

České vysoké učení technické v Praze
Fakulta stavební

155ADKG: Digitální model terénu

Michael Kala
Anna Zemánková

1 Zadání

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

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. navrhnete 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 tři 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: 3 týdny

2 Údaje o bonusových úlohách

3 Popis a rozbor problému

Mějme množinu bodů $P\{p_i\}$, $p_i = \{x_i, y_i, z_i\}$. Nad touto množinou chceme vytvořit síť trojúhelníků t_j pomocí Delaunay triangulace DT , následně vytvořit vrstevnice a pro vizualizaci DMT určit sklon a expozici jednotlivých trojúhelníků.

3.1 Delaunay triangulace

Vlastnosti:

- Uvnitř kružnice opsané trojúhelníku $t_j \in DT$ neleží žádný jiný bod množiny P .
- DT maximalizuje minimální úhel v $\forall t_j$, avšak DT neminimalizuje maximální úhel v t_j .
- DT je lokálně optimální i globálně optimální vůči kritériu minimálního úhlu.
- DT je jednoznačná, pokud žádné čtyři body neleží na kružnici.

3.2 Vrstevnice

Vrstevnice byly určeny za využití lineární interpolace, při které se předpokládá, že spád terénu je mezi podrobnými body p_i , mezi nimiž se provádí interpolace, konstantní.

Mějme trojúhelník t_j , tvořený hranami e_1, e_2, e_3 a rovinu vrstevnice ρ o dané výšce. Vztah hrany trojúhelníku a roviny vrstevnice:

1. $(z - z_i) * (z - z_{i+1}) < 0 \longrightarrow e_i \cap \rho$
2. $(z - z_i) * (z - z_{i+1}) > 0 \longrightarrow e_i \notin \rho$
3. $(z - z_i) * (z - z_{i+1}) = 0 \longrightarrow e_i \in \rho$

Pokud $e_1, e_2, e_3 \in \rho$, jedná se o trojúhelník náležící rovině ρ a není nutné vrstevnici pro tento trojúhelník řešit.

Jestliže $e_i \cap \rho$, je vypočten průsečík hrany $e_i = (p_1, p_2)$ a roviny vrstevnice ρ o výšce z :

$$x = \frac{(x_2 - x_1)}{(z_2 - z_1)}(z - z_1) + x_1,$$
$$y = \frac{(y_2 - y_1)}{(z_2 - z_1)}(z - z_1) + y_1.$$

3.3 Sklon

Sklon je úhel φ mezi svislicí n a normálou trojúhelníku n_t . Rovina trojúhelníku t_j je určena vektory u, v .

$$n = (0, 0, 1)$$

$$n_t = \vec{u} \times \vec{v}$$

$$\varphi = \arccos\left(\frac{n_t n}{|n_t| |n|}\right)$$

3.4 Expozice

Expozice je orientace trojúhelníku vůči světovým stranám.

$$A = \arctan 2\left(\frac{n_x}{n_y}\right);$$

kde n_x, n_y jsou vektorové součiny u a v .

4 Popis algoritmů

4.1 Delaunayova triangulace

Triangulace byla realizována metodou inkrementální konstrukce, body jsou tedy do triangulace přidávány postupně a to tak, aby vybraný bod ležel v levé polorovině od orientované hrany, poloměr opsané kružnice byl minimální a zároveň jsou preferovány body, jejichž střed opsané kružnice leží v pravé polorovině. Pokud žádný bod těmito kritériím nevyhovuje, je orientace hrany obrácena a bod je vybírán znovu. Jakmile je bod nalezen, jsou k němu vytvořeny orientované hrany a vše je uloženo do triangulace.

Pro "manipulaci" s hranami se používá struktura Active Edges List *AEL*, do ní jsou ukládány hrany, ke kterým je třeba nalézt třetí bod a vytvořit trojúhelník. Jakmile je *AEL* prázdná, algoritmus končí.

4.1.1 Implementace metody

1. $p_1 = rand(P), ||p_2 - p_1|| = min \dots$ náhodný a nejbližší bod
2. Vytvoř hranu $e = (p_1, p_2)$
3. Inicializuj: $p_{min} = argmin_{\forall p_i \in \sigma_L(e)} r'(k_i), k_i = (a, b, p_i), e = (a, b)$
4. Pokud $\nexists p_{min}$, prohod' orientaci $e \leftarrow (b, a)$. Jdi na 3)
5. $e_2 = (p_1, p_{min}), e_3 = (p_{min}, p_1) \dots$ zbývající hrany trojúhelníku
6. $AEL \leftarrow e, AEL \leftarrow e_2, AEL \leftarrow e_3$
7. $DT \leftarrow e, DT \leftarrow e_2, DT \leftarrow e_3$
8. while *AEL* not empty:
9. $AEL \rightarrow e, e = (p_1, p_2) \dots$ vezme první hranu z *AEL*
10. $e = (p_2, p_1) \dots$ prohodí její orientaci
11. $p_{min} = argmin_{\forall p_i \in \sigma_L(e)} r'(k_i), k_i = (a, b, p_i), e = (a, b)$
12. if $\exists p_{min}$:
13. $e_2 = (p_1, p_{min}), e_2 = (p_{min}, p_1) \dots$ zbývající hrany trojúhelníku
14. $DT \leftarrow e$
15. $add(e_2, AEL, DT), add(e_3, AEL, DT)$

Dílčí algoritmus Add:

1. Vytvoř hranu $e' = (b, a)$
2. if $(e' \in AEL)$
3. $AEL \rightarrow e' \dots$ Odstraň z AEL
4. else:
5. $AEL \leftarrow e \dots$ Přidej do AEL
6. $DT \leftarrow (a, b)$ Přidej do DT

4.1.2 Problematické situace

Pokud je vstupními daty grid, dochází k nejednoznačnosti Delaunay triangulace - na opsané kružnici leží čtyři body. V tom případě nefunguje výpočet vrstevnic ani následných analýz.

5 Vstupní data

Vstupními daty je množina bodů, kterou lze zadat dvěma způsoby:

1. klikáním na kanvas, výška se generuje náhodně
2. nahráním textového souboru se souřadnicemi bodů $[X,Y,Z]$ *.txt pomocí tlačítka *Load points*

Součástí příloh je textový soubor s testovacími daty *testovací data.txt*.

6 Výstupní data

Výsledky aplikace jsou vizualizovány v kanvasu grafického rozhraní.

7 Ukázka vytvořené aplikace

Nejprve je třeba zadat vstupní data - viz kapitolu Vstupní data a následně je možné vypočítat Delaunay triangulaci, teprve poté jsou zpřístupněna i ostatní tlačítka aplikace.

Při výpočtu vrstevnic je třeba zadat krok vrstevnic a (ne)zaškrtnout zvýraznění hlavních vrstevnic.

V sekci analysis je možné vypočítat a vizualizovat sklon a expozici.

Vše je uvedeno do původního stavu (smazány body i vypočtené výsledky) kliknutím na tlačítko Clear.

7.1 Algorithms

V třídě Algorithms jsou staticky implementovány algoritmy počítající kovexní obálku a minimální ohraničující obdélník (včetně vodící linie).

- Výčtový typ **TPosition**

- Typ využitý jako návratová hodnota členské metody **getPointLinePosition**.
- **LEFT = 0**
- **RIGHT = 1**
- **ON = 2**

- Metoda **getPointLinePosition**

- Tato metoda slouží k určení polohy bodu vůči přímce. Návratovou hodnotou je výčtový typ **TPosition**.
- Vstup
 - * **QPointF &q** - určovaný bod
 - * **QPointF &a, &b** - body přímky
- Výstup
 - * **LEFT** - bod vlevo od přímky
 - * **RIGHT** - bod vpravo od přímky
 - * **ON** - bod na přímce

- Metoda **getTwoVectorsAngle**

- Tato metoda slouží k určení úhlu mezi 2 přímkami. Její návratovou hodnotu je **double**.
- Vstup
 - * **QPointF &p1, &p2** - body první přímky
 - * **QPointF &p3, &p4** - body druhé přímky
- Výstup
 - * Úhel mezi 2 přímkami

- Metoda **getPointLineDistance**

- Tato metoda slouží k výpočtu vzdálenosti bodu od přímky. Její návratovou hodnotou je **double**
- Vstup
 - * **QPointF &q** - určovaný bod
 - * **QPointF &a, &b** - body přímky
- Výstup
 - * Vzdálenost bodu od přímky

- Přetížená metoda **rotateByAngle**

- Tato metoda slouží k rotaci dané množiny o úhel. Jejím návratovým typem je **void**.
- Vstup
 - * Přetížení 1
 - **std::vector<QPointF> &points** - vektor bodů, jež má být orotován
 - **double angle** - úhel, o který má rotace být provedena
 - * Přetížení 2
 - **QPolygonF &points** - polygon, jež má být orotován
 - **double angle** - úhel, o který má rotace být provedena
 - * Přetížení 3
 - **QLineF &points** - úsečka, jež má být orotována
 - **double angle** - úhel, o který má rotace být provedena

- Metoda **getDistance**

- Tato metoda slouží k výpočtu vzdálenosti dvou bodů. Jejím výstupním typem je **double**.
- Vstup
 - * **QPointF &a, &b** - body, mezi kterými je vzdálenost počítána
- Výstup
 - * Vypočtená vzdálenost

- Metoda **jarvisScanCH**

- Tato metoda slouží k výpočtu konvexní obálky pomocí algoritmu Jarvis Scan. Během výpočtu je ošetřována singularita existence kolineárních bodů v data-setu. Jejím výstupním typem je **QPolygonF**.
- Vstup
 - * **std::vector <QPointF> points** - vektor bodů, kolem nichž má být vytvořena konvexní obálka.
- Výstup
 - * Polygon obsahující konvexní obálku.

- Metoda **grahamScanCH**

- Tato metoda slouží k výpočtu konvexní obálky pomocí algoritmu Graham Scan. Jejím výstupním typem je **QPolygonF**.
- Vstup
 - * **std::vector <QPointF> points** - vektor bodů, kolem nichž má být vytvořena konvexní obálka.
- Výstup

- * Polygon obsahující konvexní obálku.
- Metoda **quickHullCH**
 - Tato metoda slouží k výpočtu konvexní obálky pomocí algoritmu Quick Hull. Jejím výstupním typem je **QPolygonF**.
 - Vstup
 - * **std::vector <QPointF> points** - vektor bodů, kolem nichž má být vytvořena konvexní obálka.
 - Výstup
 - * Polygon obsahující konvexní obálku.
- Metoda **quickHullLocal**
 - Pomocná metoda k výpočtu konvexní obálky metodou Quick Hull. Jejím výstupním typem je **void**.
 - Vstup
 - * **int s, e** - index počátečního a koncového bodu dělicí přímky
 - * **std::vector <QPointF> &points** - vektor bodů, kolem nichž má být vytvořena konvexní obálka.
 - * **QPolygonF &poly_ch** - polygon obsahující body konvexní obálky
 - Výstup
 - * Polygon obsahující konvexní obálku.
- Metoda **sweepLineCH**
 - Tato metoda slouží k výpočtu konvexní obálky pomocí algoritmu Sweep Line. Jejím výstupním typem je **QPolygonF**.
 - Vstup
 - * **std::vector <QPointF> points** - vektor bodů, kolem nichž má být vytvořena konvexní obálka.
 - Výstup
 - * Polygon obsahující konvexní obálku.
- Metoda **generatePoints**
 - Metoda pro generování zadaného počtu a tvaru bodů. Jejím výstupním typem je **std::vector <QPointF> points**.
 - Vstup
 - * **QSizeF &canvas_size** - rozměry kreslicího plátna, ze kterých se determinuje rozsah generovaných bodů
 - * **int point_count** - počet bodů, který se má generovat
 - * **std::string shape** - tvar vytvářené množiny bodů (random, grid, na kružnici, na elipse, na čtverci)

- Výstup
 - * Vektor nagenерованých bodů.
- Metoda **minimalRectangle**
 - Metoda pro výpočet minimálního ohraničujícího obdélníku a hlavní linie. Jejím výstupním typem je **void**.
 - Vstup
 - * **QPolygonF &poly_ch** - polygon obsahující konvexní obálku
 - * **QPolygonF &minimal_rectangle** - polygon, do kterého jsou počítány body minimálního ohraničujícího obdélníku
 - * **QLineF &direction** - hlavní linie minimálního ohraničujícího obdélníku (resp. do této proměnné je počítaná)
 - * **bool compute_dir_line** - ukazatel určující zda-li má být počítána hlavní linie minimálního ohraničujícího obdélníku

7.2 Draw

Třída `draw` slouží k vykreslení vygenerovaných (nebo naklikaných) bodů, vypočteného minimálního ohraničujícího obdélníku a hlavní linie minimálního ohraničujícího obdélníku. V této třídě jsou zároveň nagenеровané body zbavené duplicit a vypočtené konvexní obálky se zde omezují na striktní konvexní obálky (vše v metodě `setCH`). Třída dědí od třídy `QWidget`.

- Členské proměnné
 - `std::vector <QPointF> points` - vektor obsahující nagenеровané nebo naklikané body
 - `QPolygonF ch` - polygon obsahující body konvexní obálky
 - `QPolygonF rect` - polygon obsahující body minimálního ohraničujícího obdélníku
 - `QLineF direction` - hlavní linie minimálního ohraničujícího obdélníka
- Metoda `paintEvent`
 - Tato metoda slouží k vykreslení nagenеровaných (nebo naklikaných) bodů, konvexní obálky, minimálního ohraničujícího obdélníka a hlavní linie minimálního ohraničujícího obdélníka. Metoda se volá pomocí metody `repaint()`. Návrátovým typem je `void`.
 - Vstup
 - * `QPaintEvent *e`
- Metoda `mousePressEvent`
 - Metoda sloužící k uložení bodu do členské proměnné `points` určeného kliknutím myši nad kreslícím plátnem. Jejím návratovým typem je `void`.
 - Vstup
 - * `QMouseEvent *e`
- Metoda `setCH`
 - Tato metoda slouží pro kontrolu duplicity generovaných bodů, pro kontrolu alespoň 3 bodů, k zavolání příslušného algoritmu pro vypočtení konvexní obálky a k omezení konvexní obálky na striktně konvexní obálku. Metoda počítá dobu trvání výpočetních algoritmů. Jejím návratovým typem je `double`.
 - Vstup
 - * `std::string &selected_algorithm` - uživatelsky vybraný algoritmus pro počítání konvexní obálky
 - Výstup
 - * Čas trvání výpočtu.
- Metoda `setRect`

- Tato metoda slouží pro zavolání algoritmu pro výpočet minimálního ohraničujícího obdélníku a jeho hlavní linie. Jejím návratovým typem je **void**.
- Vstup
 - * **bool draw_dir_line** - uživatelsky nastavený indikátor, zda-li se má vypočítat hlavní linie minimálního ohraničujícího obdélníku
- Metoda **setPoints**
 - Metoda volající algoritmus pro generování bodů daného počtu a tvaru. Jejím návratovým typem je **void**.
 - Vstup
 - * **QSizeF &canvas_size** - rozměr kreslicího plátna pro pozdější určení rozsahu generování bodů
 - * **int count** - počet bodů, jež se má generovat
 - * **std::string &shape** - tvar, do kterého se body mají generovat
- Metoda **clearCanvas**
 - Metoda, která maže obsah kreslicího okna. Jejím návratovým typem je **void**. Do metody nevstupují žádné parametry.

7.3 SortByXAsc, SortByYAsc, SortByAngleAsc

Třídy sloužící jako sortovací kritérium - podle rostoucí souřadnice x resp. souřadnice y (při stejných souřadnicích x resp. y je druhým kritériem druhá souřadnice) a podle rostoucího úhlu mezi body (při stejném úhlu je druhým kritériem vzdálenosti mezi body).

7.4 Widget

Tato třída slouží ke komunikaci s GUI. Třída dědí od třídy `QWidget`. Všechny její metody slouží jako sloty k signálům z GUI, nemají žádné vstupní hodnoty a jejich návratovým typem je `void`.

- Metoda **`on_createCHButton_clicked`** - reaguje na zmáčknutí tlačítka pro vypočtení konvexní obálky, volá metodu **`setCH`** z třídy **`Draw`**, zapisuje čas výpočtu do GUI.
- Metoda **`on_generateButton_clicked`** - reaguje na zmáčknutí tlačítka pro generování bodů, volá metodu **`setPoints`** z třídy **`Draw`**.
- Metoda **`on_clearButton_clicked`** - reaguje na zmáčknutí tlačítka pro vymazání obsahu kreslicího plátna, volá metodu **`clearCanvas`** z třídy **`Draw`**.
- Metoda **`on_createRectButton_clicked`** - reaguje na zmáčknutí tlačítka pro vypočtení minimálního ohraničujícího obdélníka, volá metodu **`setRect`** z třídy **`Draw`**.
- Metoda **`on_helpButton_clicked`** - reaguje na zmáčknutí tlačítka pro volání nápovědy, volá okno s nápovědou **`help_dialog`** z třídy **`HelpDialog`**.

7.5 HelpDialog

Třída sloužící pro vykreslení okna s nápovědou.

8 Přílohy

- Příloha č.1: Testování výpočetních dob algoritmů - "Testovani.pdf"

9 Závěr

9.1 Politování se a vysvětlení našeho problému

9.2 Návrhy na vylepšení

10 Zdroje

1. BAYER, Tomáš. 2D triangulace, DMT [online][cit. 1.12.2018].
Dostupné z: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk5.pdf>
2. BAYER, Tomáš. 2D triangulace, DMT [online][cit. 30.11.2018].
Dostupné z: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adkcv3.pdf>