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

I. Obsah

1. Zadání

Vstup: $množina\ budov\ B = \{B_i\}_{i=1}^{n}, budova\ B_i = \{P_{i,j}\}_{j=1}^{m}$

Výstup: $G(B_i)$

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

2. Bonusové úlohy

V této úloze byly zpracovány následující bonusové úlohy:

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

3. Popis a rozbor problémů + vzorce.

Definice konvexní obálky: Konvexní obálka $\mathcal{H}$ 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.

4. Popisy algoritmů

I. Metody konstrukce konvexní obálky

a. Jarvis scan

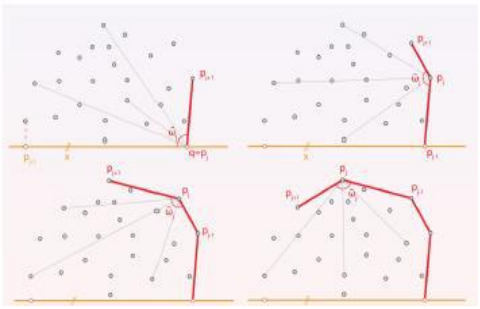
Tato metoda je používána pro vyhledávání bodů konvexní obálky podle největšího úhlu. Nevýhodou je, že tři body nesmí ležet na přímce. Tato skutečnost musí být tedy ošetřena.

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, q_y).

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:

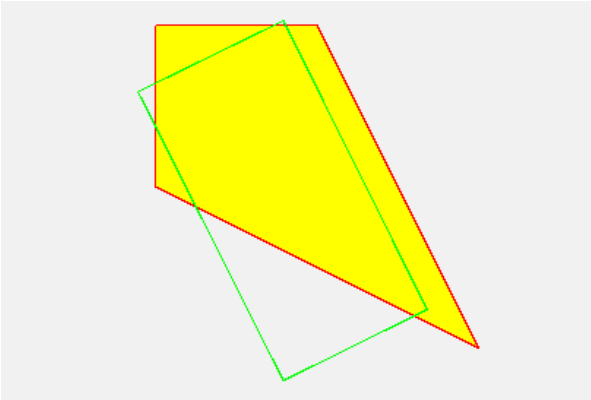
1. Nalezení pivota q, $q = \min(y_i)$
2. Přidání q do konvexní obálky H
3. Inicializace $p_{j-1} \in X$, $p_j = q$, $p_{j+1} = p_{j-1}$
4. Dokud $p_{j+1} \neq q$
   - Nalezení bodu s maximálním úhlem p_{j+1}
   - Přidání bodu p_{j+1} do konvexní obálky
   - Změna indexu, $p_{j-1} = p_j$; $p_j = p_{j+1}$



Obrázek 2: Ukázka algoritmu Jarvis Scan (1)

II. <u>Minimum Area Enclosing Rectangle</u>

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)



Obrázek 3: Ukázka metody detekce natočení budov Minimum Area Enclosing Rectangle

III. Detekce úhlu natočení budov

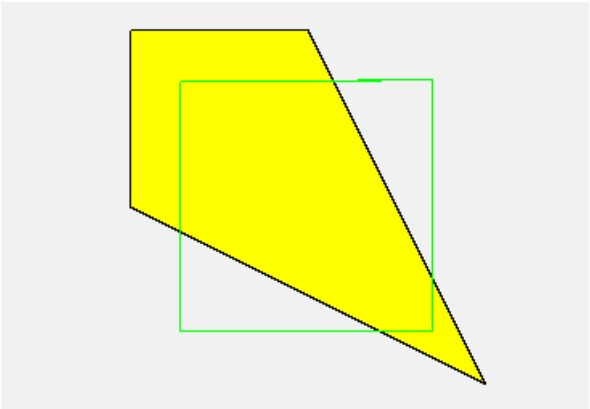
Budova před generalizací a po generalizaci musí mít uchovánu orientaci vzhledem k ostatním obsahovým prvkům mapy (např. zachování uliční čáry). Nutnost detekovat tzv. hlavní směry budovy (jsou na sebe zpravidla kolmé). Popisují orientaci (tj. natočení) budovy vzhledem k ostatním prvkům mapy. (1)

a. 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 σ_i všech hran. Poté redukce σ_i o úhel σ ' (natočení budovy || s osami x, y). (1)

Metoda, která vytváří ohraničují obdélník za pomoci průměru strany. Je vypočítán počáteční směr a následně směry pro všechny segmenty. Pro tyto výpočty je potřeba spočítat směry a délky, směrové rozdíly a vážené průměrné součty. Následně přichází na řadu vážený průměr, kdy je obdélník otočen o záporný vážený průměr.

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



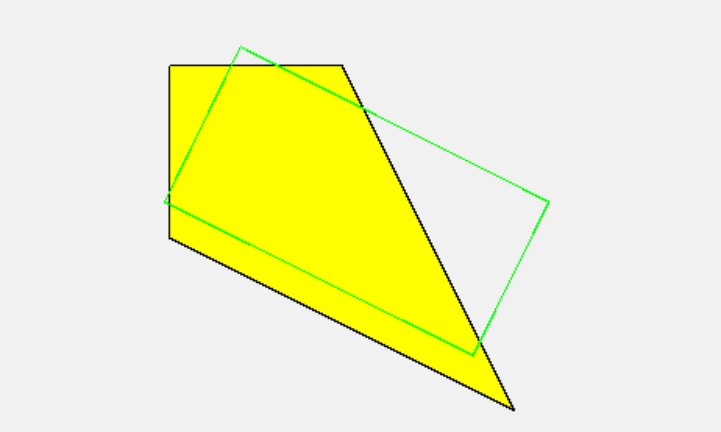
Obrázek 4: Ukázka metody detekce natočení budov Wall Average

b. 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.



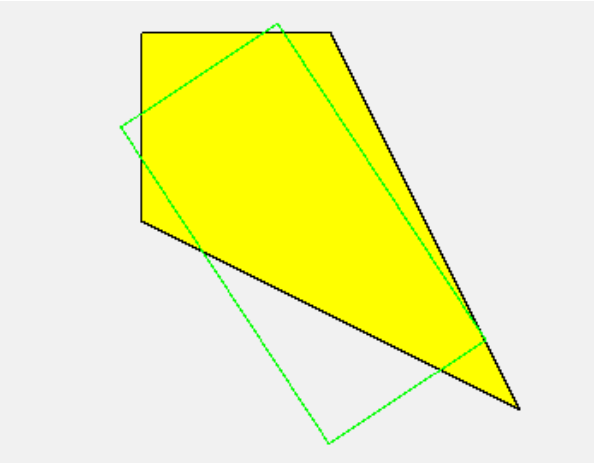
Obrázek 5: Ukázka metody detekce natočení budov Longest Edge

c. Weighted Bisector

Hledány dvě nejdelší úhlopříčky, směrnice σ_1, σ_2, délky s_1, s_2. Hlavní směr dán váženým průměrem $\sigma = \frac{s_1\sigma_1 + s_2\sigma_2}{s_1+s_2}$. *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.



Obrázek 6: Ukázka metody detekce natočení budov Weighted Bisector

5. Problematické situace

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;}
```

Předání dat do metod

Další problémovou situací bylo předání správných dat do metod. Kdy z canvasu byl převzat vektor polygonů. Ten se následně pomocí cyklu musel rozdělit na jednotlivé polygony a ty poté na jednotlivé vektory bodů. Poté se vytvořila podmínka, podle které byly brány jen ty vektory, které obsahovaly více než dva body.

6. Vstupní data, formát vstupních dat, popis.

Vstupní data byla přejata z 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

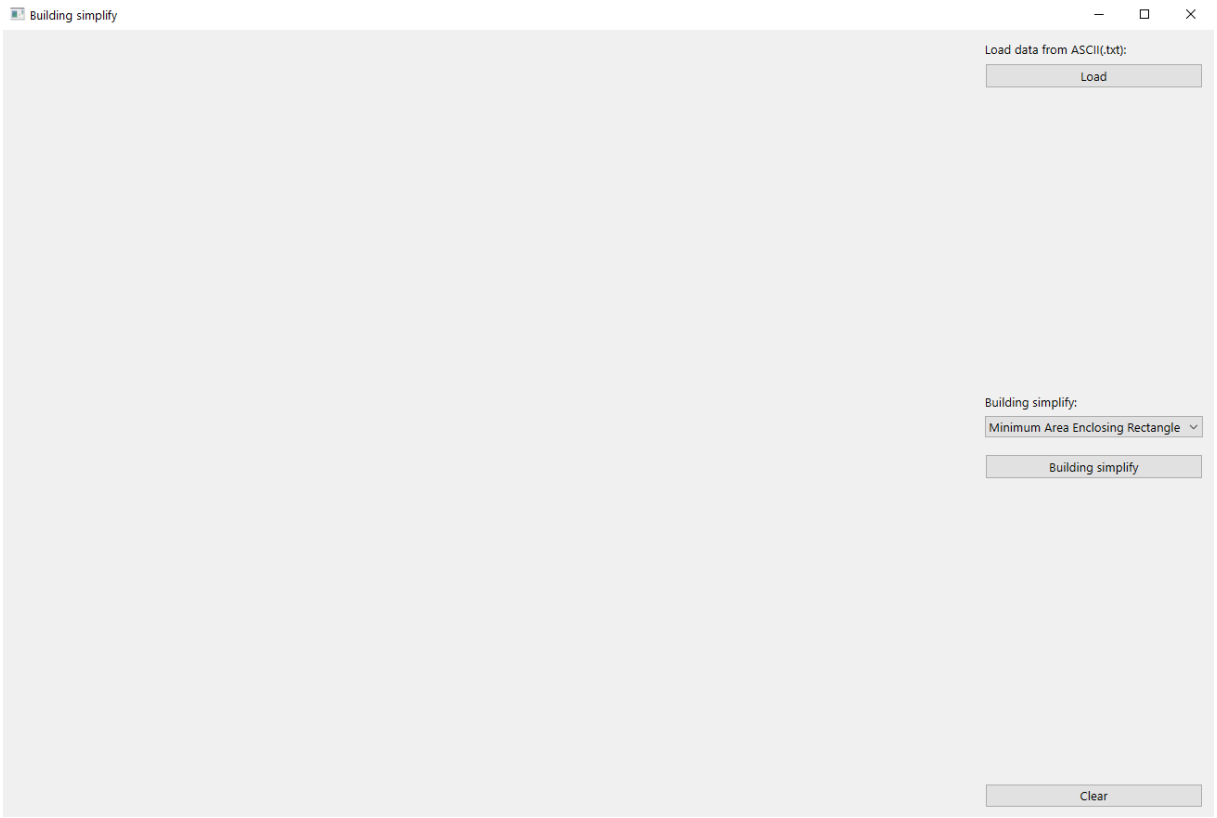
fid – číslo původního polygonu

7. Výstupní data, formát výstupních dat, popis

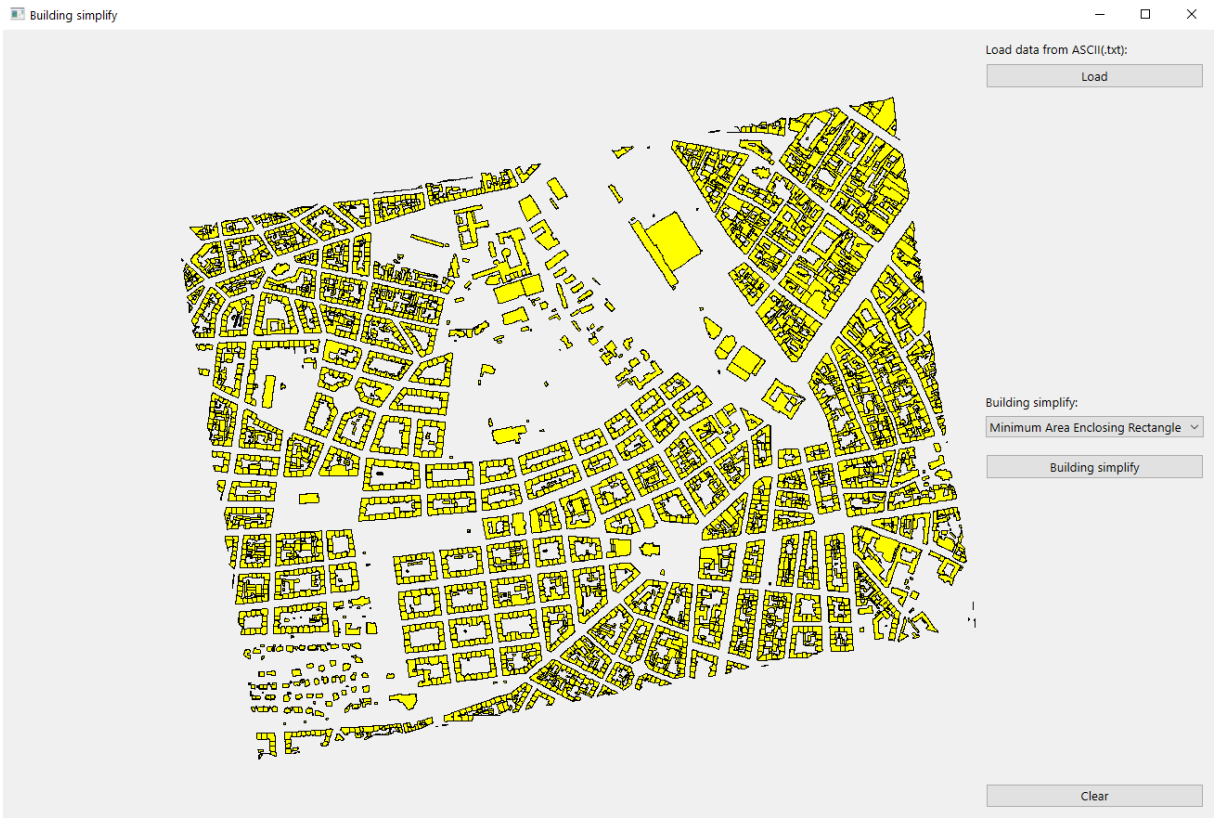
Výstupem je aplikace, která dokáže nad vhodně zpracovanými daty vytvořit konvexní obálku pomocí několika metod.

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, tlačítko pro načtení dat, tlačítko vytvoření generalizace polygonů a tlačítko clear, které smaže okno.



Obrázek 7: Okno aplikace po spuštění



Obrázek 8: Okno aplikace po nahrání dat

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

```
class Algorithms
public:
  Algorithms();
  double get2LinesAngle(QPoint &p1, QPoint &p2, QPoint &p3, QPoint &p4);
```

- funkce zjišťuje hodnoty úhlu mezi dvěma hranami polygonu
- vstup: souřadnice bodů vektorů

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

- funkce vytvoří konvexní obálky pomocí algoritmu Jarvis Scan

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

- funkce otáčí datovou sadu podle úhlu

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

- funkce pro urcení souřadnic ohraničujícího obdélníku dat

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

- funkce metody detekce natočení budov Minimum Area Enclosing Rectangle

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

- funkce metody detekce natočení budov Wall Average

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

- funkce počítá plochy budov podle L'Huillierova vzorce

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

- funkce mění velikost obdélníků

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

- funkce metody detekce natočení budov Longest Edge

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

- funkce metody detekce natočení budov Weighted Bisector

```
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> pols;
                - 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> getPolygons(){return pols;};
                - 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
```

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

Námi vytvořená aplikace dokáže z načteného souboru obsahujícího souřadnice bodu, číslo bodu a číslo polygonu, vykreslit a vyznačit polygony a spočíst a vytvořit nad nimi konvexní obálku či jejich generalizaci.

Z bonusových úloh byly vyřešeny následující: Generalizace budov metodou Longest Edge a Generalizace budov metodou Weighted Bisector.

Aplikace by se dala vylepšit možností „zoomu“ na plátno, aby byly lépe vidět vykreslované struktury a následnou možností hýbat se zobrazovanou plochou.

U metody Minimum Area Enclosing Rectangle je potřeba vytvořit konvexní obálky, což z ní dělá nejsložitější zpracovávanou metodu.

Nejvhodnější metodou se nám jeví metoda Wall Average, a to z hlediska výpočtu. Metoda není ideální pro budovy s ostrými úhly, ale jinak dává dobré výsledky.

Metoda Longest Edge nedosahuje nejpřesnějších výsledků, ale její zápis tvoří jednoduchý algoritmus.

U metody Weighted Bisector při předání velkého množství dat nám aplikace padá. Tento problém by se odhadem dal vyřešit optimalizací, ale na to bohužel postrádáme zkušenosti.

Efektivitu metod Wall Average a Minimum Area Enclosing Rectangle nemůžeme zhodnotit více, neboť při vykreslení a výpočtu na minutu a více zamrzne PC.

	procentuální úspěšnost metody na testovaných datech
Minimum Area Enclosing Rectangle	100%
Wall Average	92%
Longest Edge	90%
Weighted Bisector	86%

10.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.

11. Seznam obrázků

V Praze dne 14.12.2021