Navigation World Models

Supplementary Material

The structure of the Appendix is as follows: we start by describing how we plan navigation trajectories via Standalone Planning in Section 7, and then include more experiments and results in Section 8.

7. Standalone Planning Optimization

As described in Section 3.3, we use a pretrained NWM to standalone-plan goal-conditioned navigation trajectories by optimizing Eq.5. Here, we provide additional details about the optimization using the Cross-Entropy Method [48] and the hyperparameters used. Full standalone navigation planning results are presented in Section 8.2.

We optimize trajectories using the Cross-Entropy Method, a gradient-free stochastic optimization technique for continuous optimization problems. This method iteratively updates a probability distribution to improve the likelihood of generating better solutions. In the unconstrained standalone planning scenario, we assume the trajectory is a straight line and optimize only its endpoint, represented by three variables: a single translation u and yaw rotation ϕ. We then map this tuple into eight evenly spaced delta steps, applying the yaw rotation at the final step. The time interval between steps is fixed at $k = 0.25$ seconds. The main steps of our optimization process are as follows:

- **Initialization**: Define a Gaussian distribution with mean $\mu = (\mu_{\Delta x}, \mu_{\Delta y}, \mu_\phi)$ and variance $\Sigma = \text{diag}(\sigma^2_{\Delta x}, \sigma^2_{\Delta y}, \sigma^2_\phi)$ over the solution space.
- **Sampling**: Generate $N = 120$ candidate solutions by sampling from the current Gaussian distribution.
- **Evaluation**: Evaluate each candidate solution by simulating it using the NWM and measuring the LPIPS score between the simulation output and input goal images. Since NWM is stochastic, we evaluate each candidate solution M times and average to obtain a final score.
- **Selection**: Select a subset of the best-performing solutions based on the LPIPS scores.
- **Update**: Adjust the parameters of the distribution to increase the probability of generating solutions similar to the top-performing ones. This step minimizes the cross-entropy between the old and updated distributions.
- **Iteration**: Repeat the sampling, evaluation, selection, and update steps until a stopping criterion (e.g. convergence or iteration limit) is met.

For simplicity, we run the optimization process for a single iteration, which we found effective for short-horizon planning of two seconds, though further improvements are possible with more iterations. When navigation constraints are applied, parts of the trajectory are zeroed out to respect these constraints. For instance, in the "forward-first" scenario, the translation action is $u = (\Delta x, 0)$ for the first five steps and $u = (0, \Delta y)$ for the last three steps.

8. Experiments and Results

8.1. Experimental Study

We elaborate on the metrics and datasets used.

Evaluation Metrics. We describe the evaluation metrics used to assess predicted navigation trajectories and the quality of images generated by our NWM.

For visual navigation performance, **Absolute Trajectory Error** (ATE) measures the overall accuracy of trajectory estimation by computing the Euclidean distance between corresponding points in the estimated and ground-truth trajectories. **Relative Pose Error** (RPE) evaluates the consistency of consecutive poses by calculating the error in relative transformations between them [57].

To more rigorously assess the semantics in the world model outputs, we use Learned Perceptual Image Patch Similarity (LPIPS) and DreamSim [14], which evaluate perceptual similarity by comparing deep features from a neural network [75]. LPIPS, in particular, uses AlexNet [33] to focus on human perception of structural differences. Additionally, we use Peak Signal-to-Noise Ratio (PSNR) to quantify the pixel-level quality of generated images by measuring the ratio of maximum pixel value to error, with higher values indicating better quality.

To study image and video synthesis quality, we use Fréchet Inception Distance (FID) and Fréchet Video Distance (FVD), which compare the feature distributions of real and generated images or videos. Lower FID and FVD scores indicate higher visual quality [23, 64].

Datasets. For all robotics datasets, we have access to the location and rotation of the robots, and we use this to infer the actions as the delta in location and rotation. We remove all backward movement which can be jittery following NoMaD [55], thereby splitting the data to forward walking segments for SCAND [30], TartanDrive [60], RECON [52], and HuRoN [27]. We also utilize unlabeled Ego4D videos, where we only use time shift as action. Next, we describe each individual dataset.

- SCAND [30] is a robotics dataset consisting of socially compliant navigation demonstrations using a wheeled Clearpath Jackal and a legged Boston Dynamics Spot. SCAND has demonstrations in both indoor and outdoor settings at UT Austin. The dataset consists of 8.7 hours, 138 trajectories, 25 miles of data and we use the corresponding camera poses. We use 484 video segments for

data	unknown environment	known environments			
	Go Stanford	RECON	HuRoN	SCAND	TartanDrive
in-domain data	0.658 ± 0.002	**0.295** ± 0.002	**0.250** ± 0.003	0.403 ± 0.002	**0.414** ± 0.001
+ Ego4D (unlabeled)	**0.652** ± 0.003	0.368 ± 0.003	0.377 ± 0.002	**0.398** ± 0.001	0.430 ± 0.000

Table 5. **Training on additional unlabeled data improves performance on unseen environments.** Reporting results on unknown environment (Go Stanford) and known one (RECON). Results reported by evaluating LPIPS 4 seconds into the future.

training and 121 video segments for testing. Used for training and evaluation.

- TartanDrive [60] is an outdoor off-roading driving dataset collected using a modified Yamaha Viking ATV in Pittsburgh. The dataset consists of 5 hours and 630 trajectories. We use 1,000 video segments for training and 251 video segments for testing.
- RECON [52] is an outdoor robotics dataset collected using a Clearpath Jackal UGV platform. The dataset consists of 40 hours across 9 open-world environments. We use 9,468 video segments for training and 2,367 video segments for testing. Used for training and evaluation.
- HuRoN [27] is a robotics dataset consisting of social interactions using a Robot Roomba in indoor settings collected at UC Berkeley. The dataset consists of over 75 hours in 5 different environments with 4,000 human interactions. We use 2,451 video segments for training and 613 video segments for testing. Used for training and evaluation.
- GO Stanford [24, 25], a robotics datasets capturing the fisheye video footage of two different teleoperated robots, collected at at least 27 different Stanford building with around 25 hours of video footage. Due to the low resolution images, we only use it for out of domain evaluation.
- Ego4D [18] is a large-scale egocentric dataset consisting of 3,670 hours across 74 locations. Ego4D consists a variety of scenarios such as Arts & Crafts, Cooking, Construction, Cleaning & Laundry, and Grocery Shopping. We use only use videos which involve visual navigation such as Grocery Shopping and Jogging. We use a total 1619 videos of over 908 hours for training only. Only used for unlabeled training unlabeled training. The videos we use are from the following Ego4D scenarios: "Skateboard/scooter", "Roller skating", "Football", "Attending a festival or fair", "Gardener", "Mini golf", "Riding motorcycle", "Golfing", "Cycling/jogging", "Walking on street", "Walking the dog/pet", "Indoor Navigation (walking)", "Working in outdoor store", "Clothes/other shopping", "Playing with pets", "Grocery shopping indoors", "Working out outside", "Farmer", "Bike", "Flower Picking", "Attending sporting events (watching and participating)", "Drone flying", "Attending a lecture/class", "Hiking", "Basketball", "Gardening", "Snow sledding", "Going to the park".

Visual Navigation Evaluation Set. Our main finding when constructing visual navigation evaluation sets is that forward motion is highly prevalent, and if not carefully accounted for, it can dominate the evaluation data. To create diverse evaluation sets, we rank potential evaluation trajectories based on how well they can be predicted by simply moving forward. For each dataset, we select the 100 examples that are least predictable by this heuristic and use them for evaluation.

Time Prediction Evaluation Set. Predicting the future frame after k seconds is more challenging than estimating a trajectory, as it requires both predicting the agent's trajectory and its orientation in pixel space. Therefore, we do not impose additional diversity constraints. For each dataset, we randomly select 500 test prediction examples.

8.2. Experiments and Results

Training on Additional Unlabeled Data. We include results for additional known environments in Table 5 and Figure 11. We find that in known environments, models trained exclusively with in-domain data tend to perform better, likely because they are better tailored to the in-domain distribution. The only exception is the SCAND dataset, where dynamic objects (e.g. humans walking) are present. In this case, adding unlabeled data may help improve performance by providing additional diverse examples.

Known Environments. We include additional visualization results of following trajectories using NWM in the known environments RECON (Figure 12), SCAND (Figure 13), HuRoN (Figure 14), and Tartan Drive (Figure 15). Additionally, we include full FVD comparison of DIAMOND and NWM in Table 6.

dataset	DIAMOND	NWM (ours)
RECON	762.734 ± 3.361	**200.969** ± 5.629
HuRoN	881.981 ± 11.601	**276.932** ± 4.346
TartanDrive	2289.687 ± 6.991	**494.247** ± 14.433
SCAND	1945.085 ± 8.449	**401.699** ± 11.216

Table 6. **Comparison of Video Synthesis Quality.** 16 second videos generated at 4 FPS, reporting FVD (lower is better).

Planning (Ranking). Full goal-conditioned navigation results for all in-domain datasets are presented in Table 7. Compared to NoMaD, we observe consistent improvements when using NWM to select from a pool of 16 trajectories, with further gains when selecting from a larger pool of 32.

model	RECON		HuRoN		Tartan		SCAND	
	ATE	RTE	ATE	RTE	ATE	RTE	ATE	RTE
Forward	1.92 ± 0.00	0.54 ± 0.00	4.14 ± 0.00	1.05 ± 0.00	5.75 ± 0.00	1.19 ± 0.00	2.97 ± 0.00	0.62 ± 0.00
GNM	1.87 ± 0.00	0.73 ± 0.00	3.71 ± 0.00	1.00 ± 0.00	6.65 ± 0.00	1.62 ± 0.00	2.12 ± 0.00	0.61 ± 0.00
NoMaD	1.95 ± 0.05	0.53 ± 0.01	3.73 ± 0.04	0.96 ± 0.01	6.32 ± 0.03	1.31 ± 0.01	2.24 ± 0.03	0.49 ± 0.01
NWM + NoMaD ($\times 16$)	1.88 ± 0.03	0.51 ± 0.01	3.73 ± 0.05	0.95 ± 0.01	6.26 ± 0.06	1.30 ± 0.01	2.18 ± 0.05	0.48 ± 0.01
NWM + NoMaD ($\times 32$)	1.79 ± 0.02	0.49 ± 0.00	$\mathbf{3.68} \pm 0.03$	$\mathbf{0.95} \pm 0.01$	6.25 ± 0.05	1.29 ± 0.01	2.19 ± 0.03	0.47 ± 0.01
NWM (only)	$\mathbf{1.13} \pm 0.02$	$\mathbf{0.35} \pm 0.01$	4.12 ± 0.03	0.96 ± 0.01	$\mathbf{5.63} \pm 0.06$	$\mathbf{1.18} \pm 0.01$	$\mathbf{1.28} \pm 0.02$	$\mathbf{0.33} \pm 0.01$

Table 7. **Goal Conditioned Visual Navigation**. ATE and RPE results on on all in domain datasets, predicting trajectories of up to 2 seconds. NWM achieves improved results on all metrics compared to previous approaches NoMaD [55] and GNM [53].

For Tartan Drive, we note that the dataset is heavily dominated by forward motion, as reflected in the results compared to the "Forward" baseline, a prediction model that always selects forward-only motion.

Standalone Planning. For standalone planning, we run the optimization procedure outlined in Section 7 for 1 step, and evaluate each trajectories for 3 times. For all datasets, we initialize $\mu_{\Delta y}$ and μ_ϕ to be 0, and $\sigma^2_{\Delta y}$ and σ^2_ϕ to be 0.1. We use different $(\mu_{\Delta x}, \sigma^2_{\Delta x})$ across each dataset: $(-0.1, 0.02)$ for RECON, $(0.5, 0.07)$ for TartanDrive, $(-0.25, 0.04)$ for SCAND, and $(-0.33, 0.03)$ for HuRoN. We include the full standalone navigation planning results in Table 7. We find that using planning in the stand-alone setting performs better compared to other approaches, and specifically previous hard-coded policies.

Real-World Applicability. A key bottleneck in deploying NWM in real-world robotics is inference speed. We evaluate methods to improve NWM efficiency and measure their impact on runtime. We focus on using NWM with a generative policy (Section 3.3) to rank 32 four-second trajectories. Since trajectory evaluation is parallelizable, we analyze the runtime of simulating a single trajectory. We find that existing solutions can already enable real-time applications of NWM at 2-10HZ (Table 8).

NWM	+Time Skip	+Distillation.	+Quant. 4-bit
30.3 ± 0.2	14.7 ± 0.1	0.4 ± 0.1	0.1 (est. [12])

Table 8. Runtime (seconds) on an NVIDIA RTX 6000 Ada card.

Inference time can be accelerated by composing every adjacent pair of actions (via Eq. 2) then simulating only 8 future states instead of 16 ("Time Skip"), which does not degrade navigation performance. Reducing the diffusion denoising steps from 250 to 6 by model distillation [70] further speeds up inference with minor visual quality loss.[3] Taken together, these two ideas can enable NWM to run in real time. Quantization to 4-bit, which we haven't explored, can lead to a $\times 4$ speedup without performance hit [12].

[3]Using the distillation implementation for DiTs from https://github.com/hao-ai-lab/FastVideo

CDiT-L	context 2	action only	goals 2	ours	ours + TTA
0.656	0.655	0.661	0.654	0.652	**0.650**

Table 9. Results in *unknown environment* ("Go Stanford"). Reporting lpips on 4 seconds future prediction. Lower is better.

Test-time adaptation. Test-time adaptation has shown to improve visual navigation [13, 16]. What is the relation between planning using a world model and test-time adaptation? We hypothesize that the two ideas are orthogonal, and include test-time adaptation results. We consider a simplified adaptation approach by fine-tuning NWM for 2k steps on trajectories from an *unknown environment*. We show that this adaptation improves trajectory simulation in this environment (see "ours+TTA" in Table 9), where we also include additional baselines and ablations.

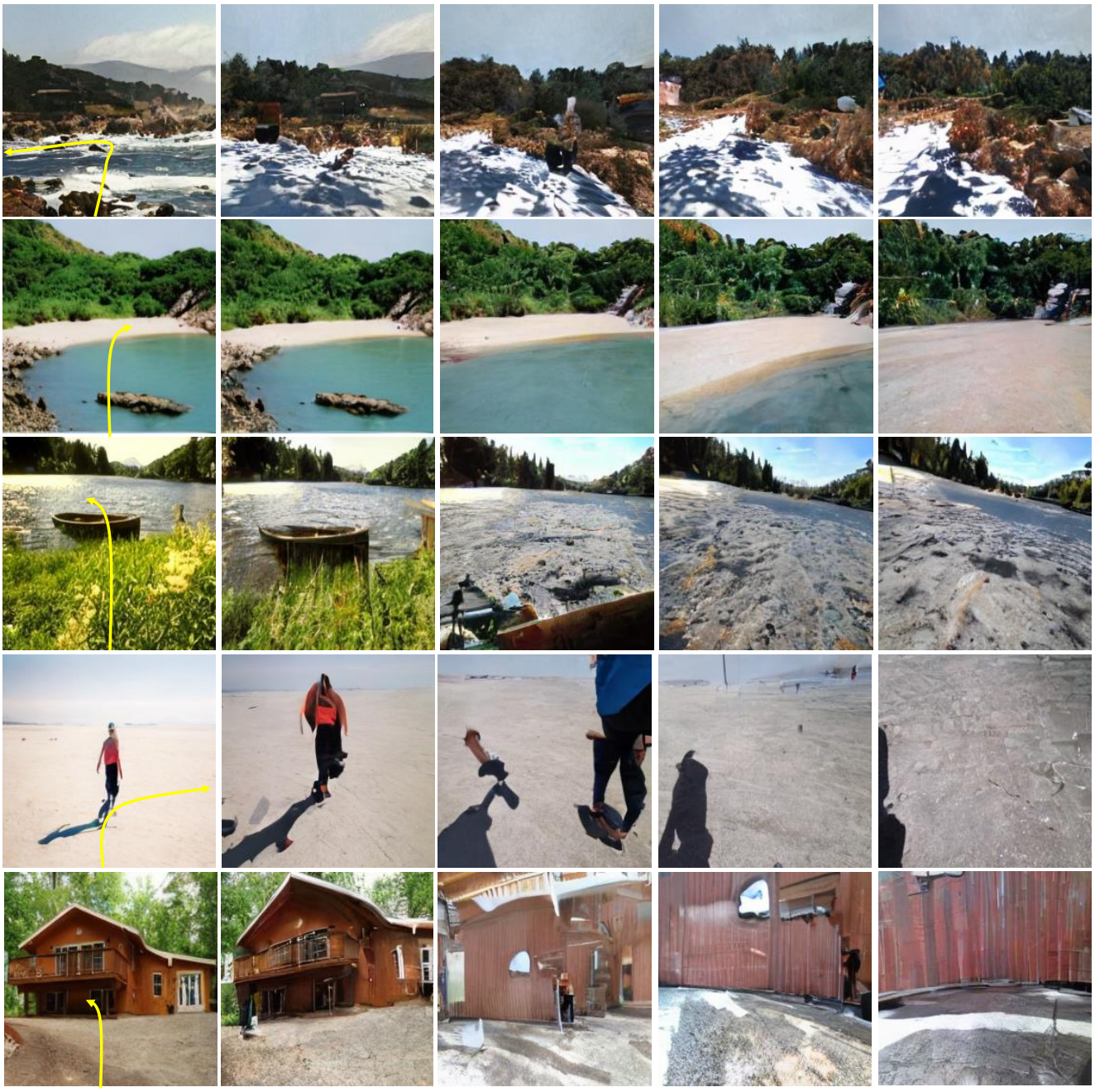

Figure 11. **Navigating Unknown Environments**. NWM is conditioned on a single image, and autoregressively predicts the next states given the associated actions (marked in yellow) up to 4 seconds and 4 FPS. We plot the generated results after 1, 2, 3, and 4 seconds.

Figure 12. **Video generation examples on RECON**. NWM is conditioned on a single first image, and a ground truth trajectory and autoregressively predicts the next up to 16 seconds at 4 FPS. We plot the generated results from 2 to 16 seconds, every 1 second.

Figure 13. **Video generation examples on SCAND**. NWM is conditioned on a single first image, and a ground truth trajectory and autoregressively predicts the next up to 16 seconds at 4 FPS. We plot the generated results from 2 to 16 seconds, every 1 second.

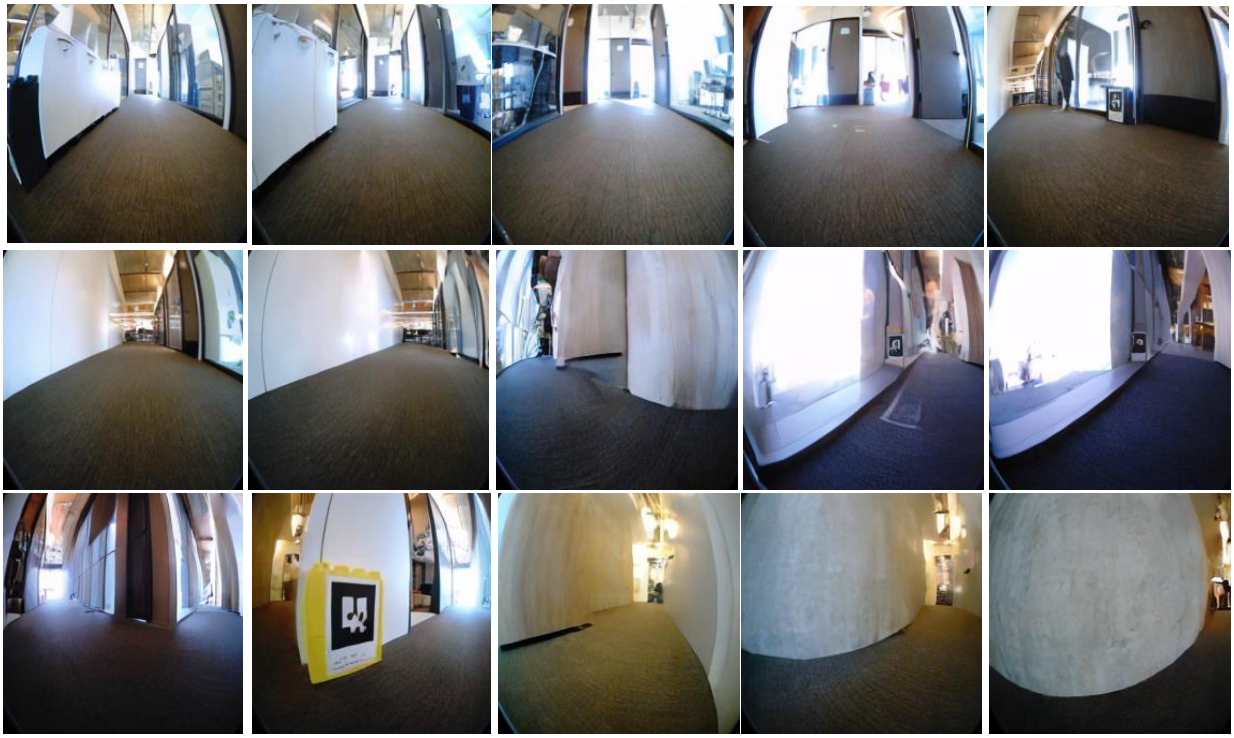

Figure 14. **Video generation examples on HuRoN**. NWM is conditioned on a single first image, and a ground truth trajectory and autoregressively predicts the next up to 16 seconds at 4 FPS. We plot the generated results from 2 to 16 seconds, every 1 second.

Figure 15. **Video generation examples on Tartan Drive**. NWM is conditioned on a single first image, and a ground truth trajectory and autoregressively predicts the next up to 16 seconds at 4 FPS. We plot the generated results from 2 to 16 seconds, every 1 second.