Erkennen eines Graphen aus einer Punktwolke

Wir sind hier:

Das Problem

Eingabe ein metrischer Raum (X, d), z.B.

1. GPS-Koordinaten eines Autos, das in einer Stadt herumfährt
2. Koordinaten der schwarzen Pixel in einem Schwarz-Weiß-Bild

Ausgabe eine Approximation des metrischen Graphen, der dem metrischen Raum zugrunde liegt, z.B.

1. das Straßennetz der Stadt
2. der Graph, der im Bild abgebildet wird
   $\rightarrow$ Schrifterkennung

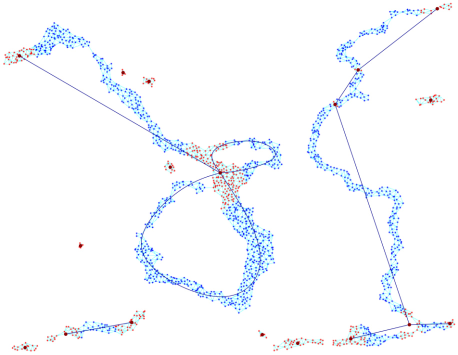
Wozu ist das nützlich?

1. Struktur in große Mengen geometrischer Daten bringen um deren Analyse zu ermöglichen/erleichtern

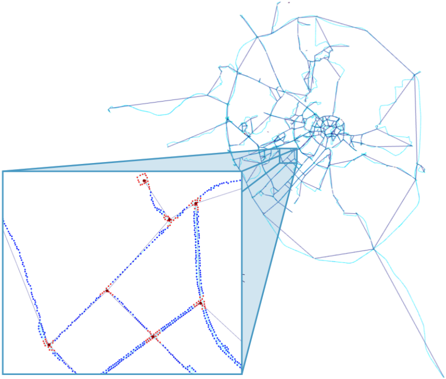
Wozu ist das nützlich?

1. Struktur in große Mengen geometrischer Daten bringen um deren Analyse zu ermöglichen/erleichtern
2. Häufig enthalten die Daten rauschen und sind umfangreich $\rightarrow$ Ziel: Datenmenge kompakt in ihren wichtigsten Verzweigungen darstellen

Beispiel (Erdbeben)



Beispiel (Straßennetz)



Wir sind hier:

Einleitung

Vorverarbeitung

Der Algorithmus

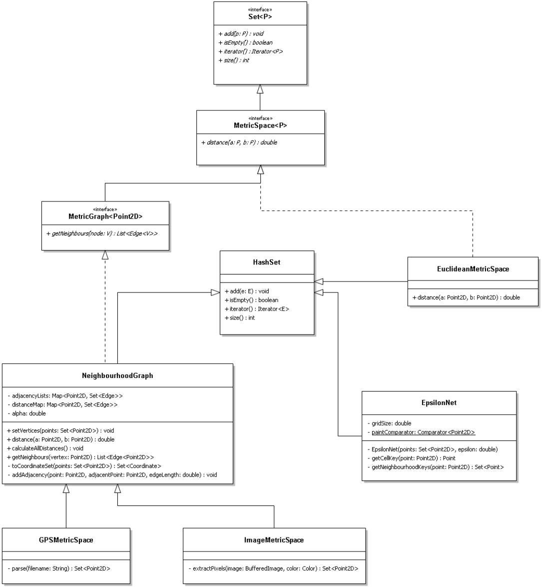
Organisation

Probleme und Lernerfolge

Vorverarbeitung

1. Menge von Punkten erstellen
   - Koordinaten aus GPS-Spur-Dateien extrahieren
   - Koordinaten der schwarzen Pixel eines Bildes bestimmen
2. ggf. Punktmenge auf repräsentative Teilmenge reduzieren
   - metrisches ε-Netz
   - Laufzeit: $O(n)$ (unter Verwendung eines Gitters)
3. auf die Punktmenge basierenden α-Komplex erstellen
   - α-Komplex: Unterkomplex der Delaunay-Triangulierung
   - lange Kanten der Triangulierung werden entfernt
   - verwenden JTS Topology Suite (DelaunayTriangulationBuilder)
4. Abstandsmethode implementieren
   - $d(x, y) :=$ Länge des kürzesten Pfades zwischen x und y
   - Algorithmus von Floyd-Warshall

Klassendiagramm



Verbesserungsmöglichkeiten

- α-Komplex
  - um Lücken zu vermeiden, muss man u. U. ein großes α wählen
  - führt dazu, dass mehr Knoten durch Kanten verbunden werden $\rightarrow$ Struktur schwieriger zu erkennen, Abstände verzerrt
  - Idee: Zeitstempel in den GPS-Spur-Dateien in Betracht ziehen
- Laufzeit der Abstandsberechnung
  - Floyd-Warshall Laufzeit: $O(n^3)$
  - α-Komplexe enthalten $O(n)$ Kanten
  - Idee: Algorithmus von Dijkstra verwenden $\rightarrow$ $O(n^2 \log n)$
- Abstandsberechnung für GPS-Koordinaten
  - aktuell interpretieren wir die Koordinaten als Punkte auf einer Ebene und berechnen den euklidischen Abstand
  - Idee: Orthodrome berechnen $\rightarrow$ Abstand in km

Wir sind hier:

Drei wesentliche Schritte

1. **Labeling**: Welche Punkte gehören zu Kanten, welche zu Knoten? $\rightarrow$ Punkte mit entsprechendem Label versehen

Drei wesentliche Schritte

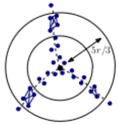
1. **Labeling**: Welche Punkte gehören zu Kanten, welche zu Knoten? $\rightarrow$ Punkte mit entsprechendem Label versehen
2. **Rekonstruktion**: Anhand der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun $\rightarrow$ Rekonstruktion des der Punktmenge zugrundeliegenden Graphen

Drei wesentliche Schritte

1. **Labeling**: Welche Punkte gehören zu Kanten, welche zu Knoten? $\rightarrow$ Punkte mit entsprechendem Label versehen
2. **Rekonstruktion**: Anhand der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun $\rightarrow$ Rekonstruktion des der Punktmenge zugrundeliegenden Graphen
3. **Metrik wiederherstellen**: Kanten mit Abständen versehen

Zu Schritt 1: Labeling

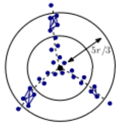
1. Betrachten einen kreisförmigen Ausschnitt um jeden Punkt und bestimmen Anzahl der Zusammenhangskomponenten darin



- Grad des Punkts = # Zusammenhangskomponenten im kreisförmigen Ausschnitt

Zu Schritt 1: Labeling

1. Betrachten einen kreisförmigen Ausschnitt um jeden Punkt und bestimmen Anzahl der Zusammenhangskomponenten darin

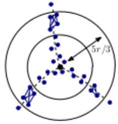


   - Grad des Punkts = # Zusammenhangskomponenten im kreisförmigen Ausschnitt

2. Grad == 2 $\rightarrow$ *edge point* (wird später zu einer Kante gehören)

Zu Schritt 1: Labeling

1. Betrachten einen kreisförmigen Ausschnitt um jeden Punkt und bestimmen Anzahl der Zusammenhangskomponenten darin

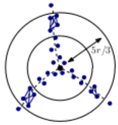


   - Grad des Punkts = # Zusammenhangskomponenten im kreisförmigen Ausschnitt

2. Grad == 2 → *edge point* (wird später zu einer Kante gehören)
3. Grad != 2 → *preliminary branch point* (wird später zu einem Knoten gehören, aber noch umbenannt zu *branch point*)

Zu Schritt 1: Labeling

1. Betrachten einen kreisförmigen Ausschnitt um jeden Punkt und bestimmen Anzahl der Zusammenhangskomponenten darin



   - Grad des Punkts = # Zusammenhangskomponenten im kreisförmigen Ausschnitt

2. Grad == 2 → *edge point* (wird später zu einer Kante gehören)
3. Grad != 2 → *preliminary branch point* (wird später zu einem Knoten gehören, aber noch umbenannt zu *branch point*)
4. Alle weniger als x von einem *preliminary branch point* entfernten Punkte werden als *branch points* eingeordnet (Knoten)

Zu Schritt 2: Rekonstruktion des Graphen

- Mithilfe der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun

Zu Schritt 2: Rekonstruktion des Graphen

- Mithilfe der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun
  - Punkte sind jetzt nach dem Labeling in zwei Mengen aufgeteilt:
    - *edge points*
    - *branch points*

Zu Schritt 2: Rekonstruktion des Graphen

- Mithilfe der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun
  - Punkte sind jetzt nach dem Labeling in zwei Mengen aufgeteilt:
    - *edge points*
    - *branch points*
  - Um die Anzahl der späteren Kanten und Knoten zu bestimmen, wird Rips-Vietoris-Graph für beide Mengen erstellt

Zu Schritt 2: Rekonstruktion des Graphen

- Mithilfe der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun
  - Punkte sind jetzt nach dem Labeling in zwei Mengen aufgeteilt:
    - *edge points*
    - *branch points*
  - Um die Anzahl der späteren Kanten und Knoten zu bestimmen, wird Rips-Vietoris-Graph für beide Mengen erstellt
    - Im Rips-Vietoris-Graphen werden alle Punkte, die innerhalb eines gewissen Abstands zueinander liegen, zu einer Zusammenhangskomponente gefasst

Zu Schritt 2: Rekonstruktion des Graphen

- Mithilfe der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun
  - Punkte sind jetzt nach dem Labeling in zwei Mengen aufgeteilt:
    - *edge points*
    - *branch points*
  - Um die Anzahl der späteren Kanten und Knoten zu bestimmen, wird Rips-Vietoris-Graph für beide Mengen erstellt
    - Im Rips-Vietoris-Graphen werden alle Punkte, die innerhalb eines gewissen Abstands zueinander liegen, zu einer Zusammenhangskomponente gefasst
  - Jede Zusammenhangskomponente entspricht einem Knoten, bzw. einer Kante

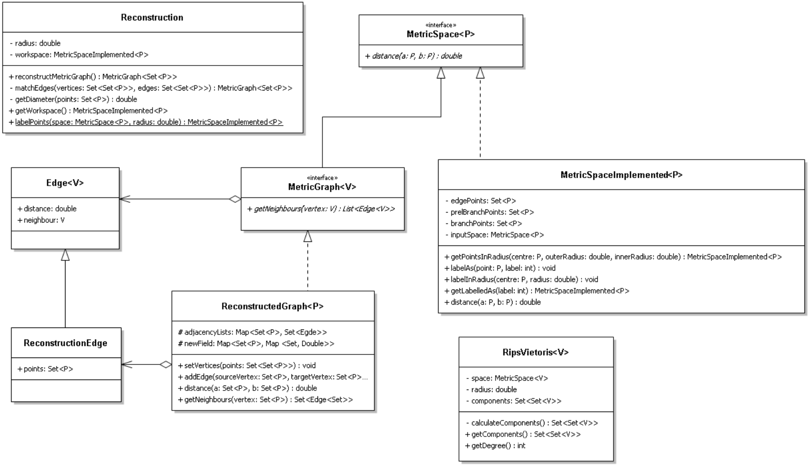
Zu Schritt 2: Rekonstruktion des Graphen

- Mithilfe der Labels bestimmen, welche Punkte sich zu einem Knoten und welche sich zu einer Kante zusammentun
  - Punkte sind jetzt nach dem Labeling in zwei Mengen aufgeteilt:
    - *edge points*
    - *branch points*
  - Um die Anzahl der späteren Kanten und Knoten zu bestimmen, wird Rips-Vietoris-Graph für beide Mengen erstellt
    - Im Rips-Vietoris-Graphen werden alle Punkte, die innerhalb eines gewissen Abstands zueinander liegen, zu einer Zusammenhangskomponente gefasst
  - Jede Zusammenhangskomponente entspricht einem Knoten, bzw. einer Kante
  - Zwei Punkte werden durch eine Kante verbunden, wenn sie Punkte in ihrer Zusammenhangskomponente haben, die einen gewissen Abstand zu Punkten in der selben Zusammenhangskomponente einer Kante nicht überschreiten

Zu Schritt 3: Metrik wiederherstellen

- Kanten mit Abständen versehen
  - Jeder Kante wird als Länge der Durchmesser (längster kürzester Weg) ihrer Zusammenhangskomponente zugewiesen

Klassendiagramm



Wir sind hier:

Kommunikation und Tools

- Kommunikation und Tools
  - regelmäßige Treffen
  - Skype
  - E-Mail
  - Trello
  - GitHub
- Programmierung
  - Festlegung von Interfaces
  - 3 Gruppen

Wir sind hier:

Probleme und Lernerfolge

- Probleme
  - Testen der Komponenten schwierig
  - Absprung eines Gruppenpartners
- Lernerfolge
  - Arbeit in Teilgruppen
  - Planung der Software
  - Zusammenführen der Teilstücke