EFFICIENT TRAJECTORY PLANNING WITH MULTIMODAL LARGE LANGUAGE MODELS VIA SPECULATIVE MULTI-TRAJECTORY DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language model (MLLM) planners typically serialize trajectories into token sequences and decode them autoregressively, making sequential generation a major inference bottleneck that becomes more severe when multiple candidates are required. To address this issue, we propose Speculative Multi-Trajectory Decoding (SMTD), a target-verified framework that accelerates trajectory generation through a unified draft–verify–select process while keeping the pretrained OneVL target model frozen. Lightweight Medusa and anchored-GRU drafters propose blocks of future trajectory tokens for parallel target verification, and speculative multi-sampling extends this mechanism to multiple candidates through shared scene prefill and batched verification. The resulting efficiency can expose more candidate-set potential within a candidate-generation latency budget, while a Unified Online Candidate Selector (UOCS) converts part of that potential into a final trajectory. Experiments on NAVSIM, ROADWork, and Impromptu demonstrate consistent latency reductions with competitive planning quality. On NAVSIM, single-trajectory decoding decreases from 5.221 s to 1.349 s; moreover, SMTD generates eight candidates in 8.10 s, less than the 10.43 s required by Sequential AR Sampling for two, and obtains 90.26/94.11 Selected/Oracle PDMS. The remaining Oracle–Selected gap identifies online selection as a key bottleneck after generation is accelerated.

1 INTRODUCTION

Trajectory generation maps visual observations and driving context to future ego-motion states Seff et al. (2023); Phan-Minh et al. (2020). A scene may admit several plausible future motions, making multiple candidate trajectories useful for choosing a plan online. Each additional candidate incurs generation cost, and a selector may miss the strongest available trajectory. Multi-candidate planning therefore depends on candidate-set potential, candidate-generation latency, and the quality of the final online choice.

Multimodal large language model (MLLM) planners still decode tokenized trajectories autoregressively despite advances in vision–language–action planning Zhou et al. (2026). A T-token trajectory requires T sequential target steps, and N alternatives repeat this cost. OneVL offers a task- and protocol-matched setting for studying this bottleneck: it supports NAVSIM, ROADWork, and Impromptu, performs latent reasoning during prefill, and retains a standard autoregressive trajectory head Dauner et al. (2024); Ghosh et al. (2025); Chi et al. (2026); Lu et al. (2026).

We propose Speculative Multi-Trajectory Decoding (SMTD) to turn generation efficiency into candidate capacity while keeping the OneVL target frozen. Medusa and anchored-GRU drafters propose K-token blocks for parallel target verification Xia et al. (2023); Cai et al. (2024); Ankner et al. (2024). Both are evaluated for single-trajectory decoding, while the current multi-candidate implementation uses anchored-GRU. Training-set calibration chooses $K^\star$ by balancing committed tokens against drafting and verification cost Mamou et al. (2024); Leviathan et al. (2023). For multiple candidates, speculative multi-sampling shares scene prefill and batches branch verification without intentionally changing the target sampling rule. The Unified Online Candidate Selector (UOCS)

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.

scores the explicit trajectories to choose a final plan; higher candidate-set potential need not yield a better online choice.

Related driving VLA accelerators address different stages. Fast-dDrive combines a block-diffusion driving VLA with Scaffold Speculative Decoding and shared-prefix stochastic rollouts Zhang et al. (2026). Reasoning-aware Speculative Decoding accelerates autoregressive reasoning through routine and deliberative paths Dinh et al. (2026). SMTD instead verifies serialized trajectory tokens and retains explicit candidates for online selection.

We compare speculative multi-sampling with Sequential AR Sampling, Beam Search Freitag & Al-Onaizan (2017), Diverse Beam Search Vijayakumar et al. (2018), and Arithmetic Sampling Vilnis et al. (2023) under the same frozen target model and a geometrically scaled candidate budget. Oracle PDMS measures empirical candidate-set potential, while Selected PDMS measures online selection.

In summary, our contributions are threefold:

- We propose Speculative Multi-Trajectory Decoding (SMTD), an integrated framework that combines target-verified single-trajectory decoding, shared scene prefill and batched verification for multi-candidate generation, and a unified online candidate selector while keeping the OneVL target frozen.
- We evaluate multi-candidate generation against sampling and search baselines under the official NAVSIM v1.1 `navtest` PDMS protocol, and single-trajectory decoding against representative baselines on NAVSIM, ROADWork, and Impromptu. The results show substantial latency reductions with competitive planning quality.
- We analyze candidate budgets chosen by a predefined geometric schedule using Oracle PDMS, Selected PDMS, marginal gains, and Pareto dominance. This analysis identifies dominated configurations and shows where candidate-set potential does not translate into online selection gains under candidate-generation latency constraints.

2 METHODOLOGY

Given scene $\mathbf{x}$, OneVL represents the future trajectory as coordinate, heading, and formatting tokens. SMTD keeps OneVL frozen and uses shared scene prefill and batched target verification to generate multiple candidates. Anchored-GRU provides the proposals in the current multi-candidate implementation; we also study Medusa as an alternative single-trajectory drafter. UOCS then chooses from the generated candidates and the deterministic OneVL reference. Figure 1 shows the full path. We assess candidate-set potential, online selection quality, and candidate-generation latency separately.

2.1 SPECULATIVE MULTI- TRAJECTORY DECODING FRAMEWORK

For a requested candidate count N, SMTD encodes each scene once and initializes branch-local decoding states from the shared prefill cache. For each branch, anchored-GRU proposes blocks of calibrated length $K^\star$ (Section 2.2); frozen OneVL verifies compatible branches in batches. UOCS selects the final trajectory after generation. Figure 1 summarizes this path.

Shared prefill and batched branch verification. The shared prefill KV cache is replicated across N branch-local states. Candidates at the same decoding depth undergo batched target verification; each branch retains its own decoding state and cache Qian et al. (2024); Yang et al. (2025).

Stochastic acceptance and correction. For stochastic generation, the anchored-GRU drafter defines the proposal distribution q_ϕ and frozen OneVL provides the target distribution p_θ. A proposed token d is accepted with probability

$$\alpha(d) = \min\left(1, \frac{p_\theta(d)}{q_\phi(d)}\right). \tag{1}$$

On rejection, the replacement is sampled from $p_{\text{res}}(v) \propto [p_\theta(v) - q_\phi(v)]_+$. Under exact sampling and numerical evaluation, this correction preserves the target distribution of each candidate. Specu-

lative multi-sampling therefore targets generation efficiency rather than an inherent improvement in per-candidate quality. Branches are then regrouped by cache depth.

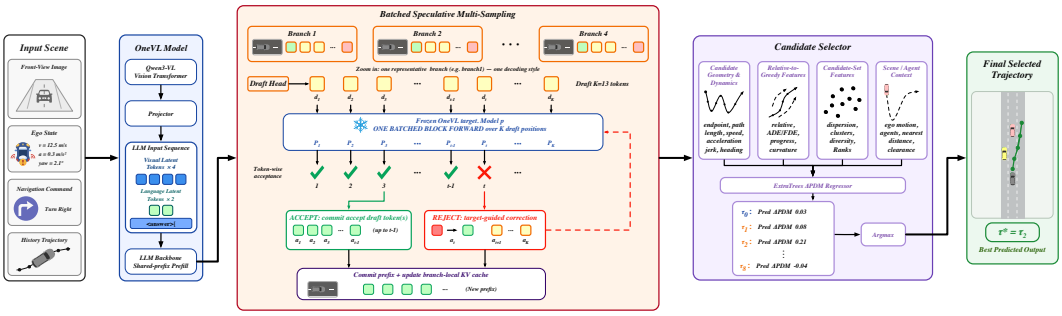


Figure 1: Speculative multi-sampling and online selection. A shared OneVL prefill initializes N branch-local states; the drafter proposes $K^\star$ tokens per branch, and the frozen target verifies branches at compatible decoding depths in batched forward passes. Accepted prefixes are committed, rejected positions are corrected, and UOCS returns the candidate with the largest predicted selection score. The deterministic OneVL reference trajectory is omitted for clarity.

Sharing scene prefill and compatible verification calls reduces repeated prefill and token-level serialization. Drafting, residual correction, branch-local caches, and regrouping add method-specific overhead; their net cost is measured in the experiments. We next detail the drafters and deterministic verification before describing selection in Section 2.3.

2.2 Single-Trajectory Speculative Decoding

Medusa and anchored-GRU are alternative lightweight drafters for single-trajectory decoding; anchored-GRU also supplies proposals for the multi-candidate system in Section 2.1. Let $\mathbf{y} = (y_1, \ldots, y_T)$ be the tokenized trajectory. Frozen OneVL predicts $p_\theta(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$. At decoding round t, $\mathbf{h}_t$ is the verified target hidden state, r_t the current root token, and $\mathbf{d}_t = (d_{t,1}, \ldots, d_{t,K})$ a K-token proposal. Deterministic verification commits only the target's greedy prefix.

Draft architectures. We consider two ways to propose a token block: independent parallel predictions and recurrent predictions conditioned on earlier draft tokens. Medusa attaches K parallel heads to the target state, with head j predicting offset j Cai et al. (2024); later predictions cannot condition on earlier draft tokens.

Inspired by Hydra Ankner et al. (2024), anchored-GRU instead generates the draft sequence recurrently. Its GRU state is initialized from $\mathbf{h}_t$ and updated using the previous draft-token embedding together with a fixed projection of the verified anchor:

$$\mathbf{s}_{t,j} = \mathrm{GRU}\left(\mathbf{s}_{t,j-1}, [E(d_{t,j-1}); A(\mathbf{h}_t)]\right). \tag{2}$$

The fixed anchor limits drift and the recurrent state captures draft-token dependencies. Only the lightweight drafters are trained.

Target verification. A lightweight drafter q_ϕ proposes $\mathbf{d}_t$ conditioned on $(\mathbf{h}_t, r_t)$. Frozen OneVL evaluates the proposed block in one forward pass. If a_t draft tokens are accepted, the confirmed root token and accepted prefix are committed:

$$\mathcal{C}_t = [r_t, d_{t,1}, \ldots, d_{t,a_t}], \qquad 0 \le a_t \le K. \tag{3}$$

Here a_t is the longest prefix matching the target's greedy predictions; the KV cache is cropped accordingly. Every committed token is target-verified, while sequential target calls are reduced Leviathan et al. (2023); Chen et al. (2023).

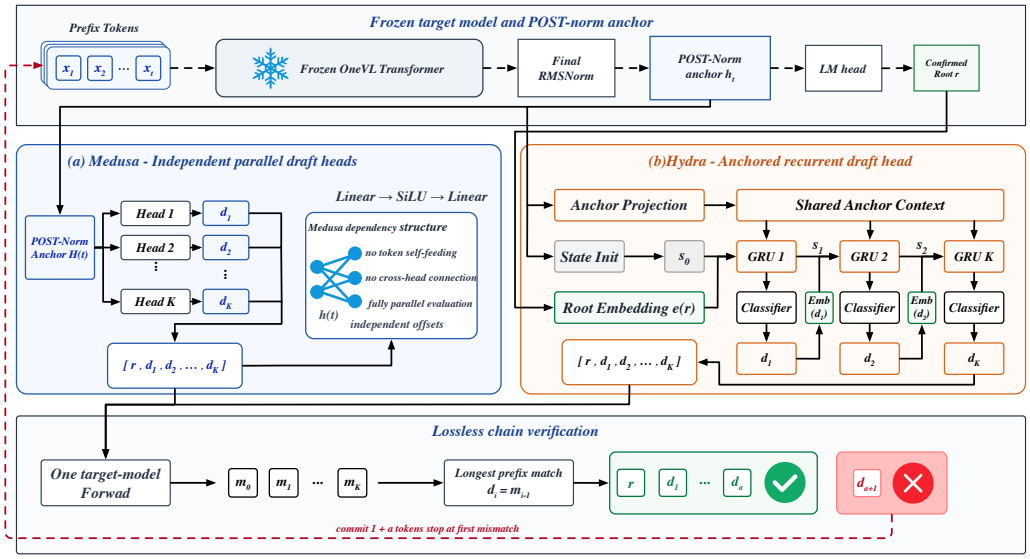


Figure 2: Speculative decoding with two alternative draft architectures for the frozen OneVL target model. Given the post-norm hidden anchor, (a) Medusa predicts the next K token offsets using independent parallel heads, whereas (b) anchored-GRU recurrently generates the draft block using a shared anchor context and previously proposed token embeddings. The complete draft block is verified in one target-model forward pass, after which the confirmed root token and the longest accepted draft prefix are committed. The remaining suffix is discarded at the first mismatch, so only target-verified prefixes are committed.

Cost–acceptance selection of the draft length. We calibrate the draft length to balance the number of tokens committed per round against the cost of proposing and verifying a longer block. Let α_j denote the conditional probability that the j-th draft token is accepted given that all preceding draft tokens have been accepted. Following prior cost–acceptance analyses Mamou et al. (2024); Leviathan et al. (2023), the expected number of tokens committed in one verification round is

$$C(K) = 1 + \sum_{j=1}^{K} \prod_{i=1}^{j} \alpha_i. \tag{4}$$

The leading term corresponds to the target-model token committed in every round. Let r be the normalized marginal cost of proposing and verifying one additional draft position. We select the draft length that maximizes expected committed tokens per unit cost:

$$K^\star = \arg\max_{K \in \mathcal{K}} \frac{C(K)}{1 + rK}. \tag{5}$$

We estimate $\{\alpha_j\}$ and profile the marginal drafting and verification cost on the complete official training partition. Each drafter–dataset $K^\star$ is fixed before evaluation on the disjoint test partition.

2.3 Online Candidate Selection and Quality–Cost Analysis

UOCS selects the final trajectory using information available at inference time. We define its scoring procedure, separate Selected PDMS from Oracle PDMS, and measure the marginal quality gain per added candidate. Generation latency is considered separately from selection latency.

Unified Online Candidate Selector. A single UOCS model is shared across all candidate generators and budgets. For each candidate $\boldsymbol{\tau}^{(j)}$, it maps 34 geometry/kinematics, 16 candidate-relative,

four set-diversity, and 37 scene-level features, collectively denoted by $\mathbf{f}^{(j)} \in \mathbb{R}^{91}$, to a selection score $s^{(j)} = S_\omega(\mathbf{f}^{(j)})$. The selector is trained on the complete official training partition and frozen for evaluation on `navtest`, without access to future ground truth or the offline evaluator. It selects

$$\widehat{\boldsymbol{\tau}} = \underset{\boldsymbol{\tau}^{(j)} \in \{\boldsymbol{\tau}^{(s)}\} \cup \mathcal{T}_N}{\arg\max} S_\omega\Big(\mathbf{f}^{(j)}\Big), \tag{6}$$

where $\mathcal{T}_N$ is the generated candidate set and $\boldsymbol{\tau}^{(s)}$ is the deterministic OneVL reference trajectory.

Selected and Oracle PDMS. We evaluate online selection and offline candidate-set potential separately to measure how much of the available quality the selector realizes. Selected PDMS evaluates UOCS. Oracle selection instead maximizes the offline candidate score q_{off} over the same set; both outputs use official NAVSIM v1.1 final aggregation. Oracle PDMS is thus a non-deployable candidate-set upper bound, and Oracle–Selected measures the remaining selection opportunity.

Marginal candidate efficiency. To quantify the planning-quality gain per additional candidate, let N_ℓ denote the ℓ-th candidate budget and $Q_m(N_\ell)$ the planning-quality score of generation method m at that budget. We define the marginal candidate efficiency between adjacent operating points as

$$\eta_m^Q(N_\ell) = \frac{Q_m(N_\ell) - Q_m(N_{\ell-1})}{N_\ell - N_{\ell-1}}, \qquad \ell \geq 1. \tag{7}$$

Candidate count is the nominal budget, whereas latency is the measured generation cost. We apply η_m^Q to Oracle PDMS to quantify the marginal improvement in candidate-set potential, and separately to Selected PDMS to assess the gains realized by online selection.

Candidate-generation latency and Pareto dominance. Candidate-generation latency includes scene prefill and candidate decoding but excludes UOCS feature extraction and scoring. We therefore compare Oracle PDMS with candidate-generation latency to describe candidate-set potential versus generation cost; Selected PDMS is analyzed separately. A configuration is dominated if another has no greater candidate-generation latency and no lower Oracle PDMS, with at least one strict inequality. Selection latency is not included in these measurements, so the reported multi-candidate times are not end-to-end generate–select latency.

3 Experiments

Evaluation protocol. Training, hyperparameter selection, and draft calibration use the complete official training partitions; configurations are frozen before disjoint test evaluation. Official NAVSIM v1.1 `navtest` has 12,146 scenes, all valid for our 88.9015 Greedy reproduction. Multi-candidate evaluation uses NAVSIM, with Greedy fallback for missing candidates to preserve the full denominator. Every candidate is scored with official v1.1 PDMS before Oracle selection. Single-trajectory evaluation also covers all 2,313 ROADWork and 6,390 Impromptu samples. Single-trajectory latency is mean model inference time; multi-candidate latency covers candidate generation and excludes UOCS.

Candidate-budget schedule. We evaluate multi-candidate generation at geometrically spaced budgets:

$$N_\ell = N_{\min}\rho^\ell, \qquad \ell = 0, \ldots, L, \quad \rho > 1. \tag{8}$$

For NAVSIM, $N_{\min} = 2$, $\rho = 2$, and $L = 2$, giving $N \in \{2, 4, 8\}$. Each step increases candidate count by the same factor, allowing marginal quality gains to be compared across budgets Li et al. (2018); Wu et al. (2025). Smaller sets reuse prefixes of a fixed ordering chosen before evaluation without ground-truth scores.

3.1 Multi-Candidate Generation and Candidate-Set Potential

We compare Sequential AR Sampling, Beam Search Freitag & Al-Onaizan (2017), Diverse Beam Search Vijayakumar et al. (2018), Arithmetic Sampling Vilnis et al. (2023), and speculative multi-sampling with frozen OneVL at the budgets in Eq. 8. Speculative multi-sampling uses anchored-GRU proposals; Medusa is evaluated only in the single-trajectory study.

Table 1: Candidate-generation efficiency and candidate-set quality under the official NAVSIM v1.1 navtest protocol. Each entry reports mean candidate-generation latency in seconds / Oracle PDMS; selection latency is not included. Bold separately denotes the lowest candidate-generation latency and highest Oracle PDMS within each candidate budget.

Method	**Candidate-generation latency (s) ↓ / Oracle PDMS ↑**		
	$N = 2$	$N = 4$	$N = 8$
Sequential AR Sampling	10.43 / 91.94	18.42 / 92.37	36.49 / 93.19
Beam Search	5.63 / 89.94	6.37 / 90.17	12.11 / 90.31
Diverse Beam Search	5.66 / 90.56	6.69 / 91.87	8.20 / 93.17
Arithmetic Sampling	5.66 / **92.17**	6.82 / 92.43	8.32 / 93.34
Speculative Multi-Sampling	**4.34** / 91.92	**6.32 / 93.13**	**8.10 / 94.11**

At $N = 2$, Arithmetic Sampling has the highest Oracle PDMS (92.17), while speculative multi-sampling is fastest (4.34 s, 91.92 Oracle PDMS). At $N = 4$ and $N = 8$, speculative multi-sampling leads both metrics, reaching 6.32 s/93.13 and 8.10 s/94.11, respectively. These realized candidate-set differences do not imply a better per-candidate target distribution.

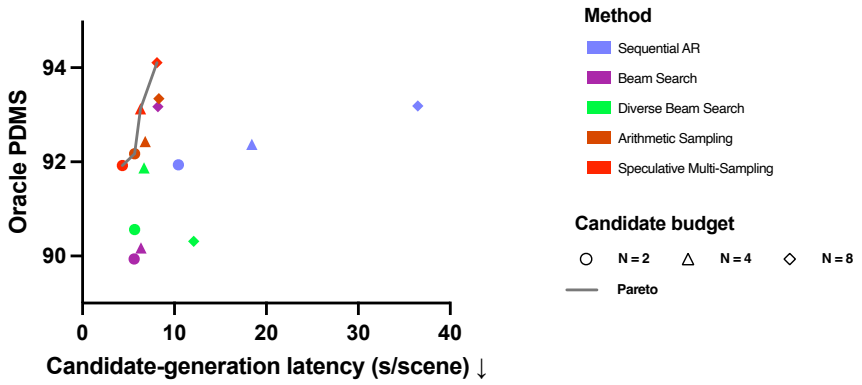


Figure 3: Candidate-set quality–latency trade-off under the official NAVSIM v1.1 navtest protocol. The horizontal axis shows mean candidate-generation latency (s/scene), excluding online selection; the vertical axis shows Oracle PDMS, an offline candidate-set upper bound. Colors denote generation methods, while circles, triangles, and diamonds denote $N = 2, 4, 8$, respectively. The line connects the globally Pareto-optimal configurations.

For speculative multi-sampling, Oracle gains per added candidate decline from 0.605 ($N = 2$ to $N = 4$) to 0.245 ($N = 4$ to $N = 8$); raw gains across methods do not uniformly diminish. The Oracle PDMS–candidate-generation latency frontier contains speculative multi-sampling at all three budgets and Arithmetic Sampling at $N = 2$. The frontier offers speculative multi-sampling at $N = 2$ for tighter budgets, Arithmetic Sampling at $N = 2$ for 0.25 more Oracle PDMS at 1.32 s added generation time, and speculative multi-sampling at $N \in \{4, 8\}$ for larger budgets. All Sequential AR, Beam, and Diverse Beam configurations, plus Arithmetic Sampling at $N = 4, 8$, are dominated. Notably, speculative multi-sampling produces eight candidates in 8.10 s with 94.11 Oracle PDMS, versus 10.43 s and 91.94 for two Sequential AR candidates. Figure 3 visualizes this candidate-set potential versus generation cost; selection is assessed next.

3.2 Candidate Selection Results

We next examine how much of the candidate-set potential identified in Section 3.1 is realized by online selection. Table 2 reports Selected PDMS using the same frozen UOCS across all candidate generators and budgets, with the deterministic Greedy trajectory included as a fallback option, on all 12,146 navtest scenes.

Table 2: Online candidate-selection results under the official NAVSIM v1.1 navtest protocol. Each entry reports Selected PDMS / Δ Greedy / Oracle–Selected. Δ Greedy is the absolute gain over the exact 88.9015 greedy baseline and is displayed to two decimal places. Bold denotes the highest Selected PDMS within each candidate budget.

Method	**Selected PDMS ↑ / Δ Greedy / Oracle–Selected**		
	$N = 2$	$N = 4$	$N = 8$
Sequential AR Sampling	89.50 / +0.60 / 2.43	89.51 / +0.61 / 2.86	89.85 / +0.95 / 3.34
Beam Search	89.04 / +0.14 / 0.90	89.19 / +0.29 / 0.98	89.15 / +0.25 / 1.16
Diverse Beam Search	89.23 / +0.33 / 1.33	89.43 / +0.53 / 2.44	89.95 / +1.05 / 3.22
Arithmetic Sampling	**89.65** / +0.75 / 2.51	**89.90** / +1.00 / 2.53	90.06 / +1.16 / 3.28
Speculative Multi-Sampling	89.55 / +0.65 / 2.38	89.75 / +0.85 / 3.38	**90.26** / +1.36 / 3.85

Arithmetic Sampling leads Selected PDMS at $N = 2$ (89.65 versus 89.55) and $N = 4$ (89.90 versus 89.75); speculative multi-sampling is 1.32 s faster at $N = 2$ and has higher Oracle PDMS at $N = 4$ (93.13 versus 92.43). At $N = 8$, speculative multi-sampling leads Selected PDMS at 90.26 (+1.36 over Greedy), but its Oracle–Selected gap is 3.85 points. Selected gains from doubling the candidate count vary: speculative multi-sampling gains 0.20 then 0.51, Arithmetic Sampling gains 0.25 then 0.16, and Beam Search falls from 89.19 at $N = 4$ to 89.15 at $N = 8$. Candidate-set potential therefore does not automatically translate into online selection gains.

3.3 Single-Trajectory Acceleration and Quality

We next examine the single-trajectory decoding components that underpin SMTD. Training-set calibration selects Medusa $K^\star = 14, 14, 12$ and anchored-GRU $K^\star = 13, 15, 13$ on NAVSIM, Impromptu, and ROADWork, respectively.

Tables 3–5 compare our drafters with reproduced Greedy and three frozen-OneVL, single-trajectory baselines: Jacobi Santilli et al. (2023), Lookahead Fu et al. (2024), and Token Recycling Luo et al. (2025). Thus differences arise from decoding, not the target planner or downstream selection.

NAVSIM uses official v1.1 PDMS as a planning-quality score (higher is better). ROADWork and Impromptu report average and final displacement errors (ADE/FDE; lower is better), measured in pixels and meters, respectively. Latency is mean inference time per scene or sample; speedup is Greedy latency divided by method latency.

Table 3: Single-trajectory results on all 12,146 NAVSIM navtest scenes using official v1.1 PDMS. Latency is seconds/scene and speedup is relative to Greedy. Bold numeric values mark the best displayed result, including ties; bold method labels identify our methods.

Method	Latency (s) ↓	Speedup ↑	PDMS ↑
OneVL AR Greedy	5.221	1.00×	88.9015
Jacobi Decoding	4.316	1.21×	88.9517
Lookahead Decoding	2.324	2.25×	88.9483
Token Recycling	1.963	2.66×	88.8950
Medusa Speculative	1.428	3.66×	**88.9523**
Anchored-GRU Speculative	**1.349**	**3.87**×	**88.9523**

Table 4: Single-trajectory results on all 2,313 ROADWork samples. ADE/FDE are average/final displacement errors in pixels; latency is seconds/sample and speedup is relative to Greedy. Bold numeric values mark the best displayed result, including ties; bold method labels identify our methods.

Method	Latency (s) ↓	Speedup ↑	ADE (px) ↓	FDE (px) ↓
OneVL AR Greedy	5.524	1.00×	12.450	28.764
Jacobi Decoding	3.669	1.51×	12.652	29.267
Lookahead Decoding	2.627	2.10×	12.366	28.390
Token Recycling	2.156	2.56×	**12.351**	**28.358**
Medusa Speculative	**1.427**	**3.87**×	12.652	29.267
Anchored-GRU Speculative	1.555	3.55×	12.652	29.267

Table 5: Single-trajectory results on 6,390 Impromptu samples. ADE/FDE are average/final displacement errors in meters; latency is seconds/sample and speedup is relative to Greedy. Bold numeric values mark the best displayed result, including ties; bold method labels identify our methods.

Method	Latency (s) ↓	Speedup ↑	ADE (m) ↓	FDE (m) ↓
OneVL AR Greedy	4.817	1.00×	1.3467	3.7098
Jacobi Decoding	3.876	1.24×	1.3485	3.7143
Lookahead Decoding	2.539	1.90×	1.3422	3.7000
Token Recycling	1.847	**2.61**×	**1.3381**	**3.6925**
Medusa Speculative	**1.844**	**2.61**×	1.3485	3.7143
Anchored-GRU Speculative	1.991	2.42×	1.3469	3.7127

Single-trajectory analysis. The fastest proposed drafter depends on the dataset. Anchored-GRU is fastest on NAVSIM, reducing mean latency from 5.221 s to 1.349 s ($3.87\times$), while Medusa reaches 1.428 s ($3.66\times$). On the complete 2,313-sample ROADWork evaluation, Medusa is fastest at 1.427 s ($3.87\times$), followed by anchored-GRU at 1.555 s ($3.55\times$). On Impromptu, Medusa is also fastest at 1.844 s ($2.61\times$), narrowly ahead of Token Recycling at 1.847 s; anchored-GRU reaches 1.991 s ($2.42\times$).

Quality is dataset-dependent. On NAVSIM, Medusa and anchored-GRU obtain 88.9523 PDMS versus the 88.9015 Greedy baseline, while Token Recycling obtains 88.8950. We treat the proposed drafters' 0.0508-point difference from Greedy as consistent with finite-precision execution-path variation under BF16, not as a planning-quality gain (Appendix C). On ROADWork, Token Recycling achieves the lowest ADE/FDE (12.351/28.358), while both proposed drafters obtain 12.652/29.267 versus Greedy's 12.450/28.764. On Impromptu, Token Recycling again achieves the lowest ADE/FDE (1.3381/3.6925), followed by Lookahead at 1.3422/3.7000. Medusa obtains 1.3485/3.7143 and anchored-GRU obtains 1.3469/3.7127, both slightly worse than Greedy's 1.3467/3.7098.

Thus, the proposed drafters consistently reduce latency, while their aggregate quality remains close to Greedy with dataset-dependent metric variations. These comparisons are descriptive and do not imply statistical significance.

Draft-length sweep. We validate the cost–acceptance choice by sweeping K on training-set calibration data for every drafter–dataset pair and measuring both single-trajectory speedup over greedy autoregressive decoding and the mean number of committed tokens per target-model verification forward. Figure 4 shows that increasing K generally increases or saturates the committed-token count but does not monotonically improve latency, because longer drafts also increase proposal and verification cost. At the selected operating points, Medusa reaches speedups of $3.66\times$, $2.61\times$, and $4.49\times$ on NAVSIM, Impromptu, and ROADWork, with 4.86, 3.24, and 5.73 committed tokens per verification forward. Anchored-GRU reaches $3.91\times$, $3.64\times$, and $4.16\times$, with corresponding committed-token counts of 5.27, 5.23, and 5.43. Each fixed $K^\star$ attains the highest measured speedup in its displayed neighborhood. Because every speedup is normalized by its architecture-matched greedy

run on calibration data, these sweep values validate the choice of $K^\star$ but should not be read as the test-set latencies in Tables 3–5.

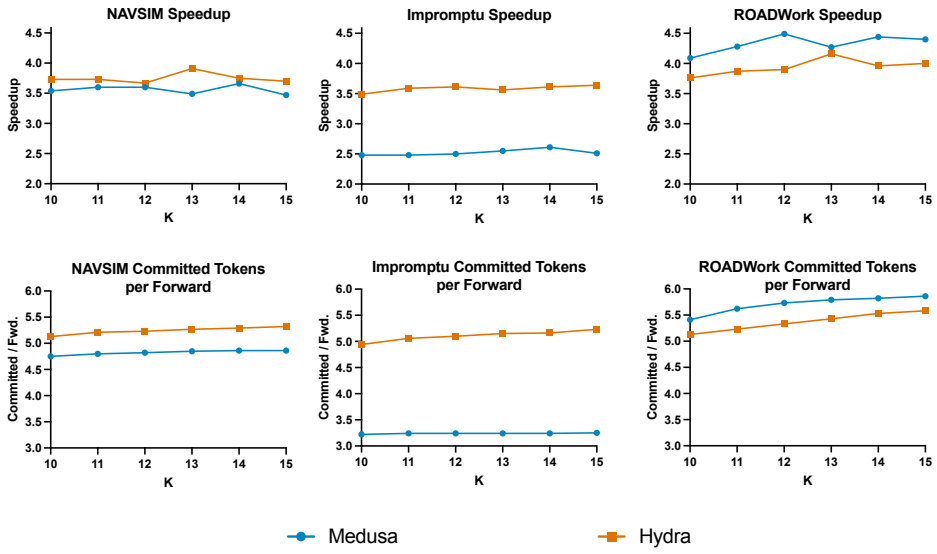


Figure 4: Sensitivity of speculative decoding to the draft length K on NAVSIM, Impromptu, and ROADWork. The top row reports single-trajectory decoding speedup over the matched greedy decoder, and the bottom row reports mean committed tokens per target-model verification forward. Hydra denotes anchored-GRU.

4 Discussion and Limitations

SMTD increases candidate capacity within a candidate-generation latency budget: eight candidates take 8.10 s versus 10.43 s for two with Sequential AR Sampling, with Selected/Oracle PDMS of 90.26/94.11 versus 89.50/91.94. This comparison concerns realized sets, not an intrinsically better per-candidate target distribution. The 3.85-point Oracle–Selected gap at $N = 8$ shows that selection does not realize all candidate-set potential.

Single-trajectory decoding reduces latency with dataset-dependent quality: anchored-GRU is fastest on NAVSIM, Medusa on ROADWork and Impromptu. NAVSIM's small Greedy difference is consistent with finite-precision execution-path variation, not a quality gain (Appendix C). Draft length requires hardware-aware calibration (Figure 4).

Multi-candidate latency excludes UOCS, and multi-candidate quality is evaluated only on NAVSIM. End-to-end generate–select latency remains unmeasured; closed-loop safety, transfer beyond OneVL, peak memory, throughput, energy, and rare-scenario performance require further study.

5 Conclusion

We presented SMTD, a target-verified framework that keeps the OneVL planner frozen. Shared scene prefill and batched verification reduce multi-candidate generation cost. Under official NAVSIM v1.1 evaluation, speculative multi-sampling at $N = 8$ generates eight candidates in 8.10 s, less than Sequential AR Sampling requires for two, and obtains 90.26 Selected and 94.11 Oracle PDMS. This result illustrates candidate-set potential within a candidate-generation latency budget; the 3.85-point Oracle–Selected gap shows that online selection does not realize all of that potential. The single-trajectory drafters also reduce latency on NAVSIM, ROADWork, and Impromptu, with dataset-dependent quality. Efficient planning therefore depends on both generation and selection, while end-to-end latency remains to be measured.

REFERENCES

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Haohan Chi, Huan-ang Gao, Ziming Liu, Jianing Liu, Chenyu Liu, Jinwei Li, Kaisen Yang, Yangcheng Yu, Zeda Wang, Wenyi Li, et al. Impromptu vla: Open weights and open data for driving vision-language-action models. *Advances in Neural Information Processing Systems*, 38, 2026.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

Anh Dung Dinh, Simon Khan, and Flora Salim. Reasoning-aware speculative decoding for efficient vision-language-action models in autonomous driving. *arXiv preprint arXiv:2606.31160*, 2026.

Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pp. 56–60, 2017.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

Anurag Ghosh, Shen Zheng, Robert Tamburo, Khiem Vuong, Juan Alvarez-Padilla, Hailiang Zhu, Michael Cardei, Nicholas Dunn, Christoph Mertz, and Srinivasa G Narasimhan. Roadwork: A dataset and benchmark for learning to recognize, observe, analyze and drive through work zones. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6132–6142. IEEE, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52, 2018.

Jinghui Lu, Jiayi Guan, Zhijian Huang, Jinlong Li, Guang Li, Lingdong Kong, Yingyan Li, Han Wang, Shaoqing Xu, Yuechen Luo, et al. Xiaomi onevl: One-step latent reasoning and planning with vision-language explanation. *arXiv preprint arXiv:2604.18486*, 2026.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. Turning trash into treasure: Accelerating inference of large language models with token recycling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6816–6831, 2025.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Dynamic speculation lookahead accelerates speculative decoding of large language models. *arXiv preprint arXiv:2405.04304*, 2024.

Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14062–14071. IEEE, 2020.

Haifeng Qian, Sujan Kumar Gonugondla, Sungsoo Ha, Mingyue Shang, Sanjay Krishna Gouda, Ramesh Nallapati, Sudipta Sengupta, Xiaofei Ma, and Anoop Deoras. Bass: Batched attention-optimized speculative sampling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 8214–8224, 2024.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, 2023.

Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8545–8556. IEEE, 2023.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Luke Vilnis, Yury Zemlyanskiy, Patrick Murray, Alexandre Tachard Passos, and Sumit Sanghai. Arithmetic sampling: parallel diverse decoding for large language models. In *International conference on machine learning*, pp. 35120–35136. PMLR, 2023.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *International Conference on Learning Representations*, volume 2025, pp. 55824–55845, 2025.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, 2023.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 335–348. Springer, 2025.

Kewei Zhang, Jin Wang, Sensen Gao, Chengyue Wu, Yulong Cao, Songyang Han, Boris Ivanovic, Langechuan Liu, Marco Pavone, Song Han, et al. Fast-ddrive: Efficient block-diffusion vlm for autonomous driving. *arXiv preprint arXiv:2605.23163*, 2026.

Zewei Zhou, Tianhui Cai, Seth Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *Advances in Neural Information Processing Systems*, 38: 27920–27956, 2026.

A Implementation Details

Draft-head training. The OneVL target planner remains frozen throughout all experiments. We train the Medusa and anchored-GRU draft heads on the official training partitions using cached target-model hidden states. Both draft heads are trained for 40 epochs with AdamW, a learning rate of 10^{-3}, weight decay of 10^{-4}, and a training batch size of 8,192 token positions. This batch size refers to draft-head training, not to the number of trajectories generated during inference.

Table 6: Draft-head configurations and calibrated inference lengths. All values of K are selected using training data and fixed before test-set evaluation.

Drafter	**Hidden size**	**Inner width**	**NAVSIM** K	**ROADWork** K	**Impromptu** K
Medusa	2560	1024	14	12	14
Anchored-GRU	2560	2048	13	13	15

Draft-length calibration. Each drafter is trained with a maximum of 15 draft positions. We scan candidate draft lengths using only the training partition, balancing accepted tokens against the computational cost of longer drafts as described in Section 2.2. The resulting drafter–dataset specific lengths are reported in Table 6. Neither test trajectories nor test-set quality scores are used to select K.

Candidate generation and timing. Multi-candidate experiments use the candidate budgets reported in the main paper. The deterministic OneVL greedy trajectory is retained as a reference in the selection pool alongside the generated stochastic candidates. Multi-candidate latency measures candidate generation in seconds per scene and excludes UOCS selection; it must therefore not be interpreted as end-to-end generate–select latency. Single-trajectory latency is reported separately as mean model inference time per scene or sample.

UOCS represents each trajectory with a 91-dimensional feature vector computed from information available at inference time. The representation combines candidate geometry and motion, relationships to the deterministic OneVL reference and other candidates, candidate-set diversity, and scene-level context. The latter includes both candidate-set summaries and observable ego/agent information. Future ground-truth trajectories and offline PDMS scores are not used as test-time features.

Trajectory statistics are computed from consecutive waypoints with $\Delta t = 0.5\,\mathrm{s}$. Endpoint modes are obtained by single-linkage clustering with a distance threshold of 2.0 in the trajectory coordinate system. Candidate-relative and set-level features, including normalized within-set ranks, are recomputed for each scene, decoding method, and candidate budget.

A single shared UOCS model is trained on the complete official training partition and then frozen for evaluation on `navtest`. The same fitted model and feature interface are used for all candidate generators and budgets, without method-specific or budget-specific fine-tuning.

Table 7: Composition of the 91-dimensional UOCS feature vector. Greedy denotes the deterministic OneVL reference trajectory, and the minimum-progress candidate has the smallest final longitudinal displacement.

Main group	Feature subset	Dim.	Included information
Candidate	Endpoint and path geometry	10	Initial and final position, final heading, lateral deviation, path length, displacement, and trajectory straightness.
	Motion increments	10	Statistics of waypoint displacement and longitudinal/lateral motion, including backward-motion indicators.
	Kinematics	10	Mean, variation, extrema, and absolute extrema of finite-difference speed, acceleration, and jerk.
	Heading and length	4	Accumulated, maximum, and standard deviation of heading changes, together with the number of waypoints.
Candidate-relative	Relative to Greedy	6	Average and final distance from Greedy, plus differences in progress, path length, curvature, and lateral displacement.
	Relative to other candidates	6	Distance from the mean trajectory and endpoint, separation from neighboring candidates, and endpoint-cluster membership.
	Candidate ranks	4	Four normalized within-set rank features used by the selector.
Candidate set	Set diversity	4	Endpoint dispersion, longitudinal and lateral endpoint variation, and the number of endpoint modes.
Scene context	Candidate disagreement	12	Near-term, late-horizon, endpoint, heading, and curvature disagreement, maximum pairwise endpoint distance, and braking tendency.
	Mode structure	4	Number and entropy of trajectory modes, within-mode dispersion, and between-mode separation.
	Reference geometry	8	Geometry of the Greedy and minimum-progress trajectories, including their absolute and normalized progress difference.
	Ego state	5	Ego velocity, acceleration, and speed magnitude.
	Agent context	8	Agent availability and density, nearest-agent distance, agents ahead, trajectory clearance, and relative clearance gain.
Total		**91**	

At test time, UOCS scores the candidates generated by the evaluated method together with the deterministic OneVL reference trajectory and returns the trajectory with the highest predicted score.

B Calibration and Timing Details

Training-set calibration. Draft-length calibration is performed exclusively on the official training partition. For each drafter and dataset, we scan candidate lengths from $K = 10$ to $K = 15$ and estimate the conditional acceptance probabilities and the hardware-dependent verification cost. The final value is selected using the cost–acceptance objective in Equation 5 and is fixed before test-set evaluation. The resulting values are reported in Table 6. Neither test-set quality metrics nor test trajectories are used during calibration.

A larger K allows more tokens to be accepted in one verification round, but also increases draft computation and verification cost. Consequently, the selected value need not be the length with the highest acceptance rate. Instead, it represents the best measured balance between accepted tokens and wall-clock cost on the training partition.

Hardware and inference configuration. Latency experiments are conducted on a single NVIDIA RTX 5090D GPU. GPU persistence mode is enabled and the streaming-multiprocessor clock is fixed

at 2692 MHz to reduce run-to-run variation. OneVL inference and target verification use bfloat16 precision. Each input scene is processed individually; in Speculative Multi-Sampling, batching occurs across candidate branches from the same scene rather than across different scenes.

Table 8: Hardware and timing configuration. Candidate batching refers to parallel branches within one scene and should not be confused with the draft-head training batch size.

Setting	Configuration
GPU	Single NVIDIA RTX 5090D
Numerical precision	bfloat16 model inference and verification
GPU configuration	Persistence mode enabled; SM clock fixed at 2692 MHz
Scene batch size	One scene per timed inference instance
Training batch size	8,192 cached token positions
Candidate batching	Candidate branches at compatible decoding depths
Single-trajectory unit	Seconds per scene/sample
Multi-candidate unit	Seconds per scene for generating all N candidates

Single-trajectory latency. Single-trajectory latency is the mean model inference time required to produce one trajectory for one scene or sample. Greedy and accelerated decoders are evaluated under the same hardware configuration and output format. Dataset loading, offline metric computation, and result-file writing are excluded. Speedup is computed from the unrounded mean latencies as

$$\text{Speedup} = \frac{T_{\text{Greedy}}}{T_{\text{method}}}. \tag{9}$$

Multi-candidate latency. Multi-candidate latency measures the model computation required to generate all N candidates for one scene. It includes the shared target-model prefill and candidate decoding but excludes image loading, prompt construction, official PDMS evaluation, and result-file writing. It also excludes UOCS feature extraction and candidate scoring. Accordingly, the reported values and the horizontal axis of the Pareto plot are described as *candidate-generation latency*, not end-to-end generate–select latency.

Because UOCS latency is not included, the quality–latency analysis based on Oracle PDMS characterizes candidate-set potential versus generation cost. Selected PDMS is reported separately to measure how much of that potential is realized by online selection. We therefore do not interpret the reported multi-candidate latency as deployment-level end-to-end latency.

C Deterministic Output-Equivalence Audit

In exact arithmetic, deterministic speculative decoding with strict target verification returns the same token sequence as token-wise Greedy decoding. Our experiments use BF16 inference, however, and blockwise verification and single-token decoding evaluate the target through different tensor shapes and execution paths. Near-tied logits may therefore round differently and change an argmax decision; subsequent tokens can then diverge even though each accepted token agrees with the target prediction computed in its blockwise verification call. Thus, *target-verified* does not imply bitwise identity to a separate token-wise BF16 run.

We audited the saved predictions and official per-scene NAVSIM v1.1 scores for all 12,146 `navtest` scenes, joining records by scene token rather than row order. Medusa and anchored-GRU produced identical saved prediction files. Their token sequences match Greedy in 4,919 scenes (40.50%), while parsed trajectories match exactly in 4,931 scenes (40.60%). Every exactly matched trajectory has an identical official score. The remaining 7,215 changed trajectories account for the entire aggregate difference: 2,351 have a higher per-scene score, 2,451 have a lower score, and 2,413 are unchanged. Eight decoder-invalid outputs use the stipulated Greedy fallback and contribute no score difference.

We additionally isolated the numerical execution path on 256 fixed-seed `navtest` scenes. Standard Greedy generation and an independent token-wise cached implementation agree on all 256 outputs,

whereas a teacher-forced target forward with block length 14 agrees with the standard Greedy sequence in 103 cases. Among the 153 first mismatches, the blockwise top-two logit margin is at most 0.0625 in 119 cases, including 81 exact ties at BF16 precision. This control links the observed divergence to block-versus-token target execution rather than to the draft-head architecture.

Table 9: Full-set audit of Greedy and deterministic speculative outputs under BF16 inference on official NAVSIM v1.1 `navtest`. PDMS values are reconstructed from official per-scene scores.

Audit item	**Result**
Evaluated scenes	12,146
Token-identical to Greedy	4,919 (40.50%)
Trajectory-identical to Greedy	4,931 (40.60%)
Equal trajectories with different scores	0
Greedy PDMS	88.9014518
Speculative PDMS	88.9522646
PDMS difference	+0.0508128

The mixture of positive, negative, and unchanged per-scene differences, and the small net change of 0.0508 PDMS, are consistent with finite-precision execution-path variation rather than a systematic planning-quality benefit from the drafter. We therefore report the observed scores in the main tables but do not interpret their difference from Greedy as a quality gain. This audit localizes the score change to output divergence; a matched higher-precision rerun would be required to isolate BF16 rounding from other kernel-level execution differences.