Supplementary Materials: Video Extrapolation in Space and Time

Yunzhi Zhang and Jiajun Wu

Stanford University
{yzzhang,jiajunwu}@cs.stanford.edu

A Architecture Details

The architecture used for the MPI encoder is specified in Table 1.

Input	**k**	**c**	**Output**	**Input**	**k**	**c**	**Output**
Concat(I_{t-1}, I_t)	7	32	down1	down1	7	32	down1b
MP2(down1b)	5	64	down2	down2	5	64	down2b
MP2(down2b)	3	128	down3	down3	3	128	down3b
MP2(down3b)	3	256	down4	down4	3	256	down4b
MP2(down4b)	3	512	down5	down5	3	512	down5b
MP2(down5b)	3	512	down6	down6	3	512	down6b
MP2(down6b)	3	512	mid1	mid1	3	512	mid2
Up2(mid2) + down6b	3	512	up6	up6	3	512	up6b
Up2(up6b) + down5b	3	512	up5	up5	3	512	up5b
Up2(up5b) + down4b	3	256	up4	up4	3	256	up4b
Up2(up4b) + down3b	3	128	up3	up3	3	128	up3b
Up2(up3b) + down2b	3	64	up2	up2	3	64	up2b
Up2(up2b) + down1b	3	64	post1	post1	3	64	post2
post2	3	64	up1	up1	3	64	up1b
up1b	3	64 x D	conv1	Reshape(conv1)	3	64	conv2
conv2	7	7	conv3	ReshapeBack(conv3)	-	-	output

Table 1: MP2 is max pooling with stride 2, Up2 is nearest-neighbor upsampling with scale 2, + is concatenation. Reshape transforms a tensor with $C \times D$ channels into C channels, and D is merged to the batch dimension, and ReshapeBack is the reverse operation. All layers up till up1b use ReLU activation and the layers for conv1, conv2 and conv3 use LeakyReLu with a negative slope 0.2. There is no activation following the very last layer. All layers use Instance Norm for activation normalization and Spectral Norm for weight normalization.

B Implementation details

To have a better gradient flow, similar to Tucker et al. [4], we add a harmonious bias $1/i$ to the alpha channel prediction, so that w_i from Equation (12) becomes

D	Extrapolation in Space			Extrapolation in Time		
	LPIPS↓	PSNR ↑	SSIM ↑	LPIPS↓	PSNR↑	SSIM↑
4	0.0987	19.3453	0.7180	0.0792	22.9415	0.7880
8	0.0874	20.5795	0.7881	0.0784	23.1073	0.7922
16	0.0786	21.1889	0.8188	0.0757	23.3812	0.7971
32	0.0762	21.2279	0.8207	0.0726	23.7882	0.8083

Table 2: Ablation on the number of MPI planes D. Increasing the plane count improves the performance but also increases the training time. We adopt $D = 16$ in the main paper since further increasing D results in diminishing returns.

uniformly $1/D$ during initialization. We also add an identity bias to f^θ such that each MPI plane is associated with zero motion during initialization.

In all experiments, we set the number of MPI planes to be $D = 16$. The depth values for MPI planes are linear in the inverse space, with $d_1 = 1000$ and $d_D = 1$.

C Training details

C.1 KITTI

Since videos from KITTI are taken by stereo cameras with fixed relative poses, the depth scale is consistent across scenes and therefore we set it to be a constant $\sigma = 1$. We use $\lambda_1^{\text{space}} = 1000$, $\lambda_{\text{spec}}^{\text{space}} = 100$, $\lambda_1^{\text{time}} = 1000$, and $\lambda_{\text{perc}}^{\text{time}} = 10$. We use Adam Optimizer [3] with an initial learning rate 0.0002, which we exponentially decrease by a factor of 0.8 for every 5 epochs. We train our model for 200K iterations on two NVIDIA TITAN RTX GPUs for about two days. During training, we apply horizontal flip with 50% probability and apply color jittering as data augmentation.

C.2 RealEstate10K

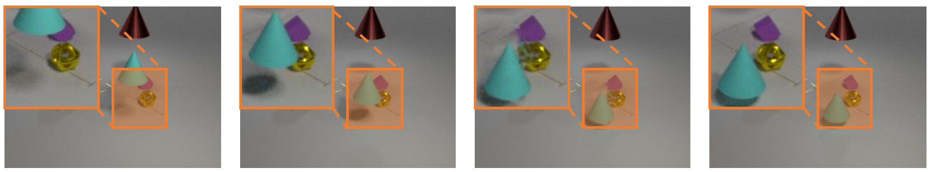
We train our model for 200K iterations on one NVIDIA GeForce RTX 3090 GPU, which takes about one day. We use $\mathcal{L}_1^{\text{space}} = 10, \mathcal{L}_{\text{perc}}^{\text{space}} = 10, \mathcal{L}_1^{\text{time}} = 10, \mathcal{L}_{\text{perc}}^{\text{time}} = 0$. We use Adam Optimizer [3] with a constant learning rate 0.0002.

C.3 Ablations on the number of MPI planes

To study the effect of the number of MPI planes, we perform an ablation study on the KITTI [1] dataset with resolution 128×384. As shown in Table 2, a small number of MPI planes ($D = 4$ or 8) results in degraded model performance. Further increasing the number of planes from 16 to 32 results in marginal performance gain, with a cost of 2.1$\times$ slower training time. Therefore, we use $D = 16$ for all other experiments.

Method	LPIPS↓	PSNR↑	SSIM↑
PredRNN [5]	0.0600	37.02	0.9643
Ours	**0.0122**	**42.58**	**0.9762**

Table 3: Results of next-frame prediction on CATER [2]. Our model achieves better performance compared to PredRNN [5].



(a) Input $t-1$ (a) Input t (c) Prediction $t+1$ (d) GT $t+1$

Fig. 1: Model prediction on an example scene with occlusion. (a) and (b) are two historical frames as model inputs, (c) and (d) are the predicted and ground truth next frame, respectively. Top-left corners of subfigures are zoomed-in views for occluded regions.

C.4 Modeling dynamic scenes

To test whether our method is able to model more dynamic scenes, we test our method on CATER [2], a dataset of scenes with 5-10 individually moving objects. We show a quantitative comparison with a video prediction baseline PredRNN [5]. As shown in Table 3, our model achieves better performance across all three metrics.

Qualitatively, our method makes temporal prediction consistent with the ground truth object motions on this dataset. In Fig. 1, the model correctly recovers the purple object and the gold object occluded by the blue cone. Our model effectively handles object occlusions by warping from neighboring pixels with similar RGB values.

C.5 Discussions

While we focus on demonstrating the possibility of simultaneous extrapolation in both space and time, specific modules can be further optimized for each task. For example, it is possible to improve the dynamic scene representation to better handle video prediction with long horizons or highly complex motion, or to synthesize novel views with a large viewpoint change.

In the meantime, while our method is designed for natural scenes with many potential positive impacts such as interactive scene exploration for family entertainment, like all other visual content generation methods, our method might be exploited by malicious users with potential negative impacts. We expect such impacts to be minimal as our method is not designed to work with human videos. In our code release, we will explicitly specify allowable uses of our system with appropriate licenses. We will use techniques such as watermarking to label visual content generated by our system.

References

1. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
2. Girdhar, R., Ramanan, D.: CATER: A diagnostic dataset for compositional actions and temporal reasoning. In: ICLR (2020)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
4. Tucker, R., Snavely, N.: Single-view view synthesis with multiplane images. In: CVPR. pp. 551–560 (2020)
5. Wang, Y., Wu, H., Zhang, J., Gao, Z., Wang, J., Yu, P., Long, M.: Predrnn: A recurrent neural network for spatiotemporal predictive learning. IEEE TPAMI (2022)