UNIVERZITA KARLOVA

Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

Studijní program: Geografie (magisterské studium)

Studijní obor: Kartografie a geoinformatika

Bc. Rostislav BERKA

VYUŽITÍ AUTOMATIZOVANÉHO SBĚRU VLÍCOVACÍCH BODŮ PRO ORIENTACI ARCHIVNÍCH LETECKÝCH MĚŘICKÝCH SNÍMKŮ S POMOCÍ SOUČASNÝCH DATOVÝCH SAD

USE OF AUTOMATIC COLLECTION OF GROUND CONTROL POINTS FOR ORIENTATION OF ARCHIVAL AERIAL IMAGERY USING EXISTING DATA SETS

Diplomová práce

Praha 2024

Vysoká škola: Univerzita Karlova **Fakulta:** Přírodovědecká

Katedra: Aplikované geoinformatiky a kartografie **Školní rok:** 2021/2022

Zadání diplomové práce

pro Bc. Rostislava Berku

obor Kartografie a geoinformatika

Název: Využití automatizovaného sběru vlícovacích bodů pro orientaci archivních leteckých měřických snímků s pomocí současných datových sad

Zásady pro vypracování

Diplomová práce se zaměří na využití metod vyhledávání identických bodů (vzorů) v obraze pro nepřímé určení prvků vnější orientace (EO) archivních leteckých měřických snímků (LMS) s využitím existujících ortofot, digitálních modelů reliéfu a vektorových datových sad. Zatímco automatické, případně semi-automatické vyhledávání a určování pixelových souřadnic rámových značek a spojovacích bodů je ve většině fotogrammetrických softwarů implementováno, v případě vlícovacích bodů (VLB) je nutné provést jejich sběr ručně. Typicky se jedná o časově náročnou manuální činnost. Za referenční práce směrem k automatizaci tohoto procesu je možno považovat například studii Oh, Toth, Grejner-Brzezinska (2010) a Höhle, Potůčková (2001). Oh, Toth, Grejner-Brzezinska (2010) využívají satelitní snímky s vysokým rozlišením z Ikonosu jakožto referenční podklad pro vyhledávání vlícovacích bodů s pomocí operátoru SIFT pro georeferencování LMS. Špatně spárované body následně filtrují s pomocí RANSACu, přičemž jako geometrický model využívají rovnice kolinearity. Höhle, Potůčková (2001) vyhledávají VLB s pomocí současného ortofota a křižovatek silnic (vektorová data), přičemž v postupu jim jakožto operátor slouží obrazová korelace.

Konkrétní cíle diplomové práce jsou následující:

1) Vytvořit (naprogramovat) plně automatizovanou metodiku pro sběr VLB s pomocí již existujících dat s přesnou georeferencí a metod pro detekci a popis bodů v obraze.
   Pro vyhledávání totožných (jednoznačně identifikovatelných a časově invariantních) bodů mezi dvěma obrazy odlišného data pořízení budou v diplomové práci vyzkoušeny různé operátory, např. operátor SIFT, který je invariantní vůči rotacím, změně měřítka či kontrastu v obraze, SURF, ORB a obrazová korelace založená na Pearsonově korelačním koeficientu, případně jiné. Na základě testování bude vybrán nejvhodnější operátor a s jeho pomocí bude naprogramována automatizovaná metodika, jejímž vstupem budou archivní LMS a referenční data s přesnou georeferencí. Metodika bude zahrnovat i odfiltrování špatně spárovaných bodů s pomocí robustních metod jako je např. RANSAC. Výstupem pak bude soubor či soubory s VLB ve formátu importovatelném do standardních fotogrammetrických a GIS SW (MicMac, QGIS).
2) Vytvořit bezešvé ortomozaiky pro daná zájmová území
   Archivní LMS a automatizovaně nalezené VLB budou využity pro vytvoření bezešvých ortomozaiek s pomocí open-source fotogrammetrického softwarového vybavení. Postup bude zahrnovat veškeré

obvyklé kroky tvorby ortofota jako je výpočet prvků vnitřní a vnější orientace snímků, svazkové vyrovnání, generování DMP a výsledných ortomozaiek.

3) Zhodnotit přesnost výstupů (RMSE) v závislosti na úrovni změny krajiny (suburbánní zástavba, scelování polí), na fyzickogeografických charakteristikách (rovina vs. zvlněná krajina) a kvalitě snímků z různých časových období.

Archivní snímky z 30. a 80. let dosahují různé radiometrické kvality. Změny v otevřené krajině i zástavbě jsou ve vybraných modelových územích (Milovice a Kobylí) zásadní.

Pro automatizaci sběru VLB budou vytvořeny skripty v jazyce Python s využitím open-source knihoven pro zpracování obrazu a prostorových dat. Snímkové orientace budou provedeny v programu MicMac (taktéž open-source). Plně automatizován tedy bude samotný sběr VLB, ostatní standardizované kroky zpracování ortofota ve fotogrammetrickém SW budou vyžadovat propojení ze strany uživatele.

S ohledem na předchozí projekty katedry byla vybrána zájmová území Milovice a Kobylí a následující roky snímkování – Milovice 1938 a 1989, Kobylí 1938 a 1984. Téma bylo podpořeno vedením Zeměměřického úřadu a datové sady (současné ortofoto, archivní LMS, ZABAGED) byly poskytnuty pro účely zpracování práce zdarma nad rámec běžné studentské licence.

Seznam odborné literatury:

FÖRSTNER, W., WROBEL, B. P. (2016): Photogrammetric Computer Vision. Springer, New York City.

HARTMANN, W., HAVLENA, M., SCHINDLER, K. (2016): Recent developments in large-scale tie-point matching. ISPRS Journal of Photogrammetry and Remote Sensing, 115, 47–62.

HODAČ, J., ZEMÁNKOVÁ, A. (2018): Historical orthophotos created on base of single photos - specifics of processing. The Civil Engineering Journal, 3, 14.

HÖHLE, J., POTŮČKOVÁ, M. (2001): Towards the Full Automatic Production of Orthoimages. Photogrammetrie, 6/2001, 397–404.

CHEN, H. R., TSENG, Y. H. (2016): Study of automatic image rectification and registration of scanned historical aerial photographs. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, July, 41, 1229–1236.

SZELISKI, R. (2022): Computer Vision: Algorithms and Applications, 2nd Edition. Springer, New York.

ŠAFÁŘ, V., TLAPÁKOVÁ, L. (2016): Alternative Methods of the Processing of Archival Aerial Photos. Geodetický a Kartografický obzor, 104, 62, 253–257.

OH, J., TOTH, C. K., GREJNER-BRZEZINSKA, D. A. (2010): Automatic Georeferencing of Aerial Images Using Stereo High-Resolution Satellite Images. ASPRS 2010 ANNUAL CONFERENCE.

PAVELKA, K. (2003): Fotogrammetrie. Západočeská univerzita v Plzni, Plzeň.

Datum zadání diplomové práce: 5. 12. 2021

Termín odevzdání diplomové práce: duben 2024

…… ……

Vedoucí diplomové práce Vedoucí katedry

V Praze dne

Prohlášení

Prohlašuji, že jsem závěrečnou práci zpracoval samostatně a že jsem uvedl všechny použité informační zdroje a literaturu. Tato práce ani její podstatná část nebyla předložena k získání jiného nebo stejného akademického titulu.

I am not afraid. :-)

Reporter: When you had that third failure in a row, did you think: I need to pack this in?
Elon Musk: Never.
Reporter: Why not?
Elon Musk: I don't ever give up. I mean I'd have to be dead or completely incapacitated. For my part, I will never give up, and I mean never.

VYUŽITÍ AUTOMATIZOVANÉHO SBĚRU VLÍCOVACÍCH BODŮ PRO ORIENTACI ARCHIVNÍCH LETECKÝCH MĚŘICKÝCH SNÍMKŮ S POMOCÍ SOUČASNÝCH DATOVÝCH SAD

Abstrakt

Diplomová práce se zaměřuje na automatizované vyhledávání vlícovacích bodů (VLB) za účelem zpracování archivních leteckých měřických snímků (LMS) a vytvoření ortomozaiky. V práci jsou testovány vybrané operátory z oblasti počítačového vidění (SIFT, SURF, ORB a obrazová korelace) a různé přístupy pro vyhledávání a následně párování identických bodů mezi referenčním podkladem (současným ortofotem) a archivními LMS. Díky georeferencím LMS je známa oblast k prohledávání ve starých LMS pro nalezení odpovídajícího bodu k referenčnímu podkladu (současnému ortofotu). Bohužel se ukázalo, že georeference archivních LMS dodané ČÚZK jsou chybně metodicky zpracované (velmi nepřesné), proto musely být před automatizovaným vyhledáváním VLB opraveny. Téma bylo podpořeno vedením Zeměměřického úřadu a datové sady (současné ortofoto, archivní LMS, ZABAGED) byly poskytnuty pro účely zpracování práce zdarma (nad rámec běžné studentské licence).

Na základě testování operátorů byla sestavena plně automatizovaná metodika naprogramovaná v jazyce Python založená na obrazové korelaci. Vstupem jsou archivní LMS, současné ortofoto a síť komunikací, přičemž jako spolehlivé a časově invariantní body slouží křižovatky silnic. Součástí postupu je i odfiltrování špatně spárovaných bodů s pomocí robustního algoritmu RANSAC v kombinaci s transformací DLT. Výstupem jsou soubory s VLB importovatelné do standardních fotogrammetrických a GIS nástrojů.

Výsledkem jsou bezešvé ortomozaiky pro území Milovice a Kobylí vždy ve dvou časových řezech (30. a 80. léta), pro jejichž zpracování byly použity automatizovaně nalezené VLB a SW MicMac. V MicMacu byly provedeny standardní kroky pro zpracování ortofota jako je výpočet prvků vnitřní a vnější orientace snímků, svazkové vyrovnání, generování DMP a výsledných ortomozaiek. Nakonec je zhodnocena přesnost výstupů s pomocí kontrolních bodů a RMSE.

Klíčová slova: automatizovaný sběr vlícovacích bodů, časové řady, snímkové orientace, archivní letecké měřické snímky, ortofoto, DMP, RANSAC

USE OF AUTOMATIC COLLECTION OF GROUND CONTROL POINTS FOR ORIENTATION OF ARCHIVAL AERIAL IMAGERY USING EXISTING DATA SETS

Abstract

The diploma thesis is focused on the automated searching of ground control points (GCPs) for processing archival aerial images (AAIs) and creating orthomosaics. The study evaluates selected computer vision operators (SIFT, SURF, ORB and image correlation) and various approaches for identifying and subsequently matching identical points between the reference dataset (current orthophoto) and AAIs. Thanks to AAIs georeferences, the searching area in AAI is known for finding the corresponding point to the reference one (from current orthophoto). Unfortunately, the georeferencing of AAIs provided by the Czech Office for Surveying, Mapping, and Cadastre (ČÚZK) was methodologically incorrect (very bad accuracy) and it had to be corrected before automated GCPs searching. The topic was supported by ČÚZK and datasets (current orthophoto, AAIs, ZABAGED vector data) were provided for processing purposes free of charge (beyond the standard student license).

Based on operator testing, a fully automated methodology was developed in Python, relying on image correlation. The input includes AAIs, current orthophoto and a road network, with road intersections as reliable and time invariant points. The procedure also includes filtering poorly matched points using the robust RANSAC algorithm combined with DLT transformation. The output consists of files containing GCPs that can be imported into standard photogrammetric and GIS tools.

The result is seamless orthomosaics for the Milovice and Kobylí areas, each captured at two different time slices (1930s and 1980s). The orthophoto processing used automatically detected GCPs and the MicMac software. In MicMac, standard steps for orthophoto processing were performed, including computation of internal and external orientation, bundle adjustment, digital surface model (DSM) generation, and final orthomosaic production. The accuracy of the outputs was evaluated using control points and the root mean square error (RMSE).

Keywords: automatic collection of ground control points, time series, image orientation, archival aerial imagery, orthophoto, DTM, RANSAC

Obsah

Použité zkratky

3D	**T**roj**d**imenzionální
ČSSR	**Č**eskoslovenská **s**ocialistická **r**epublika
ČÚZK	**Č**eský **ú**řad **z**eměměřický a **k**atastrální
DLT	**D**irect **l**inear **t**ransformation
DMP	**D**igitální **m**odel **p**ovrchu
EO	**E**xterior **O**rientation (vnější orientace)
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
IO	**I**nterior **O**rientation (vnitřní orientace)
ISPRS	**I**nternational **S**ociety for **P**hotogrammetry and **R**emote **S**ensing
LMS	**L**etecký **m**ěřický **s**nímek
ORB	**O**riented FAST and **R**otated **B**RIEF
RANSAC	**RAN**dom**SA**mple **C**onsensus
RMSE	**R**oot **m**ean **s**quare **e**rror (střední kvadratická odchylka)
SFM	**S**tructure **f**rom **m**otion
SIFT	**S**cale-**i**nvariant **f**eature **t**ransform
SMO-5	**S**tátní **m**apa **o**dvozená 1 : **5**000
SW	**S**oft**w**are
VGHMÚř	**V**ojenský **g**eografický a **h**ydro**m**eteorologický **ú**řad generála Josefa Churavého
VLB	**Vl**ícovací **b**od

Použité obrázky

Použité tabulky

Použité grafy

1 Několik slov úvodem

1.1 Obecně

Předkládaná diplomová práce se zaměřuje na orientaci archivních leteckých měřických snímků (LMS) s využitím existujících datových sad. Zpracování leteckých snímků se provádí v několika krocích. Při jejich provádění je typicky nutné přesně identifikovat nejrůznější body. Jedná se například o rámové značky, spojovací body a body vlícovací. Zatímco automatické, případně semi-automatické vyhledávání a určování pixelových souřadnic rámových značek a spojovacích bodů je v některých SW implementováno (např. (Pierrot Deseilligny, Clery 2012)), SW MicMac, Metashape), případně existuje množství prací na toto téma (např. (Hartmann, Havlena, Schindler 2016)), v případě vlícovacích bodů (VLB) je většinou nutné provést jejich sběr ručně. Jedná se o manuální a poměrně časově náročnou činnost. Tyto body jsou nutné pro výpočet prvků vnější orientace (EO) a následnou ortorektifikaci snímků.

Automatizovaný sběr VLB je někdy uplatňován u nově pořízených dat, která je třeba zpracovat. U dat archivních je provedení takového úkonu mnohem větší výzvou z důvodu změn ve využití krajiny (Obrázek 1). Z hlediska automatizace se poměrně běžně využívá automatické vyhledávání tzv. spojovacích bodů (mezi archivními snímky navzájem), nikoliv však vyhledávání VLB mezi archivním snímkem a současným ortofotem. Oh, Toth, Grejner-Brzezinska (2010) využívají satelitní snímky s vysokým rozlišením z Ikonosu jakožto referenční podklad pro vyhledávání vlícovacích bodů s pomocí operátoru SIFT pro georeferencování LMS. Spatně spárované body následně filtrují s pomocí RANSACu, přičemž jako geometrický model využívají rovnice kolinearity. Höhle, Potůčková (2001) vyhledávají VLB s pomocí současného ortofota a křižovatek silnic (vektorová data), přičemž v postupu jim jakožto operátor slouží obrazová korelace.

S archivními LMS pracují např. Hodač, Zemánková (2018), Šafář, Tlapáková (2016) a Řáda (2016). Ve všech případech se však jedná pouze o ryze manuální zpracování archivního ortofota. Také využití MicMacu je v českém prostředí poměrně ojedinělé. Nebyla nalezena žádná jiná práce, kde by byl tento SW použit pro generování ortofota. Program je jinak v akademickém prostředí oblíben a existuje množství zahraničních studií, kde je využit.

Obrázek 1: Současné vs. archivní ortofoto

Zdroj: vlastní zpracování, podkladová data ČÚZK (2022)/Cenia (2022)

Obecně při problematice párování totožných bodů mezi 2 obrazy je třeba nejprve najít body vhodné k tomuto účelu. Musí se jednat o body jednoznačně identifikovatelné v cílovém obraze i ve vzoru. U historických dat musí být body navíc časově invariantní. K nalezení a spárování takových bodů se používají operátory z oblasti počítačového vidění (computer vision). Lze zmínit ty nejjednodušší (Moravcův operátor) až po ty složitější a v dnešní praxi užívané – Harrisův operátor, FAST, SIFT, SURF, ORB a další.

Zortorektifikované, případně zgeoreferencované LMS nachází mnoho využití v nejrůznějších GIS analýzách. Elznicová (2008) využívá zortorektifikované LMS k nalezení tzv. agrárních valů a porovnává historickou situaci se současností. Šmejda (2009) identifikuje nejrůznější archeologicky významná místa mimo jiné také s pomocí archivních LMS. Jedná se například o pohřebiště, hradiště, areály dřívějšího osídlení, příkopy, valy apod. Taktéž v Digitálním atlase zaniklých krajin spadajícím pod projekt NAKI je pro každé modelové území vytvářen model krajiny s využitím zortorektifikovaných LMS. Kardoš a kol. (2017) využívají historickou ortofoto mapu Slovenska vytvořenou z archivních LMS pro identifikaci původních pozemků před scelováním polí.

1.2 Cíle práce a hypotézy

Ve velkém množství prací z různých oblastí včetně fotogrammetrie je použit SIFT operátor pro vyhledávání a párování bodů v obraze (např. (Oh, Toth, Grejner-Brzezinska 2010)). Jedná se o

hojně užívaný operátor, který by měl být odolný vůči otočení, změnám v kontrastu a měřítku v obraze aj. Vzniká tak otázka, zda je jinak hojně užívaný SIFT taktéž vhodný pro párování bodů mezi obrazy z různých časových období. Höhle, Potůčková (2001) sestavili pracovní postup využívající vektorová data (křižovatky silnic) a obrazovou korelaci. Metodiku testovali na současné datové sadě určené ke zpracování. Postup je výrazně odlišný od předchozího zmíněného založeného na SIFTu. Snahou bude prozkoumat, zda je obrazová korelace, SIFT a další operátory vhodná i pro archivní LMS.

Zároveň by bylo vhodné zjistit, jak si kompletně celá navržená metodika dokáže poradit ve scénách s různou úrovní změny krajiny (suburbánní zástavba, scelování polí), s různými fyzickogeografickými charakteristikami (rovina vs zvlněná krajina) a se snímky z různých časových období. Za tímto účelem je zvoleno několik scén, přičemž na závěr je provedeno kvantitativní srovnání výsledků (RMSE). Je nutné zdůraznit, že navržená metodika nemá ambice fungovat v územích, kde došlo ke kompletnímu odstranění krajinného krytu, jako jsou například oblasti povrchové těžby surovin nebo oblasti, kde je obecně problém s hledáním význačných bodů (zalesněné pohraniční horské oblasti, vojenské újezdy) a další území s extrémními rysy.

Konkrétní cíle jsou následující:

1) Automatizovat sběr VLB v archivních LMS s pomocí současných dat a metod pro detekci a popis bodů v obraze.
2) S pomocí sestavené metodiky automatizovaně vyhledat VLB a vytvořit ortomozaiky pro všechny datové sady.
3) Zhodnotit přesnost výstupů s pomocí kontrolních bodů (RMSE) a hlavně porovnat jednotlivá zájmová území a časová období mezi sebou z hlediska přesnosti.

Snahou zároveň bude v co největší míře používat zdarma dostupná open-source řešení.

2 Rešeršní část

2.1 LMS v historickém a současném kontextu

Letecké měřické snímky představují cenný zdroj informací. První stereofotogrammetrické měření proběhlo v ČSR již v období 1. republiky. Ještě před 2. sv. válkou bylo zmapováno 67 000 km^2 území. Počátkem 50. let se začalo zhotovovat souvislé topografické mapování s pomocí fotogrammetrie, které prováděl Vojenský topografický ústav a pracoviště Ústřední správy geodézie a kartografie. Již v polovině 50. let bylo nasnímáno cca 70 % území tehdejší ČSSR a fotogrammetrická data sloužila jako jeden z hlavních podkladů při tvorbě polohopisu i výškopisu topografických map. Díky tomu se výrazně zkrátila finanční i časová náročnost mapování. Veškerý pořízený fotogrammetrický materiál podléhal utajení až do pádu železné opony (Pavelka 2003a).

V archivu VGHMÚř se nachází velké množství v minulosti pořízených LMS. Snímky jsou pracovníky VGHMÚř postupně zpřístupňovány ve spolupráci s ČÚZK, poskytovány jsou však pouze v „syrovém" stavu a netvoří souvislé ortofoto. Nebyly ortorektifikovány, mají sice určené prvky vnější orientace (EO) a georeferenci prostřednictvím World file souboru, nicméně tento způsob zpracování způsobuje nepřesnosti obzvlášť citelné ve zvlněném terénu. LMS na této úrovni zpracování lze použít pouze k pracím s nízkými nároky na výslednou přesnost a typicky v rovinných územích (Pavelka 2003a). Georeferenci v podobě *.tfw World file souboru vytváří ke snímkům pracoviště ČÚZK. V průběhu zpracování práce však bylo zjištěno, že tyto georeference jsou metodicky chybně zpracovány, proto musely být opraveny. Problém je popsán v metodické části práce v kapitole „Předzpracování dat pro automatizované vyhledávání VLB".

Pro korektní vyhodnocení obsahu LMS je bezpodmínečně nutný přesný výpočet prvků EO a následná ortorektifikace. Za tímto účelem se provádí sběr vlícovacích bodů (VLB). Jedná se o manuální a časově velmi náročnou činnost. Vzhledem k objemu LMS, kterými disponuje VGHMÚř není možné tímto způsobem zpracovat v rozumném čase celý archiv. Tyto kroky proto musí provést koncový uživatel GIS. Pro usnadnění vytěžení dat z archivních LMS by tak bylo velmi vhodné automatizovat sběr VLB. Situaci nadále komplikuje absence kalibračních protokolů.

2.2 Manuální zpracování archivních LMS

Nejjednodušší metodou, s pomocí které lze umožnit práci s LMS, je georeferencování. Korektní je využití projektivní transformace, avšak způsob je použitelný pouze pro rovinné území. Je přitom nutné akceptovat nižší přesnost vyhodnocení. Vzniká tak fotoplán (Pavelka 2003a). Koeficienty projektivní transformace navíc nelze uložit do World file souboru, který podporuje pouze transformaci podobnostní nebo afinní. Někteří autoři využívají pro LMS méně vhodné transformace polynomické 1. řádu (Dědková 2012), či dokonce vyšších řádů (Sádovská 2011).

Pinto a kol. (2019) využívají 3D fotogrammetrickou transformaci DLT pro zpracování archivních LMS z oblasti v Portugalsku. Mimo jiné konstatují, že automatizovaná metodika pro zpracování archivních LMS v podstatě neexistuje z důvodu významných změn v krajině. Na rozdíl od předchozích zmíněných transformací pracuje DLT i s nadmořskou výškou a dokáže si tak poradit s nerovnostmi terénu. Autoři vytvořili program „Projective 3D“ ovládaný z příkazové řádky pro ortorektifikaci LMS s pomocí DLT. S pomocí zmíněného SW vygenerovali ortorektifikované snímky pro svou zájmovou oblast. Na rozdíl od rovnic kolinearity není třeba při užití DLT znát konstantu kamery a zároveň je možno vyhodnocovat jednotlivé snímky bez stereodvojic. Toto může být velmi užitečné, jelikož u archivních LMS často může chybět snímek do stereodvojice (Pinto a kol. 2019).

Šafář, Tlapáková (2016) diskutují nejrůznější úskalí a specifika, se kterými je třeba se vyrovnat u zpracování ortofota z archivních LMS. Diskutují problémy související s absencí kalibračních protokolů apod. Uvádí například, že pro ortogonalizaci je možné použít DMR5G se stereoskopickou kontrolou aktuálnosti vůči době pořízení snímků. Jejich postup je však ryze manuální. Zmiňují však netradiční metodiku, kdy je pracováno s archivními LMS jako s ryze digitálním obrazem a ke zpracování jsou použity techniky na bázi SFM (structure from motion). S pomocí SFM zpracovává archivní LMS z roku 1947 z oblasti Porta v Portugalsku Gonçalves (2016) v SW Agisoft Photoscan.

Elznicová (2008) pracovala s archivními LMS s pouze 20 % překrytem (jeden snímek v původní stereodvojici vždy vynechán). Pracuje se snímky tradičním způsobem, VLB jsou sbírány manuálně. V důsledku absence kalibračních protokolů bylo se snímky pracováno jako s neměřickými, a to v SW Leica Photogrammetry Suite. Snímky byly ortorektifikovány s pomocí DMR vytvořeného z vrstevnic databáze ZABAGED.

Hodač, Zemánková (2018) konstatují, že se pro ortorektifikaci archivních LMS často využívá současné DMT. Takové DMT je vhodné alespoň upravit s pomocí stereodvojice snímků (stereoplotting) v případě, že se část území v zájmové oblasti výrazněji změnila, co se výškopisu týče. Autoři konstatují, že archiv VGHMÚř obsahuje kolem 900 000 archivních LMS. V metodické části je pracováno s jednotlivými archivními LMS, které jsou zpracovány v SW PhoTopoL. Z prvků IO byla známa pouze konstanta kamery, pixelové souřadnice rámových značek byly změřeny a snímkové souřadnice byly poté dopočítány s pomocí velikosti pixelu. Ostatní prvky IO mohou být dopočítány s pomocí svazkového vyrovnání. Toto je však možné pouze při větším množstvím snímků. Proběhl též pokus o vygenerování DMP z historických snímků s pomocí algoritmu obrazové korelace. Nicméně z důvodu nevyhovující kvality nebyl tento DMP použit pro ortorektifikaci.

Řáda (2016) ve své diplomové práci sbírá VLB ručně ze současného ortofota a popisuje standardní postup zpracování ortofota z archivních LMS. Zájmovým územím je obec Houstoň v okrese Kladno (Obrázek 2), přičemž byly použity pouze 2 archivní LMS. Práce obsahuje popis zpracování ortofota v SW PhoTopol včetně printscreenů a vesměs postrádá inovativní prvky. Může sloužit jako výukový materiál zabývající se obsluhou zmíněného SW.

Obrázek 2: Porovnání historického a současného ortofota

Zdroj: (Řáda 2016)

2.3 Automatizované zpracování LMS

Postupně se dostáváme k automatizovanému zpracování a hledání význačných bodů v LMS. Cléri, Pierrot-Deseilligny, (2014) využívají liniové prvky v krajině (komunikace, obrysy budov) a RANSAC pro zpracování archivních LMS. Jedná se o postup výrazně odlišný od metodiky této práce. Linie (konkrétně ulice) využívá také Nagarajan, Schenk (2016). Existují samozřejmě i studie, které se zabývají automatizovaným georeferencováním map (např. Burt et al. (2020), Li, Briggs (2006)). Burt et al. (2020) sestavuje (semi-)automatizovaný postup pro georeferencování starých map. V případě map je úkol poněkud usnadněn, jelikož lze jako VLB využít např. průsečíky zeměpisné sítě, značky v mapovém rámu apod.

Chen, Tseng (2016) láká na automatické zpracování historických leteckých snímků. Jedná se však „pouze" o automatické vyhledávání a spojování spojovacích bodů mezi stereodvojicemi historických snímků. Samotné vlícovací body jsou identifikovány manuálně. Vstupními daty jsou skeny archivních leteckých snímků zabírající území Taiwanu, přičemž prvky vnitřní orientace nejsou vůbec známy. Jakožto referenční podkladová data jsou použity satelitní snímky FORMOSAT-2. Chen, Tseng (2016) zmiňují, že i manuální identifikace bodů je problematická z důvodu významných změn v krajině.

K detekci význačných bodů je použit SIFT operátor. V následujících krocích používají 2D afinní a 2D projektivní transformaci pro popsání vztahu mezi dvěma snímky ve stereodvojici. Cílem je dostat transformaci, která bude sloužit k převedení snímkových souřadnic jednoho snímku do snímkových souřadnic sousedního snímku. Zmíněné transformace používají jako geometrický model v RANSACu pro odfiltrování odlehlých měření. Snímky jsou pouze georeferencovány, nikoliv ortorektifikovány. K vyhledávání spojovacích bodů mezi snímky z dronu používá SIFT také Gotovac a kol. (2017). Body následně filtruje pomocí RANSACu a afinní transformace.

Oh, Toth, Grejner-Brzezinska (2010) využívají satelitní snímky s vysokým rozlišením z Ikonosu jakožto referenční podklad pro vyhledávání VLB s pomocí operátoru SIFT pro georeferencování LMS. Špatně spárované body následně filtrují s pomocí RANSACu, přičemž jako geometrický model využívají rovnice kolinearity. Nicméně LMS byly pořízeny pouze 2 roky po snímcích z Ikonosu. Feurer, Vinatier (2018) používají SIFT a Structure from Motion (SFM) pro zpracování LMS z různých období (LMS pořízené od roku 1971 do 2001) v jediném bloku. Označují SIFT jako časově invariantní operátor. Li a kol. (2009) využívají upravený SIFT (Robust SIFT) pro registraci družicových snímků. Operátor testují na datech z Landsatu, SPOTu

a dalších senzorů. Na závěr konstatují, že Robust SIFT dosahuje lepších výsledků pro dané účely než klasický SIFT a i další operátory. Chureesampant, Susaki (2014) používají SIFT operátor pro vyhledání vlícovacích bodů u radarových snímků. Použití SIFTu pro takto méně typickou úlohu obhajují jeho obecnými vlastnostmi jako je odolnost vůči stočení, změně měřítka, změně úhlu snímání, částečná odolnost vůči změnám osvětlení, šumu v obraze apod. S pomocí minimalizace RMSE u pseudoafinní transformace odstraňují špatně spárované body. Mírně kriticky přistupují k RANSACu. Se SIFT operátorem a radarovými snímky pracují také Liu, Yu (2008), Canny operátor je použit pro zvýraznění hran. Berveglieri, Tommaselli (2014) a Berveglieri, Tommaselli (2017) využívají SIFT pro párování odpovídajících si bodů mezi LMS a pozemními snímky vytvořenými širokoúhlým objektivem s jednoznačně určitelnými prvky v krajině zaměřenými GPS, které slouží jako VLB. Hartmann, Havlena, Schindler (2016) podrobně popisuje metody párování odpovídajících si bodů nalezených s pomocí SIFTu a i dalších operátorů pro využití například při SFM. Hayder Dibs a kol. (2020) popisují metodiku pro vylepšení kvality párování SIFT bodů pro jejich užití v dálkovém průzkumu Země.

Z výše uvedeného vyplývá, že SIFT je hojně užívaný operátor sloužící pro vyhledávání VLB ve fotogrammetrii a dálkovém průzkumu Země. Problémem většiny uvedených studií využívajících SIFT je užití pouze současných datových sad, respektive mezi referenčními a cílovými obrazovými daty je pouze několik let rozdíl v datu pořízení.

Craciun, Le Bris (2022) sestavují metodiku pro automatizované georeferencování archivních LMS mimoměstských scén. Ve svém postupu využívají DMP pro automatizovanou extrakci VLB. Nepoužívají SIFT ani jiný podobný operátor, jelikož se nehodí pro čistě rurální krajinu vyznačující se velkou homogenitou s malým množstvím jednoznačně určitelných a identifikovatelných krajinných prvků (Craciun, Le Bris 2022). DMP pro automatizovanou extrakci VLB využívají také Zhang, Rupnik, Pierrot-Deseilligny (2021). Nurminen a kol. (2015) se věnují georeferencování archivních LMS zalesněných scén z oblasti Finské Karélie.

Höhle, Potůčková (2001) nabízí jiný přístup k řešení problému. Autoři zapojují do workflow i vektorová data pro identifikaci význačných bodů. Nepracují s tak starými snímky, nicméně představují způsob využívající pattern matching algoritmus (obrazovou korelaci) z oblasti počítačového vidění.

Vstupem je vektorová vrstva silnic z veřejné databáze obsahující zároveň nadmořskou výšku objektů. Vstupními obrazovými daty jsou současné ortofoto a novější snímek určený ke zpracování. Prvky EO byly odhadnuty na základě GPS dat pořízených při snímání. Nejedná se

však o historická data, mezi referenčními a novějšími snímky je rozdíl v době pořízení pouze 25 měsíců.

V databázi jsou vyhledány křižovatky významnějších komunikací. V případě komunikací se jedná o velmi vhodné objekty z hlediska umístění VLB oproti například budovám. U objektů vystupujících nad terén je totiž třeba počítat s nepřestnostmi způsobenými radiálními posuny bodů.

Pro každou křižovatku je z rastrového současného ortofota extrahována odpovídající malá plocha zobrazující dané křížení (31 × 31 pixelů). Musí mít lichý počet řádků a sloupců, aby bylo možno určit její prostřední pixel. Následně je pixel po pixelu procházena oblast (61 × 61 pixelů) ve snímku určeném k ortorektifikaci a pro každý pixel je počítán korelační koeficient. Pixel, pro který korelační koeficient vyjde nejvyšší, je považován za cílový. Takto jsou nalezeny potenciální vlícovací body.

Při zpracování samozřejmě docházelo i k chybnému párování VLB mezi referenčním a cílovým obrazem. Taková párování byla značně eliminována použitím pevného prahu u korelačního koeficientu 0,7. Body s nižším korelačním koeficientem byly odmítnuty. Tento práh bylo třeba pečlivě zvážit tak, aby byla plocha snímku dostatečně pokryta body (aby nebyly odstraněny i žádoucí body) a zároveň aby byla eliminována většina chybných spojení. Pro monitorování vhodného rozmístění spojovacích bodů bylo sledováno těžiště vlícovacích bodů. Následuje robustní vyrovnání pro odfiltrování zbytku chybných spojení a výpočet prvků EO.

Výsledkem jsou vypočtené prvky EO a ortorektifikovaný snímek, jehož přesnost je stejná jako přesnost podkladového ortofota. Höhle, Potůčková (2001) zmiňují, že jedním z možných problémů k dalšímu výzkumu je nevhodné (nerovnoměrné) rozmístění bodů po snímku. Jinak se podařilo dosáhnout kompletně automatizovaného postupu.

2.4 Vybraná fotogrammetrická open-source řešení

MicMac je open-source program ovládaný z příkazové řádky pro fotogrammetrické účely. Za projektem stojí IGN (French National Geographic Institute) a ENSG (French national school for geographic science). K MicMacu existuje i externí grafické rozhraní s názvem AperoDeDenis naprogramované v Pythonu (MicMac 2017). Další obecné informace k SW uvádí Rupnik, Daakir, Pierrot Deseilligny (2017). MicMac zvládá mnoho různých úloh jako například zpracování bezešvého ortofota z LMS (skenované analogové i digitální snímky) nebo například generování 3D modelů pomocí SFM (structure from motion). Je určen k pozemní i letecké fotogrammetrii (MicMac 2017). Jelikož je SW open-source, je aktivně vyvíjen komunitou. Pierrot Deseilligny, Clery (2012) například vyvinul modul APERO, který slouží pro kalibraci a orientaci sady LMS. Přímo na stránkách SW je k dispozici několik tutoriálů, mimo jiné i pro zpracování ortofota z archivních LMS. Dále se zde nachází dokumentace k jednotlivým modulům (bohužel nekompletní). Stránky i názvy jednotlivých modulů, parametrů funkcí apod. jsou bohužel někdy ve francouzštině. Velmi komplexním zdrojem informací je elektronická kniha napsaná IGN (2022), která obsahuje poměrně detailní popis modulů a různých způsobů použití MicMacu.

Pro MicMac existuje rozšíření spymicmac naprogramované v Pythonu, které obsahuje i automatizované párování VLB s pomocí obrazové korelace a následně také odstranění odlehlých měření metodou RANSAC. RANSAC je však k dispozici pouze v kombinaci s rovinou afinní transformací. Body jsou navíc rozesety po snímku v pravidelné síti, není nijak zjišťováno, zda se jedná skutečně o význačný bod snadno dohledatelný v referenčním podkladu (McNabb 2023).

Gobbi a kol. (2018) vytvořili ortofoto s pomocí modulů Grass GISu. Zpracovávané území bylo pokryto velkým množstvím snímků (téměř 100). Neteler, Mitasova (2004) taktéž popisují zpracování ortofota v uvedeném SW. Další zdarma dostupný SW pro zpracování ortofota je Efoto vyvíjený brazilskou univerzitou Rio de Janeiro State University. Program získal prestižní ocenění od ISPRS. (Ramón, Batlle 2018) úspěšně zpracovává skenované analogové LMS v OpenDroneMap, který byl původně vyvinut pro zpracování snímků z dronů.

Z dalších open-source programů, se kterými má autor pozitivní zkušenost je Meshroom, který je však určen pro pozemní SFM a generování 3D modelů objektů. Nevýhodou je závislost na proprietární Nvidia CUDA technologii, která vyžaduje grafickou kartu od Nvidie. Existuje však i

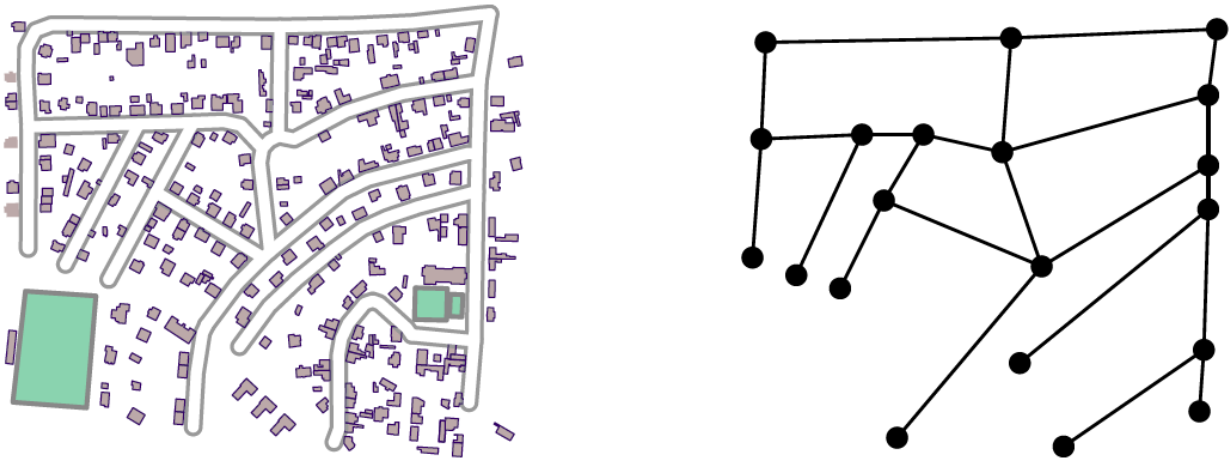
alternativní řešení MeshroomCL běžící na OpenCL, díky kterému je možné spustit Meshroom i na kartách od AMD a Intelu. Samotný program má jinak velmi intuitivní uživatelské rozhraní.

3 Teoretická část

3.1 Grafy

Graf je datová struktura tvořená vrcholy (uzly) a jejich spojnicemi (Bayer 2021). Výhodou grafů je jejich jednoduché názorné zakreslení v rovině obrázkem. Jinak je možno graf zadat také algebraicky maticemi, což lze využít při implementaci algoritmů. Graf s vrcholy n, který obsahuje všechny možné hrany se označuje jako kompletní. Stupeň vrcholu grafu je definován jako počet hran, které z daného vrcholu vycházejí (jsou incidentní) (Kovář 2021). V geoinformatice se grafy využívají pro reprezentaci cestní sítě a následně pro síťové analýzy (Obrázek 3).

Obrázek 3: Cestní síť převedená na graf



Zdroj: Mareš, Valla (2017)

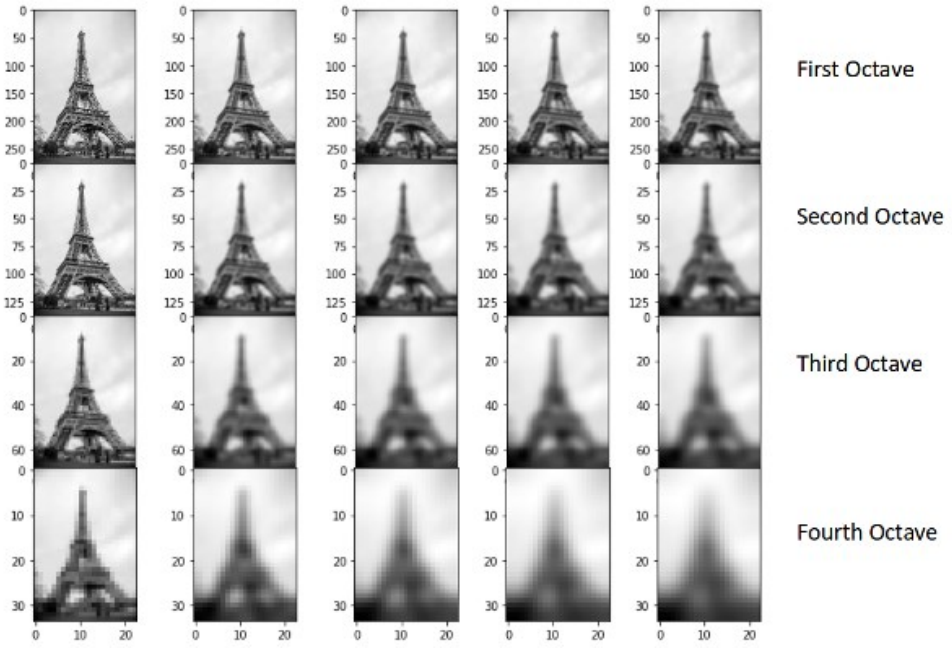
Grafy se dají dělit na neorientované, orientované a částečně orientované (Bayer 2021). U orientovaných grafů je určen směr procházení hran. Jelikož jsou v této práci grafy použity pouze pro reprezentaci sítě komunikací a následně pro export uzlů (křižovatek), je tento teoretický popis velmi stručný a nezahrnuje např. procházení grafů do hloubky/šířky, hledání nejkratší cesty a mnoho dalšího. V metodice je využit například stupeň vrcholu grafu při exportu křižovatek. Pro práci s grafy se hodí knihovna NetworkX pro Python, která obsahuje všechny potřebné základní algoritmy a datové struktury.

3.2 SIFT

SIFT je jeden z operátorů pro vyhledávání a následné párování identických bodů mezi dvěma obrazy. Operátor je odolný vůči změnám v měřítku a rotaci a měl by si poradit např. s afinním zkreslením, změnou úhlu záběru, šumem a změnami v osvětlení (Lowe 2004). Operátor obecně pracuje v několika fázích. První je detekce, v této fázi jsou identifikovány spolehlivé význačné snadno identifikovatelné body. Následuje deskripce, kdy je s pomocí vektoru popsáno okolí bodu pro pozdější párování. Při samotném párování je snahou nalézt odpovídající si body mezi dvěma obrazy.

Ve fázi detekce je na obraz použit gaussovský nízkofrekvenční filtr. Zároveň z obrazu vytvoříme nový obraz s nižším rozlišením. Takto postupujeme několikrát až vytvoříme 4 obrazy (oktávy) vždy s poloviční rozlišením oproti předchozímu rastru a podobně aplikujeme pro každé rozlišení 5 gaussovských filtrů (Obrázek 4) s různou úrovní rozostření a následně odšumění obrazu (Aishwarya 2023).

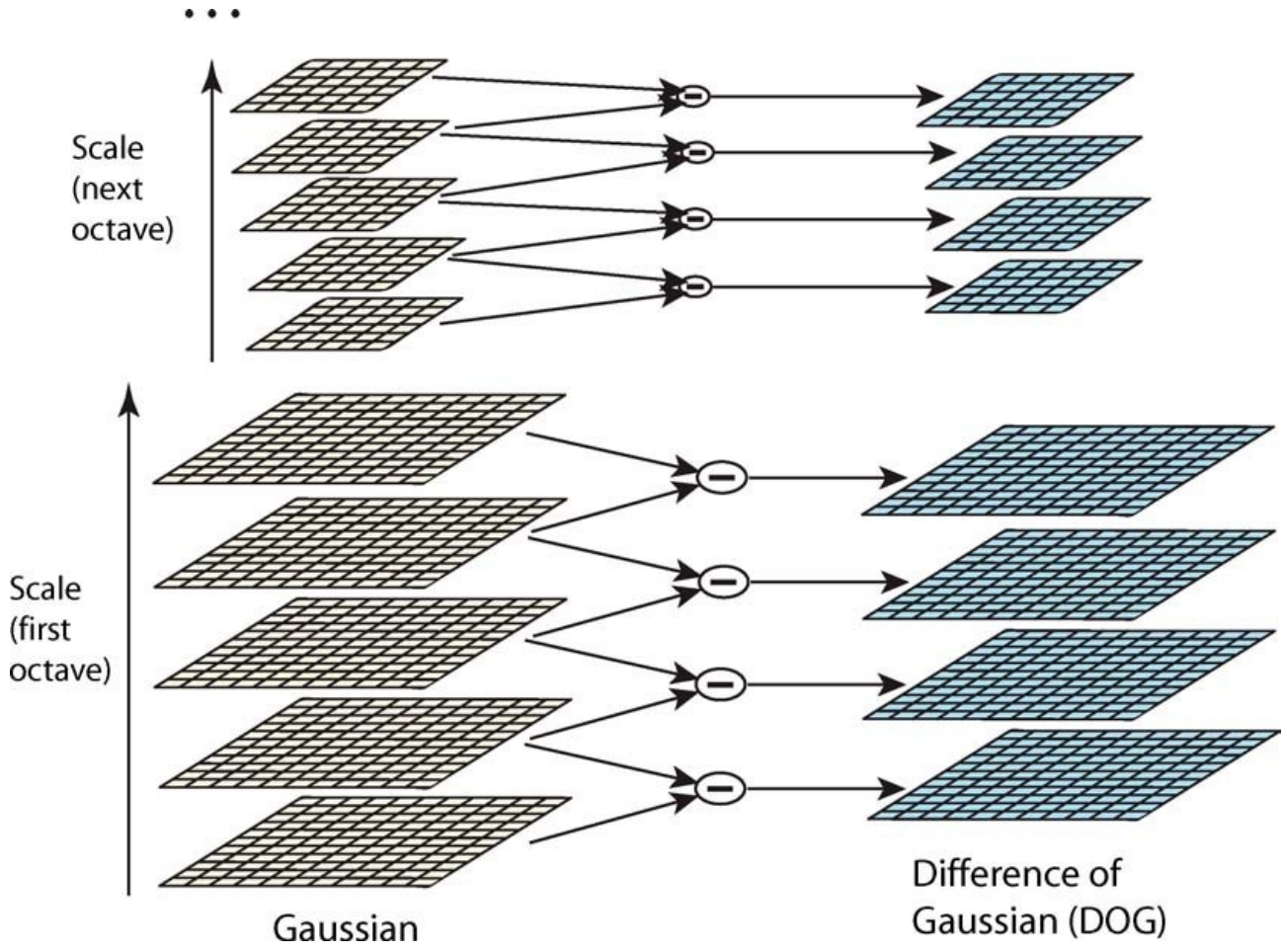
Obrázek 4: Jednotlivé rozostřené oktávy rastru



Zdroj: Aishwarya (2023)

Následuje vytvoření rozdílu gaussiánů (DoG – Difference of Gaussians), tj. v rámci každé oktávy odečteme vždy sousední obrazy s různou úrovní rozostření (Obrázek 5).

Obrázek 5: Rodíly geussiánů v rámci oktávy (DoG)

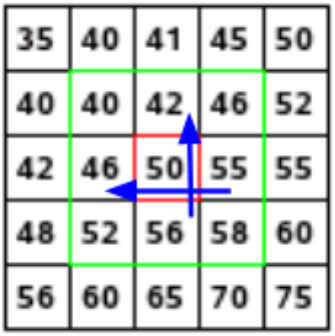


Pozn.: pozor, obrazy v rámci oktávy jsou nyní umístěny vertikálně, nikoliv horizontálně jako na předchozím obrázku.

Zdroj: Aishwarya (2023)

Následuje vyhledávání lokálních minim a maxim v obraze. Při tomto prohledávání je porovnávána hodnota pixelu s hodnotami sousedních pixelů. K tomu dochází nejen v rámci daného obrazu ale i v sousedních obrazech v téže oktávě. Takto jsou nalezeny body (keypoints) invariantní vůči změně měřítka. Pro odfiltrování bodů s nízkým kontrastem je aplikován taylorův rozvoj a následně je taktéž vypočítána hessova matice pro vyloučení bodů blízko hran. Aby byly body (keypoints) invariantní vůči rotaci, je třeba vypočítat orientaci a magnitudo (Aishwarya 2023). Magnitudo vyjadřuje intenzitu pixelu. Výpočet magnituda a orientace je uveden na další straně.

Obrázek 6: Příklad výpočtu magnituda a orientace



Zdroj: Aishwarya (2023)

Výše (Obrázek 6) je uveden výřez rastru, který slouží pro demonstraci výpočtu magnituda a orientace bodu (pixelu).

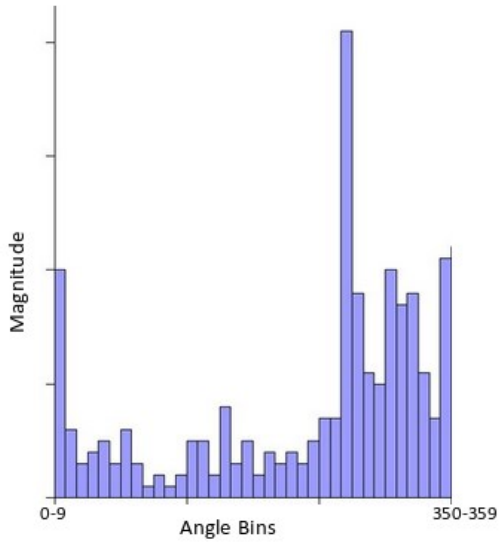
$$G_x, G_y - gradienty\ X, Y \qquad M - magnitudo \qquad \varphi - orientace$$

$$G_x = 55 - 46 = 9$$
$$G_y = 56 - 42 = 14$$

$$M = \sqrt{G_x^2 + G_y^2} = 16{,}64$$
$$\varphi = \text{atan}\left(\frac{G_y}{G_x}\right) = \text{atan}(1{,}55) = 57{,}17$$

Následuje umístění hodnot orientace všech bodů v okolí keypointu do grafu (Graf 1). Na ose X jsou vyneseny úhly orientace φ všech pixelů v okolí keypointu. X je rozdělena do intervalů po deseti stupních s posledním intervalem 350-359. Na Y je magnitudo. Maximum grafu je pak prohlášeno za orientaci bodu. Takto získáme význačné body invariantní vůči rotaci (Aishwarya 2023).

Graf 1: Histogram orientovaných gradientů



Zdroj: Aishwarya (2023)

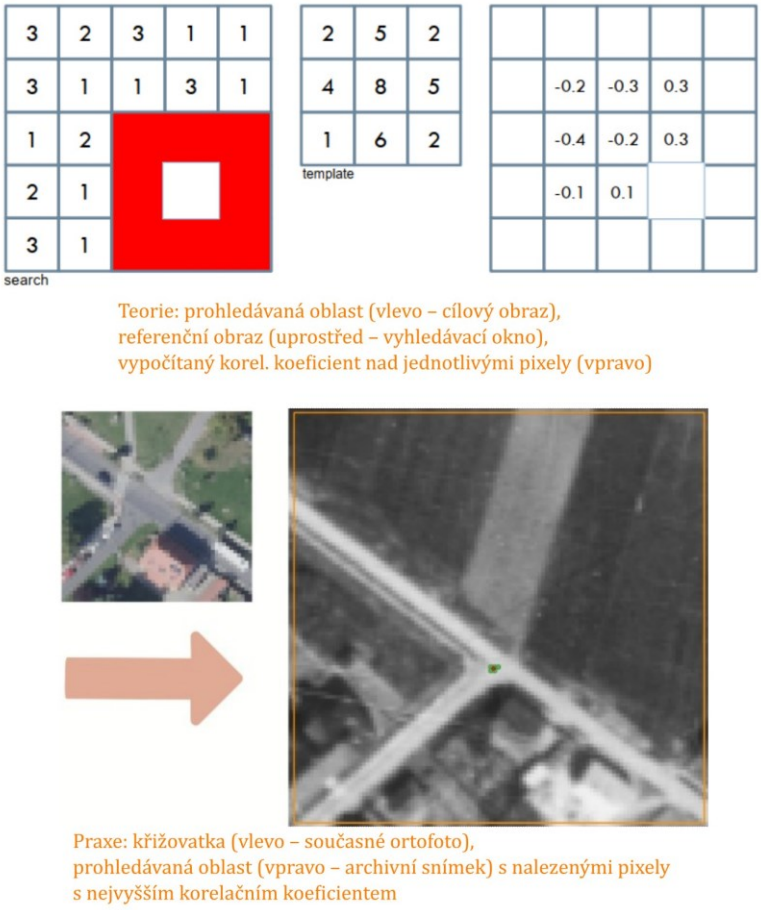
Pro výpočet deskriptoru keypointu se využívají i okolní body, proto je operátor částečně odolný vůči změně osvětlení a kontrastu. Výsledný deskriptor se skládá ze 16 histogramů gradientů po 8 orientačních kategoriích – 128 dimenzionální vektor. Principy operátoru SIFT přehledně vysvětluje Aishwarya (2023), Lowe (2004) nebo Barborka (2016). Pro nalezení odpovídajících si bodů se párují vektory (deskriptory) podle nejmenší eukleidovské vzdálenosti (Hartmann, Havlena, Schindler 2016). V praxi se běžně používá přibližný nejbližší soused (approximate nearest neighbor – ANN) a párování identických bodů mezi dvěma obrazy s pomocí vyhledávacích stromů (Hartmann, Havlena, Schindler 2016). Dle výpisu do konzole je ANN také použito ve výchozím stavu v MicMacu pro párování spojovacích bodů pro výpočet relativní orientace snímků.

Praktickou práci v Pythonu s operátorem SIFT a knihovnou Open CV popisuje Solem (2012), případně jsou k dispozici tutoriály přímo na stránkách knihovny. Existuje také podobný operátor SURF, který je optimalizován pro rychlost výpočtu či operátor ORB, který vytvořili autoři knihovny Open CV. ORB byl taktéž vytvořen pro rychlý výpočet a zároveň není zatížen patentovou ochranou.

3.3 Obrazová korelace

Obrazová korelace (Obrázek 7) je metoda založená na Pearsonově korelačním koeficientu, která slouží pro párování odpovídajících si obrazů. Dochází k porovnání hodnot šedi pixelů mezi referenčním a cílovým obrazem, přičemž cílový obraz je postupně procházen vyhledávacím okénkem (referenčním obrazem) a zároveň je počítán korelační koeficient mezi těmito dvěma výřezy. Výřez cílového obrazu, u kterého vyjde korelační koeficient nejvyšší je náš hledaný výřez (resp. jeho středový pixel). Jedná se o výřez, který nejvíce odpovídá referenčnímu obrazu. Korelační koeficient je vztažen právě ke středovému pixelu oblasti. Aby bylo možno určit středový pixel, musí mít referenční okno lichý počet řádků a sloupců. U barevných snímků je korelováno většinou pouze jedno pásmo (zelené), může však být použita i jejich kombinace (Potůčková 2020). Korelační koeficient je bezrozměrná veličina, která standardně nabývá hodnot <-1, 1>.

Obrázek 7: Princip obrazové korelace



Zdroj: vlastní tvorba/Potůčková (2020)

Při použití popisované metody je nutné, aby rastry měly stejné prostorové rozlišení a stejně orientované buňky (stejné otočení rastrů vůči sobě). Obrazovou korelaci je možno vypočítat v Open CV, které má pro tyto účely rozhraní. Na stránkách zmíněné knihovny je i tutoriál pro „Template matching“ (Open CV 2024). Alternativně je možno použít knihovnu NumPy pro práci s maticemi.

3.4 Podobnostní (Helmertova) transformace (2D)

Jedná se o rovinnou 2D transformaci, která zahrnuje posuny v osách X a Y, otočení a změnu měřítka stejnou v obou osách. Pro výpočet 4 koeficientů mezi 2 kartézskými soustavami jsou potřeba minimálně 2 vlícovací body. Je konformní, ekvidistantní a ekvivalentní, jelikož zachovává úhlové, délkové a plošné poměry. Ve fotogrammetrii se tato transformace používá při převodu vlícovacích bodů z místní soustavy do soustavy referenční (Pavelka 2003b). Převod mezi výchozí (xy) a cílovou (XY) souřadnicovou soustavou je stanoven vztahy (Bayer 2020):

$$\Omega - rotace, \qquad \Delta x, \Delta y - \ posuny\ ve\ směru\ X\ a\ Y, \qquad q - \ měřítko$$
$$\lambda_1, \lambda_2 - \ koeficienty\ transformace, \qquad xy - \ výchozí\ souř., \qquad XY - \ cílové\ souř.$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} = q \begin{bmatrix} cos\Omega & -sin\Omega \\ sin\Omega & cos\Omega \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \lambda_1 & -\lambda_2 \\ \lambda_2 & \lambda_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

V případě nadbytečného počtu identických bodů (více než 2), je třeba uplatnit vyrovnání s pomocí metody nejmenších čtverců. V takovém případě hovoříme o Helmertově transformaci (Bayer 2020):

$$A - matice\ plánu, \quad l - vektor\ měření, \quad \lambda - vektor\ neznámých$$

$$A = \begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ . & . & . & . \\ . & . & . & . \\ x_n & -y_n & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ . & . & . & . \\ . & . & . & . \\ y_n & x_n & 0 & 1 \end{bmatrix} \qquad l = \begin{bmatrix} X_1 \\ . \\ . \\ X_n \\ Y_1 \\ . \\ . \\ Y_n \end{bmatrix}$$

Řešení rovnic:

$$\lambda = (A^T A)^{-1} A^T l \qquad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \Delta x \\ \Delta y \end{bmatrix}$$

$$M - měřítko, \qquad \Omega - stočení$$

$$M = \sqrt{\lambda_1^2 + \lambda_2^2} \qquad \Omega = \arctan\frac{\lambda_2}{\lambda_1}$$

3.5 Afinní transformace (2D)

V případě afinní transformace dochází při převodu souřadnic ke dvěma posunům (podle osy X a Y), ke dvěma rotacím a ke změně měřítka podél obou os (podél každé osy jiné měřítko). K vypočítání 6 koeficientů transformace jsou potřeba minimálně 3 vlícovací body (Bayer 2020). Ve fotogrammetrii se afinní transformace využívá hlavně k transformaci rastrových souřadnic na snímkové. Ke stanovení vztahu v takovém případě slouží rámové značky LMS (Pavelka 2003b). Afinní transformace oproti podobnostní zavádí zkosení obrazu (různé koeficienty měřítka v ose X a Y (Obrázek 8)), které ale není součástí geometrického modelu středového promítání. Z tohoto důvodu není afinní transformace příliš vhodná pro georeferencování LMS. Převod mezi výchozí (xy) a cílovou (XY) souřadnicovou soustavou je stanoven vztahy (Bayer 2020):

$$\Omega_x, \Omega_y - rotace, \qquad \Delta x, \Delta y - \ posuny\ ve\ směru\ X\ a\ Y, \qquad q_x, q_y - \ měřítka$$
$$\lambda_1, \lambda_2, \lambda_3, \lambda_4 - \ koeficienty\ transformace, \qquad xy - \ výchozí\ souř., \qquad XY - \ cílové\ souř.$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} q_x\, cos\Omega_x & -q_y sin\Omega_y \\ q_x\, sin\Omega_x & q_y\, cos\Omega_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_3 & \lambda_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

V případě nadbytečného počtu identických bodů (více než 3), je třeba uplatnit vyrovnání s pomocí metody nejmenších čtverců (Bayer 2020):

$$A - matice\ plánu, \quad l - vektor\ měření, \quad \lambda - vektor\ neznámých$$

$$A = \begin{bmatrix} x_1 & -y_1 & 0 & 0 & 1 & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_n & -y_n & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & x_n & y_n & 0 & 1 \end{bmatrix} \qquad l = \begin{bmatrix} X_1 \\ . \\ . \\ X_n \\ Y_1 \\ . \\ . \\ Y_n \end{bmatrix}$$

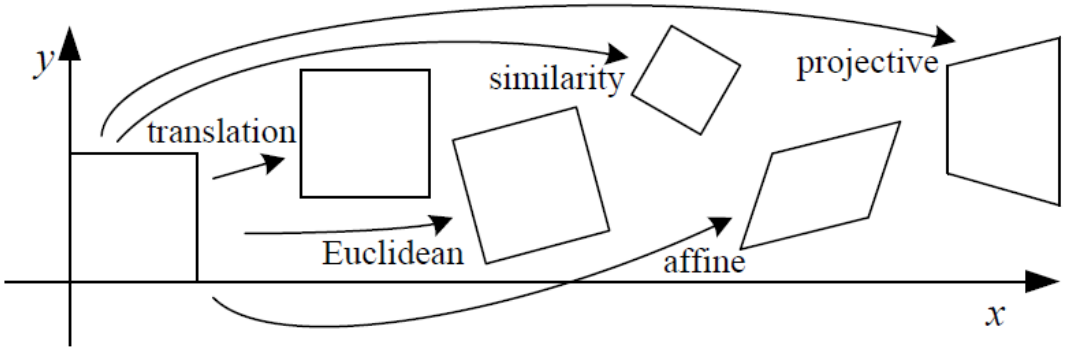
Řešení rovnic:

$$\lambda = (A^T A)^{-1} A^T l \qquad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \Delta x \\ \Delta y \end{bmatrix}$$

$$M_x, M_y - \text{měřítka}, \quad \Omega_x, \Omega_y - \text{stočení}$$

$$M_x = \sqrt{\lambda_1^2 + \lambda_3^2} \quad M_y = \sqrt{\lambda_2^2 + \lambda_4^2} \quad \Omega_x = \arctan\frac{\lambda_3}{\lambda_1} \quad \Omega_y = \arctan\frac{\lambda_4}{\lambda_2}$$

Obrázek 8: Vizualizace transformace podobnostní, afinní, projektivní



Zdroj: Szeliski (2022)

3.6 World file

Rastrová data v GIS oblasti jsou často distribuována v běžných formátech jako je *.tif, *.jpg aj. Samotný rastr má pixelový souřadnicový systém (sloupec, řádek – x, y). Jednou z možností, jak umístit rastr do geodetického souřadnicového systému pro jeho korektní vizualizaci a analýzu v GIS systémech je vytvoření tzv. World file souboru. World file je jednoduchý textový soubor pojmenovaný stejně jako samotný rastr s příponou, která se odvíjí od použitého rastrového formátu. Pro *.jpg je to *.jgw, pro *.tif pak *.tfw, existují však i další formáty. Alternativní možností je uložení informace o souřadnicovém systému přímo do hlavičky samotného rastrového souboru. World file obsahuje pouze 6 řádků (Obrázek 9). Na každém je pak jeden z koeficientů afinní transformace (Esri 2024), viz předchozí kapitola.

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_3 & \lambda_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$x = \lambda_1 x + \lambda_2 y + \Delta x$$
$$y = \lambda_3 x + \lambda_4 y + \Delta y$$

Obrázek 9: Obsah World file pro snímek Kobylí 1984 s popisem

-0.3540303707910937	λ_1 – velikost pixelu ve směru x
0.03754364929849707	λ_3 – rotace kolem y
0.03754364929849707	λ_2 – rotace kolem x
0.3540303707910937	λ_4 – velikost pixelu ve směru y
-576837.6652261121	Δx – x souřadnice středu levého horního pixelu
-1195069.5395885226	Δy – y souřadnice středu levého horního pixelu

Zdroj: vlastní tvorba/popis koeficientů dle Wikipedia (2024b)

Pokud jsou koeficienty λ_1 a λ_4 / λ_2 a λ_3 stejné (absolutní hodnota), pak se nejedná o afinní transformaci, ale o transformaci podobností. Z World file souboru lze tak velmi snadno určit, s pomocí jaké transformace georeference vznikla. Výše (Obrázek 9) je zobrazena georeference jednoho LMS pro oblast Kobylí 1984, která byla vytvořena s pomocí vlastních vlícovacích bodů, podobnostní (Helmertovy) transforamace a World file byl vygenerován skriptem `0step_georeference2worldfile.py`. LMS byl zgeoreferencován do S-JTSK, jedná se o pro GIS typické EPSG 5514, proto je prohozena X a Y souřadnice (poslední 2 řádky World file).

3.7 Projektivní (kolineární) transformace (2D)

Jedná se o transformaci, která popisuje geometrický model středového zobrazení dvou rovinných souřadnicových systémů. Všechny projekční paprsky jsou přímé a prochází jedním společným bodem (středem promítání) (Pavelka 2003b). Rovnice mají tvar:

$$x, y - rastrové\ souřadnice \quad X, Y - geodetické\ souřadnice$$
$$a_1 \dots a_n, b_1 \dots b_n, c_1, c_2 - koeficienty\ projektivní\ transformace$$

$$x = \frac{a_1 X + a_2 Y + a_3}{c_1 X + c_2 Y + 1} \qquad y = \frac{b_1 X + b_2 Y + b_3}{c_1 X + c_2 Y + 1}$$

Vynásobením zlomků jmenovatelem lze rovnice linearizovat. Členy rovnic je pak možno sepsat do matic a vypočítat koeficienty transformace při znalosti všech souřadnic v obou souřadnicových systémech u 4 bodů pro dopočet 8 neznámých. Jelikož dále v metodice byla uplatněna metoda RANSAC (skript `7step_RANSAC.py`), kde je tato transformace využívána v základním tvaru bez nadbytečného množství bodů a metody vyrovnání pomocí nejmenších čtverců, byly využity níže uvedené vztahy pro výpočet všech koeficientů transformace s pomocí 4 vlícovacích bodů (Pavelka 2003a):

$$\begin{bmatrix} X & Y & 1 & 0 & 0 & 0 & -xX & -xY \\ 0 & 0 & 0 & X & Y & 1 & -yX & -yY \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} x \\ y \\ . \\ . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Výše je dosazen pouze 1 *bod, dalších* 6 řádků *je určeno pro zbylé* 3 *body. Maticové* řešení *rovnice*:

$$A\ a = X$$
$$a = A^{-1}\ X$$

Žádné 3 vlícovací body nesmí ležet na jedné přímce. Transformace zachovává linearitu, průsečíky přímek a dvojpoměr délkových úseků mezi body ležícími na jedné přímce. Nezachovává úhlové ani délkové poměry (takže ani plošné). Projektivní transformace je základem jednosnímkové fotogrammetrie (Pavelka 2003b). Jelikož se však jedná jen o rovinnou 2D transformaci, je vhodná pouze pro georeferencování LMS rovinných oblastí. Pokud není zemský povrch zcela rovinný, dochází k radiálním posunům bodů (Pavelka 2003a).

3.8 DLT – direct linear transformation (3D)

Přímá lineární transformace (direct linear transformation – DLT) je prostorová 3D transformace, kterou lze použít pro zpracování LMS. Využívá přetvořených rovnic kolineární transformace. Pro výpočet 11 koeficientů je potřeba minimálně 6 vlícovacích bodů, u nichž známe souřadnice ve snímkovém i geodetickém souřadnicovém systému (včetně nadmořské výšky). Vlícovací body nesmí ležet v jedné rovině. DLT lze použít i pro práci se snímky z neměřických komor mimo jiné k určení prvků vnitřní orientace. Výhodou DLT je možnost vyhodnocovat snímky bez znalosti konstanty kamery a možnost práce i s výřezy snímků (Pavelka 2003b). Rovnice transformace jsou následující:

$$x, y - rastrové\ souřadnice \quad X, Y, Z - geodetické\ souřadnice$$
$$a_1 \dots a_n, b_1 \dots b_n, c_1 \dots c_n - koeficienty\ projektivní\ transformace$$

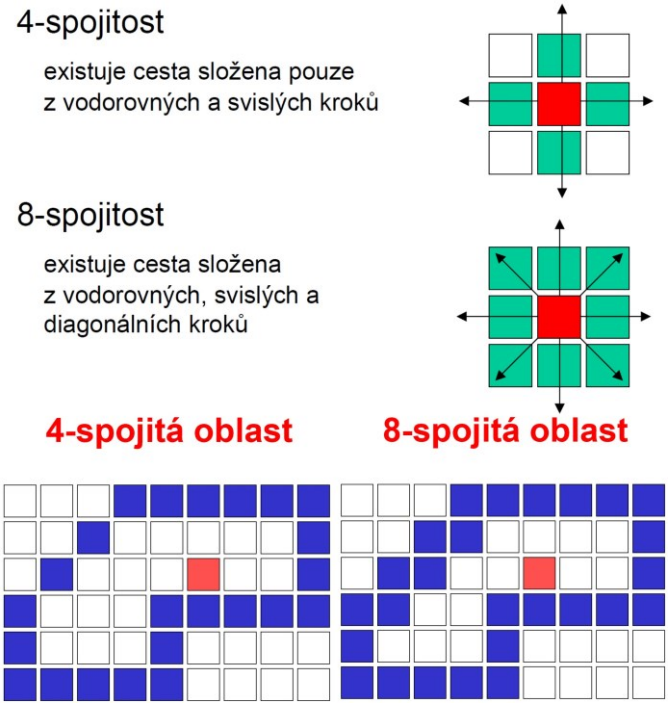
$$x = \frac{a_1X + a_2Y + a_3Z + a_4}{c_1X + c_2Y + c_3Z + 1} \qquad y = \frac{b_1X + b_2Y + b_3Z + b_4}{c_1X + c_2Y + c_3Z + 1}$$

Vynásobením rovnic jmenovatelem získáme linearizované rovnice. Poté lze použít vyrovnání metodou nejmenších čtverců, výsledkem však může být často singulární matice nebo matice, která se jí blíží (Szeliski 2022). V počítačové grafice jsou koeficienty DLT jednotlivé členy matice kamery. Využity jsou homogenní souřadnice, přičemž rovnice jsou řešeny s pomocí tzv. singulárního rozkladu matice (SVD – singular value decomposition). Praktický výpočet matice kamery v Pythonu uvádí Solem (2012), případně lze využít hotovou implementaci výpočtu koeficientů DLT využívající SVD v Pythonu publikovanou na GitHubu, kterou vytvořil Ankita (2019) (použito v této práci).

3.9 SeedFill algoritmus

Jednou ze základních metod pro barevné vyplňování oblastí v rastru, které jsou vymezeny rastrovou hranicí je algoritmus SeedFill (semínkové vyplňování). Důležitým parametrem skriptu jsou souřadnice semínka, což je jeden z vnitřních bodů (pixelů) oblasti. Od tohoto semínka je následně prohledáváno jeho okolí a jednotlivým vnitřním pixelům oblasti je přiřazena požadovaná barva. Postupně se tímto záplavovým způsobem vyplňuje celá oblast, dokud algoritmus u každého pixelu v jeho okolí nenarazí na hranici nebo na již vyplněný pixel požadovanou barvou. Za sousední pixely (Obrázek 10) mohou být považovány pixely v horizontálním a ve vertikálním směru (čtyřspojité okolí), případně navíc i v diagonálním směru (osmispojité okolí). Použitému okolí musí odpovídat i definice rastrové hranice. Např. pokud by byla hranice sestavená v některé části obrazce pouze z diagonálních pixelů, pak by při použití osmispojitého okolí nebyla respektována a vyplněna by byla i oblast mimo obrazec. Algoritmus lze implementovat rekurzivně, což však není příliš efektivní, jelikož jsou pixely testovány opakovaně (Žára a kol. 2005). Na základě tohoto algoritmu by teoreticky mohlo být založeno vyplňování oblastí v jednoduchém SW Malování přítomném defaultně ve Windows (nástroj se symbolem plechovky). Tímto je možno ilustrovat použití algoritmu. Modifikovaný SeedFill byl použit ve skriptu `6step_flood_collapse_clusters.py`, kde je algoritmus použit pro shlukování bodů.

Obrázek 10: Algoritmus SeedFill

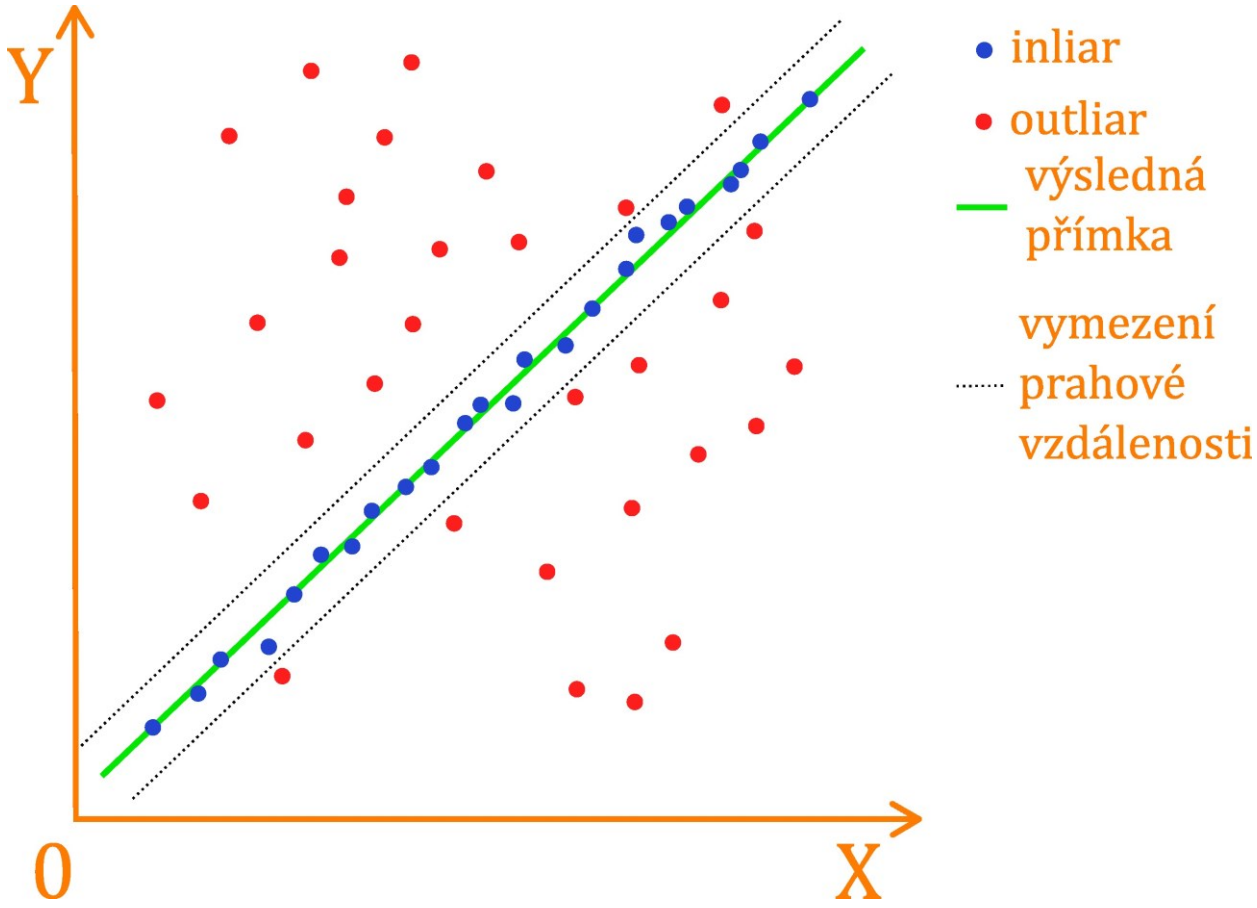


Zdroj: Ježek (2021)

3.10 RANSAC algoritmus při transformaci obrazu

RANSAC (RANdomSAmple Consensus) je iterativní metoda pro nalezení korektního (geometrického) modelu pro (nejen obrazová) data, která obsahují odlehlá měření. Měření v datech lze rozdělit na korektní (inliers – vyhovují modelu) a odlehlá (outliars – nevyhovují modelu). Při použití metody RANSAC je cílem odfiltrovat právě odlehlá měření (Solem 2012). Na data vyhovující modelu je pak typicky uplatněno vyrovnání s pomocí metody nejmenších čtverců. Za příklad mimo oblast geoinformatiky lze uvést proložení regresní přímky daty, přičemž nejprve jsou odfiltrována odlehlá měření s pomocí RANSACu (Obrázek 11). Samotná přímka je následně vypočítána již jen z bodů vyhovujících modelu (inliers), které splňují podmínku prahové vzdálenosti.

Obrázek 11: RANSAC – inliers a outliers



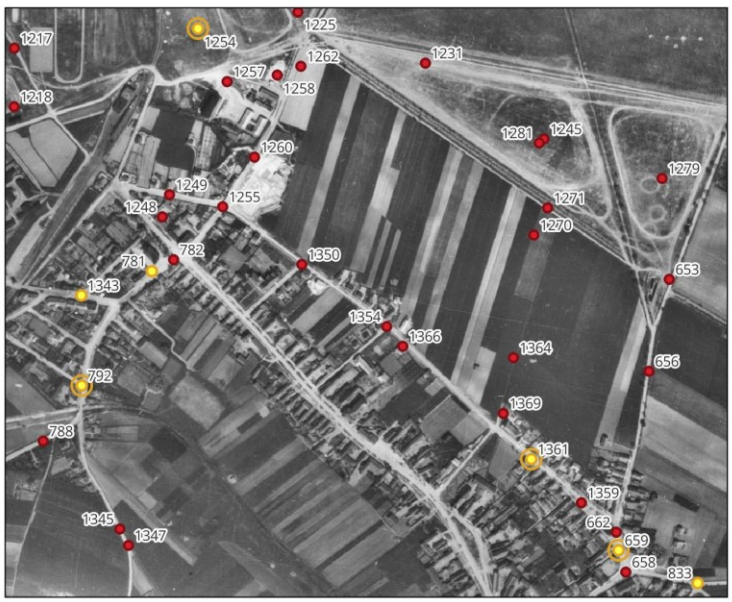
Inliers – body v okolí přímky vyhovující modelu (splňují prahovou vzdálenost), outliars – nevyhovují modelu, jsou odfiltrovány. Metodou nejmenších čtverců je z inliers vypočítána přímka.

Zdroj: upraveno dle Wikipedie (2024)

Při použití v geoinformatice lze metodu využít pro nalezení transformace mezi 2 obrazy s pomocí VLB za situace, kdy je součástí množiny VLB mnoho zcela nekorektních špatně sesbíraných VLB (odlehlých měření, které vznikly například při automatizovaném vyhledávání, což je případ této práce). V metodice je RANSAC implementován v kombinaci s 3D fotogrammetrickou transformací DLT, případně alternativně lze ve skriptu použít 2D projektivní transformaci. V každé iteraci je z vlícovacích bodů vybráno 6 bodů, přičemž díky znalosti geodetických i rastrových souřadnic je možno dopočítat koeficienty transformace DLT. Počet iterací a prahová vzdálenost jsou dány parametrem na vstupu. V dané iteraci je provedena transformace všech VLB s pomocí dopočítaných koeficientů DLT a zkoumá se, kolik bodů vstupujících do algoritmu splňuje podmínku prahové vzdálenosti (mezi bodem na snímku dopočítaným s pomocí DLT a bodem nalezeným obrazovou korelací – rastrové souřadnice v LMS). Body z iterace, ve které nejvíce bodů splňuje podmínku prahové vzdálenosti, jsou uloženy do výstupního souboru (pouze inliers). Jedná se o finální body určené pro zpracování ortofota. Body z téže iterace, které však nesplňovaly podmínku prahové vzdálenosti (outliars) jsou odfiltrovány a nejsou používány při další práci. Parametry pro algoritmus jsou tedy počet iterací, prahová vzdálenost od výsledného modelu, v případě této práce a použité aplikační oblasti ještě také transformace (2D projektivní/3D DLT). S vyšším počtem iterací roste pravděpodobnost nalezení správného, popřípadě lepšího modelu.

Na následujících dvou stranách je ukázka čtyř iterací RANSACu (Obrázek 12). Ukázka slouží pouze pro demonstrační účely, testovacím územím byly Milovice 1938, použita byla projektivní transformace, 40 000 iterací a 4 m prahová vzdálenost. V praxi (v metodice) dále v práci byly použity jiné parametry. V mapách je znázorněn postupný průběh algoritmu, od nejhorší iterace, kdy do výsledného modelu spadaly pouze 4 VLB, až po iteraci nejlepší s 11 VLB. Obstojné byly i iterace č. 704, 6 485 a 8 356, ve kterých modelu vyhovovalo 10 VLB. Pokud se vyskytne více modelů se stejným počtem bodů (inliers), pak se zkoumá průměrná odchylka VLB od modelu. Model s nižší odchylkou je považován za lepší. V atributových tabulkách v ukázkách (Obrázek 12) jsou uvedeny již pouze inliers.

Obrázek 12: Ukázka průběhu (iterací) RANSAC algoritmu



Referenční podklad – současné ortofoto

(pouze pro informativní účely, vizualizace současného ortofota s křižovatkami, obsahuje pouze vybrané křižovatky, u kterých vyšel korelační koeficient obrazové korelace v LMS > 0,4)

- křižovatka s číselným ID (VLB v referenčním podkladu)

Níže následují ukázky 4 iterací RANSACu. Jsou seřazeny vzestupně od nejméně úspěšné iterace, kdy jsou 4 body ustanovující transformaci mezi souřadnicovými systémy (geodetické souř.-referenční ortofoto --> rastrové souř.-LMS) téměř všechny invalidní a modelu vyhovují právě pouze tyto 4 body. Na konci ukázky je pak nejúspěšnější iterace s největším počtem vyhovujících bodů. Pro každý bod jsou známy geodetické a rastrové souřadnice do LMS.

RANSAC parametry:

testovací snímek: Milovice 1938
počet iterací: 40 000
prahová vzdálenost: 4 m
použitá transformace: projektivní

Popis atributů výstupních VLB:

cross_id: ID křižovatky
quality: hodnota korelačního koeficientu
RAN_dist_pix: odchylka od modelu v RANSACu (v pixelech, přičemž 1 pixel = 1 metr), body tvořící transformaci mají pochopitelně nulovou odchylku

Iterace č. 1:

cross_id	quality	RAN_dist_pix	
656	0,473	0	počet inliers: 4
833	0,403	0	
1245	0,404	0	průměrná odchylka
1257	0,403	0	od modelu: 0 m

V podstatě všechny VLB u iterace 1 jsou až na VLB ID 833 chybně spárované (resp. nespárovatelné kvůli změnám v krajině) s referenčním podkladem. Výsledkem je vadný geometrický model, do kterého nespadaly žádné další VLB, kromě 4 VLB tvořících transformaci.

- VLB – 4x inlier (body ustanovující model sloužící k výpočtu koeficientů transformace --> nulová odchylka od modelu)
- VLB – inlier (odchlyka od modelu nižší než prahová vzdálenost --> vyhovuje modelu)
- VLB – outliar (odchylka od modelu vyšší než prahová vzdálenost --> nevyhovuje modelu)

Iterace č. 17:

cross_id	quality	RAN_dist_pix	
659	0,47	0	počet inliers: 7
792	0,438	0	
1254	0,518	0	průměrná odchylka
1361	0,44	0	od modelu: 1,064 m
1343	0,663	0,317	
833	0,403	3,282	
781	0,498	3,849	

VLB s ID 1254 je zcela chybný a ID 1361 nepřesný, proto je model poměrně nedokonalý a obsahuje pouze 7 VLB.

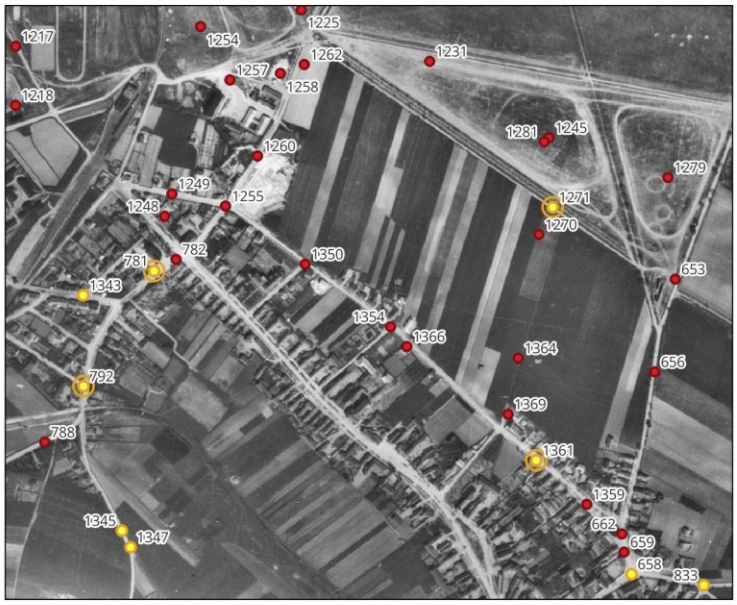
0 250 500 m

Referenční podklad – současné ortofoto

(ortofoto zopakováno pro informativní účely)

- křižovatka s číselným ID (VLB v referenčním podkladu)

Níže jsou uvedeny další 2 iterace RANSACu tentokrát již úspěšnější oproti předchozím.



Iterace č. 168:

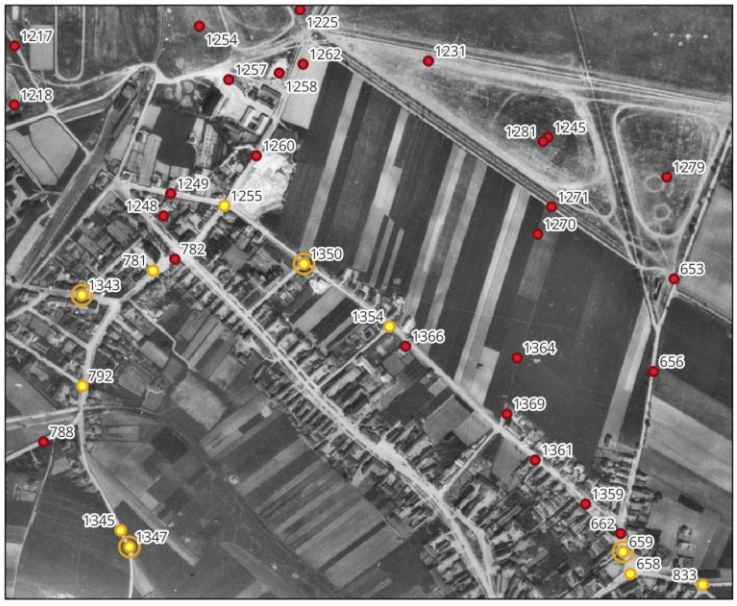
cross_id	quality	RAN_dist_pix
781	0,498	0
792	0,438	0
1271	0,406	0
1361	0,44	0
658	0,457	2,096
1343	0,663	2,73
1347	0,593	3,329
1345	0,436	3,914
833	0,403	3,937

počet inliers: 9

průměrná odchylka od modelu: 1,778 m

VLB ID 1361 a 1271 jsou nepřesně nalezené. Jedná se o výrazně lepší model oproti předchozím případům.

- VLB – 4x inlier (body ustanovující model sloužící k výpočtu koeficientů transformace --> nulová odchylka od modelu)
- VLB – inlier (odchlyka od modelu nižší než prahová vzdálenost --> vyhovuje modelu)
- VLB – outliar (odchylka od modelu vyšší než prahová vzdálenost --> nevyhovuje modelu)



Iterace č. 35 968:

cross_id	quality	RAN_dist_pix
659	0,47	0
1343	0,663	0
1347	0,593	0
1350	0,416	0
1255	0,544	0,51
1345	0,436	1,308
833	0,403	3,025
781	0,498	3,36
658	0,457	3,497
792	0,438	3,611
1354	0,424	3,761

počet inliers: 11

průměrná odchylka od modelu: 1,733 m

Jedná se o nejlepší nalezený model s celkem 11 body. Všechny body, které mu vyhovují, jsou správně spárovány s referenčním podkladem. Jinak se v iteracích č. 704, 6485 a 8356 se podařilo nalézt modely vždy o 10 bodech, které byly také vesměs kvalitní.

0 250 500 m

Zdroj: vlastní tvorba, podkladová data ČÚZK (2023), Cenia (2022)

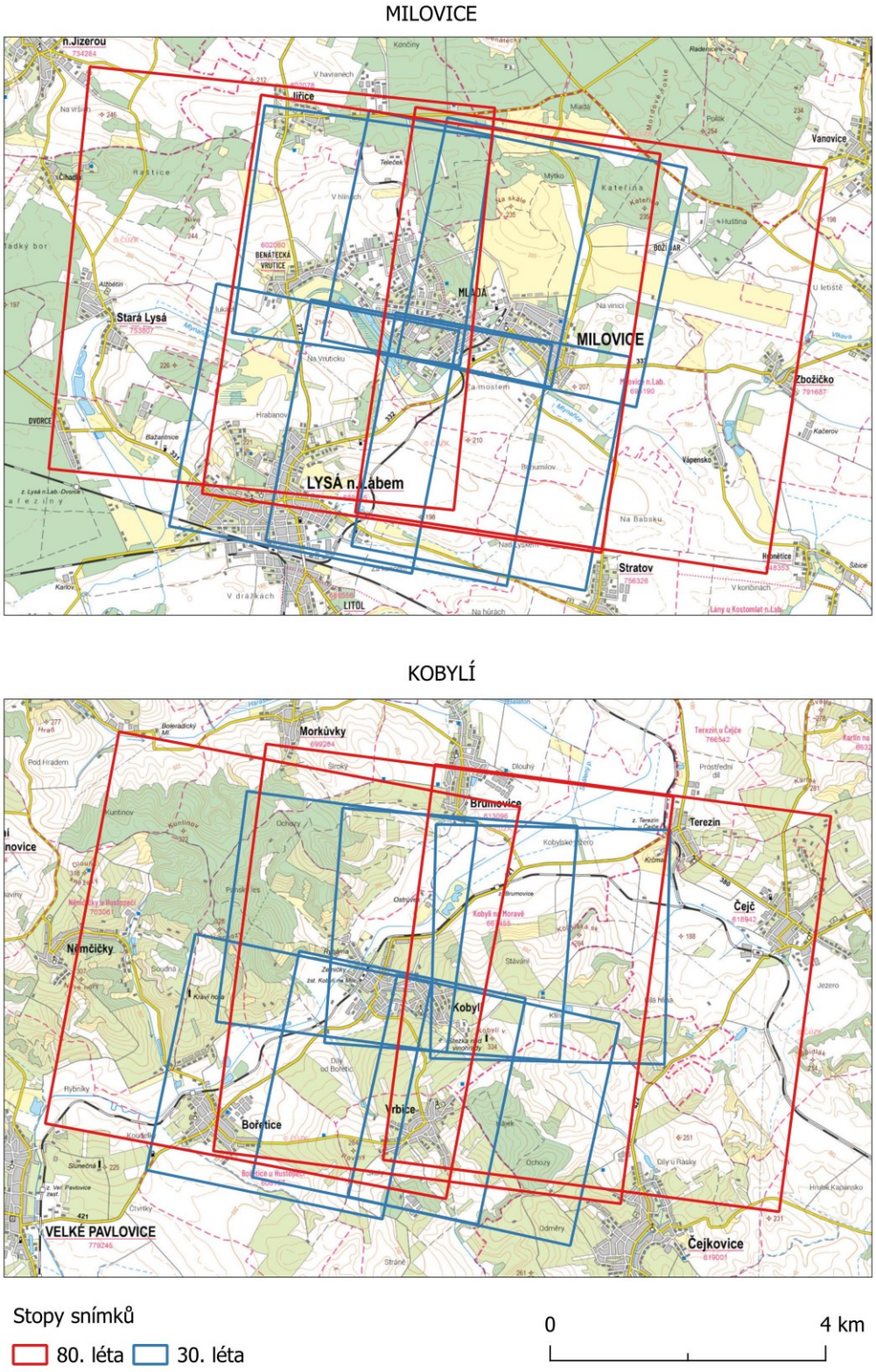
4 Metodická část

4.1 Charakteristika zájmových území

Zájmová území byla volena s ohledem na faktory, které mají potenciál významným způsobem ovlivnit kvalitu výsledného ortofota odvozeného z archivních LMS. Pro zhodnocení a porovnání kvality výsledků metodiky byly uvažovány jak podmínky socioekonomické, tak i fyzickogeografické. Z hlediska socioekonomického se jednalo hlavně o prozkoumání rozdílů venkovských a městských území, v nichž lze obecně zaznamenat významné změny ve složení a struktuře krajinného pokryvu projevující se například rozšiřováním komerční, suburbánní či průmyslové zástavby, scelováním polí nebo výstavbou nových komunikací. Toto lze považovat za klíčové s ohledem na zvolenou metodiku. Výškovou členitost povrchu, rozdíly mezi nížinou a pahorkatinou, lze pak označit za jeden z důležitých fyzickogeografických činitelů, které mají obecně vliv na přesnost ortogonálního překreslení LMS.

S ohledem na předchozí projekty katedry Aplikované geoinformatiky a kartografie Přf UK byla vybrána zájmová území Milovice a Kobylí, a to vždy ve dvou časových řezech – Milovice 1938, 1989 a Kobylí 1938, 1984. Tato území zároveň splňují výše zmíněné charakteristiky. Vzhledem k velkému časovému odstupu mezi pořízením archivních a současných referenčních podkladových dat lze předpokládat vysokou úroveň změn v krajině. Přesnější vymezení daných oblastí je zobrazeno na následujícím obrázku (Obrázek 13). Území byla součástí projektu NAKI Dědictví zaniklých krajin (www.zaniklekrajiny.cz), v rámci kterého vznikl například Digitální atlas zaniklých krajin Česka (www.zaniklekrajiny.cz/atlas/) a další výstupy.

Obrázek 13: Vymezení zájmových území s pomocí stop snímků



Zdroj: vlastní zpracování/podkladová data: Základní mapa (ČÚZK 2022b), Archiv leteckých měřických snímků (VGHMÚř Dobruška 2022)

Město Milovice se nachází ve Středočeském kraji v okrese Nymburk nedaleko od Lysé nad Labem. Od 90. let zde několikanásobně vzrostl počet obyvatel z cca 1200 až na 9000 (Digitální atlas zaniklých krajin 2019). První písemná zmínka je z roku 1396. V místě byl už od doby Rakouska-Uherska vojenský prostor, ve kterém se postupně vystřídaly armády rakousko-uherská, československá, německá a sovětská. V roce 1991 z města odešla sovětská vojska včetně jejich rodinných příslušníků, kteří zde se sloužícími vojáky pobývali. Poté došlo k úpravě uvolněných městských bytů a nastal poměrně masivní nárůst obyvatel. Vojenský prostor zanikl, zbylá infrastruktura včetně letiště je často opuštěná a chátrá. Rozvoj zástavby je znázorněn dále v textu (Obrázek 14). V oblasti se nachází některé pamětihodnosti a zajímavosti jako například kostel sv. Kateřiny Alexandrijské, Vojenský hřbitov Milovice, Muzeum Milovicka nebo přírodní rezervace Pod Benáteckým vrchem (Wikipedia 2023). Z dalších zajímavých míst lze jmenovat cvičiště pro obrněná vozidla nebo velký zábavní park.

Reliéf je nízký plochý tvořený Milovickou tabulí. Jedná se o typickou polabskou rovinu s minimálními výškovými rozdíly nepřesahujícími 20 m. Na sever od Milovic je pak nízká Vrutická pahorkatina s nejvyšším vrcholem 255,4 m v katastru obce Lipník, z dalších vrcholů lze jmenovat Benátecký vrch (251,4 m), v jehož blízkosti je rozsáhlá skládka odpadů, která převýšila původní přirozený vrchol.

Bývalé vojenské cvičiště je tvořeno lesní a lesostepní krajinou, na jih od Milovic se nachází intenzivně využívaná rozsáhlá pole Polabské nížiny. V oblasti byla vyhlášena evropsky významná lokalita NATURA 2000 Milovice – Mladá. Předmětem ochrany jsou chráněné druhy rostlin, obojživelníků a hmyzu, které se vyskytují na stanovištích nezasažených chemizací zemědělství v bývalém vojenském prostoru. V území se dále nachází dvě přírodní rezervace. V oplocené rezervaci žijí stáda divokých koní, praturů a zubrů, která spásáním udržují lesostepní až stepní krajinu. Kompletní a ucelené informace o území je možno nalézt v Digitálním atlase zaniklých krajin (2019), z něhož vychází i tento stručný popis zájmového území.

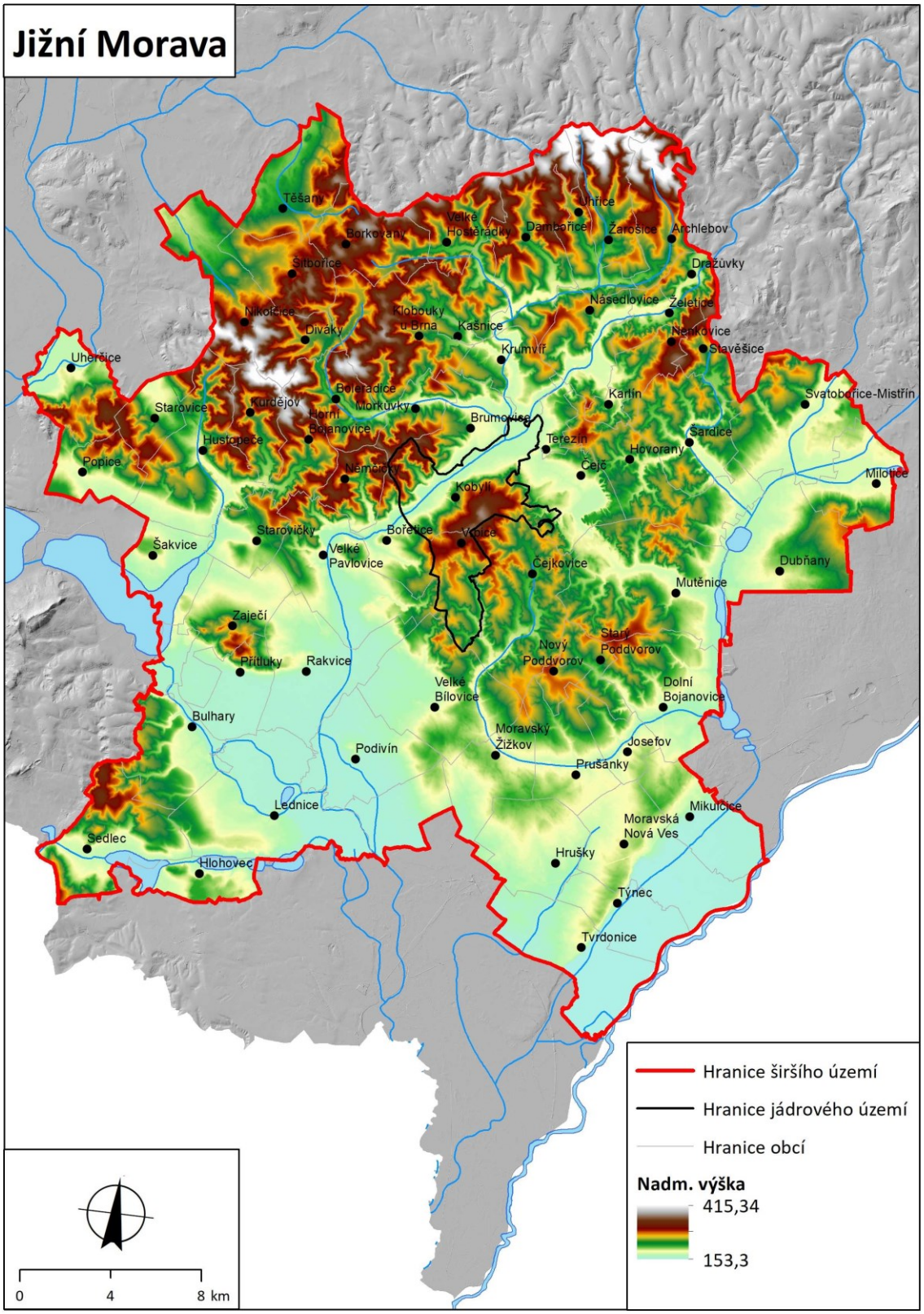
Obrázek 14: Rozvoj zástavby v Milovicích – rok 1938 vs 2022

Zdroj: ČÚZK (2023), Cenia (2022)

Obec Kobylí leží v Jihomoravském kraji v okrese Břeclav. Její vznik se odhaduje na přelom 12. a 13. století. Jedná se o jednu z největších vinařských oblastí, známou také jako Velkopavlovická, náležící do regionu hanáckého Slovácka. Kromě vinné révy se v oblasti pěstují i další ovocné plodiny. Obec leží na svahu Kobylího vrchu. V minulosti se východně od Kobylího nacházelo Kobylské jezero, které však bylo po vyschnutí v roce 1836 rozoráno. V současnosti je na jeho místě napřímené koryto říčky Trkmanky a mokřadní Biocentrum Ostrůvek, napomáhající biodiverzitě a zadržování vody v krajině. Ze zajímavých míst stojí za zmínku Stezka nad vinohrady (netradiční rozhledna), Muzeum obce Kobylí či kostel sv. Jiří (Wikipedia 2023a, Obec Kobylí 2023).

Oblast leží v geomorfologickém celku Ždánický les, který typologicky odpovídá pahorkatině (Obrázek 15) s nadmořskými výškami od cca 170 m do 330 m s typickými výškovými rozdíly nejčastěji kolem 100 m a nejvyšším bodem nazývaným Kobylí vrch (334 m). Typické jsou široké a ploché hřbety se zarovnanými povrchy. Území je odvodňováno říčkou Trkmankou, v jejíž nivě se nachází soustava rybníků a několik menších pískoven. Právě v této nivě bylo v minulosti průtočné Kobylské jezero cca 4 km dlouhé a 2 km široké. Oblast je spíše sušší a intenzivně zemědělsky využívaná. Z tohoto důvodu se zde nezachovala žádná území, která by mohla být vyhlášena jako chráněná. Uvedený text vychází z Digitálního atlasu zaniklých krajin (2020), v němž lze nalézt širší popis tohoto území.

Obrázek 15: Výškové poměry v širším okolí Kobylího

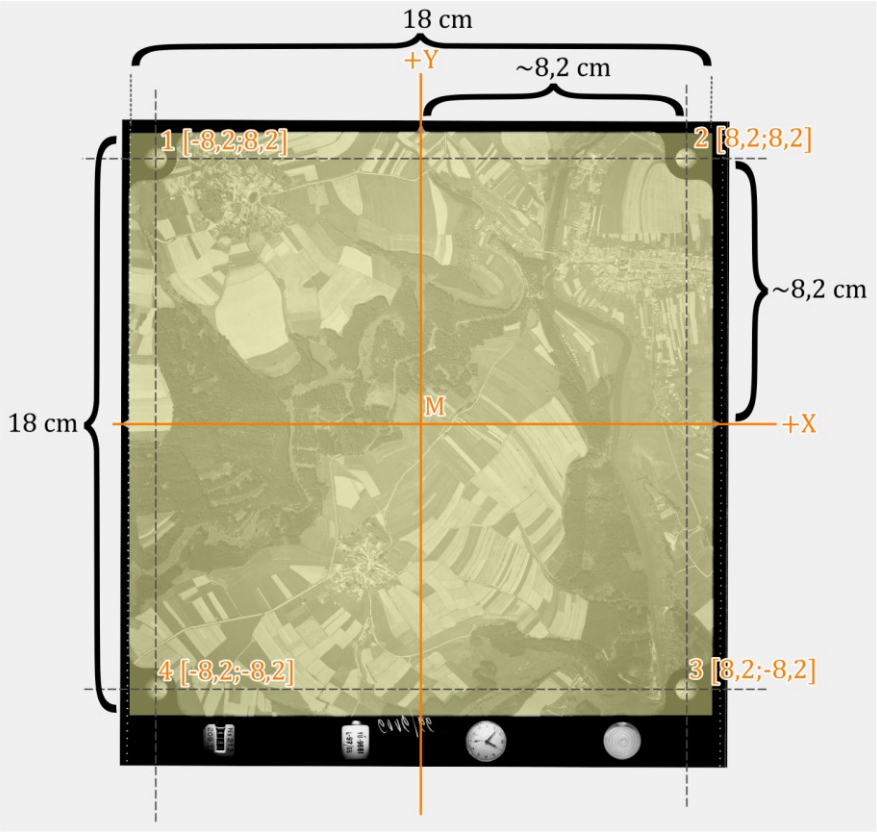


Zdroj: Digitální atlas zaniklých krajin (2020)

4.2 Použitá data

Základním datovým zdrojem této práce jsou archivní letecké měřické snímky (Obrázek 16), které poskytuje VGHMÚř. K snímkům nejsou k dispozici tzv. kalibrační protokoly obsahující prvky vnitřní orientace (IO). Přímo na snímku se však nachází údaj o konstantě kamery (Šafář, Tlapáková 2016). Snímky byly skenovány s rozlišením 0,015 mm. Jedná se o skeny analogových snímků ve formátu *.tif či *.jpg. Při objednání snímků přes ČÚZK je dodán také *.tfw/*.jgw World file obsahující georeferenci snímků. Navíc jsou také poskytovány přibližně určené prvky EO a stopy snímků v *.shp formátu. Tyto informace byly k dispozici pouze u snímků ze 30. let.

Obrázek 16: Ilustrační ukázka – letecký měřický snímek z roku 1966



Snímek byl standardních rozměrů 18 × 18 cm, rozměry se vztahují k samotné ploše snímku zvýrazněné na obrázku žlutým filtrem. Veškeré rozměry či souřadnice jsou uvedeny v centimetrech. Obrázek je jen pro ilustraci, v této práci figurují jiná zájmová území.

Zdroj: vlastní zpracování/podkladová data: Archiv leteckých měřických snímků (VGHMÚř Dobruška 2022)

Snímky z 80. let byly skenovány na objednávku speciálně pro účely této práce VGHMÚř v Dobrušce. Některé snímkové sady z 80. let mají špatnou kvalitu způsobenou kopírováním originálních negativů na jiný nehořlavý materiál (Šafář, Tlapáková 2016). VGHMÚř v Dobrušce však zvolil takové snímky a roky snímání, které nemají popsaný defekt a z hlediska ostrosti jsou kvalitní. Tyto snímky získané zvláštní objednávkou však neprošly zpracováním ČÚZK a nemají

georeferenci (nemají World files) či přibližně vypočtené prvky EO. Byly však překvapivě poskytnuty stopy snímků. Dodané *.shp se stopami bylo navíc v souřadnicovém systému WGS 84 UTM zone 33N (EPSG: 32633), ačkoliv státní správa běžně pracuje v S-JTSK (EPSG: 5514). Zároveň soubor neobsahoval stopy všech dodaných snímků. Snímky zapůjčil ČÚZK bezúplatně nad rámec běžné studentské licence, jelikož má zájem o výsledky práce.

Chybějící georeference nedodané ČÚZK u snímků z 80. let nakonec nečinily problém. Jak bylo později zjištěno, u snímků ze 30. let byly georeference poskytnuté ČÚZK velmi nepřesné (RMSE i přes 200 m, viz Tabulka 3 dále v metodice), což lze považovat za chybu v metodickém zpracování georeferencí. Zároveň byla pro georeferencování dle údajů v dodaných *.tfw souborech použita afinní transformace. Pro georeferencování LMS je vhodné použít transformaci projektivní, tento geometrický model však nejde uložit do World file souboru. Pavelka (2003a) uvádí, že v případě, kdy není projektivní transformace k dispozici, lze užít transformaci podobnostní, afinní, či polynomickou. Žádná z nich ale přesně nevystihuje středové promítání. Teoreticky by podobnostní tr. měla být vhodnější než afinní tr., jelikož afinní tr. zahrnuje i zkosení obrazu, které však není součástí geometrického modelu středového promítání. Z důvodu velké nepřesnosti musely být georeference stejně opraveny i u snímků z 30. let před automatizovaným vyhledáváním VLB. Vše je detailně vysvětleno dále v metodice práce. Snímky byly poskytnuty ČÚZK nad rámec běžné studentské licence, jelikož jich bylo třeba větší množství a zároveň má ČÚZK zájem o výsledky práce. Pro 30. léta bylo zvoleno vždy 6 snímků, zatímco pro 80. pouze 3. Snímky mají různé měřítko a cílem bylo pokrýt přibližně stejnou zájmovou oblast. Přehled vlastností snímků je uveden níže (Tabulka 1).

Tabulka 1: Vlastnosti použitých archivních snímků

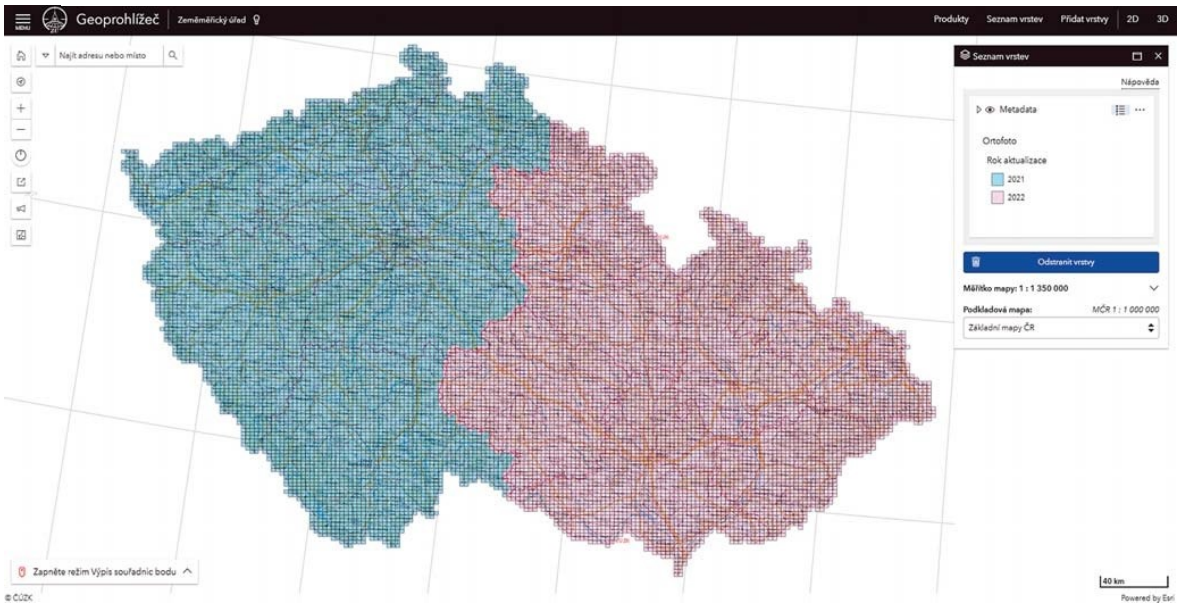
	Milovice 1938	*Milovice 1989*	*Kobylí 1938*	*Kobylí 1984*
Počet snímků	6	3	6	3
Formát snímků	*.tif	*.jpg	*.tif	*.jpg
Konstanta kamery [mm]	211,59	151,92	211,25	151,99
Počet rámových značek	4	4	4	4
Rozlišení skeneru [µm]	15	15	15	15
Georeference ČÚZK	Ano*	Ne	Ano*	Ne
Prvky EO	Ano**	Ne	Ano**	Ne
Stopy snímků	Ano	Ano***	Ano	Ano***

*Pozn.: * dodané georeference byly chybné a musely být před automatizovaným vyhledáváním VLB opraveny. ** Prvky EO dodané ČÚZK nebyly v práci vůbec použity. *** Dodané shapefily neobsahovaly stopy ke všem poskytnutým snímkům. Stopy musely být zvektorizovány později ručně.*

Zdroj: Vlastní zpracování/podkladová data VGHMÚř Dobruška (2022)

Kromě archivních snímků byla nutná data s přesnou georeferencí jakožto referenční podklad. Nejdůležitější bylo současné ortofoto ČR. Jedná se o produkt poskytovaný ČUZK ve formě rastru v kladu listů SMO-5 (2 × 2,5 km). Polohová chyba dosahuje hodnoty 0,2 metru. Ortofoto ČR je tvořeno od roku 2002 ČÚZK ve spolupráci s VGHMÚř a Ministerstvem obrany. Periodicita snímání jsou 2 roky (Obrázek 17), přičemž každý rok je nasnímána polovina státu. Na základě ortofota ČR jsou aktualizovány databáze topografických dat a státní mapové dílo. Slouží také jako podklad v nejrůznějších aplikacích, z nichž nejznámější je například Nahlížení do katastru nemovitostí. Ortofoto je možno používat pro vlastní aplikace díky existenci služeb WMS, WMTS a ArcGIS Server v souřadnicových systémech S-JTSK (EPSG: 5514) a Web Mercator (EPSG: 3857) pro kompatibilitu s mapovými dlaždicemi Google maps a dlaždicemi dalších poskytovatelů online mapových vrstev (ČÚZK 2023). Jelikož bylo ortofoto pro zájmové oblasti pořizováno v různých letech snímkování, bylo jeho prostorové rozlišení pro Kobylí 20 cm a pro Milovice 12,5 cm.

Obrázek 17: Rozdělení ČR na jednotlivé oblasti snímání



Zdroj: ČÚZK (2023)

Tvorba databáze ZABAGED začala v roce 1995 a od té doby je průběžně aktualizována v cyklech nejdéle 5 let na základě dat fotogrammetrických, topografickým terénním šetřením, získáváním informací od ostatních orgánů veřejné správy nebo i z dat leteckého laserového skenování povrchu. Z polohopisné složky databáze ZABAGED byly v práci použity vrstvy „SilniceDalnice“, „Ulice“ a „Cesta“. Z výškopisné složky ZABAGED byl použit DMP 1G pro zjištění nadmořských výšek potenciálních VLB. DMP 1G zobrazuje zemský povrch včetně rostlinného krytu, staveb a

dalších objektů na zemském povrchu. Střední chyba výšky dosahuje 0,4 m pro přesně vymezené objekty (budovy) a 0,7 m pro neohraničené objekty (lesy). DMP 1G je poskytován ve formě nepravidelné sítě výškových bodů (TIN). Pro účely této práce byl použit rastrový model s velikostí pixelu 2 m. Aktualizace DMP 1G bude probíhat metodou leteckého laserového skenování povrchu a metodami obrazové korelace leteckých měřických snímků. Ortofoto ČR, databáze ZABAGED a DMP 1G jsou od léta roku 2023 poskytovány v režimu otevřených dat (ČÚZK 2023).

V neposlední řadě bylo použito ortofoto Cenia zobrazující území ČR v 50. letech minulého století. Jedná se o ortofoto vytvořené firmou Geodis Brno v roce 2010 pro agenturu Cenia. Původním účelem bylo zachycení stavu krajiny ČR před kolektivizací a scelováním polí v 50. letech. Výstup vznikl v rámci projektu Národní inventarizace kontaminovaných míst. Nicméně pro celé území nebyly dostupné snímky z počátku 50. let, proto byla data doplněna o snímky ze 30. let nebo naopak o snímky z konce 50., či dokonce ze 60. i 70. let (Cenia 2022). Při tvorbě nebyly dostupné prvky IO ani EO kromě konstanty kamery. Skenování bylo provedeno v rozlišení 14 μm s geometrickou přesností 2 μm. V současnosti je ortofoto poskytováno zdarma agenturou Cenia pod otevřenou licencí v kladu listů SMO-5 v S-JTSK (EPSG: 5514) s prostorovým rozlišením 0,5 m ve formátu *.jpg s přesností cca 1,5 m pro poválečné a 1,3 m pro předválečné snímky. Zájmová území v této práci by měla být pokryta snímky z let 1953 a 1954 (Sukup 2010). Využít lze taky WMS, WMTS webové služby. Do budoucna je přislíbena existence webových služeb ve Web Mercatorovi (EPSG: 3857) (Cenia 2022). Pro vybraná místa ČR existuje ortofoto ze 30. let s prostorovým rozlišením 1 m a polohovou přesností 3 m, nicméně zájmová území této práce nejsou zahrnuta (Cenia 2020).

4.3 Volba operátoru pro detekci a párování identických bodů

Stěžejním úkolem pro sestavení funkční metodiky pro automatizovaný sběr VLB byl výběr vhodného operátoru z oblasti počítačového vidění. Požadavky na operátor jsou popsány v kapitole „Cíle práce a hypotézy“ v úvodu práce. Jelikož bylo žádoucí, aby metodika fungovala ve všech zájmových územích, pro testování různých operátorů byl zvolen snímek z oblasti Milovic z roku 1938 zobrazující zástavbu i rozsáhlé zemědělské plochy. U těchto snímků jsou totiž největší změny v krajině, které komplikují vyhledávání VLB. Snímek i referenční ortofoto byly převzorkovány na prostorové rozlišení 2 m. Následovalo testování samotných operátorů v různých scénářích použití (Tabulka 2), jmenovitě se jednalo o SIFT, SURF, ORB a obrazovou korelaci.

Tabulka 2: Varianty metodických přístupů

	Operátor – varianta	*Testovaný metodický přístup*
1	*SIFT – D*	Operátorem z oblasti počítačového vidění byly vyhledány a popsány *(detekce a deskripce)* význačné body v archivním LMS, následně byla vytyčena přibližná oblast k prohledávání v současném ortofotu. Tato oblast byla následně procházena pixel po pixelu a v každém pixelu byl vypočítán SIFT *(pouze deskripce)*. Pro každou oblast byla vyexportována část bodů s nejmenší euklidovskou vzdáleností mezi SIFT vektory (největší pravděpodobnost spárování s referenčním bodem – *matching*).
2	*SIFT – DD*	Operátorem byly vyhledány a popsány *(detekce a deskripce)* význačné body v archivním LMS, následně byla vytyčena přibližná oblast k prohledávání v současném ortofotu. V této oblasti byly operátorem také nalezeny a popsány význačné body *(detekce a deskripce)*. Následovalo spojování bodů (*matching*) stejně jako u předchozí varianty. Jedná se o typický způsob užití SIFT operátoru.
3	*SIFT – VD*	Do metodiky byla zapojena vektorová data – křižovatky silnic reprezentované body. Pro každou křižovatku byla ze současného ortofota extrahována odpovídající malá ploška rastru a byla popsána operátorem *(pouze deskripce)*. Následovalo vytyčení přibližné oblasti k prohledávání v archivním LMS a jejím procházení pixel po pixelu *(pouze deskripce)*. Zbytek metodiky je shodný s první variantou.
4	*SIFT – VDD*	Opět byly použity křižovatky silnic, avšak v oblasti k prohledávání
5	*SURF*	v archivním LMS byly detekovány a popsány význačné body *(detekce a*
6	*ORB*	*deskripce)*.
7	*Obrazová korelace*	Obdobné se 3. variantou, jenom u obrazové korelace se nehovoří o fázi detekce ani deskripce. Přibližně vytyčená oblast v archivním snímku byla procházena malou ploškou křižovatky a pro každý pixel byl počítán korelační koeficient. Pixely s nejvyšším korelačním koeficientem (*matching*) byly vyexportovány.

Zdroj: vlastní tvorba

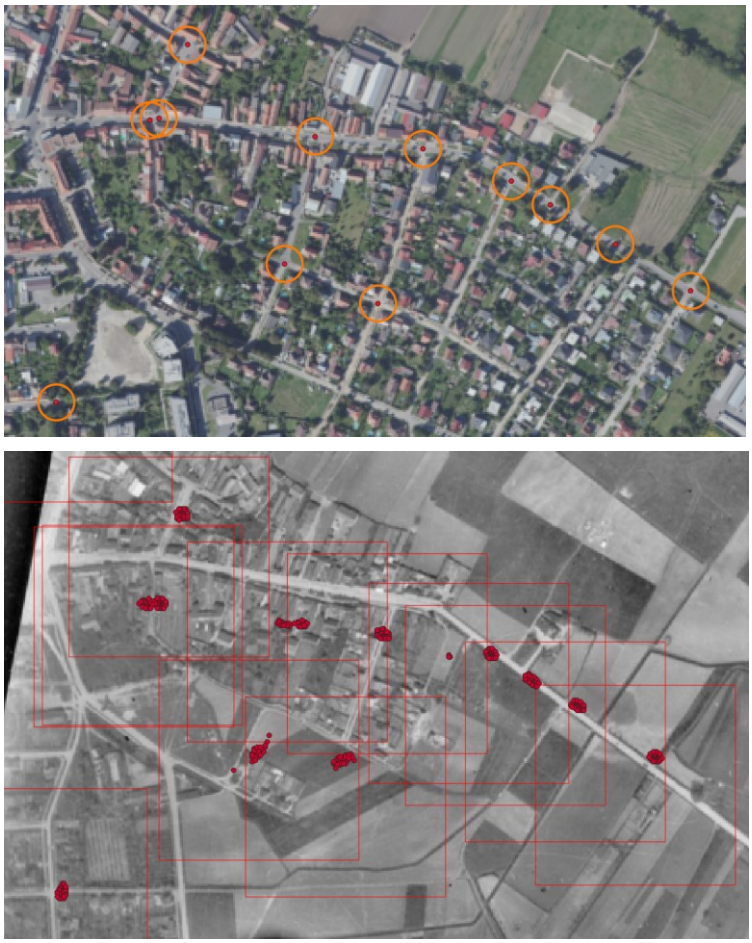
Testování těchto 7 metodických variant (Tabulka 2) bylo věnováno mnoho času a úsilí. Několikrát byly zkontrolovány veškeré vytvořené skripty, zda neobsahují implementační chybu. Byly provedeny i nejrůznější modifikace rozličných parametrů operátorů u jednotlivých metodických postupů s cílem dosáhnout co možná nejvyšší úspěšnosti při párování bodů. Bohužel výsledky byly velmi špatné a při celém naprosto frustrujícím testování nebylo dosaženo výraznějších úspěchů. SIFT, který je jinak užívaný pro automatizované vyhledávání VLB pro současné snímky (např. Oh, Toth, Grejner-Brzezinska (2010)) naprosto selhával a ani jedna z variant založená na SIFT operátoru se nedala označit za úspěšnou. Podobně lze zhodnotit i operátory SURF a ORB, které dosahovaly ještě horších výsledků. U drtivé většiny potenciálních VLB bylo párování bodů mezi archivním LMS a referenčním podkladem chybné. Obzvlášť mimo zástavbu byly v podstatě všechny body chybně spárované, případně dokonce nespárovatelné. Zde se velmi negativně na úspěšnosti podepsalo scelování polí. Z testovaných operátorů si vedla nejlépe obrazová korelace, což bylo poměrně překvapující, jelikož se jedná o nejstarší a nejjednodušší algoritmus ve srovnání se zbylými operátory. Z těchto důvodů byla právě obrazová korelace zvolena pro automatizovaný postup prezentovaný dále v metodice, přestože ani v tomto případě nebyly výsledky vůbec ideální. Níže jsou uvedeny ukázky z testování jednotlivých variant (Obrázek 18).

Obrázek 18: Ukázky z testování operátorů

Označení variant viz Tabulka 2. Varianta 1: červeným ohraničením je zvýrazněna tatáž oblast v referenčním obraze i archivním LMS. Je použita původní georeference obrazů, která se samozřejmě liší. Vlevo bod v archivním LMS označený SHIFTem jako význačný, vpravo body, které by mu měly odpovídat. Cílová střecha

není označena ani jedním bodem, byly však nalezeny střechy se stejnou orientací. Žlutě jsou označeny body s nejmenší euklidovskou vzdáleností (největší pravděpodobnost párování).

Varianta 2: stejný bod jako v předchozím případě, tentokrát byla cílová střecha nalezena, avšak spárován by byl bod označen šipkou, další body s nejmenší vzdáleností příznaku jsou žlutě zvýrazněné, cílový bod mezi nimi není.



Varianta 3: horní obrázek – kružnicemi je vyznačena oblast (křižovatka) v současném ortofotu nad kterou je počítán SIFT deskriptor (průměr kružnice: 40 m). Dolní obrázek – podobně jako u obrazové korelace i při

použití operátoru SIFT docházelo při procházení oblasti pixel po pixelu ke vzniku shluků. Pro každou křižovatku je vyznačena příslušná oblast k prohledávání (červený čtverec). Výsledek je velmi neideální.

Varianta 7: použity stejné křižovatky jako v předchozím případě (varianta 3). Je evidentní, že úspěšnost nalezení křižovatek s pomocí obrazové korelace je oproti SIFT operátoru (varianta 3) výrazně vyšší.

Zdroj: vlastní tvorba/podkladová data VGHMÚř Dobruška (2022), ČÚZK (2022a)

4.4 Předzpracování dat pro automatizované vyhledávání VLB

Metodický postup s obrazovou korelací vyžadoval extrakci křižovatek komunikací. Zatímco pro testování a porovnání operátorů v předchozí kapitole byly křižovatky vektorizovány ručně, ve výsledné metodice byl export křižovatek zcela automatický. Za tímto účelem musela být předzpracována data ZABAGED. Vrstva „SilniceDalnice“ je tvořena liniemi reprezentujícími silnice, které v místě křížení s vrstvou „Ulice“ a „Cesta“ nejsou přerušeny, jedná se o souvislé linie. Z tohoto důvodu tak ve výchozím stavu nelze na dané vrstvě identifikovat křižovatku. Nejprve proto bylo nutné na vrstvě „SilniceDalnice“ křižovatky vytvořit (Obrázek 19). K tomu posloužila funkce implementovaná v QGISu `Split with lines`, která vstupní liniovou vrstvu („SilniceDalnice“) rozdělí v místě křížení s jinou liniovou vrstvou („Cesta“, „Ulice“). Vrstva vzniklá tímto postupem byla případně ještě spojena s vrstvami „Cesta“ a „Ulice“ s pomocí funkce `Merge vector layers`, pokud to bylo pro zpracování konkrétní sady snímků třeba. Pro některé sady byly totiž použity pouze „SilniceDalnice“, pro jiné pak navíc i vrstvy „Cesta“ a „Ulice“. Konkrétní soupis parametrů vstupující do automatizovaného postupu vyhledávání VLB je uveden na konci následující kapitoly (Tabulka 6). Jelikož byly v práci třeba 3D VLB, byl použit DMP 1G ve formě rastru s prostorovým rozlišením 2 metry pro obě zájmové oblasti.

Obrázek 19: „SilniceDalnice“ po aplikaci funkce Split with lines

Horní obrázek: před aplikací funkce byly silnice vrstvy „SilniceDalnice“ celistvé linie v místě napojení ulic. Nešlo proto na zmíněné vrstvě identifikovat křižovatky. Spodní obrázek: po aplikaci funkce byly silnice z vrstvy „SilniceDalnice“ rozsekány na kusy pro umožnění detekce křižovatek. Konce linií jsou zvýrazněny červeně.

Zdroj: vlastní tvorba/podkladová data ČÚZK (2023), ČÚZK (2022a)

Při zpracování archivních LMS bylo bohužel zjištěno, že snímky z 30. let mají metodicky chybně zpracovanou georeferenci. Georeference těchto snímků poskytnuté ČÚZK byly velmi nepřesné (RMSE i přes 200 m, viz Tabulka 3 dále v metodice), což lze považovat za chybu v metodickém zpracování georeferencí. Zároveň byla pro georeferencování dle údajů v dodaných *.tfw souborech použita afinní transformace. Pro georeferencování LMS je vhodné použít transformaci projektivní, tento geometrický model však nejde uložit do World file souborů distribuovaných se snímky. Pavelka (2003a) uvádí, že v případě, kdy není projektivní

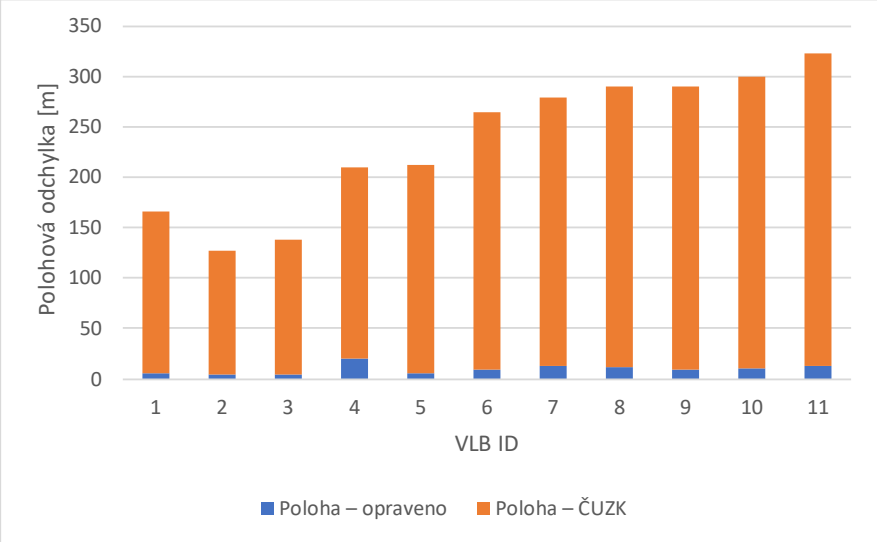
transformace k dispozici, lze užít transformaci podobnostní, afinní, či polynomickou. Žádná z nich ale přesně nevystihuje středové promítání. Teoreticky by podobnostní tr. měla být vhodnější než afinní tr., jelikož afinní tr. zahrnuje i zkosení obrazu (různé koeficienty zvětšení v ose X a Y), které však není součástí geometrického modelu středového promítání. Z důvodu velké nepřesnosti musely být dodané georeference stejně opraveny u snímků z 30. let před automatizovaným vyhledáváním VLB. Hlavním problémem byla evidentně přesnost použitých VLB. Toto bylo možné ověřit v zájmovém území Milovic, které je rovinné a použitá transformace by proto neměla mít takový vliv na přesnost výsledku. Přesto se po vytvoření nových georeferencí i v tomto případě zlepšila přesnost o desítky metrů. Veškeré testování operátorů v předchozí kapitole bylo bohužel provedeno se špatnými georeferencemi, což pravděpodobně přispělo k fatálnímu selhání při testování většiny operátorů. Přesto je testování alespoň částečně relevantní, jelikož je možné porovnat výsledky jednotlivých variant a operátorů mezi sebou. Pro kvantifikaci problematiky georeferencí bylo v oblasti pokryté snímkem LMSA08.1938.HUST25.00859.tif (Kobylí 1938) rozmístěno 11 kontrolních bodů, následně mohla být spočítána RMSE (Tabulka 3). Poté byla ještě ilustrována polohová odchylka jednotlivých kontrolních bodů (Graf 2, Obrázek 20). V případě tohoto snímku se jedná záměrně o extrémní případ, nicméně i u ostatních LMS byla polohová odchylka u opravených georeferencí výrazně lepší, typicky o vyšší desítky metrů.

Tabulka 3: Porovnání přesnosti georeferencí

	X [m] RMSE	Y [m] RMSE
ČÚZK georeference	213,27	99,97
Opravené georeference	6,83	8,08

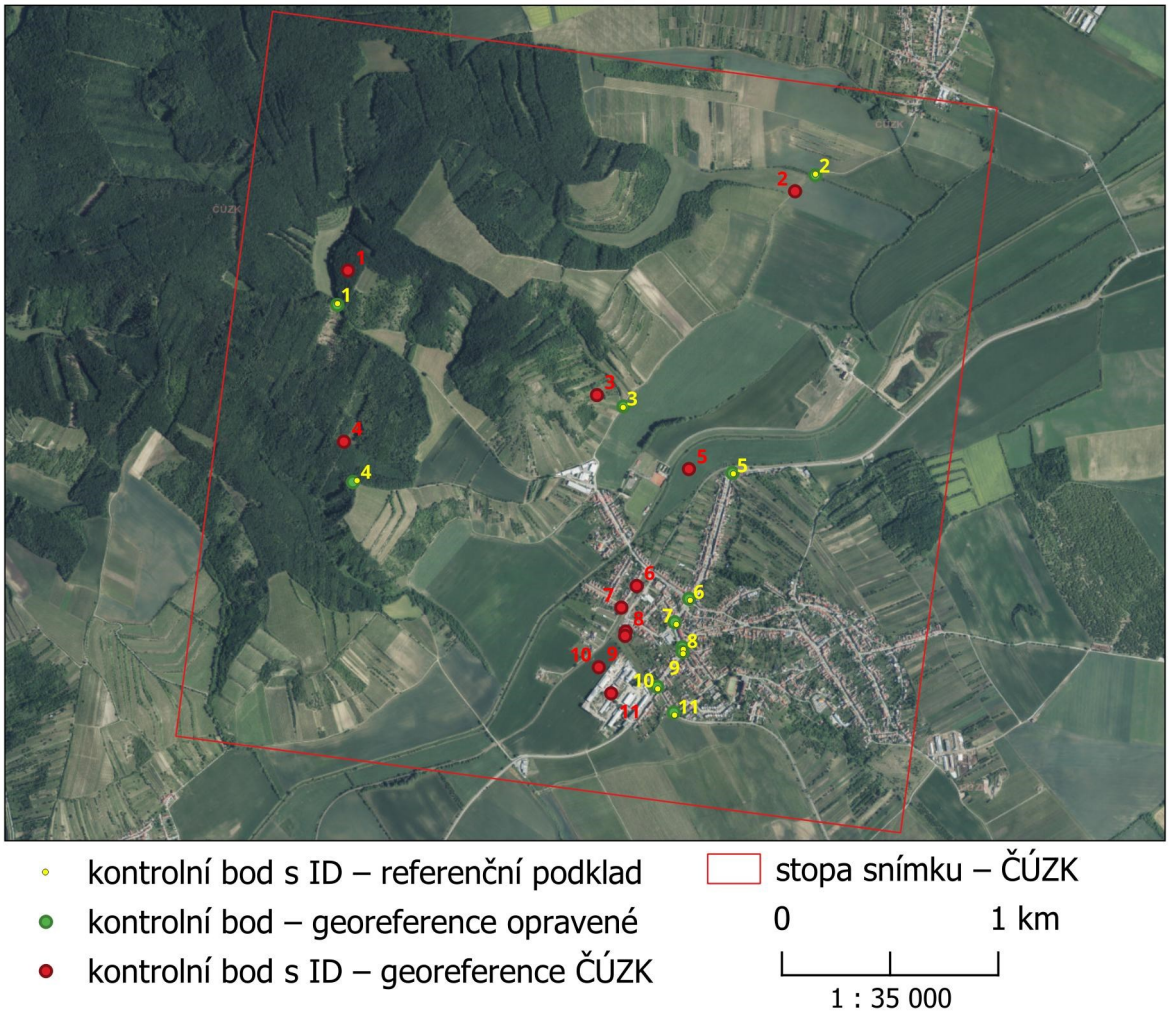
Zdroj: vlastní tvorba

Graf 2: Polohové odchylky kontrolních bodů



Zdroj: vlastní tvorba

Obrázek 20: Vizualizace přesnosti georeferencí



Zdroj: vlastní tvorba/podkladová data ČÚZK (2022a)

Jednalo se o zcela zásadní poznatek, oprava georeferencí velmi pozitivně ovlivnila přesnost automaticky nalezených VLB pro tvorbu ortofota, jelikož se výrazně zmenšila oblast k prohledávání (Obrázek 20). Díky tomu došlo taky k výrazné úspoře výpočetního času. U testovacích snímků s georeferencí ČÚZK z oblasti Milovic nebylo možno odfiltrovat špatně spárované VLB ani při použití metody RANSAC dále v metodice. Po opravě georeferencí byl postup úspěšný. Sběr vlastních VLB proběhl v SW QGIS 3.30 v modulu „Layers – georeferencer". Jako referenční podklad posloužilo současné ortofoto, pro každý LMS bylo zvoleno cca 3 až 5 VLB. Pro uložení sesbíraných VLB do souboru *.points je nutné v nastavení „Transformation settings" zatrhnout možnost „Save GCP Points". Soubory *.points jsou textové soubory obsahující rastrové i referenční geodetické souřadnice VLB a další informace. Pro jejich vygenerování je nutné spustit georeferencování (Start Georeferencing), což vygeneruje nový rastrový obraz a právě *.points soubor. Vygenerované převzorkované rastry nebyly dále použity. Místo toho byl naprogramován skript `0step_georeference2worldfile.py`, který ve zdrojovém adresáři nalezne veškeré soubory *.points. Dále jsou ve skriptu vyextrahovány souřadnice VLB, z nichž jsou pomocí podobnostní (Helmertovy) transformace s vyrovnáním metodou nejmenších čtverců vypočítány jednotlivé transformační koeficienty. Tyto koeficienty jsou poté uloženy do příslušných World file souborů. Uvedené řešení umožňuje pracovat s původními kvalitními snímky skenovanými VGHMÚř v Dobrušce se zachovanými prvky IO ve snímku a zároveň se vyhnout vzniku nového rastru a s ním spojeného převzorkování, které má negativní vliv na kvalitu obrazu. Tímto způsobem byla vytvořena georeference pro všechny snímky ze všech datových sad.

4.5 Automatizované vyhledávání VLB

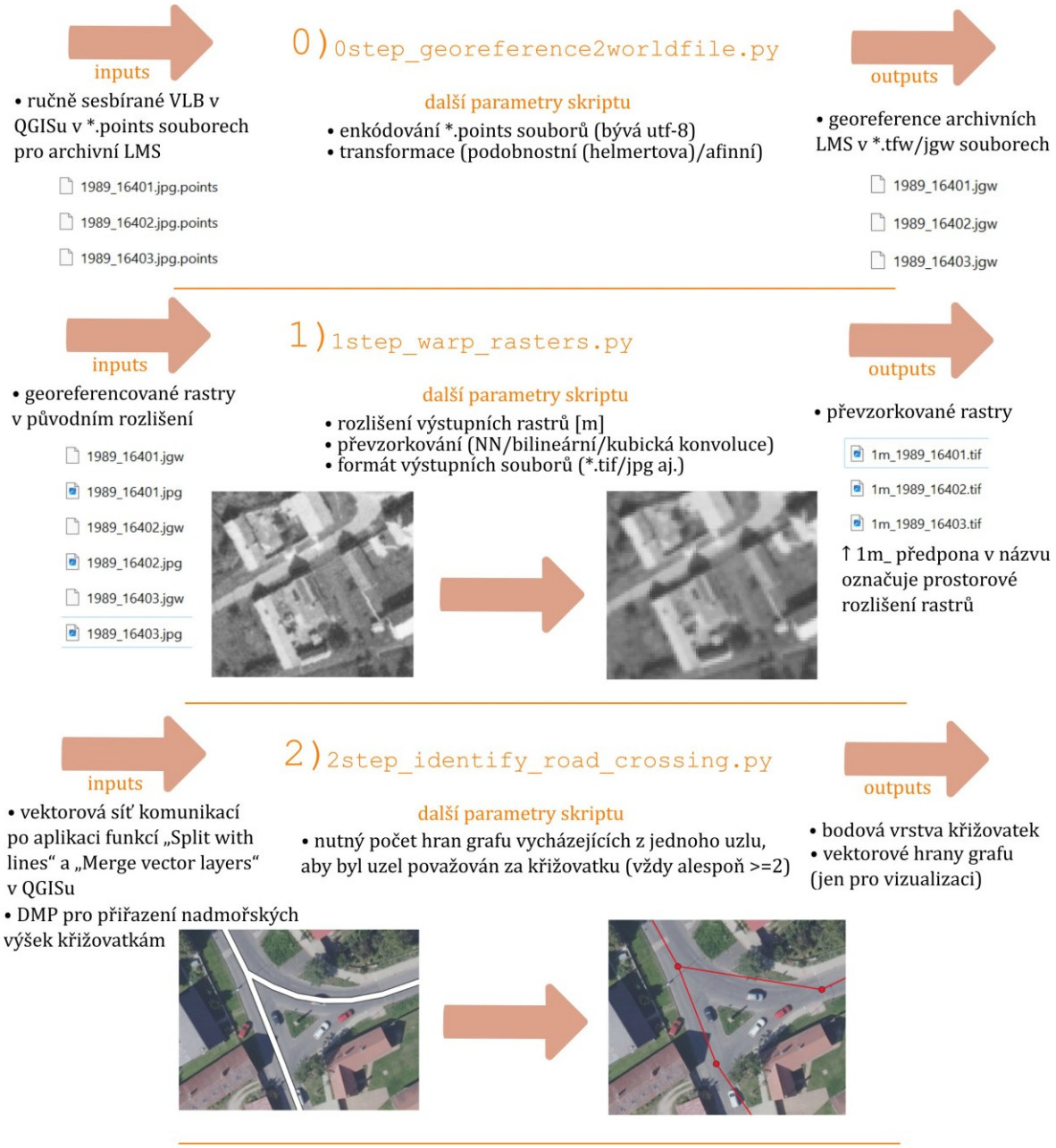
Za účelem automatizovaného vyhledávání a párování bodů byla vytvořena ucelená sada skriptů v jazyce Python 3.10 využívající knihovny (Obrázek 21) OpenCV obsahující algoritmy počítačového vidění, Numpy pro práci s maticemi i vektory, Rasterio a GDAL (OSGeo projekt) pro zpracování rastrů, Matplotlib pro tvorbu grafů a dalších obrazových výstupů, Geopandas, Pandas, Shapely pro zpracování vektorových dat, NetworkX pro práci s grafy a případně ještě další knihovny. Jelikož metodika dlouho produkovala značně neuspokojivé výsledky, bylo žádoucí mezivýstupy skriptů kontrolovat, aby byla případně odhalena implementační či jiná chyba.

Obrázek 21: Loga použitých knihoven

Zdroj: vlastní tvorba/webové stránky příslušných knihoven

Na dalších stranách je diagramy popsána celá metodika (Obrázek 22). Následuje samotný textový popis.

Obrázek 22: Diagram – jednotlivé kroky metodiky



inputs

- listy referenčního ortofota
- bodová vrstva křižovatek

3) 3step_intersect_road_crossing.py

outputs

- bodová vrstva křižovatek → intersect s bounding boxy listů referenčního ortofota
- polygonová vrstva → bounding boxy listů referenčního ortofota

další parametry skriptu

- souřadnicový systém všech vrstev (EPSG 5514 v této práci)

inputs

- bodová vrstva křižovatek → intersect s bounding boxy listů referenčního ortofota
- listy referenčního ortofota

4) 4step_extract_patches_offline.py

outputs

- rastrové plošky (křižovatky) → 51 m × 51 m pro všechny sady
- ohraničení rastrových plošek → vektorová data – pouze pro vizualizaci

50.tfw
50.tif
51.tfw
51.tif
52.tfw
52.tif
53.tfw
53.tif
54.tfw
54.tif

další parametry skriptu

- velikost plošky (křižovatky) [m] → 51 m pro všechny datové sady
- rozlišení rastrů [m] → 1 m pro všechny datové sady

inputs

- bodová vrstva křižovatek → intersect s bounding boxy listů referenčního ortofota
- rastrové plošky (křižovatky)
- archivní LMS
- stopy archivních LMS

5) 5step_correl_patches.py

outputs

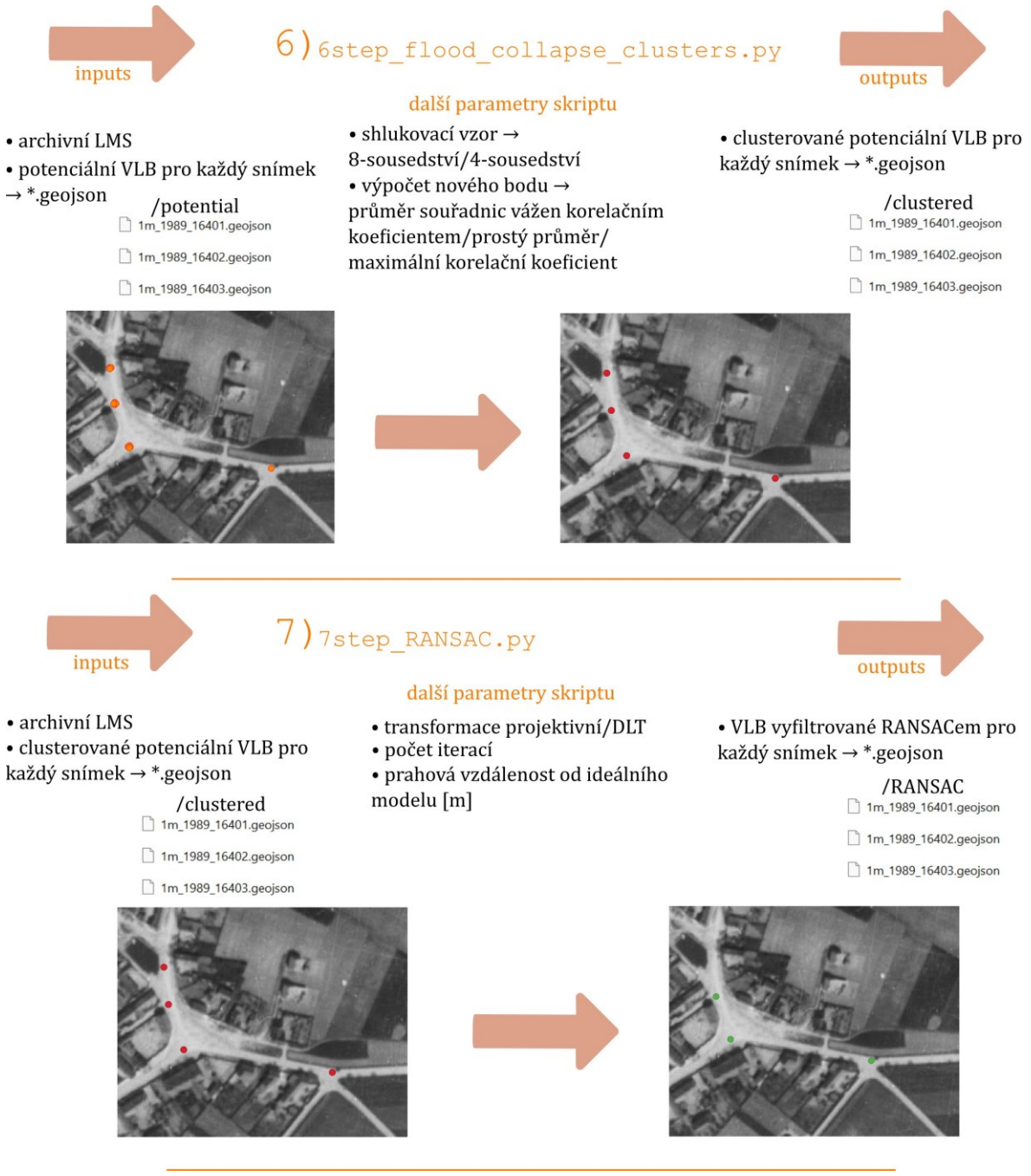
- polygony → ohraničení prohledávané oblasti pro každou křižovatku – pouze pro vizualizaci
- potenciální VLB pro každý snímek → *.geojson

1m_1989_16401.geojson
1m_1989_16402.geojson
1m_1989_16403.geojson

další parametry skriptu

- pásmo referenčního ortofota → zelené/RGB2grey
- velikost prohledávané oblasti [m]
- počet vyexportovaných VLB pro každou křižovatku
- limit korelačního koeficientu

- v ukázce je počet exportovaných bodů nastaven na 5, nicméně mnoho VLB neprošlo přes limit korelačního koeficientu, proto některé křižovatky nemají žádný odpovídající VLB nebo jen 5> VLB.



Kompletní dokumentace ke skriptům je v příloze na konci práce, tento schématický popis je pouze zkrácený.

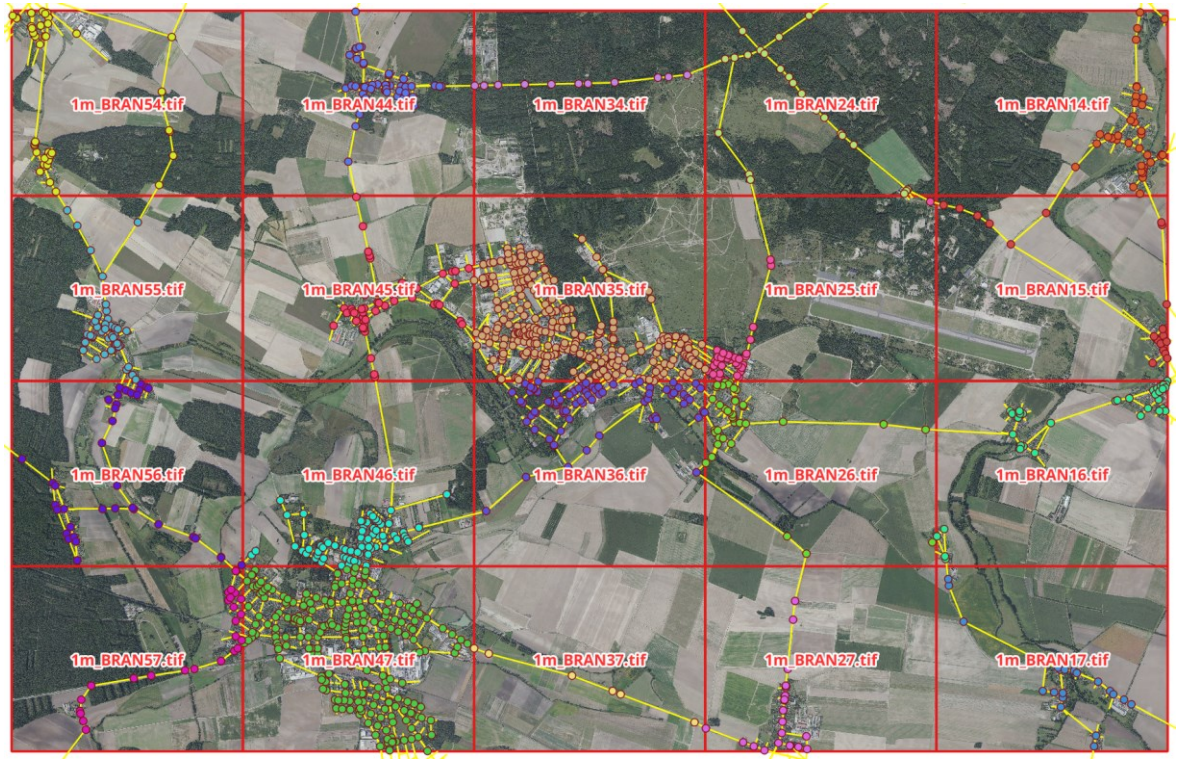
Zdroj: vlastní tvorba/ podkladová data ČÚZK (2023), ČÚZK (2022a), VGHMÚř Dobruška (2022), ČÚZK (2022b)

Pro automatickou extrakci křižovatek z vektorové vrstvy komunikací (výstup z funkce `Split with lines`, případně `Merge vector layers`) byl vytvořen skript `1step_identify_road_crossing.py`, jehož vstupem je vektorová vrstva komunikací a rastrový DMP 1G. Nejprve byl definován graf, do kterého byly přidány hrany, každá hrana byla tvořena uzly – počátečním a koncovým bodem komunikací. Takto byla grafem reprezentována celá síť komunikací. Následně bylo iterováno přes uzly grafu a zkoumal se počet hran, které vycházely z daného uzlu. Uzel, ze kterého vycházely 2 a více hran (pokud byly vstupními liniemi pouze rozsegmentované „SilniceDalnice"), případně 3 a více hran (pokud byla vstupem vrstva z funkce `Merge vector layers`) byl vyexportován. Pro export křižovatek tak byl využit stupeň vrcholu grafu. Těmto výstupním bodům reprezentujícím křižovatky byla ještě přiřazena nadmořská výška z DMP 1G a výsledná bodová vrstva byla uložena do souboru *.geojson. Pro zajímavost a vizualizaci byly uloženy také hrany použitého grafu (Obrázek 23, Obrázek 28).

Jelikož bylo třeba poměrně mnoho rastrů (současné ortofoto, ortofoto Cenia z 50. let, archivní LMS) převzorkovat do stejného prostorového rozlišení kvůli obrazové korelaci, byl naprogramován skript `2step_warp_rasters.py`, který převede veškeré georeferencované rastry ze zdrojového adresáře do cílového prostorového rozlišení se zvoleným typem převzorkování i formátem a uloží je do zvoleného výstupního adresáře. Pro zpracování všech čtyř datových sad byly použity rastry s prostorovým rozlišením 1 m. Tento skript by mohl být nahrazen funkcí Warp (reproject) v QGISu, pokud by byla spuštěna dávkově (Execute as Batch process), nicméně pro ucelený metodický postup v jazyce Python byl i tento krok realizován tímto jednoduchým skriptem.

Dále byl vytvořen skript `3step_intersect_road_crossing.py`, který má na vstupu složku obsahující referenční podkladová data (jednotlivé rastry ortofota v kladu SMO-5) a bodovou vrstvu křižovatek (výstup skriptu `1step_identify_road_crossing.py`). Skript postupně otevře veškeré listy ortofota a do vektorové vrstvy uloží polygony – ohraničující obdélníky (bounding boxy) rastrů, přičemž do atributu je uložena cesta ke každému listu. Následně je proveden intersect mezi těmito obdélníky a vrstvou křižovatek. Díky tomuto kroku tak bylo známo, ve kterém listu ortofota se nachází každá křižovatka (Obrázek 23).

Obrázek 23: Ukázka intersectu s bounding boxy listů ortofota v Milovicích a Lysé n./L.



V obrázku jsou zobrazeny hrany grafu, bounding boxy listů ortofota a křižovatky příslušícím jednotlivým listům ortofota.

Zdroj: vlastní zpracování/podkladová data ČÚZK (2023), ČÚZK (2022a)

Následuje skript `4step_extract_patches_offline.py`, jehož vstupem je bodová vrstva křižovatek z předchozího kroku, dále pak požadovaný rozměr plošky (hrana čtverce) reprezentující křižovatku. Nejprve byla otevřena veškerá ortofota, k nimž byla známá cesta díky údaji v atributu bodové vrstvy křižovatek. Následně byly procházeny jednotlivé křižovatky a pro každou z nich byla z listu příslušného ortofota vyextrahována rastrová ploška (Obrázek 24) o zadaném rozměru s lichým počtem řádků a sloupců. Georeference plošek byla vytvořena s pomocí World file souborů *.tfw. Velikost plošky byla pro všechny čtyři datové sady 51 m.

Obrázek 24: Ukázka vyexportovaných plošek křižovatek v Milovicích

V obrázku jsou zobrazeny i body reprezentující křižovatky získané zpracováním dat ZABAGED

Zdroj: podkladová data ČÚZK (2023), ČÚZK (2022a)

Nosnou částí metodiky je vyhledávání odpovídajících si bodů mezi referenčním ortofotem a archivními LMS. Tato úloha byla realizována skriptem `5step_correl_patches.py`. Vstupem je adresář obsahující rastrové plošky křižovatek, bodová vrstva křižovatek, adresář s archivními LMS, vrstva obsahující ohraničující polygony (bounding boxy) poskytnutých LMS (dodává ČÚZK). U LMS pro 80. léta musely být některé polygony zvektorizovány ručně, jelikož data ČÚZK nebyla kompletní. Parametry skriptu jsou dále údaje o použitém pásmu referenčního ortofota (zelené pásmo/šedotónový obraz vzniklý průměrováním všech 3 pásem), velikost prohledávané oblasti v metrech, počet vyexportovaných bodů s nejvyšším korelačním koeficientem pro každou křižovatku, hraniční limit korelačního koeficientu a hraniční vzdálenost od okraje snímku. Tento poslední parametr vycházel z předpokladu, že potenciální VLB v těsné blízkosti okraje snímku nemusí být dostatečně kvalitní. Pro každý archivní LMS byl otevřen příslušný bounding box a byl proveden intersect mezi tímto bounding boxem a bodovou vrstvou křižovatek. Tímto bylo zjištěno, které křižovatky se v daném snímku nachází.

Díky přibližné georeferenci LMS mohla být stanovena oblast k prohledávání pro všechny příslušné křižovatky. Velikost oblasti byla nastavena parametrem na vstupu skriptu. Tyto oblasti byly procházeny rastrovými ploškami (křižovatkami) a nad každým pixelem byl počítán korelační koeficient. Pixely (body), u kterých byl koeficient pod hraničním limitem, nebyly dále zpracovávány. Stávalo se tak, že pro některou křižovatku nebyl vyexportován žádný potenciální VLB, jelikož byly všechny odmítnuty pro nízký korelační koeficient. Pixely (body) s nejvyšším

korelačním koeficientem pro danou křižovatku byly uloženy do nové vrstvy *.geojson včetně korelačního koeficientu a dalších údajů (viz níže). Pro každou křižovatku bylo uloženo maximálně 5 bodů s nejvyšším korelačním koeficientem. Množství bodů lze jinak ovlivnit parametrem na vstupu skriptu. Nakonec byly oříznuty VLB, které byly blíže okraji LMS než byla hraniční vzdálenost od okraje snímku, což byl jeden z parametrů skriptu. Tímto postupem pro každý LMS vznikla vektorová vrstva potenciálních VLB, které měly v atributech uloženy údaje o nadmořské výšce, kvalitě (hodnota korelačního koeficientu), referenční geodetické X a Y souřadnice příslušné křižovatky, rastrové souřadnice řádek, sloupec namapované do převzorkovaného archivního LMS a identifikační ID původní křižovatky. Jelikož úspěšnost tohoto kroku je naprosto stěžejní pro celou metodiku, byly vyzkoušeny nejrůznější kombinace vstupů (Tabulka 4), aby mohla být zvolena nejideálnější kombinace parametrů pro finální řešení.

Tabulka 4: Porovnání variant s odlišnými vstupy pro skript obrazové korelace – 80. léta

varianta	*1*	*2*	*3*	*4*	*5*
referenční ortofoto	současné	50. léta	současné	současné	současné
převzorkování ref. orto.	bilineární	bilineární	bilineární	bilineární	bilineární
rozlišení rastrů [m]	1	1	0,5	0,35	0,35
velikost plošky [m]	51	51	51	51	51
*pásmo ref. ortofota**	RGB2grey	panchro.	RGB2grey	zelené	RGB2grey
korel. koef. – průměr	0,521	0,438	0,483	0,472	0,480
X RMSE [m]	4,18	11,07	4,24	4,18	4,08
Y RMSE [m]	5,40	11,80	5,39	5,38	5,40
průměr XY RMSE [m]	4,79	11,43	4,81	4,78	4,74

varianta	*6*	*7*	*8*	*9*
referenční ortofoto	50. léta	současné	současné	současné
převzorkování ref. orto.	původní rozlišení	bilineární	bilineární	nejbližší soused
rozlišení rastrů [m]	0,5	0,35	0,35	0,35
velikost plošky [m]	51	41	61	51
*pásmo ref. ortofota**	panchro.	RGB2grey	RGB2grey	RGB2grey
korel. koef. – průměr	0,428	0,526	0,396	0,463
X RMSE [m]	10,66	6,76	4,78	4,08

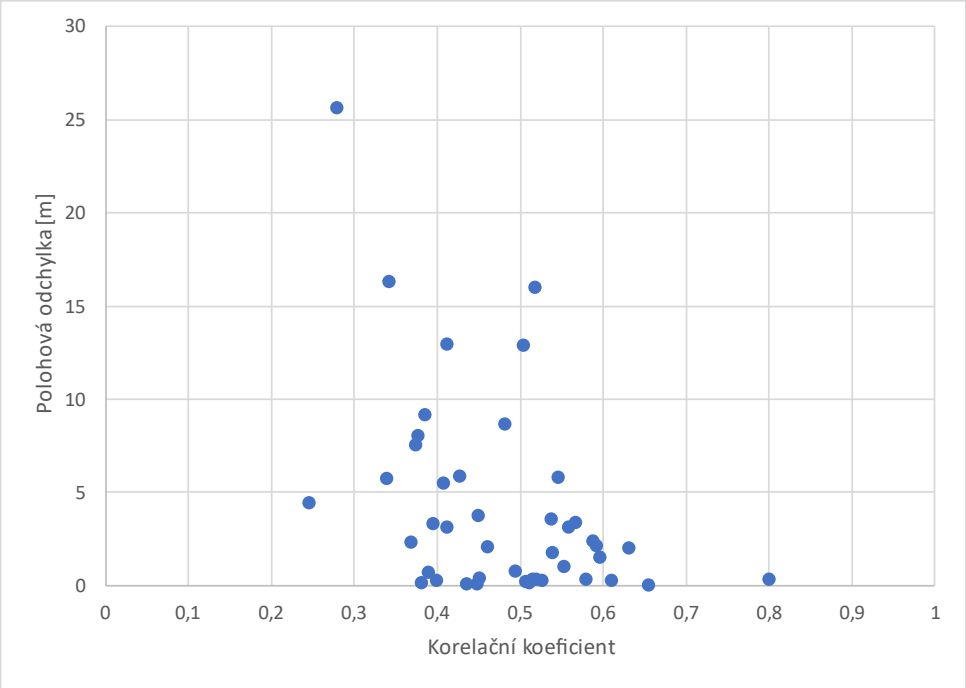
Y RMSE [m]	10,74	6,67	5,33	5,40
průměr XY RMSE [m]	10,70	6,71	5,06	4,74

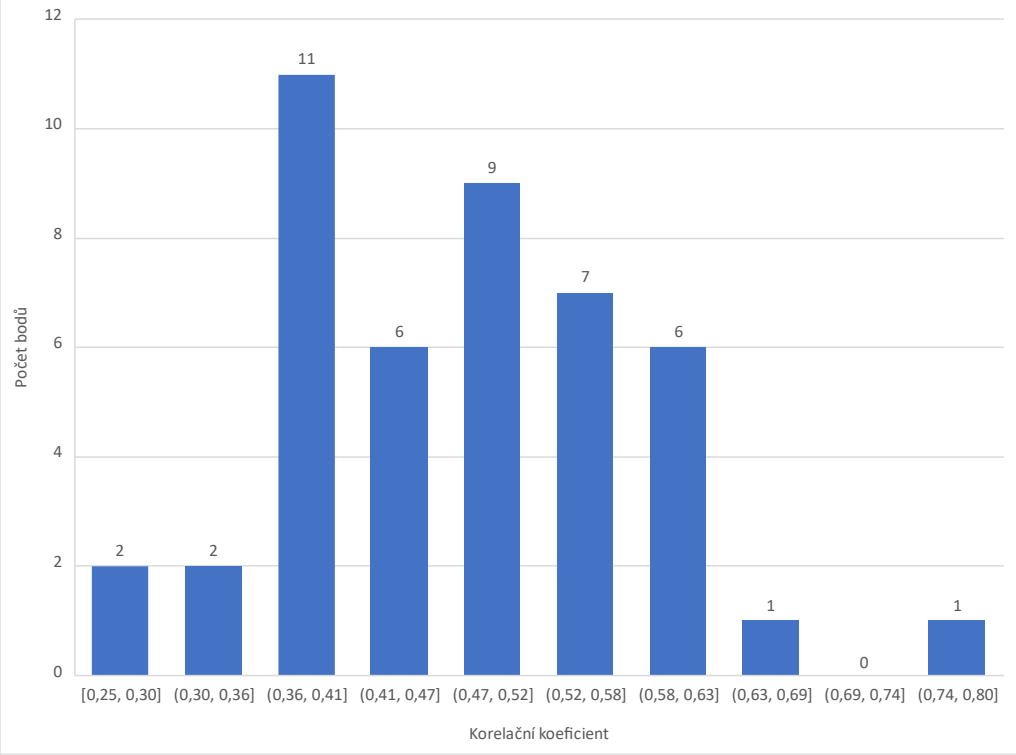
** Jako „RGB2grey" jsou pojmenovány varianty se šedotónovým obrazem vzniklým jmenovanou funkcí v OpenCV.*

Zdroj: vlastní tvorba

Testování nejrůznější parametrů (Tabulka 4) bylo provedeno na jednom ze snímků pro oblast Kobylí 1984, přičemž vstupem byly pouze křižovatky vytvořené na vrstvě „SilniceDalnice". Velikost prohledávané oblasti byla 82 m (hrana čtverce), pro každou křižovatku byl exportován pouze 1 odpovídající potenciální VLB, limit korelačního koeficientu nebyl uplatňován a ořezávací vzdálenost od okraje byla 5 m. Korelační koeficient a RMSE byly počítány ze 45 testovacích bodů. Z uvedených dat lze soudit, že použité pásmo (varianta 4 vs 5) má minimální vliv na výsledek a RMSE podobně i použité převzorkování referenčního ortofota (varianta 5 vs 9) a rozlišení použitých rastrů (varianta 1 vs 3 vs 5). Signifikantní vliv má pouze velikost plošky reprezentující křižovatky (varianta 5 vs 7 vs 8). Poměrně překvapivé byly velmi špatné výsledky při použití ortofota Cenia z 50. let (varianta 3 vs 6, případně 1 vs 2). Díky uvedeným skutečnostem bylo rozhodnuto pro využití parametrů odpovídajících variantě 1. Varianty 4, 5, 9 využívající lepší prostorové rozlišení rastrů vykazovaly sice ještě o trochu lepší výsledky, jednalo se však o minimální rozdíly v průměrné RMSE. Nižší rozlišení rastrů u varianty 1 má výhodu v menší datové náročnosti a nižších požadavcích na výpočetní výkon. Tyto parametry byly použity pro všechny datové sady. Níže (Graf 3) je ukázka kvality 45 testovacích bodů pro 5. variantu z testování (Tabulka 4). Také s pomocí tohoto testování byl stanoven limit korelačního koeficientu pro skript obrazové korelace. Kompletní soupis všech parametrů pro všechny datové sady je uveden na konci této kapitoly (Tabulka 6).

Graf 3: Kvalita výstupních testovacích bodů





Zdroj: vlastní tvorba

Jelikož však uvedené testování (Tabulka 4) bylo provedeno na snímku z 80. let, vyvstala otázka, zda jsou všechny výsledky relevantní i pro snímky z 30. let. Bylo proto provedeno další již méně rozsáhlé testování se snímkem z oblasti Kobylí 1938 s cílem zjistit, jestli by jakožto referenční podklad nebylo lepší ortofoto Cenia z 50. let (Tabulka 5).

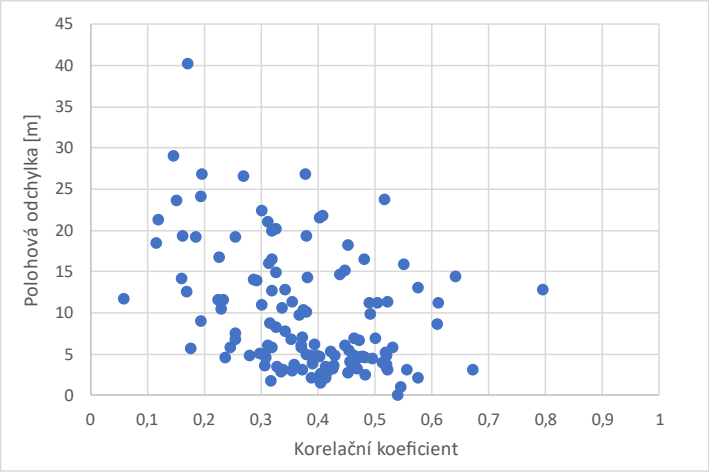
Tabulka 5: Porovnání různých referenčních podkladů – 30. léta

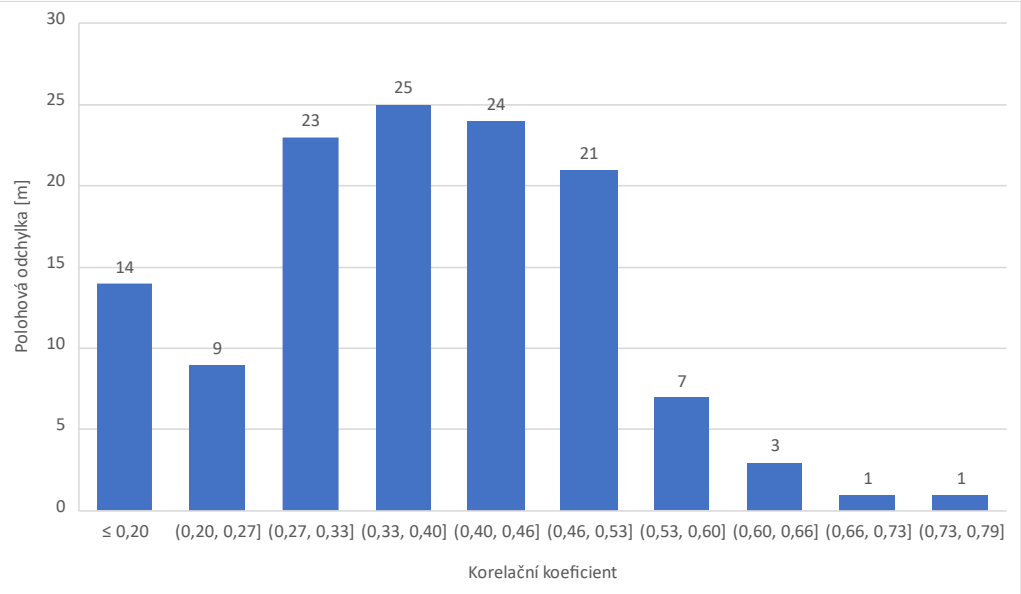
	50. léta	*současné – zelené pásmo*	*současné – RGB2grey*
korel. koef. – průměr	0,409	0,379	0,372
X RMSE [m]	9,90	8,55	8,77
Y RMSE [m]	8,65	8,70	8,86
průměr XY RMSE [m]	9,28	8,63	8,81

Zdroj: vlastní zpracování

Z výsledků je patrné (Tabulka 5), že ani pro 30. léta není ortofoto Cenia z 50. let lepší než současné ortofoto. Tyto závěry jsou poměrně překvapující, jelikož toto ortofoto zobrazuje krajinu ještě před scelováním polí, podobně jako je na LMS ze 30. let. Má však poměrně špatnou vizuální kvalitu (ostrost). Zajímavostí je, že pro 50. léta byl vyšší korelační koeficient. Za relevantní byla však považována průměrná RMSE, která byla lepší pro současné ortofoto. Zatímco pro datové sady z 80. let bylo použito současné ortofoto s průměrovanými pásmy s pomocí funkce RGB2grey v OpenCV, pro 30. léta bylo použito zelené pásmo současného ortofota. Níže (Graf 4) jsou uvedeny grafy pro zelené pásmo současného ortofota pro testovací snímek z oblasti Kobylí 1938.

Graf 4: Porovnání 30. léta

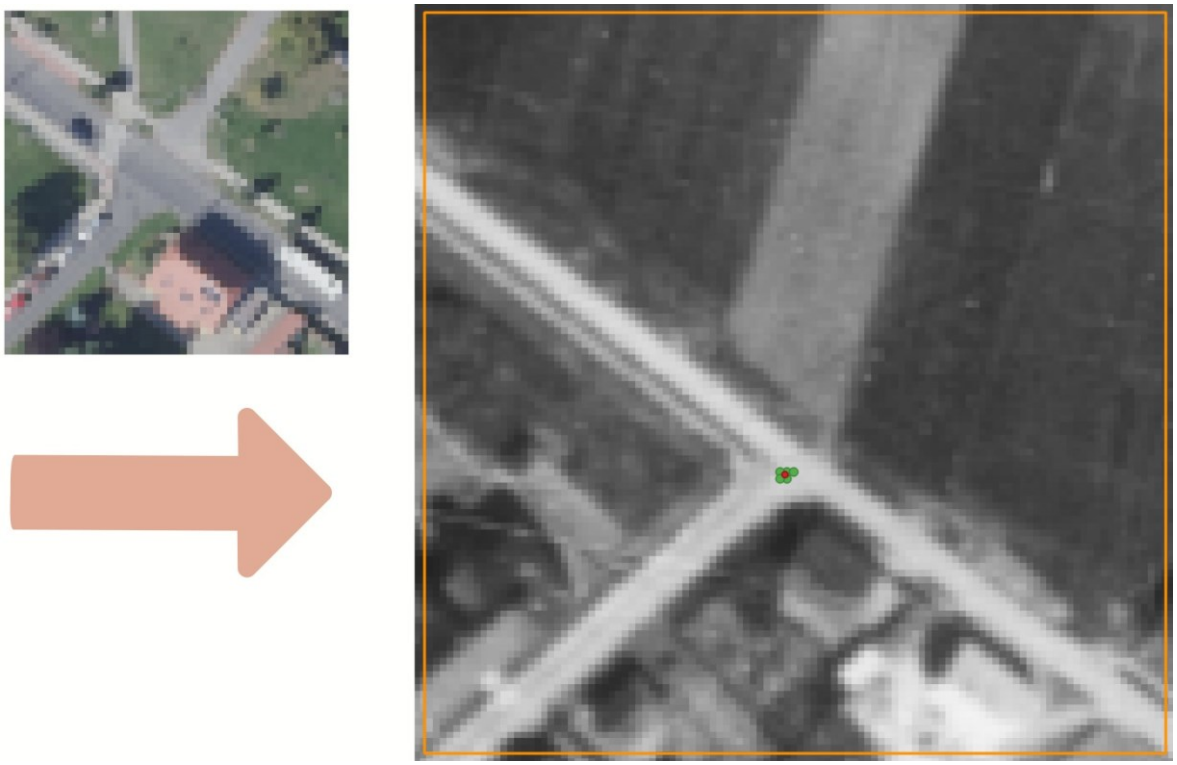




Zdroj: vlastní zpracování

Pro každou křižovatku bylo vyexportováno více potenciálních VLB. Tyto body tvořily jakési shluky (clustery), většinou uprostřed dané křižovatky. Byl tak naprogramován skript `6step_flood_collapse_clusters.py`, jehož vstupem jsou vrstvy potenciálních VLB z předchozího skriptu. Cílem bylo odstranit clustery a vytvořit z nich jeden výsledný potenciální VLB (Obrázek 25). Velké množství bodů v jednom místě by mohlo negativně ovlivňovat později použitou metodu RANSAC. Skript shlukuje body algoritmem vycházejícím ze Seed Fill algoritmu pro vyplňování rastrových oblastí, používá 8-sousedství a výsledné souřadnice bodu jsou počítány jako vážený průměr, přičemž nejvyšší váhu mají body s nejvyšším korelačním koeficientem v daném shluku. Výstupem skriptu jsou jednotlivé *.geojson soubory uložené do cílového adresáře, přičemž každý soubor obsahuje potenciální clusterované VLB k jednotlivým LMS.

Obrázek 25: Výsledek obrazové korelace a clusterování



Na obrázku je 5 bodů s nejvyšším korelačním koeficientem (zeleně) a výsledek clusterování (červeně). Clusterování často pomohlo přesněji nalézt střed křižovatky.

Zdroj: vlastní tvorba/podkladová data VGHMÚř Dobruška (2022), ČÚZK (2022a)

Ačkoliv díky hraničnímu limitu korelačního koeficientu byly odfiltrovány potenciální VLB nejhorší kvality, stále se v datech nacházelo množství odlehlých měření. Z tohoto důvodu byl naprogramován skript `7step_RANSAC.py`, jehož nejdůležitějším parametrem je cesta k adresáři obsahujícím veškeré *.geojson soubory s potenciálními VLB z předchozího kroku. Dále je možno nastavit počet iterací RANSACu, použitou transformaci, přičemž lze volit mezi 2D a 3D fotogrammetrickými transformacemi (projektivní tr./DLT) a prahovou vzdálenost, kterou je určeno, zda bod vyhovuje danému modelu. Bohužel Open CV neobsahuje RANSAC v kombinaci s vhodnou 3D transformací (pouze 2D projektivní/afinní). Proto musel být RANSAC naprogramován od algoritmické úrovně. Pro výpočet koeficientů DLT byla použita Python implementace dostupná na GitHubu (Ankita 2019). Každou vrstvou obsahující potenciální VLB k jednotlivým LMS je iterováno, přičemž počet iterací je dán parametrem na vstupu. V každé iteraci jsou z vrstvy náhodně vybrány 4 resp. 6 bodů (projektivní tr./DLT), díky znalosti geodetických i rastrových souřadnic uložených v atributech mohou být dopočítány koeficienty dané transformace. Následně je provedena transformace všech bodů a zkoumá se, kolik bodů z celé vrstvy splňuje podmínku prahové vzdálenosti. Body z iterace, ve které nejvíce bodů

splňovalo tuto podmínku, jsou uloženy do výstupního *.geojson souboru (Obrázek 26). Pokud se vyskytne více iterací se stejným počtem bodů, pak se zkoumá průměrná odchylka VLB od modelu. Iterace s nižší odchylkou je považována za lepší. Jedná se o typický RANSAC algoritmus, podrobnější popis je v teoretické části práce. Výstupem jsou vrstvy *.geojson pro každý LMS (Obrázek 27), které obsahují vyfiltrované VLB, jenž mají v atributu uloženou vzdálenost od nejlepšího geometrického modelu.

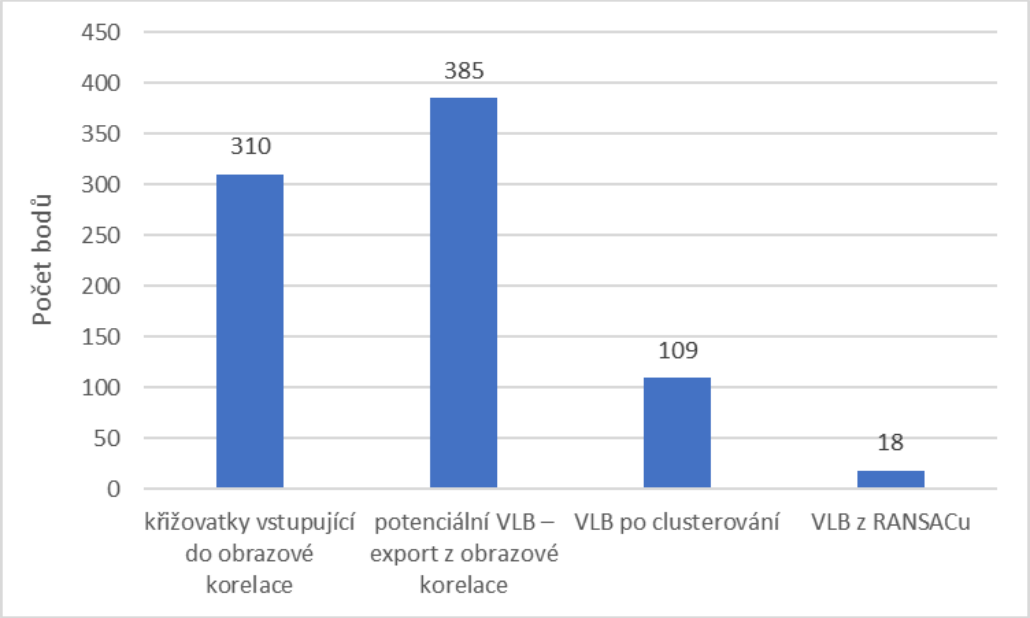
Obrázek 26: Ukázka výpisu skriptu 7step_RANSAC.py pro jeden snímek z oblasti Kobylí 1984

```
##################################################

Final number of points fitting threshold distance: 26 in image: 1m_1984_15879

 Point IDs setting up transformation of final best model --> [29, 18, 8, 20, 4, 35]

 --> Points were found in 73435 RANSAC iteration. Sum of all distances computed in
RANSAC for best model: 22.32 pixels  Mean distance for whole dataset: 0.858 pixels

 Number of not unique selection / singular matrixes / zero_division: 2442 / 0 / 0
```

Zdroj: vlastní tvorba

Na další straně jsou vizualizovány výstupy skriptů (Obrázek 28) na příkladu území části Milovic. V archivním LMS jsou už zobrazeny pouze clusterované body a body vyfiltrované RANSACem. Všechny křižovatky z referenčního podkladu (současného ortofota) samozřejmě nemají odpovídající křižovatku v archivním LMS, jelikož body s příliš nízkým korelačním koeficientem vůbec nebyly vyexportovány ve skriptu obrazové korelace (`5step_correl_patches.py`). Pro jednu křižovatku se může vyskytnout více než jeden potenciální clusterovaný VLB, byť ve většině případů byl přítomen pouze jediný. Případně se clusterované body překrývají v důsledku velmi malé vzdálenosti. Lze si všimnout, že v RANSACu byla odfiltrována většina bodů. Níže (Graf 5) je uveden počet bodů v jednotlivých krocích metodiky. RANSAC drasticky snížil počet potenciálních VLB – došlo k odfiltrování odlehlých měření. Data se vztahují ke stejnému snímku z oblasti Milovic jako Obrázek 28.

Graf 5: Počet potenciálních VLB v jednotlivých fázích metodiky pro snímek z oblasti Milovic 1938



Zdroj: vlastní zpracování

Obrázek 27: Ukázka atributů výstupní vrstvy z RANSACu pro 1 snímek

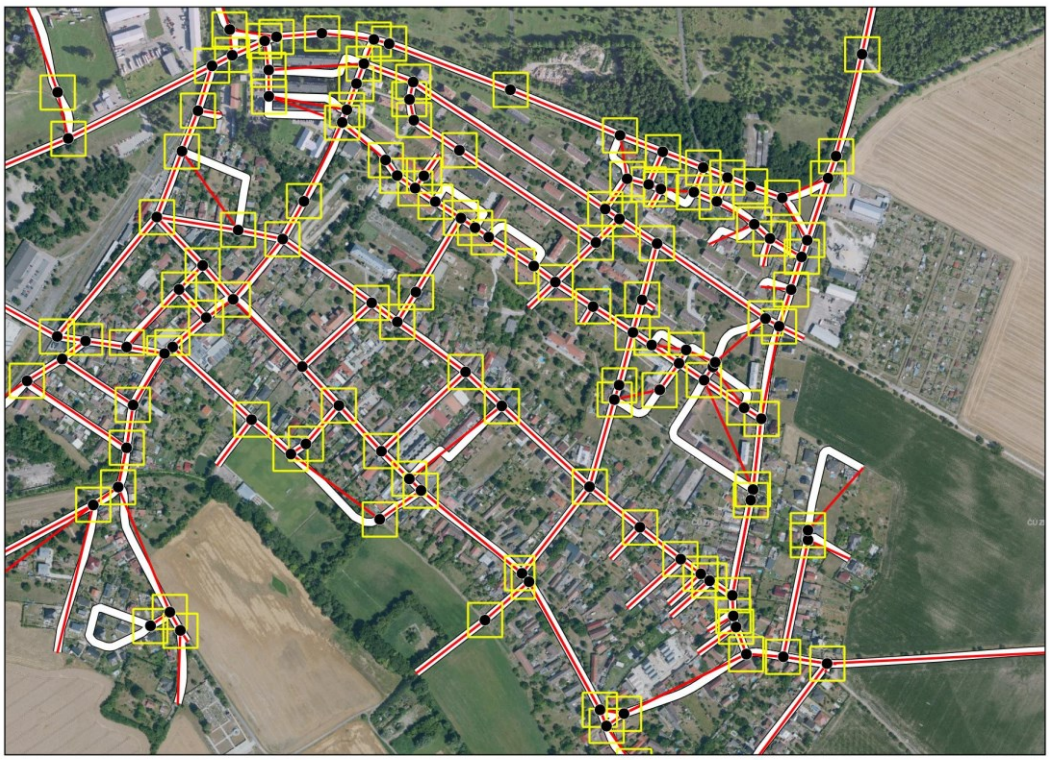
	cross_id	Z	quality	X_ref	Y_ref	x_col	y_row	id	RAN_dist_pix▲
1	659	197	0,47	-707199,61	-1032405,86	2111,001	3410,003	9	0
2	1343	191	0,663	-708107,69	-1031988,29	1228,595	2991,197	85	0
3	1347	193	0,593	-708027,71	-1032411,07	1306	3402	88	0
4	1350	198	0,416	-707742,55	-1031922,32	1590,798	2940,402	89	0
5	1255	194	0,544	-707875,17	-1031827,11	1460,801	2845,6	57	0,51
6	1345	193	0,436	-708043,29	-1032383,79	1291,801	3375	86	1,308
7	833	199	0,403	-707063,98	-1032460,48	2241	3464	18	3,025
8	781	192	0,498	-707989,24	-1031944,49	1344,503	2951,502	11	3,36
9	658	197	0,457	-707184,18	-1032446,4	2123	3445,999	8	3,497
10	792	191	0,438	-708107,8	-1032138,61	1229,2	3139,398	17	3,611
11	1354	199	0,424	-707602,66	-1032025,44	1730,193	3042,396	90	3,761

Atributová tabulka z QGISu se vztahuje k vrstvě, na které byl demonstrován RANSAC v teoretické části práce (Obrázek 12) – Milovice 1938, iterace č. 35 968. (jiný snímek než předchozí graf a následující obrázek)

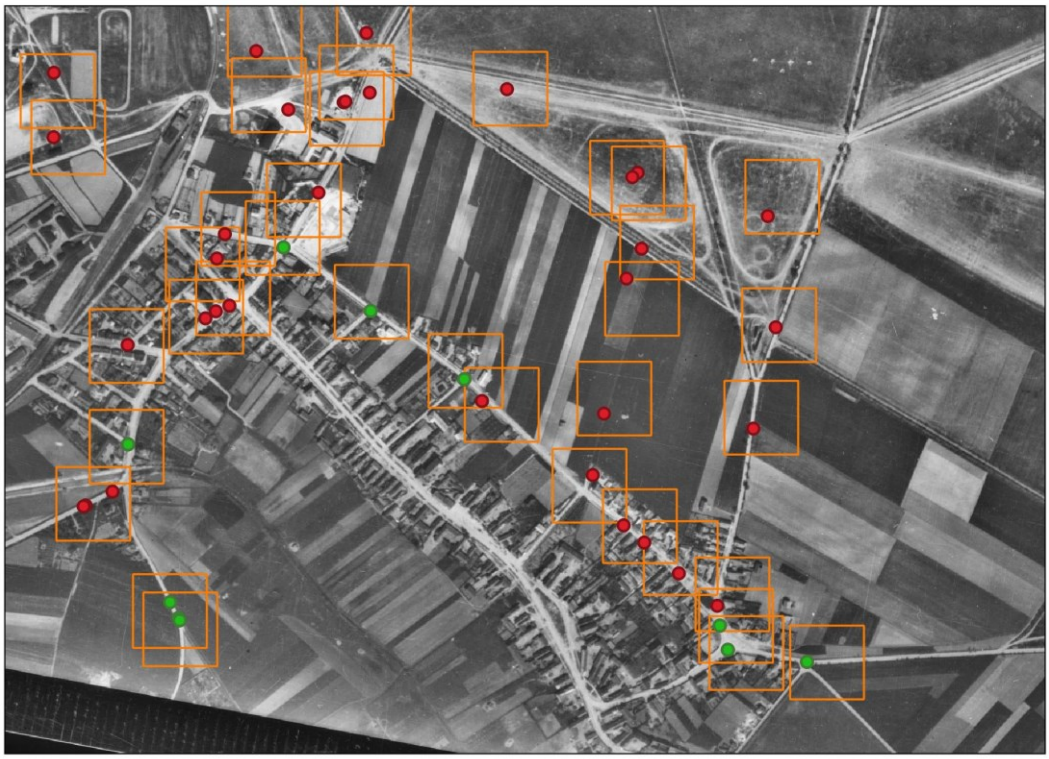
cross_id: id křižovatky, Z: nadmořská výška, quality: korelační koef., X_ref Y_ref: geodetické souř., x_col y_row: rastrové souř., id: individuální id pro každý VLB, RAN_dist_pix: odchylka od modelu v RANSACu v pixelech

Zdroj: vlastní tvorba

Obrázek 28: Ukázka automatizovaného zpracování dat z oblasti Milovic 1938



Referenční podklad
- exportované křižovatky
- ohraničení plošky reprezentující křižovatky
- hrany grafu
- komunikace („SilniceDalnice" + „Ulice")



Archivní LMS
- prohledávaná oblast
- body po clusterování
- body po clusterování a po RANSACu (finální VLB)

V archivním LMS již nemají odpovídající bod všechny křižovatky, jelikož mnoho potenciálních VLB neprošlo přes limit korelačního koeficientu.

0 250 500 m

Zdroj: vlastní tvorba/podkladová data VGHMUř Dobruška (2022), ČUZK (2022a)

Kompletní popis rozhraní skriptů je v příloze na konci práce. Na následující straně jsou uvedeny parametry (Tabulka 6) vstupující do metodiky pro všechny datové sady. Při volbě jednotlivých parametrů bylo nutno zohlednit polohovou přesnost snímků, rovnoměrnost rozmístění komunikací po snímcích, měřítko snímků a další faktory.

Pro datové sady z 80. let nebyly použity ulice ani cesty, jelikož snímky zabíraly větší oblast a byly dostatečně pokryty křižovatkami pouze z vrstvy „SilniceDalnice“. Pro 30. léta musely být použity i výše zmíněné vrstvy, jelikož při použití samotných křižovatek silnic by nebyly s pomocí VLB propojeny některé stereo dvojice, což by následně znemožnilo tvorbu ortofota. Oblast k prohledávání musela být volena větší u snímků z 80. let, jelikož měly proměnlivější měřítko v ploše snímku. Oblast zahrnuje i samotnou křižovatku. Z toho vyplývá, že při velikosti plošky reprezentující křižovatku 51 m a velikosti pixelu 1 m teoreticky připadá v úvahu nejmenší oblast k procházení 52 m. Prohledávaná oblast byla volena spíš mírně větší, aby byly nalezeny VLB na okraji snímku, kde byla vyšší polohová odchylka, ale zároveň byly tyto VLB velmi důležité při následném zpracování ortofota pro propojení stereo dvojic. U VLB pro snímky z 30. let byl nastaven nižší limit korelačního koeficientu, jelikož v těchto datových sadách vycházel korelační koeficient horší a bylo by odfiltrováno příliš mnoho bodů. Při pohledu na data (Tabulka 6) je možné soudit, že z RANSACu bylo vyexportováno velké množství bodů, nicméně tyto VLB byly rozmístěny značně nerovnoměrně. Počet se zároveň nevztahuje k množství unikátních bodů, např. VLB nacházející se ve 3 snímcích je započítán 3 krát.

Pro Milovice 1989 byl proveden malý experiment. Oblast k prohledávání byla nastavena záměrně velká, což mělo simulovat méně přesné georeference. Taktéž byla v RANSACu použita projektivní transformace místo DLT, jelikož se jednalo o rovinné území. Přesto se podařilo dosáhnout uspokojivého výsledku a VLB i ortofoto byly kvalitní. Nejednalo se však o samozřejmost. Metodika byla vyzkoušena s datovou sadou Milovice 1938 s chybnými georeferencemi ČÚZK, v tomto případě se však nepodařilo dosáhnout úspěchu a ani RANSAC nedokázal odfiltrovat chybně nalezené VLB. Milovice 1938 se podařilo zpracovat až po opravě georeferencí.

Tabulka 6: Parametry použité v metodice pro jednotlivé datové sady

	Milovice 1938	*Milovice 1989*	*Kobylí 1938*	*Kobylí 1984*
Zpracování podkladových dat ↓				
počet LMS	6	3	6	3
transformace ve world file souborech pro LMS	podobnostní (Helmertova) tr.	podobnostní (Helmertova) tr.	podobnostní (Helmertova) tr.	podobnostní (Helmertova) tr.
převzorkování LMS i současného ortofota	bilineární	bilineární	bilineární	bilineární
rozlišení všech rastrů	1 m	1 m	1 m	1 m
komunikace	Vrstva „SilniceDalnice“ rozsekaná vrstvami „Ulice“ a „Cesta“ + vrstva „Ulice“	Vrstva „SilniceDalnice“ rozsekaná vrstvou „Ulice“	Vrstva „SilniceDalnice“ rozsekaná vrstvami „Ulice“ a „Cesta“ + vrstvy „Ulice“, „Cesta“	Vrstva „SilniceDalnice“ rozsekaná vrstvou „Ulice“
nadmořská výška křižovatky	DMP 1G, rastr 2 m	DMP 1G, rastr 2 m	DMP 1G, rastr 2 m	DMP 1G, rastr 2 m
referenční podklad	současné ortofoto	současné ortofoto	současné ortofoto	současné ortofoto
Zpracování potenciálních VLB – obrazová korelace ↓				
velikost plošky reprezentující křižovatku	51 m	51 m	51 m	51 m
použité pásmo	zelené	RGB2grey	zelené	RGB2grey
velikost prohledávané oblasti (hrana čtverce)	110 m	440 m	90	160 m
počet potenciálních VLB pro každou křižovatku	5	5	5	5
limit korelačního koeficientu	0,4	0,45	0,4	0,45
vzdálenost pro odmazání od okraje LMS	5 m	5 m	5 m	5 m
Clusterování ↓				
vzor	8-sousedství	8-sousedství	8-sousedství	8-sousedství
výpočet nového bodu	průměr souřadnic vážen korelačním koeficientem	průměr souřadnic vážen korelačním koeficientem	průměr souřadnic vážen korelačním koeficientem	průměr souřadnic vážen korelačním koeficientem
RANSAC ↓				
transformace	DLT	projektivní	DLT	DLT
počet iterací	100 000	100 000	100 000	100 000
prahová vzdálenost	4 m	3 m	2 m	2 m
počet výstupních bodů	121	55	157	74
průměrná vzdálenost bodů od ideálního modelu	1,79 m	1,28 m	0,95 m	0,89 m

Zdroj: vlastní zpracování

4.6 Zpracování archivních LMS v SW MicMac

Po automatizovaném vyhledávání VLB bylo přistoupeno k dalšímu zpracování archivních LMS. Standardní kroky zpracování ortofota byly provedeny v open source fotogrammetrickém SW MicMac ovládaném z příkazového řádku. Pracovní postup byl vesměs realizován podle tutoriálu, který je dostupný přímo na stránkách MicMacu (Girod 2022). Nicméně v tomto tutoriálu jsou nepřesnosti i chyby, např. v pojmenování modulů (`SaisieAppuisInit` vs `SaisieAppuisInitQT`). Nejen z tohoto důvodu nebyl tutoriál následován zcela přesně. Dále musel být postup přizpůsoben importu VLB z automatizované metodiky popsané v předchozí kapitole a specifikům zpracovávaných datových sad. Z dalších dostupných velmi užitečných publikací lze jmenovat elektronickou knihu kompletně popisující MicMac (IGN 2022) a dokumentaci k modulům, která je k dispozici na stránkách SW. Jednotlivé moduly MicMacu jsou pojmenované často zkratkami (např. „Campari" pro svazkové vyrovnání – **C**ompensation of **A**lter **M**easurements for **P**hotomatric **A**djustment after **R**otation (and position and etc...) **I**nitialisation).

Nejprve byly změřeny rastrové souřadnice všech čtyř rámových značek u jednoho ze snímků v grafickém SW GIMP. S pomocí známé velikosti pixelu při skenování (15 μm) byly pixelové souřadnice převedeny na snímkové. Poté byla založena složka `Ori-InterneScan` v kořenovém adresáři projektu a v ní soubor `MeasuresCamera.xml` obsahující vypočítané snímkové souřadnice rámových značek. Tyto souřadnice totiž nejsou poskytovány ze strany ČÚZK, slouží však pro stanovení vztahu mezi pixelovým a snímkovým souřadnicovým systémem. Uvedeným způsobem byl hlavní snímkový bod namapován na průsečík rámových značek. Následně bylo nutno vytvořit soubor `MicMac-LocalChantierDescripteur.xml` obsahující další prvky IO (konstanta kamery, čisté rozměry snímku vymezené rámovými značkami) pro projekt. Strukturu těchto souborů uvádí Girod (2022).

Předchozí měření rámových značek u jednoho LMS bylo provedeno za účelem získání jejich snímkových souřadnic při založení projektu, neboť je to jeden z důležitých údajů pro stanovení vnitřní orientace (IO). Následovalo měření rámových značek pro získání jejich rastrových souřadnic u všech snímků s pomocí modulu MicMacu po založení projektu. Za tímto účelem byl vytvořen soubor `id_fiducial.txt` obsahující indexy rámových značek. Následně bylo možno přistoupit k samotnému měření. K tomuto účelu slouží v MicMacu modul `SaisieAppuisInitQT`, podoba příkazu pro 1 snímek je následující:

```
mm3d SaisieAppuisInitQT "1989_16401.tif" NONE "id_fiducial.txt" "MeasuresIm-
1989_16401.tif.xml"
```

Po jednom měření lze s pomocí modulu `Kugelhupf` doměřit automaticky rámové značky na zbylých snímcích:

```
mm3d Kugelhupf .*tif Ori-InterneScan/MeasuresIm-1989_16402.tif.xml
SearchIncertitude=1000
```

Následuje poslední krok pro stanovení IO. Modul `ReSampFid` s pomocí rozlišení skenu (15 µm) převzorkuje všechny vstupní rastry do stejného prostorového rozlišení, aby bylo možno s LMS pracovat jako s digitálním obrazem (Girod 2022) a zároveň jsou LMS vyříznuty z rámů podle mapových značek (Obrázek 29). Tento krok v závislosti na poloze rámových značek ve snímku bohužel může oříznout velkou část snímku v podélném překrytu, což někdy vede k problémům (u roku 1938 – diskutováno dále v práci). Výstupem jsou převzorkované snímky `OIS-Reech_*.tif` se stanovenou vnitřní orientací, s nimiž se výhradně pracuje při veškerém následném zpracování v SW MicMac.

```
mm3d ReSampFid ".*.tif" 0.015
```

Po výpočtu IO následuje relativní orientace. Nejprve jsou automaticky nalezeny potenciální spojovací body pomocí operátoru SIFT (IGN 2022), což je realizováno krokem:

```
mm3d Tapioca File img_pairs.xml 0.4
```

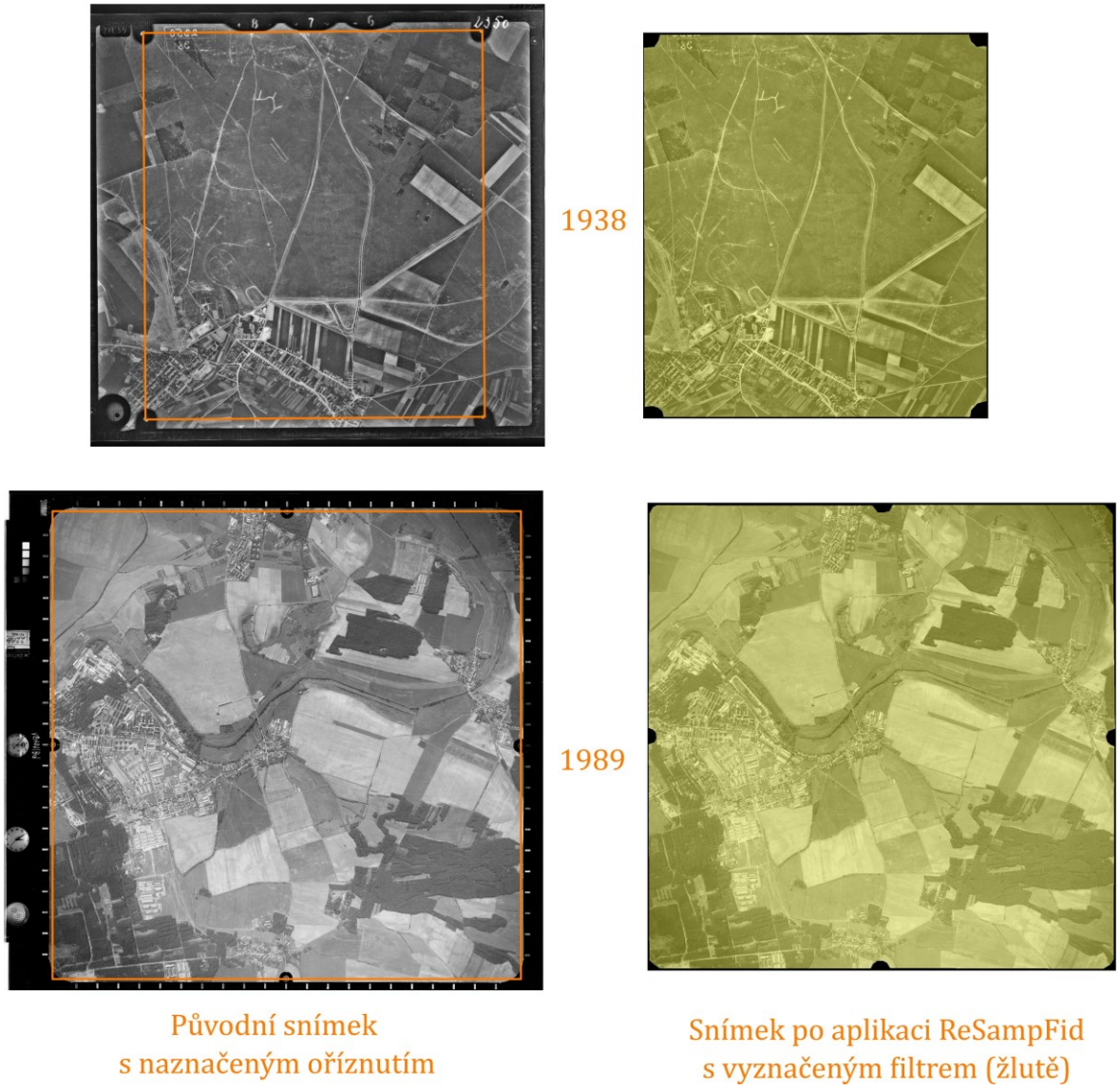
Soubor `img_pairs.xml` obsahuje seznam stereo dvojic v celé datové sadě, jeho specifikace uvádí IGN (2022). Alternativně by bylo možno použít `mm3d Tapioca MulScale`, nicméně explicitní zadání stereodvojic redukovalo nutný výpočetní čas pro párování spojovacích bodů. Pro párování bodů je využito ANN (approximate nearest neighbour) (IGN 2022). Následně byla s pomocí jednoho ze snímků vytvořena maska (modul `SaisieMasqQT`) označující čistou plochu snímku bez terčíků vyznačující rámové značky (Obrázek 29):

```
mm3d SaisieMasqQT "OIS-Reech_1989_16401.tif"
```

Výsledná maska (`filtre.tif`) byla následně aplikována na všechny snímky:

```
mm3d HomolFilterMasq "OIS.*tif" GlobalMasq=filtre.tif
```

Obrázek 29: Ukázka vybraných kroků zpracování v MicMacu



V případě snímků ze 30. let funkce ReSampFid bohužel ořízla podstatnou část snímků v podélném překrytu.

Zdroj: vlastní tvorba/podkladová data VGHMÚř Dobruška (2022)

Posledním krokem byl samotný výpočet relativní orientace:

```
mm3d Tapas RadialBasic "OIS.*tif" Out=Relative SH=HomolMasqFiltered LibFoc=0
```

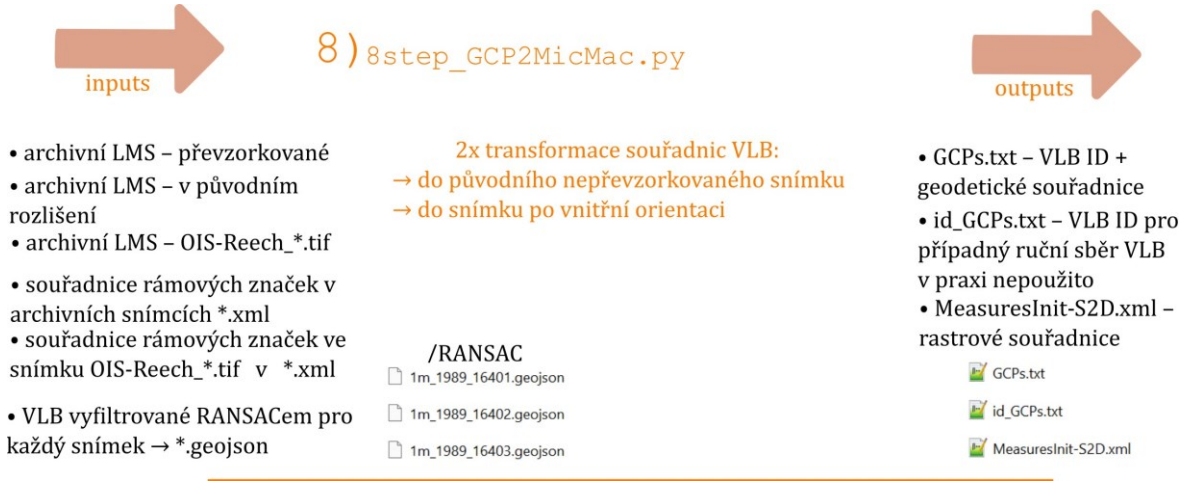
Následujícím krokem byl výpočet absolutní orientace. Pro jednu datovou sadu byl pouze na zkoušku proveden manuální sběr několika VLB. Za tímto účelem byly vytvořeny 3 soubory. Jedná se o soubor `GCPs.txt` obsahující ID a geodetické souřadnice všech bodů pro danou datovou sadu včetně nadmořských výšek. Dále pak soubor `id_GCPs.txt`, ve kterém jsou uvedeny pouze ID všech VLB pro celou sadu snímků. Tento soubor byl použit v modulu `SaisieAppuisInitQT`:

```
mm3d SaisieAppuisInitQT "OIS-Reech_1989_16403.tif" Relative id_GCPs.txt
MeasuresInit-S2D.xml
```

V tomto modulu bylo na zkoušku manuálně sesbíráno několik VLB v LMS, následně mohl být z modulu vyexportován soubor `MeasuresInit-S2D.xml`. Jedná se o *.xml soubor s proprietární strukturou obsahující ID a rastrové souřadnice všech VLB pro jednotlivé snímky. Reverzním inženýrstvím byla zjištěna struktura tohoto souboru, aby poté mohl být vytvořen vlastní *.xml soubor obsahující rastrové souřadnice VLB z automatizovaného postupu této práce.

Pro import automatizovaně nalezených VLB byl naprogramován poslední skript `8step_GCP2MicMac.py`. (Obrázek 30) Skript otevře všechny *.geojson soubory s VLB (výstupy z RANSACu) a postupně vytvoří 3 soubory uvedené v předchozích odstavcích (`GCPs.txt`, `id_GCPs.txt`, `MeasuresInit-S2D.xml`). Zejména `MeasuresInit-S2D.xml` a `GCPs.txt` jsou velmi důležité pro další zpracování snímků v MicMacu.

Obrázek 30: Diagram – import VLB do MicMacu



Zdroj: vlastní tvorba

Rastrové souřadnice VLB, které byly automatizovaně nalezené v předchozí kapitole, se vztahovaly k LMS převzorkovaným na 1 m (dostačující pro vyhledávání křižovatek), avšak v SW MicMac se pracovalo se snímky v lepším rozlišení, které navíc prošly vnitřní orientací (výstup funkce `ReSampFid – OIS-Reech_*.tif`). Ve skriptu jsou proto nejprve souřadnice namapovány do originálních nepřevzorkovaných rastrů. Toho bylo dosaženo převodem rastrových souřadnic VLB v LMS s rozlišením 1 m na souřadnice geodetické a následně jejich opětovným převodem

na rastrové, tentokrát však namapované do LMS s původním prostorovým rozlišením, tak jak je poskytuje ČÚZK.

Dále bylo nutno tyto souřadnice transformovat do rastrů `OIS-Reech_*.tif` (se stanovenou IO), se kterými se pracuje dále v SW MicMac. Tak jako v původních snímcích byly v jednom ze snímků `OIS-Reech_*.tif` změřeny souřadnice rámových značek. Následně byly vypočteny koeficienty afinní transformace mezi původními a `OIS-Reech_*.tif` snímky s pomocí rámových značek. Díky tomuto kroku mohly být rastrové souřadnice VLB transformovány do snímků se stanovenou IO. Výstupem skriptu `8step_GCP2MicMac.py` je soubor `MeasuresInit-S2D.xml` (Obrázek 31) obsahující rastrové souřadnice VLB z automatizovaného postupu pro jednu datovou sadu se strukturou kompatibilní s moduly SW MicMac.

Obrázek 31: Ukázka 3 VLB v MeasuresInit-S2D.xml pro 1 snímek

```
<?xml version='1.0' encoding='utf-8'?>
<SetOfMesureAppuisFlottants>
    <MesureAppuiFlottant1Im>
        <NameIm>OIS-Reech_1989_16401.tif</NameIm>
        <OneMesureAF1I>
            <NamePt>GCP401</NamePt>
            <PtIm>6690.907297579322 7137.620885125456</PtIm>
        </OneMesureAF1I>
        <OneMesureAF1I>
            <NamePt>GCP402</NamePt>
            <PtIm>6712.910146881075 7203.62297561638</PtIm>
        </OneMesureAF1I>
        <OneMesureAF1I>
            <NamePt>GCP403</NamePt>
            <PtIm>6742.914574768234 7307.626205230516</PtIm>
        </OneMesureAF1I>
```

Zdroj: vlastní tvorba

V dalším kroku opět již v SW MicMac byly geodetické souřadnice VLB (`GCPs.txt`) uloženy do *.xml s proprietární strukturou:

```
mm3d GCPConvert AppInFile GCPs.txt
```

Tento krok by pravděpodobně bylo možno realizovat i v Pythonu, nicméně se to zdálo být zbytečné, jelikož k těmto účelům má použitý SW implementovanou funkci. Následně byly automatizovaně nalezené VLB v souboru `MeasuresInit-S2D.xml` importovány do projektu:

```
mm3d GCPBascule "OIS-.*tif" Relative TerrainBrut GCPs.xml  MeasuresInit-S2D.xml
```

Posledním krokem absolutní orientace bylo svazkové vyrovnání (bundle adjustment) v modulu `Campari` (Obrázek 32):

```
mm3d Campari "OIS-.*tif" TerrainBrut TerrainFinal GCP=[GCPs.xml,3,MeasuresInit-
S2D.xml,5.22] SH=HomolMasqFiltered AllFree=1
```

Obrázek 32: Konečný výpis do konzole z modulu Campari pro Milovice 1989

```
    ============================== ERRROR MAX PTS FL ======================
    ||    Value=5.75276 for Cam=OIS-Reech_1989_16401.tif and Pt=GCP503 ; MoyErr=1.92471
    =======================================================================

  RES:[OIS-Reech_1989_16401.tif][C] ER2 0.687393 Nn 99.6093 Of 6655 Mul 0 Mul-NN 0 Time 0.0830002
  RES:[OIS-Reech_1989_16402.tif][C] ER2 0.778199 Nn 99.4173 Of 12184 Mul 260 Mul-NN 256 Time 0.152
  RES:[OIS-Reech_1989_16403.tif][C] ER2 0.832713 Nn 99.1536 Of 5789 Mul 0 Mul-NN 0 Time 0.072
  ----- Stat on type of point (ok/elim) ----
       *   Perc=99.4072% ;  Nb=24482 for Ok
       *   Perc=0.540036% ;  Nb=133 for PdsResNull
       *   Perc=0.0284229% ;  Nb=7 for Behind
       *   Perc=0.0243625% ;  Nb=6 for VisibIm
  ---------------------------------------
  | |  Residual = 0.768443 ;; Evol, Moy=1.15458e-05 ,Max=7.47702e-05
  | |  Worst, Res 0.832713 for OIS-Reech_1989_16403.tif,  Perc 99.1536 for OIS-Reech_1989_16403.tif
  | |  Cond , Aver 1.97403 Max 3.89837 Prop>100 0
  --- End Iter 8 STEP 0
```

Zdroj: vlastní tvorba

Snímky byly provizorně spojeny, aby mohla být nakreslena oblast zájmu, která zahrnovala v podstatě celou plochu snímků:

```
mm3d Tarama "OIS-.*tif" TerrainFinal
```

```
mm3d SaisieMasqQT TA/TA_LeChantier.tif
```

Následovalo generování DMP s rozlišením 0.5 m. Další nastavení funkce zahrnuje např. nutný počet snímků pro výpočet nadmořské výšky (`NbVI=2`) nebo minimální korelační koeficient pro zahrnutí daného bodu do výpočtu (`DefCor=0`):

```
mm3d Malt Ortho "OIS-.*tif" TerrainFinal MasqImGlob=filtre.tif NbVI=2 ZoomF=2
ResolTerrain=0.5 DefCor=0 CostTrans=4 EZA=1
```

Konečně bylo možno vygenerovat ortofoto mozaiku s prostorovým rozlišením 0,5 m jako poslední krok zpracování LMS v MicMacu:

```
mm3d Tawny Ortho-MEC-Malt Out=Orthophotomosaic.tif
```

Tento postup byl proveden pro všechny 4 datové sady. Parametry pro SW MicMac v této kapitole v ukázkách příkazů odpovídají datové sadě Milovice 1989.

5 Výsledky

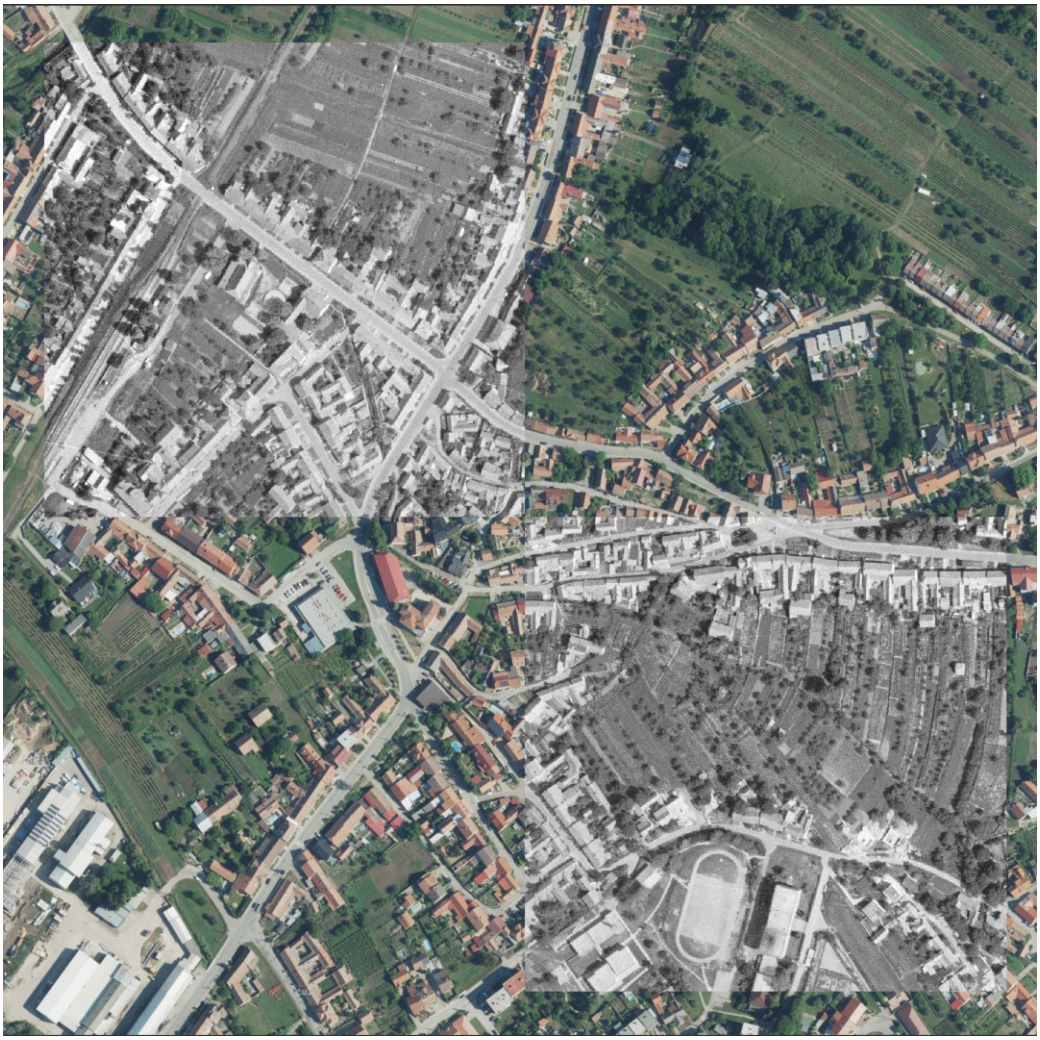
5.1 Metodika pro plně automatizované vyhledávání VLB

Výsledkem je sestavená metodika pro plně automatizované vyhledávání VLB v archivních LMS s pomocí referenčních dat realizovaná sadou 10 skriptů v jazyce Python 3 (v příloze). Tyto skripty využívají standardizované neproprietární open source knihovny a tvoří kompletní automatizovaný metodický postup, přičemž výstup jednoho skriptu je typicky vstupem pro skript další. Díky tomu bylo možno vizualizovat mezivýsledky a kontrolovat průběh celého zpracování dat. Mezivýstupy se hodí i pro správné nastavení jednotlivých parametrů v dílčích krocích metodického postupu. Vstupem pro metodiku jsou archivní LMS, dále pak jejich stopy, odpovídající listy současného ortofota, rastrový DMP a síť komunikací. Zvoleným operátorem pro vyhledávání identických bodů mezi archivním LMS a referenčním obrazem byla obrazová korelace. Konečným výstupem automatizovaného postupu je *.xml soubor obsahující VLB pro všechny snímky importovatelný do SW MicMac pro výpočet svazkového vyrovnání.

5.2 Ortofoto mozaiky

Výsledným produktem z SW MicMac jsou ortofoto mozaiky pro obě území a časové řezy s prostorovým rozlišením 50 cm. Mozaiky mají dobrou kvalitu co se týče slícování snímků ve stereo dvojicích. Při zobrazení společně se současným ortofotem byla vizuálně zkontrolována návaznost jednotlivých prvků v krajině jako jsou například silnice (Obrázek 33). Návaznost lze shrnout jako velmi dobrou. Za slabší lze označit pouze mozaiku pro Milovice 1938, která byla nejhorší i v hodnocení přesnosti. V ortofoto mozaikách ze 30. let vznikaly černé pruhy (Obrázek 34) v místech návaznosti stereo dvojic. Tento problém byl důsledkem zpracování v MicMacu a použití funkce ReSampFid (Obrázek 29 – viz předchozí kapitola), která ořezává LMS podle rámových značek. Negativní jev neplyne z automatizovaného vyhledávání VLB vyvinutého v této práci. Rozložení rámových značek bylo bohužel u snímků ze 30. let značně neideální.

Obrázek 33: Vizualizace polohové přesnosti ortofota – Kobylí 1984



Pro zajímavost: v levém čtverci starého ortofota vede řezová linie mezi 2 snímky, která je v podstatě neznatelná.

Zdroj: vlastní tvorba/podkladová data VGHMÚř Dobruška (2022), ČÚZK (2022a)

Obrázek 34: Ukázka chybějící části v ortofoto mozaice v území Kobylí 1938

Zdroj: vlastní tvorba/podkladová data VGHMÚř Dobruška (2022)

5.3 Hodnocení přesnosti

Pro zhodnocení přesnosti vytvořených ortofot byl použit vzorec střední kvadratické odchylky (RMSE) pro geodetické souřadnice X a Y. V každé mozaice bylo v území rovnoměrně rozmístěno 10 kontrolních bodů (Obrázek 35). Body pro jednotlivé časové řezy se lišily, jelikož ortomozaiky nepokrývaly přesně stejnou oblast.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\bar{y}_i - y_i)^2}{N}}$$

kde:

$$\bar{y}_1, \bar{y}_2, \dots, \bar{y}_N = souřadnice\ v\ referenčním\ ortofotu$$
$$y_1, y_2, \dots, y_N = souřadnice\ ve\ vytvořeném\ ortofotu$$
$$N = počet\ měření$$

Níže jsou zobrazeny vypočítané odchylky pro všechny ortomozaiky (Tabulka 7). Nejhorší jsou Milovice 1938, kde byly nejdrastičtější změny v krajině. Tato mozaika i z hlediska vizuálního nevypadala příliš uspokojivě. Ostatní datové sady nevykazovaly žádné abnormality a podařilo se dosáhnout hodnot typických pro ortofota ze starých LMS (viz ortofoto Cenia), případně i lepších (Kobylí 1984 – RMSE nižší než 1 m). Například ortofoto Cenia má přesnost cca 1,5 m pro poválečné a 1,3 m pro předválečné snímky. Zájmová území v této práci by měla být pokryta snímky z let 1953 a 1954 (Sukup 2010).

Tabulka 7: RMSE pro všechny vytvořené ortomozaiky

	Kobylí 1938	*Kobylí 1984*	*Milovice 1938*	*Milovice 1989*
X [m]	1,21	0,60	3,52	1,30
Y [m]	2,20	0,88	7,57	1,56
průměr XY [m]	1,70	0,74	5,55	1,43

Zdroj: vlastní tvorba

Obrázek 35: Ukázka rozmístění kontrolních bodů v mozaice Milovice 1989

Zdroj: vlastní tvorba/podkladová data VGHMÚř Dobruška (2022)

6 Diskuze

6.1 Příčiny selhávání testovaných operátorů

V průběhu zpracování práce byl problémem nedostatek literatury zabývající se automatizovaným vyhledáváním VLB v archivních snímcích. Samozřejmě existuje množství článků, které pracují s rastry (LMS, snímky z dronu, data DPZ) určenými k ortorektifikaci nebo ke zgeoreferencování. Nicméně drtivá většina těchto studií pracuje pouze se současnými daty, resp. mezi podkladovými daty a rastry určenými ke zpracování je rozdíl v pořízení maximálně několik let. V této práci jsou však použity archivní LMS až ze 30. let 20. stol. Právě změny v krajině jsou však velmi důležitým faktorem, který komplikuje párování význačných bodů mezi cílovým a referenčním obrazem. Jedním z důvodů pro selhání SIFT operátoru, který se jinak pro tyto účely využívá (např. Oh, Toth, Grejner-Brzezinska (2010)), jsou právě významné změny v krajině zachycené na velmi starých LMS použitých v této práci (30. léta).

Díky georeferencím archivních LMS lze bez problému užít obrazovou korelaci, jelikož je zajištěna stejná orientace rastrů a také lze obrazy převzorkovat do stejného prostorového rozlišení, za účelem vytvoření stejných rastrových mřížek. Výhody operátoru SIFT jako je invariance vůči stočení a změně měřítka jsou tak pro tyto aplikace v podstatě bezvýznamné. Identické body v podstatě nemá smysl hledat mimo zástavbu, jelikož docházelo ke scelování polí, a tudíž je jejich párování v těchto oblastech nemožné. Vyhodnocení úspěšnosti operátorů samozřejmě vycházelo z testování v zástavbě, a to konkrétně v území Lysé n./L. a Milovic (Obrázek 18 v metodice). Testování bylo zároveň provedeno s chybnými georeferencemi dodanými ČÚZK. Nicméně i přesto slouží k relativnímu srovnání operátorů mezi sebou. Z testování jednoznačně vyšla metodika založená na obrazové korelaci jako nejúspěšnější. Operátor SIFT by pravděpodobně bylo zajímavé přetestovat s opravenými georeferencemi a s vytvořenou maskou zástavby pro referenční podklad. Samotná knihovna OpenCV masky podporuje.

Z časových a dalších praktických důvodů toto bohužel již nebylo možné, jelikož bylo velmi mnoho času věnováno testování metodiky na LMS s chybnými georeferencemi. Testováno bylo množství různých metodických variant s různým nastavením parametrů operátorů, nicméně nic nevedlo k úspěchu a kapitola „4.3 Volba operátoru pro detekci a párování identických bodů“ již obsahuje pouze souhrn tohoto intenzivního testování. Autor si na počátku vytyčil cíl vytvořit automatizovanou metodiku na míru archivním LMS, tak jak je poskytl ČÚZK a z tohoto důvodu dlouhou dobu nebyly georeference považovány za vadné. Chyba ve vstupních datech byla

prověřována až po několikátém důkladném zkontrolování všech ostatních částí metodiky a takovýto problém nebyl vůbec předpokládán. Autor zároveň dlouhou dobu neměl odvahu zpochybnit správnost georeferencí ČÚZK, ačkoliv nad abnormálně velkými oblastmi k prohledávání a přesností georeferencí panovaly samozřejmě od počátku pochyby. Kvůli frustrujícím výsledkům, které vycházely z testování operátorů, byla také rozdělena metodika na příliš velké množství skriptů (celkem 10), s cílem odhalit jakoukoliv nesrovnalost v implementaci metodiky, neboť v kratších a jednodušších skriptech by se snáze dala odhalit chyba. Samozřejmě by jinak dávalo smysl zredukovat množství skriptů a spojit jejich funkčnost dohromady.

Z testování vycházely naprosto katastrofální a frustrující výsledky z velké části způsobené neúměrně velkými oblastmi k prohledávání (ilustrace problému – Obrázek 36). Tyto oblasti k nalezení odpovídajícího bodu měly např. pro prezentovaný snímek (Obrázek 36 – Kobylí 1938) více než 52 krát větší plochu k prohledávání (650 m vs 90 m hrana čtverce prohledávaných oblastí) z důvodu špatných georeferencí. Výsledky byly naprosto nepřijatelné a zcela nepoužitelné. Na testované datové sadě (Milovice 1938) s chybnými georeferencemi byla vyzkoušena celá automatizovaná metodika s obrazovou korelací. Nicméně ani RANSAC jinak považovaný za robustní metodu nedokázal odfiltrovat chybně spárované body. Po opravě georeferencí se podařilo danou sadu snímků zpracovat a vygenerovat ortomozaiku, což zcela jednoznačně dokazuje, že vadné georeference jsou překážkou pro spolehlivý automatizovaný sběr VLB.

Chybné georeference byly odhaleny až v pokročilé fázi zpracování práce, proto již některé přístupy a metodické varianty nemohly být z časových důvodů otestovány. Cílem práce zároveň původně vůbec nebylo řešit georeferencování LMS, bohužel pro umožnění splnění cílů práce (automatizovaný sběr VLB) bylo nutné vypořádat se i s tímto problémem. Georeference ČÚZK lze zároveň označit za metodicky chybné, a proto není žádoucí dále zpracovávat a testovat taková data z odborného pohledu. Pracoviště ČÚZK zabývající se zpracováním archivních LMS by mělo přejít na korektní způsob georeferencování, zejména co se týče přesnosti sběru VLB a pravděpodobně také i použití lepší transformace (podobnostní, resp. Helmertova transformace). V práci byly georeference tvořeny s pomocí 3 až 5 přesně sebraných VLB a bylo dosaženo RMSE v řádu metrů (213 m ČÚZK vs 7 m opravené georeference RMSE u X souřadnice viz Tabulka 3 v metodice).

V kapitole 4.5 „Automatizované vyhledávání VLB“ bylo vyzkoušeno několik různých parametrů, které teoreticky měly potenciál ovlivnit kvalitu výsledu. Skutečně relevantní byl pouze typ

podkladového ortofota a velikost plošky reprezentující křižovatku. Zejména typ převzorkování rastrů a použité pásmo současného ortofota (RGB2grey/zelené p.) měly zanedbatelný vliv na výsledek. Právě následkem naprostých neúspěchů s testováním LMS s vadnými georeferencemi byly testovány tyto marginální parametry s cílem objevit jakoukoliv příčinu potenciálního neúspěchu. Vynaložený čas samozřejmě mohl být věnován relevantnější problematice. K LMS ČÚZK poskytuje také stopy snímků a prvky EO. Vzhledem k vadným georeferencím budou chybné i prvky EO, neboť vychází ze samotných georeferencí. Dodané prvky EO však nebyly použity jako vstup pro žádnou část metodiky. Samozřejmě byly vypočteny s pomocí automatizovaně nalezených VLB při zpracování ortofota v MicMacu.

Obrázek 36: Vizualizace přesnosti georeferencí – Kobylí 1938

Relativní srovnání velikosti oblasti nutné k prohledávání v archivním LMS Kobylí 1938 pro křižovatku (kontrolní bod č. 10 viz mapa výše) v měřítku 1 : 10 000:

LMS: georeference ČÚZK – 650 m

LMS: georeference opravené – 90 m

Současné ortofoto: velikost plošky reprezentující křižovatku – 51 m

Zdroj: vlastní tvorba/podkladová data ČÚZK (2022a)

Je nutné také podotknout, že veškeré testování a vzájemné porovnávání operátorů (kapitola 4.3 Volba operátoru pro detekci a párování identických bodů) bylo provedeno na snímku z území Milovic 1938. Vygenerovaná ortomozaika pro Milovice 1938 však měla ve výsledku nejhorší přesnost (Tabulka 7 ve výsledcích) kvůli největším změnám v krajině ve srovnání s ostatními sadami snímků. Účelem výběru tohoto nejobtížnějšího území pro testování operátorů bylo sestavit metodiku odolnou pro jakoukoliv situaci. Je tak pravděpodobné, že při využití jiných testovacích snímků použitých v práci by úspěšnost operátorů SIFT, SURF, ORB byla vyšší.

6.2 Další komentáře k metodice

Některé snímkové sady z 80. let mají špatnou kvalitu způsobenou kopírováním originálních negativů na jiný nehořlavý materiál (Šafář, Tlapáková 2016). VGHMÚř v Dobrušce však zvolil takové snímky a roky snímání, které nemají popsaný defekt a z hlediska ostrosti jsou kvalitní. Tyto snímky získané zvláštní objednávkou však neprošly zpracováním ČÚZK a nemají georeferenci (nemají World files) či přibližně vypočtené prvky EO.

Na ortofotu Cenia z 50. let i na LMS ze 30. let jsou pole, a tedy i cestní síť ve stavu před zcelováním polí v 50. letech. Existoval tedy předpoklad, že by toto staré ortofoto mohlo být pro vyhledávání VLB ve snímcích z 30. let vhodnější. Na základě testování (viz Tabulka 5 v metodice) bylo však rozhodnuto pro 30. i 80. léta použít ortofoto současné, jelikož se nepotvrdilo, že by pro 30. léta bylo ortofoto Cenia lepší. Důležitým faktorem také pravděpodobně byla kvalita podkladového ortofota (ostrost, zašumění apod.). Ta byla viditelně horší u ortofota Cenia z 50. let oproti ortofotu současnému. Pro další zvýšení úspěšnosti párování identických bodů by se daly použít např. starší verze databáze ZABAGED a referenčního ortofota. ČÚZK poskytuje ortofoto snímané od roku 1998 i jako WMS službu. Snahou bylo použít vždy hlavně křižovatky silnic, v případě nedostatečného pokrytí byly užity také křižovatky ulic (Milovice 1938). U snímků z 30. let bylo problémem nerovnoměrné pokrytí VLB, neboť tyto snímky měly menší formát a zabíraly výrazně menší oblast.

Oblasti k prohledávání byly voleny spíš větší, aby byly spárovány VLB i na okraji snímků, kde bývá typicky horší polohová přesnost. Tyto okrajové VLB měly totiž velký význam pro propojení stereo dvojic při zpracování v MicMacu. Na druhou stranu příliš velké oblasti k prohledávání způsobují chybné párování VLB. U Milovic 1989 byly zvoleny velké oblasti k prohledávání s hranou čtverce 440 m. Takto velké oblasti díky přesným georeferencím samozřejmě nebyly potřeba, cílem však bylo otestovat metodiku v situacích s obtížnými

podmínkami a simulovat metodicky vadné georeference ČÚZK. I s takto velkými oblastmi k prohledávání se podařilo pro Milovice 1989 vygenerovat kvalitní ortomozaiku. Nutno připomenout, že Milovice 1938 bylo možné zpracovat až po opravě georeferencí, s chybnými georeferencemi nedokázal odlehlá měření odfiltrovat ani RANSAC.

Vyšší korelační koeficient u výstupních bodů nutně neznamenal vyšší úspěšnost párování (danou RMSE chybou), což dobře ilustruje Tabulka 5. Velké množství bodů na výstupu z obrazové korelace pro každou křižovatku se ukázalo být spíš kontraproduktivní. Pro obě území a období (30. i 80. léta) bylo vyexportováno pouze 5 potenciálních VLB s nejvyšším korelačním koeficientem pro každou křižovatku. Body byly následně shlukovány ve skriptu `6step_flood_collapse_clusters.py`. Větší množství výstupních VLB se ukázalo jako nežádoucí, neboť bylo výrazně obtížnější najít správný geometrický model v RANSACu a zanášela se chyba do finálních VLB.

Požadavek na export mnoha bodů původně vyplynul ze špatných georeferencí. Jelikož byly prohledávané oblasti výrazně větší, bylo obtížnější nalézt odpovídající bod. Pro zvýšení pravděpodobnosti nalezení dané křižovatky bylo rozhodnuto exportovat více než 1 VLB pro každou křižovatku. Nicméně po opravě georeferencí se významně zvýšila úspěšnost párování a pravděpodobně by stačilo exportovat pro každou křižovatku pouze 1 VLB s nejvyšším korelačním koeficientem a skript `6step_flood_collapse_clusters.py` by nebylo vůbec třeba programovat. I přesto má svůj význam, neboť se typicky zvýšila přesnost nalezení středu křižovatky díky shlukovacímu algoritmu.

RANSAC algoritmus musel být naprogramován od algoritmické úrovně, jelikož OpenCV obsahuje RANSAC pouze v kombinaci s 2D rovinnými transformacemi. Obdobně spymicmac obsahuje pouze rovinnou afinní transformaci (McNabb 2023). V RANSACu byla použita implementace DLT, která je k dispozici na GitHubu (Ankita 2019).

Limit vzdálenosti v RANSACu měl velký vliv na rovnoměrnost rozložení a počet výstupních VLB ve snímku. Pokud byl limit nastaven příliš přísně, výstupních VLB nebyl dostatek a VLB byly rozmístěny nerovnoměrně, resp. nedocházelo k propojení stereo dvojic s pomocí VLB, což bránilo následnému zpracování ortofota. V rovinném území je možno jako geometrický model v RANSACu použít pouze 2D projektivní transformaci (viz Milovice 1989 v této práci) místo 3D DLT. Z rychlého testování vyplynulo, že v rovinném území byly zvoleny v podstatě tytéž body, u DLT byly rovnoměrněji rozloženy a bylo jich o něco více.

Z dalších způsobů pro odfiltrování odlehlých měření se nabízí například využití epipolární geometrie, která umožňuje stanovit vztah mezi snímky ve stereo dvojicích. Tento vztah popisuje tzv. fundamentální matice F. Pokud existuje bod x1 na snímku s1, pak tento bod musí ležet na snímku s2 na přímce p2 = F * x1 (Szeliski 2022). Stanoví se prahová vzdálenost V, pokud by byla vzdálenost bodu x2 od přímky p2 větší než V, byl by bod vyřazen z dalších výpočtů. Toto by bylo provedeno mezi všemi stereo dvojicemi. Samotná fundamentální matice se vypočítá právě z bodů nalezených v obou snímcích (třeba SIFT operátorem) tzv. eight-point algoritmem (Szeliski 2022). Je však otázkou, zda by byl tento kontrolní mechanismus přínosem, neboť by teoreticky kvůli principu obrazové korelace docházelo k nalezení VLB na stejném (špatném) místě v obou snímcích ve stereodvojici.

Spíše než fyzickogeografické podmínky (převýšení terénu), činily problém socioekonomické faktory – změny v krajině, které byly obzvlášť významné u snímků z 30 let hlavně v území Milovic. U datových sad z 30. let bylo proto dosaženo horší přesnosti. Snímky z 30. let byly zároveň menšího formátu a pokrývaly menší území. V důsledku toho v nich bylo možno nalézt méně VLB (problém při propojování stereo dvojic) s horším rozložením v prostoru (nerovnoměrné rozložení VLB oproti 80. létům).

Použitá metodika vychází z postupu Höhle, Potůčkové (2001), kteří použili obrazovou korelaci a křižovatky silnic jakožto VLB. Na rozdíl od zmíněné studie je však v metodice této práce využito archivních LMS, shlukovacího algoritmu, a navíc ještě robustní metoda RANSAC s DLT pro odfiltrování odlehlých měření. Také byla otestována rovnou 2 území ve 2 časových obdobích. V Milovicích došlo od 30. let k velmi významným změnám v krajině. Zároveň je celá práce postavena pouze na open-source SW.

Jedním z dalších metodických řešení, které se nabízí je kombinace operátoru SIFT použitého pouze jako detektor význačných bodů a následně vyhledání odpovídajícího bodu v archivním LMS s pomocí obrazové korelace, jelikož SIFT zcela selhával při párování odpovídajících si bodů mezi archivním LMS a současným ortofotem. Takto by bylo možno vynechat z metodiky vektorová data (křižovatky komunikací). Zároveň by bylo nutné použít masku zástavby. Následně by byl použit shlukovací algoritmus a RANSAC.

6.3 Zpracování ortofota

Výsledné ortomozaiky byly zpracovány v SW MicMac, který se ovládá z příkazové řádky a je tak určen spíše pokročilejším uživatelům. Při práci s programem a dostupnými materiály k SW se bohužel lze setkat s francouzštinou, což komplikuje zpracování výstupů. Zároveň se podařilo úspěšně vygenerovat ortofotomozaiku s pomocí tutoriálu dostupného přímo na stránkách MicMacu (Girod 2022). Ortomozaika je pouze mírně neostrá po okrajích. Řezové linie jsou trochu znatelné. Funkce SW mají typicky mnoho parametrů, které se dají uživatelsky nastavit. Při použití modulu Tapioca, který slouží pro vyhledávání spojovacích bodů, selhal postup „MulScale“, kdy by měl SW sám sestavit stereo dvojice. Postup doporučuje Girod (2022). Místo toho byl použit postup „File“. V tomto případě je nutné stereodvojice specifikovat ručně v konfiguračním xml souboru. Ve zmíněném tutoriálu (Girod 2022) jsou přítomny chyby například v názvech modulů (`SaisieAppuisInit` místo `SaisieAppuisInitQT`). MicMac je zároveň stavěný na ruční sběr VLB s pomocí speciálního modulu. Tento modul vygeneruje xml soubor s proprietární strukturou. Pomocí reverzního inženýrství byla zjištěna struktura tohoto souboru a následně byl naprogramován skript `8step_GCP2MicMac.py` pro uložení VLB z automatizovaného postupu do xml souboru se strukturou kompatibilní s SW MicMac. Tento skript zároveň zahrnuje transformaci rastrových souřadnic VLB (převzorkované LMS na 1 m) z automatizovaného postupu do LMS v původním rozlišení, tak jak je poskytuje ČÚZK a následně další transformaci do snímků převzorkovaných po stanovení vnitřní orientace s pomocí rámových značek.

Z hlediska geometrické přesnosti jsou všechny mozaiky velmi dobré (viz Tabulka 7 ve výsledcích). Ortomozaiku pro Kobylí 1984 s RMSE 0,74 m lze považovat za výbornou. Slabší je ortomozaika pro Milovice 1938 (RMSE 5,55 m), kde byly nejdrastičtější změny v krajině. Například ortofoto Cenia z 50. let má přesnost cca 1,5 m pro poválečné a 1,3 m pro předválečné snímky. Zájmová území v této práci by měla být pokryta snímky z let 1953 a 1954 (Sukup 2010).

7 Závěr

V práci bylo otestováno několik různých operátorů a metodických přístupů pro automatizovaný sběr VLB v archivních LMS, přičemž byla použita současná data jako referenční podklad. Z testování operátorů vyplynula obrazová korelace jako nejúspěšnější. Ačkoliv existují práce zabývající se automatizovaným sběrem VLB, v drtivé většině případů je časový odstup mezi referenčními a cílovými daty pouze několik let. Jen výjimečně se vyskytují skutečně archivní data. Téma bylo podpořeno vedením ČÚZK, neboť má zmíněná instituce zájem o výsledky práce. Z tohoto důvodu byla poskytnuta data nad rámec běžné studentské licence zdarma.

Velkým problémem se ukázaly být dodané archivní LMS s georeferencemi, které zpracovává ČÚZK. Tyto georeference byly chybné, odchylky se pohybovaly v řádu vyšších desítek metrů (nejhorší polohová odchylka u kontrolního bodu i více než 250 m), což je zcela nepřípustné při georeferencování LMS. Z tohoto důvodu musely být georeference opraveny, což původně vůbec nebylo cílem práce. Sada snímků pro Milovice 1938 byla testována i s chybnými georeferencemi ČÚZK, nicméně ani po filtraci metodou RANSAC se nepodařilo získat spolehlivé VLB.

Veškeré cíle práce se podařilo splnit, jelikož byla naprogramována plně automatizovaná metodika pro vyhledávání VLB, která byla realizována sadou několika skriptů v jazyce Python 3 s využitím pouze open-source knihoven. Vstupem pro metodiku jsou archivní LMS, současná podkladová data (ortofoto, DMP a vektorová vrstva silnic z databáze ZABAGED), výstup pak tvoří *.geojson soubory s VLB, které lze vizualizovat v QGISu a následně také *.xml soubor s proprietární strukturou importovatelný do SW MicMac.

Výsledné ortomozaiky byly vytvořeny právě v programu MicMac. Na závěr byla zhodnocena přesnost výstupů s pomocí RMSE, která je pro všechny sady snímků velmi dobrá. Za slabší lze označit pouze Milovice 1938, kde však došlo k významným změnám v krajině, což velmi komplikovalo hledání spolehlivých VLB. Odchylka ostatních mozaiek byla rámcově srovnatelná s ortofotem Cenia z 50. let, případně byla i lepší (Kobylí 1984).

S archivními LMS pracují např. Hodač, Zemánková (2018), Šafář, Tlapáková (2016) a Řáda (2016). Ve všech případech se však jedná pouze o ryze manuální zpracování archivního ortofota. Také využití MicMacu je v českém prostředí poměrně ojedinělé. Nebyla nalezena žádná jiná práce, ve které by byl tento SW použit pro generování ortofota. Program je jinak v akademickém prostředí oblíben a existuje množství zahraničních studií, kde je využit.

8 Zdroje literatury

AISHWARYA, S. (2023): SIFT Algorithm | How to Use SIFT for Image Matching in Python (Updated 2023), https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/ (17. 2. 2024).

ANKITA, V. (2019): DLT, https://github.com/acvictor/DLT (12. 4. 2023).

BARBORKA, P. (2016): Analýza metod pro detekci příznaků v digitalizovaném obraze. Fakulta aplikovaných věd Katedra kybernetiky, Západočeská univerzita v Plzni.

BAYER, T. (2020): Kartometrické analýzy starých map (prezentace – Historická kartografie), https://web.natur.cuni.cz/~bayertom/index.php/teaching/historicka-kartografie (13. 2. 2024).

BAYER, T. (2021): Grafové algoritmy I., https://web.natur.cuni.cz/~bayertom/index.php/teaching/historicka-kartografie (14. 2. 2024).

BERVEGLIERI, A., TOMMASELLI, A. M. G. (2014): Multi-scale matching for the automatic location of control points in large scale aerial images using terrestrial scenes. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 3W1, 40, 27–31.

BERVEGLIERI, A., TOMMASELLI, A. M. G. (2017): Automatic Measurement of Control Points for Aerial Image Orientation. Photogrammetric Record, 158, 32, 141–159.

BURT, J. E., WHITE, J., ALLORD, G., THEN, K. M., ZHU, A. X. (2020a): Automated and semi-automated map georeferencing. Cartography and Geographic Information Science, 1, 47, 46–66.

BURT, J. E., WHITE, J., ALLORD, G., THEN, K. M., ZHU, A. X. (2020b): Automated and semi-automated map georeferencing. Cartography and Geographic Information Science, 1, 47, 46–66.

CENIA (2020): Historická ortofotomapa ze 30. let , https://www.cenia.cz/2020/12/08/historicka-ortofotomapa-z-let-1933-1938/ (14. 10. 2023).

CENIA (2022): Historická ortofotomapa z 50.let se otevírá, https://www.cenia.cz/2022/02/17/historicka-ortofotomapa-z-50-let-se-otevira/ (14. 10. 2023).

CLÉRI, I., PIERROT-DESEILLIGNY, M., VALLET, B. (2014): Automatic Georeferencing of a Heritage of old analog aerial Photographs. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, II–3, 33–40.

CRACIUN, D., LE BRIS, A. (2022): AUTOMATIC ALGORITHM FOR GEOREFERENCING HISTORICAL-TO-NOWADAYS AERIAL IMAGES ACQUIRED IN NATURAL ENVIRONMENTS. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives. International Society for Photogrammetry and Remote Sensing, 21–28.

ČÚZK (2022a): Ortofoto ČR, roky snímkování 2020 a 2021.

ČÚZK (2022b): Základní mapa České republiky 1:10 000, https://geoportal.cuzk.cz/(S(1sccdm25uec0kdtkmk5m0t2o))/Default.aspx?menu=223&mode=TextMeta&side=mapy10&text=dsady_mapy10 (10. 3. 2022).

ČÚZK (2023): Katalog produkce Zeměměřického úřadu. Praha.

DĚDKOVÁ, P. (2012): 3D VIZUALIZACE ZANIKLÉ OBCE A JEJÍ HODNOCENÍ Z HLEDISKA UŽIVATELSKÉ KOGNICE. Přírodovědecká fakulta, Univerzita Palackého, Olomouc, 57.

DIGITÁLNÍ ATLAS ZANIKLÝCH KRAJIN (2019): Militární a postmilitární krajina zrušeného VVP Milovice - Mladá, https://web.natur.cuni.cz/sekce-gr/zaniklekrajiny/atlas/modelova-uzemi/2019/milovice (12. 10. 2023).

DIGITÁLNÍ ATLAS ZANIKLÝCH KRAJIN (2020): Vinohradnická krajina jižní Moravy - Kobylí, Vrbice, https://web.natur.cuni.cz/sekce-gr/zaniklekrajiny/atlas/modelova-uzemi/2020/kobyli (12. 10. 2023).

ELZNICOVÁ, J. (2008): Zpracování archivních leteckých snímků pro identifikaci změn rozšíření agrárních valů během 20. století. Fakulta životního prostředí UJEP, 1–8.

ESRI (2024): World files for raster datasets, https://pro.arcgis.com/en/pro-app/3.1/help/data/imagery/world-files-for-raster-datasets.htm (14. 2. 2024).

FEURER, D., VINATIER, F. (2018): Joining multi-epoch archival aerial images in a single SfM block allows 3-D change detection with almost exclusively image information. ISPRS Journal of Photogrammetry and Remote Sensing, 146, 495–506.

GIROD, L. (2022): Historical Orthoimage, https://micmac.ensg.eu/index.php/Historical_Orthoimage (25. 11. 2023).

GOBBI, S., MAIMERI, G., TATTONI, C., CANTIANI, M. G., ROCCHINI, D., LA PORTA, N., CIOLLI, M., ZATELLI, P. (2018): Orthorectification of a large dataset of historical aerial images: Procedure and precision assessment in an open source environment. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 4W8, 42, 53–59.

GONÇALVES, J. A. (2016): Automatic orientation and mosaicking of archived aerial photography using structure from motion. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives. International Society for Photogrammetry and Remote Sensing, 123–126.

GOTOVAC, D., KRUŽI´C, S. K., GOTOVAC, S., PAPI´C, V. P. (2017): A Model for Automatic Geomapping of Aerial Images Mosaic Acquired by UAV.

HARTMANN, W., HAVLENA, M., SCHINDLER, K. (2016): Recent developments in large-scale tie-point matching. ISPRS Journal of Photogrammetry and Remote Sensing, 115, 47–62.

HAYDER DIBS, SHATTRI MANSOR, NOORDIN AHMAD, BISWAJEET PRADHAN, NADHIR AL-ANSARI (2020): Automatic Fast and Robust Technique to Refine Extracted SIFT Key Points for Remote Sensing Images. Journal of Civil Engineering and Architecture, 6, 14.

HODAČ, J., ZEMÁNKOVÁ, A. (2018): Historical orthophotos created on base of single photos - specifics of processing. The Civil Engineering Journal, 3, 14.

HÖHLE, J., POTŮČKOVÁ, M. (2001): Towards the Full Automatic Production of Orthoimages. Photogrammetrie, 6/2001, 397–404.

CHEN, H. R., TSENG, Y. H. (2016): Study of automatic image rectification and registration of scanned historical aerial photographs. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, July, 41, 1229–1236.

CHUREESAMPANT, K., SUSAKI, J. (2014): Automatic GCP extraction of fully polarimetric SAR images. IEEE Transactions on Geoscience and Remote Sensing, 1, 52, 137–148.

IGN (2022): MicMac, Apero, Pastis and Other Beverages in a Nutshell!, https://github.com/micmacIGN/Documentation/blob/master/DocMicMac.pdf (2. 12. 2023).

JEŽEK, B. (2021): Vyplňování oblastí – studijní text k předmětu Počítačová grafika, FIM UHK.

KARDOŠ, M., TUČEK, J., SLATKOVSKÁ, Z., TOMAŠTÍK, J. (2017): Historická ortofoto mapa Slovenska – analýza polohovej presnosti a jej aplikácie pri edintifikácii parciel. Kartografické listy, 1, 25, 37–47.

KOVÁŘ, P. (2021): Úvod do Teorie grafů. Vysoká škola báňská – Technická univerzita Ostrava a Západočeská univerzita v Plzni, Plzeň.

LI, Q., WANG, G., LIU, J., CHEN, S. (2009): Robust Scale-Invariant Feature Matching for Remote Sensing Image Registration. IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, 2, 6.

LI, Y., BRIGGS, R. (2006): Automated Georeferencing Based on Topological Point Pattern Matching. The University of Texas at Dallas.

LIU, J., YU, X. (2008): RESEARCH ON SAR IMAGE MATCHING TECHNOLOGY BASED ON SIFT. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

LOWE, D. G. (2004): Distinctive Image Features from Scale-Invariant Keypoints.

MAREŠ, M., VALLA, T. (2017): Průvodce labyrintem algorimů. CZ.NIC, Praha.

MCNABB, R. (2023): automatically finding control points, https://spymicmac.readthedocs.io/en/latest/tutorials/micmac_processing/register.html (18. 3. 2024).

MICMAC (2017): Présentation, https://micmac.ensg.eu/index.php/Presentation (6. 3. 2024).
NAGARAJAN, S., SCHENK, T. (2016): Feature-based registration of historical aerial images by Area Minimization. ISPRS Journal of Photogrammetry and Remote Sensing, 116, 15–23.
NETELER, M., MITASOVA, H. (2004): Open Source GIS: A GRASS GIS Approach. Springer, Boston.
NURMINEN, K., LITKEY, P., HONKAVAARA, E., VASTARANTA, M., HOLOPAINEN, M., LYYTIKÄINEN-SAARENMAA, P., KANTOLA, T., LYYTIKÄINEN, M. (2015): Automation aspects for the georeferencing of photogrammetric aerial image archives in forested scenes. Remote Sensing, 2, 7, 1565–1593.
OBEC KOBYLÍ (2023): Kobylí – vinařská obec, https://kobyli.cz/ (12. 10. 2023).
OH, J., TOTH, C. K., GREJNER-BRZEZINSKA, D. A. (2010): Automatic Georeferencing of Aerial Images Using Stereo High-Resolution Satellite Images. ASPRS 2010 ANNUAL CONFERENCE.
OPEN CV (2024): Template Matching, https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html (24. 2. 2024).
PAVELKA, K. (2003a): Fotogrammetrie. Západočeská univerzita v Plzni, Plzeň.
PAVELKA, K. (2003b): Fotogrammetrie 10. Vydavatelství ČVUT, Praha.
PIERROT DESEILLIGNY, M., CLERY, I. (2012): Apero, an Open Source Bundle Adjusment Software for Automatic Calibration and Orientation of Set of Images. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, January 2011, XXXVIII-5/, 269–276.
PINTO, A. T., GONÇALVES, J. A., BEJA, P., HONRADO, J. P. (2019): From archived historical aerial imagery to informative orthophotos: A framework for retrieving the past in long-term socioecological research. MDPI AG, 11.
POTŮČKOVÁ, M. (2020): Fotogrammetrie III – výukové materiály. Přf UK, Praha.
RAMÓN, J., BATLLE, M. (2018): Digital photogrammetry of historical aerial photographs using open-source software. Autonomous University of Santo Domingo.
RUPNIK, E., DAAKIR, M., PIERROT DESEILLIGNY, M. (2017): MicMac – a free, open-source solution for photogrammetry. Open Geospatial Data, Software and Standards, 1, 2.
ŘÁDA, T. (2016): Tvorba digitálního ortofota z archivních snímků – obec Houstoň (Kladno). Fakulta stavební ČVUT, Praha, 92.
SÁDOVSKÁ, P. (2011): Vývoj urbanizovaného území na základě leteckých snímků. Přírodovědecká fakulta Univerzita Palackého, Olomouc, 62.
SOLEM, J. E. (2012): Programming Computer Vision with Python. O'Reilly Media, Sebastopol.
STACK OVERFLOW (2020): numpy.corrcoef() doubts about return value, https://stackoverflow.com/questions/61557443/numpy-corrcoef-doubts-about-return-value (15. 5. 2022).

SUKUP, K. (2010): Tvorba ortofotomap z historických leteckých snímků území ČR. Technická zpráva, Geodis Brno.

SZELISKI, R. (2022): Computer Vision: Algorithms and Applications, 2nd Edition. Springer.

ŠAFÁŘ, V., TLAPÁKOVÁ, L. (2016): Alternative Methods of the Processing of Archival Aerial Photos. Geodetický a Kartografický obzor, 104, 62, 253–257.

ŠMEJDA, L. (2009): Mapování archeologického potenciálu pomocí leteckých snímků. Typos, tiskařské závody, s.r.o., Plzeň.

VGHMÚŘ DOBRUŠKA (2022): Archiv leteckých měřických snímků, http://www.mapy.army.cz/historicke-lms (14. 3. 2022).

WIKIPEDIA (2023a): Kobylí, https://cs.wikipedia.org/wiki/Kobyl%C3%AD (12. 10. 2023).

WIKIPEDIA (2023b): Milovice, https://cs.wikipedia.org/wiki/Milovice (11. 10. 2023).

WIKIPEDIA (2024a): Random sample consensus, https://en.wikipedia.org/wiki/Random_sample_consensus (4. 3. 2024).

WIKIPEDIA (2024b): World file, https://en.wikipedia.org/wiki/World_file (30. 3. 2024).

ZHANG, L., RUPNIK, E., PIERROT-DESEILLIGNY, M. (2021): Feature matching for multi-epoch historical aerial images. ISPRS Journal of Photogrammetry and Remote Sensing, 182, 176–189.

ŽÁRA, J., BENEŠ, B., SOCHOR, J., FELKEL, P. (2005): Moderní počítačová grafika. Computer Press, Brno.

9 Přílohy

9.1 Přílohy vložené na konci práce

Každý skript v příloze dále má popsané uživatelsky modifikovatelné klíčové proměnné, které jsou na začátku skriptu (datové vstupy a parametry algoritmů). Příloha obsahuje pouze skripty, s pomocí kterých byl vytvořen finální automatizovaný postup. Neobsahuje proto skripty pro testování operátorů SIFT, SURF, ORB.

9.2 Externí přílohy

1. Složka se všemi použitými skripty v práci

Příloha 1: 0step_georeference2worldfile.py

Z důvodu umožnění publikace skriptů zde v příloze práce jsou některé řádky zalomeny nepříliš vhodným způsobem. Skripty jsou však zveřejněny i v externí příloze práce. Další popis skriptů je uveden v metodice.

Parametry skriptu:

```
inp_ptn_path = cesta k adresáři se soubory *.points obsahující VLB
encoding = kódování znaků, nejčastěji "utf-8"
lines2skip = počet řádků v *.points souborech, které neobsahují
souřadnice VLB, defaultně 2 při výstupu z QGIS
transform = druh transformace pro georeferencování, helmert / affine
```

```
#################################################################
import geopandas
import glob
from functions import *
from pathlib import Path
import pyfiglet
import numpy as np

#################################################################
# points only for *.tif and *.jpg images are accepted!
# only QGIS output GCP from georeferencing with default names are
accepted!!! Raster origin in left top corner, but NEGATIVE y coords.
# script tested with EPSG:5514 --> S-JTSK with negative coords and QGIS
point input (negative y raster coords)

inp_ptn_path =
"MILOVICE1989_SKRIPT_TEST/Milovice1989_rotated_tfw/*.points"

encoding = "utf-8" # encoding of *.points files
lines2skip = 2 # number of lines without data (at the top of the file,
usually 2 for points from QGIS)
transform = "helmert" # helmert / affine        helmert trans. with
minimum points = similarity trans.
#################################################################
gcp_files = glob.glob(inp_ptn_path)

for file in gcp_files:
    #print(file)
    with open(file, mode="r", encoding=encoding) as f:
        data = f.read().splitlines() # list lines

    pure_data = data[lines2skip:]

    pix_top_list = []
    pix_bottom_list = []
    ref_top_list = []
    ref_bottom_list = []

    for line in pure_data:
        splited = line.split(",") # line splited by given character
```

```
        # print(splited)
        # problems with different orientation of axes sometimes occurs

        if transform == "helmert":
            pix_top_list.append([float(splited[2]), -float(splited[3]),
1, 0])
            pix_bottom_list.append([float(splited[3]), float(splited[2]),
0, 1])

        elif transform == "affine":
            pix_top_list.append([float(splited[2]), float(splited[3]), 0,
0, 1, 0])
            pix_bottom_list.append([0, 0, float(splited[2]), -
float(splited[3]), 0, 1])

        ref_top_list.append(float(splited[0])) # X ref
        ref_bottom_list.append(float(splited[1])) # Y ref

    coefs = least_square_solver(pix_top_list + pix_bottom_list,
ref_top_list + ref_bottom_list)
    print(coefs)

    if coefs.size == 4: # helmert trans. use just 4 coefs, its necessary
to upgrade to 6 for world file. 2 params are same.
        coefs = np.array([coefs[0], -coefs[1], coefs[1], -coefs[0],
coefs[2], coefs[3]])

    ptn_path = Path(file)
    #print(ptn_path.stem, ptn_path.name, ptn_path.parent.absolute(),
ptn_path.suffix, ptn_path.resolve(), ptn_path.resolve().parent)
    img_path = Path(str(ptn_path.parent.absolute()) + "/" +
str(ptn_path.stem))
    out_path = img_path.parent.absolute()
    out_name = img_path.stem
    img_suffix = img_path.suffix

    if img_suffix == ".jpg":
        world_file_suffix = ".jgw"
    else:
        world_file_suffix = ".tfw"

    print(transform + " coefs of " + str(out_name) + ":", coefs)
    save_tfw(str(out_path) + "/" + str(out_name) + world_file_suffix,
coefs[0], -coefs[1], coefs[2], coefs[4], coefs[5], pixel_size_y=coefs[3])
# adding minus to coef to get it worked... Different orientation of
angles???

print(pyfiglet.figlet_format("Successfully executed ! :-)",
font="slant"))
```

Příloha 2: 1step_warp_rasters.py

Parametry skriptu:

```
source_files = cesta k adresáři se zdrojovými rastry
out_directory = cesta k adresáři pro výstupní rastry
resolution = prostorové rozlišení v metrech
resampling = převzorkování - "near" / "cubic" / "bilinear"
output_format = výstupní formát .tif nebo .jpg
```

```
#################################################################
from functions import warp
import glob
from pathlib import Path
import pyfiglet

#####################################################################
# inputs:

source_files = "MILOVICE1989_SKRIPT_TEST/Milovice1989_rotated_tfw/*.tif"

out_directory = "MILOVICE1989_SKRIPT_TEST/Milovice1989_1m/" # with
backslash

resolution = 1 # in meters
resampling = "bilinear" # near / cubic / bilinear
output_format = ".tif" # .tif  or .jpg
#####################################################################

prefix = str(resolution) + "m_"
source_list = glob.glob(source_files)
print(source_list)
for file in source_list: # list files in given directory with given
suffix
    p = Path(file)
    # testing pathlib library:
    print(p.stem, p.name, p.parent.absolute(), p.suffix, p.resolve(),
p.resolve().parent)
    out_path = out_directory + p.stem + output_format
    res = warp(resolution, file, out_path=out_path,
resampling=resampling, prefix=prefix, suffix=None,
format_out=output_format)
    del res

print(pyfiglet.figlet_format("Successfully executed ! :-)",
font="slant"))
```

Příloha 3: 2step_identify_road_crossing.py

Parametry skriptu:

```
road_path = cesta k vrstvě se sítí komunikací
dtm_path = cesta k DMP
id_attribute = atribut u road_path pro klasifikaci typů komunikací (jedná
se o název atributu - v konečné verzi skriptů není klasifikace využita)
out_path = cesta k výstupnímu souboru s křižovatkami silnic
out_edges = cesta k výstupnímu souboru s hranami vytvořeného grafu
atr_Z_name = atribut, který obsahuje nadmořskou výšku u výstupních VLB
(název atributu)
atr_cross_id = atribut, který obsahuje ID křižovatky u výstupních VLB
(název atributu)
limit = minimální počet komunikací vycházející z jedné křižovatky nutný
pro export dané křižovatky, defaultně 2 a více
```

```
#################################################################
import geopandas
import networkx as nx
import rasterio
import matplotlib.pyplot as plt
from functions import save_points_atr, save_edges, geo2ras, plot_img
import cv2
import pyfiglet

#################################################################
# inputs Milovice:
road_path = "MILOVICE1989_SKRIPT_TEST/silnice/splited_silnice_ulice.shp"
# 1989
dtm_path = "MILOVICE1989_SKRIPT_TEST/silnice/dmp_1g_2m.tif" # elevation
raster
id_attribute = "TYPSIL_K" # attribute of path type (just id or something)

# outputs:
out_path = "MILOVICE1989_SKRIPT_TEST/silnice/roads_crossing_1989.geojson"
# output road crossing
out_edges = "MILOVICE1989_SKRIPT_TEST/silnice/roads_edges_1938.geojson" #
output edges

atr_Z_name = "Z" # attribute with elevation name
atr_cross_id = "cross_id" # name of attribute with road crossing ID
limit = 2 # number of connected roads needed to export their road
crossing

#################################################################
# body of script:
print("Opening input roads, creating new graph.")
gdf = geopandas.read_file(road_path) # open roads
crs = gdf.crs # coordinate system of input
gdf = gdf.explode() # multipart geometries to singlepart
G = nx.Graph() # new graph

# round every coordinate to 2 decimal places
point_rounded = lambda p : (round(p[0],2), round(p[1], 2))
```

```
# list for edges
edges = []
print("Adding edges and nodes to graph.")
for idx, row in gdf.iterrows(): # iterate over lines
    coords = row.geometry.coords # coordinates of every line (road)
    id = row[id_attribute] # id of line (road)

    first_point = point_rounded(row.geometry.coords[0]) # first and last
point of line (road)
    last_point = point_rounded(row.geometry.coords[-1])

    G.add_edge(first_point, last_point) # add edge to graph
    G.edges[first_point, last_point]["id"] = id # add id to the edge
    edges.append((first_point, last_point)) # add edge to the list of
edges

print("Graph filled.")
print("Opening provided DTM.")
raster_dtm = rasterio.open(dtm_path)
dtm_data = raster_dtm.read(1) # first band (the only band)
ras_heigt, ras_width = raster_dtm.shape # Yrow, Xcol

print("Provided DTM info:", raster_dtm.profile)

Z = []
points = [] # list for resulting road crossing
# iterate over nodes
# append nodes with number of edges more than limit
for node in list(G.nodes):
    containing_edges = G.edges(node) # list of edges, which are connected
to given node
    if len(containing_edges) >= limit: # just nodes with more than limit
edges are wanted
        row_Y, col_X = geo2ras(raster_dtm, node[0], node[1])
        # if node lies in raster:
        if (row_Y >= 0) and (col_X >= 0) and (ras_width > col_X) and
(ras_heigt > row_Y):
            elevation = round(float(dtm_data[row_Y, col_X]), 3)
            points.append(node)
            Z.append(elevation)
        else: # outliar point
            print("Point with coordinates " + str(node[0]) + " " +
str(node[1]) +
            " lies out of provided DTM, can't get Z value.")

save_edges(edges, out_edges) # save edges of graph
save_points_atr([], [], out_path, [Z], [atr_Z_name], points, crs=crs,
new_idx=True, idx_name=atr_cross_id) # save road crossing

print(pyfiglet.figlet_format("Successfully executed ! :-)",
font="slant"))
```

Příloha 4: 3step_intersect_road_crossing.py

Parametry skriptu:

```
current_orto_path = cesta k adresáři s listy referenčního ortofota
road_cross_path = cesta k vrstvě obsahující křižovatky silnic
road_cross_path_intersect = cesta k výstupnímu souboru s křižovatkami
silnic (intersect s bounding boxy referenčního ortofota)
out_orto_rec = cesta k výstupnímu souboru s polygony (bounding boxy)
referenčního ortofota
crs = použitý souřadnicový systém pro všechny vrstvy, např. "epsg:5514"
```

```
#################################################################
import glob
import pathlib
import rasterio
import os
from functions import *

#########################################################################
#####
# *.jpg for listing all jpg files
# inputs Milovice 1989
current_orto_path = "MILOVICE1989_SKRIPT_TEST/current_orto_1m/*.tif"
road_cross_path =
"MILOVICE1989_SKRIPT_TEST/silnice/roads_crossing_1989.geojson"
# outputs
road_cross_path_intersect =
"MILOVICE1989_SKRIPT_TEST/silnice/roads_crossing_intersect_1989.geojson"
out_orto_rec =
"MILOVICE1989_SKRIPT_TEST/current_orto_1m/ortho_rec.geojson"

crs="epsg:5514" # coordinate system !!! must be same for all layers
#########################################################################
#####

homedir = pathlib.Path.cwd() # home directory
print("\n current home directory:", homedir)
# in_dir = cwd / "milovice"  # move to milovice directory

rasters = []
atr_values = [[],[],[]]
# list files in given directory with given suffix and iterate over them
for file in glob.glob(current_orto_path):
    raster = rasterio.open(file)
    p = Path(file)
    relative_path = os.path.relpath(file)
    raster_name = p.name
    bbox = raster.bounds
    absolute_path = str(p.resolve())

    # testing pathlib library:
    print("\n", p.stem, p.name, p.parent.absolute(), p.suffix,
p.resolve(), p.resolve().parent)
    print(relative_path, raster_name, bbox)
    rasters.append(bbox)
```

```
    atr_values[0].append(relative_path)
    atr_values[1].append(raster_name)
    atr_values[2].append(absolute_path)

# create ortho bounding box
create_bbox(in_path=None,
            out_path=out_orto_rec,
            distance=None,
            atr_values=atr_values,
            names=["path_rel", "name_ras", "path_absolute"],
            bboxes=rasters)

# intersect on roads crossing and ortho rectangles
ortho_rec = geopandas.read_file(out_orto_rec)
road_cross = geopandas.read_file(road_cross_path)
# set crs
ortho_rec = ortho_rec.set_crs(crs, allow_override=True)
road_cross = road_cross.set_crs(crs, allow_override=True)

# probably deprecated according stackowerflow:
# intersect = ortho_rec.overlay(road_cross, how="intersection")
# intersect = road_cross.intersection(ortho_rec)

intersect = geopandas.overlay(ortho_rec, road_cross, how='intersection',
keep_geom_type=False)

if road_cross_path_intersect[-3:] == "shp":
    driver = "ESRI Shapefile"
    encoding = "windows-1250"
elif road_cross_path_intersect[-7:] == "geojson":
    driver = "GeoJSON"
    encoding = "UTF-8"

intersect.to_file(road_cross_path_intersect, driver=driver,
encoding=encoding)

print("Done :-)")
```

Příloha 5: 4step_extract_patches_offline.py

Parametry skriptu:

```
road_crossing_intersect = cesta ke vstupní vrstvě s křižovatkami silnic
(intersect křižovatek s bounding boxy referenčního ortofota z předchozího
skriptu)
atr_path_rel = atribut s relativními cestami k příslušnému listu
referenčního ortofota (název atributu), který obsahuje danou křižovatku
vrstvy road_crossing_intersect, ve výchozím stavu "path_rel"
atr_cross_id = atribut, který obsahuje ID křižovatky (název atributu),
defaultně "cross_id"
patches_path = cesta k adresáři s výstupními ploškami reprezentujícími
křižovatky
out_orto_rec = cesta k výstupnímu souboru s polygony (bounding boxy) –
ploškami reprezentující křižovatky (vektorová data pro vizualizaci
výstupu)
road_size = velikost plošky reprezentující křižovatku v metrech (hrana
čtverce)
ortho_res = prostorové rozlišení ortofot v metrech
```

```
############################################################
import geopandas
import rasterio
import json
from functions import *
import cv2 as cv
import pyfiglet

#########################################################
# inputs
road_crossing_intersect =
"MILOVICE1989_SKRIPT_TEST/silnice/roads_crossing_intersect_1989.geojson"

atr_path_rel = "path_rel" # name of attribute with path to provided ortho
atr_cross_id = "cross_id" # name of attribute with road crossing ID

# outputs Milovice 1989
patches_path =
"MILOVICE1989_SKRIPT_TEST/1989_patches_1m_splited_silnice_ulice/" # with
slash at the end of path
out_orto_rec =
"MILOVICE1989_SKRIPT_TEST/1989_patches_1m_splited_silnice_ulice/1989_patc
hes.geojson" # rectangles with patches bounding box

road_size = 50  # meters /// edge of rectangle
ortho_res = 1 # pixel size of orthos
#########################################################

road_size_pix = m2pix(ortho_res, road_size)  # pixels
print("Extracting patches.")
print("road crossing size in meters:", road_size, "road crossing size in
pixels:", road_size_pix)
gdf = geopandas.read_file(road_crossing_intersect)

# keys: relative path to ortho, values: points
```

```
dic = {}
for idx, row in gdf.iterrows():
    coords = row.geometry.coords

    path_rel = row[atr_path_rel]

    if path_rel in dic:
        dic[path_rel].append(row)
    else:
        dic[path_rel] = []
        dic[path_rel].append(row)

rasters = [] # bboxes
# iterate over dictionary
for ras_path, point_list in dic.items():

    cv_raster = cv.imread(ras_path) # open with open cv
    geo_raster = rasterio.open(ras_path) # open with rasterio

    # raster = cv.cvtColor(raster, cv.COLOR_BGR2GRAY)
    # print(type(raster))

    for pt in point_list:
        patch, x, y = extract_patch(geo_raster, cv_raster, pt,
road_size_pix)
        #print(type(patch), x,y)
        patch_id = str(pt[atr_cross_id])
        file_name = patches_path + patch_id
        try:
            cv.imwrite(file_name + ".tif", patch) # save tif
            #print(patch_id + " --> succesfully written")
        except cv.error: # --> lies on boundary of 2 orthos, can't
extract patch
            print(patch_id + " --> lies on boundary of 2 orthos, can't
extract patch")
        tfw_path = file_name + ".tfw"
        save_tfw(tfw_path, ortho_res, 0, 0, x, y) # save tfw

        # raster
        try:
            raster = rasterio.open(file_name + ".tif")
            bbox = raster.bounds
            rasters.append(bbox)
        except rasterio.errors.RasterioIOError: # no such file or
directory
            pass

# create ortho bounding box
create_bbox(in_path=None,
            out_path=out_orto_rec,
            distance=None,
            atr_values=None,
            names=None,
            bboxes=rasters)

print(pyfiglet.figlet_format("Successfully executed ! :-)",
font="slant"))
```

Příloha 6: 5step_correl_patches.py

Popis výpočtu korelačního koeficientu nad dvěma poli v NumPy je uveden na Stack Overflow (2020).

Parametry skriptu:

```
# inputs:
patches_path = cesta k adresáři s ploškami reprezentujícími křižovatky
patches_format = formát plošek, defaultně ".tif"
old_images_path = cesta k adresáři s archivními LMS
road_crossing_path = cesta k souboru s vektorovými křižovatkami, výstup
skriptu 3step_intersect_road_crossing.py
old_img_bbox_path = cesta k polygonové vrstvě bounding boxů starých LMS,
poskytuje ČÚZK
img_name_atr = atribut ve vrstvě old_img_bbox_path, který obsahuje názvy
příslušných LMS bez předpony (1 polygon = 1 bounding box LMS), defaultně
"LMS_TIF_UP" (název atributu)
old_images_prefix = předpona pro názvy starých LMS, defaultně prostorové
rozlišení, např. "1m_"

path_size = hrana plošky reprezentující křižovatku v pixelech
id_attribute = atribut obsahující ID křižovatky (název atributu),
defaultně "cross_id"
elevation_attribute = atribut obsahující nadmořskou výšku (název
atributu)
crs = souřadnicový systém pro všechny vrstvy, např. "epsg:5514"

# outputs:
road_cross_gcp_path = cesta k adresáři s výstupními VLB z obrazové
korelace, pro každý LMS jedna vrstva s VLB
road_rectangles_path = cesta k výstupnímu souboru, který obsahuje
polygony ohraničující prohledávané oblasti pro jednotlivé křižovatky
output_format = formát pro soubory s výstupními VLB, např. ".geojson"

# parameters:
chanel = určení použitého pásma referenčního ortofota, "grey" pro průměr
všech 3 pásem s pomocí funkce RGB2grey(), "green_only" pro zelené pásmo
area_size = polovina hrany čtverce prohledávané oblasti v metrech
percent_best_point = počet výstupních bodů pro každou křižovatku, lze
zadat absolutně např. 5, případně relativně s pomocí floatu např. 0.05 =
5 %,
quality_limit = limit korelačního koeficientu, body z obrazové korelace
s horším korelačním koeficientem než tento limit, nebudou vyexportovány,
<-1,1>, např. 0.45
erase_dist = vzdálenost od okraje starého LMS. Body, které jsou okraji
blíže než tato vzdálenost, nebudou vyexportovány
temp_file_pref = předpona dočasných souborů, které budou nakonec smazány,
defaultně "__temp"
del_temp_files = True/False, smazání dočasných souborů
```

```
############################################################
import geopandas
import rasterio
from functions import *
import cv2
```

```
import glob
from pathlib import Path
import pyfiglet
import os

############################################################
# for Milovice 1989
# inputs:
patches_path =
"MILOVICE1989_SKRIPT_TEST/1989_patches_1m_splited_silnice_ulice/" #
folder with patches to process
patches_format = ".tif" # .tif format of patches and old_images
old_images_path = "MILOVICE1989_SKRIPT_TEST/Milovice1989_1m/*.tif" #
folder with old images to process .tif ONLY, automatically load in
greyscale
road_crossing_path =
"MILOVICE1989_SKRIPT_TEST/silnice/roads_crossing_intersect_1989.geojson"
# path to vector road crossing
old_img_bbox_path =
"MILOVICE1989_SKRIPT_TEST/silnice/1989_Milovice_sjtsk.shp" # path to
vector bboxes of raster provided by CUZK
img_name_atr = "LMS_TIF_UP"
old_images_prefix = "1m_"

path_size = 51 # PIXELS! edge size of input path in pixels
id_attribute = "cross_id" # name of attribute with road crossing ID
elevation_attribute = "Z"
crs = "epsg:5514"  # must be same for all layers --> inputs & outputs

# outputs:
road_cross_gcp_path = "MILOVICE1989_SKRIPT_TEST/1989_potential_1m/" #
with backslash
road_rectangles_path =
"MILOVICE1989_SKRIPT_TEST/silnice/roads_crossing_rectangles1989.geojson"
output_format = ".geojson"  # .geojson or .shp, must be with point

# parameters:
chanel = "grey"  # green_only or grey --> select band, in case of 1 band
image, all band are same, so select green_only
area_size = 220 # meters, half of rectangle edge 220 m --> in this case
very hight value for simulating non precious georeferencing
# 1.0 = 100 % --> % percent of pixels with best correl coeficient / OR
1,2,3... absolute values (not percentage):
percent_best_point = 5
quality_limit = 0.45 # worse point will be rejected---cor. coef <-1,1>
erase_dist = 5 # meters! at least same as path_size, or rather bigger
temp_file_pref = "__temp"
del_temp_files = True
############################################################

# there may be some file from previos session
if del_temp_files:
    for file in Path(road_cross_gcp_path).iterdir():
        file_name = Path(file).stem
        if str(file_name[0:6]) == temp_file_pref:
            try:
                os.remove(file)
            except PermissionError:
                pass

gdf = geopandas.read_file(road_crossing_path)
old_img_bbox = geopandas.read_file(old_img_bbox_path)
old_images = glob.glob(old_images_path)
```

```
for img in old_images:
    print("\n",
"############################################################")
    print("Processing", img)
    # strange error sometimes occurs:
https://stackoverflow.com/questions/72964575/typeerror-invalid-path-or-
file-error-even-when-the-file-exists
    img_geo = rasterio.open(img)
    img_cv = cv2.imread(img, cv2.IMREAD_GRAYSCALE)

    # plot_img(img)

    p = Path(img)
    raster_name = p.stem
    raster_name_suffix = p.name

    # create polygon bbox
    prefix_len = len(old_images_prefix)
    suffix_len = len(patches_format)
    img_original_name = raster_name_suffix[prefix_len : ]
    original_bbox_path = road_cross_gcp_path + temp_file_pref + "_bbox_"
+ img_original_name[:-suffix_len] + ".geojson"

old_img_bbox[old_img_bbox[img_name_atr]==img_original_name].to_file(origi
nal_bbox_path, driver="GeoJSON", encoding="UTF-8",crs=crs)
    # create line bbox:
    bbox = geopandas.read_file(original_bbox_path)
    bbox_geom = list(bbox["geometry"][0].exterior.coords)
    line_path = road_cross_gcp_path + temp_file_pref + "_bboxline_" +
img_original_name[:-suffix_len] + ".geojson"
    create_bbox(in_path=None, out_path=line_path, distance=0,
atr_values=None, names=None, bboxes=bbox_geom, crs=crs)
    # create buffer
    line = geopandas.read_file(original_bbox_path)
    buff_path = road_cross_gcp_path + temp_file_pref + "_buff_" +
img_original_name[:-suffix_len] + ".geojson"
    line_buff = line.to_crs(crs).buffer(-erase_dist, single_sided=True)
    line_buff.to_file(buff_path, driver="GeoJSON", encoding="UTF-
8",crs=crs)

    # just points, which lies in image:
    intersect = geopandas.overlay(bbox, gdf, how='intersection',
keep_geom_type=False)

    # print(p.stem, p.name, p.parent.absolute(), p.suffix, p.resolve(),
p.resolve().parent)

    x_res_old = abs(img_geo.transform[0])
    area_size_pix = m2pix(x_res_old, area_size)
    # print(area_size_pix, x_res_old)

    if str(type(percent_best_point)) == "<class 'float'>":
        pix_count = int((area_size_pix * 2 - (2 * int(path_size / 2))) *
(
            area_size_pix * 2 - (2 * int(path_size / 2))) *
percent_best_point)
    else:
        pix_count = percent_best_point

    width = img_geo.width
    height = img_geo.height

    # print(width, height)
```

```
    x_ras = []
    y_ras = []
    X_geo = []
    Y_geo = []
    X_geo_reference = []
    Y_geo_reference = []
    ids = []
    Zs = []
    coefs = []

    for idx, pt in intersect.iterrows():
        coords = pt.geometry.coords
        id = pt[id_attribute]
        elevation = pt[elevation_attribute]
        x_geo_reference = coords[0][0]
        y_geo_reference = coords[0][1]
        row_Y, col_X = geo2ras(img_geo, x_geo_reference, y_geo_reference)

        # print(pt)
        print("Searching potential GCP with " + id_attribute +
            " " + str(id) + " in " + raster_name_suffix + ".")

        top = row_Y - area_size_pix
        bottom = row_Y + area_size_pix
        left = col_X - area_size_pix
        right = col_X + area_size_pix

        if (top < 0) or (bottom < 0) or (left < 0) or (right < 0):
            print("Road patch number ", id, " lies too close to the edge
of image " +
                  img + " , can't set search area, I am skipping it.")
            continue  # skip one iteration of for loop
        # print(top, bottom, right, left)

        img_patch = img_cv[top: bottom, left: right]
        # open proper road patch
        road_patch = cv2.imread(patches_path + str(id) + patches_format)

        if str(type(road_patch)) == "<class 'NoneType'>":
            print("Can't open road patch number ", id, "I am skipping
it.")
            continue  # skip one iteration of for loop

        # plot_img(img_patch)
        # plot_img(road_patch)
        if (chanel == "grey"):
            # img_patch = cv.cvtColor(img_patch, cv.COLOR_BGR2GRAY) # old
image is already grey image
            road_patch = cv2.cvtColor(road_patch, cv.COLOR_BGR2GRAY)
        elif (chanel == "green_only"):
            # (B, G, R) = cv.split(img_patch) # old image already has
just one band
            # img_patch = G
            (B, G, R) = cv2.split(road_patch) # in case of 1 band image,
all band are same
            road_patch = G

        # plot_img(img, x, y)
        # plot_img(img_patch)
        try:
            x, y, coefs_list = img_corr(
                img_patch, road_patch, pix_count, quality_limit)
        except ValueError:
            print("Patch number ", id,
```

```
                    "has inapropriate size, I am skipping it.")
                continue
            if x == None:
                print("All points in patch number ", id, "has too low quality
(below " +
                str(quality_limit) + "), empty output.")
                continue
            count = len(coefs_list)
            id_list = [id] * count
            ids = ids + id_list
            Z_list = [elevation] * count
            Zs = Zs + Z_list
            coefs = coefs + coefs_list

            x_geo_list = [x_geo_reference] * count
            y_geo_list = [y_geo_reference] * count

            X_geo_reference = X_geo_reference + x_geo_list
            Y_geo_reference = Y_geo_reference + y_geo_list

            for i in range(count):
                # recalculate coordinates from img_patch to whole image
                x[i] = (x[i] - area_size_pix + col_X)
                y[i] = (y[i] - area_size_pix + row_Y)

                x_geo_coor, y_geo_coor = ras2geo(img_geo, y[i], x[i])
                X_geo.append(x_geo_coor)
                Y_geo.append(y_geo_coor)

            x_ras = x_ras + x
            y_ras = y_ras + y
            # plot_img(img_cv, x, y)

        print("Total amount of output points:", len(x_ras))
        out_path = road_cross_gcp_path + raster_name + output_format
        try:
            save_points_atr(X_geo, Y_geo, out_path,
                            [ids, Zs, coefs, X_geo_reference,
                                Y_geo_reference, x_ras, y_ras],
                            [id_attribute, elevation_attribute, "quality",
"X_ref",
                                "Y_ref", "x_col", "y_row"],
                            tuple_coords=None, crs=crs, new_idx=True,
idx_name="id")

            points_gdf = geopandas.read_file(out_path)
            buff_gdf = geopandas.read_file(buff_path)
            intersect_points = geopandas.overlay(buff_gdf,points_gdf,
how='intersection', keep_geom_type=False)

            if out_path[-3:] == "shp":
                driver = "ESRI Shapefile"
                encoding = "windows-1250"
            # elif out_path[-7:] == "geojson":
            else:
                driver = "GeoJSON"
                encoding = "UTF-8"
            intersect_points.to_file(out_path, driver=driver,
encoding=encoding)

        except ValueError:
            print("Can't save GCP of image " + raster_name)
        except PermissionError:
```

```
        print("Can't save GCP of image " + raster_name + ", because file
is open in another programe.")

try:
    create_bbox(in_path=road_crossing_path,
out_path=road_rectangles_path,
            distance=area_size, atr_values=None, names=None, bboxes=None,
crs=crs)
except PermissionError:
        print("Can't save road crossing rectangles, because file is open
in another programe.")

if del_temp_files:
    for file in Path(road_cross_gcp_path).iterdir():
        file_name = Path(file).stem
        if str(file_name[0:6]) == temp_file_pref:
            try:
                os.remove(file)
            except PermissionError:
                pass

print(pyfiglet.figlet_format("Successfully executed ! :-)",
font="slant"))
```

Příloha 7: 6step_flood_collapse_clusters.py

Parametry skriptu:

```
road_cross_gcp_path = adresář se soubory s VLB z obrazové korelace
old_images_path = adresář se starými LMS
images_format = formát starých LMS, např. ".tif"

# output:
out_path_folder = adresář s výstupními clusterovanými VLB
output_format = formát výstupních souborů s clusterovanými VLB, např.
".geojson"

# parameters:
pattern = shlukování bodů s pomocí čtyř/osmi okolí, "rook"/"queen"
res_type = metoda výpočtu souřadnic nového bodu ze shluku bodů, "max" /
"mean" / "weighted_average"
crs = souřadnicový systém pro všechny vrstvy, např. "epsg:5514"

# attribute names:
X_ref = atribut s geodetickými X souřadnicemi (název atributu), defaultně
"X_ref"
Y_ref = atribut s geodetickými Y souřadnicemi (název atributu), defaultně
"Y_ref"
x_col = atribut s rastrovými x souřadnicemi (název atributu), defaultně
"x_col"
y_row = atribut s rastrovými y souřadnicemi (název atributu), defaultně
"y_row"
elevation_attribute = atribut s nadmořskou výškou (název atributu),
defaultně "Z"
id_road_crossing = atribut s ID křižovatky (název atributu), defaultně
"cross_id"
id_point = atribut s ID pro každý jednotlivý VLB (název atributu),
defaultně "id"
corr_atr = atribut s korelačním koeficientem (název atributu), defaultně
"quality"
```

```
#################################################################
import glob
import geopandas
from pathlib import Path
from functions import *
import rasterio
import pyfiglet
#################################################################
# input Milovice 1989:
# *.format --> for listing all files with given format
road_cross_gcp_path =
"MILOVICE1989_SKRIPT_TEST/1989_potential_1m/*.geojson"
old_images_path = "MILOVICE1989_SKRIPT_TEST/Milovice1989_1m/"  # must be
with backslash
images_format = ".tif"  # must be with dot

# output:
out_path_folder = "MILOVICE1989_SKRIPT_TEST/1989_clustered/"  # must be
with backslash
```

```
output_format = ".geojson"  # .geojson or .shp, must be with dot

# parameters:
pattern = "queen"  # rook / queen
res_type = "weighted_average" # max / mean / weighted_average
crs = "epsg:5514"  # must be same for all layers --> inputs & outputs

# attribute names:
X_ref = "X_ref"  # geodetic coordinates
Y_ref = "Y_ref"
x_col = "x_col"  # raster coordinates
y_row = "y_row"
elevation_attribute = "Z" # attribute with elevation
id_road_crossing = "cross_id"  # name of attribute with road crossing ID
id_point = "id" # attribute name for every single point
corr_atr = "quality"  # (correlation coef.)

######################################################################

potential_gcp_files = glob.glob(road_cross_gcp_path)

# group points with same road crossing ID
for file in potential_gcp_files:
    file_name = Path(file).stem
    raster_path = old_images_path + file_name + images_format
    img_geo = rasterio.open(raster_path)
    print("\n",
"#########################################################")
    print("Clustering points of", file_name, "image.")
    print("Grouping points by road crossing ID.")
    gdf = geopandas.read_file(file)

    unique_road_ids = gdf[id_road_crossing].unique().tolist()
    grouped = gdf.groupby(id_road_crossing)
    # max values of atributes in every group
    maxs = list(grouped.max(numeric_only=True).iterrows())
    # min values of atributes in every group
    mins = list(grouped.min(numeric_only=True).iterrows())
    grouped = list(grouped.__iter__())
    print("Grouping done.")

    x_ras = []
    y_ras = []
    X_geo = []
    Y_geo = []
    X_geo_reference = []
    Y_geo_reference = []
    ids = []
    Zs = []
    coefs = []

    # for every group
    for i in range(len(maxs)):
        # print(maxs[i], mins[i])
        max_group = maxs[i]
        min_group = mins[i]
        road_cross_group = grouped[i]
        id = road_cross_group[0]
        print("Clustering road crossing with ID " + str(id) + ".")
        min_x = int(min_group[1][x_col])  # [0] is id
        max_x = int(max_group[1][x_col])
        min_y = int(min_group[1][y_row])
        max_y = int(max_group[1][y_row])
```

```
        x_pivot, y_pivot, coef_pivot = collapse_clusters(min_x, max_x,
min_y, max_y,

road_cross_group, corr_atr,
                                                         pattern,
res_type,
                                                         x_col, y_row)

        count = len(x_pivot)
        id_list = [id] * count
        ids = ids + id_list
        Z_list = [road_cross_group[1][elevation_attribute].iloc[0]] *
count # iloc[0] first item of group
        Zs = Zs + Z_list
        coefs = coefs + coef_pivot

        x_geo_list = [road_cross_group[1][X_ref].iloc[0]] * count
        y_geo_list = [road_cross_group[1][Y_ref].iloc[0]] * count

        X_geo_reference = X_geo_reference + x_geo_list
        Y_geo_reference = Y_geo_reference + y_geo_list

        for i in range(count):
            x_geo_coor, y_geo_coor = ras2geo(img_geo, y_pivot[i],
x_pivot[i])
            X_geo.append(x_geo_coor)
            Y_geo.append(y_geo_coor)
        x_ras = x_ras + x_pivot
        y_ras = y_ras + y_pivot

    print("Total amount of output points for image ",
          file_name, ": ", len(x_ras))
    out_path = out_path_folder + file_name + output_format
    save_points_atr(X_geo, Y_geo, out_path,
                    [ids, Zs, coefs, X_geo_reference, Y_geo_reference,
x_ras, y_ras],
                    [id_road_crossing, elevation_attribute, "quality",
"X_ref",
                        "Y_ref", "x_col", "y_row"],
                    tuple_coords=None, crs=crs, new_idx=True,
idx_name=id_point)

fig = pyfiglet.figlet_format("Successfully executed ! :-)", font="slant")
print(fig)
```

Příloha 8: 7step_RANSAC.py

Parametry skriptu:

```
potential_gcp_path = cesta k adresáři se soubory s clusterovanými VLB
old_images_path = cesta k adresáři se starými LMS
old_images_format = formát starých LMS, např. ".tif"
# output
out_path = cesta k adresáři s výstupními soubory s VLB z RANSACu

# names of attributes:
Stejné jako předchozí skript: X_ref, Y_ref, x_col, y_row,
id_road_crossing, id_point, elevation_attribute

# parameters
parameter_count = počet parametrů transformace v RANSACu 4/6 pro
projektivní tr./DLT, → volba transformace 2D/3D
iteration = počet iterací, defaultně 100 000
threshold_distance = prahová vzdálenost v RANSACU v metrech, typicky
jednotky metrů
quality_attribute = atribut se vzdáleností VLB od ideálního modelu
RANSACu (v pixelech) u výstupních souborů (název atributu)
print_er_messages = True/False, tisk zpráv do konzole, nedoporučuje se
pro náročnost na výpočetní výkon, použito pro ladění skriptu
```

```
#################################################################
from functions import *
import glob
import random
import math
import pyfiglet
import numpy as np

#################################################################
# inputs milovice 1989
potential_gcp_path = "MILOVICE1989_SKRIPT_TEST/1989_clustered/*.geojson"
old_images_path = "MILOVICE1989_SKRIPT_TEST/Milovice1989_1m/" # with
backslash
old_images_format = ".tif"
# output
out_path = "MILOVICE1989_SKRIPT_TEST/1989_RANSAC/"

# names of attributes:
X_ref = "X_ref" # geodetic coordinates
Y_ref = "Y_ref"
x_col = "x_col" # raster coordinates
y_row = "y_row"
id_road_crossing = "cross_id" # unique for every road crossing (more
points have this ID)
id_point = "id" # unique for every point
elevation_attribute = "Z" # attribute with elevation

# parameters
parameter_count = 4 # 4 for projective, 6 for DLT --> (2D / 3D
transformation)
iteration = 100000 # number of iteration
```

```
threshold_distance = 3 # meters
quality_attribute = "RAN_dist_pix" # name for output attribute for points
fitting threshold distance (pixels)
# print error messages (singular matrix in transformation / 0 in
denominator etc.) --> skipping these iterations
print_er_messages = False # !!!very long outputs!!! inefective, just for
testing purposes

#################################################################
potential_gcp_files = glob.glob(potential_gcp_path)

messages = []
# for every image and appropriate GCP point data in given directories
for file in potential_gcp_files:
    gdf = geopandas.read_file(file)
    points = list(gdf.iterrows())
    vector_path = Path(file)
    print("\n ##########################################################")
    print("Processing file " + str(vector_path) + "\n")
    raster_path = old_images_path + vector_path.stem + old_images_format
    img_geo = rasterio.open(raster_path) # open appropriate raster
    size = img_geo.shape # size of raster: height x width
    x_res_old = abs(img_geo.transform[0]) # resolution of raster in x
direction (same as in y direction most time)
    threshold_distance_pix = m2pix(x_res_old, threshold_distance) #
threshold distance in pixels

    # best model necessarities:
    best_model = []
    distances = []
    best_random_selection = []
    best_count = 0
    best_distance = 0
    best_iteration = 0

    # error, exception counter and bad selection counter:
    not_unique = 0
    singular = 0
    zero_division = 0

    # number of iteration throught point dataset
    for i in range(iteration):
        i += 1 # iteration starts at 0, +1 for printing
        if (i % 100) == 0: # inform about every 100th iteration
            print("\n Iteration counter: " + str(i) + " in file " +
str(vector_path.name))

        # random selection from given points, selection is without
repetition
        random_selection = random.sample(points, parameter_count)
        # check if list contains only points with different X_geo and
Y_geo coords
        # (if it belongs to different road crossing)
        if not unique(random_selection, X_ref, Y_ref):
            not_unique += 1
            if print_er_messages:
                print("\n Selected points with same " + id_road_crossing
+ " \n \
                Skipping iteration no " + str(i) + ". Model: ",
[ptn[1][id_point] for ptn in random_selection])
            continue

        #print(random_selection)
        X_geo = []
```

```
        Y_geo = []
        Z_geo = []
        x_ras = []
        y_ras = []

        for p in random_selection: # process points of random selection
            X_geo.append(p[1][X_ref]) # p[0] is id, p[1] is point info
(attributes + geometry)
            Y_geo.append(p[1][Y_ref])
            Z_geo.append(p[1][elevation_attribute]) # necessary only for
DLT
            x_center, y_center = origin2center(p[1][x_col], p[1][y_row],
size[1], size[0])
            x_ras.append(x_center)
            y_ras.append(y_center)
            #print(p[1][x_col], p[1][y_row], x_center, y_center, size)

        if parameter_count == 4:
            coefs = get_projective_coefs(X_geo, Y_geo, x_ras, y_ras)
        else:
            #coefs, err = get_DLT_coefs(X_geo, Y_geo, Z_geo, x_ras,
y_ras) # my function, using just 11 coordinates
            coefs, err = get_DLT_coefs2(X_geo, Y_geo, Z_geo, x_ras,
y_ras) # function from github, using 12 coordinates and least square

        if not str(type(coefs)) == "<class 'numpy.ndarray'>":
            singular += 1
            if print_er_messages:
                print("\n Matrix in transformation is singular. Can't
compute inverse matrix. Invalid configuration of points. \n \
                Skipping iteration no " + str(i) + ". Model: ",
[ptn[1][id_point] for ptn in random_selection])
            continue

        potential_best_model = []
        potential_distances = []
        potential_count = 0
        potential_best_distance = 0

        # iterate over whole dataset and transform all points using
computed coeficients:
        for p in points:
            if parameter_count == 4:
                xn, yn = projective_trans(p[1][X_ref], p[1][Y_ref],
coefs)
            else:
                xn, yn = DLT_trans(p[1][X_ref], p[1][Y_ref],
p[1][elevation_attribute], coefs)

            if xn == None: # denominator in transformation is 0
                zero_division += 1
                if print_er_messages:
                    print("\n Denominator in projective transformation is
0. Skipping iteration. Point with " + id_point + ": \n" +
                    str(p[1][id_point]) + ", model: ", [ptn[1][id_point]
for ptn in random_selection])
                continue
            x_center, y_center = origin2center(p[1][x_col], p[1][y_row],
size[1], size[0])
            distance = round(math.dist([xn, yn], [x_center, y_center]),
3) # compute distance
            # distance = math.sqrt((Xn - p[1][X_ref]) ** 2 + (Yn -
p[1][Y_ref]) ** 2)  alternative distance computation
```

```
            if distance < threshold_distance_pix:
                potential_best_model.append(p)
                potential_distances.append(distance)
                potential_count += 1
                potential_best_distance = potential_best_distance +
distance

        # Model with more point is new best. If number of points is same,
check RANSAC distances.
        if (potential_count > best_count) or \
        ((potential_count == best_count) and (potential_best_distance <
best_distance)):

            best_model = potential_best_model
            distances = potential_distances
            best_random_selection = random_selection
            best_count = potential_count
            best_distance = potential_best_distance
            best_iteration = i

            print("\n Best model updated in iteration " +
str(best_iteration) +
            ". Number of points fitting threshold distance: " +
str(best_count) +
            ". Total RANSAC distance for them: " +
str(round(best_distance, 3)) + " pixels")
            print("Point IDs setting up transformation of new best model:
",
            [ptn[1][id_point] for ptn in random_selection])

    msg = "\n ######################################################" + \
    "\n Final number of points fitting threshold distance: " +
str(best_count) + \
    " in image: " + vector_path.stem + \
    "\n Point IDs setting up transformation of final best model --> " + \
    str([ptn[1][id_point] for ptn in best_random_selection]) + \
    " \n --> Points were found in " + str(best_iteration) + " RANSAC
iteration. " + \
    "Sum of all distances computed in RANSAC for best model: " + \
    str(round(best_distance, 3)) + " pixels" + "  Mean distance for whole
dataset: " + \
    str(round(best_distance / best_count, 3)) + " pixels" + \
    "\n Number of not unique selection / singular matrixes /
zero_division: " + \
    str(not_unique) + " / " + str(singular) + " / " + str(zero_division)
    messages.append(msg)

    out = out_path + vector_path.name
    save_geodataframe(out, best_model, gdf, [quality_attribute],
[distances])

print("\n \n \n \n")
print(pyfiglet.figlet_format("#######", font="slant"))
print(pyfiglet.figlet_format("--> RESULTS <--", font="slant"))
for msg in messages:
    print(msg)

print("\n \n")
print(pyfiglet.figlet_format("Successfully executed ! :-)",
font="slant"))
```

Příloha 9: 8step_GCP2MicMac.py

Parametry skriptu:

```
# inputs Milovice 1989:
GCP_folder_path = cesta k adresáři s VLB z RANSACu
GCP_json_prefix = předpona názvů souborů s VLB, označuje prostorové
rozlišení použitých rastrů, např. "1m_"
original_imgs_folder = cesta k adresáři s originálními nepřevzorkovanými
LMS, tak jak je poskytuje ČÚZK
original_imgs_format = formát originálních nepřevzorkovaných LMS, např.
".tif"

# outputs:
GCP_ref_file_path = cesta k výstupnímu souboru, který obsahuje geodetické
souřadnice VLB, např. "GCPs.txt"
GCP_list_file_path = cesta k výstupnímu souboru, který obsahuje VLB ID,
např. "id_GCPs.txt"
out_GCP_XML_path = cesta k výstupnímu souboru, který obsahuje rastrové
souřadnice VLB, např. "MeasuresInit-S2D.xml"

visualisation = True/False, vizualizace jednotlivých mezikroků skriptu
s pomocí knihovny Matplotlib pro kontrolní účely
exclude_negative = True/False, vyloučení bodů, které jsou v oříznutém
snímku OIS-Reech_*.tif mimo obraz (mají negativní jednu z rastrových
souřadnic), není třeba je ignorovat, jelikož si s nimi MicMac dokáže
poradit.

# atributes name:
Viz předchozí skript: X_ref, Y_ref, x_col, y_row.

id_attribute = atribut obsahující ID křižovatky (název atributu),
defaultně "cross_id"
quality_atr = atribut s korelačním koeficientem pro jednotlivé VLB (název
atributu), defaultně "quality"
RANSAC_distance_atr = atribut se vzdáleností VLB od ideálního modelu
RANSACu (v pixelech) u výstupních souborů (název atributu), defaultně
"RAN_dist_pix"
Z_ref = atribut s nadmořskou výškou jednotlivých VLB (název atributu)

# advanced parameters for micmac
# dont change it if you dont know, what you are doing
GCP_prefix = předpona u ID jednotlivých VLB, defaultně "GCP"
first_lines = první 2 řádky s označením souřadnicového systému u souboru
GCPs.txt, důležité pro MicMac
encoding = kódování výstupních souborů, defaultně "utf-8"
prefered = i po RANSACu se může vyskytnou pro jednu křižovatku více než
jeden odpovídající bod v archivním LMS. Pro MicMac však musí být jen
jeden. Tímto parametrem se nastavuje, zda bude preferován bod na základě
vyššího korelačního koeficientu nebo menší vzdálenosti v RANSACu (není
implementováno), tj. defaultně "correl_coef"

# MicMac folder:
OIS_xmls_prefix = předpona souborů, které obsahují souřadnice rámových
značek, defaultně "MeasuresIm-"
OIS_xmls_suffix = přípona souborů, které obsahují souřadnice rámových
značek, defaultně ".tif.xml"
micmac_folder = cesta k projektu MicMacu
OIS_reference = soubor se souřadnicemi rámových značek namapovaných do
snímku OIS-Reech_*.tif
OIS_img_prefix = předpona OIS-Reech_*.tif snímků, defaultně "OIS-Reech_"
```

```
OIS_img_suffix = formát OIS-Reech_*.tif snímků, defaultně ".tif"
OIS_folder = cesta k adresáři se soubory, které obsahují souřadnice
rámových značek OIS-Reech_*.tif snímků
```

```
#############################################################
import geopandas
import glob
from pathlib import Path
from functions import *
import xml.etree.ElementTree as ET
import rasterio
import cv2
################################################
# inputs Milovice 1989:
GCP_folder_path = "MILOVICE1989_SKRIPT_TEST/1989_RANSAC/*.geojson"
GCP_json_prefix = "1m_"
original_imgs_folder =
"MILOVICE1989_SKRIPT_TEST/Milovice1989_rotated_tfw/"
original_imgs_format = ".tif"

# outputs:
GCP_ref_file_path = "GCPs.txt" # output with GCP geo reference
coordinates
GCP_list_file_path = "id_GCPs.txt" # txt file, just list of IDs of GCPs
for possible manual collecting
out_GCP_XML_path = "MeasuresInit-S2D.xml" # raster coords of GCP for
MicMac

visualisation = False # True / False   --> visualisation with matplotlib
exclude_negative = False

# atributes name:
X_ref = "X_ref"
Y_ref = "Y_ref"
Z_ref = "Z"
x_col = "x_col" # required just for visualisation
y_row = "y_row"
id_attribute = "cross_id" # name of attribute with road crossing ID
quality_atr = "quality"
RANSAC_distance_atr = "RAN_dist_pix"

# advanced parameters for micmac
# dont change it if you dont know, what you are doing
GCP_prefix = "GCP"
first_lines = "#F= N X Y Z\n\
#Here the coordinates are in EPSG:5514 X=Easting Y=Northing Z=Altitude
\n"
encoding = "utf-8" # encoding of *.points files
prefered = "correl_coef" # preffer correl_coef or RANSAC_distance (not
implemented), removing duplicities

# MicMac folder:
OIS_xmls_prefix = "MeasuresIm-"
OIS_xmls_suffix = ".tif.xml"
micmac_folder = "milovice/1989_micmac1/"
OIS_reference = "milovice/1989_micmac1/MeasuresIm_OIS_reference.xml"
OIS_img_prefix = "OIS-Reech_"
OIS_img_suffix = ".tif"
OIS_folder = micmac_folder + "Ori-InterneScan/"
```

```
################################################

GCP_files_list = glob.glob(GCP_folder_path)

dic = {}
list_dic = [] # every img has own dic with GCP in this list

for file in GCP_files_list:
    gdf = geopandas.read_file(file)
    current_img_dic = {}

    for idx, ptn in gdf.iterrows():
        cross_id = ptn[id_attribute]
        if not cross_id in dic:
            dic[cross_id] = ptn

        # every img has own dic with GCP in this list:
        if not cross_id in current_img_dic: # remove duplicities
            current_img_dic[cross_id] = ptn
        elif prefered == "correl_coef":
            if current_img_dic[cross_id][quality_atr] < ptn[quality_atr]:
#previous ptn has smaller quality than next
                current_img_dic[cross_id] = ptn # replace previus with
new

        elif prefered == "RANSAC_distance":
            pass # may implement in the future
    list_dic.append((file, current_img_dic))


dic = dict(sorted(dic.items()))
# save list of GCPs as txt (just names for possible manual collectiong of
GCP), save txt with GCPs and geo coordinates for MicMac
with open(GCP_ref_file_path, mode="w", encoding=encoding) as GCP_ref:
    GCP_ref.write(first_lines)
    with open(GCP_list_file_path, mode="w", encoding=encoding) as
GCP_list:
        for cross_id, ptn in dic.items():
            cross_name = GCP_prefix + str(cross_id)

            GCP_ref_line = cross_name + " " + str(ptn[X_ref]) + " " +
str(ptn[Y_ref]) + " " + str(ptn[Z_ref]) + "\n"
            GCP_list_line = cross_name + "\n"

            GCP_ref.write(GCP_ref_line)
            GCP_list.write(GCP_list_line)


# transform all GCP raster coords to MicMac XML OIS images
OIS_ref_points = get_XML_points(OIS_reference)
print("OIS_ref_points", OIS_ref_points)
print("####################################################")
print("####################################################")
print("####################################################")
root = ET.Element("SetOfMesureAppuisFlottants") # start building XML tree
structure

for path, ptn_dic in list_dic:
    file_path = Path(path)
    img_name = file_path.stem[len(GCP_json_prefix):]
    OIS_file_path = OIS_folder + OIS_xmls_prefix + img_name +
OIS_xmls_suffix
    current_OIS_points = get_XML_points(OIS_file_path)
```

```
    img_path = original_imgs_folder + img_name + original_imgs_format
    current_raster = rasterio.open(img_path)

    print("Processing image:", img_path)
    print("Current_OIS_points:", current_OIS_points)

    pix_top_list = []
    pix_bottom_list = []
    ref_top_list = []
    ref_bottom_list = []

    for i in range(len(OIS_ref_points)): # set up points creating
transformation
        pix_top_list.append([current_OIS_points[i][0],
current_OIS_points[i][1], 0, 0, 1, 0]) # --> source coordinates
        pix_bottom_list.append([0, 0, current_OIS_points[i][0],
current_OIS_points[i][1], 0, 1])

        ref_top_list.append(OIS_ref_points[i][0]) # X ref --> targed
coordinates
        ref_bottom_list.append(OIS_ref_points[i][1]) # Y ref

    coefs = least_square_solver(pix_top_list + pix_bottom_list,
ref_top_list + ref_bottom_list)
    print("Coefs of affine transformation using least square:", coefs)

    img_XML = ET.SubElement(root, "MesureAppuiFlottant1Im") # for 1 img
    img_name_xml = ET.SubElement(img_XML, "NameIm") # name of raster
    img_name_xml.text = OIS_img_prefix + img_name + OIS_img_suffix

    # raster coords for visualisation with matplotlib
    if visualisation:
        x_list = []
        y_list = []
        x_ref = []
        y_ref = []

    for cross_id, ptn in ptn_dic.items():
        coords = ptn.geometry.coords
        x_geo_reference = coords[0][0]
        y_geo_reference = coords[0][1]

        # geo2ras --> transform coords from resampled img to original img
from CUZK
        row_y, col_x = geo2ras(current_raster, x_geo_reference,
y_geo_reference)
        # affine tr. --> transform coords from original img from CUZK to
OIS img from MicMac
        x,y = affine_trans(col_x, row_y, coefs)

        # OIS imgs cover smaller area, so some of GCPs are with negative
coords, remove them
        if exclude_negative:
            if (x<0) or (y<0):
                continue

        # for 1 GCP:
        one_measure = ET.SubElement(img_XML, "OneMesureAF1I")
        namept = ET.SubElement(one_measure, "NamePt")
        namept.text = GCP_prefix+str(cross_id)
        ptim = ET.SubElement(one_measure, "PtIm")
        ptim.text = str(x) + " " + str(y)

        if visualisation:
```

```
            x_list.append(x) # OIS imgs
            y_list.append(y)
            x_ref.append(col_x) # original CUZK imgs
            y_ref.append(row_y)

    if visualisation:
        ras_cuzk = cv2.imread(img_path)
        ras_cuzk = cv2.cvtColor(ras_cuzk, cv2.COLOR_BGR2GRAY) # function
plot_img can handle just grey img
        plot_img(ras_cuzk, x_ref, y_ref)

        ras_OIS = cv2.imread(micmac_folder + OIS_img_prefix + img_name +
OIS_img_suffix)
        ras_OIS = cv2.cvtColor(ras_OIS, cv2.COLOR_BGR2GRAY)
        plot_img(ras_OIS, x_list, y_list)

    print("################################################")

# save XML with proper encoding, indentation etc.
tree = ET.ElementTree(root)
ET.indent(tree, space="\t", level=0)
tree.write(out_GCP_XML_path, encoding="utf-8", xml_declaration = True)
```

Příloha 10: functions.py – funkce k předchozím skriptům

```
import cv2 as cv
import matplotlib.pyplot as plt
import rasterio
import numpy as np
import geopandas
import pandas as pd
# import gdal    # for Python 3.8
from osgeo import gdal    # for Python 3.10
from pathlib import Path
from shapely.geometry import Polygon, LineString
import math
import json
import xml.etree.ElementTree as ET

def geo2ras(ras, x, y):
    row, col = ras.index(x, y)
    return row, col # probably ok

def ras2geo(ras, row, col):
    return ras.xy(row,col) # probably ok

def m2pix(resolution, perimeter):
    pix_amount = perimeter/resolution
    return int(pix_amount)

def save_points_atr(xs, ys, path, atr = [], names = [], tuple_coords =
None, crs="epsg:5514", new_idx=False, idx_name="ID"):

    if tuple_coords != None:
        for p in tuple_coords:
            xs.append(p[0])
            ys.append(p[1])

    ids = range(0, len(xs), 1)

    dic = {}
    for i in range(len(atr)):
        dic[names[i]] = atr[i]

    if new_idx:
        dic[idx_name] = ids # indexes
    df = pd.DataFrame(dic) # attributes to dataframe

    if path[-3:] == "shp":
        driver = "ESRI Shapefile"
        encoding = "windows-1250"
    elif path[-7:] == "geojson":
        driver = "GeoJSON"
        encoding = "UTF-8"

    gdf = geopandas.GeoDataFrame(
        df, geometry = geopandas.points_from_xy(xs, ys))
    gdf.to_file(path, driver = driver, encoding = encoding, crs=crs)

def create_bbox(in_path=None, out_path=None, distance=None,
atr_values=None, names=None, bboxes=None, crs="epsg:5514"):
    rectangles = []
    if distance == 0:
        line = LineString(bboxes)
        rectangles.append(line)
```

```
    elif distance!=None:
        gdf = geopandas.read_file(in_path)
        for idx, c in gdf.iterrows():
            coords = c.geometry.coords

            xmax = coords[0][0]+distance
            xmin = coords[0][0]-distance
            ymax = coords[0][1]+distance
            ymin = coords[0][1]-distance

            rec = Polygon([(xmin, ymax),(xmax, ymax),(xmax, ymin),(xmin,
ymin)])
            rectangles.append(rec)
    else:
        for bbox in bboxes:
            xmax = bbox[2]
            xmin = bbox[0]
            ymax = bbox[1]
            ymin = bbox[3]

            rec = Polygon([(xmin, ymax),(xmax, ymax),(xmax, ymin),(xmin,
ymin)])
            rectangles.append(rec)

    dic = {}
    ids = range(0, len(rectangles), 1)
    dic["rec_id"] = ids
    if atr_values != None:
        for i in range(len(atr_values)):
            dic[names[i]] = atr_values[i]

    df = pd.DataFrame(dic)  # attributes to dataframe
    gdf = geopandas.GeoDataFrame(df, geometry=rectangles)

    if out_path[-3:] == "shp":
        driver = "ESRI Shapefile"
        encoding = "windows-1250"
    #elif out_path[-7:] == "geojson":
    else:
        driver = "GeoJSON"
        encoding = "UTF-8"

    gdf.to_file(out_path, driver=driver, encoding=encoding, crs=crs)

# save edges of graph
# just for vizualization and testing
def save_edges(lines, targed_file):
    num_lines = len(lines)
    json_lines = []
    for i in range(num_lines - 1):
        one_json_line = {
          "type": "Feature",
          "properties": {"type": i},
          "geometry": {
            "type": "LineString",
            "coordinates": lines[i]
          }
        }
        json_lines.append(one_json_line)

    gj_structure = {"type":"FeatureCollection",
    "crs": {"type": "name", "properties": {"name":
"urn:ogc:def:crs:EPSG::5514"}},
    "features": json_lines
```

```
    }
    # save output geojson file:
    with open(targed_file, "w", encoding = "utf-8") as f:
        json.dump(gj_structure,f, indent = 2)

def warp(resolution, in_path, out_path=None, resampling="bilinear",
prefix=None, suffix=None, format_out=".tif"): # bilinear  near
    p = Path(in_path)

    if prefix == None:
        prefix = ""
    if suffix == None:
        suffix = ""

    if out_path == None:
        name_res = str(p.parent.absolute()) + "\\" + prefix + str(p.stem)
+ suffix + format_out
        out_path = name_res
    else:
        p_out = Path(out_path)
        name_res = str(p_out.parent.absolute()) + "\\" + prefix +
str(p.stem) + suffix + format_out
        out_path = name_res

    xres = resolution
    yres = resolution

    ds = gdal.Warp(out_path, in_path, xRes=xres, yRes=yres,
resampleAlg=resampling) # ds = gdal.Warp(out_path,
    # in_path, warpoptions=dict(xRes=xres, yRes=yres,
resampleAlg=resample_alg))
    return ds

plots_num = [1]
def plot_img(image, pointX=None, pointY=None):
    image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
    plt.figure(plots_num[0])
    plt.imshow(image)
    if not (pointY==None or pointY==None):
        plt.scatter(pointX, pointY, c="r", s=40)
    plt.show()
    plots_num[0] = plots_num[0] + 1

def extract_patch(geo_ras, cv_ras, ptn, patch_size):
    half_size = int(patch_size / 2)
    coords = ptn.geometry.coords
    row, col = geo2ras(geo_ras, coords[0][0], coords[0][1])

    left = int(col - half_size)
    right = int(col + half_size)
    bottom = int(row + half_size) # bottom because of reverse axis order
    top = int(row - half_size)

    #print(left, right, bottom, top)
    #plot_img(cv_ras, [left, left, right, right, col], [top, bottom, top,
bottom, row]) # plot corners of rectangle

    path_cross = cv_ras[int(top): int(bottom + 1),
                 int(left): int(right + 1)]

    #plot_img(path_cross)
    x, y = ras2geo(geo_ras, top, left)
    return path_cross, x, y
```

```
def save_tfw(path, pixel_size, rot_x, rot_y, x , y, pixel_size_y = None):
    with open(path, "w") as f:
        f.write(str(pixel_size)) # pixel size
        f.write("\n")

        f.write(str(rot_x))  # rotation x
        f.write("\n")

        f.write(str(rot_y))  # rotation y
        f.write("\n")

        if pixel_size_y == None:
            f.write(str(-pixel_size)) # pixel size
        else:
            f.write(str(pixel_size_y)) # pixel size
        f.write("\n")

        f.write(str(x))  # left upper corner
        f.write("\n")

        f.write(str(y))  # left upper corner

def img_corr(search_area, searched_path, quantity, limit):
    # plot_img(search_area)
    # plot_img(searched_path)
    row, col = searched_path.shape
    half_row = int(row/2)
    half_col = int(col/2)

    shape = search_area.shape
    row_area = shape[0]
    col_area = shape[1]
    # print(half_row, half_col, row_area, col_area)

    key_roads = []
    correl_coefs = []
    for x in range(half_col, col_area-half_col):
        for y in range(half_row, row_area-half_row):
            # direction of indexes yx
            part_path = search_area[y - half_row : y + half_row + 1, x -
half_col: x + half_col + 1]
            # print(x, y) # mid pix
            # plot_img(part_path[:,:,0:1])
            # print("shape", part_path.shape, searched_path.shape)

            #plot_img(search_area, [y-half_row, y+half_row,y], [x-
half_col, x+half_col,x])

            #print(y - half_row, y + half_row + 1, x - half_col, x +
half_col + 1)

            # https://stackoverflow.com/questions/61557443/numpy-
corrcoef-doubts-about-return-value
            part_path_fl = part_path.flatten()
            searched_path_fl = searched_path.flatten()
            correl = round(np.corrcoef(part_path_fl, searched_path_fl)[0,
1], 3) # round to 3 decimal places
            # print(correl)
            if correl > limit:
                key_roads.append((x, y, correl))
                correl_coefs.append(correl)

    if len(correl_coefs) == 0:
        return None, None, None
```

```
    key_roads.sort(key=lambda x : x[2], reverse=True) # sort according to
correl coef
    correl_coefs.sort(reverse=True)

    if len(correl_coefs) > quantity:
        key_roads = key_roads[ : quantity]
        correl_coefs = correl_coefs[ : quantity]

    x = [item[0] for item in key_roads]
    y = [item[1] for item in key_roads]

    #print(x,y)
    # plot_img(searched_path)
    # plot_img(search_area, x, y)

    return x, y, correl_coefs

def get_projective_coefs(obj_X, obj_Y, col_x, row_y):
    coefs_list = []
    num_list = []
    for i in range(4):
        # for X:
        coefs_list.append(obj_X[i])
        coefs_list.append(obj_Y[i])
        coefs_list.append(1)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(-col_x[i]*obj_X[i])
        coefs_list.append(-col_x[i]*obj_Y[i])
        # for Y:
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(obj_X[i])
        coefs_list.append(obj_Y[i])
        coefs_list.append(1)
        coefs_list.append(-row_y[i]*obj_X[i])
        coefs_list.append(-row_y[i]*obj_Y[i])

        # right side of equation:
        num_list.append(col_x[i])
        num_list.append(row_y[i])

    P = np.array([coefs_list[0:8],
                  coefs_list[8:16],
                  coefs_list[16:24],
                  coefs_list[24:32],
                  coefs_list[32:40],
                  coefs_list[40:48],
                  coefs_list[48:56],
                  coefs_list[56:64],
                  ])
    nums = np.array(num_list)
    # print(P)
    # print(nums)
    # res_solved = np.linalg.inv(P).dot(nums) # tradition equation
solver, transposion is done automatically
    try:
        res_solved = np.linalg.solve(P, nums) # numpy solver, inverse
matrix is compute automatically
        # solution: a1, a2, a3, b1, b2, b3, c1, c2
```

```
    except np.linalg.LinAlgError: # P is singular matrix, cant compute
inverse matrix, invalid configuration of poitns
        return False
    return res_solved

def get_DLT_coefs(obj_X, obj_Y, obj_Z, col_x, row_y):
    coefs_list = []
    num_list = []

    for i in range(6):
        # for X:
        coefs_list.append(obj_X[i])
        coefs_list.append(obj_Y[i])
        coefs_list.append(obj_Z[i])
        coefs_list.append(1)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(-col_x[i]*obj_X[i])
        coefs_list.append(-col_x[i]*obj_Y[i])
        coefs_list.append(-col_x[i]*obj_Z[i])
        # for Y:
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(0)
        coefs_list.append(obj_X[i])
        coefs_list.append(obj_Y[i])
        coefs_list.append(obj_Z[i])
        coefs_list.append(1)
        coefs_list.append(-row_y[i]*obj_X[i])
        coefs_list.append(-row_y[i]*obj_Y[i])
        coefs_list.append(-row_y[i]*obj_Z[i])

        # right side of equation:
        num_list.append(col_x[i])
        num_list.append(row_y[i])

    P = np.array([coefs_list[0:11],
                  coefs_list[11:22],
                  coefs_list[22:33],
                  coefs_list[33:44],
                  coefs_list[44:55],
                  coefs_list[55:66],
                  coefs_list[66:77],
                  coefs_list[77:88],
                  coefs_list[88:99],
                  coefs_list[99:110],
                  coefs_list[110:121]
                  ])

    num_list.pop() # delete last coordinate, cause it should have just 11
elements
    nums = np.array(num_list)

    # print(P)
    # print(nums)
    # res_solved = np.linalg.inv(P).dot(nums) # tradition equation
solver, transposion is done automatically
    try:
        res_solved = np.linalg.solve(P, nums) # numpy solver, inverse
matrix is compute automatically
        # solution: a1, a2, a3, a4, b1, b2, b3, b4, c1, c2, c3
```

```
    except np.linalg.LinAlgError: # P is singular matrix, cant compute
inverse matrix, invalid configuration of poitns
        return False
    return res_solved, None

def origin2center(x_col, y_row, width, height):
    x = x_col - (width - 1) / 2
    y = (height - 1) / 2 - y_row
    return x, y

def projective_trans(X, Y, coefs):
    denominator = coefs[6] * X + coefs[7] * Y + 1
    if denominator == 0:
        return None, None
    # return x, y
    return float((coefs[0] * X + coefs[1] * Y + coefs[2])/denominator), \
           float((coefs[3] * X + coefs[4] * Y + coefs[5])/denominator)

def DLT_trans(X, Y, Z, coefs):
    denominator = coefs[-3] * X + coefs[-2] * Y + coefs[-1] * Z + 1
    if denominator == 0:
        return None, None
    # return x, y
    return float((coefs[0] * X + coefs[1] * Y + coefs[2] * Z +
coefs[3])/denominator), \
           float((coefs[4] * X + coefs[5] * Y + coefs[6] * Z +
coefs[7])/denominator)

def save_geodataframe(path, feature_list, source_gdf, names, values):
    gdf = geopandas.GeoDataFrame()
    # fill geodataframe with data:
    for col in source_gdf.columns:
        gdf[col] = [feat[1][col] for feat in feature_list]
    # add additional attributes:
    for i in range(len(names)):
        gdf[names[i]] = values[i]

    if path[-3:] == "shp":
        driver = "ESRI Shapefile"
        encoding = "windows-1250"
    elif path[-7:] == "geojson":
        driver = "GeoJSON"
        encoding = "UTF-8"
    gdf.to_file(path, driver=driver, encoding=encoding,
crs=source_gdf.crs)

def compute_pivot(inp, comp):
    y = np.array(inp[0])
    x = np.array(inp[1])
    c = np.array(inp[2])
    if comp == "weighted_average":
        c_sum = np.sum(c)
        y_out = round(np.sum(y*c)/c_sum,3)
        x_out = round(np.sum(x*c)/c_sum,3)
        c_out = round(np.mean(c),3)
    elif comp == "mean":
        count = c.size
        y_out = round(np.sum(y)/count,3)
        x_out = round(np.sum(x)/count,3)
        c_out = round(np.mean(c),3)
    else:
        c_idx = c.argmax()
        c_out = c[c_idx]
        y_out = y[c_idx]
```

```
        x_out = x[c_idx]
    return y_out, x_out, c_out

# recursive function for rook scheme
def collapse_rook(in_y,in_x,dic,arr,outputs):
        # test inputs coords:
        arr_shape = arr.shape # y, x
        if not ((in_y < arr_shape[0]) and (in_y >= 0) and
        (in_x < arr_shape[1]) and (in_x >= 0)):
            return # coords lies out of array, it would crash

        quality_val = arr[in_y, in_x]
        if quality_val != None:
            del dic[(in_y, in_x)] # delete element from dic
            arr[in_y, in_x] = None
            # append values to list in outputs tuple
            # lists in tuples are modified
            outputs[0].append(in_y)
            outputs[1].append(in_x)
            outputs[2].append(quality_val)

            # recurse in rook schema
            collapse_rook(in_y+1, in_x, dic, arr, outputs)
            collapse_rook(in_y-1, in_x, dic, arr, outputs)
            collapse_rook(in_y, in_x+1, dic, arr, outputs)
            collapse_rook(in_y, in_x-1, dic, arr, outputs)

# recursive function for queen scheme
def collapse_queen(in_y,in_x,dic,arr,outputs):
        # test inputs coords:
        arr_shape = arr.shape # y, x
        if not ((in_y < arr_shape[0]) and (in_y >= 0) and
        (in_x < arr_shape[1]) and (in_x >= 0)):
            return # coords lies out of array, it would crash

        quality_val = arr[in_y, in_x] # get quality in coordinates
        if quality_val != None:
            del dic[(in_y, in_x)] # delete element from dic
            arr[in_y, in_x] = None
            # append values to list in outputs tuple
            # lists in tuples are modified
            outputs[0].append(in_y)
            outputs[1].append(in_x)
            outputs[2].append(quality_val)

            # recurse in rook scheme
            collapse_queen(in_y+1, in_x, dic, arr, outputs)
            collapse_queen(in_y-1, in_x, dic, arr, outputs)
            collapse_queen(in_y, in_x+1, dic, arr, outputs)
            collapse_queen(in_y, in_x-1, dic, arr, outputs)

            # recurse in queen scheme
            collapse_queen(in_y+1, in_x+1, dic, arr, outputs)
            collapse_queen(in_y-1, in_x-1, dic, arr, outputs)
            collapse_queen(in_y-1, in_x+1, dic, arr, outputs)
            collapse_queen(in_y+1, in_x-1, dic, arr, outputs)

def collapse_clusters(min_x,max_x,min_y,max_y,
                      points,quality_atr="quality",type_pattern="queen",
                      quality_computation="weighted_average",
                      x_col_atr="x_col",y_row_atr="y_row"):

    width_ar = max_x-min_x+1
    height_ar = max_y-min_y+1
```

```
    array = np.full((height_ar, width_ar), None) # (rowY, colX), fill
value

    # fill array and dic with coordinates
    dic_ptn = {}
    for ptn in list(points)[1].iterrows():
        val = ptn[1][quality_atr]
        yrow_ar = ptn[1][y_row_atr]-min_y
        xcol_ar = ptn[1][x_col_atr]-min_x
        array[yrow_ar, xcol_ar] = val
        dic_ptn[(yrow_ar,xcol_ar)] = val

    # input state of array and dic
    # print(array)
    # print(dic_ptn)

    if (type_pattern=="queen"):
        scheme_function = collapse_queen
    else:
        scheme_function = collapse_rook

    x_out = []
    y_out = []
    c_out = []

    while dic_ptn:
        key = next(iter(dic_ptn)) # get first key of dic
        cluster = ([],[],[]) # y_row, x_col, quality
        # recurse
        # collapse function modifies lists in cluster tuple
        scheme_function(key[0], key[1],
                 dic_ptn, array, cluster)

        # print(array)
        # print(dic_ptn)
        # print(cluster)
        y,x, coef = compute_pivot(cluster, quality_computation)

        x_out.append(x+min_x)
        y_out.append(y+min_y)
        c_out.append(coef)
    return x_out, y_out, c_out

def unique(points, x_atr, y_atr):
    x = []
    y = []
    for ptn in points:
        X_geo = ptn[1][x_atr]
        Y_geo = ptn[1][y_atr]

        if (X_geo in x) or (Y_geo in y):
            return False
        else:
            x.append(X_geo)
            y.append(Y_geo)
    return True

# implementation of DLT with with least squares
# available at
# https://github.com/acvictor/DLT/blob/master/DLT.py
np.set_printoptions(precision=3, suppress=True,formatter={'float': '{:
0.3f}'.format})

def Normalization(nd, x):
```

```
    '''
    Normalization of coordinates (centroid to the origin and mean
distance of sqrt(2 or 3).
    Input
    -----
    nd: number of dimensions, 3 here
    x: the data to be normalized (directions at different columns and
points at rows)
    Output
    ------
    Tr: the transformation matrix (translation plus scaling)
    x: the transformed data
    '''

    x = np.asarray(x)
    m, s = np.mean(x, 0), np.std(x)
    if nd == 2:
        Tr = np.array([[s, 0, m[0]], [0, s, m[1]], [0, 0, 1]])
    else:
        Tr = np.array([[s, 0, 0, m[0]], [0, s, 0, m[1]], [0, 0, s, m[2]],
[0, 0, 0, 1]])

    Tr = np.linalg.inv(Tr)
    x = np.dot(Tr, np.concatenate((x.T, np.ones((1, x.shape[0])))))
    x = x[0:nd, :].T

    return Tr, x


def DLTcalib(nd, xyz, uv):
    '''
    Camera calibration by DLT using known object points and their image
points.
    Input
    -----
    nd: dimensions of the object space, 3 here.
    xyz: coordinates in the object 3D space.
    uv: coordinates in the image 2D space.
    The coordinates (x,y,z and u,v) are given as columns and the
different points as rows.
    There must be at least 6 calibration points for the 3D DLT.
    Output
    ------
     L: array of 11 parameters of the calibration matrix.
     err: error of the DLT (mean residual of the DLT transformation in
units of camera coordinates).
    '''
    if (nd != 3):
        raise ValueError('%dD DLT unsupported.' % (nd))

    # Converting all variables to numpy array
    xyz = np.asarray(xyz)
    uv = np.asarray(uv)

    n = xyz.shape[0]

    # Validating the parameters:
    if uv.shape[0] != n:
        raise ValueError('Object (%d points) and image (%d points) have
different number of points.' % (n, uv.shape[0]))

    if (xyz.shape[1] != 3):
        raise ValueError('Incorrect number of coordinates (%d) for %dD
DLT (it should be %d).' % (xyz.shape[1], nd, nd))
```

```
    if (n < 6):
        raise ValueError(
            '%dD DLT requires at least %d calibration points. Only %d
points were entered.' % (nd, 2 * nd, n))

    # Normalize the data to improve the DLT quality (DLT is dependent of
the system of coordinates).
    # This is relevant when there is a considerable perspective
distortion.
    # Normalization: mean position at origin and mean distance equals to
1 at each direction.
    Txyz, xyzn = Normalization(nd, xyz)
    Tuv, uvn = Normalization(2, uv)

    A = []

    for i in range(n):
        x, y, z = xyzn[i, 0], xyzn[i, 1], xyzn[i, 2]
        u, v = uvn[i, 0], uvn[i, 1]
        A.append([x, y, z, 1, 0, 0, 0, 0, -u * x, -u * y, -u * z, -u])
        A.append([0, 0, 0, 0, x, y, z, 1, -v * x, -v * y, -v * z, -v])

    # Convert A to array
    A = np.asarray(A)
    # Find the 11 parameters:
    U, S, V = np.linalg.svd(A)

    # The parameters are in the last line of Vh and normalize them
    L = V[-1, :] / V[-1, -1]
    # print(L)
    # Camera projection matrix
    H = L.reshape(3, nd + 1)
    # print(H)

    # Denormalization
    # pinv: Moore-Penrose pseudo-inverse of a matrix, generalized inverse
of a matrix using its SVD
    H = np.dot(np.dot(np.linalg.pinv(Tuv), H), Txyz)
    # print(H)
    H = H / H[-1, -1]
    # print(H)
    L = H.flatten()
    # print(L)

    # Mean error of the DLT (mean residual of the DLT transformation in
units of camera coordinates):
    uv2 = np.dot(H, np.concatenate((xyz.T, np.ones((1, xyz.shape[0])))))
    uv2 = uv2 / uv2[2, :]
    # Mean distance:
    err = np.sqrt(np.mean(np.sum((uv2[0:2, :].T - uv) ** 2, 1)))

    return L, err

def get_DLT_coefs2(obj_X, obj_Y, obj_Z, col_x, row_y):
    # implementation of DLT with with least squares
    # available on
    # https://github.com/acvictor/DLT/blob/master/DLT.py

    points3D = []
    points2D_image = []
    for i in range(len(obj_X)):
        pt3D = [obj_X[i], obj_Y[i], obj_Z[i]]
        points3D.append(pt3D)
```

```
            pt2D = [col_x[i], row_y[i]]
            points2D_image.append(pt2D)

    nd = 3
    P, err = DLTcalib(nd, points3D, points2D_image)
    return P[0:11], err # 12th element is 1

def least_square_solver(ras_list, ref_list):

    A = np.array(ras_list)
    l = np.array(ref_list)

    # print(A)
    # print(l)

    lambdas = np.linalg.inv(A.T.dot(A)).dot(A.T).dot(l) # Matlab: lambdas
= inv(A'*A)*A'*l
    return lambdas

def get_XML_points(xml_path):
    tree = ET.parse(xml_path)
    root = tree.getroot()
    ptns = []
    for ptn in root.iter("PtIm"): # find all
        pt = ptn.text.split()
        ptns.append((float(pt[0]), float(pt[1])))
    return ptns

def affine_trans(X, Y, coefs):
    x = X * coefs[0] + Y * coefs[1] + coefs[4]
    y = X * coefs[2] + Y * coefs[3] + coefs[5]
    # print(coefs)
    # print(X,Y)
    # print(x,y)
    return x, y

# development purposes only:
def get_affine_coefs(reference, targed):
    P = np.array(reference)
    nums = np.array(targed)
    print(P)
    print(nums)
    try:
        res_solved = np.linalg.solve(P, nums) # numpy solver, inverse
matrix is compute automatically
    except np.linalg.LinAlgError: # P is singular matrix, cant compute
inverse matrix, invalid configuration of poitns
        return False
    return res_solved

def affine_trans2(X, Y, coefs): # just testing purposes
    x = X * coefs[0] + Y * coefs[1] + coefs[2]
    y = X * coefs[3] + Y * coefs[4] + coefs[5]
    return x, y
```