# LIM: Large Interpolator Model for Dynamic Reconstruction

Remy Sabathier
University College London and Meta

Niloy J. Mitra
University College London

David Novotny
Meta

## Abstract

*Reconstructing dynamic assets from video data is central to many in computer vision and graphics tasks. Existing 4D reconstruction approaches are limited by category-specific models or slow optimization-based methods. Inspired by the recent Large Reconstruction Model (LRM) [15], we present the Large Interpolation Model (LIM), a transformer-based feed-forward solution, guided by a novel causal consistency loss, for interpolating implicit 3D representations across time. Given implicit 3D representations at times $t_0$ and $t_1$, LIM produces a deformed shape at any continuous time $t \in [t_0, t_1]$, delivering high-quality interpolated frames in seconds. Furthermore, LIM allows explicit mesh tracking across time, producing a consistently uv-textured mesh sequence ready for integration into existing production pipelines. We also use LIM, in conjunction with a diffusion-based multiview generator, to produce dynamic 4D reconstructions from monocular videos. We evaluate LIM on various dynamic datasets, benchmarking against image-space interpolation methods (e.g., FiLM [41]) and direct triplane linear interpolation, and demonstrate clear advantages. In summary, LIM is the first feed-forward model capable of high-speed tracked 4D asset reconstruction across diverse categories. Video results and code are available via the project page.*

## 1. Introduction

Reconstructing dynamic 4D assets from video data is a fundamental problem in computer vision and graphics, with many virtual and augmented reality applications. Existing 4D reconstructors follow two main paradigms: category-specific articulated reconstruction and image-or-text conditioned 4D distillation. Hence, they are either restricted to a specific class of objects such as humans [32] and animals [3, 44], or are optimization-based [42, 63] making them slow, requiring minutes to hours per reconstruction.

Recently, in the context of static reconstruction, the large reconstruction model (LRM) [16] has been proposed as an elegant feed-forward network that, starting from a fixed rig of multiview images, directly produces 3D implicit representation, which can then be rendered for novel view generation. In this work, in the context of dynamic reconstruction, we ask *if a similar feed-forward approach can be developed to reconstruct a tracked explicit representation across time*.

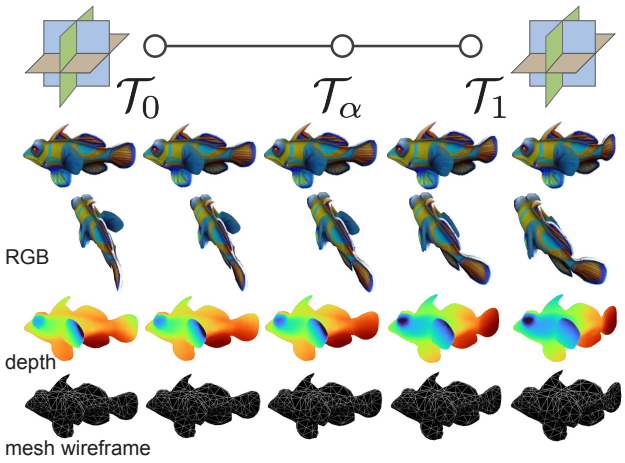


Figure 1. **Large Interpolator Model (LIM)** outputs a 4D video reconstruction by interpolating 3D implicit representations between two consecutive keyframes at times $t = 0$ and $t = 1$, which can then be used to produce 3D-consistent RGB, depth, or decoded as tracked mesh sequences.

Here, L4GM [43] proposed a feedforward 4D video reconstructor which, for each video keyframe, accepts few views of the reconstructed object and outputs a mixture of 3D Gaussian Splats [23]. However, this approach has limitations as it can only reconstruct the keyframes at their exact timesteps without the ability to interpolate the shape through time. Additionally, establishing correspondences between Gaussian mixtures from different timesteps is challenging, which complicates tracing the deformation of the underlying object geometry through time. This limitation hinders many important downstream applications, such as gaming, where we require tracked meshes in the form of the 3D shape and texture of a single mesh to be defined in a static canonical pose, with only its geometry (i.e., vertices) allowed to be deformed across time.

We thus present Large Interpolation Model (LIM) as a

transformer-based feed-forward solution that accepts an implicit representation of an object at two different keyframe times $t_0$ and $t_1$ of a video, and interpolates between the two at any continuous intermediate timestep $t \in [t_0, t_1]$. We enable this with a novel self-supervised *causal consistency loss* that allows us to meaningfully interpolate continuously in time, even when supervised with keyframes from distinct time stamps. LIM is not only an efficient interpolator, but can also track a source mesh across time producing a functional deformable 3D asset with a shared uv texture map. Here, LIM tracks the mesh by means of an additional volumetric function that maps each time-specific 3D implicit-surface point to a unique coordinate on the intrinsic (time-invariant) surface of the object. This is unique – unlike any other competing dynamic reconstructor [43], LIM outputs a mesh with time-invariant texture and topology, and time-dependent vertex deformation. This renders LIM directly applicable in existing production setups.

Our LIM module also enables dynamic reconstruction from monocular video. Specifically, given keyframes of a monocular video, a pretrained image diffusion model generates additional object views which, using a multiview LRM, we convert to keyframe-specific implicit 3D representations. Then, LIM directly interpolates the 3D representations yielding a dynamic 4D asset.

Our experiments demonstrate that LIM outperforms existing alternatives in terms of the overall quality of the implicit-shape interpolations while being several times faster. Furthermore, we also evaluate the quality of the mesh tracing, where LIM records significant performance improvements.

## 2. Related Work

**3D Reconstruction.** Early work, introduced by DreamFusion [39] optimizes a 3D scene via *score distillation sampling* from a pretrained text-to-image diffusion model [35, 40, 48]. However, these methods are slow to optimize and suffer from inconsistencies (like the Janus problem). Zero123 [45] learns to condition diffusion models on a single-view image and camera transformation, which allows novel view generation. Multiple novel views of a single object can then be used to optimize a NeRF reconstruction which, however, is often impaired by view-inconsistencies of the novel-view generator. SyncDreamer [31] proposes an extension that improves the consistency of novel views and transitivelly of the generated 3D shapes. Due to the highly-challenging task of reconstructing any 3D asset from a single image, several works [13, 14, 19–21, 25, 26, 28, 28, 29, 52, 55, 56, 60] learn 3D reconstructors of a specific category which simplifies learning of shape, deformation, and appearance priors. We also note some works on learning a generalizable dynamic radiance field from monocular videos [47, 49] which, however, are not designed for outputting a time-deforming 3D mesh. Recent methods [15, 53], trained on large 3D datasets such as *Objaverse* [9, 10], propose feed-forward reconstructors which directly predict 3D representation of an object, conditioning on a single or multiple views. These methods dramatically reduce reconstruction speed as they don't rely on any optimization loop.

**4D Representations.** Extending the popular research on representing static 3D scenes with implicit shapes, recent works proposed new time-deforming alternatives. *Dynerf* [12] extends static neural radiance field [36] with an additional compact latent code to represent time deformation. However, similar to the static implicit shape reconstructors [36, 61], its optimization process is relatively slow. [4, 8, 11] factorize a dynamic representation into multiple low-rank components, which dramatically speeds up the optimization. Notably, *Hexplane* [4] proposes a 6-plane representation which extends the spacial triplane representation [7] to a spatio-temporal one. With the emergence of *3D Gaussian Splatting* [22] (3DGS), [34, 54] propose its extension to dynamic scenes, relying either on a per-frame optimization with dynamic constraints, or on a temporal network to deform the gaussians in time.

**4D Generation and Reconstruction.** Several works focus on text-to-4D generation: MAV3D [46] optimizes a Hexplane [4] representation via score distillation sampling from a text-to-image and a text-to-video diffusion model. 4Dify [2] introduces a 3D-aware text-to-image diffusion model, and parameterizes the representation with a multi-resolution hash encoding [37]. However, these methods tend to produce very limited and simple motions. TC4D [1] proposes an extension to decompose movement into local deformation and global rigid motion. [30] applies same SDS supervision with Gaussian splatting.

Similar to us, recent work focused on video-to-4D reconstruction. Consistent4D [18] generates 4D content from monocular video via SDS supervision, optimizing a Cascade DyNerf [12]. Simiarly, 4DGen [62] and DreamGaussian4D [42] encode the 4D asset as a set of static 3D gaussians and a regularized deformation field. [38, 64] leverages diffusion models to generate frames across views and timestamps, and optimizes a dynamic gaussian splats based on these frames. [17, 57] decompose motion and appearance in gaussian splatting: instead of predicting a deformation for each gaussian in a canonical frame, they deform Gaussians by means of sparse control points. All above methods are relatively slow due to the 2nd reconstruction stage that optimizes each 4D asset from scratch. Furthermore, these methods cannot easily trace the resulting 4D asset through time, which prohibits their application in production setups.

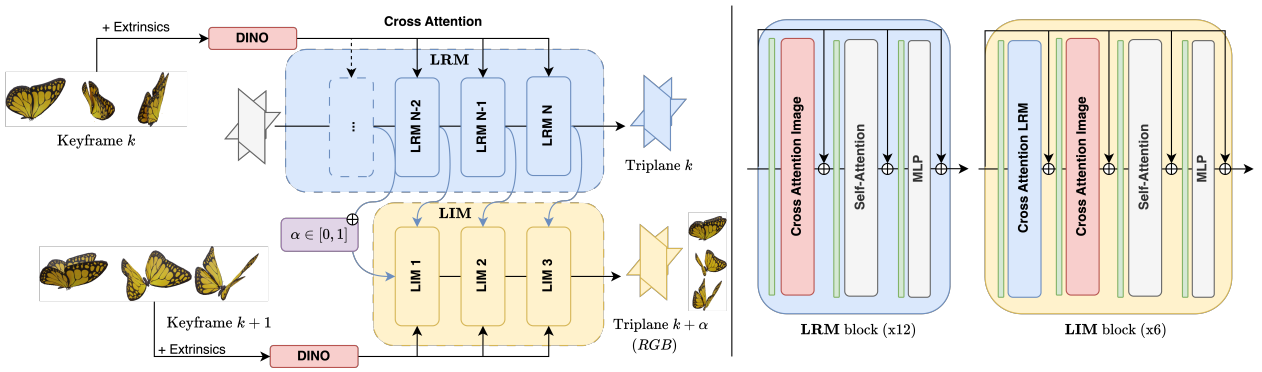


Figure 2. **LIM framework**. (Left) Given multi-view images on 2 timesteps $k$ and $k+1$, LIM interpolates any intermediate 3D representation at $k+\alpha, \alpha \in [0,1]$. It achieves this notably via cross-attention with the latest intermediate features of LRM on keyframe $k$. In practice, our LIM architecture has 6 blocks and LRM 12 blocks. (Right) Block structure of LRM and LIM. We include layer normalization before each module in blocks.

## 3. Method

In Sec. 3.1, we review the LRM [15] architecture which our method is based on; in Sec. 3.2, we introduce LIM, our large interpolator model, for efficient 3D interpolation and; in Sec. 3.4, we show how LIM can be used for fast 4D reconstruction and mesh tracking.

### 3.1. Preliminaries

Our Large Interpolation Model (LIM) is built upon the multi-view version of Large Reconstruction Model (LRM) [15]. We first review LRM and its multi-view version.

**LRM.** The LRM [15] is a single-view reconstructor. Given a source image $I_{\text{src}}$ and its camera $\pi_{\text{src}}$, LRM reconstructs a triplane [7] representation $\mathcal{T} := \text{LRM}_\theta(I, \pi)$ of the depicted scene. The triplane may be rendered from any target view $\pi_{\text{tgt}}$ using Emission-Absorption raymarching yielding an RGB render $R(\pi_{\text{tgt}}, \mathcal{T})$, depth render $R(\pi_{\text{tgt}}, \mathcal{T})_D$ and alpha-mask render $R(\pi_{\text{tgt}}, \mathcal{T})_\alpha$. In practice, we use the Lightplane renderer [5] to implement $R$.

In a single-view setting, 3D reconstruction is highly ambiguous. Indeed, as depicted in Fig. 3, when applied to reconstruct monocular-video frames, LRM outputs triplanes with significantly time-inconsistent shape and texture.

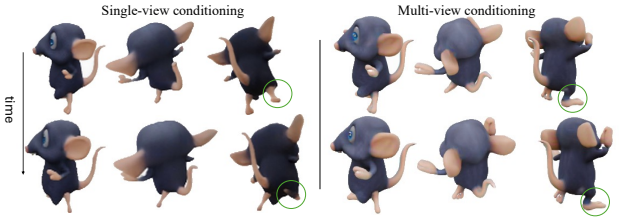


Figure 3. LRM conditioned on a single-view [50] is sensitive to small changes on the input image, which gives inconsistent result from one video frame to another. The multi-view LRM prevents this instability. For each model, left shows an input-view, right shows two target views. Each line is a different timestep.

**Multi-view LRM setup.** Hence, in order to minimize reconstruction ambiguity, we leverage a few-view conditioned version of LRM. Formally, given a set $\mathcal{I}^{\text{src}} := \{I^i_{\text{src}}\}_{i=1}^{N^{\text{src}}}$ of $N^{\text{src}}$ source images with corresponding cameras $\Pi^{\text{src}} := \{\pi^i_{\text{src}}\}_{i=1}^{N^{\text{src}}}$ we predict a triplane $\mathcal{T} := \text{LRM}_\theta(\mathcal{I}^{\text{src}}, \Pi^{\text{src}})$, where we overload the same symbol for the multi-view and single-view versions for compactness. The architecture follows [27, 59] – the pixels of each source image $I^{\text{src}}$ are first concatenated with the Plucker ray coordinates encoding the corresponding camera pose $\pi^{\text{src}}$ and then fed to DinoV2 [6] yielding image tokens. Then, these tokens enter cross-attention layers inside a large 12-layer transformer that refines a set of fixed shape tokens into the final triplane representation $\mathcal{T}$ of the reconstructed scene.

**Multi-view LRM training.** We train LRM in a fully-supervised manner on a large dataset of artist-created meshes, similar to Objaverse [10]. We render each mesh from a set of pre-defined camera viewpoints $\Pi$. The latter rendering, besides the RGB image $I$, also provides the ground-truth depth map $D$ and the alpha mask $M$. For each training scene, we sample $N_{\text{src}} = 4$ random images as input views, and render into $N_{\text{tgt}} = 4$ randomly sampled held-out target views where losses are optimized.

We optimize three losses. (i) The photometric loss $\mathcal{L}_{\text{photo}} := \sum_{i=1}^{N_{\text{tgt}}} \|I^i - R(\pi^i, \mathcal{T})\|^2 + \text{LPIPS}(I^i, R(\pi^i, \mathcal{T}))$; (ii) mask loss $\mathcal{L}_{\text{mask}} := \sum_{i=1}^{N_{\text{tgt}}} \text{BCE}(M^i, R(\pi^i, \mathcal{T})_\alpha)$, where BCE is binary cross-entropy; and (iii) depth loss $\mathcal{L}_{\text{depth}} := \sum_{i=1}^{N_{\text{tgt}}} \|D^i - R(\pi^i, \mathcal{T})_D\|$. Recall that $\mathcal{T} := \text{LRM}_\theta(\mathcal{I}^{\text{src}}, \Pi^{\text{src}})$ is the triplane output by LRM given the 4 source views. The total loss $\mathcal{L}_{\text{photo}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{mask}}$ is minimized with the Adam optimizer [24] with a learning rate of $10^{-4}$ until convergence.

### 3.2. LIM: Large Interpolator Model

Given a monocular video, our aim is to predict the 3D representation of the scene at any continuous timestep. Furthermore, we aim to achieve this in a feed-forward manner, and we require the ability to trace the 3D representation in time, which eventually yields a practically applicable animated mesh with a shared UV texture.

**Multi-view** LIM**.** As mentioned in Sec. 3.1, reconstructing monocular videos is a highly ambiguous task and, hence, we first focus on the simpler multi-view version with access to multiple views at each timestep. At the end of this section, we describe how to tackle the harder monocular task by converting it to the multi-view setting described here.

Formally, we are given a multi-view RGB video $\{\mathcal{I}_k\}_{k\in(1,2,\dots,N_f)}$ composed of $N_f$ timesteps where, for each integer timestep $k$, we have a set $\mathcal{I}_k = \{I_k^i\}_{i=1}^{N_v}$ of $N_v$ view-points with cameras $\Pi_k = \{\pi_k^i\}_{i=1}^{N_v}$. In order to 4D-reconstruct the latter we can, in principle, use LRM to predict a set $\{\mathcal{T}_k\}_{k\in(1,2,\dots,N_f)}$ containing a triplane for each keyframe in the video. However, the latter remains **discrete** in time and, hence, we cannot obtain a 3D representation at any intermediate continuous timestep $k+\alpha, \alpha \in [0,1]$. Furthermore, such frame-specific triplanes encode implicit shapes disconnected across different timesteps. This prevents us from converting the time-series of reconstructions into a time-varying mesh.

Thus, to achieve continuous reconstruction in time, and to enable surface tracking, we introduce our Large Interpolator Model (LIM). Given 2 keyframe sets $\mathcal{I}_k, \mathcal{I}_{k+1}$ at discrete timesteps $k$ and $k+1$, LIM predicts an interpolated triplane $\hat{\mathcal{T}}_{k+\alpha}$ at any continuous timestep $t = k+\alpha, \alpha \in [0,1]$:

$$\hat{\mathcal{T}}_{k+\alpha} := \text{LIM}_\psi(\mathcal{F}_k(\mathcal{I}_k, \Pi_k), \mathcal{I}_{k+1}, \alpha). \tag{1}$$

The architecture of LIM, illustrated in Fig. 2, takes advantage of the pretrained multiview LRM model from Sec. 3.1. More specifically, we begin by calculating the intermediate features $\mathcal{F}_k$ as predicted by LRM from the frame set $\mathcal{I}_k$ at the start timestep $k$. These features are extracted after each of the last $L = 6$ transformer blocks of LRM. Then, we broadcast and concatenate a positional encoding of the interpolation time $\alpha$ to $\mathcal{F}_k$ and feed the result to LIM. This input is then refined by series of cross-attentions with the image tokens of the next keyframes $\mathcal{I}_{k+1}$ to predict the final interpolated triplane $\hat{\mathcal{T}}_{k+\alpha}$.

### 3.3. Training LIM

We train LIM on a large dataset of artist-created meshes animated with a range of motions. For each scene, we render the asset from several random viewpoints at each key-frame of the animation.

In order to train LIM, for each scene, we first sample a pair of keyframe interpolation endpoints at timesteps $k_{\text{src}}$ and $k_{\text{tgt}}$ such that $k_{\text{tgt}} - k_{\text{src}} \in \{2,3,4\}$. Then, we additionally sample a middle keyframe $k_m$ such that $k_{\text{src}} \leq k_m \leq k_{\text{tgt}}$. We then task LIM to predict the interpolated triplane $\hat{\mathcal{T}}_{k_{\text{src}}+\alpha_m} := \text{LIM}(\mathcal{F}_{k_{\text{src}}}, \mathcal{I}_{k_{\text{tgt}}}, \alpha_m)$ at an intermediate keyframe $k_m$ given the source and target conditioning $\mathcal{F}_{k_{\text{src}}}, \mathcal{I}_{k_{\text{tgt}}}$ and the interpolation time $\alpha_m = \frac{k_m - k_{\text{src}}}{k_{\text{tgt}} - k_{\text{src}}}$, which converts the discrete timestep $k_m$ into a continuous interpolation time $\alpha_m \in [0,1]$. The interpolated triplane $\hat{\mathcal{T}}_{k_{\text{src}}+\alpha_m}$ is then compared to the pseudo-ground-truth triplane $\mathcal{T}_{k_m} = \text{LRM}(\mathcal{I}_{k_m}, \Pi_{k_m})$ output by LRM at the interpolated keyframe $k_m$ with the following MSE loss:

$$\mathcal{L}_\mathcal{T} := \|\hat{\mathcal{T}}_{k_{\text{src}}+\alpha_m} - \mathcal{T}_{k_m}\|^2, \alpha_m = \frac{k_m - k_{\text{src}}}{k_{\text{tgt}} - k_{\text{src}}}. \tag{2}$$

**Causal consistency for continuous-time interpolation.** The loss $\mathcal{L}_\mathcal{T}$ provides a basic supervisory signal which, however, only supervises LIM at keyframe times $k_m$ that are discrete. The latter prevents the model from becoming a truly temporally-smooth interpolator because, during training, it is never exposed to arbirary interpolation times $\alpha$ spanning the whole continuous range of $[0,1]$.

To address this, we introduce a causal consistency loss $\mathcal{L}_{\text{causal}}$. In a nutshell, the loss enforces that a triplane interpolated directly from time $k_{\text{src}}$ to $k_{\text{src}} + \delta, \delta \in [0,1]$ has to match a triplane that is first interpolated to an arbitrary intermediate timestep $k_{\text{src}} + \alpha_{\text{rand}}, \alpha_{\text{rand}} \in \mathcal{U}(0,\delta)$ and then further interpolated to the target timestep $k_{\text{src}} + \delta$.

More formally, we define the causal consistency loss as:

$$\mathcal{L}_{\text{causal}} := \left\| \text{LIM}\left(\hat{\mathcal{F}}_{k_{\text{src}}+\alpha_{\text{rand}}}, \mathcal{I}_{k_{\text{src}}+\delta}, \frac{\delta - \alpha_{\text{rand}}}{1 - \alpha_{\text{rand}}}\right) - \hat{\mathcal{T}}_{k_{\text{src}}+\delta} \right\|^2, \tag{3}$$

where $\hat{\mathcal{F}}_{k_{\text{src}}+\alpha_{\text{rand}}}$ stands for the intermediate features predicted by LIM when interpolating from $k_{\text{src}}$ to $k_{\text{src}} + \alpha_{\text{rand}}$. Note that we feed into the second LIM pass the intermediate features $\hat{\mathcal{F}}_{k_{\text{src}}+\alpha_{\text{rand}}}$ output by LIM as opposed to features $\mathcal{F}$ output by LRM as prescribed by the original LIM formulation in (1). We empirically observed that this works as we can assume that the intermediate features of LIM follow approximately the same distribution as the intermediate features of LRM. Hence, LIM can, in a recurrent manner, accept its own intermediate features to ground the interpolation of the next timesteps. As demonstrated in Sec. 4.1 (Tab. 4), the causal loss $\mathcal{L}_{\text{causal}}$ significantly improves the temporal consistency and the quality of the interpolations.

Note that for LIM training, LRM model weights $\theta$ are already optimized and we keep them frozen. We minimize the total loss $\mathcal{L}_\mathcal{T} + \mathcal{L}_{\text{causal}}$ using the Adam optimizer [24] with a learning rate of $10^{-4}$ until convergence.

**Monocular 4D reconstruction.** Given the trained LIM and LRM models, we can now predict the 3D scene representation at any continuous timestep. However, as men-

tioned above, the LIM model relies on multiple views at each timestep, which are not always available in practice.

Our method can, however, also be used in the monocular-video to 4D setting. Here, we leverage a pretrained diffusion model [58] to recover 3 videos at different viewpoints, consistent in shape and motion with the monocular source. We then reconstruct 3D per timestep with LRM, and add in-between timesteps (depending on the user frame-rate need), by interpolating with LIM. This replaces the optimization of a 4D representation from the multi-view videos [2, 46], which in practice takes minutes to hours for a single scene.

### 3.4. Tracing shapes with LIM

In Secs. 3.2 and 3.3, we have described how LIM, together with LRM, can be trained to predict a continuous-time 3D representation of a scene given a set of multi-view images at discrete timesteps. As mentioned before, a key goal of our model is to also trace the deformable shape through time. Next, we describe how to extend LIM and LRM to output canonical surface coordinates to enable surface tracing, and how to use these coordinates to output a time-deforming mesh with fixed topology and texture.

**Interpolating canonical surface coordinates.** To simplify the surface tracing task, we extend LIM and LRM to label the interpolated implicit surface with intrinsic coordinates defined in the canonical coordinate of the object to be interpolated. More specifically, aside from tasking the interpolated triplanes $\mathcal{T}$ with representing the RGB color and geometry of the implicit shape, we also task them with supporting a volumetric function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ that maps each point in the 3D space to its canonical surface coordinate. Without loss of generality, we set the canonical coordinates of time-deforming shape to the XYZ coordinates of the corresponding surface points in the start timestep $k_{\text{src}}$. Since we have a dataset of artist-created meshes with a known deformation of each vertex in time, we can easily calculate the canonical coordinate function $f_{k_{\text{src}}}$ of each vertex at the start timestep $k_{\text{src}}$ and then transport those using the known animation to any other deformation time $k_{\text{tgt}}$ yielding a function $f_{k_{\text{tgt}}}$. Importantly, $f$ can be rendered at any point in time from a viewpoint $\pi^i$ yielding a 3-channel canonical coordinate render $C^i$.

Given the above, we train a second LRM, dubbed $\overline{\text{LRM}}$, which shares the same architecture, but predicts triplanes $\mathcal{T}^C := \overline{\text{LRM}}(\mathcal{C}_{\text{src}}, \Pi_{\text{src}})$ supporting the coordinate function $f$, by accepting a set $\mathcal{C}_{\text{src}} := \{C^i_{\text{src}}\}_{i=1}^{N_{\text{src}}}$ of multi-view source canonical renders $C^i_{\text{src}}$. This canonical-coordinate $\overline{\text{LRM}}$ is supervised with the following canonical loss:

$$\mathcal{L}_{\text{can}} := \|C^i_{\text{src}} - R(\pi^i, \mathcal{T}^C_{\text{src}})\|^2, \tag{4}$$

and with the same depth, and mask losses as the LRM in Sec. 3.1. Similarly, we train $\overline{\text{LIM}}$ to interpolate the canonical-coordinate triplane as,

$$\hat{\mathcal{T}}^C_{\text{src}} := \overline{\text{LIM}}(\mathcal{F}^C_{k_{\text{src}}}, \mathcal{I}_{k_{\text{src}}}, \mathcal{I}_{k_{\text{tgt}}}, \alpha). \tag{5}$$

Note that this $\overline{\text{LIM}}$, analogous to LIM, is conditioned on the features $\mathcal{F}^C_{k_{\text{src}}}$ of the canonical-coordinate $\overline{\text{LRM}}$. However, differently from LIM, $\overline{\text{LIM}}$ accepts target and source RGB frames $\mathcal{I}_{k_{\text{src}}}$ and $\mathcal{I}_{k_{\text{tgt}}}$ instead of the target-time canonical image $\mathcal{C}_{k_{\text{tgt}}}$. This is because the canonical coordinates can only be carried forward in time and, as such, are not available for the target timestep $k_{\text{tgt}}$. Hence, $\overline{\text{LIM}}$ instead learns how to propagate the coordinates by analyzing the RGB frames that are available in both timesteps. We supervise $\overline{\text{LIM}}$ with the MSE loss $\mathcal{L}^C_{\mathcal{T}}$ and the causal consistency loss $\mathcal{L}^C_{\text{causal}}$ that are defined analogously to (2) and (3) but with the canonical-coordinate triplanes $\mathcal{T}^C$ and images $\mathcal{C}$.

**Mesh tracing.** Given the RGB and canonical-coordinate versions of LIM and LRM, we can trace a mesh multi-view frames $\mathcal{I}_{\text{src}}$ and $\mathcal{I}_{\text{tgt}}$ (recall that, using an image diffusion model, this is also possible for a monocular video).

We start by extracting the color triplane $\mathcal{T}_{k_{\text{src}}}$ at timestep $k_{\text{src}}$ with LRM. We then render $\mathcal{T}_{k_{\text{src}}}$ to obtain a depth map $D_{k_{\text{src}}}$ that we unproject to form 3D points yielding the multi-view canonical coordinates $\mathcal{C}_{k_{\text{src}}}$. Given $\mathcal{C}_{k_{\text{src}}}$, we can predict the canonical-coordinate triplane $\mathcal{T}^C_{k_{\text{src}}}$ with $\overline{\text{LRM}}$.

Then, for a series of monotonic time offsets $\alpha_0, \ldots, \alpha_N; |\alpha_{j+1} - \alpha_j| \rightarrow 0$ the canonical coordinate triplane $\mathcal{T}^C_{k_{\text{src}}}$, together with $\mathcal{I}_{\text{src}}$ and $\mathcal{I}_{\text{tgt}}$, is fed to $\overline{\text{LIM}}$ to interpolate the canonical-coordinate triplane $\hat{\mathcal{T}}^C_{k_{\text{src}}+\alpha_j}$ at all continuous timesteps $k_{\text{src}} + \alpha_j$.

The series of resulting canonical-coordinate triplanes $\hat{\mathcal{T}}^C_{k_{\text{src}}+\alpha_j}$ provides a series of implicit shapes annotated with surface coordinates. To obtain a time deforming mesh, we first run Marching Cubes (MC) [33] on the first triplane $\hat{\mathcal{T}}^C_{k_{\text{src}}+\alpha_0}$ resulting in a mesh $\mathcal{M}_{k_{\text{src}}+\alpha_0}(V_{k_{\text{src}}+\alpha_0}, F)$ with time-dependent vertices $V_{k_{\text{src}}+\alpha}$ and time-invariant faces $F$. We then run MC on the next triplane $\hat{\mathcal{T}}^C_{k_{\text{src}}+\alpha_1}$ and match the vertices of the previous mesh to the surface on the next mesh using nearest neighbor search in the space of canonical coordinates defined by the triplanes $\hat{\mathcal{T}}^C_{k_{\text{src}}+\alpha_0}$ and $\hat{\mathcal{T}}^C_{k_{\text{src}}+\alpha_1}$, respectively. Afterwards, we replace the vertices $V_{k_{\text{src}}+\alpha_0}$ with the corresponding nearest neighbors from the next time $k_{\text{src}} + \alpha_1$, and repeat the process for all the remaining timesteps $\alpha_2, \ldots, \alpha_N$

## 4. Experiments

### 4.1. Feed-forward Triplane Interpolation

In this section, we evaluate the ability of LIM to interpolate triplanes so that the renders of the latter match the ground-truth views extracted at the target interpolation timestep. More specifically, given a multi-view video of a deforming object, which contains the frame set $\{\mathcal{I}_k\}_{k=1}^{N_f}$ at each

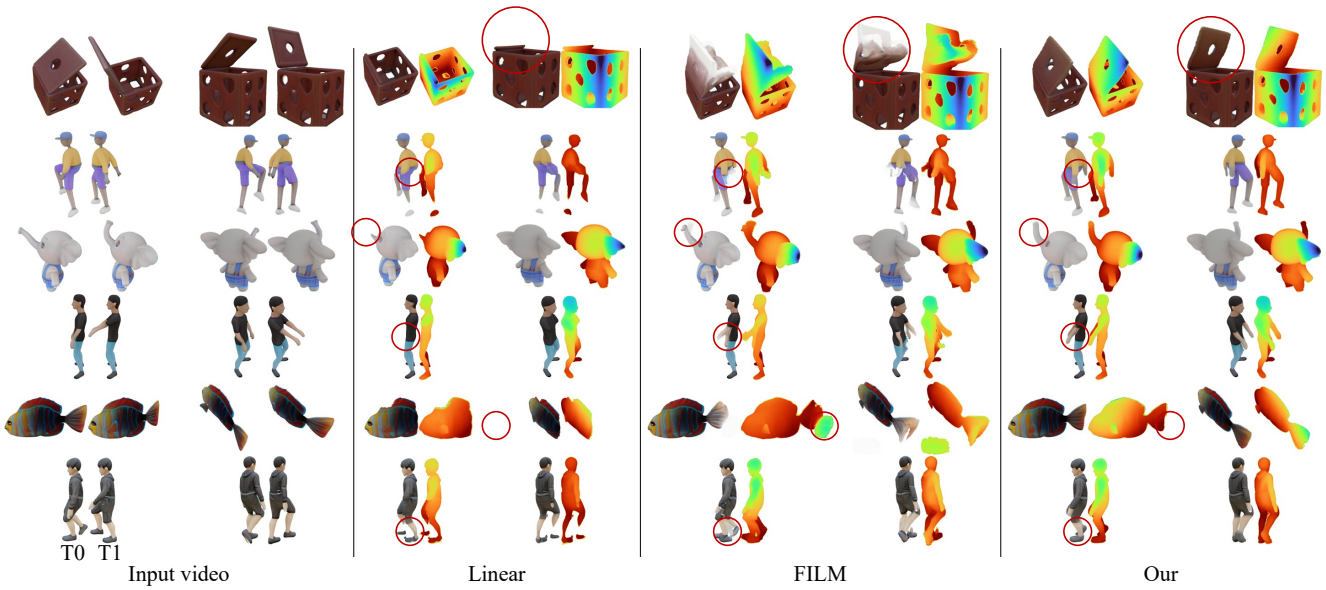


Figure 4. **Interpolation results** comparing (i) linear interpolation in triplane space, which fails on dynamic parts; (ii) image-based interpolator [41] (FILM), yielding view-consistent frame interpolations leading to defective reconstructions (ghosting around dynamic parts; for example, the tip of the elephant's trunk or fish's tail); and (iii) our LIM-based interpolation, which yields the most plausible results.

timestep $k$, we first split the frame sets into adjacent triplets $\mathcal{I}_k, \mathcal{I}_{k+1}, \mathcal{I}_{k+2}$ for a $k$-th triplet. In each triplet, we then evaluate the ability of a method to interpolate the 3D representation $\hat{\mathcal{T}}_{k+1}$ at the mid-point $k+1$ given the boundary frames $\mathcal{I}_k, \mathcal{I}_{k+2}$. Note that, since the frames are sampled at uniform time intervals, the continuous interpolation index is $\alpha = 0.5$. For evaluation, we select 256 scenes from our heldout dataset of animated objects.

Given the interpolant $\hat{\mathcal{T}}_{k+1}$, we then evaluate its quality by rendering into the set of (novel) evaluation views, and reporting three photometric errors measuring the discrepancy between the renders and the corresponding ground-truth images: (i) peak signal-to-noise ratio **PSNR**; (ii) perceptual loss **LPIPS** [65]; and (iii) **PSNR$_{\text{FG}}$** calculating PSNR only over the foreground pixels.

**Baselines.** We compare our LIM interpolation with two baselines. The first baseline (Linear) is a simple linear interpolation, which defines the interpolated triplane $\hat{\mathcal{T}}^{\text{linear}}_{k+1} = (1-\alpha)\mathcal{T}_k + \alpha\mathcal{T}_{k+2}$ as a linear combination of the two triplanes predicted by LRM for each set of boundary frames. The second baseline (FILM) is image-based. Specifically, we first interpolate in the image-space using a pre-trained deep image interpolator *FILM* [41] which, given the boundary views $I^i_k, I^i_{k+2}$, generates the interpolant $\hat{I}^i_{k+1}$ followed by multi-view LRM reconstruction yielding the interpolated triplane $\hat{\mathcal{T}}^{\text{FILM}}_{k+1}$. We also report results from an Oracle method, which has access to the ground-truth images $\mathcal{I}_{k+1}$ and reconstructs the corresponding triplane $\hat{\mathcal{T}}^{\text{Oracle}}_{k+1}$ with LRM. The latter provides an upper performance limit.

Table 1. **Interpolation results** comparing LIM to a linear triplane interpolation, and to an image-based interpolation implemented with the FiLM image interpolator [41].

| | PSNR ↑ | PSNR$_{\text{FG}}$ ↑ | LPIPS ↓ |
|---|---|---|---|
| Linear | 20.96 | 11.04 | 0.093 |
| FILM [41] | 22.05 | 14.98 | 0.082 |
| LIM (Our) | 23.11 | 16.12 | 0.075 |
| Oracle | 24.43 | 17.51 | 0.064 |

**Results.** Tab. 1 presents the results. We also provide Fig. 4 and a supplemental video for visual evaluation. LIM outperforms linear interpolation and image-based interpolation on all three metrics. Here, we notice that linear interpolation in the triplane space fails to correctly represent dynamic elements, which often disappear after being interpolated. Furthermore, the image-based interpolation often results in artifacts in the color and opacity fields, which is due to view-inconsistencies between the images interpolated at the same timestep. LIM scores closest to the Oracle bound.

**Ablating causal consistency.** Table 2 reports the performance of a LIM trained without the causal consistency loss $\mathcal{L}_{\text{causal}}$ on the aforementioned benchmark. The evaluation reveals a significant drop in performance, confirming the merit of the causal loss. See supplementary material.

### 4.2. Deformable Mesh Reconstruction

In this section, we evaluate the quality of the dynamic mesh reconstructions output by LIM's mesh-tracing method

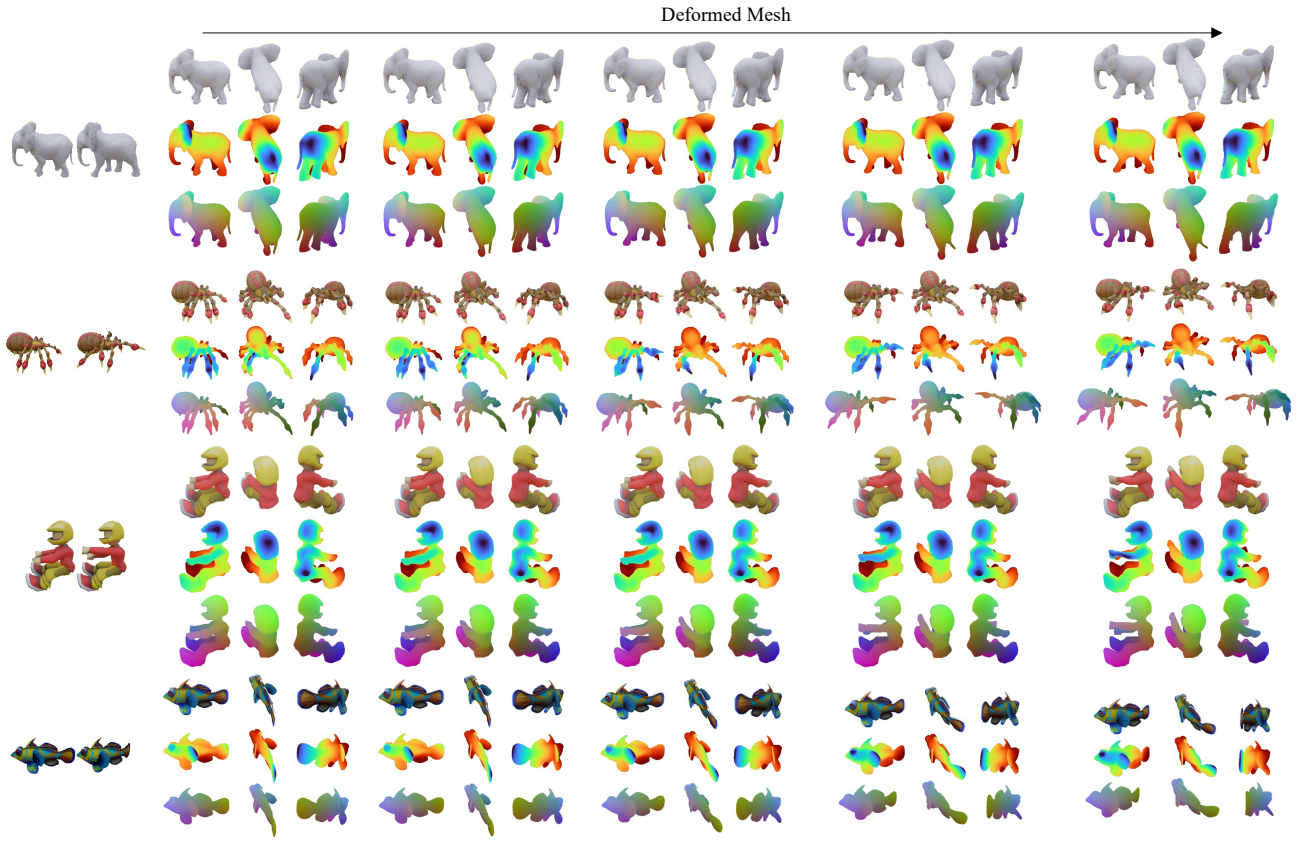


Figure 5. **Mesh Tracking results.** Given two implicit 3D representations, LIM can interpolate densely in time and hence can track a source mesh to produce a deforming mesh sequence. For each scene, we show (top to bottom) RGB rendering of the tracked mesh, depth and canonical-coordinate interpolation. See supplemental video on the project page.

Table 2. **Ablating $\mathcal{L}_{\text{causal}}$.** Interpolation accuracy comparing our LIM with its ablation removing the causal consistency loss $\mathcal{L}_{\text{causal}}$.

| | PSNR ↑ | $\text{PSNR}_{\text{FG}}$ ↑ | LPIPS ↓ |
|---|---|---|---|
| LIM- wo/ $\mathcal{L}_{\text{causal}}$ | 22.2 | 15.38 | 0.084 |
| LIM | **23.11** | **16.12** | **0.075** |

(Sec. 3.4). More specifically, we create a dataset of 8-step test sequences heldout from the train set. As before, the $k$-th timestep of the 8 timesteps contains a frame set $\mathcal{I}_k$. First, we reconstruct a mesh in the canonical pose, defined as the shape at the first frame ($k = 1$). We then use our mesh-tracing method to deform the vertices of the mesh so they follow the motion observed in the frame sets $\mathcal{I}_k, k \in [2, \ldots, 8]$. Note that we keep the topology of the first-frame mesh, as well as its texture shared across all 8 timesteps. After the mesh is traced, we render it at timesteps $\{4, 6, 8\}$ to several heldout views and again evaluate the PSNR, LPIPS, and $\text{PSNR}_{\text{FG}}$. We compare our LIM mesh tracing, described in Sec. 3.4, with a baseline (Nearest Matching) that iteratively deform the vertices of the mesh at timestep $i$ to the nearest match on the surface of the mesh at timestep $i + 1$, where the matching is performed over distance and RGB-features.

Table 3. **Evaluation of deformable mesh tracking** comparing our LIM with Nearest-Neighbor tracing

| | PSNR ↑ | $\text{PSNR}_{\text{FG}}$ ↑ | LPIPS ↓ |
|---|---|---|---|
| NN-tracing | 20.33 | 16.09 | 0.122 |
| LIM (Our) | **21.56** | **17.11** | **0.096** |

**Results.** Tab. 3 present the quantitative results. Our main observation is that LIM's ability to densely and accurately interpolate RGB and XYZ values over time, allows one to avoid explicitly solving the challenging correspondence problem between keyframed poses, separated by non-trivial deformations. Our evaluations show that LIM works well

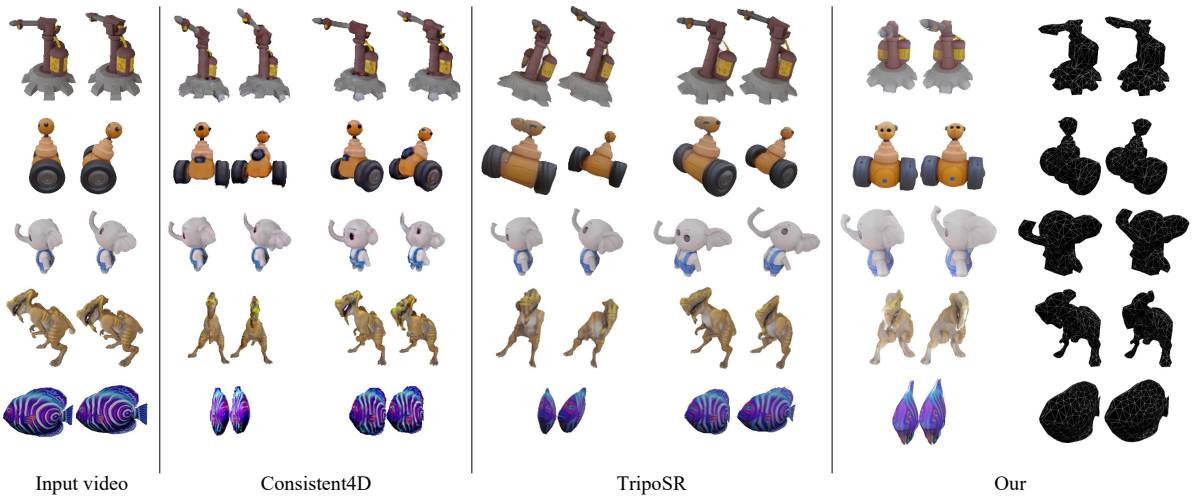


Figure 6. **Monocular 4D Reconstruction** comparing LIM with Consistent4D and TripoSR applied to each input frame separately. Our method is the only one to output a time-deforming mesh with fixed topology and texture. For our method, we render the topology for the second view.

overall, although the results degrade around thin structures. Additional results are available at the project page.

### 4.3. 4D Reconstruction

Finally, we evaluate LIM on 4D reconstruction from a monocular video.

For each evaluation scene, we extract a video sequence of 16 frames $\{I_k^1\}_{k=1}^{16}$. We then leverage an external diffusion-based model to generate 3 additional views of the scene $\{\hat{I}_k^i\}_{k\in[1,\dots,16],i\in\{2,3,4\}}$ for each timestep. We then reconstruct the 3D representation on the odd frames with LRM and interpolate with our LIM to predict the representation on the even frames. We evaluate on all the even frames, on a set of four random views, outside the training views. We report two metrics: the LPIPS error between the ground-truth images and the renders of the reconstructed triplanes, and the FVD [51] measuring comparing generated new-view sequence against ground-truth temporal sequence of renders.

**Baselines.** We compare two baselines: (i) Consistent4D [18], an optimization-based model, conditioned on monocular video and supervised via SDS loss; (ii) TripoSR [50], an open-source LRM conditioned on a single image, *i.e.*, the model is at a disadvantage.

**Results.** Evaluation results are in Tab. 4. The combination of LIM with multi-view diffusion model outperforms the competing methods by a significant margin.

## 5. Conclusion

We have proposed LIM, a novel method paired with multi-view LRM to enable continuous and feed-forward rendering in both space and time. As opposed to image-based interpolators or direct triplane baselines, we demonstrated that LIM results in high-quality and consistent 4D interpolations, realistically capturing deformations, and can support different type of modalities. A key advantage of ours is that we can interpolate in RGB and (canonical) XYZ, which in turn allows to directly output consistently-textured dynamic mesh assets that applicable in production workflows.

A key limitation of our approach is it being trained on synthetic data. Since LIM learns to interpolate deformation/motion, rather than appearance, we expect the results to carry over to real data. However, we would need an LRM model trained on real-world data to test this hypothesis. We leave this for later exploration. Also, in the future, we would like to extend our framework to handle extrapolation, instead of interpolation. The challenge would be to effectively use video data and video generators for training.

Table 4. **Monocular video reconstruction** results comparing our LIM to Consistent4D [18] and to TripoSR [50] applied independently to each frame of the input video.

| | Feed-fwd. | Inf. Time | LPIPS ↓ | FVD ↓ |
|---|---|---|---|---|
| Consistent4D [18] | ✖ | ∼1.5hours | 0.429 | 1136.3 |
| TripoSR [50] | ✔ | ∼30secs | 0.504 | 1427.2 |
| LIM (Ours) | ✔ | ∼3min | **0.142** | **811.1** |

## Acknowledgment

The work was partially supported by the UCL AI Centre and a UCL-Meta PhD support. We thank Tom Monnier for his valuable guidance during the rebuttal stage.

## References


[1] Sherwin Bahmani, Xian Liu, Yifan Wang, Ivan Skorokhodov, Victor Rong, Ziwei Liu, Xihui Liu, Jeong Joon Park, Sergey Tulyakov, Gordon Wetzstein, Andrea Tagliasacchi, and David B. Lindell. TC4d: Trajectory-conditioned text-to-4d generation, . 2

[2] Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, Gordon Wetzstein, Leonidas Guibas, Peter Wonka, Sergey Tulyakov, Jeong Joon Park, Andrea Tagliasacchi, and David B. Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling, . 2, 5

[3] Benjamin Biggs, Thomas Roddick, Andrew W. Fitzgibbon, and Roberto Cipolla. Creatures great and SMAL: recovering the shape and motion of animals from video. In *Proc. ECCV*, 2018. 1

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *arXiv.cs*, abs/2301.09632, 2023. 2

[5] Ang Cao, Justin Johnson, Andrea Vedaldi, and David Novotny. Lightplane: Highly-scalable components for neural 3d fields. 3

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. ICCV*, 2021. 3

[7] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proc. CVPR*, 2022. 2, 3

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *arXiv*, 2022. 2

[9] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10M+ 3D objects. *CoRR*, abs/2307.05663, 2023. 2

[10] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proc. CVPR*, 2023. 2, 3

[11] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *arXiv.cs*, abs/2301.10241, 2023. 2

[12] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. 2

[13] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoint without keypoints. 2

[14] Paul Henderson and Vittorio Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. 2

[15] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3d. 1, 2, 3

[16] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *Proc. ICLR*, 2024. 1

[17] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. SC-GS: Sparse-controlled gaussian splatting for editable dynamic scenes. 2

[18] Yanqin Jiang, Li Zhang, Jin Gao, Weimin Hu, and Yao Yao. Consistent4d: Consistent 360{\deg} dynamic object generation from monocular video. 2, 8

[19] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. 2

[20] Hiroharu Kato and Tatsuya Harada. Learning view priors for single-view 3d reconstruction.

[21] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. 2

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. 2

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *Proc. SIGGRAPH*, 42(4), 2023. 1

[24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015. 3, 4

[25] Filippos Kokkinos and Iasonas Kokkinos. Learning monocular 3d reconstruction of articulated categories from motion. 2

[26] Filippos Kokkinos and Iasonas Kokkinos. To the point: Correspondence-driven monocular 3d category reconstruction, 2021. 2

[27] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3D: Fast text-to-3D with sparse-view generation and large reconstruction model. *Proc. ICLR*, 2024. 3

[28] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. 2

[29] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3D reconstruction via semantic consistency. In *Proc. ECCV*, 2020. 2

[30] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. 2

[31] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. SyncDreamer: Generating multiview-consistent images from a single-view image. *arXiv*, (2309.03453), 2023. 2

[32] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: a skinned multi-person linear model. *ACM Trans. on Graphics (TOG)*, 2015. 1

[33] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Computer Graphocs*, 21(24), 1987. 5

[34] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. 2

[35] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. RealFusion: 360 reconstruction of any object from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[36] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. ECCV*, 2020. 2

[37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *Proc. SIGGRAPH*, 2022. 2

[38] Zijie Pan, Zeyu Yang, Xiatian Zhu, and Li Zhang. Fast dynamic 3d object generation from a single-view video. 2

[39] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *Proc. ICLR*, 2023. 2

[40] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3D object generation using both 2D and 3D diffusion priors. *arXiv.cs*, abs/2306.17843, 2023. 2

[41] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion, 2022. 1, 6

[42] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4d: Generative 4d gaussian splatting. 1, 2

[43] Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4gm: Large 4d gaussian reconstruction model, 2024. 1, 2

[44] Remy Sabathier, Niloy J. Mitra, and David Novotny. Animal avatars: Reconstructing animatable 3d animals from casual videos. In *Computer Vision – ECCV 2024*, pages 270–287, Cham, 2025. Springer Nature Switzerland. 1

[45] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv.cs*, abs/2310.15110, 2023. 2

[46] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, and Yaniv Taigman. Text-to-4d dynamic scene generation. 2, 5

[47] Samarth Sinha, Roman Shapovalov, Jeremy Reizenstein, Ignacio Rocco, Natalia Neverova, Andrea Vedaldi, and David Novotný. Common pets in 3D: Dynamic new-view synthesis of real-life deformable categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[48] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-It-3D: High-fidelity 3d creation from A single image with diffusion prior. *arXiv.cs*, abs/2303.14184, 2023. 2

[49] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a generalizable dynamic radiance field from monocular videos. 2

[50] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. TripoSR: Fast 3d object reconstruction from a single image. 3, 8

[51] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. In *ICLR*, 2019. 8

[52] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. 2

[53] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. MeshLRM: large reconstruction model for high-quality mesh. *arXiv*, 2404.12385, 2024. 2

[54] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering, . 2

[55] Shangzhe Wu, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. DOVE: Learning deformable 3d objects by watching videos, . 2

[56] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. MagicPony: Learning articulated 3D animals in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[57] Zijie Wu, Chaohui Yu, Yanqin Jiang, Chenjie Cao, Fan Wang, and Xiang Bai. SC4d: Sparse-controlled video-to-4d generation and motion transfer, . 2

[58] Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency, 2024. 5

[59] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. DMV3D: Denoising multi-view diffusion using 3D large reconstruction model. In *Proc. ICLR*, 2024. 3

[60] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. BANMo: Building animatable 3d neural models from many casual videos. 2

[61] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv.cs*, abs/2106.12052, 2021. 2

[62] Yuyang Yin, Dejia Xu, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4dgen: Grounded 4d content generation with spatial-temporal consistency. 2

[63] Yuyang Yin1, Dejia Xu2, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4DGen: Grounded 4D content generation with spatial-temporal consistency. *arXiv.cs*, 2023. 1

[64] Yifei Zeng, Yanqin Jiang, Siyu Zhu, Yuanxun Lu, Youtian Lin, Hao Zhu, Weiming Hu, Xun Cao, and Yao Yao. STAG4d: Spatial-temporal anchored generative 4d gaussians. 2

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

# LIM: Large Interpolator Model for Dynamic Reconstruction

## Supplementary Material

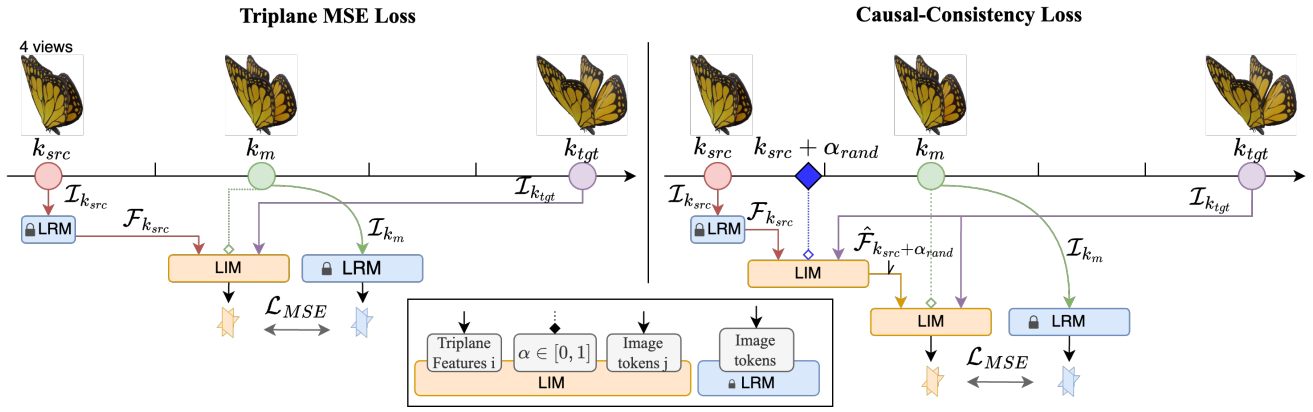


Figure 7. **LIM training losses**. (Left) The triplane MSE loss $\mathcal{L}_{\mathcal{T}}$ only supervises LIM on keyframes $k_m$. (Right) The causal consistency loss $\mathcal{L}_{\text{causal}}$ samples in-between keyframes with an additional forward-pass to LIM. Note that the second pass of LIM takes as input the intermediate features from LIM instead of the intermediate features from LRM.

## A. Additional Evaluations

We recommend looking at the project page [https://remysabathier.github.io/lim.github.io], to see the video results. In particular, the webpage contains video result of RGB interpolation, XYZ canonical tracking, monocular reconstruction and mesh reconstruction.

**Evaluation on OOD data** We provide qualitative results on the Consistent4D eval set, which includes *real-world scenes*, in Table 5.

## B. Additional Method Insights

**Weight Initialization.** The composition of blocks in LIM and LRM is presented in Fig. 2. We initialize LIM with LRM to take advantage of the learned 3D intermediate representation. More specifically, the intermediate-features cross-attention layers are derived from the self-attention layers from LRM. Furthermore, the image cross-attention layers are initialized using the image cross-attention layers from LRM, and the self-attention layers are initialized from the self-attention layer of LRM. Initialization is similar for $\overline{\text{LRM}}$ and $\overline{\text{LIM}}$ (presented in Fig. 9).

**Model size.** We ablate the choice of the number of layers in Tab. 6. We observe that LIM accuracy is proportional to the number of blocks in the architecture. However, adding more blocks in LIM slows down the interpolation. We set $N_{layer} = 6$ as a good trade-off between speed and accuracy.

**Dataset details** Our *3D* dataset includes 142,123 assets, while the *4D* dataset comprises 6,052 rigged models, each with 16 to 128 keyframes. We render the keyframes using Blender and the Cycles engine.

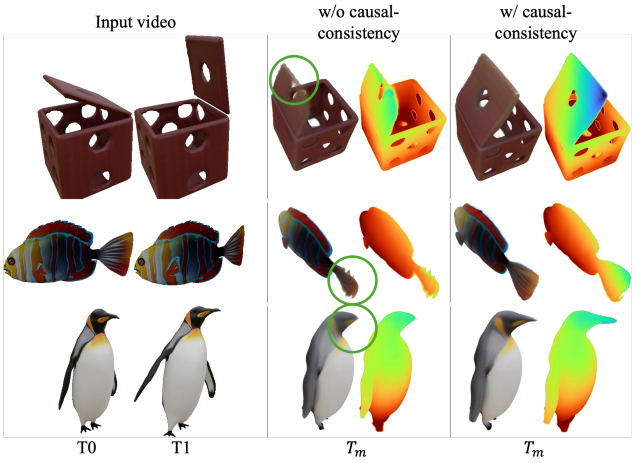


Figure 8. **Causal-loss ablation.** We show triplane interpolation result from LIM models trained either with the triplane MSE loss $\mathcal{L}_{\mathcal{T}}$ only, or with both $\mathcal{L}_{\mathcal{T}}$ and the causal-consistency loss $\mathcal{L}_{\text{causal}}$.

Table 5. **Monocular reconstruction (out of distribution OOD).**

| | Inf. Time | Consistent4D set | |
|---|---|---|---|
| | | LPIPS | FVD |
| Consistent4D | ~90 min | 0.428 | 1134.7 |
| TripoSR | ~0.5 min | 0.497 | 1428.2 |
| LIM (Our) | ~3 min | **0.114** | **781.9** |

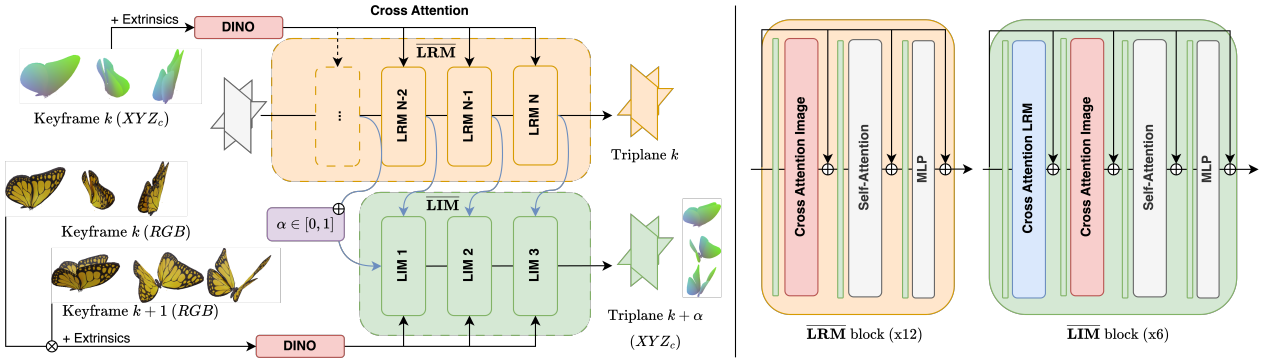


Figure 9. $\overline{\text{LIM}}$ **framework**. (Left) Given multi-view RGB images on 2 timesteps $k$ and $k+1$ and XYZ canonical renders on timestep $k$, $\overline{\text{LIM}}$ interpolates any intermediate 3D representation of the XYZ canonical coordinate at $k+\alpha, \alpha \in [0,1]$. This gives direct correspondences in 3D space between the source shape at $k$ and the interpolated shape at $k+\alpha$. In practice, our $\overline{\text{LIM}}$ architecture has 6 blocks and $\overline{\text{LRM}}$ 12 blocks. (Right) Block structure of $\overline{\text{LRM}}$ and $\overline{\text{LIM}}$. We include layer normalization before each module in blocks.

| | PSNR ↑ | $\text{PSNR}_{\text{FG}}$ ↑ | LPIPS ↓ |
|---|---|---|---|
| LIM- 3 layers | 22.35 | 14.56 | 0.079 |
| LIM- 8 layers | 23.19 | 16.2 | 0.075 |
| LIM | 23.11 | 16.12 | 0.075 |

Table 6. **Performance as a function of # layers** reporting interpolation accuracy of LIM while varying the number of transformer blocks in the architecture.

**Causal consistency loss.** We illustrate in Fig. 7 the behavior of the triplane MSE loss $\mathcal{L}_{\mathcal{T}}$ and the causal-consistency loss $\mathcal{L}_{\text{causal}}$ (see Sec. 3). $\mathcal{L}_{\mathcal{T}}$ involves a single pass of LIM and two passes of LRM, while $\mathcal{L}_{\text{causal}}$ involves 2 passes of LRM and 2 passes of LIM. Note that during LIM training, the weights of LRM are frozen. In practice, we discovered that the causal consistency loss was essential to achieve precise and accurate interpolation over a range of shapes and motions. We show interpolation results (in the same setting as Sec. 4.1) in Fig. 8, with a LIM model trained either with $\mathcal{L}_{\text{causal}}$ activated or deactivated.

**Positional Encoding** We apply positional encoding to the interpolation time $\alpha \in [0,1]$ with $\phi : \mathbb{R} \to \mathbb{R}^{2D}$, such that $\forall i \in [1, D], \phi(\alpha)[2i] = \cos(\alpha f_{2i}); \phi(\alpha)[2i+1] = \sin(\alpha f_{2i+1})$, and $f_i = \exp[-\frac{\log 10.000}{D}.i]$; we set $D = 512$ so that $2D$ matches the LRM embedding dimension.

**4D reconstruction with ARAP regularization** . We observe that our mesh-tracking framework can incorporate ARAP regularization to mitigate issues like triangle inversion or self-intersection. Instead of relying solely on direct matching through nearest neighbor search in the space of canonical coordinates (refer to Section Sec. 3.4), we implement a concise optimization loop. This loop incorporates both canonical-coordinate matching and ARAP energy as objectives to minimize.