# Efficient Trajectory Forecasting and Generation with Conditional Flow Matching

Sean Ye[1] and Matthew C. Gombolay[1]

***Abstract*— Trajectory prediction and generation are crucial for autonomous robots in dynamic environments. While prior research has typically focused on either prediction or generation, our approach unifies these tasks to provide a versatile framework and achieve state-of-the-art performance. While diffusion models excel in trajectory generation, their iterative sampling process is computationally intensive, hindering robotic systems' dynamic capabilities. We introduce Trajectory Conditional Flow Matching (T-CFM), a novel approach using flow matching techniques to learn a solver time-varying vector field for efficient, fast trajectory generation. T-CFM demonstrates effectiveness in adversarial tracking, real-world aircraft trajectory forecasting, and long-horizon planning, outperforming state-of-the-art baselines with 35% higher predictive accuracy and 142% improved planning performance. Crucially, T-CFM achieves up to 100× speed-up compared to diffusion models without sacrificing accuracy, enabling real-time decision making in robotics. Codebase: https://github.com/CORE-Robotics-Lab/TCFM**

## I. INTRODUCTION

Robots of the future will require fast and accurate trajectory forecasting techniques to navigate complex, dynamic environments and interact with other agents safely and efficiently. Trajectory forecasting deals with the problem of estimating an agent's future behavior while trajectory generation deals with planning feasible paths for an agent to follow. These techniques are crucial for various robotics applications, such as autonomous driving [19], multi-robot coordination [25], and social navigation [4]. By generating long-horizon plans and accurately predicting the future trajectories of dynamic agents, robots can make better decisions and adapt to changing conditions in real-time.

In recent years, deep learning approaches have achieved impressive results on trajectory forecasting benchmarks by learning complex patterns and distributions from large datasets. Compared to traditional methods such as Kalman Filters [14] and Particle Filters [7], learning-based methods excel at tasks where models of an agent's behavior are unknown or hard to predict. In particular, generative models such as variational autoencoders (VAEs) and generative adversarial networks (GANs) have shown promise in modeling the inherent multimodality and stochasticity in agent behaviors. More recently, denoising diffusion probabilistic models (DDPMs) [21] have emerged as a powerful class

*This work is supported in part by Naval Research Laboratory (NRL) under grant number N00173-21-1-G009 and the National Science Foundation under grant CNS-2219755,

[1]All authors are associated with the Institute of Robotics and Intelligent Machines (IRIM), Georgia Institute of Technology, Atlanta, GA, USA.

Corresponding Author: Sean Ye, seancye@gatech.edu

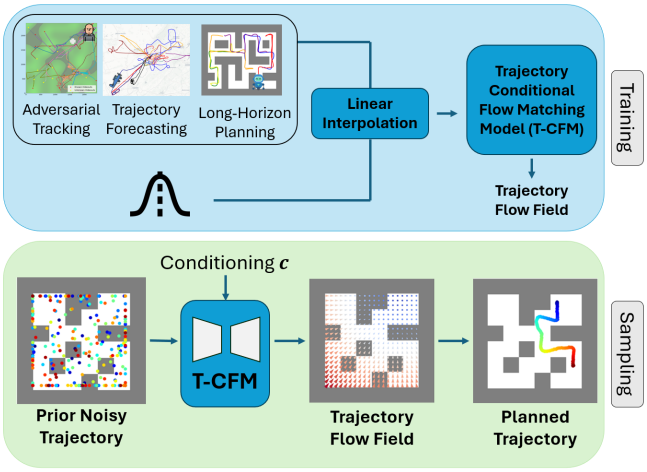



Fig. 1: Trajectory Conditional Flow Matching (T-CFM) is our novel trajectory prediction and generation framework. The model is capable of generating trajectories in a single step, outperforming prior generative modeling work by learning a time-varying vector field to sample trajectories.

of generative models, demonstrating superior performance in sample quality and diversity across various domains. However, a key limitation of diffusion models is their slow sampling speed, which hinders their real-time applicability in robotics. VAEs and GANs are fast but struggle with multimodal sample quality compared to diffusion models.

In this paper, we introduce a novel trajectory forecasting and generation framework that employs flow matching [15], a method that transforms between data distributions using a learned time-varying vector field. Our technique, named Trajectory Conditional Flow Matching (T-CFM), maintains the sample quality of diffusion models while generating samples an order of magnitude faster by circumventing the iterative sampling approach used in diffusion models.

We demonstrate the efficacy of T-CFM on three robotics tasks, shown in Figure 3. In the adversarial tracking scenario, autonomous pursuing agents predict the future trajectories of an adversarial evader. We also showcase T-CFM's performance on a real-world aircraft trajectory prediction dataset, which has important implications for the development of autonomous aerial robots. Accurate trajectory prediction enables these robots to avoid collisions and coordinate with other aircrafts. Finally, we apply T-CFM to long-horizon planning in complex 2D maze environments [9], demonstrating our model's ability to generate long-horizon plans, which is crucial for robot navigation.

We show that T-CFM outperforms state-of-the-art base-

lines, including diffusion models, in terms of predictive accuracy of generated trajectories and sample quality of generated plans. Notably, our approach can generate high-quality trajectory samples with as few as one sampling step, leading to significant speed-ups compared to diffusion models, without sacrificing performance.

**Contributions:** Our key contributions are three-fold.

- We propose T-CFM, a novel flow matching framework for conditional trajectory forecasting and generation that is both accurate and efficient. To the best of our knowledge, we are the first to apply flow matching to trajectory prediction and trajectory planning tasks.
- We demonstrate state-of-the-art performance on three challenging robotics tasks: adversarial tracking, aircraft trajectory forecasting, and long-horizon planning, achieving up to in 35% increase in prediction accuracy and 142% in planning performance.
- T-CFM achieves significant sampling time speed-ups compared to prior generative modeling approaches, reducing sampling time by up to 100×. Our framework is versatile and can generate high-quality trajectories using as few as one sampling step or multi-step sampling when needed.

## II. Related Works

Traditionally, trajectory forecasting and trajectory planning have occupied very different and distinct avenues of robotics research. However, with the advent of deep learning, these tasks have become increasingly intertwined. In this section, we review traditional methods for both trajectory forecasting and generation (Section II-A). Then we describe why learning-based approaches can address the key limitations of traditional methods and introduce flow matching and generative modeling.

### A. Symbolic Methods for Trajectory Forecasting & Planning

*a) Target Tracking:* Target tracking is a well-studied problem in the robotics community [2], with numerous applications, including surveillance [10], crowd monitoring [22], and wildlife monitoring [8]. Traditional methods for target tracking and trajectory prediction, such as Kalman Filters [5], [14] and Particle Filters [7], [17], have been widely used in various in applications like autonomous navigation, object tracking in video surveillance, robotics, and radar systems. However, their performance degrades when faced with sparse observations, lack of accurate target behavior models, and long prediction horizons [18]. In our work, we address these limitations by leveraging a flow matching-based approach that learns to model the target's behavior from data, allowing for accurate predictions even in sparse observation settings and over long horizons.

*b) Planning and Navigation:* Traditional methods for path planning, such as RRT* and PRM* [13], are widely used in environments with a known representation of the world. However, they can be computationally expensive for large state spaces and require a priori knowledge, which may not always be available or can change dynamically. Learning-based approaches, such as ours, offer the promise of generalization and the ability to provide solutions in dynamic environments. Our approach learns to generate feasible trajectories directly from data, eliminating the need for explicit maps and enabling fast planning in complex, dynamic environments.

### B. Learning-Based Approaches

*a) Supervised Learning Methods for Trajectory Prediction:* Recent works in trajectory prediction for various domains, such as aircraft navigation (FlightBERT) [11], social navigation [4], and autonomous driving [19], have employed supervised learning methods. These approaches often utilize autoregressive models and log-likelihood based training to learn predictive models from data. Graph-based Adversarial Modeling with Mutual Information (GrAMMI) [27] is a recent framework that explicitly models a multimodal distribution using a combination of a Gaussian Mixture Model regularized by mutual information. While effective in certain scenarios, these models can struggle with capturing long-horizon multi-modal distributions, which are common in real-world trajectory data. We leverage a generative modeling approach rather than a discriminative one, enabling more diverse multimodal outputs for accurate prediction in complex real-world scenarios.

*b) Generative Modeling:* Generative modeling techniques, which have shown great success in computer vision tasks [21], provide a promising avenue to augment trajectory prediction by learning to model complex, multimodal distributions. Recently, diffusion-based probabilistic models [21] have dominated many generative modeling tasks. Diffusion models generate samples by iteratively denoising a Gaussian distribution, allowing them to capture complex, multimodal distributions. Diffuser [12] and Motion Planning Diffusion [3], are diffusion based approaches for learning trajectories, representing the state of the art in long horizon planning and offline reinforcement learning. Constrained Agent-based Diffusion for Enhanced Multi-Target Tracking (CADENCE) [26], similarly extends the Diffuser framework for Adversarial Tracking and modeling multi-agent behaviors. The main drawback of diffusion models is their iterative denoising process, which can be computationally slow, limiting their real-time applicability.

As an efficient alternative to diffusion models, flow matching techniques [15], [23] learn a generative model using ordinary differential equations (ODEs) instead of stochastic differential equations (SDEs). This formulation allows for faster sampling while maintaining the ability to model complex distributions. To the best of our knowledge, our work is the first to apply flow matching techniques for learning trajectories for prediction and planning tasks.

## III. Preliminaries

Trajectories play a crucial role in various robotics domains as they represent the behavior and evolution of an agent over time. Formally, we define a trajectory $\tau$ as a sequence of states $s^1, s^2, \ldots, s^T$, where $s^t \in \mathcal{S}$ represents the state of

the agent at time horizon step $t$, and $\mathcal{S}$ is the state space. In some cases, trajectories may also include actions, represented as $\tau = \{(s^1, a^1), (s^2, a^2), \ldots, (s^T, a^T)\}$, where $a^t \in \mathcal{A}$ is the action taken by the agent at time step $t$, and $\mathcal{A}$ is the action space.

By modeling trajectories in this general form, we can develop methods for trajectory forecasting and planning that are applicable across different domains. In trajectory forecasting, the goal is to predict an agent's future trajectory given its past states and additional context information, $c$. This can be formalized as learning a conditional distribution $p_\theta(\tau^{t+1:T}|\tau^{1:t}, c)$, where $\tau^{1:t}$ represents the observed trajectory up to time $t$, $\tau^{t+1:T}$ represents the future trajectory to be predicted, and $c$ represents any additional context. Similarly, in adversarial tracking, we can use the same framework to predict an adversary's $(\tau)$ future trajectory given prior detection or environmental information, $c$.

In long-horizon planning, the goal is to generate a trajectory that leads an agent from an initial state to a goal state. This can be formalized as drawing a trajectory sample from $p_\theta(\tau|c)$, where the context information, $c$ are the start and goal states. Rather than a trajectory optimization problem, this formulation allows us to view the problem as a conditional generation task, where we sample trajectories from a learned distribution conditioned on the desired start and goal states.

## IV. Method

In this section, we describe our flow matching formulation and how we model our problem. The goal primary goal is to generate trajectories $\tau$ given the conditioning factor $c$. Trajectories are defined simply as a sequence of states or a sequence of states and actions.

### A. Flow Matching Formulation

The goal of flow matching, similar to diffusion models, is to learn a process that can generate samples through an iterative process that lies in the data distribution. To do this, we model the starting random Gaussian noise distribution as $q(\tau_0)$ and the trajectory data distribution as $q(\tau_1)$. We refer to these distributions as $q_0, q_1$ where the generative modeling task is to transform $q_0$ to $q_1$.

To learn a model that can transform $q_0$ to $q_1$, we model a time-varying vector field $u : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ and a probability path $p : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^{d+}$. The vector field $u$ is defined by an ordinary differential equation (Equation 1).

$$d\tau = u_t(\tau)dt \tag{1}$$

Intuitively, the vector field defines the direction and magnitude to push each sample such that a sample from $q_0$ arrives at its corresponding location in $q_1$ by following the probability path $p$ over time.

We aim to approximate the true vector field $u$ using a neural network represented by $v_\theta(t, \tau)$, where $v_\theta(t, \tau)$ defines a time-dependent vector field parameterized by weights, $\theta$. The flow matching objective is to minimize the difference between the predicted vector field $v_\theta(t, \tau)$ and the true vector field $u_t(\tau)$, as expressed in Equation 2.

$$\min_\theta \mathbb{E}_{t,\tau\sim p_t(\tau)} \left\| v_\theta(t, \tau) - u_t(\tau) \right\|^2 \tag{2}$$

However, this objective is intractable as there is no closed form representation for the true vector field $u_t(\tau)$. Instead, prior work [1], [15], [23] has proposed to estimate the conditional form of the vector field $u_t(\tau|z)$ which is conditioned on a random variable $z$. In our work, we use the formulation where $q(z) = q(\tau_0)q(\tau_1)$, meaning $z$ captures the starting and ending points of the trajectory.

We assume a Gaussian flow between $\tau_0$ and $\tau_1$ with standard deviation $\sigma$ and model the probability path $p_t(\tau|z)$ and vector field $u_t(\tau|z)$ as shown in Equations 3. Equation 3a defines the probability path as a Gaussian distribution centered at a linear interpolation between $\tau_0$ and $\tau_1$ at time $t$. In the top portion of Figure 2, we show a visualization of the linear interpolation used to generate intermediate trajectories between $\tau_0$ and $\tau_1$. Equation 3b defines the target vector field simply as the difference vector pointing from the starting point $\tau_0$ to the end point $\tau_1$.

$$p_t(\tau|z) = \mathcal{N}\left(\tau|t\tau_1 + (1-t)\tau_0, \sigma^2\right) \tag{3a}$$

$$u_t(\tau|z) = \tau_1 - \tau_0. \tag{3b}$$

With this formulation, we now have a computable target vector field that we can regress our neural network to. Algorithm 1 summarizes the training steps:

1) Draw a starting trajectory $\tau_0$ from the Gaussian noise distribution $q(\tau_0)$ and a random timestep $t$ from a uniform distribution (Lines 2 - 3).
2) Draw a ground truth end trajectory $\tau_1$ and conditioning factor $c$ from the dataset (Line 4).
3) Compute the intermediate trajectory $\tau$ at time $t$ by linearly interpolating between $\tau_0$ and $\tau_1$ (Line 5).
4) Match the vector field $v_\theta(t, \tau)$ predicted by the neural network to the target vector field $u_t(\tau|z) = \tau_1 - \tau_0$ (Lines 6 - 8).

By repeating these steps and updating the neural network weights to minimize the difference between the predicted and target vector fields, the model learns to approximate the true time-dependent vector field that transforms samples from the starting noise distribution to the data distribution.

The neural network model used to parameterize $v_\theta(t, \tau, c)$ is a 1D Convolutional Temporal U-Net based on prior diffusion work [12], [26]. 1D convolutions slide over the time dimension of the input trajectory $\tau$, capturing temporal patterns and dependencies without being autoregressive. This allows for efficient parallel processing of the entire trajectory. The model also incorporates Feature-Wise Linear Modulation (FiLM) Layers [16] to condition the model with relevant context information, $c$. For each domain, we will describe the context vector ($c$) used (Section V). By using the same base architecture as prior diffusion work, we demonstrate that our training methodology generates better models irrespective of model parameter count and architecture.

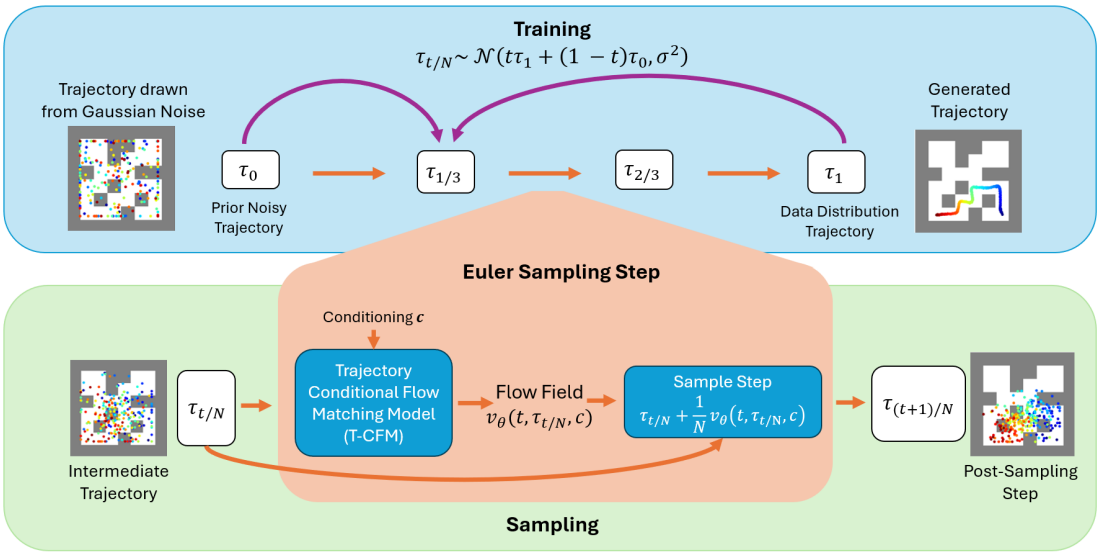



Fig. 2: Overview of Trajectory Conditional Flow Matching. The flow matching formulation defines intermediate trajectories as a linear combination between the prior noise distribution ($\tau_0$) and data distribution ($\tau_1$). The sampling procedure then utilizes the learned flow field generated by the model to create samples.

**Algorithm 1** Conditional Flow Matching Training

**Require:** Dataset $\mathcal{D}$, computable $u_t(x|z)$ and network $v_\theta(t, \tau, c)$.

1: **while** Training **do**
2: $\quad \tau_0 \sim q(\tau_0)$; ▷ Draw source trajectory
3: $\quad t \sim U(0, 1)$ ▷ Draw timestep
4: $\quad \tau_1, c \sim \mathcal{D}$ ▷ Draw target trajectory and conditioning
5: $\quad \tau \sim p_t(\tau|z) = \mathcal{N}\left(\tau|t\tau_1 + (1-t)\tau_0, \sigma^2\right)$ ▷ Eq. 3a
6: $\quad u_t(\tau|z) = \tau_1 - \tau_0$ ▷ Eq. 3b
7: $\quad \mathcal{L}_{CFM}(\theta) = ||v_\theta(t, \tau, c) - u_t(\tau|z)||^2$ ▷ Loss
8: $\quad \theta = \theta + \alpha \nabla_\theta \mathcal{L}_{CFM}(\theta)$ ▷ Update Model
9: **end while**

### B. Sampling

Given a trained flow model $v_\theta(t, \tau, c)$, the sampling procedure utilizes an ODE solver to recover the solution to Equation 1. We can denote the solution of the ODE with $\phi_t(\tau)$, where $\phi_0(\tau) = \tau$ and $\phi_t(\tau)$ is the transformation of our trajectory $\tau$ transported along the vector field from time 0 to time $t$. In Algorithm 2, we show the sampling procedure using the Euler method (Line 7) but any off the shelf ODE solver can be used. In our experiments, we choose to use the Euler sampling method as the number of sampling steps is easily adjustable. The bottom portion of Figure 2 shows how the sampling procedure moves from a prior noisy trajectory $\tau_0$ to a trajectory that lies within the data distribution $\tau_1$.

One key difference between the trajectory generation and planning tasks is the planning task requires constraints on the sampled trajectory. We provide a formulation to constrain the generated trajectory $\tau$ to start at the current robot state and end at the desired goal state. For each sampling step, we set these states of the trajectory rather than interpolate from noise (Alg 2, Line 4-5), where the horizon of the trajectory is denoted $h$. In this formulation, we allow the model to infill the trajectory to generate a cohesive plan.

**Algorithm 2** Euler Sampling

**Require:** Samplable $q(\tau_0) = \mathcal{N}(0, I)$, trained flow network $v_\theta(t, \tau, c)$, number of sampling timesteps $N$
**Optional**: Start State $s^0$, End State $s^h$

1: $\tau_0 \sim q(\tau_0)$ ▷ Sample $\tau_0$
2: **for** $t = 1, ..., N$ **do**
3: $\quad$ **if** Planning **then**
4: $\quad\quad \tau^0_{t/N} \leftarrow s^0$ ▷ Set Start State
5: $\quad\quad \tau^h_{t/N} \leftarrow s^h$ ▷ Set Goal State
6: $\quad$ **end if**
7: $\quad \tau_{(t+1)/N} \leftarrow \tau_{t/N} + \frac{1}{N} v_\theta(t, \tau_{t/N}, c)$
8: **end for**

The key feature of parameterizing the probability flow and vector field through Equation 3 is that it enables our model to learn *straight* flows as compared to diffusion models. In diffusion models, the sampling process involves gradually denoising a Gaussian noise sample over many steps, following a complex path in the data space. This typically requires a large number of sampling steps to generate high-quality samples. In contrast, our flow matching approach learns a direct, straight path from the starting noise distribution to the target data distribution by modeling the probability path as a linear interpolation between the starting and ending points (Equation 3), encouraging the model to find the most efficient trajectory that matches the true data distribution.

Intuitively, the straight flows learned by our model can be thought of as a shortcut from the noise distribution to the data distribution. Instead of taking a meandering path through the

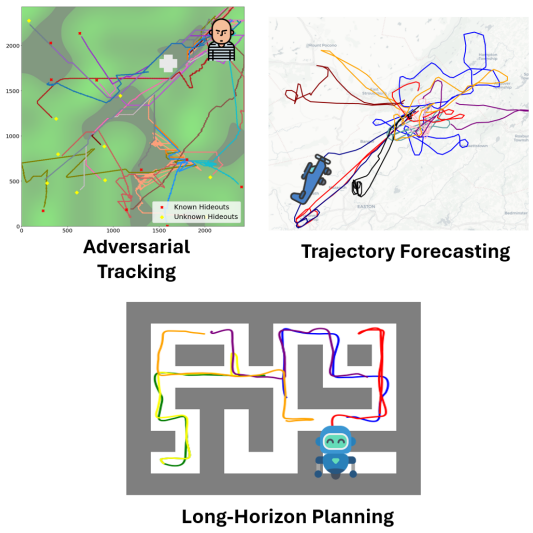



Fig. 3: Trajectory Forecasting and Planning Domains: Our T-CFM framework is applicable many trajectory modeling tasks, with Adversarial Tracking, Trajectory Forecasting, and Long-Horizon Planning domains shown here.

data space, the model learns to follow a direct route guided by the target vector field, enabling our method to reduce the number of intermediate steps needed to generate high-quality trajectories. This straight path allows for faster sampling with fewer steps.

## V. Evaluation and Domains

We test our model in three different tasks and domains: 1) Adversarial Tracking 2) Trajectory Forecasting, and 3) Long-Horizon Planning. These domains test our model's capability of generating accurate multimodal trajectory predictions and plans for robots to use. Visualizations for the training data and domains are shown in Figure 3 and a summary of the states and context vectors used are in Table I.

| Domain | State in Trajectory ($\tau$) | Context Vector ($c$) |
|---|---|---|
| Adversarial Tracking | Position (x, y) coordinates | Historical detection information |
| Aircraft Trajectory | Longitude, Latitude, Altitude | 5-minute history of past states |
| Long-Horizon Planning | Position (x, y) coordinates | Start and goal states |

TABLE I: Summary of State and Context Vectors for Different Domains

### A. Adversarial Tracking

Adversarial tracking aims to predict an adversary's future trajectory $\tau$ given past historical information, $c$. These domains are challenging due to the adversary's potential multiple strategies and the observers' often incomplete or sparse data. We assess our flow-matching tracking models using the Prison Escape scenarios, as introduced in previous work [27].

The Prisoner Escape and Narco Traffic Interdiction simulations share similar pursuit-evasion dynamics, with tracking agents collaborating to locate and apprehend an adversary attempting to reach predetermined hideouts. The agents face the challenge of operating in large environments with sparse detections of the opponent. Key differences between the domains include the type of fog-of-war, agent capturing dynamics, and destination types. We refer the reader to prior work for more details [27].

For both scenarios, we utilize open-sourced datasets from prior work [27]. The Prison Escape scenario consists of three datasets (Prisoner-Low, Prisoner-Medium, Prisoner-High) with opponent detection rates of $12.9\%$, $44.0\%$, and $63.1\%$, respectively. The Narco Interdiction scenario uses two datasets with opponent detection rates of $13.8\%$ and $31.5\%$, adjusted by modifying the pursuit agents' detection radius. We evaluate our models using Average Displacement Error (ADE), which computes the average $l_2$ distance between each sampled trajectory and the ground truth trajectory over all timesteps.

### B. Aircraft Trajectory Forecasting

To demonstrate the capabilities of our model on real data, we retrieved two years of data for a single Cessna aircraft from the OpenSky database [20]. Individual trajectories were extracted from the dataset, resulting in a total of 474 trajectories and a train/val/test split of 80/10/10% was used. The Cessna was chosen to constrain the range of trajectories while maintaining significant variability, allowing for testing the multimodal performance of our algorithm. The goal is to predict the future trajectory $\tau$ given the 5-minute history of past states $c$, which we use as our context vector. We evaluate the forecasting performance using two common metrics: mean absolute error (MAE) and root mean square error (RMSE) for longitude, latitude, and altitude.

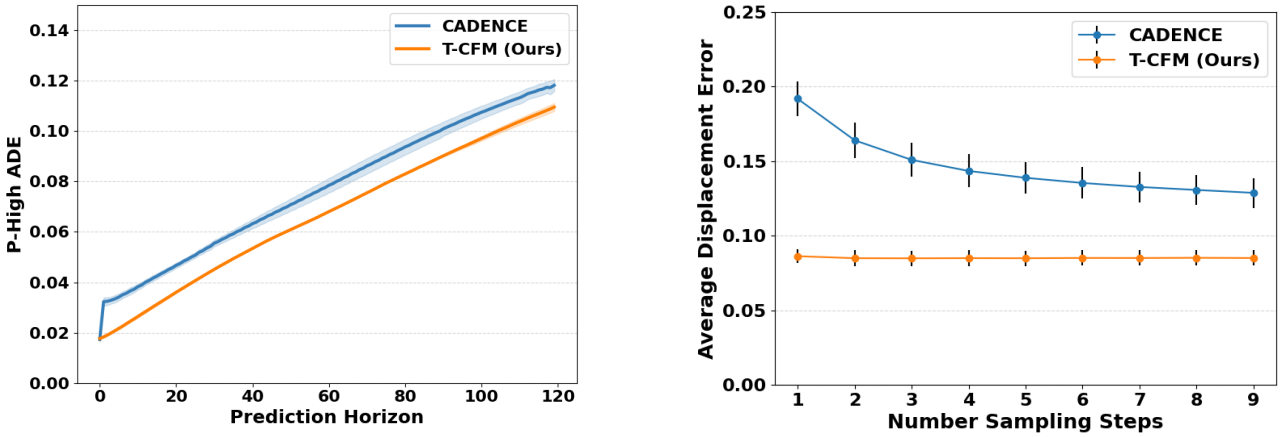
### C. Long-Horizon Imitation Learning - Maze2D

Learning to plan for long horizons is crucial for robots to navigate autonomously in complex domains. The performance of our models in long-horizon planning is evaluated using the Maze2D environments [9]. In this task, the agent must traverse from a starting location to a goal location. The algorithm is tested on three maps of increasing difficulty: U-Maze, Medium, and Large. Following prior work, the performance is reported in terms of *score*, which represents the agent's success in reaching the final goal. The score is normalized between 0 and 100 based on an expert policy.

Two different evaluations are performed: single-task and multi-task. In the single-task evaluation, the goal location remains constant while in the multi-task setting, the goal location is randomly selected at the beginning of each episode. Training data consists of successful trajectories between randomly selected start and end goals.

## VI. Results and Discussions

This section presents results and analysis for three tasks: Adversarial Tracking, Trajectory Forecasting, and Long-Horizon Planning. Three models were trained for each task using different random seeds.




(a) ADE vs Prediction Time-Horizon

(b) ADE vs Total Sampling Steps

Fig. 4: Comparison our model with the diffusion-based CADENCE model. Our method achieves better ADE on the entire prediction horizon (left) while also maintaining performance when the number of sampling steps is lowered (right).

### A. *Adversarial Tracking*

Our approach is benchmarked against several state-estimation baselines including 1) VRNN [6], 2) GRaMMI [27] and 3) CADENCE [26].

| | | Prediction Horizon | | | | |
|---|---|---|---|---|---|---|
| | | 0 min | 30 min | 60 min | 90 min | 120 min |
| P-Low | Particle Filter | 0.120 | 0.148 | 0.161 | 0.171 | 0.183 |
| | VRNN | 0.106 | 0.093 | 0.119 | 0.146 | 0.177 |
| | GrAMMI | 0.060 | 0.080 | 0.110 | 0.154 | 0.163 |
| | CADENCE | 0.057 | 0.077 | **0.100** | **0.127** | 0.154 |
| | T-CFM (Ours) | **0.055** | **0.076** | 0.101 | 0.128 | **0.153** |
| P-Med | Particle Filter | 0.099 | 0.129 | 0.141 | 0.152 | 0.163 |
| | VRNN | 0.172 | 0.086 | 0.110 | 0.144 | 0.167 |
| | GrAMMI | 0.049 | 0.077 | 0.110 | 0.146 | 0.167 |
| | CADENCE | 0.046 | 0.076 | 0.103 | 0.129 | 0.153 |
| | T-CFM (Ours) | **0.030** | **0.058** | **0.088** | **0.118** | **0.146** |
| P-High | Particle Filter | 0.041 | 0.084 | 0.102 | 0.119 | 0.133 |
| | VRNN | 0.105 | 0.059 | 0.100 | 0.117 | 0.145 |
| | GrAMMI | **0.015** | 0.056 | 0.092 | 0.122 | 0.162 |
| | CADENCE | 0.017 | 0.054 | 0.078 | 0.099 | 0.118 |
| | T-CFM (Ours) | 0.018 | **0.044** | **0.067** | **0.089** | **0.110** |

TABLE II: Average Displacement Error Results for three Prisoner Escape (P-low, P-med, P-high) Datasets. Bolded values represent the best performing model.

*1) Tracking Capabilities of Flow Matching:* We report our results on the three Prisoner Escape datasets in Table II. We show that T-CFM outperform or matches the prior baselines on all prediction horizons with the greatest advantages on the Prisoner-Medium and Prisoner-High datasets, showcasing a 17% and 12% increase in predictive accuracy respectively. We hypothesize that the flow-matching models are able to better incorporate the dense detection history information than the diffusion models because the flow field is deterministic and does not include adding an additional noise component. This may benefit the flow matching models to generate more confident and correct trajectories as compared to the diffusion models.

We also show the ADE for the entire prediction horizon on the Prisoner-Medium dataset in Figure 4a. The VRNN and GRaMMI models are not shown as they do not predict full trajectories and also are not as competitive as the diffusion baseline. We find that our flow matching model reduces the ADE over all time horizons and has a tighter standard deviation than the diffusion model. Furthermore, we observe a performance dip in CADENCE between the first and second prediction timesteps, characterized by the sudden increase in ADE. This occurs because CADENCE employs an inpainting formulation that sets the first timestep to the detected location, if available. Consequently, this formulation introduces a risk of discontinuities in the diffusion tracks. Our results show that our flow matching model does not encounter the same issue and can outperform the diffusion-based model even without an explicit inpainting formulation.

*2) Sampling Speed Analysis:* We analyze the accuracy of our model compared to the diffusion model by reducing the total number of sampling steps $N$. In diffusion models, sampling steps refer to denoising steps, while in our method, they refer to Euler sampling steps. Both formulations require a neural network function call at each sampling step, making the number of sampling steps the primary bottleneck in reducing overall sampling time as the underlying neural network architecture is the same.

We find that our flow-based model can generate high quality samples with just a single sampling step (Figure 4b). This is due to the difference between flow matching and diffusion objectives. Flow matching enforces a straight probability flow between the starting distribution $q(\tau_0)$ and the ending distribution $q(\tau_1)$. Consequently, while diffusion models may need to adjust the sample direction during denoising, the flow matching framework learns a good initial estimate of how to move samples from the noisy distribution, enabling sample generation without multiple steps.

| | Lon MAE | | | Lat MAE | | | Alt MAE | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 15 | 30 | 0 | 15 | 30 | 0 | 15 | 30 |
| FlightBERT | 0.036 | 0.127 | 0.164 | 0.024 | 0.087 | 0.102 | 390.1 | 1060.8 | 1014.1 |
| T-CFM (Ours) | **0.010** | **0.098** | **0.130** | **0.006** | **0.067** | **0.075** | **145.3** | **853.3** | **782.6** |
| | Lon RMSE | | | Lat RMSE | | | Alt RMSE | | |
| | 0 | 15 | 30 | 0 | 15 | 30 | 0 | 15 | 30 |
| FlightBERT | 0.057 | 0.188 | 0.267 | 0.035 | 0.122 | 0.161 | 509.9 | 1430.6 | 1375.1 |
| T-CFM (Ours) | **0.014** | **0.160** | **0.234** | **0.009** | **0.107** | **0.141** | **205.5** | **1242.6** | **1135.4** |

TABLE III: Aircraft Trajectory Forecasting: T-CFM achieves lower MAE and RMSE on Latitude, Longitude, and Altitude.

### *B. Aircraft Trajectory Forecasting*

The trajectory forecasting task tests our model's generative capabilities on real-world data rather than simulated data. We compare our method against FlightBERT [11], a modern transformer-based framework built specifically for aircraft trajectory forecasting. We modify FlightBERT's attention mechanism, as the original framework assumed access to aircraft velocities. We also train with a negative log-likelihood loss to better model the variance in our dataset.

Table III shows our model's performance as compared to FlightBERT. We find that our method outperforms FlightBERT on all metrics with an average improvement of $35.4\%$ over all metrics. We hypothesize two main reasons that our method outperforms FlightBERT. First, our method is not autoregressive generates the whole trajectory at once. This provides an advantage as errors may not accumulate over the prediction time horizon. FlightBERT was only tested for shorter horizon predictions. Meanwhile, we are interested in longer horizon predictions of up to 30 minutes compared to the 5 minute horizon for the dataset in FlightBERT. Second, we hypothesize that the multimodal capabilities of our model is important for our flight trajectory dataset. Unlike commercial flights, the behavior of the Cessna aircraft does not travel in straight paths and consists of multiple heading changes throughout its path. We show that T-CFM better models these diverse trajectories than prior work.

### *C. Long-Horizon Planning through Imitation Learning*

We compare how well our models perform against Diffuser [12], the current state-of-the-art method for learning how to plan solely from data on the Maze2D task (Table IV). We find that with just a single sampling step, our method significantly outperforms the diffusion models, achieving a 142% increase in score. Similar to our analysis in Adversarial Tracking, the linear probability flows allows us to immediately infer high quality samples. Additionally, we provide a visualization of the sampling procedure with just two sampling timesteps in Figure 5. Here we show that Diffuser produces a plan that intersects with the wall, as it is requires a large number of sampling steps to produce coherent plans. Meanwhile our method plans a collision free path. Additionally, the middle trajectory $\tau_{1/2}$ shows more coherence and has less spread in T-CFM than Diffuser. This supports the hypothesis that our T-CFM's flow field is more efficient at transforming noisy trajectories into realistic ones.

While T-CFM outperforms Diffuser in the U-Maze and

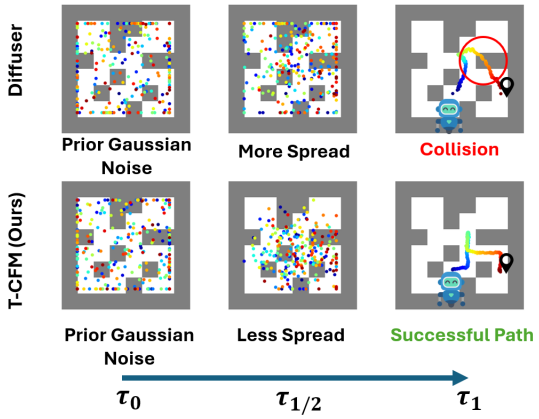



Fig. 5: Visualization of Sampling Procedure between Diffuser (top) and T-CFM (bottom) in Maze2D-Medium. In just two sampling steps, we show that T-CFM can successfully plan a path between the start and end unlike Diffuser.

Medium mazes with more sampling steps, Diffuser outperforms our model in the Large maze. T-CFM occasionally generates good plans but sometimes produce paths that collide with walls, lowering the overall score. Sampling a single trajectory from noise increases the chance of generating inaccessible plans compared to Diffuser. This may be because linear flows from flow matching struggle to correct certain noise initializations. In contrast, the diffusion model's sampling procedure is less dependent on initial noise, allowing it to reason better in the larger domain.

| Environment | N=1 | | N=256 | |
|---|---|---|---|---|
| | Diffuser | T-CFM (Ours) | Diffuser | T-CFM (Ours) |
| Maze2D U-Maze | 50.7$\pm$6.7 | **106.7**$\pm$2.7 | 112.5$\pm$11.2 | **122.1**$\pm$1.4 |
| Maze2D Medium | 21.7$\pm$13.5 | **112.2**$\pm$1.5 | 123.3$\pm$1.6 | **123.8**$\pm$3.5 |
| Maze2D Large | 30.3$\pm$6.9 | **111.0**$\pm$13.5 | **112.6**$\pm$16.3 | 104.3$\pm$3.4 |
| Single-Task Average | 34.2 | **109.9** | 116.1 | **116.7** |
| Multi2D U-Maze | 69.8$\pm$16.0 | **129.8**$\pm$3.0 | 127.3$\pm$3.3 | **129.5**$\pm$0.9 |
| Multi2D Medium | 58.4$\pm$5.3 | **116.5**$\pm$2.8 | 124.2$\pm$1.2 | **126.5**$\pm$4.1 |
| Multi2D Large | 35.8$\pm$4.2 | **121.7**$\pm$5.0 | **138.7**$\pm$5.9 | 127.3$\pm$8.6 |
| Multi-Task Average | 54.7 | **122.7** | **130.1** | 127.8 |

TABLE IV: The performance of T-CFM and Diffuser on the long-horizon Maze2D compared when given a single sampling timestep $N = 1$ and maximum sampling timesteps, $N = 256$. Our model (T-CFM) is able to drastically reduce the number of sampling steps required to generate feasible plans whereas the Diffuser model fails at $N = 1$.

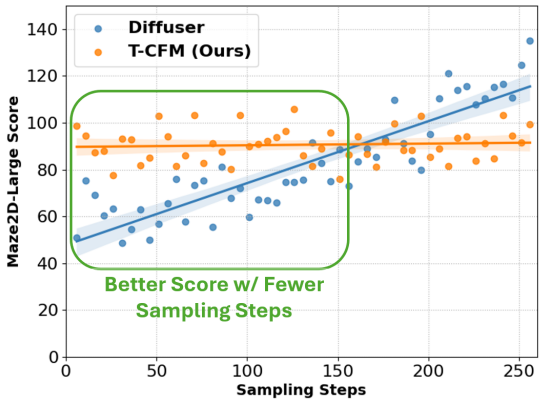



Fig. 6: Compared to Diffuser, our T-CFM model does not drop in performance as we reduce the number of sampling steps on the Large Maze2D domain.

## VII. Limitations and Future Work

Our current approach does not explicitly consider multi-agent interactions. Future work includes extending to social navigation and autonomous driving scenarios, which require reasoning about other agents. We aim to increase our flow matching models' expressiveness, potentially incorporating stochastic bridge matching [24] to combine deterministic ODE and stochastic SDE formulations. Further experiments with T-CFM in dynamic situations will better demonstrate its capabilities for real-world robotic tasks.

## VIII. Conclusion

T-CFM is a novel approach for efficient trajectory forecasting and planning in robotics. By learning time-varying vector fields through flow matching, T-CFM achieves state-of-the-art performance on tasks like adversarial tracking, aircraft trajectory prediction, and long-horizon planning. T-CFM offers significant speed-ups compared to diffusion-based models without compromising accuracy, paving the way for more autonomous and responsive robots operating in complex, dynamic environments.

## References


[1] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

[2] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[3] J. Carvalho, A.T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[4] Yuhang Che, Allison M. Okamura, and Dorsa Sadigh. Efficient and trustworthy social navigation via explicit and implicit robot–human communication. *IEEE Transactions on Robotics*, 36(3):692–707, 2020.

[5] Rong Chen and Jun S Liu. Mixture kalman filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(3):493–508, 2000.

[6] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

[7] Petar M Djuric, Mahesh Vemula, and Mónica F Bugallo. Target tracking by particle filtering in binary sensor networks. *IEEE Transactions on signal processing*, 56(6):2229–2238, 2008.

[8] Matthew Dunbabin and Lino Marques. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1):24–39, 2012.

[9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[10] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas. Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3):16–25, 2006.

[11] Dongyue Guo, Edmond Q. Wu, Yuankai Wu, Jianwei Zhang, Rob Law, and Yi Lin. Flightbert: Binary encoding representation for flight trajectory prediction. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):1828–1842, 2023.

[12] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[13] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[14] William F Leven and Aaron D Lanterman. Unscented kalman filters for multiple target tracking with symmetric measurement equations. *IEEE Transactions on Automatic Control*, 54(2):370–375, 2009.

[15] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[17] G Mallikarjuna Rao and Ch Satyanarayana. Visual object target tracking using particle filter: a survey. *International Journal of Image, Graphics and Signal Processing*, 5(6):1250, 2013.

[18] X. Rong Li and V.P. Jilkov. Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1333–1364, 2003.

[19] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[20] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. Bringing up opensky: A large-scale ads-b sensor network for research. In *Proceedings of the 13th IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 83–94, April 2014.

[21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[22] Pratap Tokekar, Volkan Isler, and Antonio Franchi. Multi-target visual tracking with aerial robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3067–3072.

[23] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

[24] Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching. *arXiv preprint 2307.03672*, 2023.

[25] Zixuan Wu, Sean Ye, Manisha Natarajan, Letian Chen, Rohan Paleja, and Matthew C Gombolay. Adversarial search and tracking with multiagent reinforcement learning in sparsely observable environment. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 43–49. IEEE, 2023.

[26] Sean Ye, Manisha Natarajan, Zixuan Wu, and Matthew C. Gombolay. Diffusion models for multi-target adversarial tracking. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 142–148, 2023.

[27] Sean Ye, Manisha Natarajan, Zixuan Wu, Rohan Paleja, Letian Chen, and Matthew C Gombolay. Learning models of adversarial agent behavior under partial observability. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3688–3695. IEEE, 2023.