Generative Camera Dolly: Extreme Monocular Dynamic Novel View Synthesis

Supplementary Material

A Overview

The appendix is structured as follows: in Section B, we analyze what the equivalent number of source views given to HexPlane would have to be to match our method's performance, as well as our model's metrics as a function of the geometric "difficulty" of the camera controls. In Section C, we elaborate on implementation details in terms of the model architecture, how training is done, how datasets are processed, how evaluations are performed, and how the baselines are adapted. In Section D, we discuss failure cases. To view video visualizations of extra qualitative results, we recommend viewing gcd.cs.columbia.edu in a modern web browser.

B More quantitative evaluations

B.1 Comparison to multi-view methods

Our method is able to synthesize novel views of a dynamic scene from just a single-viewpoint input video. One other hand, the results from per-scene optimization methods (e.g., HexPlane [3]) get better with an increasing number of input views. A natural question is that how many input views are needed for those methods in order to obtain similar performance as compared to ours from a single view. We try to answer this question by training HexPlane per scene

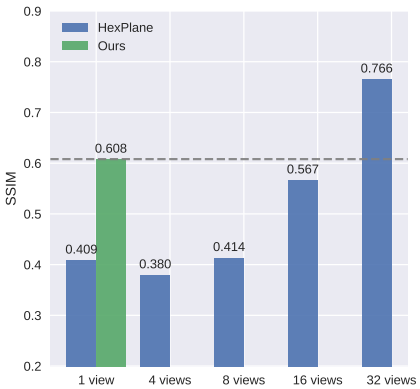

Fig. 1: Comparative study over number of views. We plot the SSIM over the test set as a function of the number of input views that HexPlane uses for training. The numbers are averaged over 20 scenes.

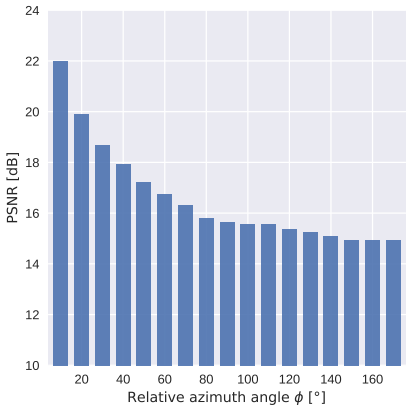

Fig. 2: Comparative study over camera rotation magnitude in Kubric-4D. Note that PSNR is measured at the last output frame, because only then the desired horizontal azimuth angle has been reached. We conclude that the main difficulty in performing dynamic view synthesis comes from handling roughly the first 80 degrees, after which the performance stays mostly flat.

with K training views (i.e., K input videos), with $K \in \{1, 4, 8, 16, 32\}$. As shown in Figure 1, our results (from a single input view) give rise to even better quality than HexPlane's results from 16 input views.

B.2 Error as a function of rotation angle

In Figure 2, we plot the average PSNR over the test set as a function of how significantly the final destination (target) camera pose differs from the source (input) camera pose. Specifically, we evaluate the Kubric-4D (*gradual, max 180°, finetuned*) model on a sequence of horizontal rotations to the right of varying amounts between 0° and 180°. The elevation angle θ is held constant at 10°, to encourage obstructed objects from the input view, and the radius r at 15m.

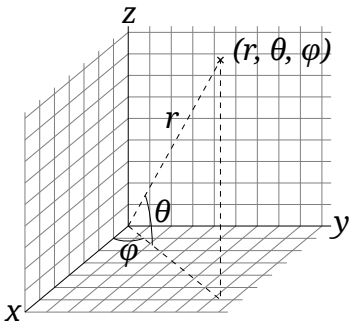

Fig. 3: Spherical coordinate system. Models trained on Kubric-4D accept an azimuth ϕ, elevation θ, and radius r as input to condition the video generation process. (Illustration adapted from [11].)

C Implementation details

C.1 Training

We adopt the SVD variant that predicts $T = 14$ frames, but due to computational constraints, we downscale the input and output resolution to $W \times H = 384 \times 256$. This allows us to scale the batch size up to 56 when training with Kubric-4D on 7x A100 GPUs with 80 GB VRAM each. We finetune all models for 10k iterations using the Adam optimizer, which takes roughly 3 days. On ParallelDomain-4D, we instead finetune models for 13k iterations with an effective batch size of 24 through a gradient accumulation factor of 4 on 3x A6000 GPUs with 48 GB VRAM each, which also takes roughly 3 days. The network ϵ does not predict noise directly, instead adopting v-parameterization for preconditioning [9].

C.2 Inference

We generate conditional samples from the resulting diffusion model by running the EDM sampler [5] for 25 steps. SVD uses classifier-free guidance [4] at test time with a guidance strength w that linearly increases as a function of the video frame index (not the diffusion timestep) from start to end within the range $[1, 2.5]$ by default [1], but we found better performance by adjusting this range to $[1, 1.5]$ instead. Producing one output video takes roughly 10 seconds.

C.3 Coordinate system

We use a spherical coordinate system, where (ϕ, θ, r) represents the azimuth angle, elevation angle, and radial distance respectively. Note that as shown in Figure 3, θ is the *elevation* angle as measured starting from the XY-plane, which is not the same as the *inclination* angle as measured starting from the Z-axis.

C.4 Architecture

Figure 4 describes the model architecture in more detail. It is based on SVD [1], which in turn is based on Video LDM [2], modified for camera pose conditioning. The U-Net ϵ accepts input feature maps of dimensionality $2D \times T \times \frac{H}{F} \times \frac{W}{F}$, where D and F are the VAE embedding size and downsampling factor respectively, and produces output feature maps of dimensionality $D \times T \times \frac{H}{F} \times \frac{W}{F}$ that represent a less noisy sample. It consists of a factorized 3D U-Net that interleaves convolutional, spatial, and temporal blocks, of which the latter two establish correspondences between features across locations (per frame), and across time (per spatial position) respectively. Spatiotemporal attention can consequently take place between all pairs of input and output frames, as well as any pair of regions within both videos. Moreover, there are now $T = 14$ different CLIP embeddings $\{c(\boldsymbol{x}_t)\}$ that condition the U-Net layers at each matching frame. Specifically, these CLIP embeddings are fed to the network via multiple spatial (S-Attn)

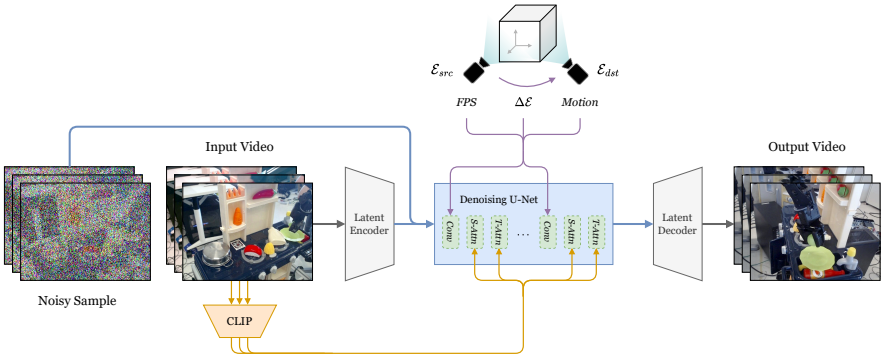

Fig. 4: Network architecture. Our model performs diffusion in latent space [6, 8]. The input video is encoded by a KL-VAE, and then channel-concatenated with the noisy sample. At training time, the output video is estimated and supervised; at inference time, multiple denoising steps are performed. In both cases, per-frame CLIP embeddings condition the U-Net by means of cross-attention, and and other relevant pieces of information (frame rate, desired camera pose transformation, and motion value) condition the U-Net by adding their embeddings onto the feature vectors in-between convolutions.

and temporal (T-Attn) cross-attention blocks throughout the network. Separately, the *micro-conditioning* mechanism takes place to pass the embeddings of the diffusion timestep, frame rate, camera transformation, motion bucket value, and conditioning augmentation strength to the network by summing it together with feature channels at various residual blocks placed throughout the network, with additional linear projections in-between to accommodate varying embedding sizes. Concretely, assuming the camera always looks at the same location in 3D space for simplicity,[1] the relative extrinsics matrix $\Delta\mathcal{E}$ is parameterized and given as $(\Delta\phi, \Delta\theta, \Delta r)$. The angles are subsequently encoded with Fourier positional encoding before being embedded through an MLP. Note that the input camera poses are not required to be known – only the desired relative transformation should be given.

C.5 Data and training

In Kubric-4D, pairs of input and output video clips are always temporally synchronized, but with $T = 14$ frame indices sampled randomly within the 60 available frames from the dataset. The original FPS is 24, and since the frame stride is randomly uniformly sampled among $\{1, 2, 3, 4\}$, the actual FPS when finetuning therefore belongs to $\{6, 8, 12, 24\}$. In ParallelDomain-4D, each scene has 50 frames available at 10 FPS, from which we randomly subsample clips but only at a frame stride in $\{1, 2\}$ determined by a coin flip, which implies an FPS value in $\{5, 10\}$.

[1] This is $(0, 0, 1)$, *i.e.* 1m above the center of the ground plane, in Kubric-4D.

In Kubric-4D, the camera pose $\mathcal{P} = (\phi, \theta, r)$ respects the following bounds (both across time and across input/output) with respect to the spherical coordinate system: azimuth angle $\phi_{1...T} \in [0°, 360°]$, elevation angle $\theta_{1...T} \in [0°, 50°]$, radial distance $r_{1...T} \in [12, 18]$.[2] The target camera pose transformation for the default model (*max 90°*) has a limited maximum transformation "strength" in the sense that from start to end, the azimuth, elevation, and radius all vary within the following bounds: $|\Delta\phi| \leq 90°, |\Delta\theta| \leq 30°, |\Delta r| \leq 3$. The horizontal field of view is 53.1° everywhere.

For the more extreme view synthesis variant (*max 180°*), the bounds are: $\phi_{1...T} \in [0°, 360°], \theta_{1...T} \in [0°, 90°], r_{1...T} \in [12, 18], |\Delta\phi| \leq 180°, |\Delta\theta| \leq 60°, |\Delta r| \leq 3$.

The trajectories are typically uniformly sampled, except for the elevation angle θ; in this case, uniform sampling for the starting point happens in terms of $\sin\theta$ instead of the angle θ directly. This is done in order to ensure an equal spread over (*i.e.* a uniform distribution on the surface of) the (relevant subset of the) unit sphere. The input camera extrinsics $\mathcal{E}_{src,t}$ is static, and the output camera extrinsics $\mathcal{E}_{dst,t}$ interpolates linearly over time in pose description space, *i.e.* in spherical coordinates with $\alpha = \frac{t}{T-1}$.

In ParallelDomain-4D, the source viewpoint is a forward-facing camera mounted on the virtual ego car at a fixed position of $(1.6, 0, 1.55)$ in 3D world space, where X points forward and Z points up. For simplicity, the camera pose is not controllable in the experiments described in our paper – instead, the destination viewpoint is fixed at $(-8, 0, 8)$, looking forward and down at $(5.6, 0, 1.55)$. To maximize the temporal smoothness of the generated video, the camera trajectory is interpolated in Euclidean space, not linearly but rather according to a sine wave function, *i.e.* following $\alpha = \frac{1-\cos\left(\frac{t}{\pi(T-1)}\right)}{2}$, assuming t increases step-wise from 0 to $T-1$. The horizontal field of view is 85° everywhere.

Early on in our experiments, we observed that synchronizing the motion bucket value, which conditions the model, with the strength of the camera transformation leads to better performance. Therefore, for Kubric-4D, we linearly scale this value along with the magnitude of the relative camera rotation (specifically, the $\mathcal{L}_2$ norm of $(\Delta\phi, \Delta\theta)$) where the minimum value corresponds to 0 and the maximum value corresponds to 255. This indication of camera motion hints the model that it should generate a video with a high degree of optical flow when the relative angles are high and vice versa.

We keep conditioning augmentation [2] enabled with a noise strength of 0.02.

C.6 Loss

We apply a focal $\mathcal{L}_2$ loss function between the estimated and ground truth latent feature maps, which focuses on the top fraction of embeddings incurring

[2] Since the dataset is synthetic and the radius r does not have an inherent meaning, it is worth nothing that the average diameter of an object is 1.88m, and that all objects are randomly spawned within these bounds in Euclidean coordinates: $x \in [-7, 7], y \in [-7, 7], z \in [0, 7]$ (where Z is up).

the biggest mismatch. This fraction linearly decreases from 100% to 10% in the first 5000 iterations, and then remains constant at 10%. In addition, for semantic completion in ParallelDomain-4D, we weight the categories involving vehicles (*i.e. Bus, Car, Caravan/RV, ConstructionVehicle, Bicycle, Motorcycle, OwnCar, Truck, WheeledSlow*) and people (*i.e. Animal, Bicyclist, Motorcyclist, OtherRider, Pedestrian*) to be respectively 3× and 7× as important as other categories, by multiplying the loss values at the corresponding spatial positions with the appropriate scaling factor before averaging. We observe that this strategy tends to reduce false negative prediction rates, especially for visually smaller objects occupying fewer pixels.

C.7 Evaluation

For each dataset separately, all models and all variants are evaluated on the same test split. For each scene, we randomly sample a subclip within the available video with $T = 14$ frames and a variable frame rate chosen within the same range as during training time. Then, for Kubric-4D, four different target camera poses (with angles up to azimuth $\pm 90°$ for Kubric-4D) are randomly sampled once. To encourage difficult input videos with higher than average degrees of occlusion, we set the starting elevation angle to be always $\theta_1 = 5°$, but all other angles are chosen randomly within the same ranges as during training. These randomization parameters at test time are only chosen once and then fixed across all evaluation experiments. We let probabilistic (*i.e.* diffusion) models (Ours, Vanilla SVD, ZeroNVS) generate four samples for each of these trajectories, averaging results, but the other methods (HexPlane, 4D-GS, DynIBaR) are only executed once for each scene and for each set of output camera angles.

C.8 Baselines

Vanilla SVD [1]. Since Stable Video Diffusion's last training stage involved finetuning at a resolution of 1024×576, and changing the resolution at test time gives rise to artefacts, it is probably optimal to evaluate the model at its original resolution. We center crop and resize all input images and target videos as needed to 1024×576 when evaluating this baseline. We keep the motion bucket at its default value of 127.

ZeroNVS [10]. Like Zero-1-to-3 [7], ZeroNVS was trained only on square images of resolution 256×256. Similarly to Vanilla SVD, we center crop and resize all input and ground truth frames accordingly. Moreover, since ZeroNVS learns a scale-invariant means of transforming camera poses in a way that depends on estimated depth maps, the translation component of the relative camera extrinsics matrix $\mathcal{E}_{src}^{-1} \cdot \mathcal{E}_{dst}$ fed to the model can incur variable meanings with respect to absolute 3D space depending on the observed scene. A scale parameter is hence tuned visually for each video separately until the output qualitatively aligns with the ground truth.

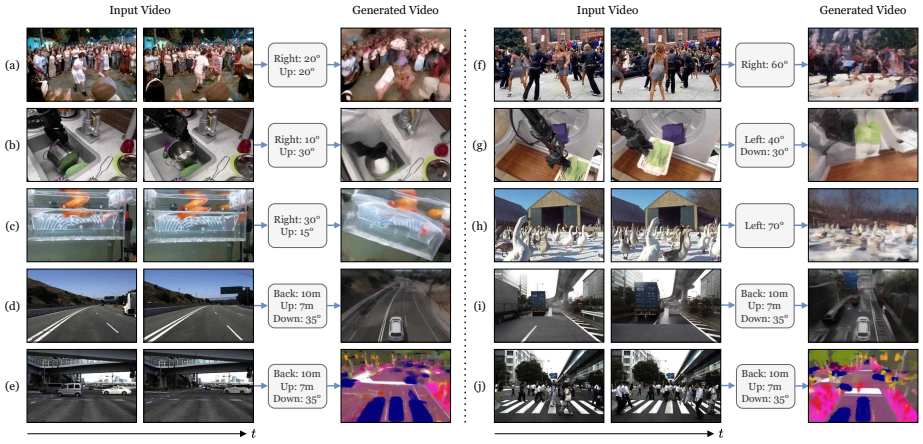

Fig. 5: Failure cases. We show inputs and predictions of real-world examples. Since deformable objects are not present in our Kubric-4D finetuning set, our model occasionally struggles with reconstructing their shape, appearance, and motion correctly. This can sometimes lead to objects becoming vague or blending in with each other. Similarly, videos in the bottom two rows are possibly related to them bordering on being out-of-distribution with respect to ParallelDomain-4D.

D Failure cases

Our model exhibits strong performance in many cases, but also fails to accurately generalize to some real-world videos, especially those involving humans, animals, or deformable objects. In Figure 5, we show representative failure cases. In (a), while the general layout is somewhat preserved, the people themselves become blurry. In (b), the robot arm gets cut off when performing view synthesis from the top, presumably because Kubric-4D does not contain robots or robotic motion patterns. In (c), the model appears to be confused as to what the initial camera pose is, and interprets it as a top-down rather than a sideways perspective of an aquarium, which leads to a roll effect when rotating the azimuth. In (d), the highway sign gets missed. In (e), the overpasses are not reconstructed, which seems to cause blurriness in the rest of the prediction. In (f), (g), and (h), both shape and dynamics are not well-respected. In (i), the perceived depth of the large blue truck is wrong. In (j), there are an unusually large amount of pedestrians crossing the street, which the model groups into "cars".

References

1. Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al.: Stable video diffusion: Scaling latent video diffusion models to large datasets. arXiv preprint arXiv:2311.15127 (2023)

2. Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S.W., Fidler, S., Kreis, K.: Align your latents: High-resolution video synthesis with latent diffusion models. In: CVPR (2023)
3. Cao, A., Johnson, J.: Hexplane: A fast representation for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 130–141 (2023)
4. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
5. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. Advances in Neural Information Processing Systems **35**, 26565–26577 (2022)
6. Ling, H., Kim, S.W., Torralba, A., Fidler, S., Kreis, K.: Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. arXiv preprint arXiv:2312.13763 (2023)
7. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9298–9309 (2023)
8. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
9. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
10. Sargent, K., Li, Z., Shah, T., Herrmann, C., Yu, H.X., Zhang, Y., Chan, E.R., Lagun, D., Fei-Fei, L., Sun, D., et al.: Zeronvs: Zero-shot 360-degree view synthesis from a single real image. arXiv preprint arXiv:2310.17994 (2023)
11. Wikipedia contributors: Spherical coordinate system — Wikipedia, the free encyclopedia (2024), `https://en.wikipedia.org/wiki/Spherical_coordinate_system`, [Online; accessed 2024]