

AI alapú pontfelhő feldolgozás és automatizált BIM modell építés

TDK dolgozat

Pécsi Tudományegyetem

Műszaki és Informatikai kar

Pécs, 2024.04.16.

Készítette: Rácz Viktor Norbert

Témavezető: Dr. Máder Patrik Márk

Tartalom

1. Bevezetés	3
2. Szakirodalmi kutatás	4
2.1 Pontfelhő.....	4
2.1.1 Mi a pontfelhő?.....	4
2.1.2 Modellépítés pontfelhő alapján.....	4
2.1.3 Információtartalom.....	5
2.2 A kutatáshoz használt szoftverek.....	5
2.2.1 Autodesk Revit	5
2.2.2 CloudCompare	6
2.3. Építészet és programozás	6
3. Módszertan	6
3.1. Módszertan bevezetés.....	6
3.1.2 Információtartalom szintjei (LOI, LOD)	8
3.1.3. Kiindulási állományok	8
3.2 Pontfelhő AddOn	8
3.2.1. Pontfelhő műveletek.....	8
3.2.2. Pontfelhő klasszifikáció.....	11
3.2.3 Pontfelhő tisztítási módok	12
3.2.4 Pontfelhő ritkítás.....	13
3.2.5 Sík meghatározás	14
3.2.6 Pontok kinyerése és csoportosítása.....	16
3.2.7. Elemek létrehozása	16
4. Konklúzió	17
5. Távlati célok	18
6. Forrásjegyzék	19
7. Ábrajegyzék.....	20

1. Bevezetés

Az építészeti felmérések és közülük is kifejezetten a pontfelhő alapú megoldások kiemelten fontosak az építőipar számára, mivel manapság többek között a tervezési, minőség-, és mennyiségellenőrzési feladatok kiindulási állományaiként használatosak. Alkalmazhatók meglévő épületek, kivitelezés alatt álló építmények, vagy megvalósult létesítmények esetén is. Előállításuk a korszerű eszközök segítségével egyszerűnek nevezhető, azonban a szoftveres utófeldolgozásuk (pl.: pontfelhők tisztítása, ritkítása, osztályozása) nehézségekbe ütközhet. Annak ellenére, hogy a programok számos funkciót kínálnak, az utófeldolgozás mégis sok, időigényes manuális munkavégzést igényel. Amennyiben ezt követően a pontfelhőből 3D, vagy BIM modell is épül az további manuális munkát igényel. Tapasztalataim alapján azonban ezek a folyamatok is jelentősen felgyorsíthatók és részben, vagy egészben automatizálhatók korszerű módszertanok használatával.

Egy korábbi kutatásom során már vizsgáltam a 2D DWG és PDF állományok feldolgozásának módszertanait. A kutatás során feltártam az automatizált modellezéshez szükséges hiányzó információkat, majd ezek létrehozása után egy általam megírt program segítségével meg tudtam modellezni automatizáltan a helyiségek falburkolatait.

Ezt szem előtt tartva a kutatás-fejlesztési tevékenységem célja az építészeti munkafolyamatok során használt pontfelhő állományok feldolgozási módszereinek vizsgálata és a mesterséges intelligencia, valamint az AI modellek segítségével történő automatizálási lehetőségek bemutatása egy általam megírt program segítségével.

A jelen és megelőző kutatásaim a későbbi doktori disszertációm alapjául fognak szolgálni.

Jelen kutatás a "Kriszbacher Ildikó Tehetséggondozó Program" 2023/24 tanévi, valamint a BIM SKILLS LAB kutatócsoport támogatásával valósulhatott meg.

2. Szakirodalmi kutatás

2.1 Pontfelhő

2.1.1 Mi a pontfelhő?

A bevezetőben már említésre került, hogy az építészeti folyamatok során gyakran használunk pontfelhőket, azonban az nem került tisztázásra, hogy mi is pontosan a pontfelhő. A pontfelhő olyan pontokból álló adathalmaz, amely egy fizikai tárgy (a kutatás tárgyánál maradva, például egy épület, vagy egy épületrészlet) háromdimenziós koordináta-rendszerben való digitális leképezését tartalmazza. (1. ábra) Előállítható lézerszkennerral és fotogrammetriai eljárással is.[1] Minden bennük tárolt pont az adott koordináta-rendszerben X, Y, Z koordinátával jellemezhető, azonban – a formátumtól függően – ettől több adatot is tartalmazhatnak. Ilyen többletadatnak számít például a pont színe (R, G, B), intenzitása (0-1, -2024-+2024) és normál vektor komponensek is, amelyek nem minden esetben tárolódnak el.



1. ábra - Egy épületről készült külső pontfelhő

2.1.2 Modellépítés pontfelhő alapján

Mivel a pontfelhő a valóság digitális leképezése, ezért megfelelő kiindulási alapot képezhet modellépítéshez. A modellek elkészítését el lehet végezni hagyományos módon, manuálisan modellezve az objektumokat és automatizáltan, a digitalizáció eszköztárát kihasználva is. Ez a tanulmány leginkább az automatizált, AI módszertanokra fókuszál, de akár ezt, akár a manuális lehetőséget szükséges alkalmazni, a modellépítés előtt fontos meghatározni a modellezés célját és részletességét. A cél és a részletesség nagymértékben hatással van a modell létrehozásához szükséges időre és erőforrásokra. Például egy tetőszerkezet modelljének elkészítése előtt fontos definiálni, hogy a tető szerkezeti részletei is fontosak-e, vagy kizárólag a tetőfelület a mérvadó. Szerencsés esetben ez már az épület helyszíni felmérése során, vagy már korábban tisztázódik, hiszen abban a munkafolyamatban sem mindegy, hogy milyen módszertani lépések szükségesek. Nagyon sok szempont és döntés határozza meg a folyamatok menetét és lehetőségeit, amelyeket egymásra kell építeni.

Ezt a fokozatosságot a jelenlegi kutatás-fejlesztési projektben is figyelembe kell venni. A módszertanok egyelőre a falak és a födémek egyszerűsített automatizált modellezésére

fókuszálnak, azonban távlati cél a lehető legtöbb épületszerkezet (pl.: nyílászárók, vagy a korábban említett tetőszerkezet) programozott elkészítése, akár magasabb geometriai részletezettségben. Utóbbit azért fontos kiemelni, mivel komoly kihívást tartogat (a manuális modellépítéssel szemben) olyan programozott eszközt készíteni, amely a magasabb geometriai részletességhez szükséges információkat (pl.: ablak szárnyak méreteit és eloszlását) képes automatikusan felismerni és kezelni.

2.1.3 Információtartalom

Egy pontfelhő állományban, a formátumtól függetlenül, általában a pontok térbeli pozíciójáról, színéről, esetenként normál vektoráról van információnk. A programsorok számára egyértelműen és egyszerűen nem lehet rámutatni, valamint azonosítani például építészeti elemeket. Ahhoz, hogy a pontfelhőn belül az egyes ponthalmazok forrását azonosítani lehessen (hogy milyen objektum térbeli alakját rajzolja ki) mélyreható analízisek és vizsgálatok szükségesek. Ez azt jelenti, hogy hiába látható egy emberi szemlélő számára, hogy az adott pontok összessége falat, vagy asztalt ábrázol, a program ezt így nem tudja érzékelni. Részben ezért, hogy a tapasztalásból adódó hiányt át lehessen hidalni alkalmazni lehet úgynevezett "klasszifikálási" eljárásokat. Ezekben lefuttatott komplex vizsgálatok képesek arra, hogy a korábban említett ponthalmazok összefüggéseit értelmezzék és az összetartozókat összerendeljék. Ezzel el lehet érni például, hogy egy falsíkon található pontok, vagy a bútorok és berendezési tárgyak egy-egy csoportba rendeződjenek. A csoportosítás segít a nem szükséges objektumok eltávolításában, a pontfelhőben található "zaj" csökkentésében és a kutatás területét is érintő síkok, felületek meghatározásában. Utóbbi esetén az összerendezett ponthalmazokra síkok feszíthetők, amelyek kirajzolják például a falak és födécek határoló elemeit.

2.2 A kutatáshoz használt szoftverek

A pontfelhő feldolgozáshoz törekedtem az ingyenesen elérhető, nyílt forráskódú szoftverek használatára, így esett a választásom a CloudCompare [2] programra. Ezen felül fő célom volt, hogy a kialakítandó módszertan átültethető legyen más hasonló környezetekre, egyéb pontfelhő klasszifikálásra alkalmas AI megoldásokba is. A modell elkészítéséhez Autodesk Revit [3] szoftvert használtam, melyben már számos egyéb kiegészítőt fejlesztettem és magabiztos BIM modellezői tapasztalatot is szereztem. Ezek során lehetőségem volt mélyen megismerni a szoftver korlátait és lehetőségeit is.

2.2.1 Autodesk Revit

Az Autodesk Revit egy olyan programkörnyezet, amely BIM módszertanok szerinti modelleket építését és a modellekészítés, tervezés során tervezői együttműködést tesz lehetővé. A létrehozott modell elemek módosítása és a modell elemek információval való feltöltése is könnyedén elvégezhető a szoftver funkcionalitását kihasználva. Ezen felül a program API dokumentációja is publikusan elérhető, jól strukturált és átlátható, így támogatja egyedi kiegészítők fejlesztését. Ezen fő indokok miatt választottam a kutatás modell építési fázisának lebonyolításához.

2.2.2 CloudCompare

A CloudCompare egy nyílt forráskódú pontfelhő feldolgozó szoftver, mely képes pontfelhő beolvasására, vágására és részben klasszifikálására is. A nyílt forráskód, valamint előzetes tapasztalataim miatt választottam kutatási eszköznek.

2.3. Építészet és programozás

A digitalizáció iránti igény növekedésével az építészeti tervező szoftverek fejlesztői elkezdtek nagyobb hangsúlyt fektetni a szoftverek API oldali dokumentálására, hozzáférhetőbbé tételére. Ez azért fontos, mert a parametrikus tervezés és modellezés terjedésével felmerült az igény speciális beépülő modulok (plugin/addon) fejlesztésére. Ez mára olyan irányt vett, hogy a mindennapi felhasználó számára is lehetővé vált egyszerű „scriptek”/”algoritmusok” létrehozása, amelyekkel támogathatók a különböző munkafolyamataik.

A hagyományos szöveges programozási lehetőségek mellett a szoftverek tartalmazhatnak vizuális programozási környezeteket is, melyek lehetőséget adnak minimális programozási tudással, vagy akár programozói ismeretek nélkül is egyszerűbb automatizmusok, algoritmusok létrehozására.

Ezek közül az építészek számára is elérhető két legelterjedtebb környezet a Dynamo[4] és a Grasshopper.[5] Előbbi az Autodesk Revit szoftver szerves része, utóbbi pedig több építőiparban használatos szoftverrel is képes együttműködni. Kiemelendő szempont, például a Revit és Dynamo esetén, hogy egy Revit Addon futtatásához nem szükséges egy plusz programot elindítani, így a futási idő alacsonyabb, valamint a felhasználó nem látja az algoritmus mögötti „kódot”. Ezáltal az algoritmus szerzői jogai is védettebbek. Hátránya, hogy nem lehet olyan könnyen átalakítani az algoritmust, ha nem nyújt teljes körű megoldást az adott problémára.

Még kell említeni még a Param-O[6] felületet is, mely a Graphisoft Archicad vizuális programozásra használható új modulja. Ez azonban jelenleg a korábban említettekhez képest korlátozottabb funkcionalitással rendelkezik. Például a Dynamo és a Grasshopper magas szinten támogatja a paraméter alapú tervezési módszertanok (parametrikus tervezés, generatív tervezés) alkalmazását.

Minden szempontot figyelembe véve a kutatás során alkalmazandó környezetet végül sikerült leszűkítenem a Dynamo-ra. Mélyen integrált a Revit programba és közvetlenül hozzáfér a Revit funkcióihoz (Revit API). Natívan létre tud hozni modell elemeket (pl.: falakat, födémekeket) és python programozási nyelven írt saját scriptekkel is bővíthető.

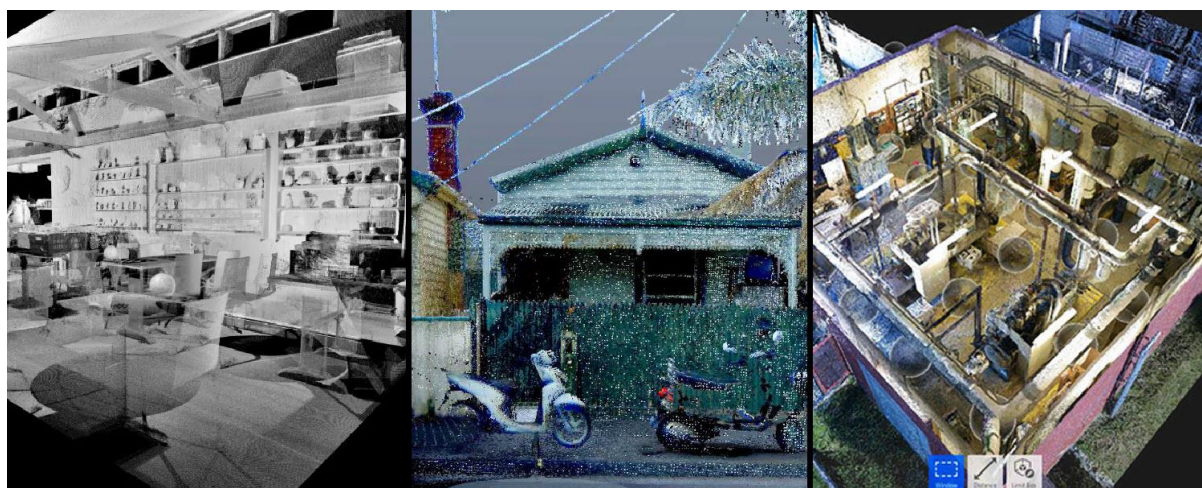
3. Módszertan

3.1. Módszertani alapelvek

Jelen kutatás problémafelvetése, miszerint a pontfelhő alapú 3D modell építés a pontfelhő függvényében idő és erőforrás igényes feladat, mely véleményem szerint különböző programok funkcionalitásának kombinálásával és programkódok segítségével automatizálható, meghatározza a módszertan fő lépéseit. Ezek két fő csoportra oszthatók. A pontfelhő elkészítésére és utófeldolgozására, valamint a modellek elkészítésére.

A kutatás szempontjából három osztályba soroltam a pontfelhőket: Tisztán beltéri, tisztán kültéri és vegyes. A pontfelhők megkülönböztetésének a terekben, felületeken megjelenő különböző geometriák és tárgyak miatt van jelentősége. Például egy épület homlokzatán találhatunk díszeket; párkányokat; gépészeti elemeket (pl.: klíma kültéri egység); valamint egyéb építészeti elemeket, melyek geometriájukkal jelentős problémát jelentenek az automatizált feldolgozás szempontjából. A falak és födémek meghatározásához függőleges, valamint vízszintes felületeket kell keresni a pontfelhőben. Egy épület homlokzat esetén a homlokzaton megjelenő elemek megszakítják a homlokzat folytonosságát, így egy "lyuk" alakul ki a modellben ezekben a pozíciókban. A kutatás jelenlegi fázisában ez szerint működik és nem terjed ki a homlokzati elemek részletes geometriájának vizsgálatára, az elemek azonosítására és modellezésére.

Tisztán beltéri pontfelhő esetén a helyiségekben tárolt bútorok, használati tárgyak jelentenek problémát. A felületek vizsgálatánál az algoritmus megtalálja a bútorok függőleges és vízszintes felületeit is, amelyek figyelembevétele esetén hibás elemek jelennek meg a modellben. A probléma áthidalása érdekében a kutatás jelenlegi fázisában manuális tisztítási módszereket használtam, amelyek során ezeket a felesleges objektum pontokat eltávolítottam. A bútorok mellett további nehézséget jelentenek a beltéri lépcsők is, melyeket szintén kézi módszerrel töröltem a pontfelhőből. Ezekkel a lépésekkel a nehézségek megszüntethetők, csökkenthetők.



2. ábra - Balról jobbra: Beltéri pontfelhő, kültéri pontfelhő, vegyes pontfelhő

Az 2. ábrán láthatók példák az előző bekezdésben említett esetekre. A két szélső képen jól megfigyelhető, hogy belső tér esetén is milyen változatos lehet a pontfelhő. A bal oldali képen egy iroda látható, melyben a polcos szekrényeken észrevehető használati tárgyak geometriájukkal meg tudják zavarni az algoritmus működését, emiatt fontos a klasszifikálás és a tisztítás. A jobb oldali képen egy gépház pontfelhője látható. Ebben az esetben a gépészeti szerelvények jelenléte jelenti a legnagyobb problémát. Például egy, a fal előtt függőlegesen végigfutó szerelvény kitakarhatja részben vagy egészben a mögötte húzódó falat, így a klasszifikáció után a szerelvény helyén egy teljes sávban "lyuk" keletkezik a pontfelhőben. A kialakult hézag miatt a felület létrehozás során két külön felület jön létre ugyan arról a falról (egy a hézag felett, egy a hézag alatt), így ezt utólag a modellben kezelni, javítani szükséges.

A tisztítást követően következhet az algoritmusok alkalmazása. A pontfelhő feldolgozásáért és a modellelemek létrehozásáért két külön algoritmus felel, melyek két külön programnyelven

íródnak. Jelenleg a pontfelhő feldolgozás befejeztével az algoritmus létrehoz a számítógép egy előre meghatározott helyén egy CSV fájlt, melybe beírja a falak létrehozásához szükséges adatokat. A második algoritmus a Revit elemek létrehozásának végeztével törli a létrehozott fájlt.

A kutatás jelenlegi szakaszában a második algoritmus csak a falak lehelyezésére fókuszál. A falak lehelyezésének módszertana a 3.2.6 és 3.2.7 fejezetekben kerül bővebben kifejtésre.

3.1.2 Részletezettség szintjei (LOD)

A BIM modellek esetében a részletezettség szintjét általában úgynevezett LOD (Level of Development) értékkel jelzik (például: LOD 200). A LOD szintekről és a hozzájuk tartozó geometriai elvárásokról, valamint információtartalomról átfogó képet ad a hazai Lechner BIM-kézikönyv is. [7] A kutatás során létrehozandó részleges felmérési modellt LOD 200 szintű részletezettséggel kívánom ellátni. A modell jelenlegi fázisában kizárólag a falakat és esetlegesen a födémeket fogja tartalmazni.

3.1.3. Kiindulási állományok

Kiindulási állományom két pontfelhő. Az egyik egy iroda, mely tisztán beltér, a másik egy középület homlokzati részlete.

A kutatási-fejlesztési tevékenységem során Autodesk Revit 2022 szoftvert a modell elkészítéséhez, Visual Studio Community 2022 szoftvert a programozáshoz, valamint Cloud Compare szoftvert használok a pontfelhő tisztításához. A fejlesztés során két algoritmust hozok létre, az első python programnyelven készül a pontfelhő feldolgozásához az Open3D könyvtár segítségével, a második C# programnyelven készül, mert ezen a nyelven lehet a Revit-hez beépülő modult fejleszteni. Utóbbi felel a Revit modellelemek létrehozásáért.

3.2 Pontfelhő AddOn

A pontfelhő feldolgozó eszköz kiválasztásának fő szempontjai a következők voltak: dokumentáltság mértéke, költsége, valamint, hogy nyílt forráskódú-e. A kiválasztás során törekedtem jól dokumentált, nyílt forráskódú és ingyenes szoftverek összehasonlítására, melyek közül a legmegfelelőbbnek a CloudCompare és az általa használt Open3D Python könyvtár bizonyult. A kiválasztást megelőzően megvizsgáltam a Revitben rendelkezésre álló parancsokat és műveleteket is, azonban ezek nem bizonyultak elegendőnek a célként kitűzött feldolgozottság eléréséhez.

A pontfelhő feldolgozásához a CloudCompare-ben is használt Open3D könyvtárat használtam [8]. A pontfelhőt a beolvasás előtt meg kellett tisztítani a zajtól, továbbá a kiterjedését is csökkenteni kellett a tesztelés megkönnyítése érdekében. A pontfelhő részlet végül PTS formátumban került kimentésre, mert az algoritmus nem tudta feldolgozni a többi formátumot.

A falak létrehozásához magassági értékre, kezdő és végpontra, valamint honszinttől való magasságra van szükség. Mint 3.1-es bekezdésben említettem, az algoritmus síkokat fog meghatározni a pontfelhőből. A sík illesztést végző parancs először csoportosítja a pontokat, majd ezekre illeszti a síkot, melyeket végül listába rendezve visszaad. A létrehozott síkok általában több mint 4 sarokponttal rendelkeznek, amelyekből a 4 szélső sarokpont koordinátáit írja a kommunikációs fájlba. A födémek létrehozásához minden kapott pont koordinátáira szükség lesz.

A Z koordinátákból meghatározható a fal magassága, valamint a modellben előre meghatározott modellszintek pozícióival összevetve meghatározható a fal szintje és szinttől való magassága is. Az X és Y koordinátákból pedig meghatározható a fal térbeli pozíciója. A további műveleteket a modellező algoritmus végzi.

3.2.1. Pontfelhő műveletek

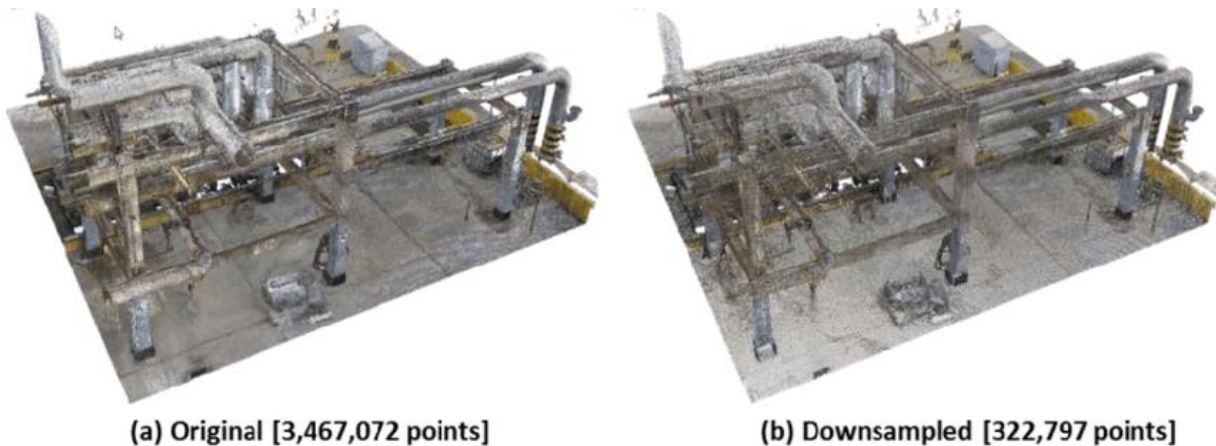
Az Open3D könyvtárhoz képest a Revit natív pontfelhő műveletei jelentősen korlátozottak. A funkcionalitás kimerül a megadott síkokra illeszkedő pontok szűrésében, valamint rész pontfelhők kinyerésében. A feldolgozáshoz matematikai műveletekre és különböző vizsgálatokra is szükség van, így szükséges külső könyvtár használata.

A pontfelhő beolvasásakor teszteltem a beolvasható állományokat az eredmény javítása és információtartalom növelése szempontjából. A tesztelés eredményeként a PTS állomány bizonyult optimálisnak mind fájl méret, mind információtartalom szempontjából. Emellett kiderült, hogy az algoritmus kizárólag ezt a fájlformátumot tudta feldolgozni, amire sajnos nem találtam megoldást. Ezt követően kalkuláltam a normál vektorokat, a felületek könnyebb azonosítása érdekében.

A pontfelhő fájl méretének csökkentése, így jobb kezelhetőség elérésének érdekében megvizsgáltam az elérhető pontfelhő ritkítási parancsokat, majd teszteltem őket a felületek létrehozásának segítése céljából. A legjobb megoldásnak a „Voxel downsampling” bizonyult, mely során az algoritmus térbeli négyzeteket képez a pontfelhőben, majd a négyzetben található pontok koordinátáit, színét, valamint normál vektorát átlagolja és egyetlen pontként adja vissza. A négyzetek mérete állítható.

A 3. ábrán látható egy pontfelhő pontjainak száma ritkítás előtt és után. Az ábra alapján felvetődhet a kérdés, hogy milyen hatása van a pontfelhő információtartalmára, valamint a feldolgozás folyamatára. A ritkításra a klasszifikáció után kerül sor, így véleményem szerint a falakból és födémekből nem vész el lényeges információ, mivel a felületek meghatározásához továbbra is elegendő pont áll rendelkezésre. A ritkítás kedvezően hat a feldolgozhatóságra, mivel az algoritmusnak kevesebb pontot kell átvizsgálnia a felületek meghatározása során. Az (a) és

(b) képek alatti pontszámokból megállapítható, hogy a ritkítás jelentősen csökkentheti a pontfelhő pontjainak számát, ezáltal növelve a feldolgozás sebességét.



3. ábra - Kiindulási (balra) és a ritkított (jobbra) pontfelhők

A következő lépésben meg kellett határozni a felületeket. A parancs különböző paraméterekkel futtatható. Meghatározható a pontok normálvektorainak maximális szögeltérése, a meghatározott síktól való maximális távolság, a távolabbi pontok összes pontra vetített aránya, az illesztett felület legkisebb oldalhossza, a felület illesztésénél részt vevő pontok minimális száma, valamint a vizsgálandó szomszédok száma minden pont esetén. A parancs eredménye egy lista, amely tartalmazza az összes felületet, amelyet a megadott paraméterekkel létrehozott. A visszkapott elemek orientált befoglaló dobozok (oriented bounding box), amelyek sarokpontjai alapján felületek hozhatók létre. A felületek létrehozása azért lényeges lépés, mert a befoglaló dobozokból csak a két átellenes sarokpont nyerhető ki.



4. ábra - Planar patch detection (balra) és Plane segmentation (jobbra) sík létrehozási parancsok eredményei

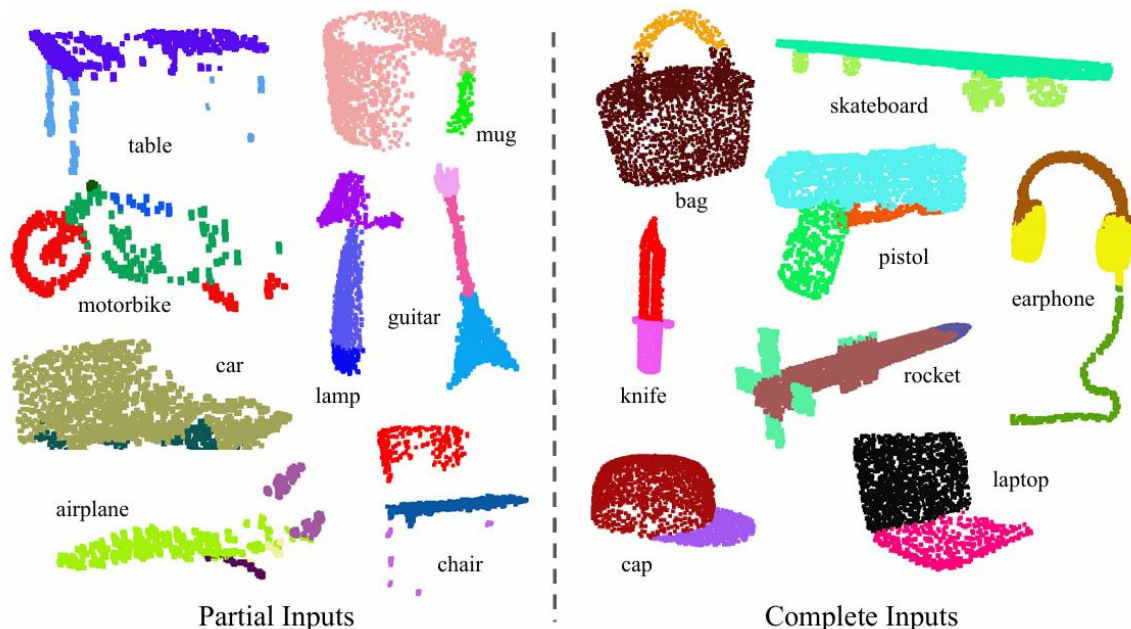
A 4. ábrán a felület létrehozására szolgáló két parancs látható. A „Plane segmentation” parancs egyetlen síkot határoz meg, amely a legjobban illeszkedik a pontfelhőhöz. A „Planar patch detection” parancs hasonló eljárást követ, azonban itt a program normálvektorok alapján rész pontfelhőket hoz létre, amint azt az előző bekezdésben is említettem. Ezekre a rész pontfelhőkre illeszti a felületeket, majd egy összevont felületként, úgynevezett „patch”-ként adja vissza őket.

A felületek létrehozása után megvizsgáltam a program által meghatározott összetartozó felületeket, kizárólag a függőleges felületeket megtartva, majd kinyertem a négy szélső sarokpontot a falak elhelyezéséhez.

3.2.2. Pontfelhő klasszifikáció

Az épületek belső tereiről készült pontfelhők általában tartalmazzák a helyiség bútorait is, amelyek jelenléte akadályozza az automatizált modellezési folyamatot. Az eredmények javítása érdekében a kutatás következő fázisában a pontfelhő klasszifikálása szükséges, amelyhez egy klasszifikáló mesterséges intelligenciára lesz szükség.

A pontfelhő klasszifikáció egy mesterséges intelligencia (AI) alapú folyamat, melynek során az AI megpróbálja azonosítani a memóriájában tárolt elemeket a pontfelhőben. Ez azt jelenti, hogy az AI tanítása során bútorokról és tárgyakról készült, címkézett pontfelhőket adtam neki, amelyek tartalmuk szerint lettek kategorizálva. A tanítás végén az AI önellenőrzés és pontosságának meghatározása céljából lefuttatta a klasszifikációt egy címkézett pontfelhő halmazon, lehetővé téve az AI által adott osztályozás összehasonlítását a valós besorolással. Az AI tanításához a Princeton Egyetem „ModelNet40” [9] pontfelhő készletét használtam. Jelen kutatás során az elemek alatt bútorokat és használati tárgyakat értünk. Számos nyílt forráskódú és ingyenes AI modell elérhető, amelyek építészeti pontfelhők klasszifikálására lettek fejlesztve. [10] E kutatáshoz a PointNet modellt választottam, amely a paperswithcode.com [11] weboldal szerint 89,2%-os pontossággal rendelkezik pontfelhők klasszifikálásában. A modell programozási nyelve Python, de elérhető C++ nyelven is, ami megkönnyítheti a későbbi fázisokban történő implementálását az Archicad szoftverkörnyezetébe.



5. ábra - Pontfelhő klasszifikálás eredménye

Az 5. ábra szemlélteti a klasszifikáció és szegmentálás eredményét részleges pontfelhő (bal oldalon) és teljes pontfelhő (jobb oldalon) esetében. Az ábra alapján megállapítható, hogy a

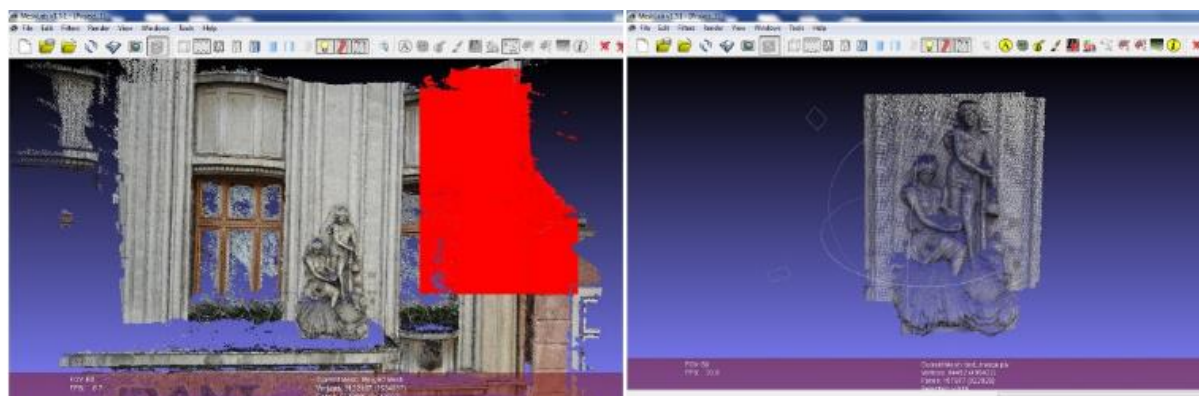
modell képes tárgyak, valamint azok részeinek azonosítására még részleges pontfelhők esetén is.

A modell kiválasztásának további indoka, hogy a pontfelhő szegmentálás funkciója – amely képes egy tárgy részeit megkülönböztetni, például az asztal esetében az asztallábakat és az asztallapot – kiemelkedő pontossággal működik a többi elérhető AI modellhez képest. Ezen funkció különösen fontos a jelen kutatás szempontjából, mivel a kutatás későbbi fázisában, a nyílászárók elhelyezése során, több adat nyerhető ki az ajtók és ablakok részpontfelhő állományaiból. Ilyen többletinformáció lehet például az ajtószárny szélessége, különösen kétszárnyú ajtók esetén, vagy egy ablak osztása, ha nem egy egyszárnyú, felülvilágító nélküli ablakról van szó.

A kutatás jelen fázisában megvizsgáltam a fent említett klasszifikáló AI lehetőségeit, és elkészítettem a saját, tanított AI modelletem. A kutatás későbbi implementációs szakaszában össze kívánom hasonlítani az előre tanított AI és az általam tanított AI eredményeit.

3.2.3 Pontfelhő tisztítási módok

A pontfelhő tisztítása során eltávolítjuk azokat az elemeket, amelyek nem szükségesek a modellezéshez. Például egy épület felmérése esetén ide tartozhatnak a környező épületek és a növényzet; belső felmérés esetén pedig azok a helyiségek, amelyekről nem kell modellt készíteni, vagy az ablakokon keresztül látható környezeti elemek, amelyeket a műszer felmért. Ezeket a modellezés szempontjából szükségtelen részeket manuális vagy automatikus tisztítással távolíthatjuk el. Jelenlegi ismereteim szerint a belső terek tisztítását nem lehet automatizálni, míg a külső terek esetén a növényzet elkülöníthető az épületektől különböző pontfelhő-feldolgozó szoftverek, például a CloudCompare, segítségével.



6. ábra - Pontfelhő tisztítás előtt (bal) - utána (jobb)

A 6. ábrán egy példa látható a pontfelhő manuális tisztításának folyamatáról. A bal oldali kép a kiindulási pontfelhőt mutatja, amely tartalmazza az épület homlokzatának egy részletét, valamint egy szobrot. A jobb oldali kép a tisztított pontfelhőt ábrázolja, ahol csak a szobor maradt meg, jelezve, hogy a felmérés során kizárólag a szobor volt releváns, így az épület többi részére nem volt szükség.

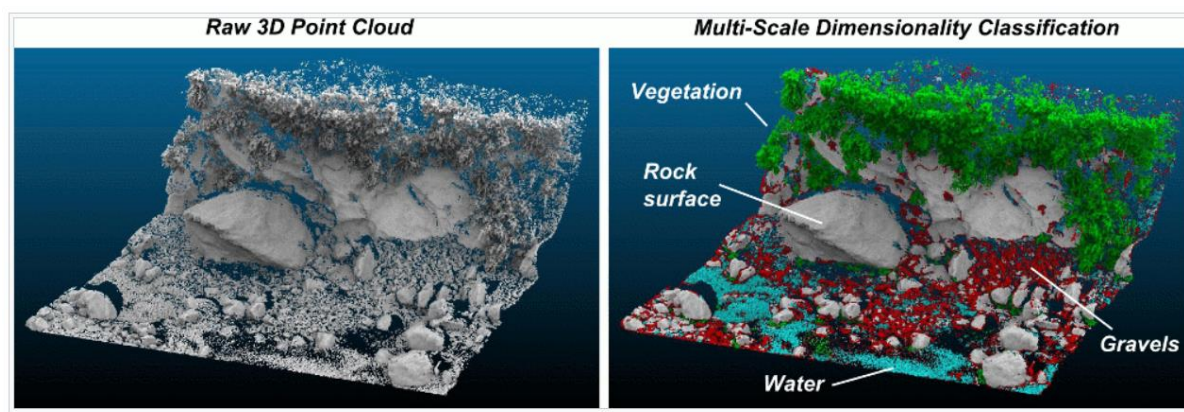
3.2.3.1 Manuális tisztítás

Manuális tisztítás során a felhasználó kézzel jelöli ki a pontfelhő azon részeit, amelyek nem szükségesek a modellezéshez. Ezzel a módszerrel a külső terek viszonylag gyorsan tisztíthatók,

azonban a belső terek esetében különös figyelmet kell fordítani arra, hogy ne kerüljenek kijelölésre és eltávolításra olyan elemek, amelyeket meg kívánunk tartani.

3.2.3.2 Automatizált tisztítás

Az automatizált pontfelhőtisztítás klasszifikációs technikák alkalmazásán alapul. Ebben a folyamatban meghatározzuk, hogy a pontfelhő mely elemei és részei lényegesek számunkra. Például egy nagyobb terület LIDAR szkennelése során, amely épületeket és növényzetet is tartalmaz, a klasszifikálás segítségével elválaszthatjuk az épületeket a növényzettől, mivel csak az épületek pozíciója releváns. A manuális eltávolítás ebben az esetben jelentős munkaigénnyel járna. A CloudCompare klasszifikáló funkciója alkalmas erre a feladatra, mivel képes két különböző osztályt egymástól elválasztani.



7. ábra - CloudCompare többszintű klasszifikálás

A 7. ábrán szemléltetve van a fent említett klasszifikációs funkció működése. A bal oldalon látható az eredeti pontfelhő, amely egy sziklafalat ábrázol. A klasszifikációs folyamat többszöri lefuttatása különböző osztályokkal történt, majd az eredményeket színekkel jelölték, ahogy az a jobb oldalon megfigyelhető. Az ábra azt mutatja, hogy a klasszifikáló modell nagy pontossággal különítette el egymástól a különböző részpontfelhőket.

3.2.4 Pontfelhő ritkítás

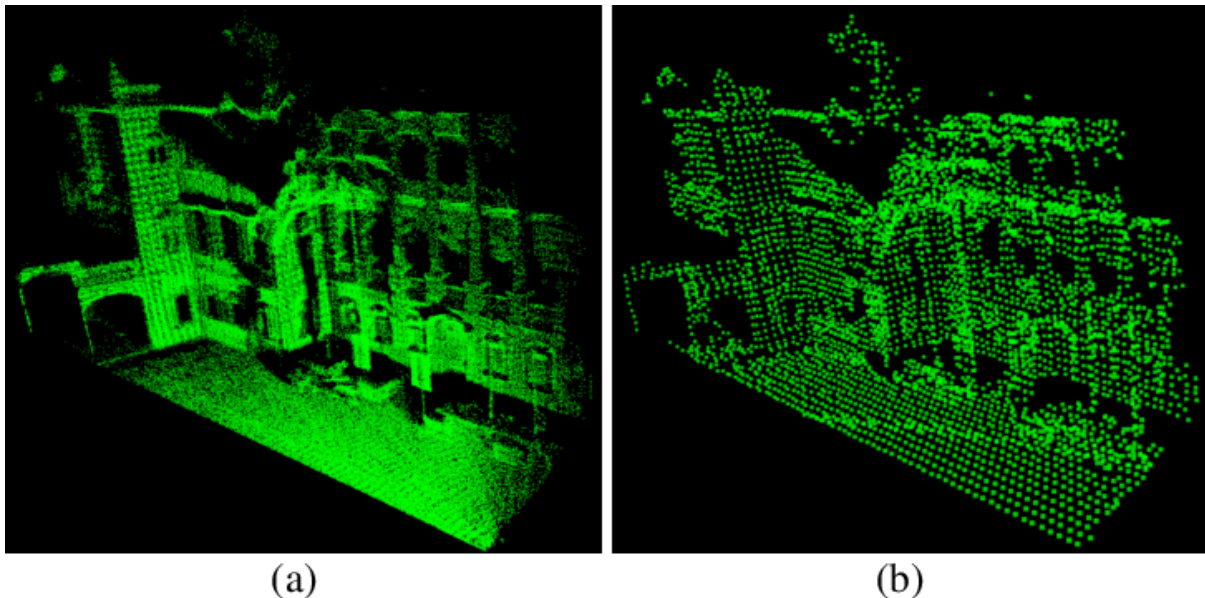
Korábban említettem a pontfelhő tisztításának fontosságát, azonban a ritkítás szintén lényeges lehet. Ritkítás során a pontfelhő pontjainak számát egy meghatározott szabály alapján csökkentjük anélkül, hogy lényeges információt veszítenénk. A ritkítás a tisztítással szemben egy automatizált folyamat.

A kutatás során alkalmazott Open3D [8] könyvtárban többféle ritkítási módszer is elérhető. Ezek a véletlenszerű, az egyenletes, a négyzetes és a legtávolabbi pontok ritkítási technikáit.

- A véletlenszerű ritkítás során, a funkció nevéből adódóan, a pontfelhő pontjainak eltávolítása véletlenszerű módon történik, ami sűrűbb és ritkább területek kialakulásához vezethet. A folyamat során meghatározható az eltávolított pontok aránya a pontfelhő teljes pontszámához viszonyítva.
- Az egyenletes ritkításnál minden k -dik pont kerül eltávolításra a pontfelhőből, a 0. ponttól kezdődően. A funkció futtatása során a k értéke beállítható.

- A négyzetes ritkítás során a funkció először kockákat helyez el a pontfelhőben úgy, hogy ezek csak lapjaikon érintkezzenek. Ezután az egyes kockákban található pontok koordinátáit, normál vektorait és színét átlagolja. Az így kapott új koordinátákat használja az új pontok létrehozásához, melyek rendelkeznek az átlagolt normál vektorral és színnel. A funkció során a kockák mérete is beállítható, ami meghatározza az átlagolás sűrűségét is.
- A legtávolabbi pont ritkítási módszernél a funkció kiválaszt egy kiinduló pontot, majd meghatározza a ponthoz képest legtávolabbi pontot és eltávolítja azt. A folyamat során meghatározható, hogy hány ilyen vizsgálatot szeretnénk végrehajtani. A kezdőpont kiválasztása nem véletlenszerűen történik; a pontokat előzetesen listába rendezzük, és a lista sorrendje határozza meg a kezdőpontok sorrendjét.

A kutatás során a négyzetes ritkítási módszert alkalmaztam, mivel azt vélem, hogy ez a legmegfelelőbb módszer a falak és födécek síkjainak meghatározásához, és az AI implementálásáig, az állomány méretének csökkentésére. A módszer nem jár adatvesztéssel a kutatás szempontjából; bár a falak pozíciója elhanyagolható mértékben változhat.



8. ábra - Négyzetes ritkítás kiinduló állapot (a) és eredmény (b)

A 8. ábra szemlélteti egy eredeti kiindulási pontfelhő (a) és annak négyzetes ritkítással elért eredményének (b) összehasonlítását, amely során jól megfigyelhető, hogy a ritkítás jelentősen csökkentette a pontfelhő pontjainak számát. Ezzel együtt, mint korábban említettem, ez a folyamat befolyásolható. A (b) képen egyértelműen látható, hogy a pontok sűrűsége közel azonos minden területen, ami megkönnyíti a falak és födécek síkjainak meghatározását.

3.2.5 Sík meghatározás

A síkok meghatározása előtt szükség volt a pontok csoportosítására, amelyet „patch”-ek létrehozásával oldottam meg az Open3D „detect_planar_patches” parancsával. A parancs statisztikai megközelítéssel vizsgálja a pontfelhőt, először kisebb egységekre bontva azt. Ezután minden egyes egységre próbál egy síkot illeszteni. A sík meghatározásához az egység medián

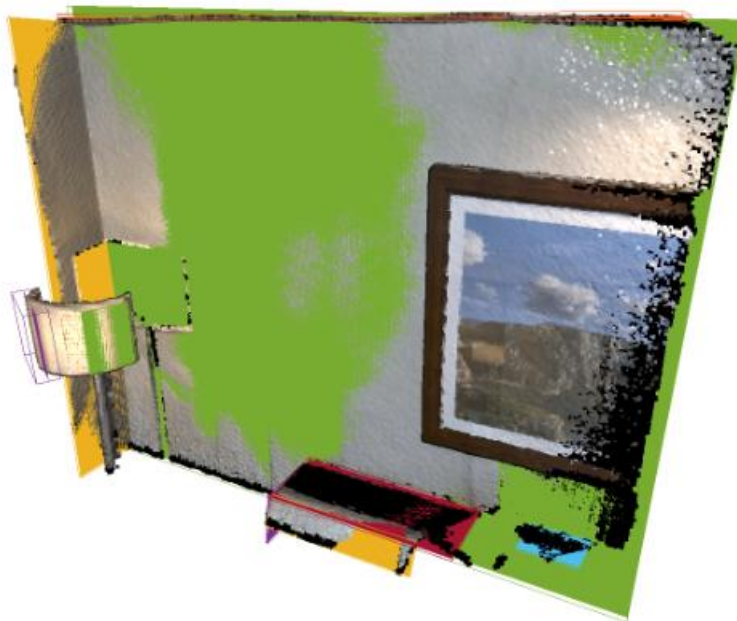
pontjának koordinátáit és normál vektorát használja, majd a „ $ax + by + cz + d = 0$ ” egyenlettel határozza meg a síkot. Ezt követően robosztus síkvizsgálatot végez két fő lépésben.

Az első részben a létrehozott sík és a pontok normálvektorainak egymáshoz viszonyított szögét vizsgálja. Ha a kapott szögek szórása meghaladja a meghatározott küszöbértéket, a rendszer elveti a síkot és újraindítja a folyamatot. A második részben a pontok távolságát vizsgálja a kapott síktól. Ha a távolságok szórása túl nagy, szintén elveti a síkot.

A vizsgálat után jóváhagyott síkokat megpróbálja összeolvasztani, hogy nagyobb, egybefüggő síkok jöjjenek létre, amelyek akár egy külső burkot alkothatnak. A folyamat végén a parancs orientált határoló dobozokat ad vissza, melyekből létrehozhatóak az összefüggő síkok.

Ezt követően az Open3D dokumentációja azt javasolja, hogy a kapott határoló dobozokat a normál vektorok mentén kicsinyítsük le, így megkapjuk a meghatározott síkot. Azonban ez a módszer nem egy síkot eredményez, hanem inkább egy nagyon vékony téglatestet. A kutatás során más megközelítést alkalmaztam.

Miután megkaptam a beforgatott határoló dobozokat, kinyertem belőlük azon pontok indexeit, melyek a dobozon belül találhatóak. Ezt követően a ponthalmazon újra lefuttattam a sík illesztését, de most a „Planar patch detection” parancs segítségével, amely hasonlóan működik, mint korábban bemutatott síkfeszítés. Az illesztés eredményeképpen visszakaptam a sík által meghatározott minimális és maximális pontokat, valamint ezek normál vektorait.



9. ábra - Meghatározott síkok összessége és a pontfelhő

A 9. ábrán a folyamat eredménye látható. A síkok színezéséből megfigyelhető, hogy a parancs egyetlen futtatással több síkot is képes meghatározni. Az ábrán látható síkok automatikusan kerültek színezésre, ezért különböző irányokba álló síkok is azonos színt kaphatnak. A zöld sík és a pontfelhő átfedése alapján megállapítható, hogy a kapott sík előtti és mögötti pontok hasonló arányban helyezkednek el, ami azt jelzi, hogy a sík meghatározása sikeres volt.

3.2.6 Pontok kinyerése és csoportosítása

Az előző folyamat eredményeként kinyert normálvektorokat, valamint a minimális és maximális pontokat végül egy előre meghatározott formátumú TXT fájlba mentettem el.

```
23.611915,-51.38195299999999,4.5601695;24.00571952380952,-51.37761190476191,5.205961428571428
23.346808148148153,-48.405197450980396,-2.8901033333333332;23.603144909090915,-48.3236388,0.3100399999999999
22.898476,-48.39348,2.7740266666666664;23.497329999999998,-48.216698,3.89486
22.233866666666668,-51.76499999999999,-3.003185;22.42077,-51.173153333333325,0.270195
21.855775,-51.66563,-0.6419324999999999;22.12476,-50.94308,0.009542222222222222
23.05009192307692,-51.34577363636363,1.7554833333333333;23.956063454545454,-51.237746666666667,1.9043599999999998
22.059420000000003,-51.29866105263158,-3.033115652173913;22.82793,-51.080479999999994,0.49636
22.771817619047617,-51.313895769230754,0.5795264;24.701953333333336,-48.465355555555555,0.618318
20.16771,-51.5938175,-0.018218709677419342;23.005130555555557,-47.735965,-0.00364
21.80578,-51.42665,3.7726599999999997;24.69349,-48.117113333333336,3.8318600000000003
19.91255,-51.801413000000004,-3.0574257142857135;23.943831999999997,-47.169213333333334,-2.981846666666667
22.65529,-51.946907499999995,-3.0395141666666667;22.952820444444445,-50.9641815151515,4.224026666666666
23.401105428571423,-49.363904,-3.01564;23.767574444444445,-48.223275,0.7598599999999999
21.967437500000003,-48.1250035,-2.9848282352941173;23.32674,-47.7160575,0.24731999999999998
22.424526500000002,-51.26407795918368,0.40029181818181814;23.603014545454545,-48.312020800000006,0.4301494736842106
24.097347499999998,-51.40531105263158,0.5214292391304347;24.795783333333333,-49.058753333333335,5.20489
```

10. ábra - A kimentett pontok formátuma

A 10. ábra szemlélteti a TXT állomány formátumát. Az ábrán látható, hogy a pontok és koordináták elkülönítésére a „,” és „;” karaktereket alkalmaztam. Jelenleg a TXT állomány csak a minimális és maximális pontokat tartalmazza. A födécek létrehozásához azonban minden sarokpontra szükség lesz. A kutatás jelen fázisában a fókusz a falak létrehozásán volt, ezért az állomány nem tartalmazza a födécek pontjait.

A létrejött TXT fájl a falakat létrehozó algoritmus olvassa be. Az algoritmus először létrehozza a pontokat, majd a magassági koordináták különbsége alapján meghatározza, hogy az elem fal vagy födém lesz. Jelenleg az algoritmus csak falak létrehozására képes, de már elő van készítve a födécek létrehozására is. Ha falat kell létrehoznia, a pontok alapján meghatározza a fal nyomvonalát, magasságát és függőleges pozícióját. Ezt követően létrehozza a falat, módosítja a szükséges beállításokat, és végül menti a változtatásokat, ha az összes fal elkészült.

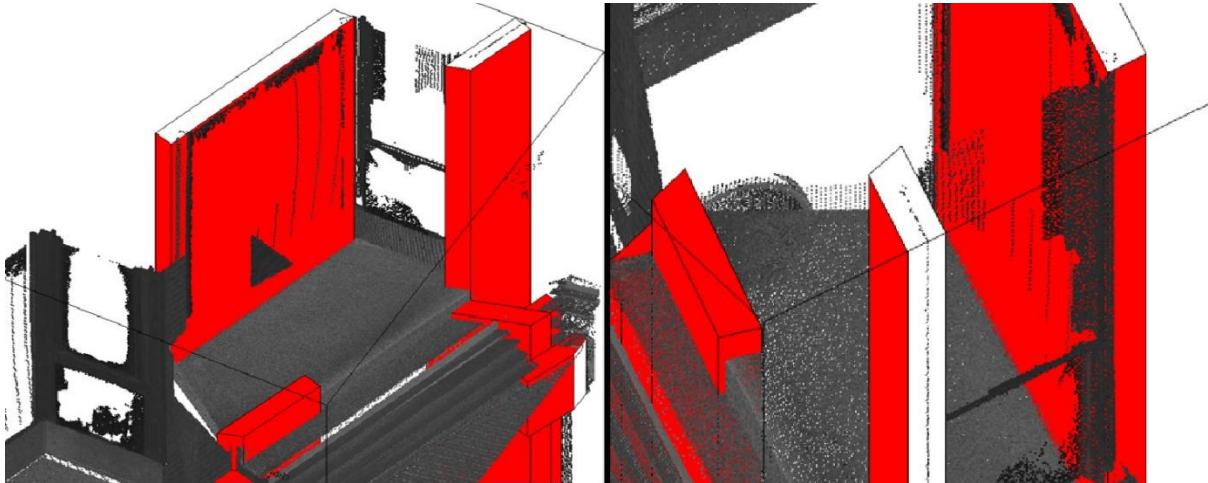
3.2.7. Elemek létrehozása

A Revit API több parancsot kínál a fal modellelemek létrehozásához. A pontfelhőből kinyerhető a fal függőleges pozíciója és magassága, ezért olyan parancsot választottam, amely lehetővé teszi ezen adatok megadását. A fal létrehozásakor megadható a fal típusa is, amelyhez jelenleg az alapértelmezett típust használtam.

A fal létrehozása után a magassági pozíció és magasság paramétereket a pontok alapján kell módosítani. Erre azért van szükség, mert a használt parancs alapértelmezett magassággal (4000 mm) és magassági pozícióban (az elem szintjétől mért 0 magasság) helyezi el az elemet. Az algoritmus ezen pontján azonban már rendelkezésünkre állnak ezek az adatok. Az említett értékek az elem két meghatározott paraméterében („Base Offset” és „Unconnected Height”) vannak tárolva, így csak ezen paraméterek módosítása szükséges.

4. Konklúzió

A kutatás jelenlegi szakaszában az algoritmus már képes azonosítani a falakat a függőleges felületek alapján. Jelenleg a fal tengelyét helyezi a meghatározott felületre, a falak felső és alsó síkjai, valamint kezdő és végpontjai pontosan kerültek meghatározásra.



11. ábra - A kiindulási, szerkesztetlen pontfelhő és a lehelyezett fal modellelemek

A 11. ábrán piros színnel vannak kiemelve a létrehozott fal modellelemek, továbbá a jobb oldali nagyított képen látható, hogy a létrehozott fal tengelyében helyezkedik el a meghatározott sík.

Az előző fejezetben már kifejtettem, hogy a kutatás jelen fázisában a falak elhelyezésére összpontosítottam. Az elért eredmények alapján úgy vélem, hogy ezen szakasz eredményei sikeresnek tekinthetők.

Jelen megoldásnak még vannak hátrányai, azonban jelen állapotában is beilleszthető az irodai gyakorlatba és gyorsíthatja a munkafolyamatokat.

5. Távlati célok

A kutatás jelenlegi fázisában a fejlesztett beépülő modul futtatásához a számítógépre telepített Python programnyelvre is szükség van. Később, amennyiben lehetséges, célom függetleníteni a létrehozott algoritmust.

Mint ahogy a 3. fejezetben már említésre került, a kutatás jelen fázisa a falak létrehozására fókuszált, a födécek létrehozását még nem tartalmazza, azonban már megvizsgáltam a födém sarokpontok kinyerésének lehetőségét, valamint a födécek létrehozásának lehetőségeit is. A kutatás következő fázisa ezen födém elemek létrehozásáról, valamint a falak lehelyezésének továbbfejlesztéséről fog szólni.

A födécek létrehozása utáni fázisban a pontfelhő klasszifikáló AI létrehozása és implementálása lesz fókuszban. Már megvizsgáltam a különböző AI modellek által kínált lehetőségeket, mint azt a 2. fejezetben is említettem. Úgy gondolom az AI implementálása nagy mértékben fogja növelni az algoritmus hatékonyságát, valamint jelentős mérföldkönek fog számítani a kutatás tekintetében.

Ez eddigieken felül a pontfelhő tisztításhoz használható számos Open3D könyvtár további részletes vizsgálata és ezek elérhető funkcióinak tesztelése, implementálása is előttem álló feladat.

Jelen kutatás, valamint a megelőző 2D DWG és PDF állományok automatizált feldolgozását célzó kutatás eredményei is a doktori disszertációm megalapozására fognak szolgálni.

6. Forrásjegyzék

- [1] „Point cloud”, *Wikipedia*. 2024. március 11. Elérés: 2024. március 12. [Online]. Elérhető: https://en.wikipedia.org/w/index.php?title=Point_cloud&oldid=1213173432
- [2] „CloudCompare – Open Source project”. Elérés: 2024. április 12. [Online]. Elérhető: <https://www.danielgm.net/cc/>
- [3] „Autodesk Revit | Autodesk Revit 2024 hivatalos verzió árszabása és megvásárlása”. Elérés: 2024. április 12. [Online]. Elérhető: <https://www.autodesk.hu/products/revit/overview>
- [4] „About | English | Dynamo”. Elérés: 2024. június 14. [Online]. Elérhető: <https://primer2.dynamobim.org>
- [5] R. M. & Associates, „Learn to use Rhino”, www.rhino3d.com. Elérés: 2024. június 14. [Online]. Elérhető: <https://www.rhino3d.com/learn/>
- [6] „PARAM-O”, Graphisoft. Elérés: 2024. június 14. [Online]. Elérhető: <https://graphisoft.com/downloads/param-o>
- [7] Dr. Zagorác Márk és Szabó Beatrix, *BIM-kézikönyv – BEVEZETÉS AZ ÉPÜLETINFORMÁCIÓS MODELLEZÉSBE*, 2. kiad., köt. 1. 1111 Budapest, Budafoki út 59.: Lechner Nonprofit Kft., 2019. [Online]. Elérhető: http://bim.lechnerkozpont.hu/sites/default/files/2019-10/lechner-tudaskozpont-bim-kezikonyv-1-kotet-2-kiadas_3.pdf
- [8] „Open3D: A Modern Library for 3D Data Processing — Open3D 0.17.0 documentation”. Elérés: 2023. november 25. [Online]. Elérhető: <http://www.open3d.org/docs/release/>
- [9] Zhirong Wu és mtsai., „3D ShapeNets: A deep representation for volumetric shapes”, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA: IEEE, jún. 2015, o. 1912–1920. doi: 10.1109/CVPR.2015.7298801.
- [10] „Papers with Code – 3D Point Cloud Classification”. Elérés: 2023. november 25. [Online]. Elérhető: <https://paperswithcode.com/task/3d-point-cloud-classification>
- [11] „Papers with Code – PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. Elérés: 2024. április 13. [Online]. Elérhető: <https://paperswithcode.com/paper/pointnet-deep-learning-on-point-sets-for-3d>

7. Ábrajegyzék

1. ábra - Egy épületről készült külső pontfelhő	4
2. ábra - Balról jobbra: Beltéri pontfelhő, kültéri pontfelhő, vegyes pontfelhő	7
3. ábra - Kiindulási (balra) és a ritkított (jobbra) pontfelhők	Hiba! A könyvjelző nem létezik.
4. ábra - Planar patch detection (balra) és Plane segmentation (jobbra) sík létrehozási parancsok eredményei	10
5. ábra - Pontfelhő klasszifikálás eredménye	11
6. ábra - Pontfelhő tisztítás előtt (bal) - utána (jobb)	12
7. ábra - CloudCompare többszintű klasszifikálás	13
8. ábra - Négyzetes ritkítás kiinduló állapot (a) és eredmény (b)	14
9. ábra - Meghatározott síkok összessége és a pontfelhő	15
10. ábra - A kimentett pontok formátuma	16
11. ábra - A kiindulási, szerkesztetlen pontfelhő és a lehelyezett fal modellelemek	17

1. ábra – Forrás: https://pure-surveying.com/2020/09/22/the-real-value-of-point-clouds-from-images/	
2. ábra – Saját ábra	
3. ábra Forrás: https://www.researchgate.net/figure/Downsampling-point-cloud-data_fig2_315483930	
4. ábra – Saját ábra	
5. ábra - Forrás: https://arxiv.org/pdf/1612.00593v2.pdf	13
6. ábra - Forrás: https://www.researchgate.net/profile/Silviu-Butnariu/publication/259442166/figure/fig3/AS:297021694070786@1447826898555/Clean-up-the-point-cloud-a-Sub-sampled-point-cloud-b.png	14
7. ábra Forrás: https://www.cloudcompare.org/doc/wiki/index.php/CANUPO_(plugin)	
8. ábra Forrás: https://www.researchgate.net/figure/Point-Cloud-Downsample-a-the-point-cloud-before-downsampling-b-the-point-cloud_fig2_349520013	
9. ábra – Forrás: https://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html	
10. ábra – Saját ábra	
11. ábra – Saját ábra	