České vysoké učení technické v Praze

Fakulta stavební



Algoritmy v digitální kartografii

Úloha č. 1: Generalizace budov

Skupina:

Sabina Kličková

Martin Vajner

Zimní semestr 2021/2022

Obsah

[1. Zadání 3](#_Toc87013117)

[2. Bonusové úlohy 3](#_Toc87013118)

[3. Popis a rozbor problémů + vzorce. 3](#_Toc87013119)

[4. Popisy algoritmů 4](#_Toc87013120)

[5. Problematické situace a popsání bonusových úloh 6](#_Toc87013121)

[6. Vstupní data, formát vstupních dat, popis. 6](#_Toc87013122)

[7. Výstupní data, formát výstupních dat, popis 7](#_Toc87013123)

[9. Závěr, možné či neřešené problémy, náměty na vylepšení 8](#_Toc87013124)

[II. Citovaná literatura 8](#_Toc87013125)

[III. Seznam obrázků 8](#_Toc87013126)

# Zadání

Vstup:

Výstup:

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 orientovaným v obou hlavních směrech, jeho plocha bude stejná jako plocha budovy. Výsledky generalizace vhodně vizualizujte.

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

|  |  |
| --- | --- |
| Generalizace budov metodami Minimum Area Enclosing Rectangle a Wall Average | 15b |

# Bonusové úlohy

V této úloze byly zpracovány následující bonusové úlohy:

|  |  |
| --- | --- |
| Krok | Hodnocení |
| Generalizace budov metodou Longest Edge | +5b |
| Generalizace budov metodou Weighted Bisector | +8b |

1. Popis a rozbor problémů + vzorce.

Definice konvexní obálky: Konvexní obálka konečné bodové množiny S je nejmenší konvexní mnohoúhelník P, který obsahuje množinu S.

Konvexní obálka je hranice spojující body množiny tak, aby každý bod množiny ležel uvnitř nebo na hranici obálky.



Obrázek 1: Ukázka konvexní obálky nad budovou

Množina S je konvexní, leží-li spojnice libovolných dvou prvků uvnitř této množiny.

1. Popisy algoritmů
2. Metody konstrukce konvexní obálky
   1. **Jarvis scan**

Tato metoda je používána pro vyhledávání bodů konvexní obálky podle největšího úhlu. Nevýhodou je, že tři body nesmí ležet na přímce. Tato skutečnost musí být tedy ošetřena.

U tohoto algoritmu se začíná nalezením pivota q. Pivot je bod s nejmenší souřadnicí y. Tento bod je pak počátečním bodem konvexní obálky. Poté se vytvoří bod se souřadnicemi (0, qy).

Následuje cyklus hledání bodu takového, jehož spojnice s posledním bodem konvexní obálky svírá s poslední hranou konvexní obálky maximální úhel. Po nalezení takového bodu je bod přidán do konvexní obálky. (1)

Algoritmus:

1. Nalezení pivota *q*, *q* = min(*yi*)

2. Přidání *q* do konvexní obálky *H*

3. Inicializace 𝑝𝑗−1∈𝑋 , 𝑝𝑗=𝑞 , 𝑝𝑗+1=𝑝𝑗−1

4. Dokud 𝑝𝑗+1≠𝑞

• Nalezení bodu s maximálním úhlem 𝑝𝑗+1

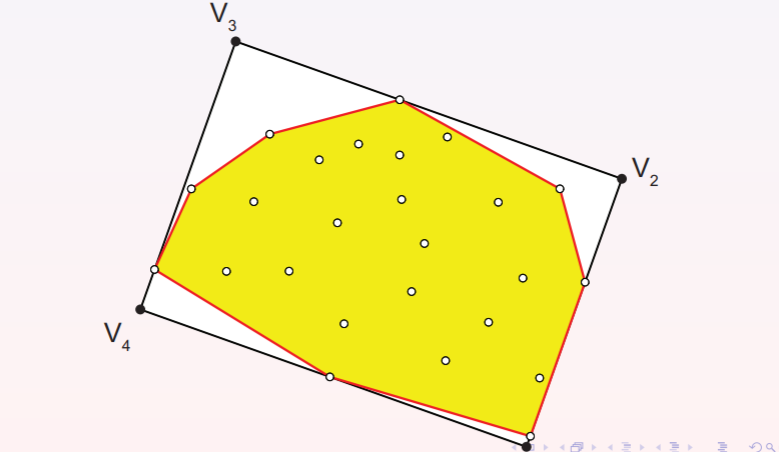
• Přidání bodu 𝑝𝑗+1 do konvexní obálky

• Změna indexu, 𝑝𝑗-1=𝑝𝑗; 𝑝𝑗= 𝑝𝑗+1

* 1. **Minimum Area Enclosing Rectangle**

Metoda je založena na principu minimální obdélníkové plochy obklopující objekt či soubor bodů. Jejím základem je vytvoření konvexní obálky. Poté se vytvoří prvotní ohraničující obdélník bez natočení. Poté se hledá obdélník takový, který má nejmenší plochu. Hledání probíhá pomocí natáčení minimálního oblélníku vytvořeného pod úhlem sigma. Úhel sigma je počítán vždy ze dvou po sobě jdoucích bodů.

Algoritmus dává dobré výsledky, problémy s budovami tvaru L a Z. (1)



1. **Detekce úhlu natočení budov**

*Budova před generalizací a po generalizaci musí mít uchovánu orientaci vzhledem k ostatním obsahovým prvkům mapy (např. zachování uliční čáry). Nutnost detekovat tzv. hlavní směry budovy (jsou na sebe zpravidla kolmé). Popisují orientaci (tj. natočení) budovy vzhledem k ostatním prvkům mapy.* (1)

* 1. **Wall Average**

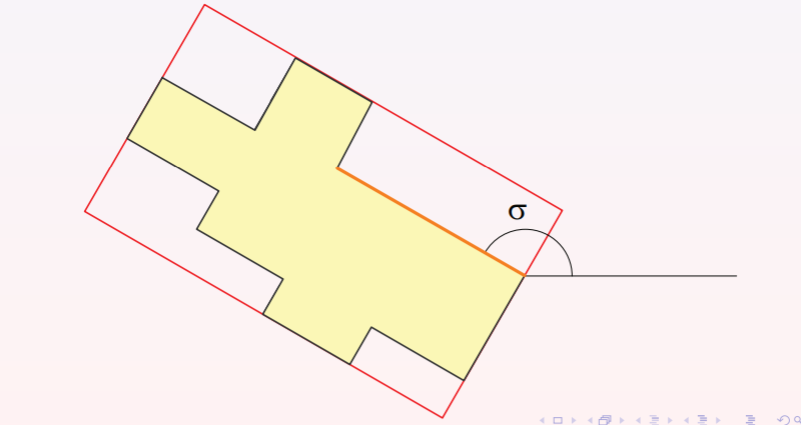
*Na každou stranu budovy aplikována operace mod( π/2 ). Ze „zbytků” hodnot spočten vážený průměr, váhou je délka strany. Nejkomplexnější metoda, citlivá na “nepravé” úhly. Nejprve určeny směrnice σi všech hran. Poté redukce σi o úhel σ ′ (natočení budovy || s osami x, y).* (1)

* 1. **Longest Edge**

*První hlavní směr budovy představován nejdelší stranou v budově, druhý hlavní směr na ní kolmý. Nedosahuje příliš dobrých výsledků. Nejdelší strana nemusí reprezentovat hlavní směr.* (1)

Principem metody je nalezení nejdelší hrany budovy. Spočítají se tedy postupně souřadnicové rozdíly všech po sobě jdoucích bodů a určí se z nich vzdálenosti. Pokud je spočtená vzdálenost větší než doposud největší nalezená, přiřadí se do hodnoty největší vzdálenosti. Současně s ní se uloží i její směrník sigma.

Poté se nad body vytvoří nejmenší ohraničující obdélník, jehož hlavní strana má orientaci sigma.

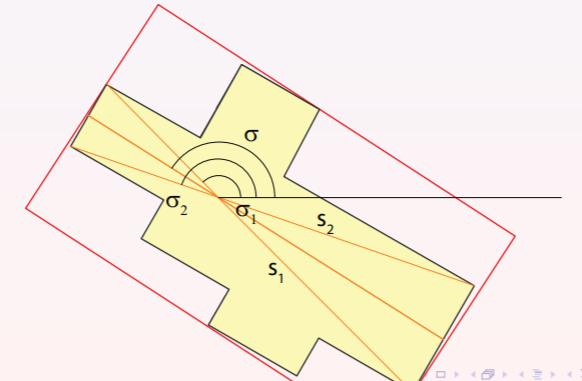


* 1. **Weighted Bisector**

*Hledány dvě nejdelší úhlopříčky, směrnice σ1, σ2, délky s1, s2. Hlavní směr dán váženým průměrem . Dává velmi dobré výsledky.* (1)

Principem metody je nalezení nejdelší úhlopříčky budovy. Spočítají se tedy postupně souřadnicové rozdíly všech bodů a určí se z nich vzdálenosti. Pokud je spočtená vzdálenost větší než doposud největší nalezená, přiřadí se do vektoru vzdáleností a současně s ní se přiřadí i směrník do vektoru směrníků sigma. Z těchto vektorů se poté použijí dvě poslední přidané hodnoty. Hodnota výsledného hlavního směru je dána váženým průměrem, viz. výše.

Poté se nad body vytvoří nejmenší ohraničující obdélník, jehož hlavní strana má orientaci sigma dle vzorce.



1. Problematické situace a popsání bonusových úloh

Weighted average

U této metody může nastat problém při počítání váženého průměru hlavního směru. Pokud jsou totiž dvě nejdelší vzdálenosti mezi body a jejich směry nevhodně orientovány, může dojít ke špatnému zvolení hlavního směru. Tento směr je pak na správný hlavní směr kolmý. Při odečítání směrníků je tedy třeba dbát na to, aby jejich rozdíl nikdy nepřesáhl 90° nebo - 90°. Pokud je tato podmínka nesplněna, je třeba k jednomu ze směrníků přičíst 180°. Tím je zaručeno, ze vždy bude zvolen ten správný hlavní směr.

if (sigma1<0)

{sigma1=sigma1+M\_PI;}

if (sigma2<0)

{sigma2=sigma2+M\_PI;}

if ((sigma2-sigma1) > (M\_PI/2) || (sigma2-sigma1 < (-M\_PI/2)) )

{sigma2= sigma2-M\_PI;}

1. Vstupní data, formát vstupních dat, popis.

Vstupní data byla přejata z [www.geoportalpraha.cz](https://www.geoportalpraha.cz/). Následně byla v ArcGis Pro oříznuta a převedena z polygonů na bodovou vrstvu. Ta byla následně exportována do formátu ASCII(.Txt).

Formát dat textového souboru:

x << y << id << fid

kde: x - souřadnice x bodu

y – souřadnice y bodu

id – pořadové číslo bodu

fid – číslo původního polygonu

1. Výstupní data, formát výstupních dat, popis

Výstupem je aplikace která dokáže nad vhodně zpracovanými daty vytvořit konvexní obálku pomocí několika metod.

Výstupem a výsledkem je tedy grafické okno, které obsahuje ovládací prvky aplikace a okno pro grafické zobrazení dat.

Ovládacími prvky jsou combobox pro výběr algoritmu,tlačitko pro načtení dat, tlačítko vytvoření generalizace polygonů a tlačítko clear, které smaže okno.

1. Dokumentaci: popis tříd, datových položek a jednotlivých metod

class **Algorithms**

public:

**Algorithms**();

double **get2LinesAngle**(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);

* zjištění hodnoty úhlu mezi dvěma hranami polygonu
* vstup: souřadnice bodů vektorů

QPolygon **cHull** (std::vector <QPoint> &points);

std::vector <QPoint> **rotate**(std::vector <QPoint> &points, double sigma);

std::tuple<std::vector<QPoint>, double> **minMaxBox**(std::vector <QPoint> &points);

QPolygon **minAreaEnclosingRectangle**(std::vector <QPoint> &points);

QPolygon **wallAverage**(std::vector <QPoint> &points);

double **LH**(std::vector <QPoint> &points);

std::vector <QPoint> **resizeRectangle**(std::vector <QPoint> &points, std::vector <QPoint> &er);

QPolygon **longestEdge**(std::vector <QPoint> &points);

QPolygon **weightedBisector**(std::vector <QPoint> &points);

class **Draw** : public QWidget

private:

std::vector<QPoint> points;

* definice proměnné uložení jednotlivých bodů

QPolygon ch, er;

* definice proměnné pro uložení konvexní obálky a generalizovaných objektů

std::vector<QPolygon> ch\_v, er\_v;

* definice proměnné pro uložení vícero konvexních obálek a generalizovaných objektů

std::vector<QPolygon> pol;

* definice proměnné pro načtení polygonu

public:

explicit **Draw**(QWidget \*parent = nullptr);

void ***paintEvent***(QPaintEvent \*event);

* vykreslení polygonu a jeho grafické zvýraznění

void **clear**();

* mazaní vykreslených proměnných

std::vector<QPoint> **getPoints**(){return points;}

* vrací souřadnice bodů

void **setCh**(QPolygon &ch\_){ch = ch\_;}

* přiřazuje konvexní obálku

void **setEr**(QPolygon &er\_){er = er\_;}

* přiřazuje generalizovaný objekt

void **loadFile**(std::string &path);

* načtení dat formátu ASCII(.txt)

std::vector<QPolygon> **getPolygon**(){return pol;};

* vrací souřadnice vykreslených polygonů

void **addCh**(QPolygon &ch\_){ch\_v.push\_back(ch\_);}

* připojí konvexní obálku do seznamu

void addEr(QPolygon &er\_){er\_v.push\_back(er\_);}

* připojí generalizaci do seznamu

1. Závěr, možné či neřešené problémy, náměty na vylepšení

# Citovaná literatura

1. **Tomáš, Bayer.** Perslonal page of Bayer Tomas. *Charles University of Prague.* [Online] [Citace: 17. 10 2021.] https://web.natur.cuni.cz/~bayertom/index.php/teaching/algoritmy-v-digitalni-kartografii.

2. **Topiwala, Anirudh.** *towards data science.* [Online] 2020. [Citace: 17. 10 2021.] https://towardsdatascience.com/is-the-point-inside-the-polygon-574b86472119.

# Seznam obrázků

[Obrázek 1: Princip Winding Number algoritmu 4](file:///C:\Users\Sabča\Downloads\Úloha_1%20(1).docx#_Toc86750476)

[Obrázek 2: Princip upraveného modelu Ray Crossing (2) 5](#_Toc86750477)

[Obrázek 3: Okno aplikace po spuštění 7](#_Toc86750478)

[Obrázek 4: Okno aplikace - analýza polohy bodu 7](#_Toc86750479)

V Praze dne 2.11.2021