LoopNav: Benchmarking Spatial Consistency in World Models

Kewei Lian [1] **Shaofei Cai** [2] **Yitao Liang** [2] **Anji Liu** [1]

Abstract

The ability to simulate the world in a spatially consistent manner is a crucial requirement for effective world models. Such a model enables high-quality visual generation, and also ensures the reliability of world models for downstream tasks such as simulation and planning. It must not only retain long-horizon observational information, but also enables the construction of explicit or implicit internal spatial representations. However, existing datasets do not explicitly enforce spatial consistency constraints, limiting both the ability to systematically evaluate this capability and to learn it through data-driven approaches. Furthermore, most existing benchmarks primarily emphasize visual coherence or generation quality, neglecting the requirement of long-range spatial consistency. To bridge this gap, we propose LOOPNAV, a dataset and corresponding benchmark centered on loop-based navigation for evaluating spatial consistency. The dataset comprises 250 hours (20 million frames) of loop-based navigation videos with actions, collected from diverse locations in the open-world environment of Minecraft. We further introduce a Scene Graph Consistency Score to quantify spatial consistency while remaining invariant to pixel-level variations. Dataset, benchmark, and code are open-sourced to support future research.

1. Introduction

Although recent advances in world models have significantly improved the visual quality of generated observations (Zhu et al., 2024; Ding et al., 2024; Brooks et al., 2024; Bruce et al., 2024), these models often struggle to preserve spatial consistency over extended rollouts (Decart et al., 2024; Guo et al., 2025). Spatial consistency, the ability to preserve coherent and stable spatial structures across time, is essential for the reliability of world models in downstream applications such as model-based reinforcement learning (Bar et al., 2024; Alonso et al., 2024; Valevski et al., 2024), autonomous driving (Wang et al., 2023a; Gao et al., 2024), and model predictive control (Du et al., 2023; Hafner et al., 2020; 2024; Yang et al., 2023; 2024). When violated, it can lead to severe failure modes: hallucinated structures, contradictions with prior observations, and visually incoherent scenes. For instance, in a world model guided navigation task, inconsistent rendering of revisited locations impedes global planning and loop closure.

A key challenge underlying spatial inconsistency is the need to retain and utilize information over long temporal horizons. In principle, addressing this challenge requires models to maintain representations of previously observed spatial structures and re-anchor them upon revisitation. However, mainstream Transformer-based architectures are typically limited to short context windows due to the quadratic scaling of the attention mechanism (Vaswani et al., 2017), making it difficult to directly incorporate long-range dependencies spanning hundreds or thousands of frames.

To mitigate this limitation, prior work has explored augmenting world models with memory mechanisms that enable long-term information retention (Graves et al., 2014). These mechanisms may take various forms, including explicit spatial representations such as maps (Gupta et al., 2019; Yang et al., 2025), coordinate systems (Wang et al., 2023b), or graph-based structures (Savinov et al., 2018; Kim et al., 2022), as well as implicit representations based on learned embeddings or attention (Parisotto & Salakhutdinov, 2017; Wang* et al., 2025). While these approaches provide promising directions, they highlight a broader challenge: enabling models to maintain coherent spatial representations over long horizons remains an open problem.

Despite its importance, spatial consistency remains underexplored, particularly in terms of how it is supported, learned, and evaluated. A major reason for this gap is the absence of datasets that explicitly demand spatial consistency. Most existing datasets are designed for open-ended exploration, where agents continually encounter novel structures and objects while rarely revisiting previously seen locations. As a result, world models trained on such data tend to rely on

[1]National University of Singapore [2]Peking University. Correspondence to: Anji Liu <anjiliu@nus.edu.sg>.

"*Accepted to the 1st Workshop on Combining Theory and Benchmarks*, CTB@ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s)."

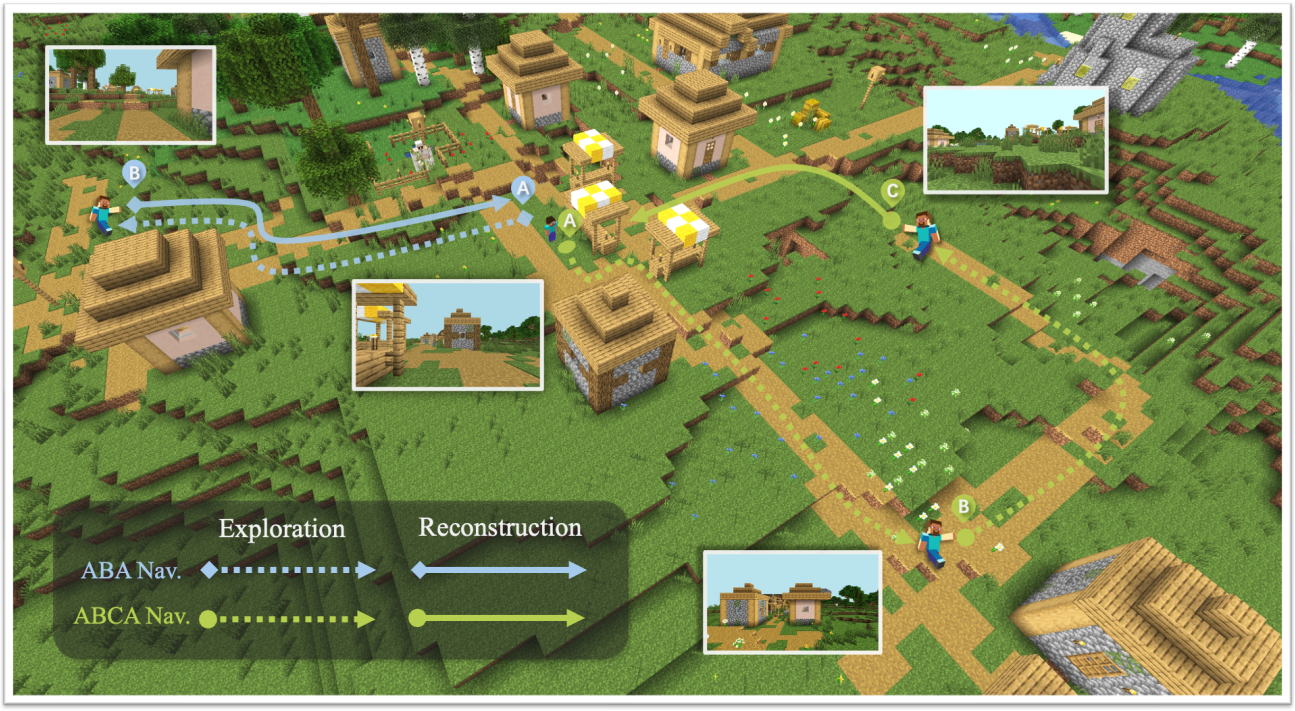


Figure 1. **The Loop-Style Trajectory Data for Training and Benchmarking**. To be able to explicitly enforce spatial consistency, our dataset follows a loop-style navigation trajectory, including both ABA and ABCA types. Our benchmark design explicitly separates the generation phase from the exploration phase and only evaluates the video quality of the generation part.

global environmental priors to generate plausible observations, rather than leveraging episodic memory to reconstruct previously observed content. Meanwhile, evaluation metrics often prioritize visual fidelity and short-term temporal smoothness over long-term spatial coherence or logical consistency. This leads to models that function more as visually impressive but impractical "dream machines", limiting their applicability in downstream decision-making tasks.

Therefore, we argue that such a dataset must feature looped trajectories that revisit the same locations and objects from diverse viewpoints. We construct such a dataset, LOOPNAV, in the open-world environment of Minecraft. The dataset comprises over 250 hours of **loop-style** navigation trajectories across 147 diverse locations. Each trajectory follows a loop exploration pattern, ensuring that the same locations and layouts are revisited from varying camera views and times. This loop-based data structure naturally incentivizes models to learn long-horizon spatial consistency. Besides, we carefully select positions with diverse landmark objects to ensure that observations from different locations are **visually distinguishable**. To support progressive learning, the dataset also includes a **curriculum of sequence lengths**, enabling models to transition from short-term to long-term tasks.

For benchmarking, we adopt an **explore-then-generate** protocol to evaluate spatial consistency. Specifically, we sample a loop trajectory (e.g., $A \rightarrow B \rightarrow A$), where the first segment ($A \rightarrow B$) is provided as context, and the remaining segment ($B \rightarrow A$) serves as the generation target. Existing evaluation metrics, such as SSIM and LPIPS (Zhang et al., 2018), primarily assess visual quality. However, in world model generation, even when conditioned on identical action sequences, the generated trajectory may exhibit slight deviations—such as small translations, scaling, or accumulated drift—due to the inherent stochasticity of action execution and cumulative errors over time. These metrics can be sensitive to such misalignments, failing to reflect true spatial consistency.

To address this limitation, we introduce the **Scene Graph Consistency Score (SGCS)**, an object-centric metric for evaluating spatial consistency. Specifically, video frames are first decomposed into category-wise object masks, and consistency is measured through object-level matching based on semantic category and spatial alignment. By comparing the matched object structures across trajectories, SGCS abstracts away pixel-level differences while remaining sensitive to semantic inconsistencies such as object disappearance, replacement, and large spatial layout changes.

In summary, our contributions are as follows:

- **A dataset and benchmark for spatial consistency.**

We introduce LOOPNAV, a dataset and benchmark centered on loop-based navigation, which bridges the gap between existing world model training setups and the need to evaluate long-horizon spatial consistency. It provides a unified platform for both training and evaluation under explicit spatial consistency constraints.

- **A structure-aware metric for spatial consistency.** We propose the Scene Graph Consistency Score(SGCS), a structure-aware metric that evaluates spatial consistency while abstracting away pixel-level variations. It is robust to minor geometric misalignments, while remaining sensitive to semantic inconsistencies.
- **An open-source data collection framework.** We release a scalable and flexible pipeline for collecting navigation trajectories in Minecraft, which can be readily extended to other data collection settings. We also open-source the dataset and evaluation code to facilitate future research on spatial consistency in world models. Dataset, benchmark, and code are available [1] [2].

2. Related Work

2.1. World Models

The goal of a world model is to simulate the environment: Given the current state and action, it predicts the next state and the corresponding reward. World models were originally proposed to improve sample efficiency in reinforcement learning (Oh et al., 2015; Ha & Schmidhuber, 2018; Hafner et al., 2020; 2022; 2024; Robine et al., 2023; Alonso et al., 2024). Different architectures, to varying degrees, retain information from the past to aid in future image prediction; however, they still lack an effective memory design capable of maintaining long-horizon spatial consistency.

Beyond model-based reinforcement learning, the potential of world models has also been increasingly explored in other domains such as game engines (Menapace et al., 2021; Valevski et al., 2024; Decart et al., 2024), autonomous driving (Hu et al., 2023; Wang et al., 2023a; Gao et al., 2024), robot manipulation (Yang et al., 2023; Wu et al., 2022; Bar et al., 2024). In those downstream tasks, spatial consistency is even more important.

2.2. Minecraft as an AI Testbed

Minecraft is an open-world environment characterized by diverse terrains and rich interaction dynamics. Recent research has increasingly adopted Minecraft as a platform for training generative agents (Baker et al., 2022; Lifshitz et al., 2024; Cai et al., 2023; 2024; 2025b;a) and constructing digital world models (Guo et al., 2025; Decart et al., 2024; Hong et al., 2024; Song et al., 2025).

A more comprehensive review of related work is provided in the Appendix E.

3. LOOPNAV Dataset

3.1. Environment

We collect data in the open-world environment of Minecraft, which provides a suitable platform for large-scale and controlled data generation. Its procedurally generated worlds offer diverse and non-repetitive environments, while its mature ecosystem enables flexible customization and extension. Compared to real-world data collection, it also allows efficient, scalable, and highly parallelizable simulation. Further details are provided in Appendix A.1.

3.2. Collection Pipeline

We propose three principles for trajectory data collection: *visual discriminability*, *loop closure*, and *curriculum-based progression*. Below, we detail each principle, followed by our implementation.

Principle 1: Visual Discriminability

The sequence of observations along the trajectory should be visually distinguishable over time to capture meaningful scene variation.

Within this procedurally generated world, we identify 6 categories comprising a total of 120 distinct villages, covering diverse terrains and architectural layouts. As players navigate through a village, visually distinctive structures such as houses, farms, and pathways serve as salient landmarks for spatial reasoning and cognitive map construction. The "visually distinguishable" principle is particularly important for evaluating spatial consistency. If a scene lacks distinctive objects or structural variation, visually different states may become difficult to distinguish, making it ambiguous whether a generated frame is truly spatially consistent or merely visually similar. By ensuring sufficient variation in object appearance and spatial layout, our dataset enables more reliable evaluation of long-horizon spatial consistency. Detailed descriptions are provided in Appendix B.

Principle 2: Loop Closure

The generated trajectories must form at least one spatial loop, ensuring repeated visits to the same locations.

[1] Dataset page: https://huggingface.co/datasets/kevinLian/LoopNav

[2] Code page: https://github.com/Kevin-lkw/LoopNav

We propose two types of trajectory structures: $A \to B \to A$ and $A \to B \to C \to A$, where A, B, and C denote distinct locations within the environment.

We use the ABA trajectory as a representative case to illustrate our data collection process. As detailed in Algorithm 1, we first select a location S from one of our collected sites,then randomly sample a starting point A within the start range of S. This is done to ensure diversity in the starting positions within the same location. A target location B is sampled within a specified navigation range. Trajectory recording begins: the agent performs a full 360-degree rotation at location A to observe the surrounding environment and establish spatial context. It then navigates from A to B using the A* algorithm, and subsequently returns from B to A, completing the loop. We put visualization of our dataset, including frames and bird-eye view in Appendix B.4.

Principle 3: Curriculum-based Progression

The trajectories follow a curriculum design, gradually expanding the spatial exploration radius to support progressive learning and evaluation.

According to the exploration radius, we define four difficulty levels: 5, 15, 30, and 50 blocks. To improve training robustness and introduce variability, the target location B is not sampled at a fixed distance. Instead, for a difficulty level x, we sample distances within the interval $[0.8x, \sqrt{2}x]$, ensuring diverse trajectory lengths while minimizing overlap between different difficulty levels. The full sampling procedure is also detailed in Algorithm 1. Collect dataset trajectory length is visualized in Figure 6.

3.3. Implementation Details

We collect 20 trajectories for each combination of location and exploration radius. The final dataset contains approximately 20 million frames, amounting to roughly 250 hours of navigation trajectories.

We collect navigation trajectories in Minecraft using a customized data collection pipeline built on Mineflayer. Further details on the data collection system are provided in Appendix A. Action design, and collected data format are in Appendix B.

4. Benchmark

4.1. Loop-Based Evaluation Protocol

Our benchmark is designed around $A \to B \to A$ loop trajectories. In this setting, we treat the $A \to B$ segment as an exploration phase, which serves as the contextual input to the model, and the $B \to A$ segment as a reconstruction phase, where the agent returns to a previously visited location. We provide an illustration of these phases in Figure 1. As such, we evaluate model performance only on the $B \to A$ segment, as it provides a clear test of the model's ability to maintain spatial consistency during long-horizon rollouts. Similarly, for longer $A \to B \to C \to A$ trajectories, we evaluate only the $C \to A$ segment. A detailed discussion is on Appendix C.3.

4.2. Limitations of Visual Quality Metrics

Given aligned ground-truth and generated trajectories, a natural approach is to evaluate their similarity using existing video generation metrics. However, such metrics are not well suited for assessing spatial consistency in world models. The generated trajectory may exhibit small geometric deviations due to accumulated errors in action execution. These deviations can manifest as slight translations, scaling, or viewpoint shifts over long rollouts. Importantly, such perturbations do not necessarily indicate a failure of spatial consistency. However, visual quality metrics are highly sensitive to these geometric perturbations. They may penalize predictions that are semantically consistent but not perfectly aligned with the ground truth, leading to low scores. At the same time, these metrics fail to capture spatial consistency itself. They primarily assess visual fidelity or perceptual similarity, rather than whether the generated trajectory remains consistent with previously observed scenes. Consequently, a visually plausible but spatially inconsistent generation may still receive a high score.

4.3. Scene Graph Consistency Score

To address the limitations of pixel-level metrics, we introduce the **Scene Graph Consistency Score (SGCS)**. SGCS is designed to be robust to minor geometric perturbations, such as translation, rotation, and scaling, while remaining sensitive to changes in spatial consistency, including object disappearance, replacement, and incorrect spatial relationships.

Figure 2 illustrates our evaluation pipeline. Given a generated video and its corresponding ground-truth video of equal length, we first sample aligned frame pairs at a fixed temporal stride. For each pair of frames, we compute SGCS based on category-wise object matching.

Specifically, for each semantic category c (e.g., building, tree, flower, grass), we segment both frames into object masks using SAM 3 (Carion et al., 2026). We compute the centroid of each mask and identify matchable pairs based on their normalized spatial distance. Intuitively, two masks are considered a valid match if their centers are sufficiently close relative to the image size.

Since multiple instances of the same object category may appear in a frame, we perform bipartite matching (via the

Algorithm 1 $A \to B \to A$ Navigation Data Collection in Location S

Require: Start Range R_{st}, Navigation Range R_{nav}, location S
Ensure: A navigation trajectory T from point A to point B and back to A
$A \leftarrow \text{SamplePoint}(S, R_{st})$
$B \leftarrow \text{SamplePoint}(A, R_{nav})$
Trajectory $T \leftarrow []$
Teleport the agent to location A
Agent performs a 360° rotation at point A to observe surroundings
$T \leftarrow \text{Navigate}(A, B, T)$
$T \leftarrow \text{Navigate}(B, A, T)$
return Trajectory T

function SamplePoint($A = (x, y)$, r)
while True **do**
  Sample $\Delta x \sim \text{Uniform}(-r, r)$
  Sample $\Delta y \sim \text{Uniform}(-r, r)$
  $d \leftarrow \sqrt{(\Delta x)^2 + (\Delta y)^2}$
  if $d < 0.8 \cdot r$ **then**
    continue
  end if
  $B \leftarrow (x + \Delta x, y + \Delta y)$
  if IsValidBlock(B) **then**
    return B
  end if
end while

function Navigate(A, B, T)
$S \leftarrow A$
while $S \neq B$ **do**
  $P \leftarrow \text{AStarPlan}(S, B)$
  for all $P_i \in P$ **do**
    $a \leftarrow \text{GetAction}(S, P_i)$
    $S \leftarrow \text{PerformAction}(S, a)$
    Append (S, a) to T
  end for
end while
return T

Hungarian algorithm (Kuhn, 1955)) to obtain the maximum number of matched object pairs. Based on the number of matched and unmatched masks, we compute a category-level consistency score that reflects how well objects of category c are preserved. To aggregate across categories, we weight each category by its average visible area in the two frames, giving more importance to dominant structures. The final frame-level SGCS is obtained as the weighted average over all categories.

For video-level evaluation, we compute SGCS for all sampled frame pairs and report their average. Importantly, SGCS abstracts away pixel-level differences and instead evaluates consistency at the level of object presence and spatial arrangement. As a result, it is robust to minor geometric misalignments, while remaining sensitive to spatial inconsistencies. We leave the details of implimentation in Appendix C.1.

5. Experiments

5.1. SGCS Metric Validation

We evaluate SGCS under controlled perturbations to validate its robustness and sensitivity. Specifically, we apply geometric perturbations (colour change, translation, rotation, scaling) and semantic modifications (object deletion and swapping) to ground-truth videos. We compare SGCS (ours), SSIM, and sim-LPIPS (defined as 1-LPIPS). All metrics are normalized such that higher values indicate better consistency and GT achieves a score of 1. As shown in Figure 3a, SGCS remains stable under geometric perturbations, demonstrating robustness to pixel-level misalignment. In contrast, SSIM and LPIPS degrade significantly under some transformations.

Conversely, under semantic modifications, SGCS decreases sharply, correctly identifying violations of spatial consistency, while SSIM and LPIPS may still assign relatively high scores due to preserved visual appearance.

These results confirm that SGCS achieves the desired balance between invariance to geometric perturbations and sensitivity to spatial inconsistencies. We leave more demonstrations of synthetic perturbation on Figure 9 and the details of implementation on Appendix D.1.

5.2. World Model Baselines

We evaluate the following baselines on our dataset and benchmark. We set the context length of all baselines to 32 frames.

Oasis (Decart et al., 2024) is a world model with a ViT (Dosovitskiy et al., 2021) as spatial autoencoder, and a DiT

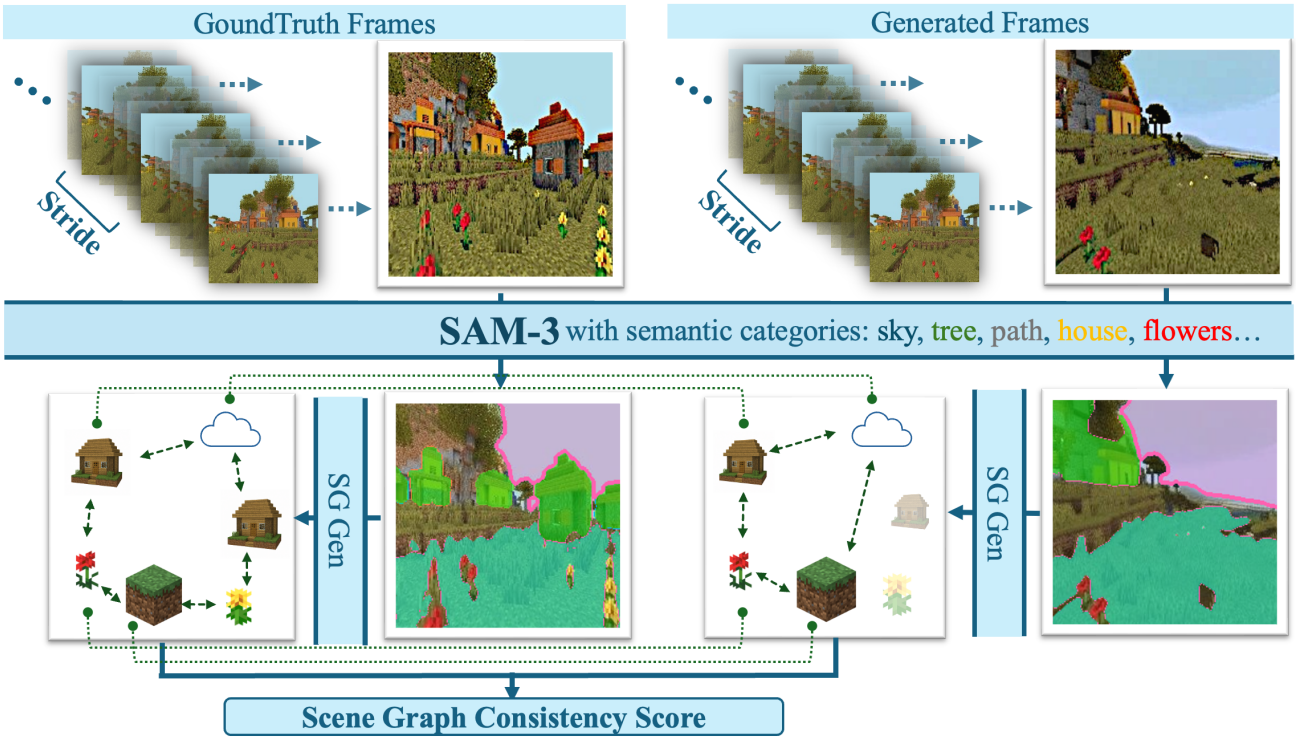


Figure 2. **Evaluation pipeline: Scene Graph Consistency Score(SGCS).** We sample aligned frames from generated and ground-truth videos, extract category-wise object masks using SAM 3, and perform bipartite matching based on centroid distance. The final score aggregates category-level consistency with area weighting. SGCS is robust to geometric misalignment while capturing inconsistencies in object presence and spatial layout.

(Peebles & Xie, 2022) as latent diffusion backbone. We use the pretrained checkpoint. **Mineworld** (Guo et al., 2025) is another interactive world model driven by a visual-action autoregressive Transformer pretrained on Minecraft. We use the pretrained checkpoint. **DIAMOND** (Alonso et al., 2024) is a diffusion-based world model built upon the UNet architecture (Ronneberger et al., 2015). We pretrain a model using our dataset from scratch. **Navigation World Model** (Bar et al., 2024) is a controllable video generation model that predicts future visual observations conditioned on past observations and navigation actions. We pretrain NWM model using our dataset from scratch. The implementation details and additional results of the four baselines are provided in the Appendix D.

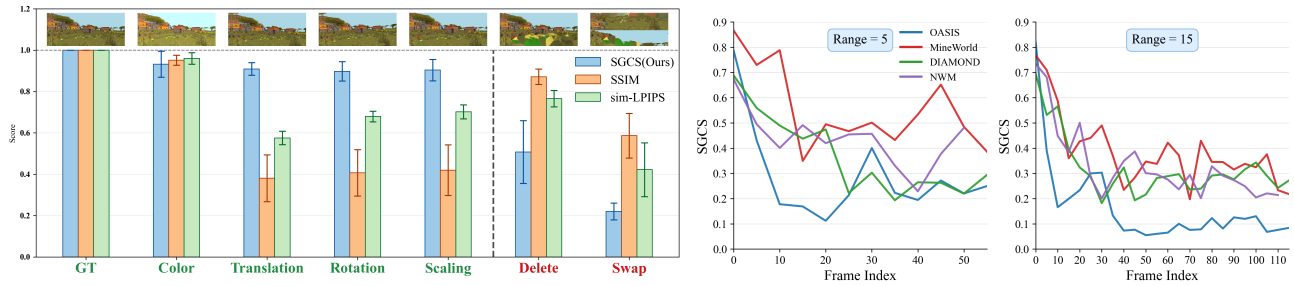
5.3. Experiment Settings

Although our dataset includes data from villages, biomes, and structures, we focus solely on village environments for training and evaluation. This decision is based on the large number of villages in Minecraft and the inherent diversity of them. We use traject number 1-16 as training set, 17-18 as validation set and 19-20 as test set.

Evaluation We evaluate each model across all navigation ranges. Due to the large scale of the test dataset, we do not perform evaluation on the entire set. Instead, we sample the first three trajectories in lexicographic order from each of the 6 villages with index 20, resulting in 18 evaluation trajectories per navigation range. Each trajectory is assessed using the metrics described in the benchmark section. We report the average performance across all 18 trajectories, as shown in Table 1. In addition, we randomly select one trajectory to present qualitative results, shown in Figure 4.

5.4. Analysis

Overall Performance Overall, all four baselines exhibit unsatisfactory performance in maintaining spatial consistency. As shown in Figure 3b and Figure 4, MineWorld and NWM are able to generate approximately spatially consistent frames during the first ∼20 frames, but fail to preserve spatial consistency in later stages. In contrast, Oasis and Diamond quickly degrade into severe visual artifacts, eventually collapsing into blurry and incoherent outputs, especially as the generation horizon increases. In these cases, the models gradually deteriorate from minor imperfections to complete visual failure over time. As shown in Figure 3b, the severe visual distortions in Oasis lead to consistently low SGCS scores. This observation indicating that SGCS aligns well with visual perception.



(a) Controlled perturbation comparison.

(b) Per-frame SGCS over time.

Figure 3. **Left: Comparison of metrics under controlled perturbations.** From left to right, Geometric Perturbations: ground truth, color change, small translation, small rotation, and scaling; Spatial Inconsistencies: object deletion and object position swapping. We compare SGCS (ours), SSIM, and sim-LPIPS. Higher values indicate better. **Right: Per-frame SGCS.** Spatial consistency degrades across all baselines as generation progresses.

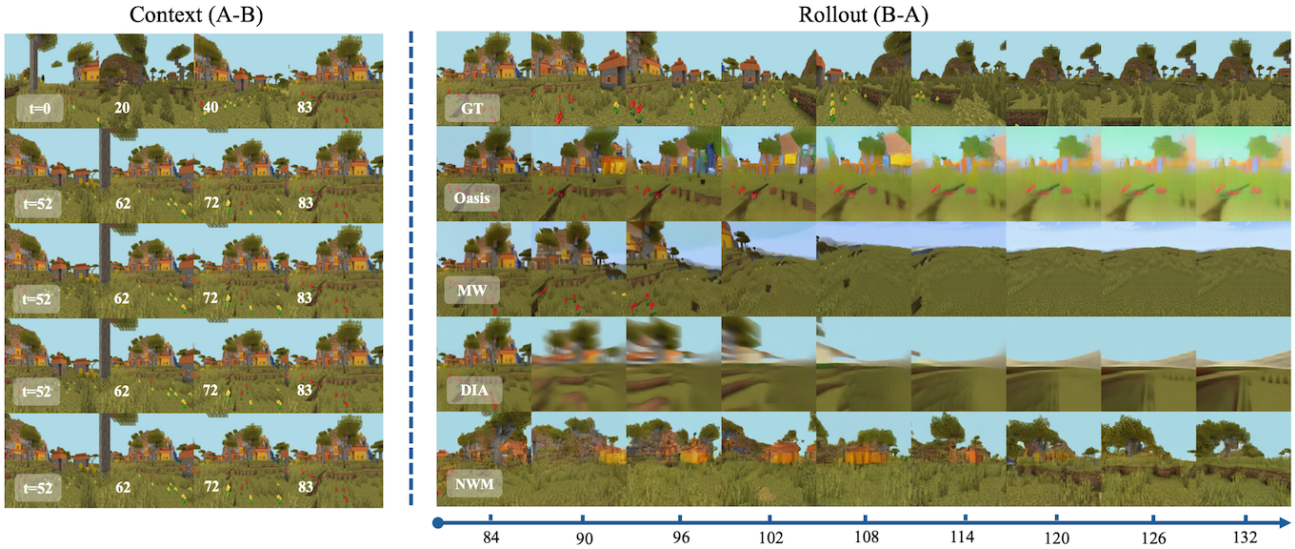


Figure 4. **Qualitative Result of Four Baselines**. Top to buttom: Ground Truth(GT), Oasis, Mineworld(MW), DIAMOND(DIA), Navigation World Model(NWM). Leftmost label "t=52" indicates the start context range accepted by each model. All models begin rollout from frame 84.

Metric Fidelity to Model Performance and Task Difficulty As shown in Table 1, our metric exhibits strong correlation with perceived visual quality. MineWorld achieves the best performance, followed by NWM and DIAMOND, while Oasis performs the worst. This ranking is consistent with human qualitative observations. Furthermore, the results reveal a clear difficulty gradient in our task design: as the navigation range increases, the task becomes more challenging, leading to a consistent decrease in SGCS.

In contrast, traditional visual quality metrics, including SSIM, LPIPS, and FVD, fail to faithfully reflect the relative performance of the baseline models and do not capture the increasing task difficulty. These observations are consistent with our earlier findings, highlighting the limitations of conventional visual similarity metrics in evaluating spatial consistency.

Outlier Analysis We observe a single outlier where MineWorld and DIAMOND achieve higher performance at ABCA type navgation, trajectory length 50 compared to length 30. This occurs because, at length 50, the task corresponds to extremely long-horizon navigation (approximately 1200 frames; see Fig. 6). Additionaly, ABCA trajectories involve more extensive exploration and wandering, often leading the agent to move beyond the village. In such scenarios, many frames encountered during navigation consist primarily of simple backgrounds, such as grass and sky. As a result, visually discriminative objects may become sparse or absent. On the other hand, even when the model fails to maintain spatial consistency, it can still generate visually similar base scenes (i.e., combinations of grass and sky; see later frames in Fig. 4). Consequently, this leads to artificially higher consistency scores.

Table 1. **Evaluation Results of Four Baselines under Different Navigation Ranges**. The ✅ icon indicates that the model was trained on our dataset, while ❌ indicates using a pretrained checkpoint. ABA and ABCA denotes the navigation type, and the suffix of model(-5,-10...) denotes the navigation range.

Model	Train	SGCS ↑		SSIM ↑		LPIPS ↓		FVD ↓	
		ABA	ABCA	ABA	ABCA	ABA	ABCA	ABA	ABCA
Oasis-5	❌	**0.26**±0.18	**0.20**±0.14	0.36±0.13	0.34±0.12	0.76±0.09	0.82±0.11	2615±1067	2583±647
Oasis-15	❌	**0.15**±0.14	**0.26**±0.17	0.37±0.12	0.38±0.14	0.82±0.08	0.81±0.10	2516±567	3146±1055
Oasis-30	❌	**0.17**±0.19	**0.07**±0.08	0.33±0.11	0.35±0.11	0.86±0.08	0.85±0.09	3131±713	3199±1000
Oasis-50	❌	**0.02**±0.03	**0.09**±0.07	0.36±0.12	0.36±0.11	0.86±0.09	0.83±0.07	3334±658	3162±1245
Mineworld-5	❌	**0.54**±0.12	**0.38**±0.10	0.31±0.09	0.32±0.10	0.73±0.05	0.72±0.07	2089±1007	1914±660
Mineworld-15	❌	**0.37**±0.23	**0.39**±0.12	0.34±0.13	0.32±0.11	0.74±0.08	0.74±0.07	2367±770	2009±921
Mineworld-30	❌	**0.32**±0.14	**0.20**±0.14	0.33±0.13	0.28±0.09	0.77±0.08	0.77±0.08	2316±945	2094±1047
Mineworld-50	❌	**0.24**±0.17	**0.40**±0.15	0.31±0.16	0.32±0.12	0.78±0.12	0.75±0.10	2077±632	2144±898
DIAMOND-5	✅	**0.34**±0.25	**0.34**±0.16	0.40±0.10	0.37±0.09	0.75±0.09	0.79±0.09	3353±1242	3336±1392
DIAMOND-15	✅	**0.29**±0.22	**0.24**±0.16	0.38±0.10	0.39±0.10	0.78±0.08	0.79±0.09	3691±937	3302±1191
DIAMOND-30	✅	**0.27**±0.23	**0.11**±0.09	0.37±0.10	0.35±0.10	0.81±0.07	0.81±0.08	3708±1243	3473±1355
DIAMOND-50	✅	**0.12**±0.15	**0.17**±0.13	0.37±0.10	0.38±0.09	0.83±0.09	0.81±0.08	3249±833	2994±906
NWM-5	✅	**0.40**±0.27	**0.37**±0.16	0.33±0.11	0.31±0.09	0.64±0.05	0.67±0.05	1950±380	2240±664
NWM-15	✅	**0.33**±0.20	**0.33**±0.16	0.30±0.12	0.33±0.12	0.67±0.03	0.65±0.05	2132±916	2338±1010
NWM-30	✅	**0.31**±0.19	**0.29**±0.14	0.32±0.11	0.30±0.11	0.69±0.04	0.71±0.03	1893±1047	2437±429
NWM-50	✅	**0.26**±0.18	**0.27**±0.20	0.28±0.13	0.33±0.11	0.72±0.08	0.65±0.04	2715±883	1537±415

This exception does not affect our overall conclusions. Rather, it further underscores the importance of the "visually distinguishable" principle in dataset construction. At the same time, it points to a promising direction for future work, such as incorporating different weighting across trajectory segments to better account for varying scene complexity.

6. Limitations and Future Work

Exploration of More Advanced SGCS Variants. In this work, we adopt a relatively simple object-level matching strategy. However, several promising extensions may further improve the metric. First, different segments of a trajectory could be assigned different weights, allowing the metric to emphasize visually informative or spatially challenging regions. Second, beyond object-level matching, future versions of SGCS could explicitly model and compare relational edges between objects, enabling more structured scene-level consistency evaluation. Finally, for objects belonging to the same category, consistency could also be evaluated at the shape or internal structure level, either through finer-grained segmentation or more detailed matching strategies.

Lack of data from real-world environments: Currently, our data is only collected in the Minecraft environment. Given the gap between Minecraft and real-world environments, Out-of-Distribution (OOD) scenarios are still a potential concern. Future works that focus on collecting data from other environments, such as real-world simulations or different game environments, will be a great supplement to improve the generalizability of the dataset.

Static structures in the dataset: The structures in our dataset (such as houses and terrain) are static, and we intentionally excluded moving objects like mobs. This decision was made to simplify the task of modeling spatial consistency. However, in real-world environments, the positions of vehicles, people, and other dynamic objects change over time. Maintaining spatial consistency in such dynamic environments is a more challenging task and a promising direction for future exploration. We leave this task for future work.

References

Alonso, E., Jelley, A., Micheli, V., Kanervisto, A., Storkey, A., Pearce, T., and Fleuret, F. Diffusion for world modeling: Visual details matter in atari. In *Thirty-eighth Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.12399.

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos, 2022. URL https://arxiv.org/abs/2206.11795.

Bar, A., Zhou, G., Tran, D., Darrell, T., and LeCun, Y. Nav-

igation world models, 2024. URL https://arxiv.org/abs/2412.03572.

Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., and Ramesh, A. Video generation models as world simulators, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Bruce, J., Dennis, M., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., Aytar, Y., Bechtle, S., Behbahani, F., Chan, S., Heess, N., Gonzalez, L., Osindero, S., Ozair, S., Reed, S., Zhang, J., Zolna, K., Clune, J., de Freitas, N., Singh, S., and Rocktäschel, T. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

Cai, S., Zhang, B., Wang, Z., Ma, X., Liu, A., and Liang, Y. Groot: Learning to follow instructions by watching gameplay videos, 2023.

Cai, S., Zhang, B., Wang, Z., Lin, H., Ma, X., Liu, A., and Liang, Y. Groot-2: Weakly supervised multi-modal instruction following agents, 2024. URL https://arxiv.org/abs/2412.10410.

Cai, S., Mu, Z., Liu, A., and Liang, Y. Rocket-2: Steering visuomotor policy via cross-view goal alignment, 2025a. URL https://arxiv.org/abs/2503.02505.

Cai, S., Wang, Z., Lian, K., Mu, Z., Ma, X., Liu, A., and Liang, Y. Rocket-1: Mastering open-world interaction with visual-temporal context prompting, 2025b. URL https://arxiv.org/abs/2410.17856.

Carion, N., Gustafson, L., Hu, Y.-T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K. V., Khedr, H., Huang, A., Lei, J., Ma, T., Guo, B., Kalla, A., Marks, M., Greer, J., Wang, M., Sun, P., Rädle, R., Afouras, T., Mavroudi, E., Xu, K., Wu, T.-H., Zhou, Y., Momeni, L., Hazra, R., Ding, S., Vaze, S., Porcher, F., Li, F., Li, S., Kamath, A., Cheng, H. K., Dollár, P., Ravi, N., Saenko, K., Zhang, P., and Feichtenhofer, C. Sam 3: Segment anything with concepts, 2026. URL https://arxiv.org/abs/2511.16719.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset, 2018. URL https://arxiv.org/abs/1705.07750.

Chen, B., Monso, D. M., Du, Y., Simchowitz, M., Tedrake, R., and Sitzmann, V. Diffusion forcing: Next-token prediction meets full-sequence diffusion, 2024. URL https://arxiv.org/abs/2407.01392.

Contributors, P. mineflayer: Create minecraft bots with a stable, high-level javascript api. https://github.com/PrismarineJS/mineflayer, 2013–2024.

Decart, Quevedo, J., McIntyre, Q., Campbell, S., Chen, X., and Wachen, R. Oasis: A universe in a transformer, 2024. URL https://oasis-model.github.io/.

Ding, J., Zhang, Y., Shang, Y., Zhang, Y., Zong, Z., Feng, J., Yuan, Y., Su, H., Li, N., Sukiennik, N., Xu, F., and Li, Y. Understanding world or predicting future? a comprehensive survey of world models, 2024. URL https://arxiv.org/abs/2411.14499.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Du, Y., Yang, M., Florence, P., Xia, F., Wahid, A., Ichter, B., Sermanet, P., Yu, T., Abbeel, P., Tenenbaum, J. B., Kaelbling, L., Zeng, A., and Tompson, J. Video language planning, 2023. URL https://arxiv.org/abs/2310.10625.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

Gao, S., Yang, J., Chen, L., Chitta, K., Qiu, Y., Geiger, A., Zhang, J., and Li, H. Vista: A generalizable driving world model with high fidelity and versatile controllability, 2024. URL https://arxiv.org/abs/2405.17398.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

Guo, J., Ye, Y., He, T., Wu, H., Jiang, Y., Pearce, T., and Bian, J. Mineworld: a real-time and open-source interactive world model on minecraft. *arXiv preprint arXiv:2504.08388*, 2025.

Gupta, S., Tolani, V., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. Cognitive mapping and planning for visual navigation, 2019. URL https://arxiv.org/abs/1702.03920.

Guss, W. H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., and Salakhutdinov, R. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models, 2022. URL https://arxiv.org/abs/2010.02193.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Hong, Y., Liu, B., Wu, M., Zhai, Y., Chang, K.-W., Li, L., Lin, K., Lin, C.-C., Wang, J., Yang, Z., Wu, Y., and Wang, L. Slowfast-vgen: Slow-fast learning for action-driven long video generation, 2024. URL https://arxiv.org/abs/2410.23277.

Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., and Corrado, G. Gaia-1: A generative world model for autonomous driving, 2023. URL https://arxiv.org/abs/2309.17080.

Kim, N., Kwon, O., Yoo, H., Choi, Y., Park, J., and Oh, S. Topological semantic graph memory for image-goal navigation, 2022. URL https://arxiv.org/abs/2209.08274.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

Lifshitz, S., Paster, K., Chan, H., Ba, J., and McIlraith, S. Steve-1: A generative model for text-to-behavior in minecraft, 2024. URL https://arxiv.org/abs/2306.00937.

Menapace, W., Lathuilière, S., Tulyakov, S., Siarohin, A., and Ricci, E. Playable video generation, 2021. URL https://arxiv.org/abs/2101.12195.

Micheli, V., Alonso, E., and Fleuret, F. Transformers are sample-efficient world models, 2023. URL https://arxiv.org/abs/2209.00588.

Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. Action-conditional video prediction using deep networks in atari games, 2015. URL https://arxiv.org/abs/1507.08750.

Parisotto, E. and Salakhutdinov, R. Neural map: Structured memory for deep reinforcement learning, 2017. URL https://arxiv.org/abs/1702.08360.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Robine, J., Höftmann, M., Uelwer, T., and Harmeling, S. Transformer-based world models are happy with 100k interactions, 2023. URL https://arxiv.org/abs/2303.07109.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Savinov, N., Dosovitskiy, A., and Koltun, V. Semi-parametric topological memory for navigation, 2018. URL https://arxiv.org/abs/1803.00653.

Song, K., Chen, B., Simchowitz, M., Du, Y., Tedrake, R., and Sitzmann, V. History-guided video diffusion, 2025. URL https://arxiv.org/abs/2502.06764.

Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. Towards accurate generative models of video: A new metric & challenges, 2019. URL https://arxiv.org/abs/1812.01717.

Valevski, D., Leviathan, Y., Arar, M., and Fruchter, S. Diffusion models are real-time game engines, 2024. URL https://arxiv.org/abs/2408.14837.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang*, Q., Zhang*, Y., Holynski, A., Efros, A. A., and Kanazawa, A. Continuous 3d perception model with persistent state. In *CVPR*, 2025.

Wang, X., Zhu, Z., Huang, G., Chen, X., Zhu, J., and Lu, J. Drivedreamer: Towards real-world-driven world models for autonomous driving, 2023a. URL https://arxiv.org/abs/2309.09777.

Wang, Z., Li, X., Yang, J., Liu, Y., and Jiang, S. Gridmm: Grid memory map for vision-and-language navigation, 2023b. URL https://arxiv.org/abs/2307.12907.

Wu, P., Escontrela, A., Hafner, D., Goldberg, K., and Abbeel, P. Daydreamer: World models for physical robot learning, 2022. URL https://arxiv.org/abs/2206.14176.

Yang, M., Du, Y., Ghasemipour, K., Tompson, J., Schuurmans, D., and Abbeel, P. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

Yang, S., Walker, J., Parker-Holder, J., Du, Y., Bruce, J., Barreto, A., Abbeel, P., and Schuurmans, D. Video as the new language for real-world decision making, 2024. URL https://arxiv.org/abs/2402.17139.

Yang, Y., Yang, H., Zhou, J., Chen, P., Zhang, H., Du, Y., and Gan, C. 3d-mem: 3d scene memory for embodied exploration and reasoning, 2025. URL https://arxiv.org/abs/2411.17735.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zhao, G., Lian, K., Lin, H., Fu, H., Fu, Q., Cai, S., Wang, Z., and Liang, Y. Optimizing latent goal by learning from trajectory preference, 2024. URL https://arxiv.org/abs/2412.02125.

Zhu, Z., Wang, X., Zhao, W., Min, C., Deng, N., Dou, M., Wang, Y., Shi, B., Wang, K., Zhang, C., You, Y., Zhang, Z., Zhao, D., Xiao, L., Zhao, J., Lu, J., and Huang, G. Is sora a world simulator? a comprehensive survey on general world models and beyond. *arXiv preprint arXiv:2405.03520*, 2024.

Appendix Overview

Our appendix is organized as follows:

- **Appendix A** describes the platform setup for our data collection, including the Minecraft environment setup and three javascript plugins we modified to manipulate the bot and collect data.
- **Appendix B** presents detials of our dataset, including B.2 data storage format, B.3 statistical information and B.4 visualization of our LOOPNAV dataset.
- **Appendix C**
- **Appendix D** presents the detailed experimental settings and additional experimental results:

  D.1: Synthetic Perturbation Details. **D.2**: Open-oasis, **D.3**: Mineworld, **D.4**: DIAMOND, **D.5**: Navigation World Model.

A. Platform Setup

We set up a local Minecraft server and use the Mineflayer platform to control the agent and collect trajectory data. For path planning, we employ the Mineflayer Pathfinder plugin, which computes shortest paths between waypoints using the A* algorithm. We use the Prismarine Viewer to render and visualize the agent's behavior and collected observations. To better support learning from the collected data and simulate human-like control dynamics, we make several key modifications to the default behavior of these plugins. These choices define some of the core characteristics of our dataset:

Restricted action space: The agent is limited to using only three actions during navigation — forward, jump, and camera rotation — closely matching the action primitives of a human player in Minecraft. See Figure 5 for demostrations.

Sequential action execution: At any given time step, only a single action is allowed. That is, the agent cannot rotate the camera and move forward simultaneously. This avoids entangled motion patterns and improves the clarity of spatial transitions in the data.

Smooth camera control: We impose a maximum angular velocity on camera rotations to prevent abrupt viewpoint changes between frames, ensuring more stable visual continuity for training.

Removal of irrelevant elements: To focus the dataset on pure navigation, we disabled all entity spawning (e.g., mobs), and removed UI elements such as the hotbar and hands, eliminating visual distractions and ensuring task purity.

A.1. Minecraft

Minecraft is an open-world environment characterized by diverse terrains and rich interaction dynamics. We selected Minecraft for the following reasons:

- **Rich environmental diversity**: Minecraft worlds are procedurally generated using random seeds, featuring a wide range of biomes and uniquely structured villages, enabling diverse and non-repetitive environments.
- **Extensive community resources**: Minecraft has a mature modding ecosystem that allows us to incorporate a wide variety of custom elements and tasks. In addition, there is a large amount of publicly available Minecraft gameplay footage on platforms like YouTube, which could be leveraged for future large-scale pretraining.
- **Rising research interest**: An increasing number of recent works have begun to study world modeling and agent learning in Minecraft, making it a timely and

relevant platform for evaluating spatial consistency and memory mechanisms.

- **Efficient simulation**: Compared to real-world data collection, Minecraft enables faster, cheaper, and highly parallelizable simulation, making it well-suited for large-scale controlled data generation.

We established a local Mineworld server running **Java Edition** version **1.16.5**, utilizing the seed value of **42**. The server was configured in survival mode with the difficulty set to peaceful, and mob spawning was disabled to eliminate the presence of animals. This configuration was implemented to prevent interference from mobs during trajectory-related experiments.

We utilized the Chunkbase website [3] to locate various villages and biomes. A total of 120 villages were collected—comprising 6 types, with 20 samples each—along with locations of 18 distinct biomes and 8 structures. The agent was teleported to each initial position using Minecraft's teleport command before commencing navigation.

A.2. Mineflayer

Minefayer [4] is a powerful open-source JavaScript API designed for flexibly controlling in-game bots within Minecraft. It interacts with the Minecraft server by continuously reading block information around the bot based on its current x, y, and z coordinates, which is loaded locally. Users are allowed to specify actions for the bot to perform. Mineflayer simulates the bot's movement physically, computes its resulting state and position after each action, and subsequently sends updated coordinates (x, y, z), yaw, and pitch values to the server. Benefiting from this flexible interaction mechanism, diverse action policies can be developed to enable the agent to accomplish various tasks.

We introduced a modification to Mineflayer by constraining the horizontal rotation speed per time step, such that each action can result in a **maximum rotation of 0.1 radians**. This adjustment ensures smoother and more gradual viewpoint transitions.

A.3. Mineflayer-pathfinder

The Mineflayer Pathfinder [5] is an application built on top of Mineflayer. It is capable of planning a path from the bot's current position to a target coordinate (x, y, z) using the A* algorithm. The bot is then controlled through a series of actions to navigate toward the destination.

In the original implementation, camera rotation and movement were coupled within the same action. In our modified version, at any given time step, **only a single action is allowed**. That is, the agent cannot rotate the camera and move forward simultaneously. This avoids entangled motion patterns and improves the clarity of spatial transitions in the data.

Furthermore, the original Mineflayer Pathfinder often exhibited sharp turns and other non-smooth behaviors during navigation, particularly causing rapid camera jitter near jagged block edges. To mitigate this issue, we prevent the bot from moving along the edges of 1×1 blocks, thereby eliminating such undesirable motion artifacts.

A.4. Prismarine-viewer

Prismarine Viewer [6] is a complementary plugin designed to work alongside Mineflayer, capable of rendering the Minecraft environment from either a first-person or third-person perspective. In this sense, the combination of Mineflayer and Prismarine Viewer effectively constitutes a functional Minecraft client. We employed Prismarine Viewer to render and generate gameplay visuals.

The rendering logic of the viewer operates by periodically capturing the current coordinates (x, y, z) yaw pitch and rendering the scene at a fixed time interval. We set this sampling frequency to 50 Hz, matching both Minecraft's default refresh rate and the action-sending frequency. As a result, each frame is rendered every 0.02 seconds, synchronized with the execution of a new action.

B. Dataset details and Visualization

B.1. Overview

A total of 120 distinct villages were collected, comprising 6 types with 20 instances each. Our experiments were primarily conducted on this dataset of 120 villages. In addition, 18 different biomes and 8 types of locations were also gathered. Detailed information regarding the specific types of villages, biomes, and locations is provided in the table 2 below.

For each location, we collected two types of trajectories: **{ABA, ABCA}**. For each type, trajectories were generated with four different navigation ranges: **5, 15, 30, and 50**. Each length comprises **20** distinct trajectories with varied start and end points.

This resulted in a total of 19,200 trajectories collected across all villages. With an average length of approximately 1,000 frames per trajectory, the dataset contains around **19.2 million** frames, equivalent to roughly **250 hours** of gameplay.

[3] http://www.chunkbase.com/
[4] https://github.com/PrismarineJS/mineflayer
[5] https://github.com/PrismarineJS/mineflayer-pathfinder
[6] https://github.com/PrismarineJS/prismarine-viewer

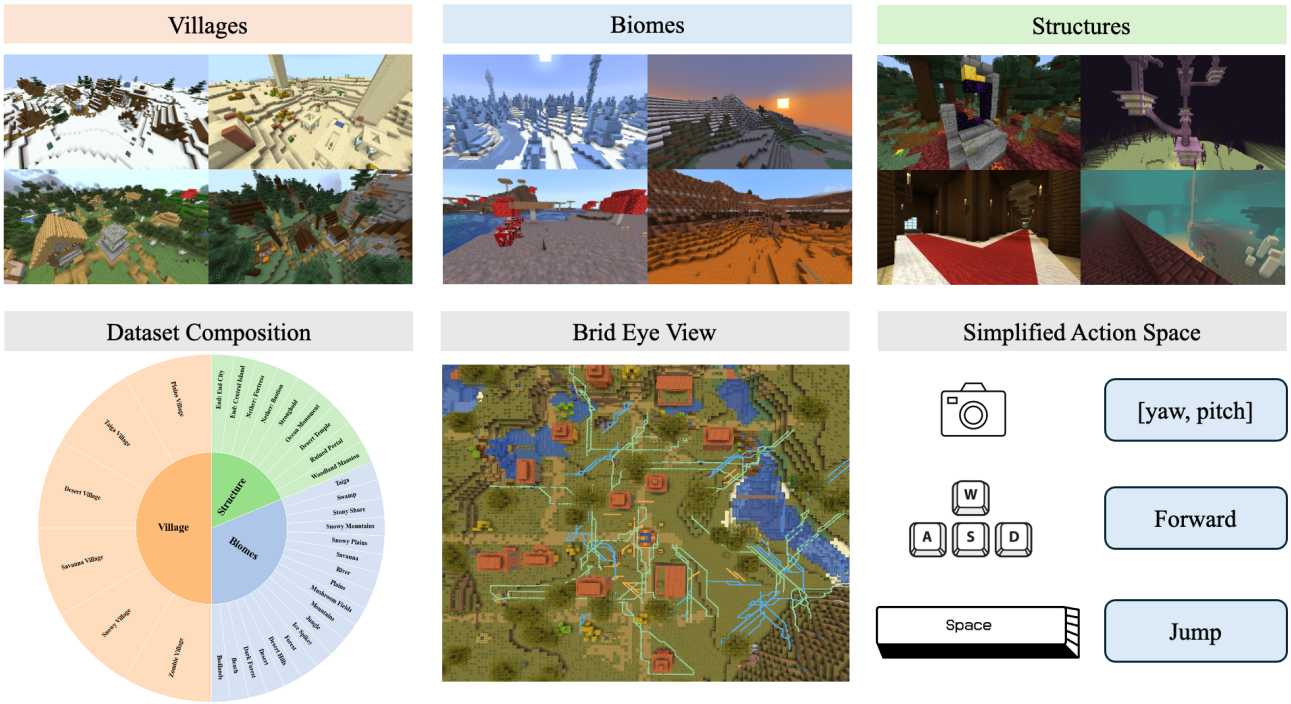


Figure 5. **Overview of the Minecraft Elements**. Top row, left to right: Examples of villages, biomes, and structures in Minecraft. Bottom row, left to right: composition of sampling locations in our dataset; a bird-eye view of real ABA-type exploration trajectories (display steps 5, 15, and 30 for simplicity); Simplified Action Space used for data collection and agent interaction.

Table 2. Minecraft locations: Villages, Biomes, and Structures

Village	Biome		Structure
Plains Village	Badlands	Plains	Woodland Mansion
Savanna Village	Beach	River	Ruined Portal
Snowy Village	Dark Forest	Savanna	Desert Temple
Taiga Village	Desert	Snowy Plains	Stronghold
Desert Village	Desert Hills	Snowy Mountains	Nether Bastion
Forest Village	Forest	Stony Shore	Nether Fortress
Zombie Village	Ice Spikes	Swamp	End Mainland
	Jungle	Taiga	End City
	Mushroom Fields	Mountains	

For experimental purposes, the villages were partitioned into training, validation, and test sets as follows: villages 1-16 were used for training, villages 17–18 for validation, and villages 19–20 for testing. The trajectories were not shuffled randomly across splits to prevent the model from memorizing structural features seen during training. Instead, this partitioning strategy encourages the model to reconstruct environmental structures from contextual information, as each village has a unique layout.

B.2. Data Format

The smallest unit in our dataset consists of a pair of files: an **.avi video file** and a corresponding **.json file**, both of equal length. The .avi file stores the visual observations, while the .json file contains the associated state and action data. To avoid introducing inter-frame dependencies, we use MJPG (Motion JPEG) compression instead of H.264. Each frame is stored independently, and the resolution of the images is 640×360 at BGR format. The value of each BRG pixel is from [0,255]. The videos are recorded at 20 frames per second (FPS). This frequency aligns with the physical tick rate of Minecraft, ensuring consistent synchronization between control and simulation.

One step of recoreded trajectories in .json have following keys:

- x: current x coordinate, rounded to three decimal

places.

- y: current y coordinate, namely hight, rounded to three decimal places.
- z: current z coordinate, rounded to three decimal places.
- yaw: the agent's horizontal viewing angle, measured in radians, ranging from $-\pi$ to π, where 0 indicates the agent is facing the positive z-axis.
- pitch: the agent's vertical viewing angle, measured in radians, ranging from $-\pi/2$ to $\pi/2$, where 0 indicates the agent is looking straight ahead (parallel to the ground plane).
- action: a dictionary with three possible keys: forward, jump, and camera. The values of forward and jump are booleans indicating whether the corresponding action is executed. The camera key holds a tuple [yaw, pitch], representing the **change** in the agent's viewing angle. Our sampling strategy ensures that these three actions do not occur simultaneously.
- goal: a dictionary containing the target's coordinates, with keys x and z representing the target position on the x- and z-axes, respectively.
- frame count: the index of the current frame. Due to rendering initialization, the first 20 frames are skipped, so frame count starts at 20. Since we ensure that the entire trajectory is recorded properly, the actual frame index within the trajectory can be obtained by simply subtracting 20.
- extra info: seed, location, navigation type $\in \{ABA, ABCA\}$, navigation range $\in \{5, 15, 30, 50\}$.

B.3. Statistical Information

B.3.1. Trajectory Length

To better illustrate the scale and difficulty gradient of the dataset, the distribution of trajectory lengths is presented in the Figure 6 and Table 3 below.

From the perspective of sequence length, even for the shortest navigation range (i.e., range = 5), the average trajectory length reaches as high as 180 frames. This implies that during evaluation, models must be capable of attending to visual inputs from over 100 frames ago. However, most current world models are limited to a history (context window) of only 32 frames, indicating that the required temporal context in our setting substantially exceeds the capacity of existing models.

In terms of distribution, different navigation ranges exhibit minimal overlap in their length distributions, which supports the validity of our difficulty curriculum design.

B.3.2. Trajectory Variance

Since both the $A \rightarrow B$ and $B \rightarrow A$ paths are generated using A* search, a natural question arises: Are the forward ($A \rightarrow B$) and backward ($B \rightarrow A$) paths different? If so, how significant is the discrepancy?

To quantify this difference, we compute the area enclosed by this loop to measure the spatial deviation between the two paths. Additionally, we normalize this area by the trajectory length to obtain a per-step deviation metric, which reflects the average divergence between forward and backward paths. The results are in the Table 4.

The results indicate that the $A \rightarrow B$ and $B \rightarrow A$ paths are not perfectly identical, but the deviation remains within an acceptable range. For the longest navigation distance of 50 grids, the average enclosed area between the two paths is 87.59, while the average total trajectory length is 108.39. If we approximate the path as forming a rectangle, this translates to about 1.62 grid units of deviation per unit path length on average. For shorter navigation distances (5, 15, 30 grids), the deviation is smaller.

Given that the navigation is driven by the A* algorithm, we believe this level of deviation is acceptable and does not significantly compromise spatial consistency or view reconstruction. Moreover, our bird's-eye visualizations that qualitatively illustrate forward and return paths.

B.4. Dataset Visualization

We illustrate in Figure 7 two trajectories extracted from datasets with a navigation type of ABA and a navigation range of **5**. The upper subfigure corresponds to plains village (village ID:20, trajectory ID:05-10_14-09-44), while the lower subfigure is taken from snowy village (village ID:20, trajectory ID:05-10_17-46-35).

In the plains village, we observe that **frame 1 is visually similar to frame 129, and frame 41 is similar to frame 153.** In the snowy village, **frame 17 resembles frame 129, and frame 25 is similar to frame 153.** These observations suggest that, in order for a world model to accurately reconstruct future scenes, it must retain information from more than 100 frames earlier, and be capable of leveraging spatial context to infer and restore visual content.

To provide an intuitive visualization of the trajectories in our dataset, we present bird's-eye views of real trajectories. In Figure 8, the location is taken from plains village (village ID: 2). The left panel shows a trajectory of type ABA, while the right panel displays a trajectory of type ABCA. Yellow, blue, and green indicate navigation ranges of 5, 15, and 30, respectively. For visual clarity, trajectories with a navigation range of 50 are omitted.

Table 3. Mean Trajectory Frames ($\pm$ Std) for Different Trajectory Types

Trajectory Type	Range 5	Range 15	Range 30	Range 50
ABA	180.5 ± 30.4	356.5 ± 71.6	627.1 ± 130.3	967.8 ± 195.4
ABCA	251.0 ± 47.1	544.7 ± 113.1	968.1 ± 203.7	1362.6 ± 260.1

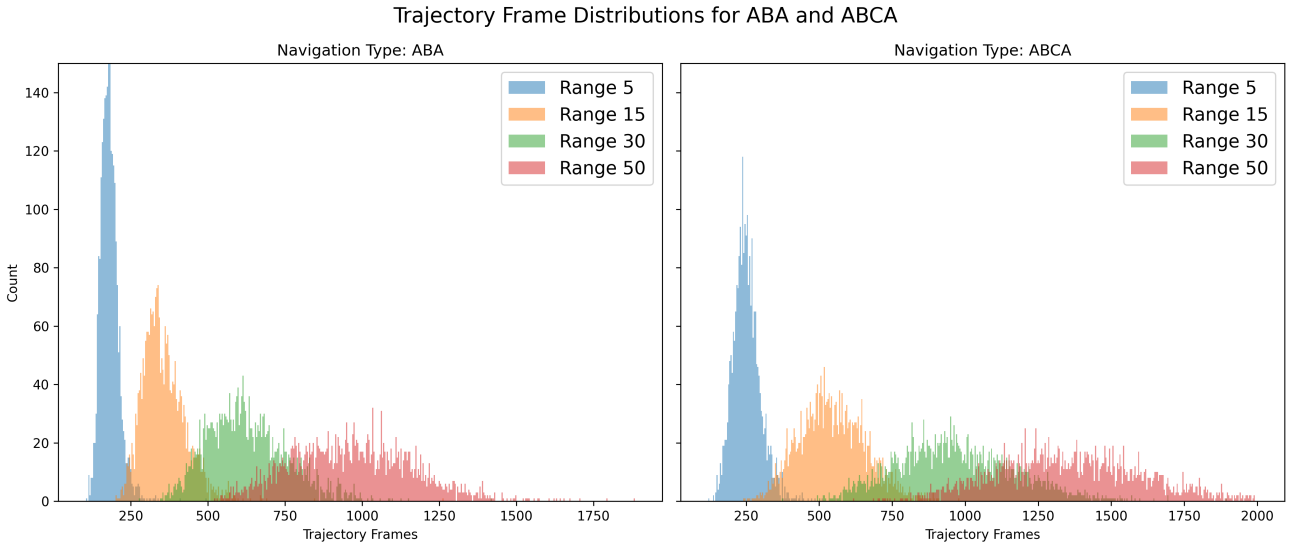


Figure 6. **Trajectory Frame Distribution of Navgation Type ABA and ABCA**

C. Evaluation Pipeline Details

C.1. SGCS Computation Details

We describe the full evaluation pipeline used to compute the Scene Graph Consistency Score (SGCS) between two videos. The implementation follows a frame-wise comparison protocol with object-level matching and category-wise aggregation.

Frame Sampling and Alignment. Given two videos A and B, we first sample frames with a fixed stride s (default $s = 5$). For each sampled frame index t, we extract the corresponding frame pair (I_t^A, I_t^B). If the two frames have different spatial resolutions, they are resized to a common resolution defined by the minimum height and width across the pair.

Object Segmentation via Text Prompts. For each frame, we extract object masks using a text-conditioned segmentation model (SAM3). Specifically, we define a fixed set of semantic categories (we use *building, tree, flower, grass, path, water, sky, sand, dirt, snow* for Minecraft), and for each category, we query the model to obtain a set of binary masks. Masks with low confidence (threshold 0.5 by default) are discarded. This produces, for each category c, a set of masks $\mathcal{M}_c^A$ and $\mathcal{M}_c^B$ for the two frames.

Mask Representation. Each mask is converted into a compact geometric representation consisting of:

- **Centroid:** computed as the mean of foreground pixel coordinates;
- **Area:** computed as the number of foreground pixels.

Masks with zero area are ignored. This representation allows efficient spatial comparison while remaining robust to shape irregularities.

Instance Matching. For each category c, we match instances across the two frames using centroid distance. Let $\{x_i^A\}$ and $\{x_j^B\}$ denote the centroids. We compute pairwise Euclidean distances and normalize them by the image diagonal $D = \sqrt{H^2 + W^2}$. A pair is considered a valid match if:

$$\|x_i^A - x_j^B\|_2 < \tau \cdot D,$$

where $\tau = 0.1$ by default. We then perform bipartite matching using the Hungarian algorithm to maximize the number of valid matches.

Category-Level Score. For each category c, let m and n denote the number of detected instances in the two frames, and p the number of matched pairs. We define the category-level consistency score as:

$$S_c = \frac{2p}{m + n}$$

Table 4. Deviation Between $A \to B$ and $B \to A$ Paths

Metric	Range 5	Range 15	Range 30	Range 50
Enclosed Area	1.83 ± 2.20	10.95 ± 12.91	39.20 ± 45.20	87.59 ± 97.77
Normalized Deviation	0.35 ± 0.07	0.67 ± 0.13	1.19 ± 0.17	1.62 ± 0.20

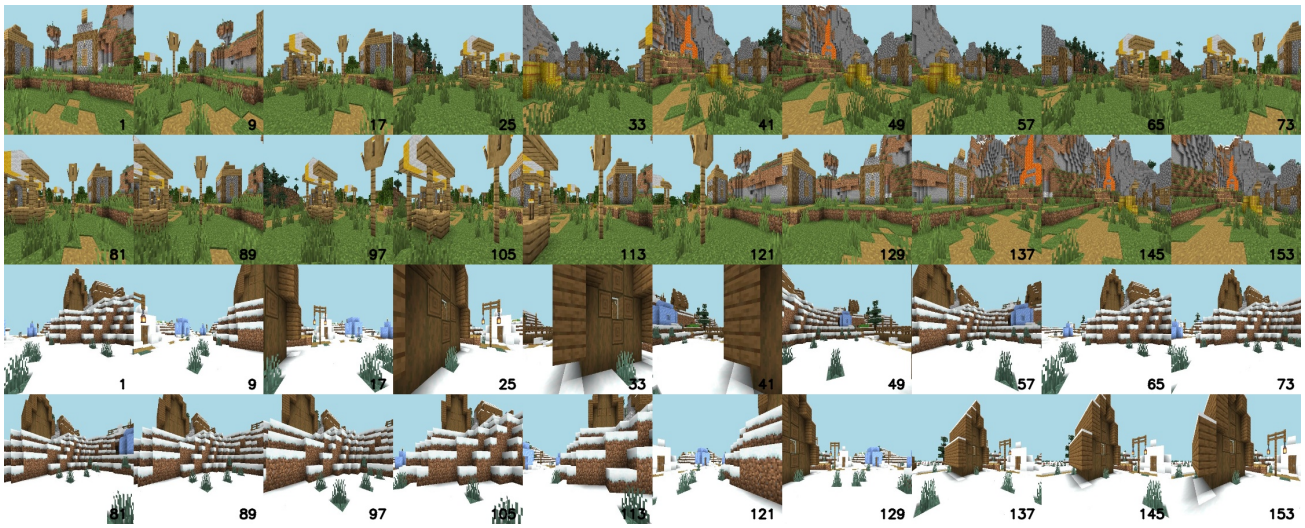


Figure 7. Demonstration of two LOOPNAV Trajectories. Upper half: Plains village. Lower Half: Snowy village. Black number indicates coresponding frame number.

This formulation corresponds to an instance-level F1 (Dice) score, penalizing both missing and mismatched objects.

Area-Weighted Aggregation. To account for object importance, we weight each category by its spatial extent:

$$w_c = \frac{\text{area}_c^A + \text{area}_c^B}{2}.$$

The frame-level SGCS is computed as:

$$\text{SGCS}_t = \frac{\sum_c w_c S_c}{\sum_c w_c}.$$

Video-Level Aggregation. Finally, we compute the video-level SGCS by averaging over all sampled frames:

$$\text{SGCS}_{video} = \frac{1}{T}\sum_{t=1}^{T} \text{SGCS}_t.$$

In dataset-level evaluation, we further report the mean and standard deviation across trajectories, where each trajectory-level score is obtained via temporal averaging.

C.2. Traditonal Evaluation Metrics

Fréchet Video Distance (FVD) We adopt Fréchet Video Distance (FVD) (Unterthiner et al., 2019) as a primary metric. FVD computes the distance between the distribution of real and generated video features extracted by an Inflated 3D ConvNet (I3D) (Carreira & Zisserman, 2018), capturing both spatial and temporal statistics at a high level.

Learned Perceptual Image Patch Similarity (LPIPS) We use LPIPS (Zhang et al., 2018) to evaluate the perceptual similarity between generated and ground truth frames. LPIPS leverages deep features from pretrained networks (e.g., VGG) to assess semantic-level differences, which aligns better with human judgment than pixel-wise metrics. It is robust to minor texture or color variations while remaining sensitive to object structure and layout.

Structural Similarity Index Measure (SSIM) SSIM is employed to complement LPIPS by quantifying structural fidelity and luminance similarity between generated and reference frames. While it is more sensitive to low-level details than LPIPS, it still emphasizes perceptual structure over exact pixel match.

C.3. Context Coverage in ABCA Trajectories

For longer $A \to B \to C \to A$ trajectories, we evaluate only the $C \to A$ segment, treating $A \to B \to C$ as the exploration context. It is worth noting that in these trajectories, the return path from $C \to A$ is not guaranteed to be fully covered by the earlier observations from exploration context. However, in practice, we constrain the exploration range such that the majority of $C \to A$ trajectories traverse areas that have been previously observed, albeit from different viewpoints or at different times(Figure 8 for bird-eye view

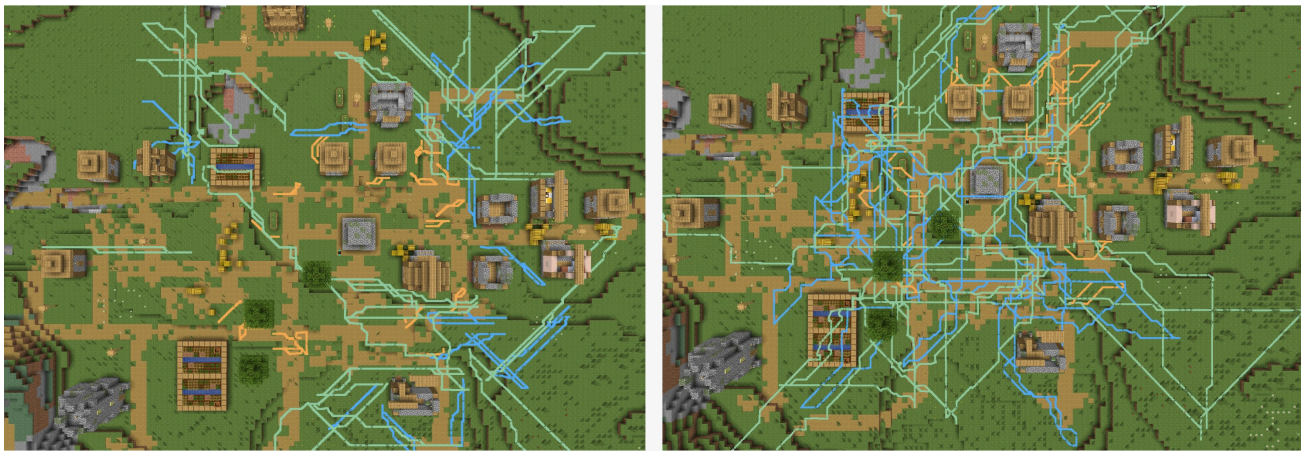


Figure 8. Demonstration of bird eye views for real dataset. Left: ABA type trajectories. Right: ABCA type trajectories. Yellow, blue, green indicate range 5, 15, 30 respectively.

illustrations). As a result, the model typically has access to sufficient contextual information to support the reconstruction of the return path, making our evaluation of spatial consistency meaningful.

D. Experiment Settings and Results

We provide a detailed description of the experimental settings to facilitate reproduction. All inference and training experiments were conducted on NVIDIA GeForce RTX 4090 GPUs and A800-80GB, with a total compute time of approximately 1000 GPU hours respectively.

D.1. Synthetic Perturbation Details

This experiment is conducted on the navigation dataset with navigation type ABA and sequence length 5, covering all 6 distinct villages. Since our evaluation applies synthetic perturbations directly to ground-truth frames, the navigation type and trajectory length do not significantly affect the results. For reporting variability, we compute the standard deviation over video-level scores, where each video-level score is obtained by averaging frame-level metrics within a single video. We introduce all the synthetic perturbation details below, and the demonstration is in Figure 9.

Color change. For the color-change perturbation, we modify only the appearance of the input frame while preserving its spatial layout. Specifically, each RGB frame is first adjusted by applying a contrast multiplier of 1.12 and adding a brightness offset of 18.0 to all pixel values. The result is clipped to the valid range $[0, 255]$. We then convert the image to HSV space and further increase the saturation channel by a factor of 1.15, followed by conversion back to RGB. This perturbation changes low-level color and illumination statistics without modifying object geometry or object layout.

Small translation. For the small-translation perturbation, we apply a global 2D shift to the entire frame. The horizontal and vertical displacements are set to 4% of the frame width and height, respectively. Formally, for a frame of size $W \times H$, the translation offsets are $\Delta x = 0.04W$ and $\Delta y = 0.04H$, rounded to integer pixel values with a minimum displacement of one pixel. The translated frame is generated using bilinear interpolation, and missing boundary regions are filled using reflection padding. This perturbation introduces pixel-level misalignment while preserving the relative spatial arrangement of objects.

Small rotation. For the small-rotation perturbation, we rotate the entire frame around its image center by 5° while keeping the image resolution unchanged. The transformation is implemented using an affine rotation matrix with scale fixed to 1.0. Bilinear interpolation is used for resampling, and reflection padding is used to fill boundary regions. Since all objects undergo the same rigid transformation, the global image alignment changes, but the underlying scene structure and object relationships are largely preserved.

Scaling via crop-and-resize. For the scaling perturbation, we simulate a zoom-in transformation using center crop followed by resizing. Specifically, we take a centered crop covering 90% of the original frame height and width, and then resize the cropped region back to the original resolution. This produces a mild global scaling effect while maintaining the main scene layout. Because the transformation affects the whole frame consistently, it should not substantially alter object-level spatial consistency, although pixel-level similarity metrics may be affected by the resampling and scale change.

Object deletion. For the object-deletion perturbation, we first obtain object masks using SAM with the predefined

Figure 9. Synthetic Perturbation Demonstrations.

text categories. Candidate object masks are filtered by area: masks smaller than 0.2% of the image area or larger than 40% of the image area are discarded. Duplicate masks are removed using an IoU threshold of 0.85. Among the remaining candidates, the largest object mask is selected for deletion. The selected mask is dilated with a 5×5 kernel and removed using Telea inpainting. This perturbation directly violates spatial consistency by removing a detected object from the scene while keeping the surrounding visual content plausible.

Object position swapping. For the object-swapping perturbation, we again use SAM-derived object masks filtered with the same area constraints: minimum area fraction 0.002, maximum area fraction 0.4, and duplicate removal with IoU threshold 0.85. If at least two valid objects are found, we select the two largest object candidates. The union of their masks is first removed from the image using inpainting. Then, each object patch is pasted into the other object's bounding box. During pasting, the source patch and source mask are resized to match the target bounding box, and only pixels inside the resized mask are copied. This perturbation preserves the presence of objects but changes their spatial locations, thereby creating a structural inconsistency in object layout.

D.2. Open Oasis

D.2.1. Oasis Settings

We use the open-sourced Oasis-500M model for inference only. We do not train the Oasis model ourselves, as it is already pretrained on Minecraft VPT contractor data, and the official training code has not been released. We use 32 frames as the conditioning input, wthich is the maximum supported context length, and perform inference in an autoregressive manner.

Their observation space is defined as (640, 360, 3), which is fully consistent with our dataset and thus requires no additional modifications. Their action space follows VPT's CameraQuantizer, with a maximum value of 20 and a bin size of 0.5, resulting in 40 discrete buckets that are normalized to the range [-1, 1]. In contrast, our actions are defined in radians within the range [-0.1, 0.1], and we directly use the raw values as actions. Additionally, their definitions of cameraX and cameraY are reversed compared to ours(and VPT's). Other actions, such as forward and jump, are consistent between the two settings. Inference hyperparameters are shown in Table 5

D.2.2. Oasis Results

However, the results fall short of expectations. As shown in Figure 10, we observe that errors compound over time: small imperfections quickly snowball into collapsed frames. After approximately 30 frames, the generated images become increasingly blurred and fail to recover. We include a sample from plains village and a sample from taiga village. To demonstrate that the blurriness is not caused by our dataset or action configuration, we additionally include the official inference demo (Player729-f153ac423f61-20210806-224813, 256 frames) in line 3 for comparison.

D.3. Mineworld

D.3.1. Mineworld Settings

Mineworld is also a model pretrained on Minecraft VPT contractor data, and its official training code has not been re-

Table 5. Oasis Inference Hyperparameters

Model	Video Encoder	Sampling
Model: `oasis-500M`	Model: ViT-VAE-L/20	DDIM steps: 10
Structure: DiT-S/2	VAE patch size: 20	Max noise level: 1000
Model.max_frames: 32	# Prompt frames: 32	Noise absolute max: 20
		Stabilization level: 15
		Beta schedule: Sigmoid

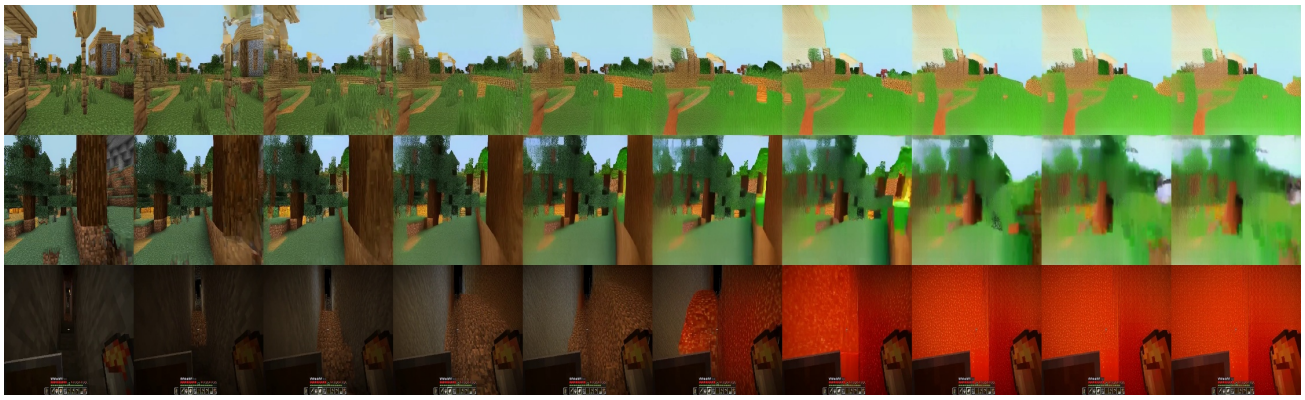


Figure 10. Results of Oasis. The first row corresponds to plains village, the second row to taiga village, and the third row shows the official inference demo. All three cases exhibit increasing blurriness.

leased. For the same reason, we directly evaluate Mineworld on our benchmark without further training. Specifically, we evaluate the largest publicly available model, Mineworld-1200M-16f and Mineworld-1200M-32f, which support 16 frames and 32 frames as context respectively.

Mineworld supports up to 32 frames of total context, which includes both historical and previously generated frames. Its generation process is chunk-based. We experiment using sliding window strategies: using 31 frames as context to autoregressively generate the remaining frames.

Mineworld defines its observation space as 384×224. To align with this, we resize our original 640×360 frames to 384×224 using an area-based interpolation method, and apply standard normalization to the pixel values. In Mineworld, visual observations are further compressed by a VAE into a latent representation of size 24×14. As for the action space, since Mineworld uses degrees to represent camera rotations, we convert our radian-based actions into degrees before passing them to the model.

D.3.2. Mineworld Results

To qualitatively evaluate the model's predictions, we visualize rollout results alongside ground truth frames in Figure 11. The first and third rows show rollouts for plains village (village ID:20, trajectory ID: 05-10_14-09-18) and desert village(village ID:20, trajectory ID: 05-10_14-11-32), respectively, while the second and fourth rows present their corresponding ground truth trajectories. We observe that, unlike Oasis, Mineworld does not exhibit visual collapse. However, it similarly lacks spatial consistency and fails to reconstruct the corresponding scene structure.

D.4. DIAMOND

D.4.1. DIAMOND Settings

DIAMOND (Alonso et al., 2024) is a diffusion-based world model built upon the UNet architecture (Ronneberger et al., 2015). It generates video frames autoregressively, conditioning on both past observations and actions, allowing it to model complex temporal dynamics in sequential decision-making environments. In its original design, DIAMOND is used as part of a model-based reinforcement learning pipeline, where the learned world model is used to train an RL agent and evaluated on the Atari 100k benchmark. And in CS:GO branch, DIAMOND also exhibits extraordinary ability to modeling dynamics. We follow the `CS:GO` configuration and focus solely on training the world model, omitting the reinforcement learning (RL) agent training phase. For the observation space, following the default settings of the CS:GO branch in DIAMOND, we first resize the input images from 640×360 to 320×180, and then further downscale them to 64×36. Diffusion is performed on the 64×36 images, and an additional upsampler is trained to reconstruct the images back to 320×180.

For the action space, we adopt VPT-style camera quantization with a maximum value of 0.1 and a bin size of 0.02, resulting in 11 discrete bins for both yaw and pitch. Combined

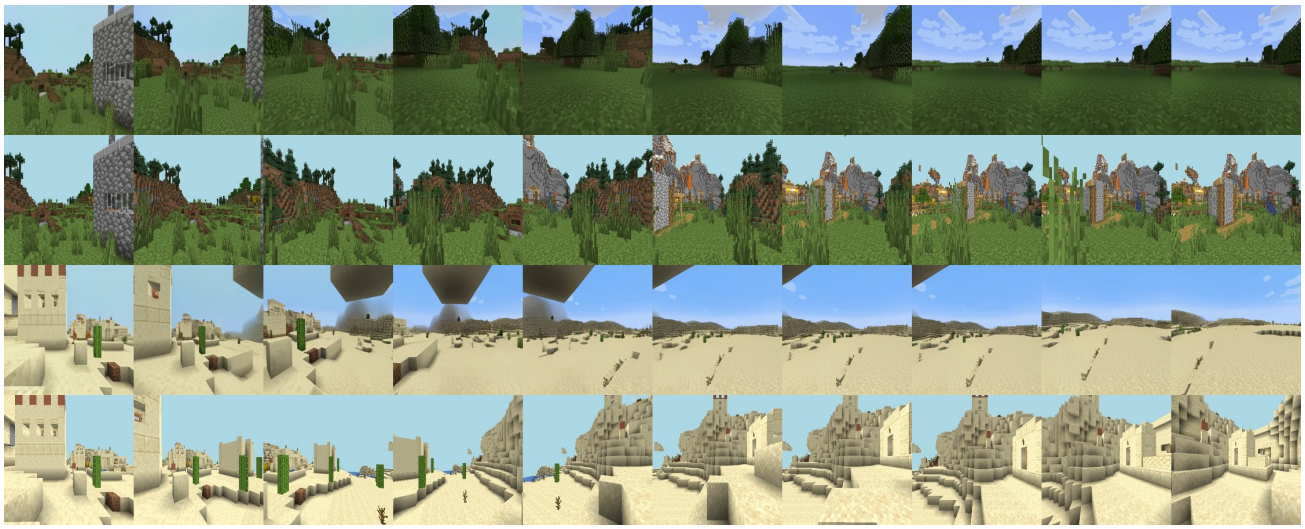


Figure 11. Results of Mineworld.The first and second rows show the rollout and ground truth for plains village, respectively; the third and fourth rows show the same for desert village.

with forward and jump actions, the total action dimension is 24. The training hyperparameters are summarized in Table 6.

D.4.2. DIAMOND Results

In Figure 12, we present the rollout results of DIAMOND. We observe that DIAMOND does not suffer from visual collapse and is able to reconstruct the initial frames relatively accurately, which aligns with its context window of length 32. However, as the rollout progresses, the generated scenes gradually converge to empty grasslands, indicating that the model forgets previously observed structures. This suggests that DIAMOND, like others, fails to maintain spatial consistency over long horizons.

D.5. Navigation World Model

D.5.1. NWM Settings

Navigation World Model (NWM) proposes a world model based on the Conditioned Diffusion Transformer (CDiT). Regarding the observation space, we first align with NWM's input format by decomposing trajectory videos into individual images. Following the official setting, each image is resized to 224×224, normalized, and then compressed into a 32×32 latent representation using the Stable Diffusion VAE.

For the action space, NWM differs from prior models in that it conditions on the agent's (x, z) position and yaw angle, using absolute location and orientation to reconstruct future observations. Since our dataset records the (x, z) coordinates and yaw at each step, we can directly adopt this format. Note that pitch information is omitted here. Empirically, we find that for navigation tasks, pitch tends to be less critical than yaw.

We train a CDiT-L/2 model with a context window of 32 frames. Detailed training hyperparameters are provided in Table 7.

D.5.2. NWM Results

In Figure 13, we present the rollout results of Navigation World Model. We observe that NWM retains a considerable amount of memory in the initial frames. However, the generated frames are noticeably blurry and distorted, suggesting that the model is unable to faithfully reconstruct spatial details or maintain long-range spatial consistency.

E. Extra Related Works

E.1. World Models

The goal of a world model is to simulate the environment: Given the current state and action, it predicts the next state and the corresponding reward. World models were originally proposed to improve sample efficiency in reinforcement learning (Oh et al., 2015; Ha & Schmidhuber, 2018). In the context of model-based reinforcement learning, numerous studies have explored various architectural designs of world models. Dreamer (Hafner et al., 2020; 2022; 2024) uses a Recurrent State Space Model (RSSM) and achieves human-level performance on Atari games. TWM (Robine et al., 2023) adapts DreamerV2's RSSM to use a transformer architecture. IRIS (Micheli et al., 2023) builds image tokens with discrete autoencoder and adopts an autoregressive transformer. DIAMOND (Alonso et al., 2024) leverages diffusion models to generate future frames. These architectures, to varying degrees, retain information from the past to aid in future image prediction; however, they still lack an effective memory design capable of maintaining long-horizon

Table 6. Model Configuration Parameters

Category	Parameter	Denoiser	Upsampler
General	sigma_data	0.5	0.5
	sigma_offset_noise	0.1	0.1
	noise_previous_obs	true	false
	upsampling_factor	null	5
Inner Model	img_channels	3	3
	num_steps_conditioning	4	1
	cond_channels	2048	2048
Training	num_autoregressive_steps	4	1
	start_after_epochs	0	0
	steps_first_epoch	400	400
	steps_per_epoch	400	400
	sample_weights	null	null
	batch_size	64	16
	grad_acc_steps	2	2
	lr_warmup_steps	100	100
	max_grad_norm	10.0	10.0
Optimizer	lr	1e-4	1e-4
	weight_decay	1e-2	1e-2
	eps	1e-8	1e-8
Diffusion Sampler	num_steps_denoising	3	10
	sigma_min	2e-3	1
	sigma_max	20.0	5.0
	rho	7	7
	order	1	1
	s_churn	0.0	10.0
	s_tmin	0.0	1
	s_tmax	∞	5
	s_noise	1.0	0.9
	s_cond	0.005	0

spatial consistency.

Beyond model-based reinforcement learning, the potential of world models has also been increasingly explored in other domains. Playable Video Generation (Menapace et al., 2021), gameNGen (Valevski et al., 2024), Oasis (Decart et al., 2024) investigate the potential of using world models as game engines. GAIA (Hu et al., 2023), DriveDreamer (Wang et al., 2023a), vista (Gao et al., 2024) explores their application in autonomous driving. In robotics, two of the most significant challenges are the scarcity of training data and the high cost of model training. The emergence of world models offers a promising direction for addressing these two major challenges. UniSim (Yang et al., 2023) learns a universal simulator for robot manipulation. DayDreamer (Wu et al., 2022) employs world model for real-world robot learning. NWM (Bar et al., 2024) investigates visual navigation problem in world model. In almost all downstream tasks—especially those involving decision-making such as autonomous driving and navigation—maintaining spatial consistency is a critical requirement.

From a more unified perspective, world models can be viewed as action-conditioned video generation models. In computer vision, generating videos has been a long standing challenge (Yang et al., 2024). Recent approaches leverage transformers, and diffusion models (Chen et al., 2024; Song et al., 2025) to generate longer, more coherent video sequences. From the perspective of video modeling, maintaining spatial consistency across frames—especially over long temporal horizons—remains a critical and unsolved challenge.

E.2. Minecraft as an AI Testbed

Minecraft is an open-world environment characterized by diverse terrains and rich interaction dynamics. As one of the most popular games globally, it benefits from extensive community-driven resources and an active user base. Recent research has increasingly adopted Minecraft as a platform for training generative agents (Baker et al., 2022; Lifshitz et al., 2024; Cai et al., 2023; 2024; 2025b;a; Zhao et al., 2024) and constructing digital world models (Guo

Figure 12. Results of Mineworld. First row: Rollout results. Second row: Ground truth.

Figure 13. Results of NWM. First row: Rollout results. Second row: Ground truth.

Table 7. NWM Training Configuration

Parameter	**Value**
Batch size	8
Number of workers	12
Model	CDiT-L/2
Learning rate	8×10^{-5}
Normalize action space	True
Gradient clipping value	10.0
Context size	32
Distance Prediction	
Min distance category	-64
Max distance category	64
Action Output	
Predicted trajectory length	64
Dataset Settings	
Image size	224
Goals per observation	4

et al., 2025; Decart et al., 2024; Hong et al., 2024; Song et al., 2025). Several Minecraft-based datasets and benchmarks have been introduced, including MineDoJo (Fan et al., 2022), which provides multimodal knowledge and human gameplay videos sourced from YouTube; VPT (Baker et al., 2022), which releases large-scale trajectories of human players engaged in building and mining; and MineRL (Guss et al., 2019), which focuses on trajectories related to cave exploration. However, these efforts primarily emphasize prospective exploration and object interaction, with limited attention to the role of historical context. In contrast, our proposed dataset and benchmark are designed to promote and assess models' abilities to leverage past observations and perform spatial reasoning. Our data collection pipeline is built on top of Mineflayer (Contributors, 2013–2024), a widely used rule-based Minecraft automation framework.