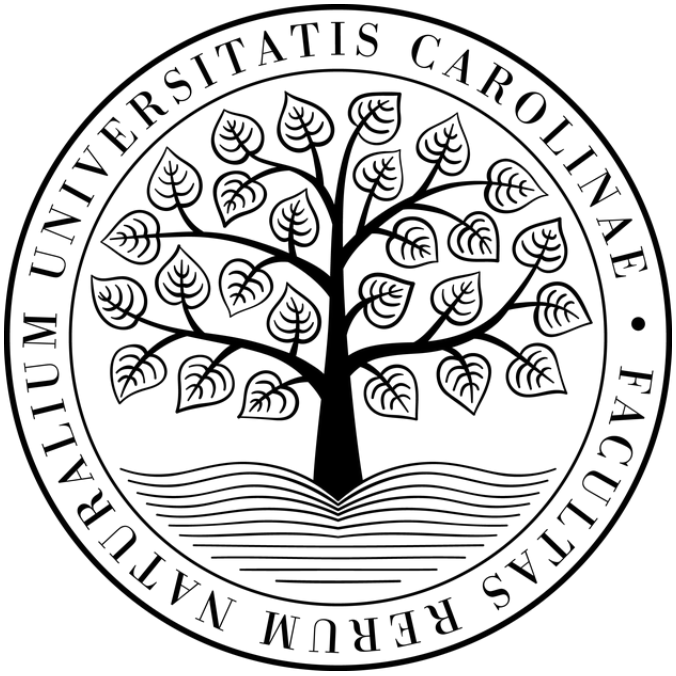
Přírodovědecká fakulta
Univerzita Karlova



Algoritmy počítačové kartografie

Úkol č. 2: Generalizace budov

Anna Brázdová, Petra Pajmová, Jakub Zapletal

N-GKDPZ

Praha 2024

Vstup: množina budov $B = \{B_i\}_{i=0}^{n}$,*budova* $B_i = \{P_{i,j}\}_{j=1}^{m}$.

Výstup: $G(B_i)$.

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

Pro každou budovu určtete 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 detailu LOD0 nahraďte obdélníkem orientovaným v obou hlavních směrech, se středem v těžišti budovy, jeho plocha bude stejná jako plocha budovy. Výsledky generalizace vhodně vizualizujte.

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

Hodnocení

V rámci úlohy byly řešeny následující části:

Krok	**hodnocení**
Generalizace budov metodami Minimum Area Enclosing Rectangle a PCA.	15 b.
Generalizace budov metodou Longest Edge.	*+ 5 b.*
Generalizace budov metodou Wall Average.	*+ 8 b.*
Generalizace budov metodou Weighted Bisector.	*+ 10 b.*
Celkem	**43 b.**

Popis a rozbor problému

Úvod

Automatická generalizace budov v mapách je proces, který se snaží přizpůsobit detaily a rozsah budov na mapě tak, aby byly lépe čitelné a vhodné pro dané použití. Tento proces je důležitý pro vytváření map různých měřítek a účelů, jako jsou turistické mapy, navigační mapy nebo urbanistické plány. Generalizace budov v mapách je tedy komplexní problém, který zahrnuje technické aspekty algoritmů, ale také estetické a uživatelské hledisko. Úspěšná generalizace vyžaduje vyvážený přístup k těmto různým faktorům (Punt, Watkins 2010).

Automatická generalizace map není na poli kartografie nové téma. S tímto tématem se můžeme setkat už od 60. let minulého století. K zásadnějšímu vývoji ale dochází až v posledních dvou dekádách zejména díky rychlému rozvoji geoinformačních systémů a tím tedy přesunu od běžné kartografie k té digitální. Přestože geoinformační systémy a jejich softwary přinášejí celou řadu nástrojů a funkcí pro zpracování a analýzu prostorových dat, nástroje pro automatickou generalizaci jejich součástí nebývají (LI a kol. 2004).

Hlavním cílem při generalizaci budov je zjednodušit objekty v mapě takovým způsobem, aby zůstal zachován charakter a vztahy mezi sousedními prvky. Celý proces automatické generalizace budov bývá často rozdělen do tří kroků. V první části je popsána a provedena analýza prostorového kontextu geografických prvků. Následně je s ohledem na zjištěné parametry provedeno algoritmické zpracování. Třetím krokem je pak vyhodnocení výsledků generalizace (LI a kol. 2004). Při procesu automatické generalizace budov je nutné znát jejich orientaci. Při práci s body nebo úsečkami je to poměrně jednoduchý úkol. Ovšem při práci s mnohoúhelníky se úloha komplikuje. Orientace budov odpovídá buď orientaci jejích stěn (slouží pro porovnání orientace dvou rovnoběžných stěn) nebo obecné orientaci (slouží k prodloužení budovy) (Duchene 2003).

Důležitým kritériem kartografické generalizace je to, aby budova měla před i po generalizaci stejnou orientaci vzhledem k ostatním prvkům na mapě. Je tedy nutné definovat hlavní směry budovy, které právě tuto orientaci popisují. Algoritmy, pomocí kterých lze určit hlavní směry budov a budou testovány v této práci jsou Minimum Area Enclosing Rectangle, Principal Component Analysis, Longest Edge, Wall Average a Weighted Bisector.

Konvexní obálka

Konvexní obálka je jedna z nejpoužívanějších geometrických struktur. Konvexní obálkou nazýváme takovou množinu bodů, která splňuje následující vlastnosti: ohraničení určité množiny bodů je minimální, žádný z bodů množiny nesmí být mimo obálku, pokud spojíme libovolné dva body množiny úsečkou, potom se musí celá úsečka nacházet uvnitř naší obálky. Konvexní obálky využívá řada algoritmů. Běžně se používá v kartografii při detekci tvaru a orientace budov. Dále se využívá například k plánování pohybu robotů

(předcházení kolizí), analýze shluků nebo pro statistické analýzy.

Pro sestrojení konvexní obálky se využívá řada algoritmů. Jednotlivé algoritmy se liší paměťovými a časovými nároky, ale především svojí robustností. Z tohoto důvodu se liší také jejich použití v různých případech. Mezi běžně používané algoritmy patří například: *Jarvis Scan*, *Graham Scan*, *Quick Hull*, *Inkrementální konstrukce* nebo *Divide and Conquer*.

Jarvis Scan

Jarvis Scan, známý také jako *Gift Wrapping Algorithm*, je jedním z nejpoužívanějších algoritmů používaných pro konstrukci konvexní obálky. Je založen na principu opakovaného hledání maximálního úhlu ω_{max}. Vlastní algoritmus je poměrně jednoduchý a snadno implementovatelný. Metoda ovšem není příliš robustní, a proto se doporučuje aplikovat ji na menší datové sady, které mají menší počet vstupních bodů n.

Algoritmus nejprve vybere ze vstupní množiny počáteční bod P. Takový bod označujeme jako *pivot* a je vybrán podle minimální hodnoty souřadnice x. O tomto bodu lze jednoznačně tvrdit, že bude ležet uvnitř konvexní obálky. Následně je bod přidán do seznamu vrcholů, které tvoří konvexní obálku. Z bodu P je hledán další bod q, který leží na přímce procházející bodem P tak, aby maximalizovat úhel ω.

Minimum Area Enclosing Rectangle

Účelem algoritmu Minimum Area Enclosing Rectangle je určit hlavní směr budovy a zjistit tak její orientaci. Pro fungování algoritmu je nejprve zapotřebí zkonstruovat konvexní obálku pro množinu bodů S. Výsledný minimální ohraničující obdélník R má vždy alespoň jednu hranou rovnoběžnou s některou z hran konvexní obálky (Eberley 2020).

$$S_r = R(-\sigma)S \Rightarrow \begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \cos(-\sigma) & -\sin(-\sigma) \\ \sin(-\sigma) & \cos(-\sigma) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Po sestrojení konvexní obálky dochází k jejímu opakovanému otočení pomocí matice rotace o úhel - σ. Otočení proběhne tolikrát, kolik hran má sestrojená konvexní obálka. V této otočené poloze je poté nalezen minimální $MinimumBoundingBoxMMB$ a je vypočítána jeho plocha. Pokud bude plocha nového MMB menší než doposud nalezená, MMB a jeho úhel otočení se zapamatují (Bayer 2024).

Principal Component Analysis

Pro nalezení hlavních směrů se v praxi často využívá metoda Principal Componenet Analysis (PCA). Základ pro PCA tvoří kovarianční matice, která je pomocí metody Singular Value Decomposition (SVD) rozložena na součin tří samostatných matic $U\Sigma V$.

$$C = U\Sigma V^T \tag{1}$$

Matice U a V jsou ortogonální a normované a jsou složeny z vlastních vektorů, které poté vypovídají o orientaci budovy. Matice S je diagonální a obsahuje na diagonále tzv. singulární čísla, jejichž odmocniny odpovídají vlastním číslům (velikostem vlastních vektorů).

Longest Edge

Základní myšlenkou tohoto velmi jednoduchého algoritmu je, že hlavní směr polygonu je totožný s jeho nejdelší hranou. Druhý hlavní směr je pak kolmý k nejdelší hraně. Obdobně, jako u předchozích algoritmů, je i zde budova otočena o úhel - σ, který odpovídá směrnici nejdelší hrany budovy. V této poloze je sestrojen $MinimumBoundingBox$ který je otočen o úhel σ. V posledním kroku je MMB upraven tak, aby jeho plocha odpovídala ploše budovy (Bayer 2024).

Wall Average

Dalším algoritmem pro hledání orientace budov je *Wall Average*. Pro orientaci budov je uvažována hodnota modulo $\frac{\pi}{2}$ v rozmezí 0 a $\frac{\pi}{2}$, kdy je vypočítán průměr těchto směrů. Nejprve je určena směrnice σ' a následně směrnice σ_i pro všechny strany, které jsou redukovány právě o hodnotu σ' (Bayer 2024).

$$\Delta\sigma_i = \sigma_i - \sigma'$$

Následně lze vypočítat zaokrouhlený podíl k_i jako:

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

V případě, že je hodnota zbytku $r_i < \frac{\pi}{4}$, je daná hrana odchýlena od vodorovného směru $(0 \pm k\pi)$. Pokud $r_i > \frac{\pi}{4}$, je hrana odchýlena od svislého směru $(\frac{\pi}{2} \pm k\pi)$. Výsledný směr natočení budovy je určen pomocí výpočtu průmerného úhlu (Bayer 2024).

Popisy algoritmů formálním jazykem

Pseudokód pro Jarvis Scan

Algorithm 1 *Jarvis Scan*

inicializuj konvexní obálku
najdi počáteční bod q s minimální hodnotou souřadnice y
najdi počáteční bod s s minimální hodnotou souřadnice x
inicializuj poslední dva body
přidej počáteční bod do konvexní obálky
dokud $p_{j+1} \neq p_{min}$:
    pro každý bod z množiny:
        vypočti úhel ω
        $p_{j+1} = max\omega$
        přidej p_{j+1} do konvexní obálky
        aktualizuj poslední dva body konvexní obálky

Pseudokód pro Minimum Area Enclosing Rectangle

Algorithm 2 *Minimum Area Enclosing Rectangle*

vytvoř konvexní obálku
inicializuj MMB a vypočítej jeho plochu A
pro každou hranu konvexní obálky
    vypočti úhel σ
    otoč konvexní obálku o úhel $-\sigma$
    spočti MMB_i a vypočítej jeho plochu A_i
    pokud $A_i > A$:
        $A = A_i$
        $MMB = MMB_i$
        $\sigma = \sigma_i$
otoč MMB_i o σ_i a přeškáluj jeho plochu, tak, že $A_i = A$

Principal Component Analysis

Algorithm 3 *Generalizace pomocí PCA*

inicializuj prázdné listy x a y pro planární souřadnice
pro p v *pol*
    přidej $p.x()$ do x
    přidej $p.y()$ do y
vytvoř matici P z x a y ($P = [[x], [y]]$)
vyočítej kovarianční matici C z P ($C = \mathrm{cov}(P)$)
proveď singulární rozklad C na U, S, a V ($[U, S, V] = \mathrm{SVD}(C)$)
vypočítej vlastní vektory
otoč *pol* o $-\sigma$ (jako *pol_unrot*)
vytvoř minimální ohraničující obdelník (MMB) z *pol_unrot* ($MMB = \mathrm{createMMB}(pol\_unrot)$)
otoč MMB o σ (označeno jako er)
změň velikost er tak, aby obsahoval *pol* ($er\_r = \mathrm{resizeRectangle}(er, pol)$)
vrať er_r

Longest Edge

Algorithm 4 *Longest Edge*

1: inicializuj počet vrcholů n a nejdelší hranu $l_{max} = 0$
2: **pro** každou hranu:
3: vypočti délku hrany l
4: **pokud** $l > l_{max}$:
5: $l_{max} = l$
6: vypočti směrnici l_{max} σ
7: otoč budovu o $-\sigma$
8: zkonstruuj MMB a vypočti jeho plochu A
9: otoč MMB_i o σ_i
10: vypočti plochu budovy A_b a přeškáluj plochu MMB tak, že $A = A_b$

Wall Average

Algorithm 5 *Wall Average*

1: vypočítej směrnici σ pro libovolnou hranu
2: inicializuj zbytek $r = 0$
3: **pro** každou hranu:
4: vypočti směrnici σ_i
5: vypočti $\Delta\sigma$
6: vypočti ki a ri
7: vypočti průměrný úhel σ:
8: otoč budovu o $-\sigma$
9: zkonstruuj MMB a vypočti jeho plochu A
10: otoč MMB o σ
11: vypočti plochu budovy A_b a přeškáluj plochu MMB tak, že $A = A_b$

Weighted Bisector

Algorithm 6 *Weighted Bisector*

1: Inicializovat prázdný slovník `dist` pro hledání nejdelších úhlopříček
2: Inicializovat seznam `points` s body polygonu ($points = [(x_i, y_i)|p_i \in pol]$)
3: **Najít dvě nejdelší uhlopříčky a získat jejich délky:**
4: **pro** $i = 0$ **do** $n - 1$
5: pro $j = i + 1$ **do** $n - 1$
6: Označit body i a j jako `point1` a `point2` (`point1 = points[i]`, `point2 = points[j]`)
7: Do `key` uložit souřadnice bodů (`key` = (`point1`, `point2`))
8: Do `dist` uložit `point1` a `point2` a vzdálenost bodů (`dist[key]`)
9: Seřadit slovník `dist` podle hodnot vzdálenosti sestupně (`dist_sort` = `sort(dist, descending)`)
10: Inicializovat prázdné seznamy `distances` a `angles` pro vzdálenosti a úhly
11: $i = 0$
12: **Vypočítat úhel mezi osou x a dvě nejdelšími uhlopříčkami:**
13: **pro** *key*, *value* **in** `dist_sort`
14: Označit body v klíči jako `point1` a `point2` (`point1, point2 = [QPointF(*point) for point in key]`)
15: Spočítat změnu v x a y (`delta_x = point2.x() - point1.x()`, `delta_y = point2.y() - point1.y()`)
16: Spočítat úhel vzhledem k ose x (`angle_x = atan2(delta_y, delta_x)` - $\pi/2$)
17: Přidat úhel `angle_x` do `angles` a délku`distance` do `distances`
18: Spočítat vážený úhel (`angle_weighted = (distances[0]` · `angles[0] + distances[1]` · `angles[1]) / (distances[0]+distances[1])`)
19: **Váha je délka úhlopříčky**
20: Otočit polygon o σ (`pol_r = rotate(pol,` σ`)`)
21: Vytvořit nejmenší ohraničující obdélník z `pol_r` (`mmb = createMMB(pol_r)`)
22: Otočit `mmb` o σ (`er = rotate(mmb,` σ`)`)
23: Změnit velikost `er` tak, aby obsahoval `pol` (`er_r = resizeRectangle(er, pol)`)
24: **Vrať** `er_r`

Vstupní a výstupní data

Vstupními daty pro analýzu a testování problematiky generalizace budov byla využita volně dostupná data Geoportálu Praha ve formátu shapefile. Data obsahují celkem tři oddělené datasety, z nichž každý reprezentuje budovy v různých typech zástavby (historické centrum měst, intravilán - sídliště, intravilán - izolovaná zástavba). Data jsou přiložena k textové zprávě ve složce buildings.

Výstupem je aplikace vytvořená v rozhraní QT, která umožňuje grafické znázornění analýzy. Aplikace umožňuje načtení dat (polygonů) a výběr algoritmu pro analýzu. V rámci této úlohy bylo také vypracováno hodnocení efektivnosti jedotlivých algoritmů.

Dokumentace

Třída Ui_MainWindow

Třída Ui_MainWindow se nachází ve skriptu MainFrom.py a slouží ke spuštění uživatelského rozhraní aplikace a zajišťuje propojení jednotlivých algoritmů definovaných v ostatních třídách. Celkem obsahuje jedenáct metod. Metody setupUi a restranslateUi byly vytvořeny automaticky na základě vytvořeného prostředí v Qt Creator a obsahují napojení jednotlivých položek na menu a tlačítek na připravené metody. Zbylými metodami jsou: openClick, resizeDisplay, mbrClick, pcaClick, longest_edgeClick, wall_avergeClick, weighted_BisectorClick, clearResultsClick, clearAllClick.

Třída Algorithms

Třída Algorithms je uložena v souboru algorithms.py a slouží k vlastnímu procesu generalizace budov. Třída obsahuje následující metody:

- analyzePointPolygonPosition - vrací polohu bodu vůči polygonu (uvnitř, vně)
- get2LineAngle — vypočtení úhlu mezi dvěma hranami
- distance_points - vypočtení eukleidovské vzdálenosti mezi dvěma body
- cHull - sestrojení konvexní obálky pomocí algoritmu Jarvis scan
- createMMB - vytvoření min-max boxu pro polygon
- rotate - otočení polygonu o daný úhel
- getArea - vypočtení plochy polygonu
- resizeRectangle - změní velikost generalizované budovy podle její původní rozlohy
- createMBR - vytvoření minimálního ohraničujícího obdélníku konvexní obálky

- createERPCA - vytvoření ohraničujícího obdélníku pomocí metody PCA
- longestEdge, wallAverage, weighted_bisector - algoritmy pro generalizaci budov
- calculate_accuracy - definice hodnotícího kritéria

Třída Draw

Třída Draw je uložena v souboru draw.py a slouží k zajištění grafického rozhraní aplikace. Třída obsahuje následující metody:

- loadData - načtení vstupních dat ve formátu shapefile
- resizeData - roztažení vstupních dat podle velikosti okna aplikace (pomocí parametrů width a height widgetu)
- mousePressEvent - vykreslení bodů tvořících polygon pomocí polohy kurzoru myši
- paintEvent - vizualizační nástroj pro znázornění dat
- getBuildings - vrací seznam vstupních polygonů
- setMBR - vrací seznam generalizovaných polygonů
- clearResults - odstraní výsledek analýzy
- clearData - odstraní vše z Canvasu

Výsledky

Dataset Vinohrad

Městská čtvrť Vinohrady je sestavena převážně z bloků budov. Budovy k sobě přímo přiléhají a mají většinou obdélníkový tvar. I proto zde byla analýza značně jednodušší než u ostatních datasetů. Tři z pěti testovaných algoritmů dokázaly správně určit hlavní směr budovy.

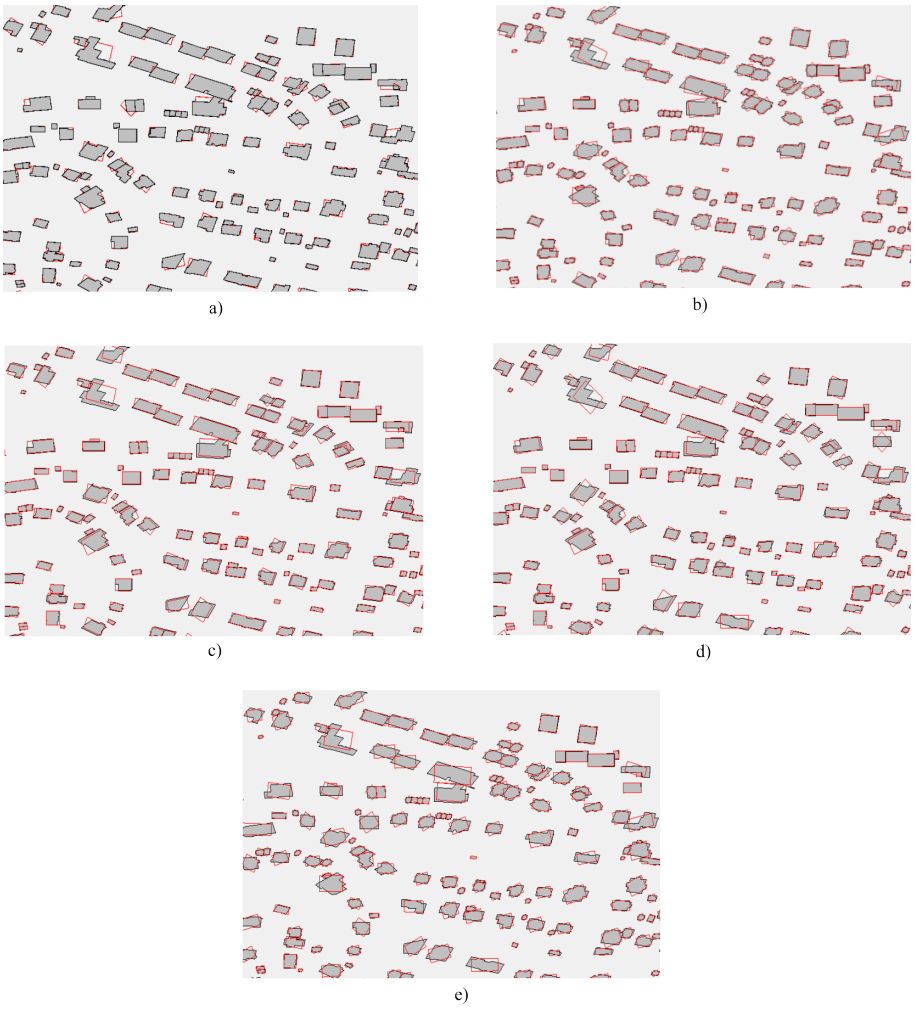
V tomto hledisku patří mezi neúspěšné metody PCA a Weighted Bisector, u kterých byly generalizované obrazy naprosté většiny budov otočeny o 45°. Tím, že k sobě budovy v tomto datasetu plynule navazují, se výsledek těchto metod jeví jako značně nedostačující. Metody MAER a Longest Edge podávají téměř totožné a velmi uspokojivé výsledky. Metoda Wall Average se odlišuje tím, že hůře odráží hlavní směr rohových budov.

Obrázek 1: Porovnání generalizačních algoritmů pro Vinohrady. Algoritmy: a) *MAER*, b) *PCA*, c) *Longest Edge*, d) *Wall Average* e) *Weighted Bisector*. (vlastní zpracování).

Dataset Braník

Městská čtvrť Braník je reprezentována jako jednotlivě oddělené domy, které na sebe ve většině případů nenavazují a nejsou nijak spojené. Na rozdíl od datasetu Vinohrad jsou ovšem domy členitější.

I z tohoto důvodu je zde aproximace všeobecně o stupeň horší než v případě Vinohrad. Efektivnost algoritmů ale kopíruje předchozí dataset. Metody PCA a Weighted Bisector podávají nevhodné výsledky a aproximace neodráží skutečnou orientaci budov. Metodám MAER, Longest Edge a Wall Average se dařilo o poznání lépe. Všechny tyto algoritmy dobře odráží hlavní směry budov. V případě metody Wall Average je ovšem znatelné zhoršení aproximace u budov s půdorysem ve tvaru písmene L.

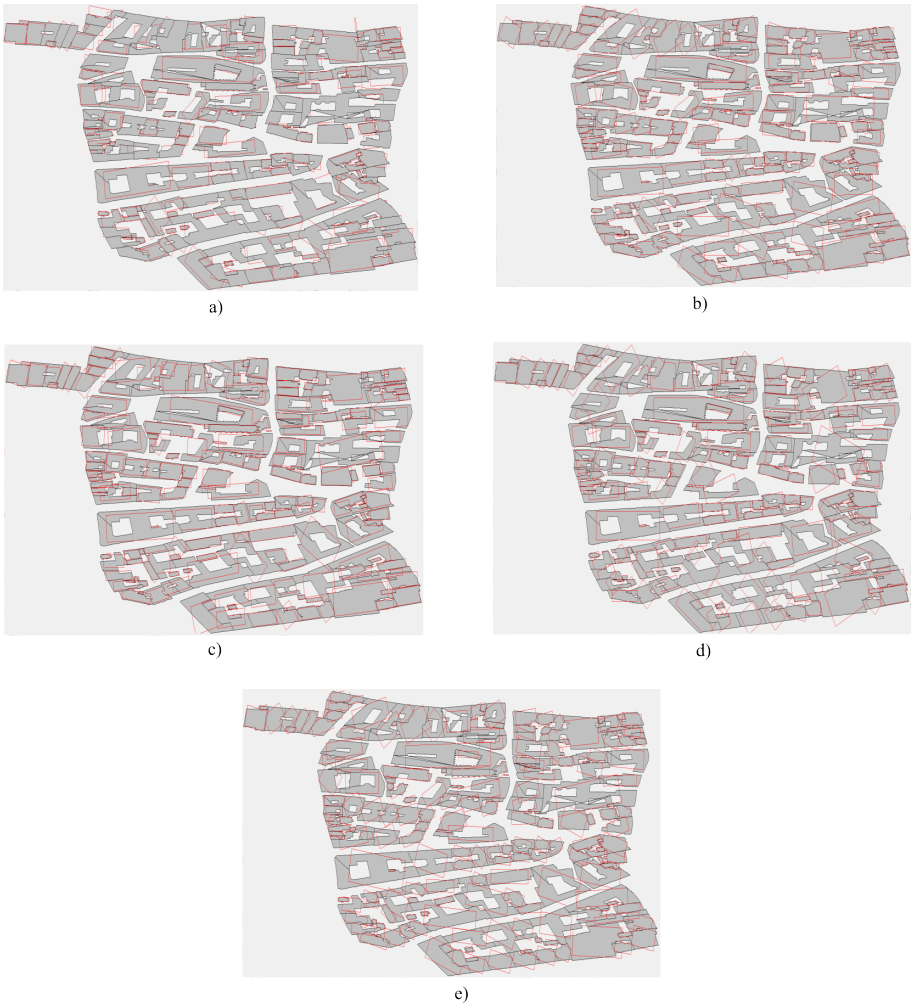


Obrázek 2: Porovnání generalizačních algoritmů pro Braník. Algoritmy: a) *MAER*, b) *PCA*, c) *Longest Edge*, d) *Wall Average* e) *Weighted Bisector*. (vlastní zpracování).

Staré Město

Městská čvrť Staré Město je jednoznačně datasetem s nejrozmanitějším půdorysem budov. Budovy v tomto datasetu obsahují téměř věšechny typy půdorysu. Pravidelných obdélníkových budov je zde pouze malé množství. Většina budov nemá pravé úhly a je nějakým způsobem vykrojená, například do tvaru písmene L, U, nebo O.

Také v tomto případě se objevuje stejný trend, jako u předchozích dvou typů zástavby. Algoritmus PCA a Weighted Bisector podávají viditelně slabší výsledky. Ikdyž výsledky zblých tří metod jsou podobné, lze pozorovat malé odlišnosti. Metodou, která aproximuje budovy zdánlivě nejlépe, je zde MAER. Algoritmus Longest Edge je problematický v případech, kdy budeva má půdorys tvaru písmene O, tedy uzavařená budova s dírou uvnitř. Oproti tomu algoritmus Wall Average hůře zvládá aproximovat protáhlé budovy. U takových budov metoda sestrojí aproximaci blížící se čtverci a nedostatečně tedy odráží její charakter.



Obrázek 3: Porovnání generalizačních algoritmů pro Staré Město. Algoritmy: a) *MAER*, b) *PCA*, c) *Longest Edge*, d) *Wall Average* e) *Weighted Bisector*. (vlastní zpracování).

Hodnocení efektivity detekce hlavních směrů

Hodnocení efektivity detekce hlavních směrů budov bylo provedeno dvojím způsobem. Nejprve pomocí vizuální interpretace grafických výsledků a následně pomocí hodnotícího kritéria. Bohužel z tabulky 1 je patrné, že tyto dvě metody mezi sebou ne zcela korespondují.

Nejhorších výsledků v průřezu všech datasedů jednoznačně dosahují algoritmy PCA a Weighted Bisector, které na první pohled nedokázaly určit hlavní směr budov. Nicméně výsledky hodnotícího kritéria naznačují opak. Právě tyto dvě metody často dosahují nejvyšších hodnot. V případě datasetu Vinohrad vykazuje algoritmus Weighted Bisector nejvyšší hodnoty. V případě datasetu Braníku dopadl na druhém místě a v případě Starého Města právoplatně na posledním místě. Algoritmus PCA pak dopadl ve všech případech na třetím místě. Hodnotící kritérium tedy nefunguje bez chyb a nabízí se jeho další vylepšení.

Metoda	**Vinohrady**	**Braník**	**Staré Město**
Minimum Area Enclosing Rectangle	63.369	62.378	64.0
Principal Component Analysis	65.775	67.184	66.337
Longest Edge	62.567	66.408	74.257
Wall Average	67.914	69.709	76.733
Weighted Bisector	71.925	67.961	59.406

Tabulka 1: Hodnocení efektivity detekce hlavních směrů pomocí hodnotícího kritéria

Závěr

V rámci této úlohy byla vytvořena aplikace, která umožňuje jistou míru automatizace při řešení problému generalizace budov. Testovanými generalizačními algoritmy byly *Minimum Area Enclosing Rectangle, , Principal Component Analysis, Longest Edge, Wall Average a Weighted Bisector*.

Výsledná aplikace nabízí ještě spoustu prostoru do další vylepšení. Aplikace by například mohla využít více než jednu metodu pro konstrukci konvexní obálky, která je základem pro generalizační algoritmy. Kromě zde testovaného algoritmu *Jarvis Scan* by bylo možné využít například také algoritmu *Graham Scan*. Mezi další vylepšení patří také navrhnutí hodnotícího kritéria, které bude lépe odrážet skutečný výsledek generalizace.

Seznam literatury

BAYER, T. (2024): Konvexní obálka množiny bodů. Přednáška pro předmět Algoritmy počítačové kartografie, Katedra aplikované geoinformatiky a kartografie. Přírodovědecká fakulta UK, dostupné zde (cit. 18. 4. 2024).

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

EBERLEY, D. (2020): Minimum-Area Rectangle Containing a Set of Points. Geometric Tools, Redmond WA 98052, dostupné zde (cit. 18. 4. 2024).

LI, Z. a kol. (2004): Automated building generalization based on urban morphology and Gestalt theory. International Journal of Geographical Information Science, 18, 5, 513–534.

PUNT, E., WATKINS, D. (2010): User-directed generalization of roads and buildings for multi-scale cartography. 13th Workshop of the ICA commission on Generalisation and Multiple Representation.