DynOMo: Online Point Tracking by Dynamic Online Monocular Gaussian Reconstruction

Jenny Seidenschwarz[1,2,*] Qunjie Zhou[3] Bardienus P. Duisterhof[2] Deva Ramanan[2] Laura Leal-Taixé[3]

[1]Technical University of Munich [2]Carnegie Mellon University [3]NVIDIA

Figure 1. **Monocular Online Point Tracking**: In this work, we present DynOMo for the task of monocular online point tracking from *pose-free* videos through joint 3D reconstruction and camera localization based on a dynamic 3D Gaussian representation. Please find the code and more visualizations on our project page https://jennyseidenschwarz.github.io/DynOMo.github.io.

Abstract

Reconstructing scenes and tracking motion are two sides of the same coin. Tracking points allow for geometric reconstruction [13], while geometric reconstruction of (dynamic) scenes allows for 3D tracking of points over time [22, 36]. The latter was recently also exploited for 2D point tracking to overcome occlusion ambiguities by lifting tracking directly into 3D [35]. However, above approaches either require offline processing or multi-view camera setups both unrealistic for real-world applications like robot navigation or mixed reality. We target the challenge of ***online 2D and 3D point tracking*** *from unposed monocular camera input introducing* ***Dy****namic* ***O****nline* ***Mo****nocular Reconstruction (**DynOMo**). We leverage 3D Gaussian splatting to reconstruct dynamic scenes in an online fashion. Our approach extends 3D Gaussians to capture new content and object motions while estimating camera movements from a single RGB frame.* ***DynOMo*** *stands out by enabling emergence of point trajectories through robust image feature reconstruction and a novel similarity-enhanced regularization term, without requiring any correspondence-level supervision. It sets the first baseline for online point tracking with monocular unposed cameras, achieving performance on par with existing methods. We aim to inspire the community to advance online point tracking and reconstruction, expanding the applicability to diverse real-world scenarios.*

*Correspondence to j.seidenschwarz@tum.de

1. Introduction

Modeling static geometry and dynamic object motion over a long-term time range is highly correlated to point tracking. In fact, they can even be addressed by point tracking and vice versa. Early point tracking methods have been primarily utilized in Simultaneous Localization and Mapping (SLAM) [2, 3, 7] to simultaneously reconstruct a 3D scene map and localize a camera moving through the scene. SLAM approaches typically focus on static scene contents to ensure accurate localization. To eliminate the influence of dynamic objects, they often track sparse features that lie on static elements in the scene. Recently, works like [13, 18, 24, 44, 45] also explored dense SLAM for static scenes based on 3D Gaussians Splatting [14] (3DGS) and Neural Radiance Fields [25].

2D Point Tracking. To model dynamic content, TAP-Vid [4] recently introduced the challenge of "Tracking Any Point" in 2D across frames of a video. Building upon prior works [32], recent approaches [4, 5, 10] learn correspondence matchers that can be augmented to exploit correlation between multiple tracks with a transformer [12]. However, such methods require training data with correspondence-level annotations. Because of the difficulty in labeling, such motion annotations are often limited to pure 2D pixel correspondences. Yet motion projected in 2D is inherently more ambiguous due to the discarded depth information [36].

3D Point Tracking. As such, much recent work proposes to track points directly in a (pseudo) 3D space [22, 35, 36] by removing the need for training data and instead optimizing a spacetime reconstruction directly on the test video of interest. Omni-motion [35] and the concurrent work of shape-of-motion [36] optimize a dynamic reconstruction of an entire scene in an offline fashion, requiring 2D flow and 2D point tracks as supervision, respectively. Such offline 2D supervision and batch optimization can be compute intensive and prone to 2D errors, as described above. Conversely, D-3DGS[22] achieves *online* point tracking by formulating the optimization as a dynamic 3D Gaussian reconstruction. However, D-3DGS requires a large number (27) of static cameras and an initial 3D point cloud of the scene. Thus existing approaches are difficult to apply to real-world scenarios like robot navigation and mixed reality, which require online and interactive tracking of both static and dynamic content.

Towards that end, we present `DynOMo`, the first online point tracking algorithm for monocular videos in-the-wild (with unknown camera poses). Our approach draws inspiration from SpaTAM [13], D-3DGS [22], and 3D feature distillation [15], combining the strengths of dynamic scene modeling, online tracking, and camera localization using Gaussian Splatting. Our key insight is to augment the RGB reconstruction loss with pre-trained 2D pixel encoders, including monocular depth, semantic object class labels, and feature descriptors. Interestingly, we make use of *no* temporal supervision, and demonstrate that 3D tracking *emerges* naturally from static 2D encoders by simply augmenting each dynamic 3D Gaussian with a descriptor (that is inferred during online optimization). Crucially, descriptors can be used to *group* Gaussians when applying spatial regularizers during the optimization. Past work often groups Gaussians by proximity [22], but we find it better to group by descriptor (i.e., points should move like other points with similar appearance rather than those are simply nearby). Our experiments corroborate the well-known result from perceptual grouping that proximity is the result, not the cause, of similarity [8]. Because our optimization explicitly recovers camera pose and new regions of the scene via densification [13], our online reconstructions allow for exploration of newly emerging scene content while keeping track of previously observed regions.

In summary, our model can track any point in a dynamic scene over time through 3D reconstruction from monocular video input. Our main contributions are as follows: (i) We introduce a *tracking-by-reconstruction* baseline for tackling online TAP (tracking any point) in 2D and 3D, with unposed monocular cameras. (ii) We demonstrate that temporal correspondence can emerge without correspondence-level optical flow supervision or known camera poses in challenging monocular settings. (iii) We show through evaluation on common benchmarks that our approach significantly outperforms baseline methods in the same online monocular setting and is competitive with state-of-the-art offline 2D trackers. We see our work as an initial step towards online TAP and reconstruction with unposed monocular cameras in the wild, and we encourage future research to further enhance its accuracy and efficiency.

2. Related Work

2.1. Dynamic Scene Reconstruction

Neural Radiance Fields. NeRF [25] has given rise to a new class of scene representations for novel view synthesis. While its earlier work was limited to static scenes [1, 23, 26, 31], more recent works [9, 19, 20, 28–30, 38, 40] have extended the topic to also model dynamic scenes. However, NeRF represents a scene implicitly and thus requires time-consuming volumetric rendering.

3D Gaussian Splatting. 3DGS [14] has recently gained great attention in neural rendering. In contrast to NeRF, it explicitly represents a scene with 3D Gaussian primitives and uses efficient differentiable rasterization techniques. Following a similar road map, 3DGS methods started from handling only static scenes but recently expanded to modeling dynamic scenes[6, 17, 21, 22, 34, 37, 41, 42]. In contrast to NeRFs, the fast per-sequence optimization of 3DGS allows for more real-world use cases. Hence, our work adopts 3DGS to achieve dense reconstruction in dynamic scenes in a monocular setting towards real-world applications.

2.2. Point Tracking

Simultaneous Localization and Mapping. Establishing point tracks over a video sequence is the core task of SLAM [3, 7]. However, exisiting SLAM algorithms focus on reconstructing static scenes for accurate geometry and camera motion estimation. Sparse SLAM [2, 3, 7] commonly relies on keypoint detection and matching to tracking points between frames. Recent works leverage dense scene representations such as NeRF [16, 18, 33, 44, 45] and 3DGS [13, 24] to perform dense SLAM via pose optimization from view synthesize on static scenes. Our method builds on top of SplaTAM [13] to enable its reconstruction on dynamic scenes.

Tracking Any Point. In contrast to the SLAM task, the task of tracking any point aims to track both static and dynamic points in a scene. The seminal work Partical Video [32] models video motion using a set of particles that move through time where they refine optical flow estimates for long-term consistency with occlusion handling. Its recent revisit PIPs [10] upgrades the particle motion estimates with a feed-forward network in an offline sliding-window manner. TAPIR [5] combines PIPs with a better matching model inspired by TAPNet [4], while PointOdyssey [43] provides an updated version of PIPs. Instead of tracking

each point independently [4, 5, 10], CoTracker [12] exploits the correlation between all image tracks with a transformer network, resulting in significantly improved accuracy.

Different from the above works that operate tracking in the 2D image space, several recent work [22, 35, 39] propose to lift 2D points into 3D and perform tracking in the 3D space to better handle occlusions and maintain temporal and spatial coherence. Omnimotion [35] optimizes a volumetric representation per video with view synthesize and optical flow supervisions at test time. SpaTracker [39] leverages an off-the-shelf monocular depth estimator to lift pixels into 3D with associated CNN features and construct a triplane representation for dense feature sampling. Then the authors train a transformer to update the 3D trajectory over local windows of temporal frames based on their triplane features in a feed-forward manner. D-3DGS [22] extends static 3D Gaussian splatting [14] to cope with dynamic scenes. With the explicit Gaussian representation, reconstruction on both static and dynamic scene points over time naturally solves the point tracking task in a global 3D space. Different from [35, 39], it is supervised using view synthesize and physics-inspired regularization without the need of correspondence-level supervisions. Another recent work [36] refines long-term 2D tracks from [5] by lifting them to 3D Gaussians using depth information meanwhile enable novel view synthesize.

Dynamic Online Monocular Point Tracking. Most point tracking methods are designed for offline tracking except for SLAM methods and D-3DGS [22]. However, SLAM methods struggle with dynamic scenes and D-3DGS [22] requires posed videos in a multi-camera setting. Compared to them, our goal is to perform online point tracking with unposed monocular videos, which enables point tracking for real-time response applications such as live video analysis, surveillance systems and mixed reality applications.

3. Method

In this section, we present **DynOMo**, our online point tracking pipeline via dynamic 3D Gaussian reconstruction for unposed monocular camera videos. Building on top of [13, 22], we combine the powerful 3DGS-based dynamic scene representation and the flexible online tracking paradigm for pose-free videos through simultaneous scene reconstruction and camera localization. The key to DynOMo's performance lies in three technical adaptions for the online tracking setting, namely, 1) reconstruction signal enhancement with stronger image features and depth supervisions, 2) semantic-based foreground and background separation to enable camera tracking, and 3) motion regularization bootstrapping via a feature-similarity-guided weighting mechanism.

In the following we first detail how we represent the scene via 3DGS in Sec. 3.1 and explain the online tracking pipeline in Sec. 3.2, followed by details about optimization supervisions (*cf*. Sec. 3.3 and Sec. 3.4). Finally, we show how to estimate 2D and 3D trajectories from a query point in Sec. 3.5.

3.1. Dynamic Gaussian Scene Representation

We first introduce how we model a dynamic scene using 3D Gaussians. We start with a recap on 3D Gaussian splatting (3DGS) technique on static scene and its recent extension for dynamic scene representaiton by D-3DGS [22]. Finally, we present our adaption of D-3DGS to cope with challenges faced with monocular online point tracking.

Preliminaries: 3DGS for static scenes. 3D Gaussian Splatting [14] represents the appearance and geometry of a static 3D scene with a set of M explicit anisotropic 3D Gaussian distributions $\mathcal{G} = \{G_i\}_{i=0}^{M}$. Each 3D Gaussian is parametrized by its mean μ_i and covariance matrix Σ_i and defined as:

$$G_i(x; \mu_i, \Sigma_i) = e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}. \tag{1}$$

The covariance matrix is defined as $\Sigma_i = R_i s_i s_i^T R_i^T$ where s_i is a scale vector and R_i a rotation matrix represented by a quaternion vector q_i. Each G_i is additionally assigned with color c_i and opacity o_i attributes. To efficiently render a RGB image at a given camera, the 3D Gaussians are splatted into 2D Gaussians $G_{i,2D}(p; \mu_{i,2D}, \Sigma_{i,2D})$ on the image plane utilizing an affine approximation of the camera projection [46]. Finally, the color of a pixel p is computed by α-composing the color values of the projected 2D Gaussians that intersect with pixel p:

$$C_p = \sum_{i \in H} T_i \alpha_i c_i \tag{2}$$

where $\alpha_i = o_i e^{-\frac{1}{2}(p-\mu_{i,2D})^T \Sigma_{i,2D}^{-1}(p-\mu_{i,2D})}$ and $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$. The splatting-based rendering process is fully differentiable thus the Gaussian parameters $(\mu_i, q_i, s_i, c_i, o_i)$ are usually optimized directly by an image reconstruction loss [14] between a training image and the synthesized image rendered at that training camera pose.

Preliminaries: 3DGS for Dynamic Scenes. As mentioned in Sec. 2.1, many recent works [6, 17, 21, 22, 37, 41, 42] explored how to model dynamic scenes with 3DGS. Among those, we adopt the formulation of D-3DGS [22] where a dynamic Gaussian at timestamp τ is defined by $G_i^\tau \equiv (\mu_i^\tau, q_i^\tau, s_i, c_i, o_i)$. Here μ_i^τ and q_i^τ are the Gaussian mean and rotation at timestamp τ while the other parameters are constant over time as in the original 3DGS. In contrast to its related work [21, 41, 42] that require offline training of the scene representation over a whole video, D-3DGS is online optimized enabling online tracking-by-reconstruction via 3D Gaussians, which intersects with our interest. However, this work requires an unrealistic multi-camera setting

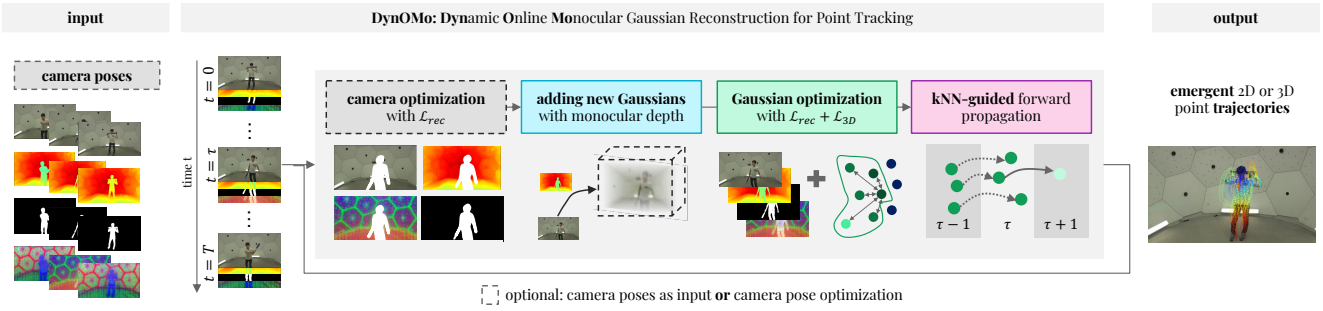

Figure 2. **Tracking points with online dynamic monocular reconstruction:** Our pipeline assumes an input video sequence, (predicted) depth maps, sparse segmentation masks as well as image features as input. In our online reconstruction pipeline we optimize for the camera pose $\mathcal{C}$, add a set of new Gaussians based on the densification concept [13], optimize all Gaussians together and forward propagate $\mathcal{G}$ and $\mathcal{C}$. Finally, we directly extract 3D point trajectories from single Gaussians G_p and project them to the image plane to obtain 2D trajectories.

of 27 multi-view images per timestamp to guarantee quality reconstruction and tracking performance. Moving it directly to a monocular setting leads to significantly degraded performance as we show in Tab. 1.

Adaptation for Monocular Online Reconstruction. In the monocular online reconstruction setting, each timestamp we only have access to a monocular RGB image, which contains much sparser supervision signals to constrain the Gaussian optimization process compared to multi-view setting. Therefore, to compensate this, we extend each Gaussian with two extra parameters, *i.e.*, a semantic instance label $g_i \in R$ and a visual feature $f_i \in R^D$ which are constant overtime and optimized jointly with other parameters. Specifically, our monocular dynamic Gaussian is defined as $G_i^\tau \equiv (\mu_i^\tau, q_i^\tau, s_i, c_i^\tau, o_i, g_i, f_i^\tau)$ at a timestamp τ. Inspired by recent correspondence-supervised 2D point tracking approaches [5, 10, 12, 39], we use the visual feature i) as a stronger version of photometric information to constrain the reconstruction to agree with each other in the feature space and ii) to provide weighting guidance for the optimization regularization. We further explore semantic level information to assign an instance id for each Gaussian to decouple foreground and background particles. It not only allows us to enforce semantic-level consistency when a particle moves around but also supports us to design regularizations (*cf*. Sec. 3.4) specialized for the moving particles.

To initialize and optimize our augmented dynamic 3DGS parameters, in addition to the RGB frames, we assume to have access to extra information of that image specifically its image visual feature map extracted with DinoV2 [27], its depth image and its (sparse) instance segmentation mask where only a few objects are segmented. We use the segmentation mask to constrain our motion regularization as well as to obtain a binary foreground/background segmentation, which is used to constrain the motion of the static background (see Sec. 3.4). We provide more details about how to obtain those information in the supplementary material.

3.2. Monocular Online Tracking Pipeline

With our augmented dynamic 3DGS scene representation, we are limited to reconstruct dynamic scenes from a monocular video with known camera poses or that taken from a static camera. To further enable monocular online point tracking for *pose-free* videos, we build our tracking pipeline on top of SplaTAM [13] to simultaneously reconstruct the scene and localize camera poses. As shown in Fig. 2, our method takes in a set of required inputs per-timestamp and performs online point tracking with the following steps.

Gaussian and Camera Initialization. Given the first frame of a video, we initialize the Gaussians by lifting 2D image pixels to 3D using its pre-computed depth map and its camera pose $\mathcal{C}$ as identity, *i.e.*, the reference of the scene.

Camera Optimization. We then optimize the camera pose initialized for the current timestamp using the reconstruction losses L_{rec} (*cf*. Eq. (5)) with fixed Gaussian parameters. We only apply the loss over pixels that have been observed before and belong to the background which we assume to be static. We identify the prior based on the rendered pixel density which captures the epistemic uncertainty of the reconstruction [13] and the latter by the sparse segmentation masks. As a known limitation to SLAM-based approaches, our method also struggles with extreme camera motions. When the camera poses are available, we can turn off this step and focus on the Gaussian optimization.

Adding New Gaussians. Next, we explore the world by adding a set of new Gaussians using the same initialization defined in the first step. We add Gaussians only for sparse and under-represented Gaussian regions identified by a densification mask [13]. We provide its detailed definition in the supplementary material.

Gaussian Optimization. Afterwards, we fix camera pose

parameters and optimize all 3D Gaussians using a mixture of reconstruction losses defined in Sec. 3.3 and 3D regularization losses defined in Sec. 3.4, *i.e.*, $\mathcal{L}_{total} = \mathcal{L}_{rec} + \mathcal{L}_{3D}$.

Gaussian Forward Propagation. After that, we forward propagate the Gaussian means μ_i and rotations q_i to the next timestamp. We assume that close-by Gaussians in the 3D space should move in a similar way if they share similar semantics. Hence, we forward propagate μ_i based on a kNN-constant velocity assumption:

$$\mu_i^{\tau+1} = \mu_i^{\tau} + \sum_{j \in \mathcal{N}_i} \sigma(s_{i,j})(\mu_j^{\tau} - \mu_j^{\tau-1}), \tag{3}$$

where $\sigma(s_{i,j})$ is the softmax function over the cosine similarity $s_{i,j}$ between the feature vectors of Gaussian i and a Gaussian j in its k-nearest neighbour Gaussians $\mathcal{N}_i$ in 3D space. For the rotation q_i, we forward propagate it based on a simple constant-rotation assumption:

$$q_i^{\tau+1} = \Delta q_i^{\tau-1 \to \tau} * q_i^{\tau}\,, \tag{4}$$

where $\Delta q_i^{\tau-1\to\tau} = q_i^{\tau} * (q_i^{\tau-1})^{-1}$ is the relative rotation from time step $\tau - 1$ to τ.

Camera Forward Propagation. Finally, we forward propagate the camera poses for the next timestamp by applying the velocity forward projection proposed in [13] over the camera pose from the previous timestamp.

For a each frame, we repeat the steps of camera optimization to camera forward propagation. In the following, we the loss functions to optimize $\mathcal{C}$ and $\mathcal{G}$.

3.3. Reconstruction Supervision

As the core supervisory signal for both scene reconstruction and camera localization, we compute reconstruction losses across multiple image types, including RGB color, depth, visual features, and semantics. By leveraging this mixture of information, we impose stronger constraints on point movement over time, requiring consistency with the input frame from multiple semantic perspectives, beyond just photometric errors. Our reconstruction loss $\mathcal{L}_{rec}$ is specifically defined by a weighted sum over those reconstruction losses:

$$\mathcal{L}_{rec} = \lambda_I \mathcal{L}_I + \lambda_F \mathcal{L}_F + \lambda_D \mathcal{L}_D + \lambda_B \mathcal{L}_B. \tag{5}$$

We obtain the rendered image of different information by applying the same alpha-composition over different Gaussian attributes a_i:

$$A_p = \sum_{i \in H} T_i \alpha_i a_i, \tag{6}$$

with $a_i \in \{c_i, f_i, d_i, g_i\}$ where d_i is the distance of G_i to the camera for rendering the depth map. We use $l2$ distance for feature reconstruction $\mathcal{L}_F$ and $l1$ distance for RGB reconstruction $\mathcal{L}_I$, depth reconstruction $\mathcal{L}_D$ and background mask reconstruction. We ablate the importance of individual reconstruction losses in Sec. 4.3.

3.4. 3D Regularization

In addition to constraining the optimization in 2D over rendered images, [22] introduced three regularization terms applied to a local neighborhood of each Gaussian to enforce the optimization procedure to be physically plausible directly in the 3D space. Briefly, they define a short-term local-rigidity loss $\mathcal{L}^{rigid}$ to enforce neighbouring Gaussians to move obeying a similar rigid transform, a local-rotation similarity loss $\mathcal{L}^{rot}$ to force nearby Gaussians to have the same rotations over time, and a long-term local-isometry loss $\mathcal{L}^{iso}$ which constrains the relative distance between two Gaussian means remains the same over time. Due to the space limit, we provide their equations in the supplementary material.

Instance-guided Neighbourhood. [22] defines the k nearest neighbours $\mathcal{N}_i$ of a Gaussian G_i based on their 3D distance, which may wrongly define Gaussians from another object to be neighbours around the current object boundary. To avoid this case, we leverage the instance id to select neighbouring Gaussians only within the same object.

Feature-guided Weighting. Each regularization loss $\mathcal{L}^x \in \{\mathcal{L}^{rigid}, \mathcal{L}^{rot}, \mathcal{L}^{iso}\}$ is further computed by a weighted sum of every Gaussian pair (G_i, G_j), defined by:

$$\mathcal{L}^x = \frac{1}{k|\mathcal{G}|} \sum_{G_i \in \mathcal{G}} \sum_{G_j \in \mathcal{N}_i} w_{i,j} \mathcal{L}^x_{i,j}\,. \tag{7}$$

[22] defines the pair-wise weight $w_{i,j} = ||\mu_i - \mu_j||$ based on their 3D distance. However, different parts of an object can be close to each other but still not connected. Hence they do not move together, *e.g.*, a hand and a leg. Therefore, we consider to measure the Gaussian distance in the feature space with the cosine similarity between their features f_i and f_j, *i.e.*, $w_{i,j} = s_{i,j}$ which significantly improves the tracking performance (*cf*. Sec. 4.3).

Temporal Smoothness Regularization. We exploit additional priors that the same Gaussian point should not vary much in color and feature space over time and background Gaussians should be static. Thus, we restrict individual Gaussian feature f_i and color c_i as well as the mean of background Gaussians to remain close from their values in the previous timestamp using l_1 distance. We provide the detailed formula of this smoothness loss $\mathcal{L}_{sm}$ in the supplementary.

Our total 3D regularization loss is defined as $L_{3D} = \lambda_{sm}\mathcal{L}_{sm} + \lambda_{rigid}\mathcal{L}_{rigid} + \lambda_{iso}\mathcal{L}_{iso} + \lambda_{rot}\mathcal{L}_{rot}$. We provide ablation study on the importance of all loss functions in Sec. 4.3.

3.5. Trajectory Estimation

As we perform *tracking-by-reconstruction* with 3D Gaussians, a 2D pixel can be represented via multiple 3D Gaussians. Therefore, there exists no one-to-one mapping between the 2D pixel and the 3D Gaussian we track over time.

To estimate the 2D or 3D trajectory over time for a query pixel p, we choose its corresponding 3D Gaussians G_p at time τ by:

$$G_p = \min_{G_i \in \mathcal{G}_v^\tau} (||p - \Pi(W^\tau \mu_i^\tau)||_2), \tag{8}$$

where $\mathcal{G}_v^\tau$ is the set of Gaussians with visibility $v_i^\tau > 0.5$ and W^τ is the camera projection matrix and Π is the perspective projection operation. To identify whether a Gaussian is visible, we accumulate the Gaussian opacity in an α-composition manner over a set of pixels P_{G_i} that intersect with the 2D Gaussian projection. We thus define the visibility score v_i^τ of a Gaussian as:

$$v_i^\tau = \sum_{p \in P_{G_i}} T_i \alpha_i \tag{9}$$

Finally, we fully exploit the explicit Gaussian particle representation and obtain the 3D and 2D trajectories by just following G_p through time:

$$x_p^{3D,\tau} = \mu_p^\tau, \qquad x_p^{2D,\tau} = W^\tau \mu_p^\tau \tag{10}$$

4. Experiments

We now evaluate our model for monocular online point tracking on different benchmarks to compare with various baselines. We provide implementation details of our method in the supplementary material.

4.1. Evaluation on Panoptic Sports

As the first part of our experiment, we compare `DynOMo` to D-3DGS [22] to highlight the challenge of online monocular point tracking and validate the effectiveness our method.

Dataset. To this end, we evaluate our method for both 2D and 3D point tracking on the PanopticSport dataset [22]. It consists of the six sports sequences of the original Panoptic Studio dataset [11] where each sequence contains multi-view images taken from 31 temporally aligned cameras with 150 frames being sampled per camera at 30 FPS.

Metrics. For 2D point tracking evaluation, we report averaged tracking accuracy ($\delta_{avg,2D}$) [4] that computes the mean percentage of points of a trajectory within distance errors of $\{1, 2, 4, 8, 16\}$px to the ground truth trajectory, the median translation error MTE_{2D} in pixels, the survival rate S_{2D} that evaluates the percentage of points below 16px translation error. For 3D point tracking, we report the median translation errors MTE_{3D} in cm, the survival rate S_{3D} at a $50cm$ error threshold and $\delta_{avg,3D}$ with thresholds of $h = \{1, 2, 4, 8, 16\}$ in cm.

Baselines. We compare to the original version of our baseline online tracking approach D-3DGS which is optimized using 27 (out of 31) cameras for each timestamp. To reduce to a monocular setting, we choose a single camera such that the tracked object is observed from a reasonable view point.

Results. As shown in Tab. 1, D-3DGS-Mono's performance degrades significantly with a 121.9% and 308.8% increase in MTE errors for 2D and 3D point tracking respectively. This highlights that D-3DGS relies heavily on multi-view images to guarantee effectiveness of Gaussian optimization, while `DynOMo` explores other cues from visual features and instance semantics to constrain optimization signals and compensate the lack of multi-view information. Comparing both methods between their 2D and 3D tracking performance, we also observe that the monocular setting leads to a bigger drop in performance in 3D compared to D-3DGS optimized in an extreme multi-view setting. This reveals that multi-view images provides accurate geometric supervision for 3D reconstruction in general. In comparison, our method relies on monocular depth maps for geometric supervisions. Thus, we believe more accurate depth maps can largely improve our 3D tracking performance in the future. Finally, we point out that our method surprisingly outperforms D-3DGS on 2D point tracking in a much more challenging camera setting, which strongly suggests the effectiveness of our design choice.

Influence of trajectory estimation. With our oracle baseline (`DynOMo`*), we show that there exists a 3D Gaussian that corresponds better to the query point whose trajectory are closer to the ground truth trajectory compared to the one chosen by Eq. (8). Thus, our work can be largely improved with a better trajectory estimation mechanism.

4.2. Evaluation on TAPVid-DAVIS

Different from D-3DGS that requires not only multi-view images but also known camera poses for online tracking, our method is able to model the camera motion jointly with point tracking, which allows us to compare to the state-of-the-art 2D point tracking methods that are evaluated on *pose-free* videos.

Dataset. We evaluate our method on the popular TAPVid-DAVIS dataset introduced as part of the TAPVid benchmark [4]. It consists of 30 highly varying real-world video sequences with unknown camera poses, among which some contains extreme camera motions. This dataset only provides ground truth for 2D point tracking.

Metrics. We evaluate our 2D point tracking performance utilizing the TAPVid [4] metrics, *i.e.*, averaged tracking accuracy ($\delta_{avg,2D}$) introduced previously, occlusion accuracy (**OA**) which computes the percentage of frames with correctly predicted visibility and average jaccard (**AJ**) that combines the both metrics.

Baselines. While our method focuses on online tracking, we are also curious about the performance gap from the stat-of-the-art offline 2D point trackers. Sepecifically, we compare to Pips [10], TAP-Net [4], TAPIR [5], CoTracker [12] and SpaTracker [39] which are all feed-forward methods that require a large amount of data for model pre-training.

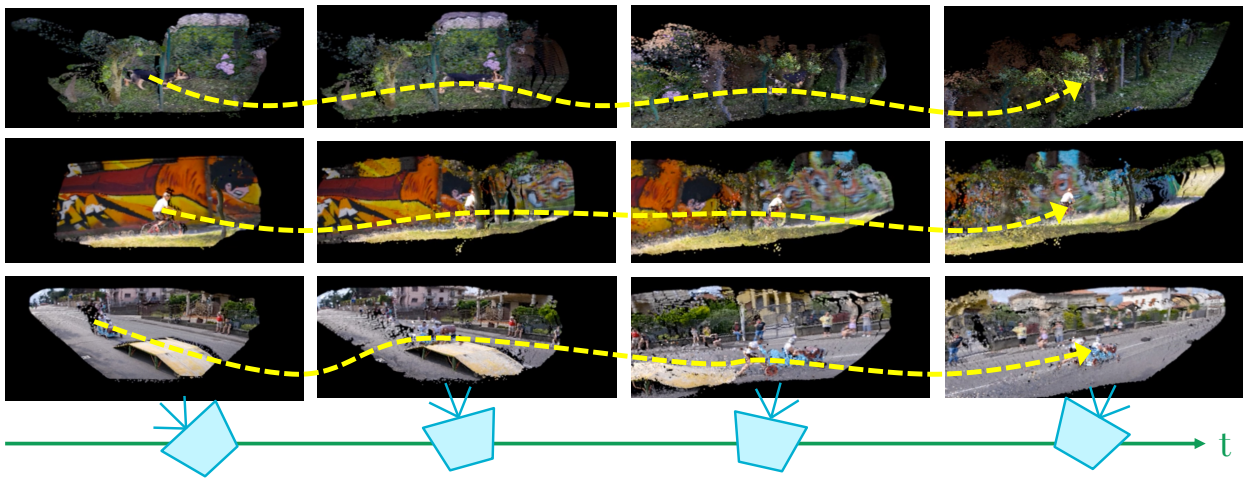

Figure 3. **Increasing the World Online:** Our pipeline is able to gradually add Gaussians to the world. This allows to explore the underlying world as the video progresses. We visualize the increase of the world for two sequences of TAPVid-DAVIS.

Method	# Cameras	$\text{MTE}_{2D} \downarrow$	$S_{2D} \uparrow$	$\delta_{avg,2D} \uparrow$	$MTE_{3D} \downarrow$	$S_{3D} \uparrow$	$\delta_{avg,3D} \uparrow$
D-3DGS [22]	27	10.5	76.8	57.4	**13.7**	**86.8**	**43.7**
D-3DGS-Mono	1	23.3	41.0	30.6	56.0	44.5	7.3
`DynOMo`	1	**6.3**	**85.7**	**61.8**	26.1	73.0	12.7
`DynOMo`*	1	5.8	85.7	66.9	24.5	70.8	10.1

Table 1. **Comparison on Panoptic Studio [22]:** We compare our performance on PanopticSports as described in Sec. 4.1. D-3DGS-Mono represents the monocular setting of [22]. We observe that despite significantly more challenging setting, for 2D point tracking `DynOMo` outperforms [22]. Additionally, `DynOMo` outperforms D-3DGS-Mono considerably for 3D showing the effectiveness of our method. Finally, `DynOMo`* shows that shows that there exists a 3D Gaussian that corresponds even better to the query point.

	Method	Pretrain	AJ $\uparrow$	$\delta_{avg,2D} \uparrow$	OA $\uparrow$
Offline	TAP-Net [4]	✓	33.0	48.6	78.8
	Pips [10]	✓	42.2	64.8	77.7
	TAPIR [5]	✓	56.2	70.0	86.5
	CoTracker [12]	✓	62.2	75.7	89.3
	OmniMotion [35]	✗	51.7	67.5	85.3
	SpaTracker [39]	✓	**61.1**	**76.3**	**89.5**
Online	SpaTracker†	✓	49.7	64.5	81.0
	SplaTAM	✗	13.0	22.1	72.4
	`DynOMo`	✗	45.8	63.1	81.1
	`DynOMo`*	✗	52.7	69.2	84.3

Table 2. **Comparison on TAPVID-Davis [4] on 2D Point Tracking**: Compared to existing approaches that mostly operate offline and are pre-trained using motion supervision, `DynOMo` tracks points in an online way without *any* motion signal as supervision. Our emergent motion achieves comparable performance while our oracle baseline (`DynOMo`*) shows that there exists a 3D Gaussian that corresponds even better to the query point. SpaTracker† represents [39] evaluated in an online fashion with time window of 2.

We further consider the recent work OmniMotion [35] that performs test-time optimization over a whole video sequences. Among those method, SpaTracker and OmniMotion are conceptually more similar to ours as they also solve the 2D point tracking by lifting it to 3D, yet with different types of 3D representations. We also consider a relatively more comparable version of SpaTracker by reducing their sliding window size to 2 (SpaTracker†) for an online point tracking evaluation. Finally, we compare to SplaTAM [13] since we adopted its pipeline and extend it for dynamic scenes.

Results. As shown in Tab. 2, when we compare ourselves to offline tracking methods (*upper table*), `DynOMo` achieves on par tracking accuracy to OmniMotion and outperforms Pips and TAPIR. Despite that Omnimotion also performs tracking in 3D, their MLP-based implicit representation requires more expensive optimization over the whole sequence and slow runtime which impedes their application to online tracking ($\sim$18 hrs/sequence on A100 vs `DynOMo` $\sim$51 min/sequence, *i.e.*, $\sim$45s/frame on RTX 3090). While SpaTracker can directly perform online tracking, it leads to 15.5% drop in averaged tracking accuracy since their model is trained to explore cues from multiple temporal frames.

Within an online tracking category (*lower table*), we significantly outperform our baseline SplaTAM despite using a similar tracking pipeline, since our augmented GS representation is able to handle dynamic objects in the scene. Compared to SpaTracker, we are slightly lower in performance

Method	AJ ↑	δ_{avg} ↑	OA ↑
DynOMo	45.4	63.0	81.4
Reconstruction Supervision			
w/o $\mathcal{L}_B$	44.0	60.8	79.9
w/o $\mathcal{L}_I$	43.2	60.4	80.0
w/o $\mathcal{L}_D$	37.2	52.6	71.9
w/o $\mathcal{L}_F$	35.9	51.1	77.2
3D Regularization			
w/o $\mathcal{L}_{rot}$	43.9	61.2	79.8
w/o $\mathcal{L}_{iso}$	43.7	60.9	79.1
w/o $\mathcal{L}_{rigid}$	37.9	55.3	77.2
w/o instance-guided kNN	42.3	59.6	79.5
w 3D distance-based weighting	39.8	55.6	75.8
Gaussian Attribute Regularizations			
w/o kNN forward propagation	42.0	59.0	78.6
w/o temporal smoothness	42.4	59.3	78.6
w/o fixing s_i, o_i, and g_i	40.5	60.4	72.9

Table 3. **Ablation of Importance of Design Choices:** We show the importance of the reconstruction as well as the 3D loss functions. We see that for the prior the embedding as well as the depth losse and for the latter the rigidity loss as well as the semantic weighting are the most important, respectively. We also show the importance of our Gaussian attibute regularizations.

since their method has been train on a large amount of ground truth data while our method performs online tracking by online 3DGS optimization. We show that object motion emerges automatically when we constrain the 3DGS optimization through carefully designed 2D reconstruction supervision and 3D regularization. Similar to the previous experiment, DynOMo† leads to a boost in performance surpassing SpaTracker, suggesting the need for a better trajectory estimation for evaluation.

Qualitative visualization. In Fig. 3, we visualize our gaussian reconstruction across different timestamps. Given our optimized camera pose, we zoom out of the scene and rotate the camera pose slightly. We show that our method gradually explores the world as the video progress and is able to generate quality 3D reconstruction in complex dynamic scenes from just a single viewpoint frame per timestamp.

4.3. The Magic Sauce

To the end of our experiments, we ablate the most important design choices of our approach. Please refer to the supplementary material for more ablation experiments.

Influence of Reconstruction Losses. In Tab. 3, we show that our reconstruction losses play a significant role for accurate tracking performance. Among those, the reconstruction supervision from visual features $\mathcal{L}_F$ and depth maps $\mathcal{L}_D$ are more important. The rich semantic information from the feature maps enforce the Gaussian motion to agree with its feature observation, while the depth provides signal for geometrically correct reconstruction especially in the absence of multi-view images for monocular online tracking. While RGB information and background masks have lower impact, they still help when the other two signals are noisy due to imperfect off-the-shelf predictions.

Influence of 3D Regularization. In Tab. 3, we further study the importance of our proposed 3D regularization. Within the three physical-based regularization, the local-rigidity loss $\mathcal{L}_{rigid}$ is more influential than the other two. It forces Gaussians to move rigidly in the 3D space which can prevent 3D Gaussians to move randomly only to fit the 2D observation, which is significant when we lack of multi-view supervisions. We also show that our feature-guided regularization weighting is similarly crucial to the rigidity constraint and they cooperate together since Gaussians initialized from monocular depth can be rather noisy. Our feature guidance provides a measure of uncertainty to ignore outlier Gaussian pairs. Since the visual features are also predicted hence imperfect, instance-guided kNN still improves performance in helping the kNN selection process.

Gaussian Attribute Regularization. Finally, we show that compared to per-point forward propagation, our kNN forward propagation filters outlier motions in previous time steps leading to a significant performance increase. The temporal smoothness losses impede the Gaussians from drastically change color and features while enable necessary update for viewpoint changes. We also observe that updating s_i and o_i may cause some Gaussians to disappear and thus fix them together with the instance id attribute.

5. Conclusion

In this work, we address the task of online point tracking from pose-free monocular camera videos. We introduce **DynOMo**, a method that achieves online point tracking through dynamic 3D scene reconstruction using 3D Gaussian Splatting alongside simultaneous camera localization. To overcome the challenges of monocular input, we enhance the 3DGS representation with depth maps, instance IDs, and visual features, guiding Gaussian optimization from multiple semantic perspectives and enforcing physically meaningful constraints with robust 3D regularizers. Notably, we demonstrate that accurate point trajectories can emerge from this reconstruction without the need for explicit 2D/3D correspondences as supervision. Our approach is designed for applications requiring real-time reconstruction and tracking, such as robotic navigation and mixed reality. The ability to extend the 3D Gaussian scene over time as the video progresses enhances online interaction potential. We hope our work inspires the community to explore this path further. While our method establishes a strong baseline for online point tracking with monocular inputs, it has yet to achieve real-time performance. We anticipate that future work in this area will significantly improve runtime.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2

[2] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1, 2

[3] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007. 1, 2

[4] 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*, 2022. 1, 2, 3, 6, 7

[5] Carl Doersch, Yi Yang, Mel Vecerík, Dilara Gokay, Ankush Gupta, Yusuf Aytar, João Carreira, and Andrew Zisserman. TAPIR: tracking any point with per-frame initialization and temporal refinement. In *ICCV3*, 2023. 1, 2, 3, 4, 6, 7

[6] Bardienus Pieter Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *CoRR*, 2023. 2, 3

[7] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006. 1, 2

[8] Charless C. Fowlkes, David R. Martin, and Jitendra Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *CVPR*, 2003. 2

[9] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 2

[10] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, 2022. 1, 2, 3, 4, 6, 7

[11] 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. 6

[12] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. 2023. 1, 3, 4, 6, 7

[13] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *CVPR*, 2024. 1, 2, 3, 4, 5, 7

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 1, 2, 3

[15] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, 2022. 2

[16] Xin Kong, Shikun Liu, Marwan Taher, and Andrew J Davison. vmap: Vectorised object mapping for neural field slam. *CVPR*, 2023. 2

[17] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiV*, 2023. 2, 3

[18] Heng Li, Xiaodong Gu, Weihao Yuan, Luwei Yang, Zilong Dong, and Ping Tan. Dense RGB slam with neural implicit maps. In *ICLR*, 2023. 1, 2

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

[20] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, 2023. 2

[21] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, 2024. 2, 3

[22] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 1, 2, 3, 5, 6, 7

[23] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[24] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *CVPR*, 2024. 1, 2

[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. 1, 2

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2

[27] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024. 4

[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*, 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.*, 2021.

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[31] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 2

[32] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *International journal of computer vision*, 80:72–91, 2008. 1, 2

[33] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 2

[34] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *ECCV*, 2024. 2

[35] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *International Conference on Computer Vision*, 2023. 1, 2, 3, 7

[36] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. In *arXiv preprint arXiv:2407.13764*, 2024. 1, 2, 3

[37] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*, 2024. 2, 3

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

[39] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. Spatialtracker: Tracking any 2d pixels in 3d space. In *CVPR*, 2024. 3, 4, 6, 7

[40] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, et al. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 2

[41] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 2, 3

[42] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *ICLR*, 2024. 2, 3

[43] 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. 2

[44] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *CVPR*, 2022. 1, 2

[45] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2024. 1, 2

[46] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. EWA volume splatting. In *12th IEEE Visualization Conference, IEEE Vis 2001, San Diego, CA, USA, October 24-26, 2001, Proceedings*, 2001. 3