Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Кластеризация данных LiDAR

Студент ___ИУ5-61Б__ _____ __**Бондаренко Д. К.**__
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ _____**Канев А. И.**___
(Подпись, дата) (И.О.Фамилия)

Москва, 2023 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ___ИУ5___
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме _____Кластеризация данных LiDAR_____

_____

_____

Студент группы ____ИУ5-61Б____

_____Бондаренко Денис Константинович _____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____Исследовательская_____

Источник тематики (кафедра, предприятие, НИР) _____НИР_____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание ***Исследовать использование методов машинного обучения для решения задачи сегментации деревьев из облака точек***

_____

_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _29_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

Дата выдачи задания «07» февраля 2023 г.

Руководитель НИР _____ **Канев А.И.**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **Бондаренко Д.К.**
(Подпись, дата) (И.О.Фамилия)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

LiDAR – технология, позволяющая определять удаленность какой-либо точки от точки съемки путем расчета расстояния с использованием измеренного времени, которое необходимо испущенному лидаром сигналу для достижения этой точки и возвращения на приемник лидара после отражения. Лидары используют в системах машинного зрения, системах подводного зрения, в строительстве и горном деле, в исследовании ландшафтов, управлении лесным хозяйством и многих других сферах деятельности человека.

Результатом съемки лидара является облако точек, каждая из которых соответствуют тому или иному объекту и их элементам. Эти объекты часто нужно отделить друг от друга в тех или иных целях, объединив точки, относящиеся к одному и тому же объекту, в группы. Такая задача называется задачей сегментации облака точек. Она особенно актуальна сейчас, в то время, когда развиваются беспилотные автомобили, системы помощи водителю, системы картографии и другие системы, связанные с топографией и в которых требуется разграничивать объекты друг от друга.

Целью данной работы является исследование применения методов машинного обучения для сегментации деревьев из облака точек. В данной работе решаются задача сегментации деревьев из облака точек с использованием методов машинного обучения, задача оценки качества сегментации деревьев из облака точек.

1 Постановка задачи

Имеется множество точек, которое получено в результате съемки лидаром нескольких деревьев. Необходимо произвести сегментацию деревьев из облака точек, то есть необходимо отнести точки, относящиеся к одному и тому же дереву, к одному классу.

Решение этой задачи может быть полезным для получения пригодных для анализа данных, которые позволят усовершенствовать практики ведения лесного хозяйства. Коммерческим работникам фруктовых садов важно понимать динамику роста деревьев. Проведение измерений вручную может быть трудоемким, может занимать много времени или может быть затруднено погодными условиями. Использование технологии LiDAR является эффективным решением этой проблемы, так как технология LiDAR активно развивается, удобна для получения больших объемов данных и легко автоматизируема [1] [5].

Решение задачи сегментации деревьев из облака точек может быть полезным и для отслеживания влияния антропогенных факторов на деревья. После сегментации деревьев из облака точек можно отслеживать изменения отдельных деревьев в ответ на такие явления, как заболевания деревьев, биологические инвазии, засухи, получая не только конкретные данные об изменении состояния отдельных деревьев, но и общее понимание влияния вышеперечисленных явлений на деревья [2].

Знание местоположения деревьев и их качеств важно для коммерческих лесоводов, так как оно позволит им расходовать меньшее количество ресурсов при лесозаготовке и оптимальнее продумывать развертывание цепей поставок. Сегментирование деревьев из облаков точек, полученных при съемке высокоточным воздушным лидаром, является подходящей опцией для получения коммерческими лесоводами такого знания [3].

Решение рассматриваемой задачи может быть использовано для исследования лесных ресурсов. Беспилотные летательные аппараты, оборудованные технологией LiDAR, играют важную роль в таких

исследованиях. Сегментация деревьев из облака точек, полученных в результате съемки лесных массивов такими аппаратами, может помочь получить информацию об отдельных деревьях, которая даст понимание текущей ситуации, связанной с лесными ресурсами и экологическими выгодами, которые они предоставляют, а также понимание того, как эта ситуация развивается [4].

Задача сегментации деревьев из облака точек может быть решена различными методами. В качестве исходных данных в статье [1] для сегментации деревьев из облака точек используют данные, полученные при сканировании деревьев ручным лидаром. Для сегментации используется метод, основанный на графах. Точки из облака помещаются в воксели заданного размера, а узлы графа определяются средним положением всех точек в вокселе.

Метод включает в себя обход облака точек по графу: отслеживаются пути от каждого узла ствола до каждого узла в графе. Каждый узел дерева, который был достигнут по некоторому пути, соотносится узлу ствола, который породил этот путь. После того, как все узлы соотнесены соответствующим узлам ствола, сегментация распространяется на все точки, принадлежащие соответствующим вокселям. Для оценки качества сегментации использовалась метрика v-measure: в среднем удалось достичь показателя, равного 0,915.

Данные для анализа в статье [2] были взяты с низменного заповедника тропических лесов в Сабахе. Для сегментации использовался алгоритм MCGC (Multi-Class Graph Cut). Перед его применением данные были предобработаны путем очищения от шумов с помощью пакета LASTools. Сначала по алгоритму MCGC на основе координат точек формировалось графовое представление облака точек. Затем с использованием информации о геометрии облака точек и локальных плотностей их расположения для каждой точки рассчитывался вектор центроида скопления точек, к которому она принадлежит. По схожести направления векторов центроида определялась принадлежность двух точек к одному кластеру. Для оценки качества сегментации использовалась метрика

DBH (Diameter at Breast Height), которая была заготовлена на основе набора данных Sepilok.

Данные статьи [4] были взяты с различных лесных полигонов площадью 0,067 га с использованием технологии LiDAR. Для сегментации деревьев из облака точек использовались различные методы, одним из которых был метод Layer Stacking. Облако точек разбивалось на слои, в каждом из которых применялся метод кластеризации K-means. После этого в каждом слое строились полигоны Тиссена. При наложении друг на друга полигонов различных слоев образовывались области плотного перекрытия, которые свидетельствовали о наличии отдельного дерева в области перекрытия. Качество сегментации оценивалось с использованием таких метрик, как precision, recall и F1 score. Значение recall не было ниже 78,95%, значение precision не было ниже 71,43%, значение F1 score не было ниже 0,765.

2 Описание данных, используемых методов и метрик

Используемые данные представляют собой двумерный массив координат 10000 точек (Рисунок 2.1), полученных в результате съемки лидаром трех близких друг к другу деревьев (вариант 7). Точки, принадлежащие стволу, распределены с более высокой плотностью, чем точки, принадлежащие листве деревьев, что может сказаться на точности методов машинного обучения.

Рисунок 2.1 – облако точек

Данные будут исследоваться методом DBSCAN – плотностным алгоритмом пространственной кластеризации с присутствием шума. Для работы DBSCAN необходимо выбрать 2 параметра – радиус ε-окрестности eps и количество соседей min_samples. Точка будет считаться корневой, если в ее ε-окрестности будет находиться не менее min_samples соседей. Алгоритм выбирает из датасета какую-либо корневую точку и помещает ее соседей в список обхода. Если сосед также является корневым, то и его соседи включаются в список обхода, иначе – нет. Таким образом происходит образование кластеров точек, которое будет продолжаться до тех пор, пока обход не пройдет по всем точкам. Те точки, которые не являются корневыми и которые в ε-окрестности не содержат ни одного корневого соседа, определяются методом как выбросы.

Использование DBSCAN для данных большой размерности неэффективно. Наши данные таковыми не являются, поэтому DBSCAN подойдет для их исследования. Для оценки качества кластеризации будет использоваться популярная для использования с этой целью метрика Silhouette Score. Метрика Silhouette Score является внутренней метрикой оценки качества кластеризации. Эта метрика подойдет для работы с нашими данными, так как она не требует внешней информации о наборе данных и производит оценку только на основе исходных данных и результатах проведения кластеризации.

3 Сегментация деревьев из облака точек

С использованием метода DBSCAN и функций для представления набора точек в формате pcd, сегментации, графического отображения облака точек и присваивания точкам цветов была произведена сегментация облака точек и его отображение для параметров eps = 0,5 и min_samples = 240 (Рисунок 3.1). Результат оказался неудовлетворительным, так как количество кластеров и качество сегментации не были приемлемы: было распознано 2 объекта.

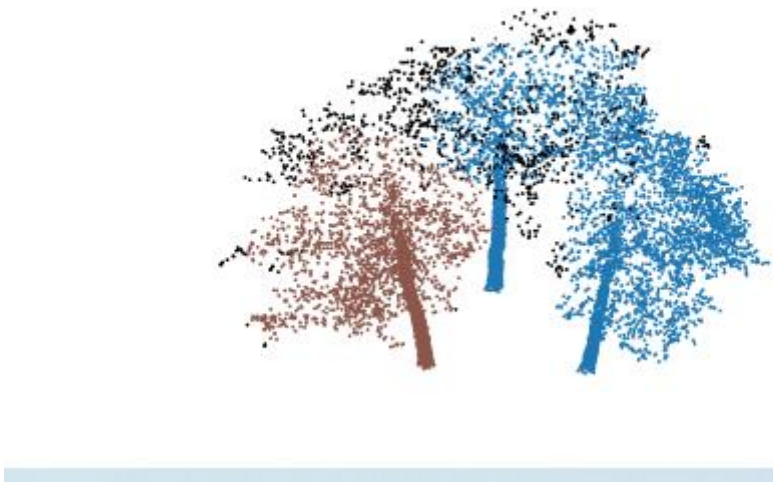

Рисунок 3.1 – результат сегментации при eps = 0,5, min_samples = 240

Было решено для каждого eps от 0,02 до 0,98 с шагом 0,02 и каждого min_samples от 2 до 298 с шагом 2 определить количество кластеров. Был создан датафрейм, который ставит каждой паре eps и min_samples в соответствие количество кластеров + 1 (так как подсчет ведется по заголовкам кластеров точек, а заголовок точек, относящихся к шумам, также подсчитывается) (Рисунок 3.2, Рисунок 3.3).

```
Ввод [46]: %%time
eps_search_values = np.arange(0.02, 1, 0.02)
min_pts_search_values = np.arange(2, 300, 2)
segm_pars = list(product(eps_search_values, min_pts_search_values))
# segm_pars
checked_eps = []
checked_min_pts = []
clusters_cnt = []
for i in segm_pars:
    search_clustering = DBSCAN(eps=i[0], min_samples=i[1], algorithm='ball_tree').fit(X)
    checked_eps.append(i[0])
    checked_min_pts.append(i[1])
    clusters_cnt.append(len(np.unique(search_clustering.labels_)))
zipped_data = list(zip(checked_eps, checked_min_pts, clusters_cnt))
check_df = pd.DataFrame(zipped_data, columns=['eps', 'min_pts', 'clusters_cnt'])
```

Рисунок 3.2 – код создания датафрейма подсчета кластеров

```
Ввод [58]: pd.set_option("display.max_rows", None, "display.max_columns", None)
           check_df
```

Out[58]:

	eps	min_pts	clusters_cnt
0	0.02	2	1038
1	0.02	4	205
2	0.02	6	53
3	0.02	8	8
4	0.02	10	3
5	0.02	12	2
6	0.02	14	1
7	0.02	16	1
8	0.02	18	1
9	0.02	20	1
10	0.02	22	1

Рисунок 3.3 – датафрейм подсчета кластеров

Из всех записей были отобраны только те, clusters_cnt которых оказался равен 4 (Рисунок 3.4).

```
Ввод [59]: check_df_4 = check_df[check_df['clusters_cnt']==4]
           check_df_4
```

Out[59]:

	eps	min_pts	clusters_cnt
317	0.06	40	4
318	0.06	42	4
467	0.08	42	4
480	0.08	68	4
481	0.08	70	4
631	0.10	72	4
632	0.10	74	4
633	0.10	76	4
634	0.10	78	4
643	0.10	96	4
644	0.10	98	4

Рисунок 3.4 – отфильтрованный датафрейм

При малых eps и min_samples (min_pts) кластеры содержат малое количество тесно сгруппированных точек, подавляющее большинство точек определяется как шумы (Рисунок 3.5).

```
Ввод [60]: # настройка основных параметров модели (эксп)
           # eps - размер окрестности точки
           # min_pts - минимальное кол-во точек в окрестности

           eps_exp = 0.06
           min_pts_exp = 40

Ввод [61]: new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps_exp, min_pts_exp)
           print(f"Распознано {max_label} объектов в облаке точек")

           Распознано 3 объектов в облаке точек

Ввод [62]: points = np.asarray(pcd.points)
           new_pcd, colors = add_color(new_pcd, colors)

           draw_plot(points, colors)
```

Рисунок 3.5 – результат сегментации при малых eps и min_samples

При увеличении eps и min_samples, все еще остающихся малыми, кластеры становились больше и находились в области стволов деревьев, так как точки там расположены наиболее плотно. Листва деревьев же распознавалась как выбросы (Рисунок 3.6, Рисунок 3.7).

```
Ввод [69]: # настройка основных параметров модели (эксп)
           # eps - размер окрестности точки
           # min_pts - минимальное кол-во точек в окрестности

           eps_exp = 0.14
           min_pts_exp = 144

Ввод [70]: new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps_exp, min_pts_exp)
           print(f"Распознано {max_label} объектов в облаке точек")

           Распознано 3 объектов в облаке точек

Ввод [71]: points = np.asarray(pcd.points)
           new_pcd, colors = add_color(new_pcd, colors)

           draw_plot(points, colors)
```

Рисунок 3.6 – результаты сегментации при eps = 0,14, min_samples = 144

```
# eps - размер окрестности точки
# min_pts - минимальное кол-во точек в окрестности

eps_exp = 0.16
min_pts_exp = 58
```

Ввод [76]:
```
new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps_exp, min_pts_exp)
print(f"Распознано {max_label} объектов в облаке точек")
```

```
Распознано 3 объектов в облаке точек
```

Ввод [77]:
```
points = np.asarray(pcd.points)
new_pcd, colors = add_color(new_pcd, colors)

draw_plot(points, colors)
```

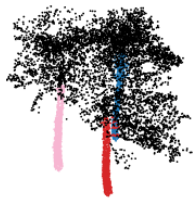

Рисунок 3.7 – результаты сегментации при eps = 0,16, min_samples = 58

При eps=0,3 и min_samples=80 наблюдается более точная кластеризация, меньшее количество точек, соответствующих листве, определилось как выбросы. Однако при большем значении min_samples для данного показателя eps либо количество обнаруженных объектов не равно 3, либо в кластеры входят только точки стволов (Рисунок 3.8).

Ввод [102]:
```
# настройка основных параметров модели (эксп)
# eps - размер окрестности точки
# min_pts - минимальное кол-во точек в окрестности

eps_exp = 0.3
min_pts_exp = 80
```

Ввод [103]:
```
new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps_exp, min_pts_e
print(f"Распознано {max_label} объектов в облаке точек")
```

```
Распознано 3 объектов в облаке точек
```

Ввод [104]:
```
points = np.asarray(pcd.points)
new_pcd, colors = add_color(new_pcd, colors)

draw_plot(points, colors)
```

Рисунок 3.8 – результаты сегментации при eps = 0,3, min_samples = 80

Наилучшие результаты были достигнуты при eps=0,5 и min_samples=276 (Рисунок 3.9) и при eps=0,315 и min_samples=92 (Рисунок 3.10) (найденных при отдельном рассмотрении диапазона eps от 0,29 до 0,32), но в первом случае количество выбросов меньше, так как параметр eps больше.

```
Ввод [213]:
# настройка основных параметров модели (эксп)
# eps - размер окрестности точки
# min_pts - минимальное кол-во точек в окрестности

eps_exp = 0.5
min_pts_exp = 276

Ввод [214]:
new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps_exp, min_pts_exp)
print(f"Распознано {max_label} объектов в облаке точек")

Распознано 3 объектов в облаке точек

Ввод [215]:
points = np.asarray(pcd.points)
new_pcd, colors = add_color(new_pcd, colors)

draw_plot(points, colors)
```

Рисунок 3.9 – результаты сегментации при eps = 0,5, min_samples = 276

```
Ввод [458]:
# настройка основных параметров модели (эксп)
# eps - размер окрестности точки
# min_pts - минимальное кол-во точек в окрестности

eps_exp = 0.315
min_pts_exp = 92

Ввод [459]:
new_pcd, colors, max_label, obj_points = segment_pcd(X, pcd, points, eps_exp, min_pts_exp)
print(f"Распознано {max_label} объектов в облаке точек")

Распознано 3 объектов в облаке точек

Ввод [460]:
points = np.asarray(pcd.points)
new_pcd, colors = add_color(new_pcd, colors)

draw_plot(points, colors)
```

Рисунок 3.10 – результаты сегментации при eps = 0,315, min_samples = 92

Были найдены значения метрики Silhouette Score для каждой пары eps и min_samples для небольшого интервала значений eps, содержащего eps с наилучшими результатами (в первом случае eps от 0,29 до 0,32 с шагом 0,001 и min_samples от 50 до 150 с шагом 1, во втором – eps от 0,48 до 0,51 с шагом 0,005 и min_samples от 200 до 319 с шагом 1) (Рисунок 3.11, Рисунок 3.12, Рисунок 3.13, Рисунок 3.14). Наилучшие показатели были достигнуты при eps = 0,32 и min_samples = 92 (sil_score = 0,282503) и при eps = 0,485 и min_samples = 256 (sil_score = 0,287403). Значение метрики для параметров eps и min_samples, найденных ранее, рознится со значением метрики для новых параметров в тысячных долях. Отображение сегментированных облаков точек также не претерпевает значимых изменений.

Ввод [17]:
```
%%time
eps_search_values_sil1 = np.arange(0.29, 0.32, 0.001)
min_pts_search_values_sil1 = np.arange(50, 151, 1)
segm_pars_sil1 = list(product(eps_search_values_sil1, min_pts_search_values_sil1))
# segm_pars_sil1
checked_eps_sil1 = []
checked_min_pts_sil1 = []
clusters_cnt_sil1 = []
sil_score1 = []
for i in segm_pars_sil1:
    search_clustering_sil1 = DBSCAN(eps=i[0], min_samples=i[1], algorithm='ball_tree').fit(X)
    checked_eps_sil1.append(i[0])
    checked_min_pts_sil1.append(i[1])
    clusters_cnt_sil1.append(len(np.unique(search_clustering_sil1.labels_)))
    sil_score1.append(metrics.silhouette_score(X, search_clustering_sil1.labels_))
zipped_data_sil1 = list(zip(checked_eps_sil1, checked_min_pts_sil1, clusters_cnt_sil1, sil_score1))
check_df_sil1 = pd.DataFrame(zipped_data_sil1, columns=['eps', 'min_pts', 'clusters_cnt', 'sil_score'])
check_df_sil1
```

```
Wall time: 1h 55min 53s
```

Out[17]:

	eps	min_pts	clusters_cnt	sil_score
0	0.29	50	5	0.008541
1	0.29	51	5	0.008870
2	0.29	52	5	0.004174
3	0.29	53	6	-0.054901
4	0.29	54	6	-0.061995
...	...	...	...	...
3126	0.32	146	7	0.127582
3127	0.32	147	8	0.061288

Рисунок 3.11 - код создания датафрейма 1 подсчета значения метрики

```
Ввод [19]: check_df_4_sil1 = check_df_sil1[check_df_sil1['clusters_cnt']==4]
           check_df_4_sil1
```

3044	0.320	64	4	0.155156
3045	0.320	65	4	0.155350
3046	0.320	66	4	0.154824
3047	0.320	67	4	0.154457
3050	0.320	70	4	0.119331
3056	0.320	76	4	0.041061
3057	0.320	77	4	0.042155
3058	0.320	78	4	0.042028
3059	0.320	79	4	0.042004
3060	0.320	80	4	0.039205
3072	0.320	92	4	0.282503
3073	0.320	93	4	0.281828
3074	0.320	94	4	0.281367

Рисунок 3.12 – отфильтрованный датафрейм 1 подсчета значения метрики

```
Ввод [29]: %%time
eps_search_values_sil2 = np.arange(0.48, 0.51, 0.005)
min_pts_search_values_sil2 = np.arange(200, 320, 1)
segm_pars_sil2 = list(product(eps_search_values_sil2, min_pts_search_values_sil2))
# segm_pars_sil2
checked_eps_sil2 = []
checked_min_pts_sil2 = []
clusters_cnt_sil2 = []
sil_score2 = []
for i in segm_pars_sil2:
    search_clustering_sil2 = DBSCAN(eps=i[0], min_samples=i[1], algorithm='ball_tree').fit(X)
    checked_eps_sil2.append(i[0])
    checked_min_pts_sil2.append(i[1])
    clusters_cnt_sil2.append(len(np.unique(search_clustering_sil2.labels_)))
    sil_score2.append(metrics.silhouette_score(X, search_clustering_sil2.labels_))
zipped_data_sil2 = list(zip(checked_eps_sil2, checked_min_pts_sil2, clusters_cnt_sil2, sil_score2))
check_df_sil2 = pd.DataFrame(zipped_data_sil2, columns=['eps', 'min_pts', 'clusters_cnt', 'sil_score'])
check_df_sil2
```

550	0.500	270	3	0.212141
551	0.500	271	3	0.212544
552	0.500	272	3	0.212544
553	0.500	273	3	0.212593
554	0.500	274	3	0.212813
555	0.500	275	3	0.212055
556	0.500	276	4	0.286697
557	0.500	277	4	0.286864
558	0.500	278	5	0.236950
559	0.500	279	5	0.236134
560	0.500	280	5	0.236502
561	0.500	281	5	0.236029
562	0.500	282	5	0.237077

Рисунок 3.13 – код создания датафрейма 2 подсчета значения метрики

```
Ввод [30]: check_df_4_sil2 = check_df_sil2[check_df_sil2['clusters_cnt']==4]
           check_df_4_sil2
```

Out[30]:

	eps	min_pts	clusters_cnt	sil_score
48	0.480	248	4	0.285982
49	0.480	249	4	0.286075
50	0.480	250	4	0.285627
51	0.480	251	4	0.285530
175	0.485	255	4	0.286785
176	0.485	256	4	0.287043
177	0.485	257	4	0.286984
299	0.490	259	4	0.286766
300	0.490	260	4	0.287014
301	0.490	261	4	0.287010
429	0.495	269	4	0.286708
430	0.495	270	4	0.286873
431	0.495	271	4	0.286750
432	0.495	272	4	0.286793
556	0.500	276	4	0.286697
557	0.500	277	4	0.286864
681	0.505	281	4	0.259682
682	0.505	282	4	0.260026
806	0.510	286	4	0.259631
807	0.510	287	4	0.259314
808	0.510	288	4	0.259985

Рисунок 3.14 - отфильтрованный датафрейм 2 подсчета значения метрики

4 Сегментация облака точек большой размерности

Была произведена сегментация облака точек большой размерности. Используемые данные представляют собой двумерный массив координат 39569319 точек (Рисунок 4.1), полученных в результате съемки лидаром множества деревьев.

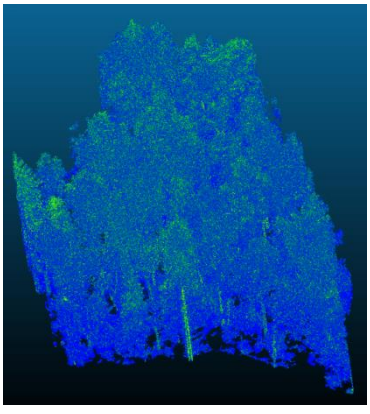

Рисунок 4.1 – облако точек большой размерности

В силу высокой ресурсоемкости алгоритма DBSCAN для сегментации облака, содержащего большое количество точек, перед сегментацией производилось прореживание облака (Рисунок 4.2, Рисунок 4.3).

```
In [8]: # снижение размерности облака точек до ~num_points
        def down_pcd(data, num_points):
            if data.shape[0] > num_points:
                factor = data.shape[0] // num_points
            else:
                factor = 1
            down_data = data[::factor]
            return down_data

        # down_data = down_pcd(data, 40000)
```

Рисунок 4.2 – код прореживания облака точек

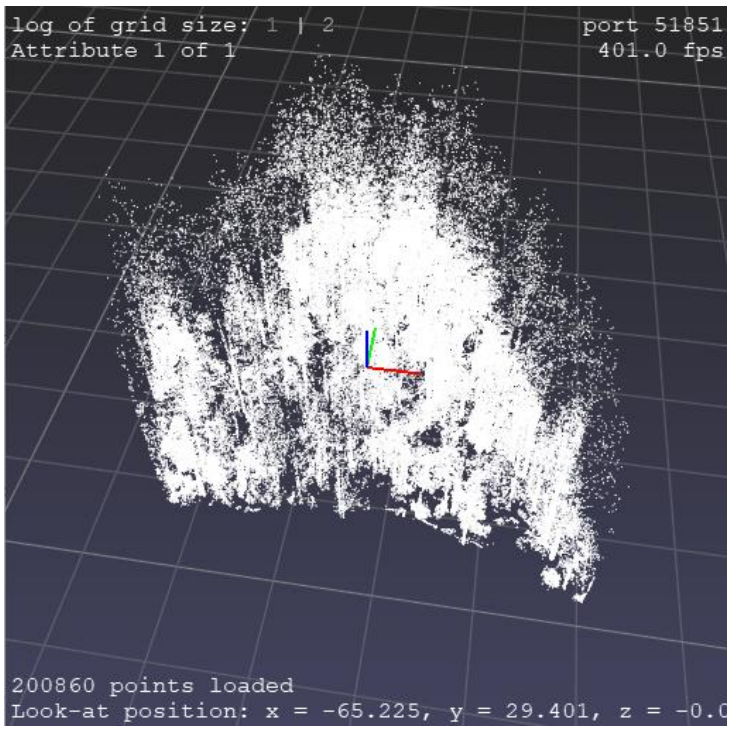

Рисунок 4.3 – прореженное облако точек

Использование найденных ранее параметров eps и min_samples, оптимальных для сегментации предыдущего облака точек, содержащего в себе 3 дерева, оказалось неэффективным для сегментации другого облака точек: почти все точки были определены как шум (Рисунок 4.4).

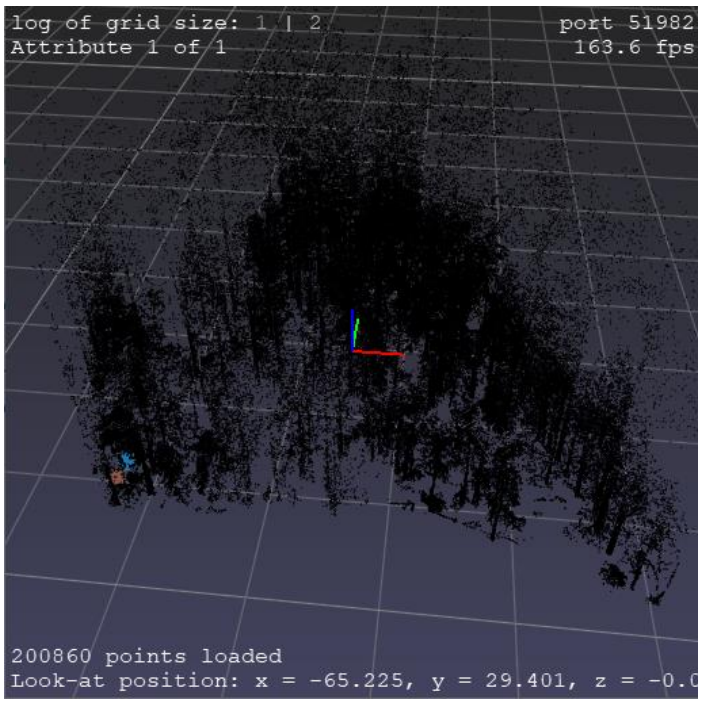

Рисунок 4.4 – результат сегментации при eps = 0,485, min_samples = 256

Было решено для каждого eps от 0,02 до 0,8 с шагом 0,04 и каждого min_samples от 2 до 300 с шагом 4 определить количество кластеров. Был создан датафрейм, который ставит каждой паре eps и min_samples в соответствие количество кластеров + 1 (так как подсчет ведется по заголовкам кластеров точек, а заголовок точек, относящихся к шумам, также подсчитывается) (Рисунок 4.5).

```
In [27]: %%time
eps_search_values3 = np.arange(0.02, 0.8, 0.04)
# eps_search_values = [0.5]
min_pts_search_values3 = np.arange(2, 300, 4)
# min_pts_search_values3 = [120]
segm_pars3 = list(product(eps_search_values3, min_pts_search_values3))
# segm_pars
checked_eps3 = []
checked_min_pts3 = []
clusters_cnt3 = []
for i in segm_pars3:
    search_clustering3 = DBSCAN(eps=i[0], min_samples=i[1], algorithm='ball_tree').fit(X)
    checked_eps3.append(i[0])
    checked_min_pts3.append(i[1])
    clusters_cnt3.append(len(np.unique(search_clustering3.labels_)))
zipped_data3 = list(zip(checked_eps3, checked_min_pts3, clusters_cnt3))
check_df3 = pd.DataFrame(zipped_data3, columns=['eps', 'min_pts', 'clusters_cnt'])
check_df3
```

1011	0.54	146	36
1012	0.54	150	37
1013	0.54	154	36
1014	0.54	158	40
1015	0.54	162	40
1016	0.54	166	36
1017	0.54	170	37
1018	0.54	174	36
1019	0.54	178	35
1020	0.54	182	33
1021	0.54	186	34
1022	0.54	190	33
1023	0.54	194	29

Рисунок 4.5 – датафрейм и код подсчета количества кластеров

При малых eps подавляющее большинство точек определяется как шумы, при средних eps с ростом min_samples кластеров становится все меньше, и они в основном локализуются в нижней части облака, так как плотность точек там выше (Рисунок 4.6).

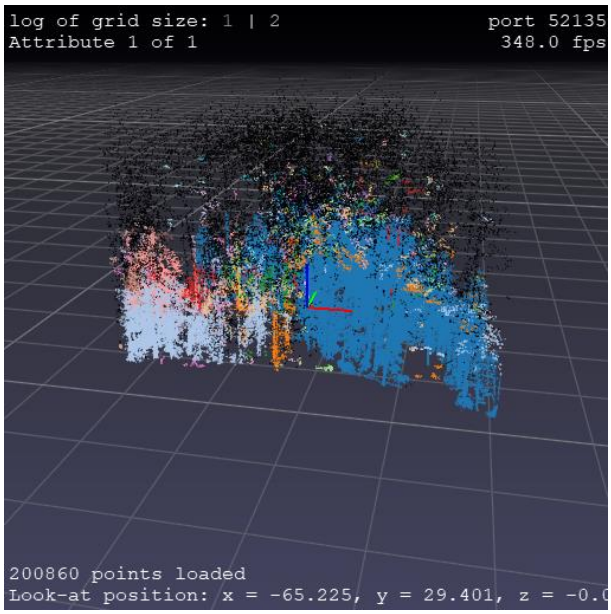

Рисунок 4.6 – результат сегментации при eps = 0,6, min_samples = 10

Было решено проредить облако точек таким образом, чтобы точки, находящиеся ниже 8 метров, были прорежены с большей интенсивностью, чем точки, находящиеся выше высоты в 8 метров. Также было решено для каждого eps от 0,2 до 0,8 с шагом 0,05 и каждого min_samples от 5 до 300 с шагом 10 определить количество кластеров и количество точек, распознаваемых как шум. Был создан датафрейм, который ставит каждой паре eps и min_samples в соответствие количество кластеров + 1 (так как подсчет ведется по заголовкам кластеров точек, а заголовок точек, относящихся к шумам, также подсчитывается) и количество точек, распознаваемых как шум (Рисунок 4.7).

Out[31]:

	eps	min_pts	clusters_cnt	noisy_cnt
0	0.2	5	6262	110729
1	0.2	15	385	191864
2	0.2	25	17	201228
3	0.2	35	4	201827
4	0.2	45	1	201946
...	...	...	...	...
385	0.8	255	7	199772
386	0.8	265	4	200830
387	0.8	275	2	201515
388	0.8	285	2	201525
389	0.8	295	2	201530

Рисунок 4.7 – датафрейм подсчета кластеров и “шумных” точек

Были произведены сегментации с параметрами, которым соответствуют оптимальные соотношения количества кластеров и “шумных” точек. Лучшего качества сегментации удалось достичь при eps = 0.63, min_samples = 10: наблюдается наличие отдельных стволов с кронами у части деревьев, однако большая часть деревьев либо входит в один кластер с ближайшими, либо сегментирована не полностью (Рисунок 4.8).

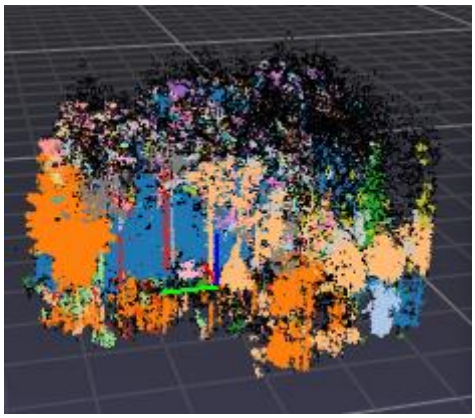

Рисунок 4.8 – результат сегментации при eps = 0,63, min_samples = 10

При изменении границы прореживания плотность точек заметно варьируется на разной высоте, что сказывается на качестве кластеризации (Рисунок 4.9).

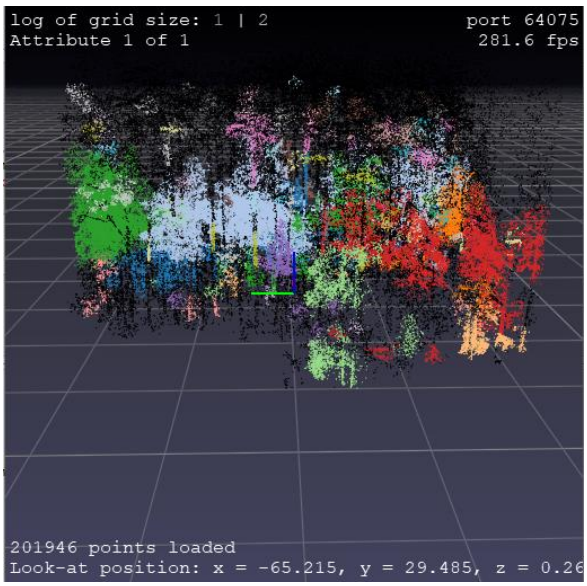

Рисунок 4.9 – результат при изменении границы прореживания

С целью повысить качество кластеризации было принято решение прореживать облако с интенсивностью, которая зависит от высоты расположения точки: чем выше находится точка, тем меньше шанс того, что она будет исключена (Рисунок 4.10, Рисунок 4.11).

```
In [9]: # снижение размерности облака точек с учетом высоты расположения точек
def down_pcd2(data, num_points):
    if data.shape[0] > num_points:
        factor = data.shape[0] // num_points
    else:
        factor = 1
    down_data = data[::factor]

    z_list = [d[2] for d in down_data]
    z_max = max(z_list)

    res_data = []
    for point in down_data:
        if z_max - point[2] < random.random() * 30:
            res_data.append(point)
    print(f'amount of points: {len(res_data)}')

    return np.asarray(res_data, dtype = np.float32)

# down_data = down_pcd(data, 40000)
```

Рисунок 4.10 – код прореживания облака с учетом высоты

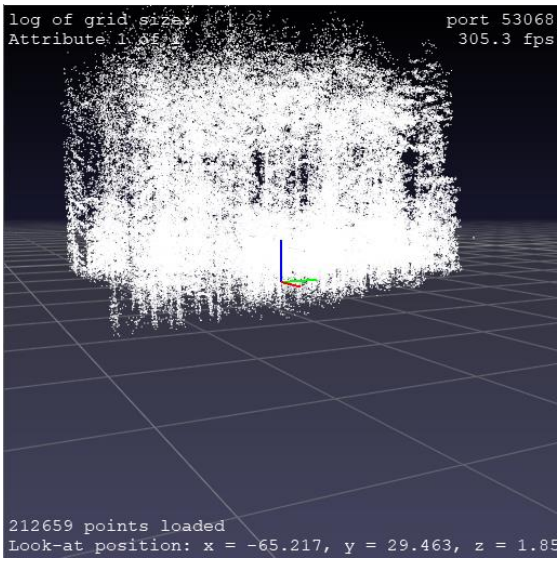

Рисунок 4.11 – прореженное облако

Внесенные изменения не оказали ожидаемого влияния на качество кластеризации: отдельных деревьев больше не стало, а кластер, который включает в себя большое множество деревьев, стал более явным (Рисунок 4.12, Рисунок 4.13, Рисунок 4.14).

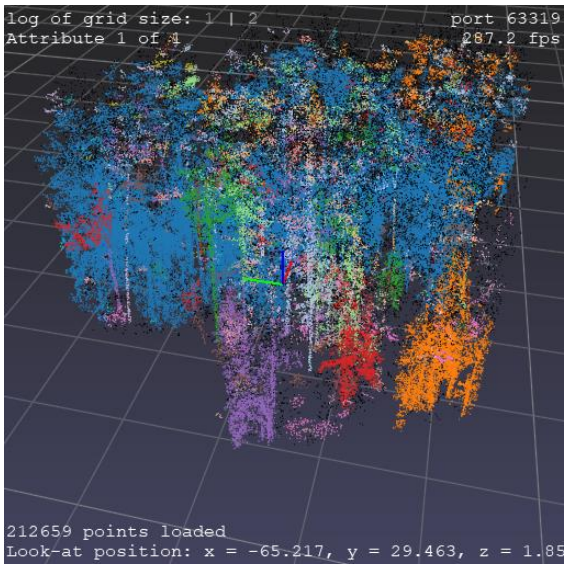

Рисунок 4.12 – результат сегментации при eps = 0,6, min_samples = 10

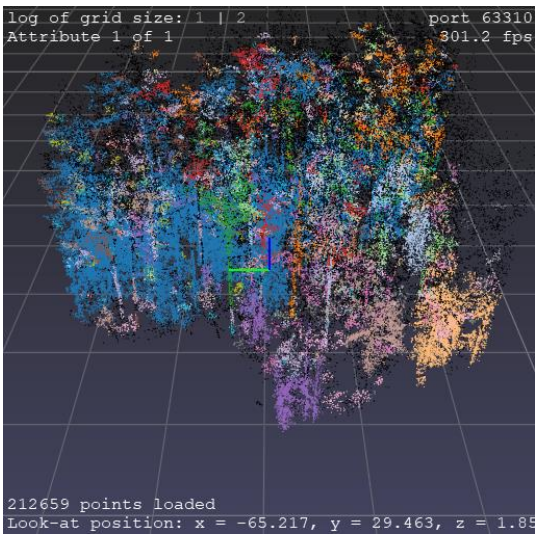

Рисунок 4.13 – результат сегментации при eps = 0,6, min_samples = 15

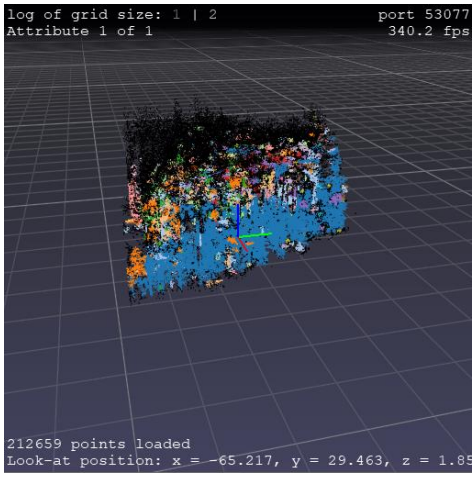

Рисунок 4.14 – результат сегментации при eps = 0,6, min_samples = 20

Было решено воспользоваться методом для нахождения оптимального eps для проведения сегментации. Метод заключается в нахождении “локтя” на графике, который можно построить с использованием библиотеки kneed, и определении eps, соответствующего этому “локтю”. Было построено несколько графиков, каждый из которых строился с учетом среднего расстояния между ближайшими n соседями (Рисунок 4.15, Рисунок 4.16, Рисунок 4.17).

```
In [49]: n_neighbors=30
nearest_neighbors = NearestNeighbors(n_neighbors=n_neighbors+1)
neighbors = nearest_neighbors.fit(X)
distances, indices = neighbors.kneighbors(X)
distances = np.sort(distances[:,n_neighbors], axis=0)

i=np.arange(len(distances))
knee = KneeLocator(i, distances, S=1, curve='convex', direction='increasing', interp_method='polynomial')
fig = plt.figure(figsize=(20, 10))
knee.plot_knee()
plt.xlabel("Points")
plt.ylabel("Distance")

print(distances[knee.knee])
#grad

1.1578496386337689

<Figure size 2000x1000 with 0 Axes>
```

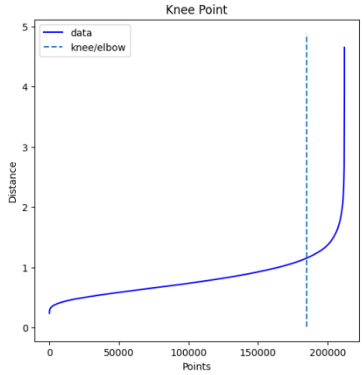

Рисунок 4.15 – код и график для n = 30

```
In [20]: n_neighbors=50
nearest_neighbors = NearestNeighbors(n_neighbors=n_neighbors+1)
neighbors = nearest_neighbors.fit(X)
distances, indices = neighbors.kneighbors(X)
distances = np.sort(distances[:,n_neighbors], axis=0)

i=np.arange(len(distances))
knee = KneeLocator(i, distances, S=1, curve='convex', direction='increasing', interp_method='polynomial')
fig = plt.figure(figsize=(20, 10))
knee.plot_knee()
plt.xlabel("Points")
plt.ylabel("Distance")

print(distances[knee.knee])
#grad

1.428256934974196

<Figure size 2000x1000 with 0 Axes>
```

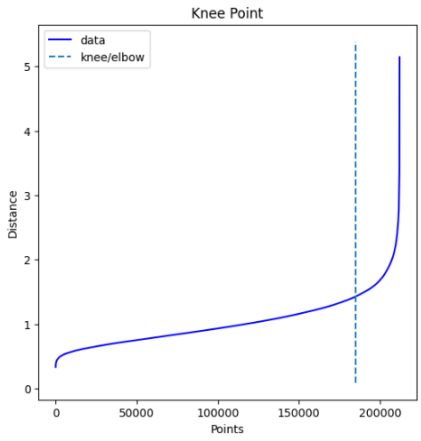

Рисунок 4.16 – код и график для n = 50

```
In [65]: n_neighbors=80
nearest_neighbors = NearestNeighbors(n_neighbors=n_neighbors+1)
neighbors = nearest_neighbors.fit(X)
distances, indices = neighbors.kneighbors(X)
distances = np.sort(distances[:,n_neighbors], axis=0)

i=np.arange(len(distances))
knee = KneeLocator(i, distances, S=1, curve='convex', direction='increasing', interp_method='polynomial')
fig = plt.figure(figsize=(20, 10))
knee.plot_knee()
plt.xlabel("Points")
plt.ylabel("Distance")

print(distances[knee.knee])
#grad

1.7122821171022493

<Figure size 2000x1000 with 0 Axes>
```

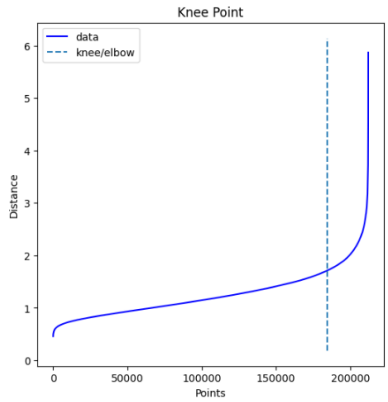

Рисунок 4.17 – код и график для n = 80

Для каждого из найденных eps была произведена сегментация с различными значениями min_samples, причем при малых min_samples почти все точки входят в один кластер. С ростом min_samples количество кластеров и “шумных” точек растет, наблюдаются почти полностью сегментированные деревья, но большинство все же либо сегментированы частично, либо входят в чужой кластер (Рисунок 4.18 – Рисунок 4.22).

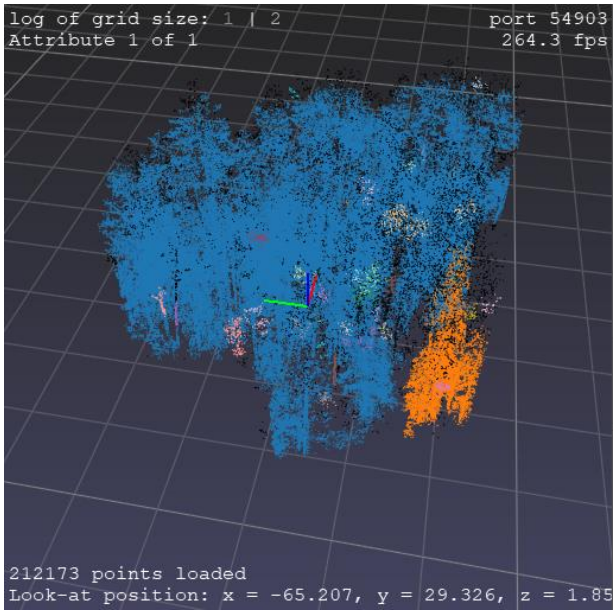

Рисунок 4.18 – результат сегментации при eps = 1,158, min_samples = 30

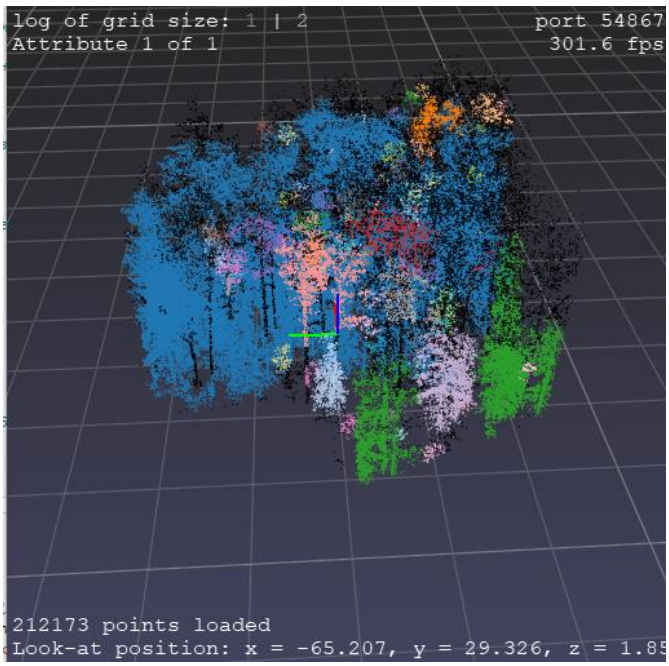

Рисунок 4.19 – результат сегментации при eps = 1, 158, min_samples = 50

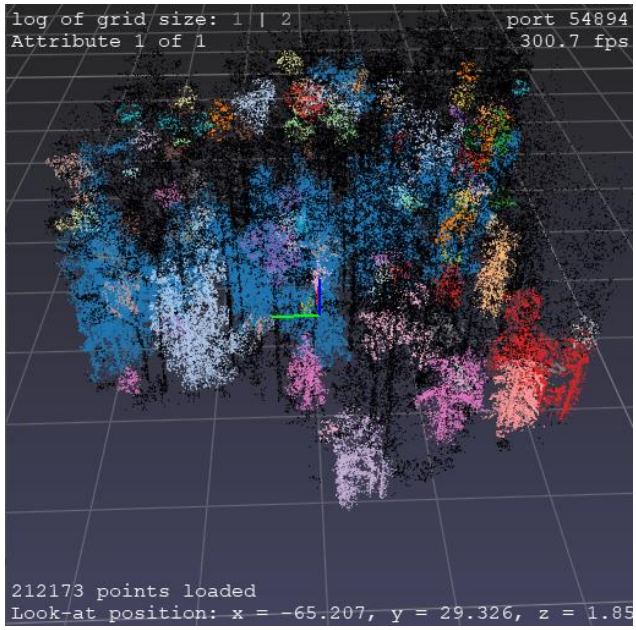

Рисунок 4.20 – результат сегментации при eps = 1,158, min_samples = 90

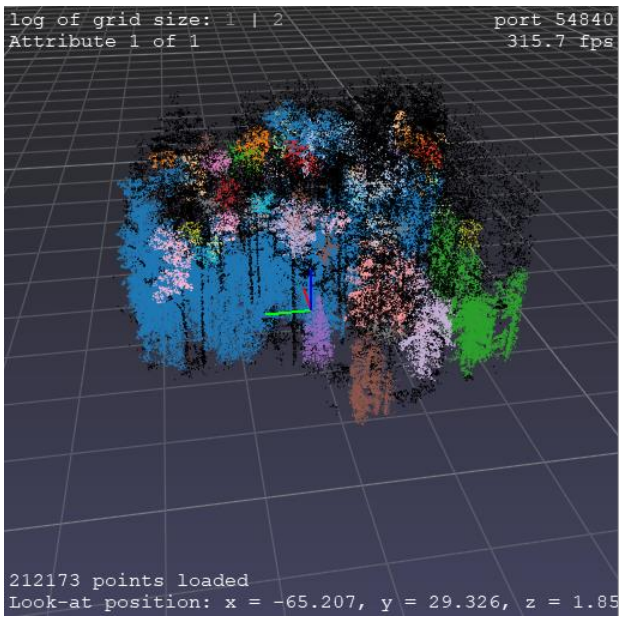

Рисунок 4.21 – результат сегментации при eps = 1,428, min_samples = 120

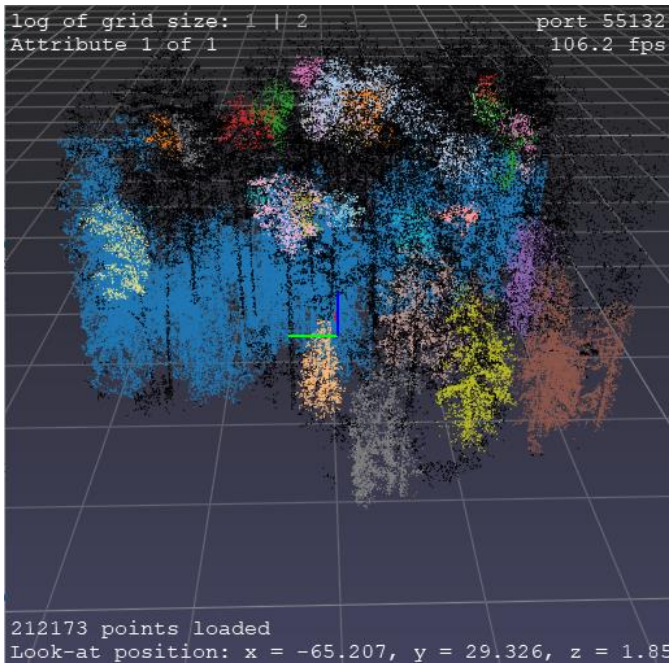

Рисунок 4.22 – результат сегментации при eps = 1,712, min_samples = 200

ЗАКЛЮЧЕНИЕ

В результате выполнения научно-исследовательской работы были выполнены следующие задачи:

1. С использованием метода машинного обучения DBSCAN была произведена сегментация деревьев из облака точек. В результате сегментации было получено три кластера, соответствующие отдельным деревьям.
2. Была произведена оценка качества кластеризации с использованием метрики Silhouette Score. Значение sil_score = 0,287403 может быть обусловлено тем, что деревья являются протяженными объектами, поэтому среднее расстояние от точки одного кластера до других точек этого же кластера может оказаться больше, чем для более типичных кластеров, что вносит вклад в значение метрики Silhouette Score.
3. С использованием метода машинного обучения DBSCAN была произведена сегментация деревьев из облака точек большой размерности. Облако точек было прорежено различными способами. Удалось достичь частичной сегментации облака точек, что может быть связано с тем, что DBSCAN имеет меньшую эффективность для сегментации облаков точек большой размерности, а также с тем, что точки распределены по облаку неравномерно в определенных его частях. Можно предположить, что для решения такой задачи больше бы подошел алгоритм, не связанный с плотностью распределения точек.
4. Был использован графический метод определения оптимального eps для сегментации облака точек. Причиной, по которой не удалось достичь полной сегментации облака точек, может быть нарушение совершаемого при использовании такого метода допущения об одинаковой размерности и средней плотности кластеров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Fred Westling, James Underwood, Mitch Bryson. Graph-based methods for analyzing orchard tree structure using noisy point cloud data: [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 02.02.2021. URL: https://arxiv.org/abs/2009.13727 (Дата обращения: 29.03.2023).

2. Jonathan Williams, Carola-Bibiane Schonlieb, Tom Swinfield, Juheon Lee, Xiaohao Cai, Lan Qie, David A. Coomes. Three-dimensional Segmentation of Trees Through a Flexible Multi-Class Graph Cut Algorithm (MCGC): [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 20.03.2019. URL: https://arxiv.org/abs/1903.08481 (Дата обращения: 29.03.2023).

3. Lloyd Windrim, Mitch Bryson. Forest Tree Detection and Segmentation using High Resolution Airborne LiDAR: [Электронный ресурс]. // arXiv.org. 2023. Дата обновления: 30.10.2018. URL: https://arxiv.org/abs/1810.12536 (Дата обращения: 29.03.2023).

4. Kaisen Ma, Zhenxiong Chen, Liyong Fu, Wanli Tian, Fugen Jiang, Jing Yi, Zhi Du, Hua Sun. Performance and Sensitivity of Individual Tree Segmentation Methods for UAV-LiDAR in Multiple Forest Types: [Электронный ресурс]. // mdpi.com. 2023. Дата обновления: 10.01.2022. URL: https://www.mdpi.com/2072-4292/14/2/298 (Дата обращения: 29.03.2023).

5. Feiyu Wang, Mitch Bryson. Tree Segmentation and Parameter Measurement from Point Clouds Using Deep and Handcrafted Features: [Электронный ресурс]. // researchgate.net. 2023. Дата обновления: 16.02.2023. URL: https://www.researchgate.net/publication/368612705_Tree_Segmentation_and_Parameter_Measurement_from_Point_Clouds_Using_Deep_and_Handcrafted_Features (Дата обращения: 29.03.2023).