Michał Ciołczyk
Michał Janczykowski

Vertex clustering

Dokumentacja techniczna

1. Wymagania

Projekt został wykonany w języku Python, z wykorzystaniem biblioteki CGAL dostępnej za pośrednictwem tzw. bindingów. W celu uruchomienia programu należy zapewnić:

- Python 2.7
- CGAL - wersja 4.4 lub nowsza
- SWIG - wersja 2.0 lub nowsza
- cgal-swig-bindings - wersja 4.4 lub nowsza

Instalacja bindingów została szczegółowo opisana na stronie projektu cgal-swig-bindings: https://github.com/CGAL/cgal-swig-bindings/wiki/Installation

W przypadku systemu Windows możliwe jest zainstalowanie gotowej wersji binarnej, dostępnej na stronie: http://www.lfd.uci.edu/~gohlke/pythonlibs/

2. Struktura projektu

`bucket.py`:

- **`Bucket`** - klasa reprezentująca pojedynczy klaster, pozwala na wyznaczenie reprezentanta.
  - Konstruktor przyjmuje następujące parametry:
    - **`coordinates`** - współrzędne środka klastra (**`tuple`** z 3 wartościami typu **`float`**)
    - **`representative_function`** - funkcja wyznaczania reprezentanta (z pliku **`representative_functions.py`**),
    - **`epsilon`** - **`float`**
  - **`append(vertex)`** - dodaje vertex do klastra
  - **`representative`** - pobiera reprezentanta klastra (zwraca **`tuple`** z trzema współrzędnymi typu **`float`**)
- **`get_bucket_for_vertex(vertex, epsilon)`** - funkcja pomocnicza zwracająca współrzędne środka klastra zawierającego dany wierzchołek. Parametry:
  - **`vertex`** - uchwyt do wierzchołka z bindingów do biblioteki CGAL (typu **`Polyhedron_3_Vertex_handle`**),
  - **`epsilon`** - wartość **`float`**

`clustering.py`:

- **`cluster(mesh, epsilon, representative_method, filename)`** - funkcja dokonująca klasteryzacji (serce programu). Parametry:
  - **`mesh`** - siatka typu **`Polyhedron_3`** z bindingów do biblioteki CGAL,
  - **`epsilon`** - wartość typu float,
  - **`representative_method`** - funkcja wyznaczania reprezentanta (z pliku **`representative_functions.py`**),
  - **`filename`** - nazwa pliku wyjściowego (koniecznie plik OFF).

`representative_methods.py`:

- **`dummy_representative(bucket)`**,
  `mean_representative(bucket)`,
  `median_representative(bucket)`,
  `quadric_errors_representative(bucket)`
  - poszczególne metody wyznaczania reprezentanta klastra. Parametr:
    - **`bucket`** - klaster, dla którego wyznaczamy reprezentanta (typu **`Bucket`**).

`mesh_loader.py`

- Klasy: **`ObjLoader`** oraz **`OffLoader`** pozwalające wczytać siatki z plików typu OBJ oraz OFF. Ważniejsze metody:
  - Konstruktor - parametr:
    - **`filename`** - nazwa pliku do załadowania.
  - **`to_polyhedron()`** - zwraca siatkę typu **`Polyhedron_3`** z bindingów do biblioteki CGAL.

`main.py`:

- Funkcja **`cluster_mesh`** oraz możliwość uruchomienia programu (opisane poniżej).

3. Użycie funkcji `cluster_mesh` w kodzie

W pliku main.py (a także po zaimportowaniu pakietu vertex_clustering) dostępna jest funkcja cluster_mesh, która pozwala na uruchomienie algorytmu. Wygląda ona następująco:

```
def cluster_mesh(mesh_filename, epsilon, function, output_filename):
    pass
```

Przyjmuje ona następujące parametry:

- `mesh_filename` to nazwa pliku wejściowego,
- `epsilon` to wartość epsilon użyta w algorytmie,
- `function` to jedna wartość z: center, mean, median, quadric i jest to metoda wyznaczania reprezentanta klastra,
- `output_filename` to nazwa pliku wyjściowego (koniecznie plik OFF).

Obsługiwane są dwa formaty plików wejściowych **pod warunkiem, że siatki w nich zawarte są rozmaitościami oraz są zorientowane poprawnie**:

- ***.off** - OFF geometry format - prosty format, pozwalający na przechowywanie jedynie informacji o współrzędnych wierzchołków oraz listy elementów utworzonych w oparciu o zdefiniowane wierzchołki. Na potrzeby zrealizowanego programu, plik

w formacie *.off należy wygenerować bez dodatkowej informacji o kolorze wierzchołków.

- ***.obj** - Wavefront .obj file - zaawansowany format, pozwalający przechowywać wiele dodatkowych danych opisujących siatkę. Zrealizowany program wykorzystuje jedynie informacje o współrzędnych wierzchołków oraz o face'ach utworzonych na ich podstawie.

4. Użycie programu

Aby uruchomić program, wystarczy wydać komendę:

```
python main.py <in_file> <epsilon> <representative> <out_file>
```

Gdzie:

- `in_file` to nazwa pliku wejściowego,
- `epsilon` to wartość epsilon użyta w algorytmie,
- `representative` to jedna wartość z: center, mean, median, quadric i jest to metoda wyznaczania reprezentanta klastra,
- `out_file` to nazwa pliku wyjściowego (koniecznie plik OFF).

Np.:

```
python main.py data/mesh_in.off 0.2 quadric out.off
```

Obsługiwane są dwa formaty plików wejsciowych **pod warunkiem, że siatki w nich zawarte są rozmaitościami oraz są zorientowane poprawnie**:

- ***.off** - OFF geometry format - prosty format, pozwalający na przechowywanie jedynie informacji o współrzędnych wierzchołków oraz listy elementów utworzonych w oparciu o zdefiniowane wierzchołki. Na potrzeby zrealizowanego programu, plik w formacie *.off należy wygenerować bez dodatkowej informacji o kolorze wierzchołków.
- ***.obj** - Wavefront .obj file - zaawansowany format, pozwalający przechowywać wiele dodatkowych danych opisujących siatkę. Zrealizowany program wykorzystuje jedynie informacje o współrzędnych wierzchołków oraz o face'ach utworzonych na ich podstawie.