Budapesti Műszaki és Gazdaságtudományi Egyetem
Építőmérnöki Kar
Általános- és Felsőgeodézia Tanszék

Közel fotogrammetriával előállított objektum modellek pontosság vizsgálata

Konzulens:
Dr. Siki Zoltán

Készítette:
Janurik Zalán

Budapest, 2023

Tartalom

1. Bevezetés

A mérnöki gyakorlatban gyakran szükség van objektumok részletes geometriai felmérésére, és habár ez hagyományos geodéziai módszerekkel is elvégezhető, sokszor a kívánt részletességben nem gazdaságos. Az mérési technológiák és az automatizálás fejlődésével, valamint a számítógépek kapacitásának növekedésével elterjedtek a közel egyidejű, tömeges adatnyerést támogató eljárások, mint például a földi lézerszkennelés, drón- és földi fotogrammetria is. Mivel a lézerszkennelés még most is viszonylag költséges technológia, nem kifizetődő egyszeri projektek, kisebb felmérésekhez beruházni rá, így ismét előtérbe kerülnek a fotogrammetriai eljárások. Ezt támogatják az egyre hatékonyabb gépi algoritmusok, és nagyobb felbontású kamerák, amelyek lehetővé teszik, hogy bizonyos keretek között már akár mobiltelefonos felvételekből is elő tudunk állítani pontos geometriai adatokat. Azonban, ha geodéziában szeretnénk ezeket az eljárásokat alkalmazni, fontos vizsgálni a módszerek pontosságát és megbízhatóságát, valamint optimális mérési módszereket kidolgozni.

Dolgozatomban röviden bemutatom a közel fotogrammetria módszereit, és a BME K épületet a könyvtárral összekötő Sóhajok hídja déli oldalának modellezésén keresztül vizsgálom a különböző módszerekkel és paraméterekkel készített és feldolgozott amatőr eszközökkel készült fényképekből levezethető pontfelhők pontosságát. Céljaim között szerepel, hogy költséghatékony megoldásokat használjak, melyeket egy kisebb geodéziai vállalkozás is gazdaságilag megengedhet magának, emellett a lehetőségekhez képest teljesen automatizált feldolgozás kialakítására törekszem. A vizsgálat során nagy felbontású fényképeket készítek, melyeket olyan nyílt forráskódú szoftverekben dolgozok fel, mint a Meshroom, és a Web Open Drone Map. Az előállított pontfelhőket a CloudCompare szoftverben vizsgálom, és levezetem az ellenőrző pontokban jellemző pontosságokat. Vizsgálom továbbá, hogy a különböző terepi felbontások, valamint ezek kombinációi hogyan befolyásolják az eredményeket. A dolgozat végén igyekszem javaslatot tenni, hogy a felhasznált kamerák és szoftverek esetén melyek az optimális paraméterek a legnagyobb pontosság eléréséhez.

2. Közelfotogrammetria rövid bemutatása

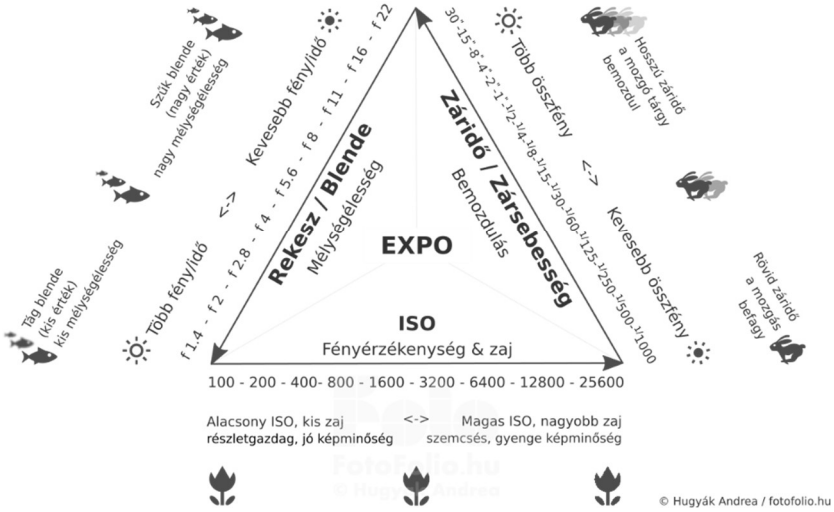
A napjainkban használatos digitális közelfotogrammetria a XIX. század közepén feltalált fotogrammetriai gyakorlatból nőtte ki magát, melyet akkoriban még analóg módszerekkel végeztek. Később a számítástechnika, és a számítógépek kapacitásának rohamos növekedésével azonban megnőtt az igény a feldolgozás minél nagyobb automatizálására. Megfelelő minőségű és mennyiségű fénykép segítségével már képesek vagyunk a (szintén nem régiben elterjedt) lézerszkenneléshez hasonló sűrűségű – bár általában kisebb pontosságú – 3D pontfelhők előállítására. A következőkben a jelenleg elterjedt fotogrammetriai technológia lépéseit szeretném bemutatni.

2.1 Fényképezés

Mivel a végtermék minősége nagyban függ a képi alapoktól fontos, hogy jó minőségű fényképekből induljunk ki. Ha a fényképek műszaki tekintetben nem megfelelőek, a belőlük előállított pontfelhő sem lesz használható, ezért a képek készítésénél oda kell figyelnünk bizonyos szabályokra.

2.1.1 Kameraválasztás és beállítás

Az előállítható pontfelhő felbontását és minőségét nagyban befolyásolja a készített fényképek terepi felbontása (angol rövidítéssel GSD) vagyis, hogy a képen 1 pixel mekkora valós területet fed le, ezért a terepi felbontást az elvárt pontossággal összhangba kell hozni (általában a GSD két-háromszorosára becsülhető az előállított pontfelhő pontossága). Emellett fontos, hogy a fényképeket ugyanazon eszközzel, ugyanazon beállítások mellett készítsük. Ne változtassuk a kamera fókusztávolságát, fényérzékenységét, a rekesznyílást és a záridőt, hiszen ezek változtatásával egy kvázi új kamerát hozunk létre, mely nehezíti a későbbi feldolgozást. Az optimális beállítások, és azok tartása azért is fontos, hogy a készített felvételek élesek legyenek és a lehető legkevesebb zajt tartalmazzák. A kamera e paraméterei összefüggnek, és együtt határozzák meg a készített képek minőségét. A fényképészetben ezek kapcsolatát és a fényképre gyakorolt hatását az úgynevezett „expozíciós háromszög” segítségével szokták szemléltetni (1. ábra).



1. ábra Expozíciós háromszög (https://fotofolio.hu/fotozas-alapjai-expozicios-osszefuggesek)

Fotogrammetriai szempontból a két lényegesebb paraméter a rekesznyílás, és a fényérzékenység. Nagyobb rekesznyílással több fényt engedünk a kamerába, és ezáltal világosabb képet kapunk, mely előnyös lehet gyengébb fényviszonyok között, viszont ezzel együtt romlik a mélységélesség is, vagyis csak egy bizonyos távolságban lévő részletek lesznek élesek, a kamerához közelebbiek, vagy távolabbiak pedig homályosak. Fotogrammetriai célra készült fényképek esetén fontos a mélység élesség, így célszerű kis rekesznyílással dolgozni. A fényérzékenységet az ISO számmal szokás jelölni, mely a fényképezőgép szenzorának érzékenységét mutatja. Minél nagyobb az ISO szám, a szenzor annál érzékenyebb, így kevesebb fény is elegendő egy fénykép elkészítéséhez, azonban ezzel együtt az eredmény rosszabb minőségű, zajosabb lesz. Ha a fényviszonyok megengedik, érdemes alacsony ISO beállítással dolgozni, hogy a lehető legkevesebb zaj terhelje a képeinket. A harmadik paraméter a záridő, mely főként akkor számít, amikor mozgó objektumot fényképezünk. Mivel fotogrammetriával jellemzően álló tárgyakat mérünk fel, ez a beállítás nem annyira releváns, ugyanakkor egy rövid záridő segíthet a kézből készített képeken a kezünk remegésből adódó elmosódás csökkentésében.

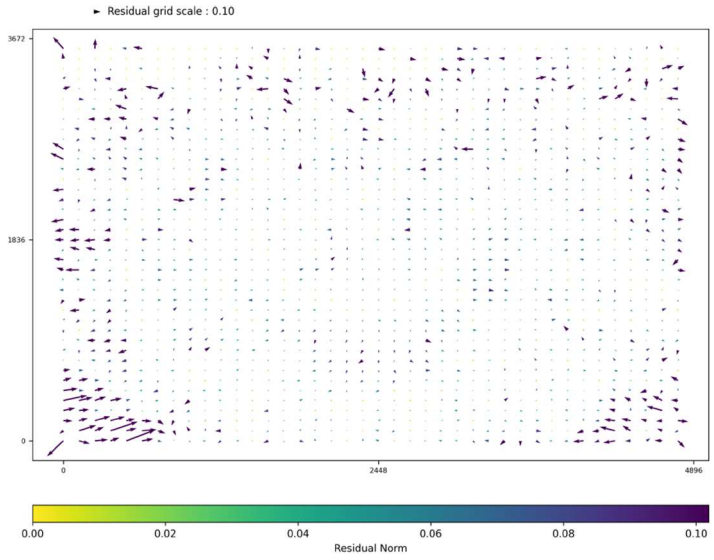
2.1.2 Kalibráció

A pontos geometriai kiértékeléshez lyukkamerával készült felvételekre lenne szükség, azonban a hagyományos geodéziai műszerekhez hasonlóan, a fényképezőgépek sem mentesek a geometriai tökéletlenségekből eredő szabályos hibáktól, ennek fő oka a használt lencsék torzításai. E szabályos hibák hatásait lehet csökkenteni kalibrációval, mely során egy választott torzítási modellben számszerű paraméterekkel írjuk le a torzítás jellegét és mértékét. A

paraméterek ismeretében a torzítás okozta elrajzolás kiküszöbölhető (2. ábra). A leggyakrabban használt modell a Brown modell, mely radiális, és tangenciális komponensekre bontja az elrajzolást, és jellemzően 5 paraméterrel (k_1, k_2, k_3 → radiális; p_1, p_2 → tangenciális) írja le (1. egyenlet).

$$x_{torzított} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + \left(2p_1 xy + p_2(r^2 + 2x^2)\right)$$
$$y_{torzított} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + (p_1(r^2 + 2x^2) + 2p_2 xy))$$

1. egyenlet Brown-féle 5 paraméteres elrajzolás számítás



2. ábra Elrajzolási diagram (WebODM feldolgozásból)

Fontos megjegyezni, hogy a kalibráció csak a kamera egy bizonyos beállítására érvényes, ha például változtatunk a fókusztávolságon, megváltozik az elrajzolás is. Az elrajzolási paraméterek ismeretében feldolgozás előtt előállíthatók a torzulás-mentes képek.

A szabatos kamera kalibráció mellett a modern feldolgozó szoftverekben már van lehetőség a kalibrációs paraméterek meghatározására a feldolgozás közben is. Ez lehetővé teszi előzőleg nem kalibrált amatőr eszközök használatát is, mint például a mobiltelefonokba épített kamerák. Azonban ez nem feltétlen vált ki egy teljes értékű kalibrációt, minden egyes futtatás után is lehetnek különbségek az így meghatározott elrajzolási paraméterek között.

2.1.3 Fényviszonyok

A megfelelő kamera, és kamerabeállítások mellett fontosak még a megfelelő fényviszonyok. Az előzőekben tárgyalt beállítások mellett látható, hogy fontos a jó megvilágítás. Sötét környezetben készített fényképeknél kénytelenek lennénk az ISO és/vagy a

rekesznyílás növelésére, mely rontaná a képek mélységélességét illetve növelné a zajosságát. Amellett, hogy jó megvilágítás mellett érdemes a képeket készíteni fontos az is, hogy ne legyenek a képen túl nagy különbségek a megvilágítás mértékében. Habár az emberi szem képes igen nagy dinamikatartományú látvány befogadására, a használatos kamerák ilyen tekintetben jóval korlátozottabbak. Ha egy képen egyszerre vannak nagyon sötét és nagyon világos részek, a világos részek „kiéghetnek”, a sötétek pedig kivehetetlenül sötétek lesznek. A dinamikatartomány korlátait figyelembe véve, szórt fényben érdemes fényképezni, ideális szabadtéren az enyhén borult idő, hiszen ilyenkor nincsenek éles árnyékok, és a világos felületek sem verik vissza túlságosan a napsütést.

3. ábra Kedvezőtlen és ideális fényviszonyok

A fenti képeken jól látszik, hogy a fényes déli napsütésben készült felvételen a fényképezett objektum jelentős része teljesen „kiégett”, a részletek kivehetetlenek rajta, míg a délután készült – már beárnyékolt képen – az egész átjáró jól látható.

2.1.4 Elrendezés

A fényképek készítésénél oda kell figyelni, hogy a képek között legyen megfelelő átfedés. Az ajánlott képek közötti átfedés drón fotogrammetriában 75-80%. Közel fotogrammetriában konvergens hálózattal akár 90% feletti átfedés is elérhető. E nélkül a feldolgozás során a képek relatív tájékozása problémákba ütközhet, hiszen a feldolgozó szoftverek sokszor csak olyan pontokat adnak hozzá az eredmény pontfelhőkhöz, amelyek 3-4 vagy több képen is azonosíthatók.

2.2 Feldolgozás

Ugyan a feldolgozás konkrét lépései az alkalmazott program függvényében változhatnak, a főbb elvi lépések a legtöbb jelenleg elterjedt szoftverben megegyeznek, elsősorban a lépések felbontásában, bizonyos támogatott algoritmusokban és az állítható paraméterekben térnek el.

2.2.1 Metaadatok

A megadott fényképekből elsősorban kiolvassák az eltárolt metaadatokat, melyek a készítéshez használt kamerára vonatkoznak. Ilyenek lehetnek a kamera típusa, az alkalmazott ISO, rekesznyílás és záridő, a kamera fókusztávolsága. Ha az adatok hiányoznak, vagy hibásak, általában kézzel is megadhatók/javíthatók, illetve egyes szoftverek a kamera szenzor típusának ismeretében egy saját adatbázisból is ki tudják olvasni a szükséges paramétereket. A fotogrammetriai feldolgozás szempontjából elsősorban a fókusztávolság, és a szenzor méretei fontosak, hiszen ezekre szükség van a centrális vetítés elvein alapuló számításokhoz. A feldolgozást gyorsíthatja, ha a felvétel készítés közelítő GPS pozícióját is tartalmazzák a metaadatok, hiszen ezek kiindulópontot biztosíthatnak a külső tájékozásokhoz és a kulcspontok párosításához.

2.2.2 Illesztőpontok

Néhány szoftverben lehetőség van még a feldolgozás elején a georeferáláshoz használatos illesztőpontok megadására azok képkoordinátáival az egyes felvételeken, és valós terepi 3D koordinátáikkal a cél koordináta-rendszerben. Más szoftverek csak a már előállított pontfelhőben kijelölt pontok alapján tudják végrehajtani a transzformációt.

2.2.3 Relatív tájékozás

A fotogrammetriai feldolgozás első jelentős lépése a fényképek relatív tájékozása. Erre a legjobban elterjedt az úgynevezett „Structure from Motion” (SfM) eljárás. Az eljárást eredetileg gépi látás céljára találták ki és feltételezi, hogy egy mozgó kamerával készítenek képeket egy mozdulatlan tárgyról, így jól alkalmazható fotogrammetriában is. Napjainkban elterjedt változata az úgynevezett „Incremental SfM” (Schönberger & Frahm, 2016).

Az SfM kezdetén, az egyes képeken a programok megkeresik a jól azonosítható pontokat, őket nevezzük kulcspontoknak. Ezek a pontok jól elkülöníthetők az egyes képeken,

olyanok, mint sarkok, éles színváltozások, vagy más nagy kontrasztú pontok. A kulcspontok azonosítására használt egyik algoritmus a „scale-invariant feature transform” (SIFT). A SIFT és a hozzá hasonló eljárások a képeket kisebb (állítható számú pixel) részekre bontva vizsgálják. A SIFT előnye hasonló eljárásokkal szemben, hogy nem érzékeny a mintázatok méretváltozására és elfordulására, viszont így nem is a leggyorsabb megoldás.

Az azonosított kulcspontokat képkoordinátáikkal együtt tárolják. Azokat a kulcspontokat, amelyeket nem lehetett elegendő képen azonosítani a programok kiszűrik, és a maradék a képek között megegyező kulcspontok alapján egy kiegyenlítés után előáll a képek relatív tájékozása. Ennek az eljárásnak a „melléktermeke” az úgynevezett ritka pontfelhő, amely a 3D-ben rekonstruált kulcspontokat tartalmazza, emellett szintén előállhatnak a kamera elrajzolási paraméterei.

2.2.4 Sűrű pontfelhő rekonstrukció

A képek relatív tájékozása után következik a sűrű pontfelhő előállítása. A feldolgozó szoftverek például „multi-view stereo” (MVS) megoldással vezetik le a pontok térbeli helyzetét. Ehhez iteratív módon minden képből egy mélység térképet számítanak, ezeket képpáronként tökéletesítik, majd az így előállt összes mélység adatot egyesítik. Azok a pontok amelyek a szoftver implementációban megadott határértéken belül megfelelően sok kép alapján ugyanott vannak, bekerülnek a sűrű pontfelhőbe, a „hibás” pontokat pedig eldobják. A legtöbb program a rekonstrukció közben a pontok színének meghatározását is elvégzi.

A sűrű pontfelhő előállítása után egyes szoftverekben lehetőség van háló modell, ortofotó illetve további levezetett termékek (terepmodell, felszínmodell, stb.) készítésére is.

3. Felhasznált eszközök

Az eszközök kiválasztásánál igyekeztem olyan műszereket és szoftvereket választani, amelyek egy geodéziai vállalkozásban már valószínűleg eleve megvannak, vagy viszonylag kis költséggel beszerezhetők, így elsősorban amatőr kamerákban és ingyenes szoftverekben gondolkodtam.

3.1 Műszerek és számítógépek

A feldolgozás alapjául szolgáló fényképek elkészítéséhez igyekeztem minél nagyobb felbontású amatőr eszközt választani. Szóba jött egy saját Canon gyártmányú digitális fényképező gép, viszont az Általános- és Felsőgeodézia Tanszékről kölcsön kapott Sony DSC-WX350 eszköz valamivel nagyobb – 4896 x 3672 – felbontással tudott dolgozni, így végül erre esett a választás.

A fényképekből levezetett pontfelhő ellenőrzéséhez olyan eszközt kellett választani, melynek pontossága legalább egy nagyságrenddel jobb. A Leica TS15i robotmérőállomás gyári adatai szerint a műszer szögmérési pontossága 1”, míg távmérési pontossága 1 mm + 1,5 ppm, mely a vizsgált távolságban 1-2 milliméteres pontosságot jelent a bemért pontok pozíciójában. Mivel a pontfelhő várható pontossága a tapasztalatok szerint a terepi felbontás két-háromszorosa (jelen esetben 1-2 cm), a mérőállomás pontossága elegendő az eltérések kimutatásához. Ugyan ez a műszer a felsőbb ár kategóriákba tartozik, ilyen léptékben hasonlóan megfelelő pontosság elérhető olcsóbban beszerezhető műszerekkel is.

A fényképek feldolgozásához kézenfekvő volt egyrészt a saját Lenovo Legion 5 laptop használata, másrészt az Általános- és Felsőgeodézia Tanszék szervere, melyen a WebODM futtatása történt. A két számítógép lényegesebb jellemzői a következők:

Lenovo Legion 5 laptop:

- Windows 10 Home és Ubuntu 22.04.3 LTS operációs rendszer
- Intel Core i7-10750H CPU
- NVIDIA GTX 1660TI GPU (6GB VRAM)
- 16GB RAM + 16GB SWAP
- 32GB pendrive (Ubuntu 22.04.3)
- 512GB SSD (Windows 10 Home)
- 1TB HDD (adattárolás)

Általános- és Felsőgeodézia Tanszék szervere:

- Debian 10 (4.19 kernel)
- Intel Core i7-10700 CPU
- 16GB RAM + 120GB SWAP
- 1TB + 2TB SSD (adattárolás)

3.2 Szoftverek

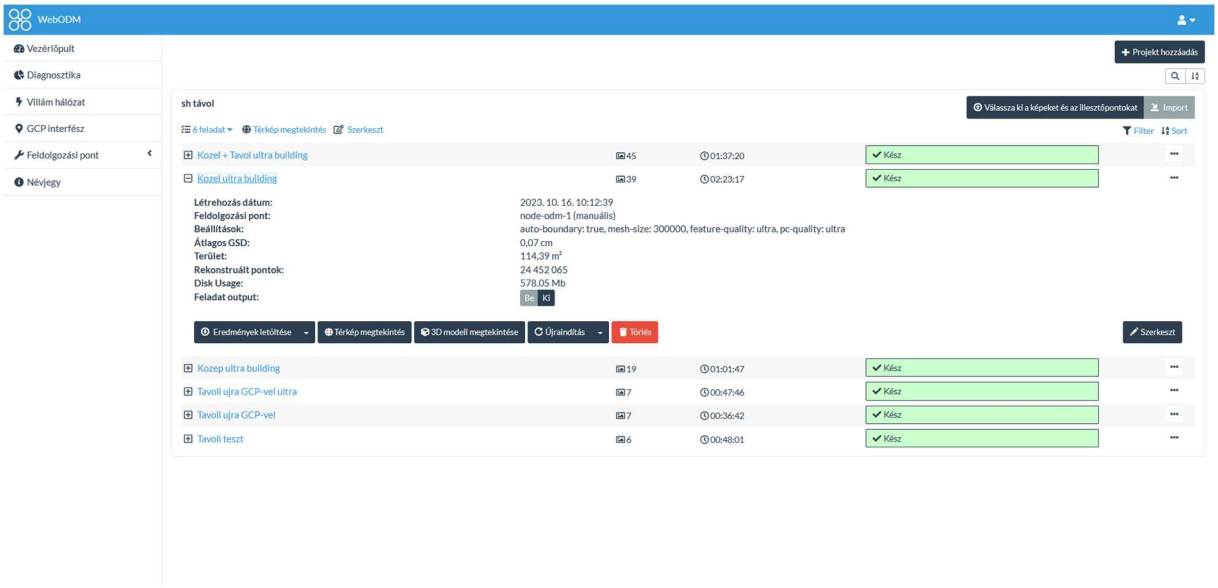
A szoftverek kiválasztásánál lényeges szempont volt, hogy nyílt forráskódú projektek legyenek, és aktív fejlesztés alatt álljanak, így a választás a feldolgozáshoz WebODM-re és Meshroom-ra, a kiértékeléshez pedig CloudCompare-re esett. ODM/WebODM eredetileg kifejezetten drónnal készített fotók feldolgozására készült, ezzel szemben Meshroom objektummodellezésre, és ezekből háromszög modellek előállítására specializált, így a két megoldás eredményeinek összehasonlítása is érdekes lehet. CloudCompare egy kifejezetten pontfelhők elemzésére kifejlesztett szoftver, mely jelentős méretű állományokat is megbízhatóan tud kezelni. A Matplotlib Python könyvtárat (mely szintén nyílt forráskódú), a grafikonok elkészítésére használtam.

3.2.1 Web Open Drone Map (WebODM)

WebODM egy szerveren futtatható és weben keresztül elérhető nyílt-forráskódú drón fotogrammetriára kifejlesztett program, mely a már régebb óta létező Open Drone Map (ODM) parancssoros feldolgozó szoftverre épül. Az első ODM beta verzió 2016-ban jelent meg, melyet

2020-ban követett az első teljes értékű verzió hosszú fejlesztés után. Az első WebODM verzió 2017-ben jelent meg, majd 2019-ben az 1.0 kiadás. Jelenleg ODM a 3.2.1-es verziónál tart, míg WebODM 2.2.0-nál. Az ODM alap Python nyelven íródott és számos nyílt forráskódú C/C++ nyelven készített könyvtárat használ fel, a WebODM grafikus felülete pedig JavaScript és Python keveréke.

A nagy előrelépés a WebODM megjelenésével a grafikus felhasználói felület (4. ábra). Ugyan a grafikus felületen kevesebb beállítási lehetőséget lehet elérni, mint a parancssoros verzióban, ez a felület, és a webes elérés megkönnyíti a szoftver alkalmazását.



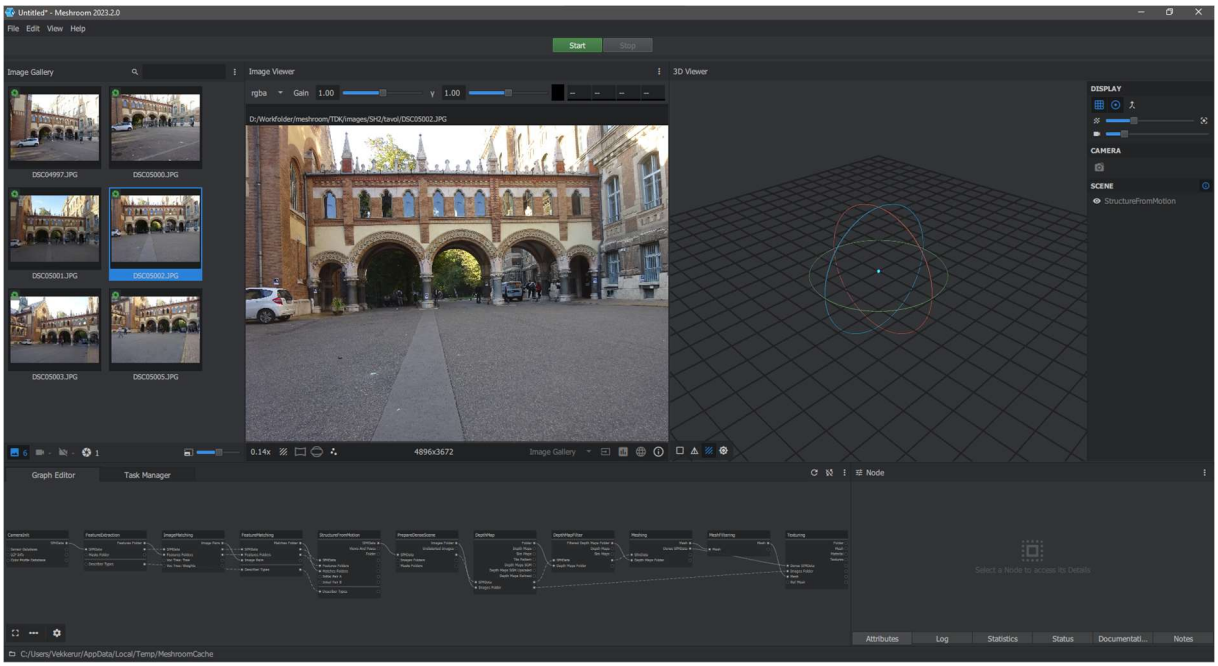
4. ábra WebODM felhasználói felület

Habár eredetileg drón fotogrammetriai feldolgozásra készült, manapság jól használható bármely közelfotogrammetriai feladatra is. Elő tud állítani sűrű pontfelhőt, ebből térháló modellt, ortofotót, terep- és felszínmodellt is (Caughlin, Wickersham, Zaiats, & Marie, 2023).

3.3.2 Meshroom

A Meshroom egy ingyenes, nyílt-forráskódú, 3D rekonstrukciós szoftver, mely az AliceVision gépi látás keretrendszerre épül. A program jelenlegi legújabb kiadása a 2023.2.0 verzió, fejlesztését főként az AliceVision Association végzi, de természetesen közösségi hozzájárulásokat is fogadnak (Griwodz, és mtsai., 2021). A programot 2010 óta fejlesztik, az

első teljes körű modellezési folyamatot támogató kiadása pedig 2018-ban jelent meg az AliceVision 2.0.0 keretrendszerre alapozva. Az AliceVision keretrendszer elsősorban C++ nyelven íródott, míg Meshroom Python-ban, és a Qt5 keretrendszert használja a grafikus felhasználó felületéhez (5. ábra). A teljes modellezési folyamathoz minimum CUDA Compute 3.5-öt támogató GPU-t igényel, ennek hiányában csak a ritka pontfelhőt lehet előállítani benne.



5. ábra Meshroom felhasználói felület

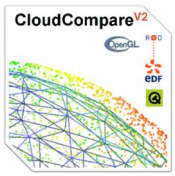
A szoftver főként 3D térhálós modellek előállítására készült, de emellett a „gyári" feldolgozási folyamatokkal használható kamera útvonal rekonstrukcióra és HDR (High Dynamic Range) panoráma képek előállítására is. Mivel az egyes feldolgozási lépések külön-külön blokkokban érhetőek el és paraméterezhetőek, az alapértelmezett a programba beépített feldolgozási folyamatok mellett a felhasználók egyéb tetszőleges feldolgozási folyamatokat is összeállíthatnak.

Feldolgozások futtathatók mind parancssoros interfészen keresztül, mind a grafikus felhasználói felületről. A feldolgozási sor grafikus felülete a napjainkban több területen egyre jobban elterjedő (pl.: Blender 3D Geometry Nodes, Unreal Engine Blueprint Visual Scripting) csomópont alapú elrendezést követi. A feldolgozás egyes lépéseit gráf szerű csomópontok tartalmazzák, a lépések a csomópontokon keresztül parametrizálhatók, a köztük lévő élek pedig az egyes lépések közötti ki- és bementi adat kapcsolatokat szemléltetik (6. ábra).

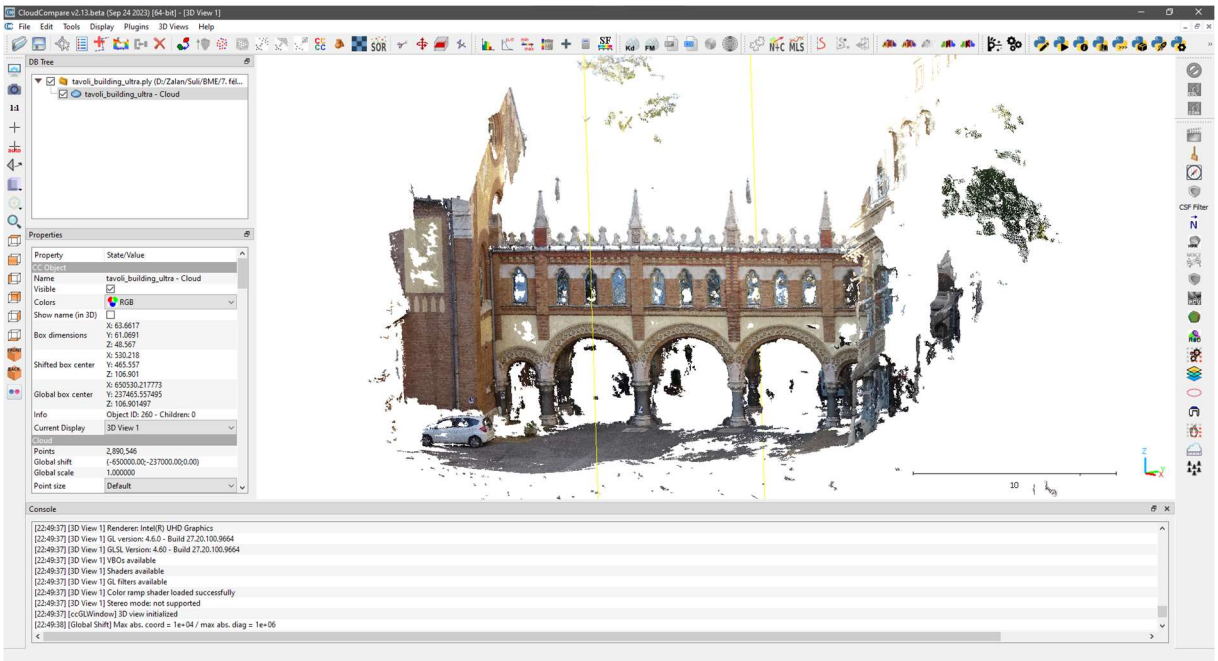
6. ábra Feldolgozási sor Meshroom-ban

A program minden lépés eredményeit külön mappákba menti, és a csomópontok beállításai alapján egyedi azonosítókkal látja el őket, így könnyen lehet lépkedni a különböző beállításokkal lefuttatott verziók eredményei között.

3.3.3 CloudCompare



CloudCompare egy nyílt-forráskódú 3D pontfelhő és hálós modell feldolgozó szoftver. Eredetileg sűrű pontfelhők egymás- és modellek közötti összehasonlítására készült, későbbi fejlesztések során azonban egy ennél általánosabb csomaggá nőtte ki magát. Több 10 millió pontból álló pontfelhők kezelése mellett már támogat pontfelhő illesztéseket, újra-mintavételezést, statisztikai számításokat és még sok más összetett funkciót. Jelenlegi legfrissebb kiadása a 2.13 beta verzió, mely az előzőekhez képest egy Python integrációval bővíti a lehetőségek amúgy is bőséges tárházát (7. ábra).



7. ábra CloudCompare felhasználói felület

A program bizonyos funkciói Meshroom-hoz hasonlóan a grafikus felület mellet elérhetők parancssorból is. CloudCompare C++ nyelven íródott, a grafikus felületet és megjelenítést pedig OpenGL és Qt hajtja.

3.3.4 Matplotlib

Matplotlib egy Python programnyelvhez írt ingyenes ábrázolási könyvtár. Elsősorban a NumPy matematikai számítási modul kiegészítése, és lehetővé teszi, hogy a MathWorks MATLAB környezethez hasonlóan, egyszerűen lehessen minőségi szemléltető ábrákat, diagrammokat előállítani számítási eredményekből. A könyvtár elsőként 2003-ban jelent meg, és jelenleg a 3.8.1-es kiadásnál tart.

A könyvtár tervezésének hála akár néhány sornyi program kóddal is gyorsan lehet szemléltetni a lényeges adatokat egyszerű ábrákon, de a teljes funkcionalitás ismeretében bármely bonyolult összeállítás is elkészíthető (8. ábra). Ideális numerikus adatok kiértékelésének segítéséhez.

8. ábra Grafikon példák a Matplotlib hivatalos weboldalán

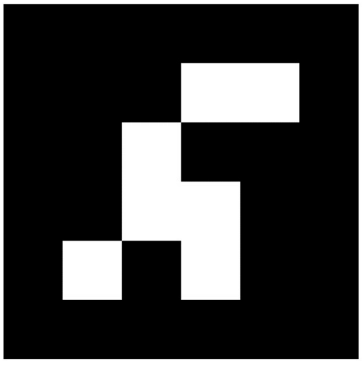
4. Felmérés

A felmérés végrehajtását úgy terveztem, hogy az illesztőpontok kijelölését, bemérését, és a fényképek elkészítését egy délután, minél rövidebb idő alatt el tudjam végezni, ezáltal elkerülve, hogy az illesztőpontok esetleg elmozduljanak a fényképezés előtt.

4.1 Illesztőpontok megjelölése

A fotogrammetriával előállított pontfelhők automatikus georeferálására használhatók előre kihelyezett illesztőpont markerek, melyek olyan jellemzően fekete-fehér mintázatok melyek számítógépes algoritmusokkal felismerhetők. Az egyes marker családok mintázatai egyediek és a mintában kódolva egyedi módon azonosítják az egyes markereket a családon belül. Az egyik ilyen marker család az ArUco, melynek egyik változata a 4x4 ArUco. Ezek a markerek egy négyzetes fekete kereten belül 4x4-es mátrixban fekete és fehér mezőkkel kódolják a marker azonosító számát, illetve követelmény, hogy a minta nem lehet forgási- és tengelyesen szimmetrikus, így az irányultság is egyértelműen meghatározható.

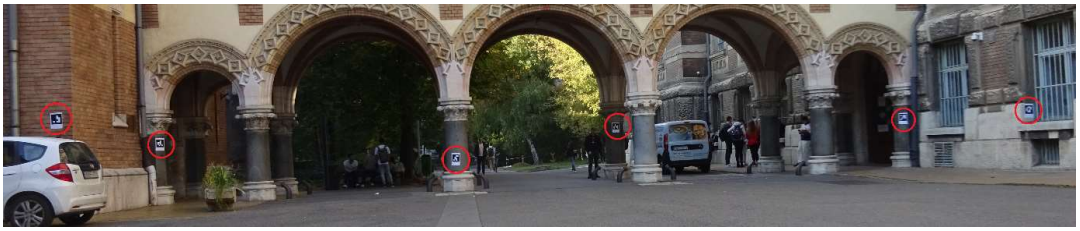
Eredetileg ilyen 4x4 ArUco markereket terveztem kihelyezni az illesztőpontok jelölésére és automatikus detektálására, melyeket az OpenCV Python verziót használó Find-GCP program felismer (Siki & Takács, 2021), és WebODM-ben használható illesztőpont koordinátákat tud előállítani. Kiderült azonban, hogy a Meshroom újabb verziói a Find-GCP-hez hasonlóan támogatják a 4x4 ArUco-hoz hasonló 16h5 AprilTag markerek automatizált felismerését, így inkább emellett a típus mellett döntöttem.



9. ábra AprilTag 16h5 marker

A vizsgált területen összesen 6 illesztőpontot jelöltem meg 20x20 cm-es AprilTag 16h5 markerekkel. A pontos georeferálás érdekében a pontokat különböző távolságokban és

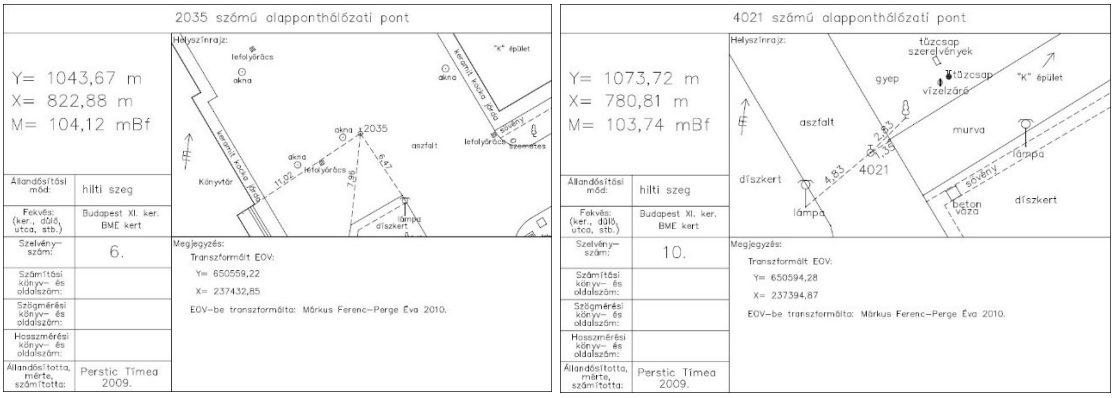
magasságokban, a területet lefedve raktam fel. Ugyan ideális lett volna az átjáró magasabb részeit is lefedni, sajnos a pontokat nem tudtam 2 méternél magasabbra helyezni.



10. ábra Kihelyezett illesztőpontok

4.2 Illesztő- és ellenőrző pontok bemérése

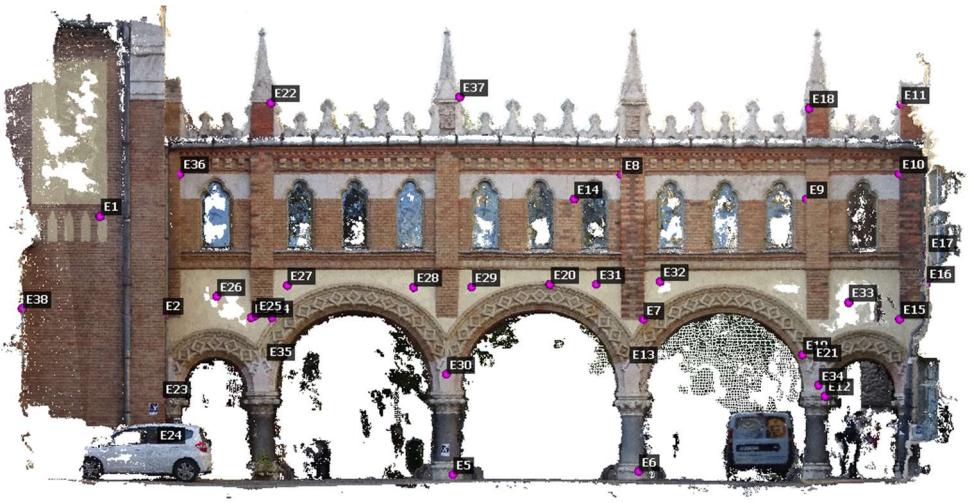
Az előállított pontfelhők transzformációjához a kihelyezett illesztőpontokat mérőállomással mértem be (12. ábra). Ehhez az egyetem kertjében található alapponthálózat pontjait (11. ábra) használtam fel.



11. ábra Alappontok pontleírásai

12. ábra Felállított Leica TS15i robot mérőállomás

A műszer felállítása után EOV koordinátákkal bemértem a 6 kihelyezett illesztőpontot. Ezen felül meghatároztam még 38 további ellenőrző pontot az átjárón (13. ábra). Ehhez jellemző, jól azonosítható pontokat választottam, hogy szükség esetén használhatók legyenek georeferálásra is (ha Meshroom-ban nem működne megbízhatóan a markerdetektálás), illetve néhány pontot a viszonylag részlet menetes síkfelületeken is.



13. ábra Ellenőrző pontok elhelyezkedése

4.3 Fényképezés

A fényképeket a korábban bemutatott Sony DSC-WX350 fényképezőgéppel készítettem, három különböző távolságból. Az első sorozatot körülbelül 20 méter távolságból összesen 6 képet, a második sorozatot 10-12 méterről összesen 18 képet, majd 5-7 méterről összesen 38 felvételt. A jobb metszési szögek érdekében igyekeztem nem csak a vízszintes irányán változtatni a képeknek, hanem a magasságán is, így a képek egy részét guggoló helyzetből, másik részét fej fölül készítettem.

A képeket igyekeztem a fentebb tárgyalt irányelvek szerint – kis ISO, és kis rekesznyílás mellett – készíteni (ISO 80 fényérzékenység, f/3,5 blende, 1/100s záridő). Sajnos az első sorozat készítésekor a fényviszonyok korántsem voltak ideálisak, így a képek letöltése után kiderült, hogy a legtöbb kép annyira kiégett, hogy nem lehetett volna jó eredményt kinyerni belőlük (fényképezéskor a helyszínen a gép kisméretű képernyőjén ez nem látszott). E miatt a fényképezést néhány órával később megismételtem, ekkor a nap már lemenőben volt, és a Sóhajok hídja nagyrészt árnyékba került (14. ábra).

14. ábra Kiégett és az ismételt fényképek

5. Feldolgozás

A terepi felmérés végeztével az elkészült összesen 62, JPG formátumú fényképet lementettem a fényképezőgépről, valamint a mérőállomásról CSV formátumban letöltöttem a bemért illesztő- és ellenőrző pontokat. Ezek után elkezdtem az állományok számítógépes feldolgozását.

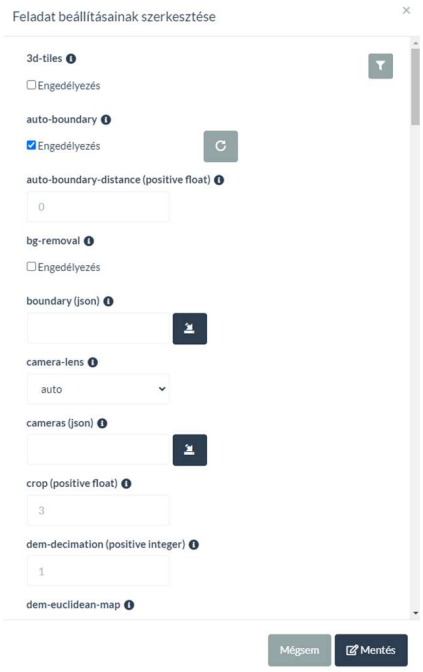
5.1 Előkészítés

A feldolgozás során igyekeztem minden olyan lépést automatizálni, amit lehet, így igyekeztem a pontfelhők georeferálását is a feldolgozás részeként kézi beavatkozás nélkül megoldani. WebODM-nek szerencsére kész megoldása van a problémára. A képek feltöltésekor lehetőség van egy „gcp_list.txt” fájl feltöltésére is, melyben az egyes felvételeken lévő illesztőpontokat lehet megadni képi- és valós 3D koordinátáikkal. Ennek az állománynak a generálására jól használható a Find-GCP, mely egy Python-ban írt kisegítő program. Az egyes képeken végiglépkedve, az OpenCV Python verziójára támaszkodva képes azonosítani sok különböző marker családot (beleértve az itt alkalmazott AprilTag 16h5 típust), és a mérőállomással bemért EOV koordinátákat tartalmazó állományt betöltve, létre tudta hozni a WebODM által használható lista fájlt (2. melléklet). Az alapértelmezett paraméterekkel ugyan csak kevés marker találatot tudott előállítani, de a „minrate” paramétert 0.01-re állítva már majd minden pontot sikeresen detektált, és szerencsére nem keletkezett hamis találat. Azon markerek, melyek széle összemosódott egyéb tárgyakkal, vagy túl éles szögben álltak a kamerához képest, nem lettek megjelölve.

WebODM-hez hasonlóan, az újabb Meshroom verziókban már lehetséges az előállított pontfelhő georeferálása, transzformálása. A transzformáció végrehajtható például kézzel megadott paraméterek szerint, fényképek meta-adataiból kinyert GPS pozíciók alapján és markerek segítségével is. Mivel Meshroom gyárilag nem támogatja a marker koordináták beolvasását külső állományokból, illetve nem lehet benne megadni a markerek képkoordinátáit, a feldolgozás automatizáláshoz elkészítettem néhány kiegészítő programot, melyeket a későbbiekben bemutatok. Emellett a Find-GCP segédprogram kódját is módosítottam, hogy a Meshroom számára értelmezhető eredmény fájlokat tudjon előállítani a felismert markerekre.

5.2 WebODM

A WebODM-ben futtatott feldolgozáshoz az Általános- és Felsőgeodézia Tanszék szerverét használtam. A három távolságból készült képek feldolgozását külön-külön futtattam, és mindegyikhez feltöltöttem a korábban előkészített illesztőpontokat leíró „gcp_list.txt” fájlt. A távolról készült 6 képen lefuttattam néhány tesztet, és ezek alapján a teljes feldolgozásoknál a legmagasabb pontfelhő- és kulcspont-leíró minőségi beállításokat használtam. A maximális pontsűrűség elérése érdekében a képek átméretezését letiltottam.



15. ábra WebODM beállítások felülete

16. ábra WebODM feldolgozási feladat felülete

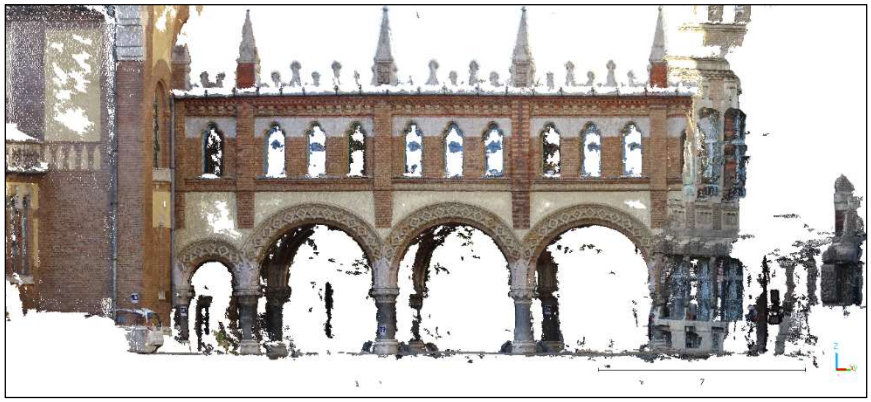
A távolról készült 6 fénykép feldolgozása 48 percig tartott, a közepes távról készült 18 képé 1 óra 2 percig, míg a közelről készült 38 kép 2 óra 23 perc alatt futott le. Ugyan nagyságrendileg jó tájékoztatást adnak, a futásidőkből komolyabb következtetést itt nem lehet levonni, mivel a szerveren a képek feldolgozásával egy időben egyéb más erőforrás igényes feladatok is futottak. A képcsoportok külön-külön feldolgozásán kívül WebODM-ben

lehetőségem volt két kombináció futtatására is, így feldolgoztam a közeli és távoli képeket együtt, illetve a közeli és középtávról készült képeket párosítva is. A kombinációk feldolgozásának célja megvizsgálni, hogy vegyes képtávolságok alkalmazásával javítható-e az eredmény, vagy a különbség elhanyagolható.

	Távol	Közép	Közel	Közel + Közép	Közel + Távol
Képek száma [db]	6	18	38	56	44
Rekonstruált pontok [db]	2 890 546	9 038 028	24 452 065	35 721 622	27 129 297
Átlagos terepi felbontás [cm]	0.12	0.09	0.07	0.08	0.08
Átlagos maradék ellentmondás az illesztőpontokban [mm]	13	6	8	6	6

1. táblázat WebODM feldolgozás főbb jellemzői

17. ábra WebODM, távoli képekből előállított pontfelhő



18. ábra WebODM, középtávról készült képekből előállított pontfelhő

19. ábra WebODM, közeli képekből előállított pontfelhő

20. ábra WebODM, közeli és középtávról készült képekből előállított pontfelhő

21. ábra WebODM, közeli és távoli képekből előállított pontfelhő

A pontfelhőkből (19. ábra, 20. ábra, 21. ábra) jól látszik, hogy a különböző képtávolságokat kombinálva javítani lehetett a lefedettségen. A közeli képeken nagyrészt kitakart tetődíszítő elemeket például rekonstruálni lehetett a távolabbi képek alapján.

A futtatások végeztével az elkészült pontfelhőket letöltöttem EOV vetületben, LAZ formátumban, valamint a PDF formátumú minőségi jelentéseket is. A LAZ állományokat, és az ellenőrző pontok állományát betöltve CloudCompare-ben számítottam az eltéréseket a 38 ellenőrző pontban pontfelhő-pontfelhő távolságként.

Az eltérések számításánál 10 cm-es küszöbértéket állítottam be. Azon pontok, melyek ennél nagyobb értéket kaptak, valójában nem képződtek le a pontfelhőben. A számítás után az ellenőrző pontokat elmentettem „ASCII Cloud” – karakteres – formátumban, hogy kinyerjem az eltérések értékeit az egyes pontokban. Az eltéréseket az ellenőrző- és illesztőpontokban a 4. melléklet és 5. melléklet tartalmazza, az értékelés pedig a 6.3 Pontfelhők pontossága-as fejezetben következik.

5.3 Meshroom

5.3.1 Telepítés

A Meshroom-ban végzett feldolgozás nem ment olyan gördülékenyen, mint WebODM. Ugyan a legújabb verzióban kiválasztható a „tag16h5” AprilTag opció a kulcspont keresésénél, kiderült, hogy valójában csak a program Linux verziójában működik, a Windows kiadásban az „Invalid describer type: tag16h5” üzenettel meghiúsul a futtatás. Kezdetben USB adathordozóról telepítés nélkül próbáltam meg egy Linux verziót futtatni, és abban elvégezni a feldolgozást, azonban az operációs rendszer telepítése nélkül nem tudtam a laptopban lévő videó kártyát használni. A megoldást egy 32 GB-os pendrive-ra telepített Ubuntu rendszer nyújtotta. A Windows rendszerekhez képest a Linux verziók kevesebb tárhelyet és memóriát igényelnek, így a külső adathordozóról futtatás teljesen működőképesnek bizonyult. A megfelelő illesztőprogramok telepítése után az NVIDIA videó kártya is kifogástalanul működött. A közepes távolságról és a közelről készült képek feldolgozásához a 16 GB memória nem bizonyult elegendőnek, így az SSD-n létrehoztam egy 16 GB-os virtuális-memória fájlt.

5.3.2 Saját kiegészítők

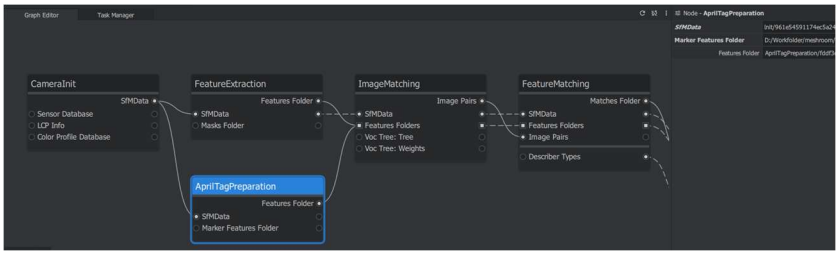
Ugyan a feldolgozási folyamatok a blokkokra bontásának köszönhetően könnyen átszabhatók, néhány a jelen vizsgálatban szükséges funkció csak nehezen, vagy kevéssé paraméterezhető. A program dokumentációjából kiderült, hogy mivel a program nagyrészt Python nyelven íródott, a grafikus felületen használható a blokkok maguk is Python kódból kerülnek betöltésre a Meshroom indításakor, így bizonyos keretek között lehet saját, egyedi blokkokat létrehozni. A minimális elérhető dokumentációt, valamint a létező blokkok kódját felhasználva elkészítettem két egyedi blokkot a feldolgozás segítéséhez. A blokkok forráskódja az 1. mellékletben van hivatkozza.

AprilTagPreparation blokk

Az első blokkot a markerek detektálásának kiegészítéséhez készítettem. Ugyan Meshroom támogatja az AprilTag 16h5 típusú markerek felismerését, mivel ezt együtt kezeli a SIFT és más kulcspont kereső algoritmusokkal, nincs lehetőség komolyabban paraméterezni a marker detektálást. A beállítások finomításának hiányában a tesztek alapján csak a kimondottal jól látható, és perspektivikusan nem torzult markereket tudja megtalálni a képeken, emellett a programban egyelőre nincs lehetőség a markerek képkoordinátáinak megadására. Hogy javítsam az automatikus transzformáció pontosságát, igyekeztem megoldást találni arra, hogy a Find-GCP találatait Meshroom-ban is használni lehessen, ehhez azonban dokumentáció híján komolyabb kutatást kellett folytatnom.

A „FeatureExtraction” blokk minden képhez létrehoz fájl párokat, melyek az azonosított kulcspontokat írják le a detektálási algoritmusnak megfelelő módon. Ha több különböző típusú kulcspont leírás is engedélyezve volt a blokkban, egy képhez több leíró pár is tartozhat. Az AprilTag marker pontok esetében a fájlok „kép_azonosító.tag16h5.feat” és „kép_azonosító.tag16h5.desc” nevet kapnak. A „.feat” kiterjesztésű fájlok ASCII karakteres állományok, melyek soronként egy-egy azonosított kulcspont képi koordinátáit, méretét és irányát írják le. A „.desc” kiterjesztésű állományok bináris adatokat tartalmaznak, és a „.feat” fájlokban listázott kulcspontok leíró adatait tartalmazzák. Az AprilTag markerek esetében ez a leíró adat kizárólag a marker azonosító indexére vonatkozik. (A fájlok formátumával bővebben a 3. melléklet foglalkozik.)

A leíró fájlok formátumának birtokában kiegészítettem a Find-GCP kódot, hogy a markerek detektálása után a Meshroom által használt állományokat is elő tudja állítani. Hogy az így elkészült leíró fájlokat használni tudjam Meshroom-ban, létrehoztam egy új „AprilTagPreparation” nevű feldolgozó blokkot, melynek az a feladata, hogy a leíró állományokat átnevezze a Meshroom által használt kép azonosítók alapján, és bemásolja őket a Meshroom munka könyvtárba.

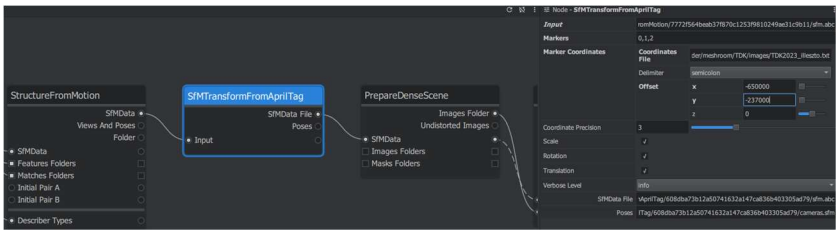


22. ábra AprilTagPreparation blokk használata

A funkció használatához az „AprilTagPreparation” blokkot párhozamosan be kell kötni a „FeatureExtraction” mellé (22. ábra). Bemenetként meg kell adni a „CameraInit”-ből érkező „SfMData” adatot, a „Features Folder” kimenetet pedig az „ImageMatching” blokkba kell kötni. Emellett meg kell adni a mappa elérési útját, ahol az előkészített leíró állományok vannak. Fontos, hogy a „FeatureExtraction” blokkban ne legyen engedélyezve a „tag16h5” kulcspont típus, viszont az „ImageMatching” blokkban pedig igen.

SfMTransformFromAprilTag blokk

A markerek detektálása mellett a transzformációs blokk beállításai voltak még kifogásolhatók. Az alap „SfMTransform” blokkban csak kézzel lehet bevinni az illesztőpont koordinátákat, és a program olyan szempontból sem robosztus, hogy a koordinátával megadott, de meg nem talált illesztőpontokat nem tudja automatikusan kihagyni, helyette a transzformáció meghiúsul. A koordináták betöltésének problémáját sikerült megoldani egy „SfMTransformFromAprilTag” nevű egyedi blokkal. A blokk kimenete ugyanaz, mint a gyári transzformációnak, de bemenetként kézi megadás helyett egy CSV fájlt fogad, melyet futtatáskor beolvas, és a benne talált koordináták alapján összeállítja a megfelelő parancsot a transzformációhoz. A blokkban megadható, hogy melyik azonosítójú markereket használja fel és a CSV fájl elválasztó karaktere, valamint egy XYZ koordináta eltolás. A lebegőpontos tört számok használata miatt az egyes programokban nagy koordináták esetén a pontosság romlik. Meshroom esetében például, ha valós EOV koordinátákkal georeferáljuk a pontfelhőt, a PLY formátumba átkonvertált állomány csak méteres élességgel tartalmazza az XY koordinátákat, és így az eredmény használhatatlan. Erre megoldást nyújthat a koordináták eltolása egy konstans értékkel oly módon, hogy az eltolt koordináták 0 közelében legyenek (a vizsgálat helyszínén például egy -650000, -237000 XY irányú eltolás elegendő).



23. ábra SfMTransformFromAprilTag blokk használata

A használathoz blokkot sorba kell kötni a „StructureFromMotion” és a „PrepareDenseScene” blokkok közé (23. ábra).

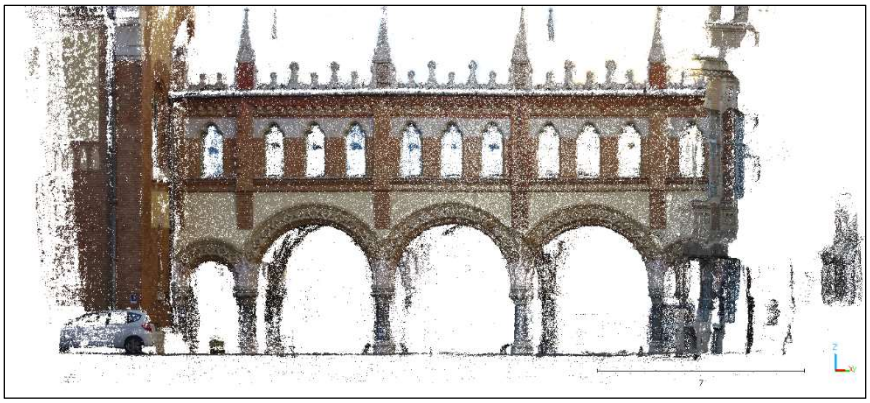
5.3.3 Futtatás

A feldolgozáshoz a használt laptopot a korábban előkészített Ubuntu pendrive-ról indítottam el. Az előkészített állományok és a saját kiegészítők birtokában összeállítottam egy részben egyedi feldolgozási sort. A standard fotogrammetria sablonból kiindulva a megfelelő helyekre beillesztettem a kiegészítő blokkokat, valamint kitöröltem a számomra szükségtelen modell textúrázás részt. A kamera modellt a Brown-féle 5 paraméteres modellre állítottam, a „FeatureExtraction” blokkban a kulcspontok minőségét és mennyiségét is a magas beállításra tettem, a „DepthMap” blokkban a felbontás csökkentést kikapcsoltam a „Meshing” blokkban engedélyeztem a sűrű pontfelhő színezését. A „Meshing” blokk után hozzáadtam két „ConvertSfMFormat” blokkot, melyek egyrészt átkonvertálták az eredmény „.abc” (Alembic fájl) állományt PLY formátumba, valamint a belső tájékozási paramétereket kimentették egy JSON állományba. A PLY és JSON kimeneteket pedig egy „Publish” blokk az eredmény kimenetek mappájába másolta.

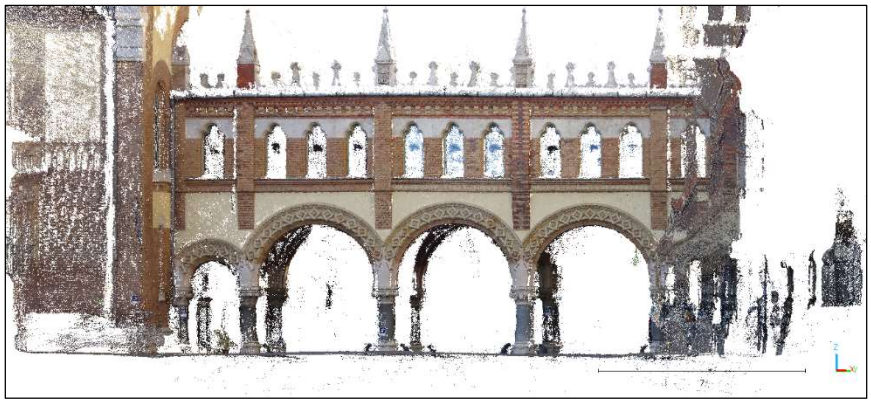
24. ábra Meshroom feldolgozási folyamat

	Távol	Közép	Közel
Képek száma [db]	6	18	38
Rekonstruált pontok [db]	755 498	2 258 190	3 200 876
Átlagos maradék ellentmondás az illesztőpontokban [mm]	14	8	9

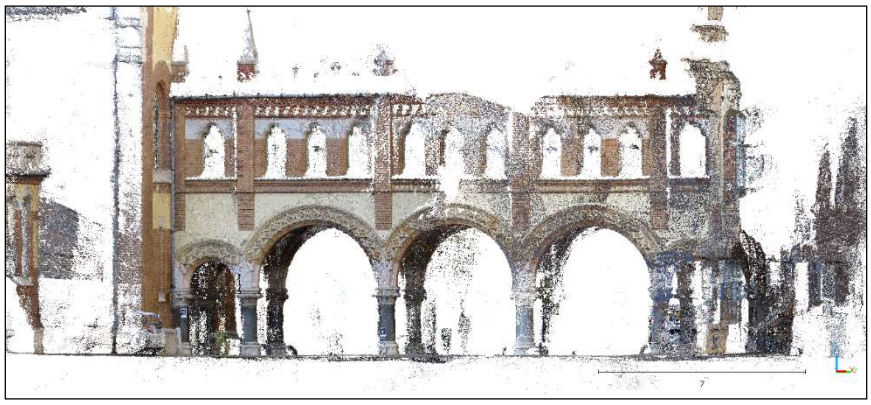
2. táblázat Meshroom feldolgozás főbb jellemzői



25. ábra Meshroom, közeli képekből előállított pontfelhő



26. ábra Meshroom, középtávról készült képekből előállított pontfelhő

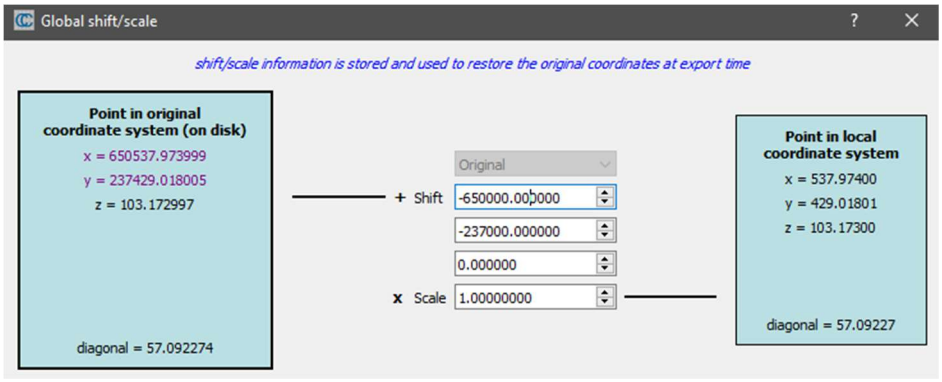


27. ábra Meshroom, közelről készült képekből előállított pontfelhő

A távolról készült 6 fénykép feldolgozásánál bőven elegendő volt a laptopban lévő 16GB RAM, azonban már a 18 középtávról készült képhez is éppen kevés volt, a 38 közeli képhez pedig különösen, ezért a memória kiegészítésére létrehoztam még egy 16GB-os SWAP (virtuális memória) fájlt. Ugyan a virtuális memóriából futtatás lassabb, de lehetővé tette a feldolgozás befejezését. A középtávról készült képekhez összesen 18-19GB memória volt szükséges, míg közeli képek feldolgozása közel 30GB-ot igényelt, és a többi futtatáshoz képest rendkívül lassú volt (a sűrű pontfelhő előállításánál kb. 15 perc alatt kifogyott a fizikai memória, és onnan kb. 75 percig tartott, míg a sokkal lassabb virtuális memóriával befejeződött ez eljárás, ahogy a 28. ábra mutatja). A memóriahiány miatt a két kombinált feldolgozást (középtáv + közeli és távoli + közeli) nem tudtam Meshroom-ban lefuttatni.

28. ábra Meshroom. memória használat a sűrű pontfelhő előállításánál a közeli képekből

A WebODM-ben végzett feldolgozáshoz hasonlóan az eltéréseket itt is CloudCompare-ben számítottam. Ehhez először betöltöttem az ellenőrző pontok állományát, majd pedig a PLY-ben tárolt pontfelhőket egyenként. A számítás előtt a Meshroom-ban beállított koordináta eltolást CloudCompare-ben is beállítottam az „Edit global shift and scale” funkcióval, hogy az algoritmus a megfelelő koordinátákat használja.



29. ábra Koordináta eltolás beállítása CloudCompare-ben

A távolságok számításához itt is 10 cm-es küszöböt állítottam be, majd az eredményeket „ASCII Cloud” formátumban mentettem ki. Az eltéréseket az ellenőrző- és illesztőpontokban a 4. melléklet és 5. melléklet tartalmazza.

6. Eredmények értékelése

6.1 Futási idők

Mivel WebODM és Meshroom teljesen különböző hardvereken futottak, a tanszéki szerveren egyéb feldolgozások is futottak párhuzamosan, illetve mivel a laptopon egyértelmű memória hiány volt, a futási időket nem lehet érdemben összehasonlítani. Az azonban elmondható, hogy ha a használt laptopban lett volna megfelelő mennyiségű fizikai memória, és nem kellett volna a sokkal lassabb virtuális memóriával dolgozni a közeli és középtávú képek feldolgozásánál, akkor lényegesen gyorsabban lefutott volna a folyamat mint a WebODM a tanszéki szerveren. Köszönhető ez az NVIDIA GPU-nak, melyet Meshroom a sűrű pontfelhő előállításakor érdemben használ (a WebODM-et futtató szerverben jelenleg nincs GPU, így ott a teljes folyamat CPU-n zajlott). Érdekes tapasztalat, hogy a különböző távolságból készült felvételek kombinálása a futási időt csökkentette.

Képtávolság	Képek száma	Futási idők [óra:perc]	
		WebODM	Meshroom
Távol	6	0:48	0:12
Közép	18	1:02	0:45
Közel	38	2:23	2:35
Közel + közép	56	2:21	-
Közel + távol	44	1:37	-

3. táblázat Feldolgozási idők összehasonlítása

6.2 Pontfelhők sűrűsége

Ahogy a 4. táblázat mutatja, nagyságrendi különbség van a két szoftverben előállítható pontfelhők sűrűségében. Ez nagyrészt annak tudható be, hogy mivel Meshroom elsősorban objektum modellezésre – és azon belül is térhálós háromszög modellek előállítására – van tervezve, a nyers sűrű pontfelhőt WebODM-nél sokkal agresszívabban szűri, és így jellemzően egy nagyságrenddel kevesebb pontot állít elő. Valójában a tapasztalatok szerint a teljes feldolgozás egyik legerőforrásigényesebb része a szűrés, és sok RAM-ot igényel (28. ábra).

Képtávolság	Rekonstruált pontok [db]	
	Meshroom	WebODM
Közel	3 200 876	24 452 065
Közép	2 258 190	9 038 028
Távol	755 498	2 890 546

4. táblázat Rekonstruált pontok számának összehasonlítása

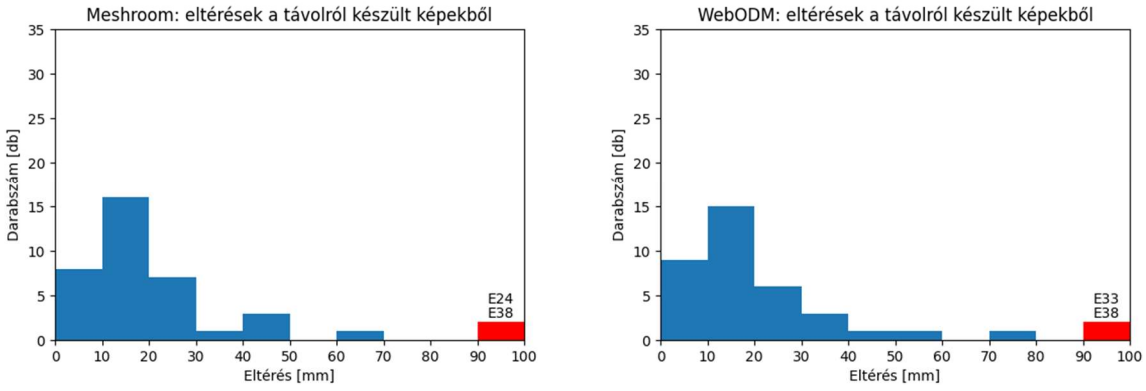
Ha rendelkezésre állna GPU, a WebODM-ben végzett feldolgozások is jelentősen felgyorsulnának. Erre nézve több részletes dokumentáció is található az internetes közösségi fórumon (pl.: (Shiva, 2022)).

6.3 Pontfelhők pontossága

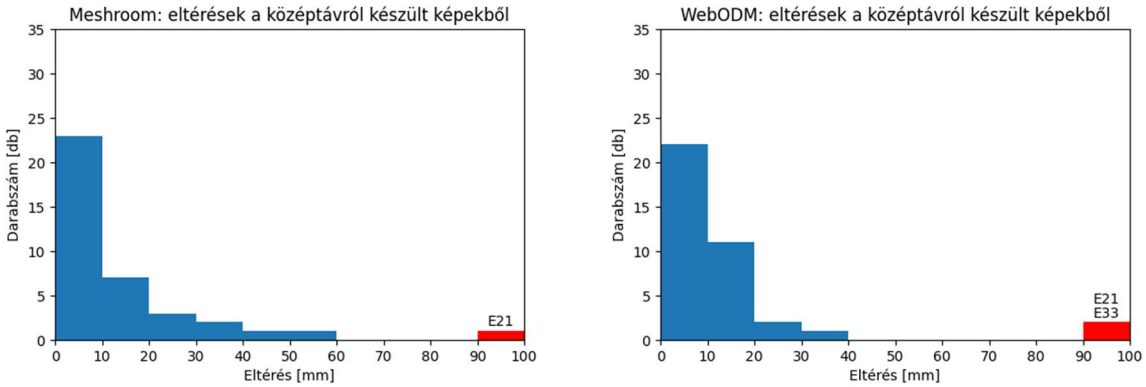
A hisztogramokon látszik, hogy a feldolgozott kép csoportokban Meshroom-ból és WebODM-ből hasonló az eltérések eloszlása (30. ábra, 31. ábra és 32. ábra), az átlagos eltérések is közel megegyeznek (5. táblázat).

Képtávolság	Átlagos eltérések [mm]	
	Meshroom	WebODM
Távol	19	19
Közép	12	10
Közel	11	10

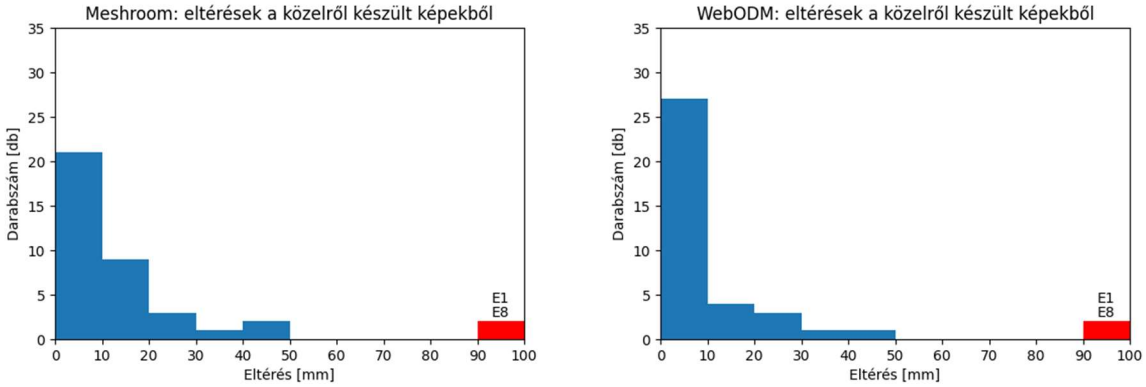
5. táblázat Átlagos eltérések az ellenőrző pontokban



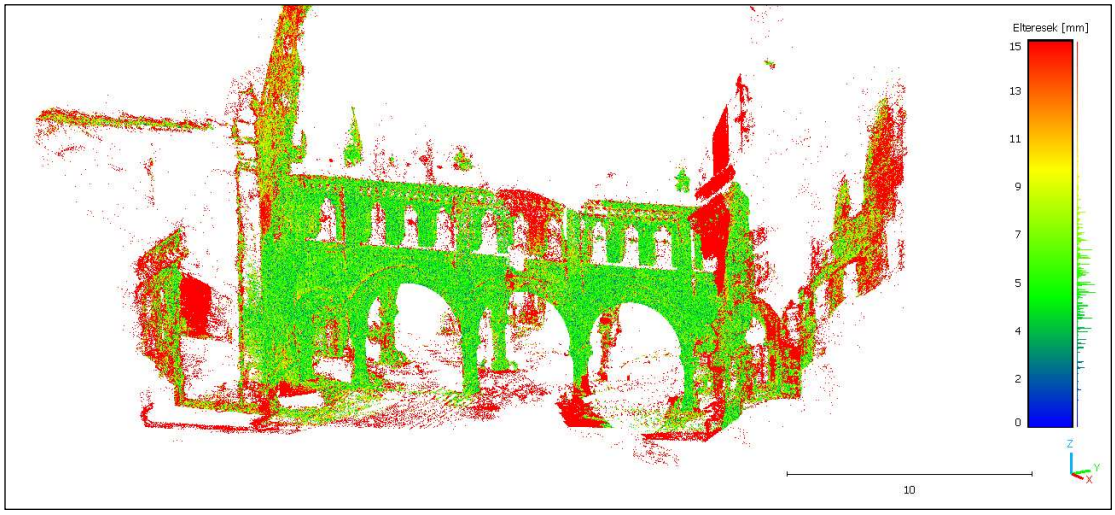
30. ábra Ellenőrző pontokban számított eltérések eloszlásai, távoli képek



31. ábra Ellenőrző pontokban számított eltérések eloszlásai, középtávról készült képek



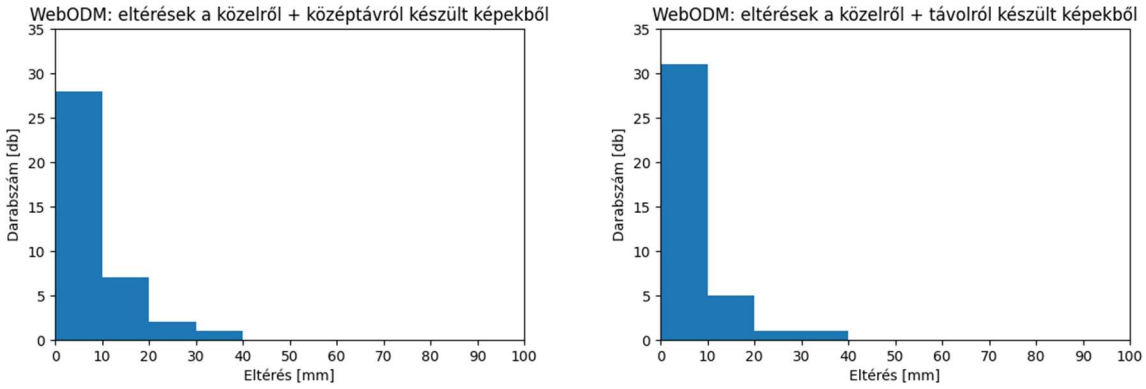
32. ábra Ellenőrző pontokban számított eltérések eloszlásai, közeli képek



33. ábra Meshroom közeli pontfelhő a WebODM eredményhez hasonlítva

A közeli képekből a Meshroom-mal előállított ritkább pontfelhőt a WebODM eredményéhez hasonlítva elmondható, hogy az eltérések zöme 10 mm alatti, és a 15 mm-nél nagyobb eltérések főleg a nem átfedő területeken látszanak (33. ábra). Az egyetlen kivétel ez alól az átjáró felső részének egy darabja, ahol a fényképek valamivel gyengébb átfedése miatt Meshroom több pontot is elvetett, és ezért ezen a részen az átlagosnál ritkább a pontfelhő.

Mivel WebODM-ben fel tudtam dolgozni kombinációkat is, az így előállított pontfelhők eltéréseit is számoltam. A közeli képeket kombinálva a távoli, illetve a középtávról készül képekkel, a pontosság javulása látható. Nincsenek le nem képződött ellenőrző pontok, és 40 mm-nél nagyobb eltérések sem. A közeli és távoli képek kombinációja esetén az ellenőrző pontoknál 80%-ban 1 cm vagy az alatti az eltérés (34. ábra).



34. ábra Ellenőrző pontokban számított eltérések eloszlásai, vegyes képek

Összességében a legjobb eredményt a WebODM-ben feldolgozott közeli és távoli képek keveréke adta, ahol mindegyik ellenőrző pont leképződött, az átlagos eltérés 8 mm, és az eltérések 80%-a 1 cm vagy az alatti.

Fontos megjegyezni, hogy kézi georeferálás esetén vizsgálni kellene azt is, hogy a transzformáció hibája mennyi plusz hibát vitt az ellenőrző pontokba, azonban mivel itt most a transzformáció is közvetlen, automatikus része a folyamatnak, az egész terméket egyben vizsgáljuk.

6.4 Szoftverek

A WebODM-et futtató szerver régóta használatban van, így a WebODM beállításával nem voltak gondok, a feldolgozás problémák nélkül lefutott. Ezzel szemben a Meshroom-os feldolgozás az 5.3.1-2-ben leírt módon meglehetősen körülményes volt. A probléma főleg az AprilTag markerek használatából eredt, ami miatt a program Linux verzióját kellett futtatni, viszont a hibákat sikerült kiküszöbölni, és a programot az 5.3.2-ben leírt módon ki is tudtam egészíteni a nagyobb automatizáció érdekében.

Ugyan WebODM kevesebb problémával működött, nem elhanyagolható, hogy fix feldolgozási folyamatok közül választhat a felhasználó. Ezzel szemben Meshroom igen rugalmas a feldolgozási lépések tekintetében, a rendelkezésre álló program blokkokból

tetszőleges eljárások állíthatók össze, és bizonyos keretek között bővíthető is a teljes kódbázis újra fordítása nélkül is. Az egyéni bővítmények készítésének legnagyobb nehézsége, hogy Meshroom-nak ezen lehetőségei csak alig dokumentáltak, csak az eredeti forráskód birtokában sikerült minden problémát leküzdeni. Megjegyzem, hogy zárt forráskódú szoftverek esetén, ilyen fajta egyéni kiegészítés egyáltalán nem is lett volna lehetséges.

7. Összefoglalás

A vizsgálat során igyekeztem olyan eszközöket alkalmazni, melyekkel egy geodéziai vállalkozás jó eséllyel rendelkezik, vagy kis ráfordítással beszerezheti, így a felmérést amatőr kamerával és mérőállomással végeztem, a feldolgozáshoz pedig ingyenes, nyílt-forráskódú szoftvereket használtam. Az eredmények alapján elmondható, hogy javít a pontosságon a vegyes képtávolságok használata. A legjobb eredményt ugyan a közeli és távoli képek kombinációja adta, de a közeli képeket a középtávról készültekkel kombinálva is kimutathatóan javult a levezetett pontfelhő pontossága.

A nyílt-forráskódú szoftverek a vizsgálatban több előnnyel jártak egyes kereskedelmi szoftverekkel szemben. Egy részről a nyílt-forráskódú programok használatához sokszor valamivel nagyobb felkészültség és több kreativitás szükséges, mint a kereskedelmi megfelelők esetén, ugyanakkor ezek a közösségi fejlesztésű szoftverek általában rugalmasabbak, így kísérletezésre, kutatási feladatokban a használatuk kedvezőbb. A kereskedelmi szoftverek sokszor kevésbé paraméterezhetők, és az esetleges hibák kijavítása is nehézkes lehet, szemben a nyílt-forráskódú szoftverekkel, ahol szaktudás függvényében szükség esetén akár saját magunk is eszközölhetünk változtatásokat. Ezt az előnyt a jelen vizsgálatnál is kihasználtam a nagyobb automatizálás érdekében. Emellett a tapasztalatok szerint a kereskedelmi és a kiforrott nyílt-forráskódú megoldások között nincs lényeges különbség az eredmény pontosságának tekintetében. Megfelelő forrásanyag és beállítások mellett hasonló az elérhető pontosság (Nagy, 2021), (Vacca, 2019).

A feldolgozás terén fontos megemlíteni, hogy a fotogrammetriai feldolgozás rengeteg számítást igényel, így a memória szükséglet is igen magas. Nagy mennyiségű fénykép feldolgozásához sok fizikai memória szükséges, és bár ez bővíthető virtuális memóriával, ez jelentősen növeli a futási időket. Meshroom-ban már a felhasznált 38 közeli kép feldolgozásához sem volt elég a 16GB fizikai memória, és ki kellett bővíteni még 16GB SWAP-pal. A WebODM-ben több száz kép együttes feldolgozásához 128GB virtuális memóriát kellett beállítani a fizikai 16GB mellé. A teljes folyamathoz szükséges idő jelentősen csökkenthető egy megfelelő GPU használatával, melyet bizonyos szoftverek alkalmazni tudnak a kulcspontok keresésénél, és a sűrű pontfelhő előállításnál.

A két alkalmazott szoftverről elmondható, hogy az előállítható adatmennyiség Meshroom-ban jóval kisebb, ugyanakkor a pontosságuk összességében hasonló. Mivel a futási sebességek a hardver különbségek miatt itt nem voltak összehasonlíthatók, a jövőben érdekes lehet egy olyan számítógép összeállítása, melyen mind a két programot lehet futtatni, és azonos környezetben már lehetne a sebességet is érdemben vizsgálni.

A dolgozat elkészítése közben született egyedi kiegészítő program részleteket lehetne még tovább fejleszteni, és némi tisztázás után a nyílt-forráskódú projektekhez hozzájárulásként benyújtani.

8. Ábra- és táblázatjegyzék

9. Irodalomjegyzék

Caughlin, T. T., Wickersham, R., Zaiats, A., & Marie, V. (2023). *Open Drone Map: Structure-from-Motion Workflow.* Boise State University. doi:10.7923/92HF-GP09

Griwodz, C., Gasparini, S., Calvet, L., Gurdjos, P., Castan, F., Maujean, B., . . . Lillo, G. (2021). AliceVision Meshroom: An open-source 3D reconstruction pipeline. *Proceedings of the 12th ACM Multimedia Systems Conference.* ACM Press. doi:10.1145/3458305.3478443

Nagy, Z. (2021). *Fotogrammetriai szoftverek összehasonlító vizsgálata.* TDK dolgozat, Budapesti Műszaki és Gazdaságtudományi Egyeteme, Építőmérnöki Kar.

Nagy, Z. (2022). *Fotogrammetria és lézerszkennelés együttes használata műemlék jellegű épület felmérésénél.* Diploma terv, Budapesti Műszaki és Gazdaságtudományi Egyetem, Építőmérnöki Kar.

Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-Motion Revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Las Vegas. doi:10.1109/CVPR.2016.445

Shiva. (2022). *Hardware Recommendations – CPU cores, graphics card, CUDA, NVIDIA, memory, RAM, storage.* Letöltés dátuma: 2023.. 10. 28., forrás: OpenDroneMap Community: https://community.opendronemap.org/t/hardware-recommendations-cpu-cores-graphics-card-cuda-nvidia-memory-ram-storage/13705

Siki, Z., & Lehoczky, M. (2020). Fotogrammetriai feldolgozószoftverek. *Geodézia és Kartográfia (72. évf.)*, 23.

Siki, Z., & Takács, B. (2021). Automatic Recognition of ArUco Codes. *Baltic Journal of Modern Computing*. doi:10.22364/bjmc.2021.9.1.06

Vacca, G. (2019). *Overview of open source software for close range photogrammetry.* University of Cagliari, Department of Civil, Environmental and Architecture Engineering. doi:10.5194/isprs-archives-XLII-4-W14-239-2019

10. Mellékletek

1. melléklet: Egyedi feldolgozási blokkok forráskódjai

AprilTagPreparation:

https://gist.github.com/MrClock8163/87fc2a1d162e53cf4ca71a340f3c680b

SfMTransformFromAprilTag:

https://gist.github.com/MrClock8163/f48ccf93d589f33a217aa459cdfe2ba3

2. melléklet: WebODM gcp_list.txt

A fájl első sorában az alkalmazott vetület azonosítója szerepel. Az utána következő sorokban az illesztőpontok 3D-s koordinátája után a képkoordináták a képfájl neve és az illesztőpont egyedi azonosítója található.

```
EPSG:23700
650541.346 237445.675 105.686 4026 2208 DSC05025.JPG 1
650543.348 237442.130 106.084 1668 1907 DSC05025.JPG 0
650541.346 237445.675 105.686 1430 1928 DSC05027.JPG 1
650541.346 237445.675 105.686 1180 3262 DSC05029.JPG 1
650541.346 237445.675 105.686 3224 1644 DSC05030.JPG 1
650541.346 237445.675 105.686 3062 2606 DSC05031.JPG 1
650546.933 237449.903 105.013 3163 1948 DSC05033.JPG 2
650546.933 237449.903 105.013 3329 3538 DSC05034.JPG 2
650541.346 237445.675 105.686 2762 1502 DSC05036.JPG 1
650541.346 237445.675 105.686 2738 2831 DSC05037.JPG 1
650556.310 237457.278 105.561 3526 1885 DSC05042.JPG 6
650556.310 237457.278 105.561 3742 3262 DSC05043.JPG 6
650556.310 237457.278 105.561 3232 1614 DSC05045.JPG 6
650548.663 237455.590 105.742 785 1456 DSC05045.JPG 3
650556.310 237457.278 105.561 3393 2340 DSC05046.JPG 6
650548.663 237455.590 105.742 972 2339 DSC05046.JPG 3
650548.663 237455.590 105.742 454 3268 DSC05047.JPG 3
650556.310 237457.278 105.561 3118 3210 DSC05047.JPG 6
650541.346 237445.675 105.686 2210 1849 DSC05048.JPG 1
650541.346 237445.675 105.686 1387 3581 DSC05050.JPG 1
650548.663 237455.590 105.742 1544 3468 DSC05052.JPG 3
650556.310 237457.278 105.561 2798 1507 DSC05054.JPG 6
650556.310 237457.278 105.561 4019 2874 DSC05057.JPG 6
650541.346 237445.675 105.686 644 2839 DSC05058.JPG 1
650556.310 237457.278 105.561 4452 2945 DSC05059.JPG 6
650548.663 237455.590 105.742 1018 2817 DSC05059.JPG 3
650541.346 237445.675 105.686 499 2599 DSC05060.JPG 1
650559.170 237454.281 105.505 2930 2330 DSC05006.JPG 4
650548.663 237455.590 105.742 2512 2334 DSC05007.JPG 3
650559.170 237454.281 105.505 4705 2437 DSC05007.JPG 4
650541.346 237445.675 105.686 58 2317 DSC05007.JPG 1
650548.663 237455.590 105.742 4776 2332 DSC05008.JPG 3
650559.170 237454.281 105.505 3096 2925 DSC05009.JPG 4
650548.663 237455.590 105.742 1046 2748 DSC05009.JPG 3
650548.663 237455.590 105.742 3200 2735 DSC05010.JPG 3
650541.346 237445.675 105.686 880 2717 DSC05010.JPG 1
650541.346 237445.675 105.686 2073 2510 DSC05011.JPG 1
650548.663 237455.590 105.742 4478 2454 DSC05011.JPG 3
650559.170 237454.281 105.505 3415 2160 DSC05012.JPG 4
650556.310 237457.278 105.561 2546 2130 DSC05012.JPG 6
650548.663 237455.590 105.742 543 2101 DSC05012.JPG 3
650548.663 237455.590 105.742 2488 2150 DSC05014.JPG 3
650546.933 237449.903 105.013 1161 2417 DSC05014.JPG 2
650556.310 237457.278 105.561 4540 2189 DSC05014.JPG 6
650546.933 237449.903 105.013 3600 2427 DSC05015.JPG 2
650559.170 237454.281 105.505 4404 3486 DSC05016.JPG 4
650556.310 237457.278 105.561 3346 3466 DSC05016.JPG 6
650548.663 237455.590 105.742 1266 3497 DSC05016.JPG 3
650548.663 237455.590 105.742 3337 3410 DSC05017.JPG 3
650541.346 237445.675 105.686 3072 3471 DSC05018.JPG 1
650543.348 237442.130 106.084 2118 3340 DSC05018.JPG 0
650556.310 237457.278 105.561 2770 2190 DSC05019.JPG 6
650548.663 237455.590 105.742 1467 2182 DSC05019.JPG 3
650541.346 237445.675 105.686 614 2356 DSC05020.JPG 1
650543.348 237442.130 106.084 232 2110 DSC05020.JPG 0
650543.348 237442.130 106.084 3190 2290 DSC05021.JPG 0
650541.346 237445.675 105.686 3461 2510 DSC05021.JPG 1
650556.310 237457.278 105.561 4814 3467 DSC05023.JPG 6
650556.310 237457.278 105.561 2828 3275 DSC05024.JPG 6
650548.663 237455.590 105.742 1430 3272 DSC05024.JPG 3
```

```
650546.933 237449.903 105.013 2323 2455 DSC04997.JPG 2
650543.348 237442.130 106.084 1040 2286 DSC04997.JPG 0
650548.663 237455.590 105.742 2321 1598 DSC05000.JPG 3
650556.310 237457.278 105.561 3329 1592 DSC05000.JPG 6
650546.933 237449.903 105.013 1938 1713 DSC05000.JPG 2
650543.348 237442.130 106.084 564 1480 DSC05000.JPG 0
650546.933 237449.903 105.013 2148 2528 DSC05001.JPG 2
650559.170 237454.281 105.505 4668 2466 DSC05001.JPG 4
650541.346 237445.675 105.686 878 2406 DSC05001.JPG 1
650548.663 237455.590 105.742 2850 2391 DSC05001.JPG 3
650543.348 237442.130 106.084 454 2334 DSC05001.JPG 0
650548.663 237455.590 105.742 2793 1829 DSC05002.JPG 3
650546.933 237449.903 105.013 2096 1966 DSC05002.JPG 2
650541.346 237445.675 105.686 805 1902 DSC05002.JPG 1
650543.348 237442.130 106.084 354 1797 DSC05002.JPG 0
650548.663 237455.590 105.742 3511 2385 DSC05003.JPG 3
650541.346 237445.675 105.686 1713 2376 DSC05003.JPG 1
650548.663 237455.590 105.742 3403 1750 DSC05005.JPG 3
650543.348 237442.130 106.084 1107 1679 DSC05005.JPG 0
```

3. melléklet: Meshroom leíró fájlok formátuma

.tag16h5.feat

A „.feat” fájlok ASCII karakteres állományok, melyeknek minden sora 1-1 kulcspontot azonosít a következő adatokkal:

x koordináta, y koordináta, kulcspont mérete, kulcspont irányultsága

Az AprilTag 16h5 markerek esetében minden markernek 5 pontját tárolja az állomány, kötött sorrendben: középpont (c), bal alsó sarok (bl), jobb alsó sarok (br), jobb felső sarok (tr), bal felső sarok (tl). A kulcspont iránya minden esetben 0, hiszen a markereket nem gradiensek alapján azonosítja, így az irányultság nem releváns. A kulcspont mérete a négy sarokpont által határolt négyszög maximális oldalhosszának a fele, a négy oldalból, és a két átlóból számítva (ez az érték az egy markerhez tartozó 5 pont esetén megegyezik). A számítást a következő Python kódrészlet szemlélteti:

```
max_sides = max(norm(tl-bl), norm(bl-br), norm(br-tr), norm(tr-tl))
max_diagonal = max(0.707*norm(tl-br), 0.707*norm(tr-bl)) # 0.707 a sqrt(2)/2-nek egyszerűsített alakja
size = 0.5 * max(max_sides, max_diagonal)
```

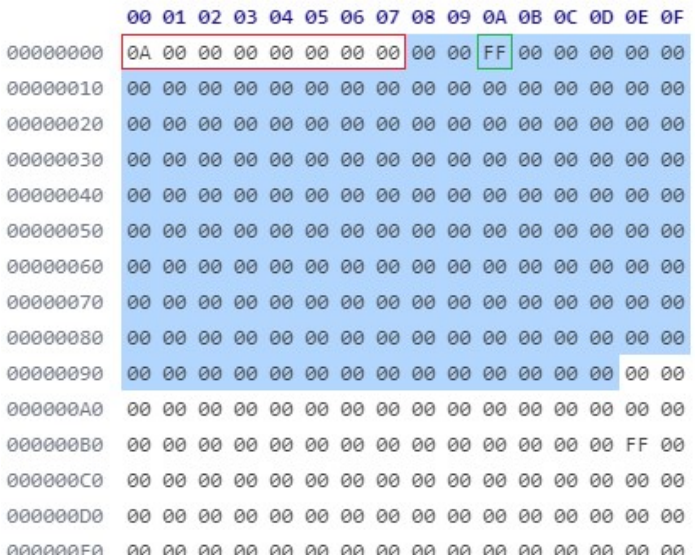
Példa egy „.tag16h5.desc” állomány tartalmára:

```
2323 2455 15.5081 0
2308 2470 15.5081 0
2337 2471 15.5081 0
2338 2440 15.5081 0
2309 2439 15.5081 0
1040 2286 25.0200 0
1014 2310 25.0200 0
1064 2308 25.0200 0
1065 2262 25.0200 0
1015 2263 25.0200 0
```

A sorok száma minden esetben a markerek számának ötszöröse.

.tag16h5.desc

A „.desc” fájlok minden esetben bináris állományok, melyek a kapcsolódó „.feat” fájlban listázott kulcspontok matematikai leírásait tartalmazzák. Az AprilTag 16h5 markerek esetében a bináris leírás csak a marker azonosító indexétől függ.



35. ábra tag16h5.desc fájl szerkezete hexadecimális nézetben

A fájl LE (Little-Endian) byte sorrendet alkalmaz. Az első 8 byte (35. ábra, piros jelölés) egy 64-bites előjel nélküli egyész szám, amely a fájlban tárolt leíró blokkok számát mutatja. Ezután a leíró adat típusától függő hosszúságú adat mezők következnek a fájl elején jelzett mennyiségben. Egy-egy leíró blokk hossza az AprilTag markerek esetében 150 byte (35. ábra, kék kijelölés), míg SIFT és több más kulcspont típusnál például csak 128 byte.

A 150 byte-os blokkban egyetlen byte értéke FF (35. ábra, zöld jelölés), a többi érték mind 00. Az FF byte elhelyezkedése azonosítja, hogy a hozzá tartozó „.feat” fájl sor melyik marker melyik pontját (c, bl, br, tr, tl) azonosítja.

Az alábbi Python kódrészlet szemlélteti az FF byte helyének számítását:

```
marker_id = 2
point = "c" # középpont
point_types = ["c", "br", "bl", "tl", "tr"]

# FF byte helye az adatmező elejétől számolva, 0-val kezdődő indexeléssel
ff_offset = marker_id + point_types.index(point) * 30 # 2
```

Mivel a „.feat” fájlban azonosított miden ponthoz tartozik egy leíró blokk a „.desc” állományban, a bináris formátumban is 5 blokk tartozik egyetlen markerhez. A következő Python kódrészlet bemutatja egy „.desc” állomány elkészítését adott azonosítójú markerekhez:

```
import struct

marker_ids = [2, 3, 5]

with open("output.tag16h5.desc", "wb") as file:
    file.write(struct.pack('<Q', len(marker_ids) * 5))

    for id in marker_ids:
        for i in range(5):
            data = bytearray(150)
            data[id + i * 30] = 255

            file.write(data)
```

A bináris állomány hossza mindig $8 + 5 \times 150 \times$ *markerek száma* byte.

4. melléklet: Eltérések az ellenőrző pontokban

Pontszám	Eltérések [mm]							
	Meshroom			WebODM				
	Távol	Közép	Közel	Távol	Közép	Közel	Közel + Közép	Közel + Távol
E1	12	21	-	11	4	-	11	4
E2	9	11	2	9	5	9	9	5
E3	8	2	1	8	2	8	2	2
E4	13	21	4	14	9	12	4	14
E5	9	8	6	8	6	5	5	5
E6	11	5	5	6	6	8	6	6
E7	19	5	6	14	5	5	5	5
E8	48	53	-	44	35	-	19	39
E9	40	19	20	36	16	5	5	5
E10	19	8	13	34	12	12	7	7
E11	16	9	10	25	5	6	5	6
E12	23	11	6	7	6	6	6	6
E13	28	10	3	24	17	8	15	18
E14	20	5	12	19	5	9	10	6
E15	21	4	6	3	7	7	3	7
E16	6	21	22	31	12	17	21	9
E17	23	8	10	27	15	5	5	5
E18	5	12	40	15	12	3	3	8
E19	13	5	12	11	5	10	7	9
E20	11	11	6	11	6	9	6	6
E21	60	-	17	54	-	33	31	27
E22	9	9	19	8	8	22	16	16
E23	7	5	6	6	6	6	6	6
E24	-	31	40	79	12	7	6	6
E25	10	4	3	11	3	8	3	3
E26	8	6	3	13	28	5	5	5
E27	17	7	2	16	7	7	7	7
E28	18	7	4	15	4	6	4	6
E29	15	3	6	17	11	7	7	7
E30	17	4	7	14	4	4	4	4
E31	18	2	11	17	4	4	4	4
E32	17	6	4	20	6	8	6	6
E33	23	12	2	-	-	4	4	4
E34	40	5	8	9	2	2	2	2
E35	31	37	20	29	15	21	21	15
E36	11	8	18	11	10	20	11	10
E37	23	48	36	25	20	45	11	8
E38	-	2	4	-	13	2	2	2
Max.	60	53	40	79	35	45	31	39
Min.	5	2	1	3	2	2	2	2
Átlag	19	12	11	19	10	10	8	8
Medián	17	8	6	15	7	7	6	6

Az érték nélküli cellák azokat a pontokat jelölik, melyek nem képződtek le.

5. melléklet: Maradék ellentmondások az illesztőpontokban

Pontszám	Eltérések [mm]							
	Meshroom			WebODM				
	Közel	Közép	Távol	Közel	Közép	Távol	Közel + Közép	Közel + Távol
I0	-	9	11	-	5	5	5	5
I1	2	13	10	7	7	15	7	7
I2	10	-	18	5	5	14	5	5
I3	13	4	16	11	7	17	7	4
I4	-	7	-	-	5	-	5	-
I6	11	8	-	8	8	-	8	8
Min.	2	4	10	5	5	5	5	4
Max.	13	13	18	11	8	17	8	8
Átlag	9	8	14	8	6	13	6	6
Medián	11	8	14	8	6	15	6	5

Az érték nélküli cellák azokat a pontokat jelölik, melyek nem lettek felhasználva az adott transzformációhoz.