G-PECNET: TOWARDS A GENERALIZABLE PEDESTRIAN TRAJECTORY PREDICTION SYSTEM

Aryan Garg[†]
aryangarg019@gmail.com

Renu M. Rameshan[†]
Lead Research Scientist
Vehant Technologies, India
renur@vehant.com

ABSTRACT

Navigating dynamic physical environments without obstructing or hurting humans is of quintessential importance for social robots. In this work, we solve autonomous *drone* navigation's sub-problem of predicting out-of-domain human and agent trajectories using a deep generative model. Here, we introduce General-PECNet that improves 9.5% on the Final Displacement Error (FDE) on 2020's benchmark: PECNet (Mangalam et al., 2020b) through a combination of architectural improvements inspired by periodic activation functions (Sitzmann et al., 2020) and synthetic trajectory (data) augmentations using Hidden Markov Models (HMMs) and Reinforcement Learning (RL). Additionally, we propose a simple geometry-inspired loss and evaluation metric for trajectory non-linearity analysis.

1 INTRODUCTION

Multimodal human or pedestrian trajectory prediction is an ill-posed problem of predicting the final and intermediate steps of some or all pedestrians when only a limited context of their previous trajectories and the scene is known. This is further complicated by implicit personal values and social rules that pre-define the pedestrians' interaction. Autonomous navigation for robots or social agents (Bennewitz et al., 2002), can only be enabled by accurate predictions for further downstream planning tasks. For the prediction problem, we contribute a) a novel reinforcement learning-based synthetic dataset and b) a variational autoencoder (Kingma & Welling, 2013) based pedestrian prediction network, which achieves state-of-the-art performance on the goal-point or final destination prediction error (FDE). G-PECNet is an improved adaptation of PECNet (Mangalam et al., 2020b).

2 METHOD

2.1 AUGMENTING WITH RL SYNTHETIC TRAJECTORIES

Synthetic trajectories were created using traditional Newtonian equations of motion and interaction modeling using a Hidden Markov Model. Finally, we train RL-based bots/agents deployed in the aforementioned interaction (HMM) model using Deep Policy Gradients (DPG). DPG agents were modeled with two major goals: reaching the destination quickly and avoiding collisions with fellow agents/pedestrians. Apart from acceleration, stopping for another crossing pedestrian (being considerate) was implicitly decided by the agent's *randomly pre-defined* sociability, fitness, and patience attributes. We add a circular proximity (fixed radius) detection mechanism to penalize agents that collide with others in the playground. Mathematically, the reward function at time step t: R_t for the agents to finally reach the goal G is defined by $R_t = AF^t \left(n_{ICS}+1\right)^{(AS+AP)} / \, t^2 \left(1+\|G-x_t\|_2\right)$ where AF, AP, and $AS \; \epsilon \; [0,1]$. x_t is the agent's current position, and n_{ICS} is the number of impending collision states. AS and AP are its sociability and patience respectively, determining its recklessness. AF: Agent's Fitness enforces reaching the goal quickly. Finally, the loss function is the one used in standard deep-policy gradients methods: $J(\phi) = -\sum_{t=1}^{t=N} log(P_\phi(a_t|s_t)) \, R_t$ where, $P(.)$ is parameterized by ϕ, a simple neural network that emulates the agent's action and state space at any time t. Training evolution is shown in figure 1.

[†] This work was done at the Indian Institute of Technology, Mandi. Code: Github

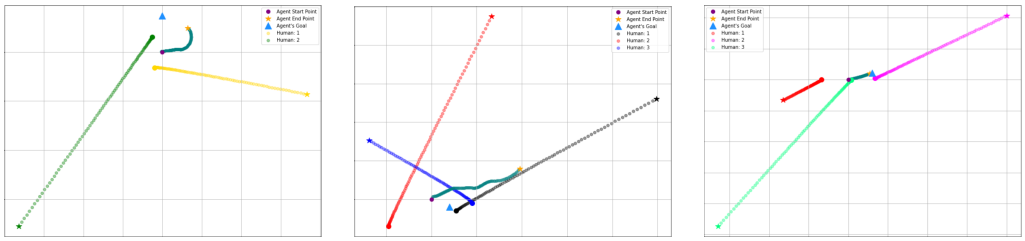

Figure 1: The RL agent is inserted and trained in an HMM interaction playground. Agent's trajectory is turquoise. Evolution of the samples produced. First, the agent learns to turn. The second depicts a complicated scene where the agent learns to avoid multiple collisions. The last scene depicts the agent successfully avoiding a collision and reaching its goal.

2.2 Periodic Activation: SIREN Improvement

We adapt PECNet to capture finer spatial and temporal details (Sitzmann et al., 2020) by replacing all ReLU (Agarap, 2018) activations with a simple sinusoidal function: $\mathbf{x_i} \rightarrow \phi(\mathbf{x_i}) = sin(\mathbf{W_i x_i} + \mathbf{b_i})$, where i denotes the i^{th} layer of the neural network. This choice is motivated by the finding from Sitzmann et al. (2020) that current neural network activations are insufficient for modeling high-frequency signals. They fail to represent a signal's spatial and temporal derivatives which are essential for the solution to the implicitly defined partial differential equations. We notice significant gains when SIRENs are added after our data augmentation. FDE improves by 41.4% when SIREN-infused PECNet is trained with the 6% augmented dataset compared to PECNet on our augmented dataset. See the ablation Tab. 6. Furthermore, we present a quantitative comparison in 3.

Learning Rate	ADE	FDE	Best FDE epoch
0.001	**22.20**	9.32	915
0.0005	29.91	9.05	834
0.0003	25.92	9.37	998
0.0002	26.75	**9.04**	908
0.0001	25.57	9.05	**235**

Table 1: State-of-the-art FDE of GPECNet: Trained on 6% augmented SDD with no standardization and decoupled ADE and FDE for 1000 epochs. We observe that no social pooling results in higher ADE.

2.3 Novel Loss and Evaluation Metric: *AbScore*

We introduce a simple criterion: Abruptness Score or *AbScore* to measure the turns and variability or non-linearity in each trajectory. An areal-scaled (bounding box area) of the metric is used for outlier detection (data cleaning) and assisted our synthetic dataset creation process. The *AbScore* statistics for SDD trajectories are in Tab. 2. Note that we do not use *AbScore* for training GPECNet. See A.6 for further mathematical formulation and motivations.

Statistic	Value
Max. AbScore	494866.37
Min. AbScore	0.0
Mean	3430.665
Std. Deviation	11987.34

Table 2: Abruptness Score statistics of SDD: A novel loss and evaluation metric for quantifying trajectories' non-linearity.

URM STATEMENT

The authors acknowledge that the first author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018. URL https://arxiv.org/abs/1803.08375.

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Learning motion patterns of persons for mobile service robots. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pp. 3601–3606, 2002.

Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional flow variational autoencoders for structured sequence prediction, 2020.

M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 994–999, 1997. doi: 10.1109/CVPR.1997.609450.

Nachiket Deo and Mohan M. Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans, 2021.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks, 2018.

Nikolaus Hansen. The cma evolution strategy: A tutorial, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents, 2017.

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26, 2007.

Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Conditional generative neural system for probabilistic trajectory prediction, 2019.

Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints and paths to long term human trajectory forecasting, 2020a.

Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. *CoRR*, abs/2004.02025, 2020b. URL https://arxiv.org/abs/2004.02025.

S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, 2009. doi: 10.1109/ICCV.2009.5459260.

A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory prediction in crowded scenes. 2016. URL http://cvgl.stanford.edu/projects/uav_data/.

Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, S. Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints, 2018.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. URL https://arxiv.org/abs/2006.09661.

Conghao Wong, Beihao Xia, Ziming Hong, Qinmu Peng, Wei Yuan, Qiong Cao, Yibo Yang, and Xinge You. View vertically: A hierarchical network for trajectory prediction via fourier spectrums, 2022.

Jiangbei Yue, Dinesh Manocha, and He Wang. Human trajectory prediction via neural social physics, 2023.

A Appendix

The code is available at Anonymous-Github-repo. All datasets, created and augmented, will be released upon publication.

A.1 Quantitative Comparison

We present a concise summary of previous seminal pedestrian prediction networks in Tab 3. All previous works use the following 3 datasets: ETH (Pellegrini et al., 2009), UCY (Lerner et al., 2007), and Stanford Drone dataset or SDD (Robicquet et al., 2016).

Method	ADE	FDE
DESIRE (Lee et al., 2017)	19.25	34.05
Social GAN (Gupta et al., 2018)	27.23	41.44
Sophie (Sadeghian et al., 2018)	16.27	29.38
CGNS (Li et al., 2019)	15.6	28.2
CF-VAE (Bhattacharyya et al., 2020)	12.60	22.30
P2TIRL (Deo & Trivedi, 2021)	12.58	22.07
PECNet (Mangalam et al., 2020b)	9.96	15.88
Y-Net (Mangalam et al., 2020a)	7.85	11.85
V^2-Net (Wong et al., 2022)	7.12	11.39
NSP-SFM (Yue et al., 2023)	**6.52**	10.61
G-PECNet	26.75	**9.04**

Table 3: ADE is the average displacement error and FDE is the final displacement error. All networks are evaluated on original SDD (Robicquet et al., 2016), with the total number of pedestrians to consider for predictions as 20 (K = 20), except ours. Our ADE (26.75) is not low as we use a decoupled PECNet; not using the social pooling layers (Alahi et al., 2016). We primarily focus on predicting the goal point of pedestrians. The intuition is that all intermediate steps could then be refined from coarser estimates after the endpoint is fixed, similar to the training procedure of denoising diffusion probabilistic models (Ho et al., 2020).

A.2 Stanford Drone Dataset: Data Analysis

Based on the unique quantitative (table: 4), we augmented the training dataset to keep the statistical properties of the training dataset intact. We perform a classification of the training dataset based on the number of unique points in each trajectory. See table 4.

Then we manually identify 7 qualitative classes for each trajectory as follows:

- Type 1: Stationary pedestrians

- Type 2: 3-8 unique points in trajectory bounded in a 5x5 box
- Type 2F: F means category Flying: A straight line trajectory will be shifted into the line starting from a new point if the perspective of the viewer (here: drone) changes. We call this scenario the flying category trajectory. The drone usually translates along an axis here.
- Type 3: 3-9 unique points loosely bounded in a 100x100 box
- Type 3F: (Flying) Same as 2F
- Type 4: Start and Goal points are within a 5x5 box
- Type 5: Flying randomly. This is different from 2F and 3F in the sense that the drone translated haphazardly here.
- Type 6: Backtracker: The pedestrian re-traces his steps after a while. Usually after 6-7 steps.
- Type 7: Perfectly Linear to Moderately linear trajectories that could be modelled by simple Newtonian mechanics.

These two classifications were done to emulate the statistical properties of the training dataset for augmentation purposes. Based on this data analysis and classifications (Tab. 4 and Sec. A.2 respectively), we augmented the training dataset using a Deep Policy Gradient Network A.3 + HMM interaction model and some Newtonian trajectories to keep the statistical properties of the training dataset intact. We manually discard any trajectories that did not fit the 7 types of trajectories identified in A.2.

A.3 Deep Policy Gradient Network

We use a simple fully connected ReLU-activated neural network (nodes: $8 \rightarrow 16 \rightarrow 8 \rightarrow 4$) with 8 inputs: current x-coordinate, current y-coorinate, x-goal, y-goal, fitness, patience, sociability, and distance to nearest person/agent and 4 output nodes defining the action space: the speed, direction, acceleration magnitude and acceleration direction to take another step. An overview of the whole workflow can be found in Fig 2.

HMMs were considered for the interaction modeling due to their high success in spatiotemporal tasks (Brand et al., 1997).

Unique Points	Trajectories	% dataset
1	145	5.13%
2	62	2.19 %
3	71	2.51 %
4	69	2.44%
5	57	2.01%
6	41	1.45%
7	51	1.80%
8	28	0.99%
9	26	0.92%
10	24	0.85%
11	22	0.78%
12	25	0.88%
13	17	0.60%
14	24	0.85%
15	22	0.78%
16	22	0.78%
17	39	1.38%
18	30	1.06%
19	76	2.69%
20	1978	69.92%

Table 4: SDD: Trajectories' unique points. 145 trajectories had 1 unique point, i.e, the goal and starting point as the same with all other points being sampled there itself: Stationary

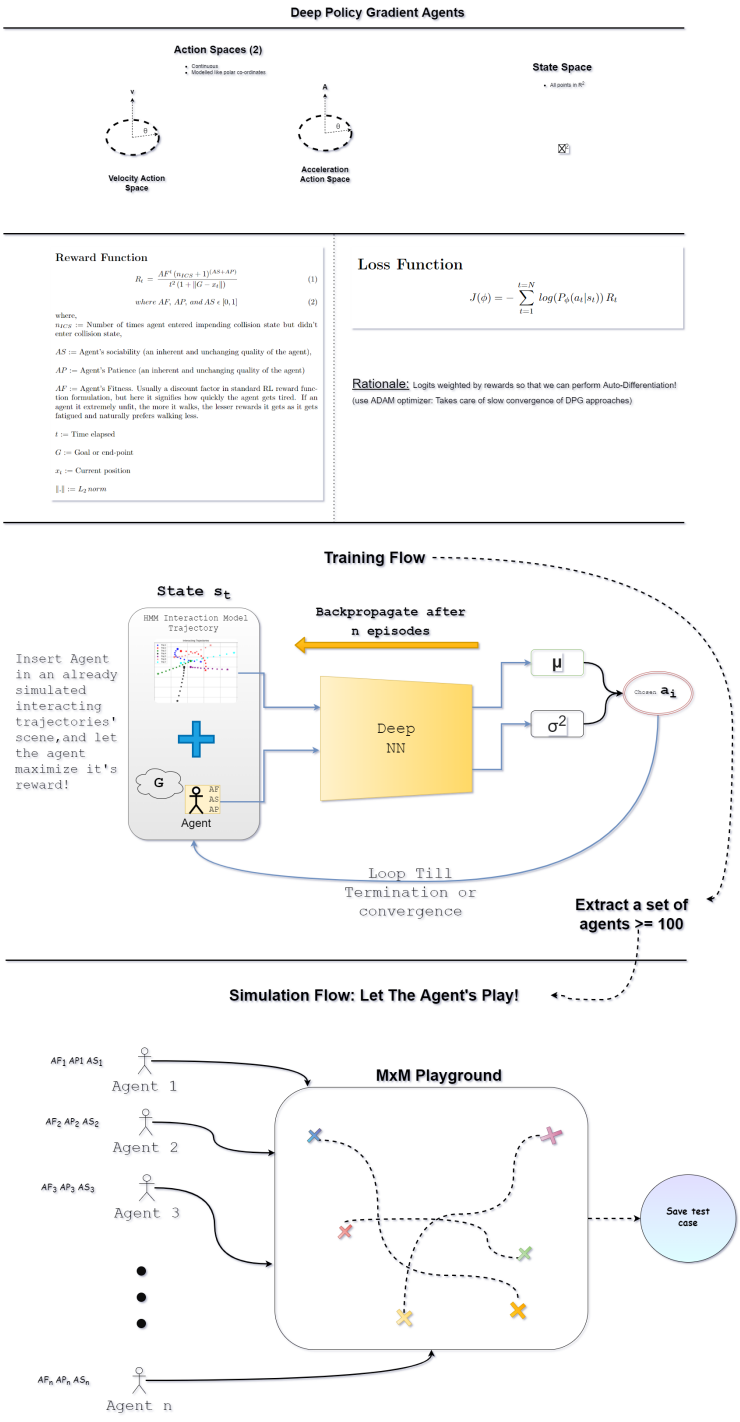

Figure 2: RL modeling

A.4 Architecture Brief

PECNet integrates a Conditional Variational Autoencoder (CVAE) while infusing probabilistic elements into the trajectory generation process. PECNet is equipped with specialized components, including 3 dedicated encoders for a) the past trajectories, b) the destination c) a latent space encoder, and finally a predictor for forecasting future trajectories. We depart from PECNet by using custom sinusoidally activated multi-layer perceptrons (MLPs). The incorporation of sine activations

is a departure from traditional ReLU and variants to capture high frequency spatial and temporal details of a signal, as also demonstrated in (Sitzmann et al., 2020). Notably, PECNet employs non-local social pooling mechanisms facilitated by three critical MLPs named: non-local-theta, non-local-phi, and non-local-g. They capture intricate long-range interactions among pedestrians. Since we do not use these networks or decouple the system, we see a high average displacement error.

During training, the model utilizes destination information to produce diverse and probabilistic future trajectories. During inference, it predicts future trajectories given historical context only. Please refer to our codebase for exact parameters and layer definitions.

A.5 Further Experiments

A.5.1 Ablation: Decoupled ADE & FDE or No Social Pooling in PECNet

We performed two ablation studies. First, by decoupling the ADE and the FDE metrics. See table 5.

Learning Rate	ADE	FDE	Best FDE epoch
0.001	>50	15.68	457
0.0005	>50	15.76	301
0.0003	>50	15.9	541
0.0002	>50	15.65	420
0.0001	>50	15.92	391

Table 5: Decoupled or without social pooling - PECNet (ADE and FDE) trained on original SDD with different learning rates. Here, ADE is independent of FDE. No SIREN improvement or data augmentations were applied either.

A.5.2 Effects of Data Augmentations

We sample the RL and Newtonian trajectories in a fashion to keep the statistics in Tab.4 similar. We report the ADE and FDE metrics of various levels of augmentations from 1% to 18% in table 6.

We also observe that PECNet heavily overfits on the SDD dataset and it is probable that Adam Kingma & Ba (2017) finds a deep crevice in the gradient surface. An evolutionary optimization strategy like Covariance Matrix Adaptation (CMA-ES) Hansen (2016) would highlight the shortcomings of the robustness of PECNet.

Augment %	Total Trajectories	ADE	FDE
1%	18328	64.34	19.18
3%	19048	53.22	15.63
5%	19766	45.17	15.72
6%	20126	51.73	**15.43**
8%	20844	46.37	15.75
10%	21564	51.51	15.54
13%	22642	**40.76**	15.90
15%	23360	51.40	18.36
18%	24438	56.56	15.90

Table 6: Effects of augmenting SDD with our synthetic trajectories in varying proportions and training PECNet with it. Standard learning rate: $3e-4$, 1000 epochs, and no social pooling were used across runs. Note that the training dataset has $\sim$18k trajectories due to simple augmentations(rotations & translations) also saved. This was introduced by Mangalam et al. (2020b). Since the agent simulations and Newtonian trajectory simulators are inherently random, we decided to sample $18k$ trajectories at once and subsequently used that purely synthetic dataset to sample & augment SDD. The sampling is deterministic as we always select the first $k\%$ from the fixed ordering.

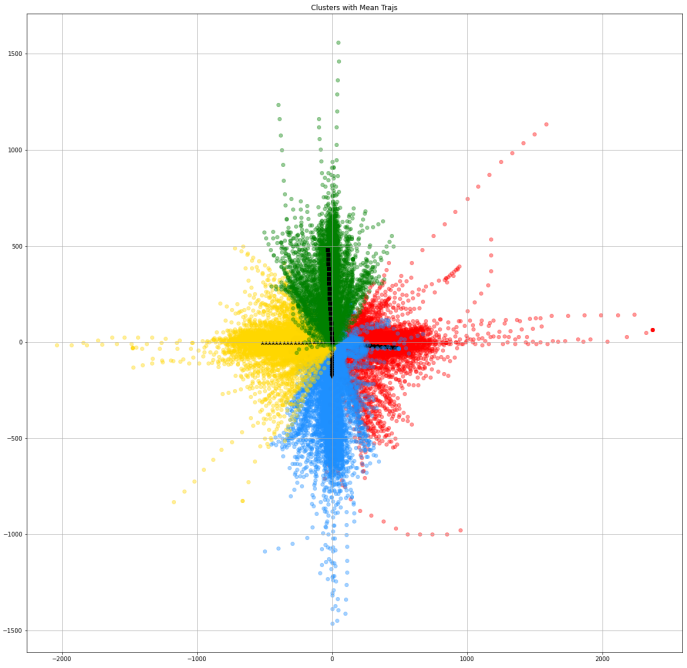

Figure 3: SDD training dataset. Frobenius norm-based clustering of trajectories, with black trajectories representing the cluster.

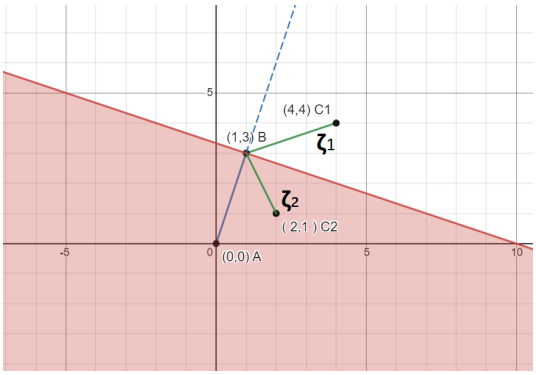

Figure 4: Defining Turns

A.5.3 Baseline Metrics' Reproduction (PECNet)

Learning Rate	Best ADE	Best FDE
0.001	11.01	15.62
0.0005	12.52	15.68
0.0003	**10.47**	**15.60**
0.0002	10.65	15.78
0.0001	10.65	15.78

Table 7: Sanity Run on different learning rates on original Stanford Drone Dataset, with social pooling. Bold represents benchmark reproduction.

A.6 Non-linearity analysis: Abruptness-Score

We clustered SDD's trajectories (3) based on the bounding boxes to get an estimate of the maximal displacement and turn in each trajectory. Based on this information, we introduce a novel and simple metric: *Abruptness Score* to measure the turns and variability or non-linearity in each trajectory. An areal-scaled and an unscaled version of the metric is used for analysis and outlier detection. The intuition and mathematical formulation are as follows:

In the example figure 4, the trajectories $\zeta_1 = \{A, B, C1\}$ and $\zeta_2 = \{A, B, C2\}$ are shown. The dotted blue line is normal to the red danger zone. Points that fall under this danger zone will form an obtuse turn trajectory like ζ_2. Naturally, we want the score to assign a larger value to ζ_2 than ζ_1 since the turn is huge and the trajectory (more) abruptly changes direction.

Mathematically,

$$AbScore = \left\lceil \frac{180.\theta}{10\pi} \right\rceil |\vec{a} \times \vec{b}| \tag{1}$$

where

$$\vec{a} = \vec{AB}, \vec{b} = \vec{BC} \tag{2}$$

$$\theta = |\arcsin \frac{\vec{a} \times \vec{b}}{|\vec{a}||\vec{b}|}| \tag{3}$$

If θ is obtuse, we add $pi/2$ to θ before sending it to equation 1.

For scaling we simply divide the abruptness score by the area of the tightest-bounding-box of the trajectory or divide by $(max(\zeta_x) - min(\zeta_x)) * (max(\zeta_y) - min(\zeta_y)$. For perfectly linear trajectories, we use the length of the trajectory for scaling.

We need areal-scaling to get an unbiased estimate of trajectories' non-linearity that spans widely different sizes or regions. Based on this metric we analyze SDD and report that the trajectories are not non-linear on average however the distribution contains outliers with huge non-linearity scores. This analysis provided us with an estimate of the dataset's non-linearity for synthetic dataset generation purposes. See table 2 and figure 5.

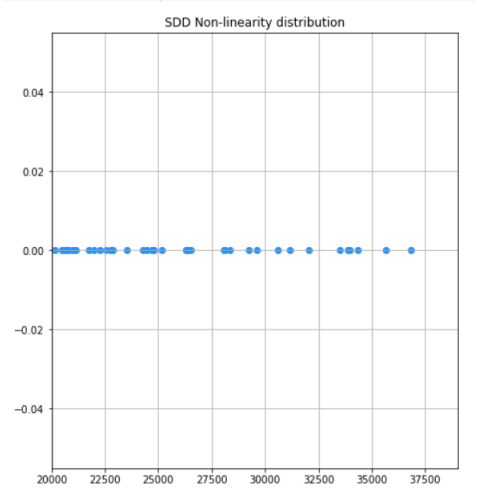

Figure 5: SDD's Non-Linearity Distribution Tail. (Thresholded to remove outliers)

A.7 Discussion

We demonstrated state-of-the-art final displacement errors on the Stanford Drone Dataset with our method GPECNet. The core improvements originate from our rich synthetic data augmentations coupled with SIRENs (Sitzmann et al., 2020) that can capture better high-frequency spatial and temporal dependencies. Even though our method achieves the best FDE results, the critical nature of systems that could employ our algorithm necessitates introducing a confidence metric for larger controllability and explainability. Another avenue to extend our work is generating multi-modal predictions simultaneously to move towards a real deployable system.