Принцип отбора аналогов.

Задача сжатия облака точек заключается в сокращении количества точек облака без потери геометрии и искажений. Поэтому, аналоги для рассмотрения были выбраны исходя из соответствия данной задаче.

Аналоги

1. 3D Point Cloud Simplification Based on k-Nearest Neighbor and Clustering

Данный алгоритм сначала разбивает исходное облако точек на кластеры используя метод k-ближайших соседей. Далее для всех полученных кластеров вычисляется энтропия Шеннона. После чего кластеры с наименьшей энтропией удаляются, пока не останется необходимое количество точек. В результате получается упрощенное облако точек. За счет того, что регулируется количество изначальных и удаленных кластеров, данный метод можно применять к различным формам облаков точек. Однако, кластеризация методом k-ближайших соседей является достаточно медленной, поэтому данный алгоритм на больших облаках точек будет осуществлять долгое сжатие.

Сложность.

$O(n \cdot k \cdot d \cdot i)$ – сложность метода k-ближайших соседей, где

n – количество точек размерности d $(d = 3)$.

k – итоговое количество кластеров.

d – размерность точек.

i - количество итераций.

$O(\sum_{i=1}^{k} N_i^2)$ – сложность расчета энтропии для всех кластеров, где

k – итоговое количество кластеров.

N_i – количество точек в i-ом кластере.

2. A Linear Programming Approach for 3D Point Cloud Simplification

Данный алгоритм сначала разбивает исходное облако точек на кластеры, используя EM-алгоритм. Далее задача удаления точек решается путем задачи минимизации функции в линейном программировании. В результате получается упрощенное облако точек. Данный алгоритм сохраняет границы фигур (граничные точки) облаков точек. Однако, недостатком является то, если входное облако точек имеет малую плотность точек, то при увеличении количества кластеров сжатое облако точек получается неоднородным.

Сложность.

$O(K \cdot M^2 \cdot N \cdot i)$ – сложность EM алгоритма, где

K – количество кластеров.

M – количество параметров точки.

N – количество точек.

i – заданное число итераций.

$O(N^2)$ – сложность оценки кривизны точек, где

N – количество точек.

$O(n^{2.5})$ – сложность решения задачи линейного программирования, где

n – количество переменных (кластеров)

3. Point cloud simplification with preserved edge based on normal vector

Данный алгоритм оставляет максимальное количество граничных точек в сжатом облаке, а неграничные точки удаляет итерационно до тех пор, пока не достигнет заданной степени сжатия. Преимущество заключается в том, что сохраняются граничные точки. Однако, при этом на участках с небольшой кривизной фигура получается неоднородной.

Сложность.

$O(\log(N^3))$ – сложность построения октодерева для облака точек

N – количество точек в облаке.

$O(N \log(N^3))$ – сложность поиска всех кравевых точек в октодереве

N – количество точек в облаке.

$O(N \cdot M^2)$ – сложность расчёта векторов нормали

N – количество точек.

M – размерность точки.

$O((N-M)\log(N^3))$ – сложность оценивания величины важности для некраевых точек

N – количество точек.

M – количество краевых точек.

4. A new point cloud simplification algorithm

В данном алгоритме строится диаграмма Вороного с помощью метода Fast marching. Во время ее построения при нахождении новой вершины Вороного время, за которое она была найдена и добавляется в кучу (max heap). Затем из кучи вынимается корень и вершина Вороного, которая ему соответствует попадает в итоговое облако точек. Алгоритм продолжается до тех пора, пока не будет получен необходимый размер облака точек, или не будет получена необходимая плотность. Данный алгоритм позволяет сделать сжатие облака точек с указанной плотностью.

Сложность.

$O(N\log(N))$ – общая сложность алгоритма

$O(\log(W))$ – сложность вставки и удаления из кучи. W – количество элементов в куче

$O(N\log(N))$ – сложность нахождения вершин диаграмм Вороного

5. Feature Preserving and Uniformity-controllable Point Cloud Simplification on Graph

Данный алгоритм производит сжатие облака точек, используя возможности обработки графических сигналов. В нем рассматриваются основные понятия теории спектральных графов, в том числе граф Лаплациана и граф сигнала. Данный алгоритм позволяет выбрать баланс

между сохранением граничных точек и поддержанием равномерной плотности или приоритет первого, или второго.

Сложность.

$O(M \cdot N^3)$ – сложность алгоритма

M – количество кубов, на которое разбито облако точек

N – количество точек в каждом кубу (от 3000 до 8000)

$O(n^3)$ – сложность решения задачи квадратического программирования для n переменных

Критерии сравнения аналогов

A. Возможность задавать плотность сжатия облака точек.

B. Возможность сжатия любого облака точек (от размера с кошку до размера с пятиэтажный дом).

C. Производится быстрое сжатие облака точек.

D. В упрощенном облаке точек сохраняются граничные точки.

E. Упрощенное облако точек получается однородным.

Альтернатива/Критерий	A	B	C	D	E
1	-	+	-	-	+
2	-	+	-	+	-
3	-	+	+	+	-
4	+	+	+	-	+
5	+	+	?	-+	-+

Выводы

Из таблицы видно, что задавать плотность можно только в двух алгоритмах. Причем один балансирует между сохранением граничных точек и однородности в упрощенном облаке, а у второго однородность сохраняется по умолчанию и не все могут терять граничные точки. В остальных алгоритмах нельзя задавать плотность и один из них выделяется тем, что он осуществляет быстрое сжатие.

Ссылки

1. https://www.hindawi.com/journals/am/2020/8825205/
2. https://www.researchgate.net/publication/313881735_A_Linear_Programming_Approach_for_3D_Point_Cloud_Simplification
3. https://www.sciencedirect.com/science/article/pii/S0030402615004052
4. https://www.researchgate.net/publication/2885227_A_New_Point_Cloud_Simplification_Algorithm
5. https://www.researchgate.net/publication/330037178_Feature_Preserving_and_Uniformity-controllable_Point_Cloud_Simplification_on_Graph