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

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. 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.

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.

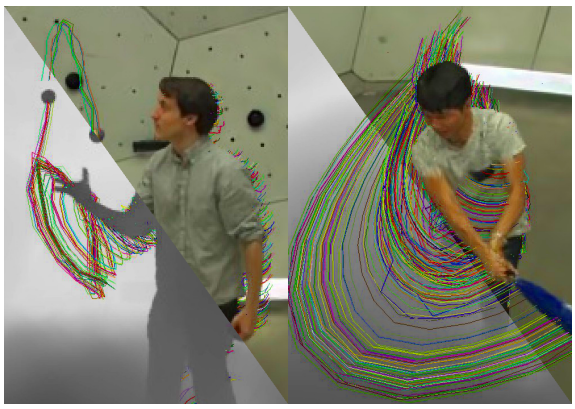

Figure 1. **Persistent Dynamic Novel-View Synthesis and Tracking Results.** Novel-view (unseen) renders of color images and depth maps across 2 scenes. Each scene is parameterized by 200-300k Dynamic 3D Gaussians which move over time. We render (with occlusions) the 3D trajectories of 2.5% of these over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

3D Gaussian Splatting [2] 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, while regularizing the motion with a local-rigidity prior, which ensures that local neighborhoods of particles move approximately rigidly between timesteps.

We perform experiments using synchronized multi-view video (27 training cameras, 4 testing cameras) from the CMU Panoptic Studio dataset [1]. 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 ac-

Figure 2. **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.

curate 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, 2 and 3.

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 4. Furthermore, the extremely fast rendering and training time make them much easier to work with than previous approaches for dynamic reconstruction, and enable real-time rendering applications.

2. Method

Overview. Given a set of images from different timesteps and cameras, with each camera's respective intrinsic (K_c) and extrinsic ($E_{t,c}$) matrices, our approach reconstructs the dynamic 3D scene observed by these cameras in a temporally persistent manner via test-time optimization via gradient descent through a differentiable renderer. 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.

Dynamic 3D Gaussians. Our dynamic scene representation is parameterized by a set of Dynamic 3D Gaussians, each of which has the following parameters:

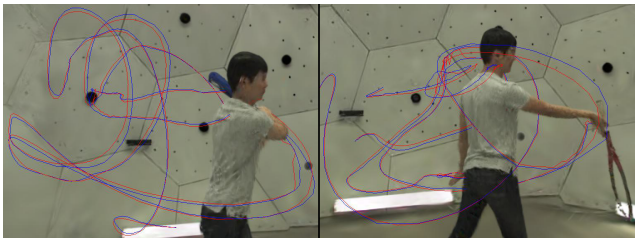

Figure 3. **Ground-truth Comparison.** Comparing our result (blue) to the ground-truth (red). Where ground-truth is noisy, our result may be more accurate.

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. Scenes are represented by between 200-300k Gaussians, with only 30-100k part of foreground.

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. 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, and the change in rotation over time in Fig 2.

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 [2] and extend its use to dynamic scenes. This works by splatting 3D Gaussians into the image plane by approximating the projection of the in-

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 $\mathcal{L}_{\text{Bg}}$	✓	✓	✓	✗	✓	✓	24.14	0.82	0.34	8.46	60.0	6.40	63.2
3	No Param Fixing	✓	✓	✓	✓	✗	✓	27.14	0.89	0.22	30.7	57.7	19.15	58.8
4	No Forward Prop	✓	✓	✓	✓	✓	✗	28.48	0.91	0.16	6.32	54.87	5.4	57.7
5	3GS-O [2]	✗	✗	✗	✗	✗	✗	28.19	0.90	**0.15**	32.81	13.6	23.86	17.1

Table 1. **Ablation results on PanopticSports.** See text for details on the dataset, metrics, tasks and methods.

tegral of the influence function f along the depth dimension of the 3D Gaussian into a 2D Gaussian influence function in pixel coordinates. This is incredibly fast an allows our method to render at 850 FPS. See [2] for details.

Physically-Based Priors. In order to correctly model the non-rigid physical motion of parts of the scene, we introduce a local-rigidity loss $\mathcal{L}^{\text{rigid}}$:

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

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

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.

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 $\sim$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.

Optimization Details. Following [2], in the first timestep we initialize the scene using a coarse point cloud from available depth sensors, and use the densification from [2] 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.

For 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 rotation.

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.

3. 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 [1]. 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. Fig 1 shows an example.

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 [3]. 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.

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. For 2D long-term point tracking we use the metrics from the recent point-odyssey benchmark [4]: median trajectory error (MTE) and position accuracy (δ). For 3D long-term point tracking there is no prior relevant work, so we decide adapt the 2D metrics from [4] 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 and δ is calculated at 1, 2, 4, 8 and 16cm thresholds.

Results and Ablations. In Table 1 we show results on our `PanopticSports` dataset ablating the various different components of our approach. We identify 4 key components

Figure 4. **Creative 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.

of our approach which are above and beyond that of the original 3D Gaussian splatting approach [2], as described in Section 2. 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 [2] 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.

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 4.

4. 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.

References

[1] Hanbyul Joo et. al. Panoptic studio: A massively multiview system for social motion capture. In *CVPR*, 2015. 1, 3

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 2023. 1, 2, 3, 4

[3] T. Simon, H. Joo, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. *CVPR*, 2017. 3

[4] 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