Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

TVORBA DIGITÁLNÍHO MODELU POVRCHU Z BODOVÝCH MRAČEN VYTVOŘENÝCH OBRAZOVOU KORELACÍ LETECKÝCH SNÍMKŮ

Diplomová práce

Bc. Jindřich HORÁK

Vedoucí práce RNDr. Jakub MIŘIJOVSKÝ, Ph.D.

Olomouc 2023
Geoinformatika a kartografie

ANOTACE

Diplomová práce se zabývá nalezením postupu tvorby digitálního modelu povrchu z bodových mračen vytvořených obrazovou korelací leteckých měřických snímků.

Řešení je založeno na postupném zpracování vstupních bodových mračen pomocí skriptů v programovacím jazyce Python. Bodová mračna byla opravena, byla stanovena pravidla pro spojení bodových mračen v překryvových pásech jednotlivých stereomodelů a bodová mračna byla ořezána na bloky Státní mapy 1 : 5 000, v jejichž území zpracování dále probíhalo. Jelikož obrazová korelace selhává v místech vodních ploch, byla tato místa opravena s využitím polygonové vrstvy 3D vodních těles. Z opravených bodových mračen byl vygenerován finální digitální model povrchu i digitální model povrchu ve formě stínovaného reliéfu.

Bodová mračna byla dále klasifikována do tří klasifikačních tříd (terén, budovy a vegetace) a v bodových mračnech byly též detekovány hrubé chyby, jejichž místa byla uložena do k tomu určené polygonové vrstvy.

Digitální model povrchu byl vytvořen též v programu Agisoft Metashape. V daném prostředí byl zpracován kompletní proces získání digitálního modelu povrchu z leteckých měřických snímků, tedy včetně samotné obrazové korelace.

Výsledkem práce je funkční postup tvorby digitálního modelu povrchu i samotný finální spojitý digitální model povrchu. Za výsledek lze považovat i konstatování, že i z leteckých měřických snímků lze produkovat digitální modely povrchu využitelné v mnoha aplikacích bez nutnosti provádění i leteckého laserového skenování, jehož výsledky jsou sice přesnější, ale provedení je mnohem nákladnější.

KLÍČOVÁ SLOVA

digitální model povrchu; bodové mračno; obrazová korelace; fotogrammetrie; letecké snímky

Počet stran práce: 55

Počet příloh: 19 (z toho 1 volná a 18 elektronických)

ANNOTATION

The master's thesis deals with finding a procedure for creating a digital surface model from point clouds created by image correlation of aerial imagery.

The approach is based on step-by-step processing of the input point clouds using scripts in the Python programming language. The point clouds were repaired, the rules for merging the point clouds in the overlapping zones of the respective stereomodels were established, and the point clouds were cropped to the State Map 1 : 5 000 map blocks, in whose area the processing was further carried out. Since the image correlation fails at the locations of water bodies, these locations were repaired using a polygon layer of 3D water bodies. From the repaired point clouds, a final digital surface model as well as a digital surface model in the form of shaded relief was generated.

The point clouds were further classified into three classification classes (ground, buildings, and vegetation) and gross errors were also detected in the point clouds and their locations were placed in a designated polygon layer for this purpose.

A digital surface model was also created in Agisoft Metashape. The complete process of obtaining the digital surface model from the aerial survey imagery, including the image correlation itself, was processed in this environment.

The result of the master's thesis is a functional procedure for the creating of the digital surface model as well as the final continuous digital surface model itself. It can also be considered as a result that even from aerial surveying images it is possible to produce digital surface models usable in many applications without the need to perform also airborne laser scanning, whose results are more accurate, but much more expensive to perform.

KEYWORDS

digital surface model, point cloud, image correlation, photogrammetry, aerial imagery

Number of pages: 55

Number of appendixes: 19

Prohlašuji, že

- diplomovou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.

- jsem si vědom, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. – autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,

- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3),

- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP,

- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona,

- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

babičce Marii, babičce Zdeňce,
dědovi Alfredovi, dědovi Karlovi

Děkuji vedoucímu práce RNDr. Jakubu Miřijovskému, Ph.D. za odborné vedení, cenné rady a připomínky týkající se řešení této práce a za možnost využití jeho výkonného PC pro zpracování výpočetně náročného úkonu.

Dále děkuji Mgr. Petru Dušánkovi, vedoucímu Oddělení správy dat a vývoje Zeměměřického úřadu, za velmi podnětné konzultace a návrhy řešení mnoha částí práce.

Poděkování patří též Zeměměřickému úřadu jako celku, především za zapůjčení programů pro zpracování práce.

Děkuji rovněž Ústavu pro hospodářskou úpravu lesů Brandýs nad Labem za poskytnutí dat pro tuto práci. Bez těchto dat by práce nevznikla.

Nakonec, ale rozhodně ne nejméně, bych chtěl poděkovat své rodině za obrovskou a neustálou podporu v celém studiu.

UNIVERZITA PALACKÉHO V OLOMOUCI

Přírodovědecká fakulta

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jindřich HORÁK**
Osobní číslo:	**R21850**
Studijní program:	**N0532A330009 Geoinformatika a kartografie**
Téma práce:	**Tvorba digitálního modelu povrchu z bodových mračen vytvořených obrazovou korelací leteckých snímků**
Zadávající katedra:	**Katedra geoinformatiky**

Zásady pro vypracování

Cílem práce je nalezení postupů pro vytvoření spojitého digitálního modelu povrchu z bodových mračen vytvořených obrazovou korelací leteckých měřických snímků. Student se bude zabývat zejména:
a) nalezením nástrojů pro automatickou detekci hrubých chyb v bodových mračnech,
b) stanovením pravidel pro spojení bodových mračen v překryvových pásech jednotlivých stereomodelů,
c) stanovením pravidel pro automatickou klasifikaci mračna bodů do minimálně tří klasifikačních tříd (terén, vegetace, stavby),
d) stanovením postupů pro opravu digitálního modelu povrchu v místech, kde obrazová korelace selhává (především na vodních plochách).
Student současně otestuje metody obrazové korelace v programu Agisoft Metashape a porovná získané bodové mračno s bodovým mračnem, které získal od pořizovatele dat.
Pro zpracování DP budou využita data z obrazové korelace vytvořená Úřadem pro hospodářskou úpravu lesů, jako pomocná data budou využity DMR 5G a DMP 1G.
Text práce s vybranými přílohami bude odevzdán ve dvou svázaných výtiscích na sekretariát katedry. O diplomové práci student vytvoří webovou stránku v souladu s pravidly dostupnými na stránkách katedry. Práce bude zpracována podle zásad dle Voženílek (2002) a závazné šablony pro diplomové práce na KGI. Povinnou přílohou práce bude poster formátu A2.

Rozsah pracovní zprávy:	**max. 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

Hauf, M. et al. (1989): Geodézie, Praha
Rapant, P.: Družicové polohové systémy. VŠB-TU Ostrava, 2002. 200 str.
LILLESAND, T., KIEFER, R., CHIPMAN, J. Remote Sensing and Image Interpretation. John Wiley & Sons, Inc., New York, 756 s., 2008.
CAMPBELL, J., B., WYNNE, R. H. Introduction to Remote Sensing. 5th ed. Guilford Press, New York, 667 s., 2011.
Pavelka, K. (2002). Fotogrammetrie 10, 2. přeprac. vyd. Praha: FSv ČVUT, 198 s.
Manuály programů Trimble Inpho a Agisoft
Voženílek, V. (2002): Diplomové práce z geoinformatiky. Vydavatelství Univerzity Palackého, Olomouc, UP, 61 s.

Vedoucí diplomové práce: **RNDr. Jakub Miřijovský, Ph.D.**
Katedra geoinformatiky

Datum zadání diplomové práce: **9. prosince 2021**
Termín odevzdání diplomové práce: **5. května 2023**

UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA GEOINFORMATIKY
17. listopadu 50, 771 46 Olomouc

L.S.

doc. RNDr. Martin Kubala, Ph.D.
děkan

prof. RNDr. Vít Voženílek, CSc.
vedoucí katedry

V Olomouci dne 16. prosince 2021

OBSAH

SEZNAM POUŽITÝCH ZKRATEK

Zkratka	Význam
3D	trojdimenzionální, trojrozměrný
ASCII	American Standard Code for Information Interchange
ASPRS	American Society for Photogrammetry and Remote Sensing
Bpv	Balt po vyrovnání
CIR	Color InfraRed
ČSÚ	Český statistický úřad
ČÚZK	Český úřad zeměměřický a katastrální
DMP	digitální model povrchu
DMR	digitální model reliéfu
DMR 5G	Digitální model reliéfu 5. generace
DPZ	dálkový průzkum Země
DSM	Digital Surface Model
EPSG	EPSG Geodetic Parameter Dataset
EVLRs	Extended Variable Length Records
GDB	ArcGIS geodatabáze
GIS	geografický informační systém
GPS	Global Positioning System
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
ISPRS	International Society for Photogrammetry and Remote Sensing
LiDAR	Light Detection and Ranging
LLS	letecké laserové skenování
LMS	letecký měřický snímek/letecké měřické snímkování
MBR	Minimum Bounding Rectangle
OGC	Open Geospatial Consortium
px	pixel
RAD	Rapid Application Development
RGB	Red-Green-Blue
RÚIAN	Registr územní identifikace, adres a nemovitostí
SGM	Semi-Global Matching
SHP	shapefile
S-JTSK	systém Jednotné trigonometrické sítě katastrální
SM5	Státní mapa 1 : 5 000
SM50	Státní mapa 1 : 50 000
TIN	Triangulated Irregular Network
UAV	Unmanned Aerial Vehicle
ÚHÚL	Ústav pro hospodářskou úpravu lesů Brandýs nad Labem

Zkratka	**Význam**
UTM	Universal Transverse Mercator
VLRs	Variable Length Records
WGS84	World Geodetic System 1984
WMS	Web Map Service
ZABAGED®	Základní báze geografických dat České republiky
ZM ČR	Základní mapa České republiky
ZTM 5	Základní topografická mapa 1 : 5 000
ZÚ	Zeměměřický úřad

ÚVOD

Vedle široce využívaných digitálních modelů reliéfu nachází své využití rovněž i digitální modely povrchu. U digitálního modelu povrchu je průběh georeliéfu doplněn o všechny přírodní i umělé prvky, které jsou s terénem spojeny – stromy, vegetace, budovy a jiné stavební objekty, a proto představuje věrnou kopii území (G4D, 2023). Díky tomu je využíván například při 3D modelování měst a krajiny pro účely urbanismu a rozvoje, při mapování zemních prací nebo archeologických nalezišť (G4D, 2023), při analýzách viditelnosti, modelování šíření radiových vln, modelování šíření škodlivých látek a nečistot v ovzduší nebo při generování virtuálních pohledů na terén v leteckých simulátorech a trenažérech (ČÚZK, 2022).

Digitální model povrchu může vznikat metodou leteckého laserového skenování, či právě metodou obrazové korelace leteckých snímků. Oba tyto přístupy mají své výhody i nevýhody. Rozvoj digitální fotogrammetrie dnes dovoluje získávat bodová mračna za velmi dobrých ekonomických podmínek, ale výsledek tohoto přístupu je velmi závislý na mnoha faktorech – správně naplánovaný a provedený snímkový let, správné umístění vlícovacích bodů, prostorové rozlišení snímků, vhodné algoritmy obrazové korelace a hustota výsledného bodového mračna. Oproti tomu metoda leteckého laserového skenování je velmi nákladná, ovšem jejím výsledkem je přímo bodové mračno (Braun a kol., 2021).

Na Zeměměřickém úřadu tedy vyvstala motivace zjistit, zda by bylo možné generovat digitální modely povrchu pouze metodou obrazové korelace z leteckých měřických snímků, které již jsou pořizovány pro tvorbu ortofota, bez nutnosti provádění nákladnějšího leteckého laserového skenování.

1 CÍLE PRÁCE

Cílem diplomové práce je nalezení postupů pro vytvoření spojitého digitálního modelu povrchu (DMP) z bodových mračen vytvořených obrazovou korelací leteckých měřických snímků (LMS).

V rámci řešení hlavního cíle práce bude rozpracováno několik podcílů. Nejprve bude sestavena odborná rešerše již existujících postupů jiných autorů a zároveň bude zpracována teorie, z jakých principů celý postup tvorby DMP zmíněným způsobem vychází.

V praktické části bude řešeno hledání nástrojů pro automatickou detekci hrubých chyb v bodových mračnech, budou stanovena pravidla pro spojení bodových mračen v překryvových pásech jednotlivých stereomodelů, dále též pravidla pro automatickou klasifikaci mračna bodů do alespoň tří klasifikačních tříd (terén, vegetace a stavby) a rovněž bude stanoven postup pro opravu výsledného DMP v místech, kde obrazová korelace selhává (především na vodních plochách).

Hlavní cíl i všechny jeho podcíle budou zpracovány na datech z obrazové korelace (bodových mračnech) poskytnutých Ústavem pro hospodářskou úpravu lesů Brandýs nad Labem (ÚHÚL) ve spolupráci se Zeměměřickým úřadem (ZÚ).

Současně budou otestovány metody obrazové korelace v programu Agisoft Metashape. Bodové mračno získané z programu Agisoft Metashape mělo být původně porovnáno s mračnem od poskytovatele dat pro práci – ÚHÚL. ÚHÚL ovšem pro práci poskytl bodová mračna silně zředěná, nikoli originální z obrazové korelace. Porovnání bodových mračen s úplně rozdílnou hustotou bodů by nedávalo smysl, nebylo tedy proto provedeno.

Výsledky práce umožní Zeměměřickému úřadu zpracovávat DMP pouze z LMS, které již jsou každoročně pořizovány na polovině České republiky pro tvorbu barevně vyrovnaného zdánlivě bezešvého georeferencovaného ortofotografického zobrazení zemského povrchu. Pro tvorbu DMP tedy odpadne nutnost provádění nákladnějšího leteckého laserového skenování (LLS), které bude moci být uskutečňováno pouze tam, kde je to z hlediska přesnosti a podstaty LLS bezpodmínečně nutné, a to obzvláště na menších územích.

2 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Tato kapitola je orientována na popis a rozvinutí teorie a používaných metod ohledně řešené problematiky a shrnuje též některé práce, které se zabývaly řešenou problematikou či příbuznými tematikami.

2.1 Obrazová korelace

Technologie automatického digitálního zpracování fotogrammetrických snímků využívá princip obrazové korelace dvou subobrazů. Vždy je cílem automatizovaně zjistit polohu dvou odpovídajících si bodů a na základě známých prvků vnitřní i vnější orientace určit výšku zkoumaného bodu nad srovnávací rovinou (Pavelka, 2003).

Pokud by se vybranému bodu na jednom obraze hledal odpovídající bod na druhém obraze jen v rámci jediného pixelu, bylo by nalezeno mnoho (v řádech statisíců) stejných pixelů. Vychází se z předpokladu, že každý bod má své unikátní okolí, pomocí kterého jej lze identifikovat i na druhém obraze. Čím větší je stanovené okolí, tím pravděpodobnější je i nalezení správného odpovídajícího bodu na druhém obraze, ale zároveň i tím vyšší jsou výpočetní nároky. Jestliže nebude okolí bodu dostatečně unikátní (vodní hladina, zasněžená plocha), může obrazová korelace selhávat (Pavelka, 2003).

Výška zkoumaného bodu nad srovnávací hladinou neboli z pohledu kamery hloubka bodu může být spočítána ze dvou odpovídajících si bodů pomocí triangulace. Hloubku v libovolném bodě lze vypočítat, pokud je známa disparita (rozdíl či podle Pavelky (2003) posunutí; angl. disparity) v tomto bodě. Disparita měří posunutí bodu mezi dvěma obrazy, čím vyšší je disparita, tím blíže kameře se daný bod nachází (MathWorks, 2023).

Algoritmy pro odhad disparity jsou rozdělovány do dvou kategorií: lokální metody a globální metody. Lokálními metodami je vyhodnocován jeden pixel po druhém a jsou brány v úvahu vždy pouze okolní pixely vybraného pixelu. Globálními metodami jsou vyhodnocovány informace z celého obrazu. Lokálními metodami jsou poměrně špatně detekovány náhlé změny hloubky a tzv. okluze (zakryté či zastíněné části obrazu, které negativně ovlivňují výsledek obrazové korelace; angl. occlusions), a proto jsou upřednostňovány globální metody (MathWorks, 2023).

Zcela speciální kategorií jsou pologlobální metody (angl. Semi-Global Matching (SGM)) využívající k výpočtu disparity bodu informace ze sousedních pixelů ve více směrech (MathWorks, 2023). S konceptem SGM poprvé přišel Hirschmüller v roce 2005 (Hirschmüller, 2005) a v roce 2011 jej popisuje jako metodu úspěšně kombinující lokální a globální stereo metody pro rychlé a přesné zjišťování disparity bodu. Hlavní myšlenkou SGM je výpočet „nákladů“ na dosažení pixelu s určitou disparitou podél více směrů (nejčastěji osm směrů – od krajů obrazu k danému bodu). Pro každý pixel a každou disparitu se náklady ze všech osmi směrů sčítají, poté je každému pixelu vybrána disparita s nejnižšími náklady. Výsledky metody SGM nejsou ani příliš ovlivňovány radiometrickými rozdíly (efekt vinětace, rozdílné časy expozice, změna polohy zdroje světla apod.) mezi snímky tvořícími stereodvojici. Hlavní síla SGM může být demonstrována při tvorbě DMP z leteckých snímků v městských oblastech. Zde byly vytvořeny velmi přesné DMP, zejména na souborech dat s dostatečným překryvem snímků a byla rovněž potvrzena konkurenceschopnost SGM ve srovnání s metodami LLS (Hirschmüller, 2011).

Srovnání na standardních stereosnímcích ukazuje, že SGM patří v současnosti mezi nejlépe hodnocené algoritmy a je nejlepší, pokud se bere v úvahu subpixelová přesnost (Hirschmüller, 2008).

2.2 Bodové mračno

Bodové mračno je soubor bodů v prostoru, které představují nejčastěji povrch objektu a které lze použít k mnoha různým účelům. Může být též popsáno jako digitální reprezentace objektu či scény, které jsou složeny z velkého množství bodů v trojrozměrném prostoru. Každý bod většinou obsahuje informace o geometrických, barevných či případně i jiných vlastnostech (např. klasifikace). Bodová mračna jsou vytvářena různými technologiemi, nejčastěji technologií laserového skenování – Light Detection and Ranging (LiDAR) či fotogrammetrickými metodami – jako například obrazovou korelací. Při LiDAR je využíváno laserových pulzů pro měření vzdáleností mezi snímačem a skenovaným objektem. V dálkovém průzkumu Země (DPZ) je pomocí LiDAR měřena vzdálenost mezi rovinou senzoru a body na zemském povrchu v pravidelných časových intervalech a tato měření jsou vztahována k prostorovým souřadnicím. Každé jedno měření pak tvoří jeden bod v bodovém mračnu. Ve fotogrammetrii jsou využívány snímky objektu pořízené z různých úhlů k výpočtu jeho rozměrů a vytvoření bodového mračna pomocí triangulace, konkrétně metodou obrazové korelace (Puzzo, 2021).

Obě výše zmíněné techniky tvorby bodových mračen mají své výhody i nevýhody. Vytvoření 3D modelu (bodového mračna) fotogrammetrickými metodami, tedy zpracování velkého množství snímků fotogrammetrickými programy často trvá velmi dlouho, zatímco technologií LiDAR jsou data zachycena během několika sekund a zpracována jsou téměř stejně rychle. Kvalita výsledného bodového mračna je u fotogrammetrie velmi ovlivněna počtem a kvalitou vstupních snímků. Nízký počet snímků či jejich špatná kvalita zapříčiní špatnou kvalitu bodového mračna. U technologie LiDAR je téměř vždy zajištěna vysoká přesnost výsledného mračna, neboť každý jeden odraz tvoří prostorově určený bod bodového mračna. Za největší rozdíl mezi oběma technikami lze považovat barvu. Mračna bodů vytvořená pomocí fotogrammetrie mají pro každý bod hodnoty RGB či jiných barevných kanálů, které obsahují vstupní snímky (Puzzo, 2021).

Bodová mračna mohou nalézt využití v mnoha spektrech lidské činnosti, všude, kde lze používat digitální reprezentace reálného prostředí. Zeměměřictvím, geoinformatikou a obecně geografií počínaje (tvorba digitálních modelů reliéfu a povrchu, skenování různých prvků infrastruktury, přírodních útvarů, silnic apod.), přes životní prostředí (např. sledování pohybu a změny v čase erodujícího svahu) či stavebnictví (využití modelů stávajících staveb pro plánování rekonstrukcí nebo kontrola zhoršování stavu budovy v průběhu času) až po historii a archeologii (vytváření modelů historických památek či archeologických nalezišť) nebo též při automatizovaném zpracování obrazu (vývoj samořiditelných automobilů, dále také v lékařství) (Puzzo, 2021).

2.3 Formát LAS

Soubory ve formátu LAS jsou určeny pro záznam mračen bodů primárně z LLS technologií LiDAR. Data jsou do tohoto formátu vkládána pomocí programového vybavení, které kombinuje údaje GPS (Global Positioning System), IMU (Inertial Measurement Unit) a údaje o echách laserových pulzů, čímž jsou vytvářeny body o souřadnicích X, Y a Z (ASPRS, 2013). Formát ovšem kromě dat z LLS podporuje výměnu jakýchkoli trojrozměrných dat se souřadnicemi X, Y a Z (Library of Congress, 2022).

Prvotním záměrem vydání formátu LAS bylo poskytnout otevřený datový formát, který umožní různým hardwarovým a softwarovým nástrojům pracujícím s LiDAR daty poskytovat výstupy ve stejném a vzájemně srozumitelném formátu (ASPRS, 2013).

Specifikace LAS byla vyvinuta a je nadále udržována Americkou společností pro fotogrammetrii a dálkový průzkum Země (ASPRS – American Society

for Photogrammetry and Remote Sensing) (Library of Congress, 2022) a od roku 2017 je tento formát veden jako tzv. OGC Community Standard mezinárodní standardizační organizací pro geoprostorová data Open Geospatial Consortium (OGC) (OGC, 2018).

Formát byl ASPRS vyvinut jako binární alternativa k proprietárním systémům a systému výměny souborů ve formátu ASCII (American Standard Code for Information Interchange). Problém proprietárních formátů je zřejmý v tom, že je nelze používat napříč různými systémy. Dále čtení a interpretace formátu ASCII může být velmi pomalé a velikost souboru může být extrémně velká, a to i pro malé množství dat. Zároveň tento formát nezachovává specifické informace pro LiDAR data. Oproti tomu je formát LAS specifický pro povahu LiDAR dat a je mnohem jednodušší, jelikož jsou v něm čísla uložena v binární podobě, čímž je šetřeno místo na disku a je eliminován čas potřebný ke zpracování, který je u ASCII nutný k převodu do podoby použitelné pro výpočty (Library of Congress, 2022).

Formát obsahuje binární data tvořící čtyři různé skupiny. První skupinou je *Public Header Block*, tedy hlavička obsahující obecná data, jako jsou identifikační čísla jednotlivých bodů a hranice bodových dat. Druhá skupina je nazvána *Variable Length Records (VLRs)*, tedy údaje, které mohou obsahovat informace o projekci dat, metadata či data o uživatelské aplikaci. Tato skupina není povinná a je omezená na 65 535 bajtů. Třetí skupinou jsou *Point Data Records*, což jsou samotná nejdůležitější data o každém bodu (nejčastěji o jeho laserovém nasnímání). Pro každý bod jsou zde evidovány atributy jako souřadnice X, Y a Z, intenzita zpětného laserového pulzu, číslo echa (při vícenásobných echách vznikajících při laserovém skenování), celkový počet ech pro daný pulz, informace o klasifikaci bodu do kategorií atd. Poslední skupinou jsou *Extended Variable Length Records (EVLRs)*, neboli údaje umožňující uložit více informací než VLRs a s možností připojení až na konec LAS souboru. To je výhodou, neboť může být jejich prostřednictvím přidána do LAS souboru například informace o projekci bez nutnosti přepisování celého souboru (ASPRS, 2013).

2.4 Programovací jazyk Python

Programovací jazyk Python se řadí mezi interpretované (bez kompilování spustitelné), vysokoúrovňové (s vyšší mírou abstrakce – kód je podobnější myšlení člověka; angl. high-level), univerzální (s širokou škálou oblastí použití – napříč aplikačními doménami; angl. general-purpose), objektově orientované programovací jazyky s dynamickou sémantikou. Implementované datové struktury na vysoké úrovni v kombinaci s dynamickým typováním a dynamickými vazbami jej činí velmi atraktivním pro tzv. Rapid Application Development (RAD) (jedna z agilních metod vývoje aplikací) a také pro použití jako skriptovací nebo spojovací jazyk (angl. glue language) pro propojení existujících komponent (Van Rossum, 2023).

Jednoduchá a snadno naučitelná syntaxe jazyka Python klade důraz na čitelnost, a tím snižuje náklady na údržbu programu. Python podporuje moduly a balíčky, což podporuje modularitu programu a opakované použití kódu. Interpret jazyka Python a rozsáhlá standardní knihovna (angl. standard library) jsou k dispozici zdarma pro všechny hlavní platformy a lze je volně šířit (Van Rossum, 2023).

Python je také použitelný jako rozšiřující jazyk pro aplikace, které potřebují programovatelné rozhraní (Python, 2023). Takovým případem je například knihovna ArcPy, tedy knihovna spojená s geografickými informačními systémy (GIS) od společnosti Esri poskytující užitečný a produktivní způsob, jak provádět analýzu geografických dat, konverzi dat, správu dat a automatizaci map pomocí jazyka Python (Esri, 2023f).

2.5 Práce zabývající se řešenou problematikou

Assessment of the Automatic Digital Surface Model from Digital Aerial Camera and from Lidar Point Clouds Data (Elshehaby a Taha, 2018)

Elshehaby a Taha (2018) se ve svém článku *Assessment of the Automatic Digital Surface Model from Digital Aerial Camera and from Lidar Point Clouds Data* zaměřili na porovnání dvou různých metod získávání digitálních modelů povrchu (DMP). Autory byly srovnány DMP získané z digitálních leteckých kamer a z technologie LiDAR a byla analyzována jejich přesnost a vhodnost pro různé účely.

Autoři zmínili, že s nástupem digitálních leteckých kamer schopných pořizovat stereo snímky o velmi vysokém rozlišení započala nová éra možností produkování digitálních modelů povrchu. Dříve bylo k tomuto účelu používáno pouze letecké laserové skenování (LLS), tedy technologie LiDAR. Proto byly autory pro účely tohoto článku vytvořeny DMP oběma metodami (včetně popisu postupu tvorby), tyto DMP byly porovnány a byla vyhodnocena jejich přesnost.

Pro tvorbu DMP byl použit ukázkový dataset zahrnující oblast centra města Toronto v Kanadě, který byl autorům poskytnut Mezinárodní společností pro fotogrammetrii a dálkový průzkum Země (ISPRS – International Society for Photogrammetry and Remote Sensing). Letecké snímky v něm byly pořízeny kamerou Microsoft Vexcel's UltraCam-D a data LiDAR byla získána leteckým laserovým skenerem Optech ALTM Orion M.

Použité letecké snímky byly pouze dva, a to s podélným překryvem 60 % a byly pořízeny z výšky 1 600 m nad zemí. Ke snímkům byly poskytnuty též prvky vnitřní i vnější orientace. Laserový paprsek skeneru Optech má vlnovou délku 1 064 nm (blízké infračervené spektrum), skenuje zemský povrch v úhlu 20 ° a jeho frekvence je 50 Hz. Letadlo nesoucí zmíněný skener letělo při pořizování dat rychlostí 120 uzlů a ve výšce 650 m nad povrchem. Výsledné bodové mračno obsahuje šest pásů bodů a hustota bodů je přibližně šest bodů na jeden metr čtvereční. Mračno je poskytnuto ve formátu LAS od společnosti ASPRS a ve stejném souřadnicovém systému jako letecké snímky z kamery UltraCam-D.

Vytváření DMP z leteckých snímků proběhlo v modulu Leica Photogrammetric Suite programu Erdas Imagine 2014. Po nahrání snímků do programu a nastavení souřadnicového systému WGS84 a zóny UTM17 byly spočítány pyramidy ke zkrácení doby výpočtu. Dále byly programu poskytnuty prvky vnitřní i vnější orientace snímků a byly automatizovaně nalezeny spojovací body. Především na základě zjištěných spojovacích bodů byl v programu metodou obrazové korelace (image-matching) vytvořen výsledný DMP.

LiDAR data jako bodové mračno byla zpracována v programu LAStools od společnosti rapidlasso. Pro tvorbu DMP byla nastavena interpolace z bodového mračna do buněk rastru o velikosti 3 × 3 m a pro interpolaci byly použity pouze body vzniklé z prvního echa každého jednoho laserového pulzu.

Porovnání výsledných produktů proběhlo pomocí srovnání výšek (souřadnice Z) v obou DMP v kontrolních bodech, které byly získány z automatického generování spojovacích bodů. Výška DMP vytvořeného z laserových dat byla brána jako referenční. Výsledky srovnání výšek z obou produktů ukázaly, že průměrná vertikální odchylka DMP z leteckých snímků je 0,28 m.

Autoři závěrem zmiňují, že DMP z dat LiDAR je stále přesnější než DMP vzniklé fotogrammetrickými metodami, především co se týče absolutní výškové přesnosti. Na druhou stranu ale také vyzdvihují, že bodové mračno vzniklé z leteckých snímků je mračnem s velmi vysokou hustotou bodů ve srovnání s bodovým mračnem z leteckého laserového skenování.

Generating of high-resolution Digital Surface Models for Urban Flood Modelling using UAV Imagery (Yalcin, 2018)

Yalcin se ve svém článku *Generating of high-resolution Digital Surface Models for Urban Flood Modelling using UAV Imagery* zabýval vytvořením DMP o vysokém rozlišení ze snímků pořízených UAV systémem (bezpilotní letadla – drony; angl. Unmanned Aerial Vehicle) pro použití v analýzách hydrodynamických modelů ke zjišťování efektů možných povodní v centru města Kırşehir v Turecku. Zmiňuje, že jeden z nejdůležitějších zdrojů nejistoty v hydrodynamických modelech je rozlišení použitého DMP. Změna rozlišení DMP může způsobovat výrazné rozdíly ve výsledcích simulací povodní, ať už se jedná o rozsah povodně, hloubku vody či rychlost proudění. Proto jsou v komplexních městských oblastech pro přesnější simulace pohybu vody nezbytná topografická data s vysokým rozlišením, aby mohla být v případě skutečné povodně přijata správná a včasná opatření ke snížení následků.

Použití UAV systémů je vhodným řešením v případě, kdy nemůže být použito LLS nebo snímky z družic z důvodu ceny, času či potřeby rozlišení. Autorem byl použit dron DJI Matrice 600 vybavený kamerou DJI Zenmuse X5.

Celý proces tvorby byl rozdělen na tři hlavní fáze: Zaměření a vyznačení vlícovacích bodů v terénu, pořízení leteckých snímků s použitím dronu, zpracování pořízených snímků.

V první fázi autor v zájmové oblasti zaměřil 30 vlícovacích bodů a tři kontrolní body metodou RTK GPS zařízením Leica Viva GNSS GS15 s využitím referenčních stanic sítě CORS-TR. Všechny tyto body označil červeným sprejem.

Zájmová oblast byla dronem automatizovaně nasnímána podle předem připraveného letového plánu z programu UgCS. Letový plán obsahoval určená místa startů dronu (celkem šest míst), překryvy snímků (60 % podélný, 40 % příčný), letovou výšku (120 m), rychlost (5 m/s) a cílové prostorové rozlišení.

Zpracování dat proběhlo v programu Pix4Dmapper Pro. Prvním krokem bylo počáteční zpracování spočívající ve vyhledání spojovacích bodů na překrývajících se snímcích pomocí algoritmu Structure from Motion (SfM). V druhém kroku byly do programu nahrány pozice vlícovacích a kontrolních bodů a v integrovaném editoru rayCloud byla na snímcích identifikována jejich přesná umístění. Následně byly díky vlícovacím bodům zpřesněny prvky vnitřní orientace svazkovým vyrovnáním bloku a tím mohla být korigována poloha spojovacích bodů vyhledaných na snímcích metodou automatické aerotriangulace (AAT) – zpřesnění prvků vnější orientace. Po těchto krocích bylo vygenerováno řídké bodové mračno a následně s vhodným nastavením i husté bodové mračno. Body hustého bodového mračna byly triangulovány a vznikla tzv. 3D texturovaná mesh. V posledním kroku byl vygenerován samotný DMP za použití metody vážených inverzních vzdáleností (IDW) s filtrováním šumu a vyhlazováním ostrých povrchů. Na závěr bylo vytvořeno i ortofoto zájmové oblasti.

Výsledné husté bodové mračno má průměrnou hustotu 116 bodů/m^2 a prostorové rozlišení DMP je 4,32 cm/px. Autor konstatuje, že toto rozlišení je dostatečné k využití v analýzách hydrodynamických modelů a že vzniklé ortofoto o vysokém rozlišení lze použít při přípravě map povodňového nebezpečí a povodňových rizik k jasnému určení dopadů možných povodní.

S využitím tří kontrolních bodů byla vyhodnocena chybovost, která byla vyjádřena pomocí RMSE. Průměrná hodnota RMSE z dílčích hodnot pro každou souřadnici je 8,5 cm.

Závěrem autor shrnuje, že dle výsledků je možné tvořit DMP s vysokým rozlišením pro povodňové modely i s relativně nízkými náklady a v krátkém čase pomocí snímků z UAV systémů.

Point Cloud and Digital Surface Model Generation from high resolution Multiple View Stereo Satellite Imagery (Gong a Fritsch, 2018)

Motivace autorů ke zpracování článku s názvem *Point Cloud and Digital Surface Model Generation from high resolution Multiple View Stereo Satellite Imagery* vznikla díky zveřejnění testovacích Multiple View Stereo (MVS) dat z komerční družice WorldView-3 ze strany *John Hopkins University Applied Physics Laboratory, USA*. Tato zveřejněná data motivovala autory ke zkoumání metody, která dokáže generovat přesné digitální modely povrchu z velkého množství družicových snímků s vysokým rozlišením.

Autoři uvádí, že v poslední době prošly různé druhy senzorů umístěné na družicích neskutečným vývojem a že se startem družice WorldView-3 se prostorové rozlišení dostává až ke 30 cm/px. Družice často přelétající nad určitou lokalitou poskytují velké množství snímků dané lokality a tím umožňují pořizování Multiple View Stereo (MVS) dat. Přestože se povrch terénu může měnit v průběhu ročních období, jsou družicové MVS snímky pořízené v různých termínech vhodným zdrojem dat pro tvorbu DMP rozsáhlých území. Díky velkému pokrytí a submetrovému prostorovému rozlišení jsou družicové MVS snímky cenné pro globální 3D mapování, monitorování životního prostředí, městské plánování, detekce změn terénu apod.

Zveřejněná data, na kterých autoři svou studii provedli, zahrnují 50 panchromatických a multispektrálních snímků z družice WorldView-3, které zobrazují území velké asi 100 km^2 nedaleko San Fernando v Argentině. Pro nadirové snímky bylo prostorové rozlišení přibližně zmíněných 30 cm/px. Snímky byly pořízeny od listopadu 2014 do ledna 2016. Jako referenční data bylo rovněž zveřejněno mračno bodů pořízené LiDAR technologií a z něj vytvořený DMP a prostorovém rozlišení též 30 cm/px.

Autoři vygenerovali samostatně DMP pro každou zvolenou stereodvojici snímků a všechny vygenerované DMP nakonec spojili do finálního výsledku. Postup práce byl rozdělen na čtyři kroky: výběr snímků, relativní orientace a rektifikace snímků, tvorba hustého bodového mračna a triangulace, slučování jednotlivých DMP.

Při výběru snímků jsou nejdůležitější následující faktory, které mohou nejvíce ovlivnit kvalitu generovaného DMP: datum pořízení každého snímku ze stereodvojice, úhel dopadu a tzv. úhel protnutí dvojice snímků. Ohledně data pořízení bylo zjištěno, že kvalitnější DMP je ze snímků ze stejného ročního období, i když se liší rokem, než ze stejného roku ale jeden snímek mimo vegetační období a jeden z vegetačního období. Z výběru snímků pro utvoření párů byly eliminovány snímky s vyšším úhlem dopadu než 35 ° a ponechány byly snímky s úhlem protnutí mezi 5 a 40 °.

Družicové senzory s vysokým rozlišením neposkytují klasické prvky vnitřní a vnější orientace, namísto toho poskytují tzv. Rational Polynomial Coefficients (RPCs), které jsou matematickým vztahem mezi snímkovými souřadnicemi a souřadnicemi objektu. K orientaci snímků s RPCs se využívá tzv. bias-compensated RPCs bundle block adjustment (volně přeloženo jako svazkové vyrovnání bloku s kompenzací zkreslení). Jelikož tato metoda vyžaduje kromě spojovacích bodů také vlícovací body, které ale často nejsou pro družicová data k dispozici, je pro MVS snímky lepší relativní orientace. Proto bylo autory k orientaci MVS snímků využito relativní svazkové vyrovnání bloku s kompenzací zkreslení, v jehož průběhu byly vytvořeny virtuální vlícovací body a virtuální povrch terénu, který lze převést do skutečného.

K vytvoření hustého bodového mračna byla využita modifikovaná metoda SGM zvaná tSGM, která je implementována v knihovně libTsgm programovacího jazyka C++. Metodou tSGM byla vytvořena mapa disparity pro každý stereopár, z níž byly odvozeny odpovídající si pixely na stereosnímcích. Husté bodové mračno bylo vytvořeno protínáním vpřed odpovídajících si pixelů.

Při slučování jednotlivých DMP nebyla pro bodová mračna nutná žádná další tzv. registrace, neboť již byla vyrovnána na zmíněném virtuálním povrchu terénu. Bodová mračna byla promítnuta do pravidelné diskrétní sítě v souřadnicovém systému UTM a pro sloučení byl využit jednoduchý mediánový filtr. Jako konečná výška sloučeného DMP je vybrána mediánová výška každé buňky sítě.

Závěrem autoři konstatují, že po porovnání s poskytnutými referenčními daty je přesnost jejich DMP obecně dobrá. Mediánová chyba je půl metru a RMSE je 2,5 m. RMSE je velká v porovnání s prostorovým rozlišením družice WorldView-3, autoři se domnívají, že chyby mohly být způsobeny hustou zástavbou v zájmovém území. Přesto se autoři i na základě jiných ukazatelů odvažují tvrdit, že výsledek je robustní a většina bodů má jen malý rozdíl oproti referenčním datům.

Katedra geoinformatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci

Na zmíněném pracovišti se příbuzným tématem – zejména využitím UAV systémů pro fotogrammetrický sběr a zpracování geodat – zabýval především RNDr. Jakub Miřijovský, Ph.D. ve své disertační práci s názvem *Fotogrammetrický přístup při sběru geodat pomocí bezpilotních leteckých zařízení* (Miřijovský, 2013b) a z ní vycházející knihy *Bezpilotní systémy: sběr dat a využití ve fotogrammetrii* (Miřijovský, 2013a).

3 METODY A POSTUP ZPRACOVÁNÍ

Po vyhotovení finální verze zadání práce vedoucím práce byla nejdříve provedena rešerše daného okruhu – dané problematiky. Bylo zpracováno několik odborných článků zahraničních autorů, jejichž cílem bylo rovněž vytvoření digitálních modelů povrchu fotogrammetrickými metodami. Někteří jako zdroj použili letecké snímky (jako v této práci), v dalším článku byly použity snímky z družice, a ještě v jiném článku byly zpracovány snímky z UAV systémů.

Jakmile byla kladně vyřízena žádost o poskytnutí vstupních dat (zředěných bodových mračen) ze strany Ústavu pro hospodářskou úpravu lesů Brandýs nad Labem (ÚHÚL), byly s Mgr. Petrem Dušánkem, vedoucím Oddělení správy dat a vývoje Zeměměřického úřadu, prodiskutovány nápady a návrhy možných postupů řešení dílčích cílů i hlavního cíle práce.

Ihned poté byly diskutované nápady a návrhy rozpracovány autorem práce a byly jedna po druhé převáděny do praxe. Nejprve byly metody testovány na minimálním vzorku dat „ručním“ (ne naprogramovaným) zpracováním. Po ověření jejich funkčnosti a získání výsledků blízkých očekávání bylo použití těchto metod převedeno do programového kódu (do několika skriptů) v programovacím jazyce Python.

Sestavené skripty byly poté postupně spouštěny, nyní již na všechna data. Každý skript provedl na datech jednu nebo více operací. Jedním z posledních skriptů byl generován primární výstup práce – opravený digitální model povrchu. Dalšími výstupy jsou i některé samotné skripty či jejich součásti – na opravu DMP, na klasifikaci bodového mračna a na detekci hrubých chyb (děr primárně nezpůsobených vodou) v bodovém mračnu. Tyto výstupy jsou zároveň výsledky práce.

Dále byl využit program Agisoft Metashape, ve kterém byl obrazovou korelací vygenerován DMP z LMS poskytnutých od ZÚ pro zjištění, nakolik je tento program způsobilý generovat DMP v dostatečné kvalitě.

Na závěr byl dokončen text práce, sestaveny byly webové stránky a zcela nakonec i poster.

Použité metody

Po provedení rešerše současného stavu problematiky na začátku řešení práce byly zpracovány návrhy možných postupů řešení, tedy obecně teoretická metoda analýzy celého problému tvorby DMP z bodových mračen. Tento problém byl rozebrán na dílčí kroky, jejichž postupné provedení vedlo ke vzniku výsledného DMP a souvisejících prací. Dílčí metody jsou nastíněny zde:

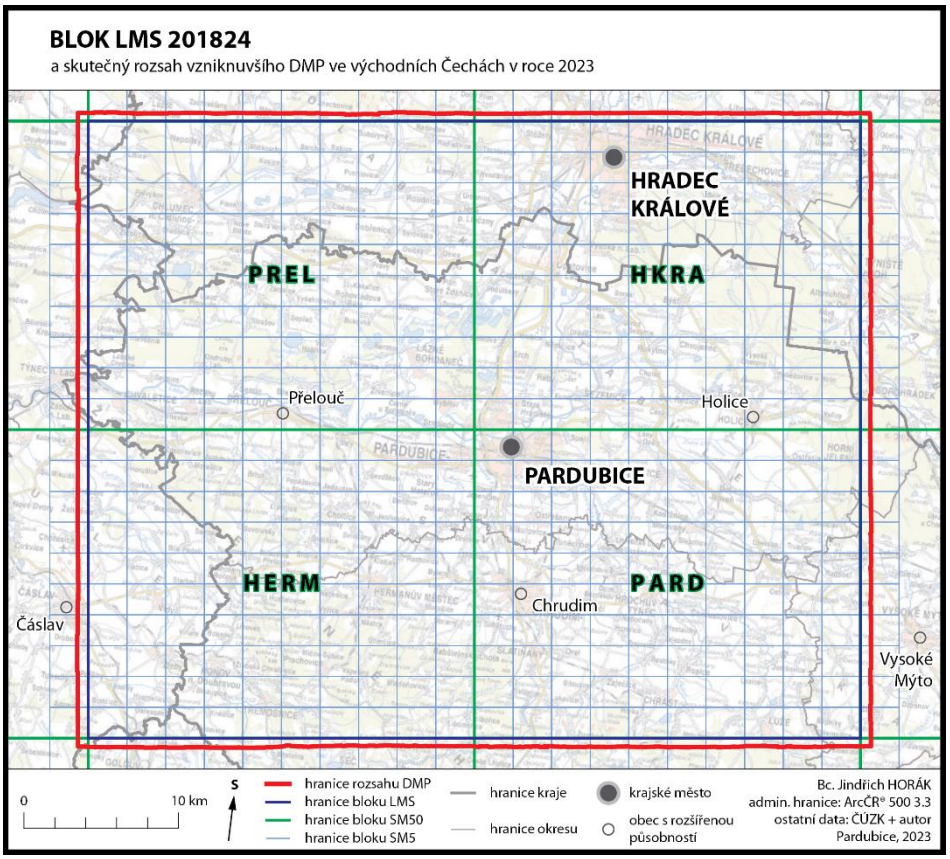
- Metody oprav bodových mračen
- Metoda ohraničujícího polygonu – bodovému mračnu je vytvořen ohraničující polygon
- Metoda spojení ohraničujících polygonů jednotlivých bodových mračen do jednoho celku
- Metoda získání polygonové vrstvy snímků (ze kterých původně vzniklo bodové mračno) pro každý zájmový blok
- Metoda zjištění chybějících bodových mračen v blocích + zjištění, ze kterých leteckých snímků by tato chybějící mračna měla být dokorelována
- Metoda obrazové korelace – vytvoření chybějících bodových mračen pro doplnění bloků
- Metoda získání centroidů ohraničujících polygonů bodových mračen

- Metoda výpočtu Voroného diagramů (Thiessenových polygonů) podle centroidů ohraničujících polygonů bodových mračen
- Metoda ořezání bodových mračen příslušným Voroného diagramem
- Metoda tvorby DMP v listech Státní mapy 1 : 5 000 (SM5) z bodových mračen
- Metoda opravy bodových mračen (a současně DMP) v místech vodních toků a vodních ploch
- Metoda klasifikace bodového mračna
- Metoda vyhledání děr (hrubých chyb) v opraveném bodovém mračnu

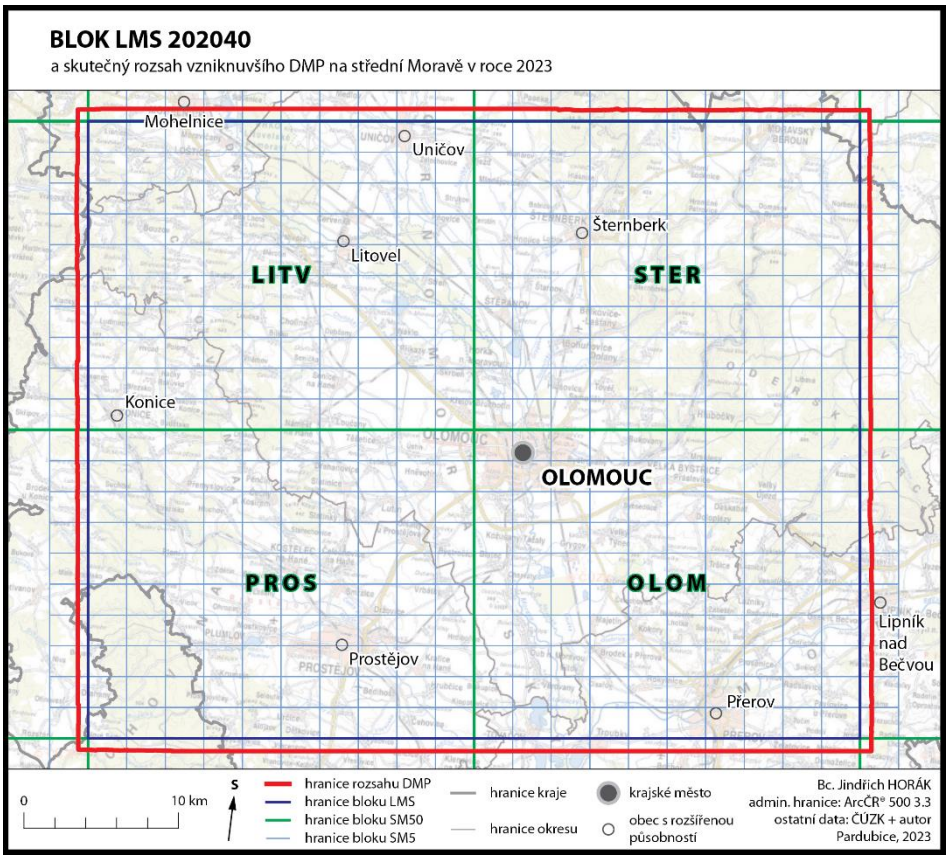
Použitá data

Data použitá v této práci jsou **bodová mračna** ve formátu LAS. Byla poskytnuta na základě kladně vyřízené žádosti od ÚHÚL. Tato bodová mračna vznikla na ÚHÚL obrazovou korelací leteckých snímků pořízených pro ZÚ a používaných ZÚ pro tvorbu barevně vyrovnaného zdánlivě bezešvého georeferencovaného ortofotografického zobrazení zemského povrchu. Po obrazové korelaci byla však na ÚHÚL výsledná bodová mračna zředěna do podoby, která odpovídá vnitřním potřebám ÚHÚL.

Pro diplomovou práci byla data poskytnuta v rozsahu území pěti bloků leteckého měřického snímkování (LMS), vždy ještě s mírným přesahem do vedlejších bloků. Všech pět poskytnutých bloků LMS se skládá ze čtyř **bloků Státní mapy 1 : 50 000** (SM50). Zájmovým územím byla města Olomouc a Pardubice. Obě tato města jsou v datech zachycena ve dvou rocích – 2018 a 2020. Tři bloky LMS z poskytnutých pěti jsou z leteckého snímkování z roku 2018. Blok LMS 201824 se skládá z následujících bloků SM50: PREL (Přelouč), HKRA (Hradec Králové), HERM (Heřmanův Městec) a PARD (Pardubice) (obr. 1). Je v něm tedy zachyceno území a okolí těchto měst. Pro město Olomouc v roce 2018 byly použity dva bloky LMS, neboť samotné město Olomouc se nacházelo přesně na rozhraní dvou těchto LMS bloků – 201835 a 201836. Blok LMS 201835 zahrnuje tyto bloky SM50: ZABR (Zábřeh), RYMA (Rýmařov), LITV (Litovel), STER (Šternberk) a blok LMS 201836 tyto bloky SM50: PROS (Prostějov), OLOM (Olomouc), VYSK (Vyškov) a KROM (Kroměříž). Zbývající dva poskytnuté bloky LMS jsou z roku 2020, kdy platilo již jiné rozvržení bloků. Blok LMS 202029 sestává ze stejných bloků SM50 jako blok LMS 201824, pouze blok SM50 HERM zde není kompletní a vedle bloku SM50 PREL je větší přesah do cizího bloku SM50 KOLI (Kolín) z důvodu, že blok LMS 202029 leží na hranici snímkovaného území v daný rok, která byla vymezena tak, aby přibližně odpovídala krajským hranicím. Blok LMS 202040 je tvořen těmito bloky SM50: LITV, STER, PROS a OLOM (obr. 2).



Obr. 1 Území bloku LMS 201824.



Obr. 2 Území bloku LMS 202040.

Dále byla použita různá data od ZÚ potažmo od Českého úřadu zeměměřického a katastrálního (ČÚZK). Při zjišťování chybějících bodových mračen ve vstupních blocích a při opravě bodových mračen byla využita vektorová polygonová **vrstva všech vodních toků a vodních ploch** v České republice ve formátu shapefile (SHP). Tato vrstva je i výškově určena – lomové body obsahují i souřadnici Z. 3D vrstva vodních toků a vodních ploch vznikla na ZÚ při tvorbě produktu ZABAGED® – Výškopis – Vrstevnice. **Základní báze geografických dat České republiky** (ZABAGED®) je komplexní digitální geografický model území České republiky, je spravován ZÚ ve veřejném zájmu a je využíván jako základní informační vrstva v územně orientovaných informačních a řídících systémech veřejné správy České republiky (ČÚZK, 2023d). Vrstevnice produktu ZABAGED® – Výškopis – Vrstevnice jsou odvozeny z **Digitálního modelu reliéfu 5. generace** (DMR 5G) a představují základní zobrazení výškopisu celého území České republiky s podrobností odpovídající měřítku **Základní topografické mapy 1 : 5 000** (ZTM 5) (Tippner a Dušánek, 2022). Pro vygenerování správného průběhu vrstevnic (ve výše zmíněném produktu) přes vodní toky a vodní plochy bylo DMR 5G doplněno o 3D vektory vodní sítě. Výšková složka 3D vektorů byla odvozena z DMR 5G poloautomatickými nástroji z 2D objektů ZABAGED® *Vodní tok* a *Břehová čára*. Břehovým čarám na vodních plochách byla přiřazena konstantní výška hledáním optimálního průniku 3D vizualizace DMR 5G a vodorovné roviny. U dvoučarých vodních toků byla sledována stejná výška protilehlých bodů na obou březích. Součástí tvorby 3D vodních toků a břehových čar byla poloautomatizovaná kontrola a případná korekce odchylek od DMR 5G. Po dokončení 3D vektorů v ucelených částech území byla kontrolována návaznost dílčích prvků vzniklé sítě, případně byly prováděny drobné korekce (Tippner a Dušánek, 2022).

Při klasifikaci bodových mračen byly rovněž využity produkty z produkce ZÚ/ČÚZK. Klasifikace budov proběhla s využitím dvou vektorových polygonových vrstev ze ZABAGED® a jedné vrstvy z **Registru územní identifikace, adres a nemovitostí** (RÚIAN). RÚIAN je jedním ze základních registrů veřejné správy České republiky. Je veřejným seznamem, jedinečným zdrojem adres a obsahuje také údaje o územních prvcích (ČÚZK, 2023b). Ze ZABAGED® byly použity vrstvy *Budova jednotlivá nebo blok budov* a *Nadzemní zásobní nádrž*, z RÚIAN byla použita vrstva *Stavební objekty*. Tyto tři vrstvy byly sloučeny do jedné a v dané podobě použity pro klasifikaci budov. Klasifikace vegetace byla uskutečněna dvěma různými metodami. V jedné z nich byla použita vrstva ze ZABAGED®, konkrétně vrstva *Lesní půda se stromy*. Ke klasifikaci do třídy ground (země) byl využit produkt **ZABAGED® – Výškopis – DMR 5G**. Již dříve zmíněný DMR 5G představuje zobrazení přirozeného nebo lidskou činností upraveného zemského povrchu v digitálním tvaru ve formě výšek diskrétních bodů v nepravidelné trojúhelníkové síti (TIN) bodů o souřadnicích X, Y, a H, kde H reprezentuje nadmořskou výšku ve výškovém referenčním systému Balt po vyrovnání (Bpv) s úplnou střední chybou výšky 0,18 m v odkrytém terénu a 0,3 m v zalesněném terénu. Model vznikl z dat pořízených metodou leteckého laserového skenování v letech 2009 až 2013 (ČÚZK, 2023e).

Jak již bylo nastíněno výše, v různých částech práce byla využívána data o rozdělení České republiky do bloků, konkrétně byly použity **bloky LMS**, **SM50** a **SM5**. Všechny opět od ZÚ/ČÚZK.

Při tvorbě náhledových map území dvou LMS bloků a území, na kterém byl skutečně vytvořen DMP, byla kromě výše zmíněných bloků jako podklad použita **Základní mapa České republiky** (ZM ČR) od ČÚZK jako Web Map Service (WMS služba) a vektorová polygonová data krajů a okresů. Tato data byla získána z digitální vektorové geografické databáze České republiky **ArcČR® 500** verze 3.3, která vznikla v roce 2016

a je poskytována v měřítku 1 : 500 000 ve spolupráci společnosti ARCDATA PRAHA, s. r. o., ZÚ a Českého statistického úřadu (ČSÚ).

Pro zpracování obrazové korelace v prostředí programu Agisoft Metashape bylo od ZÚ poskytnuto celkem 39 leteckých měřických snímků (LMS) – vždy 13 LMS ve třech řadách – s 60% podélnými a 40% příčnými překryvy. Použité LMS byly pořízeny digitální leteckou měřickou kamerou **Leica DMC III** a jedná se o tzv. **Color InfraRed (CIR) snímky**. CIR (v nepravých barvách) snímky vznikají tak, že červené pásmo je nahrazeno blízkým infračerveným pásmem, zelené pásmo červeným a modré zeleným. Ortofoto z CIR snímků je využíváno především při vyhodnocování stavu vegetace a ZÚ jej vytváří pro ÚHÚL (ČÚZK, 2023a).

Použité programy

Veškeré skripty v programovacím jazyce Python byly vyvíjeny v prostředí programu **PyCharm** verze **2019.3.4 Community Edition** od české vývojářské společnosti **JetBrains**. PyCharm je specializované integrované vývojové prostředí (angl. integrated development environment – IDE) pro jazyk Python poskytující širokou škálu nezbytných nástrojů, a tím vytváří pohodlné prostředí pro produktivní vývoj nejen v jazyce Python. Verze Community je vyvíjena s otevřeným zdrojovým kódem a je poskytována zdarma (JetBrains, 2023). PyCharm umožňuje nastavit Python nainstalovaný spolu s geografickým informačním systémem (GIS) **ArcGIS Pro** jako svůj tzv. interpreter, tedy že bude využíván tento Python při spouštění skriptů. Tím je velmi snadno umožněna integrace knihovny ArcPy a tím volání geoprocessingových nástrojů z prostředí ArcGIS Pro ve skriptech.

Naprostá většina operací s bodovými mračny ve formátu LAS byla zpracovávána nástroji z programového balíku **LAStools** od německé společnosti **rapidlasso**, který byl autorovi práce zapůjčen od ZÚ. LAStools je soubor vysoce efektivních, dávkově skriptovatelných nástrojů pro příkazový řádek. Obsahuje nástroje například pro klasifikaci, filtraci, rasterizaci či ořezávání dat LiDAR (či jakýchkoli bodových mračen). Všechny nástroje mají i grafické rozhraní a lze je ve formě tzv. toolboxu (sady nástrojů) připojit i do GIS programů jako ArcGIS Pro či QGIS (Rapidlasso, 2023j). Nástroje z tohoto balíku byly při práci s bodovými mračny spouštěny v Python skriptech, grafická prostředí ani připojení do GIS programů nebyla žádným způsobem využita.

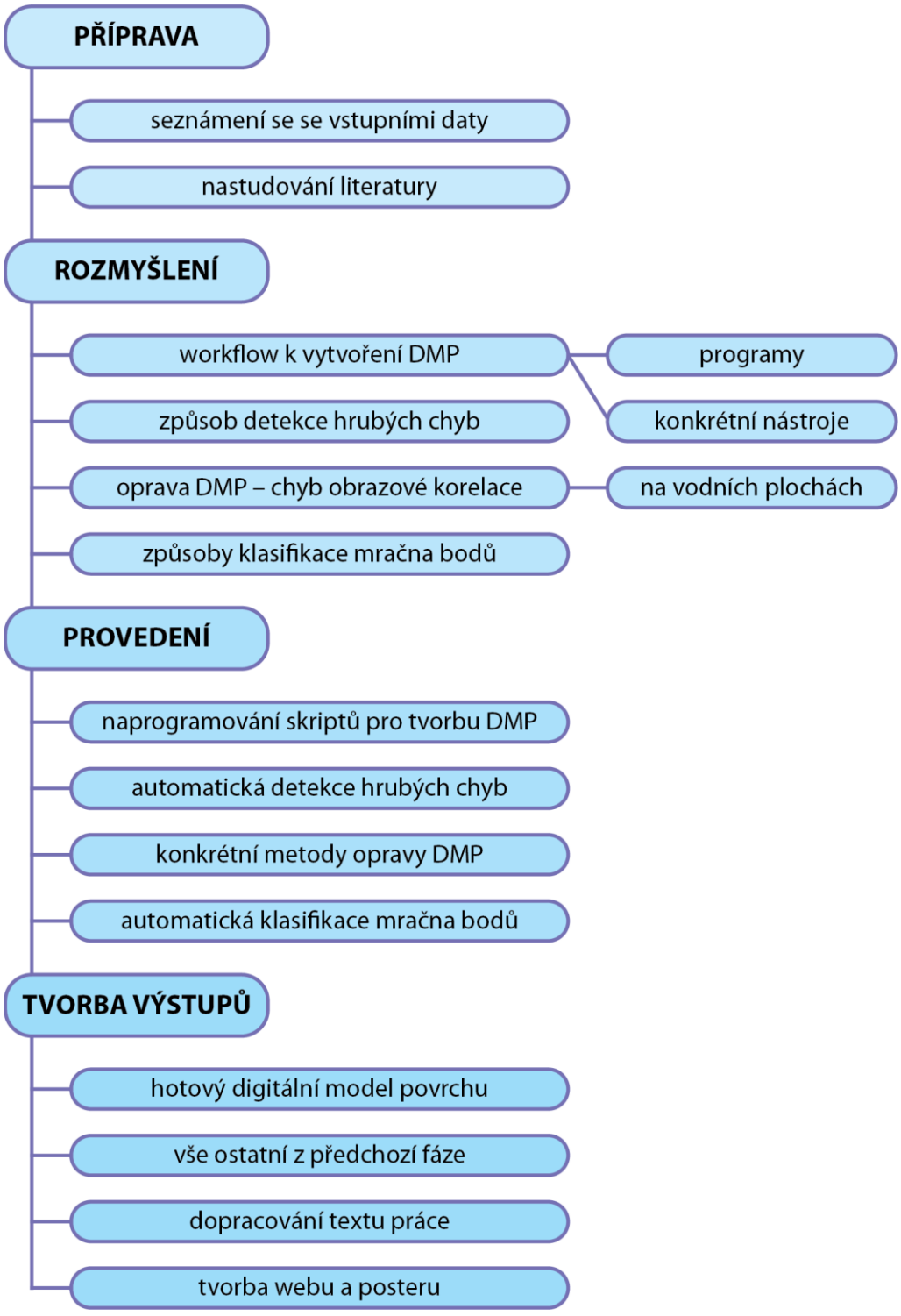
Před samotným skriptováním byly operace nepracující s bodovými mračny zkoušeny v prostředí GIS programu **ArcGIS Pro** verze **3.0.0** od americké společnosti **Esri**. Tento program spolu s dalším GIS programem – **QGIS** verze **3.24.1-Tisler** – byl využit též pro rychlé zobrazení různých meziproduktů i výsledného DMP. Program **QGIS** byl využit rovněž pro zobrazování výsledných i klasifikovaných bodových mračen či pro stahování dat RÚIAN. V programu **ArcGIS Pro** byla připravena data na tvorbu náhledových map území LMS bloků a dalších datových náhledů.

Program **SURE Aerial** byl využit pro obrazovou korelaci leteckých snímků k získání chybějících bodových mračen v datech od ÚHÚL. Těmito nově dokorelovanými bodovými mračny byly doplněny bloky LMS a teprve poté začal proces tvorby DMP z bodových mračen. SURE Aerial je program vyvinutý společností **nFrames**, která je nejnovějším výzkumným a vývojovým centrem společnosti Esri v německém Stuttgartu (NFrames, 2023d). Jeho funkce umožňují uživatelům bezproblémově pořizovat, zpracovávat a analyzovat vysoce kvalitní 3D data ze snímků a dat LiDAR (NFrames, 2023b).

Program **Agisoft Metashape** byl použit též pro obrazovou korelaci, v jeho prostředí byl vytvořen DMP z CIR snímků. Agisoft Metashape od ruské společnosti Agisoft je programové vybavení, které provádí fotogrammetrické zpracování digitálních snímků a generuje 3D prostorová data pro použití v GIS (Agisoft, 2023).

Postup zpracování

Postup zpracování je patrný z grafického znázornění (obr. 3).



Obr. 3 Postup zpracování.

4 VLASTNÍ ŘEŠENÍ

V práci byla zpracována vstupní data – bodová mračna vzniklá obrazovou korelací leteckých měřických snímků na ÚHÚL, kde byla rovněž zředěna hustota bodů. Ze vstupních dat byl dále popsaným způsobem vytvořen spojitý DMP. V místech, kde obrazová korelace selhává (především na vodních plochách) byl DMP opraven. Byly nalezeny nástroje pro detekci hrubých chyb (děr primárně nevyskytujících se na vodních plochách) v bodových mračnech. Bodové mračno bylo též automatizovaně klasifikováno do třech tříd.

Rovněž byl DMP vytvořen v programu Agisoft Metashape především k ověření, zda je možné v jeho prostředí zpracovávat dostatečně kvalitní DMP z leteckých měřických snímků běžně pořizovaných pro ZÚ pro tvorbu celoplošného bezešvého ortofota.

4.1 Počáteční příprava dat

Několik prvních skriptů pouze upravilo vstupní data či vytvořilo pomocné soubory, aby mohl být vytvořen DMP.

4.1.1 Oprava vstupních bodových mračen

Nejprve byla data opravena a vyčištěna. K tomuto účelu byl využit nástroj **las2las** z programového balíku **LAStools** a jeho specifické pro tento záměr určené argumenty. Konkrétně argumenty *auto_reoffset* kvůli ušetření objemu dat, kdy jsou data v každé ose krácena o určitou konstantu a *rescale 0.01 0.01 0.01* na zmenšení počtu desetinných míst, 0,01 znamená centimetrovou přesnost. Obecně nástroj **las2las** čte a zapisuje LiDAR data ve formátech LAS/LAZ/ASCII a může je filtrovat, transformovat, nastavovat jim projekci či jinak upravovat jejich obsah (Rapidlasso, 2023c).

4.1.2 Ohraničení jednotlivých bodových mračen

Poskytnutým a opraveným bodovým mračnům byla vytvořena ohraničení jako digitální vektorová vrstva ve formátu SHP. Konkrétně byl použit nástroj **lasboundary** opět z programového balíku **LAStools**, který načte LiDAR data z formátu LAS/LAZ/ASCII a pro body bodové mračna vypočítá ohraničující polygon (Rapidlasso, 2023d).

Pomocí knihovny ArcPy byla rovněž vytvořenému polygonu definována projekce (*Define Projection*) do S-JTSK (systém Jednotné trigonometrické sítě katastrální) ovšem do jeho podoby s již otočenými osami (osa X směřuje k východu a osa Y k severu) a zápornými souřadnicemi, tedy S-JTSK/Krovak East North; EPSG: 5514. Dále byl též u tohoto polygonu opět GIS nástroji z knihovny ArcPy (*Add Field*, *Calculate Field*) vložen do nově vytvořeného atributu *name* název ohraničeného bodového mračna.

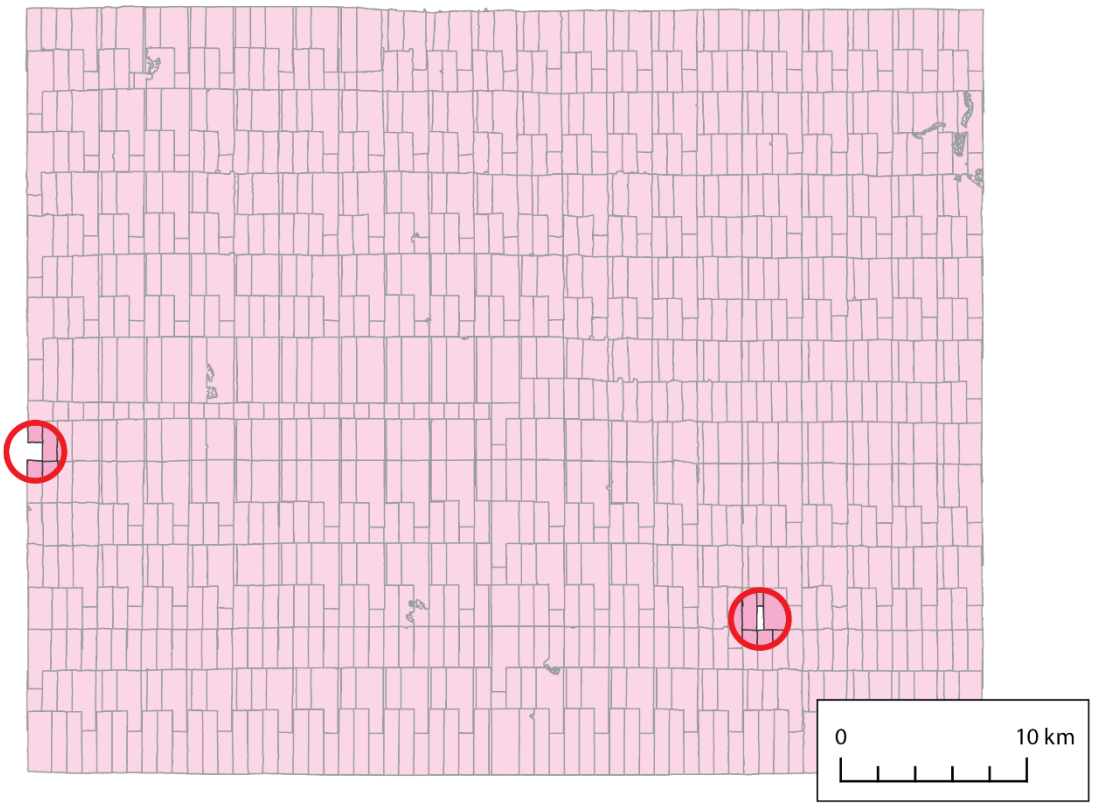
4.1.3 Spojení ohraničení do bloku LMS

Tento jednoduchý skript pouze spojuje (pomocí knihovny ArcPy – nástroj *Merge*) předchozím skriptem vytvořené ohraničující polygony pro každé bodové mračno do jednoho SHP souboru – vždy podle bloku LMS.

4.2 Doplnění nekompletních bloků

Po zobrazení SHP souboru všech ohraničení bodových mračen pro jeden blok LMS zde bylo zjištěno, že bloky jsou nekompletní (obr. 4), a tedy že od ÚHÚL nebyla dodána všechna bodová mračna, která dodána být měla. Vytvoření souvislého DMP v rozsahu celých

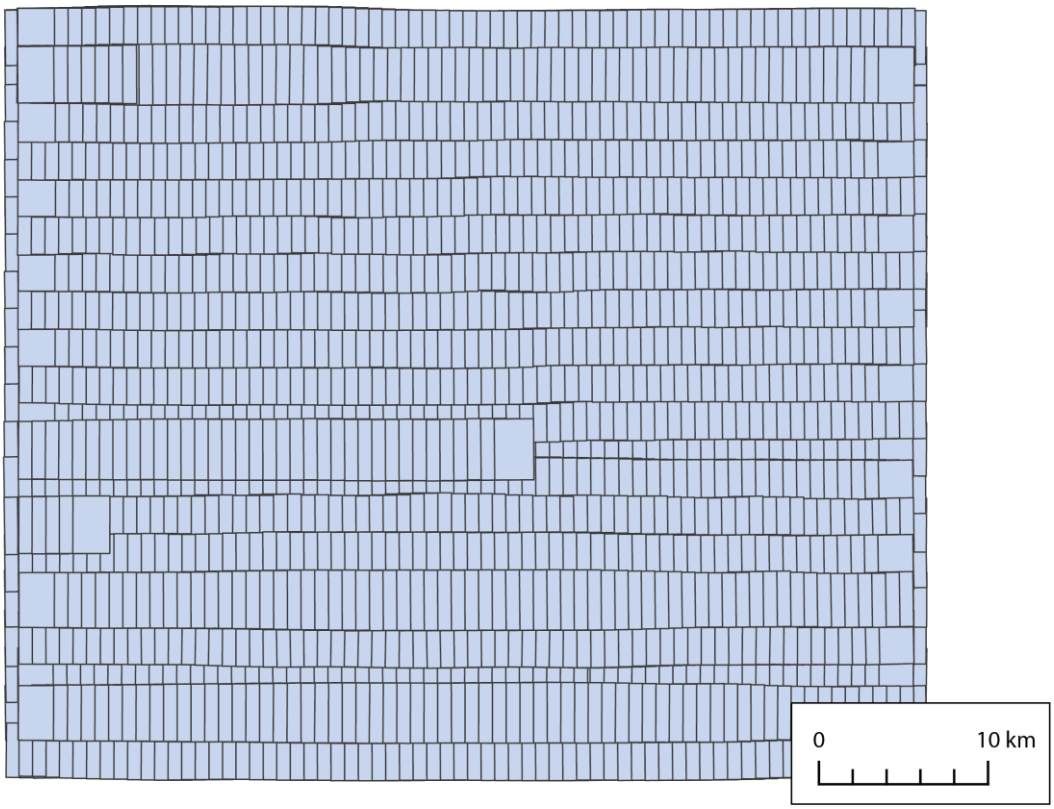
daných bloků vyžaduje kompletnost bloků, proto byly automatizovaně zjištěny snímky (stereodvojice), které měly tvořit chybějící bodová mračna v blocích. Na těchto zjištěných snímcích byla provedena dodatečná obrazová korelace a vzniklými bodovými mračny byly bloky doplněny.



Obr. 4 Nekompletní blok LMS 201835 se znázorněnými místy chybějících bodových mračen.

4.2.1 Získání ohraničení snímků, ze kterých jsou bodová mračna

Od ZÚ byla poskytnuta ArcGIS geodatabáze (GDB) obsahující ohraničení všech pořízených snímků od roku 1936 do roku 2021. Pomocí výběru podle umístění (*Select Layer By Location*) a výběru podle atributů (*Select Layer By Attribute*) byly z této GDB vybrány a exportovány všechny snímky pro zpracovávané LMS bloky v této práci. Nejprve byla ze zmíněné GDB výběrem podle umístění vybrána všechna ohraničení snímků mající společnou polohu (intersect) s ohraničeními bodových mračen spojených do bloku. A z tohoto výběru byla subvýběrem pro každý blok vybrána pouze ta ohraničení snímků, která byla z požadovaného roku, tedy buď 2018 či 2020. Výsledná ohraničení snímků pro daný rok a daný blok LMS byla uložena jako jeden SHP soubor (obr. 5).



Obr. 5 Ohraničení snímků pro blok LMS 201835.

4.2.2 Zjištění snímků k dodatečné obrazové korelaci

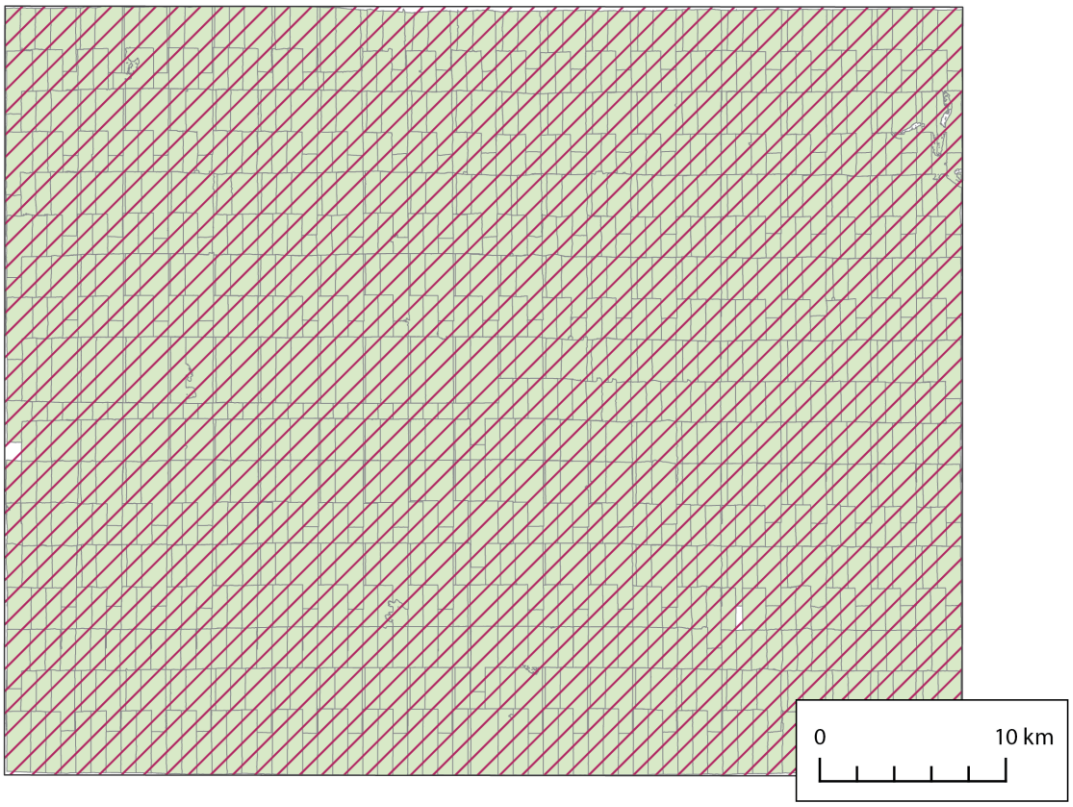
Tímto komplexním skriptem byly z děr v SHP souboru všech ohraničení pro jeden blok LMS zjištěny snímky, na kterých byla později provedena dodatečná obrazová korelace. Výsledkem je jednak SHP soubor ohraničení chybějících snímků pro každý blok LMS a jednak textový soubor obsahující vždy název každého snímku a ze kterého LMS bloku daný snímek je. Tento textový soubor byl použit na vygenerování projektů pro dodatečnou obrazovou korelaci.

Nejdříve byly sestaveny dva seznamy (dvě pole polí). První seznam zahrnuje cesty k SHP souborům všech ohraničení bodových mračen pro jeden blok LMS. Vnitřní pole obsahuje cesty k daným souborům z jednoho zpracovávaného roku. Zpracovávány byly dva roky, vnější pole proto obsahuje dvě vnitřní pole. Druhý seznam stejnou logikou zahrnuje cesty k SHP souborům ohraničení snímků pro daný rok a daný blok LMS (vytvořeno předchozím skriptem).

Následně byly oba seznamy současně procházeny integrovanou funkcí programovacího jazyka Python *zip()*. Funkce *zip()* v jazyce Python vytvoří iterátor, který sdruží prvky ze dvou nebo více iterovatelných souborů a umožňuje nad těmito sdruženými soubory paralelně iterovat (Ramos a kol., 2019). Oběma seznamy bylo paralelně iterováno, tedy bylo v jeden okamžik operováno s SHP souborem všech ohraničení bodových mračen pro jeden blok LMS a s SHP souborem ohraničení snímků pro tentýž blok LMS.

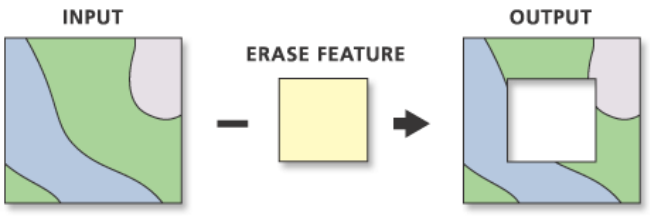
Bylo vytvořeno deset dočasných složek pro ukládání výstupů deseti následujících operací, které všechny využívají nástroje knihovny ArcPy. Nejprve byla vytvořena kopie (*Copy Features*) SHP souboru všech ohraničení bodových mračen pro jeden blok LMS, a to z důvodu, že následující operace na opravu geometrie (*Repair Geometry*) přímo modifikuje vstupní data, tak aby byla opravena kopie a originální vstupní data byla

zachována v nezměněné podobě. Oprava geometrie odstranila prvky s nulovou geometrií, které se vyskytovaly v některých ohraničeních bodových mračen při ohraničování vodních ploch. Takto opravenému souboru všech ohraničení bodových mračen byla vypočítána minimální ohraničující geometrie nástrojem *Minimum Bounding Geometry*. Tento nástroj vytvoří třídu prvků obsahující polygony, které představují zadanou minimální ohraničující geometrii obklopující každý vstupní prvek nebo každou skupinu vstupních prvků (Esri, 2023d). Aby byl danému bloku vytvořen jeden v podstatě minimální ohraničující obdélník (angl. Minimum Bounding Rectangle (MBR)), byl daný nástroj nastaven tak, aby všechny vstupní prvky byly považovány za jednu skupinu (obr. 6 – šrafovaně).



Obr. 6 MBR bloku LMS 201835 nad ohraničeními bodových mračen.

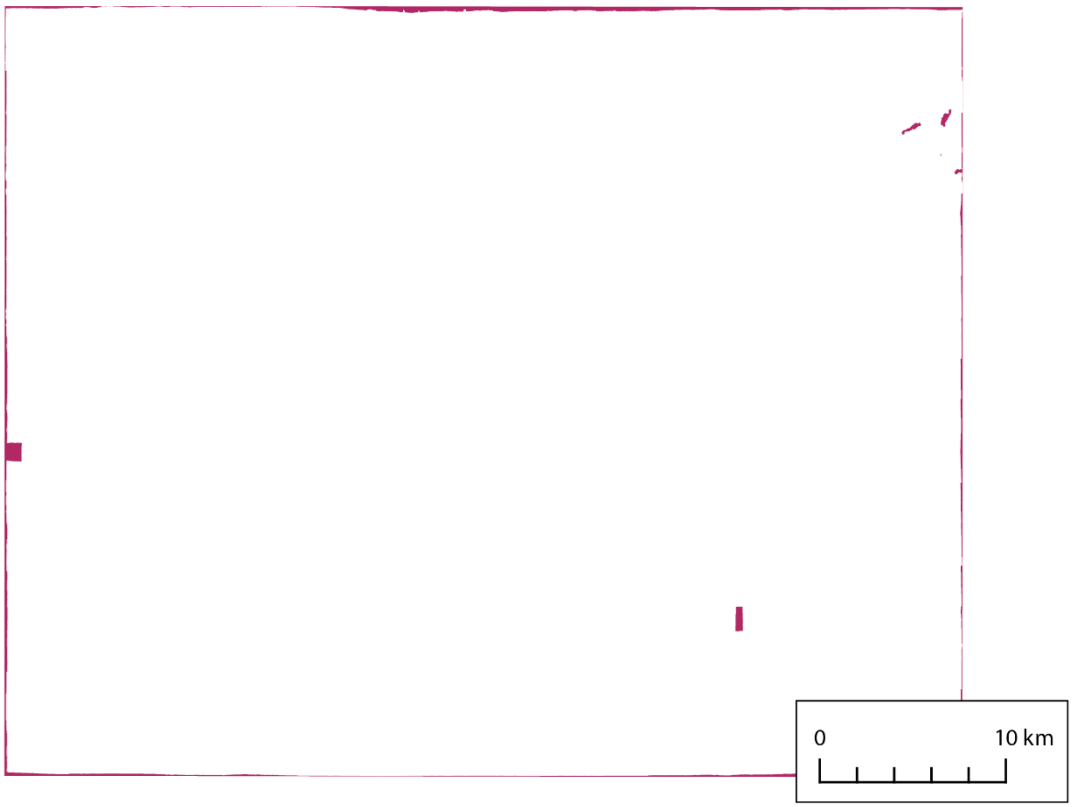
Poté byl použit nástroj k mazání geometrie, tedy *Erase*. Zmíněný nástroj pracuje na principu překrytí dvou vrstev. Geometrie shodná ve vstupní vrstvě a „mazací“ vrstvě bude ze vstupní vrstvy odstraněna (Esri, 2023b) (obr. 7).



Obr. 7 Princip nástroje Erase (Esri, 2023b).

Vstupní vrstvou byl vytvořený MBR bloku všech ohraničení bodových mračen a mazací vrstvou byl samotný tento blok ohraničení bodových mračen. Tímto krokem byla z MBR odstraněna všechna ohraničení bodových mračen a zůstaly díry po chybějících bodových

mračnech a díry po vodních plochách. Kromě děr zůstala i „polygonová obvodová linie“ MBR neboli oblast, o kterou byl MBR po obvodu větší než blok ohraničení bodových mračen (obr. 8).



Obr. 8 Z MBR odstraněná ohraničení bodových mračen.

Tato oblast byla několika dalšími kroky odstraněna. MBR byl nejdříve nástrojem *Polygon To Line* převeden na obvodovou linii, této linii byla pomocí nástroje *Buffer* spočítána obalová zóna o vzdálenosti 350 m. Tato hodnota byla určena částečně empiricky a částečně expertním odhadem. Konečné odstranění zmíněné oblasti proběhlo opět pomocí nástroje *Erase*. Jako vstupní vrstva byl použit výsledek přechozího využití nástroje *Erase* a jako „mazací“ vrstva byla použita vytvořená obalová zóna kolem obvodové linie MBR.

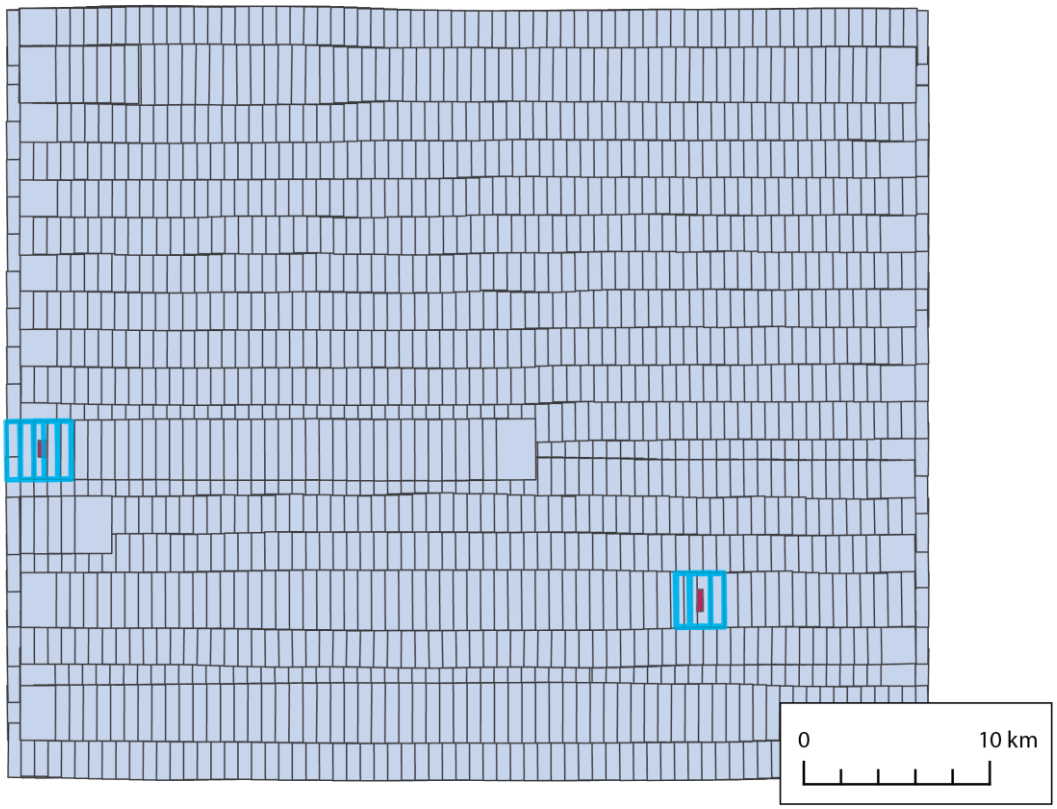
V aktuální vrstvě děr byly multipolygony separovány na jednotlivé polygony nástrojem *Multipart To Singlepart*, aby mohly být později jednotlivě vybírány. Pomocí nástroje *Get Count* byl zjištěn počet vzniklých polygonů, a jestliže byl tento počet nenulový, byl všem jednotlivým polygonům vypočítán geometrický atribut rozlohy nástrojem *Calculate Geometry Attributes*, rovněž k umožnění pozdější selekce.

Ještě před selekcí byla zpracována voda, respektive díry po vodních plochách. Nástrojem *Clip* byla polygonová vrstva všech vodních toků a vodních ploch (popsána v podkapitole Použitá data v kapitole 3) oříznuta vždy podle souboru ohraničení snímků pro daný blok LMS. Tím byla získána veškerá voda v území snímků daného bloku LMS. Tyto multipolygony vodních ploch byly také separovány na jednotlivé polygony.

Následujícími několika kroky byla osamostatněna geometrie děr po chybějících bodových mračnech. Výběrem podle umístění (*Select Layer By Location*) byly z vrstvy všech děr vybrány takové polygony, které se nijak neprotínají (*intersect*) s polygony vodních ploch, ovšem byly k nim přidány i ty, které celou vodní plochu obsahují uvnitř (*contains*). Opět po ověření, že počet polygonů je nenulový, byly pomocí výběru podle atributů (*Select Layer*

By Attribute) z výběru odebrány polygony s menší rozlohou než 20 000 m^2 a polygony s větší rozlohou než 10 km^2. Hranice odebrání podle rozlohy byly rovněž určeny empiricky. Vybrané zůstaly již pouze díry po chybějících bodových mračnech, tento výběr byl uložen do nového souboru nástrojem *Copy Features*.

S využitím vrstvy děr po chybějících bodových mračnech byla výběrem podle umístění vybrána příslušná ohraničení snímků (obr. 9), na kterých byla později provedena dodatečná obrazová korelace, jejímiž výslednými bodovými mračny byly doplněny bloky bodových mračen v rozsahu bloků LMS. Konkrétně byla vybrána ta ohraničení, která se protínala se zmíněnou vrstvou děr, ovšem ve zvláštní vzdálenosti -100 m. Tato záporná vzdálenost byla zvolena proto, aby se u chybějících bodových mračen (děr) vybraly skutečně pouze dané snímky (většinou jedna stereodvojice), které mají bodové mračno tvořit, a ne všechny, které se pouze dané díry dotýkají. Takto vybraná ohraničení snímků byla nástrojem *Copy Features* uložena do výsledného SHP souboru.



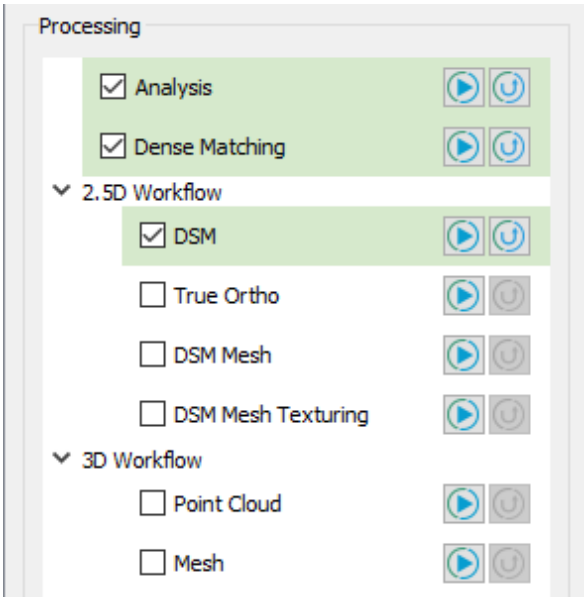
Obr. 9 Vybraná ohraničení snímků k provedení dodatečné obrazové korelace.

Na závěr byl vygenerován již zmíněný textový soubor s názvy snímků a označením bloku k vytvoření projektů pro dodatečnou obrazovou korelaci. Po ověření nástrojem *Get Count*, že výsledný SHP soubor s ohraničeními snímků (pro dokorelování děr) v daném bloku LMS je nenulový, byl do tohoto SHP souboru přidán atribut (*Calculate Field*) *block* obsahující vždy označení daného bloku LMS (např. 201824). K exportu názvů snímků (atribut *NAME*) a označení bloků (přidaný atribut *block*) byl využit nástroj *Export Feature Attribute To ASCII* (*ExportXYv*), který exportuje souřadnice třídy prvků a hodnoty atributů do textového souboru ASCII odděleného mezerou, čárkou, tabulátorem nebo středníkem (Esri, 2023c). Jako oddělovač byla použita mezera. Jelikož souřadnice nebyly žádoucí, byly z textového souboru vzniklého z *ExportXYv* vybrány pouze zmíněné atributy názvu a označení bloku, které byly automatizovaně přepsány do jiného a současně výsledného textového souboru.

Zcela nakonec byly ještě odstraněny všechny dočasné složky i se svými obsahy. Jednu z těchto složek nebylo možné z neznámých důvodů odstranit přímo v tomto skriptu, byla tedy odstraněna pomocným samostatným skriptem.

4.3 Dodatečná obrazová korelace

Dodatečná obrazová korelace byla zpracována v programu **SURE Aerial**. Postup práce v tomto programu byl velmi jednoduchý. Vydavatel programu popisuje workflow tak, že stačí naimportovat snímky a prvky vnější i vnitřní orientace a poté jen vybrat požadované výstupní produkty (NFrames, 2023c). Předpokladem bylo, že pro vygenerování jednoho bodového mračna k zaplnění díry v bloku LMS postačí do programu nahrát dva snímky (jednu stereodvojici). Ovšem minimální počet snímků, který lze do SURE Aerial nahrát, jsou tři. K dané stereodvojici byly tedy přidány ještě dva vedlejší snímky – každý z jedné strany (jeden ze západu a jeden z východu), nahrávány byly tedy vždy čtyři snímky. Z podstaty práce a nahrávaných snímků byl zvolen scénář *Aerial Nadir*, tedy že vstupem jsou letecké nadirové snímky. Tento scénář je využíván pro tvorbu 2,5D produktů (NFrames, 2023a). Kvalita výstupu byla nastavena na *Ultra*. V rámci *2,5D Workflow* byl pak jako požadovaný výstupní produkt vybrán pouze *DSM* (Digital Surface Model, česky DMP) (obr. 10).

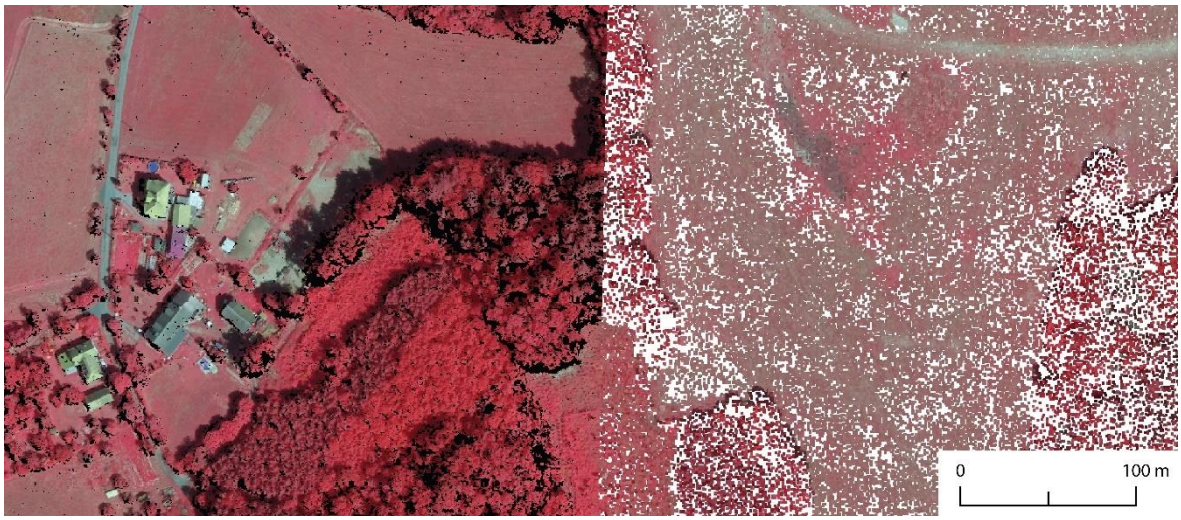


Obr. 10 Aerial SURE: Nastavení workflow.

S tímto nastavením byl vygenerován DMP jednak ve formě bodového mračna v několika souborech formátu LAS a jednak ve formě rastru ve formátu TIFF (ten nebyl nijak použit). Soubory formátu LAS byly nástrojem **las2las** převedeny do formátu LAZ, tedy do bezeztrátově komprimované varianty LAS. Pomocí nástroje **lasmerge** byly tyto soubory spojeny do jediného souboru LAZ. Nástroj **lasmerge** spojuje mnoho souborů do jednoho nebo jeden rozděluje do mnoha. Při slučování načte více bodových mračen ve formátu LAS/LAZ/ASCII a spojí je do jediného LAS/LAZ/ASCII souboru (Rapidlasso, 2023h). DMP byl automaticky vygenerován pouze z požadované stereodvojice (ze dvou snímků uprostřed z vložených čtyř).

Tímto popsaným způsobem byla vytvořena všechna bodová mračna, která chyběla v datech od ÚHÚL. Bloky LMS byly tedy těmito bodovými mračny doplněny. Od bodových mračen poskytnutých z ÚHÚL se tato dodatečně dokorelovaná bodová mračna nejvíce

odlišují tím, že nebyla již nijak ředěna, hustota bodů je u nich tedy vyšší. Zatímco hustota bodů u bodových mračen od ÚHÚL se pohybuje kolem hodnoty 1,5 bodu/m^2, hustota neředěných bodových mračen z SURE Aerial je 10–12 bodů/m^2. Vizuálně je rozdíl více než markantní (obr. 11 – nalevo mračno z SURE Aerial, napravo mračno od ÚHÚL).



Obr. 11 Vizuální srovnání rozdílné hustoty bodových mračen z různých zdrojů.

4.4 Příprava spojení bodových mračen v překryvech

Po vytvoření všech chybějících bodových mračen dodatečnou obrazovou korelací byla tato bodová mračna doplněna do příslušných bloků LMS, čímž se tyto bloky staly kompletními.

Na kompletních blocích byla nejprve zopakována série kroků (na nekompletních blocích již proběhlých) spočívající v opravě bodových mračen, vytvoření ohraničení všech bodových mračen a spojení těchto ohraničení do jednoho SHP souboru pro každý blok LMS (podrobněji v kapitole 4.1). Tím byla zajištěna konzistence všech dat pro následné zpracování. Konkrétně zde byla stanovena pravidla pro spojení bodových mračen v překryvových pásech.

4.4.1 Centroidy ohraničení bodových mračen

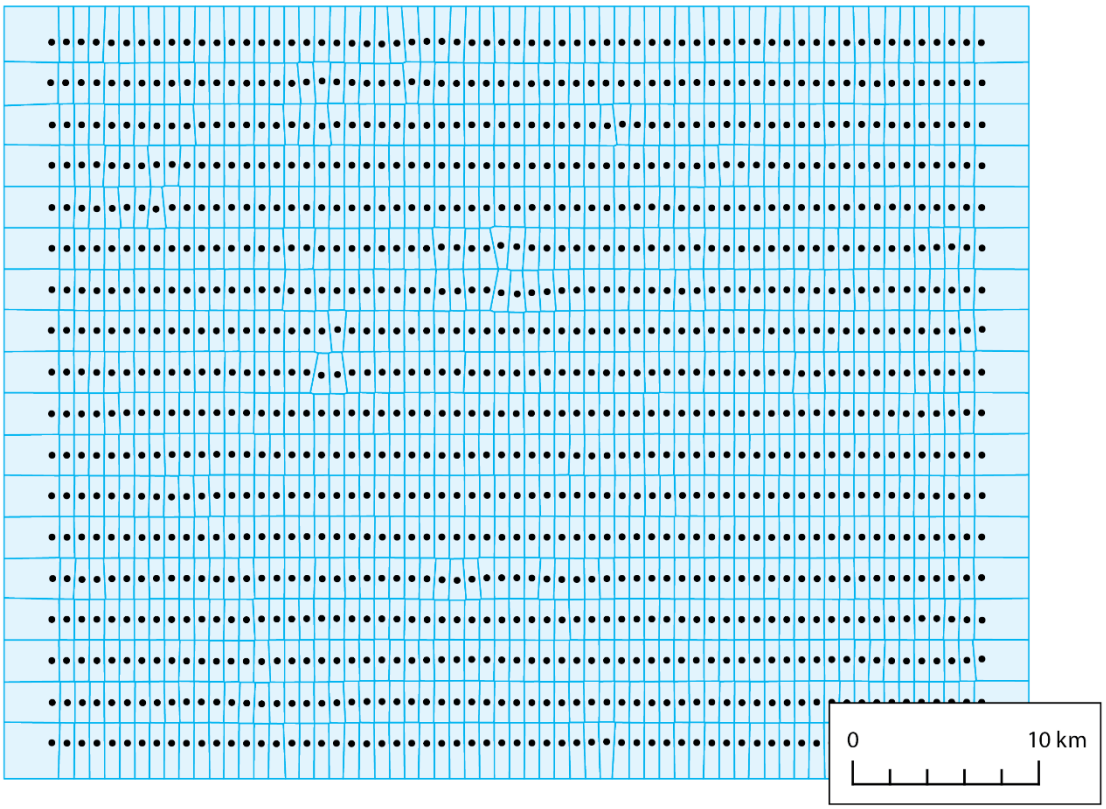
Prvním krokem při spojení bodových mračen bylo zpracování vrstvy centroidů. Centroid vznikl pro každé ohraničení bodového mračna v daném bloku LMS (obr. 12). Zpracování proběhlo prostřednictvím nástroje *Feature To Point* z knihovny ArcPy, kdy vstupem do tohoto nástroje byla vždy vrstva všech ohraničení bodových mračen v jednom bloku LMS vzniklá v předchozím kroku pracovního postupu.

Obr. 12 Kompletní blok LMS 201824 a centroidy ohraničení bodových mračen.

4.4.2 Thiessen polygony (Voronoi diagramy)

Kolem bodové vrstvy centroidů byla vytvořena vrstva Thiessen polygonů (obr. 13). Každý Thiessen polygon obsahuje pouze jeden bodový vstupní prvek. Jakékoli místo uvnitř Thiessen polygonu je blíže k přidruženému bodu než k jakémukoli jinému vstupnímu bodu (Esri, 2023a). Díky kompletnosti bloků – tím pádem i díky rovnoměrnosti rozmístění centroidů – byly Thiessen polygony až na výjimky vytvořeny jako téměř pravidelné a téměř stejně velké obdélníky (kromě krajních).

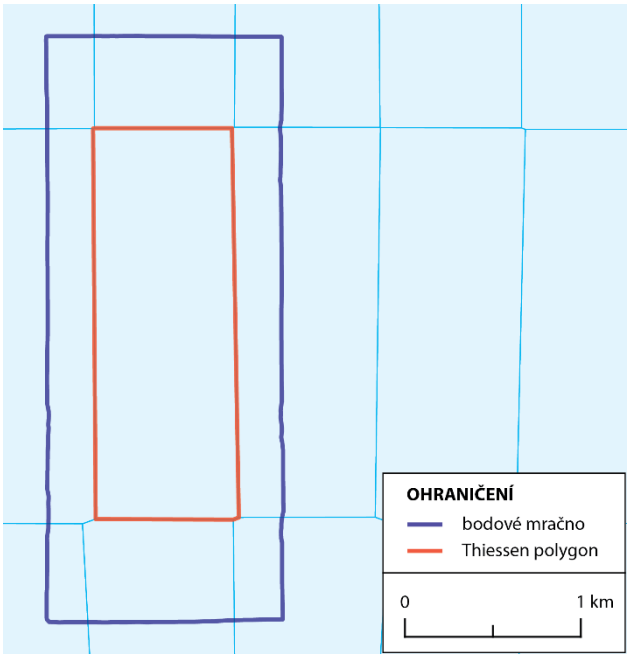
Pro vytvoření Thiessen polygonů byl použit odpovídající nástroj z knihovny ArcPy *Create Thiessen Polygons*, kdy vstupem byla právě bodová vrstva centroidů.



Obr. 13 Thiessen polygony kolem centroidů v bloku LMS 201824.

4.4.3 Ořezání bodových mračen Thiessen polygony

Posledním krokem přípravy spojení bodových mračen tam, kde se překrývají, bylo jejich ořezání vytvořenými Thiessen polygony. Z každého bodového mračna tak byla vybrána jeho středová část (obr. 14) a jelikož na sebe plynule navazují Thiessen polygony, tak na sebe navazují i takto ořezaná bodová mračna.



Obr. 14 Ořezání bodových mračen Thiessen polygony.

I zde byly nejdříve sestaveny dvě pole polí (ve skutečnosti jedno pole polí polí a jedno pole polí). První s cestami k bodovým mračnům a druhé s cestami k blokům Thiessen polygonů. Oběma poli bylo opět paralelně iterováno pomocí funkce *zip()*, ovšem ne ve stejné úrovni zanoření. Zatímco u bodových mračen bylo skutečně procházeno jedno mračno po druhém, tak k danému mračnu byl z druhého pole vzat vždy celý odpovídající LMS blok Thiessen polygonů. Díky tomu, že vrstva Thiessen polygonů obsahuje atribut *name* vždy s názvem, ke kterému bodovému mračnu daný Thiessen polygon patří (atribut přenesen z bodového mračna přes vrstvu centroidů), byl konkrétní Thiessen polygon k ořezu daného bodového mračna získán výběrem podle atributů. Výběr podle atributů byl proveden tak, aby se k bodovému mračnu vybral Thiessen polygon podle shodného názvu (*name*) jako bodové mračno. Takto vybraný Thiessen polygon byl uložen do dočasného SHP souboru pomocí *Copy Features*.

Když bylo připravené samotné bodové mračno i soubor odpovídajícího Thiessen polygonu, bylo provedeno samotné ořezání. To proběhlo pomocí nástroje **lasclip**, jehož vstupem je jednak bodové mračno ve formátech LAS/LAZ/TXT a jednak SHP/TXT soubor s jedním nebo více polygony. Nástroj pak vyřízne body z bodového mračna podle vložené polygonové vrstvy a do nového LAS/LAZ/TXT souboru uloží body buď uvnitř polygonu/polygonů či vně – podle nastavení. Kromě samotného ořezávání lze tímto nástrojem body uvnitř/vně vložené polygonové vrstvy i klasifikovat (Rapidlasso, 2023f). Vstupními vrstvami tedy bylo konkrétní bodové mračno ve formátu LAZ a dočasný SHP soubor příslušného Thiessen polygonu. Výstupní ořezané bodové mračno bylo uloženo do nového LAZ souboru. Nakonec byl odstraněn dočasný SHP soubor jednoho Thiessen polygonu.

4.4.4 Alternativní řešení

Byla provedena též alternativní příprava na spojení bodových mračen v překryvových oblastech (a následně i tvorba DMP z této alternativy). Toto řešení tkvělo ve vytvoření obalových zón o vzdálenosti 500 m (nástrojem *Buffer*) kolem Thiessen polygonů a ořezání bodových mračen těmito obalovými zónami. Předpokladem bylo, že tímto způsobem nebude ztraceno tolik bodů z originálních bodových mračen, a navíc že v oblastech překryvů obalových zón bude bodů ještě více (vždy ze sousedních mračen) a tím bude výsledný DMP kvalitnější. Tento předpoklad se po vytvoření DMP neukázal jako správný, proto bylo od tohoto řešení upuštěno. V programovém kódu tvorby Thiessen polygonů je pozůstatek popsaného postupu zanechán v komentářích (není aktivní).

4.5 Digitální model povrchu

Díky předchozím operacím byla data připravena k samotné tvorbě DMP. Po vytvoření DMP byla bodová mračna v místech, kde obrazová korelace selhává – nejčastěji na vodních plochách, opravena. Z opravených bodových mračen byl DMP vygenerován též v rastrové podobě ve formátu TIFF a rovněž jako stínovaný reliéf (*hillshade*) ve formátu TIFF. Následně poté proběhla klasifikace opravených bodových mračen a na závěr vyhledání hrubých chyb v bodových mračnech.

4.5.1 Tvorba

Digitální model povrchu ve všech podobách zmíněných výše byl tvořen v listech Státní mapy 1 : 5 000 (SM5). Jeden mapový list SM5 zaujímá území o rozloze 2,5×2 km (ČÚZK, 2023c). Od ZÚ byla poskytnuta vrstva všech bloků SM5 na území České republiky. Z této vrstvy byly iteračně výběrem podle umístění (*Select Layer By Location*) vybrány všechny

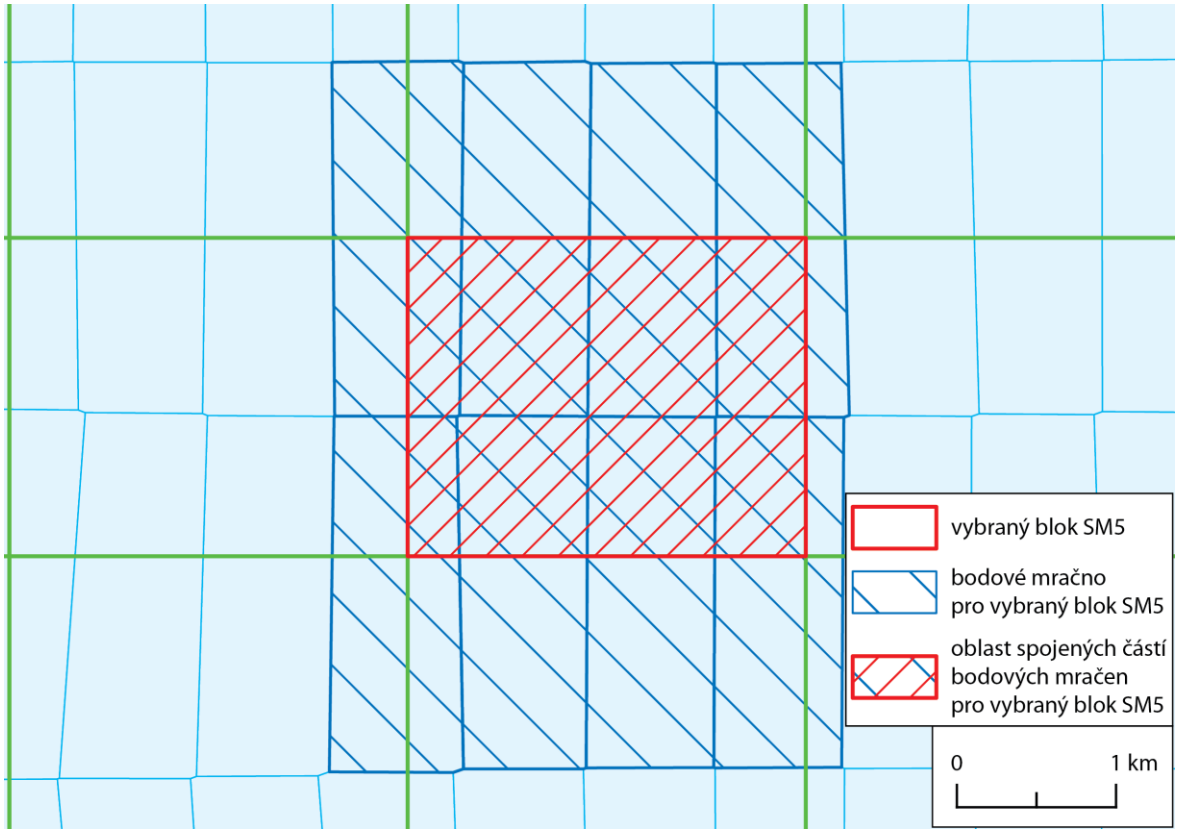
bloky SM5, které měly společnou polohu (*intersect*) vždy s daným blokem Thiessen polygonů a tyto bloky SM5 byly pomocí *Copy Features* uloženy do nového souboru. Byly tak vybrány a uloženy všechny bloky SM5 pro daný blok LMS (a okolí ze západu a východu bloku) jako jeden SHP soubor. Následně byla pomocí nástroje *Search Cursor* procházena atributová tabulka tohoto souboru SM5 bloků na základě atributu *SM5_NAME* (obsahuje jedinečný název každého listu SM5). Nástroj *Search Cursor* umožňuje v režimu „jen pro čtení“ přístup k záznamům v atributové tabulce třídy prvků (Esri, 2023e).

Nejdříve byl název jednoho SM5 bloku využit do výběru podle atributů (*Select Layer By Attribute*), do kterého vstupoval právě soubor SM5 bloků (jehož atributová tabulka byla procházena) a právě podle názvu byl vybrán odpovídající blok SM5, který byl uložen do samostatného SHP souboru.

Následně byl proveden výběr podle umístění, z LMS bloku Thiessen polygonů byly vybrány takové Thiessen polygony, které měly společnou polohu (*intersect*) se souborem jednoho bloku SM5 (vytvořeným v předchozím kroku). Bodová mračna ořezaná podle těchto vybraných Thiessen polygonů byla použita pro tvorbu DMP v daném bloku SM5.

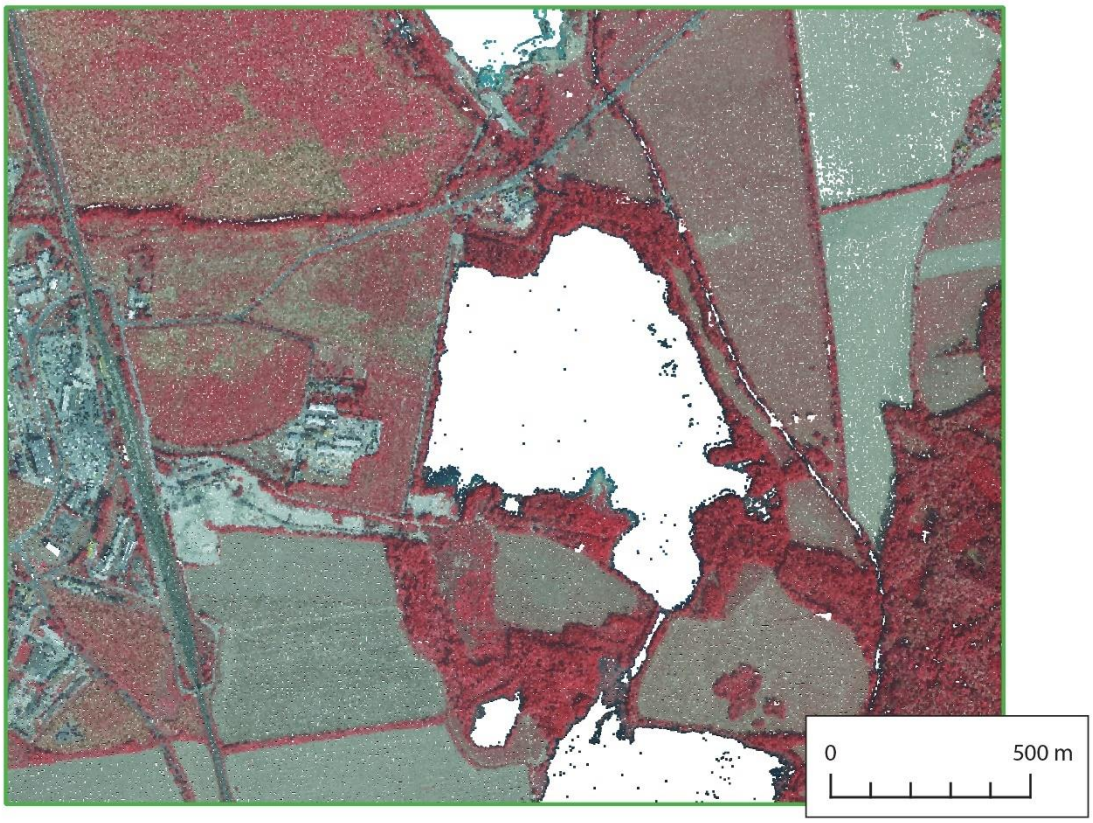
U těchto vybraných Thiessen polygonů byla opět nástrojem *Search Cursor* procházena atributová tabulka podle atributu *name* (název bodového mračna, které bylo daným Thiessen polygonem ořezáno). Pro každý blok SM5 byl založen textový soubor, do něhož byly zapisovány cesty na disku k ořezaným bodovým mračnům vyplňujícím (a přesahujícím) daný blok SM5 právě s využitím iterování přes atribut *name* vybraných Thiessen polygonů.

Závěrem této série kroků byl pro každý blok SM5 použit nástroj **lasclip** k jedné z nejdůležitějších operací. Jeho vstupem byl textový soubor z předchozího kroku a soubor odpovídajícího bloku SM5. Nástrojem **lasclip** byla načtena **ořezaná bodová mračna** pro daný blok SM5, **byla sloučena** a souborem bloku SM5 **oříznuta** (obr. 15).



Obr. 15 Spojení bodových mračen do bloku SM5.

Do nového souboru tak bylo uloženo **spojené bodové mračno ve formátu LAZ v bloku SM5**. Takto **vznikl prvotní digitální model povrchu** (obr. 16).



Obr. 16 Prvotní digitální model povrchu (jako bodové mračno) v bloku SM5 ZABR69.

4.5.2 Oprava a tvorba v rastrové podobě

Jak již bylo nastíněno, obrazová korelace selhává nejvíce na vodních plochách, a proto byla bodová mračna v těchto místech opravena. Pro tuto opravu byla jako zdroj využita již dříve zmíněná 3D vrstva všech vodních ploch a vodních toků na území České republiky od ZÚ.

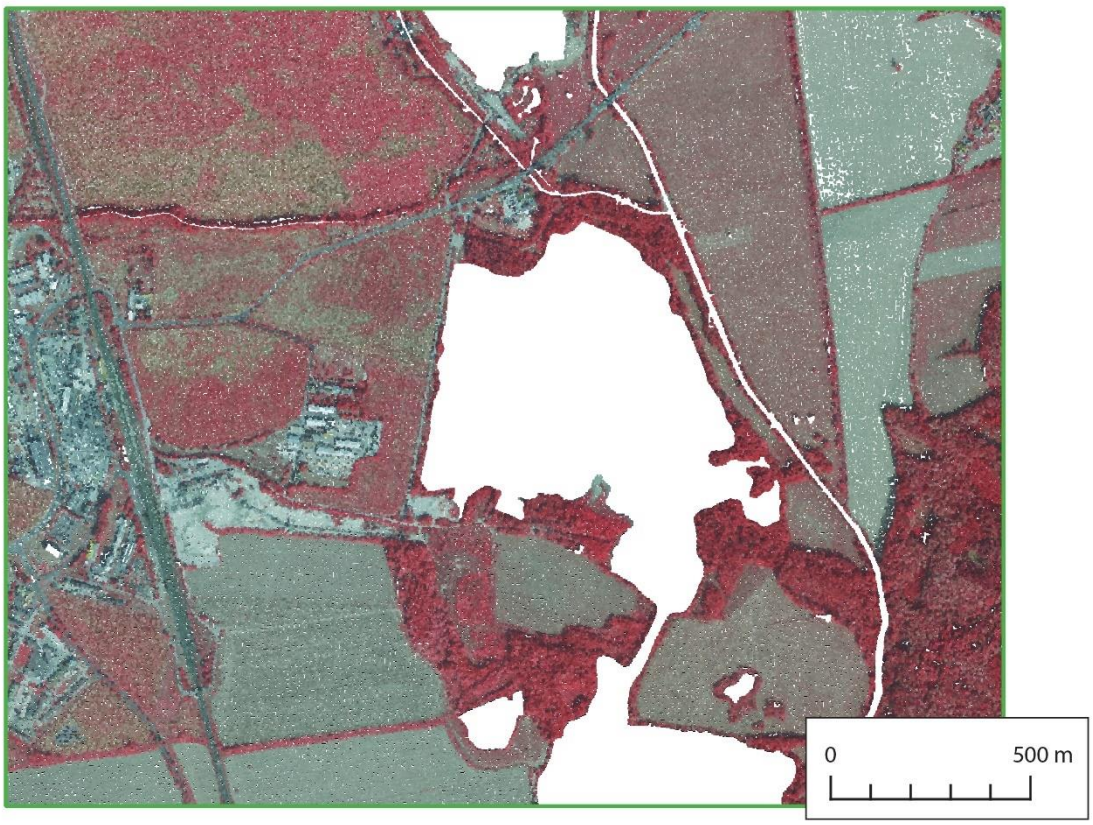
Nejdříve byla vrstva vodních ploch připravena do zájmových území zpracovávaných bloků LMS (a jejich okolí – vždy podle vrstvy bloků SM5). Pro každý tento blok byla zmíněná vrstva nástrojem *Clip* oříznuta tak, aby vznikl samostatný SHP soubor 3D vodních ploch pro daný blok. Dále byly pouze z této nově vzniklé vrstvy vod pro daný blok připravovány opět nástrojem *Clip* vrstvy 3D vodních ploch i pro jednotlivé bloky SM5. Tento postupný „dvoustupňový" ořez byl zvolen pro urychlení, neboť provádění ořezu pro každý blok SM5 z celorepublikové vrstvy vod by bylo časově náročnější. Ovšem vrstva vod již nevznikla pro všechny bloky SM5, nýbrž pouze pro skutečně použité bloky SM5. Nepoužité byly ty, které na západě i východě výrazně přesahovaly blok LMS a byly původně vybrány proto, že měly společnou polohu (*intersect*) s krajními Thiessen polygony. Tyto krajní Thiessen polygony ovšem ve svých krajích nepokrývaly rozsah krajních původních bodových mračen, a proto takto vzniklé bloky SM5 nebyly pro tvorbu DMP použity, a i zde byly vypuštěny.

Poté co byla připravena vrstva 3D vod pro všechny (použité) bloky SM5 ve všech blocích LMS, byla sestavena dvě polí polí (dvě trojrozměrná pole). Do prvního byly vloženy cesty k bodovým mračnům v SM5 blocích vzniklých v předchozí podkapitole a do druhého pak

cesty k vrstvám 3D vod v SM5 blocích. Obě trojrozměrná pole tedy obsahovala cesty k 2 420 souborům – vzájemně si odpovídajícím podle bloků SM5.

Tato pole byla současně procházena pomocí funkce *zip()*, vždy bylo operováno s jedním bodovým mračnem a s jedním souborem 3D vodních ploch ze stejného bloku SM5. Jelikož ne ve všech blocích SM5 se nachází vodní plocha či vodní tok, byla přítomnost vody ověřena nástrojem *Get Count*, kterým bylo zjištěno, zda počet záznamů v atributové tabulce vrstvy vod je nulový či nenulový. V bloku SM5, ve kterém se žádná voda nenacházela, nebylo na bodovém mračnu co opravovat, proto bylo toto bodové mračno pouze zkopírováno do výstupní složky a přejmenováno na finální název.

V blocích SM5 obsahujících vodu byla v několika následujících krocích provedena oprava bodového mračna. Nejprve byly nástrojem **lasclip** s využitím vrstvy vod ze vstupních bodových mračen vyříznuty/odebrány (ve většině) chybné body nacházející se právě uvnitř vodních ploch. Toto bezvodé bodové mračno bylo uloženo jako dočasný LAZ soubor *without_water* (obr. 17).

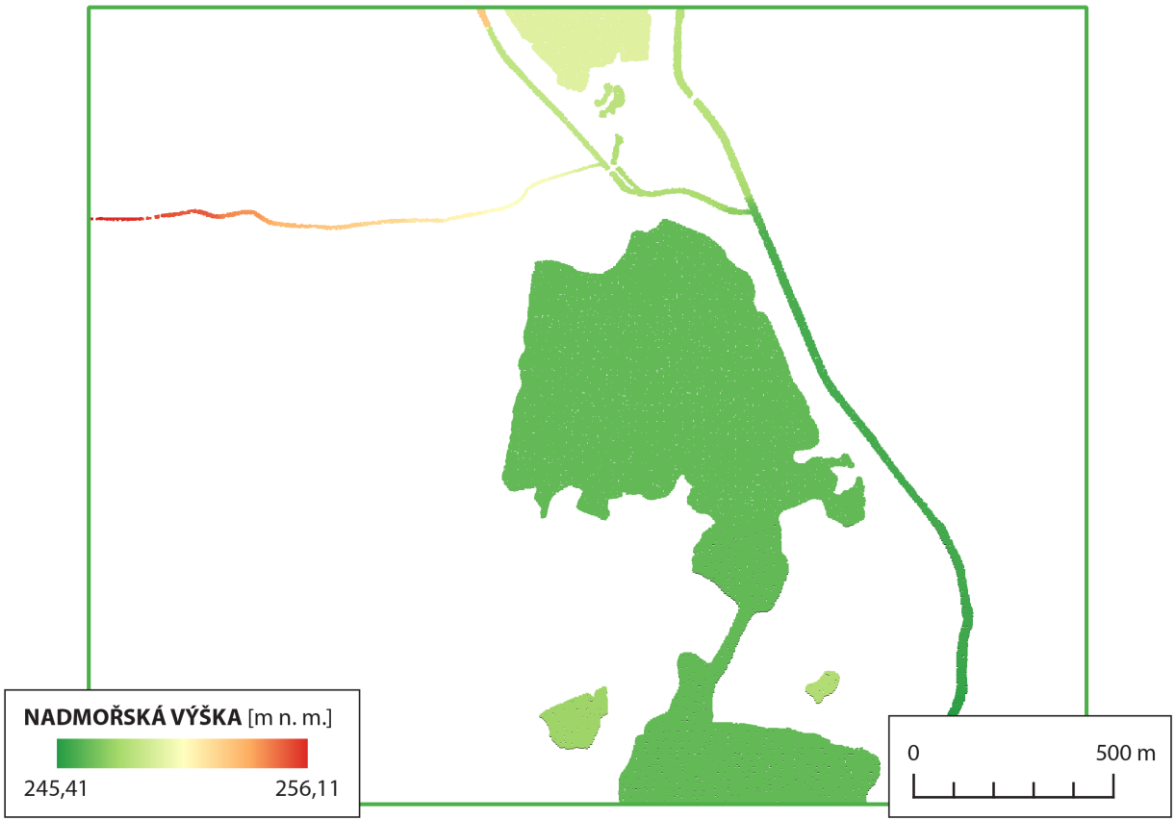


Obr. 17 Dočasné bodové mračno *without_water* v bloku SM5 ZABR69.

Dále byl použit nástroj **las2dem**, který se běžně využívá tak, že je ním načteno bodové mračno ve formátu LAS/LAZ, body jsou triangulovány do dočasné nepravidelné trojúhelníkové sítě (angl. triangulated irregular network (TIN)), která je pak rasterizována do digitálního modelu (reliéfu/povrchu) (Rapidlasso, 2023b). V této situaci byl ale tento nástroj využit způsobem, který není popsán v jeho dokumentaci. Vstupní vrstvou bylo bodové mračno *without_water* a pomocí argumentu *lakes* i vrstva 3D vod. Bylo nastaveno prostorové rozlišení na jeden metr a jako výstup nikoli rastrový model, nýbrž opět bodové mračno ve formátu LAZ. V mračnu *without_water* tak byl triangulován dočasný TIN – v místech (prázdných) vodních ploch podle vrstvy 3D vod, mimo vodní plochy ze vstupních

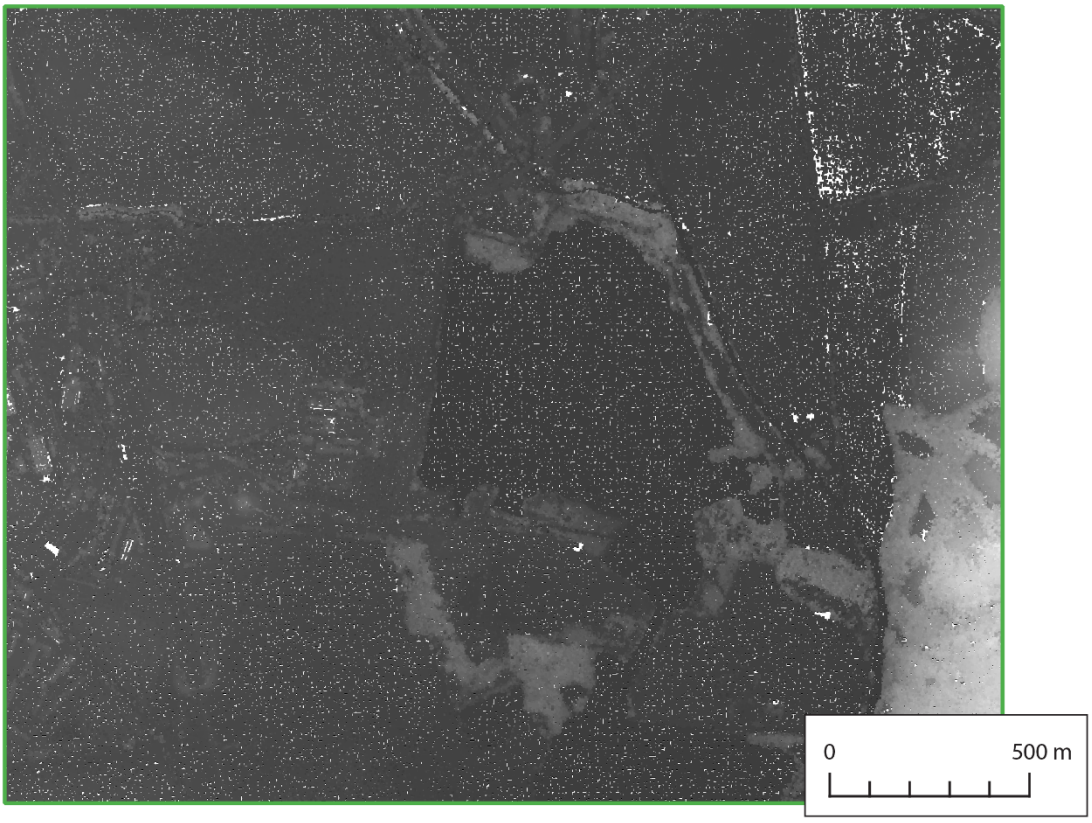
bodů. Z tohoto TINu bylo vytvořeno bodové mračno o zmíněném prostorovém rozlišení – tedy jeden bod na metr čtvereční – a bylo uloženo jako dočasný LAZ soubor *interpolated*.

Poté byl využit opět nástroj **lasclip** k vyříznutí a osamostatnění interpolovaných bodů v místech vodních ploch z bodového mračna *interpolated*. Mračno *interpolated* bylo tedy první vstupní vrstvou a druhou vstupní vrstvou byla opět vrstva 3D vod. Vyříznuté a osamostatněné body v místech vodních ploch byly uloženy jako dočasný LAZ soubor *water* (obr. 18).



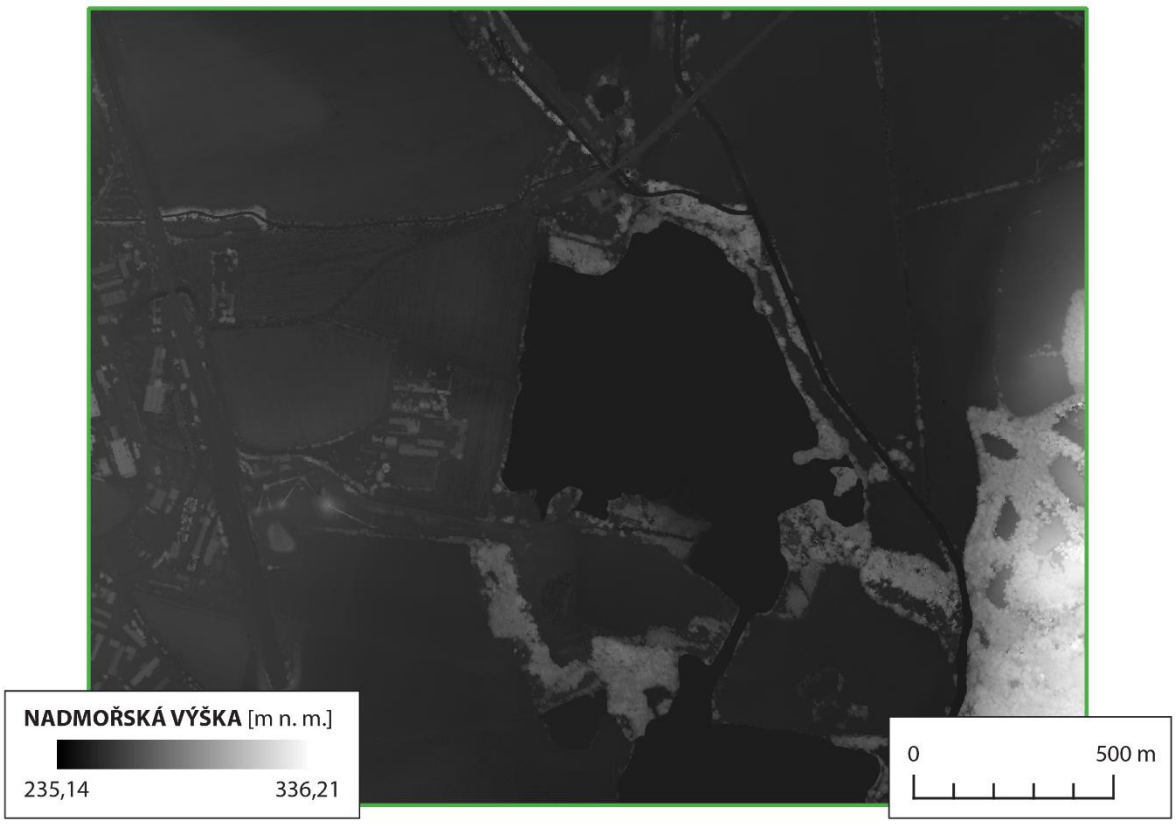
Obr. 18 Dočasné bodové mračno *water* v bloku SM5 ZABR69.

Za určitých podmínek ale bodové mračno *water* vůbec nevzniklo. Nevzniklo, když se v bloku SM5 sice nacházela vodní plocha či vodní tok, ale tyto nijak nezasahovaly do bodového mračna v daném SM5 bloku. Tyto případy nastávaly v některých krajních blocích SM5, které nebyly celé vyplněny bodovými mračny. V dalším postupu opravy bylo tedy nejdříve ověřeno, zda existuje bodové mračno *water*. Jestliže neexistovalo, pak bylo bodové mračno *without_water* (zde s neodpovídajícím označením, neboť žádná voda ze vstupního mračna ani nebyla odstraněna) pouze přejmenováno a vzniklo **finální bodové mračno ve formátu LAZ v daném bloku SM5**. Jestliže bodové mračno *water* existovalo, byl použit nástroj **lasmerge** ke sloučení bodového mračna *without_water* a právě bodového mračna *water*. Tím bylo získáno **finální opravené bodové mračno ve formátu LAZ v bloku SM5** (obr. 19). Poté už byla pouze odstraněna dočasná bodová mračna *without_water*, *interpolated* a *water*.



Obr. 19 Finální opravené bodové mračno v bloku SM5 ZABR69.

Po získání finálních bodových mračen byl ve všech blocích SM5 vygenerován DMP v rastrové podobě, konkrétně ve dvou různých rastrových podobách. Pomocí nástroje **las2dem** (nyní již použitého k účelu, který popisuje dokumentace), jehož vstupním souborem bylo finální opravené bodové mračno, jeho argumentu *elevation* a nastavení prostorového rozlišení na 1 m/px, byl vygenerován surový **finální digitální model povrchu ve formátu TIFF v bloku SM5** (obr. 20). Pro DMP ve formě stínovaného reliéfu (*hillshade*) byl v nástroji **las2dem** pouze změněn argument *elevation* na *hillshade* a tímto způsobem byl vyprodukován též **finální digitální model povrchu v podobě stínovaného reliéfu ve formátu TIFF v bloku SM5** (obr. 21).



Obr. 20 Finální digitální model povrchu v bloku SM5 ZABR69.

Obr. 21 Finální digitální model povrchu v podobě stínovaného reliéfu v bloku SM5 ZABR69.

4.6 Klasifikace bodových mračen

V kapitole 2.3 bylo zmíněno, že formát LAS je tvořen čtyřmi různými skupinami dat. Ve skupině *Point Data Records* jsou obsaženy právě i informace o klasifikaci bodů bodového mračna do klasifikačních tříd definovaných ASPRS potažmo OGC. Klasifikační třídy ještě záleží i na formátu skupiny dat *Point Data Records*, který se označuje číselně a nabývá hodnot od 0 do 10 (OGC, 2018). Bodová mračna použitá v této práci mají formát skupiny *Point Data Records* 0 nebo 2, proto je lze klasifikovat do klasifikačních tříd znázorněných na obrázku 22 (obr. 22).

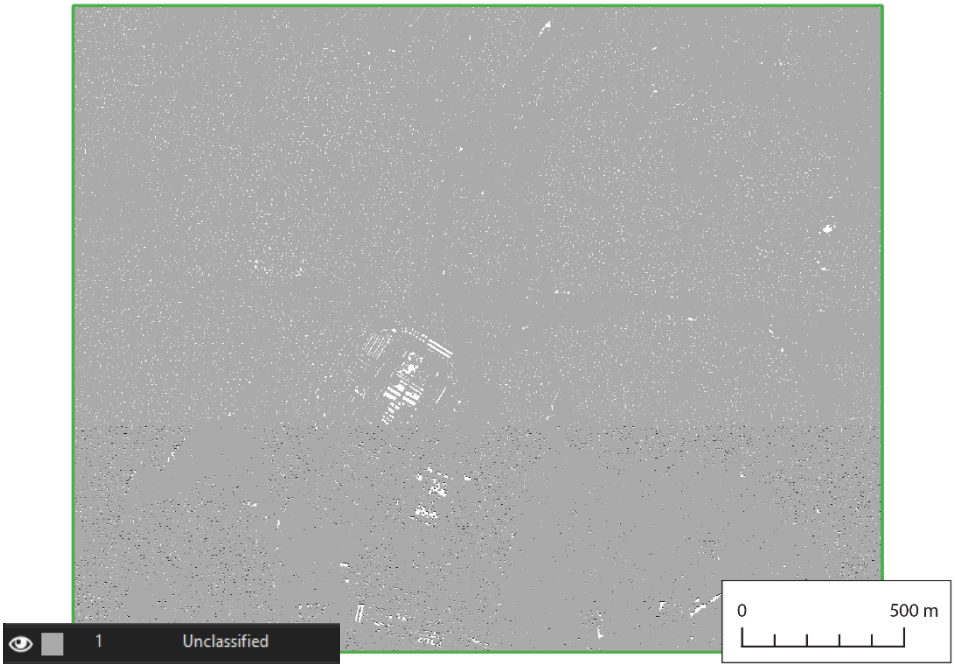
Table 9: ASPRS Standard LIDAR Point Classes (Point Data Record Format 0-5)

Classification Value (bits 0:4)	***Meaning***
0	Created, never classified
1	Unclassified[1]
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	Model Key-point (mass point)
9	Water
10	*Reserved for ASPRS Definition*
11	*Reserved for ASPRS Definition*
12	Overlap Points[2]
13-31	*Reserved for ASPRS Definition*

Obr. 22 Klasifikační třídy formátu LAS pro formát skupiny *Point Data Records* 0–5 (OGC, 2018).

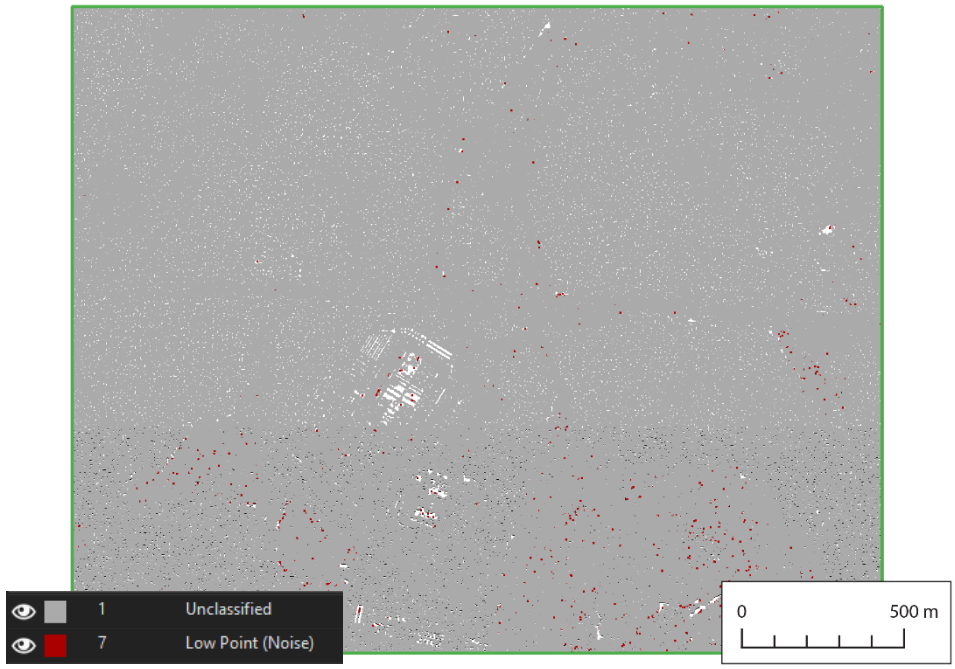
Ze všeho nejdříve byla připravena externí data vstupující do procesu klasifikace. Těmito daty byly vektorové polygonové vrstvy mostů, budov a lesů (popsané blíže v podkapitole Použitá data v rámci kapitoly 3) poskytnuté od ZÚ pro území celé České republiky. Totožně jako v kapitole 4.5.2 byla tato data ořezána do bloků SM5 postupně ve dvou stupních. Nejprve byly zmíněné vrstvy nástrojem *Clip* ořezány na území bloků LMS (a jejich okolí – vždy podle vrstvy bloků SM5) a vznikl takto vždy soubor SHP pro mosty, budovy a lesy v popsaném území. Tyto SHP soubory byly uloženy do příslušných dočasných složek (též označených jako mosty, budovy a lesy), které byly pro tento účel vytvořeny. A až následně byly z těchto na území bloku LMS ořezaných vrstev připraveny opět nástrojem *Clip* všechny tři zmíněné vrstvy pro území každého bloku SM5 ve všech blocích LMS jako samostatné SHP soubory uložené opět v příslušných dočasných složkách.

Po přípravě pomocných dat bylo přistoupeno k samotné klasifikaci bodových mračen v sérii několika kroků. V prvním kroku byla všem bodům pomocí nástroje **las2las** a jeho argumentu *set_classification* nastavena klasifikační třída 1 – *Unclassified* a tím byla klasifikace v bodovém mračnu založena. Třídu 0 mají totiž ty body, které nebyly klasifikací dotčeny. Výstupem bylo bodové mračno označené jako *class_one* (obr. 23).



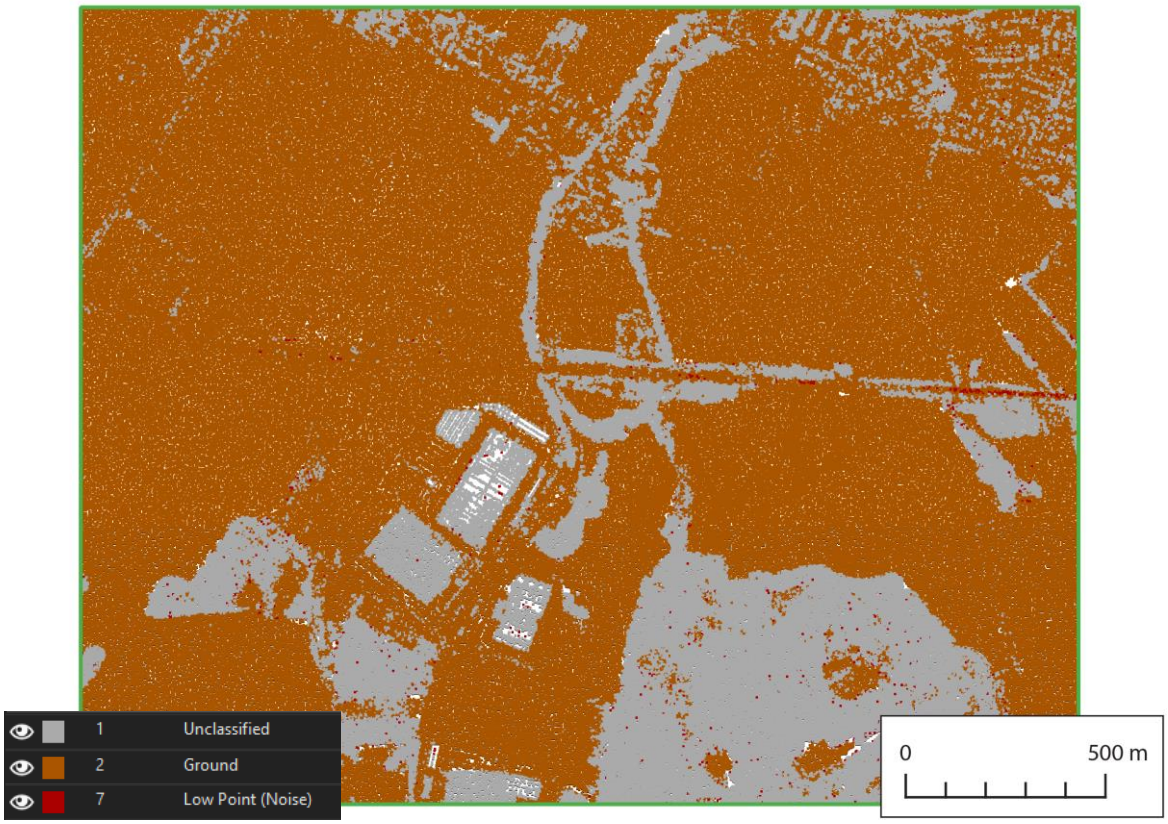
Obr. 23 Dočasné bodové mračno *class_one* pro klasifikaci v bloku SM5 HERM10.

Druhým krokem bylo použití nástroje **lasnoise**, který v souborech LAS/LAZ/BIN/ASCII vyhledá izolované body (tvořící šum) podle zadaných či výchozích kritérií a buď je může označit právě jako body šumu, či je přímo z bodového mračna odstranit (Rapidlasso, 2023i). Vstupní vrstvou bylo bodové mračno *class_one* a nastavení nástroje bylo ponecháno výchozí. Při výchozím nastavení je bod jako izolovaný označen tehdy, když v jeho okolí 4×4×4 m je méně než pět jiných bodů. Samotný posuzovaný bod je uprostřed definovaného okolí (Rapidlasso, 2023i). Takto vyhledané izolované body byly klasifikovány do klasifikační třídy 7 – *Low Point (noise)* a výstupní bodové mračno bylo označeno jako *noise* (obr. 24).



Obr. 24 Dočasné bodové mračno *noise* pro klasifikaci v bloku SM5 HERM10.

Jako třetí krok klasifikace bylo nalezení a oklasifikování bodů ležících na povrchu země (bodů, které by měl DMP shodné s digitálním modelem reliéfu (DMR)). Pro tento účel byl použit nástroj **lasheight** primárně sloužící k výpočtu výšky každého bodu bodového mračna nad zemí. Při výchozím nastavení nástroje se předpokládá, že body na povrchu země již byly klasifikovány, aby z nich mohl být vytvořen TIN, od kterého budou počítány výšky bodů. Nebo mohou být body tvořící povrch země dodány v externím LAS/CSV souboru. Nástroj načte bodové mračno ve formátu LAS/LAZ/ASCII, trianguluje body na povrchu země do TIN a vypočítá výšku každého bodu vzhledem k vytvořenému TINu. Případně lze pomocí argumentu *replace_z* nahradit nadmořskou výšku daného bodu vypočítanou výškou od země (body zemského povrchu budou mít souřadnici Z nulovou). Vypočítanou výšku lze využít také právě ke klasifikaci pomocí argumentů *classify_below*, *classify_above* a *classify_between*, kde jsou nastaveny výšky případně rozsahy výšek, ve kterých jsou body klasifikovány do určených klasifikačních tříd (Rapidlasso, 2023g). Jako vstupní vrstva bylo použito bodové mračno *noise* a DMR 5G v území zpracovávaného bloku SM5 v podobě bodového mračna ve formátu LAZ jako externí vrstva s body tvořícími povrch země. Byly použity argumenty *replace_z*, *store_in_user_data* a *all_ground_points* (aby z vrstvy DMR 5G byly k triangulaci TIN použity všechny body, a nikoliv pouze výběr) a především klasifikační argumenty ke klasifikaci podle výšky. Bodům s výškou menší než -4 m byla argumentem *classify_below* přiřazena klasifikační třída 7 – *Low Point (noise)*, bodům s výškou větší než 115 m byla argumentem *classify_above* přiřazena klasifikační třída 12 – *Overlap Points* (ovšem bráno jako opak *Low Point*, tedy tzv. *High Point*) a konečně **bodům s výškou mezi -3 a 3 m byla** argumentem *classify_between* **přiřazena klasifikační třída 2 – *Ground***. Výstupní bodové mračno bylo uloženo jako *ground* (obr. 25).

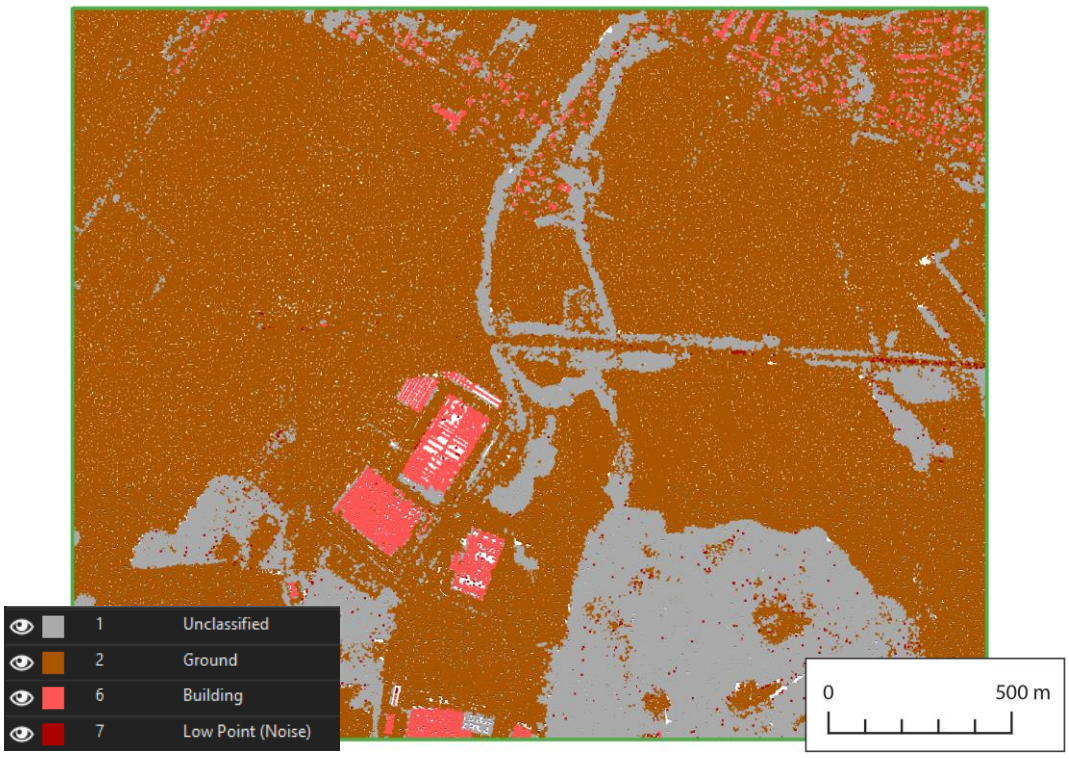


Obr. 25 Dočasné bodové mračno *ground* pro klasifikaci v bloku SM5 HERM10.

Následně byly v klasifikaci využity tři na začátku připravené vrstvy – mostů, budov a lesů. U všech těchto vrstev byl pro klasifikaci použit nástroj **lasclip** a jeho klasifikační argument *classify*.

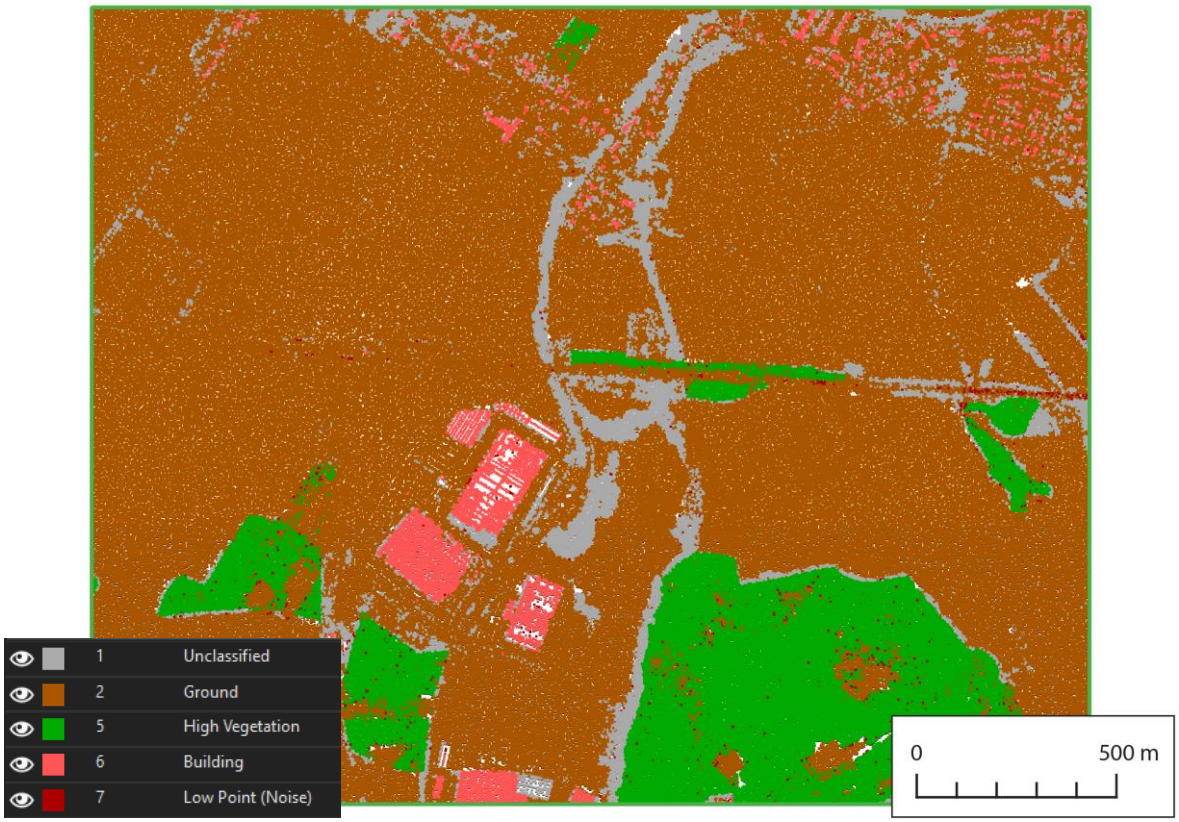
Nejprve byla ze zmíněných vrstev využita vrstva mostů. Body v bodových mračnech nacházející se na mostech byly pomocí vrstvy mostů klasifikovány též jako body na povrchu země. Jedná se totiž také o body na povrchu země, ale jelikož se mosty z podstaty nenachází v DMR, kterým byla klasifikována třída *ground* v předchozím kroku, byly tímto krokem do třídy *ground* doplněny. Před samotným použitím nástroje **lasclip** bylo nástrojem *Get Count* ověřeno, zda se v daném bloku SM5 vůbec nějaké mosty nacházejí. Jestliže se v bloku SM5 žádné mosty nenacházely, bylo pouze bodové mračno *ground* přejmenováno na *bridges*. Jestliže blok SM5 mosty obsahoval, byl použit nástroj **lasclip** následujícím způsobem. Vstupní vrstvou nástroje **lasclip** bylo bodové mračno *ground* a vrstva mostů v daném bloku SM5, argumenty byly použity *classify* (s hodnotou 2), *interior* (aby byly klasifikovány body nacházející se uvnitř vložené SHP vrstvy mostů), *quiet* (čistě z důvodů nevypisování dlouhých komentářů do konzole při běhu skriptu) a též důležitý argument *ignore_class* (nímž byly nastaveny klasifikační třídy, jejichž body byly při současné klasifikaci již ignorovány, jednalo se o třídy 2, 7 a 12). Výsledkem byly **body nacházející se na mostech klasifikované do klasifikační třídy 2 – *Ground***. Toto bodové mračno bylo uloženo jako *bridges*.

Stejnou logikou bylo postupováno i u klasifikace budov. Body nacházející se na budovách byly s využitím vrstvy budov klasifikovány jako budovy. Opět bylo nástrojem *Get Count* ověřeno, zda se v bloku SM5 nacházejí budovy, když ne, bylo bodové mračno *bridges* přejmenováno na *buildings*. Jestliže se budovy v bloku SM5 nacházely, byly klasifikovány nástrojem **lasclip** stejně jako mosty v předchozím případě. Pouze argument *classify* měl z logických důvodů nastavenu hodnotu 6. Výsledkem byly **body nacházející se na budovách klasifikované do klasifikační třídy 6 – *Building***. Bodové mračno bylo uloženo jako *buildings* (obr. 26).



Obr. 26 Dočasné bodové mračno *buildings* pro klasifikaci v bloku SM5 HERM10.

První část klasifikace vegetace proběhla stejně jako u mostů a budov. Body nacházející se v lesích byly s využitím vrstvy lesů klasifikovány jako vegetace. Pro bloky SM5, ve kterých se nevyskytovala žádná vegetace podle vrstvy lesů, bylo bodové mračno *buildings* přejmenováno na *forests*. V blocích SM5, kde vegetace podle vrstvy lesů byla přítomna, byla tato nástrojem **lasclip** klasifikována opět téměř totožně jako v předchozích případech. Argument *classify* nabýval hodnoty 5 a argumentem *ignore_class* byly ignorovány klasifikační třídy 2, 6, 7 a 12. Výstupem byly **body nacházející se v lesích klasifikované do klasifikační třídy 5 – *High Vegetation***. Bodové mračno bylo uloženo jako *forests* (obr. 27).



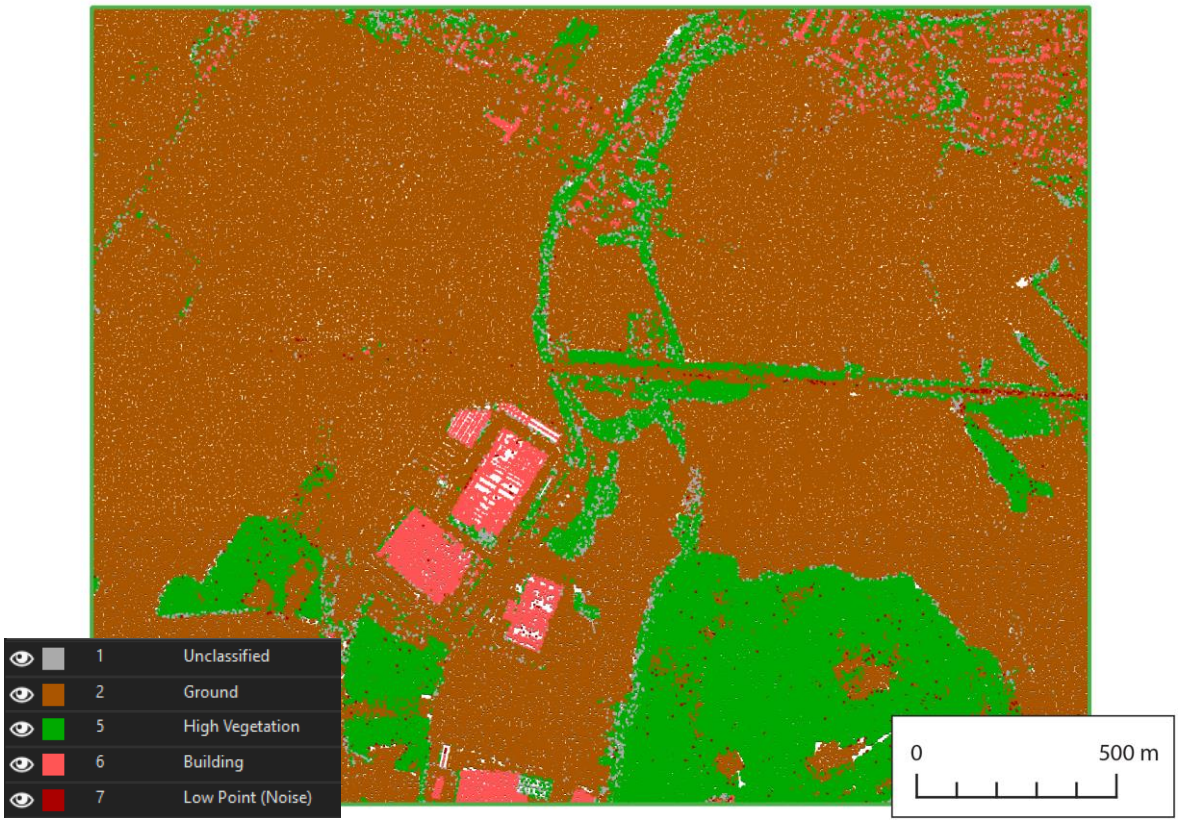
Obr. 27 Dočasné bodové mračno *forests* pro klasifikaci v bloku SM5 HERM10.

Druhá část klasifikace vegetace proběhla přímo nástrojem **lasclassify** určeným pro klasifikaci budov (třída 6) a vysoké vegetace (třída 5) v souborech LAS/LAZ. Nástroj hledá sousední body (výchozí nastavení je oblast 2×2×2 m), které jsou alespoň dva metry nad zemí a tvoří buď rovinné oblasti (střechy) nebo naopak členité oblasti (stromy). Od jaké úrovně rovinatosti budou body klasifikovány jako budova a od jaké úrovně členitosti budou klasifikovány jako vysoká vegetace lze ovlivnit nastavením argumentů *planar* a *rugged* (Rapidlasso, 2023e). Jelikož bylo požadováno tímto nástrojem najít a oklasifikovat pouze do této doby neoklasifikovanou vegetaci a budovy nikoliv, byly dostupné argumenty nastaveny výhradně na rozpoznávání vegetace. Argument *planar* byl nastaven na 0,001, aby limitně žádná rovinatost bodů nebyla dostatečná k oklasifikování těchto bodů jako budov, argument *rugged* byl nastaven na 0,05, aby byla dobře rozpoznávána vegetace, dále byly použity argumenty *no_gutters* k „nedokončování střech podél jejich potenciálních okrajů“, *small_trees*, aby nebyly z klasifikace vegetace odebírány malé stromy, a nakonec pochopitelně argument *ignore_class*, díky kterému byly ignorovány všechny předchozí

již klasifikované třídy, tedy 2, 5, 6, 7 a 12. Výsledkem byly **body vegetace klasifikované do klasifikační třídy 5 – *High Vegetation***. Bodové mračno bylo uloženo jako *vegetation*.

Tímto dosavadním postupem klasifikace byly oklasifikovány již téměř všechny body. Některé body zůstaly neklasifikované – klasifikační třída 1 – *Unclassified*, ty byly považovány již pouze za další šum, který nevyhověl žádné klasifikační podmínce.

Jelikož byla v tento moment **klasifikace bodového mračna v bloku SM5 hotova**, bylo bodové mračno *vegetation* již pouze přejmenováno na *FINAL*, čímž bylo uloženo **finální oklasifikované bodové mračno pro blok SM5 do třech** hlavních **klasifikačních tříd – země**, **budovy** a (vysoká) **vegetace** (obr. 28).



Obr. 28 Finální oklasifikované bodové mračno bloku SM5 HERM10.

Závěrem již byly pouze odstraněny meziprodukty *class_one*, *noise*, *ground*, *bridges*, *buildings* a *forests*. Meziprodukt *vegetation* již neexistoval, neboť byl dříve přejmenován na *FINAL*. Úplně nakonec byly odstraněny též dočasné složky pro vrstvy mostů, budov a lesů v jednotlivých blocích SM5.

4.7 Detekce hrubých chyb v bodových mračnech

Hrubými chybami v bodových mračnech byly především díry primárně nevyskytující se na vodních plochách. Tyto díry byly automatizovaně vyhledány, na jejich místech vznikla vektorová polygonová vrstva, která byla posléze uložena do souboru s formátem SHP.

Vyhledání děr proběhlo především s vícenásobným využitím nástroje **lasboundary** a jeho argumentů. Argument *holes* umožňuje vyhledat a ohraničit v bodovém mračnu vnitřní díry, při použití argumentu *merged* při ohraničování více vstupních bodových mračen jsou ohraničení jednotlivých vstupních bodových mračen sloučena do jednoho

velkého ohraničení, které je pak výstupním souborem. Dále pomocí hodnoty argumentu *concavity* lze ovládat co bude považováno za díru a co nikoliv. Například hodnota 50 argumentu *concavity* (která je též výchozí) znamená, že prázdná místa s alespoň jednou vzdáleností alespoň 50 m budou považována buď za vnější (tímto způsobem je určena hranice bodového mračna) či za vnitřní (právě jako součást děr uvnitř bodových mračen) při současném použití argumentu *holes* (Rapidlasso, 2023d).

Jelikož nelze v jednom kroku nástrojem **lasboundary** vyhledat a ohraničit v bodovém mračnu pouze vnitřní díry, protože jsou vždy současně ohraničena i celá vstupní bodová mračna (argumentem *merged* sloučená do jednoho), bylo ohraničení vnitřních děr provedeno v následující sérii kroků.

Vstupními daty do nástroje **lasboundary** byla všechna bodová mračna vždy z jednoho bloku LMS. Při jeho prvním běhu byly použity výše zmíněné argumenty *holes*, *concavity* s hodnotou 30 (určeno empiricky a expertním odhadem) a argument *merged*. Výsledkem byla vrstva nazvaná *lasboundary_holes*, neboť obsahovala ohraničení všech vstupních bodových mračen v blocích SM5 sloučená do jednoho – jednalo se tedy o ohraničení bodových mračen v rozsahu bloku LMS – a také ohraničení vnitřních děr, které byly takto vyhledány. Jako vnitřní díry byla vykonaným postupem uložena i místa nízké hustoty bodů v bodových mračnech. Konkrétně podle hodnoty argumentu *concavity* (30) byla jako díry v bodových mračnech označena i všechna místa, ve kterých byly od sebe jednotlivé body vzdáleny 30 a více metrů. V atributové tabulce této vrstvy bylo každé nalezené ohraničení jako samostatný záznam.

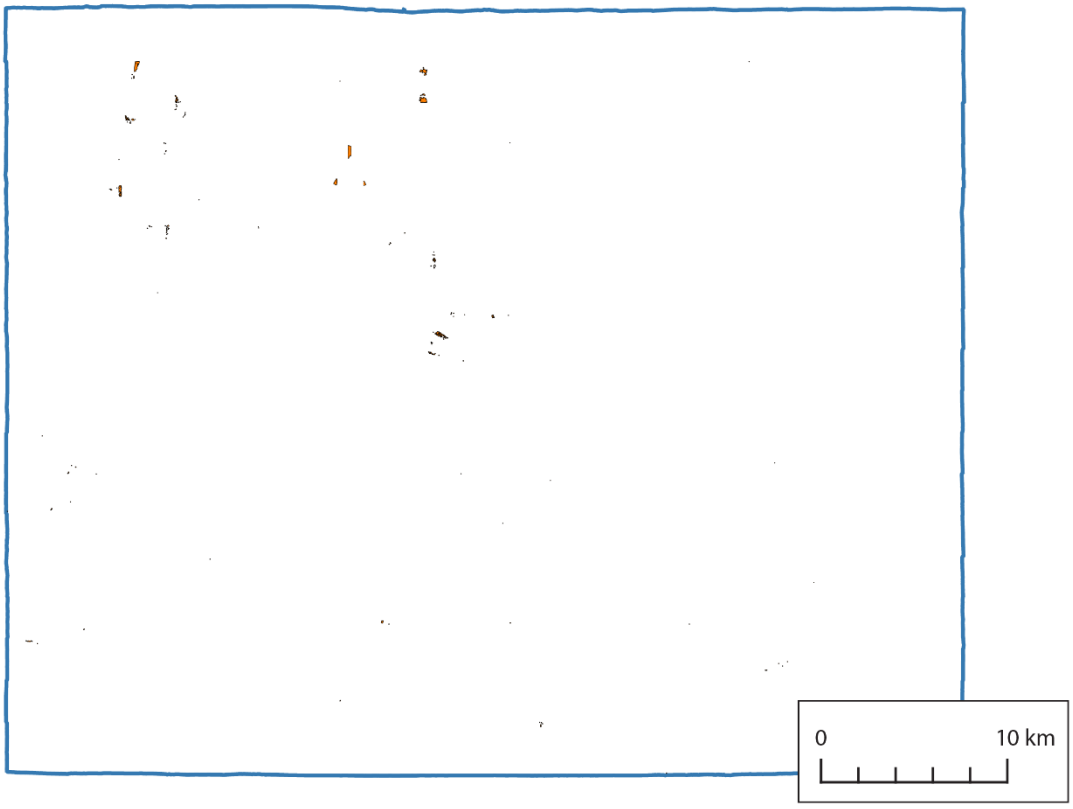
Při druhém spuštění nástroje **lasboundary** byl využit pouze argument *merged*, čímž byla vytvořena pouze ohraničení všech vstupních bodových mračen v blocích SM5 sloučená do jednoho. Tato dočasná vrstva byla nazvána *lasboundary*, poněvadž obsahovala pouze popsané ohraničení.

Obě takto připravené vrstvy vstoupily do výběru podle umístění (*Select Layer By Location*). Z vrstvy *lasboundary_holes* byly vybrány takové prvky, které nebyly uvnitř (*within*) vrstvy *lasboundary*. Tuto podmínku splnil vždy pouze jediný záznam, a sice právě spojené ohraničení všech bloků SM5 v rámci bloku LMS.

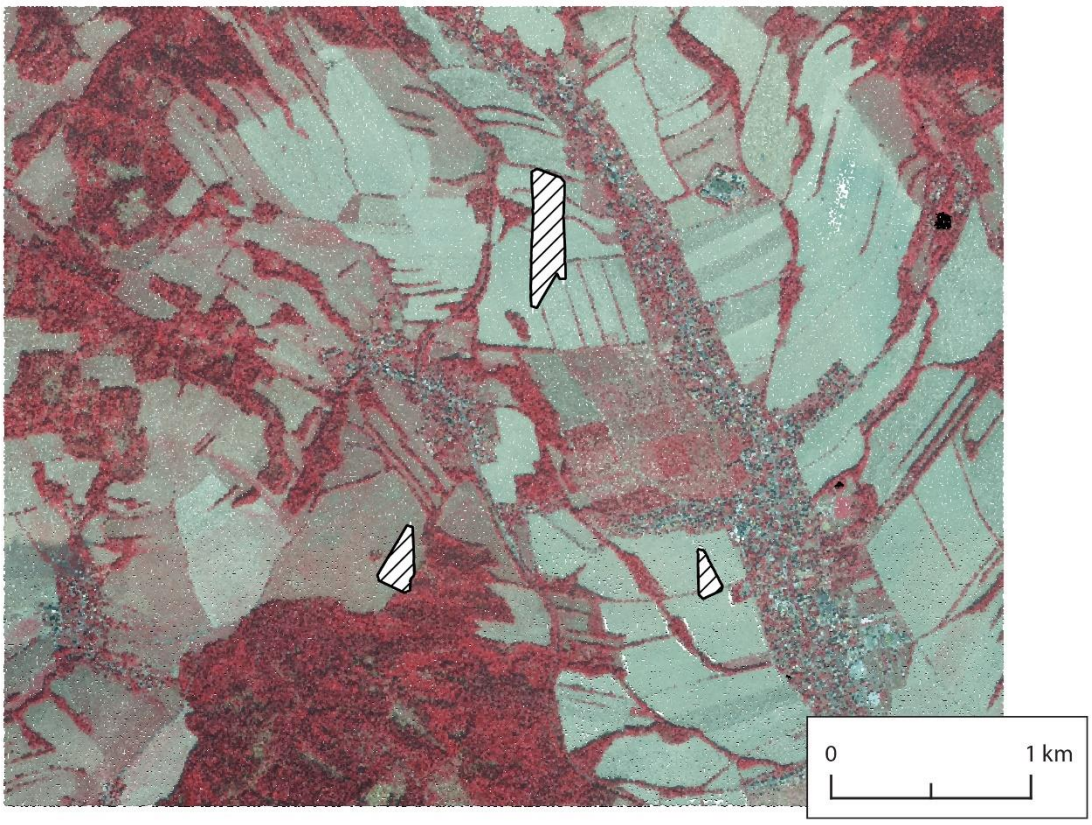
Vybraný záznam odpovídající spojenému ohraničení všech bloků SM5 v rámci bloku LMS byl z vrstvy *lasboundary_holes* odstraněn nástrojem *Delete Features* z knihovny ArcPy. Nástrojem *Delete* byla odstraněna rovněž i dočasná vrstva *lasboundary*.

Vrstva *lasboundary_holes* už obsahovala pouze výsledné vnitřní díry, byla tak v rámci finalizace přejmenována jen na *holes*. Protože takto vzniklý SHP soubor neobsahoval žádné informace o projekci, byla mu nástrojem *Define Projection* přiřazena projekce S-JTSK v podobě S-JTSK/Krovak East North; EPSG: 5514.

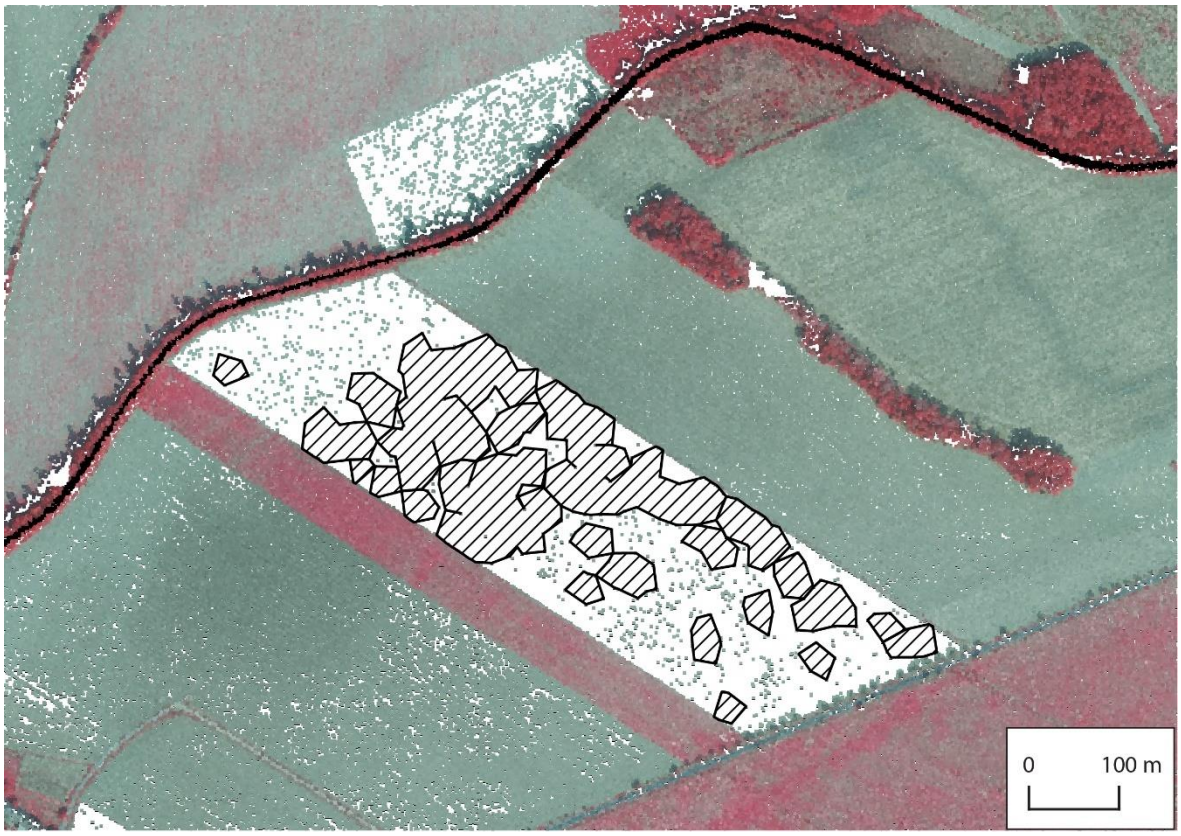
Provedenými kroky byla detekována **finální místa hrubých chyb v bodových mračnech pro každý blok LMS** (obr. 29). Jedná se o **místa děr primárně nezpůsobených vodními plochami** (obr. 30) **a místa nízké hustoty bodů** (obr. 31).



Obr. 29 Místa hrubých chyb v bodových mračnech v bloku LMS.



Obr. 30 Díry v bodových mračnech primárně nezpůsobené vodními plochami (území 4 bloků SM5).



Obr. 31 Díry způsobené oblastmi s nízkou hustotou bodů.

4.8 Tvorba DMP v programu Agisoft Metashape

Do prostředí programu Agisoft Metashape bylo nahráno všech 39 CIR snímků určených v této práci výhradně pro účel vytvoření DMP v tomto programu. Po nahrání snímků byly v okně *Camera Calibration* nastaveny prvky vnitřní orientace, tedy údaje charakterizující kameru, která použité snímky pořídila. Konkrétní hodnoty velikosti pixelu (*pixel size)*, ohniskové vzdálenosti (*focal length*) a souřadnice hlavního bodu (*principal point*) byly zjištěny z kalibračního protokolu použité kamery, který byl poskytnut od ZÚ. Podle kalibračního protokolu byla velikost pixelu nastavena 0,0039×0,0039 mm a ohnisková vzdálenost na 92 mm. Ohnisková vzdálenost byla poté automaticky přepočítána na pixely do parametru *f* na 23 589,743 px. Souřadnice hlavního bodu byly též dle protokolu ponechány nulové. Hodnoty z kalibračního protokolu byly považovány za naprosto přesné, proto byly parametry *f* (ohnisková vzdálenost v pixelech) a *cx*; *cy* (souřadnice hlavního bodu) nastaveny jako fixní, aby nebyly při pozdějším „zarovnání“ snímků (*Align Photos*) upravovány.

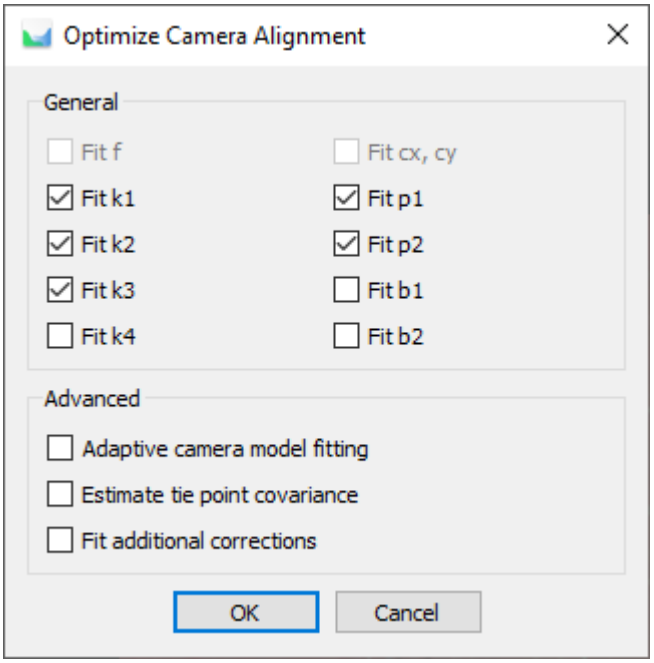
Po nastavení prvků vnitřní orientace byly přes *Import Reference* nahrány prvky vnější orientace snímků (dodané spolu se snímky od ZÚ), tedy informace o poloze kamery (středu optického systému) pro každý snímek (souřadnice X, Y a H) a úhly rotace každého snímku (ω, φ a κ).

Po nastavení zmíněných parametrů bylo přistoupeno k prvnímu kroku pracovního postupu, kterým je zarovnání snímků (*Align Photos*). V tomto kroku jsou nahrané snímky umístěny do prostoru podle souřadnic použitého souřadnicového systému a v místech překryvů snímků jsou nalezeny spojovací body (*tie points*), ze kterých je vytvořeno tzv. řídké bodové mračno (*sparse point cloud*). V dialogu *Align Photos* byla zvolena možnost *Generic Preselection*, díky které byly spojovací body pro každý snímek hledány pouze

v překrývajících se oblastech daného snímku s jeho sousedními snímky (Agisoft, 2019). Hodnoty parametrů *Key Point Limit* a *Tie Point Limit* byly ponechány ve výchozím nastavení, tedy 40 000 resp. 4 000.

Po vytvoření řídkého bodového mračna byly do projektu nahrány dříve vytvořené vlícovací body. Tyto body byly připraveny odměřením z Ortofota České republiky (WMS služba od ČÚZK) v prostředí ArcGIS Pro, kdy zaměřeny byly na Ortofotu i na použitých snímcích dobře viditelné poklopy kanálů. Souřadnice X a Y vlícovacích bodů byly vypočteny nástrojem *Calculate Geometry Attributes* a souřadnice Z byla nástrojem *Extract Values To Points* převzata z DMR 5G. Do programu Agisoft Metashape byly vlícovací body nahrány opět pomocí *Import Reference* a na řídkém bodovém mračnu byla (přibližná) poloha vyznačena značkami (*markers*). S využitím *Filter Photos By Markers* byly pro každý vlícovací bod vybrány vždy jen ty snímky, na kterých se daný bod nacházel. Na všech těchto vybraných snímcích byl vlícovací bod zaměřen s co největší přesností. Takto byly všechny body zaměřeny na všech jim odpovídajících snímcích.

Následně byly zpřesněny prvky vnitřní orientace na základě zaměřených vlícovacích bodů nástrojem *Optimize Camera Alignment*. V dialogu umožňujícím nastavit tento proces bylo ponecháno výchozí nastavení (obr. 32). Parametry *f* a *cx*; *cy* (popsané výše) nebylo možné vybrat, neboť byly dříve nastaveny jako fixní. K optimalizaci tak byl ponechán výběr parametrů *k1*, *k2* a *k3* (koeficienty radiální distorze) a *p1* a *p2* (koeficienty tangenciální distorze).

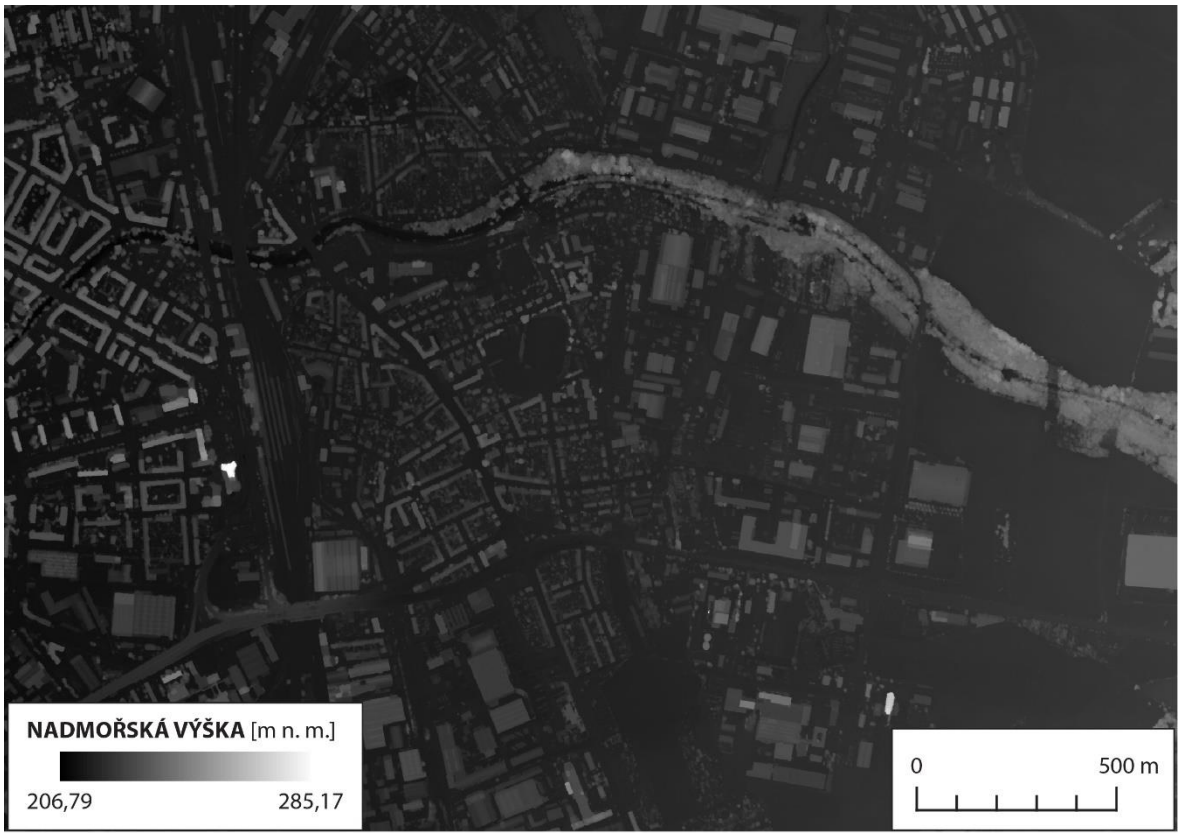


Obr. 32 Dialog *Optimize Camera Alignment* v prostředí programu Agisoft Metashape.

Následně už bylo přistoupeno k vytvoření hustého bodové mračna (*dense point cloud*), tedy k **obrazové korelaci** vstupních **CIR snímků**. V dialogu tohoto nástroje *Build Point Cloud* byla nastavena kvalita na vysokou (*High*) a *Depth filtering* na režim *Moderate*. *Depth filtering* slouží k nalezení a odstranění odlehlých bodů (angl. outliers). Je-li požadováno správně rozlišit drobné detaily, je použit režim *Mild*, aby nebyly vytříděny a odstraněny důležité prvky jako odlehlé. Jestliže území neobsahuje významné drobné detaily, je zvolen režim *Aggressive*, kterým je vytříděna a odstraněna většina odlehlých hodnot. Režim *Moderate* je kompromisem mezi oběma předchozími režimy (Agisoft, 2019). Výstupem bylo

finální bodové mračno vytvořené z CIR snímků ve formátu LAS s hustotou **12,33 bodu na metr čtvereční**.

V prostředí Agisoft Metashape byl z finálního bodového mračna vytvořen též DMP nástrojem *Build DEM*. V dialogu byla nastavena tvorba DMP právě z finálního hustého bodového mračna a metoda interpolace byla ponechána podle výchozího nastavení na *Enabled*. V režimu *Enabled* je DMP vygenerován pro všechny oblasti, které jsou viditelné alespoň na jednom snímku (Agisoft, 2019). Prostorové rozlišení bylo nastaveno na 0,33 m/px. Výstupem byl **finální DMP vytvořený z bodového mračna ve formátu TIFF** o zmíněném prostorovém rozlišení – **0,33 m/px** (obr. 33).



Obr. 33 Finální DMP vytvořený v programu Agisoft Metashape.

Na závěr byl nástrojem **blast2dem**, který se od dříve popsaného **las2dem** liší pouze tím, že zvládne pracovat s bodovými mračny o miliardách bodů (Rapidlasso, 2023a), vygenerován **finální DMP v podobě stínovaného reliéfu ve formátu TIFF** o stejném prostorovém rozlišení, tedy rovněž **0,33 m/px** (obr. 34).

Obr. 34 Finální DMP z programu Agisoft Metashape ve formě stínovaného reliéfu.

5 VÝSLEDKY

Výsledkem práce **jsou** mimo jiné především **skripty** v programovacím jazyce **Python** využívající především programový balík pro práci s bodovými mračny **LAStools** a nástroje knihovny **ArcPy**. Funkcionalita těchto skriptů, která byla popsána v kapitole Vlastní řešení, vedla k vygenerování výsledných produktů. Do funkcionality skriptů byly začleněny také výsledné verze metodik a postupů, neboť pouze díky nim mohly být výsledné produkty vytvořeny. Samotné skripty jsou uvedeny v přílohách práce.

Vedlejším **výsledkem** práce je také zpracovaný **DMP** v prostředí programu **Agisoft Metashape**, v němž byl absolvován kompletní proces tvorby DMP z LMS, tedy včetně samotné obrazové korelace.

Samotné výsledné produkty včetně popisů výsledných metodik a postupů byly většinou představeny již v rámci kapitoly Vlastní řešení. Kapitola Výsledky obsahuje jejich shrnutí či náhled jinou optikou.

5.1 Pravidla pro spojení bodových mračen v překryvech

V rámci nalezení postupu pro samotnou tvorbu DMP jakožto hlavního cíle byly vyřešeny i některé vedlejší cíle. Byla zde stanovena pravidla pro spojení bodových mračen v překryvových pásech jednotlivých stereomodelů a podle stanovených pravidel bylo toto spojení provedeno. Jak je zmíněno v podkapitole 4.4.4, byl navržen a vyzkoušen i **alternativní postup** s vizí kvalitnějšího výsledného DMP, tato **vize byla vyvrácena**. DMP ve formě stínovaného reliéfu z alternativního postupu je znázorněn na obrázku č. 35 (obr. 35), k porovnání DMP (stínovaný reliéf) z finálního postupu na obrázku č. 36 (obr. 36).

Obr. 35 DMP (stínovaný reliéf) z alternativního postupu spojení bodových mračen v překryvech.

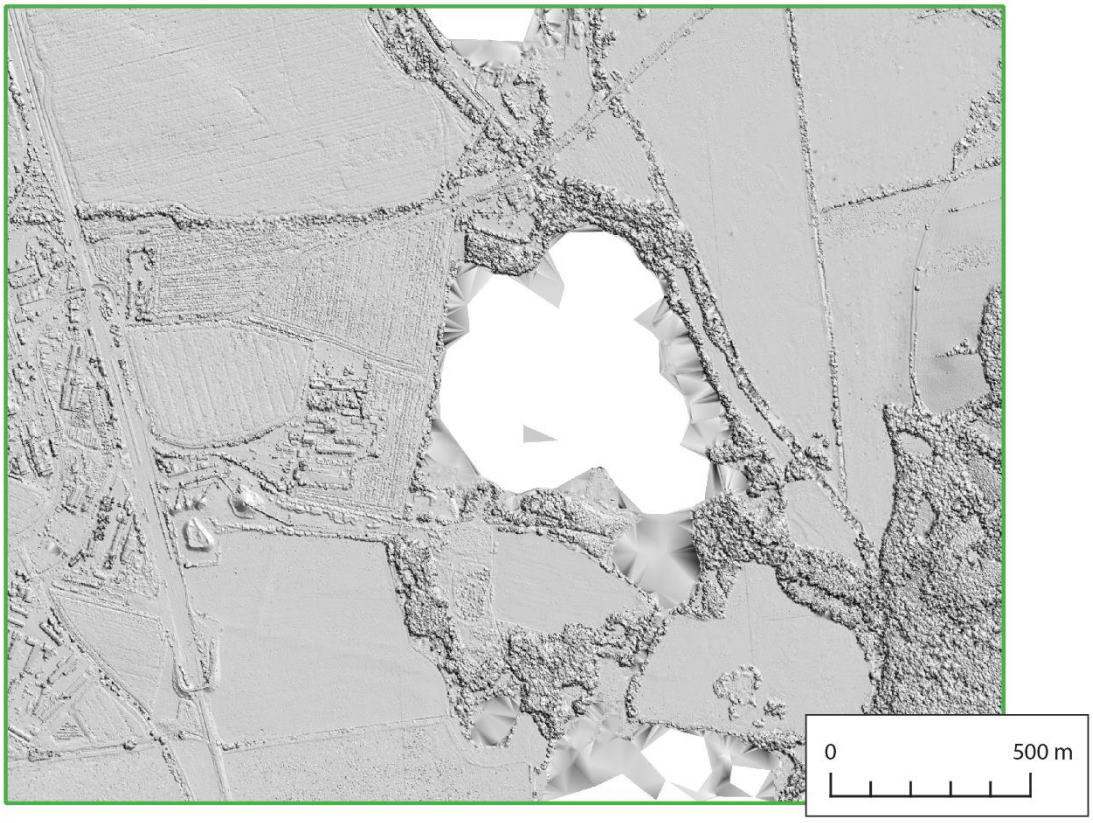
Obr. 36 DMP (stínovaný reliéf) z finálního způsobu spojení bodových mračen v překryvech.

Jak je detailně popsáno v podkapitole 4.4 a jejích podkapitolách, výsledná **pravidla pro spojení bodových mračen** v oblastech jejich překryvů byla stanovena a provedena přes zpracování **vrstvy centroidů** bodových mračen a na jejich základě přes zpracovanou **vrstvu Thiessen polygonů**. Těmito **Thiessen polygony byla bodová mračna ořezána** (z každého byla vybrána pouze jeho centrální část) a tato **ořezaná bodová mračna** byla později **spojována při tvorbě DMP** v blocích SM5.

5.2 Postup pro opravu DMP při selhání obrazové korelace

Jelikož obrazová korelace selhává především na vodních plochách, byl v této části práce stanoven a proveden postup opravy bodových mračen na vodních plochách. Tento vedlejší cíl byl rovněž vyřešen v rámci hlavního cíle – nalezení postupu pro vytvoření spojitého DMP.

Detailně je toto téma řešeno v podkapitole 4.5.2, zde je na obrázku představen prvotní DMP (stínovaný reliéf), tedy před opravou (obr. 37) a poté **finální výsledek** stanoveného a provedeného **postupu opravy bodového mračna** rovněž ve formě DMP (stínovaného reliéfu) (obr. 38). Jak vyplývá z obrázku č. 38, **byla oprava bodového mračna úspěšná**. **Vodní plochy** i **vodní toky byly rekonstruovány** a **u vodních toků odpovídá výškový průběh toku**. **Nadmořská výška se po směru toku snižuje** i s dostatečně věrným zachycením **výrazné změny nadmořské výšky např. na jezech** vyskytujících se na vodním toku.

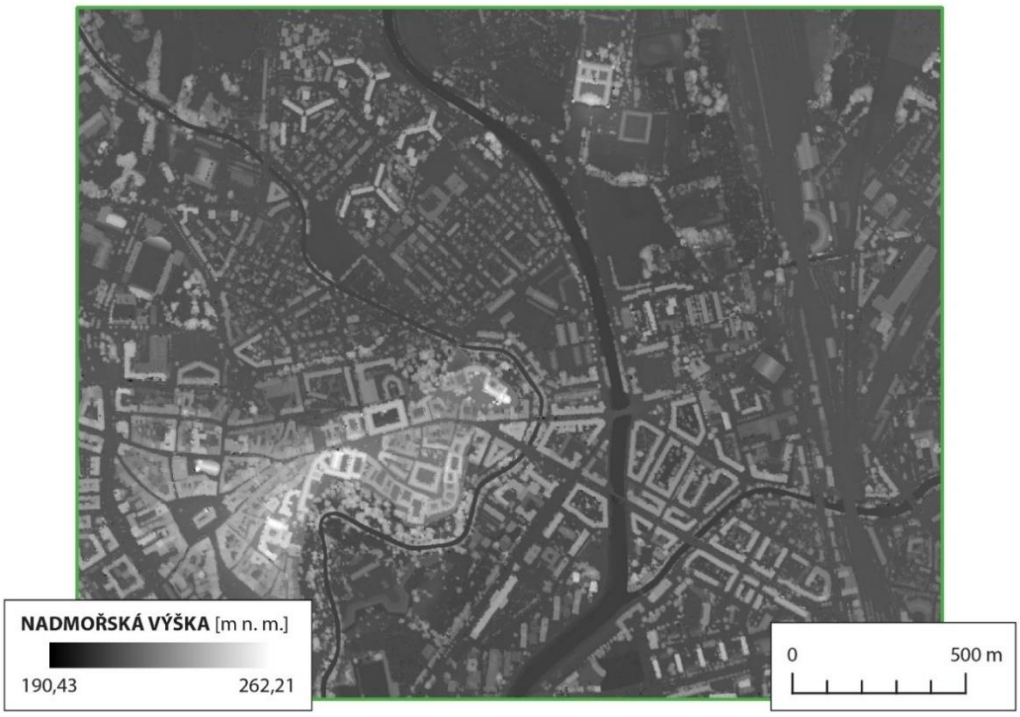


Obr. 37 Prvotní DMP (stínovaný reliéf) – vytvořený z neopraveného bodového mračna.

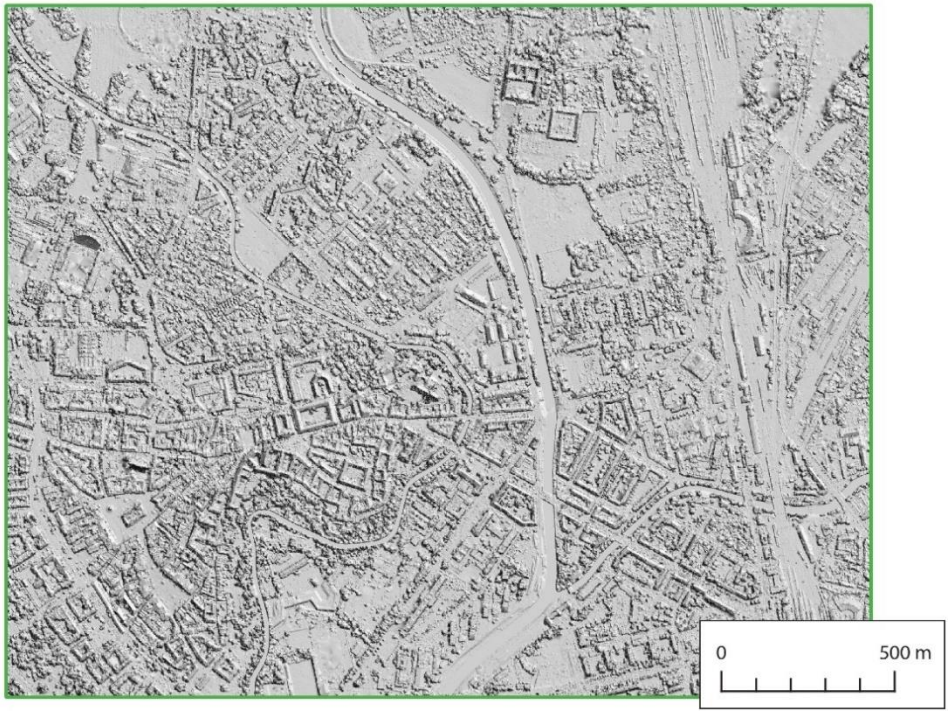
Obr. 38 DMP (stínovaný reliéf) – vytvořený z bodového mračna po opravě.

5.3 Vytvoření spojitého DMP z bodových mračen

Po vyřešení a zpracování předchozích podcílů byl konečně **vytvořen** výsledný produkt – **finální spojitý digitální model povrchu** z bodových mračen vytvořených obrazovou korelací leteckých měřických snímků, čímž byl **naplněn hlavní cíl práce**. Na obrázku 39 je znázorněn tento produkt v městském prostředí (obr. 39) (na obrázku 40 jako stínovaný reliéf (obr. 40)).

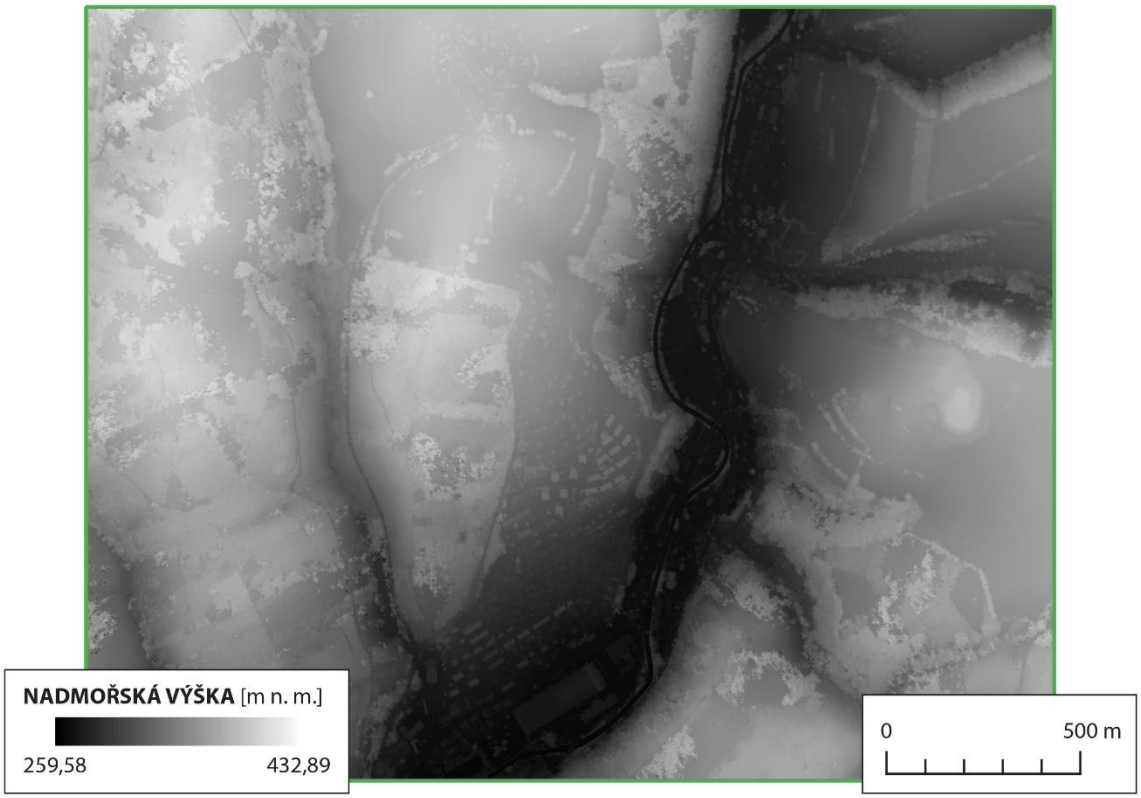


Obr. 39 Finální spojitý digitální model povrchu v městském prostředí.



Obr. 40 Stínovaný reliéf DMP v městském prostředí.

Na obrázku 41 je znázorněn DMP v přírodnějším prostředí (obr. 41) (na obrázku 42 jako stínovaný reliéf (obr. 42)).

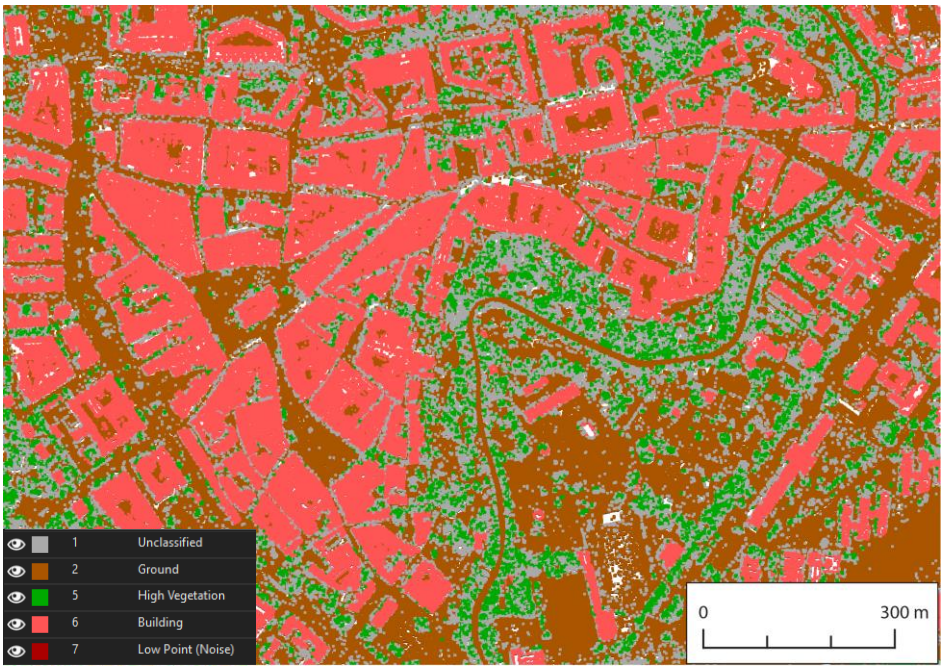


Obr. 41 Finální spojitý digitální model povrchu v přírodnějším prostředí.

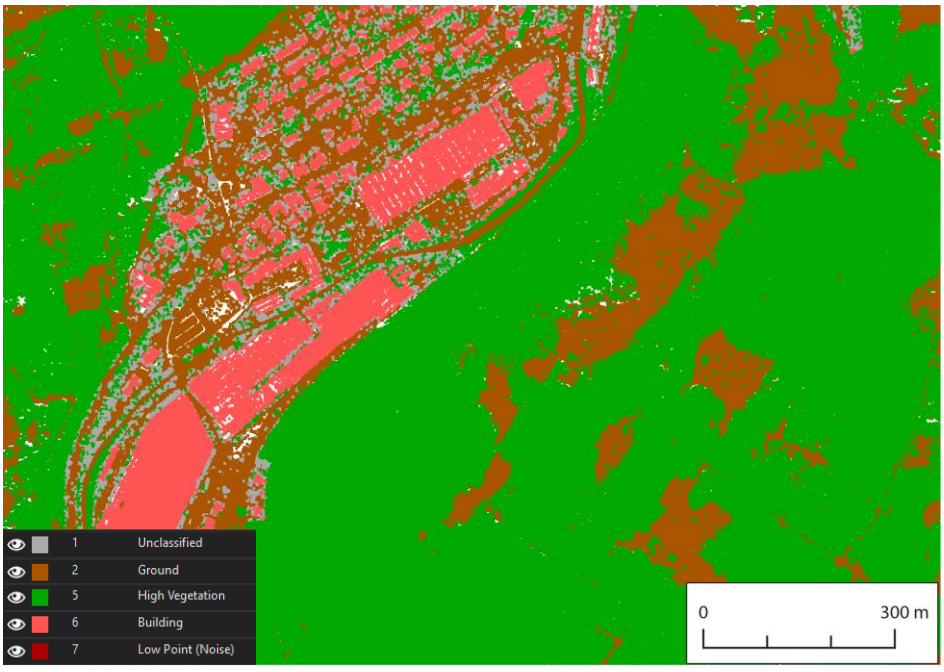
Obr. 42 Stínovaný reliéf DMP v přírodnějším prostředí.

5.4 Pravidla automatické klasifikace bodového mračna

Výsledná **bodová mračna byla klasifikována** do tří základních klasifikačních tříd – **terén** (*ground)*, **budovy** (*buildings*) a **vegetace** (*vegetation*). Ve výsledných klasifikovaných bodových mračnech jsou ovšem body ještě i v dalších klasifikačních třídách, a sice *neklasifikováno* (*unclassified*) a *šum* (*noise*). Klasifikace byla provedena sérií kroků, kdy každý krok byl zaměřen na klasifikaci do jedné konkrétní klasifikační třídy. Podrobně se klasifikaci bodových mračen věnuje podkapitola 4.6. Na obrázku 43 je znázorněno klasifikované bodové mračno v městském prostředí (obr. 43) a na obrázku 44 v přírodnějším prostředí (obr. 44).



Obr. 43 Klasifikované bodové mračno v městském prostředí.



Obr. 44 Klasifikované bodové mračno v přírodnějším prostředí.

5.5 Nástroje detekce hrubých chyb v bodových mračnech

Byl sestaven postup pro automatizovanou detekci hrubých chyb v bodových mračnech spočívající ve vyhledání děr primárně nezpůsobených špatnou kvalitou obrazové korelace na vodních plochách. Rovněž byla vyhledána v bodových mračnech místa nízké hustoty bodů. Všechny oblasti zmíněných děr a míst nízké hustoty bodů byly uloženy do **finálního polygonového SHP souboru**. Detailně je celý postup vysvětlen v podkapitole 4.7, kde jsou rovněž na obrázcích 29, 30 a 31 zobrazeny **finální výsledky**.

5.6 Obrazová korelace v programu Agisoft Metashape

V prostředí programu Agisoft Metashape bylo z LMS obrazovou korelací vygenerováno bodové mračno, které se hustotou bodů může rovnat bodovému mračnu vytvořenému pro potřeby doplnění bloků LMS v programu SURE Aerial. Konkrétně mělo **finální bodové mračno** z programu **Agisoft Metashape** hustotu přesahující **12 bodů/m²**. Z takto hustého bodového mračna proto byl ve stejném programu vytvořen i DMP s **prostorovým rozlišením 0,33 m/px**.

Celý proces je zevrubně popsán v podkapitole 4.8, na jejím konci jsou zobrazeny i výsledky celého postupu. Na obrázku 33 je vyobrazen **finální DMP** a obrázek 34 zachycuje **finální DMP ve formě stínovaného reliéfu**.

6 DISKUZE

V průběhu řešení práce se objevila řada problémů či nedokonalostí dat, které musel autor řešit, a postupy i výstupy byly těmito problémy ovlivněny.

Nejdříve stojí za zmínku nízká hustota vstupních bodových mračen, která je výsledkem zředění původních bodových mračen vzniklých obrazovou korelací LMS na ÚHÚL. Původní bodová mračna byla na ÚHÚL zředěna pro vnitřní potřebu ústavu. Fenomén nízké hustoty bodových mračen se propagoval napříč celou prací, jelikož se jednalo o hlavní vstupní data. Naplnění samotného hlavního cíle práce – nalezení postupu pro vytvoření spojitého DMP z bodových mračen – se přesto nízká hustota dat příliš nedotkla, neboť postup samotný byl vymyšlen a proveden i na těchto relativně řídkých bodových mračnech (s hustotou kolem 1,5 bodu/m^2). Kde se tato nízká hustota bodových mračen nejvíce promítla, jsou pochopitelně finální produkty, tedy finální DMP a finální DMP ve formě stínovaného reliéfu. Vyprodukovány byly v prostorovém rozlišení 1 m/px a jejich kvalita je limitovaná. Tímto jsou ovlivněny a nepřímo determinovány možnosti využití finálních produktů – pro oblasti s převažující vegetací či oblasti holého terénu je kvalita finálních produktů dostatečná, v oblastech intravilánů obcí je kvalita již přinejmenším diskutabilní, poněvadž povrch budov byl z bodových mračen do DMP interpolován jen z velmi malého množství bodů. Obecně jsou budovy v bodových mračnech vzniklých obrazovou korelací LMS vyjádřeny méně body než například právě holý terén z důvodu efektu centrální projekce snímků potažmo radiální distorze, kvůli kterým je na budovách nalezeno méně shodných bodů.

Problémy se vyskytovaly u vektorové polygonové vrstvy 3D vodních ploch a vodních toků používané nejvíce pro opravu bodových mračen právě v místech vodních ploch. Tyto problémy lze rozdělit do dvou kategorií – problémy s polohopisem této vrstvy a problémy s výškopisem.

Hlavním polohopisným problémem je, že geometrie celé vrstvy (pro území celé České republiky) nevyjadřuje stav k jednomu konkrétnímu okamžiku, nýbrž vznikala postupně – i z různých podkladů (na základě různých ortofot). Neustálé změny vodního stavu všech vodních těles i tvaru koryta meandrujících vodních toků souvisí i s dalším problémem. Stav vrstvy nemůže pochopitelně odpovídat skutečnému stavu vodních těles v době pořízení snímků, na kterých je následně provedena obrazová korelace a vzniknuvší bodové mračno má být vrstvou vod opraveno. Hrozí tedy, že bodové mračno bude opraveno méně (v případě vyššího vodního stavu a s tím související větší polohopisnou velikostí vodního tělesa ve skutečnosti (v době snímkování) než jaký je ve vrstvě vod obsažen), tedy že v bodovém mračně zůstanou i neopravené části vodních těles, nebo naopak, že bude opraveno více (nižší vodní stav ve skutečnosti než je obsažen ve vrstvě vod), tedy budou z bodového mračna odstraněny i body ležící již na terénu mimo vodní těleso a budou nahrazeny „vodními body“ z vrstvy vod.

Z výškopisného pohledu je problém, že souřadnice Z lomových bodů dané vrstvy vod byly sice odvozeny z DMR 5G a případně ručně doopravovány, přesto se v této vrstvě nacházela spousta lomových bodů s nulovou hodnotou souřadnice Z. Tyto nulové lomové body pak byly důvodem nedostatečně kvalitní opravy bodových mračen. Po zjištění těchto nulových hodnot, byla sama vrstva částečně opravena odebráním všech nulových lomových bodů. Tím ale nebyl problém nedostatečné výškopisné kvality vrstvy vodních těles vyřešen, neboť v ní i nadále bylo přítomno mnoho nesprávných bodů.

Na přesnost určení úrovně terénu s použitím produktu DMR 5G při klasifikaci bodových mračen mohlo mít zvláště v blocích LMS zahrnujících město Olomouc a okolí vliv, že LLS pro vznik DMR 5G bylo na tomto území prováděno až ve vegetačním období, čímž byla negativně ovlivněna kvalita zmíněného produktu. Tento aspekt byl vyřešen

poměrně širší tolerancí, co bylo považováno za úroveň terénu. Jako třída *ground* byly při klasifikaci klasifikovány ty body bodového mračna, které se nacházely v intervalu -3 až +3 výškové metry od bodů produktu DMR 5G.

V souvislosti s externími vstupními vrstvami do procesu klasifikace mohl nastávat totožný problém, jaký byl výše popsán u vrstvy vodních těles při opravě bodových mračen, tkvící v rozdílném čase vzniku externích vrstev a LMS, ze kterých byla připravena bodová mračna. Nejmarkantněji se mohl tento problém objevit při klasifikaci do třídy budov s využitím externí vrstvy budov. Starší bodové mračno ještě nemuselo obsahovat nově postavenou budovu. Klasifikační algoritmus by pak byl nucen oklasifikovat dané body jako budovu, i když na bodovém mračnu ještě nebyla. Tomuto bylo zamezeno tím, že při klasifikaci budov byl již terén oklasifikován a jeho body byly pro další klasifikační procesy ignorovány.

DMP byl generován v blocích SM5, kdy uvnitř těchto jednotlivých bloků jsou hranice mezi jednotlivými bodovými mračny ořezanými Thiessen polygony téměř neznatelné. Ovšem mezi sousedními bloky SM5 vznikají v DMP *edge artifacts*, tedy tzv. hranové artefakty. Hranové artefakty se mohou vyznačovat buď zcela chybějícími pixely či pixely se špatnými hodnotami (ať už nadmořské výšky či stínovaného reliéfu). Toto je oblast, která by měla být řešena v budoucnu, aby mohl být DMP z jednotlivých bloků SM5 bezproblémově spojován do větších území.

Do budoucna by rovněž měl být celý zjištěný postup v této práci aplikován na nezředěná bodová mračna, aby se dokázalo, nakolik DMP vzniklý tímto způsobem dokáže konkurovat DMP z LLS.

Jak zmiňují Elshehaby a Taha (2018), DMP z LLS je stále kvalitnější z hlediska absolutní přesnosti. Současnými možnostmi obrazové korelace LMS vedoucí až ke tvorbě DMP se ale výsledný DMP svou kvalitou tomu z LLS přibližuje (ovšem záleží na kvalitě vstupních LMS, překryvech mezi LMS, konkrétní metodě obrazové korelace, hustotě bodového mračna atd.). Provádění LLS navíc často bývá nákladnější než provádění LMS, proto má smysl tvořit DMP i z dat LMS, především pokud kvalita DMP z LMS odpovídá účelu a požadavkům využití takového produktu.

7 ZÁVĚR

Hlavním cílem diplomové práce bylo nalezení postupu pro vytvoření spojitého digitálního modelu povrchu z bodových mračen vytvořených obrazovou korelací leteckých měřických snímků. Jako vstupní data byla použita zředěná bodová mračna z obrazové korelace poskytnutá Ústavem pro hospodářskou úpravu lesů Brandýs nad Labem v území pěti bloků leteckého měřického snímkování. Veškeré operace se vstupními daty byly prováděny sestavenými skripty v programovacím jazyce Python, do nichž byly implementovány nástroje pro práci s bodovými mračny z programového balíku LAStools a na geoprocessingové operace byla využívána knihovna ArcPy z geografického informačního systému ArcGIS Pro.

Surová data byla nejdříve upravena do použitelné podoby pro zpracování, po zjištění, že dodaná data netvoří kompletní bloky, byla vyhledána chybějící bodová mračna a bylo určeno, na jakých LMS by měla být provedena dodatečná obrazová korelace k doplnění bloků. Tato dodatečná obrazová korelace byla následně provedena v programu SURE Aerial a nově vytvořenými bodovými mračny byly bloky doplněny.

Jakmile byly bloky bodových mračen kompletní, byla **stanovena pravidla pro spojení bodových mračen v překryvových pásech** jednotlivých stereomodelů. Jelikož byl výsledný DMP tvořen v blocích SM5, byla spojena vždy bodová mračna, která zasahovala do vybraného bloku SM5, a tímto blokem byla též ořezána. Bodové mračno v území bloku SM5 bylo nazváno prvotním DMP. Prvotní DMP byl chybový v místech, kde obrazová korelace selhává – tedy především na vodních plochách. S využitím polygonové vrstvy 3D vodních ploch byly chybné body z bodových mračen odstraněny, vytvořeny byly nové výškově odpovídající body, které byly sloučeny s bodovými mračny s odstraněnými chybnými body. Takto **vznikla finální opravená bodová mračna**, ze kterých byl vygenerován **finální digitální model povrchu** a finální DMP ve formě stínovaného reliéfu.

Na opravených bodových mračnech byla **stanovena pravidla klasifikace do minimálně tří klasifikačních tříd**. S využitím produktu DMR 5G byla klasifikována **třída terénu** (*ground*), s využitím vrstev staveb jak ze ZABAGED®, tak z RÚIAN, byla klasifikována **třída budov** (*building*) a jednak pomocí vrstvy lesů ze ZABAGED® a vlastním klasifikačním nástrojem z LAStools byla klasifikována **třída vegetace** (*high vegetation*).

Opravená bodová mračna i přes opravu na vodních plochách obsahovala hrubé chyby – spočívající nejčastěji v náhodně se vyskytujících dírách (právě nezpůsobených vodními plochami). Tyto **díry byly automaticky vyhledány** a v jejich místech **vznikla polygonová vrstva**, v níž jsou **místa děr uložena**. Současně **vyhledána a uložena** do polygonové vrstvy **byla** i **místa příliš nízké hustoty bodů** v bodových mračnech vzniklá specifickými vlastnostmi povrchů, které způsobily, že obrazovou korelací na těchto površích vzniklo bodů jen velmi málo.

Závěrem byl ještě **vytvořen DMP** v programu **Agisoft Metashape**, kde byl **proveden kompletně celý proces tvorby DMP z LMS**. **Finální DMP je kvalitním produktem** obzvláště v místech **budov**, na **vegetaci** je **kvalita slabší**. Algoritmy obrazové korelace ve zmíněném programu na vegetaci detekovaly méně spojovacích bodů než právě na budovách, proto při tvorbě DMP z bodového mračna proběhla v místech vegetace interpolace z méně bodů, což je důvodem mírně nižší kvality. **Celkově je ale produkt kvalitní**.

Jak již bylo nastíněno v kapitole Diskuze, kvalita DMP vytvořených z LMS nedosahuje kvalit DMP vzniklých z LLS. Přesto **tato práce dokázala, že i z dat LMS**, která jsou pro jakékoli místo v České republice pořizována a zpracovávána každé dva roky primárně pro tvorbu ortofota, **lze produkovat DMP využitelný v mnoha aplikacích** bez nutnosti

provádění ještě i celoplošného LLS (pro celé území republiky), které je jednak nákladnější a jednak složitější na zpracování. LLS tak může být realizováno pouze pro aplikace a formy využití – především na menších územích, které precizní kvalitu LLS nezbytně vyžadují.

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

AGISOFT, 2023. Agisoft Metashape. *Agisoft* [online]. 2023 [cit. 2023-04-26]. Dostupné z: https://www.agisoft.com/

AGISOFT, 2019. Agisoft Metashape User Manual: Professional Edition, Version 1.5. *Agisoft* [online]. 2019 [cit. 2023-04-26]. Dostupné z: https://www.agisoft.com/pdf/metashape-pro_1_5_en.pdf

ASPRS, 2013. LAS SPECIFICATION: VERSION 1.4 – R13. *American Society for Photogrammetry & Remote Sensing* [online]. 15 July 2013 [cit. 2023-04-03]. Dostupné z: https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf

BRAUN, Jaroslav, Hana BRAUNOVÁ, Tomáš SUK, Ondřej MICHAL, Patrik PEŤOVSKÝ a Ivan KURIC, 2021. Structural and Geometrical Vegetation Filtering - Case Study on Mining Area Point Cloud Acquired by UAV Lidar. *Acta Montanistica Slovaca* [online]. 2021, **26**(4), 661-674 [cit. 2023-04-29]. ISSN 1335-1788. Dostupné z: doi:10.46544/AMS.v26i4.06

ČÚZK, 2023a. Prohlížecí služba WMS - ortofoto CIR. *ČÚZK* [online]. 2023-03-28 [cit. 2023-04-26]. Dostupné z: https://geoportal.cuzk.cz/(S(ypmzz5aolypwrq2txsenuzqp))/Default.aspx?mode=TextMeta&side=wms.verejne&metadataID=CZ-CUZK-WMS-ORTOCIR&metadataXSL=metadata.sluzba&head_tab=sekce-03-gp&menu=3132

ČÚZK, 2023b. Registr územní identifikace, adres a nemovitostí (RÚIAN). *ČÚZK* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://www.cuzk.cz/ruian/

ČÚZK, 2023c. Státní mapa 1:5 000. *ČÚZK* [online]. 2023-03-28 [cit. 2023-04-21]. Dostupné z: https://geoportal.cuzk.cz/(S(hupkewmrbwcs3u4qcdw3ikrf))/Default.aspx?mode=TextMeta&side=dSady_archiv&metadataID=CZ-CUZK-SMO5-R&head_tab=sekce-02-gp&menu=2905

ČÚZK, 2023d. ZABAGED® - polohopis - úvod. *ČÚZK* [online]. 11.01.2023 [cit. 2023-04-08]. Dostupné z: https://geoportal.cuzk.cz/(S(v34ksue5irr4p4fkbf2r2asm))/default.aspx?mode=TextMeta&text=dSady_zabaged&side=zabaged&menu=24

ČÚZK, 2023e. ZABAGED® - Výškopis - DMR 5G. Digitální model reliéfu České republiky 5. generace. *ČÚZK* [online]. 2023-03-28 [cit. 2023-04-08]. Dostupné z: https://geoportal.cuzk.cz/(S(a53jglc1ud1he03qg3g3umwg))/Default.aspx?lng=CZ&mode=TextMeta&side=vyskopis&metadataID=CZ-CUZK-DMR5G-V&mapid=8&menu=302

ČÚZK, 2022. ZABAGED® - Výškopis - DMP 1G. Digitální model povrchu České republiky 1. generace. *ČÚZK* [online]. 2022-02-28 [cit. 2023-04-29]. Dostupné z: https://geoportal.cuzk.cz/(S(0nc0sukzkzd2cptwc1vgzeua))/Default.aspx?mode=TextMeta&side=vyskopis&metadataID=CZ-CUZK-DMP1G-V&head_tab=sekce-02-gp&menu=303

ELSHEHABY, Ayman Mohamed Rashad a Lamyaa Gamal El-deen TAHA, 2018. Assessment of the Automatic Digital Surface Model from Digital Aerial Camera and from Lidar Point Clouds Data. *Australian Journal of Basic and Applied Sciences* [online]. 13 September 2018, **12**(9), 52–57 [cit. 2023-04-04]. ISSN 19918178. Dostupné z: doi:10.22587/ajbas.2018.12.9.9

ESRI, 2023a. Create Thiessen Polygons (Analysis). *Esri* [online]. 2023 [cit. 2023-04-20]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/create-thiessen-polygons.htm

ESRI, 2023b. Erase (Analysis). *Esri* [online]. 2023 [cit. 2023-04-14]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/erase.htm

ESRI, 2023c. Export Feature Attribute To ASCII (Spatial Statistics). *Esri* [online]. 2023 [cit. 2023-04-18]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/export-feature-attribute-to-ascii.htm

ESRI, 2023d. Minimum Bounding Geometry (Data Management). *Esri* [online]. 2023 [cit. 2023-04-14]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/minimum-bounding-geometry.htm

ESRI, 2023e. SearchCursor. *Esri* [online]. 2023 [cit. 2023-04-21]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/arcpy/data-access/searchcursor-class.htm

ESRI, 2023f. What is ArcPy?. *Esri* [online]. 2023 [cit. 2023-04-06]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/arcpy/get-started/what-is-arcpy-.htm

G4D, 2023. 3D mapy a modely terénu. *G4D* [online]. 2023 [cit. 2023-04-29]. Dostupné z: https://www.g4d.cz/3d-mapy-a-modely-terenu

GONG, Ke a Dieter FRITSCH, 2018. POINT CLOUD AND DIGITAL SURFACE MODEL GENERATION FROM HIGH RESOLUTION MULTIPLE VIEW STEREO SATELLITE IMAGERY. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. 2018, **XLII-2**, 363-370 [cit. 2023-04-09]. ISSN 2194-9034. Dostupné z: doi:10.5194/isprs-archives-XLII-2-363-2018

HIRSCHMÜLLER, Heiko, 2011. Semi-Global Matching – Motivation, Developments and Applications. *Photogrammetric Week 11* [online]. Stuttgart, Germany, September 2011, 173-184 [cit. 2023-03-28]. Dostupné z: https://elib.dlr.de/73119/1/180Hirschmueller.pdf

HIRSCHMÜLLER, Heiko, 2008. Stereo Processing by Semi-Global Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. **30**(2), 328-341 [cit. 2023-03-28]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2007.1166

HIRSCHMÜLLER, Heiko, 2005. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* [online]. IEEE, s. 807-814 [cit. 2023-03-28]. ISBN 0-7695-2372-2. Dostupné z: doi:10.1109/CVPR.2005.56

JETBRAINS, 2023. PyCharm – Quick start guide. *JetBrains* [online]. 10 March 2023 [cit. 2023-04-09]. Dostupné z: https://www.jetbrains.com/help/pycharm/quick-start-guide.html

LIBRARY OF CONGRESS, 2022. LAS (LASer) File Format, Version 1.4. *Library of Congress* [online]. 08/08/2022 [cit. 2023-04-03]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000418.shtml

MATHWORKS, 2023. Stereo Disparity Using Semi-Global Block Matching. *MathWorks* [online]. 2023 [cit. 2023-03-27]. Dostupné z: https://www.mathworks.com/help/visionhdl/ug/stereoscopic-disparity.html

MIŘIJOVSKÝ, Jakub, 2013a. *Bezpilotní systémy: sběr dat a využití ve fotogrammetrii*. Olomouc: Univerzita Palackého v Olomouci pro katedru geoinformatiky. Terra notitia. ISBN 978-80-244-3923-5.

MIŘIJOVSKÝ, Jakub, 2013b. *Fotogrammetrický přístup při sběru geodat pomocí bezpilotních leteckých zařízení*. Olomouc. Disertační práce (Ph.D.). Univerzita Palackého v Olomouci. Přírodovědecká fakulta.

NFRAMES, 2023a. Camera and Flight Specifications. *NFrames* [online]. 2023 [cit. 2023-04-18]. Dostupné z: https://experience.arcgis.com/experience/af8d4766bf4d4cdd91ba97d45e9d6c89/page/Flight-Planning/

NFRAMES, 2023b. Empowering Photogrammetry with SURE for ArcGIS Software. *NFrames* [online]. 2023 [cit. 2023-04-09]. Dostupné z: https://experience.arcgis.com/experience/af8d4766bf4d4cdd91ba97d45e9d6c89/page/About-SURE-for-ArcGIS/

NFRAMES, 2023c. NFrames. *NFrames* [online]. 2023 [cit. 2023-04-18]. Dostupné z: https://www.nframes.com/

NFRAMES, 2023d. NFrames - About Us. *NFrames* [online]. 2023 [cit. 2023-04-09]. Dostupné z: https://www.nframes.com/about-us/

OGC, 2018. LAS Specification 1.4: OGC Community Standard. *Open Geospatial Consortium* [online]. 2018-03-01 [cit. 2023-04-03]. Dostupné z: http://www.opengis.net/doc/CS/las/1.4

PAVELKA, Karel, 2003. *Fotogrammetrie 20*. Vyd. 2. přepracované. Praha: Vydavatelství ČVUT. ISBN 80-01-02762-7.

PUZZO, Lynn, 2021. LiDAR vs Photogrammetry: Which is better for point cloud creation?: What's the best technique for creating a point cloud? Let's face off lidar vs photogrammetry and find out. *Mosaic51: Robust 360° Cameras & Solutions* [online]. December 12, 2021 [cit. 2023-03-28]. Dostupné z: https://www.mosaic51.com/technology/lidar-vs-photogrammetry-which-is-better-for-point-cloud-creation/

PYTHON, 2023. General Python FAQ: What is Python?. *Python.org* [online]. Apr 05, 2023 [cit. 2023-04-06]. Dostupné z: https://docs.python.org/3/faq/general.html#what-is-python

RAMOS, Leonidas Pozo, Aldren SANTOS, Geir Arne HJELLE, Jaya ZHANÉ, Joanna JABLONSKI a Mike DRISCOLL, 2019. Using the Python zip() Function for Parallel Iteration. *Real Python Tutorials* [online]. 2019 [cit. 2023-04-13]. Dostupné z: https://realpython.com/python-zip-function/

RAPIDLASSO, 2023a. Blast2dem. *Rapidlasso* [online]. 2023 [cit. 2023-04-28]. Dostupné z: https://downloads.rapidlasso.de/readme/blast2dem_README.md

RAPIDLASSO, 2023b. Las2dem. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/las2dem_README.md

RAPIDLASSO, 2023c. Las2las. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/las2las_README.md

RAPIDLASSO, 2023d. Lasboundary. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/lasboundary_README.md

RAPIDLASSO, 2023e. Lasclassify. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/lasclassify_README.md

RAPIDLASSO, 2023f. Lasclip. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/lasclip_README.md

RAPIDLASSO, 2023g. Lasheight. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/lasheight_README.md

RAPIDLASSO, 2023h. Lasmerge. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/lasmerge_README.md

RAPIDLASSO, 2023i. Lasnoise. *Rapidlasso* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://downloads.rapidlasso.de/readme/lasnoise_README.md

RAPIDLASSO, 2023j. LAStools. *Rapidlasso* [online]. 2023 [cit. 2023-04-09]. Dostupné z: https://rapidlasso.de/product-overview/

TIPPNER, Aleš a Petr DUŠÁNEK, 2022. *Technická zpráva k produktu ZABAGED® - Výškopis - Vrstevnice* [online]. 29. dubna 2022 [cit. 2023-04-08]. Dostupné z: https://geoportal.cuzk.cz/Dokumenty/TECHNICKA_ZPRAVA_ZABAGED-Vyskopis-Vrstevnice.pdf

VAN ROSSUM, Guido, 2023. What is Python? Executive Summary. *Python.org* [online]. 2023 [cit. 2023-04-06]. Dostupné z: https://www.python.org/doc/essays/blurb/

YALCIN, Emrah, 2018. GENERATION OF HIGH-RESOLUTION DIGITAL SURFACE MODELS FOR URBAN FLOOD MODELLING USING UAV IMAGERY. *WIT Transactions on Ecology and the Environment* [online]. 2018, **215**, 357-366 [cit. 2023-04-09]. ISSN 1743-3541. Dostupné z: doi:10.2495/EID180321

PŘÍLOHY

SEZNAM PŘÍLOH

Elektronické přílohy

Příloha 1	ÚHÚL: Souhlas s využitím dat „Mračna bodů z obrazové korelace“
Příloha 2	Skript 00_rename_laz.py
Příloha 3	Skript 00_repair_laz.py
Příloha 4	Skript 01_lasboundary.py
Příloha 5	Skript 02_block_merge.py
Příloha 6	Skript 03_images_for_given_block_and_year.py
Příloha 7	Skript 04_images_to_correlate_for_missing_stereopairs.py
Příloha 8	Skript 04_x_delete_remaining_temporary_folder.py
Příloha 9	Skript 05_repair_laz_all.py
Příloha 10	Skript 06_lasboundary_all.py
Příloha 11	Skript 07_block_merge_all.py
Příloha 12	Skript 08_centroids.py
Příloha 13	Skript 09_thiessen_polygons.py
Příloha 14	Skript 10_lasclip.py
Příloha 15	Skript 11_make_dsm_in_sm5.py
Příloha 16	Skript 12_final_dsm.py
Příloha 17	Skript 13_classification.py
Příloha 18	Skript 14_detect_holes.py

Volné přílohy

Příloha 19	Poster

Popis struktury odevzdávaných digitálních dat na datové úložiště katedry

Adresáře:

- Poster
- Skripty
- Text_Prace (text formát MS Word a PDF)
- Vstupni_Data
- Vystupni_Data
- WEB

Pozn.: Po individuální domluvě s Mgr. Danielem Pavlačkou obsahuje složka vstupních i výstupních dat z důvodu jejich extrémní velikosti pouze textový soubor s informací, že tato data budou katedře předána prostřednictvím jejich fyzického překopírování právě Mgr. Pavlačkovi.

Bodová mračna z obrazové korelace byla Ústavem pro hospodářskou úpravu lesů Brandýs nad Labem poskytnuta pouze pro zpracování této diplomové práce.

Příloha 1: ÚHÚL: Souhlas s využitím dat „Mračna bodů z obrazové korelace“

www.uhul.cz
Informace o lesích

ÚSTAV PRO HOSPODÁŘSKOU ÚPRAVU LESŮ BRANDÝS NAD LABEM
Nábřežní 1326, 250 01 Brandýs nad Labem – Stará Boleslav

Organizační složka státu
zřízená Ministerstvem zemědělství ČR
zřizovací listinou čj. 27819/2001-3030

Přírodovědecká fakulta
Univerzity Palackého v Olomouci
RNDr. Jakub Miřijovský, Ph.D.
17. listopadu 1192/12
779 00 Olomouc

VÁŠ DOPIS ZNAČKY ZE DNE	NAŠE ZNAČKA	VYŘIZUJE	DNE
	UHUL/1122/2022/ISaT	Ing. Říha Milan	16. 3. 2022

Věc: Souhlas s využitím dat „Mračna bodů z obrazové korelace“

Ústav pro hospodářskou úpravu lesů Brandýs nad Labem souhlasí s využitím dat „Mračna bodů z obrazové korelace“ jako podklad pro diplomovou práci studenta Bc. Jindřicha Horáka (Univerzita Palackého v Olomouci, Přírodovědecká fakulta, Katedra geoinformatiky), a to v souladu s Vaší žádostí o výdej dat z IDC ÚHÚL ze dne 4. 3. 2022.

Data jsou vytvářena pro potřeby Ústavu pro hospodářskou úpravu lesů a předávána k využití Zeměměřickému úřadu v Pardubicích. Bc. Jindřich Horák převezme data od ZÚ v Pardubicích.

Data „Mračna bodů z obrazové korelace“ budou využita pouze pro daný účel diplomové práce a bude označen zdroj dat – ÚHÚL Brandýs nad Labem.

.....

Ing. Martin Bureš
Náměstek ISaT ÚHÚL

Příloha 2: Skript 00_rename_laz.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 00_rename_laz.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\00_stereopairs_laz_UHUL\\'
   opath = curpath + '\\00_stereopairs_laz_UHUL_renamed\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'laz'
   osuf = 'laz'
   sep = '.'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)
           print('files: ', fls)
           for fl in fls:
               fl = fl.replace('.laz', '')
               inn = ipath + year + '\\' + block + '\\' + fl + sep + isuf
               semiout = opath + year + '\\' + block + '\\' +
fl.replace('_py_0_0', '').upper()
               out = semiout.replace('.', '_') + sep + osuf

               try:
                   os.makedirs(opath + year + '\\' + block)
                   print('created folder: ', opath + year + '\\' + block)
```

```
            except:
                pass

            execute = 'copy ' + inn + ' ' + out
            print(execute)
            subprocess.call(execute, shell=True)

        print('------------------------------------------------------------')
    print('============================================================')
print('======================END OF SCRIPT!======================')

end = datetime.now()
print('time elapsed: ', end - start)
```

Příloha 3: Skript 00_repair_laz.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 00_repair_laz.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\00_stereopairs_laz_UHUL_renamed\\'
   opath = curpath + '\\00_stereopairs_laz_UHUL_repaired\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'laz'
   osuf = 'laz'
   sep = '.'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   las2las = lastools_base_path + lastools_version + '\\bin\\las2las64.exe'

   las2las_settings = '-auto_reoffset -rescale 0.01 0.01 0.01 -
repair_zero_returns'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)
           print('files: ', fls)
           for fl in fls:
               try:
```

```
                    os.makedirs(opath + year + '\\' + block)
                    print('created folder: ', opath + year + '\\' + block)
                except:
                    pass

                fl = fl.replace('.laz', '')
                execute = las2las + ' -i ' + ipath + year + '\\' + block + '\\' +
fl + sep + isuf + ' -odir ' + opath + year + '\\' + block + ' -o' + osuf + ' ' +
las2las_settings
                print(execute)
                subprocess.call(execute, shell=True)

            print('-------------------------------------------------------------')
        print('=============================================================')
    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 4: Skript 01_lasboundary.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 01_lasboundary.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\00_stereopairs_laz_UHUL_repaired\\'
   opath = curpath + '\\01_stereopairs_laz_UHUL_lasboundary\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'laz'
   osuf = 'shp'
   sep = '.'
   epsg5514 = 'PROJCS[\"S-
JTSK_Krovak_East_North\",GEOGCS[\"GCS_S_JTSK\",DATUM[\"D_S_JTSK\",SPHEROID[\"Bess
el_1841\",6377397.155,299.1528128]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0
174532925199433]],PROJECTION[\"Krovak\"],PARAMETER[\"False_Easting\",0.0],PARAMET
ER[\"False_Northing\",0.0],PARAMETER[\"Pseudo_Standard_Parallel_1\",78.5],PARAMET
ER[\"Scale_Factor\",0.9999],PARAMETER[\"Azimuth\",30.28813975277778],PARAMETER[\"
Longitude_Of_Center\",24.83333333333333],PARAMETER[\"Latitude_Of_Center\",49.5],P
ARAMETER[\"X_Scale\",-
1.0],PARAMETER[\"Y_Scale\",1.0],PARAMETER[\"XY_Plane_Rotation\",90.0],UNIT[\"Mete
r\",1.0]]'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   lasboundary = lastools_base_path + lastools_version +
'\\bin\\lasboundary64.exe'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
```

```
        print('blocks: ', blocks)
        for block in blocks:
            print('block: ', block)
            fls = os.listdir(ipath + year + '\\' + block)
            print('files: ', fls)

            for fl in fls:
                try:
                    os.makedirs(opath + year + '\\' + block)
                    print('created folder: ', opath + year + '\\' + block)
                except:
                    pass

                fl = fl.replace('.laz', '')
                execute = lasboundary + ' -i ' + ipath + year + '\\' + block +
'\\' + fl + sep + isuf + ' -odir ' + opath + year + '\\' + block + ' -o' + osuf
                print(execute)
                subprocess.call(execute, shell=True)

                # define projection EPSG 5514 to fl.shp
                ap.env.workspace = opath + year + '\\' + block
                outputfl = fl + sep + osuf
                ap.management.DefineProjection(outputfl, epsg5514)
                print('projection EPSG 5514 defined to output lasboundary file: ',
outputfl)

                # add field to fl.shp
                ap.management.AddField(outputfl, 'name', 'TEXT')
                print("field 'name' added to output lasboundary file: ", outputfl)

                # set the file name as the name attribute
                ap.management.CalculateField(outputfl, 'name', 'fl', 'PYTHON3')
                print("attribute 'name' set to: ", fl)

            print('-------------------------------------------------------------')
        print('=============================================================')
    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 5: Skript 02_block_merge.py

```
   # ------------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 02_block_merge.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\01_stereopairs_laz_UHUL_lasboundary\\'
   opath = curpath + '\\02_stereopairs_laz_UHUL_block_merged_boundary\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('-----------------------------------------------------------------')

   isuf = 'shp'
   osuf = 'shp'
   sep = '.'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)

           inputs = []
           shps = []
           for fl in fls:
               try:
                   os.makedirs(opath + year + '\\' + block)
                   print('created folder: ', opath + year + '\\' + block)
               except:
                   pass

               if fl[-3:] == isuf:
```

```
                shps.append(fl)
                fl = fl.replace(fl, ipath + year + '\\' + block + '\\' + fl)
                inputs.append(fl)

        print('boundaries to be merged: ', shps)

        # merge files from inputs list
        ap.env.workspace = opath + year + '\\' + block
        outputfl = block + '_block_merged_boundary' + sep + osuf
        ap.management.Merge(inputs, outputfl)
        print('boundaries merged to file: ', outputfl)

        print('------------------------------------------------------------')
    print('============================================================')
print('=======================END OF SCRIPT!=======================')

end = datetime.now()
print('time elapsed: ', end - start)
```

Příloha 6: Skript 03_images_for_given_block_and_year.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 03_images_for_given_block_and_year.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\02_stereopairs_laz_UHUL_block_merged_boundary\\'
   opath = curpath +
'\\03_stereopairs_laz_UHUL_images_for_given_block_and_year\\'
   dbpath = curpath + '\\DB\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'shp'
   osuf = 'shp'
   sep = '.'
   dbname = 'ALS_20211123.gdb'
   dbImagesLayer = dbpath + dbname + '\\' + 'LMS_META_F'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)

           for fl in fls:
               try:
                   os.makedirs(opath + year + '\\' + block)
                   print('created folder: ', opath + year + '\\' + block)
               except:
                   pass
```

```
                if fl[-3:] == isuf:
                    # select by location - select images in given area
                    ap.env.workspace = ipath + year + '\\' + block
                    selected = ap.management.SelectLayerByLocation(dbImagesLayer,
'INTERSECT', fl, '', 'NEW_SELECTION')
                    # select by attributes - create subset from current selection
by attributes - by given year
                    sql = "NAME LIKE '" + year + "%'"
                    ap.management.SelectLayerByAttribute(selected,
"SUBSET_SELECTION", sql)
                    # save selected images as a new file
                    selectedImagesName = 'block_' + block + '_images'
                    outputfl = opath + year + '\\' + block + '\\' +
selectedImagesName + sep + osuf
                    ap.management.CopyFeatures(selected, outputfl)
                    print('images for block ', block, ' were selected and saved
as: ', selectedImagesName)

            print('------------------------------------------------------------')
        print('============================================================')
    print('=======================END OF SCRIPT!=======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 7: Skript 04_images_to_correlate_for_missing_stereopairs.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 04_images_to_correlate_for_missing_stereopairs.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import ntpath
   import os
   import shutil

   curpath = os.path.dirname(os.getcwd())
   ipath_boundary = curpath + '\\02_stereopairs_laz_UHUL_block_merged_boundary\\'
   ipath_images = curpath +
'\\03_stereopairs_laz_UHUL_images_for_given_block_and_year\\'
   opath = curpath +
'\\04_stereopairs_laz_UHUL_images_to_correlate_for_missing_stereopairs\\'
   waterpath = curpath + '\\Vody\\'
   print('boundary input folder: ', ipath_boundary)
   print('images input folder: ', ipath_images)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'shp'
   osuf = 'shp'
   txtsuf = 'txt'
   xmlsuf = 'xml'
   sep = '.'
   water = waterpath + '5514_vody_poly.shp'

   start = datetime.now()

   lstOfLst = []
   years = os.listdir(ipath_boundary)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath_boundary + year)
       print('blocks: ', blocks)
       allFiles = []
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath_boundary + year + '\\' + block)
```

```
        for fl in fls:
            if fl[-3:] == isuf:
                flPath = ipath_boundary + year + '\\' + block + '\\' + fl
                allFiles.append(flPath)
    lstOfLst.append(allFiles)
print('all blocks and their boundaries:')
print(lstOfLst)

print('------------------------------------------------------------')

lstOfLst1 = []
years1 = os.listdir(ipath_images)
# print('years: ', years1)
for year1 in years1:
    # print('year: ', year1)
    blocks1 = os.listdir(ipath_images + year1)
    # print('blocks: ', blocks1)
    allFiles1 = []
    for block1 in blocks1:
        # print('block: ', block1)
        fls1 = os.listdir(ipath_images + year1 + '\\' + block1)
        for fl1 in fls1:
            if fl1[-3:] == isuf:
                flPath1 = ipath_images + year1 + '\\' + block1 + '\\' + fl1
                allFiles1.append(flPath1)
    lstOfLst1.append(allFiles1)
print('all blocks and their images:')
print(lstOfLst1)

print('------------------------------------------------------------')

for boundaries, images in zip(lstOfLst, lstOfLst1):
    for boundary, image in zip(boundaries, images):
        try:
            # create temporary folders
            copyDir = opath + '01_copy'
            os.makedirs(copyDir)
            print('01_copy dir created')
            mbrDir = opath + '02_mbr'
            os.makedirs(mbrDir)
            print('02_mbr dir created')
            eraseDir = opath + '03_erase'
            os.makedirs(eraseDir)
            print('03_erase dir created')
            polToLineDir = opath + '04_polToLine'
            os.makedirs(polToLineDir)
            print('04_polToLine dir created')
            bufferDir = opath + '05_buffer'
            os.makedirs(bufferDir)
```

```
                print('05_buffer dir created')
                eraseEraseDir = opath + '06_eraseErase'
                os.makedirs(eraseEraseDir)
                print('06_eraseErase dir created')
                multiToSingleDir = opath + '07_multiToSingle'
                os.makedirs(multiToSingleDir)
                print('07_multiToSingle dir created')
                clipDir = opath + '08_clip'
                os.makedirs(clipDir)
                print('08_clip dir created')
                waterMultiToSingleDir = opath + '09_waterMultiToSingle'
                os.makedirs(waterMultiToSingleDir)
                print('09_waterMultiToSingle dir created')
                selectionEraseDir = opath + '10_selectionEraseDir'
                os.makedirs(selectionEraseDir)
                print('10_selectionErase dir created')
                print('-------------------------------------------------------------
-')
            except:
                pass

            try:
                # duplicate original block_merged_boundary because Repair Geometry
tool modifies input data
                origBasename = ntpath.basename(boundary)
                newBasename = origBasename.replace(sep + isuf, '') + '1' + sep +
osuf
                outFlCopy = copyDir + '\\' + newBasename
                ap.management.CopyFeatures(boundary, outFlCopy)
                print(origBasename + ' copied as: ' + newBasename)

                # repair geometry
                ap.management.RepairGeometry(outFlCopy)
                outFlRepair = outFlCopy
                print('repaired geometry on shp: ' + newBasename)

                # create Minimum Bounding Rectangle
                outNameMBR = 'block_' + newBasename[:6] + '_MBR' + sep + osuf
                outFlMBR = mbrDir + '\\' + outNameMBR
                ap.management.MinimumBoundingGeometry(outFlRepair, outFlMBR, '',
'ALL')
                print('Minimum Bounding Rectangle around ' + newBasename + '
created as: ' + outNameMBR)

                # erase
                outNameErase = 'block_' + newBasename[:6] + '_Erase' + sep + osuf
                outFlErase = eraseDir + '\\' + outNameErase
                ap.analysis.Erase(outFlMBR, outFlRepair, outFlErase)
                print('erased area of ' + newBasename + ' from ' + outNameMBR + '
and created: ' + outNameErase)
```

```
                # MBR polygon to line
                outNamePolToLine = 'block_' + newBasename[:6] +
'_MBR_PolygonToLine' + sep + osuf
                outFlPolToLine = polToLineDir + '\\' + outNamePolToLine
                ap.management.PolygonToLine(outFlMBR, outFlPolToLine)
                print('MBR polygon ' + outNameMBR + ' transformed to line feature:
' + outNamePolToLine)

                # buffer around line (created in previous step)
                outNameBuffer = 'block_' + newBasename[:6] + '_MBR_Line_Buffer' +
sep + osuf
                outFlBuffer = bufferDir + '\\' + outNameBuffer
                bufferDistance = '350 Meters'
                ap.analysis.Buffer(outFlPolToLine, outFlBuffer, bufferDistance)
                print('buffer around line feature ' + outNamePolToLine + ' created
as: ' + outNameBuffer)

                # erase erase
                outNameEraseErase = 'block_' + newBasename[:6] + '_Erase_Erase' +
sep + osuf
                outFlEraseErase = eraseEraseDir + '\\' + outNameEraseErase
                ap.analysis.Erase(outFlErase, outFlBuffer, outFlEraseErase)
                print('erased area delimited by buffer feature ' + outNameBuffer +
' from the first erased feature ' + outNameErase + ' and created: ' +
outNameEraseErase)

                # multipart to singlepart and calculate Area attribute
                outNameMultiToSingle = 'block_' + newBasename[:6] +
'_Erase_Erase_single' + sep + osuf
                outFlMultiToSingle = multiToSingleDir + '\\' +
outNameMultiToSingle
                ap.management.MultipartToSinglepart(outFlEraseErase,
outFlMultiToSingle)
                count = ap.management.GetCount(outFlMultiToSingle)
                count = int(str(count))
                print('count of single features: ', count)
                if count != 0:
                    ap.management.CalculateGeometryAttributes(outFlMultiToSingle,
[['Area', 'AREA']])
                    print('attribute Area calculated')
                else:
                    print('attribute Area not calculated because count of single
features is: ', count)
                print('multipart feature ' + outNameEraseErase + ' transformed to
singlepart as: ' + outNameMultiToSingle)

                # clip water
                outNameClip = 'block_' + newBasename[:6] + '_water' + sep + osuf
                outFlClip = clipDir + '\\' + outNameClip
                ap.analysis.Clip(water, image, outFlClip)
```

```
                print('water clipped to extent of images of block, clipped water
saved as: ' + outNameClip)

                # water multipart to singlepart
                outNameWaterMultiToSingle = 'block_' + newBasename[:6] +
'_water_single' + sep + osuf
                outFlWaterMultiToSingle = waterMultiToSingleDir + '\\' +
outNameWaterMultiToSingle
                ap.management.MultipartToSinglepart(outFlClip,
outFlWaterMultiToSingle)
                print('multipart water feature ' + outNameClip + ' transformed to
singlepart as: ' + outNameWaterMultiToSingle)

                # selections
                selected = ap.management.SelectLayerByLocation(outFlMultiToSingle,
'INTERSECT', outFlWaterMultiToSingle, '', 'NEW_SELECTION', 'INVERT')
                ap.management.SelectLayerByLocation(selected, 'CONTAINS',
outFlWaterMultiToSingle, '', 'ADD_TO_SELECTION')
                if count != 0:
                    sql = 'Area <= 20000 OR Area >= 10000000'
                    ap.management.SelectLayerByAttribute(selected,
'REMOVE_FROM_SELECTION', sql)
                outNameSelectionErase = 'block_' + newBasename[:6] +
'_Erase_Erase_correct' + sep + osuf
                outFlSelectionErase = selectionEraseDir + '\\' +
outNameSelectionErase
                ap.management.CopyFeatures(selected, outFlSelectionErase)
                print('selected correct empty places as: ' +
outNameSelectionErase)

                # final images selection
                searchDistance = '-100 Meters'
                selected = ap.management.SelectLayerByLocation(image, 'INTERSECT',
outFlSelectionErase, searchDistance)
                outNameFinalImgSelection = 'block_' + newBasename[:6] +
'_images_to_correlate' + sep + osuf
                outBlockDir = opath + newBasename[:4] + '\\' + newBasename[:6]
                os.makedirs(outBlockDir)
                outFlFinalImgSelection = outBlockDir + '\\' +
outNameFinalImgSelection
                ap.management.CopyFeatures(selected, outFlFinalImgSelection)
                print('FINAL IMAGES TO CORRELATE SAVED AS: ' +
outNameFinalImgSelection)

                # make list of images to correlate
                count = ap.management.GetCount(outFlFinalImgSelection)
                count = int(str(count))
                print('count of images to correlate: ', count)
                if count != 0:
                    ap.management.CalculateField(outFlFinalImgSelection, 'block',
newBasename[:6])
```

```
                    print('attribute block added')

                    tmpOut = opath + 'tmpOut' + sep + txtsuf
                    tmpOutXml = tmpOut + sep + xmlsuf
                    imgsLst = opath + 'imgsLst' + sep + txtsuf
                    ap.stats.ExportXYv(outFlFinalImgSelection, ['NAME', 'block'],
'SPACE', tmpOut)
                    x = open(tmpOut, 'r')
                    y = open(imgsLst, 'a')
                    for line in x:
                        if line.strip():
                            y.write(' '.join(line.split()[2:]) + '\n')
                    x.close()
                    y.close()
                    os.remove(tmpOut)
                    os.remove(tmpOutXml)
                    print('name of image and number of block added to file: ',
imgsLst)
                else:
                    print('attribute block not added because count of images to
correlate is: ', count)

                print('-------------------------------------------------------------
-')
            except:
                pass

    shutil.rmtree(copyDir)
    shutil.rmtree(mbrDir)
    shutil.rmtree(eraseDir)
    shutil.rmtree(polToLineDir)
    shutil.rmtree(bufferDir)
    shutil.rmtree(eraseEraseDir)
    # shutil.rmtree(multiToSingleDir) # will be deleted in next standalone script
    shutil.rmtree(clipDir)
    shutil.rmtree(waterMultiToSingleDir)
    shutil.rmtree(selectionEraseDir)
    print('temporary directories deleted')

    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 8: Skript 04_x_delete_remaining_temporary_folder.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 04_x_delete_remaining_temporary_folder.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   from datetime import datetime
   import os
   import shutil

   curpath = os.path.dirname(os.getcwd())
   opath = curpath +
'\\04_stereopairs_laz_UHUL_images_to_correlate_for_missing_stereopairs\\'

   start = datetime.now()

   try:
       multiToSingleDir = opath + '07_multiToSingle'
       shutil.rmtree(multiToSingleDir)
       print(multiToSingleDir, ' also deleted')
   except:
       pass

   print('=======================END OF SCRIPT!=======================')

   end = datetime.now()
   print('time elapsed: ', end - start)
```

Příloha 9: Skript 05_repair_laz_all.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 05_repair_laz_all.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\06_stereopairs_laz_UHUL_images_added_missing_images\\'
   opath = curpath + '\\07_stereopairs_laz_UHUL_all_images_repaired\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('----------------------------------------------------------------')

   isuf = 'laz'
   osuf = 'laz'
   sep = '.'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   las2las = lastools_base_path + lastools_version + '\\bin\\las2las64.exe'

   las2las_settings = '-auto_reoffset -rescale 0.01 0.01 0.01 -
repair_zero_returns'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)
           print('files: ', fls)
           for fl in fls:
               try:
```

```
                    os.makedirs(opath + year + '\\' + block)
                    print('created folder: ', opath + year + '\\' + block)
                except:
                    pass

                fl = fl.replace('.laz', '')
                execute = las2las + ' -i ' + ipath + year + '\\' + block + '\\' +
fl + sep + isuf + ' -odir ' + opath + year + '\\' + block + ' -o' + osuf + ' ' +
las2las_settings
                print(execute)
                subprocess.call(execute, shell=True)

            print('-------------------------------------------------------------')
        print('=============================================================')
    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 10: Skript 06_lasboundary_all.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 06_lasboundary_all.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------


   import arcpy as ap
   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\07_stereopairs_laz_UHUL_all_images_repaired\\'
   opath = curpath + '\\08_stereopairs_laz_UHUL_all_images_lasboundary\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'laz'
   osuf = 'shp'
   sep = '.'
   epsg5514 = 'PROJCS[\"S-
JTSK_Krovak_East_North\",GEOGCS[\"GCS_S_JTSK\",DATUM[\"D_S_JTSK\",SPHEROID[\"Bess
el_1841\",6377397.155,299.1528128]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0
174532925199433]],PROJECTION[\"Krovak\"],PARAMETER[\"False_Easting\",0.0],PARAMET
ER[\"False_Northing\",0.0],PARAMETER[\"Pseudo_Standard_Parallel_1\",78.5],PARAMET
ER[\"Scale_Factor\",0.9999],PARAMETER[\"Azimuth\",30.28813975277778],PARAMETER[\"
Longitude_Of_Center\",24.83333333333333],PARAMETER[\"Latitude_Of_Center\",49.5],P
ARAMETER[\"X_Scale\",-
1.0],PARAMETER[\"Y_Scale\",1.0],PARAMETER[\"XY_Plane_Rotation\",90.0],UNIT[\"Mete
r\",1.0]]'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   lasboundary = lastools_base_path + lastools_version +
'\\bin\\lasboundary64.exe'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
```

```
        print('blocks: ', blocks)
        for block in blocks:
            print('block: ', block)
            fls = os.listdir(ipath + year + '\\' + block)
            print('files: ', fls)

            for fl in fls:
                try:
                    os.makedirs(opath + year + '\\' + block)
                    print('created folder: ', opath + year + '\\' + block)
                except:
                    pass

                fl = fl.replace('.laz', '')
                execute = lasboundary + ' -i ' + ipath + year + '\\' + block +
'\\' + fl + sep + isuf + ' -odir ' + opath + year + '\\' + block + ' -o' + osuf
                print(execute)
                subprocess.call(execute, shell=True)

                # define projection EPSG 5514 to fl.shp
                ap.env.workspace = opath + year + '\\' + block
                outputfl = fl + sep + osuf
                ap.management.DefineProjection(outputfl, epsg5514)
                print('projection EPSG 5514 defined to output lasboundary file: ',
outputfl)

                # add field to fl.shp
                ap.management.AddField(outputfl, 'name', 'TEXT')
                print("field 'name' added to output lasboundary file: ", outputfl)

                # set the file name as the name attribute
                ap.management.CalculateField(outputfl, 'name', 'fl', 'PYTHON3')
                print("attribute 'name' set to: ", fl)

            print('-------------------------------------------------------------')
        print('=============================================================')
    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 11: Skript 07_block_merge_all.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 07_block_merge_all.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\08_stereopairs_laz_UHUL_all_images_lasboundary\\'
   opath = curpath +
'\\09_stereopairs_laz_UHUL_all_images_block_merged_boundary\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('----------------------------------------------------------------')

   isuf = 'shp'
   osuf = 'shp'
   sep = '.'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)

           inputs = []
           shps = []
           for fl in fls:
               try:
                   os.makedirs(opath + year + '\\' + block)
                   print('created folder: ', opath + year + '\\' + block)
               except:
                   pass
```

```
            if fl[-3:] == isuf:
                shps.append(fl)
                fl = fl.replace(fl, ipath + year + '\\' + block + '\\' + fl)
                inputs.append(fl)

        print('boundaries to be merged: ', shps)

        # merge files from inputs list
        ap.env.workspace = opath + year + '\\' + block
        outputfl = block + '_block_merged_boundary' + sep + osuf
        ap.management.Merge(inputs, outputfl)
        print('boundaries merged to file: ', outputfl)

        print('------------------------------------------------------------')
    print('============================================================')
print('=======================END OF SCRIPT!=======================')

end = datetime.now()
print('time elapsed: ', end - start)
```

Příloha 12: Skript 08_centroids.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 08_centroids.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath +
'\\09_stereopairs_laz_UHUL_all_images_block_merged_boundary\\'
   opath = curpath +
'\\10_stereopairs_laz_UHUL_all_images_centroids_of_boundaries\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'shp'
   osuf = 'shp'
   sep = '.'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)

           for fl in fls:
               try:
                   os.makedirs(opath + year + '\\' + block)
                   print('created folder: ', opath + year + '\\' + block)
               except:
                   pass

               if fl[-3:] == isuf:
```

```
                    altfl = fl.replace('block_merged_boundary',
'block_boundaries_centroids')

                    # make centroids of every boundary in merged boundaries
                    ap.env.workspace = ipath + year + '\\' + block
                    outputfl = opath + year + '\\' + block + '\\' + altfl
                    ap.management.FeatureToPoint(fl, outputfl)
                    print('boundary centroids of the whole block made to file: ',
altfl)

            print('------------------------------------------------------------')
        print('============================================================')
    print('=======================END OF SCRIPT!=======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 13: Skript 09_thiessen_polygons.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 09_thiessen_polygons.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os
   # import shutil

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath +
'\\10_stereopairs_laz_UHUL_all_images_centroids_of_boundaries\\'
   opath = curpath + '\\11_stereopairs_laz_UHUL_all_images_thiessen_polygons\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf = 'shp'
   osuf = 'shp'
   sep = '.'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           fls = os.listdir(ipath + year + '\\' + block)

           for fl in fls:
               try:
                   os.makedirs(opath + year + '\\' + block)
                   print('created folder: ', opath + year + '\\' + block)
               except:
                   pass

               if fl[-3:] == isuf:
```

```
                    # folderWithTmp = opath + year + '\\' + block + '\\' + 'tmp'
                    # os.makedirs(folderWithTmp)
                    altfl = fl.replace('block_boundaries_centroids',
'block_thiessen_polygons')

                    # make thiessen polygons around centroids of boundaries
                    ap.env.workspace = ipath + year + '\\' + block
                    outputfl = opath + year + '\\' + block + '\\' + altfl
                    ap.analysis.CreateThiessenPolygons(fl, outputfl, 'ALL')
                    print('thiessen polygons around centroids of boundaries made
to file : ', altfl)

                    # # make buffers around thiessen polygons
                    # bufferAltFl = altfl.replace('block_thiessen_polygons',
'block_thiessen_polygons_buffer')
                    # bufferOutFl = opath + year + '\\' + block + '\\' +
bufferAltFl
                    # ap.analysis.Buffer(outputfl, bufferOutFl, '500 Meters')
                    # print('buffers around thiessen polygons around centroids of
boundaries made to file : ', bufferAltFl)
                    #
                    # # delete temporary folder with thiessen polygons
                    # shutil.rmtree(folderWithTmp)

            print('------------------------------------------------------------')
        print('============================================================')
    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 14: Skript 10_lasclip.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 10_lasclip.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------


   import arcpy as ap
   from datetime import datetime
   import ntpath
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath_laz = curpath + '\\07_stereopairs_laz_UHUL_all_images_repaired\\'
   ipath_thiessen = curpath +
'\\11_stereopairs_laz_UHUL_all_images_thiessen_polygons\\'
   opath = curpath + '\\12_stereopairs_laz_UHUL_all_images_clipped_laz_files\\'
   print('laz input folder: ', ipath_laz)
   print('thiessen input folder: ', ipath_thiessen)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   isuf_laz = 'laz'
   isuf_shp = 'shp'
   osuf = 'laz'
   sep = '.'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   lasclip = lastools_base_path + lastools_version + '\\bin\\lasclip64.exe'

   start = datetime.now()

   lstOfLst = []
   years = os.listdir(ipath_laz)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath_laz + year)
       print('blocks: ', blocks)
       allFiles = []
       for block in blocks:
```

```
            print('block: ', block)
            fls = os.listdir(ipath_laz + year + '\\' + block)
            blockFiles = []
            for fl in fls:
                flPath = ipath_laz + year + '\\' + block + '\\' + fl
                blockFiles.append(flPath)
            allFiles.append(blockFiles)
        lstOfLst.append(allFiles)
    print('all blocks and their laz files:')
    print(lstOfLst)

    print('------------------------------------------------------------')

    lstOfLst1 = []
    years1 = os.listdir(ipath_thiessen)
    # print('years: ', years1)
    for year1 in years1:
        # print('year: ', year1)
        blocks1 = os.listdir(ipath_thiessen + year1)
        # print('blocks: ', blocks1)
        allFiles1 = []
        for block1 in blocks1:
            # print('block: ', block1)
            fls1 = os.listdir(ipath_thiessen + year1 + '\\' + block1)
            for fl1 in fls1:
                if fl1[-3:] == isuf_shp:
                    flPath1 = ipath_thiessen + year1 + '\\' + block1 + '\\' + fl1
                    allFiles1.append(flPath1)
        lstOfLst1.append(allFiles1)
    print('all blocks and their thiessen polygons:')
    print(lstOfLst1)

    print('============================================================')

    for lazFiles_all, thiessens in zip(lstOfLst, lstOfLst1):
        for lazFiles, thiessen in zip(lazFiles_all, thiessens):
            try:
                os.makedirs(opath)
                print('created folder: ', opath)
            except:
                pass

            try:
                for lazFile in lazFiles:
                    origBasename = ntpath.basename(lazFile)
                    newBaseName = origBasename.replace(sep + isuf_laz, '')
                    print('laz file: ', newBaseName, '; block of thiessen
polygons: ', thiessen)
```

```
                    # create clipping geometry for each laz file from block of
thiessen polygons
                    sql = "name = '" + newBaseName + "'"
                    selected = ap.management.SelectLayerByAttribute(thiessen,
'NEW_SELECTION', sql)
                    clipShp = opath + newBaseName + '_for_clip' + sep + isuf_shp
                    ap.management.CopyFeatures(selected, clipShp)
                    print('clipping shapefile ' + clipShp + ' successfully
created')

                    # create output folders
                    outFolder = str(ntpath.dirname(lazFile).split('\\')[-2]) +
'\\' + str(ntpath.dirname(lazFile).split('\\')[-1]) + '\\'
                    try:
                        os.makedirs(opath + outFolder)
                        print('created folder: ', opath + outFolder)
                    except:
                        pass

                    # lasclip
                    execute = lasclip + ' -i ' + lazFile + ' -poly ' + clipShp + '
-o ' + opath + outFolder + newBaseName + '_clipped' + sep + osuf
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('laz file ', newBaseName, ' successfully clipped and
saved!')

                    # delete clipping geometry after clipping
                    ap.management.Delete(clipShp)
                    print('clipping shapefile successfully deleted')

                    print('----------------------------------------------------------
-----')
            except:
                pass
            print('============================================================')

    print('=======================END OF SCRIPT!=======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 15: Skript 11_make_dsm_in_sm5.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 11_make_dsm_in_sm5.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath_block = curpath +
'\\11_stereopairs_laz_UHUL_all_images_thiessen_polygons\\'
   ipath_laz_files = curpath +
'\\12_stereopairs_laz_UHUL_all_images_clipped_laz_files\\'
   opath_SM5 = curpath +
'\\13_stereopairs_laz_UHUL_lists_of_thiessen_polygons_for_corresponding_SM5\\'
   opath_dsm = curpath + '\\14_stereopairs_laz_UHUL_dsm_before_repair\\'
   dbpath = curpath + '\\DB\\'
   print('input block folder: ', ipath_block)
   print('input laz files folder: ', ipath_laz_files)
   print('output SM5 folder: ', opath_SM5)
   print('output DSM folder: ', opath_dsm)
   print('------------------------------------------------------------')

   shpsuf = 'shp'
   tifsuf = 'tif'
   lazsuf = 'laz'
   txtsuf = 'txt'
   sep = '.'
   dbname = 'all_images_and_sm5.gdb'
   dbSM5Layer = dbpath + dbname + '\\' + 'SM5_5514'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   lasclip = lastools_base_path + lastools_version + '\\bin\\lasclip64.exe'
   # las2dem = lastools_base_path + lastools_version + '\\bin\\las2dem64.exe'
   # las2dem_step = '-step 1 '
   # las2dem_step_text = '_step_1'

   start = datetime.now()
```

```
    years = os.listdir(ipath_block)
    print('years: ', years)
    for year in years:
        print('year: ', year)
        blocks = os.listdir(ipath_block + year)
        print('blocks: ', blocks)
        for block in blocks:
            print('block: ', block)
            fls = os.listdir(ipath_block + year + '\\' + block)

            for fl in fls:
                try:
                    os.makedirs(opath_SM5 + year + '\\' + block)
                    print('created folder: ', opath_SM5 + year + '\\' + block)
                except:
                    pass

                if fl[-3:] == shpsuf:
                    altfl = fl.replace('block_thiessen_polygons', 'block_SM5')

                    ap.env.workspace = ipath_block + year + '\\' + block
                    selected = ap.management.SelectLayerByLocation(dbSM5Layer,
'INTERSECT', fl)
                    selected_SM5_for_block = opath_SM5 + year + '\\' + block +
'\\' + altfl
                    ap.management.CopyFeatures(selected, selected_SM5_for_block)
                    print('SM5 shapefile ' + selected_SM5_for_block + '
successfully created')

            print('------------------------------------------------------------')
        print('============================================================')
    print('============================================================')

    years = os.listdir(opath_SM5)
    print('years: ', years)
    for year in years:
        print('year: ', year)
        blocks = os.listdir(opath_SM5 + year)
        print('blocks: ', blocks)
        for block in blocks:
            print('block: ', block)
            fls = os.listdir(opath_SM5 + year + '\\' + block)

            for fl in fls:
                try:
                    os.makedirs(opath_dsm + year + '\\' + block)
                    print('created folder: ', opath_dsm + year + '\\' + block)
                except:
                    pass
```

```
            if fl[-3:] == shpsuf:

                ap.env.workspace = opath_SM5 + year + '\\' + block
                field = 'SM5_NAME'
                with ap.da.SearchCursor(fl, field) as cursor:
                    for row in cursor:
                        print('SM5: ', row[0])

                        # make one shapefile for each SM5 in given block
                        sql = field + " = '" + str(row[0]) + "'"
                        selected = ap.management.SelectLayerByAttribute(fl,
'NEW_SELECTION', sql)
                        one_SM5 = ap.env.workspace + '\\' + str(block) + '_' +
str(row[0]) + sep + shpsuf
                        ap.management.CopyFeatures(selected, one_SM5)
                        print('corresponding SM5 shapefile ', one_SM5, '
created')

                        # select those thiessen polygons that intersect with
corresponding SM5
                        block_name = fl.replace('block_SM5',
'block_thiessen_polygons')
                        block_file = ipath_block + year + '\\' + block + '\\'
+ block_name
                        selected =
ap.management.SelectLayerByLocation(block_file, 'INTERSECT', one_SM5)
                        print('laz files in SM5 ', str(row[0]), ' selected')

                        # create txt file with whole paths to LAZ files
clipped by thiessen polygons in corresponding SM5
                        with ap.da.SearchCursor(selected, 'name') as
block_cursor:
                            for block_row in block_cursor:
                                laz_files_in_SM5_file = ap.env.workspace +
'\\' + str(block) + '_' + str(row[0]) + '_files' + sep + txtsuf
                                x = open(laz_files_in_SM5_file, 'a')
                                x.write(ipath_laz_files + year + '\\' + block
+ '\\' + str(block_row[0]) + '_clipped' + sep + lazsuf + '\n')
                                x.close()
                            print('laz files in SM5 ', str(row[0]), ' written
to file ', laz_files_in_SM5_file)

                        # lasclip and merge
                        execute = lasclip + ' -lof ' + laz_files_in_SM5_file +
' -merged -poly ' + one_SM5 + ' -odir ' + opath_dsm + year + '\\' + block + ' -o
' + str(block) + '_' + str(row[0]) + sep + lazsuf
                        print('execute: ', execute)
                        subprocess.call(execute, shell=True)
```

```
                            print('laz files for corresponding SM5 ', str(row[0]),
' successfully clipped, merged and saved as ', str(block) + '_' + str(row[0]) +
sep + lazsuf)

                            # # las2dem DSM elevation
                            # final_laz = opath_dsm + year + '\\' + block + '\\' +
str(block) + '_' + str(row[0]) + sep + lazsuf
                            # if os.path.exists(final_laz):
                            #     if not os.path.exists(opath_dsm + year + '\\' +
block + '\\' + str(block) + '_' + str(row[0]) + las2dem_step_text + sep +
tifsuf):
                            #         execute = las2dem + ' -i ' + final_laz + ' -
elevation ' + las2dem_step + '-odir ' + opath_dsm + year + '\\' + block + ' -o '
+ str(block) + '_' + str(row[0]) + las2dem_step_text + sep + tifsuf
                            #         print('execute: ', execute)
                            #         subprocess.call(execute, shell=True)
                            #         print('DSM tif file for SM5 ', str(row[0]),
' successfully created as ', str(block) + '_' + str(row[0]) + las2dem_step_text +
sep + tifsuf)
                            #
                            # # las2dem DSM hillshade
                            # if os.path.exists(final_laz):
                            #     if not os.path.exists(opath_dsm + year + '\\' +
block + '\\' + str(block) + '_' + str(row[0]) + '_hillshade' + las2dem_step_text
+ sep + tifsuf):
                            #         execute = las2dem + ' -i ' + final_laz + ' -
hillshade ' + las2dem_step + '-odir ' + opath_dsm + year + '\\' + block + ' -o '
+ str(block) + '_' + str(row[0]) + '_hillshade' + las2dem_step_text + sep +
tifsuf
                            #         print('execute: ', execute)
                            #         subprocess.call(execute, shell=True)
                            #         print('DSM hillshade tif file for SM5 ',
str(row[0]), ' successfully created as ', str(block) + '_' + str(row[0]) +
'_hillshade' + las2dem_step_text + sep + tifsuf)

                            print('----------------------------------------------
-------------')

            print('------------------------------------------------------------')
        print('============================================================')

    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 16: Skript 12_final_dsm.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 12_final_dsm.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import ntpath
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath_SM5 = curpath +
'\\13_stereopairs_laz_UHUL_lists_of_thiessen_polygons_for_corresponding_SM5\\'
   ipath_dsm = curpath + '\\14_stereopairs_laz_UHUL_dsm_before_repair\\'
   opath_water_SM5 = curpath +
'\\15_stereopairs_laz_UHUL_water_in_corresponding_SM5\\'
   opath_final_dsm = curpath + '\\16_stereopairs_laz_UHUL_FINAL_DSM\\'
   waterpath = curpath + '\\Vody\\'
   print('input SM5 folder: ', ipath_SM5)
   print('input DSM before repair folder: ', ipath_dsm)
   print('output water in SM5 folder: ', opath_water_SM5)
   print('output final DSM folder: ', opath_final_dsm)
   print('------------------------------------------------------------')

   shpsuf = 'shp'
   tifsuf = 'tif'
   lazsuf = 'laz'
   sep = '.'
   water = waterpath + 'doplnena_voda_sjtsk_bpv.shp'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   lasclip = lastools_base_path + lastools_version + '\\bin\\lasclip64.exe'
   las2dem = lastools_base_path + lastools_version + '\\bin\\las2dem64.exe'
   lasmerge = lastools_base_path + lastools_version + '\\bin\\lasmerge64.exe'
   las2dem_step = ' -step 1 '
   las2dem_step_text = '_step_1'

   start = datetime.now()

   years = os.listdir(ipath_SM5)
```

```
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath_SM5 + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)

           # make list of really used SM5s and only for these SM5 make water shp
file
           used_SM5_in_block = []
           years1 = os.listdir(ipath_dsm)
           for year1 in years1:
               blocks1 = os.listdir(ipath_dsm + year1)
               for block1 in blocks1:
                   fls1 = os.listdir(ipath_dsm + year1 + '\\' + block1)
                   for fl1 in fls1:
                       if fl1[-3:] == lazsuf:
                           used_SM5_in_block.append(fl1[:-4])
           # print('used SM5 in block: ', used_SM5_in_block)

           fls = os.listdir(ipath_SM5 + year + '\\' + block)

           for fl in fls:
               try:
                   os.makedirs(opath_water_SM5 + year + '\\' + block)
                   print('created folder: ', opath_water_SM5 + year + '\\' +
block)
               except:
                   pass
               try:
                   if fl[-3:] == shpsuf and fl[-7:-4] == 'SM5':
                       altfl = fl.replace('SM5', 'SM5_water')
                       block_water = opath_water_SM5 + year + '\\' + block + '\\'
+ altfl
                       ap.env.workspace = ipath_SM5 + year + '\\' + block
                       ap.analysis.Clip(water, fl, block_water)
                       print('water clipped to current block ', block, ' and
successfully saved as ', altfl)
               except:
                   pass

           for fl in fls:
               if fl[-3:] == shpsuf and fl[-7:-4] != 'SM5' and block_water[-3:]
== shpsuf and fl[:-4] in used_SM5_in_block:
                   altfl = fl.replace(sep + shpsuf, '_water' + sep + shpsuf)

                   SM5_water = opath_water_SM5 + year + '\\' + block + '\\' +
altfl
                   ap.env.workspace = ipath_SM5 + year + '\\' + block
```

```
                    ap.analysis.Clip(block_water, fl, SM5_water)
                    print('water clipped to extent of ', fl[7:13], ' SM5 and
corresponding file ', altfl, ' successfully created')

            print('------------------------------------------------------------')
        print('============================================================')
    print('============================================================')

    # traverse through SM5 laz files before repair (14) and add them to the list
    years = os.listdir(ipath_dsm)
    print('years: ', years)
    lstOfLst = []
    for year in years:
        print('year: ', year)
        blocks = os.listdir(ipath_dsm + year)
        print('blocks: ', blocks)
        allFiles = []
        for block in blocks:
            print('block: ', block)
            fls = os.listdir(ipath_dsm + year + '\\' + block)
            SM5_laz_before_repair_Files = []
            for fl in fls:
                if fl[-3:] == lazsuf:
                    flPath = ipath_dsm + year + '\\' + block + '\\' + fl
                    SM5_laz_before_repair_Files.append(flPath)
            allFiles.append(SM5_laz_before_repair_Files)
        lstOfLst.append(allFiles)
    print('all blocks and their SM5 laz files before repair:')
    print(lstOfLst)

    print('============================================================')

    # traverse through SM5 water shp files (15) and add them to the list
    years1 = os.listdir(opath_water_SM5)
    # print('years: ', years1)
    lstOfLst1 = []
    for year1 in years1:
        # print('year: ', year1)
        blocks1 = os.listdir(opath_water_SM5 + year1)
        # print('blocks: ', blocks1)
        allFiles1 = []
        for block1 in blocks1:
            # print('block: ', block1)
            fls1 = os.listdir(opath_water_SM5 + year1 + '\\' + block1)
            SM5_water_Files = []
            for fl1 in fls1:
                if fl1[-3:] == shpsuf and fl1[13:16] != 'SM5':
                    flPath1 = opath_water_SM5 + year1 + '\\' + block1 + '\\' + fl1
                    SM5_water_Files.append(flPath1)
```

```
            allFiles1.append(SM5_water_Files)
        lstOfLst1.append(allFiles1)
    print('all blocks and their water shp files for each SM5:')
    print(lstOfLst1)

    print('============================================================')

    for all_SM5_laz_files, all_SM5_water_files in zip(lstOfLst, lstOfLst1):
        for year_SM5_laz_files, year_SM5_water_files in zip(all_SM5_laz_files,
all_SM5_water_files):
            for block_SM5_laz_files, block_SM5_water_files in
zip(year_SM5_laz_files, year_SM5_water_files):
                print('input SM5 laz file before repair: ', block_SM5_laz_files,
'; input SM5 water shp file: ', block_SM5_water_files)
                whole_opath = opath_final_dsm +
ntpath.basename(block_SM5_laz_files)[:4] + '\\' +
ntpath.basename(block_SM5_laz_files)[:6] + '\\'
                # print(whole_opath)
                try:
                    os.makedirs(whole_opath)
                    print('created folder: ', whole_opath)
                except:
                    pass

                count = ap.management.GetCount(block_SM5_water_files)
                count = int(str(count))
                outFl_FINAL = whole_opath +
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL' + sep + lazsuf
                if count != 0:
                    # lasclip - remove areas with water from laz
                    print('--- LASCLIP - remove areas with water from laz')
                    outFl_without_water = whole_opath +
ntpath.basename(block_SM5_laz_files)[:-4] + '_without_water' + sep + lazsuf
                    execute = lasclip + ' -i ' + block_SM5_laz_files + ' -poly ' +
block_SM5_water_files + ' -interior -quiet -o ' + outFl_without_water
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('laz file without water for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_without_water' + sep + lazsuf)

                    # las2dem - interpolate water areas from water shp file
                    print('--- LAS2DEM - interpolate water areas from water shp
file')
                    outFl_interpolated = whole_opath +
ntpath.basename(block_SM5_laz_files)[:-4] + '_interpolated' + sep + lazsuf
                    execute = las2dem + ' -i ' + outFl_without_water + ' -lakes '
+ block_SM5_water_files + las2dem_step + '-o ' + outFl_interpolated
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
```

```
                    print('interpolated laz file for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_interpolated' + sep + lazsuf)

                    # lasclip - clip the interpolated water from the interpolated
laz file -> make laz only with interpolated water
                    print('--- LASCLIP - clip the interpolated water from the
interpolated laz file -> make laz only with interpolated water')
                    outFl_water = whole_opath +
ntpath.basename(block_SM5_laz_files)[:-4] + '_water' + sep + lazsuf
                    execute = lasclip + ' -i ' + outFl_interpolated + ' -poly ' +
block_SM5_water_files + ' -quiet -o ' + outFl_water
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('laz file only with interpolated water for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_water' + sep + lazsuf)

                    if os.path.exists(outFl_water):
                        # lasmerge - merge laz without water and laz only with
water
                        print('--- LASMERGE - merge laz without water and laz only
with water')
                        execute = lasmerge + ' -i ' + outFl_without_water + ' ' +
outFl_water + ' -o ' + outFl_FINAL
                        print('execute: ', execute)
                        subprocess.call(execute, shell=True)
                        print('FINAL laz file with repaired water for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL' + sep + lazsuf)
                    else:
                        # rename - water in given SM5 does not intersect point
cloud in laz file -> water exists but has no effect -> without_water will be
renamed to FINAL
                        print('--- RENAME - water in given SM5 does not intersect
point cloud in laz file -> water exists but has no effect -> without_water will
be renamed to FINAL')
                        os.rename(outFl_without_water, outFl_FINAL)
                        print('FINAL laz file for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL' + sep + lazsuf)

                    # delete intermediate products
                    if os.path.exists(outFl_without_water):
                        os.remove(outFl_without_water)
                    if os.path.exists(outFl_interpolated):
                        os.remove(outFl_interpolated)
                    if os.path.exists(outFl_water):
                        os.remove(outFl_water)

                else:
                    execute = 'copy ' + block_SM5_laz_files + ' ' + outFl_FINAL
```

```
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('no water in SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' , input file copied as output
FINAL file ', ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL' + sep +
lazsuf)

                # las2dem FINAL DSM laz to tif
                print('--- LAS2DEM - FINAL DSM laz to tif')
                outFl_FINAL_tif = whole_opath +
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL_DSM' + las2dem_step_text +
sep + tifsuf
                execute = las2dem + ' -i ' + outFl_FINAL + ' -elevation' +
las2dem_step + '-o ' + outFl_FINAL_tif
                print('execute: ', execute)
                subprocess.call(execute, shell=True)
                print('FINAL DSM tif file for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL_DSM' + las2dem_step_text +
sep + tifsuf)

                # las2dem FINAL DSM laz to hillshade tif
                print('--- LAS2DEM - FINAL DSM laz to hillshade tif')
                outFl_FINAL_hillshade_tif = whole_opath +
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL_DSM_hillshade' +
las2dem_step_text + sep + tifsuf
                execute = las2dem + ' -i ' + outFl_FINAL + ' -hillshade' +
las2dem_step + '-o ' + outFl_FINAL_hillshade_tif
                print('execute: ', execute)
                subprocess.call(execute, shell=True)
                print('FINAL DSM hillshade tif file for SM5 ',
ntpath.basename(block_SM5_laz_files)[-10:-4], ' successfully created as ',
ntpath.basename(block_SM5_laz_files)[:-4] + '_FINAL_DSM_hillshade' +
las2dem_step_text + sep + tifsuf)

                print('------------------------------------------------------------
-')
            print('============================================================')
        print('============================================================')
    print('=======================END OF SCRIPT!=======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 17: Skript 13_classification.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 13_classification.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------

   import arcpy as ap
   from datetime import datetime
   import os
   import shutil
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath_SM5 = curpath +
'\\13_stereopairs_laz_UHUL_lists_of_thiessen_polygons_for_corresponding_SM5\\'
   ipath_laz = curpath + '\\16_stereopairs_laz_UHUL_FINAL_DSM\\'
   opath_bridges_SM5 = curpath + '\\17_1_stereopairs_laz_UHUL_bridges_in_SM5s\\'
   opath_buildings_SM5 = curpath +
'\\17_2_stereopairs_laz_UHUL_buildings_in_SM5s\\'
   opath_forests_SM5 = curpath + '\\17_3_stereopairs_laz_UHUL_forests_in_SM5s\\'
   opath_final_classified_laz = curpath +
'\\17_stereopairs_laz_UHUL_CLASSIFICATION\\'
   dtm_path = curpath + '\\BPR\\'
   ruian_path = curpath + '\\RUIAN\\'
   zabaged_path = curpath + '\\ZABAGED\\'
   print('input SM5 folder: ', ipath_SM5)
   print('input laz folder: ', ipath_laz)
   print('output bridges in SM5s folder: ', opath_bridges_SM5)
   print('output buildings in SM5s folder: ', opath_buildings_SM5)
   print('output forests in SM5s folder: ', opath_forests_SM5)
   print('output classification folder: ', opath_final_classified_laz)
   print('------------------------------------------------------------')

   shpsuf = 'shp'
   lazsuf = 'laz'
   sep = '.'
   buildings = ruian_path + '__RUIAN_ZABAGED_budovy_CZ.shp'
   bridges = zabaged_path + 'ZABAGED_mostovky_poly.shp'
   forests = zabaged_path + 'ZABAGED_lesni_puda_se_stromy.shp'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   las2las = lastools_base_path + lastools_version + '\\bin\\las2las64.exe'
```

```
   lasnoise = lastools_base_path + lastools_version + '\\bin\\lasnoise64.exe'
   lasheight = lastools_base_path + lastools_version + '\\bin\\lasheight64.exe'
   lasclip = lastools_base_path + lastools_version + '\\bin\\lasclip64.exe'
   lasclassify = lastools_base_path + lastools_version +
'\\bin\\lasclassify64.exe'

   ground_settings = ' -replace_z -store_in_user_data -all_ground_points -
classify_below -4.0 7 -classify_above 115.0 12 -classify_between -3.0 +3.0 2'
   bridges_settings = ' -classify 2 -interior -quiet -ignore_class 2 7 12'
   buildings_settings = ' -classify 6 -interior -quiet -ignore_class 2 7 12'
   forests_settings = ' -classify 5 -interior -quiet -ignore_class 2 6 7 12'
   vegetation_settings = ' -planar 0.001 -rugged 0.05 -no_gutters -small_trees -
ignore_class 2 5 6 7 12'

   start = datetime.now()

   years = os.listdir(ipath_SM5)
   print('years: ', years)
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath_SM5 + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)

           # make list of really used SM5s and only for these SM5 make
corresponding shp files
           used_SM5_in_block = []
           years1 = os.listdir(ipath_laz)
           for year1 in years1:
               blocks1 = os.listdir(ipath_laz + year1)
               for block1 in blocks1:
                   fls1 = os.listdir(ipath_laz + year1 + '\\' + block1)
                   for fl1 in fls1:
                       if fl1[-3:] == lazsuf:
                           used_SM5_in_block.append(fl1[:-4])
           # print('used SM5 in block: ', used_SM5_in_block)

           fls = os.listdir(ipath_SM5 + year + '\\' + block)

           for fl in fls:
               try:
                   os.makedirs(opath_bridges_SM5)
                   print('created folder: ', opath_bridges_SM5)
                   os.makedirs(opath_buildings_SM5)
                   print('created folder: ', opath_buildings_SM5)
                   os.makedirs(opath_forests_SM5)
                   print('created folder: ', opath_forests_SM5)
               except:
                   pass
```

```
                try:
                    if fl[-3:] == shpsuf and fl[-7:-4] == 'SM5':
                        altfl_bridges = fl.replace('SM5', 'SM5_bridges')
                        altfl_buildings = fl.replace('SM5', 'SM5_buildings')
                        altfl_forests = fl.replace('SM5', 'SM5_forests')
                        block_bridges = opath_bridges_SM5 + altfl_bridges
                        block_buildings = opath_buildings_SM5 + altfl_buildings
                        block_forests = opath_forests_SM5 + altfl_forests
                        ap.env.workspace = ipath_SM5 + year + '\\' + block
                        ap.analysis.Clip(bridges, fl, block_bridges)
                        print('bridges clipped to current block ', block, ' and
successfully saved as ', altfl_bridges)
                        ap.analysis.Clip(buildings, fl, block_buildings)
                        print('buildings clipped to current block ', block, ' and
successfully saved as ', altfl_buildings)
                        ap.analysis.Clip(forests, fl, block_forests)
                        print('forests clipped to current block ', block, ' and
successfully saved as ', altfl_forests)
                except:
                    pass

            for fl in fls:
                # bridges in one SM5
                if fl[-3:] == shpsuf and fl[-7:-4] != 'SM5' and block_bridges[-3:]
== shpsuf and (fl[:-4] + '_FINAL') in used_SM5_in_block:
                    altfl = fl.replace(sep + shpsuf, '_bridges' + sep + shpsuf)

                    SM5_bridges = opath_bridges_SM5 + altfl
                    ap.env.workspace = ipath_SM5 + year + '\\' + block
                    ap.analysis.Clip(block_bridges, fl, SM5_bridges)
                    print('bridges clipped to extent of ', fl[7:13], ' SM5 and
corresponding file ', altfl, ' successfully created')

                # buildings in one SM5
                if fl[-3:] == shpsuf and fl[-7:-4] != 'SM5' and block_buildings[-
3:] == shpsuf and (fl[:-4] + '_FINAL') in used_SM5_in_block:
                    altfl = fl.replace(sep + shpsuf, '_buildings' + sep + shpsuf)

                    SM5_buildings = opath_buildings_SM5 + altfl
                    ap.env.workspace = ipath_SM5 + year + '\\' + block
                    ap.analysis.Clip(block_buildings, fl, SM5_buildings)
                    print('buildings clipped to extent of ', fl[7:13], ' SM5 and
corresponding file ', altfl, ' successfully created')

                # forests in one SM5
                if fl[-3:] == shpsuf and fl[-7:-4] != 'SM5' and block_forests[-3:]
== shpsuf and (fl[:-4] + '_FINAL') in used_SM5_in_block:
                    altfl = fl.replace(sep + shpsuf, '_forests' + sep + shpsuf)

                    SM5_forests = opath_forests_SM5 + altfl
```

```
                    ap.env.workspace = ipath_SM5 + year + '\\' + block
                    ap.analysis.Clip(block_forests, fl, SM5_forests)
                    print('forests clipped to extent of ', fl[7:13], ' SM5 and
corresponding file ', altfl, ' successfully created')

            print('------------------------------------------------------------')
        print('============================================================')
    print('============================================================')

    years = os.listdir(ipath_laz)
    print('years: ', years)
    for year in years:
        print('year: ', year)
        blocks = os.listdir(ipath_laz + year)
        print('blocks: ', blocks)
        for block in blocks:
            print('block: ', block)
            fls = os.listdir(ipath_laz + year + '\\' + block)
            for fl in fls:
                try:
                    os.makedirs(opath_final_classified_laz + year + '\\' + block)
                    print('created folder: ', opath_final_classified_laz + year +
'\\' + block)
                except:
                    pass

                if fl[-3:] == lazsuf:
                    print('block: ', block, ' SM5: ', fl[7:13])

                    whole_ipath = ipath_laz + year + '\\' + block + '\\' + fl
                    whole_opath = opath_final_classified_laz + year + '\\' + block
+ '\\' + fl

                    # classify all points in given SM5 as class 1
                    print('--- CLASS 1 - LAS2LAS - classify all points in given
SM5 as class 1')
                    outFl_class_one = whole_opath[:-4] + '_cl1' + sep + lazsuf
                    execute = las2las + ' -i ' + whole_ipath + ' -
set_classification 1 -o ' + outFl_class_one
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('all points in SM5 ', fl[7:13], ' successfully
classified as class 1')

                    # classify noise
                    print('--- NOISE - LASNOISE - classify outlier points as
noise')
                    outFl_noise = outFl_class_one[:-4] + '_noise' + sep + lazsuf
                    execute = lasnoise + ' -i ' + outFl_class_one + ' -o ' +
outFl_noise
```

```
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('outlier points in SM5 ', fl[7:13], ' successfully
classified as noise')

                    # classify ground points
                    print('--- GROUND - LASHEIGHT - according to DTM product DMR5G
classify ground points as ground')
                    outFl_ground = outFl_noise[:-4] + '_ground' + sep + lazsuf
                    dtm_fl = dtm_path + 'BPR_SJTSKBPV_' + fl[7:13] + sep + lazsuf
                    execute = lasheight + ' -i ' + outFl_noise + ' -ground_points
' + dtm_fl + ground_settings + ' -o ' + outFl_ground
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('points considered as ground in SM5 ', fl[7:13], '
successfully classified as ground')

                    # classify bridges as ground
                    bridges_fl = opath_bridges_SM5 + block + '_' + fl[7:13] +
'_bridges' + sep + shpsuf
                    outFl_bridges = outFl_ground[:-4] + '_bridges' + sep + lazsuf
                    count = ap.management.GetCount(bridges_fl)
                    count = int(str(count))
                    if count != 0:
                        print('--- BRIDGES - LASCLIP - classify points on bridges
as ground')
                        execute = lasclip + ' -i ' + outFl_ground + ' -poly ' +
bridges_fl + bridges_settings + ' -o ' + outFl_bridges
                        print('execute: ', execute)
                        subprocess.call(execute, shell=True)
                        print('points on bridges in SM5 ', fl[7:13], '
successfully classified as ground')
                    else:
                        os.rename(outFl_ground, outFl_bridges)
                        print('--- NO BRIDGES in SM5 ', fl[7:13], ' skipping to
next step')

                    # classify buildings
                    buildings_fl = opath_buildings_SM5 + block + '_' + fl[7:13] +
'_buildings' + sep + shpsuf
                    outFl_buildings = outFl_bridges[:-4] + '_buildings' + sep +
lazsuf
                    count = ap.management.GetCount(buildings_fl)
                    count = int(str(count))
                    if count != 0:
                        print('--- BUILDINGS - LASCLIP - classify buildings')
                        execute = lasclip + ' -i ' + outFl_bridges + ' -poly ' +
buildings_fl + buildings_settings + ' -o ' + outFl_buildings
                        print('execute: ', execute)
                        subprocess.call(execute, shell=True)
```

```
                    print('points on buildings in SM5 ', fl[7:13], '
successfully classified as buildings')
                else:
                    os.rename(outFl_bridges, outFl_buildings)
                    print('--- NO BUILDINGS in SM5 ', fl[7:13], ' skipping to
next step')

                # classify vegetation according to ZABAGED forests
                forests_fl = opath_forests_SM5 + block + '_' + fl[7:13] +
'_forests' + sep + shpsuf
                outFl_forests = outFl_buildings[:-4] + '_forests' + sep +
lazsuf
                count = ap.management.GetCount(forests_fl)
                count = int(str(count))
                if count != 0:
                    print('--- ZABAGED VEGETATION - LASCLIP - classify
vegetation according to ZABAGED forests')
                    execute = lasclip + ' -i ' + outFl_buildings + ' -poly ' +
forests_fl + forests_settings + ' -o ' + outFl_forests
                    print('execute: ', execute)
                    subprocess.call(execute, shell=True)
                    print('points in ZABAGED forests in SM5 ', fl[7:13], '
successfully classified as vegetation')
                else:
                    os.rename(outFl_buildings, outFl_forests)
                    print('--- NO ZABAGED VEGETATION in SM5 ', fl[7:13], '
skipping to next step')

                # classify vegetation
                print('--- VEGETATION - LASCLASSIFY - classify vegetation')
                outFl_vegetation = outFl_forests[:-4] + '_vegetation' + sep +
lazsuf
                execute = lasclassify + ' -i ' + outFl_forests +
vegetation_settings + ' -o ' + outFl_vegetation
                print('execute: ', execute)
                subprocess.call(execute, shell=True)
                print('vegetation points in SM5 ', fl[7:13], ' successfully
classified as vegetation')

                # rename vegetation output file to FINAL output file
                outFl_FINAL = whole_opath[:-4] + '_CLASSIFICATION' + sep +
lazsuf
                os.rename(outFl_vegetation, outFl_FINAL)
                print('FINAL CLASSIFICATION laz file for SM5 ', fl[7:13], '
successfully created as ', outFl_FINAL, '!')

                # delete intermediate products
                if os.path.exists(outFl_class_one):
                    os.remove(outFl_class_one)
                if os.path.exists(outFl_noise):
                    os.remove(outFl_noise)
```

```
                    if os.path.exists(outFl_ground):
                        os.remove(outFl_ground)
                    if os.path.exists(outFl_bridges):
                        os.remove(outFl_bridges)
                    if os.path.exists(outFl_buildings):
                        os.remove(outFl_buildings)
                    if os.path.exists(outFl_forests):
                        os.remove(outFl_forests)

                    print('------------------------------------------------------------
-----')
            print('------------------------------------------------------------------')
        print('==============================================================')

    # delete intermediate folders
    shutil.rmtree(opath_bridges_SM5)
    print('intermediate folder: ', opath_bridges_SM5, ' deleted')
    shutil.rmtree(opath_buildings_SM5)
    print('intermediate folder: ', opath_buildings_SM5, ' deleted')
    shutil.rmtree(opath_forests_SM5)
    print('intermediate folder: ', opath_forests_SM5, ' deleted')

    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```

Příloha 18: Skript 14_detect_holes.py

```
   # ------------------------------------------------------------------------
------------------------------------------------------------------
   # Title: 14_detect_holes.py
   # Author: Bc. Jindřich HORÁK
   # Date: April 26, 2023
   # Created for Master Thesis: Creating a digital surface model from point
clouds created by image correlation of aerial imagery (in Czech language)
   # Palacký University Olomouc, Faculty of Science, Department of Geoinformatics
   # ------------------------------------------------------------------------
------------------------------------------------------------------


   import arcpy as ap
   from datetime import datetime
   import os
   import subprocess

   curpath = os.path.dirname(os.getcwd())
   ipath = curpath + '\\16_stereopairs_laz_UHUL_FINAL_DSM\\'
   opath = curpath + '\\18_stereopairs_laz_UHUL_holes_in_DSM\\'
   print('input folder: ', ipath)
   print('output folder: ', opath)
   print('------------------------------------------------------------')

   shpsuf = 'shp'
   lazsuf = 'laz'
   sep = '.'
   epsg5514 = 'PROJCS[\"S-
JTSK_Krovak_East_North\",GEOGCS[\"GCS_S_JTSK\",DATUM[\"D_S_JTSK\",SPHEROID[\"Bess
el_1841\",6377397.155,299.1528128]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0
174532925199433]],PROJECTION[\"Krovak\"],PARAMETER[\"False_Easting\",0.0],PARAMET
ER[\"False_Northing\",0.0],PARAMETER[\"Pseudo_Standard_Parallel_1\",78.5],PARAMET
ER[\"Scale_Factor\",0.9999],PARAMETER[\"Azimuth\",30.28813975277778],PARAMETER[\"
Longitude_Of_Center\",24.83333333333333],PARAMETER[\"Latitude_Of_Center\",49.5],P
ARAMETER[\"X_Scale\",-
1.0],PARAMETER[\"Y_Scale\",1.0],PARAMETER[\"XY_Plane_Rotation\",90.0],UNIT[\"Mete
r\",1.0]]'

   lastools_base_path = curpath + '\\programy\\'
   lastools_version = 'LAStools220926'

   lasboundary = lastools_base_path + lastools_version +
'\\bin\\lasboundary64.exe'

   lasboundary_settings1 = ' -holes -concavity 30 -merged'
   lasboundary_settings2 = ' -merged'

   start = datetime.now()

   years = os.listdir(ipath)
   print('years: ', years)
```

```
   for year in years:
       print('year: ', year)
       blocks = os.listdir(ipath + year)
       print('blocks: ', blocks)
       for block in blocks:
           print('block: ', block)
           whole_ipath = ipath + year + '\\' + block + '\\'
           whole_opath = opath + year + '\\' + block + '\\'
           try:
               os.makedirs(whole_opath)
               print('created folder: ', whole_opath)
           except:
               pass

           # create SHP file with boundary and holes
           outFl_boundary_holes = whole_opath + block +
'_block_lasboundary_holes' + sep + shpsuf
           execute = lasboundary + ' -i ' + whole_ipath + '*' + sep + lazsuf +
lasboundary_settings1 + ' -o ' + outFl_boundary_holes
           print('execute: ', execute)
           subprocess.call(execute, shell=True)
           print('SHP file with boundary and holes for block ', block, '
successfully created')

           # create SHP file only with boundary
           outFl_boundary = whole_opath + block + '_block_lasboundary' + sep +
shpsuf
           execute = lasboundary + ' -i ' + whole_ipath + '*' + sep + lazsuf +
lasboundary_settings2 + ' -o ' + outFl_boundary
           print('execute: ', execute)
           subprocess.call(execute, shell=True)
           print('SHP file only with boundary for block ', block, ' successfully
created')

           # select and delete the boundary - preserve the holes
           selected = ap.management.SelectLayerByLocation(outFl_boundary_holes,
'WITHIN', outFl_boundary, '', 'NEW_SELECTION', 'INVERT')
           ap.management.DeleteFeatures(selected)
           print('boundary deleted, holes preserved in block ', block)

           # delete boundary SHP file
           ap.management.Delete(outFl_boundary)
           print('only boundary SHP file for block', block, ' deleted')

           # rename final holes SHP file
           outFl_holes = outFl_boundary_holes.replace('_lasboundary', '')
           ap.management.Rename(outFl_boundary_holes, outFl_holes)
           print('final holes SHP file for block ', block, ' created as ',
outFl_holes, '!')
```

```
            # define projection EPSG 5514 to final holes SHP file
            ap.management.DefineProjection(outFl_holes, epsg5514)
            print('projection EPSG 5514 defined to final holes SHP file: ',
outFl_holes)

            print('------------------------------------------------------------')
        print('============================================================')
    print('======================END OF SCRIPT!======================')

    end = datetime.now()
    print('time elapsed: ', end - start)
```