DS-NeRV: Implicit Neural Video Representations with Decomposed Static and Dynamic Codes

Supplementary Material

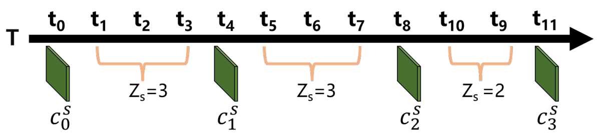
T
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_{10} t_9 t_{11}
Z_s=3
Z_s=3
Z_s=2
c_0^s
c_1^s
c_2^s
c_3^s

Figure 1. Static Codes Sampling.

A. Video Modeling Details

Static Codes. In our implementation, for simplicity, we determine the static codes length l_s by directly specifying it, bypassing the formula that contains the sampling rate r_s. Typically the value of l_s is the sum of a factor of the video length T, denoted as T_f, and 1 (*i.e.* $l_s = T_f + 1$), where the additional 1 refers to the last static code placed on the final frame of the video. This ensures that a fixed interval z_s between most static codes (*i.e.* $z_s = T/l_s$), excluding the last two.

Specifically, as shown in Fig. 1, given $T = 12$, we choose the T_f as 3, thus the length of the static codes l_s is computed as $l_s = T_f + 1 = 4$. The intervals of the first three static codes is set to $T/l_s = 3$, while the interval of the last two static codes is 2.

Dynamic Codes. Similar to the static codes, we also directly specify the value of l_d to determine the length of the dynamic codes, thus bypassing the calculation containing the sampling rate r_d. Since we do not employ the sampling method of weighted sum used in static code sampling but use interpolation when sampling dynamic code corresponding to frame index t, so the dynamic codes length l_d has greater flexibility and can be set freely. Typically, we set the l_d to be approximately half of the video length T. This approach not only avoids the storage overhead of saving a dynamic code for each frame but also ensures sufficient dynamic information for subsequent interpolation operations.

B. Additional Setup

B.1. Datasets

We conduct experiments using 7 videos from UVG: Beauty, Bosphorus, HoneyBee, Jockey, ReadySetGo, ShakeNDry, and YachtRide. Except for ShakeNDry, which consists of 300 frames, the remaining videos contain 600 frames each. The resolution of all the videos is 960×1920. We also select 10 videos in DAVIS to conduct experiments. The chosen videos include Blackswan, Bmx-trees, Boat, Breakdance, Camel, Car-roundabout, Car-shadow, Cows, Dance, and Dog. These videos have a relatively small number of frames, presenting significant challenges for our experiments.

B.2. Implementation.

For NeRV and HNeRV, we conduct experiments using their open-source implementations. As for DNeRV, we develop our implementation based on the open-source E-NeRV code. When comparing the model sizes between DS-NeRV and other implicit methods, the total size of HNeRV comprises the sum of its embedding and decoder, whereas in the case of DNeRV, the total size is calculated by summing the diff embedding, content embedding, and the decoder. As for DS-NeRV, the total size includes the sum of the static and dynamic codes as well as the fusion decoder.

In our typical implementation, for a $960 \times 1920 \times 3$ video frame, we configure the dimensions of each static code as $4 \times 8 \times 64$. The dimensions of each dynamic code is set to $20 \times 40 \times 2$. The lengths of the static and dynamic codes, while depending on the extent of changes in video dynamics, are significantly shorter than the original video length. In a few videos with strong dynamic changes, such as Ready and Jockey, we fine-tune the dynamic codes dimensions to accommodate high dynamics.

We provide more architecture details for our video reconstruction approach on Bunny and UVG in Tab. 1. $l_s \times h_s \times w_s \times dim_s$ and $l_d \times h_d \times w_d \times dim_d$ represent the dimensions of the static and dynamic codes, respectively. c_1 is the number of channels of the fused code. Ch_{min} is the lowest channel width in the NeRV blocks. We adopt the settings from [1] to set the stride list, kernel size and the channel reduction rate in NeRV blocks. To match the spatial dimensions of the static codes with those of the dynamic codes, the first NeRV block performs upsampling with a upscale factor of 5. The $Conv_q$, $Conv_k$, $Conv_v$ in CCA are all set to 2D convolution with a step size of 1, kernel size of 1, and with the number of input and output channels both set to c_1.

C. Additional Ablation Results

C.1. Fusion Mechanism

We explore different fusion mechanisms for integrating static and dynamic codes on UVG, which can be categorized into three main approaches: **a)** Summation. **b)** Spatial attention. **c)** Channel attention. As indicated in Tab. 2, a simple summation of static and dynamic codes leads to

Video	size	resolution	$l_s \times h_s \times w_s \times dim_s$	$l_d \times h_d \times w_d \times dim_d$	c_1	Ch_{min}	strides
Bunny	0.35	640×1280	$13{\times}4 \times 8 \times 64$	$66{\times}20 \times 40 \times 1$	36	16	(5,2,2,2,2,2)
Bunny	0.75	640×1280	$13{\times}4 \times 8 \times 64$	$66{\times}20 \times 40 \times 1$	48	28	(5,4,2,2,2)
Bunny	1.5	640×1280	$13{\times}4 \times 8 \times 64$	$66{\times}20 \times 40 \times 2$	70	38	(5,4,2,2,2)
Bunny	3	640×1280	$13{\times}4 \times 8 \times 64$	$66{\times}20 \times 40 \times 4$	92	70	(5,4,2,2,2)
Beauty	3	960×1920	$61{\times}4 \times 8 \times 64$	$300{\times} \times 20 \times 40 \times 2$	80	56	(5,4,3,2,2)
Bosph	3	960×1920	$61{\times}4 \times 8 \times 64$	$300{\times}20 \times 40 \times 2$	80	56	(5,4,3,2,2)
Honey	3	960×1920	$61{\times}4 \times 8 \times 64$	$300{\times}20 \times 40 \times 2$	80	56	(5,4,3,2,2)
Yacht	3	960×1920	$61{\times}4 \times 8 \times 64$	$300{\times}20 \times 40 \times 2$	80	56	(5,4,3,2,2)
Ready	3	960×1920	$31{\times}4 \times 8 \times 64$	$300{\times}20 \times 40 \times 4$	76	44	(5,4,3,2,2)
Jockey	3	960×1920	$31{\times}4 \times 8 \times 64$	$400{\times}20 \times 40 \times 4$	70	38	(5,4,3,2,2)
Shake	3	960×1920	$101{\times}4 \times 8 \times 64$	$150{\times}20 \times 40 \times 2$	82	58	(5,4,3,2,2)

Table 1. Architecture details of DS-NeRV on various tasks.

	Beauty	Bosph	Honey	Jockey	Ready	Shake	Yacht
Sum	33.89	34.95	39.39	32.77	26.97	34.88	29.29
S-A	33.80	34.75	39.45	31.42	25.69	35.00	28.85
Ours	**33.97**	**35.22**	**39.56**	**32.86**	**27.10**	**35.04**	**29.4**

Table 2. Ablation study for fusion mechanisms.

poor performance, and performing cross-spatial attention even perform worse. Cross-channel attention helps identify the most relevant channels when fusing static code and dynamic code, thereby improving the performance.

C.2. Spatial Dimension of the Dynamic Code

We maintain a constant overall model size of 3M while testing the impact of varying the dimensions of dynamic codes on the results. The results, as shown in the Tab. 3, indicate that when the spatial dimension of dynamic codes is small, it becomes challenging to recover high-frequency motion information from low spatial resolution codes. Our experiments suggest that setting the dynamic codes spatial size $h_d \times w_d$ to 20×40 can achieve the best results.

Dynamic codes dimension	PSNR	MS-SSIM
$4 \times 8 \times 32$	37.96	0.9877
$4 \times 8 \times 64$	37.69	0.9869
$20 \times 40 \times 2$	38.55	0.9895
$20 \times 40 \times 4$	**38.65**	**0.9897**

Table 3. Ablation study for the dimension of dynamic codes.

D. Additional Quantitative Results

D.1. Video Reconstruction

	Beauty	Bosph	Honey	Jockey	Ready	Shake	Yacht
NeRV	32.79	31.98	37.91	30.04	23.48	32.89	26.26
DNeRV	31.62	30.18	33.53	29.62	22.68	32.45	25.75
HNeRV	31.37	31.37	38.2	31.35	24.54	33.29	27.64
Ours	**33.29**	**34.31**	**38.98**	**32.64**	**26.41**	**34.04**	**28.72**

Table 4. Video Reconstruction on UVG with 1080p

In prior works, such as HNeRV, the resolution is adjusted to 960×1920 to maintain a 1:2 aspect ratio, ensuring a small initial image embeddings (2×4 spatial size) to improve model performance. We also conduct experiments on the standard UVG with 1080p, as shown in the Tab. 4. Our method still achieves best performance, while others suffer performance degradation due to large initial image embeddings caused by inappropriate scaling.

D.2. Video Decoding

Methods	H.264	HEVC	NeRV	Ours
FPS	15	14	60.08	**63.54**

Table 5. Decoding FPS results

In real-world applications, inference time is a critical metric. A video is typically encoded once and requires decoding numerous times, akin to a movie that is encoded only once but viewed millions of times. Consequently, decoding time holds significance as a performance metric. Our DS-NeRV has the advantages of faster decoding speed and does not require to do frame decoding in a sequential manner like H.264 and HEVC, as shown in Tab. 5.

D.3. Video Inpainting

Method	PSNR(↑)	MS-SSIM(↑)	LPIPS(↓)	FPS(↑)
IIVI	**27.66**	**0.9574**	0.044	3.53
Ours	26.45	0.9515	**0.037**	**63.54**

Table 6. Additional video inpainting results.

We also conduct comparative experiments on video inpainting with SOTA inpainting method IIVI [2]. As shown in in Tab. 6, even without the specific design and complicated pipeline, we achieve competitive performance to contemporary SOTA inpainting methods, while achieving the fastest inference speed. IIVI requires 20 times the inpainting time we need.

E. Additional Qualitative Results

E.1. Visualization of video reconstruction.

We present an additional qualitative comparison of video reconstruction on UVG in Fig. 2. DS-NeRV preserves more high frequency details compared to other methods, such as the gloss on the lips in Beauty, the distant trees on the mountains in Bosphorus, the horseshoes in Jockey, and the buildings in Ready. Qualitative comparisons of video reconstruction in DAVIS are also shown in Fig. 3. DS-NeRV excels in reconstructing the feathers of the swan and the grassy bank in Blackswan. In Breakdance, DS-NeRV exhibits fewer artifacts in high dynamic areas, such as the dancer's shoes.

DS-NeRV also provides improved reconstruction of background foliage in Camel and offers higher-quality reconstruction of the audience and grass in Dance.

E.2. Visualization of video inpainting.

In video inpainting task, DS-NeRV is still capable of partially inferring reasonable content in the masked regions, achieving better results. Additionally, it outperforms other methods in preserving high frequency details in the rest of the image, as shown in the Fig. 4.

E.3. Visualization of video interpolation.

Qualitative results for video interpolation can be viewed in Fig. 5, where all the video frames shown can not seen during training. While HNeRV and DNeRV use the frames to be interpolated itself as input to obtain embeddings during testing, our method interpolates the static and dynamic codes, obtaining the static and dynamic information of the interpolated frames, and then proceeds with decoding. DS-NeRV still achieves competitive results.

F. Limitations and Future Work

While DS-NeRV has achieved excellent performance, specifying the length of static and dynamic codes for each video is still a manual process during training. This provides some flexibility but limits the model's ability for adaptive adjustments. Finding the optimal static and dynamic code dimensions for each video requires time for testing. In the future, we may explore more automated and adaptive training methods to assist the model in finding the optimal static and dynamic code decomposition solutions.

References

[1] Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. Hnerv: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10270–10279, 2023. 1

[2] Hao Ouyang, Tengfei Wang, and Qifeng Chen. Internal video inpainting by implicit long-range propagation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14579–14588, 2021. 2

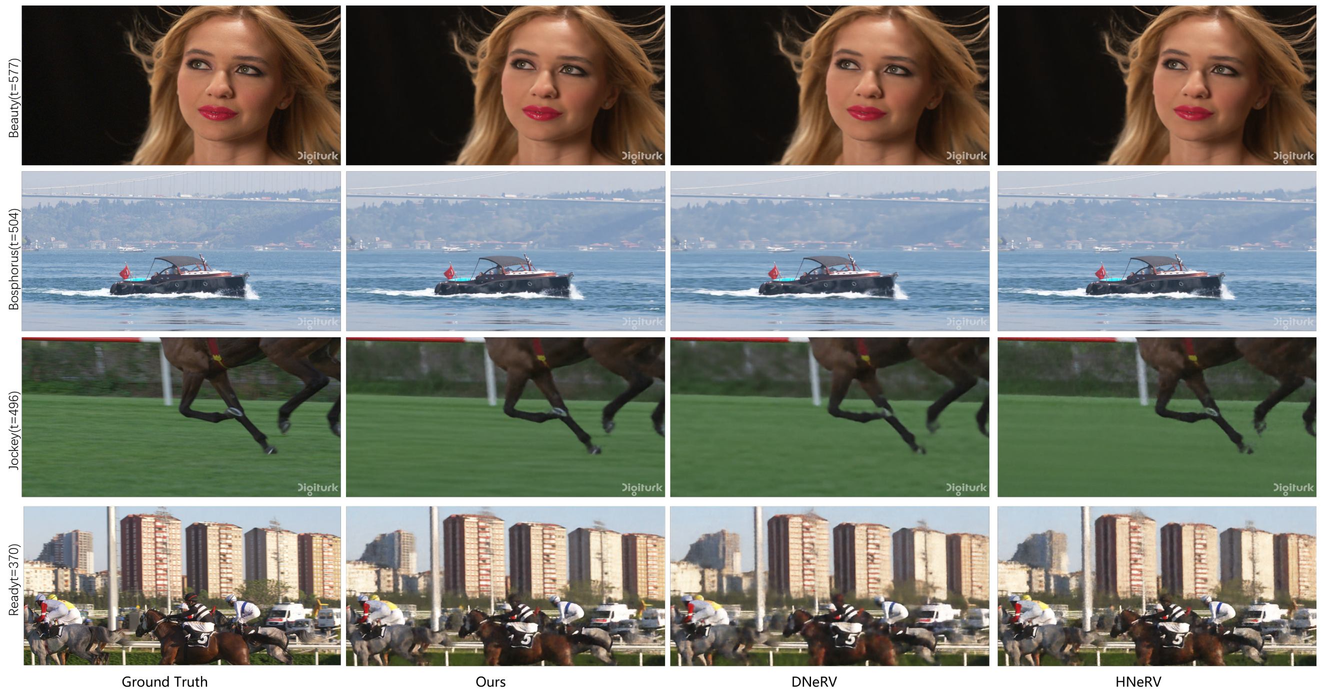

Figure 2. Additional **reconstruction** results on UVG.

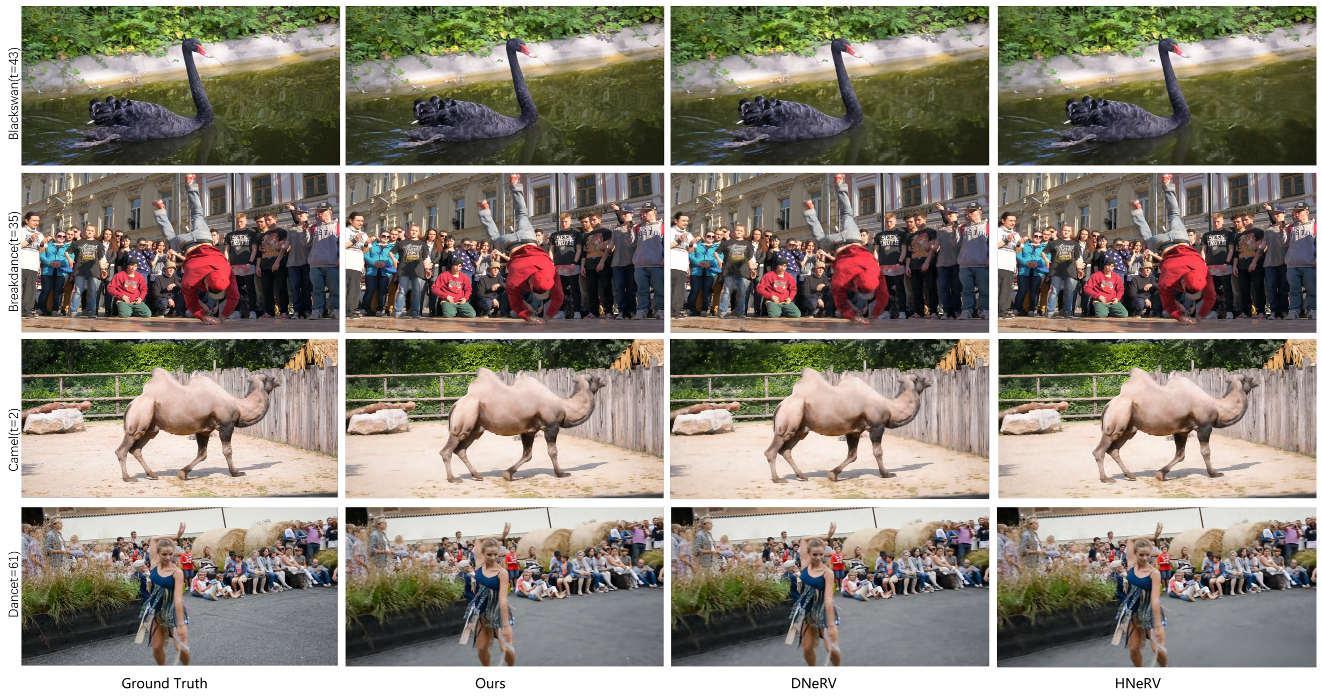

Figure 3. Additional **reconstruction** results on DAVIS.

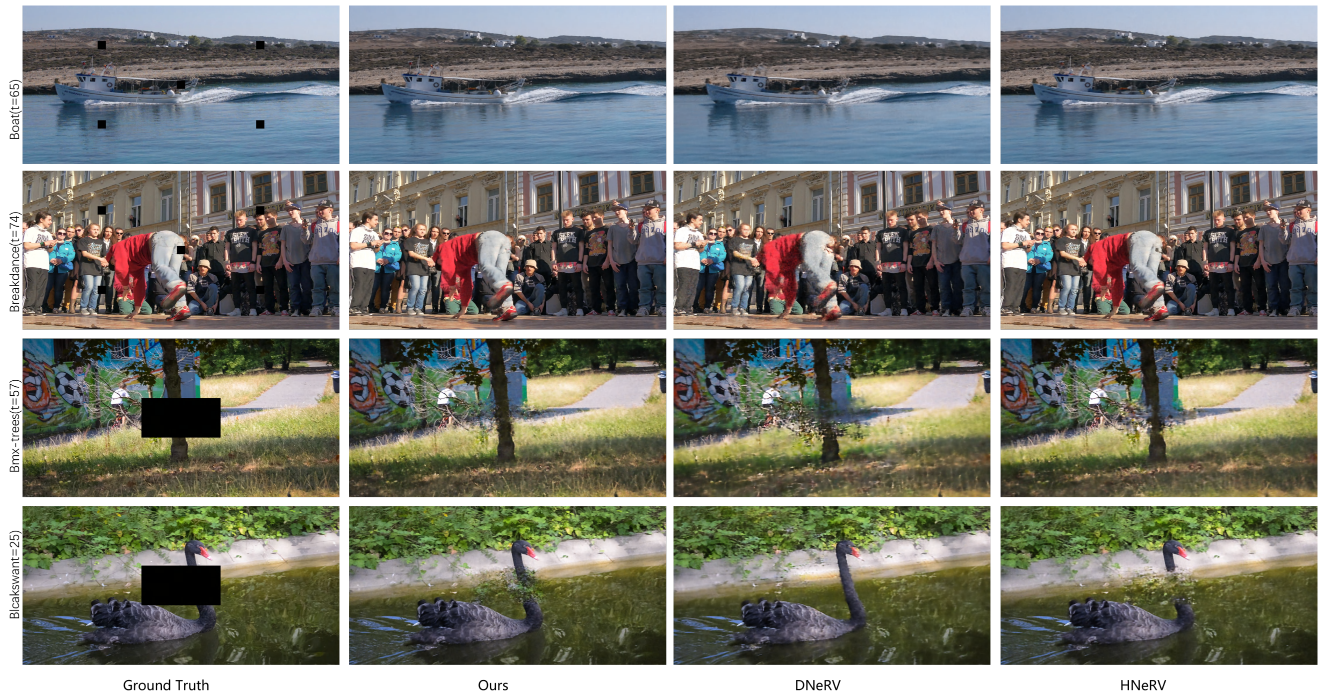

Figure 4. Additional **inpainting** results.

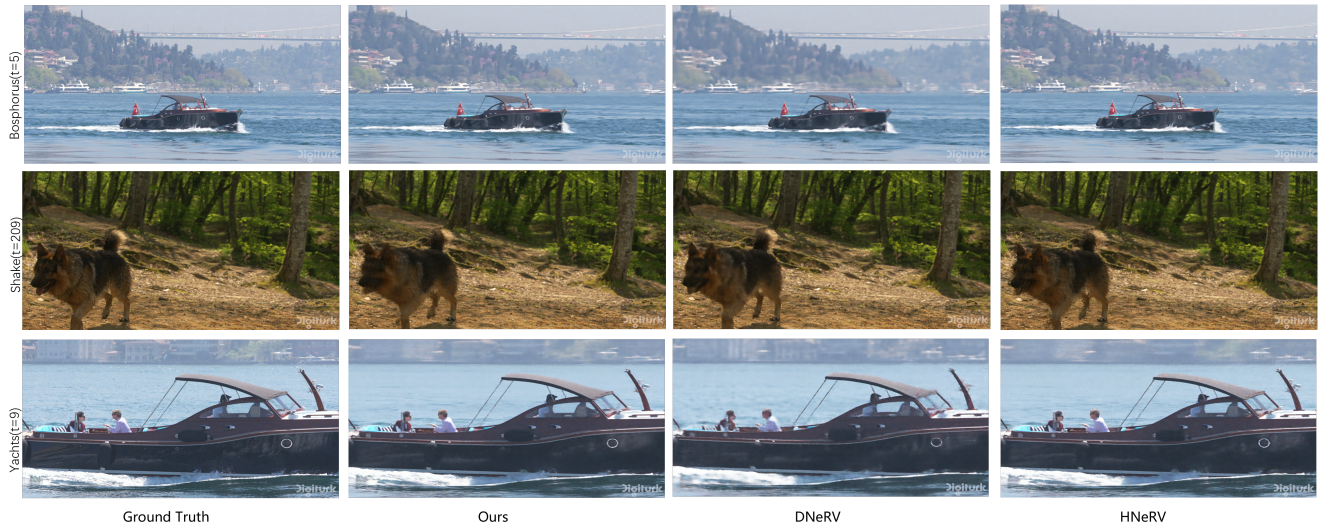

Figure 5. Additional **interpolation** results on UVG.