Umetna inteligenca za igranje namiznega nogometa

David Hožič

Mentorja: Andrej Zdešar in Matevž Bošnak

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška c. 25, 1000 Ljubljana, Slovenija

dh8091@student.uni-lj.si, andrej.zdesar@fe.uni-lj.si, matevz.bosnak@fe.uni-lj.si

Artificial intelligence for table football control

The article presents the development of agents or strategies for playing table football using reinforcement learning methods, which are from the field of artificial intelligence. First, the game of table football itself is described, followed by a description of reinforcement learning, and then the development of the algorithm or strategy for the game is presented. Before development or training of the strategy, a table football simulation was created in the MuJoCo simulation environment. Additionally, a manually programmed agent in the form of a state machine with expert knowledge was developed for the purpose of strategies training. Using reinforcement learning, a defensive strategy was developed for the goalkeeper and defense, where the manually programmed agent attempted to score a goal while the learned strategy aimed to prevent it. The results were evaluated in the simulated environment by comparing the ratio of saved shots to total shots taken. Furthermore, the strategy was tested on a real table against multiple human teams, where it was integrated into a hybrid system alongside the manually programmed agent.

Kratek pregled prispevka

Članek predstavlja razvoj agentov oz. strategij za igranje namiznega nogometa z uporabo metod spodbujevanega učenja, ki sodijo med algoritme umetne inteligence. Najprej je predstavljena igra namiznega nogometa, sledi analiza problematike spodbujevanega učenja, nato pa je predstavljen razvoj algoritma oziroma strategije za igro. Pred razvojem oz. učenjem strategije smo v okolju MuJoCo izdelali simulacijo namiznega nogometa. Prav tako smo za namene učenja strategij ročno razvili algoritem vodenja v obliki končnega avtomata z ekspertnim znanjem. Z uporabo spodbujevanega učenja smo nato razvili strategijo branjenja za vratarja in obrambo, kjer je ročno programirani agent poskušal zadeti gol, učena agenta pa sta gol poskušala ubraniti. Rezultate smo vrednotili v simuliranem okolju s primerjavo razmerja ubranjenih in vseh izvedenih strelov. Dodatno smo strategijo preizkusili na dejanski mizi proti več človeškim ekipam, kjer je bila strategija integrirana v hibridni sistem skupaj z ročno programiranim agentom.

1 Uvod

Namizni nogomet je pomanjšana različica nogometa, kjer, namesto človeških igralcev, na igrišču stojijo figure na igralni mizi. Te so vodene preko štirih palic (na vsaki strani), običajno s strani človeškega para. Namizni nogomet je pogosto uporabljen za namene zabave in sprostitve, izvajajo pa se tudi tekmovanja, kot je svetovno prvenstvo pod okriljem Mednarodne zveze za namizni nogomet ITSF, zato je včasih obravnavan tudi kot vrsta športa.

Za igro sta potrebna dva igralca (eden na vsaki strani), za bolj zanimivo in dinamično igro pa štirje. V primeru, da štirih igralcev ni na voljo, je smiselno eno ekipo zamenjati z avtomatiziranim sistemom. Avtomatizacija ekipe namiznega nogometa je poleg nadomestitve manjkajočih igralcev uporabna tudi za učenje igranja človeškega igralca. Na primer, ob adaptivnem algoritmu lahko človeški igralec izboljša svoje sposobnosti. Prav tako je avtomatizacija smiselna za področje vodenja sistemov, saj omogoča študijo algoritmov v nadzorovanih laboratorijskih pogojih.

V članku se osredotočimo na razvoj umetne inteligence za igranje namiznega nogometa. Pri razvoju upoštevamo pravila igre namiznega nogometa; na primer, igralec ne sme vrteti palic, zadrževati žogice, itd.

1.1 Avtomatizirani sistem namiznega nogometa

Cilj dela je izdelava avtonomnih agentov za igranje namiznega nogometa, ki dobro delujejo tako v simuliranem okolju kot tudi na pravi fizični napravi. Ciljno okolje je avtomatizirana miza namiznega nogometa, prikazana na sliki 1. Stran modre ekipe je namenjena človeški ekipi, rdeča pa ima na vsaki palici nameščena dva servo motorja za translacijo in rotacijo palice. Sistem zaznavanja in merjenja žogice ter palic je izveden na podlagi dveh visokofrekvenčnih kamer (100 Hz), kjer sta zaznavanje in merjenje lege palic podrobno opisana v [1]. Pošiljanje ukazov za premik palic, kot tudi branje stanja dogajanja na

Slika 1: Ciljno okolje, tj. avtomatizirana miza namiznega nogometa.

mizi, je mogoče preko HTTP-vmesnika. Podatki se pošiljajo in prejemajo v obliki JSON. Prejeti podatki, ki podajajo položaj in hitrost žogice ter položaje vseh palic, opisujejo trenutno stanje igre na mizi. V sistemu je mogoče zaznati prisotnost časovne zakasnitve, ki znaša približno 40 ms, kar vključuje čas procesiranja kamer oz. slik, čas komunikacije in mrtvi čas motorjev.

1.2 Spodbujevano učenje

Spodbujevano učenje (angl. *reinforcement leaning*) predstavlja nabor algoritmov za iskanje optimalne rešitve danega problema, ki temelji na (naključnem) poskušanju in vrednotenju preizkusov.

Glavna sestavna dela spodbujevanega učenja sta agent in okolje, v katerem agent izvaja akcije. Akcije so odvisne od stanja okolja in morebitnega lastnega stanja agenta, ki sta združena v skupno stanje s_t. Agent po vsakem koraku v okolju t, oz. akciji $a_t(s_t)$, prejme nagrado $r(s_t, a_t)$ [2]. Vsem algoritmom spodbujevanega učenja je cilj izdelati preslikavo iz stanja na akcijo, ki bo vodila do največje akumulirane nagrade v epizodi. Epizoda je definirana kot zaporedje stanj, akcij in nagrad po vsakem koraku od začetnega stanja do zaključnega stanja, neveljavnega stanja ali drugega pogoja (npr. omejitev števila korakov). V sklopu spodbujevanega učenja se akumulirana nagrada imenuje donos (angl. *return*), preslikavo stanja na akcijo pa imenujemo strategija (angl. *policy*).

V splošnem se algoritmi delijo po tem ali upo-

rabljajo oz. izdelajo model okolja — modelni in brez-modelni (angl. *model-based* & *model-free*), kot tudi po podatkih uporabljenih za učenje — neposredni in posredni (angl. *on-policy* & *off-policy*). Neposredni algoritmi se učijo le iz podatkov, ki jih pridobijo po zadnji posodobitvi strategije, posredni pa podatke črpajo iz pomnilnika preteklih izkušenj (angl. *replay buffer*).

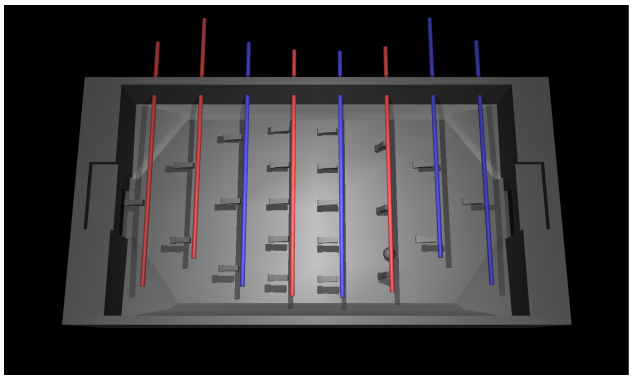
Dodatno ločimo tudi algoritme, ki pri učenju strategijo optimizirajo z uporabo gradienta donosa. To vrsto algoritmov imenujemo gradient strategije (angl. *policy gradient*). Algoritmi te vrste iterativno povečujejo verjetnost izbire akcij, ki vodijo k največjemu pričakovanemu donosu iz trenutnega stanja [3].

2 Metodologija

Algoritmi spodbujevanega učenja za doseg uporabnih strategij običajno potrebujejo veliko realnega časa oz. vzorcev. Za pohitritev učenja je zato smiselna uporaba simuliranega okolja.

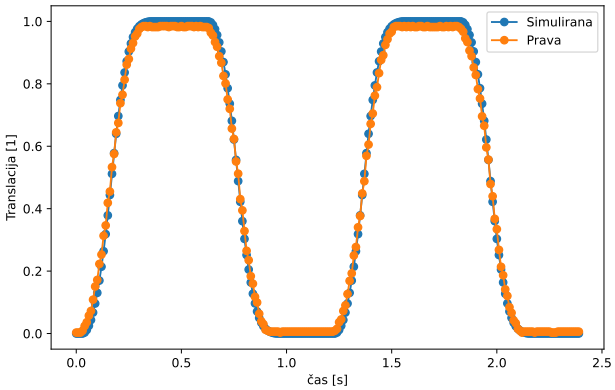
2.1 Simulacija sistema namiznega nogometa

Za pohitritev učenja strategij smo v okolju MuJoCo [4] izdelali simulacijo realnega okolja. Simulacija je prikazana na sliki 2. Pri izdelavi smo izhajali iz 3D-modela mize. Ker simulacijsko okolje MuJoCo ne omogoča zaznave trkov nekonveksnih geometrij [5], smo 3D-model razrezali na več konveksnih kosov. Pri tem smo osnovnejše geometrijske objekte modelirali z MuJoCo-vimi bazičnimi geometrijami, kompleksnejše pa vključili v obliki razrezane mreže (angl. *mesh*).



Slika 2: Simulacija mize namiznega nogometa.

Poleg statične geometrije je potrebno modelirati tudi dinamiko objektov — igralnih palic z aktuatorji in žogico. Aktuatorje posameznih palic smo modelirali z MuJoCo-vimi pozicijskimi regulatorji. Za določitev parametrov aktuatorjev smo v pravem okolju posneli odzive na več stopnic, za vsako palico ločeno. Posnete odzive smo nato uporabili kot referenco v Bayesovi optimizaciji [6]. Pri tem smo poskušali zmanjšati povprečno kvadratno napako med pravim in simuliranim odzivom. Odziv translacijske stopnje vratarja po optimizaciji prikazuje slika 3, podobno pa izgledajo tudi odzivi ostalih palic.



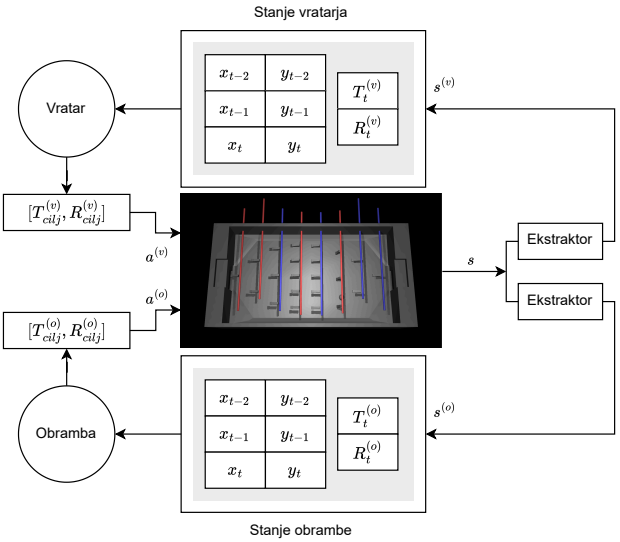
Slika 3: Translatorni odziv palice vratarja po določitvi parametrov. Modra krivulja predstavlja odziv simulirane palice, oranžna pa odziv prave.

Dinamične parametre žogice smo določili na podlagi "metode ostrega pogleda", dodatno pa smo za izboljšanje prenosa na fizično mizo žogici dodali pasivno dušenje, ki se je naključno spreminjalo.

2.2 Razvoj strategij

Po izdelavi simulacije smo okoli nje z uporabo knjižnice Gymnasium [7] izdelali okolje za interakcijo algoritma spodbujevanega učenja. V okolju epizodo opredelimo kot zaporedje korakov, ki se začne v naključnem stanju in konča ob nastopu enega izmed naslednjih dogodkov:

- prejet ali zadet gol,
- ustavitev igre zaradi mrtve žogice ali žogice izven igrišča,
- doseg omejitve 250 korakov (5 sekund).



Slika 4: Interakcija agentov z okoljem za učenje strategije vratarja (v) in obrambe (o).

Učili smo dve strategiji: strategijo za vratarja (za prvo palico z eno figuro) in strategijo za obrambo (za drugo palico z dvema figurama). Obe strategiji imata skupen primaren cilj, tj. branjenje lastnega gola, in delujeta istočasno. Učeni sta bili z algoritmom PPO (angl. *Proximal Policy Optimization*) [8], ki spada med algoritme z neposrednim učenjem strategije in v skupino algoritmov gradienta strategije.

Stanje smo obema strategijama definirali na enak način. Ob vsakemu koraku t prejmeta podatke o poziciji žogice $(x,\ y)$ ter translaciji T in rotaciji R pripadajoče palice. Hitrosti žogice v stanje nismo vključevali neposredno z namenom boljšega prenosa na fizično napravo. Vključili smo jo implicitno, s pomnjenjem zadnjih nekaj preteklih pozicij žogice. Akcije strategij smo modelirali kot ciljno translacijo in rotacijo pripadajoče palice, saj to zahteva dejanski sistem namiznega nogometa. Grafičen prikaz interakcije agentov z okoljem prikazuje slika 4.

Za namene učenja strategij smo predhodno izdelali nasprotnika. Ta je implementiran kot končni avtomat (angl. *state machine*). Z vključitvijo nasprotnika smo želeli doseči strategiji, ki bi se poleg branjenja neposrednih strelov naučili tudi držanja dobrega obrambnega položaja.

Obeh strategij nismo učili hkrati. Najprej smo učili vratarja in nato še obrambo. Pri učenju obrambe je strategija vratarja delovala kot del okolja in se ni učila. Učenje strategije vratarja je trajalo 10 milijonov korakov (55,5 ur simuliranega časa), prav tako je toliko korakov trajalo učenje strategije obrambe.

3 Rezultati

Rezultati vključujejo vrednotenje sposobnosti branjenja znotraj simuliranega okolja, kjer sta strategiji branili strele nasprotnika. Dodatno smo strategiji preizkusili v pravem okolju, kjer smo opazovali le zadete in prejete gole.

Uspešnost branjenja v simuliranem okolju prikazuje tabela 1. Uspešnost branjenja u je definirana z enačbo (1), kjer $n_{obramba}$ predstavlja število ubranjenih strelov, n_{prejet} pa število prejetih golov. Pri tem smo strele kot ubranjene šteli le, če bi ti brez posredovanja strategij vodili v prejem gola.

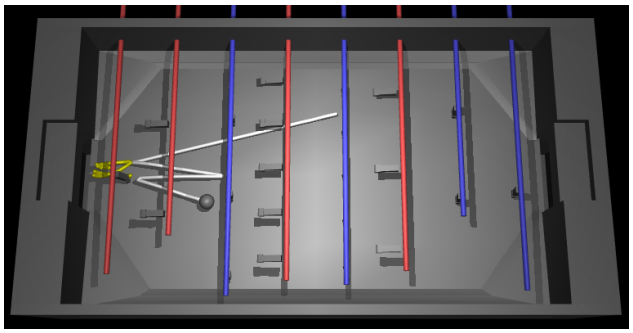
$$u = \frac{n_{obramba}}{n_{obramba} + n_{prejet}} \tag{1}$$

V idealnem simulacijskem okolju brez časovnih zakasnitev že vratar sam, brez prisotnosti obrambe, ubrani skoraj vse strele. Primer branjenja je prikazan na sliki 5. Pomanjkljivost se kaže predvsem po vključitvi časovne zakasnitve stanja, ki ustreza pravem okolju, ko uspešnost precej upade (približno 10 % slabše branjenje). Ob vključitvi obrambe k vratarju se uspešnost še dodatno izboljša. Prav tako je opazen manjši upad uspešnosti ob vključitvi časovnega zamika, kar izhaja predvsem iz boljše pokritosti gola in dobre obrambne drže igralcev.

Zaradi nezmožnosti avtomatičnega zaznavanja trkov, vrednotenja ubranjenih strelov v pravem okolju nismo izvedli. Vrednotenje na pravi mizi namiznega nogometa je bilo izvedeno na podlagi prejetih in zadetih golov, kjer sta učeni strategiji delovali vzporedno z agentom vezne vrste in agentom napada. Slednja sta implementirana s končnim avtomatom. Slika 6 prikazuje porazdelitev razlik med zadetimi in prejetimi goli znotraj

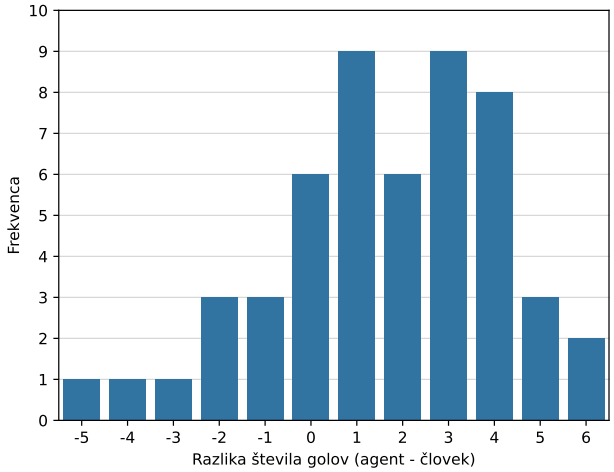
Pogoji	Uspešnost [%]	
	Vratar	Vratar in obramba
Brez zakasnitve	98.043 ± 0.097	99.246 ± 0.018
Z zakasnitvijo (40 ms)	87.782 ± 0.017	94.655 ± 0.129

Tabela 1: Uspešnost branjenja v simuliranem okolju proti agentu izdelanim s končnim avtomatom. Rezultati so predstavljen v obliki $\mu \pm \sigma$, kjer μ pomeni povprečje in σ negotovost treh zaporednih vrednotenj. Vsako vrednotenje je trajalo 5000 epizod po največ 500 korakov.



Slika 5: Primer delovanja učenega vratarja v simuliranem okolju. Rumena sled označuje vratarjevo gibanje, bela pa pot žogice.

posamezne igre, kjer je vsaka igra trajala največ dve minuti. Na fizični mizi je bilo odigranih 52 iger. Rezultati na sliki 6 potrjujejo, da razviti strategiji delujeta dobro tudi na pravi fizični mizi.



Slika 6: Porazdelitev razlik med zadetimi in prejetimi goli iger na fizični mizi.

4 Zaključek

Z delovanjem obrambnih strategij smo zadovoljni, saj strategiji skupaj ubranita precejšen delež strelov. V prihodnosti nameravamo izdelati tudi strategijo napada, s čimer želimo še dodatno izboljšati celotno gručo agentov.

Literatura

[1] M. Bošnak in G. Klančar. Fast and reliable alternative to encoder-based measurements of multiple 2-dof rotary-linear transformable objects using a network of image sensors with application to table football. *Sensors*, zv. 20, št. 12, 2020.

[2] OpenAI. Spinning up in deep RL. `https://spinningup.openai.com/en/latest/spinningup/rl_intro.html`, 2018. Dostopano: 6. marec 2025.

[3] R. S. Sutton in A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2. izd., 2018.

[4] E. Todorov, T. Erez in Y. Tassa. Mujoco: A physics engine for model-based control. V *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, str. 5026–5033. IEEE, 2012.

[5] E. Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. V *2014 IEEE International Conference on Robotics and Automation (ICRA)*, str. 6054–6061. 2014.

[6] J. Mockus, V. Tiesis in A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, zv. 2, str. 117–129, 1978.

[7] M. Towers, A. Kwiatkowski in sod. Gymnasium: A standard interface for reinforcement learning environments, 2024.

[8] J. Schulman, F. Wolski in sod. Proximal policy optimization algorithms, 2017.