České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky

Technická zpráva

Algoritmy v digitální kartografii

Úloha č. 3: Digitální model terénu

Bc. Pane Kuzmanov
Bc. František Mužík

Studijní program: Geodézie a kartografie
Specializace: Geomatika

Praha 2021

Úloha č. 3: Digitální model terénu

Vstup: množina $P = \{p_1, ..., p_n\}$, $p_i = \{x_i, y_i, z_i\}$.

Výstup: polyedrický DMT nad množinou P *představovaný vrstevnicemi doplněný vizualizací sklonu trojúhelníků a jejich expozicí.*

Metodou inkrementální konstrukce vytvořte nad množinou P vstupních bodů 2D Delaunay triangulaci. Jako vstupní data použijte existující geodetická data (alespoň 300 bodů) popř. navrhněte algoritmus pro generování syntetických vstupních dat představujících významné terénní tvary (kupa, údolí, spočinek, hřbet, ...).

Vstupní množiny bodů včetně níže uvedených výstupů vhodně vizualizujte. Grafické rozhraní realizujte s využitím frameworku QT. Dynamické datové struktury implementujte s využitím STL.

Nad takto vzniklou triangulací vygenerujte polyedrický digitální model terénu. Dále proveďte tyto analýzy:

- S využitím lineární interpolace vygenerujte vrstevnice se *zadaným krokem* a v *zadaném intervalu*, proveďte jejich vizualizaci s rozlišením zvýrazněných vrstevnic.
- Analyzujte sklon digitálního modelu terénu, jednotlivé trojúhelníky vizualizujte v závislosti na jejich sklonu.
- Analyzujte expozici digitálního modelu terénu, jednotlivé trojúhelníky vizualizujte v závislosti na jejich expozici ke světové straně.

Zhodnoťte výsledný digitální model terénu z kartografického hlediska, zamyslete se nad slabinami algoritmu založeného na 2D Delaunay triangulaci. Ve kterých situacích (různé terénní tvary) nebude dávat vhodné výsledky? Tyto situace graficky znázorněte.

Zhodnocení činnosti algoritmu včetně ukázek proveďte alespoň na **3 strany** formátu A4.

Hodnocení:

Krok	**Hodnocení**
Delaunay triangulace, polyedrický model terénu.	10b
Konstrukce vrstevnic, analýza sklonu a expozice.	10b
Triangulace nekonvexní oblasti zadané polygonem.	*+5b*
Výběr barevných stupnic při vizualizaci sklonu a expozice.	*+3b*
Automatický popis vrstevnic.	*+3b*
Automatický popis vrstevnic respektující kartografické zásady (orientace, vhodné rozložení).	*+10b*
Algoritmus pro automatické generování terénních tvarů (kupa, údolí, spočinek, hřbet, ...).	*+10b*
3D vizualizace terénu s využitím promítání.	*+10b*
Barevná hypsometrie.	*+5b*
Max celkem:	**65b**

Čas zpracování: 4 týdny

1 Údaje o bonusových úlohách

2 Popis a rozbor problému

3 Popisy algoritmů formálním jazykem

4 Problematické situace a jejich rozbor

5 Vstupní data

Vstupem je textový soubor s prostorovými souřadnicemi X,Y,Z geodetických bodů. Tyto body jsou považovány za vstupní množinu P.

6 Výstupní data

Za výstup je považována grafický výstup vytvořené aplikace. Grafickým výstupem je polyedrický DMT nad množinou P představovaný vrstevnicemi doplněný vizualizací sklonu trojúhelníků a jejich expozicí.

7 Snímky obrazovky vytvořené aplikace a její popis

8 Dokumentace

8.1 Třída Algorithms

Tato třída obsahuje výpočetní vzorce pro použité algoritmy.
Třída algorithms obsahuje následující veřejné metody:

```
double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4)
```

Metoda vypočte velikost úhlu, který svírají dvě přímky. První přímku tvoří body p_1, p_2 a druhou přímku body p_3, p_4.

```
int getPointLinePosition(QPoint &a,QPoint &p1,QPoint &p2)
```

Metoda určuje, zda – li bod leží v levé či v pravé polorovině od přímky (hrany polygonu). Vstupními parametry jsou určovaný bod a a body p_1, p_2, které tvoří přímku.

```
double getPointLineDistance(QPoint &a, QPoint &p1, QPoint &p2)
```

Metoda určuje vzdálenost bodu a od přímky tvořenou body p_1, p_2.

```
QPolygon cHullJarvisScan(std::vector <QPoint> &points)
```

Výpočet konvexní obálky načtených bodů na základě algoritmu Jarvis Scan (kapitola ??).

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

Otočení načtených bodů o zadaný úhel *sigma*.

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

Proměnná vracející min – max box, tedy vrcholy obdélníka a jeho výměru. Vstupem jsou načtené body.

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

Určení hlavního směru polygonu s využitím algoritmu Minimum Area Eclosing Rectangle (kapitola ??). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

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

Určení hlavního směru polygonu s využitím algoritmu Wall Average (kapitola ??). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

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

Výpočet plochy obecného mnohoúhelníků pomocí LH vzorce.

`std::vector<QPoint> resizeRectangle(std::vector<QPoint> &points, std::vector<QPoint`

Přepočet velikosti ohraničujícího obdélníku na základě výměry původního polygonu.

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

Určení hlavního směru polygonu s využitím algoritmu Longest Edge (kapitola ??). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

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

Určení hlavního směru polygonu s využitím algoritmu Weighted Bisector (kapitola ??). Na vstupu jsou body polygonu. Výstupem je generalizovaný a pootočený polygon.

`QPolygon cHullQuickHull(std::vector <QPoint> &points)`

Výpočet konvexní obálky načtených bodů na základě algoritmu Quick Hull (kapitola ??).

`void quickHullLocal(int ps, int pe, std::vector<QPoint> &points, QPolygon &ch)`

Metoda pro pomocný výpočet rekurentního určení konvexní obálky bodů algoritmem Quick Hull, lokální procedura (kapitola ??). Vstupními parametry jsou indexy počátečního a koncového bodu *ps* a *pe* vstupní hrany, načtené body a konvexní obálka.

8.2 Třída Draw

Tato třída umožňuje vykreslování bodu a polygonů.

Třída draw obsahuje následující privátní metody a proměnné:

```
std::vector<QPoint> points
```

Proměnná se souřadnicemi načtených bodů.

```
QPolygon ch, er;
```

Polygony pro ukládání konvexní obálky a ohraničujícího obdélníku.

```
std::vector<QPolygon> polygons, chs, ers
```

Proměnná uchovávající pomocné polygony v průběhu výpočtu a pro vykreslení.

Třída draw obsahuje následující veřejné metody a proměnné:

```
explicit Draw(QWidget *parent = nullptr)
```

Prvotní vykreslení bodu mimo okno aplikace.

```
void paintEvent(QPaintEvent *event)
```

Metoda, která vykresluje bod či polygony.

```
void mousePressEvent(QMouseEvent *event)
```

Metoda určující souřadnice určeného bodu.

```
void clearAddedData()
```

Metoda mazající přidaná data z obrazovky.

```
void clearDrawing()
```

Metoda mazající nakreslený polygon z obrazovky.

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

Vrací souřadnice lomových bodů polygonu.

```
std::vector<QPolygon> getPolygons(){return polygons;}
```

Vrací souřadnice polygonů.

`void setCh(QPolygon &ch_) {chs.push_back(ch_);}`

Nastavení polygonu konvexní obálky.

`void setEr(QPolygon &er_) {ers.push_back(er_);}`

Nastavení polygonu ohraničujícího obdélníku.

`void clearChs(){chs.clear();}`

Smazání polygonu konvexní obálky.

`void clearErs(){ers.clear();}`

Smazání polygonu ohraničujícího obdélníku.

`void drawPolygons(std::vector<QPolygon> &data);`

Vykreslení polygonů načtených z textového souboru.

8.3 Třída Load

Třída draw obsahuje následující veřejnou proměnnou:

`static std::vector<QPolygon> load_file(std::string &filename)`

Umožňuje načítání polygonů z textového souboru.

8.4 Třída SortByX

Třída draw obsahuje následující veřejnou proměnnou:

`bool operator() (QPoint &p1, QPoint &p2)`

Ražení bodů dle jejich x – ové souřadnice.

8.5 Třída SortByY

Třída draw obsahuje následující veřejnou proměnnou:

`bool operator() (QPoint &p1, QPoint &p2)`

Ražení bodů dle jejich y – ové souřadnice.

8.6 Třída Widget

Tato třída propojuje uživatelské rozhraní aplikace s kódem. Je vytvořena v sekci *Design*.

Třída widget obsahuje následující privátní metody a proměnné:

`void on_pushButtonClear_clicked()`

Vymazání kresby. Propojení s tlačítkem *Clear drawing*.

`void on_pushButtonLoad_clicked()`

Načtení lomových bodů polygonů z textového souboru. Propojení s tlačítkem *Load file with polygon*.

`void on_pushButton_clicked()`

Spuštění funkce Building Simplify s vybraným algoritmem. Propojení s tlačítkem *Building Simplify*.

`void on_pushButtonClearData_clicked()`

Vymazání přidaných dat. Propojení s tlačítkem *Clear added data*.

`void processPoints(std::vector<QPoint> &points)`

Volba použitého algoritmu na základě výběru z combo boxů.

9 Závěr

9.1 Možné či neřešené problémy

9.2 Náměty na vylepšení

V Praze 30.11.2021

Bc. Pane Kuzmanov
Bc. František Mužík

Použitá literatura