```
1 from google.colab import drive
 2 drive.mount('/content/drive')
→ Mounted at /content/drive
 1 import csv
 2 import numpy as np
 3 import pandas as pd
 4 import math
 5 import time
 1 import matplotlib.pyplot as plt
 2 from mpl_toolkits.mplot3d import Axes3D
 3 from mpl toolkits.mplot3d.art3d import Poly3DCollection
 4
 1 import os
 2 import torch
 3 import numpy as np
 4 import pickle
 5 import logging
 6 import random
 7 from tqdm import tqdm
 8 #from torch_geometric.data import Dataset
Read data
```

```
1 import sys
2 sys.path.append('/content/drive/MyDrive/action/data/processed/mmr_kp')
1 filename = "/content/drive/MyDrive/action/data/processed/mmr_kp/data.pkl"
1 import os
2 import numpy as np
3 import pandas as pd
4 import pickle
5 import matplotlib.pyplot as plt
6 from mpl_toolkits.mplot3d import Axes3D
7
8 class MMRDataAnalyzer:
      def __init__(self, data_path):
9
          """Initialize analyzer with data path"""
10
```

```
11
           self.data_path = data_path
           self.data = self.load processed data()
12
13
           self.train_data = self.data['train']
14
           self.full df = None
15
       def load_processed_data(self):
16
17
           """Load processed MMR data from pickle file"""
           with open(self.data_path, 'rb') as f:
18
               return pickle.load(f)
19
20
       def get_frame_data(self, data_point):
21
           """Convert a single data point to DataFrame"""
22
           x = data point['new x']
23
           df = pd.DataFrame({
24
               'frame_num': np.zeros(len(x)),
25
               'x': x[:, 0],
26
               'y': x[:, 1],
27
               'z': x[:, 2]
28
29
           })
30
           if 'cluster_labels' in data_point:
31
               df['cluster'] = data_point['cluster_labels']
32
           return df
33
       def analyze_dataset(self):
34
           """Analyze entire dataset"""
35
           all_points = []
36
           all frames = []
37
           frame count = 0
38
39
           for data point in self.train data:
40
               x = data_point['new_x']
41
               num points = len(x)
42
43
               all_points.append(x)
               all_frames.extend([frame_count] * num_points)
44
               frame count += 1
45
46
47
           all points = np.concatenate(all points, axis=0)
48
           self.full_df = pd.DataFrame({
               'frame_num': all_frames,
49
               'x': all points[:, 0],
50
51
               'y': all_points[:, 1],
52
               'z': all_points[:, 2]
53
           })
54
           return self.full_df
55
       def visualize_frame(self, frame_idx=0, title="Point Cloud Visualization")
56
           """Visualize a single frame of point cloud data"""
57
           data point = self.train data[frame idx]
58
           x = data_point['new_x']
59
60
           fig = plt.figure(figsize=(12, 8))
61
```

```
62
            ax = fig.add_subplot(111, projection='3d')
            scatter = ax.scatter(x[:, 0], x[:, 1], x[:, 2],
 63
 64
                               c=x[:, 2],
                               cmap='viridis',
 65
                               marker='o',
 66
 67
                               s=50)
 68
 69
            ax.set_xlabel('X')
            ax.set_ylabel('Y')
 70
            ax.set_zlabel('Z')
 71
 72
            ax.set_title(title)
            plt.colorbar(scatter, label='Z coordinate')
 73
            plt.show()
 74
 75
 76
       def analyze_by_action(self):
 77
            """Analyze data grouped by action labels"""
            if 'y' not in self.train data[0]:
 78
 79
                return None
 80
 81
            action data = {}
 82
            action_labels = [d['y'] for d in self.train_data]
 83
            for data_point, action in zip(self.train_data, action_labels):
 84
 85
                if action not in action data:
                    action_data[action] = []
 86
                action_data[action].append(data_point['new_x'])
 87
 88
            stats = {}
 89
            for action, points in action_data.items():
 90
                points = np.concatenate(points, axis=0)
 91
                df = pd.DataFrame({
 92
                    'x': points[:, 0],
 93
                    'y': points[:, 1],
 94
                    'z': points[:, 2]
 95
 96
                })
                stats[action] = df.describe()
 97
 98
            return stats
 99
        def plot_coordinate_distributions(self):
100
            """Plot distributions of x, y, z coordinates"""
101
102
            if self.full_df is None:
103
                self.analyze_dataset()
104
            fig, axes = plt.subplots(1, 3, figsize=(15, 5))
105
106
            for i, coord in enumerate(['x', 'y', 'z']):
107
                axes[i].hist(self.full df[coord], bins=50)
108
                axes[i].set_title(f'{coord.upper()} Distribution')
109
                axes[i].set_xlabel(f'{coord.upper()} Coordinate')
110
                axes[i].set_ylabel('Frequency')
111
112
```

```
113
            plt.tight_layout()
            plt.show()
114
115
       def print detailed statistics(self):
116
            """Print detailed statistics for each coordinate"""
117
            if self.full df is None:
118
119
                self.analyze dataset()
120
            # Percentile analysis
121
           for coord in ['x', 'y', 'z']:
122
                print(f"\nDetailed {coord.upper()} coordinate analysis:")
123
                percentiles = [0, 1, 5, 25, 50, 75, 95, 99, 100]
124
125
                for p in percentiles:
                    value = np.percentile(self.full df[coord]. p)
126
                    print(f"{p}th percentile: {value:.6f}")
127
128
                print(f"Mean: {self.full df[coord].mean():.6f}")
129
                print(f"Std: {self.full_df[coord].std():.6f}")
130
                print(f"Skewness: {self.full_df[coord].skew():.6f}")
131
                print(f"Kurtosis: {self.full df[coord].kurtosis():.6f}")
132
133
134
           # Non-zero analysis
135
            print("\nAnalysis of non-zero points:")
           for coord in ['x', 'y', 'z']:
136
                non_zero = self.full_df[self.full_df[coord] != 0][coord]
137
                print(f"\n{coord.upper()} coordinate (non-zero):")
138
                print(f"Count: {len(non zero)}")
139
                print(f"Mean: {non zero.mean():.6f}")
140
                print(f"Std: {non_zero.std():.6f}")
141
                print(f"Min: {non zero.min():.6f}")
142
                print(f"Max: {non_zero.max():.6f}")
143
144
       def analyze_points_per_frame(self):
145
            """Analyze and visualize points per frame distribution"""
146
            points_per_frame = [len(d['new_x']) for d in self.train_data]
147
148
149
            print("\nPoints per frame:")
            print(f"Mean: {np.mean(points_per_frame):.2f}")
150
            print(f"Std: {np.std(points_per_frame):.2f}")
151
152
            print(f"Min: {np.min(points per frame)}")
            print(f"Max: {np.max(points_per_frame)}")
153
154
            plt.figure(figsize=(10, 5))
155
            plt.hist(points_per_frame, bins=50)
156
            plt.title('Distribution of Points per Frame')
157
158
            plt.xlabel('Number of Points')
            plt.ylabel('Frequency')
159
160
            plt.show()
161
162 def main():
       # Initialize analyzer
163
```

```
164
        data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
        analyzer = MMRDataAnalyzer(data_path)
165
166
167
       # Run analyses
       print("\nAnalyzing training data...")
168
169
170
       # Single frame analysis
171
       print("\nSingle Frame Analysis:")
172
        frame_df = analyzer.get_frame_data(analyzer.train_data[0])
        print("\nFrame Statistics:")
173
       print(frame_df.describe())
174
175
       # Visualize first frame
176
177
        print("\nVisualizing first frame...")
178
        analyzer.visualize_frame(0)
179
       # Full dataset analysis
180
       print("\nFull Dataset Analysis:")
181
182
       full_df = analyzer.analyze_dataset()
183
        print("\nDataset Statistics:")
184
       print(full_df.describe())
185
       # Plot distributions
186
187
       analyzer.plot_coordinate_distributions()
188
189
       # Action analysis
190
        action stats = analyzer.analyze by action()
        if action stats:
191
            print("\nAnalysis by Action:")
192
            for action, stats in action stats.items():
193
                print(f"\nAction {action} Statistics:")
194
                print(stats)
195
196
197
       # Points per frame analysis
        analyzer.analyze_points_per_frame()
198
199
200
       # Detailed statistics
201
        analyzer.print_detailed_statistics()
202
203 if __name__ == "__main__":
        main()
204
 \rightarrow
     Analyzing training data...
     Single Frame Analysis:
     Frame Statistics:
        frame num
                                X
                                             У
     count 110.0 110.000000 110.000000 110.000000
                 0.0 \quad 0.047783 \quad 0.536345 \quad -0.038962
     mean
```

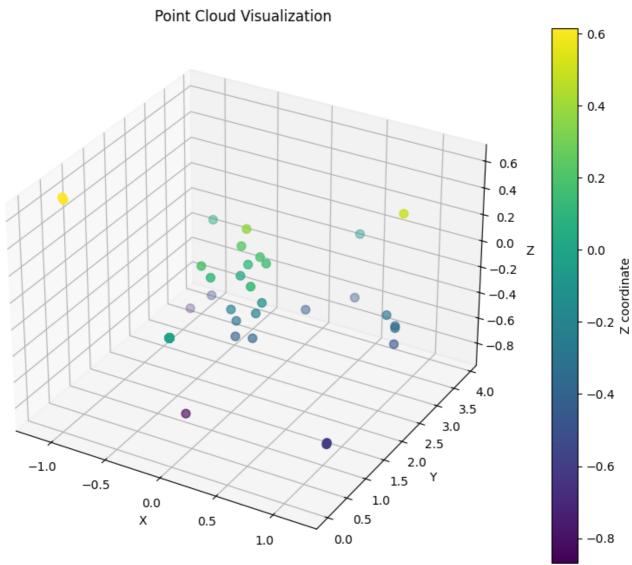
0.0 0.349468 0.975685 0.212701 0.0 _1 158141 0.000000 _0.868396

std

0.0

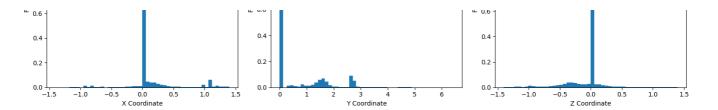
111711	U . U	- + • + > 0 + = +	$\cup \bullet \cup \cup \cup \cup \cup$	-0.0000000
25%	0.0	0.00000	0.00000	0.00000
50%	0.0	0.00000	0.00000	0.00000
75%	0.0	0.00000	0.577146	0.00000
max	0.0	1.202685	3.890844	0.614703

Visualizing first frame...



Full Dataset Analysis:

Datase	t Statistics:				
	frame_num	X	У	Z	
count	1.873696e+07	1.873696e+07	1.873696e+07	1.873696e+07	
mean	8.516750e+04	7.687180e-02	5.204076e-01	-5.673060e-02	
std	4.917177e+04	3.291378e-01	9.468305e-01	2.585428e-01	
min	0.000000e+00	-1.399950e+00	0.000000e+00	-1.399992e+00	
25%	4.258375e+04	0.000000e+00	0.000000e+00	0.000000e+00	
50%	8.516750e+04	0.000000e+00	0.000000e+00	0.000000e+00	
75%	1.277512e+05	0.000000e+00	8.577272e-01	0.000000e+00	
max	1.703350e+05	1.399950e+00	6.434681e+00	1.399989e+00	
1e7	X Distribution	1e7	Y Distribution	167	Z Distribution
1.4 -	1	1.4		1.4 -	
1.2 -		1.2 -		1.2 -	
		1.0 -		1.0 -	
1.0 -				1.0	
requency 8 8.0		- 8.0 August - 8.0		- 8.0 	
redn		inba.		l de l	



Analysis by Action:

	0.4				
Action	.3 .1	S+ 2+	7 (c+1~c	
ACCIOII	$_{\rm J}$	blat		コレエレコ	

	X	У	Z
count	378400.000000	378400.000000	378400.000000
mean	0.074577	0.514265	-0.061778
std	0.335568	0.939826	0.261655
min	-1.371951	0.00000	-1.398615
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.825487	0.00000
max	1.399950	6.349585	1.399224

Action 11 Statistics:

	X	У	Z
count	468820.000000	468820.000000	468820.000000
mean	0.077840	0.527854	-0.056401
std	0.317026	0.960928	0.250767
min	-1.389769	0.00000	-1.398883
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.854694	0.00000
max	1.393587	6.434681	1.398990

Action 12 Statistics:

	X	У	Z
count	471350.000000	471350.000000	471350.000000
mean	0.078124	0.485886	-0.059419
std	0.338254	0.917437	0.281582
min	-1.359224	0.00000	-1.399949
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.513116	0.00000
max	1.393587	6.399363	1.398434

Action 46 Statistics:

0	- 0 0000-00-00		
	X	У	Z
count	157080.000000	157080.000000	157080.000000
mean	0.076483	0.431295	-0.043606
std	0.309860	0.886490	0.219364
min	-1.309590	0.00000	-1.398176
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.00000	0.00000
max	1.393587	6.285223	1.391910

Action 10 Statistics:

	X	У	Z
count	468930.000000	468930.000000	468930.000000
mean	0.072497	0.508579	-0.056327
std	0.323098	0.925929	0.264549

min	-1.374497	0.000000	-1.399579
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.849805	0.000000
max	1.399950	6.319821	1.399556
Action count mean std	5 Statistics: x 467720.000000 0.076633 0.343420	y 467720.000000 0.516826 0.954382	2 467720.000000 -0.047413 0.253734
min	-1.374497	0.000000	-1.395063
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.777470	0.000000
max	1.399950	6.379820	1.397185
Action	40 Statistics:	У	Z
count mean std min 25% 50% 75% max	343640.000000	343640.000000	343640.000000
	0.078217	0.534478	-0.058406
	0.336558	0.955172	0.264493
	-1.389769	0.000000	-1.399087
	0.000000	0.000000	0.000000
	0.000000	0.000000	0.000000
	0.000000	0.894839	0.000000
	1.399950	6.383748	1.399817
Action	28 Statistics: x 403040.000000	y 403040.000000	z 403040.000000
mean	0.076906	0.549277	-0.068857
std	0.345861	0.969178	0.269449
min	-1.374497	0.000000	-1.399150
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.929228	0.000000
max	1.399950	6.346957	1.398393
	37 Statistics:	у	Z
count mean std min 25% 50% 75% max	374440.000000	374440.000000	374440.000000
	0.078752	0.522425	-0.066635
	0.315767	0.955456	0.264576
	-1.394860	0.000000	-1.399194
	0.000000	0.000000	0.000000
	0.000000	0.000000	0.000000
	0.000000	0.934513	0.000000
	1.399950	6.434008	1.399444
Action	48 Statistics: x 90420.000000	y 90420.000000	z 90420.000000
mean	0.078932	0.436856	-0.056474
std		0.868073	0.238996
min 25%	-1.276500 0.000000	0.868073	0.238996 -1.391721 0.000000

50%	0.00000	0.00000	0.00000
75%	0.00000	0.00000	0.00000
max	1.399950	6.269374	1.395553
Action	9 Statistics:		
	X	У	Z
count	462990.000000	462990.000000	462990.000000
mean	0.073573	0.554251	-0.058386
std	0.330726	0.967275	0.261660
min	-1.389769	0.00000	-1.399277
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.167069	0.00000
max	1.399950	6.417338	1.397870
Action	18 Statistics:		
	X	y 464200 000000	Z
count		464200.000000	464200.000000
mean	0.076618	0.537162	-0.042527
std min	0.327533 -1.399950	0.959041	0.258339 -1.398597
25%	0.00000	0.00000	0.000000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.082601	0.000000
max	1.399950	6.393705	1.398511
max	1.377730	0.373703	1.370311
Action	3 Statistics:	У	Z
count		467390.000000	467390.000000
mean	0.077899	0.568150	-0.071368
std	0.348494	0.986090	0.279974
min	-1.397405	0.00000	-1.398430
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.091420	0.00000
max	1.399950	6.383855	1.399461
Action	4 Statistics:		
	X	у	Z
count	470030.000000	470030.000000	470030.000000
mean	0.083634	0.573610	-0.073397
std	0.360205	1.006856	0.290792
min	-1.399950	0.000000	-1.398132
25% 50%	0.00000	0.000000	0.000000
75%	0.00000	0.00000	0.00000
max	1.399950	6.405042	1.399950
Illax	1.399930	0.403042	1.399930
Action	19 Statistics:	У	Z
count	466180.000000	466180.000000	466180.000000
mean	0.076934	0.500493	-0.049903
std	0.324334	0.937175	0.255963
min	-1.374497	0.00000	-1.399920
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.646444	0.00000
max	1.399950	6.424575	1.399923

7	20		
Action	32 Statistics:		
a	x 382580.000000	У 382580.000000	z 382580.000000
count mean	0.075206	0.479069	-0.048354
std	0.075200	0.900482	0.249378
min			
	-1.378315	0.000000	-1.397405
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.00000
75%	0.000000	0.534835	0.000000
max	1.397405	6.147089	1.399785
Action	2 Statistics:		
	X	У	Z
count	470470.000000	470470.000000	470470.000000
mean	0.075689	0.579819	-0.063618
std	0.346258	1.001837	0.275747
min	-1.399950	0.000000	-1.399992
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	1.155810	0.000000
max	1.399950	6.408193	1.399882
IIIax	1.399930	0.400193	1.399002
Action	35 Statistics:		
	X	У	Z
count	343970.000000	343970.000000	343970.000000
mean	0.065449	0.479796	-0.041497
std	0.302885	0.897658	0.235044
min	-1.351588	0.000000	-1.399636
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.508344	0.000000
max	1.393587	6.342800	1.399777
max	1.373307	0.542000	1.333111
Action	17 Statistics:		
	X	У	Z
count	437250.000000	437250.000000	437250.000000
mean	0.080128	0.562249	-0.044236
std	0.332713	0.988308	0.257440
min	-1.374497	0.00000	-1.398787
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.174127	0.00000
max	1.399950	6.434319	1.399356
Action	33 Statistics:		
	x	У	Z
count	408430.000000	408430.000000	408430.000000
mean	0.081345	0.529303	-0.059197
std	0.346095	0.949509	0.266641
min	-1.378315	0.00000	-1.399966
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.849508	0.00000
max	1.399950	6.434681	1.398520

Action 44 Statistics:

count 180730.000000 180730.000000 180730.000000 180730.000000 mean 0.068136 0.405919 -0.045828 std 0.330267 0.848062 0.217462 min -1.389769 0.000000 -1.398622 0.000000 -1.398622 50% 0.000000 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.000000 0.000000 max 1.393587 6.163586 1.385596 Action 30 Statistics: y z count 407000.000000 407000.000000 407000.00000 mean 0.082047 0.489071 -0.073012 50262658 min -1.374497 0.00000 -1.399952 538 0.00000 -1.399952 25% 0.000000 0.000000 -1.399952 528 0.000000 -1.399745 2 count 375210.000000 375210.000000 375210.000000 375210.000000 -0.061462 <th></th> <th>X</th> <th>У</th> <th>Z</th>		X	У	Z
mean 0.068136 0.405919 -0.045828 std 0.330267 0.848062 0.217462 min -1.389769 0.000000 -1.398622 25% 0.000000 0.000000 -1.398622 25% 0.000000	aoun+			_
std 0.330267 0.848062 0.217462 min -1.388769 0.000000 -1.398622 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.00000 0.000000 0.000000 Action 30 Statistics: V Z count 407000.00000 407000.00000 407000.00000 mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399952 25% 0.000000 0.000000 -1.399955 5% 0.000000 0.712131 0.00000 75% 0.000000 0.712131 0.00000 75% 0.000000 0.712131 0.00000 max 1.399950 6.279720 1.396795 Action 375210.000000 375210.00000 375210.00000 mean 0.076987 0.559731 -0.061462				
min -1.389769 0.000000 -1.398622 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.000000 0.000000 max 1.393587 6.163586 1.385596 Action 30 Statistics: 30 Statistics: <td< td=""><td></td><td></td><td></td><td></td></td<>				
25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.000000 0.000000 max 1.393587 6.163586 1.385596 Action 30 Statistics: Y Z count 407000.00000 407000.00000 407000.00000 mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.00000 -1.399925 25% 0.000000 0.00000 0.00000 50% 0.000000 0.712131 0.00000 75% 0.000000 0.712131 0.00000 max 1.399950 6.279720 1.396795 Action 26 Statistics: Z Z count 375210.000000 375210.00000 375210.00000 mean 0.076987 0.559731 -0.061462 std 0.000000 0.000000 0.00000 <t< td=""><td></td><td></td><td></td><td></td></t<>				
50% 0.000000 0.000000 0.000000 75% 0.000000 0.000000 0.000000 max 1.393587 6.163586 1.385596 Action 30 Statistics: X Y Count 407000.000000 407000.000000 407000.000000 me 20000000 407000.000000 407000.000000 0.0073012 std 0.0325658 0.896445 0.262465 0.000000 -1.399925 26465 0.000000 -1.399925 26465 0.000000 -1.399925 26465 0.000000 -1.399925 0.000000 0.00000				
75% 0.000000 0.000000 0.000000 max 1.393587 6.163586 1.385596 Action 30 Statistics: X y Z count 407000.000000 407000.000000 407000.000000 min 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 -1.399925 50% 0.000000 0.700000 -1.399925 50% 0.000000 0.000000 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: y z z count 375210.000000 375210.00000 375210.00000 375210.00000 -1.399356 2283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 -1.399356 28 0.00000 <	25%	0.00000		0.00000
max 1.393587 6.163586 1.385596 Action 30 Statistics: x y z count 407000.000000 407000.000000 407000.000000 mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 -0.00000 50% 0.000000 0.712131 0.00000 75% 0.000000 0.712131 0.00000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y x count 375210.00000 375210.00000 375210.00000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397495 0.000000 -1.399356 Action 13 Statistics: y y z count 469040.00000 469	50%	0.00000	0.00000	0.00000
Action 30 Statistics:	75%	0.00000	0.00000	0.00000
Action 30 Statistics:	max	1.393587	6.163586	1.385596
count 407000.000000 407000.000000 407000.000000 407000.000000 mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 0.000000 50% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.00000 0.00000 5% 0.000000 0.00000 0.00000 5% 0.000000 0.00000 0.00000 5% 0.000000 0.00000 0.00000 5% 0.000000 0.00000 0.00000 8 0.00000 469040.00000				
count 407000.000000 407000.000000 407000.000000 407000.000000 mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 0.000000 50% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.00000 0.00000 5% 0.000000 0.00000 0.00000 5% 0.000000 0.00000 0.00000 5% 0.000000 0.00000 0.00000 5% 0.000000 0.00000 0.00000 8 0.00000 469040.00000	Action	30 Statistics:		
count 407000.000000 407000.000000 407000.000000 mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 0.000000 50% 0.000000 0.000000 0.000000 50% 0.000000 0.712131 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: y x y c count 375210.00000 375210.00000 375210.0000 375210.0000 375210.0000 0.001462 3168 0.001462 0.000000 0.000000	11001011		V	7
mean 0.082047 0.489071 -0.073012 std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.00000 -1.399356 25% 0.000000 0.000000 0.00000 50% 0.000000 1.024297 0.00000 75% 0.000000 469040.00000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821	count		_	_
std 0.325658 0.896445 0.262465 min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.337405 0.00000 -1.399356 25% 0.000000 0.00000 -1.399356 30 0.00000 0.00000 0.00000 75% 0.000000 0.00000 0.00000 75% 0.000000 469040.00000 469040.00000 25% 0.00000 0.00000 0.00000 25% 0.00000 0.00000 0.00000 <t< td=""><td></td><td></td><td></td><td></td></t<>				
min -1.374497 0.000000 -1.399925 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: v v v count 375210.000000 375210.000000 375210.00000 0.00000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 0.001462 std 0.347518 0.974459 0.283168 0.0016462 std 0.347518 0.974459 0.283168 0.1399356 25% 0.000000 0.000000 -1.339356 25% 0.000000				
25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics:				
50% 0.000000 0.000000 0.000000 75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 ary 2 count 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 469040.000000 469040.000000 469040.000000 469040.000000 469040.000000 469040.000000 469040.000000 -0.053410 std 0.264821 -0.053410 std 0.264821 -0.053410 std 0.264821 -0.053410 std 0.264821 -0.053410 std				
75% 0.000000 0.712131 0.000000 max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 -1.399356 50% 0.000000 0.000000 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: y z z count 469040.000000 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 0.053410 20 std 0.340723 0.928638 0.264821 0.239950 st 0.000000 0.000000 0.000000 0.000000 </td <td></td> <td></td> <td></td> <td></td>				
max 1.399950 6.279720 1.396795 Action 26 Statistics: x y z count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.00000 469040.00000 469040.00000 mean 0.071510 0.550502 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.00000 -1.399950 25% 0.000000 0.00000 0.00000 75% 0.00000 0.801803 <td< td=""><td>50%</td><td>0.00000</td><td>0.00000</td><td>0.00000</td></td<>	50%	0.00000	0.00000	0.00000
Action 26 Statistics: x	75%	0.00000	0.712131	0.00000
count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.00000 469040.00000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.00000 -1.399950 25% 0.000000 0.00000 -1.399950 25% 0.000000 0.00000 0.00000 50% 0.000000 0.000000 0.000000 75% 0.000000 349250.000000 349250.00000 75% 0.000000 0.000000 0.0	max	1.399950	6.279720	1.396795
count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.00000 469040.00000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.00000 -1.399950 25% 0.000000 0.00000 -1.399950 25% 0.000000 0.00000 0.00000 50% 0.000000 0.000000 0.000000 75% 0.000000 349250.000000 349250.00000 75% 0.000000 0.000000 0.0				
count 375210.000000 375210.000000 375210.000000 mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 1.024297 0.000000 75% 0.000000 46904297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.00000 469040.00000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 50% 0.000000 0.000000 0.000000 0.000000 50% 0.000000 0.801803 0.000000 75% 0.000000 349250.000000 349250.000000 mean 0.079881 <t< td=""><td>Action</td><td>26 Statistics:</td><td></td><td></td></t<>	Action	26 Statistics:		
mean 0.076987 0.559731 -0.061462 std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 1.024297 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399550 6.296864 1.399496 Action 349250.000000 349250.000000 349250.000000 max 0.00000		X	У	Z
std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.00000 469040.00000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399590 6.296864 1.399496 Action 349250.00000 349250.00000 349250.00000 0.063319 0.063319 0.063319	count	375210.000000	375210.000000	375210.000000
std 0.347518 0.974459 0.283168 min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.00000 469040.00000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399590 6.296864 1.399496 Action 349250.00000 349250.00000 349250.00000 0.063319 0.063319 0.063319				
min -1.397405 0.000000 -1.399356 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.000000 469040.000000 469040.00000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0				
25% 0.000000 0.000000 0.000000 50% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics:				
50% 0.000000 0.000000 0.000000 75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.801803 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769<				
75% 0.000000 1.024297 0.000000 max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 0.000000 50% 0.000000 0.801803 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.00000 -1.397785 25% 0.00000				
max 1.399950 6.432427 1.399950 Action 13 Statistics: x y z count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.00000 -1.397785 25% 0.00000 0.00000 0.00000 0.00000 50% 0.00000 0.948160 0.00000 75% 0.000000 0.948160				
Action 13 Statistics:	75%			
x y z count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.801803 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.00000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 0.000000 75% 0.000000 0.000000 <	max	1.399950	6.432427	1.399950
x y z count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.801803 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.00000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 0.000000 75% 0.000000 0.000000 <	7	12 31 1 1 1		
count 469040.000000 469040.000000 469040.000000 mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 349250.000000 349250.000000 349250.00000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z	Action			
mean 0.071510 0.505082 -0.053410 std 0.340723 0.928638 0.264821 min -1.374497 0.000000 -1.399950 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.00000 349250.00000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.948160 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y				
std0.3407230.9286380.264821min-1.3744970.000000-1.39995025%0.0000000.0000000.00000050%0.0000000.0000000.00000075%0.0000000.8018030.000000max1.3999506.2968641.399496Action36 Statistics:xyzcount349250.00000349250.00000349250.00000mean0.0798810.528928-0.063319std0.3204220.9564550.263092min-1.3897690.000000-1.39778525%0.0000000.0000000.00000050%0.0000000.0000000.00000075%0.0000000.9481600.000000max1.3935876.3111011.399598Action16 Statistics:xyzcount466400.000000466400.000000466400.000000				
min-1.3744970.000000-1.39995025%0.0000000.0000000.00000050%0.0000000.0000000.00000075%0.0000000.8018030.000000max1.3999506.2968641.399496Action 36 Statistics:xyzcount349250.000000349250.000000349250.000000mean0.0798810.528928-0.063319std0.3204220.9564550.263092min-1.3897690.000000-1.39778525%0.0000000.0000000.00000050%0.0000000.0000000.00000075%0.0000000.9481600.000000max1.3935876.3111011.399598Action 16 Statistics:xyzcount466400.000000466400.000000466400.000000				
25% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics:	std	0.340723	0.928638	
50% 0.000000 0.000000 0.000000 75% 0.000000 0.801803 0.000000 max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	min	-1.374497	0.00000	-1.399950
75% 0.000000 0.801803 0.0000000 max 1.399950 6.296864 1.399496 Action 36 Statistics:	25%	0.00000	0.00000	0.00000
max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.948160 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	50%	0.00000	0.00000	0.00000
max 1.399950 6.296864 1.399496 Action 36 Statistics: x y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.948160 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	75%	0.00000	0.801803	0.00000
Action 36 Statistics: x y y z count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000				
count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	1110111	1,00000	01230001	1,000,100
count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	Action	36 Statistics:		
count 349250.000000 349250.000000 349250.000000 mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000			V	7.
mean 0.079881 0.528928 -0.063319 std 0.320422 0.956455 0.263092 min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	count		_	_
std 0.320422 0.956455 0.263092 min -1.389769 0.0000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.0000000				
min -1.389769 0.000000 -1.397785 25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000				
25% 0.000000 0.000000 0.000000 50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics:				
50% 0.000000 0.000000 0.000000 75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000				
75% 0.000000 0.948160 0.000000 max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z z count 466400.000000 466400.000000 466400.000000				
max 1.393587 6.311101 1.399598 Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000				
Action 16 Statistics: x y z count 466400.000000 466400.000000 466400.000000	75%			
x y z z count 466400.000000 466400.000000 466400.000000	max	1.393587	6.311101	1.399598
x y z z count 466400.000000 466400.000000 466400.000000				
count 466400.000000 466400.000000 466400.000000	Action	16 Statistics:		
			_	
mean 0.073473 0.502445 -0.040275	count			466400.000000
	mean	0.073473	0.502445	-0.040275

std	0.325360	0.954712	0.252284
min	-1.389769	0.00000	-1.399508
25%	0.000000	0.000000	0.000000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.523012	0.00000
max	1.399950	6.393885	1.398629
Action	20 Statistics:		
	X	У	Z
count	409420.000000	409420.000000	409420.000000
mean	0.079091	0.513077	-0.053115
std	0.317295	0.940046	0.256633
min	-1.389769	0.00000	-1.399572
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.000000	0.846486	0.000000
max	1.399950	6.398951	1.398761
Action	1 Statistics:		
0-011	X	У	Z
count	467390.000000	467390.000000	467390.000000
mean	0.073163	0.567240	-0.066120
std	0.349041	1.006595	0.294535
min	-1.399950	0.00000	-1.399929
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.898223	0.00000
max	1.399950	6.429296	1.399778
		0011110	
Action	25 Statistics:		
Action	25 Statistics:	V	Z
	X	y 402490.000000	_
count	x 402490.000000	402490.000000	402490.000000
count mean	x 402490.000000 0.072725	402490.000000 0.451899	402490.000000 -0.048821
count mean std	x 402490.000000 0.072725 0.316602	402490.000000 0.451899 0.883519	402490.000000 -0.048821 0.227775
count mean std min	x 402490.000000 0.072725 0.316602 -1.374497	402490.000000 0.451899 0.883519 0.000000	402490.000000 -0.048821 0.227775 -1.399848
count mean std min 25%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000
count mean std min	x 402490.000000 0.072725 0.316602 -1.374497	402490.000000 0.451899 0.883519 0.000000	402490.000000 -0.048821 0.227775 -1.399848
count mean std min 25%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000
count mean std min 25% 50%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.0000000
count mean std min 25% 50% 75%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.0000000 0.0000000
count mean std min 25% 50% 75% max	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.0000000 0.0000000
count mean std min 25% 50% 75% max Action	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190
count mean std min 25% 50% 75% max	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics:	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190
count mean std min 25% 50% 75% max Action	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190
count mean std min 25% 50% 75% max Action count	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190
count mean std min 25% 50% 75% max Action count mean	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 0.000000 1.397190
count mean std min 25% 50% 75% max Action count mean std min	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089
count mean std min 25% 50% 75% max Action count mean std min 25%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.0000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000
count mean std min 25% 50% 75% max Action count mean std min 25% 50%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 0.000000
count mean std min 25% 50% 75% max Action count mean std min 25% 50%	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75% max	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 0.000000
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75% max	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000 1.393587	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 0.000000
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75% max	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000 1.393587	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499 6.269374	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 0.000000 1.398368
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75% max Action	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000 1.393587 24 Statistics: x	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499 6.269374	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 0.000000 1.398368
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75% max Action count count mean std min count count mean	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000 1.393587 24 Statistics: x 404470.000000 0.079350	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499 6.269374	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 1.398368
count mean std min 25% 50% 75% max Action count mean std min 25% 50% 75% max Action	x 402490.000000 0.072725 0.316602 -1.374497 0.000000 0.000000 0.000000 1.399950 47 Statistics: x 149820.000000 0.071963 0.302505 -1.389769 0.000000 0.000000 0.000000 1.393587 24 Statistics: x 404470.000000	402490.000000 0.451899 0.883519 0.000000 0.000000 0.000000 0.000000 6.392045 Y 149820.000000 0.479796 0.915010 0.000000 0.000000 0.000000 0.509499 6.269374	402490.000000 -0.048821 0.227775 -1.399848 0.000000 0.000000 0.000000 1.397190 2 149820.000000 -0.067595 0.254481 -1.399089 0.000000 0.000000 0.000000 1.398368

25%	0.00000	0.00000	0.00000
50%	0.000000	0.00000	0.000000
75%	0.000000	1.025297	0.000000
max	1.399950	6.381793	1.399262
1110221	1.00000	0.001750	1.0000
Action	43 Statistics:	37	Z
count	179740.000000	У 179740.000000	179740.000000
mean	0.061394	0.457617	-0.069759
std	0.287983	0.883638	0.222477
min	-1.256137	0.00000	-1.398671
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.00000	0.00000
max	1.393587	6.240797	1.394823
Action	8 Statistics:		
	X	У	Z
count	469040.000000	469040.000000	469040.000000
mean	0.084970	0.522866	-0.055848
std	0.319263	0.944904	0.258823
min	-1.374497	0.000000	-1.399687
25% 50%	0.000000	0.000000	0.000000
75%	0.00000	0.000000 0.882933	0.00000
max	1.399950	6.432488	1.399646
Illax	1.399930	0.432400	1.399040
Action	23 Statistics:	У	Z
count	374550.000000	374550.000000	374550.000000
mean	0.074803	0.548074	-0.053603
std	0.332200	0.971611	0.241057
min	-1.374497	0.00000	-1.399892
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.062627	0.00000
max	1.399950	6.422101	1.397105
Action	0 Statistics:		
	X	У	Z
count	451770.000000	451770.000000	451770.000000
mean	0.070893	0.547336	-0.055559
std	0.321600	0.992501	0.260741
min	-1.399950	0.000000	-1.398672
25%	0.000000	0.000000	0.000000
50% 75%	0.00000	0.000000 0.906558	0.00000
max	1.399950	6.431149	1.399431
Illax	1.399930	0.431149	1.399431
Action	45 Statistics:	**	
count	x 158840.000000	У 158840.000000	z 158840.000000
mean	0.077193	0.410146	-0.036707
std	0.313086	0.858483	0.210863
min	-1.389769	0.000000	-1.397101
25%	0.000000	0.000000	0.000000
50%	0.000000	0.00000	0.000000
75%	0.000000	0.000000	0.000000

max	1.393587	6.186941	1.398242
IIIdA	1.373307	0.100741	1.370242
Action	21 Statistics:		
	X	У	Z
count	376750.000000	376750.000000	376750.000000
mean	0.078417	0.541347	-0.065800
std	0.334038	0.950795	
min	-1.389769	0.00000	-1.398807
25%	0.000000	0.00000	0.000000
50% 75%	0.000000	0.000000	0.000000
75% max	0.000000 1.393587	0.991691 6.413658	0.000000 1.390261
Illax	1.393307	0.413030	1.390201
Action	6 Statistics:		
	X	У	Z
count	466950.000000		466950.000000
mean	0.083345	0.542249	-0.055615
std	0.327001	0.977326	0.260309
min	-1.389769	0.00000	-1.399695
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.858658	0.00000
max	1.399950	6.429239	1.399429
Action	34 Statistics:		
	X	У	Z
count	372900.000000	372900.000000	372900.000000
mean	0.078543	0.504012	-0.050469
std	0.319428	0.918820	0.245226
min	-1.399950	0.00000	-1.398905
25%	0.00000	0.00000	0.00000
50%	0.00000	0.000000	0.000000
75%	0.000000	0.855156	
max	1.399950	6.414887	1.394047
Action	39 Statistics:		
	X	У	Z
count	376640.000000	376640.000000	376640.000000
mean	0.072222	0.548899	-0.071456
std	0.326717	0.960930	0.276787
min	-1.384678	0.00000	-1.399840
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.043640	0.00000
max	1.397405	6.356513	1.397997
Action	15 Statistics:		
	X	У	Z
count	469260.000000		469260.000000
mean	0.076804	0.539083	-0.043463
std	0.311330	0.953184	0.237928
min	-1.399950	0.00000	-1.399516
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	1.158413	0.00000

1.393587 6.414969

max

1.399753

Action	27 Statistics:		
	X	У	Z
count	405130.000000	405130.000000	405130.000000
mean	0.080284	0.547597	-0.053680
std	0.322925	0.947382	0.242774
min	-1.356679	0.00000	-1.399793
25%	0.00000	0.00000	0.00000
50%	0.000000	0.000000	0.000000
75%	0.000000	1.279008	0.000000
max	1.399950	6.390528	1.398357
max	1.377730	0.370320	1.370337
Action	14 Statistics:		
11001011	X	У	Z
count	440550.000000	440550.000000	440550.000000
mean	0.084835	0.504918	-0.049325
std	0.328976	0.937369	0.261018
min	-1.389769	0.00000	-1.399944
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.808561	0.00000
max	1.393587	6.176641	1.399989
Action	22 Statistics:		
	X	У	Z
count	375870.000000	375870.000000	375870.000000
mean	0.073941	0.568886	-0.058043
std	0.322990	0.966665	0.253392
min	-1.389769	0.00000	-1.399231
25%	0.00000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	1.260711	0.000000
max	1.393587	6.431538	1.399225
IIICX	1.373307	0.431330	1.377223
Action	29 Statistics:		
11001011	X	У	Z
count		380490.000000	_
mean	0.082675	0.440326	-0.052754
std	0.323891	0.440320	0.209361
min	-1.221775		
		0.000000	-1.398298
25%	0.000000	0.000000	0.000000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.308037	0.00000
max	1.399950	5.861362	1.399806
	40 71 11		
Action	42 Statistics:		
	X	У	Z
count	174240.000000	174240.000000	174240.000000
mean	0.071184	0.474571	-0.048048
std	0.310809	0.923397	0.244577
min	-1.374497	0.00000	-1.399148
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.000000	0.000000
max	1.399950	6.420762	1.399556
		00120702	
Action	38 Statistics:		
0011	X	У	Z
	22	y	

count 375210.000000 375210.000000 375210.000000

004110	3/321000000	3,321000000	3,3210.00000
mean	0.072853	0.501803	-0.058829
std	0.311792	0.929197	0.238388
min	-1.389769	0.00000	-1.399300
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.659137	0.00000
max	1.393587	6.399340	1.399584
Action	41 Statistics:		
	X	У	Z
count	340560.000000	340560.000000	340560.000000
mean	0.082395	0.511885	-0.051959
std	0.338019	0.919939	0.256697
min	-1.389769	0.00000	-1.399533
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.860548	0.00000
max	1.399950	6.267469	1.395117
Action	7 Statistics:		
	X	У	Z
count	469920.000000	469920.000000	469920.000000
mean	0.076454	0.489076	-0.063943
std	0.321035	0.931456	0.259399
min	-1.389769	0.00000	-1.399851
25%	0.00000	0.00000	0.00000
50%	0.00000	0.00000	0.00000
75%	0.00000	0.422368	0.00000
max	1.393587	6.386649	1.398395

Points per frame:

Mean: 110.00 Std: 0.00 Min: 110

Max: 110

Distribution of Points per Frame 160000 140000 120000 60000 40000 20000 109.6 109.8 110.0 110.2 110.4

Detailed X coordinate analysis:

Oth percentile: -1.399950 1th percentile: -0.916331 5th percentile: -0.058543 25th percentile: 0.000000 50th percentile: 0.000000 75th percentile: 0.000000 95th percentile: 1.065235 99th percentile: 1.207775 100th percentile: 1.399950 Mean: 0.076872 Std: 0.329138 Skewness: 1.463836 Kurtosis: 5.981577 Detailed Y coordinate analysis: Oth percentile: 0.000000 1th percentile: 0.000000 5th percentile: 0.000000 25th percentile: 0.000000 50th percentile: 0.000000 75th percentile: 0.857727 95th percentile: 2.688806 99th percentile: 3.350385 100th percentile: 6.434681 Mean: 0.520408 Std: 0.946831 Skewness: 1.786532 Kurtosis: 2.654240 Detailed Z coordinate analysis: Oth percentile: -1.399992 1th percentile: -1.075179 5th percentile: -0.538880 25th percentile: 0.000000 50th percentile: 0.000000 75th percentile: 0.000000 95th percentile: 0.148700 99th percentile: 0.693422 100th percentile: 1.399989 Mean: -0.056731Std: 0.258543 Skewness: -1.148323 Kurtosis: 8.433358 Analysis of non-zero points: X coordinate (non-zero): Count: 4988251 Mean: 0.288747 Std: 0.587996 Min: -1.399950Max: 1.399950 Y coordinate (non-zero): Count: 5281486 Mean: 1.846233 Std: 0.855954 Min: 0.300078

Max: 6.434681

Z coordinate (non-zero):
Count: 5276678
Mean: -0.201445
Std: 0.456296
Min: -1.399992
Max: 1.399989

Small tool functions

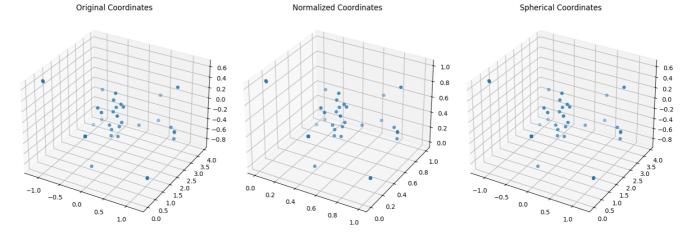
```
1 import numpy as np
 2 from sklearn.preprocessing import MinMaxScaler
 3
 4 class MMRUtils:
      @staticmethod
      def normalize(x, x_min=None, x_max=None):
 6
           """Normalize data to [0,1] range"""
 7
           if x_min is None:
 8
 9
               x_min = np_min(x)
10
           if x_max is None:
               x_max = np_max(x)
11
           return (x - x_min) / (x_max - x_min)
12
13
14
      @staticmethod
      def normalize_points(points):
15
16
           """Normalize point cloud data"""
```

```
17
           normalized = np.zeros_like(points)
18
           for i in range(points.shape[1]):
19
               normalized[:, i] = MMRUtils.normalize(points[:, i])
20
           return normalized
21
22
      @staticmethod
23
      def weight_by_intensity(data_point):
           """Convert intensity values to weights in [0,1] range"""
24
25
           if 'intensity' in data_point:
               intensities = data_point['intensity']
26
27
               return MinMaxScaler().fit_transform(
                   np.asarray(intensities).reshape(-1,1).reshape(1,-1)[0]
28
29
           return None
30
      @staticmethod
31
      def to_vertical(array):
32
           """Convert array to vertical format"""
33
34
           return array.reshape(-1, 1)
35
36
      @staticmethod
37
      def to_horizontal(array):
           """Convert array to horizontal format"""
38
39
           return array.reshape(1, -1)
40
41
      @staticmethod
      def vector_angle(vec1, vec2):
42
           """Calculate normalized angle between two vectors"""
43
           return np.abs(np.dot(vec1, vec2)) / (np.linalg.norm(vec1) * np.linal
44
45
46
      @staticmethod
      def cartesian_to_spherical(points):
47
           """Convert cartesian coordinates to spherical
48
49
50
               points: Array of shape (N, 3) containing [x, y, z] coordinates
51
           Returns:
52
               Array of shape (N, 3) containing [r, theta, phi] coordinates
53
54
           x, y, z = points[:, 0], points[:, 1], points[:, 2]
           r = np.sqrt(x**2 + y**2 + z**2)
55
           theta = np.arctan2(y, x)
56
57
           theta[theta < 0] += 2 * np.pi
58
           phi = np.arctan2(np.sqrt(x**2 + y**2), z)
           return np.stack([r, theta, phi], axis=1)
59
60
61
      @staticmethod
      def spherical_to_cartesian(points):
62
63
           """Convert spherical coordinates to cartesian
64
           Args:
               points: Array of shape (N, 3) containing [r, theta, phi] coordir
65
66
           Returns:
67
               Array of shape (N, 3) containing [x, y, z] coordinates
```

```
.....
 68
            r, theta, phi = points[:, 0], points[:, 1], points[:, 2]
 69
 70
            x = r * np.cos(theta) * np.sin(phi)
 71
            y = r * np.sin(theta) * np.sin(phi)
 72
            z = r * np.cos(phi)
 73
            return np.stack([x, y, z], axis=1)
 74
 75
       @staticmethod
 76
       def get_cluster_colors(labels):
            """Get colors for different clusters/labels"""
 77
           colors = ['r', 'b', 'g', 'c', 'm', 'darkorange', 'deepskyblue',
 78
                     'blueviolet', 'crimson', 'orangered', 'k']
 79
            return [colors[label % len(colors)] for label in labels]
 80
 81
 82
       @staticmethod
       def process_frame(data_point):
 83
            """Process a single frame of point cloud data
 84
            Returns normalized coordinates and spherical coordinates"""
 85
 86
            points = data_point['new_x']
 87
 88
            # Normalize cartesian coordinates
 89
            normalized = MMRUtils.normalize_points(points)
 90
 91
            # Convert to spherical coordinates
            spherical = MMRUtils.cartesian_to_spherical(points)
 92
 93
            return {
 94
                'original': points,
 95
                'normalized': normalized,
 96
 97
                'spherical': spherical
            }
 98
99
100 # Enhanced MMRDataAnalyzer class with utility functions
101 class EnhancedMMRAnalyzer(MMRDataAnalyzer):
       def __init__(self, data_path):
102
103
            super().__init__(data_path)
104
            self.utils = MMRUtils()
105
       def analyze_frame_geometry(self, frame_idx=0):
106
            """Analyze geometric properties of a frame"""
107
108
            data_point = self.train_data[frame_idx]
109
            processed = self.utils.process_frame(data_point)
110
            # Calculate geometric properties
111
            points = processed['original']
112
            center = np.mean(points, axis=0)
113
            distances = np.linalg.norm(points - center, axis=1)
114
115
            print("\nGeometric Analysis:")
116
            print(f"Center of mass: {center}")
117
            print(f"Mean distance from center: {np.mean(distances):.4f}")
118
```

```
119
            print(f"Max distance from center: {np.max(distances):.4f}")
120
121
            # Visualize in both coordinate systems
122
            fig = plt.figure(figsize=(15, 5))
123
            # Original coordinates
124
125
            ax1 = fig.add subplot(131, projection='3d')
            ax1.scatter(points[:, 0], points[:, 1], points[:, 2])
126
            ax1.set_title('Original Coordinates')
127
128
            # Normalized coordinates
129
            ax2 = fig.add_subplot(132, projection='3d')
130
            norm points = processed['normalized']
131
132
            ax2.scatter(norm_points[:, 0], norm_points[:, 1], norm_points[:, 2])
            ax2.set_title('Normalized Coordinates')
133
134
135
            # Spherical coordinates
            ax3 = fig.add_subplot(133, projection='3d')
136
137
            sph_points = processed['spherical']
            ax3.scatter(sph_points[:, 0] * np.sin(sph_points[:, 2]) * np.cos(sph_points[:, 2])
138
139
                       sph_points[:, 0] * np.sin(sph_points[:, 2]) * np.sin(sph_
140
                       sph_points[:, 0] * np.cos(sph_points[:, 2]))
            ax3.set_title('Spherical Coordinates')
141
142
            plt.tight_layout()
143
            plt.show()
144
145
146
            return processed
147
148 # Example usage
149 if __name__ == "__main__":
        data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
150
        analyzer = EnhancedMMRAnalyzer(data_path)
151
152
153
       # Analyze a frame with new utilities
154
        frame_data = analyzer.analyze_frame_geometry(0)
155
156
       # Example of using utilities
157
        points = analyzer.train_data[0]['new_x']
158
        normalized = analyzer.utils.normalize points(points)
        spherical = analyzer.utils.cartesian_to_spherical(points)
159
160
        print("\nPoint Cloud Statistics:")
161
        print(f"Original range: [{points.min():.4f}, {points.max():.4f}]")
162
        print(f"Normalized range: [{normalized.min():.4f}, {normalized.max():.4f}
163
164
        print(f"Spherical coordinates shape: {spherical.shape}")
```

```
Geometric Analysis:
Center of mass: [ 0.04778343  0.53634458 -0.03896176]
Mean distance from center: 0.8495
Max distance from center: 3.3704
```



```
Point Cloud Statistics:
Original range: [-1.1581, 3.8908]
Normalized range: [0.0000, 1.0000]
Spherical coordinates shape: (110, 3)
```

Plot data points

```
1 import os
 2 import numpy as np
 3 import pandas as pd
 4 import pickle
 5 import matplotlib.pyplot as plt
 6 from mpl_toolkits.mplot3d import Axes3D
 7
 8 class MMRUtils:
      @staticmethod
 9
      def normalize_points(points):
10
           """Normalize point cloud data to [0,1] range"""
11
           min_vals = np.min(points, axis=0)
12
           max_vals = np.max(points, axis=0)
13
           normalized = (points - min_vals) / (max_vals - min_vals)
14
           return normalized
15
16
```

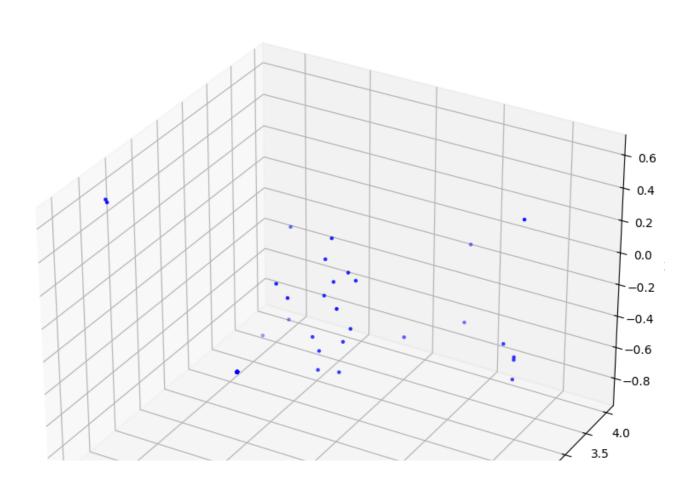
```
17
      @staticmethod
      def cartesian_to_spherical(points):
18
           """Convert cartesian to spherical coordinates"""
19
20
           x, y, z = points[:, 0], points[:, 1], points[:, 2]
           r = np.sqrt(x**2 + y**2 + z**2)
21
           theta = np.arctan2(y, x)
22
23
           phi = np.arccos(z/r)
24
           return np.stack([r, theta, phi], axis=1)
25
26
      @staticmethod
      def spherical_to_cartesian(points):
27
           """Convert spherical to cartesian coordinates"""
28
           r, theta, phi = points[:, 0], points[:, 1], points[:, 2]
29
           x = r * np.sin(phi) * np.cos(theta)
30
           y = r * np.sin(phi) * np.sin(theta)
31
           z = r * np.cos(phi)
32
           return np.stack([x, y, z], axis=1)
33
34
35 class EnhancedMMRAnalyzer:
36
      def init (self, data path):
37
           self.utils = MMRUtils()
38
           self.data_path = data_path
           self.load data()
39
40
      def load_data(self):
41
           """Load and verify data"""
42
43
           try:
               with open(self.data_path, 'rb') as f:
44
                   self.data = pickle.load(f)
45
               self.train data = self.data['train']
46
               print(f"Loaded {len(self.train_data)} training samples")
47
48
           except Exception as e:
               print(f"Error loading data: {e}")
49
50
               raise
51
52
      def visualize_frame(self, frame_idx=0):
53
           """Basic frame visualization"""
54
           try:
               data_point = self.train_data[frame_idx]
55
               points = data point['new x']
56
57
58
               # Create simple 3D scatter plot
               fig = plt.figure(figsize=(10, 10))
59
               ax = fig.add_subplot(111, projection='3d')
60
61
               # Plot points
62
63
               scatter = ax.scatter(points[:, 0], points[:, 1], points[:, 2],
                                  c='blue', marker='.')
64
65
               ax.set_xlabel('X')
66
67
               ax.set_ylabel('Y')
```

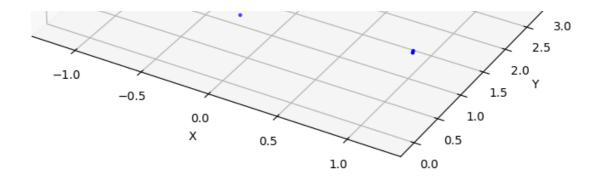
```
68
                ax.set_zlabel('Z')
 69
                ax.set_title(f'Frame {frame_idx} Point Cloud')
 70
 71
                # Print statistics
                print("\nFrame Statistics:")
 72
                print(f"Number of points: {len(points)}")
 73
 74
                print(f"X range: [{points[:, 0].min():.4f}, {points[:, 0].max():
 75
                print(f"Y range: [{points[:, 1].min():.4f}, {points[:, 1].max():
                print(f"Z range: [{points[:, 2].min():.4f}, {points[:, 2].max():
 76
 77
 78
                plt.show()
 79
                return points
 80
 81
            except Exception as e:
                print(f"Error visualizing frame: {e}")
 82
 83
                raise
 84
 85
        def visualize_frame_enhanced(self, frame_idx=0):
 86
            """Enhanced visualization with multiple views"""
 87
            try:
                data_point = self.train_data[frame_idx]
 88
                points = data_point['new_x']
 89
 90
 91
                # Calculate geometric properties
 92
                center = np.mean(points, axis=0)
 93
                distances = np.linalg.norm(points - center, axis=1)
 94
 95
                # Print analysis
                print("\nGeometric Analysis:")
 96
                print(f"Center of mass: {center}")
 97
                print(f"Mean distance from center: {np.mean(distances):.4f}")
 98
                print(f"Max distance from center: {np.max(distances):.4f}")
 99
100
101
                # Create figure with three views
102
                fig = plt.figure(figsize=(15, 5))
103
104
                # Original coordinates
                ax1 = fig.add_subplot(131, projection='3d')
105
                ax1.scatter(points[:, 0], points[:, 1], points[:, 2], c='blue',
106
                ax1.set title('Original Coordinates')
107
                ax1.set_xlabel('X')
108
                ax1.set_ylabel('Y')
109
                ax1.set_zlabel('Z')
110
111
112
                # Normalized coordinates
                norm_points = self.utils.normalize_points(points)
113
114
                ax2 = fig.add_subplot(132, projection='3d')
                ax2.scatter(norm_points[:, 0], norm_points[:, 1], norm_points[:,
115
                           c='red', marker='.')
116
                ax2.set_title('Normalized Coordinates')
117
118
                ax2.set_xlabel('X')
```

```
119
                ax2.set_ylabel('Y')
                ax2.set_zlabel('Z')
120
121
122
                # Spherical coordinates view
                sph_points = self.utils.cartesian_to_spherical(points)
123
                sph_cart = self.utils.spherical_to_cartesian(sph_points)
124
125
                ax3 = fig.add_subplot(133, projection='3d')
                ax3.scatter(sph_cart[:, 0], sph_cart[:, 1], sph_cart[:, 2],
126
                           c='green', marker='.')
127
                ax3.set_title('Spherical Coordinates')
128
129
                ax3.set_xlabel('X')
                ax3.set_ylabel('Y')
130
                ax3.set_zlabel('Z')
131
132
133
                plt.tight_layout()
                plt.show()
134
135
136
                # Print ranges
137
                print("\nPoint Cloud Statistics:")
138
                print(f"Original range: [{points.min():.4f}, {points.max():.4f}]
                print(f"Normalized range: [{norm_points.min():.4f}, {norm_points
139
140
                print(f"Spherical coordinates shape: {sph_points.shape}")
141
142
                return points
143
144
            except Exception as e:
                print(f"Error in enhanced visualization: {e}")
145
146
                raise
147
        def compare frames(self, frame indices):
148
            """Compare multiple frames"""
149
150
            try:
                n_frames = len(frame_indices)
151
                fig = plt.figure(figsize=(5*n_frames, 5))
152
153
154
                for i, idx in enumerate(frame_indices):
155
                    ax = fig.add subplot(1, n frames, i+1, projection='3d')
156
                    points = self.train data[idx]['new x']
157
                    ax.scatter(points[:, 0], points[:, 1], points[:, 2],
158
159
                              c='blue', marker='.')
                    ax.set_title(f'Frame {idx}')
160
161
                    ax.set_xlabel('X')
162
                    ax.set_ylabel('Y')
                    ax.set zlabel('Z')
163
164
165
                plt.tight_layout()
                plt.show()
166
167
            except Exception as e:
168
                print(f"Error comparing frames: {e}")
169
```

```
170
                raise
171
172 # Example usage
173 if __name__ == "__main__":
174
       data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
       analyzer = EnhancedMMRAnalyzer(data_path)
175
176
177
       # Basic visualization
178
       print("\nBasic Visualization:")
179
       analyzer.visualize_frame(0)
180
       # Enhanced visualization
181
       print("\nEnhanced Visualization:")
182
183
       analyzer.visualize_frame_enhanced(0)
184
185
       # Compare multiple frames
       print("\nComparing Multiple Frames:")
186
       analyzer.compare_frames([0, 1, 2])
187
 Loaded 170336 training samples
     Basic Visualization:
     Frame Statistics:
     Number of points: 110
     X range: [-1.1581, 1.2027]
     Y range: [0.0000, 3.8908]
     Z range: [-0.8684, 0.6147]
```

Frame 0 Point Cloud





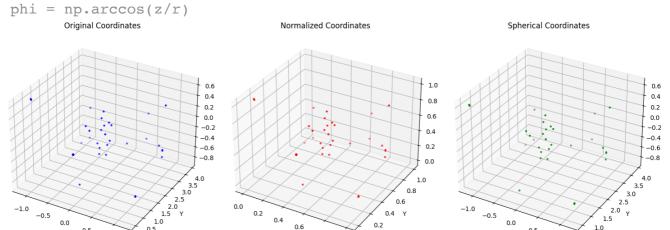
Enhanced Visualization:

Geometric Analysis:

Center of mass: [0.04778343 0.53634458 -0.03896176]

Mean distance from center: 0.8495 Max distance from center: 3.3704

<ipython-input-9-f8f139a9df25>:23: RuntimeWarning: invalid value encountere

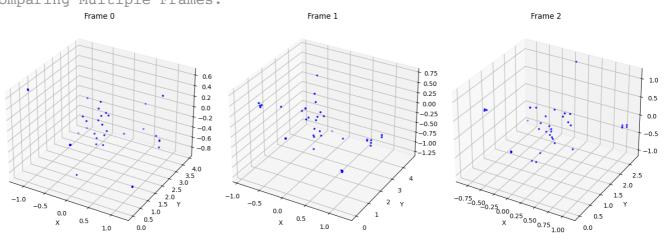


1.0

Point Cloud Statistics:

Original range: [-1.1581, 3.8908]
Normalized range: [0.0000, 1.0000]
Spherical coordinates shape: (110, 3)

Comparing Multiple Frames:



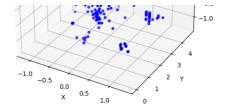
Choose frame window

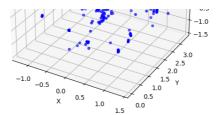
```
1 \text{ WINDOW} = 5
 1 def analyze_mmr_windows(data_path, window_size=5):
       """Analyze MMR data with sliding windows"""
 2
 3
 4
       # Load data
 5
       with open(data_path, 'rb') as f:
           all_data = pickle.load(f)
 6
 7
       train_data = all_data['train']
 8
       print(f"Analyzing {len(train_data)} frames with window size {window_size}
 9
10
       # Single frame analysis
11
       frame_points = []
12
       for data_point in train_data:
13
           points = data_point['new_x']
14
15
           frame_points.append(len(points))
16
       frame_series = pd.Series(frame_points)
17
       print("\nPoints per Frame Statistics:")
18
       print(frame_series.describe())
19
20
21
       # Window analysis
       window_points = []
22
```

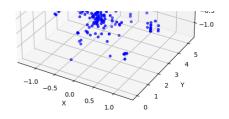
```
23
       num_windows = len(train_data) // window_size
24
25
      for i in range(num windows):
26
           start idx = i * window size
           end_idx = start_idx + window_size
27
           window_count = sum(frame_points[start_idx:end_idx])
28
29
           window_points.append(window_count)
30
      window_series = pd.Series(window_points)
31
       print(f"\nPoints per {window size}-Frame Window Statistics:")
32
      print(window_series.describe())
33
34
      # Visualize first window
35
      visualize_window(train_data, 0, window_size)
36
37
      # Compare first few windows
38
      compare_windows(train_data, [0, 1, 2], window_size)
39
40
41
       return frame_series, window_series
42
43 def visualize_window(train_data, window_idx, window_size):
44
      """Visualize all points in a window"""
       start idx = window idx * window size
45
46
      end_idx = start_idx + window_size
47
      # Collect points
48
      all points = []
49
      frame_counts = []
50
51
      for i in range(start idx, min(end idx, len(train data))):
52
           points = train_data[i]['new_x']
53
           all points.append(points)
54
           frame_counts.append(len(points))
55
56
57
      all_points = np.vstack(all_points)
58
59
      # Create visualization
      fig = plt.figure(figsize=(15, 5))
60
61
      # 3D scatter plot
62
      ax1 = fig.add_subplot(121, projection='3d')
63
64
      scatter = ax1.scatter(all_points[:, 0], all_points[:, 1], all_points[:,
                            c='blue', marker='.', s=50)
65
      ax1.set xlabel('X')
66
      ax1.set_ylabel('Y')
67
      ax1.set_zlabel('Z')
68
      ax1.set_title(f'Window {window_idx} (Frames {start_idx}-{end_idx-1})')
69
70
      ax1.grid(True)
71
72
      # Point count distribution
      ax2 = fig.add_subplot(122)
73
```

```
74
        ax2.bar(range(start_idx, end_idx), frame_counts)
 75
        ax2.set_xlabel('Frame Number')
 76
        ax2.set_ylabel('Number of Points')
        ax2.set title('Points per Frame in Window')
 77
 78
        ax2.grid(True)
 79
 80
        plt.tight layout()
        display(plt.gcf())
 81
        plt.close()
 82
 83
 84
       # Print statistics
        print(f"\nWindow {window_idx} Statistics:")
 85
        print(f"Total frames: {len(frame counts)}")
 86
 87
        print(f"Total points: {len(all_points)}")
        print(f"Average points per frame: {np.mean(frame_counts):.2f}")
 88
        print(f"Point range: [{all_points.min():.4f}, {all_points.max():.4f}]")
 89
 90
 91
        return all_points, frame_counts
 92
 93 def compare_windows(train_data, window_indices, window_size):
        """Compare multiple windows"""
 94
 95
        n windows = len(window indices)
        fig = plt.figure(figsize=(5*n windows, 5))
 96
 97
        for i, idx in enumerate(window_indices):
 98
            start_idx = idx * window_size
 99
            end idx = start idx + window size
100
101
            # Collect window points
102
            all points = []
103
            for j in range(start_idx, min(end_idx, len(train_data))):
104
                points = train data[i]['new x']
105
                all_points.append(points)
106
107
            all_points = np.vstack(all_points)
108
109
            # Plot
110
            ax = fig.add subplot(1, n windows, i+1, projection='3d')
            ax.scatter(all_points[:, 0], all_points[:, 1], all_points[:, 2],
111
                      c='blue', marker='.', s=50)
112
            ax.set title(f'Window {idx}\n(Frames {start idx}-{end idx-1})')
113
            ax.set_xlabel('X')
114
115
            ax.set_ylabel('Y')
            ax.set_zlabel('Z')
116
            ax.grid(True)
117
118
        plt.tight_layout()
119
120
        display(plt.gcf())
        plt.close()
121
122
123 # Example usage
124 if __name__ == "__main__":
```

```
125
         # Set parameters
         data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
126
127
         WINDOW_SIZE = 5
128
129
         # Run analysis
130
         frame_stats, window_stats = analyze_mmr_windows(data_path, WINDOW_SIZE)
 \rightarrow
      Analyzing 170336 frames with window size 5
      Points per Frame Statistics:
      count
                 170336.0
                     110.0
      mean
                        0.0
      std
      min
                     110.0
      25%
                     110.0
                     110.0
      50%
      75%
                     110.0
                     110.0
      max
      dtype: float64
      Points per 5-Frame Window Statistics:
      count
                 34067.0
                    550.0
      mean
                      0.0
      std
      min
                    550.0
      25%
                    550.0
      50%
                    550.0
      75%
                    550.0
                    550.0
      dtype: float64
              Window 0 (Frames 0-4)
                                                                 Points per Frame in Window
                                             100
                                             80
                                   0.5
                                           Number of Points
                                   0.0 Z
                                             60
                                   -0.5
                                   -1.0
                                             40
         -1.0 <sub>-0.5</sub> <sub>0.0</sub>
                                             20
                 0.5
                                                                      Frame Number
      Window 0 Statistics:
      Total frames: 5
      Total points: 550
      Average points per frame: 110.00
      Point range: [-1.3165, 4.7211]
                  Window 0
                                                  Window 1
                                                                                  Window 2
                  (Frames 0-4)
                                                 (Frames 5-9)
                                                                                (Frames 10-14)
                                                                   1.0
```







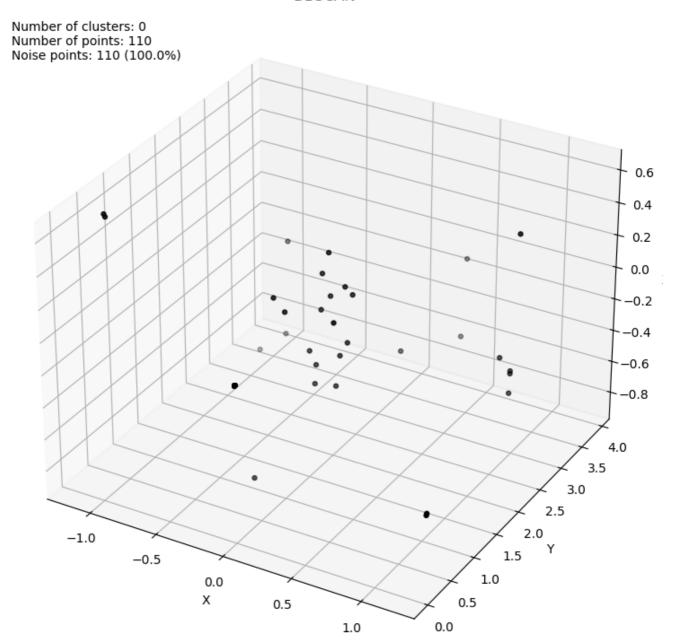
DBSCAN clustering

```
1 from sklearn.cluster import DBSCAN
 1 # Define global variables
 2 DBSCAN EPS = 0.3
 3 DBSCAN_SAMPLES = 6
 1 import numpy as np
 2 import matplotlib.pyplot as plt
 3 from sklearn.preprocessing import MinMaxScaler
 5 def get_colors(label_list):
      """Get colors for cluster labels"""
     7
8
      return [color_list[label % len(color_list)] for label in label_list]
9
10
11 def normalize weights(points):
      """Normalize point weights to [0,1]"""
12
      return MinMaxScaler().fit_transform(
13
         np.linalg.norm(points, axis=1).reshape(-1, 1)
14
      ).ravel()
15
```

```
16
17 def model_cluster(points, model, use_weights=True, show_plot=True, return_clu
18
19
      Cluster point cloud data using the provided model
20
21
      Args:
           points: numpy array of shape (N, 3) containing point cloud coordinate
22
23
           model: clustering model (e.g., DBSCAN, KMeans)
24
           use_weights: whether to use point distances as weights
25
           show_plot: whether to show clustering visualization
26
           return_cluster: whether to return the clustering model
27
28
      Returns:
           clustering model if return_cluster is True
29
      .....
30
31
      # Prepare weights if needed
32
      if use_weights:
           sample_weights = normalize_weights(points)
33
34
      else:
35
           sample_weights = None
36
37
      # Perform clustering
38
      clustering = model.fit(points, sample_weight=sample_weights)
39
40
      # Visualize if requested
      if show_plot:
41
           fig = plt.figure(figsize=(12, 8))
42
           ax = fig.add_subplot(111, projection='3d')
43
44
45
           # Plot points colored by cluster
           scatter = ax.scatter(points[:, 0], points[:, 1], points[:, 2],
46
                              c=get_colors(clustering.labels_),
47
48
                              marker='.',
                              s=50)
49
50
51
           # Add cluster centers if available (e.g., for KMeans)
52
           if hasattr(model, 'cluster_centers_'):
53
               centers = model.cluster centers
54
               ax.scatter(centers[:, 0], centers[:, 1], centers[:, 2],
55
                         c='black',
                         marker='*',
56
57
                         s=200,
58
                         label='Cluster Centers')
59
           # Configure plot
60
           ax.set xlabel('X')
61
           ax.set_ylabel('Y')
62
63
           ax.set zlabel('Z')
           ax.set_title(f'Clustering Results\n{model.__class__.__name__}')
64
65
           ax.grid(True)
66
           # Add statistics annotation
67
```

```
n_clusters = len(set(clustering.labels_)) - (1 if -1 in clustering.la
 68
           stats_text = f'Number of clusters: {n_clusters}\n'
 69
           stats_text += f'Number of points: {len(points)}\n'
 70
 71
           if -1 in clustering.labels_:
               n_noise = np.sum(clustering.labels_ == -1)
72
 73
               stats_text += f'Noise points: {n_noise} ({n_noise/len(points)*100}
           ax.text2D(0.02, 0.98, stats_text,
 74
 75
                     transform=ax.transAxes,
                     verticalalignment='top')
 76
 77
 78
           plt.tight_layout()
 79
           display(plt.gcf())
           plt.close()
 80
 81
           # Print cluster sizes
 82
 83
           print("\nCluster Sizes:")
           unique_labels = sorted(set(clustering.labels_))
 84
 85
           for label in unique_labels:
               count = np.sum(clustering.labels_ == label)
 86
               if label == -1:
 87
                   print(f"Noise points: {count} ({count/len(points)*100:.1f}%)"
 88
               else:
 89
 90
                   print(f"Cluster {label}: {count} points ({count/len(points)*1
 91
 92
       if return_cluster:
           return clustering
 93
 94
 95 # Example usage
 96 if __name__ == "__main__":
 97
       # Load your data
       data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/data
98
       with open(data_path, 'rb') as f:
99
           data = pickle.load(f)
100
101
       # Get points from first frame
102
       points = data['train'][0]['new x']
103
104
       # Try different clustering models
105
       from sklearn.cluster import DBSCAN, KMeans
106
107
108
       # DBSCAN clustering
       dbscan = DBSCAN(eps=0.3, min samples=5)
109
       print("\nDBSCAN Clustering:")
110
       dbscan_result = model_cluster(points, dbscan, use_weights=True, return_cl
111
112
       # KMeans clustering
113
       kmeans = KMeans(n_clusters=5, random_state=42)
114
115
       print("\nKMeans Clustering:")
116
       kmeans_result = model_cluster(points, kmeans, use_weights=True, return_cl
```

Clustering Results DBSCAN



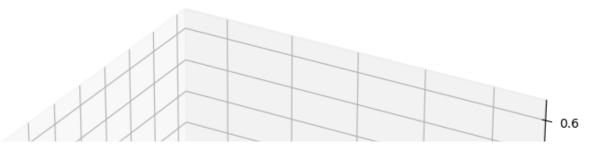
Cluster Sizes:

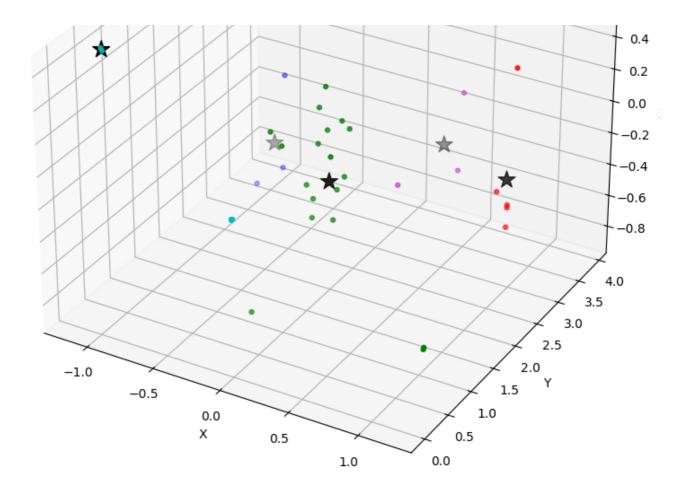
Noise points: 110 (100.0%)

KMeans Clustering:

Clustering Results KMeans

Number of clusters: 5 Number of points: 110





```
Cluster Sizes:
Cluster 0: 5 points (4.5%)
Cluster 1: 3 points (2.7%)
Cluster 2: 18 points (16.4%)
Cluster 3: 81 points (73.6%)
Cluster 4: 3 points (2.7%)
```

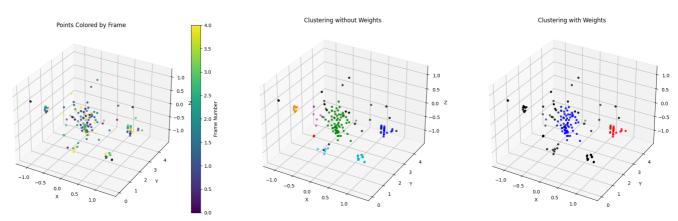
Compare with and without sample_weight

```
3
       Cluster points in a specific frame window
 4
 5
      Args:
 6
           data points: List of frame data points
 7
           start_frame: Starting frame index
           window size: Number of frames in window
 8
           eps: DBSCAN epsilon parameter
 9
           min_samples: DBSCAN min_samples parameter
10
           use_weights: Whether to use intensity weights
11
       .....
12
13
      # Collect points from window
14
      window points = []
       frame_indices = []
15
16
17
       end_frame = start_frame + window_size
       for i in range(start_frame, end_frame):
18
           if i < len(data points):</pre>
19
20
               points = data_points[i]['new_x']
21
               window_points.append(points)
22
               frame_indices.extend([i] * len(points))
23
24
      window_points = np.vstack(window_points)
25
       frame indices = np.array(frame indices)
26
      # Create DBSCAN model
27
28
       dbscan = DBSCAN(eps=eps, min_samples=min_samples)
29
30
      # Prepare sample weights if needed
31
       if use_weights:
           weights = normalize weights(window points)
32
33
       else:
34
           weights = None
35
36
      # Perform clustering
       clustering = dbscan.fit(window_points, sample_weight=weights)
37
38
39
      # Visualize results
       fig = plt.figure(figsize=(20, 6))
40
41
      # Original points colored by frame
42
       ax1 = fig.add_subplot(131, projection='3d')
43
44
       scatter1 = ax1.scatter(window_points[:, 0], window_points[:, 1], window_
                             c=frame_indices, cmap='viridis',
45
                             marker='.', s=50)
46
       ax1.set_title('Points Colored by Frame')
47
       ax1.set_xlabel('X')
48
       ax1.set_ylabel('Y')
49
       ax1.set zlabel('Z')
50
       plt.colorbar(scatter1, label='Frame Number')
51
52
53
      # Clustering without weights
```

```
54
       dbscan_no_weights = DBSCAN(eps=eps, min_samples=min_samples)
        labels_no_weights = dbscan_no_weights.fit_predict(window_points)
 55
 56
       ax2 = fig.add subplot(132, projection='3d')
 57
 58
        scatter2 = ax2.scatter(window_points[:, 0], window_points[:, 1], window_
                              c=get_colors(labels_no_weights),
 59
                              marker='.', s=50)
 60
       ax2.set_title('Clustering without Weights')
 61
 62
       ax2.set_xlabel('X')
       ax2.set_ylabel('Y')
 63
 64
       ax2.set_zlabel('Z')
 65
       # Clustering with weights
 66
 67
       ax3 = fig.add_subplot(133, projection='3d')
       scatter3 = ax3.scatter(window_points[:, 0], window_points[:, 1], window_
 68
                              c=get_colors(clustering.labels_),
 69
                              marker='.', s=50)
 70
 71
       ax3.set_title('Clustering with Weights')
 72
       ax3.set_xlabel('X')
 73
       ax3.set_ylabel('Y')
 74
       ax3.set_zlabel('Z')
 75
 76
       plt.tight layout()
 77
       display(plt.gcf())
 78
       plt.close()
 79
 80
       # Print statistics
       print(f"\nWindow Statistics (Frames {start_frame}-{end_frame-1}):")
 81
       print(f"Total points: {len(window_points)}")
 82
 83
       print("\nClustering without weights:")
 84
       n_clusters_no_weights = len(set(labels_no_weights)) - (1 if -1 in labels
 85
       print(f"Number of clusters: {n_clusters_no_weights}")
 86
 87
        if -1 in labels_no_weights:
            n_noise = np.sum(labels_no_weights == -1)
 88
 89
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 90
 91
       print("\nClustering with weights:")
       n_clusters = len(set(clustering.labels_)) - (1 if -1 in clustering.label
 92
       print(f"Number of clusters: {n clusters}")
 93
       if -1 in clustering.labels:
 94
 95
            n_noise = np.sum(clustering.labels_ == -1)
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 96
 97
 98
        return clustering, window_points, frame_indices
 99
100 def analyze_window_clusters(clustering, window_points, frame_indices):
       """Analyze clusters in a window"""
101
       unique_labels = sorted(set(clustering.labels_))
102
103
       # Create visualization
104
```

```
105
        n_clusters = len(unique_labels)
        fig_cols = min(3, n_clusters)
106
107
        fig_rows = (n_clusters + fig_cols - 1) // fig_cols
108
        fig = plt.figure(figsize=(6*fig_cols, 5*fig_rows))
109
110
111
        for i, label in enumerate(unique_labels):
112
            mask = clustering.labels_ == label
113
            cluster_points = window_points[mask]
            cluster_frames = frame_indices[mask]
114
115
116
            ax = fig.add_subplot(fig_rows, fig_cols, i+1, projection='3d')
            scatter = ax.scatter(cluster_points[:, 0],
117
118
                               cluster_points[:, 1],
119
                               cluster_points[:, 2],
                                c=cluster_frames,
120
121
                                cmap='viridis',
122
                               marker='.',
123
                                s=50)
124
125
            title = "Noise Points" if label == -1 else f"Cluster {label}"
126
            title += f"\n{len(cluster_points)} points"
            ax.set title(title)
127
            ax.set_xlabel('X')
128
            ax.set_ylabel('Y')
129
            ax.set_zlabel('Z')
130
            plt.colorbar(scatter, label='Frame Number')
131
132
        plt.tight_layout()
133
134
        display(plt.gcf())
        plt.close()
135
136
137 # Example usage
138 if __name__ == "__main__":
        # Parameters
139
140
       WINDOW = 5
141
        DBSCAN EPS = 0.3
142
        DBSCAN SAMPLES = 5
143
        START_FRAME = 0
144
145
        # Load data
146
        data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
        with open(data_path, 'rb') as f:
147
            data = pickle.load(f)
148
149
150
        # Perform clustering
        clustering, points, frames = cluster_window(
151
            data['train'],
152
153
            start_frame=START_FRAME,
154
            window_size=WINDOW,
            eps=DBSCAN_EPS,
155
```

```
min_samples=DBSCAN_SAMPLES
157
        )
158
159
       # Analyze clusters
       analyze_window_clusters(clustering, points, frames)
160
```



Window Statistics (Frames 0-4):

Total points: 550

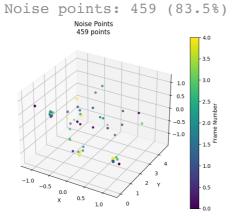
156

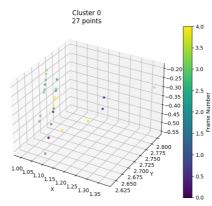
 $\overline{2}$

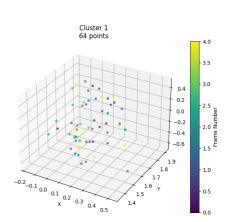
Clustering without weights:

Number of clusters: 7 Noise points: 20 (3.6%)

Clustering with weights: Number of clusters: 2





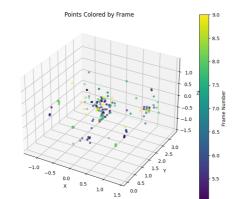


```
1 def cluster_window(data_points, start_frame, window_size, eps=0.3, min_sampl
 2
 3
       Cluster points in a specific frame window
 4
 5
       Args:
           data_points: List of frame data points
 6
 7
           start_frame: Starting frame index
           window size: Number of frames in window
 8
           eps: DBSCAN epsilon parameter
 9
           min_samples: DBSCAN min_samples parameter
10
           use_weights: Whether to use intensity weights
11
       .....
12
      # Collect points from window
13
14
      window_points = []
       frame_indices = []
15
16
17
       end_frame = start_frame + window_size
18
       for i in range(start_frame, end_frame):
19
           if i < len(data points):</pre>
               points = data_points[i]['new_x']
20
21
               window_points.append(points)
22
               frame_indices.extend([i] * len(points))
23
24
      window_points = np.vstack(window_points)
25
       frame_indices = np.array(frame_indices)
26
27
      # Create DBSCAN model
       dbscan = DBSCAN(eps=eps, min_samples=min_samples)
28
29
30
      # Prepare sample weights if needed
31
       if use weights:
32
           weights = normalize_weights(window_points)
33
      else:
34
           weights = None
35
36
      # Perform clustering
37
       clustering = dbscan.fit(window_points, sample_weight=weights)
38
39
      # Visualize results
       fig = plt.figure(figsize=(20, 6))
40
41
      # Original points colored by frame
42
       ax1 = fig.add_subplot(131, projection='3d')
43
       scatter1 = ax1.scatter(window_points[:, 0], window_points[:, 1], window_
44
45
                             c=frame_indices, cmap='viridis',
                             marker='.', s=50)
46
47
       ax1.set_title('Points Colored by Frame')
48
       ax1.set_xlabel('X')
49
       ax1.set_ylabel('Y')
50
       ax1.set zlabel('Z')
       plt.colorbar(scatter1, label='Frame Number')
51
```

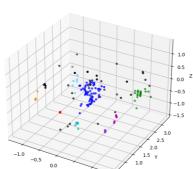
```
52
       # Clustering without weights
 53
       dbscan_no_weights = DBSCAN(eps=eps, min_samples=min_samples)
 54
        labels_no_weights = dbscan_no_weights.fit_predict(window_points)
 55
 56
       ax2 = fig.add_subplot(132, projection='3d')
 57
 58
        scatter2 = ax2.scatter(window_points[:, 0], window_points[:, 1], window_
 59
                              c=get_colors(labels_no_weights),
                              marker='.', s=50)
 60
       ax2.set_title('Clustering without Weights')
 61
 62
       ax2.set_xlabel('X')
       ax2.set_ylabel('Y')
 63
       ax2.set_zlabel('Z')
 64
 65
       # Clustering with weights
 66
       ax3 = fig.add_subplot(133, projection='3d')
 67
        scatter3 = ax3.scatter(window_points[:, 0], window_points[:, 1], window_
 68
                              c=get_colors(clustering.labels_),
 69
                              marker='.', s=50)
 70
 71
       ax3.set_title('Clustering with Weights')
 72
       ax3.set_xlabel('X')
 73
       ax3.set_ylabel('Y')
 74
       ax3.set zlabel('Z')
 75
       plt.tight_layout()
 76
 77
       display(plt.gcf())
 78
       plt.close()
 79
 80
       # Print statistics
       print(f"\nWindow Statistics (Frames {start frame}-{end frame-1}):")
 81
       print(f"Total points: {len(window_points)}")
 82
 83
       print("\nClustering without weights:")
 84
        n_clusters_no_weights = len(set(labels_no_weights)) - (1 if -1 in labels
 85
       print(f"Number of clusters: {n_clusters_no_weights}")
 86
 87
        if -1 in labels_no_weights:
            n noise = np.sum(labels no weights == -1)
 88
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 89
 90
       print("\nClustering with weights:")
 91
       n_clusters = len(set(clustering.labels_)) - (1 if -1 in clustering.label
 92
 93
       print(f"Number of clusters: {n_clusters}")
        if -1 in clustering.labels_:
 94
            n_noise = np.sum(clustering.labels_ == -1)
 95
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 96
 97
 98
        return clustering, window_points, frame_indices
 99
100 def analyze_window_clusters(clustering, window_points, frame_indices):
        """Analyze clusters in a window"""
101
        unique_labels = sorted(set(clustering.labels_))
102
```

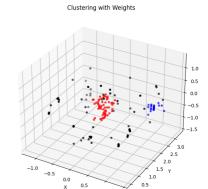
```
103
104
       # Create visualization
105
        n_clusters = len(unique_labels)
        fig cols = min(3, n clusters)
106
        fig_rows = (n_clusters + fig_cols - 1) // fig_cols
107
108
        fig = plt.figure(figsize=(6*fig cols, 5*fig rows))
109
110
        for i, label in enumerate(unique_labels):
111
            mask = clustering.labels_ == label
112
            cluster_points = window_points[mask]
113
            cluster_frames = frame_indices[mask]
114
115
116
            ax = fig.add_subplot(fig_rows, fig_cols, i+1, projection='3d')
            scatter = ax.scatter(cluster_points[:, 0],
117
                               cluster_points[:, 1],
118
                               cluster_points[:, 2],
119
                               c=cluster_frames,
120
121
                               cmap='viridis',
122
                               marker='.',
123
                               s=50)
124
125
            title = "Noise Points" if label == -1 else f"Cluster {label}"
            title += f"\n{len(cluster_points)} points"
126
            ax.set_title(title)
127
            ax.set_xlabel('X')
128
129
            ax.set ylabel('Y')
            ax.set_zlabel('Z')
130
            plt.colorbar(scatter, label='Frame Number')
131
132
        plt.tight_layout()
133
        display(plt.gcf())
134
135
        plt.close()
136
137 # Example usage
138 if __name__ == "__main__":
139
       # Parameters
140
       WINDOW = 5
       DBSCAN\_EPS = 0.3
141
       DBSCAN SAMPLES = 5
142
143
       START_FRAME = 5
144
145
       # Load data
       data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
146
       with open(data_path, 'rb') as f:
147
148
            data = pickle.load(f)
149
       # Perform clustering
150
        clustering, points, frames = cluster_window(
151
152
            data['train'],
            start_frame=START_FRAME,
153
```











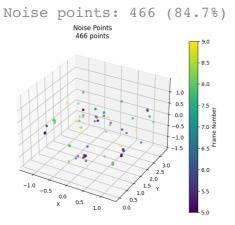
Window Statistics (Frames 5-9):

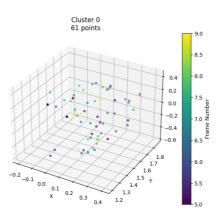
Total points: 550

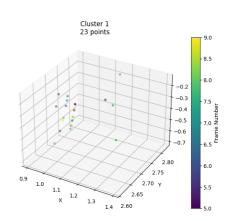
Clustering without weights:

Number of clusters: 8
Noise points: 35 (6.4%)

Clustering with weights: Number of clusters: 2







1 def cluster_window(data_points, start_frame, window_size, eps=0.3, min_sampl
2 """

3 Cluster points in a specific frame window

4 5

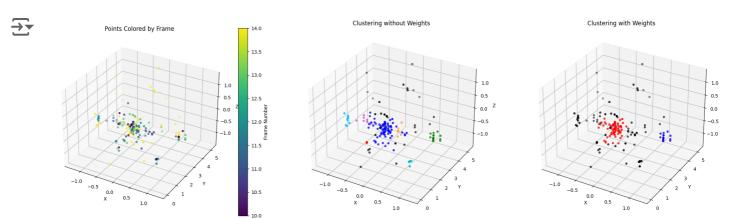
Args:

```
6
           data_points: List of frame data points
 7
           start_frame: Starting frame index
 8
           window_size: Number of frames in window
 9
           eps: DBSCAN epsilon parameter
           min_samples: DBSCAN min_samples parameter
10
           use_weights: Whether to use intensity weights
11
       .....
12
      # Collect points from window
13
14
      window_points = []
       frame_indices = []
15
16
17
       end_frame = start_frame + window_size
       for i in range(start_frame, end_frame):
18
19
           if i < len(data_points):</pre>
20
               points = data_points[i]['new_x']
               window_points.append(points)
21
22
               frame_indices.extend([i] * len(points))
23
24
      window_points = np.vstack(window_points)
25
       frame_indices = np.array(frame_indices)
26
27
      # Create DBSCAN model
       dbscan = DBSCAN(eps=eps, min samples=min samples)
28
29
30
      # Prepare sample weights if needed
31
       if use_weights:
32
           weights = normalize weights(window points)
33
      else:
34
           weights = None
35
      # Perform clustering
36
       clustering = dbscan.fit(window_points, sample_weight=weights)
37
38
39
      # Visualize results
       fig = plt.figure(figsize=(20, 6))
40
41
42
      # Original points colored by frame
       ax1 = fig.add_subplot(131, projection='3d')
43
       scatter1 = ax1.scatter(window_points[:, 0], window_points[:, 1], window_
44
                             c=frame indices, cmap='viridis',
45
                             marker='.', s=50)
46
47
       ax1.set_title('Points Colored by Frame')
       ax1.set_xlabel('X')
48
       ax1.set_ylabel('Y')
49
       ax1.set zlabel('Z')
50
       plt.colorbar(scatter1, label='Frame Number')
51
52
      # Clustering without weights
53
       dbscan_no_weights = DBSCAN(eps=eps, min_samples=min_samples)
54
55
       labels_no_weights = dbscan_no_weights.fit_predict(window_points)
56
```

```
57
       ax2 = fig.add_subplot(132, projection='3d')
        scatter2 = ax2.scatter(window_points[:, 0], window_points[:, 1], window_
 58
 59
                              c=get_colors(labels_no_weights),
                              marker='.', s=50)
 60
       ax2.set_title('Clustering without Weights')
 61
       ax2.set_xlabel('X')
 62
 63
       ax2.set ylabel('Y')
       ax2.set_zlabel('Z')
 64
 65
       # Clustering with weights
 66
       ax3 = fig.add_subplot(133, projection='3d')
 67
       scatter3 = ax3.scatter(window_points[:, 0], window_points[:, 1], window_
 68
                              c=get_colors(clustering.labels_),
 69
 70
                              marker='.', s=50)
       ax3.set_title('Clustering with Weights')
 71
 72
       ax3.set_xlabel('X')
 73
       ax3.set ylabel('Y')
 74
       ax3.set_zlabel('Z')
 75
 76
       plt.tight layout()
 77
       display(plt.gcf())
 78
       plt.close()
 79
 80
       # Print statistics
       print(f"\nWindow Statistics (Frames {start_frame}-{end_frame-1}):")
 81
       print(f"Total points: {len(window_points)}")
 82
 83
       print("\nClustering without weights:")
 84
       n_clusters_no_weights = len(set(labels_no_weights)) - (1 if -1 in labels
 85
       print(f"Number of clusters: {n clusters no weights}")
 86
       if -1 in labels_no_weights:
 87
            n noise = np.sum(labels no weights == -1)
 88
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 89
 90
       print("\nClustering with weights:")
 91
       n_{clusters} = len(set(clustering.labels_)) - (1 if -1 in clustering.label)
 92
 93
       print(f"Number of clusters: {n clusters}")
 94
       if -1 in clustering.labels:
            n_noise = np.sum(clustering.labels_ == -1)
 95
            print(f"Noise points: {n noise} ({n noise/len(window points)*100:.11
 96
 97
 98
        return clustering, window_points, frame_indices
 99
100 def analyze_window_clusters(clustering, window_points, frame_indices):
       """Analyze clusters in a window"""
101
       unique_labels = sorted(set(clustering.labels_))
102
103
       # Create visualization
104
       n_clusters = len(unique_labels)
105
       fig_cols = min(3, n_clusters)
106
       fig_rows = (n_clusters + fig_cols - 1) // fig_cols
107
```

```
108
        fig = plt.figure(figsize=(6*fig_cols, 5*fig_rows))
109
110
111
        for i, label in enumerate(unique labels):
            mask = clustering.labels_ == label
112
            cluster_points = window_points[mask]
113
            cluster_frames = frame_indices[mask]
114
115
            ax = fig.add_subplot(fig_rows, fig_cols, i+1, projection='3d')
116
            scatter = ax.scatter(cluster_points[:, 0],
117
                               cluster_points[:, 1],
118
                                cluster_points[:, 2],
119
                                c=cluster_frames,
120
121
                                cmap='viridis',
122
                               marker='.',
                                s=50)
123
124
            title = "Noise Points" if label == -1 else f"Cluster {label}"
125
126
            title += f"\n{len(cluster_points)} points"
127
            ax.set title(title)
            ax.set_xlabel('X')
128
129
            ax.set_ylabel('Y')
            ax.set zlabel('Z')
130
            plt.colorbar(scatter, label='Frame Number')
131
132
        plt.tight_layout()
133
        display(plt.gcf())
134
        plt.close()
135
136
137 # Example usage
138 if __name__ == "__main__":
        # Parameters
139
140
       WINDOW = 5
141
        DBSCAN EPS = 0.3
        DBSCAN SAMPLES = 5
142
143
        START_FRAME = 10
144
145
       # Load data
        data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
146
        with open(data path, 'rb') as f:
147
148
            data = pickle.load(f)
149
150
        # Perform clustering
        clustering, points, frames = cluster_window(
151
            data['train'],
152
153
            start_frame=START_FRAME,
            window_size=WINDOW,
154
            eps=DBSCAN_EPS,
155
            min_samples=DBSCAN_SAMPLES
156
        )
157
158
```

Analyze clusters
analyze_window_clusters(clustering, points, frames)

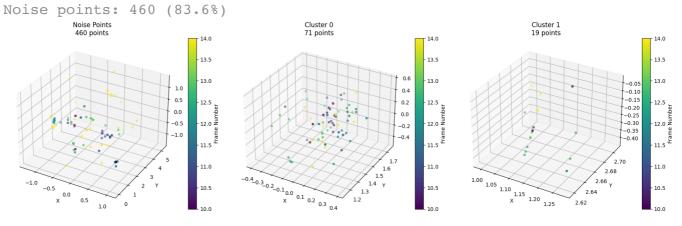


Window Statistics (Frames 10-14): Total points: 550

Clustering without weights:

Number of clusters: 7
Noise points: 31 (5.6%)

Clustering with weights: Number of clusters: 2



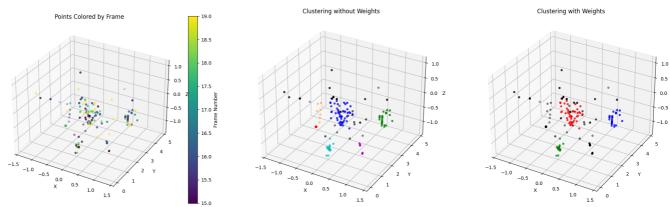
```
3
       Cluster points in a specific frame window
 4
 5
      Args:
 6
           data points: List of frame data points
 7
           start_frame: Starting frame index
           window size: Number of frames in window
 8
           eps: DBSCAN epsilon parameter
 9
           min_samples: DBSCAN min_samples parameter
10
           use_weights: Whether to use intensity weights
11
       .....
12
13
      # Collect points from window
14
      window points = []
       frame_indices = []
15
16
17
       end_frame = start_frame + window_size
       for i in range(start_frame, end_frame):
18
           if i < len(data points):</pre>
19
20
               points = data_points[i]['new_x']
21
               window_points.append(points)
22
               frame_indices.extend([i] * len(points))
23
24
      window_points = np.vstack(window_points)
25
       frame indices = np.array(frame indices)
26
      # Create DBSCAN model
27
28
       dbscan = DBSCAN(eps=eps, min_samples=min_samples)
29
      # Prepare sample weights if needed
30
31
       if use_weights:
           weights = normalize weights(window points)
32
33
       else:
34
           weights = None
35
36
      # Perform clustering
       clustering = dbscan.fit(window_points, sample_weight=weights)
37
38
39
      # Visualize results
       fig = plt.figure(figsize=(20, 6))
40
41
      # Original points colored by frame
42
       ax1 = fig.add_subplot(131, projection='3d')
43
44
       scatter1 = ax1.scatter(window_points[:, 0], window_points[:, 1], window_
                             c=frame_indices, cmap='viridis',
45
                             marker='.', s=50)
46
       ax1.set_title('Points Colored by Frame')
47
       ax1.set_xlabel('X')
48
       ax1.set_ylabel('Y')
49
       ax1.set zlabel('Z')
50
       plt.colorbar(scatter1, label='Frame Number')
51
52
53
      # Clustering without weights
```

```
54
       dbscan_no_weights = DBSCAN(eps=eps, min_samples=min_samples)
        labels_no_weights = dbscan_no_weights.fit_predict(window_points)
 55
 56
       ax2 = fig.add subplot(132, projection='3d')
 57
        scatter2 = ax2.scatter(window_points[:, 0], window_points[:, 1], window_
 58
                              c=get_colors(labels_no_weights),
 59
                              marker='.', s=50)
 60
       ax2.set_title('Clustering without Weights')
 61
 62
       ax2.set_xlabel('X')
       ax2.set_ylabel('Y')
 63
 64
       ax2.set_zlabel('Z')
 65
       # Clustering with weights
 66
 67
       ax3 = fig.add_subplot(133, projection='3d')
        scatter3 = ax3.scatter(window_points[:, 0], window_points[:, 1], window_
 68
                              c=get_colors(clustering.labels_),
 69
                              marker='.', s=50)
 70
 71
       ax3.set_title('Clustering with Weights')
 72
       ax3.set_xlabel('X')
 73
       ax3.set_ylabel('Y')
 74
       ax3.set_zlabel('Z')
 75
 76
       plt.tight layout()
 77
       display(plt.gcf())
 78
       plt.close()
 79
 80
       # Print statistics
       print(f"\nWindow Statistics (Frames {start_frame}-{end_frame-1}):")
 81
       print(f"Total points: {len(window_points)}")
 82
 83
       print("\nClustering without weights:")
 84
       n_clusters_no_weights = len(set(labels_no_weights)) - (1 if -1 in labels
 85
       print(f"Number of clusters: {n_clusters_no_weights}")
 86
 87
        if -1 in labels_no_weights:
            n_noise = np.sum(labels_no_weights == -1)
 88
 89
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 90
 91
       print("\nClustering with weights:")
       n_clusters = len(set(clustering.labels_)) - (1 if -1 in clustering.label
 92
       print(f"Number of clusters: {n clusters}")
 93
       if -1 in clustering.labels:
 94
 95
            n_noise = np.sum(clustering.labels_ == -1)
            print(f"Noise points: {n_noise} ({n_noise/len(window_points)*100:.11
 96
 97
 98
        return clustering, window_points, frame_indices
 99
100 def analyze_window_clusters(clustering, window_points, frame_indices):
       """Analyze clusters in a window"""
101
       unique_labels = sorted(set(clustering.labels_))
102
103
       # Create visualization
104
```

```
105
        n_clusters = len(unique_labels)
        fig_cols = min(3, n_clusters)
106
107
        fig_rows = (n_clusters + fig_cols - 1) // fig_cols
108
        fig = plt.figure(figsize=(6*fig_cols, 5*fig_rows))
109
110
111
        for i, label in enumerate(unique_labels):
112
            mask = clustering.labels_ == label
113
            cluster_points = window_points[mask]
            cluster_frames = frame_indices[mask]
114
115
116
            ax = fig.add_subplot(fig_rows, fig_cols, i+1, projection='3d')
            scatter = ax.scatter(cluster_points[:, 0],
117
118
                               cluster_points[:, 1],
119
                               cluster_points[:, 2],
                                c=cluster_frames,
120
121
                                cmap='viridis',
122
                               marker='.',
123
                                s=50)
124
125
            title = "Noise Points" if label == -1 else f"Cluster {label}"
126
            title += f"\n{len(cluster_points)} points"
            ax.set title(title)
127
            ax.set_xlabel('X')
128
            ax.set_ylabel('Y')
129
            ax.set_zlabel('Z')
130
            plt.colorbar(scatter, label='Frame Number')
131
132
        plt.tight_layout()
133
134
        display(plt.gcf())
        plt.close()
135
136
137 # Example usage
138 if __name__ == "__main__":
        # Parameters
139
       WINDOW = 5
140
141
        DBSCAN EPS = 0.3
142
        DBSCAN SAMPLES = 5
143
        START_FRAME = 15
144
145
        # Load data
146
        data_path = '/content/drive/MyDrive/action/data/processed/mmr_action/dat
        with open(data_path, 'rb') as f:
147
            data = pickle.load(f)
148
149
150
        # Perform clustering
        clustering, points, frames = cluster_window(
151
            data['train'],
152
153
            start_frame=START_FRAME,
154
            window_size=WINDOW,
            eps=DBSCAN_EPS,
155
```

```
min_samples=DBSCAN_SAMPLES

157
)
158
159  # Analyze clusters
160  analyze_window_clusters(clustering, points, frames)
```



Window Statistics (Frames 15-19):

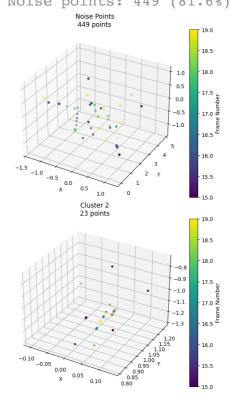
Total points: 550

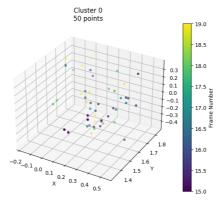
 $\overline{2}$

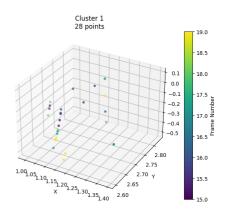
Clustering without weights:

Number of clusters: 6
Noise points: 28 (5.1%)

Clustering with weights: Number of clusters: 3 Noise points: 449 (81.6%)







Change DBSCAN eps, min_samples [TODO]

Estimate boundary for each cluster (extension, orientation)

```
1 # Clustering frame_num = 0~4
2 \text{ startFrame} = 0
3 dbscan = DBSCAN(eps=0.4, min_samples=10)
4 datalist = data[(data.frame_num>=startFrame)&(data.frame_num<startFrame+WINDO
6 clustering = ModelCluster(datalist, dbscan, sample_weight=True, show_plot=Fal
7 clustering.labels
1 # Number of points in each cluster
2 def ClusterAnalysis(labels):
     print('Total:', len(labels), 'points,', len(np.unique(labels))-1, 'cluste
3
4
     for i in range(np.max(np.unique(labels))+1):
         print('Cluster', i, ':', np.sum(labels==i), 'points')
5
     print('Noise:', np.sum(labels==-1), 'points')
1 # Number of points in each cluster
2 def ClusterAnalysis(labels):
     print('Total:', len(labels), 'points,', len(np.unique(labels))-1, 'cluste
3
4
     for i in range(np.max(np.unique(labels))+1):
         print('Cluster', i, ':', np.sum(labels==i), 'points')
5
     print('Noise:', np.sum(labels==-1), 'points')
```