České vysoké učení technické v Praze

Fakulta stavební



Algoritmy v digitální kartografii

Úloha č. 3: Digitální model terénu a jeho analýzy

Skupina:

Sabina Kličková

Martin Vajner

Zimní semestr 2021/2022

Obsah

[1. Zadání 3](#_Toc87100785)

[2. Bonusové úlohy 3](#_Toc87100786)

[3. Popis a rozbor problémů + vzorce. 3](#_Toc87100787)

[4. Popisy algoritmů 4](#_Toc87100788)

[5. Problematické situace a popsání bonusových úloh 6](#_Toc87100789)

[6. Vstupní data, formát vstupních dat, popis. 7](#_Toc87100790)

[7. Výstupní data, formát výstupních dat, popis 7](#_Toc87100791)

[9. Závěr, možné či neřešené problémy, náměty na vylepšení 8](#_Toc87100792)

[10. Citovaná literatura 9](#_Toc87100793)

[11. Seznam obrázků 9](#_Toc87100794)

# Zadání

Vstup: množina P = {p1, ..., pn}, pi = {xi , yi , zi}. 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 významn 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 proved´te tyto analýzy:

• S využitím lineární interpolace vygenerujte vrstevnice se zadaným krokem a v zadaném intervalu, proved’te 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. Zhodnot’te 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.

|  |  |
| --- | --- |
| Delaunay triangulace, polyedrický model terénu. | +10b |
| Konstrukce vrstevnic, analýza sklonu a expozice. | +10b |

# Bonusové úlohy

V této úloze byly zpracovány následující bonusové úlohy:

|  |  |
| --- | --- |
| Krok | Hodnocení |
| Automatický popis vrstevnic. | +3b |
| Výběr barevných stupnic při vizualizaci sklonu a expozice. | +3b |

1. Popis a rozbor problémů

Zadáním úlohy je vytvořit aplikaci pomocí sw QT Creator. Cílem této aplikace je vytvořit DMT nad množinou bodů pomocí Delaunay triangulace. Množina bodů je vytvořena pomocí „klikání“ v grafickém okně či nahráním bodů ze souboru txt. DMT je vizualizováno pomocí vrstevnic s popisy a sklonu.

1. Popisy algoritmů

**Delaunay triangulace**

Pro Delaunay triangulaci byla v této úloze použita metoda Inkrementální konstrukce.

Tato metoda vytváří trojúhelníkovou síť. V počátku se hledá minimální opsaná kružnice vedoucí bodem k již vytvořené a orientované hraně. Existuje-li takový bod, změní se orientace hrany a hledání pokračuje dál.

Aktivním seznamem hran(AEL) se v této úloze rozumí ty hrany, ke kterým ještě nebyl nalezen třetí bod. Pokud je nalezena nová hrana, provede se kontrola, jestli již v seznamu není stejná hrana pouze s opačnou orientací. Pokud není, přidá se hrana do seznamu a hledání pokračuje. Když se k nějaké hraně v AEL nalezne třetí bod, je hrana ze seznamu odstraněna. „Výpočet“ končí, když je AEL prázdný.

Delaunay triangulace maximalizuje minimální úhel v trojúhelníku a je jednoznačná neleží-li čtyři body na kružnici.

Algoritmus:

1. Nalezení pivota p1=min(x) a nalezení nejbližšího bodu k p1
2. Vytvoření hrany e (p1, p2)
3. Nalezení optimálního Delaunay bodu p
   1. Pokud p neexistuje, prohodí se orientace hrany e a hledání se opakuje
4. Vytvoření hran trojúhelníku e2, e3
5. AEL ← e\_\_ AEL ← e2\_\_AEL ← e3
6. DT ← e\_\_ DT ← e2\_\_DT ← e3
7. Pokud AEL není prazdná:
   1. AEL→e //První hrana z AEL
   2. Prohození orientace hrany e
   3. Nalezení optimálního Delaunay bodu p
   4. Pokud existuje p:
      1. Vytvoření e2, e3
      2. DT ← e, DT ← e2, DT ← e3
      3. Prohození orientace hran e2, e3
      4. Pokud je e2 v AEL
         1. AEL→e2
      5. Jinak AEL ←e2
      6. Pokud je e3 v AEL
         1. AEL→e3
      7. Jinak AEL ←e3

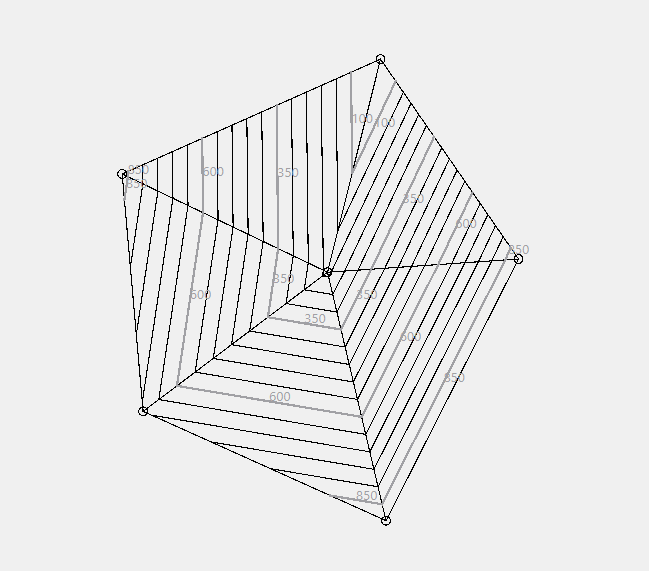
**Tvorba vrstevnic**

Vrstevnice je křivka spojující místa se stejnou nadmořskou výškou. Dělí se na základní a hlavní, kdy za hlavní je standartně považována každá pátá vrstevnice. Dalšími typy vrstevnic jsou pomocné a doplňkové.

Zobrazování vrstevnic závisí na jejich typu, kde základní jsou kresleny čarou tenkou, hlavní zvýrazněnou a doplňkové čerchovanou. Popis je přítomen standartně u vrstevnic hlavních, a to ve směru z kopce.

Pro tuto úlohu byla použita metoda zobrazení pomocí lineární interpolace. Jejím základem je hledání průsečnice roviny trojúhelníku a vodorovné roviny ve výšce h. Body A, B této průsečnice jsou určeny z podobnosti trojúhelníků:

Po spojení těchto hran vznikne hrana tvořící v daném trojúhelníku vrstevnici o dané výšce.



**Expozice (Orientace)**

Výpočet x a y části normálového vektoru:

ux = Δx2, x1; uy = Δy2, y1; uz = Δz2, z1;

vx = Δx2, x3; uy = Δy2, y3; uz = Δz2, z3;

Expozice:

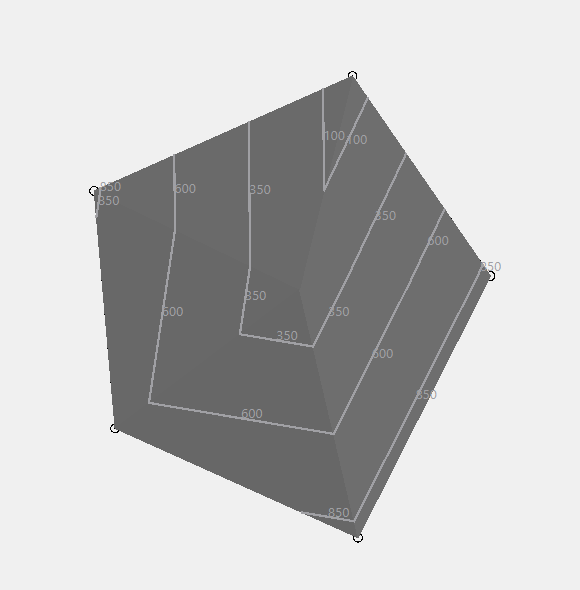
**Sklon**

Výpočet normálového vektoru roviny trojúhelníku:

ux = Δx2, x1; uy = Δy2, y1; uz = Δz2, z1;

vx = Δx2, x3; uy = Δy2, y3; uz = Δz2, z3;

Sklon:



1. Popsání bonusových úloh

**Automatický popis vrstevnic**

Popis vrstevnic byl generován pouze pro vrstevnice hlavní. Z bodů hran o souřadnicích x,y,z byla získána hodnota z a vykreslena na vrstevnici. Umístění na vrstevnici realizováno průměrem souřadnic x, y koncových bodů hran.

**Barevné stupnice**

Nejprve byl vytvořen combobox pro výběr typu vykreslení zklonu s možnostmi vykreslení barevné či šedé stupnice. Poté byl zvolen způsob vykreslování, kde podle vzorce barva=255 – sklon \* 255/PI. Tato hodnota pak byla přiřazena každému trojúhelníku pomoci setBrush(QColor (col,col,col)). U stupnice barevné byl použit systém intervalů, kde každý interval má jiný odstín barvy.

1. Vstupní data, formát vstupních dat, popis.

Vstupní data byla generována z datové sady ArcCR500. Data byl exportována v ArcGis Pro do formátu ASCII(.Txt).

Formát dat textového souboru:

y << x << z

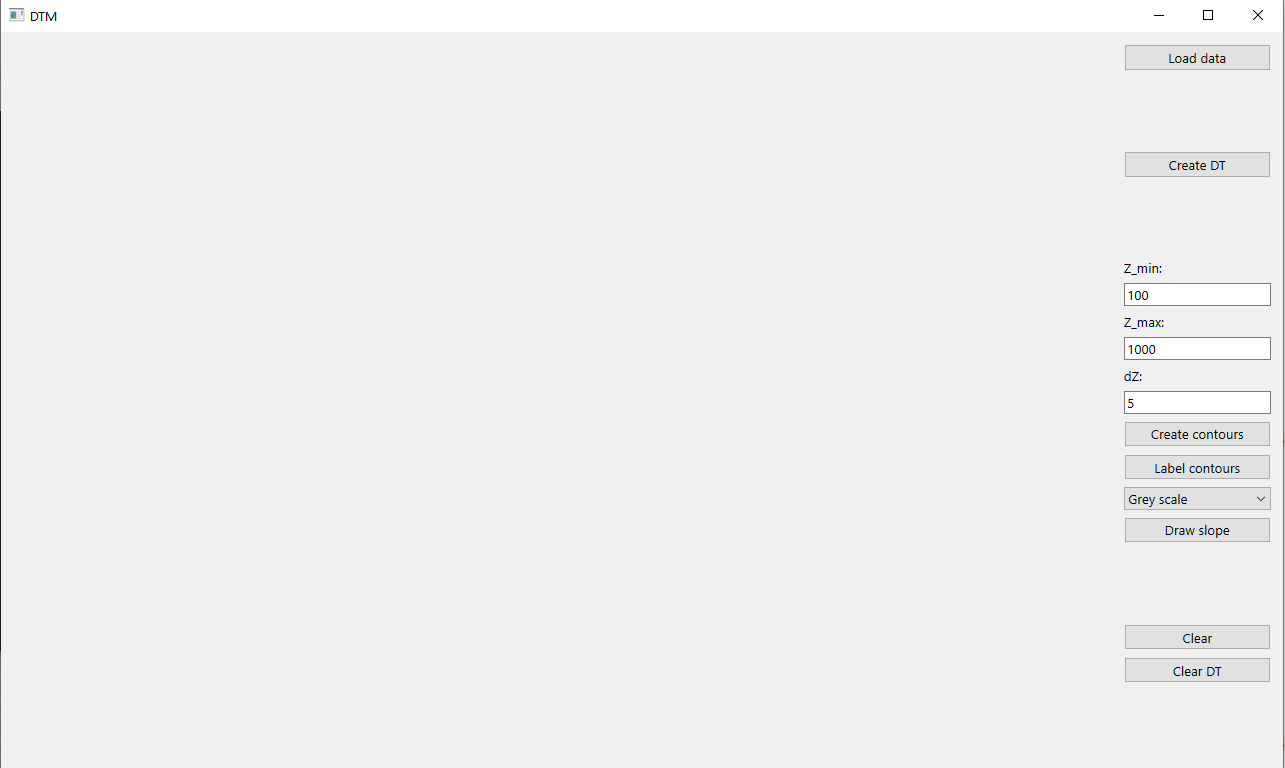
kde:

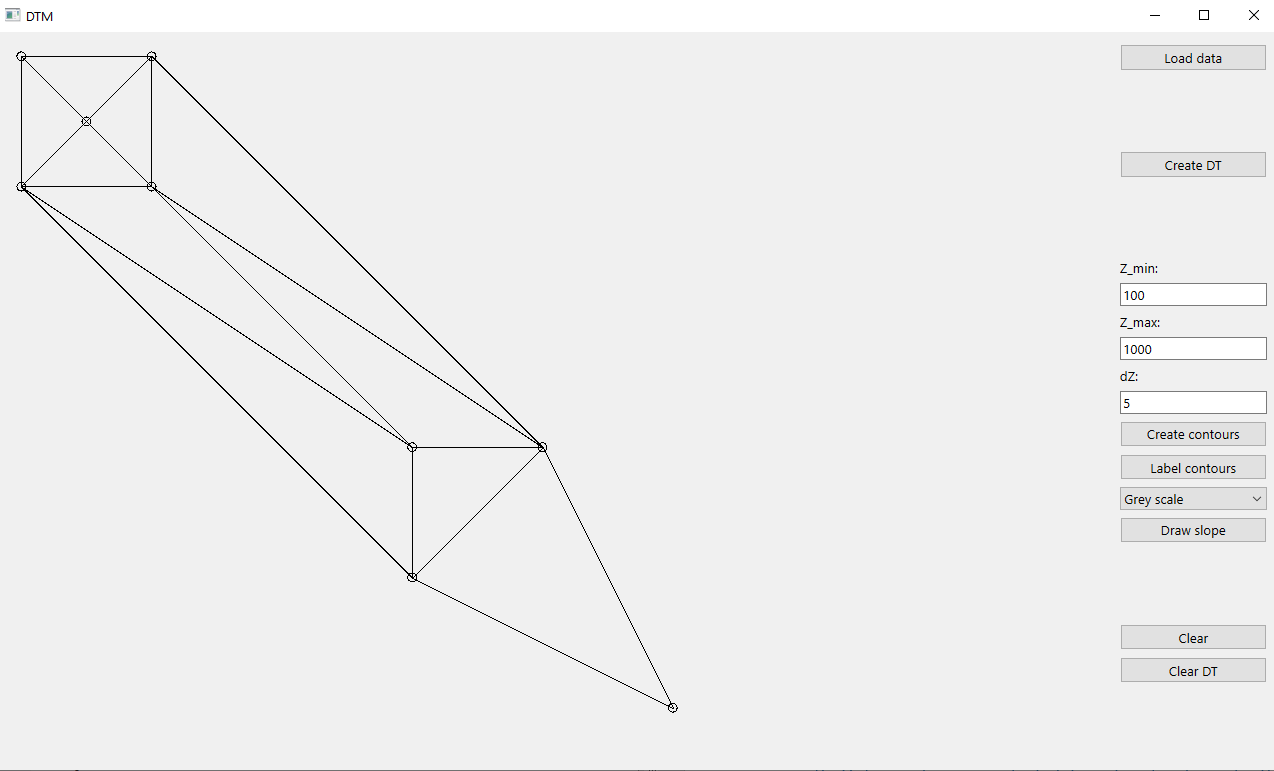
x - souřadnice x bodu

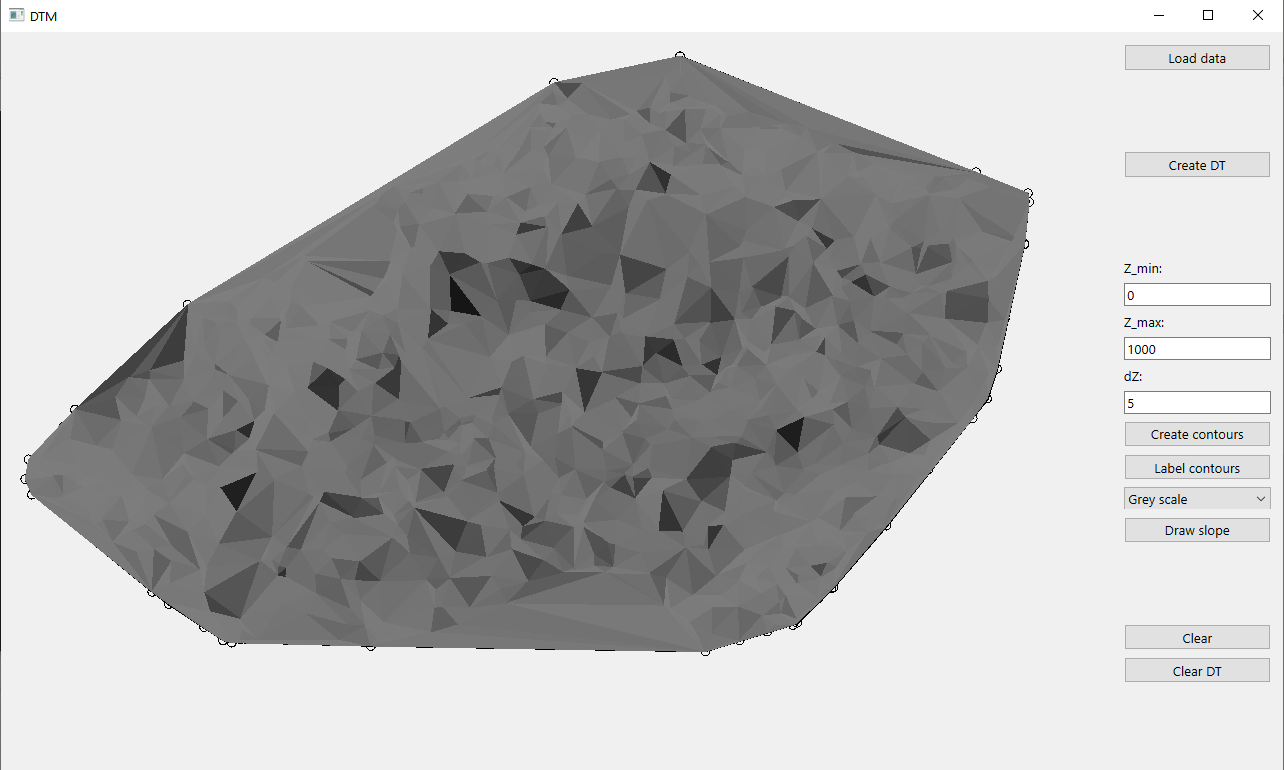
y – souřadnice y bodu

1. Výstupní data, formát výstupních dat, popis

Výstupem je aplikace, která dokáže nahrát data a vytvořit nad nimi trojúhelníkovou síť. Rovněž dokáže vykreslit vrstevnice a jejich popis a sklon pomocí barevných škál.







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

class **Draw** : public QWidget

private:

std::vector<QPoint3D> points;

* definice proměnné uložení jednotlivých bodů(x,y,z)

std::vector<Edge> dt;

* definice proměnné uložení Delaunay trojúhelníků

std::vector<Edge> contours;

* definice proměnné uložení vrstevnic

std::vector<Triangle> triangles;

* definice proměnné uložení trojúhelníků

std::vector<Edge> contourslabel,contourslabel2, contoursmain;

* definice proměnné uložení hlavních vrstevnic a jejich popisů

boolean greycol, colors;

* definice proměnné uložení nastaveného vykreslení barev

int width,height;

public:

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

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

* vykreslení všech grafických prvků a jehjich grafické zvýraznění

void ***mousePressEvent***(QMouseEvent \*event);

* přidání proměnné(bod) do seznamu pomocí grafického okna

void **clear**();

* mazaní vykreslených proměnných

std::vector<QPoint3D> **getPoints**(){return points;}

* vracení proměnných(bod)

void **setDT**(std::vector<Edge> &dt\_){dt = dt\_;}

* přiřazení vektoru Delaunay trojúhelníků

std::vector<Edge> **getDT**(){return dt;};

* vracení vektoru Delaunay trojúhelníků

void **setContours**(std::vector<Edge> &contours\_){contours = contours\_;}

* přiřazování vrstevnic

std::vector<Edge> **geContours**(){return contours;};

* vracení vrstevnic

std::vector<Triangle> **getTriangles**(){return triangles;};

* vracení trojůhelníku pro funkci slope

void **setTriangle**(std::vector<Triangle> &triangles\_){triangles = triangles\_;};

* přiřazování trojúhelníku pro funkci slope

void **clearDT**();

* smazání vektoru Delaunay trojúhelníků

void **setLabelContours**(std::vector<Edge> &contourslabel\_){contourslabel = contourslabel\_;}

* přiřazení popisů vrstevnic

std::vector<Edge>& **getLabelContours**() {return contourslabel;}

* vracení popisků vrstevnic

void **setMainContours**(std::vector<Edge> &contoursmain\_){contoursmain = contoursmain\_;}

* přiřazení hlavní vrstevnice (tzn. každá pátá)

std::vector<Edge>& **getMainContours**() {return contoursmain;}

* vracení hlavních vrstevnic (tzn. každá pátá)

void **setGreycol**(bool &greycol\_){greycol = greycol\_;};

* nakreslení vykreslování stupňů šedi profunkci slope

void **setColors**(bool &colors\_){colors = colors\_;};

* nastavení vykreslování barevné škály pro funkci slope

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

* načtení dat formátu ASCII(.txt)

int **getWidth**(){return width;}

int **getHeight**(){return height;}

void **setWidth**(int width\_){width=width\_;};

* nastavení šířky QWidgetu pro vykreslení

void **setHeight**(int height\_){height=height\_;};

* nastavení výšky QWidgetu pro vykreslení

class **Algorithms**

{

public:

**Algorithms**();

int **getPointLinePosition**(QPoint3D &a,QPoint3D &p1,QPoint3D &p2);

* zjištění vzájemné polohy přímky a bodu (vlevo, vpravo, hrana)
* vstup: souřadnice bodu q a lomových bodů polygonu

std::tuple<QPoint3D,double> **getCircleCenterAndRadius**(QPoint3D &p1,QPoint3D &p2,QPoint3D &p3);

* zjištění středu a poloměru kruhu, který je definován pomocí tří bodů

int **getDelaunayPoint**(QPoint3D &s,QPoint3D &e,std::vector<QPoint3D> &points);

* vracení tzv. Delaunayova bodu

int **getNearestPoint**(QPoint3D &p, std::vector<QPoint3D> &points);

* vracení nejbližšího bodu

std::vector<Edge> **dT**(std::vector<QPoint3D> &points);

* definice vektoru hran tvořených 3D body

void **updateAEL**(Edge &e, std::list<Edge> &ael);

* update seznamu aktivních trojúhelníků

QPoint3D **getContourPoint**(QPoint3D &p1, QPoint3D &p2, double z);

* vracení bodu pro vkreslení vrstevnic

std::vector<Edge> **getContourLines**(std::vector<Edge> &dt, double zmin, double zmax, double dz);

* vracení vrstevnic

double **getSlope**(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);

* vracení sklonu

double **getExposition**(QPoint3D &p1, QPoint3D &p2, QPoint3D &p3);

* vracení expozice

std::vector<Triangle> **analyzeDTM**(std::vector<Edge> &dt);

* výpočet analýzy DTM z Delaunay triangulace

1. Závěr, zhodnocení algoritmů
2. Citovaná literatura

1. **Tomáš, Bayer.** Perslonal page of Bayer Tomas. *Charles University of Prague.* [Online] [Citace: 05. 11 2021.] https://web.natur.cuni.cz/~bayertom/index.php/teaching/algoritmy-v-digitalni-kartografii.

1. Seznam obrázků

[Obrázek 1: Ukázka konvexní obálky nad budovou4](#_Toc87188232)

[Obrázek 2: Ukázka algoritmu Jarvis Scan (1) 4](file:///C:\Users\Sabča\Downloads\ADKI-main\Uloha2\TZ_uloha2.docx#_Toc87188233)

[Obrázek 3: Ukázka metody detekce natočení budov Minimum Area Enclosing Rectangle (1) 5](#_Toc87188234)

[Obrázek 4: Ukázka metody detekce natočení budov Wall Average (1) 5](#_Toc87188235)

[Obrázek 5: Ukázka metody detekce natočení budov Longest Edge (1) 6](#_Toc87188236)

[Obrázek 6: Ukázka metody detekce natočení budov Weighted Bisector (1) 7](#_Toc87188237)

[Obrázek 7: Okno aplikace po spuštění 8](#_Toc87188238)

[Obrázek 8: Okno aplikace po nahrání dat 9](#_Toc87188239)

V Praze dne 7.11.2021