World Model Robustness via Surprise Recognition

Supplementary Material

A. Sensors

CARLA [6]

- *Lidar*: Captures a 3D point cloud of the surrounding environment.
- *Camera*: Provides a forward-facing RGB view from the vehicle.
- *Bird's-Eye View (BEV) Raw*: Displays the roadmap and surrounding vehicles from a top-down perspective.
- *Collision*: Detects collisions with objects in the environment.
- *Bird's-Eye View with Traffic Signals*: Renders traffic lights and signs directly on the BEV.
- *Bird's-Eye View with Waypoints*: Renders only waypoints on the BEV (GPS setting).
- *Bird's-Eye View Full*: Renders all available semantic information on the BEV.

Safety Gymnasium [19]

- *Bird Eye View*: A view of the agent and the entire map from above.
- *Front Vehicle View*: The immediate front view of the agent.
- *Dash Camera View*: The immediate front view of the agent from the dash cam perspective.

B. CARLA Noise Corruption and Sensor Degradation

B.1. Environment and Representations

In CARLA [6], an open-source urban driving simulator, the agent is equipped with a diverse sensor suite comprising seven modalities (see A). We evaluate performance on three representative scenarios—Stop Sign, Right Turn, and Four Lane Driving—which test the agent's ability to perceive, plan, and act under different traffic configurations.

B.2. Robustness under Visual Corruptions

We evaluate the robustness of the driving agent in the CARLA simulator by subjecting input images to five types of visual corruptions:

- Chromatic Aberration: introduces small spatial shifts between RGB channels, producing color fringing and edge distortions similar to lens dispersion.
- Gaussian Noise: adds pixel-level random variations, modeling low-light sensor interference.
- Glare: causes extreme overexposure that whitens the whole frame, erasing most visual information.
- Jitter: applies random shifts in contrast or brightness, simulating unstable or high-noise sensor conditions.
- Occlusion: masks random regions of the frame, simulating partial blockage of the camera view.

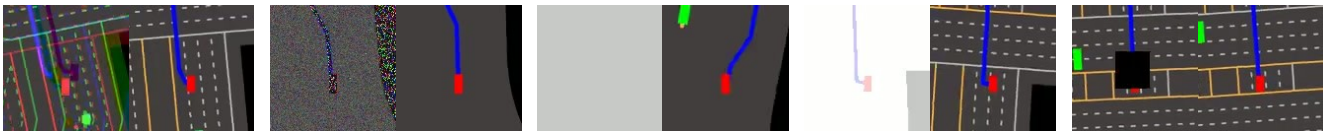
The goal is to test how well the agent can still perform its task under degraded visual inputs. Quantitatively, we plot 3D surface graphs of the task performance metric as a function of corruption intensity and proportion. Each surface corresponds to one CARLA task—Stop Sign, Right Turn, and Four-Lane Driving—and compares four methods: the baseline world-model agent, Median Filtering, our proposed Rejection Sampling, and HRSSM. For all tasks shown in Figure 8, our Rejection Sampling maintains a much smoother and higher performance surface, indicating stable behavior even under severe sensor degradation. This trend demonstrates that the proposed Filter not only delays the onset of failure but also preserves task continuity across extreme perturbations, achieving consistently superior robustness and reliability in all CARLA environments.

B.3. Efficiency of Surprise-Guided Filter

Figure 9 illustrates the efficiency of our surprise-guided filtering approach by comparing the $O(n \log n)$ sensor-selection strategy against the exhaustive 2^N subset search across the three CARLA tasks—Right Turn, Four-Lane, and Stop Sign—under four corruption types: Gaussian, Glare, Jitter, and Lag. The specific perturbations applied to the camera view are defined as follows:

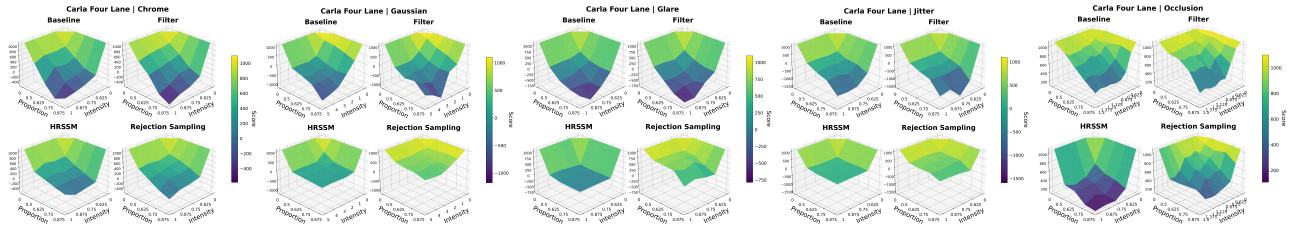
- Gaussian Noise: adds pixel-level random variations, modeling low-light sensor interference.
- Glare: causes extreme overexposure that whitens the whole frame, erasing most visual information.
- Jitter: applies random shifts in contrast or brightness, simulating unstable or high-noise sensor conditions.
- Latency (Lag): causes the observation to lag behind the true state by reusing stale frames for several steps, emulating delayed or frozen sensor updates.

The evaluation spans a wide range of augmentation intensities to test how each method scales as visual degradation increases. For Gaussian Noise and Jitter, both methods display non-monotonic trends: mild noise can be mitigated by masking high-surprise views, while stronger perturbations introduce sharp drops, particularly in the Four-Lane task at intensity level 4. Glare yields more stable performance because overexposed channels are consistently identified and suppressed. Lag produces a near-monotonic decline across all tasks due to its strong disruption of temporal coherence. Crucially, the green $O(n \log n)$ curves typically track the red 2^N curves closely, despite the exponential cost

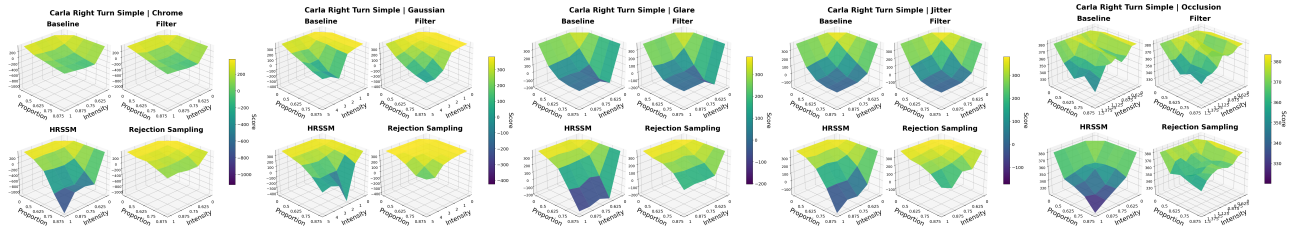


(a) Chrome (b) Gaussian (c) Glare (d) Jitter (e) Occlusion

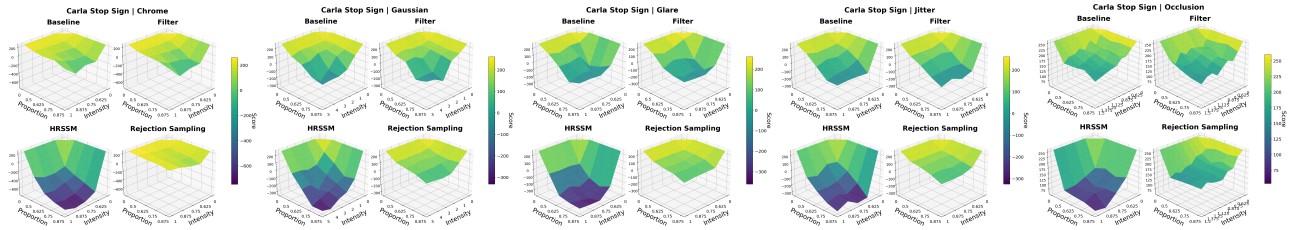
Figure 7. Visual comparison of the clean and corrupted from the CARLA BEV perspective. The left frame is the corrupted observation, and the right frame is the ground truth observation.



(a) Four-Lane Driving



(b) Right Turn



(c) Stop Sign

Figure 8. Qualitative results across three CARLA tasks under five corruption types: Chrome, Gaussian, Glare, Jitter, and Occlusion.

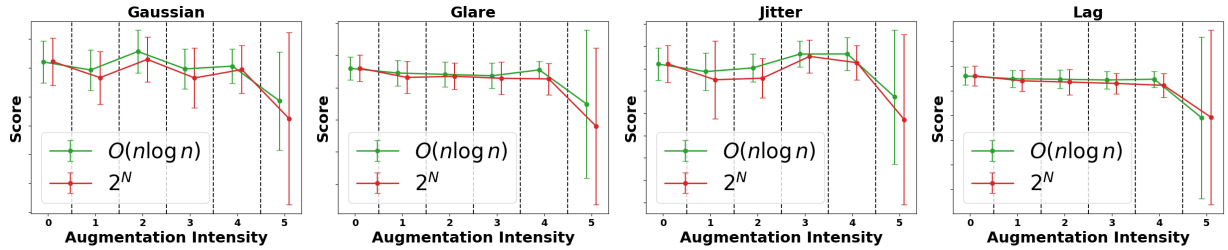
associated with the brute-force method. This demonstrates that the surprise-guided filter attains robustness comparable to exhaustive subset evaluation at only a fraction of the computational burden. The results confirm that the learned surprise signal provides a reliable ranking over sensor utility, making combinatorial search unnecessary and enabling practical, scalable deployment in multi-sensor RL settings.

B.4. Sensor Failure Analysis

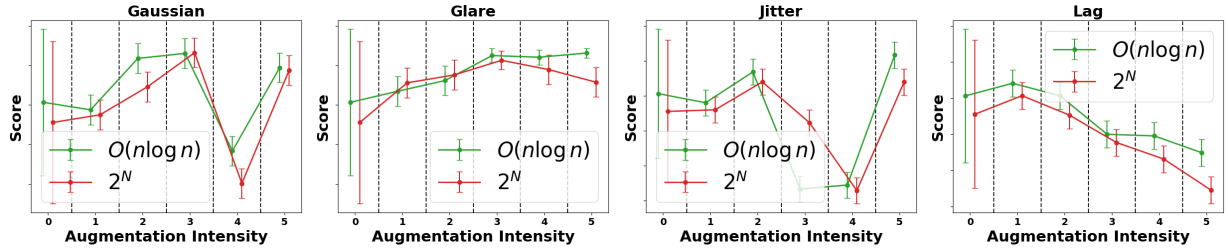
Figure 10 provides a comprehensive breakdown of how increasing sensor failures affect agent performance across all three CARLA tasks—Right Turn, Stop Sign, and Four-Lane Driving—under four primary corruption types detailed in Section B.3: Gaussian Noise, Glare, Jitter, and Latency.

Each subfigure in Figure 10 plots episodic score as a function of the number of failed sensors, comparing four methods: the Baseline world-model agent, the Augmented agent trained with Gaussian noise, the RME masked-encoder agent, and our Confident Representation approach. Across all tasks and corruption types, Baseline exhibits the steepest decline, often collapsing to near-zero or negative scores as failures accumulate. RME reduces variance but remains sensitive to structured failures such as Glare and Latency. Augmented improves stability under moderate Gaussian noise, but still degrades noticeably when failures intensify.

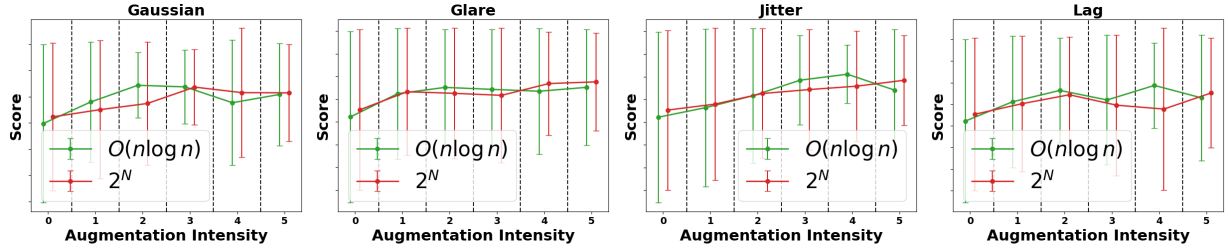
In contrast, Confident Representation consistently maintains the highest performance curve, showing only mild degradation even with multiple failed sensors. Its advantage is particularly notable under Glare and Jitter, where corrupted visual channels severely mislead other methods. Under Latency, all agents experience some decline, yet Confident Representation preserves stable, positive performance while avoiding the abrupt failures observed in Baseline and



(a) Right Turn



(b) Four Lane Driving



(c) Stop Sign

Figure 9. Comparison across three tasks under different noise conditions. We compare the 2^N brute force variant against our proposed $O(n \log n)$ Algorithm variant 2.

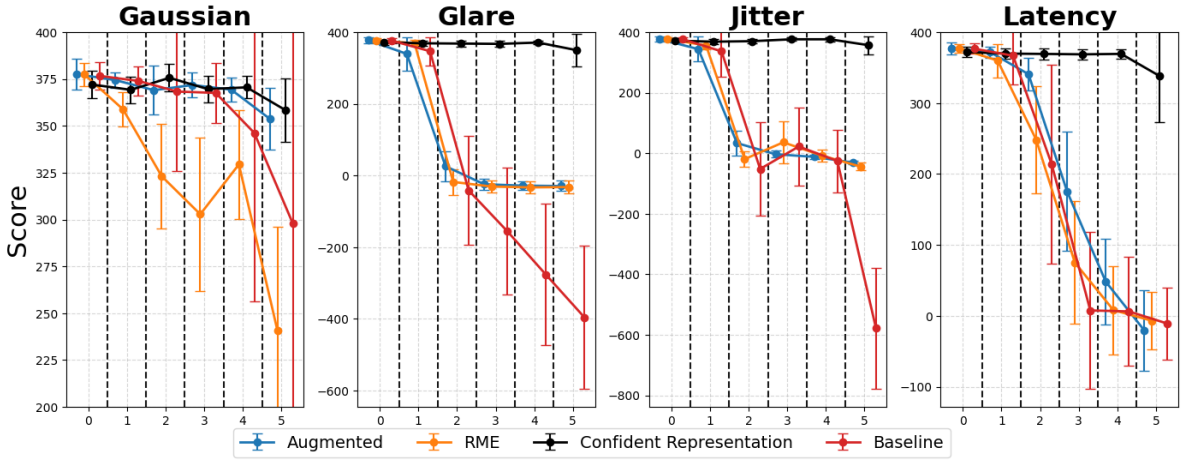
RME.

Overall, these results show that surprise-guided representation selection remains effective across diverse CARLA configurations, enabling the agent to identify corrupted views, suppress unreliable inputs, and sustain robust control even as sensor failures accumulate.

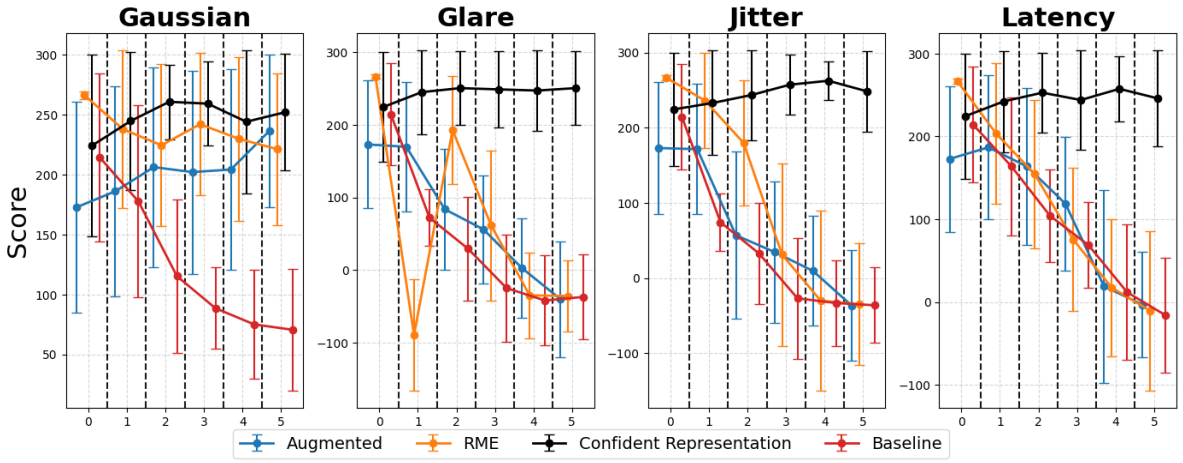
C. Safety Gymnasium Experiments

In the Safety Gymnasium domain [19], the agent's goal is to navigate safely to objectives in an environment with obstacles. We provide three overlapping camera views as described in Section A, which capture similar scene content from complementary perspectives. We evaluate performance on three tasks: PointGoal1, where a point-mass agent must reach a goal location while avoiding hazards; PointButton1, where a point-mass must press buttons scattered in the map while avoiding obstacles, and CarGoal1, where a car-like agent must reach a goal on a road with obstacles. Each task also penalizes unsafe collisions with a cost.

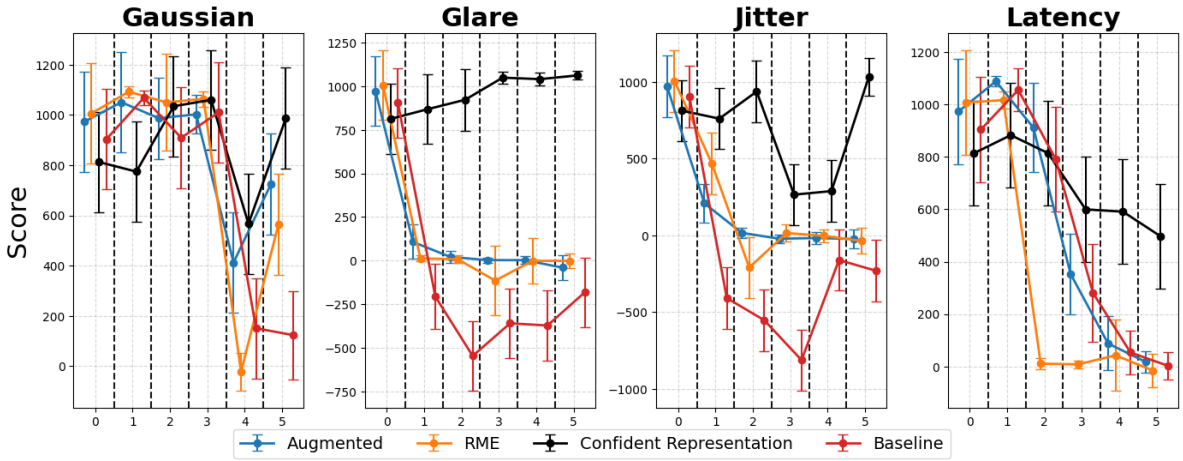
Figure 3 in the main text summarizes performance under four visual corruption types—Gaussian, Jitter, Glare, and Occlusion—applied independently to each of the three camera views. Since these views are partially redundant, corruption in a single view does not immediately prevent task completion but can mislead the policy when the corrupted view dominates the latent inference. The surprise-guided filter addresses this by downweighting or suppressing views whose inferred representations deviate strongly from the world model's predictive prior. As shown in the figure, this leads to consistent improvements in both performance and the Score–Cost ratio across all Safety Gymnasium tasks. In particular, the method effectively handles Glare and Occlusion, where one view becomes unreliable but the remaining views still convey meaningful structure. The resulting gains in safety and stability demonstrate that even in environments with overlapping visual inputs, surprise-guided representation selection provides measurable robustness against view-specific corruptions.



(a) Right Turn



(b) Stop Sign



(c) Four-Lane Driving

Figure 10. Agent performance as sensor failures increase.

D. Hardware Details and Hyperparameters

Name	Symbol	Value
General		
Replay capacity	—	5×10^6
Batch size	B	16
Batch length	T	64
Activation	—	$\mathrm{RMSNorm} + \mathrm{SiLU}$
Learning rate	—	4×10^{-5}
Gradient clipping	—	$\mathrm{AGC}(0.3)$
Optimizer	—	$\mathrm{LaProp}(\epsilon = 10^{-20})$
World Model		
Reconstruction loss scale	β_{pred}	1
Dynamics loss scale	β_{dyn}	1
Representation loss scale	β_{rep}	0.1
Latent unimix	—	1%
Free nats	—	1
Actor Critic		
Imagination horizon	H	15
Discount horizon	$1/(1-\gamma)$	333
Return lambda	λ	0.95
Critic loss scale	β_{val}	1
Critic replay loss scale	β_{repval}	0.3
Critic EMA regularizer	—	1
Critic EMA decay	—	0.98
Actor loss scale	β_{pol}	1
Actor entropy regularizer	η	3×10^{-4}
Actor unimix	—	1%
Actor RetNorm scale	S	$\mathrm{Per}(R, 95) - \mathrm{Per}(R, 5)$
Actor RetNorm limit	L	1
Actor RetNorm decay	—	0.99

Table 2. Hyperparameters of the Dreamerv3 model. We use the exact same parameters as discussed in [14].

Crafter All experiments can be reproduced on a system equipped with two NVIDIA GeForce GTX 1080 GPUs with 8 GB VRAM each to handle environment complexity, an AMD Ryzen 5 5600X 6-core processor, and at least 500 MB of storage for files (excluding training data, which depends on the environment and model hyperparameters).

CARLA and Safety Gymnasium All reported experiments related to Safety-Gymnasium and CARLA domains can be conducted on a workstation equipped with an NVIDIA GeForce RTX 4090 GPU (24 GB VRAM), an AMD Ryzen 9 7950X (16-core, 32-thread) CPU, 128 GB DDR5 RAM, and running Ubuntu 22.04 LTS (CUDA 12.4).

Cosmos All reported experiments relating to the Cosmos world model can be conducted on a workstation equipped with dual Intel Xeon 6737P CPUs (64 cores total), 2.0 TiB of DDR5 system memory, a NVIDIA RTX PRO 6000 utilizing driver version 580.82.07 (CUDA 13.0).

E. Further Results

E.1. Cosmos World Model

To further verify that the proposed rejection-based filtering mechanism generalizes beyond latent-state world models, we abstract the process of Figure 5 and extend our study to the Cosmos world model—a Diffusion-based video world model that performs pixel-space prediction through large-scale diffusion and spatio-temporal self-attention. Unlike the DreamerV3 architecture used in our main experiments, which relies on a Variational Autoencoder with explicit latent variables and KL-based surprise, Cosmos models dynamics directly in the image domain via autoregressive diffusion-based video prediction. This enables high-fidelity frame synthesis but also makes it highly sensitive to corruption in its first-frame conditioning.

We evaluate Cosmos on the robot pouring video, where a robotic manipulator pours liquid between jars on a tabletop. In our experiments, 75% of the input video is randomly perturbed with one of the noises discussed in B.2. Because Cosmos conditions its rollout on the last N frames of the input video, this corruption propagates through immediate subsequent predictions, as shown in the top row of Figure 11. After enabling our rejection mechanism, the system detects the corrupted immediate next frame as high-surprise and instead relies on internally predicted latent trajectories to reconstruct the scene. The bottom row shows markedly improved geometry and color fidelity around the robot arm and jars, indicating a safer frame to accept, demonstrating that selective rejection of corrupted conditioning frames can effectively suppress artifact propagation even in Transformer-based video world models.

E.1.1. Video Prompt: Robotic Arm Pouring Sequence

Scene Description

A robotic arm, primarily white with black joints and cables, is shown in a clean, modern indoor setting with a white tabletop. The arm, equipped with a gripper holding a small, light green pitcher, is positioned above a clear glass containing a reddish-brown liquid and a spoon. The robotic arm is in the process of pouring a transparent liquid into the glass. To the left of the pitcher, there is an open jar with a similar reddish-brown substance visible through its transparent body. In the background, a vase with white flowers and a brown couch are partially visible, adding to the contemporary ambiance. The lighting is bright, casting soft shadows on the table. The robotic arm's movements are smooth and controlled, demonstrating precision in its task. As the video progresses, the robotic arm completes the pour, leaving the glass half-filled with the reddish-brown liquid. The jar remains untouched throughout the sequence, and the spoon inside the glass remains stationary. The other robotic arm on the right side also stays stationary throughout the video. The final frame captures the robotic arm with the pitcher finishing the pour, with the glass now filled to a higher level, while the pitcher is slightly tilted but still held securely by the gripper.

(a) Chromatic Aberration Frame 1

(b) Chromatic Aberration Frame 2

(c) Clean Observation Frame 1

(d) Clean Observation Frame 2

Figure 11. First-frame conditioning comparison under the Cosmos model: the first column shows the input frame, and the second column shows the corresponding generated frame that was conditioned by the first frame. We utilize the Mean Squared error between two frames as our rejection score $M(x_*)$.

E.1.2. Cosmos Evaluation Metrics

We evaluate the denoised and base Cosmos generations using metrics from the PAI-Bench [36], a comprehensive benchmark for assessing physical-world video generation and prediction quality. Following the official protocol, we use the last frame of the ground truth clean input video as the reference image to compute the image-to-video (i2v) fidelity scores. The metrics are described as follows:

- **Aesthetic Quality.** Measures the visual appeal and perceptual realism of the generated video, including composition, color balance, and absence of artefacts.
- **Background Consistency.** Evaluates the temporal stability of background regions across frames—high scores indicate minimal flicker or spatial drift.
- **Imaging Quality.** Quantifies camera-like clarity, sharpness, and signal-to-noise ratio of the generated video frames.
- **Motion Smoothness.** Assesses the temporal coherence of motion; low values indicate jitter or unnatural frame transitions.
- **Overall Consistency.** Aggregates spatial and temporal coherence into a single holistic quality measure of the entire video.
- **Subject Consistency.** Examines whether the primary object or agent remains stable in shape, color, and identity over time.
- **i2v Background.** Image-to-video fidelity for background regions, comparing each frame to the last frame of the input video.
- **i2v Subject.** Image-to-video fidelity for the subject region, capturing how faithfully the generated subject matches the reference frame.

Higher scores across these dimensions indicate smoother, more coherent, and visually faithful video synthesis. In our experiments shown in Table 1, the proposed rejection sampling consistently improves both the i2v metrics and overall perceptual quality compared to the baseline generations.

E.2. Semantic Noise Experiments

E.2.1. Crafter Experiments

Although outside the main scope of our work, we briefly consider experimentation with semantic noise.

We choose the Crafter domain [13] due to its utility in embedding abstract skins and unknown colors during inference time, primarily meant to distract the agent.

To furnish the agent with a rich state representation, we provide and train with six complementary observation channels (visualizations provided in Figure 14):

- *Bird's Eye View*: A top-down, default view of the Crafter gameplay centered on the agent and a few blocks around it.
- *Grayscale*: A grayscale version of Bird's Eye View.
- *Semantic View*: A top-down view of the entire game map showing object locations; built-in Crafter.

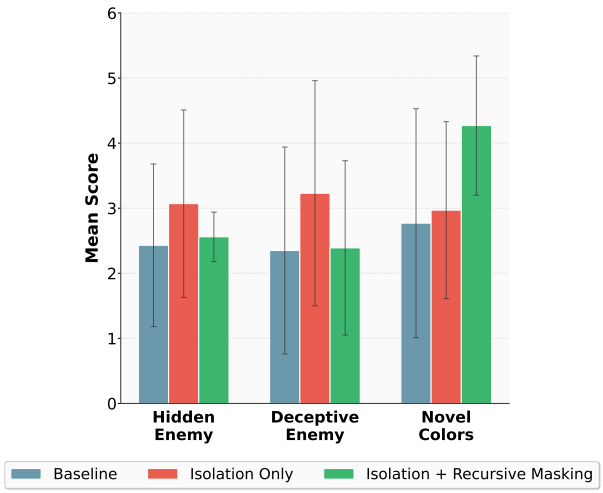
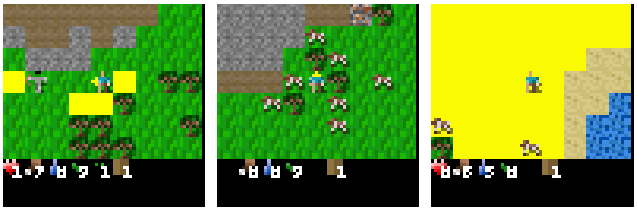


Figure 12. Crafter: Baseline vs Confident Representation Performance. Scores improve as the agent utilizes masking in Algorithm 2 to adapt to different types of novelties.



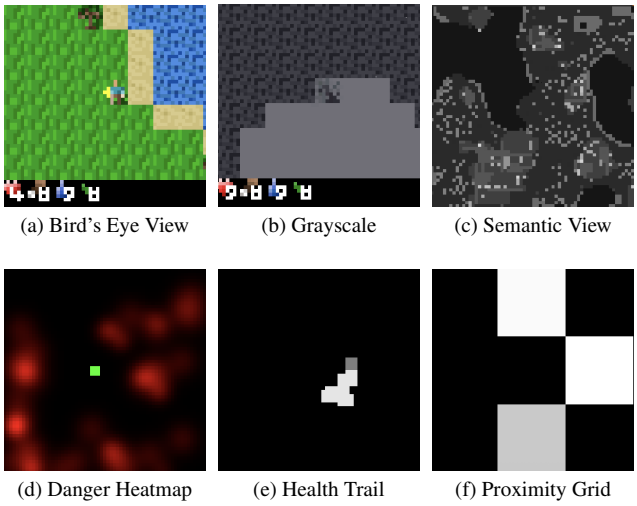
(a) Hidden Enemy (b) Deceptive Enemy (c) Novel Colors

Figure 13. Qualitative visualization of agents' behavior in different semantic novelty scenarios. Figure 13a shows the *Hidden Enemy* scenario, where a yellow blob attacks the agent like zombies despite appearing harmless on the danger map. Figure 13b shows the *Deceptive Enemy* scenario, where certain cows are reprogrammed to behave like zombies. Figure 13c shows the *Novel Colors*, in which all grass blocks turn bright yellow, creating a purely visual domain shift.

- *Danger Heatmap*: Shows sources of danger (zombies, skeletons, lava) as red with Gaussian decay; player represented with a green dot.
- *Health Trail*: Records agent health at its location through pixel intensity, widened to a 3x3 region to improve training.
- *Proximity Grid*: A 3x3 grid representing the eight cardinal directions around the agent; intensities reflect the presence of objects cumulatively, scaled up to 64x64 for training.

Unlike Safety Gymnasium and CARLA, the Crafter experiments use a single sensor (top down local view) but with multiple representations (grayscale, danger heatmap, proximity heatmap, etc.).

We first train the world model in the Crafter domain with Algorithm 1 to prepare for missing representations. To simulate unknown semantic noise, we inject unfamiliar contexts (invisible enemies, different enemy skins, visual color shifts) during inference time. In this setting, we create alterations that have the potential to conditionally affect multiple representations at once rather than independently (see Table 15 for a brief summary of expected representations affected given the semantic noise introduced).



(a) Bird's Eye View (b) Grayscale (c) Semantic View

(d) Danger Heatmap (e) Health Trail (f) Proximity Grid

Figure 14. Six different representations of the Crafter environment, capturing a diverse range of data, were used to train the agent.

We monitor the performance of a (1) Base agent, (2) an agent that only implements Step 1 *Isolation* (an agent only selects a single representation) and (3) an agent that implements Step 1 and Step 3 *Recursive Masking* (an agent that recursively masks the observations in order of surprise). We include hardware details in Appendix D.

We report our findings in Figure 13. We hypothesize that the level of depth generally affects the performance of the agent, depending on how focused the semantic noise is on a subset of representations. Recursive masking appears to improve the performance on changes that are expected to focus on a single representation, such as color based alterations that primarily affect the image representation of the state (Novel Colors). Whereas representation isolation tends to be more useful towards returning the agent towards a predictable policy when many of the representations are affected (Hidden Enemy, Deceptive Enemy, Invert Health). Although we primarily test with out-of-distribution sensor failures, we find agents equipped with Algorithm 2 appear to show some potential towards exhibiting improved stability through automated selection of representations in this setting, potentially opening avenues for future research.

E.2.2. Out Of Distribution Distractions

To reiterate our objective: *The objective of our rejection sampling mechanism is to enable the agent to produce* pre-

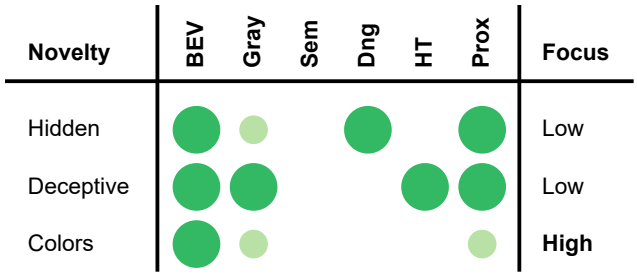
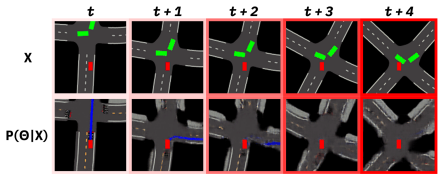


Figure 15. Representation modalities affected by each novelty type. Large circles indicate representations directly modified by the novelty, while small circles indicate representations indirectly affected through downstream effects. The Focus column indicates whether the novelty's impact is concentrated (High) or distributed across multiple modalities (Low).



	No Response	Mandated Response	Ours
Score	72.17 ± 46.32	266.07 ± 270.85	390.19 ± 134.43
Failure Rate	100.0	63.29	**7.69**

Figure 16. BEV Illustration of posterior collapse during the execution of the mandated response, with corresponding score comparisons; red indicates the magnitude of surprise/rejection score.

dictable and conservative responses *in the presence of unknown sensor failures. Since the purpose of rejection sampling is to preserve the predicted latent state from corruption, downstream policies can then implement conservative strategies—such as pulling over—minimizing the risk of unpredictable behavior occurring during their execution.*

We expect that in both sensor-corrupted and novel scenarios without prior knowledge, the agent is confronted with observations far outside its training distribution, for which it can not provide reliable grounding. This expectation is consistent with our experiments and prior literature [8]: integrating unforeseen observations into the latent state can lead to unpredictable behavior, as the primitive policy has not been trained to act upon them.

In such regimes, our method prevents latent corruption by maintaining stable representations to enable predictable policies, rather than relying on the agent's unreliable interpretation of the unknown. To highlight the above concept, we demonstrate a conservative response to a potentially novel event in the Figure 16.

We observe a key result: if a response was mandated (e.g., pulling over or redirecting) in a novel setting (such as a car accident or blockade), execution at the primitive level hinges on the stability of the latent state. Our methods maintain stability under *extreme* uncertainty, preventing latent states from collapsing into inevitable degradation.

E.3. Ablations

E.3.1. Representation Dropout Training

We further conduct an ablation study shown as Figure 17 to evaluate the effect of representation-dropout training on the stability and robustness of the world model. In this setting, a random subset of input modalities or encoded features is masked during each training step, encouraging the model to infer consistent latent dynamics even under partial observations.

Across all six tasks— PointGoal2, CarGoal1, PointButton1, Four-Lane Driving, Right Turn, and Stop Sign —we observe that the dropout-trained agents achieve comparable or slightly higher final scores than normal training while exhibiting smoother and more stable learning dynamics. Although the early stages of training progress more slowly due to random masking, the representation-dropout models converge to similar or better performance with noticeably reduced variance. This demonstrates that exposing the world model to incomplete or corrupted representations during training improves its ability to maintain coherent latent dynamics under missing-sensor or noisy conditions. The results validate that representation dropout enhances generalization and forms the foundation for the robust filtering and rejection mechanisms used in later experiments.

E.3.2. Alternative Rejection Scores $M(x_*)$

We test additional choices of $M(x_*)$ that a user may provide on the Carla Four Lane Task (Denoising is still determined by $D(x_*)$ used in the main results):

- Surprise: To decide whether to enter predictive mode, we experiment with utilizing the Bayesian surprise measure discussed in Eq 2.
- Resnet-18 [15]: To decide whether to enter predictive mode, we train a Binary Resnet-18 classifier on a synthetic dataset of observations (200,000) with and without Gaussian noise, the goal of the classifier is to reject or accept the observation given to the agent.

We report our results across two levels of noise *proportion* in Table 3.

Method	Gaussian	Chromatic	Jitter	Glare	Occlusion
Res-18. (.75)	571.9±223.6	-188.4 ± 359.9	-466.1 ± 474.7	-578.1 ± 633.9	483.5±285.9
Surp. (.75)	241.7±287.7	105.5±191.3	71.7±194.9	100.4±130.7	275.6±332.8
Res-18. (.875)	391.2±191.4	-439.1 ± 421.7	-1237.0 ± 638.3	-717.2 ± 622.4	228.6±199.5
Surp. (.875)	164.1±259.1	-18.14 ± 250.1	39.18±104.38	55.1±96.93	257.5±281.4

Table 3. Comparison of surprise and trained ResNet-18 rejectors across five noise types with different proportions (.75 and .875).

E.3.3. Alternative Denoising Methods $D(x_*)$

We test additional choices of $D(x_*)$ that a user may provide on the Carla Four Lane Task (Acceptance is still determined

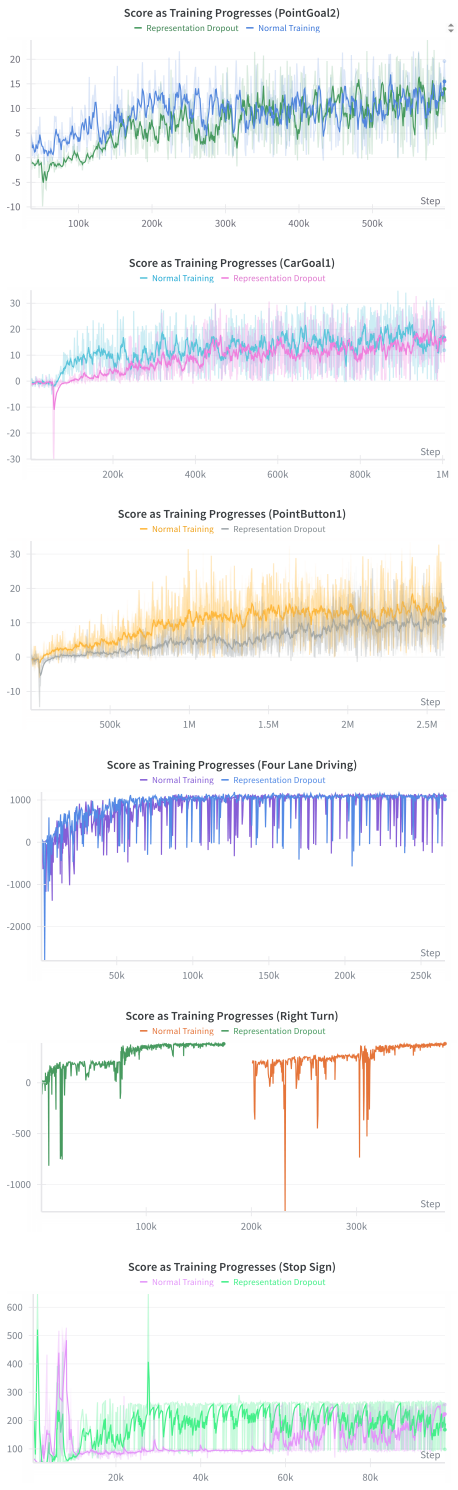


Figure 17. We compare the original world model training in Safety Gymnasium and Carla Domains to the representation dropout training.

by $M(x_*)$ used in the main results):

- Prior Interpolation: We experiment with a linear interpolation between the world model's expected observation and the current observation.
- Restormer [34]: We leverage an off-the-shelf denoiser with a transformer architecture.

We report our results across two levels of noise *intensity* in Table 4.

Method	Gaussian	Chromatic	Jitter	Glare	Occclusion
Interp (.625)	316.7±256.1	228.2±194.6	265.3±295.0	466.4±242.9	334.8±163.0
Restormer (.625)	262.7±221.1	180.1±177.3	196.1±170.1	308.8±211.7	305.7±185.4
Interp (.75)	258.3±288.6	205.6±273.4	183.8±401.4	448.2±274.3	255.9±189.9
Restormer (.75)	237.0±299.1	144.7±218.4	186.9±193.3	256.6±217.4	215.4±185.4

Table 4. Comparison of Interpolation and Restormer denoising methods across five noises types with different intensities (.625 and .75).

E.4. Additional Algorithms

Algorithm 1: Random Multi-Representation Dropout Training

Input: Data dictionary $\mathcal{D}$ with image keys K, mask value m
Output: Masked data dictionary $\mathcal{D}'$

Step 1: Select available images;
$K' \leftarrow \{k \in K : k \in \mathcal{D}\}$;
$n \leftarrow |K'|$;
if $n = 0$ **then**
  return $\mathcal{D}$

Step 2: Sample images to mask each step;
For each batch b and timestep t:;
  Draw $u_{b,t} \sim \text{Uniform}\{0, \ldots, n-1\}$;

Step 3: Assign random rankings to images;
For each (b, t):;
  Draw random values $r_{b,t,1}, \ldots, r_{b,t,n}$;
  Compute rankings $\pi_{b,t}$ by sorting these values;

Step 4: Construct masking matrix;
For each (b, t, i):;
  Set $\text{mask}[b, t, i] \leftarrow 1$ if $\pi_{b,t}(i) < u_{b,t}$, else 0;

Step 5: Apply masking;
For each $k_i \in K'$:;

$$\mathcal{D}'[k_i][b,t] \leftarrow \begin{cases} m, & \text{if mask}[b,t,i] = 1 \\ \mathcal{D}[k_i][b,t], & \text{otherwise} \end{cases};$$

return $\mathcal{D}'$;

Algorithm 2: $O(n \log n)$ Surprise-Guided Representation Selection

Input: Observation dictionary $\mathbf{obs_t}$, sensor keys K, world model WM, prev state z_{t-1}, prev action a_{t-1}, Optional Depth D
Output: Surprise values $\mathcal{S}$, latents $\mathcal{Z}$

Step 1: Compute surprise for each sensor individually;
foreach $k \in K$ **do**
    Initialize empty observation:;
    $\mathbf{obs}_{t'} \leftarrow \mathbf{0}$;
    Isolate observation sensor:;
    $\mathbf{obs}^k_{t'} \leftarrow \mathbf{obs}^k_t$;
    Encode: $e_t \leftarrow \mathrm{Encoder}(\mathbf{obs}_{t'})$;
    Predict posterior distribution:;
    $P_\phi(z^k_t | e_t, h_t) \leftarrow \mathrm{WM}(z_{t-1}, a_{t-1}, e_t)$;
    Surprise:;
    $S_k \leftarrow \mathrm{KL}\left[P_\phi(z^k_t | e_t, h_t) \parallel P_\phi(z^k_t | h_t)\right]$;
    Append S_k to $\mathcal{S}$ and $z^k_t \sim P_\phi(z^k_t | e_t, h_t)$ to $\mathcal{Z}$;

Step 2: Sort sensors by decreasing surprise;
$\pi \leftarrow \mathrm{argsort}\big([S_k]_{k \in K}\big)$;

Step 3: Iteratively mask sensors in sorted order;
$\mathbf{obs}_{t'} \leftarrow \mathbf{obs}_t$;
for $i \leftarrow 1$ ***to*** $min(|K|, D)$ **do**
    Mask sensor π_i:;
    $\mathbf{obs}^{\pi_i}_{t'} \leftarrow \mathbf{0}$;
    Encode: $e_t \leftarrow \mathrm{Encoder}(\mathbf{obs}_{t'})$;
    Predict posterior distribution:;
    $P_\phi(z^k_t | e_t, h_t) \leftarrow \mathrm{WM}(z_{t-1}, a_{t-1}, e_t)$;
    Surprise:;
    $S_k \leftarrow \mathrm{KL}\left[P_\phi(z^k_t | e_t, h_t) \parallel P_\phi(z^k_t | h_t)\right]$;
    Append S_k to $\mathcal{S}$ and $z^k_t \sim P_\phi(z^k_t | e_t, h_t)$ to $\mathcal{Z}$;
$\mathcal{Z}^* \leftarrow \arg\min_{\mathcal{Z}} \mathcal{S}(\mathcal{Z})$
return $\mathcal{Z}^*$