Generating Synthetic Ground Truth Distributions for Multi-step Trajectory Prediction using Probabilistic Composite Bézier Curves

Ronny Hug[†] and Stefan Becker[†] and Wolfgang Hübner[†] and Michael Arens[†]

Abstract— **An appropriate data basis grants one of the most important aspects for training and evaluating probabilistic trajectory prediction models based on neural networks. In this regard, a common shortcoming of current benchmark datasets is their limitation to sets of sample trajectories and a lack of actual ground truth distributions, which prevents the use of more expressive error metrics, such as the Wasserstein distance for model evaluation. Towards this end, this paper proposes a novel approach to synthetic dataset generation based on composite probabilistic Bézier curves, which is capable of generating ground truth data in terms of probability distributions over full trajectories. This allows the calculation of arbitrary posterior distributions. The paper showcases an exemplary trajectory prediction model evaluation using generated ground truth distribution data.**

I. Introduction

An integral component for training and evaluating neural network models is the use of an appropriate data basis. Taking multi-step human trajectory[1] prediction as an example, where given a sequence of spatial positions $\{\mathbf{x}_1, ...\mathbf{x}_N\}$, the observation, the next M positions $\{\mathbf{x}_{N+1}, ..., \mathbf{x}_{N+M}\}$ are to be predicted, obtaining such a data basis is especially difficult. This is because an adquate representation for the future progression of an observed trajectory is given by a conditional multi-modal probability distribution $p(\mathbf{x}_{N+1}, ..., \mathbf{x}_{N+M}|\mathbf{x}_1, ...\mathbf{x}_N)$, hence favors the use of probabilistic prediction models, which learn to approximate this distribution. In this case an ideal data basis would provide these conditional ground truth distributions. However, despite efforts in learning-based assignment of probability to samples (e.g. [1]), there is no reliable way of deriving required conditional distributions from commonly used trajectory datasets, e.g. provided by benchmarks such as *Thör* [2] or *TrajNet++* [3], only leaving the option of resorting to synthetically generated data in case distribution-based ground truth data is desired[2].

A common approach for synthetically generating trajectory data consists of first generating paths through a virtual scene, either using waypoints or simple motion patterns. Virtual agents then follow these paths, optionally complying with physical constrains (e.g. [4]) or interacting with other agents [5], whereby their trajectory is recorded. Finally, sensor or annotation noise is simulated by applying a noise model to each individual trajectory point. However, this approach of injecting uncertainty often limits the dataset to representing probability distributions on a per point basis instead of distributions over full trajectories, due to a lack of knowledge about intra-trajectory and inter-path correlations.

Towards this end, this paper proposes a novel approach to synthetic trajectory data generation for probabilistic trajectory prediction models, which is capable of generating ground truth data in terms of probability distributions over full trajectories. The approach utilizes probabilistic (composite) Bézier curves ($\mathcal{N}$-Curves, [6]) for modeling individual paths and arranges multiple curves in a mixture distribution for building multi-path datasets. By exploiting the $\mathcal{N}$-Curve's equivalence with Gaussian processes, datasets defined this way enable the calculation of conditional distributions over trajectories given arbitrary observations and thus the use of more expressive performance metrics, such as the Wasserstein distance alongside the commonly used negative log-likelihood, for model evaluation. In order to showcase the application of the proposed approach for benchmarking probabilistic trajectory prediction models, an exemplary evaluation following the common benchmarking approach is provided.

II. Dataset Generation Approach

In this paper, the primary goal is to derive an approach for trajectory dataset generation, which yields a probability distribution over full trajectories covering multiple paths through a virtual (structured) environment and further allows for the calculation of conditional distributions during training or test given different observations. The proposed dataset generation approach consists of multiple stages, which are explained in more detail in the following:

1) Definition of paths through a virtual (structured) environment in terms of probabilistic Bézier curves.
2) Definition of velocity profiles by curve discretization.
3) Derivation of the dataset prior distribution.
4) As required: Calculation of posterior distributions given specific observed trajectories.

A variant of this dataset generator is implemented and utilized in the scope of the latest version[3] of the STSC benchmark and is available at `https://github.com/stsc-benchmark/stsc-lib`.

a) Defining paths through a virtual (structured) environment: In order to provide an intuitive way of defining paths within a dataset, $\mathcal{N}$-Curves [6], a probabilistic extension of Bézier curves [8] which add uncertainty to points

[†]Fraunhofer IOSB, Fraunhofer Center for Machine Learning, Ettlingen, Germany. `firstname.lastname@iosb.fraunhofer.de`

[1]Here, a trajectory is defined as a sequence of locations along a path with some velocity profile attached to it.

[2]E.g. in case an evaluation aims at a more nuanced model evaluation not achievable by the common approach of using the negative log-likelihood or top-k metrics together with plain trajectory data.

[3]Building on the core concepts presented in the initial STSC paper [7]

along the curve, are chosen as the basic building block for representing individual paths. By stringing together multiple $\mathcal{N}$-Curves into a composite curve, complex paths can easily be pieced together.

Formally, $\mathcal{N}$-Curves of degree L, defined by $(L+1)$ independent d-dimensional Gaussian control points $\mathcal{P} = \{P_0, ..., P_L\}$ with $P_l \sim \mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$, are employed as a foundation for a pattern-based trajectory dataset description. Through the curve construction function

$$X_t = B_{\mathcal{N}}(t, \mathcal{P}) = (\mu_{\mathcal{P}}(t), \Sigma_{\mathcal{P}}(t)), \text{ with} \tag{1}$$

$$\mu_{\mathcal{P}}(t) = \sum_{l=0}^{L} b_{l,L}(t)\boldsymbol{\mu}_l \text{ and } \Sigma_{\mathcal{P}}(t) = \sum_{l=0}^{L} (b_{l,L}(t))^2 \boldsymbol{\Sigma}_l, \tag{2}$$

where $b_{l,L}(t) = \binom{L}{l}(1-t)^{L-l}t^l$ are the Bernstein polynomials [9], the stochasticity is passed from the control points to the curve points $X_t \sim \mathcal{N}(\mu_{\mathcal{P}}(t), \Sigma_{\mathcal{P}}(t))$, yielding a sequence of Gaussian distributions $\{X_t\}_{t\in[0,1]}$ along the underlying Bézier curve. Here, the curve parameter $t \in T = [0,1]$ indicates the position on the curve. Stringing together N_{seg} $\mathcal{N}$-Curves with respective Gaussian control points $\mathcal{P}_j = \{P_0^j, ..., P_{L_j}^j\}$, where $j \in \{1, ..., N_{seg}\}$, yields a composite curve. In this case, t still remains in $[0,1]$ and traverses all segments of the composite curve. For curve point calculation, only the control points of the segment the curve point resides in are used. The segment is determined from t by the mapping $j = m_c(t) = \max\{1, \lceil \frac{t}{\tau} \rceil\}$ with $\tau = \frac{1}{N}$. For segment-specific calculations t is mapped onto the segment local position $\frac{t - m_c(t)\cdot\tau}{(m_c(t)+1)\cdot\tau - m_c(t)\cdot\tau} = \text{loc}(t) \in [0,1]$. Fig. 1 depicts a composite $\mathcal{N}$-Curve.

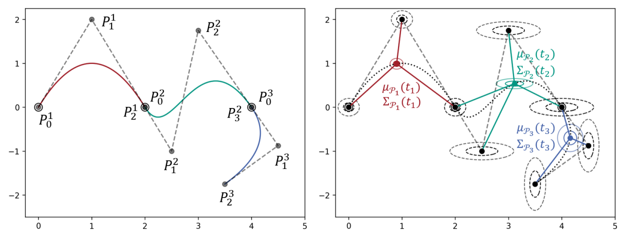

P_1^1
P_2^2
P_0^2
P_2^1
P_0^3
P_3^2
P_0^1
P_1^2
P_1^3
P_2^3
$\mu_{\mathcal{P}_2}(t_2)$
$\Sigma_{\mathcal{P}_2}(t_2)$
$\mu_{\mathcal{P}_1}(t_1)$
$\Sigma_{\mathcal{P}_1}(t_1)$
$\mu_{\mathcal{P}_3}(t_3)$
$\Sigma_{\mathcal{P}_3}(t_3)$

Fig. 1. Exemplary composite $\mathcal{N}$-Curve consisting of $N_{seg} = 3$ segments with control point sets $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$, where $\mathcal{P}_1 = \{P_0^1, P_1^1, P_2^1\}$, $\mathcal{P}_2 = \{P_0^2, P_1^2, P_2^2, P_3^2\}$ and $\mathcal{P}_3 = \{P_0^3, P_1^3, P_2^3\}$, with $L_1 = L_3 = 2$ and $L_2 = 3$. Each control point $P_l^j \sim \mathcal{N}(\cdot|\cdot)$ follows a Gaussian distribution with respective mean μ_l^j and covariance matrix Σ_l^j. Left: The resulting mean curve with control point locations. Covariance ellipses are omitted for clarity. Right: Gaussian curve points along the composite $\mathcal{N}$-Curve at curve positions $t_1 = 0.15$, $t_2 = 0.55$ and $t_3 = 0.8$. The influence of control points on each curve point is indicated by solid lines.

Finally, for modeling multiple paths in a single dataset, K composite $\mathcal{N}$-Curves with respective control point sets $\mathcal{P}_k$ can be combined into a mixture with prior weight distribution $\pi = \{\pi_1, ...\pi_K\}$ over the curves. In this way, complex datasets can be represented, where each mixture component models a specific path through the scene.

b) Defining velocity profiles by $\mathcal{N}$-Curve mixture discretization: In order to extract trajectory data from a set of paths defined in terms of a mixture of K (continuous) composite $\mathcal{N}$-Curves, discrete subsets T_N^k of T can be employed for extracting length N trajectories with (Gaussian) trajectory points $X_i = B_{\mathcal{N}}(t_i, \mathcal{P}_k)\ \forall t_i \in T_N^k$ from single mixture components. Altering the relative placement of the t_i yields different trajectory velocity profiles, e.g. reflecting constant or accelerating movement speed. For example, constant speed can be achieved by setting T_N^k, such that constant distance $\|\mu_{\mathcal{P}_k}(t_{i+2}) - \mu_{\mathcal{P}_k}(t_{i+1})\| = \|\mu_{\mathcal{P}_k}(t_{i+1}) - \mu_{\mathcal{P}_k}(t_i)\|$ between subsequent curve points is achieved. The curve parameter subsets can be defined on a per path basis and be of varying length.

c) Deriving the dataset's prior distribution: Given a dataset defined in terms of a mixture of composite $\mathcal{N}$-Curves and a discrete curve parameter subset $T_{N_k}^k$[4] for each mixture component, the dataset's prior distribution over the set of possible trajectories has to be calculated. This can be achieved by exploiting the equivalence of $\mathcal{N}$-Curves and a specific class of Gaussian processes (GP) [10] in order to derive a vector-valued mean and matrix-valued covariance function from each mixture component. This ultimately converts the mixture of composite $\mathcal{N}$-Curves into a mixture of GPs. Using this mixture of GPs, a prior distribution modeling trajectories with lengths N_k can be derived in terms of a Gaussian mixture distribution with weights $\{\pi_k\}_{k\in\{1,..,K\}}$, mean vectors $\{\mu_k\}_{k\in\{1,..,K\}}$ and covariance matrices $\{\Sigma_k\}_{k\in\{1,..,K\}}$. The derivation of the μ_k and Σ_k is detailed in the following. For conciseness, the derivation focuses on a single mixture component and omits the component index k for reducing visual clutter.

Starting with the vector-valued mean function $\mathbf{m}_{\mathcal{P}}$, each mixture component's prior mean vector μ is given by the concatenation of all mean vectors $\boldsymbol{\mu}_{\mathcal{P}}(t_i)$ (see Eq. 2) along the underlying Bézier curve, i.e.

$$\mu = (\mathbf{m}_{\mathcal{P}}(T_N))^\top = \left((\boldsymbol{\mu}_{\mathcal{P}}(t_1))^\top \quad \cdots \quad (\boldsymbol{\mu}_{\mathcal{P}}(t_N))^\top\right). \tag{3}$$

In the case of 2d trajectories, the resulting vector consists of N_k 2d vectors, yielding a $(2 \cdot N_k \times 1)$ vector.

The covariance matrix Σ is given by the block-partitioned Gram matrix calculated from the respective matrix-valued covariance function $\mathbf{K}_{\mathcal{P}}(t_{i_1}, t_{i_2})$ with $t_{i_1}, t_{i_2} \in T_N$, which connects[5] two Gaussian curve points X and Y on the same composite $\mathcal{N}$-Curve, where $X = B_{\mathcal{N}}(t_{i_1}, \mathcal{P}_{m_c(t_{i_1})})$ and $Y = B_{\mathcal{N}}(t_{i_2}, \mathcal{P}_{m_c(t_{i_2})})$. With X and Y potentially residing on different curve segments, and thus potentially being calculated from different control points, multiple cases have to be considered when deriving $\mathbf{K}_{\mathcal{P}}(t_{i_1}, t_{i_2})$[6]:

1) $m_c(t_{i_1}) = m_c(t_{i_2})$: X and Y reside on the same $\mathcal{N}$-Curve segment. This is the case covered by [10].
2) $m_c(t_{i_1}) + 1 = m_c(t_{i_2})$: X and Y reside on subsequent $\mathcal{N}$-Curve segments.
3) $m_c(t_{i_1}) + 1 < m_c(t_{i_2})$ There is at least one curve segment in between the segments both.

[4] $T_{N_k}^k$ denotes the finite curve parameter subset of the k'th mixture component, covering trajectories of length N_k.

[5] In terms of their correlation.

[6] Here, $t_{i_1} \leq t_{i_2}$ is assumed.

In case both X and Y reside on the same segment (see e.g. t_1 and t_2 in Fig. 3), the covariance function

$$
\begin{aligned}
\mathbf{K}_{\mathcal{P}_{m_c(t_{i_1})}}(t_{i_1}, t_{i_2}) &= \mathbb{E}[(\boldsymbol{X} - \mu_X)(\boldsymbol{Y} - \mu_Y)^\top] \\
&= \mathbb{E}\left[\boldsymbol{X}\mathbf{Y}^\top\right] - \mu_X \mu_Y^\top \\
&= \sum_{l=0}^{L} b_{l,L}(\mathrm{loc}(t_i)) b_{l,L}(\mathrm{loc}(t_j)) \left(\Sigma_l + \mu_l \mu_l^\top\right) \\
&+ \sum_{l=0}^{L} \left(\sum_{l'=0, l' \neq l}^{L} b_{l,L}(\mathrm{loc}(t_i)) b_{l',L}(\mathrm{loc}(t_j)) \mu_l \mu_{l'}^\top \right) \\
&- \mu_X \mu_Y^\top,
\end{aligned}
\tag{4}
$$

derived in [10] can be used. Here, μ_X and μ_Y denote the mean vectors of X and Y. The calculation is based on the $m_c(t_{i_1})$'th segment's Gaussian control points $\mathcal{P}_{m_c(t_{i_1})}$.

In case X and Y reside on subsequent segments (see e.g. t_2 and t_3 in Fig. 3), Eq. 4 has to be modified to consider control point dependencies emerging from the composite curve's continuity in its connecting points. While C^0 continuity is inherent, C^1 and C^2 continuity emerge from equal tangency (C^1 and C^2) together with equal curvature (C^2) in the connecting point for two connected segments. Geometrically, considering 2 connected segments j and $j+1$, C^1 continuity can be enforced by arranging $P^j_{L_j-1}$, $P^j_{L_j} = P^{j+1}_0$ (connecting point) and P^{j+1}_1 on a straight line. C^2 continuity is given for equidistant points, i.e. $\|P^j_{L_j} - P^j_{L_j-1}\| = \|P^{j+1}_1 - P^j_{L_j}\|$. An example for a C^0 and a C^1 continuous composite curves is depicted in Fig. 2. Following this, given C^1 or C^2

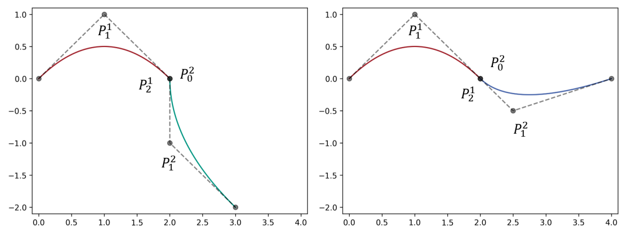


Fig. 2. Example for a C^0 (left) and a C^1 (right) continuous curve.

continuity, correlations between curve points of subsequent segments emerge through the geometric dependency of the control point sets, given by $P^{j+1}_1 = P^j_{L_j} + s \cdot (P^j_{L_j} - P^j_{L_j-1})$ with $s = \frac{\|P^{j+1}_1 - P^{j+1}_0\|}{\|P^j_{L_j} - P^j_{L_j-1}\|}$. Hence the calculation of $\mathbb{E}[\boldsymbol{X}\boldsymbol{Y}^\top]$, which combines all Gaussian control points of both segments $m_c(t_{i_1})$ and $m_c(t_{i_2})$, is adjusted accordingly:

$$
\mathbb{E}[\boldsymbol{X}\boldsymbol{Y}^\top] = \sum_{l_1=0}^{L_1} \sum_{l_2=0}^{L_2} b_* \cdot
\begin{cases}
(\Sigma_{l_1} + \mu_{l_1}\mu_{l_2}^\top), & c_1 \\
(\mu_{l_1}\mu_{L_1}^\top + s \cdot (\mu_{l_1}\mu_{L_1}^\top - \mu_{l_1}\mu_{l_1}^\top)), & c_2 \\
(\mu_{L_1}\mu_{L_1}^\top + s \cdot (\mu_{L_1}\mu_{L_1}^\top - \mu_{L_1}\mu_{L_1-1}^\top)), & c_3 \\
\mu_{l_1}\mu_{l_2}^\top, & c_4
\end{cases}
.
\tag{5}
$$

Here, $b_* = b_{l_1,L_1}(\mathrm{loc}(t_{i_1})) b_{l_2,L_2}(\mathrm{loc}(t_{i_2}))$ is the weighting factor and the cases c_1 through c_4 are given by

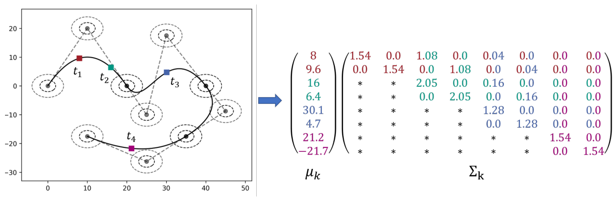
$$
\begin{aligned}
c_1 &\equiv l_1 = L_1 \wedge l_2 = 0 \text{ (connecting point)} \\
c_2 &\equiv l_1 = L_1 - 1 \wedge l_2 = 1 \\
c_3 &\equiv l_1 = L_1 \wedge l_2 = 1 \\
c_4 &\equiv \text{else (independent points).}
\end{aligned}
$$

The outer ($\sum_{l_1=0}^{L_1}$) and inner ($\sum_{l_2=0}^{L_2}$) sums iterate over the control points of segment $m_c(t_{i_1})$ and $m_c(t_{i_2})$, respectively.

Lastly, in case X and Y reside on disconnected segments,

$$
\mathbb{E}[\boldsymbol{X}\boldsymbol{Y}^\top] = \sum_{l_1=0}^{L_1} \sum_{l_2=0}^{L_2} b_{l_1,L_1}(\mathrm{loc}(t_{i_1})) b_{l_2,L_2}(\mathrm{loc}(t_{i_2})) \mu_{l_1}\mu_{l_2}^\top
\tag{6}
$$

collapses into the independent points case (c_4). A schematic illustrating the three cases with the resulting prior mean vector and covariance matrix is depicted in Fig. 3.



$$
\underset{\mu_k}{\begin{pmatrix} 8 \\ 9.6 \\ 16 \\ 6.4 \\ 30.1 \\ 4.7 \\ 21.2 \\ -21.7 \end{pmatrix}}
\underset{\Sigma_k}{\begin{pmatrix}
1.54 & 0.0 & 1.08 & 0.0 & 0.04 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.54 & 0.0 & 1.08 & 0.0 & 0.04 & 0.0 & 0.0 \\
* & * & 2.05 & 0.0 & 0.16 & 0.0 & 0.0 & 0.0 \\
* & * & 0.0 & 2.05 & 0.0 & 0.16 & 0.0 & 0.0 \\
* & * & * & * & 1.28 & 0.0 & 0.0 & 0.0 \\
* & * & * & * & 0.0 & 1.28 & 0.0 & 0.0 \\
* & * & * & * & * & * & 1.54 & 0.0 \\
* & * & * & * & * & * & 0.0 & 1.54
\end{pmatrix}}
$$

Fig. 3. Example of a mean vector and covariance matrix derived from an $\mathcal{N}$-Curve covering $N = 4$ Gaussian curve points.

d) Calculating posterior distributions: While the prior distribution suffices for generating trajectory data through sampling, calculating posterior distributions given different observations is the final missing piece for obtaining conditional ground truth distributions. The Gaussian mixture-based prior distribution models full trajectories of length N[7], which yields a joint probability distribution $p(X_1, ..., X_N)$ over the trajectory points. By partitioning the joint prior distribution into a partition containing the N_{obs} observed time steps to condition on and the remaining N_{pred} time steps, i.e. $p(X_1, ..., X_N) = p(\{X_1, ..., X_{N_{\text{obs}}}\} \cup \{X_{N_{\text{in}}+1}, ..., X_N\}) = p(X_A \cup X_B)$, the conditional distribution $p(X_B|X_A)$ can be calculated directly (see e.g. [11], [12]). The probability distribution for individual trajectory points can be extracted through marginalization An example for different posterior distributions using different subsets of the same sample trajectory is depicted in Fig. 4.

III. Exemplary Evaluation based on Gaussian Mixture Datasets and the Wasserstein Distance

This section provides a brief showcase on how the Gaussian mixture-based dataset can be used within the standard evaluation approach for employing the Wasserstein distance alongside the commonly used negative log-likelihood (*NLL*, [13]). The Wasserstein distance [14] $W_p(P, Q)$ quantifies the dissimilarity between two probability distributions P and Q by measuring the work required to transport the probability

[7]For simplicity equal trajectory length is assumed for each component.

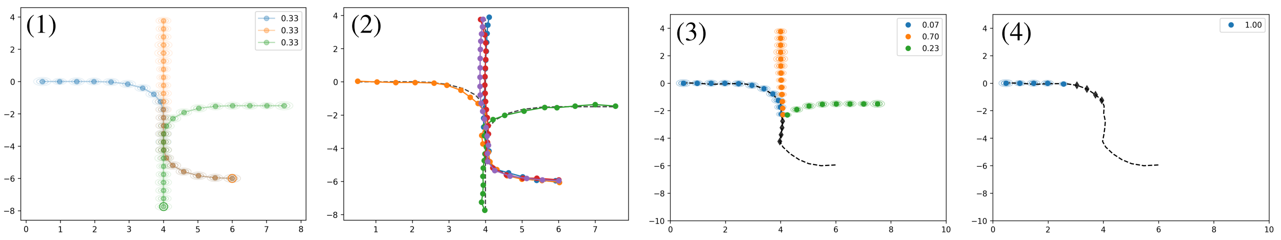


Fig. 4. 1 & 2: A dataset's prior distribution in terms of a Gaussian mixture covering full trajectories and samples drawn from the prior. 3 & 4: Posterior distributions given by conditioning on different subsets of the same trajectory. Note that mixture components may cover different sequence lengths.

mass from P to Q. For dimensions $d > 1$, an approximation, such as the sliced Wasserstein distance [15], has to be used.

For this evaluation, a dataset with 3 paths consisting of straight and curved segments is defined. All mixture components model trajectories of the same constant movement speed. The dataset's prior distribution and trajectory samples drawn from the dataset are depicted in Fig. 4.

a) Training: As a reference probabilistic trajectory prediction model, an extension to RED [16], which enables multi-modal predictions, is used. The model is trained on a set of 200 trajectories sampled from the dataset, each capped at a length of $N = 19$[8], using an NLL-based loss function. Here, the observation length is set to $N_{obs} = 4$ and the prediction length is set to $N_{pred} = 6$.

b) Evaluation: For model evaluation, 20 additional trajectories are sampled. In order to evaluate the trained model's performance, the following steps are performed for each test trajectory $\mathcal{X} = \{\mathbf{x}_1, ...\mathbf{x}_{19}\}$:

1) Extract a sub-trajectory $\mathcal{X}_{N_s} = \{\mathbf{x}_i, ..., \mathbf{x}_{i+N_s-1}\}$ of length $N_s = N_{obs} + N_{pred}$, e.g. $\{\mathbf{x}_2, ...\mathbf{x}_{12}\}$. The first N_{obs} points of $\mathcal{X}_{N_s}$ will be denoted as $\mathcal{X}^{obs}_{N_s}$.
2) Using the dataset's prior mixture distribution $p(X_1, ..., X_N)$, for each mixture component find the subset $p_k(X_j, ..., X_{j+N_{obs}})$ with the mean sequence $\{\mu^k_j, ..., \mu^k_{j+N_{obs}}\}$ closest to $\mathcal{X}^{obs}_{N_s}$.
3) Obtain the conditional ground truth distribution[9] $P = \sum_k \pi_{k,\text{pred}|\text{obs}} \cdot p_k(X_{j+N_{obs}+1}, ..., X_{j+N_{pred}} | \mathcal{X}^{obs}_{N_s})$.
4) Pass $\mathcal{X}^{obs}_{N_s}$ through RED in order to obtain a prediction Q for the conditional distribution P.
5) Approximate the distance between P and Q on a per point basis using the sliced Wasserstein distance. Take the average as the final score.
6) When using the NLL as an additional performance metric, use Q and $\mathcal{X}^{pred}_{N_s}$ for calculation.

c) Merits of using the Wasserstein distance: Using the above approach and considering all test trajectories, the RED predictor reached an overall score of -0.74 when considering the NLL and a score of 0.47 when considering the Wasserstein distance. Looking at these scores, one of the main advantages of the Wasserstein distance becomes apparent: *interpretability*. While the NLL score only allows relative comparisons due to being unbound, a perfect prediction will have a Wasserstein distance of 0, meaning the score's face value directly allows drawing conclusions about the prediction quality. Apart from that, it can be observed that both metrics yield proportional results overall with deviations when ordering model predictions by score. These deviations occur due to the Wasserstein distance being more accurate in scoring the similarity between distributions, especially when considering the distribution's variance. To give an example, this can be observed looking at the input sample producing the best score *in each respective metric* depicted in Fig. 5, where the Wasserstein distance does a much better job at incorporating variance estimation errors in the output of the prediction model, leading to a prediction closely matching the actual variance receiving a better score than the one with over-estimated variance as is the case for the NLL. As a final remark, it should be noted that calculating the Wasserstein distance considerably increases the computation time of the evaluation. For the test dataset used in this exemplary evaluation, calculating the Wasserstein distances took about 125 times longer than calculating the NLL scores.

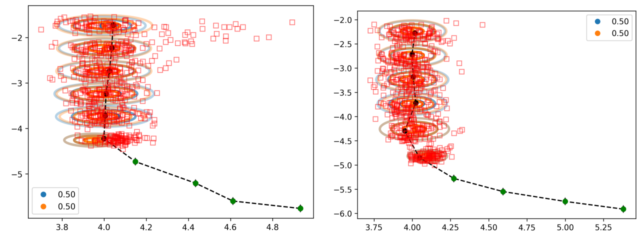


Fig. 5. Sample-based predictions generated by the RED predictor for inputs giving the *best* score in terms of the NLL (left) and the Wasserstein distance (right), respectively. The conditional ground truth is indicated by per point Gaussian mean locations and covariance ellipses obtained through marginalization. The labels indicate the mixture weights.

IV. Summary

In this paper, a novel approach for generating synthetic trajectory datasets in terms of probability distributions over full trajectories has been proposed. The approach allows the calculation of arbitrary conditional probability distributions required for a more nuanced evaluation of probabilistic trajectory prediction models by allowing the use of the more expressive Wasserstein distance instead of the negative log-likelihood. An exemplary evaluation based on this data generation approach has been conducted, which was concluded with a brief discussion on the merits of the Wasserstein distance compared to the negative log-likelihood.

[8]This is the shortest length among the mixture components
[9]For the calculation of the conditional weights see e.g. [11], [12]

References

[1] T. R. de Almeida and O. M. Mozos, “Likely, light, and accurate context-free clusters-based trajectory prediction,” in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 1269–1276.

[2] A. Rudenko, T. P. Kucner, C. S. Swaminathan, R. T. Chadalavada, K. O. Arras, and A. J. Lilienthal, “Thör: Human-robot navigation data collection and accurate motion trajectories dataset,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 676–682, 2020.

[3] P. Kothari, S. Kreiss, and A. Alahi, “Human trajectory forecasting in crowds: A deep learning perspective,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2021.

[4] S. Becker, R. Hug, W. Huebner, M. Arens, and B. T. Morris, “Generating versatile training samples for uav trajectory prediction,” in *Robotics, Computer Vision and Intelligent Systems*, P. Galambos, E. Kayacan, and K. Madani, Eds. Cham: Springer International Publishing, 2022, pp. 208–229.

[5] D. Helbing and P. Molnar, “Social force model for pedestrian dynamics,” *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[6] R. Hug, W. Hübner, and M. Arens, “Introducing probabilistic bézier curves for n-step sequence prediction,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 162–10 169.

[7] R. Hug, S. Becker, W. Hübner, and M. Arens, “A complementary trajectory prediction benchmark,” in *ECCV Workshop on Benchmarking Trajectory Forecasting Models (BTFM)*, vol. 3, 2020.

[8] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*. Springer Science & Business Media, 2002.

[9] G. G. Lorentz, *Bernstein polynomials*. American Mathematical Soc., 2013.

[10] R. Hug, S. Becker, W. Hübner, M. Arens, and J. Beyerer, “B\’ezier curve gaussian processes,” *arXiv preprint arXiv:2205.01754*, 2022.

[11] K. B. Petersen and M. S. Pedersen, “The matrix cookbook,” *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008.

[12] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. [Online]. Available: probml.ai

[13] B. Ivanovic and M. Pavone, “The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2375–2384.

[14] S. Kolouri, S. R. Park, M. Thorpe, D. Slepcev, and G. K. Rohde, “Optimal mass transport: Signal processing and machine-learning applications,” *IEEE signal processing magazine*, vol. 34, no. 4, pp. 43–59, 2017.

[15] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister, “Sliced and radon wasserstein barycenters of measures,” *Journal of Mathematical Imaging and Vision*, vol. 51, pp. 22–45, 2015.

[16] S. Becker, R. Hug, W. Hubner, and M. Arens, “Red: A simple but effective baseline predictor for the trajnet benchmark,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.