Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Mérnökinformatikus Szak

2023 évi Tudományos Diákköri Konferencia

Valós idejű pontfelhő-feldolgozó és önvezető szoftver fejlesztése

TDK Dolgozat

Készítette: Czimber Márk

Konzulensek:

Dr. Németh Krisztián

Dr. Czimber Kornél

Baranyai Dániel

Budapest

2023

Kivonat

Az önvezető járművek egy rendkívül gyorsan fejlődő iparág és kutatási terület, ami a közlekedés jövőjének felettébb ígéretes alternatívája. Jelenleg is számos kutatás és fejlesztés folyik a teljes önvezetés elérésének érdekében. A kamerák és műholdas helymeghatározás mellett az önvezető autók egyik kulcs érzékelőjévé fejlődhet a lézeres letapogatás (LiDAR), amely az autó környezetében lévő személyek és tárgyak térbeli detektálását teszi lehetővé.

A kutatásom során egy integrált valós idejű pontfelhő-feldolgozó és részleges önvezetést megvalósító algoritmus és szoftver fejlesztését tűztem ki célul. A rendszer alkalmas járművekre szerelt lézeres letapogatókból származó pontfelhők feldolgozására, ezek alapján egy jármű irányítására, beleértve a kormányzást, megállást és elindulást.

A rendszer első komponense a pontfelhő kiszolgáló modul, melynek célja az autóra szerelt lézeres letapogató szimulálása. A modul nagyméretű pontfelhő állományok betöltését, indexelését, majd pozíció és irány alapú térbeli lekérdezését valósítja meg.

A második modul feladata az autó környezetében lévő pontfelhő lekérdezése, a pontfelhőre domborzatmodell illesztése, és a modell feletti tereptárgyak voxeles detektálása, amelyek alapján az autót irányító parancsok kiadhatók.

A harmadik modul két részből áll: egy valós közlekedési helyzeteket megoldó, paraméterezhető virtuális oktatóból és az oktatótól megerősítéses tanulással tanuló önvezető modulból. Ez a páros képes felgyorsítani a tanulási folyamatot, az önvezetőt több közlekedési helyzet hatékonyabb felismerésére és kezelésére teszi alkalmassá.

A szoftver negyedik komponense egy háromdimenziós megjelenítő. Ez valós időben mutatja az autót körülvevő pontfelhőt, felületmodellt, és a tereptárgyakat.

A fejlesztés során új és továbbfejlesztett eljárásokat hoztam létre, mint térbeli indexelés, pontfelhő-lekérdezés, felület- és objektumkinyerő algoritmusok, virtuális oktató és megerősítéses tanulás alapú önvezető modul. Bízom benne, hogy a szoftverben megvalósított innovációkkal hozzájárulhatok az önvezető jármű és mobilitási kutatásokhoz.

Development of a Real Time Point Cloud Processing and Self-driving Software

Abstract

Self-driving vehicles represent a rapidly evolving industry and research area, offering a highly promising alternative for the future of transportation. Currently, numerous research and development efforts are underway to achieve full autonomy. In addition to cameras and satellite positioning, one key sensor for autonomous cars could be Light Detection and Ranging (LiDAR), which enables the spatial detection of individuals and objects in the vehicle's vicinity.

The goal of my research is to develop an integrated real-time point cloud processing algorithm and software capable of partial autonomous driving. The system is designed to process point clouds from LiDAR sensors mounted on vehicles and use them to control a vehicle, including steering, stopping, and starting.

The first component of the system is the point cloud provider module, which aims to simulate the LiDAR sensors mounted on the vehicle. The module handles the loading, indexing, and position- and direction-based spatial querying of large point cloud datasets.

The task of the second module is to query the point cloud in the vehicle's environment, fit a terrain model to the point cloud, and detect voxels of objects above the model. This information is used to issue commands for controlling the vehicle.

The third module consists of two parts: a virtual instructor capable of solving real traffic situations and a self-driving module that learns from the instructor through reinforcement learning. This pair can accelerate the learning process, making the self-driving system more capable of efficiently recognizing and handling various traffic situations.

The fourth component of the software is a three-dimensional visualizer. It provides real-time visualization of the point cloud, surface model, and surrounding terrain objects.

During the development process, I created new and improved procedures such as spatial indexing, point cloud querying, surface and object extraction algorithms, a virtual instructor, and a reinforcement learning-based self-driving module. I believe that the innovations implemented in the software can contribute to research in self-driving vehicles and mobility.

Tartalomjegyzék

1. Bevezetés

1.1. A téma aktualitása

Az álom a vezetés teljes automatizációjára először a 1939-es New York-i Világkiállításon született meg, ahol a General Motors Futurama előadása elkápráztatta a közönséget (Fabian Kröger, 2016), és nagy hullámot indított a járműiparban. Napjainkban már az önvezető autóipart óriási cégek irányítják, mint a Waymo és a GM Cruise, akiknek megvan az erőforrás- és létszámbeli lehetőségük, hogy gazdaságilag és biztonságtechnikailag vonzó autonóm autókat gyártsanak (Justin Kek, 2019).

Napjainkban sokan osztják azt a jövőképet, hogy az utakon való közlekedés okozta halálos balesetek minimalizálhatók, vagy akár teljesen megelőzhetők. A vezetés összetettsége miatt a közúti balesetek több mint 90%-a százaléka emberi hibából ered (Treat et al., 1979), mely magába foglalja a szabályszegéseket, a figyelmetlenséget és a lassú vagy rossz döntéshozatalt. Ezek az emberi hibák évente több mint 1,35 millió személy életébe kerülnek (Rezapur-Shahkolai et al., 2022).

Merat és Lee (2012) szerint, a környezet érzékelésére, döntéshozatalra és jármű működtetésre képes összetett elektromos és mechanikai, azaz automatizált járművezetési rendszerek (ADS) beépítése gyakorlatilag elkerülhetetlen. Továbbá, szerintük a jelenlegi vezetés biztonságának javulása a rendszer és a vezető közötti kapcsolaton alapul. Kihívást jelent azonban meghatározni, milyen mértékben legyen aktív a rendszer, hol hibázhat egy adott ADS, mikor kell beavatkoznia a sofőrnek, és lesz e valaha teljes mértékben önvezető jármű.

A járművek önvezetésének fokait hat különböző szinten kategorizálta a Society of Automotive Engineers (SAE) szervezete által (SAE International., 2018), amelyek meghatározzák a jármű képességét a környezet érzékelésére, illetve a kézi beavatkozás szükségességét. Ezek a szintek magukban foglalják a teljesen kézi vezérlést is, amit a 0. szint reprezentál, ahol a vezetőnek teljes felelőssége van az autó irányításában (Collet és Musicant, 2019). Az automatizált, autonóm, önvezető és vezető nélküli fogalmak, SAE 1-5 szintekkel önállóan működő járműveket jelentenek. Nagyfokú Automatizáltságú Járműveknek (Highly Automotive Vehicles, HAV) tekintjük továbbá a 3-tól 5-ös SAE szintű járműveket, ahol már az ADS kerül előtérbe (Hyungjun Park et al., 2018).

A növekvő energiafelhasználás és járműszám potenciális problémája a mobilitásnak. A becslések azt mutatják, hogy teljes automatizáció esetén növekedni fog az utazás iránti igény, és a közlekedést használók száma, így az energiafelhasználás is. Mindkét problémára megoldást adhat a megosztott járművek szolgáltatás, mely egy autó több személy általi használatát jelenti (Catherine Ross et al., 2017).

Környezetvédelmi és infrastrukturális szempontokat figyelembe véve, megosztott teljesen automatizált járművek esetén Lokhandwala and Cai (2018) számításai szerint New York-ban

naponta mintegy 59%-kal kevesebb taxira lenne szükség, ami nagymértékben csökkentené a CO_2 kibocsátást. Fontos lenne továbbá az önvezető járművek különböző természeti környezetekre gyakorolt hatásainak vizsgálata, a létező átfogó tanulmányok mellett (Óscar Silva, 2022).

1.2. A kutatás célkitűzései

A kutatásom fő célja egy valós idejű pontfelhő-feldolgozó és szimulációs modul elkészítése, melyben az autó navigálása megtanítható megerősítéses tanulás segítségével. A lézeres letapogatókból származó pontfelhőt a szoftver gyorsan kell tudja feldolgozni, ezzel biztosítva a környező obiektumok detektálását és felismerését. A cél az autó navigálása egy létező utca pontfelhőjén, haladás közbeni valós idejű adatfeldolgozás az esetleges átkelő gyalogosok és autók figyelembe vételével. A szoftvert négy fő komponens fogja alkotni.

Az első modul a betöltő és kiszolgáló, amely képes a pontfelhőt bináris pontfelhő állományból beolvasni, majd a memóriában effektíven tárolni. Ezután a lekérdező és feldolgozó, ami az autó haladásával képes valós időben lekérdezni az autó környezetében lévő pontokat, azokat feldolgozni, domborzatmodellt illeszteni és objektumokat felismerni.

A következő az önvezető modul, amellyel az autó képes lesz kanyarodni, illetve gyorsítani és lassítani, az előtte található tereptárgyakat érzékelni, kikerülni. Mindezzel vezetői szokások lesznek megtaníthatók, mind például egy nyugodt, vagy dinamikus vezetés.

Az utolsó modul lesz a 3D megjelenítő, ami a szimuláció gyors térbeli megjelenítését végzi. Összességében a szoftver képes lesz nagy állományok gyors betöltésére, ezek feldolgozására, pontfelhőből domborzatmodell és objektumok kinyerésére, a szimulált autó vezetésére megerősítéses tanulással, és mindezek megjelenítésére.

1.3. Motiváció

A legfőbb motivációm a lenyűgöző önvezető technológia iránti érdeklődésem volt, mely utópikus képet tár elénk. Egy automatizált autókkal teli városban megosztott elven tökéletes az infrastruktúra terhelése, és balesetmentes a közlekedés. Eltűnnének olyan problémák, mint a közlekedési szabályok áthágása, agresszív és szélsőségesen defenzív vezetési attitűdök, a mobiltelefonálás és üzenetküldés vezetés közben, fáradtság miatti figyelemvesztés, és számos további biztonságot veszélyeztető probléma.

Az önvezető technológiához számos olyan szakterület kapcsolódik amik kiemelkedően érdekelnek, mint például a gépi látás és gépi tanulási módszerek, 3D megjelenítés, pontfelhő feldolgozás és a világunk digitális leképezése és azon keresztüli tökéletesítése.

2. Irodalmi áttekintés

2.1. Érzékelő rendszerek

Az autonóm autóipar gyors fejlődéséhez nagyban hozzájárul a szenzortechnológia széleskörű fejlődése és a különböző feldolgozó rendszerek. A külső érzékelők lehetővé teszik a környezet minél pontosabb érzékelését, míg belső érzékelők a jármű állapotáról, mozgásáról és orientációjáról nyújtanak fontos információkat (Sean Campbell et al., 2018; Shahian-Jahromi Babak et al., 2017).

2.1.1. Külső érzékelők

A kamerák passzív fényérzékelőket alkalmaznak a környezet digitális térképezéséhez, akár mozgó és statikus objektumok detektálásához. Megfelelő szoftverrel képesek a színek és textúrák felismerésére, ami nagy előny az önvezető járműveknél, mivel képesek útfelfestések, közlekedési táblák és lámpák azonosítására. Komplex feldolgozási algoritmusok is léteznek, amelyek segítségével akár távolságot is mérhetnek (Xiaoming, L. et al., 2010). A kamerák viszonylag olcsók, bár kevésbé hatékonyak rossz időjárás és gyenge fényviszonyok esetén.

A radar technológia rádióhullámokat használ különböző frekvenciákon, távolságok, szögek és sebességek mérésére. Az autóba épített radarok általában 50-100 méter hatótávolságúak, bizonyos típusok pedig akár 150 méterre is képesek. A radar szenzorok stabilak különböző környezeti körülmények között, főként ütközésdetektálásra és sebességtartásra használják, de képesek az objektumok relatív mozgásának meghatározására is (Rohling és Moller, 2008). A radarok további előnye a kamerákkal szemben, hogy időjárás- és napszakfüggetlenek. Hátrányuk, hogy viszonylag nagy a mérési zaj, amelyet komplex algoritmusokkal kell szűrni.

A lézer alapú távérzékelés (Light Detection and Ranging, LiDAR) egy költséges, de rendkívül megbízható, pontos technológia (több száz méteren is cm-es pontosság), többnyire időjárásfüggetlen, éjszaka is működik. Ezenkívül a nagy teljesítmény jellemzi, például egy Velodyne VLP16 (Velodyne LiDAR, 2018) szkenner képes 16 sávban, 5-20 hertzes frekvencián, 360°-on 2°os szögfelbontással néhány százezer pontot meghatározni másodpercenként, amelyet önvezető járművekben szívesen alkalmaznak (Veli Ilci et al., 2020). Kevésbé terheli zaj, mint a radaros érzékelőket és hatékonyabbak az érzékelő algoritmusok. Gyakorlatilag ez a lézeres változata a radar technológiának. (A LiDAR részletesen lesz tárgyalva a 2.2. alfejezetben)

Az ultrahangos érzékelők hanghullámokat használnak a távolság mérésére. Egy adott frekvenciájú hanghullámot küldenek az objektum felé, majd a hanghullám visszaverődésének időtartamát használják a távolság kiszámításához. Az ultrahangos érzékelők általában a legolcsóbbak, de sajnos hatótávolságuk kicsi és érzékenyek a rossz időjárási körülményekre.

2.1.2. Belső érzékelők

A GNSS (GPS, Glonass, Beidou, Galileo) globális navigációs műholdas helymeghatározó rendszer, amely elsősorban a jármű navigálására, térkép követésre, illetve földmérésben helymeghatározására, kitűzésre és térképezésre használnak. Pontossága néhány centimétertől néhány méterig terjed, kiépítéstől és mérési módtól függően.

Az inerciális mérőegység (Inertial Motion Unit, IMU) egy jármű gyorsulásának meghatározására szolgáló elektronikus eszköz. Az IMU-k széles körben alkalmazottak például navigációs rendszerekben, repülőgépekben, drónokban és autonóm járművekben ahol fontos az objektum mozgásának pontos követése és az irányváltások érzékelése.

Valamint az enkóderek, melyek a jármű kerekeire elhelyezett kerékelfordulásokat mérő szenzorok. Ezek a gyűjtött információk alapján számolják ki a jármű által megtett távolságot, illetve relatív pozícióját, azaz a jármű elhelyezkedését egy másik ponthoz vagy tárgyhoz viszonyítva. Lehetővé teszik a pontos navigációt és pozíciókövetést a jármű számára.

Ha az érzékelők összességét tekintjük, nagy előny, ha képesek vagyunk a különböző módon látott környezet adatainak összefésülésére, ezzel leküzdve az egyes szenzorok gyengeségeit.

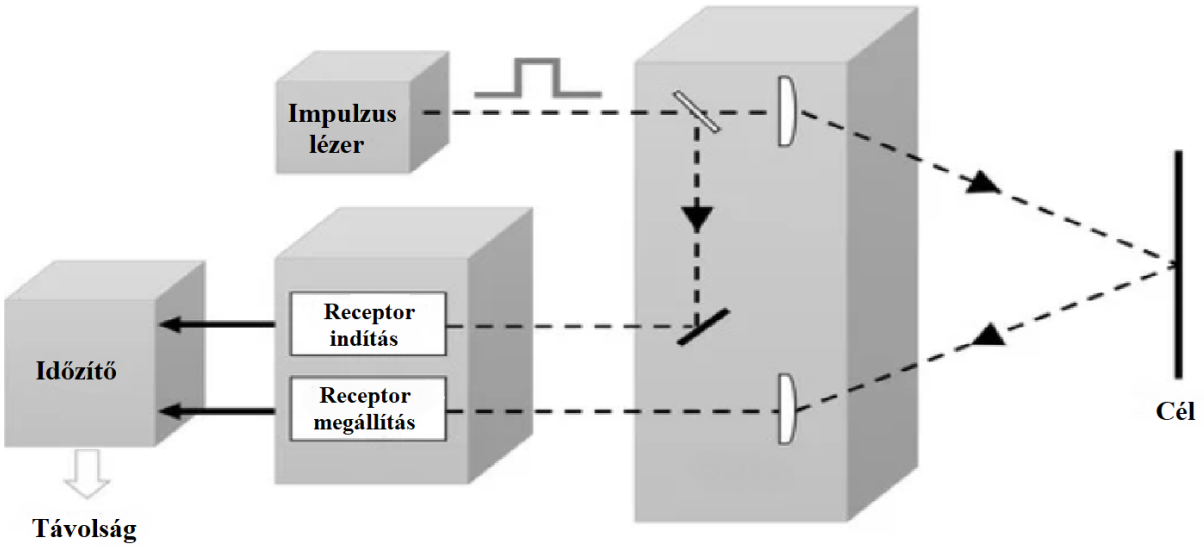
Vegyük például a Lidar és a kamerák közötti különbségeket. Míg az olcsóbb kamera nappal képes színek, textúrák és tárgyak mozgásának detektálására, a drágább LiDAR éjszaka és esőben is hatékonyabban határoz meg mélységeket, pontosabb méréseket nyújtva.

Ezért sokan különböző érzékelők ötvözését, azaz többérzékelős információösszefésülést javasolnak, amelynél többféle szenzor javíthatja az önvezető képességet (Peide Wang, 2021).

2.2. Lézeres letapogatás

A LiDAR fejlődése és a távolságok lézersugárral való mérése egészen az 1960-as évekig tekint vissza, alapelve a fény visszaverődésén alapul (2.2-1. ábra). A rendszer a lézersugarakat bocsájt a célpont felületére, majd méri a visszaverődött fény hullámhossz és érkezési idejének változását. A lézerfény sebessége és az eszköz gyorsasága lehetővé teszi a távolságok kiszámítását, elősegítve a komplex 3D környezet digitális reprezentálását. A LiDAR alapvető jellemzője, hogy a szenzor rendkívül pontos helyzetet szolgáltat egy adott pontnak, mely mennyiségi növelésével precíz képet kaphatunk az entitás alakjáról, akár nagyobb távolságból is.

A lézeres letapogató és a benne rejlő lehetőségek az önvezető jármű, optika, fotonika, geodézia, távérzékelés és számos más tudományterület és iparág egyik legfelkapottabb témája (Ninad Mehendale al., 2020). A környezet pontos detektálása nagy versenyt robbantott ki, melynek megnyerésére ideális lehet a LiDAR-al történő adatgyűjtés és az adatok 3D pontfelhőkként való értelmezése.



2.2-1. ábra: Lézer pályaidő mérési elve (forrás: Santiago Royo et al., 2019)

Az ökológiai érzékelők közül a LiDAR rendelkezik a legmagasabb mérési pontossággal, de magasabb költséggel is jár. Az autóipari LiDAR képalkotó rendszerek jellemzői a nagy hatótávolság és teljesítmény (pont/másodperc), magas mérési pontosság és napfénytől zavartalan működés. Ez a teljes autonómiához még kevés, de az előbbi jellemzők elérhetők a magas sebességgel forgó, kerék konfigurációs, több rétegben elhelyezett képalkotókkal (Santiago Royo et al., 2019).

Számos hordozóra lehet LiDAR eszközt illeszteni, például: földi (Terrestrial Laser Scanner (TLS), légi (Airborne Laser Scanner, ALS) repülőgépre, drónra, műholdra szerelt vagy mozgó (Mobile Laser Scanner, MLS) ami lehet kézi, hátizsákos, autóra, és robotkutyára szerelt. A letapogatók változó szögtartományban pásztáznak 1, 2 vagy 3 dimenzióban mérnek a különböző irányokba terjedő fénysugarak által. Visszaverődés alapján Rayleigh, Mie és Raman LiDAR-okat különböztetünk meg, ezenkívül beszélhetünk koherens fázisérzékenységen alapuló és inkoherens amplitudómérésen alapuló eszközökről (Ninad Mehendale et al., 2020).

Működési elvet, felépítést és képességüket tekintve, számos különböző típus létezik, akár gyártóktól függően (Santiago Royo et al., 2019).

LiDAR szenzorok esetében mechanikus forgóelemet alkalmaznak az optikai sugár pásztázásához, ami korlátozza megbízhatóságukat, növeli méretüket, de csökkenti az előállítási költségüket. Ezeket a szilárdtest megoldással lehet leküzdeni, melyek a félvezető nanotechnológiának hála olcsóbbak és könnyen tömeggyárthatók. A jövőben, a még kezdetleges szilárdtest LiDAR megújíthatja az ipart, köszönhetően a gyors sebességének, alacsony energiafogyasztásának, széles látómezejének és nagy felbontásának (Nanxi Li et al., 2022).

2.3. Pontfelhők

A pontfelhő egy olyan nagyméretű (big data) térinformatikai adatszerkezet, mely több 10 vagy 100 millió háromdimenziós pontot tartalmaz. A pontfelhőt leggyakrabban környezetünk lézeres letapogatásával vagy eltérő pontbók készült fényképek egyeztetésével állítunk elő. Pontonként a térbeli x,y,z koordináták mellett szín, intenzitás, időpont, és további kiegészítő adatokat tárolhatnak.

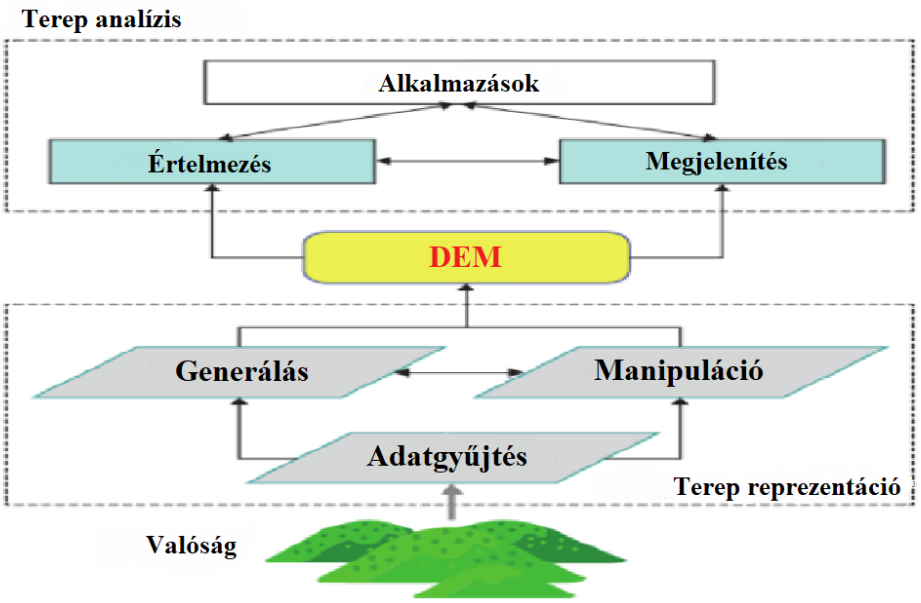
Egy pontfelhő egyik fontos jellemzője a pontsűrűség, mely a pontok számát jelenti négyzetméterenként (1 m^2 alapterületű hasábba eső pontok száma). A pontsűrűség függ a lézeres letapogató szögfelbontásától, hatótávolságától, és a teljesítményétől.

A térbeli környezetünket részletesen reprezentáló pontfelhőt több lépésben, komplex térbeli eljárásokkal dolgozzák fel (2.3-1. ábra), hogy térbeli objektumok helyzetét, méretét, mozgását határozzák meg belőle. Az algoritmusok első csoportja az elszórt, nem kívánt pontok eltávolításával indul. A jelenleg használt zajszűrő algoritmusok az alábbi kategóriába sorolhatók (Xian-Feng Han et al., 2017):

1. Statisztikai alapú szűrési technikák, mint Kálmán-szűrő autók pozíciójának meghatározására, becslésére (Yoganandhan, A. et al., 2020).
2. Szomszédság alapú szűrési technikák
3. Vetítés alapú szűrések
4. Jelfeldolgozáson alapuló módszerek
5. Parciális differenciálegyenletek (PDE) alapú szűrési technika
6. Hibrid szűrési technikák
7. Egyéb módszerek, mint a voxel alapú szűrés, amit 2.5. fejezetben ismeretek.

A feldolgozást a pontok osztályozása követi, ahol a pontfelhő pontjait különböző tematikus kategóriákba soroljuk, például talajfelszín pontok, növényzet, épület vagy járművek pontjai. Az osztályozás a pontok jellemzői, a környező pontok közelsége és attribútumai alapján történik. Az osztályozás számos speciális aggregáló funkciót használ, raszter és voxel technikát, térbeli indexelést, gráfokat, sűrűség modelleket.

A osztályozás után fontos a talajfelszín pontokra egy felületmodellt illeszteni (Peter Axelsson, 1998), hogy a felszín feletti objektumokat könnyebb legyen elkülöníteni. A felületmodellezés történhet raszteres vagy vektoros módszerekkel. A modell előállítása után speciális algoritmusokkal nyerik ki a felszín feletti objektumokat. A szűrés, osztályozás és modellezés lépéseit legtöbbször az adott feladat célja határozza meg.



2.3-1. ábra: Terep analízis (forrás: Qiming Zhou, 2017)

2.4. Domborzatmodell

A lézeres letapogatás által gyűjtött adatok feldolgozásának célja a különböző tárgyak precíz meghatározása, amiket feladattól függően fogunk felhasználni. Ehhez el kell különíteni a talajfelszín feletti pontokat, hiszen ezek fogják alkotni a lényegi tárgyakat.

Az elkülönített domborzatmodellre két különböző terminológia létezik a tudományágban:

- A digitális domborzatmodell (Digital Elevation Model, DEM) a puszta talajfelszínt jelenti, tereptárgyaktól, növényzettől és egyéb domborzat feletti objektumtól mentesen.
- Digitális felszínmodell (Digital Surface Model, DSM), mindent tartalmaz ami felszín felett van, de nem szeparálódik el a felszíntől. Épületeket, fákat igen, de egy repülőgépet például nem.

Különböző módszerek léteznek a domborzatmodell létrehozására, és adatmodellek a domborzat tárolására. Ilyen adatmodell például a raszteres, mely sorokba és oszlopokba rendezett cellákban tárolja a magasságokat. A másik gyakran használt adatmodell a vektoros háromszög háló modell, azaz adaptív TIN módszer (Axelsson, 2000; Dong et al. 2018), amely domborzat pontjai között szabálytalan háromszögeket hoz létre.

Qiming Zhou (2017) szerint hatékonyabb a raszteres megközelítés, mert könnyebb létrehozni, egyszerűbbek az interpolációs és feldolgozó algoritmusok, a raszter cella szomszédsági viszonyai egyértelműek. Mindezek nem mondhatók el a vektoros háromszögháló modellről.

A DEM előállításának első fontos lépése, a talajfelszínt definiáló pontok kiválogatása, amelyek elengedhetetlenek a domborzatmodell szempontjából. A második fontos lépés a

kiválogatott pontok interpolációja, azokra helyekre ahol nincsenek talajpontok. Xuelian Meng és társai (2010) adott területek legalacsonyabb pontjait a talajmeredekség küszöbértékével válogatják ki, és simítási értékkel interpolálják a felszínt.

A domborzatmodell előállítására számos megoldás született, például Kraus-Pfeifer (1997) sávos kiválogatása és átlagolása, mely egy raszteres felületmodellt állított elő a LiDAR pontokból. Ez a pontok magassági eloszlásának alsó tartományát választja ki és átlagolja iteratív módon. Morfológiai szűrővel válogatott pontokat Keqi Zhang et al. (2003). Evans és Hudak (2007) többszintű görbület vizsgálattal szelektálja a pontokat. Újfajta megközelítés a ruha szimulációs algoritmus, amely megfordítja a modellt és egy lepelt illeszt a így legfelülre került minimum pontokra a gravitáció segítségével. Ezután a lepelt fokozatosan finomítja újabb pontok bevonásával, amelyek a lepelhez közel vannak (Cai et al., 2019).

Nelson Diaz és társai (2021) pedig gyorsabb, valós idejű algoritmust találtak a talajfelszín pontok osztályozására a vektor alapú algoritmusnál, amelyet javasolnak pontfelhők jövőbeli szegmentálásához és osztályozásához. Az algoritmus voxel (Kaufman et al. 1993) elven működik, ami egy szabályos volumetrikus (térbeli) adatmodell, mely a teret egybevágó kockákra bontja.

2.5. Voxel kinyerés

A voxel a korai 3D játék rendering enginekből származik, ahol háromdimenziós térben a térfogatot (térrészt) kis egységekbe, vagy téglalatestekbe osztották fel a könnyebb számítás és megjelenítés érdekében. Felhasználásuk mára már megjelenik az objektumdetektálás opciójaként, szűrési eljárásoknál, vagy akár a neuropszichológiában MRI vizsgálat után, az agy struktúrájának háromdimenziós reprezentálásához (Sekerak, R., 2011).

Pontfelhők szűrésére létezik a voxel rács szűrő, amely a háromdimenziós pontfelhőket rendszerezi, a térben apró, egymás mellett elhelyezkedő 3D kockák létrehozásával, csökkentve a pontfelhő részletességét. Minden egyes 3D kockában, azaz voxelben, a pontokat a voxel középpontjával helyettesítik, ezáltal a a pontfelhő könnyebben kezelhetővé és kisebbé válik, lényegi információkat megtartva (Xian-Feng Han et al., 2017).

Ismert voxeleket használó algoritmus a VoxelNet, mely 3D objektumok detektálását végzi, kiemelkedve a jelenlegi modellek közül. Az algoritmus olyan újítást vezet be, mint a Voxel tulajdonság kódoló (Voxel Feature Encoder, VFE), amely voxelekre bontja a pontfelhőt és ezeket összefűzi. Magas számítási komplexitása és memóriaigénye miatt további fejlesztésre szorul (Alireza Ghasemieh et al., 2022).

Yuki Endo és társai (2023) pedig egy mély voxelizációs tulajdonságtérképet javasolnak, amely a jármű voxelekhez viszonyított pozíciójának meghatározásra szolgál. Az algoritmus hatékonyan kezeli a környezeti információkat és pontos pozíciómeghatározást biztosít. Az innovatív megoldás az önvezetés jövőjében nagy szerepet játszhat.

2.6. Döntéshozó algoritmusok

A döntéshozás az önvezetés utolsó, de egyik legfontosabb pillére, ahol a jármű a gyűjtött vagy kiszámolt adatok alapján továbbhalad, gyorsít, megáll, kikerülést vagy egyéb fontossággal bíró műveletet kísérel meg. Ezzel a résszel főleg klasszikus szabályalapú, optimalizációs eljárásokkal, vagy valószínűség alapú, illetve a tanulásalapú módszerek foglalkoznak (Qi Liu et al., 2021).

Az első csoportból érdemes megemlíteni a véges állapotgépeket, melyek egyszerűen értelmezhetőek, de komplex vezetési döntéshozatalok sorozata kivitelezhetetlen bennük. Az optimalizációs eljárások főleg jutalom vagy hasznosságfüggvény alapúak, mint például a játékelméletben a stratégiaoptimalizálás egyéb ügynökök döntései alapján. A valószínűségi metódusok könnyen kombinálhatók más módszerekkel, de összetett helyzetekben a valószínűségek sokasága miatt nem ad optimális döntést.

A második csoportba tartozik a gépi tanulás, mely problémaspecifikus adatokból tanul, automatizálja az analitikus modellépítés folyamatát és az ehhez kapcsolódó feladatok megoldását (Janiesch, C. et al., 2021). Más megfogalmazásban a gépi tanulás a mesterséges intelligencia része, amely a gépeket a tanulás képességével ruházza fel ezáltal adatvezérelt döntési algoritmusokat hoz létre (Hyungjun Park et al., 2018).

Részhalmazuk a mesterséges neurális hálózatok (Artificial Neural Network, ANN), melyek nemlineáris statisztikai adatfeldolgozó eljárások. A biológiai neuronok szerkezete és funkciója inspirálta őket, mintha neuronok egymásnak adott jelzéseit utánoznák az emberi agyaban. Mintakeresésre és komplex bemenet-kimenet kapcsolatok modellezésére használják (Anastasius S. Moumtzoglou, 2019).

Döntéshozatal tanulás alapú módszerei (Qi Liu et al., 2021):

- Statisztika alapú tanulás, nagy statisztika alapú adatokkal tanított ANN, de alkalmas egyszerűbb esetek megvalósítására. Ezeknek a hálóknak döntéshozatalt tekintve magasabb lehet a hibaaránya.

- Mélytanulás, egy gépi tanulási koncepció, amely mesterséges neurális hálókon alapul. (Janiesch, C. et al., 2021). Magas döntéshozatali pontossággal rendelkeznek, azonban nagyban függenek az adathalmaz minőségétől.

- Megerősítéses tanulás (Reinforcement Learning, RL, Watkins, 1989), ahol az algoritmus tanulja meg az optimális utat, szekvenciális döntéshozatali problémák megoldásához, a környezettel való interakció során.

A RL modellek a legmegfelelőbb cselekvést választják egy adott helyzetben annak érdekében, hogy maximalizálják a megadott jutalmat (Salim Dridi, 2022). A legjobb modellezése a bizonytalan és változó körülményű környezetnek, viszont bizonytalan struktúra jellemzi, mert megtanulhat nem kívánt viselkedést is.

Egy újabb kategória a mély megerősítéses tanulás, mely mélytanulási és megerősítéses tanulási modellek kombinációja (Kai Arulkumaran et al, 2017). Ezek robosztusabbak, komplexebb adathalmazok tömör, alacsony dimenziós reprezentációit is lehetővé teszik, elősegítve a megerősítéses tanulás gyorsabb konvergenciáját.

Qi Liu és társai (2021) szerint az autóipart érintő főbb döntéshozó algoritmusok felhasználása a jármű-gyalogos interakciók, a biztonságos döntéshozó rendszerek és bonyolult környezetet kezelő rendszerek, vagy ezek fúziója .

Asanka Wasala és társai (2019) bemutattak egy új megerősítéses tanuláson alapuló oldaltartás módszert, amelynek során az ügynök meghatározott útvonalakat kapott bemenetként, majd ezen alapulva adta ki a kormányzás parancsait. Újdonsága, hogy az ügynök képes volt hatékonyan irányítani egy korábban nem látott járművet, akár egy ismeretlen pályán.

A gépi tanulás számos új lehetőséget és új problémákat is adott az önvezető autók fejlődéséhez. A gép már képes értelmezni egy közlekedés helyzetet, kiszűrni a zajokat, fontossági sorrendet felállítani, például egy utcára rohanó gyerek, egy közeli autó és egy benzinkút között. Sajnos ez még nem jelenti azt, hogy versenyre kelhet az emberi döntéshozatallal.

2.7. Önvezető technológia

A vezetés egy összetett folyamat, amely magában foglalja a jármű környezetének átfogó érzékelését, a tervezett útvonal megtételét a kiindulópontból a célpontig, valamint a jármű működtetését és irányítását, így nem is meglepő az önvezetés komplexitása. Ahhoz, hogy megbízható és intelligens járműveket tudjunk gyártani összetettebb kérdéseket is figyelembe kell vennünk, mint a gyalogosok viselkedése, véletlenszerű tárgyak felbukkanása és detektálása, illetve a forgalomban történő döntéshozatal.

Az önvezetés három fázisa az érzékelés, fúzió és döntéshozatal, amiből az első a jármű adatgyűjtése különböző környezeti érzékelők, szenzorok segítségvel. A másodikban összesíti a begyűjtött adatokat a környezet modelljének létrehozásához, illetve a harmadik a jármű döntése a továbbhaladásról. (Hyungjun Park et al., 2018)

Jelenleg kulcs technológiák közé tartoznak a 3D LiDAR és a képfeldolgozó eljárások, mint például az átlagpontos klaszterezés, gépi tanulási módszerek, útvonaltervező eljárások, sávtartás, döntési algoritmusok, ezen belül minőségi tanulás (Q-learning) és szürke előrejelzési modell (Gray Prediction Modell), melyeket továbbiakban ismertetek (Darsh Parekh et al., 2022).

A szenzorok nagy mennyiségű adatot továbbítanak, amelyeket csoportosítani és rendszerezni kell. Ebben segítenek a klaszterezési eljárások, amelyek többváltozós adatok hasonlóság vagy eltérésen alapuló nem felügyelt osztályozását végzik (Patrizia Firman, 2018). Közülük a legnépszerűbb az átlagpontos klaszterezés, amely az adatpontok és a klaszter középpont között a négyzetes hibát minimalizálja (Abiodun M. Ikotun, 2022). Összetett vezetési

körülmények mellett, bonyolult akadályokat közel lehetetlen meghatározni, ezért Wuhua Jiang et al., (2023) egy új eljárást javasoltak, melynek neve adaptív szomszédsági klaszterezés. A módszer javítja a pontfelhők térbeli egyenlőtlenségeit egy lapozási szög bevezetésével, amely hatékonyabb akadálydetektálást eredményezett.

A múltban számos útvonaltervezési eljárás született, azonban nemrég egy új gyors felderítésű véletlenszerű fa (Rapid Exploration Random Tree, RRT) algoritmust publikáltak (Yong Zhang et al., 2023). Ez véletlenszerű mintavételezést alkalmaz az útvonal kiterjesztéséhez a kiindulási ponttól a célpontig. Hatékonyabban talál útvonalakat, mivel minimalizálja a költséget, és így javítja az autonóm járművek útvonalkeresését.

Stabilabb sávfelismerést eredményezett a geometria modellek és figyelem mechanizmusok bevezetése, utóbbi a gépi tanulás és a természetes nyelvfeldolgozás (Natural Language Processing) kulcseleme (Noor Jannah Zakaria et al., 2021). Taewon Ahn és társai (2021) már létező sávkövetési eljárásokat kombináltak csökkentve a jármű oldalirányú és kormányzási hibáit. Az irányító rendszer nagyban elősegíti az autonóm járművek stabilitását és pályakövetési képességét, mely nagyban kapcsolódik az autók biztonságos közlekedéséhez. Autonóm járművek sávváltásához használható a GPM, mely környező járművek mozgását próbálja kiszámítani. Xiaodong Wu et al. (2020) egy egyszerűsített GPM-et fejlesztettek, amely a megfelelő időablakot keresi a sávváltáshoz.

A vezeték nélküli kommunikációs eszközzel rendelkező, más járművel, infrastruktúrával, utasokkal vagy felhővel kommunikáló összekapcsolt járművek (CV, Connected Vehicles) (Hyungjun Park et al., 2018) technológiája jelentős előrelépést mutat a rohamosan növekvő forgalom és útkereszteződések optimalizálásában. Működésük lehet jármű-infrastruktúra (V2I), jármű-jármű (V2V) és különböző közlekedési eszközök, jármű-x (V2X) közti kommunikáció. Ha minden jármű összekapcsolt lehetne, az egy globális optimumot eredményezne (Alireza Ansariyar, 2023; Fayez Alanazi, 2023).

Hatalmas a verseny a teljes autonómia megvalósítására. Ez érinti az önvezető autókat vagy más autonóm járműveket földön, levegőben vagy vizen. Santiago Royo és társa (2019) szerint az ideális megoldás a LiDAR, radar, videokamera és mély tanulási eljárások kombinációja lehet.

Végezetül fontos megjegyezni az önvezető járművek piaca exponenciálisan növekszik, ezért számos új technológia láthat napvilágot ezen a területen. (Precedence Research, 2023).

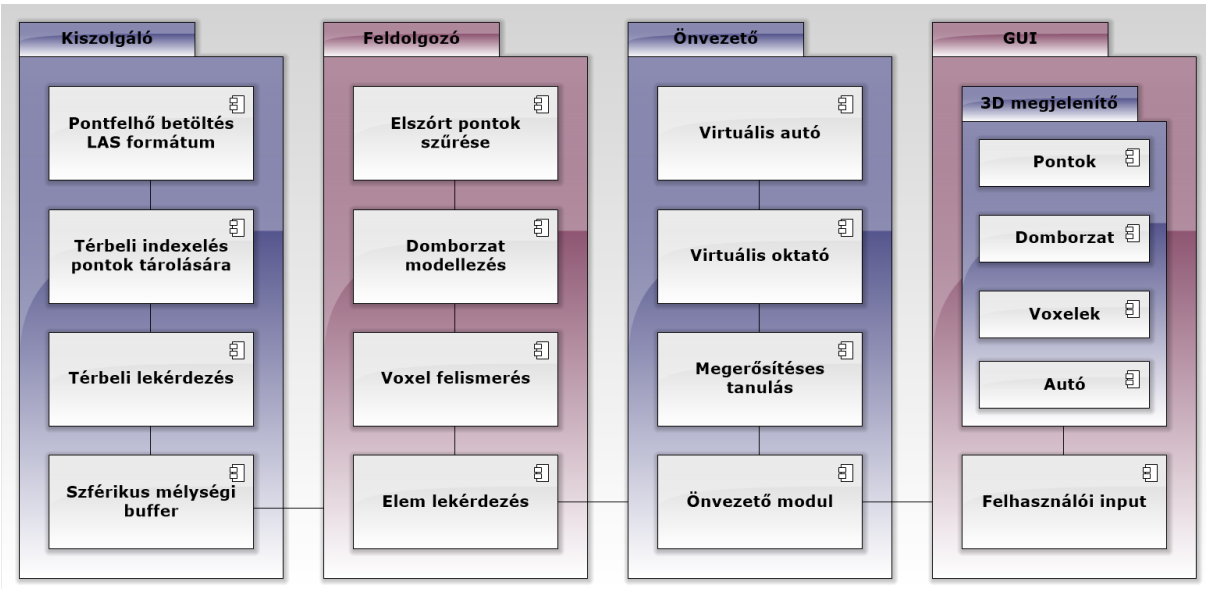
3. Anyag és módszer

Ebben a fejezetben ismertetem a program tervezési lépéseit, felépítését, az alkalmazott módszereket, algoritmusokat, és bemutatásra kerülnek az újonnan fejlesztett, vagy továbbfejlesztett eljárások, úgymint az új RP-Tree térbeli indexelés, a szférikus mélységi puffer, a továbbfejlesztett morfológiai szűrő. Végül bemutatom a szoftver két fontos elemét, a paraméterezhető virtuális oktatót és a megerősítéses tanulást használó önvezető modult.

3.1. Programtervezés

A programtervést a komponensek definiálásával kezdtem, hierarchikus dekompozícióval. Meghatároztam, hogy melyek az önvezető szoftver elkülöníthető nagyobb, majd kisebb moduljai, és ezek hogyan kommunikálnak egymással. A program négy fő komponense a következő (3.1-1. ábra):

1. Pontfelhő betöltő, kiszolgáló modul
2. Pontfelhő-feldolgozó modul
3. Önvezető modul
4. 3D grafikus felhasználói felület



3.1-1. ábra: Az önvezető szoftver komponens diagramja

A kiszolgáló modul első eleme a pontfelhő betöltő. Második komponense a betöltött pontokat indexelt térbeli adatszerkezetben tárolja. A modul harmadik része pontokat kérdez le, és negyedik eleme a szférikus mélységi puffert valósítja meg, amely az autóra illesztett lézeres letapogatót szimulálja.

A pontfelhő-feldolgozó modul delegált interfészként kapja meg a kiszolgálót, amelyből a adott körzetben lévő pontokat kérdezi le és dolgozza fel. A feldolgozási modul az elszórt pontok szűrését, a domborzat modellezését és a voxelfelismerést végzi. A modul a feldolgozott, detektált elemek eléréséhez szükséges funkciókat is tartalmazza.

Az önvezető modul tartalmazza a virtuális autót, valósítja meg a tanítást, melyet egy paraméterezhető virtuális oktató biztosít, és itt kap helyet a megerősítéses tanulás alapú önvezető komponens, amely a szakértő elven működő virtuális oktatótól tanul.

Grafikus felület alkalmas az eredeti és a feldolgozott pontfelhő, a virtuális környezetben előbb az oktatóval, majd az önvezetéssel közlekedő jármű térbeli megjelenítésére. Ezen túl ez a modul fogad felhasználói billentyű és egér parancsokat, amellyel a 3D nézőpont és az önvezető szoftver működése vezérelhető.

A hierarchikus komponens tervezést követték az egyes modulokat, almodulokat megvalósító osztályok kidolgozása, a belső változók, a tagfüggvények és a függvény paraméterek megadása, valamint az osztályok közötti kapcsolatok, egyszeres és többszörös tartalmazás, öröklődés, interfész definíciók meghatározása. A program fejlesztése során az osztály tervezési fázishoz gyakran visszatértem, mikor egy újabb tagfüggvényre, vagy paraméterekre volt szükség, vagy átalakításokra, osztály leválasztásra, refaktorálásra.

Az önvezető szoftver C++ programnyelven, Visual Studio környezetben valósítottam meg.

Készítettem egy saját string osztályt a az openai ChatGPT szoftvere (ChatGPT, 2023) segítségével kísérlet képpen, hogy teszteljem a C++ forráskód író képességeit. A GPT 3.5 által készített kódot ellenőriztem, az osztály gördülékenyen és hibamentesen működik.

A 2022/23. tavaszi félévében C++ házi feladat keretében az itt ismertetett szoftverből bizonyos modulok már elkészültek, nevezetesen a pontfelhő betöltő, a domborzatmodell és voxel előállító, valamint ezek OpenGL megjelenítője. Az elkészült modulok továbbfejlesztésével, új modulok fejlesztésével, új funkciók megírásával készült el mostani önvezető szoftver.

A program számos Unit tesztel rendelkezik, tesztelve a pontfelhő betöltést, feldolgozást, domborzatmodell készítést, voxelek felismerését, az önvezető modult és számos kisebb komponenst.

3.2. Mintaterület

Az önvezető szoftver és eljárások fejlesztéséhez egy városi környezetben kézi lézeres letapogatással készült pontfelhő állományt választottam, melyet a DendroComplex Kft. bocsátott rendelkezésünkre.

A pontfelhő állományt LAS adatformátumban kaptuk meg. Ezt a formátumot lézeres letapogató hardverek és feldolgozó szoftverek használják, amelyek összekapcsolják a GNSS, IMU és lézerimpulzus távolságadatait, létrehozva térbeli pontfelhőt. A LAS formátum egy nyílt, szabványos formátum pontfelhők tárolására, továbbítására (American Society for Photogrammetry & Remote Sensing, 2011).

A dolgozat mintaállománya a GeoSLAM Zeb Horizon (GeoSLAM, 2023) lézeres letapogatóval készült 2020. 06. 19-én. Az állomány Budapest VIII. kerületi Kálvária terének felmérését tartalmazza, mérete 1,7 GB, mely 70 millió pontot tartalmazó pontfelhőt jelent.

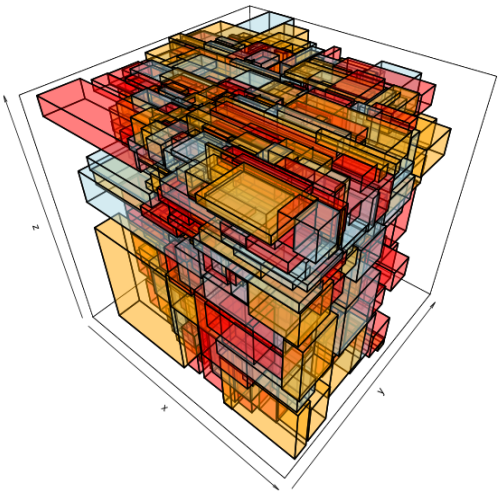
3.3. Pontfelhő kiszolgáló

A fejlesztett program első számú modulja a pontfelhő kiszolgáló, mely a pontfelhő állományok betöltését, indexelését és lekérdezését valósítja meg. A kiszolgáló LAS bináris formátumú állományokban tárolt pontfelhőket tud betölteni. Megnyitás után betölti a fájl fejrészét, azonosítja a formátumot, értelmezi a pontfelhő jellemzőket, majd elkezdi egyesével a pontok betöltését és tárolását.

3.3.1. Térbeli indexelési módszerek

Geoinformatikában gyakran használt indexelési eljárás az R-fa (R-Tree, 3.3.1-1. ábra) index (Guttman, 1984). Ez az eljárás kétdimenziós, kiegyensúlyozott fát hoz létre, de nagyszámú pontok tárolására nem igazán hatékony, mert a befoglaló koordinátákat (x1, x2, y1, y2, z1, z2) is tárolja. Az R-fában a tárolók telítődése esetén a pontok két új tárolóba sorolása időigényes ($O(n^2)$). A kettéválasztás akkor hatékony, ha tárolók térbeli átfedése minimális.

Pontfelhők indexelésére gyakran használt eljárás a nyolcasfa módszer (Donald Meagher, 1980), mely a teret 8 egyenlő részre osztja hierarchikusan. Itt nem kell tárolni a pontok befoglalóját, a megosztás gyors, de nem kiegyensúlyozott a fa, változó pontsűrűségű állományt nem tud hatékonyan tárolni.



3.3.1-1. ábra: Térbeli R-fa illusztrációja (David Moten, 2022)

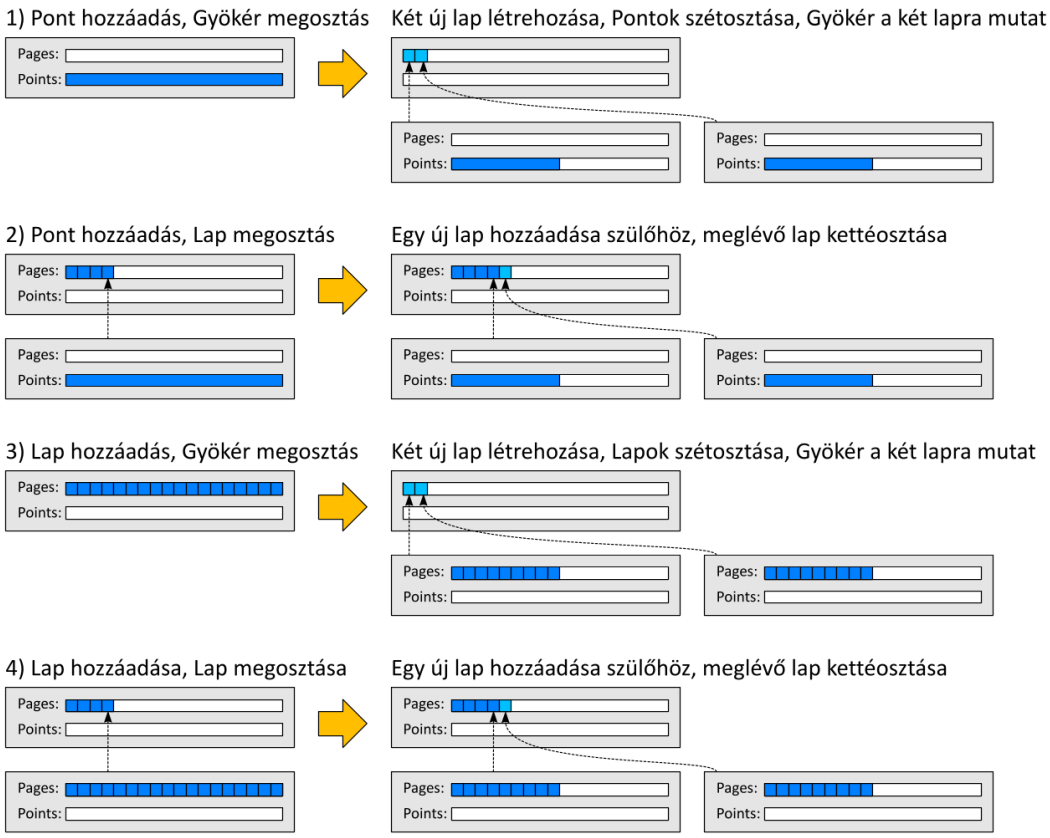
3.3.2. C-Tree-P térbeli indexelés

A pontok indexelt tárolásához egy új adatszerkezetet és algoritmust fejlesztettem, mely az R-fa továbbfejlesztésének tekinthető. Térbeli pontok indexelésére szolgál, nem tárolja a pontok befoglaló koordinátáit, de a tárolókét már igen, ezért helytakarékosabb, mint az R fa. Ez is egy kiegyensúlyozott fa, mely gyors tároló algoritmussal rendelkezik, és a tárolók térbeli átfedéseit minimalizálja.

Az új térbeli indexelési eljárás neve téglatest fa-pont indexelés (Cuboid-Tree for Points, C-Tree-P). Az eljárás bemenete a térbeli pontok, amelyeket hierarchikus téglatest tárolókban helyez el. Kimenete egy kiegyensúlyozott adatszerkezet, amely gyorsan, a bináris kereséséhez hasonlóan képes térbeli befoglaló koordináták alapján a befoglalón belüli pontokat leválogatni.

A téglatestek előre megadott számú pontot tárolhatnak. Egy pont hozzáadásakor a legfelső tárolóból kiindulva próbálja elhelyezni abban a tárolóban, hierarchikusan lépkedve lefelé, ahol a legkisebb a térfogat növekedés a hozzáadással. Ha egy tároló vagy lap megtelik, akkor az eljárás két új lapot hoz létre és a pontokat a súlypont alapján, a legnagyobb kiterjedés mentén (x vagy y vagy z tengely mentén) két részre osztja, és a két új tárolóhoz hozzáadja. A szülő tároló átalakul és a két újonnan létrehozott lap hivatkozását tárolja (3.3.2-1. ábra). Ha az előbbi két pont tárolóból az egyik megtelik, akkor egy új lap jön létre, a pontok kettéosztása itt is végbemegy, és az új tároló referenciája hozzáadódik az legfelső tárolóhoz. A hivatkozások száma egy tárolóban is előre megadható. Ha betelik, akkor ebben az esetben is két új tároló jön létre, és a szülő fogja tárolni a két új lap referenciáját. A tárolók kitöltöttsége 75% körüli, hasonlóan az R-fa indexeléshez. Jelenleg 64 lap vagy 8192 pont

lehet egy tárolóban. Az indexelés több szinten történik, az előbbi méretekkel (64, 8192) három szinten akár 33 millió pont tárolható.



3.3.2-1. ábra: Téglatest fa indexelés 4 alapesete

3.3.3. Szférikus mélységi puffer szűrés

A program fejlesztése közben gondolkodtam el azon, hogy területet egy kézi lézeres letapogatóval vették fel, ami eltérő pontfelhőt eredményez, mint az autókra szerelt. Ugyanis a kézi letapogató minden irányban mér és a mozgás következtében a tereptárgyakat (fák, autók, épületek) több irányból is felveszi. Ez az autóra szerelt szenzorokra nem jellemző, mivel ezek egy adott irányban, mind vízszintesen, mind függőlegesen szűkített tartományban érzékelnek.

Azért fejlesztettem egy új, innovatív módszert a pontfelhő szűrésére, hogy az autókra helyezett fedélzeti LiDAR-ok érzékelését jobban tudjam szimulálni. A módszer neve szférikus mélységi puffer szűrés (spherical Z-buffer filtering). Előnye, hogy nem kellett speciális érzékelőket rögzíteni autókra, és járművel történő közlekedéssel több letapogatást végezni, hanem mindezt virtuális környezetben lehet modellezni.

Az eljárás lényege, hogy a térbeli (x, y, z) koordinátákból, gömbi koordinátákat (φ, λ, d) számolok a következő képletekkel:

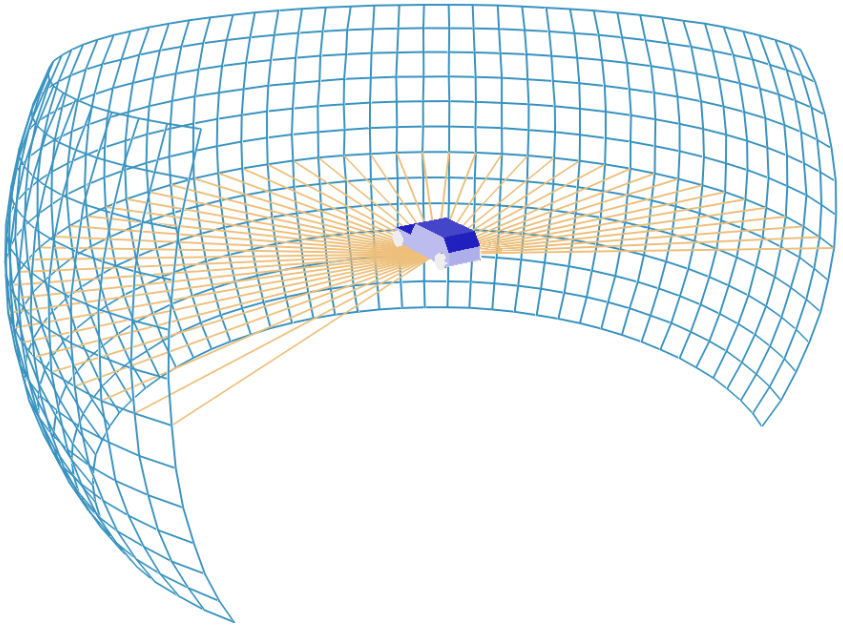
$$d = \sqrt{x^2 + y^2 + z^2}$$

$$\varphi = arctan2(x, y)$$

$$\lambda = arctan2\left(z, \sqrt{x^2 + y^2}\right)$$

A koordináta transzformáció után a φ, λ koordinátákból az eljárás a mélységi puffer felbontásának megfelelően egész cella koordinátákat számol. Az így kiválasztott cellában tárolja a középponthoz legközelebbi pont térbeli koordinátáit, melyeket a lekérdezés után átad a kiszolgáló hívója felé egy referenciaként átadott pont vektorban.

Ez az eljárás hatékonyan tudja modellezni kézi lézeres letapogatóval felvett pontfelhőből a virtuális gépjárműre illesztett lézeres letapogatót, megadott szögfelbontás mellett a legközelebbi látható térbeli pontokat adja át (3.3.3-1. ábra).



3.3.3-1. ábra: Szűkített tartományú szférikus mélységi puffer

Az eljárás továbbfejleszthető, hogy megadott horizontális és vertikális szögtartomány között működjön, például az autó előtti 60 fokos vízszintes látószögben és 40 fokos függőleges látószögben. Az algoritmussal az is szimulálható, hogy letapogatási sávokat modellezzen, például 0-5 fok között, majd 10-15 fok között és így tovább, lásson csak pontokat, ahogy a valós fedélzeti lézeres letapogatók is működnek.

3.4. Pontfelhő-feldolgozó

A pontfelhő-feldolgozó a kiszolgálóból kéri le az autó pozíciója közelében lévő, és az autó által éppen látható (azaz nem takarásban lévő) pontokat. A kiszolgáló ehhez a térbeli lekérdezőt és a szférikus mélységi puffert használja. A pontfelhő-feldolgozó számos belső adatszerkezettel és tagfügyvénnyel rendelkezik az objektumok kinyeréséhez.

A feldolgozás legfontosabb eleme, ahogy a szakirodalomban is szerepel (Peter Axelsson, 1998), a talajpontok kiválasztása, és a kiválogatott pontokra domborzatmodell (DEM) illesztése. A domborzat modell kinyerése azért fontos, mert ezen közlekedik az autó, és a modell feletti tereptárgyakat kell figyelnie az önvezető modulnak. A domborzatmodellt az autó előrehaladtával dinamikusan, minden egyes elmozdulás után valós időben újra előállítom.

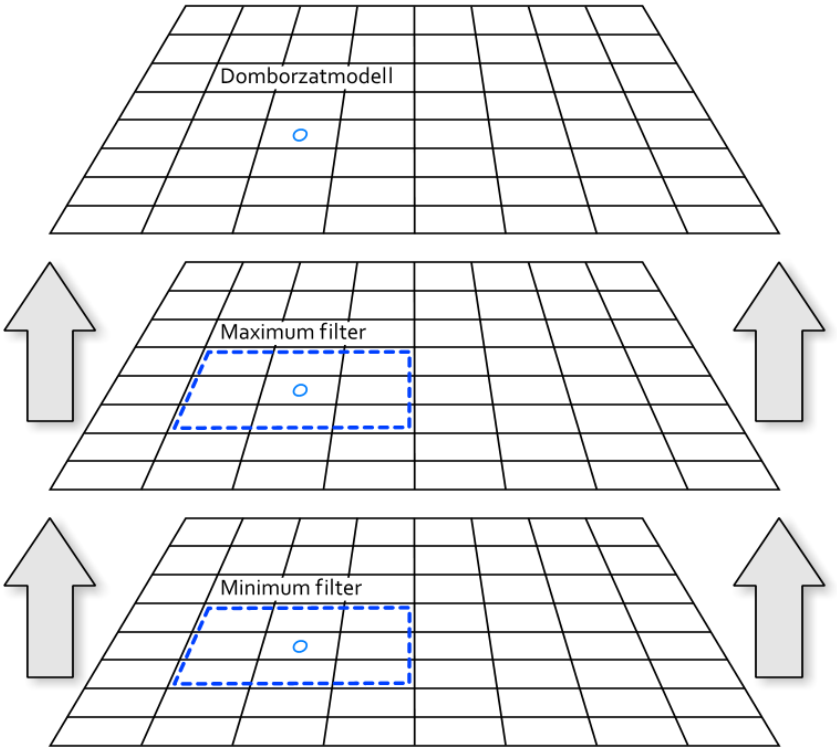
Az általam fejlesztett algoritmus egy megadott felbontású, a pontokkal azonos kiterjedésű raszteres felületmodellt hoz létre. Minden raszteres cellába kigyűjti pontfelhőből a legalacsonyabb pontokat majd elkezdi a pontok szűrését.

Az szűrés első eleme egy speciális minimum szűrő, mely a pixel magasságát és környezetének minimumát hasonlítja össze. Ha a pixel alatta van a környezet minimumának, akkor a pontot törlöm, mivel nagy valószínűséggel ez egy elszórt pont, amelyet általában víz vagy tükröződő felület okoz.

Ezután egy módosított morfológiai, azaz raszteres geoinformatikában használt fokális (pixel és közvetlen környezetében működő) szűrőt alkalmazok a talajpontok szűrésére. A morfológiai szűrő egy minimum szűrővel indít, minden egyes pixel négyzetes környezetében erodál, rombol, csökkenti a felületet, a környezet minimumával helyettesíti a pixel értéket. Ezután egy maximum szűrővel folytatom a már erodált rasztert dilatálja, építi, a pixelt a környezet maximum értékével helyettesíti. Az erodálás és dilatálás egymás utáni futtatása a kiugró elemek, tereptárgyak (fák, bokrok, épületek, járművek) leradírozására szolgál. Az általam továbbfejlesztett morfológia szűrő egyedi (3.4-1.ábra). A módosítás abban rejlik, hogy a négyzetes pixelszám adott százalékát el kell érnie a nem üres pixeleknek, hogy értelmes eredmény születhessen. Ezzel képes a szűrő a szkennelt terület szélén lévő anomáliákat eltüntetni, például felemelkedéseket és völgyeket.

A szűrések által létrejött minimum felület tartalmazza azokat a talajpontokat, melyek interpolálhatók az üres pixelekre, és ezzel a domborzatmodell előáll. Interpolációra negyedfokú kernel súlyozást használtam (Altman, 1992), mely könnyebben implementálható, mint a Gauss simítás, és határozott széle van, adott sugár fölött nulla értéket vesz fel. A kernel a környező pontok magasságának súlyozott átlagát számolja ki, hogy becsülje a középpontból egy adott pont magasságát egy adott sugáron belül. A középpont magassága nem közvetlenül mérhető, de a környező pontokból becsülhető. Ha a súlyok összege nulla, akkor a visszatérés is nulla lesz, ellenkezőleg a súlyozott magasság értéke. Ezzel pontos becslést kaphatunk a domborzat magasságára amiből előáll a domborzatmodell. A modell

előállítása elengedhetetlen a talajfelszín, útburkolat feletti pontok elkülönítését, amiket a vezetés szempontjából észlelni kell.



3.4-1. ábra: Domborzatmodellezés elve morfológiai szűrőkkel

A következő lépés a voxeles modell felállítása és feltöltése a pontok alapján. A voxeles adatszerkezet térkockákban számolja a pontokat, ezzel térképezi a pontsűrűsödéseket, a térbeli objektumokat. Egy voxelben maximum 255 pont előfordulást engedek, ezáltal egy voxel helyigénye 1 byte, amiben a voxel belső pontjainak számát tárolom. 255-nél több pont esetén a voxel értéke 255 lesz. A voxel, térkocka oldal mérete megegyezik a domborzatmodell síkbeli felbontásával, ezáltal x és y irányú cella mérete azonos a domborzatmodellel, z irányban pedig 8 cella magas a szerkezet, amely cellánként 8 bájton ábrázolható. Ez 0,5 méteres felbontásnál 4 méteres voxel modellt jelent.

A pontokat voxelekbe sorolom, majd ha ezek kisebbek egy adott szűrési értéknél, akkor a voxelt törlöm, ezzel megtartva a tényleges pontsűrösödéseket. Az elevetés oka, hogy az alacsony pontszámú voxelek általában zajt reprezentálnak, vagy nagyon kis méretű objektumokat, például a vezetés szempontjából nem releváns kismadarat. Ezért a voxel a detektáláson kívül egyben szűrőként is funkcionál, a lényegi objektumokat emeli ki, a vezetés szempontjából elhanyagolható pontokat törli. A voxeles adatszerkezet 3D szabályos felépítése olyan előnyöket rejt, mint a gyors szomszédsági viszonyok elemzése, melynek köszönhetően a szomszédos voxelek összekapcsolhatók. Ezekből térbeli testek, például autók, oszlopok, fák, épületek ismerhetők fel nagy hatékonysággal.

3.5. Oktatott megerősítéses tanulás

Az önvezető komponens áll egy virtuális autóból, egy oktatóból, és oktatótól megerősítéses tanulással tanuló önvezető modulból. A virtuális autó tartalmaz minden paramétert, ami az autó pozícióját, irányát, sebességét és gyorsulását leírja. Az oktató a környezet felmérése, a voxelek közelsége és az autó paraméterei után szakértői döntéseket hoz. Ezekből a döntésekből tanul a önvezetést végző modul megerősítéses tanulással, amely megfelelő számú tanulási iteráció után önállóan vezeti a virtuális autót.

3.5.1 Virtuális oktató

A vezető emberek tudhatják, hogy mindenki más vezetési stílust követ, valaki óvatosabb, körültekintőbb, csökkentett sebességgel halad, más valaki sportosaban, lendületesebben, de még a szabályok betartásával vezet (például egy taxi), vagy esetleg környezettudatosabb (mérsékelt gyorsítások és lassítások), ezért érdemes önvezető autókba különféle vezetési stílusokat beépíteni. Egy óvatos, vagy megfontolt vezető óvatosan közlekedne, sokkal korábban kezdene el fékezni egy egyirányú utcában, mint például a pontfelhő mintaterület, egy kilépő gyalogos észlelése esetén, vagy a jobbkézszabály miatt a kereszteződés előtt. Emellett autópályákon, ahol ilyenekre minimális valószínűséggel lehet csak számítani, dinamikusabb vezetésre tudna kapcsolni.

Ezek figyelembevételével létrehoztam egy virtuális oktatót, melynek célja a közlekedési helyzetek szakértői kezelése és a következő részben bemutatott megerősítéses tanuló tanítása. Azért is használjuk a virtuális oktató, mert az önvezető modul tanításhoz sok idő kellene, sok iteráció, hogy kialakuljon a döntési mechanizmus. A virtuális oktató előfeldolgozza az adatokat, segíti megerősítéses tanulás jutalmazási és a büntetési fázisát és gyorsabb konvergenciáját.

Az oktatót különböző paraméterek definiálják, például maximális sebesség lakott területen és azon kívül, távolságtartás, oldaltartás, gyorsulás, lassulás mértéke és megkezdése, kanyarodási sebesség, ezáltal számos oktatási viselkedés definiálható.

A virtuális oktató a paraméterek alapján végigvezeti az autót a pontfelhőn, ugyanazt a detektálási elvet használja, mint a megerősítéses tanulás, de szakértőként kezeli le a vezetési helyzeteket, ez szolgál bemenetként a megerősítéses tanulás számára. A virtuális oktató nagy előnye, hogy felkészül a várható szimulált közlekedési helyzetre anélkül, hogy tudná az akadály jellegét. Voxelként észleli az előtte lévő akadályokat, megpróbálja azokat kikerülni, ha nem tudja, akkor lelassít. A voxelek közelsége és iránya alapján szakértőként reagál, és elkezdi a kanyarodás, fékezés akciókat, ha nincs akadály, akkor pedig a gyorsítást. Az önvezető ezekből a műveletekből tanul, a voxelek iránya és közelsége jelenti a környezet állapotát, a virtuális oktató műveletei pedig az akciókat.

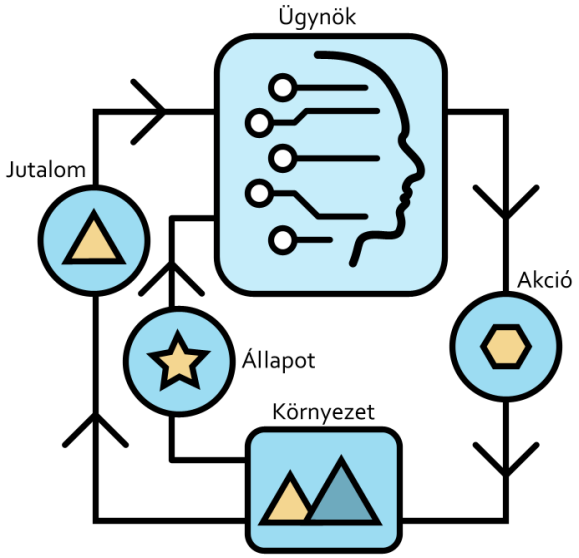
Az egyes virtuális oktatók eltérő módon kezelik a szimulált közlekedési helyzeteket. Egy sportos vezető hirtelen gyorsít, a maximális megengedett sebességgel közlekedik és gyorsan lassít, ezáltal a legrövidebb idő alatt éri el a célját. Egy energiahatékony vezető ellenben

fokozatosan gyorsít és időben elkezd fékezni, gurulni és az akadály előtt megállni. Az óvatos vezető szintén fokozatosan gyorsít, a megengedett sebességhatár alatt marad, és korán elkezd fékezni.

3.5.2. Megerősítéses tanulás

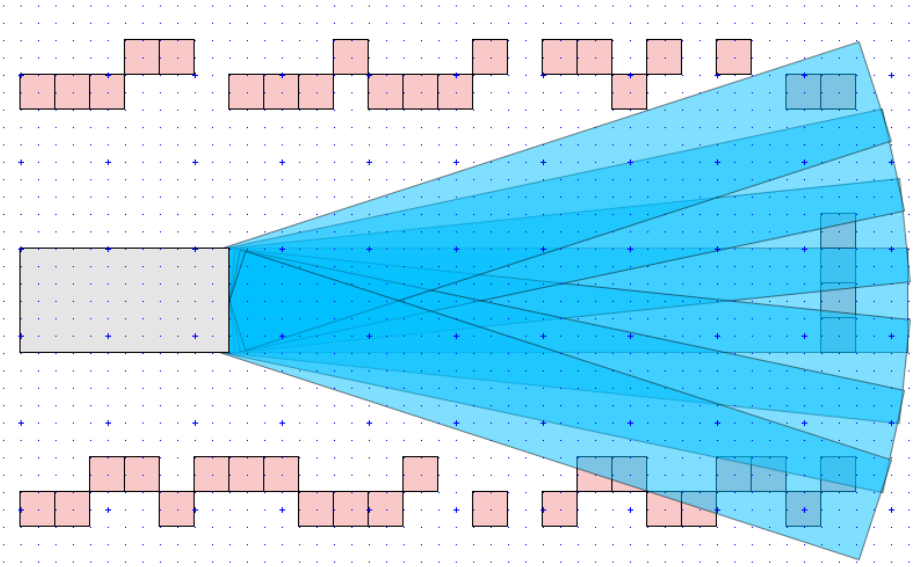
Az önvezető modul megvalósításához a megerősítéses tanítást választottam, mely a virtuális oktatótól fog tanulni. Egy közúti vezetői szakoktatótól is tanulhatna, de folyamatosan rögzíteni kellene mind a környezetet, mind az oktató tanácsait. Ennél hatékonyabb a virtuális oktató, főleg a megerősítéses tanulásnál, mert több száz, sőt több ezer esetből képes a megfelelő döntéseket megtanulni és a közlekedési szabályokat elsajátítani.

A megerősítéses tanulások közül a minőségi tanulást (Quality learning, Q-Learning, Watkins, 1989; Watkins et al., 1992) választottam. Ez a módszer egy ügynököt alkalmaz, amely a környezet állapota alapján hoz döntést, hogy milyen akciót hajtson végre (3.5.2-1. ábra). Az akciókért jutalom vagy büntetés jár, amelyről lehet, hogy csak a legvégén, több akció után kap értesítést. A teljes döntési sorozatot epizódnak hívják, egy döntést, mely az állapot alapján választott akciót jelenti, pedig lépésnek.



3.5.2-1. ábra: Megerősített tanulás folyamata

Az általam felállított Q-learningben a környezet állapotát a virtuális autó előtti, különböző irányú sávokban (3.5.2-2. ábra) észlelt voxelek kombinációja adja. A sávokat 0 vagy 1 értékkel látom el, az alapján, hogy tartalmaz-e akadályokat (0=nem, 1=igen). A sávok számától függően 2^s (ahol s a sávszám) kombináció van, vagyis ennyi állapot lehet. Ezekre az állapotokra a virtuális oktató meghozza a maga döntését, amely alapján az ügynököt jutalmazni lehet. Az ügynök akciói lehetnek a különböző mértékű kanyarodások, gyorsítás, sebesség tartás, lassítás, megállás. Vannak olyan helyzetek, amikor több irányban is mehet a virtuális autó, ilyenkor később kerül jutalmazásra, vagy büntetésre az ügynök.



3.5.2-2. ábra: Az autó előtti sávok, amelyben a közeli voxeleket vizsgálja a program (a hét kék színű sáv mindegyikében találhatók rózsaszínű voxelek)

A Q-Learninghez egy *állapotszám*akciók száma* méretű Q mátrixot hoztam létre, melyet az egyes állapotok és az ahhoz tartozó döntések, valamint a jutalom alapján aktualizálok a következő képlet alapján (Watkins, 1989; Leslie Pack et al., 1996):

*Q-mátrix(állapot, akció) = Jutalom(állapot, akció) + tanulási ráta * Max(következő állapot)*

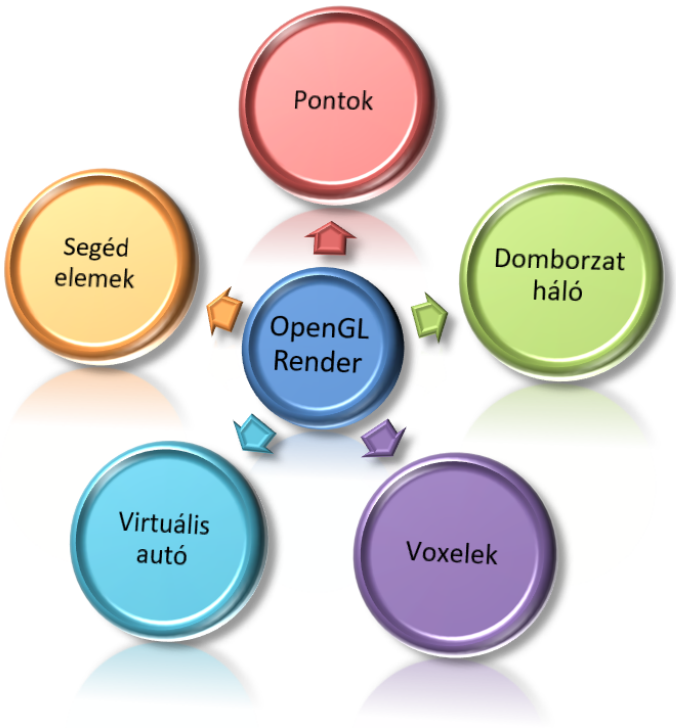
ahol :

- *Jutalom* az aktuális állapot és választott akció szerinti jutalom, melyet a virtuális oktató döntése alapján számolok
- a *tanulási ráta* egy 0…1 közötti érték, 0 közeli érték az aktuális, 1 közeli érték a hosszú távú jutalmat részesíti előnyben.
- *Max(következő állapot*) pedig a Q-Mátrix következő állapothoz tartozó akciók maximumát jelenti.

A virtuális környezetben több száz vagy több ezer epizód is gyorsan futtatható, melyekből az ügynök tanulni tud. Az úton az akadályok (autó, gyalogos) előfordulása véletlenszerű, tehát az ügynök nem abból tanul, hogy az út mely részén kell kanyarodni vagy megállni, hanem kizárólag a voxel közelségi adataiból. Az ügynök ezen közelségi adatok szerint választ akciót, és ha ez egybevág a virtuális oktató döntéseivel, akkor az akciók megfelelő jutalmat kapnak. Büntetést kap az ügynök, ha ütközik az úttesten lévő akadályokkal. A legtöbb esetben a jutalom vagy büntetés azonnal megadható, hisz kerülni kell a szegéllyel, vagy parkoló autókkal való ütközést. Ezek az epizód közbeni jutalmak, büntetések lerövidítik a tanulási folyamatot, ezért a virtuális oktató kifejlesztése és döntéseinek felhasználása hatékonyan tudja segíteni a Q-mátrix konvergenciáját, azaz a helyes döntési mátrix számítását.

3.6. 3D megjelenítés és szimuláció

A megjelenítő modul a gyors vizualizáció érdekében az OpenGL grafikus könyvtár segítségével szemlélteti a pontfelhőt és az önvezetést. Egy 3D rajzosztály (render) végzi a pontok, domborzatmodell, voxelek, virtuális autó és szemléltető elemek térbeli megjelenítését (3.6-1. ábra). Ezen elemek megjelenítése ki-be kapcsolható. A modul segítségével lehetőség nyílik a kameraállás gyors változtatására, az autó követésére, nagyításra, forgatásra, eltolásra.



3.6-1. ábra: OpenGL rajzosztály felépítése

A megjelenítés az OpenGL inicializálásával kezdődik ahol beállítom a pixelformátumot, és a hozzá tartozó paramétereket, például a színmélységet. Ezután létrehozom a rajzolási felületet, amelyen a 3D rajzfüggvények dolgoznak, illetve beállítom a Z-puffer-t a képobjektumok megfelelő takarásos megjelenítéséhez. Végül inicializálom a világítást, fényforrások konfigurációját, valamint kezelem a rajzolási és a renderelési beállításokat, amik a végleges kinézetért felelősek.

A modul tartalmazza az autó adatainak tárolására szolgáló struktúrát, illetve ehhez egy példányt az autó elhelyezkedésével és méreteivel. A modulhoz saját autórajzolót fejlesztettem. A domborzatmodell rajzolása a modell befoglaló koordinátái, szélessége és magassága alapján történik egy drótháló modellel.

A voxelek rajzolása egy kocka rajzoló segédrutin segítségével oldottam meg, ami a szűrés után megmaradt voxeleket rajzolja ki a középpontjuk és egységes oldalméretük alapján. A voxel rajzolás átlátszósággal történik, tehát az egymás mögötti voxelek is láthatók. A dolgozat ábráinak egy része ezzel a megjelenítővel készült.

5. Eredmények értékelése

5.1. Térbeli indexelés

Az új, R-fa mintája alapján kifejezetten nagy mennyiségű pontok indexelésére létrehozott C-Tree-P eljárás hatékonyságát az R-fa és a nyolcasfa algoritmusokkal hasonlítottam össze, amelyeket külön segédprogramokban implementáltam és futtattam ugyanazon a számítógépen (AMD Ryzen 5, 16 GB RAM, 1 TB SSD). A tesztekhez ugyanazt a 70 millió pontból álló, a szoftverhez is felhasznált mintaállományt használtam (5.1-1. Táblázat).

5.1-1. Táblázat: Három indexelési módszer összehasonlítása

Módszer	Felépítés: pontok tárolása és indexelése (sec)	Lekérdezési idő (sec/1000 lekérdezés)	Memória használat: pontok és index (GB)
R-fa	1,84	0,34	3,58
Nyolcasfa	0,63	0,23	2,44
C-Tree-P	0,57	0,16	2,18

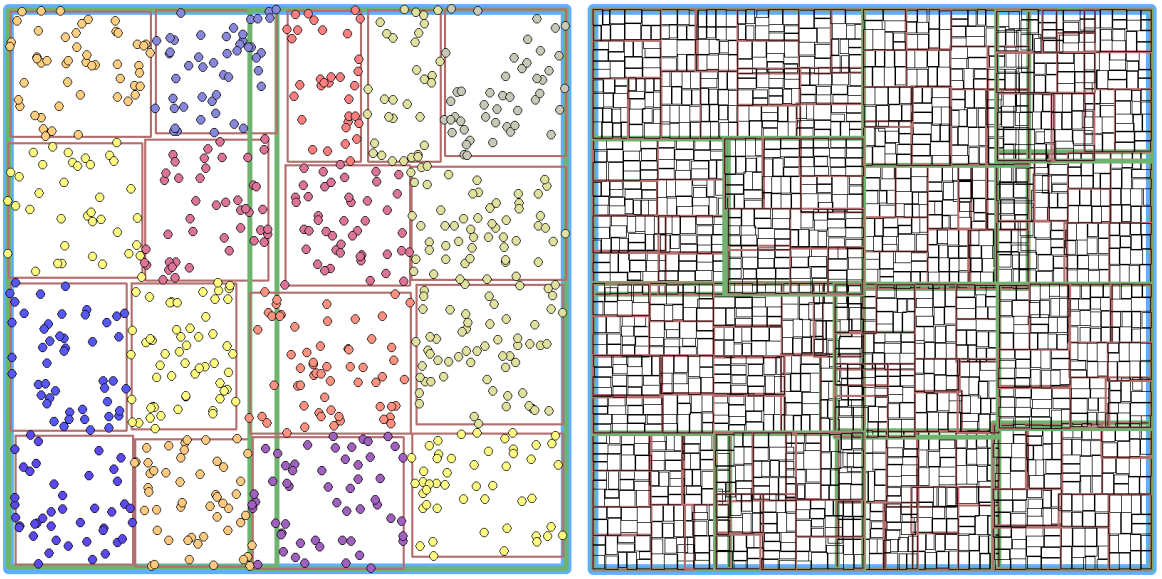
Az indexek felépítési ideje nem tartalmazza a pontok betöltési idejét az állományból. A vizsgálathoz a már memóriába betöltött pontokat használtam. R-fa esetén a javasolt 64-es lapméret helyett 256-ot választottam, mely nagyméretű pontállomány indexeléséhez ideális. Nyolcasfa indexelésnél a lapméret 4096 pont laponként, a felsőbb tárolók 8 gyerek tárolóra mutatnak. C-Tree-P esetén 64 és 4096 a lapméret, laponként 64 gyerek referenciát és 4096 pontot tud tárolni. Az adatszerkezetekben a tárolók kihasználtsága 75% körüli. A felépítés időadatai mutatják, hogy 256 lapméret esetén, a sok kettéosztás miatt az R-fa lassú. A nyolcasfa esetén a nem kiegyensúlyozott fa és az eltérő XY és Z irányú méret miatt több elágazásra és megosztásra van szükség, mint a C-Tree-P esetén.

A lekérdezési időket 1000 szimulált lekérdezéssel vizsgáltam. Az önvezető algoritmusban is használt lekérdezési ablakméretettel dolgoztam. Az R-fa 256-os lapmérete produkálta a leglassúbb lekérdezési időt, de még így is látszik, hogy a térbeli indexelés elengedhetetlen nagyméretű pontfelhők valós idejű lekérdezéshez. A lekérdezési időt tekintve a C-Tree-P eredményezte a legjobb időt. Itt nagyban segít a 64-es lapméret elágazás a nyolcasfa 8 elágazásához, vagy az R-fa 256 elágazásához képest. A nyolcasfának az egyenlőtlen pontsűrűség és a kisebb Z irányú kiterjedés sem kedvez.

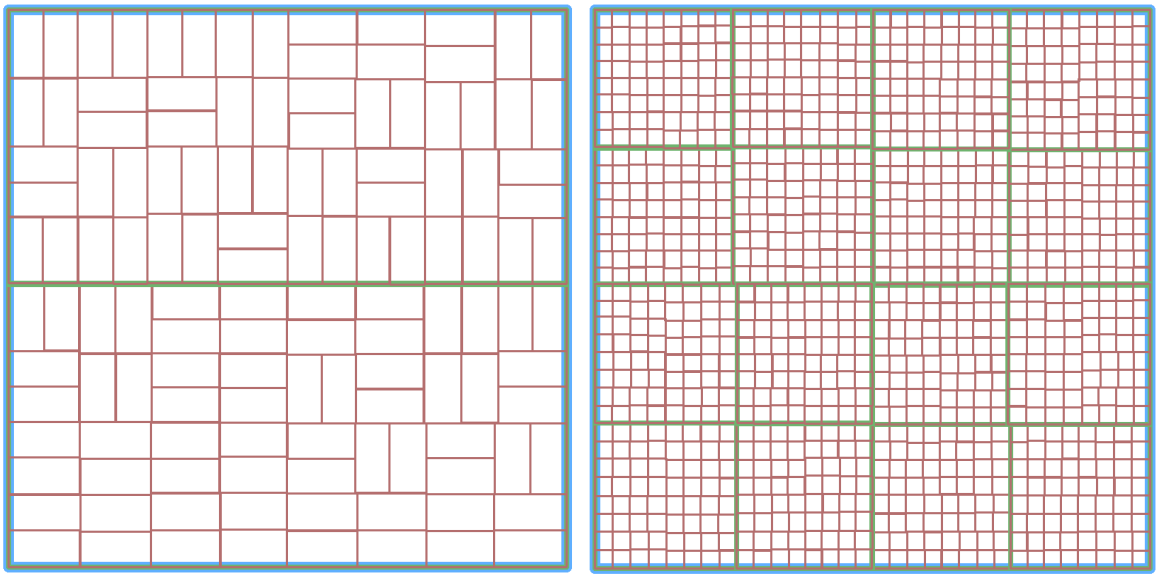
Az algoritmusok memória használatát a foglalt lapok, a tárolt pontok száma és a lapok memóriaigénye alapján számoltam. Egy pont tárolásához 24 bájt (3 double érték) szükséges, 70 millió pont tárolásához 1.6 GB. Itt látható, hogy az R-fa csaknem mégegyszer annyi

memóriát igényel az indexeléshez, de a nyolcasfa is több memóriát, több lapot igényel a pontfelhő egyenlőtlen eloszlása, anizotrópiája miatt.

Tesztelem a C-Tree-P-t, hogy milyen lapméret konfigurációnál lehet a legkisebb átfedést elérni a lapok között. A lapon tárolt pontszám növelése 64-ről 1024-re, majd 4096-ra előnyösen hat az átfedések csökkentésére (5.1-1. és 5.1-2. ábra). A különböző beállítások igazolták, hogy lap referenciák esetén 64 az ideális, pontméretnél pedig a 4096 a vizsgált állomány esetén.



5.1-1. ábra: 800 pont indexelése felülnézetben 16 lap/64 pont konfigurációval (bal oldal), és 80 000 pont indexelése 16 lap/64 pont konfigurációval (jobb oldal)

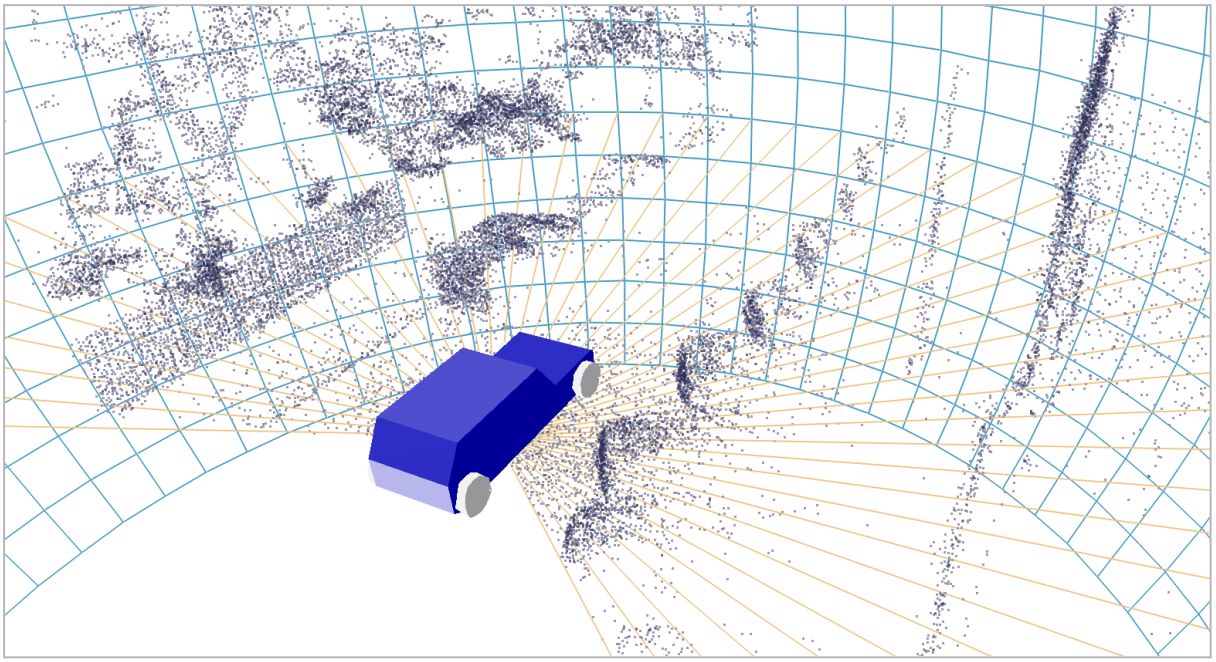


5.1-2. ábra: 80 000 pont indexelése felülnézetben 64 lap/1024 pont konfigurációval (bal oldal), 800 000 pont indexelése 64 lap/1024 pont konfigurációval (jobb oldal)

Meg kell említeni, hogy az R-fának számos változata, továbbfejlesztése készült el. Ezek implementációjával, tesztelésével nem foglalkoztam, mivel ezek elsősorban a nagyméretű objektumok (vonalláncok, poligonok) hatékony tárolására vagy lap megosztási eljárás finomítására fókuszálnak.

5.2. Szférikus mélységi puffer

Az új, gömbfelületre illesztett mélységi puffer (Z-buffer) segít a fedélzeti LiDAR-ok működésének szimulálásában. Megadható a puffer középpontja (autó középpontjához viszonyított relatív pozíciója), horizontális és vertikális szöge, illetve szögfelbontása. Alapbeállításként a jármű haladási irányához képest -120 .. +120 horizontális szög lehatárolást, valamint -40 .. +40 vertikál szög lehatárolást és fél fokos szögfelbontást használtam (5.2-1. ábra). Ettől eltérő értékekkel is próbálkoztam, amelyek gyorsan megadhatók a programban. A szoftver ezen funkciója arra is használható, hogy optimalizáljuk a fedélzeti LiDAR-ok számát és elhelyezését.



5.2-1. ábra: A szférikus mélységi puffer a lekérdezett pontfelhővel

A puffer paramétereinek kiválasztása során bebizonyosodott, hogy magasabb telepítés jobb domborzatmodellt eredményez, hiszen az érzékelő magasabbról lát az út felszínére, ami több felszíni pontot eredményez. A voxel detektálásra a szenzor magasabb elhelyezése nincs befolyással, de a voxelek felismerése alapvetően függ a domborzatmodelltől.

A fenti konfigurációjú puffer hátránya, hogy az autó mögött nem, vagy nagyon kevés pontot eredményez. Ezért az autó mögötti domborzatmodell pontatlanabb, az autó haladásával a modell a felismert pontok minimuma alapján dinamikusan változhat, hullámozhat. Ez az

anomália viszont nem befolyásolja az autó oldalánál és előtte lévő domborzat modellezését, valamint fölötte a voxelek kinyerését, végső soron az önvezetést.

LiDAR szimulálásához szférikus mélységi pufferrel vagy hasonló megoldással a szakirodalomban eddig nem találkoztam ezért nehéz összehasonlítani. Előfordulnak azonban más szimulációs eljárások amelyek általában 3D városmodellek alapján állítanak elő pontfelhőt (Hu Su et al., 2019).

5.3. Pontfelhő-feldolgozó

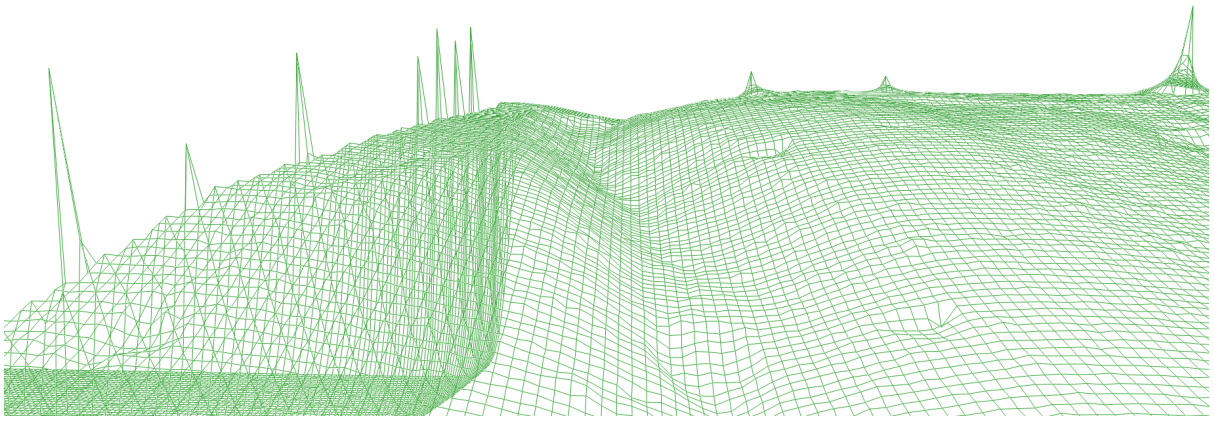
A feldolgozás során előáll egy domborzatmodell (DEM), mely lehetővé teszi a terep valós szimulálását, elkülönítve a talajfelszín feletti pontokat. Ez kihagyhatatlan a helyes voxelek előállítása érdekében. A vezetés szempontjából lényeges tereptárgyak detektálhatóak a voxelek segítségével, amelyek egyértelműen reprezentálják egy akadály jelenlétét. Mind a domborzat előállítás, mind a voxelfelismerés műveletei a valós világ pontosabb leképezését segíti, melyek a LiDAR adatokkal precízen felderítik az autó környezetét.

5.3.1. Módosított morfológia szűrő

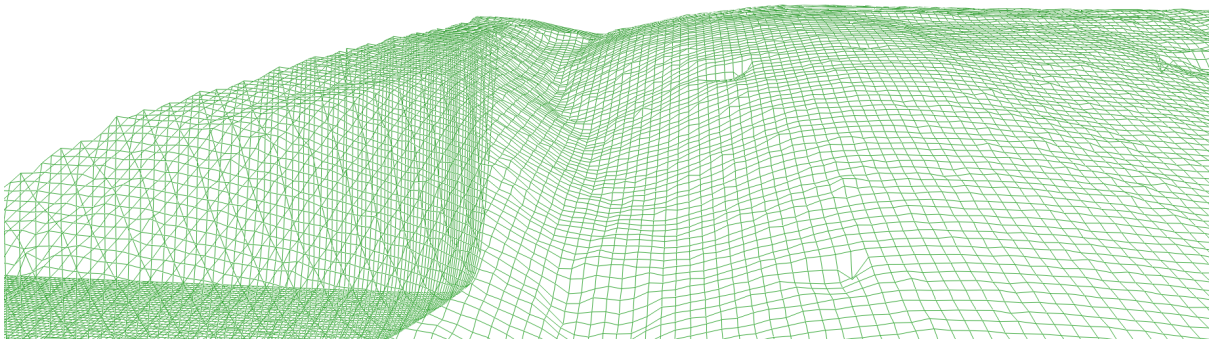
A domborzatmodell előállításának motorja a módosított morfológia szűrő, mely első körben szelektálja a talajfelszíni pontokat, majd ezután a hiányos cellákra a kiválasztott talajfelszín cellák interpolációjával domborzatmodellt illeszt.

A mikrodomborzati elemek (padka, lépcső) megőrzéséhez egy már ismert módszert (Keqi Zhang, 2003) használtam, mely egy cella magasságát hasonlítja a szomszédos cellákhoz. Ha a magasságkülönbség kisebb egy beállított értéknél, amely esetemben 0,2 méter, akkor megtartja, mint a domborzat kismértékű kiemelkedése vagy besüllyedése.

Az előbbi algoritmust olyan módon fejlesztettem tovább, hogy a domborzatmodell szélén, vagy hiányos területeken lévő, kevés ponttal rendelkező területeket is hatékonyan kezelje. A minimum és maximum szűrőknél bevezettem egy küszöbértéket. A szűrőket csak akkor alkalmazom, ha a nem üres pixelek száma meghaladja ezt a küszöbértéket, esetemben a szűrőben vizsgált pixelek 15%-át. Ha meghaladja, akkor a szűrést végrehajtom, ha nem, akkor üres pixelt eredményez az adott szűrő, vagyis nem vesz részt a kiválogatott talajfelszín pontok között. Ha a domborzatmodell szélén nincs elegendő pixel, akkor nulla értéket kap és az interpoláció kezeli a rácsháló meghatározását. Ezzel minimalizálható az ugrálás és precízebben besorolható a pontfelhő széle és a hiányos területek a domborzatmodellbe (5.3.1-1. ábra és 5.3.1-2. ábra).



5.3.1-1. ábra: Domborzat modellezés, hagyomás morfológia szűrővel



5.3.1-2. ábra: Domborzat modellezés, módosított morfológia szűrővel

A feldolgozóba került továbbá két véletlenszerűen elhelyezett akadály, melyek az önvezető modul tesztelésére szolgálnak. Az akadályok két virtuális gyalogost szimbolizálnak, akik tesztelik az önvezetést és átsétálnak a virtuális úttesten.

5.4. Virtuális oktató

Az autót a virtuális oktató, és oktatótól megerősítéses tanulással tanuló önvezető modul is tudja navigálni a virtuális környezetben. A virtuális oktatót többféleképp fel lehet paraméterezni, ezáltal vezetési viselkedéseket lehet beállítani és tesztelni, valamint ez szolgálhat a megerősítéses tanulás jutalmazási és büntetési számításaihoz. A virtuális oktató ugyanazt a voxelérzékelési metodikát használja, amit az önvezető modul is.

A virtuális oktatót a következőképpen paramétereztem fel:

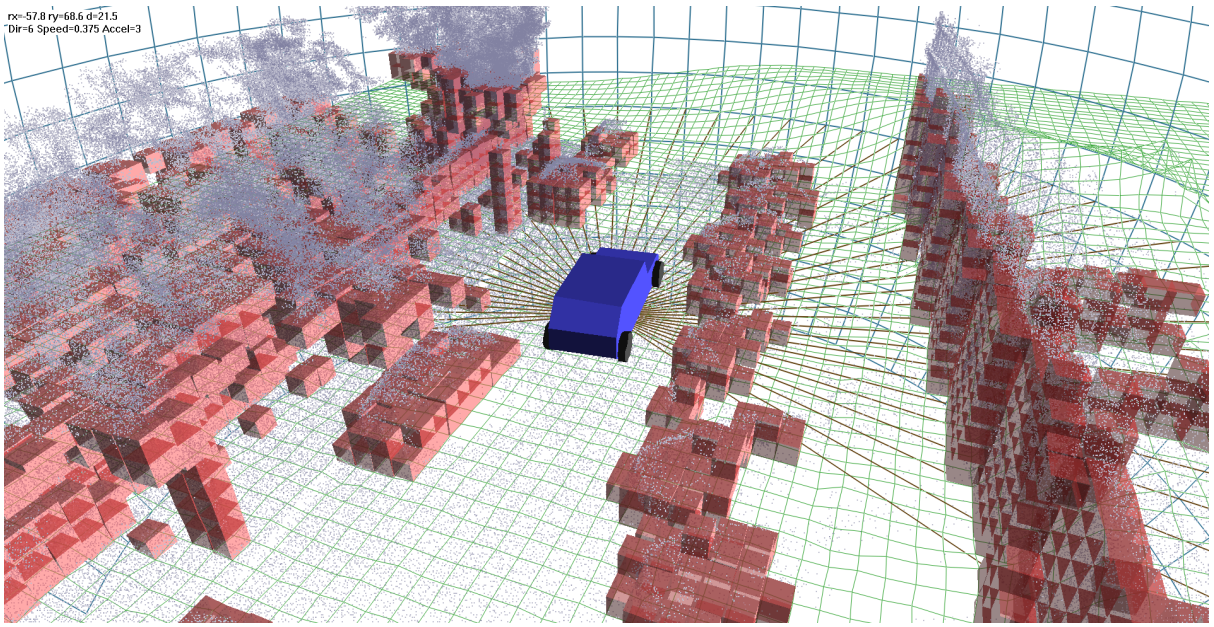
- Kezdősebesség: 0 km/h
- Maximális sebesség: 30 km/h
- Gyorsulás: 3 m/s^2
- Lassulás: -6 m/s^2 (tekintve, hogy a sebesség és a gyorsulás is pozitív irányú)

- Oldaltávolság tartás: 80 cm
- Kezdő irány: 0° (ez alapján számítjuk a gyorsulás és sebességvektor komponenseit)

Természetesen ettől eltérő paraméterek, viselkedések gyorsan beállíthatók.

Az autó haladási irányához képest voxeldetektálási sávokat definiálok a vízszintes síkon, amelyek iránya a haladási irány és a következő szögértékek összege: 0; 2; -2; 4; -4; 8; -8; 16, -16 fok. Ez összesen 9 sávot jelent. Ugyanezeket a sávokat kapja meg az önvezető modul is.

Az virtuális oktató első futtatásakor az autó az utca végén lelassított, lassan közelítette meg a keskeny útszakaszt és óvatosan haladt át rajta (5.4-1. ábra). A jelenség rendkívül érdekes, az oktató által kezelt sávokba voxelek kerültek, ezért lassított, míg tiszta pálya esetén gyorsan haladt az útszakaszon.

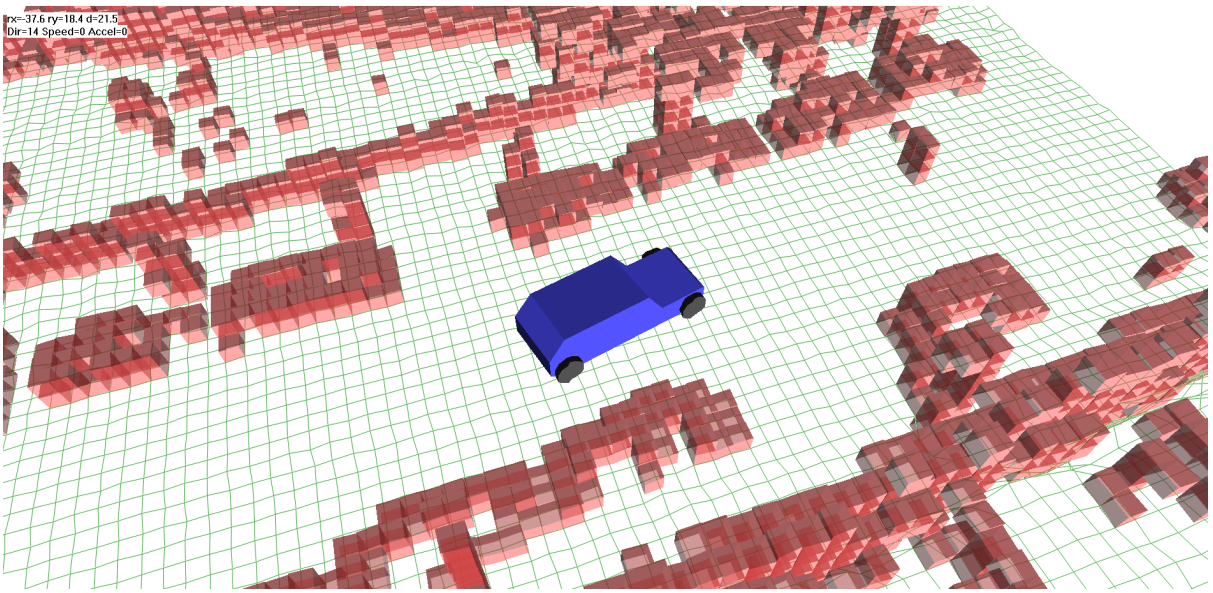


5.4-1. ábra: Keskeny szakasz megközelítése

A virtuális oktató statikus voxelek detektálására kiválóan reagál, lassan átkelő gyalogos esetén mindig megáll. Gyors dinamikus objektumok vagy balesetveszélyesen lelépő gyalogos esetén a szoftver későn érzékelhet, erre azonban egy sofőr is későn reagálna. Ez a tesztelés közben 3 m/s^2 gyorsulás és -3 m/s^2 lassulás mellett jelentkezett, ahol későn kerültek bele a voxelek az autó által választott sávba. A probléma megoldásához vizsgálni kéne egy mozgó akadály, azaz mozgó voxel sávokon való áthaladását, és becsülni a következő sávba való érkezését. Ez jelenleg nem része az önvezető modulnak, de a jövőben implementálásra kerülhet a javasolt módon.

Ha módosítjuk a lassítást -6 m/s^2-re a jármű minden veszélyes helyzetben megfelelően reagált a dinamikus akadályokra, annak ellenére, hogy mozgás becslése még nincs beépítve. A tesztfuttatásokból kirajzolódik, hogy az autó érzékeli a gyalogost, megáll és megvárja az

átkelését, majd helyesen végigmegy az útszakaszon (5.4-2. ábra). Hasonló tesztekkel könnyen szemléltethető a voxelfelismerés és az önvezetés hatékonysága.



5.4-2. ábra: Az autó megállással várja a gyalogos átkelését

5.5. Önvezető modul

A megerősítéses tanulási Q-Learning modell állapotait a kilenc érzékelési sávból képezem a következő módon: ha egy sáv tartalmaz voxelt, akkor a neki megfelelő bitet a 9 bitből bekapcsolom. Ez összesen 512 állapotot jelent (2^9). A modellt tizenkét akcióra képeztem ki, ebből kilenc a detektálási sávokba történő kanyarodás, valamint egy a lassítás, egy a megállás és egy a gyorsítás. A Q mátrix mérete ezek alapján 512x12 = 6144. Közel 3500 lépésen, nyolc epizódon keresztül jutalmazza vagy bünteti a virtuális oktató az önvezető modult. Egy epizód az autó eljutása az utca elejétől a végéig. 10%-os felfedezést engedtem, és 50%-os tanulási rátát (learning rate) használtam. Átlagosan 33 lekérdezés, feldolgozás, önvezetési döntés és 3D megjelenítés történik másodpercenként, ami jól mutatja a teljesítményt és a valós idejű működést. A méréshez ismét az AMD Ryzen 5 processzorral rendelkező személyi számítógépet használtam.

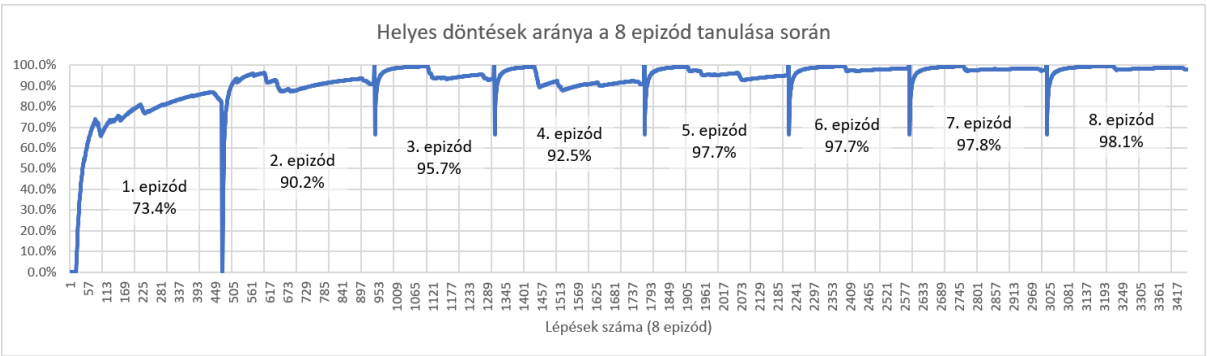
Az önvezető modul tehát a 0-ás bittel rendelkező sávokat választja, illetve megáll, ha mindegyik sáv foglalt. Ha az oktató is javasolja a megállást akkor nagyobb jutalmat adok a modellnek. Többféle tesztelést próbáltam, beleértve a büntetés nélkülit, azaz nullával való jutalmazást, illetve nagyobb büntetési értékek használatát. A legjobb konvergenciát a mínusz tízes büntetés, egyes nem hibás, de saját döntés, illetve az oktató döntéseinek követésénél tízes jutalmazással értem el. A tanulás folyamata a helyes előrehaladás Q értékének ábrázolásán figyelhető meg, amelyen látható a konvergencia emelkedése a végleges érték felé (5.5-1. ábra).

5.5-1. ábra: A helyes előrehaladás tanulási görbéje

A 3500 lépésből álló tanítás során, már az első epizód után, közelítőleg 500 lépésnél erősen felfelé konvergál a tanulás folyamata, és 2000 lépésnél állandósul. A 10%-os felfedési ráta és a tanulás közbeni jutalmazás sokat segít a tanulási folyamat konvergenciáján, sokkal kevesebb lépésre van szükség, mintha csak a végén jutalmaznám az ügynököt.

Az önvezető modul a tanítás után a sávokban érzékelt voxel objektumok esetén a megfelelő parancsokat választja, nagyrészt elfogadva az oktató tanácsait. Az akadályok elhárulásával folytatja útját gyorsítással a célpontig.

A modell jelenleg 98%-os sikerrátával működik, amely nyolc epizódon keresztül a tanulás és a helyes döntések (virtuális oktatóval megegyező döntések) arányában értelmezhető (5.5-2. ábra). A modell tehát pozitívan megtanulja az oktató által biztosított vezetési viselkedést. A virtuális oktató és a folyamatos jutalmazás-büntetés szerepét hangsúlyozza, hogy már az első epizódban 73% a helyes (virtuális oktatót követő) döntések aránya, és már a harmadik epizódban eléri a 95%-ot.



5.5-2. ábra: Az önvezető modul helyes döntéseinek aránya 8 epizódon keresztül

A kidolgozott megerősítéses tanulási eljárásban a felfedezési és tanulási ráta, a jutalmazási és büntetési stratégia mind-mind módosíthatók, ezáltal számos tanulási folyamat hozható létre, amelyek különbözőképp konvergálnak. Hasonló modelleknél rendkívül nehéz megtalálni a pontos paramétereket, így hiperparaméter optimalizációval mindig javítható a képességük. A hiperparaméter optimalizáció alatt a modell teljesítményének maximalizálsát értjük.

6. Összefoglalás és továbbfejlesztési lehetőségek

A lézeres letapogatási adatokkal működő önvezető szoftver fejlesztése során számos új eredmény, meglévő eljárások továbbfejlesztése született meg. Az első a továbbfejlesztett térbeli indexelés, amely kifejezetten nagy mennyiségű pont adatok indexelésére és tárolására hoztam létre. Az R-fához képest jóval kisebb a memória igénye és használatával gyorsabb felépítés és lekérdezés érhető el.

Hasonló innovatív fejlesztés a szférikus mélységi puffer, melynek segítségével a fedélzeti lézeres letapogatók szimulálhatók. Továbbfejlesztettem a morfológia szűrőket a domborzatmodell kinyeréshez, amely elsősorban a modell szélein és hiányos területein jelent előrelépést.

Kidolgoztam egy különböző vezetési stílusokkal felruházható virtuális oktatót, amely a pontfelhőn keresztül képes vezetni a virtuális autót szakértői döntésekkel. Felállítottam egy Q-Learning megerősítéses tanulási modellt, amely elsősorban a virtuális oktatótól tanul és a tanítás után képes a következő önvezető funkciók ellátására: kanyarodás, lassítás, megállás, gyorsítás. A megerősítéses tanulási modellel 98% fölötti helyes döntési arányt értem el.

Valamennyi algoritmus fejlesztésénél fontos szempont volt a valós idejű feldolgozás és a virtuális környezet térbeli megjelenítése, ezáltal egy szimulációs környezet létrehozása. A fejlesztett alkalmazás 33 feldolgozás/másodperc teljesítményre képes, ezért a címben is szereplő valós idejű célkitűzés teljesült. Ebben a környezetben valós időben vizsgálhatók a pontfelhő-feldolgozó algoritmusok hatékonysága, anomáliái, valamint a két önvezető modul (virtuális oktató és megerősítéses tanulás) döntési folyamatai.

Az elért eredmények biztatóak, de a kutatás-fejlesztés során számos továbbfejlesztési lehetőség ötlete merült fel. A szoftver jelenleg egy kisebb szeletét képezi az autonóm járművezetési rendszereknek, ezért több újítás kiindulópontja lehet.

Hasznos továbbfejlesztés lenne a voxel objektumok mozgásának becslése a virtuális térben. Az egyes időfázisokban történő voxelfelismerések összevetésével, a mozgó voxelcsoportok érzékelhetők, irányuk és sebességük előre jelezhető. Ebben nehézség lehet a voxel csoportok azonosítása és az összetartozó voxelek összekapcsolása.

Hasonló továbbfejlesztési irány a mély megerősítéses tanulás beépítése, mely ugyan komplexebb, de előnyösebb nem diszkrét helyzetek kezelésére, mint a vezetés alatt felmerülő problémák halmaza vagy az akadályok távolsága. Ezek a modellek több tanítási adatot igényelnek, de robosztusabbak és könyebben kezelnek komplex vezetési szituációkat is.

Komoly fejlesztést igényel, de megtérülhet egy városrész vagy egy teljes város lézeres letapogatása, majd több jármű útraindítása a virtuális környezetben. Minden autó kiindulási és megérkezési pontjának megadásával sokkal több és komplexebb vezetési helyzetet lehetne szimulálni, ezáltal jobb megerősítéses tanulási modellt létrehozni.

7. Köszönetnyilvánítás

Szeretnék köszönetet mondani a konzulenseimnek a támogatásukért, segítségükért és a dolgozathoz fűzött rendkívül értékes, építő jellegű kritikáikért. Tanácsaik és iránymutatásuk sokat segített a dolgozat letisztultságában és teljességében.

Külön köszönet Dr. Németh Krisztiánnak, hogy időt szakított rám, és sokszor éjszakába nyúlóan foglalkozott a leírtak helyességével. A számos megjegyzés, precíz és könnyen követhető támpontot adott a szakmai és nyelvtani felépítést illetően.

Szeretném megköszönni Dr. Czimber Kornélnak is a támogatását, akinek ösztönzése és problémamegoldó meglátásai kulcsfontosságúak voltak a dolgozat elkészítésében. Édesapám nagyban felkeltette az önvezető téma iránti érdeklődésem még a 2023-as év kezdetén, ami azóta karrier opcióként is funkcionál.

Továbbá köszönet Baranyai Dánielnek, aki a lézeres letapogatással kapcsolatban számos hasznos forrással és javaslattal látott el. Tudása a LiDAR-ról elősegítette a pontos specifikációt és a gördülékeny haladásom.

Mégegyszer köszönöm a segítségüket és a lehetőséget, hogy Önökkel vehettem részt a 2023-as Tudományos Diákkonferencián.

8. Szakirodalmi hivatkozások

1. Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, Jia Heming. (2022). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. Information Sciences, Volume 622, Pages 178-210.
2. Alireza Ansariyar. (2023). A Comprehensive Literature Review of Connected and Autonomous Vehicles (Cavs) Impacts On Mobility and Environment. SSRN Electronic Journal. doi:10.2139/ssrn.4433637
3. Alireza Ghasemieh és Rasha Kashef. (2022). 3D object detection for autonomous driving: Methods, models, sensors, data, and challenges. Transportation Engineering, Volume 8, 100115.
4. Altman, N. S. (1992). An introduction to kernel and nearest neighbor nonparametric regression. The American Statistician, Volume 46, Pages 175–185. doi:10.1080/00031305.1992.10475879. hdl:1813/31637
5. American Society for Photogrammetry and Remote Sensing (ASPRS). (2013). LAS SPECIFICATION, VERSION 1.4 – R13.
6. Anastasius S. Moumtzoglou. (2019). Quality Assurance in the Era of Individualized Medicine. doi: 10.4018/978-1-7998-2390-2
7. Asanka Wasala, Donal Byrne, Philip Miesbauer, Joseph O'Hanlon, Paul Heraty, Peter Barry. (2019). Trajectory based lateral control: A Reinforcement Learning case study. Engineering Applications of Artificial Intelligence, Volume 94, September 2020, 103799.
8. Catherine Ross és Subhrajit Guhathakurta. (2017). Autonomous Vehicles and Energy Impacts: A Scenario Analysis. World Engineers Summit – Applied Energy Symposium & Forum: Low Carbon Cities & Urban Energy Joint Conference, WES-CUE 2017, Singapore
9. ChatGPT. (2023). Válasza a "Write me a string class including strcat, strcpy, stoi and stod with basic functions and operators" parancsra. https://chat.openai.com. ChatGPT 3.5 version.
10. Collet, C., és Musicant, O. (2019). Associating Vehicles Automation with Drivers Functional State Assessment Systems: A Challenge for Road Safety in the Future. Front. Hum. Neurosci. 2. doi:10.3389/fnhum.2019.00131
11. Darsh Parekh, Nishi Poddar, Aakash Rajpurkar, Manisha Chahal, Neeraj Kumar, Gyanendra Prasad Joshi és Woong Cho. (2022). A Review on Autonomous Vehicles: Progress, Methods and Challenges.
12. David Moten. (2022). R*-tree split Visualization. Github adattár davidmoten/rtree-3d. https://github/davidmoten/rtree-3d.com.
13. Dong, Y.; Cui, X.; Zhang, L.; Ai, H. (2018). An Improved Progressive TIN Densification Filtering Method Considering the Density and Standard Variance of Point Clouds. ISPRS Int. J. Geo-Inf.,Volume 7, Page 409.
14. Evans, J. S. és Hudak, A. T. (2007). A Multiscale Curvature Algorithm for Classifying Discrete Return LiDAR in Forested Environments. IEEE Transactions on Geoscience and Remote Sensing, vol. 45, no. 4, pp. 1029-1038.
15. Fabian Kröger. (2016). Automated Driving in Its Social, Historical and Cultural Contexts. https://link.springer.com/chapter/10.1007/978-3-662-48847-8_3
16. Fayez Alanazi. (2023). A Systematic Literature Review of Autonomous and Connected Vehicles in Traffic Management. Traffic Safety Measures and Assessment. Applied Sciences, Volume 13, Issue 3.
17. GeoSLAM (FARO Technologies Inc.). (2023). https://geoslam.com/solutions/zeb-horizon/

18. Guttman, Antonin. (1984). R Trees: A Dynamic Index Structure for Spatial Searching. Sigmod Record, Volume 14, Pages 47-57. doi: 10.1145/971697.602266.
19. Hu Su, Rui Wang, Kaixin Chen, Yizhan Chen. (2019). A Simulation Method for LIDAR of Autonomous Cars. IOP Conference Series: Earth and Environmental Science. 234. 012055. doi: 10.1088/1755-1315/234/1/012055.
20. Hyungjun Park, Zulqarnain Khattak, and Brian Smith. (2018). Glossary of Connected and Automated Vehicle Terms. VERSION 1.0. University of Virginia Center for Transportation Studies. Connected Vehicle Pooled Fund Study.
21. Janiesch, C.; Zschech, P. és Heinrich. K. (2021). Machine learning and deep learning. Electron Markets 31, 685–695. https://doi.org/10.1007/s12525-021-00475-2
22. Justin Kek. (2019). Alphabet's Waymo: A Technical Study on Autonomous Vehicle Tech. https://justinkek.medium.com/alphabets-waymo-a-technical-study-on-autonomous-vehicle-tech-c128180ab2c5#ece4
23. Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage és Anil Anthony Bharath. (2017). A Brief Survey of Deep Reinforcement Learning. IEEE Signal Processing Magazine, Special Issue On Deep Learning For Image Understanding (Arxiv Extended Version).
24. Kaufman, A.; Cohen, D.; YAGEL, R. (1993). Volume graphics. IEEE Computer, vol. 26, 51–64.
25. Keqi Zhang, Shu-Ching Chen, D. Whitman, Mei-Ling Shyu, Jianhua Yan és Chengcui Zhang. (2003). A progressive morphological filter for removing nonground measurements from airborne LIDAR data. IEEE Transactions on Geoscience and Remote Sensing, vol. 41, no. 4, pp. 872-882. doi: 10.1109/TGRS.2003.810682.
26. Kraus, K. és Pfeifer, N.. (1997). A new method for surface reconstruction from laser scanner data. Int. Arch. Photogramm. Remote Sens, vol. 32, pt. 3–2W3.
27. Leslie Pack Kaelbling, Michael L. Littman és Andrew W. Moore. (1996). Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, Volume 4, Pages 237-285.
28. Lokhandwala, M.; Cai, H. (2018). Dynamic ride sharing using traditional taxis and shared autonomous taxis: a case study of NYC. Transp. Res. Part C: Emerg. Technol., pp. 45-60,
29. Meagher, Donald. (1980). Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer.
30. Merat, N. és Lee, J. D. (2012). Preface to the special section on human factors and automation in vehicles: designing highly automated vehicles with the driver in mind. Hum. Fact. 54, 681–686. doi: 10.1177/0018720812461374
31. Nanxi Li, Chong Pei Ho, Jin Xue, Leh Woon Lim, Guanyu Chen, Yuan Hsing Fu, Lennon Yao Ting Lee. (2022). A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors. Laser & Photonics Reviews, Volume16, Issue11.
32. Nelson Diaz, Omar Gallo, Jhon Caceres, Hernan Porras. (2021). Real-time ground filtering algorithm of cloud points acquired using Terrestrial Laser Scanner (TLS). International Journal of Applied Earth Observation and Geoinformation, Volume 105, 102629.
33. Ninad Mehendale és Srushti Neoge. (2020). Review on Lidar Technology. SSRN papers: http://dx.doi.org/10.2139/ssrn.3604309
34. Noor Jannah Zakaria, Mohd Ibrahim Shapiai, Rasli Abd Ghani, Mohd Najib Mohd Yassin, Mohd Zamri Ibrahim, Nurbaiti Wahid. (2023). Lane Detection in Autonomous Vehicles: A Systematic Review. IEEE Access, Volume 11.
35. Óscar Silva, Rubén Cordera, Esther González-González, Soledad Nogués. (2022). Environmental impacts of autonomous vehicles: A review of the scientific literature. https://www.sciencedirect.com/science/article/pii/S0048969722017089#bb0315

36. Patrizia Firmani, Siewert Hugelier, Federico Marini, Cyril Ruckebusch. (2018). MCR-ALS of hyperspectral images with spatio-spectral fuzzy clustering constraint. Chemometrics and Intelligent Laboratory Systems, Volume 179, Pages 85-91.
37. Peide Wang. (2021). Research on Comparison of LiDAR and Camera in Autonomous Driving. J. Phys.: Conf. Ser. 2093 012032.
38. Peter Axelsson. (1998). Processing of laser scanner data—algorithms and applications. ISPRS Journal of Photogrammetry & Remote Sensing 54, 1999, Pages 138–147.
39. Peter Axelsson. (2000). DEM Generation from Laser Scanner Data Using Adaptive TIN Models.ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, Volume 33, 110-117.
40. Precedence Research. (2023). Autonomous Vehicle Market - Global Industry Analysis, Size, Share, Growth, Trends, Regional Outlook, and Forecast 2023 - 2032. https://www.precedenceresearch.com/table-of-content/1074
41. Qi Liu, Xueyuan Li, Shihua Yuan, Zirui Li. (2021). Decision-Making Technology for Autonomous Vehicles: Learning-Based Methods, Applications and Future Outlook. https://arxiv.org/ ftp/arxiv/papers/2107/2107.01110.pdf
42. Qiming Zhou. (2017). Digital Elevation Model and Digital Surface Model. The International Encyclopedia of Geography. doi:10.1002/9781118786352.wbieg0768
43. Rezapur-Shahkolai, F.; Afshari, M.; Doosti-Irani, A.; Bashirian, S.; Maleki, S. (2022) Interventions to Prevent Road Traffic Injuries among Pedestrians: A Systematic Review; Taylor & Francis: Abingdon, UK. [Google Scholar] [CrossRef]
44. Rohling, H. és Moller, C. (2008). Radar waveform for automotive radar systems and applications. IEEE Radar Conference, Pages 1–4.
45. SAE International. (2018). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. J3016_201806. https://www.sae.org/ standards/content/j3016_201806/
46. Salim Dridi. (2022). Reinforcement Learning - A Systematic Literature Review. Preprint.
47. Santiago Royo és Maria Ballesta-Garcia. (2019). An Overview of Lidar Imaging Systems for Autonomous Vehicles. LiDAR and Time-of-flight Imaging. Applied Sciences, Volume 9, Issue 19.
48. Sean Campbell, Niall O' Mahony, Lenka Krpalkova, Daniel Riordan. (2018). Sensor Technology in Autonomous Vehicles : A review. 29th Irish Signals and Systems Conference (ISSC).
49. Sekerak, R. (2011). Voxel. In: Kreutzer, J.S., DeLuca, J., Caplan, B. (eds) Encyclopedia of Clinical Neuropsychology. https://doi.org/10.1007/978-0-387-79948-3_82
50. Shahian-Jahromi Babak, Syed A Hussain, Burak Karakas és Sabri Cetin. (2017). Control of autonomous ground vehicles: a brief technical review. IOP Conference Series: Materials Science and Engineering, Volume 224, 4th International Conference on Mechanics and Mechatronics Research (ICMMR 2017).
51. Shangshu Cai, Wuming Zhang, Xinlian Liang, Peng Wan, Jianbo Qi, Sisi Yu, Guangjian Yan és Jie Shao. (2019). Filtering Airborne LiDAR Data Through Complementary Cloth Simulation and Progressive TIN Densification Filters. Frontiers in Spectral Imaging and 3D Technologies for Geospatial Solutions. Remote Sensing, Volume 11, Issue 9.
52. Taewon Ahn, Yongki Lee és Kihong Park. (2021). Design of Integrated Autonomous Driving Control System That Incorporates Chassis Controllers for Improving Path Tracking Performance and Vehicle Stability. Systems & Control Engineering. Electronics, Volume 10, Issue 2.
53. Treat, J. R. (1977). “Tri-level study of the causes of traffic accidents: an overview of final results,” in Proceedings of the American Association for Automotive Medicine Annual

Conference, vol. 21 (Chicago, IL: Association for the Advancement of Automotive Medicine), 391–403.
54. Veli Ilci és Charles Toth. (2020). High Definition 3D Map Creation Using GNSS/IMU/LiDAR Sensor Integration to Support Autonomous Vehicle Navigation. Positioning and Navigation. Sensors, Volume 20, Issue 3.
55. Velodyne LiDAR. (2018). Velodyne LiDAR Puck REAL-TIME 3D LiDAR SENSOR. https://www.mapix.com/wp-content/uploads/2018/07/63-9229_Rev-H_Puck-_Datasheet_Web-1.pdf
56. Watkins, J.C.H. Christopher (1989). Learning from Delayed Rewards. Ph.D. thesis. King's College, Cambridge, UK.
57. Watkins, J.C.H. Christopher és Dayan, Peter. (1992). Technical Note Q-Learning. Machine Learning, Volume 8, Pages 279-292. Kluwer Academic Publishers, Boston.
58. Wuhua Jiang, Chuanzheng Song, Hai Wang, Ming Yu, és Yajie Yan. (2023). Obstacle Detection by Autonomous Vehicles: An Adaptive Neighborhood Search Radius Clustering Approach. Selected Papers from the 1st International Electronic Conference on Machines and Applications.
59. Xian-Feng Han, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, Liping Xiao. (2017). A review of algorithms for filtering the 3D point cloud. Signal Processing: Image Communication, Volume 57, Pages 103-112.
60. Xiaodong Wu, Bangjun Qiao és Chengrui Su. (2020). Trajectory Planning with Time-Variant Safety Margin for Autonomous Vehicle Lane Change. Intelligent Transportation Systems. Applied Sciences, Volume 10, Issue 5.
61. Xiaoming, L.; Qin Tian, Chen Wanchun, és Xingliang, Y. (2010). Real-time distance measurement using a modified camera. IEEE Sensors Applications Symposium (SAS), Pages 54-58.
62. Xuelian Meng, Nate Currit és Kaiguang Zhao. (2010). Ground Filtering Algorithms for Airborne LiDAR Data: A Review of Critical Issues. Remote Sensing, Volume 2, Issue 3.
63. Yoganandhan, A.; Subhash, S.D.; Hebinson Jothi, J.; Mohanavel, V. (2020). Fundamentals and development of self-driving cars. materialstoday: Proceedings, Volume 33, Part 7, Pages 3303-3310.
64. Yong Zhang, Feng Gao és Fengkui Zhao. (2023). Research on Path Planning and Tracking Control of Autonomous Vehicles Based on Improved RRT* and PSO-LQR. Intelligent Techniques Used for Robotics. Processes, Volume 11, Issue 6.
65. Yuki Endo és Shunsuke Kamijo. (2023). Deep Voxelized Feature Maps for Self-Localization in Autonomous Driving. Artificial Intelligence (AI) and Machine-Learning-Based Localization. Sensors, Volume 23, Issue 12.