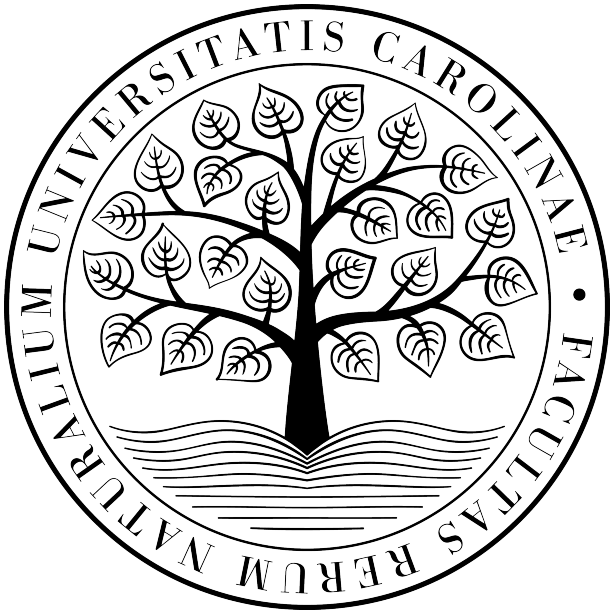
Přírodovědecká fakulta
Univerzita Karlova



Algoritmy počítačové kartografie

Úkol č. 2: Generalizace budov LOD0

Martínek David, Nováková Lucie a Zikešová Anna

1.N-GKDPZ

Praha 2024

1 Zadání

Úloha č. 2: Generalizace budov LOD0

Vstup: množina budov $B = \{Bi\}$*, budova* $Bi = \{P_{i,j}\}$
Výstup: $G(Bi)$*.*

Ze souboru načtěte vstupní data představována 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 města, intravilán - sídliště, intravilán - 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. Výsledky generalizace vhodně vizualizujte.

Otestujte a porovnejte 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.	15 b.
Generalizace budov metodou Longest Edge.	*+ 5 b.*
Generalizace budov metodou Wall Average.	*+ 8 b.*
Generalizace budov metodou Weighted Bisector.	*+ 10 b.*
Max celkem:	**38b**

2 Popis a rozbor problému

2.1 Generalizace budov

Kartografická generalizace je jednosměrným procesem, jehož cílem je redukovat množství informací z původní mapy v závislosti na měřítku, typu nové mapy nebo druhu nesené informace. Dochází ke zjednodušení objektů. Změny původní mapy vyvolané primárně změnou měřítka zpravidla působí proti sobě, čímž nelze dodržet veškerá generalizační pravidla. Proces generalizace je časově i finančně náročný. Z toho důvodu je potřebná automatizace tohoto procesu. Nejprve se automatizovaná generalizace věnovala generalizaci linií nebo přírodních ploch, poté se pozornost obrátila na kontextuaální generalizaci, při které jde o generalizaci s ohledem na prostor. Jedním z typů kontextuální generalizace je typifikace, při níž je redukován počet objektů ve skupině při zachování původního vzhledu skupiny. Je respektována hustota, pořadí objektů, velikost, orientace. Typyfikace je prezentována převážně jako generalizace budov. Z důvodu neustálé aktualizace zástavby, jako je nová výstavba či bourání budov, je tendence k automatizaci procesu. Protože je nutné tyto mapy neustále aktualizovat (Gottstein, 2021).

Online prostředí vyžaduje mapy v mnoha měřítkách. Je v něm nutné zobrazit celou Zemi až po přiblížení jednotlivých budov. Online mapy jako jsou Google Maps, ArcGIS Online World Streetmap atd. zobrazují ulice, proto je nutné je stále aktualizovat (Punt, Watkins, 2010).

Operace generalizace budov je z důvodu složitého uspořádání v prostoru, kvůli rozpoznávání objektů komplikovanou operací.Výstupem je generalizace zachovávající charakter a vztahy mezi sousedními prvky a zjednodušené zobrazení zachovávající obrazovou přesnost (Punt, Watkins 2010). Pro automatizovanou generalizaci existuje mnoho algoritmů pro dílčí zobecnění, dle problému např. *zobecnění, agregace*, i přesto existuje spousta případů, které nejsou zobecněny adekvátně (Sester, Feng, Thiermann, 2018).

Atomatizovaná generalizace se zpravidla dělí do třech kroků. Nejprve je provedena analýza prostorových souvislostí geografických prvků, dále je provedeno algoritmické zpracování a posledním krokem vyhodnocení výsledků generalizace.

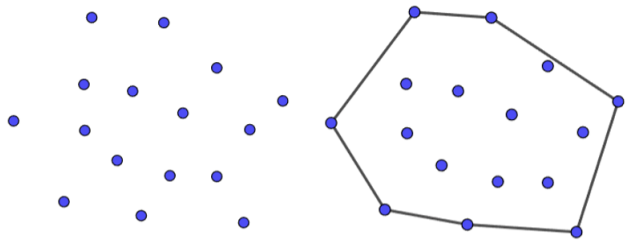
Pro automatizovanou generaci budov je podstatné znát orientaci budov, tedy orientaci mnohoúhelníku, což je složitý proces. Orientace budov se zjistí buď orientací stěn mnohoúhelníku nebo obecnou orientací (Duchene, 2003).

Pro generalizaci budov je podstatné to, aby měla před generalizací i po ní stejnou orientaci vzhledem k sousedním prvkům na mapě. Z toho důvodu je nutné určit hlavní směry budovy, které jsou pro orientaci určující. Algoritmy, pomocí kterých lze určit hlavní směry budov, jsou například *Longest Edge*, *Weighted Bisektor*, *Minimum Area Enclosing Rectangle*, *Wall Average* nebo *Metoda hlavních komponent*.

Tato úloha se zabývá generalizací budov za využití *konvexní obálky* a *min-max boxu*, označovaného také jako *bounding box*.

2.2 Konvexní obálka

Konvexní obálka je nejmenší konvexní sada obklopující všechny body množiny a tvoří tak konvexní mnohoúhelník. Algoritmus je důležitý při aplikacích zpracování obrazu, plánování tras, modelování objektů, generalizaci. Konvexní obálka množiny bodů je nejmenším konvexním mnohoúhelníkem (obr. 1), který obklopuje všechny body množiny v dvojrozměrném prostoru se jedná o konvexní mnohoúhelník v trojrozměrném prostoru se jedná o konvexní mnohostěn. Pro konvexní mnohoúhelník platí, že všechny vnitřní úhly jsou menší nebo rovny 180°.



Obrázek 1: Množina bodů a konvexní obálka

Konvexní obálka se řadí mezi nejpoužívanější geometrické struktury a je pomocnou strukturou pro mnoho algoritmů. Využívá se pro detekci kolizí, konstrukci *maximum bounding rectangle*, analýzu tvarů, statistickou analýzu, analýzu klastrů.

Pro sestrojení konvexních obálek existuje několik algoritmů, například: *Jarvis Scan*, *Graham Scan*, *Quick Hull*, *Inkrementální konstrukce* nebo *Divide and Conquer*. Jednotlivé algoritmy se pak od sebe liší především časovou a paměťovou náročností.

2.2.1 Jarvis Scan

Jarvis Scan nebo také *Gift Wrapping Algorithm* patří mezi nejpoužívanější algoritmy pro konstrukci konvexní obálky a to z důvodu jednoduchosti implementace. Je vhodné ho využít pro vstupní data, jejichž množina bodů n není příliš velká.

Algoritmus nejprve vybere bod q s minimální hodnotou y nebo bod nejvíce vlevo, tento bod je označen jako *Pivot*. Následně je bod přidán do seznamu vrcholů tvořící konvexní obálku. Následně je vytvořena pomocná přímka procházející bodem q rovnoběžná s osou x. Dalším bodem přidaným do seznamu je bod q_{j1}, který svírá největší úhel ω s pomocnou přímkou. Algoritmus je ukončen ve chvíli, kdy hledaný bod odpovídá *Pivotu*.

Algorithm 1 *Jarvis Scan*

```
Inicializuj konvexní obálku ch
Najdi bod q s minimální y souřadnicí
Najdi bod s s minimální x souřadnicí
Inicializuj poslední dva body konvexní obálky q_j = q a q_j1 = [s, q]
Přidej pivot q do konvexní obálky ch
Iteruj přes všechny body z polygonu pol
    Inicializuj hodnoty ω_max = 0.0 a index_max = −1
    Pokud q_j ≠ pol_(i):
        Spočítej úhel ω mezi úsečkou danou body q_j1 a q_j a úsečkou q_j a pol_(i)
            Pokud ω > ω_max:
                Aktualizuj ω_max = ω a index_max = i
    Přidej pol(index_max) do ch
    Skonči cyklus, pokud q = pol(index_max)
    Aktualizuj poslední dva body qj1 = qj a qj = pol(index_max)
```

Pseudokód metody Jarvis Scan

2.3 Minimum Area Enclosing Rectangle

Algoritmus *Minimum Area Enclosing Rectangle* funguje na principu nalezení obdélníku, který má minimální plochu. Algoritmus je též označován jako *Minimum Area Bounding Rectangle*. Je dána množina bodů v rovině. A výstupem je obdélník opisující právě tuto množinu. Obdélník nelze zkonstruovat hned v prvním kroku, ale je nutné množinu předzpracovat do konvexní obálky. Obdélník na výstupu má alespoň jednu hranu kolineární s hranou konvexní obálky.

Následně je nalezen *Minimum Bounding Box* pro kaýdou hranu konvexní obálky. Hrana *Minimum bauding boxu* je rovnoběžná s právě procházenou hranou konvexní obálky. Ze všech vytvořených obdélníků je následně vybrán ten s minimální plochou. V rámci kódu je tedy nutné uchovávat informaci o nejmenší ploše obdélníka a zároveň uchovávat kolineární hranu s hranou minimálního opsaného obdélníku. Následně je čku zjistíme poměr mezi plochou budovy a plochou minimálního MMB:

$$k = \frac{S_b}{S_{er}}$$

Algoritmus slouží k určení hlavních směrů budovy, na jejichž základě lze určit orientaci budovy. Prvním krokem je vytvoření konvexní obálky, která je opakovaně otáčena na základě rotační matice o úhel $-\sigma$, který je směrnicí konvexní obálky.

$$P_1 = R(-\sigma)P \Rightarrow \begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \cos(-\sigma) & -\sin(-\sigma) \\ \sin(-\sigma) & \cos(-\sigma) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Na závěr jsou vypočítány polohy nově vzniklých bodů pomocí těžiště, kdy do výpočtu vstupuje vektor pro

všechny vrcholy obdélníku MMB.

Pseudokód metody Minimum Area Enclosing Rectangle

Algorithm 2 *Minimum Area Enclosing Rectangle*

```
Vytvoř konvexní obálku ch pro množinu bodů v polygonu pol
Najdi MMB_min a vypočítej jeho plochu S_min
Pro každou hranu v ch:
    Vypočítej souřadnicové rozdíly dx a dy
    Vypočítej směrnici σ pro dx a dy
    Rotuj ch o −σ
    Zkontroluj MMB_rotovaný a jeho plochu S_rotovaný
    Pokud S_rotovaný < S_min:
        MMB_min = MMB_rotovaný
        σ_min = σ
        S_min = S_rotovaný
Otoč zpátky MMB_min o σ_min
Přepočítej plochu tak, aby plocha MMB_nerotovaný odpovídala ploše polygonu pol
```

2.4 Metoda hlavních komponent

Metoda hlavních komponent neboli *PCA* je taktéž algoritmem zjišťující hlavní směry budovy, tedy orientaci. V rámci *PCA* je aplikován statistický postup využívající ortogonální transformaci. Pro nalezení hlavních směrů je využíván singulární rozklad matice. Nejprve je potřeba vypočítat kovarianční matice C:

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

Kovariance obecně měří variabilitu mezi dvěma nebo více proměnnými, její hodnota může být kladná, záporná, tak i nulová. V dalším kroku jsou vypočítány vlastní čísla a vlastní vektory matice U, V.

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

Matice $\sum$ má singulární hodnoty a tvoří ji velikosti vlastních vektorů:

$$\sum = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} = \begin{bmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \end{bmatrix}$$

Následně je proveden singulární rozklad matice:

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

Rotace množiny o úhel $\pm\omega$ je:

$$P_0 = PVP = V^{-1}P_0 \tag{1}$$

Pseudokód metody PCA

Algorithm 3 *PCA*

Inicializuj prázdné seznamy souřadnic x a y
Pro všechny body plygonu *pol*:
    Přidej souřadnice každého bodu do seznamů x a y
Vytvoř pole P souřadnic
Vypočítej kovarianční matici C
Proveď singulární rozklad nad kovarianční maticí C
Vypočítej úhel σ funkcí *atan* ze směru první hlavní komponenty
Rotuj *pol* o $-\sigma$
Spočítej MMB z rotovaného *pol*
Vytvoř obdélník *er* rotací MMB zpátky o σ
Přepočítej plochu tak, aby plocha *er* odpovídala ploše polygonu *pol*

2.5 Longest Edge

Hlavní směr budovy v rámci algoritmu *Longest Edge* je určen nejdelší stranou v budově. Druhý směr je pak kolmý na první směr. Podél nejdelší strany budovy je pak natočen výsledný obdélník, jehož delší strana je rovnoběžná s nejdelší hranou.

Budova je následně otočena o úhel $-\sigma$ Kolem nejdelší hrany je zkonstruován $min-maxbox$, který je následně otočen o hodnotu úhlu σ a dále je upraven, tak aby jrho plocha odpovídala ploše budovy.

Pseudokód metody Longest Edge

Algorithm 4 *Longest Edge*

Inicializuj proměnnou pro nejdelší hranu $edge_{longest}$ na 0
Pro všechny body v polygonu *pol*:
    Vypočítej Euklidovské vzdálenosti $edge_{length}$ ze souřadnicových rozdílů dx a dy
    Pokud $edge_{length} > edge_{longest}$:
        Aktualizuj nejdelší hranu $edge_{length} = edge_{longest}$
        Vypočítej její směrnici σ
Rotuj *pol* o $-\sigma$
Zkonstruuj MMB z rotovaného *pol*
Vytvoř obdélník *er* rotací MMB zpátky o σ
Přepočítej plochu tak, aby plocha *er* odpovídala ploše polygonu *pol*

2.6 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'.

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

Následně se pro každou hodnotu vypočítá zaokrouhlený podíl k_i:

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

Dále je potřeba vypočítat zbytky r_i a odchylky od $0 \pm k\pi$:

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

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:

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

kde s_i je délkou hrany i.

Pseudokód metody Wall Average

Algorithm 5 *Wall Average*

Vypočítej úhel σ_0 mezi dvěma prvními po sobě jdoucími body
Inicializuj proměnnou $r_{average}$ pro uložení součtu orientovaných zbytků
Pro každou hranu:
    Vypočítej úhel σ mezi po sobě jdoucími body
    Vypočítej vnitřní úhel ω jako absolutní hodnotu z rozdílu $\sigma - \sigma_0$
    Rozděl ω na násobky π
    Vypočítej orientované zbytky $r = (\omega - k) * \pi/2$
    Sečti r s $r_{average}$
Vypočítej výsledý úhel $\sigma_{average} = r_{average}/n + sigma_0$
Rotuj *pol* o $-\sigma_{average}$
Zkonstruuj MMB z rotovaného *pol*
Vytvoř obdélník *er* rotací MMB zpátky o $\sigma_{average}$
Přepočítej plochu tak, aby plocha *er* odpovídala ploše polygonu *pol*

2.7 Weighted Bisector

Při zjišťování orientace pomocí algoritmu **Weighted Bisector** jsou nalezeny dvě nejdelší úhlopříčky v polygonu. Jde o nalezení úhlopříček, které neprotínají žádnou z hran polygonu. V případě, že nově vzniklá úsečka prochází skrz hranu budovy, je vyloučena jako možná úhlopříčka. Jestli úsečka prochází nebo neprochází hranou je zjištěno pomocí průsečíků, kdy jsou porovnány polohy koncových bodů jedné úsečky oproti koncovým

bodům úsečky druhé. Jde o výpočet determinantu směrových vektorů:

$$t_1 = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_4 - x_1 & y_4 - y_1 \end{vmatrix} \quad t_2 = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} \quad t_3 = \begin{vmatrix} x_4 - x_3 & y_4 - y_3 \\ x_2 - x_3 & y_2 - y_3 \end{vmatrix} \quad t_4 = \begin{vmatrix} x_4 - x_3 & y_4 - y_3 \\ x_2 - x_3 & y_2 - y_3 \end{vmatrix}$$

Podle znaménka výsledného determinantu směrových vektorů lze určit, zda existuje průsečík s hranou. V případě, že by $t1$ a $t2$ nebo $t3$ a $t4$ měly stejné znaménko, průsečík neexistuje, úsečka tedy neprochází hranou, jedná se o úhlopříčku. Dále jsou vybrány dvě nejdelší úhlopříčky, pro které je vypočítána jejich směrnice. Výsledná orientace budovy je dána váženým průměrem vypočítaných směrnic. Jako váhy do vzorečku vstupují délky daných úhlopříček (s_1 a s_2).

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

Pseudokód metody Weighted Bisector

Algorithm 6 *Weighted Bisector*

1: Vytvoř konvexní obálku *ch* pro množinu bodů v polygonu *pol*
2: Pokud má některý polygon v *ch* méně než 4 vrcholy:
3: Vrať hodnotu *None*
4: Inicializuj proměnné pro nejdelší úhlopříčky $diag_1$ a $diag_2$ a jejich počáteční a koncové body
5: Pro všechny body v *ch*:
6: Spočítej vzdálenost mezi body $diag_potential$
7: Pokud je $diag_potential > diag_1$ a $diag_potential > diag_2$:
8: Aktualizuj příslušné úhlopříčky a jejich odpovídající počáteční a koncové body
9: Vypočítej úhly σ_1 a σ_2 dvou nejdelších úhlopříček
10: Vypočítej úhel σ jako vážený průměr σ_1 a σ_2, kde váhy jsou délky $diag_1$ a $diag_2$
11: Rotuj *pol* o $-\sigma$
12: Zkonstruuj MMB z rotovaného *pol*
13: Vytvoř obdélník *er* rotací MMB zpátky o σ
14: Přepočítej plochu tak, aby plocha *er* odpovídala ploše polygonu *pol*

3 Aplikace

Vytvořená aplikace pro generalizaci budov bylavytvořena v prostředí *QT Creator*. V rámci aplikace pak byly implementovány popsané algoritmy: *Minimum Area Enclosing Rectangle*, *PCA*, *Longest Edge*, *Wall Average* a *Weighted Bisector*.

Po spuštění aplikace přes soubor *MainForm.py* je otevřeno okno, jehož hlavní část tvoří plocha pro vykreslení polygonů. V horní části okna se nachází menu s jednotlivými funkcemi. V záložce *File* či na liště je pomocí funkce *Open* možné vyvolat dialogové okno a vybrat vstupní data ve formátu *.shp*, která jsou následně vykreslena (obr. 2).

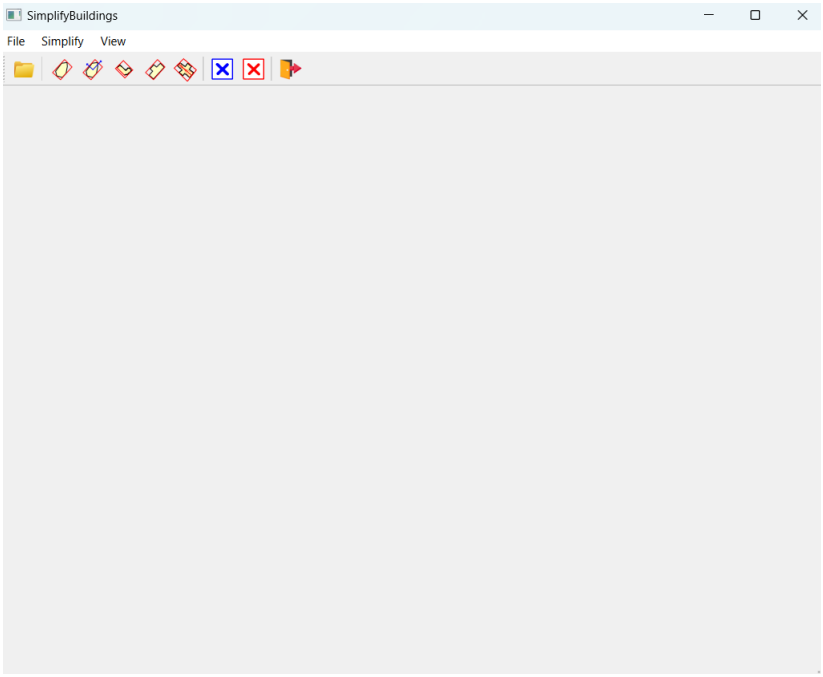
Nahrané polygony je možné generalizovat funkcemi v záložce *Simplify* či jednotlivými funkcemi na liště. Lze tedy využít algoritmy pro generalizaci: *Minimum Area Enclosing Rectangle*, *PCA*, *Longest Edge*, *Wall Average* a *Weighted Bisector*. Možností *ClearResults* jsou vymazány generalizované polygony a funkcí *ClearAll* jsou vymazány jak výsledky generalizace, tak nahrané polygony. Funkce *Exit* slouží k zavření celého okna.

3.1 Vstupní data

Aplikace je přizpůsobena k načítání geografických vstupních informací ve formátu *.shp*. Jednotlivé soubory byly získány z webové stránky *GeoportalPraha*, konkrétně z digitální technické mapy Prahy. Vstupní data jsou v souřadnicovém systému $S-JTSK$. Vstupní data jsou uloženy ve složce *polygony budov male* a vybrané městské části jsou uloženy zvlášť ve složkách se svým jménem:

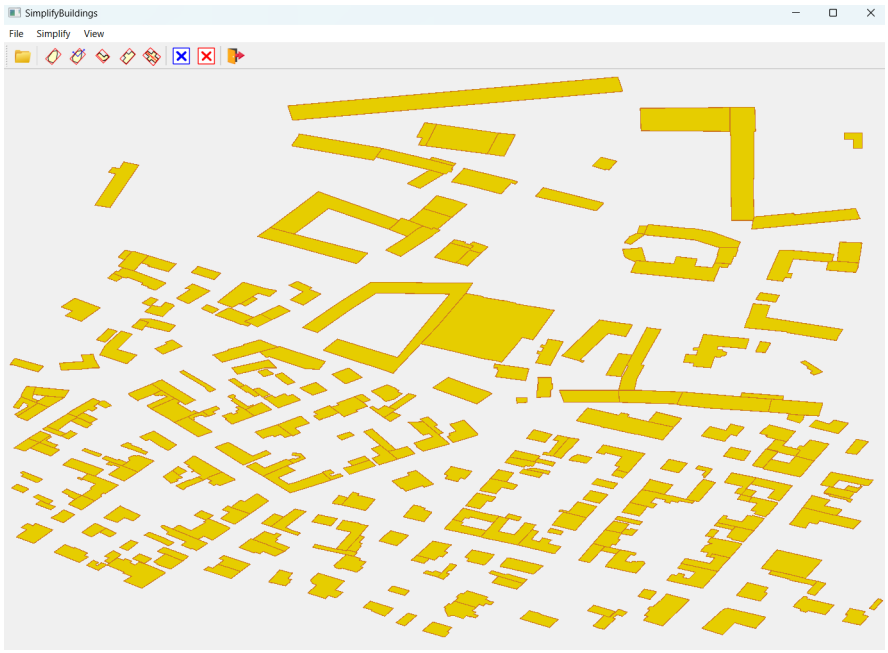
- Bohnice (obr. 3),
- Ďáblice (obr. 4),
- Vinohrady (obr. 5).

Pro práci s daty bylo potřeba jednotlivé vrstvy oříznou, aby se s nimi následně lépe pracovalo.



Obrázek 2: Ukázka aplikace

Obrázek 3: Vstupní data Bohnice



Obrázek 4: Vstupní data Ďáblice

Obrázek 5: Vstupní data Vinohrady

4 Dokumentace

Program byl vytvořen v prostředí PyCharm v programovacím jazyce Python s využitím *QT Creator* a modulu *pyshp*, který umožňuje práci se shapefily. Program se skládá ze souborů *MainForm.py*, *draw.py* a *algorithms.py*, které názvem odpovídají příslušným třídám. Ve složce *icons* se nachází obrázkové soubory využité pro ikony jednotlivých funkcí.

4.1 Třída MainForm:

Třída *MainForm* zabezpečuje inicializaci okna samotné aplikace, horní lišty, panelu nástrojů, ikon a tlačítek. Propojuje jednotlivé interaktivní položky s metadami vykonávající specifické akce.

Metody *setupUi* a *retranslateUi* byly vytvořeny automaticky na základě vytvořeného prostředí v *Qt Creator*. V první z nich je mimo jiné provedeno napojení jednotlivých položek menu a tlačítek na připravené metody.

Metoda *openClick* ukládá do proměnných w a h aktuální šířku a výšku okna pro vykreslování polygonů. Následně volá metody *getData* a *setView* třídy *Draw*, čímž dochází k načtení vstupních dat a jejich vykreslení.

Metody *pcaClick*, *maerClick*, *longestEdgeClick*, *wallAverageClick* a *weightedBisectorClick* zajišťují generalizaci nahraných polygonů. Všechny nejprve pomocí metody *getBuilding* z třídy *Draw* získají seznam vykreslených polygonů. Následně prochází všechny polygony a pomocí odpovídající metody z třídy *Algorithms*, které předává aktuální polygon, získává obdélník generalizující danou budovu. Tu následně metodou *setMBR* předává třídě *Draw* a okno je překresleno metodou *repaint*. Po spuštění každé metody se v terminálu ukáže

procentuální hodnocení úspěšnosti spuštěné metody.

Metoda *clearAllClick* slouží k vymazání nahraných budov a jejich generalizovaných polygonů a volá metodu *clearData* třídy *Draw*.

Metoda *clearClick* slouží k vymazání generalizovaných polygonů a volá metodu *clearResults* třídy *Draw*.

4.2 Třída Draw:

Třída *Draw* složí k inicializaci proměnných a zajišťuje grafické rozhraní aplikace. Inicializační metoda má dva poziční argumenty a dochází v ní k inicializaci pěti proměnných. Proměnná *shapefile* sloužící pro ukládání objektů z načteného shapefilu je inicializována jako *None*. Dále je inicializován seznam *border* pro ukládání minimálních a maximálních souřadnic polygonů a proměnná *missingFile*. Proměnná *buildings* je seznam sloužící pro ukládání vykreslovaných polygonů a seznam *mbr* pro ukládání generalizovaných polygonů.

Metoda *getData* slouží k načtení vstupních dat. Prvně dojde k vyvolání dialogového okna, ze kterého je získána cesta k souboru vstupních dat. Pokud soubor není vybrán a okno je zavřeno, dojde k aktualizaci proměnné *missingFile* na hodnotu *True* a metoda se ukončí. Když je soubor vybrán, tak je shapefile načten. Nakonec jsou všechny polygony procházeny a jsou získány minimální a maximální hodnoty souřadnic X a Y.

Metoda *setView* slouží k úpravě souřadnic bodů jednotlivých polygonů, aby je bylo možné vykreslit v okně aplikace. Na vstupu má argumenty w pro šířku vykreslení a h pro výšku vykreslení, které odpovídají aktuální šířce a výšce okna pro vykreslení polygonů. Pokud je proměnná *missingFile* nastavena na hodnotu *True*, vykreslí se jeden polygon, který leží mimo vykreslené okno, čímž je zabráněno pádu aplikace při nezvolení cesty k souboru se vstupními daty. V případě, že byl soubor vybrán, tak dojde k inicializaci seznamu polygonů, dle počtu objektů proměnné *buildings*. Následně jsou tyto objekty procházeny a na základě minimálních a maximálních souřadnic X a Y, velikosti okna jsou přepočítány souřadnice objektů. Ve chvíli, kdy jsou souřadnice přepočteny dojde k vytvoření bodu typu *QPointF* a jeho přidání do polygonu.

Metoda *paintEvent* má na vstupu argument *QPaintEvent*. Tato metoda slouží k vykreslení budov a jejich generalizovaných polygonů. Pomocí cyklu *for* jsou procházeny všechny budovy a je nastavené jednotné vybarvení budov a následně jsou vykresleny. Stejným způsobem jsou vykresleny i jejich generalizované polygony uložené v seznamu *mbr*.

Metoda *getBuildings* předává vytvořený seznam uchovávající všechny nahrané budovy.

Metoda *setMBR* má na vstupu polygon typu *QPolygonF*, který přidává do seznamu generalizovaných budov.

Metoda *clearData* slouží k vymazání obsahu vykresleného okna, což je provedeno vymazáním obsahu seznamů *buildings* a *mbr*. A vyvoláním metody *repaint*, které okno překreslí.

Metoda *clearResults* slouží k vymazání výsledků generalizovaných polygonů. K čemuž dojde vymazáním obsahu seznamu *mbr*. Následně je okno překresleno. Polygony budov vykreslené zůstavájí.

4.3 Třída Algorithms

Třída *Algorithms* implementuje algoritmy pro generalizaci budov. Obsahuje metodu pro konstrukci konvexní obálky, metody pro určení hlavních směrů budov, pomocné metody a metodu pro hodncení efektivity použitých generalizačních algoritmů.

Metoda *getTwoLineAngle* má na vstupu čtyři body typu $QPointF$ tvořící dvě linie, mezi kterými je počítán úhel. Nejprve jsou vypočteny dva vektory, jejich skalární součin a normy obou vektorů. Následně je vytvořeno opatření pro pád algoritmu a to tak, že ve chvíli, kdy je normový vektor roven nule je vrácena nula. Následně je spočítán argument pro funkci *cosinus*, který je zaokrouhlen na hodnotu 1 nebo -1, pokud je větší, respektive menší. Metoda pak vrací vypočítaný úhel.

Metoda *accuracyEvaluation* má na vstupu dva polygony typu $QPolygonF$ a slouží k hodnocení přesnosti jednotlivých metod. Metoda vyhodnocuje přesnost orientace výsledného zjednodušeného polygonu *er* vzhledem k polygonu budovy *pol*. Vypočítá rozdíl úhlů mezi odpovídajícími hranami dvou polygonů a zkontroluje, zda spadá do prahové hodnoty. Pokud ano, znamená to vysokou přesnost, vrátí 1, jinak vrátí 0.

Metoda $cHull$ slouží k vytvoření konvexní obálky pomocí algoritmu *Jarvis scan*. Metoda má na vstupu polygon typu $QPolygonF$. Podrobněji je algoritmus popsán v kapitole *Popis a rozbor problému*. Metoda vrací vytvořenou konvexní obálku typu $QPolygonF$.

Metoda $minmaxBox$ má na vstupu polygon typu $QPolygonF$. Slouží k nalezení minimálních a maximálních souřadnic polygonu a k vytvoření obdélníku, jehož vrcholy jsou definovány těmito hodnotami. Metoda pak vrací vytvořený obdélník.

Metoda *rotate* má na vstupu polygon typu $QPolygonF$ a úhel datového typu $float$. Metoda slouží k otočení vybraného polygonu o daný úhel. Nejprve je vytvořena proměnná pro rotovaný polygon pol_r typu $QPolygonF$, pro všechny body polygonu jsou přes *maticirotace* vypočítány souřadnice rotovaného polygonu. Metoda vrací zrotovaný polygon.

Metoda *getArea* slouží k vypočtení plochy polygonu. Na vsupu má polygon typu $QPolygonF$. A vrací vypočítánou plochu polygonu.

Metoda *resizeRectangle* má na vstupu polygon obdélníku a polygon budovy typu $QPolygonF$. Metoda počítá poměr plochy původní a generalizované budovy. Pro problémové budovy je vrácena hodnota $None$, jsou to takové budovy pro které se plocha generalizované budovy rovná nula. Na základě vypočteného poměru se plocha generalizované budovy zvětšuje či zmenšuje, tak aby odpovídala ploše původní budovy. Metoda vrací zmenšený či zvětšený obdélník typu $QPolygonF$.

Metoda *mbr* má na vstupu polygon typu $QPolygonF$. Metoda slouží k inicializaci algoritmu $MinimumArea EnclosingRectangle$. Podrobněji je algoritmus popsán v kapitole *Popis a rozbor problému*. Metoda vrací polygon typu $QPolygonF$.

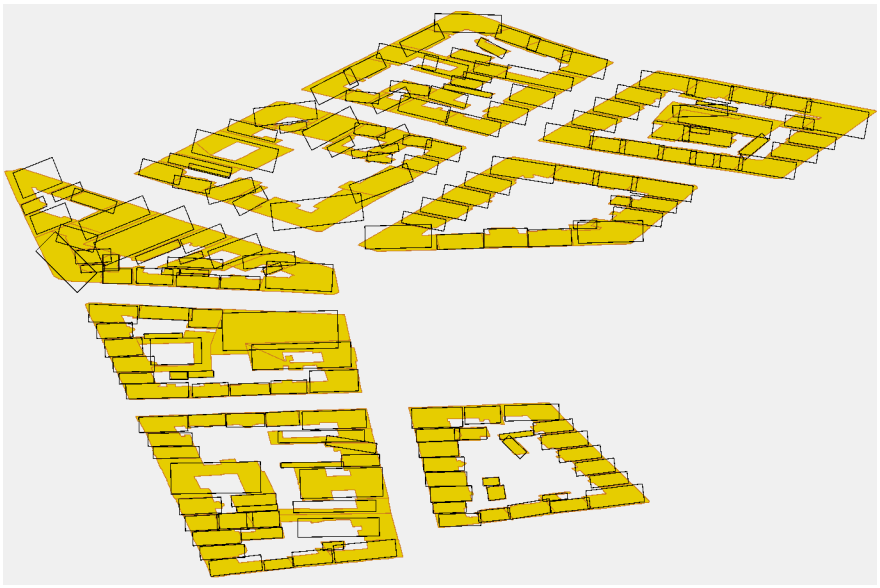
Metoda $createERPCA$ slouží k inicializaci algoritmu PCA. Na vstupu má polygon typu $QPolygonF$. Podrobněji je algoritmus popsán v kapitole *Popis a rozbor problému*. Metoda vrací polygon typu $QPolygonF$.

Metody $longestEdge$, $wallAverage$ a $weightedBisector$ provadí aloritmy $LongestEdge$, $WallAverage$ a $WeightedBisector$. Na vstupu mají každá polygon typu $QPolygonF$. Jsou popsány podrobně v kapitole *Popis a rozbor problému*. Metody vrací polygon typu $QPolygonF$.

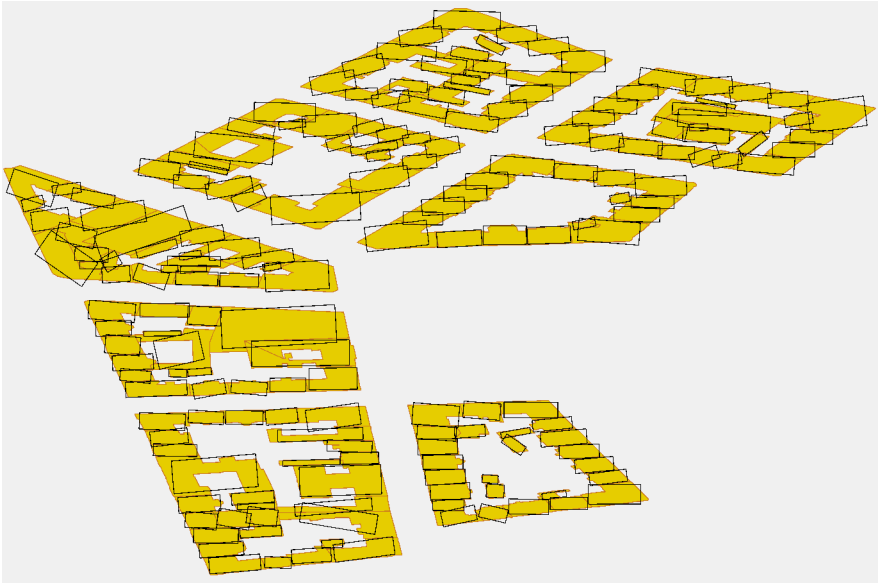
5 Výsledky

Generalizace budov pomocí jednotlivých algoritmů byla testována na třech datasetech. Dataset použitý v ukázkách je centrum města Praha-Vinohrady (obr. 5).

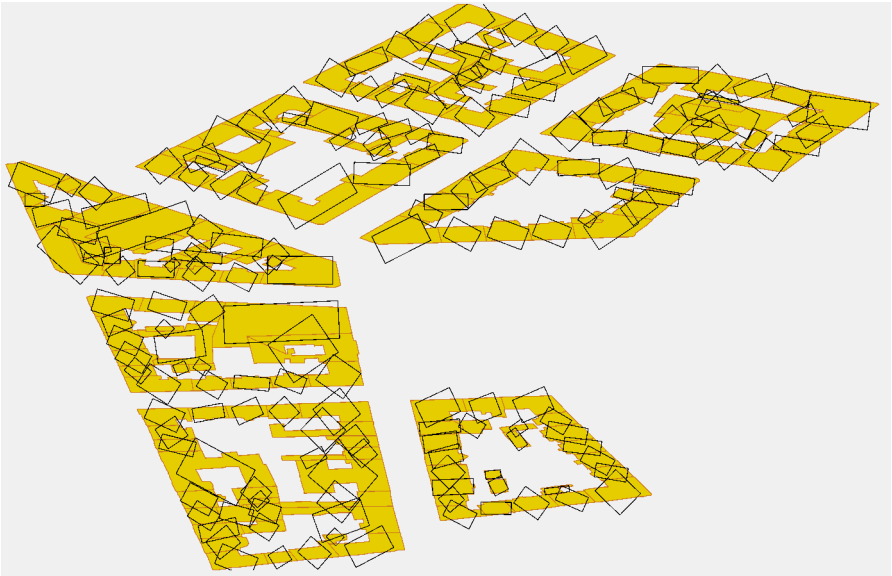
Na obrázcích 6 až 10 je možné porovnat výsledky generalizace celého datasetu všemi implementovanými metodami. Při vizuálním porovnání nejlepších výsledků dosáhl algoritmus *Minimum Area Enclosing Rectangle*. Téměř ve všech případech generalizované budovy kopírují hlavný směry budovy a natočení odpovídá původnímu natočení budovy. Obdobně dobrých výsledků dosáhl i algoritmus *Longest Edge*. V tomto případě nejdelší hrana vždy nekopíruje hlavní směr budovy. Stejně tak algoritmus *PCA* dosahuje vizuálně dobrých výsledků. V těchto třech algoritmech jsou generalizované budovy jinak natočeny. Algoritmus *Weighted Bisector* dosahuje již mnohem horších výsledků. U části budov nedopovídá natočení generalizovaného polygonu směru původní budovy a u některých obdélníkových budov je jejich generalizace spíše čtvercová. Tyto nedostatky se ještě výrazněji projevují u algoritmu *Wall Average*, který dosáhnul nejhorších výsledků.



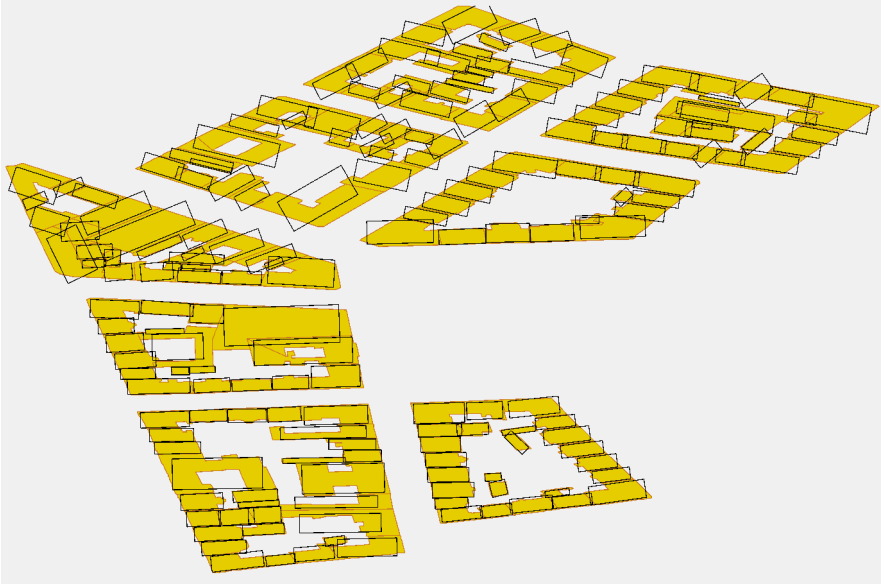
Obrázek 6: Ukázka generalizovaných budov – Minimum Area Enclosing Rectangle



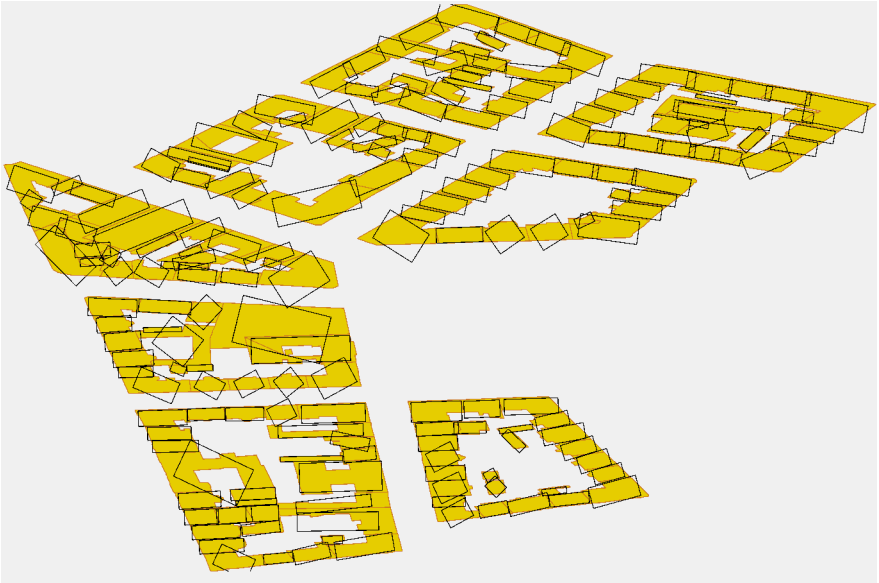
Obrázek 7: Ukázka generalizovaných budov – PCA



Obrázek 8: Ukázka generalizovaných budov – Wall Average

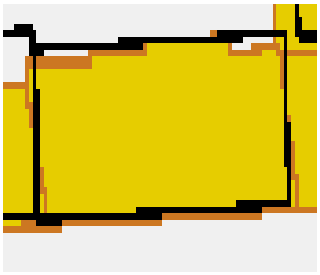


Obrázek 9: Ukázka generalizovaných budov – Longest Edge

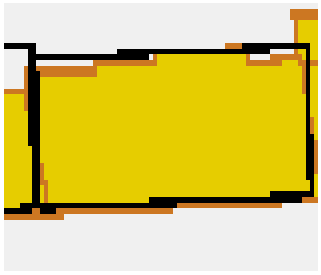


Obrázek 10: Ukázka generalizovaných budov – Weighted Bisector

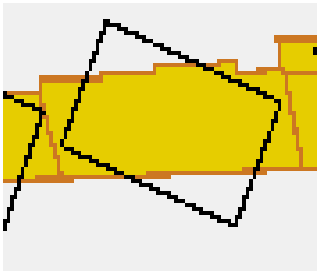
Výsledky jednotlivých algoritmů byly otestovány i na jednotlivých tvarech budov. Obrázky 11 až 15 zobrazují generalizovanou budovu obdélníkového tvaru. Nejlepších výsledků dosahují algoritmy *Minimum Area Enclosing Rectangle*, *PCA* a *Longest Edge*, které tvar budovy dobře kopírují. U algoritmů *Wall Average* a *Weighted Bisector* se pak liší poměr stran polygonu a i natočení je oproti původní budově rozdílné.



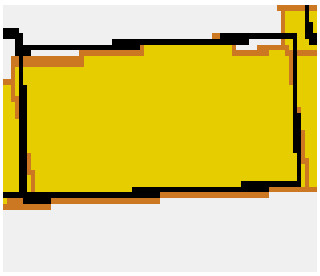
Obrázek 11: Pravidelná budova – Minimum Area Enclosing Rectangle



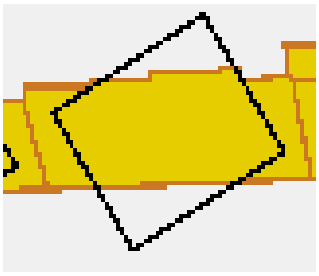
Obrázek 12: Pravidelná budova – PCA



Obrázek 13: Pravidelná budova – Wall Average

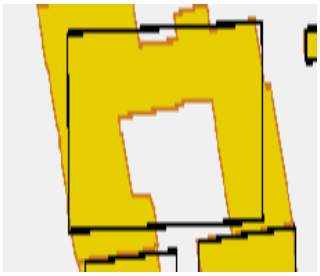


Obrázek 14: Pravidelná budova – Longest Edge

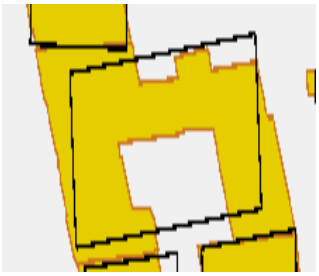


Obrázek 15: Pravidelná budova – Weighted Bisector

Na obrázcích 16 až 20 je zobrazena budova nepravidelného tvaru, u níž dosáhly algoritmy *Minimum Area Enclosing Rectangle*, *PCA* a *Longest Edge* obdobného výsledku, který dobře generalizuje budovu. Generalizované budovy algoritmem *Wall Average* a *Weighted Bisector* tvarem odpovídají předchotím, ale polygon je jinak natočený, nekopíruje hlavní směry budovy.



Obrázek 16: Nepravidelný tvar budovy – Minimum Area Enclosing Rectangle



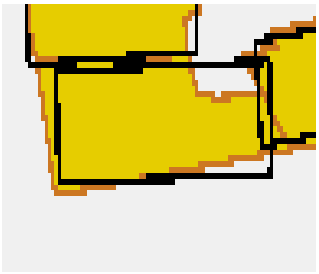
Obrázek 17: Nepravidelný tvar budovy – PCA

Obrázek 18: Nepravidelný tvar budovy – Wall Average

Obrázek 19: Nepravidelný tvar budovy – Longest Edge

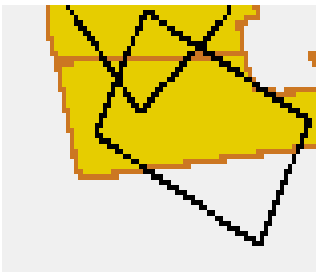
Obrázek 20: Nepravidelný tvar budovy – Weighted Bisector

Na obrázcích 21 až 25 je porovnávána generalizace na typu bodov do tvaru L. Nejlepších výsledků zde dosahují algoritmy *Minimum Area Enclosing Rectangle* a *Longest Edge*. Dobrého výsledku dosahuje i algoritmus *PCA* a *Wall Average*, kde dochází k mírnému natočení oproti vizuálnímu hlavnímu směru budovy. Nejhoršího výsledku pak dosáhl algoritmus *Weighted Bisector*, kde je taktéž natočení vůči hlavnímu směru budovy a generalizovaný polygon se také blíží spíše čtvercovému tvaru.

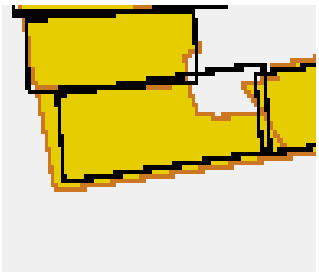


Obrázek 21: Budova ve tvaru L – Minimum Area Enclosing Rectangle

Obrázek 22: Budova ve tvaru L – PCA



Obrázek 23: Budova ve tvaru L – Wall Average



Obrázek 24: Budova ve tvaru L – Longest Edge

Obrázek 25: Budova ve tvaru L – Weigted Bisector

Porovnání přesnosti algoritmů pro celé 3 datasety je uvedeno v tabulce 1.

Budovy	MAER	PCA	Longest Edge	Wall Average	Weighted Bisector
Vinohrady (centrum)	72.43 %	65.40 %	74.05 %	61.08 %	78.91 %
Bohnice (sídliště)	63.63 %	57.22 %	62.57 %	70.05 %	82.49 %
Ďáblice (vilová čtvrť)	64.60%	63.23%	66.67 %	62.89 %	77.66 %
průměr	66.89 %	61.95 %	67.76%	64.67 %	79.%

Tabulka 1: Úspěšnost jednotlivých algoritmů

6 Závěr

V rámci této úlohy byla vytvořena aplikace, která generalizuje budovy na základě uživatelem zvoleného algoritmu (*Minimum Area Enclosing Rectangle*, *PCA*, *Wall Average*, *Longest Edge* a *Weighted Bisector*). Datasety, na které lze algoritmy aplikovat obsahují různé typy zástavby, historickouo část města - Vinohrady, sídliště - Bohnice a nesouvislou vilovou zástavbu - Ďáblice.

7 Zdroje

přednášky z předmětu *Algoritmy počítačové kartografie*, dostupné z: http://web.natur.cuni.cz/ bayertom/ index.php/teaching/algoritmy-pocitacove-kartografie

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

GOTTSTEIN, O. (2021): Generalizace zástavby s využitím typikiface. Academia.

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.

SESTER, M., FENG, Y., THIERMANN, F. (2018): Building generalization using deep learning. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-4, 2018 ISPRS TC IV Mid-term Symposium "3D Spatial Information Science – The Engine of Change".