A cikkben ismertetett osztályozási módszer csupán a LiDAR adatokkal dolgozik, nem alakítja át ezeket más formátumra (pl. DEM) és nem kombinálja más forrású adatokkal (pl. műholdfelvételek). A pontfelhő adatokon belül az első visszatérés (X,Y,Z)-koordinátáira és a mért intenzitásértékre támaszkodik.

Az osztályozás több lépésben történik, és először előzetes kategóriákba (alsó és felső kontúrok, egyenletes és nem egyenletes felületek) lesznek besorolva a pontok. A számításigény csökkentése érdekében a besoroláshoz csupán a pontok közvetlen szomszédjainak Z-koordinátában való eltérését kell vizsgálni, ahol a szomszédságot a pontok síkbeli Voronoi-diagramja határozza meg, ezért is lényeges, hogy a nem első visszatérést leíró LiDAR pontok ne legyenek ugyanúgy figyelembe véve. A diagram előállításához a C++-os boost library voronoi csomagját használtuk fel, ami a helyes kimenet létrehozásához egész koordináta-értékeket vár. A LAS-fájl fejléc információt áttekintve kiderült, hogy ez nyers (raw) elnevezésű adatok formájában amúgy is tárolva van, viszont mindez egyúttal azt is jelentette, hogy a módszer során szükséges távolság-vizsgálatoknál a megfelelő nagyságrend (scale) alkalmazására volt szükség, arra is felkészülve, hogy ez a transzformáció fájlonként változó lehet.

A felületek szétválasztásához iránymutatás híján két megközelítést is kipróbáltunk, melyek közül az összesített négyzetes eltérést alapul vevő küszöböt választottuk a pontonkénti konstans határhoz képest, ugyanis a nem egyenletes felületekre a továbbiak során nem támaszkodik a módszer, így a másodfajú hibát részesíthettük előnyben az elsőfajúhoz képest. A kontúrok esetén a cikk nem tért ki a felső és alsó kategóriába egyszerre beleeső pontokra. A tapasztalataink azt mutatták, hogy ezek a pontok vegyesen helyezkednek el főként a házfalakon és az amúgy is nagy pontsűrűségű lomkoronáknál, ezért nem vettük figyelembe őket a módszer további lépéseinél.

A cikkben szűkszavúan volt ismertetve, hogyan lehet az egyenletes felületek pontjai közül a háztetőket meghatározni. Az implementáció során felső kontúrpontokból induló mohó kiterjesztést alkalmaztunk. Azonban a pontok hosszú lehetséges szomszédsági láncolatai miatt a rekurzív megvalósítás nem jöhetett szóba, így egy iteratív megközelítést alkalmaztunk, ami ha nem is a legtökéletesebb megoldás, viszont nincs további memóriaigénye a kényszerűen memóriában tárolt Voronoi-diagramhoz képest, mint egy szélességi stratégiát használó algoritmusnak lett volna.

A másik nehézség ennél a lépésnél az volt, hogy a csupán két dimenziót felhasználva generált Voronoi-diagram nem várt szomszédsági viszonyai és az előforduló pontszerű outlier osztályozások miatt az épületek tetőpontjainak kiterjesztésénél nem sikerült csupán a fenti eszközökkel megfelelő megállási feltételt megfogalmazni. A vizsgált terület leíró adataiból adódó magasságértékekre vonatkozó korlátozásokkal a használt pontfelhőkön számottevő javulás volt

tapasztalható a kimenet alapján (ugyanakkor feltehetőleg ez korlátozást jelent speciális épület-konstrukciók felismerhetőségét illetően, viszont maga a cikk is azt sugallta, hogy a módszer alapvetően nagyvárosi környezetre vonatkozik, emeletes házakkal mint vizsgált épületek).

Felmerült lehetőségként, hogy a diagramon való szomszédság mellett az eredeti pontok magasságértékeit is össze lehetne hasonlítani, de ez egyrészt továbbra sem oldja meg a téves pontokból induló kiterjesztést, másrészt a feladat fókuszában alapvetően a cikk módszerének reprodukálása állt, és a sok saját optimalizálási ötlet beemelése vélelmezhetően nem feltétlenül ezt a cél szolgálta volna.

A tetőpontokból kiindulva kellett az épületekhez sorolandó összes pontot kiválasztani. Itt az épületszegmensek egyszerűsítésére és pontatlanságainak kiküszöbölésére a Voronoi-diagramra adaptált digitális képelemzésből ismert morfológiai műveleteket (nyitás és zárás) hajtottunk végre. A hagyományos értelmezés szerinti implementáció alkalmazásakor (vagyis minden szomszédot egyformán figyelembe véve) az alábbiakat tapasztaltuk:

- a nyitási művelet során az erózióval fontos épületrészek is elvesznek (pl. teraszok), melyek a dilatáció után nem sorolódnak vissza
- a pontok magasságából adódóan nem sikerült elkerülni, hogy a lombkoronák felületként beazonosított része ne minősüljön szintén tetőpontnak, így amennyiben az eróziót alkalmazásával az adott rész a nagy kiterjedése miatt nem tűnik el az egyszerűsítés részeként, akkor az a végeredményben is téves osztályozásként jelenik meg
- a tetőpontok és az épület alsó, valamint felső kontúrpontjainak összevonása nem valósul meg a Voronoi-diagram szomszédsági kapcsolatai miatt az ezt célzó külön dilatációs lépésben

Továbbfejlesztési lehetőségként érdemes megjegyezni, hogy jelenleg a morfológiai lépéseknél rögzítetten két szomszéd nagyságú "kernel-mérettel" dolgoztunk, ezt lehetne egy paraméterezhető változatra lecserélni. Emellett a lombkorona-épület tévesztések ellen egy szegmentálási lépés keretében a kis kiterjedésű tető-alakzatokat be lehetne azonosítani.

Az egyenletes felület kategóriából kellett képezni az utak pontjait is (az innen megmaradó pontok pedig implicit módon talajként lesznek értelmezve). Ennek a lépésnek a kulcsa az lett volna, hogy a pontok intenzitás értékeit megvizsgálva (statikus agy dinamikus módon) egy jól közelítő intenzitás-intervallumot tudjuk megállapítani. Számos különböző pontfelhőre futtatva a programot azt tapasztaltuk, hogy a konkrét területeket leíró adathalmaztól függően a maximális intenzitás rendkívüli módon szór (esetünkben nagyságrendileg 1000 és 60000 között), vagyis az egész terület intenzitás-intervallumából nem lehet általános módon leképezni az

utakat jellemző intervallumot. Meglepő volt továbbá, hogy a teljes intervallum méretét tekintve kismértékű (<1%) változtatások is drasztikusan hatottak az útként beazonosított pontok halmazára.

A fentiek alapján arra a következtetésre jutottunk, hogy az utak intenzitás-tartományát fájlonként manuálisan lehet leginkább meghatározni (és a holland településekre jellemző különböző színű aszfaltok együttes kezeléséhez nem is elegendő egyetlen intervallum), vagy esetleg a lépések felcserélésével az épületek valamekkora környezetét vizsgálva kell elindulni a teljes pontfelhő helyett (de ezt már nem jutott idő kipróbálni).

Az utak megkeresése egyben az a lépés is, ami a módszer futási idejét meghatározza. Ugyanis a cikk szerint csak az épületektől egy választott távolságon belüli pontok lehetnek utak (emiatt mondhatjuk azt, hogy a módszer (nagy)városi környezetre lett tervezve). Egy naiv implementációval ez négyzetes futási időhöz vezet, ami sok (száz)milliós pontfelhők esetén kivárhatatlan. Ezért mi megpróbáltuk a korábbi lépésekből származtatva az épületpontok egy olyan minél kisebb, de minél jobban reprezentáló részhalmazát elkülöníteni, ami gyorsíthat az algoritmuson. Ezzel már elfogadható eredményt sikerült elérni, azonban tudni kell, hogy sok épülettel rendelkező területetek esetén kisebb mértékű a javulás, és a módszer továbbra is rosszul skálázható (ld. mért futási idők).

A előállt osztályozás finomítására majority filter lett alkalmazva. A művelet során egy adott pont osztályát alakíthatjuk át olyan módon, hogy a szomszédok osztályait vizsgáljuk. Ha a szomszédok osztályai közt egy adott kategória többségben van, akkor a központi elem osztálya is lecserélődik erre az értékre. Az osztályváltoztatás során a módosított elemeket először külön jelöljük, hogy az adott iteráció során még az eredeti osztályokkal dolgozzunk. A művelet mentén figyeljük, hogy az összes pont hány százalékán történt változtatást, és addig ismételjük míg ez a mennyiség egy adott küszöböt el nem ér.

A terveknek megfelelően a program kimenete egy HD felbontású PNG-kép OpenCV segítségével előállítva. Látni kell, hogy a tesztek során kivárható ideig futó pontfelhő csak éppenhogy elegendő adatot szolgáltat ehhez a felbontáshoz (vö. 1280*720 ~= 1M vs. ~1.8M).

A tesztfuttatásokat a publikusan elérhető (holland) AHN3 pontfelhőkből műholdkép alapján választott és általunk kivágott részein végeztük. Egy területre készítettünk a lépéseket és módszereket egyesével bemutató képanyagot. A cikkben kapott eredményekkel való közvetlen összehasonlítás sajnos nem volt lehetséges, mert nem állt rendelkezésre az ott használt adatok pontos elérhetősége.

citySMALL.las: 1,8 millió pont, 0,3 km^2, 15 perc, 2GB memória