NeRF与3DGS在动态三维重建中的应用

姚遥

南京大学 NJU-3DV

南京大学 NANJING UNIVERSITY

动态三维重建

静态三维重建

动态三维重建

2D Observations → 3D

2D Observations → 4D

动态三维重建 | 关键挑战

动态三维重建的关键挑战：

- 高效连续的四维空间表达方式
- 图像观测不足，重建求解欠定
- …

动态三维重建

课程大纲

- 基于四维空间 (4D Representation) 的动态三维重建方法
- 基于形变场 (Deformation Field) 的动态三维重建方法
- 基于逐帧重建 (Per-frame Reconstruction) 的动态三维重建方法
- 基于模型先验 (Model Priors) 的动态三维重建方法

4D Representations

- **NeRF Related**: DyNeRF
- **Triplane Related**: K-Planes, HexPlane, Tensor4D
- **3DGS Related**: 4DGS, 4D-Rotor GS

Neural 3D Video Synthesis from Multi-view Video

CVPR 2022 (oral)

Tianye Li[1,3,*] Mira Slavcheva[1,*] Michael Zollhoefer[1]
Simon Green[1] Christoph Lassner[1] Changil Kim[2] Tanner Schmidt[1]
Steven Lovegrove[1] Michael Goesele[1] Richard Newcombe[1] Zhaoyang Lv[1]
[1]Reality Labs Research [2]Meta [3]University of Southern California

K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. Fridovich-Keil et al. CVPR 2023.

4D Representations | 2D Planes | DyNeRF

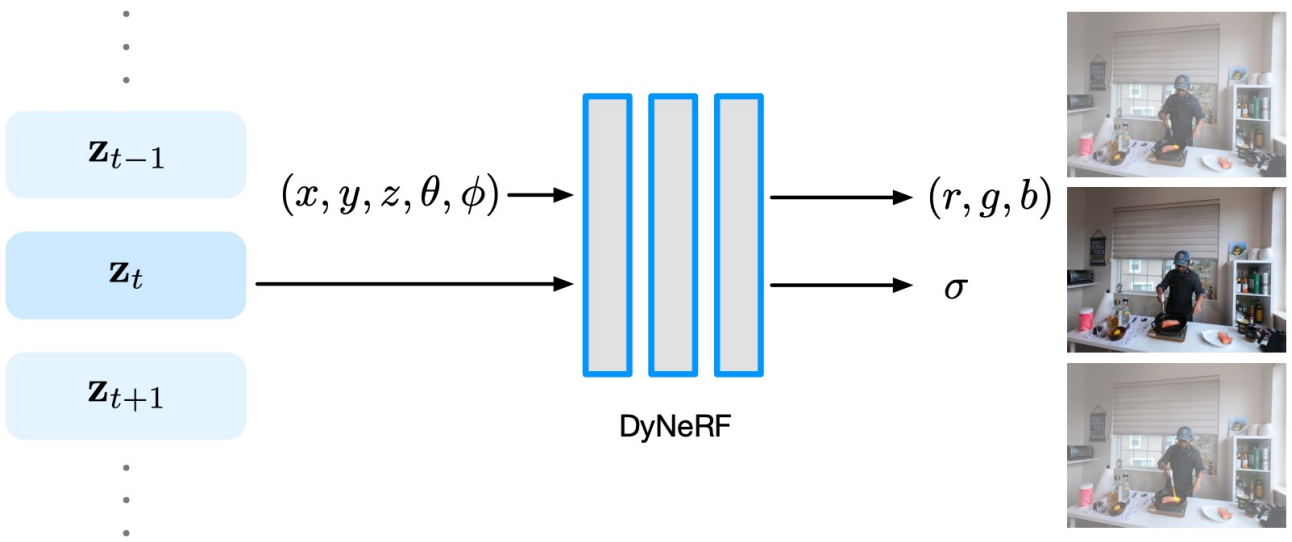


Table 1. **Quantitative comparison** of our proposed method to baselines of existing methods and radiance field baselines trained at 200K iterations on a 10-second sequence.

Method	PSNR ↑	MSE ↓	DSSIM ↓	LPIPS ↓	FLIP ↓
MVS	19.1213	0.01226	0.1116	0.2599	0.2542
NeuralVolumes	22.7975	0.00525	0.0618	0.2951	0.2049
LLFF	23.2388	0.00475	0.0762	0.2346	0.1867
NeRF-T	28.4487	0.00144	0.0228	0.1000	0.1415
DyNeRF†	28.4994	0.00143	0.0231	0.0985	0.1455
DyNeRF	**29.5808**	**0.00110**	**0.0197**	**0.0832**	**0.1347**

K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. Fridovich-Keil et al. CVPR 2023.

4D Representations | 2D Planes | K-Planes

K-Planes: Explicit Radiance Fields in Space, Time, and Appearance

Sara Fridovich-Keil*
UC Berkeley
sfk@berkeley.edu

Giacomo Meanti*
Istituto Italiano di Tecnologia
giacomo.meanti@iit.it

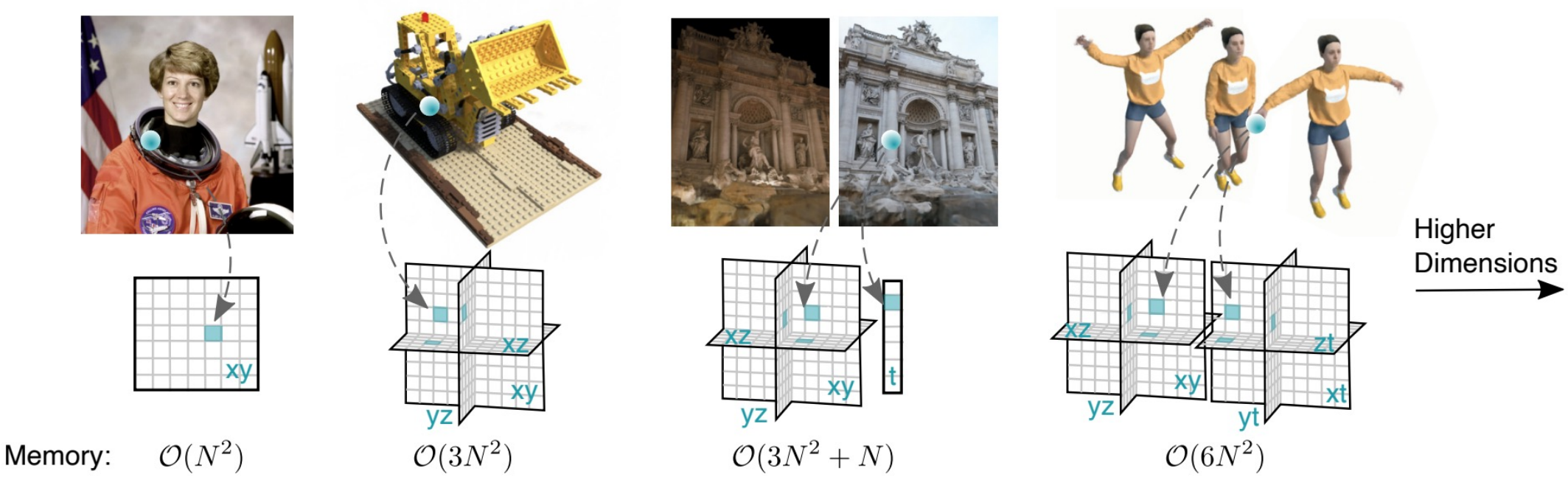
Frederik Rahbæk Warburg
Technical University of Denmark
frwa@dtu.dk

Benjamin Recht
UC Berkeley
brecht@berkeley.edu

Angjoo Kanazawa
UC Berkeley
kanazawa@berkeley.edu

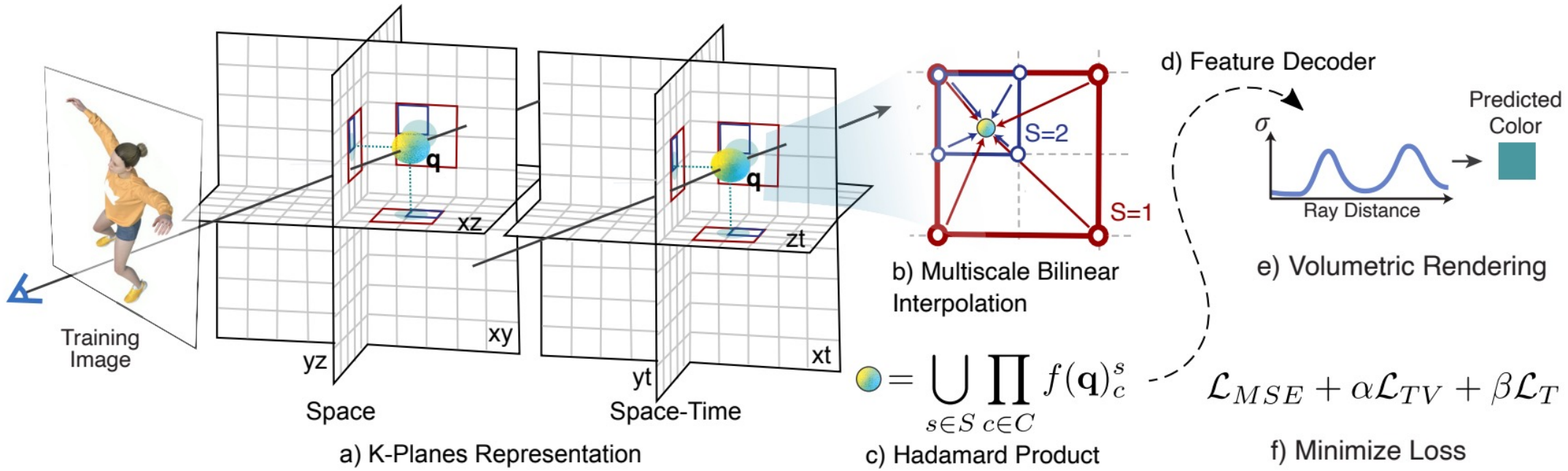
K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. Fridovich-Keil et al. CVPR 2023.

4D Representations | 2D Planes | K-Planes



K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. Fridovich-Keil et al. CVPR 2023.

4D Representations | 2D Planes | K-Planes

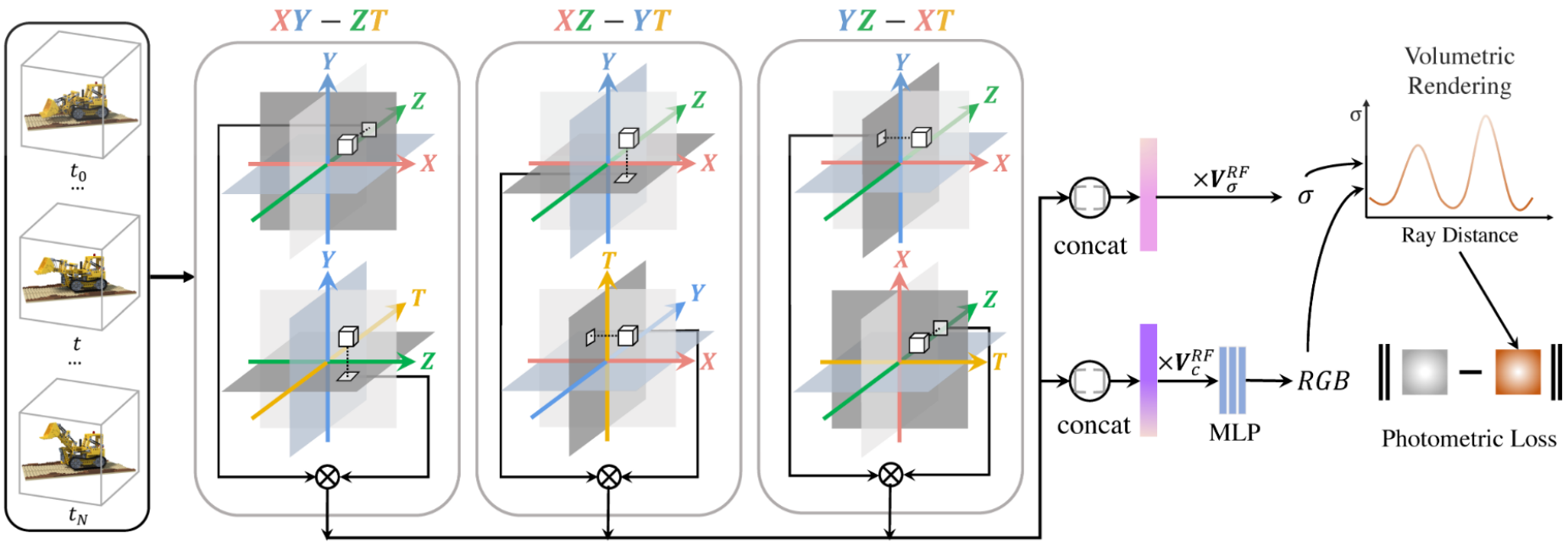


K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. Fridovich-Keil et al. CVPR 2023.

4D Representations | 2D Planes | HexPlane

HexPlane: A Fast Representation for Dynamic Scenes

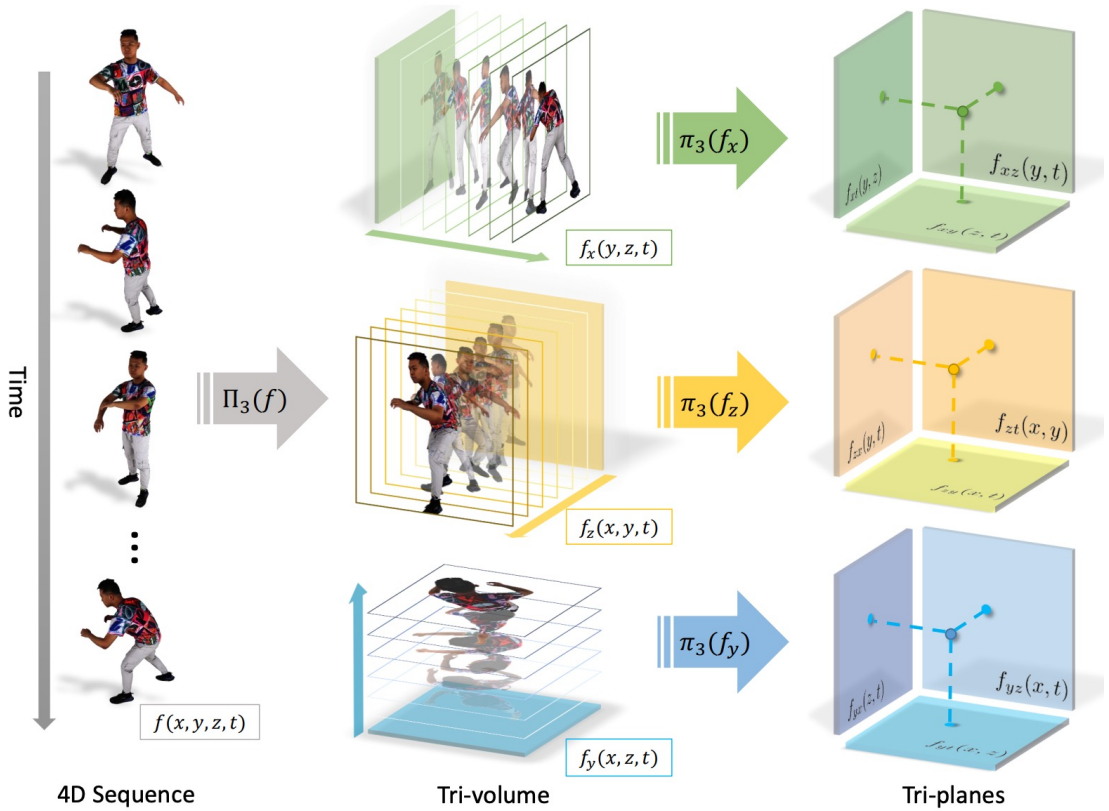
Ang Cao, Justin Johnson
University of Michigan
CVPR 2023



HexPlane: A Fast Representation for Dynamic Scenes. Cao et al. CVPR 2023.

南京大学 NANJING UNIVERSITY

4D Representations | 2D Planes | Tensor4D



- 4D space as 3D volumes

$$\Pi_3(f(x,y,z,t)) = \{f_z(x,y,t), f_y(x,z,t), f_x(y,z,t)\}$$

- 3D volume as 2D maps

$$\pi_3(f_z) = \{f_{zt}(x,y), f_{zy}(x,t), f_{zx}(y,t)\}$$
$$\pi_3(f_y) = \{f_{yt}(x,z), f_{yz}(x,t), f_{yx}(z,t)\}$$
$$\pi_3(f_x) = \{f_{xt}(y,z), f_{xz}(y,t), f_{xy}(z,t)\}$$

- Deeper correlation with t axis

$$(xt, yt, zt) \longrightarrow (xt, yt, zt, xyt, yzt, xzt, xyzt)$$

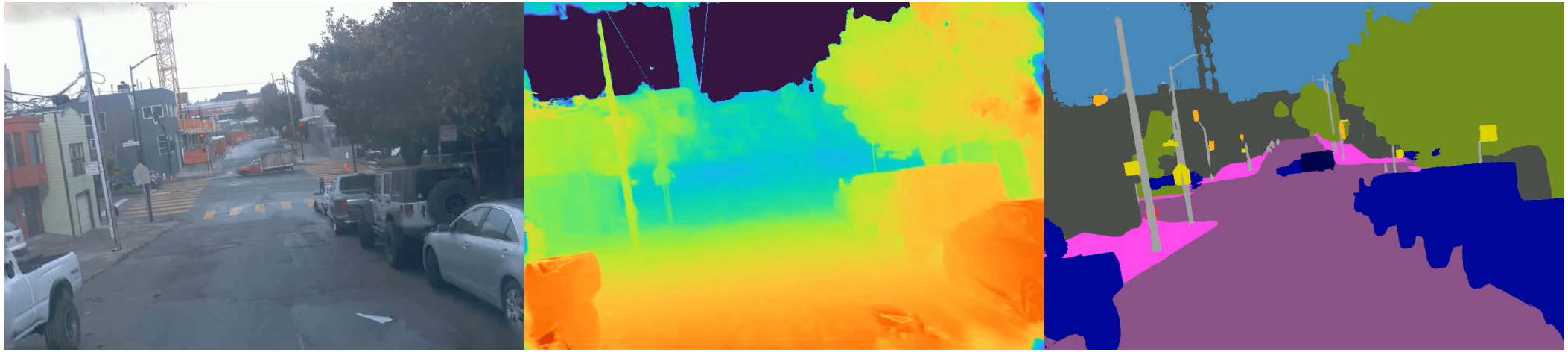
Tensor4D : Efficient Neural 4D Decomposition for High-fidelity Dynamic Reconstruction and Rendering. Shao et al. CVPR 2023.

南京大学 NANJING UNIVERSITY

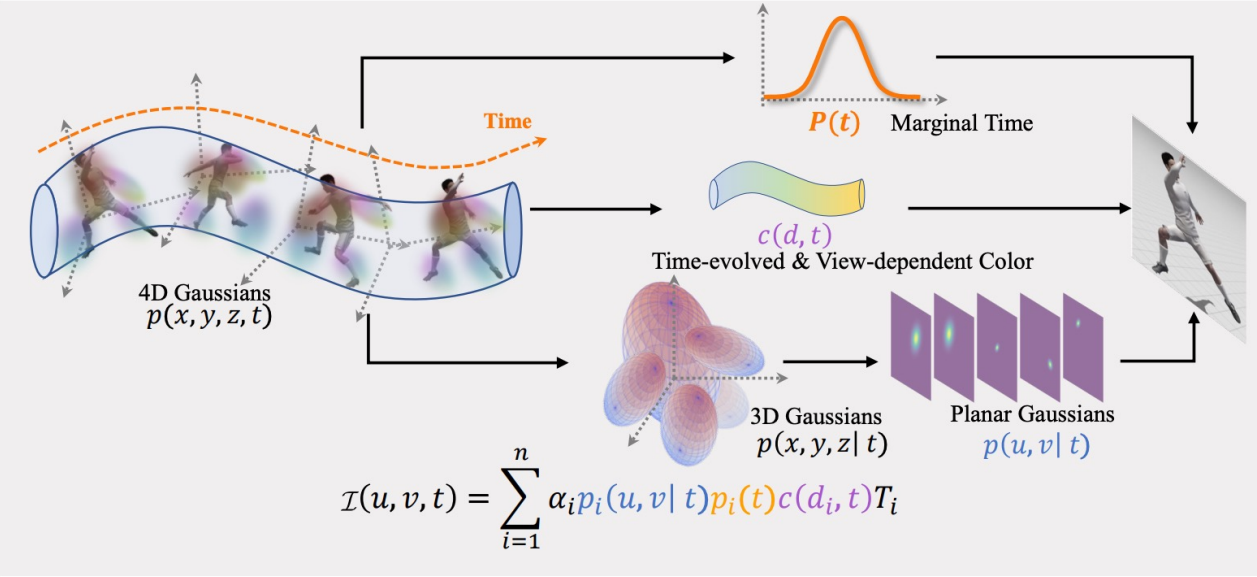
Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting

ICLR 2024

Zeyu Yang, Hongye Yang, Zijie Pan, Li Zhang
Fudan University



4D Representations | Gaussians | 4DGS



Method	PSNR ↑	DSSIM ↓	LPIPS ↓	FPS ↑
- Plenoptic Video (real, multi-view)				
Neural Volumes (Lombardi et al., 2019)[1]	22.80	0.062	0.295	-
LLFF (Mildenhall et al., 2019)[1]	23.24	0.076	0.235	-
DyNeRF (Li et al., 2022b)[1]	29.58	0.020	0.099	0.015
HexPlane (Cao & Johnson, 2023)	31.70	0.014	0.075	0.56[3]
K-Planes-explicit (Fridovich-Keil et al., 2023)	30.88	0.020	-	0.23[3]
K-Planes-hybrid (Fridovich-Keil et al., 2023)	31.63	0.018	-	-
MixVoxels-L (Wang et al., 2023)	30.80	0.020	0.126	16.7
StreamRF (Li et al., 2022a)[1]	29.58	-	-	8.3
NeRFPlayer (Song et al., 2023)	30.69	0.035[2]	0.111	0.045
HyperReel (Attal et al., 2023)	31.10	0.037[2]	0.096	2.00
4DGS (Wu et al., 2023)[4]	31.02	0.030	0.150	36
4DGS (Ours)	**32.01**	**0.014**	**0.055**	**114**

- Defining a Gaussian point in 4D space

Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. Yang et al. ICLR 2024.

南京大学 NANJING UNIVERSITY

4D-Rotor Gaussian Splatting: Towards Efficient Novel View Synthesis for Dynamic Scenes

Yuanxing Duan*
Peking University
China
mjdyx@pku.edu.cn

Fangyin Wei*
Princeton University
USA
fwei@princeton.edu

Qiyu Dai
Peking University
China
State Key Laboratory of General AI
China
qiyudai@pku.edu.cn

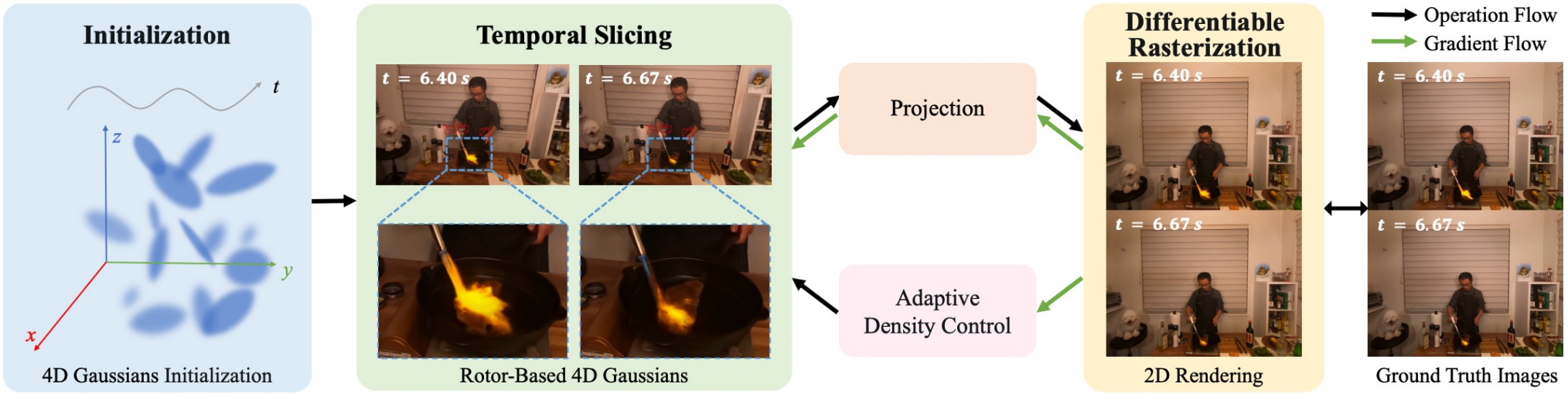
Yuhang He
Peking University
China
2100014725@stu.pku.edu.cn

Wenzheng Chen†
Peking University
China
NVIDIA
Canada
wenzhengchen@pku.edu.cn

Baoquan Chen†
Peking University
China
State Key Laboratory of General AI
China
baoquan@pku.edu.cn

- Input: multi-view video
- Output: 4D Gaussian

4D Representations | Gaussians | 4D-Rotor GS



Method	PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑
D-NeRF [Pumarola et al. 2021]	29.17	0.95	0.07	24 h	0.13
TiNeuVox [Fang et al. 2022]	32.87	0.97	0.04	28 min	1.60
K-Planes [Fridovich-Keil et al. 2023]	31.07	0.97	0.02	54 min	1.20
Deformable3DGS [Yang et al. 2023]	39.31	0.99	0.01	26 min	85.45
Deformable4DGS [Wu et al. 2023]	32.99	0.97	0.05	13 min	104.00
RealTime4DGS [Yang et al. 2024]	32.71	0.97	0.03	10 min	289.07
Ours	34.26	0.97	0.03	5 min	1257.63

$$G_{4D}(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{4D})^T \Sigma_{4D}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{4D})}.$$

$$\Sigma_{4D} = \mathbf{R}_{4D}\mathbf{S}_{4D}\mathbf{S}_{4D}^T\mathbf{R}_{4D}^T.$$

$$\Sigma_{4D} = \begin{pmatrix} \mathbf{U} & \mathbf{V} \\ \mathbf{V}^T & \mathbf{W} \end{pmatrix} \text{ and } \Sigma_{4D}^{-1} = \begin{pmatrix} \mathbf{A} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{Z} \end{pmatrix},$$

4D-Rotor Gaussian Splatting: Towards Efficient Novel View Synthesis for Dynamic Scenes. Duan et al. **SIGGRAPH 2024**

南京大学 NANJING UNIVERSITY

Deformation Fields

- **NeRF Related**: D-NeRF, Nerfies, Dynamic NeRF
- **Explicit NeRF Related**: MixVoxel, NeRFPlayer
- **3DGS Related**: 4D-GS, Deformable 3DGS, Gaussian-Flow, Spacetime GS, SC-GS

Deformation Fields | NeRFs | D-NeRF

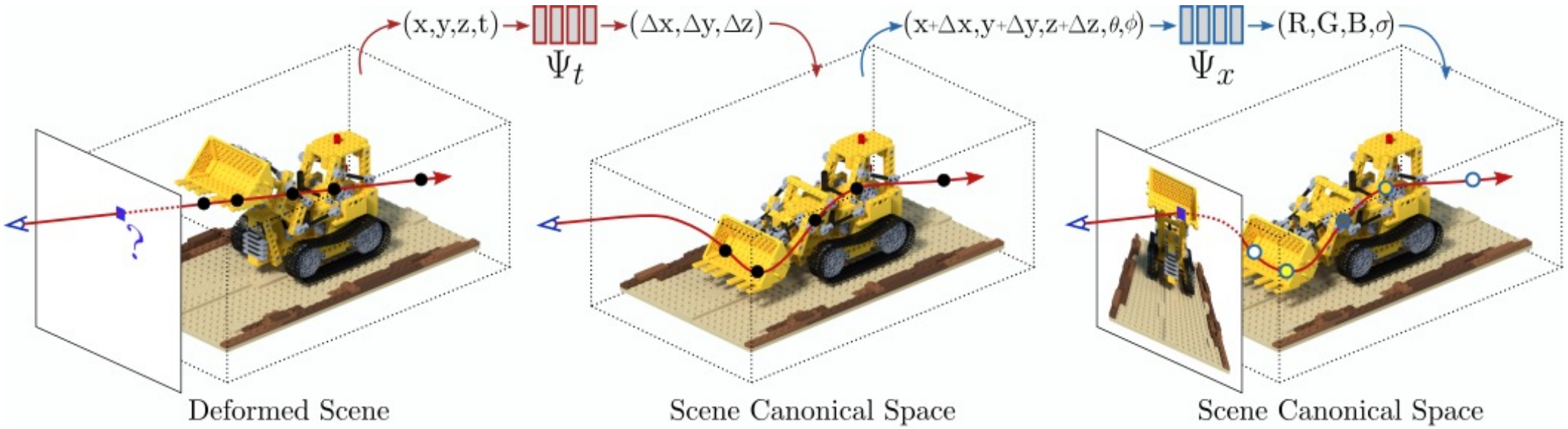
D-NeRF: Neural Radiance Fields for Dynamic Scenes

Albert Pumarola[1] Enric Corona[1] Gerard Pons-Moll[2,3] Francesc Moreno-Noguer[1]
[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC
[2]University of Tübingen
[3]Max Planck Institute for Informatics

- Input: monocular video
- Output: radiance & deformation fields

D-NeRF: Neural Radiance Fields for Dynamic Scenes. Pumarola et al. CVPR 2020

Deformation Fields | NeRFs | D-NeRF



Deformation Field: $\Psi_t : (\mathbf{x}, t) \rightarrow \Delta\mathbf{x}$

Radiance Field: $\Psi_x : (\mathbf{x} + \Delta\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$

D-NeRF: Neural Radiance Fields for Dynamic Scenes. Pumarola et al. CVPR 2020

Deformation Fields | NeRFs | Nerfies

Nerfies: Deformable Neural Radiance Fields

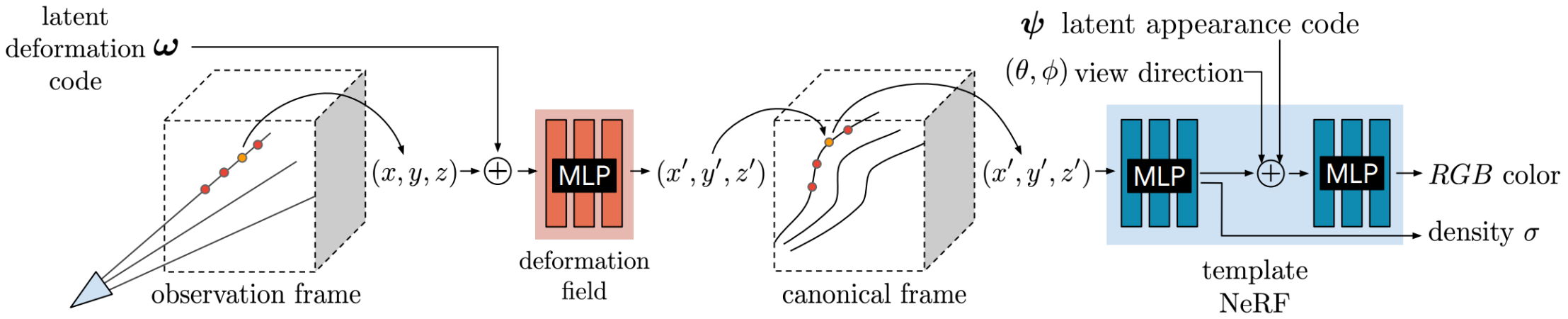
Keunhong Park[1*] Utkarsh Sinha[2] Jonathan T. Barron[2] Sofien Bouaziz[2]
Dan B Goldman[2] Steven M. Seitz[1,2] Ricardo Martin-Brualla[2]
[1]University of Washington [2]Google Research

(a) Capture Process (b) Input (c) Nerfie (d) Nerfie Depth

- Input: multi-view images of non-rigid scenes
- Output: radiance & deformation fields

Nerfies: Deformable Neural Radiance Fields. Park et al. ICCV 2021.

Deformation Fields | NeRFs | Nerfies



Deformation Field: $T : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow \mathbf{x}'$

(with per-frame deformation code)

Radiance Field: $F : (\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_i) \rightarrow (\mathbf{c}, \sigma)$

(with per-frame appearance code)

Nerfies: Deformable Neural Radiance Fields. Park et al. ICCV 2021.

Deformation Fields | NeRFs | Dynamic NeRF

Dynamic View Synthesis from Dynamic Monocular Video

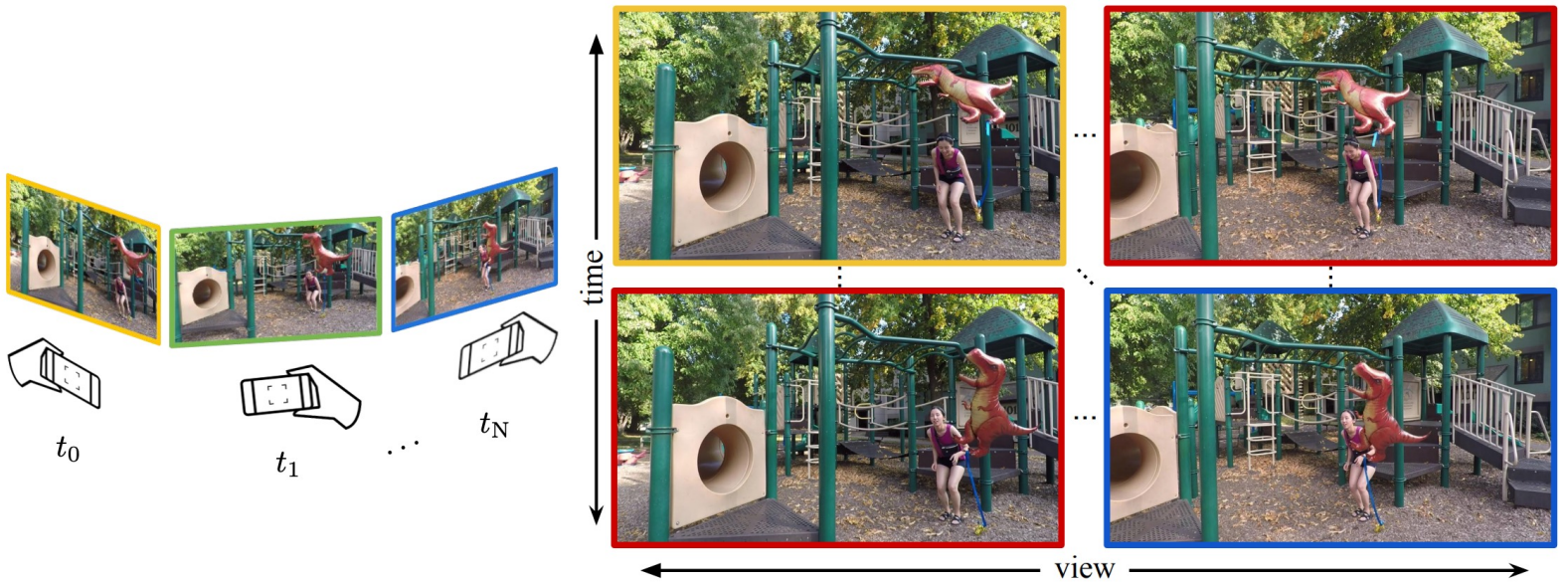
Chen Gao
Virginia Tech
chengao@vt.edu

Ayush Saraf
Facebook
ayush29feb@fb.com

Johannes Kopf
Facebook
jkopf@fb.com

Jia-Bin Huang
Virginia Tech
jbhuang@vt.edu

https://free-view-video.github.io

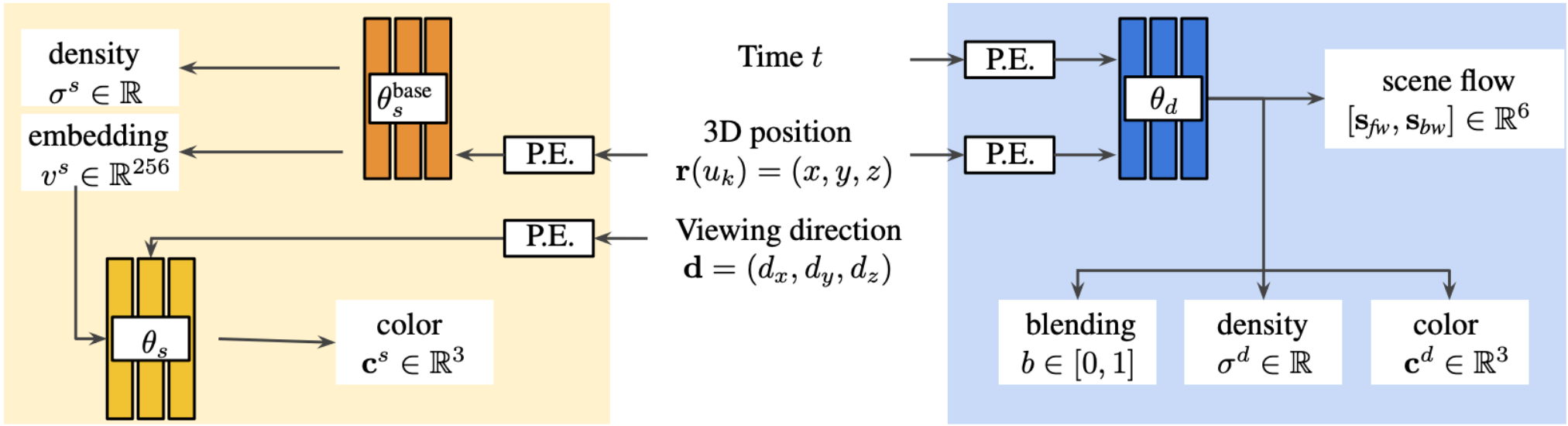


- Input: monocular video
- Output: static NeRF + dynamic NeRF with scene flow

Dynamic View Synthesis from Dynamic Monocular Video. Gao et al. ICCV 2021.

南京大学 NANJING UNIVERSITY

Deformation Fields | NeRFs | Dynamic NeRF



(a) Static NeRF (Section 3.2)

(b) Dynamic NeRF (Section 3.3)

Scene Flow and Radiance Field: (for dynamic foreground areas)

$$\left(\mathbf{s}_{fw}, \mathbf{s}_{bw}, \sigma_t^d, \mathbf{c}_t^d, b\right) = \mathrm{MLP}_{\theta_d}\left(\mathbf{r}(u_k), t\right)$$

Static Radiance Field: (for static background areass)

$$\left(\sigma^s, \mathbf{c}^s\right) = \mathrm{MLP}_{\theta}\left(\mathbf{r}(u_k)\right)$$

Nerfies: Deformable Neural Radiance Fields. Park et al. ICCV 2021.

Deformation Fields | Voxel | MixVoxel

Mixed Neural Voxels for Fast Multi-view Video Synthesis

Feng Wang[1] Sinan Tan[1] Xinghang Li[1] Zeyue Tian[2] Yafei Song[3] Huaping Liu[1] *

[1]Beijing National Research Center for Information Science and Technology(BNRist),
Department of Computer Science and Technology, Tsinghua University
[2]Hong Kong University of Science and Technology
[3] XR Lab, DAMO Academy, Alibaba Group

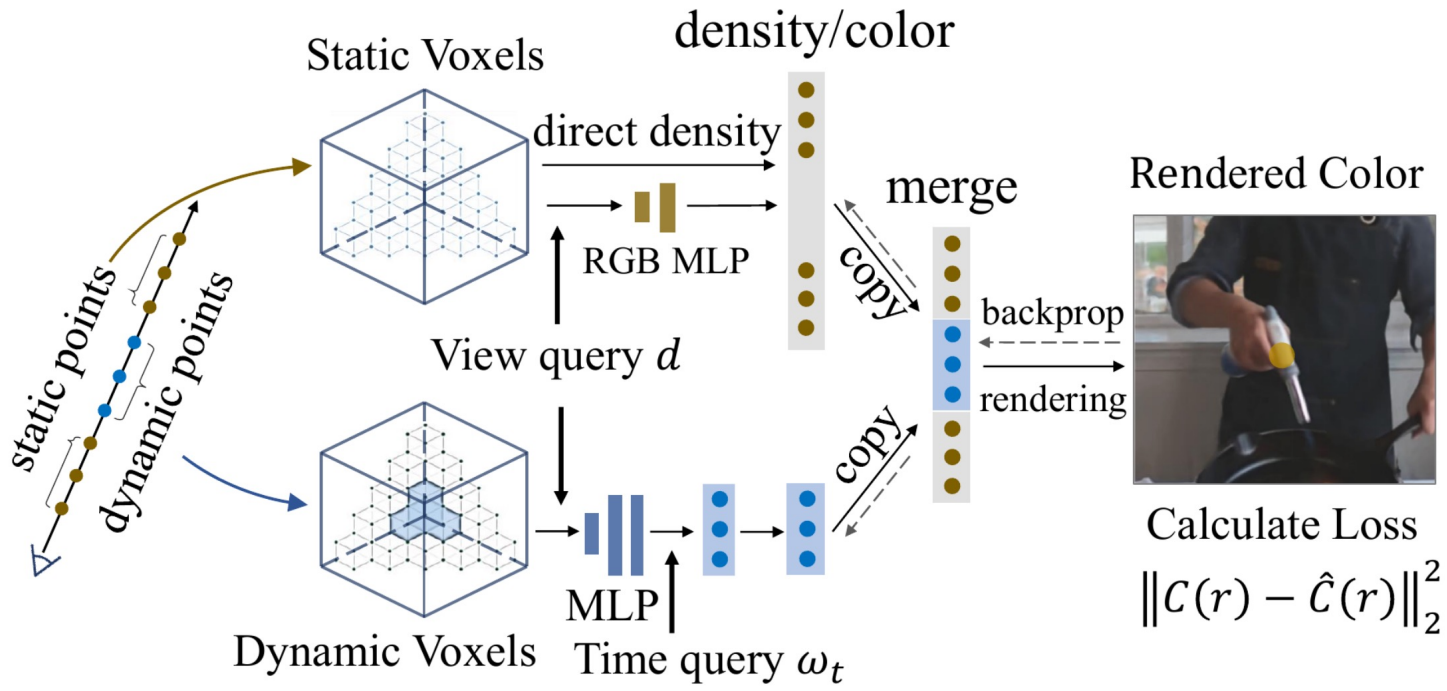
wang-f20@mails.tsinghua.edu.cn, hpliu@tsinghua.edu.cn

- Input: monocular/multi-view video
- Output: static voxels + dynamic voxels

Mixed Neural Voxels for Fast Multi-view Video Synthesis. Wang et al. ICCV 2023.

南京大学 NANJING UNIVERSITY

Deformation Fields | Voxel | MixVoxel



Static Radiance Field:
(for static background areas)

$$\sigma(x, y, z) = \mathcal{S}^{\sigma}_{x,y,z}, \quad c(x, y, z, \boldsymbol{d}) = \mathcal{C}_{\theta}(\mathcal{S}^{c}_{x,y,z}, \boldsymbol{d}).$$

Dynamic Radiance Field:
(for dynamic foreground areas)

$$\sigma(x, y, z, t) = \omega^{\sigma}_{t} \cdot \mathcal{T}^{\sigma}_{\theta_1}(\mathcal{G}^{\sigma}_{x,y,z}),$$

$$c(x, y, z, \boldsymbol{d}, t) = \omega^{c}_{t} \cdot \mathcal{T}^{c}_{\theta_2}(\mathcal{G}^{c}_{x,y,z}, \boldsymbol{d}).$$

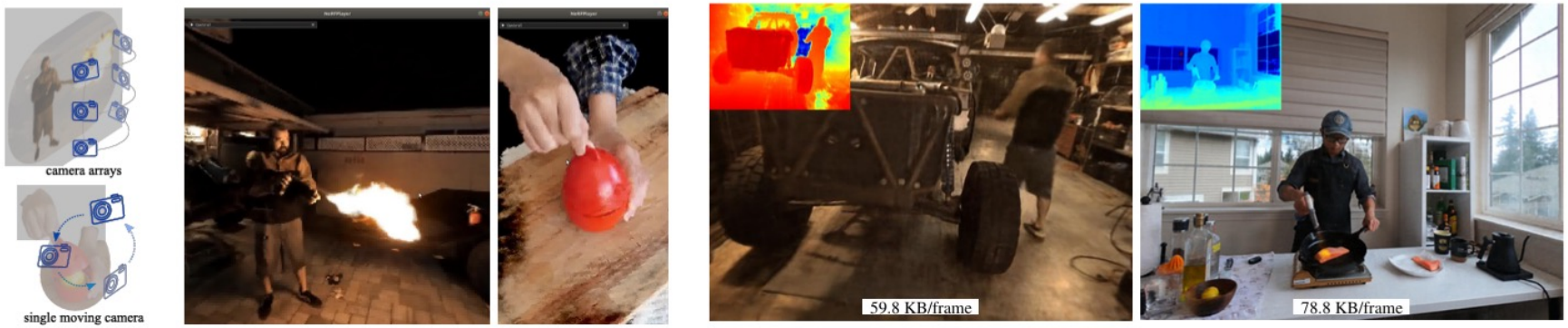
Mixed Neural Voxels for Fast Multi-view Video Synthesis. Wang et al. ICCV 2023.

南京大学 NANJING UNIVERSITY

Deformation Fields | Instant-NPG | NeRFPlayer

NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields

Liangchen Song[1] Anpei Chen[2,4] Zhong Li[3] Zhang Chen[3] Lele Chen[3]
Junsong Yuan[1] Yi Xu[3] Andreas Geiger[4]
[1]University at Buffalo [2]ETH Zürich
[3]OPPO US Research Center, InnoPeak Tech [4]University of Tübingen
https://bit.ly/nerfplayer



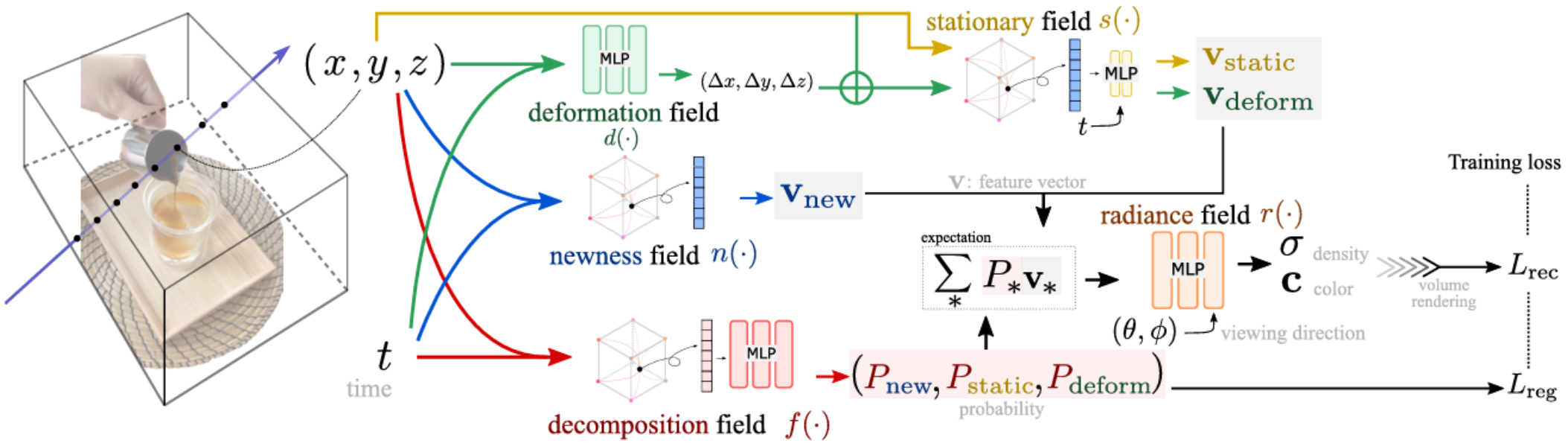
(a) Inputs (b) Real-time rendering (c) Low bitrate streaming

- Input: monocular/multi-view video
- Output: static NeRF + deform MLP + new MLP

NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. Song et al. ICCV 2021.

南京大学 NANJING UNIVERSITY

Deformation Fields | Instant-NPG | NeRFPlayer



Deformation Field: $d(\cdot) : (\boldsymbol{p}, t) \mapsto (\Delta \boldsymbol{p}).$

$$f(\cdot) : (\boldsymbol{p}, t) \mapsto (P_{\text{static}}, P_{\text{deform}}, P_{\text{new}}),$$

$$\boldsymbol{v} = \sum_* P_* \boldsymbol{v}_*, * \in \{\text{static}, \text{deform}, \text{new}\}.$$

NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. Song et al. ICCV 2021.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | Deformable 3DGS

Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction

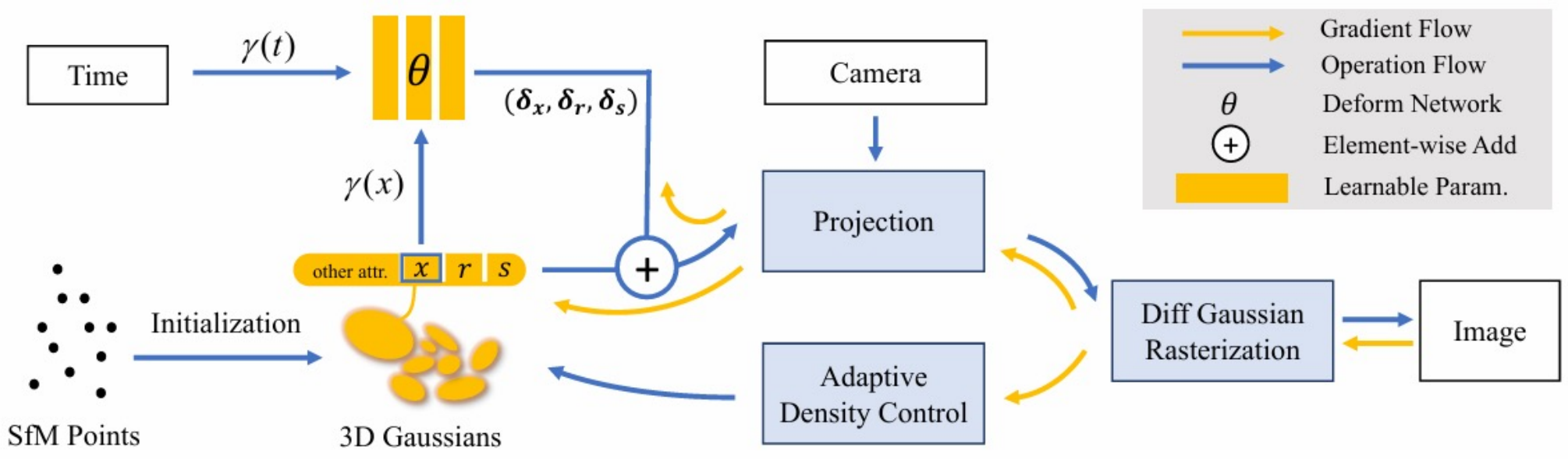
Ziyi Yang[1,2] Xinyu Gao[1] Wen Zhou[2] Shaohui Jiao[2] Yuqing Zhang[1] Xiaogang Jin[1†]

[1]State Key Laboratory of CAD&CG, Zhejiang University [2]ByteDance Inc.

- Input: multi-view/monocular video
- Output: dynamic gaussian

Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. Yang et al. CVPR 2024.

Deformation Fields | Gaussians | Deformable 3DGS

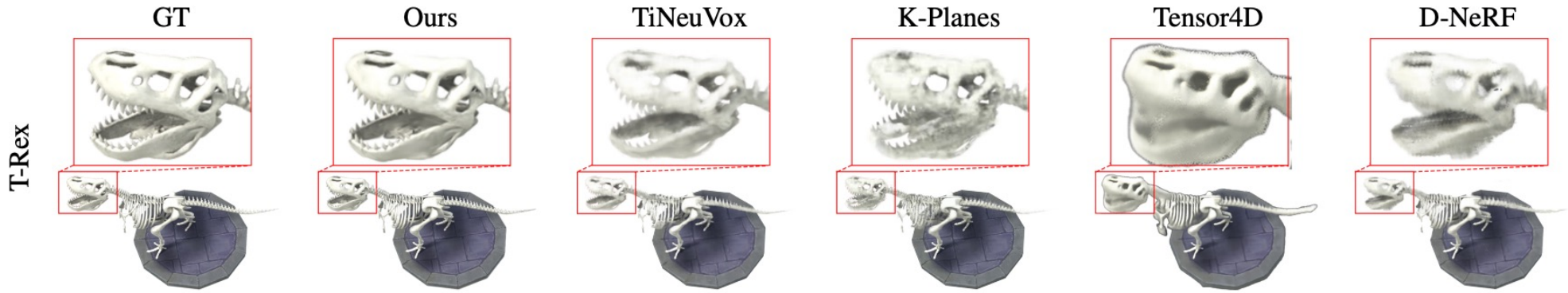


Deformation Field: $(\delta \boldsymbol{x}, \delta \boldsymbol{r}, \delta \boldsymbol{s}) = \mathcal{F}_\theta(\gamma(\mathrm{sg}(\boldsymbol{x})), \gamma(t))$

Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. Yang et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | Deformable 3DGS



Method	Hell Warrior PSNR↑	Hell Warrior SSIM↑	Hell Warrior LPIPS↓	Mutant PSNR↑	Mutant SSIM↑	Mutant LPIPS↓	Hook PSNR↑	Hook SSIM↑	Hook LPIPS↓	Bouncing Balls PSNR↑	Bouncing Balls SSIM ↑	Bouncing Balls LPIPS↓
3D-GS	29.89	0.9155	0.1056	24.53	0.9336	0.0580	21.71	0.8876	0.1034	23.20	0.9591	0.0600
D-NeRF	24.06	0.9440	0.0707	30.31	0.9672	0.0392	29.02	0.9595	0.0546	38.17	0.9891	0.0323
TiNeuVox	27.10	0.9638	0.0768	31.87	0.9607	0.0474	30.61	0.9599	0.0592	40.23	0.9926	0.0416
Tensor4D	31.26	0.9254	0.0735	29.11	0.9451	0.0601	28.63	0.9433	0.0636	24.47	0.9622	0.0437
K-Planes	24.58	0.9520	0.0824	32.50	0.9713	0.0362	28.12	0.9489	0.0662	40.05	0.9934	0.0322
Ours	41.54	0.9873	0.0234	42.63	0.9951	0.0052	37.42	0.9867	0.0144	41.01	0.9953	0.0093

Method	Lego PSNR↑	Lego SSIM↑	Lego LPIPS↓	T-Rex PSNR↑	T-Rex SSIM↑	T-Rex LPIPS↓	Stand Up PSNR↑	Stand Up SSIM↑	Stand Up LPIPS↓	Jumping Jacks PSNR↑	Jumping Jacks SSIM ↑	Jumping Jacks LPIPS↓
3D-GS	22.10	0.9384	0.0607	21.93	0.9539	0.0487	21.91	0.9301	0.0785	20.64	0.9297	0.0828
D-NeRF	25.56	0.9363	0.0821	30.61	0.9671	0.0535	33.13	0.9781	0.0355	32.70	0.9779	0.0388
TiNeuVox	26.64	0.9258	0.0877	31.25	0.9666	0.0478	34.61	0.9797	0.0326	33.49	0.9771	0.0408
Tensor4D	23.24	0.9183	0.0721	23.86	0.9351	0.0544	30.56	0.9581	0.0363	24.20	0.9253	0.0667
K-Planes	28.91	0.9695	0.0331	30.43	0.9737	0.0343	33.10	0.9793	0.0310	31.11	0.9708	0.0468
Ours	33.07	0.9794	0.0183	38.10	0.9933	0.0098	44.62	0.9951	0.0063	37.72	0.9897	0.0126

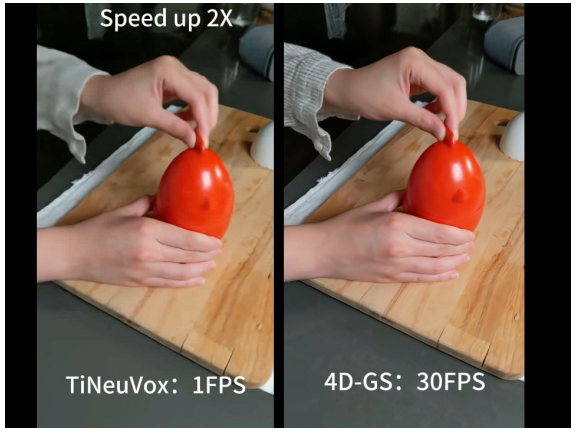
Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. Yang et al. CVPR 2024.

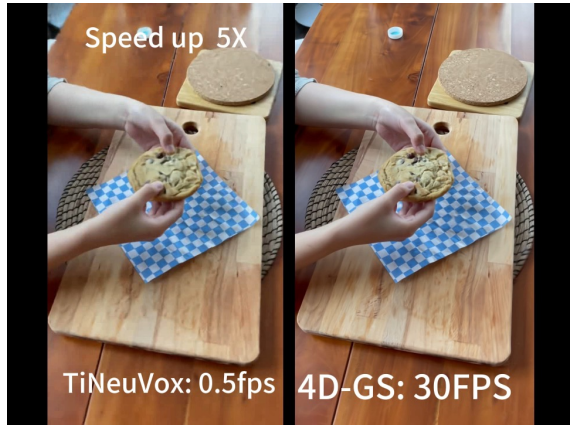
南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | 4D-GS

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering

Guanjun Wu[1]*, Taoran Yi[2]*, Jiemin Fang[3]†, Lingxi Xie[3], Xiaopeng Zhang[3],
Wei Wei[1], Wenyu Liu[2], Qi Tian[3], Xinggang Wang[2]†‡
[1]School of CS, Huazhong University of Science and Technology
[2]School of EIC, Huazhong University of Science and Technology [3]Huawei Inc.

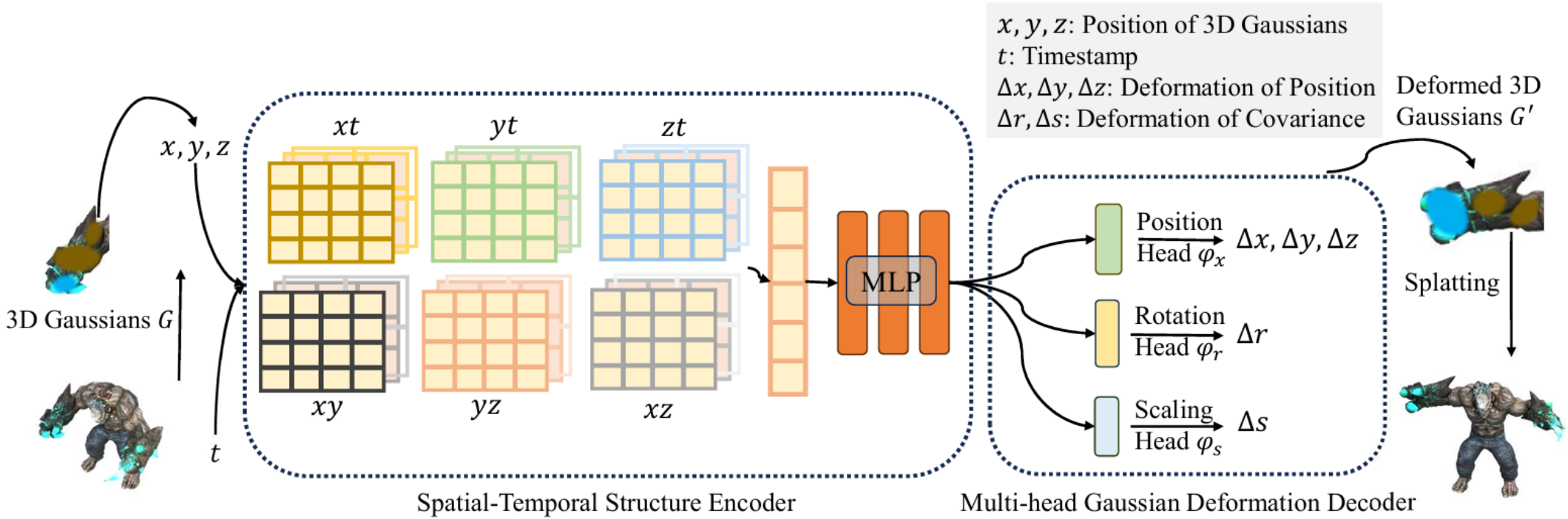




- Input: multi-view/monocular video
- Output: dynamic gaussian

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. Wu et al. CVPR 2024.

Deformation Fields | Gaussians | 4D-GS



Deformation Field:

$$f_h = \bigcup_l \prod \text{interp}(R_l(i,j)),$$

$$(i,j) \in \{(x,y),(x,z),(y,z),(x,t),(y,t),(z,t)\}.$$

$$f_d = \phi_d(f_h). \quad \Delta\mathcal{X} = \phi_x(f_d) \quad \Delta r = \phi_r(f_d) \quad \Delta s = \phi_s(f_d)$$

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. Wu et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | Gaussian-Flow

Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle

Youtian Lin[1] Zuozhuo Dai[2] Siyu Zhu[3] Yao Yao[1]✉
[1]Nanjing University [2]Alibaba Group [3]Fudan University

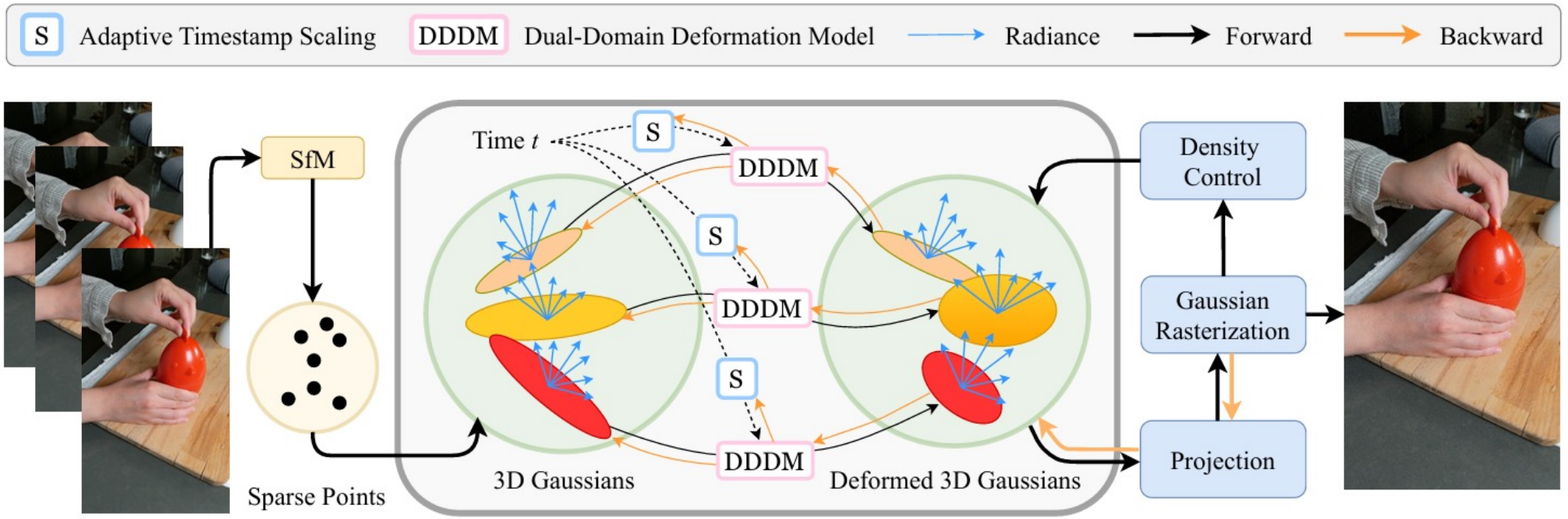


- Input: multi-view/monocular video
- Output: dynamic gaussian

Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. Lin et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | Gaussian-Flow



Deformation Field: $D(t) = P_N(t) + F_L(t) \quad P_N(t) = \sum_{n=0}^{N} a_n t^n, \quad F_L(t) = \sum_{l=1}^{L} \left(f_{sin}^l \cos(lt) + f_{cos}^l \sin(lt) \right).$

Regularizations: $\mathcal{L}_t = \|D(t) - D(t+\epsilon)\|_2. \quad \mathcal{L}_s = \sum_{j \in \mathcal{N}_i} \|D(t)_i - D(t)_j\|_2$

- Explicit motion modeling
- As fast as static 3DGS

Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. Lin et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | Gaussian-Flow

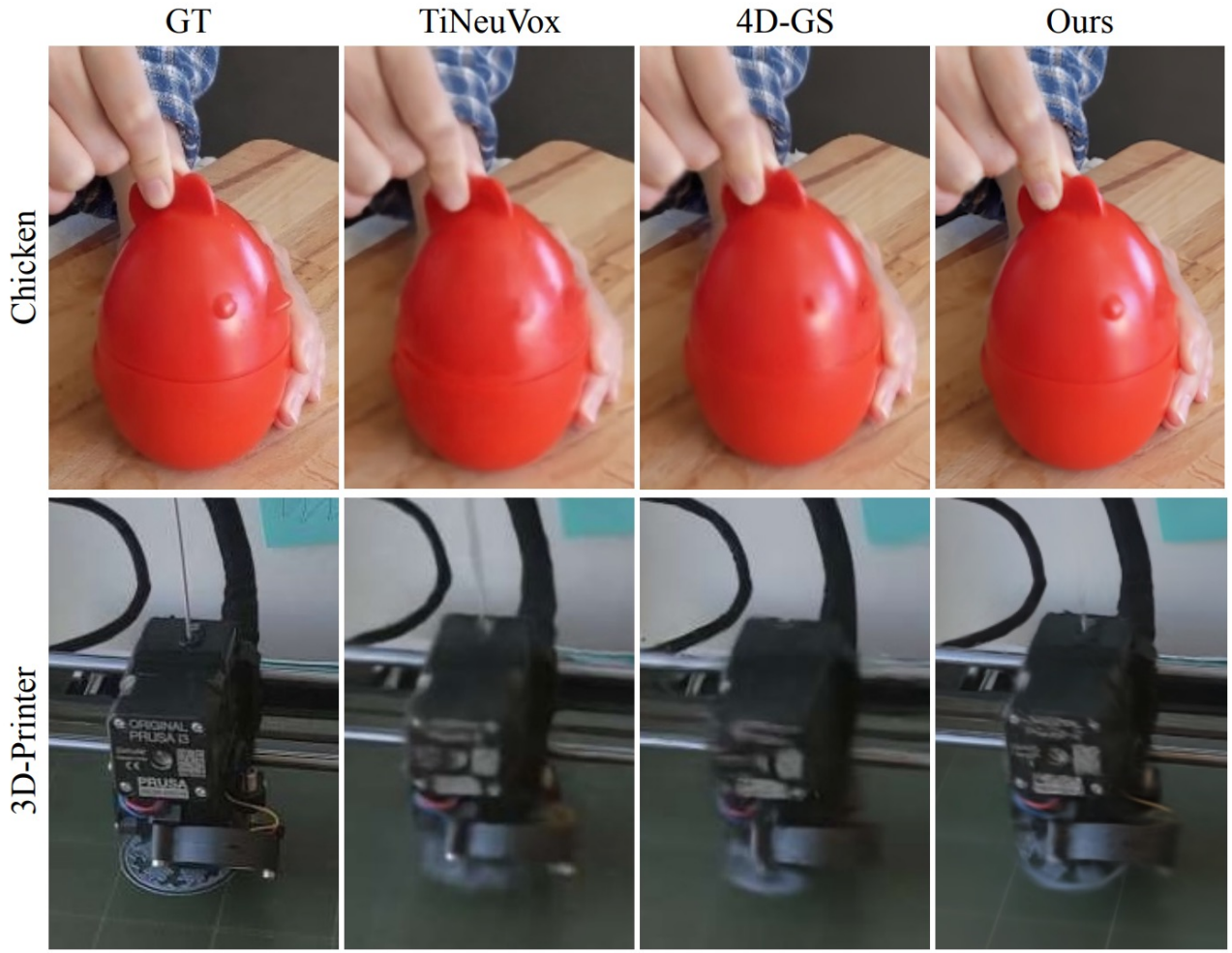


Table 4. Quantitative comparison on Plenoptic Video dataset [18]. Our training speed is 5× faster of magnitude faster than previous leading approaches, . Also, we achieved the highest PSNR score among all methods.

Method	Train Time ↓	PSNR ↑	SSIM ↑
LLFF [24]	-	23.2	-
DyNeRF [18]	1344 hours	29.6	0.96
NeRFPlayer [32]	5.5 hours	30.7	-
K-Planes [8]	1.8 hours	31.6	0.96
4D-GS [35]	2 hours	31.0	0.94
Ours (30K)	22.5 min	30.5	0.97
Ours (60K)	41.8 min	32.0	0.97

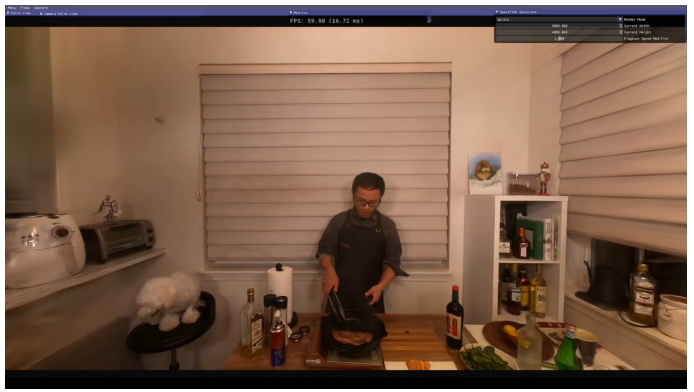
Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. Lin et al. CVPR 2024.

Deformation Fields | Gaussians | Spacetime Gaussian

Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis

Zhan Li[1,2*] Zhang Chen[1†] Zhong Li[1†] Yi Xu[1]
[1] OPPO US Research Center [2] Portland State University
lizhan@pdx.edu zhang.chen@oppo.com zhong.li@oppo.com yi.xu@oppo.com



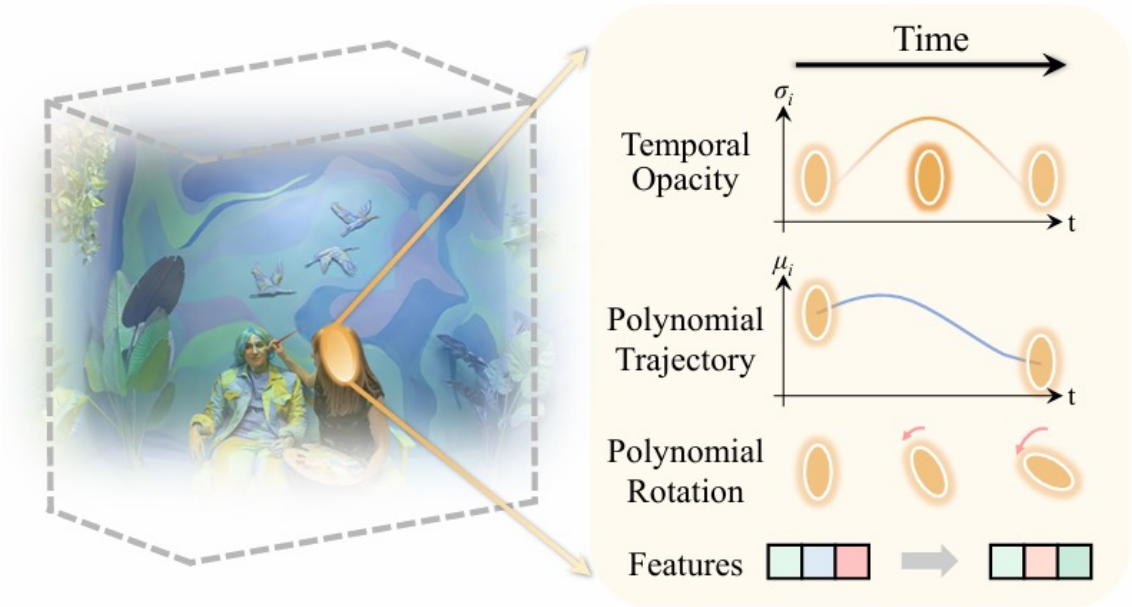


- Input: multi-view video
- Output: per-frame gaussian

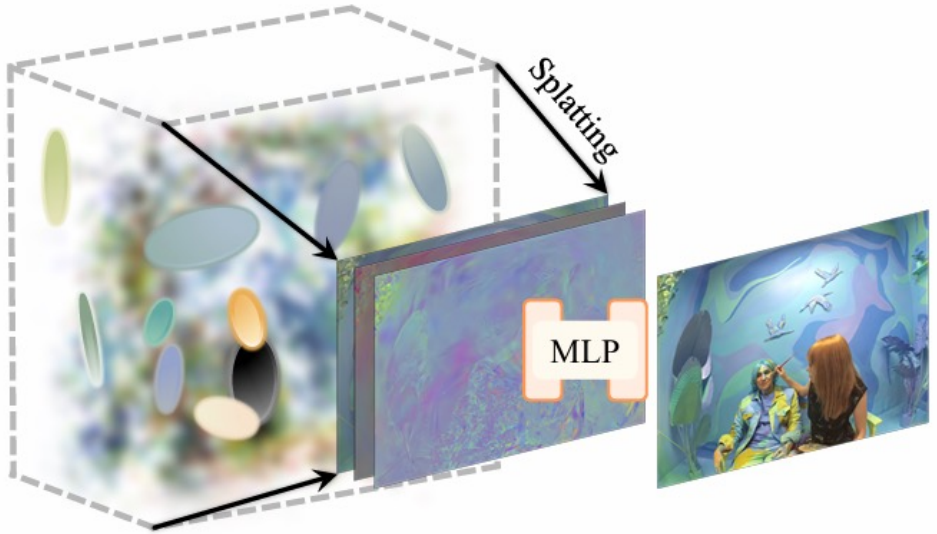
Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. Li et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | Spacetime Gaussian



(a) Spacetime Gaussians



(b) Feature Splatting and Rendering

- Motion trajectory modeling
- Feature map rendering
- RGB image decoding

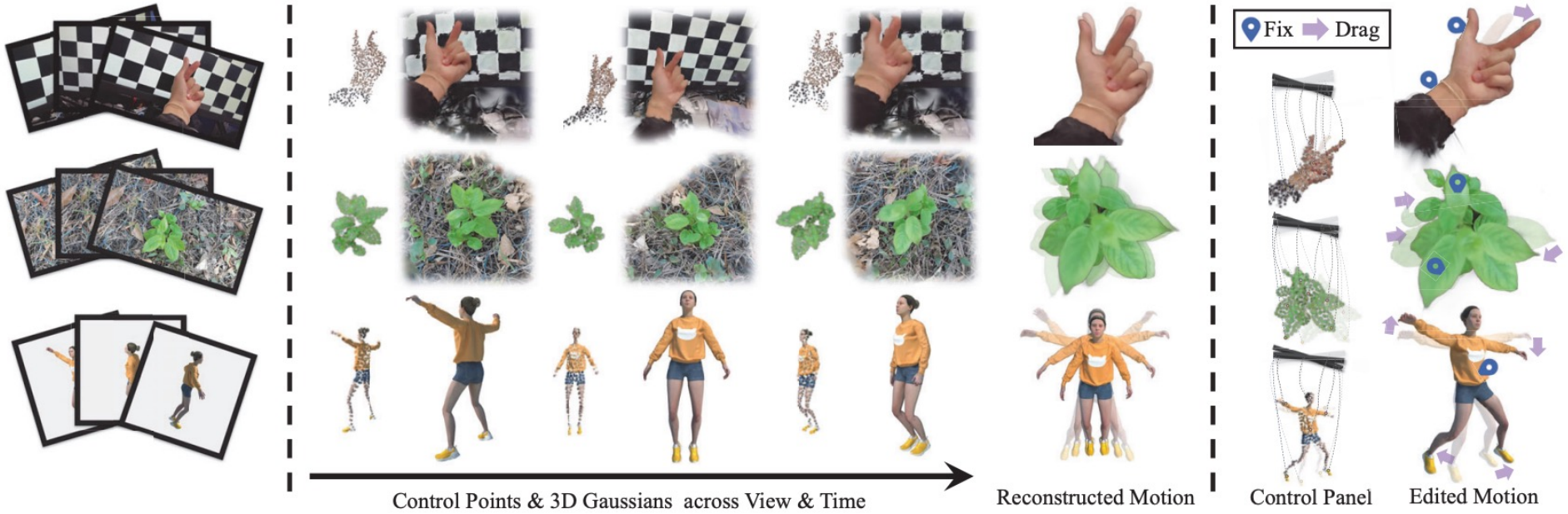
Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. Li et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | SC-GS

SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes

Yi-Hua Huang[1*] **Yang-Tian Sun**[1*] **Ziyi Yang**[3*] **Xiaoyang Lyu**[1] **Yan-Pei Cao**[2†] **Xiaojuan Qi**[1†]
[1] The University of Hong Kong [2] VAST [3] Zhejiang University

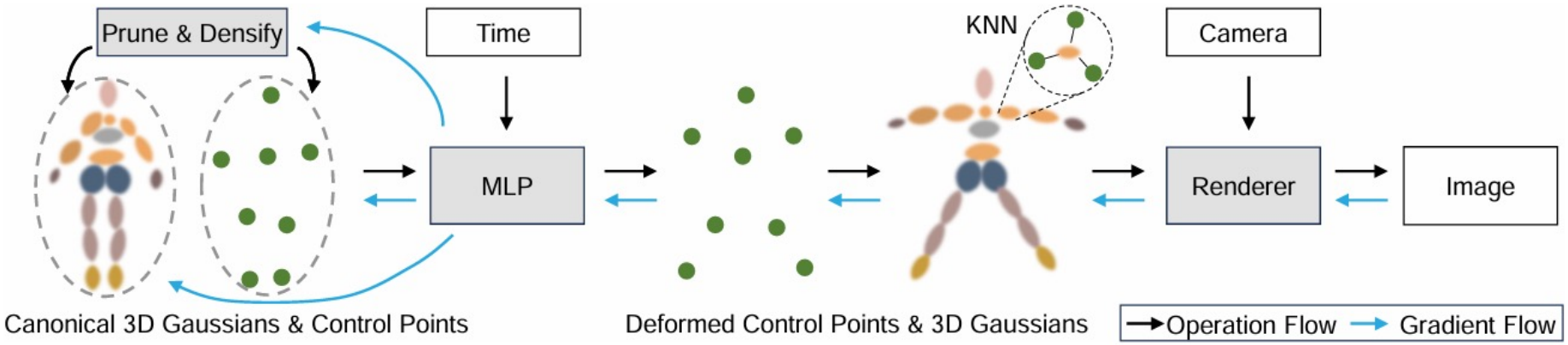


- Input: monocular video
- Output: dynamic gaussian

SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. Huang et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Deformation Fields | Gaussians | SC-GS



- Sparse control points for modeling scene motion
- Rigid constraints during optimization
- Allow motion editing benefiting the learned control points

Deformation Field:

$$\Psi : (p_i, t) \rightarrow (R_i^t, T_i^t). \qquad \mu_j^t = \sum_{k \in \mathcal{N}_j} w_{jk} \left(R_k^t (\mu_j - p_k) + p_k + T_k^t \right),$$

$$q_j^t = \left(\sum_{k \in \mathcal{N}_j} w_{jk} r_k^t \right) \otimes q_j,$$

SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. Huang et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

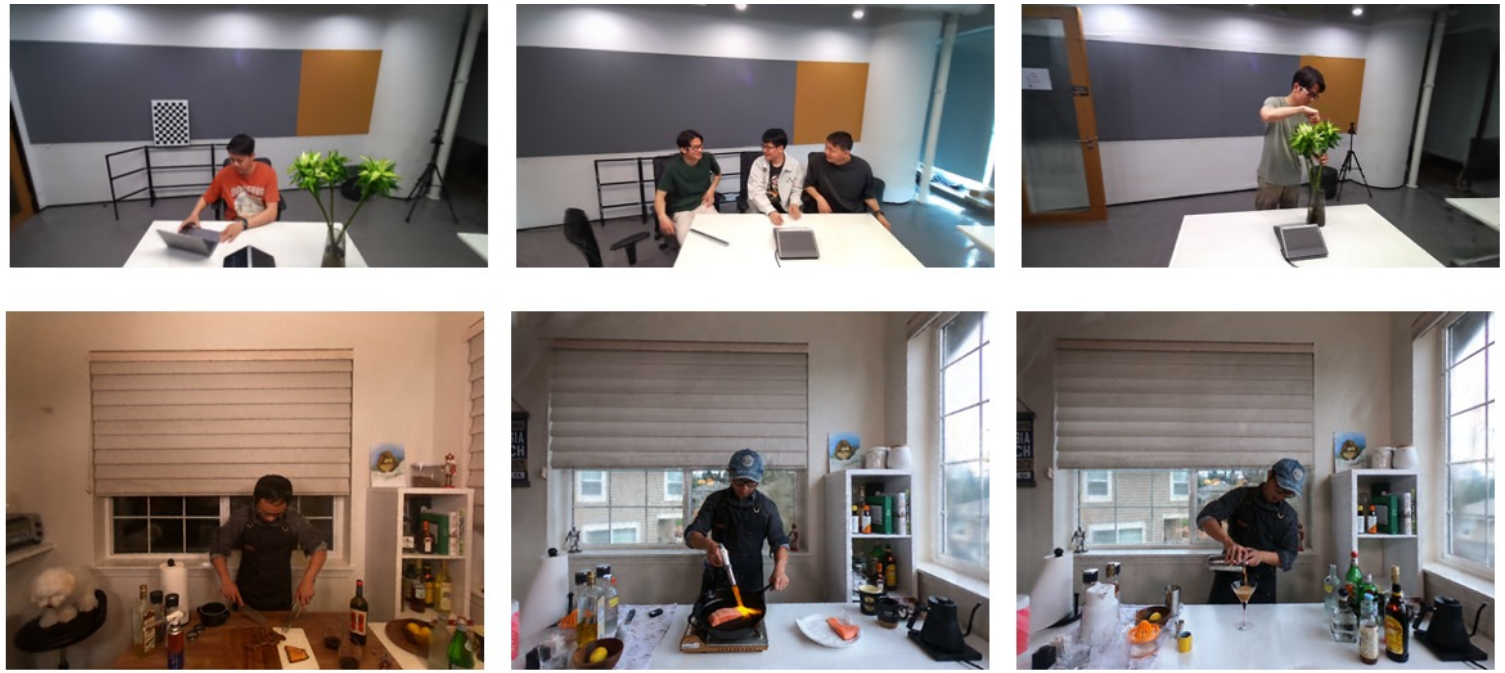
Per-frame Reconstruction

- **NeRF Related**: Streaming NeRF, 4K4D
- **3DGS Related**: Dynamic 3DGS, 3DGStream
- **Feed Forward**: ENeRF, GPS-Gaussian

Streaming Radiance Fields for 3D Video Synthesis

Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, Ping Tan

Alibaba Group



- Input: multi-view video
- Output: compressed per-frame Voxels

Streaming Radiance Fields for 3D Video Synthesis. Li et al. NeurIPS 2022.

Table 1: **Comparison with related works on the N3DV and Meet Room datasets.** We report *per frame* PSNR, training and inference time and storage cost in average for a clear comparison. All the results are recorded with our 3090 GPU except the results of N3DV are referred to the numbers in the original paper. Compared to all the baselines, our method can achieve remarkable speedup on training time with competitive performance on the other metrics.

Methods	N3DV				Meet Room			
	PSNR (dB)	Train (minutes)	Inference (seconds)	Storage (MB)	PSNR (dB)	Train (minutes)	Inference (seconds)	Storage (MB)
JaxNeRF*[13]	28.53	485	67	14	27.11	473	40	14
LLFF*[18]	23.23	8	**0.004**	3192	22.88	3	**0.003**	1500
N3DV [10]	**29.58**	260	>67	**0.1**	-	-	-	-
Plenoxels*[11]	28.68	23	0.12	4106	**27.15**	14	0.1	1015
Ours	28.26	**0.25**	0.12	17.7/31.4	26.72	**0.17**	0.1	**5.7/9.0**

* Denote training from scratch per frame.

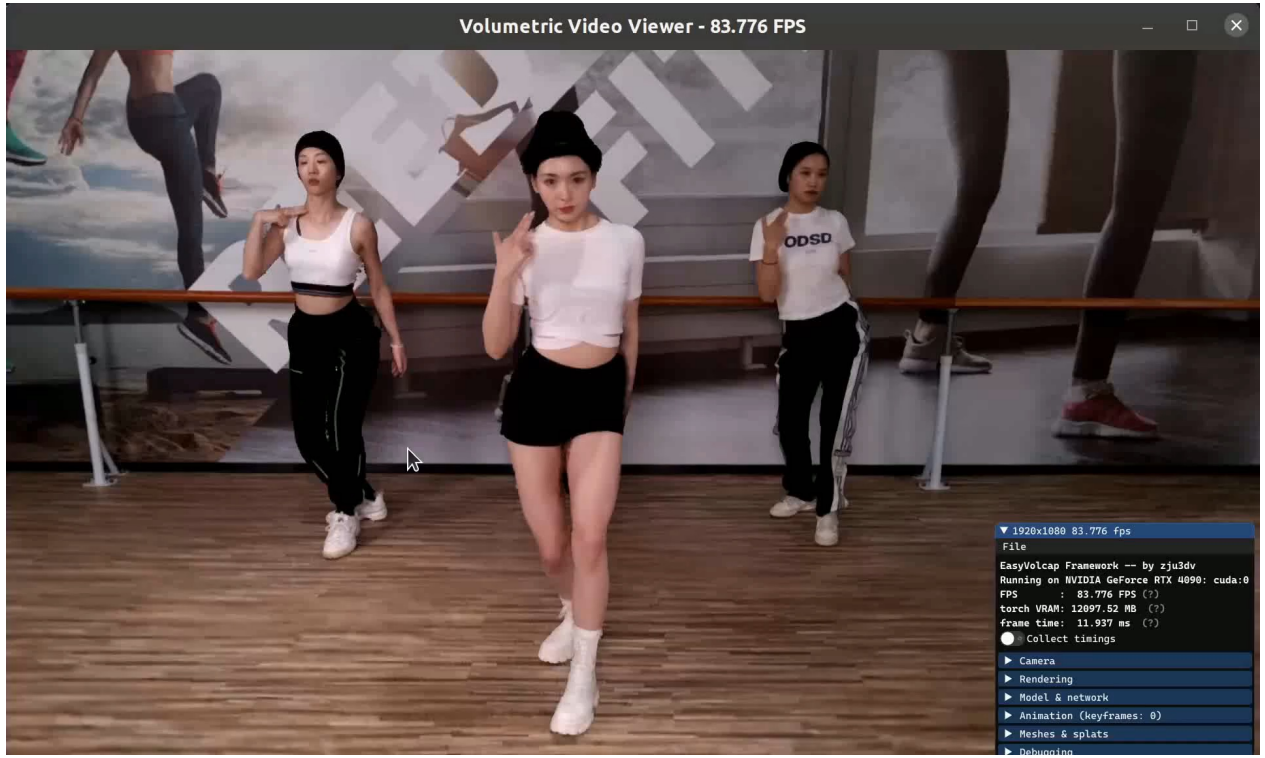
Streaming Voxel Reconstruction: $\mathbf{V}^i = \mathbf{V}^{i-1} + \delta_i = \mathbf{V}^0 + \sum_{j=1}^{i} \delta_j .$

Streaming Radiance Fields for 3D Video Synthesis. Li et al. NeurIPS 2022.

Per-frame | Points | 4K4D

4K4D: Real-Time 4D View Synthesis at 4K Resolution

Zhen Xu[1] Sida Peng[1] Haotong Lin[1] Guangzhao He[1]
Jiaming Sun[2] Yujun Shen[3] Hujun Bao[1] Xiaowei Zhou[1]

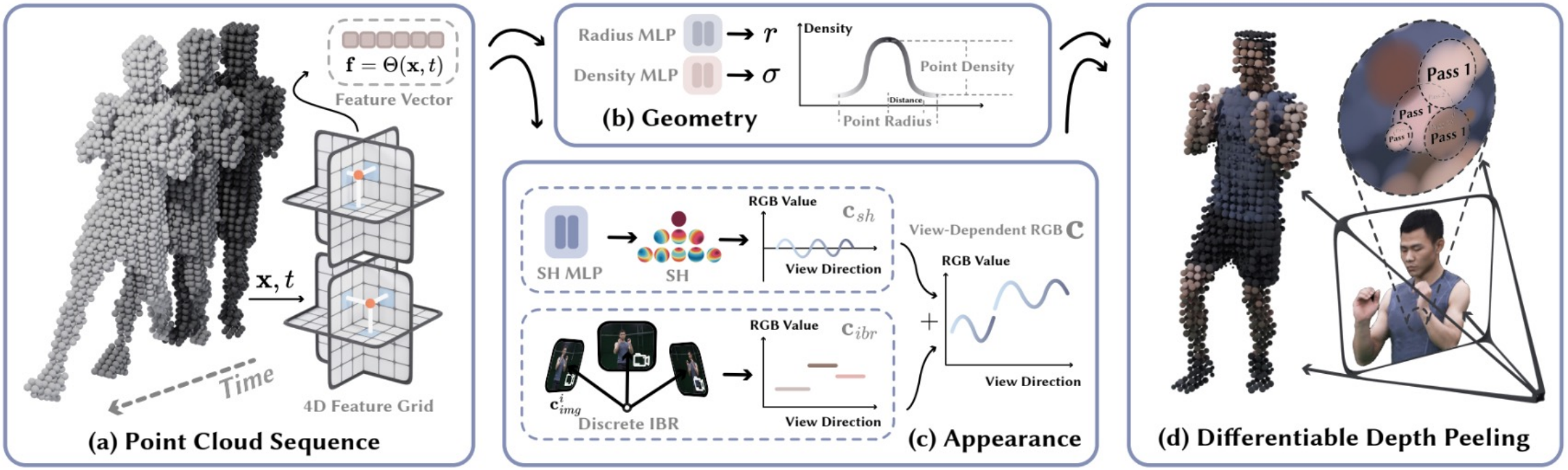


- Input: multi-view video
- Output: per-frame point cloud representation

4K4D: Real-Time 4D View Synthesis at 4K Resolution. Xu et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Per-frame | Points | 4K4D



- Point cloud from space carving
- Geometry and appearance info decoded from feature grid
- Differentiable depth peeling

4K4D: Real-Time 4D View Synthesis at 4K Resolution. Xu et al. CVPR 2024.

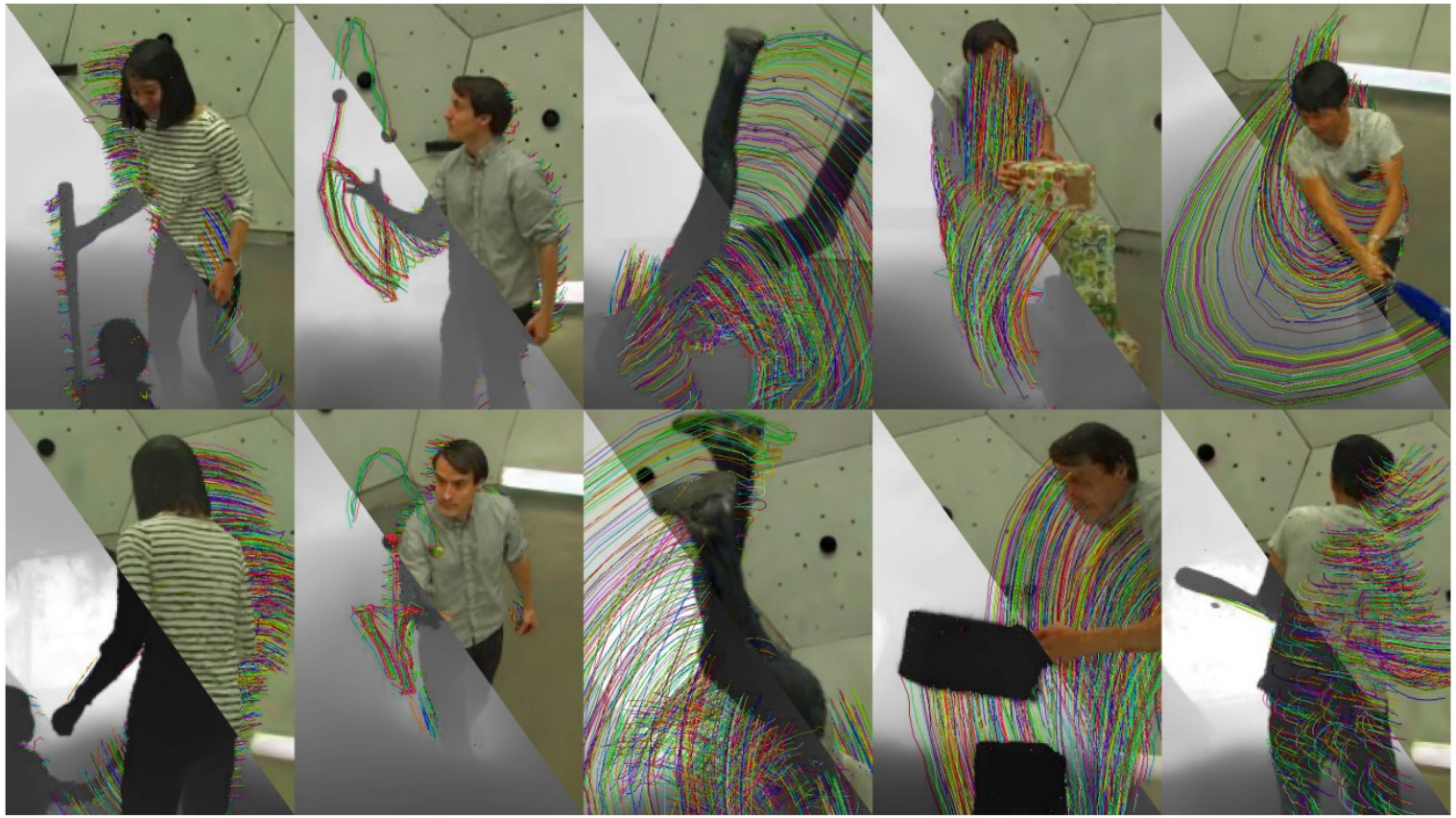
南京大学 NANJING UNIVERSITY

Per-frame | Gaussians | Dynamic 3DGS

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 & Universite Cote d'Azur, France



Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. Luiten et al. 3DV 2024.

南京大学 NANJING UNIVERSITY

Per-frame | Gaussians | Dynamic 3DGS

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

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.

- Multi-view video input
- Per-frame optimization based on previous frame
- Local rigidity loss

Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. Luiten et al. 3DV 2024.

南京大学 NANJING UNIVERSITY

Per-frame | Gaussians | 3DGStream

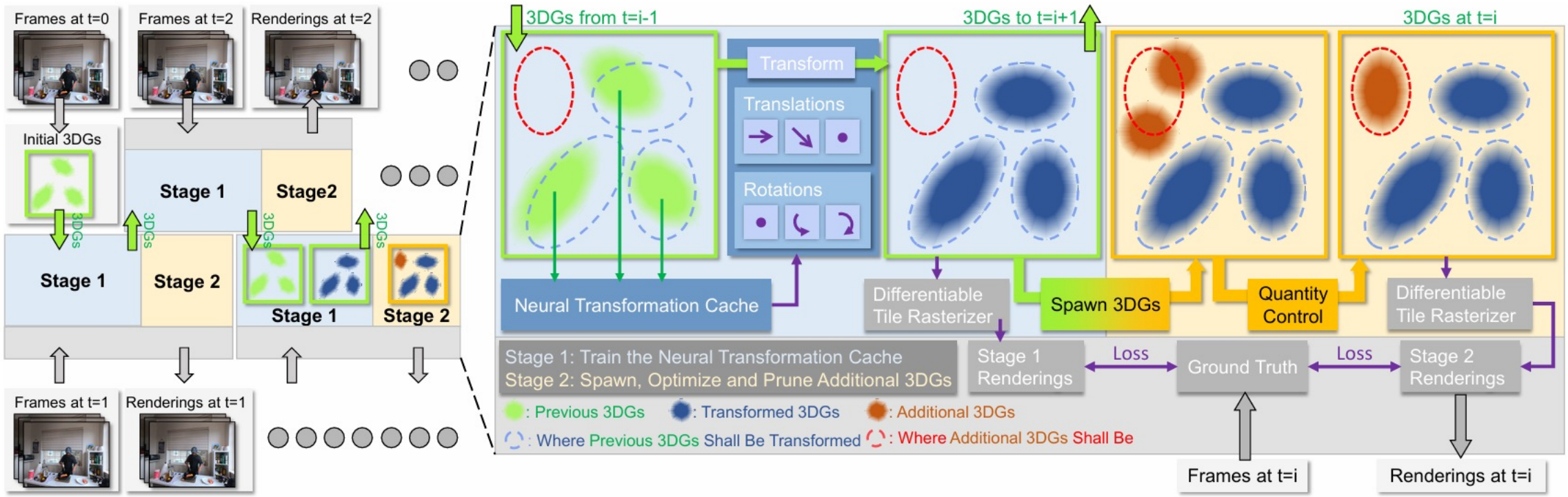
3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos

Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao*, Wei Xing*
Zhejiang University



- Input: multi-view video
- Output: per-frame gaussian

Per-frame | Gaussians | 3DGStream



- Storing per-frame deformation rather than per-frame 3DGS
- Spawn, optimize, and prune additional 3DGS in the new frame

3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos. Sun et al. CVPR 2024

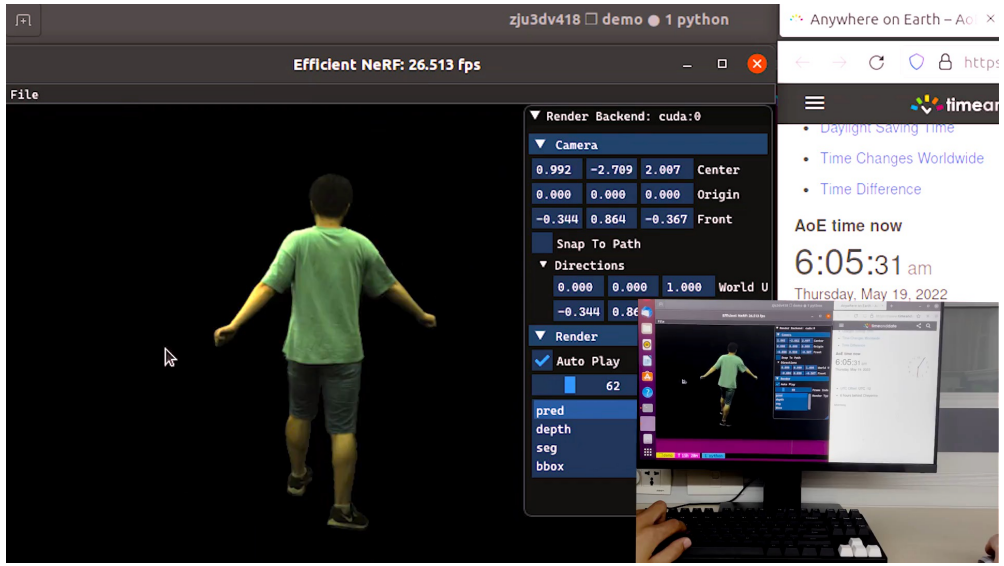
Per-frame | Feed Forward | ENeRF

Efficient Neural Radiance Fields for Interactive Free-viewpoint Video

Haotong Lin*
Sida Peng*
haotongl@zju.edu.cn
pengsida@zju.edu.cn
State Key Laboratory of CAD&CG,
Zhejiang University
China

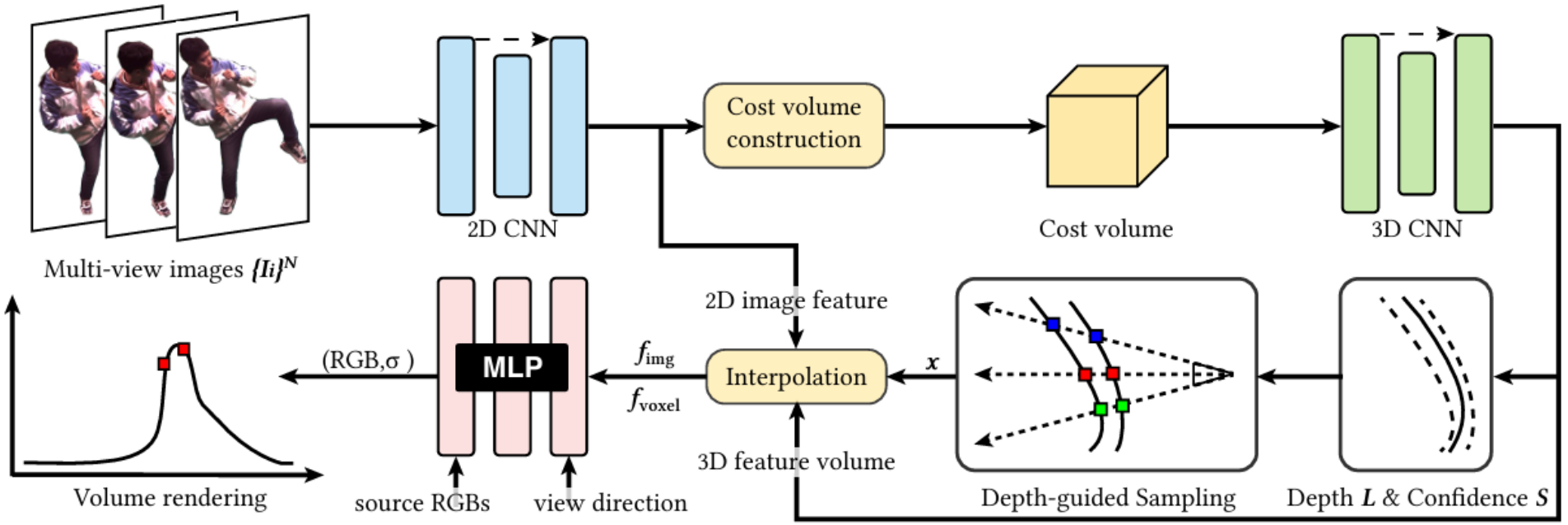
Zhen Xu
Yunzhi Yan
Qing Shuai
zhenx@zju.edu.cn
yanyz@zju.edu.cn
s_q@zju.edu.cn
Zhejiang University, China

Hujun Bao
Xiaowei Zhou†
bao@cad.zju.edu.cn
xwzhou@zju.edu.cn
State Key Laboratory of CAD&CG,
Zhejiang University
China



南京大学 NANJING UNIVERSITY

Per-frame | Feed Forward | ENeRF



- Per-frame Generalized NeRF (e,g., MVSNet + NeRF)
- Fast and efficient sampling via MVS cost volume

Efficient Neural Radiance Fields for Interactive Free-viewpoint Video. Lin et al. SIGGRAPH 2022

南京大学 NANJING UNIVERSITY

Per-frame | Feed Forward | GPS-Gaussian

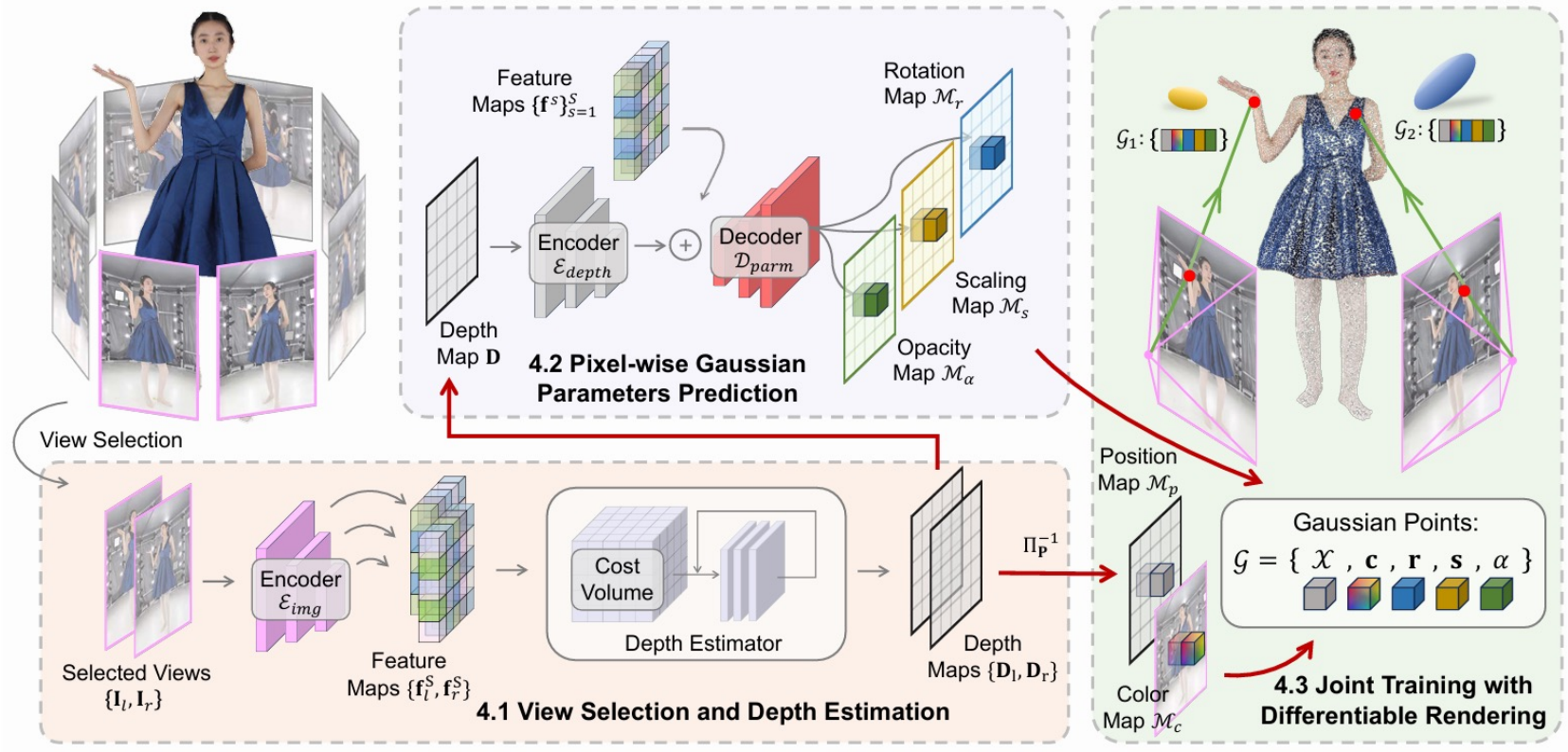
GPS-Gaussian: Generalizable Pixel-wise 3D Gaussian Splatting for Real-time Human Novel View Synthesis

Shunyuan Zheng[†,1], Boyao Zhou[2], Ruizhi Shao[2], Boning Liu[2], Shengping Zhang[*,1,3],
Liqiang Nie[1], Yebin Liu[2]
[1]Harbin Institute of Technology [2]Tsinghua University [3]Peng Cheng Laboratory

- Input: multi-view video
- Output: per-frame gaussian

GPS-Gaussian: Generalizable Pixel-wise 3D Gaussian Splatting for Real-time Human Novel View Synthesis. Wang et al. CVPR 2024.

Per-frame | Feed Forward | GPS-Gaussian



- Per-frame Generalized 3DGS
- Learned MVS and features as 3DGS initialization
- Real-time 4D reconstruction

GPS-Gaussian: Generalizable Pixel-wise 3D Gaussian Splatting for Real-time Human Novel View Synthesis. Wang et al. CVPR 2024.

Model Prior

- **Depth and Flow**: Robust NeRF, DynIBaR, MoDGS, Shape of Motion
- **Dynamic Physics**: NVFi, PAC-NeRF, DeformGS
- **Parametric Human**: Animatable NeRF, GaussianAvatars, Animatable Gaussians
- **(TBD) Large Vision Model**: Cat3D/ReconX, Consistent4D/STAG4D/SV4D

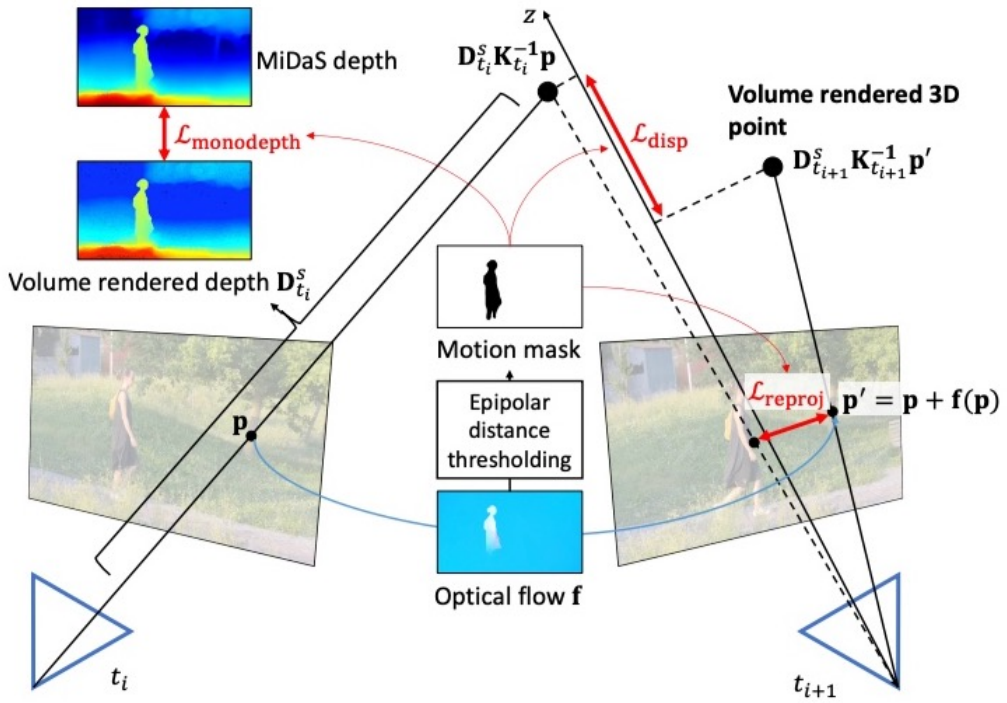
Priors | Depth & Flow | RoDynRF

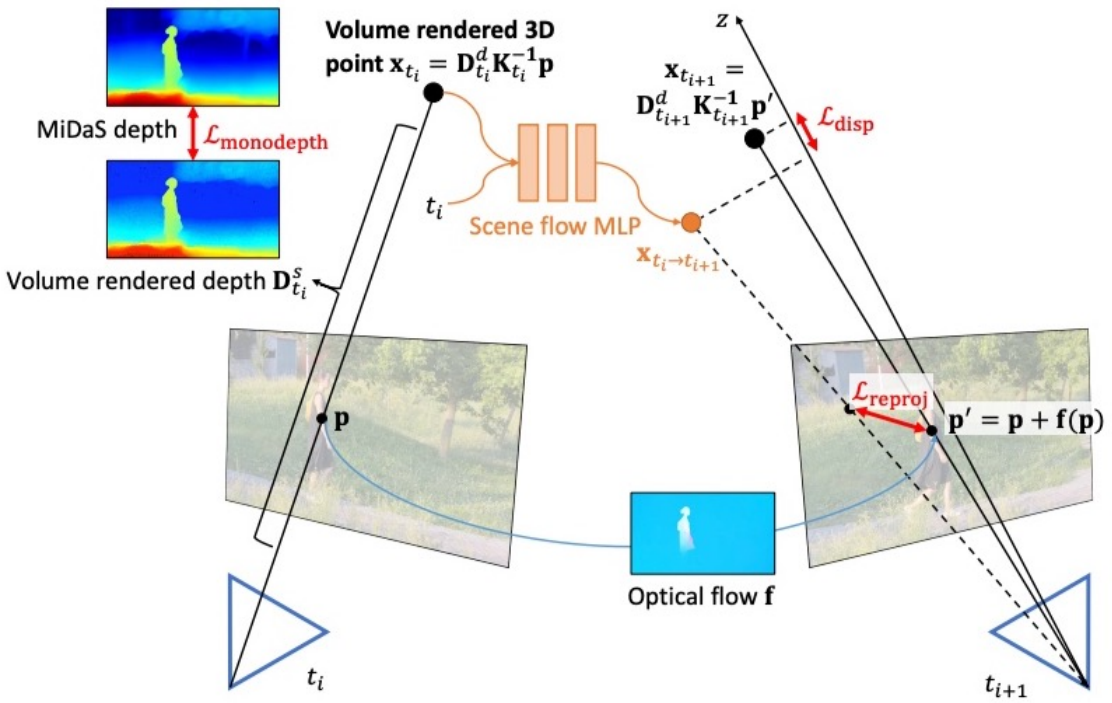
Robust Dynamic Radiance Fields

Yu-Lun Liu[2*] Chen Gao[1] Andreas Meuleman[3*] Hung-Yu Tseng[1] Ayush Saraf[1]
Changil Kim[1] Yung-Yu Chuang[2] Johannes Kopf[1] Jia-Bin Huang[1,4]
[1]Meta [2]National Taiwan University [3]KAIST [4]University of Maryland, College Park
https://robust-dynrf.github.io/

RoDynRF: Robust Dynamic Radiance Fields. Liu et al. CVPR 2023.

Priors | Depth & Flow | RoDynRF





- MiDaS for Depth Supervision
- Guide the scene flow of reproject volume render point with RAFT

RoDynRF: Robust Dynamic Radiance Fields. Liu et al. CVPR 2023.

南京大学 NANJING UNIVERSITY

Priors | Depth & Flow | DynIBaR

DynIBaR: Neural Dynamic Image-Based Rendering

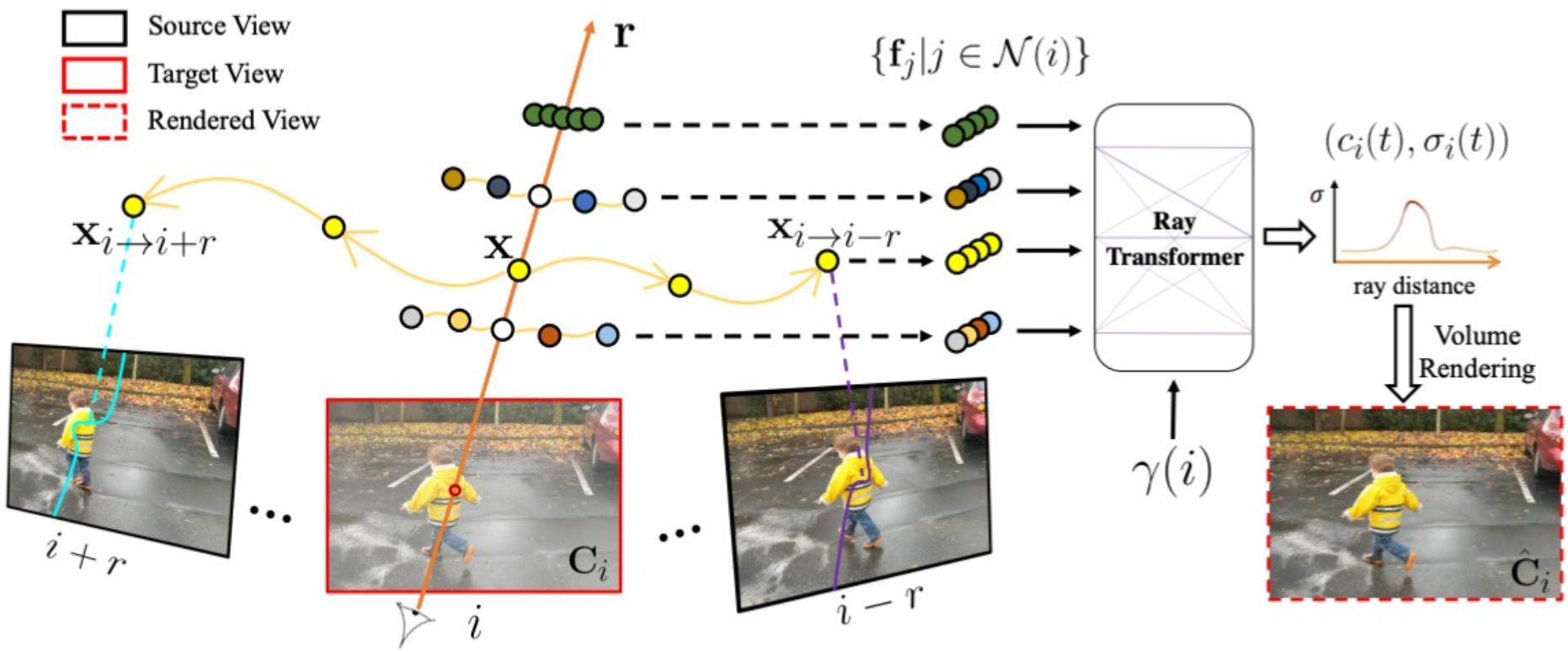
Zhengqi Li[1], Qianqian Wang[1,2], Forrester Cole[1], Richard Tucker[1], Noah Snavely[1]

[1]Google Research [2]Cornell Tech

DynIBaR: Neural Dynamic Image-Based Rendering. Li et al. CVPR 2023.

南京大学 NANJING UNIVERSITY

Priors | Depth & Flow | DynIBaR



- Aggregate multi-view image features for dynamic NeRF rendering
- Leverging optical flow consistency priors from RAFT

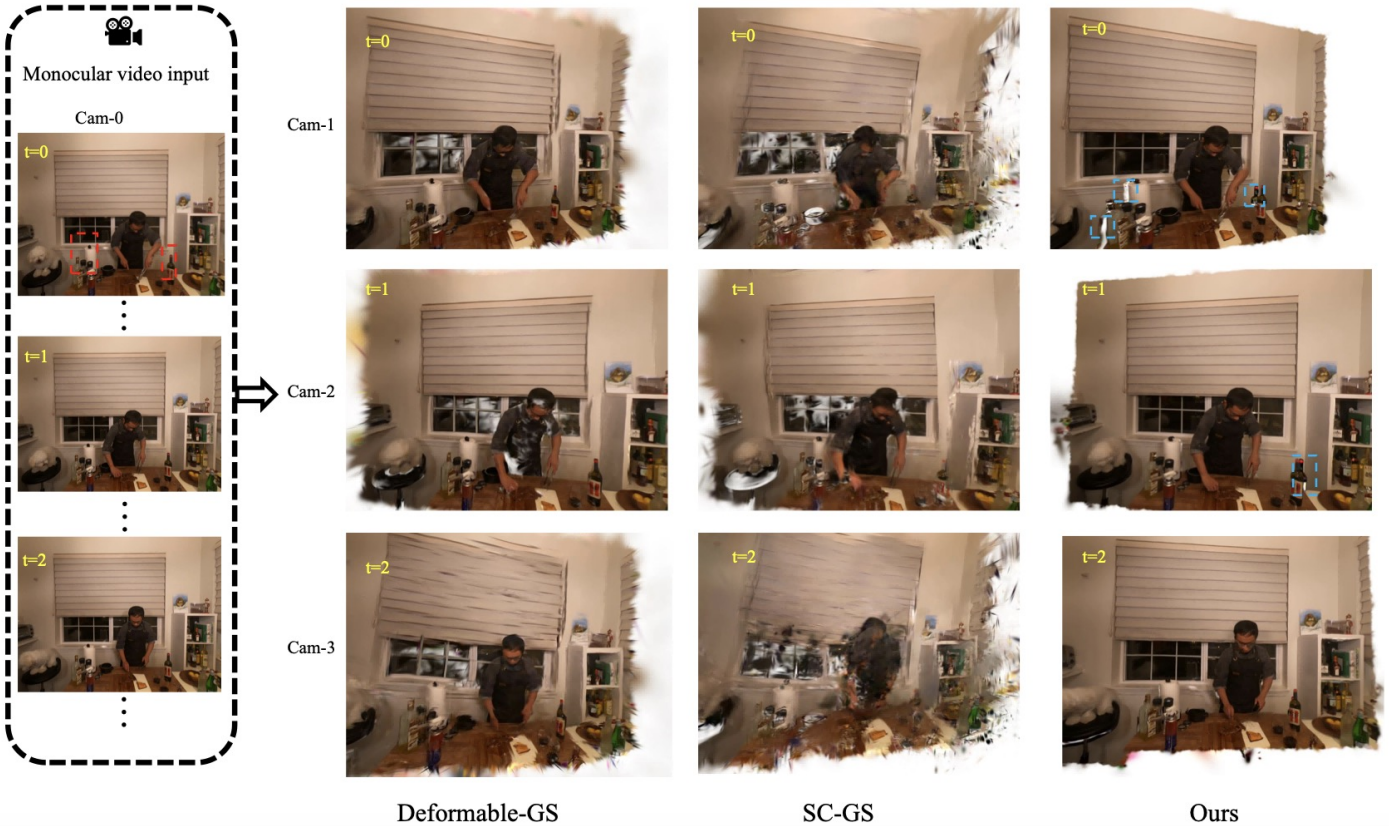
DynIBaR: Neural Dynamic Image-Based Rendering. Li et al. CVPR 2023.

南京大学 NANJING UNIVERSITY

Priors | Depth & Flow | MoDGS

MoDGS: Dynamic Gaussian Splatting from Causually-captured Monocular Videos

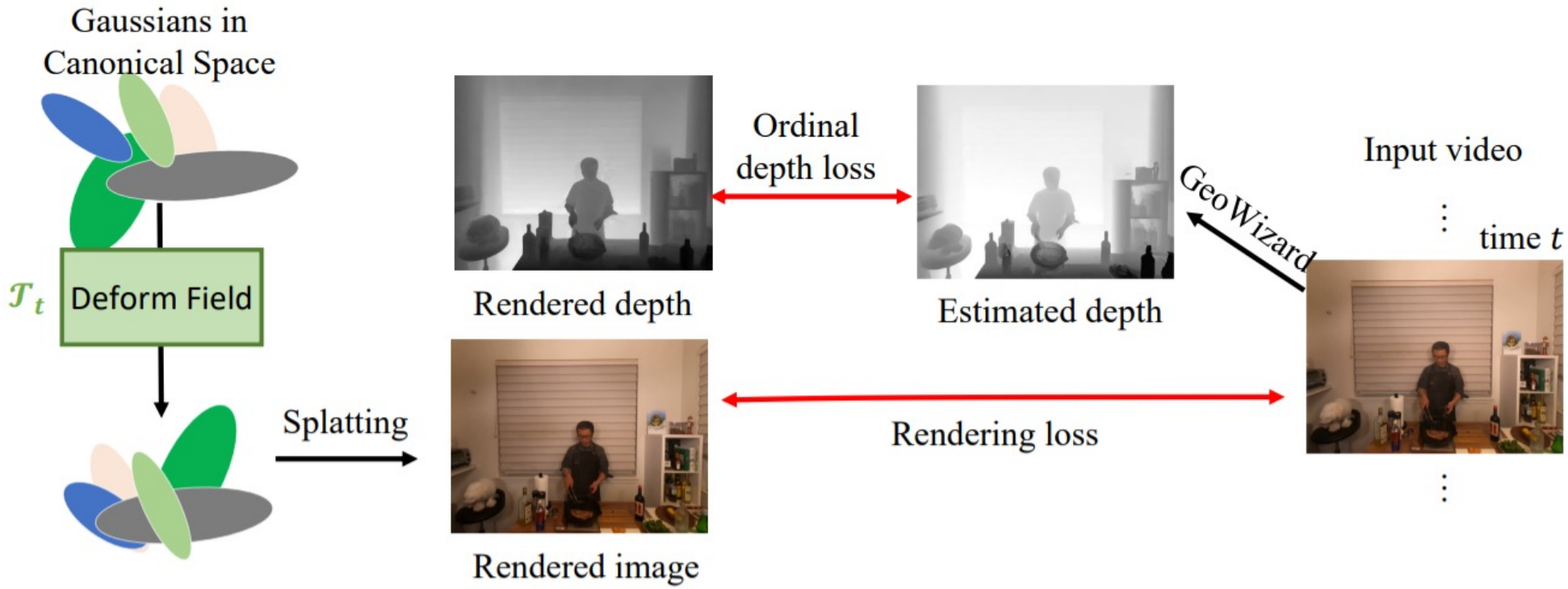
Qingming Liu[1*], Yuan Liu[2*], Jiepeng Wang[2], Xianqiang Lv[1],
Peng Wang[2], Wenping Wang[3], Junhui Hou[1†],



MoDGS: Dynamic Gaussian Splatting from Causually-captured Monocular Videos. Liu et al. ariXv 2024.

南京大学 NANJING UNIVERSITY

Priors | Depth & Flow | MoDGS



- Computing 3D flow by lifting 2D flow (RAFT) to 3D with mono depth (GeoWizard)
- Gaussian (deformation) initialization with 3D flow
- Resolve depth inconsistency between frames

MoDGS: Dynamic Gaussian Splatting from Causually-captured Monocular Videos. Liu et al. ariXv 2024.

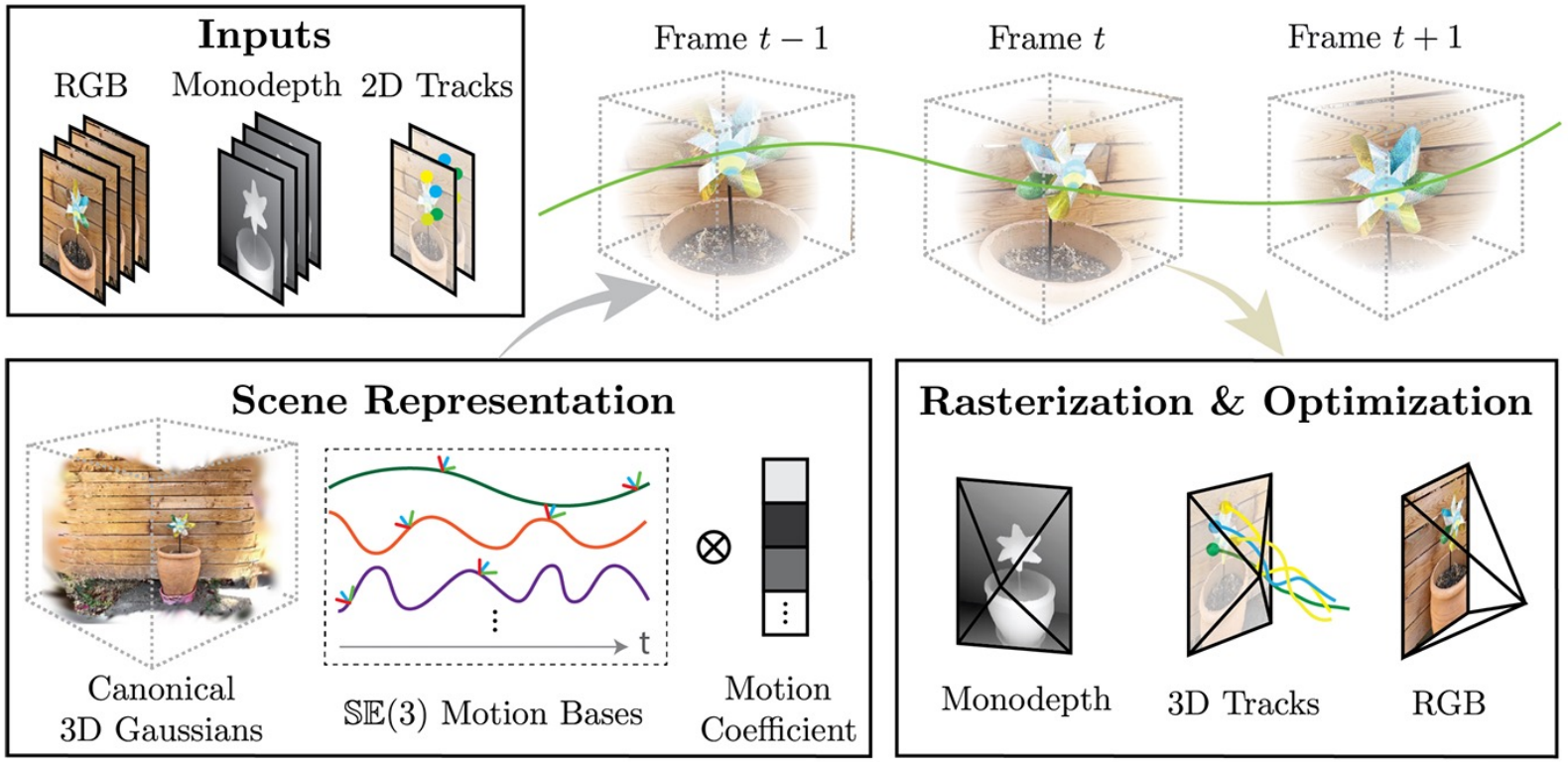
Shape of Motion: 4D Reconstruction from a Single Video

Qianqian Wang[1,2*], Vickie Ye[1*], Hang Gao[1*],
Jake Austin[1], Zhengqi Li[2], and Angjoo Kanazawa[1]

[1] UC Berkeley [2] Google Research

Shape of Motion: 4D Reconstruction from a Single Video. Wang et al. ECCV 2024.

Priors | Depth & Flow | Shape of Motion



- Explicit trajectory motion modeling
- Foreground mask by Track–Anything
- Depth prior by Depth–Anything
- Tracking prior by TAPIR

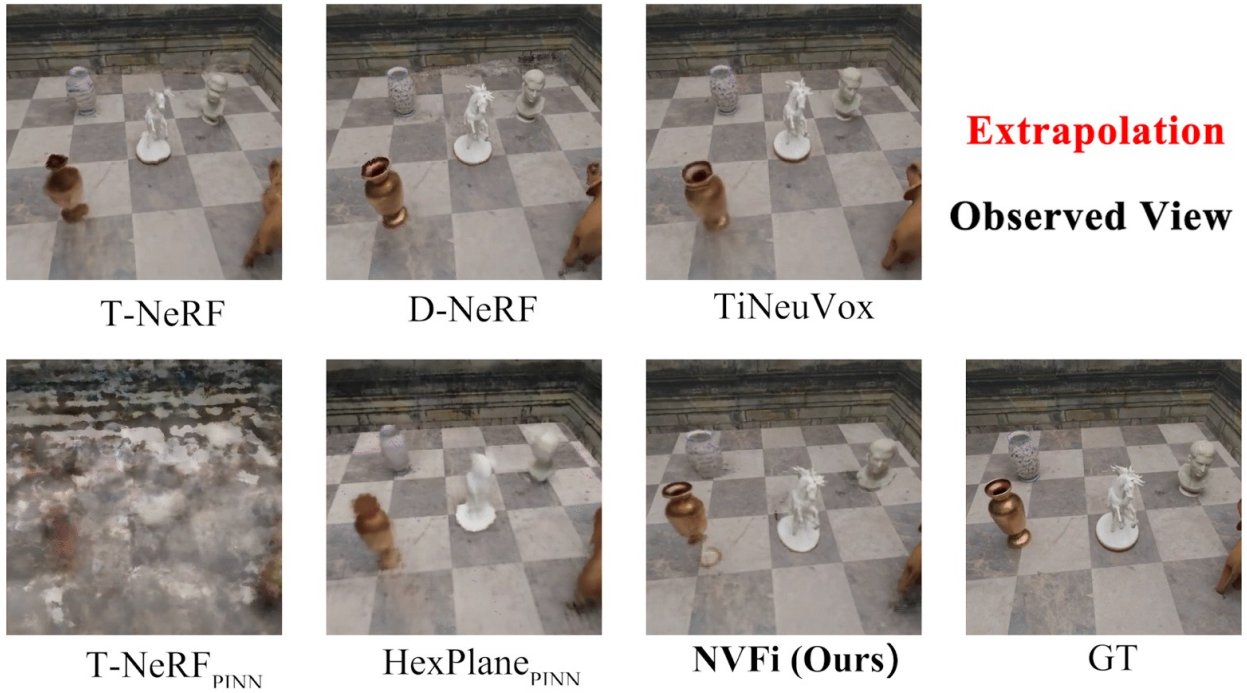
Shape of Motion: 4D Reconstruction from a Single Video. Wang et al. ECCV 2024.

南京大学 NANJING UNIVERSITY

NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos

Jinxi Li Ziyang Song Bo Yang

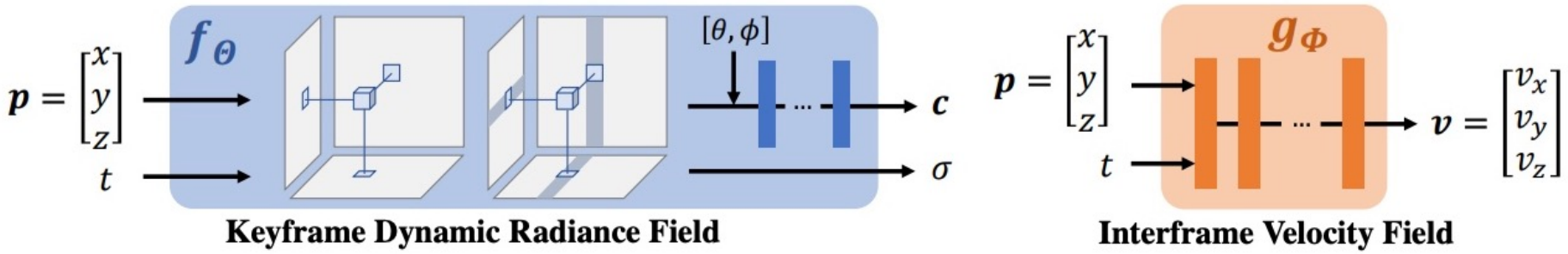
vLAR Group, The Hong Kong Polytechnic University



- Input: multi-view video
- Output: 4D NeRF & Physical Velocity

NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos. Li et al. NeurIPS 2023.

Priors | Physics | NVFi

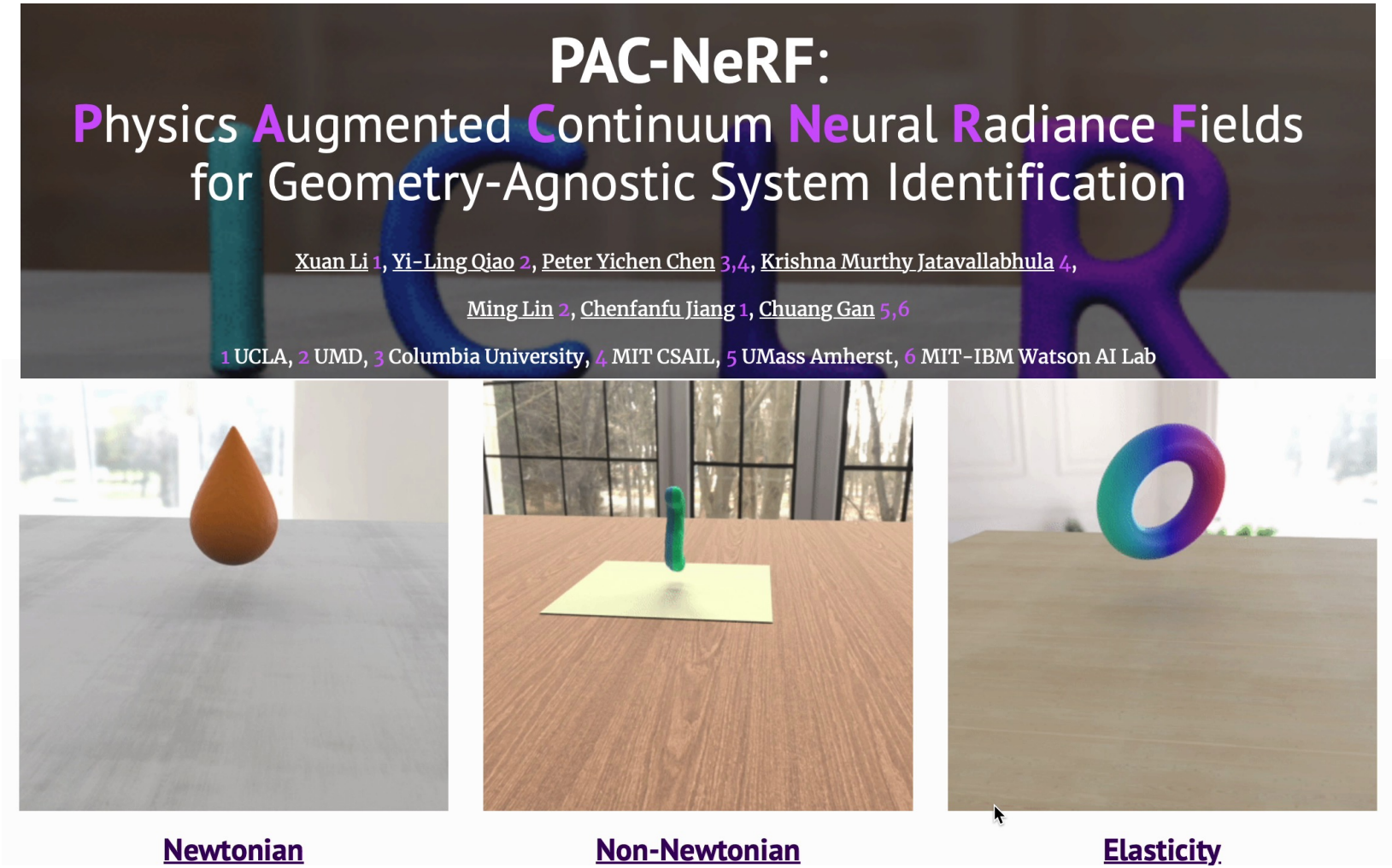


- Physics–informed radiance fields (velocity field modeling)
- Joint optimazation of radiance fileds and velocity field

NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos. Li et al. NeurIPS 2023.

南京大学 NANJING UNIVERSITY

Priors | Physics | PAC-NeRF

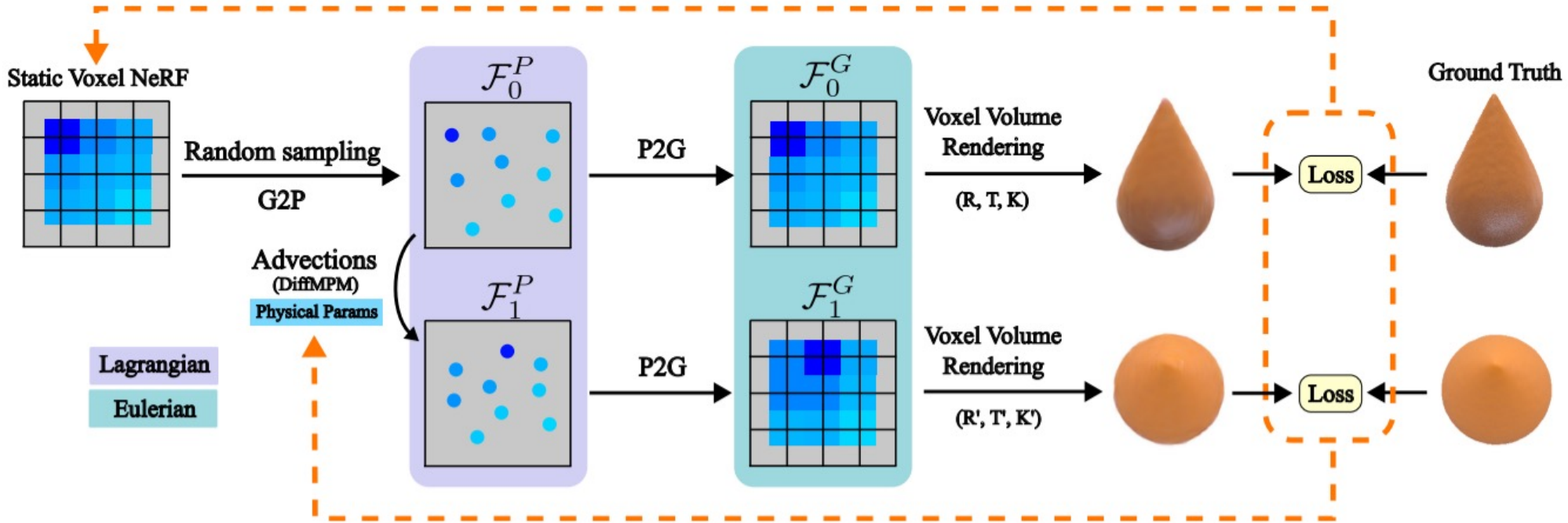


- Input: multi-view video
- Output: static NeRF with physical properties

PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. Li et al. ICLR 2023.

南京大学 NANJING UNIVERSITY

Priors | Physics | PAC-NeRF



- Conservation laws of continuum mechanics
- Optimizing velocity field for continuum materials

PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. Li et al. ICLR 2023.

南京大学 NANJING UNIVERSITY

DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation

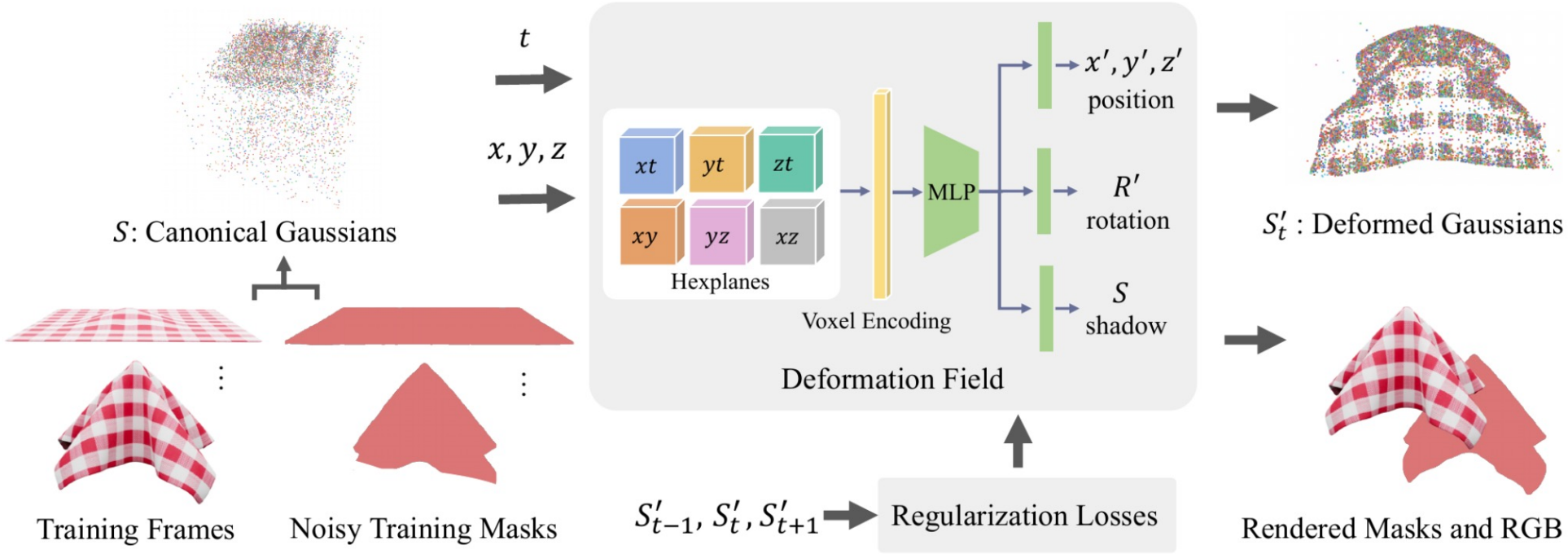
Bardienus P. Duisterhof[1], Mandi Zhao[2], Yunchao Yao[1], Jia-Wei Liu[4], Jenny Seidenschwarz[1,5], Mike Zheng Shou[4], Deva Ramanan[1], Shuran Song[2], Stan Birchfield[3], Bowen Wen[3], and Jeffrey Ichnowski[1]

[1] Carnegie Mellon University, The Robotics Institute {bduister,jeffi}@cmu.edu
[2] Stanford University
[3] NVIDIA
[4] National University of Singapore
[5] Technical University of Munich

- Input: multi–view video
- Output: dynamic gaussian

DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation. Duisterhof et al. WAFR 2024.

Priors | Physics | DeformGS



- Momentum conservation loss for gaussian position
- Local distance loss for adjacent gaussian

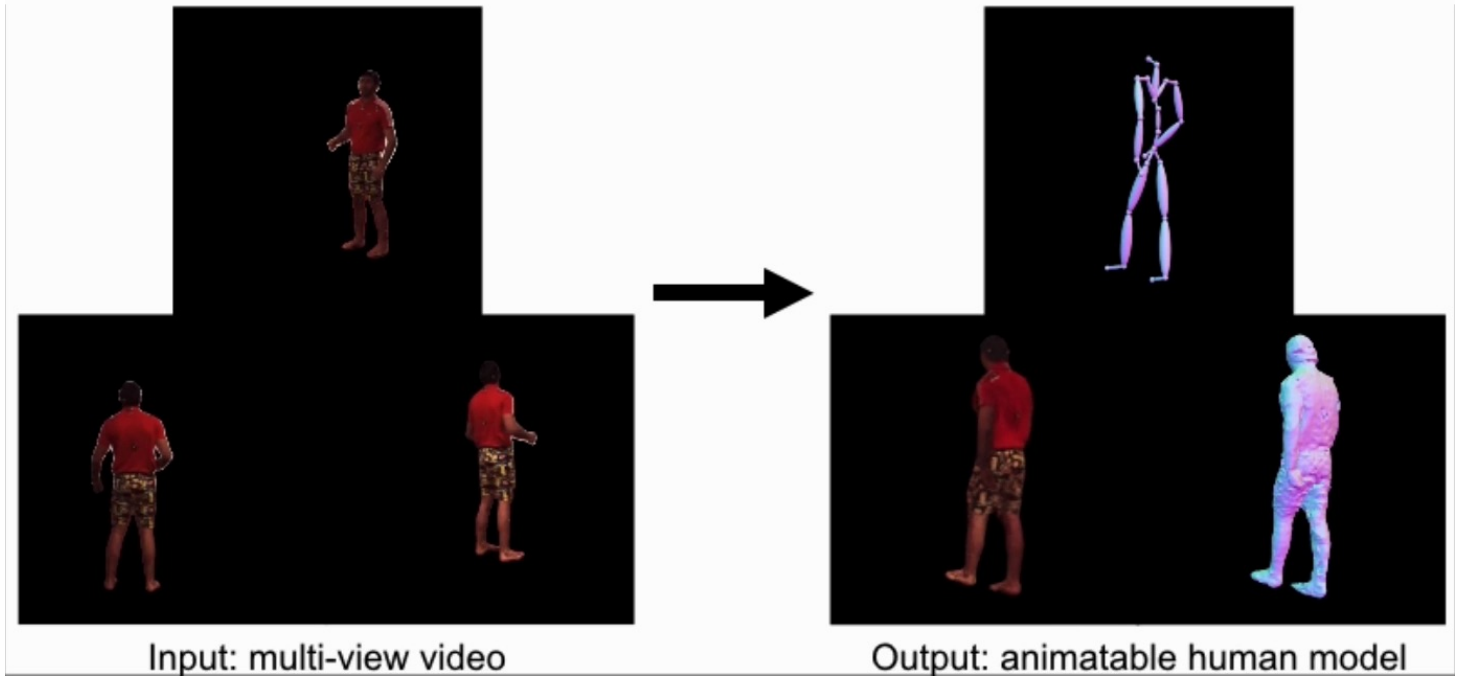
DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation. Duisterhof et al. WAFR 2024.

南京大学 NANJING UNIVERSITY

Priors | Human | Animatable NeRF

Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies

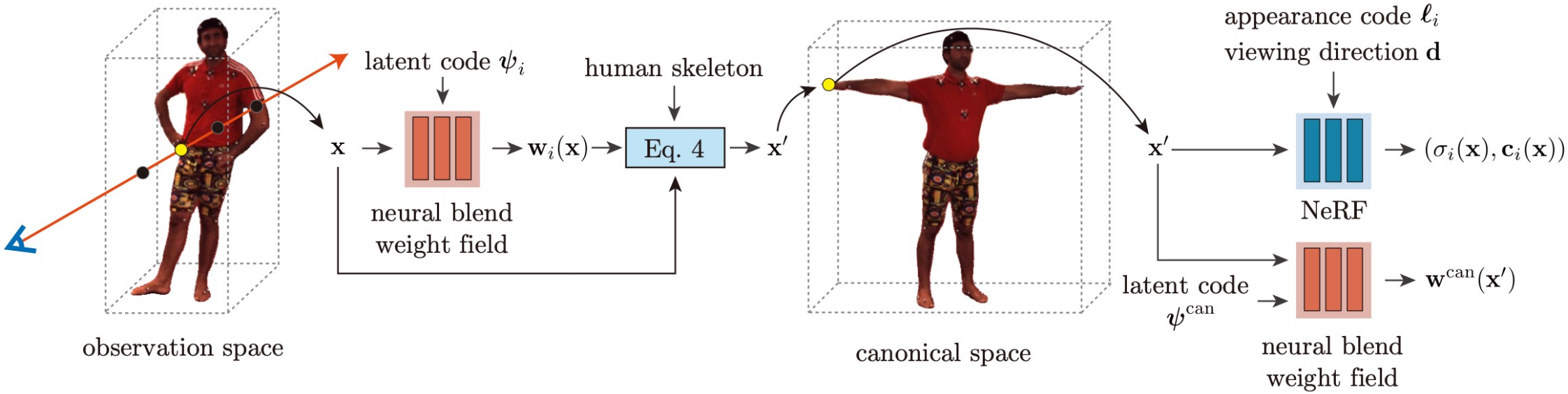
Sida Peng[1*] Junting Dong[1*] Qianqian Wang[2] Shangzhan Zhang[1]
Qing Shuai[1] Xiaowei Zhou[1] Hujun Bao[1†]
[1]Zhejiang University [2]Cornell University



- Input: multi–view human centric video
- Output: animatable human model

Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies. Peng et al. ICCV 2021.

Priors | Human | Animatable NeRF



- Skeleton–driven deformation modeling
- Warping field from Linear Blend Skinning (LBS)
- Animatable radiance field

Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies. Peng et al. ICCV 2021.

南京大学 NANJING UNIVERSITY

Priors | Human | GaussianAvatars

GaussianAvatars:
Photorealistic Head Avatars with Rigged 3D Gaussians

Shenhan Qian[1] Tobias Kirschstein[1] Liam Schoneveld[2] Davide Davoli[3] Simon Giebenhain[1] Matthias Nießner[1]

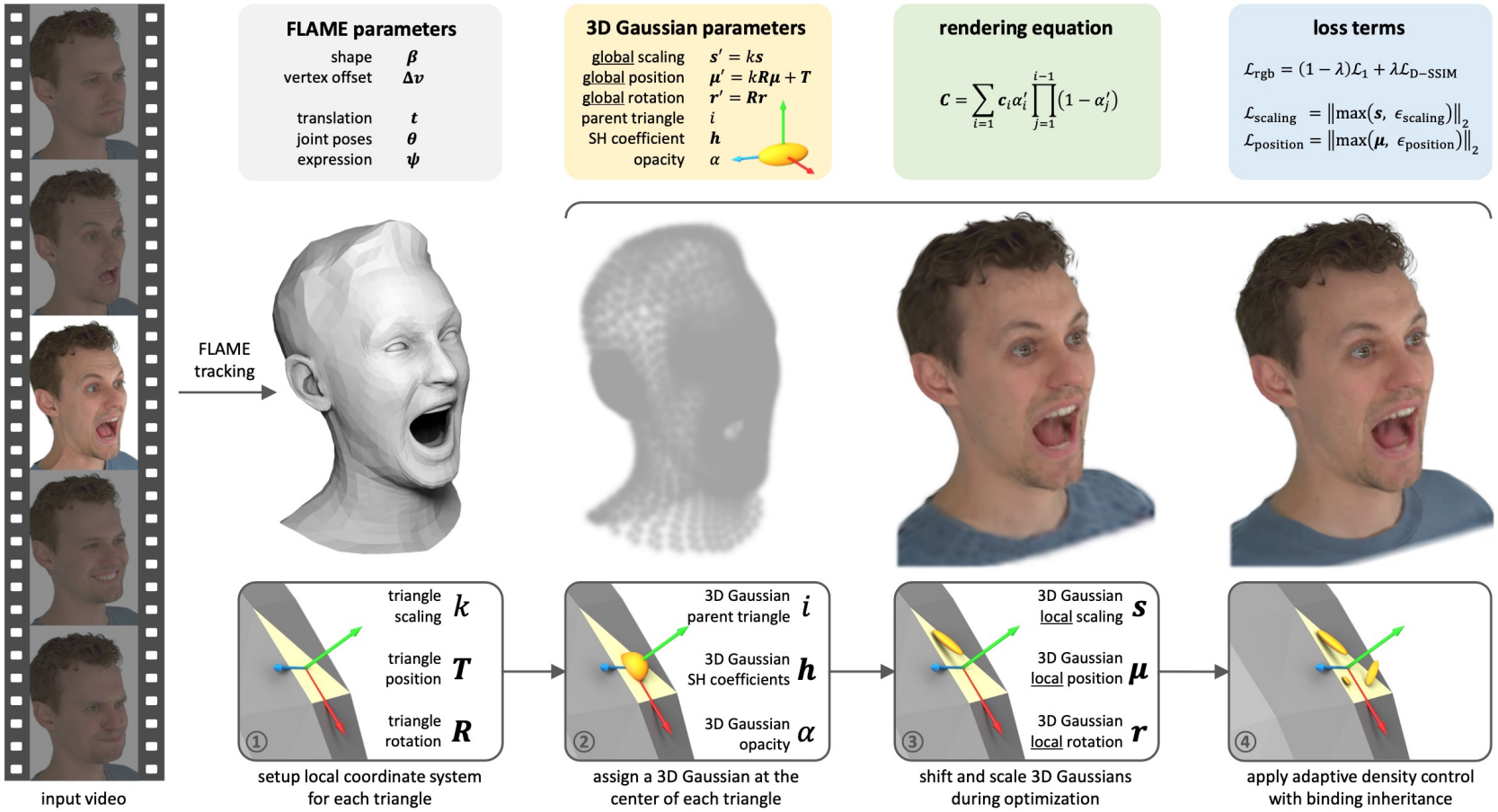
[1]Technical University of Munich [2]Woven by Toyota [3]Toyota Motor Europe NV/SA
associated partner by contracted services

- Input: multi-view human centric video
- Output: dynamic gaussian

GaussianAvatars: Photorealistic Head Avatars with Rigged 3D Gaussians. Qian et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Priors | Human | GaussianAvatars



- Flame parametric modeling
- Binding mesh with 3D gaussian

GaussianAvatars: Photorealistic Head Avatars with Rigged 3D Gaussians. Qian et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Priors | Human | Animatable Gaussians

Animatable Gaussians:
Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling

Zhe Li[1], Zerong Zheng[2], Lizhen Wang[1], Yebin Liu[1]

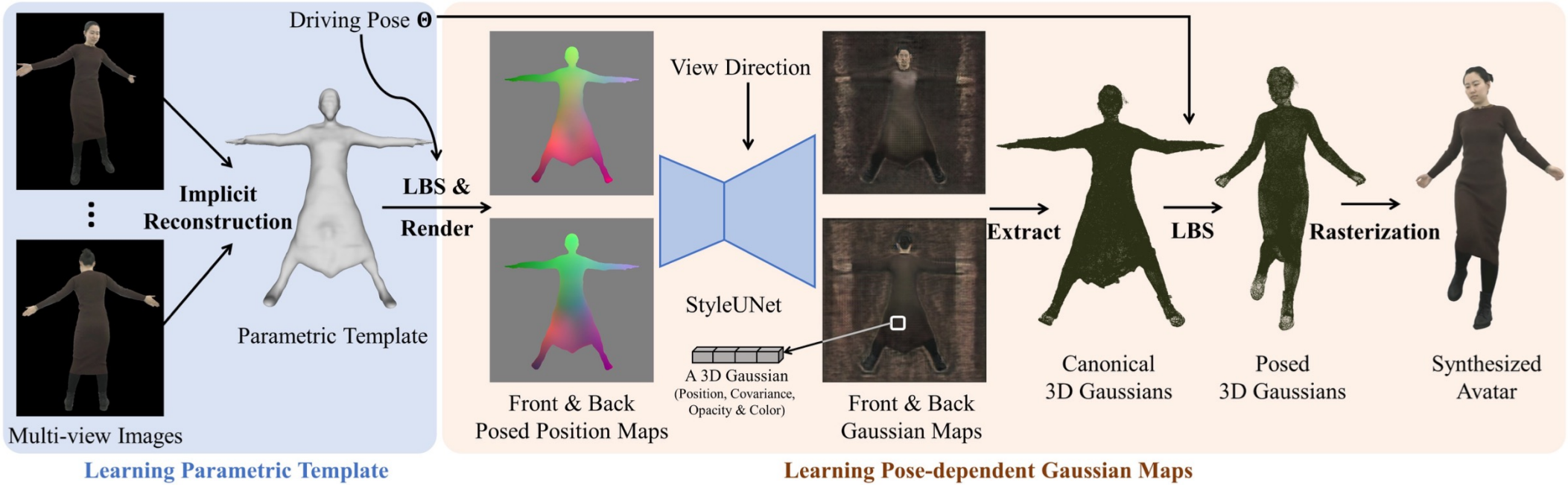
[1]Tsinghua University [2]NNKosmos Technology

- Input: multi-view human centric video
- Output: dynamic gaussian

Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling. Li et al. CVPR 2024.

南京大学 NANJING UNIVERSITY

Priors | Human | Animatable Gaussians

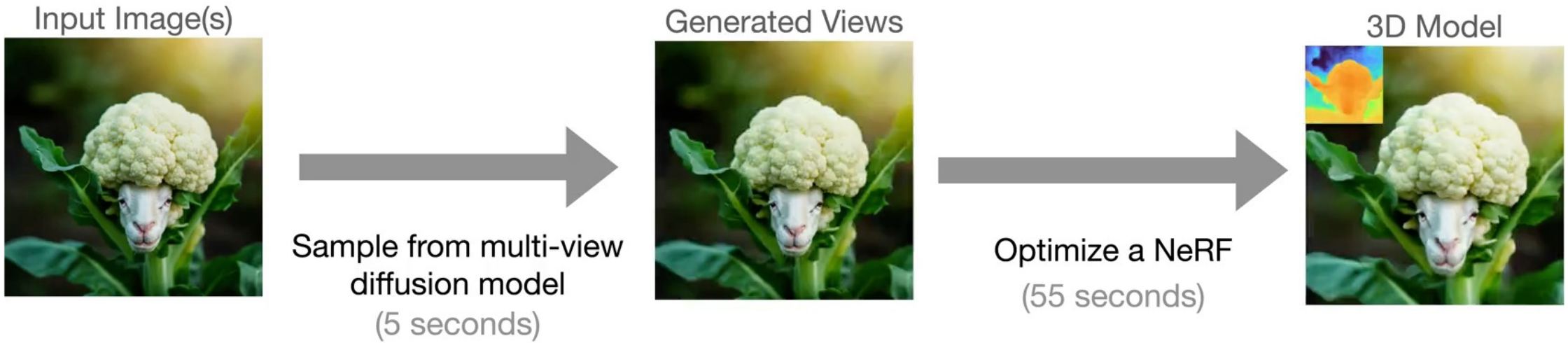


- Replace MLPs with 2D CNNs
- LBS parametric modeling
- Parameterizes 3D Gaussians onto front & back Gaussian maps

Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling. Li et al. CVPR 2024.

CAT3D: Create Anything in 3D with Multi-View Diffusion Models

Ruiqi Gao[1]* Aleksander Holynski[1]*
Philipp Henzler[2] Arthur Brussee[1] Ricardo Martin-Brualla[2]
Pratul P. Srinivasan[1] Jonathan T. Barron[1] Ben Poole[1]*
[1]Google DeepMind [2]Google Research *equal contribution



CAT3D: Create Anything in 3D with Multi-View Diffusion Models. Zeng et al. ECCV 2024.

ReconX: Reconstruct Any Scene from Sparse Views with Video Diffusion Model

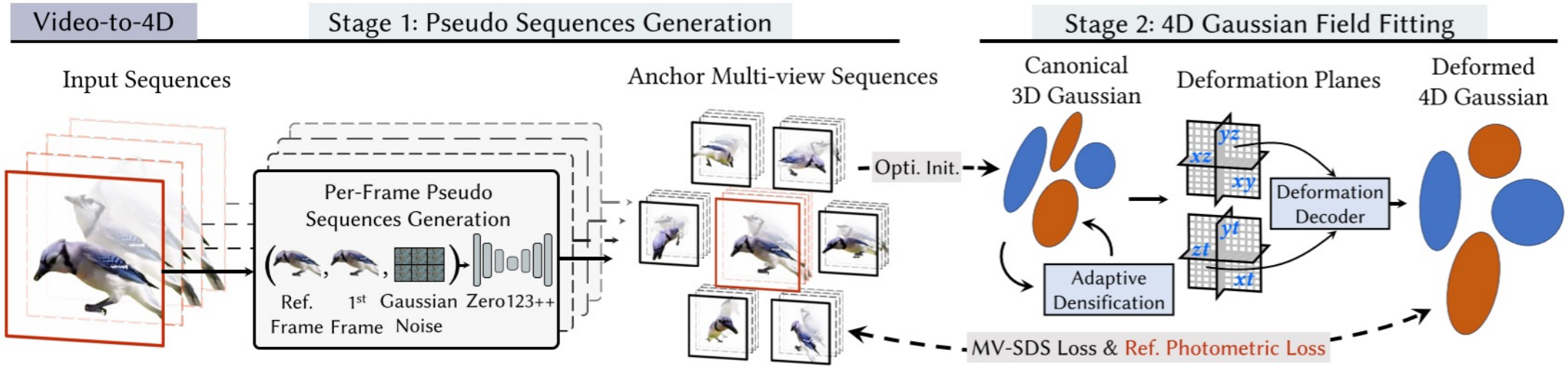
Fangfu Liu[1*], Wenqiang Sun[2*], Hanyang Wang[1*], Yikai Wang[1], Haowen Sun[1],
Junliang Ye[1], Jun Zhang[2], Yueqi Duan[1]
[1] Tsinghua University [2] HKUST

CAT3D: Create Anything in 3D with Multi-View Diffusion Models. Zeng et al. ECCV 2024.

STAG4D: Spatial-Temporal Anchored Generative 4D Gaussians

Yifei Zeng*[1], Yanqin Jiang*[2], Siyu Zhu[3], Yuanxun Lu[1], Youtian Lin[1], Hao Zhu[1], Weiming Hu[2], Xun Cao[1], Yao Yao[1]✉

[1]Nanjing University, [2]CASIA, [3]Fudan University,



STAG4D: Spatial–Temporal Anchored Generative 4D Gaussians. Zeng et al. ECCV 2024.

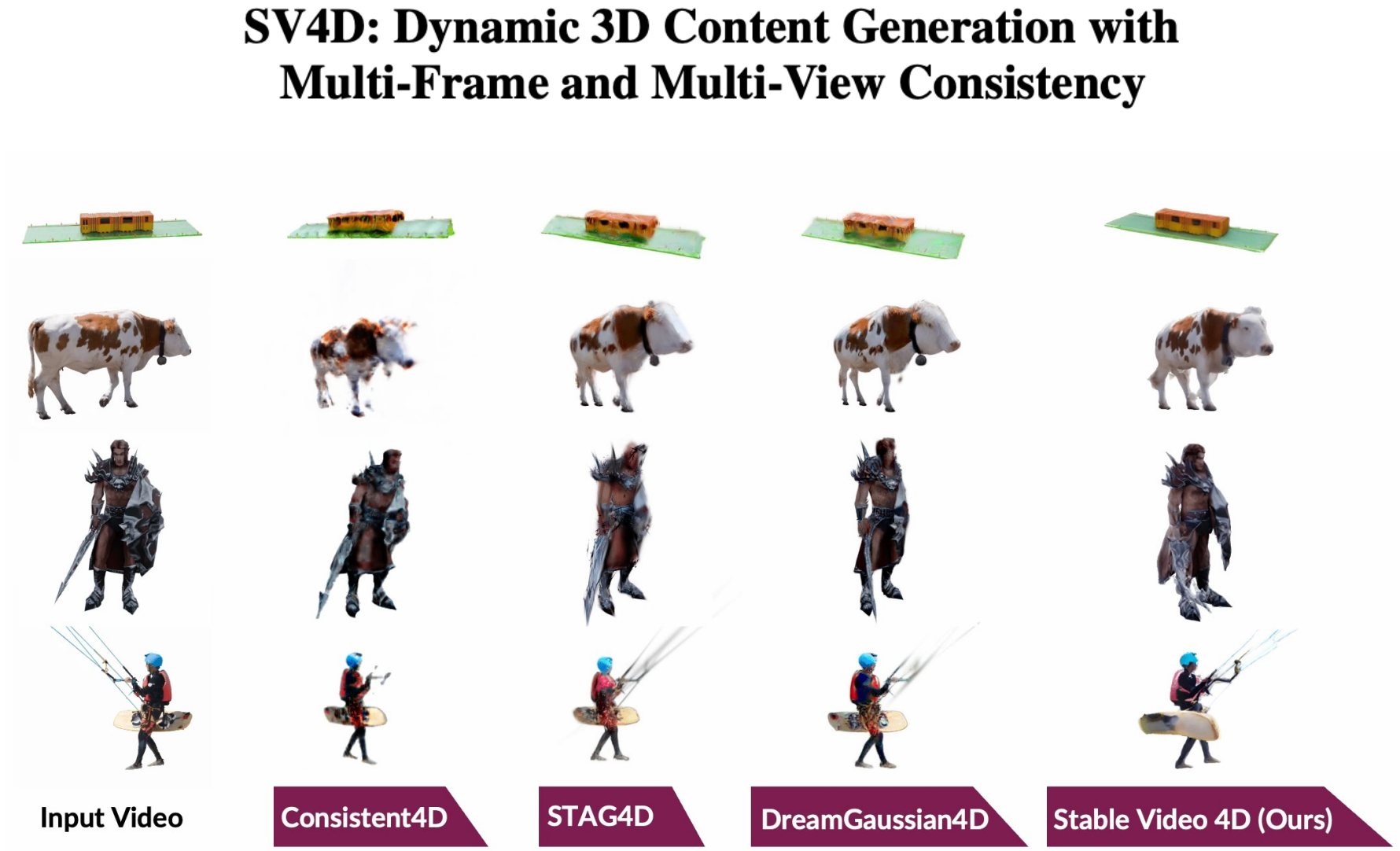
南京大学 NANJING UNIVERSITY

Priors | LVM | Video-to-4D Generation

STAG4D: Spatial-Temporal Anchored Generative 4D Gaussians. Zeng et al. ECCV 2024.

南京大学 NANJING UNIVERSITY

Priors | LVM | Video-to-4D Generation



SV4D: Dynamic 3D Content Generation with Multi-Frame and Multi-View Consistency. Xie et al. arXiv 2024.

内容回顾

- 基于四维空间 (4D Representation) 的动态三维重建方法
- 基于形变场 (Deformation Field) 的动态三维重建方法
- 基于逐帧重建 (Per-frame Reconstruction) 的动态三维重建方法
- 基于模型先验 (Model Priors) 的动态三维重建方法

未来展望

- More efficient 4D representations
- Large vision model for better 4D reconstruction
- Blurred boundaries between 4D reconstruction and generation

Pointrix | 点云可微渲染框架

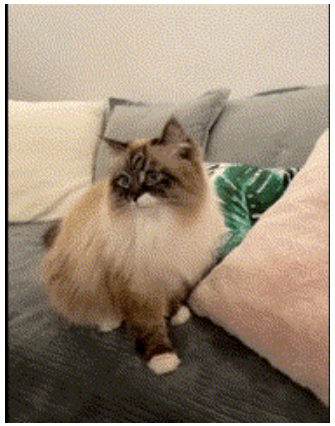
A differentiable point-based rendering library.

Document🏠 | Paper(Comming soon)📄 | DPTR Backend🌐

visitors 13 / 8380 stars repo not found Pointrix document

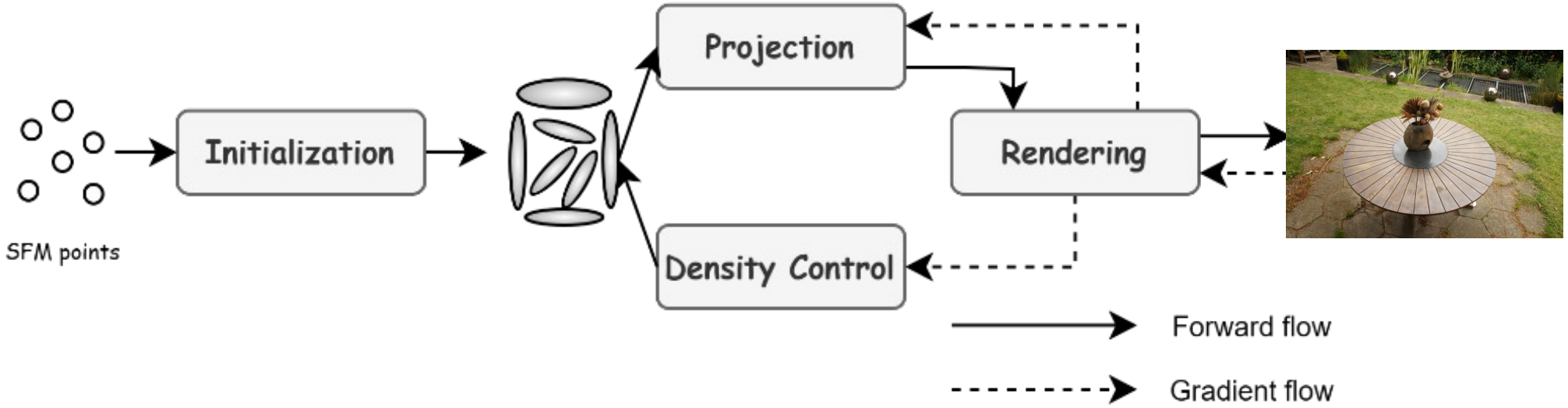
Pointrix is a differentiable point-based rendering library which has following properties:

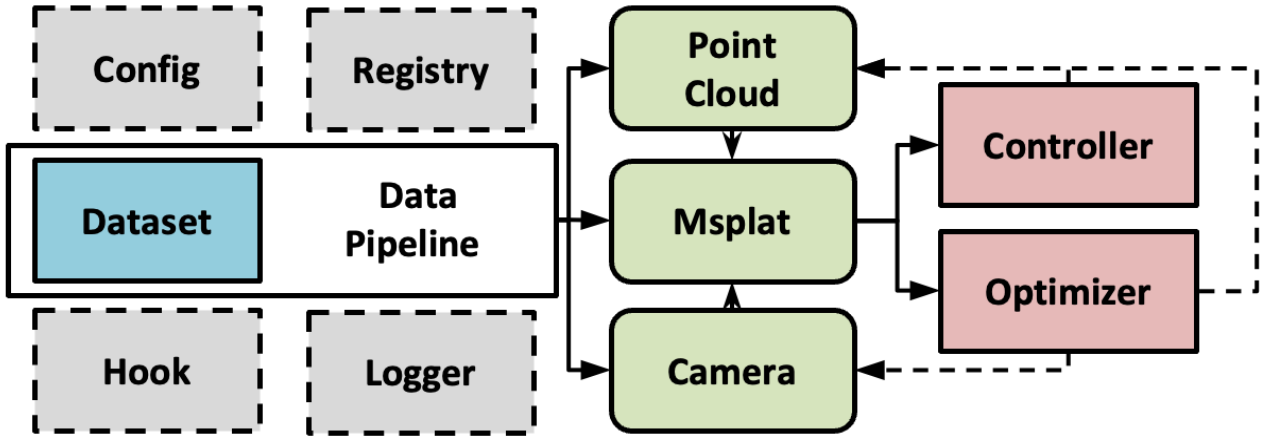
- **Highly Extensible**:
  - Python API
  - Modular design for both researchers and beginners
  - Implementing your own method without touching CUDA
- **Powerful Backend**:
  - CUDA Backend
  - Forward Anything: rendering image, depth, normal, optical flow, etc.
  - Backward Anything: optimizing even intrinsics and extrinsics.
- **Rich Features**:
  - 3D Reconstruction: Vanilla 3DGS (Siggraph 2023), Relightable 3DGS(WIP)
  - 4D Reconstruction: Deformable 3DGS (CVPR 2024), Gaussian-Flow (CVPR 2024, WIP)
  - 3D Generation: GSGen (CVPR 2024, WIP), DreamGaussian (ICLR 2024, TBD), LGM (arXiv 2024, TBD)
  - 4D Generation: STAG4D (arXiv 2024, WIP), DreamGaussian4D (arXiv 2023, TBD)



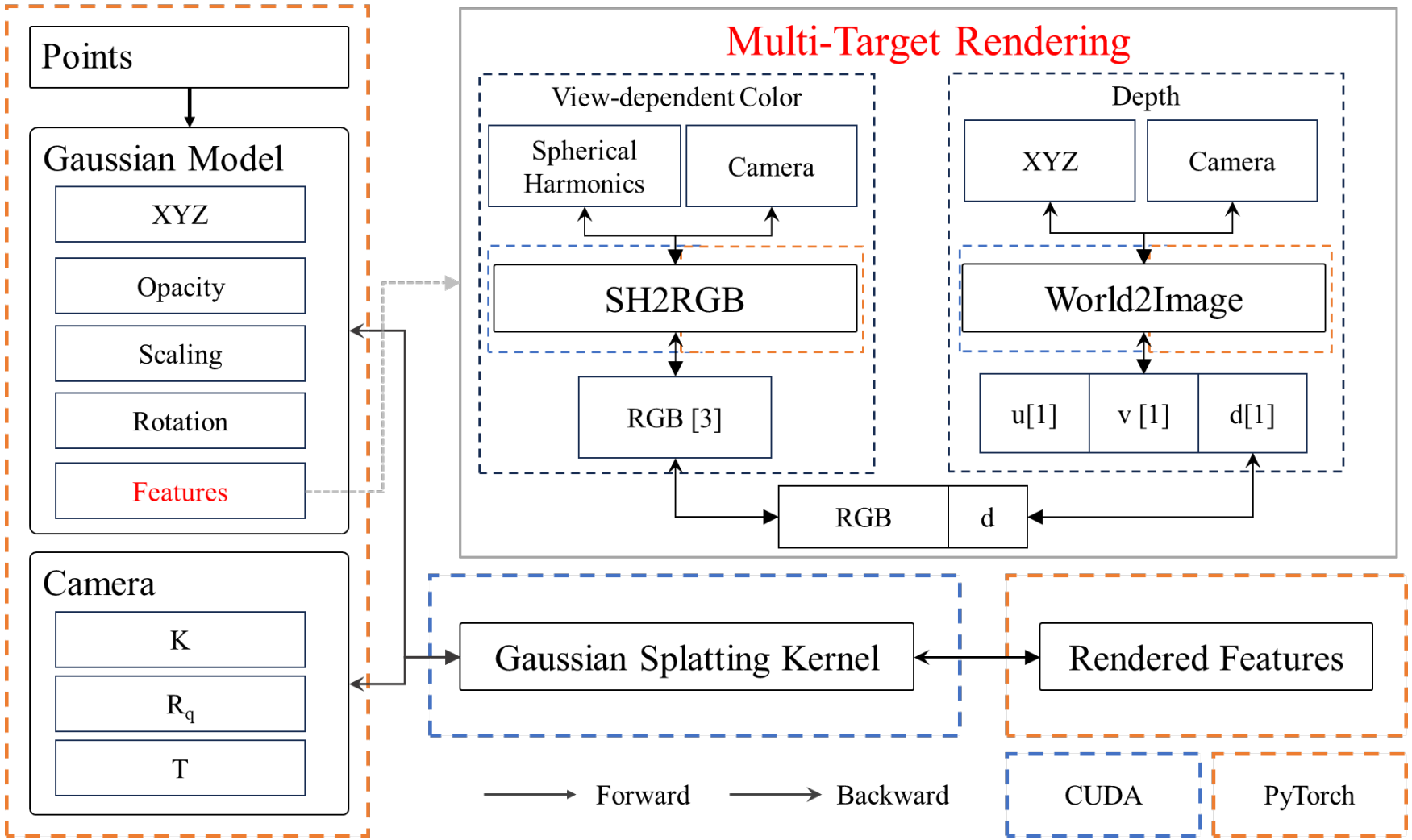
南京大学 NANJING UNIVERSITY

Pointrix | 设计框架





Pointrix: CUDA后端



Pointrix | 动态三维重建示例

```python
@MODEL_REGISTRY.register()
class DeformGaussian(BaseModel):
    def __init__(self, cfg, datapipeline, device="cuda"):
        super().__init__(cfg, datapipeline, device)

        # you can refer to projects/deformable_gaussian/model.py
        # if you want to know the detail of DeformNetwork.
        self.deform = DeformNetwork(is_blender=False).to(self.device)

        # The gaussian point cloud is implemeted in BaseModel, we
        # do not need to care about the detail here.

    def forward(self, batch):
        camera_fid = torch.Tensor([batch[0]['camera'].fid]).float().to(self.device)
        position = self.point_cloud.get_position
        time_input = camera_fid.unsqueeze(0).expand(position.shape[0], -1)
        d_xyz, d_rotation, d_scaling = self.deform(position, time_input)

        render_dict = {
            "position": self.point_cloud.position + d_xyz,
            "opacity": self.point_cloud.get_opacity,
            "scaling": self.point_cloud.get_scaling + d_scaling,
            "rotation": self.point_cloud.get_rotation + d_rotation,
            "shs": self.point_cloud.get_shs,
        }

        return render_dict

```

Pointrix | Normal Prior 示例

Modify configuration to read Normal data automatically.

```yaml
trainer:
    datapipeline:
        data_set: "ColmapDepthNormalDataset"
        shuffle: True
        batch_size: 1
        num_workers: 0
        dataset:
            data_path: "/home/linzhuo/gj/data/garden"
            cached_observed_data: ${trainer.training}
            scale: 0.25
            white_bg: False
            observed_data_dirs_dict: {"image": "images", "normal": "normals"}
```

new dataset func name

variable name: folder name

We highlight the modificated part.

```python
# Registry
@DATA_SET_REGISTRY.register()
class ColmapDepthNormalDataset(ColmapDataset):
    def _transform_observed_data(self, observed_data, split):
        cached_progress = ProgressLogger(description='transforming cached observed_data', suffix=
        cached_progress.add_task(f'Transforming', f'Transforming {split} cached observed_data', l
        with cached_progress.progress as progress:
            for i in range(len(observed_data)):
                # Transform Image
                image = observed_data[i]['image']
                w, h = image.size
                image = image.resize((int(w * self.scale), int(h * self.scale)))
                image = np.array(image) / 255.
                if image.shape[2] == 4:
                    image = image[:, :, :3] * image[:, :, 3:4] + self.bg * (1 - image[:, :, 3:4])
                observed_data[i]['image'] = torch.from_numpy(np.array(image)).permute(2, 0, 1).fl
                cached_progress.update(f'Transforming', step=1)

                # Transform Normal
                observed_data[i]['normal'] = \
                    (torch.from_numpy(np.array(observed_data[i]['normal'])) \
                    / 255.0).float().permute(2, 0, 1)
        return observed_data
```

process normal data

Pointrix | Normal Prior 示例

```python
@MODEL_REGISTRY.register()
class NormalModel(BaseModel):
    def forward(self, batch=None, training=True, render=True, iteration=None) -> dict:

        if iteration is not None:
            self.renderer.update_sh_degree(iteration)
        frame_idx_list = [batch[i]["frame_idx"] for i in range(len(batch))]
        extrinsic_matrix = self.training_camera_model.extrinsic_matrices(frame_idx_list) \
            if training else self.validation_camera_model.extrinsic_matrices(frame_idx_list)
        intrinsic_params = self.training_camera_model.intrinsic_params(frame_idx_list) \
            if training else self.validation_camera_model.intrinsic_params(frame_idx_list)
        camera_center = self.training_camera_model.camera_centers(frame_idx_list) \
            if training else self.validation_camera_model.camera_centers(frame_idx_list)

        point_normal = self.get_normals
        projected_normal = self.process_normals(
            point_normal, camera_center, extrinsic_matrix)

        render_dict = {
            "extrinsic_matrix": extrinsic_matrix,
            "intrinsic_params": intrinsic_params,
            "camera_center": camera_center,
            "position": self.point_cloud.position,
            "opacity": self.point_cloud.get_opacity,
            "scaling": self.point_cloud.get_scaling,
            "rotation": self.point_cloud.get_rotation,
            "shs": self.point_cloud.get_shs,
            "normals": projected_normal
        }
        if render:
            render_results = self.renderer.render_batch(render_dict, batch)
            return render_results
        return render_dict
```

get normal of points

```python
    def get_loss_dict(self, render_results, batch) -> dict:
        loss = 0.0
        gt_images = torch.stack(
            [batch[i]["image"] for i in range(len(batch))],
            dim=0
        )
        normal_images = torch.stack(
            [batch[i]["normal"] for i in range(len(batch))],
            dim=0
        )
        L1_loss = l1_loss(render_results['rgb'], gt_images)
        ssim_loss = 1.0 - ssim(render_results['rgb'], gt_images)
        loss += (1.0 - self.cfg.lambda_ssim) * L1_loss
        loss += self.cfg.lambda_ssim * ssim_loss
        # normal 监督的损失
        normal_loss = 0.1 * l1_loss(render_results['normal'], normal_images)
        loss += normal_loss
        loss_dict = {"loss": loss,
                     "L1_loss": L1_loss,
                     "ssim_loss": ssim_loss,
                     "normal_loss": normal_loss}
        return loss_dict
```

add normal loss

```python
@RENDERER_REGISTRY.register()
class MsplatNormalRender(MsplatRender):
    def render_iter(self, height, width, extrinsic_matrix, intrinsic_params, camera_center, posit
                    scaling, rotation, shs, normals, **kwargs) -> dict:
        direction = (position -
                     camera_center.repeat(position.shape[0], 1))
        direction = direction / direction.norm(dim=1, keepdim=True)
        rgb = msplat.compute_sh(shs.permute(0, 2, 1), direction)
        extrinsic matrix = extrinsic matrix[:3, :]
```

...

...

```python
        # sort
        (gaussian_ids_sorted, tile_range) = msplat.sort_gaussian(
            uv, depth, width, height, radius, tiles_touched
        )

        Render_Features = RenderFeatures(rgb=rgb, depth=depth, normal=normals)
        render_features = Render_Features.combine()

        ndc = torch.zeros_like(uv, requires_grad=True)
        try:
            ndc.retain_grad()
        except:
            raise ValueError("ndc does not have grad")

        # alpha blending
        rendered_features = msplat.alpha_blending(
            uv, conic, opacity, render_features,
            gaussian_ids_sorted, tile_range, self.bg_color, width, height, ndc
        )
        rendered_features_split = Render_Features.split(rendered_features)

        normals = rendered_features_split["normal"]

        # convert normals from [-1,1] to [0,1]
        normals_im = normals / normals.norm(dim=0, keepdim=True)
        normals_im = (normals_im + 1) / 2

        rendered_features_split["normal"] = normals_im

        return {"rendered_features_split": rendered_features_split,
                "uv_points": ndc,
                "visibility": radius > 0,
                "radii": radius
                }
```

Add new feature for alpha blending

Pointrix | 运行界面

```
Welcome to Pointrix

 _____       _       _          _
|  __ \     (_)     | |        (_)
| |__) |___   _  _ __  | |_  _ __  _ __  __
|  ___// _ \ | || '_ \ | __|| '__|| |\ \/ /
| |   | (_) || || | | || |_ | |   | | >  <
|_|    \___/ |_||_| |_| \__||_|   |_|/_/\_\

A light-weight differentiable point-based rendering framework.

Thank you
[20:42:23] The experiment name is /NASdata/clz/log/pointrix_new/test_update/garden@20240811-204223        log_hook.py:44
Loading train cached image ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 1222.38 iters/s    161/   161 0:00:00 00:00
Transforming train cached observed data ━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 1.86 iters/s    161/   161 0:01:25 01:25
Loading val cached image ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 2937.50 iters/s     24/    24 0:00:00 00:00
Transforming val cached observed data ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 1.89 iters/s     24/    24 0:00:12 00:12
[20:44:03] Number of points at initialisation :  138766                                      utils.py:29
Setting up [LPIPS] perceptual loss: trunk [vgg], v[0.1], spatial [off]
Loading model from: /home/clz/anaconda3/envs/pointrix_test/lib/python3.9/site-packages/lpips/weights/v0.1/vgg.pth
Training Progress ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17% 9.24 iter/s   4990/ 30000 0:06:08 45:08 loss: 0.0561943 num_pts: 3252125
Validation Progress ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0.71 iter/s     24/    24 0:00:32 00:32

[ITER 5000] Evaluating test:
```

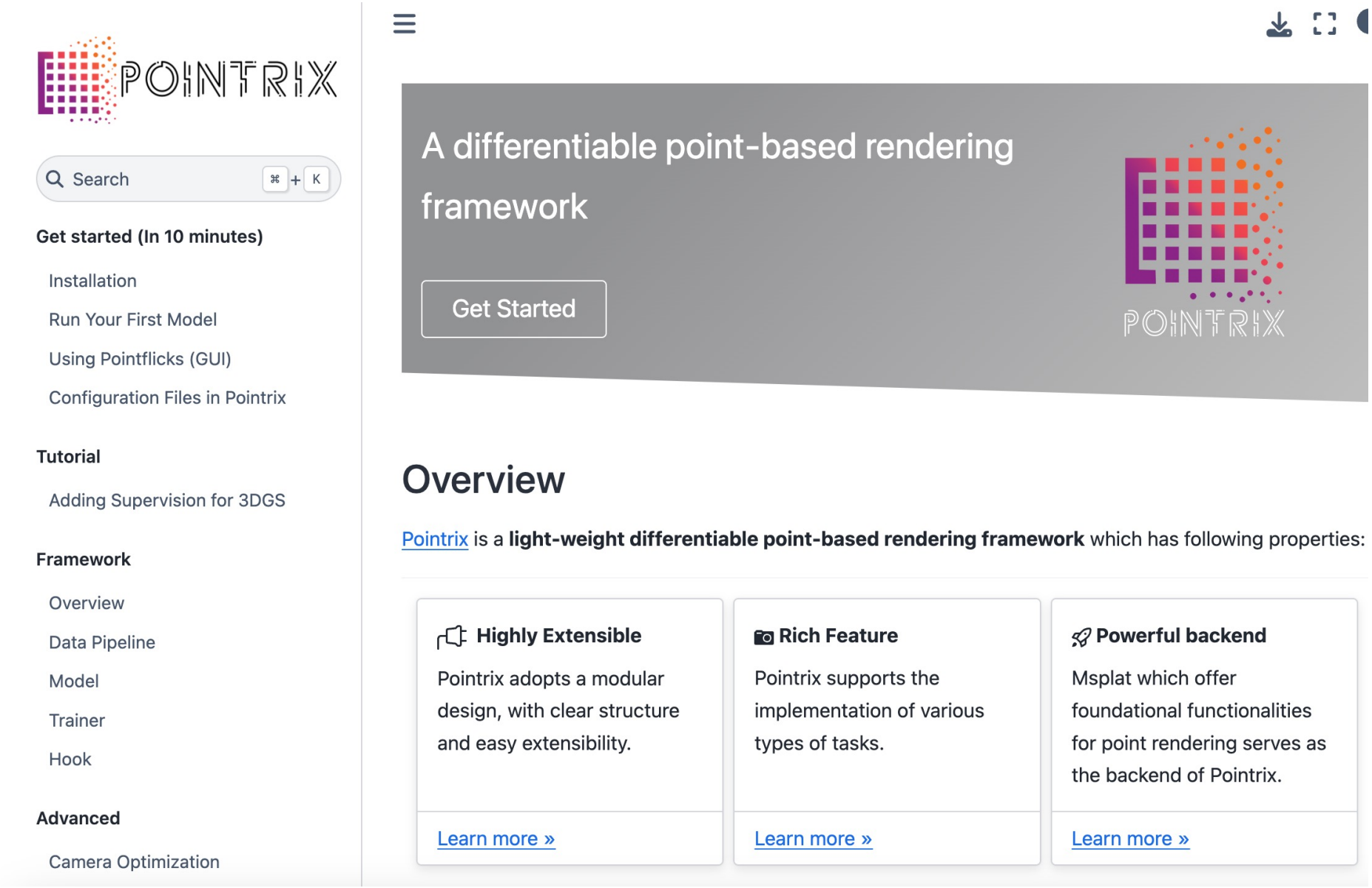
L1_loss	psnr	ssims	lpips
0.03359	25.80855	0.80729	0.19114

```
Training Progress ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33% 7.16 iter/s   9990/ 30000 0:17:53 46:34 loss: 0.0437142 num_pts: 4570833
Validation Progress ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0.73 iter/s     24/    24 0:00:31 00:31
```

南京大学 NANJING UNIVERSITY

Pointrix | Interactive GUI

Pointrix | Documentation



南京大学 NANJING UNIVERSITY

Pointrix | 中文文档

POINTRIX

Search ⌘ + K

开始（10分钟内学习Pointrix）

- 安装
- 运行你的第一个模型
- Pointrix 中的配置文件
- 使用 Pointflicks (GUI) 可视化

教程

- 为点云渲染添加监督

架构

- 总览
- 数据流水线
- 模型
- 训练器 (Trainer)
- 钩子函数

Advanced

- 相机优化

Overview

Pointrix 是一个 **轻量可微分点云渲染框架** 并具有以下几个特点:

高度可扩展性

Pointrix 采用模块化设计，具有清晰的结构和强大的扩展性。

丰富的特性

Pointrix 支持各种不同类型的工作实现，包括静态动态场景重建，生成以及PBR渲染。

强大的后端

MSplat 作为Pointrix的后端，支持各类点云渲染功能：包括各类特征的渲染，超高阶的sh阶数，以及所有输入的梯度反传。

南京大学 NANJING UNIVERSITY

Thanks!

Yao Yao

Nanjing University

Linzhuo Chen

Youtian Lin

南京大學
NANJING UNIVERSITY