



SZEGEDI TUDOMÁNYEGYETEM
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
MÉRNÖKI KAR

***VALÓS IDEJŰ KÉPFELDOLGOZÁSSAL TÁMOGATOTT
ÖNVEZETŐ RC AUTÓ MEGVALÓSÍTÁSA***

TDK DOLGOZAT

Jernei Mihály

mechatronikai mérnök MSc

hallgató

Témavezetők:

Dr. Sarcevic Péter

főiskolai docens

Kóczi Dávid

tanársegéd

SZEGED

2024

TARTALOMJEGYZÉK

1. BEVEZETÉS	3
2. SZAKIRODALMI ÁTTEKINTÉS	4
2.1. Önvezető autók fejlődése, szenzorikai és mélytanulási technológiái	4
2.1.1. <i>Az iparág technológiai fejlődése</i>	5
2.1.2. <i>Az autonóm járművekben használt szenzorok típusai</i>	9
2.2. Helymeghatározás és térképezés az autonóm járművekben	12
2.2.1. <i>Helymeghatározás</i>	12
2.2.2. <i>Térképek típusai</i>	13
2.2.3. <i>SLAM technikák alkalmazása a lokális térképezésnél</i>	14
2.3. Az önvezetésnél alkalmazott képfeldolgozási technikák, módszerek	16
2.3.1. <i>Objektumfelismerés (Object Detection)</i>	16
2.3.2. <i>Szemantikai szegmentáció (Semantic Segmentation)</i>	17
2.3.3. <i>Sztereo látás (Stereo Vision)</i>	18
2.3.4. <i>Monokuláris mélységbecslés (Monocular depth estimation).....</i>	18
2.3.5. <i>Mozgásbecslés optikai áramlással (Optical flow).....</i>	19
2.3.6. <i>Vizuális odometria (Visual odometry)</i>	19
2.3.7. <i>Objektumkövetés (Object Tracking)</i>	20
2.3.8. <i>Vonaldetektálás és követés (Lane Detection and Tracking)</i>	21
2.4. Mélytanulási módszerek az autonóm járművek vezérlésében	22
2.4.1. <i>Mélytanulás és az autonóm járművek kapcsolata</i>	22
2.4.2. <i>Érzékelés és észlelés mélytanulási módszerekkel</i>	23
2.4.3. <i>Döntéshozatal és útvonaltervezés mélytanulással</i>	24
2.4.4. <i>Kihívások és jövőbeni fejlődési irányok.....</i>	25
3. VALÓS IDEJŰ KÉPFELDOLGOZÁSSAL TÁMOGATOTT ÖNVEZETŐ RC AUTÓ	27
3.1. Kritériumok meghatározása	27
3.1.1. <i>Az önvezető RC autó kritériumai:</i>	27
3.2. Időterv és célkitűzések	28
3.3. Hardver tervezése és kivitelezése	28
3.3.1. <i>Autóváz és a működtető elektronika kiválasztása</i>	28
3.3.2. <i>A központi vezérlő egységet képező hardver elemei</i>	31
3.3.3. <i>Rögzítő váz tervezése -3D nyomtatás</i>	33
3.3.4. <i>Rendszer blokkdiagramja és az elkészült autonóm modellautó.....</i>	33
4. A FEJLESZTETT SZOFTVER BEMUTATÁSA	35

4.1. A központi vezérlő egységen futtatott programok ismertetése	35
4.1.1. Alap program	35
4.1.2. Sebesség és kormányzás vezérlése	36
4.1.3. Tanító adatok gyűjtése.....	37
4.1.4. Előrejelzést végző program	40
4.2. A Windows környezetben futó kiegészítő programok bemutatása.	41
4.2.1. A tanító adatbázis elkészítése	41
4.2.2. CNN betanítása a tanítóadatok segítségével.....	42
5. A CNN MODELL BETANÍTÁSA ÉS TESZTELÉSE VALÓS KÖRNYEZETBEN.	44
5.1. Az első tesztpályán végzett kísérletek.....	44
5.1.1. Tanító adatok gyűjtése és a tanító adatbázis létrehozása.....	44
5.1.2. CNN betanítása és tesztelése	45
5.1.3. Pályakövetési pontosság mérése	52
5.2. Kísérletek a második tesztpályán.....	55
5.2.1. A betanított modellek teszteredményei.....	56
6. ÖSSZEFOGLALÁS ÉS JAVASLATTÉTEL	58
IRODALOMJEGYZÉK	
ÁBRAJEGYZÉK	
MELLÉKLETEK	

1. BEVEZETÉS

A projekt célja egy önvezető RC autó fejlesztése egy beágyazott, mesterséges intelligencia platform segítségével, amely gépi tanulással támogatott valós idejű képfeldolgozással képes egy előre felrajzolt pályát kamera felhasználásával folyamatosan megfigyelni és az útvonal mentén önállóan navigálni külső, emberi beavatkozás nélkül.

A témaválasztásom oka a manapság egyre nagyobb jelentőséggel bíró önvezető autók megjelenése, melyek már a mindennapokban is számos gépjárműben jelen vannak eltérő önállósági szinteken. Ezek a rendszerek a legtöbb esetben összetett szoftverek és szenzorok fúziójából épülnek fel, melyek magas szintű számításokkal képesek egy gépjármű vezérlésére a begyűjtött információk alapján valós környezetben, azonban a mesterséges intelligencia alkalmazása csak a közelmúltban kezdett jelentősebb térhódításba ezen a területen. Ebből kiindulva érdekes tématerületnek gondoltam egy pusztán mesterséges intelligenciával vezérelt autonóm modellautó elkészítését.

A fejlesztést projektszemlélettel végeztem, amely értelmében időtervet és hozzá kapcsolódó célkitűzéseket határoztam meg, melyek következetes betartásával lépésről lépésre haladtam a tervezéstől a kivitelezésig.

A dolgozat első fejezetében szakirodalmi áttekintést végzek, melyben kitérek az önvezetésnél alkalmazott különböző technológiákra, az alkalmazott szenzorokra, lokalizációs és térképezési megoldásokra, képfeldolgozási technikákra. Ismertetem a dolgozat alapjául szolgáló gépi tanulási módszerek önvezető járművekben használt típusait.

A második részben bemutatom az általam fejlesztett rendszer tervezési és kivitelezési lépéseit hardveres és szoftveres megközelítésből. A harmadik fejezetben részletesen ismertetem a saját fejlesztésű szoftver felépítését és működését. A negyedik részben tárgyalom az elkészített prototípussal végzett valós környezeti tesztek, azok eredményeit és a kiértékelésüket. Az utolsó fejezetben értékelő összefoglalást készítek, továbbá módosítási és továbbfejlesztési javaslatokat teszek.

A dolgozatban törekszem lényegre törően prezentálni és szemléltetni a fejlesztés menetét elméleti és gyakorlati szinten, az elkészült eszköz működését, valamint a gépi tanulási módszerek létjogosultságának vizsgálatát az önvezetés tudományterületén. A modellautó reprodukálhatósága nem célokom, melynek okán a dolgozatban csak a fejlesztés fontos lépései kerülnek teljes részletességgel bemutatásra.

2. SZAKIRODALMI ÁTTEKINTÉS

Ebben a fejezetben áttekintem azokat a szakirodalmakat, amelyek a dolgozatomban témájának alapjául szolgálnak és annak megértéséhez szükségesek. Bemutatom az önvezető autonóm járművek fejlődésének fontosabb mérföldköveit, részletezem az alkalmazott szenzorok típusait, azok működését, valamint ismertetem a térképezési és helymeghatározási módszereket. Kitérek az önvezetésnél használt képfeldolgozási technikákra, módszerekre, továbbá áttekintem a gépi tanulási módszereket az autonóm járművek szemszögéből.

2.1. Önvezető autók fejlődése, szenzorikai és mélytanulási technológiái

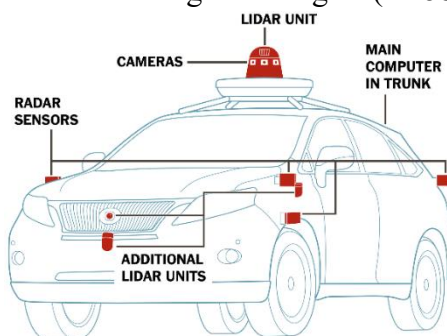
Az autonóm járművekkel kapcsolatban a technológia fejlődése során több elnevezés is ismertté vált a köztudatban. Ilyenek például a vezető nélküli autó, az önvezető autó, a számítógép által vezérelt autó, vagy az intelligens mobilrobot, melyek mind olyan felépítésű járműre utalnak, amely emberi beavatkozás nélkül képes navigálni saját mozgását. A biztonságos és megbízható vezérléshez a fedélzetén megtalálható érzékelők folyamatosan gyűjtik az információkat a környezetről, a jármű helyzetéről, az útviszonyokról, esetleges akadályokról, valamint a közelben lévő egyéb járművekről. Ezeket az adatokat a kamerák képeivel együttesen egy nagysebességű központi számítógép valós időben feldolgozza, kiértékeli, majd a navigációs térképekkel összevetve meghatározza a szükséges beavatkozásokat (**1.ábra**). (Murphey, Kolmanovsky & Watta, 2023)

Az önvezető képességgel rendelkező járműveket az automatizáltságuk és az önálló tevékenységük alapján 5 osztályba sorolhatjuk az **NHTSA**, vagyis az amerikai közlekedésbiztonságért felelős hivatal meghatározása szerint, akik többek között a különböző gépjármű-biztonsági szabványok betartatásáért is felelősek. (Anderson et al, 2016)

- **0. Szint: Nem automatizált járművek**, melyben kizárólag a vezető dönt az elsődleges funkciók felett, mint a kormányzás, fékezés, sebesség, hajtóerő. Az ember felelős a biztonságos közlekedésért.
- **1. Szint: Funkcióspecifikus automatizáltság**, ahol a vezető továbbra is teljes körű hozzáféréssel rendelkezik a jármű felett, azonban egyes részeket átengedhet az úgynevezett vezetéstámogató rendszereknek, melyek egymástól függetlenül láthatnak el fő funkciókat. Ide tartozik a menet- és kormánystabilizátor, vagy a vészhelyzetben működésbe lépő dinamikus fékrásegítés is.
- **2. Szint: Kombinált funkciók automatizálással** rendelkező járművek, melyekben legalább két elsődleges vezérlést együttesen ellátó rendszer található. A vezetőnek továbbra is teljes körű hozzáférése van az irányításhoz, felelős a biztonságos közlekedésért, viszont átadhatja az irányítás fő részeit az automata rendszereknek, mint például a kormányzást és a

sebességszabályozást. Ezen a szinten a rendszer elvárja a vezetőtől, hogy bármikor rendelkezésre álljon és biztonságosan vissza tudja venni az irányítást előzetes értesítés nélkül is. Ilyen rendszernek számít a sávtartással párhuzamosan működő adaptív sebességtartás is.

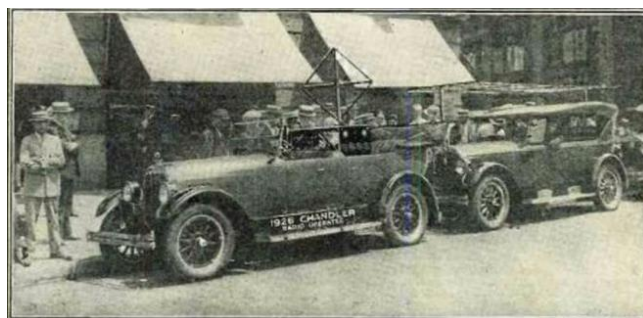
- **3. Szint: Korlátozott önvezető automatizáltság.** Ezen a szinten a vezető adott forgalmi helyzetekben a biztonság szempontjából kritikusnak számító elsődleges funkciók felett teljes mértékben átadhatja az irányítást a központi számítógépnek. A rendszer azonban időközönként visszacsatolást kér a vezetőtől, hogy bármikor képes átvenni az irányítást. Ilyen visszajelzés lehet a kormány megfogása, vagy a figyelem úton tartása, melyet kamera ellenőriz.
- **4. Szint: A teljes önvezető automatizálás** a legmagasabb autonóm szintnek felel meg. A biztonságos közlekedésért már csak a járműrendszer felelős, a vezetőnek csak az úticélt kell kijelölnie, minden mást az autó magától elvégez. (Anderson et al, 2016)



1. ábra: Önvezető autó felépítési ábrája (www.nytimes.com, 2024.08.08)

2.1.1. Az iparág technológiai fejlődése

Az autonóm járművek technológiai fejlődésének első, kezdetleges állomása a Houdina Radio Control által 1925-ben bemutatott rádió vezérelt autó volt, melyet rádióantennáin keresztül egy mögötte haladó másik autóból rádióimpulzusokkal lehetett irányítani (2. ábra). Az autó kormányzását, fékezését és sebességszabályozását a rádiójelekkel kapcsolgatott megszakítókra kötött kisméretű villanymotorok végezték. (Bimbrow, 2015)



2. ábra: Houdina Radio Control rádióvezérelt autója (www.discovermagazine.com , 2024.08.08)

Az ezt követő évtizedekben a nagy autógyártók, mint például az Oldsmobile, a Chrysler és társai számos technológiai újítást vezettek be, melyek alapjául szolgáltak és szolgálnak a mai vezetéssegítő rendszerek kialakulásának, amik elengedhetetlenek az autonóm járművek működéséhez. Ilyen például az önindító rendszer a Cadillac részéről, az automata sebességváltó az Oldsmobiltól, a hidraulikus kormányzás, a sebességtartó automatika és a blokkolásgátló fékrendszer a Chrysler által. (Du, 2023).

A gépjárművek fejlődésével egyre több egyetem kezdett autóiipari vállalatokkal együttműködve kutatásokat, tanulmányokat végezni az autonóm közlekedés tématerületén belül, melyek két fő irányt vizsgáltak. Az egyik koncepció az automatizált autópályák lehetőségeit elemezte, melynek első felvetése New York-ban az 1939-es Futurama című világkiállításon vetődött fel, ahol olyan járműveket vázoltak fel, melyek az útburkolatba épített elektronikai eszközök segítségével képesek saját magukat navigálni. Az első ilyen jellegű rendszer demonstrálására 1997-ben Californiában került sor „DEMO 97” néven, ahol bemutattak egy nyolc autóból álló konvojt, amely az autópálya egy 7,6 mérföldes szakaszán teljesen önállóan és összehangoltan képes volt végig menni (3. ábra). A másik fő irány a kutatásokban a félig autonóm, illetve a teljesen autonóm járművek fejlesztésére összpontosult, melyek teljesen függetlenek az útburkolatba épített infrastruktúráktól. Ezen a területen jelentősebb áttöréseket értek el a kutatások, például az 1980-as évek elején a müncheni Bundeswehr egyetem bemutatott egy látás vezérelt járművet, ami forgalom nélkül 100 km/h-s sebességgel tudott navigálni. A Carnegie Mellon Egyetem NavLab projekt keretein belül egy 11 modellből álló járműsorozatot fejlesztett ki a 2000-es évek elejéig, melyek közül az 5. modellel végzett, Amerikát átívelő tesztvezetésen, az idő 98%-ban az autó önállóan kormányzott, még a gázt és a féket a vezető személy kezelte (4. ábra). (Anderson et al, 2016)



3. ábra: Az automata autópálya bemutatója 1997-ben a DEMO 97 programmal (www.researchgate.net, 2024.08.08)



4. ábra: Az Amerikát átutazó NavLab 5-ös számú autonóm járműve (www.researchgate.net , 2024.08.08)

Az önvezető autonóm járművek jelentős fejlődését segítették elő a különböző kihívás alapú versenyek is, melyek közül az egyik legjelentősebb a DARPA névre keresztelt autonóm járműveknek hirdetett verseny az Egyesült Államok Fejlett Védelmi Kutatási Projektek Ügynöksége rendezésében. Ezen rendezvény célja az önvezetés tudományterületében rejlő technológiai újítások felgyorsítása volt. Az első versenyt 2004-ben rendezték a Mojave sivatagban, ahol autonóm robotokká fejlesztett gépjárművekkel kellett egy ismeretlen útvonalat teljesíteni a sivatagban off-road terepen 10 óra alatt. Első alkalommal senkinek sem sikerült célba érnie, azonban 2005-ben a Stanford Egyetem fejlesztőcsapata megnyerte a kihívást. A kijelölt 175 mérföldes távot 6 óra 53 perc alatt teljesítették egy „Stanley” névre hallgató Touareg R5 TDI típusú gépjárművel (5. ábra), amely 5 SICK lézeres távolságmérővel, RGB kamerával, 25 GHz-es radarral és GPS pozícionáló rendszerrel, valamint 6 db Pentium M számítógéppel volt felszerelve. A járműnek nagy sebességgel történő útkeresést, valamint az akadályok észlelését és elkerülését kellett megvalósítani a verseny során. (Thrun et al, 2006)



5. ábra: A 2004-es DARPA győztes "Stanley" autója (www.si.edu , 2024.08.08)

A 2000-es évektől kezdve a szenzorika és a számítástechnika területén bekövetkezett nagy mértékű fejlődés hatására egyre nagyobb cégek és vállalatok álltak az autonóm járművek fejlesztői sorába, mint a Google, a Tesla, a General Motors tulajdonában lévő Cruise, a kínaiak oldaláról a Baidu, az Apple, a Ford és a Volkswagen által támogatott Argo AI és még további autógyártók.

A Google 2009-ben önvezető autók fejlesztésébe kezdett, amely során 2012-re már 300 000 autonóm módon levezetett mérföldet tudhattak maguk mögött. Ebből a kezdeményezésből született meg a Waymo alvállalat, amely már 2017-ben nyílt közúti tesztet kezdett az arizonai Phoenix városában. Az önvezető járműveik érzékelői között megtalálható volt egy kis

hatótávolságú, egy nagy felbontású és egy nagy hatótávolságú LiDAR, egy milliméter hullámú radarrendszer, melyek ködben, esőben és havas napokon is megfelelő látást tudtak biztosítani az autónak. Ezek segítségével a 360°-os látómezővel az autó képes mérni a jármű körül közlekedő más járművek sebességét. A rendszer további alkotóeleme az összetett kamerarendszer, melyet az emberi látás formájára alakítottak ki, viszont a látómezőt 120°-ról 360°-ra bővítették. Képes érzékelni a színeket, így elősegítve az útjelzések, táblák, kritikus járművek detektálását akár gyenge fényviszonyokban is, továbbá hangérzékelők segítségével a megkülönböztető járművek szirénáit is észlelni tudja. A Waymo 2020-óta Phoenix nagyvárosi körzetében teljesen vezető nélküli fuvarszolgáltatást lát el, amit bárki igénybe vehet (6. ábra). (Du, 2023)



6. ábra: Waymo önvezető autó 2017-ben és napjainkban
(europe.autonews.com , www.selfdrivingcars360.com, 2024.08.09)

A Tesla autógyártó 2014-ben mutatta be az izraeli Mobileye vállalattal együtt fejlesztett autonóm vezetési szoftverét, amely fél autonóm vezetési és parkolási funkciókat tudott megvalósítani, kereskedelmi forgalomba azonban csak 2015-ben került Autopilot 7.0 néven. A továbbiakban újabb és újabb verziókat jelentettek meg, melyekkel például a jármű már távolról, vezető nélkül is mozgathatóvá vált. A Tesla érzékelői között más autógyártókkal szemben nem szerepelt a **LiDAR** technológia, csak a kamerák és a radarrendszer kapott helyet az autókban. A mélységérzékelést így a **LiDAR** helyett a fejlett fedélzeti szoftverek segítségével valósították meg, melyek a kamerák képeinek összefűzését használták fel. A Tesla 2016-ban piacra dobott elektromos autókban már elérhető volt az Autopilot 8.0, amely a kamerák helyett a fejlett radarrendszerre támaszkodott, így lehetővé téve a rossz látási viszonyokban való navigálást. A 2016-os év végétől gyártott autók mindegyike megkapta a megfelelő hardvercsomagot, amely szükséges a teljes önvezetéshez. Az új Enhanced Autopilot-ot támogató autókkal elérhetővé vált az automatikus sávváltás, az autópályák közötti önálló átmenet és a szintén önálló párhuzamos és merőleges parkolási funkciók. 2019-ben már a 3-as autonóm szintnek megfelelő funkciók is megjelentek, így az autók már megerősítés nélkül is képesek voltak az autópályán navigálni, viszont a vezetőknek továbbra is visszacsatolást kellett adnia időközönként a

szoftvernek a figyelméről. Napjainkban már a teljes önvezetés bétaverziói is elérhetőek a Tesla autókban (7. ábra). (Maanyu et al, 2020)



7. ábra: A Tesla teljes önvezető rendszerének béta verziója által modellezett környezet (www.autoevolution.com, 2024.08.09)

A jelenlegi trendek azt mutatják, hogy a jövőben az autonóm járművek fejlődése főként az infrastruktúrától független, önállóan működő járművek területén várható. Erre mutat példát a 2024-ben első alkalommal megrendezett autonóm járművek versenye Abu Dhabiban, ahol Forma 1-es versenyautók átalakított verziói sofőr nélkül mérték össze önvezető rendszereiket. Ezen kezdeményezés is erős szerepet játszik az önvezetés technológiájának gyorsuló fejlődésében, melynek következményeként egyre több önvezető autó jelenhet meg az utakon. Az autonóm járművek tudományterületén megjelenő új technológiák potenciálisan javíthatják a közlekedésbiztonságot, csökkenthetik a balesetek számát, és pozitívan befolyásolhatják a közlekedésben résztvevők morálját.

2.1.2. Az autonóm járművekben használt szenzorok típusai

Az önvezetés technológiájában a járművek szenzorokon keresztül képesek információt gyűjteni a környezetükről. Ahhoz, hogy a központi vezérlőrendszer valós időben nagy sebességgel, magas minőségben elemezze a környezetét és döntést hozzon a szükséges beavatkozásról, az érzékelőknek észlelő és helymeghatározó képet kell adniuk a környezetről. Két fő osztályukat alkalmazzák ezen a területen, ezek az **exteroceptív**, és a **proprioceptív érzékelők**, másnéven a külső és belső érzékelők. Előbbivel a környezet érzékelése és az objektumoktól való távolság számítása történik, utóbbi a rendszer belső értékeinek méréséért felelős. Ezek számos kombinációja megtalálható a gyakorlatban alkalmazott önvezető járművekben (8. ábra). (Campbell et al, 2018)

Exteroceptív érzékelők: Ide sorolhatók a **LiDAR**, a **radar**, a **kamerák** és az **ultrahangos távolságérzékelők**. A **LiDAR (Light Detection and Ranging)** technológia egy lézer alapú fényérzékelési rendszer, amely működése közben fényimpulzus sorozatokat bocsát ki, majd azokról a környezetben lévő tárgyakról való visszaverődésük alapján háromdimenziós

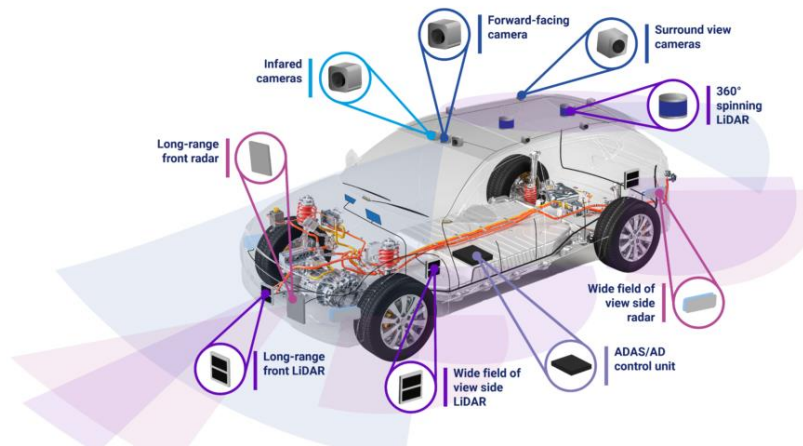
pontfelhőt alkot. Használhat látható, UV vagy infra tartománybeli fényt is, látószöge 360° is lehet, így autók számára ideális leképezést valósít meg, viszont a szkennelési pontok sűrűsége erősen befolyásolja az eredményül kapott kép felbontását. Pontossága erősen függ az időjárási körülményektől, mivel a fényvisszaverődést a felhő, a köd, az eső jelentősen torzíthatja, így ilyenkor szükségessé válik más szenzorok alkalmazása is a lokalizáció és térképezés minőségének fenntartása érdekében. (Ng, 2021)

A **radartechnológiát** számos területen használják objektumok távolságának, orientációjának és sebességének mérésére. Az elektromágneses sugárzás elvén működik, több frekvenciasávban. A magasabb frekvenciájú radarokkal magas felbontás érhető el, így a járművek esetében valós időben megkülönböztethetővé válnak a különböző objektumok. Hatótávolságuk 50-100 m, de adott esetben 150 m feletti távolságok érzékelésére is alkalmazhatók. Használatuk kifejezetten előnyös, mivel pontosságukat nem befolyásolja az időjárás, így szélsőséges körülmények között is megbízhatóan működnek, továbbá képesek átlátni az objektumokon is és érzékelni a látómezőn kívüli mozgó tárgyakat is. Az objektumok relatív sebessége is mérhetővé válik általuk, amely a forgalomban lévő többi jármű mozgáselemzésénél elengedhetetlen paraméter. (Campbell et al, 2018)

Szinte minden önvezető tulajdonsággal felruházott jármű esetében megtalálhatóak a **kamerák**, melyekkel a környezetről kaphatunk jó dinamikartományú, nagy felbontású digitális képeket. Ezek a képességek lehetővé teszik az objektumok észlelését, osztályozását, textúrák pontos detektálását. A mai kamerák ára egyre alacsonyabbá válik, így nem ritka, hogy az autógyártók több típust is alkalmaznak a járművekben. Ilyenek a **sztereó**, a **monokuláris**, az **infravörös** és a **mélységérzékelő kamerák**, melyek rengeteg adatot képesek generálni, így azok valós idejű feldolgozásához komoly számítástechnikai erőforrásokra van szükség. Gyenge fényviszonyokban azonban pontosságuk romlik, ezért nem elegendő csak rájuk hagyatkozni, így sok esetben együttesen használják **LiDAR** és **radar** rendszerekkel egyfajta fúziót alkotva, melynek köszönhetően a különböző szenzorok kiküszöbölik egymás gyengeségeit a szélsőséges esetekben. (Ng, 2021)

Az **ultrahangos érzékelők** hanghullámok segítségével képesek mérni objektumok távolságát. Ez egy jól bevált, megbízható technológia, mivel számos iparágban alkalmazzák évek óta, köztük az autóiparban is, mint parkolószenzor. Működése során egy adó meghatározott frekvenciájú hanghullámokat bocsát ki, a hullámok visszaverődnek a környezetben lévő tárgyakról és visszatérnek a vevőhöz. A kibocsátás és a visszaverődés között eltelt időből számítható a távolság, amely más érzékelőkhöz képest a legpontosabb eredményt adja. A hanghullámokat azonban a hőmérséklet és a páratartalom változása könnyen

befolyásolhatja, így rontva a mérés pontosságát. Ennek kiküszöbölésre már léteznek beépített algoritmusok, melyek figyelembe veszik a környezeti feltételeket is a mérések során. (Campbell et al, 2018)



8. ábra: Autonóm autók exteroceptív szenzorrendszere
(www.linkedin.com, 2024.08.10)

Proprioceptív érzékelők: A belső érzékelők közé tartozik többek között a **GPS**, az **inerciális mérőegység (IMU)** és az **enkóderek**, melyek előfordulnak az önvezető funkciókkal bíró járművekben. A **Global Positioning System**, vagy ismertebb rövidítésével a **GPS**, egy műholdas navigációs rendszer, amely szintén egy régóta alkalmazott technológiának számít az autópárházban. Működési elve a Föld körül keringő navigációs műholdak által sugárzott rádiójeleken alapul. Az autóban lévő vevőegység egyszerre 4 műhold pontos időjelzéseit fogadja, majd a vételi időeltérésekből számítható a műholdaktól mért távolság, amelyből következik a Földön vett 3 dimenziós pozíció meghatározása. Ennek a rendszernek azonban vannak hátrányai egy gyorsan mozgó járműnél. Ilyen a több méteres pontatlanság és az alacsony frissítési frekvencia. Ezek javítására fejlesztették ki a **differentiális GPS (DGPS)** rendszert, amely a helymeghatározás pontosságát frekvenciapont segítségével képes javítani. További megoldásként szolgál a szenzorfüzió bevezetése is, ahol például az **inerciális mérőegység (IMU)** adatait **Kalman-szűrő** segítségével egyesítik a **GPS** adataival annak érdekében, hogy pontosabb valós idejű pozíciót és mozgásinformációt kapjanak. (Du, 2023)

Az **inerciális mérőegység**, vagy rövidítve az **IMU** szenzor segítségével mérhetővé válik egy test gyorsulása, szögsebessége és a mágneses tere. Felépítésében általánosan három gyorsulásmérő, giroszkóp és magnetométer található, melyek a derékszögű koordináta-rendszer x, y és z tengelyei mentén helyezkednek el. Autonóm járművek vezérléséhez gyakran alkalmazzák, mivel általuk meghatározható azok lineáris sebessége, dőlésszöge és szöghelyzete egy globális referencia kerethez viszonyítva, melyek a precíz irányításhoz elengedhetetlen

adatok. Hátránya azonban mint minden szenzornak, ennek is akad. Az egyik, hogy önmagában nem képes meghatározni egy jármű aktuális helyzetét, csak a mozgásadatait, továbbá hajlamos az időben egyre növekvő hibahalmozódásra, melyek úgynevezett „drift” hibát okoznak a mért értékekben. Ezek javítására használhatóak a **GPS** által szolgáltatott adatok, melyekkel fuzionálva megkaphatók a pontos helyadatok és a „drift” hiba folyamatos korrekciója is megvalósítható. (Campbell et al, 2018)

Az autonóm járművek pontos pozíció és helyzetmeghatározásához használhatók még az elektromechanikus **enkóderek**, melyek egy tengely lineáris vagy szöghelyzetét képesek digitális jelekké alakítani. Két fő típusuk létezik, ezek az inkrementális és az abszolút enkóderek. Előbbi minden fordulatnál egy meghatározott számú impulzussorozatot generál, még az utóbbi a diszkrét helyzetnek megfelelő bitfolyamot eredményez. Ezek az eszközök elsősorban a kerekeknél helyezkednek el, így a kilométer számlálást tudják segíteni, viszont gyakran az autó relatív helyzetének meghatározására is felhasználják, úgynevezett holtpontszámításra. Ezzel a technikával költséghatékonyan és egyszerűen becsülhető egy jármű helyzete, viszont a kerekek és a talaj közötti csúszások magas mértékű mérési pontatlanságot okozhatnak, így ezeket az eszközöket is szenzorfüzókban alkalmazzák (Campbell et al, 2018)

A fentiekben bemutatott érzékelőknél látható, hogy mindegyik rendelkezik olyan hátrányokkal, melyek az önvezetésnél megengedhetetlen hibákat okoznának. Ebből kifolyólag különböző kombinációkban, szenzorfüzókban alkalmazzák ezeket az érzékelőket a gyengeségek kiküszöbölése érdekében. Ilyenek például a **képfeldolgozó rendszerekkel** kombinált **LiDAR** és **radar**, a **GPS** és **IMU** párosítás, valamint a **képkalkotó** és **odometriai rendszerek** ötvözése. Tökéletes megoldás a technológia jelenlegi állása szerint még nem alakult ki, de számos kutatás zajlik az összes kritériumnak megfelelő multiszenzoros fúzió kifejlesztésére. (Zhang et al, 2023)

2.2. Helymeghatározás és térképezés az autonóm járművekben

2.2.1. Helymeghatározás

Az autonóm járművek helymeghatározó rendszerei alapvető fontosságúak a járművek biztonságos és pontos navigációjában. Ezeket 3 fő kategóriába sorolhatjuk a felhasznált érzékelők alapján:

- **Jelpozícionálás alapú rendszerek (GNNS):** Ezek a rendszerek a **globális navigációs műholdrendszert (GNSS)**, például a **GPS**-t használják a jármű helyzetének meghatározására. Bár a **GPS** technológia széles körben elterjedt és megbízható, különféle környezeti hatások, mint például az ionoszféra zavarai vagy a műholdak pozíciós hibái, korlátozhatják a pontosságát.

- **Inerciális mérőegységen alapuló rendszer:** Ezek a rendszerek a korábban említettek alapján belső mérőegységeket használnak, amelyek a jármű mozgási információit (például gyorsulást, szögsebességet) mérik. Az **IMU** lehetővé teszi a jármű mozgásának és orientációjának folyamatos követését.
- **Környezeti jellemzőkön alapuló rendszer:** Ezek a rendszerek a jármű környezetéből származó adatokat használják a helymeghatározáshoz, amelyhez a **kamerák**, **LiDAR** és **radar** szenzorok segítségével gyűjtenek információkat. Ezek az érzékelők lehetővé teszik a jármű számára, hogy a környezetéhez viszonyítva határozza meg a helyzetét, például az út menti objektumok, sávjelzések vagy épületek alapján. (Murphey, Kolmanovsky & Watta, 2023)

Ezen rendszereknek mindegyikének megvannak a gyengeségei a korábban taglalt szenzorok tulajdonságaiból adódóan, így az önvezetés biztonságos megvalósítása miatt ezek kombinációját, fúzióját részesítik előnyben. Például a **GPS** és **IMU** szenzorok párosítása lehetővé teszi, hogy a **GPS** jel elvesztésekor is folytatódjon a navigáció az **IMU** szenzor által nyújtott mérési adatokra támaszkodva. Hasonlóképpen a **LiDAR** és **kamerák** által kivitelezett egyidejű lokalizációs és térképezési módszerek, melyeket a korszerű **SLAM** technikákhoz tartoznak, szintén pontos helymeghatározást nyújthatnak a navigáció során függetlenül a **GNSS** rendszerektől. Ennek működését egy későbbi fejezetben taglalom részletesebben. (Murphey, Kolmanovsky & Watta, 2023)

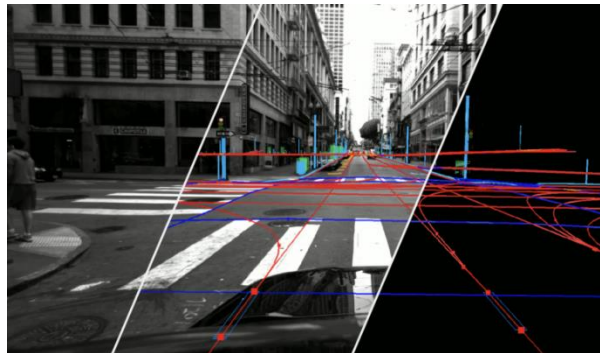
2.2.2. Térképek típusai

A szenzorok bemutatása alapján az autonóm járművek számos érzékelővel tájékozódnak a körülöttük lévő környezetről, melyek adataiból digitális leképezést végeznek arról. Ezt a folyamatot térképezésnek hívjuk, amely elengedhetetlen eleme az önvezetés technológiájában a navigáció, lokalizáció és útvonaltervezés számára. A térképek meghatározására több definíció is létezik, viszont általánosságban elmondható, hogy egy térkép a valós világ jellemzőinek, illetve az objektumok elhelyezkedésének reprezentációja. Az autonóm járművekben **offline**, illetve **online** térkép is rendelkezésre áll, melyek általában részletesebb információkat hordoznak, mint a hagyományos változatok. (Wong & Gu, 2020).

Az **offline**, vagy más néven **globális térképek** előzetesen készülnek el a navigációs és helymeghatározási rendszerek támogatása érdekében. Ezen a területen jelentős piaci verseny áll fenn, hiszen az önvezetés képességét jelentősen képes befolyásolni a rendelkezésre álló térképek minősége és részletessége. Három alapvető térképtípus tartozik ebbe az osztályba, ezek a **digitális térképek**, melyek utcai elemeket és útvonalakat ábrázolnak, a **továbbfejlesztett digitális változatok**, melyek már az útszakaszok íveit, oszlopokat és sávszinteket is

tartalmaznak, valamint a **nagyfelbontású (HD) térképek** (9. ábra), ahol a környezet már háromdimenziós módon, több rétegben van rögzítve. Ilyen rétegek az alap réteg, a geometriai térkép réteg, a szemantikai réteg, a prior térkép réteg és a valós idejű réteg. A HD térképeknek tartalmazniuk kell előzetes számításokat, pontos információkat az objektumok jellemzőiről, elhelyezkedéséről, 2D-s vagy 3D-s szemantikus objektumokat, általános adatokat a forgalomról, robusztusan kell viselkednie a dinamikus objektumokkal szemben, valamint globálisan megoszthatónak kell lennie. Az **offline** térképeknél gyakran találhatóak hibák, mivel a valós világ könnyedén megváltozhat. A változás hatására az adatok gyorsan elévülnek, így szükségessé teszi azok folyamatos frissítését. (Charroud et al, 2024)

Az **online vagy lokális térképezés** esetén a térképek készítése valós időben történik a jármű szenzorai által gyűjtött környezeti információkból. A korabeli önvezetési megoldásokban a térképezést és a lokalizációt külön kezelték, azonban az elmúlt években számos sikeres kutatási eredmény született az egyidejű lokalizáció és térképezési, másnéven **SLAM** technikák területén, így az autonóm járművek rendszerébe is igen hamar integrálódtak az újítások. Ezek a módszerek különösen hasznosak olyan területeken, ahol a térképek hiányosak, vagy egyáltalán nem állnak rendelkezésre, mivel a jármű ezek segítségével saját maga képes az aktuális környezetről térképet készíteni, ahol saját helyzetét pontos becslésekkel tudja felügyelni. Hátránya, hogy meglehetősen komplex algoritmusok szükségesek a hatékony működéséhez. (Charroud et al, 2024)



9. ábra: Valós városi környezet leképezése nagy felbontású térképpel
(www.autonomousvehicleinternational.com, 2024.08.11)

2.2.3. **SLAM technikák alkalmazása a lokális térképezésnél**

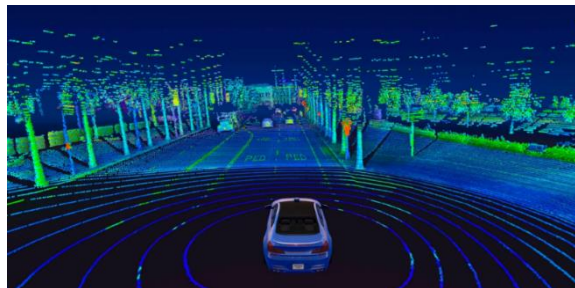
A lokális térképkészítés során az autonóm jármű a környezetének mérhető adataiból képes egy digitális, nagy felbontású térképet készíteni, melyhez több szenzor adatait is felhasználja. A korábban ismertetett **radar** és **LiDAR** érzékelők segítségével a környezet letapogathatóvá, szkennelhetővé válik technikától függően rádióhullámokkal, vagy lézernyalábok sorozatával. Ezek az érzékelők az autóhoz illesztett alap koordináta-rendszerben elhelyezve egy háromdimenziós pontfelhőt képeznek, melyek a környezetben lévő objektumok

alakját, kiterjedését, helyzetét tartalmazzák adott felbontásban (**10. ábra**). Ezeket az információkat nevezzük geometriai adatoknak, melyekre nagy mértékben támaszkodnak az egyidejű helymeghatározó és térképező algoritmusok, vagyis a **SLAM** technikák, ahol a rendszer térképkészítés közben párhuzamosan képes becsülni a jármű pozícióját az éppen készülő térképen. (Kamijo, Gu & Hsu, 2015)

A **SLAM** algoritmusok az elmúlt évtizedben jelentős fejlődésen mentek keresztül, amely lehetővé tette azok elterjedését. Az online lokalizáció során igen elterjedt megoldássá vált a kiterjesztett **Kalman szűrővel** kombinált **SLAM(EKF-SLAM)** és a **részecskeszűrő alapú SLAM**, azonban újabb kutatásokban az úgynevezett gráf alapú, **GraphSLAM** algoritmusok még az előbbi kettő módszert is képesek felülmúlni, mivel akár nagy léptékű térképezést is meg tudnak valósítani **GPS** információk beépítésével. Ezzel a típussal akár magas minőségű kétdimenziós útfelszín térkép is készíthető, ha a **GPS**, **IMU**, **kerékodometria** és **LiDAR** szenzorfüzést integrálnak egy autonóm járműben. (Wong & Gu, 2020)

Az autonóm jármű pozicionálásánál a következő környezeti tulajdonságok felismerésében képes a **SLAM** segítséget nyújtani:

- **Járdaszegélyek észlelése** az út és szegélyek közötti szintkülönbségek alapján, amely 2D **LiDAR** által mért távolsági értékekben egyértelmű kiugrásként jelenik meg, viszont a kétdimenzióban adathiány jelentkezhet, így célszerű 3D-s érzékelőt alkalmazni, amely elnyomja ezt a hibát.
- **Útburkolati jelek detektálása** a **LiDAR** érzékelők infravörös fénytartományba való kiterjesztésével, amely során a különböző felületekről, festékekről visszaverődő fény intenzitási adataiból hisztogramok segítségével kiszűrhetők az útburkolati jelek, így azok digitálisan leképezhetővé válnak kamerák használata nélkül is.
- **Tereptárgyak és épületek detektálása hibrid térkép alapú SLAM** technikával, ahol a jellemzőtérkép és rácstérkép kombinálásával az épületeket a rácsookban, a kisebb objektumokat, a fákat és közlekedési lámpákat pedig a jellemzőtérképen lehet elhelyezni. (Charroud et al, 2024)



10. ábra: SLAM technikával leképezett háromdimenziós térkép
(medium.com , 2024.08.16)

2.3. Az önvezetésnél alkalmazott képfeldolgozási technikák, módszerek

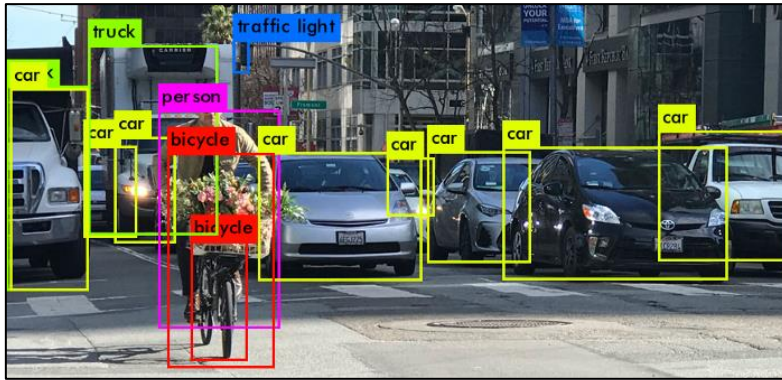
Az önvezető járművek érzékelői között szereplő kamerák elsősorban nyers képet szolgáltatnak a környezetről. Ezek feldolgozásával számos olyan információ birtokába kerülhet a fedélzeti számítógép, melyek elengedhetetlenek a navigációban, a környezeti tényezők és szituációk felismerésében. A képfeldolgozási módszerek közül az autonóm járművekre nézve az *objektumfelismeréssel*, *szemantikus szegmentálással*, a *sztereó látással*, a *mélységi becsléssel*, az *optikai áramlással*, a *vizuális odometriával*, *objektumkövetéssel* és a sávtartáshoz szükséges *vonal detektálással és követéssel* találkozhatunk a leggyakrabban.

2.3.1. Objektumfelismerés (Object Detection)

Az *objektumfelismerési* eljárás során egy képen található objektumok azonosítása és lokalizációja végezhető el. Az algoritmusok között két megközelítés létezik: az egyik módszer szerint a képen minden lehetséges részletet meg kell vizsgálni, viszont ez meglehetősen időigényes és gyakran hibás eredményhez vezet. A másik megoldás célzottan csak adott részletekkel foglalkozó speciális detektorokat használ, mint az arcfelismerők, gyalogosdetektorok és mélytanulás alapú algoritmusok. Ezek gyorsabbak és hatékonyabbak, különösen az autonóm járművek esetében, ahol fontos a gyalogosok, közlekedési táblák és járművek pontos felismerése. (Szeliski, 2022)

Az elmúlt években a rendelkezésre álló tesztadatok, melyek alapján ezen algoritmusok képesek az objektumok azonosítására, már több ezer jól definiált osztályt tartalmaznak, így magas megbízhatósági szintű algoritmusok fejlesztését segítik. Ilyen nyilvános adatkészlet például a **Microsoft COCO** adatkészlete is. (Szeliski, 2022)

Az objektumfelismerés fejlődésének kulcsfontosságú tényezője a mélytanulás elterjedése volt, melynek köszönhetően a modern objektumfelismerők nagy része már mélytanulási módszereket alkalmaz. Ide tartozik az **R-CNN (Region-based Convolutional Network)** modell, amely első lépésben téglalap alakú régiókra bontja a képet, majd azokat több lépésben osztályozza. Gyorsabb változatai az egylépcsős algoritmusok, mint például az **SSD (Single Shot Multibox Detector)** és a **YOLO (You Only Look Once)** (11. ábra) modellek, melyek csak egyetlen lépésben végzik az elemzést. A legelterjedtebb megoldások között a **YOLOv4** és a **RetinaNet** modellek találhatók, amelyek gyors és megbízható eredményeket biztosítanak. (Szeliski, 2022)

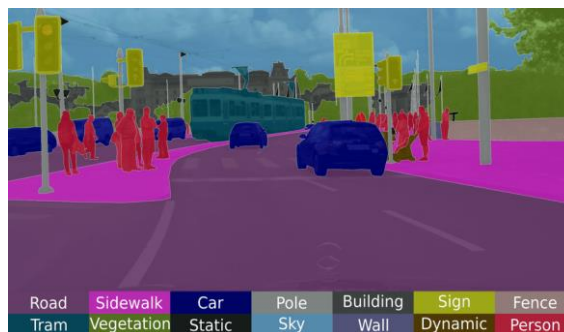


11. ábra: YOLO modell eredménye egy közlekedési kép esetén
(medium.com, 2024.08.17)

2.3.2. Szemantikai szegmentáció (Semantic Segmentation)

A gyakran alkalmazott képfeldolgozási módszerek közé tartozik a **szemantikai szegmentáció**, melynek feladata a képek vizuálisan jelentős régiókra való felbontása további elemzéshez és értelmezéshez. Ezt a képpontok egyenkénti osztályozásával, enkódolásával lehet megvalósítani. A módszer az autonóm vezetés területén lehetővé teszi a közlekedési szituációk pontos értelmezhetőségét, kiértékelhetőségét, amely fontos része a biztonságos navigációnak. A kamerák által kapott képeken elkülöníthetővé válnak többek között az útfelületek, járdák, járművek, gyalogosok (12. ábra). (Fan, Guo & Bocus, 2023)

A technológia ezen területén is jelentős előrelépést eredményezett a mélytanulás alapú modellek megjelenése. Elsők között a **Fully Convolutional Network (FCN)** segítségével már bármilyen méretű kép feldolgozható volt. Az úgynevezett piramis alapú megközelítések között a **DeepLab** és az **Atrous Spatial Pyramid Pooling (ASPP)** modellekkel vált elérhetővé a felbontás romlása nélküli nagyobb látómező. Egyes modellek a kontextuális információk kódolása mellett dekóder modulokat alkalmazva képesek a szegmentációs eredményeken javítani az objektumhatárokat. A különböző modellek, mint a Tesla által használt **UNet**, a **SegNet** vagy az **ENet** mindegyike egyedi módszerekkel próbálja a dekódolást optimalizálni annak érdekében, hogy a lehető leggyorsabban a legpontosabb eredményt adják kimenetként. (Fan, Guo & Bocus, 2023)

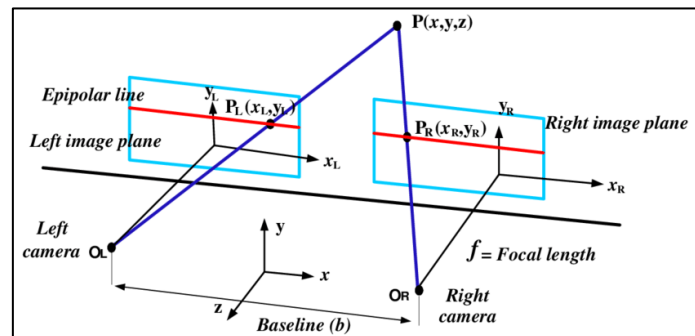


12. ábra: Szegmentált utcakép pontos objektumhatárokkal
(www.researchgate.net, 2024.08.17)

2.3.3. Sztereó látás (Stereo Vision)

A számítógépes látás alapvető feladata a sztereó illesztés, amely során két kamera által készített kép együttes vizsgálatával objektumok mélységérzékelését lehet megvalósítani. Ehhez egy diszparitási térképet alakítanak ki a képek alapján, amely megmutatja, hogy egy-egy tárgypont milyen mértékben mozdul el a két képen, ami fordított arányosságban áll a vizsgált pont távolságával. Az autonóm járművek esetében a mélységérzékelés már a korábban ismertetett érzékelőkkel és módszerekkel is lehetséges, azonban a sztereó illesztés is kiválóan megfelel erre a célra, így biztosítva, hogy az autonóm rendszer redundáns legyen. (Fan, Guo & Bocus, 2023)

A két kép közötti illesztés rektifikálással történik, ahol az *epipoláris geometriai* számításokkal egy közös koordinátarendszerbe illeszthetők az eltérő szögben keletkezett képek, így a komplexitás jelentősen csökkenthető (13. ábra). A számításokat végző algoritmusokat két kategóriába sorolhatóak. Ezek az *explicit programozás alapú* és a *gépi tanulásra épülő* módszerek. Az előbbi esetében hagyományos algoritmusok kerülnek felhasználásra a lokális, blokkalapú és a globális vizsgálatoknál, hátrányuk a különböző textúrákból adódó pontatlanság. Utóbbinál a mélytanulás térhódításával új megközelítések jelentek meg, melyek az eltérő, komplex textúrákat nagy pontossággal tudják megtanulni, így a lehető legjobb eredményeket adják a vizsgálat végén, azonban nagy számításigénnyel rendelkeznek és nehéz általánosítani őket. (Fan, Guo & Bocus, 2023)



13. ábra: Sztereó látás epipoláris geometriája
(www.researchgate.net, 2024.08.18)

2.3.4. Monokuláris mélységbecslés (Monocular depth estimation)

A mélységtérképek létrehozásának egy speciális változata a *monokuláris mélységbecslés*, amely egyetlen képből képes távolsági adatokat megállapítani a sztereó látáshoz képest (14. ábra). Korábbi megoldásokban hagyományos képosztályozási és szegmentációs algoritmusokat alkalmaztak a módszer kivitelezéséhez, manapság azonban a gépi tanulás elérhetőségével már ezen a területen is a mélytanulásra támaszkodnak a legújabb fejlesztések során, mivel általuk sokkal pontosabb mélységtérkép készíthető a vizsgált képből.

A módszer az autonóm járművekben jelenleg még nem került integrálásra, mivel a több kamerát tartalmazó autonóm rendszerekben a sztereó látás nagyobb biztonságot ad, viszont a **monokuláris mélységbecslési** modellek a folyamatosan bővülő tanítási adatkészletek segítségével gyors ütemben fejlődnek, így a jövőben új aspektusokat nyithat az önvezetés világában is. (Szeliski, 2022)



14. ábra: Monokuláris mélységbecslés (vladlen.info, 2024.08.18)

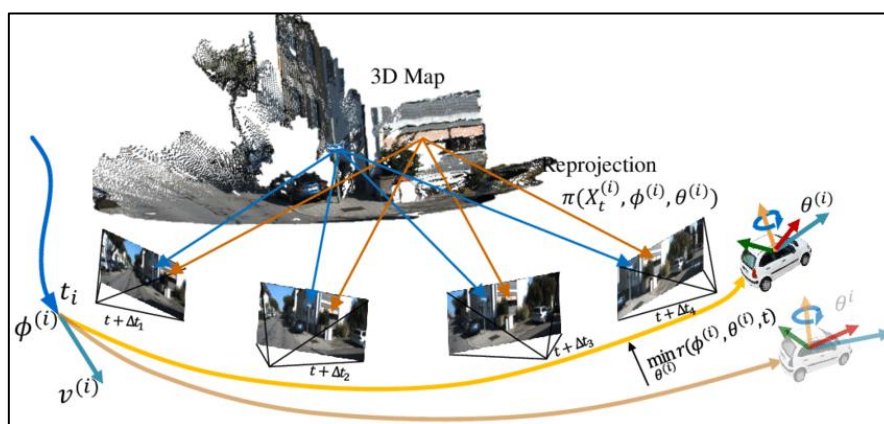
2.3.5. *Mozgásbecslés optikai áramlással (Optical flow)*

A képfeldolgozási technikák között régóta jelen van a mozgásbecslés problémája, melynek legáltalánosabb módszere az úgynevezett **optikai áramlás** számítása. A művelet során az egymást követő képkockákon minden pixelhez kiszámításra kerül a mozgási iránya és a sebessége, melynek köszönhetően követhetővé válik, hogy az egyes pixelek hogyan mozognak a képkockák között. Ez a technika meglehetősen számításigényes, klasszikus módszerekkel az átfedő régiók közötti lokális összegzés és globális minimum keresés algoritmusai kerülnek felhasználásra, azonban a modern algoritmusok mindegyike mélytanulási módszereket alkalmaz, melyek jelentősen javítják a becslési értékek pontosságát. (Szeliski, 2022)

2.3.6. *Vizuális odometria (Visual odometry)*

A **vizuális odometria (VO)** egy, az **optical flow**-ra támaszkodó technika, amely során egy mozgó eszköz, jármű, robot pozíciójának és mozgásának becslése végezhető el a felszerelt egy vagy több kamera képei alapján. Az elnevezés hasonlít a kerék odometriához, ahol egy jármű mozgását a kerekre szerelt enkóder adataiból lehet becsülni, viszont **VO** esetében a pozíciót a képkockákon látható mozgásból származó változások elemzésével lehet meghatározni (**15. ábra**). Hatékony működéséhez szükséges a megfelelő megvilágítás, statikus jelenet elegendő textúrával, valamint az egymást követő képkockák közötti átfedés a mozgás következetes becsléséhez. (Scaramuzza & Fraundorfer, 2011)

Előnyei közé tartozik, hogy a kerék odometriában pontatlanságot okozó kerék megcsúszás a **VO** működését nem befolyásolja. Tanulmányok szerint ez a technika pontosabb útvonalbecslést ad mindössze 0,1-2%-os relatív pozícióhibával, melynek köszönhetően előszeretettel alkalmazzák más rendszerek kiegészítésére, mint a **GPS**, az **IMU** rendszer, a **LiDAR** érzékelők és a **SLAM** technikák támogatásához. A módszer elsősorban sztereó kamerarendszerekre lett fejlesztve, azonban monokuláris felállásban is megállja a helyét. Számos előnye mellett azonban figyelembe kell venni az időben egyre növekvő hiba halmozódását, mivel az aktuális pozíciót minden esetben az előző állapotból határozza meg, így a kisebb pontatlanságok gyorsan nagy mértéket tudnak ölteni, így más érzékelő rendszerek adatai alapján korrekciót kell végezni a kimeneti értékein. (Scaramuzza & Fraundorfer, 2011)



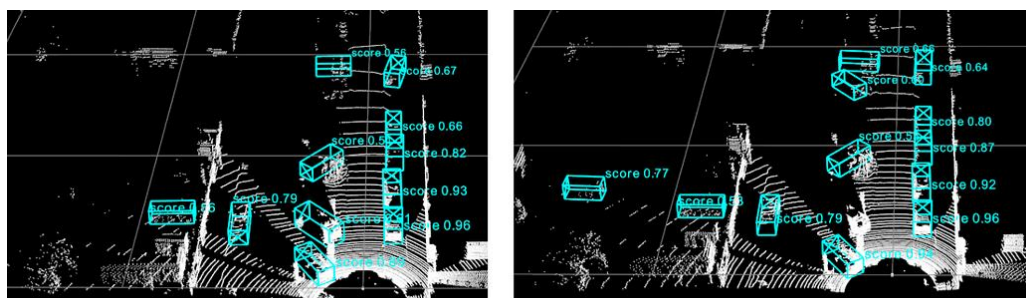
15. ábra: Vizuális odometria lépéseinek szemléltetése
(www.researchgate.net, 2024.08.18)

2.3.7. Objektumkövetés (Object Tracking)

Az objektumkövetés fontos szerepet foglal el az autonóm járművek, a robotika és az intelligens rendszerek feladatai között, hiszen erősen hozzájárul a biztonságos navigációhoz, pályatervezéshez, ütközések elkerüléséhez. A nyomon követés történhet RGB kameraképeken és háromdimenziós pontfelhőben is. A kameraképek esetében a cél az, hogy az első képkockán detektált objektum mozgását nyomon kövessük a további képkockákon. Ennek két elterjedt módszere létezik, melyek a **DBT** és **DFT** követési eljárások. A **detekció-alapú követés (DBT)** során előzetesen betanított detektorokat alkalmaznak az azonosításra. Ez a módszer az autonóm járművek esetében népszerűbb az automatikus működésüknek köszönhetően. A **detekciófüggetlen módszereknél (DFT)** manuális beállítás szükséges. (Murphey, Kolmanovsky & Watta, 2023)

A radarok és **LiDAR** érzékelők által előállított háromdimenziós pontfelhőkben is megvalósítható az egyes objektumok követése (**16. ábra**). Ezek három lépésben képesek nyomon követni egy objektum mozgását. Elsőként detektálni kell a vizsgált objektumot, ezt követően egy **Kalman-szűrő** előrejelzést készít annak jövőbeli helyzetéről. Harmadik lépésben

a pontfelhő időnkénti újra azonosítása történik, amely lehetővé teszi az objektum helyes nyomon követését abban az esetben is, ha az takarásba kerülne. (Murphey, Kolmanovsky & Watta, 2023)

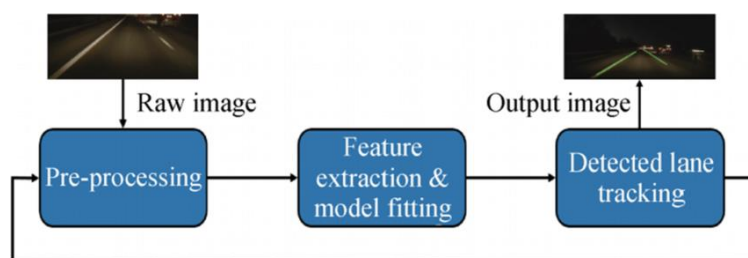


Detected Car Objects on frame t Detected Car Objects on frame $t + 1$
16. ábra: Objektumkövetés 3D pontfelhőben (www.kitware.com, 2024.08.18)

2.3.8. Vonaldetektálás és követés (Lane Detection and Tracking)

Az önvezető autók irányításában nélkülözhetetlen az útburkolati jelek észlelése, azon belül is a sávjelző vonalak detektálása az egyik legfőbb feladat. Ezeknek köszönhetően tudnak működni a sávtartó és sávelhagyáskor figyelmeztető rendszerek, melyek az adaptív vezetéssegítő rendszerek (ADAS) szerves részét képezik évek óta, azonban értékelésük az autonómítás növekedésével minden szinten szigorúbbá vált. Mivel az út és sávérzékelés elsősorban az emberi látásra épülnek, más rendszerek, mint a **GPS** és **LiDAR** csak kiegészítő funkciókat láthatnak el, így a sávérzékelésnek nagyon magas szintű képességekkel kell rendelkeznie a biztonságos navigálás érdekében. (Sapkal et al, 2023)

A sávjelző vonalak detektálásakor sok esetben a hagyományos képfeldolgozási technikákra támaszkodnak, mint a kamera kalibrációs korrekciója, szintér átalakítás, zajeltávolítás, elmosódás csökkentése, inverz perspektíva leképezés, valamint az érdekes területek kiválasztása (17. ábra). További lépésekben történik a sávjellemzők kinyerése éldetektálással, akár **Canny detektorral**, vagy **Hough transzformáció** segítségével. Nem ritka azonban, hogy mélytanulási módszerekre alapozzák az előbbi feladatokat, amely során a konvolúciós neurális hálózatokkal (CNN) részletes szegmentációt végeznek az adott képen, majd a szegmentált képen detektálják az útburkolati jeleket, sávelfestéseket. Előnyük, hogy könnyedén betanulják a kívánt funkciókat és kellően pontosabb eredményt érhetünk el a használatukkal. Ilyen gyakran alkalmazott modellek a **SegNet** és az **UNet** is, azonban egyes megoldásokban a **Generative Adversarial Network**, rövidítve **GAN** modellekkel is találkozhatunk, melyek két egymás ellen dolgozó neurális hálózattal egyfajta belső versenyhelyzetet teremtve képesek fejleszteni magukat. (Sapkal et al, 2023)



17. ábra: A sávérzékelő rendszerek működésének alapvető lépései
(www.researchgate.net, 2024.08.18)

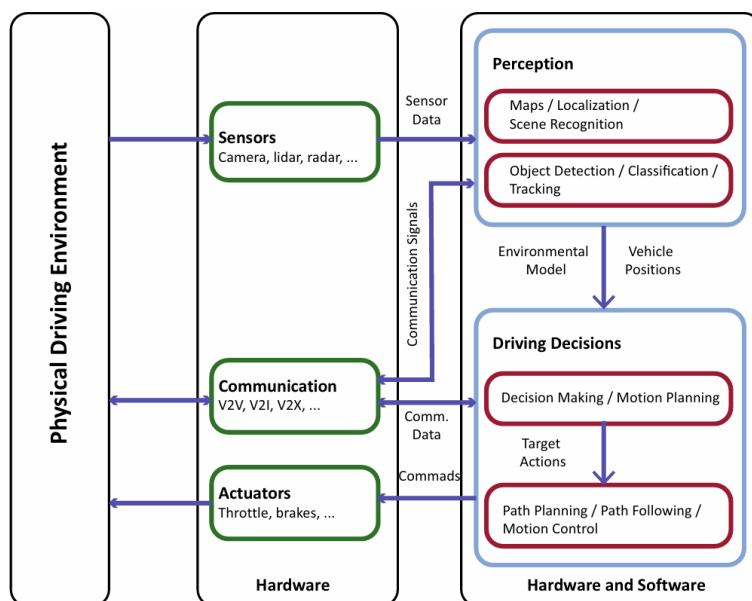
2.4. Mélytanulási módszerek az autonóm járművek vezérlésében

2.4.1. Mélytanulás és az autonóm járművek kapcsolata

A mesterséges intelligencia fejlődése a közelmúltban jelentős mérföldköveket lépett át, így nem meglepő, hogy az önvezető autók vezérlésébe is igen hamar megjelentek a különböző gépi tanulást alkalmazó szoftverek, melyek a bonyolult környezeti és közlekedési szituációkban a korábbi szabályalapú rendszerekhez képest önállóan képesek tanulni és adaptálódni. (Dong et al, 2023)

Az autonóm járművek fejlődésében az önállóság és a könnyű adaptálódás miatt a mélytanulási módszerek nélkülözhetetlenné váltak, különösen a neurális hálózatok, melyek az egyre növekvő számítási kapacitás és az egyre több elérhető nyilvános adatkészleteknek köszönhetően képesek összetett mintázatokat felismerni és összetett cselekvéseket végezni, melyek az emberi vezetés szintjét közelítik meg. (Dong et al, 2023)

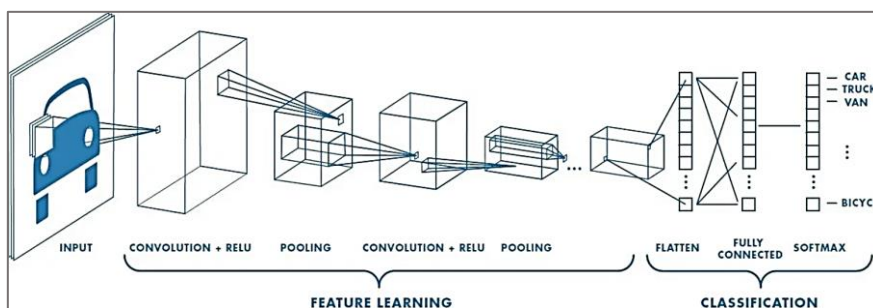
Ezen módszerek beültetése az autonóm vezérlési feladatokba több szinten is lehetséges. Alapvetően a következő fő lépésekből tevődik össze egy önvezetési rendszer: **érzékelés és észlelés**, **döntéshozatal és útvonaltervezés**, valamint a **beavatkozás, cselekvés**. Neurális hálózatokkal legfőképpen az első kettő rétegben találkozhatunk (18. ábra). (Murphey, Kolmanovsky & Watta, 2023)



18. ábra: Autonóm jármű vezérlési lépései (link.springer.com, 2024.08.10)

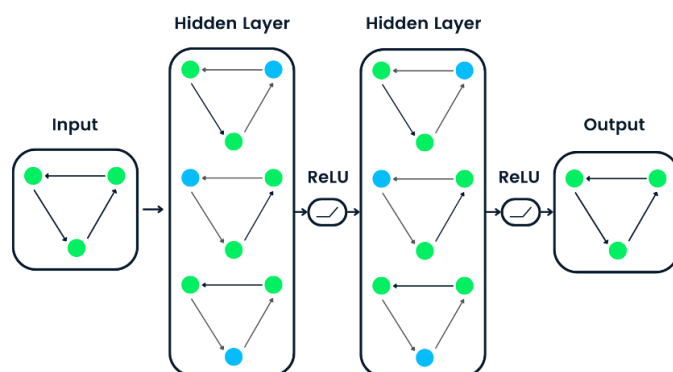
2.4.2. Érzékelés és észlelés mélytanulási módszerekkel

Az érzékelőktől származó környezeti információk képek elemzésében a **konvolúciós neurális hálózatok (CNN-ek)** nagy hatékonysággal képesek részt venni, mivel rácsszerű adatfeldolgozásuk révén magas szintű jellemzőket tudnak kinyerni konvolúciók révén képekből, hangokból és videókból egyaránt. Ezek a modellek helyi kapcsolatokkal, megosztott súlyokkal, valamint rétegmélységgel működnek, így különösen hatékonyak a képfeldolgozás területén, továbbá kevesebb paramétert használnak, így a betanításuk egyszerűbb és gyorsabb folyamat (19. ábra). A közlekedés során a **CNN-ek** magas pontossággal felismerik a közlekedési lámpákat, járműveket, gyalogosokat, akadályokat. Ebbe az osztályba sorolhatóak a korábban említett **UNet** és **SegNet** modellek is, melyek megtalálhatók a nagyobb autógyártók önvezető szoftvereiben is. (Dong et al, 2023)



19. ábra: CNN működési ábrája (saturncloud.io, 2024.08.19)

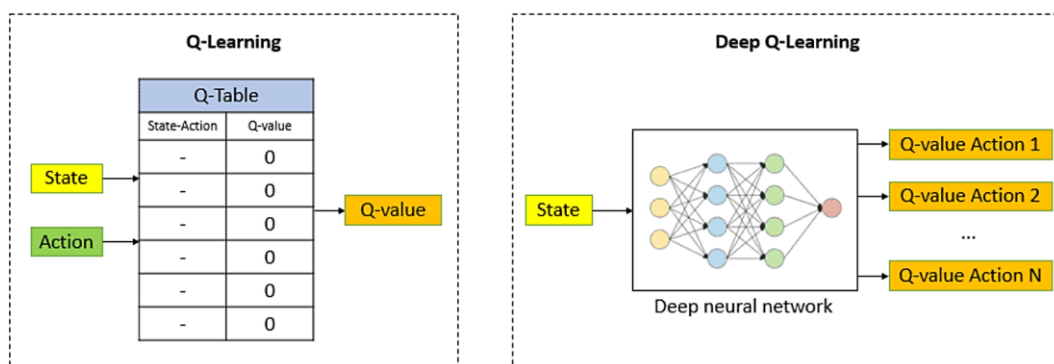
A **CNN-ek** mellett a vezérlés ezen szintjén megtalálhatók a **gráf neurális hálózatok**, a **GNN-ek** is, melyek a közlekedési környezet összetett struktúrájának és kapcsolatrendszerének modellezésére kiválóan alkalmasak, így az autonóm rendszerekben képesek elősegíteni a dinamikus helyzetek értelmezését és a megfelelő döntéshozatalt is. A nevéből adódóan felépítésük csomópontokból és az azokat összekötő élekből tevődik össze, melyek a csomópontok közötti kapcsolatot írják le (20. ábra). Ez alapján mondható rájuk, hogy nem euklédieszi adatstruktúrával rendelkeznek. Működésük során minden csomópont üzenetet küld a szomszédos pontnak a saját és szomszédjai állapotáról. Ezt követően a kapott információk alapján minden csomópont frissíti állapotát, majd ez a két lépés adott számú iteráción keresztül ismétlődik. Végeredményben a csomópontok állapota fokozatosan a helyes és stabil érték felé fog konvergálni. Különböző változatai a konvolúciós műveleteket alkalmazó **Graph Convolutional Network (GCN)**, a **Graph Attention Network (GAT)**, amely működésekor a fontosabb csomópontokat kiemeli, valamint a **Graph Recurrent Network (GRN)**, amely a grafikonok időbeli változásainak modellezésére rekurzív hálózatokat alkalmaz. (Zhou et al, 2021)



20. ábra: GNN architektúrája (www.datacamp.com, 2024.08.19)

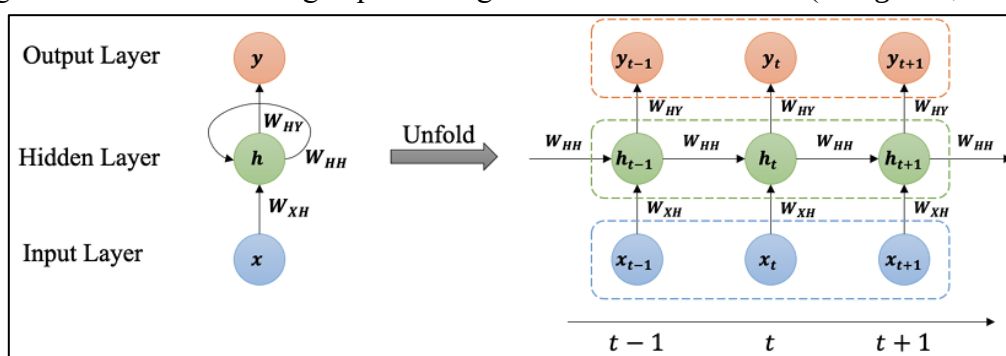
2.4.3. Döntéshozatal és útvonaltervezés mélytanulással

A döntéshozatalban nagy segítséget nyújtanak a **Deep Q-Network (DQN) modellek**, melyek a megerősítéses tanulást alkalmazó módszerek tipikus változatai. Ezen a területen az algoritmusok jutalomrendszer alapján keresik a lehető legjobb döntési stratégiát, így a modell a környezettel való interakciói során tanulja meg azokat a készségeket, melyekkel a teljesítményét javítani tudja. Működése során a **Q-learning**-ben használt Q táblát kombinálja neurális hálózattal. A tanító hálózat az élménytárban tárolt korábbi tapasztalatok alapján, vagyis az állapot-akció párosok és azok jósági értékei szerint tanítja be a modellt, majd a célhálózat ellenőrzést végez az eredményen és meghatározza a jutalmat. Nagy előnye a Q-learning-hez képest, hogy az állapotok és akciók száma jelentős növekedésével sem fog exponenciális memórianövekedés fellépni. A módszer azonban kihívást is tartalmaz, hiszen első lépésben a kutató ágensnek fel kell fedezni a környezetet, hogy új, potenciális stratégiákat tudjon megtalálni, viszont egyidejűleg a meglévő tudása alapján a leghatékosabb jutalom megszerzésére kell törekednie. A **DQN** modellek tulajdonságaik alapján hasznosak a dinamikus környezetekben, ahol a jármű vezérlésének adaptív módon döntéseket kell hoznia, mint a sávváltási, vagy az előzési szituációkban (21. ábra). (Dong et al, 2023)



21. ábra: Q-learning és a DQN módszer összehasonlítása (www.baeldung.com, 2024.08.19)

Az érzékelőktől származó környezeti értékek vizsgálatánál felmerülhet az időbeni folyamatok kezelése, mint a jövőbeni állapotok előrejelzése a kamerák képei alapján a múltbeli képkockák figyelembevételével. Erre nyújtanak megoldást az **ismétlődő neurális hálózatok (RNN)**, melyek visszacsatolásaik révén az előző rétegekbe való információáramlással lehetővé teszik a döntéshozatalnál, útvonaltervezésnél azt, hogy a vezérlés a korábbi eseményeket is figyelembe vegye (22. ábra). Alapvetően három komponensből épülnek fel, ezek a bemeneti egység, a memóriaként funkcionáló rejtett egység és a kimeneti egység, melyek az idő függvényében módosulnak. Tanításuk során kihívást jelent a hosszú távú függőségek megtanulása, mivel a gradiensrobbanás és gradiensvesztés problémái gyakran megjelennek, így hosszútávú memória nem alakul ki. Ezek megoldására fejlesztették ki a **Long Short-Term Memory (LSTM)** és a **Gated Recurrent Unit (GRU)** hálózatokat, melyek úgynevezett „cellák” segítségével már hosszabb ideig képesek megőrizni az információkat. (Dong et al, 2023)



22. ábra: RNN alapvető felépítése és a kihajtogatott változata (DONG et al, 2023)

2.4.4. Kihívások és jövőbeni fejlődési irányok

A jelenlegi trendeket figyelembe véve az autonóm járművek fejlődése területén jól látható a korábban elmondottak alapján, hogy a gépi tanulást alkalmazó algoritmusok a vezérlési szinteken belül szinte mindenhol integrálódtak. Ennek oka a számos ingyenesen elérhető szimulációs környezet és magas minőségű adatkészlet, melyen a kutatók könnyedén tudják javítani és finom hangolni a különböző módszereket, beállításokat, melyek így éles környezetbe is bevezethetők lesznek. Mindemellett a robusztusságra és a biztonságra különösen nagy hangsúlyt kell fektetni ezekben a modern szoftveres technológiákban. (Murphey, Kolmanovsky & Watta, 2023)

A tanulási folyamat során fontos, hogy a modellek/ágensek a lehető legváltozatosabb adatokon tudjanak betanulni, hiszen valós körülmények között változatos környezetben kell majd helytállniuk. Emellett fontos a kellő mennyiségű és minőségű adat, amelyeket pontosan kell felcímkézni a hatékony betanulás érdekében. A megerősítő tanulásnál (RL) minden esetben az exploráció (új akciók kipróbálása) és az exploitáció (a már ismert, jól teljesítő akciók

alkalmazása) közötti megfelelő egyensúly megtartására kell törekedni. Az ágensnek ismeretlen helyzetekben kíváncsinak kell lennie, hiszen csak így fog tudni tanulni, viszont az nem megengedhető, hogy ezen fázisokban veszélyes döntéseket hozzon, amelyek a közlekedés során a kezdeti felfedezés szakaszában akár súlyos balesethez vezethetnek. Ezek a hibák veszélyességük mellett jelentősen lassítják a betanulás folyamatát is. (Murphey, Kolmanovsky & Watta, 2023)

A problémák megoldására a legtöbb ilyen rendszerben megtalálható egy biztonsági ellenőrzést végző lépcső, amely a tanulás során rövidtávú értékeléseket végez a modell kimenetként adott akciójának kockázatosságáról. Ezt egy szabályrendszerrel összevetve gyorsan feltárul a tévedés és az ellenőrző egység a biztonságna megfelelő korrekciót tesz a tanítás alatt. Ezek segítségével nem csak a tanítási folyamat tehető gyorsabbá és biztonságosabbá, hanem lehetőség nyílik arra is, hogy egy autonóm jármű folyamatosan és hatékonyan tudjon adaptálódni az új kihívásokhoz. (Murphey, Kolmanovsky & Watta, 2023)

3. VALÓS IDEJŰ KÉPFELDOLGOZÁSSAL TÁMOGATOTT ÖNVEZETŐ RC AUTÓ

Ebben a fejezetben ismertetem az RC modellautó tervezéséhez és megvalósításához kapcsolódó előzetes kritériumokat és követelményeket, melyek fontos szerepet játszottak a projekt irányvonalának kialakításában. Bemutatom az időtervet, amely megalapozta a projekt kivitelezésének ütemezését, így elősegítve a fejlesztés szervezett és következetes lebonyolítását.

Továbbá bemutatam a projekt során elkészített modellautó tervezési folyamatát, lépéseit, melyben kitérek az egyes hardverelemek kiválasztására és azok autonóm rendszerbe való integrálására.

3.1. Kritériumok meghatározása

A fejlesztési projekt első lépéseként rögzítettem az eszközzel szemben elvárt fő kritériumokat, tulajdonságokat. Ezek szem előtt tartásával megfelelő határok mentén tudtam a rendszer tervezését és kivitelezését elvégezni.

3.1.1. Az önvezető RC autó kritériumai:

- 1:10 méretarányú autóváz alkalmazása.
- Stabil mechanikai felépítéssel rendelkezzen.
- Közepes teljesítményű DC és szervó motor, nem cél a nagy sebesség.
- PWM jellel vezérelhető elektromos sebességszabályozó (ESC).
- A központi egység egy egylapkás számítógép (Single-Board Computer) legyen (NVIDIA Jetson vagy Raspberry Pi 5).
- A motorok vezérlését egy külső PWM vezérlő áramkör végezze a központi egység utasítására.
- Rendelkezzen Wi-Fi kapcsolattal.
- Legyen lehetőség távoli hozzáféréssel „Headless” üzemmódban konfigurálni a központi egységet.
- Legyen legalább egy fizikai bemenet (nyomógomb) és néhány státusz LED, melyek a programok indítását és az egyes állapotok visszajelzését teszik lehetővé.
- A képeket egy minimum 8 Mp felbontású kamera készítse.
- A szoftverek több szálon fussanak a valós idejű feldolgozás érdekében.
- Az autó legyen képes önvezetést megvalósítani teljesen önállóan egy fizikailag kijelölt pályavonal mentén.

- Az autó egy betanított CNN modell segítségével a kamera legyen képek gyors mintavételezésével becslést adni a pályán tartáshoz szükséges kanyarodási szögről
- Az autó legyen alkalmas eltérő környezetben kijelölt ismeretlen pályán önállóan navigálni.

3.2. Időterv és célkitűzések

Az alábbi táblázatban látható az előzetesen felállított ütemterv hónapokra bontva a tervezéstől kiindulóan a kivitelezésig és a tesztelésig (**23. ábra**). A zöld mezők jelölik az eredeti időtervet, a narancssárga mezők szemléltetik a projekt azon fázisait, amelyekkel az ütemtervtől eltérve korábban vagy később is foglalkoztam. Az tervezettel kapcsolatban elmondhatom, hogy a kitűzött időintervallumokat az eszköz fejlesztésének alapjainál megfelelően sikerült tartanom, a szoftveres rész kivitelezésénél és a kamera integrálásánál azonban több eltérést is be kellett iktatnom a sikeres végkimenetel érdekében.

3.3. Hardver tervezése és kivitelezése

IDŐTERV ÉS CÉLKITŰZÉSEK										
2024		02	03	04	05	06	07	08	09	10
HARDVER	Mechanikai tervezés									
	Elektronikai tervezés									
	Hardver összeállítása									
SZOFTVER	Kormány szervó és sebességszabályzó vezérlése szoftveresen									
	Kamerekép feldolgozása									
	Tanító adatok begyűjtése és feldolgozása									
	Neurális háló betanítása									
	Betanított modell alkalmazása az autón									
TESZTELEÉS	Tesztelés különböző környezetben/finomhangolás									
DOKUMENTÁLÁS										

23. ábra: A fejlesztési projekt időterve és célkitűzései

A hardver kialakításánál fontosnak tartottam, hogy az autót működtető motorok és szabályzók rendszere, valamint a számításokat végző egy lapkás számítógép és annak érzékelői különálló módon is kezelhetők és üzemképesek legyenek, így fenntartva az autóváz, vagy a vezérlő egység egyszerű cserélhetőségének lehetőségét. Ebből kiindulva a központi vezérlőegység kiegészítő elemként került fel az autóra, amely segítségével a hagyományos autórendszer önvezetési funkcióval bővül ki.

3.3.1. Autóváz és a működtető elektronika kiválasztása

Az eszköz alapjául egy kereskedelmi forgalomban kapható 1:10-es méretarányú, utcai RC modellautó vázat választottam (**24. ábra**). Ez a modell alumínium alvázal, olajnyomású lengéscsillapítókkal, öszkerékmeghajtással rendelkezik, amelyek együttesen biztosítják a szükséges strapabírást a további elemek felszereléséhez. Ezek mellett a futómű számos

finomhangolási lehetősége további tartalékokat kínál a menettulajdonságok személyre szabásában.



24. ábra: "Reely FS53113CEI" típusú modellautó váz
(www.inshop.hu, 2024.08.23)

Az autó meghajtásával kapcsolatban nem volt cél a nagy sebesség elérése, így erre a célra egy közepes teljesítményű 7,2V-os névleges feszültségű, **szénkefés DC motort** választottam, ez a **25. ábrán** látható. Kis teljesítményfelvételének köszönhetően biztosítja az akkumulátor hosszabb üzemidejét, miközben elegendő nyomatékot ad az autó megfelelő mozgatásához. A motor egy fogaskerék-áttételen keresztül csatlakozik a kardántengelyhez, amely az első és hátsó differenciálműveken keresztül meghajtja a kerekeket, így mozgásba lendítve az autót.



25. ábra: "GPX Extreme SWIFT 550 7,2 V" típusú
szénkefés motor (jakomodell.hu, 2024.08.23)

A kormánymű mozgatásához egy **PowerHD LW-13MG** típusú (**26. ábra**), RC modellekhez használható **szervó motort** választottam, amely fém fogaskerekeinek és golyócsapágyazásának köszönhetően egyenletes, finom mozgásra képes, ütközésekkel szemben strapabíró, valamint akár 13kg/cm-es nyomatékot is ki tud fejteni. Elfordulási szögtartománya 160 ± 10 fok, így a kormánymű mozgástartományához viszonyítva megfelelőnek bizonyul a feladatra. Az üzemi feszültsége 4,8V és 6,6 V között állítható a kívánt nyomaték függvényében. A vezérlése PWM jelekkel történik, melyet általában a sebességszabályzó állít elő, azonban ebben a projektben a kiegészítő elektronika fogja végezni a vezérlését. Ezt a későbbiekben részletezem.



26. ábra: "Power HD LW-13MG" típusú kormánysszervó
(jakomodell.hu, 2023.08.23)

A hajtómotor szabályzására szükséges egy **elektronikus sebességszabályozó** is, másik nevén **ESC (*Electric Speed Control*)** használata, amely a bemenetére kapott vezérlőjeleket képes átalakítani a motor fordulatszámának megfelelő feszültségszintre és mozgásiránynak megfelelő áramirányra, így a fordulatszám és a menetirány szabályozhatóvá válik a bemeneti PWM jel függvényében. Erre a célra egy szénkefés DC motorokhoz tervezett elektronikus sebességszabályzót választottam (**27. ábra**), amely 4,8V-12,6V üzemi feszültségtartománnyal rendelkezik, valamint 60A-es folyamatos áramot is tud biztosítani a motor számára, amennyiben ilyen teljesítményű motort párosítanak mellé. Emellett tartalmaz egy 5V/2A -es kapcsolóüzemű tápegységet is (**BEC**), amely jelen esetben a kiegészítő elektronika PWM generátor moduljának tápellátását fogja ellátni.



27. ábra: "SURPASS ROCKET ESC 1/10 RC 60A 12V" típusú szabályzó (jakomodell.hu, 2024.08.23)

A motorok és a szabályzó tápellátásáról egy 7,4 V-os 16,28 Wh-s **LiPo akkumulátor** gondoskodik (**28. ábra**), amely közvetlenül az ESC-hez van csatlakoztatva. Az akkumulátoron feltüntetett jelzés értelmében 2200mAh-s kapacitással rendelkezik, a bank 2 cella soros kapcsolatából épül fel, melyek akár 30 A-es folyamatos merítéssel is terhelhetőek. Ezekkel a paraméterekkel elegendő üzemidő érhető el az autó számára, amely navigálás közben csak alacsony teljesítményen fog üzemelni.



28. ábra: "CNHL 2200mAh 7,4V 30C" típusú LiPo akkumulátor (chinahobbyline.com, 2024.08.23)

Az eddig ismertetett hardverelemek a modellautó közvetlen vezérlési rendszerét képezik. Ennek összeállított állapotát szemlélteti a **29. ábra**, ahol az elemek a korábban említettek alapján önálló egységet alkotnak a következőkben bemutatásra kerülő központi vezérlőrendszer mellett.



29. ábra: RC autó váz összeszerelt vezérlőegységgel

3.3.2. A központi vezérlő egységet képező hardver elemei

Az önvezetéshez szükséges fedélzeti számítógép esetében a kritériumoknak megfelelően egy **NVIDIA Jetson Nano** típusú *egy lapkás számítógép* platformot választottam (**30. ábra**). Ezt az eszközt kifejezetten mesterséges intelligenciára, valamint gépi tanulásra épülő alkalmazások fejlesztésére tervezték, így ideális a projekthez szükséges számítógépes látás és döntéshozás megvalósítására. A technikai paramétereket az **1. melléklet** tartalmazza.



30. ábra: NVIDIA Jetson Nano B01 Development Kit
(www.cytron.tech.vn, 2024.08.23)

Futtatható operációs rendszerek tekintetében a Jetson Nano az ARM architektúra miatt a Linux alapú rendszereket támogatja. Ilyen az általam is használt NVIDIA fejlesztésű **JetPack SDK** is, amely egy ingyenesen letölthető Ubuntu alapú operációs rendszer. Használatával kapunk egy teljes értékű Linux környezetet, amely támogatja a parancssori eszközöket, a fejlesztői eszközöket, mint a Python, C++, TensorFlow, PyTorch, Docker, valamint a grafikus felhasználói felületek különböző változatait. Ezen kívül távoli kapcsolaton keresztül is elérhetővé teszi a rendszert, így Wi-Fi adaptert alkalmazva vezeték nélküli kapcsolattal úgynevezett „Headless” üzemmódban is programozható és kezelhető az eszköz, amely fontos része a rugalmas és hatékony fejlesztésnek.

Az ESC és a kormány szervó vezérléséhez a korábban elmondottak alapján PWM jelekre van szükség. A Jetson Nano maga is képes ezeket előállítani, melyeket a kitüntetett GPIO pin-eken keresztül el is érhetünk, azonban ez erőforrásigényes és lefoglalja a processzort. Ennek elkerülése érdekében egy **PCA9685 típusú 16 csatornás, 12 bit-es szervó meghajtó** modult

választottam (**31. ábra**), amely önállóan képes a szükséges vezérlő jelek generálására és fenntartására. A modul támogatja az I²C kommunikációt, így a Jetson Nano-n mindösszesen 2 pin lefoglalása szükségeltetik, használatakor pedig a processzornak elegendő egyszer elküldeni a kívánt jel paramétereit a modul számára, amely az utasítás hatására önállóan cselekszik.



31. ábra: "PCA9685" típusú 16 csatornás szervó vezérlő
(www.hestore.hu, 2024.08.24)

Az előzetes követelményeknek megfelelően a központi vezérlő egységen elhelyezésre került egy egyedi tervezésű **indikátor áramkör** is (**32. ábra**), amely 6 db SMD LED-et és egy nyomógombot tartalmaz. A NYÁK segítségével a szoftver állapotai az egyes LED-ek kapcsolgatásával fizikailag is visszajelezhetőek, a nyomógombnak köszönhetően pedig a Jetson Nano kiegészül 1 db fizikai bemenettel, amely a programok indításánál és leállításánál használható. Az áramkör kapcsolási rajza és NYÁK tervei a **3. mellékletben** tekinthetők meg.



32. ábra: Indikátor áramkör

Az önvezetéshez elengedhetetlen a projekt tekintetében a gépi látás megvalósítása, amely általános esetben környezeti érzékelőként egy kamerát igényel. Erre a célra egy **Raspberry Pi Camera Module v2** típusú kamerát választottam (**33. ábra**). Az eszköz egy Sony IMX219-es szenzorral van felszerelve, 8 MP felbontású, maximális képfelbontása 3280x2464 képpont, videófelvételek területén támogatja az 1080p/30 FPS, a 720p/60 FPS és a 480p/60 FPS felbontásokat és képkockasebességeket, emellett fix fókusszal és 73 fokos látószöggel rendelkezik. Kis méretének köszönhetően könnyedén elhelyezhető az autón speciális rögzítés nélkül is. Csatlakozását tekintve kompatibilis a Jetson Nano-n lévő CSI, nagy sebességű kamera csatlakozófelületekkel, így külön átalakító egységet sem igényel.



33. ábra: Raspberry Pi Camera Module v2
(thepihut.com, 2024.08.24)

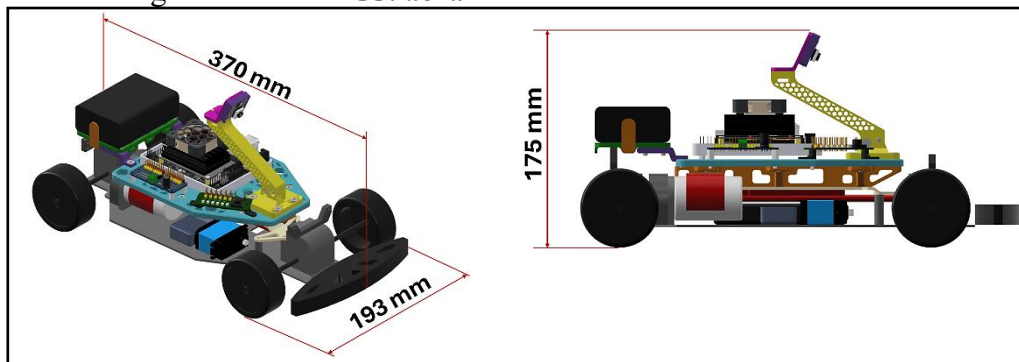
A központi vezérlő egység tápellátása egy különálló, 20000 mAh-s Li-Ion Powerbank-ról történik(34. ábra), így az autó vezérlő egységének esetleges tápellátási ingadozásai, melyet a motorok idéznek elő, nem hatnak ki a Jetson Nano működésére. Az akkumulátor 5V/5A kimeneti feszültséggel és áramerősséggel rendelkezik.



34. ábra: Vipfan F09 20000mAh Powerbank (kulsoaksi.hu , 2024.08.24)

3.3.3. Rögzítő váz tervezése -3D nyomtatás

Az előző alfejezetben ismertetett központi vezérlőegység felszereléséhez szükség volt külön tartóváz tervezésére, amely illeszkedik az autó méreteihez és könnyedén fel-, illetve leszerelhető arról. Ehhez egy több elemből álló vázat terveztem, amely néhány csavar segítségével rögzíthető az autó vázszerkezetén található középső kardántengelyt védő lemezalkatrészre. Ezt követően felszerelhetők a hardverelemek, a **Jetson Nano**, a **PCA9685 PWM vezérlő**, az **indikátor áramkör**, a **kamera** és az **akkumulátor** is. A rögzítővázat 3D nyomtatással készítettem el. Az alkatrészek anyagául **PETG**-t (polietilén-tereftalát glikol) választottam, mivel kiváló ütésállóságot, rugalmasságot és megfelelő merevséget biztosít a szerkezet számára. A hardverelemek elrendezését és a tervezett több elemű tartóvázat, valamint az egész autó befoglaló méreteit a 35. ábra szemlélteti.

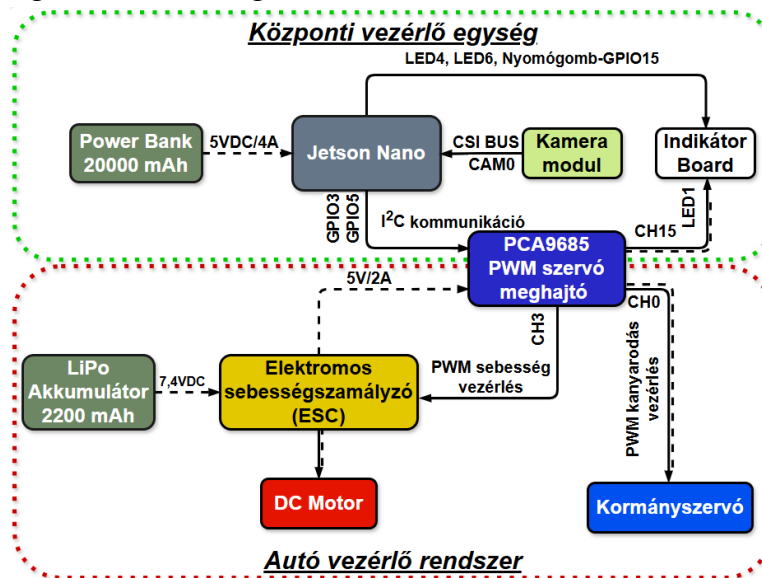


35. ábra: A tervezett tartószerkezet és a befoglaló méretek

3.3.4. Rendszer blokkdiagramja és az elkészült autonóm modellautó

A hardver architektúráját leíró blokkdiagram a 36. ábrán látható. A rendszer részletesebb kapcsolási rajza a 4. mellékletben található. Az egyes elemek közötti kapcsolatot jelző folytonos vonallal rendelkező nyilak a kommunikációs kapcsolatot jelentik, a szaggatott vonalú nyilak pedig a tápellátást jelölik. Az ábrán jól látható az autó vezérlő rendszerének és a központi vezérlőegység különállósága, melyeknek egyetlen közös pontjuk van, amely nem más, mint a **PCA9685** típusú PWM vezérlő áramkör. A két rendszer földpont közösítése is ezen a

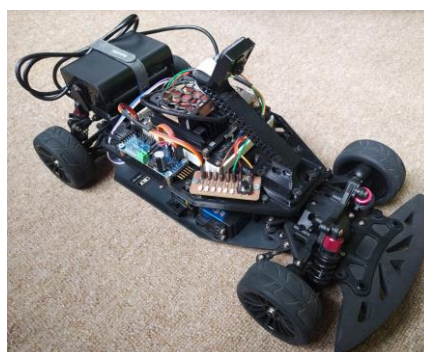
modulon történik meg, így elkerülve az üzem közben kialakuló, eltérő referenciafeszültségekből adódó meghibásodásokat.



36. ábra: Az önvezető modellautó rendszerét leíró blokkdiagram

A teljes rendszer három fő elemének elhelyezkedését nézve a következő kapcsolatok állnak fenn. A Jetson Nano vezérlő utasításokat küld I²C kommunikáción keresztül a PWM vezérlőnek, emellett fogadja a kamera modultól származó képeket, az indikátor panel 3 db LED-jét használja fel visszajelzésre, valamint a 15-ös GPIO pin-en keresztül detektálja fizikai nyomógomb lenyomását. A PWM szervó vezérlő fogadja a Jetson Nano által küldött utasításokat a szükséges jelek előállításához. A 16 csatorna közül 3-at használ fel, melyek a kormány szervóhoz, az ESC sebességszabályzó bemenetéhez és az indikátor panel 6-os számú visszajelző LED-jéhez csatlakoznak. Tápellátását az ESC belső kapcsolóüzemű tápegységétől kapja, saját maga pedig az indikátor panelnak és a szervónak biztosít 5V-os áramforrást. Az **elektromos sebességszabályzó** fogadja a PWM vezérlő által előállított, adott kitöltésű PWM jelet, melynek függvényében szabályozza a DC motor felőli kimenetét, így elérve a kívánt fordulatszámot és sebességet.

A terveknek megfelelően a legyártott vázelemekből, valamint az elektronikai hardverek blokkdiagram szerint történő összekötésével elkészült a projekt alapeszköze, amely alkalmas az irányító szoftverrel való együttműködésre (37. ábra).



37. ábra: Az tervek alapján kivitelezett hardver

4. A FEJLESZTETT SZOFTVER BEMUTATÁSA

A projekthez készített szoftver a kódok futtatási helyei alapján két részre osztható. Az egyik rész az autó központi számításokat végző modulján, a Jetson Nano eszközön futó programokat foglalja magába, míg a másik rész Windows operációs rendszerkörnyezetben működik, ahol a tanító adatok előfeldolgozása és a modell betanítása történik. Ebben a fejezetben az előbbi csoportosítás alapján két alfejezetben ismertetem a fejlesztett programokat a működésüket leíró logikai folyamatábrákon keresztül. Az egyes programok teljes forráskódjai a mellékletekben találhatók, melyekre a leírásuk során számszerűen hivatkozok.

4.1. A központi vezérlő egységen futtatott programok ismertetése

Az autó autonóm működéséhez három Python program egyidejű futása szükséges. A tanító adatok felvétele azonban egy különálló programot igényel, így összességében négy program került fejlesztésre a szoftver ezen csoportjában.

4.1.1. Alap program

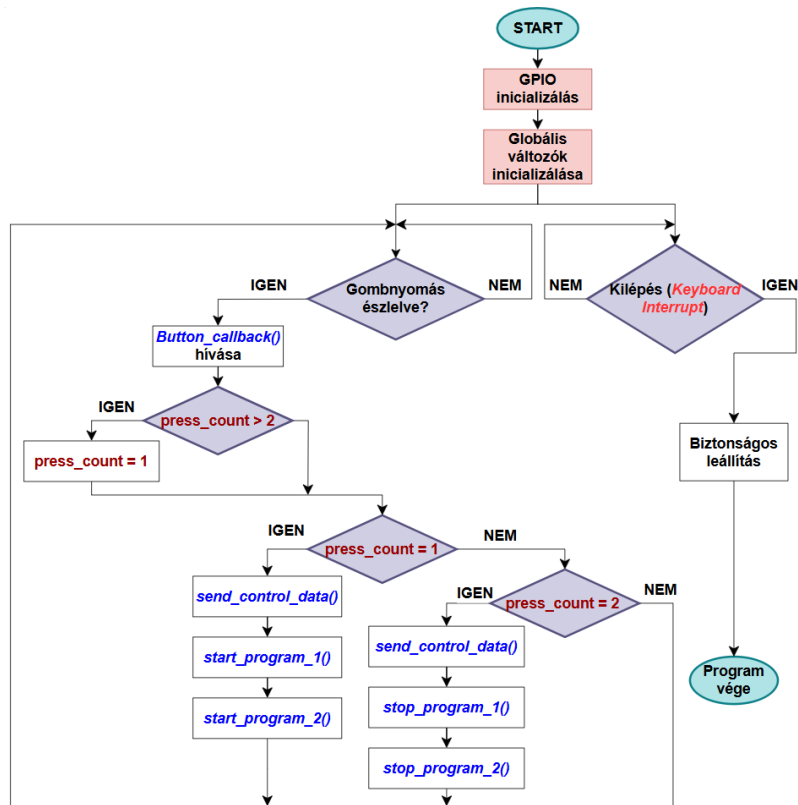
A rendszer szoftverének alapját egy Python 3.6.9-es verzió alatt futtatható program képezi, amely elsősorban az indikátor áramkörön elhelyezett indítógomb állapotát figyeli. Emellett képes elindítani és leállítani két párhuzamosan működő programot. A teljes programkód az **5. mellékletben** tekinthető meg, a működést leíró folyamatábra a **38. ábrán** látható, ami alapján a következő feladatok hajtódnak végre:

A kód indulásakor első lépésben inicializáció történik, ahol a szükséges GPIO pinek és a globális változók kerülnek beállításra. A program futásáról az indikátor áramkörön lévő 1-es LED felkapcsolása ad visszajelzést az indítás pillanatától kezdődően. Ezt követően két egymás mellett futó ág következik, ahol az egyik ciklus a nyomógomb állapotát figyeli, a másik a program leállításának feltételét vizsgálja. A nyomógomb megnyomásakor a 15-ös GPIO pin érzékeli a keletkezett lefutó élt, majd annak hatására meghívásra kerül a **button_callback()** függvény, amely a lenyomás számlálójának értéke alapján különböző eseményeket indít el.

A nyomógomb első megnyomásakor(**press_count = 1**) a **send_control_data()** függvény egy globálisan elérhető bitsor alapú fájlba menti a sebesség és kanyarodási szög alapértelmezett értékeit, így biztosítva, hogy az autó (sebesség és kormányvezérlése) mindig alaphelyzetből induljon. Következő lépésként a **subprocess.Popen** segítségével párhuzamos folyamatként elindul a két további program. Ezek közül az első minden esetben az autót közvetlenül vezérlő program, a második a felhasználás függvényében lehet a tanító adatokat gyűjtő kód, vagy a betanított CNN modellt használó előrejelző program. A két program jelen esetben eltérő python verziókkal kerül elindításra, mivel a Jetson Nano-n elérhető OpenCV szoftvermodul, melyet a tanító adatokat gyűjtő program és az önvezető modellt alkalmazó kód is használ, nem

kompatibilis a PCA9685-ös modult kezelő Adafruit függvénykönyvtár Python támogatottságával.

A nyomógomb másodszori megnyomásakor (***press_count*** = 2) az előbbieken indított két program leállításra kerül. Harmadik lenyomás esetén az egész folyamat kezdődik előlről, mivel a ***press_count*** értéke 1-re vált és újra elindul a két külön szálon futó program. Ezzel a módszerrel a programok indítása, leállítása ciklikusan ismételhető.



38. ábra: "start_button_v2.py" program folyamatábrája

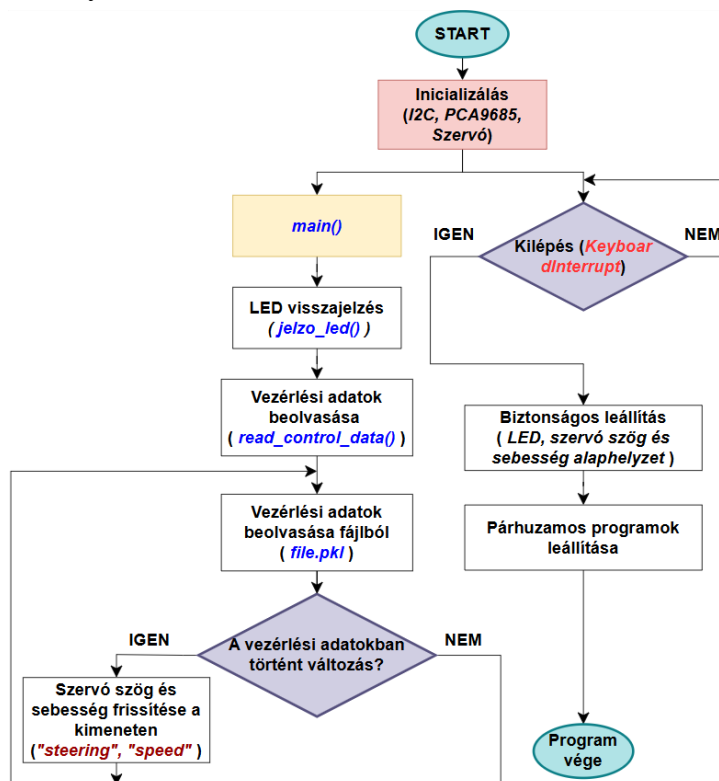
4.1.2. Sebesség és kormányzás vezérlése

Az autó közvetlen vezérlését végző program az elektromos sebességszabályzó és a kormány szervó működtetéséért felelős. A program az Adafruit függvénykönyvtár segítségével kommunikál a PCA9685 PWM modullal, ezért a program csak Python 3.8-as vagy annál magasabb verzióban futtatható. Az induláskor hasonlóan az előző programhoz megtörténik az inicializáció, amely magába foglalja az I²C kommunikáció paramétereinek és a PWM modul szükséges csatornáinak beállítását. A program ezt követően szintén két ágban folytatódik, ahol a baloldali ág gondoskodik a fő feladat végrehajtásáért felelős függvény indításáról, a másik oldal a kilépési feltételek meglétét ellenőrzi.

A ***main()*** függvény meghívásakor elindul a fő programrész. Első lépésben a PWM vezérlő 15-ös csatornáján keresztül az indikátor áramkör visszajelzi a kód futását a ***jelzo_led()*** függvény segítségével. Következő lépésben a ***read_control_data()*** függvény ciklikusan elkezd

beolvasni a programok közötti kommunikációért felelős globális, bit alapú fájlt (**file.pkl**), amely a vezérléshez szükséges sebesség- és kanyarodási szögértékeket tartalmazza. Abban az esetben, ha ezek az adatok frissítésre kerülnek más programok által, akkor a 0. és a 3. PWM-csatorna kimenetei az új értékeknek megfelelően módosulnak, így az autó felveszi a kívánt sebességet és kanyarodási szöget.

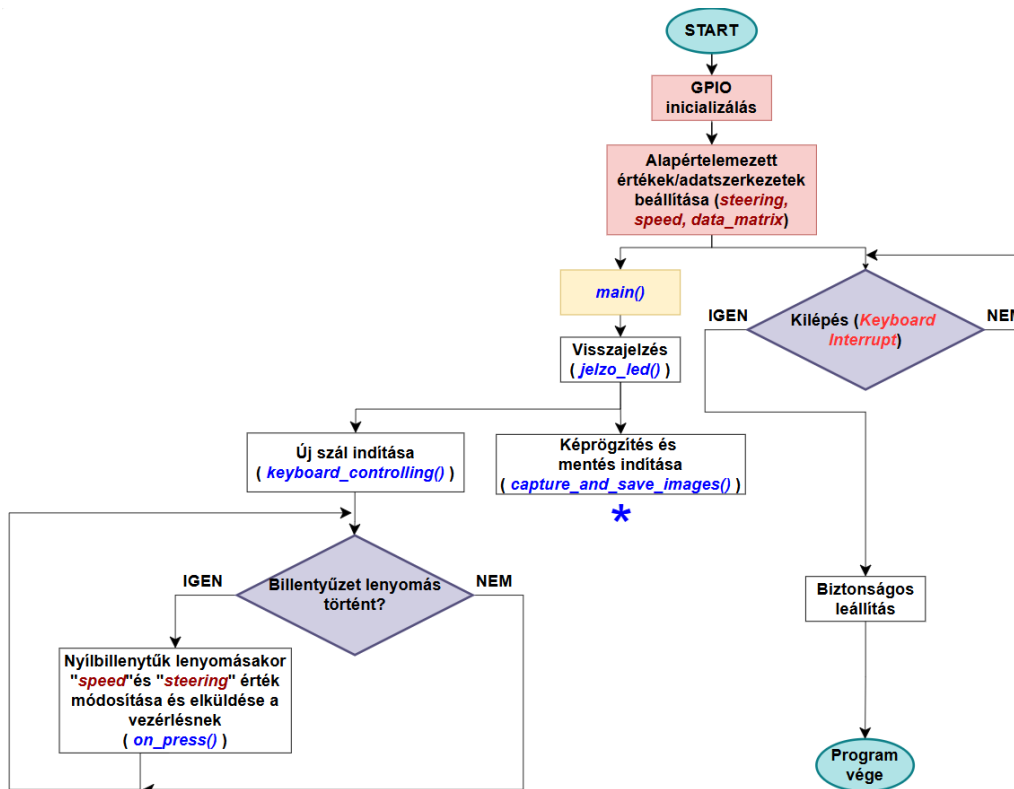
A program leállításakor, melyet vagy a **Keyboard Interrupt**, vagy az alapprogram vált ki a nyomógomb második megnyomásával, a PWM-csatornák kimenetei alapértelmezett értékeket vesznek fel, amely eredményeként az autó megáll és a kormányzó kerekek egyenes irányba állnak. Az autó vezérlését végző program kódja a **6. mellékletben** található, a fent leírt működést szemléltető folyamatábra a **39. ábrán** látható.



39. ábra: "car_control".py folyamatábrája

4.1.3. Tanító adatok gyűjtése

Az autó önvezető funkciójának megvalósításához a neurális hálózatot megfelelő módon be kell tanítani, azonban ehhez tanító adatbázis szükséges. Ennek elkészítéséhez készült a következő program, ami az autó manuális, ember által vezetett üzemmódjában a kamera segítségével képes a padlóra felrajzolt pályavonalról megfelelő mintavételi időintervallummal képeket rögzíteni és menteni, valamint regisztrálni az aktuális pillanatban érvényben lévő sebesség és kanyarodási szög értékeit. A tanító adatok gyűjtését végző program folyamatábrája **40. ábrán** látható.

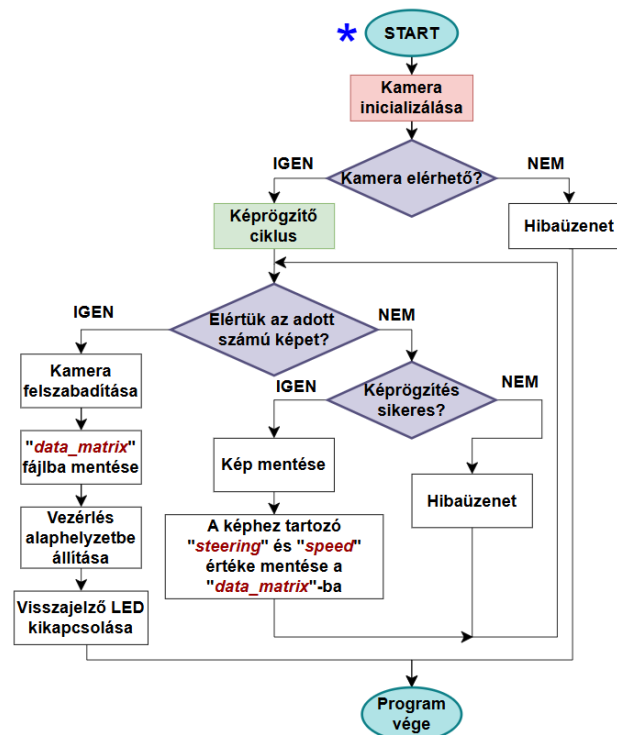


40. ábra: "training_data_collect.py" programot leíró folyamatábra

Az ábra alapján a program a következő lépések alapján működik. Induláskor meghatározásra kerül az indikátor áramkörhöz kapcsolt GPIO pin, a sebesség- és kanyarodási szögértékek az alapértelmezett értékeket veszik fel, valamint létrejön a tanító adatok tárolásához szükséges mátrix(*data_matrix*) is. Az előző programokhoz hasonlóan itt is két ágban folytatódik a kód, ahol baloldalt a fő programrész fut, jobb oldalt a kilépési feltételek ellenőrzése zajlik.

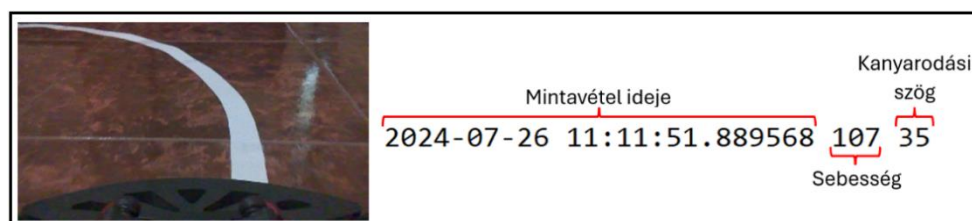
A *main()* függvény hívásával első lépésként a *jelzo_led()* függvény felkapcsol egy LED-et az indikátor áramkörön, így adva visszajelzést a program futásáról. Következő lépésben két párhuzamosan futó folyamatként kezdetét veszi az adatgyűjtés, valamint az autó manuális vezérlését végző számítógépes billentyűzet gombnyomásának folyamatos figyelése. Az autó manuális vezérlése a nyíl billentyűk lenyomásával lehetséges, ahol a jobb-bal nyilak segítségével a kanyarodási szög növelhető, illetve csökkenthető konstans értékkel a kanyarodási szögtartományon belül, a fel-le nyilakkal a sebesség értéke növelhető és csökkenthető a kívánt módon. Minden gombnyomás észlelésénél frissítésre kerülnek a sebesség és szög értékei, melyeket a *send_control_data()* függvény egyből a globálisan elérhető bitsor alapú fájlba ment(*file.pkl*), így elküldve az új vezérlési paramétereket az előző pontban bemutatott autóvezérlő programnak.

Az adatgyűjtés az irányítástól függetlenül, párhuzamosan fut a `capture_and_save_images()` függvény segítségével, melynek folyamatábrája a **41. ábrán** látható. Első lépésben a kamerát kell inicializálni, csak ezt követően lehet lekérdezni annak képét. Abban az esetben, ha a kamera az inicializálás után elérhető és lekérdezhető, elindul a képeket és adatokat rögzítő ciklus, ahol a függvény argumentumában megadott képkockaszámának megfelelően időbélyeggel ellátva mentésre kerülnek a kamera által látott aktuális képek. Ugyan ebben a lépésben a pillanatképekhez tartozó vezérlő értékek bekerülnek a `data_matrix` adatszerkezetbe, melynek tartalma csak a kívánt mintakép elkészítését követően kerül kiírásra egy szöveges fájlba (`data.txt`). A mintavétel befejezését követően a visszajelző LED kikapcsol, valamint az autó megáll.



41. ábra: "capture_and_save_image()" függvény folyamatábrája

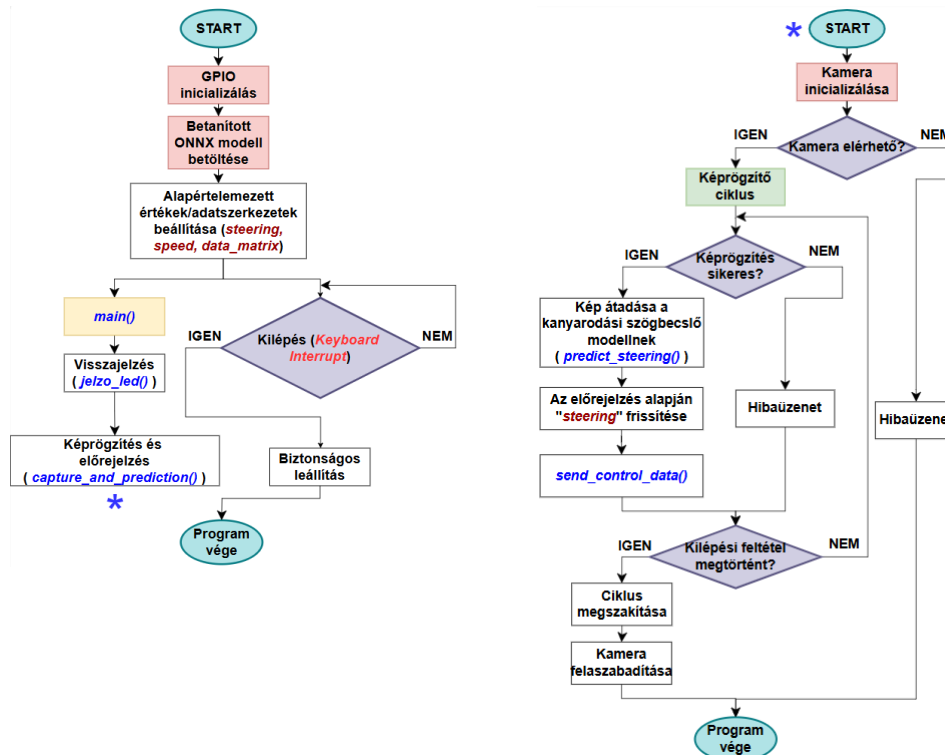
Ezzel a módszerrel a valós vezetési körülményeket megfelelően tükröző adatpárosokat kapunk képek és vezérlési értékek formájában, melyeket feldolgozva és megfelelő struktúrába rendezve előállítható az önvezetést végző modell betanításához szükséges mintaadatbázis (**42. ábra**). A további feldolgozást végző programot egy későbbi pontban ismertetem. Az adatgyűjtő program forráskódja **7. mellékletben** tekinthető meg.



42. ábra: Példa egy nyers adatpárosra – Pillanatkép és a hozzá tartozó vezérlő értékek

4.1.4. Előrejelzést végző program

A önvezetési funkcióért felelős program a tanító adatokat gyűjtő program módosított verziójának feleltethető meg (43. ábra), ahol a manuális irányítást végző billentyűzettel már csak a sebesség vezérelhető, a `capture_and_save_image()` függvény helyét a `capture_and_prediction()` függvény vette át.



43. ábra: "onnx_model_vezerlo.py" működési folyamatábrája

A képrögzítés és előrejelzés során a kamera inicializációja és elérhetőségének vizsgálata után a rögzített kép átadásra kerül a betanított modellt tartalmazó `predict_steering()` függvénynek. A kép átadásakor a függvény előzetes képfeldolgozással (`preprocess_image()`), átméretezéssel, színtérmódosítással, világosítással, adatkonverzióval, dimenzió bővítéssel és normalizálással felkészíti a pillanatképet a gépi tanulási modell számára. A betanított CNN modell ezt követően a képen látható epizód alapján egy becslést, előrejelzést ad az autó pályán tartásához szükséges kanyarodási szögértékről. A kapott érték ezt követően bekerül a vezérlési adatsorba, melyet a `send_control_data()` függvény továbbít az autót közvetlenül vezérlő programnak a korábban ismertetett módon.

A forráskód a 8. mellékletben megtalálható. A kód ugyan úgy az OpenCV-re támaszkodik, mint az adatgyűjtő változata, így szintén csak Python 3.6.9-es verzió alatt futtatható.

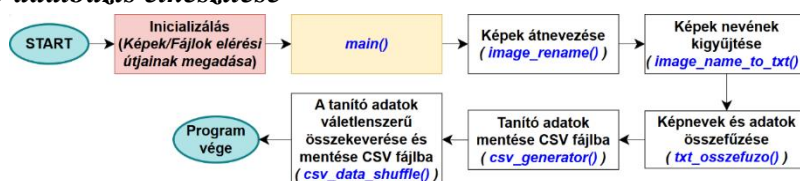
A fenti program csak az egy célértékkel rendelkező CNN modelleket képes kezelni, melynek értelmében a vezérlésnek csak a kanyarodási szögértéket továbbítja. Ahhoz, hogy a

sebesség értékét megadó modelleket is fogadni tudja a kód minimális módosítására van szükség. Az így kapott program logikai működése nem tér el a fentiekben tárgyaltaktól. A módosított verzió kódja a **11. mellékletben** található.

4.2. A Windows környezetben futó kiegészítő programok bemutatása.

A tanító adatbázis előkészítése, valamint az előre jelző CNN modell betanítása az autó rendszerétől független környezetben történik Windows operációs rendszer alatt **PyCharm** fejlesztőkörnyezetben. Ezen lépéseket két program végzi el, melyek működését a következő pontokban ismertetem.

4.2.1. A tanító adatbázis elkészítése



44. ábra: "tanito_adatok_elokeszites.py" program működési folyamatábrája

Az CNN modell betanításához az autó által begyűjtött nyers minta adatokat egy rendezett, strukturált adatbázisba kell elhelyezni. A gépi tanulás területén ezt az eljárást a képek címkézésének (**image labeling**) is hívják. Ennek elkészítését szintén egy Pythonban fejlesztett program fogja segíteni, melynek működését a **44. ábra** mutatja. A forráskód teljes változata a **9. mellékletben** tekinthető meg.

A nyers adatok tekintetében képek és a hozzájuk tartozó vezérlési értékek állnak rendelkezésre az adatbázis elkészítéséhez, melyeket korábban a **42. ábra** szemléltetett. A program folyamatábrája alapján első lépésként a nyers adatok elérési útjai kerülnek beolvasásra. Ez tartalmazza az összes képet magába foglaló mappa elérési útvonalát, valamint az adatokkal felöltött szöveges fájlt, ahol a képek indexe összhangban van a szöveges fájl sorainak számozásával, így biztosítva az adatok párosíthatóságát.

Következő lépésként a **main()** függvény meghívásával az **image_rename()** függvény az összes mintaképet átnevezi, így a nehezebben kezelhető időbélyeget tartalmazó képnevek helyett **image1.jpg**, **image2.jpg**, **image3.jpg**...stb. módon hivatkozhatunk a mintákra. Harmadik lépésként az összes kép nevét kilistázza az **image_name_to_txt()** függvény egy szöveges fájlba, amely szükséges lesz a tanító adatszerkezet kialakításához. Az előkészítés negyedik lépéseként a képek nevét tartalmazó fájl, valamint a tanító adatokat tartalmazó fájl sorai összefűzésre kerülnek a **txt_osszefuzo()** függvény által, így létrejön az adatpárok első változata.

Ötödik lépésként történik a végleges adatpárokat tartalmazó **csv** kiterjesztésű adatsor létrehozása (**csv_generator()**), amely a betanításhoz elengedhetetlen, mivel ennek segítségével

fogja a modell megtanulni, hogy az egyes mintaképekhez milyen kimeneti értéket kell adnia. Az így létrejött **driving_data.csv** fájl struktúráját a **45. ábra** mutatja, ahol az adatok a mintavételi időpont szerint követik egymást.

Az adatok túlzott rendezettsége azonban nem előnyös a CNN modellek betanításakor, mivel azok könnyedén megtanulhatják az adatok egymásutániságából eredő mintázatokat. Ez a tanulás során túlilleszkedést (**overfitting**) eredményez. Ennek megoldására a program elvégz egy hatodik lépést is, ahol a **driving_data.csv** fájl sorait véletlenszerűen összekavarja, majd kimentti a **shuffled_driving_data.csv** fájlba. Ennek segítségével az adatok továbbra is párban maradnak, viszont változatosabban fogják egymást követni, így segítve a modell általánosítási képességét a tanulás során.

driving_data.csv		shuffled_driving_data.csv	
1	image, speed, angle	1	image, speed, angle
2	image1.jpg, 107, 100	2	image753.jpg, 107, 105
3	image2.jpg, 107, 100	3	image2599.jpg, 105, 120
4	image3.jpg, 107, 105	4	image3203.jpg, 105, 125
5	image4.jpg, 107, 105	5	image3116.jpg, 105, 155

45. ábra: Példa a rendezett és véletlenszerűen összekevert és feldolgozott tanító adatokból

4.2.2. CNN betanítása a tanítóadatok segítségével

Az önvezetéshez használatos CNN modell betanítása szintén egy Python programban történik Windows környezetben, melynek működését a **47. ábra** szemlélteti, forráskódja a **10. mellékletben** található.

A kód a következő lépéseket hajtja végre a feladat elvégzéséhez. Az inicializálást követően a **main()** függvény meghívja a **training_data_loading_and_preprocessing()** függvényt, amely első lépésben az előzetesen elkészített tanító adatokat tartalmazó **driving_data.csv** fájlt betölti. Az így elérhető adatokat ezt követően három listába rendezi, melyek tartalmazni fogják a tanító képek elérési útjait(**image_path**), a kormányzási szöveget(**angles**), valamint a sebesség adatokat(**speeds**). Következő lépésben a **load_images()** függvény betölti, majd átméretezi a képeket 64x64 képpontra, tömbbé alakítja és hozzáadja a képek listájához. A tanítás hatékonyságának növelése érdekében a képek képpontjai normalizáláson is keresztül mennek, így azok értékei 0 és 1 közötti tartományra skálázódnak. Utolsó lépésben a rendelkezésre álló adatokat tanító és teszt adatokra osztja, amely segítségével a tanulás során az elkülönített tesztadatokkal folyamatosan ellenőrizhető a modell pillanatnyi teljesítménye.

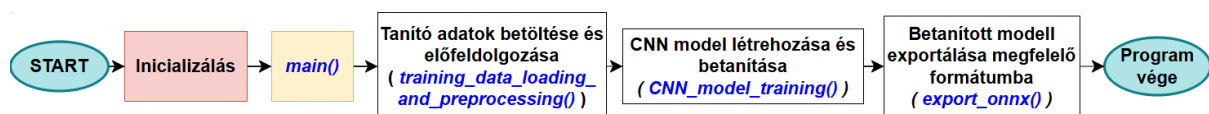
Az előkészítést követően kezdetét veszi a betanítás, melyet a **CNN_model_training()** függvény végez el. Kiindulásként a **CNN_model_create()** létrehoz egy konvolúciós neurális hálózatot. Struktúrája rétegek sorozatából áll, ahol a **konvolúciós rétegek** kis területeket dolgoznak fel, majd ezekből építenek magasabb szintű jellemzőket. A **pooling** rétegek az

adatok dimenziójának csökkentését végzik, a **dropout rétegek** a túltanulás megelőzéséért felelősek. A hálózat végén lévő **teljesen összekapcsolt rétegek** a képjellemzőket összegzik, a végső **kimeneti réteg** pedig megadja a modell által előre jelzett értéket, amely jelen esetben a kanyarodási szöveget jelenti. Az egy kimenettel rendelkező modell felépítését a **46. ábra** mutatja.

RÉTEG TÍPUSA	SZŰRŐK SZÁMA	SZŰRŐ MÉRETE	AKTIVÁCIÓS FÜGGVÉNY	EGYÉB MŰVELETEK	MEGJEGYZÉS
Input réteg	-	64x64x3	-	-	Képek bemenete
Conv2D	32	3x3	ReLU	-	Első konvolúciós réteg
MaxPooling2D	-	2x2	-	-	Első pooling réteg
Dropout	-	-	-	0.1	Csökkenti a képméretet és megőrzi a fontos jellemzőket
Conv2D	64	3x3	ReLU	-	Első dropout réteg
MaxPooling2D	-	2x2	-	-	Túltanulás elleni védelem, neuronok véletlenszerű kikapcsolása
Dropout	-	-	-	0.2	Második konvolúciós réteg
Conv2D	128	3x3	ReLU	-	Második pooling réteg
MaxPooling2D	-	2x2	-	-	Második dropout réteg
Dropout	-	-	-	0.3	Harmadik konvolúciós réteg
Flatten	-	-	-	-	Harmadik pooling réteg
Dense	128	-	ReLU	L2 regularizáció (0.001)	Harmadik dropout réteg
Dropout	-	-	-	0.3	Adatok átalakítása 1D vektorrá (laposítás)
Dense	64	-	ReLU	L2 regularizáció (0.001)	Első teljesen összekapcsolt réteg
Dropout	-	-	-	0.3	Dropout réteg 1
Dense	1	-	-	-	Második teljesen összekapcsolt réteg
					Dropout réteg 2
					Kimeneti réteg (kanyarodási szög)

46. ábra: Az autó által használt egy kimenetű CNN modell struktúrája

A háló létrehozását követően a TensorFlow keretrendszer segítségével a **model.fit()** függvény megkezdje a modell betanítását, ahol az **epoch** szám határozza meg, hogy hányszor fut végig a képzés a tanító adatbázison. A **validációs érték** segítségével a modell képes a saját teljesítményét ellenőrizni a betanulás közben. A **LossHistory()** függvény segítségével egy szöveges fájlba folyamatosan rögzítésre kerül a tanulás közbeni **veszteség(loss)** és **validációs veszteség(val_loss)** értéke, melyek grafikonos ábrázolásával a tanítási folyamat minőségéről lehet hasznos információkat kiolvasni. A betanított modellt ezt követően megfelelő formátumba kell konvertálni a Jetson Nano-n futó Linux számára, ezért a modell **ONNX(Open Neural Network Exchange)** formátumba kerül elmentésre az **export_onnx()** függvény által. Ezután a betanított model könnyedén áthelyezhető az autó rendszerére és felhasználható az előrejelzést végző program (**onnx_model_vezerlo.py**) által. A tanító program teljes forráskódja a **10. mellékletben** található.



47. ábra: A "neuralis_halo_tanitas.py" program folyamatábrája

A fentiekben bemutatott tanítási módszerrel létrehozott CNN modell egy kimenettel rendelkezik, amelyen keresztül a kanyarodási szög becsült értékét képes továbbítani. A sebességi érték meghatározásához a modell felépítésében egy újabb kimeneti réteg hozzáadása szükséges, így két célértékű lesz a hálózat. Ennek megvalósításához a bemutatott program egy módosított változata szükséges, amely már a sebesség mintaadatait is felhasználja a modell betanításánál. A módosított verzió forráskódja a **12. mellékletben** található.

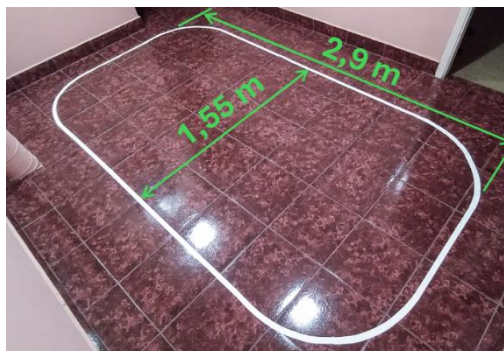
5. A CNN MODELL BETANÍTÁSA ÉS TESZTELÉSE VALÓS KÖRNYEZETBEN

A hardver és szoftver elkészítését követően a rendszer valós körülmények közötti tesztelése következett, melynek elvégzéséhez két különböző helyszínen kialakított, eltérő felépítéssel rendelkező tesztpályát alakítottam ki. A tanító adatok megfelelő eloszlási arányát, valamint a CNN modellek betanításánál használt számos paraméter és tényező optimális értékeit az első pályán elvégzett kísérletek eredményeinek kiértékelésével állítottam be. A második pályán az így meghatározott módszer alkalmasságát teszteltem egy eltérő környezetben.

A következő alpontokban a tanító minták elkészítési folyamatát, a betanítás során kapott eredményeket, valamint a modellek működési tesztjeit ismertetem és értékelem.

5.1. Az első tesztpályán végzett kísérletek

Az első tesztpályát egy kisméretű helységben alakítottam ki, ahol a lehető legnagyobb pályát rajzoltam fel fehér papírcsíkok segítségével a bordó színű járólapon. A kivitelezett pályát a **48. ábra** szemlélteti. Felépítését tekintve szimmetrikus elrendezésű, egyenesekből és ívelt kanyarokból áll, amely kiindulásként megfelelő alapot biztosított a tanító adatbázis rendszerének kidolgozásához és a CNN tanítási paramétereinek behangolásához.



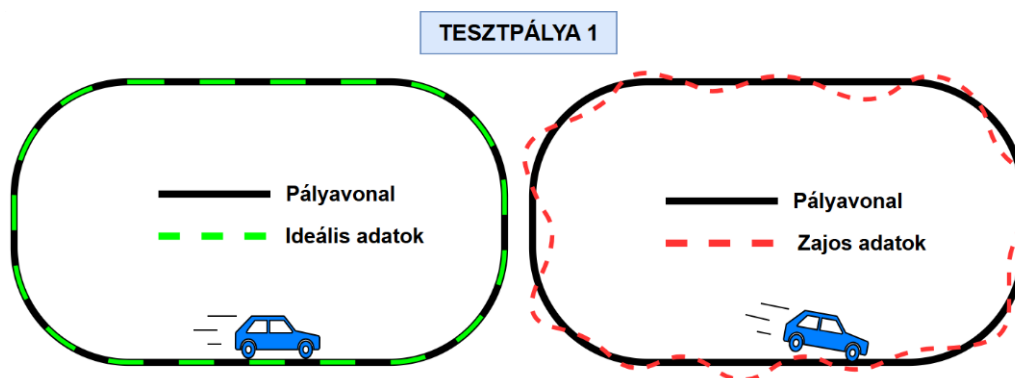
48. ábra: 1-es tesztpálya

5.1.1. Tanító adatok gyűjtése és a tanító adatbázis létrehozása

A kísérletek elvégzéséhez első lépésként a programok bemutatásánál említett módszer alapján tanító adatokat gyűjtöttem adott mintavételi frekvenciával az autó billentyűzettel történő manuális vezetése közben. Elsőnek az ideális tanító adatokat vettem fel, ahol a lehető legpontosabban vezéreltem végig az autót a pályán, miközben azonos képkockamennyiséget rögzítettem mindkét menetirányban (jobb és bal). Az ideális adatok mellé azonban szükséges volt úgynevezett zajos adatok felvételére is, melyeket pontatlanabb vezetési stílussal idéztem elő. Ezek rögzítése elengedhetetlen tényező volt, mivel azok a neurális hálózatok, melyek csak ideális adatokon tanulnak, kevésbé lesznek képesek a valós helyzetekben általánosítani, így az adott feladatot kevésbé tudják helyesen elvégezni. Ez a negatív jelenség megszüntethető zajos,

pontatlan adatok korlátozott mértékű bevitelével a tanító adatok közé. Ezek begyűjtésénél ismételten mindkét irányba azonos mennyiségű fényképet és vezérlési adatot rögzítettem.

Az így szerzett minták ezres nagyságrendben mérhetőek, például a negyedik mintavételezésnél a bal és jobb menetirányban 4.000-4.000 képet rögzítettem, még zajos adatokból mindkét irányba 1.000-1.000 mintát vettem fel, így összességében egy 10.000 mintából álló adatbázishoz gyűjtöttem nyers adatokat. A nyers adatok felépítését az előző fejezet **42. ábrája** szemlélteti. Az első pályán folytatott tesztek során négy különböző mintavételből származó adatkészlet felhasználására volt szükség a megfelelő betanítási paraméterek meghatározásához, ahol fontos tényezőként jelent meg az adatok minősége.



49. ábra: Ideális és zajos adatok felvételének elvi ábrája az első tesztpályán

Következő lépésben a felvett nyers adatokat a „*tanito_adatok_elokeszítése.py*” program segítségével rendeztem a tanításra alkalmas adatbázis formátumába. Mivel a modellek betanításánál az egyik legfőbb paraméter a minták egymáshoz viszonyított aránya volt, a helyes tanítási paraméterek meghatározásához vezető számos kísérletből adódóan egy adatkészletből több adatbázist is készítettem, ahol az ideális és zajos adatok arányait módosítottam a soron következő kísérlet szükségleteinek megfelelően.

A kialakított adatbázisban a korábbiakban említett módon adatpárok kerültek elhelyezésre, ahol minden egyes képhez párosul egy sebesség és egy kanyarodási szögérték is, melynek felépítését az előző fejezet **45. ábrája** szemlélteti.

5.1.2. CNN betanítása és tesztelése

A betanítás során az első pálya kis méretéből adódóan az autó sebességének jelentős mértékű változtatására minimális lehetőség volt. A tanító adatok felvételénél láthatóvá vált, hogy a manuális vezetés közben az autó a kijelölt kis méretű ívek pontos bevitelénél problémába ütközik a magasabb sebességből adódóan. Ebből kifolyólag ezen a pályán a pontos manőverezésre helyeztem a hangsúlyt, amely a kanyarodási szögek minél pontosabb becslését vonja maga után. Ennek megfelelően az itt elvégzett betanítási kísérleteknél az utolsó teszt kivételével (*modell_28*) csak a szögértékek meghatározására tanítottam be a modelleket a

neural_halo_tanitas.py program segítségével, melynek működését a 3.2.2-es fejezetben ismertettem.

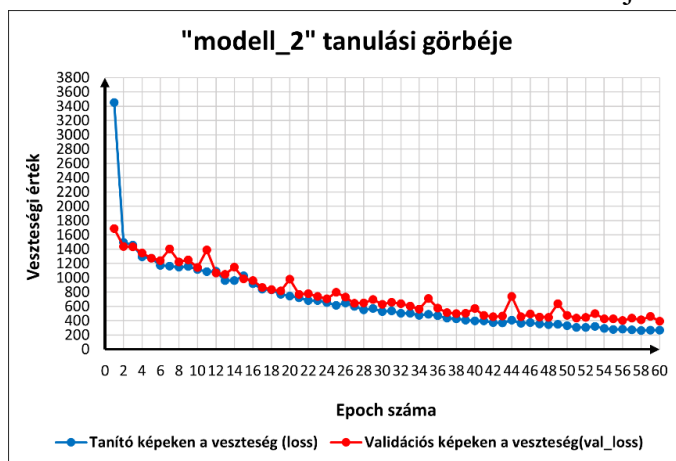
A betanítás során számos paramétert alkalmaztam, melyek értékeit az egyes kísérletekre lebontva a **13. mellékletben** lévő táblázat tartalmazza. A kísérletek során a következő paramétereket használtam:

- **Ideális mintaadatok száma (Mintakép BAL, Mintakép JOBB):** Az alkalmazott tanítóadatbázisban található ideális adatok számát mutatja.
- **Zajos adatok száma (Mintakép BAL_zajos, Mintakép JOBB_zajos):** Az alkalmazott tanítóadatbázisban elhelyezett zajos adatok számát adja meg.
- **Szegmentált mintaképek:** Megadja, hogy a tanítás alatt felhasznált mintaképek szegmentált formában kerültek-e felhasználásra.
- **Kanyarodási szög léptéke:** A kormányzó szervó motor szögértékei közötti lépésközt adja meg (pl: 10°-os lépésköz: 95°→105°→115°→125° stb...). Számítható belőle, hogy modellnek hányféle kimeneti értéken keresztül kell betanulnia.
- **Epoch szám:** Annak száma, hogy a tanítóadatokon hányszor ment végig a modell a betanulás során.
- **Tanulási ráta (Learning rate):** Megadja, hogy a modell súlyainak módosítása milyen léptékkal történhet a tanulás során.
- **Korai leállás:** Megmutatja, hogy volt-e teljesítmény alapú korai leállítást végző függvény alkalmazva a tanítás során. (Stagnáló teljesítménynél leállítja a tanítást).
- **Előre kevert adatok:** Azt jelzi, hogy a mintaadatok rendezett vagy kevert formában álltak rendelkezésre az adatbázisban.
- **Véletlenszerű adatbeolvasás tanításkor:** A modell tanításakor engedélyezve volt-e a mintaadatok véletlen sorrendű betöltése és feldolgozása.
- **Elkülönített tesztadatok aránya:** Megadja, hogy a rendelkezésre álló tanító adatok összességének hány százaléka kerül elkülönítésre validációs célokból a betanítás alatt.

A tanulási folyamat során a modellek mindegyikénél rögzítésre kerültek a veszteségfüggvényből származó négyzetes hibaértékek, melyek a tanító adatokon és a validációs adatokon nyújtott teljesítményből következnek. Ezek ábrázolásával megkapható az adott tanítási folyamat úgynevezett **tanulási görbéje**, amely a neurális háló fejlődését írja le. A paraméterek finomhangolása során ezen grafikonok elemzéséből származó információk segítettek abban, hogy mely részeket kell módosítani a tanulási minőség javulása érdekében. A következőkben **13. mellékletben** lévő táblázat szerint haladva, a modellek tanulási görbéit

felhasználva ismertetem a tapasztalatokat, azonosított jelenségeket és a levont következtetéseket.

Az első adatkészlet esetében 1.000-1.000 ideális adatot használtam fel az **1.** és **2. kísérlethez**. Ebben a kezdeti szakaszban a modell struktúrája egyszerűbb volt, kevesebb réteggel rendelkezett. A paramétereket tekintve a minták a rögzítésüknek megfelelően időrendi sorrendben kerültek feldolgozásra keverés nélkül. A kanyarodási szögértékek lépésközeként 10° -ot használtam, melyből következően a 25° -tól 165° -ig terjedő mozgástartományban 15 különböző kimeneti célértéket biztosítottam a neurális hálónak a tanulás során. Validációs adatnak 20%-ot, vagyis 400-képet különítettem el, így a tanuláshoz 1600 kép maradt. Az **50. ábrán** látható tanulási görbe alapján a 60-as **epoch** számmal tanított modell megfelelően konvergált a helyes értékek felé a veszteségi függvény által adott négyzetes hibaértékek alapján, azonban a teszteléskor a modell nem volt képes a pályavonalat követni, mivel rögtön eltévedt. A hibás működést a kevés rendelkezésre álló mintának tulajdonítottam.

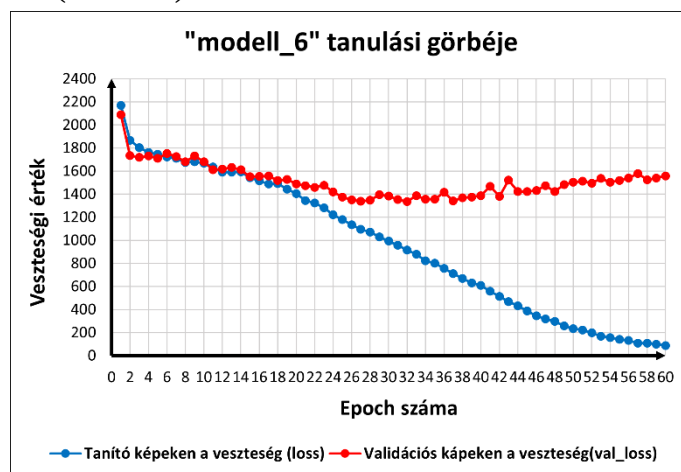


50. ábra: "modell_2" tanulási görbéje

A következő kísérleteknél új adatkészletet vettem fel nagyobb elemszámmal. Ideális adatból 4.000-4.000, zajos adatból pedig 2.000-2.000 mintát rögzítettem. A finomabb kormányzás elérése érdekében 5° -ra csökkentettem a szervómotor vezérlésénél a lépésköz értékét, amelyről utólagosan kiderült, hogy tovább rontotta a modell működését, mivel így már 15 helyett 30 különböző kimeneti értéket kellett megtanulnia a neurális hálózatnak, így nagymértékű pontatlanságok jelentek meg a predikciókban. Ez a hibás lépés a következő adatkészlet felvételénél került korrigálásra.

Az **5. modelltől** kezdődően elkezdtem használni a tanító adatok véletlenszerű feldolgozását, mivel az időrendben lévő adatok esetén a pálya kis méretéből adódóan kialakult egy ciklikusság a mintákban a sok, egymás után megtett kör hatására. A túlzott ciklikusság miatt egy mintázat alakult kis az adatkészletben, melynek hatására megjelent a **túlilleszkedés jelensége**, amely a neurális hálózatoknál egy komoly és gyakori probléma. A jelenség

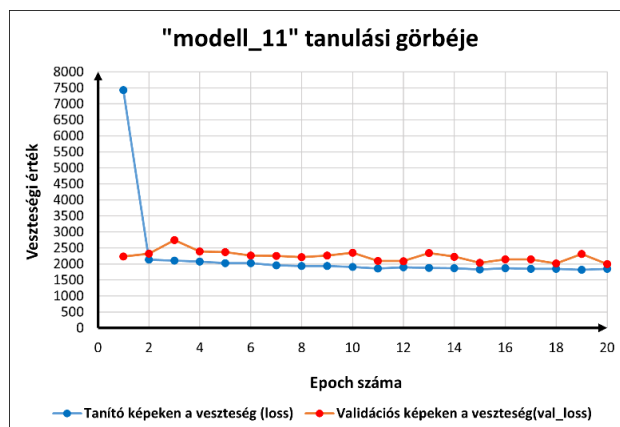
értelmében a modell a tanító adatokon jól teljesít, mivel megtanulja az adatokban lévő mintázatot, a validációs adatoknál viszont egyre rosszabb eredményeket ér el, vagyis az általánosító képessége az új, ismeretlen bemenetekre nem fejlődik. Ilyenkor egy jellegzetes szétívelő tanulási görbét kapunk eredményül, amely a **6. és 9. kísérletnél** jelentkezett a legnagyobb mértékben (**51. ábra**).



51. ábra: Túlilleszkedés jelensége a 6. kísérletnél

A **túlilleszkedés** problémáját első sorban az **epoch** szám csökkentésével próbáltam megszüntetni, viszont a veszteségi érték így sok esetben nagyon magasan maradt. További próbálkozások során kipróbáltam a szegmentált képeken való betanulást, ahol a fehér pályavonalat elkülönítettem a háttértől, viszont így még egyértelműbb mintázatot hoztam létre a tanító adatbázisban, így ez az irány gyorsan elvetésre került. A kísérletek során több eltérő aránnyal rendelkező adatbázist is felállítottam, azonban azokkal sem értem el jobb eredményt. Egyes esetekben az autó minimálisan elkezdte követni a vonalat, viszont sok esetben egyáltalán nem tudta ezt megtenni.

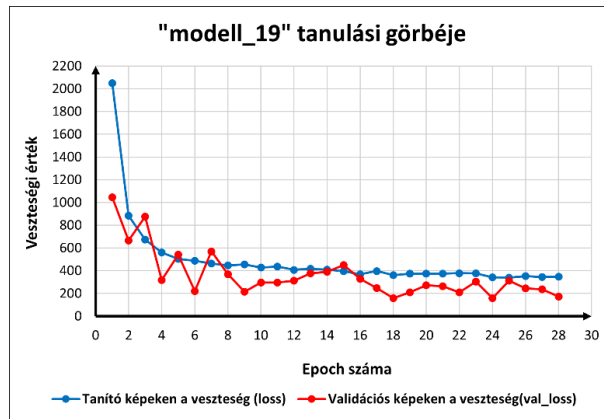
A nagymértékű túlilleszkedési jelenséget a **11-es modellnél** bevezetett módosításokkal sikerült megszüntetnem. A tanulási ráta csökkentésével finomítottam a neurális háló tanulása közben történő súlyainak változtatási lépésközét. Emellett csökkentettem az adatmennyiséget és a modell struktúrájához újabb, **dropout** rétegeket adtam, melyek a **túlilleszkedés** ellen védik a modellt a neuronok egy részének véletlenszerű deaktiválásával. Az így kapott tanulási görbénél (**52. ábra**) a tanító és validáló adatokon nyújtott veszteségi értékek kellően közel helyezkedtek el egymáshoz képest és csökkenő tendenciát mutattak, viszont a végeredményt tekintve a veszteségi érték nagyon magasan maradt, amely szintén hibás működést okozott a teszteknel.



52. ábra: A 11. modell tanulási görbéje

Következő lépésként elvégeztem a harmadik adatgyűjtést is, így egy új adatkészleten folytattam a teszteket, ahol 3.000-3.000 ideális és 1.000-1.000 zajos adatot vettem fel. A szervó motor lépésközét visszavettem 10 °-ra, így csökkentve a kimeneti variációk számát. A korábbi beállítások során az adatokat rendezett formában használtam fel, majd a betanításkor véletlenszerűen adagoltam a neurális hálózathoz, így próbálva a kis pályán gyűjtött adatok ciklikusságából adódó erős mintázatot csillapítani. A következőkben ezen módosítva a mintákat már előre, véletlenszerűen összekevertem, melynek hatására a **12-es modelltől** fogva minimális pályakövetéseket produkált az autó a teszteken. A **15-ös modelltől** kezdődően a tanulási ráta értékét dinamikusra váltottam, így az a tanulás közbeni teljesítmény függvényében tudott dinamikusan változni. Mindemellett kiegészítésre került a tanító program a korai leállás módszerével, amely folyamatosan figyeli a hálózat teljesítményét. Ennek lényege, hogy ha adott **epoch** számon belül stagnálást mutat a veszteségi érték, akkor leállítja a tanulást és a legjobb eredményhez fűződő súlyokat állítja be a modellben.

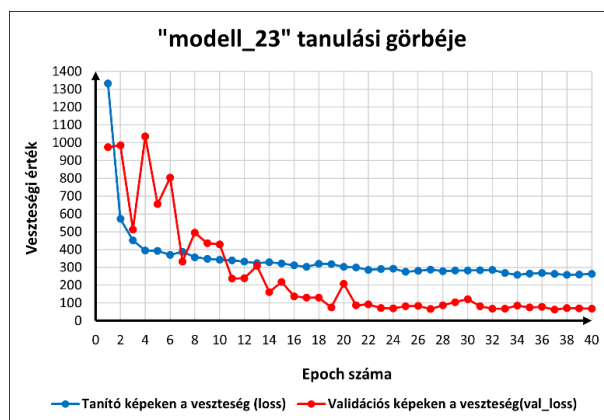
Az első nagy áttörés a 19-es modellnél következett be, ahol a validációs tesztadatok arányát 15% -ra állítottam és újabb rétegeket adtam a modell felépítéséhez, amely ezen a ponton érte el a végleges struktúráját, melyet a korábbi **46. ábra** alapján ismertettem a 3.2.2-es pontban. Az így betanított modellt felhasználva a teszt során az autó már egyszer körbe tudott menni a pályán teljesen önállóan. A tanulási görbét az **53. ábra** szemlélteti, amely alapján jól látható, hogy a validáció során nyújtott teljesítménynél csökkent az ingadozás és szinte végig jobb eredményt adott, mint a tanító adatok esetében.



53. ábra: Az áttörésnél betanított modell tanulási görbéje

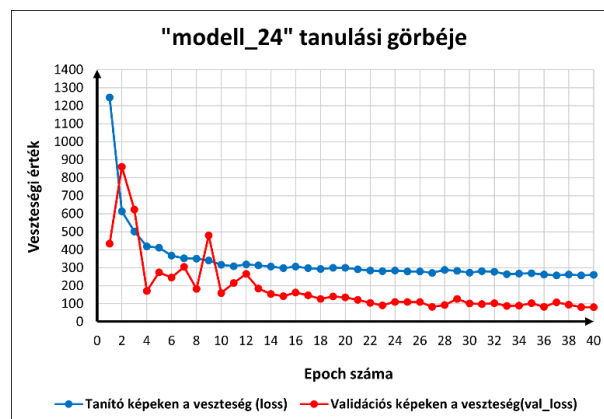
A **19-es modell** sikeres működéséből kiindulva újabb adatkészletet rögzítettem, amelynél fokozottan ügyeltem a minél pontosabb vezetésre, mivel a korábbi adatkészletek minőségét vizsgálva az adatgyűjtéskor ejtett vonalkövetési hibák jól láthatóan hozzájárultak a tanulási minőség ingadozásához. Az adatkészlet ez esetben 4.000-4.000 ideális és 1.000-1.000 zajos adatot foglalt magába. Paramétereiket tekintve az adatok arányán kívül a mászt nem változtattam, a **19-es modell** mintáját követtem. A **23, 24, 25, 26 és 27-es kísérleteknél** a zajos adatok arányát nullától kiindulva lépcsőzetesen növeltem, így meg tudtam vizsgálni, hogy milyen hatással van az egyre nagyobb arányban lévő pontatlan adat a modell teljesítményére.

A **23-as modell** tanulási görbéje alapján elmondható, hogy a neurális háló általánosító képessége nagyságrendekkel javult (**54. ábra**). Ez a tanító és a tesztképeken nyújtott veszteségi értékek összehasonlításánál figyelhető meg, hiszen a betanítás végén közel negyedére csökkent a tesztadatokon produkált veszteségi érték. A tesztelés során az autó könnyedén körbement többször is a pályán, azonban egy pár kör elteltével, amikor pontatlanabb ívet vett az autó gyorsan eltévedt. Ez azzal magyarázható, hogy a zajos adatok hiányában nincs felkészítve a modell arra, hogy a nagyobb eltéréseket hogyan kell korrigálnia.



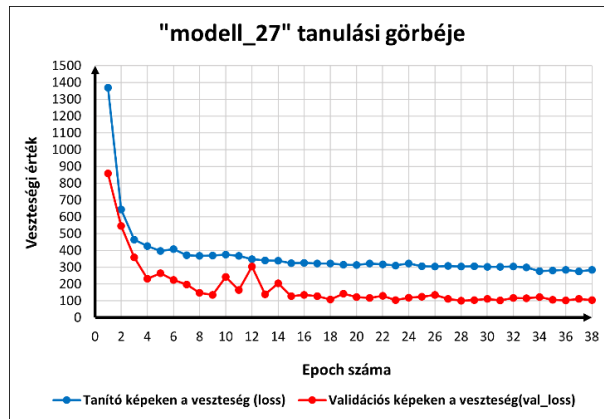
54. ábra: Tanulási görbe a 23-as kísérletnél

Az összes kísérlet közül a 24-es számú modell tanítása sikerült a legjobban (55. ábra), melynek tesztelése alatt az autó már percekig képes volt mindkét irányba manőverezni a pályán anélkül, hogy eltévedt volna. A használt adatbázisban a 8.800 mintaadatból 1.320 került elkülönítésre validációs célokból, a zajos és ideális adatok közötti arány 1:10 volt. A **23-as kísérlettel** ellentétben a pontatlan adatok beépítésével sikerült felkészíteni a neurális hálót az esetleges eltérésekre, melynek köszönhetően az autó már megfelelően tudta követni a pályát. A működésről készült videófelvétel a **15. mellékletben** lévő link vagy QR kód segítségével tekinthető meg.



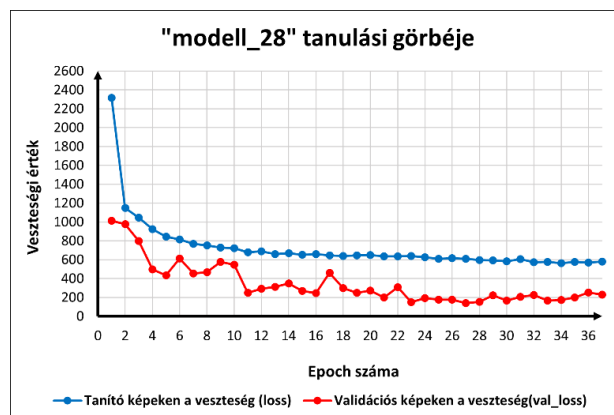
55. ábra: A legsikeresebb modell tanulási görbéje

A **25, 26 és 27-es kísérletek** esetében elkezdtem növelni a zajos adatok számát a tanító adatbázisban, így több lépcsőben tudtam vizsgálni azok hatását a pontosságra. Összességében mindegyikről elmondható, hogy a tesztek során az autó már egyre lazábban kezelte a kanyarokat és az egyenes szakaszok követését, melynek következtében könnyebben elvesztette a pályát és eltévedt. A **27-es modell** esetében a negyedik adatkészlet összes mintáját beépítettem az adatbázisba, így a modell összesen 10.000 tanító adatot dolgozhatott fel a tanulás során. A **25-ös és 26-os modellek** tesztelésénél tapasztalt egyre növekvő hibaarányból kiindulva ennél a variációnál még nagyobb mértékű romlást vártam a pályakövetés pontosságában, azonban ennek az ellenkezője történt meg. A tanulási görbéket összevetve ugyan nem ért el olyan jó végeredményt, mint a **24-es kísérlet**, viszont a tesztelés során szinte ugyan úgy zökkenőmentesen körbe tudta vezetni az autót a pályán (**56. ábra**). Ennek oka feltehetően a zajos adatokban lévő nagyobb és kisebb eltéréseket tartalmazó minták közötti egyenetlen eloszlás lehet, ahol az ezredik adathoz közelítve már az ideálishoz hasonló adatok kerülhettek rögzítésre. Ebből a jelenségből jól látható, hogy milyen mértékben számít a tanító adatok minősége egy neurális hálózat tanításakor.



56. ábra: A legnagyobb adatbázissal tanított modell tanulási folyamata

Az első pályán végzett utolsó kísérlet (*modell_28*) esetén a **11. és 12. mellékletben** lévő módosított programokat használtam fel egy két kimenetel rendelkező CNN modell betanításához, amelynél megvizsgáltam, hogy milyen hatással van a korábbi kísérleteknél tapasztalt teljesítményre, ha a modellnek több értéket (kanyarodási szög és sebesség) is meg kell határoznia, így megosztva az erőforrásait. A **27-es kísérlethez** hasonlítva az itt elért eredményeket tekintve érzékelhető a modell teljesítményének megosztottsága, a tanulási görbén látható utolsó veszteségi érték az előzőek duplájára nőtt, a tanulás során nagyobb ingadozás jelentkezett, valamint az utolsó *epoch*-oknál sem ált be egy stabilitás (**57. ábra**). A tesztelés során jelentkezett a pálya kis mérete és a sebesség között fennálló probléma, melyet korábban ismertettem. Ennek következményében az autó könnyedén eltévedt, azonban jó viszonyítási alapot kaptam az egy, illetve két kimenetű neurális hálózatok várható teljesítményével kapcsolatban.



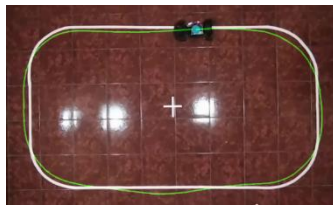
57. ábra: A két kimenettel rendelkező modell tanulási görbéje

5.1.3. Pályakövetési pontosság mérése

Az előzőek alapján látható, hogy az első pályán sikeresen megvalósult az önvezető funkciót biztosító CNN modellek betanítása, melyek körbe tudták vezetni az autót a kijelölt pályán. A tesztek során szembetűnő volt, hogy az autó által megtett nyomvonal és az eredeti

pályavonal között kisebb eltérés mutatkozik. Ez arra utal, hogy a CNN modell által megvalósított szabályzás adott mértékű hibával rendelkezik. Ezen hiba mérésére elvégeztem egy képfeldolgozás alapú pályakövetési mérést, melynek eredményeképpen megkaptam az eredeti/referencia pálya és az autó által megtett útvonala között fennálló eltéréseket, melyekből számítható a szabályzás pontosságával szorosan összefüggő pályakövetési hiba.

A méréshez az autó által megtett útvonalat egy felülnézetből készített videófelvétellel rögzítettem, amelyen a teljes pálya látható. A vizsgálathoz az autón egy, a környezettől eltérő színű korongot helyeztem el, amely lehetővé teszi az autó által megtett út képfeldolgozási módszerekkel történő nyomon követését. A méréshez mindkét menetirányban rögzítettem felvételeket, amelyekből 3-3 egymást követő kört emeltem ki. A videófelvételeken ezt követően elvégeztem a megfelelő pozicionálási és nagyítási korrekciókat annak érdekében, hogy a felvételtől kinyerhető pixel koordinátákból álló útvonalat minél pontosabban át lehessen konvertálni valós, méterben értelmezett pályakoordinátákra. Az előkészített videón ezt követően a **18.mellékletben** található program segítségével a mozgó korong pozícióit képkockáról képkockára rögzítettem, ezáltal megkaptam az autó útvonalát leíró pixel koordinátákat, melynek kirajzolt képét szemlélteti az **58. ábra**. A koordinátákat ezt követően átskáláztam a valóságnak megfelelő méter mértékegységbe.

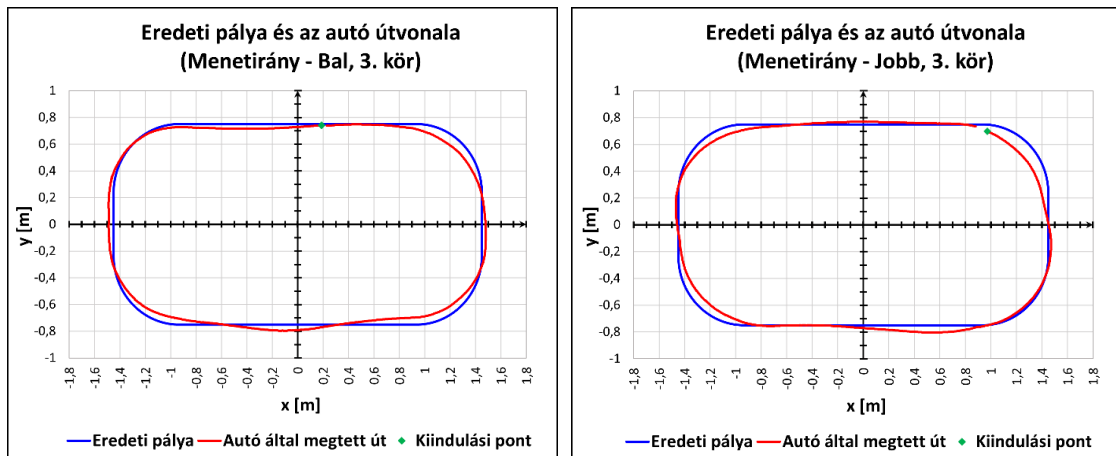


58. ábra: Python-OpenCV segítségével feldolgozott videó eredménye az autó által megtett útvonal kirajzolásával

A videófelvételből kinyert pályakoordinátákat ezt követően körökre bontva közös koordinátarendszerben ábrázoltam az eredeti pálya ismert koordinátaival. A két menetirányból kiemelt egy-egy kör ábrázolt grafikóját a **59. ábra** mutatja.

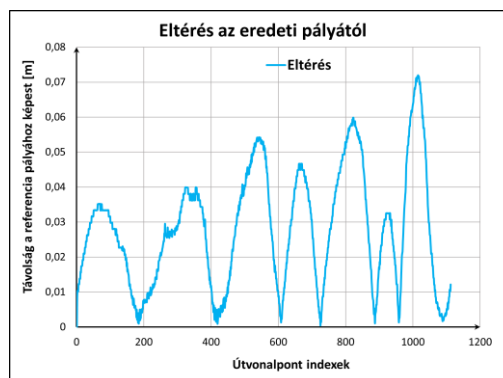
Az ábrák alapján a következők mondhatóak el: Az egyenes szakaszokon minimális eltérés mutatkozik a pályakövetésnél, amely a CNN stabil és pontos működésére utal. A kanyarok esetében már nagyobb eltérések tapasztalhatóak mindkét irányban, mivel az autó kissé szélesebb ívet vesz, alulkormányoz, amely azt mutatja, hogy a szabályzás ezen a területen finomításra szorul. Itt azonban fontos megjegyezni, hogy a szabályzás minősége mellett a pálya kis méretei nem kedveznek az autó fordulási körének, melyet maga a mechanika határoz meg, így az itt jelentkező hibát ez is befolyásolja. Az autó által leírt pályát vizsgálva a jobb és bal menetiránynál is elmondható, hogy a pályavonal nem mutat instabilitást, nagy ugrásokat vagy

kanyargós részeket, így a szabályzás megfelelően kezeli a pálya kisebb módosulásainak lekövetését, valamint a nagyobb eltérések korrigálását.



59. ábra: Az eredeti pálya és az autó által megtett út ábrázolása közös koordináta-rendszerben – Jobb/Bal menetirány – 3.kör

A mérés következő lépéseként a rendelkezésre álló referencia pálya, illetve az autó pályájának koordinátái alapján a két pálya eltéréséből számításra került a pályakövetési hiba. Ennek meghatározását segíti a **19.mellékletben** található programkód. A program első lépésként beolvassa a két pálya koordinátáit tartalmazó .csv fájlokból. Ezt követően a referencia pálya koordinátáinak elemszámát illeszteni kell az autó pályáját reprezentáló pontok számához, amely az eltérő ívek miatt minden esetben több pontot fog tartalmazni. Az eredeti pálya pontjainak kibővítését a program lineáris interpolálással elvégzi, így egyenlő számú pontpárokat kapunk, melyek már összehasonlíthatóak. Következő lépésben a program elvégzi az autó pályapontjainak és a referencia pontok közötti távolságok számítását. Ezek grafikonos ábrázolását a **60. ábra** szemlélteti, melyen jól látható, hogy az autó a pálya mely szakaszán produkálta a legnagyobb eltérést az eredeti pályához viszonyítva.



60. ábra: A számított távolságok a bal menetirány 3. körében

A program végül a már rendelkezésre álló eltérési távolságokból kiszámítja a pályakövetés átlagos, valamint a nagyobb eltérésekre érzékenyebb négyzetes átlag hibáját, melyek táblázatos eredményeit a vizsgált körökre lebontva a **61. ábra** mutatja. Az eredmények

alapján elmondható, hogy az eredeti pálya, valamint az autó által leírt nyomvonal közötti hiba a jobb menetirányú üzem közben nem haladja meg a 4,5 [cm]-t, még a bal menetiránynál vizsgált mozgásvonalaknál ez az érték 7 [cm] -en belül marad. A mérés leírása és a vizsgált pályák grafikonnjai a **17. melléklet** tekinthetők meg.

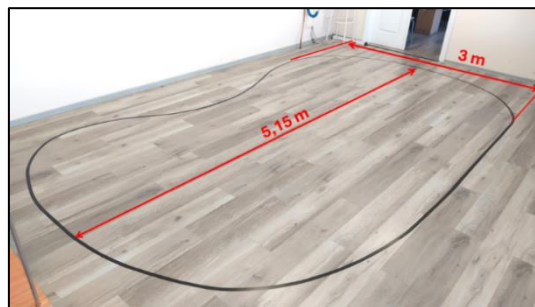
Pályakövetési hiba mérési eredményei			
Menetirány	Kör	Átlagos hiba [m]	RMS hiba [m]
Jobb	1.	0,0332	0,0413
Jobb	2.	0,0265	0,0324
Jobb	3.	0,0284	0,0380
Bal	1.	0,0380	0,0475
Bal	2.	0,0534	0,0695
Bal	3.	0,0282	0,0329

61. ábra: A pályakövetési hiba számított eredményei

A mérési eredmények alapján megállapítható, hogy az autó 19,3 [cm]-es szélességéhez mérten az **átlagos 7 [cm]-es követési hiba** jó eredménynek minősíthető egy ilyen kis méretű jármű esetében, hiszen a CNN modell szabályzása során az autó nagy része mindig a pálya vonalán halad és nem jelentkezik nagy mértékű eltérés. A szabályzás így stabil pályakövetést biztosít az autónak az önvezetés során.

5.2. Kísérletek a második tesztpályán

A második tesztpályát egy teljesen eltérő környezetben alakítottam ki, ahol a nagyobb rendelkezésre álló hely lehetővé tette egy nagyobb pálya felrajzolását, amely az elsőhöz viszonyítva közel kétszer akkora méretekkkel rendelkezett. A vonalakat fekete színnel jelöltem a szürkésbarna padlóburkolaton, a pálya szerkezetét tekintve bonyolultabb felépítést választottam, így egyenes elemekből csak egy rövid szakasz került elhelyezésre, a pálya többi része nagyobb, egybefüggő ívekből állt. Az elkészült pályáról elmondható, hogy a komplexebb elrendezéséből adódóan nagyobb kihívást nyújt az autó önvezető képességének. A második tesztpálya kialakítását és méreteit a **62. ábra** szemlélteti.



62. ábra: A 2. tesztpálya kialakítása

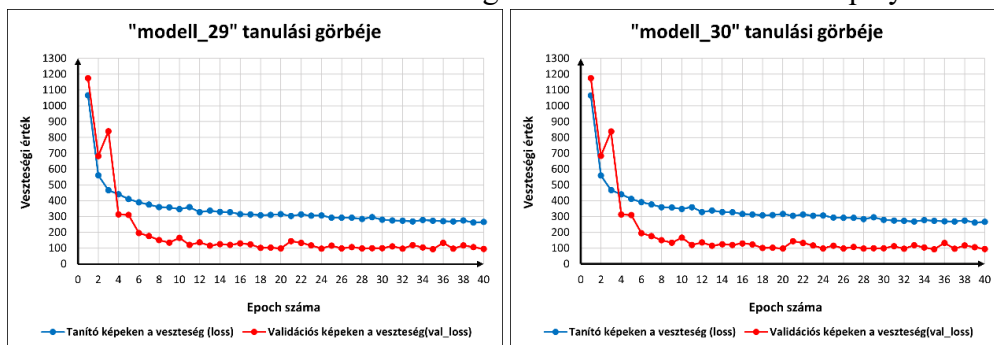
A különböző neurális hálózatok teszteléséhez első lépésben ismételten tanító mintákat gyűjtöttem. A korábban ismertetett módon ideális és zajos adatokat vettem fel, majd azokat tanító adatbázisokba rendeztem a modellek betanításához. A kísérletek során két különböző mintagyűjtésből származó adatkészletre volt szükség, mivel a modellek tesztelési eredményei

ráműtött arra, hogy a pálya méretének megnövelésével a tanító adatok számát is növelni kell a sikeres betanulás érdekében. Ezen jelenség részleteire a teszteredmények leírásánál térek ki.

5.2.1. A betanított modellek teszteredményei

A második tesztpályán folytatott kísérletek paraméterei megegyeznek az elsőnél használtakkal. Ezek táblázatos összefoglalóját az itt alkalmazott modellekre lebontva a **14. melléklet** tartalmazza, amely alapján a következőkben ismertetem az elért eredményeket és tapasztalatokat.

Az ötödik adatkészlet esetében 5.000-5.000 ideális és 1.500-1.500 zajos adatot mintavételeztem, majd három tanító adatbázist alakítottam ki, melyeket a **29, 30, 31, 32 és 33-as kísérleteknél** használtam fel. Minden adatbázison két modellt tanítottam be, melyek közül az egyik csak a kanyarodási szöveget adta meg, a másik a sebességi értékkel együtt két kimenettel rendelkezett, így összehasonlítható volt az egy, illetve két kimenetű hálózatok teljesítménye. Az első modellek esetében, melyek a **29, 30-as kísérlet** során tanítottam be, a tanulási görbék képe erősen hasonlított az első pályán kapott sikeres modellek görbéjére, viszont a teszteléskor az autót nem voltak képesek a pályán tartani (**63. ábra**). A **30-as modell** esetében a sebesség szabályozásánál a hálózat helyesen detektálta a kanyarokat, így csökkentette a sebességet, azonban a szabályzásban túl nagy lépésközök jelentek meg. Ennek hatására túl nagy sebességugrások alakultak ki az autó mozgása közben, melynek okán az autó gyorsan letért a vonalról és eltévedt. A két tanulási görbénél megfigyelhető, hogy a két célértékkel rendelkező hálózatonál korábban tapasztalt teljesítmény romlás a **30-as modell** esetében nem jelent meg, mivel közel azonos veszteségi értéket produkált, mint az egy kimenetű változata. Ez feltehetően a pálya nagyobb méretének köszönhető, amely lehetővé tette, hogy a mintavételezskor a manuális vezetés alatt változatosabb sebességekkel lehetett közlekedni a pályán.



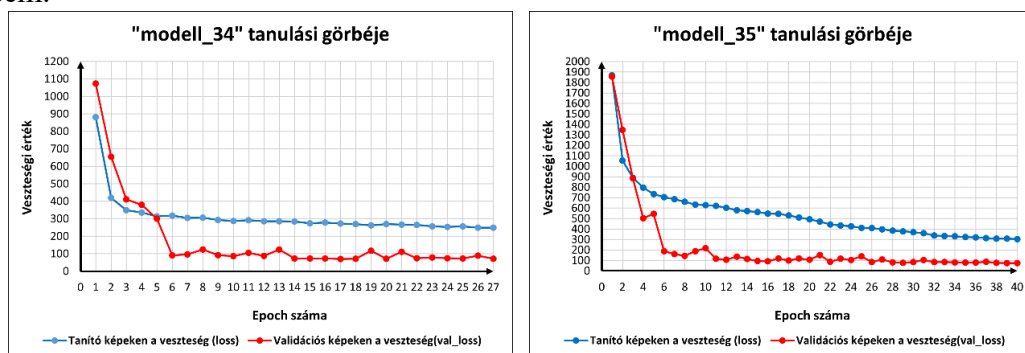
63. ábra: A második tesztpálya első CNN modelljeinek tanulási görbéi

A **31-es és 32-es hálózatok** esetében két, egy kimenettel rendelkező modellt tanítottam be eltérő mennyiségű mintaadaton. A két modell összehasonlításánál az eltérésnek köszönhetően megmutatkozott, hogy a kevesebb és a több adattal betanult hálózatok között

lényeges különbség tapasztalható, mivel a **32-es modell** működésénél a nagyobb mintaszám hatására a pálya egyes szakaszait már követni tudta az autó.

Következő lépésként új adatkészlet felvételére volt szükség, ahol a korábbi tapasztalatok alapján megnövelt adatmennyiséget rögzítettem. A kanyarodási adatok mellett igyekeztem az autó sebességét enyhébben változtatni annak érdekében, hogy a korábbi teszteknel tapasztalt durva sebességszabályozást a későbbiekben finomítani tudjam. Az utolsó két modell tanításához 8.000-8.000 ideális és 800-800 zajos adatból álló adatbázist alkalmaztam. A két modellt összehasonlítva a **34-es modell** tanítása korábban befejeződött, mivel a veszteségi értéke stagnálásnak indult, így a korai leállítási függvénye megállította azt. A **35-ös modellt** vizsgálva, amely már két célértékre tanult be, hasonló veszteségi értéket ért el a tanulása végén (**64. ábra**). Ez alapján elmondható, hogy a változatosabb mintákon és nagyobb adatmennyiségen való betanítással ellensúlyozható a több kimenettel rendelkező neurális hálózatoknál korábban megfigyelt teljesítményromlás.

A kísérletek alatt a **34-es modell** kanyarodási szögbecslései segítségével többször is körbe tudta vezetni az autót a pályán nagyobb elakadás nélkül, viszont az látható volt, hogy hajlamos az eltévedésre. Ez a tulajdonság feltehetően az adatkészlet minőségére vezethető vissza. A működésről készített videófelvétel a **16. mellékletben** elhelyezett hivatkozásokon keresztül tekinthető meg. A **35-ös modell**, amely már a sebességet is szabályozza, szintén körbe tudta vezetni az autót a nyomvonalon, azonban két-három kör után egy-egy nagyobb sebességérték hatására eltérítette az autót a helyes útvonalról, amely így eltévedt. Ebből kifolyólag elmondható, hogy a sebességadatok mintavételezésénél nem sikerült a kanyarodási szögértékekhez hasonló, megfelelő minőségű adatokat rögzíteni a finom manuális szabályzás révén sem.



64. ábra: A második tesztpálya sikeres modelljeinek tanulási görbéi

A második pályán elvégzett kísérletek során sikerült két működőképes modellt betanítani az autó megfelelő vezérlésére, amely alapján megállapítható, hogy az első pályánál kidolgozott módszer alkalmazható más környezetben kialakított pályákon is.

6. ÖSSZEFOGLALÁS ÉS JAVASLATTÉTEL

A fejlesztési projekt keretében sikeresen megvalósítottam egy önvezető funkcióval rendelkező modellautót, amely konvolúciós neurális hálózat alkalmazásával képes önállóan, emberi beavatkozás nélkül, kamera segítségével navigálni egy kijelölt pályán. Az eszköz sikeres teszteredményei alapján bizonyítani tudtam a koncepció működőképességét.

A munkám során nagy segítséget adott az előzetesen rögzített kritériumrendszer, amely a tervezési szakasztól a kivitelezésig irányvonalat mutatott a projekt különböző fázisainak teljesítéséhez. Az autóváz alkatrészei tekintetében megfelelő elemeket sikerült választani, amik kompatibilitási problémák nélkül biztos alapot nyújtottak a számításokat végző központi vezérlőegységnek, a hozzá kapcsolódó indikátor áramkörnek, érzékelőnek és az egyéb kiegészítő komponenseknek, melyek összetett rendszerként való működésükkel egy önvezető funkcióval rendelkező járművet hoztak létre.

A rendszert működtető szoftveres környezet elkészítésénél az egyszerű, logikus és stabil működés volt a célom, amit a fentiekben ismertetett programoknál sikerült elérnem. A részfeladatokat elvégző programok képesek a saját feladataikat megfelelő minőségben ellátni és hatékony információcserét tudnak megvalósítani egymás között. Az architektúra adta többszörös futtathatóságnak köszönhetően képesek párhuzamosan ellátni a feladatukat, így létrehozva egy valós idejű feldolgozó és vezérlési rendszert. Az eredmények tekintetében a szoftverrendszer hatékonyan képes támogatni az autó önvezető funkcióját, mind a képfeldolgozási és vezérlési szinten egyaránt.

A modellek betanításánál elmondhatom, hogy lépésről-lépésre végig jártam egy konvolúciós neurális hálózat integrálásának minden szakaszát. A betanításhoz szükséges tanító adatbázis létrehozásához egy univerzálisan alkalmazható metódust alakítottam ki, amellyel elvégezhető a mintavételezés, adatecímkézés és rendszerezés tesztkörnyezettől függetlenül. A neurális hálózat szerkezeti kialakításánál sikerült egy célnak megfelelő felépítésű modellt elkészíteni, melynek pályakövetési pontosságát mérési eredményekkel tudtam alátámasztani. A kidolgozott eljárás alkalmazhatóságát a második tesztpályán végzett kísérleteknél sikeresen igazolni tudtam, így alátámasztva a neurális hálózatok gyors adaptálhatósági tulajdonságát.

A fejlesztett eszköz prototípus mivoltából adódóan rendelkezik kezdeti problémákkal, melyek megoldása további fejlesztések tárgyát képezheti. A működési kísérletek eredményei alapján a következő javaslatokat tudom tenni.

Az autóváz mechanikai kivitelezésében a differenciálművekben lévő fogaskerekek fogszámának növelése elősegítheti, hogy az autó a minimális sebességnél, éles kanyaroknál egyenletesen mozogjon.

A szoftverek futásakor időnként fellépő késedelmek, melyek befolyásolják az autó valós idejű vezérlését, pályaelhagyási hibákat tudnak generálni, így érdemes lehet a programkódokat további optimalizációnak alávetni, esetlegesen más, gyorsabb futással rendelkező programnyelvekre átfordítani azokat.

A tanító adatok létrehozásánál kiemelt hangsúlyt kell fektetni azok minőségére, melyek magas szinten befolyásolják a betanított modellek feladatvégzés közben tapasztalható teljesítőképességét. Az emberi, manuális vezetést érdemes lehet felváltani vonaldetektálást megvalósító automata szoftverrel, amely elősegítené az mintaadatok pontossági értékének javulását. Itt azonban ügyelni kell továbbra is az ideális és valós körülmények közötti megfelelő egyensúlyra, amely elengedhetetlen a valós környezetben szükséges magas fokú általánosítási képesség kialakulásához.

A teszteknel tapasztalt vonalkövetési hibák, eltérések sok esetben a kamera kis látószögére vezethetők vissza, így érdemes azt egy nagyobb látószöggel rendelkező kamerára cserélni, amely kiküszöbölné a nagyobb sebességeknél jelentkezett pályaelhagyási problémákat.

A neurális hálózat által prediktált sebességi értékek esetében egy köztes szabályzó rendszer kialakítása, amely a modell által adott sebességet tekintené célértéknek. Ez elősegítheti az ugrásszerű sebességkülönbségek kiszűrését, melyek szintén pályaelhagyáshoz vezethetnek kisebb pályákon, élesebb kanyarokban.

Összegzőképpen elmondhatom, hogy a projekt kivitelezése során új ismeretekre tettem szert az autonóm járművek önvezetésével kapcsolatban. A valós idejű szoftveres vezérlés, a konvolúciós neurális hálózatok adott feladatra való felhasználása és a modellek tanítási paramétereinek finomhangolása területén számos új tapasztalatot gyűjtöttem a kipróbált módszereknek köszönhetően. Az önvezető autó megvalósításának sikeressége érdekében sok időt fektettem a fejlesztésbe, melynek köszönhetően az elvárásoknak megfelelő, összetett mechatronikai rendszert tudtam létrehozni. Bízom benne, hogy az így megszerzett tudást és tapasztalatot a későbbi szakmai pályafutásom során hasznosítani tudom.