
Pázmány Péter Katolikus Egyetem

Információs Technológiai és Bionikai Kar

Nagy Balázs

# Dinamikus utcai környezet háromdimenziós analízise mobil lézerszkenner mérései alapján

Konzulensek:

Dr. Benedek Csaba, Börcs Attila

Budapest, 2014

# Tartalomjegyzék

# Kivonat

Dinamikus városi környezetek felügyelete központi feladat forgalomirányítási, környezetvédelmi, baleset- és bűnmegelőzési alkalmazásokban. A környezet információinak minél jobb kiaknázása érdekében a hagyományos képi szenzorok adatai mellett egyre nagyobb szerepet kapnak a háromdimenziós érzékelők. A mozgó autóra helyezhető Lidar szenzor olyan mérési sorozatot szolgáltat, ahol az egyes képkockák háromdimenziós pontfelhők, koordinátarendszerük pedig a jármű haladását követve változik. Így közvetlenül nyerhetünk pontos 4D (tér+idő) geometriai információt a helyszínről, azonban a ritka pontfelhő sorozatok elemzése számos kihívást tartogat az automatikus gépi felismerő algoritmusoknak. A környezet értelmezésének fontos eleme különböző városi objektumok automatikus elkülönítése a háttértől, majd az osztályzásuk és felismerésük a megfigyelt vizuális jellemzők alapján.

A dolgozatban áttekintést adok több szakirodalomban meglévő módszerről, melyek a Lidar szenzor által rögzített pontfelhők szegmentálását, tehát a különböző objektumokhoz tartozó pontfelhő részek elkülönítését végzik. Az egyes módszereket megvizsgálva megállapítottam, hogy kritikus feladat az egymáshoz közel álló alakzatok szétválasztása, ezért a munkám során egy hierarchikus, kétszintű rács alapú szegmentációs eljárást javasoltam az összetartozó ponthalmazok meghatározására. A saját módszer előnyeit bemutattam a hagyományos 3D elárasztásos szegmentációs megközelítésekkel szemben.

A következő lépésben módszereket javasoltam járművek, gyalogátkelőhelyek és közlekedési táblák elkülönítésére a Lidar pontfelhőkben tárolt geometriai és intenzitás információkat együttesen felhasználva. A gyalogátkelőhelyek észleléséhez kihasználtam, hogy az útfestéseknek az úttesttől eltérő anyagáról nagyobb intenzitással verődik vissza a lézerfény, és az átkelők jellegzetes mérettel valamint alaki paraméterekkel rendelkeznek. Az útjelző táblákat a keskeny oszlopok tetején elhelyezkedő nagy intenzitású alakzatokként azonosítottuk.

A munka során felhasznált pontfelhőket az MTA SZTAKI Velodyne HDL-64E S2 típusú Lidar készülékével rögzítettük Budapest utcáin, amely 15-20 Hz közötti frekvenciával szolgáltatott pontfelhő-kereteket egy mozgó autó tetejéről. Az egyes pontfelhők 60-100 ezer pontot tartalmaznak a szenzor 100-150 méteres körzetéből.

Munkánk távlati célja olyan információk kinyerése, melyet egy autonóm rendszer felhasználhat automatikus 3D rekonstrukcióra, önjáró autó vagy drónok navigálására, objektumok - objektumcsoportok detektálására vagy adott objektumok nyomon követésére.

# Abstract

Visual surveillance is central task of traffic management and control environment protection, accident and crime prevention applications in dynamic urban environments. To ensure an optimal exploitation of environmental information, besides traditional image sensors 3D data sources are frequently used nowadays. A Lidar laser scanner mounted on the top of a moving car provides a measurement sequence, where each frame is a three-dimensional point cloud, and the coordinate system is changing following the trajectory of the vehicle. Thus, we can directly obtain accurate 4D (space + time) geometric information from the scene, however, the analysis of the sparse point cloud sets present several challenges for the automatic pattern recognition algorithms. Crucial tasks are the automatic separation of the different urban objects from the background and recognition of objects based on the observed visual features. In this thesis I give an overview on several methods from the literature which deal with point clouds captured by Lidar sensor and fulfill segmentation tasks, ie. they separate the different semantic parts of the point cloud. By examining several methods I have found that separating closely located objects is a critical issue. For this reason, in my work I proposed a hierarchical two-level grid-based segmentation method for appropriately classifying the point sets. The advantages of the new method have been demonstrated versus conventional 3D segmentation approaches.

In the next step, I proposed methods to recognize vehicles, pedestrian crossings and traffic signs based on a joint utilization of geometric and intensity information stored in the Lidar point clouds. Detection of pedestrian crossings took the advantage of the fact that asphalt painting yields a greater intensity of the reflected laser light, and zebracrossings have typical size and shape parameters. The road signs are also identified by shape and intensity features. In the course of my work, I used point clouds obtained by the Velodyne HDL 64E S2 Lidar device of MTA SZTAKI on the streets of Budapest. The streams have been recorded with a frequency between 15-20 Hz from the top of a moving car. Each point cloud frame includes around 60-100 thousand points, and the radius of each scan is between in 100 to 150 meters.

My long-term goal is to extract information which can be used by an autonomous system for automatic 3D city reconstruction, navigation, object detection, recognition and tracking.

# 1. Bevezetés és Motiváció

A technika fejlődése során egyre nagyobb szerepet kapnak az olyan rendszerek, melyek önállóan képesek dönteni és működni vagy az ember számára hosszas, monoton feladatokat gyorsan és hatékonyan képesek elvégezni.

Ez a tendencia a számítógépes látás területén is megfigyelhető. Ahogy az embereknél az egyik legfontosabb érzékszerv a szem, a környezetükkel való kapcsolat egyik legfontosabb pillére úgy a gépek is minél több információt próbálnak kinyerni kameráik, képalkotó eszközeik adataiból. A hagyományos kétdimenziós képi információk mellett egyre nagyobb szerepet kapnak az olyan háromdimenziós érzékelők, mint a Lidar.

A Google önjáró autó vizuális szenzorrendszerének alapját egy Lidar alapú eszköz a Velodyne HDL-64E szenzor képzi. A jelen tanulmányban bemutatott munkám során lehetőségem nyílt egy ugyanilyen szenzorral dolgoznom, amely a következőekben ismertetésre kerülő algoritmusok adatait szolgáltatta. A Google autók már majdnem fél millió km-t tettek meg balesetmentesen. A Velodyne szenzor segítségével egy 3D-s térképet készítenek a környezetükről, melyben meg tudják különböztetni a gépjárműveket, az utakat, embereket, fákat, jelzőtáblákat, házakat és egyéb tereptárgyakat. A Lidar készülék mellett egyéb szenzorok is megtalálhatók az autókban. Egy nagy pontosságú GPS mely segítségével pontosan meg tudják állapítani egy nagy felbontású térképen, a helyzetüket. Az autók összefésülik a Velodyne 3D-s térképét a nagy felbontású térképpel így pontos képet kapnak a helyzetükről a valós világban, majd ezután mesterséges intelligencia algoritmusok segítségével döntenek a következő lépésről. Az autók elején és hátulján radar szenzorok találhatók, melyek segítségével elég távolra látnak, gyors forgalmi viszonyok mellett is. Kamerák is találhatóak a visszapillantó tükröknél, mely segítségével érzékelni tudják a közlekedési lámpák fényeit [11].

Hasonló rendszereket fejleszt a Toyota, Lexus, Audi és a Volvo is. A Lexusban beépített holttér figyelő rendszer áll rendelkezésre, amely infrakamerája segítségével meg tudja különböztetni a fém és fa alapú objektumokat az élő szövet objektumoktól [10], [11].

Az önjáró autók biztonságosabban képesek vezetni, mint az ember. Több információt képesek érzékelni a környezetükből, és sokkal gyorsabban képesek megfelelő döntést hozni. A „robotpilóta” nem iszik, és nem fárad, sötétben és rossz látási viszonyok között is képes gyorsan és jól dönteni szenzorjai segítségével.

Larry Page és Sergey Brin, a Google alapítói szerint a technológia nemcsak emberéleteket menthet meg, hanem a környezetre és a forgalomra is jó hatással lesz. Az önjáró autók képesek egymással kommunikálni, ezáltal gyorsabban elhárítani a közlekedési dugókat. Képesek konvojban haladni, akár egymástól négy méteres távolságban. A konvoj segítségével jobb kihasználtságot érünk el az autópályák terén és a légellenállás is kisebb lesz, mely jótékony hatással van a fogyasztásra, mely által gazdaságosabb is lesz a működés [11].

Az önjáró autóknak és robotoknak fel kell dolgozniuk a szenzorjaik által gyűjtött adatokat. Egy háromdimenziós pontfelhő esetén megkeresik az összetartozó pontokat, ponthalmazokat valamilyen geometriai vagy egyéb tulajdonságok alapján (intenzitás, rgb színkoordináta). Az így kapott objektumokat, objektumcsoportokat osztályokba sorolják, az egyes objektumokra jellemző tulajdonságok alapján.

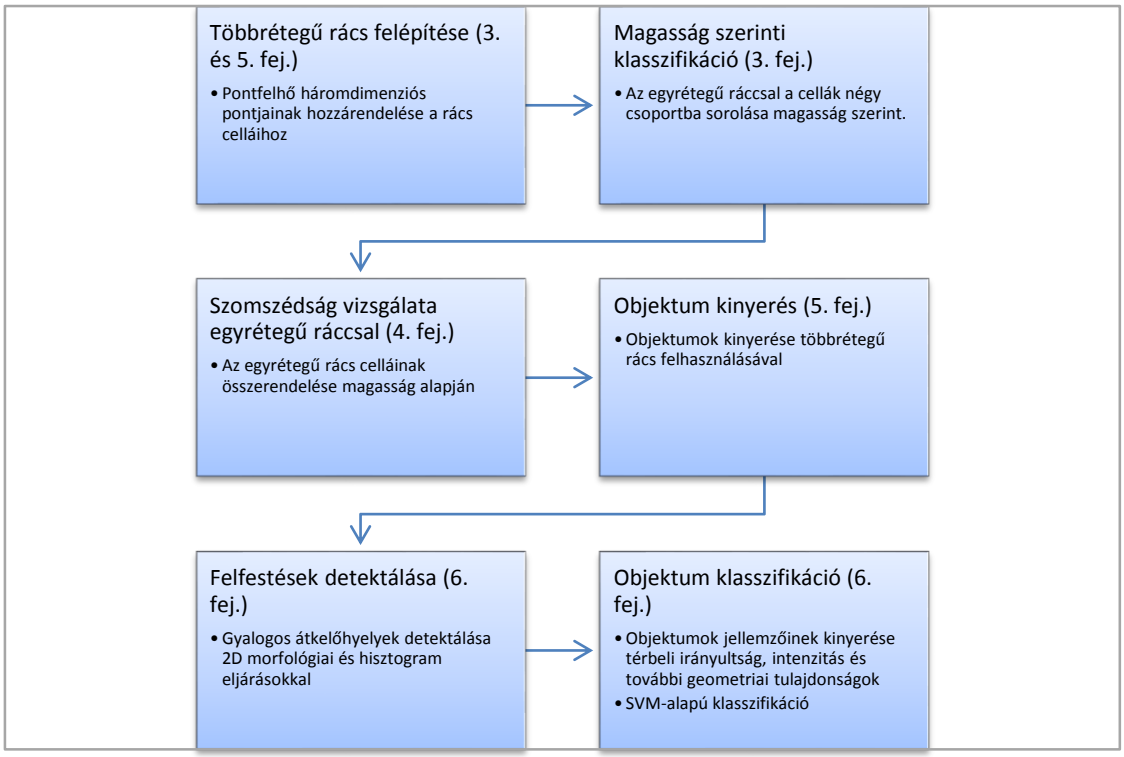
Az önjáró autó vagy robot így az adott objektumtípusnak megfelelő utasítást tudja végrehajtani. Például, ha az önjáró autó 10 méterrel maga előtt észlel egy másik autót, akkor elegendő kissé lassítania, de ha ez az objektum egy gyalogos, akkor már vészfékeznie kell.

Az objektumok szegmentálása és klasszifikálása nem csak az önjáró gépek esetében fontos. Légi vagy földi Lidar felvételek esetén, háromdimenziós városi rekonstrukció alkalmával, vagy forgalomfigyelésnél is fontos, hogy detektálni tudjunk különböző objektumokat. Egy ilyen alkalmazás lehet jelzőtáblák és útburkolati felfestések detektálása, majd annak elemzése, hogy a felfestések megfelelően látszódnak-e, esetleg melyiket kell újrafesteni.

Az önjáró autók és robotok elterjedésének a környezet értelmezésének nehézségén túl jogi és etikai korlátai is vannak, ezek részletezése azonban nem célja a dolgozatnak.

Az egymás közti kommunikációnál esetleges adatbiztonsági és kriptográfiai problémák merülhetnek fel. Meg kell oldani, hogy tényleg valós információt kapjunk, tehát egy támadó ne tudjon hamis információkat bejuttatni a kommunikációba, vagyis kell valami féle azonosítás, de azt se szeretnénk, ha mindenki tudná, hogy valamilyen fontos személy ül az autóban.

Olyan kérdések merülnek továbbá fel, hogy ki a hibás, ha baleset történik, a gyártó, a sofőr, aki nem vezet, vagy az autó? A számos nyitott kérdés ellenére a gyártók szerint a technológia nagyjából 5 éven belül bevezetésre kerülhet.




*1.1 ábra A feldolgozási folyamat lépései*

Dolgozatomban eljárást mutatok be mozgó objektumok kinyerésére, detektálására Lidar pontfelhő sorozatokon, majd az objektumok jellemzőinek kinyerésére. A jellemzők alapján különböző típusú objektumok (közlekedési tábla, gyalogos átkelőhely, személygépjármű) felismerésére teszek javaslatot, végül egy saját fejlesztésű klasszifikációs módszert ismertetek személygépjárművek felismerésére.

Munkám során arra is törekedtem, hogy a kifejlesztett algoritmusok futási ideje valós idejű legyen, azaz esetünkben a szenzor 15Hz-es frissítési frekvenciájával képesek legyünk a rögzített adatfolyamot online feldolgozni. Az 1.1 ábra szemlélteti a feldolgozási folyamat lépéseit:

- A dolgozat 2. fejezetében az érzékelő technológiákról kapunk egy rövid összefoglalót.
- A 3. fejezetben betekintést adunk a szakirodalomban szereplő megoldásokról és a szenzorból érkező adatok (pontfelhő pontjai) rács adatstruktúrába foglalásáról. Továbbá bemutatásra kerül az algoritmus elő-szegmentációs folyamata.
- A 4. fejezetben az objektumok szeparációjára alkalmazott egyrétegű rács kerül bemutatásra, annak alkalmazási lehetőségeivel, hátrányaival és előnyeivel együttvéve.
- Az 5. fejezet az objektum kinyerés továbbfejlesztett folyamatát mutatja be egy hierarchikus kétrétegű rács használatával.

- A 6. fejezetben bemutatom az objektum osztályozásához felhasznált jellemzőket, és ezen leírók alapján egy SVM alapú klasszifikációs folyamatot.
- A 7. fejezetben az algoritmus tesztelésének bemutatása és a kapott eredmények kiértékelése történik. Végül a 8. fejezetben néhány implementációs kérdéssel foglalkozik a dolgozat.

A dolgozatban bemutatott eredmények egyes lépései több nemzetközi konferenciapublikációban is ismertetésre kerületek: IEEE CogInfocom 2013 [1], ECCV 2014 Workshop (Lecture Notes in Computer Science) [2], és ACCV 2014 Workshop (Lecture Notes in Computer Science) [3].

# 2. Érzékelő technológiák

## 2.1 Optikai kamerák

Az optikai kamerák az emberi látás modelljét követik. Egy tipikus video felügyeleti rendszerben egy vagy több kamera figyeli a teret és a szoftverek próbálják a háromdimenziós világot kétdimenziós képek alapján értelmezni.

A hagyományos optikai kamerák több előnyös tulajdonsággal is rendelkeznek:

- Integrálhatóak olcsóbb ipari rendszerekbe is.
- Az RGB vagy grayscale kamerák adatain, a színek vagy a fényintenzitás alapján könnyen megkülönböztethetőek például a jelzőlámpák, utcai lámpák és az egyéb városi fények.
- Elérhetők kisebb képfrissítésű (20-30 fps) kamerák nagy (HD, UHD) felbontással, így kisebb objektumok felismerése is lehetővé válik, ugyanakkor a nagy képfrissítésű (60 < fps) kamerák is elterjedtek, melyek kisebb felbontással rendelkeznek, de a dinamikai változásokra sokkal érzékenyebbek.

A fentiek mellett azonban több hátránnyal is számolnunk kell:

- A számítógépes látás szakirodalma számos konkrét objektum felismerési feladatra még nem talált megbízható megoldást így az automatizált video felügyeleti rendszerek még közel sincsenek olyan szinten, hogy az elvárt fontos információkat automatikusan szolgáltassák a rendszerek biztonságos működtetéséhez.
- A megvilágítás és a fényviszonyok nagyban befolyásolják az optikai kamerák képeinek minőségét. Megjelennek árnyékok a megvilágítás irányától függően, melyek nehézségeket támasztanak a gépi felismerő algoritmusoknak.
- Sötétben vagy rossz látási viszonyok mellett külső megvilágítás szükséges, ami nem mindig kivitelezhető.
- A kamerák rendkívül érzékenyek az időjárási viszonyok megváltozására, nagy esőben vagy ködben nem látnak jól.
- A kameraképeken az objektumok mérete a kamerától való távolság függvényében változik.
- Nincs mélység információ, így nehéz a háttér és előtér elkülönítése.

## 2.2 Kinect és ToF szenzorok

A Kinect-ben egy RGB videokamera és egy mélységszenzor működik együtt, hogy létrehozzon egy úgynevezett 2,5 dimenziós modellt a környezetről és az objektumokról, ami színezett pontfelhőként vagy mélységképként is megjeleníthető. A kamera 640x480 pixel felbontású és 30 fps-re képes. A mélységszenzor tartalmaz egy CMOS szenzort és egy infravörös projektort, amik a háromdimenziós pontfelhő szintézisében játszanak szerepet. Az infravörös tartományban működő lézerprojektor egy ál-véletlen mintát vetít ki, melynek torzulása a tárgyakon lehetővé teszi a mélység meghatározását.
A Kinect a már a nagyközönség számára is elérhető kedvező ára miatt, de használata csak 3-4 m-es környezetben lehetséges és a napfényben lévő infra összetevők is megzavarják, ezért kültéri alkalmazásokra jelenleg nem alkalmas.

**Time of Flight szenzorok**

A Time of Flight kamerák aktív fényforrással vannak felszerelve (általában gyors működésre képes LED-ek), amik emberi szemnek láthatatlan, infra tartományú fényt bocsájtanak ki. Az objektumról, vagy felületről visszaverődő fény egy lencse segítségével a képalkotó szenzorra vetül. Az impulzus kibocsájtása és visszaérkezése között eltelt idő segítségével kiszámolhatjuk a megvilágított objektum távolságát minden egyes pixel esetében. A ToF kamerák, a Kinect-hez hasonlóan, egy valósidejű 2,5 dimenziós (az objektumnak csak a kamera szemszögéből látható felülete figyelhető meg) modellt alkotnak az objektumokról.

Napjainkban az egyik széleskörűen elterjedt ToF kamerát a MESA Imaging AG gyártja. A MESA kamerák chipjeiben használt CCD/CMOS szenzoroknak nagy a zajtűrő képességük így nagy pontosságú mérési eredményeket kaphatunk. A MESA szenzorok mérési tartománya 5-10 méter.

A Kinectet egyértelműen beltéri használatra tervezték, mert külső tereken, még szórt fényű nappali megvilágítás esetén is sokat romlik a mélységkép minősége. A MESA kamera valamivel jobban tűri a kültéri viszonyokat, de közvetlen napsütés esetén nagyon zajossá válik a mélységképe.

## 2.3 Lidar

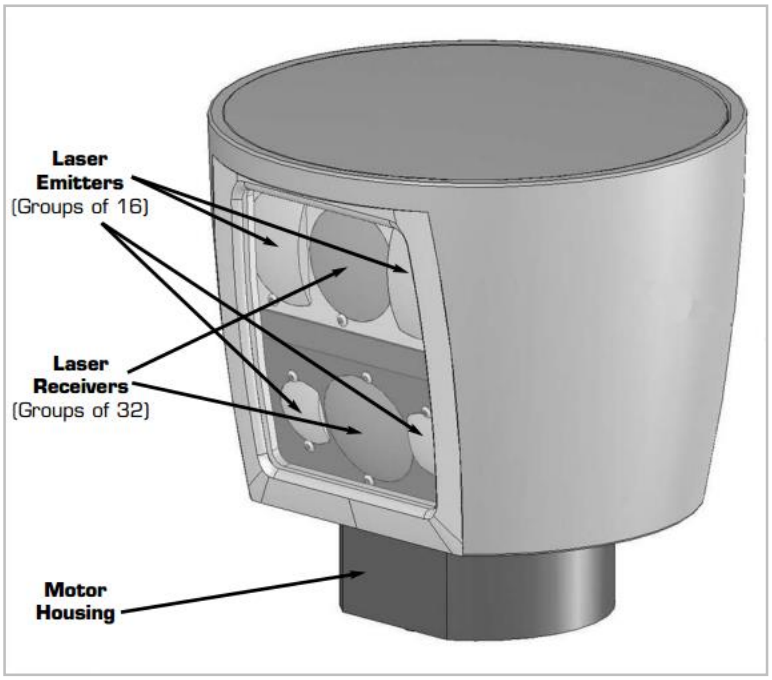
A Lidar az angol *Light Detection And Ranging* kifejezésből alkotott mozaikszó. Az 1960-as évek elején fejlesztették ki rövidesen a lézer feltalálása után. A Lidar-ban egyesült a lézer pontos

fókuszálhatósága a radar távolság meghatározó képességével. A Lidar nem más, mint egy optikai távérzékelő technológia, amely lézer fényt használ a környezet sűrű mintavételezéséhez. Az objektumról visszaverődő lézernyalábot a Lidar szenzor detektálja. A Lidar a lézer kibocsájtási és visszaérkezési idejéből kiszámolja a pontos távolságot a szenzor és a célobjektum között. A mérés eredménye egy nagy pontosságú háromdimenziós pontfelhő, ahol az egyes pontok helykoordinátája valamilyen lokális vagy globális koordinátarendszerben adottak.

A Lidar szenzorokat már napjainkban is széleskörűen használják, köszönhetően az eszközben rejlő adottságoknak. A Lidar eszközöket több nagyobb csoportba sorolhatjuk: léteznek légi vagy földi Lidar eszközök, sőt vannak víz alatti Lidarok is.
A légi Lidar eszközöket felhasználják többek között a térképészetben, erdőgazdálkodásban, várostervezésben és térfogatszámításban is.
A hagyományos infravörös lézer nyalábok a vízfelületekről visszaverődnek, de a zöld lézer képes keresztülhatolni vízen is és így a tengerek és óceánok fenekeinek feltérképezésére is lehetőség nyílik [17]. Ez a képesség sokat segít a tengerpartok, kikötők biztonságosabbá tételében.
A földi Lidar szenzorok felhasználása is nagyon széleskörű. Ezek az eszközök nagy pontosságú pontfelhőt adnak vissza, így lehetőség nyílik akár kisebb objektumok detektálására is. Az eszközt felhasználják 3D városmodellezésben, utak, városok infrastruktúrájának az elemzésében, önjáró autók működtetésében és még számos más területen.
A Lidar szenzorok típusoktól függően egy adott ponthoz az *x, y, z* helykoordinátákon kívül még számos további mért paramétereket rendelnek. Ilyen tulajdonságok a visszavert lézerfény intenzitása vagy a visszaverődések száma. Egyes mobil térképező rendszerek a Lidar szenzoron kívül kamerákat, IMU-t és GPS vevő készüléket is tartalmaznak, így az adott pontokhoz GPS koordinátát és RGB színkoordinátát is rendelnek.

### 2.3.1 A Velodyne Lidar szenzor

A Velodyne HDL-64E Lidar szenzort önjáró földi járművek és hajók navigálására fejlesztették ki. A szenzor 360˚-os horizontális és 26.8˚-os vertikális látószöggel rendelkezik. Az eszköz 5-15 Hz-es szkennelési sebességre képes és 1,3 millió pontot rögzítésére alkalmas másodpercenként.
A szenzor fején 64 lézersugár forrás található és minden fordulatkor egy teljes háromdimenziós szkennelés készül a környezetről. A szenzor 100-150 m-es hatótávolsággal rendelkezik. Az adatokat UTP kábel segítségével lehet letölteni az eszközről.




*2.1 ábra Velodyne HDL-64E LIDAR szenzor*

A szenzor az *x, y, z* relatív pozíció koordinátákon kívül intenzitás értéket is rendel az egyes pontokhoz. Az intenzitás egy adott pontról visszaérkező lézerfény erősségét adja meg. Az intenzitás értéke nagyban függ a lézernyaláb beesési szögétől, valamint az eltalált objektum felületétől. Az intenzitás kalibrálatlan érték, azaz nem lehet elvárni, hogy ugyanarra az objektumra különböző távolságokból és magasságokból ugyanazt az értéket kapjuk. Segíthet azonban különböző borítású felületelemek elkülönítésében.

A szenzort maximális, 15 Hz-es forgási teljesítményén használva keretenként 60-100 ezer pontot tartalmazó pontfelhőkből álló idősorozatot hoz létre. Ezzel a sebességgel és adatsűrűséggel lehetőség nyílik a környezet dinamikai változásainak a megfigyelésére. A lassabban forgó eszköz sűrűbb és nagyobb pontfelhőkből álló idősorozatot állít elő, de az egyes pontfelhők közötti dinamikai változások nagyobbak lesznek.

A Velodyne HDL-64E Lidar szenzorok nagy sebességük mellet nagy pontosságú (< 2cm) pontfelhőt képesek előállítani. Az eszköz ára jelenleg még lényegesen drágább, mint a sztereo kamera alapú rendszereké, de a lézeres méréssel nagyobb pontosság érhető el. A szenzor aktív fénnyel dolgozik, ezért éjjel-nappal működhet.

A Velodyne szenzor által készített pontfelhőben az objektumok geometriai tulajdonságai jól kezelhetőek. Egy háromdimenziós pontfelhőben meghatározható egy adott objektum térbeli elhelyezkedése, ami lehetővé teszi az objektumok szeparálását.

A Velodyne Lidar szenzorok széleskörű elterjedésének a legfőbb akadálya jelenleg a magas forgalmi áruk, de ahogy a kamerák és egyéb szenzorok ára is drasztikusan csökkent az elterjedésükkel így ez valószínűleg a Lidar szenzorok esetében is igaz lesz.

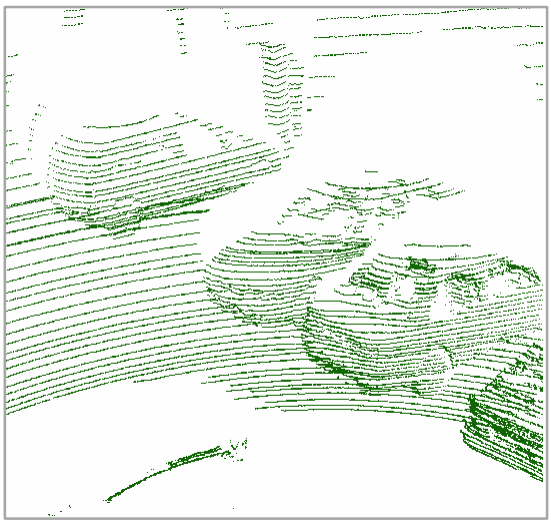
A jelen dolgozatban kizárólag Lidar felvételek kiértékelésével foglalkozunk, azonban megjegyezzük, hogy a Lidar kamera természetesen felhasználható többszenzoros mobil térképező rendszerek egy elemeként is [pl. Google autó [11]], együtt használva videokamerákkal és radar szenzorokkal több modalitású információt rögzítve a megfigyelt világról [11].


*2.2 ábra Az MTA SZTAKI Velodyne típusú szenzora*

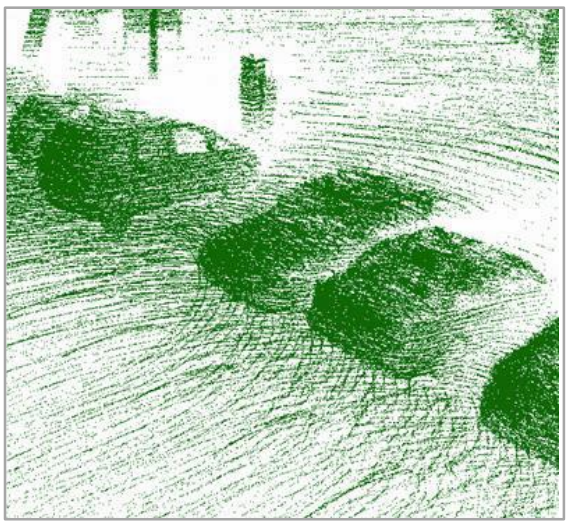
A dolgozatban felhasznált adatok a (2.2) ábrán szereplő szenzor segítségével készültek. Az eszközt egy autó tetejére szereltük és normál forgalmi sebességek (20-50 km/h) mellett Budapest utcáin rögzítettünk adatokat.

# 3. A kapcsolódó szakirodalmi eredmények áttekintése

Ebben a fejezetben a pontfelhő feldolgozással kapcsolatos szakirodalmi eredményeket tekintjük át. A Velodyne Lidar szenzor által készített nyers pontfelhőkből információkat szeretnénk kinyerni. Feldolgozhatjuk az egyes időkereteket (azaz pontfelhőket) egyesével vagy összeregisztrálhattunk több pontfelhőt azaz, egy közös koordinátarendszerbe transzformáljuk azokat, hogy sűrűbb, részletesebb pontfelhőt kapjunk a későbbi feldolgozásokhoz. Az, hogy a feldolgozó rutinjainkat az egyes időkereteken vagy az összefűzött pontfelhőn hajtjuk végre, nagyban függ az alkalmazás céljától. Egy városrekonstrukciónál sűrűbb, pontosabb felhőre van szükség, ezért érdemes összeregisztrált pontfelhőt használni. A nagy felhő miatt a feldolgozás offline lesz, de itt a cél nem az azonnali eredmény elérése, hanem a pontosság és élethűség. Viszont, ha egy autonóm járművet szeretnénk vezérelni, akkor sokkal fontosabb a dinamikai változások nyomon követése és elengedhetetlen a valós idejű döntés, így szükséges, hogy a regisztrálatlan pontfelhőn dolgoznunk.



*3.1 ábra Egy mérési időkeret, azaz regisztrálatlan pontfelhő vizualuzációja*



*3.2 ábra Regisztrált pontfelhő 10 időkeret összefűzésével készült*

A pontfelhő feldolgozásában az első lépés, hogy a pontfelhő pontjaihoz címkéket rendelünk egy előre meghatározott címkekészletből (úttest, magas objektum, alacsony objektum, egyéb), ezt nevezzük a pontfelhő szegmentációjának.

Ahhoz, hogy szegmentálni tudjuk a pontfelhőt, szükséges egy hatékony pont szomszédsági adatstruktúra felépítése. Továbbá szükséges olyan kritériumok megfogalmazása, melyek által az egyes pontokat és szomszédjaikat összefüggő objektumokba, objektumcsoportokba tudjuk sorolni. A szomszédsági modell felépítésére számos jól bevált technika létezik, úgymint a kd-fa alapú eljárások [6] vagy a rács alapú eljárások [7].

## 3.1 Kd-fa alapú eljárások

Tekintsünk egy olyan felosztást, amely a teret egy olyan síkkal vágja el, amely az objektumteret a lehető legigazságosabban felezi meg. Ez a módszer egy bináris fához vezet, amelynek neve bináris térpartícionáló fa, vagy BSP-fa. Ha a felezősík mindig merőleges a koordinátarendszer valamely tengelyére, akkor kd-fa adatszerkezetről beszélünk.

A kd-fa alapú térpartícionálás után különböző klaszterező eljárásokkal mind az összeregisztrált mind a regisztrálatlan pontfelhők szegmentációjánál is nagyon pontos eredményeket lehet elérni, de az eljárásnak több hátránya is van. Ugyan az eljárás a legtöbb esetben pontos az objektumok szeparálásában és a fában való szomszédság keresés is gyors, de a fa felépítése sokszor lassú az adott probléma megoldásához. Minden egyes új időkeretnél újból fel kell építeni a kd-fát, ami nem eredményezhet valósidejű feldolgozást.

A Velodyne szenzorral rögzített adatok egy jellegzetes tulajdonsága, hogy a szenzor 5-20m-es környezetében a pontok sűrűsége nagy, viszont ettől távolodva a pontsűrűség gyorsan csökkenő karakterisztikát mutat. A változó pontsűrűség miatt a kd-fa alapú eljárások optimális paraméterezése sem triviális feladat.

A térpartícionálás után az egyik leggyakrabban használt klaszterező eljárás a területelárasztó (floodfill) algoritmus. A területelárasztó eljárás paraméterezése jól meghatározható egy adott objektumtípusra, például autóra, gyalogosra vagy egyéb objektumra, de sok esetben nem kivitelezhető olyan paraméterezést találni, ami az összes keresett objektumra egyszerre működik. Ezért a kd-fa alapú eljárások gyakran túl- vagy alulpartícionálják a teret.

## 3.2 Szabályos 2D rács

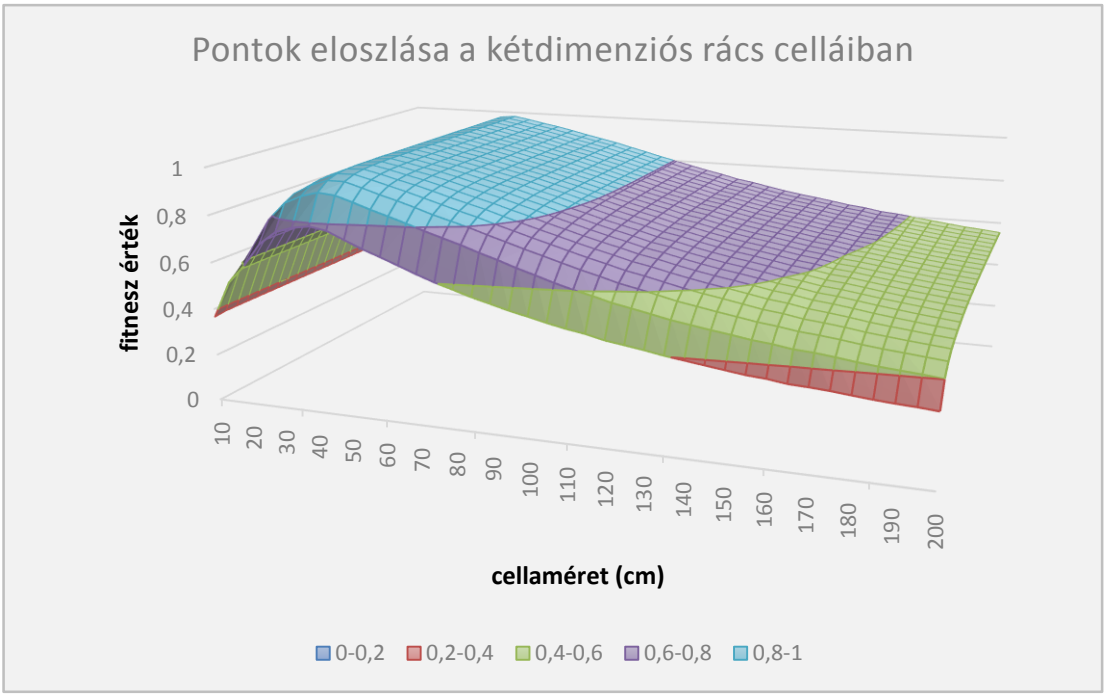
A 2D rács alapú eljárások első lépésben egy kétdimenziós négyzethálót (rácsot) definiálnak, majd ezt illesztik valamelyik vízszintes síkra, tipikusan a talaj becsült síkjára [5]. Szabályos rácsok esetében a négyzetháló cellái azonos nagyságúak. A rács létrehozása után a pontfelhő pontjait hozzárendeljük a rács megfelelő celláihoz. A hozzárendelés a talaj síkja szerinti projektálás. Ezután a rács egyes cellái tartalmazni fogják a pontfelhő egyes lokális részeit. A rács alapú szegmentálás lényege, hogy az egyes cellákhoz szegmentációs osztályt rendelünk a cellákhoz tartozó pontfelhő részletből kinyert statisztikai jellemzők (sűrűség, magasság, szórás) alapján.

A cellák mérete nagyban befolyásolja a futási időt és az eredmény pontosságát, ahol az eredmény egy olyan szegmentált pontfelhő, amiben az egyes objektumok elkülönülnek. Egyre kisebb cellaméret esetén, az egyes cellák sokszor nem tartalmaznak elég pontot, hogy releváns

jellemzőket lehessen kinyerni a lokális felhőrészletből. Továbbá a cellaméret csökkenésével a négyzetháló egyre több cellát fog tartalmazni, ami egyre lassabb futási időt fog eredményezni. Nagyobb cellaméret esetén a futási idővel gyors, viszont így sokszor túl sok pont esik egy cellába. Mivel a módszer alapegysége a cella ezért, ha egy cellába esik pl. két autó, vagy több gyalogos egy-egy részlete, akkor hibásan egy közös objektumként fognak megjelenni a detekciós eredményben.

A következőkben a megoldási algoritmusomat ismertetem. Egy cellát „megfelelő”-ként definiálok, ha az alábbi két kritérium teljesül:

- A cellának több pontot kell tartalmaznia, mint Q (esetünkben Q = 20).
- A cella maximum R pontot tartalmazhat, ezt a felső korlátot a vizsgálatok során 100 és 2500 pont között változtattam.

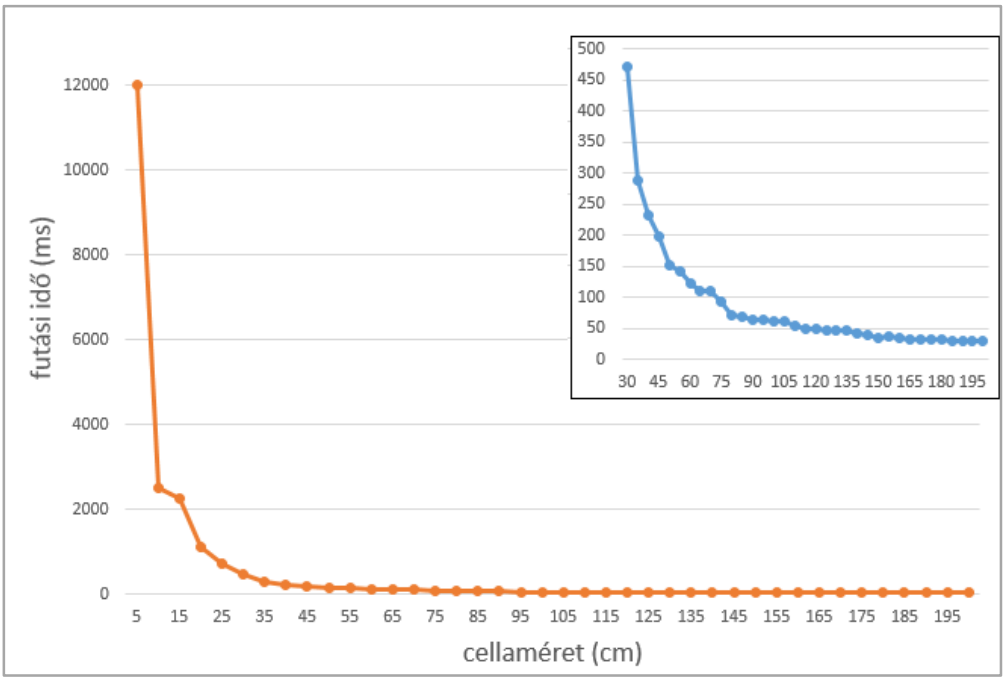



*3.3 ábra Rács cellák jósága, a cellaméret függvényében*

A kétdimenziós rácsban meghatározzuk, hogy mennyi olyan cella van, ami megfelel az elvárásainknak, azaz nem üres és több pontot tartalmaz, mint Q, de kevesebbet, mint az aktuális R felső korlát. A mérés során Q-t, azaz az alsó korlátot 20-nak vettem, mert egy 20 pontból álló objektumot a legtöbb esetben még lehet osztályozni, azaz egy kisebb utcai objektumot (gyalogos) még fel lehet ismerni. Ezután definiáltam egy *fitneszértéket*, ami megmutatja, hogy a pontfelhő pontjai milyen arányban töltik ki a kétdimenziós rácsot, azaz a megfelelő cellák száma osztva az összes cella számával.

Az eljárás feladata minimalizálni azon cellák számát, melyek túl kevés pontot tartalmaznak, ahhoz, hogy információt lehessen kinyerni belőlük, vagy éppen túl sok pontot tartalmaznak, és így elvesznek a részletek.

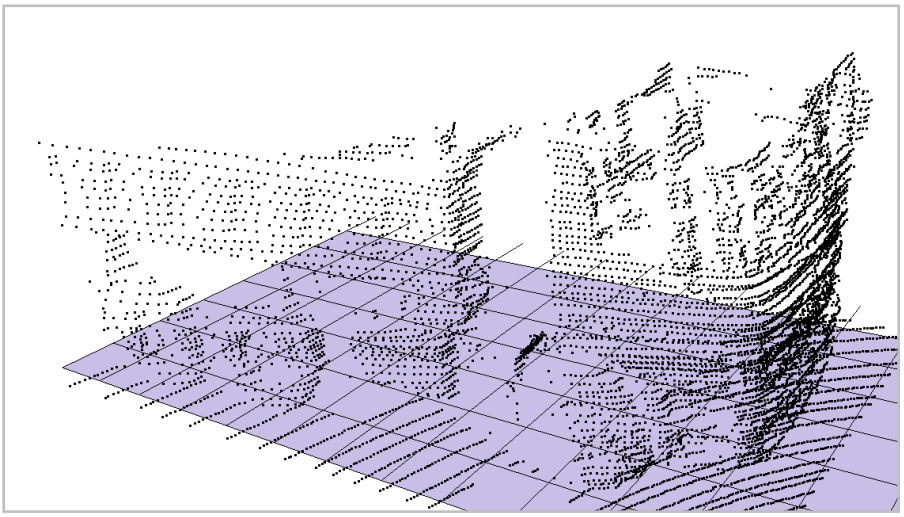
A (3.3) ábrán megfigyelhető, hogy a 0.9 fitneszérték fölötti cellák a 30-60 cm-es tartományba esnek (világoskék tartomány teteje). Ez az a cellaméret tartomány, ahol a legtöbb megfelelő kitöltöttségű cellát kapjuk. Az ábrán a harmadik dimenzió a különböző felső korlátokat (R) reprezentálja, amiből megfigyelhető, hogy a felső korlát változásától nagyjából független a mérési eredmény.




*3.4 ábra Futási idő, a cellaméret függvényében*

A (3.4) ábrán látható, hogy a cellaméret növelésével a futási idő egyre csökken. A mérés során a szenzor 5-10 Hz sebességgel forgott. Ahhoz, hogy valós időben lehessen feldolgozni az adatokat, a cellaméretnek minimum 45 cm-nek kell lennie.

A (3.3) és (3.4) mérések eredményeiből arra következtettem, hogy a legjobb eredményt (kellően gyors és pontos) 45-60 cm-es cellamérettel lehet elérni.



*3.5 ábra Rács alapú modell, a talajsíkra vetített négyzetháló szemléltetése*

A Velodyne szenzor látótávolsága 100-150m a környezettől függően. 50cm és 80cm közötti cellaméretttel a látóteret lefedő, nagyjából 20-40 ezer cellára osztódik a vízszintes látótér és ez a méret gyors feldolgozást biztosít.

A rács alapú megközelítés nagy előnye, hogy a pontfelhőt lokális kis felhők összességeként tudjuk kezelni. Ez azért hasznos, mert a mérésből és tükröződésből adódó zajpontok csak kevés cellát fognak érinteni. Ezek a zajpontok nagyban csökkentik a geometriai jellemzőkön alapuló felismerő algoritmusok hatékonyságát a rács alapú megoldásnál azonban lokálisan kezelhetők az ebből adódó problémák, hiszen az egyes zajpontok csak az adott kapcsolódó cellát érintik közvetlenül ahelyett, hogy az egész felhőre hatással lenne a jelenség.

A teljesség kedvéért megjegyezzük, hogy a szakirodalomban megtalálható az úgynevezett egocentrikus rács modell is [12], mely nem egy szabályos négyzethálóra bontja fel a teret, hanem a szenzor körül értelmezett körgyűrűk cikkeire. Ennek a modellnek az alapgondolata egybeesik a Velodyne szenzor forgás általi adatrögzítésével, és az egyes körcikkekben hatékonyan lehet megállapítani, hogy van e valamilyen objektum a szenzor előtt, azonban a tényleges objektumok kinyerése nehezebb a változó méretű és elhelyezkedésű körgyűrű szeletekből.

## 3.3 Szegmentálás szabályos 2D ráccsal

Célunk, hogy a nyers pontfelhőt objektumokra, illetve objektumcsoportokra osszuk fel. A kétdimenziós rács használatával az alábbi algoritmus alapján a felhő pontjai gyorsan és egyszerűen sorolhatók különböző szemantikai osztályokba.

Elsőként a pontfelhő pontjait az egyes cellákhoz rendeljük, és kiszámoljuk az adott cella lokális felhőjébe tartozó maximális és minimális magasságú pontok y magasságkoordinátáinak

különbségét, így egyszerű geometriai úton meghatározzuk az adott cellába eső objektum objektumrész magasságát:

$$magasság = y_{max}(s) - y_{min}(s),$$

ahol s az adott cella, $y_{max}(s)$ az s cella legmagasabb pontjának a magassága és $y_{max}(s) - y_{min}(s)$ az s cellába eső objektumrész "magassága" [5].

Azokat a cellákat, melyek kevés (maximum 8-10) pontot tartalmaznak, megjelöljük, hogy a feldolgozás során ne foglalkozzunk velük, mert túl kevés adatot tartalmaznak ahhoz, hogy releváns statisztikai információt nyerhessünk ki belőlük.

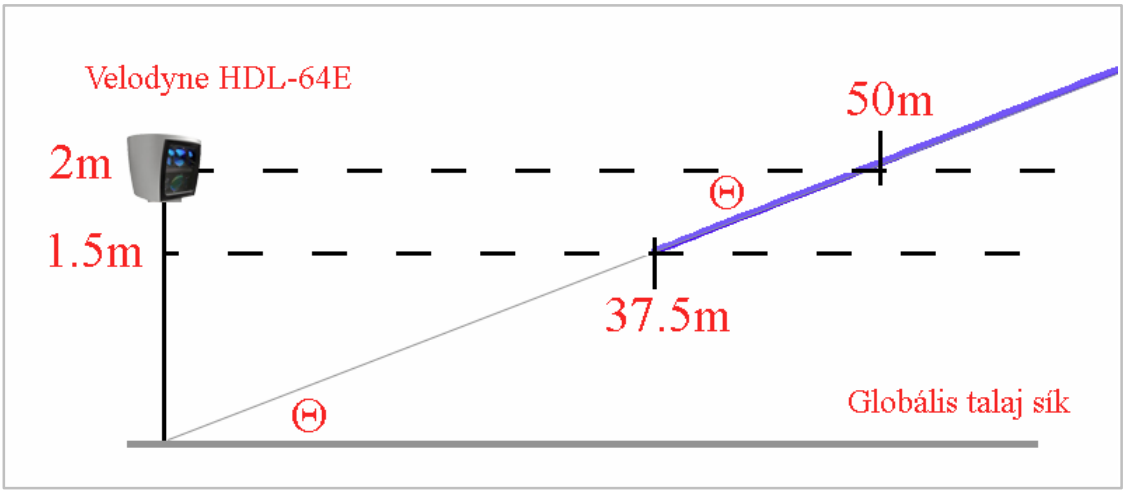
Ezek után a többi cellát három csoportba sorolhatjuk:

- úttest vagy talaj
- alacsony objektumok (autó, gyalogos, egyéb mozgó járművek)
- magas objektumok (fal, fa, jelzőlámpa, oszlop)

A talaj cellák meghatározása során az algoritmushoz tartozó kritériumok a következők voltak:
S cella címkéje „talaj", ha:

$$(1)\ y_{max}(S) - y_{min}(S) < 25\ \text{ÉS}\ y_{max}(S) < -50$$

Az első kritérium alapján egy 50-80 centiméteres cellán belül nem lehet nagyobb a talaj dőlésszöge nagyobb, mint 30˚. A második kritérium azt küszöböli ki, hogy a magasabban lévő vízszintes objektumelemeket, például járművek motorháztetőit, vagy platóit ne detektálja talajnak az eljárás. Ez a feltételrendszer azonban csak a szenzor közeli környezetében ad helyes eredményt, már az az út esetleges enyhe emelkedése esetén is a távolabbi régiókban hibák léphetnek fel, amit a következő példa szemléltet.




*3.6 ábra Talaj hibás meghatározása az út emelkedése miatt*

A (3.6) ábrán a Velodyne szenzor egy olyan útszakaszon található, ami $\theta$ enyhe fokos emelkedést mutat, úgy hogy 50m-re a szenzortól az úttest magassága már megegyezik a szenzor magasságával. Tegyük fel továbbá, hogy a szenzor 2 méter magasban helyezkedik el az úttesttől. Ekkor a $\theta$ szög kiszámítható:

$$tan\,\theta = \frac{2}{50} = 0.04 \;\rightarrow\; \theta = 2.29°$$

A (3.4) ábrán látható módon ekkor x távolságra a szenzortól, már 0.5m-nél közelebb lesz az úttest a szenzorszinthez képest, ahol:

$$\tan 2.29 = \frac{2-0.5}{\mathrm{x}} \rightarrow \mathrm{x} = \frac{1.5}{\tan 2.29} = \frac{1.5}{0.04} = 37.5\mathrm{m}$$

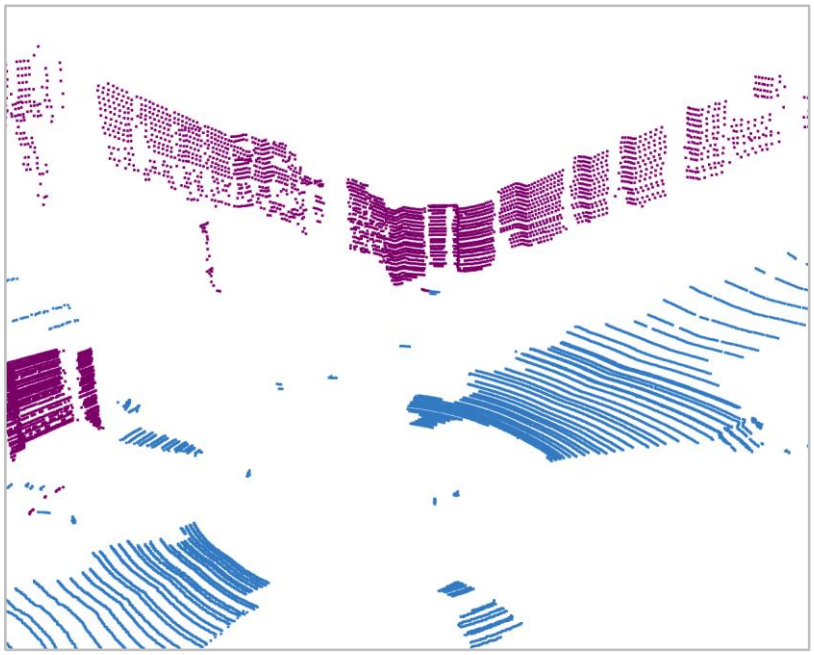
Tehát a szenzortól 37.5m távolságra a valódi talajmagasság 2 - 0.5 = 1.5m magasságban van. Ez azt jelenti, hogy azok a cellák, amik távolabb vannak a szenzortól, mint 37.5m sohasem kaphatnak talaj címként.

Ennek kiküszöbölésére javasoltam egy kiegészítő lépést [4]-hez képest. Miután az algoritmus lefutott az összes cellára (1) feltételrendszert felhasználva ($y_{max}(S) < -50$), kiszámolok minden egyes cellához egy lokális talajmagasságot. Ezután valamennyi cella magasságszintjét átállítom: az adott cella a szomszédos cellák minimummagasságainak az átlagát fogja kapni lokális átlagmagasságnak.

A magas objektumok osztályát hasonlóan írjuk le, mint a talajt, csak két kritérium VAGY kapcsolatával határozzuk meg a szűrési feltételt:

S magas objektum, ha:

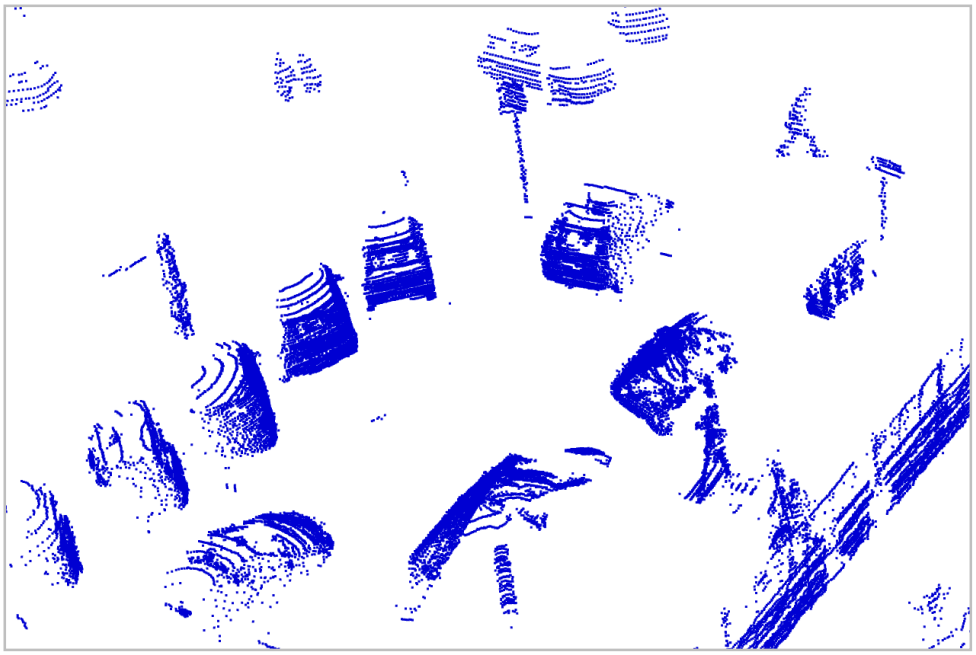
$$y_{max}(S) - y_{min}(S) > 310\; VAGY\; y_{max}(S) > 140$$



*3.7 ábra Magas objektumok talajjal (kék:talaj, lila:fal)*

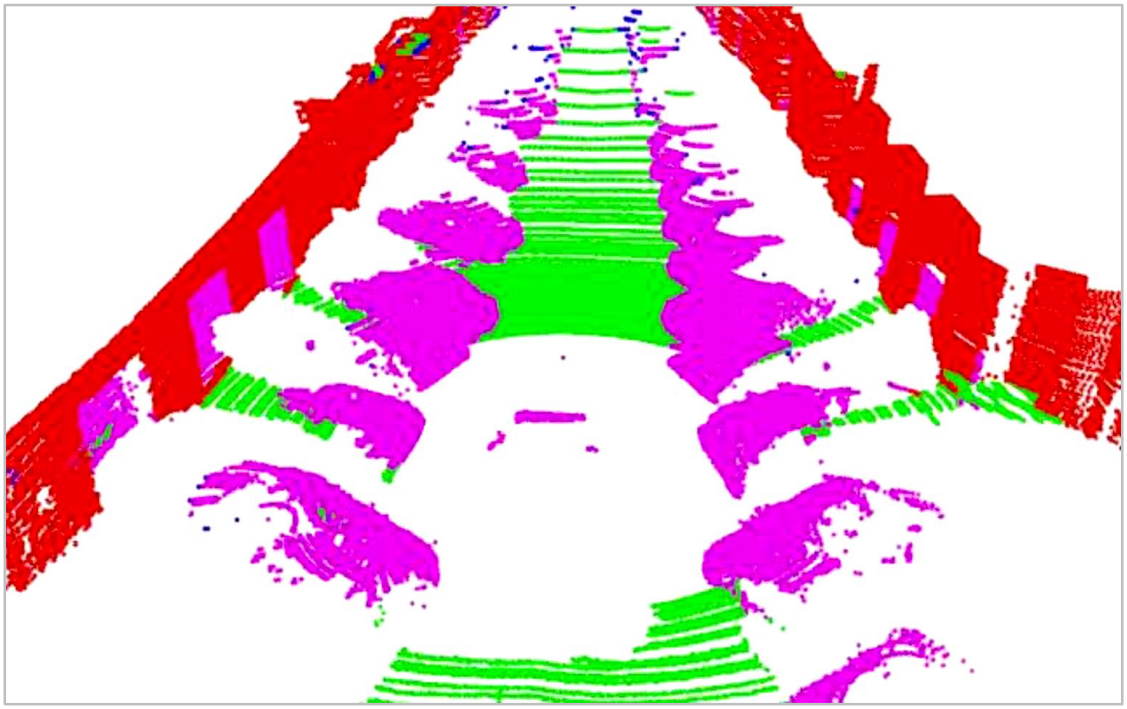
Tehát, ha az adott cellába eső objektum magassága nagyobb, mint a szenzor magassága plusz 140cm vagy a magasság különbség nagyobb, mint 310cm, akkor feltételezzük, hogy a cella magas objektumot tartalmaz.

A pontfelhő további celláit, melyek sem a talaj, sem a magas objektum osztályba nem tartoznak, az alacsony utcai objektumok kategóriába soroljuk.



*3.8 ábra A pontfelhőből egyrétegű ráccsall kinyert alacsony objektumok (tábla, autó fal, ember)*

A (3.8) ábrán látszik, hogy egyes táblákat és falrészeket is objektumnak sorolt be az algoritmus. Ez természetes, hiszen mind a táblák, mind a falak mérete nagyon változó, ráadásul gyakran a szenzor sem lát elég magasra a pontos méret megállapításához.



*3.9 ábra Magasság alapú szegmentálás (lila: alacsony utcai objektumok, piros: magas utcai objektumok, zöld: talaj)*

A magasság szerinti szegmentálással az egyes cellákat gyorsan be lehet sorolni a fent említett három osztályba, de ez csak egy bizonyos pontossággal működik a környezet szabálytalansága (pl. nincs standard táblaméret) és a szenzor tulajdonságai miatt (limitált látótér).

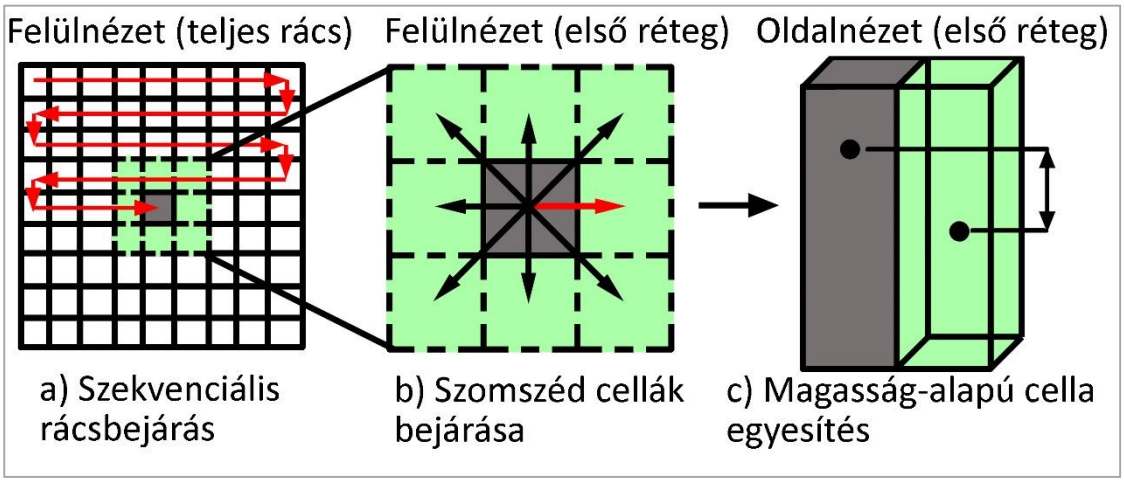
A következő lépésben természetes ötletként merül fel, hogy ne csak magasság szerint szegmentáljunk, hanem nézzük az objektumok két, másik irányú kiterjedését is. A cellaalapú feldolgozás miatt ezt könnyen meg tudjuk tenni, de csak egy-egy cellán belül. A nagyobb objektumok (autó, fal, stb.) a méretüktől függően több cellába esnek, például egy átlagos autó kb. 1.5-2m széles és 4-4.5m hosszú, így 60x60 centiméteres cellaméret mellett 15-20 cellába esik. A cellaalapú feldolgozás csak magasság szerinti szegmentálásra alkalmas, mert egy-egy cellába egy nagyobb objektumnak csak egy kis része esik.

# 4. Objektumok kinyerése városi környezetben

A 3.3-as fejezetben láttuk, hogy bár az egyrétegű rács alapú szemlélet gyors és a pontfelhő 3-4 csoportba való szegmentálására alkalmas, az egyes cellák között nincs kapcsolat. A további szegmentáláshoz (csoportokon belüli osztályok létrehozása, pl. ember, autó, tábla) új szemléletmódra van szükség.

Ahhoz, hogy egy objektumot (pl. autó, tábla) egy egységként tudjunk kezelni, valamilyen kapcsolatot kell definiálni, és feltérképezni azon cellák között, amelyek tartalmazzák az objektum különböző részeit.

## 4.1 Cellák egyesítése objektumokká




*4.1 ábra Magasság alapú hasonlóság a szomszédos cellák között. Ha a megadott értéken belül van, akkor egy objektumhoz tartozik a két cella.*

Munkám során kihasználtam, hogy tudjuk az egyes cellákban az aktuális felhő maximális magasságát. Egy olyan heurisztikát veszek alapul, hogy az egybefüggő objektumoknál az egymás melletti cellák között a maximális magasság különbség kisebb, mint 50cm. Ez a feltétel akkor dől meg, ha az adott cellaméret mellett egy cellaméretnyi távolság alatt 50cm-t változik a maximális magasság. Ez azt jelenti, hogy 50-80 centiméteres cellaméret mellett 50cm-t kellene változnia a magasságnak. Az általunk detektálni kívánt objektumok nagy részénél ez a heurisztika megfelelően működik. Tehát a módszer az egyes cellákat magasság szerint sorolja különböző csoportokba.

A szomszédos cellák közötti hasonlóság keresése során több lehetőség is felmerült, melyeket különböző okokból vetettem el. Az egyik ilyen lehetőség a maximális magasságok helyett az

átlagmagasságokkal számolt. Ez az alternatív módszer is közel olyan jól működött, mint az előzően ismertetett maximális magasság alapú csoportosítás, viszont egy jelentős hátrányát is megfigyeltem: A szenzor inhomogén sűrűségű pontfelhőt biztosít, ezért a cellákon belül mért átlagmagasságok sokszor torzulnak.

A cellák közötti hasonlóság keresése lehetne intenzitás alapú, vagy lehetne a cellák közötti pontsűrűséget figyelni, de akár az adott cella pontfelhőjének a térbeli irányultságát is lehetne vizsgálni. Azonban az utóbbi két lehetőség egyaránt a futási idő rovására mennének.

## 4.2 Objektumkereső algoritmus

A következőekben bemutatok egy objektumkereső algoritmust, ami kimenetként különálló pontfelhőkben tárolja el a megtalált objektumokat.

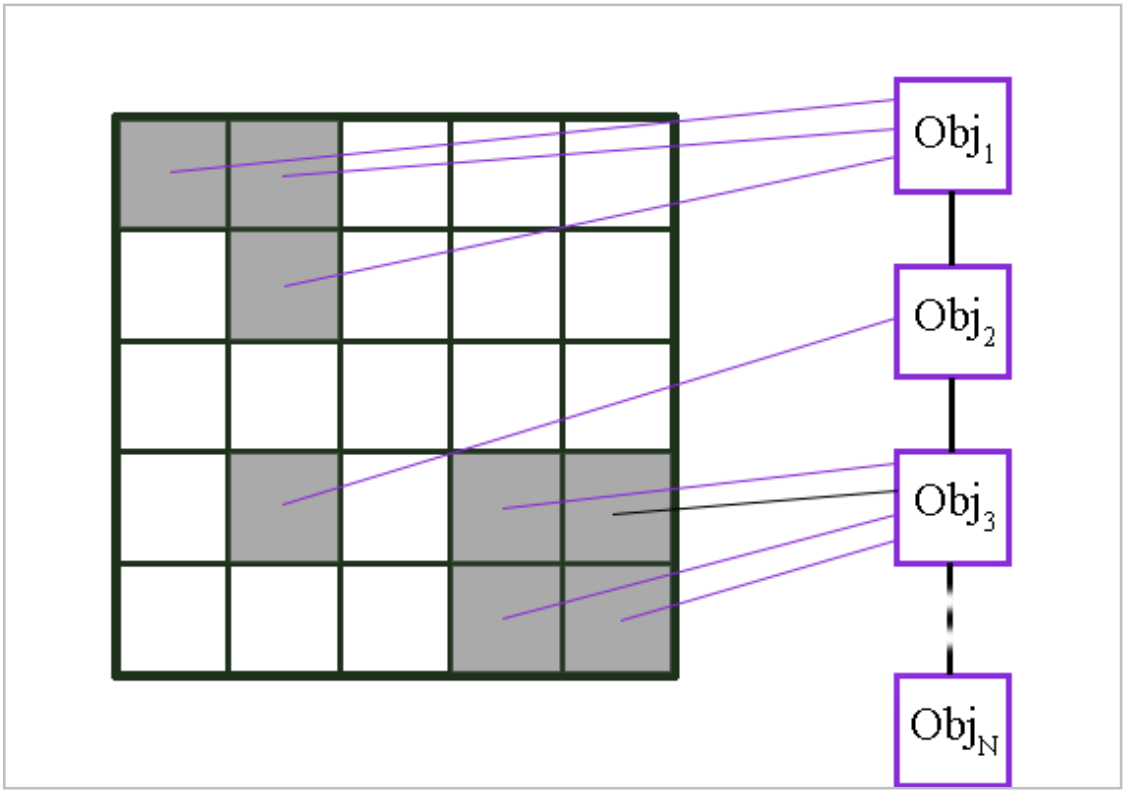
Először az adott cellákat - a (4.3) módszert felhasználva - a tartalmazott pontok magassága szerint négy csoportba soroljuk: talaj, alacsony objektum, magas objektum vagy üres cella. Erre egyrészt azért van szükség, mert a talaj cellákat szeretnénk kizárni az összefüggő utcai objektumok lehetséges régiói közül, másrészt egy-egy objektumot többnyire talaj cellák határolnak határt képezve az objektumok között. Általában egy-egy pontfelhő szekvenciánál a cellák 50-75 százaléka talaj vagy üres cella, ezért szűrésükkel az (4.1) fejezetben ismertetett algoritmuson is sokat tudunk gyorsítani.

A cellák négy csoportba való sorolása után csak az alacsony és magas objektum osztályba sorolt cellákkal dolgozunk tovább. Az algoritmus minden egyes cellának megvizsgálja a szomszédjait és dönt, hogy azok egy objektumhoz tartoznak-e.

```
1
2  // szomszédság vizsgálása
3  bool A = (y + i >= 0 && x + j >= 0 && y + i < cellnumx && x + j
   < cellnumz)
4  bool B = (grid[y+i][x+j].cluster != SPARSE)
5  bool C = (grid[y+i][x+j].cluster != ROAD)
6  bool D = (abs(grid[y][x].max_y - grid[y+i][x+j].max_y) <= 50)
7  for(int i = -1; i < 2; i++){
8      for(int j = -1; j < 2; j++){
9          if(A){
10             if(B && C && D){
11                 grid[y+i][x+j].id = grid[y][x].id;
12             }
13         }
14     }
15 }
16
```

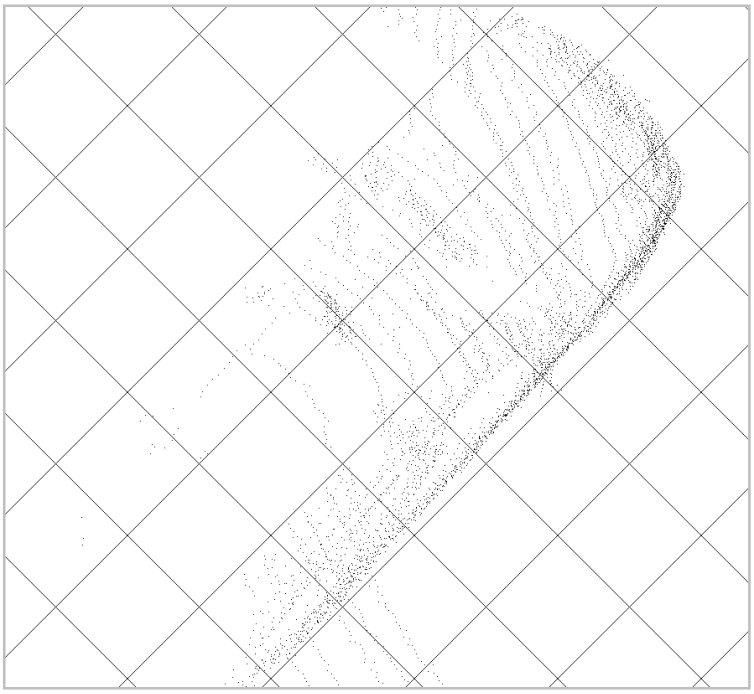
*4.2 ábra Az (4.1) algoritmus forráskódja*

Az (4.2) ábrán bemutatott forráskód szemlélteti az algoritmus működését. Az algoritmus 8 szomszédságot vesz figyelembe az objektumok keresése során. Az A feltétel garantálja, hogy a rács celláiból ne lehessen ki-indexelni. A B és C feltétel azért szükséges, mert ha a szomszédos cella üres vagy talaj, akkor nem kerülhet bele az objektumba.
A D feltétel vizsgálja a két cella közötti magasságkülönbséget. Ha az érték a megadott határon belül van, akkor a két cella ugyanazt a jelölést kapja és egy objektumhoz fognak tartozni.




*4.3 ábra Rács alapú modell és az objektum elvű szemlélet kapcsolata*

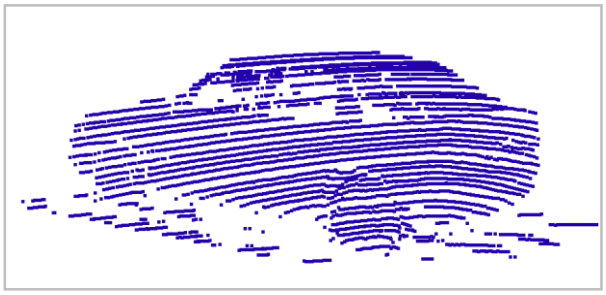
Az (4.3) ábra illusztrálja az algoritmus működését. Az összetartozó cellákat összeillesztjük egy objektummá, és ezek után már tudunk a teljes objektumokra vonatkozó jellemzőket kinyerni.
A cella alapú modell megmarad, és az egyes cellákhoz közvetlenül meghatározható, hogy a szomszédos celláik közül kikkel tartoznak egy objektumba, vagyis melyik szomszédos cellák érhetők el belőlük közvetlenül. Implementációmban az objektum-térben az egyes objektumok mutatókon (pointereken) keresztül érik el az őket alkotó cellákat.
Innentől kezdve tudunk mind az objektumtérben, mind a cellák között hatékonyan keresni, és az objektumok klasszifikálásához szükséges tulajdonságokat is ki tudjuk nyerni.



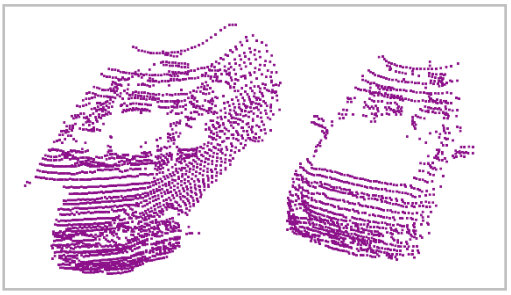
*4.4 ábra Autó objektum elhelyezkedése a 2 dimenziós rácsban felülnézetben*

Az (4.4) ábra szemlélteti a 2 dimenziós rács és a 3 dimenziós pontfelhő viszonyát. A rács cellái a fenti példában 60x60 centiméteresek.

## 4.3 Objektumkinyerési eredmények



*4.5 ábra pontfelhőből kinyert jármű*



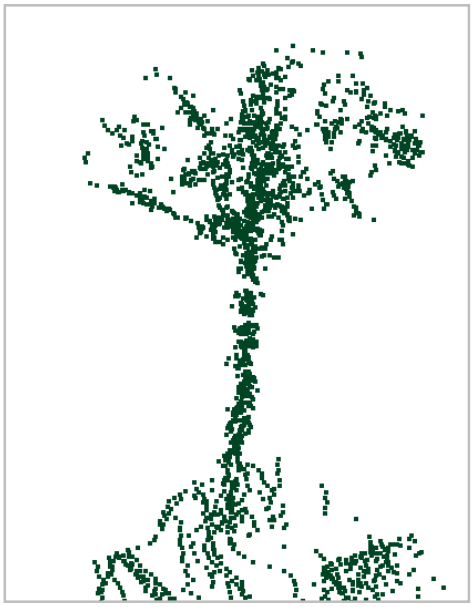
*4.6 ábra pontfelhőből kinyert járművek*

Az előző feladatban ismertetett objektumkinyerő algoritmust különböző városi pontfelhő részleteken teszteltem, eredmények a 4.5-4.11 ábrákon láthatók. Tipikus hibák figyelhetők meg a 4.6 és 4.8 ábrákon, ahol az egymáshoz közel parkoló autókat, és közel álló embereket egy objektumnak ismerte fel a javasolt módszer. Ennek oka, hogy a két objektum közel helyezkedett egymáshoz és a magasságkritérium megegyezett a szomszédos celláknál

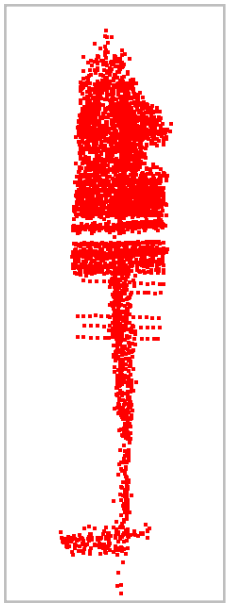
*4.7 ábra pontfelhőből kinyert gyalogos objektum*

*4.8 ábra pontfelhőből kinyert emberek objektuma*

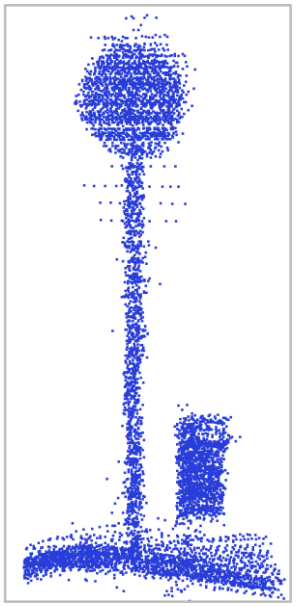
Az (4.8) ábrán bemutatott két ember szintén közel volt egymáshoz így egy objektumként érzékelte őket az algoritmus.



*4.9 ábra pontfelhőből kinyert fa*



*4.10 ábra pontfelhőből kinyert jelzőtábla*



*4.11 ábra pontfelhőből kinyert jelzőtábla és kuka*

Az (4.11) ábra szintén egy problémás esetet szemléltet: bár itt a magasságkülönbség nagyobb, mint 50cm (a tábla teteje 3m magas a mérés alapján), de a két vékony objektum egy cellába esik a rácson. Továbbá megfigyelhető a talaj egyenetlensége is.

Az objektumokat összeillesztő algoritmus lefutása után lehetőségünk van az egyes objektumokon dolgozni, ami az objektum klasszifikáció szempontjából nélkülözhetetlen. Ha pedig csak az objektum egy részén szeretnénk dolgozni, arra is van lehetőségünk, hiszen minden objektumnál nyilvántartjuk, hogy melyik cellákból tevődik össze, így dolgozhatunk az objektum egy részfelhőjén, ha arra van szükségünk.

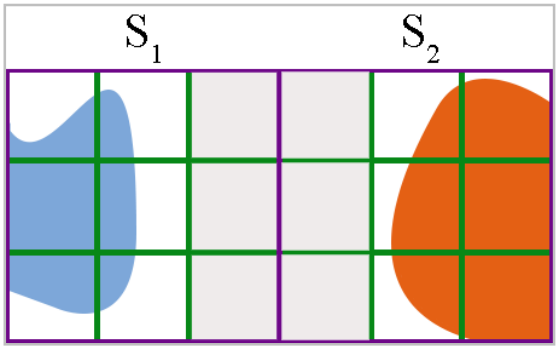
# 5. Többrétegű rács

Az (4.3) fejezetben bemutattam, hogy a háromdimenziós objektumokat előállító algoritmus nem tökéletes. Egyrészt az (4.6) és (4.8) ábrán látszik, hogy ha két objektum túl közel van, akkor nem tudja megfelelően szétválasztani azokat. Másrészt a legtöbb esetben az objektumok alatt nagy mennyiségű talajpont található. Ez a két hiba nagyban befolyásolja a későbbi objektum klasszifikációt.

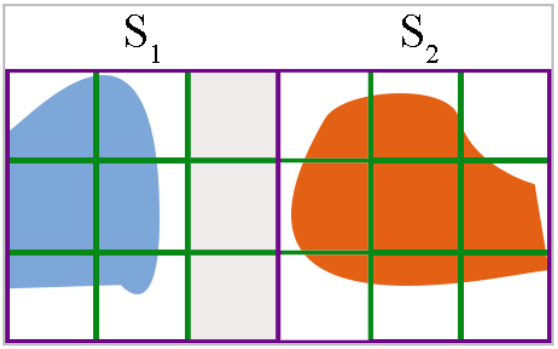
A hibás objektum szeparálás kiküszöbölése végett, az előzőekben bemutatott rács modellt fejlesztettem tovább. Implementáltam egy többrétegű rács struktúrát, ami lehetővé teszi, hogy különböző felbontású cellákkal alkalmazzuk azt detektációs feladatokra. Az első felbontási szint a már ismertetett 2D rács mérete vagyis (45-80) centiméter, a második felbontási szint pedig az első szint harmada. A mérések során az első szintet 60 centiméteres felbontással használtam, tehát a második felbontási szint 20cm volt.
A többrétegű rács lehetővé teszi újabb felbontási szintek bevezetését is, de úgy állapítottam meg, hogy a pontsűrűség egy időkeret pontfelhőnél nem elég nagy ahhoz, hogy a 20 centiméteres tartomány alá menjünk.

Az összefüggő komponensek keresése során, a rács második felbontási szintjét csak arra használtam, hogy közeli objektumok esetén a hibás többszörös detekciók számát csökkentsem. Két objektumot, akkor tudunk szeparálni, ha a távolság nagyobb köztük a második szint cellaméreténél (20cm). Ha a két objektum 20cm-nél közelebb helyezkedik egymáshoz, akkor továbbra is egy objektumnak fogja az algoritmus tekinteni őket.
Ahogy az (4.3) ábra szemlélteti, kétrétegű rács esetén is a rács első rétegében történik az objektumok összeillesztése, tehát ha az első réteg egy cellájába (60cm) több objektum is esik, akkor azokat egy objektumként kezeli az algoritmus.
A következőekben bemutatott esetek szemléltetik azokat a szituációkat, amikor a második réteggel lehetőségünk van szeparálni két objektumot.

## 5.1 Szeparálás többrétegű rács segítségével

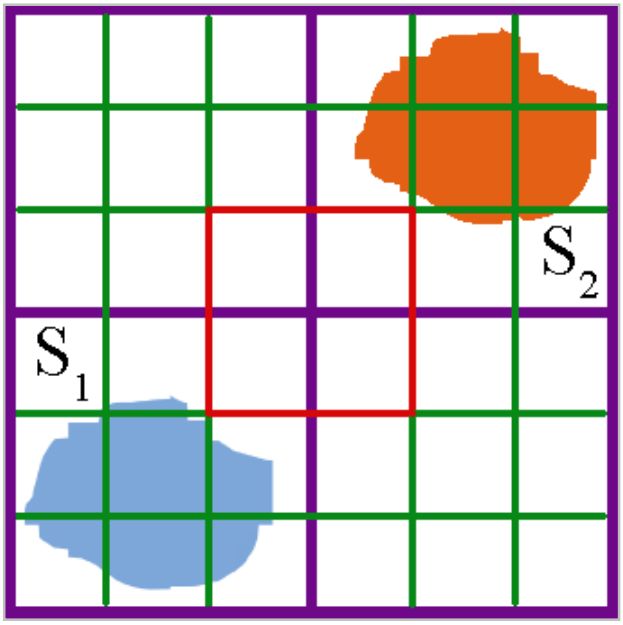



*5.1 ábra Objektumok szeparálása első eset*



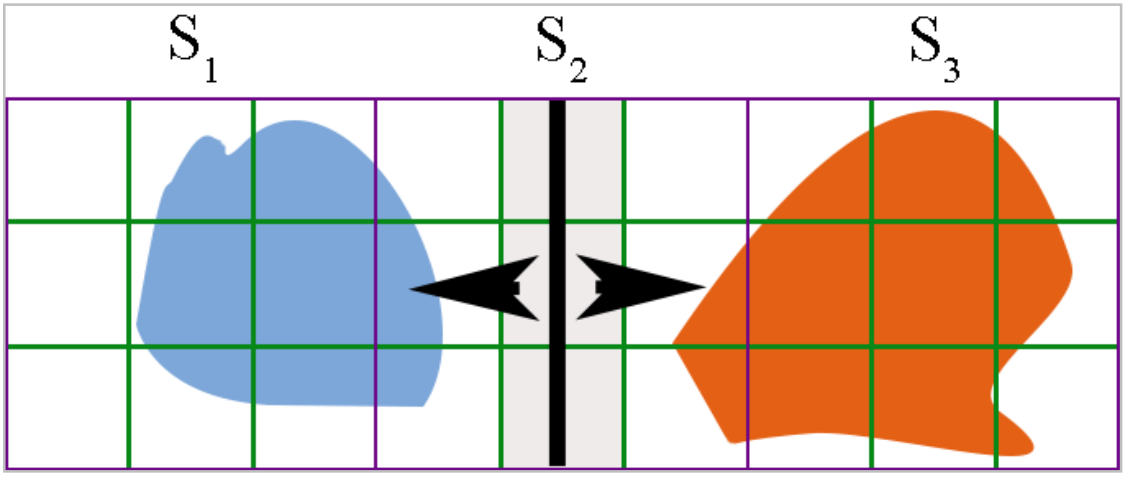

*5.2 ábra Objektumok szeparálása második eset*

Az (5.1) és (5.2) ábrákon bemutatott esetek nagyon hasonlóak. Mindkét ábrán $S_1$ és $S_2$ jelöli az első felbontási szintet. A második felbontási szintet az ábrákon a belső kis négyzetek jelölik.
Az első két eset azt szemlélteti, amikor a két objektum között nincs egy nagy felbontásnyi, azaz egy nagy cellányi hely. Az első esetben kettő, a második esetben pedig csak egy kis felbontásnyi, azaz kis cellányi hely van a két objektum között.




*5.3 ábra Objektumok szeparálása harmadik eset*

Az (5.3) ábra azt az esetet mutatja, amikor a két különálló objektum átlósan helyezkedik el egymáshoz képest a kétdimenziós rácsban. Ebben az esetben a szeparálásban a pirossal bekeretezett másodszintű cellák játszanak szerepet.



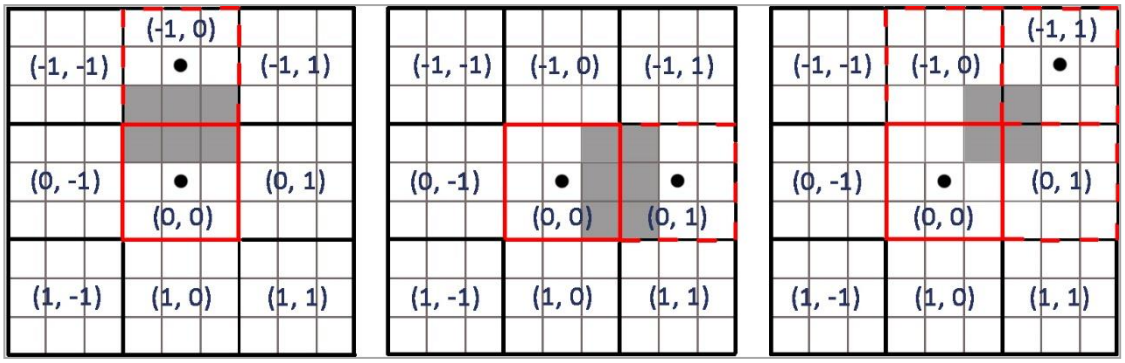

*5.4 ábra Objektumok szeparálása negyedik eset*

Az (5.4) ábra a negyedik esetet szemlélteti. Mind a két objektumnak van olyan része, ami egy közös cellába esik. Ebben az esetben az első szintű réteg egy celláján belül szeretnénk szeparálni, azonban itt az algoritmus csak vízszintes és függőleges szeparációt vesz figyelembe. A három vizsgált eset közül ez a legproblémásabb. Ugyan a két objektum között itt is van egy kis cellányi hely, de az $S_2$ cella mind a két objektumból tartalmaz részeket. Ahhoz, hogy külön objektumokra tudjuk őket bontani $S_2$ cellából a megfelelő részeket hozzá kell rendelni a megfelelő objektumhoz, vagy $S_1$ és $S_3$ objektumnak figyelmen kívül hagyjuk $S_2$-be tartozó részeit.

## 5.2 Szeparálási esetek meghatározása és vágás

Ahhoz, hogy a második szint felbontását használni tudjuk az objektumok szeparálására először is meg kell határozni, hogy a szomszédos nagy celláknak, melyik kis celláit vizsgáljuk a szeparáláshoz.

Az (5.5) ábrán a (0, 0) jelöli azt a cellát, ahol éppen az algoritmus tart. A (0, 0) cellához keressük a hozzátartozó cellákat. Az algoritmus nyolc szomszédságot vesz figyelembe az összefüggő komponenskeresés során. A (0, 0) cellához képest vizsgált első rétegben található cella pozíciója meghatározza, hogy a vágás során az adott első rétegbe tartozó cellák, melyik második rétegbe tartozó celláit kell vizsgálnia az algoritmusnak.




*5.5 ábra Szeparáláshoz szükséges második rétegbeli cellák meghatározása*

Az eseteket a következő függvénnyel határozhatjuk meg:

$$f(eset) = 2 * x_1 + 3 * x_2$$

1. $2 * (-1) + 3 * (0) = -2$
2. $2 * (0) + 3 * (1) = 3$
3. $2 * (1) + 3 * (0) = 2$
4. $2 * (0) + 3 * (-1) = -3$
5. $2 * (-1) + 3 * (-1) = -5$
6. $2 * (-1) + 3 * (1) = 1$
7. $2 * (1) + 3 * (-1) = -1$
8. $2 * (1) + 3 * (1) = 5$

Az esetek alapján meg tudjuk határozni, hogy melyik szomszédos cellára van szükségünk, és azt is, hogy melyik kis cellákat kell figyelembe vennünk a második rétegben, (utóbbi cellákat szürke szín jelöli az (5.5) ábrán).

Az aktuális esetnek megfelelően a szomszédos cella sűrűbb felbontáshoz tartozó celláit vizsgálom a (0, 0) cella második rétegbeli celláihoz képest.

Az adott esetnek megfelelő második rétegbeli cellákon belül a következő feltételek teljesülését vizsgálom:

1. Az első rétegben különböző cellákban, de egymás mellett elhelyezkedő második rétegbeli kiscellákban összesítve egy küszöbértéknek (S) megfelelő számú háromdimenziós pontnak kell lennie.
2. Az egymás mellett elhelyezkedő kiscellákban található pontszámok aránya (kevesebb pontot tartalmazó cella pontjainak száma osztva több pontot tartalmazó cella pontjainak száma) nagyobb kell legyen egy küszöbértéknél (T).

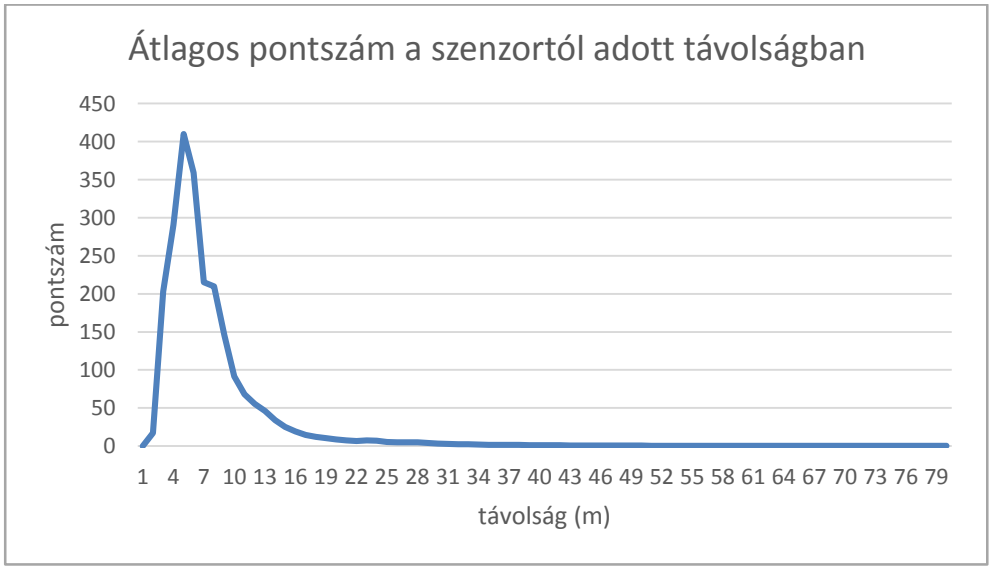
A (3.2) fejezetben megállapítottam egy küszöböt (Q), ami megadta, hogy egy első rétegbeli cellának legalább mennyi pontot kell tartalmaznia. Egy elsőrétegű cella 9 második rétegű

cellára oszlik, ezért a második rétegben legalább 2-3 db pontot kell tartalmaznia egy cellának. Az első feltételnél használt küszöbértéket (S), ezért 5-re választottam.
A második feltétel bevezetését egy egyszerű példán keresztül mutatom be. Legyen A és B két egymás melletti kiscella. A cella 5 pontot tartalmaz B cella pedig 100 pontot. Ebben az esetben a két cella aránya 0,05. Az első feltétel ugyan teljesül, de a második feltétel már arra enged következtetni, hogy szakadás van a két cella között. A második feltételnél használt küszöbértéket 0,25-nek választottam.

A fenti két feltétel együttes teljesülése során a két kiscellát összetartozónak tekinti az algoritmus és a megfelelő két első rétegbeli cellát egy objektummá alakítja, ellenkező esetben szakadás van a két első rétegbeli cella között, így külön objektumhoz fognak tartozni.

## 5.3 Objektumok szeparálása súlyozott pontszám alapján

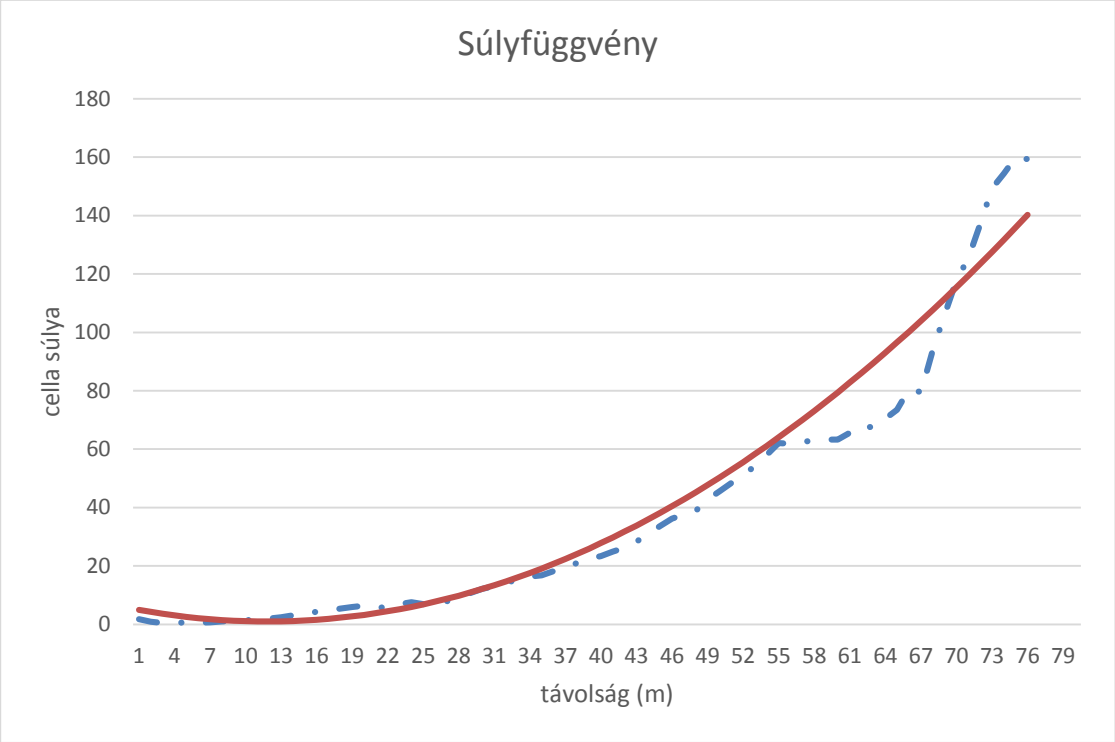
A Velodyne HDL-64E szenzorral rögzített pontfelhők jellegzetes karakterisztikát mutatnak. A szenzortól távolodva a pontsűrűség nagymértékben csökken.




*5.6 ábra A szenzortól adott távolságban található átlagos pontszám normálva a terület méretével*

Az (5.6) ábrán látható mérés közel 2000 pontfelhő átlagolásával készült. A pontfelhők három különböző jellegű városi környezetből származnak. A mérésből látszik, hogy a szenzor nagyjából 20 - 30 méteres környezetében a pontfelhő még elég sűrű, hogy összefüggő komponenseket keressünk, majd osztályozzuk azokat. 30 méter után sok esetben már egy autó méretű objektum sem mindig tartalmaz elég pontot ahhoz, hogy szeparálni lehessen.

A probléma megoldására az (5.6) mérés eredményét felhasználva kiszámoltam egy súlyozó függvényt, ami a cellák pontszámmát súlyozza fel a távolság függvényében.




*5.7 ábra Cellák súlyozása a távolság függvényében*

Az (5.7) ábrán a szagatott vonallal jelölt görbe reprezentálja az eredeti súlyfüggvényt, de az algoritmus a folytonos vonallal jelölt illesztett súlyfüggvényt használja. Elméletben azt várjuk, hogy minél több adathalmaz szolgál a mérés alapjául a görbe annál inkább megközelíti az illesztett görbét.

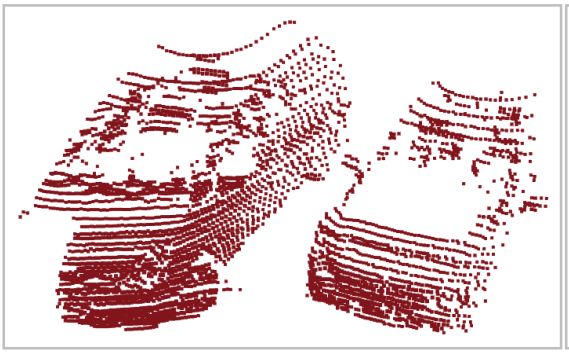
A súlyfüggvénnyel felszorozzuk az adott cellák pontszámát és ezt használjuk a komponenskeresés és vágás során. Ennek eredményeként 50-60 méter távolságban is sokkal jobban teljesít az algoritmus és így növelni tudtam az algoritmus detektálási távolságát.

## 5.4 Összefüggő komponenskeresés és az előszegmentálás

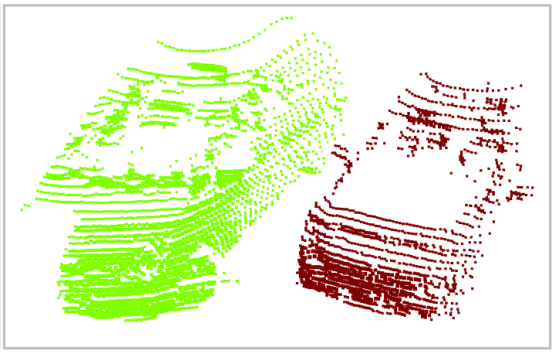
A (3.3) fejezetben láttuk, hogy az első lépésben az algoritmus négy fő csoportba (alacsony utcai objektumok, magas objektumok, talaj, ritka) sorolja be a cellákat, a bennük található pontok maximum magassága szerint. Ezt az előszegmentációt az összetett komponensek (objektumuk) keresése során felhasználja az algoritmus, ahogy azt a (4.2) fejezet láthattuk.

Az összefüggő komponenskeresést így külön futtathatjuk a magas objektumokra, alacsony objektumokra, de akár az elő szegmentációt figyelmen kívül hagyva az összes cellára is futtathatjuk az algoritmust. Ha figyelembe veszi az algoritmus az előszegmentációt, akkor gyorsulást érhetünk el a futási időben, és az alacsony objektumú cellákat nem ragasztja össze a magas objektumú cellákkal.

## 5.5 Szeparálási eredmények



*5.8 ábra Objektum kinyerése egyrétegű ráccsal*

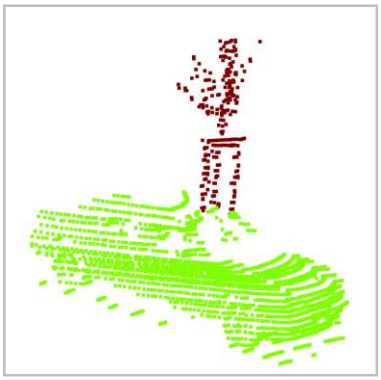


*5.9 ábra Objektum kinyerése többrétegű ráccsal*

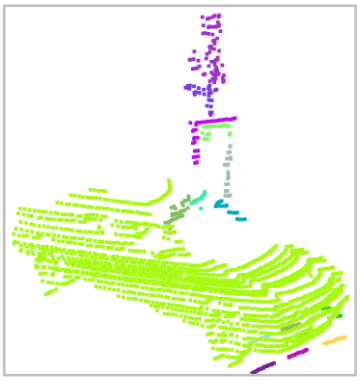
Az (5.8) és (5.9) ábrák szemléltetik az egyrétegű rács és a kétrétegű (multi rács) által kapott eredményeket. A többrétegű rács esetében az objektumot két különálló objektummá tudtuk bontani így teljesen külön tudjuk őket kezelni. A két objektum szétválasztása csak akkor lehetséges, ha legalább egy második rétegbeli cellaméretnyi hely van közöttük.

A második rétegben történő vágás arra szolgál, hogy még mielőtt a két első rétegben lévő cellát egy objektummá ragasztaná az algoritmus, megvizsgálja a vágási feltételeket és ennek megfelelően dönt az cellák összeillesztéséről vagy szeparálásáról.

*5.10 ábra Objektum kinyerése egyrétegű ráccsal*

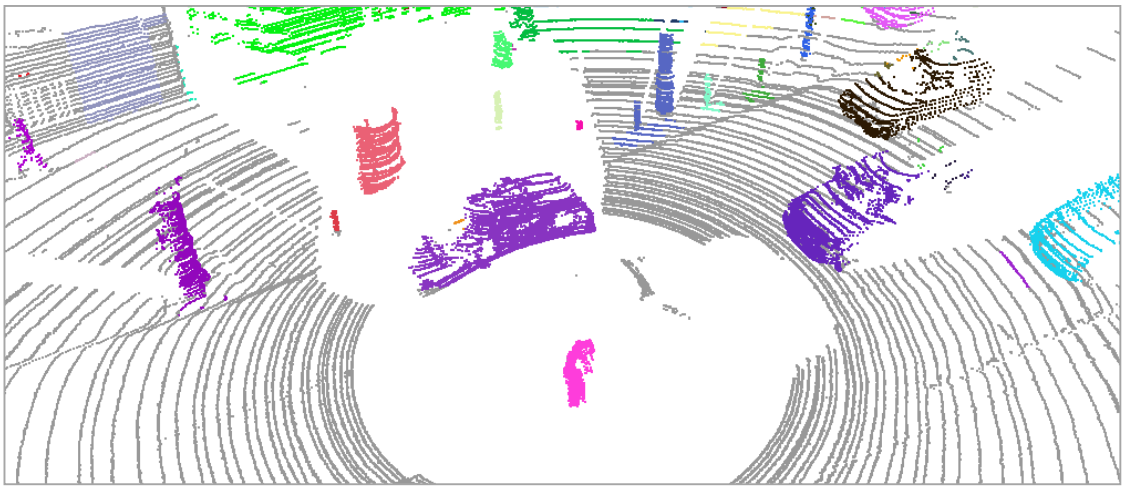


*5.11 ábra Objektum kinyerése többrétegű ráccsal*



*5.12 ábra Objektum kinyerése Kd-fa alapú eljárással*

Az (5.10) ábrán az egyrétegű rács csak egy cella mérettel dolgozik (60cm) és a két közeli objektumot nem tudja hatékonyan szétválasztani. Az (5.12) ábrán láthatjuk, hogy a kd-fa alapú eljárás képes volt szeparálni a két objektumot, de több különböző részre vágta szét, az eredetileg egy objektumhoz tartozó pontokat. A kd-fa alapú objektumdetektáló eljárások optimális paraméter beállításai nagyban függenek a detektálandó objektumok geometriai adottságaitól. Bizonyos alakú objektumokra könnyen lehet optimálisan paraméterezni ezeket az eljárásokat, de nehéz olyan paraméter beállítást adni, ami geometria függetlenül, eltérő alakú objektumokra is jó eredményeket szolgáltat (pl.: járművekre, járókelőkre, egyéb utcai objektumokra egységesen). Az (5.11) ábra a többrétegű ráccsal való szeparálást szemlélteti. A két objektum elkülönítése után a következő lépés az objektumok klasszifikálása.



*5.13 ábra A kétrétegű ráccsal kinyert objektumok halmaza egy időkeret pontfelhőn*

Az 5.13-as ábra szemlélteti a kétrétegű ráccsal kinyert objektumhalmazt egy adott időkereten belül. Mivel az algoritmus általánosságban összefüggő komponenseket keres, a járművek mellett a kinyert halmazban oszlopok, emberek, falszegmensek és egyéb utcai objektumok is megtalálhatóak, ahogyan ezt a fenti ábra is szemlélteti.

# 6. Objektumok klasszifikálása

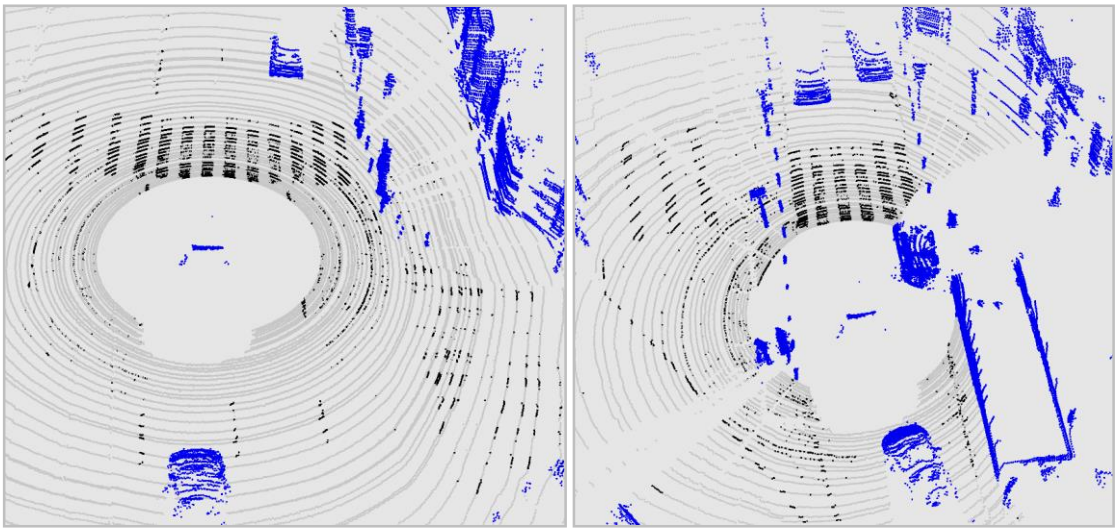
Az 5. fejezetben bemutatott többrétegű rács adatstruktúra felhasználásával objektumokat nyerhetünk ki egy adott pontfelhőből és láttuk, hogy ezek a legkülönfélébb utcai objektumok lehetnek. A 6. fejezetben lehetőségeket mutatok néhány kinyert objektumtípus osztályozására, felismerésére (autó, jelzőtábla) és gyalogos átkelőhelyek detektálására.

## 6.1 Gyalogos átkelőhely detektálása

Egy mobil robot rendszernél vagy vezetést segítő rendszerekben az egyes objektumok felismerése (autó, gyalogos, tábla) mellett fontos a gyalogos átkelőhelyek detektálása is. Az átkelő helyet idejében észre kell venni és lassítani, vagy megállni előtte.
Az útburkolati jelek detektálása nem csak az mobil robot rendszereknél lehetnek hasznosak. A felfestések detektálása után elemezhetjük a környezetet, hogy az adott útburkolati jel a megfelelő helyen van-e, vagy átfogó képet kaphatunk a felfestés állapotáról is. Ha egy felfestés állapota már nem megfelelő, akkor annak újrafestése szüksége és a festék mennyiségét a jel kiterjedéséből tudjuk

### 6.1.1 Útburkolat intenzitásának vizsgálata

A szabályos egyrétegű ráccsal történő szegmentáció után a cellákat négy csoportba soroltuk be. Az egyik csoport a talaj volt. Ezt kihasználva már csak a talaj címkéjű cellákon kell tovább vizsgálódni, hiszen az útburkolati felfestések itt találhatóak.
A munka során az útburkolati jelek közül a gyalogos átkelőhely detektálásával foglalkoztam. A gyalogátkelőhelyek észleléséhez kihasználtam, hogy az útfestéseknek az úttesttől eltérő anyagáról nagyobb intenzitással verődik vissza a lézerfény, és az átkelők jellegzetes mérettel valamint alaki paraméterekkel rendelkeznek.
Az intenzitást a szenzor rendeli hozzá az adott ponthoz, és ez az érték egy 0 és 1 közötti lebegőpontos számként fog reprezentálódni a programban, ahol az 1 a maximális intenzitás. Az intenzitás nagysága függ az anyag fényvisszaverő képességétől és a lézerfény beesési szögétől is, ezért ugyanannak a pontnak az intenzitására más-más távolságból különböző értékeket fogunk kapni.
Több mérési eredményt figyelembe véve megállapítottam egy 0 és 1 közötti küszöbértéket. Amely pont intenzitásának értéke ezen küszöbérték felett van, azt felfestésnek tekintem. A megállapított küszöbérték: 0.55.

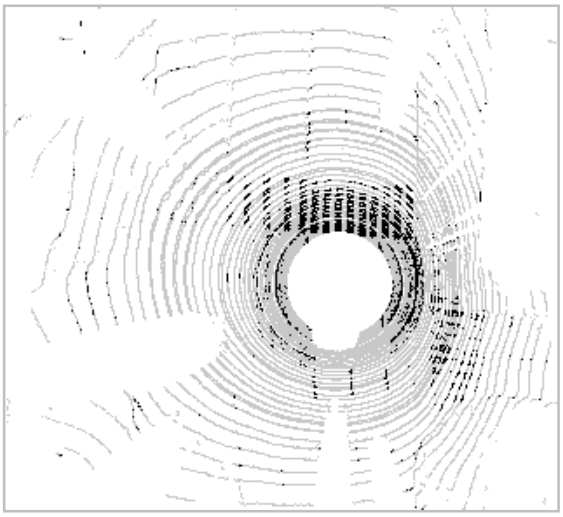


*6.1 ábra Gyalogos átkelőhely keresése intenzitás alapján*

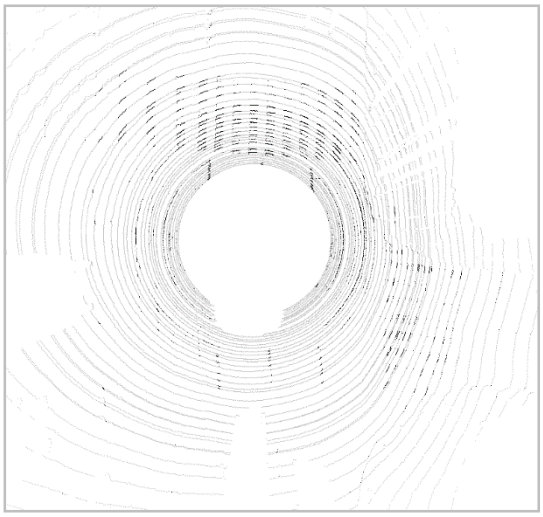
A (6.1) ábra gyalogos átkelőhelyek keresését reprezentálja. Az ábrákon megfigyelhető a szenzor működéséből adódó körkörös szkennelés és az, hogy a szenzortól távolodva egyre kevesebb pontot tartalmaz a pontfelhő. A felfestések detektálását a szenzortól körülbelül 40 méter távolságban érdemes elkezdeni, mert ettől távolabb kevés pont áll rendelkezésünkre és a lézerfény beesési szöge miatt a felfestések intenzitása is nagyon alacsony.

### 6.1.2 A pontfelhő projektálása a talaj síkjára

A pontfelhő átlagosan 60000 - 100000 pontot tartalmaz időkeretenként. A projekcióval a háromdimenziós ponthalmazt egy kétdimenziós képre vetítettem, így lehetőségem nyílik, a hagyományos kétdimenziós pixelrácson működő képfeldolgozási algoritmusok használatára.
Lehetőség van állítani, a kétdimenziós kép méretét, ezáltal befolyásolni azt, hogy mennyire legyen pontos a projekció. Ha kis értéket választunk, akkor sok háromdimenziós pont fog ugyanarra a kétdimenziós pixelre vetülni, ez nem jó, mert információt vesztünk, de ha túl nagyra választjuk az értéket, akkor meg annyira ritka lesz a kétdimenziós kép, hogy lehetetlen lesz rajta morfológiai eljárásokat végezni.
A projekciónál csak a talaj pontokat projektálom le. A projektálásnál a felhő pontjait el kell tolni, hiszen a pontfelhő lokális koordinátarendszerének a közepe a szenzor, míg a képen ez a bal felső sarok lesz. A később detektált eredményt a transzformáció inverzével vissza tudjuk helyezni a pontfelhőbe, ezáltal megkapva a keresett felfestést.



*6.2 ábra Sűrűn projektált pontfelhő*



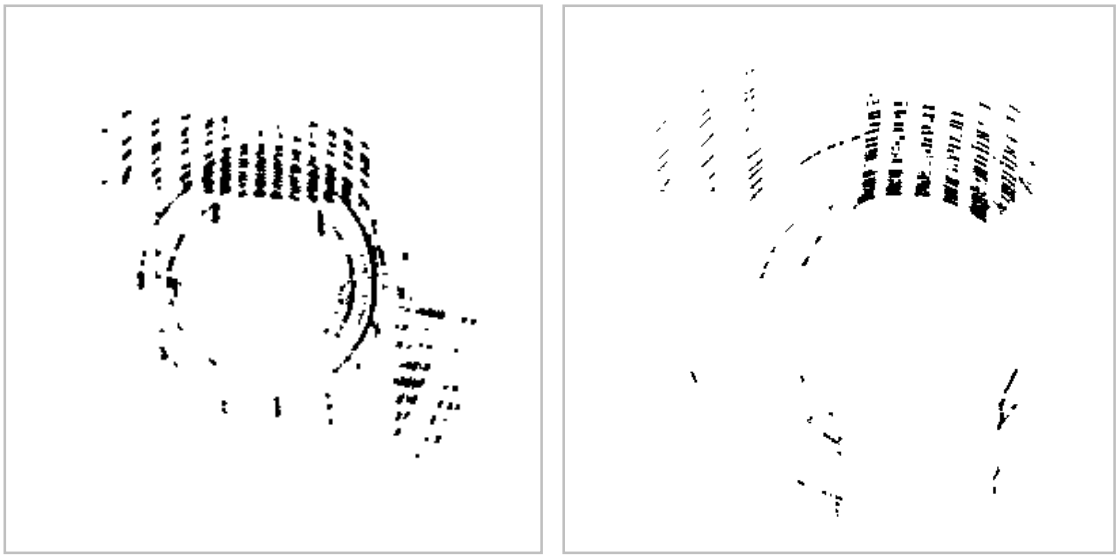
*6.3 ábra Ritkán projektált pontfelhő*

A (6.2) és (6.3) ábrán is kivehetőek a zebrák, de a (6.3) ábra már elég ritka, ami megnehezíti a későbbi morfológiai eljárásokat.

### 6.1.3 Projekció javítása morfológiával

A projekció által kapott képeken dilatációt hajtottam végre. Három különböző színű pixel látható a projektált képen (6.2). A fekete, ami az általam definiált útjelzés osztály, a szürke, ami az úttest osztály és a fehér, ahol nem volt adat a pontfelhőben. Több kernel mérettel is próbálkoztam. Minél nagyobb felbontású a kép, annál nagyobb kernelt próbáltam használni, hogy az üres térrészek növekedése mellett is legyen eredménye a dilatációnak.

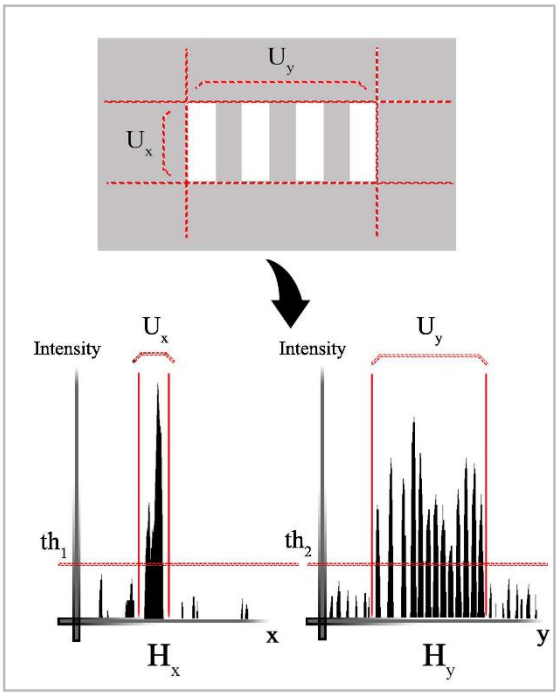
A nagy intenzitású (6.2 ábrán fekete pontok) mellett azért projektáltam le a talaj (6.2 ábrán szürke) pontokat is, mert a dilatáció során figyelembe vettem őket a zajszűréshez. Csak a nagy intenzitású (fekete) pontokon hajtottam végre dilatációt, de a kernelen belül a talajpontokat is figyelembe vettem -0.5 súllyal. Ebből adódóan az olyan nagy intenzitású pontok, melyek környezetében 4-5 talajpont található és mellette kevés másik nagy intenzitású pont nem növekedtek annyira a dilatáció során.



*6.4 ábra Projektált pontfelhőn végrehajtott dilatáció*

Megfigyelhető, hogy előnyösebb inkább pár pontot feláldozni és kisebb méretű képre projektálni (6.4 ábra), mert így a morfológiai eljárások pontosabbak lesznek és jobb eredményt kapunk. Az egyetlen hátránya az a kisebb felbontású képeknek, hogy a zaj is nagyobb lesz a képen, mert a dilatáció azt is fel fogja erősíteni. Mivel a zaj nagy része közvetlenül az autó körül található, a szenzor tulajdonságaiból adódóan, ezért a detekciót csak az autótól egy bizonyos távolságra érdemes elkezdeni.

### 6.1.4 Hisztogram alapú detekció

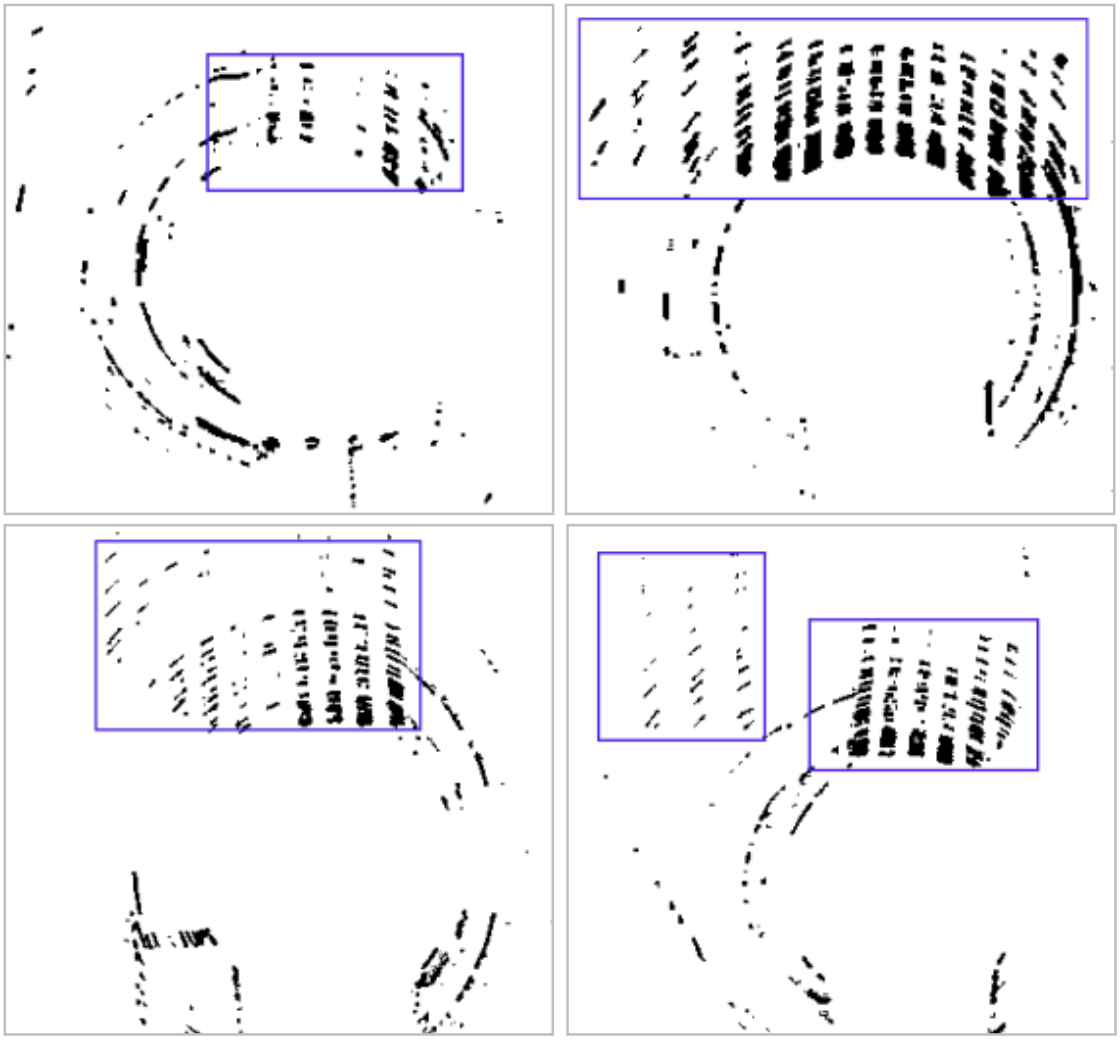



*6.5 ábra Felfestések detektálása kétirányú hisztogrammal*

A dilatált képen a nagyobb intenzitású területek (gyalogos átkelőhely) detektálásához egy x és egy y irányú hisztogramot készítettem a képről. Először soronként számoltam össze a fekete pixelek számát, majd ezt a műveletet elvégeztem oszloponként is. A (6.5) ábrán, a két hisztogramon jól látszik az a térrész, ahol a fekete pixelek a legnagyobb számban szerepeltek. A küszöbértékekkel lehet szabályozni, hogy mennyire legyen érzékeny az algoritmus. A két hisztogram metszete meghatározza azt a térrészt, ahol a keresett felfestés található.

A módszer segítségével akár több gyalogos átkelőhelyet, vagy felfestést is meg lehet találni. Elkezdjük bejárni a hisztogramot, és ha átlépjük a küszöbérték határát, akkor elkezdünk számolni addig, amíg újra át nem lépjük a küszöbértéket. Így több felfestést is detektálni tudunk. Azt hogy mekkora kiterjedésű felfestésre vagyunk kíváncsiak a küszöbértékkel tudjuk szabályozni.

### 6.1.5 Eredmények



*6.6 ábra Gyalogos átkelőhelyek detektálása hisztogram segítségével*

A gyalogos átkelőhelyeket mind a négy fenti ábrán megtalálta az algoritmus. A (6.6/jobb alsó) ábrán két zebra látható egymáshoz kissé eltolva. A bal felső zebra sokkal kisebb intenzitás választ adott vissza egyrészt a beesési szög miatt, másrészt valószínűleg a felfestése is meg volt már kopva.

A felfestések pontos detektálásához és főleg a felismerésükhöz szükséges egyéb szenzorok, mint például RGB kamerák információjára is támaszkodni. A Velodyne szenzor intenzitás

adataira támaszkodva viszont meghatározhatjuk a nagyobb intenzitású területeket. Ezeknek a területeknek az ismeretében az RGB kamerák adatain már tudjuk, hol kell keresni a számunkra érdekes területeket így könnyebben felismerhetjük azokat.

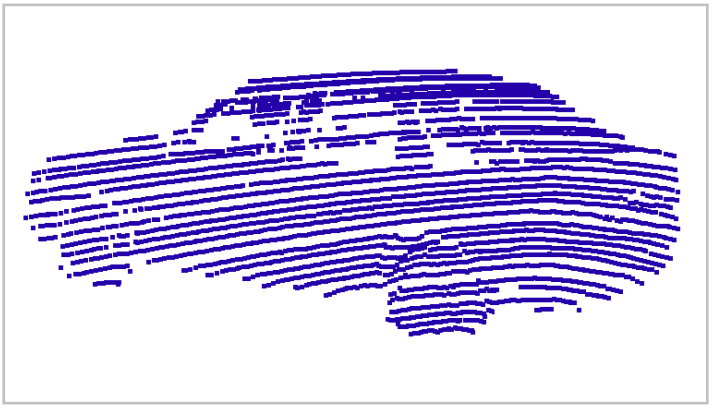
## 6.2 Objektumok osztályzása, felismerése

A kinyert objektumokat tulajdonságaik alapján kategóriákba szeretnénk sorolni. Ezek a tulajdonságok lehetnek geometriai tulajdonságok (az objektumokra jellemző méret, pontszám), intenzitás alapú tulajdonságok, és akár a térbeli elhelyezkedést is ide sorolhatjuk.
Jelen dolgozatban csak a Velodyne HDL-64E szenzor adataira támaszkodtam, mind az objektumok kinyerése és mind a felismerésük során.

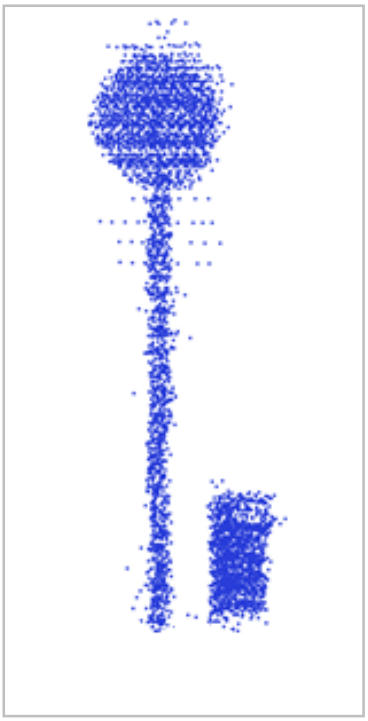
### 6.2.1 Objektumok alatti talajpontok szűrése

Az objektumok klasszifikálásának megkezdése előtt, meg kell próbálni az objektumok alatti talajpontokat kiszűrni. Ez azért fontos, mert a klasszifikáció során ezek a pontok eltorzítják az eredményt.
Első közelítésben jó ötletnek tűnik, hogy meghatározzuk a talaj síkját és az objektumok azon pontjait melyek ez a sík alatt vannak, levágjuk. Ezzel az elgondolással az a probléma, hogy a talaj nem vízszintes. Ezért a Velodyne szenzor közelében máshol lesz a talaj síkja, mint tőle távolabb és ez már 40-50 méteren belül is nagy eltérést okozhat.
Ezért ahelyett, hogy megállapítanánk egy globális talajszintet, minden cellához kiszámolunk egy lokális talajszintet. A (3.3) fejezetben négy osztályba soroltuk a cellákat, ahol az egyik osztály a talaj volt. Végigmegyünk a cellákon és azok a cellák, melyek nem talaj címkét kaptak, de van talaj szomszédjuk, megkapják a talaj szomszédjaik átlagmagasságát saját lokális magasságként. Azok a cellák, amik belső cellái egy objektumnak és nincsenek talaj szomszédjaik, megkapják a külső szomszédjaik talajmagasságát. Ezen módszer után már ki tudjuk szűrni az objektumok alól a talaj nagy részét.

A talaj szűrése így sem mindig tökéletes, de az esetek nagy részében elfogadható eredményt ad és sokkal jobb eredményeket garantál, mint a globális talajvágás. A (6.7) és (6.8) ábrákon a (4.5) és (4.11) ábrákon látható objektumok alatti talajszűrés eredménye figyelhető meg.



*6.7 ábra Talajpontok szűrése járművek esetén*



*6.8 ábra Talajpontok szűrése utcai objektumok esetén*

### 6.2.2 Kinyert objektum pontos befoglaló téglalapjának meghatározása

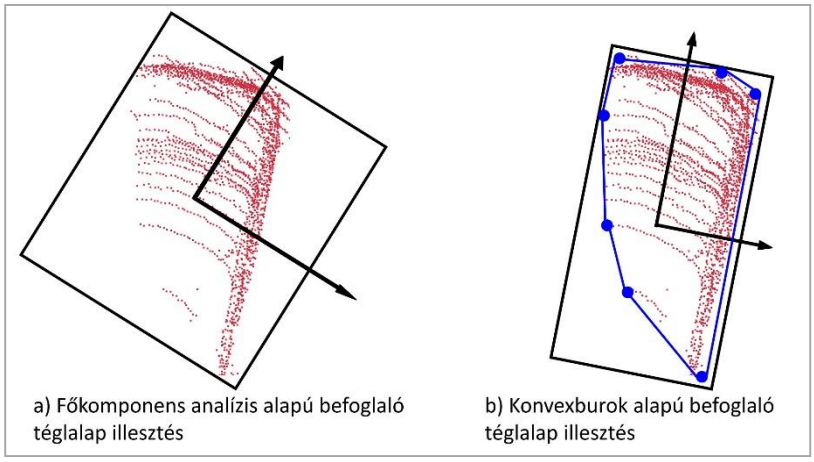
Az objektumok jellemzőinek hatékony kinyerése érdekében széleskörben alkalmazott előfeldolgozási módszer egy két vagy három dimenziós befoglaló téglatest (BT) illesztése. A szakirodalomban általánosan kétféle stratégia jellemző a BT kiszámítására. Az első módszer során, [18] az objektum alakjából és méreteiből épül fel a BT. Ez a módszer olyan esetekben jó megoldás, ha kevés objektumtípus található a teszt halmazban. Esetünkben viszont a kinyert objektumok típus és alaki diverzitása nagyon nagy, hiszen kinyerésük valós városi környezetből történik. Ezért az objektumok halmazában többek között találunk fákat, oszlopokat, töredezett fal szegmenseket, járókellőket, járműveket és egyéb utcai objektumokat is.

Másik megközelítés a BT felépítésére a főkomponens analízis (PCA). A [6] és [19] cikkekben az objektumok pontfelhőinek kovariancia mátrixa által kiszámított legnagyobb sajátértékhez tartozó sajátvektor fogja kijelölni az objektum fő terjedési irányát. A három kiszámolt irányból fog a BT létrejönni.

A BT felhasználásával történő jellemzők kinyerése mellett számos módszer található még a szakirodalomban. Ezek többek között az objektumok alaki jellemzőire és az objektumok kontextusaira támaszkodnak [20], [21], [22]. A [20] módszer gráf alapú vágást használ a pontok elő- és háttér szegmentálásához. A [21] algoritmus klasszifikálása olyan 3D jellemzőkön

alapszik, mint a spin image vagy a gömbi harmonikus leírók. A [22] algoritmus utcai objektumok detektálását és klasszifikálását végzi, gráf alapú klaszterezési algoritmussal és különböző alaki jellemzők felhasználásával, melyeket spin image-ből vagy PCA-n alapulós sajátvektorokból nyer ki. Bár ezek az általános alaki és kontextuális alapú módszerek precízebb felismerési arányt tesznek lehetővé városi objektumok klasszifikálásában, mint a BT alapú módszerek, azonban a számítási költségük sokkal nagyobb és legtöbbször nem is valós idejű.

A munkám során használt pontfelhőkben a Lidar szenzor karakterisztikája, a távolságfüggő pontsűrűség és a takarások miatt az utcai objektumokból kinyert alakzatok sok esetben hiányosak vagy torzak. Így a hagyományos geometriai vagy PCA alapú BT illesztés sok esetben nem ad jó eredményt. A 6.9 ábrán bemutatott esetben felülnézetből jelenítjük meg egy autó mért pontfelhőjét. Látható, hogy az elvárt téglalap alapú sziluett helyett az autónak csupán a felvételhez legközelebb eső sarka jelenik meg, a pontfelhőre számolt főkomponens irányok így jelentősen eltérnek az objektum valódi tengelyirányaitól




*6.9 ábra PCA vs. Konvexburok alapú befoglaló téglalap illesztés*

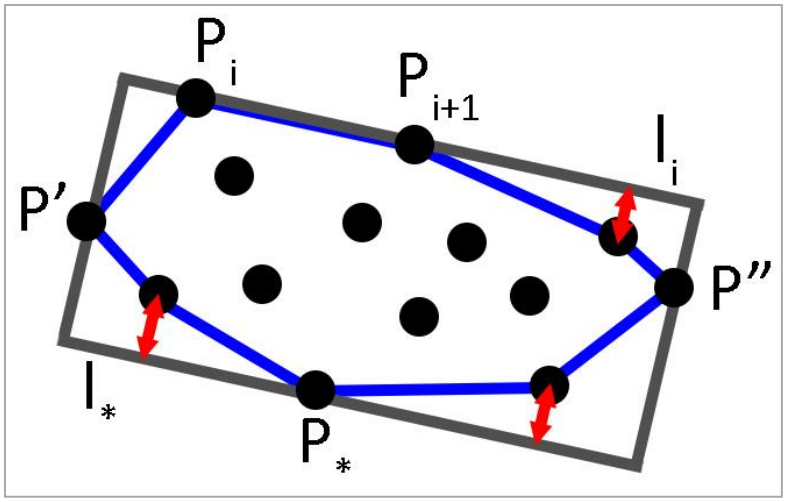
A PCA alapon meghatározott befoglaló téglalap alapján, ekkor rosszul tudnánk meghatározni az objektum méreteit, orientációját és az alaki tulajdonságait.

A helytelen BT illesztés kiküszöbölésére a munkám során egy konvex burok általi BT illesztést használtam fel, amely kutatólaboratóriumomban került kifejlesztésre [3].

**A konvex burok általi BT illesztésének algoritmusa:**

A vizsgált objektum elhelyezkedését ismerjük a már felépített hierarchikus kétrétegű rács modellben. Az objektum pontjainak csak az x és z koordinátáit használjuk, tehát a magasságot nem vesszük figyelembe, ezzel is gyorsítva a módszert. Az algoritmus első lépésben meghatározza, hogy melyek azok a cellák, amik tartalmaznak pontokat, és melyek azok, amik üresek. Ezután minden pontot tartalmazó cellának megvizsgáljuk a 3x3-as szomszédságát és eliminájuk azokat a cellákat, melyeknek minden szomszédja tartalmaz pontot. Ezzel a módszerrel az algoritmus megbecsüli azokat a cellákat, amelyek az objektum határához tartoznak, lényegesen gyorsítva ezzel a későbbi számításokat. Az utolsó lépésében a monoton lánc algoritmus [23] segítségével az algoritmus megkonstruálja az objektum konvex burkát a határ cellák pontjainak felhasználásával.

A konvexburok kiszámítása után a következő algoritmus [3] illeszti az objektumra a megfelelő befoglaló téglalapot:




*6.10 ábra Befoglaló téglalap illesztése konvexburok alapján*

- Az algoritmus egymás után bejárja a burok folyamatos pontpárjait $p_i$ és $p_{i+1}$ ($i = 1, 2, \ldots i_{max}$):
  1. Meghatározza az $l_i$ egyenest a $p_i$ és $p_{i+1}$ pontok között, ami a befoglaló téglalap egy lehetséges oldala lesz.
  2. Megkeresi a konvex burok $p_*$ pontját, aminek maximális a távolsága az $l_i$ egyenestől és a $p_*$ ponton keresztül egy $l_i$-vel párhuzamos egyenest határoz $l_*$ meg. $l_*$ lesz a téglalap másik oldala.

3. Végül a konvexburok összes pontját leprojektálja az $l_i$ egyenesre és megkeresi a két külső pontot p’ és p’’. A téglalap másik két oldalát a p’ és p’’ pontokon átmenő és $l_i$-re merőleges egyenesek adják.

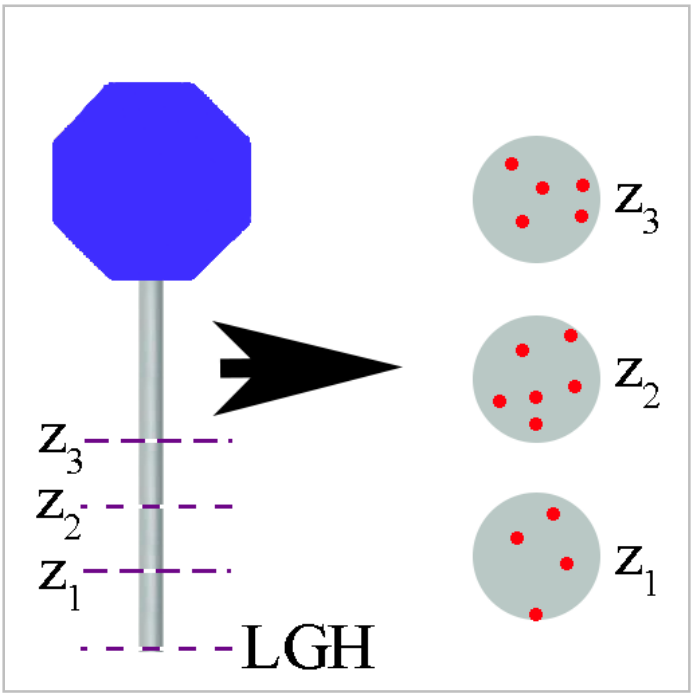
- Az algoritmus a javasolt befoglaló téglalapok közül az optimálisat választja, azaz azt, ahol a konvex burok pontjainak átlagos távolsága a befoglaló téglalaptól minimális.

Az algoritmus lefutása után az objektumok befoglaló téglalapjai sokkal pontosabban fognak illeszkedni az objektumok valós orientációjához, mint a hagyományos PCA alapú megoldások esetén (6.9 ábra). Így a jellemzők kinyerésével pontosabb képet kapunk az objektum hosszáról, szélességéről és egyéb paramétereiről.

### 6.2.3 Objektumok további geometriai szegmentációja

Az objektumkinyerés (4-5. fejezet) végrehajtása és az orientációhelyes befoglaló téglalap illesztése után lehetőség van további geometriai jellemzőket kinyerni az objektumokból. A magasság mellet az x irányú szélességet és az y irányú mélységet is meg lehet határozni. A gyalogos és a jelzőtábla objektumok egyrészt kevesebb pontot tartalmaznak, másrészt a térfogatuk is kisebb, mint egy jármű (autó, villamos) objektum. Geometriai alapokon egy gyalogost egy táblától vagy egy fától nehéz megkülönböztetni, de egy nagyobb objektumtól (pl.: autó) az esetek nagy százalékában jól elkülöníthető.
Munkám során a jelzőtáblák detektálására fejlesztettem még ki egy geometriai alapú detekciót, ami a jelzőtáblák oszlopvastagságát is figyelembe veszi.

A vizsgálataimat azokra a kinyert objektum-pontfelhőkre szűkítem, melyek magassága megfelel a táblákra vonatkozó prior feltételeknek. A 6.11 ábrán bemutatottak szerint az objektum magasságának a felétől az objektum aljáig felveszek három mérési pontot: z1, z2, z3, majd ezeknél a mérési pontoknál az objektumból kimetszek három síkszeletet. Az egyes síkszeletekbe tartozó pontok közül megkeresem az x illetve y irányoknak megfelelő maximális és minimális koordinátájú pontokat, majd kiszámolom az objektum becsült (x-y) irányú kiterjedését.

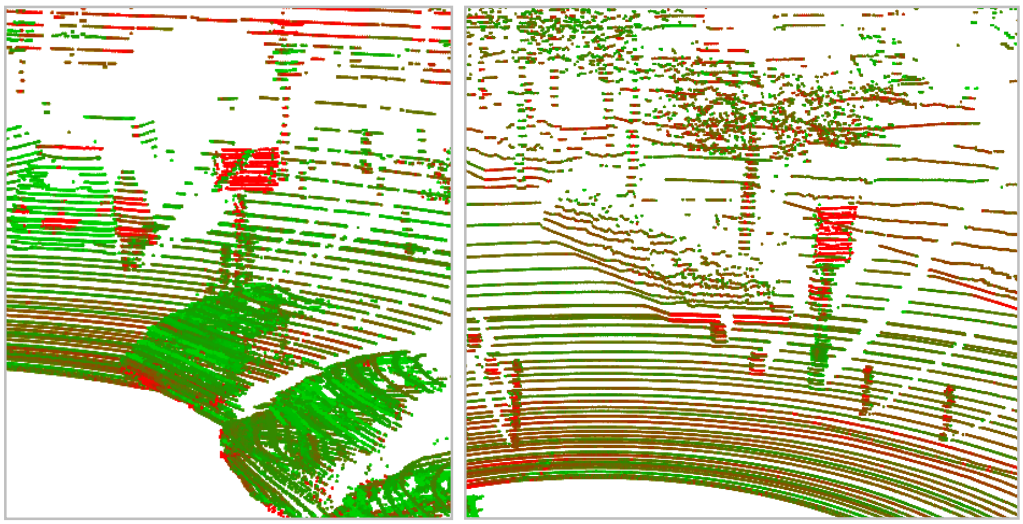



*6.11 ábra Jelzőtábla detekciója oszlopvastagság alapján*

$$átmérő_x = Pont_{\max\_x} - Pont_{\min\_x} \quad ÉS \quad átmérő_y = Pont_{\max\_y} - Pont_{\min\_y}$$

Az x és y-irányú kiterjedésből meg tudjuk határozni az adott objektum vastagságát. Ha a három mérési ponton az eredmények közel azonosak, akkor az objektum alsó része egy oszlopszerű forma. Azonban ez még így önmagában kevés, mert akár oszlop, fa, vagy ember is lehet, ezért további mérésekre van szükség.

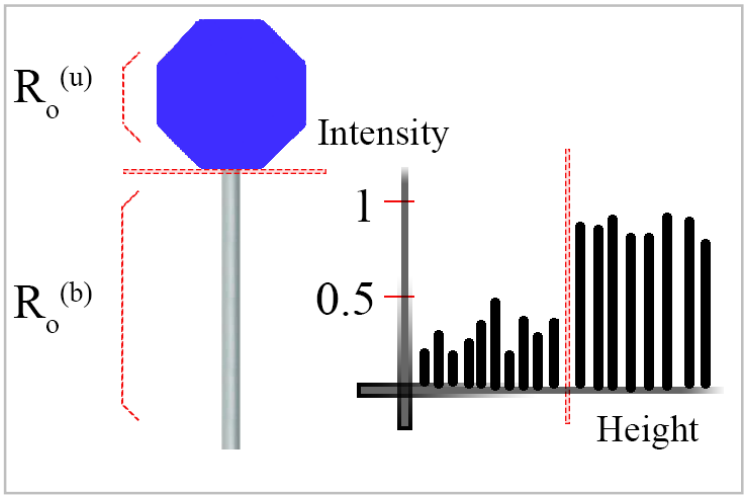
### 6.2.4 Intenzitás alapú szegmentálás



*6.12 ábra Pontfelhő intenzitásképe*

A jelzőtáblákat el kell különíteni a fáktól, oszlopoktól esetleg a gyalogosoktól is. Az általam vizsgált táblák átlagos magassága 2,5-4 méter, de a felvételeken a takarások miatt ez nem mindig igaz és ekkor magasság alapján már nem lehet egyértelműen megkülönböztetni egy gyalogostól.

A jelzőtáblák esetében a felső rész, azaz maga a tábla feje jó fényvisszaverő tulajdonságokkal rendelkezik. Ezt a tulajdonságot használjuk ki, hogy elkülönítsük őket a többi hasonló geometriájú objektumtól. A (6.12) ábrán jól látszik, hogy a tábla feje magas intenzitású pontokat tartalmaz. Ez az érték nulla és egy közötti intervallumon 0.75-1.0 közzé esik.

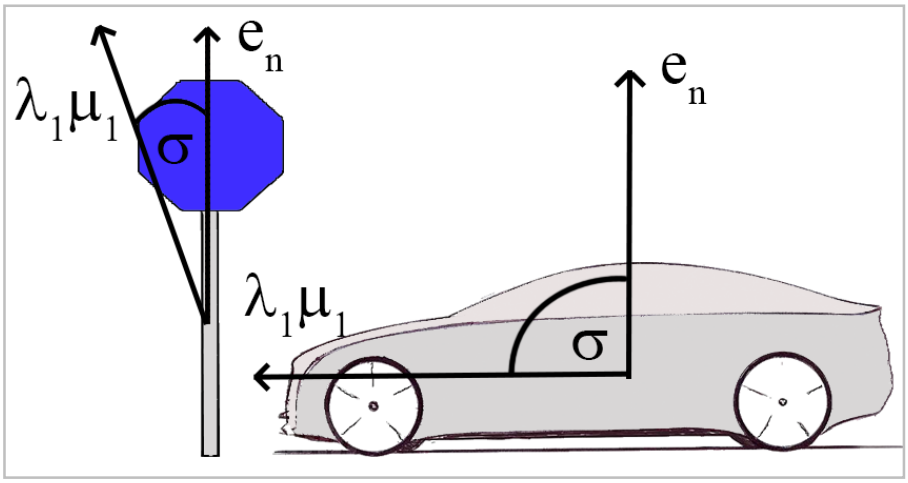



*6.13 ábra Jelzőtábla detektálása intenzitás alapján*

A (6.13) ábrán egy tábla magasság szerinti intenzitás hisztogramja látható. Egy magasságküszöb átlépése után az intenzitás megnő (itt található a tábla feje). Ez egyébként nagyjából a tábla alakját is kirajzolja a hisztogramon.

### 6.2.5 Objektumok térbeli kiterjedésének vizsgálata

A gyalogosok és jelzőtáblák pontjai mélység és szélesség tekintetében alacsony térbeli szórással rendelkeznek, viszont magas térbeli szórással rendelkeznek magasság szerint. A járművek magas térbeli szórással rendelkeznek mélység vagy szélesség szerint, attól függően, hogy az adott objektum hogyan helyezkedik el a térben. A térbeli elhelyezkedéstől függően (mélység vagy szélesség) és magasság szerint pedig közepes térbeli szórással rendelkeznek.




*6.14 ábra Objektum fő szórási irányának orientációja*

A (6.14) ábrán szereplő térbeli orientáció is a segítségünkre van az objektumok megkülönböztetésében. Ez a tulajdonság az objektumok fő orientációja és a (0, 0, 1) felfelé mutató vektor által bezárt szög méretén alapszik.

A fő orientáció megállapításához, ki kell számolni az adott objektumhoz tartozó ponthalmaz kovariancia mátrixának a három sajátértékét $\lambda_1 > \lambda_2 > \lambda_3$ a megfelelő sajátvektorokkal $\mu_1 > \mu_2 > \mu_3$ együtt. A $\mu_1$ sajátvektor fogja megmutatni az objektum legnagyobb térbeli varianciájának az irányát [1].

Azt várjuk, hogy a $\mu_1$ és a (0, 0, 1) vektor által bezárt szög járművek esetében nagy lesz, még jelzőtáblák esetében kicsi (6.14 ábra).

A következő indikátor függvény definiálja a működést:

$$f_{so} = 1\left(\arccos\left(\frac{\lambda_1 * \mu_1}{|\lambda_1 * \mu_1| * |\nu_{up}|}\right) < \sigma_{so}\right),$$

ahol $\sigma_{so}$ egy szög küszöbérték a $\mu_1$ sajátvektor és az felfelé mutató vektor között ($\nu_{up}$).

### 6.2.6 Személygépjárművek detektálása

Munkám során a személygépjárművek detektálására fektettem a fő hangsúlyt. Az algoritmus későbbi tesztelése is olyan adatbázisokon történt, ahol kifejezetten a gépjárművek voltak felannotálva. Egy átlagos autó felülete elég nagy ahhoz, hogy akár 50-60 méteres távolságból is elég pontot adjon vissza a sikeres klasszifikáláshoz.

Tipikus jellemzők személygépjárművek esetében:

1. Egy átlagos autó objektum x, y és z irányú kiterjedése méterekben mérhető, vagyis már ezen három érték alapján sok kicsi objektumot ki lehet zárni. Az autó nem mindig látszik jól, takarásban lehet, vagy esetleg túl távol volt, ezért csak ezen jellemzők alapján nem lehet postosan klasszifikálni, ráadásul egyéb nagyobb objektumokkal (pl. fal szegmensek, bokrok) is találkozunk a színtéren.
2. Az autókra jellemző, hogy az (x, z) sík valamely irányában van a fő terjedési irányuk, viszont a másik két irány kiterjedése sem elhanyagolható. Egy falszegmens a legtöbb esetben valamely vízszintes irányban nagyon és magasság szerint is jól szór, de a harmadik irány kiterjedése elhanyagolható.
3. Egy gépjármű esetében sok esetben két, de akár három oldalról is van adatunk. Ezekben az esetekben megfigyelhető, hogy a gépjármű számított közepében tipikusan kevés adat található, még az oldalfalak sok adatot adnak vissza. Ezért olyan objektumokat keresünk ahol a kitöltöttség főleg az objektum széleire korlátozódik. Egy bokor vagy fa lombkoronájának esetében a lézer behatol a levelek közzé és egy sokkal egyenletesebben kitöltött objektumot kapunk.

Az objektumokból kinyert leírókból jellemző vektorokat hozunk létre. Az SVM ezeket a jellemző vektorokra fogja használni a tanulás és osztályozás során. Definícióm szerint a jellemző vektor tartalmazza az objektum *x, y, z* irányú méreteit, kiterjedési irányait, ponteloszlását az objektum centrumából nézve, intenzitás képét, és az objektumot alkotó pontok számát.

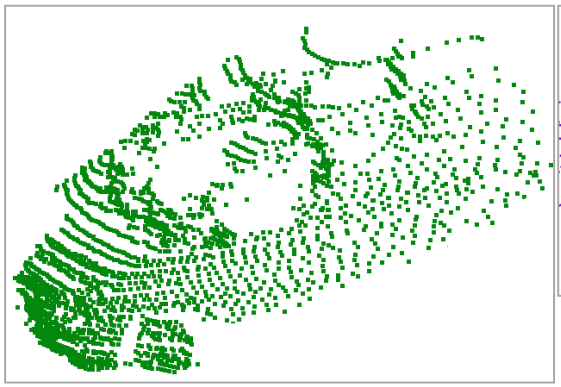
### 6.2.7 SVM alapú klasszifikálás

A Support Vector Machine (SVM) gépi tanulásnál alkalmazott felügyelt tanulási modell. Adott egy tanító halmaz, amiben minden egyes entitás meg van jelölve, hogy melyik osztályba tartozik, és az SVM ebből az annotált halmazból épít egy modellt, az új entitások klasszifikálásához. Az SVM a klasszifikáláshoz egy hipersíkot, vagy hipersíkok halmazát hozza létre egy sokdimenziós térben. Olyan hipersík keresése a cél, ami jól választja el egymástól az osztályok elemeit. A **Support Vector** a hipersíkhoz legközelebb eső pontok, az osztályokból. A **Margó** az elválasztó hipersíkkal párhuzamos hipersíkok által meghatározott térrész, amely nem tartalmaz tanítópéldákat. A módszer heurisztikája az, hogy a szeparáló hipersíkot úgy kell megválasztani, hogy a Margó maximális legyen.

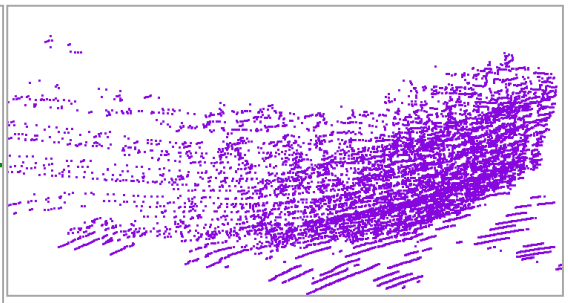
A (6.2.6) fejezetben bemutatott jellemzők kinyerése után SVM alapú tanulással és klasszifikálással döntött az algoritmus, hogy az adott objektum autó e vagy sem. Első lépésben létrehoztunk egy tanító adatbázist pozitív és negatív mintákkal, ahol pozitív minta csak a személygépjármű volt, a negatív minták pedig magukba foglalták az összes többi járműtípust, falszegmenseket, oszlopokat, fákat, bokrokat járókellőket és egyéb utcai objektumokat. A tanuló halmazba a pontfelhőből véletlenszerűen kivágott azonosíthatatlan objektum minták is kerültek negatív példaként. Az általunk annotált adatbázisban 1600 pozitív és 4000 negatív minta volt. A tanításhoz a jól ismert KITTI adatbázis [24] mintáit is felhasználtuk, mely további 12715 pozitív és 3396 negatív mintát adott a tanuló adatbázishoz.

A munka során a [23]-ban bemutatott SVM-et használta az algoritmus, mely bináris osztályozást végzett a személygépjármű és a többi osztály objektumai között. Azért, hogy minél pontosabb eredményt érjen el az algoritmus 7 db SMV-et használtunk, melyek a tanuló adatbázis különböző részeire tanultak rá random mintavételezéssel, így az átlapolódás is meg volt engedve a tanuló halmazok között. Mind a hét SVM egyformán a minták negyedére tanult rá random mintavételezve az entitásokat. Egy új egyed klasszifikálásakor mind a hét SMV szavaz, hogy szerinte autó-e vagy sem az adott objektum, majd a szavazatok lineáris kombinációja dönt a végső szavazatról. Jelen algoritmusnál, ha legalább 5 SVM igennel szavazott, akkor tekintjük autónak az objektumot.

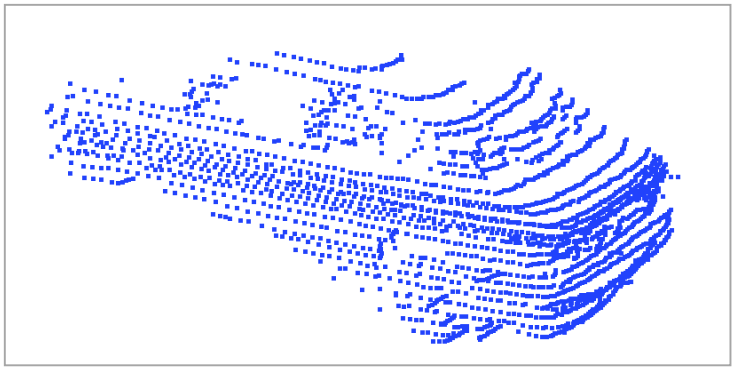
A 6.15 ábrán pozitív illetve negatív tanulóminták láthatók, melyeket az SVM tanítására használtunk fel. További tanítóminta példákat mutatunk a 4.7, 4.8, 4.9, 4.10, 4.11 ábrákon. A tanító mintapéldák között továbbá számos bokor, fa és utcai objektum is megtalálható.



*a) Személygépjármű, pozitív minta*



*b) Fal szegmens, negatív minta*



*c) Személygépjármű, pozitív minta*

*d) Ember, negatív minta*

*6.15 ábra Példák pozitív és negatív tanítómintákra, járműfelismerés feladathoz*

# 7. A munkafolyamat eredményei

A kísérletek során standard CPU (Intel Core i7) környezetben egy időkeret pontfelhő feldolgozása átlagosan 40-45 ms sebesség mellet történt. Ez átlagosan 23 időkeret feldolgozását teszi lehetővé másodpercenként, ami elég ahhoz, hogy dinamikai változásokat tudjunk nyomon követni és valós időben dönteni.

| Pontfelhő adathalmaz | Teszt obj. | PCA alapú megközelítés [6] | | Bemutatott megközelítés | |
|---|---|---|---|---|---|
| | | F-érték (%) | Átl. feldolgozási sebesség (FPS) | F-érték (%) | Átl. feldolgozási sebesség (FPS) |
| Budapest adathalmaz 1 | 567 | 73 | 15 | 89 | 24 |
| Budapest adathalmaz 2 | 1141 | 71 | 12 | 90 | 21 |
| Budapest adathalmaz 3 | 368 | 57 | 13 | 80 | 22 |
| KITTI adathalmaz [24] | 614 | 62 | 14 | 78 | 25 |
| **Összes** | **2690** | **68** | **13.5** | **86** | **23** |

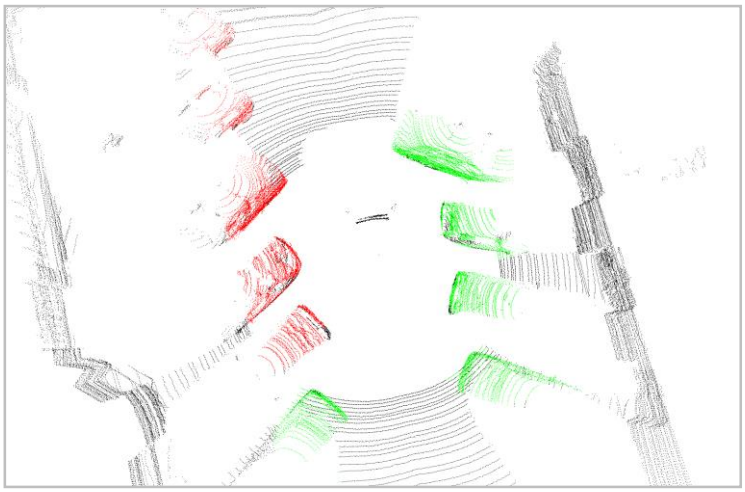
*Táblázat 7.1 Objektum klasszifikáció kiértékelése pontosság és feldolgozási sebesség alapján. A dolgozatban bemutatott módszer egy PCA alapú referenciamódszerrel került összehasonlításra.*

A dolgozatban bemutatott módszert, négy különböző típusú városi környezetben készült, Lidar pontfelhő szekvencián tesztelem. A tesztszekvenciák között voltak főutak, szűk mellékutcák és kereszteződések. Az első három teszthalmazt Budapest utcáin rögzítettük kollégáimmal, a negyedik pedig a nyilvánosan elérhető KITTI adatbázisban [24] található meg. Az összes teszt szekvencia a Velodyne HDL-64E S2 Lidar szenzor segítségével készült, 10 Hz-es forgási sebesség mellett.

A dolgozatban bemutatott modell-alapú eljárást egy referenciamódszerrel is összehasonlítottuk, ami egy egyszerű foglaltsági rácsot használ az előtér szeparálására és főkomponens analízis (PCA) alapú jellemző kinyerésre támaszkodik az objektumok klasszifikációjánál [6].

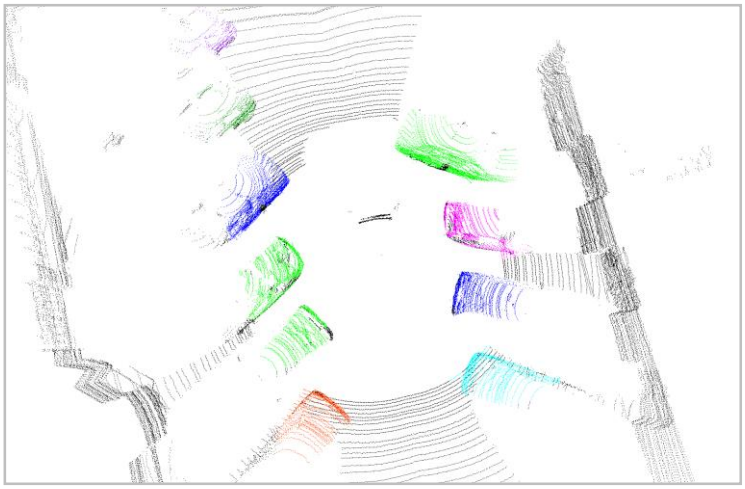
A kiértékelés során az algoritmusomat és a referenciamódszert 2690 járművön teszteltem, melyeket előzőleg manuálisan annotáltunk. Az automatikus kiértékelés során szükség volt a detektált objektumok és a referencia objektumok legjobb párosításához, amihez a Magyar módszert [25] használtam fel. A párosítást követően, a kiértékelő eljárás megszámolja a hiányzó, és a helytelenül detektált objektumokat. A kapott értékek összevetésre kerülnek a valós objektumok számával, majd kiszámoljuk a detekció F-értékét, ami a pontosság (precision) és a visszahívás (recall) harmonikus átlaga. Továbbiakban összehasonlítottuk a két módszer feldolgozási sebességét. A számszerű kiértékelést a 7.1 táblázat mutatja. A kiértékelés eredménye megmutatja, hogy a dolgozatban bemutatott módszer felülmúlja a PCA-alapú

referenciamódszert az F-érték tekintetében, ráadásul a saját módszer jelentősen gyorsabb is. A szűk mellékutcákban, ahol sokkal közelebb helyezkednek el az objektumok és így sok közülük össze is olvadhat a szegmentáció során a saját módszer ott is sokkal jobb eredményeket produkált. A saját algoritmus átlagos feldolgozási sebessége 23 FPS, ami több mint kétszer gyorsabb volt a szenzor forgási sebességénél (10 Hz).



*7.1 ábra Algoritmus futási eredménye egyrétegű ráccsal*

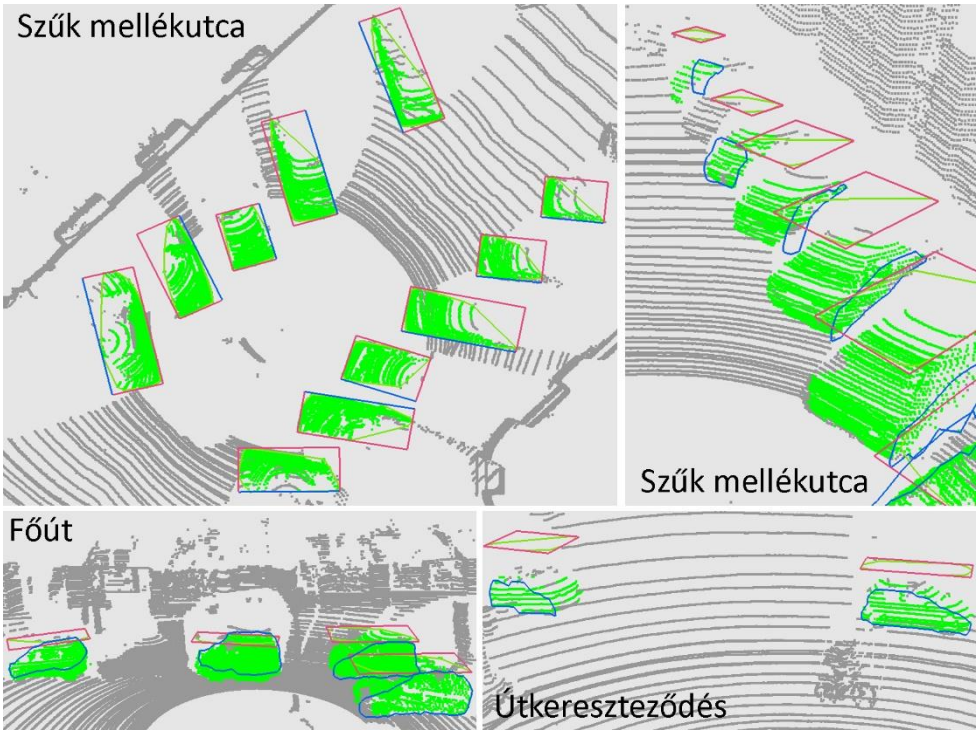
A (7.1) ábrán megfigyelhető az egyrétegű rács hátránya. Futási időben gyorsabb, mint a kétrétegű rács, de sok közeli objektumot egy objektumnak detektál.



*7.2 ábra Algoritmus futási eredménye kétrétegű ráccsal*

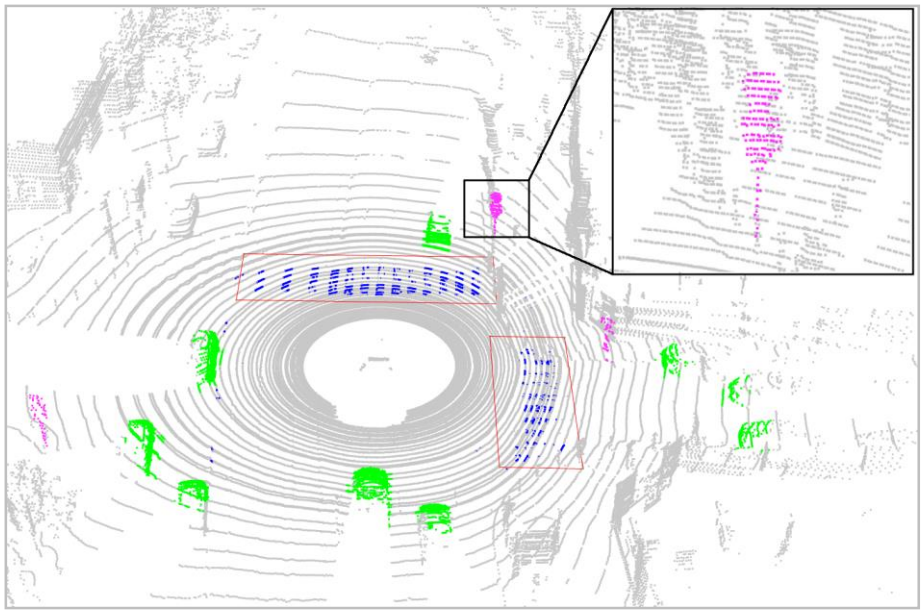
A (7.2) ábrán a kétrétegű ráccsal futó algoritmus eredménye látható. Megfigyelhető, hogy a legtöbb esetben szét tudta választani a közeli objektumokat.

Az objektumok klasszifikálása, felismerése során, kulcsfontosságú az objektumok minél jobb szeparálása. A (7.1) ábra esetén a 10 autót 3 egybefüggő objektumként észlelte az algoritmus így a felismerés nem járhatott sikerrel, de a (7.2) ábrán ugyanazt a 10 autót már 9 különálló objektumként ismerte fel az algoritmus, ami nagymértékű javulásnak tekinthető.




*7.3 ábra Objektum detekció kvalitatív eredménye. Az ábrán pirossal az objektum felülnézetének befoglaló téglalapja és kékkel az oldalnézet konvex burkolója látható, melyet az algoritmus nyert ki.*

A (7.3) ábra demonstrálja a klasszifikáció eredményét. Az ábrán megfigyelhető a konvex burkoló által javított orientációjú befoglaló téglalap illesztése.



*7.4 ábra Az objektum klasszifikációs algoritmus kimenete egy városi szcenárió esetén*

A (7.4) ábrán a zöld színű objektumok a jármű osztályba sorolt elemek, a magenta színnel jelölt objektumok a közlekedési táblák. Két gyalogos átkelőhely található az ábrán, melyek intenzitásuk alapján lett kiemelve és piros befoglaló téglalap határolja őket.

# 8. A munkafolyamat implementációs kérdései

## 8.1 Pont típus és adatstruktúra

A pontfelhő egy olyan adatstruktúra, melyet három vagy többdimenziós pontok rendezetlen halmazának reprezentálására használnak. A pontokhoz az *x, y, z* térbeli Descart-koordinátáik mellett hozzárendelhetünk szín vagy intenzitásinformációt is [5]. A pontfelhők implementáció szinten történő hatékony kezelése kritikus eleme a fejlesztési folyamatnak. A munkám során C++ programozási környezetben dolgoztam, különböző segédkönyvtárak felhasználásával, melyekről a következő alfejezetben adok rövid áttekintést.

## 8.2 Pontfelhőszekvenciák feldolgozáshoz szükséges programkönyvtárak

**PCL (Point Cloud Library)** (http://pointclouds.org/)

A PCL egy C++ nyílt függvény könyvtár, amelyet 2D/3D képek és pontfelhők kezelésére hoztak létre. Számos alapvető adatstruktúrát, algoritmust tartalmaz, különböző feladatokra, úgymint pontfelhők zajszűrése, felszín rekonstrukció, adat regisztráció, modellillesztés és szegmentálás. Ezen alapvető építőkockák hatékonyan beépíthetők a saját algoritmusokba, melyek segítségével a pontfelhők hatékonyan kezelhetővé válnak.

**Boost Library** (http://www.boost.org/)

A Boost Library a standard C++ kiegészítése. Támogatja a C++ 11 szabványt, számos hasznos algoritmus és osztály implementációját tartalmazza, amik elősegítik a hatékony és hibamentes kódolást. A projekt során a Boost könyvtár és a C++ 11 szálkezelési technikáit használtam az algoritmusok gyorsításának céljából. Ahol lehet ott az adatszerkezetet több részre osztottam és a feldolgozás több szálon történik meg vagy épen az egyes algoritmusok párhuzamosan futnak.

**Qt** (http://qt-project.org/)

A Qt egy platform-független alkalmazás-keretrendszer. A szoftver vizuális felülete, menürendszere ebben készült.

**OpenCV** (http://opencv.org/)

Az OpenCV egy nyílt forrású számítógépes látást és gépi tanulást támogató szoftverkönyvtár. Több ezer algoritmus található benne, melyek gyorsabbá és könnyebbé teszik a kép és video feldolgozással kapcsolatos problémák megoldását és támogatja a 3D pontfelhők kezelését is.

**Többrétegű rács adatstruktúra felépítése**

Ha minden egyes időkeretnél az egész pontfelhőt szeretnénk betölteni az adatstruktúrába, akkor az adott pontfelhő x, y és z irányú legnagyobb kiterjedésének meghatározása után mindig újra kellene építenünk a kétrétegű rácsot. Ez nagyon költséges megoldás, ezért egy inicializálási fázisban felépítek egy adott fix méretű rácsot és a rács celláiba eső pontokat folyamatosan frissítem az egyes időkeretek betöltésekor. Ennek hátránya, hogy mindig csak egy fix méretű tartományt tudunk vizsgálni, azonban sokkal gyorsabb lesz a feldolgozás.

A rács méretei a feldolgozás kezdetekor paraméterezhetőek és azt is be lehet állítani, ha esetleg nem két réteggel szeretnénk dolgozni. Használhatunk egy, de akár kettőnél több rétegű rácsot is. A rács réteg számainak növelésével csökken a diszkretizáció mértéke, azonban minél több réteggel dolgozunk annál lassabb lesz a feldolgozás.

# 9. Konklúzió, jövőbeli tervek

Dolgozatomban algoritmusokat ajánlottam városi objektumok szeparálására és klasszifikálására, amikhez egy új kétrétegű rács alapú valós idejű térparticionáló modellt dolgoztunk ki. A módszer közeli objektumok esetén is robusztus eredményt ad. Továbbá a dolgozat bemutat hatékony objektum felismerési módszereket, melyek geometriai és intenzitás alapú jellemzők kinyerésén alapulnak. A kinyert jellemzők hatékony kombinációjával az egyes objektumokat csoportokba sorolhatjuk (autó, jelzőtábla, gyalogos). A korábbi szakirodalmi módszerekkel ellentétben, a dolgozatban bemutatott módszer optimális paraméterezése nem függ közvetlenül az detektálandó objektumok geometriájától.

A közeli objektumok robusztus szeparálásához a kétrétegű rács modell két döntési kritériumot használ. Az első szinten a pontok közötti magasságkülönbséget használja ki, még a második szinten a pontfelhő sűrűségi változásait veszi figyelembe. A módszer előnye a legtöbb létező megoldáshoz képest, hogy az objektumok klasszifikálásához nem használ előre kézzel felcímkézett tanítóhalmazt [13], [14], [15], [16].

Az objektum jellemzők kinyerése után SVM alapú klasszifikálással az algoritmus két csoportba sorolja az objektumokat. Az egyik csoport a személygépjárművek, a másik pedig az összes többi objektum.

A módszert nagy mennyiségű teszthalmazon kiértékeltem. A teszthalmazok különböző típusú városi környezeteket fedtek le, úgymint főutak, szűk mellékutcák és nagyobb kereszteződések.

**Továbblépési lehetőségek**

Célom a jövőben, a szakirodalomban ismert és bevált többobjektumos követési technikákat beépíteni a keretrendszerbe, a precizitás növelésének érdekében. A követés eredményét fel szeretném használni kontextus alapú jellemzők kinyerésére és a környezet magasabb szintű értelmezésére. Továbbá érdekes lehetőségnek gondolom a követés eredményeinek felhasználását az egyes időkeretek regisztrációjához.

Későbbi kutatási irányom lehet más szenzorok adatainak összeegyeztetése a Velodyne szenzor 3D pontfelhőjével, ami lehetőséged ad az objektumok pontosabb klasszifikálására.

# 10. Köszönetnyilvánítás

Szeretném megköszönni konzulenseimnek Dr. Benedek Csabának és Börcs Attilának a munkám során nyújtott segítségét és szakmai támogatást. Köszönöm az MTA SZTAKI Elosztott Események Elemzése Kutatólaboratóriumának, hogy rendelkezésemre bocsátották a részleg eszközeit a munka elvégzéséhez. Továbbá szeretném megköszönni az MTA SZTAKI összes dolgozójának a munkáját, akik segítettek és biztattak a munkafolyamat során.

# Ábrajegyzék

# Irodalomjegyzék


[1] A. Börcs, B. Nagy and Cs. Benedek "On board 3D Object Perception in Dynamic Urban Scenes," In: *IEEE International Conference on Cognitive Infocommunications*, 2013, Budapest, Hungary, pp. 4.

[2] A. Börcs, B. Nagy, Cs. Benedek, ”Fast 3-D urban object detection on streaming point clouds” In: *Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving at ECCV*, Lecture Notes in Computer Science, 2014, Zürich, Switzerland

[3] A. Börcs, B. Nagy, Cs. Benedek, ”A Model-based Approach for Fast Vehicle Detection in Continuously Streamed Urban LIDAR Point Clouds ” In: *Workshop on Scene Understanding for Autonomous Systems at ACCV*, 2014, Singapore, pp. 8-9.

[4] O. Józsa, A. Börcs and Cs. Benedek: ”Towards 4D Virtual City Reconstruction From Lidar Point Cloud Sequences,” In: *ISPRS Workshop on 3D Virtual City Modeling, 2013, Regina, Saskatchewan, Canada*

[5] O. Józsa and Cs. Benedek, ’’Reconstruction of 3D Urban Scenes Using a Moving Lidar Sensor,” MTA SZTAKI, 2013, Budapest, Hungary, Tech. Rep. N ˚ i4D-1, pp. 10-13.

[6] M. Himmelsbach, A. Müller A. T. Lüttel, and H.-J. Wünsche, “LIDAR based 3D Object Perception,” In: *Proceedings of 1st International Workshop on Cognition for Technical Systems*, 2008, Munich, Germany, pp. 1-2.

[7] C. Fulgenzi, A. Spalanzani, and C. Laugier, “Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid,” In: *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007, Rome, Italy, pp. 1610–1616.

[8] Bluesky Intrenational Limited. LIDAR-UK [Online]. Available: http://www.Lidar-uk.com/

[9] Velodyne Lidar 2012. HDL-64E [Online]. Available: http://velodyneLidar.com/Lidar/hdlproducts/hdl64e.aspx

[10] Volvo Car Group. Road Sign Information [Online]. Available: https://www.media.volvocars.com/global/enhanced/engb/Media/Preview.aspx?mediaid=47896

[11] E. Guizzo (2011, Oct 18). How Google's Self-Driving Car Works [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[12] A. Petrovskaya and S. Thrun: ”Model Based Vehicle Tracking for Autonomous Driving in Urban Environments,” Auton. Robots 26, 2009, pp. 123-139.

[13] M. Samples and M. R. James, “Learning a real-time 3D point cloud obstacle discriminator via bootstrapping,” In: *Workshop on Robotics and Intelligent Transportation System*, 2010, Anchorage, Alaska

[14] K. Lai and D. Fox, “Object recognition in 3D point clouds using web data and domain adaptation,” *I. J. Robotic Res.*, vol. 29, no. 8, pp.1019–1037, 2010.

[15] A. J. Quadros, J. Underwood, and B. Douillard, “An occlusionaware feature for range images,” In: *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, St. Paul, USA, pp. 4428–4435.

[16] C. Liang-Chia, H. Hoang Hong, N. Xuan-Loc, and W. Hsiao-Wen, “Novel 3-D object recognition methodology employing a curvaturebased histogram,” *I. J. Robotic Res.*, vol. 10, 2013, pp. 1019–1037

[17] Center for Coastal and Ocean Mapping Joint Hydrographic Center [Online]. Available: http://ccom.unh.edu/theme/Lidar

[18] A. Azim, O. Aycard "Detection, classification and tracking of moving objects in a 3D environment". In: *IEEE Intelligent Vehicles Symposium (IV)*, 2012, Alcala de Henares, Spain, 802–807

[19] J.F Lalonde, N. Vandapel, D. Huber, M. Hebert ”Natural terrain classification using three-dimensional ladar data for ground robot mobility” In: *Journal of Field Robotics 23*, 2006

[20] A. Golovinskiy, V.G. Kim, T. Funkhouser ”Shape-based recognition of 3D point clouds in urban environments”, 2009, Kyoto, Japan, pp. 2-4.

[21] B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, S. Singh ”A pipeline for the segmentation and classification of 3D point clouds” In: *In ISER.*, 2010, pp. 2-4.

[22] D.Z. Wang, I. Posner, P. Newman ”What could move? finding cars, pedestrians and bicyclists in 3d laser data” In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2012, Minnesota, USA

[23] A. Andrew ”Another efficient algorithm for convex hulls in two dimensions” In: *Information Processing Letters 9*, 1979

[24] A. Geiger, P. Lenz, R. Urtasun ”Are we ready for autonomous driving? the kitti vision benchmark suite” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012

[25] H. Kuhn ”The Hungarian method for the assignment problem” In: *Naval Research Logistic Quarterly 2*, 1955

[26] B. Benfold and I. Reid ”Stable multi-target tracking in real-time surveillance video” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, Providence, RI

[27] Vo Ba-Ngu, S. Sumeetpal and A. Doucet ”Sequential monte carlo implementation of the phd filter for multi-target tracking” In: *Information Fusion, Proceedings of the Sixth International Conference of Volume 2*, 2003, Cairns, Queensland, Australia