GALILEO: Geometric RoPE with Galilean Key Transport for Dynamic Scenes

Youngmin Kim* Seonghyun Jin* Jong Chul Ye†
Kim Jaechul Graduate School of Artificial Intelligence, KAIST
Seoul, Republic of Korea
{zeromin03,jinotter3,jong.ye}@kaist.ac.kr
*Equal contribution †Corresponding author

Project page: https://bispl-wm.github.io/GALILEO/

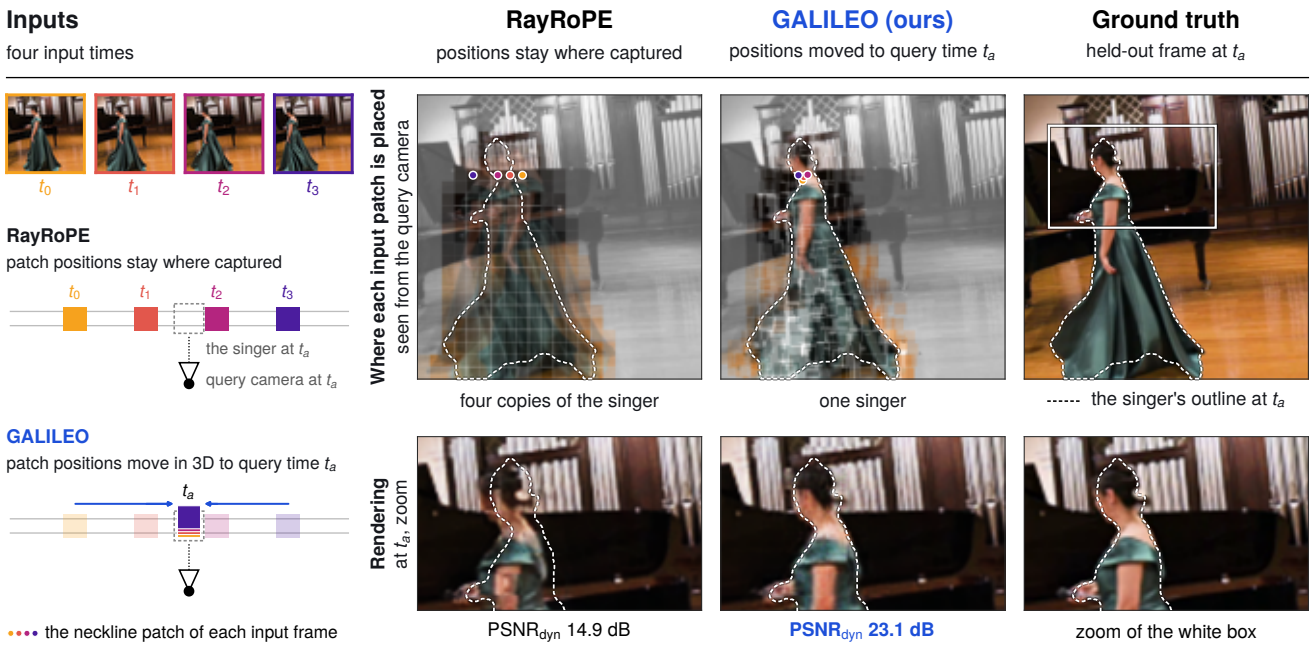

Figure 1. **A moving object's patches belong at the query time.** A novel view synthesis transformer renders the left view at a held-out query time t_a. *Left:* the inputs and a schematic of the idea. *Top row:* where the encoding places each input patch, at its predicted depth, seen from the query camera (four input times; static patches gray). RayRoPE [58] keeps each patch's 3D position where it was captured; GALILEO moves it to t_a by the patch's estimated 3D velocity. *Bottom row:* rendering at t_a (PSNR$_{dyn}$: PSNR on the moving region).

Abstract

*Geometry-aware rotary positional encodings, such as RayRoPE, help attention reason about the 3D spatial relationships between tokens and the query camera. However, they implicitly assume a static scene, encoding each key at its observed position regardless of when it was captured. For dynamic scenes with temporally separated input views, this mismatch places the same moving object at a different position in each input frame, so it can be rendered at an inaccurate position. While temporal encodings inform attention **when** a token was observed, they do not resolve **where** its content should be at the query time. We introduce GALILEO, a simple and general approach that aligns positional encodings with dynamic scene geometry through **Galilean transport**. Before positional encoding, each key's 3D position is transported to the query time using its estimated velocity and the temporal offset. By modifying only key positions, GALILEO integrates directly with existing encodings, including RayRoPE and URoPE, without altering their formula-*

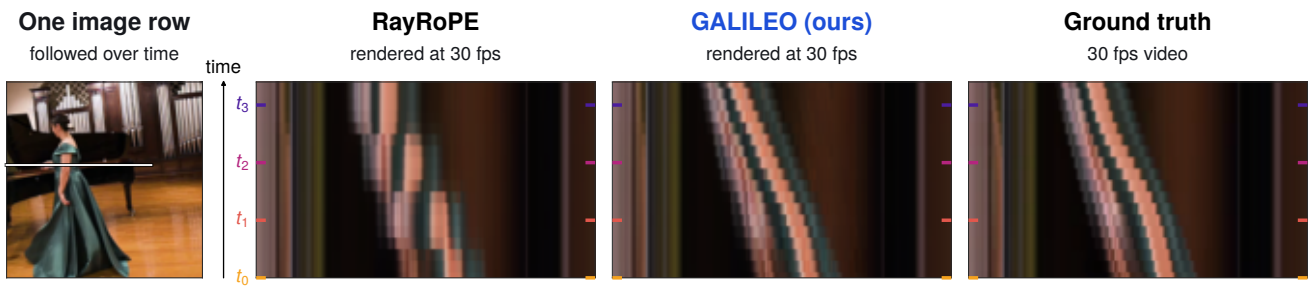

Figure 2. **A moving object is a line in space–time.** One image row (white line) over time: position runs right, time up, and ticks mark the input times. All are shown at the video's 30 fps. In the ground truth, the singer traces slanted lines. RayRoPE draws a staircase: its content stays where an input frame captured it, then jumps to the next. GALILEO follows the slanted lines. A second clip is in Appendix J.

tions. Velocities can be obtained from arbitrary sources; by default, we estimate them from token correspondences within the network. Integrated into LVSM, GALILEO improves dynamic view synthesis on Syn4D, dynamic Objaverse, and real-world Stereo4D videos over baselines. Ablations show that encoding velocities as token features yields smaller gains, while attention visualizations reveal that GALILEO attends to query-time locations of moving content, unlike RayRoPE. Beyond view synthesis, GALILEO also reduces tracking errors on dynamic objects in a pretrained 3D point tracker and improves moving-vehicle generation in a video generator.

1. Introduction

Attention is permutation invariant, so a transformer knows where its tokens are only through their positional encoding [51]. Rotary encodings [17, 45] make attention depend on the relative position of two tokens, and a recent line of work makes this relative position three-dimensional. GTA [37] and PRoPE [26] rotate queries and keys by the relative camera pose, and RayRoPE [58] and URoPE [60] also encode where along its camera ray the content of a token lies. Every position is expressed in the frame of the query camera, so attention depends on how two tokens are related in 3D and not on the choice of world frame. This matters most where a transformer must reason in 3D without an explicit 3D representation, as in LVSM [20], which renders a target view by attending from target rays to input image patches.

These existing geometry-aware encodings, however, implicitly assume that keys and queries correspond to the same instant. Each key is encoded at the 3D position of its content at capture time, even when the query corresponds to a different time. While this distinction is irrelevant for static scenes, it becomes critical for moving objects. Across input frames, the same object occupies different 3D positions, breaking the positional consistency on which RayRoPE relies [58] and deviating from the epipolar constraints used by URoPE [60]. As a result, the object may appear at multiple locations in a rendered view (Fig. 1), while temporal renderings exhibit stationary copies followed by abrupt jumps (Fig. 2). In short, existing encodings tell attention where moving content *was*, rather than where it *is*.

A common approach is to incorporate temporal information through temporal RoPE [54] or time embeddings [29, 57]. However, such signals indicate when a key was observed, not where its content is at the query time. Since all keys from the same frame share the same temporal information regardless of their motion, these approaches do not resolve the underlying geometric mismatch. Indeed, all our baselines incorporate temporal information but still encode moving keys at their observed positions.

In contrast, we argue that a geometry-aware encoding should place each key where its content is at the query time. To achieve this, we introduce GALILEO, a novel method that incorporates Galilean transport into positional encoding. For a key b observed at time t_b and a query at time t_a, its 3D points are shifted by $v_b(t_a - t_b)$, where v_b denotes the velocity of its content (Fig. 3). This transport provides a first-order approximation of motion over the short intervals between input frames. By modifying only key positions, GALILEO leaves queries and the attention operator unchanged, preserves the original encoding for stationary keys, and integrates directly with RayRoPE and URoPE. The method supports velocities from any source; by default, GALILEO estimates them within the network by matching tokens across adjacent frames and lifting the correspondences to 3D using the depths employed by the encoding.

Integrated into LVSM, GALILEO improves the rendering of moving content over baselines on synthetic Syn4D [19] and dynamic Objaverse [11] scenes, as well as real-world Stereo4D [21] videos. It also carries over to other models: attached to the pretrained tracker ARROW [14], it lowers 3D tracking error on dynamic objects, and in a 5B-parameter video generator that renders driving scenes from their geometry [36, 52], it improves moving objects with velocities from their 3D boxes.

Further analyses suggest that these improvements arise from aligning key positions with moving content, rather than

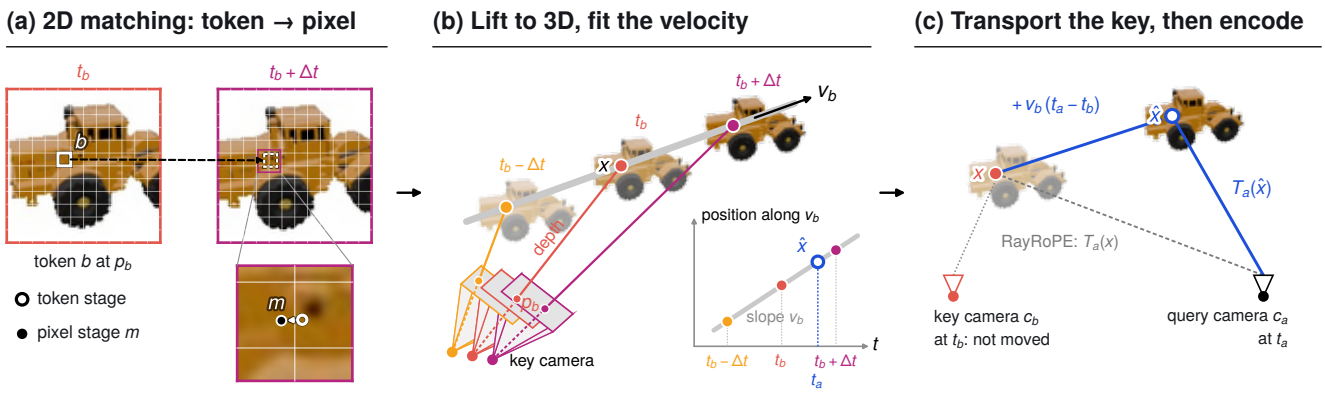

Figure 3. **Overview of GALILEO.** (a) Each input patch b is matched to its camera's frames at $t_b \pm \Delta t$, from token to pixel accuracy. (b) The patch is lifted to 3D along its ray at the encoding's depth, and so are its matches; a line through them gives a constant velocity v_b. (c) For a query at t_a, the points of key b are moved to $\hat{x} = x + v_b\,(t_a - t_b)$ then encoded as in RayRoPE. The key camera is not moved.

simply providing motion information as additional features. Attention visualizations show that GALILEO attends more to the actual locations of moving content, unlike RayRoPE, which favors positions implied by static geometry (Fig. 4). Moreover, GALILEO is far more robust to changes in the reference frame, even when the underlying scene remains unchanged (Fig. 5).

Our contributions are summarized as follows:

- We identify a key limitation of existing geometry-aware rotary encodings in dynamic scenes: they encode moving content at its observed position, not its query-time position, which temporal embeddings alone cannot resolve.
- We introduce GALILEO, a Galilean transport of keys inside the encoding that places each key where its content is at the query time. It integrates directly with RayRoPE and URoPE without modifying queries or the attention operator, and supports motion from arbitrary sources.
- We demonstrate improvements in dynamic view synthesis across three datasets, as well as in 3D tracking and video generation. Further analyses highlight the role of moving keys to the query-time position through attention visualizations and robustness tests under Galilean boosts.

2. Related Work

Geometry-aware positional encodings. Feed-forward view synthesis models [18, 20, 43] leave much of the geometry to how tokens are positioned. From absolute rays [44] and 3D positions [31, 61], encoding methods have evolved to relative ones built on rotary embeddings [17, 45]. For example, GTA [37] and PRoPE [26] transform queries, keys, and values by the relative camera pose; RayRoPE [58] and URoPE [60] also encode where along its ray a token's content lies. Modern video generators adopt such encodings for camera control [22, 25, 41] and world models [59].

Relative encodings make attention invariant to rigid transformations of the world coordinate frame, unlike absolute encodings. To account for time, temporal RoPE [53, 54] and time embeddings [29, 57] encode the observation time axis. However, they do not account for content motion, as all tokens within a frame share the same temporal information. In contrast, GALILEO transports each key to its query-time position, ensuring that, given accurate motion estimates, its content is encoded at the correct position relative to the query camera, regardless of the reference frame, including moving ones (Proposition 2).

Warping positional encodings. Training-free motion control in video diffusion transformers often warps rotary embeddings or queries along a desired 2D motion [9, 16, 24]. PE-Field [1, 2] and UCM [62] instead reproject the codes of context tokens into the target view with estimated depth and camera poses. However, these reprojections account for camera motion rather than the motion of the content itself: PE-Field 4D reprojects each source frame only to a target view at the same time, while UCM trains its model to disregard moving objects. In contrast, GALILEO transports each key according to its content's 3D motion to the query time, while encoding its position relative to the query camera. Whereas camera-only reprojection leaves moving content at its capture-time positions (Fig. 1), GALILEO aligns its observations at a common query-time position (Proposition 1), allowing attention to aggregate them (Fig. 4).

Motion in dynamic view synthesis. 4D scene representations model motion within the representation itself, through deformation or scene-flow fields [27, 38, 40], space–time planes [6, 15], or moving Gaussians [32, 56], optimized per scene or predicted feed-forward [3, 30, 34, 63]; other feed-forward models regress a separate set of Gaussians for each target time [29, 42]. In contrast, renderers that predict target views directly [13, 35, 49] lack an explicit representation to deform, while WildRayZer [8] handles dynamic content by masking it out. Motion compensation offers another approach, warping features or sampling locations, as in

video codecs [55], flow-aligned restoration [7], deformable and trajectory attention [39, 65], and DynIBaR [28]. Unlike these approaches, GALILEO incorporates scene flow into positional encoding, without requiring an explicit scene representation or resampling content. This allows seamless integration with existing token-based models (Sec. 5), avoids warping holes, and supports arbitrary query times using the same keys (Fig. 2).

3. GALILEO: Galilean Key Transport

3.1. Geometry-aware rotary encodings

Consider a model that receives input views with known cameras, captured at times t_n, and renders a target view for a given camera and time. Every view is split into patch tokens, and the transformer attends over the tokens of all views. Each token inherits the camera and timestamp of its view. Specifically, query a is associated with camera center c_a, rotation R_a, and time t_a, and key b is defined analogously.

We build on ray-based encodings [58, 60], which describe each token by its camera center c_b and a set of 3D points X_b along its rays, where its content may lie. Specifically, for query a, the points are expressed in its camera frame:

$$T_a(x) = R_a^\top (x - c_a), \tag{1}$$

and mapped to cosine and sine coefficients at a set of frequencies. The coefficients form a block-diagonal matrix $M_a(b)$ of 2×2 rotations, and attention between query a and key b uses the logit

$$\ell_{ab} = \big(M_a(a)^\top q_a\big)^\top \big(M_a(b)^\top k_b\big)/\sqrt{d}, \tag{2}$$

with head dimension d. Values are transformed by the same matrix as keys, and the output is rotated back by $M_a(a)$. Accordingly, expressing coordinates relative to the query camera makes attention invariant to rigid transformations of the world coordinate frame.

The base encodings used in this paper also incorporate temporal RoPE, with selected rotation blocks applying phases proportional to $t_a - t_b$ (Appendix C). Each input or target view serves as a query view in turn. Since its tokens share the same camera and timestamp, the key encoding matrices can be computed once per query view (Appendix A).

3.2. Key transport

The issue. Consider moving content observed at different times. The points in X_b represent the positions of key b's content at its observation time t_b, whereas Eq. (1) expresses these positions relative to the query camera at time t_a. This yields the correct query-relative position only if the content remains stationary between the two instants. For content following a trajectory $x(t)$, however, the query requires its position at time t_a, expressed as $T_a(x(t_a))$. Thus, while Eq. (1) is sufficient for static content, it encodes outdated positions for moving content.

Galilean transport. We approximate the motion of content over the interval between a key and a query using a first-order model. Let v_x be the 3D velocity of the content at point $x \in X_b$, in world coordinates. Before encoding key b for a query at time t_a, we move its points to that time,

$$\hat{x} = x + v_x\,(t_a - t_b), \qquad x \in X_b, \tag{3}$$

and compute the key's coefficients and its matrix $\hat{M}_a(b)$, which transforms the key and its value, from $T_a(\hat{x})$ instead of $T_a(x)$. In RayRoPE, all points of a token share its velocity, whereas in URoPE, each depth anchor has its own velocity.

The camera center c_b remains unchanged, as it represents the observer's position rather than that of the content. Target tokens, which carry no observed content, are not transported, and query encodings remain unchanged. Since key encodings are already computed per query view, the transport requires only shifting key points before encoding, allowing the same key to be encoded at different positions for different query times. For static content ($v_x = 0$) or when $t_a = t_b$, the method reduces exactly to the original encoding, with temporal dependence determined solely by $t_a - t_b$.

3.3. Theoretical findings

RayRoPE is designed to assign similar positional encodings to patches observing the same 3D point [58]. Galilean transport restores this property for moving content. The four dots on the neckline in Fig. 1 illustrate such observations.

Proposition 1 (Co-observation)**.** *Suppose keys b and b' observe the same 3D point moving at constant velocity v. With this velocity used for transport, their content points are both mapped to $T_a\big(x(t_a)\big)$ for any query a.*

Proof. This follows from $x(t_b) + v\,(t_a - t_b) = x(t_a)$, whereas without transport, the query-relative positions of the two observations differ by $R_a^\top v\,(t_b - t_{b'})$. □

The relative transformation in Eq. (1) makes the encoding invariant to rigid changes in the origin and orientation of the world frame. With temporally separated observations, however, the reference frame may itself move, as when camera poses are expressed relative to a moving vehicle.

Proposition 2 (Galilean boost)**.** *Consider a change of reference frame that shifts all camera and content positions at time t by $w\,t$. Then the transported coordinates $T_a(\hat{x})$ are unchanged, and the static coordinates $T_a(x)$ change by $-R_a^\top w\,(t_a - t_b)$.*

Under this transformation, velocities estimated from 3D positions become $v_x + w$, yielding

$$(x + wt_b) + (v_x + w)(t_a - t_b) - (c_a + wt_a) = \hat{x} - c_a.$$

	Stereo4D (real)					Objaverse					Syn4D				
	image		video			image		video			image		video		
Method	$\text{PSNR}_{\text{dyn}}\uparrow$	PSNR↑	tOF↓	$\delta_{\text{avg}}\uparrow$	JEDi↓	$\text{PSNR}_{\text{dyn}}\uparrow$	PSNR↑	tOF↓	$\delta_{\text{avg}}\uparrow$	JEDi↓	$\text{PSNR}_{\text{dyn}}\uparrow$	PSNR↑	tOF↓	$\delta_{\text{avg}}\uparrow$	JEDi↓
Plücker	23.59	29.17	1.52	.918	1.047	–	–	–	–	–	19.20	21.72	2.79	.616	2.632
PRoPE	24.06	29.95	1.44	.926	0.741	16.73	27.51	1.31	.748	0.250	19.34	22.57	2.39	.635	2.590
GTA	24.15	30.07	1.41	.928	0.726	16.81	27.76	1.41	.755	0.184	19.28	22.46	2.31	.632	2.789
URoPE	24.61	30.73	1.08	.939	0.493	16.33	27.07	1.44	.730	0.236	19.66	22.99	2.18	.627	1.410
+ GALILEO	25.41	31.00	0.89	.950	0.391	16.70	27.25	1.30	.739	0.197	20.14	23.11	1.82	.637	0.787
Δ	+0.79	+0.27	−0.19	+.011	−0.102	+0.37	+0.18	−0.14	+.010	−0.039	+0.49	+0.12	−0.37	+.010	−0.622
RayRoPE	24.61	31.04	1.04	.947	0.466	16.67	27.42	1.53	.743	0.346	19.88	23.17	2.19	.639	1.075
+ GALILEO	25.78	31.34	0.80	.958	0.343	17.96	28.25	1.12	.770	0.268	20.82	23.51	1.65	.668	0.380
Δ	+1.16	+0.30	−0.24	+.010	−0.122	+1.29	+0.83	−0.41	+.026	−0.078	+0.94	+0.34	−0.53	+.029	−0.695

Table 1. **Main results** on three datasets (metrics and averaging in Sec. 4.1). Every method carries a time signal, a temporal RoPE or, for Plücker, a time embedding, and each + GALILEO model differs from its base encoding only in the motion components. Δ: change from the base encoding; every change is beyond noise (90% bootstrap interval, or a paired permutation test for JEDi; Appendix C). –: Plücker failed to train on Objaverse. All image metrics are in Tab. 10.

With true content displacements, the same principle extends to arbitrary time-dependent rigid changes of reference frame. For Galilean boosts, even least-squares velocity estimates preserve this invariance regardless of the underlying content motion. Thus, while static encoding correctly aligns only content stationary in the given reference frame, transported content coordinates are invariant to Galilean boosts. This property applies to content points, not to the key camera center, which remains untransported and changes by $R_a^\top w\,(t_b - t_a)$. We evaluate the resulting robustness of trained models in Sec. 5.

Our theoretical understanding clarifies the difference from temporal RoPE. Temporal RoPE encodes when a key was observed, assigning the same temporal information to all keys from a frame regardless of their motion. Specifically, the pair $\big(T_a(\hat{x}),\, t_a - t_b\big)$ describes the content relative to the query in both space and time. Temporal RoPE provides the time offset $t_a - t_b$, which GALILEO preserves, but cannot supply the first $(T_a(\hat{x}))$. In contrast, Galilean transport adjusts where each key's content is encoded.

3.4. Velocity estimation

While GALILEO supports velocities from arbitrary sources, including optical flow and 3D tracks (Sec. 5), we estimate them within the network by default, without relying on external models at test time (Fig. 3a,b). Specifically, each input token is matched to adjacent frames from the same camera using a coarse-to-fine approach [46, 47]: initial correspondences are established at token resolution using transformer features and refined to pixel accuracy with a lightweight CNN. The matched points are then lifted to 3D using camera poses and the encoding's depths, accounting for camera motion, and their velocities are fit by least squares under a constant-velocity model (Appendix B).

Depth errors. Depth errors can be amplified when estimating velocity from closely spaced frames. A depth error δ at one end of a correspondence introduces a transported-position error of approximately $\delta\,|t_a - t_b|/\Delta t$, where Δt is the interval between adjacent frames. This error lies primarily along the capturing camera's viewing ray, making it less apparent from that viewpoint but potentially significant from others. Thus, accurate depth is important for reliable transport (Sec. 4.3), which motivates depth supervision.

3.5. Training

We train GALILEO with the backbone's image loss, a matching loss, and a depth loss:

$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{match}} + \mathcal{L}_{\text{depth}}, \tag{4}$$

where $\mathcal{L}_{\text{match}}$ supervises both stages of matching using ground-truth correspondences, and $\mathcal{L}_{\text{depth}}$ supervises the depths used in positional encoding to mitigate errors amplified during velocity estimation. Although the original RayRoPE does not use depth supervision, we apply the same depth losses to its baseline in our experiments to isolate the effect of motion transport. Loss definitions and weights are provided in Appendix A.

To mitigate instability in velocity predictions early in training, we use ground-truth velocities for key transport during the first s_0 steps. We then gradually transition to predicted velocities through linear interpolation, $v = (1 - \alpha)v_{\text{gt}} + \alpha v_{\text{pred}}$, where α increases from 0 at step s_0 to 1 at step s_1. Thereafter, and at test time, only predicted velocities are used. This schedule is inspired by mix-forcing [22, 64] and related to scheduled sampling [5].

4. Experiments

4.1. Setup

Data. We evaluate GALILEO on three datasets spanning synthetic and real-world data, from individual objects to complex scenes: dynamic Objaverse [11], consisting of animated objects rerendered by us; Syn4D [19], featuring synthetic scenes with a person and multiple animated objects; and

Stereo4D [21], containing real-world stereo videos. Inputs are a few frames from two cameras over a short interval, and targets are other frames of the same test clip: novel times from the input cameras and, except on Stereo4D, which has only two cameras, novel views from new cameras. Test clips are disjoint from training (Appendix C).

Metrics. We evaluate image quality using PSNR on dynamic regions (PSNR_{dyn}), defined as pixels displaced by more than 1.5 px relative to the nearest input time, and full-image PSNR, with SSIM and LPIPS in the supplement. For videos rendered over each clip, we assess temporal quality using tOF [10], which measures agreement with ground-truth motion; δ_{avg} [12], which evaluates how accurately a tracker [23] follows moving points; and JEDi [33], which compares distributions of V-JEPA [4] features sensitive to temporal distortions. Results average novel views and novel times with equal weight, and changes count as significant when their 90% confidence intervals exclude zero (Appendix C).

Baselines and training. All methods share the LVSM backbone and training recipe, are trained from scratch for 20k steps, and differ only in the positional encoding: Plücker ray tokens with a time embedding, and PRoPE [26], GTA [37], URoPE [60], and RayRoPE [58], each with a temporal RoPE. Unless stated otherwise, GALILEO is applied to RayRoPE, and RayRoPE receives the same depth losses. No method uses ground-truth depth, flow, or motion masks at test time (details in Appendix C).

4.2. Main results

Tab. 1 summarizes the results with Tab. 10 for the remaining image metrics. On both RayRoPE and URoPE, GALILEO improves every metric of Tab. 1 on all three datasets, including the real videos of Stereo4D, and raises PSNR_{dyn} by 0.9–1.3 dB on RayRoPE and 0.4–0.8 dB on URoPE. The video metrics indicate that the gain is not only in sharpness but also in motion: rendered content moves more like the real content (tOF), a tracker follows its points more closely (δ_{avg}), and JEDi falls by about two thirds with RayRoPE on Syn4D. The gains persist with four times longer training on Syn4D (Tab. 15): at 80k steps, GALILEO still improves every metric, and RayRoPE at 80k stays below GALILEO at 20k in PSNR_{dyn} and in every video metric. They also persist with twice as many layers on Stereo4D (Tab. 13).

Both models render any time in the input window; over that window, RayRoPE renders a moving object as a staircase and GALILEO as the slanted line of the ground truth (Fig. 2).

4.3. Ablation studies

Tab. 2 disentangles the contributions of key transport, motion information, supervision, and temporal encoding. First, to determine whether the gains arise from motion information or key placement, we compare GALILEO with two alternatives: training the matcher without transporting keys, and providing the estimated velocities as additional token features. Both yield only marginal improvements in PSNR_{dyn} and little change in motion metrics, whereas transporting keys with the same velocities improves all metrics substantially. Unlike token features, which remain fixed across queries, the transport adjusts each key's position to the query time. Second, even with image reconstruction loss alone, the transport improves PSNR_{dyn} by 0.56 dB, accounting for 60% of GALILEO's gain. Additional supervision contributes the remaining 40% and improves LPIPS (Appendix G). Token-level matching already improves PSNR_{dyn}, with the largest additional gain coming from restricting matches to adjacent frames (Appendix B). Disabling transport at test time reduces PSNR_{dyn} and all video metrics to RayRoPE's level or worse across all three datasets. This confirms that the trained model relies on the transported key positions rather than merely benefiting from motion-related supervision.

	image		video	
Model	$\text{PSNR}_{\text{dyn}}\uparrow$	PSNR↑	$\delta_{\text{avg}}\uparrow$	JEDi↓
Syn4D (Δ vs. (a); predicted warp vs. (b))				
RayRoPE (a)	19.88	23.17	.639	1.075
RayRoPE, matching losses	20.11 +0.23	23.38 +0.21	.643 +.004	1.008 −0.067
RayRoPE, velocity features	20.10 +0.22	23.38 +0.21	.644 +.005	0.993 −0.082
GALILEO (b)	20.82 +0.94	23.51 +0.34	.668 +.029	0.380 −0.695
GALILEO, GT motion	21.26 +1.38	23.66 +0.49	.694 +.055	0.398 −0.677
RayRoPE, predicted warp	20.58 −0.24	23.26 −0.25	.669 +.001	0.673 +0.293
Syn4D, GT depth (Δ vs. (a); GT warp vs. (c))				
RayRoPE	21.43 +1.55	25.09 +1.92	.705 +.066	0.603 −0.472
GALILEO, GT motion (c)	23.75 +3.87	25.66 +2.49	.786 +.147	0.131 −0.944
RayRoPE, GT warp	23.85 +0.11	25.30 −0.36	.775 −.011	0.273 +0.142
Motion off at test (Δ vs. the model with motion)				
Syn4D	19.84 −0.98	23.29 −0.22	.641 −.027	1.159 +0.779
Syn4D, GT depth and motion	21.06 −2.69	24.91 −0.75	.681 −.105	0.575 +0.445
Objaverse	16.15 −1.81	27.61 −0.64	.745 −.025	0.393 +0.125
Stereo4D	24.47 −1.31	30.95 −0.39	.947 −.011	0.514 +0.171

Table 2. **Motion and depth** (metrics as in Tab. 1; rows explained in Sec. 4.3). Small numbers: change from the reference named in each block (green: better, red: worse, gray: within noise). GT: ground truth, in training and test. GT / predicted warp: inputs warped to the query time instead. Other image metrics are in Tab. 11.

The gain of GALILEO grows as keys are placed more accurately. We train and evaluate GALILEO using ground-truth motion to transport keys, with predicted or ground-truth depth. Ground-truth motion further improves PSNR_{dyn}, while the estimated velocities recover about two-thirds of its gain over RayRoPE. Depth also limits transport: even with correct motion, inaccurate depth misplaces transported keys. With ground-truth depth, ground-truth motion improves PSNR_{dyn} by 2.3 dB, compared with 1.38 dB using predicted depth (Tab. 2 and Fig. 11).

We further compare GALILEO with two variants of warping the input images to the query time, using ground-truth depth and motion or those predicted by GALILEO. The ground-truth comparison favors warping: with the same per-token motion and a ground-truth mask of the moving pixels, it moves only those pixels within each token, whereas our

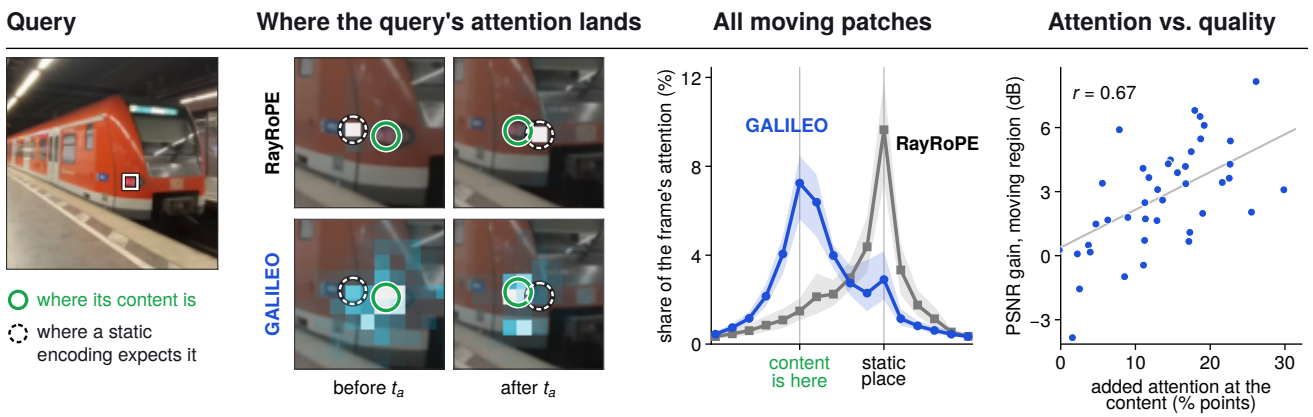

Figure 4. **Attention follows the encoding.** *Left:* a query patch on the moving train at the held-out time t_a and its attention in the two input frames farthest in time. Green circle: where the patch's content is in that frame, found by optical flow from the ground-truth frame at t_a. Dashed circle: where a static scene would put it, the patch's 3D point at t_a projected into the frame. *Middle:* attention of all moving patches along the line through the two places. *Right:* per held-out frame, the attention that GALILEO adds at the content predicts its gain in PSNR_{dyn}.

transport moves each key as a whole. Still, it only matches GALILEO in PSNR_{dyn} and is worse in full-image PSNR and all motion metrics, with twice the JEDi. The predicted variant falls below GALILEO in both PSNRs and JEDi (Tab. 2).

5. Discussion

Attention follows the encoding. To examine whether the gains arise from key placement, we analyze where target tokens attend in the input views (Fig. 4). We compare attention at two locations: the *content place*, where the target content appears in an input frame, estimated using optical flow [50] from the ground-truth target frame; and the *static place*, where the content would appear under a static-scene assumption, obtained by projecting its 3D point at the ground-truth depth into the input frame.

The two places coincide for static content, to 0.2 px in the median, and separate for moving content. On 80 Stereo4D test clips, we take every pair of a moving target token and an input view whose two places are at least 10 px apart, and measure the share of the frame's attention within a disc around each place, of radius half their distance (Appendix D).

RayRoPE puts 20.8% of a frame's attention at the static place and 6.9% at the content; GALILEO reverses this, with 22.2% at the content and 8.8% at the static place. On static content, the two models attend alike. The first layer of GALILEO, which runs before the velocities exist, still attends like RayRoPE; the shift appears in the later layers (Fig. 8). Across held-out frames, the added attention at the content place correlates with the gain in PSNR_{dyn}.

Robustness to Galilean boosts. We test Proposition 2 by expressing the same clips in a moving reference frame, without requiring additional data. Specifically, we shift all input and target camera centers by $w\,t$, while keeping images, rotations, intrinsics, and timestamps unchanged. Since the physical scene and ground-truth images remain identical, and camera poses enter the models only through positional encoding, any change in the output can be attributed to the encoding. We choose w along the image x-axis of the first camera and express its magnitude as the resulting drift at the

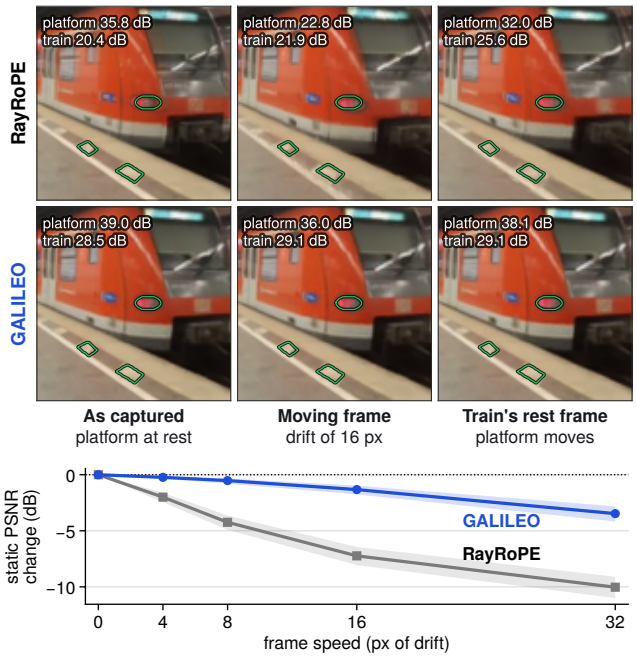

Figure 5. **The Galilean boost test.** The same clip described from a frame moving at constant velocity w: every camera position is shifted by $w\,t$, and nothing else changes. *Top:* RayRoPE renders well only what is at rest in the frame of its poses; GALILEO renders the platform and the train in every frame. Green: the lamp and two platform tiles in the ground truth. *Bottom:* change in static-region PSNR with frame speed.

	EPE↓			APD↑		
	Input	Between	After	Input	Between	After
ARROW	4.89	6.86	8.67	44.6	35.6	29.2
+ plain adapter	4.29	6.08	7.64	47.8	38.5	32.0
+ RayRoPE	3.82	4.99	7.16	50.9	43.5	33.8
+ RayRoPE + GALILEO	**3.66**	**4.47**	**6.68**	**51.8**	**46.2**	**35.6**

Table 3. **3D tracking with ARROW** on dynamic Objaverse; each adapter is added to frozen ARROW. At the input times, every query point is scored; between and after them, points that move more than 1.5 px. EPE is the 3D end-point error and APD the mean share of points within 1–16%, both relative to object size (Appendix H).

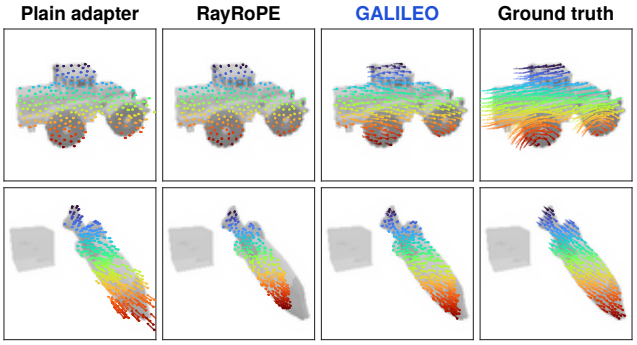

Figure 6. **3D tracks with ARROW** on two dynamic Objaverse objects, predicted from inputs at 0 and 0.1 s up to 0.3 s. The tracks of GALILEO come closest to the ground truth.

median scene depth over the input window. Fig. 5 (bottom) shows the static-region PSNR as a function of frame speed. In a moving frame, static content has a nonzero velocity, which RayRoPE does not account for; at a drift of 16 px, its static-region PSNR drops by 7.2 dB. GALILEO, whose estimated velocities include the frame motion, loses 1.3 dB. In the clip of Fig. 5 (top), setting w to minus the velocity of the train brings the train to rest and sets the platform in motion. RayRoPE then gains 5.2 dB on the train and loses 3.8 dB on the platform, while GALILEO changes by less than 1 dB in both regions.

Although far more robust, GALILEO is not exactly invariant: its first layer runs without velocities, and key camera centers are not transported (Proposition 2).

Extension to URoPE. URoPE has no per-token depth, so we lift each match at every depth anchor and move each anchor by its velocity. GALILEO improves all URoPE metrics on every dataset, generally less than on RayRoPE (Tab. 1).

3D tracking. Attached as an adapter to ARROW [14], a pretrained model for 3D reconstruction and point tracking, GALILEO tracks points more accurately than RayRoPE with a temporal RoPE at every time (Tab. 3 and Fig. 6), lowering the 3D end-point error by 10% between the two input times, 7% after them, and 4% at the input times. A plain adapter of the same size improves on ARROW less, so the gains do not come from the added parameters alone.

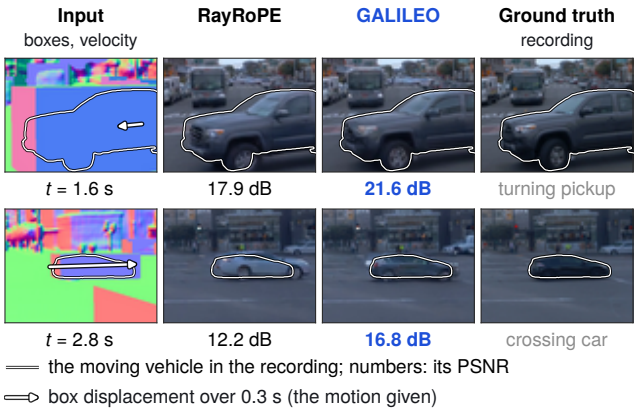

Figure 7. **A video renderer for code worlds** on held-out Waymo clips, given each vehicle as its 3D box and, for GALILEO, its motion. RayRoPE draws the moving vehicle off its place in the recording; GALILEO draws it in place.

		second after the first frame				
Conditioning	PSNR_{dyn}	0–1	1–2	2–3	3–4	IoU
Token conditioning	16.98	19.73	16.90	16.07	15.70	.628
+ RayRoPE	17.01	19.61	17.03	16.20	15.71	.660
+ GALILEO	**17.45**	**20.11**	**17.46**	**16.76**	**16.04**	**.673**
+ GALILEO, motion off	16.89	19.43	16.84	16.10	15.45	.630

Table 4. **A video renderer for code worlds** on Waymo (Appendix I): PSNR_{dyn} on moving vehicles over the 4 s generated video and per second, and their silhouette IoU (recording: .879). Motion off: the same GALILEO model with every velocity set to zero at test time. Best in bold.

Velocities from optical flow. On Stereo4D, replacing our matcher with RAFT [50] flow and dropping the matching loss keeps most of the gain: its PSNR_{dyn} is 0.18 dB below our matcher's and far above the model without motion (Tab. 10), with slightly better tOF and δ_{avg} but worse JEDi.

A video renderer for code worlds. GALILEO also attaches to a large video generator. AgentGarten [36] renders code worlds into video, conditioned on their geometry; we attach its conditioning to Wan2.2-TI2V-5B [52] and generate 4 s of Waymo [48] front-camera video from the first frame, per-frame geometry, and vehicle boxes (Appendix I). Moving each key by the velocity of its vehicle's ground-truth box raises PSNR_{dyn} on moving vehicles over RayRoPE in every second, whereas AgentGarten's token conditioning helps only in the first second (Tab. 4 and Fig. 7). Switching the motion off at test time removes the gain.

6. Conclusion

We introduced GALILEO, which encodes each key where its content is at the query time by transporting it inside the encoding along its motion, and leaves queries and the attention operator unchanged. Attached to RayRoPE and

URoPE, it improves moving content on synthetic and real multi-view videos over encodings that already carry time: attention moves from the static place of the content to the content itself, and the model becomes far less sensitive to the inertial frame of its cameras. The transport also improves a pretrained 3D tracker and a video generator.

Limitations. By default, GALILEO estimates a constant velocity between adjacent frames, which does not follow content that accelerates or turns; 3D tracks or ground-truth motion can replace the estimate where available. This estimate has a limited range, and depth and motion accuracy limit the transport (Appendix B and Sec. 4.3). The full model also relies on depth and motion supervision that unlabeled video does not provide.

References

[1] Yunpeng Bai, Haoxiang Li, and Qixing Huang. Positional encoding field. In *ICLR*, 2026. 3

[2] Yunpeng Bai, Haoxiang Li, and Qixing Huang. PE-Field 4D: Video generation models as canvas. In *SIGGRAPH Asia Conference Papers*, 2026. 3

[3] Matteo Balice, Yanik Kunzi, Chenyangguang Zhang, Matteo Matteucci, Marc Pollefeys, and Sunghwan Hong. No pose, no problem in 4D: Feed-forward dynamic gaussians from unposed multi-view videos. In *NeurIPS*, 2026. 3

[4] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *TMLR*, 2024. 6

[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 2015. 5

[6] Ang Cao and Justin Johnson. HexPlane: A fast representation for dynamic scenes. In *CVPR*, 2023. 3

[7] Kelvin C. K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. BasicVSR++: Improving video super-resolution with enhanced propagation and alignment. In *CVPR*, 2022. 4

[8] Xuweiyi Chen, Wentao Zhou, and Zezhou Cheng. WildRayZer: Self-supervised large view synthesis in dynamic environments. In *CVPR*, 2026. 3

[9] Kyobin Choo, Youngmin Kim, Hyunkyung Han, Geunrip Park, Chanyoung Kim, Sunyoung Jung, and Seong Jae Hwang. QWERTY: Training-free motion control via query-warped video diffusion transformers. In *ECCV*, 2026. 3

[10] Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for GAN-based video generation. *ACM TOG*, 39 (4), 2020. 6

[11] 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 *CVPR*, 2023. 2, 5

[12] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A benchmark for tracking any point in a video. In *NeurIPS Datasets and Benchmarks Track*, 2022. 6

[13] Baback Elmieh, Lynn Tsai, Zeman Li, Srinivas Kaza, Tiancheng Sun, Gabor Csapo, Ali Behrouz, Yuan Deng, Stephen Lombardi, Steven M. Seitz, and Xuan Luo. Online neural space time memory for dynamic novel view synthesis. *arXiv preprint arXiv:2607.15271*, 2026. 3

[14] Ilya Fradlin, Christian Schmidt, Jens Piekenbrinck, Karim Knaebel, Gonzalo Martin Garcia, and Bastian Leibe. ARROW: Arbitrary reconstruction and tracking of 4D observations in the wild. *arXiv preprint arXiv:2610.01314*, 2026. 2, 8

[15] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 3

[16] Ahmet Berke Gökmen, Yiğit Ekin, Bahri Batuhan Bilecen, and Aysegul Dundar. RoPECraft: Training-free motion transfer with trajectory-guided RoPE optimization on diffusion transformers. In *NeurIPS*, 2025. 3

[17] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *ECCV*, 2024. 2, 3

[18] 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 *ICLR*, 2024. 3

[19] Zeren Jiang, Yushi Lan, Yihang Luo, Yufan Deng, Zihang Lai, Edgar Sucar, Christian Rupprecht, Iro Laina, Diane Larlus, Chuanxia Zheng, and Andrea Vedaldi. Syn4D: A multiview synthetic 4D dataset. In *ECCV*, 2026. 2, 5

[20] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3D inductive bias. In *ICLR*, 2025. 2, 3

[21] Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4D: Learning how things move in 3D from internet stereo videos. In *CVPR*, 2025. 2, 6

[22] Seonghyun Jin, Youngmin Kim, Sunwoo Park, and Jong Chul Ye. CRePE: Curved ray expectation positional encoding for unified-camera-controlled video generation. In *NeurIPS*, 2026. 3, 5

[23] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *ICCV*, 2025. 6

[24] Cheng Lei, Jiayu Zhang, Yue Ma, Xinyu Wang, Long Chen, Liang Tang, Yiqiang Yan, Fei Su, and Zhicheng Zhao. DiTraj: Training-free trajectory control for video diffusion transformer. *arXiv preprint arXiv:2509.21839*, 2025. 3

[25] Chunyang Li, Yuanbo Yang, Jiahao Shao, Hongyu Zhou, Katja Schwarz, and Yiyi Liao. ReRoPE: Repurposing RoPE for relative camera control. *arXiv preprint arXiv:2602.08068*, 2026. 3

[26] Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. In *NeurIPS*, 2025. 2, 3, 6

[27] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 3

[28] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural dynamic image-based rendering. In *CVPR*, 2023. 4

[29] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, and Jiahui Huang. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. In *NeurIPS*, 2025. 2, 3

[30] Chenguo Lin, Yuchen Lin, Panwang Pan, Yifan Yu, Tao Hu, Honglei Yan, Katerina Fragkiadaki, and Yadong Mu. MoVieS: Motion-aware 4D dynamic view synthesis in one second. In *CVPR*, 2026. 3

[31] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETR: Position embedding transformation for multi-view 3D object detection. In *ECCV*, 2022. 3

[32] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 3

[33] Ge Ya Luo, Gian Mario Favero, Zhi Hao Luo, Alexia Jolicoeur-Martineau, and Christopher Pal. Beyond FVD: An enhanced evaluation metrics for video generation distribution quality. In *ICLR*, 2025. 6

[34] Ziqiao Ma, Xuweiyi Chen, Shoubin Yu, Sai Bi, Kai Zhang, Ziwen Chen, Sihan Xu, Jianing Yang, Zexiang Xu, Kalyan Sunkavalli, Mohit Bansal, Joyce Chai, and Hao Tan. 4D-LRM: Large space-time reconstruction model from and to any view at any time. In *NeurIPS*, 2025. 3

[35] Ziqiao Ma, Xueyang Yu, Haoyu Zhen, Yuncong Yang, Joyce Chai, and Chuang Gan. Fast spatial memory with scalable elastic test-time training. In *ECCV*, 2026. 3

[36] MirroS Team. Agentgarten: Code worlds for evolving agents. *arXiv preprint arXiv:2610.12374*, 2026. 2, 8

[37] Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. GTA: A geometry-aware attention mechanism for multi-view transformers. In *ICLR*, 2024. 2, 3, 6

[38] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021. 3

[39] Mandela Patrick, Dylan Campbell, Yuki M. Asano, Ishan Misra, Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and João F. Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. In *NeurIPS*, 2021. 4

[40] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 3

[41] Zhijian Qiao, Xinjiang Wang, Jiajie Chen, Haoming Huang, Meng Li, Chih-Chung Chou, Jing Wang, and Shaojie Shen. MeRoPE: Metric rotary position embedding for camera-controlled video generation. *arXiv preprint arXiv:2609.01252*, 2026. 3

[42] 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. In *NeurIPS*, 2024. 3

[43] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *CVPR*, 2022. 3

[44] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Frédo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021. 3

[45] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568, 2024. 2, 3

[46] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 5

[47] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, 2021. 5

[48] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 8

[49] Thomas Tanay, Mohammed Brahimi, Michal Nazarczuk, Qingwen Zhang, Sibi Catley-Chandar, Arthur Moreau, Zhensong Zhang, and Eduardo Pérez-Pellitero. GRVS: A generalizable and recurrent approach to monocular dynamic view synthesis. In *CVPR Findings*, 2026. 3

[50] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 7, 8

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[52] Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 2, 8

[53] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 3

[54] Yiming Wang, Qihang Zhang, Shengqu Cai, Tong Wu, Jan Ackermann, Zhengfei Kuang, Yang Zheng, Frano Rajič, Siyu Tang, and Gordon Wetzstein. BulletTime: Decoupled control of time and camera pose for video generation. In *CVPR*, 2026. 2, 3

[55] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE TCSVT*, 13(7):560–576, 2003. 4

[56] 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. In *CVPR*, 2024. 3

[57] Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T. Barron, and Aleksander Holynski. CAT4D: Create anything in 4D with multi-view video diffusion models. In *CVPR*, 2025. 2, 3

[58] Yu Wu, Minsik Jeon, Jen-Hao Rick Chang, Oncel Tuzel, and Shubham Tulsiani. RayRoPE: Projective ray positional encoding for multi-view attention. In *ECCV*, 2026. 1, 2, 3, 4, 6

[59] Chendong Xiang, Jiajun Liu, Jintao Zhang, Xiao Yang, Zhengwei Fang, Shizun Wang, Zijun Wang, Yingtian Zou, Hang Su, and Jun Zhu. Geometry-aware rotary position embedding for consistent video world model. *arXiv preprint arXiv:2602.07854*, 2026. 3

[60] Yichen Xie, Depu Meng, Chensheng Peng, Yihan Hu, Quentin Herau, Masayoshi Tomizuka, and Wei Zhan. URoPE: Universal relative position embedding across geometric spaces. In *ECCV*, 2026. 2, 3, 4, 6

[61] Kaixin Xiong, Shi Gong, Xiaoqing Ye, Xiao Tan, Ji Wan, Errui Ding, Jingdong Wang, and Xiang Bai. CAPE: Camera view position embedding for multi-view 3D object detection. In *CVPR*, 2023. 3

[62] Tian-Xing Xu, Zi-Xuan Wang, Guangyuan Wang, Li Hu, Zhongyi Zhang, Peng Zhang, Bang Zhang, and Song-Hai Zhang. UCM: Unified modeling of camera control and memory with time-aware positional encoding warping for world models. In *SIGGRAPH Conference Papers*, 2026. 3

[63] Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4DGT: Learning a 4D gaussian transformer using real-world monocular videos. In *NeurIPS*, 2025. 3

[64] Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. YoNoSplat: You only need one model for feedforward 3D Gaussian splatting. In *ICLR*, 2026. 5

[65] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 4