruirati neuronsku mrežu (pomoću prethodno opisanih alata u poglavlju 3) te pokrenuti njeno treniranje uz vizualizaciju napretka.

Funkcija agenta je da uz znanja stečena iz prethodnih iteracija, donese nove odluke o akcijama koje će rezultirati prikupljanjem najveće moguće nagrade zadane nagradnim signalom. Nevezano uz vrstu odabranog agenta, on će primati podatke o okruženju, podatke nagradnog signala i informaciju o tome je li simulacija izvršena („*Check if Done"* blok*)*.

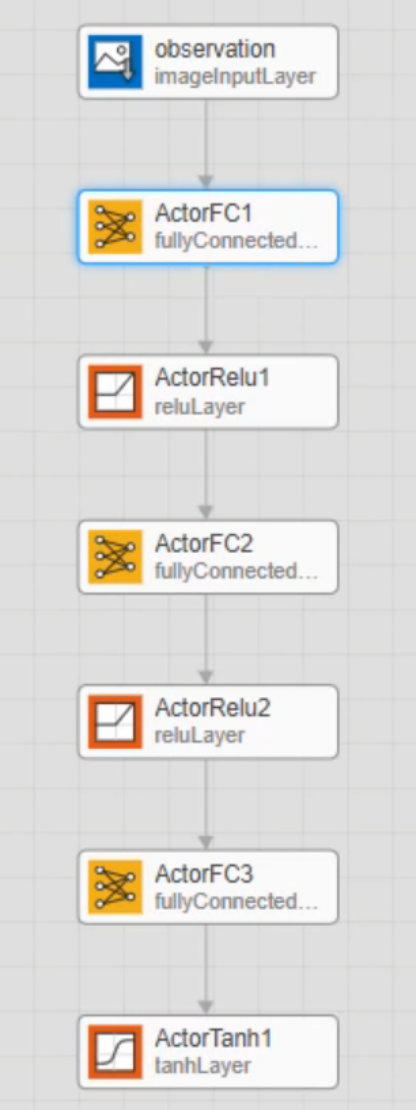
Agent je modeliran pomoću DDPG (*Deep Deterministic Policy gradient*) algoritma jer je to jedan od algoritama koji može podnijeti bilo koju zadanu kontinuiranu vrijednost akcija (bilo koja vrijednost momenta u ovom primjeru). DDPG sastoji se od prve neuronske mreže koja se naziva „Actor" i druge koja se naziva „Critic". [Slika 29.]



[10]

Slika 29. Shema DDPG algoritma

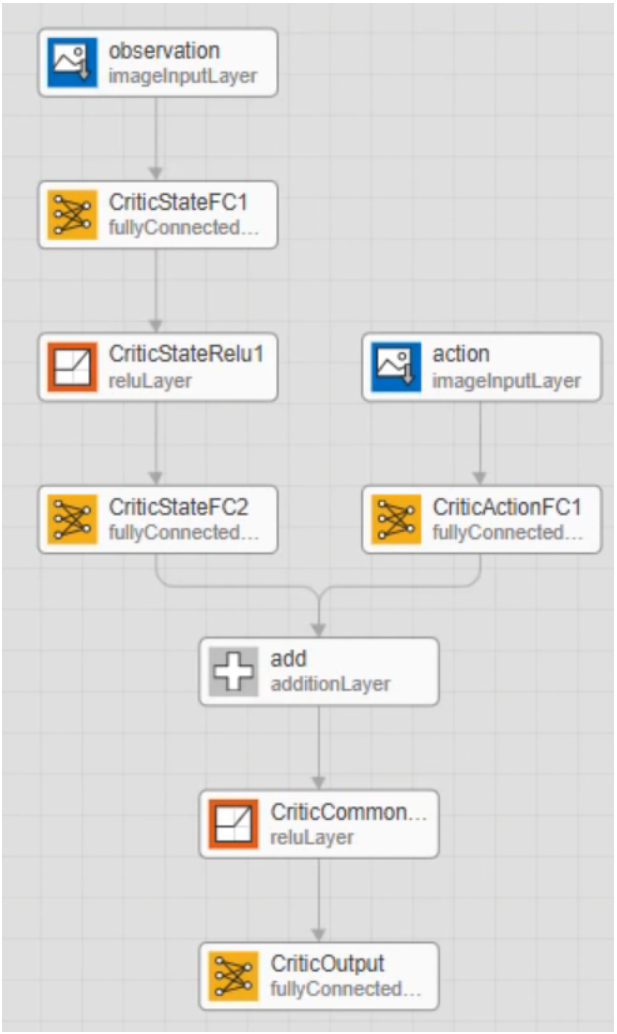
Actor djeluje kao kontroler sustava to jest primit će stanje okoline kroz zapažanje i poduzeti akciju. Međutim, to nije dovoljno za treniranje cijelog agenta pa se dodaje *Critic* neuronska mreža. Ona prima podatke o stanju okoline i podatke o poduzetim akcijama, a signal koji dalje šalje je procijenjena vrijednost nagrade koja će se prikupiti ako se poduzme određena akcija za određeno stanje. U analiziranom modelu cilj je naučiti Critic mrežu da što bolje procjeni koliko dobro će robot izvršavati radnju hodanja u određenom stanju.



Slika 30. Actor neuronska mreža

Kako je prikazano na Slici 30. Actor mreža sastoji se od ulaznog sloja, izlaznog sloja i jednog skrivenog sloja. Slojevi su potpuno povezani

Na slici 31. prikazana je Critic mreža. Naizgled je očito je da je ta mreža kompleksnija, ali primarni razlog tomu je što paralelno prima dvije vrste informacija.

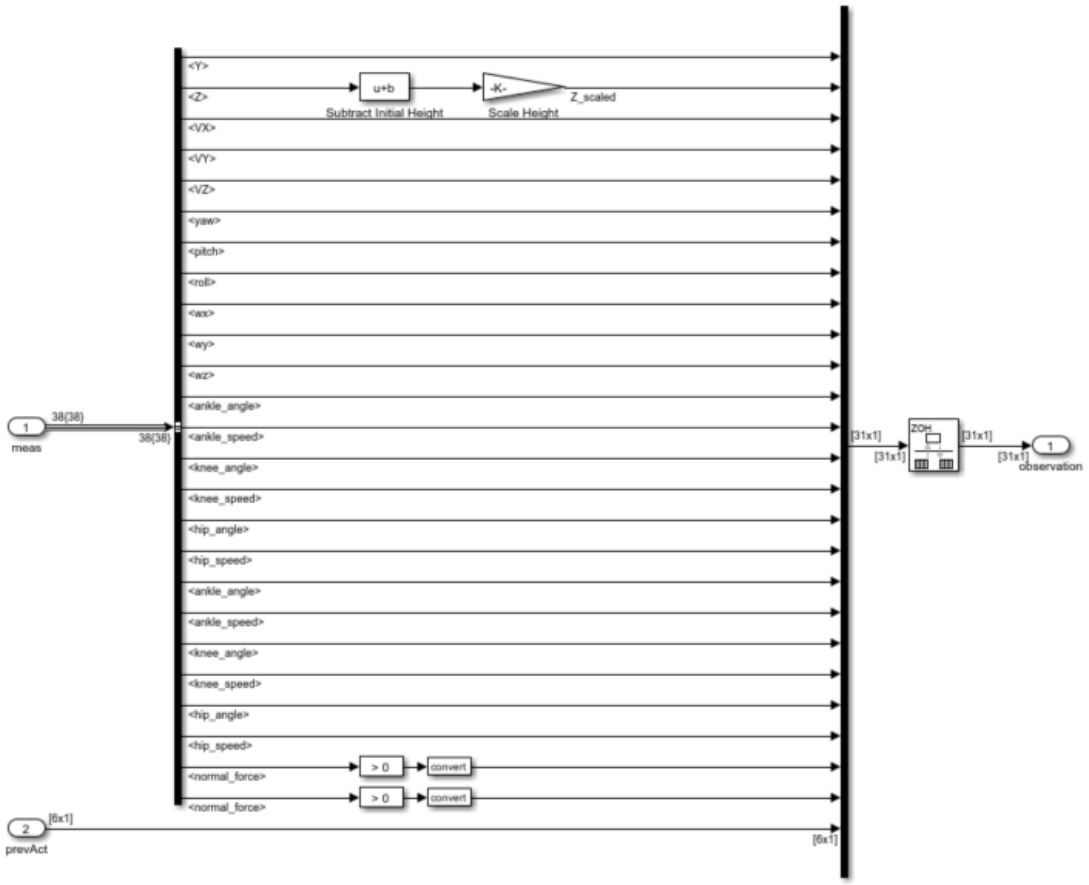


Slika 31. Critic neuronska mreža

Obje mreže mogu se modificirati u ranije opisanom alatu *Deep Network Designer*. Moguće je micati i dodavati slojeve, ali i mijenjati težinske faktore čvorova unutar slojeva. Učitavanjem mreža u modelu se stvara agent.

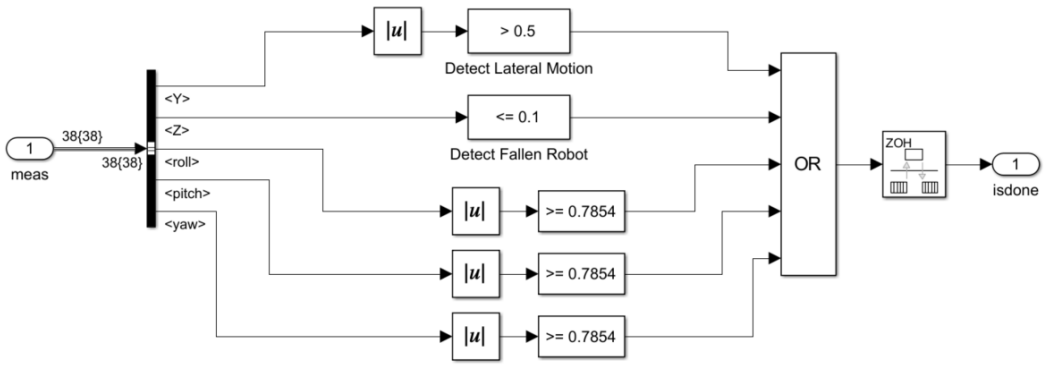
5.3. Zapažanja i uvjeti završetka simulacije

Na Slici 32. vidi se kako izgleda blok zapažanja. Radi se o bloku koji proizlazi iz okoline nakon svake akcije agenta, a sadrži podatke o svi mjerenim čimbenicima na temelju kojih agent dobiva sliku o stanju okoline.



Slika 32. Izgled bloka Zapažanja

Blok „Check if Done“ brine se o tome da se funkcija zaustavi kada krene u potpuno krivom smjeru. U njoj je moguće zadavati uvjete za koje se smatra da se nikada u procesu učenja ne bi trebali ostvariti ako učenje ide u dobro smjeru. U primjeru hodajućeg robota to su prekomjerno gibanje u lijevo ili desno, pad robota i povećanje kutova rotacije robota oko x, y i z osi iznad određene vrijednosti. [Slika 33.]. Kada je bilo koji od navedenih uvjeta ispunjen učenje se zaustavlja. Time štedimo vrijeme učenja i upućujemo agenta u kojem pravcu da nastavi proces učenja.



Slika 33. Check if Done Box

5.4. Nagradni signal

5.4.1. *Što je nagradna funkcija?*

Kako je prije pojašnjeno, učenje pojačavanjem djeluje tako što agent izvršava one akcije koje će ga dovesti do prikupljanja što veće nagrade. Nagradna funkcija je u pravilu nelinearna funkcija koja predstavlja numeričku vrijednost nagrade temeljenu na zapažanjima iz okruženja. Svaki opaženi podatak okruženja možemo povezati s brojkom ili drugim matematičkim izrazom. Cilj je pomoći algoritmu da shvati koja ga kombinacija vezanih radnji dovodi do željenog rezultata. Pojednostavljeno gledano, nagradama motiviramo kontroler da izvršava željene funkcije (npr. korak unaprijed u slučaju hodajućeg robota) i tjeramo ga da izbjegava neprikladne akcije (npr. pad robota). Korištenjem ovakvog sustava agent mijenja svoju politiku na temelju zaključaka, a ne pretpostavki.

5.4.2. *Kako osmisliti nagradnu funkciju?*

Važno je dobro osmisliti nagradnu funkciju kako bi agent izvršavao željene akcije. Pogrešno definirana nagradna funkcija može dovesti do neželjenih reakcija agenta. Nekada se može činiti da je nagradna funkcija logično osmišljena, ali računalo neće pravilno reagirati. Bitno je da agent „skuplja nagradu“ nakon strateški zadanih ciljeva.

U slučaju da je signal oblikovan tako da agent skuplja naredbu nakon nekoliko odrađenih akcija, od kojih je zadnja nepoželjna i agenta se kažnjava, on će pomisliti kako su sve prethodne akcije krive i izbjegavat će njihovo izvođenje. Nagrade dakle mogu biti postavljene nakon svakog koraka ili na kraju cijelog niza akcija ili oboje. Bitno je i ocijeniti težinsku

vrijednost nagrade za pojedinu akciju. Kontroler će uvijek težiti ka većoj nagradi te stoga moramo paziti da smanjimo vrijednost onih nagrada koje su dalje u budućnosti. U protivnom će agent težiti dolasku do te nagrade i prethodno izvršiti nepoželjne akcije. To možemo jednostavno pojasniti na primjeru hodajućeg robota. U slučaju da je robotu za prijeđenu udaljenost namijenjena veliku nagradu, ako je dovoljno visok on može pasti i u padu doći do željene udaljenosti i kada prikupi naredbu mislit će da je posao dobro obavljen. Također buduće nagrade nisu pouzdane jer izvršavanjem akcija mijenja se i okruženje robota, a promjene mogu dovesti do mijenjanja vrijednosti nagrada.

Kako postoje negativne nagrade to jest kazne, logično se čini u nagradnu funkciju uvrstiti kaznu za padanje. Negativne nagrade uvelike mogu pomoći u učenju, no s njima treba biti posebno oprezan. Kaznama treba dodijeliti odgovarajuće vrijednosti jer bi algoritam „u strahu“ od dobivanje kazne mogao izbjegavati i neke dobrodošle radnje.

Učenje pojačavanjem dozvoljava i opciju u kojoj agent tj. kontroler ne zna ništa o okruženju te može djelovati samo na temelju nagradne funkcije. Takav način učenja se zove *Model-free Reinforcement learning*. U slučaju kada su poznate dimenzije robotske konstrukcije, način rada njihovih spojeva, podaci iz senzora i na primjer brzina kretanja, šteta bi bilo ne iskoristiti te podatke u svrhu olakšavanja posla algoritmu.

Uvođenjem terminalnih uvjeta može se spriječiti agenta da djeluje u područjima koja će ga odvesti u apsolutnom krivom smjeru. Oni određuju kada je simulaciji kraj. Kraj se obično određuje nakon određenog broja ponavljanja ili vremenskog perioda. Također simulaciji se može zadati da stane, ako krene predaleko u neželjenom smjeru.

Na slici ispod teksta [Slika 34.] možemo vidjeti koji ulazni podaci (akcije) mijenjaju okruženje, a koji izlazni podaci (zapažanja) daju povratnu informaciju agentu o kvaliteti i učinkovitosti akcija. O opažanjima ovisi vrijednost prikupljene nagrade.

x,y,z koordinata robota (3)

brzina u x,y i z smjeru (3)

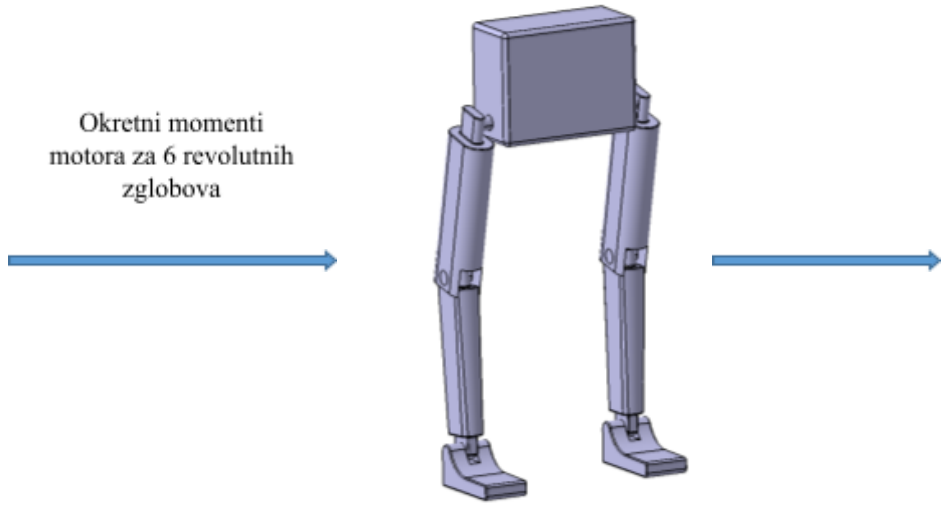
orijentacija robota (3)

vrijednost kuta orijentacije robota (3)

orijentacija zglobova (6)

vrijednost kuta orijentacije zglobova (6)

kontaktne sile na tlu (2)



Slika 34. Ulazni i izlazni podaci okruženja

5.4.3. *Nagradni signal dvonožnog hodajućeg robota*

Stečenim znanjem o načinu rada i važnosti nagradne funkcije, ali i nekim osnovnim logičnostima njenog oblikovanja u ovom poglavlju će se predstaviti nagradna funkcija hodajućeg robota. Cilj nagradne funkcije u ovom primjeru jest da se robot nauči kretati prostorom po pravcu u smjeru osi x. Najprije će se predstaviti funkcija u implicitnom zapisu. U svrhu oblikovanja nagradne funkcije koriste se sljedeće oznake

R - vrijednost nagradne funkcije

v_x - translacijska brzina u smjeru osi x

y – translacija robota u negativnom ili pozitivnom smjeru osi y u odnosu na pravac osi x

z – vertikalna translacija

u – pokretni moment zglobova

t – aktualni broj koraka ponavljanja

i – indeks određenog zgloba

T_s – vrijeme uzorka

T_f – vrijeme konačne simulacije

Počet će se sa funkcijom (5.1) u kojoj se nagrađuje algoritam ako se krene gibati u smjeru osi x, ali na način da u se funkciju uvrštava translacijsku brzinu kretanja robota u željenom smjeru umjesto same translacije.

$$R = 0.8 * v_x$$

(5.1)

Rezultat simulacije je skok robota prema naprijed i brzi pad. Kako je cilj naučiti robota hodati, skok i pad se oboje smatraju nepoželjnim pa se u sljedećoj nadogradnji (funkcija (5.2)) kažnjava pokret u smjeru osi z.

$$R = 0.8 * v_x - 50 * z^2$$

(5.2)

Ovime se dobiva malo bolji rezultat. Naime robot se počinje gibati, ali ne samo u smjeru osi x, nego se i odmiče od pravca osi x u smjeru y osi. Drukčije rečeno giba se istovremeno naprijed i lijevo. Glavni uzrok takvom gibanju leži u činjenici da lijevu nogu koristi značajno više nego desnu. U funkciji (5.3) dodaje se negativno nagrađivanje momenta zgloba iz prethodnog koraka kako bi se ujednačilo korištenja zglobova i kako bi pokreti postali glađi.

$$R = 0.8 * v_x - 50 * z^2 - 0.02 * \sum_{i=1}^{6} {u_{t-1}^i}^2$$

(5.3)

Pokretanjem simulacije vidi se veliki napredak. Robot se sasvim pristojno kreće u smjeru osi x, no krivuda lijevo pa desno i tako tokom cijelog procesa hodanja. Rješenje se nameće samo od sebe. U funkciji (5.4) penalizirat će se translacijsko gibanje u pozitivnom i negativnom smjeru osi y.

$$R = 0.8 * v_x - 50 * z^2 - 0.02 * \sum_{i=1}^{6} {u_{t-1}^i}^2 - 5 * y^2$$

(5.4)

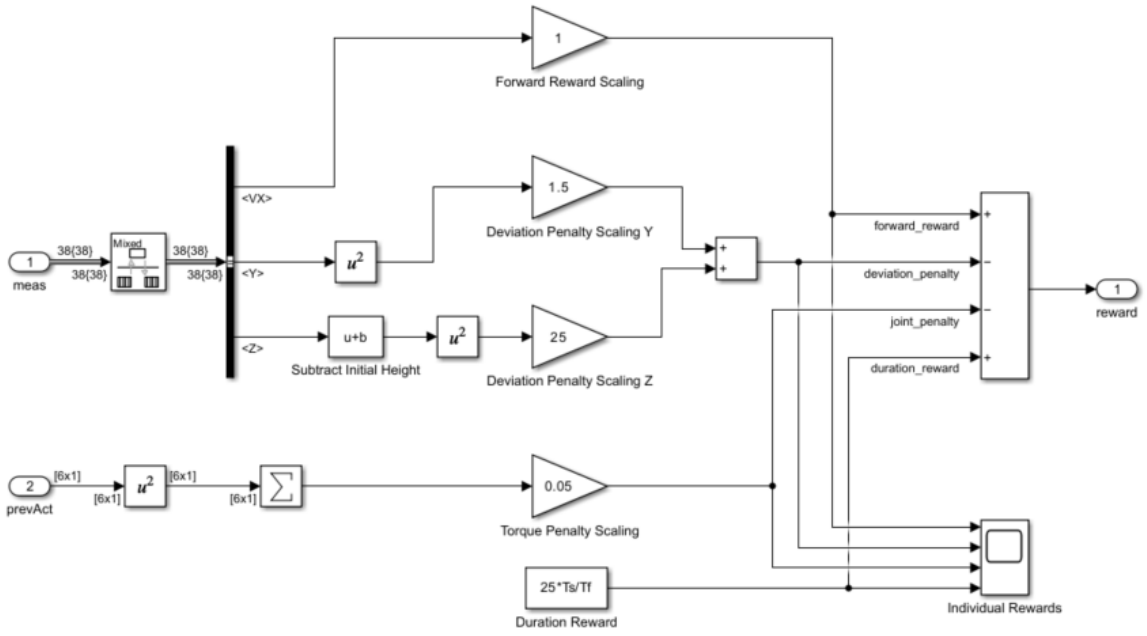
Može se reći da je ovom nadogradnjom nagradne funkcije robot naučio hodati na sasvim zadovoljavajući način. U funkciju (5.5) nadodat ćemo još jedan član koji će nagrađivati robota u svakom vremenskom koraku, izbjegavajući tako prekid hodanja.

$$R = 0.8 * v_x - 50 * z^2 - 0.02 * \sum_{i=1}^{6} {u_{t-1}^{i}}^2 - 5 * y^2 + 20\frac{T_s}{T_f}$$

(5.5)

Iako je sasvim sigurno moguće dodatno nadograđivati funkciju ili joj dati novi i bolji oblik, pokazalo se da nagradna funkcija (5.5) obavlja zadovoljavajuće dobar posao. Robot dakle hoda u pravcu osi x jednoliko se koristeći s obje noge. Također je bitno da se robot ne ruši, ne skače i ne radi neobične radnje ne bi li čim prije došao do nagrade.

Kratko sumiranje članova nagradne funkcije govori da se agent nagrađuje pomicanjem modela prema naprijed i dobivanjem brzine u smjeru osi x. Nagradom ga se također traži da izbjegava prekid radnje. Ostali članovi nagradnog signala usmjereni su na kažnjavanje agenta za promjenu pozicije u smjerovima osi y i osi z te za pretjeranu upotrebu momenta u zglobovima.



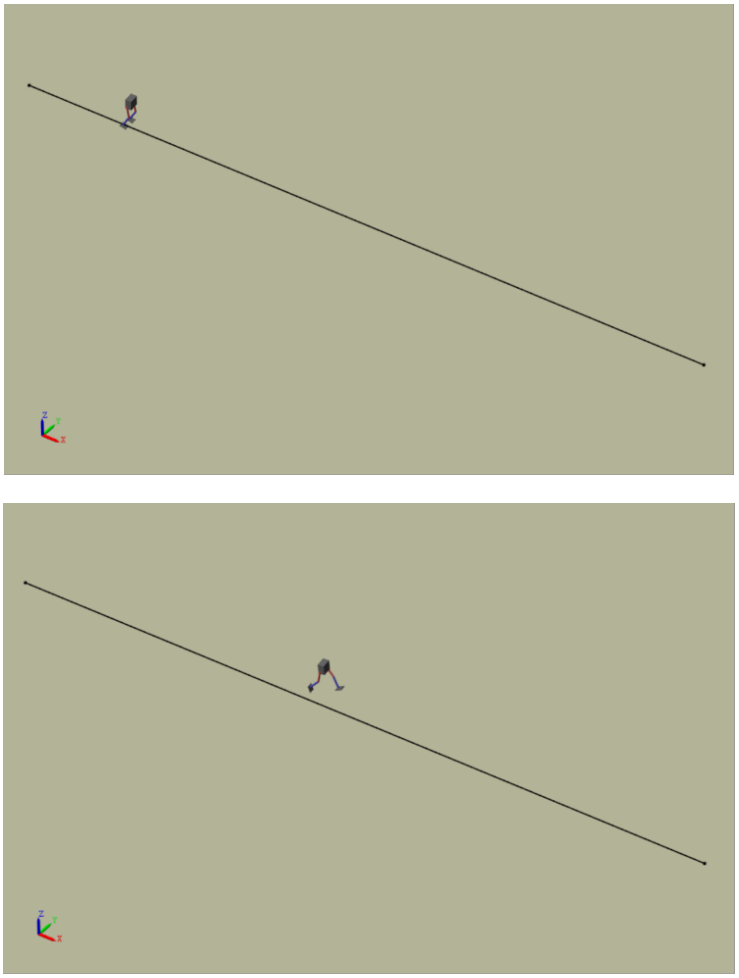
Slika 35. Nagradni signal u obliku blokovskog dijagrama

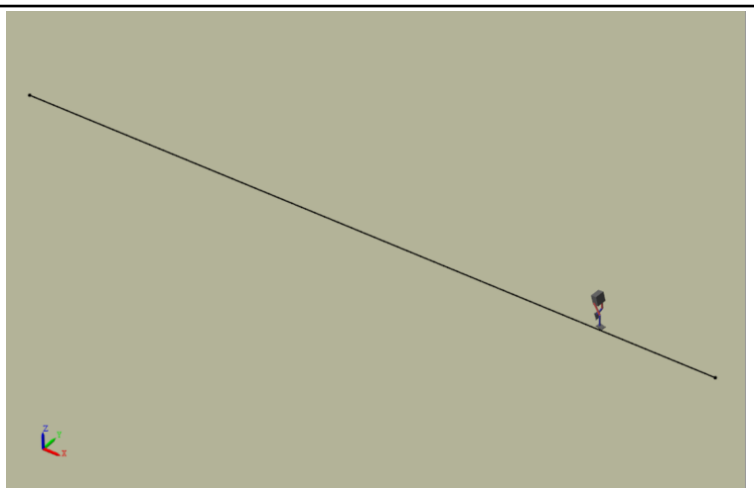
5.5. **Pokretanje simulacije**

Uz opisani blokovski model, potrebno je iz *Matlab* skripti učitati podatke koji su potrebni za postupak. Prva od njih je „startupWalkingRobot“. Njenim pokretanjem u programu se definira put kojim će se naknadno učitavati ostale skripte i *Simulink* modeli. Nakon početne skripte, pokreću se skripte „walkerInvKin“ i „robotParametersRL“. U njima će sustav pronaći sve podatke o fizičkom modelu robota to jest stvorti će se slika okoline. Pokrenut će se i skripta „walkerResetFcn“ koja omogućava ponovno pokretanje simulacije s drukčijim početnim uvjetima. Još ostaje implementirati agenta te učitati „walkingRobotUtils“ *library* unutar kojeg se nalaze podaci o određenim kompleksniim blokovima korištenima u modelu. Postaje vidljivo kako su svi potrebni parametri uneseni u *Matlab Workspace*.

5.6. **Rezultati simulacije**

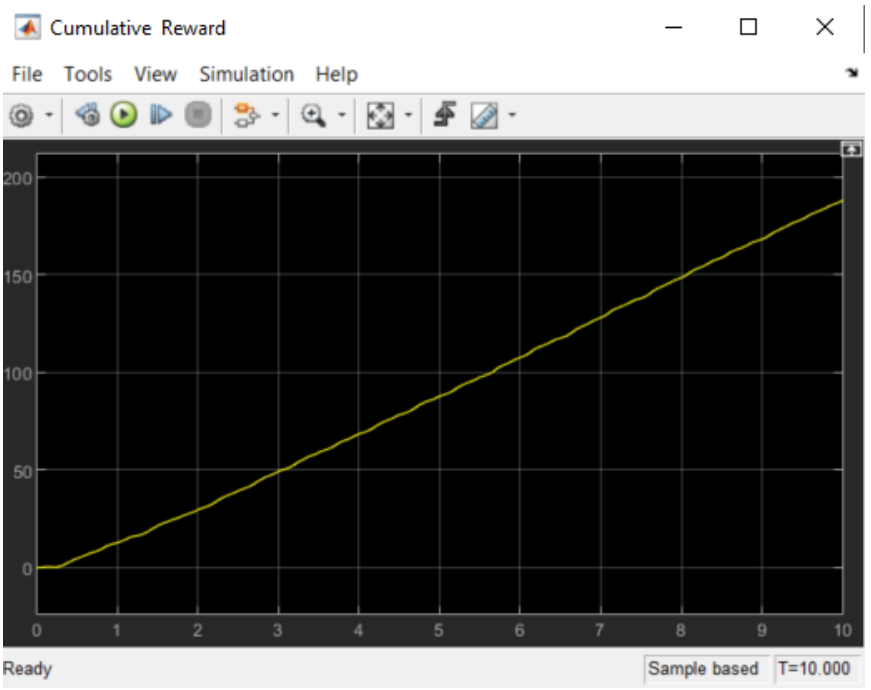
Slika 36. prikazuje robota u tri različita položaja tokom hodanja kako bi se vidjelo da je simulacija uspješna





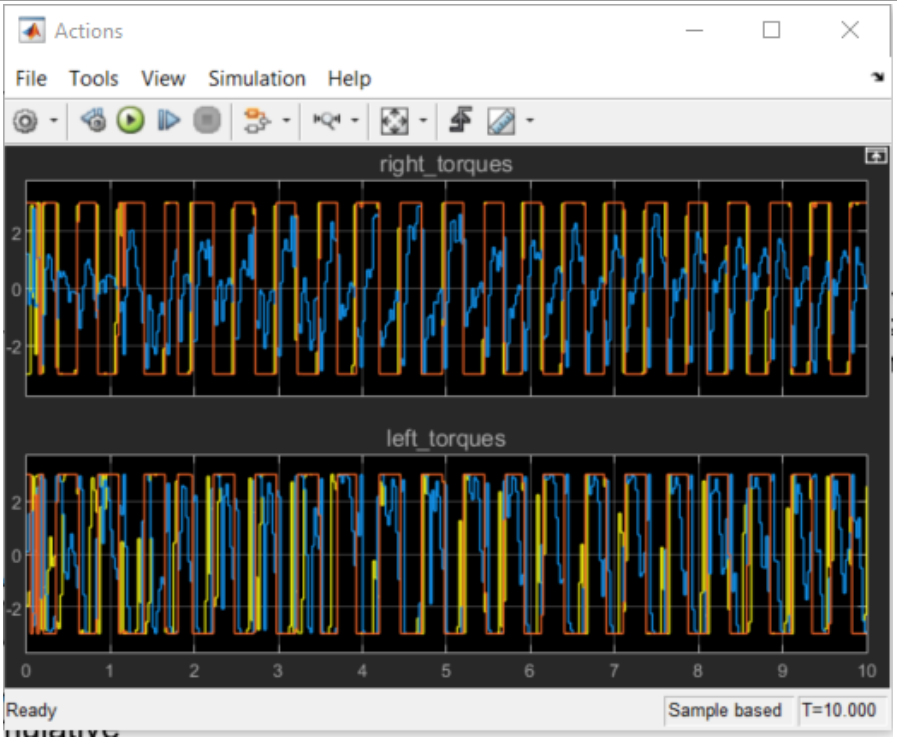
Slika 36. Simulacija kretanja robota

Slika 37. prikazuje graf vrijednosti nagradnog signala u ovisnosti o vremenu. Vrlo jasno je prikazano kako agent cijelo vrijeme teži što većoj nagradni i postiže skoro linearan rast nagrade kroz cijelu simulaciju.



Slika 37. Graf nagradnog signala

Na Slici 3. može se u vremenu pratiti upravljanje momentima zglobova. Vidi se da lijeva i desna strana rade prilično ujednačeno što omogućuje kretanje unaprijed.



Slika 38. Graf momenata zglobova

6. ZAKLJUČAK

U ovome radu pokazalo se kako su umjetna inteligencija i strojno učenje vrlo efikasan način kreiranja vrlo kompleksnih sustava. Valja naglasiti kako je za to potreban i učinkovit programski alat, a *Matlab* se pokazao upravo takvim. Jednostavnim korištenjem i nizom dodatnih sučelja uvelike je olakšan proces. Iako je analizirani model tek početak za stvarnu izradu hodajućeg robota, smatram da je odlična osnova za dodatne nadogradnje. Premda je već definirani model, daje priliku korisnicima za testiranje niza modifikacija i vizualizaciju rezultata kao i praćenje procesa učenja u stvarnom vremenu.

LITERATURA

[1] https://www.techopedia.com/topic/87/artificial-intelligence datum pristupa: 17. 09.2022.

[2] https://www.mathworks.com/videos/deep-reinforcement-learning-for-walking-robots--1551449152203.html?s_tid=srchtitle_walking%20robot_3 datum pristupa: 17. 09.2022.

[3] https://medium.datadriveninvestor.com/exploring-machine-learning-f1dc6f3ec902 datum pristupa 15.09.2022.

[4] https://nhadep247.net/machine-learning-algorithms-with-real-time-examples-1659160807/ datum pristupa 15.09.2022.

[5] https://omdena.com/blog/supervised-and-unsupervised-machine-learning/

[6] https://www.mathworks.com/products/simscape.html datum pristupa: 17. 09.2022.

[7] Curkovic, Petar, and Bojan Jerbic. "Honey-bees optimization algorithm applied to path planning problem." International Journal of Simulation Modelling (IJSIMM) 6, no. 3 (2007).

[8] https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning datum pristupa: 16. 09.2022.

[9] https://www.researchgate.net/figure/Reinforcement-learning-schematic-Reinforcement-learning-RL-can-be-formulated-as-a_fig4_322424392 datum pristuoa 16.09.2022.

[10] https://www.arxiv-vanity.com/papers/1811.07522/ datum pristupa 17.09.2022.

[11] https://serokell.io/blog/ai-ml-dl-difference datum pristupa 17.09.2022.

[12] https://www.researchgate.net/figure/Revolute-1-DOF-Joint_fig1_311414943 datum pristupa 14.09.2022.

[13] Tuomas Haarnoja, Sehoon Ha: Learning to Walk via Deep Reinforcement Learning

[14] Ćurković, Petar, and Lovro Čehulić. "Diversity maintenance for efficient robot path planning." Applied Sciences 10, no. 5 (2020): 1721.

[15] Curkovic, P.; Jerbic, B.; Stipancic, T. (2008). Hybridization of adaptive genetic algorithm and ART 1 neural architecture for efficient path planning of a mobile robot, Transactions of FAMENA, Vol. 32, No. 2, 11-21.

[16] https://www.researchgate.net/publication/335109551_Deep_Reinforcement_Learning_for_Network_Slicing_with_Heterogeneous_Resource_Requirements_and_Time_Varying_Traffic_Dynamics datum pristupa: 12. 09.2022.

[17] https://medium.com/@vishnuvijayanpv/deep-reinforcement-learning-value-functions-dqn-actor-critic-method-backpropagation-through-83a277d8c38d datum pristupa 18.09.2022.

[18] https://chatbotslife.com/how-neural-networks-work-ff4c7ad371f7 datum pristupa 16.09.2022.

[19] Ćurković, Petar, and Bojan Jerbić. "Dual-arm robot motion planning based on cooperative coevolution." In Doctoral Conference on Computing, Electrical and Industrial Systems, pp. 169-178. Springer, Berlin, Heidelberg, 2010.

PRILOZI

I. Matlab programski kod

```
% Walking Robot Startup Script
%
% Copyright 2017-2019 The MathWorks, Inc.

%% Clear everything
clc
clear
close all

%% Add folders to the path
addpath(genpath('LIPM'), ...                    % Linear inverted pendulum model
(LIPM) files
        genpath('ModelingSimulation'), ...      % Modeling and simulation files
        genpath('Optimization'), ...            % Optimization files
        genpath('ControlDesign'), ...           % Control design files
        genpath('ReinforcementLearning'), ...   % Reinforcement learning files
        genpath('Libraries'));                  % Other dependencies

%% Load basic robot parameters from modeling and simulation example
robotParameters

%% Open the README file
edit README.md
```

```
function angs = walkerInvKin(pos, upper_leg_length, lower_leg_length,dim)
% 3D Inverse Kinematics to set the robot leg initial conditions
% The "dim" argument can be set to "2D" or "3D" for the different cases.
%
% Copyright 2019 The MathWorks, Inc.

    % Unpack
    x = pos(1);
    if isequal(dim,'3D')
        y = pos(2);
    else
        y = 0;
    end
    z = pos(3);

    % Compute the hip angle
    hipRoll = -atan2(y,-z);

    % Call symbolic generated function for 2D leg
    zMod = z/cos(hipRoll);

    maxMag = (upper_leg_length + lower_leg_length)*0.99; % Do not allow fully
extended
    mag = sqrt(x^2 + zMod^2);
    if sqrt(x^2 + zMod^2) > maxMag
```

```
        x = x*maxMag/mag;
        zMod = zMod*maxMag/mag;
    end

    % Call 2D Inverse Kinematics
    theta = legInvKin(upper_leg_length,lower_leg_length,-x,zMod);

    % Address multiple outputs
    if size(theta,1) == 2
       if theta(1,2) < 0
          hipPitch = theta(2,1);
          kneePitch = theta(2,2);
       else
          hipPitch = theta(1,1);
          kneePitch = theta(1,2);
       end
    else
        hipPitch = theta(1);
        kneePitch = theta(2);
    end

    % Set ankle angles
    anklePitch = -kneePitch - hipPitch;
    ankleRoll  = -hipRoll;

    % Pack
    angs = [ankleRoll; anklePitch; kneePitch; hipRoll; hipPitch];

end



% Walking Robot Parameters -- Reinforcement Learning
% Copyright 2019 The MathWorks, Inc.

%% Model parameters
% Mechanical
density = 500;
foot_density = 1000;
if ~exist('actuatorType','var')
    actuatorType = 1;
end
world_damping = 1e-3;
world_rot_damping = 5e-2;

% Contact/friction parameters
contact_stiffness = 500;
contact_damping = 50;
mu_k = 0.7;
mu_s = 0.9;
mu_vth = 0.01;
height_plane = 0.025;
plane_x = 25;
plane_y = 10;
contact_point_radius = 1e-4;

% Foot dimensions
```

```
foot_x = 5;
foot_y = 4;
foot_z = 1;
foot_offset = [-1 0 0];

% Leg dimensions
leg_radius = 0.75;
lower_leg_length = 10;
upper_leg_length = 10;

% Torso dimensions
torso_y = 8;
torso_x = 5;
torso_z = 8;
torso_offset_z = -2;
torso_offset_x = -1;
mass = (0.01^3)*torso_y*torso_x*torso_z*density;
g = 9.80665;

% Joint parameters
joint_damping = 1;
joint_stiffness = 0;
motion_time_constant = 0.01;
joint_limit_stiffness = 1e4;
joint_limit_damping = 10;

%% Reinforcement Learning (RL) parameters
Ts = 0.025; % Agent sample time
Tf = 10;    % Simulation end time

% Scaling factor for RL action [-1 1]
max_torque = 3;

% Initial conditions
h = 18;     % Hip height [cm]
init_height = foot_z + h + ...
              torso_z/2 + torso_offset_z + height_plane/2;
vx0 = 0;    % Initial X linear velocity [m/s]
vy0 = 0;    % Initial Y linear velocity [m/s]
wx0 = 0;    % Initial X angular velocity [rad/s]
wy0 = 0;    % Initial Y angular velocity [rad/s]
% Initial foot positions [m]
leftinit =  [0;0;-h/100];
rightinit = [0;0;-h/100];

% Calculate initial joint angles
init_angs_L = walkerInvKin(leftinit, upper_leg_length/100,
lower_leg_length/100,'3D');
init_angs_R = walkerInvKin(rightinit, upper_leg_length/100,
lower_leg_length/100,'3D');
```