Surface Visualization

Abgabe über die NextCloud bis 23:59 Uhr des o.g. Datums.

Aufgabe 1 Marching Squares (5 Punkte)

In `task6_1.py` sind Skalarwerte eines rechtwinkligen Gitters in der Matrix G gegeben. Die Koordinaten der Gitterpunkte befinden sich in X und Y. Implementieren Sie den Marching Squares Algorithmus, um für jede Gitterzelle den Start- und Endpunkt der Isolinie an der Stelle 0 zu finden (falls vorhanden). Plotten Sie die Liniensegmente, um die vollständige Isolinie darzustellen.

Hinweise:

- Sie müssen für jede Zelle die vier Kanten nach Nullstellen absuchen und diese richtig interpolieren. Es ist am einfachsten, die Liniensegmente bei gefundenen Nullstellen für jede Zelle einzeln zu plotten.
- Den komplexen Fall von mehr als zwei Nullstellen innerhalb einer Zelle brauchen Sie nicht zu behandeln.
- Das Ergebnis ähnelt den Darstellungen aus Aufgabe 3.

Aufgabe 2 Laplace-Glättung (8 Punkte)

In `task5_2.py` wird eine Kugelgeometrie mit VTK erstellt und ein zufälliges Rauschen addiert. Um dieses Rauschen zu minimieren sollen Glättungsfilter implementiert werden.

a) (5 Punkte)
Implementieren Sie einen iterativen Ansatz von Laplace Smoothing mit uniformen Gewichten auf der Mesh-Kopie `mesh_laplace`. Setzen Sie die Anzahl der Iterationen $k = 5$ und den Glättungsfaktor $\lambda = 0.5$. Achten Sie darauf, dass die Reihenfolge in der einzelne Vertices (Mesh-Knotenpunkte) abgearbeitet werden, keine Auswirkung auf das Endergebnis hat.

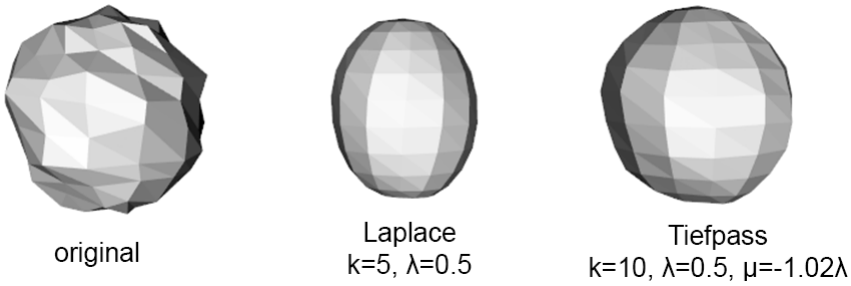
Hinweise:

- Ein Objekt `copy = vtk.vtkPolyData()` kann ein anderes PolyData-Objekt `mesh` vollständig kopieren durch `copy.DeepCopy(mesh)`.
- Die Vertices eines PolyData-Objekts sind nummeriert. Diese IDs gehen von 0 bis `mesh.GetNumberOfPoints()`.
- Verwenden Sie die bereitgestellte Methode `getConnectedVertices(mesh, seed_id)`, um die Nachbarpunkte eines Vertex mit der ID `seed_id` im PolyData-Objekt `mesh` zu erhalten. Die Methode gibt ein Python Set zurück, welches die IDs der Nachbarn (one-ring neighborhood) enthält.
- Die Koordinaten eines einzelnen Vertex erhalten Sie durch `mesh.GetPoint(point_ID)`.
- Ein einzelner Vertex kann modifiziert werden durch
  `mesh.GetPoints().SetPoint(point_ID, [x, y, z])`
  Die Koordinatenliste darf auch ein Numpy-Array sein.

b) (3 Punkte)
Implementieren Sie nun einen Tiefpassfilter, der die Laplace-Varianten *Shrink* und *Inflate* verwendet. Die beiden Varianten sollen sich in jeder Iteration abwechseln, beginnend mit *Shrink*. Insgesamt sollen $2k$ Iterationen erfolgen mit $\lambda = 0.5$ und $\mu = -1.02\lambda$. Das Ergebnis soll in der vorgegebenen Mesh-Kopie `mesh_low_pass` gespeichert werden, damit die drei Objekte (Original, Laplace und Tiefpass) verglichen werden können.

Vorgefertigte Glättungsfilter von VTK oder anderen Bibliotheken dürfen Sie natürlich *nicht* verwenden. Dies wird mit 0 Punkten bewertet.

Die Ergebnisse sehen so aus:



Aufgabe 3 Theorie: Marching Squares **(2 Punkte)**

Geben Sie die Antworten auf die Theorieaufgaben in der Multiple-Choice-Datei `MC06.txt` an. Es ist immer genau eine Auswahlmöglichkeit richtig. Bitte keine anderen Anmerkungen in diese Datei schreiben und den Dateinamen nicht verändern.

a) (2 Punkte)
Bei welcher der folgenden Darstellungen wurde der Marching Squares Algorithmus *korrekt* ausgeführt?