Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

Jonathon Luiten[1,2] Georgios Kopanas[3] Bastian Leibe[2] Deva Ramanan[1]

[1] Carnegie Mellon University, USA [2] RWTH Aachen University, Germany [3] Inria & Université Côte d'Azur, France

luiten@vision.rwth-aachen.de

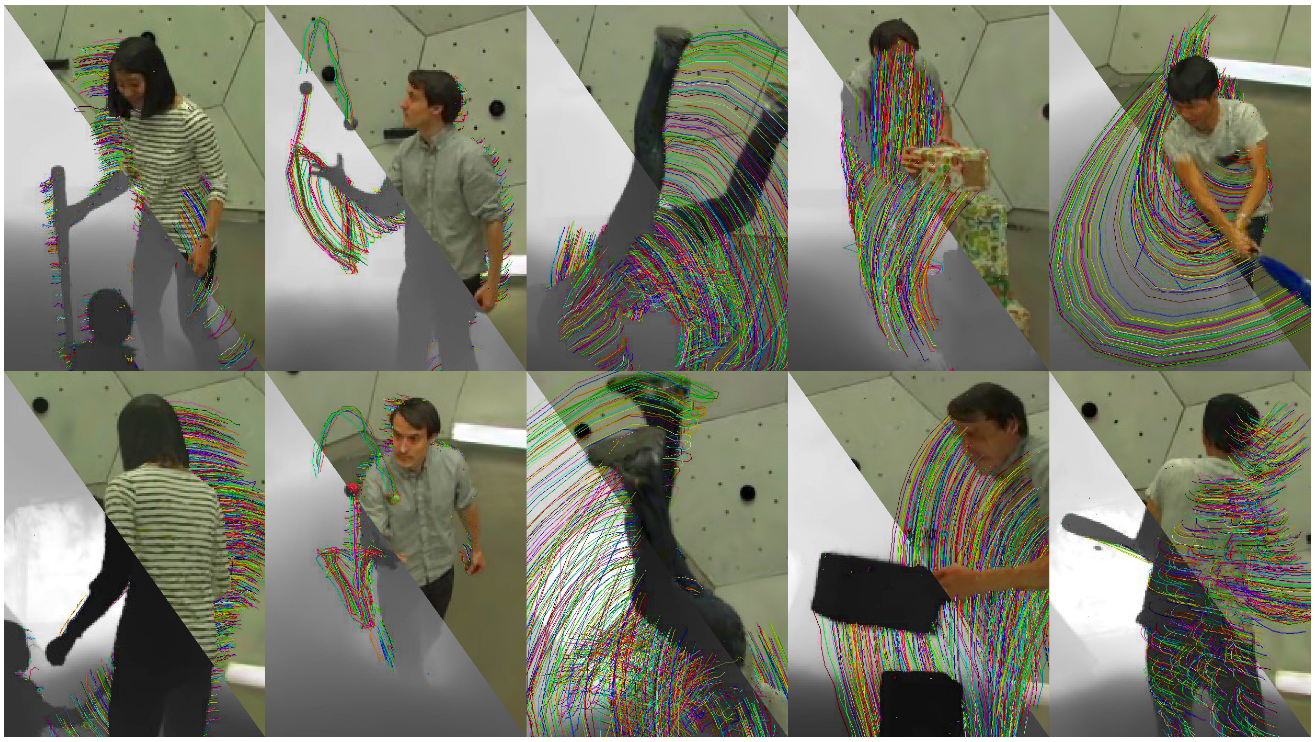

Figure 1. **Persistent Dynamic Novel-View Synthesis and Tracking Results.** Novel-view (unseen) renders of color images and depth maps across 5 scenes (columns) and 2 views (rows) at the same timestep. Each scene is parameterized by 200-300k Dynamic 3D Gaussians which move over time. We render (with occlusions) the 3D trajectories of 2.5% of these over the last 15 timesteps (0.5s). [Videos]

Abstract

We present a method that simultaneously addresses the tasks of dynamic scene novel-view synthesis and six degree-of-freedom (6-DOF) tracking of all dense scene elements. We follow an analysis-by-synthesis framework, inspired by recent work that models scenes as a collection of 3D Gaussians which are optimized to reconstruct input images via differentiable rendering. To model dynamic scenes, we allow Gaussians to move and rotate over time while enforcing that they have persistent color, opacity, and size. By regularizing Gaussians' motion and rotation with local-rigidity constraints, we show that our Dynamic 3D Gaussians correctly model the same area of physical space over time, including the rotation of that space. Dense 6-DOF tracking and dynamic reconstruction emerges naturally from persistent dynamic view synthesis, without requiring any correspondence or flow as input. We demonstrate a large number of downstream applications enabled by our representation, including first-person view synthesis, dynamic compositional scene synthesis, and 4D video editing. [1]

1. Introduction

Persistent dynamic 3D world modeling would be transformative for both discriminative and generative artificial intelligence. On the discriminative side, this would enable a metric-space reconstruction of every part of the scene over time. Modeling where everything currently is, where it has been, and where it is moving, is crucial for many applications including robotics, augmented reality and self-driving. In generative AI, such models could enable new forms of content creation such as easily controllable and editable high resolution dynamic 3D assets for use in movies, video games or the meta-verse. Many such applications require scalable approaches that can be run on high-resolution imagery in real-time. Thus far, no approach has been able to produce photo-realistic reconstructions of arbitrary dynamic scenes with highly-accurate tracks and visually-appealing novel-views, all while being able to be trained quickly and rendered in real-time.

1. Project Website.

In this paper we present such an approach by simultaneously tackling the discriminative tasks of dynamic 3D scene reconstruction and dense non-rigid long-term 6-DOF scene-tracking, while addressing the generative task of dynamic novel-view synthesis. We formulate both of these tasks in an analysis-by-synthesis framework, *i.e.*, we build a persistent dynamic 3D representation of the moving scene that is consistent with all the input observations (images from different timesteps and cameras) and from which tracking emerges as a product of correctly modelling the underlying scene with physically plausible spatial consistency priors.

3D Gaussian Splatting [17] has recently emerged as a promising approach to modelling 3D static scenes. It represents complex scenes as a combination of a large number of coloured 3D Gaussians which are rendered into camera views via splatting-based rasterization. The positions, sizes, rotations, colours and opacities of these Gaussians can then be adjusted via differentiable rendering and gradient-based optimization such that they represent the 3D scene given by a set of input images. In this paper we extend this approach from modelling only static scenes to dynamic scenes.

Our key insight is that we restrict all attributes of the Gaussians (such as their number, color, opacity, and size) to be the same over time, but let their position and orientation vary. This allows our Gaussians to be thought of as a *particle-based* physical model of the world, where oriented particles undergo rigid-body transformations over time. In order to reconstruct such particles from raw camera imagery, we exploit the property that at any time, they can be efficiently splatted onto to any camera viewpoint, allowing them to be optimized with an image reconstruction loss. Crucially, particles allow us to operationalize *physical priors* over their movement that act as regularizers for the optimization: a local rigidity prior, a local rotational-similarity prior, and a long-term local isometry prior. These priors ensure that local neighborhoods of particles move approximately rigidly between timesteps, and that nearby particles remain closeby over all timesteps.

Previous approaches to neural reconstruction of dynamic scenes can be seen as either Eulerian representations that keep track of scene motion at fixed grid locations [5, 10, 36] or Lagrangian representations where an observer follows a particular particle through space and time. We fall in the latter category, but in contrast to prior point-based representations [1, 45], we make use of oriented particles that allow for richer physical priors (as above) and that directly reconstruct the 6-DOF motion of all 3D points, enabling a variety of downstream applications (see Fig. 3 and 7).

We perform experiments using synchronized multi-view video (27 training cameras, 4 testing cameras) from the CMU Panoptic Studio dataset [15]. Our approach achieves 28.7 PSNR on dynamic novel view rendering while rendering at 850 FPS. It is trained on 150 timesteps with 27 training cameras in each timestep in only 2 hours on a single RTX 3090 GPU. Furthermore, our approach results in accurate metric 3D dense non-rigid long-term scene tracking with an average $L2$ error of only 2.21cm in 3D over 150 timesteps, and having an average of 1.57 normalized-pixel error on 2D tracking metrics, which is an order of magnitude (10x) better than previous state-of-the-art. Our method is also able to track the rotation of every 3D point in space, enabling full 6-DOF dense scene tracking. We show visual results in Fig. 1.

An remarkable feature of our approach is that tracking arises exclusively from the process of rendering per-frame images. No optical flow, pose skeletons, or any other form of correspondence information is given as input. Due to its persistent and naturally decomposable nature, Dynamic 3D Gaussians are naturally amenable to a number of creative scene editing techniques such as propagating edits over all timesteps, adding or removing dynamic objects to a scene, or having cameras follow scene elements, as seen in Fig 7. Furthermore, the extremely fast rendering and training time make them much easier to work with than previous approaches for dynamic reconstruction [14], and enable real-time rendering applications.

2. Related Work

In this work, we are interested in solving the combination of dynamic novel-view synthesis, long-term point-tracking and dynamic reconstruction in a unified analysis-by-synthesis framework.

Dynamic Novel-View Synthesis. The general field of novel-view synthesis exploded in popularity with the release of NeRF [25] in 2020. Since then there have been a large number of 'dynamic NeRF' papers extending the idea of fitting a 3D radiance field to the 4D dynamic domain. Most of these use monocular video as input, whereas we focus on using multi-camera capture. Typically these methods belong to one of the following 5 categories:

(a) Methods that fit a separate representation per timestep, and cannot model correspondence over time [2, 43].

(b) Methods that represent the scene using an Eulerian representation on a 4D space-time grid, often with various grid decompositions for efficiency such as planar decomposition or hash functions [5, 10, 36]. Such Eulerian approaches also do not give rise to correspondence over time.

(c) Methods that represent the 3D scene in a canonical timestep and use a deformation field to warp this to the rest of the timesteps [8, 21, 22, 28, 29, 32, 40, 44]. Such methods naturally result in one-directional backward correspondences between each frame and the reference frame, but by default, not the forward correspondences which are required to generate correspondences between any two timesteps. [11] advocates for expensive ad hoc root-finding to determine and evaluate such correspondences. A number of methods specifically parameterize their warp-fields in a way that is easily invertable such as linear-blend-skinning [35, 44], or reversible neural networks [40], in order to obtain correspondences. While this often works quite well [35, 40, 44], requiring a single canonical view for a scene greatly restricts the dynamic representation ability.

(d) Template guided methods [13, 20, 42], which model dynamic scenes in restricted environments where the motion can be modelled by a predefined template e.g. a set of human-pose skeleton transformations. This approach often requires a-priori knowledge of what is to be reconstructed and thus is not a solution for general scenes.

(e) Point-based methods [1, 45], which compared to all of the above categories, hold the most promise for representing dynamic scenes in a way where accurate correspondence over time can emerge due to their natural Lagrangian representation. However, these haven't achieved as much attention because the point-based rendering approaches haven't worked as well as MLP [25] or grid based approaches [26].

While a number of view-synthesis works have used Gaussians for static scenes [17, 18, 39], to the best of our knowledge we are the first to use them to reconstruct dynamic scenes. We build our dynamic Gaussian renderer upon the the static renderer '3D Gaussian splatting' [17] which is currently the state-of-the-art static scene reconstruction algorithm in terms of both accuracy and speed.

Other than rendering accuracy and speed, modeling the dynamic world with Gaussians has a distinct advantage over points as Gaussian's have a notion of 'rotation' so we can use them to model the full 6 degree-of-freedom (DOF) motion of a scene at every point and can use this to construct physically-plausible local rigidity losses.

Long-Term Point Tracking. Traditionally video tracking algorithms have been focused on tracking whole objects (as bounding boxes or segmentation masks) [19, 30, 38], or tracking dense scene points but only between two timesteps (e.g. optical-flow / scene flow) [4, 37]. Recently, a number of approaches have started tackling the task of long-term dense point tracking [6, 7, 12, 16, 40, 47], where every pixel in a video needs to be tracked across every video frame. In this paper, we extend this long-range tracking task to 3D and evaluate it in a multi-camera capture setup. Most of these prior approaches [6, 7, 12, 16, 47] are deep learning based and work by being trained on a large-dataset of ground-truth point tracks (often using synthetic training data). The most similar method to ours is OmniMotion [40] which also fits a dynamic radiance field representation using test-time optimization for the purpose of long-term tracking. They focus on monocular video while we focus on multi-camera capture and as such we can reconstruct tracks in metric-3D while they produce a 'pseudo-3D' representation. However, the largest and most significant difference between our method and OmniMotion is that they require dense optical-flow input between every pair of timesteps as an optimization target, which is incredibly expensive as it scales with the number of frames squared. The optical flow estimates already provide a (noisy and inconsistent) tracking result which is made consistent through dynamic modelling of the scene. In contrast, our method takes no correspondences at all as input and tracking emerges from fitting a persistent representation to the input frames along with physically-based priors.

Dynamic Reconstruction In addition to the radiance-field based approaches listed above, a number of other approaches have tackled the task of 'dynamic reconstruction' [3, 14, 23, 27, 31, 34]. Such approaches either rely on the presence of accurate depth cameras [27, 34], assume ground-truth object point clouds as input [31], or are heavily specific to certain domains e.g. reconstructing moving cars in driving scenes [3, 23]. The closest approach to ours is [14] which also predicts long-term 3D point tracks and also uses the Panoptic Studio data for evaluation. However this approach requires 480 input cameras (in contrast we use 27) and also requires as input pre-computed optical-flow correspondences, which it then lifts into 3D trajectories.

3. Method

Overview. Given a set of images from different timesteps and different cameras ($\mathcal{I}_{t,c}$), along with each camera's respective intrinsic (K_c) and extrinsic ($E_{t,c}$) matrices, our approach reconstructs the dynamic 3D scene ($\mathcal{S}$) observed by these cameras in a temporally persistent manner.

This reconstruction is performed via test-time optimization and no further training data other than the test scene is used. The reconstruction is performed temporally online, *i.e.*, one timestep of the scene is reconstructed at a time with each one being initialized using the previous timestep's representation. The first timestep acts as an initialization for our scene where we optimize all properties, and then fix all for the subsequent timesteps except those defining the motion of the scene. Each timestep is trained via gradient based optimization using a differentiable renderer ($\mathcal{R}$) to render the scene at each timestep into each of the training cameras.

$$\hat{\mathcal{I}}_{t,c} = \mathcal{R}(\mathcal{S}_t, K_c, E_{t,c})$$

The renderings $\hat{\mathcal{I}}_{t,c}$ are compared to the input images $\mathcal{I}_{t,c}$, and the parameters of $\mathcal{S}$ are iteratively updated using automatic differentiation in order to decrease the error between $\hat{\mathcal{I}}_{t,c}$ and $\mathcal{I}_{t,c}$. After convergence, the representation $\mathcal{S}_t$ is a 3D reconstruction of the scene given by each of the training cameras $\{\mathcal{I}_{t,c}, K_c, E_{t,c}\}$ for this timestep. By choosing a suitable representation for $\mathcal{S}$ and applying physically-based regularization losses during optimization, we can ensure that all the $\mathcal{S}_t$ are temporally consistent with one another and that there exists a one-to-one correspondence between every 3D point in every timestep, along with their corresponding changes in 3D rotation. In this way, tracking

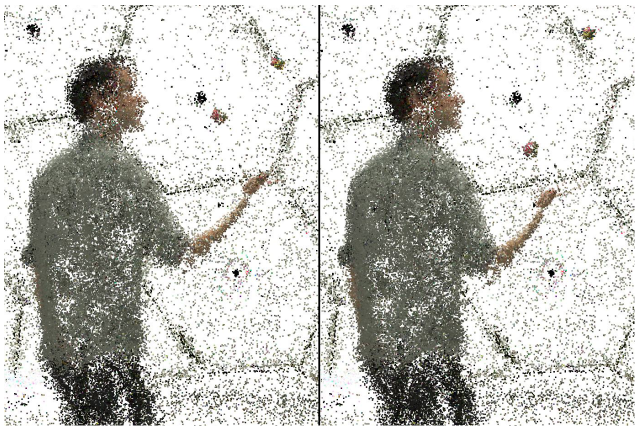

Figure 2. **Gaussian Centers.** Point-cloud of colored centers in contiguous timesteps, showing how they model scene geometry and move over time. [Videos]

emerges from persistent dynamic view synthesis.

Dynamic 3D Gaussians. Our dynamic scene representation ($\mathcal{S}$) is parameterized by a set of Dynamic 3D Gaussians, each of which has the following parameters:

1) a 3D center for each timestep (x_t, y_t, z_t).
2) a 3D rotation for each timestep parameterized by a quaternion (qw_t, qx_t, qy_t, qz_t).
3) a 3D size in standard deviations (consistent over all timesteps) (sx, sy, sz)
4) a color (consistent over all timesteps) (r, g, b)
5) an opacity logit (consistent over all timesteps) (o)
6) a background logit (consistent over timesteps) (bg)

This gives a total of $7t + 8$ parameters for each Gaussian. In our experiments, scenes are represented by between 200-300k Gaussians, of which only 30-100k usually are not part of the static background. While the code contains the ability to represent view-dependent color using spherical harmonics, we turn this off in our experiments for simplicity.

Each Gaussian can be thought of as softly representing an area of 3D physical space which is occupied by solid matter. Each Gaussian influences a point in physical 3D space (p) according to the standard (unnormalized) Gaussian equation weighted by its opacity:

$$f_{i,t}(p) = \text{sigm}(o_i) \exp\left(-\frac{1}{2}(p - \mu_{i,t})^T \Sigma_{i,t}^{-1}(p - \mu_{i,t})\right)$$

Where $\mu_{i,t} = \begin{bmatrix} x_{i,t} & y_{i,t} & z_{i,t} \end{bmatrix}^T$ is the center of each Gaussian i at timestep t, and $\Sigma_{i,t} = R_{i,t} S_i S_i^T R_{i,t}^T$ is the covariance matrix of Gaussian i at timestep t, given by combining the scaling component $S_i = \text{diag}\left(\begin{bmatrix} sx_i & sy_i & sz_i \end{bmatrix}\right)$, and the rotation component $R_{i,t} = \text{q2R}\left(\begin{bmatrix} qw_{i,t} & qx_{i,t} & qy_{i,t} & qz_{i,t} \end{bmatrix}\right)$, where q2R() is the formula for constructing a rotation matrix from a quaternion. sigm() is the standard sigmoid function.

Each Gaussian's influence (f) is both inherently local (being able to represent a small area of space), while also

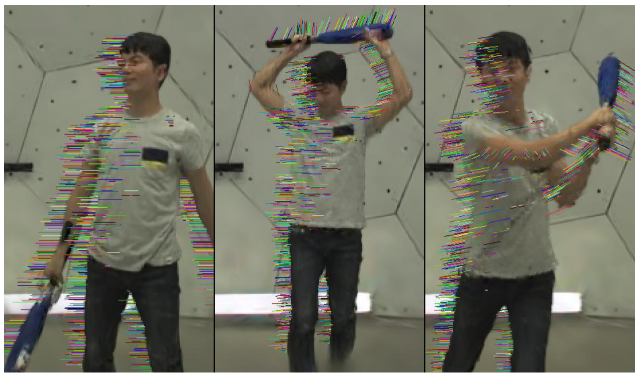

Figure 3. **Relative Rotation Tracking.** 1st panel: Left-facing coloured vectors are attached to 3% of Gaussians in the first frame. 2nd and 3rd panels: These vectors move and rotate along with the Gaussians they are attached to, showing that our approach correctly models 6-DOF motion. [Videos]

theoretically having infinite extent, such that gradients can flow to them even from a long distance, which is crucial for gradient-based tracking-by-differentiable rendering, as Gaussians which may currently be in the wrong 3D location need to get gradients pushing them towards moving to the correct 3D location through the differentiable renderer. The softness of this Gaussian representation also means that Gaussians typically need to significantly overlap in order to represent a physically solid object. As well as physical density, each Gaussian contributes its own color (r, g, b) to each of the 3D points it influences.

By fixing the size/opacity/color of the Gaussians across time, each Gaussian should represent the same physical aspect of space, even as this space dynamically moves through time. To represent this motion, each Gaussian has a center location and rotation that can move with time, enabling full dense non-rigid 6-DOF tracking of a whole scene.

We visualize the trajectories of these Gaussians in Fig 1, the locations of the Gaussian's centers in Fig 2, and the change in rotation over time in Fig 3.

Differentiable Rendering via Gaussian 3D Splatting. In order to optimize the parameters of our Gaussians to represent the scene, we need to render the Gaussians into images in a differentiable manner. In this work we use the differentiable 3D Gaussian renderer from [17] and extend its use to dynamic scenes. This works by splatting 3D Gaussians into the image plane by approximating the projection of the integral of the influence function f along the depth dimension of the 3D Gaussian into a 2D Gaussian influence function in pixel coordinates. The center of the Gaussian is splatted using the standard point rendering formula:

$$\mu^{\text{2D}} = K\left((E\mu)/(E\mu)_z\right)$$

where the 3D Gaussian center μ is projected into a 2D image by multiplication with the world-to-camera extrinsic matrix E, z-normalization, and multiplication by the intrinsic projection matrix K. The 3D covariance matrix is splat-

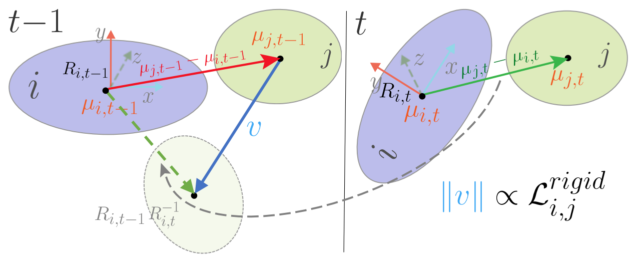

Figure 4. **Local Rigidity Loss.** For each Gaussian i, nearby Gaussians j should move in a way that follows the rigid-body transform of the coordinate system of i between timesteps.

ted into 2D using the formula from [48]:

$$\Sigma^{\text{2D}} = JE\Sigma E^T J^T$$

where J is the Jacobian of the point projection formula above, i.e. $\partial\mu^{\text{2D}}/\partial\mu$.

The influence function f can now be evaluated in 2D for each pixel for each Gaussian. The influence of all Gaussians on this pixel can be combined by sorting the Gaussians in depth order and performing front-to-back volume rendering using the Max [24] volume rendering formula (the same as is used in NeRF [25]):

$$C_{\text{pix}} = \sum_{i \in \mathcal{S}} c_i f_{i,\text{pix}}^{\text{2D}} \prod_{j=1}^{i-1} (1 - f_{j,\text{pix}}^{\text{2D}})$$

where the final rendered color (C_{pix}) for each pixel is a weighted sum over the colors of each Gaussian ($c_i = \begin{bmatrix} r_i & g_i & b_i \end{bmatrix}^T$), weighted by the Gaussian's influence on that pixel $f_{i,\text{pix}}^{\text{2D}}$ (the equivalent of the formula for f_i in 3D except with the 3D means and covariance matrices replaced with the 2D splatted versions), and down-weighted by an occlusion (transmittance) term taking into account the effect of all Gaussians in front of the current Gaussian.

The implementation of [17] uses a number of graphics and CUDA optimization techniques to achieve incredibly fast rendering speeds (*e.g.*, 850 FPS for our scenes), which therefore also enables very fast training.

Physically-Based Priors. We find that just fixing the color, opacity and size of Gaussians is not enough on its own to generate long-term persistent tracks, especially across areas of the scene where there is a large area of near uniform colour. In such situation the Gaussians move freely around the area of similar colour as there is no restriction on them doing so. Since we are trying to model physically moving scenes, it makes sense to look to non-rigid physical modelling for inspiration on how to regularize the optimization procedure to be physically plausible and give correct long-term tracking results. We introduce three regularization losses, short-term local-rigidity $\mathcal{L}^{\text{rigid}}$ and local-rotation similarity $\mathcal{L}^{\text{rot}}$ losses and a long-term local-isometry loss. The most important of these is the local-rigidity loss $\mathcal{L}^{\text{rigid}}$, defined as:

$$\mathcal{L}_{i,j}^{\text{rigid}} = w_{i,j} \left\| (\mu_{j,t-1} - \mu_{i,t-1}) - R_{i,t-1} R_{i,t}^{-1} (\mu_{j,t} - \mu_{i,t}) \right\|_2$$

$$\mathcal{L}^{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} \mathcal{L}_{i,j}^{\text{rigid}}$$

This states that, for each Gaussian i, nearby Gaussians j should move in a way that follows the rigid-body transform of the coordinate system of i between timesteps. See Fig 4 for a visual explanation.

Since we are performing online optimization, all of $\mu_{i,t-1}, R_{i,t-1}, \mu_{j,t-1}$ are fixed, and we are optimizing $\mu_{i,t}, R_{i,t}, \mu_{j,t}$ to ensure that they match the values in $t-1$ up to the rigid body transformation defined by the change in Gaussian i's own coordinate system. e.g. if i rotates, then j needs to translate (in it's own coordinate system) in a way that in equivalent to rotating around the center of i. This loss also applies the other way, forcing the rotation to match the translation, such that we obtain accurate rotation (6-DOF) tracking for every dense point in space, even though we only optimize to match the rendered images, which isn't possible with a point-based representation.

We restrict the set of Gaussians j to be the k-nearest-neighbours of i (k=20), and weight the loss by the a weighting factor for the Gaussian pair:

$$w_{i,j} = \exp\left(-\lambda_w \left\|\mu_{j,0} - \mu_{i,0}\right\|_2^2\right)$$

which is an (unnormalized) isotropic Gaussian weighting factor. We set λ_w to 2000, which gives a standard deviation of ∼2.2cm, and calculate this with the distance between the Gaussian centers in the first timestep and fix it over the rest of the timesteps. This results in the rigidity loss only being enforced locally, while still allowing global non-rigid reconstruction.

The rigidity loss is both necessary and adequate on it's own to achieve good results. Since the rigidity loss is applied on all points it is applied in both directions between any pair i and j and thus implicitly enforces i and j to have the same rotation, however we found better convergence if we explicitly force neighbouring Gaussians to have the same rotation over time:

$$\mathcal{L}^{\text{rot}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w_{i,j} \left\| \hat{q}_{j,t} \hat{q}_{j,t-1}^{-1} - \hat{q}_{i,t} \hat{q}_{i,t-1}^{-1} \right\|_2$$

where $\hat{q}$ is the normalized quaternion representation of each Gaussian's rotation, which enables smooth optimization. We use the same set of k-nearest-neighbours and weighting function as before.

We apply $\mathcal{L}^{\text{rigid}}$ and $\mathcal{L}^{\text{rot}}$ only between the current timestep and the directly preceding timestep, thus only enforcing these losses over short-time horizons. Which sometimes causes elements of the scene to drift apart, thus we apply a third loss, the isometry loss, over the long-term:

$$\mathcal{L}^{\text{iso}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w_{i,j} \left| \left\|\mu_{j,0} - \mu_{i,0}\right\|_2 - \left\|\mu_{j,t} - \mu_{i,t}\right\|_2 \right|$$

This is a weaker constraint than $\mathcal{L}^{\text{rigid}}$ in that instead of enforcing the positions between two Gaussians to be the same it only enforces the distances between them to be the same.

Optimization Details. Each timestep of a scene is optimized one-at-a time. During the first timestep, all parameters of the Gaussians are optimized. After the first timestep the size, color, opacity, and background logit are fixed and only the position and rotation are updated. Thus, our approach can be seen as first performing static reconstruction of the first frame, followed by long-term dense 6-DOF tracking throughout the remaining frames.

Following [17], in the first timestep we initialize the scene using a coarse point cloud that could be obtained from running colmap, but instead we use available sparse samples from depth cameras. Note that these depth values are only used for initializing a sparse point cloud in the first timestep and are not used at all during optimization.

We use the densification from [17] in the first timestep in order to increase the density of Gaussians and achieve a high quality reconstruction. For the rest of the frames the number of Gaussians is fixed and the densification is turned off.

We fit the 3D scene in the first frame for 10000 iterations (around 4 minutes) using 27 training cameras, where each iteration renders a single but complete image. For each timestep after that we use 2000 iterations (around 50 seconds), for a total of 2 hours for 150 timesteps. At the beginning of each new timestep we initialize the estimated Gaussian center positions and rotation quaternion parameters by using forward estimate based on a velocity estimated from the current position minus the previous position, and do the same for the quaternion using normalized quaternions (e.g. we also re-normalize the quaternion representation). We find this to be quite important to getting good results. We also reset the Adam first and second order momentum parameters at the start of each timestep.

In our test scenes the subject's shirt colours (grey) are very similar to the background. We noticed that often the shirt was being mis-tracked as it was confused with the background, while more contrastive elements like pants and hair were being tracked correctly. To increase the contrast between foreground and background parts of the scene we also render a foreground/background mask and apply a background segmentation loss $\mathcal{L}^{Bg}$ against a pseudo-ground-truth background mask, which we can easily obtain by differencing with an image from the dataset where no foreground objects are presents. We also directly apply a loss that background points shouldn't move or rotate, and restrict the above rigidity, rotation and isometry losses to only operate over foreground points to improve efficiency. This also ensures that these losses are never enforced between foreground parts of the scene and the static floor.

Finally, since we are optimizing over 27 different training cameras, each of which has different camera properties such as white balance, exposure, sensor sensitivity, and color calibration, these factors contribute to variations in color representation across images. Thus we naively model these differences by simply optimizing a scale and offset parameter for each colour channel for each camera separately. We optimize these only over the first timestep and then fix these for the rest of the timesteps.

Tracking with Dynamic 3D Gaussians. After we have trained our Dynamic 3D Gaussian scene representation we can use it to obtain dense correspondence for any point in a 3D scene. To determine the correspondence of any point in 3D space p across timesteps, we can linearize the motion-space by simply taking the point's location in the coordinate system of the Gaussian that has the most influence $f(p)$ over this point (or the static background coordinate system if $f(p) < 0.5$ for all Gaussians). There is now a well-defined and invertible one-to-one mapping for all points in space across all timesteps giving dense correspondences.

We can use this same idea to track any 2D pixel-location from any input or novel view into any other timestep or view. To do so we first have to determine the 3D point corresponding to an input pixel. We can render out depth-maps for any view by using the Max [24] rendering equation for Gaussians [17] but replacing the colour component of each Gaussian with the depth of that Gaussian's center, as seen in Figure 1. A pixel's 3D location can be found via unprojection, the highest influence Gaussian determined, tracked and then projected into a new camera frame.

4. Experiments

Dataset Preparation. We prepare a dataset which we call `PanopticSports`. We take six sub-sequences from the sports sequence of the Panoptic Studio dataset [15]. Each of our six sequences contain interesting motions and objects which we name them after: `juggle`, `box`, `softball`, `tennis`, `football` and `basketball`. We produce visual results for 3 more sequences, `handstand`, `sway` and `lift` but don't include them in evaluation results as they don't have 3D track ground-truth available. For each sequence we obtain 150 frames at 30 FPS, from the set of the HD cameras. There are 31 cameras, which we split into 27 training and 4 testing cameras (cam 0, 10, 15 and 30 are test). Cameras are temporally aligned and have accurate intrinsics and extrinsics provided. The cameras are positioned roughly in a hemisphere around an area of interest in the middle of a capture studio dome. Fig 1 shows an example.

We undistort the images from each camera using the provided distortion parameters, and resize each image to be 640x360. We create an initial point cloud for the first timestep to initialize the Gaussians by taking the points from 10 available depth cameras, synchronized to the current timestep, subsample these depth maps by a factor of 2 in both dimensions and obtain an initial colour by projecting these points into the nearest training camera. Points that do

Task	Metrics	Method	Juggle	Boxes	Softball	Tennis	Football	Basketball	Mean
View Synthesis	PSNR↑	3GS-O [17]	28.19	28.74	**28.77**	28.03	**28.49**	27.02	28.21
		Ours	**29.48**	**29.46**	28.43	**28.11**	**28.49**	**28.22**	**28.7**
	SSIM↑	3GS-O [17]	0.91	**0.91**	**0.91**	0.90	0.90	0.89	0.90
		Ours	**0.92**	0.91	0.91	**0.91**	**0.91**	**0.91**	**0.91**
	LPIPS↓	3GS-O [17]	**0.15**	**0.15**	**0.14**	**0.16**	**0.16**	**0.18**	**0.16**
		Ours	**0.15**	0.17	0.19	0.17	0.19	**0.18**	**0.17**
3D Tracking	3D MTE↓	3GS-O [17]	32.81	39.95	64.94	75.54	45.57	76.71	55.9
		Ours	**1.90**	**1.97**	**2.02**	**2.33**	**2.45**	**2.56**	**2.21**
	3D δ↑	3GS-O [17]	13.6	3.5	5.9	4.2	9.8	3.5	6.8
		Ours	**77.2**	**75.9**	**70.3**	**69.0**	**69.4**	**66.3**	**71.4**
	3D Surv↑	3GS-O [17]	56.3	60.8	37.2	16.9	59.6	31.9	43.8
		Ours	**100**	**100**	**100**	**100**	**100**	**100**	**100**
2D Tracking	2D MTE↓	3GS-O [17]	23.86	29.88	51.6	58.15	35.15	64.29	43.8
		PIPS [12]	5.76	8.42	13.3	21.0	23.2	22.6	15.7
		Ours	**1.54**	**1.42**	**1.69**	**1.36**	**1.48**	**1.93**	**1.57**
	2D δ↑	3GS-O [17]	17.1	10.5	8.9	6.5	15.0	7.2	10.9
		PIPS [12]	55.9	39.5	37.0	28.4	43.5	33.2	39.6
		Ours	**80.4**	**82.5**	**77.3**	**80.2**	**79.7**	**73.9**	**78.4**
	2D Surv↑	3GS-O [17]	71.3	74.4	42.7	23.0	69.6	47.1	54.7
		PIPS [12]	91.6	61.3	88.6	72.2	79.8	77.6	79.0
		Ours	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Table 1. **Results on our prepared PanopticSports dataset**. See text for details on the dataset, metrics, tasks and methods.

Method	PSNR↑	SSIM↑	LPIPS↓
TiNeuVox-S [9]	26.64	0.92	0.14
TiNeuVox [9]	27.28	0.91	0.13
InstantNGP [26]	24.69	0.91	0.12
Particle-NeRF [1]	27.47	0.94	0.08
Ours	**39.49**	**0.99**	**0.02**

Table 2. **Result on the Particle-NeRF dataset**. See text for details on the dataset, metrics, tasks and methods.

not project into any training camera are discarded. We obtain pseudo-ground-truth foreground-background segmentation masks by simply doing frame-differencing between each frame and a reference frame from the dataset for each camera when no foreground objects are present.

We prepare ground-truth trajectories for 2D and 3D tracking by taking the high-quality facial and hand key-point annotations that are available for the scene [33]. For each person in each scene (four scenes have one person, two have two), we take one random face key-point and one random hand key-point from each hand. We manually verify the accuracy of these trajectories by projecting them into the images and viewing them, and remove three that are not accurate. This leaves us with a total of 21 ground truth 3D trajectories.

For 2D tracking, we use the camera-visibility labels to determine if the first point in each 3D trajectory is visible in each camera, and add these videos and projected points to our 2D tracking evaluation. Each 3D point is visible in around 18 cameras for a total of 371 2D ground-truth tracks.

Evaluation Metrics. We evaluate novel-view synthesis on the hold-out 4 camera views across all 150 timesteps for the 6 sequences. We use the standard PSNR, SSIM and LPIPS metrics [41, 46]. For 2D long-term point tracking we use the metrics from the recent point-odyssey benchmark [47]: median trajectory error (MTE), position accuracy (δ), and survival rate. For 3D long-term point tracking there is no prior relevant work, so we decide adapt the 2D metrics from [47] to the 3D domain, except in terms of centimeters in 3-dimensions instead of normalized-pixels in 2D. E.g. MTE is reported as error in cm. δ is calculated at 1, 2, 4, 8 and 16cm thresholds and survival is measuring trajectories that are within 50cm of the ground-truth.

Comparisons. For all three tasks of View-Synthesis, 3D tracking and 2D tracking we compare our Dynamic 3D Gaussian method to the original 3D Gaussian Splatting [17] which we build upon. We run this in a similar online mode that we run our method for the same number of iteration steps, and thus we call this 3GS-O. We also perform comparisons on the dataset of Particle-NeRF [1], which includes 20 train and 10 test cameras, and contains much simpler synthetic scenes with both simple geometry and motion. On this benchmark we compare to Particle-NeRF [1], Instant-NGP [26], and TiNeuVox [9]. The benchmark task for this dataset [1] is defined as one in which at each timestep methods are allowed a certain number of ray renders and back-propagation steps. Since our approach doesn't work in the same way we are not able to compare exactly using these criterion. Instead as a reasonably fair criterion we run train our method for each timestep for no more wall clock time than Particle-NeRF does, e.g. 200ms per timestep.

For 3D long-term point tracking we have not found any further methods to compare against for this relatively new

Exp #	Description	Additions						View Synthesis			3D Tracking		2D Tracking	
		$\mathcal{L}_{\text{Rigid}}$	$\mathcal{L}_{\text{Rot}}$	$\mathcal{L}_{\text{Iso}}$	$\mathcal{L}_{\text{Bg}}$	Fix	Prop	PSNR↑	SSIM↑	LPIPS↓	3D MTE↓	3D δ↑	2D MTE↓	2D δ↑
0	Ours - Full	✓	✓	✓	✓	✓	✓	**29.48**	**0.92**	**0.15**	**1.90**	**77.2**	**1.54**	**80.4**
1	No $\mathcal{L}_{\text{Rigid}}$	✗	✓	✓	✓	✓	✓	28.51	0.91	0.17	4.32	55.2	3.80	58.7
2	No $\mathcal{L}_{\text{Rot}}$	✓	✗	✓	✓	✓	✓	29.43	**0.92**	0.16	1.91	76.6	1.55	79.8
3	No $\mathcal{L}_{\text{Iso}}$	✓	✓	✗	✓	✓	✓	29.36	**0.92**	0.16	1.93	76.7	1.72	79.3
4	No $\mathcal{L}_{\text{Bg}}$	✓	✓	✓	✗	✓	✓	24.14	0.82	0.34	8.46	60.0	6.40	63.2
5	No Param Fixing	✓	✓	✓	✓	✗	✓	27.14	0.89	0.22	30.7	57.7	19.15	58.8
6	No Forward Prop	✓	✓	✓	✓	✓	✗	28.48	0.91	0.16	6.32	54.87	5.4	57.7
7	3GS-O [17]	✗	✗	✗	✗	✗	✗	28.19	0.90	**0.15**	32.81	13.6	23.86	17.1

Table 3. **Ablation results on the Juggle scene of PanopticSports.** See text for details on the dataset, metrics, tasks and methods.

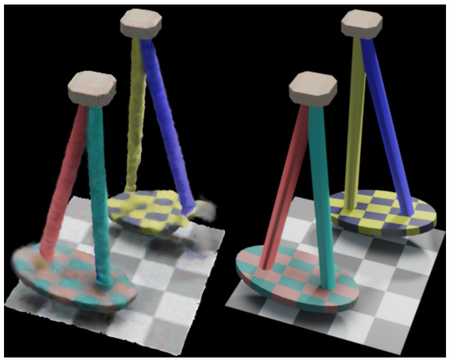

Figure 5. **Visual comparison.** Comparing Particle-NeRF (left) and Ours (right) on the Particle-NeRF dataset.

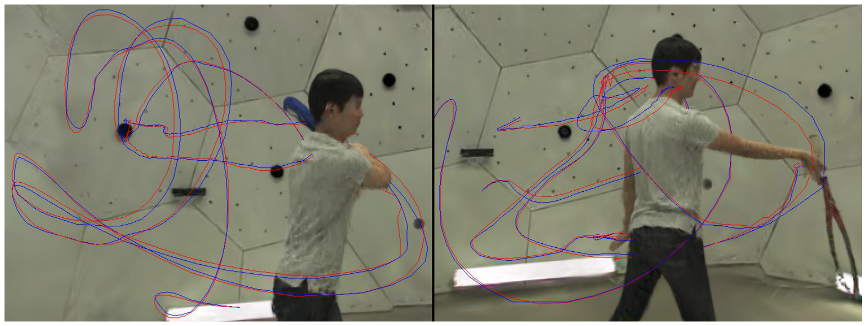

Figure 6. **Ground-truth Comparison.** Comparing our result (blue) to the ground-truth (red). Where ground-truth is noisy, our result may be more accurate. [Videos]

task. The two methods we would like most to compare to are OmniMotion [40] adapted to multi-camera metric space and MAP Visibility Estimation [14] since they seem like the most promising competitors. However, since no code is currently available for either we are unable to run such comparisons. For 2D tracking we compare against 2D long-term tracking approaches of which the canonical example is PIPs [12]. We provide comparison results with PIPs to show a comparison against a learnt method that was trained specifically for long-term 2D point tracking, and is also one of the state-of-the-art approaches for this task. Although the comparison isn't entirely fair to either method. E.g. our method sees all 27 training cameras while PIPs only sees the one that needs to be tracked in. However on the other hand PIPs was trained on 13085 training videos where ground-truth tracks were provided while our method has never seen any ground-truth tracks, nor any other video data other than the camera views for the current test scene. Regardless it is still a good test of our method to compare against such 2D point tracking methods.

PanopticSports Results. We present the results on our prepared `PanopticSports` dataset in Table 1. Our approach achieves good scores on all three novel-view synthesis metrics, with a final PSNR score of 28.7. Compared to the original 3D Gaussian Splatting [17] we achieve better PSNR and SSIM scores (although slightly worse LPIPS) across all scenes by correctly modelling the temporal consistency of the dynamic scene. Visual examples of the high-quality novel-view synthesis results can be found in Fig. 1.

In terms of 3D tracking, our method achieves outstanding results with a median trajectory error of only 2.21cm across all trajectories in all scenes. This is less than the width of a wrist across 150 timesteps of 3D tracking through extremely complex and fast motions (see Fig. 1). Our method also has 100% survival rate across all sequences never losing the point to track and an accuracy value of 71.4. The original Gaussian Splatting [17] approach doesn't correctly track points in 3D at all with with a much higher 55.9cm median trajectory error. A visual comparison of our method compared to the ground-truth can be found in Fig 6.

When measuring the 2D tracking ability of our method and comparing it to the 2D tracker PIPs [12] we can see where our method really shines. Although it's not a 1-to-1 fair comparison, by comparing our numbers against the numbers of a state-of-the-art tracker we can accurately gauge the performance of our approach. We achieve a 10x lower median trajectory error of only 1.57 pixels compares to PIPs 15.7, have a much higher trajectory accuracy of 78.4 compared to 39.6, and a 100% survival rate compared to 79%.

Overall these results show that our method performs excellently on the tasks of novel-view-synthesis, as well as both 2D and 3D tracking.

Particle-NeRF Dataset Results. On the Particle-NeRF dataset our method achieves almost perfect scores for all of PSNR, SSIM and LPIPS. This is due to the dataset being relatively simple in terms of simple synthetic objects and simple motions. A visual comparison between our method's results and those of Particle-NeRF can be found in Fig 5.

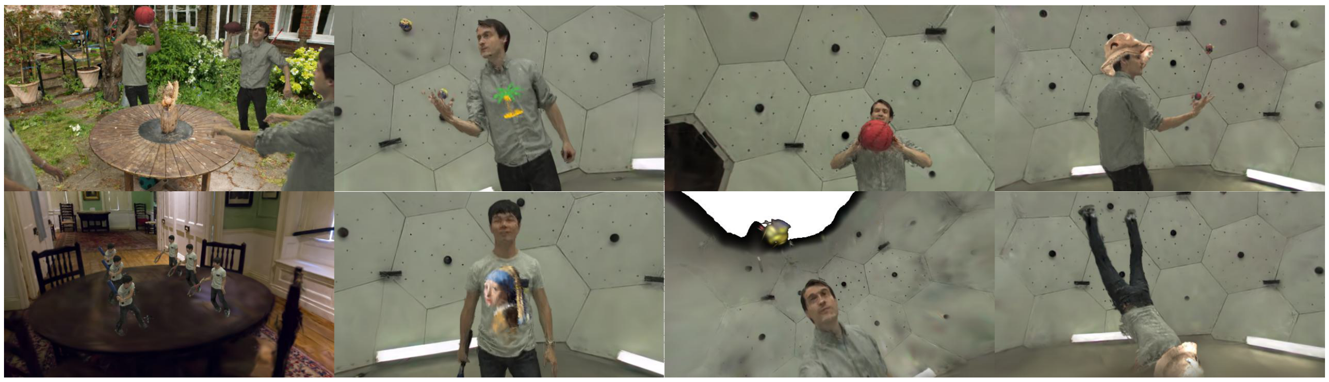

Figure 7. **Augmented reality applications enabled by Dynamic 3D Gaussians.** Left: Dynamic objects can easily be removed from scenes, duplicated and added together with other dynamic objects to new scenes. Center Left: Image edits can be lifted to 3D and then automatically propagated across time. Center Right: Camera views can be attached to dynamic Gaussians that move as the scene moves, e.g. first-person view (above) or juggling-ball's view (below). Right: Objects can be scanned and added to dynamic scenes in a way that follow the scene. E.g. the hat stays correctly on the person with the correct translation and rotation as he does a handstand. [Videos]

Ablation Study. In Table 3 we show results on the Juggle sequence of our `PanopticSports` dataset ablating the various different components of our approach. We identify 6 key components of our approach which are above and beyond that of the original 3D Gaussian splatting approach [17], as described in Section 3. We evaluate the effect of removing each of these components from our final method one-at-a-time as well as the effect of removing them all at once, which is then just the method from [17] run in an online mode over different timesteps.

For view synthesis, the original 3GS-O already works extremely well, but by correctly modelling the motion of components in the scene our full method is able to achieve a boost of 1.3 PSNR. For tracking the original doesn't accurately track the scene at all, but ours performs very accurately. In terms of key-components required for these results, all of the rigidity loss, the background segmentation loss, the colour/opacity/size parameter fixing, and the forward propagation for timestep initialization are key to obtaining both good tracking results and improvement in view-synthesis results. The rotation loss and isometric loss only provide very small improvement in the measured metrics, however visually we found the reconstruction results to be much more coherent and correct with both of these on over turning either off.

Further Applications. Our Dynamic 3D Gaussian approach also leads itself nicely to being used for editing of dynamic 3D scenes. Because Gaussians are independent, subsets of them can easily be added or removed from scenes to create all sort of interesting effects. *E.g.*, creating realistic renders by combining multiple different dynamic components from different scenes and different backgrounds. We can also very easily propagate edits over time. *E.g.*, logos can be added to surfaces in a single frame, and the colors of the Gaussians can be updated to reflect this change. Such edits will automatically propagate to all other frames of a video. Finally, because we are performing full 6-DOF tracking we can take advantage of this for all sorts of visual effects, for example we could put a camera at 'first person view' by attaching it to any particular Gaussian and it will follow where that Gaussian moves and rotates over time. The same can be done for adding objects to the scene that can move and rotate along with the Gaussians they are attached to. We show examples of all of these creative applications in Fig 7.

5. Conclusion and Limitations

Limitations While our method achieves excellent results it is not without limitations. For example, by design our method is only able to track parts of scenes that are visible in the initial frame. It would completely fail to reconstruct new objects entering the scene. Our method also requires a multi-camera setup and does not work off-the-shelf on monocular video. We believe that these limitations are the seeds of exciting future research directions to build upon and extend our Dynamic 3D Gaussian representation.

Conclusion In this work, we have introduced a novel method for dynamic 3D scene modeling, view synthesis, and 6-DOF tracking that has relevant applications across various domains, including entertainment, robotics, VR and AR. Utilizing Gaussian elements to model dynamic scenes, our approach uniquely captures movements and rotations, consistent with physical properties. The implications of our method extend beyond the immediate results, offering new avenues for real-time rendering and creative scene editing. Our approach, characterized by efficiency and accuracy, sets a promising direction for future research and practical applications in 3D modeling and tracking, underscoring the potential for further innovation in these fields.

Acknowledgements: Jonathon Luiten's research was funded, in parts, by NRW Verbundprojekt WestAI (01IS22094D). EP/N019474/1. The authors would like to thank Joanna Materzynska, Leonid Keselman, Dinesh Reddy, Jonas Schult and Adam Harley for helpful discussions.

References

[1] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: Particle based encoding for online neural radiance fields in dynamic scenes. *arXiv:2211.04041*, 2022. 2, 3, 7

[2] Aayush Bansal and Michael Zollhoefer. Neural pixel composition for 3d-4d view synthesis from multi-views. In *CVPR*, pages 290–299, 2023. 2

[3] Ioan Andrei Bârsan, Peidong Liu, Marc Pollefeys, and Andreas Geiger. Robust dense mapping for large-scale dynamic environments. In *ICRA*, pages 7510–7517. IEEE, 2018. 3

[4] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995. 3

[5] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, pages 130–141, 2023. 2

[6] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens Continente, Kucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. In *NeurIPS Datasets Track*, 2022. 3

[7] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. *arXiv:2306.08637*, 2023. 3

[8] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, pages 14304–14314. IEEE Computer Society, 2021. 2

[9] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 7

[10] 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*, pages 12479–12488, 2023. 2

[11] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *Advances in Neural Information Processing Systems*, 35:33768–33780, 2022. 2

[12] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, 2022. 3, 7, 8

[13] Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, and Matthias Nießner. Humanrf: High-fidelity neural radiance fields for humans in motion. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 3

[14] Hanbyul Joo, Hyun Soo Park, and Yaser Sheikh. Map visibility estimation for large-scale dynamic 3d reconstruction. In *CVPR*, pages 1122–1129, 2014. 2, 3, 8

[15] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015. 2, 6

[16] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv:2307.07635*, 2023. 3

[17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023. 2, 3, 4, 5, 6, 7, 8, 9

[18] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *ECCV*, 2022. 3

[19] Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Cehovin, Alan Lukežič, Ondrej Drbohlav, Jani Käpylä, Gustav Häger, Song Yan, Jinyu Yang, Zhongqun Zhang, and Gustavo Fernández. The ninth visual object tracking vot2021 challenge results. In *ICCVW*, pages 2711–2738, 2021. 3

[20] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhofer, Jurgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. In *ECCV*, 2022. 3

[21] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, pages 6498–6508, 2021. 2

[22] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *CVPR*, pages 13–23, 2023. 2

[23] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 5(2):1803–1810, 2020. 3

[24] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 5, 6

[25] 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 *ECCV*, 2020. 2, 3, 5

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 3, 7

[27] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015. 3

[28] 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*, pages 5865–5874, 2021. 2

[29] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021. 2

[30] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 3

[31] Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic point fields. *arXiv:2304.02626*, 2023. 3

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021. 2

[33] Tomas Simon, Hanbyul Joo, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. *CVPR*, 2017. 7

[34] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *CVPR*, pages 1386–1395, 2017. 3

[35] Chonghyuk Song, Gengshan Yang, Kangle Deng, Jun-Yan Zhu, and Deva Ramanan. Total-recon: Deformable scene reconstruction for embodied view synthesis. In *ICCV*, 2023. 3

[36] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *CVPR*, pages 12375–12385, 2023. 2

[37] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 722–729. IEEE, 1999. 3

[38] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *CVPR*, pages 7942–7951, 2019. 3

[39] Angtian Wang, Peng Wang, Jian Sun, Adam Kortylewski, and Alan Yuille. Voge: a differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. In *ICLR*, 2022. 3

[40] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. *arXiv:2306.05422*, 2023. 2, 3, 8

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[42] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *CVPR*, pages 16210–16220, 2022. 3

[43] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *CVPR*, 2021. 2

[44] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, pages 2863–2873, 2022. 2, 3

[45] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–12, 2022. 2, 3

[46] 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. 7

[47] Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023. 3, 7

[48] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 5