Budapesti Műszaki és Gazdaságtudományi Egyetem

Közlekedésmérnöki és Járműmérnöki Kar

Anyagmozgatási és Logisztikai Rendszerek Tanszék

Tudományos Diákköri Konferencia 2015

Kísérleti vezetőnélküli targonca modell neurális hálózat alapú navigációjának fejlesztése

Készítette:

Rácz-Szabó András

Konzulens:

Dr. Bohács Gábor

Budapest, 2015.10.19

1. Tartalomjegyzék

Tartalom

2. Bevezetés

Napjainkban az automatizált berendezések az üzemeken belüli anyagáramlási rendszerekben is egyre nagyobb teret nyernek. Ide tartoznak a vezetőnélküli targoncák is. Ezek üzeme során egyre intelligensebb funkciók megvalósítása válik szükségessé. A dolgozat a vezetőnélküli targoncás rendszerek egyik fontos funkciójára, az egységrakomány felvételre ad megoldást. A megvalósított rendszer neurális hálózatos, betanítás alapú rendszert jelent, melyet egy Lego Mindstorms építőkészletből készült mobil roboton valósítottuk meg, amely egy vezetőnélküli targonca modellje.

Lego Mindstorms Ev3 robot navigálása egy Omron F210 típusú képfeldolgozó rendszerből nyert koordináták segítségével történik. A roboton egy fekete kört és egy körívet helyzetünk el, míg a célterületet egy, a roboton lévőnél nagyobb méretű körgyűrűvel jelöltük, ezen alakzatok koordinátáit küldte el soros porton keresztül az F210 képfeldolgozó rendszernek. A két eszköz között Matlab segítségével történt a kommunikáció és az adatok elemzése. A beérkezett és a Matlab segítségével feldolgozatott adatokat egy neurális hálózatos rendszer dolgozta fel és tanította be a robotnak. Ennek eredményeképpen a robot sikeresen eljutott a célterületre, függetlenül attól, hol helyeztük el a kamera látóterében. A dolgozathoz tartozik még egy témakutatás vizuális irányítás témakörben, mely bemutatja, hogy eddig milyen hasonló megoldások születtek a témában.

Maga a kutatás folytatható, az eredmények tesztelhetőek vezetőnélküli targoncán, de ezen dolgozat csak a roboton történő munkálatokat és eredményeket mutatja be. A vezetőnélküli targoncák egy vezetővonalat követnek, mely lehet vizuális, passzív vagy aktív. Utóbbi két esetben fizikai vezetővonalról beszélünk, melyek könnyen sérülhetnek illetve nehezen vagy időigényesen lehet áthelyezni őket egy esetleges gyártósori változás esetén. A mi megoldásunk a vizuális vezetővonal előnyeit összesíti, vagyis rugalmasan kezeli a gyártósori változásokat, de nagy előnye a vizuálishoz képest, hogy a neurális hálózatos rendszer segítségével nincs szükség vezetővonalra, maga az eszköz „találja ki” a megfelelő utat a célhoz, függetlenül attól, hol helyezzük el a kamera látóterében. Jelen esetben konkrétan az egységrakomány felvételi problémát közelítettük ezzel a módszerrel, mivel a megoldáshoz szükséges szabályok nem ismertek vagy bonyolultak, illetve sok változót tartalmaznak, ezért lehetséges, hogy

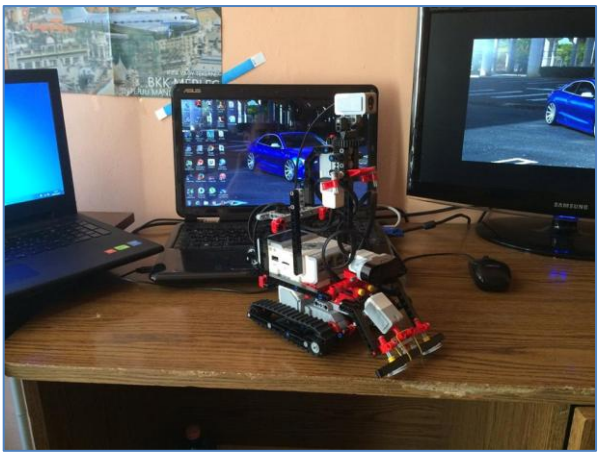
egyszerűbben lehet jobb eredményre jutni a neurális hálózatokkal, mint a „hagyományos” algoritmikus megoldásokkal.

Az eredmények a BME ALRT kutatásaiban közvetlenül hasznosításra kerülnek. A tanszéken elkészült a KTIA-AIK-12-1-2013-0009 projekt keretén belül egy bemutató rendszer, melyben az építkezések belső szállítási folyamatait egy vezetőnélküli targonca modellezi. A dolgozatban kifejlesztett funkció ebben a rendszerben is használható az anyagátadás megvalósítására. Ezen kívül a funkciót intralogisztikai rendszerekben, elsősorban termelés-kiszolgálást végző vezetőnélküli targoncák fejlesztéséhez is fel kívánják használni a jövőben.

3. Lego Mindstorms EV3 robot és megoldások saját plattformon

A LEGO MINDSTORMS egy programozható robotikai építőkészlet, melyet a Lego fejlesztett ki. Az EV3 a platform 3. generációja, mely egy programozható intelligens építőelem (PLC), amely irányítja a motorokat és érzékelőket, valamint vezető nélküli kommunikációt biztosít Bluetooth-on és Wi-Fi-n keresztül. A robot alapvetően a saját EV3 szoftver segítségével lehet egyszerűen programozni, de bonyolultabb feladatok megvalósítására nem alkalmas. A robot vezérelhető okostelefonnal vagy táblagéppel is. A készlet tartalma:

- 1 darab USB kábelt
- LEGO Technic elemeket: 594 darab
- 1 darab EV3 Építőelem
- 2 darab Nagy interaktív szervomotor
- 1 darab Közepes méretű interaktív szervomotor
- 1 darab Érintésérzékelő
- 1 darab Színérzékelő
- 1 darab Infravörös érzékelő
- 1 darab Infravörös irányjeladó

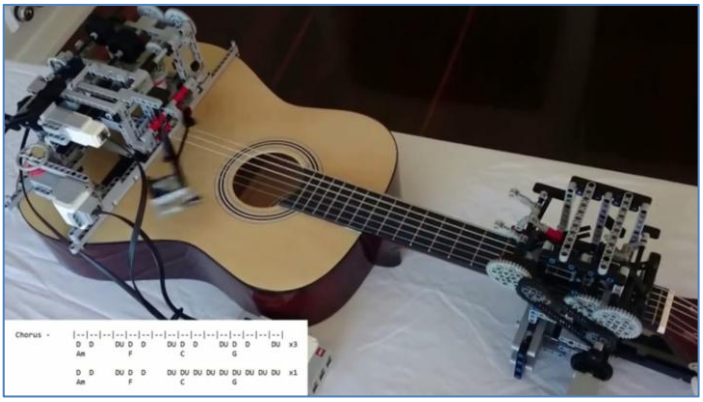


1. ábra: A projekt során használt Lego Mindstorms EV3 (a projekt során többször átépítésre került a feladat vezetőnélküli targoncán való megvalósításának szempontja alapján)
(*forrás: saját készítésű kép*)

A projekt során elég volt az EV3 Építőelemet és a két nagy interaktív szervomotort használni. A fejlesztés kezdetén használt EV3 robot modelljét az 1. ábrán láthatjuk. A következőkben nézzünk meg néhány példát az EV3 alkalmazására [1]:

3.1 A zenélő robotok

Saját programozó környezetében is lehet látványos megoldásokat kitalálni. Például egy EV3 építőelemből és négy nagy szervomotorból létre lehet hozni egy gitározó robotot, amely az előre beprogramozott mozgások alapján pengeti a gitárt (2.ábra) [2].



2. ábra: Lego Mindstorms EV3 gitározó robot
(*forrás: www.youtube.com/watch?v=cXgB3lIvPHI*)

Vagy leüti a zongora billentyűit (3. ábra) [3].

3. ábra: Lego Mindstorms EV3 zongorázó robot
(*forrás: www.youtube.com/watch?v=TlmEpdyKwpI*)

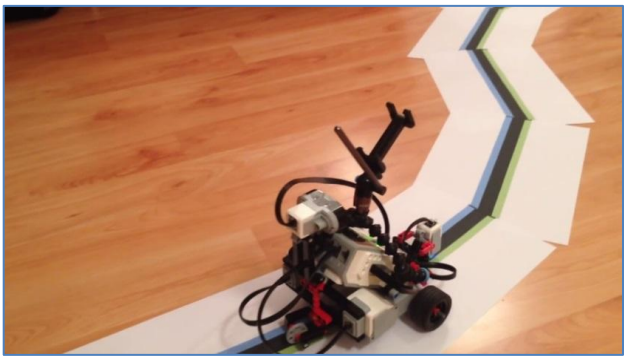
3.2 Lego Mindstorms EV3 ön-egyensúlyozó robot Matlab segítségével

Matlab-ban létrehoztak külön eszközöket, amelyek segítségével kommunikációt létesíthetünk a robottal, illetve felhasználva a Matlab programozó vagy szimulációs felületét, bonyolultabb feladatokat is meg tudunk valósítani. Például a következő ön-egyensúlyozó robotot, amit a 4. ábrán láthatunk [4].

4. ábra: Lego Mindstorms EV3 ön-egyensúlyozó robot
(*forrás: www.youtube.com/watch?v=xz1qZLN8ux0*)

3.3 Lego Mindstorms EV3 vezetőnélküli targonca modell optikai vezetősínnel

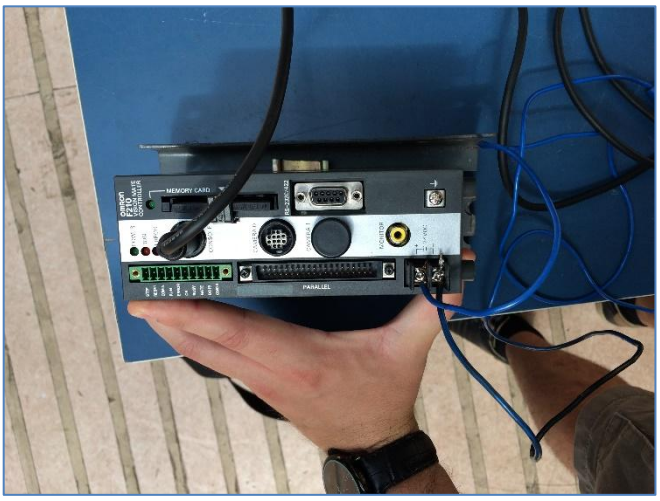
A Lego Mindstorms EV3-ban található színérzékelővel Matlab vagy akár saját programozói környezetében és létrehozható egy program, amely képes végigvezetni a robotot egy vonal mentén, amely működése megegyezik egy optikai vezetősínnel ellátott vezetőnélküli targoncával. Az 5. ábrán láthatunk egy képet a modellről működés közben [5].



5. ábra: Lego Mindstorms EV3 vezetőnélküli targonca modell optikai vezetősínnel (forrás: www.youtube.com/watch?v=I91uoc5fOcQ)

4. Az OMRON F210 ipari képfeldolgozó rendszer

A feladatunk, hogy vizuálisan nyomonkövessük a munkaterületet, az abban közlekedő eszközöket és a célpontot. Különböző jelölésekkel láttuk el az egyes területeket, melyekkel azonosítani és később irányítani tudtuk a vezetőnélküli targoncát (későbbiekben AGV). A folyamat során egy AGV-t kellett eljuttatnunk a munkaterület egy random pontjából a célig, emberi beavatkozás nélkül. A fejlesztés során egy Omron F210-es képfeldolgozó rendszert alkalmaztunk (6. ábra), mellyel a teljes munkaterületet képesek voltunk vizuálisan nyomon követni. Ezt az eszközt gyártósoroknál szokták alkalmazni minőségellenőrzésre, felügyeletre illetve azonosításokra. A fő egység, ami egy kis fekete dobozra hasonlít, tartalmazza a központi adatfeldolgozó egységet, a memóriákat, tápegységet és a különböző csatlakozó felületeket, portokat. Miután biztosítottuk a 24 voltos tápellátást, elkezdhetjük az üzembe helyezést.



6. ábra: Omron F210 központi képfeldolgozó egység
(*forrás: saját készítésű kép*)

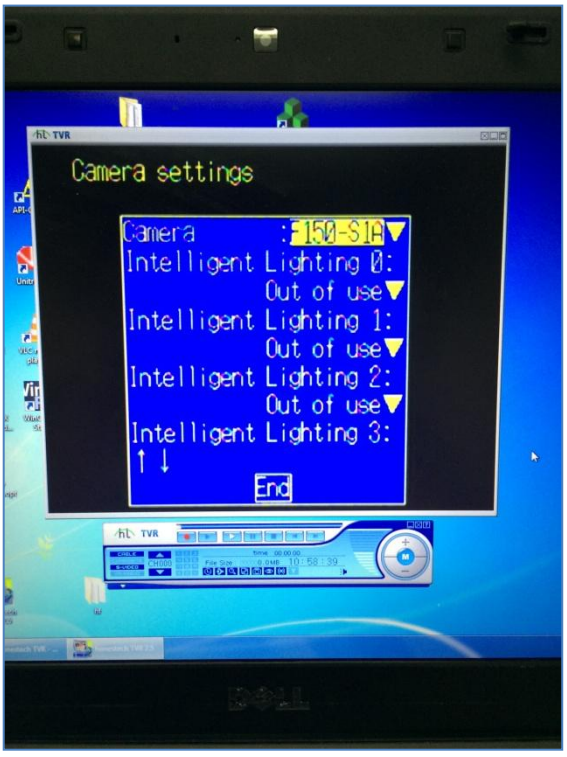
Ahhoz, hogy munkára fogjuk, a gyártó által biztosított, úgynevezett eszközöket (Items) kell telepíteni az F210-re, ami az esetünkben egy SD memória kártyáról történt. Mivel az eszköznek nincs kimondottan tároló egysége, ezért ez egy fontos lépés a beüzemelés során, hiszen tudnunk kell, mely eszközökre lesz szükségünk a feladatunk elvégzéséhez. Ezek a mi esetünkben a következők voltak:

- Camera image (ez alapbeállításként be van kapcsolva)
- Binary defect
- Flexible search
- Calculation
- Normal data

Az eszközök kiválasztását és a későbbiekben előforduló beállításokat egy Omron F150-KP konzol segítségével tudjuk megoldani, melyet az F210 console portjára csatlakoztattunk vezetékesen.

Miután kiválasztottuk az eszközöket, azokat fel kellet telepíteni az F210 belső memóriájába. Ha elkészült a telepítés, újra kell indítani az eszközt és el kell távolítanunk a memória kártyát, hogy ne arról bootoljon, hanem a már kiválasztott items-ek töltődjenek be.

Az első kép a kamera beállításait megjelenítő ablak. Itt kiválaszthatjuk, hogy milyen típusú kamerát alkalmazunk, és ha az támogatja az automatikus fókuszt és egyéb beállításokat, akkor azokat is itt tudjuk beállítani (7. ábra).



7. ábra: Az Omron F210 képfeldolgozó rendszer kamera beállításai
(*forrás: saját készítésű kép*)

Mi egy Omron F150-S1A típusú, fekete-fehér üzemű kamerát használtunk. Az F210 egyszerre két db kamerát képes vezérelni, mi csak egyet alkalmaztunk. Ezt egyébként szintén vezetékes úton csatlakoztattuk a Camera 0 portba. Az F150-S1A, ahogy korábban említettem, fekete-fehér képet tud rögzíteni illetve továbbítani. A projektben 60ms-os válaszidőt állítottunk be, ez egyébként alapbeállítás. Manuálisan történt a fényérzékenység és a fókusz beállítása is.

Miután elértük, hogy a kamera tiszta képet küldjön, elkezdhetjük a telepített items-eket alkalmazni. Fontos, hogy milyen sorrendben választjuk ki ezeket, mivel az F210 sorban elvégzi a műveleteket, méghozzá mindegyiket, amit ezek az items-ek tartalmaznak és ha nem megfelelő a sorrend, akkor vagy értelmetlen feladatot hajt végre, vagy még korábbi értékkel számol vagy egyáltalán nem hajt végre semmit.

A Camera Image itemsben kell beállítani, hogy a kamera által látható területből mekkora részt vizsgáljon. Egy négyszög alapú területet kell kijelölni úgy, hogy először az egyik sarok pontját kell meghatároznunk (a célkeresztet a megfelelő koordinátára irányítjuk), rögzítjük, majd az átellenes sarokpontját kell rögzítenünk.

A Binary defectnél be kell állítanunk a fehér egyensúlyt, hogy jól elkülöníthető legyen a munkaterület és a különböző jelzések, amiket alkalmazunk. Ez történhet manuálisan is, de mi az automatikus beállítást alkalmaztuk, mert a környezet abszolút megfelelt ennek a beállításnak.

A Flexible search gondoskodik arról, hogy az általunk használt, különböző alakú, méretű jelöléseket a rendszerünk felismerje, még akkor is, ha mozognak. Itt is létre kell hoznunk egy bizonyos region-t, egy területet kell kijelölni, hogy a munkaterület mely részén keresse azt a jelet, amit majd megadunk neki. A célt egy körgyűrűvel jelöltük meg és a hozzá tartozó region mérete nem kellett, hogy túl nagy legyen, mivel a cél nem mozgott, fix helyhez volt kötve. Az AGV-re egy fekete teli kört és egy körívet helyeztünk el, hogy forgás után is meg tudjuk határozni, hogy melyik irányba néz az AGV. Kört használtunk, mivel az a mozgások közben ugyan úgy néz ki a rendszer számára, a jelölések forgásával tehát nem kell számolnunk. Az AGV forgását pedig a két kör koordinátáiból számítjuk ki. Az AGV-hez tartozó region megegyezik a munkaterület méretével, hiszen az egész területen engedélyezett a mozgás.

A Calculation itemsben különböző matematikai műveleteket tudunk elvégezni, méghozzá a felismert alakzatok koordinátáira hivatkozva. Képes a rendszer összeadni, kivonni, osztani, szorozni, gyököt vonni vagy négyzetre emelni.

Ha idáig eljutottunk, ahhoz, hogy fel tudjuk dolgozni az adatokat, valahogy el kell küldenünk azokat egy számítógépre. A Normal data segít nekünk ebben. Ebben az itemsben megadhatjuk, hogy a felismert alakzatok adatait (mondjuk koordinátáit) írja-e ki, vagy valamely Calculation eredményét. Egyszerűen csak ki kell választani, hogy mely elemeket szeretnénk kiíratni, kiegészíteni a megfelelő karakterekkel. Azok az adatok, amik itt megjelennek, azokat tudjuk továbbítani a számítógépre. Ezt egy soros porton hajtottuk végre egy RS-232-es port segítségével, 38400bps sebességen 8bit-es adathosszal.

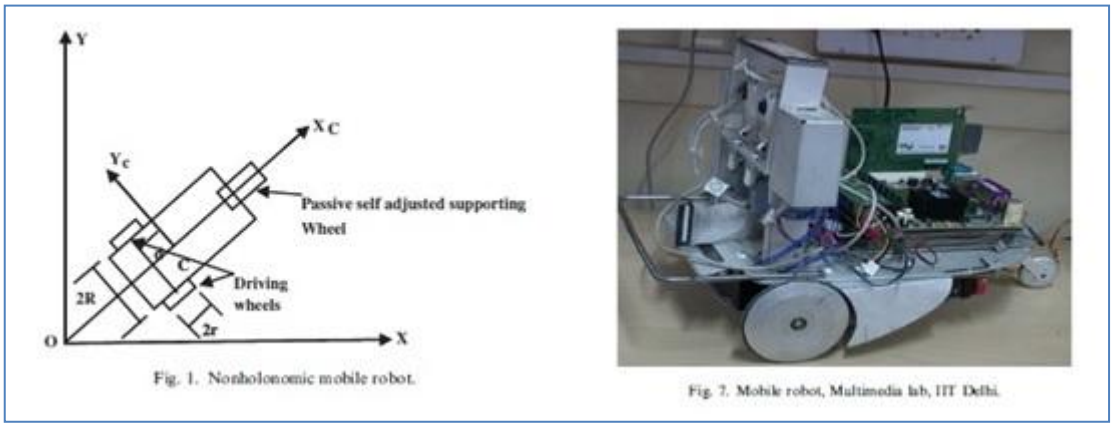
A számítógépen keresztül tudjuk nézni, hogy mit is csinálunk az F210-en. RCA – RCA+USB átalakító segítségével kötöttük össze a két eszközt és a kép megjelenítését a Honest Technology által készített TVR program segítségével oldottuk meg. Az adatokat soros porton keresztül továbbítottuk, amiket a célprogramban dolgoztunk fel [6, 7].

5. Vizuális irányítás témakutatás

A mobil robotok vizuális érzékelők alapján történő irányítása gazdag, szerteágazó szakirodalommal rendelkezik. A következő fejezet azonban csak néhány jellegzetes alkalmazási területet ölel fel, mivel a dolgozatnak nem volt célja a tématerület átfogó áttekintése. A neurális hálózatok könnyebb megértése érdekében egy rövid áttekintés található a 6. fejezet elején.

5.1 Neurális hálózat alapú vezérléssel rendelkező tankhajtású mobil robot irányítása a dinamikai jellemzők figyelembe vételével

Tamoghna Das, I.N. Kar, S. Chaudhury – Simple neuron-based adaptive controller for a nonholonomic mobile robot including actuator dynamics cikkében neurális hálózat alapú vezérléssel rendelkező tankhajtású mobil robot irányítását mutatják be a dinamikai jellemzők figyelembevételével. A robot elméleti modellje és gyakorlati megvalósítása a 8. ábrán látható.



8. ábra: A robot elméleti modellje és gyakorlati példája
(*forrás: Tamoghna Das, I.N. Kar, S. Chaudhury – Simple neuron-based adaptive controller for a nonholonomic mobile robot including actuator dynamics*)

Az irányításhoz szükséges a robot dinamikai modelljének ismerete. A robot helyzetét a rendszer a kerékszög elfordulásokból állítja elő, a hajtások jellemzőit is figyelembe véve. A robot irányításának blokkdiagramja 9. ábrán látható.

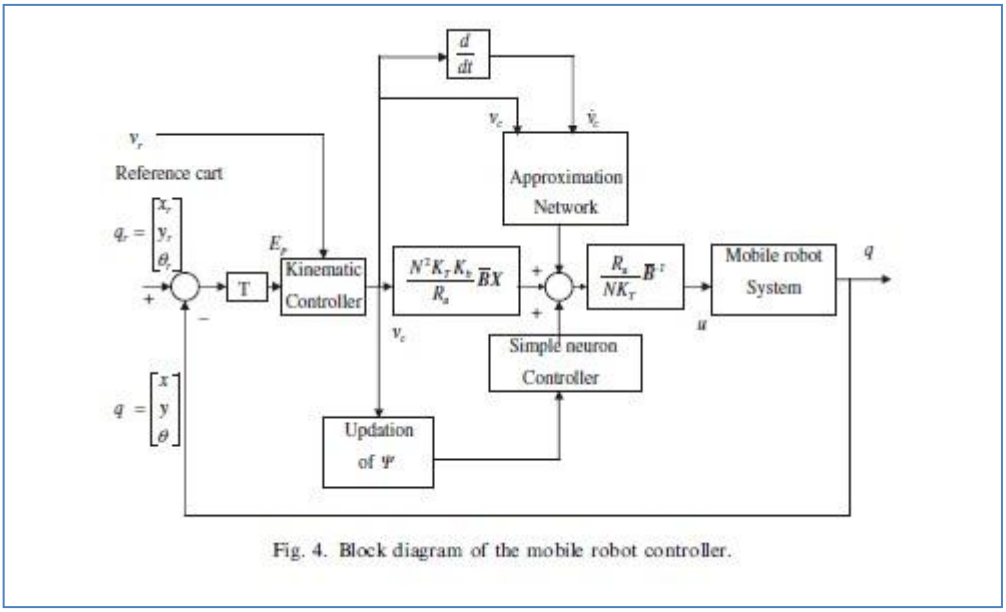


Fig. 4. Block diagram of the mobile robot controller.

9. ábra: A robot irányításának blokkdiagramja
(*forrás: Tamoghna Das, I.N. Kar, S. Chaudhury – Simple neuron-based adaptive controller for a nonholonomic mobile robot including actuator dynamics*)

A robot dinamikai paramétereit a szerzők nem határozták meg közvetlenül, ennek közelítésére szintén egy neurális hálózatot alkalmaztak. A javasolt irányítás robosztus nem csak a strukturális bizonytalanságokra, mint például a tömeg változása, hanem a zavarokra is. A mobil robot valós idejű irányítása a közelítő hálózat online betanulásával elérhető. A rendszer stabilitását és a követési hibák korlátosságát bizonyítottak a Lyapunov stabilitás elmélettel. A számítógépes szimulációkat kör alakra valamint négyzet alakú pályákra mutatják be. A tesztek bebizonyították a kidolgozott eljárás hatékonyságát. A javasolt rendszert nemcsak szoftveresen, hanem a tényleges robottal is sikeresen kipróbálták. Mindkét szimuláció és a kísérleti eredmények részletei 10. ábrán találhatóak [8].

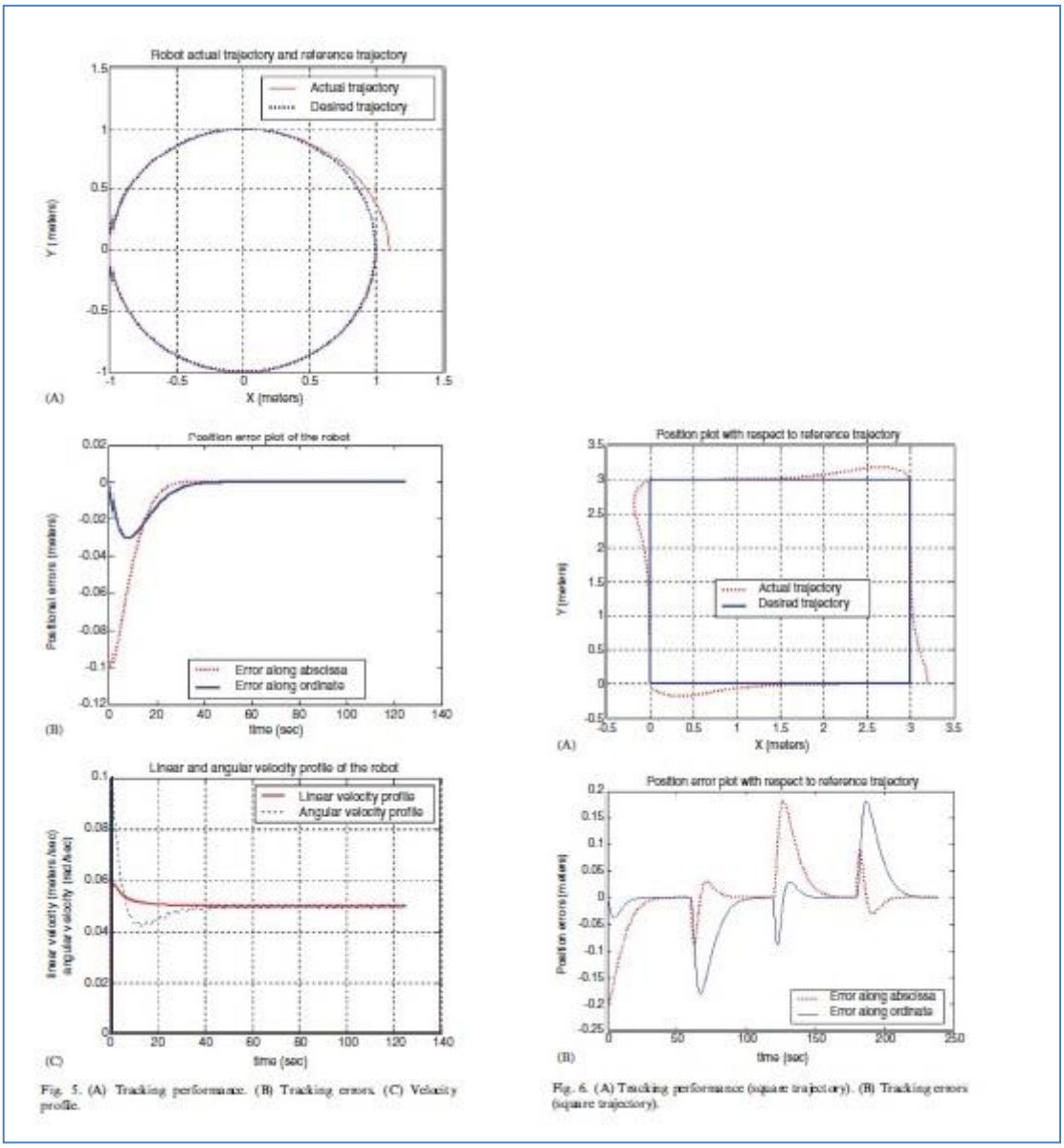


Fig. 5. (A) Tracking performance. (B) Tracking errors. (C) Velocity profile.

Fig. 6. (A) Tracking performance (square trajectory). (B) Tracking errors (square trajectory).

10. ábra: Szimulációk és kísérleti eredmények
(*forrás: Tamoghna Das, I.N. Kar, S. Chaudhury – Simple neuron-based adaptive controller for a nonholonomic mobile robot including actuator dynamics*

5.2 Adaptív dinamikus vezérlő önálló mobil robot nyomkövetéshez

Felipe N. Martins, Wanderley C. Celeste, Ricardo Carelli, Mário Sarchinelli-Filho, Teodiano F. Bastos-Filho – An adaptive dynamic controller for autonomous mobile robot trajectory tracking cikkében olyan rendszert mutatnak be, mely adaptív vezérléssel rendelkezik. Az alkalmazott robot szintén tankhajtással rendelkezik, feladata pályakorrekciós nyomkövetés. A kísérletben használt robotot és az irányítási struktúrát 11. és 12. ábrán láthatjuk.

Fig. 3. The robots used in the experiments.

11. ábra: A projekt során használt robot
(*forrás: Felipe N. Martins, Wanderley C. Celeste, Ricardo Carelli, Mário Sarchinelli-Filho, Teodiano F. Bastos-Filho – An adaptive dynamic controller for autonomous mobile robot trajectory tracking*)

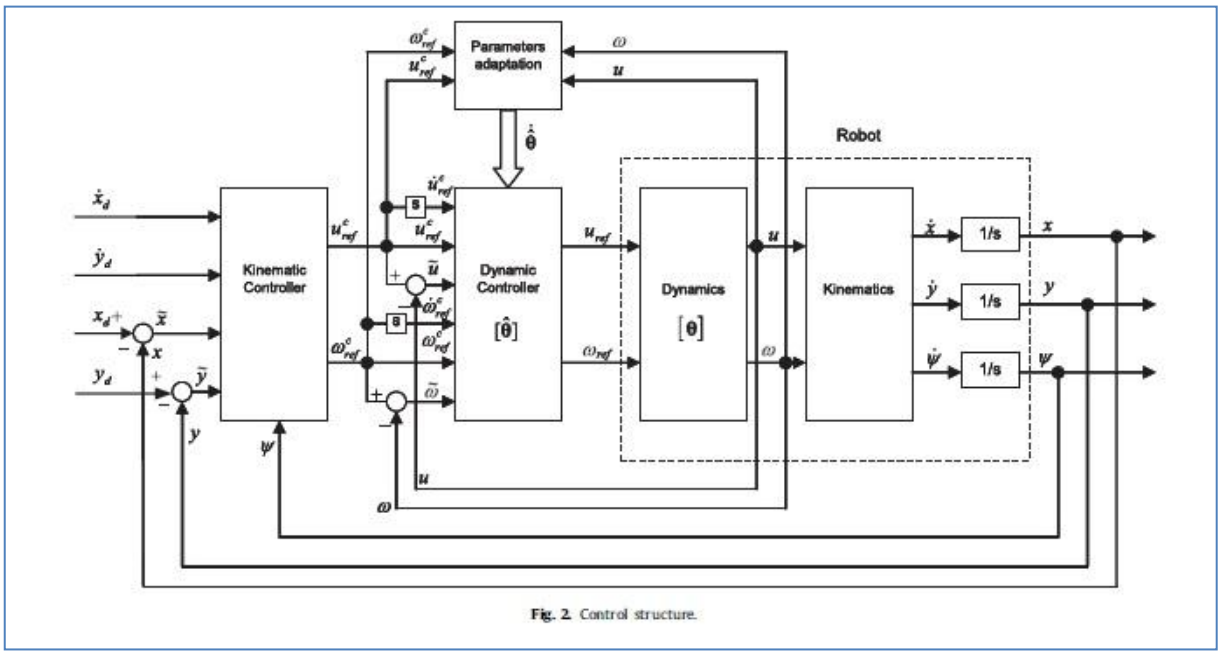
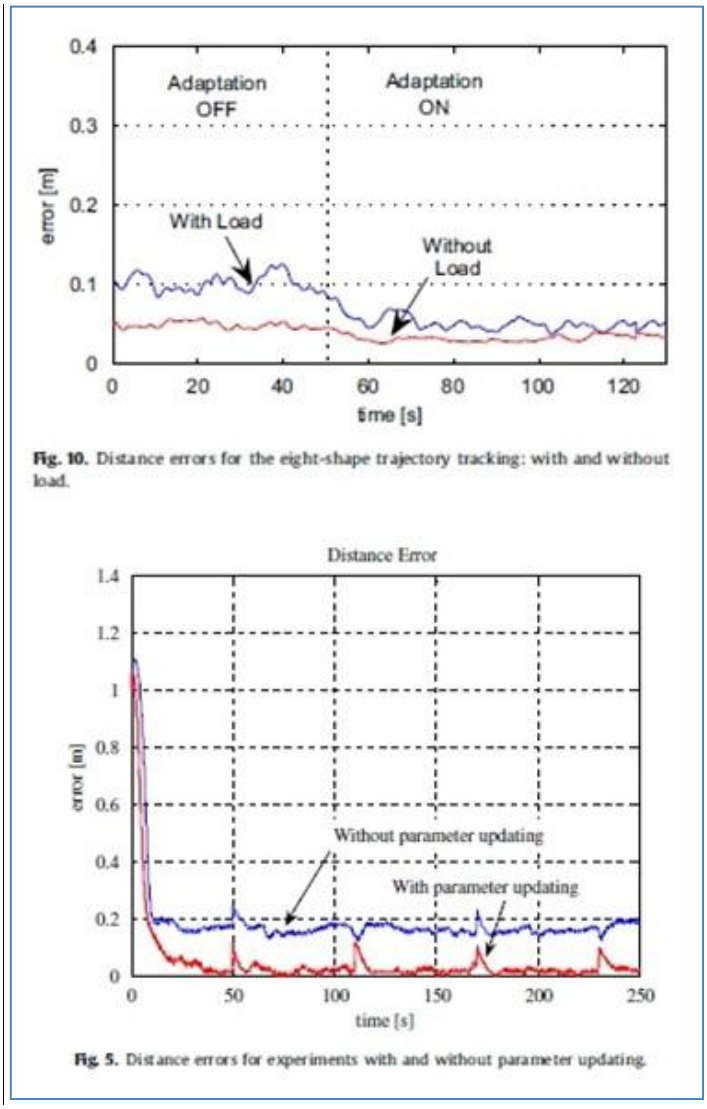


Fig. 2. Control structure.

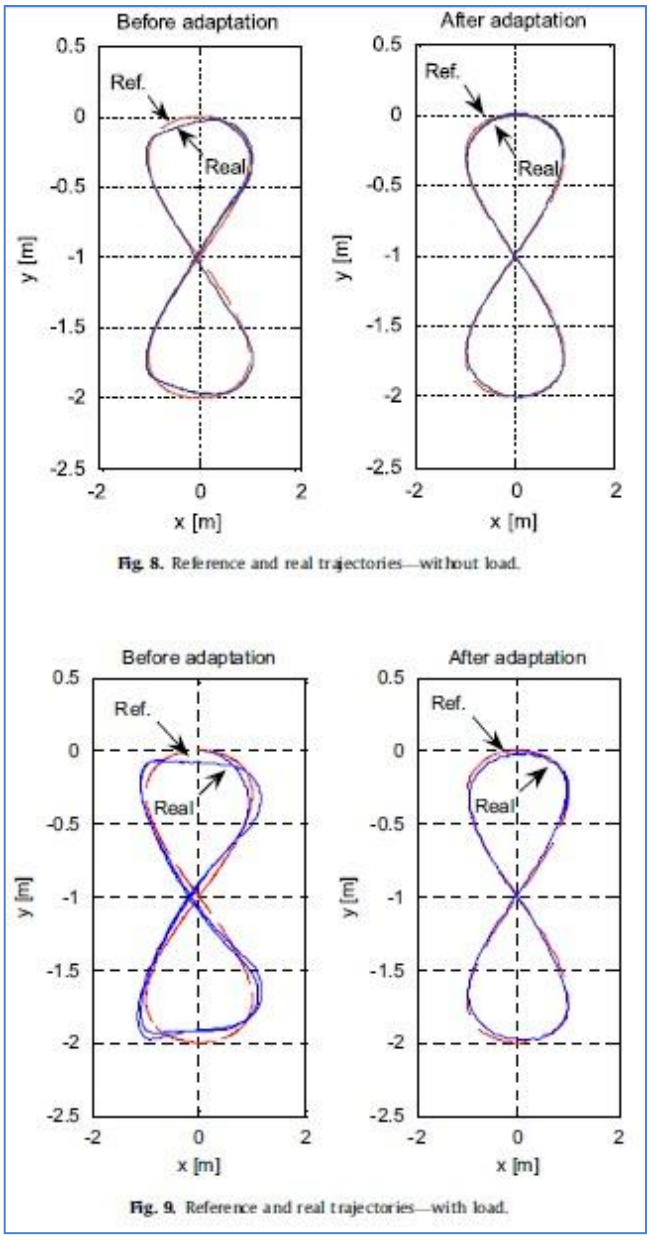
12. ábra: Irányítási struktúra
(*forrás: Felipe N. Martins, Wanderley C. Celeste, Ricardo Carelli, Mário Sarchinelli-Filho, Teodiano F. Bastos-Filho – An adaptive dynamic controller for autonomous mobile robot trajectory tracking*)

Kezdetben, a robot kinematikai modelljét vizsgálták, a lineáris és szögsebességek kívánt értékeit generálva. Következő lépésben az értékek feldolgozása történt, amely adatok segítségével a robot dinamikájának kompenzálása elvégezhető volt, így generált lineáris és szögsebesség parancsokat lehetett küldeni a robot beavatkozó szervének. A robot dinamikájának jellemző paraméterei online frissülnek, így biztosítva a kisebb hibákat és a jobb teljesítményt az alkalmazásoknál, amelyeknél ezek a paraméterek változhatnak. A paraméterfrissítés és frissítés nélküli távolsághibákat a 12. ábra jeleníti meg.



13. ábra: Paraméterfrissítés és frissítés nélküli távolsághibák (terheléssel – terhelés nélkül)
(*forrás: Felipe N. Martins, Wanderley C. Celeste, Ricardo Carelli, Mário Sarchinelli-Filho, Teodiano F. Bastos-Filho – An adaptive dynamic controller for autonomous mobile robot trajectory tracking*)

A teljes rendszer stabilitásának elemzéséhez a Lyapunov elméletet használták, és az irányítás hibák bizonyítottan korlátosak. A szimulációs és kísérleti eredmények is prezentálva vannak, melyek bemutatják a javasolt vezérlő jó teljesítményét pályakorrekciós nyomkövetésnél különböző terhelési viszonyok mellett (14. ábra) [9].



14. ábra: A szimulációs és kísérleti pályakorrekciós nyomkövetésnél különböző terhelési viszonyok mellett
(*forrás: Felipe N. Martins, Wanderley C. Celeste, Ricardo Carelli, Mário Sarchinelli-Filho, Teodiano F. Bastos-Filho – An adaptive dynamic controller for autonomous mobile robot trajectory tracking*)

5.3 Neurális hálózatok robotok irányításához

A neurális hálózatok különböző robotirányítási feladatoknál is hatásos eredményekhez vezetnek. A neurális hálózatok lényege, hogy az irányítórendszert megfigyelésből / mérésből származó adatokkal közvetlenül befolyásolhatjuk, ellentétben a klasszikus vezérlők modell alapú tervezésével. Ez a vezető nélküli targoncák esetére is kedvező megoldást jelent, mivel az eltérő egységrakományok és feladatok miatt ezeknek a gépeknek a felépítése rendkívül változatos. Ebben a kontextusban a neurális hálózatban történő tanulás hatékonyan helyettesíti néhány komplex irányítási folyamat részletes matematikai modellezését, amit a szakirodalomban már kimutattak az elmúlt néhány évben (lásd. pl. e.g. Narendra, 1990; Cembrano et, al., 1992; Hunt, et al., 1993). A robotirányítás az egyik olyan terület, ahol ezen a vonalon kiterjedt kutatás folyik, különösen fokuszálva a nagyobb fokú autonómiával bíró robotok működésének megvalósítására. Ez a fejezet röviden összefoglalja a neurális hálózatok ellenőrzési problémáinak megoldását a korszerű robotikai alkalmazások terén. A neurális hálózatok egyik legfontosabb alkalmazásai az irányításban a komplex nemlineáris rendszerek azonosítása, mert jól képesek közelíteni akár nemlineáris függvényeket is. A robotika területén a leginkább érintett identifikációs problémák a kinematika és a dinamika. A direkt és inverz verziókban a kinematikai problémák kapcsolatban vannak a leképezés, a robot közös koordinátáival, a kapott referencia pont helyzetével és irányával a fizikai térben. Hasonlóképpen a direkt és inverz dinamikai problémák a közös motorparancsok és a kapott referencia pont válasz közötti kapcsolattal foglalkozik. Azt is kimutatták, hogy a neurális hálózatok hatékonyabbak, mint a nemlineáris folyamatok vezérlői önmagukban vagy együtt más típusú vezérlőkkel. Mivel a neurális tanulási modellek adaptív tulajdonsággal rendelkeznek, ezért különösen hasznosak adaptív ellenőrzési problémákra. Az egyik legjobb példa az ilyen problémákra a robotika dinamikus szabályozásában, hogy az online generálódott megfelelő parancsok segítségével a referencia pont a kívánt pályán haladjon. Ez egy komplex nemlineáris szabályozási probléma, ahol ismeretlen és változó paraméterek jelennek meg. Egy másik széles területe a neurális hálózatok irányításban való használatának a szenzomotoros vezérlés, és különösen a képalapú vezérlés. A robotikában a vizuális irányítás használata nagy hangsúlyt kap a kutatásban, főleg a nagyobb teljesítményű számítógépek megjelenését

követően, melyek képesek a megfelelően gyors képfeldolgozásra a szabályzási körben. Ezen a területen a képfeldolgozás klasszikus technikái és a képalapú helymeghatározás módszerei megmaradnak. Ezen módszerek alkalmazását nehezíti, hogy sok esetben a környezet részben ismeretlen, erősen változó és a kamera illetve a robot műszaki lehetőségei is korlátozottak. A neurális hálózatok rugalmassága és tanulási képessége hatékonyan kihasználható a vizuális elhelyezés problémájának megoldására, még egy strukturálatlan, ismeretlen környezetben is [10].

5.4 Ígéretes kutatás a vizuális alapú robot helymeghatározásban neurális hálózat segítségével

A legtöbb vizuális alapú robotpozicionálási módszer a megfigyelt jelenet több geometriai jellemzőinek, a robot helyzete és a vetített kép koordinátái közötti kapcsolatának analitikai kifejezéseire támaszkodik. Ez általában megköveteli, hogy több egyszerű jellemzőt, mint a pontokat, egyeneseket és köröket láthassunk a képen, amelyeknek folyamatosan a látótérben kell lenniük. A funkciónak megfelelő algoritmusok, kamera kalibrációk, a kamera geometria modelljei és az objektumra jellemző kapcsolatok is szükségesek a helyzet meghatározásához. Ezek a lépések gyakran nagy számítást igényelnek, hibára hajlamosak és a kapott kifejezések ellenőrizhető szabadsági foka gyakran korlátozott számú. Gordon Wells, Christophe Venaille, Carme Torras – Promising research Vision-based robot positioning using neural networks cikkében összehasonlító vizsgálatok történnek a vizuális robot helymeghatározására vonatkozóan. Ezen túlmenően bemutat egy új neurális tanuláson alapuló technikát és egy globális kép leírást, mely leküzd sok ilyen korlátozást. Az előrecsatolt neurális hálózat a tanuláshoz a komplex implicit kapcsolatot használja egy 6-dof robot helyzetének elmozdulása és a megfigyelt képi változások globális jellemzői között, mint például (alakteny ezők, kép Fourier transzformáltja). A betanított hálózat ez után használható arra, hogy a robotot a kezdeti pozícióból a kívánt pozícióba mozgassa, kizárólag az érzékelt vizuális információk alapján (15. ábra) .

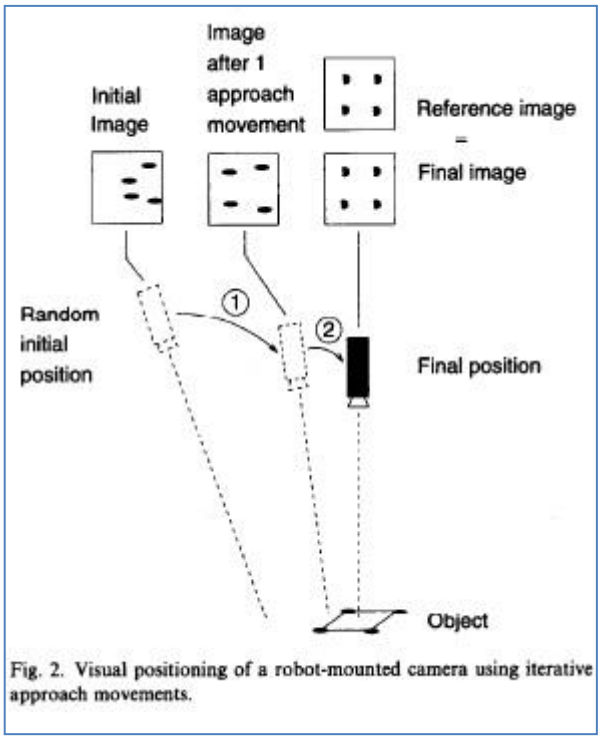
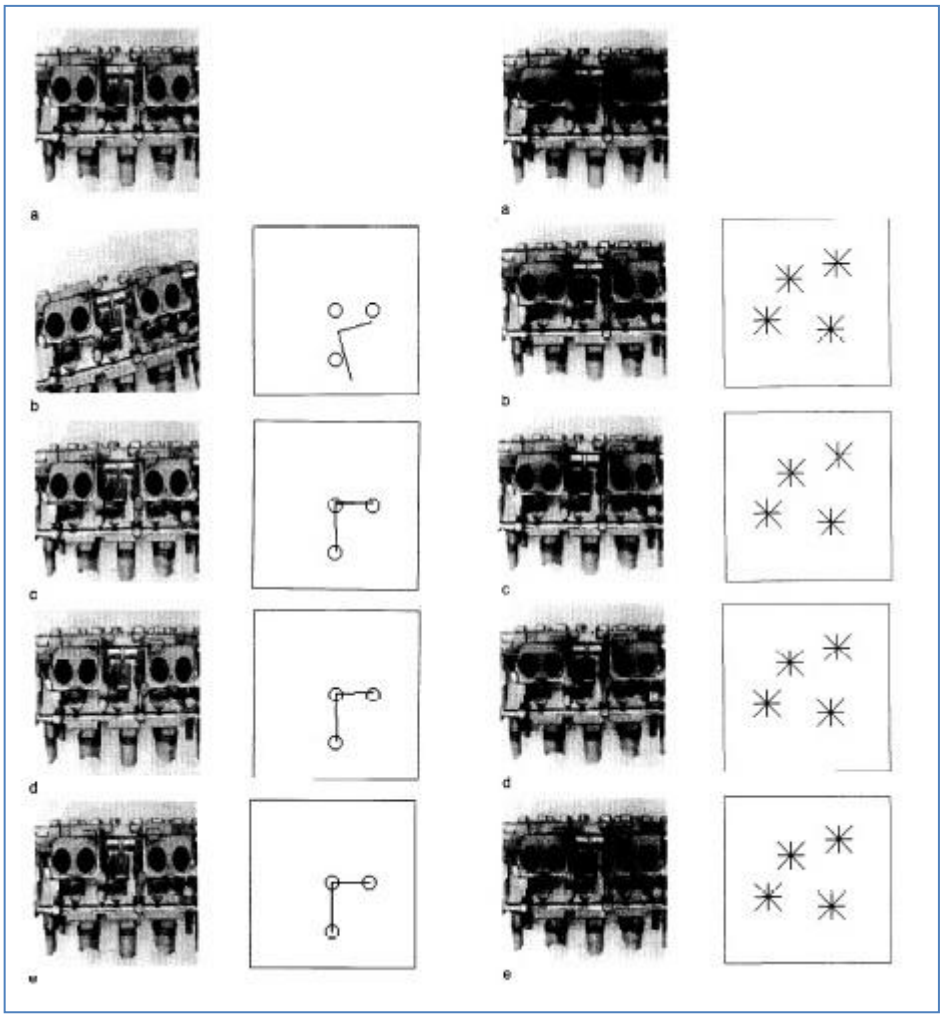


Fig. 2. Visual positioning of a robot-mounted camera using iterative approach movements.

15. ábra: Robotra szerelt kamera vizuális alapú pozicionálása iteratív közelítő mozgásokkal
(forrás: Gordon Wells, Christophe Venaille, Carme Torras – Promising research Vision-based robot positioning using neural networks)

A módszerről kimutatták, hogy képes egy ipari robot pozicionálásában, amely pozicionálás révén lehetővé válik különböző komplex tárgyak elfogadható pontossággal történő ipari ellenőrzésének alkalmazása, és hasznos lehet más egyéb valós idejű feladatoknál, mint például a navigáció, összeszerelés és megfogás. A pozicionálás menetét a 16. ábrán láthatjuk [11].



16. ábra: A robotra szerelt kamera vizuális alapú pozicionálásának iteratív közelítő mozgáslépései
(forrás: Gordon Wells, Christophe Venaille, Carme Torras – Promising research Vision-based robot positioning using neural networks)

5.5 Új becslési algoritmus a robot navigáció segítésére

Claudiu Pozna, Radu-Emil Precup, Földesi Péter – A novel pose estimation algorithm for robotic navigation tanulmánya egy új becslési algoritmust mutat be a robot navigációs problémák keretein belül. Az algoritmus megadja a mobil robot pozícióját a kívánt és jelenlegi pozíciójának különbsége alapján. A robot a 17. ábrán látható.

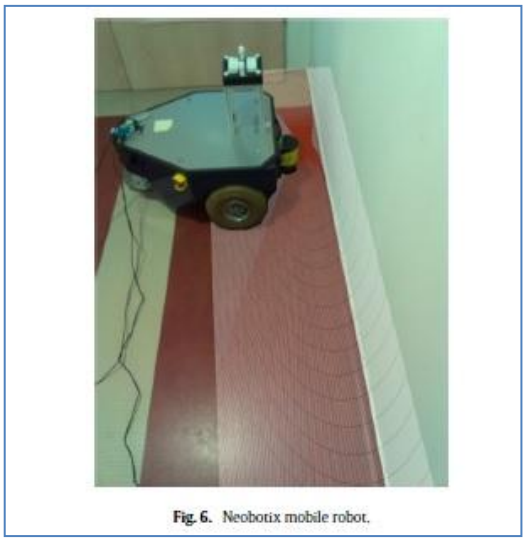


Fig. 6. Neobotix mobile robot.

17. ábra: A projekt során használt robot
(*forrás: Claudiu Pozna, Radu-Emil Precup, Földesi Péter – A novel pose estimation algorithm for robotic navigation*)

Ebben a tekintetben a mobil robot tényleges és virtuális szenzorok által mért értékek alapján dolgozik. Az algoritmus előnye összehasonlítva más becslési algoritmusokkal, hogy klasszikus szűrő megközelítéseken alapul, amelyek rövid számítási időt eredményez. A mobil robot működési elvének folyamatábrája a 18. ábrán látható.

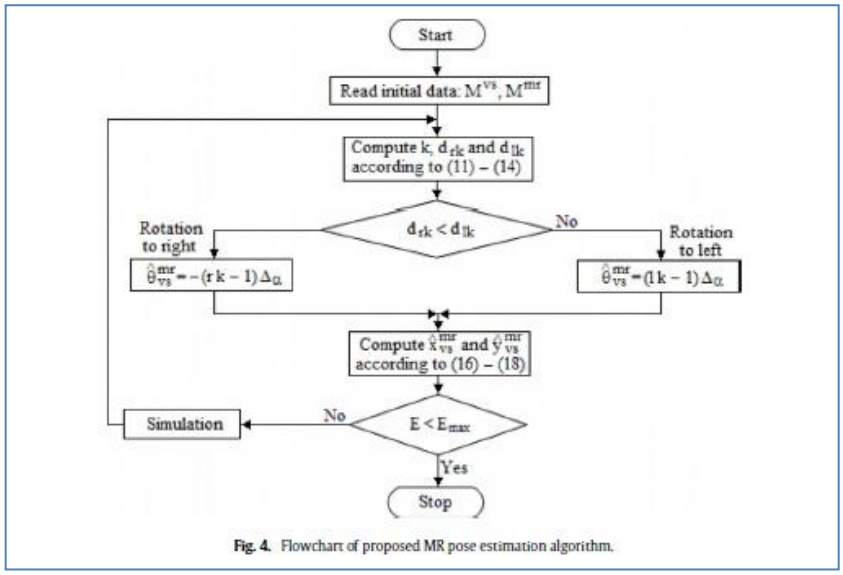


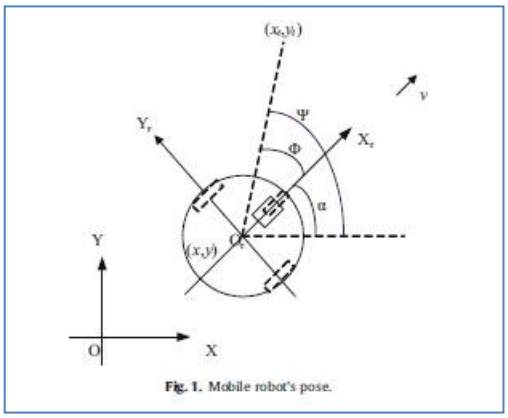
Fig. 4. Flowchart of proposed MR pose estimation algorithm.

18. ábra: A mobil robot működési elvének folyamatábrája
(*forrás: Claudiu Pozna, Radu-Emil Precup, Földesi Péter – A novel pose estimation algorithm for robotic navigation*)

A szimulációs és a valós kísérleti eredmények szerepelnek a tanulmányban, hogy bemutassák az algoritmus hatékonyságát és a benne rejlő lehetőségeket, melyek integrálhatóak a mobil robot irányítási struktúráihoz és algoritmusaihoz [12].

5.6 Mobil robotok moduláris navigációs vezérlése SNN neurális hálózattal

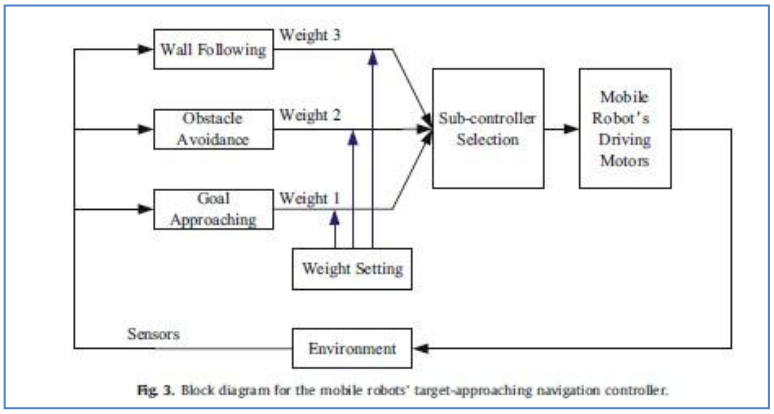
Az autonóm navigáció fontos szerepet játszik a mobil robotoknál. Mesterséges neurális hálózatok sikeresen használhatóak nemlineáris rendszereknél, azonban a modell építése nehéz. Xiuqing Wang, ZengGuang Hou, Feng Lv, Min Tan, Yongji Wang – Mobile robots' modular navigation controller using spiking neural networks című tanulmányban a szerzők a probléma kezelésére egy harmadik generációs neurális hálózatot (spiking neurális hálózatok – SNN) alkalmaznak, mely tartalmazhat olyan funkciókat, amelyek előnyösebb, mint a hagyományos neurális hálózat. Az SNN hálózatok neurális hálózatok képesek idő- és térbeli információk leképezésére, így inkább alkalmasak mobil robotok vezérlésére. Ebben a tanulmányban bemutatásra kerül a moduláris navigációs vezérlés alapján az ígéretes SNN hálózatok mobil robotok alkalmazásában. A mobil robot modelljét a 19. ábrán láthatjuk.



19. ábra: A mobil robot modellje
(*forrás: Xiuqing Wang, ZengGuang Hou, Feng Lv, Min Tan, Yongji Wang – Mobile robots' modular navigation controller using spiking neural networks*)

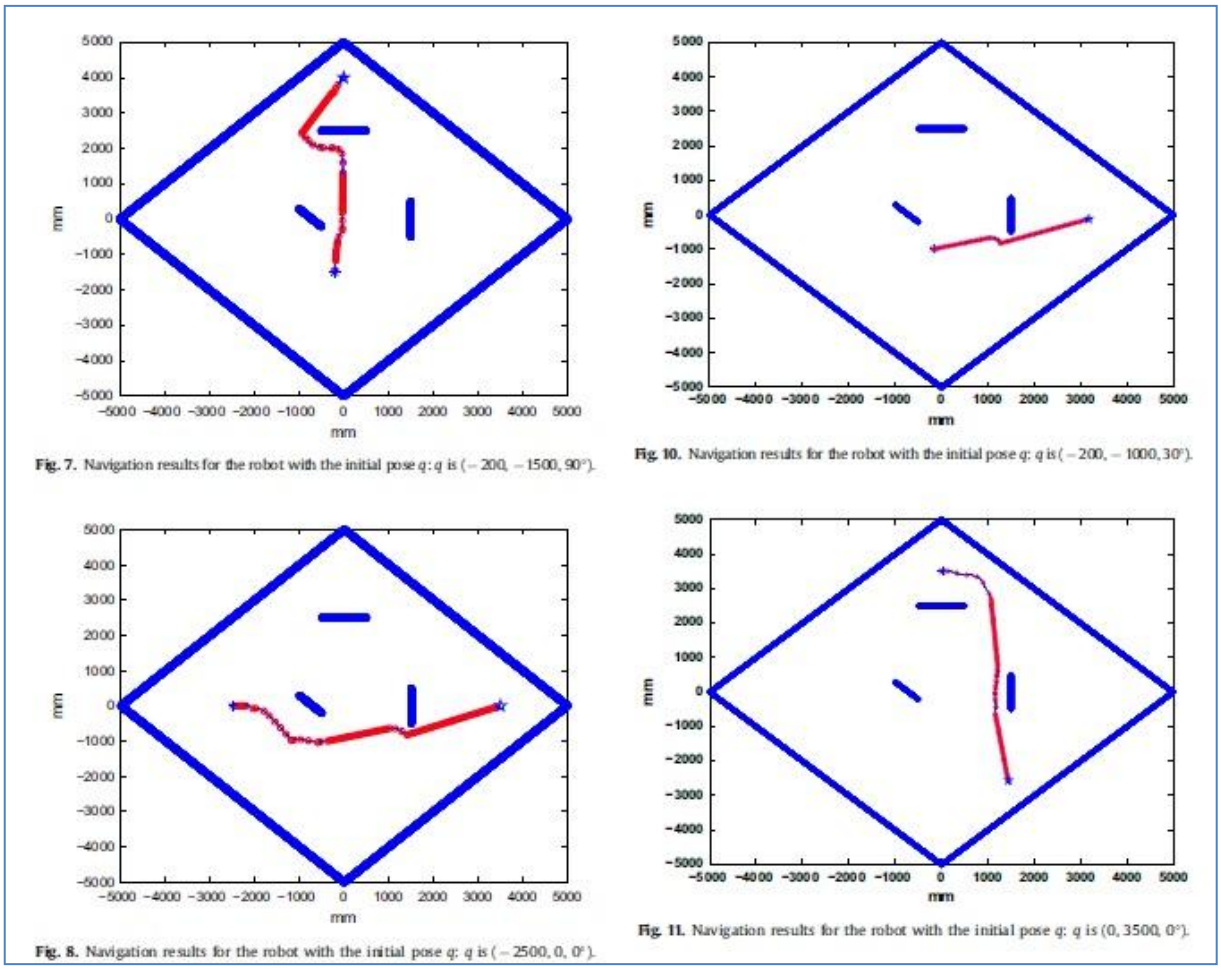
A javasolt működés – cél alapú – közelít a navigációs vezérlőhöz, amely reaktív architektúrát használ, mely három vezérlőből áll: az akadály-elkerülés SNN vezérlőből,

a fal-követő SNN vezérlőből és a cél-közeledés vezérlőből. A mobil robot célközelítő navigációs vezérlésének blokkdiagramját a 20. ábra szemlélteti.



20. ábra: A mobil robot célközelítő navigációs vezérlésének blokkdiagramja (*forrás: Xiuqing Wang, ZengGuang Hou, Feng Lv, Min Tan, Yongji Wang – Mobile robots' modular navigation controller using spiking neural networks*)

A javasolt moduláris navigációs vezérlő nem igényel pontos matematikai modelleket a környezetről, valamint ismeretlen és strukturálatlan környezet esetén is alkalmas a működésre. A szimulációs eredmények a 21. ábrán láthatóak.



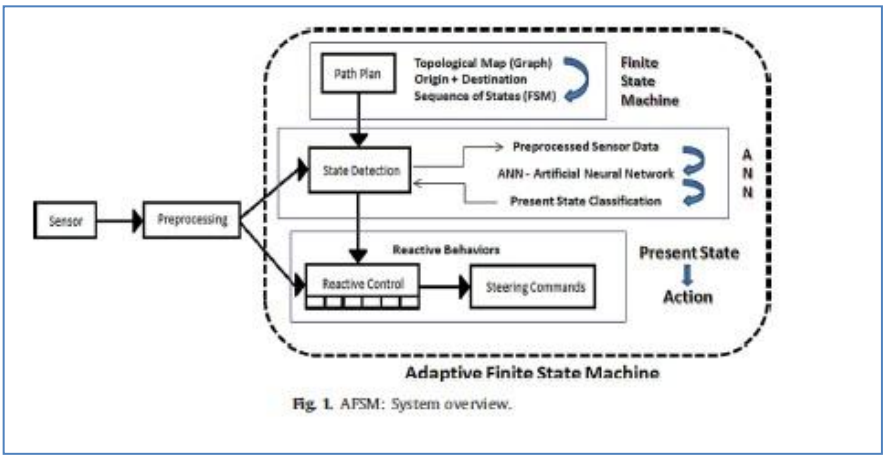
21. ábra: Szimulációs eredmények
(*forrás: Xiuqing Wang, ZengGuang Hou, Feng Lv, Min Tan, Yongji Wang – Mobile robots' modular navigation controller using spiking neural networks*)

A navigációs vezérlő vezérelheti a mobil robotot, hogy sikeresen elérje a célt, miközben szükséges egy falkövető funkció is, mellyel a lokális minimumokba történő beragadást kerüli el [13].

5.7 Adaptív, véges állapotú gép alapú vizuális önálló navigációs rendszer

Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system tanulmányában bemutatnak egy eredeti megközelítést az alkalmazott autonóm mobil

robotok navigációjára, kiegészítve helymeghatározással és a navigáció által használt topológiai térkép egy javasolt AFSM (adaptív véges állapotú gép) technikán alapul. Ebben a megközelítésben a környezet is fel van térképezve, mint egy gráf, és minden lehetséges pálya ábrázolva van az állapotok sorozatával irányított véges állapotú gép által (FSM). Egy mesterséges neurális hálózat képes mintákat felismerni input adatokból, ahol minden minta társul egy adott környezeti funkcióhoz vagy tulajdonsághoz, következésképpen reprezentálja az FSM (véges állapotú gép) összefüggéseit/állapotát. Amikor egy új input minta felismerésre kerül az ANN (megváltoztatja a jelenlegi kontextust) által, lehetővé teszi, hogy az FSM megváltoztassa a következő állapotot és a hozzá kapcsolódó cselekvést/viselkedést. Az input funkciók a környezet saját helyi tulajdonságai (nyert érzékelők adatai), mint például, egyenes út, jobbra és balra fordulás, kereszteződések. Így az FSM integrálva van az előző betanított ANN-be, amely egyfajta kulcselem a felismerésben és a jelenlegi állapot változások jelzésében, így lehetséges, hogy az AFSM (22. ábra) kiválassza az aktuális/helyes cselekvést (helyi reaktív viselkedés) minden helyzetben.



22. ábra: AFSM rendszeráttekintés
(*forrás: Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system*)

Az AFSM lehetővé teszi, hogy a mobil robot önállóan tudja követni az állapotok/viselkedések sorozatát, annak érdekében, hogy elérje a rendeltetési helyét, oly módon, hogy először kiválasztja a helyi reaktív viselkedést minden jelenlegi állapotban, következő lépésben pedig detektálja a változásokat a jelenlegi helyzetben/állapotban, és

követi az állapotok/cselekvések sorozatát, mely kódok megadják a topológiai (globális) pályát az FSM-ben (állapotok/cselekvések sorozata). Az ANN is egy nagyon fontos eleme a rendszernek, mivel lehet tanítani/adaptálni, hogy felismerjen komplex helyzeteket és állapotváltozásokat. Annak érdekében, hogy megbizonyosodjanak a javasolt módszer robosztusságáról különböző helyzetekben és az érzékelők konfigurációjánál, értékelték a megközelítést mind beltéri, mind kültéri környezetben. Egy Pioneer P3-AT robotra felszerelt Kinect érzékelővel beltéri környezetben (23. és 24. ábra), és egy automatizált szabványos RGB kamerával felszerelt járművel városi utak környezetében (25.-27. ábra) [14].

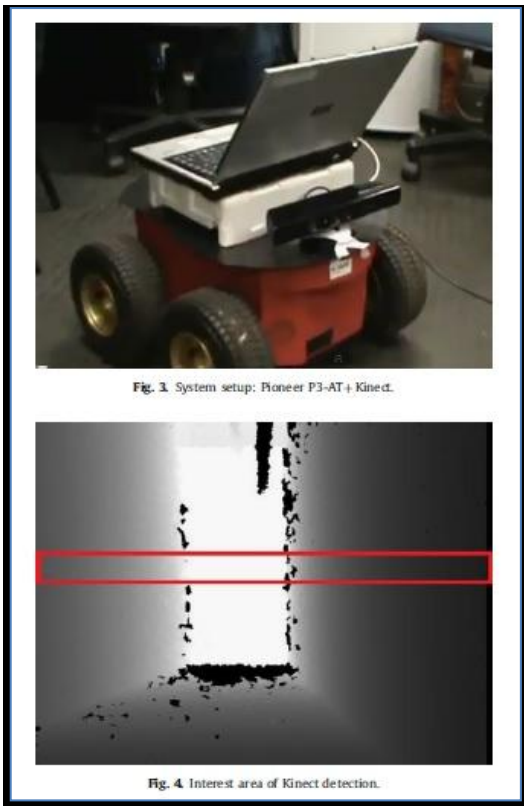


Fig. 3. System setup: Pioneer P3-AT + Kinect.

Fig. 4. Interest area of Kinect detection.

23. ábra: Pioneer P3-AT robot Kinect érzékelővel felszerelve és a Kinect érzékelő vizsgálati tartománya
(*forrás: Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system*)

Fig. 7. Part of the environment map and "left turn and straight" state being correctly detected.

24. ábra: A környezeti térkép része és a ”balra fordulás és egyenesen” helyzet helyes érzékelése
(*forrás: Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system*)

Fig. 8. Camera position.

Fig. 9. Overall area used for collecting training data for the ANN State Detection module.

25. ábra: A kamera pozíciója a járművön és az ANN modul betanításhoz szükséges adatok begyűjtésének területe
(*forrás: Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system*)

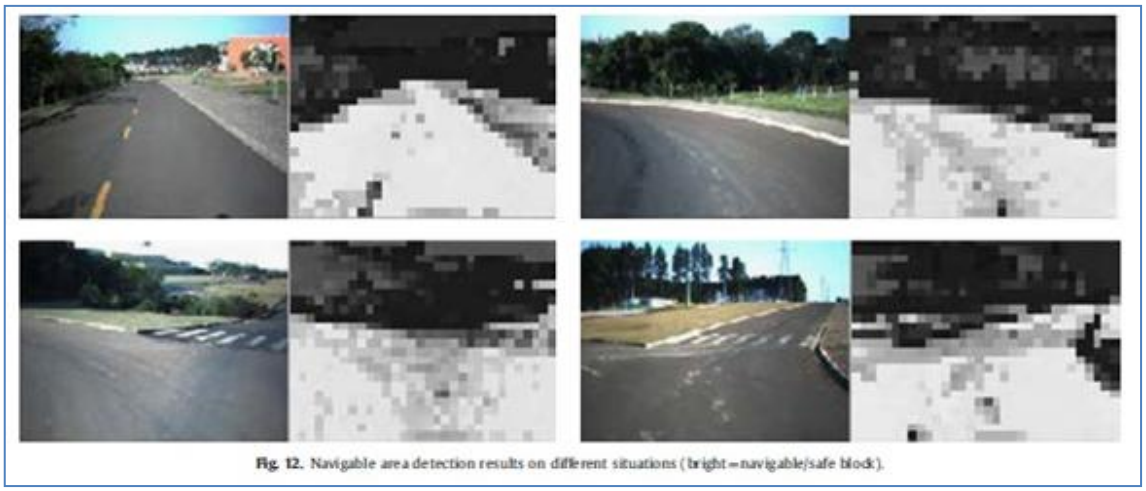
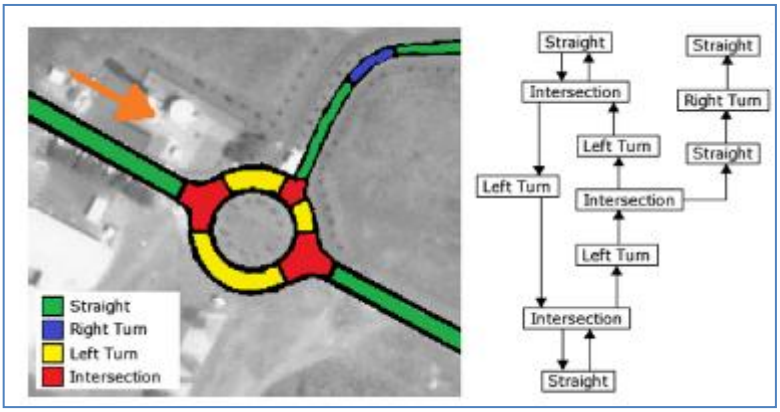


Fig. 12. Navigable area detection results on different situations (bright = navigable/safe block).

26. ábra: Navigációs terület-felismerési eredmények különböző helyzetekben (*forrás: Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system*)



27. ábra: Validációs tesztek és a topológiai ábrázolás (*forrás: Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system*)

A javasolt módszert sikerrel tesztelték különböző helyzetekben, és bizonyították, hogy ígéretes megközelítés az autonóm mobil robotok irányításához és navigációjához.

6. Neurális hálózatok rövid áttekintése és a kísérleti vezetőnélküli targonca modell neurális hálózat alapú navigációja

Ebben a fejezetben bemutatásra kerül a vezetőnélküli targonca modell neurális hálózat alapú navigációjának tervezési lépései és eredményei. A fejlesztés bemutatása előtt egy rövid áttekintés adnék annak érdekében, hogy könnyen megérthető legyen, hogy mi is az a neurális hálózat, mikor alkalmazható és milyen előnyei vannak.

6.1 Neurális hálózatok

A neurális hálózatok az információ feldolgozás egy „új” paradigmája, amely egy biológiai inspirációjú információ feldolgozás, ahol modellként az idegrendszer struktúráját és működését vesszük alapul. A tudományterület még a kezdeti stádiumban van, mégis számos alkalmazási területen az egyszerűsített modellekkel is jobb eredményeket lehet elérni, mint a „hagyományos” algoritmikus megoldásokkal. Mikor célszerű neurális hálózatokat használni:

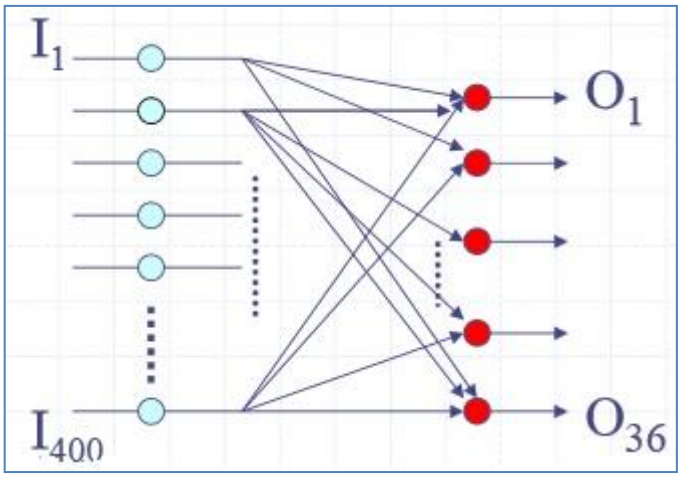
- Ha a megoldandó problémával kapcsolatban gazdag adathalmaz áll rendelkezésre
- Ha a megoldáshoz szükséges szabályok nem ismertek
- Ha a rendelkezésre álló adathalmaz nem teljes vagy hibás adatokat is tartalmazhat
- Ha sok összefüggő bemenő adat vagy kimeneti paraméter áll rendelkezésre

A neurális hálózatok nagyon egyszerű processzorokból, úgynevezett neuronokból épülnek fel. A processzorok változtatható súlytényezőjű összeköttetések hálózatán át kommunikálnak egymással. A neurális hálózatok esetében nem programozásról, hanem tanításról beszélünk. A tárolt információk a hálózatban elosztottan, a súlytényezők közvetítésével ábrázolódnak. A neurális hálózatok hibatűrőek, az elosztott párhuzamos tudásreprezentáció miatt a súlytényezők egy részének jelentős megváltozása sem befolyásolja alapvetően a hálózat működését. A hálózat működését három fő tényező

határozza meg: a processzorok átviteli függvénye, a hálózat összeköttetési sémája és a tanítási módszer. A neurális hálózatok alkalmazásának menete:

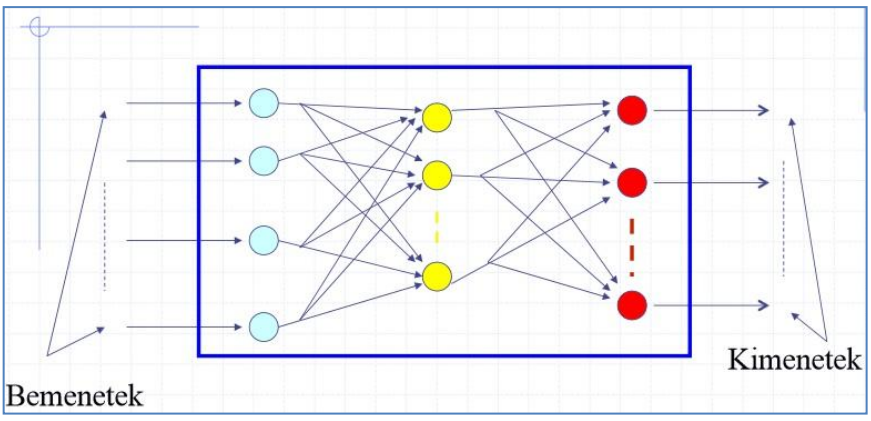
- A tanító adatok összeállítása
- Feladatspecifikus neurális hálózat (paradigma) kiválasztása
- A hálózat jellemzőinek (a processzorok átviteli függvényének, a processzorok számának, a tanítási módszereknek és paramétereknek, valamint a kezdeti súlymátrix értékeinek) kiválasztása
- Teljesítmény mérő módszer kiválasztása
- Tanítás és tesztelés, amíg a hálózat a kívánt viselkedést nem mutatja

Az első mesterséges neurális hálózat (Perceptron) ismertetésével könnyen megérthetőek a rendszer működési alapjai. Frank Rosenblatt (1957) karakterfelismerést valósított meg oly módon, hogy épített egy mesterséges idegrendszert, melyben mesterségek idegek (neuronok) voltak, és ezeket összekötötte egymással (mint ahogyan az idegsejtek is kapcsolódnak egymáshoz). A karakterfelismerő működése így hasonló volt az emberi idegrendszer működésével. Létrehozott egy ernyőt, amin 20x20-as pixelek voltak, erre az angol abc nyomtatott betűit és a számokat vetítette fel (összesen harminchat darab). Ezek alapján létrehozott egy hálózatot, amelynek négyszáz bemenő neuronja (20x20) és harminchat darab kimenő neuronja volt, majd ezeket összekötötte. Ezt követően rávetített egy képet az ernyőre és megnézte, hogy a kivetített kép megegyezik-e a rendszer kimenetével. Ha nem egyezett, akkor megváltoztatta az összeköttetések erősségeit és ezt a folyamatot ismételgette mindaddig, amíg a rendszer adott hibahatár mellett, de megfelelően működött. A megvalósított rendszer egy előrecsatolt egyrétegű hálózat volt, melynek modelljét a 28. ábrán láthatjuk.



28. ábra: A megvalósított előrecsatolt egyrétegű hálózat működési modellje (*forrás: Dr. Kutor László – Intelligens Rendszerek Elmélete 5. előadásanyag (Óbudai Egyetem NIK, 2014. tavasz*)

Az egyrétegű hálózatok csak olyan problémák megoldására alkalmas, amely esetén a kimenetek jól elkülöníthető minták voltak, nem volt köztük átfedés. Ezért az 50-es években ezt a kutatási irányt zsákutcának bélyegezték bonyolult feladatok alkalmazásában, és nem fordítottak a témára kutatási erőforrást egészen a 90-es évekig, amikor rájöttek, hogy bonyolult feladatok is - melyekben a kimeneteknél átfedések jöhetnek szóba - megoldhatóak neurális hálózatokkal oly módon, hogy közbenső rétegeket hozunk létre a hálózatban (többrétegű hálózatok). Ezen közbenső neuronok tanítása a kezdetekben okozott problémát, de megfelelő matematikai bizonyítások segítségével később elérhető vált. A többrétegű neurális hálózat modelljét a 29. ábrán láthatjuk.



29. ábra: Többrétegű neurális hálózat (*forrás: Dr. Kutor László – Intelligens Rendszerek Elmélete 5. előadásanyag (Óbudai Egyetem NIK, 2014. tavasz*)

A kísérleti vezetőnélküli targonca modell betanítása során két egyrétegű és egy kétrétegű neurális hálózatot is használtunk, melyek részleteit és eredményeinek összevetését a következő fejezetben mutatom be [15, 16].

6.2 Kísérleti vezetőnélküli targonca modell neurális hálózat alapú navigációja

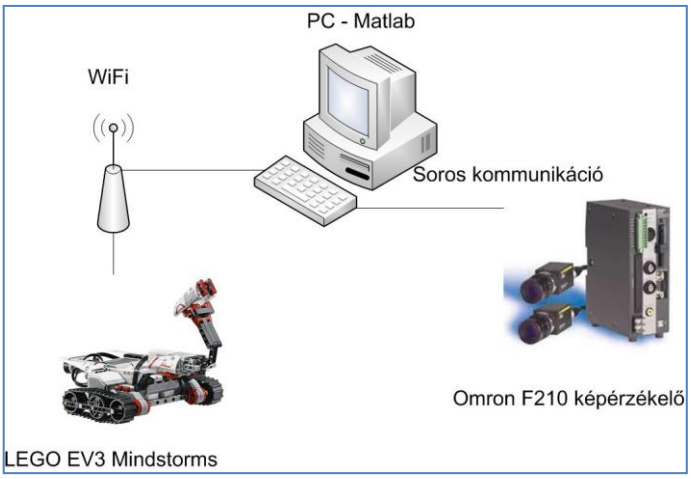
A feladat az volt, hogy egy OMRON F210 képfeldolgozó rendszerrel azonosítsuk a Lego Mindstorms EV3 robotot és a célterületet, amelyeket markerekkel jelöltünk meg, majd a koordinátákat elküldjük egy számítógépnek. Ezt követően a kapott adatokat feldolgozzuk, és valamilyen vezérléssel eljuttatjuk a robotot a célterületre. A lényeg az, hogy függetlenül attól, hogy hol helyezzük el a robotot és a célterületet a kamera látóterében, a robot mindig eljusson oda.

Első körben a Mindstorms saját programozó környezetében próbáltam megoldani a problémát. Az alapötlet az volt, hogy a kamerából soros porton kapott adatokat C#-ban fogadjuk és feldolgozzuk, majd a kapott adatokat, amik sebességek és idők lettek volna (melyik motort mekkora sebességgel meddig hajtsuk) kiírattuk volna egy text fájlba, majd ezt a fájlt olvasta volna be a Mindstorms programozó környezete. Sajnos azzal a problémával szembesültem, hogy a text fájlt mindig rá kellett tölteni a programozható intelligens építőelemre, ami problémás lett volna, hisz ahogy mozog a robot, úgy változnak a koordináták és így a kiírandó adatok is, tehát folyamatos rátöltés lenne szükséges. Ezután megpróbáltam az intelligens építőelem belső memóriájába másolni a c# program segítségével a text fájt, de nem ismerve az intelligens építőelem pontos működését inkább egy új és egyszerűbb megoldás után néztem.

Az első megoldás sikertelensége után egy másik lehetőség után kellett nézni. Matlab-ban már létezik egy kiegészítő csomag, aminek a segítségével kommunikálni lehet a LEGO Mindstorms EV3 robottal. Illetve fontos szempont volt az is, hogy az F210 képfeldolgozó rendszerből soros porton érkezett adatokat könnyen lehessen kezelni, ami Matlab-ban lehetséges. Először is a soros portos beállításokat végeztük el a Matlab-ban, amely beállítások a mellékletben találhatóak (*Soros port megnyitása, érték tárolása* melléklet). A robotra egy kinyomtatott fekete kört és egy körívet ragasztottunk, így a két pont segítségével szögeltérést is tudtunk vizsgálni, illetve a célterületet egy, a roboton lévőnél nagyobb méretű körgyűrűvel jelöltük meg. Soros porton a kör és körív

koordináta került kiküldésre, a célterület koordinátája fix koordináta, ezt kézzel adtuk meg a Matlab-ban. A koordinátákat az F210 képérzékelő segítségével kaptuk meg, illetve küldtük ki. A kapott üzenetből egy egyszerű Matlab program kigyűjtötte a két-két koordinátát. A program forráskódja a mellékletekben található (*Bejövő adatok feldolgozása* melléklet). A teszt alatt látható vált, hogy a program megfelelően kiválasztja a szükséges adatokat és meghatározza a koordinátákat. A robot helyváltoztatásával a koordináták is változnak, mely változásokat érzékeli a kamera, így a Matlab is újra és újra kiszámolja a koordinátákat, ahányszor csak új adat érkezik be. De figyelni kell, hogy egy adott helyzetben csak egyszer küldjük el az üzenetet az F210 képérzékelővel, mert különben küldés számtól függően többször dolgozza fel ugyanazt azt adatot a Matlab.

A következő lépés a kommunikáció a Mindstorms EV3-mal. Először is WiFi-n keresztül szerettünk volna kapcsolódni a robothoz, ehhez szükség volt egy routerre. A router beállításait a következő YouTube videó segítségével tudtam elvégezni: www.youtube.com/watch?v=dm428Q3GL6A [18]. A beállítások után még szükség volt egy NETGEAR N150 Wireless Adapter (WNA1100) típusú hardverzárra, amely nem volt tartozéka a LEGO EV3 Mindstormsnak. Ezt követően már rá tudtuk csatlakoztatni a router-ra a robotot és megkezdődhetett a kommunikáció kiépítése Matlab környezetben. Szerencsére a Matlab az EV3 kiegészítő csomagjához kiválóan használható súgót biztosított példákkal, így egyszerűen létre lehetett hozni a kommunikációt néhány adat kikeresése után. A forráskód szintén a mellékletben található (*Robot kommunikáció megnyitása* melléklet). Az eszközök közötti kommunikációt a 30. ábra szemlélteti.



30. ábra: Eszközök közötti kommunikáció
(*forrás: Saját készítésű Visio ábra*)

Már lehetséges volt parancsok alapján motorokat vezérelni a roboton, és fogadtuk a koordinátákat, így eljött az ideje annak a Matlab programnak a megalkotása, amely a bejövő adatok alapján távolságot és szögeltérést számol (*Helyzeteltérés meghatározása* melléklet). Miután a program elkészült, elkészítettem három egyszerű vezérlőprogramot, amely ”b” megnyomására körülbelül másfél fokkal balra, ”j” megnyomásával jobbra forgatta, illetve ”e” megnyomásával 1-2 centimétert előre vezette a robotot. Ezután szükség volt egy mátrixra, amely összesítette az egyes lépések után, hogy mekkora volt a távolság a célterülettől, a szögeltérés és hogy melyik parancsot nyomtuk meg (b, j vagy e). Tehát kézzel elvezettük a robotot lépésenként a célterületre és minden egyes lépés után elmentettük a célterülettől való távolság és a szögelfordulás értékét. Ezen értékek alapján megszületett a mátrix, mely adatainak segítségével történt a robot betanítása. A mátrixot az 1. táblázat mutatja be.

1. táblázat: A robot mozgásának adatait tartalmazó eredménymátrix
A módválasztás hatása a tervezésre
(forrás: a szerző saját munkája)

Sorszám	Szögeltérés a kamera x tengelyéhez képest [fok]	Távolság a célterülettől [egység]	Megadott parancs	Megadott parancs kódja		
0	0	437.1041	-	0	0	0
1	9.2110	429.5326	b	1	0	0
2	12.5288	425.5925	b	1	0	0
3	16.9908	421.8048	b	1	0	0
4	21.8014	417.6389	b	1	0	0
5	27.9795	413.5048	b	1	0	0
6	34.1145	409.1021	b	1	0	0
7	32.8285	378.6070	e	0	1	0
8	37.1847	371.2226	b	1	0	0
9	43.9191	366.0413	b	1	0	0
10	51.5819	360.5586	b	1	0	0
11	55.0080	354.4633	b	1	0	0
12	62.0205	349.4642	b	1	0	0
13	60.6422	323.0557	e	0	1	0
14	60.6422	292.9112	e	0	1	0
15	62.0205	263.2703	e	0	1	0
16	62.0205	241.0856	e	0	1	0
17	61.2602	217.0541	e	0	1	0
18	62.7004	196.3422	e	0	1	0
19	62.7004	178.5532	e	0	1	0
20	56.7683	176.2739	j	0	0	1
21	53.1301	172.7346	j	0	0	1
22	48.5035	170.5264	j	0	0	1
23	41.4965	168.0067	j	0	0	1
24	76.6075	185.6906	j	0	0	1
25	33.2317	161.8410	j	0	0	1
26	27.2996	158.2127	j	0	0	1
27	19.9831	154.6714	j	0	0	1
28	18.4349	152.0280	j	0	0	1
29	14.8265	149.0277	j	0	0	1
30	9.7276	145.0353	j	0	0	1
31	4.7636	141.1179	j	0	0	1
32	0	136.3094	j	0	0	1
33	0	0	e	0	1	0

A robot betanítására három különböző neurális hálózatot is használtunk, majd az eredményeket összevetettük. Az első esetben egy rejtett rétegű neurális hálózatot használtunk, ahol két bemenet és egy kimenet van. A két bemenet és a kimenet a következő:

- I1 = távolságértékek / 500 (500-nál nem volt nagyobb távolság)
- I2 = szögértékek / 90 (szögértékek maximuma 90 fok lehet)
- O1 = 0 ha [1 0 0] volt a parancskód (jobbra ment)
- O1 = 0,5 ha [0 1 0] volt a parancskód (egyenesen ment)
- O1 = 1 ha [0 0 1] volt a parancskód (balra ment)

A Matlab-ba beírt és lefuttatott program forráskódja a mellékletben található (*Matlab neurális hálózat egy rejtett rétegű, egy kimenet* melléklet). A program lefutása után az eredményeket a 2. táblázat összegzi.

2. táblázat: Egy rejtett rétegű, egy kimenetű neurális hálózat eredményei
(forrás: a szerző saját munkája)

Sorszám	Kimenet		Neurális hálózat eredményei	Valós eredmények
1	0.0034	0	b	b
2	-0.0041	0	b	b
3	0.0117	0	b	b
4	0.1487	0	b	b
5	0.2736	0	b	b
6	-0.0115	0	b	b
7	0.4254	0,5	e	e
8	0.0973	0	b	b
9	-0.0300	0	b	b
10	-0.0472	0	b	b
11	0.0126	0	b	b
12	0.2984	0	b	b
13	0.5028	0,5	e	e
14	0.6099	0,5	e	e
15	0.5019	0,5	e	e
16	0.5091	0,5	e	e
17	0.5883	0,5	e	e
18	0.6015	0,5	e	e
19	0.6330	0,5	e	e
20	0.9481	1	j	j
21	1.0253	1	j	j
22	1.0306	1	j	j
23	0.9896	1	j	j
24	0.2540	0	b	j
25	0.9954	1	j	j
26	1.0154	1	j	j
27	0.9805	1	j	j
28	0.9633	1	j	j
29	0.9516	1	j	j
30	0.9782	1	j	j
31	0.9867	1	j	j

Látható, hogy az alkalmazott betanítás egyetlen hiba kivételével leköveti a tényleges mozgást.

A következő esetben nézzük meg, hogy milyen eredményre jutunk, ha három kimenetű neurális hálózatot használunk. A bemenetek ebben az esetben is megegyeznek az előbbi példával, viszont a kimenetek mások lesznek. A kimenetek hasonlóak a robot mozgásánál definiált mozgási parancsokkal (előre, balra, jobbra) kódjával (b - 1 0 0, e – 0 1 0, j – 0 0 1). A Matlab-ba beírt és lefuttatott program forráskódja a mellékletben található (*Matlab neurális hálózat egy rejtett rétegű, három kimenet* melléklet). A feldolgozott adatok eredményeit és az összevetést az eredeti értékkel a 3. táblázat tartalmazza.

3. táblázat: Egy rejtett rétegű, három kimenetű neurális hálózat eredményei
(forrás: a szerző saját munkája)

Sorszám	1. Kimenet	2. Kimenet	3. Kimenet				Neurális hálózat eredményei	Valós eredmények
1	0.8307	0.2042	-0.0261	1	0	0	b	b
2	0.8786	0.1799	-0.0220	1	0	0	b	b
3	0.9096	0.1540	-0.0467	1	0	0	b	b
4	0.9143	0.0898	-0.0274	1	0	0	b	b
5	0.8994	0.0268	0.0510	1	0	0	b	b
6	0.9798	0.0342	-0.0109	1	0	0	b	b
7	0.8639	0.1349	-0.0249	1	0	0	b	e
8	0.8827	0.1870	-0.0494	1	0	0	b	b
9	0.9669	0.2738	-0.1807	1	0	0	b	b
10	0.8296	0.2656	-0.0791	1	0	0	b	b
11	0.6578	0.3553	-0.0091	1	0	0	b	b
12	0.6773	0.3836	0.0017	1	0	0	b	b
13	0.0816	0.8851	0.1246	0	1	0	e	e
14	-0.0207	1.0198	0.0503	0	1	0	e	e
15	0.0386	0.8754	-0.0814	0	1	0	e	e
16	0.0714	0.9406	-0.2022	0	1	0	e	e
17	0.0751	1.0819	-0.1050	0	1	0	e	e
18	0.1166	1.0594	0.0078	0	1	0	e	e
19	0.0462	1.0107	-0.0035	0	1	0	e	e
20	0.0560	0.6576	0.3345	0	1	0	e	j
21	0.0370	0.2687	0.6877	0	0	1	j	j
22	0.0092	-0.0229	0.9953	0	0	1	j	j
23	0.0134	-0.0070	1.0013	0	0	1	j	j
24	0.0327	0.1541	0.8477	0	0	1	j	j
25	-0.0258	-0.0097	1.0001	0	0	1	j	j
26	-0.0563	-0.1057	1.0502	0	0	1	j	j
27	-0.0135	-0.0371	0.9868	0	0	1	j	j
28	-0.0118	-0.0292	0.9932	0	0	1	j	j
29	-0.0009	-0.0089	0.9858	0	0	1	j	j
30	0.0025	0.0002	0.9831	0	0	1	j	j
31	-0.0003	0.0032	0.9964	0	0	1	j	j

A táblázat alapján látható, hogy az egyes mozgások váltási pontjánál a hálózat hibázik, de még mindig elfogadható eredményt ad.
A következőkben vizsgáljuk meg a többrétegű, egy kimenetű hálózatot. Ebben az esetben is a bemenetek megegyeznek az előző neurális hálózatok bemeneteivel, illetve egy kimenetű hálózatot hozunk létre az első esettel analóg módon.

- O1 = 0 ha [1 0 0] volt a parancskód (jobbra ment)
- O1 = 0,5 ha [0 1 0] volt a parancskód (egyenesen ment)
- O1 = 1 ha [0 0 1] volt a parancskód (balra ment)

A Matlab-ba beírt és lefuttatott program forráskódja a mellékletben található (*Matlab neurális hálózat többrétegű, egy kimenet* melléklet). A feldolgozott adatok eredményeit és az összevetést az eredeti értékkel a 4. táblázat tartalmazza.

4. táblázat: Többrétegű, egy kimenetű neurális hálózat eredményei
(forrás: a szerző saját munkája)

Sorszám	Kimenet		Neurális hálózat eredményei	Valós eredmények
1	0.0034	0	b	b
2	-0.0041	0	b	b
3	0.0117	0	b	b
4	0.1487	0	b	b
5	0.2736	0	b	b
6	-0.0115	0	b	b
7	0.4254	0,5	e	e
8	0.0973	0	b	b
9	-0.0300	0	b	b
10	-0.0472	0	b	b
11	0.0126	0	b	b
12	0.2984	0	b	b
13	0.5028	0,5	e	e
14	0.6099	0,5	e	e
15	0.5019	0,5	e	e
16	0.5091	0,5	e	e
17	0.5883	0,5	e	e
18	0.6015	0,5	e	e
19	0.6330	0,5	e	e
20	0.9481	1	j	j
21	1.0253	1	j	j
22	1.0306	1	j	j
23	0.9896	1	j	j
24	0.2540	0	b	j
25	0.9954	1	j	j
26	1.0154	1	j	j
27	0.9805	1	j	j
28	0.9633	1	j	j
29	0.9516	1	j	j
30	0.9782	1	j	j
31	0.9867	1	j	j

Látható, hogy az alkalmazott betanítás az egy rejtett rétegű, egy kimenetű hálózattal megegyező megoldást hozta, vagyis egyetlen hiba kivételével leköveti a tényleges mozgást, tehát a egyszerűbb egy rejtett rétegű hálózat is ugyanazt a megoldást hozta,

nem érdemes bonyolultabb többrétegű hálózatot használni, az egy rejtett rétegű, egy kimenetű neurális hálózat esetén is eljut a robot a célterületre.

7. Összefoglalás

A dolgozat célja egy kísérleti vezetőnélküli targonca modell - amely jelen esetben egy LEGO Mindstorsm EV3 robot volt - neurális hálózat alapú navigációjának fejlesztése. Egy rövid bevezető után bemutattam magát a kísérletben használt LEGO robotot és néhány megvalósított alkalmazását. A következő fejezetben prezentáltam a fejlesztésnél alkalmazott Omron F210 képérzékelőt, különös fókuszálva a navigációhoz szükséges beállításokra. Szükség volt egy témakutatásra vizuális irányítás témakörben, hisz fel kellett térképezni nemzetközi szinten, hogy jelenleg milyen kutatások és milyen eredmények születtek már. A bemutatott alkalmazások után rátértem egy rövid neurális hálózatok téma áttekintés után a saját fejlesztés és eredményeinek prezentálására. Részletesen leírtam, hogy milyen lépések vezettek el a végleges és működő verzióhoz, illetve az alkalmazott neurális hálózatok milyen eredmények hoztak. A nemzetközi szakirodalom alapján egyértelművé válik, hogy a neurális hálózatok alkalmazásának irányában komoly fejlesztések történnek az ipar különböző területein, így a logisztikában is. Jelen fejlesztés a vezető nélküli targonca anyagmozgatási problémáira adhat egy megoldást, amelyek közül a dolgozatban az egységrakomány felvételre koncentráltunk. A kísérleti eredmények azt a pozitív képet mutatják, hogy a fejlesztés folytatható és tesztelhető vezető nélküli targoncán, amely beavatkozás nélkül képes egységrakomány felvételre.

8. Irodalomjegyzék

1. LEGO Mindstorms EV3 információk (http://www.lego.com/hu-hu/mindstorms/support)
2. Little Talks Guitar Cover by Lego Mindstorms EV3 YouTube videó (www.youtube.com/watch?v=cXgB3lIvPHI)
3. On Longing Piano Cover by Lego Mindstorms EV3 YouTube videó **(**www.youtube.com/watch?v=TlmEpdyKwpI**)**
4. EV3 Lego Mindstorm Self Balancing Segway With Matlab Simulink YouTube videó (www.youtube.com/watch?v=xz1qZLN8ux0)
5. LEGO EV3 & MATLAB YouTube video (www.youtube.com/watch?v=I91uoc5fOcQ)
6. Omron F210 Operation Manual
7. Omron f210 Test/Measurement Execution
8. Tamoghna Das, I.N. Kar, S. Chaudhury – Simple neuron-based adaptive controller for a nonholonomic mobile robot including actuator dynamics, Neurocomputing 69 (2006) 2140-2150
9. Felipe N. Martins, Wanderley C. Celeste, Ricardo Carelli, Mário Sarchinelli-Filho, Teodiano F. Bastos-Filho – An adaptive dynamic controller for autonomous mobile robot trajectory tracking, Control Engineering Practice 16 (2008) 1354-1363
10. G. Cembrano, C. Torras, G. Wells – Neural Networks for robot control (1994)
11. Gordon Wells, Christophe Venaille, Carme Torras – Promising research Vision-based robot positioning using neural networks, Image and Vision Computing 14 (1996) 715-732
12. Claudiu Pozna, Radu-Emil Precup, Földesi Péter – A novel pose estimation algorithm for robotic navigation, Robotics and Autonomous Systems 63 (2015) 10-21
13. Xiuqing Wang, ZengGuang Hou, Feng Lv, Min Tan, Yongji Wang – Mobile robots' modular navigation controller using spiking neural networks, Neurocomputing 134 (2014) 230-238

14. Danie O. Sales, Diogo O. Correa, Leandro C. Fernandes, Denis F. Wolf, Fernando S. Osório – Adaptive finite state machine based visual autonomous navigation system, Engineering Applications of Artifical Intelligence 29 (2014) 152-162
15. Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, Valyon József – Neurális hálózatok (2006)
16. Dr. Kutor László – Intelligens Rendszerek Elmélete 5. előadásanyag (Óbudai Egyetem NIK, 2014. tavasz)
17. Somogyi Barnabás, Varga Alvaro Dávid – Vezető nélküli és kézi üzemre egyaránt alkalmas vontatótargonca fejlesztése (2013)
18. LEGO MINDSTORMS EV3 Programming Using Simulink YouTube videó (www.youtube.com/watch?v=dm428Q3GL6A)

9. Ábrajegyzék

10. Táblázatjegyzék

11. Mellékletek

Változók

x1,x2,x3 (y-ra ugyanez) - első koordináta százas/tizes/egyes helyiértéken lévő értéke
xc1,xc2,xc3 (y-ra ugyanez - megnézi, hogy üres-e az x1,x2,x3 érték, ha igen 1, ha nem 0 értéket vesz fel
xk1, yk1 - soroson küldött első koordináta (roboton teli kör)

x4,x5,x6 (y-ra ugyanez) - második koordináta százas/tizes/egyes helyiértéken lévő értéke
xc4,xc5,xc6 (y-ra ugyanez - megnézi, hogy üres-e az x4,x5,x6 érték, ha igen 1, ha nem 0 értéket vesz fel
xk2, yk2 - soroson küldött második koordináta (roboton körgyűrű)

xf,yf - fix koordináta (ahová eljut)

out - eltárolja a sorosan beérkezett üzenetet

eredm - létrehozott mátrix
iindex - ez kezeli hogy mindig új sorba írja a mátrixba az adatokat (iindex=iindex+1)
mymotor1,mymotor2 - így hivatkozik az ev3 motorjára
t1,t2 - timer, mennyi ideig menjen a motor
a1,b1,c1 - az (xk1;yk1), (xk2;yk2) és az x- tengely által meghatározott háromszög oldalai, ezek segítségével számolja ki a szöget a program koszinus tétellel
alfahelyzet - a szögeltérése a robotnak az x-tengelytől
t - a roboton lévő két kör középpontjainak a szakaszfelezőpontja milyen távol van a fix ponttol

s - port megnyitása

ev3 - robot megnyitása

Mátrix létrehozása

```
eredm = zeros(100,5)
iindex=1
```

Bejövö adatok feldolgozása:

```
x1 = str2num(out(6))*100;
   xc1 = isempty(x1);
x2 = str2num(out(7))*10;
   xc2 = isempty(x2);
x3 = str2num(out(8))*1;
   xc3 = isempty(x3);
if xc1 == 0
   x1 = str2num(out(6))*100;
 else x1 = 0;
end;
if xc2 == 0
   x2 = str2num(out(7))*10;
 else x2 = 0;
end;
if xc3 == 0
   x3 = str2num(out(8))*1;
 else x3 = 0;
end;
xk1 = x1 + x2 + x3

y1 = str2num(out(15))*100;
   yc1 = isempty(y1);
y2 = str2num(out(16))*10;
   yc2 = isempty(y2);
y3 = str2num(out(17))*1;
   yc3 = isempty(y3);
```

```
  if yc1 == 0
     y1 = str2num(out(15))*100;
   else y1 = 0;
  end;
  if yc2 == 0
     y2 = str2num(out(16))*10;
   else y2 = 0;
  end;
  if yc3 == 0
     y3 = str2num(out(17))*1;
   else y3 = 0;
  end;
yk1 = y1 + y2 + y3

  x4 = str2num(out(24))*100;
     xc4 = isempty(x4);
  x5 = str2num(out(25))*10;
     xc5 = isempty(x5);
  x6 = str2num(out(26))*1;
     xc6 = isempty(x6);
  if xc4 == 0
     x4 = str2num(out(24))*100;
   else x4 = 0;
  end;
  if xc5 == 0
     x5 = str2num(out(25))*10;
   else x5 = 0;
  end;
  if xc6 == 0
     x6 = str2num(out(26))*1;
   else x6 = 0;
  end;
```

```
xk2 = x4 + x5 + x6


  y4 = str2num(out(33))*100;
     yc4 = isempty(y4);
  y5 = str2num(out(34))*10;
     yc5 = isempty(y5);
  y6 = str2num(out(35))*1;
     yc6 = isempty(y6);
  if yc4 == 0
     y4 = str2num(out(33))*100;
   else y4 = 0;
  end;
  if yc5 == 0
     y5 = str2num(out(34))*10;
   else y5 = 0;
  end;
  if yc6 == 0
     y6 = str2num(out(35))*1;
   else y6 = 0;
  end;
yk2 = y4 + y5 + y6
xf = 447
yf = 96
```

Helyzeteltérés meghatározása (szögeltérés és távolság)

```
a1 = sqrt((xk1-(xk2+(xk1-xk2)/2)).^2+((yk2+((yk1-yk2)/2))-(yk2+((yk1-yk2)/2))).^2);
b1 = sqrt((xk1-(xk2+(xk1-xk2)/2)).^2+((yk1)-(yk2+((yk1-yk2)/2))).^2);
c1 = sqrt((xk1-xk1).^2+(yk1-(yk2+(yk1-yk2)/2)).^2);
alfa2= (a1.^2+b1.^2+-c1.^2)/(2*a1*b1);
alfahelyzet = acosd(alfa2);
t = sqrt((xf-(xk2+(xk1-xk2)/2)).^2+((yf)-(yk2+((yk1-yk2)/2))).^2)
eredm(iindex,1)=alfahelyzet;
eredm(iindex,2)=t;
```

```
iindex=iindex+1;
```

Soros port megnyitása, érték eltárolása

```
s = serial('COM5');
set(s, 'terminator', 'CR');
set(s,'BaudRate',38400);
 fopen(s)
fprintf(s,'*IDN?')
out = fscanf(s)
```

Robot kommunikáció megnyitása

```
myev3 = legoev3('wifi','192.168.0.101','00165345b434');
```

Robot balra mozog egy egységet

```
mymotor1=motor(myev3,'A');
mymotor1.Speed=30;
start(mymotor1);
t1 = timer;
t1.StartDelay = 0.2;
t1.TimerFcn = @(myTimerObj, thisEvent)stop(mymotor1),szogszam;
start(t1);
eredm(iindex,3)=1;
eredm(iindex,4)=0;
eredm(iindex,5)=0;
```

Robot jobbra mozog egy egységet

```
mymotor2=motor(myev3,'B');
mymotor2.Speed=30;
start(mymotor2);
t2 = timer;
t2.StartDelay = 0.2;
t2.TimerFcn = @(myTimerObj, thisEvent)stop(mymotor2),szogszam;
start(t2);
```

```
eredm(iindex,3)=0;
eredm(iindex,4)=0;
eredm(iindex,5)=1;
```

Robot előre mozog egy egységet

```
mymotor1=motor(myev3,'A');
mymotor2=motor(myev3,'B');
mymotor1.Speed=30;
mymotor2.Speed=30;
start(mymotor1);
start(mymotor2);
t1 = timer;
t2 = timer;
t1.StartDelay = 0.5;
t2.StartDelay = 0.5;
t1.TimerFcn = @(myTimerObj, thisEvent)stop(mymotor1);
t2.TimerFcn = @(myTimerObj, thisEvent)stop(mymotor2),szogszam;
start(t1);
start(t2);
eredm(iindex,3)=0;
eredm(iindex,4)=1;
eredm(iindex,5)=0;
```

Matlab neurális hálózat egy rejtett rétegű, egy kimenet

```
p=zeros(2,31)
q=zeros(1,31)
for i=1:31
p(1,i)=eredm(1+i,1)/90;
p(2,i)=eredm(1+i,2)/500;
if eredm(1+i,3)==1
        q(1,i)=0;
end
```

```
if eredm(1+i,4)==1
        q(1,i)=0.5;
end
if eredm(1+i,5)==1
        q(1,i)=1;
end
end
net = newff(p,q,20)
net1 = train(net,p,q)
ytest=zeros(1,31);
bemenet=zeros(2,1);
for i=1:31
   bemenet(1,1)=p(1,i);
   bemenet(2,1)=p(2,i);
   ytest(1,i)=sim(net1,bemenet);
end
```

Matlab neurális hálózat egy rejtett rétegű, három kimenet

```
p=zeros(2,31)
q=zeros(3,31)
for i=1:31
p(1,i)=eredm(1+i,1)/90;
p(2,i)=eredm(1+i,2)/500;
q(1,i)=eredm(1+i,3);
q(2,i)=eredm(1+i,4);
q(3,i)=eredm(1+i,5);
end

net = newff(p,q,20)
net2 = train(net,p,q)
ytest2=zeros(3,31);
bemenet=zeros(2,1);
```

```
kimenet=zeros(3,1);
for i=1:31
   bemenet(1,1)=p(1,i);
   bemenet(2,1)=p(2,i);
   kimenet=sim(net2,bemenet);
   ytest2(1,i)=kimenet(1,1);
   ytest2(2,i)=kimenet(2,1);
   ytest2(3,i)=kimenet(3,1);
end
```

Matlab neurális hálózat többrétegű, egy kimenet

```
net = newff(p,q,[20 10],{'tansig','logsig'},'trainlm')
net3 = train(net,p,q)
ytest3=zeros(1,31);
bemenet=zeros(2,1);
for i=1:31
   bemenet(1,1)=p(1,i);
   bemenet(2,1)=p(2,i);
   ytest3(1,i)=sim(net1,bemenet);
end
```