THINK ON VISUAL TRAJECTORIES: TRANSFERABLE TRAJECTORY RECOVERY VIA DRIFTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recovering dense trajectories from sparse observations supports urban mobility analysis, commercial decision-making, and regional planning. However, reliance on city-specific road-segment vocabularies limits the transferability of many existing methods. Inspired by how human experts reconstruct journeys on maps, we investigate visual maps as a shared language for cross-city trajectory recovery: observed locations are rendered on a map, and missing movement is recovered through video generation. We construct a dataset pairing these map-rendered sparse observations with videos of the corresponding dense trajectories. Building on this formulation, we propose DriftTraj, a framework that adapts Drifting Models to map-conditioned trajectory recovery. It produces each video block in a single forward pass, avoiding iterative diffusion sampling, and recovers longer trajectories through successive blocks. Because trajectory markers occupy only a small fraction of each frame, we introduce a differentiable red-marker extractor and incorporate its spatial features into a geometric training objective, providing explicit supervision for marker location and spatial structure. We evaluate DriftTraj on Porto, Beijing, Xi'an, and Chengdu, training the generator only on Porto and transferring it to the other three cities with zero-shot transfer and limited-data fine-tuning. The results support visual maps as a shared representation for cross-city trajectory recovery and highlight their potential for transferable mobility modeling.

1 INTRODUCTION

Trajectory data provide a detailed view of how people and vehicles move. Such information supports a wide range of applications, including urban region profiling, retail site selection, and public transit route planning. In practice, however, trajectories are often observed only at a limited number of locations or timestamps. Sampling constraints, energy budgets, and gaps in data collection can leave substantial portions of a journey unobserved. *Trajectory recovery* addresses this problem by reconstructing a dense trajectory from sparse observations, ideally preserving both the observed locations and the spatial and temporal consistency of the underlying movement (Chen et al., 2023b).

A prominent line of research approaches trajectory recovery through joint modeling of trajectories and road networks, using graph neural networks to encode road structure and movement patterns (Chen et al., 2023b; Wei et al., 2024). These representations provide valuable geographic constraints. However, when they are tied to a city's road-segment vocabulary and graph structure, direct transfer to another city becomes challenging: the target network contains previously unseen segments and different connectivity patterns. This motivates recovery models whose input representation can be shared across cities without relying on a fixed vocabulary of road-segment identifiers.

We therefore ask a natural question: *can trajectory recovery benefit from the way people reason about paths on a visual map?* Experts usually reconstruct trajectories by placing their observed locations on a map and tracing plausible connections between them. The relevant information is conveyed through road geometry and spatial relationships, rather than arbitrary segment identifiers. Although the layout changes in an unfamiliar city, the visual language of the map remains recognizable. This suggests that learning to complete trajectories in map space could provide a transferable alternative to city-specific symbolic representations. Figure 1 illustrates this contrast.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This perspective connects trajectory recovery to visual generative modeling. Recent work shows that video models exhibit early zero-shot visual reasoning abilities, including solving spatial tasks from visual inputs (Wiedemer et al., 2025). Inspired by these findings, we aim to train a compact video generator from scratch for trajectory recovery. We construct a trajectory recovery dataset by overlaying sparse observations on map tiles and representing the recovered movement as a video. The learning problem is thus to generate a visual continuation that connects the observations within their geographic context.

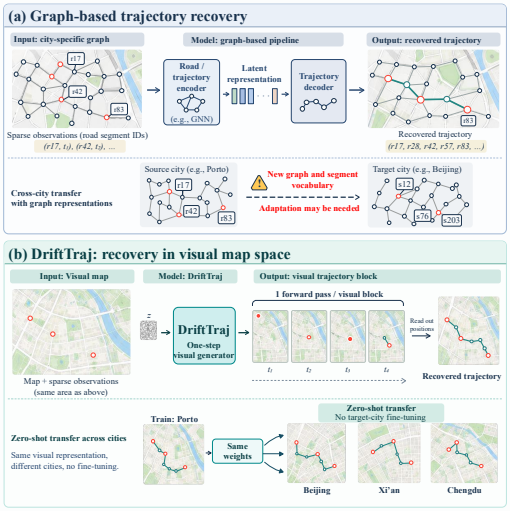


Figure 1: Conventional trajectory recovery versus DriftTraj's visual generation approach.

Making this formulation practical presents two challenges. First, large video generators are computationally expensive, particularly when generation requires iterative diffusion sampling. At the measured 13.204 seconds per video, processing 10,000 trajectories sequentially would take approximately 36.68 hours. Second, visual fidelity does not necessarily imply accurate trajectory recovery: the moving marker occupies only a small fraction of each frame, while the static map dominates the image. A lightweight model trained with a generic image-level objective may therefore reproduce the background without precisely localizing the marker. Efficient visual recovery requires both fast generation and supervision that explicitly captures trajectory geometry.

To address these challenges, we propose **DriftTraj**, a geometry-aware framework for map-conditioned visual trajectory recovery. We propose *differentiable marker-aware geometric supervision*: instead of relying on discrete marker detection, we extract a soft marker response and supervise its centroid and multiscale spatial structure. We use a drifting objective (Deng et al., 2026) to train a compact generator that produces each video block in a single forward pass, without iterative denoising. For longer trajectories, *Truth Forcing* re-renders the predicted marker on the original map and anchors successive rollouts to the supplied observations.

We evaluate DriftTraj on Porto, Beijing, Xi'an, and Chengdu, pretraining the generator only on Porto and transferring it to the other three cities through zero-shot transfer and limited-data fine-tuning. After fine-tuning, Mean@5 achieves the best results among the compared non-oracle methods on most metrics across the three target cities. For the Porto-trained model, oracle selection of the lowest-ground-truth-RMSE rollout from five candidates improves all three metrics over direct coordinate averaging across all four cities. This diagnostic reveals the quality of the candidate set and the potential for better selection or aggregation, rather than deployable prediction performance. Ablations further indicate that marker-aware supervision and multi-frame prediction stabilize training. Together, these findings support visual maps as a shared representation for cross-city recovery.

In summary, our contributions are threefold:

- **Visual formulation and dataset.** We formulate trajectory recovery as map-conditioned video generation and construct a dataset pairing map-rendered sparse observations with videos of the corresponding dense trajectories. This provides a common visual representation for learning and evaluating trajectory recovery across cities.
- **Geometry-aware, one-step visual recovery.** We propose DriftTraj, which combines differentiable marker-aware geometric supervision with a compact one-step generator. This design explicitly supervises marker localization and generates each block without iterative denoising.
- **Cross-city transfer and limited-data adaptation.** We evaluate DriftTraj on Porto, Beijing, Xi'an, and Chengdu under both zero-shot and limited-data fine-tuning settings. A shared Porto-pretrained generator supports direct transfer and target-city adaptation, demonstrating the potential of visual maps for transferable trajectory recovery.

2 Related Work

2.1 Trajectory Data Mining and Recovery

Trajectory data offer valuable insights into human mobility and urban activity, but sparse observations limit how fully this information can be exploited, making trajectory recovery an important task in trajectory data mining. To extract useful patterns from such data, early neural approaches drew on learning trajectory representations, as exemplified by t2vec (Li et al., 2018). Subsequent work enriched these representations through graph-based spatial modeling, attention, and joint spatial–temporal learning, including TrajGAT (Yao et al., 2022), TrajCL (Chang et al., 2023), and ST2Vec (Fang et al., 2022). Although primarily developed for similarity learning, these representations also provide a basis for recovery when paired with a decoder.

Alongside these advances in representation learning, recovery methods have progressed from interpolation and map-matching pipelines (Newson & Krumm, 2009). Recurrent models enabled methods such as DHTR (Wang et al., 2021) and MTrajRec (Ren et al., 2021) to capture sequential dependencies beyond simple motion assumptions. Attention mechanisms subsequently broadened the context available for recovery in methods such as AttnMove (Xia et al., 2021) and TERI (Chen et al., 2023a). In parallel, graph neural networks brought mobility transitions and road structure into recovery through methods such as PeriodicMove (Sun et al., 2021), RNTrajRec (Chen et al., 2023b), and MM-STGED (Wei et al., 2024). Despite their different architectures, these methods reason over coordinates, location tokens, or explicit graphs rather than reasoning directly about trajectories visually.

2.2 Generative Modeling and Visual Reasoning

Realistic image synthesis has long been a central goal of visual generative modeling. Early deep generative approaches pursued this goal through latent-variable learning in variational autoencoders (Kingma & Welling, 2014) and adversarial training in generative adversarial networks (Goodfellow et al., 2014). Diffusion models later offered an alternative by constructing images through successive denoising steps (Ho et al., 2020), while flow matching enabled learning continuous transformations from noise to data through vector-field regression (Lipman et al., 2023). However, the repeated model evaluations used in standard diffusion and flow sampling motivate more efficient generative formulations. More recently, Drifting Models (Deng et al., 2026) have addressed this challenge by moving distribution evolution into training, using an attraction–repulsion field to guide learning while retaining a single generator forward pass at inference.

As these models have become more capable, their applications have also extended beyond realistic synthesis to visual problem solving. For example, pretrained video models can perform tasks such as maze solving without task-specific training (Wiedemer et al., 2025). Inspired by this perspective, we study task-specific training and cross-city transfer of a compact video generator.

3 Preliminaries

Drifting Models (Deng et al., 2026) learn a direct noise-to-data map. Let p denote the data distribution and $p_z = \mathcal{N}(0, I)$ a noise prior. A generator f_θ induces the pushforward distribution

$$x = f_\theta(z), \qquad z \sim p_z, \qquad q_\theta = (f_\theta)_{\#} p_z. \tag{1}$$

Parameter updates evolve q_θ toward p during training, instead of evolving noisy samples through a multi-step inference procedure.

Drifting field. Writing $q = q_\theta$, define a similarity kernel with temperature $\tau > 0$ and its normalization:

$$\begin{aligned} k_\tau(x, y) &= \exp(-\|x - y\|_2/\tau)\,, \\ Z_r(x) &= \mathbb{E}_{y\sim r}[k_\tau(x, y)], \qquad r \in \{p, q\}. \end{aligned} \tag{2}$$

The normalized population field is

$$V_{p,q}(x) = \frac{\mathbb{E}_{y\sim p}[k_\tau(x, y)(y - x)]}{Z_p(x)} - \frac{\mathbb{E}_{y\sim q}[k_\tau(x, y)(y - x)]}{Z_q(x)}. \tag{3}$$

Data samples attract x, while generated samples repel it. Antisymmetry, $V_{p,q} = -V_{q,p}$, ensures $p = q \Rightarrow V_{p,q} = 0$; the converse requires additional assumptions.

Training objective. For $x = f_\theta(z)$, training minimizes

$$\mathcal{L}_{\text{drift}}(\theta) = \mathbb{E}_{z\sim p_z}\left[\left\|x - \text{sg}(x + V_{p,q_\theta}(x))\right\|_2^2\right], \tag{4}$$

where sg stops gradients through the entire target. Gradients move the prediction toward this frozen, drifted target without differentiating through the field. Minibatches approximate the expectations.

One-step generation. At inference, a fresh z produces $x = f_\theta(z)$ with *one generator evaluation (1 NFE)*: no drifting-field computation, training reference samples, or iterative denoising is needed. Iteration belongs to training, not to sampling an output. For DriftTraj, this one-step property applies to each generated visual block; a longer trajectory can still require multiple autoregressive blocks.

4 METHODOLOGY

As illustrated in Figure 2, we divide DriftTraj into three stages: 1) Data Construction: represents movement as a marker on a map image. 2) Pretraining: trains a one-step video generator to produce trajectory videos conditioned on the previous position and the visual map. 3) Truth Forcing: builds longer candidates by repeatedly restoring the map context and enforcing the supplied observations. We will describe each stage in detail below.

4.1 DATA CONSTRUCTION

A trajectory is a time-ordered sequence $\mathcal{X} = \{(x_i, y_i, t_i)\}_{i=1}^N$, where (x_i, y_i) denotes a geographic location and t_i its timestamp. Let $\mathcal{O} = \{(x_j, y_j, t_j)\}_{j=1}^M$ denote its sparse observations, with location (x_j, y_j) observed at time t_j. To obtain a fixed-length representation for training, we linearly interpolate each trajectory at 20 uniformly spaced timestamps. After generation, postprocessing interpolates the recovered sequence back to the original timestamps $\mathcal{T} = \{t_i\}_{i=1}^N$.

To avoid a learned vocabulary of city-specific road identifiers, we overlay the interpolated trajectory on a visual map from OpenStreetMap[1], forming a video in which each frame displays the current position as a red marker. The known observations remain visible as gray markers and are connected in temporal order so that these lines indicate their order, not the missing route between them. Road geometry, observed locations, observation order, and movement are thus presented in a shared visual representation.

4.2 PRETRAINING

We formulate trajectory recovery as autoregressive visual prediction: given the previous frame c, a generator predicts F future frames as $\widehat{Y} = f_\theta(z, c)$, with $z \sim \mathcal{N}(0, I)$. We deliberately omit separate timestamp and coordinate inputs: observations remain visible on the map, while their numerical spatial and temporal constraints are handled through postprocessing.

Visual drifting. For each training pair (c, Y), we draw n independent noise samples and generate $\widehat{Y}^{(k)} = (\hat{I}_1^{(k)}, \ldots, \hat{I}_F^{(k)})$. Let $\tilde{x}_{k,f}$ be the normalized RGB vector of $\hat{I}_f^{(k)}$. The conditional field $\widehat{V}_{c,f}$ attracts it toward the recorded frame I_f and repels it from generated samples of the same condition. We optimize

$$\mathcal{L}_{\text{visual}} = \mathbb{E}_{(c,Y),z_{1:n}}\left[\frac{1}{nF}\sum_{k=1}^{n}\sum_{f=1}^{F}\left\|\tilde{x}_{k,f} - \text{sg}\Big(\tilde{x}_{k,f} + \widehat{V}_{c,f}(\tilde{x}_{k,f})\Big)\right\|_{\text{mse}}^2\right], \tag{5}$$

where $\|v\|_{\text{mse}}^2 = \|v\|_2^2/\dim(v)$. The stop-gradient operator freezes the drifted target, moving distribution matching into training rather than iterative inference. Fields and their normalizations are computed separately for each condition and future frame.

[1] https://www.openstreetmap.org/

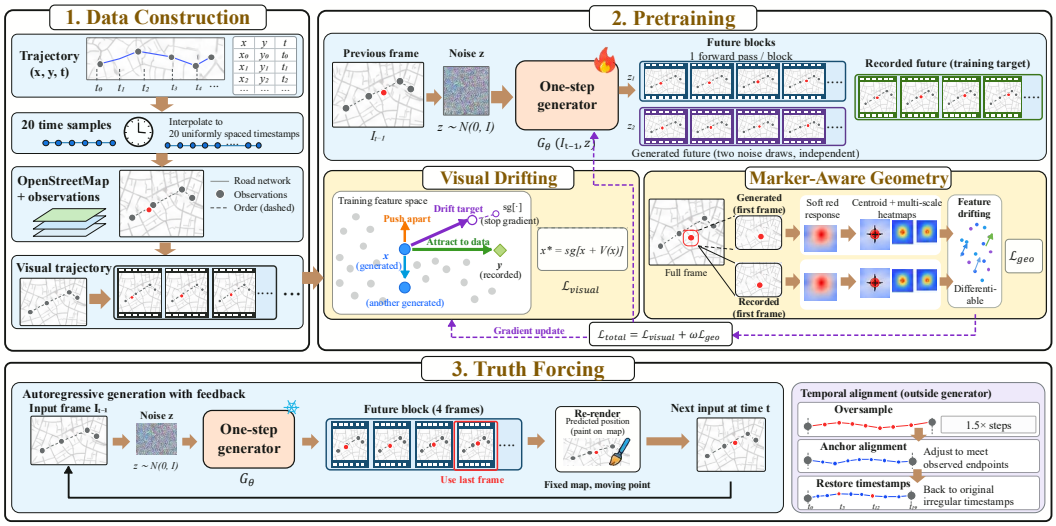


Figure 2: Overview of DriftTraj, comprising three stages: data construction, pretraining, and Truth Forcing.

Differentiable marker-aware geometric supervision. As the marker is much sparser than the map, good RGB reconstruction need not imply accurate localization. We expose marker geometry through a differentiable red-response map:

$$h_I(x,y) = \big([R - \max(G,B) - a]_+ [R-b]_+\big)^4, \qquad P_I(x,y) = \frac{h_I(x,y)}{\max\{\sum_{u,v} h_I(u,v), \epsilon\}}, \tag{6}$$

where $[v]_+ = \max(v, 0)$ and RGB channels are evaluated at (x, y). The contrast thresholds and fourth power suppress weak red map content without using a target-centered crop. With normalized pixel coordinates $\xi_{xy} = (x/(W-1), y/(H-1))$, we extract

$$\mu_I = \sum_{x,y} P_I(x,y)\xi_{xy}, \qquad \Phi(I) = (\mu_I, \mathrm{AvgPool}_4(P_I), \mathrm{AvgPool}_8(P_I))\,. \tag{7}$$

The centroid captures displacement, while the two heatmaps distinguish spatial patterns that share a centroid. Crucially, for nonvanishing red mass $Z_I = \sum_{x,y} h_I(x,y) > \epsilon$,

$$\frac{\partial \mu_I}{\partial h_I(x,y)} = \frac{\xi_{xy} - \mu_I}{Z_I}, \tag{8}$$

so localization feedback reaches the generated pixels through a position-dependent gradient, rather than a discrete detection decision.

Let $v_{k,\ell}$ be the normalized vector of feature group $\phi_\ell(\hat{I}_1^{(k)})$, and let $\widehat{V}_c^{(\ell)}$ be its drifting field, with $\phi_\ell(I_1)$ as the positive feature. The geometry and combined objectives are

$$\mathcal{L}_{\text{geo}} = \mathbb{E}_{(c,Y),z_{1:n}}\left[\frac{1}{3n}\sum_{\ell=1}^{3}\sum_{k=1}^{n}\left\|v_{k,\ell} - \mathrm{sg}\Big(v_{k,\ell} + \widehat{V}_c^{(\ell)}(v_{k,\ell})\Big)\right\|_{\text{mse}}^2\right], \tag{9}$$
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{visual}} + \omega\mathcal{L}_{\text{geo}}.$$

4.3 Truth Forcing

Autoregressive generation feeds each prediction into the next step, allowing small errors to accumulate. In our visual formulation, this affects not only the predicted location but also its appearance: changes to the map or the marker's shape and color can progressively shift the conditioning image away from the training representation. We introduce *Truth Forcing* to keep the visual context fixed and repeatedly anchor the rollout to the supplied observations.

After each generator call, we take the fourth generated frame, extract the red marker's position using the differentiable centroid in Equation 7, and redraw a standard marker on the original annotated map. This freshly rendered image becomes the next input. Re-rendering therefore keeps the static content and marker appearance consistent with training, reducing appearance-induced distribution shift without constraining the trajectory to repeat a previous position.

Observation-guided oversampling. Given that some intermediate points are known, we recover the trajectory interval by interval, starting each interval from its known observation. Because the generator does not receive explicit timestamps, a generated trajectory may not reach the next known observation within the nominal number of steps. We therefore generate approximately 1.5 times the nominal number of steps for each interval on the normalized timeline. This oversampling allows for variation in the generator's pace before assigning predictions to the observed time interval. Let a_j be the observed pixel location at time s_j, and let $\tau(t)$ map timestamps to the normalized timeline. Writing $\Delta_j = \tau(s_{j+1}) - \tau(s_j)$, the local sequence starts at $q_0 = a_j$ and uses $M_j = \max(1, \lceil 1.5\Delta_j \rceil)$ generated steps. We select a local endpoint by balancing distance to the next known observation against deviation from the expected step count:

$$\ell^\star = \max\left(1, \arg\min_{0 \le \ell \le M_j} \left\{ \|q_\ell - a_{j+1}\|_2 + \eta|\ell - \Delta_j| \right\}\right), \tag{10}$$

where η controls the timing penalty. The selected sequence is then resampled to the interval's duration, and its endpoint residual is distributed linearly along the path so that both observed endpoints are preserved. With $\mathcal{I}(q; v)$ denoting linear interpolation at local index v, this correction is

$$b_j(\tau) = \mathcal{I}(q; \alpha\ell^\star) + \alpha\big(a_{j+1} - q_{\ell^\star}\big), \qquad \alpha = \frac{\tau - \tau(s_j)}{\Delta_j}. \tag{11}$$

Returning to the requested temporal resolution. After assembling the corrected intervals, we interpolate the recovered positions at every original timestamp in $\mathcal{T}$. We then restore known observations that may have shifted during resampling by interpolating their residuals within each interval. Let $\tilde{u}_k(\tau)$ interpolate candidate k on the internal grid and define $e_{k,j} = a_j - \tilde{u}_k(\tau(s_j))$. At a requested time t in interval j, the aligned path is

$$b_k(t) = \tilde{u}_k(\tau(t)) + (1 - \alpha)e_{k,j} + \alpha e_{k,j+1}, \qquad \alpha = \frac{\tau(t) - \tau(s_j)}{\tau(s_{j+1}) - \tau(s_j)}. \tag{12}$$

The aligned path preserves the observed positions and provides the recovered trajectory at the requested timestamps. Appendix A.4 provides the alignment details.

5 Experiments

5.1 Experimental Settings

Datasets and implementation. We evaluate on four real-world trajectory datasets: Porto, Chengdu, Xi'an, and Beijing, containing 1,013,437, 2,382,422, 1,426,950, and 1,176,097 trajectories, respectively. We split each dataset into training, test, and validation sets in a $4{:}3{:}3$ ratio. Unless otherwise specified, all models of our method are pretrained exclusively on the Porto training split, without exposure to training trajectories from the other three cities. We train for 265,000 optimization steps using AdamW with a learning rate of 1.25×10^{-5}, a global batch size of 32, and $n_{\text{neg}} = 4$. The model has 8M parameters. During training, the generator conditions on one history frame and predicts four future frames. At inference time, we re-render the marker predicted in the fourth frame on the original annotated map and use the resulting image as the history frame for the next prediction, repeating this process autoregressively. We use Recall to assess road-segment recovery and MAE and RMSE to measure geographic localization error. Detailed implementation choices and metric definitions are provided in Appendix A. Fine-tuning uses fewer target-city training trajectories than the learned baselines, which use each city's full training split.

Baselines. We compare against nine baselines. Linear interpolation (Linear) estimates each missing position between consecutive observations by interpolating their coordinates according to the

Table 1: Comparative results on four cities. Best and second-best results across all methods are bold and underlined, respectively.

	Porto			Xi'an			Beijing			Chengdu		
Method	Recall	MAE	RMSE	Recall	MAE	RMSE	Recall	MAE	RMSE	Recall	MAE	RMSE
Linear	66.42	127.6	170.1	85.65	94.2	127.1	50.28	325.5	431.3	82.66	106.2	141.5
DHTR	<u>69.84</u>	135.4	181.7	85.91	162.2	211.2	54.41	486.7	629.4	83.14	168.3	222.3
MTrajRec	66.24	112.1	151.5	82.58	105.9	140.3	53.35	375.1	477.2	83.34	104.8	141.1
TrajGAT	56.44	188.6	251.8	75.06	203.3	265.1	47.95	476.5	605.4	74.42	200.4	264.2
ST2Vec	61.49	149.1	200.1	76.38	158.1	207.7	51.36	423.5	543.3	75.46	155.1	206.9
TERI	67.76	180.5	249.6	83.32	222.5	301.2	<u>56.61</u>	451.5	592.1	81.62	216.6	294.7
TrajCL	60.11	152.2	204.8	75.76	154.9	204.4	52.63	397.1	509.2	75.12	152.6	204.3
RNTrajRec	67.29	<u>102.6</u>	<u>140.6</u>	84.73	93.1	126.5	55.39	318.2	413.7	84.17	<u>95.1</u>	<u>131.8</u>
MM-STGED	67.52	112.9	153.8	84.01	98.4	132.8	55.49	415.4	551.3	83.81	103.1	140.5
Ours Mean@5	66.59	110.41	146.54	85.84	94.98	130.57	52.35	281.98	372.57	82.02	115.72	157.24
Ours(FT) Mean@5	-	-	-	<u>87.34</u>	<u>87.96</u>	<u>120.75</u>	54.46	<u>272.19</u>	<u>359.37</u>	<u>84.42</u>	104.44	141.50
Ours Top@5 (Oracle)	**75.54**	**89.77**	**120.02**	**91.64**	**80.72**	**110.62**	**65.01**	**218.20**	**294.01**	**89.46**	**94.47**	**131.65**

relative elapsed time. The eight learned baselines span three categories: (i) free-space trajectory recovery methods, including DHTR (Wang et al., 2021) and TERI (Chen et al., 2023a); (ii) road-network-constrained recovery methods, including MTrajRec (Ren et al., 2021), RNTrajRec (Chen et al., 2023b), and MM-STGED (Wei et al., 2024); and (iii) trajectory representation learning methods paired with a recovery decoder, including TrajGAT (Yao et al., 2022), TrajCL (Chang et al., 2023), and ST2Vec (Fang et al., 2022). The learned baselines are trained on each city separately with the same split as our method. Linear interpolation requires no training. Detailed descriptions are provided in Appendix B.

5.2 Comparison Results

To account for stochastic generation, we produce five rollouts for each input trajectory. **Ours Mean@5** averages their predicted coordinates at each query timestamp. **Ours Top@5 (Oracle)** selects the rollout with the lowest RMSE against the ground-truth trajectory and reports all three metrics for that same rollout. This reference-based selection assesses the quality of the generated candidate set. **Ours(FT) Mean@5** applies the same pointwise averaging after fine-tuning the Porto-pretrained model on 180,000 trajectories from each target city. Table 1 compares these three visual-only variants of DriftTraj with existing recovery methods. We have the following findings:

Visual representations support direct cross-city transfer. We first observe that Mean@5 achieves lower MAE/RMSE than every listed baseline in Beijing and remains competitive in the other cities. These results support the transferability of our method without target-city training. Rather than performing road-segment classification, our model generates trajectories over the full map image, which may partly explain its relatively lower road-segment Recall. The shared visual input avoids requiring a new road-segment vocabulary for each city.

Limited-data fine-tuning yields effective recovery. With limited target-city fine-tuning data, (FT) Mean@5 achieves lower MAE/RMSE than every listed baseline in Xi'an and Beijing. This shows that target-city training complements zero-shot transfer and improves recovery within the same visual framework. It also suggests that the Porto-trained model provides a useful base for adaptation to other cities.

Visual generation produces high-quality candidates. Although Top@5 (Oracle) relies on ground truth for selection, it achieves the best Recall, MAE, and RMSE in all four cities without any target-city training. Its improvement over Mean@5 highlights the quality of the generated candidate set. The alternative completions reflect ambiguity in the paths connecting sparse observations. For accuracy-sensitive tasks such as suspect-route reconstruction, multiple plausible trajectories may



Figure 4: Qualitative trajectory recovery examples. (a–e) Selected cases from four cities. (f–j) Five rollouts for the same Porto input.

support human review and selection better than a single fixed prediction. This motivates candidate-set evaluation through Top@5 and suggests an additional application of our method.

5.3 Ablation Study

Figure 3 presents ablations of marker supervision and Truth Forcing re-rendering on Porto.

Ablation of marker supervision. We compare our generator with and without the differentiable marker loss, starting from the same initialization with matched training settings. Over the first 55k updates, *w/o marker loss* exhibits a pronounced early validation-error spike and larger subsequent fluctuations. These results suggest that explicit localization supervision helps the model converge faster. Given limited compute, we did not extend the model without marker loss.

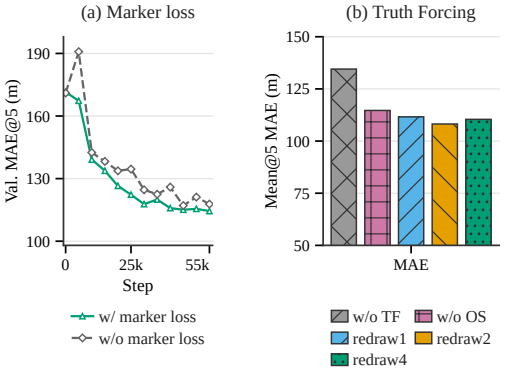


Figure 3: Ablation study of (a) marker supervision and (b) Truth Forcing (vertical axis starts at 50 m).

Ablation of Truth Forcing. We compare five variants. *w/o TF* feeds generated RGB output directly into the next call without re-rendering. *w/o OS* removes the $1.5\times$ oversampling budget, using $1\times$ generation. *redraw1*, *redraw2*, and *redraw4* retain oversampling and use the first, second, or fourth generated frame, respectively, to condition the next generation after re-rendering.

The right panel compares Mean@5 MAE on Porto and complete MAE/RMSE results are in Table 4. We make three observations: 1) Truth Forcing improves prediction quality. Re-rendering keeps the visual context fixed when the generated marker and map image become misaligned. 2) Oversampling improves recovery: removing it increases MAE from 110.41 m to 114.68 m. This is because oversampling helps the model generate longer trajectories if the model does not reach the next observation. 3) Conditioning on the second or fourth generated frame further improves recovery. Using the second frame yields lower errors but requires more generator calls than using the fourth. We therefore use the fourth generated frame to condition the next generation.

5.4 CASE STUDY

Recovery across diverse urban scenarios. Figure 4(a–e) presents five selected examples spanning all four cities. The recovered trajectories capture the overall spatial structure of the journeys across different street layouts, including successive turns, curved segments, and changes in travel direction. The examples show that DriftTraj reconstructs intermediate geometry that largely follows the ground-truth trajectory. The examples also show that this behavior is not restricted to Porto: the Chengdu, Xi'an, and Beijing cases retain meaningful route structure despite differences in urban layout and geographic scale.

Diverse completions from stochastic rollouts. Figure 4(f–j) holds a Porto input fixed and displays all five stochastic rollouts. Each produces a coherent start-to-end sequence, while the inferred geometry differs between the observed locations. Some samples take a more direct continuation, whereas the fourth rollout includes a short reversal before proceeding along the shared travel corridor. Other differences occur around intermediate turns and the approach to the destination. Thus, the generator does not merely reproduce a single deterministic completion; it provides alternative route hypotheses under sparse observations.

5.5 EFFECT OF ROLLOUT

Sparse observations can admit multiple plausible trajectories, and stochastic generation allows DriftTraj to explore these alternative completions. We vary the number of rollouts K across all four datasets using the Porto-trained model and report the Top@K results in Table 2. For $K = 1$, Mean@1 equals Top@1. We have the following findings:

Table 2: Effect of rollout count K on trajectory MAE of Top@K results.

K	Porto	Xi'an	Beijing	Chengdu
1	132.86	117.31	306.28	138.45
2	108.97	98.94	261.80	116.72
4	93.50	84.15	227.19	100.66
8	83.57	74.01	198.38	89.20
10	80.74	71.22	190.89	85.53

1) MAE decreases consistently as K increases, indicating better candidate-set quality with additional rollouts and fixed generator weights. 2) Even a single rollout achieves an MAE of 306.28 m on Beijing, below the best reported baseline in Table 1, showing the potential of our method.

As a widely used video foundation model, Wan2.2-TI2V-5B takes 13.204 seconds per 21-frame video on Pro6000 hardware, implying 36.68 hours for 10,000 sequential generations. In comparison, DriftTraj's complete recovery pipeline takes 11.07 minutes for 10,000 trajectories with $K = 5$ rollouts each on one Pro6000. The approximately $198.8\times$ ratio is relative to this sequential Wan estimate. We choose $K = 5$ as a practical trade-off between recovery quality and runtime. Although non-visual recovery baselines remain faster at approximately two minutes per 10,000 trajectories, our method has made vision-based trajectory recovery practicable. Timing settings and exclusions are detailed in Appendix A.6.

6 CONCLUSION

We presented DriftTraj, a framework for recovering trajectories through map-conditioned visual generation. By representing movement on maps rather than through city-specific road vocabularies, DriftTraj provides a shared interface for cross-city recovery. Its drifting-based generator produces each video block in one forward pass and reconstructs longer trajectories autoregressively. Experiments across four cities support the effectiveness of this formulation for transferable recovery, while stochastic rollouts provide complementary trajectory hypotheses under sparse observations. Limited-data fine-tuning further improves target-city recovery. The Porto-trained model thus offers a reusable starting point for adaptation. Together, these findings highlight visual generation as a promising approach to efficient mobility modeling. Limited computational resources prevented a systematic investigation of larger models and higher rendering resolutions, which remains an important direction for future work.

AI USE STATEMENT

The authors independently developed the research ideas and methodology presented in this paper. Codex assisted with implementation and with translation and language polishing during manuscript preparation. All AI-generated code was reviewed line by line and revised by the authors to ensure that the implementation correctly reflects the intended methodology. For figure design, we used Imagen-2 to brainstorm alternative visualizations, synthesized ideas from multiple proposals, and manually drew the final figures. The authors take responsibility for the final manuscript, implementation, and reported results.

ETHICS STATEMENT

All trajectory data used in this study originate from taxi datasets. The data used in our experiments do not include license plate numbers or direct personal identifiers such as passenger names and contact details. Our task concerns the recovery of vehicle trajectories from sparse observations. We do not seek to identify individuals or reconstruct the movements of any specific person. The absence of direct identifiers does not eliminate privacy risks. Trip origins, destinations, and repeated travel patterns may reveal residences or visits to sensitive locations and enable re-identification when combined with external information. Applications of trajectory recovery should therefore respect privacy safeguards and the access and usage conditions of the source datasets.

REPRODUCIBILITY STATEMENT

We will release the code and model checkpoints on GitHub, including scripts for dataset construction, our best Porto-pretrained model, and scripts for training, fine-tuning, and evaluation. The release will include the configurations and instructions needed to reproduce the experiments and results reported in this paper. Dataset construction scripts will operate on the source datasets under their respective access and usage conditions.

REFERENCES

Yanchuan Chang, Jianzhong Qi, Yuxuan Liang, and Egemen Tanin. Contrastive trajectory similarity learning with dual-feature attention. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*, pp. 2933–2945. IEEE, 2023. doi: 10.1109/ICDE55515.2023.00224. URL https://doi.org/10.1109/ICDE55515.2023.00224.

Yile Chen, Gao Cong, and Cuauhtemoc Anda. TERI: an effective framework for trajectory recovery with irregular time intervals. *Proc. VLDB Endow.*, 17(3):414–426, 2023a. doi: 10.14778/3632093.3632105. URL https://www.vldb.org/pvldb/vol17/p414-chen.pdf.

Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. Rntrajrec: Road network enhanced trajectory recovery with spatial-temporal transformer. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*, pp. 829–842. IEEE, 2023b. doi: 10.1109/ICDE55515.2023.00069. URL https://doi.org/10.1109/ICDE55515.2023.00069.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *CoRR*, abs/2602.04770, 2026. doi: 10.48550/ARXIV.2602.04770. URL https://doi.org/10.48550/arXiv.2602.04770.

Ziquan Fang, Yuntao Du, Xinjun Zhu, Danlei Hu, Lu Chen, Yunjun Gao, and Christian S. Jensen. Spatio-temporal trajectory similarity learning in road networks. In Aidong Zhang and Huzefa Rangwala (eds.), *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 347–356. ACM, 2022. doi: 10.1145/3534678.3539375. URL https://doi.org/10.1145/3534678.3539375.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (eds.),

Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 2672–2680, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pp. 617–628. IEEE Computer Society, 2018. doi: 10.1109/ICDE.2018.00062. URL https://doi.org/10.1109/ICDE.2018.00062.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In Divyakant Agrawal, Walid G. Aref, Chang-Tien Lu, Mohamed F. Mokbel, Peter Scheuermann, Cyrus Shahabi, and Ouri Wolfson (eds.), *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*, pp. 336–343. ACM, 2009. doi: 10.1145/1653771.1653818. URL https://doi.org/10.1145/1653771.1653818.

Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. Mtrajrec: Map-constrained trajectory recovery via seq2seq multi-task learning. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao (eds.), *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1410–1419. ACM, 2021. doi: 10.1145/3447548.3467238. URL https://doi.org/10.1145/3447548.3467238.

Hao Sun, Changjie Yang, Liwei Deng, Fan Zhou, Feiteng Huang, and Kai Zheng. Periodicmove: Shift-aware human mobility recovery with graph neural network. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (eds.), *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pp. 1734–1743. ACM, 2021. doi: 10.1145/3459637.3482284. URL https://doi.org/10.1145/3459637.3482284.

Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE Trans. Knowl. Data Eng.*, 33(3):921–934, 2021. doi: 10.1109/TKDE.2019.2940950. URL https://doi.org/10.1109/TKDE.2019.2940950.

Tonglong Wei, Youfang Lin, Yan Lin, Shengnan Guo, Lan Zhang, and Huaiyu Wan. Micro-macro spatial-temporal graph-based encoder-decoder for map-constrained trajectory recovery. *IEEE Trans. Knowl. Data Eng.*, 36(11):6574–6587, 2024. doi: 10.1109/TKDE.2024.3396158. URL https://doi.org/10.1109/TKDE.2024.3396158.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *CoRR*, abs/2509.20328, 2025. doi: 10.48550/ARXIV.2509.20328. URL https://doi.org/10.48550/arXiv.2509.20328.

Tong Xia, Yunhan Qi, Jie Feng, Fengli Xu, Funing Sun, Diansheng Guo, and Yong Li. Attnmove: History enhanced trajectory recovery via attentional network. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 4494–4502. AAAI Press, 2021. doi: 10.1609/AAAI.V35I5.16577. URL https://doi.org/10.1609/aaai.v35i5.16577.

Di Yao, Haonan Hu, Lun Du, Gao Cong, Shi Han, and Jingping Bi. Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation. In Aidong Zhang and Huzefa Rangwala (eds.), *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 2275–2285. ACM, 2022. doi: 10.1145/3534678.3539358. URL https://doi.org/10.1145/3534678.3539358.

A Method Implementation and Evaluation Protocol

This appendix details geometry-aware training, observation-anchored rollout, geographic calibration, and evaluation metrics for DriftTraj.

A.1 Datasets and Sparse Observation Sampling

We use the fixed trajectory-level training, validation, and test partitions listed in Table 3, corresponding to a $40\%/30\%/30\%$ split. The dense reference trajectories have nominal sampling intervals ϵ of 15 seconds in Porto, 12 seconds in Xi'an, 60 seconds in Beijing, and 12 seconds in Chengdu. Sparse inputs retain the first and last points and a randomly selected subset of interior points at the default retention ratio $\gamma = 0.1$. This gives a nominal observation interval of ϵ/γ, or 150, 120, 600, and 120 seconds, respectively, with individual gaps varying because of random selection and endpoint retention. We read the precomputed observation indices from the dataset and keep them fixed across generation seeds, so multiple rollouts differ in generation noise rather than in their observed inputs. All test-set results, including the main comparison in Table 1 and the inference ablations, are evaluated on the complete test partition of the corresponding city at this fixed sparsity level.

Table 3: Trajectory counts in the fixed dataset partitions.

Dataset	Total	Training	Validation	Test
Porto	1,013,437	405,374	304,031	304,032
Xi'an	1,426,950	570,780	428,085	428,085
Beijing	1,176,097	470,438	352,829	352,830
Chengdu	2,382,422	952,968	714,727	714,727

Rendering consistency. We use OpenStreetMap-based map renderings and red trajectory markers during both training and evaluation. Our cross-city transfer results assume a consistent rendering protocol, including map style, zoom-level settings, and marker colors. They do not establish invariance to different basemap providers, zoom levels, or marker colors, whose effects remain unevaluated. Thus, the shared visual representation is supported across cities under matched rendering settings, rather than across arbitrary map appearances.

A.2 Generator Architecture and Training Budget

The generator f_θ is a convolutional video U-Net with spatial self-attention. The conditioning RGB frame is broadcast over four future time slots and concatenated channel-wise with Gaussian noise, giving an input of shape $B \times 6 \times 4 \times 96 \times 96$. The output contains four RGB frames of shape $B \times 3 \times 4 \times 96 \times 96$. The encoder has four spatial scales (96, 48, 24, and 12 pixels per side), all with 96 feature channels and two residual blocks per scale. The decoder uses three residual blocks per scale with concatenated encoder skip connections. Two residual blocks surround an attention block at the bottleneck.

Each residual block uses GroupNorm with 32 groups, SiLU activations, and two factorized spatiotemporal convolutions. Each such convolution applies a 3×3 spatial convolution followed by a length-three temporal convolution with symmetric padding. Downsampling uses spatial stride two, and upsampling uses nearest-neighbor interpolation followed by convolution. Both preserve the four-frame temporal dimension. Spatial self-attention operates independently within each frame at the 24×24 and 12×12 scales and at the bottleneck, using 12 heads of width eight. Dropout is zero. The inherited conditioning MLP maps a zero-valued two-dimensional input through widths $2 \to 256 \to 256$. Its embedding supplies scale and shift modulation in the residual blocks. No external action input is used.

The generator has 8,367,425 parameters. The following breakdown counts one network, without double-counting the EMA copy used for inference.

Component	Parameters
Encoder (including input projection and downsampling)	2,685,504
Bottleneck	580,032
Decoder (including upsampling)	5,032,512
Conditioning MLP	66,560
Output normalization and RGB projection	2,817
Total	8,367,425

The Porto checkpoint used for evaluation is trained for 265,000 optimizer steps with a global batch size of 32 trajectory windows, implemented as 16 windows per GPU on two GPUs. This batch size counts conditioning windows before replication for the training noise samples.

A.3 Conditional Drifting Implementation

Input and output images have resolution 96×96, with RGB values in $[-1, 1]$. One history frame conditions four jointly generated future frames. Training uses $n = 4$ noise draws per history and a single recorded future block. For each future frame, the RGB image is flattened into a vector. The empirical drifting loss operates on the generated vectors and the recorded target for that history, with each conditioning history processed independently.

For each history and future-frame index, let D be the vector dimension and d the mean pairwise distance between generated samples and the concatenation of generated and positive samples. With gradients stopped, the implementation sets the pixel-vector scale to $\max(d/\sqrt{D}, 10^{-3})$ and normalizes kernel distances by $\max(d, 10^{-3})$. At temperatures 0.02, 0.05, and 0.2, it forms affinities from the geometric mean of row-wise and column-wise softmax weights, masking generated self-pairs. Positive affinities are multiplied by the sum of negative affinities, and negative affinities by the sum of positive affinities, to form the attraction–repulsion field. Each temperature's field is divided by its root mean square magnitude, with a numerical floor, before the fields are summed. Equation 5 applies to these scaled pixel vectors and the summed field. The loss is averaged over generated samples, pixels, future frames, and training examples.

Marker-aware geometric supervision. The red-response and feature definitions are given in Equations 6 and 7. Channels are expressed on the RGB unit scale. We use $a = 0.15$, $b = 0.35$, and $\epsilon = 10^{-8}$. Centroid coordinates are normalized to $[0, 1]$; average pooling with kernel and stride 4 or 8 produces 24×24 and 12×12 heatmaps at the current resolution. Each heatmap is flattened, and the three feature groups receive separate drifting normalization before their losses are averaged. We use $\omega = 0.05$ and compute $\mathcal{L}_{\text{geo}}$ on the first predicted frame. The combined objective supervises RGB appearance and marker geometry, with gradients propagating through the soft marker features to the generated pixels. The ablation without geometric supervision sets $\omega = 0$ in Equation 9. Inference uses exponential moving average (EMA) weights. We use n for the number of training noise samples per history and K for the number of inference rollouts.

Centroid gradient. For $Z_I > \epsilon$, write $\mu_I = A_I/Z_I$, where $A_I = \sum_{x,y} h_I(x, y)\xi_{xy}$. The quotient rule gives $\partial\mu_I/\partial h_I(x, y) = (Z_I\xi_{xy} - A_I)/Z_I^2$, yielding Equation 8. The red-response extractor is differentiable almost everywhere, with the usual subgradient conventions for ReLU and channelwise

maximum. When the mass floor is active, the denominator is fixed at ϵ, and gradients are computed through the clamped normalization in Equation 6.

A.4 Marker Decoding and Temporal Alignment

Marker decoding. We use the differentiable red-response map and centroid in Equations 6 and 7 to extract the marker position from each generated frame. The normalized centroid is converted back to pixel coordinates. If extraction fails, we retain the previous position u_{prev}. A pure-red disk of radius 1.4 pixels is then drawn on the fixed 96×96 image. The initial state of the first interval is the input image itself. Subsequent intervals begin with a marker rendered at their observed starting point. The marker position in the fourth generated frame is re-rendered to form the conditioning image for the next generator call.

Internal timeline. For the appendix notation, reindex the supplied observations as $\{(s_j, g_j)\}_{j=0}^{m}$, where s_j is a timestamp and g_j a geographic coordinate pair. Let $a_j = \Pi_B(g_j)$ be its pixel location under the map's geographic-to-pixel transformation Π_B. The 21-position internal grid follows the training renderer's frame convention:

$$\tau(t) = \begin{cases} 0, & t = s_0, \\ 1 + 19\dfrac{t - s_0}{s_m - s_0}, & s_0 < t \le s_m. \end{cases} \tag{13}$$

This retains the renderer's extra starting-frame slot. Near-duplicate anchor positions on this timeline are merged before rollout.

Interval oversampling and endpoint correction. We use an oversampling factor of 1.5 and a timing penalty $\eta = 0.65$ in the endpoint selection and correction described by Equations 10 and 11. The path starts at a_j and ends at a_{j+1}; the next interval restarts from its supplied observation. The interval paths are sampled at integer grid positions, the last anchor is restored, and unfilled internal positions are interpolated from available neighbors. The resulting compact sequence is then resampled at the requested timestamps for scoring.

Full-length reconstruction. Equation 12 restores the observed anchors after temporal resampling. The aligned trajectory passes through the observations and provides an output at every requested timestamp.

A.5 Geographic Calibration

The raster-to-geographic transform is recovered from the supplied observed pixel/GPS pairs. After rescaling pixels and reversing the image's vertical axis, we fit a uniform scale and translation in Web Mercator coordinates and invert the projection. This requires nondegenerate observed point pairs and the square-map rendering convention. We apply this transform to the temporally aligned pixel positions to obtain the recovered GPS trajectory.

A.6 Video-Diffusion Timing Reference

We measure Wan2.2-TI2V-5B in an image-to-video task on Pro6000 hardware, using BF16 precision, FlashAttention 4, 50 sampling steps, and a classifier-free guidance scale of 5. Each request generates a 21-frame video conditioned on an input image. The measured latency is 13.204 seconds, timed from supplying the input image until video generation completes. Model loading, warm-up, and MP4 export are excluded from both timing protocols. Both methods use input-to-output wall-clock timing with these same exclusions, with video completion as Wan's endpoint and recovered coordinates as DriftTraj's endpoint. GPU utilization reaches 100% in both runs, while DriftTraj's GPU memory usage is approximately one-third lower than Wan's. This is an image-to-video measurement, not an end-to-end trajectory recovery measurement including coordinate reconstruction or evaluation metrics.

DriftTraj's measured runtime on a single Pro6000 for 10,000 trajectories with five rollouts per trajectory is 11.07 minutes, or 664.2 seconds. It includes the complete recovery pipeline, including model

inference, marker decoding, re-rendering, temporal alignment, and GPS coordinate conversion, subject to the exclusions above. The resulting ratio $132{,}040/664.2 \approx 198.8$ compares DriftTraj's runtime with sequential Wan generation. The non-visual trajectory recovery baselines remain faster than DriftTraj, and our speed claim is limited to the comparison with the video-diffusion reference under these settings.

A.7 Inference Execution and Evaluation Metrics

Execution. Each candidate has its own reproducible random stream, and each generator call draws fresh Gaussian noise. Independent candidate histories can share a batch while retaining separate rollout states. A resident model and fixed-shape compiled execution with CUDA Graph replay amortize loading and launch overhead. Each rollout proceeds autoregressively through generation, marker decoding, and re-rendering. The inference pipeline operates directly on generated frames and decoded coordinates, followed by temporal alignment and conversion to geographic coordinates.

Coordinate metrics. After the final output has been constructed, let d_{it} be the Haversine distance in meters between its prediction and the reference location for trajectory i at query time t, using Earth radius $6{,}378{,}137$ meters. For a trajectory with N_i query times, compute

$$\mathrm{MAE}_i = \frac{1}{N_i}\sum_t d_{it}, \qquad \mathrm{RMSE}_i = \sqrt{\frac{1}{N_i}\sum_t d_{it}^2}. \tag{14}$$

Dataset metrics give each of the M trajectories equal weight:

$$\mathrm{MAE} = \frac{1}{M}\sum_i \mathrm{MAE}_i, \qquad \mathrm{RMSE} = \frac{1}{M}\sum_i \mathrm{RMSE}_i. \tag{15}$$

These metrics are widely used in the trajectory recovery literature.

Road-segment Recall. We evaluate road-segment recovery using sequence-based Recall. For trajectory i, we map-match the predicted coordinates to the city's directed road network using an HMM matcher and compare the resulting road IDs with the reference road IDs at the query timestamps. Let $\widehat{R}_i$ and R_i denote the predicted and reference sequences after collapsing consecutive repetitions of the same road ID. Nonconsecutive revisits are retained. With LCS denoting the longest common subsequence, we compute

$$\mathrm{Recall}_i = \frac{|\,\mathrm{LCS}(\widehat{R}_i, R_i)|}{|R_i|}, \qquad \mathrm{Recall} = \frac{100}{M}\sum_{i=1}^{M} \mathrm{Recall}_i. \tag{16}$$

Recall measures recovery of the reference road-segment sequence while respecting traversal order. As with MAE and RMSE, each trajectory receives equal weight. MAE and RMSE are computed from the final recovered coordinates. Recall is computed from their map-matched road-segment sequence.

B Baseline Details

We describe the nine baselines in Table 1 below. TrajGAT, TrajCL, and ST2Vec are trajectory representation methods and require a trajectory recovery decoder to produce missing locations. Their names refer to the representation methods, with recovery decoders used for trajectory reconstruction.

Linear interpolation (Linear). For two consecutive observed positions p_j and p_{j+1} at timestamps $t_j < t_{j+1}$, the position at a query time $t \in [t_j, t_{j+1}]$ is estimated as $\hat{p}(t) = (1-\alpha)p_j + \alpha p_{j+1}$, where $\alpha = (t - t_j)/(t_{j+1} - t_j)$. Applying this rule to each observation interval yields a piecewise-linear trajectory that preserves the observed positions and assumes constant velocity within each interval. The method requires no training.

DHTR. Deep Hybrid Trajectory Recovery (Wang et al., 2021) combines neural sequence modeling with statistical calibration. Its subsequence-to-sequence architecture uses spatial and temporal attention to infer missing locations from irregularly sampled observations. A Kalman-filter component then calibrates coarse-grained predictions into fine-grained coordinates, helping account for predictive uncertainty. The method recovers locations in free space.

MTrajRec. MTrajRec (Ren et al., 2021) formulates map-constrained recovery as multi-task sequence-to-sequence learning. A GRU-based encoder–decoder jointly predicts the road segment associated with each recovered point and its relative position within that segment. This couples discrete route identification with continuous position estimation, allowing the recovered trajectory to respect road-network constraints.

TrajGAT. TrajGAT (Yao et al., 2022) was developed for trajectory similarity computation, particularly for long trajectories. It uses a spatial quadtree to organize the region hierarchically and constructs trajectory graphs that connect observations with this spatial structure. Graph attention replaces dense self-attention in the trajectory encoder to capture long-range dependencies through the graph. For recovery, the resulting representation is passed to a trajectory recovery decoder.

ST2Vec. ST2Vec (Fang et al., 2022) learns spatiotemporal representations of road-network-constrained trajectories for similarity computation. It incorporates both spatial relationships and temporal information, allowing its representations to distinguish trajectories that follow similar routes but exhibit different temporal patterns. In the recovery setting, its encoder supplies the trajectory representation to a decoder that predicts the missing GPS positions.

TERI. TERI (Chen et al., 2023a) considers trajectory recovery with irregular time intervals, where the positions requiring recovery are not necessarily given in advance. It separates the task into detecting missing positions and imputing their locations. Both stages use an enhanced Transformer encoder with continuous time–distance encodings, transition-graph information, and trajectory contrastive learning. This design addresses both where to insert missing observations and which locations to predict.

TrajCL. TrajCL (Chang et al., 2023) learns trajectory representations through contrastive learning over augmented views of trajectories. Its dual-feature self-attention encoder combines spatial and structural information without relying on a recurrent backbone. The learned representations support similarity computation and can be adapted to recovery by adding a decoder.

RNTrajRec. RNTrajRec (Chen et al., 2023b) combines road-network representation learning with a spatial-temporal Transformer for trajectory recovery. It incorporates road topology and local subgraphs around observed GPS points to enrich the trajectory representation with its surrounding network context. The recovery model predicts road-constrained locations by combining segment identification with within-segment position estimation, bringing structural road information into the sequence reconstruction process.

MM-STGED. MM-STGED (Wei et al., 2024) combines micro-level trajectory semantics with macro-level mobility information in a graph-based encoder–decoder. Individual trajectories are represented as graphs to model observations and the relationships between them. Macro-level information, including road conditions and shared travel preferences inferred from groups of trajectories, guides the decoder. The resulting formulation connects individual movement patterns with population-level context for map-constrained recovery.

C Additional Ablation Analysis

C.1 Training Prediction Horizon

This experiment varies the number of future frames predicted during *training*, unlike the inference-time redraw interval in Section 5.3. We compare generators trained to predict one, two, or four future frames, retaining the differentiable marker loss with weight 0.05.

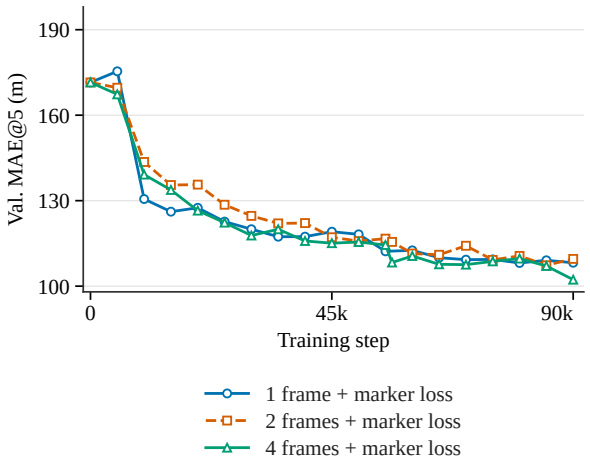


Figure 5: Effect of the training prediction horizon on Porto validation dynamics with the marker loss retained.

Figure 5 shows that all three branches improve substantially and approach similar validation errors during training. The four-frame branch avoids the early error spike visible in the one-frame branch and exhibits a relatively smooth validation curve. These traces suggest that a longer training prediction horizon can help early optimization, rather than demonstrating a uniform advantage at every checkpoint. Longer horizons also increase GPU memory requirements. Four-frame prediction provides a practical choice within the tested range.

C.2 Truth-Forcing Components

Table 4 reports the complete inference ablation on the complete Porto test set using five generation seeds. Mean@5 averages the recovered coordinate sequences before computing trajectory-macro errors. The *w/o OS* setting uses four-frame redraw with a $1\times$ rather than $1.5\times$ generation budget, retaining the endpoint timing penalty $\eta = 0.65$ and endpoint correction. It is evaluated by replaying the corresponding prefixes of the saved generated segments, without fresh model inference. Its matched control is *redraw4*. Removing oversampling increases MAE/RMSE from 110.41/146.54 to 114.68/151.55 m.

Table 4: Truth-Forcing inference ablations on Porto (Mean@5).

Variant	MAE	RMSE
w/o TF	134.51	177.28
w/o OS	114.68	151.55
redraw1	111.59	148.31
redraw2	**108.18**	**143.48**
redraw4	110.41	146.54

C.3 Parameter Sensitivity

We examine the sensitivity of Mean@5 recovery performance to the timing penalty weight η in Equation 10. We select $\eta = 0.65$ on the Porto validation set. Both $\eta = 0.65$ and the $1.5\times$ oversampling factor are transferred unchanged to all target cities, without target-city recalibration. Table 5 reports the sensitivity results on the test set. The selected value yields an MAE of 110.41 meters and an RMSE of 146.54 meters, only 0.14 and 0.19 meters above the lowest test errors, respectively, which occur at $\eta = 0.5$. Performance remains stable over $\eta \in [0.4, 0.8]$ among the evaluated settings, with MAE ranging from 110.27 to 110.53 meters and RMSE from 146.35 to 146.71 meters.

Relative to $\eta = 0$, the validation-selected setting reduces MAE from 118.33 to 110.41 meters and RMSE from 157.29 to 146.54 meters. These results support the benefit of the timing penalty and show that the validation-selected setting remains close to the best-performing region on the test set.

Table 5: Test-set sensitivity to the timing penalty weight η. The value $\eta = 0.65$ is selected on the validation set.

η	Mean@5 MAE	Mean@5 RMSE
0.00	118.33	157.29
0.10	112.14	148.86
0.20	110.63	146.85
0.30	110.39	146.50
0.40	110.29	146.41
0.50	**110.27**	**146.35**
0.60	110.30	146.36
0.65	110.41	146.54
0.80	110.53	146.71
0.90	110.55	146.75
1.00	110.56	146.77

D Robustness of Marker Extraction

Table 6: Marker extraction on ground-truth and generated samples. Localization errors are in pixels. Valid extraction rates are percentages. Generated samples use the Porto-trained model without target-city fine-tuning.

	Ground-truth samples				Generated samples
City	Mean error	RMSE	Max error	Valid (%)	Valid (%)
Porto	0.7286	0.7472	1.7269	100.00	99.73
Xi'an	0.7239	0.7379	1.6967	100.00	99.43
Beijing	0.7195	0.7328	1.5965	99.98	99.44
Chengdu	0.7295	0.7493	1.7459	100.00	99.37

We examine whether map content interferes with the differentiable marker extractor in Equation 6. Table 6 summarizes the extraction results. In an extraction experiment on 10,000 ground-truth samples, both the mean localization error and RMSE are below one pixel in each of the four cities, with valid extraction rates of at least 99.98%. These results indicate accurate localization on the evaluated map backgrounds, including the gray observation markers. Gray pixels have little red-channel contrast and are therefore suppressed by the response function. The fourth-power weighting also reduces the contribution of weak red background responses.

We additionally evaluate extraction on generated test frames from the Porto-trained model without target-city fine-tuning. Valid extraction rates range from 99.37% to 99.73% across the four cities. When extraction fails, the rollout retains the previous position and continues from that point. The high extraction rates support the practical reliability of the decoder.

E Transfer from an Alternative Pretraining City

To examine whether cross-city transfer depends on choosing Porto as the pretraining city, we additionally train DriftTraj on Beijing. Due to time constraints, this preliminary model is trained for 80,000 steps, compared with 265,000 steps for the Porto model. We retain the same dataset partitions and complete test sets for all four cities. Table 7 shows that the Beijing-trained model also transfers to the other cities, and target-city fine-tuning improves all three metrics in each destination city. Although the shorter-trained model generally has higher recovery errors than the Porto-trained model, these initial results support the applicability of the visual representation beyond a single pre-

training city. The unequal training budgets do not isolate the effect of source-city choice. We plan to extend training and update these results during the rebuttal and camera-ready stages.

Table 7: Recovery with different pretraining cities. The Porto and Beijing checkpoints use 265,000 and 80,000 training steps, respectively.

		Porto			Xi'an			Beijing			Chengdu		
Source	Variant	Recall	MAE	RMSE	Recall	MAE	RMSE	Recall	MAE	RMSE	Recall	MAE	RMSE
Porto	Mean@5	66.59	110.41	146.54	85.84	94.98	130.57	52.35	281.98	372.57	82.02	115.72	157.24
	FT Mean@5	–	–	–	87.34	87.96	120.75	54.46	272.19	359.37	84.42	104.44	141.50
	Top@5 (Oracle)	75.54	89.77	120.02	91.64	80.72	110.62	65.01	218.20	294.01	89.46	94.47	131.65
Beijing	Mean@5	53.60	163.75	213.35	83.94	99.51	134.72	53.45	296.40	388.18	77.17	124.39	165.76
	FT Mean@5	54.49	159.97	209.09	84.92	94.91	129.51	–	–	–	80.90	116.52	156.76
	Top@5 (Oracle)	62.37	138.78	181.21	90.38	83.09	112.68	66.46	216.24	289.38	85.08	106.68	142.09

F Stability across Generation Seeds

We assess the sensitivity of Porto Mean@5 results to generation randomness using four disjoint groups of five seeds with the same model checkpoint and test set. Within each group, the five coordinate predictions are averaged before evaluation, following the main Mean@5 protocol. Table 8 shows small between-group variation relative to the metric magnitudes, supporting the stability of the reported results. Given the cost of repeated full-test evaluation, the main results report one five-seed group (seeds 1–5), rather than averaging across groups. The intervals below quantify uncertainty in the mean across these four groups using a Student's t interval with three degrees of freedom. They concern inference-time randomness for a fixed checkpoint, not variation across training runs or a significance test against other methods.

Table 8: Porto Mean@5 stability across generation-seed groups. The standard deviation is the sample standard deviation across four groups. The final row gives approximate 95% t confidence intervals for the corresponding group means.

Seeds	Recall (%)	MAE (m)	RMSE (m)
1–5	66.588	110.408	146.539
6–10	66.041	110.077	146.274
11–15	66.531	111.223	147.700
16–20	67.021	110.762	147.210
Mean $\pm$ std.	66.545 ± 0.401	110.617 ± 0.491	146.931 ± 0.647
95% CI	[65.907, 67.183]	[109.836, 111.399]	[145.902, 147.960]

G Comparison with Direct Visual Regression

We compare DriftTraj Mean@5 with two additional vision-based baselines. *Coordinate regression* takes the map image with observed markers as input and directly predicts the coordinates of the next four frames. *RGB regression* directly predicts the corresponding RGB frames, without a drifting generative objective. Both baselines follow the same training protocol and use the same amount of Porto training data as DriftTraj. All three methods are trained only on Porto and transferred zero-shot to Xi'an, Beijing, and Chengdu, without target-city training or fine-tuning. We use the same test sets and rollout protocol, including Truth Forcing, for all methods. The DriftTraj results are the Porto-trained Mean@5 results from the main table, without target-city fine-tuning.

Table 9 supports the feasibility of visual trajectory recovery beyond a particular generative formulation. Coordinate regression consistently outperforms direct RGB prediction, suggesting that a visual map can provide useful information for recovering positions. DriftTraj further reduces both MAE

Table 9: Direct visual regression versus DriftTraj Mean@5. Errors are in meters, with lower values indicating better recovery. Best results in each column are bold.

	Porto		Xi'an		Beijing		Chengdu	
Method	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Coordinate regression	125.90	163.49	101.50	136.37	292.82	383.41	120.76	160.20
RGB regression	199.24	259.43	122.39	164.19	343.84	444.61	139.89	184.81
DriftTraj Mean@5	**110.41**	**146.54**	**94.98**	**130.57**	**281.98**	**372.57**	**115.72**	**157.24**

and RMSE in every city, including the three cities unseen during its pretraining. These results favor drifting-based generation with Mean@5 aggregation over the two direct regression alternatives under the evaluated settings.

H Comparison under a Matched Target-City Data Budget

To compare recovery under the same target-city data budget, we train RNTrajRec on 180,000 trajectories per target city, matching the data used to fine-tune DriftTraj from its Porto-pretrained checkpoint, and evaluate on the same complete test sets. Table 10 shows that DriftTraj(FT) Mean@5 achieves higher Recall and lower MAE/RMSE in all three target cities. These results support the effectiveness of visual transfer under limited target-city supervision, complementing the main comparison with baselines trained on the full training splits.

Table 10: Recovery with 180,000 training trajectories per target city, used to train RNTrajRec and fine-tune DriftTraj. Recall is in percent and MAE/RMSE are in meters. Best results in each column are bold.

	Xi'an			Beijing			Chengdu		
Method	Recall	MAE	RMSE	Recall	MAE	RMSE	Recall	MAE	RMSE
RNTrajRec	83.42	101.36	137.82	53.68	343.75	446.92	81.76	110.28	150.63
DriftTraj(FT) Mean@5	**87.34**	**87.96**	**120.75**	**54.46**	**272.19**	**359.37**	**84.42**	**104.44**	**141.50**