Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedés- és Járműmérnöki Kar
Természettudományi Kar

Megerősítéses Tanulás Alapú Járműdrift Szabályozás Tabuláris Q-tanulással

TDK Dolgozat

Tóth Szilárd Hunor

Témavezetők:

Dr. Bárdos Ádám

Géjárműtechnológia 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, 2022

Kivonat

A speciális vezetési technikák, például a driftelés végrehajtása még a hivatásos emberi sofőrök számára is kihívást jelenthet. Az ilyen manőverek azonban elengedhetetlenek lehetnek egyes balesetek elkerüléséhez olyan kritikus közúti helyzetekben, mint például a tapadás elvesztése csúszós útfelületen, vagy éppen egy hirtelen, az autó elé kerülő akadály (pl. akár egy vad) elől kitérő manőver végrehajtása. A jelen TDK dolgozat célja beszámolni azon saját kutatás új eredményekről, amelynek fő célja egy önvezető ágens kifejlesztése drift mozgásszabályozáshoz egy MATLAB/Simulink alapú szimulációs környezetben megerősítéses tanulás segítségével. A jelen dolgozatban bemutatott kutatási munka legfőbb motivációja a Soft Actor-Critic (SAC) algoritmusnál tapasztalt tanulási instabilitást okozó problémák kiküszöbölése, amelyhez egy Tabuláris Q-learning algoritmus került alkalmazásra. Az autonóm drifteléssel kapcsolatos eddigi kutatásokon alapulva az új eredmények közül a legfontosabb, hogy sikerült az eredendően folytonos problémát egy teljesen diszkrét alapú algoritmussal, jó eredménnyel megoldani.

Tartalomjegyzék

1. Bevezető **4**

2. Az alkalmazott járműmodell leírása **6**

2.1. Egy nyomvonalú járműmodell 6

2.2. Abroncsmodellek 8

2.3. A driftelés definíciója és egyensúlyi pontjainak kiszámítása 10

3. Megerősítéses tanulás **12**

3.1. A megerősítéses tanulás elemei és a Q-learning algoritmus 13

3.2. Felfedezési stratégiák 16

4. Driftelés autonóm járművekkel **18**

4.1. Klasszikus kontroll alapú megközelítések 18

4.2. Megerősítéses tanulás alapú megközelítések 19

4.3. A felügyelt tanulás és hierarhikus modellek alkalmazhatósága 20

5. Az állandósult drift feladat pontos leírása **22**

5.1. A driftelés definiálása megerősítéses tanulás feladatként 22

5.2. A jáműmodell és az ágens implementációja MATLAB/Simulink környezetben 24

6. Kísérletek és eredmények **27**

6.1. A folytonos környezet megfelelő véges reprezentációjának megtalálása . . 27

6.2. A driftelés megvalósítása ε-mohó felfedezéssel 29

6.3. Az adaptív felfedezéssel kapcsolatos eredmények 32

7. Következtetések és kitekintés **35**

7.1. Kitekintés 36

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[9].

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 fenntartson 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 (egy szarvas, egy gyalogos, stb.) és egyéb külső erőhatások mind olyan instabil állapotba képesek kényszeríteni a járművet, amelyet egy jól kifinomult drift manőverrel még kontrollálni lehet[34]. Azonban az átlagos vezető legtöbször nem rendelkezik a megfelelő képességekkel, hogy ezeket a manővereket megfelelően végrehajtsa, emiatt az ilyen szituációk jelentős számban balesethez vezetnek. Egy 2015-ös tanulmány[21] szerint az Egyesült Államokban - a 2013-as GES (Általános Becslések Rendszere) baleseti jelentései alapján - az irányítás elvesztésével összefüggő, illetve vadállat vagy gyalogos elgázolásával kapcsolatos balesetek relatív gyakorisága rendre 8,32% és 6,69% voltak, amik

szám szerint összesen 458 000 és 369 000 balesetet jelentettek. Ezek jelentős számok, amelyeket 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[33] szintén fontos kutatási lehetőséget biztosít.

Megelőző kutatási eredmények az autonóm driftelés megvalósítására azt mutatják, hogy érdemes további munkát folytatni a területen, ugyanis még nincsnek 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 [18]. 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, 10].

A megelőző (2021-es) TDK dolgozatom is szintén a megerősítéses tanulás alapú autonóm driftelés megvalósítását tűzte ki, és mutatott be sikeres eredményeket a Soft Actor-Critic (SAC) algoritmus alkalmazásával, amellyel akkor sikerült a Gépjárművek szekcióban egy I. helyezést elérnem. Az akkor bemutatott algoritmus azonban rendelkezett tanítás alatt felmerülő instabilitási tulajdonságokkal (pl. katasztrofális felejtés), amely problémák egy másik tanulóalgoritmus, a tabuláris Q-tanulás alkalmazásának segítségével oldódtak meg.

Ezen TDK dolgozatban bemutatott kutatás célja egy megerősítéses tanulás segítségével fejlesztett robosztus ágens megtervezése és megalkotása szimulácós 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 tabuláris Q-tanulással tanított ágens irányít. A tanításhoz semmilyen előzetes adat nem kerül alkalmazásra, kizárólag járműmodellből kinyerhető információ történik feldolgozásra.

A következő fejezetben kerül sorra a járműmodell és az egyensúlyi állapotok egyenletalapú kiszámítási módszerének leírása. Azt követően a megerősítéses tanulásról, a Q-learning algoritmusról és annak itt alkalmazott változatairól, működéséről szóló fejezet kerül majd tárgyalásra. A 4. fejezetben az autonóm drifteléssel kapcsolatos megelőző irodalom részletesebb bemutatása található. Az 5. fejezetben a jelen dolgozat keretein belül megvalósított drift feladat pontos leírása olvasható, melyt követően a kísérletek és az eredmények kerülnek tárgyalásra. A dolgozat konklúziókkal és kitekintéssel, valamint irodalomjegyzékkel zárul.

2. fejezet

Az alkalmazott járműmodell leírása

Az alkalmazott járműmodell helyes megválasztása kritikus faktora bármely járműirányítási feladat megoldásának. 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, ugyanakkor a driftelés megvalósítása nem lehetetlen elsőkerék- illetve négykerék-meghajtású gépjárművekkel sem.

A szimulációhoz használt modell egy síkbeli, egypályás dinamikus modell két különböző gumiabroncs modellel. Ennek a modellnek az előnye, hogy figyelmen kívül hagyja a dőlés dinamikákat és az aerodinamikát, amelyek sokkal kevésbé fontosak a driftelés megvalósításának szempontjából. Ennek köszönhetően a modell egyszerűen és gyorsan szimulálható, ami így nem igényel erősebb számítási kapacitással rendelkező hardver használatát.

Megelőző eredmények [31][2][3] mutatják, hogy ez az alkalmazott konstrukció helyes és jól működik szimulációban és a valóságban is.

2.1. Egy nyomvonalú járműmodell

A síkbeli járműmodellek 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 [17]:

$$F_x = m\dot{v_x} - mrv_y \tag{2.1}$$

$$F_y = m\dot{v_y} + mrv_x \tag{2.2}$$

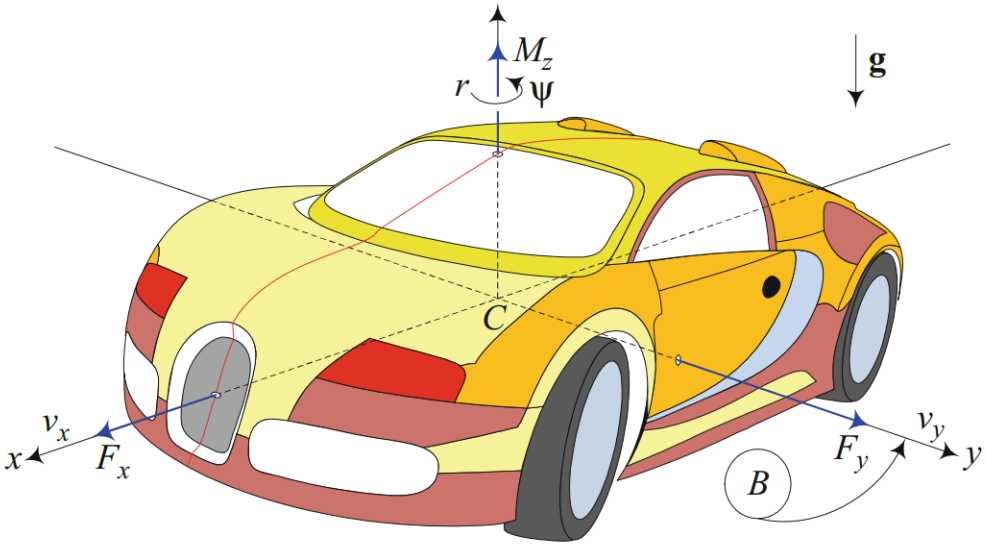
$$M_z = I_z * \dot{r} \tag{2.3}$$

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 kifejezhetőek:

$$\dot{v}_x = \frac{1}{m} F_x + r v_y \tag{2.4}$$

$$\dot{v}_y = \frac{1}{m} F_y - r v_x \tag{2.5}$$

$$\dot{r} = \frac{1}{I_z} M_z \tag{2.6}$$



2.1. ábra. A síkbeli dinamikus járműmodell karosszériájának koordinátarendszere, dinamikus változói és erőrendszere[17]

A fenti erő komponenseket továbbá levezethetjük a kerekekre ható erők összhatásaként is (levezetéshez lásd [17], 125-131 és 145-160 old.):

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

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

$$M_z = a F_{y_f} \cos\delta + a F_{x_f} \sin\delta - b F_{y_r} \tag{2.9}$$

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. Abroncsmodellek

Az gumimodell pontos és értelmes definiálása a legfontosabb feladat 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 [23]. A driftelés szimulálásának szempontjából azonban ezek a technikák túlzottan bonyolultak.

A kefe gumiabroncs modell (brush tire model) [14] három alépítményre osztja az abroncsot: a gyűrűre, a vázra és a rugalmas „kefékre", 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 hossztengelyéhez képest. Ezek a következőképpen írhatóak le [16]:

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

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

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:

$$F_{z_f} = \frac{m \cdot g \cdot b}{a + b} \tag{2.12}$$

$$F_{z_r} = \frac{m \cdot g \cdot a}{a + b} \tag{2.13}$$

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:

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

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:

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

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 bonyololultabb konstrukciót kíván, ami szükségessé teszi a kerékforgás-dinamika hozzáadását a teljes járműmodellhez. Ezt elkerülendő, Hindiyeh [16] egy egyszerű megoldást javasol olyan esetekre, ahol a kerekekre ható hosszirányú erőt bemeneti paraméterként kezeljük, ami így pontosan alkalmazható az itteni szimulációkhoz:

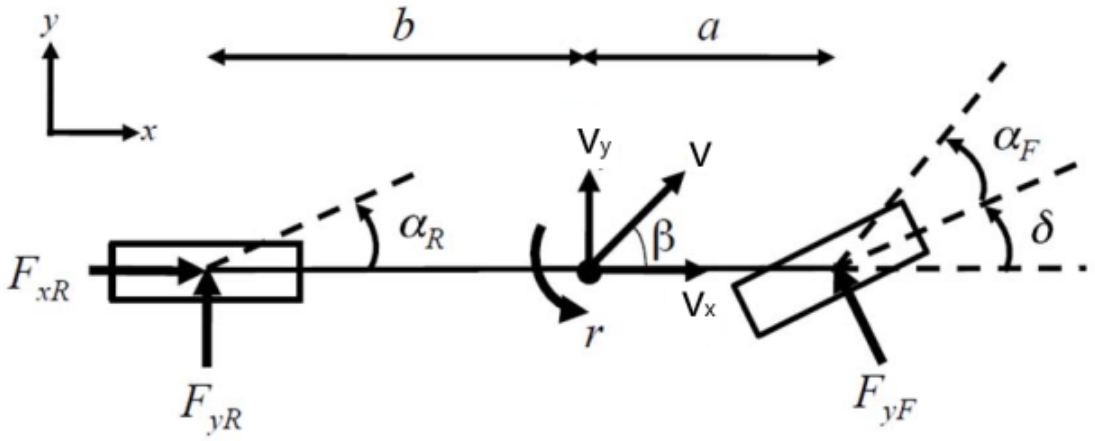
$$\xi = \frac{\sqrt{(\mu F_{z_r})^2 - F_{x_r}^2}}{\mu F_{z_r}} \tag{2.16}$$

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

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

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

Részletes levezetésekért és bizonyításokért lásd [16], 21-34. oldal.



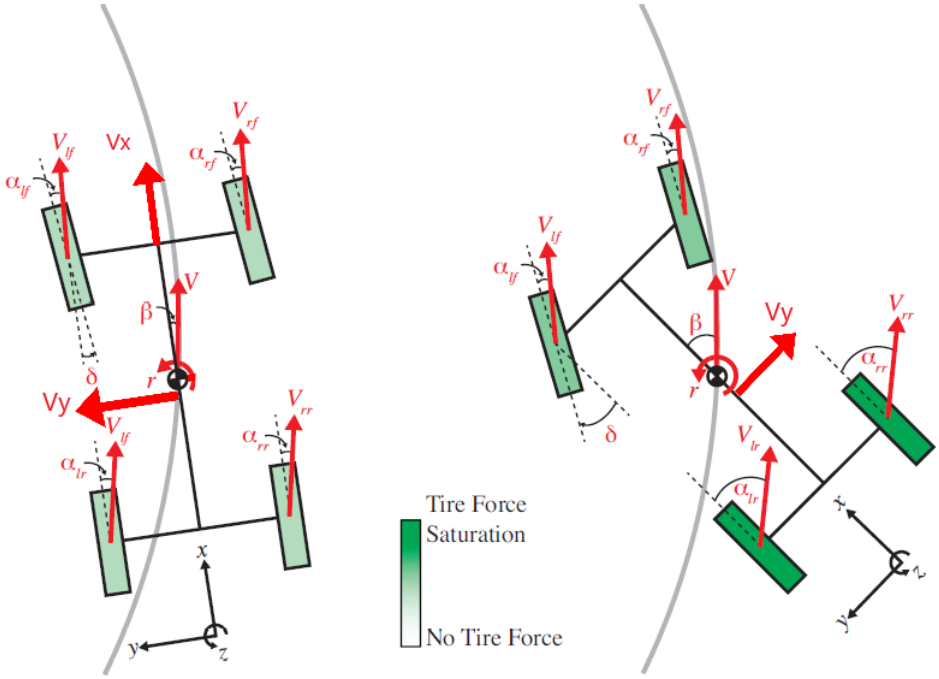
2.2. ábra. Az abroncsokra ható erőkomponensek és csúszásszögek ábrázolása a jármű koordinátarendszerében.[15]

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 (tehát elérik a fizikailag lehetséges maximális mértéküket), 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:

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

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. Az autó ebből 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.[16]

A fent ismertetett mozgásfolyamat egy példája az ún. állandósult drift feladatoknak, amelyek esetében a cél egy (vagy egymás után több) kitűntetett drift állapot megvalósítása és megtartása. Megelőző tudományos munkák[15] azt mutatják, hogy a driftelés egyensúlyi pontjai kiszámolhatóak egy, a járműmodellből kapott algebrai egyenletrendszer segítségével. Megelőző eredmények a kontroll területén[31][2][3] mutatják, hogy ez a - jelen kutatásban is alkalmazott - konstruckió helyes és jól működik szimulációban és a valóságban is.

Mivel egy állapot folyamatos fenttartása 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 következőképen írható le:

$$\dot{v}_x = \frac{1}{m} F_x + r v_y = 0 \tag{2.20}$$

$$\dot{v}_y = \frac{1}{m} F_y - r v_x = 0 \tag{2.21}$$

$$\dot{r} = \frac{1}{I_z} M_z = 0 \tag{2.22}$$

Ezekben az egyenletekben a már fentebb definiált gumimodell segítségével az F_x és F_y modellek kifejezhetőek, így egy 5 (v_x, v_y, r, F_{x_r}, δ) ismeretlenből álló algebrai egyenletrendszert kapunk. Fontos, hogy a hátsó kerék szaturáltsága meg legyen kötve, ezért a (2.18) egyenlet esetében az $|\alpha_r| > \alpha_{sl_r}$ 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 [25] 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") redelkezésre 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 következőek:

- 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. [24]

- Tekintsünk egy mobil tisztító robotot. A robot célja tekintélyes mennyiségű szemét

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. [25]

3.1. A megerősítéses tanulás elemei és a Q-learning algoritmus

A megerősítéses tanulás formális keretrendszert 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.

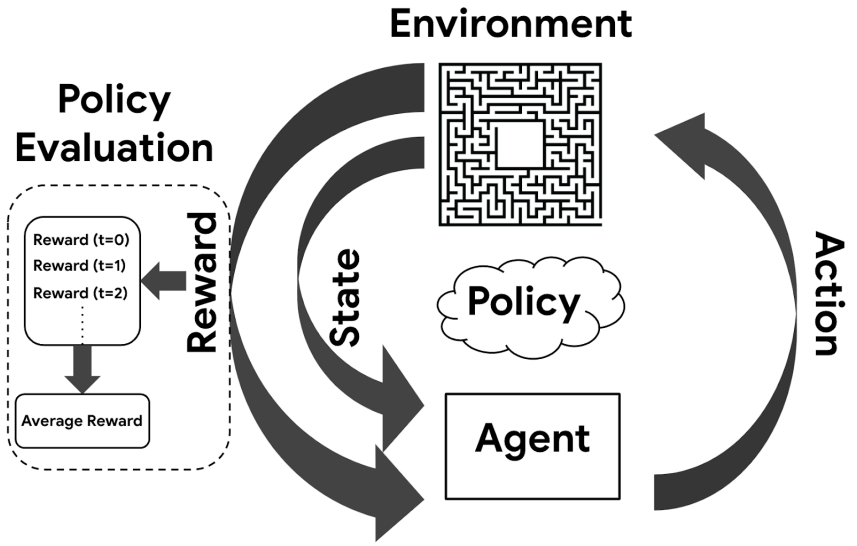
A tanulót és a döntéshozót *ágensnek* nevezzük. Az ágens kölcsönhatásba lép a *környezettel*, amely minden olyan objektumot tartalmaz, amely hatással van a döntéshozatalra. Ez akár tartalmazhat olyan információt is az ügynökről, amely azt fizikailag jellemzi, például az akkumulátor töltöttségi szintjét 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 $t = 0, 1, 2, \ldots$ 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 $\pi_t : S \to [0, 1]$ 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") ($0 < \gamma \leq 1$):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.1}$$

Szinte minden megerősítéses tanulási algoritmus magában foglalja valamilyen *érték-függvény*(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 $v_\pi : \mathcal{S} \to \mathbb{R}$ egy számérté-



3.1. ábra. Az ágens és a környezet interakciója [25]

ket rendel hozzá minden olyan állapothoz, amely az ágens tanulása során előfordulhat:

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

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

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

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 értékfüggvény*ük:

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

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

Megemlítés 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-tulajdonság*nak, vagy másképp *bootstrap-tulajdonság*nak 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. A TD algoritmusok egyik nagy előnye, hogy a bootstrap-tulajdonságnak köszönhetően jól képesek kezelni olyan feladatokat, amelyekben az ágens által leadott akció hatása a környezetre nem közvetlenül jelentkezik, hanem csak több időlépést követően.

*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 megtérülés*t (n-step return) alkalmaznak:

$$G_t^n\left(S_t\right) = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n v_\pi\left(S_{t+n}\right) \tag{3.5}$$

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

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha\left[G_t^n - V\left(S_t\right)\right] \tag{3.6}$$

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

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

$$q\left(S_t, A_t\right) \leftarrow q\left(S_t, A_t\right) + \alpha\left[G_t^n - q\left(S_t, A_t\right)\right] \tag{3.8}$$

A (3.8) egyenletben leírt frissítési szabályon alapul az ún. Q-tanulás (*Q-learning*) algoritmus, amely az egyik legelső TD tanító módszerek egyike.[32] Ebben az esetben az állapotfüggvény egy mátrixal jellemezhető, ahol minden sor egy állapotnak, és minden oszlop egy akciónak feleltethető meg, tehát kizárólag véges állapot- és akciótér esetén alkalmazható. Ennek ellenére már rengeteg féle probléma megoldására sikeresen alkalmazták, például gyártásirányításban[30], de az autonóm közlekedés területén is lehet találni példákat[12]. Továbbá az algoritmus egyszerűségéből adódóan jól testreszabható (lásd a következő alfejezetet).

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 egy eredendően folytonos probléma, az itt bemutatott véges terek felett alkalmazha-

tó Q-tanuláshoz szükséges a feladattér egy diszkrét reprezentációját meghatározni. Az ezzel kapcsolatos kísérletek az 5.1-es fejezetben olvashatóak.

3.2. Felfedezési stratégiák

A megerősítéses tanulás egyik legfontosabb kérdése a felfedezési stratégia 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 *kiaknázás* azt jelenti, hogy a jelenlegi politika alapján történik az akció megválasztása, a *felfedezés* pedig az, amikor valamely más akció történik kiválasztásra, még akkor is, 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, esetleg magasabbat, mint a korábban feltételezett optimális művelet.

Számos algoritmus használ különböző módszereket a felfedezés szabályozására. Az egyik legegyszerűbb ismert stratégia az ún. ε-mohó (ε-*greedy*) felfedezés, amely esetében az ágens minden beavatkozás előtt egy $0 \leq \varepsilon \leq 1$ paraméterű Bernoulli valószínűségi változó alapján dönt (vagy másképpen megfogalmazva „érmét dob"), hogy a következő lépésben kiaknázzon-e vagy esetleg felfedezzen. Felfedezés esetén az akció választása többféleképpen is történhet, de a legegyszerűbb véletlen egyenletes eloszlás alapján ezt megtenni. Az ε paraméter lehet a tanítás során végig azonos vagy folyamatosan (pl. lineárisan vagy exponenciálisan) csökkenő. Ez a mohó stratégia roppant egyszerű, ugyanakkor sok hangolást igényel hiperparaméterek szempontjából.

A paraméter tuningolás elkerülése érdekében továbbá léteznek olyan felfedezési módszerek is, amelyek az ágens teljesítményétől függően, adaptívan állapítják meg a felfedezést vezérlő paramétereket (ami pl. az ε). Amellett, hogy a paramétereket ilyen esetekben nem kell „kézzel" finomhangolni, az adaptív funkciók segítik az ágens átképzését olyan esetekben, ahol a környezet a tanítás vagy az operáció közben megváltozhat (pl. driftelés esetén az úttest tapadási együtthatója). Ilyen stratégiát alkalmaz a megelőző munkákban[27, 26] már alkalmazott Soft Actor-Critic (SAC)[13] algoritmus is, azonban saját tapasztalatok alapján a differenciál entrópia alapú felfedezés a legtöbb esetben nem működik elég stabilan ahhoz, hogy megfelelő konvergenciát biztosítson az ágens számára. Ezért ebben a dolgozatban (a klasszikus ε-mohó stratégia mellett) egy olyan Q-tanulás által vezérelt adaptív felfedezés is alkalmazásra kerül, amely már ért el

sikereket a gyártás területén[30].

Lényegében a módszer alapja, hogy a cselekvő ágens mellé egy felfedező ágens is definiálásra kerül, amelynek az akciói előre meghatározott ε értékekből állnak. A cselekvő ágens minden lépése előtt közvetlen a felfedező ágens a jelenlegi állapot alapján kiválaszt egy felfedezési rátát, amely szerint történik majd az „érmedobás". Az ε kiválasztása az adott állapothoz tartozó értékek szerint generált valószínűség vektor alapján történik: minél közelebb van a cellában szereplő érték az adott állapotban elérhető maximális jutalomértékhez, annál nagyobb a cella oszlopát reprezentáló ε kiválasztásának valószínűsége. A felfedező ágens táblájának frissítése ezután hasolnóképpen történik, mint a cselekvő ágens esetében:

$$\mathcal{E}\left(S_t, \epsilon_t\right) \leftarrow \mathcal{E}\left(S_t, \epsilon_t\right) + \alpha\left[G_t^n - \mathcal{E}\left(S_t, \epsilon_t\right)\right] \tag{3.9}$$

ahol $\mathcal{E} : \mathcal{S} \rightarrow E$ a felfedező ágens értékfüggvénye és E a lehetséges ε paraméterek (véges) halmaza. Fontos megjegyezni, hogy a fent leírtaknak megfelelően G_t^n bootstrap tagja is megváltozik $\gamma^n \mathcal{E}_\pi\left(S_{t+n}, A_{t+n}\right)$-ra.

Tehát az fejezetet összefoglalva, a jelen dolgozatban bemutatott ágensek tanításához a tabuláris Q-tanulás fentebb ismertetett, két különböző felfedezési módszerrel is működtethető változata kerül alkalmazásra, amelyektől a legfontosabb elvárt eredmény, hogy kiküszöbölje a SAC algoritmusnál tapasztalt instabilitásokat, miközben képesek legyenek megvalósítani a driftelést.

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, differenciálegyenlet-alapú 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, amelyekkel párhuzamosan a megerősítéses tanulás alapú módszerek is. Egyes ezen kutatások és eredményeik kerülnek ismertetésre ebben a fejezetben.

4.1. Klasszikus 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 kotroller alkalmazásával[31], más esetekben egyetlen közös kontroller alkalmazása is sikeres volt[2], de ezek mellett előfordult a kormányzás rögzítése mellett csupán a gázadás és fékezés irányítása[28]. Ezen megoldások a valós járműveken történő alkalmazáshoz is jónak bizonyultak [31][3]. Mindezek mellett négykerekű modell felhasználásával is születtek pozitív szimulációs eredmények[29].

Az MPC (Modell Predictive Control) kontroller alkalmazása eddig szintén sikeresnek bizonyult stabilizációs és nyomvonal-követéses drift feladatokhoz egyaránt[7, 11], illetve változó útviszonyok szimulálásakor is helyt állt, jelezve ezzel egy jó adaptív tulajdonságot. Nyomvonal-követést is definiáló eredmények közül még érdemes megemlíteni a

lineáris kontroll és az MPC egy hibrid alkalmazását egy driftelve elvégezendő parkoló manőver megvalósításához[18]. Az előbbi konstrukció szimulációban és távirányítású autóval egyaránt működött.

4.2. Megerősítéses tanulás alapú megközelítések

Ígéretes irányt mutat a megerősítéses tanulás alkalmazása is az autonóm driftelés megoldására. A megerősítéses 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[6] a Probabilistic Inference for Learning Control (PILCO)[8] modell-alapú politika-kereső algoritmust használta a stabil állapotú driftelés eléréséhez szimulációban és egy kis méretű távirányítós járművön. Ezen eredményeket erősítve Bhattacharjee [4] végzett hasonló munkát a PILCO és a mély Q-tanulás (DQL) alkalmazásával.

A Cai és társai [5] a driftelést egy nyomvonalkövetési feladatként határozták meg a CARLA nevezetű szimulátorban, ahol a pontos cél nagy oldalcsúszási szögek elérése volt nagy sebességeknél. Az ágenst több előre megadott pályán tanították, majd egy addig számára ismeretlen pályán kiértékelték, több különböző típusú járművel és útviszony beállítással. A tanítás itt a Soft Actor-Critic algoritmussal történt, de emellett más szerzők [22] kísérleteztek ezen a feladaton a TD3 algoritmussal is, ami szintén jól működőnek bizonyult. Az eddig említett eredményeket továbbfejlesztve Domberg és társai [10] sikeresen kifejlesztettek egy olyan ágenst, amely képes akár tetszőleges trajektóriákon is driftelni, ámbár eddig még csak a szimulációs eredményeik ígéretesek. Mindezen felsoroltak jól mutatják a megerősítéses tanulás potenciális adaptív képességeit.

Egyes fent említett munkák[10, 5, 22] esetében fontos kiemelni, hogy a definiált feladat valójában nem drift állapotok pontos követésére irányul, hanem a cél a minél nagyobb oldalcsúszási szögek elérése. Bár az ilyen ágensek hasznos eredményt szolgáltathatnak az autonóm motorsport szempontjából, biztonságtechnikai értelemben kevésbé hasznosak, ugyanis olyan esetekben a fő szempont jól megtervezett manőverek precíz megvalósítása, amelyekhez szükséges a követendő nyomvonal mellett a drift állapotok definiálása is.

Az eddigi autonóm drift irodalomban még nem található olyan megoldás, amely esetében a folytonos környezetben definiált feladatot teljes mértékben diszkrét tereken értelmeznék az ágens számára, tehát ebből a szempontból a dolgozatban bemutatott eredmények egyedülállóak. Általánosságban is kijelenthető, hogy az autonóm járműirányításban leginkább a mély megerősítéses tanulás (DRL) terjedt el ezidáig[20]. Ennek legfőbb oka, hogy a folytonos környezetben működő diszkrét ágens sok esetben elégtelenül vagy túl darabosan hajtja végre a feladatot, amely általában a véges térreprezentáció

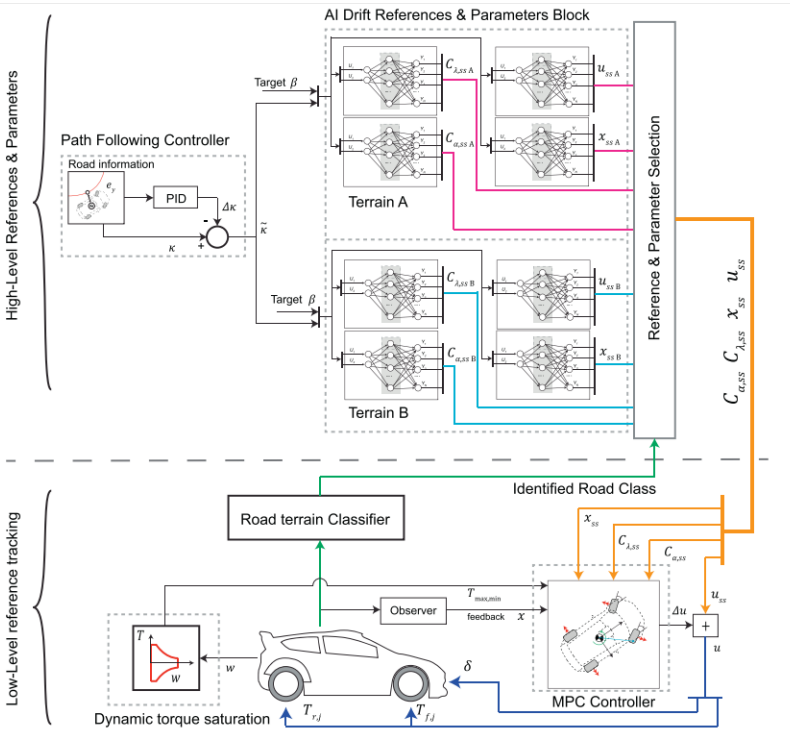
helytelen konstrukciója miatt van. Erről részletesebben az 5.1-es fejezetben esik szó.

4.3. A felügyelt tanulás és hierarhikus modellek alkalmazhatósága

A előbbiekben ismertetett módszerek 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 hierarhikus 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 útterep osztályozó, amely a referenciákat és a gumiparamétereket a különböző útfelületekhez igazítja, ahol más tapadási tényezők lépnek fel. A hálózatokat egy emberi sofőr szimulátorban való vezetés közben felvett adatai segítségével tanították.

A fent említett példa melett születtek még további hasonló munkák, pl. [19]. Az ilyen módszereknek a hátránya viszont, hogy emberi sofőr vagy egy előzetesen jól működő kontroller álltal rögzített adat felhasználása mindenképpen szükséges a hálók tanításához. Többek között ennek a problémának a megoldására is jól használható lehet a megerősítéses tanulás alkalmazása.



4.1. ábra. Neurális hálók alkalmazásával működő hierarhikus drift kontroller[1]

5. fejezet

Az állandósult drift feladat pontos leírása

A járműmodell és a megerősítéses tanulás alkalmazásához szükséges ismeretek átadása után immáron sor kerülhet a megoldandó drift feladat definiálására és megoldására. A cél továbbra is egy megadott drift egyensúlyi állapot megközelítése és megtartása tetszőleges időn keresztül. Ennek szimulációs megvalósításához definiálni kell a drift feladat paramétereit és implementálni a kiválasztott járműmodellt MATLAB/Simulink környezetben. Ezután következhet a helyes diszkrét térreprezentációk megtalálása, majd az ágensek tanítása.

5.1. A driftelés definiálása megerősítéses tanulás feladatként

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. Habár a β szög a driftelés fennállásának egyik közvetlen mértéke, nincs szükség rá, hogy az állapotváltozók közé vegyük, ugyanis a v_x és a v_y együttes ismerete már elég információt szolgáltat (lásd a (2.19)-es egyenletet). Tehát az állapottér a következőnek lett kiválasztva:

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

Az akcióteret a járműmodell bemeneti paraméterei 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 $\delta_{steer} \in [-420°, 420°]$ definiálható, ahol a pozitív értékek a bal oldali tartományt, a negatív értékek pedig a jobb oldali tartományt reprezentálják. Tehát pl. $\delta_{steer} = 90°$ azt jelenti, hogy a kormánykerék 90 fokkal balra van forgatva. A kormányszög kerékszöggé való átalakítása a következő:

$$\delta = \frac{\delta_{steer}}{15} \cdot \frac{\pi}{180} \tag{5.2}$$

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

$$\mathcal{A} := \{A = (ped_{acc}, \delta_{steer}) \,|ped_{acc} \in [0, 1] \,, \delta_{steer} \in [-420°, 420°]\} \tag{5.3}$$

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 célállpot megtartása, ezért annak mindenképpen szerepelnie kell a jutalom értékében. A célállapotot a 2.3 szekcióban leírtak alapján lehet kiválasztani. Ehhez a $\delta = -10\,rad$ és $v_x = 10^{\mathrm{m}}/_{\mathrm{s}}$ értékek kerültek kiválasztásra, és ebből a $v_y = -3.4812^{\mathrm{m}}/_{\mathrm{s}}$, $r = 0.8334^{\mathrm{rad}}/_{\mathrm{s}}$ és $F_{x_r} = 3747.8719\,N$ eredményt kaptuk. Tehát a drift célállapot:

$$S_{drift} = \left(v_{x_{drift}}, v_{y_{drift}}, r_{drift}\right) = (10^{\mathrm{m}}/_{\mathrm{s}}, -3.4812^{\mathrm{m}}/_{\mathrm{s}}, r = 0.8334^{\mathrm{rad}}/_{\mathrm{s}}) \tag{5.4}$$

Ebből a jutalomfüggvény a következőként lett meghatározva:

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

ami a következő állapot és a célállapot euklideszi távolságának $-\frac{1}{3}$-szorosa. Tehát az ágens azzal arányos „büntetést" kap, amilyen messze van a munkaponttól.

A jutalomfüggvény mellett továbbá definiálásra került egy indikátorfüggvény, amelynek szerepe, hogy leegyszerűsítse annak megállapítását, ha az ágens egy preferált drift tűréshatáron belülre helyezi a járművet:

$$Isdrift\,(S_t) = \begin{cases} 1 & \left|\frac{S_{t_i}}{S_{drift_i}} - 1\right| < 0.1 \forall i \in \{1, 2, 3\} \\ 0 & \textit{különben} \end{cases} \tag{5.6}$$

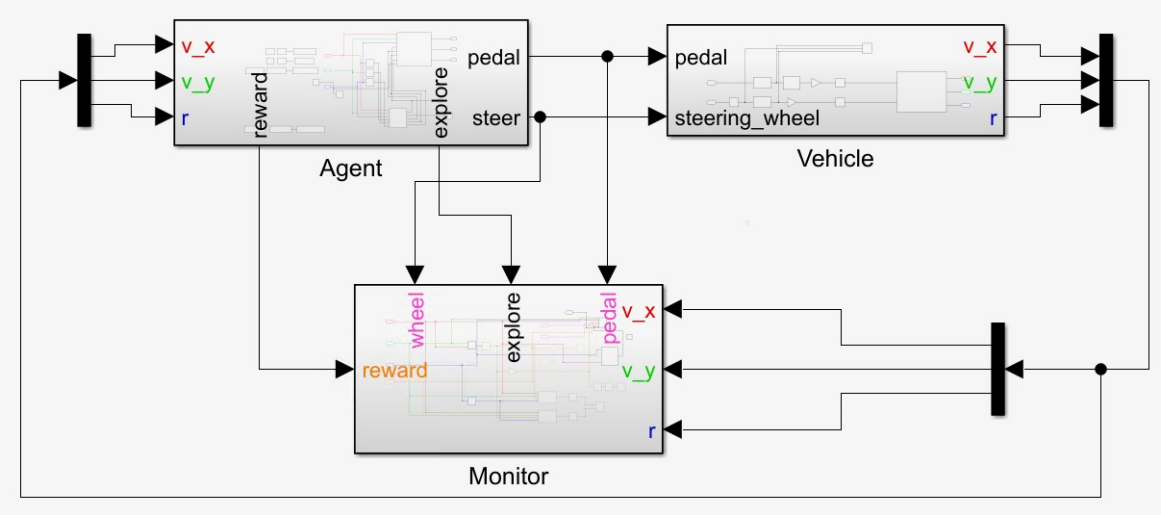
A függvénynek van továbbá egy másodlagos validációs funkciója is, ugyanis a juta-

lomfüggvény négyzetes hibát reprezentál, amíg az „isdrift” függvény az abszolút távolságon alapul.

5.2. A jáműmodell és az ágens implementációja MATLAB/Simulink környezetben

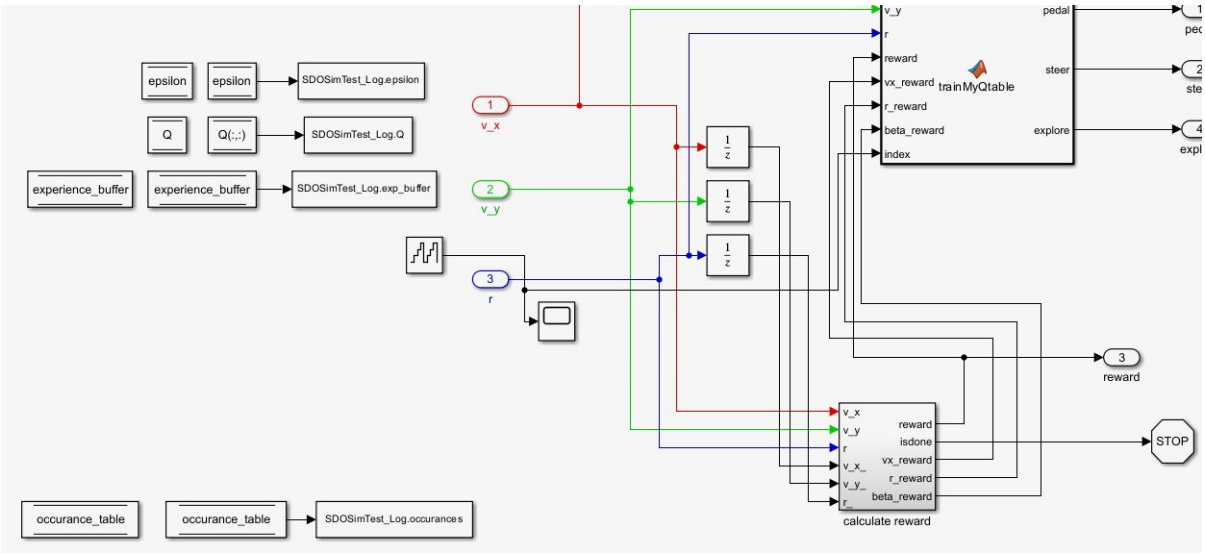
Ebben a szekcióban a Simulink modell részletei kerülnek bemutatásra.

A modell különböző szintekből és alrendszerekből 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.



5.1. 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).

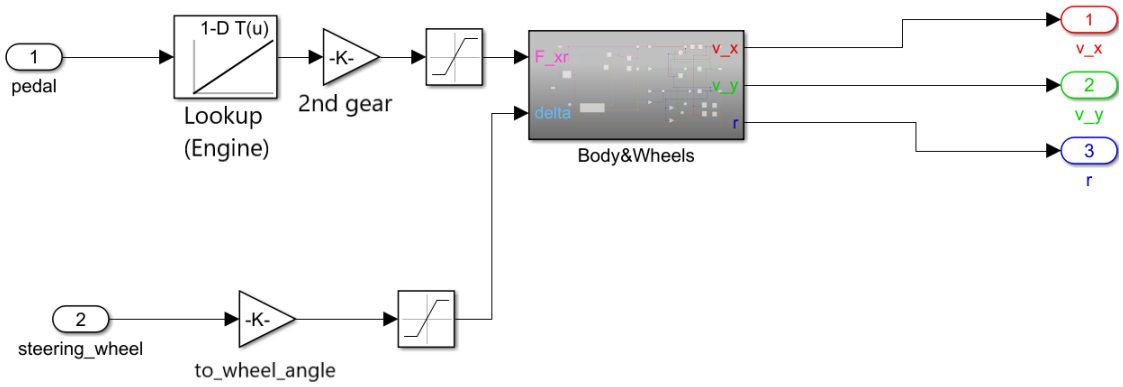
Az „Agent” rendszer tartalmazza az ágens működéséhez szükséges blokkokat. Ebbe tartozik maga az ágenst reprezentáló (MATLAB függvény) blokk, a jutalmat kiszámító „calculate reward” alrendszer, valamint a Q-táblákat és az egyéb tanítási metaadatokat tároló és kezelő memóriablokkok (ezeket lásd 5.2-es ábrán bal oldalon). A rendszer bemenetei az állapotparaméterek, kimenetei pedig az ágens által leadott akciók.



5.2. 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_{x_r}. Ezeken felül a δ_{steer} kormányszög 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.



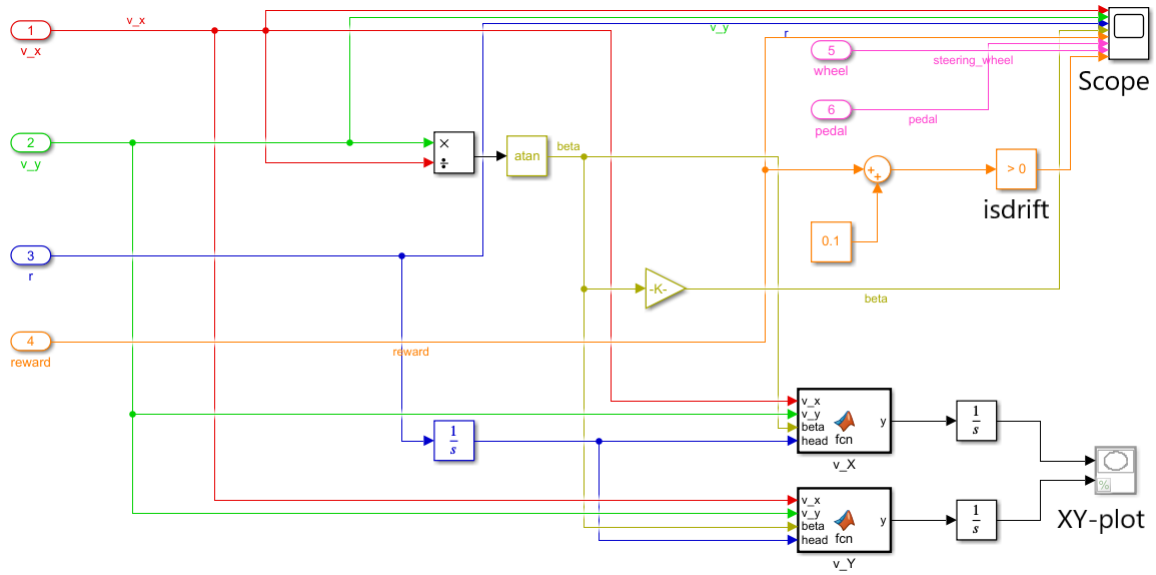
5.3. ábra. A járműmodell rendszere a Simulink modellben.

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 inerchiája	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	végső áttételi arány	3.465	-

5.1. 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 állapotváltozók, a β csúszásszög, az akcióváltozók, a jutalom és az „isdrift” változó időbeli eloszlását az adott epizódon belül. 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 ki. 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. Az eredmények tárgyalásakor (többek között) ezen grafikonok által szolgáltatott adatok kerülnek majd elemzésre.



5.4. ábra. A „Monitor” rendszer.

6. fejezet

Kísérletek és eredmények

Ismétlésként, a dolgozatban ismertetett kutatás célja egy olyan önvezető ágens kifejlesztése, amely képes kialakítani és fenntartani egy megadott céldrift állapotot. Az ágensek tanításához kétféle különböző tabuláris Q-learning algoritmus került implementálásra. Az elvégzett tanítási kísérletek során elsősorban a hiperparaméterek helyes megválasztása és a megfelelő térreprezentációk megtalálása volt a feladat. A tanítás minden esetben epizódikus, legalább $T = 5s$-os leállási idővel, hogy az ágens számára elegendő idő álljon rendelkezésre a drift állapot megközelítéséhez. Minden epizód az

$$S_0 = (9^{m}/_{s}, 0^{m}/_{s}, 0^{rad}/_{s}) \tag{6.1}$$

kezdőállapotból indul. A két algoritmus közös hiperparaméterei a 6.1-es táblázatból olvashatóak le.

Jelölés	Név	Optimális Érték
s_{time}	az ágens beavatkozási rátája ($[s]$)	0.1
n	a TD megtérülés előrelátási paramétere (lásd (3.5))	1
α	tanulási ráta	0.5
γ	diszkunt ráta	0.75

6.1. táblázat. A tanító algoritmusok közös hiperparaméterei.

6.1. A folytonos környezet megfelelő véges reprezentációjának megtalálása

Az 5.1-es alfejezetben definiált folytonos környezetreprezentáció a feladat általános megfogalmazásaként értelmezhető, viszont jelen formájában csak folytonos tanulóalgoritmus számára használható. Mivel a 3. fejezetben már bemutatott Q-tanulás csak véges feladatokra alkalmazható, ezért szükség van az állapot- és az akciótér egy véges reprezentációjára.

Az akciótér véges megfogalmazása tulajdonképpen nem más, mint a már definiált intervallumból kiválasztani véges számú értéket, amelyeket az ágens akcióként választhat. Az állapottér esetében ez már bonyolultabb, mivel az eredeti állapottér nem korlátos, továbbá szükség van egy olyan szabályra is, amely az autó állapotteréről képez az ágens állapotterére. A határok megválasztásakor ha figyelembe vesszük a már definiált célállapotot ((5.4)-es egyenlet), két dolog figyelhető meg: egyrészt a cél egy közepes sebességű drift fenntartása, így a túl alacsony, illetve túl nagy sebességek tekinthetőek egyenlően helytelennek, másrészt a jármű ebben az állapotban balra kanyarodik, így a jobbra kanyarodó mozgásálapotok is azonosan rossznak vehetők. Ezek alapján a következő határok állapíthatóak meg:

$$5^{m}/_{s} \leq v_x \leq 15^{m}/_{s} \tag{6.2}$$

$$-5^{m}/_{s} \leq v_y \leq 0^{m}/_{s} \tag{6.3}$$

$$0^{rad}/_{s} \leq r \leq 1^{rad}/_{s} \tag{6.4}$$

A definiált állapotkonverziós szabály pedig a következő:

$$S_{t_d} = \text{argmin}_{S_d \in \mathcal{S}_d} \left(|S_t - S_d| \right) \tag{6.5}$$

ahol $S_t \in \mathcal{S}$ a jármű jelenlegi állapota és $\mathcal{S}_d$ a véges állapottér. Tehát a jelenlegi állapot diszkrét megfelelője a hozzá lévő (abszolút távolság szerint) legközelebbi $\mathcal{S}_d$-beli pont.

A véges terek előállításához ezután már csak az intervallumok megfelelő beosztása van hátra, amelyek sűrűsége hiperparaméterként kezelhető, tehát ennek beállítását a futtatások során hangolni kell. A következő két szekcióban elvégzett kisérletek során megtalált legeredményesebb beosztás a következő:

$$\mathcal{S}_d = V_{x_d} \times V_{y_d} \times Yaw_d \tag{6.6}$$

$$V_{x_d} = \{5^{m}/_{s}, 6^{m}/_{s}, \ldots, 15^{m}/_{s}\} \tag{6.7}$$

$$V_{y_d} = \{-5^{m}/_{s}, -4.5^{m}/_{s}, \ldots, 0^{m}/_{s}\} \tag{6.8}$$

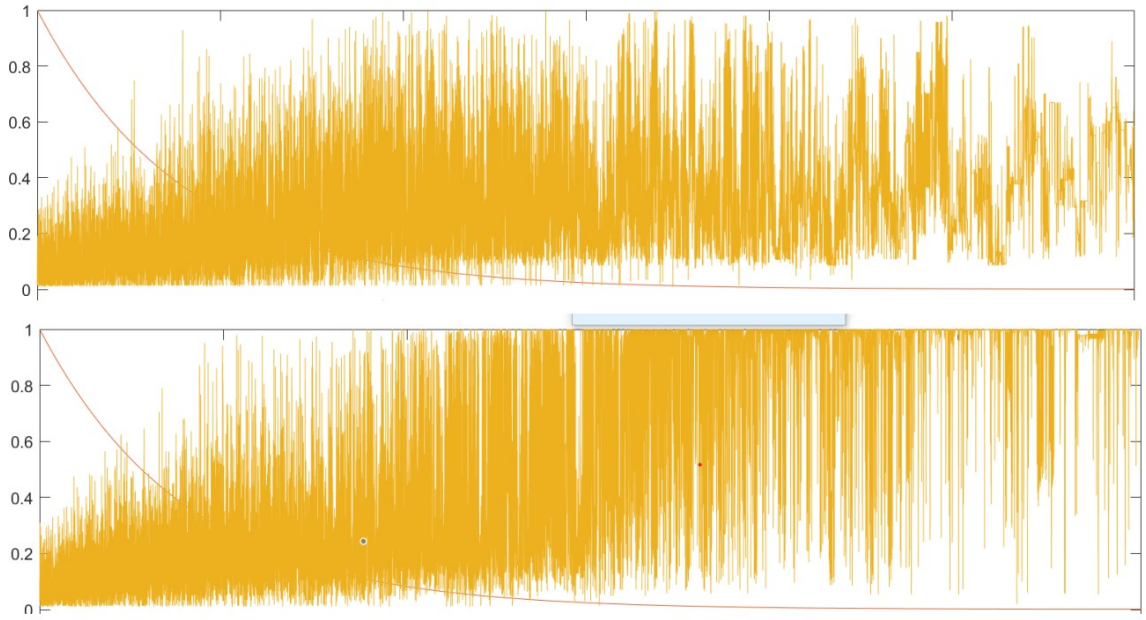
$$Yaw_d = \{0^{rad}/_{s}, 0.1^{rad}/_{s}, \ldots, 1^{rad}/_{s}\} \tag{6.9}$$

$$\mathcal{A}_d = \mathcal{P}_d \times \Delta_d \tag{6.10}$$

$$\mathcal{P}_d = \{0, 0.1, \ldots, 1\} \tag{6.11}$$

$$\Delta_d = \{-200°, -170°, \ldots, -20°, 0°, 10°, 40°, 70°, 100°\} \tag{6.12}$$

ahol lehetséges kormánykerék állásokhoz, a célállapotot figyelembe véve, az eredeti tartomány $[-200°, 100°]$ részinterrvalluma került beosztásra, ezzel is csökkentve az akciótér számosságát. Az ezek után kapott Q-táblák 1331x122-es méretűek, azaz 1331 állapot és 122 akció került definiálásra. Ennél ritkább reprezentációk (6.1-es ábra) esetén az ágens teljesítménye erősen romlott, míg a sűrűbb beosztások nem eredményeztek számottevő javulást, viszont a sikeres tanításhoz szükséges epizódok száma jelentősen megnövekedett.



6.1. ábra. A (6.6)-(6.12) egyenletekkel leírt (alul) és egy nála ritkább (felül) véges térreprezentáció teljesítményének összehasonlítása. A görbéről az 'isdrift' indikátor epizódra vetített aránya olvasható le (azaz, hogy az adott epizód mekkora hányadában jelzett az indikátor). Mindkét tanítás esetén az egyéb hiperparaméterek beállítása azonos. A vékony narancs görbe a felfedezési arány csökkenését mutatja.

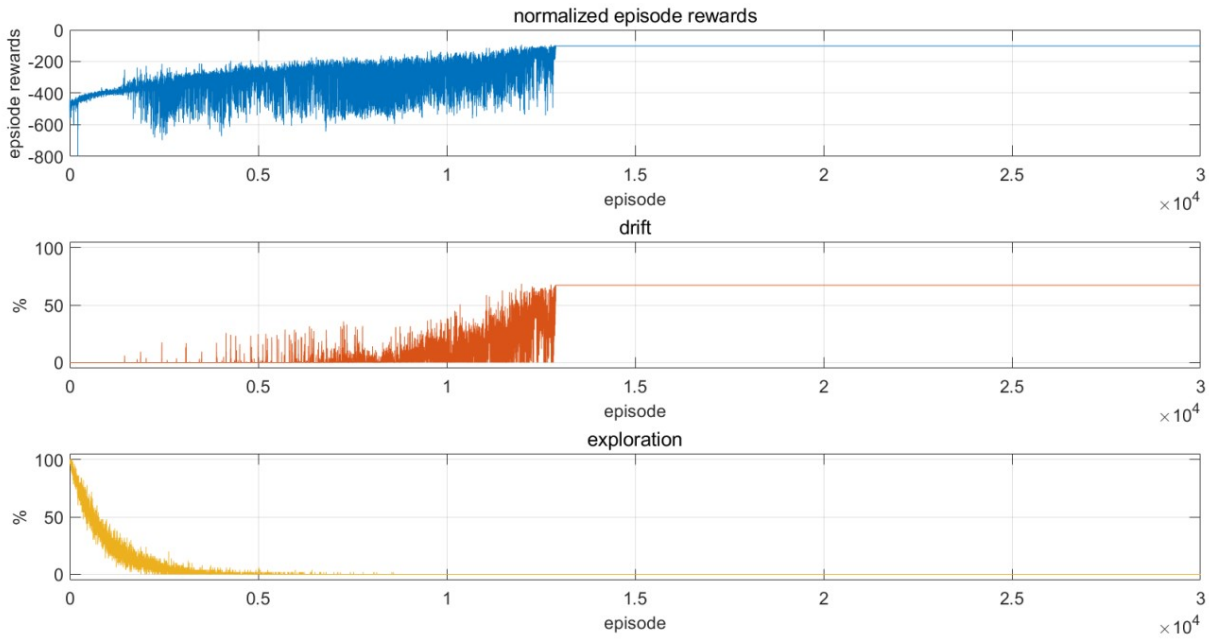
6.2. A driftelés megvalósítása ε-mohó felfedezéssel

A 3.2-es alfejezetben ismertetett ε-mohó felfedezési stratégia exponenciálisan csökkenő paraméterű változata került implementálásra, amely esetében a felfedezési ráta minden tanítási lépést követően frissül a következő szabály szerint:

$$\varepsilon = \varepsilon \left(1 - \varepsilon_{decay}\right) \tag{6.13}$$

ahol ε_{decay} a csökkenés mértékét szabályozó paraméter. A tanítás minden esetben $\varepsilon = 1$ kezdeti feltétellel indul, aminek csökkenése a tanítás végéig tart, azaz a paraméternek nincs alsó korlátja. A Q-tábla inicializáltan a csupa 0 mátrix.

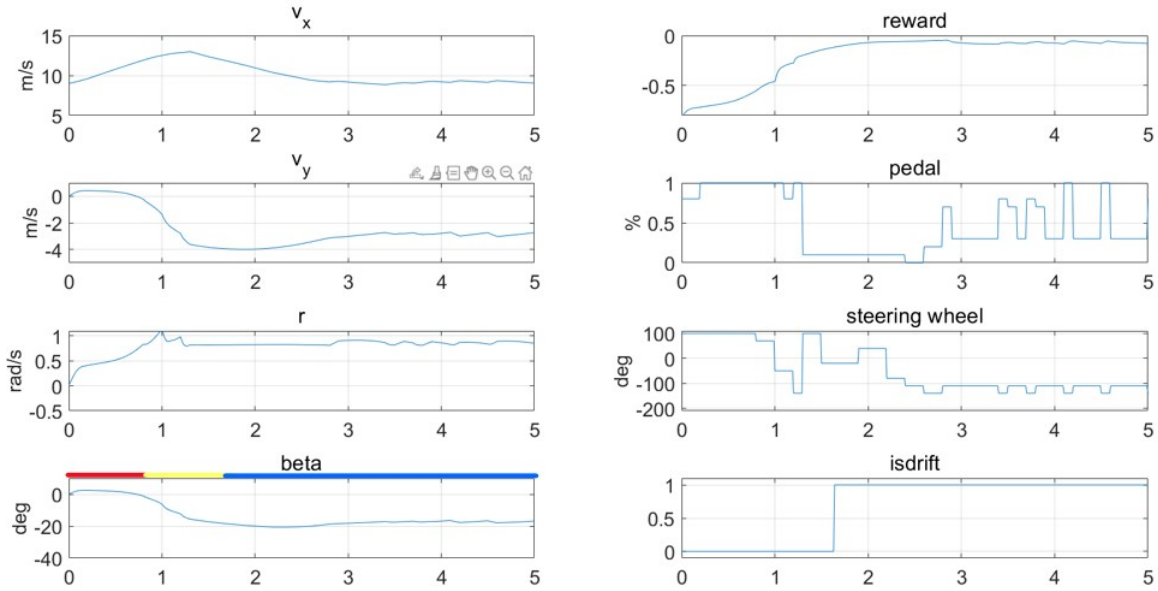
A sikeres tanítást mutató görbék a 6.2-es ábráról olvashatók le: 12900 epizódot követően sikerült megtalálni az elérhető optimumot, amivel az ágens az 5mp-es epizód 67.26%-án, azaz 3.363mp-en keresztül képes a definiált céldriftet fenntartani. Ennél nagyobb számok elérése 5 másodperc alatt (egy elhanyagolhatóan kicsi mérték kivételével) nem lehetséges, ugyanis a kiinduló helyzet nem drift állapot, és a járműnek fizikailag szükséges egy bizonyos időtartam, hogy a driftelést létrehozza. A felfedezési arányokon látható, hogy a nagyrészt már csak kiaknázást tartalmazó, 6000-13000 epizódok közötti tanítási szakaszon is még jelentősen javul az ágens teljesítménye, ami legfőképp a Q-tábla megfelelő inicializálásának köszönhető: mivel a jutalomfüggvény globális maximuma 0 és a felfedezés során még ki nem próbált állapot-akció párok értékei mind nullák, így az algoritmus kiaknázás esetén előnyben részesítheti ezeket, ezáltal egy másodlagos, „belső" felfedezést is biztosítva.



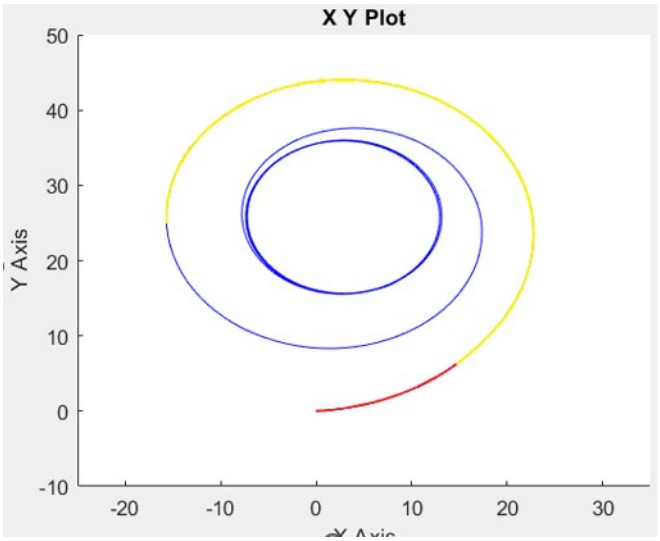
6.2. ábra. Sikeres tanítás a drift megvalósítására ε-mohó felfedezési stratégiával, kék színnel jelölve az epizód során megszerzett kommulatív jutalmakat, pirossal a százalékos 'isdrift' arányokat, és sárgával a felfedezési arányokat.

A 6.3-as ábrán egy scope grafikon látható, ami mutatja az állapotváltozók, a jutalom, az indikátor és az ágens által leadott akciók alakulását. A görbéket megfigyelve észrevehető, hogy a driftelő mozgás jóval azelőtt megvalósul, hogy a definiált indikátor driftelést jelezne. Ez abból látszik, hogy a β szög röviddel 1 mp előtt már negatív tartományban mozog, miközben az r értéke továbbra is pozitív. Ez annyit jelent, hogy a definiált célállapot nem csupán a kezelhetőségi határon, hanem azon jóval túl helyez-

kedik el. Továbbá, az akciók alakulását megnézve megállapítható, hogy az ágens képes egész precízen megválasztani azokat, nem tapasztalható jelentős „rángatás”, ami az ágens diszkrét természetét figyelembe véve egy meglepően jó eredmény. Ennek ellenére a későbbiek során az aktuátorok dinamikáinak komolyabb szabályozása feltétlen szükséges lesz bonyolultabb feladatok megoldása és valós járművön történő tesztelés céljából.



6.3. ábra. A sikeres tanítás eredményeképpen kapott ágens teljesítménye scope ábrán.



6.4. ábra. Az ágens által irányított autó mozgásnyomvonala 30mp során.

A 6.4-es ábrán az ágens által irányított jármű szerint megtett trajektória látható egy 30mp hosszú szimuláció esetében. A görbe piros részén még nem jelentkezik driftelés, a sárga részen már megvalósul az oldalcsúszás, és a kék részen már jelez az indikátorfüggvény is. Ez a színkódolás a 6.3-as ábrán is, az oldalcsúszási szög görbéje felett is látható.

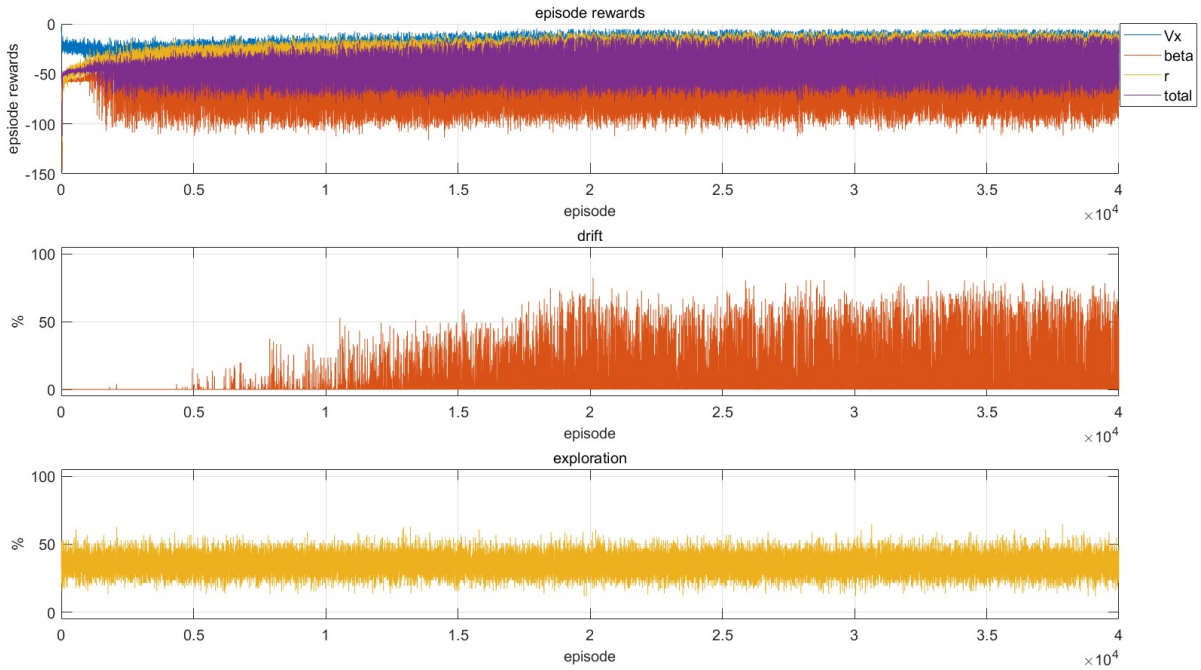
Összefoglalva, az egyszerű ε-mohó felfedezéssel működtetett tabuláris ágens sikeresen végrehajtotta a kitűzött feladatot, továbbá az előző fázisban tapasztalt tanítási instabilitásokat is sikerült kiküszöbölni, minden tanítás azonos végeredménnyel megismételhető, és a felfedezési stratégia nem okoz teljesítménybeli problémákat. Azonban fontos megjegyezni, hogy a felfedezést vezérlő paraméterek optimális értékeinek megtalálása jelentős időt vett igénybe, ami egy bonyolultabb feladat (pl. változó tapadási viszonyok) esetén komoly nehézségeket is okozhat.

6.3. Az adaptív felfedezéssel kapcsolatos eredmények

Az ε-mohó módszernél tapasztalt hiperparaméter keresésből adódó nehézségek kiküszöbölésére és az ágens adaptív képességeinek erősítésére implementálásra és tesztelésre került a már részletesen bemutatott (3.2) Q-tanulás alapú önszabályozó felfedezés is. A felfedező ágens számára szükséges egy „akciótér", azaz egy lehetséges felfedezési rátákat tartalmazó halmaz, ami a következőképpen lett definiálva:

$$\mathcal{E} = \{0, 0.05, 0.15, 0.25, 0.5, 1\} \tag{6.14}$$

Tehát a tanítás során most egy univerzális ε érték helyett egy másik, 1331x6-os méretű Q-tábla vezérli a felfedezést. Mindkét Q-tábla inicializáltan csupa -1 mátrix. Minden egyéb, az előző módszerrel közös hiperparaméter azonosan lett megválasztva.

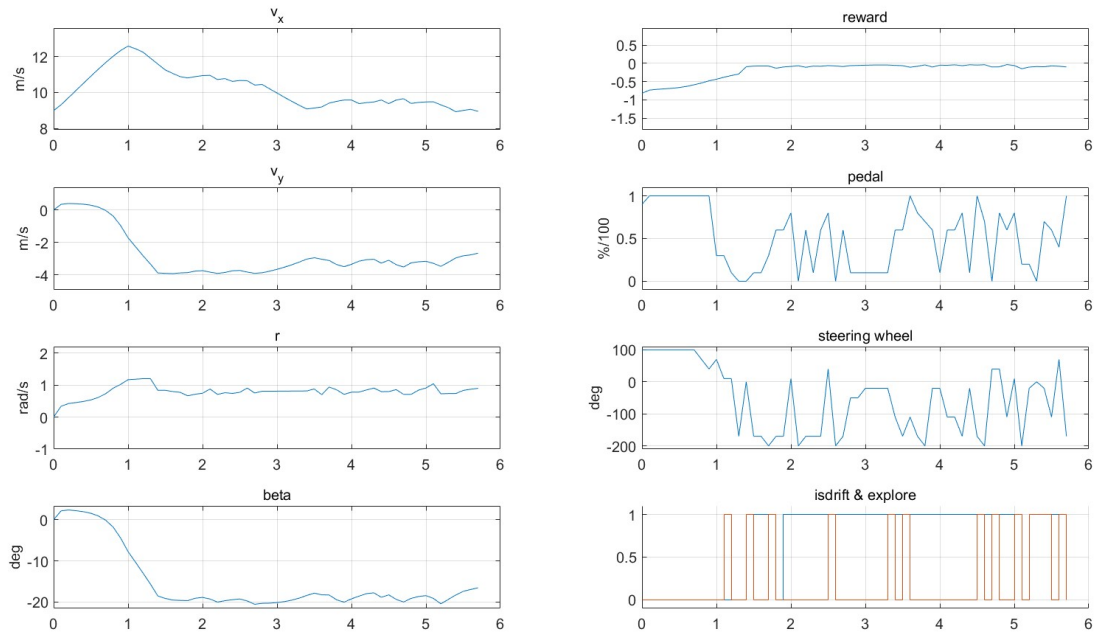


6.5. ábra. Tanítás adaptív felfedezéssel. A kommulatív epizód jutalmak esetén külön színekkel láthatóak az egyes komponensekből származó értékek is.

A tanítást mutató 6.5-ös ábrán az első észrevehető dolog, hogy a felfedezések aránya 20-40% között oszcillál, tehát a definiált értékek átlaga (32,5%) körül. Megvizsgálva a felfedező Q-tábla értékeinek alakulását a tanítás során, az állapotok döntő többségében az tapasztalható, hogy a definiált ráták értékei végig viszonylag közel vannak egymáshoz, ezért a kiválasztásuk is közel azonos valószínűséggel történik meg (6.6-os ábra). Hosszas vizsgálatok után vélhetően ennek legfőbb oka, hogy a valószínűségek megválasztásakor a ráta értékek 0-hoz viszonyított távolsága vétetik figyelembe, azonban a legtöbb állapotban a 0 érték elérése még optimális akciók esetén is lehetetlen, hiszen a jutalom a munkaponttól való távolság, ami nyilvánvalóan a legtöbb állapotban nem 0. Mindezek ellenére az ágens képes közel optimális teljesítményt elérni (6.7-es ábra), méghozzá stabil, megismételhető konvergenciával, viszont az átlagosan nagy valószínűséggel előforduló felfedezések miatt a legtöbb esetben nem tudja sokáig fenntartani a driftelést, valamint az akcióknál emiatt erős „rángatás” is tapasztalható.

```
choosing exploration:
epsilon  table value  probability
0.0000  -0.2737   0.1704
0.0500  -0.2857   0.1632
0.1500  -0.2467   0.1891
0.2500  -0.3301   0.1413
0.5000  -0.2702   0.1726
1.0000  -0.2853   0.1635
random probability variable: 0.1279
--> epsilon = 0.0000
exploration = 0
```

6.6. ábra. A felfedezési ráta egy megválasztása. Középső oszlopban az egyes ε értékekhez tartozó Q-értékek, a jobb szélső oszlopban pedig az azokból levont valószínűségek.



6.7. ábra. Az adaptívan felfedező ágens teljesítménye scope ábrán. Az 'isdrift' értékekkel közös ábrán látható barnás görbe a felfedezés indikátor.

7. fejezet

Következtetések és kitekintés

A jelen TDK dolgozatban bemutatott kutatásban egy megerősítéses tanulás alapú ágens kifejlesztése történt drift mozgás létrehozásának és stabilizálásának céljából tabuláris Q-tanulás és egy sikbeli egypályás járműmodell segítségével egy MATLAB/Simulink szimulációs környezetben. A tanulóalgoritmus számára két különböző felfedezési stratégia is implementálásra került. A tanult ágensek a célállapot megközelítését mindkét esetben sikeresen végrehajtották. A mohó felfedezés lehetővé tette az állapot fenntartásához szükséges információ megtanulását is, azonban az ehhez szükséges helyes hiperparaméterek megtalálása sok kutatási munkát vett igénybe. Az adaptív stratégia ugyan túl sok felfedezést eredményezett, de ennek okát sikerült megtalálni, és a továbbiakban van lehetőség ennek kijavítására. Az előző fázisban tapasztalt tanítási instabilitásokat továbbá sikerült kiküszöbölni, minden tanítás azonos végeredménnyel megismételhető, és a felfedezési stratégiák nem okoznak olyan teljesítménybeli problémákat, mint amilyen pl. a Soft Actor-Critic algoritmusnál tapasztalt katasztrofális elfelejtés („chatastrophic forgetfullness") volt.

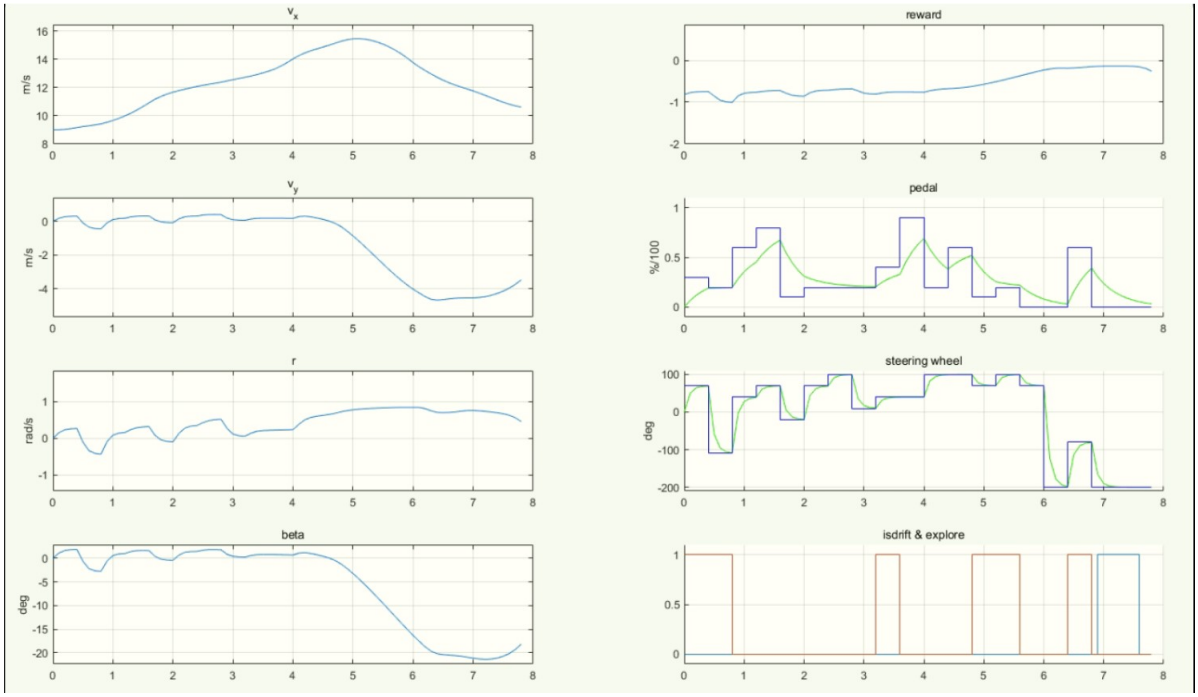
A pozitív eredményeken alapulva a jelen kutatás folytatódik, többek között az adaptív algoritmus hibáinak kijavításával és aktuátorokat szabályozó dinamikák hozzáadásával. Ezen kisérleteket követően, amennyiben az ágens továbbra is ígéretes teljesítményt mutat, sor kerülhet a valós járművön történő tesztelésekre a ZalaZONE tesztpályán.

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.

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

7.1. Kitekintés

Az aktuátor dinamikák szabályozásával kapcsolatos kutatási munkák eddigi eredményei kerülnek most rövid bemutatásra. Az aktuátor szabályozás jelen esetben azt jelenti, hogy mindkét akció input jeléhez egy késleltető, egytárolós arányos tag kerül, továbbá a kormánykerékhez egy forgatási sebesség limiter is, ezzel szimulálva egy valós jármű szabályozásának körülményeit. Az eddigi kísérletek az önszabályozó felfedezéssel azt mutatják, hogy az algoritmus képes alkalmazkodni a feladatot nehezítő körülményekhez, de csak az előrelátás mértékének (azaz a TD(n) paraméter) növelésével. Egyes esetekben továbbá az is tapasztalható, hogy a limitált akciók alakulása érdemben csak akkor éri el az ágens által kívánt célértékeket, ha annak beavatkozási időrátáját megfelelően megnöveljük (7.1-es ábra). Ennek hiányában ugyanis előfordulhat, hogy az ágens túl előre tervez, így idő előtt beavatkozik a folyamatba, amivel olykor semmissé teszi az eddig kiadott akciók hatását. A beavatkozási ráta megnövelésének viszont lehet egy olyan hátránya, hogy a drift fenntartásához szükséges precizitási képességét elveszíti.



7.1. ábra. Az adaptív algoritmus teljesítménye aktuátor dinamikák hozzáadásával, TD(7) és 0.4mp-es beavatkozási ráta esetén. A zöld görbék a valójában leadott akcióértékek.

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 kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt 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.

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. 4.3, 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, 2.3, 4.1, 5.1

[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, 2.3, 4.1

[4] Sourav Bhattacharjee, Kanak Dipak Kabara, Rachit Jain, and Kanak Kabara. Autonomous drifting rc car with reinforcement learning. *Dept. Comput. Sci., Univ. Hong Kong, Tech. Rep*, 2018. 4.2

[5] 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, 4.2

[6] 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. 4.2

[7] 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

[8] 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. 4.2

[9] 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

[10] Fabian Domberg, Carlos Castelar Wembers, Hiren Patel, and Georg Schildbach. Deep drifting: Autonomous drifting of arbitrary trajectories using deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7753–7759. IEEE, 2022. 1, 4.2

[11] Ádám Domina and Viktor Tihanyi. Ltv-mpc approach for automated vehicle path following at the limit of handling. *Sensors*, 22(15):5807, 2022. 4.1

[12] Laura García Cuenca, Enrique Puertas, Javier Fernandez Andrés, and Nourdine Aliane. Autonomous driving in roundabout maneuvers using reinforcement learning with q-learning. *Electronics*, 8(12):1536, 2019. 3.1

[13] 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. 3.2

[14] R Hadekel. The mechanical characteristics of pneumatic tires. *Clearinghouse Fed Sci & Tech Info*, 1952. 2.2

[15] 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

[16] Rami Yusef Hindiyeh. *Dynamics and control of drifting in automobiles*. PhD thesis, Stanford University, 2013. 2.2, 2.2, 2.2, 2.3

[17] Reza N Jazar. *Advanced vehicle dynamics*. Springer, 2019. 2.1, 2.1

[18] 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

[19] 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. 4.3

[20] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 4.2

[21] 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

[22] László Orgován, Tamás Bécsi, and Szilárd Aradi. Autonomous drifting using reinforcement learning. *Periodica Polytechnica Transportation Engineering*, 49(3):292–300, 2021. 4.2

[23] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992. 2.2

[24] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan 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. 3

[25] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. 3, 3.1

[26] Szilárd Hunor Tóth, Ádám Bárdos, and Zsolt János Viharos. Initiation and stabilization of drifting motion of a self-driving vehicle with a reinforcement learning agent. In *The First Conference on ZalaZONE Related R&I Activities of Budapest University of Technology and Economics 2022*, pages 53–57. Budapest University of Technology and Economics, 2022. 3.2

[27] Szilárd Hunor Tóth, Zsolt János Viharos, and Ádám Bárdos. Autonomous vehicle drift with a soft actor-critic reinforcement learning agent. In *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000015–000020. IEEE, 2022. 3.2

[28] Efstathios Velenis, Emilio Frazzoli, and Panagiotis Tsiotras. On steady-state cornering equilibria for wheeled vehicles with drift. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3545–3550. IEEE, 2009. 4.1

[29] 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

[30] Zsolt J Viharos and Richárd Jakab. Reinforcement learning for statistical process control in manufacturing. *Measurement*, page 109616, 2021. 3.1, 3.2

[31] 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, 2.3, 4.1

[32] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989. 3.1

[33] 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

[34] Tong Zhao, Ekim Yurtsever, Ryan Chladny, and Giorgio Rizzoni. Collision avoidance with transitional drift control. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 907–914. IEEE, 2021. 1