PROVIDE: A Probabilistic Framework for Unsupervised Video Decomposition (Supplementary Material)

Polina Zablotskaia[1,2,3*] **Edoardo A. Dominici**[2] **Leonid Sigal**[1,2,3,4] **Andreas M. Lehrmann**[1]

[1]Borealis AI, Vancouver, Canada
[2]University of British Columbia, Vancouver, Canada
[3]Vector Institute for Artificial Intelligence, Toronto, Canada
[4]CIFAR AI Chair

A DATASETS

Bouncing Balls. Bouncing Balls is a dataset provided by the authors of R-NEM [Van Steenkiste et al., 2018]. The dataset contains balls with different masses corresponding to their radii. The balls are initialized with random initial positions, masses and velocities. Balls bounce elastically against each other(without occlusions) and the image window. We use the train and test splits of this dataset in two different versions: binary and color. For the color version, we randomly choose 4 colors for the 4-balls (sub-)dataset. For the 6-8 balls test data, we color them in 2 different ways: 4 colors (same as train) and 8 colors (4 from train, 4 new ones). Note that the former results in identical colors for multiple objects, while the latter guarantees unique colors for each object.

CLEVRER. Each video in the CLEVRER dataset contains at least one collision and (dis)appearence event making occlusions possible and frequent. Objects' initial velocities are approximately $\pm 2.5\ m/s^2$. Each object has one of eight distinct colors and one of 38 two materials (metal or rubber). In addition, two objects can have the same color but different material.

The version of the CLEVRER dataset [Yi et al., 2020] used in this work was processed as follows:

- Train split, validation split and validation annotations were obtained from the official website: `http://clevrer.csail.mit.edu/`. We use the validation set as test set, because the test set does not contain annotations.
- For training, we use the original train split. Our minimal preprocessing consists of cropping the frames along the width axis by 40 pixels on both sides, followed by a uniform downscaling to 64x64 pixels. Since the length of each video is 128 frames and the maximum number of frames during training was 40, we split the videos into multiple sequences to obtain a larger number of training samples.
- For testing, we trim the videos to a subsequence containing at least 3 objects and object motion. We compute these subsequences by running the script (slice_videos_from_annotations.py in the attached code) from the folder with the validation split and validation annotations.
- The test set ground truth masks can be downloaded from here. The masks and the preprocessed test videos will be grouped into separate folders based on the number of objects in a video.

B HYPERPARAMETERS

Initialization. We initialize the parameters of the posterior $\boldsymbol{\lambda}$ by sampling from $\mathcal{U}(-0.5, 0.5)$. In all experiments, we use a latent dimensionality $\dim(\boldsymbol{z}) = 64$, such that $\dim(\boldsymbol{\lambda}) = 128$. Horizontal and vertical hidden states and cell states are of size 128, initialized with zeros. $q_{\boldsymbol{\lambda}}$ is the posterior probability per slot of the likelihood $p(\mathbf{x}|\mathbf{z})$, which is a Gaussian mixture model. The variance of the likelihood is set to $\sigma = 0.3$ in all experiments.

Experiments on Bouncing Balls. For this experiment, we have explored several values of R (refinement steps) and empirically found $R = 6$ to be optimal in terms of accuracy and efficiency. Refining the posterior more than 6 times does not lead to any substantial improvement, however, the time and memory consumption is significantly increased. For the 4-balls dataset, we use $K = 5$ slots for train and test. For our tests on 6-8 balls, we use $K = 9$ slots. This protocol is identical to the one used in R-NEM [Van Steenkiste et al., 2018]. Furthermore, we set $\beta = 100.0$ and scale the KL term by $\psi = 10$. The weight of the entropy term is set to $\gamma = 0.1$ in the binary case. As expected, the effect of the entropy term is most pronounced with binary data, so we set $\gamma = 0$ in all experiments with RGB data.

Experiments on CLEVRER. We keep the default number of iterative refinements at $R = 5$, because we did not ob-

*Now at Google Research, Berlin, Germany

serve any substantial improvements from a further increase. We use $K = 6$ slots during training, $K = 6$ slot when testing on 3-5 objects and $K = 7$ slots when testing on 6 objects.

C BASELINES

C.1 R-NEM

We use the R-NEM [Van Steenkiste et al., 2018] authors' original implementation and their publicly available models: `https://github.com/sjoerdvansteenkiste/Relational-NEM`.

C.2 IODINE

Our IODINE experiments are based on the following PyTorch implementation: `https://github.com/MichaelKevinKelly/IODINE`. We use the same parameters as in this code, with the exceptions of $\beta = 10$ (weight factor) and, for the Bouncing Balls experiments, $R = 6$ (refinement steps). The majority of the hyperparameters shared between PROVIDE and IODINE are identical.

C.3 SEQ-IODINE

In order to test the sequential version of IODINE, we use the regularly trained IODINE model but change the number of refinement steps to the number of video frames during testing. During each refinement step, instead of computing the error between the reconstructed image and the ground truth image, we use the next video frame. Since the IODINE model was trained on $R = 6$ refinement steps, extending the number of refinement steps to the video length leads to exploding gradients. This effect is especially problematic in the binary Bouncing Balls dataset with 20, 30 and 40 frames per video, because the scores of the static model are already low. We deal with this issue by clamping with $\max = 10$ and $\min = -10$ the gradients and the δ refinement value in this experiment[1]. SEQ-IODINE's weak performance, especially w.r.t. the ARI, reflect the gradual divergence from the optimum as the number of frames increases.

D TRAINING

We use ADAM [Kingma and Ba, 2014] for all experiments, with a learning rate of 0.0003 and default values for all remaining parameters. During training, we gradually increase the number of frames per video, as we have found this to make the optimisation more stable. We start with sequences of length 4 and train the model until we observe a stagnant loss or posterior collapse. At the beginning of training, the batch size is 32 and is gradually decreased negatively proportional to the number of frames in the video.

D.1 INFRASTRUCTURE AND RUNTIME

We train our models on 8 GeForce GTX 1080 Ti GPUs, which takes approximately one day per model.

E DISCUSSION AND FUTURE WORK

Introduction of a temporal component not only enables modelling of dynamics inside the amortized iterative inference framework but also improves the quality of the results overall. From our quantitative and qualitative comparisons with IODINE and SEQ-IODINE, we see that our model shows more accurate results on the decomposition task. We can detect new objects faster and are less sensitive to color, because our model can leverage the objects' motion cues. The ability to work with complex colored data, a property inherited from IODINE, means that we significantly outperform R-NEM. However, R-NEM is a stronger model when it comes to prediction of longer sequences, owing to its ability to model the relations between the objects in the scene. Similar ideas were used in SQAIR [Kosiorek et al., 2018] and GENESIS [Engelcke et al., 2020] by adding a relational RNN [Santoro et al., 2017]. Integration of these concepts into our framework is a promising direction for future research. Another possible route is an application of PROVIDE to complex real-world scenarios. However, given that such datasets typically contain a much higher number of objects, as well as intricate interactions and spatially varying materials, we consider the resulting scalability questions as a separate line of research.

F ADDITIONAL EXPERIMENTS

F.1 ABLATIONS

The performance of PROVIDE is governed by the function $\widehat{R} = \max(R - t, 1)$, where R is the free parameter. In Table 1 we explore values of R ranging from 2 to 10. We see performance saturation at $R \approx 4$. We also explore an alternative choice $\widehat{R}_{\text{alt.}}$ (Table 2), which shows decreased performance compared to $\widehat{R}$. The number of slots K could be determined via cross-validation, but for comparability to other SOTA methods we assume it to be given.

F.2 ADDITIONAL QUALITATIVE RESULTS

Figure 2 and Figure 3 show PROVIDE applied to both versions of the Bouncing Balls dataset.

[1] Please note that clamping was done only when applied to binary Bouncing Balls for 20, 30 and 40 frames.

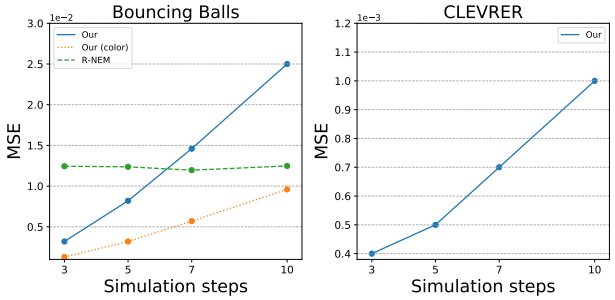


Figure 1: Mean Squared Error for the prediction experiment. We have computed MSE for the same experimental setup as Fig. 5 in the main paper. As expected the MSE increases with number of simulation steps. Similarly to ARI and F-ARI scores our model outperforms R-NEM on a first steps of simulation, however the error function of our model grows faster comparatively to R-NEM and we sooner diverge from the accurate simulation.

Table 1: **Performance as a Function of Parameter R.**

	ARI (↑)				F-ARI (↑)				MSE (↓)($\times 10^{-4}$)			
R	2	4	8	10	2	4	8	10	2	4	8	10
BB bin.	0.34	0.71	0.73	0.73	0.93	0.99	0.99	0.99	424	6	5	8
BB col.	0.48	0.72	0.73	0.73	0.93	0.99	1.0	1.0	148	3	3	4
CLEVRER	0.21	0.24	0.23	0.22	0.84	0.92	0.93	0.94	11	3	3	3

Table 2: **Ablation on the Form of the Function $\widehat{R}$.** $\widehat{R}_{\text{alt.}} = R$, when $t = 0$, and $\widehat{R}_{\text{alt.}} = 1$, when $t > 0$.

	ARI (↑)		F-ARI (↑)		MSE (↓)($\times 10^{-4}$)	
	$\widehat{R}$	$\widehat{R}_{alt}$	$\widehat{R}$	$\widehat{R}_{alt}$	$\widehat{R}$	$\widehat{R}_{alt}$
BB bin.	**0.73**	0.43	**1.0**	0.95	**4**	33.2
BB col.	**0.73**	0.57	**1.0**	0.97	**2**	11.9
CLEVRER	**0.24**	0.21	**0.93**	0.88	**3**	9

F.3 PREDICTION

Qualitative results for the prediction experiment are shown in Figure 4 for Bouncing Balls and Figure 5 for CLEVRER.

F.4 GRAND CENTRAL STATION DATASET

PROVIDE shows consistency in separating human figures in the video across the whole sequence, despite increased number of objects compare to the training set.

F.5 DISENTANGLEMENT

In Figure 8 we demonstrate that introducing a new temporal hidden state and an additional MLP in front of the spatial broadcast decoder has not impacted its ability to separate each object's representations and disentangles them based on color, position, size and other features, similar to results shown in Greff et al. [2019].

F.6 ANIMATIONS

Attached animations include the following files:

- **bb_binary_4_balls.gif** Animation of the segmentations of 4 binary Bouncing Balls. 50 frames. Here and everywhere else, unless explicitly specified, we also included full scene decomposition and each object's individual reconstruction.
- **bb_binary_6_8_balls.gif** Animation of the ability to generalize to 6-8 binary Bouncing Balls. 40 frames.
- **bb_colored_4_balls.gif** Animation of the 4 colored Bouncing Balls. 50 frames.
- **bb_colored_6_8_balls.gif** Animation of the ability to generalize to 6-8 colored Bouncing Balls. 40 frames.
- **bb_colored_predict.gif** Prediction on the Bouncing Balls colored data. With 40 normal steps of inference and 10 predicted masks and frames. Here we only included predicted masks and ground truth masks.
- **clevrer_5obj.gif** Animation of the segmentations of 5 objects CLEVRER dataset. 50 frames.
- **clevrer_6obj.gif** Animation of the ability to generalize to 6 objects CLEVRER dataset. 45 frames.

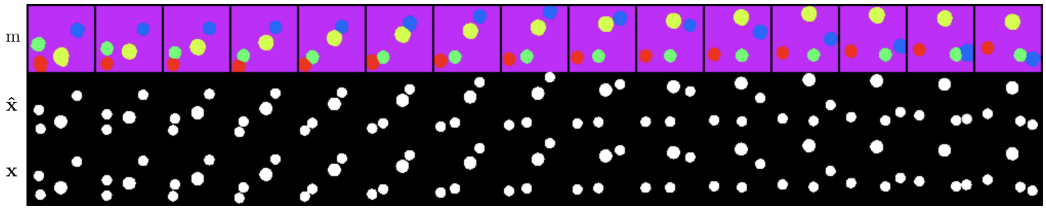


Figure 2: Video decomposition using PROVIDE applied on Bouncing Balls dataset with 4 balls.

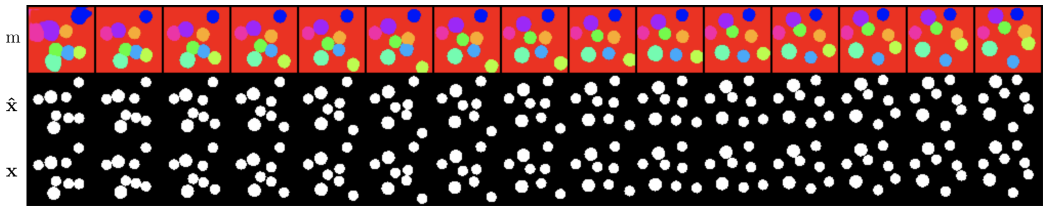


Figure 3: Video decomposition using PROVIDE applied on Bouncing Balls dataset with 6-8 balls.

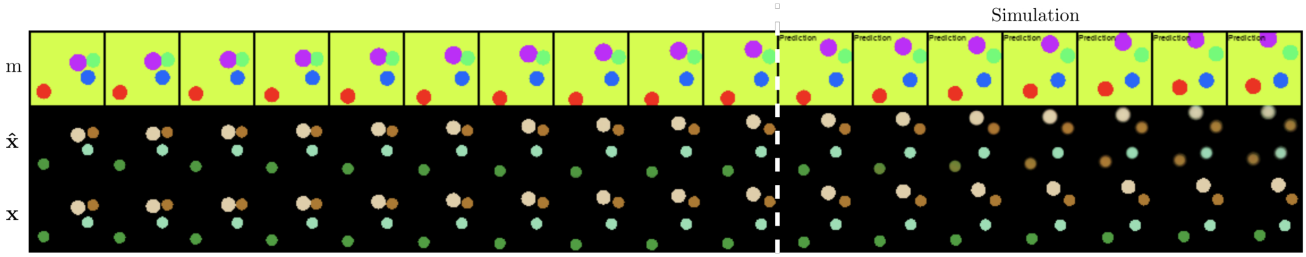


Figure 4: Prediction on Bouncing Balls (colored) dataset.

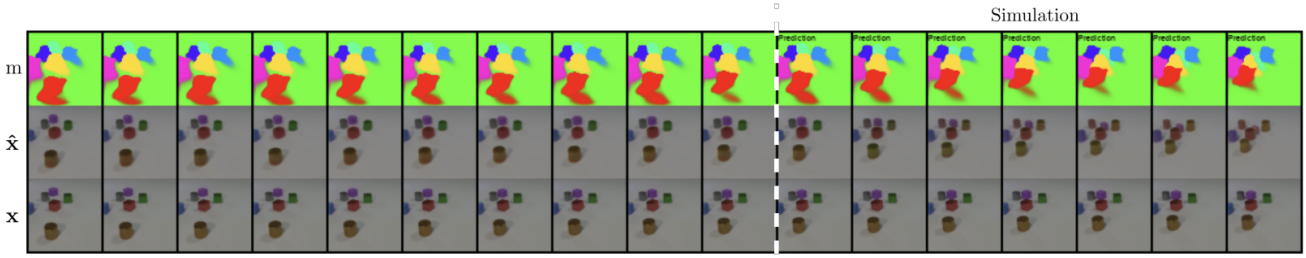
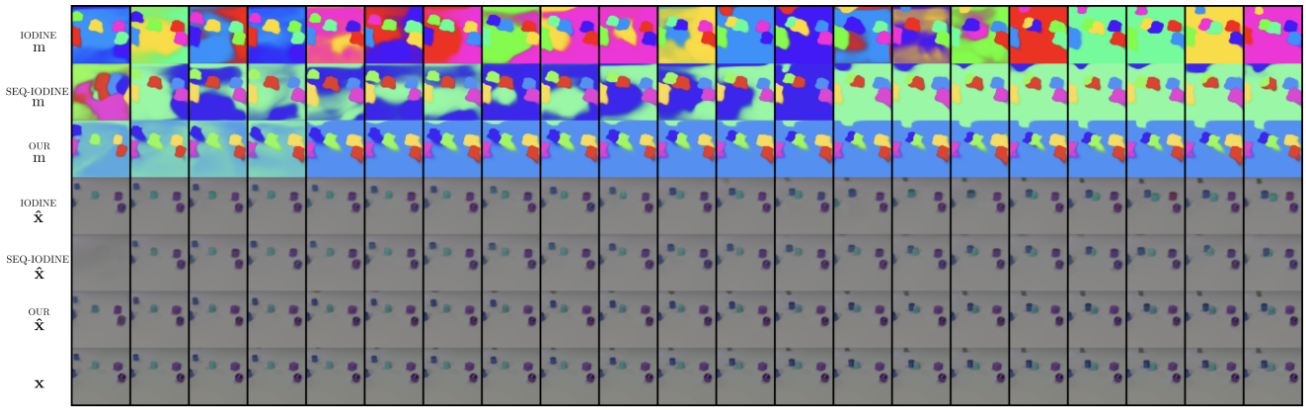
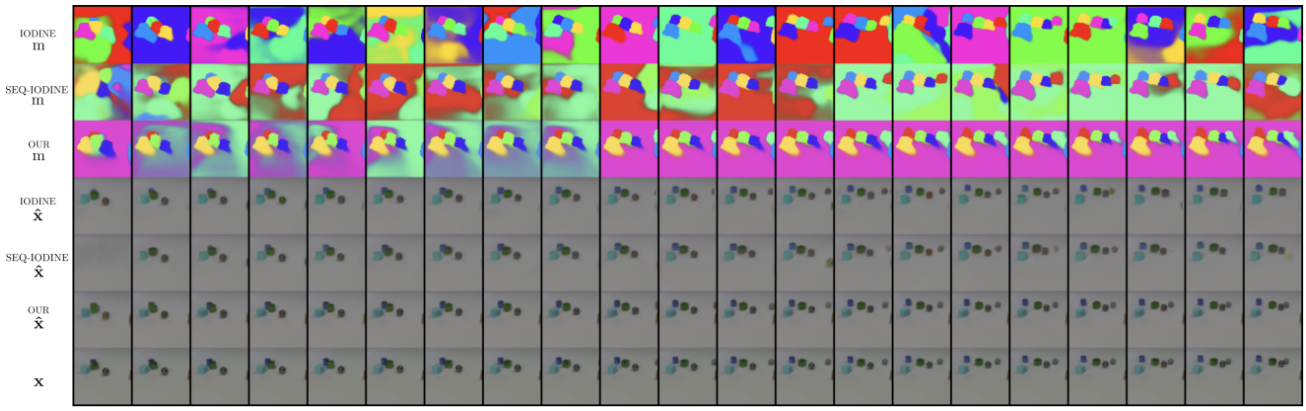


Figure 5: Prediction on CLEVRER dataset.



(a)



(b)

Figure 6: Qualitative results for **PROVIDE vs. IODINE vs. SEQ-IODINE** decomposition experiment. (a) From the figure it is clear that our model can much sooner detect new objects emerging to the frame, while SEQ-IODINE struggles to properly reconstruct and decompose them. And IODINE doesn't have any temporal consistence and reshuffles the slot order. (b). Here we can see that our model is much more stable with time and it does not fail to detect objects, unlike IODINE and SEQ-IODINE.

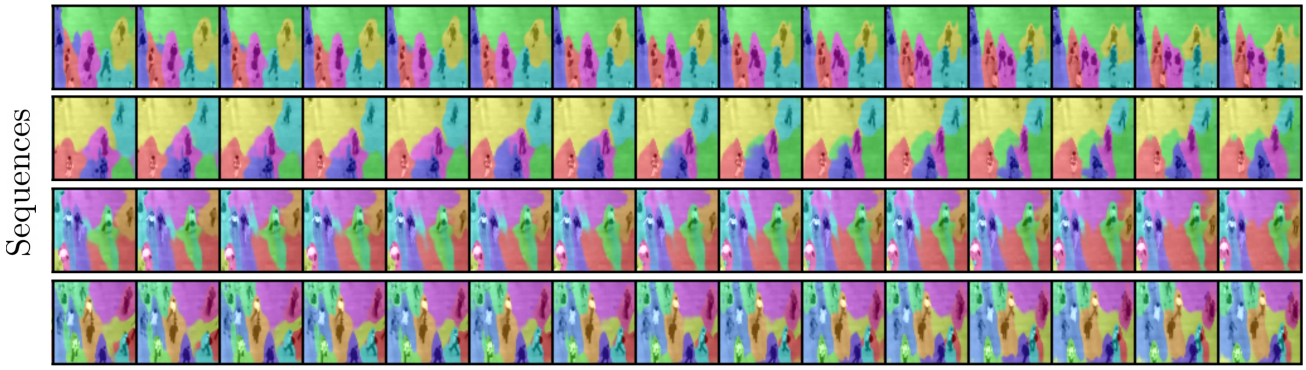


Figure 7: **Qualitative evaluation on real-world data.** Qualitative Evaluation (Grand Central Station). We can observe that PROVIDE is very consistent in separating the image regions belonging to different objects as they move in the scene. This dataset is particularly challenging for its background texture, complex lighting and shadows. Please zoom in to allow better clarity.

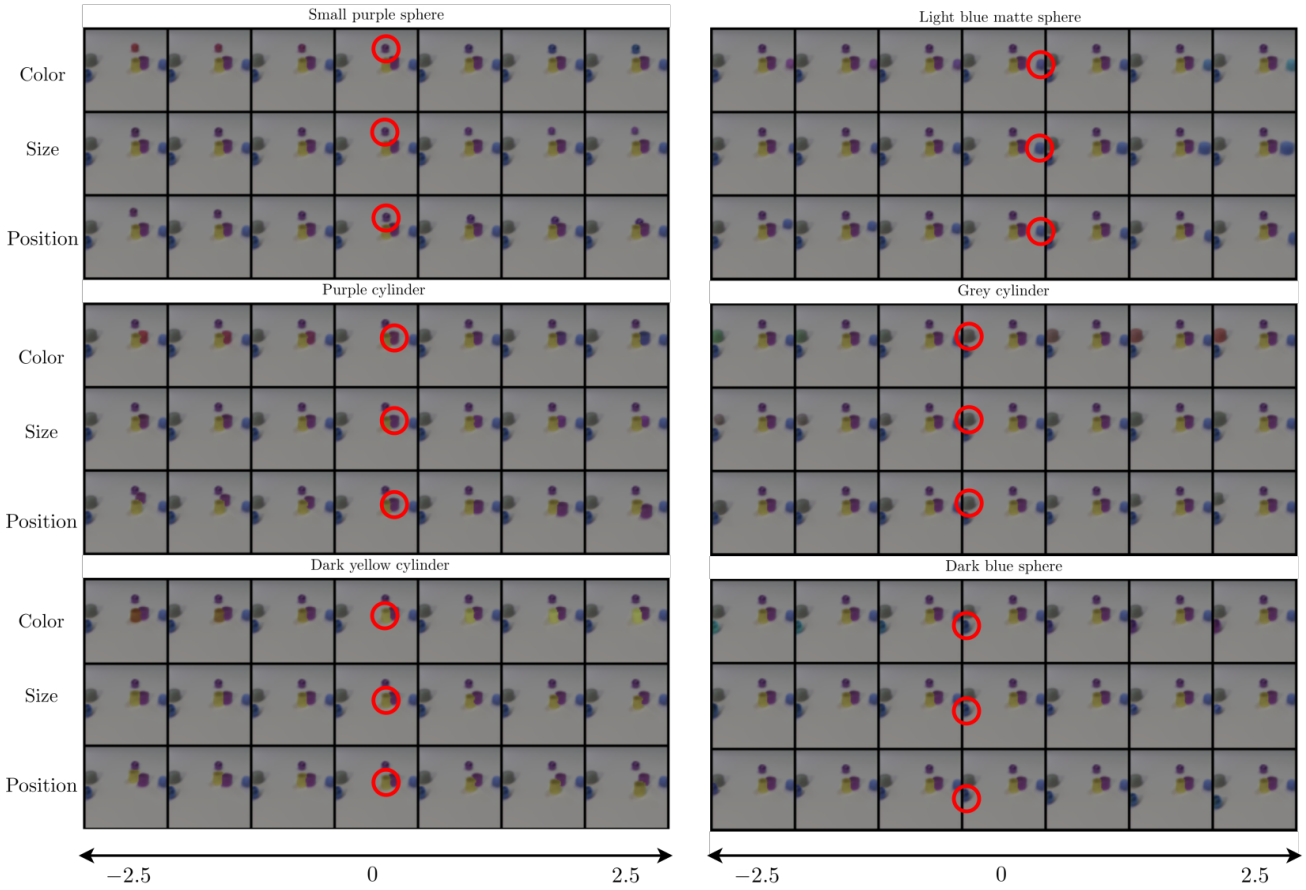


Figure 8: Disentanglement of the latent representations corresponding to distinct interpretable features. CLEVRER latent walks along three different dimensions: color, size and position. We chose a random frame and for each object's representation in the scene dimensions were traversed independently.