Úloha 2 – Generalizace budov

Algoritmy počítačové kartografie

Sabina Fukalová, Klára Linková

Praha 2023

# Zadání úlohy

## Povinná část

*Vstup: množina budov B = {Bi}, budova Bi = {Pi,j}mj=1.*

*Výstup: G(Bi).*

Ze souboru načtěte vstupní data představovaná lomovými body budov. Pro tyto účely použijte vhodnou datovou sadu, např. ZABAGED.

Pro každou budovu určete její hlavní směry metodami:

* Minimum Area Enclosing Rectangle,
* Wall Average.

U první metody použijte některý z algoritmů pro konstrukci konvexní obálky. Budovu nahraďte obdélníkem se středem v jejím těžišti orientovaných oblouků v obou hlavních směrech, jeho plocha bude stejná jako plocha budovy. Výsledky generalizace vhodně vizualizujte.

Odhadněte efektivitu obou metod, vzájemně je porovnejte a zhodnoťte. Pokuste se identifikovat, pro které tvary budov dávají metody nevhodné výsledky, a pro které naopak vhodnou aproximaci.

## Volitelná část

Generalizace budov metodou *Longest Edge* (5 b).

Generalizace budov metodou *Weighted Bisector* (8 b).

Implementace další metody konstrukce konvexní obálky (5 b).

Ošetření singulárního případu při generování konvexní obálky (2 b).

# Popis a rozbor problému

Generalizace umožňuje snížit množství informací a zrychlit proces vizualizace. Snaží se odstranit z mapy prvky, které nejsou v kontextu mapy významné. Metoda je však subjektivní, jelikož závisí na pohledu kartografa a jeho zkušenostech. Proto je vhodné zavést algoritmus, který by vhodně zjednodušil původní objekty a nezávisel na kartografovi. Běžně používané generalizační algoritmy ale nemohou být v případě budov použity, protože nezachovávají vnitřní úhly polygonů (-90° a 90°) (Bayer 2008b).

## Konvexní obálka

## *„Je dána množina bodů Pi v E2, kde Pi = [xi, yi]. Konvexní obálka množiny P představuje hranici nejmenší konvexní množiny C obsahující celou množinu P“* (Bayer, 2008a, str. 138)*.* Pro konvexní obálku platí následující:

## žádný z bodů Pi neleží vně konvexní množiny C

* spojnice libovolných dvou prvků Pi leží celá uvnitř konvexní množiny C
* všechny vnitřní úhly mezi sousedními segmenty konvexní obálky jsou menší než 180° (Bayer 2008a).

## Metody konstrukce konvexní obálky

Pro vytvoření konvexní obálky v rovině se využívá několik algoritmů. V následujících kapitolách budou blíže popsány tři nejčastěji používané algoritmy – Grahamovo prohledávání, Jarvisovo prohledávání a algoritmus Quick Hull.

### Grahamovo prohledávání

Při použití Grahamova skenování dochází k převodu nekonvexního mnohoúhelník na konvexní. Využívá se kritérium pravotočivosti, kdy se posuzuje úhel ωi, který svírají úsečky procházející body   
Pi-1, Pi, Pi+1 (obrázek 1). Pokud je úhel ωi větší než 180°, bod Pi+1 leží vlevo od spojnice Pi-1, Pi. Když je úhel ωi menší než 180°, leží bod Pi+1 vpravo od spojnice Pi-1, Pi (Bayer 2008a). Postup Grahamova prohledávání se skládá z následujících kroků:

1. Nalezení pivota Q (např. bod s maximální souřadnicí *x*)
2. Setřídění bodů Pi vzestupně podle úhlu ωmezi osou x a spojnicí Q – Pi
3. Setříděné body jsou spojeny do nekonvexního mnohoúhelníku
4. Posuzování kritéria pravotočivosti u bodů Pi-1, Pi a Pi+1
5. Když ωi > 180°: Prostřední bod Pi leží možná na konvexní obálce – posun o jeden bod vpřed a zkouška kritéria pravotočivosti u bodů Pi, Pi+1, Pi+2
6. Když ωi < 180°: Bod Pi určitě neleží na konvexní obálce – odstraní se ze seznamu, posun o jeden bod zpět a doplnění následujícím vrcholem – opakované prohledávání
7. Výpočet probíhá do té doby, dokud existuje alespoň jedna trojice vrcholů, u nichž by kritérium pravotočivosti ωi bylo menší než 180° (Bayer 2008a)

Obrázek 1: Tvorba konvexní obálky pomocí Grahamova prohledávání

Obsah obrázku tabulka

Popis byl vytvořen automaticky

Zdroj: GeeksforGeeks (2023)

### Jarvisovo prohledávání

Jarvisovo prohledávání je založeno na porovnávání úhlů ω mezi poslední stranou konvexní obálky tvořenou body Pi-1 a Pi, a úsečkou spojující bod Pi a hodnotící bod Pakt (obrázek 2). Pro hledaný bod platí, že tento úhel je maximální. Jednotlivé kroky jsou následující:

1. Nalezení pivota Q (např. bod s nejnižší souřadnicí *y*), který leží automaticky na konvexní obálce
2. Bodem Q je vedena rovnoběžka s osou x
3. Body datasetu jsou postupně zpracovávány a měřeny úhly mezi přímkou procházející bodem Q a úsečkou spojující bod Q a Pakt
4. Ze zpracovávaných bodů je vybrán ten, u něhož je úhel maximální = bod konvexní obálky
5. Opakovaný výpočet úhlu ω mezi úsečkou Pi-1Pi, která spojuje poslední a předposlední bod konvexní obálky a úsečkou Pi Pakt spojující poslední bod konvexní obálky postupně se všemi nezařazenými body – vybrán vždy bod s maximálním úhlem ω
6. Výpočet je ukončen, pokud další bod konvexní obálky bude pivot Q

Obrázek 2: Tvorba konvexní obálky pomocí Jarvisova prohledávání

Obsah obrázku diagram

Popis byl vytvořen automaticky

Zdroj: GeeksforGeeks (2022)

### Quick Hull

Algoritmus Quick Hull je založen na principu, že nepracuje se všemi body vstupní množiny bodů, ale jen s body, které se nachází v blízkosti konvexní obálky. Tvar konvexní obálky je během procesu upravován, ale oproti předchozím algoritmům nejsou body přidávány postupně. Dochází tak k expanzi do všech směrů. Tento algoritmus pracuje ve dvou částech vstupní množiny (tzv. *upper hull* a *lower hull*) a výsledná konvexní obálka pak vznikne jejich spojením (Bayer 2008a). Hranice mezi horní a dolní částí je představována spojnicí dvou bodů min-max boxu s extrémními souřadnicemi *x*. Nad touto přímkou se nachází horní část, dolní část je pod touto přímkou. Jednotlivé kroky jsou následující:

1. Naleznou se body, které mají maximální a minimální souřadnici *x* a tyto body budou definovat přímku, která rozdělí vstupní body na dvě části
2. Postupně se hledají body, které jsou vůči segmentu konvexní obálky nejvzdálenější a zároveň leží v pravé polorovině (obrázek 3)
3. Výpočet probíhá do té doby, dokud existuje bod, který leží vpravo od segmentu

Obrázek 3: Tvorba konvexní obálky pomocí algoritmu Quick Hull

Obsah obrázku tabulka

Popis byl vytvořen automaticky

Zdroj: Hoang, Linh (2015)

# Popis algoritmů

Dle Bayer (2008) by měl algoritmus pro generalizaci budov splňovat tyto požadavky:

* Rozumná časová složitost (kvadratická nebo lepší) poskytující dostačující výsledky s minimem dodatečných manuálních oprav
* Schopnost detekce a generalizace budovy v jakékoliv poloze
* Odstranění vlastních průsečíků
* Schopnost zachovat plochu budovy
* Regulace faktoru generalizace uživatelem
* Schopnost generalizace složitých a nekonvexních polygonů

V následujících kapitolách budou blíže představeny algoritmy Minimum Area Enclosing Rectangle a Wall Average, které byly použity v aplikaci pro generalizaci budov, již je výstupem této práce.

## Minimum Area Enclosing Rectangle

Výsledkem algoritmu Minimum Area Enclosing Rectanle (MAER) je obdélník, který má minimální plochu a žádný bod neleží vně obdélníku. Zároveň alespoň jedna strana obdélníku je kolineární se stranou konvexní obálky. Algoritmus je založen na opakované rotaci – obdélník je natáčen o úhel -σ představující jednu ze směrnic strany konvexní obálky. V této poloze je vždy nalezen min-max box (MMB) a spočítá se jeho plocha. Otočením min-max boxu o úhel σ je min-max box převeden na obecný obdélník, ale velikost plochy zůstane stejná. Obdélník s nejmenší plochou bude výsledkem (Bayer 2023). Pro rotaci se využívá matice

**Bližší popis algoritmu**

1. Najdi konvexní obálku datasetu S (musí být splněna podmínka, že obsahuje alespoň 3 body)
2. Inicializuj MAER na MMB, zjisti souřadnice vrcholů a spočítej plochu: A = A(MMB(S))
3. **Projdi** hrany e konvexní obálky:

* Vypočítej směrnici σ hrany e
* Otoč S o -σ: Sr = R(-σ)S
* Najdi MMB rotované množiny Sr, spočítej vrcholy a plochu A
* **Když** je A < A:
  + Zapamatuj si celý MMB (plocha, vrcholy, směrnice): A = A, MMB = MMB, σ = σ

1. MAER se získá tak, že se původní množina S narotuje o úhel σmin, který odpovídá nejmenší ploše

## Wall Average

Algoritmus Wall Average je založený na principu, že na každou stranu budovy je aplikováno modulo π/2 a z této hodnoty je poté spočten vážený průměr, kde váhy představují délky stran. Tento algoritmus dává nejlepší výsledky (Bayer 2023).

**Bližší popis algoritmu**

1. Vyber libovolnou stranu mnohoúhelníku a spočítej její směrnici σ'
2. **Projdi** směrnice σi všech hran budovy

* Spočítej ∆σi = σi - σ' a natoč budovu o σ' → strany by měly být přibližně rovnoběžné s osami *x* a *y*

1. Pro každou hodnotu ∆σi spočítej ki podle vzorce ki = 2∆σi/π a proveď zaokrouhlení
2. Pro každou hranu spočítej zbytek po dělení podle vzorce: ri = ∆σi – kiπ/2
3. Spočítej směr natočení budovy pomocí váženého průměru:

# Data

Vstupem do vytvořené aplikace jsou polygonové vrstvy ve formátu *.shp*. Pro otestování byly použity tři soubory vstupních dat s různým typem zástavby. Prvním je dataset znázorňující historické centrum Prahy (katastrální území Staré město), pro které jsou charakteristické úzké, nepravoúhlé ulice a nepravidelná zástavba. Druhý dataset pokrývá katastrální území Vinohrady, po něhož je typická pravidelná zástavba a pravoúhlé budovy. Třetí dataset pokrývá území katastrální území Suchdol typické vilovou zástavbou.

# Dokumentace

Vzhled výsledné aplikace je ukázán na obrázku 4. V aplikaci je možné otevřít soubor (Open File), provést generalizaci metodou Minimum Area Enclosing Rectangle a Wall Average a smazat výsledky. V následujících kapitolách budou blíže představeny jednotlivé funkce, které dohromady tvoří aplikaci.

Obrázek 4: Náhled uživatelského rozhraní

Obsah obrázku text

Popis byl vytvořen automaticky

## GetPointPolygonPositionR

Uvedená metoda implementuje metodu pro určení polohy bodu vzhledem k polygonu. Metoda se nazývá *getPointPolygonPositionR* a přijímá dva argumenty: *q*, což je bodový objekt, a *pol*, což je seznam bodových objektů, které definují vrcholy polygonu. Metoda používá k určení polohy bodu algoritmus *ray casting*. Nejprve inicializuje dvě proměnné *k* a *n*, které představují počet průsečíků mezi vodorovným paprskem vycházejícím z bodu a hranami polygonu, respektive celkový počet vrcholů polygonu. Poté projdou smyčkou všechny vrcholy mnohoúhelníku a redukuje se jejich souřadnice vzhledem k zadanému bodu *q*. Pro každý vrchol také vypočítá souřadnice dalšího vrcholu polygonu, což je nutné k určení hrany, která se protíná s vodorovným paprskem. Pokud se hrana s paprskem protíná, vypočítá metoda průsečík bodu *xm* pomocí vzorce pro rovnici přímky. Pokud je tento průsečík napravo od bodu *q*, zvýší se proměnná *k*. Nakonec, je-li počet průsečíků *k* lichý, je bod *q* uvnitř mnohoúhelníku a metoda vrátí 1. V opačném případě je bod mimo polygon a metoda nevrací nic (obrázek 5).

Obrázek 5: Metoda GetPointPolygonPositionR

Obsah obrázku text

Popis byl vytvořen automaticky

## Get2LinesAngle

Metoda *get2LinesAngle* přijímá čtyři argumenty: *p1*, *p2*, *p3* a *p4*, což jsou bodové objekty reprezentující dvě přímky. Metoda nejprve vypočítá směrové vektory *u* a *v* pro obě přímky pomocí souřadnic daných bodů. Poté vypočítá bodový součin *u* a *v*, který se rovná součinu velikostí obou vektorů krát kosinus úhlu mezi nimi. Dále metoda vypočítá velikosti obou vektorů pomocí Pythagorovy věty. Poté vydělí bodový součin součinem magnitud a získá kosinus úhlu. Aby bylo zajištěno, že argument funkce arccos je v platném rozsahu [-1,1], používá metoda funkce *max* a *min* k ořezání hodnoty cosinu mezi -1 a 1. Nakonec metoda vrátí hodnotu arccos, která udává úhel mezi oběma přímkami v radiánech (obrázek 6).

Obrázek 6: Metoda Get2LinesAngle

Obsah obrázku text

Popis byl vytvořen automaticky

## CreateCH

Uvedená část je implementace metody pro vytvoření konvexní obálky mnohoúhelníku pomocí algoritmu Jarvisova skenování (obrázek 7). Metoda se nazývá *createCH* a přijímá jeden argument: *Pol*, což je objekt QPolygonF reprezentující vstupní polygon.

Metoda nejprve vytvoří prázdný objekt QPolygonF s názvem *ch*, do kterého uloží vrcholy konvexní obálky. Poté najde otočný bod *q* s nejmenší souřadnicí *y* ve vstupním polygonu. Inicializuje dva body *pj1* a *pj*, které představují poslední dva body přidané do konvexního trupu, přičemž *pj1* je vlevo od *q* a *pj* se rovná *q*.Metoda přidá ke konvexní obálce otočný bod *q*.

Poté metoda vstoupí do smyčky, která pokračuje, dokud opět nedosáhne otočného bodu *q*. V každé iteraci metoda inicializuje maximální úhel *phi\_max* na hodnotu 0 a index bodu s maximálním úhlem *i\_max* na hodnotu -1.Poté projde smyčkou všechny body vstupního polygonu a vypočítá úhel mezi úsečkou z aktuálního bodu do *pj* a úsečkou z *pj1* do *pj* pomocí metody *get2LinesAngle*.

Pokud je úhel větší než aktuální maximální úhel, metoda aktualizuje maximální úhel a index bodu s maximálním úhlem. Po prohledání všech bodů vstupního polygonu metoda připojí bod s maximálním úhlem ke konvexní obálky a aktualizuje poslední dva body přidané do obálky. Nakonec metoda zkontroluje, zda se poslední bod přidaný do obálky rovná otočnému bodu *q*. Pokud ano, přeruší smyčku a vrátí konvexní obálku. V opačném případě pokračuje ve smyčce, dokud nenajde otočný bod.

Obrázek 7: Metoda createCH

Obsah obrázku text

Popis byl vytvořen automatickyObsah obrázku text

Popis byl vytvořen automaticky

## Rotate

Metoda nejprve vytvoří prázdný objekt QPolygonF s názvem *pol\_rot*, do kterého se uloží vrcholy otočeného mnohoúhelníku. Poté projde všechny vrcholy vstupního mnohoúhelníku a pro každý vrchol vypočítá otočené souřadnice pomocí následujících vzorců:

x\_rot = x \* cos(sig) - y \* sin(sig)

y\_rot = x \* sin(sig) + y \* cos(sig)

kde *x* a *y* jsou souřadnice aktuálního vrcholu a *cos* a *sin* jsou funkce cosinus a sinus úhlu natočení *sig*. Metoda pak vytvoří objekt QPointF reprezentující otočený vrchol pomocí vypočtených souřadnic a připojí jej k otočenému polygonu *pol\_rot* (obrázek 8). Nakonec metoda vrátí otočený polygon *pol\_rot*.

Obrázek 8: Metoda rotate

Obsah obrázku text

Popis byl vytvořen automaticky

## MinMaxBox

Funkce minMaxBox přijme jako vstup polygon *pol*, zjistí minimální a maximální souřadnice polygonu pro vytvoření obdélníku a vrátí obdélník jako objekt QPolygonF spolu s jeho plochou. Funkce nejprve zjistí minimální a maximální souřadnice *x* a *y* polygonu pomocí funkcí *min* a *max* a funkce *lambda*, která vrací souřadnici *x* nebo *y* bodu (obrázek 9).

Poté vytvoří čtyři objekty QPointF, které představují vrcholy obdélníku. Vrcholy jsou vytvořeny v určitém pořadí (v1, v2, v3, v4), aby byl obdélník vytvořen se správnou orientací. Nakonec funkce vrátí objekt QPolygonF reprezentující obdélník a jeho plochu, která je jednoduše součinem šířky a výšky obdélníku.

Obrázek 9: Metoda minMaxBox

Obsah obrázku text

Popis byl vytvořen automaticky

## MinAreaEnclosingRectangle

Funkce nejprve vytvoří konvexní obálku mnohoúhelníku pomocí funkce *createCH*. Poté pomocí funkce *minMaxBox* najde obdélník s minimální plochou ("minmax box") konvexního trupu. Tím se získá počáteční odhad obdélníku minimální plochy polygonu. Funkce poté otočí konvexní obálku o malý úhel (vypočtený jako úhel mezi sousedními vrcholy konvexního obalu) a zjistí minimální plochu uzavírajícího obdélníku otočeného konvexního obalu.

Pokud je plocha nového obdélníku s minimální plochou menší než předchozí odhad, funkce odhad aktualizuje a proces opakuje s otočeným konvexním trupem. Po zvážení všech rotací funkce vrátí obdélník s minimální plochou uzavírající původní polygon otočením minmax boxu o úhel, který vedl k nejmenší ploše, a změnou velikosti výsledného obdélníku tak, aby se vešel do polygonu pomocí funkce resizeRectangle (obrázek 10).

Obrázek 10: Metoda minAreaEnclosingRectangle

Obsah obrázku text

Popis byl vytvořen automatickyObsah obrázku text

Popis byl vytvořen automaticky

## ComputeArea

Funkce *computeArea* vypočítá plochu mnohoúhelníku pomocí vzorce pro tkaničku. Vzorec funguje tak, že rozdělí mnohoúhelník na trojúhelníky a sečte jejich plochy. Funkce iteruje přes všechny vrcholy mnohoúhelníku a vypočítá plochu každého trojúhelníku pomocí křížového součinu dvou hran, které mají společný vrchol. Nakonec funkce sečte plochy všech trojúhelníků a vrátí polovinu absolutní hodnoty výsledku (obrázek 11).

Obrázek 11: Metoda computeArea

Obsah obrázku text

Popis byl vytvořen automaticky

## ResizeRectangle

Tato funkce vezme ohraničující obdélník konvexní obálky a změní jeho velikost tak, aby se vešel do původního mnohoúhelníku. Proměnná k je měřítkový faktor, který zajišťuje, že plocha zmenšeného obdélníku A je rovna zlomku plochy původního mnohoúhelníku Ab.

Funkce začíná výpočtem těžiště obdélníku pomocí průměrných souřadnic jeho vrcholů (obrázek 12). Poté vypočítá čtyři vektory, které vedou z hmotového středu do každého vrcholu obdélníku. Tyto vektory jsou poté škálovány koeficientem sqrt(k) a jejich nové souřadnice jsou použity k vytvoření nových vrcholů obdélníku změněné velikosti.

Nakonec se pomocí nových vrcholů vytvoří nový mnohoúhelník, který se vrátí jako zmenšený obdélník.

Obrázek 12: Metoda resizeRectangle

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

## WallAverage

Funkce wallAverage přebírá objekt QPolygonF představující půdorys budovy a vrací objekt QPolygonF představující obdélník s minimální plochou, který obklopuje budovu a který byl získán pomocí metody průměrování stěn. Metoda průměrování stěn spočívá ve výpočtu průměrného směru hran budovy tak, že se vypočítá průměr zbytků úhlů mezi každou hranou a referenční hranou a poté se budova otočí o průměrný směr. Poté se najde minimální plocha ohraničujícího obdélníku výpočtem minimální plochy ohraničujícího obdélníku pootočené budovy, a nakonec se velikost obdélníku změní tak, aby odpovídal původní budově pomocí funkce *resizeRectangle*.

Proměnná *n* je počet vrcholů budovy. Proměnná *sigma* je směr první hrany budovy, který se používá jako referenční směr. Smyčka iteruje přes všechny hrany budovy a vypočítá směr každé hrany vzhledem k referenčnímu směru. Proměnná *delta\_sigma\_i* je rozdíl mezi směrem hrany *i* a referenčním směrem. Pokud je rozdíl záporný, opraví se přičtením 2\**pi*. Proměnná *ki* je celočíselná část 2\**delta\_sigma\_i/pi*, což je počet otáček *pi*/2 potřebných k vyrovnání hrany s referenčním směrem. Proměnná *r\_i* je zbytek úhlu po vyrovnání hrany s referenčním směrem, který se použije k výpočtu průměrného směru hrany budovy. Nakonec funkce otočí budovu o průměrný směr, vypočítá minimální plochu ohraničujícího obdélníku, otočí obdélník zpět do původní orientace a změní jeho velikost tak, aby se vešel do původní budovy. Metoda je ukázána na obrázku 13.

Obrázek 13: Metoda wallAverage

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

# Výsledky

# Závěr

Program by měl být schopný načíst data ve formátu .*shp* a provést generalizaci mnohoúhelníků dvěma vybranými metodami – Minimum Area Enclosing Rectangle a Wall Average. Program je však limitovaný jak samotnými metodami, které nejsou vhodné pro všechny typy zástavby, tak zkušenostmi autorek.

# Použité zdroje

BAYER, T. (2008a): Algoritmy v digitální kartografii. Univerzita Karlova v Praze, Nakladatelství Karolinum, Praha, 251 s.

BAYER, T. (2008b): The importance of computational geometry for digital cartography. Geoinformatics FCE CTU, 3, 15–24.

BAYER, T. (2023): Konvexní obálka množiny bodů. Výukový materiál. Dostupné z: <http://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4_new.pdf> (cit. 29.3.2023).

GEEKSFORGEEKS (2022): Convex Hull using Jarvis’ Algorithm or Wrapping. Dostupné z: <https://www.geeksforgeeks.org/convex-hull-using-jarvis-algorithm-or-wrapping/> (cit. 28.3.2023).

GEEKSFORGEESKS (2023): Convex Hull using Graham Scan. Dostupné z: <https://www.geeksforgeeks.org/convex-hull-using-graham-scan/> (cit. 28.3.2023).

HOANG, N. D., LINH, N. K. (2015): Quicker than Quickhull. Vietnam Journal of Mathematics, 43, 57–70.