Virtual Target Trajectory Prediction for Stochastic Targets

Marc Schneider^{*}, Renato Loureiro[†], Torbjørn Cunis[‡], Walter Fichter[§] University of Stuttgart, 70569, Germany

Trajectory prediction of other vehicles is crucial for autonomous vehicles, with applications from missile guidance to UAV collision avoidance. Typically, target trajectories are assumed deterministic, but real-world aerial vehicles exhibit stochastic behavior, such as evasive maneuvers or gliders circling in thermals. This paper uses Conditional Normalizing Flows, an unsupervised Machine Learning technique, to learn and predict the stochastic behavior of targets of guided missiles using trajectory data. The trained model predicts the distribution of future target positions based on initial conditions and parameters of the dynamics. Samples from this distribution are clustered using a time series k-means algorithm to generate representative trajectories, termed virtual targets. The method is fast and target-agnostic, requiring only training data in the form of target trajectories. Thus, it serves as a drop-in replacement for deterministic trajectory predictions in guidance laws and path planning. Simulated scenarios demonstrate the approach's effectiveness for aerial vehicles with random maneuvers, bridging the gap between deterministic predictions and stochastic reality, advancing guidance and control algorithms for autonomous vehicles.

I. Introduction

PREDICTION models play a critical role in the guidance and control of guided missiles. In order to effectively guide a missile to its target, it is necessary to predict the future trajectory of both the missile and the target. These predictions enable the computation of control commands within a guidance law. While the trajectory of the missile can often be predicted with high accuracy due to the deterministic nature of its dynamics, predicting the trajectory of the target is much more challenging because of its inherently stochastic behavior.

Many traditional prediction models for target motion assume deterministic behavior, either implicitly or explicitly. For example, the Proportional Navigation (PN) guidance law [1] assumes that the target maintains a constant velocity, while the Zero-Effort-Miss (ZEM) guidance law [2, 3] allows for arbitrary target dynamics within a deterministic framework.

However, relying on a deterministic model for the target can degrade the performance of the guidance law. This is

^{*}Research Associate, Institute of Flight Mechanics and Controls.

[†]Research Associate, Institute of Flight Mechanics and Controls.

[‡]Lecturer, Institute of Flight Mechanics and Controls, AIAA Member.

[§]Professor, Institute of Flight Mechanics and Controls, AIAA Associate Fellow.

because the control commands are based on a fixed prediction of the trajectory of the target, which may not align with the actual, more varied maneuvers that the target is executing. In reality, the motion of the target is not confined to a single deterministic model; it is capable of executing a wide range of possible maneuvers. Recognizing this limitation, [4] introduced an approach using an Extended Kalman Filter (EKF) to estimate not only the position and velocity of the target, but also the parameters governing its dynamics.

This concept was further advanced in [5], where a multi-hypothesis guidance approach was proposed. This method employed an Interacting Multiple Model (IMM) filter to estimate both the state of the target and the probability of various dynamics hypotheses being true at any given time. The IMM framework runs multiple EKFs in parallel, each corresponding to a different target dynamics hypothesis. In turn, multiple guidance laws are executed in parallel, with the final control commands derived by combining the outputs of the guidance laws according to the probability estimates of the various hypotheses.

While this multi-hypothesis guidance technique is powerful, it has two significant drawbacks. First, the number of hypotheses that can be considered is limited by the available computational resources, as each additional hypothesis requires one additional parallel EKF and one additional guidance law. Second, despite incorporating multiple hypotheses, the approach still assumes that the target follows one of several deterministic trajectories, which may not fully capture the stochastic nature of the motion of the target.

To address these limitations, this paper introduces a probabilistic approach. Instead of assuming that the target follows one of several fixed trajectories, we model the maneuvers of the target as a continuous probability distribution. This shift towards a probabilistic framework presents new challenges, particularly in the prediction of the future trajectory of the target, which is now described by a dynamic probability distribution rather than a single deterministic path.

In [6], a Monte Carlo method is used to approximate the distribution of future positions of a target, but due to the computational demands of the method, only a few samples can be generated.

In general, the future states of the target cannot be described as distinct trajectories, but rather as a time-varying probability distribution over the state space. Such problems are called multimodal trajectory prediction problems and have mostly been studied in the context of autonomous driving and pedestrians, but not in the context of guided missiles. One challenge for this kind of problem is that the probability distribution over the state space usually cannot be derived analytically, but rather has to be approximated in some way.

Ref. [7] gives a good overview of the different approaches that have been taken to solve this problem. The methods range from adding noise to a deterministic prediction over anchor methods [8], where likely end points of the trajectories are used to generate multiple trajectories, clustering and Gaussian Mixture Model approaches [9], over grid-based methods [10] to generative models Machine Learning (ML) techniques. One example of ML approaches is Generative Adversarial Networks (GANs) [11], where a classifier is trained to distinguish between real data and data generated by a generator network. In a minimax game, the generator network is trained to generate trajectories that are indistinguishable

from real trajectories.

Another ML approach is Variational Autoencoders (VAEs) [12], where an encoder network is trained to encode the input trajectory into a latent space, which is then decoded by a decoder network to generate a trajectory.

Lastly, Normalizing Flows (NFs) [13, 14] are a class of generative ML models that can be used to approximate probability distributions by transforming a simple base distribution into the desired, complex distribution and vice versa using a series of invertible transformations. Compared to other ML approaches like GANs or VAEs, NFs have the advantage that they can perform both sampling (of possible future states) and inference (i.e., calculating the probability of a given state). Moreover, they allow for efficient and exact computation of the probability of a given state, giving rise to heatmap-like visualizations of the probability density function (PDF) over the state space.

In [15] Conditional NFs (CNFs) are used to predict the future trajectories of pedestrians in a multimodal way. It employs recurrent neural networks to encode the past trajectory of the pedestrians which is used to calculate the conditional probability distribution over the future trajectory conditioned on the past trajectory. Each predicted trajectory consists of a sequence of positions with equal time intervals between them. In [16] a method is proposed to improve the quality and diversity of trajectories generated by NFs. By adding a diversity objective function, more diverse predictions for the future trajectory of vehicles with discrete time steps can be predicted conditioned on measurements and additional physical attributes. To accommodate the need for fast inference, [17] proposes a method to speed up the inference of NFs by reusing the results of previous computations to predict future trajectories of humans.

While most prediction methods aim to predict the future positions for discrete time steps, only the method presented in our previous work [18] is able to predict the probability distribution of the future positions for arbitrary times by using CNFs. With the CNFs approach, the distribution of the future position of stochastically moving targets can be predicted. The advantage of this approach is that the predictions in the form of a PDF can be interpreted and visualized as a heatmap, which can be used to gain insights into the predicted behavior of the target. However, almost all algorithms (guidance laws, collision avoidance, ...) expect deterministic trajectories, which is a simplification that might not hold in reality. To bridge this gap, this paper employs a clustering algorithm to cluster the samples generated by the CNFs into a set of representative trajectories, called virtual targets. Since the samples can be generated for equal time steps with equal sample sizes, most of the problems that arise when clustering trajectory data can be avoided. Thus, a time series k-means clustering algorithm [19] is used to generate trajectories of the virtual targets. These virtual targets can then be used as a drop-in replacement for deterministic trajectory predictions in guidance laws, path planning, or other applications that require deterministic trajectory predictions.

This paper leverages the data-driven approach of Conditional Normalizing Flows to learn the stochastic behavior of an aerial vehicle represented by a time-varying PDF of the future positions of the target. The contributions of this paper are two-fold:

1) A prediction framework leveraging CNFs to predict the distribution of the future positions of a stochastically

moving target, allowing for interpretable, target-agnostic, and fast predictions.

2) A time series k-means clustering algorithm is described to cluster the superimposed samples generated by the CNFs for each real target into a set of representative trajectories, called virtual target trajectories, which can be used as a drop-in replacement for deterministic trajectory predictions for various applications.

The remainder of this paper is structured as follows: In Sec. II, the problem of trajectory prediction for stochastic targets is described mathematically. Sec. III explains the theory of NFs and describes their application to the problem at hand. The subsequent clustering process to calculate the trajectories of the virtual targets is described in Sec. IV. These process is illustrated in Fig. 1. The generation of the training data is described in Sec. V. Sec. VI presents the results of the approach and Sec. VII concludes the paper.



Fig. 1 Illustration of the structure of the prediction framework.

II. Problem Statement

The main problem to be solved in this paper is the prediction of the future trajectory of the stochastically maneuvering target in a computationally efficient manner and its meaningful representation for downstream tasks that require deterministic trajectory predictions. It is assumed that either the stochastic model of the target dynamics is known or a dataset of target trajectories is available.

The future position $\mathbf{x}(t)$ at time *t* of a moving target performing random maneuvers is to be predicted given the initial position \mathbf{x}_0 at t = 0 and additional parameters ψ regarding the target dynamics, such as the ballistic coefficient or the maximum turn rate. More precisely, the PDF $p(\mathbf{x} \mid t, \mathbf{x}_0, \psi)$ is to be estimated, which describes the relative

probability of the target being at a given position x at time t conditioned on the initial position x_0 and the parameters ψ .

Additionally, an algorithm C_{n_v} to generate a set of n_v representative trajectories ϕ_i from set of PDFs $p_j(\mathbf{x} \mid t, \mathbf{x}_0, \psi)$ for n_r targets is to be developed, such that each trajectory $\phi_i = {\{\mathbf{x}_i(t_k)\}}_{k=1}^{n_t}$ consisting of the sequence of n_t positions \mathbf{x}_i at times t_k are generated for each virtual target *i*. Equation (1) describes the mapping of the input data to the output data of the CNFs model.

$$C_{n_{\nu}}: \{p_{j}(\mathbf{x} \mid t, \mathbf{x}_{0}, \psi)\}_{j=1}^{n_{r}} \mapsto \{\phi_{i}\}_{i=1}^{n_{\nu}}$$
(1)

III. Learning Stochastic Target Dynamics

The first step to solve the problem described in Sec. II is to learn the stochastic behavior of the target. In order to learn the probability distribution of the target's future position, NFs are used. NFs are a class of generative ML models that can be used to approximate probability distributions by transforming a simple base distribution into the desired, complex distribution and vice versa. Here, the complex distribution is the distribution of the position of the target at a given time and the base distribution is a Gaussian distribution. In the following, the theory of NFs is explained and their application to the problem at hand is described.

A. Normalizing Flows

NFs consist of a series of invertible transformations, which transform a sample from the base distribution into a sample from the complex distribution and vice versa. Since the transformations are invertible, the model can be used for both sampling and inference. Sampling means that the NFs model can be used to generate samples from the desired distribution by sampling from the base distribution and transforming the samples with the learned invertible transformations. Inference, or density estimation, means that the model can be used to calculate the probability density of a given sample from the complex distribution by transforming it to the base distribution and calculating the probability of the sample in the known base distribution and correcting for the volume change due to the transformation by multiplying with the determinant of the Jacobian of the transformation.

As displayed in Fig. 2, NFs describe a transformation $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$, which transforms samples \mathbf{x} with dimension d from the complex distribution $p(\mathbf{x})$ into samples \mathbf{z} with the same dimension from the base distribution $p(\mathbf{z})$. In this application, $p(\mathbf{z})$ is a Gaussian distribution with zero mean and unit variance and $p(\mathbf{x})$ denotes the distribution of the position \mathbf{x} of the target.

Equation (2) describes the transformation \mathbf{f} and its inverse \mathbf{f}^{-1} as a composition of n_1 invertible transformations \mathbf{f}_{Θ_i} , where Θ_i are the n_{Θ_i} parameters of the *i*-th transformation. The transformations \mathbf{f}_{Θ_i} can be any invertible function, provided they are easy to evaluate, invert, and differentiate. Moreover, the determinant of the Jacobian of the transformation \mathbf{f}_{Θ_i} must be easy to compute since it is needed for the calculation of the PDF of the complex distribution.



Fig. 2 Illustration of the transformation of points from a normal distribution to a complex distribution (and vice versa) using Normalizing Flows. Their relative probability is according to the probability density function above. Figure adapted from [18].



Fig. 3 Illustration of the Normalizing Flows concept with neural networks α_i and transformations f_{Θ_i} with parameters Θ_i . Figure adapted from [18].

Examples of such transformations include affine transformations (e.g., $\mathbf{f}_{\Theta_i}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$), as utilized in RealNVP [20]. In this paper, \mathbf{f}_{Θ_i} are rational quadratic spline transformations, which are elaborated upon in Sec. III.C.

The parameters Θ_i are the outputs of neural networks (NNs) α_i , which are trained to learn the parameters of the transformations.

NFs:

$$\begin{cases}
\mathbf{z} = \mathbf{f}(\mathbf{x}) = (\mathbf{f}_{\Theta_{n_{1}}} \circ \dots \circ \mathbf{f}_{\Theta_{2}} \circ \mathbf{f}_{\Theta_{1}})(\mathbf{x}) \\
\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}) = (\mathbf{f}_{\Theta_{1}}^{-1} \circ \dots \circ \mathbf{f}_{\Theta_{n_{1}}}^{-1})(\mathbf{z}) \\
\text{with } \Theta_{i} = \alpha_{i}(\mathbf{f}_{\Theta_{i-1}}(\mathbf{x}))
\end{cases}$$
(2)

Equation (3) depicts the input and output dimensions of the NNs α_i with *d* being the dimension of the input data and n_{Θ_i} being the size of the parameters of the transformation \mathbf{f}_{Θ_i} . The input for α_i is the output from the previous transformation $\mathbf{f}_{\Theta_{i-1}}$ as illustrated in Fig. 3.

$$\alpha_i : \mathbb{R}^d \to \mathbb{R}^{n_{\Theta_i}} \tag{3}$$

Using samples from the complex distribution, the weights of the NNs are optimized during the training process in a Maximum Likelihood Estimation approach, such that the observed training data points are most probable under the learned distribution. This is done by minimizing the negative log-likelihood of the training data points, which is equivalent to maximizing the likelihood of creating the training data points by sampling from the base distribution and transforming them with the NFs.

Thus, the optimal set of weights θ^* for the NNs α_l with $\theta = \{\theta_1, \dots, \theta_{n_l}\}$ is found by minimizing the negative log-likelihood of the n_d training data points \mathbf{x}_i when applying the NFs $p_{\theta}(\mathbf{x})$ with NN weights and biases θ as described in Eq. (4):

$$\theta^* = \arg\min_{\theta} \left(-\sum_{i=1}^{n_d} \log p_{\theta}(\mathbf{x}_i) \right)$$
(4)

For simplicity, the index θ indicating the set of NN weights and biases in the NFs model is dropped in the following. During the training process, singularities of the true PDF $p(\mathbf{x})$, i.e., points where the PDF has a value of infinity, can lead to problems. The reason for this is that it would violate the invertibility property since the inverse transformation \mathbf{f}^{-1} would not be defined at these points. A common solution to this problem, which is also applied in this paper, is the so-called noise injection, where a small amount of noise is added to the training data to dilute the singularities of the true PDF.

B. Conditional Normalizing Flows

The NFs model can be used to transform samples from the complex distribution to the base distribution (density estimation) and to sample from the complex distribution. However, a NFs model can only transform a simple base distribution into one complex distribution, i.e., for one time *t* and one set of parameters ψ . Since the model should not describe only one distribution, but rather a distribution for any given time *t* and additional parameters ψ regarding the target dynamics, the architecture must be extended. This can be achieved by using Conditional Normalizing Flows (CNFs) [14], which are a special kind of NFs that can be conditioned on some input to calculate a conditional probability $p(\mathbf{x} \mid t, \psi)$ in contrast to the unconditional probability $p(\mathbf{x})$ provided by normal NFs. In this case, *t* and ψ serve as additional input for the model, the so-called conditioning variables. More precisely, they are used as an additional input for the NNs desides the output of the previous transformation $\mathbf{f}_{\Theta_{i-1}}$. To differentiate these NNs with more inputs from the NNs α_i used in the NFs model, they are denoted as β_i .

As displayed in Fig. 4, CNFs describe a transformation **f**, which transforms samples **x** from the complex distribution $p(\mathbf{x} \mid t, \psi)$ into samples **z** from the base distribution $p(\mathbf{z})$.

Equation (5) describes the transformation \mathbf{f} and its inverse \mathbf{f}^{-1} as a composition of n_1 invertible transformations \mathbf{f}_{Θ_i} , where Θ_i are the n_{Θ_i} parameters of the *i*-th transformation. The parameters Θ_i are the outputs of NNs β_i , which are



Fig. 4 Illustration of the Conditional Normalizing Flows concept with neural networks β_i and transformations f_{Θ_i} with parameters Θ_i , conditioned on the time *t* and dynamics parameters ψ .

trained to learn the parameters of the transformations conditioned on t and ψ .

CNFs:

$$\begin{cases}
\mathbf{z} = \mathbf{f}(\mathbf{x}, t, \psi) = (\mathbf{f}_{\Theta_{n_{1}}} \circ \cdots \circ \mathbf{f}_{\Theta_{2}} \circ \mathbf{f}_{\Theta_{1}})(\mathbf{x}, t, \psi) \\
\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}, t, \psi) = (\mathbf{f}_{\Theta_{1}}^{-1} \circ \cdots \circ \mathbf{f}_{\Theta_{n_{1}}}^{-1})(\mathbf{z}) \\
\text{with } \Theta_{i} = \beta_{i}(\mathbf{f}_{\Theta_{i-1}}(\mathbf{x}), t, \psi)
\end{cases}$$
(5)

Equation (6) depicts the input and output dimensions of the NNs β_i . The input for β_i is the output from the previous transformation $\mathbf{f}_{\Theta_{i-1}}$, the time *t*, and the parameters ψ as illustrated in Fig. 4.

$$\beta_i : \mathbb{R}^{d+1+n_{\psi}} \to \mathbb{R}^{n_{\Theta_i}} \tag{6}$$

C. Application of Conditional Normalizing Flows

In our approach, we utilize CNFs to model the distribution of the target's position under varying conditions of time and target dynamics. Specifically, we implement CNFs with a technique known as Masked Autoregressive Flow (MAF) combined with rational quadratic splines.

MAF [21] is a method that leverages autoregressive models to generate samples from a complex distribution. It does so by modeling the distribution as a sequence of conditional distributions where each dimension depends on the previous ones, giving it a high degree of flexibility. This autoregressive property makes it suitable for our task of predicting the target's position under different conditions.

Rational quadratic splines [22] are used as transformation functions \mathbf{f}_{Θ_i} to further enhance the flexibility and expressive power of the model compared to affine transformations as used in RealNVP [20]. They allow to capture complex patterns in the data by adjusting the shape of the spline as needed, making the model adaptable to a wide range of target distributions.

While a detailed technical description of MAF and rational quadratic splines is beyond the scope of this paper, we refer the interested reader to the original papers for a more in-depth understanding of these techniques. The combination of MAF and rational quadratic splines in CNFs allows for a flexible and expressive model that can accurately capture the distribution of the target's position.

Leveraging CNFs allows to transform samples from the complex distribution to the base distribution for a given time *t* and additional parameters ψ regarding the target dynamics which serve as the conditioning variables. Thus, the PDF $p(\mathbf{x} \mid t, \mathbf{x}_0, \psi)$ can be evaluated by transforming a position \mathbf{x} from the complex distribution to the base distribution and calculating the probability of the sample in the base distribution. Moreover, the model can be used to sample from the complex distribution for a given time *t* and parameters ψ by sampling from the base distribution and transforming the samples with the learned invertible transformations. The calculation time for the evaluation of the PDF or sampling is constant, since the model learns the distribution of the target's future position and not the dynamics of the target itself. Not only is the sampling using the CNFs much faster than Monte Carlo simulation (especially for evaluations at the final time), the use of CNFs also allows for an exact computation of the PDF, which is not possible with the Monte Carlo trajectory generation method. To facilitate the training process, the training data is normalized to a range of [-1, 1], including the positions \mathbf{x} , the time *t*, and the parameters ψ .

D. Translation and Rotation of the Predicted Target Positions

Since the true distribution is independent of the absolute position and the rotation of the target for the scenarios used in this paper, the position of the target can be predicted for any initial position and orientation of the target by transforming the position of the target to the origin and rotating it such that the target is flying northbound. Thus, the prediction task can be simplified from predicting $p(\mathbf{x} \mid t, \mathbf{x}_0, \psi)$ to predicting $p(\mathbf{x} \mid t, \psi)$. This simplified PDF is then translated to the current normalized position $\mathbf{\bar{p}}$ and rotated according to the orientation of the target. The position $\mathbf{\bar{p}}$ is calculated by applying the same normalization procedure to the current position \mathbf{p} as was applied to the training data.

In a two-dimensional scenario, the rotation matrix **R** is calculated from the orientation χ of the target as described in Eq. (7). For a three-dimensional scenario, the extension is straightforward.

$$\mathbf{R} = \begin{vmatrix} \cos(\chi) & -\sin(\chi) \\ \sin(\chi) & \cos(\chi) \end{vmatrix}$$
(7)

The samples **x** predicted by the CNFs are then moved to the normalized position $\mathbf{\bar{p}}$ of the target by performing the rotation and translation as described in Eq. (8) and visualized in Fig. 5. Note that these samples \mathbf{x}_{real} are still in the normalized coordinate system.

$$\mathbf{x}_{\text{real}} = \mathbf{R}\mathbf{x} + \bar{\mathbf{p}} \tag{8}$$



Original Distribution

Fig. 5 Illustration of the translation (by \bar{p}) and the rotation (by χ) of the predicted target positions.

Nevertheless, if the true target distribution depends on the initial position and orientation, i.e., if the motion of the target is not translation and rotation invariant, the initial position and orientation of the target can be used as additional conditioning variables to allow for the prediction of the position of the target for any initial position and orientation.

E. Outlier Removal

Due to the normalization of the training data, the domain that the CNFs are trained on is limited to [-1, 1]. Thus, the performance of the CNFs outside this domain can be arbitrarily poor. Furthermore, these samples might even be physically impossible, e.g., if the velocity of the target is assumed to be constant.

To prevent the usage of such outliers, samples outside this range plus a margin of three times the standard deviation of the noise injection are removed from the set of generated samples.

IV. Generation of Virtual Target Trajectories

While the CNFs can be used to predict the distribution of the future positions of the target, the samples generated by the CNFs are not directly usable for downstream tasks, since they are not deterministic trajectories. Besides the development of guidance laws that can handle stochastic trajectories, a more straightforward approach is to generate a set of representative trajectories from the samples, which can be used as a drop-in replacement for deterministic trajectory predictions. To this end, a time series k-means clustering algorithm is used to cluster the samples into a set of representative trajectories. This approach, i.e., the combination of the predictions for multiple targets and the clustering of the samples, as well as the renormalization of the cluster means, is described in the following sections.

A. Combination of Targets

For each of the n_r real targets $i = 1, ..., n_r$, the respective learned CNFs model is used to generate $j = 1, ..., n_s$ samples $\mathbf{x}_{i,j,k}$ of the future positions of the target *i* for each time step $k = 1, ..., n_t$. First, $n_r \cdot n_s$ samples $\mathbf{z}_{i,j}$ are drawn from the base distribution, which is a Gaussian distribution with zero mean and unit variance of dimension *d*, i.e.

$$\mathbf{z}_{i,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \quad \text{for } i = 1, \dots, n_{\mathrm{r}}, \ j = 1, \dots, n_{\mathrm{s}}$$

$$\tag{9}$$

Then, the samples are transformed by the CNFs f to obtain samples from the complex distribution:

$$\mathbf{x}_{i,j,k} = \mathbf{f}^{-1}(\mathbf{z}_{i,j}, t_k, \psi_i) \quad \text{for } i = 1, \dots, n_{\mathrm{r}}, \ j = 1, \dots, n_{\mathrm{s}}, \ k = 1, \dots, n_t$$
(10)

 $\mathbf{x}_{i,j,k}$ consists of *d* dimensions, where *d* is the dimension of the position of the target which can be 2 or 3 depending on the scenario. If the targets are of the same type, i.e., their dynamics and their parameters ψ are identical, the same generated samples can be used for all the targets to save computation time. Otherwise, a separate CNFs model has to be trained for each type of target and then evaluated for each target. Since most target distributions are not dependent on the current position and azimuth, the samples from the CNFs which are generated in a normalized coordinate system with axis limits of [-1, 1] can be moved to the normalized current position $\mathbf{\bar{p}}_i$ with orientation χ_i of the real target by performing a rotation and a translation as described in Sec. III.D.

B. Clustering

After generating and then rotating and translating the samples or each real target *i*, the trajectories $\mathbf{y}_{i,j}$ of the samples are clustered to obtain a set of representative trajectories. To this end, a time series k-means clustering algorithm is used, which assigns each sample trajectory to one of n_v clusters, where n_v is the number of virtual targets that can be chosen by the user. The resulting trajectories of the cluster means are then used as the virtual targets.

The input for the clustering process is a set *Y* comprised of $n_r \cdot n_s$ flattened trajectories $\mathbf{y}_{i,j}$ of size $(n_t \cdot d \times 1)$, which are generated as follows:

$$\mathbf{y}_{i,j} = \left[\mathbf{x}_{i,j,1}^T, \dots, \mathbf{x}_{i,j,n_t}^T\right]^T$$
(11)

$$Y = \{\mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \dots, \mathbf{y}_{n_{\rm r},n_{\rm s}}\}$$
(12)

With this flattened input, a k-means clustering algorithm as described in [19] can be applied to obtain the cluster means, which are then used as the virtual targets.



Fig. 6 Illustration of the resulf of k-means clustering for two dimensions.

Figure 6 illustrates the result of the k-means clustering for two dimensions. This only serves as a visualization of the clustering process and does not represent the actual dimensions of the input data, since the input data, i.e., the flattened trajectories, are of size $n_t \cdot d$ instead of two dimensions.

By choosing a desired n_v , the number of virtual targets, the algorithm assigns each trajectory of the $n_r \cdot n_s$ samples to one of the n_v clusters C_i such that the sum of the squared Euclidean distances between the samples and the cluster means is minimized:

$$\mathbf{c}_{1}^{*}, \dots, \mathbf{c}_{n_{v}}^{*} = \operatorname*{argmin}_{\mathbf{c}_{1},\dots,\mathbf{c}_{n_{v}}} \sum_{i=1}^{n_{v}} \sum_{\mathbf{y} \in C_{i}} ||\mathbf{y} - \mathbf{c}_{i}||^{2}$$

s.t. $\forall i \neq j : \quad C_{i} \cap C_{j} = \emptyset$
 $\bigcup_{i=1}^{n_{v}} \quad C_{i} = Y$ (13)

The result of the clustering process is a set of n_v optimal cluster means $\mathbf{c}_1^*, \ldots, \mathbf{c}_{n_v}^*$, each consisting of dimensions $(n_t \cdot d, 1)$, which can then be reshaped to n_v trajectories with dimensions (n_t, d) each:

$$\mathbf{c}_{i}^{*} = \begin{vmatrix} \mathbf{c}_{i,1}^{*} & \dots & \mathbf{c}_{i,d}^{*} \\ \mathbf{c}_{i,d+1}^{*} & \dots & \mathbf{c}_{i,2d}^{*} \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{i,(n_{t}-1)\cdot d+1}^{*} & \dots & \mathbf{c}_{i,n_{t}\cdot d}^{*} \end{vmatrix}$$
(14)
$$\phi_{i}(k \cdot \Delta t) = \left(\mathbf{c}_{i,(k-1)\cdot d+1}^{*}, \dots, \mathbf{c}_{i,k\cdot d}^{*}\right)$$
(15)

C. Renormalization

As mentioned in Sec. III.C, the samples are generated in a normalized coordinate system with axis limits of [-1, 1]. To obtain the real-world positions and times of the virtual targets, the cluster means have to be renormalized to the original coordinate system, with the inverse of the normalization process mentioned in Sec. III.E. After applying the renormalization, the trajectories of the virtual targets are obtained, which can be used for further analysis and downstream tasks.

V. Data Generation

Before training the CNFs model, data has to be generated to train the model. This data consists of trajectories of simulated targets performing random maneuvers generated with a Monte Carlo simulation. If real data is available, it can be used to train the model instead of generating synthetic data.

A. Simple Target

As a practical illustration of the problem described in Sec. II, consider a target flying in the horizontal plane, i.e., $\mathbf{x} \in \mathbb{R}^2$, with a constant velocity and performing random maneuvers. This scenario serves as an instructive example of a stochastically moving target. Three different types of maneuvers are assumed: left turn, right turn, and straight flight. At the beginning of a trajectory (at the origin, flying northbound), the type and duration of the maneuver and the radius (bounded by the lateral acceleration) of the turn are randomly chosen from a uniform distribution with parameters depicted in Table 1. After the maneuver is completed, new maneuvers are randomly chosen until the total duration of the trajectory is reached. Since no additional parameters are required for this simple scenario, ψ is a vector of dimension zero ($\psi \in \mathbb{R}^0$).

Trajectories created with this approach are shown in Fig. 7. Each dot denotes the change of the maneuver. Since the trajectories are simulated with a time discretization of 0.1 s, trajectory data can be obtained and saved for all the simulated time steps.

Figure 8 depicts histograms of samples of the distribution of the target positions for different times. The data was created with a computationally expensive Monte Carlo simulation.

B. Complex Target

In order to demonstrate the capabilities of the model, a more complex scenario is considered. A ballistic target influenced by disturbances is considered, flying along a ballistic trajectory in three dimensions, i.e., $\mathbf{x} \in \mathbb{R}^3$. Due to the disturbances, the target does not fly the ballistic trajectory exactly, but with some deviations, leading to a distribution of possible trajectories, which is to be modeled. The ballistic trajectory is calculated according to Eq. (16) in the North-East-Down frame.



 Table 1
 Properties of the target maneuvers. Table reproduced from [18].

Fig. 7 Ten randomly generated simple target trajectories with a duration of 100 s. Figure adapted from [18].

$$\dot{\mathbf{x}} = \mathbf{v}$$
(16)
$$\dot{\mathbf{v}} = \mathbf{g} + \mathbf{d} + \mathbf{w}$$

$$\mathbf{d} = -\frac{1}{2 \cdot BC} \cdot \rho \cdot ||\mathbf{v}|| \cdot \mathbf{v}$$
$$\mathbf{g} = [0, 0, g]^{\top}$$
$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$
$$BC = \frac{m}{A + CD}$$

The change of the position \mathbf{x} is equal to the velocity \mathbf{v} and the change of the velocity \mathbf{v} is equal to the sum of the gravitational acceleration \mathbf{g} , the deceleration \mathbf{d} due to drag, and the disturbances \mathbf{w} . The disturbance \mathbf{w} is modeled as a zero-mean Gaussian noise with a covariance matrix $\boldsymbol{\Sigma}$.

The drag **d** is calculated with the ballistic coefficient BC, which is the ratio of the mass *m* of the target to the product of the cross-sectional area *A* and the drag coefficient C_D of the target. The goal of the model is to calculate the distribution of the position **x** of the ballistic target after a certain time *t*.

Table 2 depicts the parameters of the ballistic trajectories used as training data. The ballistic coefficient of the target is sampled from the uniform distribution $\mathcal{U}(200, 800) \text{ kg/m}^2$ to allow for the prediction of the distribution of the position of the target for any ballistic coefficient in the learned range. This serves as an example of how additional parameters ψ can be included in the model to predict the distribution of the target's position for different parameters. Thus, ψ is a vector of dimension one ($\psi \in \mathbb{R}^1$) in this scenario.

Parameter	Value
Trajectory duration	100 s
Time discretization	0.1 s
Air density ρ	1.225 kg/m ³
Target ballistic coefficient	$\mathcal{U}(200, 800) \text{ kg/m}^2$
Target initial position	[0, 0, -1000] m
Target initial velocity	[100, 0, 0] m/s
Disturbance covariance matrix Σ	$I_3 m^2/s^4$
Gravitational acceleration g	9.81 m/s ²

 Table 2
 Parameters of the ballistic trajectories used as training data. Table adapted from [18].

Compared to the scenarios in Sec. V.A, the dynamics here differ dramatically:

- The target velocity is not constant but changes nonlinearly over time due to drag and the disturbance forces.
- The scenario takes place in three dimensions, instead of two dimensions.
- An additional parameter ψ is added to the training data, namely the ballistic coefficient of the target.

with

Figure 13 depicts samples of the distribution of the target positions for different times and $BC = 500 \text{ kg/m}^2$, obtained by Monte Carlo simulation. It can be seen that the distribution of the target positions widens over time, due to the influence of the noise.

VI. Simulation Results

With the above-described CNFs approach, the distribution of the target's position can be modeled for any given time and target dynamics and then clustered to obtain a set of representative trajectories. The results of the stochastic predictions by the CNFs are presented in Sec. VI.A, followed by the generation of virtual target trajectories in Sec. VI.B.

A. Stochastic Predictions

In the following, three different scenarios are considered: First, a stochastically moving target is examined in Sec. VI.A.1. Second, the application of the CNFs to predict trajectories of targets that are moving deterministically is presented in Sec. VI.A.2. Finally, the prediction of the position of a ballistic target with stochastic disturbances is examined in Sec. VI.A.3.

For all three scenarios, the same NNs and CNFs architecture are used, which are depicted in Tables 3 and 4.

1. Stochastic Maneuvers

First, the scenario described in Sec. V.A is examined. The latent dimension of the model, i.e., the dimension of the base distribution, is set to 2, which means that the model can learn the distribution of the position of the target in two dimensions, namely the x- and y-coordinate. Thus, a two-dimensional Gaussian distribution is used as the base distribution that is transformed by the model to obtain the distribution of the position of the target.

After training the CNFs model for 1000 epochs (requiring 81 s of computation time*) on the generated data described in Sec. V and displayed in Fig. 8, the model is evaluated on the test data. The results of the model evaluation are shown in Fig. 9. The figure depicts the absolute frequency from 10^4 samples (as a representation of the PDF) of the position of the target at different times. Figure 10 depicts the learned PDF of the position of the target at different times.

When comparing the results to the test data depicted in Fig. 8, it can be seen that the model is able to predict the position of the target quite well with a test loss of -2.85. Furthermore, the calculation time for the model evaluation is independent of the time *t* at which the distribution shall be predicted. Compared to the Monte Carlo simulation, the time to create 10^4 samples is only about 0.17 s, compared to 6.64 s required for the Monte Carlo simulation for the a simulated duration of 100 s. Not only is the sampling much faster, the use of CNFs also allows for an exact computation of the PDF, which is not possible with the Monte Carlo trajectory generation method. Overall, we can conclude that CNFs are a suitable tool for modeling the distribution of the target's position.

^{*}All computations were performed with a Ryzen 7 6800U processor with 16 GB RAM.



Fig. 8 Histograms of 10^4 samples of the probability density function of the position of the target at different times obtained by Monte Carlo simulation. Figure reproduced from [18].



Fig. 9 Samples of the learned probability density function of the position of the target at different times. Figure reproduced from [18].



Fig. 10 Learned probability density function of the position of the target at different times. Figure reproduced from [18].

Parameter	Value
Number of hidden layers	2
Number of hidden units	32
Activation function	ReLU
Batch size	1000
Learning rate	0.003
Number of epochs	1000
Optimizer	Adam
Loss function	Negative log-likelihood

 Table 3
 Parameters of the Neural Network. Table adapted from [18].

Table 4 Parameters of the Conditional Normalizing Flows model. Table adapted from [18].

Parameter	Value
Number of flow layers n_l	4
Base distribution	Standard Gaussian
Number of trajectories	10^{4}
Training data	80%
Validation data	20%
Noise injection standard deviation	0.01

2. Deterministic Maneuvers

To prove that a CNFs model with the same architecture can also be used for deterministic maneuvers, it is applied to predict the position of the target described in Sec. V.A after a deterministic maneuver of infinite duration. The maneuvers tested were flying straight, a left turn, and a right turn, each with a fixed lateral acceleration of 3 m/s^2 . In essence, this means the target can choose one of three different deterministic trajectories. Compared to the previous scenario, the target dynamics have changed, thus new training data is created in a similar way as before, but with the new target dynamics. Figure 11 depicts the distribution of the target positions at different instances in the test data.

With the help of the new training data, the model is trained to predict the PDF of the position of the target at different times. Figure 12 depicts samples of the learned distribution of the predicted target positions for different times. When comparing it to the test data in Fig. 11, it can be seen that the model is able to predict the position of the target quite well. Nevertheless, there are some outliers, which are caused by the fact that the values of the learned PDF are not exactly zero. Thus, the model predicts a very low, but non-zero, probability for the target to be at a position other than the three possible positions.

3. Ballistic Targets

The same CNFs model as in the previous sections is used to predict the distribution of the ballistic trajectories described in Sec. V.B, but with a latent dimension (the dimension of the base distribution) of three, since the ballistic



Fig. 11 Test data: Histograms of samples of the probability density function of the position of the target at different times. Figure reproduced from [18].



Fig. 12 Learned distribution: Histograms of samples of the probability density function of the position of the target at different times which does not change maneuvers. Figure reproduced from [18].

trajectories are simulated in three dimensions. The training process was similar to the previous sections and required about 119 s.

With the help of the trained model, the distribution of the ballistic trajectory can be predicted without the need to simulate the ballistic trajectory multiple times. The required computation time for the prediction of 10^4 samples using the CNFs is about 0.27 s, compared to the 178 s of the Monte Carlo simulation which was used to generate the training data.

Figure 14 depicts the samples of $p(\mathbf{x} \mid t, \psi)$ created for $t \in \{0, 45, 90\}$ s into the flight of the ballistic trajectory and $\psi = BC = 500 \text{ kg/m}^2$. When comparing the learned distribution to the test data depicted in Fig. 13, which was obtained through costly Monte Carlo simulation, we observe that the model is able to predict the distribution of the ballistic trajectory quite well. While there are some samples (e.g., three samples in Fig. 14c) that do not fit the training data, the overall shape of the distribution is similar to the test data depicted in Fig. 13, with a correct position of the mean of the distribution and a correct shape of the distribution. Comparing this to the total of 10⁴ samples generated by the model, the number of such anomalies is considrabily low.

Only the distribution of the position of the target at 0 s is not predicted correctly: Whereas the mean of the distribution is predicted correctly, the shape of the distribution is not predicted correctly, being elongated along the direction of flight. A reason for this could be the training data: According to Table 2, all trajectories start at the position [0, 0, -1000] m, leading to a singularity in the PDF at this position, since the value of the PDF at this position is infinite to conserve an integral of 1. The injection of noise mitigates this problem but does not solve it completely.

B. Generation of Virtual Target Trajectories

Through the use of the CNFs, stochastic predictions of the target's position can be made for any given time and target dynamics. To make use of these predictions in applications such as guidance laws, path planning, etc., a set of representative trajectories is generated from the samples as described in Sec. IV. In the following, the results of the generation of virtual target trajectories are presented for various numbers of real targets and virtual targets, using the stochastically moving target from Sec. VI.A.1 as an example. Table 5 shows the parameters used in the clustering process.

Table 5Parameters of the clustering process.

Parameter	Value
Number of samples per target and time step	10 ⁴
Number of time steps	10
Tolerance to stop the clustering algorithm	10^{-4}



Fig. 13 Monte Carlo simulation: samples of the probability density function of the position of the target with $BC = 500 \frac{kg}{m^2}$ at different times. Figure reproduced from [18].



Fig. 14 Samples of the learned probability density function of the position of the target with the additional parameter $\psi = BC = 500 \text{ kg/m}^2$ at different times. Figure reproduced from [18].

1. Single Target

The prediction for a single real target is shown in Fig. 15 with one to four virtual targets. When only one virtual target (Fig. 15a) is predicted, the virtual target trajectory is similar to the average of the samples of the real target trajectories. For applications where multiple virtual targets are required, the number of clusters can be increased. The prediction of two virtual targets (Figures 15b) splits the samples roughly symmetrically in the East-West direction, implying the possibility of a left and a right turn. When three virtual targets are predicted (Fig. 15c), the samples are assigned to three clusters: a left turn, a right turn, and a straight flight. Interestingly, this concides with the multi-hypothesis approach in [5] where the target is assumed to perform one of multiple maneuvers, including a left turn, a right turn, and a straight flight. However, the problem of defining the radii of the turns is avoided by the virtual target approach as the k-means clustering algorithm automatically determines the representative trajectories. When four virtual targets are predicted (Fig. 15d), the samples of the real target are clustered into four virtual targets, resulting in a more detailed prediction of the target's future position.

Overall, the trajectories cover a wide range of possible virtual target trajectories, without explicitly defining the

maneuvers. While the trajectories are not very smooth, the quality of the prediction can be improved by increasing the number of samples and the number of time steps. Furthermore, smoothing techniques can be applied to the generated trajectories to obtain smoother trajectories if required.



Fig. 15 Prediction of virtual target trajectories for a single real target (initial position: [0,0] km, flying northbound) with (a) one, (b) two, (c) three, and (d) four virtual targets and the respective clustered samples at time 90 s.

2. Multiple Targets

As described in Sec. IV.A, the approach can be extended to multiple real targets by generating samples for each target or type of target and then clustering the samples to obtain a set of representative trajectories. In the following, the results for three real targets are shown, first for spatially separated targets and then for targets close to each other. The total computation time to generate the 200 samples for each of the 10 time steps is 0.042 s. Clustering the samples takes

about 0.1 s, regardless of the number of real and virtual targets.

Figure 16 shows the prediction for three real spatially separated targets with one to four virtual targets. When only one virtual target is predicted for each real target, the virtual target trajectories are similar to the average of the real target trajectories (Fig. 16a). Increasing the number of virtual targets to two or three still results in the prediction of straight trajectories, just the position of the virtual targets is shifted (Figures 16b, 16c). When the number of virtual targets is higher than the number of real targets (Fig. 16d), the samples of one real target are clustered into multiple virtual targets, similar to Fig. 15.

When the real targets are close to each other, as shown in Fig. 17, the distributions of the target positions overlap, leading to one left turn, one right turn, and one straight flight as virtual target trajectories.



Fig. 16 Prediction of virtual target trajectories for three real spatially separated targets (initial positions: [0, -40] km, [0, 0] km, [0, 40] km, all targets flying northbound) with (a) one, (b) two, (c) three, and (d) four virtual targets and the respective clustered samples at time 90 s.



Fig. 17 Prediction of virtual target trajectories for three real targets close to each other (initial positions: [0, -5], km[0, 0] km, [0, 5] km, all targets flying northbound) with (a) one, (b) two, (c) three, and (d) four virtual targets and the respective clustered samples at time 90 s.

VII. Conclusion

In this paper, an approach for the prediction of the representative trajectories of a stochastic target, denoted as virtual target trajectories, and the probability density function of its future position is presented. The approach is based on the usage of Conditional Normalizing Flows, which are trained with the help of Monte Carlo simulation data, and time series k-means clustering to generate a set of representative trajectories. The presented results demonstrate the method's effectiveness in predicting target positions for various scenarios, including targets with stochastic maneuvers, deterministic maneuvers, and ballistic trajectories with additional parameters. The approach is target-agnostic and can be applied to different target types with appropriate training data.

Using Conditional Normalizing Flows, the position of the target at any given time can be predicted either by directly calculating the learned probability density function or by sampling from it. This approach is useful in various applications where predicting the target's position is essential. Compared to Monte Carlo simulations, the approach is computationally more efficient and allows for the calculation of the probability density function of the target's position. Since most targets do not follow perfectly deterministic trajectories, the usage of a stochastic predictor can take the uncertainty of the target trajectory into account, potentially leading to more robust downstream applications. When deterministic trajectories are required, the samples can be clustered to obtain a set of representative trajectories. Thus, the presented approach allows to predict deterministic trajectories for stochastically moving targets, making it a versatile tool for trajectory prediction. It can be used as a drop-in replacement for deterministic trajectory predictions used in areas such as guidance laws, path planning, and other applications where a prediction of the target's trajectory is needed.

Acknowledgments

AI tools (namely ChatGPT and GitHub Copilot) were used to improve the readability of the manuscript and to assist with TikZ figures.

References

- Adler, F. P., "Missile Guidance by Three-Dimensional Proportional Navigation," *Journal of Applied Physics*, Vol. 27, No. 5, 1956, pp. 500–507. https://doi.org/10.1063/1.1722411, URL https://doi.org/10.1063/1.1722411.
- [2] Nesline., F. W., and Zarchan, P., "A New Look at Classical vs Modern Homing Missile Guidance," *Journal of Guidance and Control*, Vol. 4, No. 1, 1981, pp. 78–85. https://doi.org/10.2514/3.56054, URL https://doi.org/10.2514/3.56054.
- [3] Guo, Y., Hawkins, M., and Wie, B., "Applications of Generalized Zero-Effort-Miss/Zero-Effort-Velocity Feedback Guidance Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 3, 2013, pp. 810–820. https://doi.org/10.2514/1.58099, URL https://doi.org/10.2514/1.58099.
- [4] Dwivedi, P., Bhale, P., Bhattacharyya, A., and Padhi, R., "Generalized estimation and predictive guidance for evasive targets,"

IEEE Transactions on Aerospace and Electronic Systems, Vol. 52, No. 5, 2016, pp. 21111–2122. https://doi.org/10.1109/TAES. 2016.140861.

- [5] Schneider, M., and Fichter, W., "Multi-Hypothesis Guidance With Interacting Multiple Model Filter," *AIAA SCITECH 2022 Forum*, 2022, p. 1846.
- [6] Shaviv, I., and Oshman, Y., "Estimation-guided guidance," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006, p. 6217.
- [7] Huang, R., Xue, H., Pagnucco, M., Salim, F. D., and Song, Y., "Vision-Based Multi-Future Trajectory Prediction: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2025, p. 1–18. https://doi.org/10.1109/tnnls.2025.3550350, URL http://dx.doi.org/10.1109/TNNLS.2025.3550350.
- [8] Chai, Y., Sapp, B., Bansal, M., and Anguelov, D., "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction," *Conference on Robot Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:204509212.
- [9] Barratt, S. T., Kochenderfer, M. J., and Boyd, S. P., "Learning Probabilistic Trajectory Models of Aircraft in Terminal Airspace From Position Data," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 9, 2019, pp. 3536–3545. https://doi.org/10.1109/TITS.2018.2877572.
- [10] Guo, K., Liu, W., and Pan, J., "End-to-End Trajectory Distribution Prediction Based on Occupancy Grid Maps," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 2232–2241. https://doi.org/10.1109/CVPR52688.2022.00228, URL https://doi.ieeecomputersociety.org/10.1109/CVPR52688. 2022.00228.
- [11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial networks," *Commun. ACM*, Vol. 63, No. 11, 2020, p. 139–144. https://doi.org/10.1145/3422622, URL https: //doi.org/10.1145/3422622.
- [12] Kingma, D. P., and Welling, M., "Auto-Encoding Variational Bayes," 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.
- [13] Rezende, D., and Mohamed, S., "Variational Inference with Normalizing Flows," *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 37, edited by F. Bach and D. Blei, PMLR, Lille, France, 2015, pp. 1530–1538. URL https://proceedings.mlr.press/v37/rezende15.html.
- [14] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B., "Normalizing Flows for Probabilistic Modeling and Inference," *Journal of Machine Learning Research*, Vol. 22, No. 57, 2021, pp. 1–64. URL http://jmlr.org/papers/v22/19-1028.html.
- [15] Schöller, C., and Knoll, A., "FloMo: Tractable Motion Prediction with Normalizing Flows," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 7977–7984. https://doi.org/10.1109/IROS51168.2021.9636445.

- [16] Ma, Y. J., Inala, J. P., Jayaraman, D., and Bastani, O., "Likelihood-Based Diverse Sampling for Trajectory Forecasting," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2020, pp. 13259–13268. URL https://api.semanticscholar. org/CorpusID:237487995.
- [17] Maeda, T., and Ukita, N., "Fast Inference and Update of Probabilistic Density Estimation on Trajectory Prediction," 2023 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 9761–9771. https://doi.org/10.1109/ICCV51070.2023.00898, URL https://doi.ieeecomputersociety.org/10.1109/ICCV51070. 2023.00898.
- [18] Schneider, M., Loureiro, R., Cunis, T., and Fichter, W., "Trajectory Prediction for Missile Targets: A Probabilistic Approach Using Machine Learning," *Proceedings of the 2024 CEAS EuroGNC conference*, Bristol, UK, 2024. CEAS-GNC-2024-103.
- [19] MacQueen, J., et al., "Some methods for classification and analysis of multivariate observations," *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1, Oakland, CA, USA, 1967, pp. 281–297.
- [20] Dinh, L., Sohl-Dickstein, J., and Bengio, S., "Density estimation using Real NVP," International Conference on Learning Representations, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.
- [21] Papamakarios, G., Pavlakou, T., and Murray, I., "Masked Autoregressive Flow for Density Estimation," Advances in Neural Information Processing Systems, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/ file/6c1da886822c67822bcf3679d04369fa-Paper.pdf.
- [22] Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G., "Neural Spline Flows," Advances in Neural Information Processing Systems, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.