# Text-To-4D Dynamic Scene Generation

**Uriel Singer** * **Shelly Sheynin** * **Adam Polyak** * **Oron Ashual** **Iurii Makarov** **Filippos Kokkinos** **Naman Goyal**
**Andrea Vedaldi** **Devi Parikh** **Justin Johnson** **Yaniv Taigman**

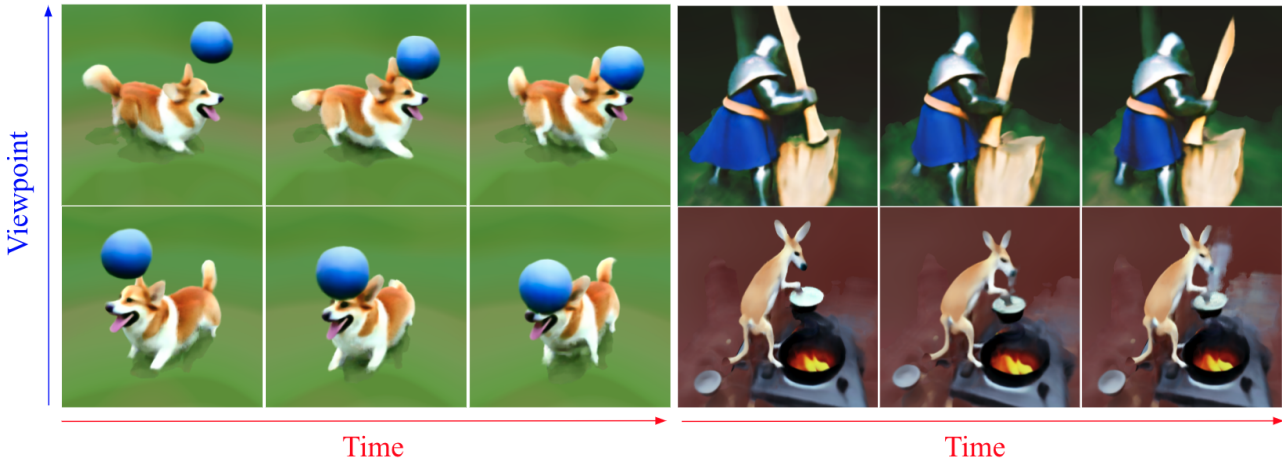



Figure 1: Samples generated by MAV3D along the temporal and viewpoint dimensions. Left: "A corgi playing with a ball". Right top:"A knight chopping wood". Right bottom: "A kangaroo cooking a meal".

## Abstract

We present MAV3D (**M**ake-**A**-**V**ideo**3D**), a method for generating three-dimensional dynamic scenes from text descriptions. Our approach uses a 4D dynamic Neural Radiance Field (NeRF), which is optimized for scene appearance, density, and motion consistency by querying a Text-to-Video (T2V) diffusion-based model. The dynamic video output generated from the provided text can be viewed from any camera location and angle, and can be composited into any 3D environment. MAV3D does not require any 3D or 4D data and the T2V model is trained only on Text-Image pairs and unlabeled videos. We demonstrate the effectiveness of our approach using comprehensive quantitative and qualitative experiments and show an improvement over previously established internal baselines. To the best of our knowledge, our method is the first to generate 3D dynamic scenes given a text description. Generated samples can be viewed at make-a-video3d.github.io.

*Equal contribution. Meta AI.
Correspondence to: Uriel Singer <urielsinger@meta.com>, Shelly Sheynin <shellysheynin@meta.com>, Adam Polyak <adampolyak@meta.com>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. 

## 1. Introduction

Generative models have seen tremendous recent progress, and can now generate realistic images from natural language prompts (Ramesh et al., 2022; Gafni et al., 2022; Rombach et al., 2022; Saharia et al., 2022; Yu et al., 2022; Sheynin et al., 2022). This success has been extended beyond 2D images both *temporally* to synthesize videos (Singer et al., 2022; Ho et al., 2022) and *spatially* to produce 3D shapes (Poole et al., 2022; Lin et al., 2022; Nichol et al., 2022b). However, these two categories of generative models have been studied in isolation to date.

In this paper we combine the benefits of video and 3D generative models and propose a novel system for *text-to-4D* (3D+time) generation. Our method, named MAV3D (**M**ake-**A**-**V**ideo**3D**), takes as input a natural-language description and outputs a dynamic 3D scene representation which can be rendered from arbitrary viewpoints. Such a method could be used to generate animated 3D assets for video games, visual effects, or augmented and virtual reality.

Differently from image and video generation where one can train on large quantities of captioned data, there is no

readily available collection of 4D models, with or without textual annotations. One approach might be to start from a pre-trained 2D video generator (Singer et al., 2022) and distill a 4D reconstruction from generated videos. Still, reconstructing the shape of deformable objects from video is a very challenging, widely known as Non-Rigid Structure from Motion (NRSfM). The task becomes simpler if one is given *multiple simultaneous viewpoints* of the object. While multi-camera setups are rare for real data, our insight is that existing video generators *implicitly model arbitrary viewpoints for generated scenes.* We can thus use a video generator as a 'statistical' multi-camera setup to reconstruct the geometry and photometry of the deformable object. Our MAV3D algorithm does so by optimizing a dynamic Neural Radiance Field (NeRF) jointly with decoding the input text into a video, sampling random viewpoints around the object.

Naively optimizing dynamic NeRF using video generators does not produce satisfying results and there are several significant challenges that must be overcome toward this goal. First, we need an effective *representation* for dynamic 3D scenes that is efficient and learnable end-to-end. Second, we need a source of *supervision* since there are no large-scale datasets of (text, 4D) pairs from which to learn. Third, we need to scale the *resolution* of the outputs in both space and time which is both memory- and compute-intensive due to the 4D output domain.

For our *representation*, we build on recent advances in neural radiance fields (NeRFs) (Mildenhall et al., 2021). We combine insights from work on efficient (static) NeRFs (Sun et al., 2022; Müller et al., 2022) and dynamic NeRFs (Cao & Johnson, 2023), and represent a 4D scene as a set of six multiresolution feature planes.

To *supervise* this representation without paired (text, 4D) data, we propose a multi-stage training pipeline for dynamic scene rendering and demonstrate the importance of each component in achieving high-quality results. One key observation is that directly optimizing a dynamic scene using Score Distillation Sampling (SDS) (Poole et al., 2022) using Text-to-Video (T2V) model leads to visual artifacts and sub-optimal convergence. Therefore, we first utilize a Text-to-Image (T2I) (Singer et al., 2022) model to fit a static 3D scene to a text prompt and subsequently augment our 3D scene model with dynamics. Additionally, we introduce a new temporal-aware SDS loss and motion regularizers that prove to be crucial for realistic and challenging motion.

We scale to higher *resolution* outputs with an additional phase of temporal-aware super-resolution fine-tuning. We use SDS from the super-resolution module of the T2V model to obtain high-resolution gradient information to supervise our 3D scene model, increasing its visual fidelity and allowing us to sample higher-resolution outputs during inference.

Our main contributions are:

- We introduce MAV3D, an effective method that utilizes T2V model and dynamic NeRFs in order to integrate world knowledge into 3D temporal representations.
- We propose a multi-stage static-to-dynamic optimization scheme that gradually incorporates gradient information from static, temporal, and super-resolution models, to enhance the 4D scene representation.
- We conduct a comprehensive set of experiments, including ablation studies, using both quantitative and qualitative metrics to reveal the technical decisions made during the development of our method.

## 2. Related work

**Neural rendering.** Our 3D scene representation builds upon recent advances in neural rendering. Neural radiance fields (NeRFs) (Mildenhall et al., 2021) represent a 3D scene with a neural network that inputs scene coordinates, and form images with volume rendering. Recent work has improved efficiency by incorporating 3D data structures such as voxel grids (Sun et al., 2022) that may be sparse (Fridovich-Keil et al., 2022) or multiresolution (Takikawa et al., 2021), and which can be further accelerated via tensor factorization (Chen et al., 2022) or hashing (Müller et al., 2022).

**Dynamic neural rendering.** We aim to generate *dynamic* scenes which can be viewed from any angle. This relates to classic work on free-viewpoint video which use videos of a moving scene to synthesize novel views (Carranza et al., 2003; Smolic et al., 2006; Collet et al., 2015). Recent approaches make NeRFs dynamic by conditioning the network on both space and time (Martin-Brualla et al., 2021; Li et al., 2022) and may incorporate additional supervision from depth (Xian et al., 2021) or scene flow (Li et al., 2021; Du et al., 2021; Gao et al., 2021). Another category of approaches learn a time-varying *deformation* of 3D points into a static *canonical* scene (Pumarola et al., 2021; Park et al., 2021a; Tretschk et al., 2021; Park et al., 2021b). Some approaches accelerate NeRFs on dynamic scenes using explicit voxel grids (Fang et al., 2022) or tensor factorization (Cao & Johnson, 2023; Fridovich-Keil et al., 2023).

**Text to 3D.** The idea of generating 3D scenes from text dates back decades (Adorni & Di Manzo, 1983); early efforts parsed geometric relations from text and built scenes from a library of known objects (Coyne & Sproat, 2001; Chang et al., 2014). Some approaches train neural networks end-to-end on paired datasets of text and shape (Chen et al., 2018; Nichol et al., 2022b) but this approach is difficult to scale due to the paucity of paired data. Other approaches instead generate 3D shapes from text without paired data using a pretrained CLIP (Radford et al., 2021) model (Jetchev, 2021; Sanghi et al., 2022; Wang et al., 2022; Jain et al., 2022a)

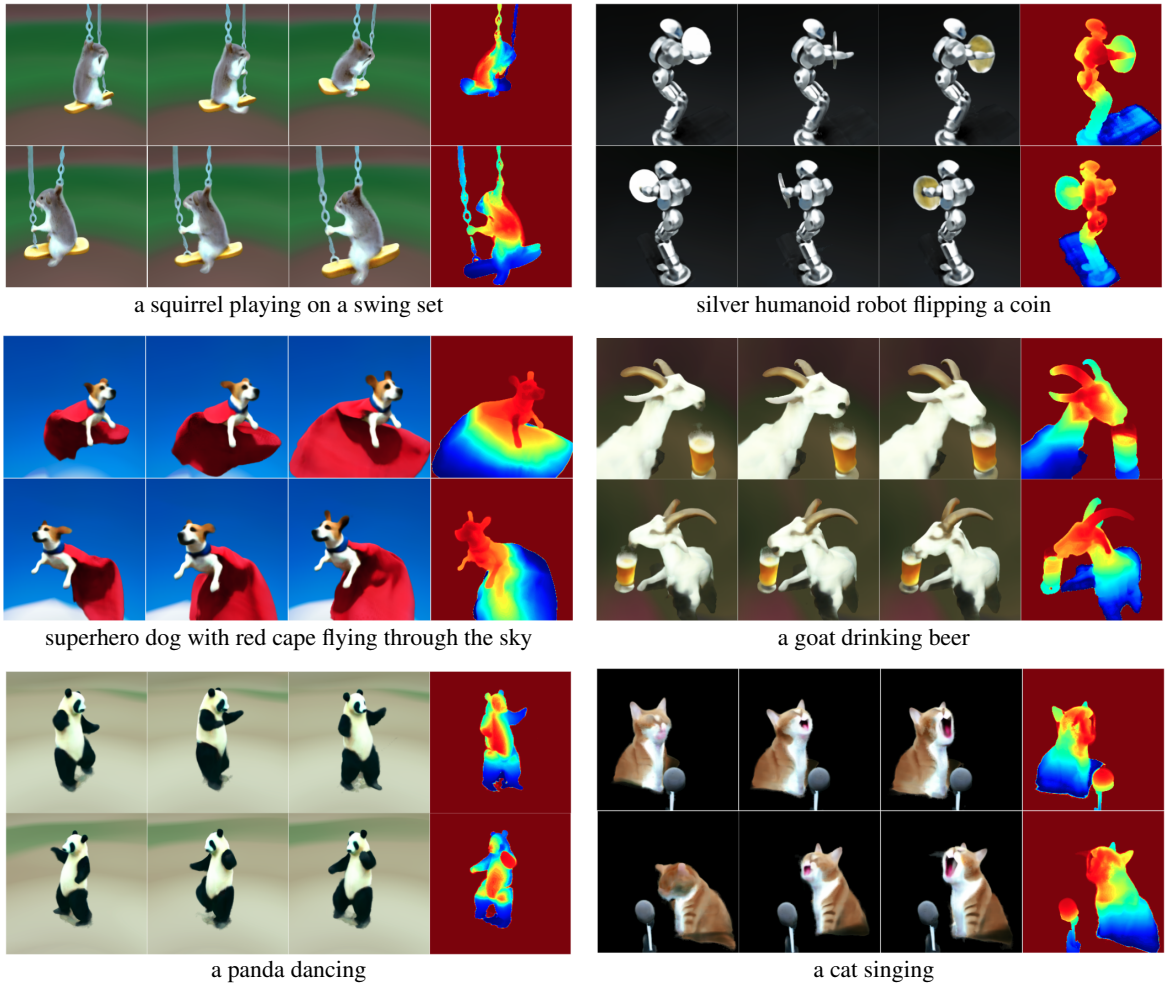



Figure 2: Samples generated by MAV3D. The rows represent variations in time, and the columns represent variations in viewpoint. The last column shows the depth image of its adjacent column.

or a text-to-image diffusion model (Poole et al., 2022; Lin et al., 2022). We use a similar strategy, but generate 4D rather than 3D content using a text-to-video model.

**Diffusion-based generative models.** Recent improvements in diffusion models (Dhariwal & Nichol, 2021) have led to highly advanced image synthesis (Ramesh et al., 2021; Esser et al., 2021; Rombach et al., 2022; Gafni et al., 2022; Nichol et al., 2022a; Ramesh et al., 2022; Saharia et al., 2022) and the creation of generative models for other forms of media, such as video(Singer et al., 2022; Ho et al., 2022; Villegas et al., 2022). Our video generator is based on Make-A-Video (MAV) (Singer et al., 2022), which expands upon a Text-To-Image (T2I) model by training on unlabeled videos.

## 3. Method

Our goal is to develop a method that produces a dynamic 3D scene representation from a natural-language description. This is a challenging task since we have neither (text, 3D) pairs nor dynamic 3D scene data for training. Instead, we rely on a pretrained *text-to-video* (T2V) diffusion model (Singer et al., 2022) as a scene prior, which has learned to model realistic appearance and motion of scenes by training on large-scale image, text, and video data.

At a high level, given a text prompt $p$ we fit a 4D scene representation $f_\theta(x, y, z, t)$ that models the appearance of a scene matching the prompt at arbitrary points in spacetime. Without paired training data, we cannot directly supervise the outputs from $f_\theta$; however given a sequence of camera poses $\{C_t\}_{t=1}^{T}$ we can *render* a sequence of images $I_t = \mathcal{R}(f_\theta, t, C_t)$ from $f_\theta$ and stack them to form a video $V$. Then, we can pass the text prompt $p$ and the video $V$ to a frozen, pretrained T2V diffusion model which scores the video's realism and alignment to the prompt; we can then use *Score Distillation Sampling* (SDS) (Poole et al., 2022) to compute an update direction for the scene parameters $\theta$.

The above pipeline can be seen as an extension of DreamFusion (Poole et al., 2022), adding a temporal dimension

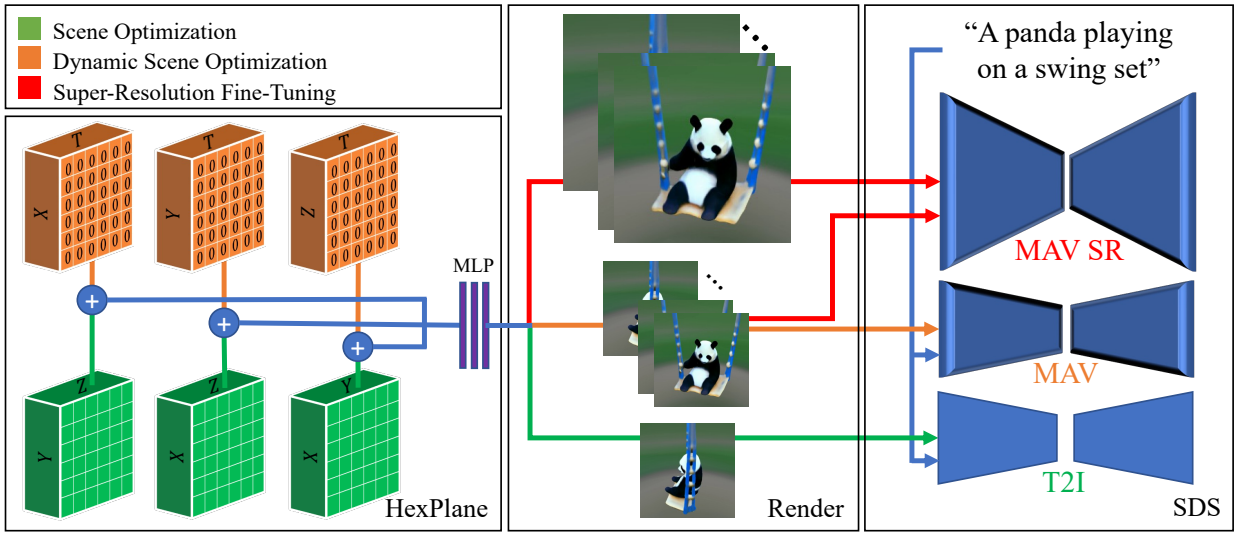



Figure 3: **Full pipeline of MAV3D.** First, we leverage only the three pure-spatial planes (colored green), render a single image, and calculate the SDS loss using the T2I model. In the second stage, we add the additional three planes (colored orange) which are initialized to zeros for smooth transition, render a full video and calculate the SDS-T loss using the T2V model. In the third stage (SRFT), we additionally render high-resolution video which is passed as input to the super-resolution component, with the low-resolution as condition.

to the scene model and supervising with a a T2V model rather than a *text-to-image* (T2I) model. However, we found that high-quality text-to-4D generation requires additional significant innovations. First, we use a new 4D scene representation (Section 3.1) which allows flexible modeling of scene motion. Second, we enhance video quality and improve model convergence with a multi-stage *static-to-dynamic* optimization scheme (Section 3.2) that utilizes several *motion regularizers* to encourage realistic motion. Third, we improve the *resolution* of the model using *super-resolution fine-tuning* (SRFT) (Section 3.3). See Fig. 3 for an illustration of the method.

### 3.1. 4D Scene Representation

Following recent advances in neural rendering (Mildenhall et al., 2021), we represent a dynamic 3D scene *implicitly*. Given a timestep $t$ and camera position, for each image pixel we cast a ray through the camera plane into the scene and sample a set of points $\{\mu_i\}_{i=1}^{N}$ along the ray; for each point we compute a *volume density* $\tau_i \geq 0$ and color $c_i$, and the color for the ray is computed via volume rendering. Similar to NeRF (Mildenhall et al., 2021), we output the color $c_i$ from an MLP, but assume that the color is view-independent (Lambertian). We explored generating albedo (scene color) and random light sources like DreamFusion (Poole et al., 2022) but found that it significantly slows training without improving quality. Our learnable scene model is thus a function $(\tau, c_i) = f_\theta(x, y, z, t)$ that outputs volume density and color for arbitrary points in spacetime.

We must then choose a suitable architecture for $f_\theta$. DreamFusion (Poole et al., 2022) adopts a variant of Mip-NeRF (Barron et al., 2022) for recovering static 3D structure from images. By analogy, we can adopt any architecture for recovering dynamic 3D structure from videos. Such architectures are often designed to operate with as few as one input view, and thus include strong scene priors (*e.g.*, temporal deformations from a static canonical scene (Pumarola et al., 2021; Park et al., 2021a)) that enable reconstruction from sparse views.

However, in our setting we need not learn from sparse views; SDS on a pretrained T2V model enables us to supervise the model along arbitrary view trajectories during training. As such, rather than adopting an architecture which regularizes and restricts scene motion, we instead would like a high-capacity architecture that can flexibly model large scene motion. We thus adopt HexPlane (Cao & Johnson, 2023), a recently proposed representation for dynamic scenes.

**HexPlane** represents a 4D scene with six *planes* of feature vectors spanning all pairs of axes in $\{X, Y, Z, T\}$. It computes an $(R_1 + R_2 + R_3)$-dimensional feature for a spacetime point $(x, y, z, t)$ via projection onto each plane:

$$[P_{xy}^{XYR_1} + P_{zt}^{ZTR_1}; P_{xz}^{XZR_2} + P_{yt}^{YTR_2}; P_{yz}^{YZR_3} + P_{yz}^{XTR_3}] \quad (1)$$

Superscripts denote shapes ($P^{XYR_1}$ has shape $X \times Y \times R_1$, and is a plane of $R_1$-dim features spanning the $XY$ axes), subscripts denote sampling via bilinear interpolation, and ; is concatenation. The resulting feature is passed to a small MLP which predicts volume density and color.

We further increase the capacity of the HexPlane model by

representing each plane as a multi-resolution grid, similar to (Müller et al., 2022) with the hash function removed. We also use a *background model* simulating a large (static) sphere surrounding the (dynamic) foreground modeled by the HexPlane; it is a small MLP receiving a sinusoidally-encoded ray direction and producing an RGB color. We also keep a coarse voxel grid of occupancy probabilities updated periodically via EMA to accelerate sampling. See Sec. A.3 of the supplementary for more details.

### 3.2. Dynamic Scene Optimization

Armed with the HexPlane model $f_\theta$ for 4D scenes, we must *supervise* it to match a textual prompt $p$. We introduce temporal *Score Distillation Sampling* (SDS-T) which is an extension of SDS (Poole et al., 2022) for pretrained conditional video generator. We first describe the loss, and then its application in MAV3D.

Recall that, given the camera trajectory $C$ and the scene parameters $\theta$, the rendering function $\mathcal{R}$ infers a parametric video $V_\theta$. We assume that the pretrained conditional video generator is based on diffusion and thus defines a *denoising function* $\hat{\epsilon}(V_{(\theta,\sigma,\epsilon)} \mid y, \sigma)$ which takes as input a noised video $V_{(\theta,\sigma,\epsilon)} = \sqrt{1-\sigma^2}V_\theta + \sigma\epsilon$ (where $\epsilon \in \mathcal{N}(0, I)$ is normal noise), the noise level $\sigma \in (0, 1)$, and additional conditioning information $y$ (textual prompt, video frame rate, etc.), and predicts an estimate $\hat{\epsilon}$ of the noise $\epsilon$.

We compute an update direction for the scene parameters $\theta$ using SDS: we add random noise $\epsilon$ to the current video $V_\theta$ to obtain the noised version $V_{(\theta,\sigma,\epsilon)}$, apply the denoiser network to obtain the noise estimate $\hat{\epsilon}$. The update direction for $\theta$ is then the (negative) gradient of the reconstruction loss $E_{\sigma,\epsilon}[w(\sigma)\|\hat{\epsilon}(V_{(\text{sg}(\theta),\sigma,\epsilon)} \mid y, \sigma) - \epsilon\|^2]$ averaged over $\epsilon$ and $\sigma$, where $w(\sigma)$ is a weighting function and $\text{sg}(\cdot)$ is the stop-grad operator.[1] The resulting SDS gradient is then

$$\nabla_\theta \mathcal{L}_{SDS-T} = E_{\sigma,\epsilon,t,fps}\left[w(\sigma)(\hat{\epsilon}(V^{(t,fps)}_{(\theta,\sigma,\epsilon)}|y,\sigma) - \epsilon)\frac{\partial V_\theta}{\partial \theta}\right] \tag{2}$$

Where $t$ is a random start time and $fps$ is a random video frame rate for the sub-sequence $V^{(t,fps)}_{(\theta,\sigma,\epsilon)} = \left[I^t_{(\theta,\sigma,\epsilon)}; I^{t+\frac{1}{fps}}_{(\theta,\sigma,\epsilon)}; \ldots; I^{t+\frac{16}{fps}}_{(\theta,\sigma,\epsilon)}\right]$ of 16 frames, where $I^t_{(\theta,\sigma,\epsilon)}$ denotes the frame $t$ in the video out of $T$ frames. We use the -T suffix to emphasise that, differently from (Poole et al., 2022), this version of SDS is applied to a video, i.e., a temporal image sequence.

**Static to dynamic.** In practice we found that directly optimizing a HexPlane using SDS from a pretrained T2V model leads to visual artifacts and sub-optimal convergence (see Sec. 4.2). We therefore adopt a multi-stage *static-to-dynamic* optimization scheme, first optimizing a *static* 3D scene matching the text prompt, then extending it to 4D.

During the first phase of static optimization, we fix the three temporal planes of the HexPlane to zero ($P^{ZTR_1}$, $P^{YTR_2}$, and $P^{XTR_3}$ in Equation 1); this is similar to the tri-plane representation used by (Chan et al., 2022). During each training iteration we sample a batch of 8 random camera poses, and render a $64\times64$ image from each view, applying view-dependent prompt engineering similar to DreamFusion. We supervise the model using SDS on the frozen T2I model from (Singer et al., 2022) which has been pretrained on a large dataset of images and text.

During the second phase of dynamic optimization, we continue updating all planes of the HexPlane. We render a batch of 8 $64\times64$ 16-frame videos from the model, and supervise it using SDS-T on the frozen T2V model from (Singer et al., 2022). We employ several *regularizers* during this phase to encourage high-quality 4D synthesis.

**Dynamic Camera.** Most videos (including those on which our T2V model was trained) have apparent motion from two sources: *object motion* and *camera motion*. During training we simulate camera motion by rendering videos from randomly generated dynamic camera trajectories.

Training with dynamic cameras gives 4D scenes with more pronounced and realistic object motion (See Section 4.2). We hypothesize that, if trained with a static camera, the HexPlane tries to model object *and* camera motion to close the domain gap with the T2V model, giving worse object motion. Dynamic cameras also reduce the *multi-face* problem common in DreamFusion: the T2V model can judge a video showing faces on both sides of an object as unrealistic.

**FPS Sampling.** Dynamic cameras randomize the *spatial* perspective from which we render videos during training; we also vary the *temporal* extent of training videos via FPS sampling. The T2V model accepts videos with $F$=16 frames, but also conditions on the *frame-rate* of those videos. For each training sample we randomly sample a frame-rate $\text{fps} \sim \mathcal{U}[0, 1/F]$ and start time $t_0 \sim \mathcal{U}[0, 1-F\cdot\text{fps}]$, where the 4D scene model assumes a temporal extent $t \in [0, 1]$.

**Gaussian Annealing.** DreamFusion biases toward central scene content by adding Gaussian-distributed density to the output from the scene model before rendering; we also use this bias during static optimization. However in dynamic scenes, content should be allowed to move away from the origin; we thus find it helpful to linearly increase the width of the Gaussian density bias during dynamic optimization.

**Total Variation Loss.** We encourage spacetime smoothness in our 4D representing by applying the following Total Variation (TV) regularizer (following (Niemeyer et al., 2022))

[1] (Poole et al., 2022) use a more complex definition of $\mathcal{L}_{\text{SDS}}$; this simpler form gives the same gradient up to a scale factor.

to each of the six planes $P$ of the HexPlane:

$$\mathcal{L}_{\text{TV}}(P;\beta) = \sum_{i,j} \left((P_{i,j+1} - P_{i,j})^2 + (P_{i+1,j} - P_{i,j})^2\right)^{\frac{\beta}{2}} \tag{3}$$

The standard TV norm is obtained for $\beta = 1$; however, we found that this resulted in noisy high-frequency artifacts in our case, similar to (Mahendran & Vedaldi, 2015); we follow the latter and set $\beta = 2$ to encourage smoothness.

### 3.3. Super-Resolution Fine-Tuning

During the static and dynamic scene optimization phases described above, our 4D scene representation is supervised via low-resolution $64 \times 64$ renderings; we found that rendering higher-resolution videos from the learned model can lack detail and exhibit artifacts. We overcome this problem with a final phase of *super-resolution fine-tuning* (SRFT).

During SRFT we make use of the pretrained and frozen video super-resolution module $\text{SR}_l^t$ from (Singer et al., 2022). This diffusion-based model inputs a high-resolution noisy $256\times256$ video along with a clean $64\times64$ low-res video, and predicts the noise of the high-resolution video.

We use $\text{SR}_l^t$ to improve high resolution renderings from our 4D scene model. During each training iteration we sample a $256\times256$ video $V_\uparrow$ from our scene model and downsample it to a $64\times64$ video $V_\downarrow$. These are used to compute an SDS gradient for $V_\uparrow$ using $\text{SR}_l^t$.

$\text{SR}_l^t$ does not condition on the text prompt $p$. Fine-tuning via SDS on $\text{SR}_l^t$ alone thus encourages realistic high-resolution videos, but not alignment to $p$; this can cause the model to collapse to an empty scene. During SRFT we therefore train jointly using SDS from $\text{SR}_l^t$ and SDS-T from Equation 2. The final equation for the SR phase is:

$$\nabla_\theta \mathcal{L}_{SDS-SR} =$$
$$(1-\alpha) \times E_{\sigma,\epsilon,t}\left[w(\sigma)(\hat{\epsilon}(V_{\downarrow(\theta,\sigma,\epsilon)}^{(t,fps)}|y,\sigma) - \epsilon)\frac{\partial V_\theta}{\partial\theta}\right] +$$
$$\alpha \times E_{\sigma,\epsilon,t}\left[w(\sigma)(\hat{\epsilon}(V_{\uparrow(\theta,\sigma,\epsilon)}^{(t,fps)}|y,\sigma,V_{\downarrow\theta}) - \epsilon)\frac{\partial V_\theta}{\partial\theta}\right]$$

Where $\alpha$ is an adjustable control parameter, which, through validation, has been set to 0.2 in all of our experiments.

## 4. Experiments

Our experiments evaluate the ability of MAV3D to generate dynamic scenes from text descriptions. First, we evaluate the effectiveness of our approach on the Text-To-4D task. To the best of our knowledge MAV3D is the first method to tackle this task, so we develop three alternative methods as baselines. Second, we evaluate simplified versions of our model on the sub-tasks of T2V and Text-To-3D, and compare them to existing baselines in the literature. Third, we conduct a comprehensive ablation study to justify our method's design. Fourth, we describe our procedure for converting dynamic NeRFs into dynamic meshes, and finally present an extension of our model to the Image-to-4D task.


Figure 4: An ablation on **temporal-aware super-resolution optimization**. **Top:** without super-resolution phase. **Bottom:** MAV3D results. The red square is a zoom-in area of the image. This stage enhances the quality of the rendered videos, resulting in high-resolution videos with finer details

**Metrics.** We evaluate the generated videos using CLIP R-Precision (Jain et al., 2022b), which measures the consistency between the text and the generated scene. The reported metric is the retrieval accuracy of input prompts from rendered frames. We use the `ViT-B/32` variant of CLIP (Wang et al., 2022) and extract frames in varying views and time steps. We also use four qualitative metrics by asking human raters their preferences among two generated videos based on: (i) video quality; (ii) faithfulness to the textual prompt; (iii) amount of motion; and (iv) realism of motion. We evaluated all baselines and ablations on the text prompts splits which were used in (Singer et al., 2022).

**Samples.** We show samples in Figures 1 and 2. For more detailed visualization, please see make-a-video3d.github.io.

### 4.1. Results

**Text-to-4D comparison.** As there were no previous methods for Text-To-4D, we established three baselines for comparison. The baselines are based on a T2V generative

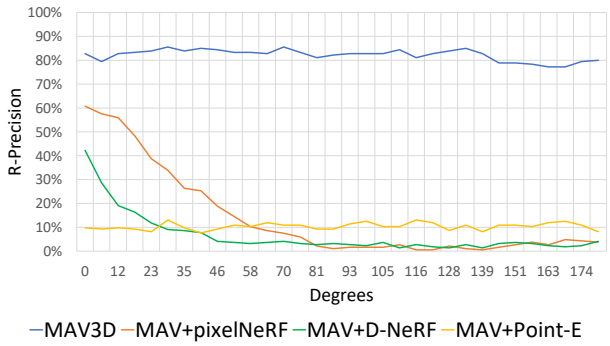



Figure 5: R-Precision per viewing angle. We render per method from cameras in a circle around the scene.

Table 1: Comparison with baselines (R-Precision and human preference). **Human evaluation is shown as a percentage of majority votes in favor the baseline compared to our model in the specific setting**.

| D | Model | R-Precision↑ | Video quality | Text-align. | More motion | Realistic motion |
|---|---|---|---|---|---|---|
| 4D | MAV+pixelNeRF | 8.8 | 0.11 | 0.16 | 0.18 | 0.16 |
| | MAV+Point-E | 10.6 | 0.20 | 0.06 | 0.26 | 0.17 |
| | MAV+D-NeRF | 6.2 | 0.17 | 0.18 | 0.19 | 0.30 |
| | **MAV3D** | 83.7 | - | - | - | - |
| 3D | Stable DF | 66.1 | 0.46 | 0.36 | - | - |
| | Point-E | 12.6 | 0.15 | 0.16 | - | - |
| | **MAV3D\t** | 82.4 | - | - | - | - |
| Video | MAV | 86.6 | 0.73 | 0.63 | 0.64 | 0.62 |
| | **MAV3D\z** | 79.2 | - | - | - | - |

method (Make-A-Video (Singer et al., 2022)), which generates a sequence of 2D frames from a text prompt. Once generated, the sequence of 2D frames is transformed into a sequence of 3D scene representations using three different methods. The first sequence is produced by applying a one-shot neural scene renderer (Point-E (Nichol et al., 2022b), the second via pixelNeRF (Yu et al., 2021)) on each frame independently, and third by applying D-NeRF (Pumarola et al., 2021) combined with camera position extracted using COLMAP (Schönberger & Frahm, 2016). We denote these adapted baselines as MAV+Point-E, MAV+pixelNeRF and MAV+D-NeRF, respectively. For the Point-E baseline, we use the `base1B` variant released by the authors. For the D-NeRF baseline, we use only videos with valid COLMAP results and substantial camera motion. Results are presented at the top of Table 1. As can be seen, our method surpasses the naive baselines in the objective R-Precision metric and is highly preferred by human raters across all metrics.

Furthermore, we explore our method performance for different camera viewing angles. This is done by calculating R-precision for frames rendered from different camera positions. Specifically, given a camera position $d = (R, \theta, \phi)$, we fix the zoom, $R$, and the tilt, $\theta$, and report R-Precision for different pan values - $\phi$. In Fig 5 we show our method is able to render the scene consistently across viewing angles while the MAV+D-NeRF and MAV+pixelNeRF performance deteriorates as $\phi$ increases. MAV+Point-E is also able to maintain a consistent R-Precision score.

Table 2: Ablation study (R-Precision and human preference (%) . **Human evaluation is shown as a percentage of majority votes in favor the baseline vs our model**.

| Model | R-Precision↑ | Video quality | Text-align. | More motion | Realistic motion |
|---|---|---|---|---|---|
| w/o SR | 84.3 | 0.41 | 0.34 | 0.42 | 0.36 |
| w/o pretraining | 63.5 | 0.27 | 0.44 | 0.46 | 0.35 |
| w/o dynamic camera | 83.6 | 0.54 | 0.48 | 0.35 | 0.42 |
| w/o gaussian anneal. | 76.3 | 0.47 | 0.50 | 0.45 | 0.38 |
| with D-NeRF | 81.9 | 0.45 | 0.47 | 0.50 | 0.47 |
| with Instant NGP | 78.4 | 0.36 | 0.40 | 0.48 | 0.42 |

**Text-to-3D comparison.** To evaluate our method on the Text-to-3D (3D) task, we remove the temporal dimension from our rendered video. Specifically, we sample MAV3D from a single time step, and denote this reduction as **MAV3D\t**. We compare this variant with: (i) Stable-DreamFusion (Stable-DF) (Tang, 2022), a public re-implementation of DreamFusion (Poole et al., 2022), and, (ii) Point-E (Nichol et al., 2022b), which generates a point cloud given a text prompt. The results are presented in Table 1. In this setting, the input to each baseline is the text prompt. Since the `base1B` variant of Point-E expects image input, we utilize a T2I model that generates an image which is then fed to the model. As can be seen, our model is preferred over these variants in quality and text alignment.

**Text-to-Video comparison.** To evaluate our method on the sub-task of T2V (Video), we remove the depth dimension from our method output. Concretely, we sample frames from our model on specific viewing directions (front, back, and side), reduceing our method from temporal dynamic scene generation to video generation. We denote this reduction as **MAV3D\z**. This variant is compared to videos generated with Make-A-Video by appending the viewing direction to the textual input prompt. Results are presented at Table 1. Note that our method utilizes Make-A-Video as a training objective and is thus bounded by its performance.

### 4.2. Ablation study

An ablation study of human preference and R-precision is provided in Tab. 2 to assess the effectiveness of our different contributions.

**Ablation on different training stages.** (i) *without SR:* a model trained without the scene super-resolution fine-tuning, for the same number of steps as MAV3D (stage 3). As can

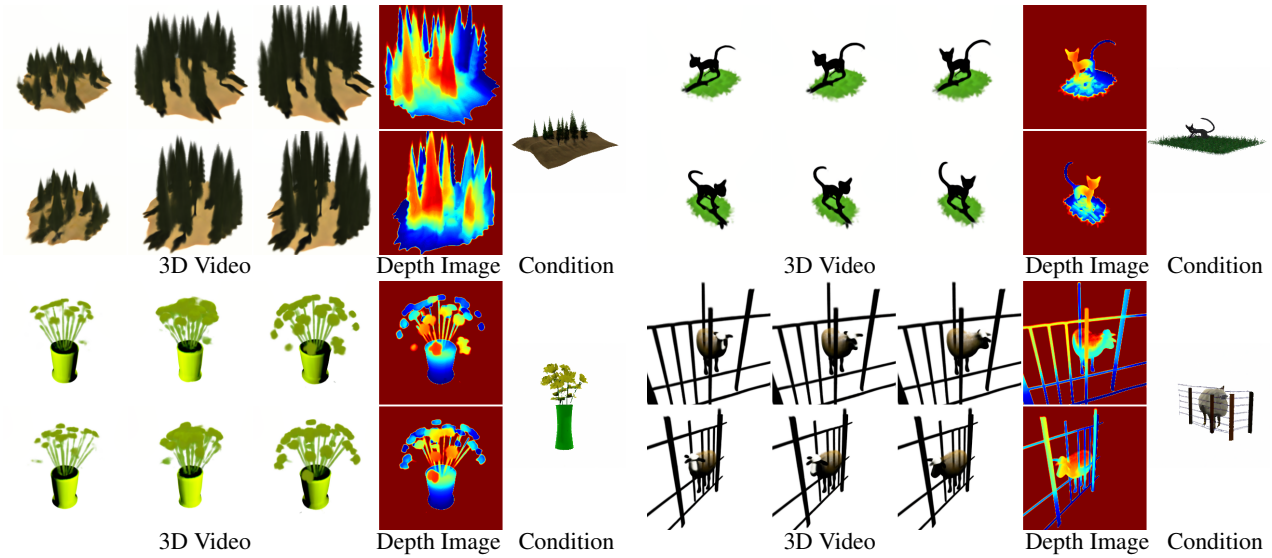



Figure 6: **Image to 4D application.** Given an input image we extract its CLIP embedding, and use it to condition MAV3D.

be seen, human raters favors the model trained with SR in both quality, text alignment and motion. In addition, as demonstrated in Fig. 4, the super-resolution fine-tuning enhances the quality of the rendered videos, resulting in high-resolution videos with finer details (e.g. the hands for the fox) and less noise. (ii) *without pre-training:* As illustrated in Tab. 2 and in Fig. 8, directly optimizing the dynamic scene (without the static scene pre-training) for the same number of steps as MAV3D, results in much lower scene quality or poor convergence: the model trained with static-pretraining is preferred for video quality and realistic motion in 73% and 65% of cases, respectively. Additional ablation of the number of static pretrain steps is available in the supplementary.

**Ablation on motion regularizers components.** (i) *dynamic camera:* here, we train a variant of our method in which the camera position is fixed across all frames. We observe that videos rendered using this variant obtain less motion, and suffer from multi-face object (see Fig. 7). This may explain the relatively high R-Precision score (since with multiple faces, the object can be recognized more easily). (ii) *Gaussian annealing:* Extending the spatial bias of the model (to focus on the larger "blob") leads to renderings with larger and more realistic motion.

**Ablation on NeRF backbone.** (i) *with D-NeRF:* to quantify the contribution of the temporal NeRF, we analyzed a variant of our method in which we replaced our temporal NeRF backbone (HexPlane) with D-NeRF (Pumarola et al., 2021; Fang et al., 2022). While HexPlane is slightly preferred in terms of overall quality and realistic motion, our approach is not sensitive to the dynamic backbone, demonstrating the robustness of our method. (ii) *with Instant-NGP:* here we replace our static NeRF backbone with Instant-NGP (Müller et al., 2022). This variation employs hash encoding and includes a color network that emits radiance values, and it is significantly less preferred.

### 4.3. Real-time rendering

Applications such as virtual reality and games that use traditional graphic engines require assets in a standard format such as textured meshes. The HexPlane model can be easily converted to animated meshes as follows. First, the marching cube algorithm is used to extract a simplicial mesh from the opacity field generated at each time $t$, followed by mesh decimation (for efficiency) and removal of small noisy connected components. The XATLAS (Young, 2016) algorithm is used to map the mesh vertices to a texture atlas and the texture is initialized using the HexPlane colors averaged in small spheres centred around each vertex. Finally, the texture is further optimized to better match a number of example frames rendered by HexPlane using a differentiable mesh rendered. This results in a collection of texture meshes that can be easily played back in any off-the-shelf 3D engine (see make-a-video3d.github.io for running examples).

### 4.4. Image To 4D

Given an input image, we can use it to condition our method to generate its 4D asset. Similar to (Singer et al., 2022), instead of conditioning the T2I and T2V components on the output of the prior, we condition them directly on the CLIP (Yu et al., 2022) embedding of the input image. This allows us to create a 4D asset that shares the same semantics as the input image. For this experiment we took images provided by (Nichol et al., 2022b) that were used there for the Image-to-3D task. Fig. 6 and Fig. 10 demonstrate our

method ability to generate both depth and motion from a given input image, resulting in a 4D asset.

## 5. Discussion

Creating animated 3D content is challenging as the tools available today are manual and catered to professionals. While current models can generate static 3D objects, synthesizing dynamic scenes is more complex. Unlike images and videos, where large amounts of captioned data are readily available, 4D models are scarce, with or without text descriptions.

In this work, we present MAV3D, a new approach that employs several diffusion models and dynamic NeRFs to integrate world knowledge into 3D temporal representations. MAV3D expands the functionality of previously established diffusion-based models, enabling them to generate dynamic scenes as described in text from a variety of viewpoints.

While our model is a step towards zero-shot temporal 3D generation, it also has several limitations. The conversion of dynamic NeRFs to a sequence of disjoint meshes for real-time applications is inefficient and could be improved if trajectories of vertices were predicted directly. Also, utilizing super-resolution information has improved the quality of the representation, but further improvement is needed for higher-detailed textures. Finally, the quality of the representation is dependent on the ability of the T2V model to generate videos from various views. While utilizing view-dependent prompts helps mitigating the multi-face problem, further control to the video generator would be beneficial.

## References


Adorni, G. and Di Manzo, M. Natural language input for scene generation. In *First Conference of the European Chapter of the Association for Computational Linguistics*, 1983.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

Cao, A. and Johnson, J. HexPlane: a fast representation for dynamic scenes. *arXiv preprint arXiv:2301.09632*, 2023.

Carranza, J., Theobalt, C., Magnor, M. A., and Seidel, H.-P. Free-viewpoint video of human actors. *ACM Transactions on Graphics (TOG)*, 22(3):569–577, 2003.

Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L. J., Tremblay, J., Khamis, S., et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022.

Chang, A., Savva, M., and Manning, C. D. Learning spatial knowledge for text to 3d scene generation. In *EMNLP*, 2014.

Chen, A., Xu, Z., Geiger, A., Yu, J., and Su, H. TensoRF: Tensorial radiance fields. In *ECCV*, 2022.

Chen, K., Choy, C. B., Savva, M., Chang, A. X., Funkhouser, T., and Savarese, S. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Asian conference on computer vision*, 2018.

Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., and Sullivan, S. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)*, 34(4), 2015.

Coyne, B. and Sproat, R. Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021.

Du, Y., Zhang, Y., Yu, H.-X., Tenenbaum, J. B., and Wu, J. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., and Tian, Q. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022.

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

Fridovich-Keil, S., Meanti, G., Warburg, F., Recht, B., and Kanazawa, A. K-planes: Explicit radiance fields in space, time, and appearance. *arXiv preprint arXiv:2301.10241*, 2023.

Gafni, O., Polyak, A., Ashual, O., Sheynin, S., Parikh, D., and Taigman, Y. Make-a-scene: Scene-based text-to-image generation with human priors. In *ECCV*, 2022.

Gao, C., Saraf, A., Kopf, J., and Huang, J.-B. Dynamic view synthesis from dynamic monocular video. In *ICCV*, 2021.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Jain, A., Mildenhall, B., Barron, J. T., Abbeel, P., and Poole, B. Zero-shot text-guided object generation with dream fields. In *CVPR*, 2022a.

Jain, A., Mildenhall, B., Barron, J. T., Abbeel, P., and Poole, B. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 867–876, 2022b.

Jetchev, N. Clipmatrix: Text-controlled creation of 3d textured meshes. *arXiv preprint arXiv:2109.12922*, 2021.

Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., and Lv, Z. Neural 3D video synthesis from multi-view video. In *CVPR*, 2022.

Li, Z., Niklaus, S., Snavely, N., and Wang, O. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021.

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., and Lin, T.-Y. Magic3D: High-resolution text-to-3d content creation. *arXiv preprint arXiv:2211.10440*, 2022.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *CVPR*, 2015.

Martin-Brualla, R., Radwan, N., Sajjadi, M. S., Barron, J. T., Dosovitskiy, A., and Duckworth, D. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022a.

Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M. Point-E: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022b.

Niemeyer, M., Barron, J. T., Mildenhall, B., Sajjadi, M. S., Geiger, A., and Radwan, N. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022.

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021a.

Park, K., Sinha, U., Hedman, P., Barron, J. T., Bouaziz, S., Goldman, D. B., Martin-Brualla, R., and Seitz, S. M. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics (TOG)*, 40(6), dec 2021b.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. DreamFusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.

Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *ICML*, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Sanghi, A., Chu, H., Lambourne, J. G., Wang, Y., Cheng, C.-Y., Fumero, M., and Malekshan, K. R. Clip-forge: Towards zero-shot text-to-shape generation. In *CVPR*, 2022.

Schönberger, J. L. and Frahm, J.-M. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Sheynin, S., Ashual, O., Polyak, A., Singer, U., Gafni, O., Nachmani, E., and Taigman, Y. Knn-diffusion: Image generation via large-scale retrieval. *arXiv preprint arXiv:2204.02849*, 2022.

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Smolic, A., Mueller, K., Merkle, P., Fehn, C., Kauff, P., Eisert, P., and Wiegand, T. 3d video and free viewpoint video-technologies, applications and mpeg standards. In *IEEE International Conference on Multimedia and Expo*, 2006.

Sun, C., Sun, M., and Chen, H.-T. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., and Fidler, S. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *CVPR*, 2021.

Tang, J. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion.

Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., and Theobalt, C. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.

Villegas, R., Babaeizadeh, M., Kindermans, P.-J., Moraldo, H., Zhang, H., Saffar, M. T., Castro, S., Kunze, J., and Erhan, D. Phenaki: Variable length video generation from open domain textual description, 2022. URL `https://arxiv.org/abs/2210.02399`.

Wang, C., Chai, M., He, M., Chen, D., and Liao, J. Clip-NeRF: Text-and-image driven manipulation of neural radiance fields. In *CVPR*, 2022.

Xian, W., Huang, J.-B., Kopf, J., and Kim, C. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9421–9431, 2021.

Xu, H., Ghosh, G., Huang, P.-Y., Okhonko, D., Aghajanyan, A., Metze, F., Zettlemoyer, L., and Feichtenhofer, C. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv:2109.14084*, 2021.

Young, J. Xatlas algorithm, 2016. https://github.com/jpcy/xatlas.

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., and Wu, Y. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

# A. Appendix

## A.1. Additional results

To further demonsrate the effectiveness of our model, we have incorporated an extra metric, VideoCLIP (Xu et al., 2021), to provide an additional quantitative evaluation of the temporal aspects of the generated content. The results are provided in Tab. 3.

## A.2. Ablations

**Static scene pretrainig steps.** In Fig. 9, we analyze the different number of steps required for the static scene pretraining. As observed, directly optimizing the dynamic scene leads to sub-optimal convergence (R-Precision of 63.5%). Furthermore, 2000 iterations are sufficient for achieving high-quality results (R-Precision of 83.7%). An interesting observation is that further increasing the number of static pretraining steps beyond 2000 does not improve the quality of the scene representation.

**Text-to-image model.** We trained our static scene phase using the open-source Stable Diffusion T2I model instead of our T2I model. The model achieves an R-Precision score and a VideoCLIP (Xu et al., 2021) score of 74.2% and 21.43%, respectively. These scores are slightly lower than those obtained by the same model trained with our T2I (83.7% and 23.86%) but are significantly higher than the baselines scores. We hypothesize that the difference in the scores between the method trained using our T2I and the method trained with SD is due to the shift in distributions. Since our T2I model was used to train MAV, it is more closely aligned with its distribution.

Table 3: Additional comparison with baselines (VideoCLIP).

| Model | VideoCLIP |
|---|---|
| MAV+pixelNeRF | 7.53 |
| MAV+Point-E | 2.17 |
| MAV+D-NeRF | 3.64 |
| MAV | 21.51 |
| MAV3D \z | 10.00 |
| **MAV3D** | 18.89 |

## A.3. Implementation details

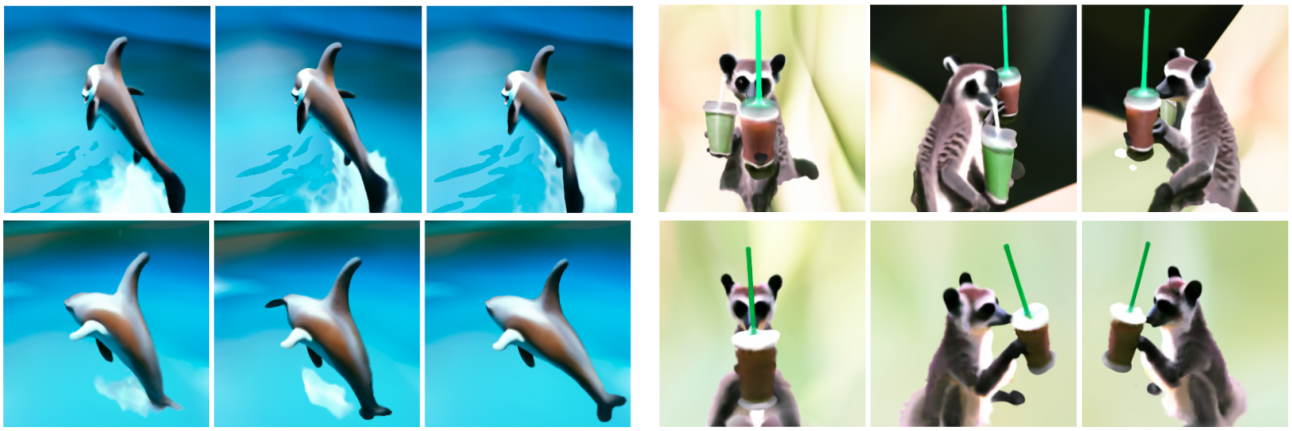
**Architecture details.** We adopt a multi-resolution grid encoding architecture, similar to (Müller et al., 2022) with 7 levels of resolutions, spanning from a minimum resolution of $16 \times 16$ up to a maximum resolution of $2048 \times 2048$. We use 5 layers MLP with 128 hidden units, each followed by ReLU activation. Similar to (Poole et al., 2022), we train another 3 layers MLP network with 64 hidden units for the background representation. The background is encoded using frequency encoder and is not conditioned on the time.

## A.4. Training details

Unless otherwise noted, we use a batch size of 8 and sample 128 points along each ray. During training, the camera position is randomly sampled in spherical coordinates, with radius in range $[1, 1.5]$, and the scene is bounded in box with radius 1.

**Dynamic Camera.** As the utilized T2V model generates videos with a moving camera $\{C_t\}_{t=1}^{T}$, we propose to bridge the distribution gap between the generated videos (by Make-A-Video) and the rendered videos (by NeRF) using a dynamic camera position. Training the model using dynamic camera trajectory simulates the movement of a real moving camera and thus enhancing the realism and coherence of the motion learned by the model.
To this end, given the first camera position $C_1 = (R, \theta, \phi)$ (randomly sampled in spherical coordinates bounded to the range $R \in [1, 1.5]$, $\theta \in [0, \frac{2\pi}{3}]$, and $\phi \in [0, 2\pi]$), we employ a random camera trajectory $\{C_1 + (i-1) \cdot \mathbf{d}\}$, in which camera $C_i$ is displaced by $(i-1) \cdot \mathbf{d}$ at each time-point $i$. Specifically, $\mathbf{d} = (\Delta R, \Delta\theta, \Delta\phi)$, where $\Delta R \sim \mathcal{U}[\frac{1-R}{F}, \frac{1.5-R}{F}]$, $\Delta\theta \sim \mathcal{U}[-\frac{\pi}{4F}, \frac{\pi}{4F}]$, $\Delta\phi \sim \mathcal{U}[-\frac{\pi}{2F}, \frac{\pi}{2F}]$.

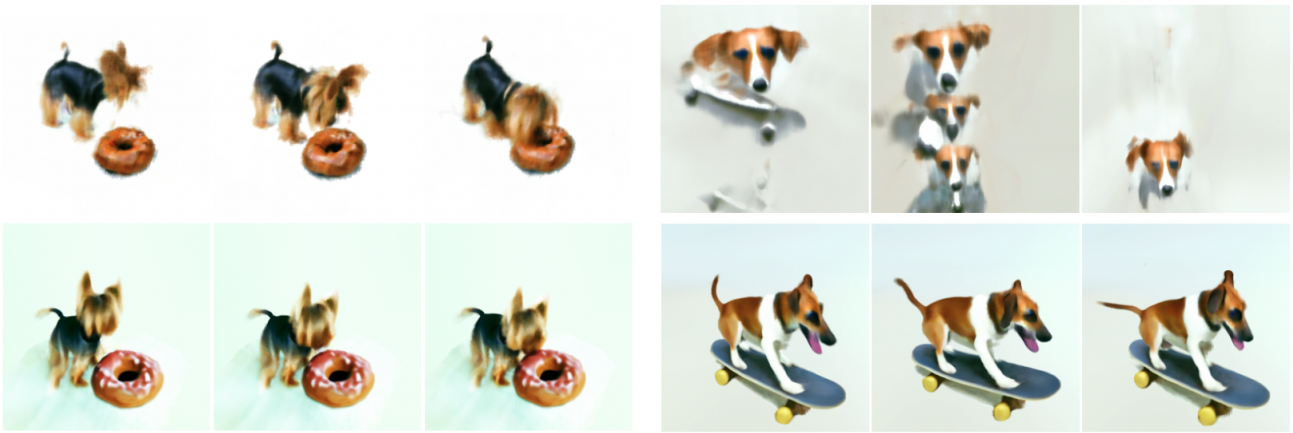
As demonstrated in the experiments, a dynamic camera also helps reducing the amount of temporal artifacts such as



A dolphin jumping out of the water

A lemur drinking boba

Figure 7: An ablation on the **dynamic camera** component. **Top:** without dynamic camera. **Bottom:** MAV3D results. The figures on the left demonstrate training with dynamic camera results in larger and more complicated motion (The dolphin manages to jump out of the water). The figures on the right demonstrate how the dynamic camera mitigates the multi-face problem.



A yorkie dog eating a donut

A dog riding a skateboard

Figure 8: An ablation on **static scene pre-training**. **Top:** without static scene pretraining. **Bottom:** MAV3D results. As can be seen in the figures on the left, pretraining the model on static scene leads to higher quality renderings. As can be seen in the figures on the right, training directly the dynamic scene (both the 3D representation and the temporal dimension) may result in lack of convergence.

unrealistic number of object parts, as it has visual of the object from multiple directions in the same sample.

**Gaussian annealing.** When optimizing a static scene model, incorporating a spatial bias towards the center of the scene can be beneficial, as it helps focusing in the center of the scene rather than directly next to the sampled cameras (Poole et al., 2022). The added noise, which is parameterized using a Gaussian PDF, causes a small "blob" of density to the origin of the scene. However, in dynamic scene rendering, objects that were originally centered at the origin may move to surrounding areas, making this bias less effective. Enlarging the standard deviation of the bias added to the density $\tau$ can encourage density not only in the center of the scene, but also in nearby locations, further enhancing the realism of the motion:

$$\lambda_\tau \cdot \exp\Big(-\frac{\|\mu\|^2}{2\sigma(ts)_\tau^2}\Big) \tag{4}$$

Where $\sigma$ is a function of the training step, $ts$. In order to anneal the bias for $M = 5000$ training steps from a minimum value $\sigma_{min} = 0.2$ to a maximum value $\sigma_{max} = 2.0$, we define a linear function as follows: $\sigma(ts) = min(\sigma_{max}, \sigma_{min} +$

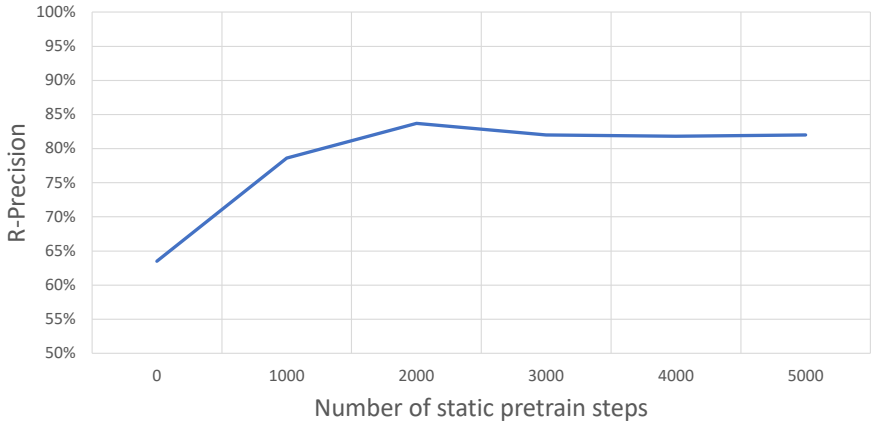



Figure 9: **Ablation on the number of static pretraining steps**

$(\sigma_{max} - \sigma_{min}) \cdot \frac{ts}{M})$.

**Optimization.** We train the model using the Adam optimizer, with cosine decay scheduler, starting from learning rate of 1e-3. The static scene representation is trained on rendered images of $64 \times 64$ for 2000 iterations with a total runtime of around 15 minutes. The dynamic stage is trained on rendered videos of $64 \times 64 \times 16$ for 5,000 iterations with a total runtime of around 3 hours. Lastly, the super resolution phase is trained on rendered videos of $256 \times 256 \times 16$ for another 2000 iterations with a total runtime of 3 hours. All runtimes were measured on 8 NVIDIA A100 GPUs. During inference, by leveraging the continuous time range, we render videos of $256 \times 256 \times 64$.

**Training objective** In order to encourage the model to make harder predictions if a specific pixel is an object or background, we add the following soft binary cross entropy regularization:

$$-\sum_{i,j} \mathrm{T}_{i,j} log(\mathrm{T}_{i,j}) + (1 - \mathrm{T}_{i,j}) log(1 - \mathrm{T}_{i,j})$$

where $\mathrm{T}_{i,j}$ denotes the accumulated density along the ray of pixel $(i, j)$, (i.e., the probability that the entire ray does not hit any particle). This regularization is added to the loss with a weight of $10^{-3}$.

The weight of the variational loss described in Sec. 3.2 is $10^{-3}$.

### A.5. Additional results for Image-to-4D

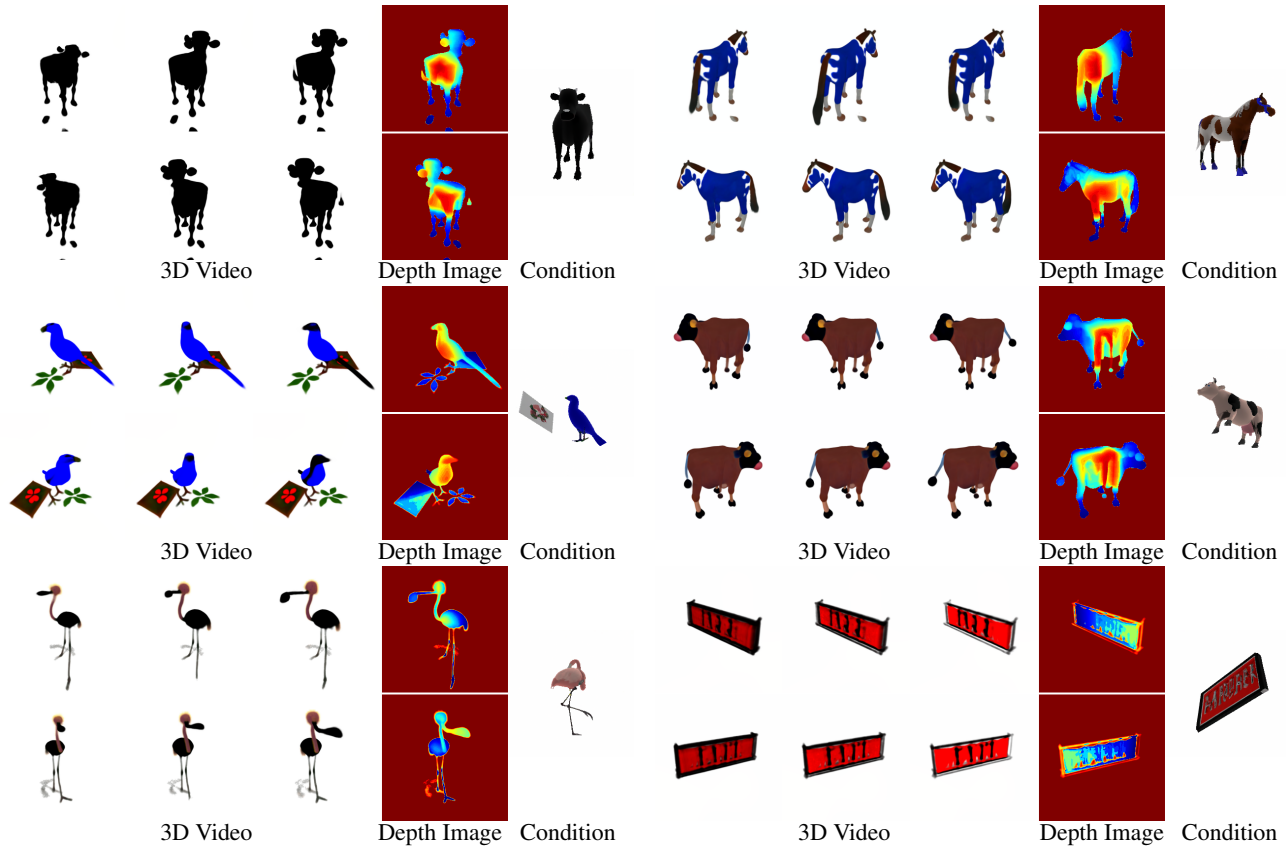



Figure 10: Additional results for the Image to 4D application.