UNIVERZITA KARLOVA
PŘÍRODOVĚDECKÁ FAKULTA

ALGORITMY POČÍTAČOVÉ KARTOGRAFIE

Generalizace budov

Jáchym Černík, Monika Novotná

1 Zadání

Generalizace budov metodami Minimum Area Enclosing Rectangle a PCA. Bonusové úkoly jsou zaneseny do Tabulky 1.

Bonusový úkol	**Ohodnocení**
Generalizace budov metodou Longest Edge.	5b
Generalizace budov metodou Wall Average.	8b
Generalizace budov metodou Weighted Bisector.	10b
Implementace další metody konstrukce konvexní obálky.	5b
Ošetření singulárního případu při generování konvexní obálky.	2b
Načtení vstupních dat ze *.shp.	10b

Tabulka 1: Bonusové úkoly

2 Popis a rozbor problému

2.1 Úvod

Generalizace je jedním z klíčových témat kartografie, slouží pro přizpůsobení detailu a zvyšuje čitelnosti mapy. Generalizace je ovlivněna několika faktory, jedná se např. o měřítko mapy nebo účel zobrazování. Při této metodě dochází ke zevšeobecnění objektů.

Generalizace má několik fází, přičemž první probíhá v terénu při sběru dat. Další fází je kartografická ganeralizace, kde dochází k úpravě nasbíraných dat, aby se informace staly ještě přehlednějšími. Automatizované metody generalizace využívají propracované algoritmy, které vyhlazují hrany, odstraňují nadbytečné budovy nebo stanoví hlavní orientaci objektů. Jednou z těchto metod je Minimum Area Enclosing Rectangle.

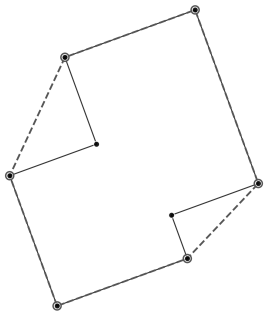
Při generalizování budov je důležité zachovat charakteristickou podobu a vzájemné vztahy s objekty v okolí. Při tomto procesu je nutné zvážit různé metody, které umí stanovit hlavní směr orientace, přičemž orientace vychází velmi často z nejdelší stěny budovy nebo z výrazně protáhlého tvaru (Duchene 2003). Z kartografického pohledu je důležité zachovat stejnou orientaci budovy po generalizaci, aby nebyla narušena čitelnost a vzájemná poloha objektů na mapě.

3 Konvexní obálka

Konvexní obálka je množina bodů s nejmnejším možným ohraničením, která obsahuje všechny body množiny. Podmínkou je, že pokud vybereme dva body uvnitř obálky a spojíme je, tak všechny další body, které se na úsečce nacházejí musí ležet v konvexní obálce (Bayer 2008). Příklad konvexní obálky lze vidět v obrázku 1, kde je vyzobrazena vytvořená konvexní obálka nad množinou bodů polygonu. V kartografii slouží konvexní obálka pro rozpoznávání tvarů a ke stanovení orientace budov. Algoritmy, které vytváří konvexní obálku se liší v náročnosti výpočtu a přesnosti. Výsledná obálka má pak mnoho využití, slouží např. pro detekci kolizí, analýzu tvaru objektů nebo pro klasifikaci (Bayer 2025)

3.1 Jarvis Scan

Jarvisův skan, známý též jako Gift Wrapping Algorithm, je jednou z klasických metod pro konstrukci konvexní obálky bodů v rovině ($\mathbb{R}^2$) i v prostoru ($\mathbb{R}^3$). Algoritmus je konceptuálně jednoduchý a snadno implementovatelný.



Obrázek 1: Příklad konvexní obálky na syntetických datech: Jarvis Scan (Vlastní Tvorba)

Princip metody spočívá v iterativním "obalování"množiny bodů S. Začíná se od zaručeného bodu na obálce (pivotu) a postupně se připojují další vrcholy. V každém kroku se z aktuálního vrcholu p_j hledá následující vrchol p_{j+1} tak, aby úhel ϕ mezi poslední hranou obálky $\mathcal{H}$ (tvořenou body p_{j-1}, p_j) a novou hranou (tvořenou body p_j, p_{j+1}) byl maximální. Tento bod p_{j+1} se vybírá ze všech bodů množiny bodů, které dosud nejsou součástí $\mathcal{H}$ (Bayer 2025). Celý postup Algoritmu lze vidět v Pseudokódu: Algoritmus 1.

Algorithm 1 Jarvis Scan - Konvexní obálka

Require: *pol* : polygon

$q \leftarrow$ bod s nejmenší y-souřadnicí ▷ najdi pivot
$pj \leftarrow q$
$px \leftarrow$ bod s nejmenší x- souřadnicí
$pj1 \leftarrow (px.x, pj.y)$
přidá pivot do konvexní obálky
while True **do**
    $\phi_{max} \leftarrow 0$ ▷ max úhel
    $idx_{max} \leftarrow -1$ ▷ index
    for všechny body p_i v *polygon* **do**
        if $pj \neq p_i$ **then**
            $\phi \leftarrow$ úhel mezi vektory $(pj1, pj)$ a (pj, p_i)
            if $\phi > \phi_{max}$ **then** ▷ najit max úhel
                $\phi_{max} \leftarrow \phi$ ▷ aktualizuj hodnotu
                $idx_{max} \leftarrow i$
            end if
        end if
    end for
    přidá bod idx_{max} do konvecní obálky
    $pj1 \leftarrow pj$
    $pj \leftarrow polygon[idx_{max}]$ ▷ aktualizace počátečního bodu
    if $pj == q$ **then** ▷ ukončí se až dobalí celý polygon
        break
    end if
end whilereturn *ch*

4 Popis algoritmů

4.1 Algoritmy generalizace budov

Detekce hlavních směrů budov je hlavním pilířem generalizace budov v počítačové kartografii, a to proto, aby generalizované budovy dodržovaly organizovanou strukturu ve vztahu k dalším objektům na mapě. Těmito objekty jsou například cesty či intravilány; antropogenní prvky na mapě jsou většinou tvořeny liniemi. Generalizace pomocí hlavního směru umožňuje zachovat tuto strukturu jakožto nástroj, který zajistí respektování polohy vůči ostatním prvkům. Nemělo by se tak stát, aby polygon budovy narušoval například linii železnice. Pro nalezení tohoto vektoru existuje mnoho metod, nicméně v této práci bylo vypracováno těchto pět:

- *Principal Component Analysis (PCA)*

- *Minimum Area Enclosing Rectangle (MBR)*
- *Longest edge*
- *Wall Average*
- *Weighted Bisector*

4.2 Principal Component Analysis

Analýza hlavních komponent je statistická metoda, která hledá hlavní směry (osy) největší variability v datech. Abychom mohli hledat hlavní směry nejprve potřebujeme znát kovarianční matici C, která má informace o rozměrech matice X.

$$C = \frac{1}{n-1} X^\top X$$

Pro nalezení vlastních vektorů a vlastních čísel používáme rozklad Singular Value Decomposition (SVD). Jedná se o ortogonální matice U a V, přičemž V je transponovaná, a diagonální matici singulárních hodnot Σ. Hlavní komponenty jsou tedy sloupce matice V a singulární hodnoty jsou dány diagonálními prvky matice Σ.

$$C = U \Sigma V^T$$

PCA tedy nachází dva největší směry rozptylu, které jsou na sebe kolmé. Pro generalizaci budov je tato metoda vhodná, protože hlavní komponenta se částečně shoduje s nejdelší osou polygonu.

V této práci bylo PCA využito při tvorbě optimalizovaného bounding boxu. Nejprve se sesbírají souřadnice polygonu a uloží se do matice, následně je spočtena kovarianční matice. Nad kovarianční maticí je proveden rozklad SVD, díky čemuž získáme hlavní osy polygonu. Polygon se následně otočí podle již známých hlavních os. Poté je vytvořen bounding box, který je rovnoběžný se zmíněnými osami. Vytvořený bounding box se vrací do původní pozice o stejný úhel jako předtím polygon. Postup této metody je zaznamenán v pseudokódu Algorithm 2

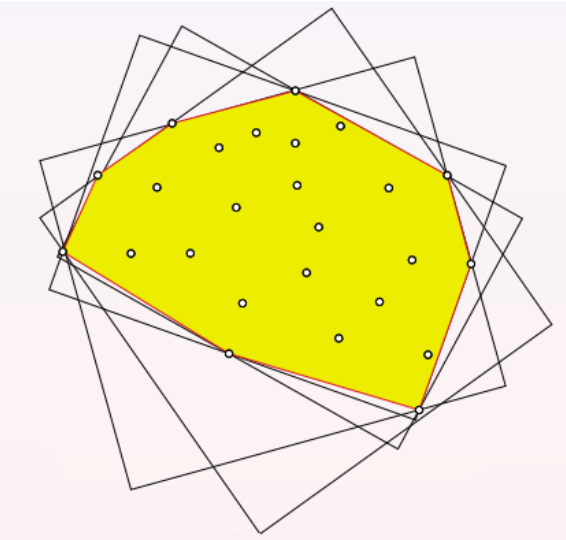
Algorithm 2 PCA - Analýza hlavních komponent

Require: *pol* : polygon

x, y = [], [] ▷ inticializace listů pro souřadnice
for p **do** pol:
    Konvertuj body na souřadnice
end for
$A = [x; y]$ ▷ Sestav matici
$C = \text{cov}(A)$ ▷ Vypočti kovarianční matici
$U, S, V = \text{SVD}(C)$ ▷ Proveď SVD rozklad
$\sigma = \arctan(V[0,1]/V[0,0])$ ▷ Urči hlavní směr
Otoč budovu o $-\sigma$
Vytvoř min-max box kolem otočené budovy
Změň velikost boxu na plochu původní budovy
Otoč box zpět o σ
return *pca_box*, σ

4.3 Minimum Area Enclosing Rectangle

Jak už název napovídá, principem metody *Minimum Area Enclosing Rectangle (MAER)* je že se usiluje o nalezení ohraničujícího obdélníku kolem polygonu budovy s co nejmenší plochou. Zmíněný obdélník má obalit množinu bodů v $\mathbb{R}^2$, avšak nikoliv přímo; nejprve je třeba upravit původní množinu bodů vytvořením konvexní obálky H, což se v přípdě tohoto vypracování provedo Jarvis Scanem z kapitoly 3.1 , kterou následně otočíme o úhel σ, který je směrnicí dané hrany. Poté algoritmus projde všechny hrany této konvexní obálky a hledá tzv. *min–max box* v každé otočené pozici (Obrázek 2), přičemž procházená hrana je vždy rovnoběžná s aktuálně vytvářenou hranou min–max boxu. Z vytvořených obdélníků je nakonec vybrán ten, který má nejmenší plochu.



Obrázek 2: Tvorba MMB v kažné poloze (Bayer 2025)

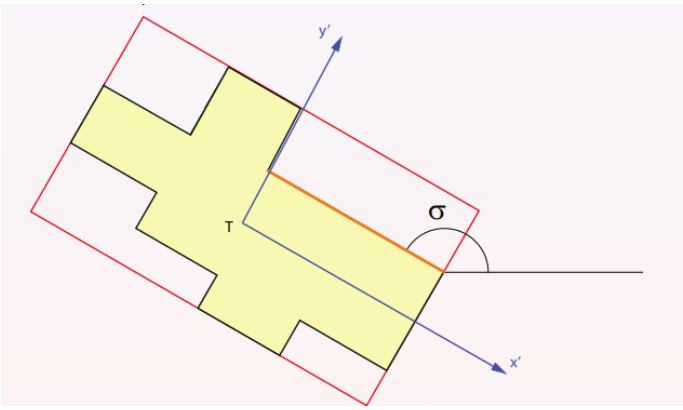
Algorithm 3 MBR - Minimum Bounding Rectangle

Require: *pol* : polygon

Vytvoř konvexní obal budovy (Jarvisův Scan)
Inicializuj min_plocha ← velké číslo
$\sigma_{\text{hlavní}} \leftarrow 0$
for každá hrana konvexního obálky **do**
    Urči směr hrany $\sigma = atan2(\Delta y/\Delta x)$ ▷ (Změna y/Změna x)
    Otoč obal o $-\sigma$
    Sestroj min-max box kolem otočeného obalu
    Vypočti plochu boxu
    if plocha < min_plocha **then**
        Ulož $\sigma_{\text{hlavní}} \leftarrow \sigma$
        Aktualizuj min_plocha ← plocha
        Ulož nejlepší box
    end if
end for
Změň velikost boxu podle plochy původní budovy
Otoč výsledný box zpět o $\sigma_{\text{hlavní}}$
return $mbr, \sigma_{\text{hlavní}}$

4.4 Longest Edge

Metoda Longest edge určuje hlavní směr na základě směru nejdelší hrany polygonu (Obrázek 3). Druhý hlavní směr je ortogonální k hlavnímu směru určenému nejdelší hranou polygonu. Po nalezení této hrany je vypočítán úhel mezi vektorem této hrany a vektorem osy x (směrnice). Tento úhel je pak využit pro rotaci původního polygonu o daný úhel. Následně je na rotovaném polygonu vytvořen min-max box, který je poté narotován zpět tak, aby byl kolineární s nejdelší hranou původního polygonu (Bayer 2025). Následně je otočný min-max box přizpůsoben velikosti původního polygonu budovy. Postup tohoto algoritmu je popsán v pseudokódu Algorithm 4.



Obrázek 3: Hlavní směr budový podle nejdelší hrany (Bayer 2025)

Algorithm 4 Longest Edge

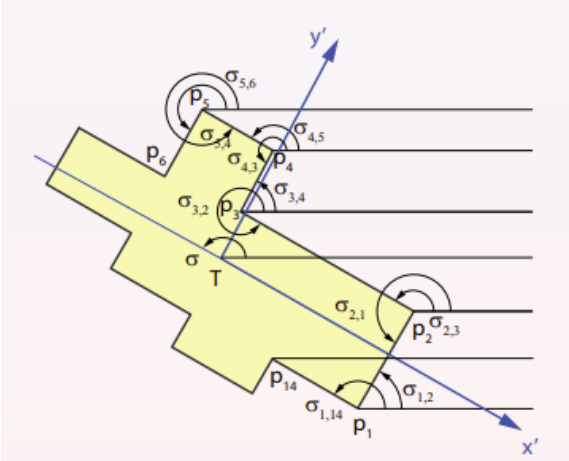
Require: *pol* : *polygon*

$n \leftarrow$ počet vrcholů polygonu
`délka` $\leftarrow 0$
for všechny vrcholy v polygonu **do**
    $p_1 \leftarrow$ vrchol i
    $p_2 \leftarrow$ vrchol $(i+1)$ mod n
    Spočítej délku mezi p_1 a p_2
    if délká mezi p_1 a p_1 je delší než nejdelší hrana ϵ **then**
        Nejdelší hrana $\leftarrow$ délka
        Vypočítej úhel nejdelší hrany
    end if
end for
Otoč budovu o směrnici nejdelší hrany
Vytvoř min-max box kolem otočené budovy
Změň velikost boxu na plochu původní budovy
Otoč box zpět o směrnici hlavní hrany
    return *mmb_r* ▷ vrátí finální generalizaci

4.5 Wall Average

Princip metody spočívá ve výpočtu váženého průměru úhlových orientací hran polygonu. Nejprve se pro každou hranu i určí její úhel orientace σ_i viz Obrázek 4. Na tyto úhly je následně aplikována operace modulo $\frac{\pi}{2}$, tj. $\sigma_i \pmod{\frac{\pi}{2}}$. Tím se získá odchylka orientace hrany od nejbližší osy v intervalu $[0, \frac{\pi}{2})$.

Z těchto „zbytkových“ úhlů se spočítá průměr, který je pak využit jako hlavní směr polygonu nad kterým se vytvoří min max box se změněnou velikostí Metoda je obecně robustní vůči drobným nepřesnostem, ale může být citlivá na výrazně, nepravé úhly delších hran, které mohou výsledek ovlivnit.



Obrázek 4: Určení úhlů σ_i orientace pro každou hranu (Bayer 2025)

Algorithm 5 Metoda průměrování stěn - Wall Average

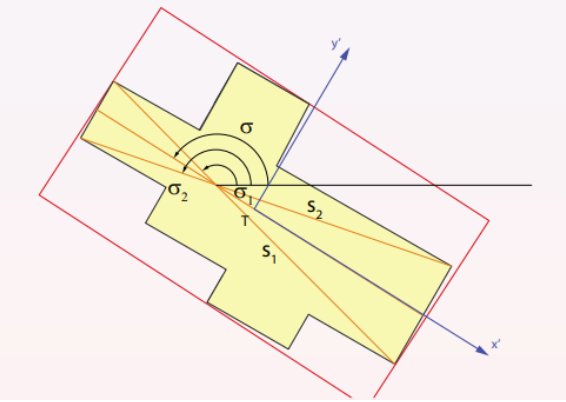
Require: *pol* : polygon

$\alpha_0 =$ úhel první hrany ▷ Základní orientace
$S_{rem} = 0$ ▷ inicializace součtu zbytků
for hrany v *pol* (kromě první hrany) **do**
    $\alpha_i =$ směrnice aktuální hrany
    $\Delta\alpha = |\alpha_i - \alpha_0|$ ▷ úhlový rozdíl
    $k_i = \lfloor 2 \cdot \Delta\alpha/\pi \rfloor$ ▷ nejbližší násobek k $\pi/2$
    $rem_i = \Delta\alpha - k_i \cdot \pi/2$ ▷ výpočet zbytku pro aktuální iteraci
    S_{rem} += rem_i ▷ aktualizace zbytků
end for
$\alpha_{avg} = \alpha_0 + S_{rem}/(n-1)$ ▷ průměrný úhel(hlavní směr)
Otoč budovu o $-\alpha_{avg}$
Vytvoř min-max box kolem otočené budovy
Změň velikost boxu na plochu původní budovy
Otoč box zpět o α_{avg}
return $wall\_avg\_box, \alpha_{avg}$

4.6 Weighted Bisector

Stejně jako ostatní metody, tento algoritmus hledá globální orientaci polygonu. Weighted Bisector hledá dvě nejdelší úhlopříčky v polygonu, které následně používá pro detekci hlavních směrů. Jako první se prochází všechnyoúpop dvojice bodů a počítá jejich vzdálenost. Následně se pro dvě nejdelší spočítá délka (d_1 a d_2) a směrnice (σ_1 a σ_2) dvou nejdelších úhlopříček (Obrázek 5). Hlavní směr budovy je dán průměrem směrnic vážený jejich vzdáleností.

$$\sigma = \frac{s_1\,\sigma_1 + s_2\,\sigma_2}{s_1 + s_2}\,.$$



Obrázek 5: Směrnice a délka nejdelších úhlopříček (Bayer 2025)

Algorithm 6 Weighted Bisector

Require: *pol* : *Polygon*

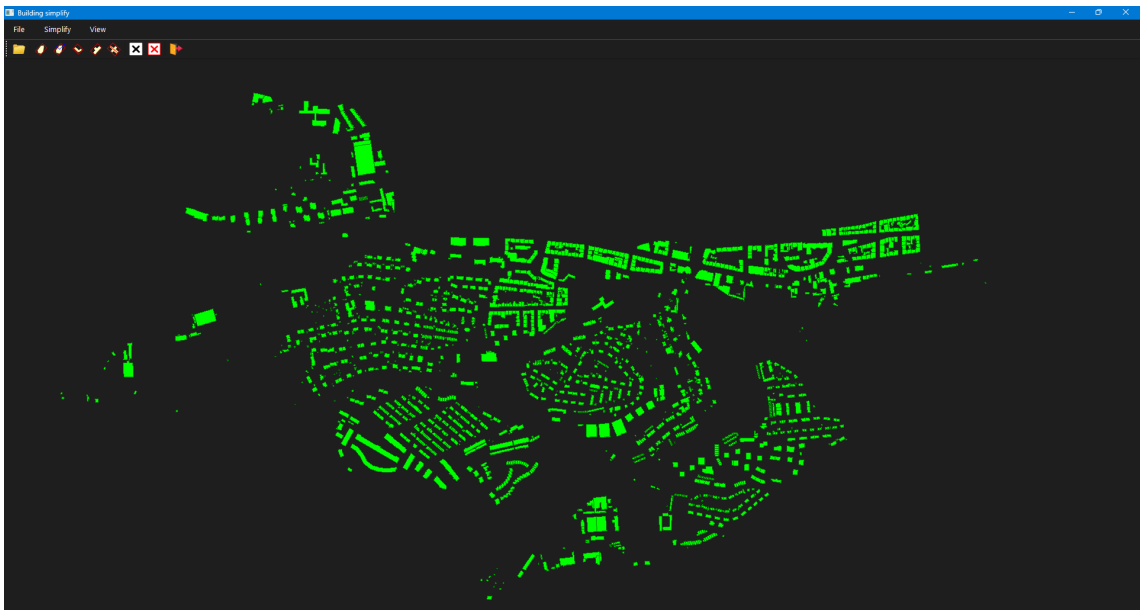
procedure WEIGHTED BISECTOR(polygon P)
    Vypočítej délky všech úhlopříček polygonu P
    Najdi dvě nejdelší úhlopříčky d_1, d_2 s délkami L_1, L_2
    Vypočítej úhly σ_1, σ_2 těchto úhlopříček (upravené o $-\pi/2$)
    Vypočítej vážený průměr úhlů $\alpha_w = (L_1 \cdot \beta_1 + L_2 \cdot \beta_2)/(L_1 + L_2)$ ▷ Delší úhlopříčka má větší váhu
    Otoč budovu o $-\alpha_{avg}$
    Vytvoř min-max box kolem otočené budovy
    Změň velikost boxu na plochu původní budovy
    Otoč box zpět o α_{avg}
    return wb_box, α_{avg}
end procedure

5 Implementace

Za pomocí nástroje QT Creator byla vytvořena aplikace *Building Symplify* pro demonstrování jednotlivých generalizačních metod. Aplikace byla přizpůsobena pro import prostorových dat formátu .shp a pro následnou generalizaci nad těmito daty.

5.1 Vstupní data

Jako vstupní data byla použita shp vrstva obsahující budovy. Vrstva s polygony byla stažena z geoportálu Prahy v rámci geodatabáze ZABAGED (*Obrázek 6*)



Obrázek 6: Ukázka načtených dat (.shp) (Košíře)

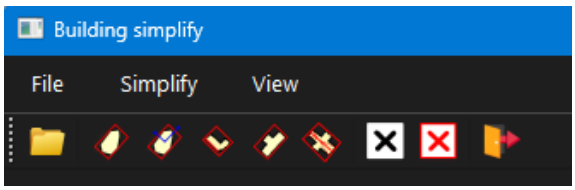
5.2 Implementace nástrojů pro nahrání dat

Pro účely načtení geodat ze souborů formátu shapefile byla využita knihovna GeoPandas. Celý proces je propojený ve všech třídách: `MainForm.py`, `draw.py` a `algorithms.py`. V `MainForm.py` je definováno uživatelské rozhraní, včetně akce pro otevření shapefile, která spouští metodu `loadData` ve třídě `Draw`. Tato metoda, definovaná v `draw.py`, zobrazí dialog pro výběr souboru a následně pomocí `geopandas.read_file(filename)` načte data ze shapefile do `GeoDataFrame`. Z tohoto objektu jsou extrahovány celkové ohraničení dat, které jsou následně využity pro transformaci a přípravu dat k zobrazení.

Transformace dat probíhá v metodě `resizeData`, která získá rozměry vykreslovacího plátna, vymaže existující polygony a vypočítá škálovací faktory na základě ohraničení dat. Následně iteruje skrze každý řádek v `GeoDataFrame` a pro každý geometrický objekt (Polygon nebo MultiPolygon) volá metodu `__add_polygon`. Tato metoda převede geometrii na `QPolygonF`, získá souřadnice bodů a transformuje je do pixelového systému plátna, zahrnující posun, škálování a inverzi osy `y`. Vykreslení dat zajišťuje metoda `paintEvent`, která vykresluje načtené a případně zjednodušené polygony.

5.3 Interakce uživatele a analýza polohy bodu

1. **Inicializace:** Při zpuštění se uživateli spustí plocha s horní lištou kde se nacházejí nástroje
2. **Načtení dat:** Následně může kliknout na tlačítko `Open` v horní liště nebo zvolit v menu `File` volbu načtení souboru ve formátu Shapefile (`.shp`). Po úspěšném načtení se zobrazí polygony v zelené barvě se zeleným okrajem.
3. **Uživatelské ovládání:** Uživatel může navigovat plátno pomoci potažení skrz pravé tlačítko a může přiblížit nebo oddálit pomocí kolečka na myši.
4. **Generalizace:**
   - **Generalizace:** Pro inicializaci generalizace uživatel klikne na jednu z metod v horní liště
   - Pro generaizaci budov lze využít metodu PCA, MBR, Wall Average, Longest Edge či Weighted Bisector (volitelné jak z menu `Analyze`, tak z nástrojové lišty(*Obrázek 7*)).
5. **Výstup analýzy:**
   - Po zpracování uživatelem požadované metody se zobrazí nová vrstva generalizovaných budov nad původními polygon. Tyto nové polygony jsou modré barvy s tmavým okrajem
   - Dodatečně se zobrazí vyskakovací okénku s výslednou přesností dané metody.
6. **Nová analýza:** `Clear data` V horní liště lze také vymazat výsledek či všechna data a začít znovu.



Obrázek 7: kontrolní prvky aplikace

6 Třídy a metody

Funkčnost aplikace zajišťují tři nezbytné třídy. Jedná se o algorithms.py, draw.py a mainForm.py.

6.1 Třída Algorithms

Třída `Algorithms` implementuje algoritmy pro zpracování polygonů budov, včetně:

- Konstrukce konvexní obálky metodou `jarvis_scan`
- Generalizace polygonů metodami:
  - `createMBR` - Minimální ohraničující obdélník
  - `createBRPCA` - Analýza hlavních komponent
  - `longestEdge` - Metoda nejdelší hrany
  - `wallAverage` - Průměrování stěn
  - `weighted_bisector` - Vážená osa úhlopříček
- Pomocné metody pro rotaci (`rotate`), výpočet plochy (`getArea`) a přesné měření úhlů (`get2VectorsAngle`)

Hlavní funkcionalita:

- Vstup: `QPolygonF` reprezentující budovu
- Výstup: Zjednodušený polygon a hlavní směr (úhel σ)
- Metody vracejí dvojici (`QPolygonF`, `float`)
- Podporuje výpočet přesnosti generalizace (`calculate_accuracy`)

Seznam funkcí třídy Algorithms:

`generalize_pols(self, polygons, method='MBR')`: Aplikuje generalizaci na seznam polygonů a vypočítá přesnost. Podporuje metody 'MBR', 'PCA', 'LongestEdge', 'WallAverage' a 'WeightedBisector'. Vrací tuple s generalizovanými polygony.

`calculate_accuracy(self, pol: QPolygonF, sigma: float)`: Vypočítá metriky přesnosti na základě úhlové odchylky vůči hlavnímu směru.

`get2VectorsAngle(self, p1: QPointF, p2: QPointF, p3: QPointF, p4:QPointF)`: Vypočítá úhel mezi dvěma vektory definovanými čtyřmi body.

`createCH(self, polygon: QPolygonF)`: Konstruuje konvexní obal pomocí Jarvisova Skanu. Začíná od bodu s nejnižší y-souřadnicí a iterativně hledá bod s maximálním úhlem.

`rotate(self, pol: QPolygonF, sigma)`: Rotuje polygon o zadaný úhel sigma pomocí rotační matice. Zachovává pořadí bodů a řeší speciální případy prázdného polygonu.

`createMMB(self, pol:QPolygonF)`: Vytváří min-max box (ohraničující obdélník) kolem polygonu. Vrací polygon a plochu boxu.

`getArea(self, pol: QPolygonF)`: Vypočítá plochu polygonu pomocí lichoběžníkové metody. Zohledňuje absolutní hodnotu a speciální případy malých polygonů.

`resizeRectangle(self, building:QPolygonF, mbr:QPolygonF)`: Změňuje velikost obdélníku tak, aby odpovídal ploše původní budovy.

`createMBR(self, building: QPolygonF)`: Vytváří minimální ohraničující obdélník (MBR) kolem budovy. Kombinuje konvexní obal a rotace Vrací polygon a hlavní směr.

`longestEdge(self, pol: QPolygonF)`: Zjednodušuje budovu pomocí metody nejdelší hrany. Hledá nejdelší hranu, rotuje podle jejího úhlu a vytváří ohraničující box. Vrací polygon a hlavní směr.

`wallAverage(self, pol: QPolygonF)`: Aplikuje metodu průměru stěn. Vypočítá vážený průměr úhlů stěn a vytváří ohraničující box podle tohoto směru. Vrací polygon a hlavní směr.

`weighted_bisector(self, pol: QPolygonF)`: Používá metodu weighted bisector. Hledá platné diagonály uvnitř polygonu a vytváří box na základě jejich váženého průměru.

`createBRPCA(self, building: QPolygonF)`: Implementuje PCA metodu pro zjednodušení budovy. Používá SVD rozklad k nalezení hlavních směrů a vytváří optimalizovaný ohraničující box.

6.2 Třída Draw

Třída `Draw` zajišťuje vizualizaci geografických dat a interakci s uživatelem. Načítá polygony budov ze shapefile pomocí GeoPandas (metoda `loadData`) a transformuje jejich souřadnice pro zobrazení (`resizeData`). Vykresluje původní polygony zeleně generalizované modře. Umožňuje zoom kolečkem myši, panování tažením. Metody `clearAllData` a `clearResults` resetují data nebo pouze výsledky generalizace. Je propojená s třídou `Algorithms` a `MainForm` pro vykreslení výsledků generalizace.

Seznam funkcí třídy draw:

`loadData(filename=None)`: Načte data ze shapefile pomocí GeoPandas a transformuje je do pixelových souřadnic

`resizeData()`: Přizpůsobí data rozměrům widgetu.

`__add_polygon(geom, ...)`: Přidá polygon do seznamu s transformací souřadnic

`mousePressEvent(e)`: Zaznamená pozici kliknutí a spustí překreslení

`wheelEvent(e)`: Zajišťuje zoom pomocí kolečka myši

`mouseMoveEvent(e)`: Implementuje panování tažením myší

`paintEvent(e)`: Vykreslí původní (černé), zjednodušené (červené) polygony a konvexní obálky (modré)

`getPolygons()`: Vrátí seznam načtených polygonů

`setSimplifiedPolygons(polygons)`: Nastaví zjednodušené polygony pro vizualizaci

`clearAllData()`: Smaze všechna data a resetuje zoom/pan

`clearResults()`: Odstraní pouze výsledky generalizace

`resetView()`: Obnoví výchozí zobrazení

6.3 Třída MainForm

V této třídě je vytvořeno okno se samotnou aplikací ve které celá analýza probíhá. Program je vytvořen s pomocí knihovny PyQt6, což zajišťuje otevření okna s ovládacími prvky. Hlavní částí je plátno, které slouží pro zobrazení polygonů a bodů. Nástrojová lišta obsahuje prvky pro tvorbu analýzy.

Seznam funkcí třídy MainForm:

`openShapefile(self)`: Načte data ze shapefile a zobrazí je v canvasu. Zobrazuje stavové zprávy o úspěchu/neúspěchu načítání.

`_show_accuracy_popup(self, method_name, accuracy1, accuracy2, percentage1, percentage2)`: Zobrazí dialogové okno s výsledky přesnosti generalizace.

`_simplify_building(self, method_name, algorithm_func, generalize_method_key=None)`: Generická funkce pro zjednodušení budov. Volá specifické algoritmy, zpracovává výsledky a aktualizuje zobrazení.

`simplifyBuildingMBR(self)`: Zjednoduší budovu pomocí MBR (Minimal Bounding Rectangle) metody. Volá `_simplify_building` s příslušným algoritmem.

`simplifyBuildingBRPCA(self)`: Zjednoduší budovu pomocí PCA (Principal Component Analysis) metody. Volá `_simplify_building` s příslušným algoritmem.

`simplifyBuildingLongestEdge(self)`: Zjednoduší budovu pomocí metody nejdelší hrany. Volá `_simplify_building` s příslušným algoritmem.

`simplifyBuildingWallAverage(self)`: Zjednoduší budovu pomocí metody průměru stěn. Volá `_simplify_building` s příslušným algoritmem.

`simplifyBuildingWeightedBisector(self)`: Zjednoduší budovu pomocí metody váženého půlení. Volá `_simplify_building` s příslušným algoritmem.

`clearResults(self)`: Vymaže pouze výsledky zjednodušení (generalizované polygony), zachová původní data. Aktualizuje stavovou zprávu.

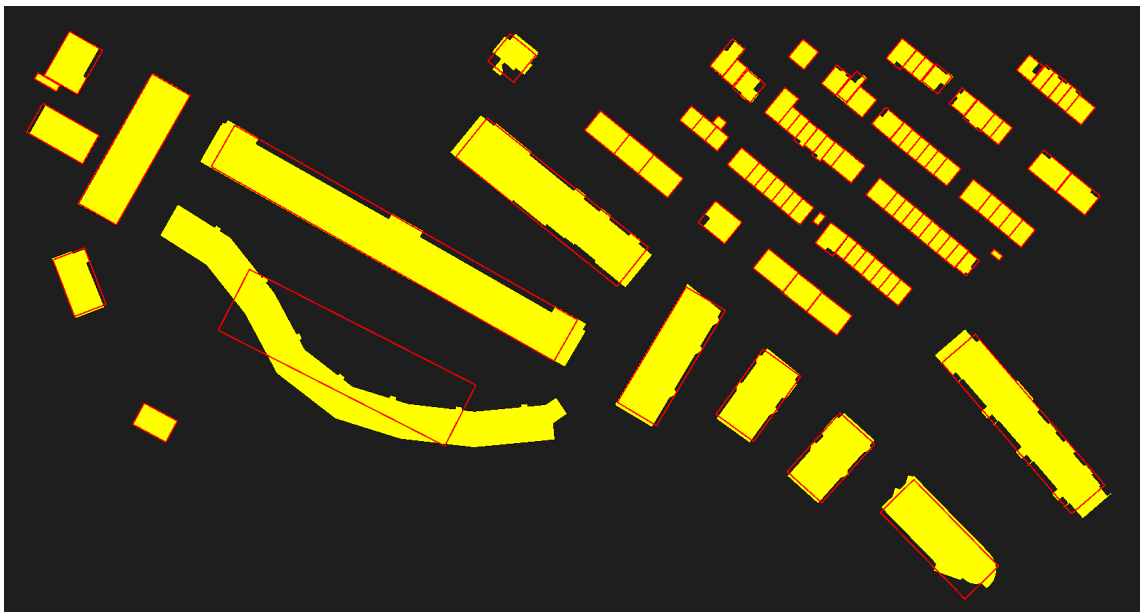
`clearAll(self)`: Vymaže všechna data včetně původních polygonů. Kompletní reset aplikace. Aktualizuje stavovou zprávu.

`exitApplication(self)`: Ukončí aplikaci. Volá Qt funkci pro ukončení programu.

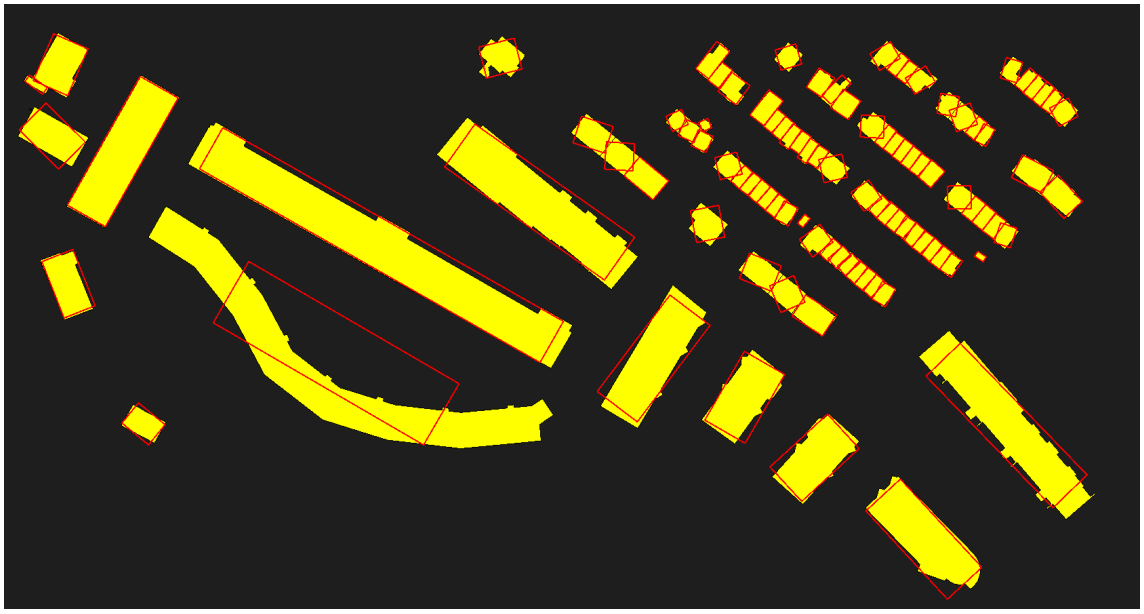
7 Výsledky

Nad vybranými daty (*Obrázek 6*) byly aplikovány metody vyhledávání hlavních směrů. V následující kapitole jsou popsány kvality výsledků těchto jednotlivých metod při generalizaci budov a to jak vizuální tak v kvantitativním měřítku.

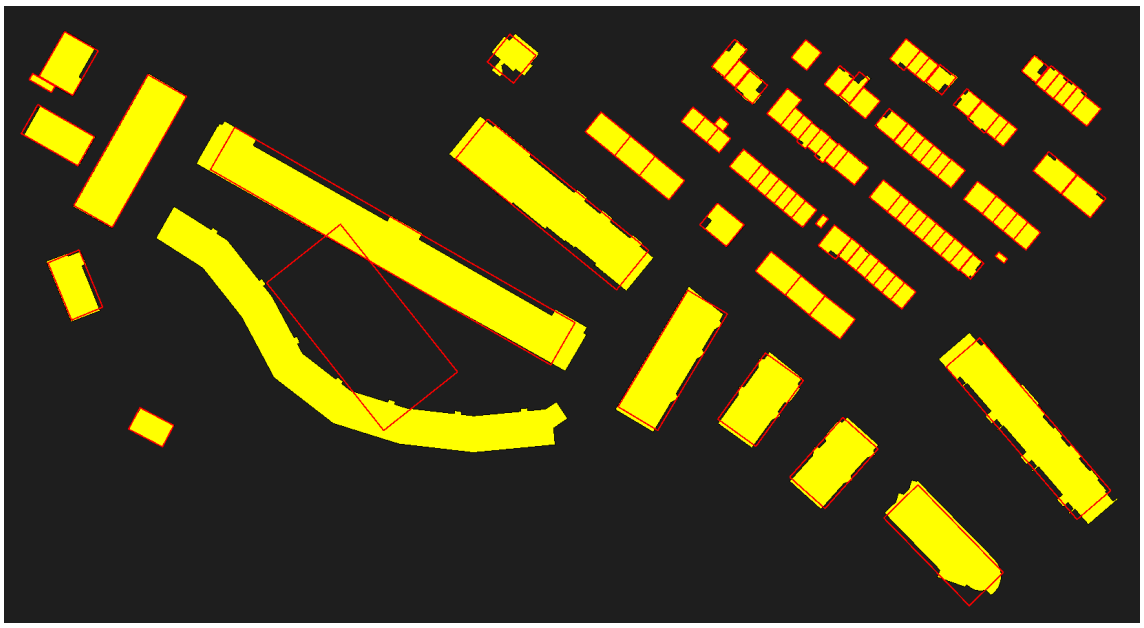
Městská část Praha-Košíře se vyznačuje značnou heterogenitou zástavby, která zahrnuje vilové čtvrti, starší i moderní sídlištní celky a historické usedlosti, jako je například Cibulka. Právě tato rozmanitost činí dané území vhodným kandidátem pro testování generalizačních algoritmů, neboť umožňuje ověřit jejich adaptabilitu na různé typy urbánních struktur. Z tohoto důvodu byla pro detailní analýzu vybrána lokalita v blízkosti autobusové zastávky Naskové, kde se kombinuje jak starší, tak nová zástavba. Vizuální prezentace výsledků aplikace jednotlivých metod v této oblasti je uvedena na obrázcích 8 až 12.



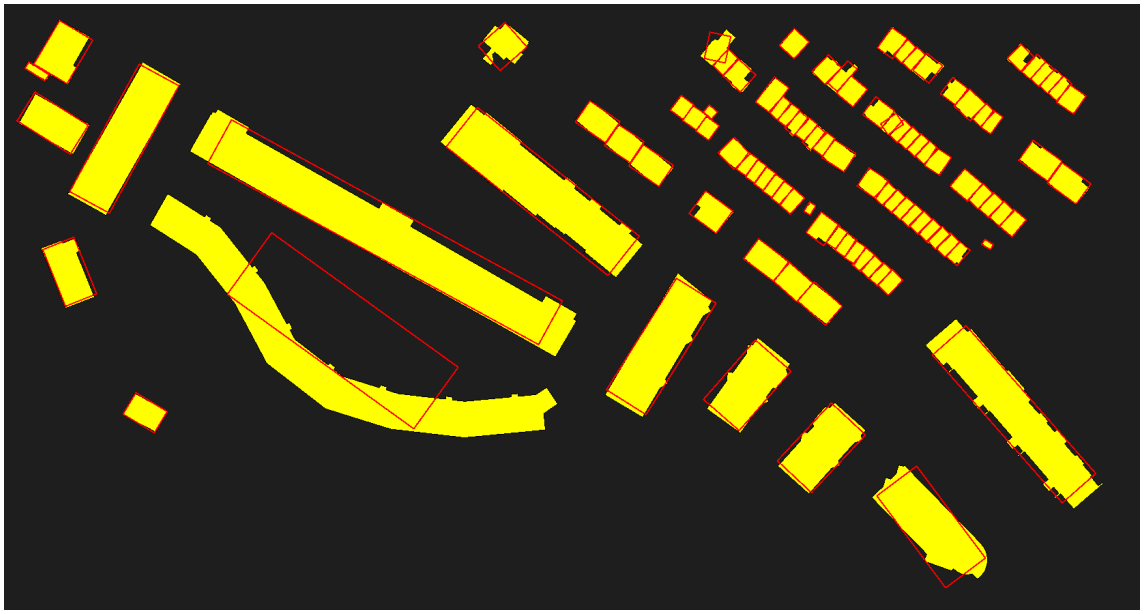
Obrázek 8: Výsledek:Minimum Bounding Rectangle



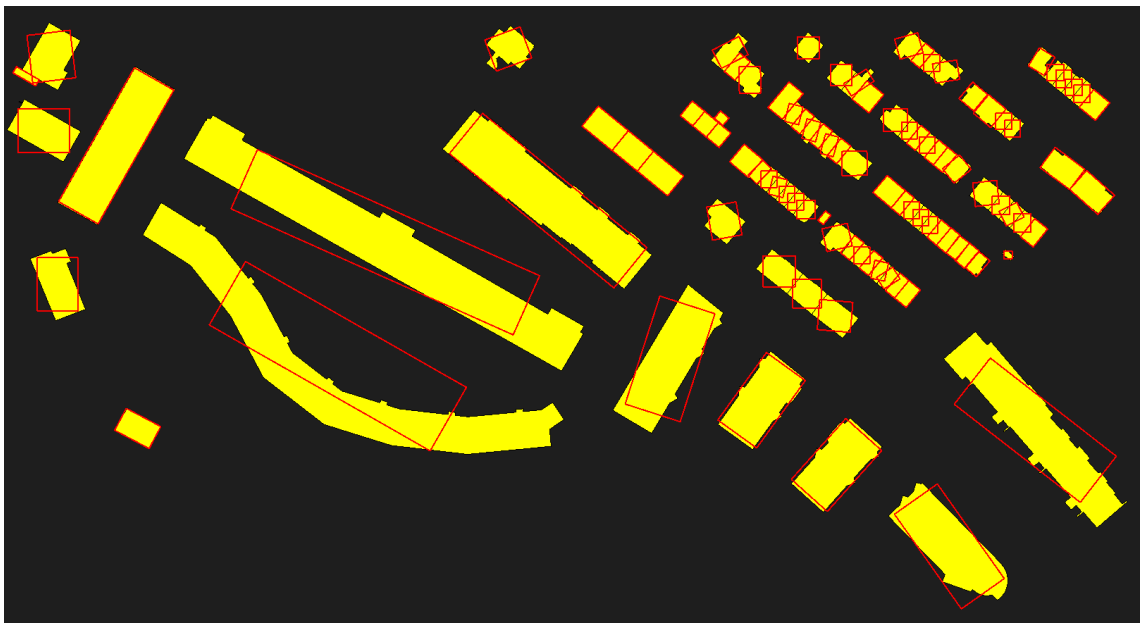
Obrázek 9: Výsledek: PCA



Obrázek 10: Výsledek: Longest Edge



Obrázek 11: Výsledek: Wall Average



Obrázek 12: Výsledek: Weighted Bisector

Výpočet přesnosti hodnotí, jak moc se tvar původní budovy blíží ideálnímu pravoúhlému tvaru orientovanému podle zadaného hlavního směru. Pro každou stěnu původní budovy změří, o kolik se její směr liší od hlavního směru. Z těchto jednotlivých úhlových odchylek pak vypočítá dvě celkové číselné hodnoty (označené $\Delta\sigma_1$ a $\Delta\sigma_2$), které popisují průměrnou odchylku a rozptyl těchto odchylek pro celou budovu. Tyto metriky tedy kvantifikují, jak dobře původní tvar odpovídá pravoúhlé struktuře v dané orientaci. Spolu s těmito dvěma hodnotami se vypočítají i odpovídající procentuální skóre, která vyjadřují míru shody. V tabulce 2 lze vidět průměrné hodnoty těchto metrik nad danou datovou sadou.

Tabulka 2: Metriky přesnosti

Metoda	**Avg $\Delta\sigma_1$ (°)**	**Avg $\Delta\sigma_1$ (%)**	**Avg $\Delta\sigma_2$ (°)**	**Avg $\Delta\sigma_2$ (%)**
MBR	4.19	90.7	5.29	88.2
PCA	4.26	90.5	5.39	88.0
Longest Edge	4.20	90.7	5.33	88.2
Wall Average	4.19	90.7	5.31	88.2
Weighted Bisector	5.39	88.0	6.99	84.6

8 Závěr, možné či neřešené problémy, náměty na vylepšení

8.1 Shrnutí výsledků

V této práci bylo použito několik algoritmů, které generalizují polygony. Po analýze lze říci, že se algortimy se liší v přesnosti.

Metoda Minimum Bounding Rectangle (Obrázek 8) se ukázala jako velmi efektivní pro tvorbu generalizovaných budov. Při pohledu na budoy sousedící vedle sebe, lze vidět, že hlavní směr budov je poměrně přesně dodržen a vytvořené polygony korespondují s původními. Pokud jde o protáhlé budovy s výstupky, algoritmus velmi dobře identifikuje hlavní směr a podle výstupků pak minimálně rotuje nově vytvořený polygon. S nejzajímavějším tvarem - výřezu, zahnutou budovou si algoritmus příliš neporadil, jelikož budova je velmi zkrácená.

Druhým nejspolehlivějším algoritmem byl Wall Average (Obrázek 11), kde až na vyjímky, zpracovává velmi dobře spojené budovy. Generelizace protáhlých budov vychází velmi podobně jako v předchozím případě. Zahnutá budova se na první pohled liší, v tomto případě jde o deformaci původního tvaru, přičemž dostáváme polygon ležící uprostřed původní polohy vstupní budovy.

Následuje Longest Enge (Obrázek 10), který opět velmi dobře vykresluje pravidelné budovy, ale pokud se dostane k zahnutým nepravidleným tvarům, přesnost klesá. Nově vytvořený polygon zahnuté budovy se velmi posunul a nekoresponduje s klavním směrem budovy.

Algoritmus PCA (Obrázek 9) si dokáže velmi poradit s pravidelnými budovami s jasně daným hlavním směrem. Pokud je však vstupní polygon blízko čtverci, metoda zaměňuje hlavní osy a výsledek rotuje až o 90°. Se zahnutým tvarem si poradí poněkud dobře, kvůli jasně danému protažení.

Poslední metodou je Weighted Bisector (Obrázek 12), kde dochází ke špatnému určení směru téměř u všech budov. Správný výsledek lze vidět pouze u jednoduchých pravoúhlých polygonů.

Vizuální výsledky jsou v souladu s vypočítanými kvantitativními metrikami (odchylek) a obecně by se dalo říct, že pro testovanou lokalitu je nejlepší metodou algoritmus MBR. Na druhém místě se nachází metoda Wall Average a nejhorší metodou je Weighted Bisector, ačkoliv tato metoda by se dle názoru autorů dala ještě vylepšit.

8.2 Omezení současné implementace a Náměty na vylepšen

Nejobtížnější při vypracování této práce byla implementace metody Weighted Bisector a hodnocení přesnosti. Jako budoucí vylepšení by bylo možné implementovat detekci pouze vnitřních diagonál budov jak je v metodě avizováno. Co se týče přesnosti, vhodnější by byl proces, který zvýrazní jednotlivé generalizace podle velikosti jejich odchylek, protože dosavadní průměrování všech výsledků způsobuje ztrátu informace o úspěšnosti daných metod. Program by tak snáze ukazoval slabiny a výhody jednotlivých metod. Daná prahová hodnota pro úspěšnou generalizaci by mohla být například odchylka 10 stupňů. V těžišti každého polygonu by se tak mohl zobrazit grafický bod, jehož barva by indikovala úspěšnost použité metody.

Uznání

Při přípravě této práce byla využita asistence umělé inteligence, konkrétně nástroje Gemini. které pomohly s úpravou syntaxe, řešením chyb a při konvertování vlastního pseudokódu do programovacího jazyka Python.

Reference

[1] BAYER, T. (2008). *Algoritmy v digitální kartografii*. Nakladatelství Karolinum, Praha.

[2] BAYER, T. (2025). *Konvexní obálky.*. Přednáška pro předmět Algoritmy počítačové kartografie, Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK. Dostupné z: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4_new.pdf [cit. 12. 4. 2025].

[3] GOOGLE(2025) Google Ai Studio, Gemini 2.5 Pro Experimental. Dostupné z https://aistudio.google.com/prompts/new_chat

[4] IPR (2023) *Vybraná data budov ZABAGED, stíhnuté z Geoportálu Prahy* https://geoportalpraha.cz/data-a-sluzby/9a3b9cdb43824509917f22e08d29c4a1_0

[5] DUCHENE, C. et al. 2003: Quantitative and qualitative description of building orientation.