KÖRNYEZETELEMZÉS KOMPAKT LÉZERSZKENNEREKKEL

Dokumentáció az NKFIH K-16 #120233 projekt keretein belül végzett kutató- és fejlesztőmunkáról

KÉSZTETTE:

Horváth Győző

PROJEKTVEZETŐ:

Dr. Benedek Csaba

BUDAPEST, 2018

Tartalomjegyzék

Összefoglaló

Napjainkban egyre inkább teret nyernek a lézeres távolságmérésen alapuló Lidar (Light Detection and Ranging) szenzorok különféle környezetelemző eljárások során. A riportban elsőként bemutatásra kerül a lézerszkennerek működési elve, a különféle típusok közti különbségek, és azok felhasználási területei. Bemutatom a szakirodalom eddigi eredményeit a különféle típusú Lidarok pontfelhőinek analízisével kapcsolatban, és a jelentősebb kihívásokat ezeken a területeken. Ezt követően autóra szerelhető RMB (Rotating Multi-Beam – forgó többsugaras) Lidarok pontfelhő sorozatának feldolgozásával foglalkozom. A szakirodalomban olvashatunk működő eljárásokról nagy méretű 64 csatornás Lidarokra. Azonban ezeket az eljárásokat szükséges lenne a jelentősen olcsóbb és kisebb, alacsonyabb felbontású Lidarokra is fejleszteni. A kedvezőbb ár és kisebb méret miatt ugyanis szélesebb körben kerülhetnének felhasználásra a szenzorok az ezekre fejlesztett algoritmusokkal. A feladatom 16 csatornás kompakt Lidar pontfelhő sorozatain valós időben működő talajpont-detektáló és objektum-elkülönítő algoritmus fejlesztése. Ehhez a szakirodalomban bemutatott, 64 csatornás Velodyne HDL64 Lidarhoz fejlesztett algoritmusok alapján új eljárásokat dolgozok ki, és illesztek a Velodyne VLP16 Lidarhoz. Az algoritmust kiegészítem még egy lombkoronapontokat felismerő és eltüntető eljárással, ami javítja az észlelt objektumok arányát. A sikeres objektumelkülönítés után feltérképezem az észlelt objektumok felismerésének lehetőségeit. Beszámolok még a mérések menetéről, és azok kiértékeléséről, továbbá dokumentálom az elkészített szoftverrendszert programozói és felhasználói szinten. Végül elemzem a megvalósított funkciók működését, feltérképezem a gyakorlati felhasználási lehetőségeket, illetve a továbblépési lehetőségeket.

Abstract

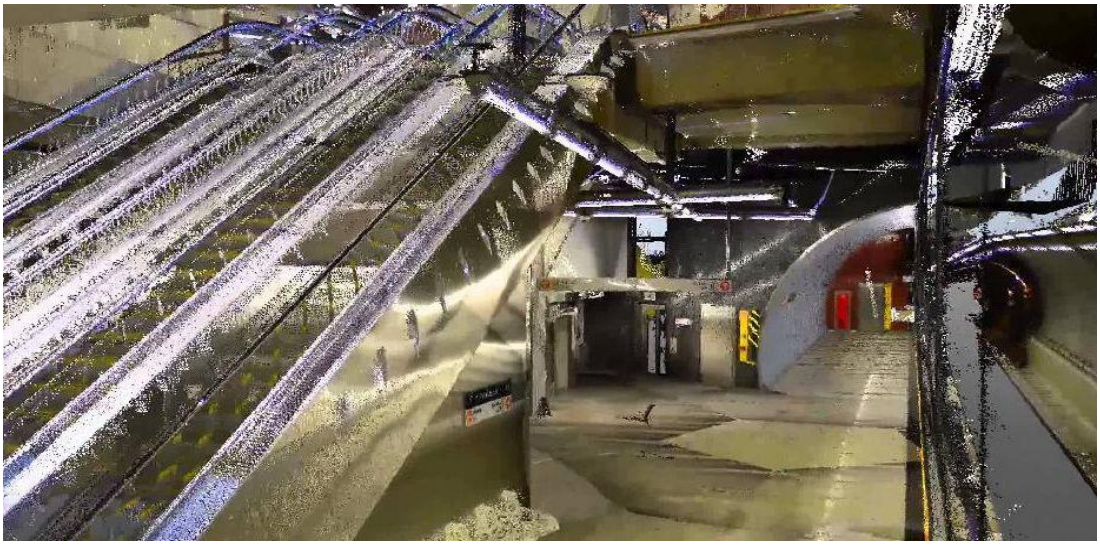
Nowadays Lidar (Light Detection and Ranging) sensors based on distance measurement with laser beams are gaining ground in many kinds of environment analysing processes. In my thesis firstly the principle of operation and the differences between various types of laser scanners are being presented. I will present some related works of the analysis of point clouds from several Lidar types, and I will mention some sorts of significant difficulties. In the following I will deal with the processing of point cloud sequences from RMB (Rotating Multi-Beam) Lidar sensors mounted on the top of a moving vehicle. In the literature we can read about working processes for 64-channeled Lidars. Nevertheless these processes need to be developed for much cheaper and lighter Lidars with lower resolution. These sensors would become more widespread with the algorithms developed for them because of the lower price and lower weight. My task will be to develop an algorithm for ground detection and object separation working in real time on point cloud sequences of 16-channeled compact Lidars. For this I will create new method based on the algorithms developed for the 64-channeled HDL64 Lidar previously shown in the related works. I will complete this process with a tree crown detecting and removing algorithm, which makes the hit rate of the detected objects even better. After the success of object detection I will map the possibilities of the recognition of these objects. I will also report the steps and the evaluation of the measurement. More than that, I will document the software framework on programmer and user level. Finally I will examine the effectiveness of the implemented algorithm, the possible usage of the achieved functions and possible further steps.

1 Bevezetés

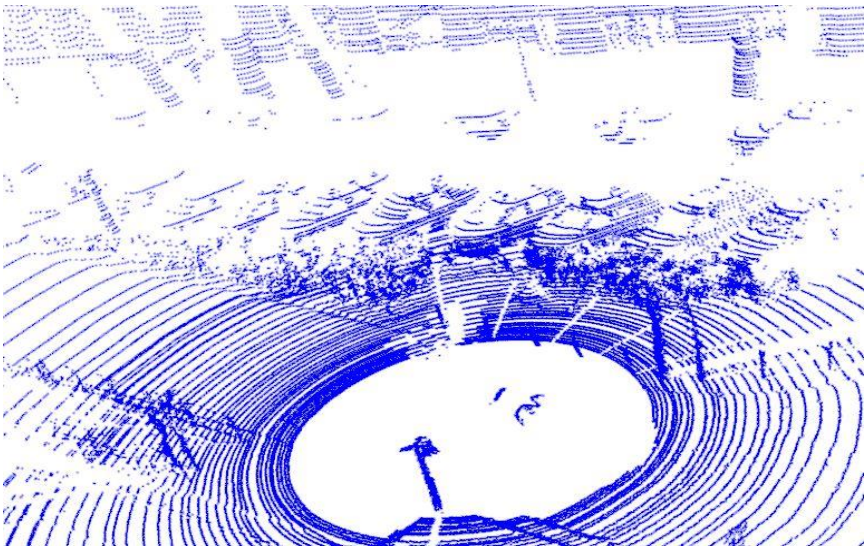
Az elmúlt időszakban egyre inkább terjedni kezdtek a lézerszkenner alapú környezetelemző eljárások. Korábban jellemzően kamerák által rögzített fényképek alapján próbáltunk információkat szerezni a környezet bizonyos tulajdonságairól, azonban az ilyen eljárások során számos nehézséggel is szembe kell néznünk. A hagyományos kameraképekre működő algoritmusok például érzékenyek az időjárási körülményekre, mivel azok a vizuális látványt jelentősen befolyásolják. További hátrány, hogy a fényképek nem tartalmaznak mélységértékeket a felületi pontokhoz, így a környezetről csak korlátozott mértékű geometriai információ áll rendelkezésre. Ezekre a problémákra egyaránt megoldást jelentenek a lézeres távolságmérésen alapuló LIDAR (Light Detection and Ranging) szenzorok, amelyek a kibocsátott lézernyaláb visszaverődése után vizsgálják annak a fázisát vagy a visszaérkezésig eltelt időt, majd ebből számítják ki az objektumok távolságát a szenzortól pontról pontra. A lézeres távolságmérés során a környezetet 3D-ben rögzítjük, így rendelkezésünkre áll mélységinformáció is. A visszaverődési tulajdonságokat nem befolyásolják az időjárási körülmények, így a kimenet ezektől is független lesz. A Lidarok hátránya, hogy nem rögzítenek színinformációt, csupán a visszavert lézernyaláb intenzitásértékét. Az összetettebb lézerszkenner alapú térképező rendszereket ezért kamerákkal is felszerelik, melyek segítségével a pontokhoz színinformáció is rendelhető *[10]*.

Különböző típusú Lidarok állnak rendelkezésre a piacon, ezek között a legjelentősebb különbségek a rögzített pontfelhők sűrűségében és a mintavételi idejükben találhatók. Az adott alkalmazási terület dönti el, hogy milyen Lidarra van szükség a feladat megoldásához. Nagyfelbontású lézerszkennereket használnak például utcák vagy városok 3D modelljének elkészítéséhez, régészeti ásatások helyszínének felméréséhez, építkezések folyamatának dokumentálásához. Az ilyen nagyfelbontású pontfelhőkön összetett algoritmusok nem futhatnak valós időben. Rendelkezésre állnak kisebb felbontású lézerszkennerek is, melyeket valós idejű feladatok megoldásához használhatunk fel. Valós idejű feladat lehet például emberek felismerése, objektumok észlelése, vagy objektumok követése. Az 1. illetve 2. ábrákon láthatók nagy sűrűségű térképezésre alkalmas, illetve kis sűrűségű valós idejű adatszolgáltatást megvalósító Lidarok pontfelhői.

A riportban főként az önjáró autóknál felhasznált kompakt Lidarok és azokon futtatott algoritmusok témakörét járja körbe. Az autóra RMB (Rotating Multi-Beam – forgó többsugaras) Lidart szerelünk, ami a környezetet 3D pontfelhő képében rögzíti körülbelül 20 pontfelhő/másodperc frissítési rátával annak mozgása közben. Ezeken a pontfelhőkön akár valós időben is végrehajthatunk különböző eljárásokat a környezet értelmezése, például autonóm navigációs vagy vezetést segítő döntések meghozatala érdekében. A 64 csatornás Lidarokra korábban már megvalósításra kerültek működő algoritmusok, azonban ezek a Lidarok túlságosan drágák és nagy méretűek ahhoz, hogy széles körben elterjedjenek. A cél az, hogy ezek az eszközök idővel egy átlagember számára is elérhetőek legyenek, ezért a Lidarok mérete és ára folyamatosan csökken. Azonban ez egyúttal a pontfelhők sűrűségének csökkenését eredményezi, ami problémákhoz vezethet. Az objektumok helyes elkülönítése a kisebb pontsűrűség miatt nehezebb, mint a többcsatornás Lidarokon. A korábban 64 csatornás Lidarokra működő algoritmusokat a jóval kisebb 16 csatornás Lidarokra kell fejleszteni, optimalizálni. Az alábbiakban a 16 csatornás Lidar alkalmazásával a környezetelemzési lehetőségeket fogom megvizsgálni.



1. ábra: nagy sűrűségű pontfelhő – M4-es metró vonalának felmérése (Budapest Közút)



2. ábra: kis sűrűségű pontfelhő – Velodyne HDL64 Lidar pontfelhője

2 Lidar alapú érzékelési technológia

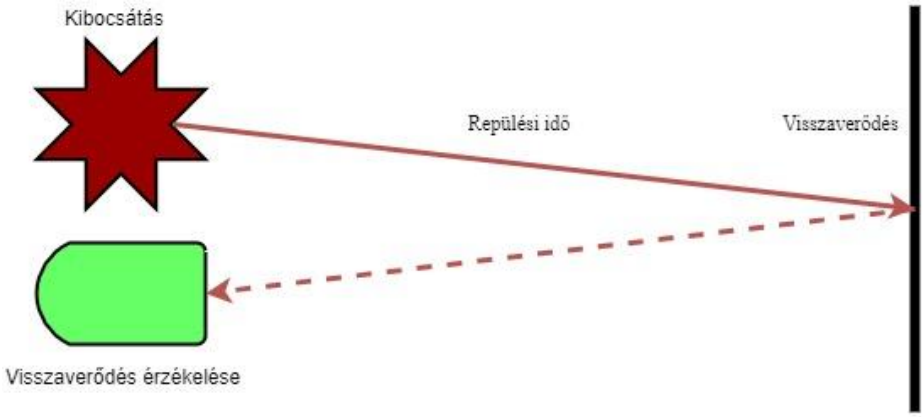
2.1 Az érzékelési technológia elve

A Lidar (Light Detection and Ranging) alapú érzékelési technológia – nevéből adódóan – fényérzékelésen alapuló távolságmérési technikát alkalmaz. Ennek alapja, hogy a kibocsátott lézersugár objektumról visszavert nyalábjának becsapódásakor lemérjük a kilövés és a becsapódás között eltelt időt, vagy a modulált lézersugár fázisát. Ezután a mért értékből számítjuk ki az objektum adott pontjának a szenzortól való távolságát. Ezt a műveletet többször megismételve különböző irányokra a szenzor környezetében található objektumok pontjainak távolságát egyenként lemérjük. Ezt az eljárást nevezzük lézerszkennelésnek. A lézerszkennelés során a környezetet gömbi koordináta-rendszerben rögzítjük, a vízszintes és függőleges síkkal bezárt szögeket minden lézersugár kibocsátáshoz ismerjük, a pontok távolságot pedig mérjük. A visszavert lézernyaláb vizsgálata alapján háromféle lézerszkennert különböztetünk meg *[1]*, *[2]*, melyeket a következőkben röviden bemutatok.

2.1.1 Impulzus alapú lézerszkenner

Az impulzus alapú (pulse-based) lézerszkenner (más néven: TOF – Time of Flight Scanner) a lézersugár kibocsátása és visszaérkezése közti időkülönbséget méri, majd ebből számítja ki az objektum távolságát az alábbi képlet alapján:

$$Távolság = \frac{Fénysebesség \cdot Időkülönbség}{2}$$

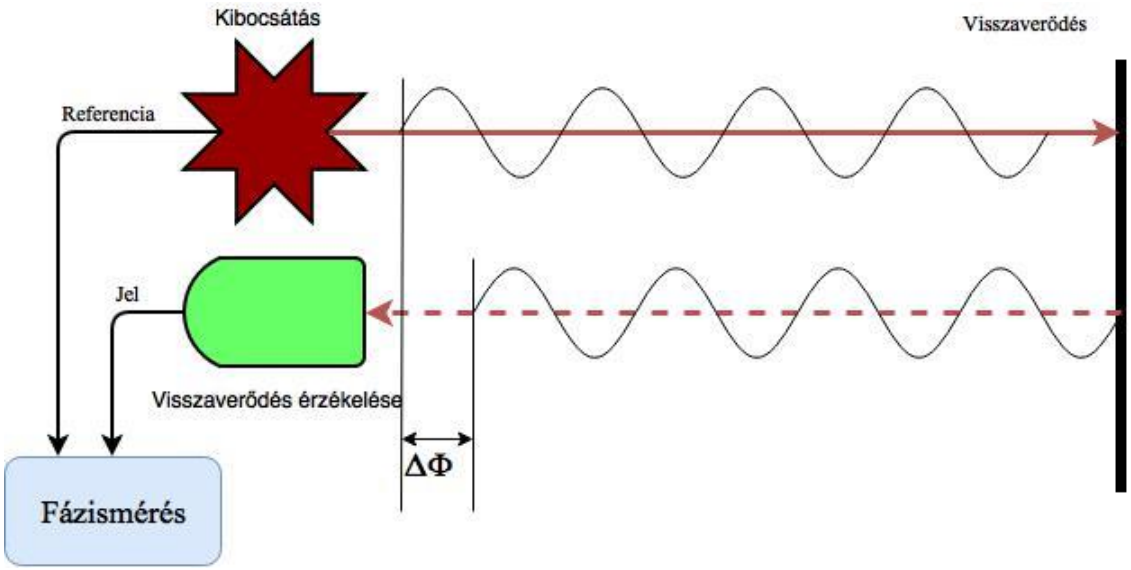


3. ábra: Impulzus alapú távolságmérés

A távolság mérésének a hibája attól függ, hogy az időt mennyire precízen vagyunk képesek mérni. Az ilyen működésű szkenner nagy távolságok mérésére alkalmas főként, szkennelési ideje illetve a pontossága nem túl jó. A vele mérhető maximális távolság elérheti akár a 6000 m-t is.

2.1.2 Fázismérés alapú lézerszkenner

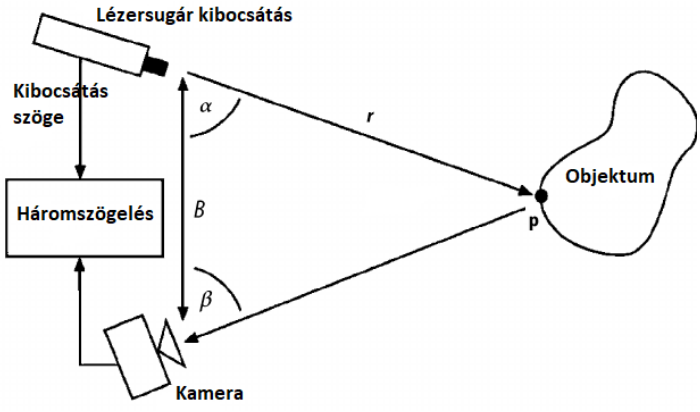
A fázismérés alapú (phase-based) lézerszkenner által kibocsátott lézersugarat a Lidar harmonikus hullámmal modulálja, majd a visszavert sugár fáziskülönbségéből számítja ki az objektum távolságát. Ez a típus nagyon magas szkennelési sebességgel rendelkezik (akár egymillió pont/másodperc), illetve a pontossága is jelentősen nagyobb (milliméter nagyságrendű), mint az impulzus alapú szkennereknek. Azonban a hatótávolsága kisebb, néhány 10 m-ig terjed.



4. ábra: Fázis alapú távolságmérés

2.1.3 Háromszögelés alapú lézerszkenner

A háromszögelés alapú (triangulation-based) lézerszkennernél a kibocsátó egység és a kamera ismert távolságra helyezkedik el egymástól. Tudjuk még az α kibocsátási szöget is. Továbbá a β szöget meghatározhatjuk az objektum lézerrel megvilágított pontjának a kamera képén elfoglalt pozíciója alapján. Ebből a három információból a megvilágított pont távolsága egyértelműen meghatározható. Az ilyen típusú szkennerek nagy pontossággal bírnak (kb. 0.1 mm), azonban a hatótávolságuk elég kicsi (kb. 1 m).



5. ábra: Háromszögelés alapú távolságmérés

2.2 Mobil lézeres adatgyűjtő rendszerek

A lézerszkennereket az alkalmazás módja szerint két csoportba sorolhatjuk. Megkülönböztetünk mobil lézerszkenner rendszereket, melyeket mozgó járművekre rögzítve használunk; illetve statikus lézerszkennereket, melyeket fix állványra helyezve alkalmazunk. A mobil lézerszkennerek között megkülönböztetünk nagyfelbontású, azonban valós idejű felhasználáshoz alkalmatlan; illetve kisfelbontású, de valós idejű feladatokhoz felhasználható Lidarokat.

2.2.1 Valós idejű lézerszkennerek

A valós idejű lézerszkennerek egyik legérdekesebb felhasználási területe az önjáró autók. Nekik képesnek kell lenniük környezetük valós idejű elemzésére, hogy ez alapján döntéseket hozzanak. Korábban ezeket az elemzési-értelmezési feladatokat kamerák 2D képének feldolgozásával próbálták megoldani. Azonban az ilyen módon működő rendszerek érzékenyek voltak az időjárási körülményekre, és a látási viszonyokra. Súlyos döntési helyzetekben a 2D kép nem elég ahhoz, hogy teljes bizonyossággal a helyes döntést hozzuk meg. Ehhez szükség lehet más környezetelemző módszerek felhasználására is. A Lidarok nagyon pontos távolságinformációkat mérnek a környezetükben, aminek köszönhetően a környezet geometriai leírói is rendelkezésünkre állnak. A különböző típusú Lidarok 5-30 Hz-es frissítési rátával készítenek pillanatképeket pontfelhő formájában a környezetükről. Városi forgalomban a 20 Hz-es frissítési ráta annyit jelent, hogy egy-egy pontfelhő között körülbelül 70 cm eltérés tapasztalható. $\left(s = v \cdot t = 50\,\frac{km}{h} \cdot \frac{1}{20} s = 50\,\frac{1000m}{3600s} \cdot \frac{1}{20} s = \ 0{,}69\ m\right)$

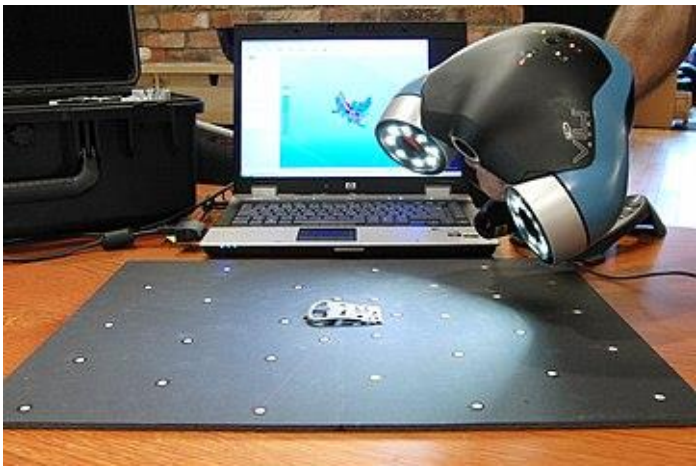
6. ábra: Velodyne-család: balról jobbra: HDL64, HDL32, VLP16

A valós idejű lézerszkennereket különböző környezetelemző eljárások során használhatjuk fel: mozgásészlelés, objektumészlelés, talaj meghatározása. Emellett alkalmasak még biztonságtechnikai feladatok megoldására is.

2.2.2 Nem valós idejű lézerszkennerek

2.2.2.1 Micro lézerszkenner

Kis objektumok nagy részletességű felvételéhez alkalmasak a MicroLS (Micro Laser Scanning) rendszerek. Előnyük, hogy nagyon nagy pontossággal bírnak háromszögelés-alapú működési módjuknak köszönhetően. A rögzíteni kívánt objektumot a lézerszkennerrel „körbejárjuk”, majd a mért pontfelhőket térben egymáshoz illesztjük. Ezeket a rendszereket felhasználhatjuk minőségellenőrzési feladatokhoz, kulturális műalkotások dokumentációjához, mérnöki alkotások visszafejtéséhez (reverse engineering), illetve prototípuskészítéshez (prototyping).

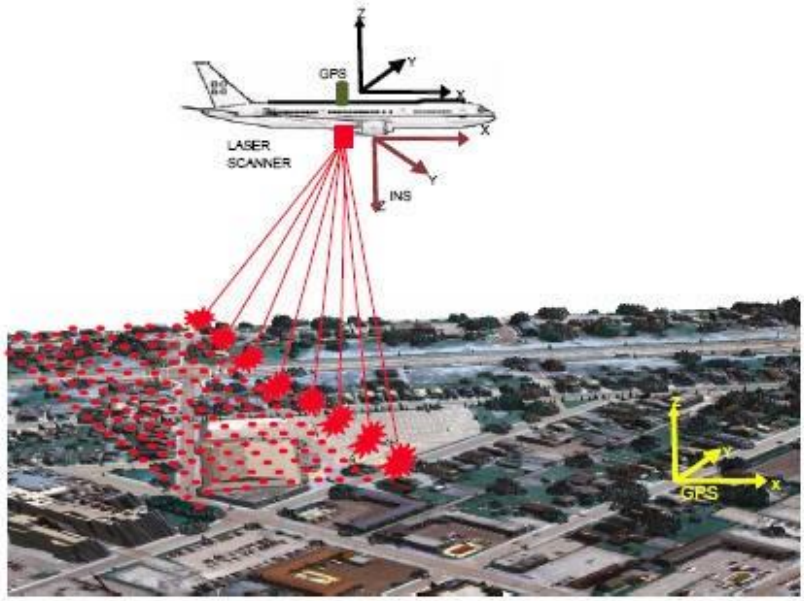


7. ábra: Mikro lézerszkennelés

2.2.2.2 Mobil lézerszkenner

Mobil lézerszkennereket alkalmazunk nagy kiterjedésű területek (pl. városok, utcák) feltérképezésére. A mobil lézer szkenner rendszerek a mérési eredményeket megfelelő méretű tárolókra mentik a mérés során, majd ezek csak később, számításigényes algoritmusokkal kerülnek feldolgozásra. A különböző pillanatokban felvett pontok pontos illesztéséhez szükségesek további információk a globális pozícióról illetve a jármű kisebb kimozgásairól. Ezeket az információkat a lézerszkenner mellett működő pozícióméró GPS (Global Positioning System) és térbeli helyzetet mérő IMU (Inertial Measurement Unit) rendszerek biztosítják. Az említett 3 egység által biztosított adatokból az adott pillanatban, adott pozícióban rögzített pontfelhők későbbi utófeldolgozás során rendkívül nagy pontossággal illeszthetők.

A mobil lézerszkennelés egyik felhasználási területe a légi távolságképek felvétele (Airborne Laser Scanning, ALS). Ehhez nagy hatótávolságú, de kisebb pontosságú Lidarokat használunk, és ezeket repülőgépre rögzítve haladunk át a megörökíteni kívánt terület felett. A nagy hatótávolság és kisebb pontosság követelményének megfelelően ToF (time of flight – impulzus alapú) elven működő lézerszkennereket alkalmazunk ilyen mérésekre. Kiterjedt területek (mezőgazdasági, erdészeti területek, városi régiók, városi növényzet) légi rögzítéséhez használunk ALS rendszereket.



8. ábra: Légi lézerszkennelés

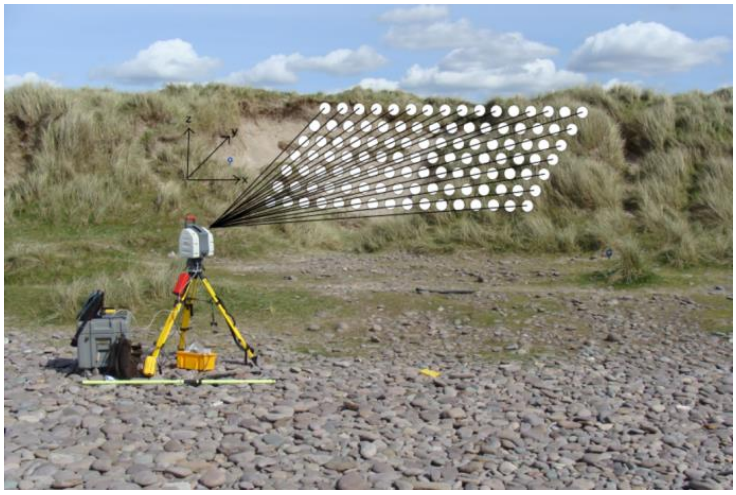
A mobil lézerszkennelés másik nagy felhasználási területe a térképező eljárások. A felvételeket mozgó járműre rögzítve készítjük el a szárazföldön (Terrestrial Laser Scanning, TLS (TLS kategóriába soroljuk a később említett statikus lézerszkennert is)). Ez a fajta eljárás alkalmas épületek, hidak, útszakaszok, vagy akár egy egész város felmérésére is. A Lidarokat nem csak autóra, hanem hajóra rögzítve is használhatjuk, így hidak szerkezetét alulról is rögzíthetjük. A városról készült nagyfelbontású pontfelhők alapján az úthálózat javításának és fejlesztésének terveit lehetséges elkészíteni. Ezen kívül az út melletti jelzőtáblákról, zebrák számáról, és egyéb fontos információkról lehetséges adatbázist létrehozni a készített pontfelhő alapján. A Budapest Közút rögzítette Budapest 3D pontfelhő modelljét az elmúlt években, hogy az előbb említett felhasználási területekben rejlő lehetőségeket kihasználja. A lézeres mobil térképező rendszerrel felszerelt autó jelenleg is folyamatosan járja a várost, hogy frissítse és bővítse a már meglévő adatokat.

9. ábra: Szárazföldi lézerszkennelés

10. ábra: Hajóról történő lézerszkennelés

2.2.2.3 Statikus lézerszkenner (TLS)

A statikus lézerszkennereket állványra helyezve fix pozícióban használjuk. Azonban az állvány pozícióját egy mérés elvégzése során többször is megváltoztatjuk, hogy minél pontosabb képet kapjunk a rögzíteni kívánt épületről, környezetről. Statikus lézerszkennert használhatunk építkezések különböző fázisainak dokumentálásához, régészeti ásatások helyszínének felméréséhez, vagy akár műemlékek 3 dimenziós megörökítéséhez. A mobil lézer szkenner által nem elérhető helyszínek is statikus lézerszkennerrel mérhetők fel. A Budapest Közút például statikus lézerszkennerrel rögzítette az M4-es metró vonalának felmérése során a megállókat, hiszen oda járművel nem lehet eljutni.



11. ábra: Statikus lézerszkennelés

3 Korábbi eredmények

3.1 Kompakt Lidaros érzékelők mérésein alapuló analízis

Az objektumészlelés és objektumfelismerés kulcsfontosságú feladat az önjáró autók környezetelemzése során, ezért ez egy jelenleg is nagyon aktív kutatási terület. Munkánk során a feladat megoldásához kompakt Lidaros érzékelőket használunk mozgó járműre szerelve. Ezek a Lidarok akár 20-30 Hz-es frissítési rátával küldenek egy-egy pillanatképet (pontfelhőt), melyeken különféle környezetelemző eljárásokat futtathatunk. A szakirodalomban főként a már közel 10 éve megjelent 64 csatornás Lidarok pontfelhő szekvenciáin futtatható eljárásokról olvashatunk *[3], [4]*.

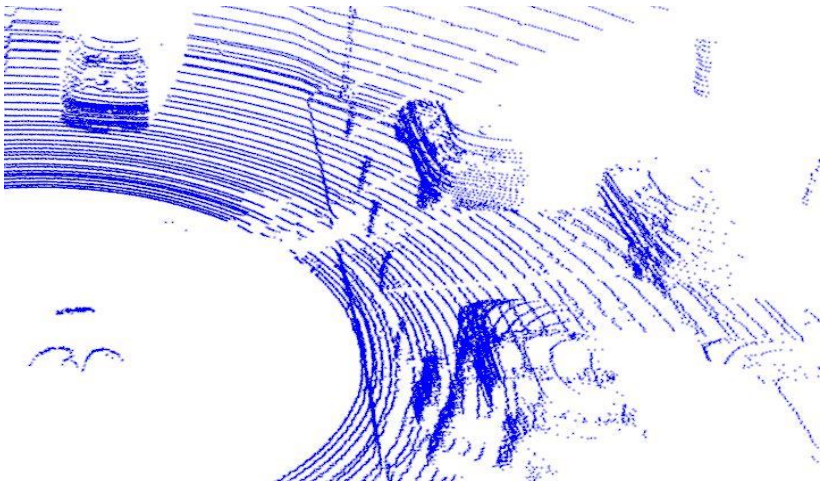
3.1.1 Nehézségek

Az objektumészlelést és -felismerést több tényező is nehezíti. A következőkben a legáltalánosabb problémákat fogom bemutatni. Ezek többsége a szenzor mérési karakterisztikájából ered. Komoly kihívások elé állítja a kutatókat az a tény, hogy a pontsűrűség a pontfelhőn belül meglehetősen alacsony a térképező rendszerek által rögzített pontfelhőkhöz képest, ráadásul a pontok térbeli eloszlása inhomogén, a szenzortól való távolság függvényében meredeken csökken. A nagyfelbontású térképező rendszerek esetén az objektumok több nézőpontból is megörökítésre kerülnek, majd az ezekben a nézőpontokban készült pontfelhők összeillesztése után alakul ki a végleges pontfelhő. Azonban a valós idejű kompakt Lidarok jelének feldolgozása során csak egy nézőpontból látjuk a környezetet, ami eltorzult, átfedésbe lépő objektumokat eredményezhet. A 12. ábra bemutatja, hogy bár négy – a valóságban hasonló – autót is látunk a képen, azok megjelenése jelentősen eltérő a pontfelhőben, mert más-más irányból látjuk őket. Ez az objektumfelismerést nehezíti.

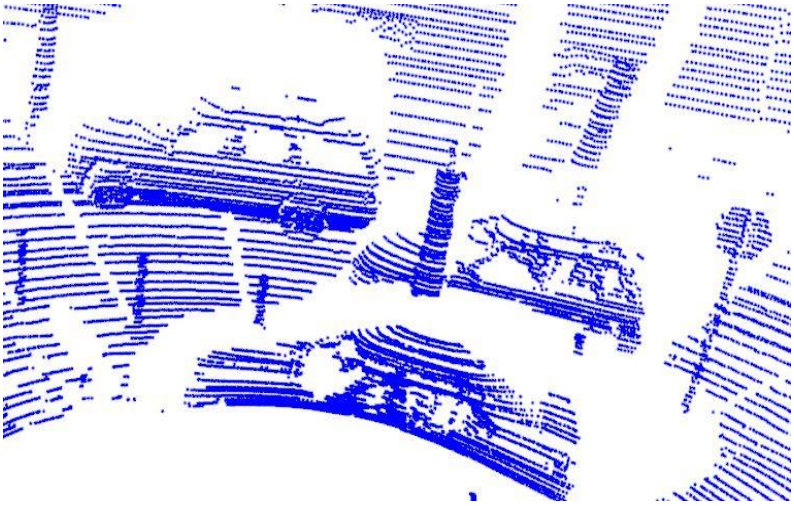
Egy másik jelentős probléma, ha objektumok átfedésbe kerülnek egymással, így bizonyos részleteiket nem „látja” a Lidar. Ezt a jelenséget szemlélteti a 13. ábra. A mellettünk elhaladó autó a parkoló járműnek egy részét kitakarja, aminek okán az a pontfelhőben vizuálisan különbözni fog a ki nem takart parkoló járművektől. A szemünknek ez nem jelent túlzottan nagy problémát a felismerésben, de az algoritmusoknál akár hibás működést is eredményezhet.

További nehézségeket okoz, hogy a megvalósított algoritmusoknak valós időben működniük kell. Emiatt nem alkalmazhatunk nagy számítási igényű eljárásokat, melyek nagyon pontos eredményekkel szolgálhatnának.

Összegezve: a legalapvetőbb probléma az, hogy ugyanazok az objektumok akár teljesen máshogy nézhetnek ki, ha másik objektumok takarásában helyezkednek el, vagy másik nézőpontból látjuk őket. Emiatt meglehetősen nehéz leírni egy-egy objektumtípust rá jellemző tulajdonsághalmazzal, így azok felismerése nehéz feladat. Ezt még súlyosbítja az a tény, hogy nem alkalmazhatunk nagy számítási igényű algoritmusokat, mert akkor a valós idejű működés követelményét nem elégítenénk ki.



12. ábra: Autók különböző nézőpontból



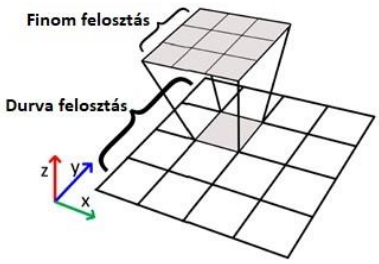
13. ábra: Objektumok átfedése

3.1.2 Szakirodalmi eredmények

A pontfelhők feldolgozása az alkalmazott adatstruktúra kiválasztásával kezdődik. Ahogy *[3]*-ban olvashatjuk, kétféle adatstruktúra használható fel, azonban a fagráf-alapú technikák (Kd-tree, Octree, range tree) gyakran nem alkalmasak felhasználásra valós idejű alkalmazásokban. *[3]* bemutat egy valós időben végrehajtható objektumdetektáló algoritmust, melyben a Lidar pontfelhők feldolgozásának alapja szabályos rács alapú technika.

3.1.2.1 Hierarchikus rács a pontfelhők szemantikus régiókra bontására

A *[3]*-ban ismertetett algoritmus alapja egy 2D hierarchikus szabályos rács, mely áll egy durva felosztásból, illetve ennek a durva felosztásnak a további finomításából (14. ábra). Erre a finomításra azért van szükség, hogy a módszer az egymáshoz közel helyezkedő objektumokat is képes legyen elkülöníteni.

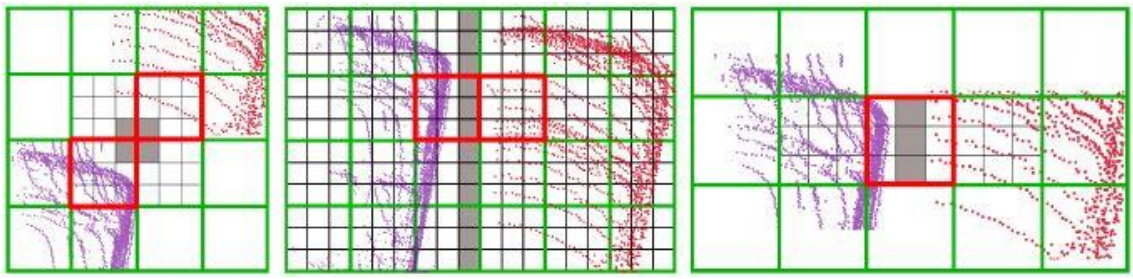


14. ábra: Hierarchikus szabályos rács

A szerzők kisebb felbontású rácsot arra használják fel, hogy ennek a celláit besorolják különböző osztályokba, amik a következők: talaj, alacsony objektum, fal, ritkás cella. A régiókba való besorolás a cellák lokális tulajdonságai alapján történik. Ritkás osztályba kerül egy cella, ha fölötte nem található pont. Talaj osztályú lesz egy cella, ha a cellán belüli magasságkülönbség, illetve a lokális maximális magasság kisebb egy-egy meghatározott értéknél. Azok a cellák, melyek nem ritkás és nem talaj osztályúak, a lokális maximális magasságértékük alapján kerülnek besorolásra a fal illetve az alacsony objektum kategóriákba. Itt meg kell még jegyezni azt, hogy a fal osztályba nem csak faldarabok, hanem további magas objektumok is bekerülnek (pl.: oszlopok, táblák, fák). A fal cellák 2D képre vetítésével és valószínűségi Hough-transzformáció alkalmazásával azonban a falak pozíciója becsülhető, és topológiai alapon az objektum és a fal osztályú cellák meghatározása pontosítható.

3.1.2.2 Objektumok összekapcsolása

A szerzők miután a durva felosztású rács celláit besorolták szemantikai régiókba, eldobják a talaj-, a ritkás-, és a magas objektum szegmenseket, majd a fennmaradó alacsony objektum cellák között próbálják megtalálni az azonos objektumhoz tartozókat. Ehhez első lépésben a szomszédos cellák közötti magasságkülönbség vizsgálatával döntik el, hogy azok ugyanazon objektumhoz tartozhatnak-e. Amíg a magasságkülönbség nem halad meg egy küszöbértéket, addig a két cellát „összevonják". Cellák összevonása úgy történik, hogy ugyanazt az ID-t kapják meg. Az összekapcsoló algoritmus kimenete egy számozott rács, melyben azonos számok jelölik az azonos objektumhoz tartozó cellákat. Ilyen módon kaphatunk egy durva becslést az objektumokra. Azonban az egymáshoz közel helyezkedő objektumok nem mindig különíthetők el ily módon, ezért kerül felhasználásra a finom felosztású rács. Ezen belül vizsgáljuk a cellákon a pontsűrűséget, és ha valahol jelentős csökkenést találunk egy hosszabb szakasz mentén, akkor elkülönítjük a korábban tévesen összekapcsolt objektumokat. Ezt a lépést szemlélteti a 15. ábra.



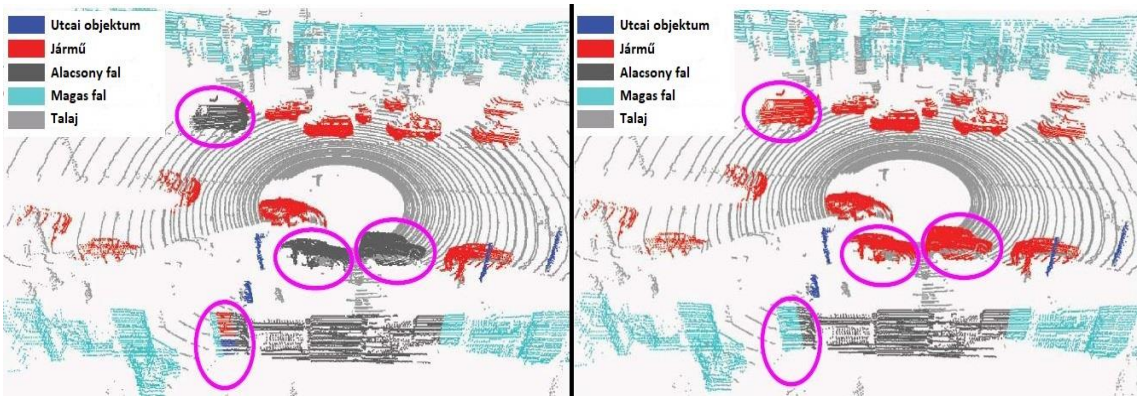
15. ábra: Közeli objektumok elkülönítése finom felosztású rácson. Zöld: durva felosztású rács; fekete: finom felosztású rács.

A cellák mérete jelentősen befolyásolja az algoritmus futási sebességét és hatékonyságát. *[3]* kitér a cellák méretének optimális megválasztására is, a hierarchikus rács alapú módszerhez 40-50 cm-es cellákat javasol.

3.1.2.3 Objektumok felismerése

Az eddig részletezett 2D hierarchikus rácson alapuló objektum észlelést alkalmazza *[4]* is, viszont ennél tovább megy: az objektumok felismerésre is bemutat egy lehetséges módszert. A felismerés alapja egy konvolúciós neurális hálózat (Convolution Neural Network – CNN). A hálózat tanítási adataira a cikk írói létrehoztak egy adatbázist, mely kézzel felcímkézett objektumokból áll (2063 db). A létrehozott hálózat képes utcai

bútorok (táblák, padok, stb.), alacsony faldarabok, autók, és gyalogosok felismerésére. Néha azonban téveszt, például éppen egymás mellett elhaladó autók, vagy nagy felületű járművek esetén (furgonok, teherautók). Sokszor több kategóriára is magas valószínűséget ad a neurális hálózat, ilyenkor előfordul, hogy a járművet alacsony falnak veszi, vagy fordítva. Az ilyen jellegű tévedéseket kiküszöbölhetjük topológia-alapú vizsgálattal (16. ábra). Amennyiben vesszük a detektált járművek és alacsony falak valódi faltól vett távolságát, akkor a neurális hálózat döntését szükség esetén felülbírálhatjuk.

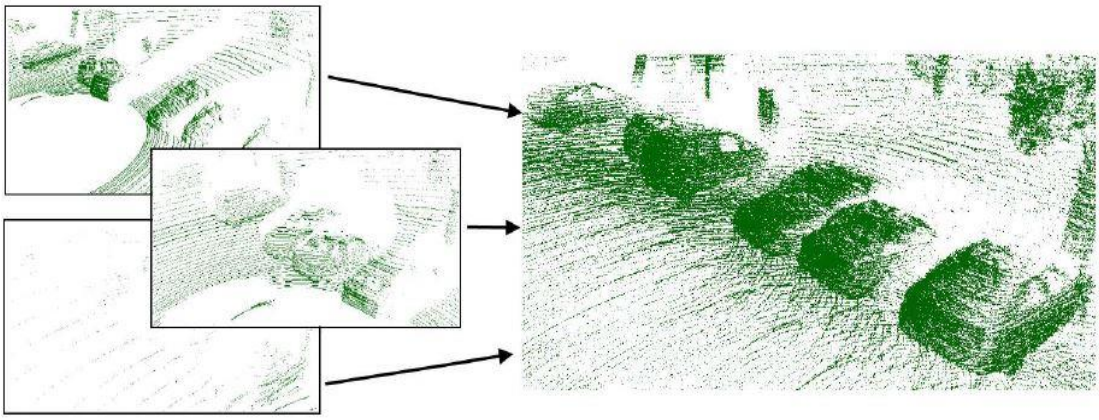


16. ábra: Megjelenés alapú versus megjelenés és topológia alapú osztályozás

3.1.2.4 Egymást követő pontfelhők regisztrációja

[5] bemutat egy eljárást, ami az egymást követő pontfelhőket képes térben összeilleszteni, amivel a pontfelhő sűrűsége növelhető. Emellett az átfedés és az egyetlen nézőpont jelentette problémák is csökkennek, ugyanis több nézőpontban készült pontfelhők kerülnek összevonásra. Az algoritmus a normális eloszlású transzformáción (Normal Distribution Transform – NDT) alapul.

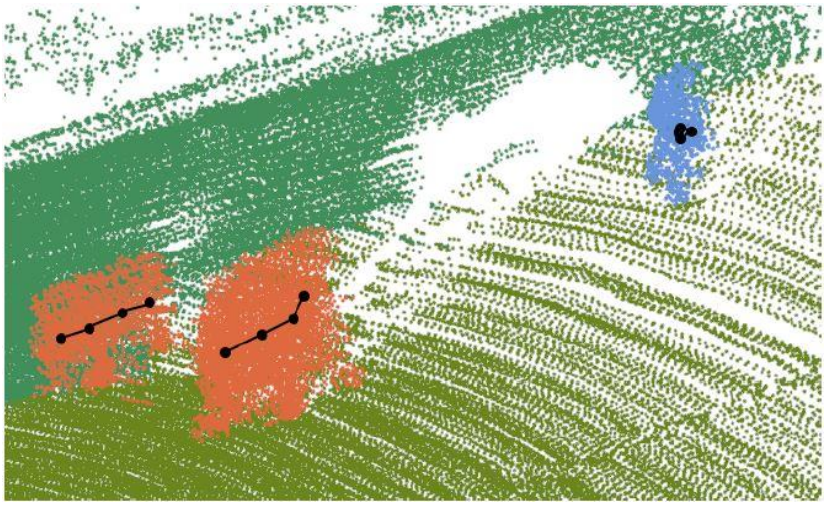
Ezt az eljárást a talajon található „gyűrűk”, és a mozgó objektumok megtéveszthetik. Ezért a cikk írói a korábban részletezett osztályozás után az eljárást csak a fal osztályú (magas) objektumokra futtatják le. Ezzel a talajpontok és a mozgó objektumok sem fogják megtéveszteni a műveletet, hiszen előbbiek a talaj cellákhoz, míg utóbbiak az alacsony objektumokhoz tartoznak. Gondot okozhat, hogy egymást követő pontfelhőkön az objektumok más-más nézőpontból máshogy néznek ki, viszont ez a különbözőség a magas objektumokra kisebb mértékű. A fal osztályú cellákra (magas objektumok) lefuttatott NDT tehát jóval pontosabb eredményt ad, mintha az egész felhőre futtatnánk le azt.



17. ábra: Egymást követő pontfelhők regisztrációjának eredménye

A regisztráció során kapott pontfelhő láthatóan nagyobb sűrűségű, és az objektumok jobban hasonlítanak egymásra, mert több szemszögből is rögzítve lettek. Ezek a tények a környezet magasabb szintű értelmezését teszik lehetővé azáltal, hogy segítik az objektumok felismerését.

A regisztráció során az időbélyegeket is figyelembe veszik a szerzők. Az időben túlságosan távoli képet már nem illesztik hozzá az egyesített felhőhöz. Ezek az időbélyegek használhatók mozgás észlelésére is. Amennyiben egyes objektumok súlypontjának pozícióját minden időpillanatban eltároljuk, akkor annak érzékelhetjük nagyobb mértékű elmozdulását, vagy épp a helyben maradását. Ezt a jelenséget a 18. ábra szemlélteti.



18. ábra: Mozgás detektálása regisztrált pontfelhőkben

3.2 Mobil térképező rendszerek mérésein alapuló analízis

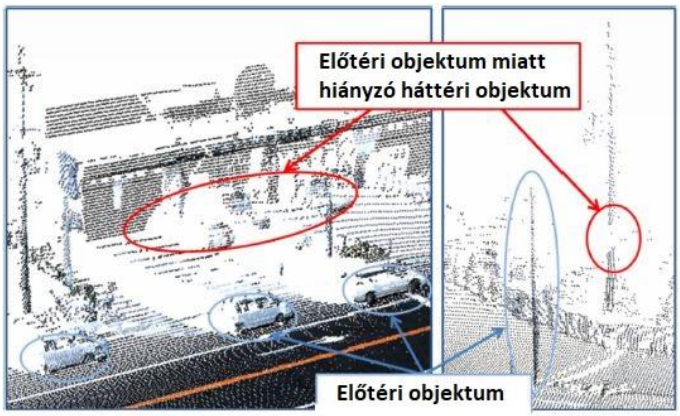
Nagy felbontású mobil lézerszkenner rendszereket (autóra vagy hajóra szerelve) használunk különböző térképező eljárások során. A lézerszkenner mellett pozícióinformációt (GPS) illetve a térbeli helyzetet (IMU) érzékelő egységeket is alkalmazunk, hogy a pontfelhők illesztése minél pontosabb legyen. Az eljárás útszakaszok, vagy akár teljes városok feltérképezésére használható. Az így nyerhető adatokból az úthálózattal kapcsolatos fejlesztési irányokat lehet meghatározni, illetve adatbázisokat lehet létrehozni fontosabb utcai objektumokról (jelzőtáblák, zebrák, stb.).

3.2.1 Nehézségek

A mobil térképező rendszerek során felmerülő nehézségek hasonlóak a valós idejű lézerszkennereknél tapasztaltakhoz. Itt is előfordulhat, hogy egy objektum kitakarja a mögötte elhelyezkedő objektumokat (19. ábra). Azonban mivel több szögből is felvétel készül ugyanazon helyszínről, ezért ez a jelenség sokkal ritkábban fordul elő, és kiküszöbölhető akár a helyszín többszöri felmérésével.

A pontfelhő itt jelentősen sűrűbb, mint a valós idejű lézerszkennereknél, így az objektum elkülönítés és -felismerés valamivel egyszerűbb feladat. Azonban az adatok mennyisége is nagyságrendekkel nagyobb, így az ezeken futtatott algoritmusok nem képesek valós időben működni, sokszor 10 másodpercekben mérhető a futási idő.

Mivel a térképező rendszerek számára csak a statikus, ember által létrehozott objektumok érdekesek, ezért problémát jelent a pontfelhőben dinamikus objektumok (gyalogosok, autók) jelenléte. Ezen kívül legtöbbször a növényzet (bokrok, fák) is nemkívánatos, zavarja az objektumok elkülönítését, felismerését.



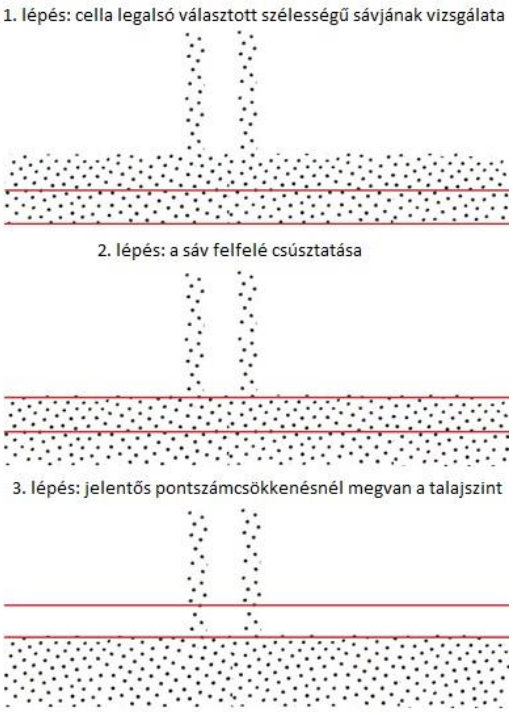
19. ábra: Kitakarás mobil térképező rendszer pontfelhőjében

3.2.2 Szakirodalmi eredmények

Az objektumfelismerést célzó algoritmusok hasonló felépítésűek mobil térképező rendszerek pontfelhőin, mint a kompakt Lidaros érzékelők esetén. A módszerek útfelszín detektálással, majd eltávolítással kezdődnek. Ezt követően történik meg az objektumok elkülönítése, hiszen az ezeket összekötő talajpontok eltávolításra kerültek már. Az objektumok elkülönítésére alkalmazhatók műveletigényes algoritmusok is mobil térképező rendszerek pontfelhői esetén, hiszen a valós idejű működés itt általában nem követelmény. Az elkülönített objektumok lokális és globális tulajdonságai az elkülönítés után már vizsgálhatók. Különféle mély tanuláson alapuló módszerek léteznek az objektumok felismerésére.

3.2.2.1 Útfelszín eltávolítása

Az útfelszín detektálása többféleként megvalósítható. Ezek alapja szabályos rács alapú felosztás, mely során a cellák azon pontokat tartalmazzák, melyek vetülete rájuk esik. A cellákat innentől külön-külön vizsgáljuk, a talaj megtalálásához többféle lokális tulajdonság vizsgálata lehetséges. Az önálló laboratórium feladatom során a cella alsó pontjától felfelé elindulva sávokat vizsgálva kerestem azt a magasságot, ahol drasztikusan csökken a felhő pontszáma, majd ezt definiáltam talajszintnek (20. ábra).



20. ábra: Pontszám csökkenés alapú talajészlelés

A talajészlelés ezen kívül történhet még a cella feletti szintkülönbség és globális magasságérték felhasználásával. A talajt tartalmazó cellák közel sík felületet tartalmaznak, így a szintkülönbség nem lehet túlságosan nagy. *[6]*-ban a szerzők magasság irányú szórásérték alapján döntik el, hogy egy-egy cella talajt tartalmaz-e. Természetesen ekkor is szükség van magasságértékekre, hogy kiszűrhessék a magasan elhelyezkedő sík felületeket.

3.2.2.2 Objektumok szegmentációja

A talaj leválasztása után megszűnik az objektumok közötti összeköttetés, így az objektumok elkülöníthetők. Nem követelmény a valós idejű működés, ezért itt akár fagráf-alapú struktúrák is felhasználhatók a hatékonyság növeléséhez.

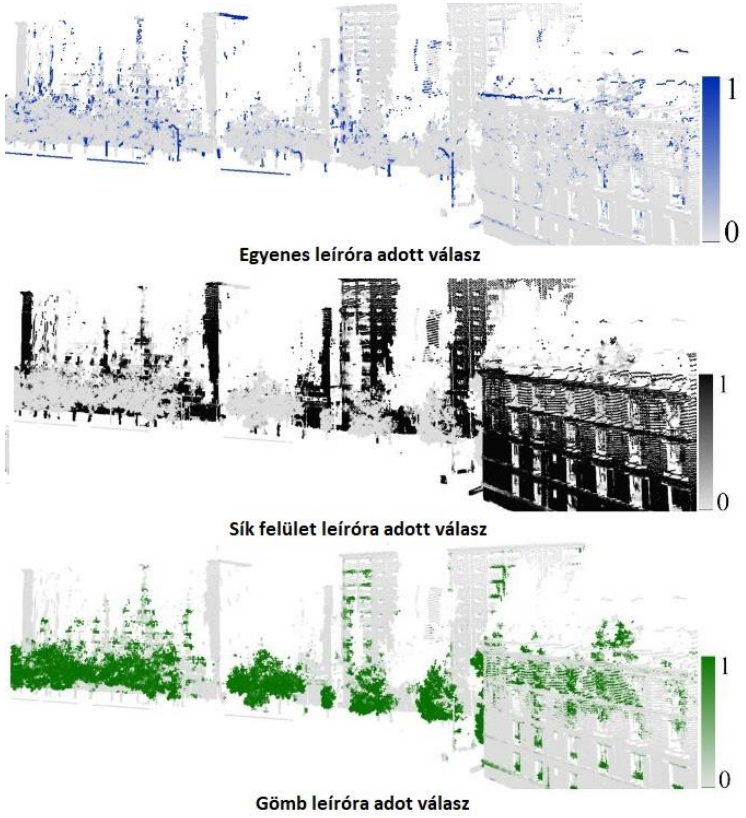
[6]-ban a szerzők szabályos rács alapú szegmentációt alkalmaznak. A pontokat 2D képre vetítik, majd ezen alkalmaznak összetartozó objektumokat megkereső eljárást (Connected Component Labelling). Kétféle probléma merülhet fel ennél az eljárásnál: különálló objektumok összetapadhatnak, vagy bizonyos objektumok széteshetnek több darabra. A szerzők itt azt szeretnék elkerülni, hogy az objektumok több részre essenek szét, ezért alulszegmentálják a felhőt. *[8]*-ban a szerzők elárasztásos módszerrel keresik az összetartozó objektumokat. Kiválasztanak egy gyökérpontot, majd ennek keresik a bizonyos távolságon (a szerzők 25 cm-t választottak) belül eső szomszédjait. Ezt követően az összes szomszédnak megkeresik az adott távolságon belüli szomszédjait, amíg van még új szomszéd. Ha már nincs új szomszéd, akkor felderítettek egy objektumot, új gyökérpontot keresnek, és az egész eljárás kezdődik elölről. Ez addig megy, amíg be nem járták a teljes felhőt.

21. ábra: Szegmentációs lépés kimenete [8]-ban

3.2.2.3 Fák detektálása

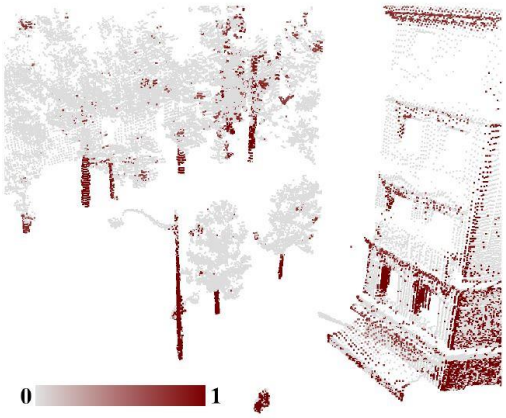
[7]-ben bemutatásra kerül egy módszer fák detektálására. Sok feladatnál a fákkal nincs tennivaló, mégis megtéveszthetik az algoritmust (pl. oszlopfelismerést), ezért nemkívánatosak. Vannak azonban olyan alkalmazások is, melyek célja a város zöld területeinek feltérképezése. Mindkét esetben felhasználhatjuk a fákat felismerő algoritmusokat.

A *[7]*-ben leírt módszer alapja, hogy geometriai leírók alapján vizsgálja a pontok környezetét. A szomszédság homogenitását, pont-felhalmozódásait vizsgálja a leíró, és ez alapján számít valószínűséget arra, hogy az adott pont egyenesnek, síknak, vagy gömbszerű felületnek a része. Az így kapott eredményképek még meglehetősen zajosak, ezért a szerzők bevezetnek egy valószínűség-alapú felülvizsgáló lépést. E lépés során nem csak egy adott pont valószínűségét nézik, hanem a környezetében lévő pontokét is. Iteratív módon frissítik minden egyes pont valószínűségi értékét a szomszédjainak valószínűségi értékei alapján. Így a kapott eredménykép már jóval letisztultabb lesz. A különböző leírókra a szomszédok valószínűségeit is figyelembe véve adott választ szemlélteti a 22. ábra.



22. ábra: Különböző geometriai leírókra adott válasz

A cikkben bevezetésre kerül egy henger-leíró is (23. ábra). Ez egy RANSAC (Random Sample Consensus) alapú eljárás, mely vizsgálja minden pont k darab legközelebbi szomszédját, és hengert próbál rájuk illeszteni. Az illesztés sikerességének függvényében visszatér egy valószínűségi értékkel, illetve az illesztett henger sugarával.



23. ábra: Henger leíróra adott válasz

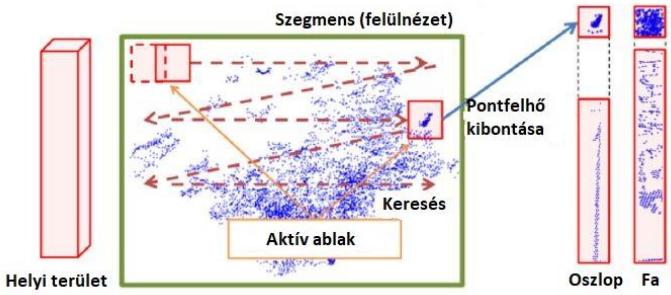
Maga a fa észlelés úgy működik, hogy a henger- és a gömbszerű leíróra adott válaszokat 2D képre vetítik a vízszintes síkra, majd ezen keresik azokat a henger felületeket, melyeket gömbszerűek vesznek körül (a lehetséges fatörzsek közül megkeressük azokat, melyekhez tartozik lombkorona). Lehetőség van a fák elkülönítésére is oly módon, hogy a lombkoronák pontjait egyesével hozzárendeljük a hozzájuk legközelebb álló fatörzshöz.

24. ábra: Fa-detektáló algoritmus kimenete

Gyakori hiba, hogy a fák alatti jelzőtáblákat, oszlopokat fatörzsnek gondolja az algoritmus. Ezt azonban könnyen kiküszöbölhetjük, ehhez elegendő mindössze a lézerszkenner által visszaadott intenzitásértékeket figyelembe venni. A fémtárgyakról visszavert lézersugár ugyanis nagyobb intenzitással érkezik vissza a szenzorhoz.

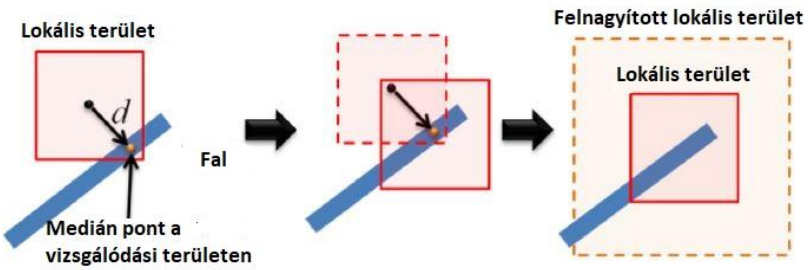
3.2.2.4 Falak, oszlopok, korlátok detektálása

Falak, oszlopok, korlátok detektálásáról *[6]*-ban olvashatunk bővebben. A megvalósított algoritmus leginkább statisztikák vezetéséhez használható fel. A módszer első lépését a fentebb említett talajdetektáló és objektum-elkülönítő algoritmus adja. Ezt követően a szerzők a képződött objektum szegmenseket nem egészükben, hanem lokális tulajdonságaik alapján vizsgálják. A szegmenseket négyzet alapú hasábbal járják be a vízszintes sík mentén, és ezen a hasábon belül határoznak meg tulajdonságokat (25. ábra). Ha nem is sikerül különválasztani egy oszlopot egy fától, még akkor is észlelhetjük az oszlopot a szegmensen belüli lokális tulajdonság alapján.



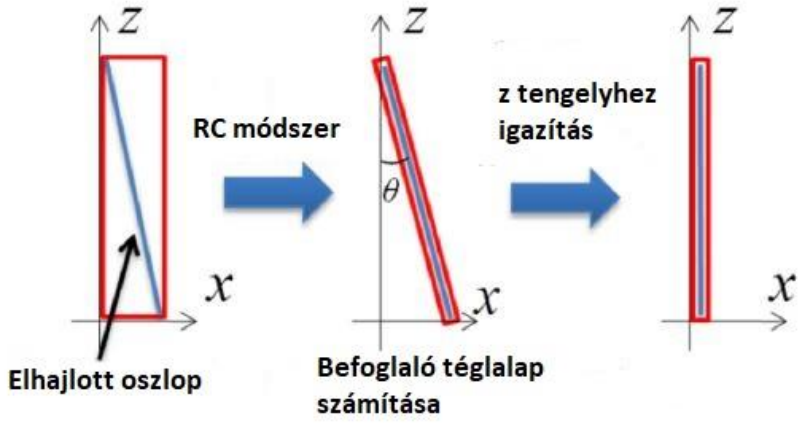
25. ábra: Szegmens bejárása négyzet alapú hasábbal

Azonban a vizsgálódási ablak pozíciójától függően téveszthet a módszer. Ez legtöbbször akkor fordul elő, ha egy fal szegmens csak a hasáb sarkába kerül bele. Ilyenkor az algoritmus a fal részleteit oszlopnak hiszi. Ezt a szerzők úgy küszöbölik ki, hogy ha a vizsgálódási ablak középpontjától túlságosan távol vannak az ablakon belüli pontok, akkor az ablakot eltolják, hogy a pontok átkerüljenek az ablak középpontjába, illetve annak méretét felnagyítják. Így már kellően nagy része az objektumnak bekerül az ablakba, ami nagyban segíti a pontos felismerést.



26. ábra: Vizsgálódási ablak eltolása, nagyítása

A szerzők bemutatnak még egy módszert, mellyel a nem teljesen függőleges oszlopokat is fel lehet ismerni. A módszert „Forgó körző módszer”-nek (Rotating Calipers method) nevezik, lényege pedig az, hogy az oszlopra minimális befoglaló téglatestet illeszt, majd ennek a z tengellyel bezárt szögét kiküszöböli forgatással. Ennek a segítségével a ferdén álló oszlopok is felismerhetők.



27. ábra: Forgó körző módszer

A felismerés mély tanulás módszerén alapul. A lokális vizsgálódási ablakok felcímkézésével hozhatunk létre a hálózatnak tanító adatokat, ami alapján később a felismerés lehetséges lesz.

4 A megvalósított algoritmus

Ahogy a korábbi eredmények ismertetése során leírtam, 64 csatornás Lidarra már fejlesztésre kerültek talajpont-, illetve objektum észlelő algoritmusok. A 16 csatornás Lidar kimenete jóval ritkásabb pontfelhő, mivel a vertikális felbontása 2°, szemben a 64 csatornás Lidar 0,5°-os felbontásával. A távolság függvényében a pontok négyzetméterre eső sűrűsége is jelentősen csökken, ezért különösen a távolabbi objektumok észlelése nehezebb, mint a nagyobb Lidaroknál.

Az alacsony sűrűséget ellensúlyozandó, két 16 csatornás Lidart használtunk fel a feladat során. Ezek szolgáltatnak egy-egy pontfelhőt, ezeket fogjuk összeilleszteni egy meghatározott transzformációval. A két szenzort időszinkronban működtetjük, hogy a két pontfelhő azonos időpillanatban reprezentálja a környezetet. A két szenzor közti transzformációt a mérés előtti kalibráció során határozzuk meg a *[9]*-ben bemutatott módszer alapján. Ennek lényege, hogy a már autóra szerelt szenzorok látóterébe dobozokat helyezünk el úgy, hogy mindkét szenzor felhőjében látsszon azoknak három különböző irányú élük. Ezt követően rögzítünk mindkét szenzorral egy-egy pillanatképet, és ezeken a pillanatképeken meghatározhatjuk a pontfelhők közötti transzformációt. A kalibráció eredménye egy forgatásmátrix illetve egy eltolás vektor, melyek segítségével a két pontfelhő közös koordináta-rendszerbe hozható.

A két Lidar használatának legnagyobb előnye, hogy kétszer olyan sűrű felhőt kapunk, mint egyetlen Lidar használata esetén. Azonban az elrendezés további előnye az is, hogy mivel két különböző nézőpontból észleljük a környezetet, ezért jóval kisebb a közeli objektumok okozta holttér, és a kitakarási jelenség jelentősége.

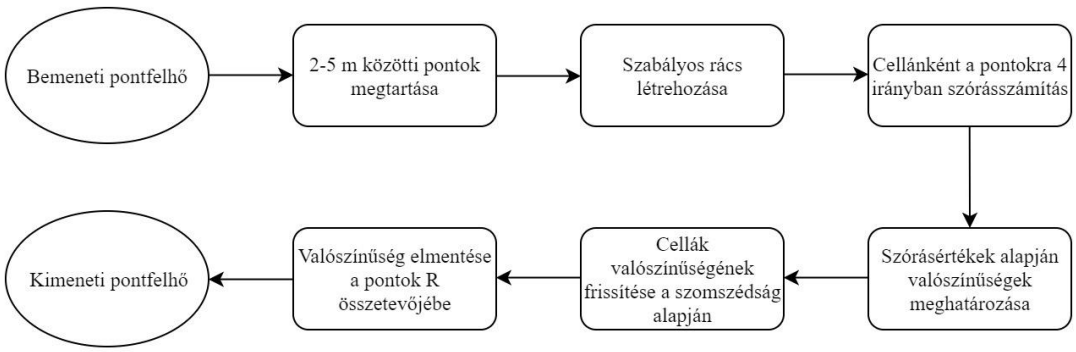
A VLP16-os szenzorokra megvalósított algoritmus alapjait a *[3]*-ban és *[4]*-ben leírt módszerek képzik majd. 2D szabályos rácsfelosztást alkalmazok, majd a rács celláinak lokális tulajdonságai alapján azokat besorolom különböző szemantikai osztályokba: talaj, alacsony objektum, fal (magas objektum), ritkás cella. A talaj osztályú cellákkal a továbbiakban nem foglalkozok. A szabályos rács „alacsony objektum” osztályú celláit bejárva, azok szomszédjainak tulajdonságait figyelembe véve eldöntöm, hogy mely cellák tartoznak ugyanazon objektumhoz, és melyek nem. Az összekapcsolt cellák csoportjai fogják adni az észlelt objektumokat.

4.1 Lombkorona pontok észlelése

A már megvalósított talajpont- és objektumészlelés tesztelése során felmerült egy probléma. Előfordul, hogy egy-egy objektum fa alatt található. Ilyenkor egy-egy objektum darabot tartalmazó cellát magas objektum osztályba sorolunk, hiszen bár az objektum 2 m-nél alacsonyabb, de felette találhatók pontok a fa lombkoronájából, ami miatt a maximális magassága meghaladja a 2 m-t (2 m a határ az objektum és a fal (magas objektum) osztály között). Ez a hiba azt okozza, hogy a teljesen vagy részben fa alatt elhelyezkedő objektumokat nem, vagy csak darabokban vagyunk képesek észlelni. Ennek a problémának a megoldására fejlesztettem egy lombkorona detektáló algoritmust. A fák lombkoronájának elhagyása után az alattuk található objektumok is felismerhetők lesznek.

A lombkorona pontok felismerését még a szegmentációs lépés előtt végre kell hajtani. Az eljárásnak természetesen kis műveletigényűnek kell lennie, hiszen a többi lépéssel együtt is valós időben működnie kell. A pontfelhő alacsony sűrűsége miatt PCA-alapú (Principal Component Analysis – Főkomponens-analízis) megoldások nem működőképesek.

Az általam megírt lomkorona felismerő algoritmus két részből áll. Az első rész a lombkoronák magasságában elhelyezkedő pontokra határoz meg valószínűséget arra, hogy az adott pont lombkoronának része. Ehhez szabályos rács struktúrának a celláin belül vizsgálok szórásértékeket. Ezt a részt a 4.1 fejezetben fogom bemutatni. Az algoritmus második része a szegmentációs lépés során kerül végrehajtásra. A korábbi valószínűségértékek és a magas pontok alatti környezet vizsgálata alapján döntök a magas pontokról. Ha egy pontot lombkoronapontnak veszek, akkor azt eltávolítom.



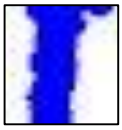
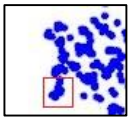
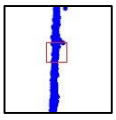
28. ábra: Lombkorona észlelő algoritmus lépései

4.1.1 Bemeneti felhő formálása

A bemeneti felhő a két Lidartól kapott egyesített pontfelhő. Először ebből a felhőből kiválasztom azokat a részeket, amelyekben lombkoronát szeretnék keresni. Az 5 m feletti pontokat elhagyom, ugyanis sem a talaj-, sem az objektumok észlelése szempontjából nincs jelentőségük. A 2 m-nél alacsonyabban lévő pontokat változtatás nélkül megtartom, mert a fák lombkoronája általában efölött kezdődik. Számomra érdekesek tehát a 2 m és 5 m közötti pontok, a továbbiakban ezeket fogom vizsgálni. A kimenő pontfelhőbe ezekhez a pontokhoz az R összetevőbe fogom menteni annak a valószínűségét, hogy az adott pont lombkorona része.

4.1.2 Szabályos rács struktúra létrehozása

A legtöbb 2 m és 5 m közötti pont falak és lombkoronák részeiből származik, ezeket kell megkülönböztetni. Erre a forrásokban is említett szabályos rács felosztást használom fel, ennek a celláin belül vizsgálom a különböző irányú szórásokat. A falaknál valamelyik irányú szórás elég kicsi, míg a lombkoronáknál várhatóan több irányú szórás is nagynak mutatkozik. Az általam alkalmazott szabályos rács viszont méreteiben különbözni fog a forrásokban (*[3]*, *[4]*) említettektől. Egyrészt azért, mert kis cellaméretű rendszer használatával jelentősen megnövekedne a futási idő, amivel akár meghaladhatnánk a valós idejű működést. Másrészt pedig azért, mert ahhoz, hogy fal esetén valóban kicsi legyen a szórás, a celláknak magukba kell foglalniuk a falat teljes szélességében, ugyanis csak ekkor lesz különböző a lombkorona és a fal ponthalmazok szórásképe.

Cellaméret	Lombkorona	Fal
0,3 x 0,3 m		
1,5 x 1,5 m		

29. ábra: Kisebb és nagyobb cellaméret esetén lombkorona és fal szegmensek

A 29. ábra bemutatja, hogy kis cellaméret esetén a vízszintes irányú minimális szórás hasonló mértékű, azonban nagyobb cellaméret alkalmazásával a vízszintes irányú szórás jelentősen eltér. A cellák méretét ezért 1,5 m-esnek választottam. A struktúra kialakítása során az első lépés az x és y koordináták minimumának és maximumának meghatározása. A második lépés a cellaméret ismeretében ezekből a minimum és maximum értékekből a szükséges cellarendszer lefoglalása. A harmadik lépés pedig a pontok cellákhoz rendelése a z = 0 síkra vett vetületük alapján.

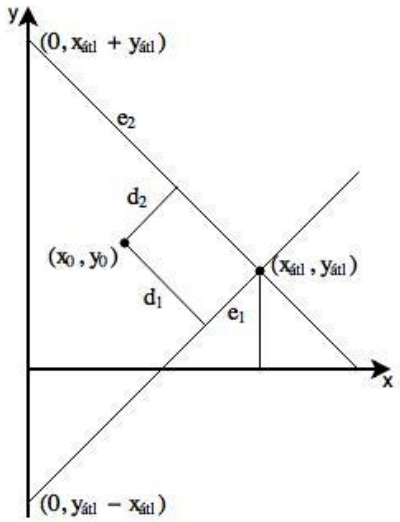
4.1.3 Szórásértékek számítása

Minden cellára négy irányban fogok szórást számolni. Azért van erre szükség, mert nem biztos, hogy a falak párhuzamosan futnak az autó mozgásának irányával, ami miatt két irány vizsgálata nem elégséges. A szabályos rács oldalai mindig párhuzamosak a mozgás irányával, így előfordulhat, hogy a falak két irányba is nagy szórást mutatnak. Négy irány vizsgálata esetén a falaknál lesz legalább egy olyan irány, ahol a szórás kellően kicsi lesz. A négy irány a négyzet alakú cellákra az oldalakkal párhuzamos irányok, illetve az átlókkal párhuzamos irányok. Az átlók irányában nézett szórást a Hesse-féle normálalakból származó távolságszámítási módszer alapján számolom ki:

$P(x_0, y_0) pontnak\ az\ egyenestől\ mért\ távolsága$:

$$d = \left| \frac{A \cdot x_0 + B \cdot y_0 - C}{\sqrt{A^2 + B^2}} \right|$$

$Ahol\ az\ egyenes\ egyenlete: A \cdot x + B \cdot y = C.$



30. ábra: Hesse-féle normálalakból távolság számítása

Az egyenesek egyenletei:

$$e_1\ egyenlete: 1 \cdot x + (-1) \cdot y = y_{átl} - x_{átl}$$

$$e_2\ egyenlete: 1 \cdot x + 1 \cdot y = x_{átl} + y_{átl}$$

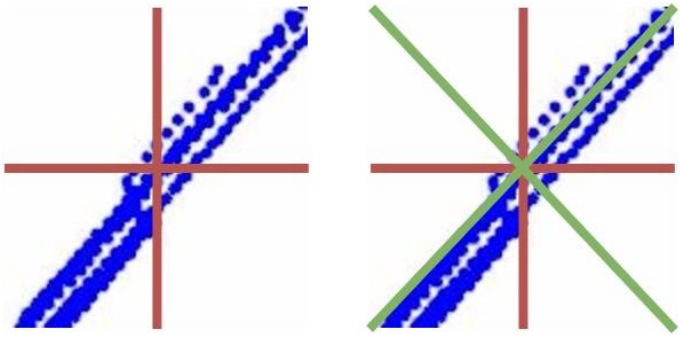
Ezekből:

$$d_1 = \left| \frac{x_0 - y_0 - (y_{átl} - x_{átl})}{\sqrt{2}} \right|$$

$$d_2 = \left| \frac{x_0 + y_0 - (x_{átl} + y_{átl})}{\sqrt{2}} \right|$$

A 31. ábra bemutatja, hogy két irány vizsgálata esetén mindkét irányban nagy szórást tapasztalunk, azonban négy irány vizsgálata esetén az egyik irányba jelentősen kisebbet.

Miután megvan a négy szórásérték, azokat növekvő sorrendbe rendezem. Szükségem van ezekből az értékekből robosztus leírók kinyerésére. Az általam választott leírók a következők lesznek: minimális szórásérték, második és harmadik legnagyobb szórásértékek összege. A legkisebb szórásérték falak esetén nagyon kicsi lesz, addig lombkorona esetén ennél nagyobb. A második és harmadik legnagyobb szórásértékek összege falak esetén kis érték lesz, míg lombkorona szegmensek esetén ennél jelentősen nagyobb. Azért nem vizsgálom a legnagyobb szórásértéket, mert az falak esetén is nagy érték lehet, ami megtéveszthetti az algoritmust.



31. ábra: 2 irányú szórásvizsgálat versus 4 irányú szórásvizsgálat

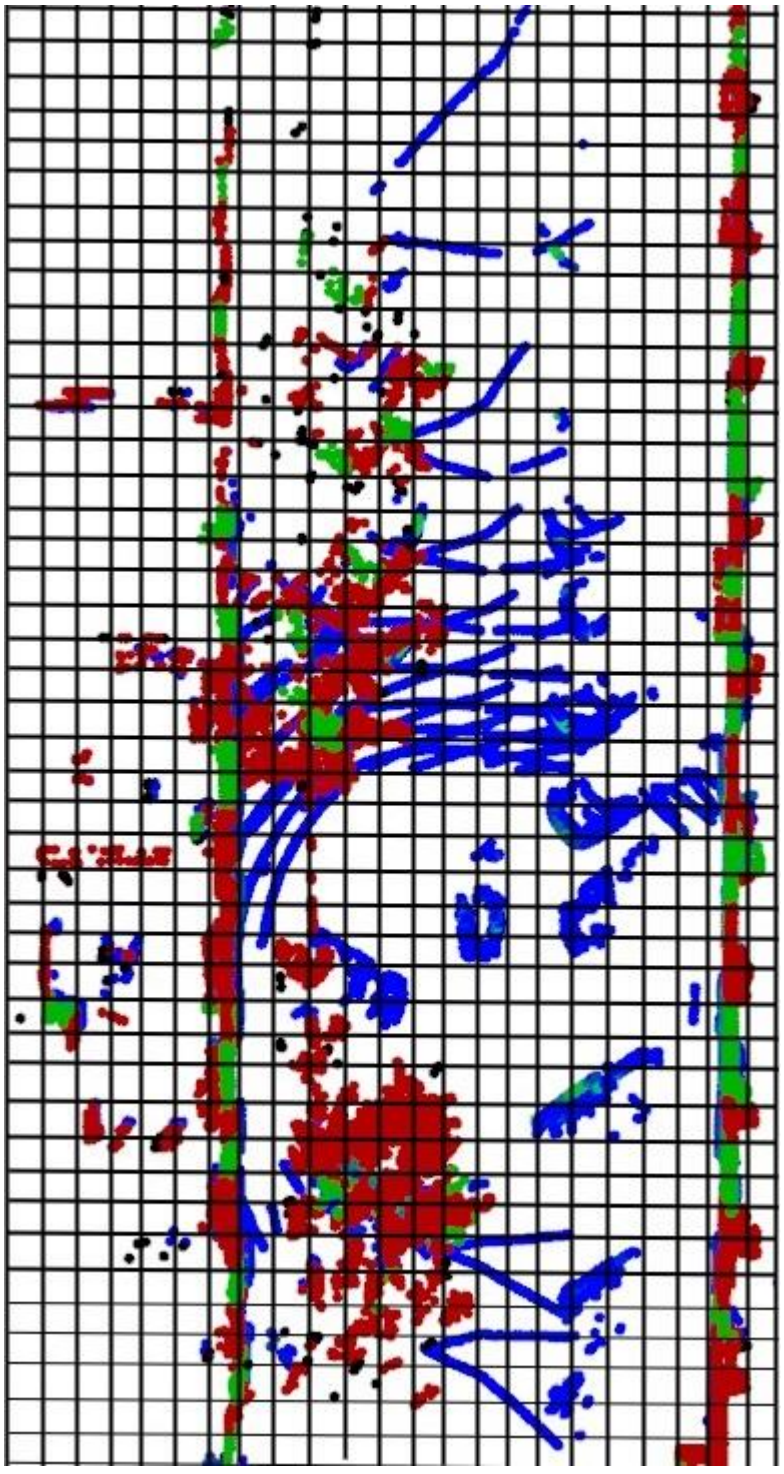
A tesztelés során azt figyeltem meg, hogy a két leíró vizsgálata nem ad kellően biztos eredményt a lombkorona pontok észlelésére. Ezért a módszert úgy fejlesztettem, tovább, hogy a számolt szórásértékekhez valószínűséget rendeltem. Így egy celláról döntést hozhatok nem csak a saját, de a szomszédjainak valószínűségei alapján is. A 33. ábra bal oldala szemlélteti a leírók alapján hozott döntés szerinti optimális kimenetet valószínűségek nélkül. A megfelelő magasságban zöld jelöli a lombkorona pontokat, piros a nem lombkorona pontokat. Kék színűek a 2 m-nél alacsonyabban található pontok.

4.1.4 Valószínűségértékek szórásokhoz rendelése

A következő feladat tehát az, hogy a kétféle leíró különböző értékeihez valószínűségeket rendeljek. Ehhez először is meg kell határoznom a vizsgált szórástartományt. Két határérték lesz: amelyik esetén egyáltalán nem találok lombkorona pontokat, illetve amelyik esetén az összes lombkorona pontot megtalálom. Az így kapott tartományt felosztom egyenlő nagyságú résztartományokra. A résztartományokhoz úgy rendelek valószínűségeket, hogy először a résztartományba eső cellákat zöldre színezem. Ezután megszámolom a helyes és a helytelen találatokat, majd ebből kiszámolom a helyes találat valószínűségét az adott szórás-résztartományra. Ezt a műveletet mindkét leíróra elvégzem azok általam meghatározott szórástartományra.

A 32. ábra egy példát mutat a valószínűség számítására. A minimális szórás leírót vizsgálja 0,03-0,04-es szórás-résztartományra. 14 db helyesen detektált lombkorona cellát számolhatunk, illetve 36 db helytelenül detektáltat. Az észlelés valószínűsége tehát a helyszínen $\frac{14}{50}(= 0{,}28)$. Természetesen a végleges valószínűség adott szórás-résztartományra több helyszín átlagából kerül majd meghatározásra.

Miután minden tartományhoz kiszámoltam a valószínűséget, képes vagyok az adott cella szórásértékeiből következtetni arra, hogy az mekkora valószínűséggel lomkorona, és mekkora valószínűséggel nem az. Ez a valószínűség tehát a teljes szabályos rácsban rendelkezésemre áll kétféle leíró alapján. Így egy cella valószínűségét a szomszédjainak figyelembe vételével határozhatom meg, ami jelentősen javítja a pontosságot. Ezt szemlélteti a 33. ábra.

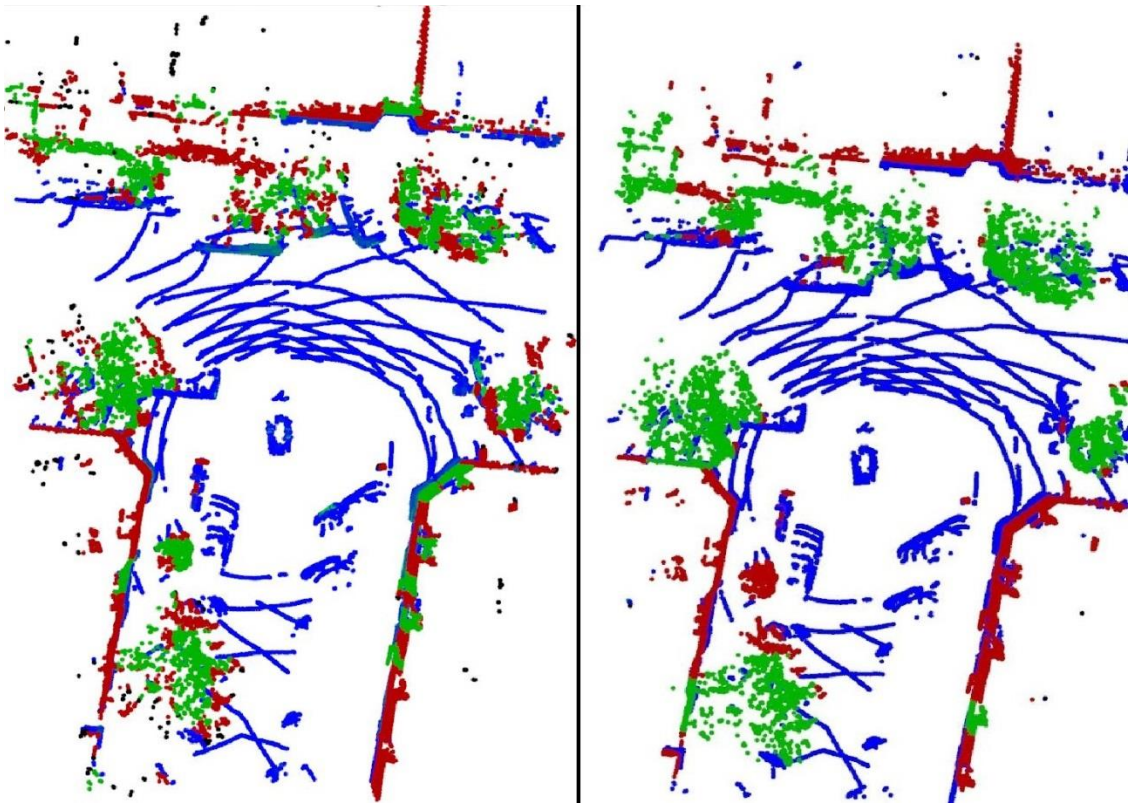


32. ábra: Valószínűségek szóráshoz rendelése

4.1.5 Valószínűségek frissítése a szomszédság alapján

Ebben a lépésben egy cella valószínűségét frissítem a szomszédjainak valószínűségei ismeretében. Ezt egy egyszerű súlyozott átlagszámítással teszem meg, ahol a vizsgált cella valószínűsége 3-as súllyal szerepel, a szomszédjainak a valószínűsége pedig 1-essel. Így kiküszöbölhető, hogy a lombkorona közepén egy cellát ne ismerjünk fel, hiszen a szomszédjai az esetleges alacsony valószínűséget felhúzhatják annyira, hogy mégis lombkoronának detektáljuk. Ugyanez fordítva is működik, ha egy fal szegmensnek véletlenül túl nagy a valószínűsége, akkor a szomszédjai azt még lehúzhatják, amivel a téves észlelés elkerülhető.

A valószínűségek cellánkénti frissítése után optimális valószínűségi küszöböt választva ($p_{küszöb} = 0.5$) az algoritmus kimenete jelentős javulást mutat. A lombkorona pontokat ebben a lépésben még nem tüntetem el, hanem a valószínűségeket tovább viszem a pontok R összetevőjében, és majd a szegmentációs lépésben használom fel.



33. ábra: Lombkorona felismerés valószínűségek nélkül (bal oldal), és valószínűségekkel(jobb oldal)

4.2 Talajpontok azonosítása

4.2.1 Szabályos rács struktúra létrehozása

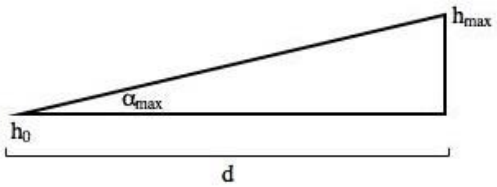
Az algoritmus bemenete 2 db időszinkronizált VLP16-os Lidar azonos koordináta-rendszerbe transzformált pontfelhője. Ebben a pontfelhőben a 2 m és 5 m közötti pontok r összetevője tartalmazza annak a valószínűségét, hogy az adott pont lombkoronapont. Az első lépés a szabályos rács struktúra felépítése. A szabályos rácsot a z = 0 síkra képzelhetjük el. Négyzet alakú cellákra bontom ezt a síkot, majd minden cellához hozzárendelem azokat a pontokat, melyeknek a vetülete az adott cellára esik. Minden cellához elmentem a fölötte található pontok három koordinátájának szélsőértékeit.

A cellák oldalhosszának megválasztása fontos kérdés, amit a fejlesztés során el kell dönteni. Túl nagy cellaméret esetén a szemantikai régiók elkülönítése nehézkes, túl kis cellaméret esetén a számítási igény nő meg jelentősen, ami ellehetetleníti a valós idejű működést. A cellák oldalhosszának választása során több értéket megvizsgáltam. Azt tapasztaltam, hogy 0,5 m-es cellaméret esetén az objektumok gyakran nem különíthetők el, míg 0,2 m-es cellaméret esetén az algoritmus már nem futott valós időben. Így tehát az ezek közötti értékekre leszűkítettem a kört. Végül a választásom 0,3 m-es oldalhosszra esett, mert ennél az értéknél jelentősen csökkent a sikertelen elkülönítések száma a nagyobb értékekhez képest. Ezzel az értékkel hatékony az objektumok elkülönítése, és az algoritmus képes valós idejű működésre.

A struktúra kialakítása során az első lépés a bemenő felhőben az x és y értékek maximum és minimum értékének megkeresése. A cellaméret megválasztása után ezek ismeretében számítható, hogy mekkora cellarácsra lesz szükségünk. Ezt követően a pontfelhő minden egyes pontját hozzárendelem a neki megfelelő cellához az x és y koordinátája alapján (hiszen a z = 0 síkon vizsgáljuk a pontok elhelyezkedését). A pontok cellákhoz rendelése során rendre frissítem az aktuális cellához tartozó maximális illetve minimális x, y, és z értékeket. Ezzel a lépéssel elkészült a szabályos rács struktúra, melynek a cellái együttesen tartalmazzák a teljes pontfelhőt.

4.2.2 Talaj cellák meghatározása

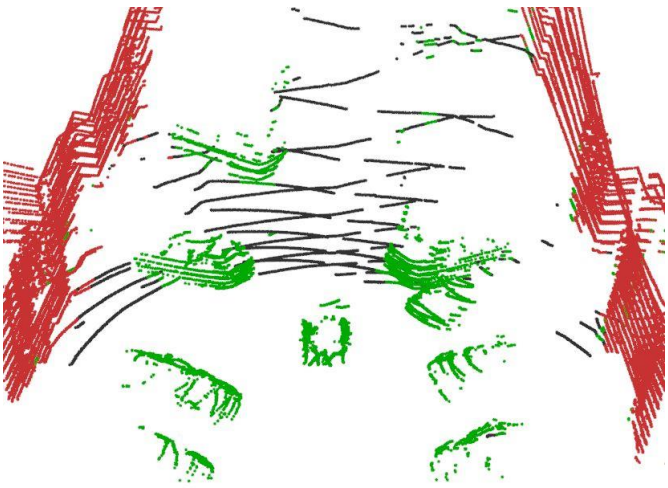
A következő lépésben a rács celláit bejárom, és azokat besorolom osztályokba különböző lokális tulajdonságaik alapján. Amennyiben egy cella fölött nem található pont, akkor a ritkás cella osztályba sorolom be, ezekkel a cellákkal további teendőm nincs. Talaj osztályú akkor lehet egy cella, ha a felette található pontfelhő „lejtése" nem túl nagy, azaz ha a legmagasabban és legalacsonyabban levő pontja közötti távolság kisebb 5,3 cm-nél ($0{,}3\ m \cdot \tan 10°$). Talaj osztályú lesz egy cella, ha emellett a maximális magasság a cella fölött kisebb a cellára becsült maximális talajszintnél. Ezt a maximális talajszintet a szenzortól valós távolság függvényében adom meg. A szenzor közvetlen közelében a talajszint ismert. A szenzortól távolodva a maximális talajszintet egy maximális lejtési szöget feltételezve a szenzortól való távolságból kiszámíthatjuk:



34. ábra: lokális maximális talajszint számítása

$$h_{max} = h_0 + d \cdot \tan \alpha_{max}$$

α_{max} értékét 10°-nak választottam. Városi környezetben ennél nagyobb lejtésű út nem található jellemzően. Azokat a cellákat, melyek mindkét feltételt kielégítik, besoroltam a talaj osztályba. Ezek a cellák együttesen alkotják a talajt, ezzel tehát elkészült a talajdetektáló algoritmus. Ennek az eredményét a 35. ábra mutatja.



35. ábra: Talaj felismerés eredménye (feketével jelölve a talajt)

4.3 Objektumok elkülönítése

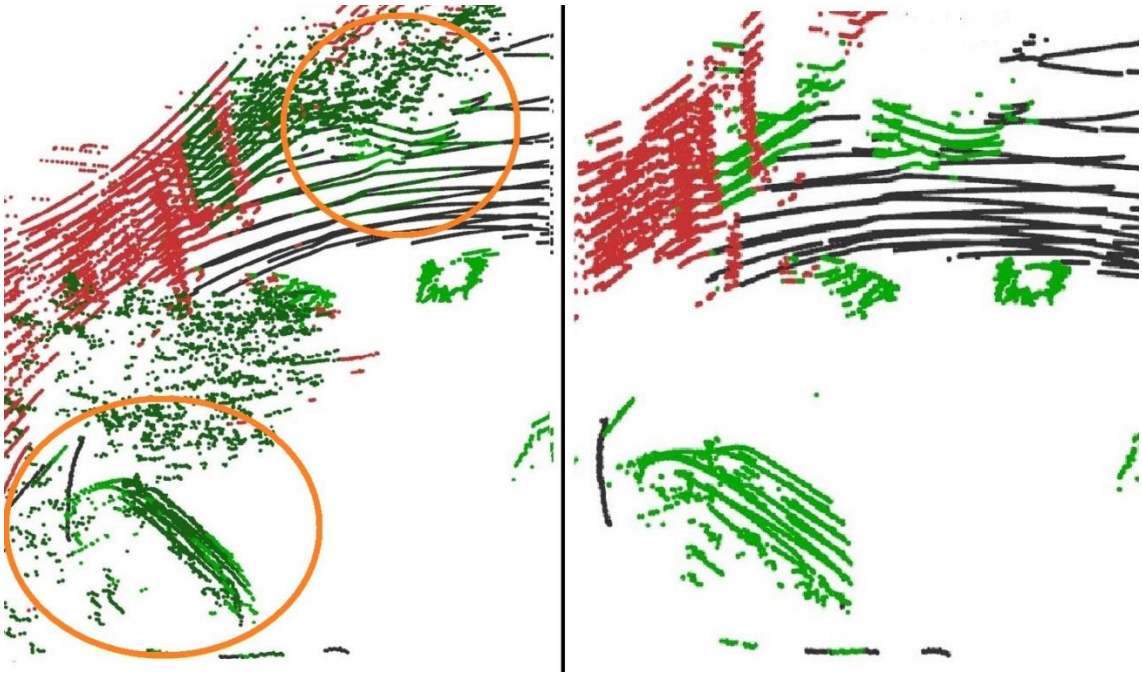
4.3.1 Alacsony és magas objektumok meghatározása

A szegmentáció még nem ért véget a ritkás- és a talaj cellák megtalálásával. A fennmaradó cellákat besorolom két további kategóriába a cella fölötti maximális magasságértékek alapján. Azon cellákat, melyek fölött a maximális magasság kisebb 2 m-nél, az objektum osztályba; melyeknél nagyobb, a fal osztályába sorolom be.

A valószínűsített lombkorona pontokat még nem távolítottam el, így a magas objektumok osztályába bekerültek azok a cellák is, melyek egy-egy objektum darabot, és a felette található lombkorona pontokat tartalmazzák. Az észlelt magas objektumokat ezért még egyszer bejárom, hogy ezeket a cellákat felülvizsgáljam. A 2 m feletti pontokból kiolvasom a lombkorona-valószínűség értékeket. Azonban nem csak a valószínűség-értékek alapján döntök, hanem vizsgálom a cella fölött 0 m és 2 m között található pontokat is. Ha egy cella a valószínűség alapján nem lombkoronát tartalmaz, de 0,5 m és 1,8 m között nem találok pontot, akkor az a cella mégis lombkoronát tartalmaz. Ha egy celláról azt gondolom, hogy lombkoronát tartalmaz, de alatta több 0,2 m szélességű sávban is találok pontot, akkor az a cella mégsem lesz lombkorona, ekkor valószínűsíthetően faldarabot vagy táblát tartalmaz.

Az lombkorona cellák fölött a magasan helyezkedő pontokat elhagyom, és frissítem a cellákra a maximális magasság értéket. A módosított maximum érték alapján a cella osztályát talajnak vagy objektumnak adom meg.

A 36. ábra a lombkorona eltűntetést mutatja be. Bal oldalt az ábrán láthatók sötétzölddel azok a részek, melyeket az eredeti algoritmus falként észlelt volna, mert a magasságuk 2 m feletti. Látható, hogy a fák alatt parkoló autók jelentős része sötétzöld, ez azt jelenti, hogy ezek az autók csak darabjaikban lesznek észlelhetők. Ezek a darabok lehetnek nagyon kicsik is (mint az ábrán alul látható autó esetében), ami nem biztosítja az objektumok kielégítő észlelését. A lombkorona pontok eltávolítása után azonban a parkoló autók is felismerhetők objektumként, mint azt az ábra jobb oldalán megfigyelhetjük. Látható még, hogy a talaj meghatározása is pontosabb lett azáltal, hogy azokat a cellákat is talajnak jelölöm, melyek fölött korábban lombkorona pontok voltak.



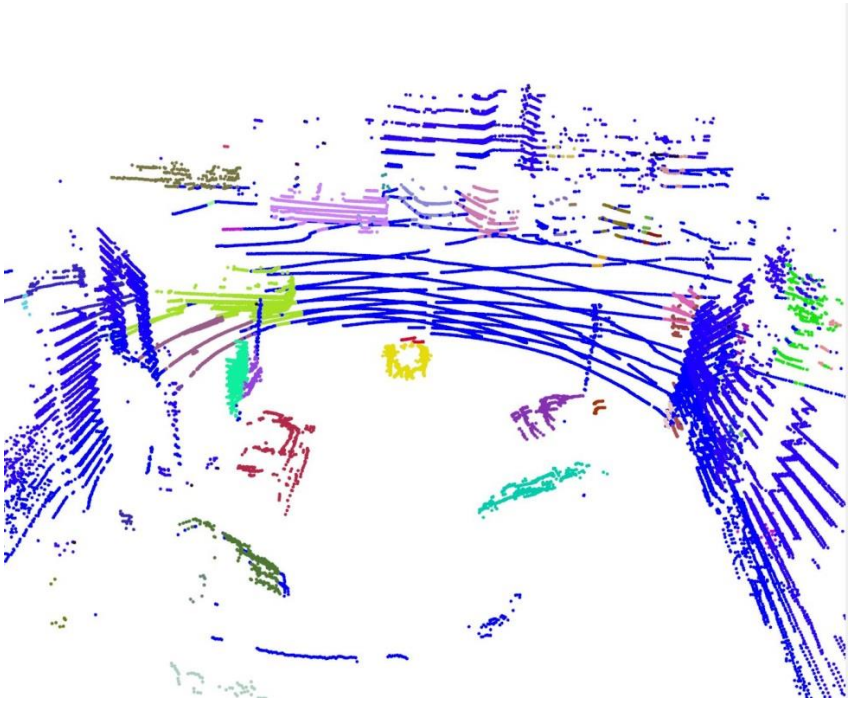
36. ábra: Lombkorona észlelés, és eltávolítás után észlelt objektumok

Megfigyelhetjük, hogy az algoritmus téveszt is, néha faldarabokat is lombkoronának vesz, majd eltávolítja a magasan található pontjait. Ilyen módon tévesen detektált objektumok keletkeznek, ez a jövőben megoldandó probléma. A megoldás lehet az, hogy valamilyen módon megbecsüljük a falak távolságát, és az ahhoz túl közeli objektumokat felüldefiniáljuk falként.

4.3.2 Objektumok összeillesztése

Az objektumokat az objektum osztály celláiból fogom előállítani oly módon, hogy a szomszédos cellákat egy kritérium alapján összevonom vagy elkülönítem. Amennyiben a szomszédos cellák maximális magassága közti különbség elég kicsi (kevesebb, mint 0,6 m), akkor a két cella ugyanazon objektumhoz tartozik. Ebben az esetben összevonom a két cellát, különben elkülönítem. Az összevonás-elkülönítés műveletét úgy valósítom meg, hogy ID-ket rendelek a cellákhoz. Két cellát összekapcsolok azáltal, hogy ugyanazt az ID-t adom nekik; elkülönítem őket, amennyiben különböző ID-t adok nekik.

A teljes cellarendszert bejárom, és vizsgálom a cellák szomszédjait. Amennyiben két szomszédos cella kielégíti az összevonhatóság kritériumát, akkor azoknak az ID-jeit egyenlővé teszem. Ezen vizsgálat során a már vizsgált cellákat megjelölöm, hogy azokkal a későbbiekben ne foglalkozzak. Mikor végeztem a bejárással, az azonos ID-vel rendelkező cellák pontfelhőit összevonom, így kapom meg az objektumokat. Ezeket egy objektum vektorba mentem bizonyos jellemzőikkel együtt. Ezeket a jellemzőket későbbi pillanatképek vizsgálata során is fel lehet használni (pl.: mozgásészlelés, objektumkövetés). Az objektum elkülönítő algoritmus kimenete az alábbi ábrán látható.



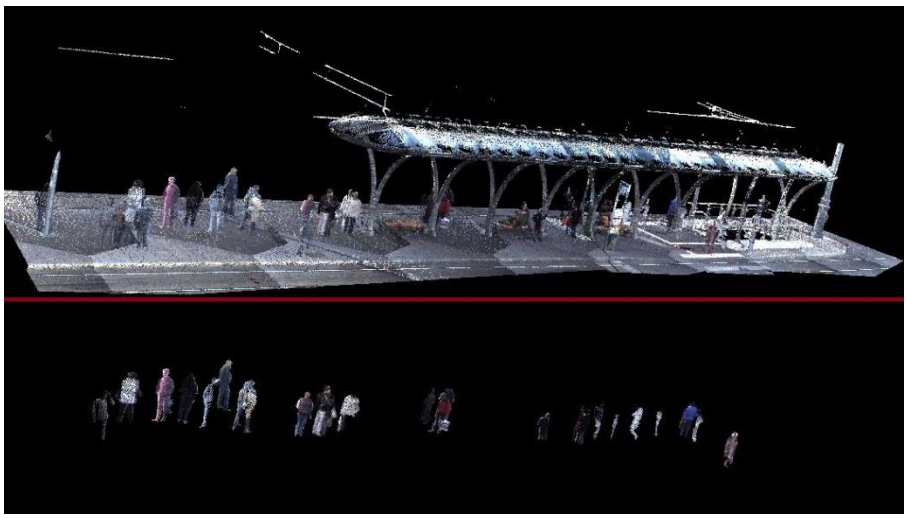
37. ábra: Objektum észlelés

5 Objektumok felismerési lehetőségei

A környezet megértése szempontjából az objektumok észlelésén túl fontos feladat az egyes alakzatok felismerése, amelyre különböző szakirodalmi megoldások léteznek. Munkám során kifejlesztettem egy főként méret és alak alapján történő alakzatosztályozó eljárást, amelyet sikeresen alkalmaztam emberek felismerésére térképező rendszerek nagysűrűségű pontfelhőiben. Az eredményeket az 5.1 alfejezetben ismertetem. Az eljárást teszteltem a dolgozatban részletezett ritkább, valós időben gyűjthető mérési adatokra is, azonban ez a megoldás ott kevésbé bizonyult hatékonynak. Az 5.2 fejezetben ezért megemlítek olyan – részben a kutatócsoportunkban készült – mély tanulás alapú eljárásokat is, amelyek a munka későbbi továbbfejlesztése során felhasználhatók lehetnek.

5.1 Méret, alak alapján

Méret és alak alapján ismert geometriai méretekkel rendelkező objektumokat lehetséges felismerni: embereket, autókat, táblákat. Az önálló laboratórium során emberek felismerése volt a feladatom. Az emberek felismerésének az a célja, hogy őket eltüntetve a városi helyszínekről statikus modellt tudjunk kialakítani, melyből kiszűrjük a dinamikusan változó szereplőket. Itt az objektumok geometriai tulajdonságait használtam ki. A felismerés alapja az volt, hogy az emberek magassága 0,5 m és 2,5 m közötti, szélességük körülbelül 30 cm és 60 cm közötti.



38. ábra: Önálló laboratórium feladatom eredménye. Felső kép: bemenet; alsó kép: kimenet

A felismerés eredményét mutatja be a 38. ábra. Látható, hogy a jól elkülöníthető (egymástól kellően távol álló) embereket megtalálja az algoritmus. Azonban azokat, melyek közvetlen egy oszlop mellett állnak, vagy padon ülnek, azokat nem. Ennek az az oka, hogy nem tudjuk őket elkülöníteni a szomszédos objektumoktól, így a geometriai paraméterei megváltoznak az emberek szegmenseinek. Ugyanez a probléma fennállhat például kézen fogva sétáló embereknél is. Erre a problémára nem jó megoldás az, ha kisebb küszöbértéket adunk meg pontok távolságára az objektumok összekapcsolásakor, mert ekkor az objektumok darabjaikra eshetnek szét.

Autók felismerésére is hasonlóképp van lehetőség, azok magassága 1 m és 2 m közötti, szélességük jelentősen nagyobb magasságuknál. Itt az okozhat problémát, hogy különböző típusok különböző paraméterekkel bírhatnak. Teherautó magassága például 2 m-nél jóval nagyobb is lehet. Előfordulnak olyan típusok is (pl. Mercedes Smart), melynek szélessége lényegében megegyezik a magasságával. Az ilyen extrém példák esetén a geometria-alapú felismerés csak a paraméterezés könnyítéseivel működhet, ami egyúttal tévesen felismert objektumokhoz is vezethet.

Valós idejű Lidarok pontfelhői esetén további problémát jelent, hogy különböző nézőpontból az objektumok akár teljesen máshogy is nézhetnek ki. Ez főleg nagyobb objektumokra jellemző. Különböző nézőpontból ugyanazon objektumok geometriai paraméterei jelentősen eltérhetnek. Emiatt a méreten, alakon alapuló felismerési algoritmusok nem lesznek kellően hatékonyak.

Ezeknek a problémának a kiküszöbölésére alkalmas lehetne egy mozgást felismerő algoritmus. Ha el tudjuk különíteni a mozgó objektumokat, akkor ezek közül már viszonylag könnyű lenne embereket és autókat felismerni, mert a statikus utcai objektumok megszűnnének mint hibalehetőség. Valós idejű Lidarok esetén tehát méret és alak alapján történő felismerés főképp emberek és autók detektálására lehet alkalmas.

Térképezésre használt nem valós idejű Lidarok esetén sokkal inkább működhet a geometriai tulajdonságokon alapuló felismerés, hiszen itt az objektumok jobban hasonlítanak egymásra. Azonban itt is fennáll az a probléma, hogy különböző típusú autóknak jelentősen különböző lehet a méretük, alakjuk. Egyéb objektumok felismerésére az ember- illetve autó felismeréshez hasonló módszereket lehet alkalmazni. Érdemes lehet az objektumokat nem egyben vizsgálni, hanem apróbb részletekből kinyerni számunkra hasznos információt, mint *[6]*-ban.

5.2 Konvolúciós neurális hálózatokkal

Konvolúciós neurális hálózatok alkalmazása valós idejű Lidaroknál jelentős előnyöket hordoz. Különböző szemszögből az objektumok különbözőképp néznek ki, eltérő geometriai tulajdonságokkal rendelkeznek. Emiatt objektumok felismerésére a geometria-alapú módszerek nem alkalmasak. Azonban a konvolúciós neurális hálózatok által alkotott tulajdonságtérben objektumok felismerése lehetséges. A kutatócsoportunk által írt cikk *[4]* bemutat egy tanuló hálózat alapú objektum felismerést valós idejű Lidar alkalmazása esetén. Ez a megoldás tesztjeink alapján meglehetősen jól működött kis sűrűségű valós idejű Lidarok pontfelhőire. A riportban bemutatott módszert jövőbeni lehetséges továbblépésként érdemes bővíteni egy konvolúciós neurális hálózat alapú objektum felismerési lépéssel. Ehhez szükséges létrehozni nagyméretű tanító adatbázist az algoritmus által elkülönített objektumok felcímkézésével.

A konvolúciós neurális hálózatok alkalmazása térképezési eljárásokhoz használt Lidarok pontfelhői esetén is hordoz előnyöket. Az objektumok címkézése során például a különböző típusú autók, különböző testtartású emberek (álló, ülő, lehajoló) is megjelölésre kerülhetnek, így azok is hatékonyabban felismerhetők, mint a tisztán geometria-alapú felismerés esetén. A szakirodalomban is olvashatunk nagysűrűségű pontfelhők esetén gépi tanuláson alapuló objektum-felismerési eljárásokról *[6]*, *[8]*.

6 Mérések elvégzése, kiértékelése

6.1 Mérés előkészítése

A mérések során autóra szerelt Lidartól Etherneten keresztül kapott adatokat mentjük egy ipari PC-re. A mérések előtt az első feladat a rendszer összeszerelése. Szükségünk van tápegységre, mellyel az ipari PC és a Lidarok menet közbeni energiaellátását biztosítjuk. A Lidarokat Etherneten keresztül csatlakoztatjuk az ipari PC-hez, melyen fut egy alkalmazás, ami a kapott UDP csomagokat feldolgozza, az adatokat menti. A hátsó ülésről nem hiányozhat a monitor és az egér sem, melyek az ember-gép interfészt biztosítják.

A mérés összeállítása után elindítjuk az adatok vételét a két Lidartól. A két kapott pontfelhő valós időben megjeleníthető. A felhők összeillesztéséhez szükség van a két Lidar közti transzformációs mátrixra, amit *[9]*-ben leírt módszer alapján határozunk meg. A kalibráció során dobozokat helyezünk el többféle elrendezésben úgy, hogy mindkét Lidar pontfelhőjében mindhárom oldaluk látszódjon. Amint sikeresen elhelyeztük a dobozokat, elmentjük a két-két pontfelhőt az összes elrendezésre, ezek alapján a transzformáció a Lidarok között meghatározható.

Mi a mérés során négy kamerát is az autóra szereltünk. Ennek a célja egyrészt az volt, hogy a kiértékelés során ez alapján ellenőrizni tudjuk, hogy a pontfelhőben található objektumok mit is ábrázolnak (az ember látása számára szokatlan a pontfelhők kinézete, így néhány objektum nehezen felismerhető). Másrészt a kamerák egy másik projekt során is szerepet kapnak, mely során a pontfelhőben detektált objektumokat visszavetítjük a kamerák képére, illetve a kamerák képe alapján színezzük a pontfelhőt.

6.2 Mérés menete

A kalibrációhoz szükséges adatok megléte után indíthatjuk el valójában a mérést. Ehhez egy korábban megírt szoftvert használunk. Többféle üzemmódban futtathatjuk a rendszert. Az első üzemmódban minden rendszer küldi az adatokat egymástól függetlenül, a második üzemmódban csak a Lidarok vannak időszinkronban, míg a harmadik üzemmódban a kamerák és a Lidarok is szinkronban működnek. Nekünk olyan üzemmódra volt szükségünk, mely során a Lidarok szinkronban működnek, ugyanis csak

ekkor tudjuk pontosan összeilleszteni a pontfelhőket. Ezt a követelmény két üzemmód is kielégíti. Annak az üzemmódnak, mely mind a kamerákat, mind a Lidarokat szinkronizálja, az a hátránya, hogy lassabban érkeznek a pillanatképek, azaz csak nagyobb mintavételi idő érhető el. A jelenlegi feladatunkhoz az az üzemmód optimális, mely csak a Lidarokat szinkronizálja, mert ekkor több pillanatképet készíthetünk ugyanannyi idő alatt. Az Etherneten keresztül érkező UDP csomagokat a szoftver kicsomagolja, és saját (MTA SZTAKI-s) fájlformátumban (.stream) menti el. A mérés eredménye két fájl lesz a két szenzortól, a későbbiek során ezt kell feldolgozni.

A fájlba a mérés során gömbi koordináták kerülnek elmentésre. A szenzor forgása során ismert az x-z síkkal bezárt szög, a csatorna pozíciójából pedig ismert az x-y síkkal bezárt szög is. A gömbi koordinátákhoz szükséges távolságot pedig a szenzor méri. Ezekből kell kiszámolni a fájlok beolvasáskor a Descartes-koordinátákat.

6.3 Mérés kiértékelése

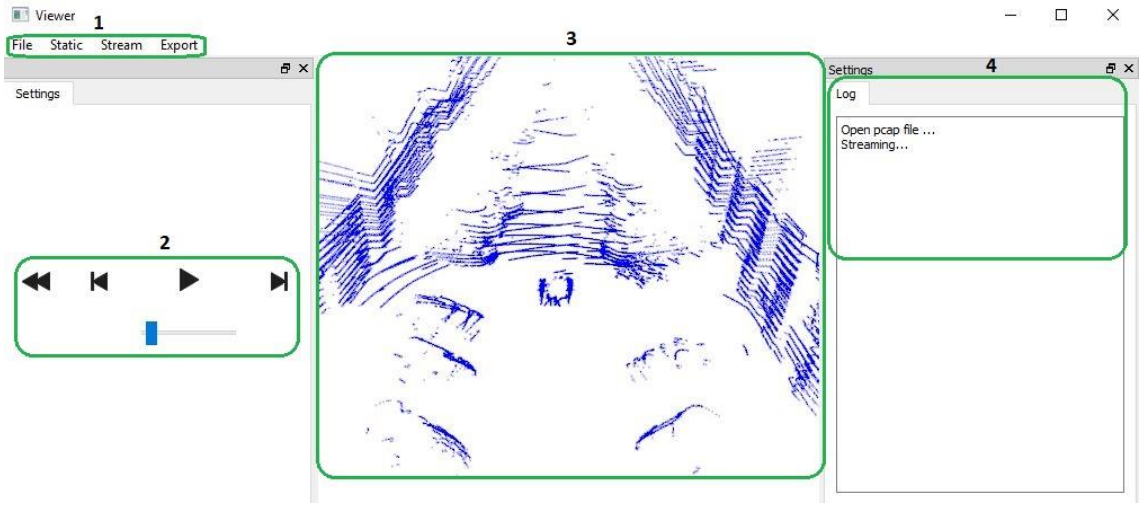
A fájl beolvasását a keretrendszeren belül kell megvalósítani. Ennek során megnyitjuk a két stream fájlt, a tartalmukat beolvassuk a memóriába. Ezután végigjárjuk a két fájlt párhozamosan, és a mentett csomagokból kiszámoljuk a számunkra szükséges x, y, z koordinátákat. A kalibráció eredményéből képzünk egy transzformációs mátrixot, mellyel az egyik pontfelhő pontjainak koordinátáit megszorozva azokat transzformáljuk a másik pontfelhő koordináta-rendszerébe. Ezután a képzett pontfelhő megjeleníthető, a fejlesztett algoritmusok rá lefuttathatók.

39. ábra: A mérési elrendezés

7 Szoftverrendszer dokumentációja

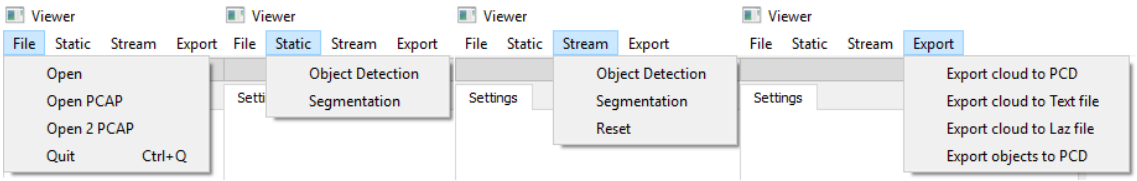
7.1 Felhasználói szintű dokumentáció

A felhasználói felületet QT szoftverfejlesztői csomag felhasználásával készítettem el. Kiindulási pontnak rendelkezésemre állt egy már kész felület, ezen kellett módosítanom a menüpontokat, illetve bekötni azokhoz a megírt függvényeket. A menüpontok kialakításakor a legfontosabb szempont az volt, hogy a felhasználó számára minden művelet elvégzéséhez szükséges lépés egyértelmű legyen.



40. ábra: Felhasználói felület

Az alkalmazás felületét négy főbb részre bonthatjuk. A menüpontokra, ahol megnyithatunk fájlokat, futtathatunk rajtuk algoritmusokat (1); a lejátszást vezérlő gombokra, ahol lejátszhatjuk a videót, léptethetjük előre és hátra, vagy akár tekerhetünk is benne a képen látható csúszkával (2); a megjelenítési felületre, ahol a megnyitott fájlok megjelennek (3), illetve a log ablakra, ahol a program aktuális állapotáról kaphatunk információkat (4).



41. ábra: Menüpontok

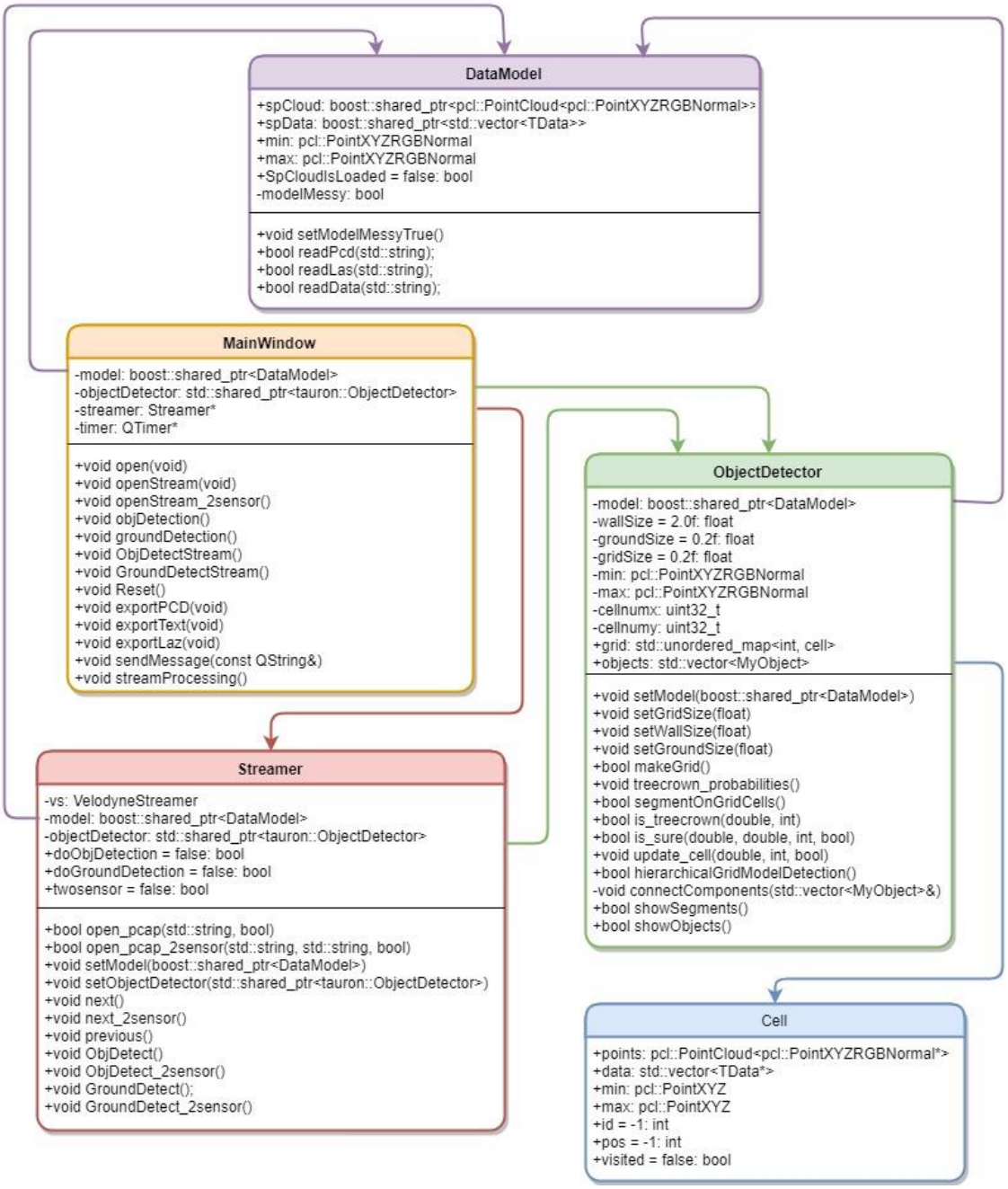
A File menüben van lehetőségünk megnyitni statikus pontfelhő fájlt, illetve pcap fájlt, ami a pontfelhő-videónak felel meg. Az Open PCAP funkció egyetlen PCAP fájlt, míg az Open 2 PCAP funkció két PCAP fájlt nyit meg, majd az előre beprogramozott transzformáció szerint illeszti azokat. A programon lehetne fejleszteni oly módon, hogy megnyitáskor meg lehessen adni a transzformációs mátrixot, akkor több különböző mérés eredményét is képes lenne vizsgálni egy futás alatt a program.

A Static menüben statikus pontfelhő képekre, míg a Stream menüben pontfelhő videóra futtathatjuk le a fejlesztett algoritmusokat. A Reset funkcióval a videó lejátszása esetén kikapcsolhatjuk a szegmentációt és az objektum észlelést, amennyiben a nyers pontfelhő sorozatot szeretnénk nézni.

Az Export menü lehetőséget kínál egy a programban épp aktív (azaz épp megjelenített) pontfelhő merevlemezre mentésére. Ez teszteléskor lehet hasznos, ha például a helyesen detektált és a helytelenül detektált objektumokat számláljuk, később ellenőrizni lehet ennek a helyességét az elmentett pontfelhő alapján. Sokféle formátumba lehet elmenteni a pontfelhőket, ezek közül kedvünkre választhatunk. Hasznos fejlesztés lehet egy olyan funkció bevezetése, mellyel csak az objektumokat lehetséges egyesével menteni, ennek a segítségével lehetőség nyílna azok vizsgálatára. Ez a funkció segítené az objektum felismerés fejlesztését, mert így elkülönítve lehetne vizsgálni egy-egy objektum tulajdonságait, így pedig jobb felismerő algoritmusokat fejleszthetnénk. Tanuló hálózatokhoz is használhatók lehetnének az exportált objektumok, azokat fel lehetne címkézni, mint a hálózat bemenetei.

7.2 Programozói szintű dokumentáció

A feladat megoldásához felhasználtam a Qt 5.8-as és a PCL 1.8-as verziójának osztályait, beépített függvényeit. Az alábbiakban a program szerkezetét fogom ismertetni. Az 5 legfontosabb osztály kapcsolatait és működését szeretném bemutatni az alábbi ábra segítségével. Az ábrán nem szerepelnek a pontfelhők vizuális megjelenítésére szolgáló, illetve az algoritmus szempontjából kevésbé fontos függvények, osztályelemek; csak azok, melyek a megvalósított algoritmus szempontjából lényegesek.



42. ábra: A program szerkezete

A DataModel adatstruktúra nagy jelentőséggel bír, ennek az spCloud változójába töltjük be a mindenkori vizsgált pontfelhőt, legyen az akár statikus, akár dinamikus. Az spData változó a Lidartól beolvasás során kapott lényeges információkat tartalmazza: a visszavert lézersugár intenzitását, a sugár kilövésének szögét ($0 - 360°$), illetve a mért távolságot. A min és max pont típusú változó x, y, z koordinátáiba az spCloud-ban tárolt pontfelhő x, y és z szerinti szélsőértékei kerülnek eltárolásra. Az spCloudIsLoaded változó azt jelzi, ha a felhő már beolvasásra került, meg lehet jeleníteni. Ehhez az

osztályhoz tartozik három beolvasó függvény, melyekkel különböző formátumú statikus pontfelhőket olvashatunk be. A setModelMessyTrue nevű függvény a modelMessy változót állítja, amely azt jelzi, hogy az spCloud tartalma megváltozott, újra ki kell rajzolni.

A MainWindow osztály tartalmazza a kezelői felületet, innen kerülnek meghívásra a beolvasó és videó lejátszó függvények. Itt lehet be- illetve kikapcsolni a szegmentáló és az objektum észlelő algoritmusokat. Az osztályhoz természetesen tartoznak megjelenítő elemek is, azonban ezeket a fenti ábrán nem jelenítettem meg. Az open függvénnyel statikus pontfelhőt olvashatunk be a model-be, az openStream illetve az openStream_2sensor függvényekkel pedig pontfelhő-szekvenciákat nyithatunk meg. Az exportáló függvények a model éppen aktuális tartalmát exportálják a kívánt formátumban. A sendMessage függvény a log ablakba ír információkat a program futásának állapotáról. Az ObjectDetector típusú változó a statikus pontfelhőkre futtatja le a szegmentáló és az objektum észlelő algoritmusokat. Az osztálynak van még egy Streamer típusú változója is, mely a pontfelhő szekvenciák lejátszására szolgál. A lejátszás időzítését szolgálja a timer nevű változó, mely 100 ms-onként meghívja a streamProcessing függvényt, ami a Streamer bool típusú változóinak értéke alapján eldönti, hogy a streamer mely függvényét hívja meg a következő pillanatkép megjelenítésére.

A Streamer osztály a pontfelhő szekvenciák lejátszására szolgál, továbbá a szekvenciákon algoritmusok futtatására. Fontos eleme a VelodyneStreamer típusú objektum. Ennek az osztálynak a működése az algoritmus szempontjából nem lényeges, ezért nem szerepel az ábrán. Feladata a teljes pontfelhő szekvencia memóriába történő beolvasása, és a már betöltött szekvenciának a bejárása, model-be való betöltése. A Streamer osztályhoz is tartozik egy model nevű változó, ami meg fog egyezni a MainWindow model nevű változójával. A Streamer-hez is tartozik egy ObjectDetector típusú változó, ami a model-be betöltött pontfelhőkön futtat algoritmusokat. A három bool típusú változó értéke dönti el, hogy a Streamer függvényei közül melyiket fogjuk meghívni a MainWindow időzített streamProcessing függvényéből.

A Cell osztály minden példánya tartalmaz egy pointertömböt, melynek elemei a model spCloudjának cella fölött található pontjaira mutatnak. Hasonlóképpen egy másik pointertömb tartalmazza az ezekhez a pontokhoz tartalmazó TData típusú információkat.

Minden cellához tartozik egy min és egy max nevű változó, ezek a cellák fölött található pontok szélsőértékeit tárolják. Az id, pos, visited változók a cellákból az objektumok összeillesztéséhez szükségesek.

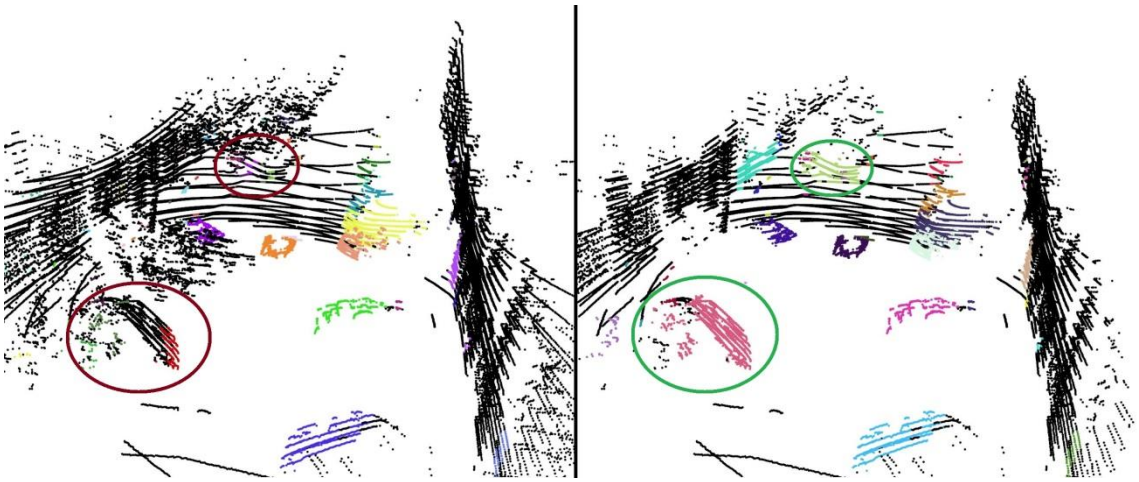
Az ObjectDetector osztály tartalmazza a megvalósított algoritmusokat, melyek alapja egy szabályos rács struktúra (grid nevezetű), amely az imént említett Cell osztály példányaiból áll. A szabályos rács létrehozása a makeGrid() függvénnyel történik. A rács celláinak a méretét a gridSize nevű változó határozza meg. A min és max nevű változók, valamint a gridSize ismerete esetén határozhatjuk meg a rács méretét, ekkor tudunk memóriát foglalni neki, és ezután tudjuk feltölteni a celláit. A szegmentációs lépés előtt futtatjuk a treecrown_probabilities nevű függvényt, mely kiszámolja a magas pontokhoz annak a valószínűségét, hogy az a pont lombkorona pont. A szegmentációs lépést a segmentOnGridCells nevű függvénnyel hajthatjuk végre, melynek során használatos paramétereket a groundSize, és a wallSize változók tartalmazzák. Az imént említett változókat (gridSize, wallSize, groundSize) a megfelelő set függvényekkel állíthatjuk. A segmentOnGridCells függvény meghívja a lombkorona pontok észlelése során az is_treecrown és az is_true függvényeket, melyek segítségével vizsgálja az adott cella fölötti 0 m és 2 m közötti magasságban elhelyezkedő pontokat. Ezeknek a függvényeknek a segítségével jobb döntést tud hozni, mint szimplán a valószínűségekkel. A detektált lombkorona cellákat az update_cell nevű függvény módosítja, hogy a további műveletekhez alkalmasak legyenek. Amennyiben objektum észlelést szeretnénk futtatni, akkor a hierarchicalGridModelDetection nevű függvényt kell meghívnunk. Ez az előző szegmentációs lépés után meghívja a connectComponents függvényt, ami összekapcsolja az azonos objektumhoz tartozó cellákat. A showObjects és showSegments vizuálisan is megjeleníti a szegmentáció és az objektum észlelés eredményét. Az ObjectDetector osztály példányosítva van a MainWindow-n és a Streameren belül is, előbbi esetben statikus, utóbbi esetben dinamikus pontfelhő elemzésére alkalmas.

8 Felhasználhatóság, továbblépési lehetőségek

8.1 Eredmények értékelése

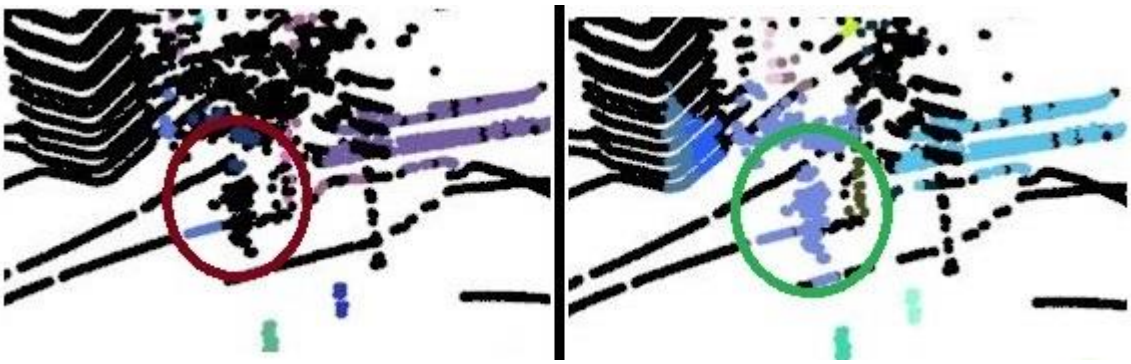
A munka során fő célom objektumok észlelése volt 3D pontfelhőkben. A korábbi kutatások és eredmények alapján megvalósítottam egy működő algoritmust, ami hatékonyan működik a saját mérésünk során mentett szekvenciákra. Mivel eredetileg problémát jelentettek a magas fák alatt található objektumok, ennek a kiküszöbölésére fejlesztettem egy lombkorona észlelő és eltávolító algoritmust. Az algoritmus a fák lombkoronáját szinte tökéletesen eltünteti, ilyen szempontból jól működik. Hátránya viszont, hogy néha az egyenetlen falakat is lombkoronának veszi, így a falakból is veszítünk részeket, bár ez nem jelentős.

Az alábbiakban néhány képen szemléltetem az objektumészlelés működését, illetve a lombkorona pontok eltüntetésének hatékonyságát, hasznosságát.



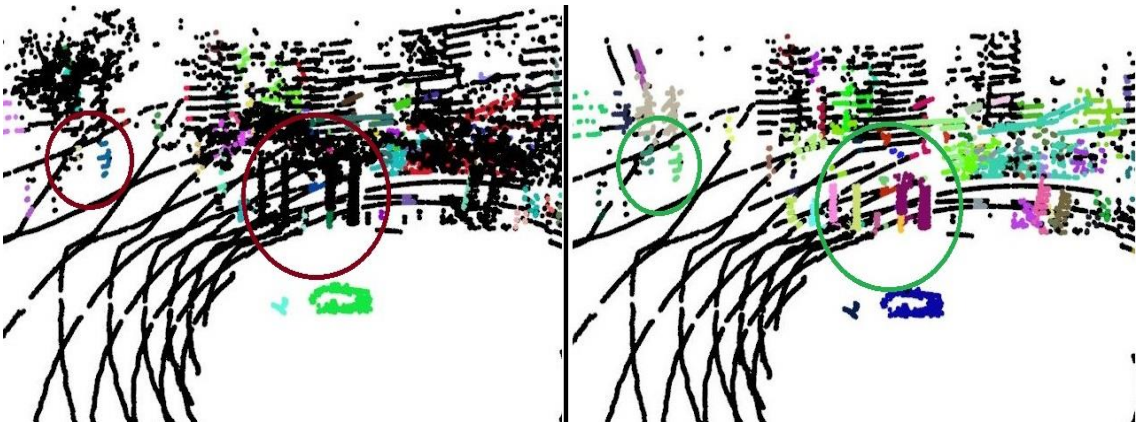
43. ábra: Eredmények szemléltetése

A 43. ábra bemutatja, hogy a lombkorona eltüntetése után a fa alatt parkoló autók is észlelhetők lesznek. A lombkorona észleléssel bővített algoritmus a fa alatt található objektumokat is hiánytalanul megtalálja, tehát több objektumot képes észlelni. Azonban a jobb oldali képen látható, hogy a fal egy darabját is objektumnak tekintete az algoritmus, ezt a hibát majd ki kell küszöbölni (fejlesztési lehetőségeknél kifejtem).



44. ábra: Eredmények szemléltetése

A 44. ábra bemutat egy olyan esetet, mikor egy ember sétál egy fa alatt. Látható, hogy a lombkorona eltávolítása után az algoritmus felismeri az embert, míg nélküle nem. Az ember közelében található alacsony lombkorona pontokat az objektumokat összekapcsoló algoritmus hozzákapcsolta az emberhez. Ez a hiba abból ered, hogy az autó egy lejtőn állt a helyszínen, emiatt a magasságértékek valamennyire elcsúsztak a valóshoz képest, így a lombkorona pontok 2 m alá kerültek.



45. ábra: Eredmények szemléltetése

A 45. ábra mutatja, hogy a járdán sétáló gyalogosok felismerését hogy segíti a lombkorona eltüntetése (a bal oldali képen a bal oldali embert nem ismerte fel az algoritmus, de a jobb oldalon már igen). Ezen felül az út szélén található oszlopok és fatörzsek észlelése is csak a lombkoronák eltüntetésével lehetséges. Ez fontos lehet az autó mozgásterének felismeréséhez.

8.2 Felhasználhatóság

A megvalósított algoritmus felhasználható önjáró autók környezetelemzési eljárásaihoz, ehhez érdemes a módszert objektum felismerési lépéssel bővíteni, ahogy az *[4]*-ben is megtörténik. Amennyiben rendelkezésre áll nagy felbontású pontfelhő az autó tartózkodási helyéről (magáról a városról), akkor lokalizációs feladatokhoz is felhasználhatóak az észlelt objektumok. Ezen kívül a valós idejű pontfelhőben észlelt objektumok alapján képesek lehetünk változásokat észlelni a nagyfelbontású pontfelhőben (például új felállított bódék, új épületek, stb.). Ezen két alkalmazáshoz (melyekhez nagyfelbontású háttérfelhő szükséges) azonban kérdéses, hogy az illesztett felhő pontsűrűsége elegendően nagy-e. Az objektumészlelés felhasználható még biztonságtechnikában, ahol nem kívánt betolakodókat szeretnénk észlelni bizonyos területeken. Az objektumok és tulajdonságok frame-enkénti mentése során lehetőség nyílhat objektumok követésére is, amiből képesek lehetünk akár emberek mozgását kielemezni. Azonban itt is kérdéses, hogy a pontfelhő vajon elég sűrű-e ehhez.

8.3 Továbblépési lehetőségek

8.3.1 A meglévő algoritmus fejlesztési lehetőségei

Az első lépés a lombkorona felismerése és eltávolítása volt. Ennek a hatékonyságát növelni lehetne, ha a Lidart másik üzemmódban futtatnánk. Az általunk használt üzemmód a Single Return Mode volt. Ez mindig a lézersugár első visszaverődésének a távolságát adja meg. Ezen kívül van még Dual Return Mode, mely az első és a második visszaverődés távolságát is megadja, amennyiben van második visszaverődés. A harmadik üzemmód pedig a Strongest Return Mode, mely a legnagyobb intenzitású visszaverődés távolságát menti el. A fák lombkoronájára jellemző, hogy több visszaverődés is érzékelhető az irányukban. Így a lombkorona észlelés javítható lehet a Dual Return Mode használatával.

A második lépés a szemantikai szegmentáció volt, ezen fejleszteni újabb osztályok bevezetésével lehetne. Szükség lehet tábla osztályra, mert a táblák a fal osztályba kerülnek a magasságuk miatt. Emellett a fal osztályt fel lehetne bontani fal osztályra illetve magas objektum osztályra. Ehhez megvalósítható lehet egy fal felismerő algoritmus, mely az ettől távoli magas objektumokat a magas objektum osztályba

sorolhatná át a fal osztályból. A harmadik lépés az objektumok elkülönítése, amelyen 2D hierarchikus rács alkalmazásával lehetne javítani. Ekkor ugyanis a közeli objektumokat nagyobb hatékonysággal tudnánk elkülöníteni egymástól, ami javítaná a találati arányt. A 2D hierarchikus rács hátránya, hogy a futási időt növeli, ezért nem alkalmaztam a megoldásomban.

8.3.2 Továbblépési lehetőségek

Az algoritmus működik előzetesen mentett adatokra. Az első továbblépési lehetőség az, hogy a program képes legyen egyből az Etherneten érkező csomagok feldolgozására is. Ez esetben a mozgó autón valós időben is ki lehetne próbálni az algoritmust. Objektum követési funkció beépítése is hasznos lenne, mert ezzel például érzékelhetnénk, ha egy ember le fog lépni a jármű elé, vagy egy másik autó épp sávot vált. Érdemes lenne még az algoritmust bővíteni konvolúciós neurális hálózaton alapuló objektum felismerő funkcióval is a magasabb szintű környezet-értelmezés érdekében.

Irodalomjegyzék

[1] Quangong Feng (2012). *Practical application of 3D laser scanning techniques to underground projects.* BeFo Rapport.

[2] David Barber és Jon Mills (2007). *3D laser scanning for heritage: Advice and guidance to users on laser scanning in archaeology and architecture.* English Heritage.

[3] Börcs Attila, Nagy Balázs, és Benedek Csaba (2014). Fast 3-D Urban Object Detection on Streaming Point Clouds. Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving.

[4] Börcs Attila, Nagy Balázs, és Benedek Csaba (2017). Instant Object Detection in Lidar Point Clouds. *IEEE GEOSCIENCE AND REMOTE SENSING LETTERS*, 992-996. IEEE.

[5] Józsa Oszkár, Börcs Attila és Benedek Csaba (2013). Towards 4D Virtual City Reconstruction From Lidar Point Cloud Sequences., (old.: 15-20). VCM 2013 - The ISPRS Workshop on 3D Virtual City Modeling.

[6] Kiichiro Ishikawa, Fumiki Tonomura, Yoshiharu Amano, és Takumi Hashizume (2013). Recognition of road objects from 3D mobile mapping data. *International Journal of CAD/CAM*, *13*(2).

[7] Fabricce Monnier, Bruno Vallet, és Bahman Soheilian (2012). Trees detection from laser point clouds acquired in dense urban areas by a mobile mapping system. *Proceedings of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Annals), Melbourne, Australia, 25*, 245-250.

[8] Alexander Velizhev, Roman Shapovalov és Konrad Schindler (2012). Implicit shape models for object detection in 3D point clouds., *2.* International Society of Photogrammetry and Remote Sensing Congress

[9] Pusztai Zoltán, Hajder Levente (2017). Accurate Calibration of LiDAR-Camera Systems Using Ordinary Boxes. The IEEE International Conference on Computer Vision (ICCV).

[10] Hartley, R. a. (2004). *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518.