A. Algorithms

A.1 Iterative inference algorithm

Algorithm 1: Iterative Inference Algorithm

Input: observation x, viewpoint v, latent Gaussian parameters $\boldsymbol{\lambda}^t = \{(\mu_k^t, \sigma_k^t)\}$
ModelParameters Φ, θ, *and the number of single-view iterations* $L(default:5)$
Initialize $\boldsymbol{\lambda}^{t(l)} = \boldsymbol{\lambda}^t$, $\boldsymbol{ELBO}^t = 0$
for $l = 1$ ***to*** L **do**
  $\mathbf{z^{t(l)}} \sim \mathcal{N}(\mathbf{z}^{t(l)};\, \boldsymbol{\lambda}^{t(l)})$; `// sample from a prior--make a guess`
  $p_\theta(x^{t(l)}|\mathbf{z}^{t(l)}, v) = g_\theta(\mathbf{z^{t(l)}}, v)$; `// render and verify`
  $\boldsymbol{ELBO}^{t(l)} = -\log p_\theta(x^{t(l)}|\mathbf{z}^{t(l)}, v) + \mathcal{D}_{\mathrm{KL}}(\mathcal{N}(z^t\,; \boldsymbol{\lambda}^{t(l)})||\mathcal{N}(z^t\,; \boldsymbol{\lambda}^t))$;
  $\boldsymbol{\lambda}^{t(l)} = \Phi(x, \boldsymbol{ELBO}^{t(l)}, \boldsymbol{\lambda}^{t(l)})$; `// refine and then repeat (until` $l = L$`)`
  $\boldsymbol{ELBO}^t += (1/L) \cdot \boldsymbol{ELBO}^{t(l)}$
Output $\boldsymbol{ELBO}^t, \boldsymbol{\lambda}^{t(l)} = \{(\mu_k^{t(l)}, \sigma_k^{t(l)})\}$

A.2 Testing algorithm

Algorithm 2: DyMON Testing Algorithm

Input: Trained parameters Φ, θ, and latent Gaussian parameters $\boldsymbol{\lambda}^0 = \{(\mu_k = \mathbf{0}, \sigma_k = \mathbf{I})\}$
Initialize $\boldsymbol{\lambda}^t = \boldsymbol{\lambda}^0$;
while Access (x^t, v^t) **do**
  $\boldsymbol{ELBO}^t$, $\boldsymbol{\lambda}^t = \mathbf{iterative\_inference}_{\Phi,\theta}(x^t, v^t, \boldsymbol{\lambda}^t)$;
  Output $\boldsymbol{\lambda}^t = \{(\mu_k^t, \sigma_k^t)\}$;

B. Implementation Details

B.1 Training configurations

We show the training configurations used in this work in Table 1.

Table 1: Training Configurations

TYPE	THE TRAININGS OF DYMON, MULMON, GSWM
OPTIMIZER	ADAM
INITIAL LEARNING RATE η_0	$3e^{-3}$
LEARNING RATE AT STEP s	$\max\{0.1\eta_0 + 0.9\eta_0 \cdot (1.0 - s/1e^6), 0.1\eta_0\}$
TOTAL GRADIENT STEPS	$300k$ FOR DYMON VS. GSWM, $200k$ FOR DYMON VS. MULMON
BATCH SIZE	2 ($2\,seqs \times 40\,images = 80\,images$)
NUMBER OF GPU/PER TRAINING	1 ($Mem >= 11GB$)
* THE SAME SCHEDULER AS THE ORIGINAL GQN EXCEPT FOR FASTER ATTENUATION	

B.2 Model implementation

We show the designs of the generative mapping function g_θ and the refinement function in Table 2 and 3 respectively. After obtaining a set of K RGBM outputs from this function, i.e. $\{(\mu_{xk}, \hat{m}_{xk})\}$ (see Table 2), we render (i.e. compose) an image as: $x = \sum_k \mathbf{softmax}(\hat{m}_{xk}) \cdot x_k$, where $x_k \sim \mathcal{N}(x_k;\, \mu_{xk}, 0.1^2\mathbf{I})$,

Table 2: Generator function g_θ

Parameters	Type	Channels (out)	Activations.	Descriptions
θ^1 (projection)	Input	$D+d$		$\mathbf{z}^t \sim \mathcal{N}(\mathbf{z}^t; \boldsymbol{\lambda}^t)$, v^t
	Linear	256	Relu	
	Linear	D	Linear	$\tilde{\mathbf{z}^t} = g_{\theta_1}(\mathbf{z}^t, v^t)$
θ^2 (rendering)	Input	D		$\tilde{z_k^t} = g_{\theta_2}(z_k^t, v^t)$
	Broadcast	D+2		* Broadcast to grid
	Conv 3×3	32	Relu	
	Conv 3×3	32	Relu	
	Conv 3×3	32	Relu	
	Conv 3×3	32	Relu	
	Conv 3×3	4	Linear	RGBM: rgb μ_{xk} + mask logits $\hat{m}_{xk}$

D: the dimension of a latent representation, set to 16 for all experiments
d: the dimension of a viewpoint vector, set to 3 for all experiments
*: see spatial broadcast decoder [6]
Stride$=1$ set for all Convs.

Table 3: Refinement Network Φ

Parameters	Type	Channels (out)	Activations.	Descriptions
Φ	Input	17		* Auxiliary inputs $\mathbf{a}(x^t)$
	Conv 3×3	32	Relu	
	Conv 3×3	32	Relu	
	Conv 3×3	64	Relu	
	Conv 3×3	64	Relu	
	Flatten			
	Linear	256	Relu	
	Linear	128	Linear	
	Concat	128+4*D		
	LSTMCell/GRUCell	128		
	Linear	128	Linear	output $\Delta\lambda$

D: the dimension of a latent representation, set to 16 for all experiments
Stride$=1$ set for all Convs.
* see IODINE[3] for details
LSTMCell/GRUCell channels: the dimensions of the hidden states

C. Datasets

C.1 DRoom (DynamicRoom)

Simulation Environment We created the DRoom simulation on the top of the CLEVR Blender environment [4, 1]. Like other multi-object datasets [2], we initialized every sequence by randomly selecting and placing 2-5 scene objects in a simulated room (with background and walls specified). These objects are randomized in terms of shapes (incl. deformations, sizes), colors, and textures. Under the Blender physics engine settings, we enabled foreground objects' movements by setting their dynamics status to "active" and disabled the background objects' (i.e. walls and ground's) movements by setting their dynamics status to "passive". We then created a centrifugal force field within a fixed center and range on the ground across all DRoom datasets. In this work, we sample the magnitude of the force using: $\mathbf{random.choice}(\text{vals} = 8500 \times \{0, 0.1, 0.2, ..., 1\}, \text{probs} = \mathcal{C}at(...))$, which allows us to simulate scene object motions of different speeds by inputing different selection categorical probability $\mathcal{C}at(...)$. Moreover, we enabled object collisions to simulate scenes with rather complex object dynamics. The control of the observer (an RGB camera) motion is independent of the scene objects. We consider an observer or camera performing random walks on the surface of a dome (top half of a sphere) whose center aligns with the center of the ground—we randomly initialize the starting position of a camera and randomly sample its next move. Note that as the camera can only move on the dome (with a fixed radius r), we can use azi and ele, i.e. the

azimuth and elevation of the camera, to represent a camera location. We sample the increment Δazi and Δele independently from: $\textbf{random.choice}_{azi}(\text{vals} = 5.0\,degs \times \{0, 0.1, 0.2, ..., 1\}, \text{probs} = \mathcal{C}at_{azi}(...))$ and $\textbf{random.choice}_{ele}(\text{vals} = 1.0\,degs \times \{0, 0.1, 0.2, ..., 1\}, \text{probs} = \mathcal{C}at_{ele}(...))$, which suggests that we can control the speed of the camera by inputting different $\mathcal{C}at_{azi}(...))$ and $\mathcal{C}at_{ele}(...))$.

Figure 1: **Left:** DRoom simulation environment setup where yellow rings denote the force fields. **Right:** One *fast-camera-slow-object* (FCSO) sample (**top row**) and *slow-camera-fast-object* (SCFO) sample (**bottom row**). Both are randomly selected from the DR-Lvl.3 dataset.

Dataset We rendered all scenes using a resolution of 64×64 for 40 frames (4-second motions)—record 40 images with their corresponding viewpoints $\{(x^t, v^t)\}_{1:40}$, where we represent the viewpoints using their 3-D Cartesion coordinates. The sampler specifications, i.e. the categorical distributions $\mathcal{C}at(...))$, used to generate the five DRoom subsets are listed in Table 4. As discussed in Sec.3.3, we clustered all the data samples based on their average camera speeds across each sequence to assign them into the FCSO and SCFO partitions. We visualize the clustering results for DR-Lvl.1 $\sim$ 3 in Figure 2

Table 4: DRoom Generator Specs

Subsets		Force Magnitude (constant in its range)	Camera Random Walk Next Move (for both azi and ele)
DR0-$\lvert\overline{f_x}\rvert$	—	$\{1, 0, 0, ..., 0\}$	$\{0, 0, 0, 0, 0, 0, 0.01, 0.11, 0.28, 0.3, 0.3\}$
DR0-$\lvert\overline{f_v}\rvert$	—	$\{0, 0, 0, 0, 0, 0.02, 0.08, 0.15, 0.35, 0.35, 0.05\}$	$\{1, 0, 0, ..., 0\}$
DR-Lvl.1	FCSO	$\{0.05, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095\}$	$\{0.05, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095\}$
	SCFO	$\{0.05, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095\}$	$\{0.05, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095, 0.095\}$
DR-Lvl.2	FCSO	$\{0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0\}$	$\{0, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2\}$
	SCFO	$\{0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2\}$	$\{0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0\}$
DR-Lvl.3	FCSO	$\{0.25, 0.38, 0.33, 0.02, 0.02, 0, 0, 0, 0, 0\}$	$\{0, 0, 0, 0, 0, 0, 0.01, 0.11, 0.28, 0.3, 0.3\}$
	SCFO	$\{0, 0, 0, 0, 0, 0.02, 0.08, 0.15, 0.35, 0.35, 0.05\}$	$\{0.3, 0.3, 0.28, 0.11, 0.01, 0, 0, 0, 0, 0\}$

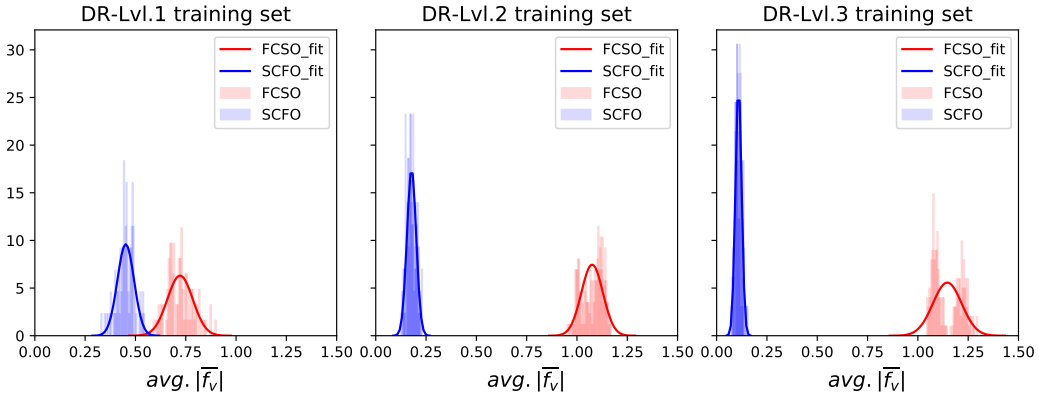

Figure 2: Visualization of the data assignment results on the DR-Lvl.1 $\sim$ 3 datasets.

C.2 MJC-Arm (Mujoco-Arm)

Simulation Environment The environment is built with MuJoCo physics simulator [5], and the Franka Emika robot arm with a Barret hand attached it the main scene object. The arm has 7 degrees of freedom and the joints of robotic hand are fixed during the data generation. 8 different collision-free

robot arm motion trajectories are pre-defined, and each has unique initial and target joint configuration. Every joint is controlled in the position-derivative manner with a constant velocity, which is the product of the nominal velocity and the sampled weight. The nominal velocities for all 7 arm joints (from base to end-effector) are $[0.65, 0.65, 0.27, 0.27, 0.03, 0.03, 0.005]$, which are related to the link lengths of the robot arm. The joint velocity weights for FCSO and SCFO data trials are sampled from $\mathbf{random.choice}_{FCSO}(\{0, 0.1, 0.2, ..., 1\}, \text{probs} = \{0.34, 0.34, 0.25, 0.07, 0.0, ..., 0.0\})$ and $\mathbf{random.choice}_{SCFO}(\{0, 0.1, 0.2, ..., 1\}, \text{probs} = \{0.0, ..., 0.0, 0.07, 0.25, 0.34, 0.34\})$. We also introduced a moving ball with random fixed direction and constant weighted velocity in the simulation. The control of the RGB camera is the same as introduced in the former section, with a fixed point of view towards the base link of the robot arm.

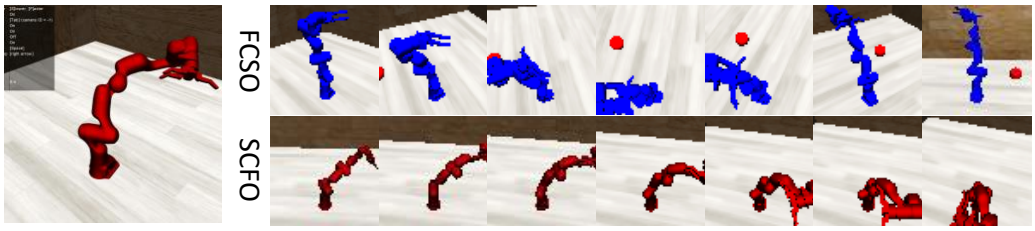

Figure 3: **Left:** Mujoco simulation environment. **Right:** One *fast-camera-slow-object* (FCSO) sample (**top row**) and *slow-camera-fast-object* (SCFO) sample. Both are randomly selected from the MJC-Arm dataset.

Dataset For each data sample, the scenes are rendered with resolution 64×64 at 10Hz for 4 seconds (40 frames per sample). At the beginning of every trail, the textures of the robot arm and the moving ball are randomly selected from a colour set. The robot arm is initialised with the starting pose of the randomly selected motion trajectory.

C.3 Real-world aataset (CubeLand)

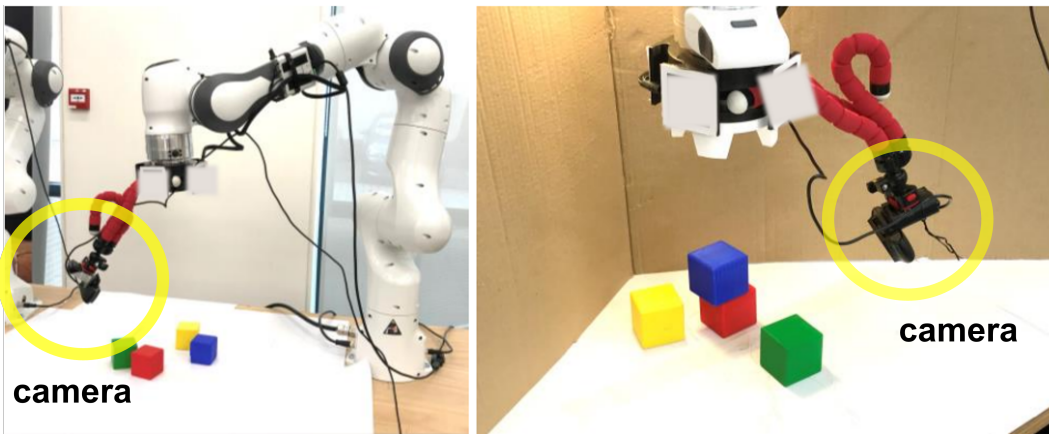

Figure 4: CubeLand data-collection platform.

Data-collection Environment We created CubeLand in a controlled real-world environment. Four cubes of different colours (i.e., red, blue, green and yellow) were placed on a table. To avoid glare, reflections and unnecessary background clutter, the surface of the table was made white by designing a bicolour data collection environment. A camera was mounted on the end effector of Franka (a robotic arm with 7 D.O.F.) as shown in Figure 4. The end effector has a fixed motion, i.e., it only rotates back and forth 120 degrees with no translation motion involved. The cubes were taped with threads at the bottom to move them freely and randomly. Moreover, the simulations had two configurations, i.e., slow camera, fast objects (SCFO) and fast camera, slow objects (FCSO) (see Figure 5). In the first configuration, the speed of the rotation of the end effector was 1.67 rpm (10 degrees per second) while the objects were manually pulled and thrown back into the scene at an arbitrary faster speed. In the latter configuration, the speed of the rotation of the end effector was set to be 4.17 rpm (25 degrees per second) whereas the objects were pulled and pushed by hand back into

the scene at a slower rate. The height of the camera is 14.5 cm and the radius this assembly (centre of the end effector to the camera) spans is 19.5 cm.

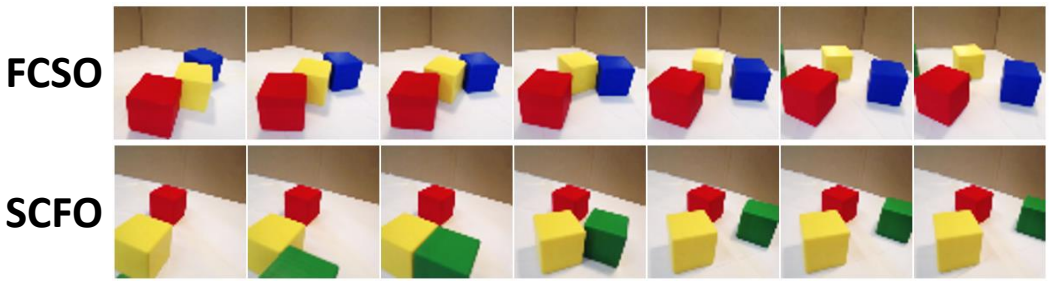

Figure 5: CubeLand data samples. **Top:** a fast camera, slow objects (FCSO) data sample. **Bottom:** a slow camera, fast objects (SCFO) data sample.

Dataset All the frames collected were initially 480x480. During the post-processing steps, these frames were resized to 64x64 after applying a median filter (the centre of the kernel is replaced by the median of all the neighbouring pixels) of kernel size 9. Overall, 100 sequences of 50 frames each were extracted. Furthermore, each of the viewpoints was converted into 3D cartesian coordinates. The classification between SCFO and FCSO is solely based on the rotations per minute of the end effector.

D. Additional Results

D.1 Assumption Validation

As discussed in Sec.3.1. of the main paper, the training of DyMON on *multi-view-dynamic-scene* is based upon two assumptions that favor high frame-rate image sequences and large difference between the speeds of an observer and scene objects. In this experiments, as we know that the average speed differences of DR-Lvl.1 $\sim$ 3 are in an ascending order, we can thus assess the robustness of DyMON against our assumptions. We trained DyMON on the DR-Lvl.1 $\sim$ 3 training sets respectively and then evaluated their performance on space-time-queried prediction of scene appearances on the DR-Lvl.1 $\sim$ 3 test sets. We visualize the MSE as a function of increased levels of speed differences in Figure 6. As shown, 1) there is no significant performance drops across different training and

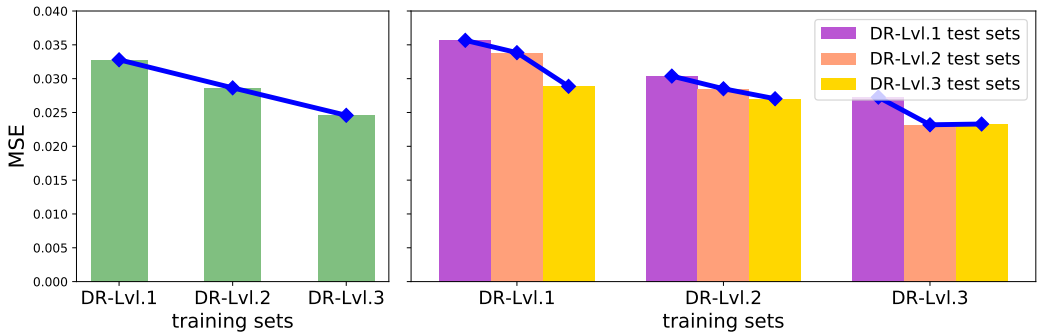

Figure 6: The space-time-queried scene appearance prediction performance comparison between three DyMONs that are trained on three levels of scene-observer speed differences, i.e. DR-Lvl.1 $\sim$ 3, respectively. **Left:** Averaged MSE achieved by the three models on three DRoom testing sets, i.e. the testing sets of DR-Lvl.1 $\sim$ 3. **Right:** The performance of the three models on each of the three testing sets.

test sets, 2) the faster the observer speeds and the scene speeds, the better the models perform. This holds for both training (see the overall performance on the left Figure) and testing (see a model's testing performance against different test sets on the right Figure). These supports our claims about DyMON's robustness against complex and potentially dynamic environments.

D.2 Ablation Study

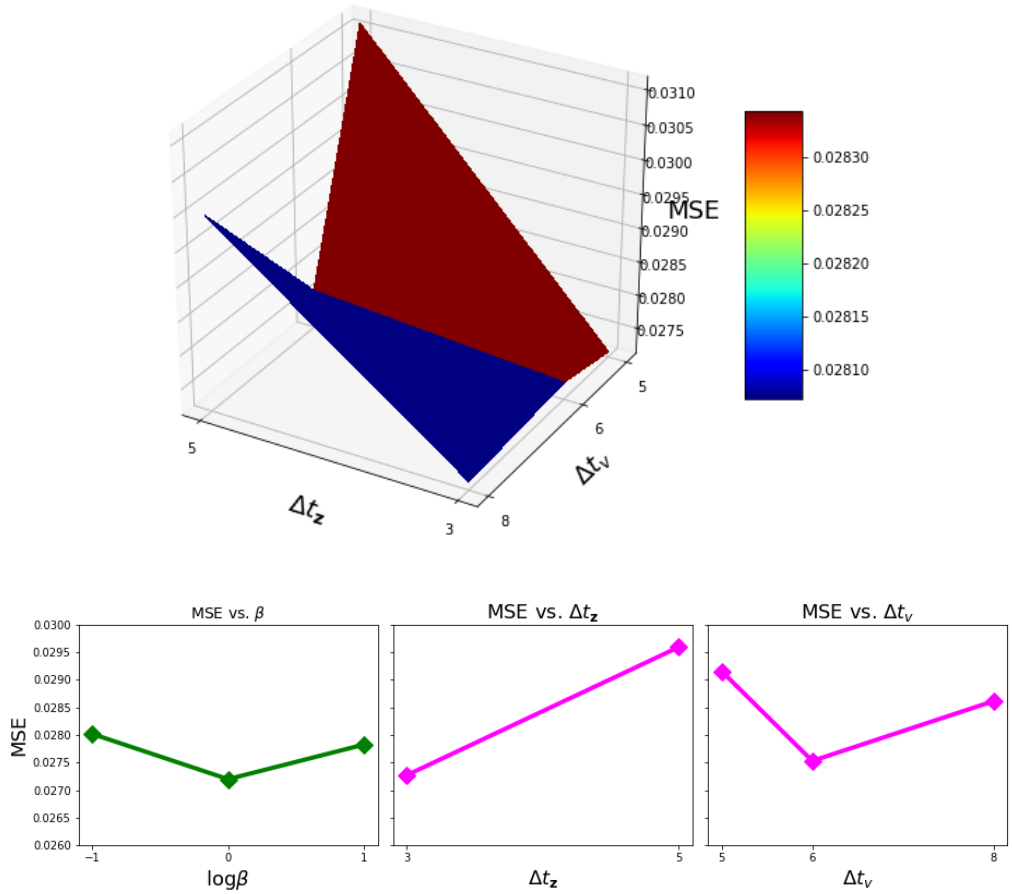

Figure 7: Ablation study results. **Top:** Space-time queried view synthesis MSE vs. nested $\Delta t_{\mathbf{z}}$ and Δt_v. **Bottom left:** MSE vs. different β (in $\log_2$ space). **Bottom middle:** MSE vs. different $\Delta t_{\mathbf{z}}$ (MSE computed by averaging across different Δt_v). **Bottom right:** MSE vs. different Δt_v (MSE computed by averaging across different $\Delta t_{\mathbf{z}}$).

We highlight two hyperparameters that play significant roles in the training of DyMON: 1) the updating periods of v and $\mathbf{z}$, i.e. Δt_v and $\Delta t_{\mathbf{z}}$, 2) weighting coefficient of viewpoint-queried generative log likelihood β. We varied these two groups of parameters and visualized their influences on DyMON—similar to Sec.D.1, we measure DyMON's novel-view synthesis performance at every time point and visualize them as a function of these hyperparameters. We varied $\Delta t_{\mathbf{z}}$ and Δt_v with values that are selected from discrete sets $\{3, 5\}$ and $\{5, 6, 8\}$, this allows us to show the joint effects of these two updating periods in a 2×3 grid (see **top** half of Figure 7). To analyze the independent effects of $\Delta t_{\mathbf{z}}$ and Δt_v, we "squeezed" the 2×3 grid by computing the MSE averaged over the $\Delta t_{\mathbf{z}}$ axes and Δt_v axes of the grid (see bottom right two plots of Figure 7 for the results). One can see that a short updating period for $\Delta t_{\mathbf{z}}$ is preferred as this allows to capture more detailed scene object motions, while the selection of Δt_v is relative subtler. One might run pre-analysis before training, e.g. visually look several sequences, to select a better Δt_v. Similarly, we varied β by setting its values to 0.5, 1.0, and 2.0 respectively and we show the results in the bottom left of Figure 7.

D.3 T-GQN Results

We used the official implementation of T-GQN (`https://github.com/singhgautam/snp`) and trained a T-GQN on the DR-Lvl.3 data. Although the training has converged (see Figure 8), we

observe that it fails to represent the underlying 3D scenes (see Figure 9) and training T-GQN with a posterior dropout, i.e. T-GQN-PD, does not fix the issue. We speculate that this is because it lacks multiple views at each time steps to resolve the temporal entanglement issue. However, future investigations are required to validate our speculation.

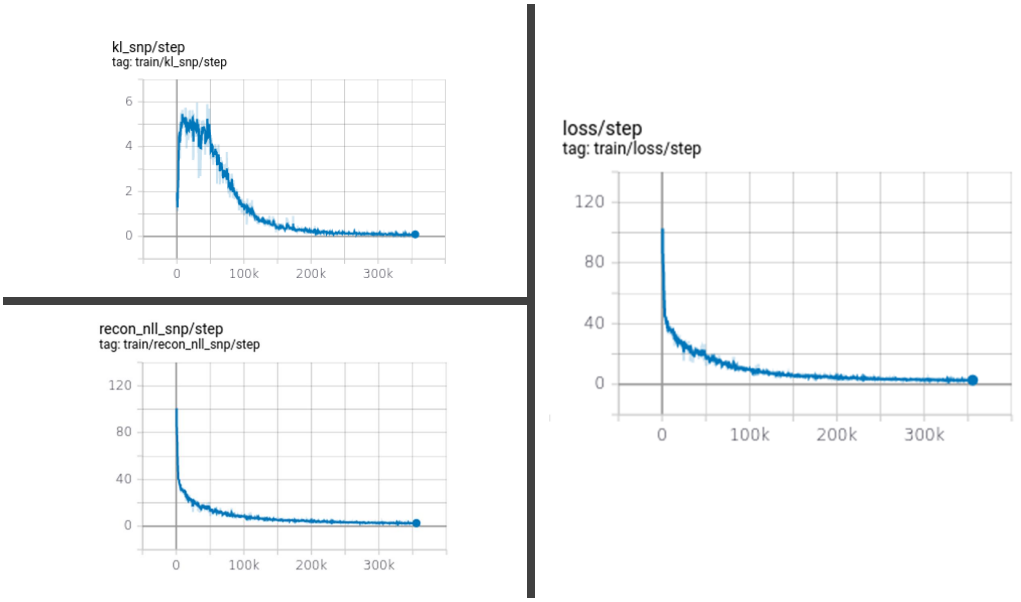

Figure 8: T-GQN training curves. We train t-GQN on our DRoom data until it converges.

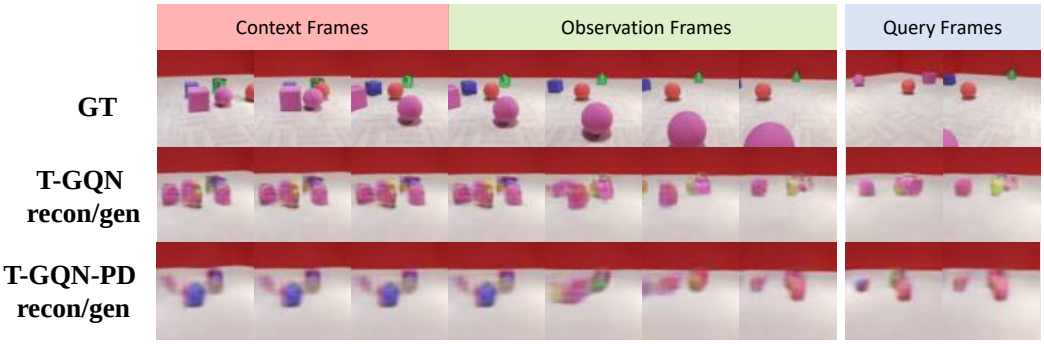

Figure 9: Qualitative results of T-GQN on DR-Lvl.3 test data.

D.4 Additional Qualitative Results

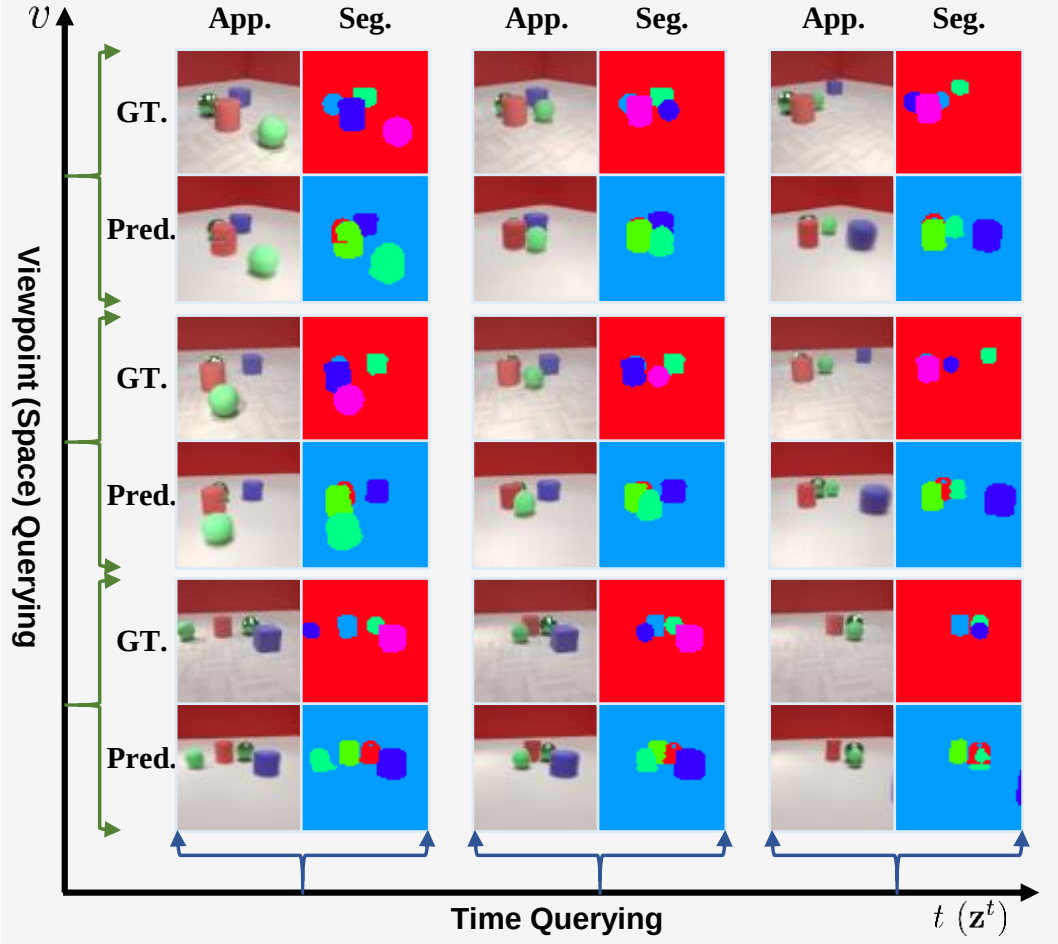

Figure 10: Spatial-temporal factorization results of a DRoom scene.

References

[1] CLEVR Blender Environment, BSD licencse. `https://github.com/facebookresearch/clevr-dataset-gen`. Accessed: 2021-06-02.

[2] DeepMind MultiObject Dataset, Apache-2.0 licencse. `https://github.com/deepmind/multi_object_datasets`. Accessed: 2021-06-02.

[3] K. Greff, R. L. Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. Multi-object representation learning with iterative variational inference. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2424–2433, 2019.

[4] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[5] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[6] N. Watters, L. Matthey, C. P. Burgess, and A. Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019.

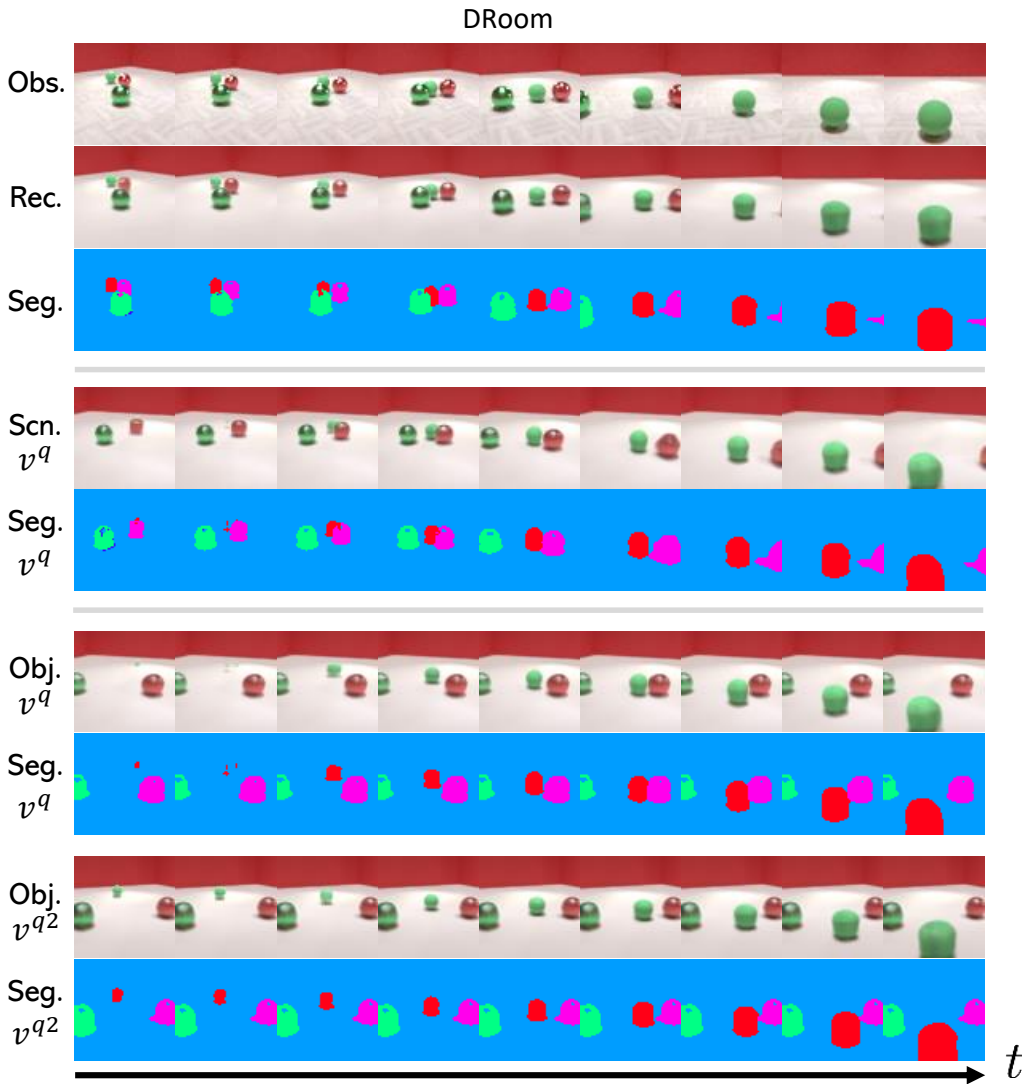

Figure 11: Dynamics replay of a DRoom scene.

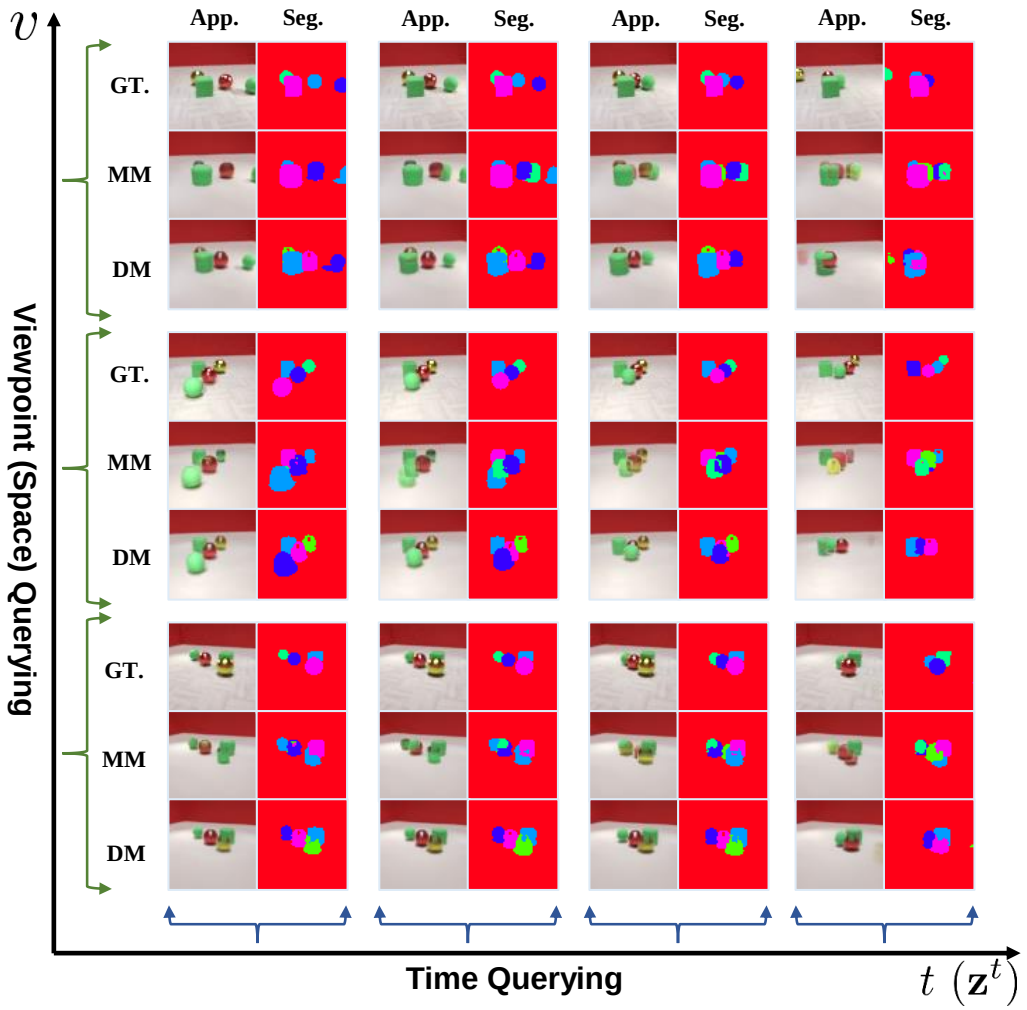

Figure 12: Qualitative comparisons of DyMON and MulMON on DRoom. **Left:** reconstruction performance. **Right:** spatial-temporal factorization performance. We train DyMON on DR-Lvl.3 and train MulMON on DR0-$|\overline{f_{\mathbf{z}}}|$.

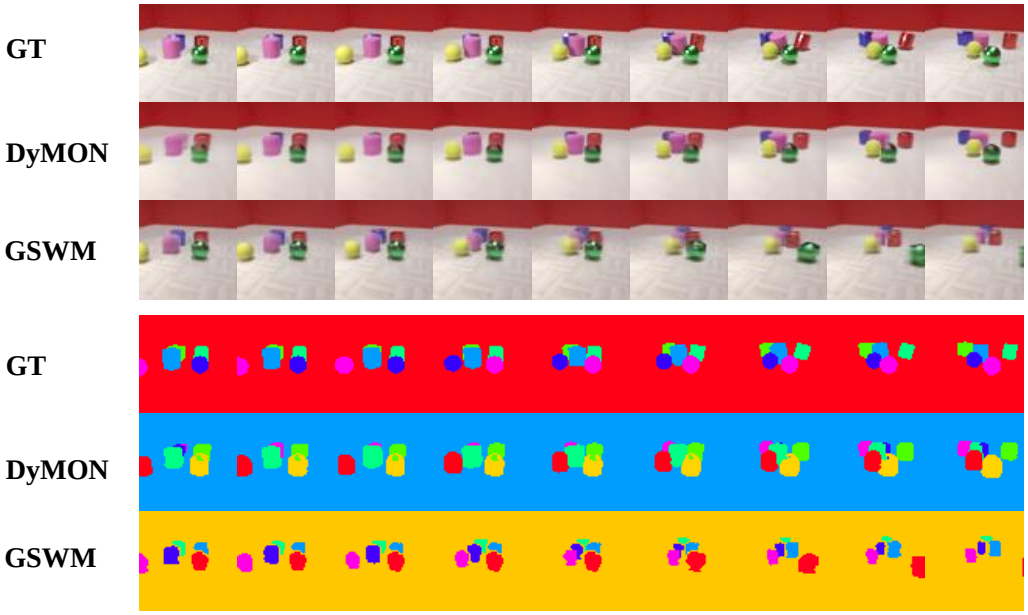

Figure 13: Qualitative comparisons of DyMON and GSWM on DR0-$|\overline{f_v}|$. **Top:** reconstruction performance. **Bottom:** segmentation performance (we observe that DyMON outperforms GSWM in segmenting scenes).

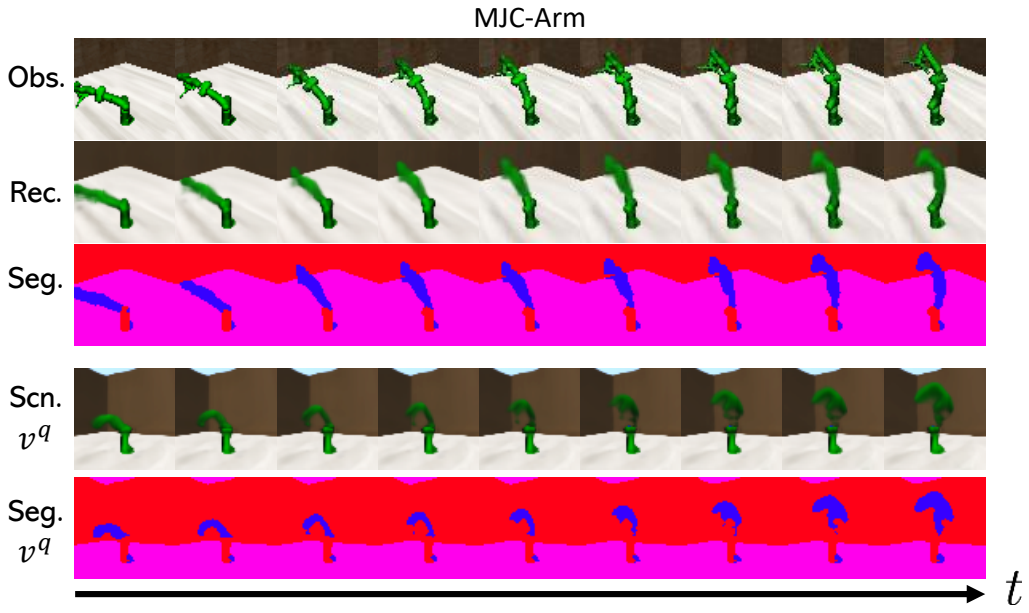

Figure 14: Dynamics replay of a MJC-Arm scene.

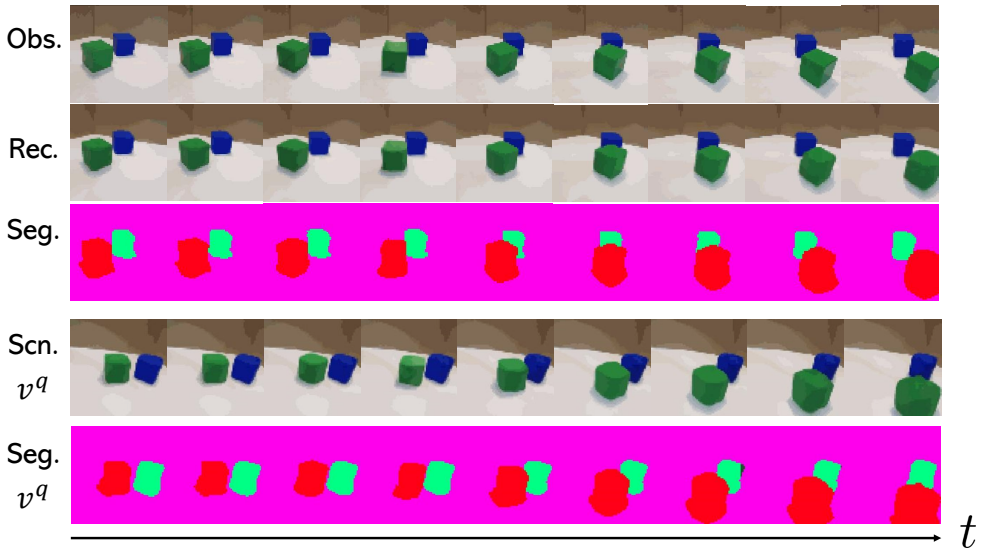

Figure 15: Dynamics replay of a real scene (i.e. CubeLand data). We conduct experiments on real-world data to show DyMON's potential for real-world applications.