TDK-dolgozat

Krisztandl Roland

LiDAR pontfelhők felszínosztályozása gépi tanulással

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

PROGRAMOZÁSELMÉLET ÉS SZOFTVERTECHNOLÓGIAI TANSZÉK
INFORMÁCIÓS RENDSZEREK TANSZÉK

Szerző:

Krisztandl Roland

Programtervező informatikus BSc

3. évfolyam

Témavezetők:

Cserép Máté

Egyetemi tanársegéd

dr. Vincellér Zoltán

Mesteroktató

Budapest, 2022

Tartalomjegyzék

1. fejezet

Bevezetés

Manapság egyre szélesebb körben érhetőek el légi felvételű LiDAR pontfelhők, melyeket többféleképpen is fel lehet használni. Azonban a felhasználhatóságban nagyban segít az, ha az adott pontfelhő osztályozva van. Osztályozott pontfelhő esetén az objektumok detektálása, vagy domborzatmodellek készítése sokkal egyszerűbbé és gyorsabbá válik.

Pontfelhők osztályozása évek óta aktívan kutatott terület. A legtöbb tanulmány célja a bináris osztályozás, azaz a talaj és nem-talaj pontok meghatározása domborzatmodell készítésének a céljából. Az utóbbi időszakban azonban több tanulmány is született azzal a céllal, hogy a pontfelhők osztályozását automatizálják. Ezen tanulmányok során alkalmaztak gépi tanulást, neuronhálót és mélytanulást, valamint teljesen egyedi megközelítéseket is, a felhasználási területtől függően.

Ezen kutatás során én is gépi tanulást fogok használni, azon belül is felügyelet nélküli klaszterezést és a felügyelt osztályozást. Az ehhez szükséges attribútumokat egy részét az előfeldolgozás során számítom ki, ilyen például a pontok lokális környezetéből való statisztikai adatok gyűjtése, valamint a talaj feletti magasság, míg mások már előre adottak a pontfelhőben, mint a térbeli pozíció, a visszatérések száma vagy éppen az intenzitás.

A dolgozat második fejezete betekintést nyújt a témát érintő fogalmakba, az általam hasznosnak vélt módszerekbe, valamint korábbi szakirodalmakba. A harmadik fejezetben bemutatásra kerül az én megoldásom a problémára, valamint, hogy milyen adatokat használtam a tanításra és a validálásra. A negyedik fejezet ír az elkészült

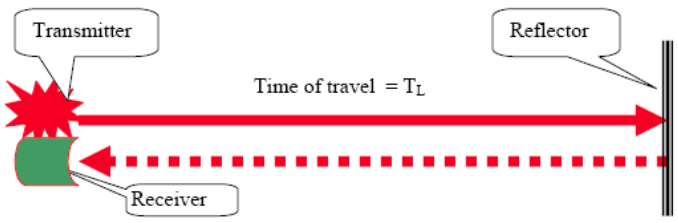
alkalmazás implementációjáról, míg az ötödik fejezet részletezi a tanulmány során elért eredményeket. Végezetül a hatodik fejezet tartalmazza a kutatás összefoglalóját és a továbbfejlesztési lehetőségeket.

2. fejezet

Irodalomkutatás

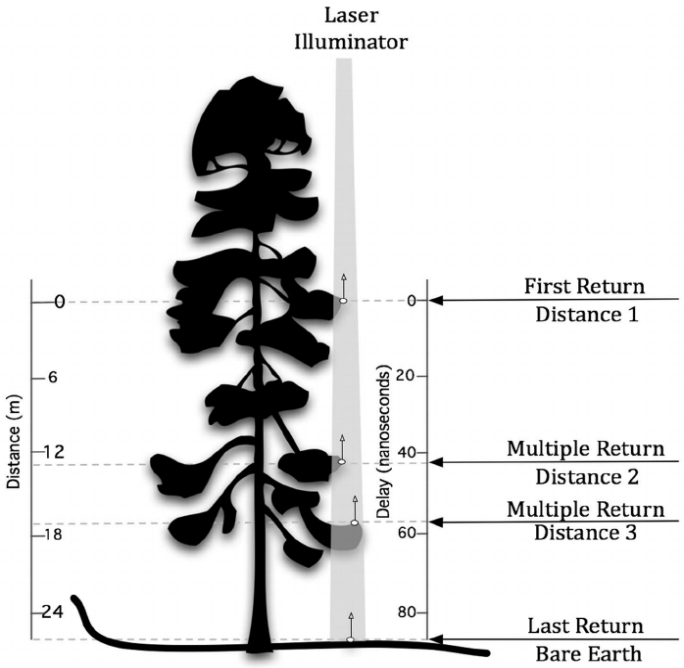
2.1. Lézeralapú távérzékelés

A LiDAR (*Light Detection and Ranging*, magyarul lézeralapú távérzékelés) [1] alapvetően egy kibocsátó eszköz és egy visszaverő felület távolságának meghatározására alapuló módszer. A módszert a lézersugár felfedezése után, az 1960-as évek elején dolgozták ki, és 1971-ben az Apollo-15 küldetés során is használták. Ha nagyon gyors ütemben nagyon sok lézeralapú távolságmeghatározást végzünk, akkor nagy felületű tárgyak, földfelszín vagy belső terek objektumainak tudjuk az elhelyezkedését meghatározni. Ezt a módszert lézerszkennelésnek nevezzük. A LiDAR a radartól eltérően az ultraibolya, az infravörös és a látható tartományban működik. A lézer által kibocsátott lézerfény vagy energiaimpulzus a terjedés irányában lévő objektumokról visszaverődik. A rendszer rögzíti az egyes impulzusok kibocsátása és visszaverődése között eltelt időt. Az elektromágneses energia terjedési sebessége ismert, így a kibocsátás és a tárgy által visszavert jel visszaérkezése idejének különbségéből meghatározható a tárgyaknak a műszertől való távolsága. Ezt szemlélteti a 2.1 ábra.



2.1. ábra. A lézeres távolságmérés szemléltetése. Forrás: [1]

Térképezési célokra a műszer speciálisan formált lézerimpulzusokat bocsát ki, és a visszaverődő jeleket (ún. *echo*) érzékeljük, a távolság az időkülönbségből számítható. Mivel a jel nem vonalszerűen, hanem a forrástól kis kúpszögben terjed, a felszínen sem egy pontot, hanem egy kiterjedt foltot világít meg, ahol több különböző távolságra lévő objektum is lehet. Ennek megfelelően nem egy, hanem legtöbbször több echót kapunk egy impulzus kibocsátása esetén. Az első beérkező visszaverődést (*first echo*) általában a növényzetnek vagy más tereptárgynak, az utolsót (*last echo*) pedig általában a talaj hatásának tulajdonítjuk. A lézerszkennelés korai időszakában a first echo/last echo módszer alkalmazták, két jelet rögzítve. Manapság egyre jobban terjed a *multiecho* vagy a teljes jelalakos (*full wave-form*, FWF) módszer. A *multiecho* módszernél valahány (pl. 3-5) echót rögzítenek (pl. első és utolsó kettő, vagy első három és az utolsó), a teljes jelalakos rögzítésnél pedig az echók jelalakját is rögzítik, így – megfelelő, általában elég időigényes utófeldolgozás esetén – lehetőség van akár 8-10 echo elkülönítésére is. Ez utóbbi pl. növényzetmonitorozásnál elengedhetetlen. A többszörös visszatérést szemlélteti a 2.2 ábra.



2.2. ábra. A többszörös visszatérés szemléltetése. Forrás: [2]

A térképezési célok esetén a mérési pontosság vertikális értelemben 5–15 cm, horizontálisan pedig a repülési magasság függvényében 10–25 cm, teljes jelalakos módszernél elérheti az 5 cm-t is. A módszer földi felbontását általában inkább a

pontsűrűséggel adják meg, a mai rendszerek tipikusan 4-16 pont/m² pontsűrűséggel mérnek, de adott célokra vannak már ezt meghaladó sűrűségű mérések is.

A LiDAR-t többféleképpen is csoportosíthatjuk, a legnépszerűbb csoportosítás a légi és földi LiDAR, ahol az előbbi esetben egy repülő tárgyon (pl.: drónon) van a műszer, utóbbi esetben pedig egy földön elhelyezkedő objektumon (pl.: autón).

A LiDAR pontfelhők általában *.LAS*[3] vagy *.LAZ*[4] kiterjesztésű fájlokban tárolják.

2.1.1. Pontok osztályai

A .LAS kiterjesztés specifikálja azt hogy egy pontot milyen osztályokba kategorizálnak és hogy ezek a kategóriák mit jelentenek pontosan. A specifikáció összesen 11 pontformátumot támogat. 0 és 5 közötti formátumok a régebbiek, míg a 6 és 10 közöttiek az újabbak. A kétféle formátum más kategóriákat használ különböző értékekhez. Az értékeket és osztályokat a 2.1 és a 2.2 táblázat mutatja.

Érték	**Jelentés**
0	Created, Never Classified
1	Unclassified
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (Noise)
8	Model Key-Point (Mass Point)
9	Water
10	Reserved for ASPRS Definition
11	Reserved for ASPRS Definition
12	Overlap Points
13-31	Reserved for ASPRS Definition

2.1. táblázat. A 0-5 pontformátumok osztályozása

Érték	Jelentés
0	Created, Never Classified
1	Unclassified
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (Noise)
8	Reserved
9	Water
10	Rail
11	Road Surface
12	Reserved
13	Wire – Guard (Shield)
14	Wire – Conductor (Phase)
15	Transmission Tower
16	Wire-Structure Connector
17	Bridge Deck
18	High Noise
19	Overhead Structure
20	Ignored Ground
21	Snow
22	Temporal Exclusion
23-63	Reserved
64-255	User Definable

2.2. táblázat. A 6-10 pontformátumok osztályozása

2.2. Digitális magasságmodellek

A digitális magasságmodell (angolul: *Digital Elevation Model*, DEM) egy raszteres domborzatleíró módszer. A felszínt egy rácshálóval írja le. Minden rácspontban egy magasságérték található. Egy magasságmodellt általában interpoláció segítségével készítünk el. A modellek típusait szemlélteti a 2.3 ábra.

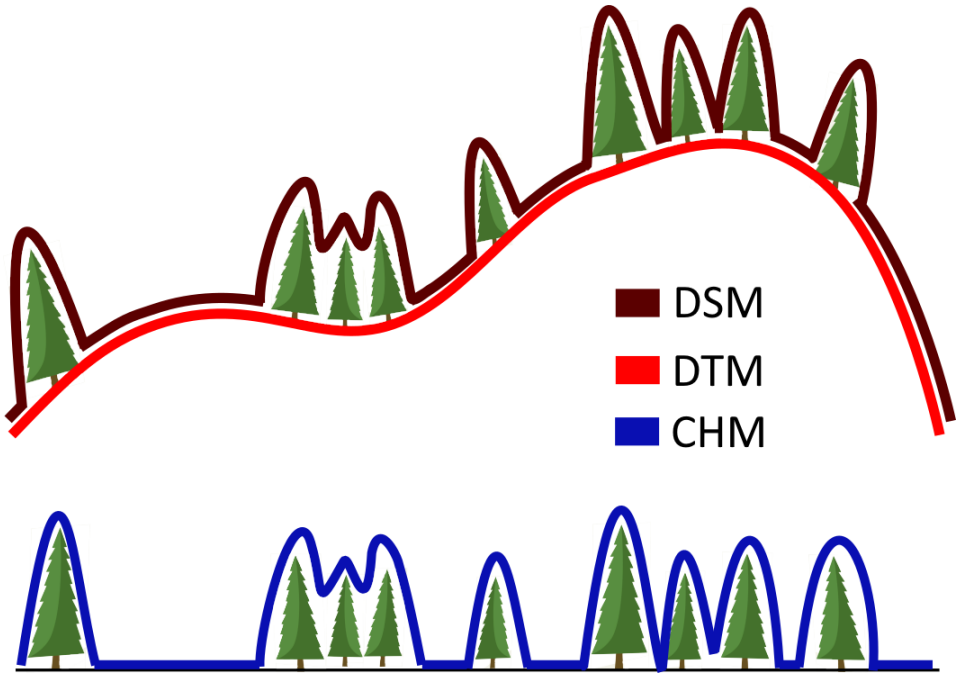
A digitális magasságmodell egyik fajtája a digitális felszínmodell (angolul: *Digital Surface Model*, DSM). Ebben az esetben a különböző tereptárgyak is szerepelnek a raszteren, mint például a fák, épületek vagy az autók.

A digitális magasságmodell egy másik fajtája a digitális domborzatmodell (angolul: *Digital Terrain Model*, DTM[1]). Ez abban különbözik a DSM-től, hogy ezen nem

[1]Egyes szakirodalmakban DEM-ként hivatkozhatnak a DTM-re, én a DEM-et általános fogalomként használom.

szerepelnek a tereptárgyak, hanem csak a talaj. Ezen állományok előállítása sokkal bonyolultabb feladat.

A *Canopy Height Model* (CHM) egy speciális magasságmodell, amit úgy kaphatunk meg, hogy a DSM-ből kivonjuk a DTM-et. Ez a modell azt mutatja meg, hogy az egyes tereptárgyak (például fák vagy épületek) milyen magasak.



2.3. ábra. A DSM, a DTM és a CHM szemléltetése. Forrás: [5]

2.3. Gépi tanulás

A gépi tanulás a mesterséges intelligencia egyik ágazata, ahol az algoritmusok képesek adatokból mintákat felismerni, ezen minták alapján egy matematikai modellt készíteni, majd újabb adatokra előrejelzéseket elvégezni. Sokféle algoritmus áll rendelkezésre, amiket a konkrét feladattól függően érdemes kiválasztani.

A gépi tanulás algoritmusai 3 fő csoportra oszthatóak:

- **Felügyelet nélküli tanulás**: Nincsenek címkék, összefüggéseket keres az adatok között.
- **Felügyelt tanulás**: Címkézett tanító adatok, kimenetel előrejelzéséhez használják.
- **Megerősítéses tanulás**: Nincsenek címkék, de képes az eredményeket minősíteni.

2.3.1. Felügyelet nélküli tanulás

A felügyelet nélküli tanulás során az algoritmus rejtett összefüggéseket keres az adatokban. Ebben az esetben a kimenet nem ismert. Két fő csoportra osztható:

- **Klaszterezés**: Célja az adatok csoportosítása.
- **Dimenziócsökkentés**: Célja a sok dimenzióból álló adatokból csak a lényeges információ kiszűrése.

Klaszterezés

A klaszterezés során a célunk az adatok csoportosítása rejtett összefüggések keresésével. A folyamat végén az egy klaszterbe tartozó adatokban van valamilyen közös tényező.

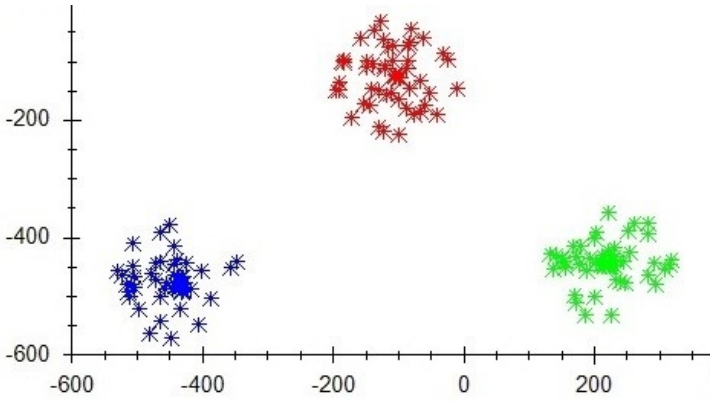
A klaszterezési problémákat megoldó algoritmusok is több csoportba oszthatóak: centroid alapú, hierarchikus vagy sűrűség alapú. Továbbá neurális hálók segítségével is megvalósítható.

K-közép módszer

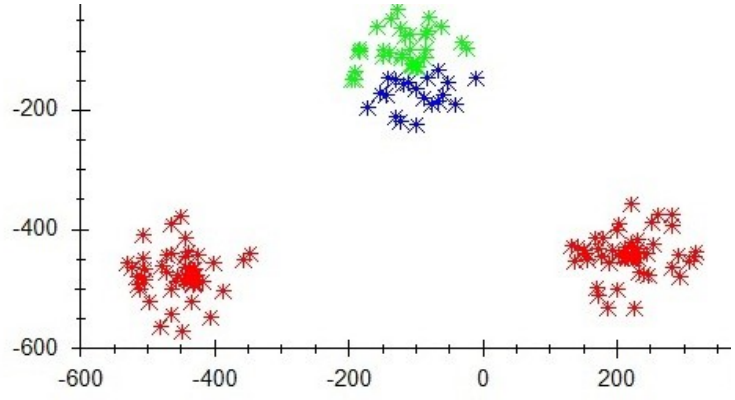
A k-közép (angolul *k-means*) az egyik legismertebb centroid alapú klaszterező algoritmus. Bemeneti paraméterként meg kell adni a klaszterek számát. Az algoritmus indulásakor a centroid pontokat véletlenszerűen választja ki. Ezután végig megyünk az adatpontokon és mindegyiket ahhoz a klaszterhez soroljuk, amelyiknek a centroid pontja a legközelebb esik hozzá. Majd újra kiszámoljuk a centroid pontokat. Ezt addig ismételjük amíg a centroid pontok változnak.

Mivel távolságokat számolunk, ezért elengedhetetlen a bemenő adatok normalizálása.

A véletlenszerű kezdőpontok miatt érdemes az algoritmust többször is lefuttatni. Ezt szemlélteti a 2.4 és a 2.5 ábra.



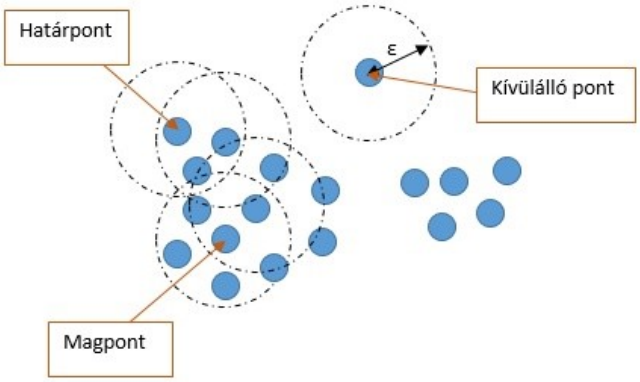
2.4. ábra. A k-közép algoritmus optimális kimenete. Forrás: [6]



2.5. ábra. A k-közép algoritmus rossz centroid választás esetén. Forrás: [6]

DBSCAN

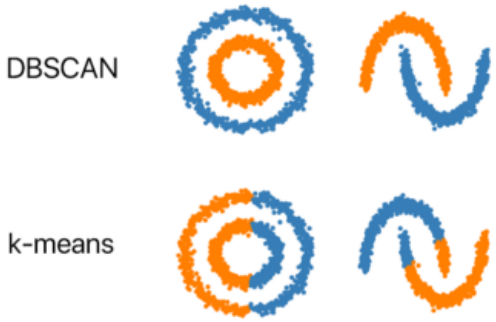
A DBSCAN az egyik legnépszerűbb sűrűség alapú klaszterező algoritmus. Két bemeneti paramétere van: a szomszédossági sugár és a sűrűségi korlát. Az algoritmus szerint egy klaszter nem más, mint azon pontok maximális halmaza, melyek sűrűség alapján elérhetőek egymásból. A működése a következő: először minden pontra meg kell határoznunk hogy a megadott sugáron belül hány szomszédja van, majd ha ez a szám nagyobb mint a sűrűségi korlát, akkor magpontnak minősítjük. Ha egy pont nem magpont, de a környezetében van magpont akkor határpont. Ha egy pont egyik sem, akkor zajpont. A pontok típusait szemlélteti a 2.6.



2.6. ábra. Adatpontok típusai adott sugár és sűrűségi korlát = 4 esetén. Forrás: [6]

Ezután kiválasztunk egy feldolgozatlan magpontot és annak az adott környezetében lévő mag- és határpontokat egy klaszternek vesszük. Majd a környezetben lévő magpontokra is megismételjük ezt a lépést, addig amíg tudjuk. Ezután ezt a folyamatot addig ismételjük amíg el nem fogynak a magpontok.

A DBSCAN algoritmus nagy előnye, hogy konkáv alakzatokra is nagyon jól működik a k-közép algoritmussal ellentétben. Ezt mutatja meg a 2.7 ábra.



2.7. ábra. A DBSCAN és a k-közép algoritmus konkáv klaszterek esetén. Forrás: [7]

2.3.2. Felügyelt tanulás

A felügyelt tanulás során egy modellt építünk ami képes előrejelzéseket készíteni. Azért felügyelt, mert a tanítás során előre ismerjük hogy a bemenetre milyen kimenetet várunk el. Két fő csoportra osztható:

- **Osztályozás**: Akkor használjuk, ha az adatokhoz diszkrét értékeket rendelünk.
- **Regresszió**: Akkor használjuk, ha az adatokhoz folytonos értéket rendelünk.

Osztályozás

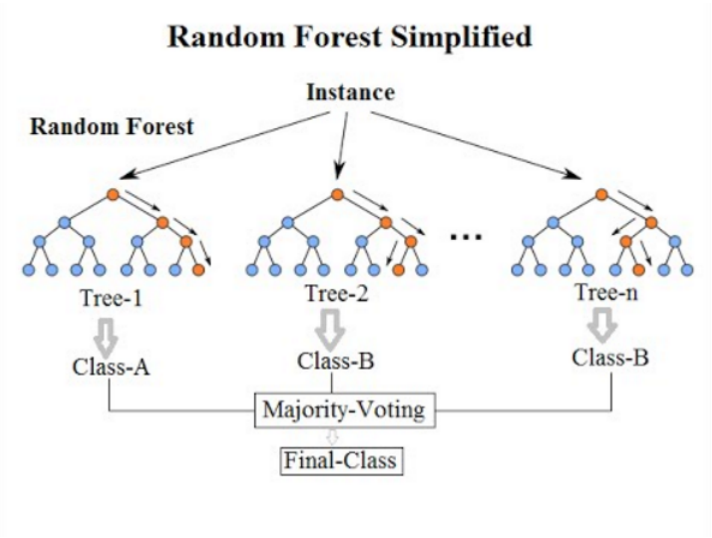
Az osztályozási problémák során a cél az hogy az adatokhoz egy kategóriát vagy osztályt rendeljünk. Ezek az osztályok diszkrétek, rendezetlenek és értelmezhetőek. Beszélhetünk bináris osztályozásról, ha csak két osztályunk van. Ha több osztályunk van akkor többosztályos osztályozásról van szó. Jelen esetben az osztályaink a talaj, a növényzet és az épület, tehát többosztályos osztályozásra van szükségünk.

Döntési fák

A tanító adataink alapján a döntési fa modellek megtanulnak egy sor kérdést, hogy ezáltal következtethessünk ismeretlen adatok osztályára. Egy döntési fában a belső csúcsok reprezentálják a kérdéseket, míg az eredményül kapott osztályt a levélcsúcsok tárolják.

Random forest

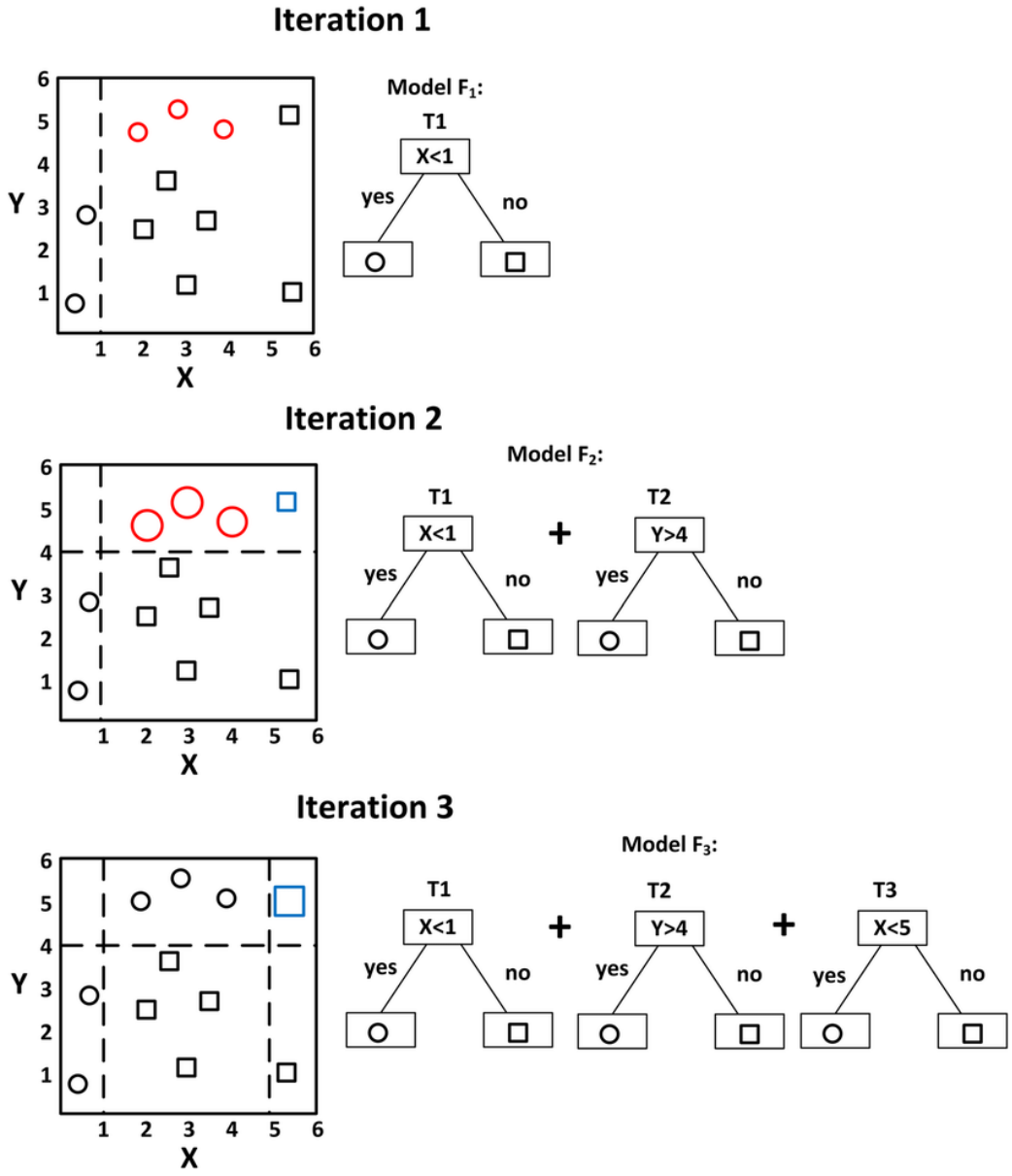
A *random forest* osztályozók a tanítás során több döntési fát is építenek, úgy hogy a fáknak az attribútumok egy véletlenszerű részhalmazát adják oda tanító adatként. Ismeretlen bemenet esetén az eredmény az az osztály amit a legtöbb döntési fa eredményez. Ezt a döntéshozatalt szemlélteti a 2.8 ábra. Általában pontosabbak mint a sima döntési fák.



2.8. ábra. A *random forest* működése. Forrás: [8]

Gradient Boosting

A *boosting* egy olyan gépi tanulásban használatos módszer ami több gyengébb osztályozó módszert kombinál, hogy egy erősebb osztályozót alkossanak. Ennek az egyik formája a *gradient boosting*. Ha a gyengébb osztályozó döntési fa, akkor *gradient boosted tree* modellről beszélünk. Míg a *random forest* több egymástól független döntési fát használnak, a *gradient boosted tree* modell a döntési fákat szekvenciálisan kombinálja, ezáltal kijavítja az előbbiek hibáit. Ezt szemlélteti a 2.9 ábra. Általában pontosabbak mint a *random forest* osztályozók.



2.9. ábra. A *gradient boosted tree* egyszerű ábrázolása a tanulás különböző iterációiban. Forrás: [9]

2.4. Korábbi kísérletek pontfelhő osztályozására

Az elmúlt évek során több kutatás is született amiknek a célja az volt hogy pontfelhőket osztályozzanak.

Angela Kim és társai [10] egy olyan gépi tanuláshoz használható attribútumhalmazt és az azok hasznosságát prezentálták, ami minden pont esetén a pontok egy környezetén alapul. Magához az osztályozáshoz k-közép klaszterezést és egy ún. *Spectral Angle Mapper, SAM* felügyelt osztályozót alkalmaztak. Az eredményeik többnyire jónak mondhatóak, kisebb hibák azonban jelen vannak, többnyire az épületek kontúrjainál.

Nesrine Chehata és társai [11] *random forest* osztályozót alkalmaztak, többféle magasságalapú, echó alapú, egységvektor alapú, lokális sík alapú, valamint teljes jelalakra épülő attribútumokat határoztak meg. Végül az osztályozáshoz 17 attribútum használtak és összesen 5,65%-os, valamint 4,99%-os hibát értek el különböző paraméterekkel.

J. Niemeyer és társai [12] szintén gépi tanulást használtak és az előzőekhez hasonló attribútumhalmazzal, azonban ők *Conditional Random Field, CRF* osztályozót használtak. Az osztályaik között szerepelt a természetes és mesterséges talaj, alacsony növényzet, fák és épületek. Nagyon jó eredményeket értek el, azonban itt is előfordult, hogy a fákat és az épületeket a hasonló tulajdonságuk miatt összekeverte az osztályozó.

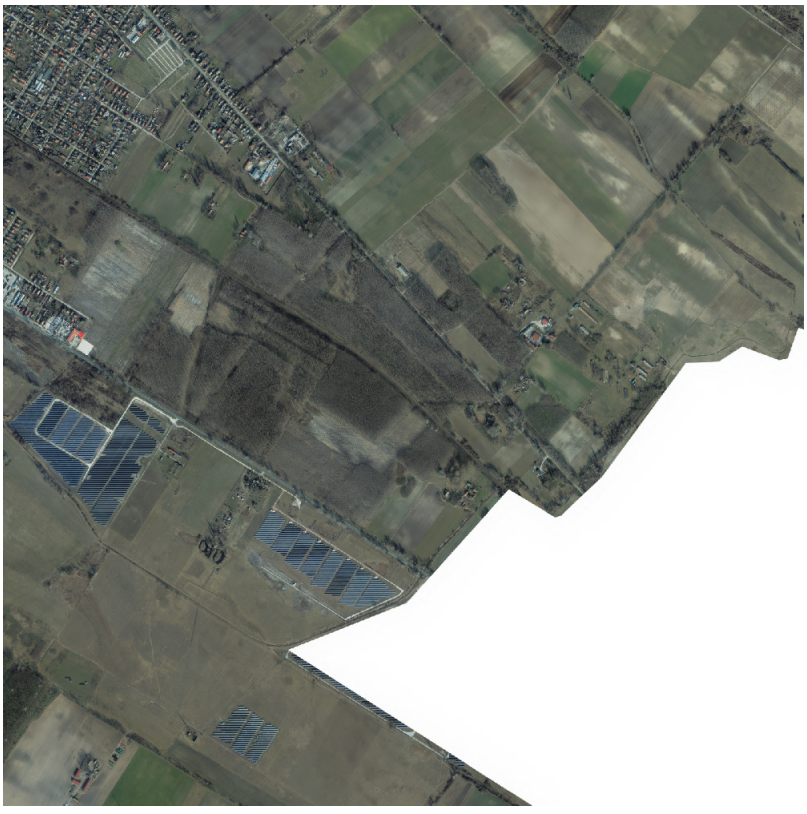
Bao Yunfei és társai [13] magasságértékek és intenzitás segítségével osztályoztak erdőkről készült pontfelhőket. A céljuk DTM generálása volt, de a feladatot felfoghatjuk talaj és növényzet osztályozásként is. Többnyire jó eredményeket értek el, azonban nem minden növényzethez tartozó pontot sikerült kiszűrniük.

3. fejezet

Módszertan

3.1. Felhasznált pontfelhők

A tanításhoz és a teszteléshez a Lechner Tudásközponttól kapott monori pontfelhőket használom. Ez összesen 23 csempéből áll, aminek a nagy része pusztás vagy erdős terület, de Monor városának egy része is megtalálható egyes felhőkben. Ezek a 7-es pontformátumot használják. A nagyjából 10 km^2-es terület ortofotója a 3.1 ábrán látható.



3.1. ábra. A monori terület ortofotója.

3.2. Pontfelhő attribútumai

A pontok koordinátái

Minden ponthoz meg van határozva a térbeli (X, Y, Z) koordinátája az adott vetületi rendszerben. Fontos tisztázni, hogy a Z koordináta határozza meg a magasságot. A koordináták fontos szerepet játszanak, hiszen ezek alapján tudjuk hatékonyan eltárolni a pontokat egy erre alkalmas adatszerkezet segítségével.

Intenzitás (*Intensity*)

Az intenzitás egy olyan természetes szám, ami a visszatérő lézer impulzus nagyságrendjét reprezentálja. Ez az érték rendszerspecifikus, valamint segíthet különböző anyagokat meghatározni.

Visszatérések száma (*Return number*)

A visszatérések száma meghatározza, hogy az adott impulzus esetén ez hanyadik visszaverődés volt. Az első visszaverődés esetén az érték 1. A 0-5 pontformátumok maximum 5, a 6-10 maximum 15 visszatérési értéket tud eltárolni. Az érték minidig 1 és az Összes visszatérés száma közé esik (zárt intervallum). Az első visszaverődést általában a magas épületekről és növényzetről érkezik, míg az utolsó általában a talajról, így ez az attribútum elengedhetetlen az osztályozáshoz.

Összes visszatérés száma (*Number of returns*)

Az összes visszatérés száma megadja, hogy egy adott impulzus esetén összesen hány visszatérésünk volt. A 0-5 pontformátumok esetén ez az érték maximum 5, a 6-10 formátumok esetén pedig 15.

Osztály

Egyes pontfelhők már korábban osztályozva lettek. Ennek a tárolására szolgál az osztály attribútum. Ezt csak a tanításhoz használjuk fel. A lehetséges értékekről és jelentésükről a 2.1 és a 2.2 táblázatok alapján tájékozódhat.

3.3. Kiszámított attribútumok

Talaj feletti magasság

Minden pontnak meghatározzuk, hogy nagyjából mi a talaj feletti magassága. Ezt úgy tudjuk elérni, hogy készítünk egy DTM-et a területről, és az adott ponthoz tartozó cella értékéből kivonjuk a pont magasságát. A DTM előállításáról a 3.4.3 fejezetben olvashat.

MinDSM feletti magasság

Minden pontra meghatározzuk, hogy mi a magassága egy speciális DSM-hez képest. Ezt a DSM-et *Minimum Height Map*-nek hívják, és az előállításáról a 3.4.2 fejezetben olvashat.

A magasság különbsége

Minden pontra kiszámítjuk, hogy egy R-sugarú környezetében mi a legmagasabban és legalacsonyabban elhelyezkedő pont magasságának a különbsége.

A magasság szórása

Minden pontra kiszámítjuk, hogy a pont R-sugarú környezetében mi a magasságértékek szórása.

Az intenzitás szórása

Minden pontra kiszámítjuk, hogy a pont R-sugarú környezetében mi az intenzitásértékek szórása.

A pontok sűrűsége

Minden pontra kiszámítjuk, hogy a pont R-sugarú környezetében mi a pontok sűrűsége. Ezt úgy érjük el, hogy a megnézzük, hogy a környezetben hány pont helyezkedik el és ezt elosztjuk az R négyzetével.

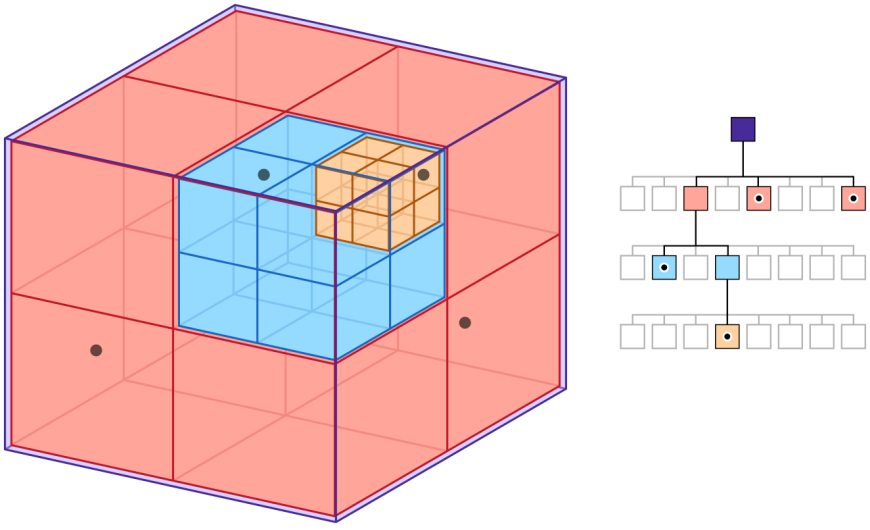
3.4. Felhasznált módszerek

3.4.1. Nyolcadoló-fa

Ha pontokat egy átlagos listában tárolnánk el, akkor csak lineárisan tudnánk megkeresni egy adott pontot, ami egy 17 millió pontól álló ponthalmaz estében borzasztóan lassú lenne, ezért erre problémára az indexelés a megoldás. Ezek az adatszerkezetek valamilyen rendszer szerint rendezetten tárolják el az adatokat, így célirányosan tudunk bennük keresni, így egy pont megkeresése logaritmikus műveletigényű lesz.

Egydimenziós indexelés-re az egyik legjobb adatszerkezet a B+-fa. Ezt használja a legtöbb adatbázis.

Többdimenziós esetben többféle megközelítés is létezik, úgynevezett területalapú, illetve adatvezérelt módszerek. Én a nyolcadoló-fát (angolul *Octree*) [14] [15] használtam ami az egyik legnépszerűbb háromdimenziós területalapú indexelő adatszerkezet. A 2 dimenziós változatát negyedő-fának (angolul *Quad Tree*) nevezik. Ezek a fák a teret mindig 2^d részre osztják (ahol d a dimenzió száma), innen származik az elnevezés, hiszen 2D-ben 4, 3D-ben 8 részre osztja.

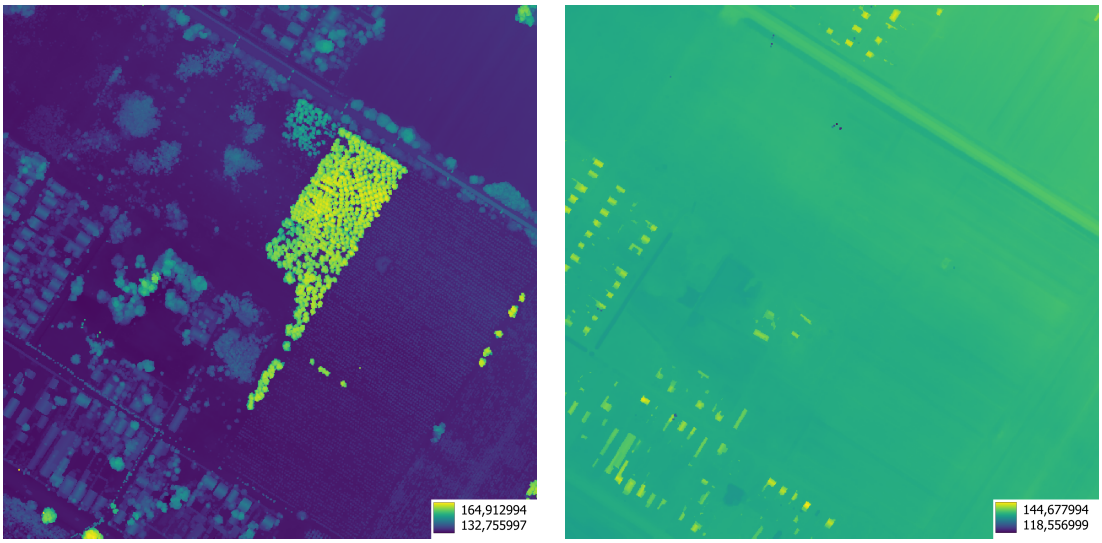


3.2. ábra. A nyolcadolófa működése. Forrás: [16]

3.4.2. *Minimum Height Map* előállítása

Korábban említettem, hogy a DSM-eket interpoláció segítségével számítják ki. A legnépszerűbb megoldások között van az *Inverse Distance Weighting*[17], ami a távolság

alapján súlyozza a magasságértékeket, vagy a maximumkiválasztásos módszer, ami minden cellának azt az értéket adja, ami a cella területén legmagasabban elhelyezkedő pont magassága. A *minimum height map* az utóbbi megoldás ellentéte, azaz a legmagasabb pont helyett, a legalacsonyabb pont magasságával számol. A LiDAR többszörös visszatérésének és különböző anyagokon való áthatolásának köszönhetően, ezzel a módszerrel egy olyan kezdeti DTM-et állítottunk elő, ami többnyire már csak a talaj és az épületek látszanak. A *Minimum Height Map* és egy hagyományos DSM közötti különbséget a 3.3 ábrán is láthatja.



(a) Maximumkiválasztással készült DSM. (b) Minimumkiválasztással készült DSM.

3.3. ábra. A monori B1 területről minimumkiválasztással készült DSM-ek.

3.4.3. DTM generálása

A talaj feletti magasság kiszámításához először egy DTM-et kell generálnunk. DTM készítésére többféle módszer is van: például a pontfelhő bináris osztályozása (talaj és nem-talaj pontok) és ezután egy DSM-et generálunk a talajpontok felhasználásával. Egy másik megoldás lehet, hogy először generálunk egy DSM-et és ezen döntjük el, hogy mely cellák tartoznak a talajhoz. Az én megoldásom az utóbbi módszert használja és a generált DSM egy *Minimum Height Map*.

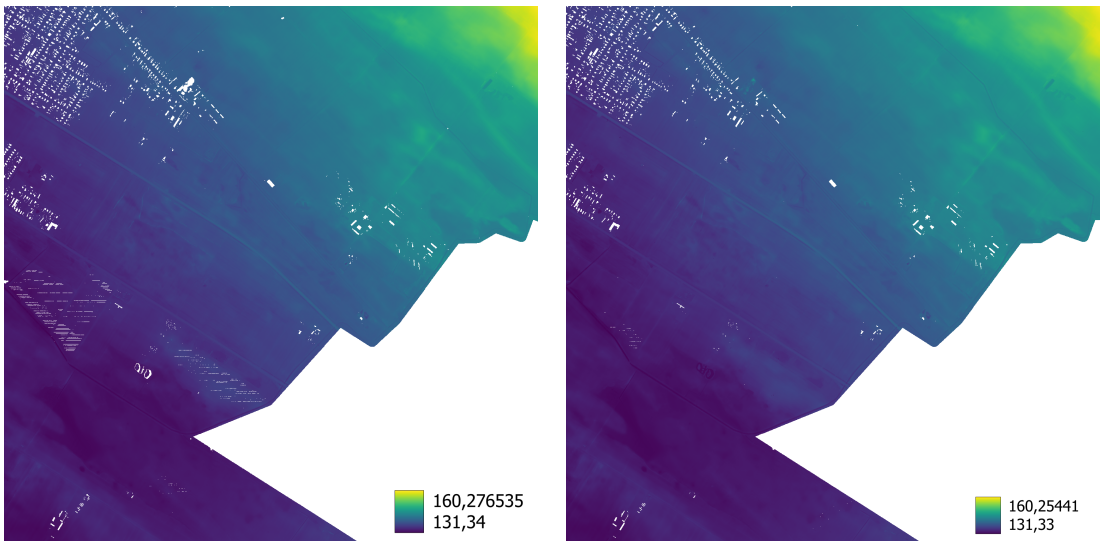
Ezután az épületek eltüntetésére több megoldás is létezik. Carla Nardinocchi és társai [18] egy szegmentálás alapú módszert fejlesztettek ki, ami az egyes szegmensek (azaz a magasságértékek alapján egy csoportba tartozó cellák) elhelyezkedése és geometriai tulajdonságuk alapján osztályozták a szegmenseket. Az én megoldásomat is ez a kutatás

ihlette, aminek az az alapelve, hogy a talaj szegmensei lejjebb helyezkednek el az épületek szegmenseihez képest.

A első a lépés a generált DSM-en a szegmensek megtalálása. Ehhez először megkeressük a raszteren a legalacsonyabb még fel nem dolgozott cellát, ez lesz a szegmens kezdőpontja. Ezután rekurzívan felfedezzük ennek a cellának a szomszédjait, és ha két cella magasságának a különbsége kisebb mint 0,3, akkor hozzávesszük a szegmenshez. Ezt addig ismételjük amíg az összes cellát fel nem fedeztük. Ezután a kis méretű szegmenseket összevonjuk a nagyobbakkal. Erre azért van szükség, hogy például az épületek kontúrjait ne a talajhoz számítsa.

Ezután kiszámoljuk, hogy a DSM celláinak hány százaléka kisebb mint a cellák összértékének átlaga. Ez a paraméter ideális esetben azt adja meg, hogy mi az talaj és az épületek aránya. Ezután alulról felfele addig vonjuk össze a szegmenseket amíg a cellák mennyisége el nem éri az összterület és az imént kiszámolt arány szorzatát.

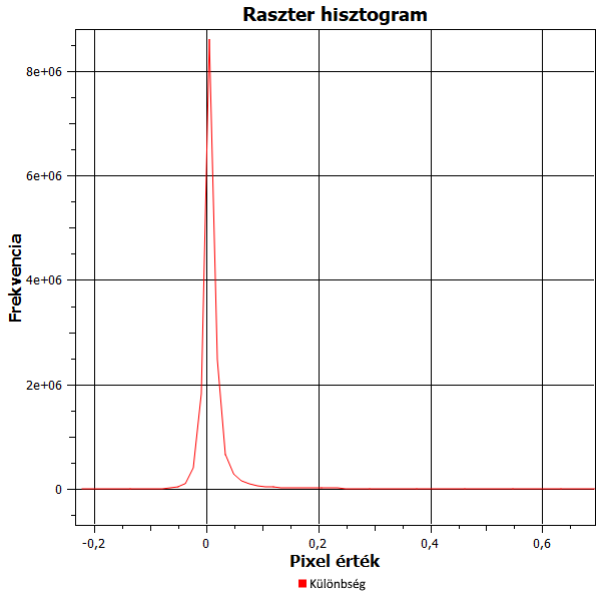
Ezzel a módszerrel a legtöbb esetben egy pontos DTM-et tudunk generálni LiDAR pontfelhőkből. A monori területen is teszteltem ezt a pontfelhőt, a módszerem által generáltat és a LiDAR osztályozás alapján készültet a 3.4 ábrán lehet megtekinteni. Az eredmények különbségét pedig a 3.5 ábrán lehet látni. Jelentősebb eltérések a talajra helyezett napelemek, sekély tavak, valamint mesterséges dombok (például sódertelep) esetén voltak.



(a) Az osztály alapján készült DTM.

(b) A szegmentálás alapú módszerrel készült DTM.

3.4. ábra. A monori területről készült DTM-ek.



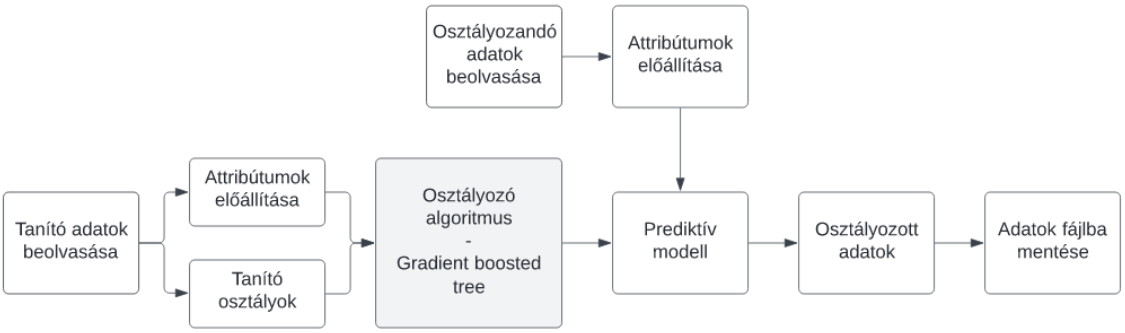
3.5. ábra. A két DTM különbségét ábrázoló hisztogram.

3.4.4. Felhasznált osztályozó és klaszterizáló módszerek

A 2.3. fejezetben leírt módszerek közül végül csak a k-közép és a *gradient boosted tree*-t használtam. A k-közép megoldást az egyszerűsége miatt tartom jobb megoldásnak, a *gradient boosted tree*-t pedig, azért mert a legtöbb esetben pontosabb és hatékonyabb, mint a *random forest* osztályozó. A döntésemet az is befolyásolta, hogy implementációs szempontból elég volt csak egy külső könyvtárat használnom.

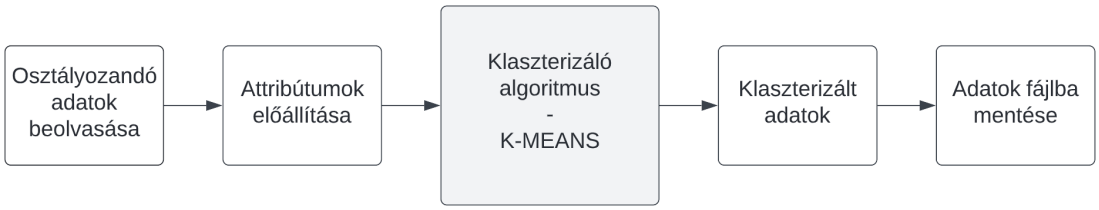
A gépi tanulás során használt attribútumok (ún. *feature*-ök) az összes kiszámított attribútum ami a 3.3. fejezetben található, valamint a pontfelhő attribútumok közül a visszatérések száma és az intenzitást használtam fel. Az *gradient boosted tree* esetén osztályozáshoz az osztály attribútumot is használjuk a tanításhoz. A tanításhoz csak azokat a pontokat használom fel, amik a 2-6 osztályokba esnek. Ekkor a többi pontot nem vesszük figyelembe.

A felügyelt osztályozás során a célunk egy modell felépítése, ami képes előrejelezni egy ismeretlen adat osztályát. A 3.6 ábra megmutatja a felügyelt osztályozás folyamatát, ahol először beolvassuk az adatokat, majd a nyers adatokból előállítjuk az attribútumokat és az osztályt. Ezután átadjuk ezeket az adatokat az algoritmusnak, ami készít egy modellt, valamint egy prediktív modellt. Ezután az osztályozandó adatokat is beolvassuk, feldolgozzuk, majd a prediktív modell segítségével osztályozzuk, majd végül elmentjük a fájlba.



3.6. ábra. Az osztályozás folyamatábrája.

A klaszterezés során beolvassuk a nyers adatokat, előállítjuk az attribútumokat, majd átadjuk őket a klaszterező algoritmusnak. A klaszterezés befejeztével az adatokat elmentjük. Ezt a folyamatot szemlélteti a 3.7 ábra.



3.7. ábra. A klaszterezés folyamatábrája.

4. fejezet

Implementáció

Az elkészült szoftver forráskódja az alábbi linken érhető el: `https://gitlab.inf.elte.hu/lechner/summer-lab-2021/project-2/-/tree/master/LiDAR%20Classification`

Bár a gépi tanuláshoz a leggyakrabban alkalmazott programozási nyelv a python, én mégis a C# nyelven dolgoztam. A választásomnak több oka is van.

A pythont azért használják, mert nagyon sok gépi tanulással kapcsolatos csomag érhető el, amelyek gyorsak és hatékonyak. A C#-nak is van egy gépi tanulásra fejlesztett könyvtára, ez pedig az ML.NET. Ez a csomag is támogatja a *TensorFlow*-t és más ML keretrendszereket, valamint a Microsoft szerint gyorsabb és pontosabb is mint a *stickit-learn*, ami a legnépszerűbb pythonos ML csomag.

Egy másik fontos tényező, hogy az C#-ban már korábban implementáltam több adatszerkezetet és módszereket LiDAR pontfelhők feldolgozásához. Ezek megtalálhatóak az ELTE Informatika Karának saját fejlesztésű térinformatikai keretrendszerében az AEGIS-ben. Többek között van lehetőségünk a pontfelhőt egy nyolcadolófában eltárolni, valamint a zajpontok detektálására, DSM, DTM és CHM készítésére, valamint mintavételezésre is.

4.1. ML.NET

Az ML.NET a C# egyik legnépszerűbb gépi tanulást segítő csomagja. Az ML.NET alapja a gépi tanulás modellje. A modellt több különböző algoritmus segítségével is

lehet tanítani, így olyan feladatok megoldására is használható, mint az osztályozás, klaszterezés, regresszió vagy anomália detektálás. Az ML.NET képes előre betanított *TensorFlow* és *ONNX* modelleket is kezelni. Klaszterezésből csak a k-közép algoritmust tartalmazza, viszont bináris osztályozásnál döntési fák, *random forest* és *gradient boosted tree* is elérhető, többosztályos osztályozásnál pedig támogatott többek között a *gradient boosted tree* és a naiv bayes osztályozó.

4.2. *Light Gradient Boosting Machine*

A *LightGBM*[19] a *Light Gradient Boosting Machine* rövidítése, ami egy döntési fákra épülő *gradient boosting* keretrendszer. A nyílt forráskódú programot eredetileg a Microsoft fejlesztette és többféle gépi tanulásos feladat megoldására is alkalmas, mint például a regresszió vagy az osztályozás. A keretrendszer a legtöbb programozási nyelven elérhető, mint a Python, az R, a C, illetve a C#, utóbbi esetén az ML.NET-be is be van építve. Ez a keretrendszer elvileg egy gyorsabb, hatékonyabb, valamint pontosabb és kevesebb memóriát használ, mint a vetélytársai.

4.3. AEGIS keretrendszer

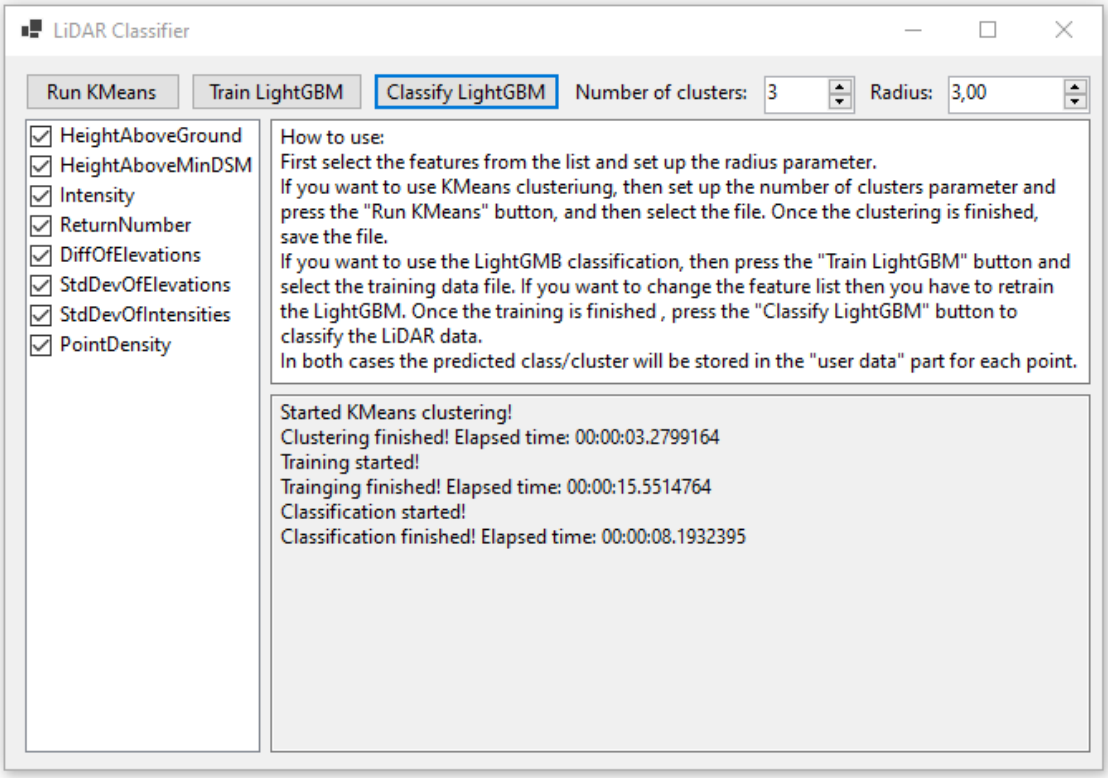
Az AEGIS keretrendszer az ELTE által fejlesztett térinformatikai keretrendszer C# nyelven [20, 21]. A rendszer támogat többféle geometriai és térinformatikai adatot, többek között vektoros adatszerkezeteket, raszteres képállományokat és LiDAR pontfelhőket is. Támogatott továbbá többféle adatszerkezet, feldolgozó algoritmus, valamint az adatok fájlból történő beolvasása és fájlba mentése.

A TDK dolgozattal egy időben készülő szakdolgozatom [22] keretében implementáltam több LiDAR pontfelhő feldolgozó algoritmust, mint a DSM-, DTM-, CHM generálás, a zajpontok detektálása vagy részminta generálása, valamint a pontok eltárolására alkalmas negyedelő-, és nyolcadoló-fát.

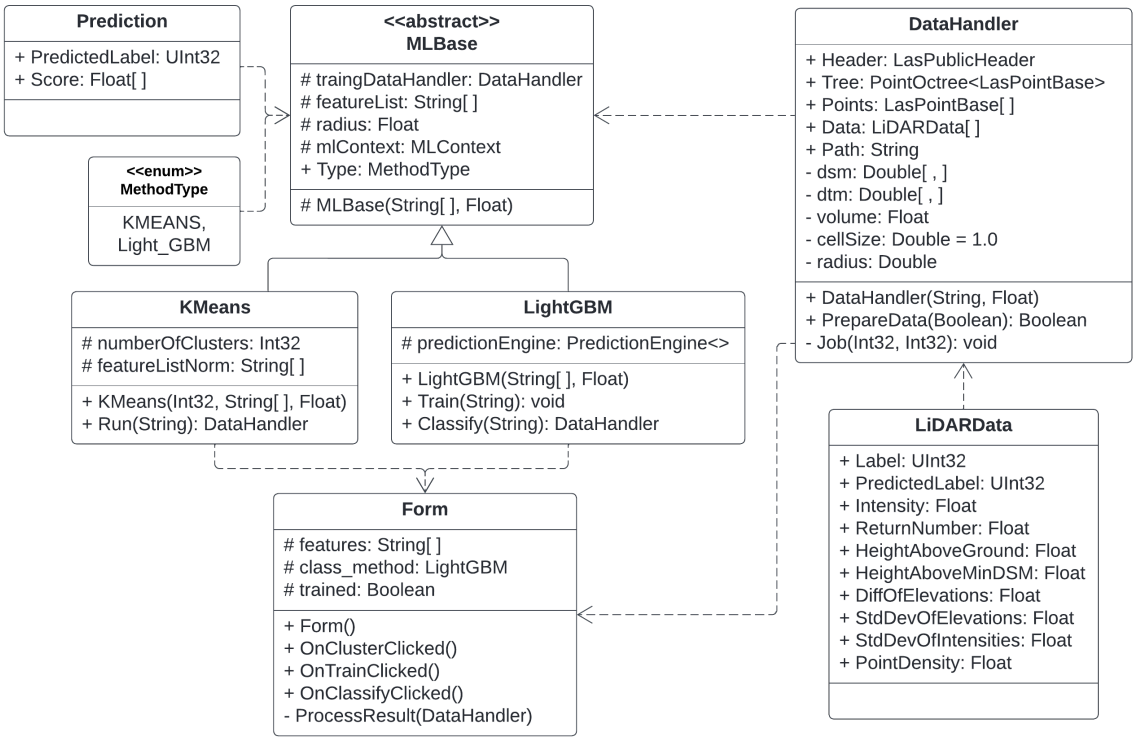
4.4. Az alkalmazás felépítése

A .NET segítségével egy *Windows Forms* alapú ablakos alkalmazást is készítettem a LiDAR pontfelhők osztályozásához (lásd 4.1 ábra). Ez az alkalmazás lehetőséget

biztosít arra is, hogy az osztályozott pontfelhőket elmentsük. Az alkalmazás UML osztálydiagramja a 4.2 ábrán látható.



4.1. ábra. A grafikus alkalmazás.



4.2. ábra. A alkalmazás UML osztálydiagramja.

A *Form* osztály felel a grafikus ablak megjelenéséért, valamint a gombok eseménykezeléséért és az eredmények elmentéséért.

Az *MLBase* osztály egy általános osztály, amely leírja azokat az adattagokat, amelyek a gépi tanulással kapcsolatos feladatok megoldásához szükségesek. A *KMeans* és *LightGBM* ennek az osztálynak a leszármazottjai, ezek felelnek a klaszterezésért és az osztályozásért.

A *Prediction* osztály tárolja a el a folyamatok kimeneti adatai, míg a *LiDARData* osztály a folyamatok bemeneti adatait (felhasznált attribútumokat) tárolja minden pontra.

A *DataHandler* osztály felel a pontfelhő beolvasásáért, tárolásáért és az attribútumok előállításáért.

4.5. Az alkalmazás használata

Az alkalmazás használatához az első lépés a tanító attribútumok kiválasztása, valamint a *radius* paraméter beállítása. A *radius* paraméter adja meg a kiszámított attribútumok esetében a környezet sugarát.

Ha klaszterizálni szeretnénk a pontfelhők adjuk meg a klaszterek számát és kattintsunk a "Run KMeans" gombra. Ekkor válasszuk ki a beolvasandó fájlt. Miután befejezte a klaszterezést egy mentési ablak fog megjelenni, hogy a klaszterizált felhőt elmenthessük. Minden pontnak a *userdata* attribútumában lesz a kapott klaszter száma.

Ha osztályozni szeretnénk a pontfelhőket, akkor ahhoz először be kell tanítani az osztályozót. Ehhez kattintsunk a "Train LightGBM" gombra, majd válasszuk ki a tanító felhőt. Ha megváltoztatjuk a kiválasztott attribútumokat, akkor újra be kell tanítani az osztályozót. A tanításhoz csak a 2-6 osztályú pontok lesznek figyelembe véve. A tanítás végeztével kattintsunk a "Classify LightGBM" gombra, hogy futtassuk az osztályozást. Ekkor ismét ki kell választani az osztályozandó felhőt. A folyamat végeztével egy mentési ablak fog megjelenni. Minden pontnak a *userdata* attribútumában lesz a kapott osztálya. Az optimális eredmény érdekében érdemes a tanító adatnak ugyanazzal a szenzorral készülnie, mint az osztályozandó felhő, hiszen az olyan paraméterek mint a visszatérések száma és az intenzitás szenzorfüggő adat.

5. fejezet

Eredmények

A monori terület városias, erdős és pusztás területből áll. Mivel utóbbi az osztályozás szempontjából nem kifejezetten izgalmas, ezért a tesztelést két kategóriára bontottam: városi és erdős.

5.1. Felhasznált területek

Mivel egyes pontfelhők nagyon sok pontból állnak, ezért nem egy teljes pontfelhőt használok, hanem azokból kivágott kisebb pontfelhőket. A fájlok nevében a betű-szám kombináció a csempe nevére utal amiből kivágtam. A felhasznált fájlok adatai és betöltött szerepei a 5.1 táblázatban tekinthető meg. Ezek a fájlok elérhetőek a `https://krr0land.web.elte.hu/TDK/` weboldalon.

Fájl neve	Pontok száma	Terület típusa	Klaszterezés	Osztályozás
A1_city.laz	2186269	városi	tesztelő	tanító
A2_city.laz	2652289	városi	-	tesztelő
B1_city.laz	1080782	városi	-	tesztelő
B1_forest.laz	325882	erdős	tesztelő	tanító
C3_forest.laz	502343	erdős	-	tesztelő

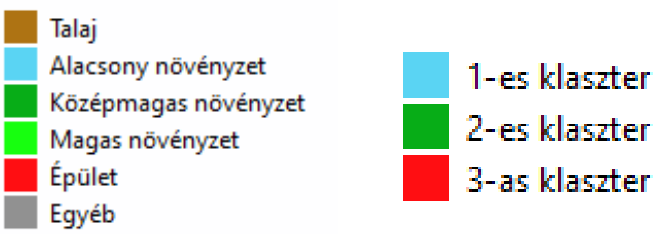
5.1. táblázat. A felhasznál fájlok és szerepük.

5.2. Színek jelentése

A továbbiakban látott ábrákon az egyes színek különböző osztályokat, valamint klasztereket fognak jelölni. A 5.1 ábra mutatja vizuálisan ezen színek jelentését.

Osztályozás esetén a barna a talajt, a világos kék az alacsony növényzetet, a sötét zöld a középmagas növényzetet, a világos zöld a magas növényzetet, a piros az épületeket, a szürke pedig az egyéb pontokat szemlélteti.

Klaszterezésnél a kék az 1-es sorszámú, a zöld a 2-es sorszámú és a piros a 3-as sorszámú klasztert szemlélteti.



(a) Az osztályozás színeinek jelentése.

(b) A klaszterezés színeinek jelentése.

5.1. ábra. Az eredmények bemutatásához használt színek magyarázata.

5.3. Városias területek

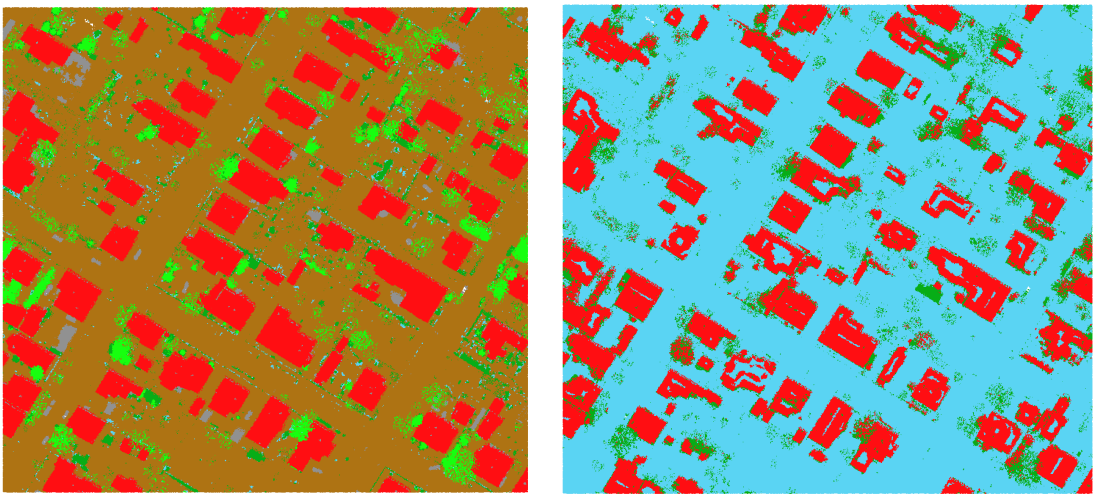
5.3.1. Klaszterezés eredménye

Városi területeként a klaszterezés során az A1 pontfelhő középső részét (*A1_city.laz*) használtam a teszteléshez. A klaszterezés eredménye és a terület eredeti osztályozása a 5.2 ábrán látható, valamint a 5.2 táblázat leírja a klaszterezés tévesztési mátrixát. Az első sor a klaszterek sorszámát, az első oszlop az osztályokat jelöli.

Az eredményeken látszódik, hogy a talaj az 1-es klaszterbe, a növényzet a 2-es klaszterbe és az épületek pontjai a 3-as klaszterbe kerültek. Azonban az eredmény nem tökéletes, hiszen a növényzetben is felfedezhető több 2-es klaszterbe sorolt pont, valamint az épületek tetején is megfigyelhetőek 1-es klaszterbe tartozó pontok. A pontosság megközelítőleg $85,8\%$.

	1	**2**	**3**
Egyéb	17447	24867	22677
Talaj	1278230	32474	6917
Al. növ.	20605	2397	429
Km. növ.	43728	69084	22458
M. növ.	23	151425	49584
Épület	31617	37640	374667

5.2. táblázat. Városi területen a klaszterezés tévesztési mátrix.



(a) A terület eredeti osztályozása. (b) A terület elkészített klaszterezése.

5.2. ábra. Az klaszterezés eredménye városias területen.

5.3.2. Osztályozás eredménye

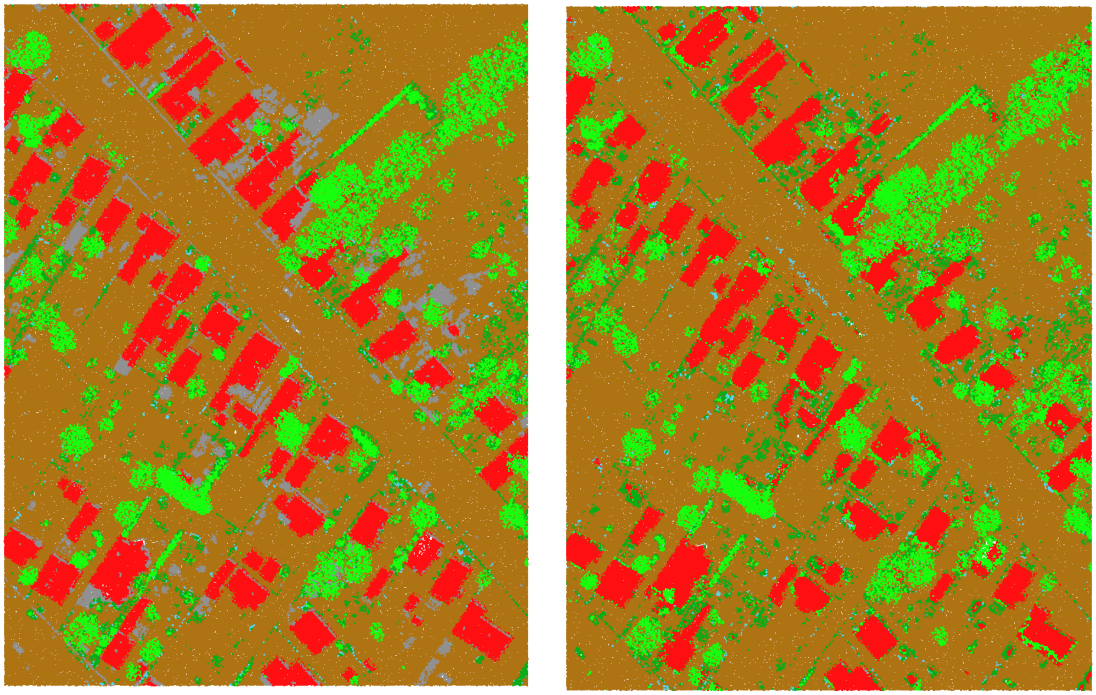
Az osztályozás során tanító területnek a *A1_city.laz* felhőt használtam, aminek az osztályozása a 5.2.a ábrán látható. A teszteléshez a *B1_city.laz* és *A2_city.laz* felhőket használtam.

A2_city.laz

A 5.3 ábrán látható a tesztelt terület eredeti osztályozása és az elkészült osztályozás, a 5.3 táblázaton pedig az osztályozás tévesztési mátrixa. A táblázatból is leolvasható, hogy az osztályozás pontossága $91,2\%$, a talaj, a magas növényzet és az épület kiemelkedően jó eredménnyel rendelkezik.

	Talaj	**Al. növ.**	**Km. növ.**	**M. növ.**	**Épület**	**Pontosság**
Egyéb	36220	7772	64514	7300	16696	-
Talaj	1752986	8805	3524	345	3337	99,1%
Al. növ.	9779	4187	1223	3	109	27,4%
Km. növ.	2615	1099	92130	2573	4680	89,4%
M. növ.	2	14	1342	224740	11041	94,8%
Épület	212	42	17812	31160	346027	97,5%

5.3. táblázat. Városias területen az osztályozás tévesztési mátrixa az *A2_city.laz* pontfelhőn.



(a) A terület eredeti osztályozása. (b) A terület elkészített osztályozása.

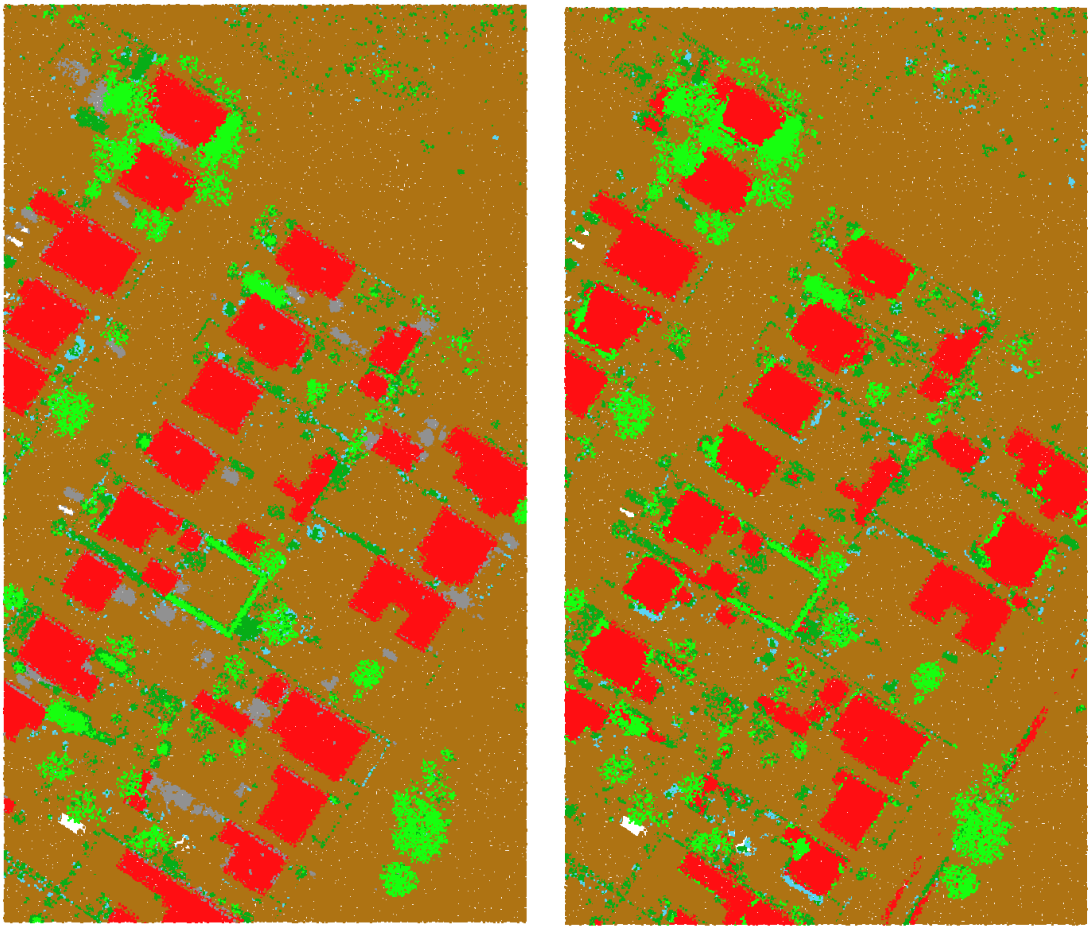
5.3. ábra. Az osztályozás eredménye városias területen az *A2_city.laz* pontfelhőn.

B1_city.laz

A 5.4 ábrán látható a tesztelt terület eredeti osztályozása és az elkészült osztályozás, a 5.4 táblázaton pedig az osztályozás tévesztési mátrixa. A táblázatból is leolvasható, hogy az osztályozás pontossága 93,7%, a talaj, a magas növényzet és az épület kiemelkedően jó eredménnyel rendelkezik.

	Talaj	**Al. növ.**	**Km. növ.**	**M. növ.**	**Épület**	**Pontosság**
Egyéb	5321	1134	13658	2291	4889	-
Talaj	758947	4607	2831	1	2095	98,8%
Al. növ.	4979	2643	847	0	54	31,0%
Km. növ.	1244	713	40316	863	2912	87,6%
M. növ.	0	0	481	57626	2668	94,8%
Épület	112	6	2535	13515	153494	90,5%

5.4. táblázat. Városias területen az osztályozás tévesztési mátrixa a *B1_city.laz* pontfelhőn.



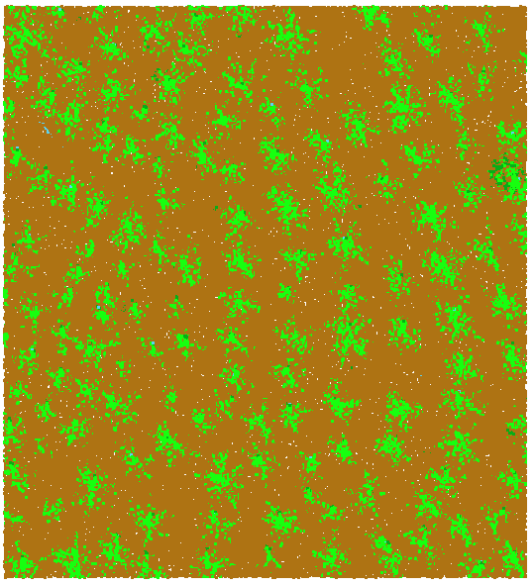
(a) A terület eredeti osztályozása. (b) A terület elkészített osztályozása.

5.4. ábra. Az osztályozás eredménye városias területen a *B1_city.laz* pontfelhőn.

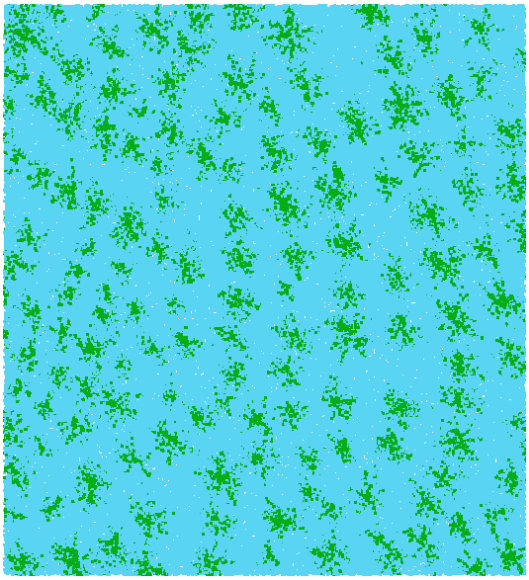
5.4. Erdős területek

5.4.1. Klaszterezés eredménye

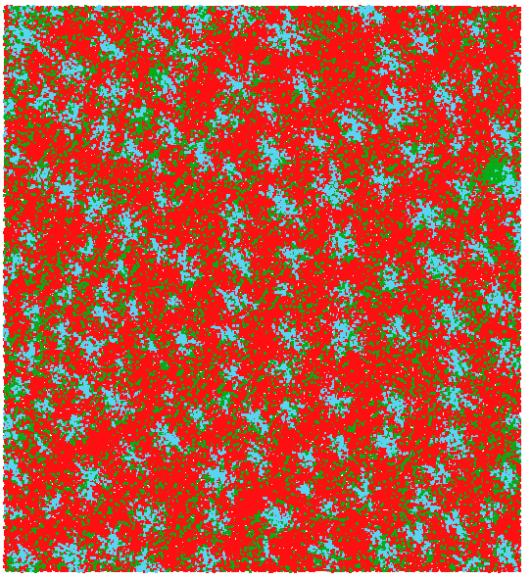
Erdős területek esetén a B1 pontfelhő erdős részét (*B1_forest.laz*) használtam a teszteléshez. Futtattam 2, valamint 3 klaszterre is. A terület eredeti osztályozása, valamint a klaszterezés eredményei a 5.5 ábrán látható. Továbbá a 5.5 és a 5.6 táblázatok leírják, hogy milyen osztályú pontok kerültek mely klaszterekben. Az első sor a klaszterek sorszámát, az első oszlop az osztályokat jelöli. A pontosság megközelítőleg 99% két klaszter esetén, három esetén pedig 81,7%.



(a) A terület osztályozása.



(b) A klaszterezés 2 klaszter esetén.



(c) A klaszterezés 3 klaszter esetén.

5.5. ábra. Az klaszterezés eredménye erdős területen.

	1	**2**
Talaj	246192	0
Al. növ.	270	0
Km. növ.	2592	0
M. növ.	322	76506

5.5. táblázat. Erdős területen 2 klaszter esetén a tévesztési mátrix.

	1	2	3
Talaj	0	59126	187066
Al. növ.	0	142	128
Km. növ.	0	2513	79
M. növ.	76422	406	0

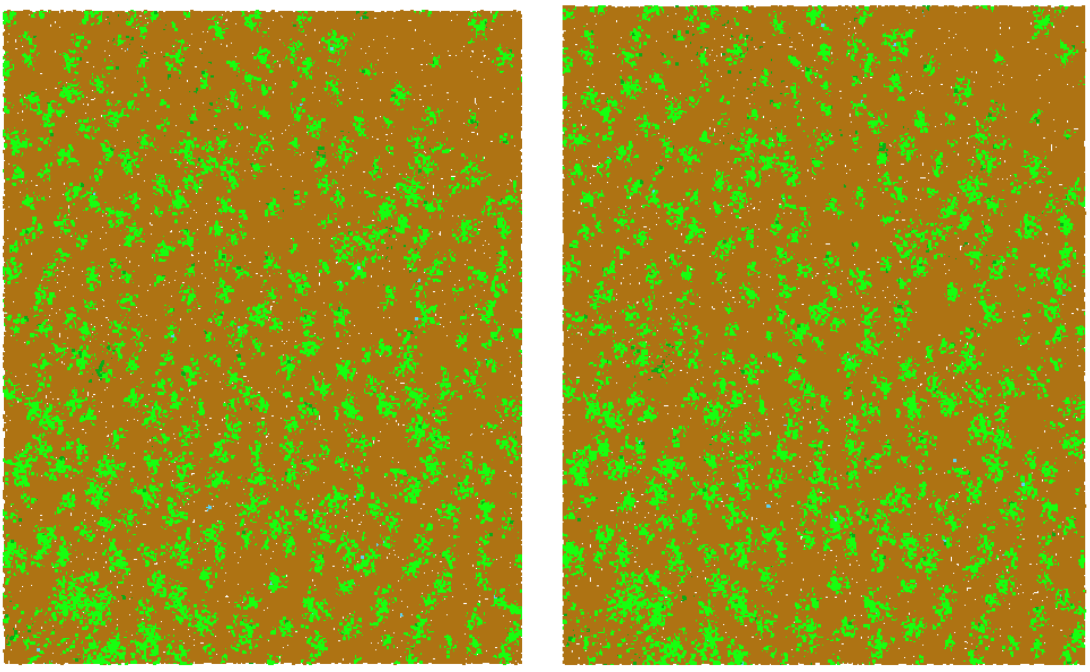
5.6. táblázat. Erdős területen 3 klaszter esetén a tévesztési mátrix.

2 klaszter esetén nagyon jól elkülönül, hogy többnyire a talaj és a fák alja került az 1-es számú klaszterbe, míg a 2-es klaszter a középmagas és magas növényzetből állt.

3 klaszter esetén az 1-es klaszter lett a fák teteje, a 2-es klaszter a fák alsó része és a talaj egy része, a 3-as pedig a talajpontok nagy része.

5.4.2. Osztályozás eredménye

Az osztályozás során a tanító területnek a *B1_forest.laz* felhőt, teszteléshez pedig a *C3_forest.laz* felhőt használtam. Az 5.5.a ábrán látható a tanító terület osztályozása. A 5.6 ábrán látható a tesztelt terület eredeti osztályozása és az elkészült osztályozás, a 5.7 táblázaton pedig az osztályozás tévesztési mátrixa.



(a) A terület eredeti osztályozása. (b) A terület elkészített osztályozása.

5.6. ábra. Az osztályozás eredménye erdős területen.

A táblázatból is leolvasható, hogy az osztályozás pontossága $99,3\%$, a talaj és a magas növényzet kiemelkedően jó eredménnyel rendelkezik.

	Talaj	**Al. növ.**	**Km. növ**	**M. növ.**	**Pontosság**
Talaj	421390	929	786	1042	99,3%
Al. növ.	141	47	88	4	16,8%
Km. növ	277	70	2487	107	84,6%
M. növ.	2	1	48	74924	99,9%

5.7. táblázat. Erdős területen az osztályozás tévesztési mátrixa.

5.5. Futási idők

Általánosságban elmondható, hogy kisebb pontfelhőkre gyorsan lefutnak a különböző módszerek, azonban nagyobb pontfelhők esetében sokkal tovább tartanak az egyes folyamatok. Az általam használt területeken a futási időt a 5.8 táblázat mutatja. A teszteléshez egy i7-8700-as processzorral és 16 GB RAM-mal rendelkező számítógépet használtam.

Fájl neve	**Pontok száma**	**Felhasználás**	**Idő**
A1_city.laz	2186269	klaszterezés	2p 18mp
A1_city.laz	2186269	osztályozó tanítás	1p 40 mp
A2_city.laz	2652289	osztályozás	4p 28mp
B1_city.laz	1080782	osztályozás	1p 36mp
B1_forest.laz	325882	klaszterezés	13 mp
B1_forest.laz	325882	osztályozó tanítás	14 mp
C3_forest.laz	502343	osztályozás	43 mp

5.8. táblázat. A futási idők a különböző fájlokra.

6. fejezet

Összefoglaló

A kutatás célja a pontfelhők osztályozására egy automatikus módszer fejlesztése és tesztelése volt, melyek a lehető legjobb eredményt adják. Viszonylag hamar nyilvánvalóvá vált, hogy ezt a gépi tanulás segítségével lehet a legjobban elérni.

Amennyiben nincsen osztályozott pontfelhőnk a tanításhoz, úgy jól megválasztott kezdőpontok esetén a klaszterezéssel megkaphatunk egy elsődleges osztályozást. A klaszterezés erdős területeken nagyon jól működik. Városias környezetben kisebb hibákkal, de meg tudja állni a helyét.

Ha már rendelkezünk előre osztályozott pontfelhővel akkor az osztályozást választva nagyon jó eredményeket érhetünk el. Ehhez azonban fontos, hogy az osztályozandó felhő és a tanító felhő lehetőleg azonos szenzorral készüljön, valamint hogy ezen területek hasonlóak legyenek. Ha egy erdős területet szeretnénk egy városi tanító területtel osztályozni, akkor az nagyon rossz eredményeket is adhat.

Továbbfejlesztési lehetőségként mindenféleképpen hasznos lenne az attribútumok csökkentése (dimenziócsökkentés). Ezáltal kiszűrnénk azokat az attribútumokat amik nem teszik egyedivé az adatokat, valamint gyorsítanák is az osztályozás folyamatát. Továbbá a klaszterezésnél a kezdőpontok megadása is hasznos funkció lehet.

Korábbi kutatásokkal ellentétben az osztályozáshoz én *gradient boosted tree*-t használtam, ami LiDAR pontfelhők osztályozására nem nagyon használt megoldás, valamint egy teljesen új attribútum (MinDSM feletti magasság) segítségével a korábbi eredményekhez hasonlót tudtam előállítani. Végül pedig elmondhatjuk, hogy az általam bemutatott módszerek segítségével automatizálni lehet a pontfelhők osztályozását.

Köszönetnyilvánítás

A kutatási munka és TDK dolgozat az ELTE Informatikai Kar[1] és az InforNess Training Kft.[2] ösztöndíjas pénzügyi támogatásával, továbbá a Lechner Tudásközpont[3] szakmai támogatásával valósult meg.

[1]ELTE Informatikai Kar - `https://inf.elte.hu/`
[2]InforNess Training Kft. - `https://www.informess.hu/`
[3]Lechner Tudásközpont - `https://lechnerkozpont.hu/`

Irodalomjegyzék

[1] Malgorzata Verőné Wojtaszek. *Fotointerpretáció és távérzékelés 3.* Nyugat-magyarországi Egyetem, 2010. URL: http://dtk.tankonyvtar.hu/xmlui/handle/123456789/7959.

[2] Celia García-Feced, Douglas J Tempel és Maggi Kelly. "LiDAR as a Tool to Characterize Wildlife Habitat: California Spotted Owl Nesting Habitat as an Example". *Journal of Forestry* 108.436–443 (2011).

[3] The American Society for Photogrammetry & Remote Sensing. *LAS Specification 1.4 - R15*. URL: http://www.asprs.org/wp-content/uploads/2019/07/LAS_1_4_r15.pdf.

[4] Martin Isenburg. "LASzip: lossless compression of LiDAR data". *Photogrammetric Engineering and Remote Sensing* 79 (2013). URL: https://downloads.rapidlasso.de/laszip.pdf.

[5] Marcela Carvalho. *Multi-Task Learning of Height and Semantics from Aerial Images*. https://deepai.org/publication/multi-task-learning-of-height-and-semantics-from-aerial-images. 2019.

[6] Zsolt Ilonczai. "Klaszter-analízis és alkalmazásai". Dipl. Eötvös Loránd Tudományegyetem, 2014.

[7] Andrew Ngai. *Understanding DBSCAN Algorithm and Implementation from Scratch*. https://towardsdatascience.com/understanding-dbscan-algorithm-and-implementation-from-scratch-c256289479c5. 2020.

[8] Hieu Tran. "Survey of Machine Learning and Data Mining Techniques used in Multimedia System". 2019.

[9] Zhongxing Zhang és tsai. "Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning". *Scientific Reports* 8 (2018).

[10] Angela Kim, Richard Olsen és F. Kruse. “Methods for LiDAR point cloud classification using local neighborhood statistics”. 8731. köt. 2013. máj., 873103. old. DOI: 10.1117/12.2015709.

[11] Nesrine Chehata, Li Guo és Clément Mallet. “Airborne LIDAR feature selection for urban classification using random forests”. *ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38 (2009. jan.).

[12] J Niemeyer, Franz Rottensteiner és Uwe Soergel. “Conditional random fields for lidar point cloud classification in complex urban areas”. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* I-3 (2012. júl.).

[13] Bao Yunfei és tsai. “Classification of Lidar Point Cloud and Generation of DTM from LiDAR Height and Intensity Data in Forested Area”. *ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 37 (2008. jan.).

[14] Hanan Samet. *The design and analysis of spatial data structures*. 85. köt. Addison-Wesley Reading, MA, 1990.

[15] Donald JR Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical és Systems Engineering Department, Rensselaer Polytechnic Institute, 1980.

[16] Saurab Dulal. *octree Construction and Nearest Neighborhood Search(NNS)*. https://dulalsaurab.github.io/computerscience/octree-construction-and-nns/. 2018.

[17] Donald Shepard. “A Two-dimensional Interpolation Function for Irregularly-spaced Data”. *Proceedings of the 1968 23rd ACM National Conference*. ACM ’68. New York, NY, USA: ACM, 1968, 517–524. old. DOI: 10.1145/800186.810616. URL: http://doi.acm.org/10.1145/800186.810616.

[18] Carla Nardinocchi, Gianfranco Forlani és Primo Zingaretti. “Classification and filtering of laser data”. *ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 34.3/W13 (2003).

[19] Guolin Ke és tsai. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. *Advances in Neural Information Processing Systems*. 30. köt. Curran Associates, Inc., 2017.

[20] Roberto Giachetta. “AEGIS-A state-of-the art component based spatio-temporal framework for education and research”. *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*. 13. köt. 1. 2013, 10. old.

[21] Roberto Giachetta. “A framework for processing large scale geospatial and remote sensing data in MapReduce environment”. *Computers & graphics* 49 (2015), 37–46. old.

[22] Roland Krisztandl. “LiDAR pontfelhő megjelenítő alkalmazás fejlesztése”. BSc szakdolgozat. 2022.

Ábrák jegyzéke

Táblázatok jegyzéke