

Városi környezet automatikus analízise és rekonstrukciója légi LIDAR mérések alapján



Börcs Attila, 2. évfolyam

Horváth Csaba, 2. évfolyam

TDK dolgozat

Konzulensek:

Dr. Benedek Csaba, tudományos főmunkatárs

Dr. Szirányi Tamás, tudományos tanácsadó

Pázmány Péter Katolikus Egyetem

Információs Technológiai Kar

Budapest, 2011

Összefoglaló

Napjainkban a környezetről gyűjtött információk kiemelt prioritást kapnak az ipari, keresdelemi és szolgáltatási folyamatok kialakításában. Ugyanakkor a szolgáltatott távérzékelte adat feldolgozása jelenleg nagy mennyiségű emberi erőforrást igényel, figyelembe véve az adatbázisok folyamatos frissülését és a szükséges kiértékelési módszerek és szempontok bővülését. A feldolgozásban ezért célszerű automatizálni a különböző felismerési, osztályozási és szegmentálási folyamatokat. Légi fotókon vagy műholdképeken ezeknek a feladatoknak az automatikus megoldása máig nagy kihívást jelent, mivel sok tényezőtől függ a felvételek minősége, továbbá az alakfelismerési és mintaillesztési feladatok is bonyolultak a kétdimenziós képeken. A LiDAR (Light Detection And Ranging) lézerszenner technológia segítségével háromdimenziós mérési adatokat, úgynevezett pontfelhőket készíthetünk a megfigyelt tájról. Tudományos diákköri munkánk során levegőből rögzített LiDAR pontfelhőkből kiindulva, automatikus városi környezet értelmezéssel, helyszín és objektum felismeréssel majd 3D városrekonstrukcióval kapcsolatos funkciókat valósítottunk meg. Az elkészült algoritmusainkat három csoportba oszthatjuk: a) Módszereket terveztünk és teszteltünk kis felbontású (8 mérés / m^2) pontfelhők osztályozására, különböző objektumok és területi osztályok (talaj síkja, háztetők, növényzet, zaj) elkülönítésére. b) Eljárást dolgoztunk ki járművek felismerésére, megszámlálására és lokalizálására a nyers pontfelhőben c) Automatikus eljárást javasoltunk virtuális épületrekonstrukció megvalósítására a struktúrátlan pontfelhőből kiindulva. A megvalósított módszereket valódi távérzékelte LiDAR adatokon teszteltünk, amit az Infoterra Astrium GEO-Information Services Hungary © bocsátott rendelkezésünkre.

Tartalomjegyzék

Összefoglaló	ii
1. Bevezetés	1
1.1. Irodalmi áttekintés	2
2. A 3D LiDAR alapú távérzékelés eszközei	8
2.1. A LiDAR rendszer általános felépítése	8
2.2. Nyers adatok vizualizációja	12
2.3. Fejlesztőeszközök	13
3. Felhasznált matematikai eszközök és algoritmusok	17
3.1. Főkomponens analízis (<i>PCA</i>)	17
3.2. Területelárasztó (<i>floodfill</i>) algoritmus	19
3.3. Térparticionáló módszerek	21
3.3.1. Oktális fa	21
3.3.2. Kd-fa	23
3.4. <i>RANSAC</i> eljárás	25
4. Elkészített ponthalmaz-elemző rendszer bemutatása	29

4.1. Pontfelhő klasszifikálás	30
4.1.1. A talaj síkjának szegmentálása	30
4.1.2. Növényzet és ritka pontthalmazok felismerése	32
4.1.3. Épületek háztetőinek detektálása	34
4.1.4. Járműdetekció	41
4.2. Virtuális rekonstrukció pontfelhőben	48
4.2.1. Tetőszerkezetek poligon approximációja	48
4.2.2. Épületek oldalfalainak szintetikus generálása	50
5. Konklúzió és további lehetőségek	53
6. Köszönetnyilvánítás	55
Irodalomjegyzék	56

1. fejezet

Bevezetés

A földfelszín megfigyelése kulcsszerepet tölt be számos alkalmazási területen: távérzékelte adatokat használnak mezőgazdasági tevékenységek felügyeletéhez, lakott területek fejlődésének vizsgálatához, környezetvédelmi és kármegelőzési feladatokban, illetve segítséget nyújthatnak nemzetbiztonsági és büntetőrendészeti kérdések megválaszolásához is. A különböző távérzékelési technológiák fejlődésével és széleskörű elterjedésével a szolgáltatott adatok mennyisége és részletgazdagsága is ugrásszerűen megnőtt az utóbbi évtizedben. Ugyanakkor a rendelkezésre álló információfolyam feldolgozása hatalmas mennyiségű emberi erőforrást igényel, figyelembe véve az adatbázisok folyamatos frissülését és a szükséges kiértékelési módszerek és szempontok bővülését. A feldolgozásban ezért nagy mértékű automatizálásra van szüksége, ami napjaink kiemelt kutatási területévé vált. Az elterjedt távérzékelési adatforrások közül dolgozatunkban légi platformra szerelt LiDAR (Light Detection and Ranging) lézerszkenner adatának elemzésével foglalkozunk.

Egy valós városi környezetről vagy domborzatról származó mérések olyan pont-

felhőket eredményeznek amelyben, a pontok irregulárisan helyezkednek el a mérési zaj valamint a mérési felbontás hiányának esetleges zavaró tényezői miatt. Az ilyen jellegű adatokból történő lényegkiemelés komoly kihívásokat tartalmaz. Egy másik kardinális kérdés, hogy a lényegkiemelés után, a már szemantikailag értelmezett pontfelhőkre hogyan tudunk úgy geometriai primitíveket illeszteni, hogy egy - a valóságot közelítő - rekonstrukciót kapjunk. A szakirodalomban a legtöbb megoldás olyan matematikai modellekre, tanulmányokra és klaszterező technikákra támaszkodik amelyek zajos, irreguláris és esetenként hiányos mérési adathalmazt próbálják hatékonyan értelmezni, illetve rekonstruálni.

1.1. Irodalmi áttekintés

Az alábbi fejezet betekintést enged a LiDAR alapú 3D pontfelhő feldolgozással foglalkozó tudományos eredményekbe. A technológiának számtalan alkalmazási területe van: növényzet és fák elemzése, városi környezetben található házak, háztetők, úthálózatok vizsgálata, járműdetekció, továbbá katasztrófa súlytott területek felmérése és egyéb virtuális rekonstrukciós feladatok ellátása. Jól használható olyan erdőgazdasági alkalmazásokban, ahol fák magasságára, illetve fajtájára akarunk becslést adni, egy kiterjedt területen. A becslésekre különböző technikák ismertek az irodalomban. Elterjedt módszer, hogy a mérési adatok magasságkoordinátáját felhasználva úgynevezett magasságtérképet (Digital Elevation Model - DEM) készítenek, ami alapján a fákat magasság szerint osztályozzák [22]. Ezzel a módszerrel jól modellezhetők kiterjedt erdőrészek szerkezete, ami a erdészet hasznos segítőeszköze. A LiDAR eszközök további előnye, hogy a pontok háromdimenziós koordinátái mellett, egyéb járulékos

adatok is kinyerhetők és eltárolhatók, amik szintén felhasználhatóak a feldolgozási folyamatban. Ilyen kiegészítő adat, a berendezés által adott irányban kibocsátott lézersugárzáshoz tartozó visszaverődések (ekhók) száma. Ez kifejezetten hasznos, amikor növényzet, erdős területek, illetve városi környezetben elhelyezkedő fák detektálása a cél, hiszen a lombkorona szerkezetéből adódóan egyes lézer nyalábok már a fa koronájáról visszaverődnek, mások átjutnak a fa koronán és lejjebb lévő ágakról verődnek vissza, esetleg érintik a talajt. Ezekből a jellemzőkből hatékonyan lehet szűrni a növényzettel borított területeket egy pontthalmazból, hiszen ahol volt többszörös visszaverődés, ott nagy valószínűséggel növényzet van. A mérési pontok szomszédsági viszonyait vizsgálva, számítható egy átlagos és egy kurrens magasság érték egy bizonyos környezetben, aminek a statisztikai vizsgálatával meghatározhatunk egy alsó, és egy felső határt az erdős részek kiszűrése érdekében. A visszaverődés vizsgálat és a magasságra vonatkozó korlátok fúzionálásával hatékony módszert kaphatunk fák detektálására [9].

Egy másik kiemelt területe a LiDAR alapú távérzékelési feladatoknak a katasztrófa súlytott területek vizsgálata. Köztudott, hogy milyen elsöprő erejű lehet egy földrengés, földcsuszamlás és egyéb természeti katasztrófa. A távérzékelés feladatköre magában hordozza azt a lehetőséget, hogy eszközeivel lehetőségünk nyílik gyorsan és jó hatékonysággal nagy kiterjedésű katasztrófa súlytott területeket elemezni és vizsgálni. Kiemelt fontosságú cél a kárfelmérés, az épületek szerkezeti megváltozásainak vizsgálata: homlokzati és tetőszerkezet rongálódása, illetve az alapzat dőlése és esetleges megsüllyedése. Ezek a vizsgálatok a katasztrófa okait kereső és lefolyását elemző szakembereknek is hasznos információt nyújtanak, ugyanis a különböző objektumok megrongálódásának mértékéből és a károk elhelyezkedéséből utólag követ-

keztethetünk a katasztrófa lefolyására.

Egyes dolgozatok kiterjedt alapuktatást végeznek az tetőtípusok szerkezeti felépítését illetően, annak érdekében, hogy minél pontosabb és a probléma megoldása szempontjából hatékony algoritmusokkal lehessen felmérni a károkat [17]. A szerzők RANSAC (Random Sample Consensus) algoritmust és Hough-transzformációt felhasználva becslést adnak tetőszerkezetek klasszifikálására. Megkülönböztetnek lapostetőket és különböző sátoztető típusokat, annak érdekében, hogy a talajjal merőleges szöget bezáró vezető síkkal tudják modellezni a vizsgált házak tetőszerkezetét. Lapos tetőtípusok esetén ez a vezetősík megegyezik a tető síkjával, a függőleges irányt pedig ennek a síknak a normálvektora reprezentálja, míg a sátoztetők esetén, a tetőket felépítő sík felületek normálisának az összegéből képeznek egy vektort, ami egyértelműen kijelöli a tetőre merőleges irányú normálist. Az algoritmus második lépése, hogy kijelöli a talaj és a talajra merőleges referencia irányokat és ezekkel az irányokkal veti össze a házak falainak, illetve tetőszerkezetének az irányát. Ha egy bizonyos szögnél jobban eltér a referencia és a ház síkjainak orientáltsága, akkor nagy valószínűséggel az az épület menthetetlenül megrongálódott. Megjegyezzük, hogy ez a módszer csak akkor működik, ha elég mérési adat érkezik az épületek oldalfalairól. Ugyanakkor sok esetben a repülőre szerelt eszköz felülről olyan szögben mér, hogy az oldalfalak pont takarásba kerülnek és nem jelennek meg a szolgáltatott 3D pontfelhőben.

Bizonyos kutatások a differenciálgeometria eszközszerét használják fel annak érdekében, hogy a kársúlytotta övezetben tartózkodó házak tetőszerkezetének állapotát fel tudják mérni, erre az úgynevezett Gauss súlyozással ellátott görbületi operátort használták fel [28].

További érdekes alkalmazási lehetősége a témának egy környezet teljes domborzati és vegetációs vizsgálata, illetve egy városi környezet topológikus jellemzőinek hatékony kinyerése. Városfejlesztésnél vagy elektromos hálózat távvezetékeinek kiépítésénél elengedhetetlen a nagy kiterjedésű terület alapos felmérése, annak érdekében megvizsgálva, hogy az odatelepített objektum számára megfelelőek-e a körülmények, vagy a másik oldalról nézve, az adott környezet mennyire képes befogadni az építendő objektumokat. Ezekben a témákban is előrehaladott eredményeket lehet találni a szakirodalomban. A [27] munkában úthálózatok detektálására ajánlanak algoritmust. Ennek a feladatnak az esetében felmerül, amikor kiterjedt városi környezet úthálózat topológiájáról, kiépítettségéről akarunk kapni releváns és naprakész információkat. Ezek az információk jól hasznosíthatóak például városfejlesztési és útfelújítási feladatoknál, vagy olyan esetekben, amikor azt szeretnék vizsgálni, hogy egy adott úthálózat topológiája mennyire befolyásolja a forgalmi dugókat és a közlekedést egy adott városban.

Az algoritmus 3D-s pontfelhőből kinyert magasságtérképből és a visszaverődött lézervény intenzitásából próbál egy becslést adni az utak pontos lokalizációjához és orientációjához. Megjegyeznénk, hogy ebben az esetben az intenzitás, a kibocsátott és visszaverődött lézervény erősségének a hányadosát jelenti, nem a fent tárgyalt többszörös jelvisszaverődést. Ez a lézer intenzitás az aszfalt felületeknél pontosan kimérhető, így nagy segítséget nyújt utak detekciójánál.

Az elmúlt öt évben előrehaladott eredmények születtek az elektromos távvezetékek kiépítését illetően is [10]. A feladat optimalizálása nagy nehézséget okoz az áramszolgáltatóknak, hiszen nem triviális dolog eldönteni azt, hogy hol fussanak a vezetékek, illetve hova telepíthetők az elektromos vezetékeket tartó póznák. A problémát az

okozza, hogy nehéz azokat az útvonalakat megtalálni, ahol se település, se növényzet nem akadályozza a telepítési munkálatokat, továbbá ha egy jól megválasztott útvonalra üzemelik be a rendszert, kevesebb karbantartási költséggel üzemeltethető a hálózat, hiszen nem kell folyamatosan a vezetékre belógó, illetve ránövő növényzetet irtani. A kutatás célkitűzése, egy olyan teljes vegetáció és domborzati modell elkészítése, amely nagy segítséget nyújthat az elektromos hálózatot telepítő szakembereknek. A nagy kiterjedésű légifelvételek kézi kiértékelése ilyen szempontból nagyon időigényes, ezért határozott segítséget nyújthat egy LiDAR alapú automatikus domborzatkiértékelés. A [11]-ben megfelelő eredmények érdekében neurális hálózatot, és morfológiai rekonstrukciót használtak.

A 3D távérzékelésben a tisztán felismerési feladatok is előtérben vannak. Egyre nagyobb igény mutatozik különböző objektumok, (pl.: közlekedési eszközök, jelzőtáblák) azonosítására, megszámlálására és csoportosítására. Az egyik fontos cél, a különböző járműforgalmi dinamikák megismerése és elemzése. A belvárosi úthálózat járműforgalmának alapos elemzése olyan információval látja el a város szervezeti egységeit, amivel hatékonyabbá tehetik a közlekedést és megelőzhetik a dugók kialakulását. Sőt, olyan hosszútávú közlekedési statisztikák alapját is képezhetik a járműszámlálással foglalkozó távérzékelési feladatok, amely hozzásegít a járműforgalmi dinamikák és viselkedési formák széleskörűbb megismeréséhez. Az irodalom járműdetekcióval foglalkozó cikkei különbözően közelítik meg a problémakört. A módszerek első csoportja [7],[8],[15],[20] az adathalmaz magasságértékeinek tartományra vágásából indul ki, majd főkomponens analízis alapú technikákkal, vagy modell alapú algoritmusokkal próbálják detektálni, illetve rekonstruálni a járműveket. Más modellek bonyolultabb gépi tanulós módszereket használnak, például az Adaboost osztályozót

[23],[26],[25].

A lézeres távérzékelés egyik dinamikusan fejlődő alterülete a virtuális városmodellezés és épület rekonstrukció. Ezekre a szolgáltatásokra manapság hatalmas igény van, példaként említhetjük a Google sikeressé vált applikációit a Google Earth, vagy a Google Maps rendszereket. A virtuális városmodellezési kérdésekkel foglalkozó tudományos munkák gyakran használják a klaszterezés, gépi tanulás továbbá a statisztikai, illetve valószínűségi modellek által kínált lehetőségeket. Első lépésként a 3D pontthalmazok dekompozíciója hatékonyabb modellezési és számolási lehetőségeket kínál. A pontok lokális szomszédságát vizsgálva, számos gyorsan számítható leíró nyerhetünk ki, ami a felhő lokális sűrűségét vagy regularitását jellemzi, illetve sík felületek, sarokpontok vagy élek jelenlétéről árulkodik [1],[13],[21]. Egyes munkák kifejezetten nagyméretű LiDAR adathalmazokra koncentrálnak és arra fektetik a hangsúlyt, hogy temporális memória használatának a segítségével, minél gyorsabban és hatékonyabban lehessen lényegkiemelési eljárásokat futtatni akár 17,7 GB méretű, 683 millió mérési pontot magában foglaló adathalmazokon [14].

2. fejezet

A 3D LiDAR alapú távérzékelés eszközei

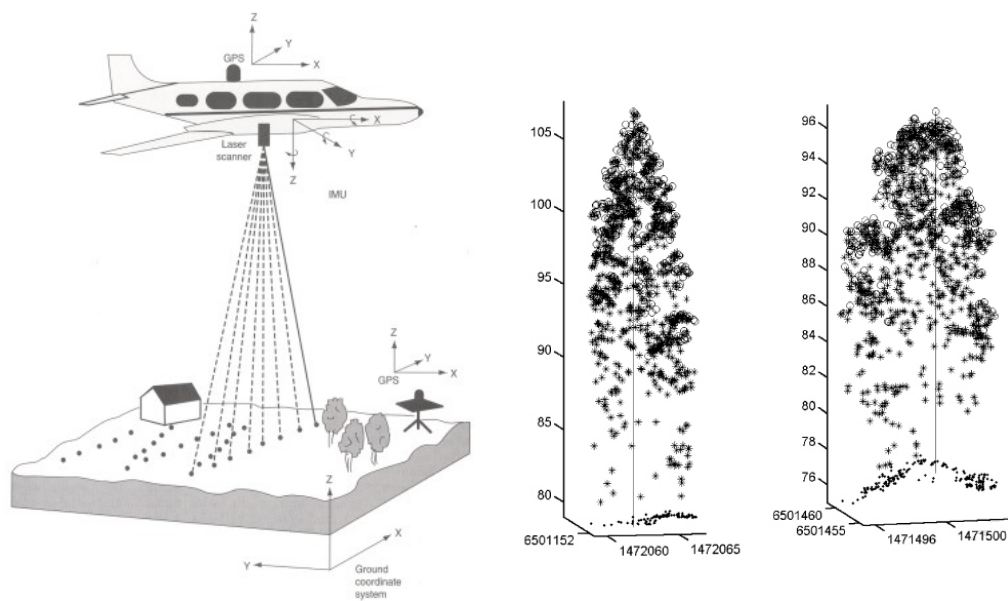
2.1. A LiDAR rendszer általános felépítése

A LiDAR rövidítés az angolszász Light Detection and Ranging elnevezésből származik. Maga a technológia az 1970-es években kezdett utat törni magának, de csak az 1990-es évektől alkalmazzák a gyakorlatban. Azóta számtalan területen bebizonyosodott, hogy egy jól használható és hatékony eszköz, távérzékelési feladatokban is. Gyakran alkalmazzák háromdimenziós adatgyűjtésre a földtudományban, meteorológiában, agrártudományokban, régészetben, hadászatban, csillagászatban és még számtalan más tudományágban. A berendezés alapja egy lézerszkenner, egy globális pozícionáló rendszer (Global Positioning System - GPS) és egy beépített navigációs rendszer (Inertial Navigation System - INS). Ezt a három berendezést összekapcsolva egy olyan eszközt kapunk, amivel nagypontosságú háromdimenziós mérési halmazt tudunk ké-

szíteni. A lézer nagy pontossága miatt akár egy repülőgépre, vagy helikopterre installálva ezeket a rendszereket, háromdimenziós adatokat tudunk készíteni. A működési elv a következő: a szkennelő lézerfényt bocsát ki az alatta elterülő tájra vagy városra. A lézer fény végig pásztázva a területet méri a fény kibocsátásától a visszaérkezéséig eltelt időt. Figyelembe véve a lézer szkennelési szögét, a GPS koordinátákat és az adott légi jármű inerciális navigációs rendszerének eredményeit, előállíthatóak a megfigyelt pontok térbeli x , y , z koordinátái viszonylag nagy pontossággal. Kardinális kérdés, a mérési adatok sűrűsége (felbontása), ez a faktor függ a LiDAR berendezés egyéni paramétereitől. Általánosságban 10^4 mérés történik másodpercenként. Például, ha egy lézerszkennelő tízezerszer lézerimpulzust bocsát ki másodpercenként, akkor hozzávetőlegesen hatszázezer mérés fog történni minden percben.

A LiDAR rendszereknek több fajtája ismert, amelyek különböző célokra alkalmazhatók:

- Topografikus LiDAR rendszert: Légi járművekre (merevszárnyú repülőgépre és helikopterre) installált rendszerek kifejezetten szárazföldi domborzat szkenneléséhez.
- Földi LiDAR rendszert: Gépjárműre installált rendszer, amivel panoráma méréseket tudunk végezni, tipikusan városi környezetben használatos, és gyakran használják együtt a topografikus rendszerekkel.
- Hidrografikus LiDAR rendszert: Tengerfenék domborzatának vizsgálatához használják. Kétféle hullámhosszú lézernyalábbal dolgozik: A piros hullámhossz tartományban lévő lézer a vízfelületről verődik vissza, míg a kék-zöld hullámhossz



2.1. ábra. Balra: Lidar mérőrendszer. Jobbra: Többszörös impulzus visszaverődések fákról

tartományban kibocsátott lézer áthatol a vízfelszínen és a tengerfenékről verődik vissza. A két különböző hullámhossz tartományban mozgó lézernyalábot egyidőben bocsátja ki a szkennert és a visszaverődések közötti időtartam különbségből képes a tenger mélységét perdictálni. Tipikus felhasználási területe a vízalatti gázvezeték beszerelése, illetve tengeri domborzat térképek készítése.

A 2.1 ábrán¹ látható egy teljes LiDAR rendszer felépítése, amely tartalmazza a földi és légi globális pozícionáló eszközt, az inerciális navigációs rendszert, illetve a lézerszkennert. Az ábra jobb oldalán érzékelhetjük, hogy egy erdős területet pásztázva milyen arányban történnek többszörös impulzus visszaverődések.

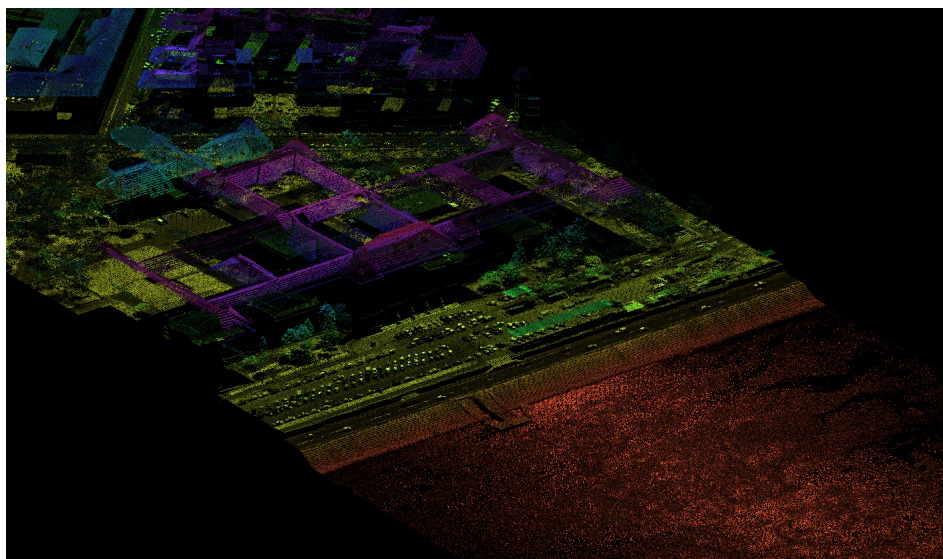
A mérések tárolására több speciális adatformátum is létezik. Az egyik legelter-

¹képek forrásai: <https://www.e-education.psu.edu>, <http://live.osgeo.org/>

Adat	Formátum	Méret	Elvárt?
X	long	4 byte	igen
Y	long	4 byte	igen
Z	long	4 byte	igen
Intenzitás	Unsigned short	2 byte	nem
Visszaverődések száma	3 bit	3 bit	igen
Szkennelési szög	1 bit	1 bit	igen
Klasszifikáció	Unsigned char	1 byte	nem
GPS idő	double	8 byte	igen

2.1. táblázat. A „las” fájlformátumban tárolt adatok a LiDAR mérésről.

jedtebb típus a „las” kiterjesztésű fájlformátum, de használatos szabványok a DEM (Digital Elevation Model), illetve az ASCII „txt” szöveges állományok is. Ahogy korábban említettük, a háromdimenziós koordináták mellett többféle kiegészítő információ is rögzítésre kerülhet a mérés során. A 2.1 táblázat a „las” formátum szintaktikáját használva felsorol pár hasznos adatelemet a teljesség igénye nélkül: a pont koordinátákon kívül, eltárolhatjuk például a lézerefény intenzitását, a többszörös visszaverődések számát és a szkennelési szöget. Minden fájlformátum tartalmaz szabadon felhasználható adattagot is, ahol a mérés utáni feldolgozás során kapott információt - például a klasszifikáció eredményét - tárolhatjuk.



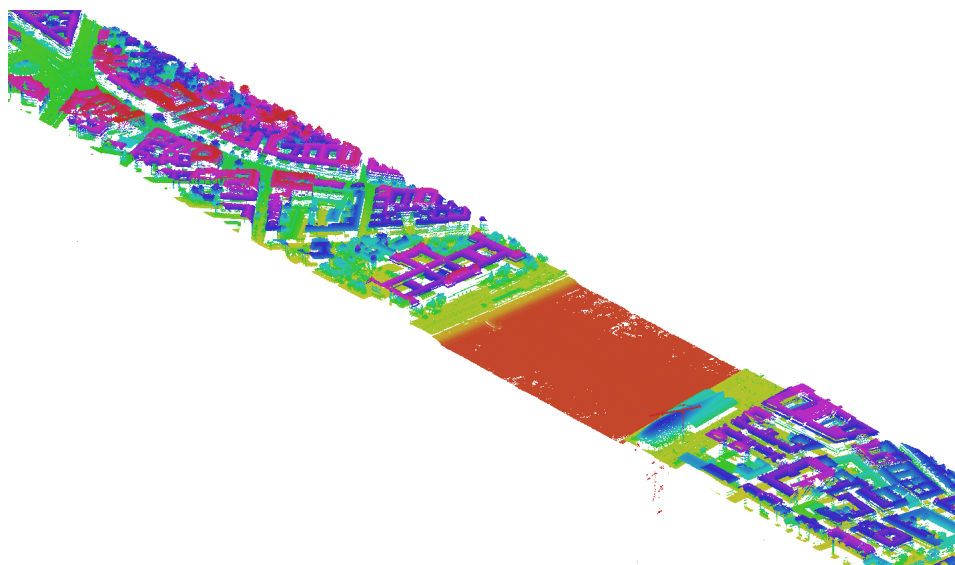
2.2. ábra. Magasságérték és intenzitás alapján történő színezés. Forrás: Infoterra
Astrium GEO-Information Services Hungary ©

2.2. Nyers adatok vizualizációja

A rendelkezésre álló jellemzőket figyelembe véve, több módon is vizualizálhatjuk a pontfelhőt, amihez számos ingyenesen letölthető megjelenítő szoftver áll rendelkezésre. A 2.2 - 2.4 ábrák generálásához a Fugroviewert szoftvert ² használtuk. A 2.2 ábrán egy városrészletet megjelenítő LiDAR pontthalmaz látható, amely a magasságértékek és a lézervény intenzitásának együttes függvényében rendelünk színértékeket a térbeli pontokhoz: ezáltal a helyszín geometriája és textúrája együttesen jelenik meg. A 2.3 ábrán elhagyjuk az intenzitás komponenst és csupán a pontok magasság koordinátájának függvényében rendelünk színértékeket a pontokhoz.

A megjelenítés során a Fugroviewer szoftverben lehetőség van, hogy a nyers háromdimenziós adathalmazt poligonokkal közelítsük. A 2.4 ábrán a Budapesti Műszaki

²ingyenesen letölthető: <http://www.fugroviewer.com>

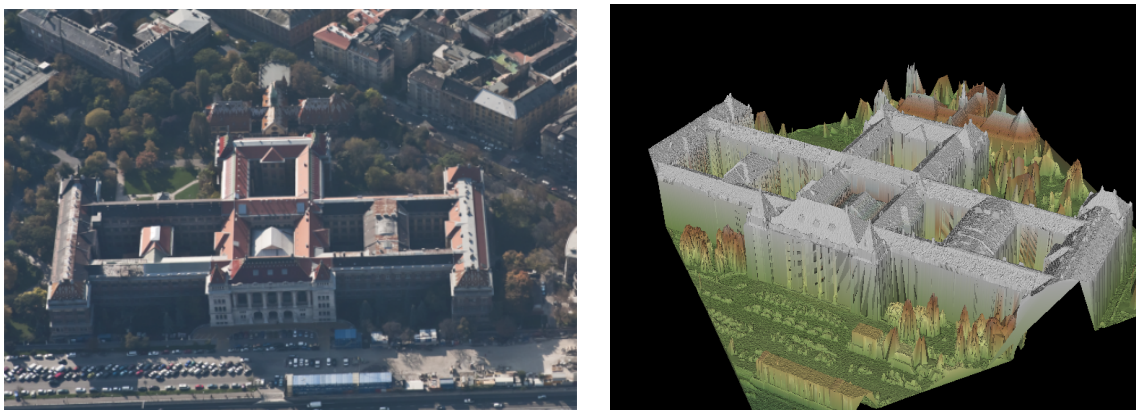


2.3. ábra. Magasságérték szerinti színezés. Forrás: Infoterra Astrium GEO-Information Services Hungary ©

és Gazdaságtudományi Egyetem főépületének nyers poligon approximációja látható. Ugyanakkor ezek a kizárólag megjelenítést végző szoftverek nem tudják elkülöníteni a pontfelhőn lévő objektumokat és a mérési zajt se tudják hatékonyan szűrni. Így nem jutunk szemantikai információhoz a helyszínnel kapcsolatban, valamint az eredményül kapott poligon „rekonstrukció” vizuálisan is gyenge minőségű. TDK munkánk során éppen az itt hiányzó funkciók megvalósítására fektettünk hangsúlyt.

2.3. Fejlesztőeszközök

Több elérhető függvénykönyvtár létezik, ami kifejezetten háromdimenziós pontfelhők hatékony feldolgozására és manipulálására készült. A feladat számítás- és memóriaigényes jellege miatt ezek a könyvtárak többségében C++ nyelvűek, bár kisebb adathalmazok kezelésére előfordulnak Matlab alapú megvalósítások is. Ugyanakkor a



2.4. ábra. Balra: Légkép a BME központi épületéről. Jobbra: A felület 3D nyers poligon approximációja LiDAR mérések alapján. Látható, hogy a nyers közelítés vizuálisan is rossz minőséget ad. Forrás: Infoterra Astrium GEO-Information Services Hungary©

feladatok sokszínűsége miatt az elérhető programkönyvtárak sem tartalmaznak teljes működő funkciókat, inkább alapvető építőköveket nyújtanak magasabb szintű alkalmazások megvalósításához.

Tapasztalataink szerint, a különböző függvénykönyvtáraknak számos előnye és emellett persze sok hiányossága is van, attól függően, hogy éppen milyen feladat elvégzésére alkalmazzák. A kutatómunkánk során kipróbált és felhasznált függvénykönyvtárak a következők:

- libLAS:

Kifejezetten a térinformatikában alkalmazott programozási környezet. Alapvető feladatai közé tartozik, a „.las” (laser) fájlformátumban tárolt adatok írása és olvasása. A „las” sajátosságai között megemlítendő, hogy egy szekvenciális bináris fájlformátum, amit kifejezetten szenzorokból történő adattárolásra fej-

lesztettek ki. A libLAS alapkönyvtára különböző interfészeket használ C++, Python és .Net alapú alkalmazásokhoz. Magához a könyvtárhoz hasznos parancssori eszköztár tartozik. Elvégezhető vele a pontfelhő adott térrészre történő vágása, illetve szűrése magasságkoordináták, intenzitásértékek, lézervisszaverődés vagy esetleg klasszifikáció szerint. A rendszer másik hasznos tulajdonsága, hogy lehetőség van a „las” formátum egyéb formátumokba történő konvertálására. A könyvtár hátránya, hogy az említett alapl műveleteken kívül nem rendelkezik más funkciókkal. Ez azt jelenti, hogy a pontfelhőn belüli objektumok értelmezését, szegmentálását segítő algoritmusokat a felhasználónak kell elkészítenie.

- PCL (Point Cloud Library): Az egyik legjobban használható és legnagyobb eszköztárral rendelkező keretrendszer. A PCL rendkívül új eszköz, az 1.0-ás verziója 2011 tavaszán jelent meg. Megtalálhatóak benne azok a 3D pontfelhő feldolgozással kapcsolatos alapfunkciók, amik hasznos építőelemként szolgálhatnak saját algoritmusaink elkészítésekor. A pontfelhő megjelenítésére, vizualizációjára szolgáló lehetőség - más könyvtárakkal ellentétben - itt jól megoldott funkció. Maga a megjelenítő VTK (Visualization ToolKit) alapokon nyugszik, azonban ez hatékonyan el van rejtve a programozó elől, így csak az absztrahált PCL specifikus függvényeket kell használni a megjelenítés érdekében, ami kényelmesen programozható. A vizualizációs fázis kód szinten beépíthető az algoritmusba, nincs szükség egyéb szoftverekre a pontfelhő megjelenítéséhez. A keretrendszer moduláris szerkezetű, akár szeparálva is lefordítható és használható, ezzel rengeteg erőforrást és tárhelyet spórolva meg a felhasználónak. További

előnyei közé tartozik a platform függetlenség: használható Linux, MacOS, illetve Windows operációs rendszerek alatt. Külön említésre méltó még, hogy a keretrendszerben lévő összes alapfunkciónak létezik egy GPU-ra párhuzamosított verziója is, amit CUDA SDK-val van lehetőség programozni. Erre támaszkodva akár valós idejű felismerési, detektálási, szegmentálási feladatok is megoldhatók a PCL segítségével.

- CGAL (Computational Geometry Algorithms Library):

Ez a nyíltforrású függvénygyűjtemény a két, illetve háromdimenziós környezetben történő geometriai algoritmusok végrehajtására alkalmas. A CGAL-t nagyon hatékonyan használják, a számítógépes látással, számítógéppel segített tervezéssel, molekuláris biológiával, orvosi képfeldolgozással, robotikával és numerikus módszerekkel foglalkozó szakemberek. A CGAL előnye, hogy nagyon kiforrottan működnek benne a geometriával kapcsolatos adatstruktúrák, mint például a Delaunay háromszögelés, Voronoi diagrammok, poligonok, felületek, rácshálók stb. A rendszerrel dolgozó szakember ennél fogva nagyon könnyen tud implementálni minden ebből a tárgykörből eredeztethető problémát, vagy épp annak a megoldását. Nemcsak adatstruktúrák, de kész alapfunkciók is megtalálhatóak a CGAL-ban. Ezek az algoritmusok a következő nagy csoportokba oszthatók: konvex burkolók, rácsháló generálás, geometrikus feldolgozás, térbeli keresés és rendezés, geometrikus optimalizálás, interpoláció és megjelenítés.

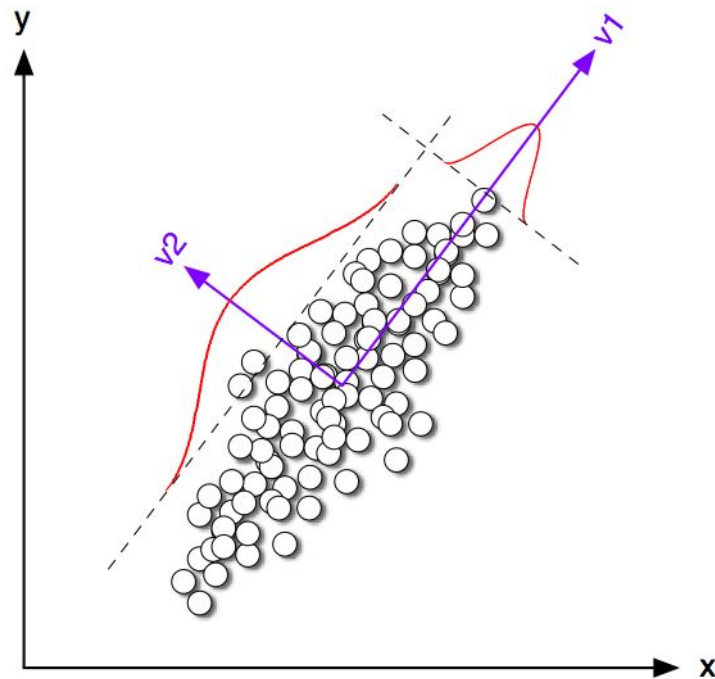
3. fejezet

Felhasznált matematikai eszközök és algoritmusok

Ebben a fejezetben a kutatómunkánk során felhasznált matematikai és algoritmikus háttér ismertetésére kerül sor. A felsorolt algoritmusok mind releváns eszközként szolgáltak a saját eljárásaink kifejlesztésében.

3.1. Főkomponens analízis (*PCA*)

A főkomponens analízis egy alapvető statisztikai eljárás adattömörítésre és dimenzió csökkentésre. Az adattömörítést az adatok hatékonyabb reprezentációjához használhatjuk, de alkalmazásával az is lehet a célunk, hogy az eredeti adatokból a későbbi feldolgozás szempontjából lényeges információt kiemeljük, míg a lényegtelen elhagyjuk [4]. Erre a célra lineáris vagy nemlineáris transzformációkat használhatunk ami az adott többdimenziós teret, egy kisebb dimenziós altérbe transzformálja. Felismerési és osztályozási feladatot megelőző lényegkiemelés akkor tekinthető hatékonynak, ha



3.1. ábra. A főkomponensek elhelyezkedése a legnagyobb szórások értelmében

az alter dimenziója minél kisebb, miközben a közelítő reprezentációban mindazon információ megmarad ami a felismeréshez és osztályozáshoz elegendő.

A főkomponens analízisnek kitüntetett szerepe van a lineáris adattömörítő eljárások körében. Működése során végrehajt egy ortogonális transzformációt, amely során egy N -dimenziós ponttérből kiindulva, kiválaszthatók a legfontosabb $M < N$ bázisvektorok, amelyek egy M -dimenziós alteret határoznak meg. Ahogy a 3.1 ábra¹ is mutatja, az alterbe transzformált bázisvektorok, vagy másnéven főkomponensek mentén jóval nagyobb tartományban szóródnak a mintapontok, mint az eredeti bázisrendszerben, tehát az egyes pontok közötti különbséget a transzformált pontok altere jobban tükrözi.

¹kép forrása: <http://web.media.mit.edu/~tristan/phd/dissertation>

3.2. Területelárasztó (*floodfill*) algoritmus

A területelárasztó (*floodfill*) algoritmus a 2D-s és 3D-s adatelemzés egyik közkedvelt eszköze. Az algoritmus jó célt szolgál olyan jellegű feladatoknál, amikor valamilyen tulajdonság alapján szeretnénk pontokat (vagy voxeleket) egy csoportba sorolni. Ha gyakorlatias szempontból szeretnénk megközelíteni ezt a kérdéskört, akkor tipikusan egy objektum szegmentálási, vagy klaszterezési feladatra gondolhatunk egy kétdimenziós képen, vagy egy csoportosításra 3D-s pontfelhőben jellemző térbeli szomszédosság alapján. Az eljárás maga arra az ötletre épül, hogy klasszifikáljuk a pontokat, amelyek egy adott kezdeti pontból (magból), a valamilyen módon definiált határ átlépése nélkül elérhetők. Ebből következik tehát, hogy szükségünk van egy feltételre ami megmondja, hogy egy adott pont beletartozik-e az adott elárasztandó területbe, vagy éppen határpont és kívül esik rajta. A pontok összetartozási feltételeként különböző megkötések deffiniálhatók. Például az azonos területhez tartozó pontoknak azonos színűnek kell lenniük, vagy háromdimenziós esetben a voxelek közötti euklidészi távolság ne haladjon egy bizonyos értéket, esetleg a voxelek normálvektorainak az orientációja ne haladjon meg egy bizonyos szórást. A kitöltő algoritmusnak ezek után két feladata van. Ellenőrzi, hogy a kurrens pont belső pont-e és ha igen, akkor kitölti. Másrészt, ha a pont belső pont, akkor a szomszéd pontok is potenciális belső pontok, így azokat ugyanúgy ellenőrizni kell és szükségszerűen kitölteni [18]. A *floodfill* algoritmus részletesebb működését az alábbi leírás szemlélteti.

Algorithm 1 *Floodfill* algoritmus működése

P : Pontfelhő

C : Üres klaszterek listájának beállítása

Q : a vizsgálandó pontok sorozatának inicializálása

while Minden $p_i \in P$ esetén, minden $p_i \in C$ **do**

for all $P_i \in P$ **do**

$Q \leftarrow P_i$

for all $P_i \in Q$ **do**

if p_i pont r sugarú környezete $<$ threshold **then**

p_i pont p_i^k szomszédjainak megkeresése

end if

 Minden $p_i^k \in P_i^k$ esetén a még nem feldolgozott pontok Q sorozatba tétele

end for

end for

end while

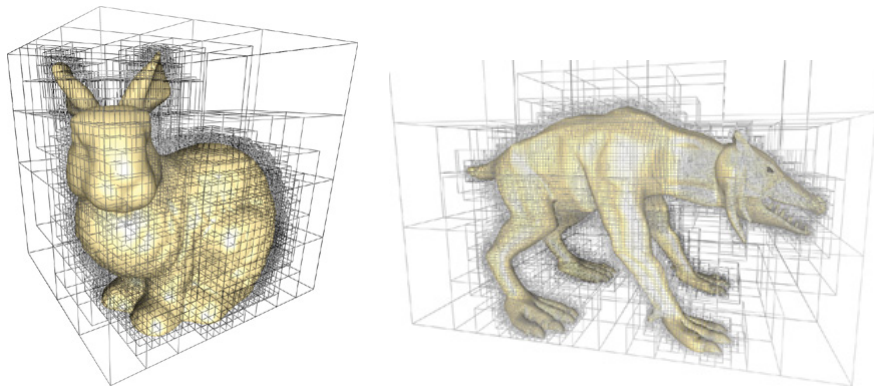
3.3. Térparticionáló módszerek

Tekintve, hogy a TDK munkánk során elkészült algoritmusok háromdimenziós adathalmazokból nyernek ki jellemzőket, majd ezen jellemzők alapján végeznek el különböző feladatokat, így elengedhetetlen, hogy nagyon gyorsan tudjunk lokális pontszomszédossági viszonyokat felállítani a 3D-s környezetben, továbbá szükségszerű, hogy gyorsan keressünk vissza adott pontokat a pontfelhőben. A hagyományos térparticionáló adatstruktúrák működése során egy szabályos tér-rácsot helyeznek az adatra és minden pontot regisztrálnak a hozzá tartozó kockából. Ez a módszer olyan esetekben működik hatékonyan, amikor a térben lévő pontok sűrűsége közel permanens. A változó sűrűségű adathalmazoknál - mint például egy valós mérésből származó 3D LiDAR pontfelhő - ezek az eljárások nem bizonyulnak hatékonyak. A hagyományos szabályos felosztás és az állandó élhosszúságú kockák, amik a tér particionálására szolgálnak ebben az esetben nem elegendőek. Mint ahogy a 3.2 ábrán² is látható, olyan eszközöket célszerű alkalmazni, amik adaptívan tudják követni a térben lezajló sűrűség változásokat és csak ott növelik a particionálás felbontását, ahol ténylegesen van adat, míg az üres részeket figyelmen kívül hagyják, ezzel számítási időt és teljesítményt spórolva [19].

3.3.1. Oktális fa

Az egyik térparticionáló algoritmust - ami teljesíti a fent említett követelményeket - oktális fastruktúrával lehet előállítani. A tér szabályos felosztása annak érdekében, hogy elérjük és hivatkozzunk a feldolgozandó adatokra sok cellalépést igényel. Az

²képek forrása: <http://http.developer.nvidia.com/GPUGems2>

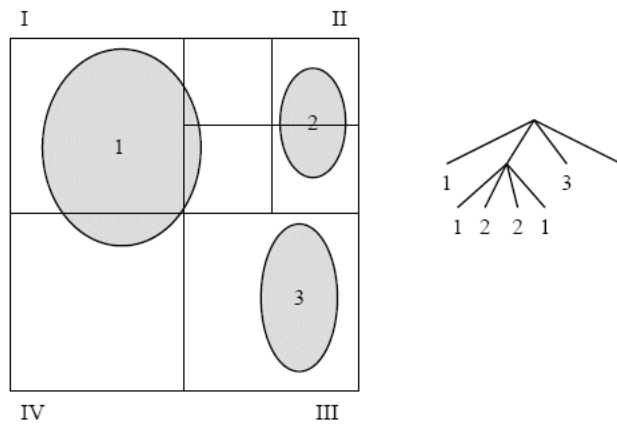


3.2. ábra. 3D térpartícionálás oktális fával

üres térrészek megvizsgálásával nem érdemes sok időt eltölteni, hiszen nincs bennük releváns információ, továbbá két szomszédos cellát is elég lenne akkor szétválasztani, ha azok mindegyike még sok pontot tartalmaz. Ezt a felismerést teszik magukévá az adaptív felosztó algoritmusok, azon belül pedig az oktális fák. A ponttér adaptív felosztása rekurzív megközelítéssel és egy hierarchikus adatszerkezet felépítésével lehetséges. A hierarchikus szerkezet általában egy fa, az adaptív algoritmusokat pedig a fa típusa szerint osztályozzuk. Az oktális fa azt a szemléletmódot követi, hogy a térben lévő nem üres csomópontoknak nyolc gyereke van.

A fastruktúra felépítésének lépései a következők:

1. Kezdetben foglaljuk a teljes ponthalmazt egy a koordinátatengelyekkel párhuzamos oldalú dobozba, ami az oktális fa gyökere lesz és egyben a rekurzió kiindulópontja, azaz az első feldolgozandó cella.
2. Ha az aktuális cellában található pontok száma nagyobb, mint egy előre definiált érték, akkor a cellát a felezősíkjai mentén 8 egybevágó részcellára bontjuk. A hierarchikus adatszerkezetben a részcellákat a felbontott cella gyerekeiként



3.3. ábra. Az négyes fa működése, (oktális fa 2D-s esetben)

vesszük fel, majd a keletkező részcellákra ugyanazt a lépést rekurzívan megismétljük.

3. A gráfépítő folyamat egy adott szinten megáll, ha az adott cellához vezető út elér egy előre definiált maximális mélységét, azaz a cellaméret egy adott szint alá csökken, vagy az adott cellában az pontok száma egy előre definiált érték alá esik. Az eljárás eredménye egy oktális fa.

A 3.3 ábrán látható a síkot felosztó négyes fa, amelynek a háromdimenziós változata az oktális fa. A felépítés során a cella oldalainak a felezését addig folytatjuk, amíg egy cellába kevés pont jut, vagy a cellaméret egy megadott minimális érték alá csökken. A fa leveleiben regisztráljuk a hozzájuk tartozó pontokat.

3.3.2. Kd-fa

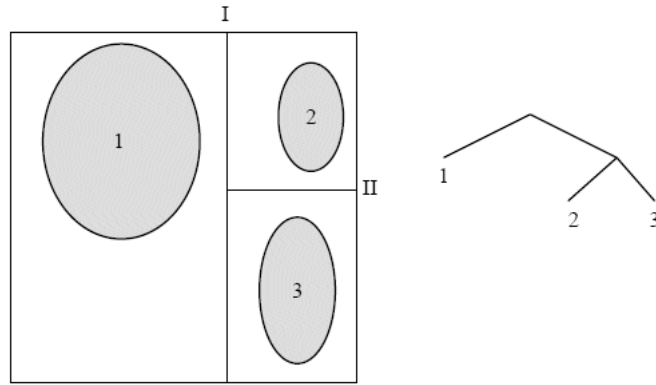
Az oktális fa egy jól működő eszköz, a felosztás azonban még mindig felezi a cellaoldalakat és ezen belül nem veszi figyelembe, hogy a pontok hol helyezkednek el, így

az adaptivitás nem tökéletes. Tekintsünk egy olyan felosztást, amely egy lépésben nem mind a három felezősík mentén vág, hanem egy olyan sík mentén, amely a pontteret a lehető legigazságosabban felezi meg. Ez a módszer egy bináris fához vezet, amelynek neve bináris tértarticionáló fa. Ha a felezősík mindig merőleges a koordinátarendszer valamely tengelyére, akkor kd-fa adatszerkezetről beszélünk, ahol a „kd” a k-dimenziós tulajdonságot jelöli. A 3.4 ábra szerint, a kd-fa alaplépése, hogy sok pontot tartalmazó cellát rekurzívan egy valamely koordinátatengelyre merőleges síkkal két cellára bontjuk.

A kd-fában a felezősík elhelyezésére többféle módszert is alkalmazhatunk:

- A térbeli középvonal módszer a befoglaló keretet mindig két egyforma részre osztja.
- A test középvonal módszer úgy osztja fel a teret, hogy annak bal és jobb oldalán egyforma számú pont legyen.
- A költségvezérelt módszer becsli azt az átlagos időt, amelyet egy sugár a kd-fa bejárása során felhasznál és ennek minimalizálására törekszik. Egy megfelelő költségmodell szerint úgy felezzük a cellát, hogy a metszés ugyanakkora valószínűséggel következzen be a gyerek cellákban. A metszési valószínűség kiszámításához az integrálgeometria egyik alapvető tételét alkalmazhatjuk:

Tétel: Ha egy konvex A test tartalmaz egy ugyancsak konvex B testet, akkor annak valószínűsége, hogy egy egyenletes eloszlású véletlen egyenes metszi a B testet feltéve, hogy metszette az A -t, egyenlő a B és az A testek felületeinek arányával.



3.4. ábra. A Kd-fa felépítése

A következőkben egy általános kd-fa építő rekurzív algoritmus kerül bemutatásra. A cella paraméter az aktuális cellát, a mélység a rekurzió mélységét, a koordináta pedig az aktuális vágósík orientációját jelenti. A cellához a két gyerekcella (`cella.jobb`, illetve `cella.bal`) és a bal alsó közeli, illetve jobb felső távoli sarokpontok (`cella.min`, illetve `cella.max`) tartoznak. A cellákhoz rendeljük hozzá a belógó pontok listáját. A felezősík irányát a fa építésekor a mélység növekedésével $k > 0$ esetén az $F_{k+1}(x, y, z) = F_k(x, y, z)$ rekurzív függvény változtatja meg. Az 3.4 ábrán látható rekurzív eljárás első hívásakor a cella a teljes pontteret tartalmazó doboz, a fa mélysége pedig zérus.

3.4. **RANSAC** eljárás

A *RANSAC* (*RANdom SAmple Consensus*) algoritmust számos gépi látáshoz és grafikához kapcsolódó alkalmazásban használják. Az eljárás egy optimalizálási folyamatnak tekinthető, amely jól alkalmazható többek között egyenesek és síkok illesztésére két-, illetve háromdimenziós adathalmazokhoz. Az eljárás előnye, hogy kevesebb iteráció szükséges a működéséhez, mint a hasonló célokat megvalósító, legkisebb

Algorithm 2 Kd-fa felépítése

```
if a cellába lógó pontok száma kevés, vagy a mélység nagy then
    return
end if
cella.bal és cella.jobb befoglalódoboza  $\leftarrow$  cella befoglalódoboza
if koordináta == x then
    cella.jobb.min.x  $\leftarrow$  cella felezősíkja x irányában
    cella.bal.max.x  $\leftarrow$  cella felezősíkja x irányában
else if koordináta == y then
    cella.jobb.min.y  $\leftarrow$  cella felezősíkja y irányban
    cella.bal.max.y  $\leftarrow$  cella felezősíkja y irányban
else if koordináta == z then
    cella.jobb.min.z  $\leftarrow$  cella felezősíkja z irányban
    cella.bal.max.z  $\leftarrow$  cella felezősíkja z irányban
end if
for all a cella o pontjaira do
    if az o pont a cella.bal befoglaló dobozában van then
        adjuk az o pontot a cella.bal listájához
    end if
    if az o pontot a cella.jobb befoglaló dobozában van then
        adjuk az o pontot a cella.jobb listájához
    end if
end for
mélység  $\leftarrow$  mélység +1
koordináta  $\leftarrow$  következő koordináta
Rekurzió:Kd-fa felépítése()
```

négyszetek módszerén alapuló algoritmusok [3]. Ebből az következik, hogy jobb hatékonysággal alkalmazható nagy adathalmazokban történő síkdetektálásra, ahol az eljárás iteráció száma kritikus a gyors feldolgozás érdekében. A módszer működése a következő: véletlenszerűen kiválaszt három pontot az adathalmazból és kiszámítja hozzá tartozó sík paramétereit. Ezután tartományra vágással meghatároz egy küszöböt a kiszámított sík és a teljes adathalmaz pontjainak távolságára vonatkozólag. Ezeket a lépéseket ismétli N alkalommal, minden lépésben összehasonlítva az eredményeket, a legutóbbi elmentetttel [24]. Ha az új eredmény jobb mint az elmentett, akkor kicseréli rá. Három dimenziós esetben az algoritmus a következő bemeneteket fogadja:

- 3D pontfelhő: X, Y, Z struktúrát magában foglaló pontlista.
- Tolerancia küszöb: A választott sík és a pontfelhőben lévő többi pont közötti t távolság függvényében értelmezhető. Ez az érték a pontfelhő felbontásától és regularitásától függ.
- Az előrelátható támogatás (*forseeable-support*): megmondja azoknak a pontoknak a lehetséges maximális számát, amik egy síkhoz tartoznak. Pontsűrűség alapján becsülhető.
- α valószínűség: annak a minimális valószínűsége, hogy legalább egy helyes észlelés lesz N darab próbából. Ez az érték optimálisan 0,90 és 0,99 között vesz fel értéket.

Algorithm 3 RANSAC eljárás működése

$bestSupport = 0$

$bestPlane(3, 1) = [0, 0, 0]$

$bestStd = \infty$ - Best standard deviation

$i = 0$

$\epsilon = 1 - \text{forseeable-support}/\text{length}(\text{pont-list})$ - hibás megfigyelések megengedett százaléka

$N = \text{round}(\log(1 - \alpha)/\log(1 - (1 - \epsilon)^3))$

while $i \leq N$ **do**

j = véletlenszerűen válasszunk ki 3 pontot a (pont listából)

$texts/pl = pts2plane(j)$ - kiszámolja a 3 kiválasztott pontra a sík paramétereit

$dis = dist2plane(pl, \text{point-list})$ - kiszámolja a távolságot a megadott pontok és a sík között

$s = \text{find}(\text{abs}(dis) \leq t)$

if $\text{length}(s) > bestSupport$ vagy $(\text{length}(s) == bestSupport \text{ és } st < bestStd)$

then

$bestSupport = \text{length}(s)$

$bestPlane = pl$

$bestStd = st$

end if

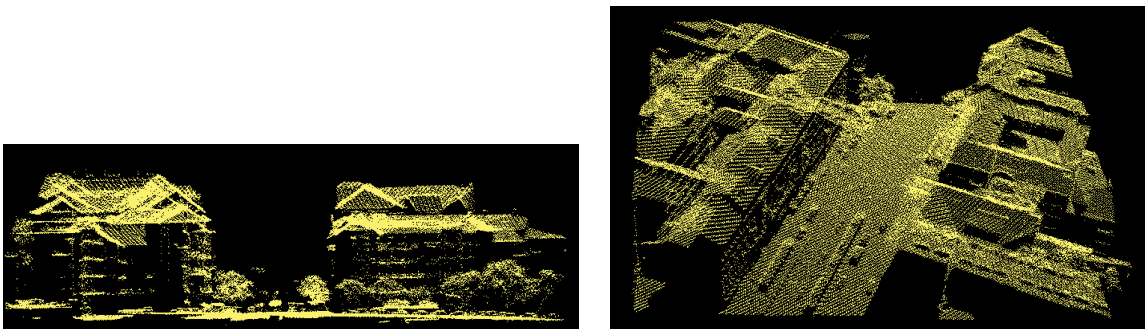
$i = i + 1$

end while

4. fejezet

Elkészített pontthalmaz-elemző rendszer bemutatása

Ebben a fejezetben a kutatómunkánk során kifejlesztett saját algoritmusaink kerülnek bemutatásra. Az eljárások háromdimenziós pontfelhőn végeznek objektum felismerési, illetve rekonstruálási feladatokat kifejezetten városi környezetben. A feladat megvalósításával kapcsolatos kihívásokat az okozza, hogy a bemeneti adat sokszor zajjal terhelt. Mint ahogy a 4.1 ábrán is látható a feldolgozás a valós életből származó, sok esetben alacsony felbontású adatokon történik. A mérés során keletkező pontfelhők



4.1. ábra. Nyers, változó pontsűrűségű és struktúrálatlan bemenet

sűrűsége mérésenként és egy mérésen belül is változó lehet. A tesztekhez használt adatokat az Infoterra Astrium GEO-Information Services © szolgáltatta, Budapest belvárosáról készült légi LiDAR méréseinek eredményeként, melynek átlagos pontsűrűsége (felbontása): $8 \text{ pont}/m^2$.

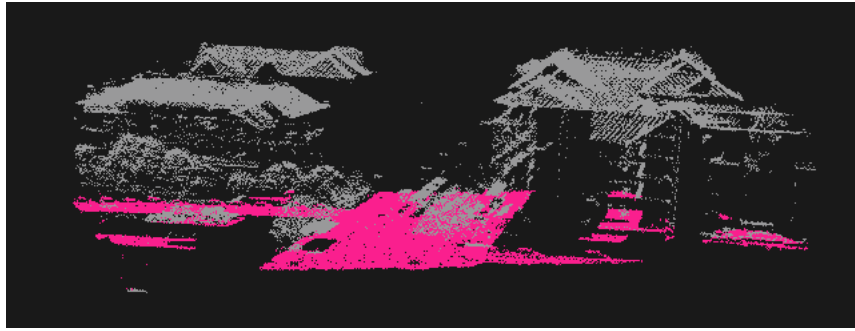
4.1. Pontfelhő klasszifikálás

Az objektumok és objektum csoportok szemantikai értelmezése és felismerése egy dinamikusan fejlődő kutatási irányvonal a háromdimenziós távérzékelésben. Ezekre a funkciókra számos felügyeleti rendszerben szükség van és a virtualizáció alapját is képezhetik. A 2. fejezetben bemutatott vizualizációs eszközök csupán a mérési adatok megjelenítésére alkalmasak, viszont a pontfelhő tartalmi analízise és az alakzatok rekonstrukciója nem valósítható meg ezekkel az eszközökkel. A következő alfejezetekben bemutatásra kerül, hogy felismerő algoritmusaink milyen objektumok, illetve objektum csoportok felismerésére képesek városi környezetben.

4.1.1. A talaj síkjának szegmentálása

Az adatok feldolgozásának első lépése a talaj síkjának detektálása és a talajhoz tartozó pontok elkülönítése a pontfelhőben. Ez a lépés azért lényeges, mert nélkül nem lenne lehetséges más régiók és objektumok felismerése az adathalmazban. A talaj síkjának detektálására különböző eszközök állnak a rendelkezésünkre. Egy viszonylag egyszerű módszer, ha a magasság koordinátákat vizsgálva, egy bizonyos küszöb (amit jelöljön: $thres$) alatt, a mérési pontokat talajpontoknak, míg a küszöb fölött egyéb pontoknak tekintjük: Legyen $P = \{p_1, p_2, \dots, p_n\}$ LiDAR mérési halmaz,

mely során p_i kurrens mérési pont talajpont, ha $p_i(z) < thres$, egyébként nem talajpont. Ennek a módszernek előnye az, hogy egyszerű eljárás és könnyű megvalósítani, viszont ellene szólnak a következők: A különböző adathalmazokban a talaj szintje változó, így a $thres$ értéket mindig az adott bemenethez kell igazítani. További hátráltató tényező, hogy a küszöbölős módszer vízszintes talajsíkot feltételez, ami nem feltétlenül teljesül a gyakorlatban, ahol a magasságértékeknek gyakran erő szórása van. A megközelítés finomítására léteznek különböző megoldások [23]. A talajpontok szórásából adódó hiányos talaj detekciókat ki lehet küszöbölni azzal, ha kis területeken vizsgálunk a talajpontokból egy bizonyos magasságérték tartományban, majd egy medián szűréssel eldobjuk azokat a pontokat, akik szélsőséges magasságértékkel rendelkeznek, vagyis nem a talaj részei, hanem a felette lévő zaj, vagy egyéb régió. Munkánk során mi egy másik megoldást, a 3.4-es fejezetben bemutatott RANSAC eljárást használtuk a talaj síkjának detektálására. Az eljárás pontosabb mint a hagyományos küszöbérték alapján történő szegmentálás és gyorsabb mint a medián alapú, lokális magasságértékeket vizsgáló algoritmusok, mivel nincs benne semmiféle rendezési művelet. Megjegyezzük, hogy ez az eljárás sem teljesen adathalmaz független. Szükségszerűen ebben az esetben is be kell állítani egy paramétert az algoritmus hatékony működése érdekében. Ez a paraméter a sík optimalizálás toleranciája, vagyis, hogy az ajánlott domináns sík és az adathalmaz pontjai között mekkora legyen a távolságértékek tűrése. Ugyanakkor a tűrési paraméterek egységesnek tekinthetők állandó feltételek között készített nagy méretű adathalmazokon is, ezért a fent említett küszöbölős módszernél robosztusabb eljárás. Tapasztalataink szerint a RANSAC algoritmus egy hatékony eszköz a talaj detektálására, azonban az adathalmazban lévő egyéb változó méretű és orientációjú síkok detektálásra (háztetők) nem megfelelő,



4.2. ábra. A talaj síkjának detektálása RANSAC algoritmussal

hiszen az optimalizálás paramétereit minden méretű síkhoz meg kéne határozni. Ahogy a 4.2 ábrán is látható, nagyméretű és nagyságrendileg azonos orientációjú sík (talaj sík) detektálására viszont kiválóan alkalmas.

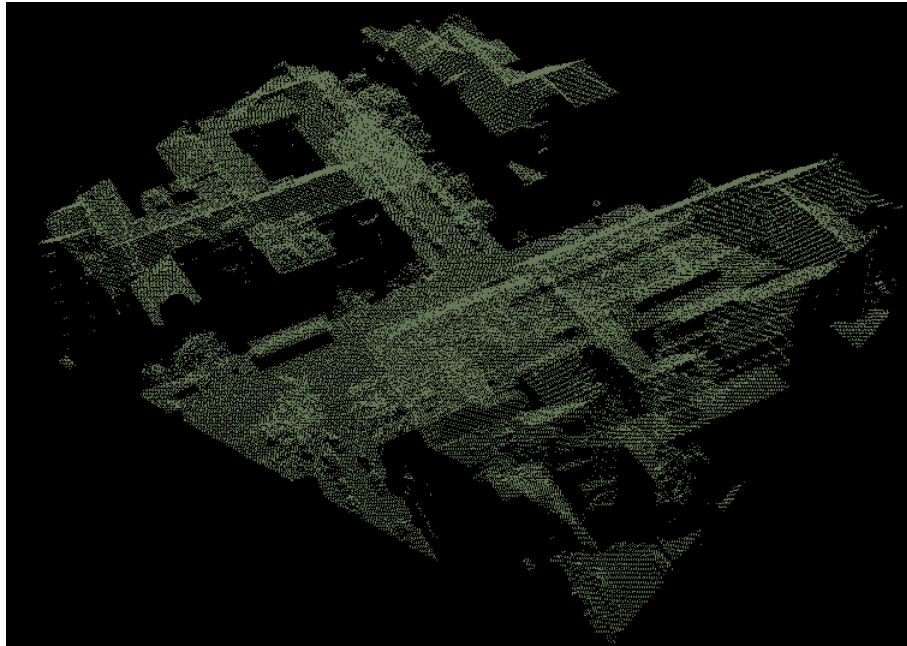
4.1.2. Növényzet és ritka ponthalmazok felismerése

A növényzet és az alacsony felbontású régiók detektálására kifejlesztett algoritmusunknak két fő lépése van.

- térbeli partícionálás a sűrű térrészek lokalizálása érdekében
- lézernyalábok visszaverődési számának alapján történő növényzet detektálás

Első lépésként érdemes részekre osztani az adathalmaz, mielőtt növényzetet lokalizálunk. Ennek több oka is van. Elsősorban az adathalmazok mérete miatt a kisebb egységek könnyebben kezelhetők egy lépésben. Másodsorban mielőtt vegetációt detektálnánk, meg kell keresnünk a térben azokat a térrészeket amik ténylegesen sűrű, jó felbontású adattal szolgálnak, míg az ezen kívül eső területeket zajnak, vagy ritka (további feldolgozás szempontjából használhatatlan) adatnak értelmezzük, ugyanis a ritka területekről kinyert statisztikai jellemzők félrevezetőek lehetnek. Ennek

érdekében egy tértarticionálást alkalmaztunk, ami a háromdimenziós adathalmazt, kis térrészekre pontja fel [13],[21]: $P = \cup P_i$, ahol P_i egy adott térrész. $\forall P_i, i = \{0, 2, \dots, n - 1\}$ esetén vizsgáljuk meg a térrész sűrűségét a következő módon. P_i sűrű, akkor és csak akkor, ha (i) $\forall p \in P_i$ -hez $\exists q \in P_i$ amire teljesül, hogy $\forall p \exists q \in P_i$ pontpárra vizsgált távolság $\|p - q\| \leq t$, ahol $t \in \mathbb{R}^+$ egy küszöbérték. A sűrűségre vonatkozó másik követelmény, hogy (ii) P_i tartalmazzon legalább N pontot. Ezt követően, a további feldolgozás előtt leválasztjuk a ritka régiókban megfigyelt zajt a bementi adatról. A ritka térrészekre egy tipikus példát mutatnak az általunk használt városi tesztadatokon az épületek falairól visszaverődött lézer mérések. Ezekben az esetekben a repülőgépre szerelt LiDAR eszköz, nem volt olyan betekintési szögben, ami optimális környezetet biztosított volna, az épületek oldalfalairól történő megfelelő felbontású felvételek készítésére, így a kész pontfelhőben az oldalfalak nem megfelelő sűrűségű régióként jelennek meg, amit igyekszünk kiszűrni. A sűrű térrészek lokalizálását követően, elkezdhetjük vizsgálni a LiDAR eszköz által kibocsájtott járulékos jellemzőket. Ilyen jellemző például a lézerfény intenzitásának a mértéke, vagy a lézernyalábok visszaverődésének a száma. A növényzet és a fák lombjainak irregurális elhelyezkedése azt eredményezi, hogy a kibocsájtott lézerfény megtörik rajta, és több példányban verődik vissza. Ahogy az 1.1-es fejezetben is említettük, a lézerfény visszaverődések számának vizsgálatával, egyértelműen el lehet különíteni a vegetációt az épületektől, hiszen ahol többszörös visszaverődés volt az nagy valószínűséggel valamilyen fás, bokros terület, míg az egyszeres visszaverődések nagy valószínűséggel sík felületekről, házakról származnak. A 4.3 ábrán a bementi pontfelhő látható, míg a 4.4 ábrán az algoritmus kimenete figyelhető meg. A fák és növényzet zöld, a sík felületek piros, míg a zaj és ritka részhalmazok kék színnel lettek

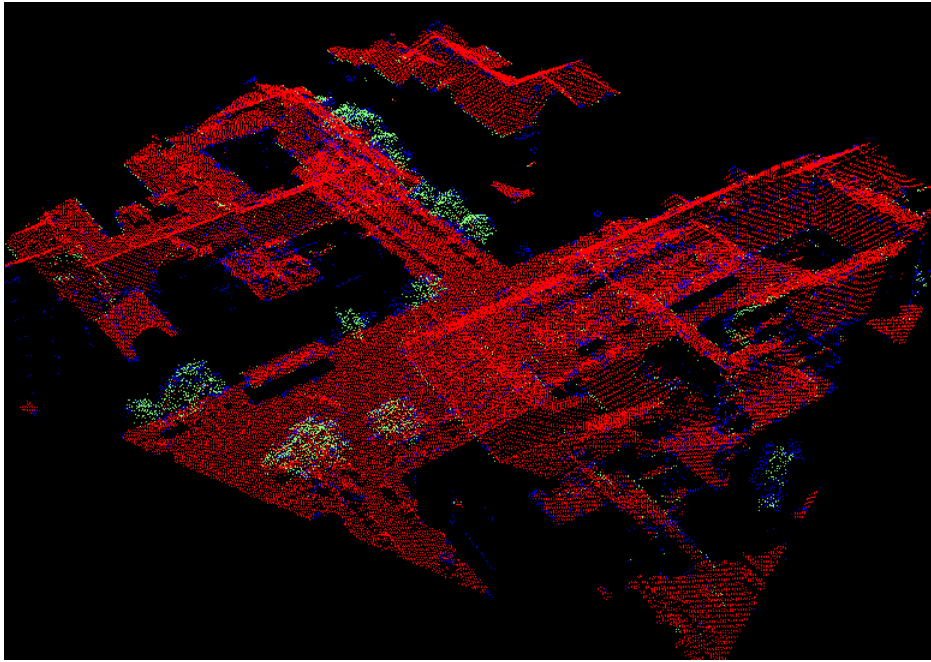


4.3. ábra. Bemeneti nyers pontfelhő

jelölve.

4.1.3. Épületek háztetőinek detektálása

Városi környezetben készült 3D-s távérzékelte adat kapcsán felmerülhet igény, az adathalmazban lévő épületek háztetőinek és a háztetőket felépítő háztetőszegmensek pontos lokalizációjára és felismerésére. Mint említettük a 2.2-es fejezetben az adathalmaz megjelenítésére szolgáló szoftverek nem képesek szemantikai értelmezésre, csupán a vizualizációt hajtják végre és azt is zajosan. Algoritmusaink két fő művelet megvalósítására alkalmasak a háztetődetektálást illetően. Képesek a különálló házak egész tetőszerkezetét, mint egy egységet lokalizálni és detektálni. Továbbá képesek az egyes háztetőket alkotó tetőszegmensek pontos helyét és orientációját meghatározni. Ez fogja majd alapját képezni a 4.2 fejezetben bemutatásra kerülő épület rekonstrukciós



4.4. ábra. Növényzet és ritka részhalmazok detektálása

algorithmus működésének. Az eljárás a következő három lépésből áll:

- pontok normálvektorának a meghatározása főkomponens analízis (*PCA*) segítségével
- normálisok klaszterezése k-means eljárással, az azonos orientációjú síkok osztályozása érdekében
- k-means eljárás kimenetének tovább bontása kisebb klaszterekre területelárasztó (*floodfill*) algoritmus felhasználásával, a síkok pozíció szerinti klaszterezése érdekében.

Annak érdekében, hogy az azonos síkhoz tartozó pontokat pontosan lokalizálni tudjuk a térben, mindenekelőtt egy megfelelő leírót kell találnunk. Ez a leíró a háromdimenziós térben elhelyezkedő pont normálvektora, vagy normálisa. Amikor

a pontok egy adott orientációjú síkhoz tartoznak, akkor a síkra merőleges irányú becsült normálvektorok is közel lesznek egymáshoz. Az adathalmaz minden pontjára vonatkoztatott lokális normálisok előállítását főkomponens analízissel végeztük. Legyen p egy tetszőleges pont és jelölje P_p^k a k legközelebbi szomszédját tartalmazó környezetet: $P_p^k = \{p_1, p_2, \dots, p_k\}$. Az eljárás célja, hogy minden p -re, előállítsuk elő a pontfelhő p pontjához tartozó lokális normálvektorát, n -et. A feladat végrehajtásához a pont környezetében kovariancia analízist kell végeznünk, majd az adathalmaz főkomponenseit felhasználva meghatározzuk a normálvektort az alábbi lépésekben [11], [16].

Állítsuk elő az összes p pont P_p^k környezetében lévő $C_p \in \mathbb{R}^{3 \times 3}$ kovariancia mátrixot a következő képpen:

$$C_p = \frac{1}{k} \sum_{i=1}^k (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \quad (4.1)$$

$$C_p \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in \{0, 1, 2\} \quad (4.2)$$

A 4.1 egyenletben előállított C_p kovariancia mátrix egy szimmetrikus és pozitív szemidefinit mátrix. A $\bar{p}$ ezen adathalmaz esetén, a p pont P_p^k környezetében értelmezett átlagos koordináta értéket jelenti:

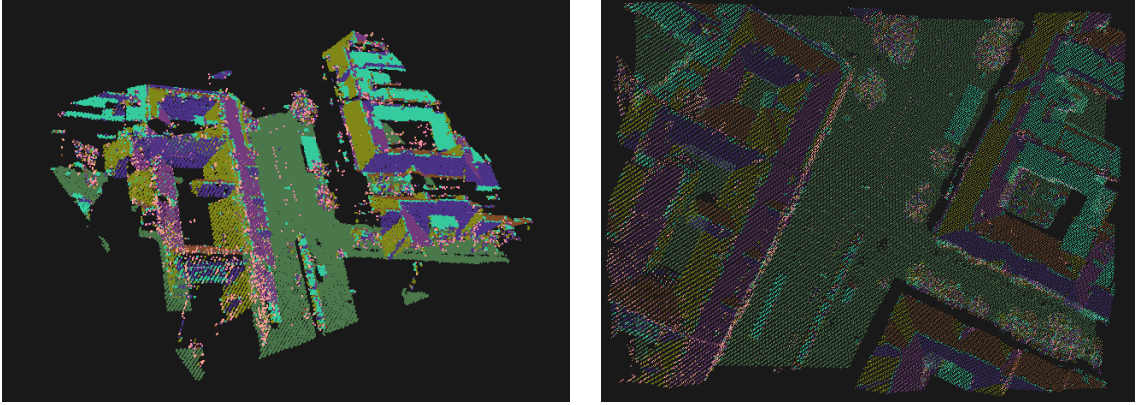
$$\bar{p} = \frac{1}{k} \sum_{i=1}^k p_i$$

A 4.2 formulával előáll C_p mátrix $\lambda_j \in \mathbb{R}$ sajátértékei. A P_p^k szomszédság főkomponenseit a $\vec{v}_j$ sajátvektorok tárolják. Az összes p ponthoz tartozó $\bar{n}$ normálvektort a $0 < \lambda_0 < \lambda_1 < \lambda_2$ sajátérték hármából tudjuk kinyerni. A λ_0 legkisebb sajátértékhez tartozó $\lambda_0 \rightarrow \vec{v}_0$ sajátvektor lesz, egy adott p pont normálvektora P_p^k környezetben.

Miután meghatároztuk a normálvektorokat minden pontra vonatkozólag, folytathatjuk az algoritmust a normálisok klasszifikálásával. Triviális, hogy ha egy pont-halmaz egy adott síkhoz tartozik akkor a normálvektoraik is azonos irányban fognak állni. Ezt az előismeretet felhasználva valósítottuk meg, a különböző tetőket alkotó szegmensek klasszifikációját az orientáció szerint *k-means* eljárással:

$$\Phi = \sum_{j=1}^k \sum_{i=1}^m (p_i - \bar{p}_j)^2 \quad (4.3)$$

ahol p_i a j -edik klaszterhez tartozó normálvektor, $\bar{p}_j$ a j -edik klaszterhez tartozó átlagos koordináta érték, k pedig a klaszterek száma. A *k-means* eljárás során, a pontokhoz tartozó normálvektorok kerülnek klasszifikálására. A folyamat lényege, hogy heurisztikusan 6 kitüntetett irányt feltételezünk a normálvektorok orientációjára. Megfigyeltük, hogy általában az adathalmazban lévő tetősíkok irányai maximum 6 különböző osztállyal jól közelíthetők. Ebből adódóan a *k-means* eljárás 6 klaszterbe fogja rendezni a normálisokat és a hozzá tartozó pontokat. Ez azt eredményezi, hogy az algoritmus lefutása után lesz 6 darab adathalmazunk és mindegyikben egy adott orientációjú tetősíkok lesznek eltárolva, a pontfelhő bármely területéről. Ez az eszköz alkalmas különböző orientációval rendelkező tetőszegmensek klasszifikálására, viszont hiányossága, hogy figyelmen kívül hagyja a tetőszegmensek pozícióját, hiszen egy klaszterbe az azonos orientációval rendelkező, de egymástól távoli síkrészletek is tarthatnak (4.5 ábra). Annak érdekében, hogy a tetőszegmenseket orientáció és pozíció szerint is külön tudjuk választani, vagyis ténylegesen egyenként tudjuk detektálni a városi pontfelhőn lévő házak tetőszegmenseit, a *k-means* eljárás klasztereit tovább kell bontani, hogy az azonos orientációjú, azonban különböző helyen lévő tetősíkokat



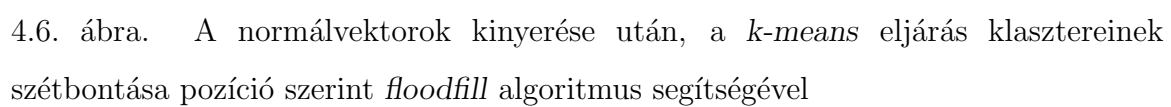
4.5. ábra. Azonos orientációjú tetősíkok klaszterezése *k-means* eljárással

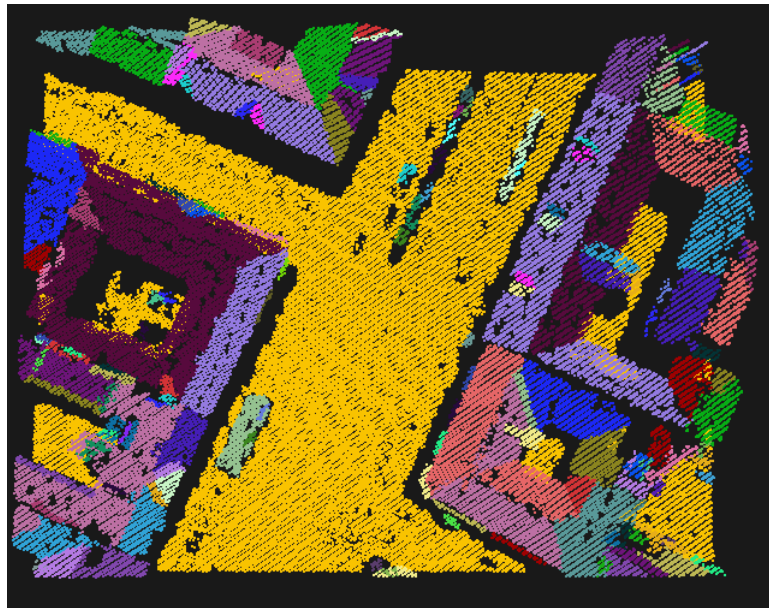
is szét tudjuk választani.

Legyen X a klaszterezendő teljes adathalmaz. A *k-means* eljárás klaszterei pedig legyenek $X = \cup_i Y_i$. Az algoritmus feladata, hogy *floodfill* eljárás segítségével tovább darabolja Y_i klasztereket a következőképpen: $Y_i = \cup_j Z_{ij}$ floodfill klaszter, hogy $\forall i_1, i_2 \in [1, k]$ esetén $Z_{i1} \cap Z_{i2} = \emptyset$

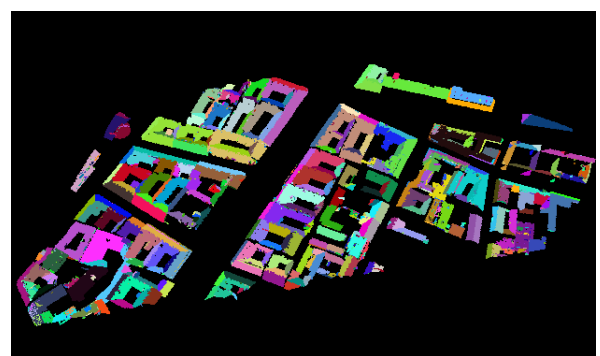
Az adatok tovább klaszterezésének eredménye, hogy az azonos orientációjú tetőszegmensek, a ponthalmaz különböző tartományaiból külön-külön klaszterekbe fognak tartozni, ami azt eredményezi, hogy a pontfelhőn lévő tetősíkok egyenként lokalizálhatók és felismerhetők orientáció és pozíció szerint is (4.6 ábra).

Algoritmusaink kimenetei a 4.7 és 4.8 ábrákon láthatók. A Budapesti Műszaki Egyetem főépületének tetőszegmens detekciójánál használt tesztadat több mint 1 millió, míg egy nagyobb budapesti városrészletről készült tesztadat több mint 2 millió mérési pontot tartalmaz. Megfigyelhető, hogy a 4.5 ábrán látottakkal ellentétben, itt nem az azonos orientációjú tetősíkoknak van egyforma színe, hanem, az összes tetőszegmensnek külön-külön van eltérő színe, ezzel is reprezentálendő, hogy külön-





4.7. ábra. Saját tetőszegmens detektor algoritmus kimenete egy budapesti városrészletre



4.8. ábra. Balra: A BME főépületének tetőszegmens detekciója. Jobbra: Egy nagyobb budapesti városrészlet tetőszegmens detekciója.

külön klaszterbe tartoznak, azaz az algoritmus alkalmas háztető síkok detekciójára és elkülönítésére orientáció és pozíció szerint.

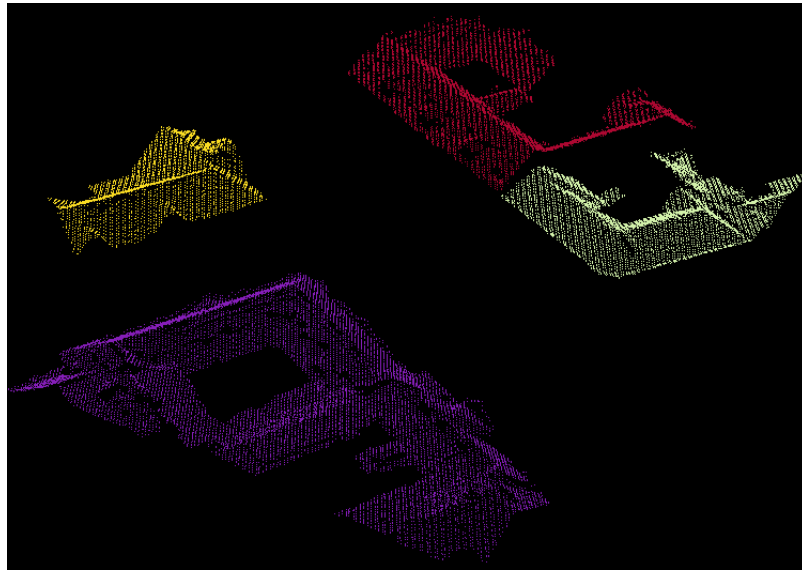
Munkánk során nem csak a tetőszegmensek detektálására szolgáló algoritmusokat fejlesztettünk és teszteltünk, hanem a házak egész tetőszerkezetét, mint egy önálló egységet detektálni képes algoritmust is megvalósítottunk. Az eljárás a következő három fő lépést követi:

- Talaj detektálása és leválasztása a pontfelhőből a 4.1.1 fejezetben bemutatott algoritmus segítségével
- Növényzet és zaj detektálása és eltávolítása az adathalmazból a 4.1.2 fejezetbe ismertetett eljárás segítségével
- Különböző háztetők detektálása területelárasztó (*floodfill*) algoritmus segítségével

A tetők detektálása előtt célszerű a talajt, a növényzetet és az alacsony felbontású területeket (például: épületek oldalfalai) leválasztani az adathalmazról, hogy ne befolyásolják az algoritmus működését. A *floodfill* eljárás csak a tetőket kapja meg bemenetként, majd detektálja őket, mint a térben különálló pontszigeteket, amelyekben a pontok kis távolságra vannak egymástól. Az eljárás kimenete a 4.9 ábrán látható. Az algoritmus megengedi, hogy az adathalmazból eltávolított régiók, a tetődetektálás után visszaépíthetők legyenek a pontfelhőbe.

4.1.4. Járműdetekció

Ebben a fejezetben bemutatunk egy általunk tervezett és tesztelt eljárást, ami járművek felismerésére, lokalizálására és megszámlálására szolgál. A módszer a korábban



4.9. ábra. Háztetők - mint különálló egységek - detektálása *floodfill* algoritmussal

ismertetett alapalgoritmusokon alapul. Mindenekelőtt a pontfelhőből eltávolítjuk azokat a pontosztályokat, amelyek nem képezik szignifikáns részét a felismerésnek: ritka pontfelhőrészeket és a növényzetet. Ezek után a pontfelhőből eltávolítjuk a talaj síkjához és a háztetűkhöz tartozó pontokat. A ponthalmaz megmaradt részeiből feltételezhetjük, hogy járműveket, további utcai objektumokat, illetve zajt tartalmaznak. Az eljárás utolsó lépésében ebből a maradék halmazból próbáljuk méret alapján elkülöníteni az egyes járműveket a területelárasztó (*floodfill*) eljárás felhasználásával. Az algoritmus a következő lépéseket követi:

- növényzet és az alacsony felbontású térrészek detektálása és eltávolítása sűrűség és lézer visszaverődések alapján történő klaszterezéssel
- talaj síkjának detektálása RANSAC algoritmussal
- háztetűk felismerése *floodfill* eljárással

- járművek szegmentálása és detektálása *floodfill* eljárással

Legyen P a teljes adathalmaz, és $p \in P$ egy tetszőleges adatpont a halmazban. Az algoritmus a következő részhalmazokat távolítja el a teljes adathalmazból.

$$P_j = P \setminus (P_n \cup P_z \cup P_t \cup P_h)$$

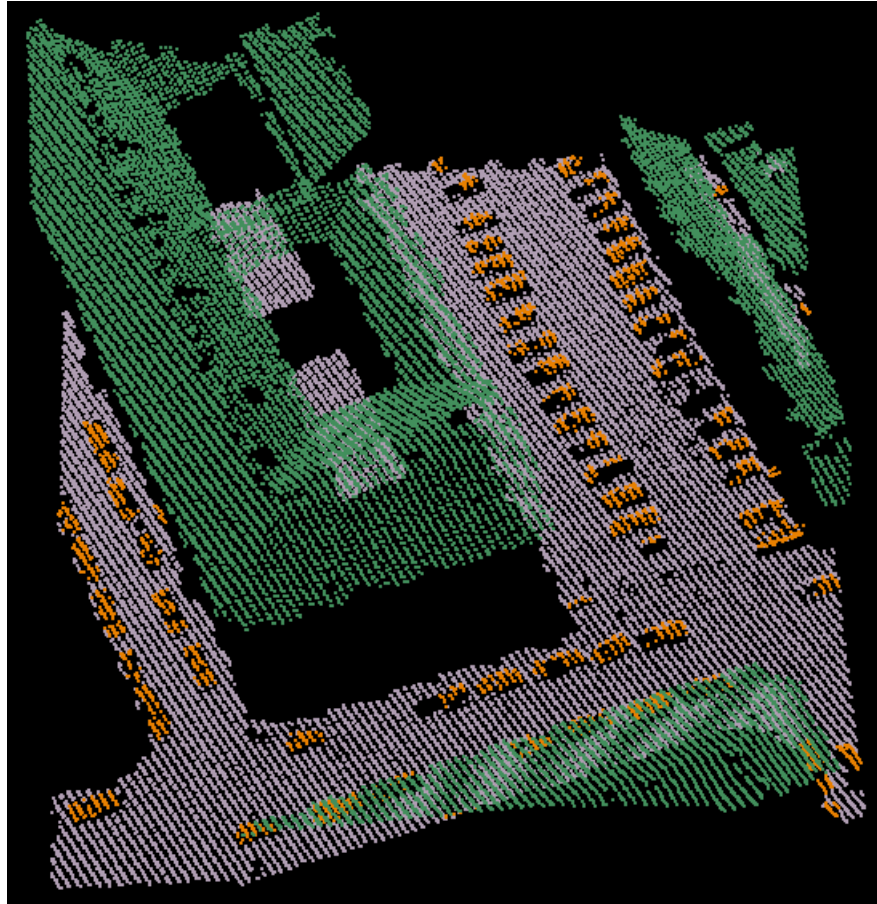
Ahol P_n a növényzethez tartozó pontok halmaza, P_z a zajhoz és ritka térrészekhez tartozó pontok halmaza, P_t a talaj síkjához tartozó pontok halmaza és P_h a háztetőkhez tartozó pontok halmaza. Továbbiakban a P_j maradékhalmazt vizsgáljuk, ami feltételezésünk szerint tartalmazza a járműveket. Az egyes gépkocsik egybefüggő szigeteket alkotnak a maradékhalmazban, így P_j partícionálásával detektáljuk őket: $P_j = \cup_{i=1} Q_i$. A partícionálást ismét a *floodfill* algoritmussal végezzük és a megfelelő méretű Q_i partíciókat azonosítjuk járművekként.

A 4.10 ábrán megfigyelhető az algoritmus működése. A bemeneti adathalmazról leválasztjuk a járműdetektálás szempontjából irreveláns részeket, majd a területelárasztó algoritmus használatával detektáljuk az egyes járműveket, így az eljárás végeztével, minden egyes jármű külön klaszterbe fog tartozni.

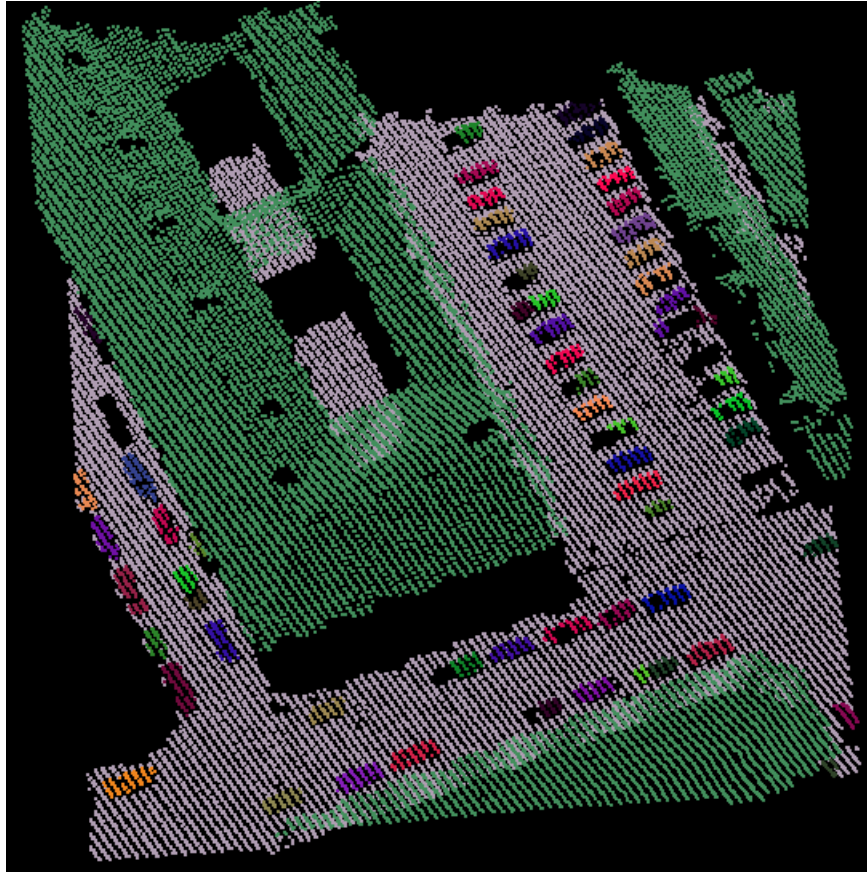
A 4.11 ábrán a Budapest belvárosában készült LiDAR felvételeken végrehajtott jármű pontosztály szegmentálása látható. A szegmentált adathalmazon *floodfill* algoritmus segítségével járműdetekciót végezhetünk, amely egyenként lokalizálja és detektálja az adathalmazon lévő járműveket. Az algoritmus kimenete a 4.12 ábrán látható. Annak érdekében, hogy lemérjük a kifejlesztett algoritmus hatékonyságát, megvizsgáltuk az eljárás pontosságát (*precision*) és fedési rátáját (*recall*). A *precision* értéke azt határozza meg, hogy a járműdetekció során az algoritmus találatai milyen



4.10. ábra. Jármű felismerő algoritmus működése



4.11. ábra. A pontfelhő szegmentáló algoritmus eredménye: a talaj (P_t , fehér), háztető (P_h , zöld) és jármű (P_j narancssárga) osztályok megjelenítése.



4.12. ábra. A járműdetektáló algoritmus eredménye. Fehérrel jelöltük a talajt, zölddel a tetőket, míg az egyes elkülönített járművek különböző színnel jelennek meg.

Járműdetekció	
Járművek száma:	56
Jól detekált:	47
Hibásan detektált:	8
False Positive:	1
False negative:	7
<i>Precision:</i>	97%
<i>Recall:</i>	84%

4.1. táblázat. A kifejlesztett járműdetekciós eljárás kiértékelése

arányban helyesek. Meghatározásához képeznünk kell a helyesen járműnek detektált objektumokat és el kell osztanunk a helyesen járműnek detektált objektumok és a hamis riasztások számának az összegével. Vezessük be a következő jelölést:

J = helyesen járműnek detektált objektumok száma, T = a valóságban jelen lévő járművek száma, F = hamis riasztások száma

$$precision = \frac{J}{J + F}$$

A *recall* értéke azt szemlélteti, hogy az adathalmazban valóban jelen lévő járművek közül hányat találtunk meg. Meghatározásához képeznünk kell a helyesen járműnek detektált objektumok és a valóságban jelen lévő járművek darabszámának a hányadosát:

$$recall = \frac{J}{T}$$

A 4.1 táblázatban a járműdetekciós algoritmus teljesítményéről láthatunk adatokat. Az tesztadaton lévő járművek teljes száma 56 darab, az algoritmus ezek közül 47-et detektált helyesen. Egy esetben detektált hamisan (*false positive*), ekkor az

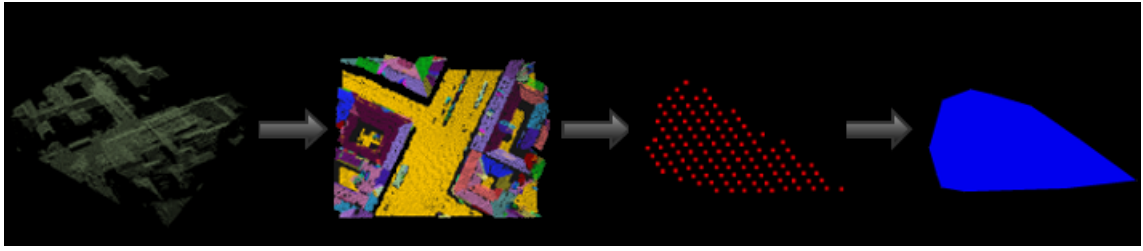
algoritmus olyan objektumokat detektált járműnek, amik eredetileg nem azok. 7 esetben történt hiányos, vagy dupla detektálás egy járműre vonatkozólag. Ilyenkor az algoritmus nem detektálta járműnek azokat az objektumokat amelyek eredetileg járművek lennének (*false negative*).

4.2. Virtuális rekonstrukció pontfelhőben

Ebben a fejezetbe a már elkülönített pontosztályokra ajánlunk egy rekonstrukciós módszert. Célunk, hogy a felismert városi régiókra geometriai síkokat illesztve létrehozzunk az értelmezett városi régiók virtuális modelljét [6]. Munkánk során az épületeket kíséreltük meg rekonstruálni. a) A már korábban detektált tetőrészekre hajtottunk végre egy poligon approximációt. b) A házak oldalfalainak hiányossága miatt, terveztünk egy algoritmust, ami szintetikus generálja az épületek oldalfalait. Algoritmusunk, a 2.2-es fejezetben bemutatott vizualizációs szoftverrel ellentétben, kiszűri a zajt a rekonstrukció előtt és tartalmilag értelmezi a rekonstruálandó régiókat.

4.2.1. Tetőszerkezetek poligon approximációja

Az algoritmus célja, hogy az adathalmaz épületeit egy egyszerű geometriai modellel közelítse. A modellkészítés előtt szükségeszerű előfeldolgozást végezni a jobb eredmény érdekében. Ismét a növényzet, zaj és talaj leválasztásával kezdjük az algoritmust a már bemutatott módszerekkel. Ezek után a már bemutatott *k-means* és *floodfill* alapú eljárásokkal felismerjük és lokalizáljuk a háztetők tetőszegmenseit. A felismert tetőszegmensekre egyenként konvex burkolók illesztünk, annak érdekében, hogy meghatározzuk a kontúrokat és a pontok domináns orientációját. Erre azért



4.13. ábra. A poligon approximációt végző algoritmusunk folyamatábrája

van szükség, mert az egy tetőszegmenshez tartozó pontok a mérés következtéből nem helyezkednek el tökéletesen egy síkba. A [16]-ban bemutatott konvex burkoló algoritmust használva, kiszűrhetjük a síkhoz tartozó, ámde kilógó pontokat és megtarthatjuk azokat, amik a legpontosabban illeszkednek a síkra. A megkapott domináns síkra illeszkedő pontokra ráilleszhetünk egy geometriai primitívet, aminek alakja követi az adott tetőszegmens pontjainak konvex burkolóját. Ezt az eljárást végrehajtjuk az adathalmazban található összes detektált tetőszegmensre, majd az eljárás végén a térbeli poligonokat visszaillesztve az adathalmazba, megkapjuk a pontfelhő geometria közelítését, vagy másnéven virtuális rekonstrukcióját. Abból adódóan, hogy egy ház teljes tetőszerkezete sok bonyolult formát és konkáv egységet tartalmaz, a tetőszegmensekre bontás nélkül a geometriai modellezése nagy kihívást jelent.

A 4.13 ábrán látható a poligon approximációt végző algoritmus folyamatábrája. Elsőként a nyers pontfelhőből lokalizáljuk és felismerjük a háztetők egyes tetőszegmenseit a már bemutatott algoritmusainkal. Ezek után a meghatározott tetőszegmensek mindegyikére végrehajthatjuk a konvex burkolót ráillesztő algoritmust. Az algoritmus kimenete egy térbeli poligon, ami a pontfelhőből kinyert tetőszegmensek formáját követi. Az előállított tetőpoligonokat összeillesztve kapjuk meg a tetők rekonstrukcióját. Az algoritmus hátránya, hogy előfordulnak olyan tetőszegmensek amelyek,

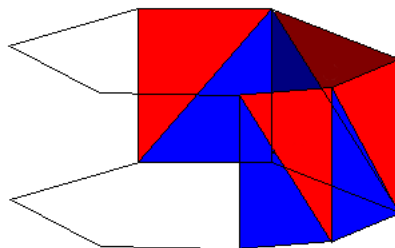
konkávak, ilyenkor a geometriai illesztés hiányos lesz, hiszen az eljárás csak konvex alakzatokat tud approximálni.

4.2.2. Épületek oldalfalainak szintetikus generálása

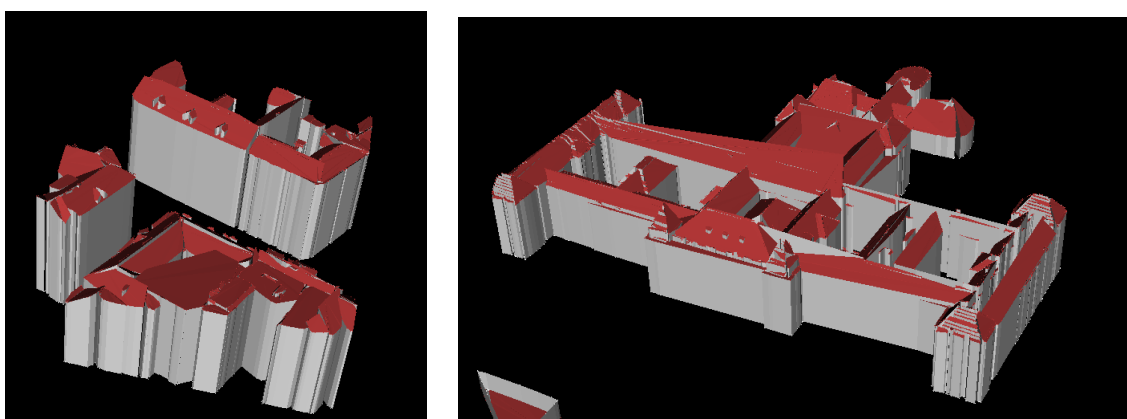
Az előző fejezetekben említettük, hogy a LiDAR szkennelések során a házak oldalfalairól készült mérések gyakran hiányosak, mivel az eszköz sok esetben nem tud megfelelő szögben mérni. Munkánk virtuális rekonstrukcióval foglalkozó részében célunk volt ezeknek a hiányos, vagy alacsony felbontású régióknak a helyreállítása. Annak érdekében, hogy egy teljes geometriai közelítést kapjunk a rekonstruálandó épületekről, szükségszerű a házak oldalfalainak szintetikus előállítása. Az elkészített oldalfal generáló algoritmus az alábbi lépésekből áll:

- az algoritmus fogadja az előzőekben előállított tetőszegmensek konvex burkolójának sarokpontjait és megkapja bemenetként a talaj szintjének a magasságértékeit.
- a konvex burkolók sarokpontjait levetítjük a talaj síkjára.
- a burkolók sarokpontjai és a levetített sarokpontok között végzett háromszögelés (*triangulation*)

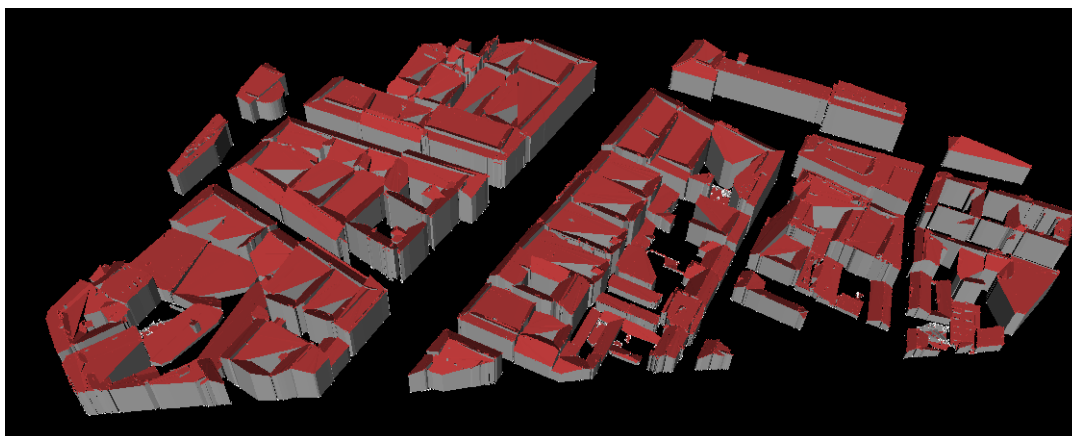
A talajsíkra vetített és az eredeti konvex burkoló pontok közötti összeköttetéseket háromszögeléssel oldottuk meg. A 4.14 ábrán látható az algoritmus működése. A talajsíkra vetített és az eredeti sarokpontok között felváltva képezünk háromszögeket. Addig ismételjük az eljárást amíg körbe nem értünk az adott épület oldalfalán. Ennek az eljárásnak a segítségével egy sokkal teljesebb épületrekonstrukciót kaphathunk eredményül.



4.14. ábra. Szintetikus oldalfal előállítás háromszögelés segítségével



4.15. ábra. Balra: Egy budapesti városrészlet poligon approximációja. Jobbra: A BME főépületének poligon approximációja



4.16. ábra. Nagyobb budapesti városrészlet poligon approximációja

A 4.15 és 4.16 ábrákon láthatóak az algoritmus kimenetei. Az eljárásokat különböző területeken teszteltük. Megfigyelhető a tetőszerkezetek térbeli poligonokkal való közelítése és az eredetileg hiányos oldalfalak pótlása.

5. fejezet

Konklúzió és további lehetőségek

A bemutatott munka célja egy automatikusan működő és a városi környezet értelmezésére és rekonstrukciójára képes algoritmusok kifejlesztése és letesztelése volt légi LiDAR pontfelhőkön. A megoldott feladatok magukban foglaltak pontosztályozást, objektumok és régiók szegmentálását és detektálását. A munka elvégzéséhez rendelkezésünkre álltak az Infoterra Astrium GEO-Information Services © adatai. Saját algoritmusaink C++ nyelven készültek és implementálásukban nagy segítséget nyújtottak a PCL és a libLAS függvénycsomagban található beépített alapfunkciók.

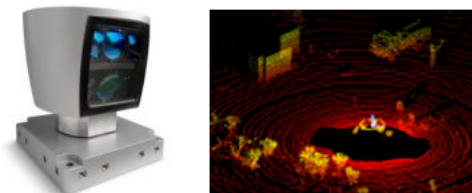
I A feladatmegoldás elején megismerkedtünk a szakirodalomban található algoritmusokkal és technikákkal, amelyek nagy segítségünkre lettek a munkánk során. Bemutattuk a LiDAR rendszer általános felépítését és az adatok lehetséges vizualizációját. Bemutattuk a pontfelhő feldolgozással és manipulációval foglalkozó fejlesztőeszközök előnyeit és hátrányait.

II Módszereket ajánlottunk alacsony felbontású LiDAR pontfelhők osztályozására és különböző területi osztályok (talaj síkja, háztetők, növényzet, zaj) elkülönítésére.

III Algoritmust dolgoztunk ki jáművek felismerésére, lokalizálására és megszámlálására a nyers pontfelhőben.

IV Egy automatikus eljárás segítségével épületrekonstrukciót hajtottunk végre a struktúrálatlan pontfelhőből.

A bemutatott munka több irányban is továbbfejleszthető. A felismerési és pontosztályozási módszereket illetően, érdemes megvizsgálni a [12] munkában ajánlott hierarchikus modelleket. Ezek a modellek teret nyitnak arra, hogy objektumok mellett, hatékonyabban kezeljünk objektum csoportokat, vagy régiókat az adathalmazban. A rekonstruálási módszerek továbbfejlesztési lehetőségeit figyelembe véve, pedig érdemes kipróbálni olyan algoritmusokat [6],[29] amelyek kezelik a mérési zajból adódó poligon illesztési pontatlanságokat, vagy hiányosságokat. A járműdetekció pontosítására és robosztussá tételére megoldás lehet az úgynevezett *Marked Point Process* - *MPP* alapú algoritmus [2], ami lehetővé teszi, hogy az adatokból kiinduló modellünket kiegészítsük prior feltételekkel és objektumok közötti kapcsolatok modellezésével. Az elkészült TDK munka továbbfejlesztési lehetősége lehet még a légi, illetve földi LiDAR mérések fúziója [5], amihez majd felhasználhatjuk a kutatócsoportunkban közelmúltban beszerezett Velodyne HDL64-E S2 típusú földi LiDAR készüléket (5.1 ábra).



5.1. ábra. Velodyne HDL64-E S2 LiDAR és mérési eredménye

6. fejezet

Köszönetnyilvánítás

Köszönettel tartozunk témavezetőinknek: Dr. Benedek Csabának és Dr. Szirányi Tamásnak, hogy töretlen bizalommal támogatták munkánk, továbbá köszönettel tartozunk az MTA SZTAKI-nak, hogy biztosította számunkra a megfelelő infrastruktúrális és szellemi háttérrel annak érdekében, hogy minél hatékonyabban végezhessük a feladatunkat. Szeretnénk köszönetet mondani az MTA SZTAKI Elosztott Események Elemzése kutatócsoportban dolgozó összes kollégánknak, akik valamilyen módon segítettek a munkánk. Végül de nem utolsó sorban szeretnénk köszönetet mondani az Infoterra Astrium GEO-Information Services Hungary © -nek, hogy biztosították számunkra a Budapest belvárosáról készített LiDAR mérési adatokat.

Irodalomjegyzék

- [1] S. Aparajithan and S. Jie. Building roof segmentation and reconstruction from lidar point cloud using clustering techniques. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, Beijing, China*, 37:279–284, 2008.
- [2] C. Benedek, X. Descombes, and J. Zerubia. Building development monitoring in multitemporal remotely sensed image pairs with stochastic birth-death dynamics. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 34(1), January 2012.
- [3] B. Gorte. Roof plane extraction in gridded digital surface models. *International Archives of Photogrammetry and Remote Sensing*, XXXVIII, 2009.
- [4] G. Horváth. *Neurális hálózatok*, chapter 10, pages 302–303. Panem, 2006.
- [5] E. Kim and G. G. Medioni. Urban scene understanding from aerial and ground lidar data. *Machine Vision and Applications*, 22(4):691–703, 2011.
- [6] F. Lafarge and C. Mallet. Modeling urban landscapes from point clouds: a generic approach. Technical Report 7612, INRIA, 2011.

- [7] T. Lovas, Á. Barsi, K. Szócs, and Z. Kibédy. Reconstruction of laser-scanned vehicles. *International Archives of Photogrammetry and Remote Sensing*, XXXVI(I/W3), 2005.
- [8] T. Lovas, C. K. Toth, and Á. Barsi. Model-based vehicle detection from lidar data. *International Archives of Photogrammetry and Remote Sensing*, XXXV(2):134–137, 2004.
- [9] H. Ma and C. Yao. A methodology for trees detection using LiDAR data in urban areas. *International Conference on Multimedia Technology, Ningbo, China*, 2010.
- [10] S. Mills, M. Gerardo, Z. Li, J. Cai, R. F. Hayward, L. Mejias, and R. A. Walker. Evaluation of areial remote sensing techniques for vegetation management in power-line corridors. *IEEE Transactions on Geoscience and Remote Sensing*, 48(9):3379–3390, 2010.
- [11] M. Pauly. *Point primitives for interactive modeling and processing of 3d geometry*. PhD thesis, ETH Zurich, 2003.
- [12] J. Porway, K. Wang, and S. Zhu. A hierarchical and contextual model for aerial image understanding. *International Journal of Computer Vision*, 88:254–283, 2010.
- [13] Z. Qian-Yi and N. Ulrich. Fast and Extensible Building Modeling from Airborne LiDAR Data. *International Conference on Advances in Geographic Information Systems, Irvine, CA, USA*, 2008.

- [14] Z. Qian-Yi and N. Ulrich. A streaming framework for seamless building reconstruction from large-scale aerial LiDAR data. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2759–2766, 2010.
- [15] Á. Rakusz, T. Lovas, and Á. Barsi. Lidar-based vehicle segmentation. *International Archives of Photogrammetry and Remote Sensing*, XXXV(2):156–159, 2004.
- [16] R. B. Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Institut für Informatik der Technischen Universität München, 2009.
- [17] Y. Shen, L. Wu, and Z. Wang. Identification of inclined buildings from aerial LIDAR data for disaster management. *International Conference on Geoinformatics, Beijing, China*, 2010.
- [18] L. Szirmay-Kalos. *Számítógépes Grafika*, chapter 7, pages 104–105. ComputerBooks, 1990.
- [19] L. Szirmay-Kalos, G. Antal, and F. Csonka. *Háromdimenziós grafika, animáció és játékfejlesztés*, chapter 6, pages 174–180. ComputerBooks, 2009.
- [20] C. K. Toth, Á. Barsi, and T. Lovas. Vehicle recognition from LiDAR data. *International Archives of Photogrammetry and Remote Sensing*, XXXIV(3/W13):162–166, 2003.

- [21] V. Vivek, K. Rakesh, and C. H. Stephen. 3d building detection and modeling from aerial LIDAR data. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2213–2220, 2006.
- [22] P. Wang. Research on extracting the tree height based on lidar data. *International Congress on Image Processing, Hong Kong, China*, 5:2236–2239, 2010.
- [23] B. Yang, S. Pramod, and N. Ram. Vehicle detection from low quality aerial LIDAR data. *Applications of Computer Vision*, pages 541–548, 2011.
- [24] M. Y. Yang and W. Foerstner. Plane detection in point cloud data. Technical Report TR-IGG-P-2010-01, Department of Photogrammetry, University of Bonn, 2010.
- [25] W. Yao, S. Hinz, and U. Stilla. Vehicle activity indication from airborne LiDAR data of urban areas by binary shape classification of point sets. *International Association People Environment Studies*, XXXVIII:35–40, 2009.
- [26] W. Yao, S. Hinz, and U. Stilla. Automatic vehicle extraction from airborne LiDAR data of urban areas aided by geodesic morphology. *Pattern Recognition Letters*, 31:1100–1108, 2010.
- [27] C. Yun-Woong, J. Young-Woon, L. Hyo-Jong, and C. Gi-Sung. Three-dimensional LiDAR data classifying to extract road point in urban area. *IEEE Geoscience and Remote Sensing Letters*, 5(4):725–729, 2008.

- [28] W. Zhi, L. Huiying, and W. Lixin. Geodesics-based topographical feature extraction from airborne lidar data for disaster management. *International Conference on Geoinformatics, Beijing, China*, pages 1–5, 2010.
- [29] Q.-Y. Zhou and U. Neumann. 2.5d building modeling with topology control. In *Conference on Computer Vision and Pattern Recognition*, pages 2489 –2496, june 2011.