



Budapesti Műszaki és Gazdaságtudományi Egyetem
Természettudományi Kar

Megerősítéses Tanulás Alapú Drift Mozgásszabályozó Rendszer Önvezető Járműhöz

TDK DOLGOZAT
Tóth Szilárd Hunor

Témavezetők:

Dr. Bárdos Ádám

Gépjárműtechnológia Tanszék, Budapesti Műszaki és
Gazdaságtudományi Egyetem (BME)

Dr. Viharos Zsolt János

Számítástechnikai és Automatizálási Kutatóintézet (SZTAKI),
Mérnöki és Üzleti Intelligencia Kutatólaboratórium (EMI),
Intelligens Folyamatok Kutatócsoport (IP)



Budapest, 2021

Kivonat

Az önvezető járművekkel kapcsolatos kutatások az utóbbi években egyre népszerűbbekké váltak. A mesterséges intelligencia technikáival kombinálva, az alkalmazások pozitív eredményeinek jelenlegi tendenciája miatt ez jó terület a kutatásra, ugyanakkor még számos kutatási kihívás van ezen a területen. Az adattudomány területén a megerősítéssel tanulás hatékony megközelítésnek bizonyult a mozgásszabályozási problémák kezelésében, ideértve a különféle robotokat, az önműködő rendszereket és a videojáték-AI-okat (például a sakkot, a GO-t stb.) [23].

A driftelés egy olyan vezetési technika, amikor a sofőr szándékosan túlkormányoz, miközben fenntartja a kanyarodás irányát. A mozgás során a hátsó kerekek megcsúsznak (a gumikra ható erők szaturálnak), miközben az első kerekek a kanyarodás ívével ellentétes irányba mutatnak. Ez egy átlagos sofőr számára nagy kihívást jelentő irányítási technika, ami nagyon fontossá és érdekessé teszi a mesterséges intelligencia és az autonóm szabályozás segítségével történő megvalósítást.

A jelenleg legkorszerűbb vezetésbiztonsági rendszerek (ABS, ESL,...) elsősorban arra fókuszálnak, hogy az autó lehetőleg minél inkább elkerülje a vezetési szempontból instabil állapotokat (pl. driftelés), amelyeket a legtöbb hétköznapi sofőr nehezen képes kontrollálni. Viszont a valóságban ezekben az instabil állapotokban rengeteg a potenciál balesetmegelőzési szempontból, ha képesek lennénk azokat megfelelően szabályozni valamilyen önvezető technika segítségével.

Ezen TDK dolgozat célja, hogy új eredményeket mutasson fel a megerősítéssel tanulás alapú drift mozgásszabályozás kutatásának területén. A feladat egy önvezető ágens kifejlesztése a drift mozgásszabályozáshoz szimulációkon keresztül, elsősorban MATLAB/Simulink környezetben. Alkalmazva a legmodernebb 'state of the art' módszereket a megerősítéssel tanulás területéről, azt szeretnénk látni, hogy az önvezető jármű képes legyen belépni egy adott céldriften állapotba, majd sikeresen képes legyen azt fenntartani.

Az ezen a területen bemutatott friss tudományos eredmények azt mutatták, hogy valamilyen (emberi járművezető képességein vagy akár egyéb kontrol technikákkal végzett szimulációkon alapuló) előzetes 'prior' adat alkalmazása a feladathoz elkerülhetetlen a jó eredmény eléréséhez. Ezért a kutatás másodlagos feladata az, hogy ilyen típusú adatok felhasználását nélkülözve is képesek legyünk elérni a kívánt eredményt.

Köszönetnyilvánítás

A publikációban szereplő kutatást, amelyet Számítástechnikai és Automatizálási Kutatóintézet valósított meg, az Innovációs és Technológiai Minisztérium és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatta az Autonóm Rendszerek Nemzeti Laboratórium keretében.

A kutatást az "Kutatások az ipari digitalizáció által nyújtott potenciál minőségi kiaknázására" című ED_18-2-2018-0006 támogatás tette lehetővé.

A jelen cikkben szereplő kutatást (részben) az European Commission támogatta a H2020 EPIC (<https://www.centre-epic.eu/>) projekt 739592. számú pályázaton keresztül.

Tartalomjegyzék

1. Bevezető	5
2. Az alkalmazott járműmodell leírása	7
2.1. Egy nyomvonalú járműmodell	7
2.2. Gumimodellek	9
2.3. A driftelés definíciója és egyensúlyi pontjainak kiszámítása	11
3. Megerősítéses tanulás	13
3.1. A megerősítéses tanulás elemei	14
3.2. Actor-Critic módszerek	17
3.3. A Soft Actor-Critic (SAC) algoritmus	19
4. Driftelés autonóm járművekkel	22
4.1. Kontroll alapú megközelítések	22
4.2. A felügyelt tanulás alkalmazhatósága	23
4.3. Megerősítéses tanulás alapú megközelítések	24
5. A feladat megfogalmazása	25
5.1. A driftelés definiálása megerősítéses tanulás feladatként	25
5.2. A járműmodell implementációja MATLAB/Simulink környezetben	28
6. Kísérletek és eredmények	31
6.1. A célállapotból indított helyzet megtartása	31
6.2. A célállapot megközelítése és megtartása kanyarodásból	34
7. Következtetések	39

1. fejezet

Bevezető

Az autonóm járművek kutatása és fejlesztése az elmúlt években jelentős teret nyert a tudományos világban, és jelenleg is nagy hangsúlyt kap. A területnek különböző aspektusai léteznek, melyeken folyamatosan zajlik a kutatás. A környezetfelismerés (detekció) foglalkozik az autó környezetének értelmezésével és feltérképezésével, mint például a többi közlekedési résztvevő típusának és helyzetének megállapításával, illetve akadályok és közlekedési jelzések érzékelésével. A mozgástervezés területe a jármű által követendő nyomvonal adaptív megtervezésére, a megfelelő elkerülő manőver megállapítására fókuszál. A járműszabályzás (kontrol) pedig a szükséges manőverek helyes végrehajtásáért, és a megadott nyomvonal pontos lekövetéséért felelős. [8]

A szabályzás területén a hátsókerék-hajtású járművekkel való driftelés komoly kihívást és fontosságot jelent, tekintettel a lehetséges alkalmazásokra vonatkozó potenciáljára. A driftelés egy olyan kanyarodó mozgást jelent, amikor a vezető folyamatosan ellenkormányoz, hogy fenntartsa egy nagy oldalcsúszási szöget, általában nagy sebességnél. Ez egy eredendően instabil mozgás, amelyet a legtöbb hétköznapi járművezető nem tud irányítani, és általában a jármű irányíthatatlan kipördüléséhez vezet. Ez a manőver leginkább az autósportokban (rally, Formula-D, stb.) látható, de ennek a mozgásnak a közúti alkalmazása is jelentős potenciált rejt magában.

Létezik számos olyan kritikus helyzet, amelyben egy drift manőver lehetővé tenné egy esetleges baleset elkerülését. Esetleges csúszós útfelületek, hirtelen az autó elé kerülő akadályok és egyéb külső erőhatások mind olyan instabil állapotba képesek kényszeríteni a járművet, amelyet egy jól kifinomult drift manőverrel még kontrollálni lehet. Azonban az átlagos vezető legtöbbször nem rendelkezik a megfelelő képességekkel, hogy ezeket a manővereket megfelelően végrehajtsa, emiatt az ilyen szituációk jelentős számban balesethez vezetnek. Egy 2015-ös tanulmány[19] szerint az Egyesült Államokban az irányítás elvesztésével összefüggő balesetek relatív gyakorisága 8,32% volt, ami a 2013-as GES (Általános Becslések Rendszere) baleseti jelentései alapján összesen 458 000 balesetet jelentett. Ez jelentős szám, amelyet csökkenteni lehetne olyan önvezető keretrendszerek

alkalmazásával, amelyek képesek fenntartani és szabályozni az instabil mozgásokat a manőverezhetőségi határokon. Mindezek mellett az automatizált járművek alkalmazása a motorsportokban[30] szintén fontos kutatási lehetőséget biztosít.

Megelőző kutatási eredmények az autonóm driftelés megvalósítására azt mutatják, hogy érdemes további munkát folytatni a területen, ugyanis még nincsenek teljesen kiforrott rendszerek. A klasszikus kontrol módszerei segítségével születtek már sikeres szimulációs illetve valós járművön elért eredmények is stabil állapotú driftelés megvalósítására és megtartására [2, 3]. A feladatot továbbá megfogalmazták már nyovonal-követésként is, pl. a driftelve parkolás műveletének elvégzésére [17]. A kontrol alapú módszerek mellett a felügyelt és a megerősítéses tanulás alkalmazása is megjelent, főképp a változó környezethez alkalmazkodó adaptív irányításban rejlő lehetőségek miatt [5, 4].

Ezen TDK dolgozatban bemutatott kutatás célja egy megerősítéses tanulás segítségével fejlesztett ágens megtervezése és megalkotása szimulációs környezetben, amely képes egy hátsókerék meghajtású járművet egy megadott drift állapot felé irányítani, majd ezt az állapotot tetszőleges ideig megtartani. A feladat megoldásához egy leegyszerűsített kétkerekű járműmodell került implementálásra MATLAB/Simulink környezetben, amelyet egy Soft Actor-Critic algoritmussal tanított ágens irányít. A tanításhoz semmilyen előzetes adat nem kerül alkalmazásra, kizárólag a járműmodellből kinyerhető információval történik a tanítás.

A következő fejezetben kerül sorra a járműmodell és az egyensúlyi állapotok egyenletalapú kiszámítási módszerének leírása. Azt követően a megerősítéses tanulásról és a Soft Actor-Critic algoritmus működéséről szóló fejezet kerül majd tárgyalásra. A 4. fejezetben az autonóm drifteléssel kapcsolatos megelőző irodalom részletesebb bemutatása található. Az 5. fejezetben a drift feladat pontos leírása olvasható. A dolgozat a kísérletek és az eredmények tárgyalását követően a konklúziókkal zárul.

2. fejezet

Az alkalmazott járműmodell leírása

Az alkalmazott járműmodell helyes megválasztása kritikus faktora a kontrol feladat megoldásához. Általánosságban elmondható, hogy a driftelés megvalósításához egy olyan jármű szükséges, amely elegendő nagy hátsó tapadóerővel rendelkezik. Ezek jellemzően többségében hátsókerék-meghajtású (RWD) járművek, ezért a választott modell is hátsókerék-meghajtást alkalmaz.

A szimulációhoz használt modell egy egypályás dinamikus modell két különböző gumiabroncs modellel. Ennek a modellnek az előnye az egyszerűsége: figyelmen kívül hagyja a dőlés dinamikákat és az aerodinamikát, amelyek sokkal kevésbé fontosak a driftelés megvalósításának szempontjából, mint az abroncsmodell, így az utóbbi pontosságára helyeződhet a hangsúly.

2.1. Egy nyomvonalú járműmodell

Az viszonylag elhanyagolható dőlés és pattogás dinamikák miatt érdemes volt síkbeli járműmodellt választani. Ezek a modellek 3 szabadsági fokkal rendelkeznek: transzláció x és y irányban, és elforgatás a z tengely körül. A jármű Newton–Euler mozgásegyenletei a karosszéria koordinátarendszerében a következőképpen írhatóak le [16]:

$$F_x = m\dot{v}_x - mr\dot{v}_y \quad (2.1)$$

$$F_y = m\dot{v}_y + mr\dot{v}_x \quad (2.2)$$

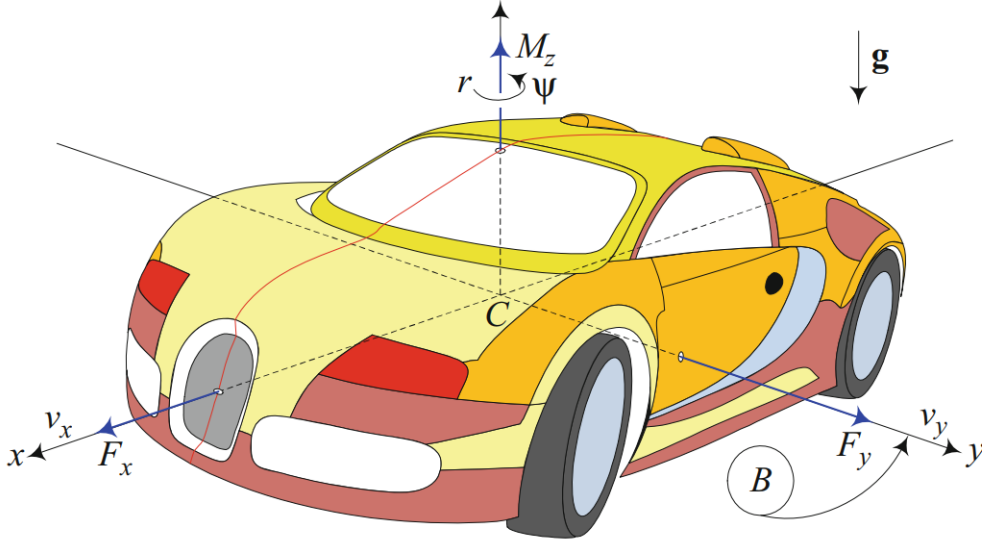
$$M_z = I_z \dot{r} \quad (2.3)$$

ahol v_x és v_y rendre az x és y irányú sebességek, r a jármű z tengely körüli forgásának sebessége, m a jármű össztömege és I_z az inercia állandója. Ez alapján a deriváltak kifejezhetők:

$$\dot{v}_x = \frac{1}{m}F_x + rv_y \quad (2.4)$$

$$\dot{v}_y = \frac{1}{m}F_y - rv_x \quad (2.5)$$

$$\dot{r} = \frac{1}{I_z}M_z \quad (2.6)$$



2.1. ábra. A síkbeli dinamikus járműmodell karosszériájának koordinátarendszere, dinamikus változói és erőrendszere[16]

A fenti erő komponenseket továbbá levezethetjük a kerekre ható erők összehatásaként is (levezetéshez lásd [16], 125-131 és 145-160 old.):

$$F_x = F_{x_f} \cos \delta + F_{x_r} - F_{y_f} \sin \delta \quad (2.7)$$

$$F_y = F_{y_f} \cos \delta + F_{y_r} - F_{x_f} \sin \delta \quad (2.8)$$

$$M_z = aF_{y_f} \cos \delta + aF_{x_f} \sin \delta - bF_{y_r} \quad (2.9)$$

Az egyszerűség kedvéért csupán egyetlen első és egyetlen hátsó kerék kerül modellezésre a későbbi egyensúlyi pont kalkulációk bonyolultságának csökkentésére való tekintettel. Mivel a választott jármű hátsókerék-meghajtású, ezért a kormányzott első kerékre ható hosszirányú erő nulla: $F_{x_f} = 0$. Ezen felül a hátsó hosszirányú erő a járművet vezérlő ágens egyik műveletparamétere lesz, tehát a jármű modell egyik bemeneti paramétere, így kiszámításának leírása nem szükséges. Emellett a δ kerékszög szintén bemeneti paraméternek lett kiválasztva. A többi fent látható erő komponens az abroncsmodellből származtatható.

2.2. Gumimodellek

Az gumimodell pontos és értelmes definiálása a legfontosabb szempont a driftelés pontos szimulációs megvalósításához. A fő szempont, hogy az abroncsokra ható oldalirányú erők szaturációja jól leírható legyen, ugyanis ez esetben valósul meg a gumik oldalirányú sodródása, ami a driftelés egyik alapvető eleme. Vannak olyan modellezési megközelítések, amelyek a gumiabroncsok viselkedésének összetettebb aspektusait igyekeznek megragadni, beleértve például az adatvezérelt empirikus modelleket, mint a Pacejka-féle „Magic Tire” formula [22]. A driftelés szimulálásának szempontjából azonban ezek a technikák túlzottan bonyolultak.

A kefe gumiabroncs modell (brush tire model) [13] három alépítményre osztja az abroncsot: a gyűrűre, a vázra és a rugalmas „kefekre”, amelyek az úttal érintkező futófelület-elemeket képviselik. Ebben a modellben az abroncs által generált nettó oldalirányú erőt két tényező kombinációja határozza meg: az abroncsból kanyarodáskor eredő erő és a súrlódásból származó teljes erő. A kanyarodáshoz szükséges erő a gumiabroncs csúszási szögének eredménye, amely megadja a gumiabroncs sebességvektorának a szögét a gumiabroncs hossztengetelyéhez képest. Ezek a következőképpen írhatóak le [15]:

$$\alpha_f = \arctan\left(\frac{v_y + ar}{v_x}\right) - \delta \quad (2.10)$$

$$\alpha_r = \arctan\left(\frac{v_y - br}{v_x}\right) \quad (2.11)$$

ahol a és b rendre a jármű féltengely távolságai. Az oldalirányú súrlódásból származó rész a μ súrlódási együttható és a kerekekre ható F_z normálerők segítségével írhatóak le:

$$F_{z_f} = \frac{m \cdot g \cdot b}{a + b} \quad (2.12)$$

$$F_{z_r} = \frac{m \cdot g \cdot a}{a + b} \quad (2.13)$$

Ha az érintkezési útvonalat egy adott csúszási szögnél az érintkezési folt hosszában integráljuk, a következő kifejezést kaphatjuk a gumiabroncs oldalirányú erejére a gumiabroncs csúszási szögének függvényében:

$$F_{y_f} = \begin{cases} -C_\alpha \tan \alpha_f + \frac{C_\alpha^2}{3\mu F_{z_f}} |\tan \alpha_f| \tan \alpha_f - \frac{C_\alpha^3}{27\mu^2 F_{z_f}^2} \tan^3 \alpha_f & |\alpha_f| \leq \alpha_{sl_f} \\ -\mu F_{z_f} \operatorname{sgn} \alpha_f & |\alpha_f| > \alpha_{sl_f} \end{cases} \quad (2.14)$$

ahol C_α a gumi kanyarmerevségi állandója, μF_{z_f} maximális elérhető oldalerő és α_{sl_f} a

legkisebb nagyságú csúszási szög, amelynél az oldalerő szaturáció bekövetkezik:

$$\alpha_{sl_f} = \arctan \frac{3\mu F_{z_f}}{C_\alpha} \quad (2.15)$$

A fenti modell ugyan nem veszi figyelembe a hosszirányú csúszást, de hátsókerék-meghajtás esetén így alkalmazható a kormányzott első kerékre. A hátsó meghajtott kerék esetében azonban számolni kell nem csak az oldalirányú csúszással, hanem a hosszirányú csúszással is, így egy kombinált csúszást szükséges figyelembe venni. Ez egy lényegesen bonyolultabb konstrukciót kíván, ami szükségessé teszi a kerékforgás-dinamika hozzáadását a teljes járműmodellhez. Ezt elkerülendő, Hindiyeh [15] egy egyszerű megoldást javasol olyan esetekre, ahol a kerekekre ható hosszirányú erőt bemeneti paraméterként kezeljük, ami így pontosan alkalmazható az itteni szimulációkhoz:

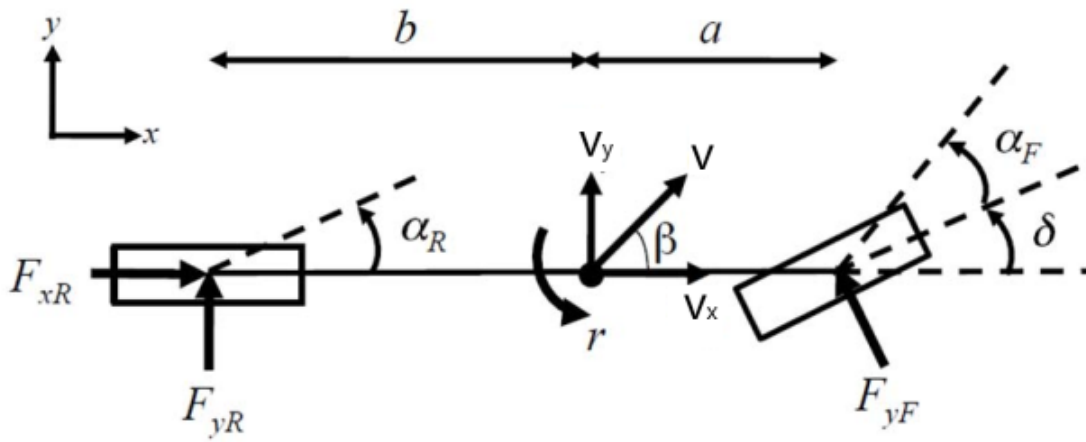
$$\xi = \frac{\sqrt{(\mu F_{z_r})^2 - F_{x_r}^2}}{\mu F_{z_r}} \quad (2.16)$$

$$\alpha_{sl_r} = \arctan \frac{3\xi\mu F_{z_r}}{C_\alpha} \quad (2.17)$$

$$F_{y_r} = \begin{cases} -C_\alpha \tan \alpha_r + \frac{C_\alpha^2}{3\xi\mu F_{z_r}} |\tan \alpha_r| \tan \alpha_r - \frac{C_\alpha^3}{27\xi^2\mu^2 F_{z_r}^2} \tan^3 \alpha_r & |\alpha_r| \leq \alpha_{sl_r} \\ -\xi\mu F_{z_r} \operatorname{sgn} \alpha_r & |\alpha_r| > \alpha_{sl_r} \end{cases} \quad (2.18)$$

ahol ξ egy olyan tényező, ami megadja a maximális elérhető oldalirányú erőt adott F_{x_r} bemenetre.

Részletes levezetésekért és bizonyításokért lásd [15], 21-34. oldal.



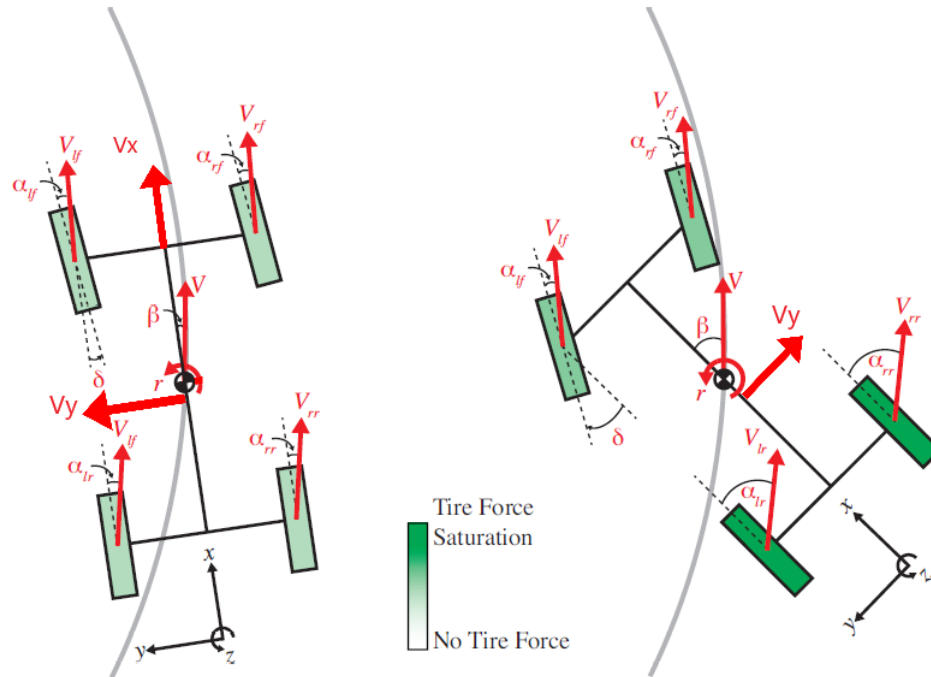
2.2. ábra. Az abroncsokra ható erőkomponensek és csúszásszögek ábrázolása a jármű koordinátarendszerében.[14]

2.3. A driftelés definíciója és egyensúlyi pontjainak kiszámítása

A driftelés egy olyan kanyarodó mozgásfolyamat, amely során folyamatos ellenkormányzás közben a hátsó kerekekre ható oldalirányú erők szaturálnak, emiatt a jármű hátsó része a kanyar külső ívére sodródik, miközben az eleje bent marad. Ez formálisan úgy jellemezhető, hogy a karosszéria csúszásszöge β nagy, és előjele ellentétes a kanyarodás szögének irányával. A β szög a következőképpen definiálható:

$$\beta = \arctan \left(\frac{v_y}{v_x} \right) \quad (2.19)$$

Driftelő mozgás előidézésének egy lehetséges folyamata a következő. A szokványos kanyarodás során a hátsó abroncsokra ható erők nem szaturálnak, és a kormányzás iránya a kanyar csúcspontja felé mutat. A driftelő állapotba lépéshez a vezetőnek elég nagy nyomatékot kell adnia a hátsó kerekek számára, hogy növelje a hátsó abroncs csúszási szögét, így az autó hátsó vége kifelé sodródhat a kanyarodás ívéről. Ezért fontos a hátsókerék-meghajtású járművek használata, mert ezek közvetlenül növelhetik a hátsó kerekekre ható hosszirányú erőket. Az autó előzőekből adódó instabil állapotának szabályozásához a vezetőnek ellenkormányoznia kell, hogy kompenzálja a magas csúszási szöget, így az autó képes fenntartani a kanyarodást. Máskülönben a jármű kipörögne.



2.3. ábra. Szokványos kanyarodás (bal oldalon) és driftelés (jobb oldalon) különbségének vizualizációja.[15]

A jelen dolgozatban kitűzött cél egyik aspektusa egy adott driftelési állapot stabil megtartása, ezért szükséges definiálni, hogy a fent ismertetett járműmodell esetében mely állapotokat nevezhetjük fenntartott driftelésnek. Megelőző tudományos munkák[14] azt mutatják, hogy a driftelés egyensúlyi pontjai kiszámolhatóak egy, a járműmodellből kapott algebrai egyenletrendszer segítségével. Megelőző eredmények a kontroll területén[29][2][3] mutatják, hogy ez a - jelen kutatásban is alkalmazott - konstrukció helyes és jól működik szimulációban és a valóságban is.

Mivel egy állapot folyamatos fenntartása azt jelenti, hogy a jármű helyzetét befolyásoló tényezők, azaz a v_x , v_y és r változók végig azonosak maradnak, tehát a deriváltak nullák, így ez a feltétel a következőképpen írható le:

$$\dot{v}_x = \frac{1}{m}F_x + rv_y = 0 \quad (2.20)$$

$$\dot{v}_y = \frac{1}{m}F_y - rv_x = 0 \quad (2.21)$$

$$M_z = I_z * \dot{r} = 0 \quad (2.22)$$

A fenti egyenletekben a már definiált gumimodell segítségével az F_x és F_y modellek kifejezhetők, így egy 5 $(v_x, v_y, r, F_{x_r}, \delta)$ ismeretlenből álló algebrai egyenletrendszert kapunk. Fontos, hogy a hátsó kerék szaturáltsága meg legyen kötve, ezért a (2.18) egyenlet esetében az $|\alpha_r| > \alpha_{slr}$ feltétel megadása is szükséges. Ez a rendszer aluldefiniált (3 feltétel és 5 ismeretlen), így a megoldáshoz legalább két kezdőérték definiálása szükséges. Jelen esetben a v_x sebesség és a δ egy ésszerű választás, mivel ezek könnyen befolyásolhatóak a bemeneti paraméterek segítségével, valamint v_x a gyakorlatban jobban értelmezhető, mint F_{x_r} .

Az általunk kiválasztott és kiszámított cél egyensúlyi állapot értékeihez lásd a 5.1 fejezetet.

3. fejezet

Megerősítéses tanulás

Ahogy Richard S. Sutton és Andrew G. Barto könyve [24] definiálja, a megerősítéses tanulás a célorientált tanulás és döntéshozatal automatizálásának és megértésének számítógépes megközelítése. A tanulás alapvető része egy olyan ágens alkalmazása, amely közvetlen kölcsönhatásban van a környezetével, anélkül, hogy felügyeletre vagy a környezet teljes modelljére támaszkodna a feladat végrehajtásához.

A megerősítéses tanulás különbözik a felügyelt tanulástól. Felügyelt tanulás esetén rendelkezésre áll egy címkézett rekordokból álló adatkészlet, amelyet valami vagy valaki (a „felügyelő”) rendelkezésre bocsát. Ennek a fajta tanulásnak a célja egy olyan modell készítése, amely képes általánosítani a válaszait úgy, hogy helyesen járjon el olyan helyzetekben, amelyek nincsenek jelen a megadott adatkészletben. Interaktív problémák esetén gyakran nem túl praktikus megoldás példákat szerezni a kívánt viselkedésre amelyek minden olyan helyzetet reprezentálnak, amelyben az ágensnek cselekednie kell. Az ágensnek tanulnia kell a saját tapasztalataiból azáltal, hogy kölcsönhatásba lép a környezettel, még akkor is, ha eleinte nem a megfelelő döntést hozza.

A megerősítéses tanulás különbözik a felügyelet nélküli tanulástól is, amely jellemzően rejtett struktúrák megtalálására fókuszál egy adott címkézetlen adathalmazban. Az ágens tapasztalataiban rejtett struktúrák feltárása minden bizonnyal hasznos lehet az interaktív tanulás szempontjából, de önmagában nem oldja meg az ágens jutalom maximalizálási problémáját. Tehát bizonyos értelemben a megerősítéses tanulás tekinthető a gépi tanulás harmadik paradigmájaként.

A megerősítéses tanulás problémakörébe tartoznak például a következők:

- A sakkot tekinthetjük megerősítéses tanulás feladatként, amelyben az ágensnek a figurák táblán elfoglalt helyzete alapján kell cselekednie. A cél az ellenfél játékos legyőzése. Az ágens megtanulja elérni ezt a célt, ha többszörösen játszik egy emberi ellenféllel, vagy saját magával. [23]

- Tekintsünk egy mobil tisztító robotot. A robot célja tekintélyes mennyiségű szemet összegyűjtése emberi beavatkozás nélkül. Ez magában foglalja annak eldöntését, hogy be kell-e lépnie egy helyiségbe, ahol további szemetet kell gyűjtenie, vagy megpróbálja megtalálni az utat az akkumulátor-töltőállomáshoz. Dönthet az akkumulátor aktuális töltöttségi szintje, a következő helyiség tisztításához szükséges becsült energia és/vagy az akkumulátorállomás jelenlegi helyzetéből való eléréséhez szükséges idő alapján. [24]

3.1. A megerősítéses tanulás elemei

A megerősítéses tanulás formális keretrendszer használ, amely meghatározza a tanuló ágens és környezete közötti interakciót állapotok, cselekvések (akciók) és jutalmak formájában. Ennek a modellnek az a célja, hogy egyszerű módja legyen az olyan alapvető jellemzők megjelenítésének, mint az ok-okozati összefüggések, a bizonytalanság, valamint az explicit célok megléte.

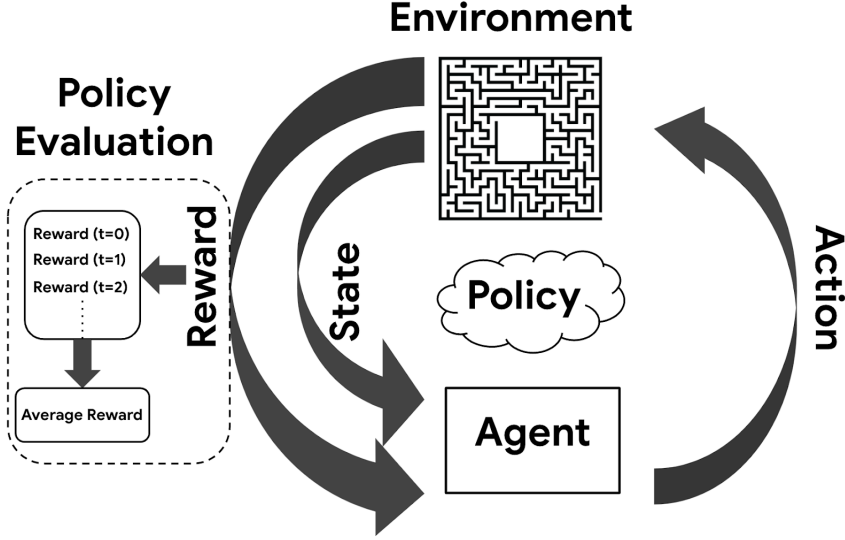
A tanulót és a döntéshozót *ágensnek* nevezzük. Az ágens kölcsönhatásba lép a *környezettel*, amely minden olyan objektumot tartalmaz, amely hatással van a döntéshozatalra. Ez akár tartalmazhat olyan információt is az ügynökről, amely azt fizikailag jellemzi, például az akkumulátor töltöttségi szintjét a tisztító robot példájában.

A környezet és az ágens folyamatos interakcióban van egymással, az ágens kiválaszt egy *akciót*, amelyre a környezet új helyzetet, *állapotot* mutat be az ágensnek. Formálisan megfogalmazva, minden $t = 0, 1, 2, \dots$ időpillanatra egy adott $S_t \in \mathcal{S}$ állapotra az ágens egy $A_t \in \mathcal{A}(S_t)$ akcióval reagál, ahol $\mathcal{S}$ az állapotok és $\mathcal{A}(S_t)$ az S_t állapotban elérhető akciók halmaza. Az akció következtében az ágens egy $R_{t+1} \in \mathcal{P} \subset \mathbb{R}$ numerikus jutalmat kap visszajelzésül, és a következő S_{t+1} helyzetbe kerül, ahol a folyamat újra megismétlődik. A folyamathoz a 3.1 ábra mutat illusztrációt.

Az ágens egy $\pi_t : \mathcal{S} \rightarrow [0, 1]$ leképezést alkalmaz az akció kiválasztására, amelyet *politikának* (angolul „policy”) nevezünk. Tehát $\pi_t(a|s)$ a valószínűség, hogy adott s állapotban az ágens az a akciót választja. Az ágens célja, hogy maximalizálja a hosszú távon kapott jutalom teljes vagy diszkuntált összegét, a *megtérülést* (angolul „return”) ($0 < \gamma \leq 1$):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (3.1)$$

A megerősítéses tanulás egyik legfontosabb kérdése a felfedezés megválasztása. Ahhoz, hogy hosszú távon képes legyen az ágens maximalizálni a megtérülést, fel kell fedeznie, hogy mely akciók mekkora jutalomhoz vezetnek az adott állapotban. A *kikukázás* azt jelenti, hogy a jelenlegi politika alapján történik az akció megválasztása, a *felfedezés* pedig az, amikor valamely más akció történik kiválasztásra, még akkor is,



3.1. ábra. Az ágens és a környezet interakciója [24]

ha ez bizonyos azonnali jutalom feláldozását jelenti. Ha az ágens mindig ragaszkodik a kiáknázáshoz, akkor megkockáztatja, hogy értékes információkat hanyagol el a meg nem látogatott állapotokról, ezáltal soha nem éri el a lehető legnagyobb megtérülést. A felfedezés során az ágens azt tapasztalhatja, hogy az aktuális optimális művelet valójában nem optimális, ezért ennek megfelelően módosíthatja a politikáját. Ez egy (még akkor) szuboptimális akció kiválasztásával történik, ami valójában magasabb megtérülést ad, mint a korábban várt, sőt, magasabbat, mint a korábban feltételezett optimális művelet. Számos algoritmus használ különböző módszereket a felfedezés szabályozására, illetve léteznek adaptív felfedezést nyújtó algoritmusok is, amelyek a tanulás során rendszeresen változtatják a felfedezés mértékét. [27]

Szinte minden megerősítéses tanulási algoritmus magában foglalja valamilyen *értékfüggvény*(ek) becslését. Ezek olyan állapotok (vagy állapot-művelet párok) függvényei, amelyek megbecsülik, hogy az ágensnek mennyire jó egy adott állapotban lenni (és a következő lépésben végrehajtani az adott műveletet), azáltal, hogy megmutatja az abban az állapotban várható megtérülést. A π állapotérték függvény $v_\pi : \mathcal{S} \rightarrow \mathbb{R}$ egy számértéket rendel hozzá minden olyan állapothoz, amely az ágens tanulása során előfordulhat:

$$v_\pi(s) = \mathbb{E}_\pi [G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right] \quad (3.2)$$

tehát a várható visszatérés értékét jelzi az adott állapotban. Hasonlóképpen, egy állapot-akció értékfüggvény $q_\pi : (\mathcal{S}, \mathcal{A}) \rightarrow \mathbb{R}$ minden (értelmes) állapot-művelet párhoz számértéket ad, így megmondja, melyik műveletet mennyire jó az aktuális állapotban:

$$q_\pi(s, a) = \mathbb{E}_\pi [G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \quad (3.3)$$

Ezen értékfüggvények valamelyikével meghatározható a legjobb politika a megerősítéses tanulás feladat megoldására. Egy π politikát akkor és csak akkor definiálunk legalább olyan jónak mint egymásik π' politika, azaz $\pi \geq \pi'$, ha $v_\pi(s) \geq v_{\pi'}(s)$ teljesül minden $s \in \mathcal{S}$ esetén. Továbbá mindig létezik legalább egy π^* politika, amely legalább olyan jó, mint az összes többi politika, amelyet *optimális politikának* neveznek, és ugyanaz az *optimális értékfüggvényük*:

$$v_*(s) = \max_{\pi} v_\pi(s) \quad \text{vagy} \quad q_*(s, a) = \max_{\pi} q_\pi(s, a) \quad (3.4)$$

Tehát az optimális politika az, amely az optimális értékfüggvény alapján választ akciót az adott állapotban.

Megemlítes képpen az *értékfüggvény-alapú* algoritmusok mellett léteznek továbbá *politika-alapú* és *modell-alapú* megerősítéses tanulás algoritmusok is. Az utóbbiakkal szemben az értékfüggvény-alapú módszereknek az előnye, hogy képes állapotok (állapot-akció párok) értékét becsülni más becslések alapján. Ezt a tulajdonságot *Bellman-tulajdonságnak*, vagy másképp *bootstrap-tulajdonságnak* nevezik. Ezen tulajdonságot használják ki az úgynevezett *időbeli differencia* (temporal difference, TD) tanulóalgoritmusok, melyeknek heurisztikus lényege, hogy ne csupán a végső interakciót követően történjen meg a tanulás, hanem minden n -edik interakciót után, így az interakció közben az ágens folyamatosan képes tanulni.

*Epizód*nak nevezik azokat a T hosszú ($T \geq 1$) interakció sorozatokat, amelyek valamilyen szempontból homogénnek tekinthetőek (pl. sakkban egy teljes játék). Tehát az epizód alatti diszkuntált megtérülés a (3.1) egyenlet T -ig szummázva. Ezzel ellentétben a TD algoritmusok úgynevezett *n -lépésbeli megtérülést* (n-step return) alkalmaznak:

$$G_t^n(S_t) = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n v_\pi(S_{t+n}) \quad (3.5)$$

ahol az utolsó tag a bootstrap. Tehát az értékfüggvény frissítése a következő:

$$V(S_t) \leftarrow V(S_t) + \alpha [G_t^n - V(S_t)] \quad (3.6)$$

ahol α a tanulórata és $\delta_t = G_t^n - V(S_t)$ az úgynevezett *TD-hiba*. A fentiek ugyanígy felírhatóak állapot-akció értékfüggvényekre is:

$$G_t^n(S_t, A_t) = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n q_\pi(S_{t+n}, A_{t+n}) \quad (3.7)$$

$$q(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha [G_t^n - q(S_t, A_t)] \quad (3.8)$$

A TD algoritmusok egyik nagy előnye, hogy a bootstrap-tulajdonságnak köszönhetően jól képesek kezelni folytonos időben zajló feladatokat, amelyek közé sorolható a driftelés is.

Összefoglalva, a megerősítéses tanulás feladata tehát az optimális politika megtalálása. Az értékfüggvény-alapú módszerek ezt egy értékfüggvény kiszámításával érik el, amely alapján frissítik az aktuális politikát. Ennek folyamata függ attól, hogy milyen számosságú állapot-, illetve akciótérrel rendelkezik az adott feladat. Mivel a driftelés minden tekintetben egy folytonos feladat, ezért a következőkben az utóbbi módszerek kerülnek tárgyalásra.

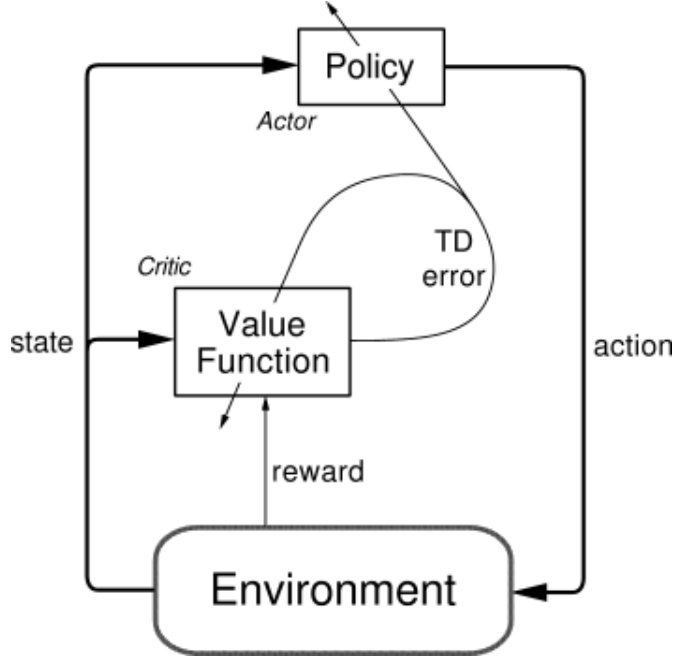
3.2. Actor-Critic módszerek

Az értékfüggvény-alapú algoritmusok egyik legfontosabb részlete az értékfüggvény kiszámításának vagy becslésének a módja. Diszkrét állapot és akcióterek esetén például úgynevezett Q-táblák segítségével megállapítható, hogy egy adott állapotban mely akciók milyen értékkel bírnak. A tábla minden sora egy állapotot, minden oszlopa egy akciót reprezentál. Egy cella eleme az adott állapotban az adott akció értéke ($q_\pi(s, a)$). A tanulás kezdetén a tábla elemei tetszőleges kezdőértékkel megválasztódnak, majd ezek az értékek frissülnek a TD-módszer alapján (3.8). Az akciók értékének a felfedezése valamilyen felfedezés-kiaknázás módszerrel történik, pl. valamilyen $1 > \varepsilon > 0$ valószínűséggel a „legjobb” akciót kerül kiválasztásra (az adott sorban a legnagyobb értékhez tartozó oszlop indexét), vagy $1 - \varepsilon$ valószínűséggel egy egyenletesen választott véletlen akciót alkalmaz az ágens (felfedez). Ez a módszer az úgynevezett *Q-tanulás* (Q-learning).

Folytonos állapot- és akcióterek esetén azonban végtelen sok sorra és oszlopra lenne szükség ezen módszer alkalmazásához, ami a gyakorlatban nem megvalósítható. Ezért az ilyen esetekben az értékfüggvény valamilyen becslésére van szükség. Formálisan, a feladat a $\Pi = \{\pi_\theta | \theta \in \Theta\}$ paraméterezett politikák terén maximalizálni a $q_\pi(s, a)$ értékfüggvényt, azaz az optimális politika megtalálása:

$$\pi^* = \operatorname{argmax}_{\pi_\theta \in \Pi} q_{\pi_\theta}(s, a) \quad \forall s \in \mathcal{S} \forall a \in \mathcal{A} \quad (3.9)$$

Az approximáció lehet lineáris, valamilyen feature selection algoritmus használatával [28], vagy nemlineáris, mesterséges neurális hálózatok segítségével, amit *mély megerősítéses tanulásnak* (deep reinforcement learning) neveznek (a mély gépi tanulás után). A driftelés feladatának megoldásához valamilyen nemlineáris módszer alkalmazása célszerűbb, ugyanis az akciótér két dimenziós lesz.



3.2. ábra. Az Actor-Critic keretrendszer.

Az Actor-Critic módszerek olyan nemlineáris közelítésen alapuló megoldások, amelyek egyszerre közelítik az optimális politikát és az adott jelenlegi politikát legjobban kiértékelő értékfüggvényt. Két legfontosabb alkotórésze a cselekvő vagy aktor (actor) és a kritikus (critic), melyeket neurális hálókkal reprezentálnak. A cselekvő feladata a döntéshozatal, azaz a megfelelő akció kiválasztása az adott állapotra (tehát a politikát reprezentálja). A kritikus (átvitt értelemben az értékfüggvény) feladata egy kritika megfogalmazása, amely kiértékeli az aktor teljesítményét, és irányítja a hálók tanítását. A kritika jellemzően a TD-hiba $\delta_t = G_t^n - q(S_t, A_t)$. A keretrendszer illusztrációja a 3.2 ábrán látható.

Formálisan, tekintsük a már előbb definiált Π politika parametrizációt, továbbá definiáljuk a $\mathbf{Q}_{\mathbf{w}} := \{q_{\mathbf{w}} | \mathbf{w} \in \mathbf{W}\}$ értékfüggvény parametrizációt is. Ekkor az aktor feladata a következő célfüggvény maximalizálása θ -ban rögzített $\mathbf{w}$ -re:

$$J(\theta) = [G_t^n - q_{\mathbf{w}}(S_t, A_t)] \log \pi_{\theta}(A_t | S_t) \quad (3.10)$$

ahol a logaritmus függvény a természetes alapú logaritmust jelenti. Ennek a tagnak a szerepe, hogy biztosítsa bizonyos gradiens számítási problémák elkerülését. Röviden, $J(\theta) = \mathbb{E}[R(S, A)] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$, aminek a deriváltja nem függ θ -tól, ezért a gradiens 0 lenne. Így a $\nabla_{\theta} \mathbb{E}[R(S, A)] = \mathbb{E}[\nabla_{\theta} \log \pi(A|S) R(S, A)]$ bizonyítható összefüggés kerül alkalmazásra.

Folytatva, a kritikus feladata pedig a várt és a valós megtérülés eltérésének minimalizálása rögzített θ -ra, azaz

$$I(\mathbf{w}) = \mathbb{E}_{\pi_{\theta}} [(q_{\pi_{\theta}}(S, A) - q_{\mathbf{w}}(S, A))^2] \quad (3.11)$$

Többféle actor-critic módszer létezik, és a fenti általános konvenciók valamelyest eltérhetnek az actor-critic algoritmus típusától függően. A továbbiakban a jelen kutatásban alkalmazott actor-critic algoritmus ismertetése következik.

3.3. A Soft Actor-Critic (SAC) algoritmus

Az egyik legjelentősebb elsők között implementált actor-critic algoritmus a DDPG (Deep Deterministic Policy Gradients) [20]. Az algoritmus egyetlen aktort és egyetlen kritikust alkalmaz az optimális politika megtalálásához, emellett egy normális eloszláson alapuló zaj modellt, amely irányítja a felfedezést. Az algoritmus gyakori hibája, hogy túlbecsli az értékfüggvényt, ezáltal szuboptimális megoldásokat eredményezhet a tanulás. Ennek a kiküszöbölésére született a TD3 (Twin Delayed Deep Deterministic) [9] algoritmus, amely már két kritikust használ és a kritikusok által adott értékek közül a kisebb alapján számítódik ki a kritika (TD-hiba). Emellett mindkét kritikus mellett célkritikust (target critic) is alkalmaz, amelynek lényege a reziduális hiba csökkentése, ezáltal csökkenti a varianciát.

A TD3 algoritmus továbbra is normális eloszlás alapú zaj modellt alkalmaz a felfedezés megvalósítására. Formálisan, az ágens által leadott valós akció $A'_t = A_t + N_t$, ahol $N_t \sim \mathcal{N}(\mu, \sigma^2)$ egymástól független változók valamilyen (μ, σ^2) választott paraméterre. A paraméterek dinamikus és értelmes megválasztása azonban nehéz és erősen feladat specifikus, továbbá elegendhetlen a jó felfedezéshez, így a helyes konvergenciához.

A sztochasztikus aktor fogalmának bevezetése ezen zajmodell megoldás kiváltásával van összefüggésben. Egy sztochasztikus aktor feladata nem az akció közvetlen megadása, hanem egy (μ, σ^2) paraméterpár optimális megválasztása, amely alapján az akció a következő: $A_t \sim \mathcal{N}(\mu, \sigma^2)$. Tehát a politika ebben az esetben egy $\pi_{(\mu, \sigma^2)} : \mathcal{S} \rightarrow \Theta$ függvény, ahol $\Theta = \{\theta = (\mu, \sigma^2)\}$ paraméterhalmaz. Ennek alkalmazásával elérhető egy ún. belső felfedezési funkció, amely az $\mathcal{N}(\mu, \sigma^2)$ normális eloszlás differenciál entropiájával mérhető, azaz $\mathcal{H}_{\pi_{\theta}}(S_t) = \mathcal{H}_{(\mu, \sigma^2)} = \ln(\sqrt{2\pi e}\sigma)$. Tehát, minnél nagyobb az entrópia, annál magasabb a felfedezés mértéke, azaz annál nagyobb a valószínűsége, hogy az optimálisnak vélt μ akciótól minnél inkább különböző akció kerül kiválasztásra.

A felfedezés mértékének ideális megválasztásához módosul az optimális politika célfüggvénye. Az ágens ebben az esetben egyszerre igyekszik maximalizálni a várható megtérülést és a várható entrópia értékét is. Ezt nevezik *maximum entrópia célfüggvénynek* (maximum entropy objective) [31]:

$$\pi^* = \operatorname{argmax}_{\pi_\theta \in \Pi} \sum_t \mathbb{E}_{(S_t, A_t)} [r(S_t, A_t) + \alpha_H \mathcal{H}_{\pi_\theta}] \quad (3.12)$$

ahol $\alpha_H \geq 0$ az entrópia súlya. Tehát $\alpha_H = 0$ esetben visszkapjuk az eredeti célfüggvényt.

A Soft Actor-Critic algoritmus (SAC) [11] ötvözi a sztochasztikus aktor felfedező tulajdonságát a TD3 algoritmussal, tehát egy belső felfedezés funkcióval rendelkezik, miközben elkerüli az értékfüggvény felülbecslését. A cél a maximum entrópia feladat (3.12) megoldása. A tanítás egy *mini-batch* és egy *tapasztalat buffer* segítségével történik. Az implementált algoritmus lépéseinek leírása a következő:

1. Definiáljuk $\mathcal{S}$ és $\mathcal{A}$ folytonos állapot és akcióttereket. Inicializáljuk θ , $\mathbf{w}_1$, $\mathbf{w}_2$ kezdeti aktor és kritikus paramétereket (azaz a neurális hálók súlyait) a π , Q_1 , Q_2 neurális hálókhoz és a kiritikusokhoz tartozó Q_{τ_1} , Q_{τ_2} célhálókhoz (a kezdő súlyok rendre azonosak), az S_0 kezdőállapotot és egy üres $\mathbf{E}$ tapasztalati buffert. Ezt követően minden $T > t \geq 0$ időlépésben:
2. Az adott S_t állapotban az aktor választ egy A_t akciót a jelenlegi θ_t paramétere alapján.
3. Az ágens válaszként megkapja a környezettől az R_{t+1} jutalmat és S_{t+1} állapotot, majd a $(S_t, A_t, R_{t+1}, S_{t+1})$ tapasztalatot eltárolja az $\mathbf{E}$ bufferben.
4. Vegyünk egy M elemű mini-batch mintát a bufferből $((S_t^{(i)}, A_t^{(i)}, R_{t+1}^{(i)}, S_{t+1}^{(i)}), i = 1, \dots, M)$
5. Frissítsük a $\mathbf{w}_j$ ($j = 1, 2$) kritikusok súlyait a következő célfüggvény minimalizálásával:

$$L_{\mathbf{w}_j} = \frac{1}{M} \sum_{i=1}^M \left(y_i - Q \left(S_t^{(i)}, A_t^{(i)} | \mathbf{w}_j \right) \right)^2 \quad (3.13)$$

ahol y_i az értékfüggvény cél (target), ami $R_{t+1}^{(i)}$, ha $t + 1 = T$, különben

$$y_i = R_{t+1}^{(i)} + \gamma * \min_j \left(\nabla_{\mathbf{w}_{\tau_j}} Q_{\tau_j} \left(S_{t+1}^{(i)}, A_t^{(i)} | \mathbf{w}_{\tau_j} \right) \right) - \alpha_H \ln \left(\mathcal{H}_{\pi_{\theta_t}}(S_t^{(i)}) \right) \quad (3.14)$$

ahol γ a diszkuntálási faktor ($0 < \gamma \leq 1$).

6. Frissítsük a θ aktor súlyait a következő célfüggvény minimalizálásával:

$$J_\theta = \frac{1}{M} \sum_{i=1}^M \left(-\min_j \left(\nabla_{\mathbf{w}_{\tau_j}} Q_{\tau_j} \left(S_t^{(i)}, A_t^{(i)} | \mathbf{w}_{\tau_j} \right) \right) - \alpha_H \ln \left(\mathcal{H}_{\pi_{\theta_t}}(S_t^{(i)}) \right) \right) \quad (3.15)$$

7. Frissítsük a célhálók paramétereit: $\mathbf{w}_{\tau_j} = \tau \mathbf{w}_j + (1 - \tau) \mathbf{w}_{\tau_j}$, ahol τ a *célsimítás faktor* (target smoothing factor).

A felfedezést meghatározó entrópia súly α_H egy nehezen hangolható hiperparaméter, ezért a szerzők egy későbbi munkájukban [12] bevezettek egy α_H önszabályozó módszert, amely a tanítás során egy megadott H_t várható célentrópia értékhez igazítja az aktor jelenlegi entrópiáját. Így az algoritmus egy plusz lépéssel bővül:

- Frissítsük az entrópia súlyát a következő célfüggvény minimalizálásával:

$$K_{\alpha_H} = \frac{1}{M} \sum_{i=1}^M \left(-\alpha_H \ln \left(\mathcal{H}_{\pi_{\theta_t}}(S_t^{(i)}) \right) - \alpha_H H_T \right) \quad (3.16)$$

A hiperparaméterek hangolásáról és megválasztásáról az 5.1 fejezetben lesz szó.

4. fejezet

Driftelés autonóm járművekkel

Az autonóm járművekkel történő driftelés megvalósítására fordított kutatások már az egyensúlyi pontok vizsgálatát követően megkezdődtek. A driftelés feladatát többféleképpen is definiálták: lehet egy kijelölt drift állapot megtartása a cél (mint jelen kutatás esetében), vagy valamilyen nyomvonalkövetési feladat reprezentálja a driftelés megvalósítását. Az első megoldások között többnyire a klasszikus kontroll módszerei jelentek meg, eleinte szimulációs környezetben alkalmazva, majd kisebb távirányítású kocsikon és valós járműveken is. Ezt követően megjelentek a felügyelt tanulás által irányított megoldások, és a természetesen adódó megerősítéses tanulás alapú módszerek is. Egyes ezen kutatások és eredményeik kerülnek ismertetésre ebben a fejezetben.

4.1. Kontroll alapú megközelítések

Drift egyensúlyi pont stabilizációjához több sikeres szimulációs eredmény született már kontroll módszerek segítségével. Elsősorban érdemes kiemelni a lineáris kontrollerek kétkerekű járműmodellen történt alkalmazásait, melyek sikeresen kihasználták az egyensúlyi pontok modellen alapuló kiszámítását. A megválasztott bemeneti paraméterek tekintetében egyes munkákban elkülönült a hosszirányú és a laterális mozgás alkalmazása két külön controller alkalmazásával [29], más esetekben egyetlen közös controller alkalmazása is sikeres volt [2], de ezek mellett előfordult a kormányzás rögzítése mellett csupán a gázadás és fékezés irányítása [25]. Ezen megoldások a valós járműveken történő alkalmazáshoz is jónak bizonyultak [29][3]. Mindezek mellett négykerekű modell felhasználásával is születtek pozitív szimulációs eredmények [26].

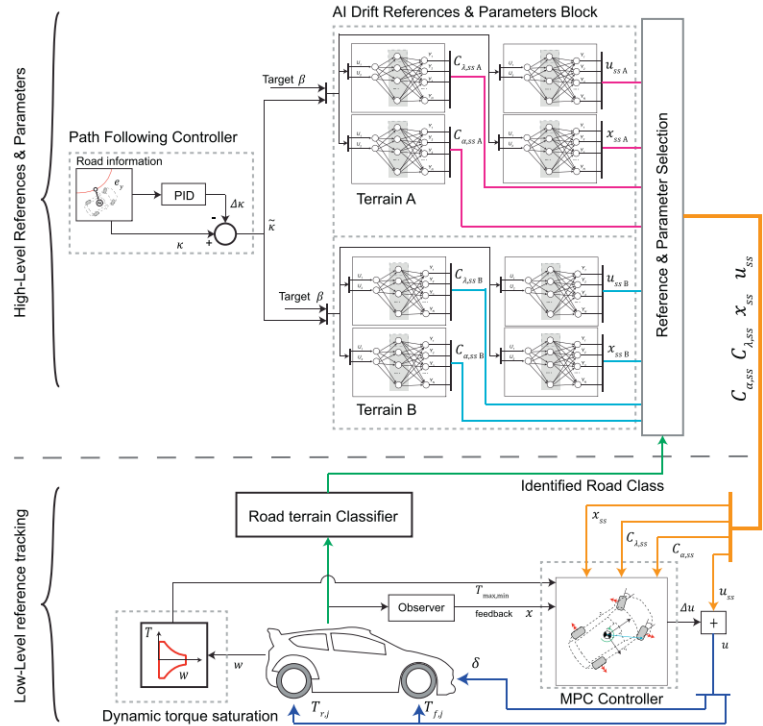
Az MPC (Modell Predictive Control) controller szimulációs alkalmazása drift stabilizációs feladatokhoz eddig szintén sikeresnek bizonyult [6], illetve változó útviszonyok szimulálásakor is helyt állt, jelezve ezzel egy jó adaptív tulajdonságot. Nyomvonalkövető feladatok közül érdemes megemlíteni a lineáris kontroll és az MPC egy hibrid alkalmazását driftelve elvégzendő parkoló manőver megvalósításához [17]. Az előbbi konstrukció

szimulációban és távirányítású autóval egyaránt működött.

4.2. A felügyelt tanulás alkalmazhatósága

A előbb említett kotrol segítségével megvalósított drifteléshez kapcsolódó eredményekhez a járműmodell és abroncsmodell pontos definiálása elengedhetetlen volt. Ennek a szükséges feltételnek a hátránya, hogy ilyen módon nehezen alkotható olyan ágens, amely egyszerűen alkalmazható lenne több különböző jármű esetén is. Pusztán a felügyelt tanulás alkalmazása a mozgásszabályozás megvalósításához nem lenne praktikus a sikeres tanításhoz összegyűjtendő adat nagy mennyisége miatt, azonban modell szempontból adaptív tulajdonság definiálásához már történt megelőző alkalmazása.

Acosta és Kanarachos cikkükben [1] egy MPC-ből és egy előrecsatolt neurális hálózatból álló hibrid vezérlőt javasoltak a driftelés megoldására, melyet szimulációban teszteltek különböző útviszonyok esetén. Két neurális háló komponenst használtak. Az első egy drift bemeneti előrejelző, amely a drifteléshez szükséges referenciákat és a gumiabroncs-paramétereket szolgáltatja az MPC-nek egy adott cél oldalcsúszási szög és útgörbület bemenetre. A másik egy úttep osztályozó, amely a referenciákat és a gumi-paramétereket a különböző útfelületekhez igazítja, ahol más tapadási tényezők lépnek fel. A hálózatokat egy emberi sofőr szimulátorban való vezetés közben felvett adatai segítségével tanították.



4.1. ábra. Neurális hálók alkalmazásával működő drift kontrollor[1]

A fent említett példa mellett születtek még további hasonló munkák, pl. [18]. Az ilyen módszereknek a hátránya viszont, hogy emberi sofőr vagy egy előzetesen jól működő kontroller által rögzített adat felhasználása mindenképpen szükséges a hálók tanításához. Többek között ennek a problémának a megoldására is jól használható lehet a megerősítéssel tanulás alkalmazása.

4.3. Megerősítéssel tanulás alapú megközelítések

Ígéretes irányt mutat a megerősítéssel tanulási módszerek alkalmazása is az autonóm driftelés megoldására. A megerősítéssel tanulás potenciális előnyei közé tartozik elsősorban a jobb általánosító képesség folyamatosan változó vezetési körülmények (például útfelületek) között, illetve a tanításhoz szükséges előzetes adat felhasználásának nélkülözhetősége.

Cutler és How [5] a Probabilistic Inference for Learning Control (PILCO) [7] modellalapú politika-kereső algoritmust használta a stabil állapotú driftelés eléréséhez szimulációban és egy kis méretű távirányítós járművön. A [4] cikkben a driftelést egy nyomvonalkövetési feladatként határozták meg a CARLA nevezetű szimulátorban, ahol a pontos cél nagy oldalcsúszási szögek elérése volt nagy sebességeknél. Az ágens több előre megadott pályán tanították, majd egy addig számára ismeretlen pályán kiértékeltek, több különböző típusú járművel és útviszony beállítással. A tanítás a - már a dolgozatban is részletezett - Soft Actor-Critic algoritmussal történt, emellett más szerzők egy ehhez hasonló munkájában [21] a TD3 algoritmus is jól működőnek bizonyult. Az eredmények jól mutatják a módszerek kedvező adaptív tulajdonságait.

A fenti munkák mindegyikénél - bár eredményeik jelentősek - beépítettek valamilyen előzetes tudást az ágens tanítása során: a PILCO számára egyszerű modellek segítségével és klasszikus kontroll technikák alkalmazásával kiinduló politikákat generáltak a tanulás inicializálására, és a CARLA-ban a jármű által követendő trajektóriákat egy emberi sofőr segítségével rögzítették. Ez jelzi azt a tudományos potenciált, hogy hogyan lehet a megerősítéssel tanulást előzetes beállítás vagy tudás nélkül alkalmazni az adott jármű komplex drift állapotának szabályozására és elindítására.

5. fejezet

A feladat megfogalmazása

A járműmodell és a megerősítéses tanulás alkalmazásához szükséges ismeretek ismertetése után immáron sor kerülhet a megoldandó drift feladat definiálására. A cél továbbra is egy megadott drift egyensúlyi állapot megközelítése és megtartása tetszőleges időn keresztül. Ennek szimulációs megvalósításához definiálni kell a tanulóalgoritmus paramétereit és implementálni a kiválasztott járműmodellt MATLAB/Simulink környezetben, majd összekötni a két környezetet egymással.

5.1. A driftelés definiálása megerősítéses tanulás feladatként

A feladathoz kiválasztott tanulóalgoritmus a Soft Actor-Critic algoritmus, amely kizárólag folytonos állapot- és akciótereken értelmezett, így ezek megválasztásakor erre ügyelni kell. A járműmodell 3 szabadsági fokkal rendelkezik, és ezek jelenlegi értékeinek ismeretében teljes egészében megállapítható az autó jelenlegi mozgásállapota. Ez azt jelenti, hogy az ágens számára elegendő információt biztosítanak a driftelés megtanulásához. Tehát az állapottér a következőnek lett kiválasztva:

$$\mathcal{S} := \{S \in \mathbb{R}^3 | S = (v_x, v_y, r)\} \quad (5.1)$$

Az akcióteret a járműmodell bemeneti paramétereit szerint kell definiálni, amelyek az F_{x_r} és a δ . Egy sofőr által vezérelhető kezelőfelület a jármű irányításához a pedálokból és a kormányból tevődik össze, ezért a bemeneti paraméterek ezekre lesznek visszavezetve, hogy az ágens egy emberi vezetőhöz hasonló módon tudja irányítani az autót. Az F_{x_r} értéke a gázpedál állásának megváltoztatásával szabályozható, aminek definíciója legyen $ped_{acc} \in [0, 1]$, tehát $ped_{acc} = 0$ a gázpedál teljes elengedése és $ped_{acc} = 1$ pedig a padlógáz. Ahhoz, hogy ez paraméterként alkalmazható legyen, szükség van valamilyen hajtáslánc modellre, amely a pedál állását motornyomatékká, majd keréknyomatékká

alakítja, amiből megkapható F_{x_r} . Ennek ismertetése a 5.2 szekcióban olvasható.

A kormánykerék állására a $\delta_{steer} \in [-420^\circ, 420^\circ]$ definiálható, ahol a pozitív értékek a bal oldali tartományt, a negatív értékek pedig a jobb oldali tartományt reprezentálják. Tehát pl. $\delta_{steer} = 90^\circ$ azt jelenti, hogy a kormánykerék 90 fokkal balra van forgatva. A kormánysszög kerékszögé váló átalakítása a következő:

$$\delta = \frac{\delta_{steer}}{15} \cdot \frac{\pi}{180} \quad (5.2)$$

ahol δ mértékegysége radián. Tehát az ágens akciótere a következőnek lett megválasztva:

$$\mathcal{A} := \{A = (ped_{acc}, \delta_{steer}) \mid ped_{acc} \in [0, 1], \delta_{steer} \in [-420^\circ, 420^\circ]\} \quad (5.3)$$

A megfelelő jutalomfüggvény definiálása kritikus a sikeres konvergenciához. Ha rossz jutalmat kap az ágens, akkor az vagy szuboptimális megoldáshoz vezet, vagy egy teljesen más feladat megoldásához. Mivel a feladat egy célállapot megtartása, ezért annak mindeképpen szerepelnie kell a jutalom értékében. A célállapotot a 2.3 szekcióban leírtak alapján lehet kiválasztani. Ehhez a $\delta = -10 \text{ rad}$ és $v_x = 10 \text{ m/s}$ értékek kerültek kiválasztásra, és ebből a $v_y = -3.4812 \text{ m/s}$, $r = 0.8334 \text{ rad/s}$ és $F_{x_r} = 3747.8719 \text{ N}$ eredményt kaptuk. Tehát a drift célállapot:

$$S_{drift} = (v_{x_{drift}}, v_{y_{drift}}, r_{drift}) = (10 \text{ m/s}, -3.4812 \text{ m/s}, r = 0.8334 \text{ rad/s}) \quad (5.4)$$

Ebből a jutalomfüggvény a következőként lett meghatározva:

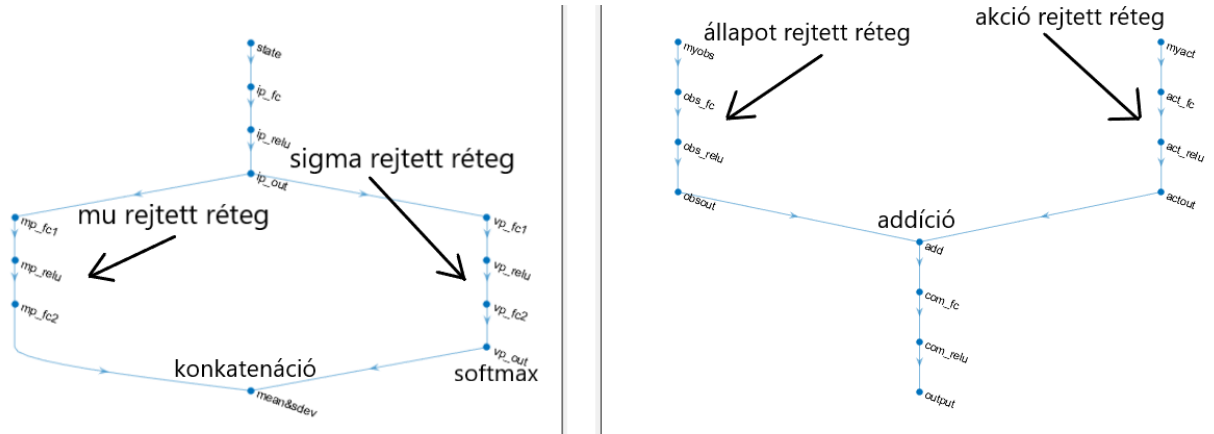
$$r(S_t, A_t) = R_{t+1} = -\frac{1}{3} \sqrt{\sum_{i=1}^3 \left(\frac{S_{t+1_i}}{S_{drift_i}} - 1 \right)^2} \quad (5.5)$$

ami a következő állapot és a célállapot euklideszi távolságának $-\frac{1}{3}$ -szorosa, ahol $A_t = \pi_{\theta_t}(S_t)$. Tehát az ágens azzal arányos „büntetést” kap, amilyen messze van a munkaponttól.

A Soft Actor-Critic algoritmushoz továbbá definiálni kell az aktor és a kritikusok neurális hálóját, valamint a szükséges hiperparamétereket. A hálók két rejtett réteggel rendelkeznek és minden rétegben 256 perceptron található. A struktúráik úgy lettek kiépítve, hogy alkalmasak legyenek a kimeneti paraméterek megfelelő kiválasztására. Az aktor háló adott állapot bemenetre külön „ösvényen” számolja a μ és σ eloszlásparamétereket. A σ prediktálásához továbbá egy softmax [10] rétegre is szükség van, mivel $\sigma > 0$ feltételt teljesíteni kell. A kritikus hálóban ugyanúgy két „ösvény” található, de ezek a bemeneti paraméterekre vonatkoznak: először egy külön majd egy közös „ösvényen” keresztül haladnak a bemeneti értékek.

Hiperparaméter	Érték
Optimalizáló	Adam
Aktivációs függvény	ReLU
Aktor és kritikus tanulórátája	0.001
Entrópia tanulórátája	0.0003
Célentrópia	-2
Kezdeti entrópia súly	0.2
Tapasztalati buffer mérete	10000
Minibatch mérete	32
Az ágens beavatkozási időköze	0.1s

5.1. táblázat. A tanításhoz megválasztott hiperparaméterek értékei a legjobb eredmények esetében



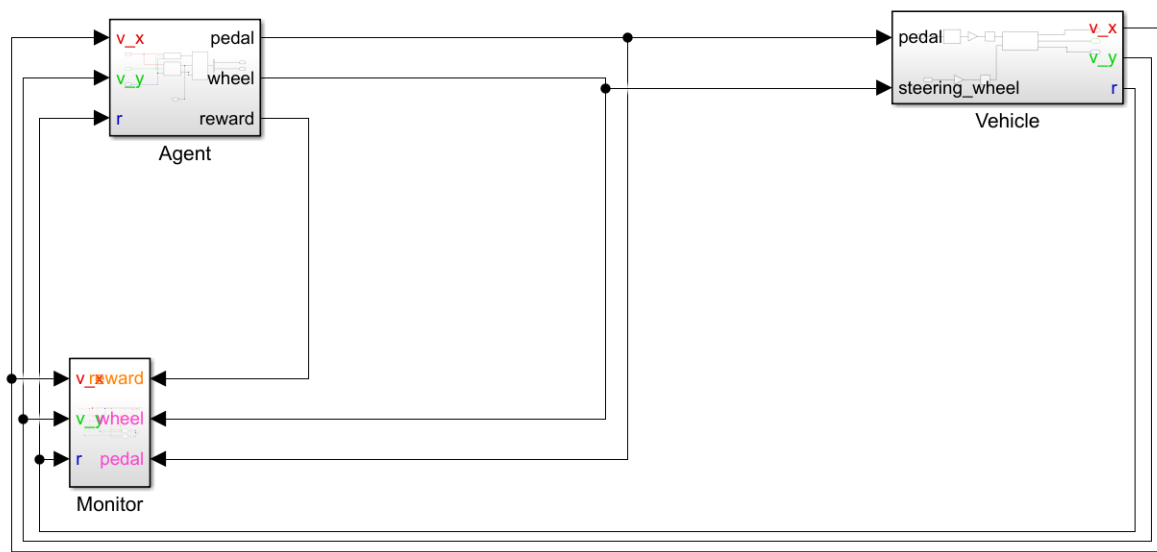
5.1. ábra. Az aktor háló (bal oldalon) és a kritikus hálók (jobb oldalon) struktúrái

Az algoritmus hiperparamétereinek első megválasztásakor a [12]-ben megadott baseline értékek kerültek alkalmazásra, ezt követően kerültek kipróbálásra különböző paraméterértékek. A legérzékenyebbnek az entrópia beállítások bizonyultak, illetve a buffer és a batch mérete. Történtek tesztelések fix és adaptív entrópia súlyozással is. Fix entrópia esetén nem sikerült konvergenciát elérni. Az entrópia súlyának adaptív változtatása sikeres eredményekhez vezetett, habár az algoritmus instabilitást mutat: magasabb jutalom értékek elérését követően egy idő után (az entrópia növekedésével) csökken az ágens teljesítménye, „elfelejti” a magasabb jutalmat eredményező akciókat. Ezt az instabilitást tanulási leállási feltételekkel sikerült kezelni: ha az elmúlt 20-50 epizód visszatéréseinek átlaga magasabb volt, mint egy megadott felső korlát, akkor leállt a tanítás.

5.2. A járműmodell implementációja MATLAB/Simulink környezetben

Ebben a szekcióban a Simulink modell részletei kerülnek bemutatásra.

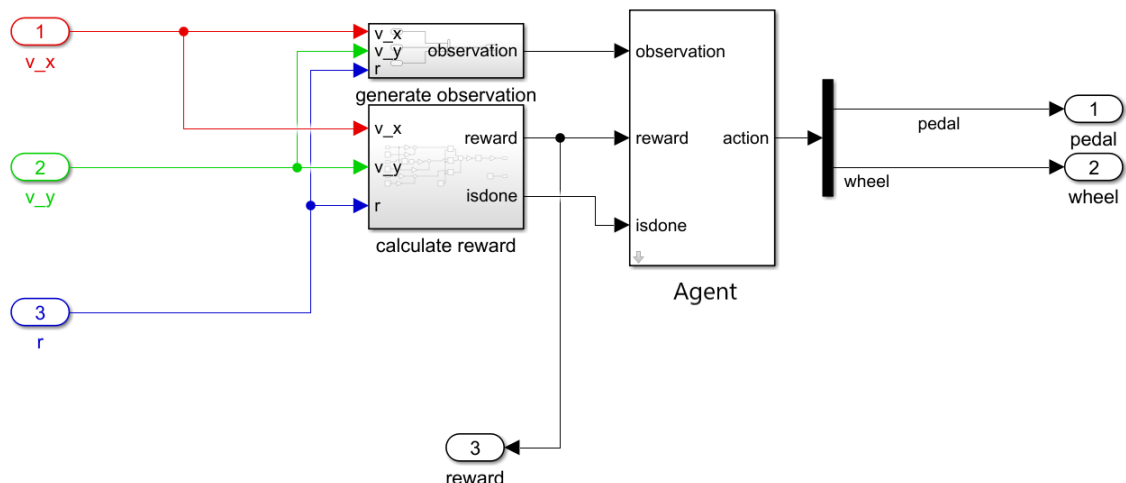
A modell különböző szintekből és alrendszeréből épül fel. A legfelső szinten három megkülönböztetett alrendszerünk van: az ágens („Agent”), a járműmodell („Vehicle”) és a figyelő rendszerek („Monitor”). Ezek ebben a sorrendben kerülnek ismertetésre.



5.2. ábra. A Simulink modell felső szintje. Az irányított vonalak a bemenet/kimenet kapcsolatokat jelentik. Például a v_x jel a jármű rendszer kimenete, valamint a „Monitor” és az ágens rendszerek bemenete. Ez az ábra jól szemlélteti a megerősítéses tanulás zárt hurok jellegét (nincs alap bemeneti vagy kimeneti port).

Az „Agent” rendszer tartalmazza az ágens működéséhez szükséges blokkokat. Ez maga az ágenst reprezentáló blokk, a jutalmat kiszámító alrendszer és a megfigyeléseket előállító rendszer, amely csak jeltípus-konverziós célokat szolgál. A rendszer bemenetei az állapotparaméterek.

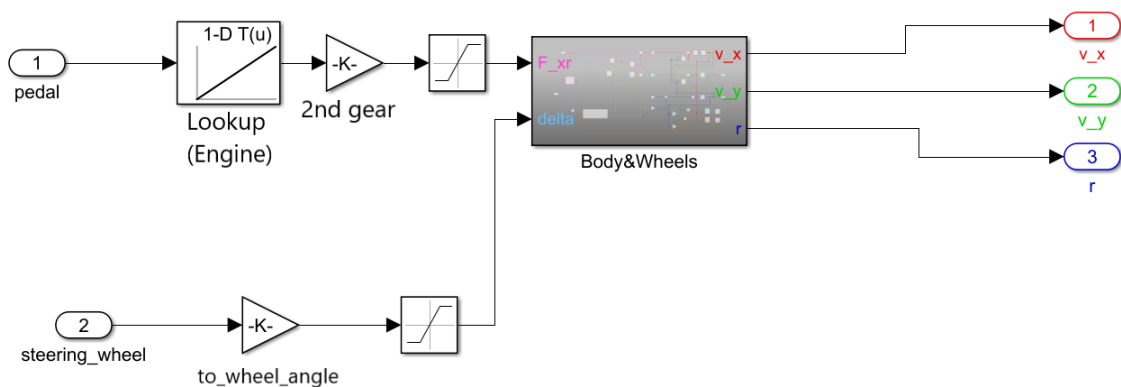
A „generate observation” alrendszer egy vektort hoz létre a változókból. A „calculate reward” rendszer kiszámítja a jutalmat az adott állapot bemenetre (lásd az 5.5 egyenletet), majd egy „isdone” jellel együtt lép ki, amely mindig „hamis” (azaz egy epizód csak akkor áll le, ha $t = T$). Az ágens (a legnagyobb blokk) bemenetei az állapotvektor, a jutalom és az 'isdone' jel, ezeket feldolgozva kimenetként adja az akciót, ami elhagyja a rendszert.



5.3. ábra. Az ágens rendszere a Simulink modellben.

A „Vehicle” rendszer tartalmaz mindent, ami az autó működéséhez szükséges. A beérkező akció paraméterek szükséges átkonvertálása történik először. A pedáljel egy legyszerűsített hajtáslánc modellen halad keresztül, amely egy lookup táblázatból és egy szorzótényezőből tevődik össze. A lookup táblázat elsőfokú lineáris interpolációs polinomként működik, amely átalakítja a ped_{acc} bemenetet motornyomatékká, amely 0 és 550 Nm között mozog. Ezután feltételezve, hogy második sebességfokozatban van a jármű, értelmesen megadott áttételi arányok [’itt mire hivatkozzak?’] segítségével kiszámítjuk a hátsó kerekekre kifejtett hosszirányú erőt F_{xr} . Ezeken felül a δ_{steer} kormányyszög bemenet is átkonvertálódik δ kerékszögre.

Az átkonvertált input paraméterek a „Body&Wheels” alrendszerbe lépnek, amely reprezentálja a 2.1 és 2.2 alfejezetekben definiált járműmodellt, aminek a kimeneti paraméterei az állapotváltozók v_x , v_y és r .



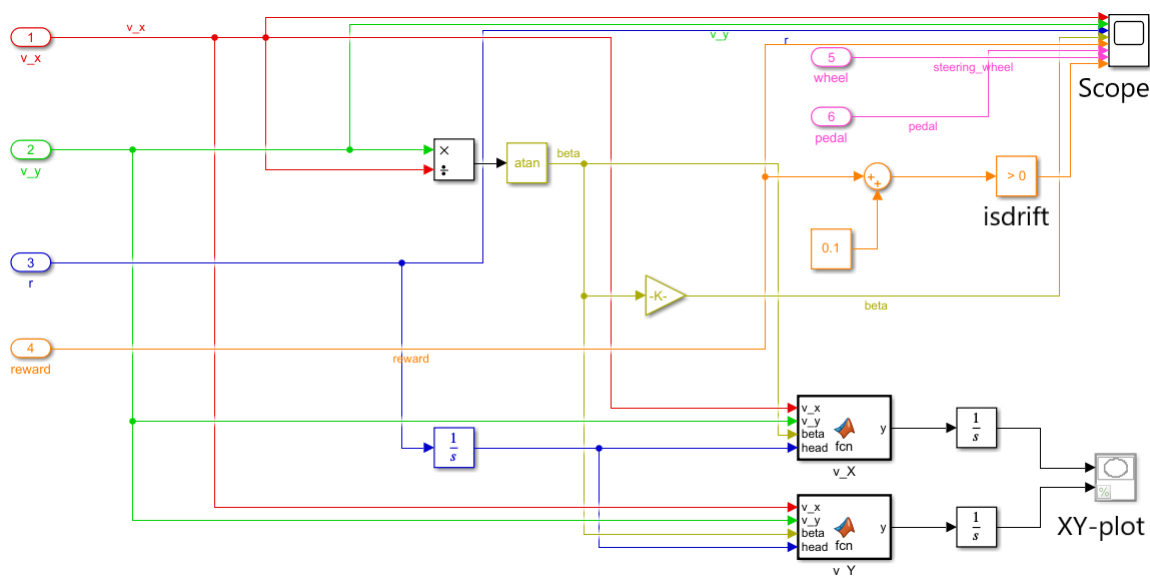
5.4. ábra. A járműmodell rendszere a Simulink modellben.

Jelölés	Név	Érték	Mértékegység
g	gravitációs gyorsulás	9.81	$\frac{m}{s^2}$
a	első féltengely	1.35	m
b	hátsó féltengely	1.37	m
m	a jármű tömege	1810	kg
I_z	a jármű inerciája	2500	$kg \cdot m^2$
C_α	kanyarodási merevség	300000	-
μ	tapadási együttható	0.95	-
k_2	2. fokozat áttételi aránya	2.59	-
k_0	végso áttételi arány	3.465	-

5.2. táblázat. A járműmodellhez alkalmazott konstans értékek[2]

A „Monitor” rendszer a járműmozgások és az ágens működésének elemzésére szolgál grafikus információk felhasználásával. Egy scope blokk grafikonokat állít elő, melyek mutatják az állapotváltozókat, a β csúszásszög, az akcióváltozókat és a jutalom időbeli eloszlását a adott epizódon belül. Ezek mellett továbbá egy empirikus bináris mérőszám („isdrift”) is definiálásra került, amely jelzi, ha a jármű a munkaponthoz közeli drift állapotban van-e vagy sem. Ezek segítségével elemezhető a jármű mozgása és ellenőrizhető a driftelés állapotának fennállása.

Az elemzéshez érdemes vizsgálni a mozgás pályáját is, amelyet egy „XY-plot” blokk rajzol k. Ehhez két MATLAB függvényblokk koordináta transzformációt hajt végre a jármű koordinátakeretéből egy globális koordináta síkra, ahol a jármű kiindulási helyzete az origó, és a kezdő haladási irány (amerre az autó eleinte néz) az x tengely pozitív iránya. Ezen vizuális eszközök minden szimuláció kezdetekor megjelennek, valamint folyamatosan rendelkezésre állnak az ágens tanítása alatt is. A következő fejezetben az eredmények tárgyalásakor (többek között) ezen grafikonok kerülnek elemzésre.



5.5. ábra. A „Monitor” rendszer.

6. fejezet

Kísérletek és eredmények

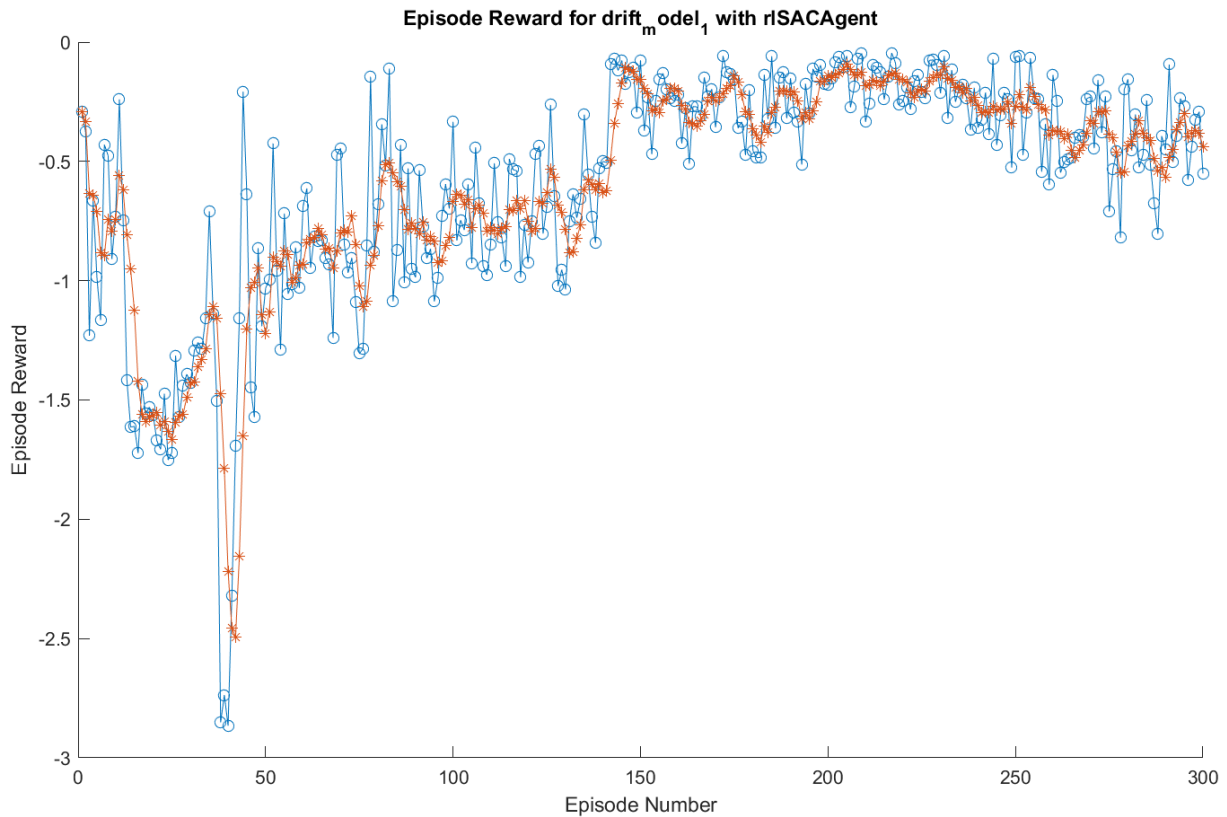
Ismétlésként, a kutatás célja egy olyan önvezető ágens kifejlesztése, amely képes kialakítani és fenntartani egy megadott céldrift állapotot. Ezen alapulva a feladat két részfeladatra lett felbontva. Először egy olyan ágens betanítása volt a cél, amely minden epizód kezdetén a célállapotból indul, és a feladata, hogy megtartsa ezt az állapotot. Mivel ez sikeresnek bizonyult, ezután egy munkapont megtalálására koncentráló feladatban az ágenszt a driftelés megvalósítására tanítottuk egy kanyaradó egyensúlyi állapotból. Ezen részfeladatok és eredményeik a következő fejezetekben kerülnek tárgyalásra. Az ágens beavatkozási időköze mindkét esetben 0.1 másodperc, tehát 10 műveletet hajt végre 1 másodperces szimulációs idő alatt. Ezen érték megválasztása elsősorban a hardveres limitációkra vezethető vissza.

6.1. A célállapotból indított helyzet megtartása

A feladat megoldásának leegyszerűsítéséhez az ágens számára egy know-how került beépítésre. A cél egyensúlyi állapot kiszámításakor (lásd a 5.1 fejezetet) az ideális bemeneti paraméterekre a következő két értéket kaptuk: $\delta = -10 \text{ rad}$ és $F_{x_r} = 3747.8719 \text{ N}$. Ebből az ágens számára átkonvertálva a következő akcióértékek jönnek ki: $ped_{acc} \approx 0.5$ és $\delta_{steer} = -150^\circ$. Ennek segítségével az ágens cselekvési tartománya értelmes módon lekorlátozható az optimális akciók egy 10%-os környezetére, azaz $ped_{acc} \in [0.45, 0.55]$ és $\delta_{steer} \in [-165^\circ, -135^\circ]$. Ebben a feladatban az ágens számára ezen akcióterület került definiálásra.

Az először kipróbált hosszabb epizódokon ($T = 5, 6, \dots, 30$) történő tanítás sikertelennek bizonyult, az ágens nem tudta megtartani a drift állapotot. Az ágens viselkedését elemezve megállapítható volt, hogy a jármű általában az első szimulációs másodpercben hamar elhagyja a munkapontot, így a hátralévő időlépések haszontalanok, mert a csökkentett akcióterület lehetetlenné teszi a célállapot ismételt megközelítését. Ezért rövidebb epizódok segítségével, először $T = 1$ értéket állítva végeztünk egy kísérletet, tehát az

ágens gyakrabban találkozhat azzal a ponttal, amikor elhagyja a célállapotot. Az eredmény ígéretes volt: 300 tanulási epizód alatt az ágens megtanulta, hogyan kell irányítani az autót, hogy egy másodpercig drift állapotban maradjon.

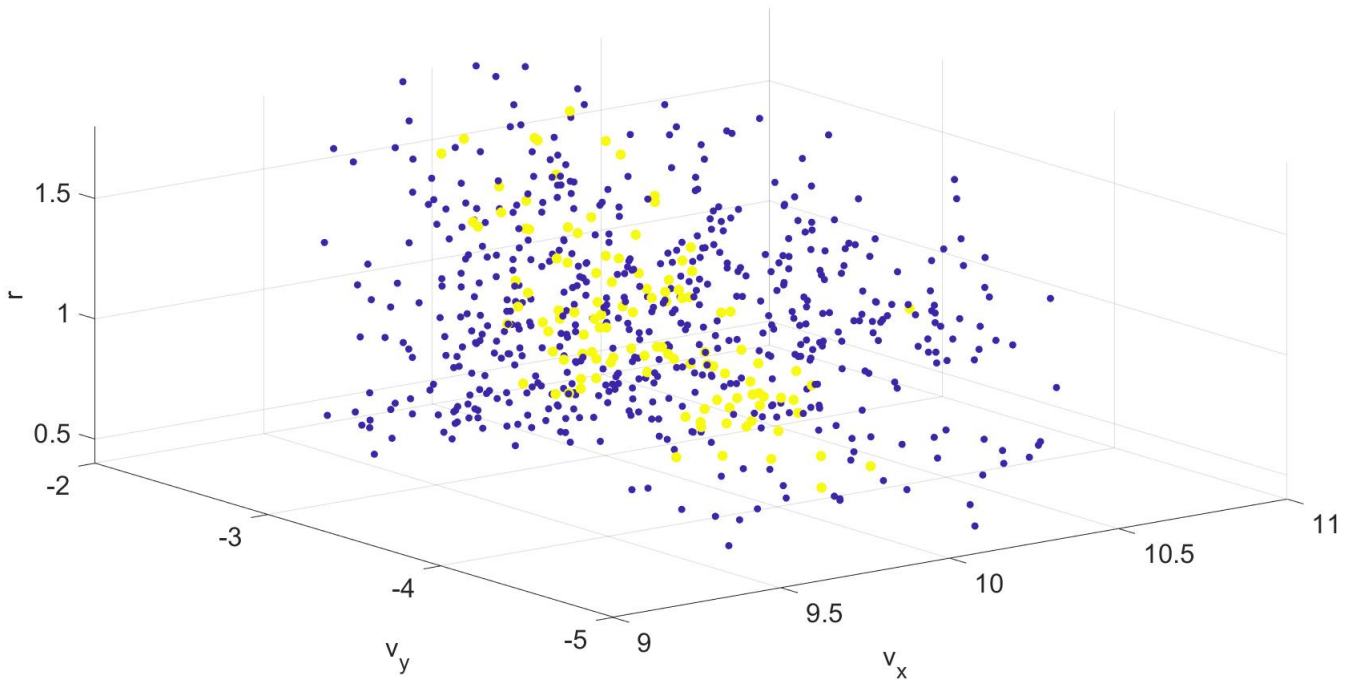


6.1. ábra. Az 1 másodperces driftelést fenntartó ágens tanulási görbéje. A kék pontok az epizód során kapott jutalmak összegének trendjét mutatják. A narancssárga görbe simító görbeként működik, az elmúlt néhány epizód átlagos jutalomösszegét mutatja. Az első 50 epizódban megfigyelhető a megnövekedett felfedezés, majd a jutalomösszegek 0 felé kezdenek konvergálni, csökkent oszcillációval, vagyis a felfedezés mértéke (tehát az entrópia és annak súlya) csökken. 150 epizód után az ágensnek sikerült lehetőséget találnia a jutalom növelésére.

Azonban az előző ágens továbbra sem volt képes a drift hosszantartó fenntartására, csupán átlagosan 1.5 másodpercig. Ezért újabb tanítást végeztünk az előző gondolatmenet alapján, most már $T = 2$ beállítással, és az előző ágens tudását felhasználva (azaz a hálók kezdeti súlyai az előző ágens hálójának súlyai). Ezt a műveletet nevezhetjük továbbtanításnak. A kitűzött cél elérése érdekében ezt a műveletet még egyszer meg kellett ismételni $T = 3$ beállítással a $T = 2$ -re tanított ágensre, ugyanis a második eredmény még mindig nem volt megfelelő. A második továbbtanítás által eredményezett ágens viszont már tetszőleges ideig képes volt tartani a drift állapotot, tehát sikeresen megoldotta a feladatot.

A kezdeti állapot megválasztását is érdemes elemezni, így mérhető az ágens állapotadaptív képessége azáltal, hogy a kezdőállapot változása mennyire befolyásolja az ágens teljesítményét. Ehhez véletlenszerűen kiválasztásra került 300 (v_x, v_y, r) állapot-pont a célállapot egy adott környezetéből, és mindegyik tesztállapothoz az ágens egy 3 másodperces szimuláción keresztül került kiértékelésre, hogy képes-e megtartani a drift állapotot. Ha az 'isdrift' indikátor egyenlő volt 1-el az utolsó időlépésben, akkor a kezdeti állapotot "drifteléshez jó" (sárga) címkével lett ellátva, ellenkező esetben pedig egy "drifteléshez nem megfelelő" (kék) címkével, majd ezek a pontok színükkel együtt egy 3 dimenziós pontábrára lettek helyezve.

A sárga pontok egy homogén tartományt alkotnak, melynek felső határain túl többségében olyan v_y és r állapotok vannak, amelyekben a driftelés már fizikailag nem irányítható, azaz a jármű kipördülése már elkerülhetetlen. A tartomány alsó határain pedig az akcióter limitációja okoz problémát: olyan kormánymozdulatokra lenne szükség a drift szabályozásához, amelyek kívül esnek a megadott tartományon.



6.4. ábra. A kezdeti állapotokat mutató 3 dimenziós szórásdiagram.

6.2. A célállapot megközelítése és megtartása kanyarodásból

Miután a leegyszerűsített feladatot sikerült megoldani a SAC ágens segítségével, ezt követően a drift állapot elérése volt a cél. A feladat megoldásához ismételten alkal-

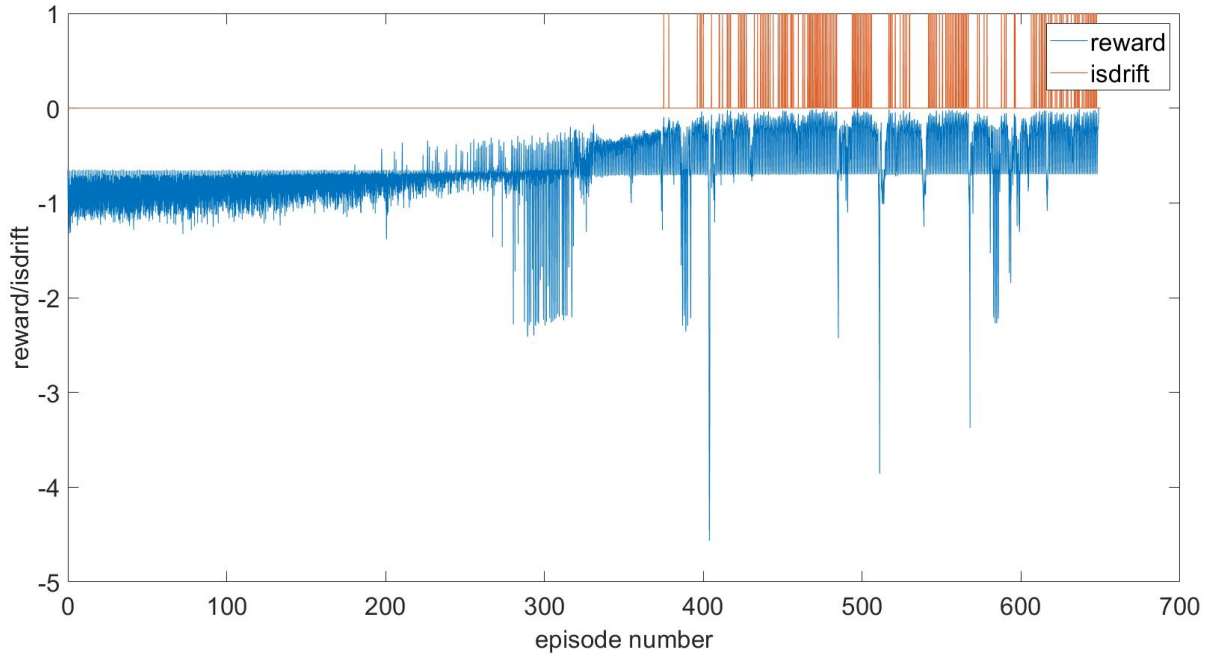
mazásra került egy akciótér szűkítés, de nem olyan mértékben, mint az előző esetben, ugyanis a driftelés megvalósításához nagyobb tartományra van szükség. Ezért ebben a feladatban az akciótér a következő: $ped_{acc} \in [0, 0.85]$ és $\delta_{steer} \in [-200^\circ, 100^\circ]$ Az epizód hossza $T = 5$ -nek lett megválasztva.

A kiinduló állapotnak itt egy balra kanyarodó állapot lett megválasztva, hasonló módszerrel, mint ahogy a cél drift állapot lett meghatározva. Azonban annyi különbség van, hogy az egyszerű kanyarodó állapotban a hátsó kerekek nem szaturálnak, így a (2.18) egyenlethez a $|\alpha_r| \leq \alpha_{sl_r}$ feltétel lett meghatározva. Az eredményül kapott kanyar egyensúlyi pont állapota a következő:

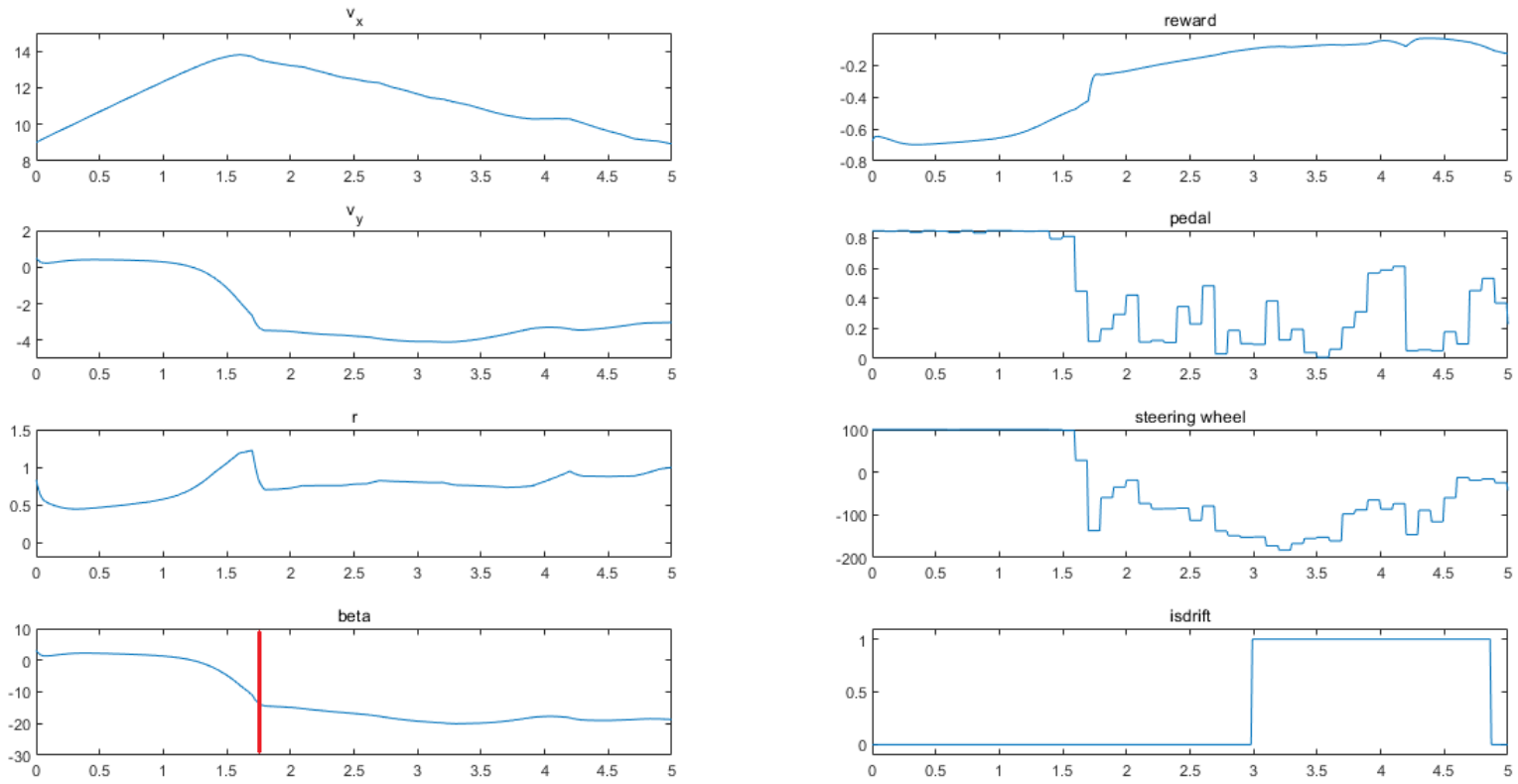
$$S_{turn} = (v_{x_{turn}}, v_{y_{turn}}, r_{turn}) = (9\text{m/s}, 0.825\text{m/s}, 0.8334\text{rad/s}) \quad (6.1)$$

ahol az egyenletrendszer megoldásához a $v_x = 9\text{m/s}$ és $r = 0.8334\text{rad/s}$ két előzetes paraméter lett megválasztva.

A sikeres tanításhoz a hiperparaméterek megválasztása kulcsfontosságúnak bizonyult, különösen az entrópiára vonatkozó paraméterekre és a buffer méretére volt érzékeny a tanítás. Néhány hiperparaméterekre vonatkozó tesztet követően sikerült megtalálni egy jól működő kombinációt, és az ágens egy 650 epizódos tanítás során sikeresen megtalálta a munkapontot a kanyarodó állapotból. Ezt az indikátorváltozó figyelése segítségével állapítottuk meg.

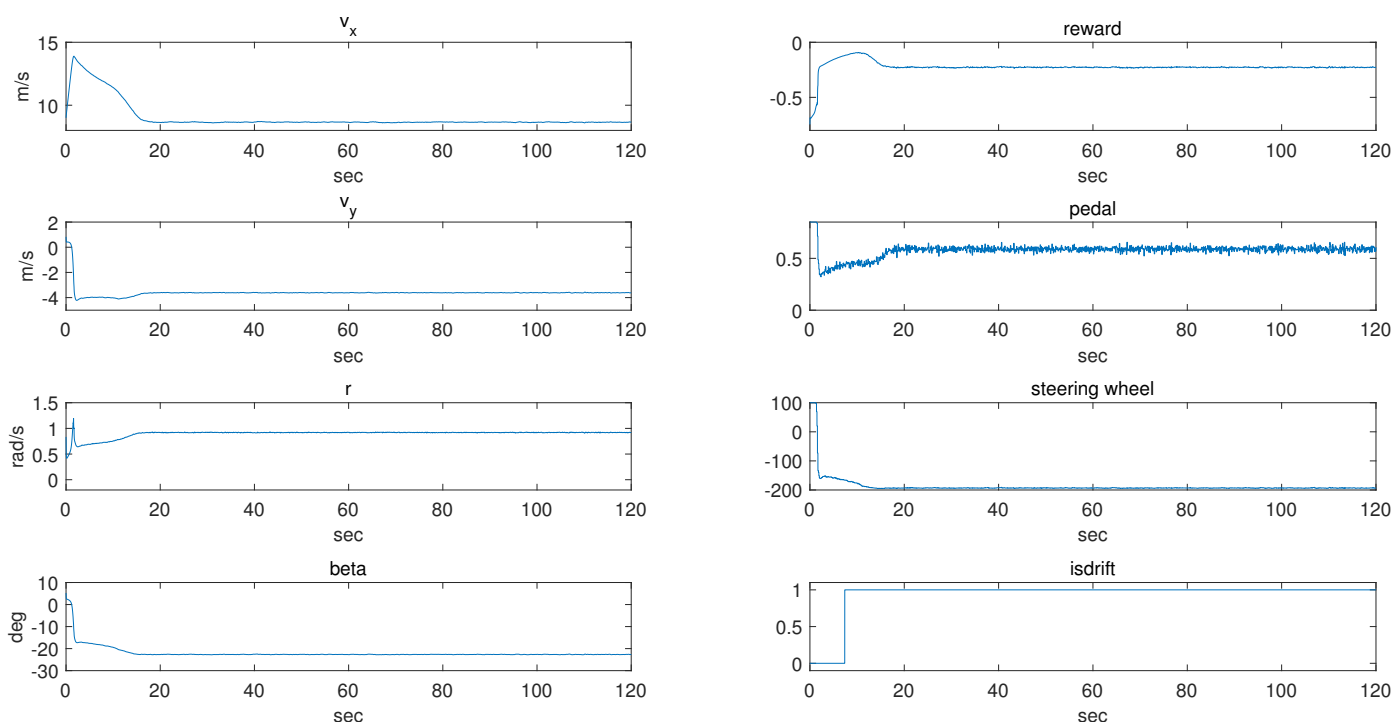


6.5. ábra. A jutalom értékek (kék) eloszlása a 650 epizódos sikeres tanítás során. A 300. epizód környékén észrevehető a jutalmak szórásának megnövekedése, valamint azt követően azok átlagának emelkedése. Sárgával van jelölve az indikátor értékeinek eloszlása.



6.6. ábra. A 468. epizód scope grafikonja. A béta értékeknél (bal alsó sarok) jelölt piros vonalnál kezdődik a driftelés, ugyanakkor a munkaponthoz közel csak a 3. másodpercben kerül a jármű. Ez jobb alul az indikátornál látszik, ahogy az is, hogy az epizód végére az autó eltávolodik a munkaponttól.

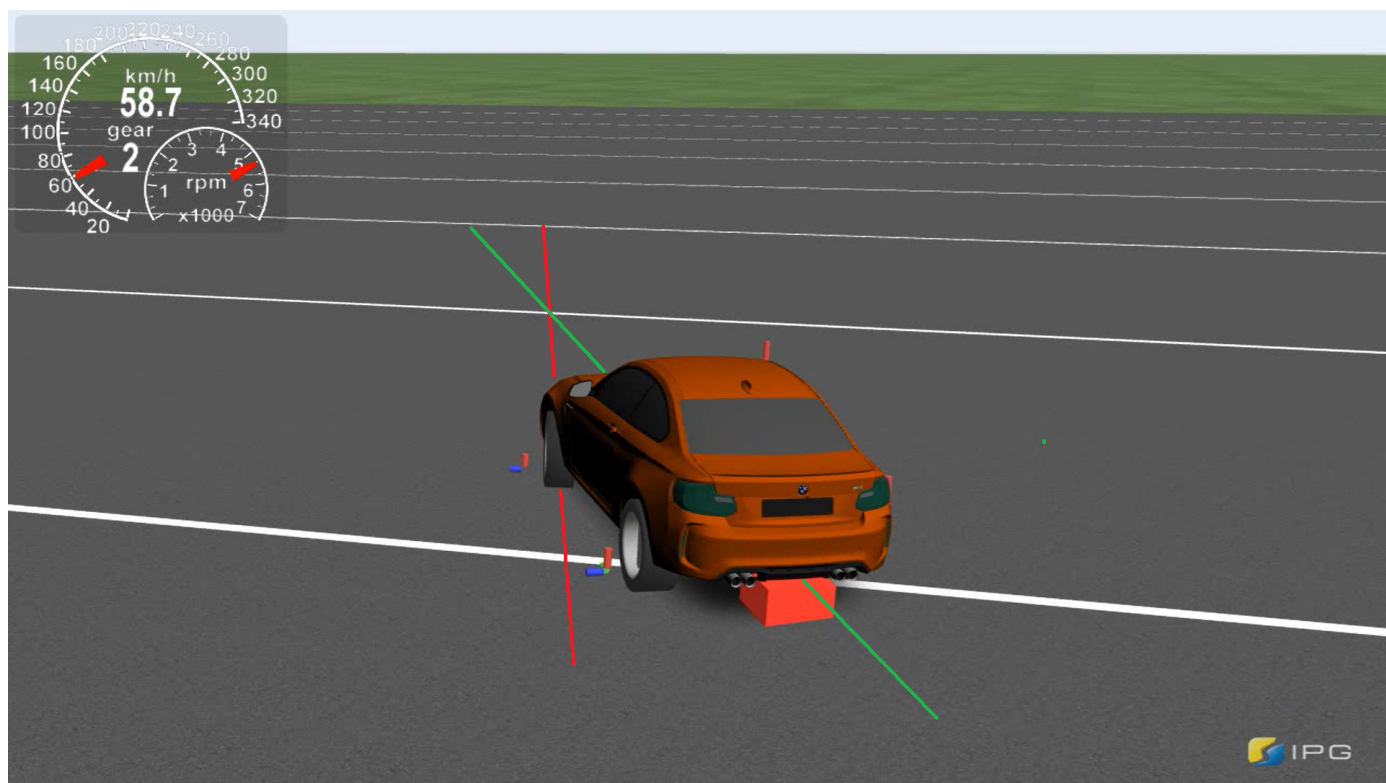
Az epizód hosszának növelésével azonban kiderült, hogy az ágens még nem volt képes a drift állapot hosszan tartó stabilizálására, az autó mindig eltávolodott a munkaponttól. Ezért ugyanaz a továbbtanítás került alkalmazásra, mint az egyszerűbb drift stabilizálás megvalósításánál, azaz a már tanított ágens néhány $T = 10$ hosszúságú epizód erejéig további tanítás alá vetettük. A módszer ebben az esetben is működött, és egy olyan ágens eredményezett, amely a drift állapot elérését követően tetszőleges időn keresztül képes volt stabilan tartani is azt.



6.7. ábra. A legjobb ágens teljesítménye egy 120 másodperces teszt szimuláció alatt. Az indikátoron láthatjuk, hogy a 8. másodperc körül 0-ról 1-re ugrik (eléri a munkapontot), és a szimuláció végéig 1 marad.

Az ágens teljesítményének validációja érdekében egy valósághűbb fizikai modellel rendelkező szoftverben is teszteltük az ágenst. Az IPG CarMaker sokkal valósághűbb járműmodelleket és környezeti fizikát biztosít, mint a bemutatott Simulink implementáció, így az ágens ottani tesztelése jelentős fontosságú, tekintettel a lehetséges valós járművön történő tesztelésre és alkalmazásra. A program összekapcsolható a MATLAB/Simulink környezettel, így a tanított ágens könnyedén tesztelhető.

A CarMaker-ben ugyanazok a jármű paraméterek kerültek betáplálásra, mint amit a Simulink modellnél is alkalmaztunk (lásd 5.2 fejezet). Az autót először egy egyszerű PID kontrollert álló helyzetből megindítja, majd átadja a kontrollt az ágensnek. A szimuláció során az ágens az álló helyzetből történő gyorsítást követően megfelelően megközelítette a drift célpontot és sikeresen stabilizálta a mozgást.



6.8. ábra. A CarMaker szimulációban operáló ágens. A piros vonal a kormányzott kerék, míg a zöld vonal az autó hossztengetyének vonalát mutatja. Ezek a vonalak a jármű előtt keresztezik egymást, ami azt jelenti, hogy ellenkormányzás történik.

7. fejezet

Következtetések

A jelen TDK dolgozatban bemutatott kutatásban egy megerősítéses tanulás alapú ágens kifejlesztése történt drift mozgás stabilizálásának céljából a SAC tanulóalgoritmus és egy síkbeli egypályás járműmodell segítségével egy MATLAB/Simulink szimulációs környezetben.

Az ágens egy drift egyensúlyi célállapotba helyezve sikeresen képes volt megtanulni annak stabil megtartását tetszőleges időn keresztül. A tanulás során az ágens az elsajátított információ segítségével továbbá nem csak egy megadott kezdőállapotból volt képes a driftelés fentartani, hanem a kezdőállapot egy bizonyos homogén környezetéből választott bármely kezdőállapot esetében is.

A célállapot megközelítését és megtartását, mint egy komplexebb feladatot, az ágens szintén sikeresen végrehajtotta egy szimpla kanyarodó egyensúlyi kezdőállapotból. Ennek teljesítménye továbbá kiértékelésre került egy valósághűbb fizikai motorral rendelkező szimulációs szoftverben is, ahol a munkapont megtalálása és megtartása szintén sikeresen megtörtént.

A tanításhoz emberi vezető által vagy egyéb szabályozó módszerek segítségével rögzített előzetes adat nem került felhasználásra. Minden, az ágens számára rendelkezésre bocsájtott know-how kizárólag járműmodellből kinyerhető információon alapult. Ez a feldolgozott előzetes publikációk alapján ez új eredménynek számít a megerősítéses tanulás alapú alkalmazások körében.

A pozitív eredményeken alapulva a jelen kutatás folytatódik. A kísérletek során alkalmazott algoritmus instabilitásából adódó problémák további munkát igényelnek. Tervezetten további algoritmusok és tanítási feltételek kipróbálására is sor kerül majd, továbbra is elkerülve bárminemű előzetes adat felhasználását.

A későbbiek során egyszerre több egyensúlyi pont alkalmazásával olyan ágens kifejlesztése a cél, amely képes drift egyensúlyi pontokat összekötő mozgásra, illetve egy megadott nyomvonalon vagy útszakaszon keresztüli folyamatos driftelés végrehajtására. További cél lehet még az útviszonyok megváltoztatására sikeresen reagáló adaptív

tulajdonság kifejlesztése.

A CarMaker-ben elért jó eredmények alapján pedig későbbi valós járművön történő tesztelések felé is nyitott a kutatás. Ennek a megvalósítására már rendelkezésre áll egy tesztautó és tesztpálya környezet Zalaegerszegen, amely korábbi valós tesztelések során is tökéletesen alkalmasnak bizonyult. [3]

Összességében kijelenthető, hogy a driftelés kezdeti szimulációs megvalósítása sikeres, és a kutatás egy kiforrottabb szabályozó rendszer megalkotására és valós alkalmazására tovább folytatódik.

Irodalomjegyzék

- [1] Manuel Acosta and Stratis Kanarachos. Teaching a vehicle to autonomously drift: A data-based approach using neural networks. *Knowledge-Based Systems*, 153:12–28, 2018. 4.2, 4.1
- [2] Ádám Bárdos, Ádám Domina, Zsolt Szalay, Viktor Tihanyi, and László Palkovics. Mimo controller design for stabilizing vehicle drifting. In *2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo)*, pages 000187–000192. IEEE, 2019. 1, 2.3, 4.1, 5.2
- [3] Ádám Bárdos, Ádám Domina, Viktor Tihanyi, Zsolt Szalay, and László Palkovics. Implementation and experimental evaluation of a mimo drifting controller on a test vehicle. pages 1472–1478, 2020. 1, 2.3, 4.1, 7
- [4] Peide Cai, Xiaodong Mei, Lei Tai, Yuxiang Sun, and Ming Liu. High-speed autonomous drifting with deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):1247–1254, Apr 2020. 1, 4.3
- [5] Mark Cutler and Jonathan P. How. Autonomous drifting using simulation-aided reinforcement learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5442–5448, 2016. 1, 4.3
- [6] Szilárd Czibere, Ádám Domina, Ádám Bárdos, and Zsolt Szalay. Model predictive controller design for vehicle motion control at handling limits in multiple equilibria on varying road surfaces. *Energies*, 14(20):6667, 2021. 4.1
- [7] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013. 4.3
- [8] Xuan Di and Rongye Shi. A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to ai-guided driving policy learning. *Transportation research part C: emerging technologies*, 125:103008, 2021. 1

- [9] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018. 3.3
- [10] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017. 5.1
- [11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 3.3
- [12] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019. 3.3, 5.1
- [13] R Hadekel. The mechanical characteristics of pneumatic tires. *Clearinghouse Fed Sci & Tech Info*, 1952. 2.2
- [14] Rami Y Hindiyeh and J Christian Gerdes. Equilibrium analysis of drifting vehicles for control design. In *Dynamic Systems and Control Conference*, volume 48920, pages 181–188, 2009. 2.2, 2.3
- [15] Rami Yusef Hindiyeh. *Dynamics and control of drifting in automobiles*. PhD thesis, Stanford University, 2013. 2.2, 2.2, 2.2, 2.3
- [16] Reza N Jazar. *Advanced vehicle dynamics*. Springer, 2019. 2.1, 2.1
- [17] Edo Jelavic, J Gonzales, and Francesco Borrelli. Autonomous drift parking using a switched control strategy with onboard sensors. *IFAC-PapersOnLine*, 50(1):3714–3719, 2017. 1, 4.1
- [18] Xuewu Ji, Xiangkun He, Chen Lv, Yahui Liu, and Jian Wu. Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits. *Control Engineering Practice*, 76:41–53, 2018. 4.2
- [19] Tianxin Li and Kara M Kockelman. Valuing the safety benefits of connected and automated vehicle technologies. In *Transportation Research Board 95th Annual Meeting*, volume 1, 2016. 1
- [20] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 3.3

- [21] László Orgován, Tamás Bécsi, and Szilárd Aradi. Autonomous drifting using reinforcement learning. *Periodica Polytechnica Transportation Engineering*, 49(3):292–300, 2021. 4.3
- [22] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992. 2.2
- [23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. (document), 3
- [24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. 3, 3.1
- [25] Efstathios Velenis, Emilio Frazzoli, and Panagiotis Tsiotras. On steady-state cornering equilibria for wheeled vehicles with drift. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3545–3550. IEEE, 2009. 4.1
- [26] Efstathios Velenis, Diomidis Katzourakis, Emilio Frazzoli, Panagiotis Tsiotras, and Riender Happee. Steady-state drifting stabilization of rwd vehicles. *Control Engineering Practice*, 19(11):1363–1376, 2011. 4.1
- [27] Zsolt J Viharos and Richárd Jakab. Reinforcement learning for statistical process control in manufacturing. *Measurement*, page 109616, 2021. 3.1
- [28] Zsolt János Viharos, Krisztián Balázs Kis, Ádám Fodor, and Máté István Büki. Adaptive, hhybrid ffeature selection (ahfs). *Pattern Recognition*, 116:107932, 2021. 3.2
- [29] Christoph Voser, Rami Y Hindiyeh, and J Christian Gerdes. Analysis and control of high sideslip manoeuvres. *Vehicle System Dynamics*, 48(S1):317–336, 2010. 2.3, 4.1
- [30] Fenghua Yang. Research on the course, form and strategy of autonomous driving competition. In *Journal of Physics: Conference Series*, volume 1948, page 012093. IOP Publishing, 2021. 1
- [31] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 3.3