A. Additional Results

A.1. Ablation of SWB

We ablate two design choices of SWB. First, we investigate whether it is necessary for an object file to look at other files when computing the match probability between the file and a slot. In Table 3 (in appendix), we note that this interaction summary is necessary. In its absence, the number of MOT-A errors increase pointing to poor performance in matching files to slots correctly. In the second ablation, the slots matched to the files were directly used to update the state of a file without using an encoding of the glimpse region. We find the performance to be similar to the model that uses the glimpse. However, when training the world model in the 3D environment, we found the glimpse encoding to lead to notably more stable convergence of the model.

	Tracking		MOT-A					Generation	
Model	Position LL	Segment LL	FP	Miss	Switch	Migrate	Ascend	Position LL	Segment LL
SWB	-53.28	16553	3.24	0.27	0.07	0.00	0.01	-184.1	15797
SWB No-Match-Interact	-83.92	13481	6.01	17.3	0.81	0.78	0.83	-194.0	7662
SWB No-Glimpse	-54.26	16416	1.30	1.06	0.26	0.20	0.35	-192.3	15716

Table 3. Ablation of File-Slot Matching and Inference in SWB in 2D Branching Sprites. All models were trained with $K = 10$. In *SWB No-Match-Interaction*, we omit the file-interaction step that is performed as the first step of file-slot matching. In *SWB No-Glimpse*, the slots that were matched to the files were directly used to update the file without using a glimpse encoding.

A.2. Qualitative Results

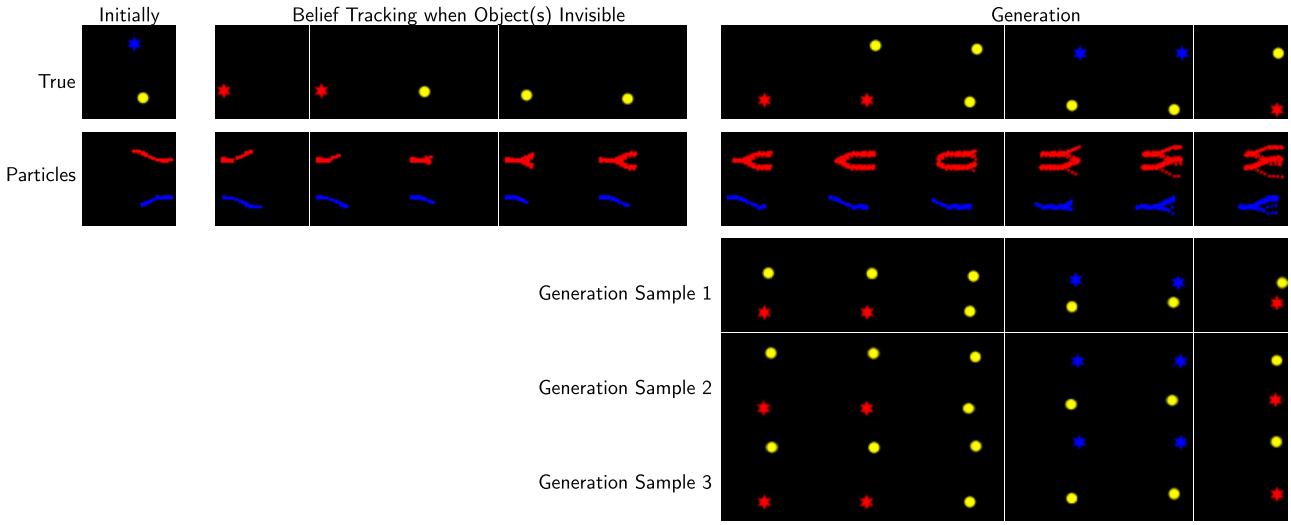


Figure 10. **Belief Tracking and Generation Samples in 2D Branching Sprites.** The aim of these qualitative results is to show generated object appearances along with object trajectories. In the top row, we show the observations. In the left block we show the object files assigned to each object. Then during belief tracking, some objects can become invisible leading to multi-modal belief. Lastly, taking the multi-modal belief, we perform generation. On the right, we see the generated object positions. We also show generated samples of object appearances. In all three samples, we see that the object appearances match what the true object appearances as shown in the top row.

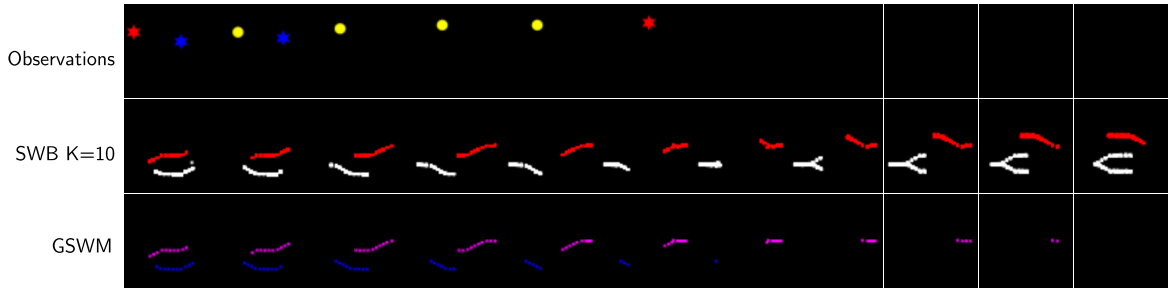


Figure 11. **Comparison of Belief Tracking in SWB and GSWM.** The top row shows the observations, the middle row shows position particles maintained by SWB with $K = 10$ and the bottom row shows the object positions maintained by GSWM. Since GSWM deletes object files for invisible objects due to lack of object permanence, we see that when objects become invisible, there is no belief in GSWM.

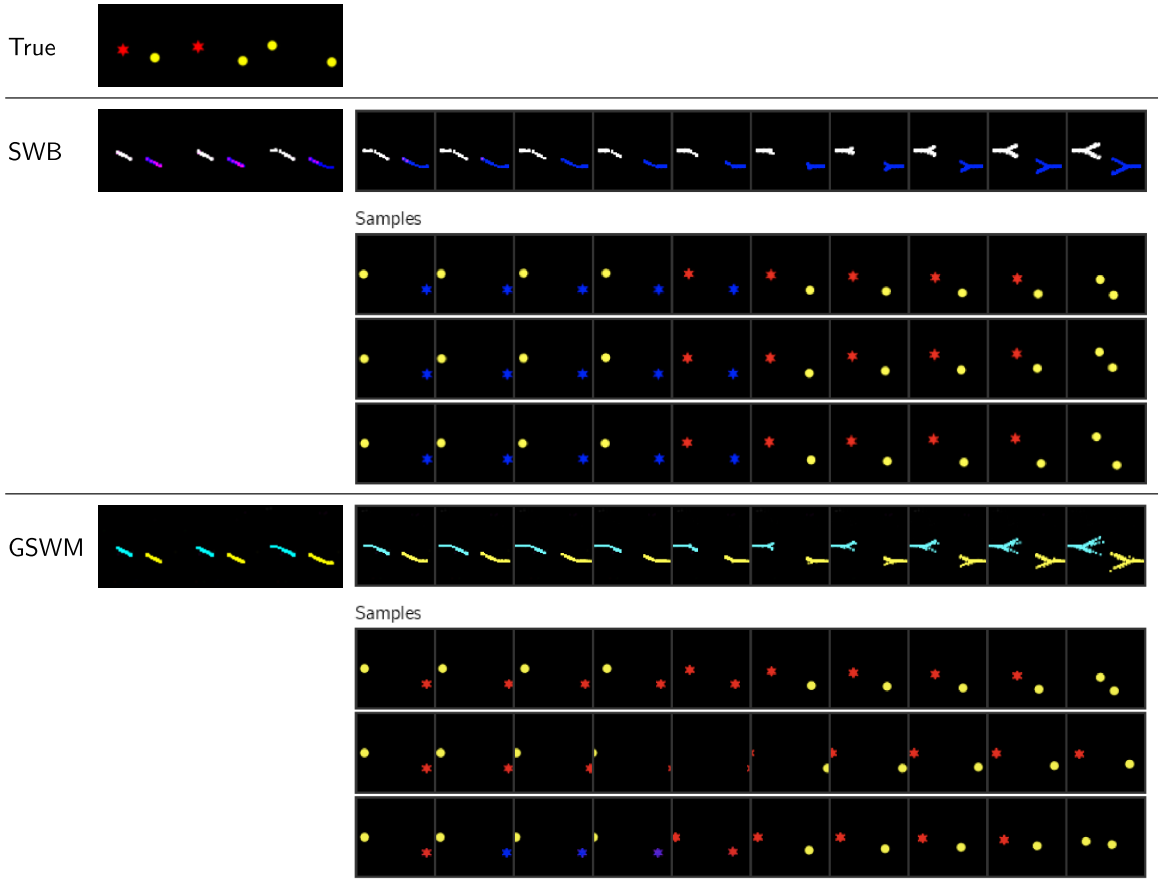


Figure 12. **Comparison of Generation in SWB and GSWM.** On the left, we show the conditioning observations and the scene representation maintained by the models. These are followed by generations paths obtained by sampling 10 samples of future trajectories. Because GSWM deletes object files for invisible objects, we made sure that the objects were visible in the conditioning period for fair comparison in this figure. We note that the generated samples from GSWM are more inaccurate.

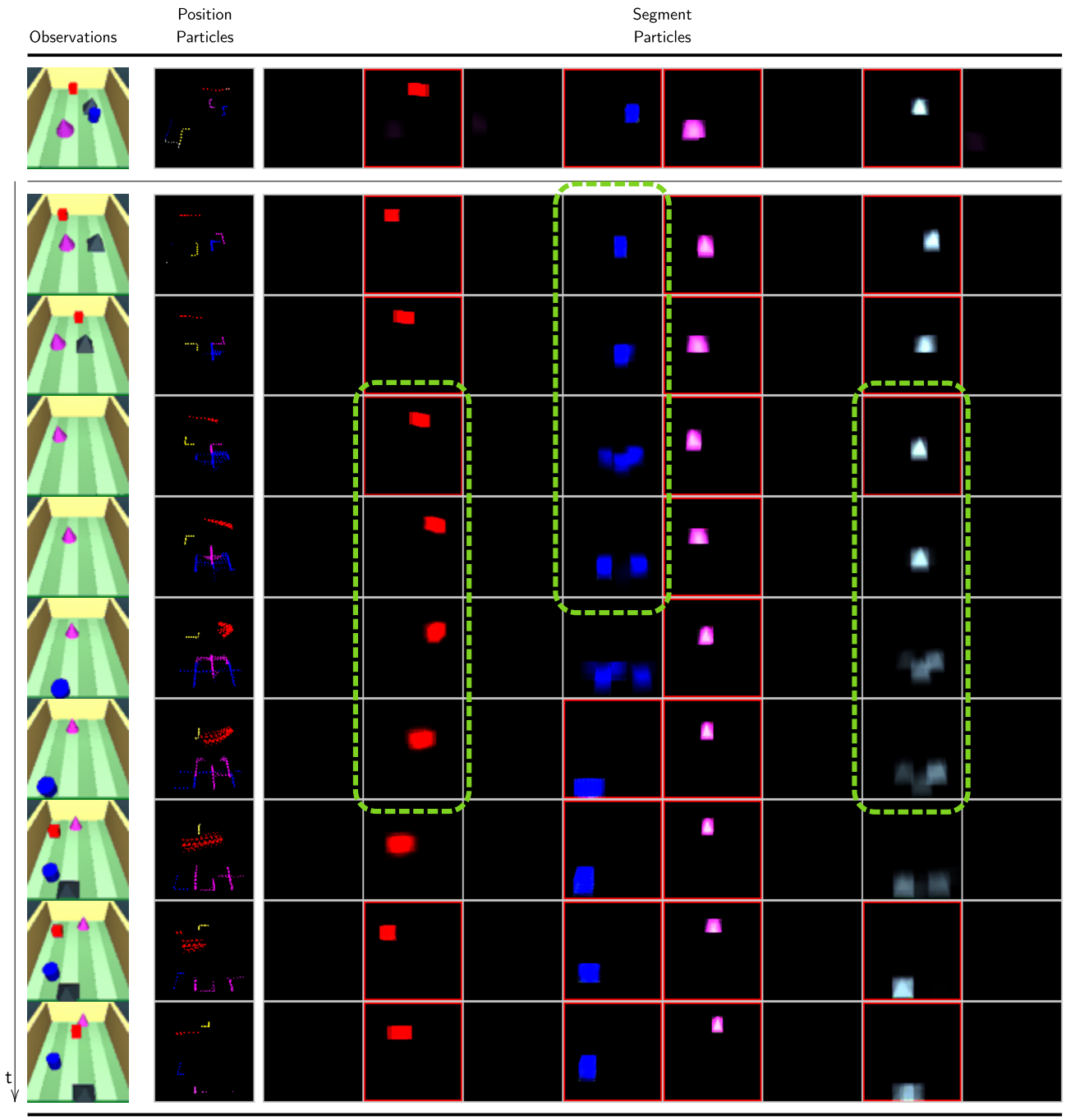


Figure 13. **Belief Tracking 3D Food Chase Game.** Each row corresponds to one time-step and we show time-steps at increments of 4. The left-most column shows the observations shown to the model. The second column shows the position particles maintained by the model. The remaining columns show the rendering of the object segments for each of the 8 object files. In these, we make the cell-border red when the belief infers the object file to be *visible* in that time-step. For each object file, we super-impose the segments for all particles by averaging over K. We note that the model can maintain belief states over the invisible objects. The dashed green cells highlight when the objects are invisible and the model is maintaining a belief of diverse and plausible states for those object files. Also note that when the objects become visible again, the correct object files can re-attach to track them.

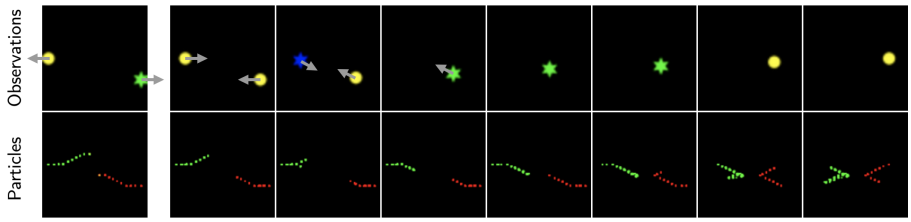


Figure 14. **Demonstration of interaction between objects in SWB.** In the observation, we see that one object becomes invisible while the other remains visible. The grey arrows show the direction of motion of the true objects. We see that invisible object file (shown as green colored particles) can simulate the collision with the visible object file (shown as red colored particles) and maintain correct belief about the invisible object.

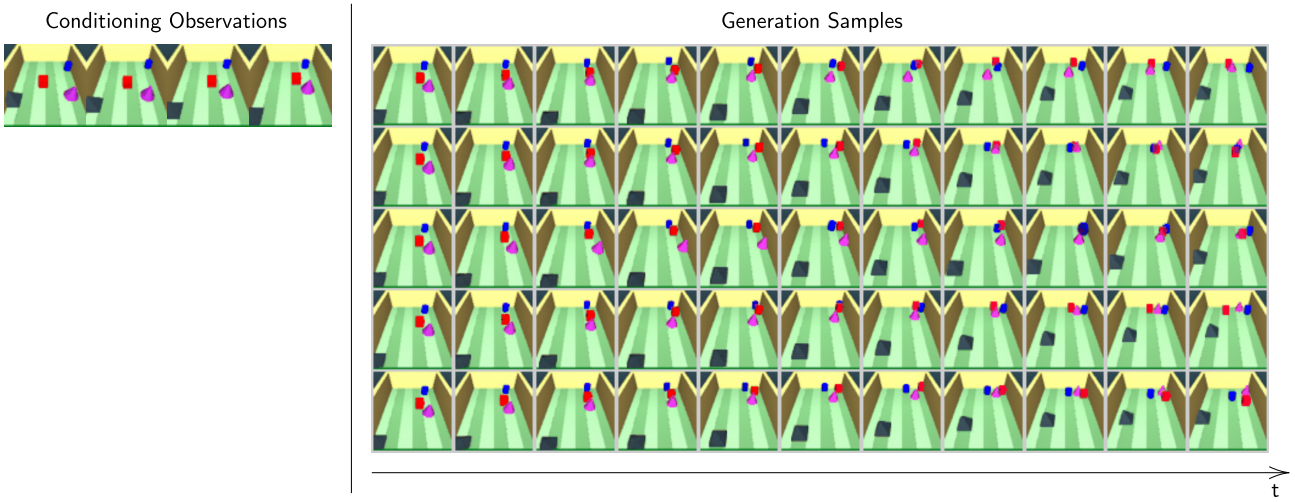


Figure 15. **Qualitative Results of Generation in SWB in 3D Food Chase Game.** On the left we show the conditioning observations. On the right, each row is one sample. Each column corresponds to time-steps at intervals of 2. We note that the generations are plausible and diverse.

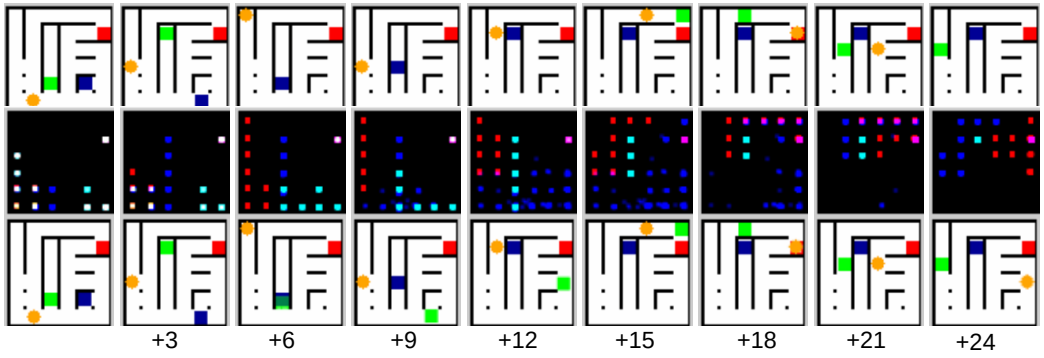


Figure 16. **Belief Tracking in 2D Maze.** In the top row, we show the observations. Middle row visualizes 8 most recent position states of each active object file. The color of the file denotes ID. The bottom row shows the reconstruction obtained from particle with index $k = 1$ at each time-step. We note in +3 that the green object is tracked by the blue object file. The green object becomes invisible in +6 and from +6 to +15, the blue object file is able to track and maintain plausible particles for it (shown in bottom row).

A.3. Additional Quantitative Results

Model	Position LL	Segment LL
SWB $K = 10$ $N = 8$	-39.18	23306
SWB $K = 20$ $N = 8$	-23.31	23353
SWB $K = 30$ $N = 8$	$\mathbf{-15.74}$	$\mathbf{23365}$
GSWM	-993.3	22457

Table 4. **Generalization to more number of objects and object files in SWB belief tracking.** We evaluate SWB trained on 2D Branching Sprites (No-Spawn-2) with 4 object files and test on 2D Branching Sprites (Spawn-4) using 8 object files. We report the average log-likelihood of ground truth position and ground truth object segments evaluated using kernel density estimation on SWB particles. We note the benefits of using higher number of particles in obtaining improved density modeling. Furthermore, we show that we outperform our baseline GSWM primarily because it does not have object permanence and does not maintain files for invisible objects.

B. Derivations

In this section, we provide the derivation of the ELBO training objective and the derivation of the belief update step.

B.1. Derivation of the ELBO

Our objective is to maximize the following marginal log-likelihood over the observations $\mathbf{x}_{1:T}$, i.e. $\log p_\theta(\mathbf{x}_{1:T})$ under our latent variable model $p_\theta(\mathbf{x}_{1:T}, \mathbf{s}_{1:T})$. Here, integrating over the high-dimensional latent $\mathbf{s}_{1:T}$ is intractable. Hence, we resort to AESMC and approximate this via an ELBO lower bound. To do this, we first re-write the objective $\log p_\theta(\mathbf{x}_{1:T})$ as:

$$\log p_\theta(\mathbf{x}_{1:T}) = \log \sum_{k=1}^{K} \frac{1}{K} p_\theta(\mathbf{x}_{1:T} | \mathbf{s}_0^k = \texttt{null}). \tag{1}$$

To evaluate this objective, we consider the following general expression and then proceed to approximate it.

$$\log \sum_{k=1}^{K} \bar{w}_{t-1}^k p_\theta(\mathbf{x}_{t:T} | \mathbf{s}_{t-1}^k). \tag{2}$$

Here, $\bar{w}_{t-1}^k$ are the normalized weights of the particles based on the observations seen thus far (bar denotes normalization), $\mathbf{s}_{t-1}^k$ are the particle states for the previous time-step and $\mathbf{x}_{t:T}$ denotes all the observations starting from the current time-step t up to the end of the training horizon T. To evaluate this objective, we first perform re-sampling of the particles $\mathbf{s}_{t-1}^{1:K}$ using the normalized weights $\bar{w}_t^{1:K}$. This is done because some particles may have vanishingly small particle weights that explain poorly the observations received so far. Hence, by performing resampling, we eliminate those particles from our particle pool.

In resampling, we sample a set of ancestor indices $a_{t-1}^k \sim q_{\text{R}}(\cdot | \bar{w}_{t-1}^{1:K})$ from a categorical distribution. In our work, we use soft-resampling as the sampling distribution q_{R} for these ancestor indices and we can write (2) as follows.

$$\log \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\text{R}}(\cdot | \bar{w}_{t-1}^{1:K})} \sum_{k=1}^{K} \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_{t:T} | \mathbf{s}_{t-1}^{a_{t-1}^k}). \tag{3}$$

where

$$q_{\text{R}}(k | \bar{w}_{t-1}^{1:K}) = \alpha \bar{w}_{t-1}^k + (1 - \alpha) \frac{1}{K}, \tag{4}$$

$$a_{t-1}^k \sim q_{\text{R}}(\cdot | \bar{w}_{t-1}^{1:K}) \quad \forall k = 1, 2, \ldots, K, \tag{5}$$

$$\tilde{w}_{t-1}^k \leftarrow \frac{\bar{w}_{t-1}^k}{\alpha \bar{w}_{t-1}^k + (1 - \alpha) \frac{1}{K}}. \tag{6}$$

where α is a trade-off hyper-parameter. It can be verified that (3) is an unbiased estimator of (2). Next, we apply Jensen's inequality on (3) to get the following.

$$\log \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \sum_{k=1}^{K} \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_{t:T}|\mathbf{s}_{t-1}^{a_{t-1}^k}) \tag{7}$$

$$\geq \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \log \sum_{k=1}^{K} \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_{t:T}|\mathbf{s}_{t-1}^{a_{t-1}^k}) \tag{8}$$

$$= \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \log \sum_{k=1}^{K} \tilde{w}_{t-1}^{a_{t-1}^k} \int p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) p_\theta(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k}) p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k)\, d\mathbf{s}_t^k \tag{9}$$

$$= \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \log \sum_{k=1}^{K} \tilde{w}_{t-1}^{a_{t-1}^k} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) \frac{p_\theta(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k})}{q_{\theta,\phi}(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k) \tag{10}$$

$$\geq \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \sum_{k=1}^{K} \underbrace{\tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) \frac{p_\theta(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k})}{q_{\theta,\phi}(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)}}_{w_t^k} p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k) \tag{11}$$

$$= \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \sum_{k=1}^{K} w_t^k p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k) \tag{12}$$

$$= \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \left(\sum_{j=1}^{K} w_t^j \right) \sum_{k=1}^{K} \frac{w_t^k}{\sum_{j=1}^{K} w_t^j} p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k) \tag{13}$$

$$= \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \left(\sum_{k=1}^{K} w_t^k \right) \tag{14}$$

$$+ \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \sum_{k=1}^{K} \frac{w_t^k}{\sum_{j=1}^{K} w_t^j} p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k) \tag{15}$$

$$= \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \left(\sum_{k=1}^{K} w_t^k \right) \tag{16}$$

$$+ \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \log \sum_{k=1}^{K} \bar{w}_t^k p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k). \tag{17}$$

Therefore, we have

$$\begin{aligned} \log \sum_{k=1}^{K} \bar{w}_{t-1}^k p_\theta(\mathbf{x}_{t:T}|\mathbf{s}_{t-1}^k) \geq & \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \underbrace{\log \left(\sum_{k=1}^{K} w_t^k \right)}_{\text{Term I}} \\ & + \mathop{\mathbb{E}}_{a_{t-1}^k \sim q_{\mathrm{R}}(\cdot|\bar{w}_{t-1}^{1:K})} \mathop{\mathbb{E}}_{\mathbf{s}_t^k \sim q_{\theta,\phi}(\mathbf{s}_1|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \underbrace{\log \sum_{k=1}^{K} \bar{w}_t^k p_\theta(\mathbf{x}_{t+1:T}|\mathbf{s}_t^k)}_{\text{Term II.}} \end{aligned} \tag{18}$$

Here, the term II can be expanded recursively by re-applying the inequality (18). Applying this inequality recursively on our original objective $\log p(\mathbf{x}_{1:T})$, we get the following ELBO.

$$\log p(\mathbf{x}_{1:T}) \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{a_{1:t-1}^k, \mathbf{s}_{1:t}^k} \log \left(\sum_{k=1}^{K} w_t^k \right). \tag{19}$$

□

B.2. Decomposing the Weight Term w_t^k

In the above derivation, we had

$$w_t^k = \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) \frac{p_\theta(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k})}{q_{\theta,\phi}(\mathbf{s}_t^k|\mathbf{s}_{t-1}^{a_{t-1}^k}, \mathbf{x}_t)} \tag{20}$$

Based on the generative model of SWB, we have

$$\begin{aligned} p_\theta(\mathbf{s}_{t,n}|\mathbf{s}_{t-1}) &= \underbrace{p_\theta(\mathbf{z}_{t,n}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1})}_{\text{Dynamics Prior}} \underbrace{p(i_{t,n}^k)}_{\text{ID Prior}}, \\ &= \underbrace{p_\theta(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1})}_{\text{Visibility Prior}} \underbrace{p_\theta(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1})}_{\text{Object Prior}} \underbrace{p(i_{t,n}^k)}_{\text{ID Prior}}. \end{aligned}$$

Additionally, for the sampled file-slot matching index $m_{t,n}$, we take a uniform categorical distribution as the prior $p(m_{t,n})$. This stems from the idea that there is no inductive bias to prefer matching one slot over another. Similarly, based on the file inference model of SWB, we have

$$q_{\theta,\phi}(\mathbf{s}_{t,n}|\mathbf{s}_{t-1}, \mathbf{x}_t) = \underbrace{q_{\theta,\phi}(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}_{\text{Object Inference}} \underbrace{q_\phi(i_{t,n}^k|\mathbf{z}_{t,n}^{\text{vis}}, i_{t-1,n}^k)}_{\text{ID Inference}} \underbrace{q_\phi(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}_{\text{Visibility Inference}} \underbrace{q_\phi(m_{t,n}^k|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}_{\text{Slot Matching}}.$$

Substituting these into (20), we get

$$w_t^k = \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) \frac{\prod_{n=1}^N p_\theta(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}) p_\theta(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}) p(i_{t,n}^k) p(m_{t,n})}{\prod_{n=1}^N q_{\theta,\phi}(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t) q_\phi(i_{t,n}^k|\mathbf{z}_{t,n}^{\text{vis}}, i_{t-1,n}^k) q_\phi(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t) q_\phi(m_{t,n}^k|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}, \tag{21}$$

$$= \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) \prod_{n=1}^N \frac{p_\theta(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}) p_\theta(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}) p(i_{t,n}^k) p(m_{t,n})}{q_{\theta,\phi}(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t) q_\phi(i_{t,n}^k|\mathbf{z}_{t,n}^{\text{vis}}, i_{t-1,n}^k) q_\phi(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t) q_\phi(m_{t,n}^k|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}, \tag{22}$$

$$= \tilde{w}_{t-1}^{a_{t-1}^k} p_\theta(\mathbf{x}_t|\mathbf{s}_t^k) \prod_{n=1}^N \underbrace{\frac{p_\theta(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1})}{q_\phi(\mathbf{z}_{t,n}^{\text{vis}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)} \frac{p_\theta(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1})}{q_{\theta,\phi}(\mathbf{z}_{t,n}^{\text{obj}}|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}}_{w_{t,n}^{k,\text{file}}} \underbrace{\frac{p(i_{t,n}^k)}{q_\phi(i_{t,n}^k|\mathbf{z}_{t,n}^{\text{vis}}, i_{t-1,n}^k)}}_{w_{t,n}^{k,\text{ID}}} \underbrace{\frac{p(m_{t,n})}{q_\phi(m_{t,n}^k|\mathbf{z}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t)}}_{w_{t,n}^{k,\text{match}}}. \tag{23}$$

□

C. Evaluation Details

C.1. Tracking and Generation Metrics

We measure the accuracy of our belief with respect to the ground truth object positions $\mathbf{y}_{t,j}^{\text{pos}}$ and segments $\mathbf{y}_{t,j}^{\text{seg}}$, where j is an object. Here, $\mathbf{y}_{t,j}^{\text{pos}} \in \mathbb{R}^2$ refers to the object coordinates and $\mathbf{y}_{t,j}^{\text{seg}} \in \mathbb{R}^{C\times W\times H}$ refers to a full-sized empty image with single object rendered on it. Note that true values of these are available even when the object is invisible. We use the particles from the models to perform kernel density estimation of the distribution over the ground truth quantities $\mathbf{y}_t$ and we evaluate and report the log-likelihood as follows.

$$\log p(\mathbf{y}_t) = \log \sum_k w_k \prod_j p(\mathbf{y}_{t,j}|\mathbf{s}_{t,n_j}^k).$$

where n_j is the object file matched with a ground truth object j at every time-step using the MOT evaluation approach in (Milan et al., 2016). If the object file $\mathbf{s}^k_{t,n_j}$ is not available due to deletion, we assign a random coordinate on the image as the object position and a black image as the object segment. We call these metrics *position log-likelihood* and *segment log-likelihood*. We also compute MOT-A components (such as False Positives or Switches) by counting them individually per particle k and taking weighted average using the particle weights w^k_t.

C.2. Environments

2D Branching Sprites. This is a 2D environment with moving sprites. To create a long-occlusion setting, the sprites can disappear for up to 40 time-steps before reappearing. To produce multi-modal position belief during the invisible period, the object paths split recursively with objects randomly taking one branch at every split (see Fig. 17). To produce belief states with appearance change during the invisible period, each sprite is associated with a pair of colors and the color switches periodically every 5 time-steps. We evaluate three versions of this environment: *i) Spawn-2:* To evaluate the model in handling new object discovery, we allow upto 2 sprites to spawn during the episode in this version. *ii) Spawn-4:* To evaluate the model in handling new object discovery, we allow upto 4 sprites to spawn during the episode in this version. We show the mean object counts during the episode in Fig. 19. *iii) No-Spawn-2:* To evaluate the accuracy of belief disentangled from the ability to handle new objects, in this version, the number of sprites remain fixed to two during the episode. We show the mean number of objects visible at each time-step in Fig. 18.

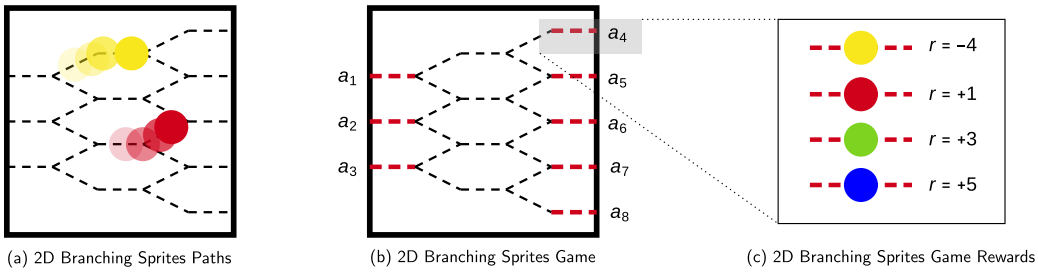


Figure 17. **2D Branching Sprites Environment Scheme.** In *(a)*, we show the paths which the sprites can take. Note the branching pattern which enables multi-modal position belief to emerge when the objects are invisible. In *(b)*, we turn the environment into a game. The edges highlighted in red can be selected as 8 + 1 actions (1 for not selecting any edge). If an object is on that edge and is also invisible, the agent receives a reward based on the color of the sprite as shown in *(c)*.

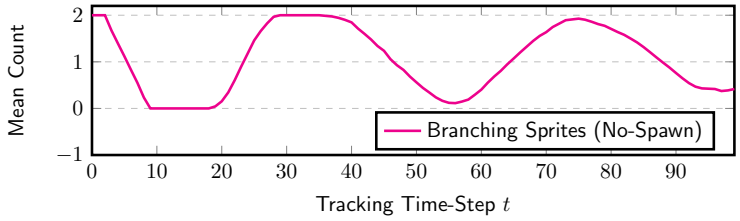


Figure 18. Mean number of objects visible for each time-step in the episode for 2D Branching Sprites (No-Spawn-2). This periodic variation takes place because the duration of invisibility is randomly sampled from $\mathrm{Uniform}(15, 20)$ and duration of visibility is sampled from $\mathrm{Uniform}(15, 30)$. The total number of objects present in the scene is 2.

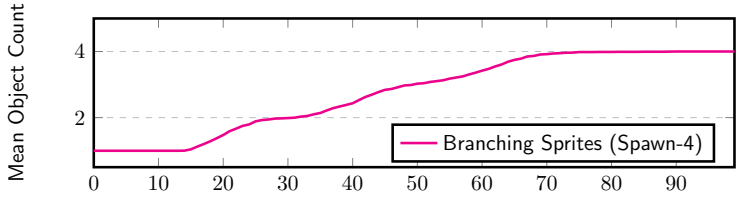
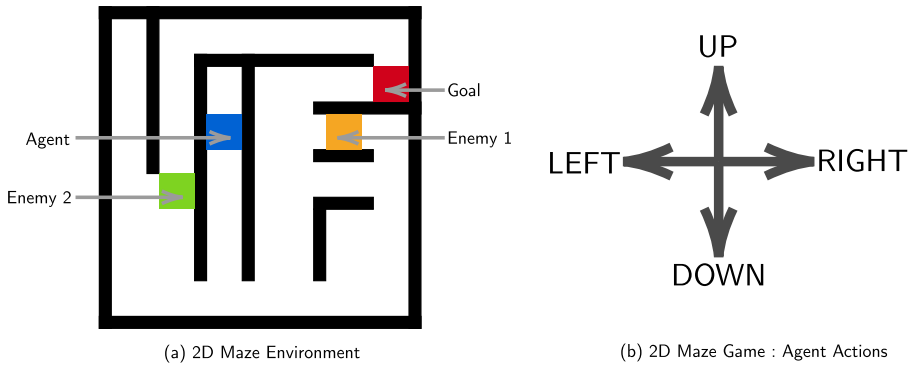


Figure 19. Mean number of objects present in the scene (visible or invisible) in the 2D Branching Sprites (Spawn-4) in which up to 4 objects spawn during the episode.

2D Branching Sprites Game. To test the benefit of our belief for agent learning, we interpret the 2D Branching Sprites environment as a game in which the agent takes an action by selecting a branch on which an invisible object is moving. This requires agent to know which branches are likely to have an object. We make the reward and penalty depend on object color. Hence, the agent also needs to accurately know the object color. The length of invisibility is sampled from range $\mathrm{Uniform}(25, 40)$ and the length of visibility is sampled from range $\mathrm{Uniform}(15, 30)$.

2D Maze Game. We consider a 2D maze environment (see Fig. 20) with the blue square as the agent, a red colored goal region and two randomly moving enemy objects. The maze is randomly generated in each episode with the objects navigating through the corridors. The enemies can disappear to create partial observability. The agent receives positive $+1$ reward for reaching the goal, -1 reward for hitting an enemy and a small positive reward of 0.025 for each step it moves to encourage exploration. The data set was created using (MattChanTK, 2020).



(a) 2D Maze Environment

(b) 2D Maze Game : Agent Actions

Figure 20. **2D Maze Game Scheme.** We illustrate the 2D Maze environment. The game comprises of an agent and a goal state which needs to be reached. There are two enemies which need to be avoided.

3D Food Chase Game. We use this environment to test our model and our agent in a visually rich 3D game. The environment is a 3D room with the agent as a red cube, food as a blue cylinder and two enemy objects. To enable multi-modal object trajectories, the food and the enemies move along lanes and can randomly change lanes at the intersections. For long invisibility, all objects can disappear for a randomly sampled duration of up to 40 time-steps. The agent receives negative reward for hitting an enemy and positive reward for eating the food. We provide higher reward for eating the food when either the agent or the food is invisible when the food is eaten. The length of invisibility is sampled from range $\mathrm{Uniform}(10, 25)$ and the length of visibility is sampled from range $\mathrm{Uniform}(20, 30)$. The data set was implemented on MuJoCo physics environment (Todorov et al., 2012).

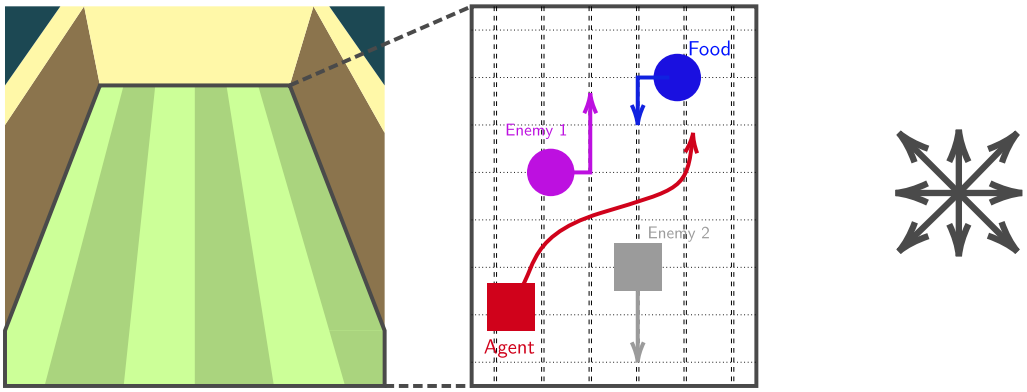
C.3. Additional Results on the Analysis of AESMC

C.3.1. 2D Billiards Dataset

We further analyse AESMC for varying number of objects in the scene and multi-modality. For this we build on the 2D Billiards task as used in GSWM. For partial observability we use object flicker and for multi-modality/randomness we have two invisible vertical reflectors at 0.33 and 0.66 of the width of the canvas. On colliding with these reflectors, the object randomly either continues to have the same velocity or reverses its velocity in the opposite direction (reflection).

C.3.2. Experiment Details

We run both AESMC and SWB with $K = 10$ particles and evaluate the positional log-likelihood as described in Section 7.1. In Figures 22, 23, 24, 25 we show qualitative results of AESMC with different objects $N = \{1, 2, 3\}$ in the presence of randomness (reflectors). We note that for smaller number of objects ($N = 1$), AESMC is able to generate coherent sequences but as we increase the number of objects (with randomness present), learning of object dynamics becomes difficult and thus generating implausible frames. We note that the capacity of the image encoder and decoder in SWB and AESMC was matched and was not the bottleneck. We hypothesize that the lack of structure in case of AESMC is the reason for its incorrect dynamics. With added randomness, the complexity in modelling the dynamics increases.



(a) 3D Food Chase Environment (b) Agent Actions

Figure 21. **3D Food Chase Game Scheme.** The game comprises of an agent, a food and two enemies. The food and the enemies move along the vertical lanes but they can randomly change to an adjacent lane at the horizontal crossings. The aim of the agent is to chase and eat the food. Once the food is eaten the food re-spawns at a random location. The actions of the agent comprise of choosing an acceleration in one of the 8 cardinal directions or choosing to not accelerate at all. There is negative reward of -2 for hitting the enemies. There is positive reward of +10 for eating the food. The reward is higher when one of the agent or the food are invisible (+20) and still higher when both of them are invisible (+30) when the food is eaten.

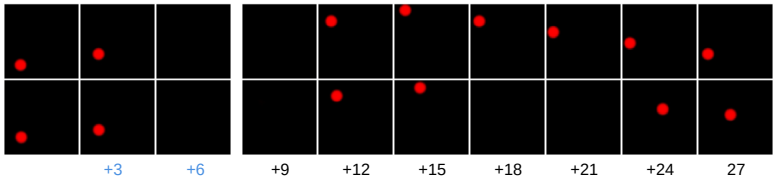


Figure 22. **AESMC Generation in 2D Billiards** We show the generation of AESMC with $N = 1$ in presence of randomness (reflectors). The top row shows the ground truth and the bottom row shows the AESMC generation. Here we find that AESMC is able to learn the dynamics well and is able to perform long term rollouts.

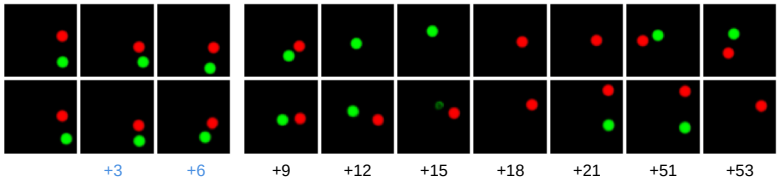


Figure 23. **AESMC Generation in 2D Billiards** We show the generation of AESMC with $N = 2$ objects and in presence of randomness. The top row shows the ground truth and the bottom row shows the AESMC generation. Time steps shown in blue are the conditioned images and the time steps shown in black are generation. We observe that as compared to $N = 1$ case, the dynamics are less coherent. For example the motion of objects in between timesteps $+6$ and $+9$ is deterministic but there is deviation in the objects' location compared to the ground truth after time step $+9$.

D. SPACE-based Implementation of Structured World Belief

We implement our model by adopting the object-centric representation of SPACE and dynamics modeling of GSWM as our framework. Hence, we decompose the object states in our object files as:

$$\mathbf{z}_{t,n}^{\text{k, obj}} = (\mathbf{z}_{t,n}^{\text{k, depth}}, \mathbf{z}_{t,n}^{\text{k, scale}}, \mathbf{z}_{t,n}^{\text{k, position}}, \mathbf{z}_{t,n}^{\text{k, velocity}}, \mathbf{z}_{t,n}^{\text{k, what}}, \mathbf{z}_{t,n}^{\text{k, dynamics}}) \tag{24}$$

Depth is used to resolve occlusions during rendering. *Scale* is the size of the bounding box that encloses the object. *Position* is the center position of the bounding box. *Velocity* is the difference between the last object position and the current object position, representing the velocity. *What* is a distributed vector representation that is used to decode the object glimpse that is rendered inside the bounding box. *Dynamics* is a distributed vector representation which encodes noise that enables

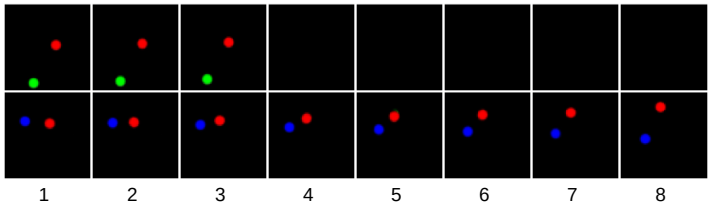


Figure 24. **AESMC Generation in 2D Billiards** We show the generation of AESMC with $N = 3$ objects in presence of randomness. The top row shows the ground truth and the bottom row shows the AESMC generation. Here, we show an example in which the dynamics of red ball is not accurate. After $t = 3$, the ball changes it's trajectory abruptly and moves upwards.

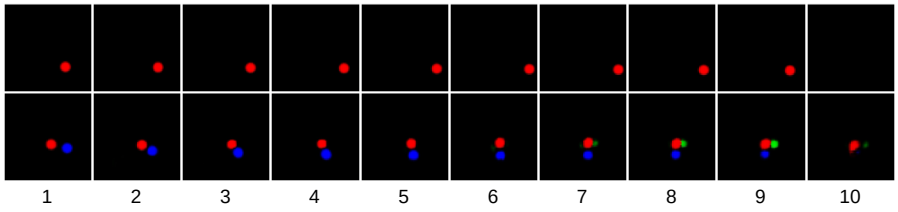


Figure 25. **AESMC Generation in 2D Billiards** We show the generation of AESMC with $N = 3$ objects in presence of randomness. The top row shows the ground truth and the bottom row shows the AESMC generation. Here, we show an example in which the model generates incoherent objects at $t = 8$. Additionally, the dynamics of the blue and red balls is incorrect. The red ball stops after colliding with the blue ball.

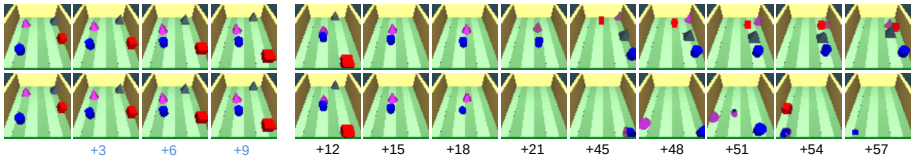


Figure 26. **AESMC Generation in 3D Task** The top row shows the ground truth and the bottom row shows the AESMC generation. We observe that for example at $t = 51$, the model is confused about the position of the pink pyramid. Also, there is an abrupt of position of the blue cylinder from extreme right at $t = 51$ to extreme left at $t = 54$, which is incorrect.

multi-modal randomness in object trajectory during generation.

D.1. Generation

Here, we describe the process of doing future generation starting from a given file state $\mathbf{s}^k_{t,1:N}$. For brevity of the description of the generative process, we will drop the superscript k.

Background Generation. We treat the background to be a special object file which is always visible. Furthermore, its position is the center of the image and its scale stretches over the entire image. We use a background module similar to GSWM (Lin et al., 2020a). The generative process for the background takes the background object file $\mathbf{z}^{\text{bg}}_{t-1}$ from the previous time-step and predicts the sufficient statistics (mean and sigma) for the new background object file. This is done as follows.

$$\boldsymbol{\mu}^{\text{bg, prior}}_{t,n}, \boldsymbol{\sigma}^{\text{bg, prior}}_{t,n} = \text{MLP}^{\text{bg}}_{\theta}(\mathbf{z}^{\text{bg}}_{t-1}).$$

Then the new background state is sampled as follows.

$$\mathbf{z}^{k,\text{bg}}_{t} \sim \mathcal{N}(\boldsymbol{\mu}^{\text{bg, prior}}_{t,n}, \boldsymbol{\sigma}^{\text{bg, prior}}_{t,n}) \tag{25}$$

The background is rendered using a CNN with deconvolution layers and sigmoid output activation.

$$\mathbf{y}^{\text{bg}}_{t} = \text{CNN}^{\text{RenderBG}}_{\theta}(\mathbf{z}^{\text{bg}}_{t}) \tag{26}$$

We then obtain separate background contexts for each object file to condition the generation. For this, a bounding box region around the last object file position is used to crop the background RGB map.

$$\mathbf{y}^{\text{bg}}_{t,n} = \text{crop}(\mathbf{y}^{\text{bg}}_{t}, \mathbf{z}^{\text{scale}}_{t,n} + \Delta_{\text{bg, proposal}}, \mathbf{z}^{\text{position}}_{t,n}) \tag{27}$$

where $\Delta_{\text{bg, proposal}}$ is a hyper-parameter taken as 0.25 in this work. The cropped context is then encoded using a CNN as follows.

$$\mathbf{e}^{\text{bg}}_{t,n} = \text{CNN}^{\text{EncodeBGProposal}}_{\theta}(\mathbf{y}^{\text{bg}}_{t,n}) \tag{28}$$

Foreground Generation. Then we perform file-file interaction of the foreground object files using a graph neural network. This approach was also taken in (Kossen et al., 2019; Lin et al., 2020a; Veerapaneni et al., 2019). However, these files also interact with the neighborhood region of the background through conditioning on the background context (Lin et al., 2020a) obtained in the previous step.

$$\boldsymbol{e}^{\text{file-interaction}}_{t,n} = \text{GNN}_{\theta}(\mathbf{z}_{t-1,1:N}, \mathbf{h}_{t-1,1:N}, \mathbf{e}^{\text{bg}}_{t,1:N}).$$

We then compute sufficient statistics for the random variables that need to be sampled.

$$\boldsymbol{\mu}^{\text{obj}}_{t,n}, \boldsymbol{\sigma}^{\text{obj}}_{t,n}, \boldsymbol{\rho}^{\text{vis}}_{t,n} = \text{MLP}_{\theta}(\boldsymbol{e}^{\text{file-interaction}}_{t,n}).$$

Next, we sample the random variables.

$$\begin{aligned}
\mathbf{z}^{\text{vis}}_{t,n} &\sim \text{Bernoulli}(\cdot|\boldsymbol{\rho}^{\text{vis}}_{t,n}), \\
\mathbf{z}^{\text{obj}}_{t,n} &\sim \mathcal{N}(\boldsymbol{\mu}^{\text{obj}}_{t,n}, \boldsymbol{\sigma}^{\text{obj}}_{t,n}).
\end{aligned}$$

We update the RNN as follows.

$$\mathbf{h}^{k}_{t,n} = \text{RNN}_{\theta}(\mathbf{z}^{k}_{t,n}, \mathbf{h}^{k}_{t-1,n}).$$

We carry over the previous object ID.

$$i_{t,n} = i_{t-1,n}.$$

D.2. Inference

D.2.1. Image Encoder

We feed the image to a CNN. We interpret the output feature cells as objects.

$$\boldsymbol{u}^{\text{CNN}}_{t,1:G\times G} = \text{CNN}_{\phi}(\mathbf{x}_t).$$

where $G \times G$ is the number of output cells of the CNN encoder.

Like SPACE (Lin et al., 2020b), we interpret each feature $\boldsymbol{u}^{\text{CNN}}_{t,m}$ to be composed of object presence, position and scale of object bounding box and other inferred features.

$$\boldsymbol{u}^{\text{CNN}}_{t,m} = (\boldsymbol{u}^{\text{pres,CNN}}_{t,m}, \boldsymbol{u}^{\text{position,CNN}}_{t,m}, \boldsymbol{u}^{\text{scale,CNN}}_{t,m}, \boldsymbol{u}^{\text{features,CNN}}_{t,m}).$$

We pick the best M features based on the value of $\boldsymbol{u}^{\text{pres,CNN}}_{t,m}$. This results in the slots that we require for file-slot matching.

$$\boldsymbol{u}_{t,1:M} = \text{SelectTopM}(\boldsymbol{u}^{\text{CNN}}_{t,1:G\times G}, \texttt{key} = \boldsymbol{u}^{\text{pres,CNN}}_{t,1:G\times G}, \texttt{count} = M).$$

To this set of slots, we add a `null` slot when the file-slot matching need not attach to any of the detected features.

$$\boldsymbol{u}_{t,0:M} = \mathbf{u}_{t,0} \cup \boldsymbol{u}_{t,1:M}.$$

D.2.2. File-Slot Attachment and Glimpse Proposal

In this step, we want each previous object file to perform attention on one of the slots provided by the image encoder. This is done by sampling a stochastic index $m^k_{t,n}$ for each previous object file $\mathbf{s}^k_{t,n}$ as described in Algorithm 1.

In our SPACE-based implementation, we take the match index and obtain the proposal bounding box as:

$$\mathbf{o}^{k,\text{proposal}}_{t,n} = (\boldsymbol{u}^{\text{position}}_{t,m^k_{t,n}}, \boldsymbol{u}^{\text{scale}}_{t,m^k_{t,n}}) \tag{29}$$

Using the proposal, we crop the image to get $\mathbf{x}^{k,\text{crop}}_{t,n}$.

$$\mathbf{x}^{k,\text{crop}}_{t,n} = \text{crop}(\mathbf{x}_t, \mathbf{o}^{k,\text{proposal}}_{t,n}) \tag{30}$$

We then encode this cropped patch using a CNN to get a glimpse encoding $\mathbf{e}^{k,\text{glimpse}}_{t,n}$.

$$\mathbf{e}^{k,\text{glimpse}}_{t,n} = \text{CNN}_\phi(\mathbf{x}^{k,\text{crop}}_{t,n}) \tag{31}$$

D.2.3. State Inference

We then use the glimpse encoding to infer new state of the object files. This is done as follows.

Background Inference. We will first infer and render the background. The inference process takes the input image and feeds it to an encoder CNN to obtain an image encoding.

$$\mathbf{e}^{\text{bg}}_t = \text{CNN}^{\text{EncodeBG}}_\phi(\mathbf{x}_t) \tag{32}$$

The encoding is concatenated with the object file of the background from the previous time-step and fed to an MLP to predict the sufficient statistics (mean and sigma) for the inference distribution.

$$\boldsymbol{\mu}^{k,\text{bg, post}}_t, \boldsymbol{\sigma}^{k,\text{bg, post}}_t = \text{MLP}^{\text{bg}}_\phi(\mathbf{z}^{k,\text{bg}}_{t-1}, \mathbf{e}^{\text{bg}}_t).$$

Then the new background state is sampled as follows.

$$\mathbf{z}^{k,\text{bg}}_t \sim \mathcal{N}(\boldsymbol{\mu}^{k,\text{bg, post}}_t, \boldsymbol{\sigma}^{k,\text{bg, post}}_t) \tag{33}$$

Lastly, we render the background using a CNN with deconvolution layers and sigmoid output activation.

$$\mathbf{y}^{\text{bg}}_t = \text{CNN}^{\text{RenderBG}}_\theta(\mathbf{z}^{\text{bg}}_t) \tag{34}$$

We then obtain separate background contexts for each object file to condition the inference. For this, a bounding box region around the last object file position is used to crop the background RGB map.

$$\mathbf{y}^{k,\text{bg}}_{t,n} = \text{crop}(\mathbf{y}^{\text{bg}}_t, \mathbf{z}^{\text{k, scale}}_{t,n} + \Delta_{\text{bg, proposal}}, \mathbf{z}^{\text{k, position}}_{t,n}) \tag{35}$$

where $\Delta_{\text{bg, proposal}}$ is a hyper-parameter taken as 0.25 in this work. The cropped context is then encoded using a CNN as follows.

$$\mathbf{e}^{k,\text{bg}}_{t,n} = \text{CNN}^{\text{EncodeBGProposal}}_\theta(\mathbf{y}^{k,\text{bg}}_{t,n}) \tag{36}$$

Foreground Inference: We first perform file-file interaction for the foreground object files using a graph neural network. This approach was also taken in (Kossen et al., 2019; Lin et al., 2020a; Veerapaneni et al., 2019) to model object interactions such as collisions in physical dynamics. This interaction also takes into account the situation of the objects in the context of the background.

$$\boldsymbol{e}^{k,\text{file-interaction}}_{t,n} = \text{GNN}_\theta(\mathbf{z}^k_{t-1,1:N}, \mathbf{h}^k_{t-1,1:N}, \mathbf{e}^{k,\text{bg}}_{t,1:N}).$$

We then compute sufficient statistics for the random variables that need to be sampled.

$$\boldsymbol{\mu}_{t,n}^{k,\text{obj, post}}, \boldsymbol{\sigma}_{t,n}^{k,\text{obj, post}}, \boldsymbol{\rho}_{t,n}^{k,\text{vis, post}} = \text{MLP}_\phi(\boldsymbol{e}_{t,n}^{k,\text{file-interaction}}, \mathbf{e}_{t,n}^{k,\text{glimpse}}).$$

We then compute sufficient statistics for the prior distribution.

$$\boldsymbol{\mu}_{t,n}^{k,\text{obj, prior}}, \boldsymbol{\sigma}_{t,n}^{k,\text{obj, prior}}, \boldsymbol{\rho}_{t,n}^{k,\text{vis, prior}} = \text{MLP}_\theta(\boldsymbol{e}_{t,n}^{k,\text{file-interaction}}).$$

Next, we sample the random variables.

$$\mathbf{z}_{t,n}^{k,\text{vis}} \sim \text{Bernoulli}(\cdot|\boldsymbol{\rho}_{t,n}^{k,\text{vis, post}}).$$

Next, we sample the remaining random variables. We perform object-wise imagination or bottom-up inference depending on whether the object is visible or not.

$$\mathbf{z}_{t,n}^{k,\text{obj}} \sim \mathcal{N}(\cdot|\boldsymbol{\mu}_{t,n}^{k,\text{obj, post}}, \boldsymbol{\sigma}_{t,n}^{k,\text{obj, post}})^{\mathbf{z}_{t,n}^{k,\text{vis}}} \mathcal{N}(\cdot|\boldsymbol{\mu}_{t,n}^{k,\text{obj, prior}}, \boldsymbol{\sigma}_{t,n}{}^{k,\text{obj, prior}})^{1-\mathbf{z}_{t,n}^{k,\text{vis}}}.$$

We update the RNN as follows using the sampled object states.

$$\mathbf{h}_{t,n}^k = \text{RNN}_\theta(\mathbf{z}_{t,n}^k, \mathbf{h}_{t-1,n}^k).$$

To update the ID, if the file is set to visible for the first time, we assign a new ID to the file. Otherwise, we simply carry over the previous ID. That is inactive file remains inactive. And active file remains active with the same ID.

$$i_{t,n}^k = \begin{cases} \texttt{new\_id()} & \text{if } \mathbf{z}_{t,n}^{k,\text{vis}} = 1 \text{ and } i_{t-1,n}^k = \texttt{null} \\ i_{t-1,n}^k & \text{otherwise} \end{cases}$$

D.2.4. Position Prediction via Velocity Offset

Similarly to STOVE (Kossen et al., 2019) and GSWM (Lin et al., 2020a), we predict the position (both during inference and generation) by adding *velocity* $\mathbf{z}_{t,n}^{k,\text{velocity}}$ to the previous object position. All other attributes are directly predicted without adding offsets.

$$\mathbf{z}_{t,n}^{k,\text{position}} = \mathbf{z}_{t-1,n}^{k,\text{position}} + \mathbf{z}_{t,n}^{k,\text{velocity}} \tag{37}$$

D.2.5. Rendering

Our final image is rendered similarly to SPACE and GSWM (Lin et al., 2020b;a). Rendering process takes the object files for the background $\mathbf{z}_t^{k,\text{bg}}$ and the foreground objects $\mathbf{z}_{t,1:N}^k$ and returns pixel-wise means of the RGB values in the final image. The sigma is set as a hyper-parameter.

$$\boldsymbol{\mu}_t^{k,\text{rendered}} = \text{render}_\theta(\mathbf{z}_t^{k,\text{bg}}, \mathbf{z}_{t,1:N}^k) \tag{38}$$

$$\boldsymbol{\sigma}_t^{k,\text{rendered}} = \text{Set as hyperparameter.} \tag{39}$$

D.3. Training

In this section, we describe details related to training that were not described in the main text.

D.3.1. Curriculum

For training iterations up to 3000, we train the image encoder using the auto-encoding objective of SPACE. This allows image encoder to learn to detect objects in images. After 3000 iterations, the SPACE objective is removed. Hereafter, the output of image encoder is directly treated as M slots which are provided as input for file-slot matching as described in Appendix D.2.1 and in the main text. The encoder is jointly trained with the rest of the model.

D.3.2. SOFT-RESAMPLING

To prevent low-weight particles from propagating, we resample the particles based on the particle weights. However, when we resample the particles from the distribution $q^{\text{resampler}}(k) = w_t^k$, we obtain a non-differentiable sampling process which prevents gradient from flowing back. This problem is commonly addressed by applying soft-resampling (Karkus et al., 2017) as follows:

$$q^{\text{resampler}}(k) = \alpha w_t^k + (1-\alpha)\frac{1}{K} \tag{40}$$

$$\text{pa}(k) \sim q^{\text{resampler}}(\cdot), \forall k \tag{41}$$

$$\mathbf{s}_{t,n}^k \leftarrow \mathbf{s}_{t,n}^{\text{pa}(k)} \text{ and } w_t^k \leftarrow \frac{w_t^{\text{pa}(k)}}{\alpha w_t^{\text{pa}(k)} + (1-\alpha)\frac{1}{K}} \tag{42}$$

where α is a trade-off parameter. We choose $\alpha = 0.6$ during training.

D.3.3. HYPER-PARAMETERS

Parameter	2D Branching Sprites	2D Maze	3D Food Chase
Image Width/Height	64	64	64
N	4	8	8
K	10	10	10
Learning Rate	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam
Image Encoder Grid Cells	8×8	16×16	16×16
Background Module	Off	On	On
Glimpse Width/Height	16	16	16
Glimpse Encoding Size	128	128	128
RNN hidden state size	128	128	128
`depth` attribute size	1	1	1
`scale` attribute size	2	2	2
`position` attribute size	2	2	2
`offset` attribute size	2	2	2
`what` attribute size	32	32	32
`dynamics` attribute size	8	8	8
Reconstruction Sigma	0.05	0.075	0.1

Table 5. Model Specifications