Přírodovědecká fakulta
Univerzita Karlova

Úkol č. 2: Generalizace budov LOD0

Algoritmy počítačové kartografie

Eliška Králová, Eliška Pospěchová

1.N-GKDPZ

Praha 2025

1 Zadání

Úloha č. 2: Generalizace budov LOD0

Vstup: množina budov $B = \{B_i\}_{i=1}^{n}$, $B_i = \{P_{i,j}\}_{j=1}^{m}$.

Výstup: $G(B_i)$.

Ze souboru načtěte vstupní data představovaná lomovými body budov a proveďte generalizaci budov do úrovně detailu LOD0. Pro tyto účely použijte vhodnou datovou sadu, např. ZABAGED, testován provďte nad třemi datovými sadami (historické centrum, sídliště, izolovaná zástavba).

Pro každou budovu určete její hlavní směry metodami:

- Minimum Area Enclosing Rectangle,
- PCA.

U první metody použijte některý z algoritmů pro konstrukci konvexní obálky. Budovu při generalizaci do úrovně LOD0 nahraďte obdélníkem orientovaným v obou hlavních směrech, se středem v těžišti budovy, jeho plocha bude stejná jako plocha budovy. Výsledeky generalizace vhodně vizualizujte.

Otestujte a prorovnejte efektivitu obou metod s využitím hodnotících kritérií. Pokuste se rozhodnout, pro které tvary budov dávají metody nevhodné výsledky, a pro které naopak poskytují vhodnou aproximaci.

Hodnocení:

Krok	**Hodnocení**
Generalizace budov metodami Minimum Area Enclosing Rectangle a PCA.	15b
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 singulárních případů při generování konvexní obálky.	*+2b*
*Načtení vstupních dat ze *.shp.*	*+10b*
Max celkem:	**55b**

2 Údaje o bonusových úlohách

V rámci této úlohy jsou řešeny tyto bonusové úlohy: *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), *Načtení vdtupních dat ze *.shp* (10b).

3 Popis a rozbor problému

Generalizace budov

Generalizace je proces, jehož cílem je zjednodušit jednotlivé objekty v závislosti na velikosti měřítka mapy. Tedy například aby při oddalování mapy byly objekty zakresleny s menším detailem. Generalizace budov zahrnuje různé aspekty, mezi něž patří zjednodušení geometrie, seskupování blízkých objektů, změna velikosti a polohy, výběr zobrazovaných objektů a zachovaní charakteristických rysů. Při těchto metodách jsou však vždy zachovány topologické vztahy objektů. Prakticky se generalizace využívá například v kartografii, GIS nebo navigačních systémech. K automatizaci generalizace se používají především algoritmy *Minimum Area Enclosing/Bounding Rectangle*, *Principal Component Analysis*, *Longest Edge*, *Weighted Bisector* nebo *Wall Average*. Všechny zmíněné algoritmy se využívají i pro detekování hlavního směru objektu, neboť základem generalizace je také zachování orientace vzhledem k ostatním objektům. U nich byla zachována též stejná plocha mezi budovou A po provedení vybraného algoritmu a původní budovou A_b. Výsledné vrcholy V_i' vytvořeného obdélníku jsou spočteny pomocí těžiště T, směrových vektorů u_i a poměru ploch k:

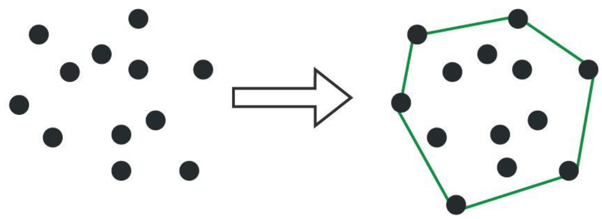
$$V_i' = T + u_i' = T + \sqrt{k}u_i, \tag{1}$$

kde poměr ploch a směrový vektor:

$$\begin{aligned} k &= \frac{A_b}{A} \\ u_i &= V_i - T. \end{aligned} \tag{2}$$

Konvexní obálka

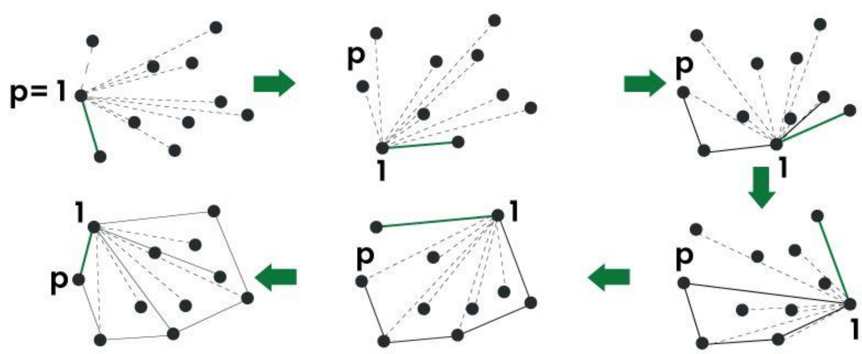
Konvexní obálka je taková množina bodů, která je minimální a všechny body leží uvnitř tohoto konvexního mnohoúhelníku (*Obrázek 1*). Pro konvexní mnohoúhelník platí, že všechny jeho vnitřní úhly nejsou větší než 180°. Konvexní obálka slouží jako pomocná struktura některých algoritmů a používá se například při plánování pohybu robotů nebo detekci tvaru a natočení objektů. Mezi metody pro konstrukci konvexní obálky patří například *Jarvis Scan*, *Graham Scan*, *Quick Hull*, *Inkrementální konstrukce* nebo *Divide and Conquer*.



Obrázek 1: Konvexní obálka (zdroj: GeeksforGeeks 2024)

Jarvis Scan

Jarvis Scan (*Obrázek 2*) je metoda konstrukce konvexní obálky vhodná pro množiny bodů menších velikostí. Principem je maximalizace úhlů ω mezi krajními body množiny. Algoritmus vezme první bod q (tzv. pivot) o minimální y-souřadnici, který s jistotou leží na konvexní obálce. Jako další bod je vybrán takový, který leží co nejvíc vlevo od pivota (co nejmenší x-souřadnice). Následně se do konvexní obálky přidávají body, které svírají s poslední dvojicí bodů v obálce největší úhel ω. Tento proces se opakuje do té doby, než se dostaneme k počátečnímu bodu (pivot).



Obrázek 2: Algoritmus Jarvis Scan (zdroj: GeeksforGeeks 2024)

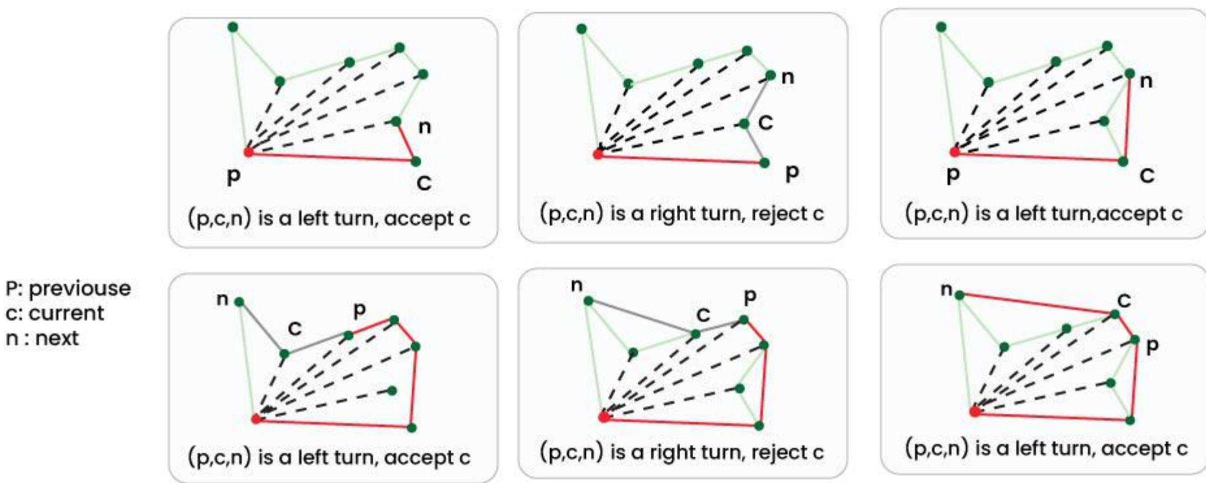
$$p_{j+1} = arg\ max_{\forall p_i \in P} \angle\ (p_{j-1}, p_j, p_i) \tag{3}$$

Graham Scan

Graham Scan (*Obrázek 3*) je oproti předchozí metodě vhodný pro větší množiny bodů. V prvním kroku je nalezen výchozí bod (pivot) stejným způsobem jako u metody Jarvis Scan. Dále jsou všechny body z množiny seřazeny podle velikosti úhlu ω, který svírají s přímkou procházející pivotem a rovnoběžnou s osou x. Takto seřazené body jsou následně procházeny a je u nich vyhodnocováno kritérium levotočivosti:

$$p_j \begin{cases} \notin \mathcal{H}, & p_{j+1} \in \sigma_r(p_{j-1}, p_j), \\ ? \in \mathcal{H}, & p_{j+1} \in \sigma_l(p_{j-1}, p_j). \end{cases} \tag{4}$$

Jestliže se následující bod p_{j+1} nachází napravo (pravá polorovina σ_r) od přímky procházející posledními dvěmi body, je do konvexní obálky přídán právě procházený bod a poslední bod p_j je naopak z konvexní obálky vyjmut.



Obrázek 3: Algoritmus Graham Scan (zdroj: GeeksforGeeks 2025)

Minimum Area Enclosing/Bounding Rectangle

Základní účel tohoto algoritmu je detekce natočení budovy. Pomocí algoritmu se také hledá nejmenší obdélník obklopující množinu bodů P. Nejprve je nutné vytvořit kolem množiny konvexní obálku, kterou následně rotujeme o úhel -σ tak, aby postupně byla každá hrana rovnoběžná s osou x:

$$P_0 = R(-\sigma)P \Rightarrow \begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} cos(-\sigma) & -sin(-\sigma) \\ sin(-\sigma) & cos(-\sigma) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{5}$$

Po každém takovém otočení se spočte plocha obdélníku, kdy jako výsledek je brán takový obdélník, který má nejmenší plochu. Tento obdélník je otočen zpět o daný úhel σ. Stejný způsob rotace je využit i u ostatních metod generalizace.

Principal Component Analysis

Dalším algoritmem pro hledání hlavních směrů oobjektů je *Metoda hlavních komponent* (PCA). K tomu se využívá singulární rozklad matice:

$$C = U\Sigma V^T, \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}^T, \tag{6}$$

kde C je kovarianční matice:

$$C = \begin{bmatrix} C(A,\ A) & C(A,\ B) \\ C(B,\ A) & C(B,\ B) \end{bmatrix}, C(A,\ B) = \frac{1}{n-1}\sum_{i=1}^{n}(A_i - \mu_A)(B_i - \mu_B), \tag{7}$$

matice U a V jsou ortogonální matice, tzn. sloupcové vlastní vektory jsou na sebe kolmé a jednotkové velikosti:

$$U = V = \begin{bmatrix} cos\ \sigma & -sin\ \sigma \\ sin\ \sigma & cos\ \sigma \end{bmatrix}, \tag{8}$$

matice Σ je diagonální matice se čtverci velikostí vlastních vektorů na hlavní diagonále:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} = \begin{bmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \end{bmatrix}. \tag{9}$$

Longest Edge

Principem tohoto algoritmu na hledání hlavního směru budovy je totožnost hlavního směru se směrem nejdelší hrany, kdy druhý hlavní směr je na něj kolmý. Následně je budova opět otočena o úhel -σ tak, aby hlavní směr byl rovnoběžný s osou. Kolem takto natočené budovy je vytvořen *Min-max box* a znovu otočen zpět o úhel σ.

Weighted Bisector

Metoda Weighted Bisector spočívá v nalezení dvou nejdelších uhlopříček polygonu budovy, je tedy potřeba, aby polygon měl alespoň 4 vrcholy. Tyto dvě uhlopříčky musí být spojnicemi dvou vrcholů polygonu a nesmí být totožné s hranou polygonu. Zároveň nesmí protínat žádnou z hran polygonu, což lze ověřit determinantovým testem. Nechť body $A = [x_A, y_A]$ a $B = [x_B, y_B]$ tvoří uhlopříčku a body $C = [x_C, y_C]$ a $D = [x_D, y_D]$ tvoří hranu budovy. Následně spočítáme determinanty:

$$t_A = \begin{vmatrix} x_D - x_C & y_D - y_C \\ x_A - x_C & y_A - y_C \end{vmatrix} \quad t_B = \begin{vmatrix} x_D - x_C & y_D - y_C \\ x_B - x_C & y_B - y_C \end{vmatrix}$$
$$t_C = \begin{vmatrix} x_B - x_A & y_B - y_A \\ x_C - x_A & y_C - y_A \end{vmatrix} \quad t_D = \begin{vmatrix} x_B - x_A & y_B - y_A \\ x_D - x_A & y_D - y_A \end{vmatrix} \tag{10}$$

Pokud se rovnají znaménka t_A a t_B nebo t_C a t_D, uhlopříčka neprotíná hranu budovy a lze jí použít pro výpočet hlavního směru σ. Ten se spočítá váženým průměrem směrnic dvou nejdelších uhlopříček, kde vahami jsou jejich délky d_1 a d_2.

$$\sigma = \frac{d_1\sigma_1 + d_2\sigma_2}{d_1 + d_2} \tag{11}$$

Wall Average

Předpokladem tohoto algoritmu jsou pravoúhlé strany objektu, kdy se jedná o vážený průměr úhlu stran, který je poté zmodulován $\frac{\pi}{2}$. Nejprve jsou vypočteny směrnice všech hran jako rozdíl směrnic σ' a σ_i:

$$\Delta\sigma_i = |\sigma_i - \sigma'| \tag{12}$$

Poté se vypočítá zaokrouhlený násobek $\frac{\pi}{2}$:

$$k_i = \frac{2\Delta\sigma_i}{\pi} \tag{13}$$

Následně je vypočten "orientovaný" zbytek po dělení:

$$r_i = \Delta\sigma_i - k_i\frac{\pi}{2} \tag{14}$$

Hlavní směr budovy se nakonec spočítá jako vážený průměr zbytků po dělení, kde vahou jsou délky stran:

$$\sigma = \sigma' + \sum_{i=1}^{n} \frac{r_i s_i}{s_i} \tag{15}$$

4 Popisy algoritmů formálním jazykem

Pseudokód metody Jarvis Scan

Algorithm 1 *Jarvis Scan*

inicializuj konvexní obálku ch
najdi pivota q s minimální y-souřadnicí
najdi bod p_x s minimální x-souřadnicí
inicializuj $p_j = q$, $p_{j+1} = [p_x, p_j]$
přidej bod p_j do konvexní obálky ch

dokud $p_{j+1} \neq q$:
    inicializuj maximální úhel ϕ_{max}
    pro každý vrchol polygonu:
        pokud $p_j \neq polygon[i]$:
            spočítej úhel ϕ
            aktualizuj maximální úhel ϕ_{max}

    přidej vrchol s maximálním úhlem do konvexní obálky ch
    pokud $p_j = q$:
        skonči cyklus

Pseudokód metody Graham Scan

Algorithm 2 *Graham Scan*

inicializuj konvexní obálku ch
najdi pivota q s minimální y-souřadnicí
přidej pivota do konvexní obálky ch
inicializuj pomocný bod v pro výpočet úhlů
inicializuj slovník, kam se budou ukládat vrcholy s úhly

pro každý vrchol polygonu p_i:
    pokud $p_i = q$:
        pokračuj
    spočti úhel $\omega(v, q, p_i)$

seřaď vrcholy polygonu podle úhlu ω vzestupně
pokud existují 2 stejné úhly ω, ponech ten odpovídající vrchol, který je dál od q
přidej vrchol p_j odpovídající nejmenšímu úhlu ω do konvexní obálky $(j = 0)$

pro zbylé vrcholy seřazené podle úhlu ω $(j > 1)$:
    dokud počet vrcholů v $ch > 1$ a p_j leží napravo od hrany tvořené posledními 2 vrcholy v ch:
        odstraň poslední vrchol z ch
    přidej p_j do ch

Pseudokód metody Minimum Area Enclosing/Bounding Rectangle

Algorithm 3 *Minimum Area Enclosing/Bounding Rectangle*

inicializuj minimální směrnici $\sigma_{min} = 0$
vytvoř konvexní obálku ch
inicializuj MMB_{min} a spočítej jeho plochu $area_{min}$

pro každou hranu konvexní obálky ch:
    spočítej rozdíly souřadnic dx, dy
    spočítej směrnici σ
    otoč konvexní obálku ch o $-\sigma$
    zkonstruuj MMB a spočítej $area$ rotované konvexní obálky ch
    pokud $area < area_{min}$:
        aktualizuj $area_{min}$, MMB_{min} a σ_{min}

    přeškáluj plochu MMB_{min} na stejnou plochu jako vstupní budova
    proveď zpětnou rotaci MMB_{min} o σ_{min}

Pseudokód metody Principal Component Analysis

Algorithm 4 *PCA*

inicializuj seznamy souřadnic x, y

pro všechny body polygonu *building*:
    přidej souřadnice bodu do seznamu x, y

vytvoř matici A
spočítej kovarianční matici C z matice A
proveď singulární rozklad matice C
spočítej směrnici σ vlastních vektorů matice C
otoč *building* o $-\sigma$
spočítej MMB rotované *building*
přeškáluj plochu MMB na stejnou plochu jako vstupní budova
proveď zpětnou rotaci MMB o σ

Pseudokód metody Longest Edge

Algorithm 5 *Longest Edge*

inicializuj nejdelší hranu $longest\_edge = 0$

pro všechny body polygonu *building*:
    spočítej délku hrany $edge\_length$

    pokud $edge\_length > longest\_edge$:
        aktualizuj $longest\_edge$
        inicializuj vrcholy a, b nejdelší hrany
    spočítej směrnici σ nejdelší hrany
    otoč polygon *building* o $-\sigma$

zkonstruuj MMB rotované *building*
proveď zpětnou rotaci MMB o σ
přeškáluj plochu MMB na stejnou plochu jako vstupní budova

Pseudokód metody Weighted Bisector

Algorithm 6 *Weighted Bisector*

pokud má vstupní polygon méně než 4 vrcholy:
    vrať prázdný polygon

najdi všechny úsečky $u_{i,j}$ spojující 2 vrcholy polygonu (kromě hran) a spočti jejich délku d
seřaď úsečky $u_{i,j}$ podle délky sestupně

pro všechny $u_{i,j}$ (seřazené):
    pokud $u_{i,j}$ neprotíná žádnou hranu polygonu (determinantový test), nalezení validní diagonály:
        spočítej směrnici σ_k úsečky $u_{i,j}$

        pokud byly nalezeny 2 validní diagonály:
            spočítej $\sigma = \frac{d_1\sigma_1 + d_2\sigma_2}{d_1 + d_2}$
            otoč polygon o $-\sigma$
            spočítej MMB rotovaného polygonu
            přeškáluj plochu MMB na stejnou plochu jako vstupní polygon
            proveď zpětnou rotaci MMB o σ
            ukonči cyklus

Pseudokód metody Wall Average

Algorithm 7 *Wall Average*

spočítej rozdíly souřadnic dx_0, dy_0 mezi dvěma prvními po sobě jdoucími body
inicializuj celkovou délku hran d_{sum} a rd ("orientované" zbytky po dělení vynásobené délkou hran)
spočítej směrnici σ_0 mezi dvěma prvními po sobě jdoucími body

pro každou hranu:
    spočítej rozdíly souřadnic dx, dy mezi dvěma po sobě jdoucími body
    spočítej směrnici σ_i mezi dvěma po sobě jdoucími body
    spočítej rozdíl směrnic $\Delta\sigma$
    spočítej k (zaokrouhlený násobek $\frac{\pi}{2}$)
    spočítej "orientovaný" zbytek po dělení r
    spočítej vzdálenost d mezi dvěma po sobě jdoucími body
    aktualizuj rd
    aktualizuj celkovou délku hran d_{sum}

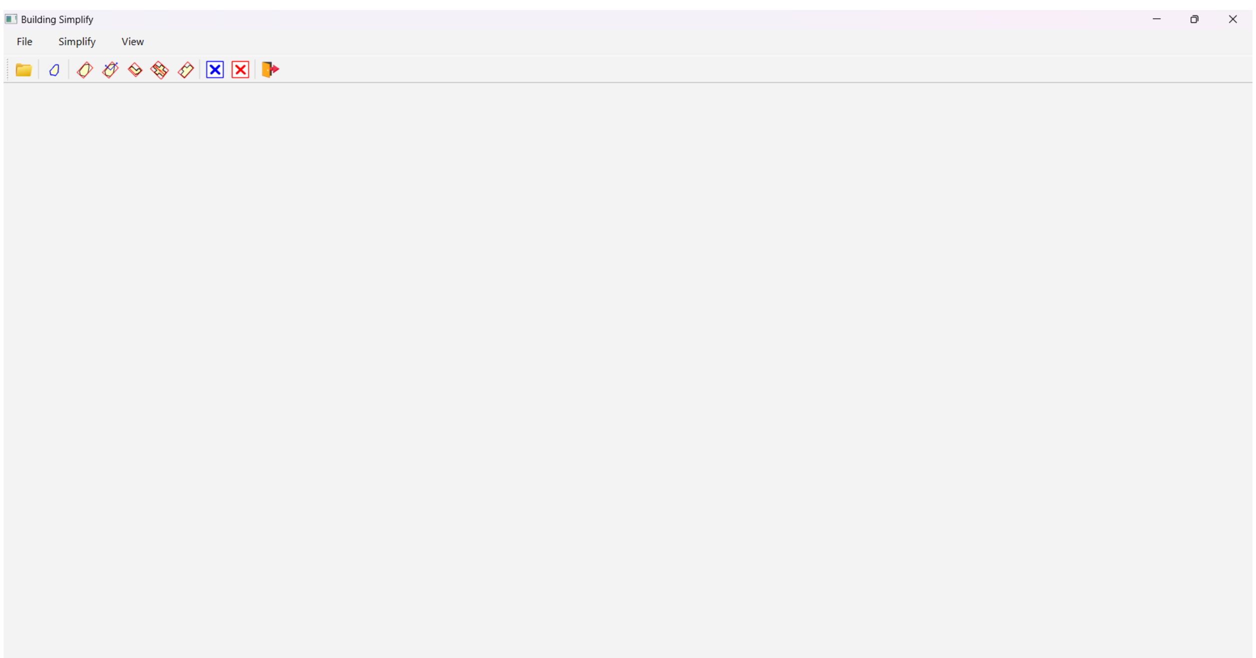
spočítej průměrný úhel σ
otoč *building* o $-\sigma$
spočítej MMB rotované *building*
proveď zpětnou rotaci MMB o σ
přeškáluj plochu MMB na stejnou plochu jako vstupní budova

5 Aplikace

Po spuštění skriptu *MainForm.py* se otevře okno aplikace. V záložce *File* jsou umístěny funkce *Open* a *Exit*. V další záložce *Simplify* jsou algoritmické funkce *MBR*, *PCA*, *Longest Edge*, *Weighted Bisector*, *Wall Average* a funkce na přepínání metody tvorby konvexní obálky *Jarvis/Graham Scan*. Poslední záložka *View* obsahuje funkce *Clear results* a *Clear all*. Všechny tyto funkce mají ikonku nacházející se na nástrojové liště (*Obrázek 4*).

Po kliknutí na fukci *Open* se otevře nabídka pro vybrání požadovaného souboru ve formátu *shapefile*, který obsahuje pouze topologicky správné polygony. Jako data pro generalizaci byly využity vrstvy budov tří městských částí Prahy (*Geoportál Praha 2024*). Přesněji se jedná o části Háje, Staré Město a Točná, které jsou přiloženy společně s kódy. Tato data byla ještě upravena v SW ArcGIS Pro, aby testovací data obsahovaly méně budov, jelikož v aplikaci nelze přibližovat. Díky tomu je lépe viditelný rozdíl mezi vstupní a generalizovanou budovou. V aplikaci lze též kreslit svůj vlastní polygon, jestliže není otevřen shapefile soubor.

Kliknutím na funkci *Clear results* se smaže generalizovaná budova (Min-max box). Pro nahrání nebo nakreslení jiného polygonu je zapotřebí spustit funkci *Clear all*. Aplikaci lze ukončit funkcí *Exit* nebo tlačítkem "křížek" v pravém horním rohu.



Obrázek 4: Ukázka aplikace

6 Dokumentace

Program řešící generalizaci budov pomocí různých metod byl vytvořen v SW Visual Studio Code v jazyce Python. Program se skládá ze tří jednotlivých souborů, které zároveň odpovídají názvům použitých tříd. Celé grafické prostředí výsledné aplikace bylo vytvořeno v SW Qt Designer.

Třída MainForm

Pomocí třídy *MainForm* se spouští celé uživatelské prostředí aplikace. Třida obsahuje načítání ikonek jednotlivých funkcí a zprostředkovává propojení s ostatními třídami. Třída dále obsahuje tyto funkce:

- **simplifyBuildingCore** - funkce zavolá metody *getBuilding* a třídu *Algorithms*. Dále je nad všemi polygony z metody *getBuilding* zavolána jedna z metod v závislosti na zvoleném parametru. Poté je zavolána metoda *setSimplifBuilding* a celá aktivní plocha aplikace se aktualizuje. Nakonec funkce zavolá metodu *evaluate* a v dialogovém okně zobrazí přesnost generalizace.
- **simplifyBuildingMBR** - funkce zavolá metodu *simplifyBuildingCore* s parametrem "MBR", která zavolá metodu *createMBR* na vstupní polygony.
- **simplifyBuildingBRPCA** - funkce zavolá metodu *simplifyBuildingCore* s parametrem "BRPCA", která zavolá metodu *createBRPCA* na vstupní polygony.
- **simplifyBuildingLE** - funkce zavolá metodu *simplifyBuildingCore* s parametrem "LE", která zavolá metodu *longestEdge* na vstupní polygony.
- **simplifyBuildingWB** - funkce zavolá metodu *simplifyBuildingCore* s parametrem "WB", která zavolá metodu *weightedBisector* na vstupní polygony.
- **simplifyBuildingWA** - funkce zavolá metodu *simplifyBuildingCore* s parametrem "WA", která zavolá metodu *wallAverage* na vstupní polygony.
- **switchClick** - funkce zavolá metodu *switchMethodCH*.
- **openClick** - funkce zavolá metodu *openFile*.
- **exitClick** - funkce zavolá metodu *exit*.
- **clearAllClick** - funkce zavolá metodu *clearAll*.
- **clearClick** - funkce zavolá metodu *clearRes*.

Třída Draw

Na začátku třídy jsou inicializovány pomocné proměnné. Třída dále obsahuje následující funkce:

- **openFile** - funkce zobrazí systémové okno s možností nahrání souboru ve formátu *shapefile*. Vybraný shapefile přečte a pro jeho zobrazení v okně aplikace zavolá funkci **geomShapefile**.
- **geomShapefile** - funkce vykreslí shapefile, při čemž zpracovává geometrii vstupních dat a upraví souřadnice dat podle velikosti plochy otevřeného okna aplikace.
- **exit** - funkce ukončí aplikační okno.
- **clearAll** - funkce vymaže všechna data (vložená, nakreslená a algoritmem vygenerovaná).
- **clearRes** - funkce vymaže výsledný generalizovaný polygon.
- **mousePressEvent** - funkce zjišťuje souřadnice, na které bylo kliknuto. Tyto souřadnice se rovnají vrcholu kresleného polygonu.
- **paintEvent** - funkce vykreslí nahrané polygony ze shapefilu, nebo vykreslí uživatelem nakresleným polygon. Dále funkce vykreslí výsledný generalizovaný polygon.
- **switchMethodCH** - funkce přepíná metody pro vytvoření konvexní obálky (Jarvis/Graham Scan).
- **getBuilding** - funkce vrací polygony budov jako seznam.
- **setSimplifBuilding** - funkce změní hodnotu proměnné s generalizovanými polygony.
- **getMethodCH** - funkce vrací danou metodu pro vytvoření konvexní obálky (1 = Jarvis Scan, 2 = Graham Scan).

Třída Algorithms

Tato třída obsahuje následující funkce:

- **calculateDistance** - funkce spočítá euklidovskou vzdálenost předaných bodů.
- **get_point_location** - funkce analyzuje vzájemnou polohu bodu a přímky pomocí determinantu pro Half-plane test.
- **get2VectorsAngle** - funkce spočítá velikost úhlu mezi vektorem $\vec{u}$ a vektorem $\vec{v}$ podle vzorce:

$$\cos\omega = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|} \tag{16}$$

- **createCHJ** - funkce zkonstruuje konvexní obálku metodou Jarvis Scan. Vyhodnocení bylo provedeno na základě rovnice (3) v kapitole 3.

- **createCHG** funkce zkonstruuje konvexní obálku metodou Graham Scan. U jednotlivých vrcholů je vyhodnocováno kritérium levotočivosti popsané rovnicí (4) v kapitole 3 za pomoci funkce **get_point_location**.
- **rotate** - funkce otočí polygon o daný úhel $-\sigma/\sigma$. K rotaci byl využit vzorec (5) uvedený v kapitole 3.
- **createMMB** - funkce spočítá min-max box pro daný polygon.
- **getArea** - funkce spočítá plochu pro daný polygonu.
- **resizeRectangle** - funkce přeškáluje generalizovanou budovu podle plochy vstupní budovy. Použité vzorce jsou vypsány v kapitole 3, přesněji byly využity rovnice (1) a (2).
- **createMBR** - funkce kolem konvexní obálky budovy vytvoří obdélník s minimální plochou.
- **createBRPCA** - funkce detekuje natočení budovy pomocí metody PCA. Při implementaci byly využity rovnice (6), (7), (8) a (9) z kapitoly 3.
- **wallAverage** - funkce detekuje natočení budovy pomocí metody Wall Average. Využity přitom byly rovnice (12), (13), (14) a (15) uvedené v kapitole 3.
- **longestEdge** - funkce detekuje natočení budovy pomocí metody Longest Edge.
- **weightedBisector** - funkce detekuje natočení budovy pomocí metody Weighted Bisector. Pro provedení metody byly použity vzorce (10) a (11) z kapitoly 3.
- **normalizeAngle** - funkce normalizuje úhel do intervalu $(-\pi, \pi\rangle$ (v radiánech).
- **evaluate** - funkce slouží k hodnocení efektivity detekce hlavních směrů. Využity k tomu byly rovnice pro výpočet střední hodnoty čtverců úhlových odchylek jednotlivých segmentů:

$$\Delta\sigma = \frac{\pi}{2n}\sqrt{\sum_{i=1}^{n}(r_i - r)^2}, \tag{17}$$

kde

$$\begin{aligned} r_i &= \Delta\sigma_i - k_i\frac{\pi}{2} \\ k_i &= \frac{2\sigma_i}{\pi}. \end{aligned} \tag{18}$$

7 Výsledky

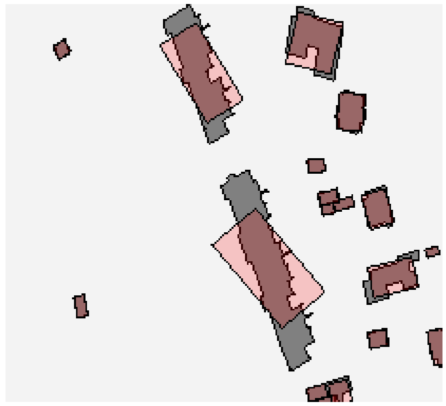
Efektivita detekce hlavních směrů budov, tj. hodnocení generalizačních metod bylo vypočteno pomocí rovnice (17). V tabulce je uvedeno procento budov, pro které platí $\Delta\sigma < 10°$, tedy hlavní směr budovy je téměř rovnoběžný s hranami budovy nebo je na ně kolmý.

Městské části	**MBR**	**PCA**	**Longest Edge**	**Weighted Bisector**	**Wall Average**
Staré Město	96,26 %	70,59 %	94,12 %	51,34 %	91,44 %
Háje	99,02 %	81,43 %	99,02 %	69,06 %	99,02 %
Točná	99,06 %	63,12 %	99,06 %	46,56 %	97,81 %
průměr	**98,11 %**	**71,71 %**	**97,40 %**	**55,65 %**	**96,09 %**

Nejlepšího hodnocení podle kritérií dosáhl Minimum Area Enclosing/Bounding Rectangle s 98,11 %. Tato metoda poskytuje celkově dobré výsledky. Jediným problémem je zástavba Starého Města, kde mají budovy nepravidelné tvary, při generalizaci dochází nevhodnému k překrývání budov, nicméně takovýto typ zástavby je obecně obtížné generalizovat na LOD0.

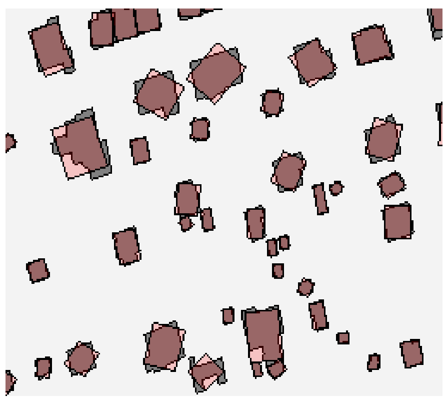
Obrázek 5: Generalizované budovy ve Starém Městě metodou MBR

Metody Longest Edge a Wall Average také dosáhly velmi dobrých výsledků. Problematickým místem se pro metodu Wall Average ukázaly být budovy s mnoha kratšími hranami, viz Obr. 6.



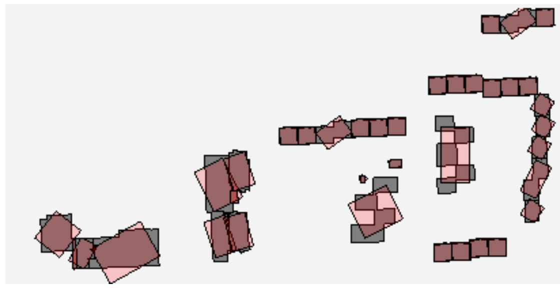
Obrázek 6: Ukázka nevhodně generalizovaných budov metodou Wall Average (Točná)

Nepříliš dobrých výsledků dosáhla metoda Principal Component Analysis. Při její aplikaci dochází k nevhodnému natočení budov, které mají výstupky.



Obrázek 7: Ukázka nevhodně generalizovaných budov metodou Principal Component Analysis (Točná)

Nejhorších výsledků dosáhla metoda Weighted Bisector. Dochází zde k natáčení budov tak, že hrany generalizované budovy nejsou paralelní s hranami budovy původní.



Obrázek 8: Ukázka nevhodně generalizovaných budov metodou Weighted Bisector (Háje)

8 Závěr

V rámci řešení problému generalizace budov LOD0 byly implementovány algoritmy a metody *Minimum Area Enclosing/Bounding Rectangle*, *Principal Component Analysis*, *Longest Edge*, *Weighted Bisector* a *Wall Average*. Jednotlivé analyzované městské části jsou vizualizovány v aplikaci, která zároveň zobrazuje výslednou generalizaci pro každou metodu.

Jako možné vylepšení skriptu by byla možnost přibližování a pohybování se v okně aplikace, aby bylo možné si prohlédnout detailní rozdíl jednotlivých metod generalizace od vstupních budov.

Zdroje

BAYER, T. (2025): Konvexní obálka množiny bodů. Přednáška pro předmět Algoritmy počítačové kartografie, Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK [cit. 24.3.2025].

GEEKSFORGEEKS (2025): Convex Hull using Graham Scan. https://www.geeksforgeeks.org/convex-hull-using-graham-scan/ [cit. 25.3.2025].

GEEKSFORGEEKS (2024): Convex Hull using Jarvis' Algorithm or Wrapping. https://www.geeksforgeeks.org/convex-hull-using-jarvis-algorithm-or-wrapping/ [cit. 25.3.2025].