DynTCG: Dynamic View Synthesis from Monocular RGBD Streams via Flow-Tracked Compact Gaussians

Zhehao Shen
2021533110
shenzhh@shanghaitech.edu.cn

Penghao Wang
2021533138
wangph1@shanghaitech.edu.cn

Yu Hong
2021533148
hongyu@shanghaitech.edu.cn

Zhirui Zhang
2021533024
zhangzhr4@shanghaitech.edu.cn

Shouchen Zhou
2021533042
zhoushch@shanghaitech.edu.cn

Abstract

We present a method that uses deformable Gaussian representation based on monocular RGBD input, aimed at accomplishing the task of dynamic novel view synthesis. The 2D images with depth are back-projected into 3D point clouds to initialize Gaussians, and the flow between frames is utilized to generate warped Gaussians as the priors of the deformable network. RANS encoding and quantization were also used to do residue compress for improvements. Our method balanced the training, rendering time, and quantities of novel view, achieving monocular dynamic novel view synthesis.

1. Introduction

The precise reconstruction and photorealistic rendering of dynamic scenes from a collection of input images are paramount for numerous applications, such as augmented reality/virtual reality (AR/VR) and 3D content production. In these contexts, achieving high-quality results is essential for an immersive and visually convincing experience.

However, the multi-view setting remains impractical for everyday consumer use. In contrast, the monocular method, utilizing a single and convenient commercial RGBD camera, proves to be more practical and appealing for daily applications.

Recent advancements in neural rendering, exemplified by Neural Radiance Fields (NeRF)[6], have facilitated photorealistic rendering with dense-view supervision. Particularly noteworthy are dynamic variants of NeRF [2], which excel in synthesizing compelling novel views of dynamic scenes, even when captured monocularly. However, these approaches depend on laborious and time-consuming per-scene training to seamlessly integrate temporal observations into the canonical space.

The recent development known as 3D Gaussian Splatting (3DGS) [4] reverts to an explicit paradigm for representing static scenes. Utilizing GPU-friendly rasterization of 3D Gaussian primitives, it enables real-time and high-quality rendering of radiance fields that were unprecedented until now.

In this paper, we propose DynTCG, a novel approach for synthesizing novel views of dynamic scenes based on flow-tracked and compact Gaussian representations, utilizing monocular dynamic RGBD input. Our key idea involves leveraging the powerful prior provided by flow tracking of 2D images and depth information as the deform field for dynamic Gaussians. Simultaneously, we discovered that optimizing the scaling of Gaussians in the deform-net has minimal impact on appearance and can even affect training speed. Consequently, we modified the deform-net, originally designed in the baseline to optimize position, rotation, and scaling simultaneously. The adjustment now focuses solely on optimizing position and rotation. This alteration aims to enhance training speed with almost negligible impact on rendering quality. Given the substantial storage overhead associated with Gaussians (requiring storage of 3rd-order spherical harmonics coefficients for each point and information for modeling Gaussian scale, position, rotation, and opacity), we introduce a quantization and entropy encoding-based residual compression method tailored for dynamic scenes. This approach transforms the representation of Gaussians into a more compact form, thereby significantly reducing storage requirements.

To summarize, our main contributions include:

- We propose a deformable Gaussian representation based on monocular RGBD input, aimed at accomplishing the task of dynamic novel view synthesis.
- Utilizing optical flow tracking from 2D images, we provide an optimized initialization for the dynamic Gaussian

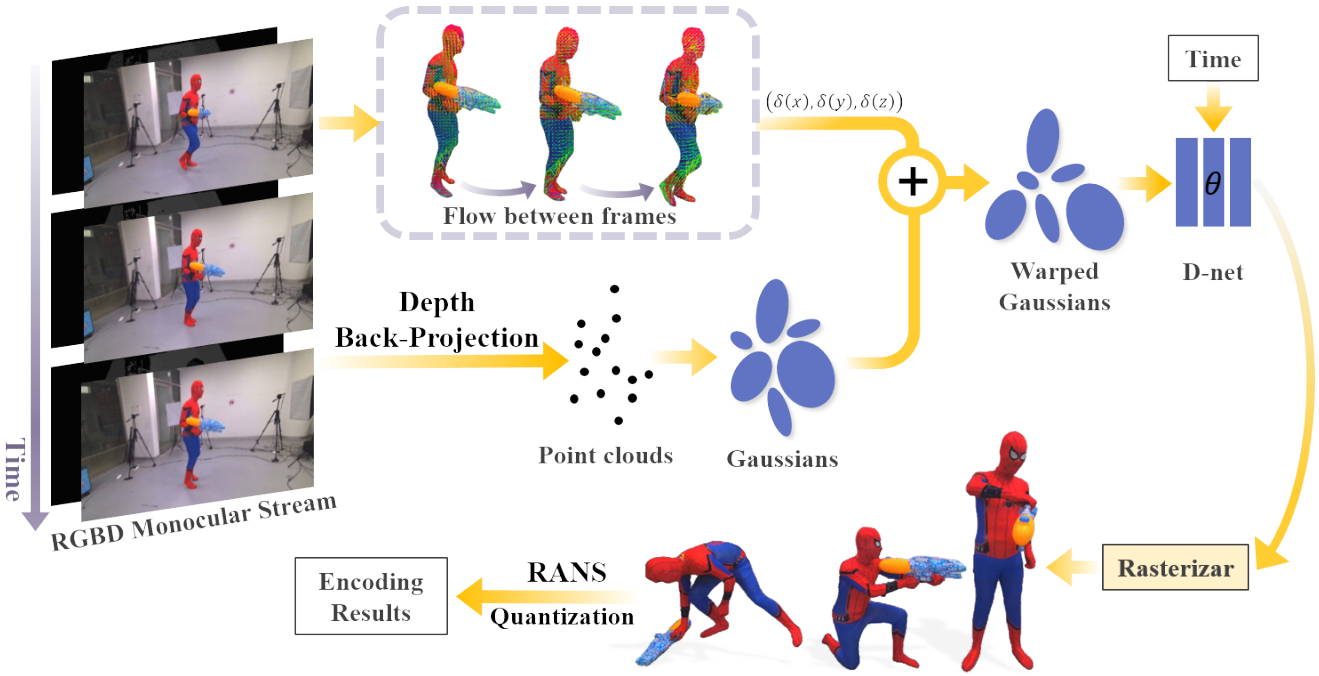

Figure 1. **Overview of DynTCG.** (a) Our setting involves an RGBD monocular stream. (b) DynTCG performs back-projection on the depth to generate initial point clouds and initialize Gaussians. (c) Utilizing the flow between frames, we warp the Gaussians, and the resulting warped Gaussians are used as a prior input into the deformable network (D-net). (d) The D-net additionally introduces time as an input to estimate the current $\delta(x)$, $\delta(y)$, $\delta(z)$, and $\delta(r)$. (e) After training, the Gaussians are then compressed using RANSAC and quantization.

deform field, significantly accelerating the training speed and enhancing the rendering quality under a monocular setting.

- We showcase a companion compression scheme, supporting high-quality rendering with low storage, even under various platforms.

Our method achieves less training time, better rendering quantity, and less per-frame storage, which greatly balances the time consumption and the rendering quality.

2. Method

Utilizing RGBD monocular streams captured by a Kinect camera, DynTCG seamlessly integrates recent advancements in differentiable rasterization with tracking flow and deformable fields[10]. This innovative blend not only enhances the optimization process but also significantly improves the rendering quality. Moreover, our approach demonstrates remarkable performance in terms of storage efficiency. A visual summary of our methodology can be found in Fig. 1.

Our approach begins with the creation of an initial point cloud via depth back-projection, followed by the initialization of Gaussians. Concurrently, we project these Gaussians onto pixels and establish the warped relationship between Gaussian point clouds across frames using a pixel tracking method between consecutive frames. This results in the creation of warped Gaussians, which serve as a prior for the deformable network. The deformable network utilizes these warped Gaussians and time as inputs to estimate the changes in position and rotation of the Gaussians, thereby training temporal Gaussians. Furthermore, we introduce a novel compression scheme. This scheme significantly reduces storage needs, enabling the immersive viewing of high-fidelity human performances with a storage requirement of less than 2 MB per frame.

2.1. Flow tracking

Inspired by Instant-NVR [3], I aim to enhance the training speed and rendering effectiveness of the deform field by introducing a geometric constraint as a robust prior. While the ed node employed in Instant-NVR represents a powerful geometric prior, its integration into the Gaussian code proves to be challenging. Drawing further inspiration from the approach in dynamic nerf [2], which utilizes optical flow to provide prior information, we integrate optical flow into the Gaussian training pipeline. This involves calculating the δx and δy from optical flow derived from the 2D image, along with δz information from the depth image. Through projection and back-projection calculations, we obtain a prior for the deform net. We employ RAFT [9] as our method for optical flow tracking. RAFT comprises a Feature Extractor, Context Extractor, Visual Similarity Calculator, and Updater to optimize the flow tracking process. The pipeline

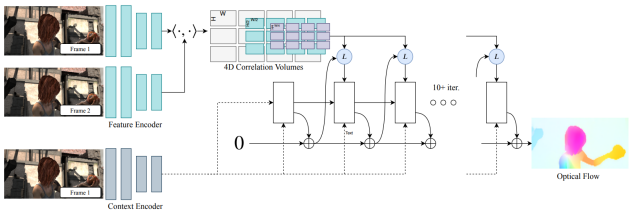

Figure 2. **Pipeline of RAFT**: containing Feature Extractor & Context Extractor, Visual Similarity Calculator and Updater

of RAFT is shown in Fig. 2.

Feature Extractor & Context Extractor extract features from two images, followed by the extraction of semantic information from one of the images.

Visual Similarity Calculator constructs a 4D correlation volume by calculating the dot product between feature vectors.

Updater updates and refines flow tracking by iteratively searching the correlation volume based on current optical flow estimates.

2.2. Compact 4D Gaussians

After Gaussian optimization, we dump the Gaussian splat points and the deformation network into separate frame Gaussian splat points. However, storage of 64-dimensional with a 3-order sh coefficient takes large disk storage, making it hard to store long dynamic sequences. To address this, we present a Residual Compression method as shown in Fig. 3, including residual computation, quantization, and RANS encoding.

Residual Compression For dynamic sequences, adjacent frames share similar Gaussian point parameters in a set of frames. We split the whole sequence into several segments and the object's movement is relatively small, then we compute the residual data between each frame and the first frame, such that the residual data distribution range of data will be much smaller. Then we apply compression separately to motion and appearance.

Quantization As for the residual data, we perform quantization separately to the delta position and delta appearance. As motion needs higher precision to maintain rendering quality, we adopt a lower quantization bit on the position attribute. We need to ensure the precision for the first frame, so we do not perform quantization to the first frame position and perform 9-bit quantization to appearance to ensure the base frame precision. For other frames in this segment, we adopt 11-bit quantization to delta position and 7-bit quantization to delta appearance.

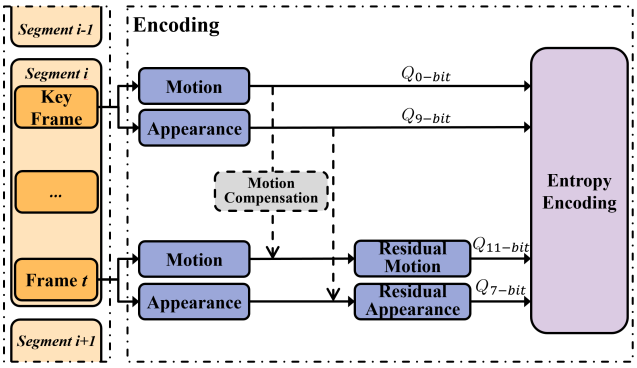

Figure 3. Illustration of residual encoding strategy for DynTCG.

RANS Encoding After quantization of the float number to an integer number, we perform RANS encoding to compress the data. This entropy encoding system encodes each attribute by calculating the numerical distribution frequency and finally encodes the attribute to an integer stream. By using our residual encoding method, we could compress the Gaussian point cloud efficiently by 20 times on the storage, which addresses long dynamic sequences on low disk storage and fast data transport.

3. Implementation Details

Firstly, for data obtained from HyperNeRF[7] and DynamicNeRF[2] datasets, we leverage the state-of-the-art single-image depth estimation[8] to estimate the input depth. Our approach to the deformable field is inspired by the work of Ziyi Yang[10] and is based on a broad framework of 3D Gaussians[4].

Regarding the training process, we conducted a total of 40,000 iterations. In the initial 3,000 iterations, we solely trained the 3D Gaussians to attain relatively stable positions and shapes. Following this phase, we commenced the joint training of both the 3D Gaussians and the deformation field. A single Adam optimizer[5], was utilized for optimization purposes. During compression, we first quantize the appearance attributes, then fix these parameters and fine-tune motion p and q of 4D Gaussians over an additional 1000 iterations. Afterward, we quantize the motion. To mitigate the influence of quantization on the outcomes, We apply different precision levels for various attributes to balance storage and quality.

All the experiments were done on 4 Nvidia 2080Ti.

4. Experiments

4.1. Ablation

Flow Tracking Prior We conduct a qualitative ablation on the flow tracking prior to assess its impact on rendering results. As shown in Tab. 1, omitting the flow tracking prior and relying solely on the deform net tends to prolong the training time. In contrast, our full pipeline enhances the preprocessing of images to establish flow between frames. This approach integrates depth and optical flow, serving as an efficient initialization method for the deform net, aiming to expedite the optimization process.

Figure 4. Qualitative comparison against monocular dynamic scene reconstruction methods

Figure 5. Comparison of residuals with and without the inclusion of scaling

Table 1. Quantitative evaluation of flow tracking prior.

	Training (min) ↓	PSNR(dB) ↑
Without Flow Tracking	93.23	31.65
Ours-full(before compression)	70.34	32.47
Ours-full(after compression)	70.64	32.30

Scaling In the Gaussian parameters (rotation, position, scaling) for shape and position, we found that optimizing the scaling parameter didn't notably improve rendering but increased training time. Consequently, in designing our deform-net, we omitted to optimize the Gaussian scaling. Results from training with and without scaling indicate a 9% time difference (with scale: 102 min, without scale: 93 min).

Furthermore, the exclusion of the scaling parameter had a negligible impact on the rendering performance, with a marginal decrease in PSNR by just 0.05. Visual comparison in Figure 5 also confirms that the residual difference between the render result and ground truth is minimal, with or without the scaling parameter in the model.

Residual Compensation As illustrated in Tab. 2, we allocate 46.23MB of storage for the 4D Gaussians of each frame before compression. Applying high-bit quantization (0-bit for motion and 9-bit for appearance) without residual compensation results in a storage requirement of 7.34MB. Using low-bit quantization (11-bit for motion and 7-bit for appearance), again without residual compensation, reduces storage to 3.23MB but compromises rendering quality. In contrast, applying the same low-bit quantization but with residual compensation significantly reduces storage needs to under 2MB per frame while maintaining the same level of rendering quality.

Table 2. Quantitative evaluation of residual compression methods.

	Per-frame Storage(MB) ↓	PSNR(dB) ↑
Raw Point Cloud	46.23	32.47
High-bit Quantization	7.34	32.23
Low-bit Quantization	3.23	29.56
Ours Residual Encoding	1.35	32.30

Table 3. Quantitative comparison of rendering results. Green and yellow cell colors indicate the best and the second-best results.

Method	PSNR ↑	SSIM ↑	Per-frame Storage(MB) ↓
Instant-NVR [3]	28.82	0.976	24.16
DynamicNeRF [2]	26.65	0.8452	21.5
TineuVox [1]	24.71	0.813	11.38
Ours(Before Compression)	**32.47**	**0.989**	**43.42**
Ours(After Compression)	**32.30**	**0.983**	**1.648**

4.2. Comparison

We compare DynTCG with the methods including Instant-NVR[3], DynamicNeRF[2], and TineuVox[1] on Dynamic Scene Dataset and our captured data. As depicted in Fig .4, we can observe that for areas with abundant texture and sharp edges, the reconstruction effect of Instant-NVR is mediocre, and its color prediction is also not very accurate. DynamicNeRF and Tinuvox show better overall object reconstruction, but their performance in texture detail is poor, resulting in significant blurring at the edges, to the extent that texture details become indiscernible. In contrast, our method DynTCG, based on Deformable 3D Gaussians, enhances its performance with depth map back-projection for point cloud re-projection constraints. Additionally, using Optical Flow Tracking as a prior provides a better initialization, leading to significantly improved results. Our method achieves better outcomes for complex textures, reconstructing them with greater clarity. Simultaneously, we also compare the average performance across different datasets in terms of PSNR, SSIM, and Per-frame Storage. It can be seen from the Tab. 3 that our metrics are superior to other algorithms in several aspects, further validating the robustness of our algorithm. The qualitative metrics prove that the rendered picture of our method surpasses other methods. What's more, the storage of our method can be reduced by about 90 percent compared with our methods, which ensures the easy preservation of each frame.

Figure 6. The results of our method

5. Results

The picture presented as Fig. 6 demonstrates the effectiveness of our method. The top-left data originates from the HyperNeRF Dataset, with the camera setting being a pinhole camera model with tangential and radial distortion. The data in the top right is from the Dynamic Scene Dataset, captured using a moving monocular camera; it represents short-term dynamic events ($\sim$ 5 seconds) recorded with a hand-held, moving camera. The bottom-left and bottom-right data are from D-Nerf, with the camera setting being a sparse set of images of a dynamic scene, captured by a monocular camera for sparse views by moving non-rigidly. Lastly, the data in the center bottom is from the Instant-NVR dataset, captured with an RGBD Kinect camera.

6. Limitations

Our method still has some limitations. The DynTCG takes RGBD images as input, which requires an RGBD camera to capture. Though we could use monodepth to estimate depth from RGB images, the error in depth estimation can lead to a bad effect on the back-projection module. Besides, our optical flow tracking will fail to deal with a non-continuous image input, limiting the data set to continuous captures.

7. Conclusions

We have presented a deformable Gaussian representation that takes monocular RGBD input and synthesizes a dynamic novel view. By utilizing optical flow tracking from 2D images into deformable dynamic Gaussian-splatting, our approach achieves high-fidelity rendering results, outperforming previous methods in terms of quality, training speed, and storage. Our optical flow and Gaussian point cloud wrap method provides a favorable prior, achieving a great acceleration on the deformable network training speed. We also present a residual encoding method to address the long sequence storage problem, by applying quantization and RANS Encoding on the residual point cloud, we achieve a compression rate of 25, making long dynamic sequences loading and training. Our experimental results show that our method outperforms other monocular methods under the same settings. We believe our method achieves a critical step in monocular dynamic novel view synthesis.

8. Acknowledgement and Contribution

We extend our gratitude to Yuheng Jiang for his assistance with the implementation of the Instant-NVR results in comparison experiments.

The contributions to this work are as follows: Yu Hong and Zhehao Shen developed the initial baseline through the design of the deform net. Zhehao Shen further en-

hanced this by integrating flow tracking as an effective prior. Shouchen Zhou contributed significant improvements to the deform net by removing redundant scale parameters and optimizing training speed. Penghao Wang introduced a novel approach to Gaussian compression using Quantization and RANS Encoding, as well as collaborating with Zhirui Zhang on the residual compression design. Zhirui Zhang also spearheaded the development of the dynamic viewer. Comparative and ablation experiments were conducted by Zhehao Shen, Yu Hong, Zhirui Zhang, Penghao Wang, and Shouchen Zhou, with Yu Hong taking the lead on poster design.

References

[1] 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*. ACM, 2022. 5

[2] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video, 2021. 1, 2, 3, 5

[3] Yuheng Jiang, Kaixin Yao, Zhuo Su, Zhehao Shen, Haimin Luo, and Lan Xu. Instant-nvr: Instant neural volumetric rendering for human-object interactions from monocular rgbd stream, 2023. 2, 5

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 3

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[6] 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, 2020. 1

[7] 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. 3

[8] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022. 3

[9] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision – ECCV 2020*, pages 402–419, Cham, 2020. Springer International Publishing. 2

[10] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2, 3