CVSP 2026 Summer Seminar

Towards Explicit 3D Representation: Mesh Reconstruction for Static and Dynamic Scenes



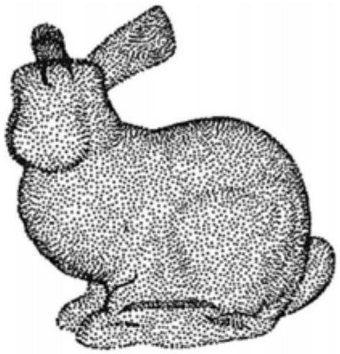
Pusan National University
Dept. of Electronics Engineering
Computer Vision & Signal Processing Lab.



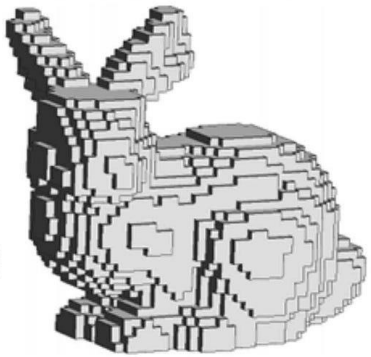
Presented By
석사과정 김태훈

Background

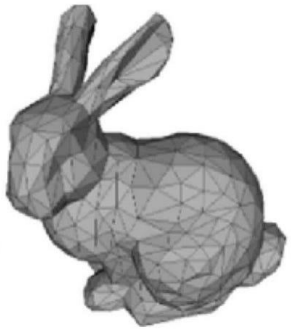
- 3D Representation
  - Explicit Representation
    - Voxel
    - Mesh
    - Primitive Splatting
      - Point Cloud, Gaussian Splatting
  - Implicit Representation
    - Signed Distance Function
    - Neural Radiance Fields (NeRF)



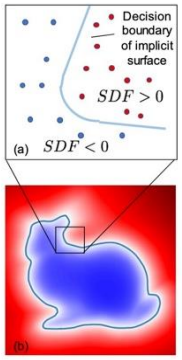
Point Cloud



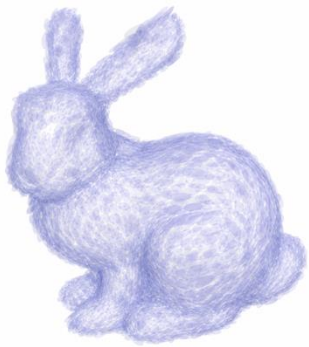
Voxel



Mesh



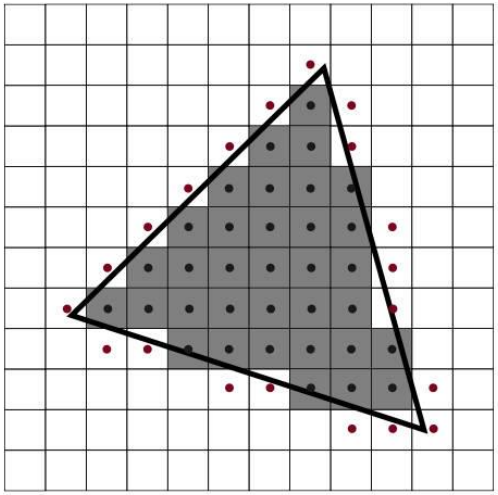
Implicit Function



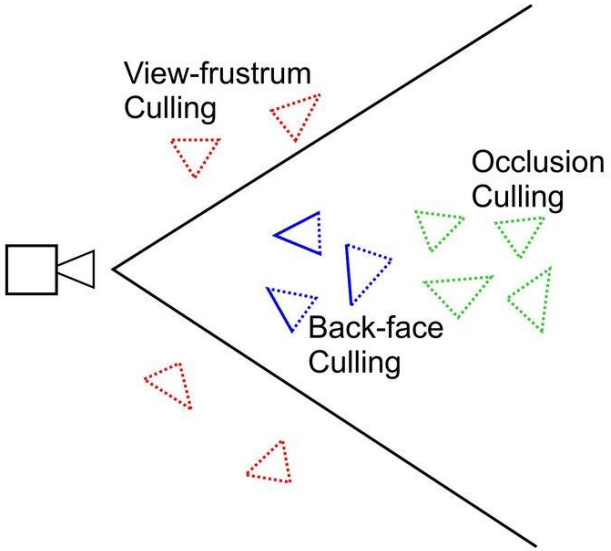
Gaussian Splatting

Background

- 3D Representation
  - Rasterization
    - 3D Primitive를 Camera View의 2D Pixel Space로 변환하는 과정
    - Primitive: 3D를 표현하는 단위 표현(Unit Representation)



Rasterization

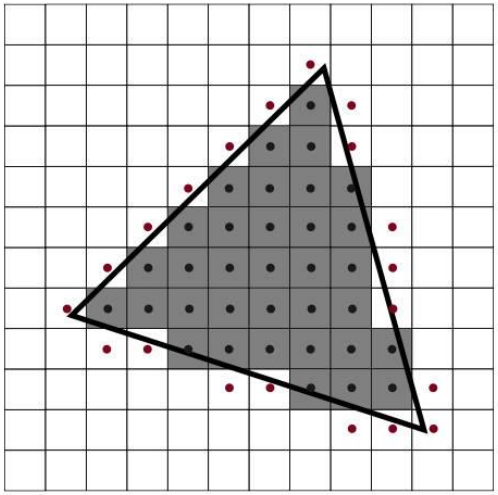


Mesh Culling

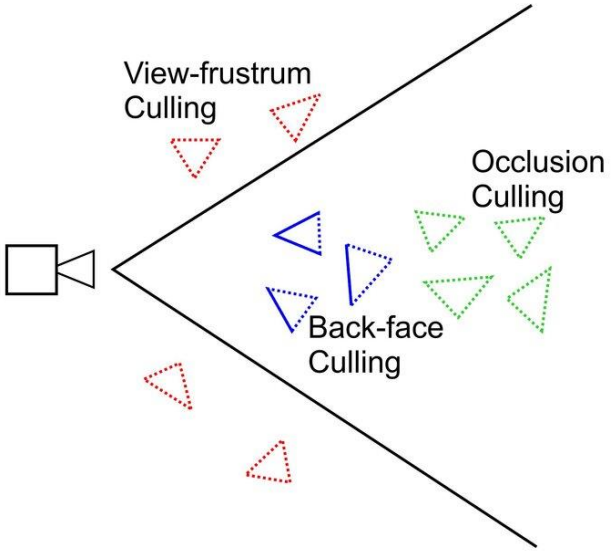
Alpha-blending

Background

- 3D Representation
  - Rasterization
    - Mesh Rendering
      - Culling: 가려지거나 보이지 않는 부분을 제외하여 효율적 렌더링
    - Alpha-blending
      - 불투명도가 1이 될 때까지 반투명 Representation을 섞어줌



Rasterization

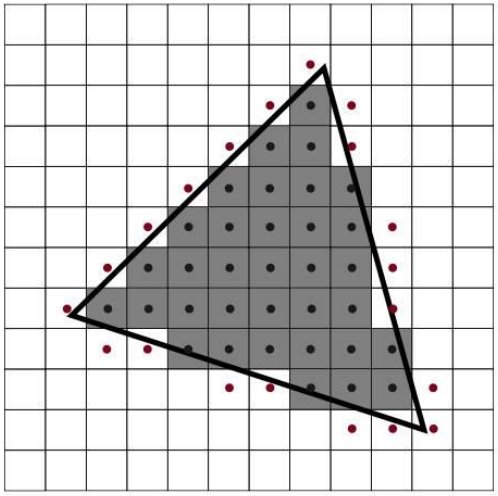


Mesh Culling

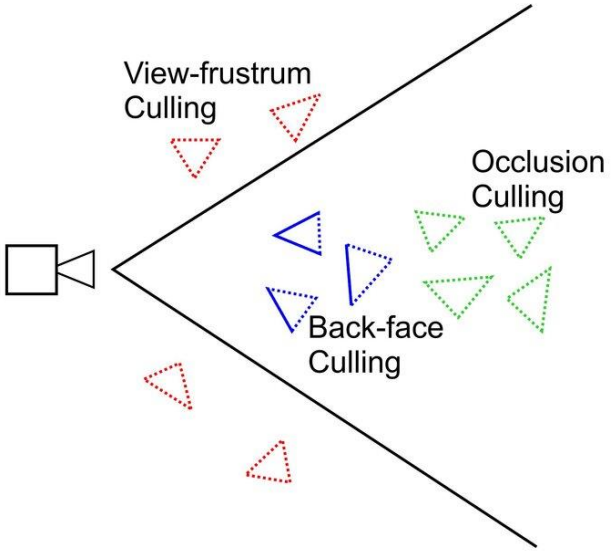
Alpha-blending

Background

- 3D Representation
  - 3D Gaussian Splatting
    - 위치와 크기를 Gaussian Kernal을 통해 Volume표현
      - Splatting: 각 Primitive가 주변 Pixel에 끼치는 영향도를 계산하는 과정



Rasterization



Mesh Culling

Alpha-blending

[CVPR 2026 Oral]
MeshSplatting:
Differentiable Rendering with Opaque Meshes

Background

- Novel View Synthesis (NVS)
  - 관측된 여러 시점(Multi-view)의 이미지로부터 새로운 view의 이미지를 합성
  - 직접 촬영하지 않은 카메라 위치에서도 scene을 렌더링 가능



MeshSplatting

- Problems
  - 3D Representation Conversion
    - 3DGS로부터 Mesh를 얻기 위해 별도의 Mesh extraction / 후처리 과정 필요
      - 추가 연산 필요, Visual Quality 감소
  - Primitive Splatting Rendering
    - 3DGS와 같은 Splatting 방식은 여러 primitive를 깊이 순서대로 정렬하고 색을 합성
      - 기존의 효율적인 Mesh rendering pipeline을 그대로 사용하기 할 수 없음
    - Mesh와 유사한 형태인 Triangle을 primitive로 사용하더라도, 각각의 Triangle이 서로 독립적으로 존재
      - Physics simulation, game engine, editing 등의 3D application에서는 Connected Mesh가 필요

MeshSplatting

- Method Overview
  - Triangle Primitive들이 Vertex를 공유하도록 구성
    - 최종 결과물이 Connected Mesh
  - 학습 초기에는 Triangle들을 자유롭게 최적화하고, 이후 서로 연결된 Mesh로 변환
    - 처음부터 연결 관계를 강제하면 최적화가 어렵기 때문에 단계적으로 Mesh를 구성
  - 최적화 iteration에 따라서 Primitive가 불투명해지도록 조절
    - 최종 결과물이 Opaque Mesh가 되어 alpha-blending 없이 일반적인 Mesh rendering 방식으로 사용 가능



Triangle Soup Optimization → Mesh Creation → Mesh Refinement

MeshSplatting

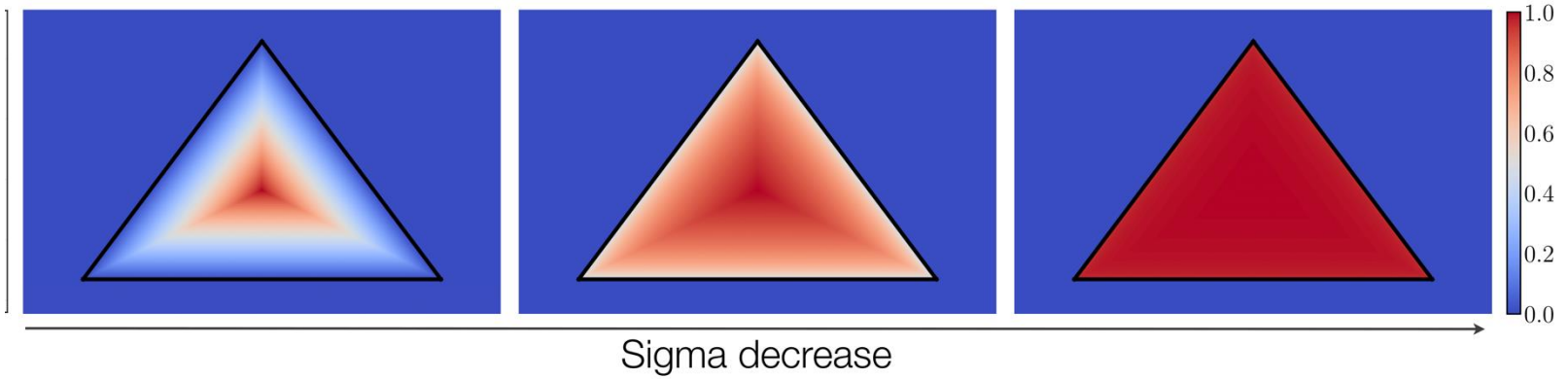
- Method
  - Preliminary
    - Triangle Primitive
      - 세 점 위치와 한 개의 면의 색/불투명도와 SDF(Signed Distance Function) ϕ 를 이용해 표현
      - 한 Pixel $\mathbf{p}$에서의 Triangle Primitive의 Contribute(Splatting) $I(\mathbf{p})$

$$I(\mathbf{p}) = \left(\mathrm{ReLU}\left(\frac{\phi(\mathbf{p})}{\phi(\mathbf{s})} \right) \right)^{\sigma}$$

✓삼각형의 경계에서 내심으로 0→1 로 값이 바뀜

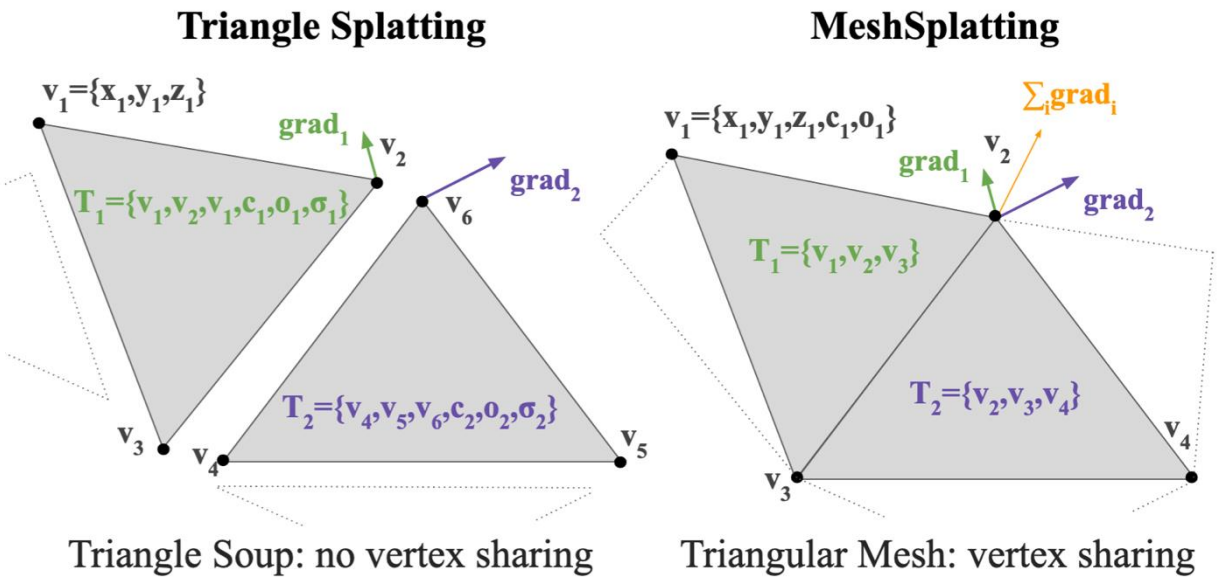
✓$\phi(\boldsymbol{s})$: 삼각형 내심에서의 SDF 값 = 해당 삼각형 내부에서의 최소값

✓σ: $1 \rightarrow 0$ 으로 줄어듦에 따라 삼각형의 경계가 뚜렷해짐



MeshSplatting

- Method
  - Vertex Sharing Triangle Primitives
    - Triangle Primitive에서 각 점에 색상과 opacity를 할당하도록 변형한 형태
      - 내부 색상은 세 점에 할당된 색상에 대해 무게 중심 거리로 가중치 합으로 결정
      - 불투명도는 세 점에 할당된 불투명도의 최소값 사용
      - Connected Triangle에 대해서 Accumulated Gradient 로 학습



MeshSplatting

- Method
  - Optimization Strategy
    - 학습 초기에는 SfM 초기화 점부터 시작
    - Triangle들을 자유롭게 최적화하고, 이후 서로 연결된 Mesh로 변환
    - 최적화 iteration에 따라서 Primitive가 불투명해지도록 조절하여 Mesh 성질



Triangle Soup Optimization



Mesh Creation



Mesh Refinement

MeshSplatting

- Method
  - Optimization Strategy
    - Triangle soup optimization
      - 초기에는 Triangle 사이의 연결을 강제하지 않고 각각 자유롭게 최적화
      - 연결 제약이 없기 때문에 Scene의 Geometry / Appearance에 빠르게 적응
      - 초기에는 반투명($\sigma = 1$)하게 설정하여 뒤쪽 Triangle까지 Gradient가 전달되도록 함



Triangle Soup Optimization



Mesh Creation



Mesh Refinement

MeshSplatting

- Method
  - Optimization Strategy
    - Mesh create & refine
      - 최적화한 Triangle Primitive의 Vertex를 각각의 점으로 취급하고 Restricted Delaunay Triangulation으로 연결
        - ✓기존 Vertex 위치를 유지하면서 Triangle 사이의 연결 관계를 생성
      - Mesh 생성 이후 Vertex Position과 Appearance를 다시 최적화



Triangle Soup Optimization



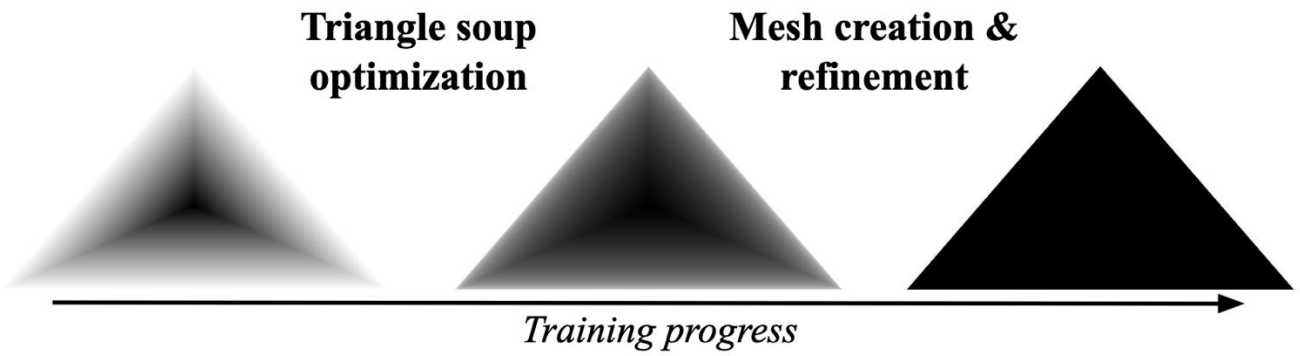
Mesh Creation



Mesh Refinement

MeshSplatting

- Method
  - Optimization Strategy
    - Mesh like Opaque
      - 학습 초기에는 반투명하게 두고, iteration에 따라 Opacity를 점차 증가
      - 최종적으로 Fully Opaque Triangle로 수렴
      - 최종 Triangle는 Alpha-Blending 없이 일반 Mesh Renderer에서 사용 가능



MeshSplatting

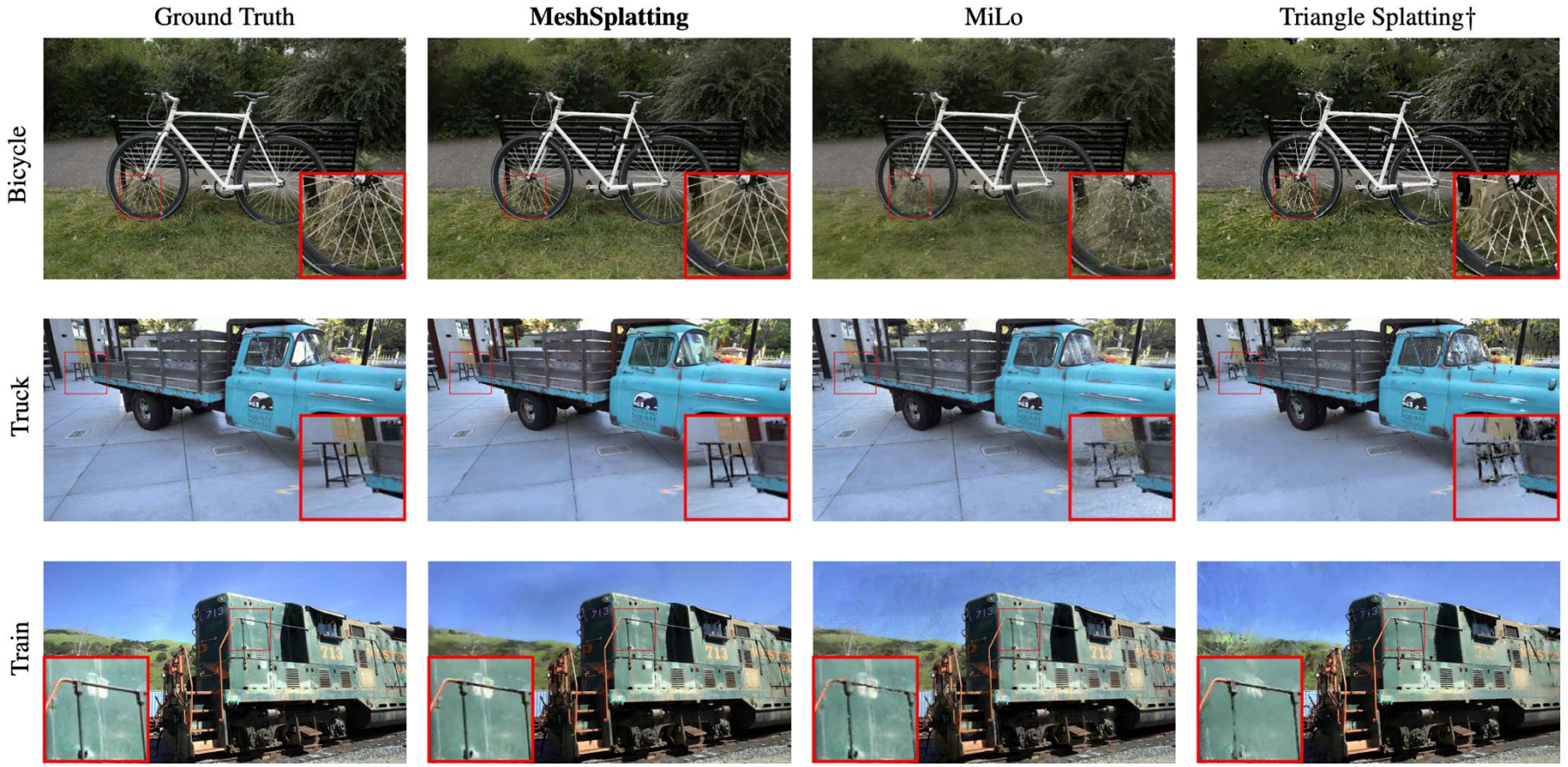
- Experiments
  - Mesh-based NVS
    - Mesh 계열에서 적은 Vertex를 가지면서도 준수한 성능 달성

	Characteristics				Mip-NeRF360 dataset				Tanks & Temples			
	Mesh	Color	Connect	Ready	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	$SSIM^{\uparrow}$	$\|V\|^{\downarrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	$SSIM^{\uparrow}$	$\|V\|^{\downarrow}$
3DGS[26]	–	–	–	✗	27.21	0.214	0.815	–	23.14	0.183	0.841	–
Triangle Splatting [16]	-	-	-	✗	27.16	0.191	0.814	-	23.14	0.143	0.857	-
2DGS [20]	✗	✗	✓	✓	15.36	0.474	0.498	2M	14.23	0.485	0.569	16M
GOF [53]	✗	✗	✓	✓	20.78	0.465	0.573	33M	21.69	0.326	0.690	12M
RaDe-GS [54]	✗	✗	✓	✓	23.56	0.361	0.668	31M	20.51	0.344	0.659	10M
MiLo [14]	✓	✗	✓	✓	24.09	0.323	0.688	7M	21.46	0.348	0.706	4M
Triangle Splatting [16] †	✓	✓	✗	✓	21.05	0.462	0.558	3M	17.27	0.402	0.600	6M
MeshSplatting	✓	✓	✓	✓	24.78	0.310	0.728	3M	20.52	0.287	0.745	2M

Method	Train ↓	FPS ↑ (HD)	FPS ↑ (Full HD)	Memory ↓
GOF	74m	OOM	OOM	1.5GB
RaDe-GS	84m	OOM	OOM	1.1GB
MiLo	106m	170	160	253MB
MeshSplatting	48m	220	190	100MB

MeshSplatting

- Experiments
  - Qualititive Results



MeshSplatting

- Experiments
  - Applications
    - Simulation(Robotics, RigidBody), RayTracing, Object Segmentation



Game Engine



Collision



Ray Traycing



Object Segmentation

[ICLR 2025]
Dynamic Gaussians Mesh:
Consistent Mesh Reconstructionfrom Dynamic Scenes

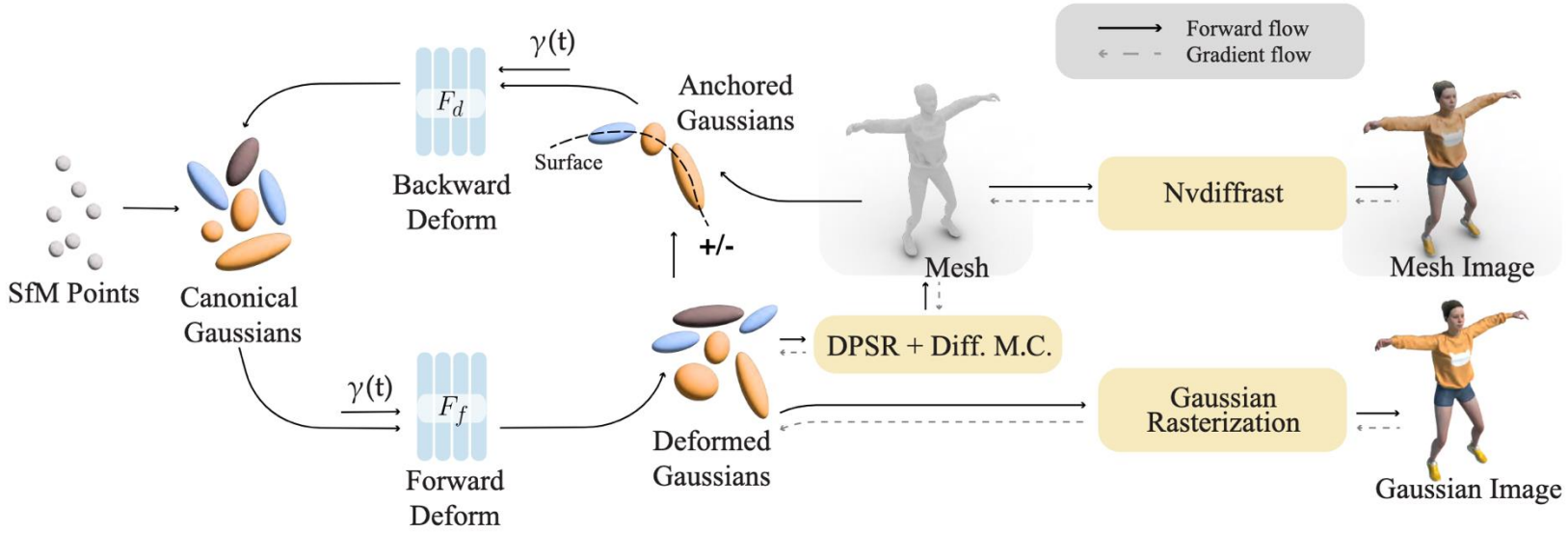


Dynamic Gaussian Mesh

- Problems
  - Dynamic Mesh Reconstruction Temporal Correspondence
    - Dynamic 3DGS는 동일한 Gaussian을 시간에 따라 변형하므로 Gaussian-level Temporal Correspondence는 존재
      - Gaussian 사이의 Spatial Connectivity가 없어 Mesh의 Vertex / Face correspondence는 정의되지 않음
    - time step에서 독립적으로 Mesh를 추출하면 Vertex index와 topology가 달라질 수 있음
      - 한 frame의 editing을 다른 frame으로 전달하기 어려움
  - Uneven Gaussian Distribution
    - 3DGS densification은 Appearance가 복잡한 영역에 Gaussian을 집중시킴
      - DPSR의 uniform sampling 가정을 위반하여 Mesh reconstruction quality 감소

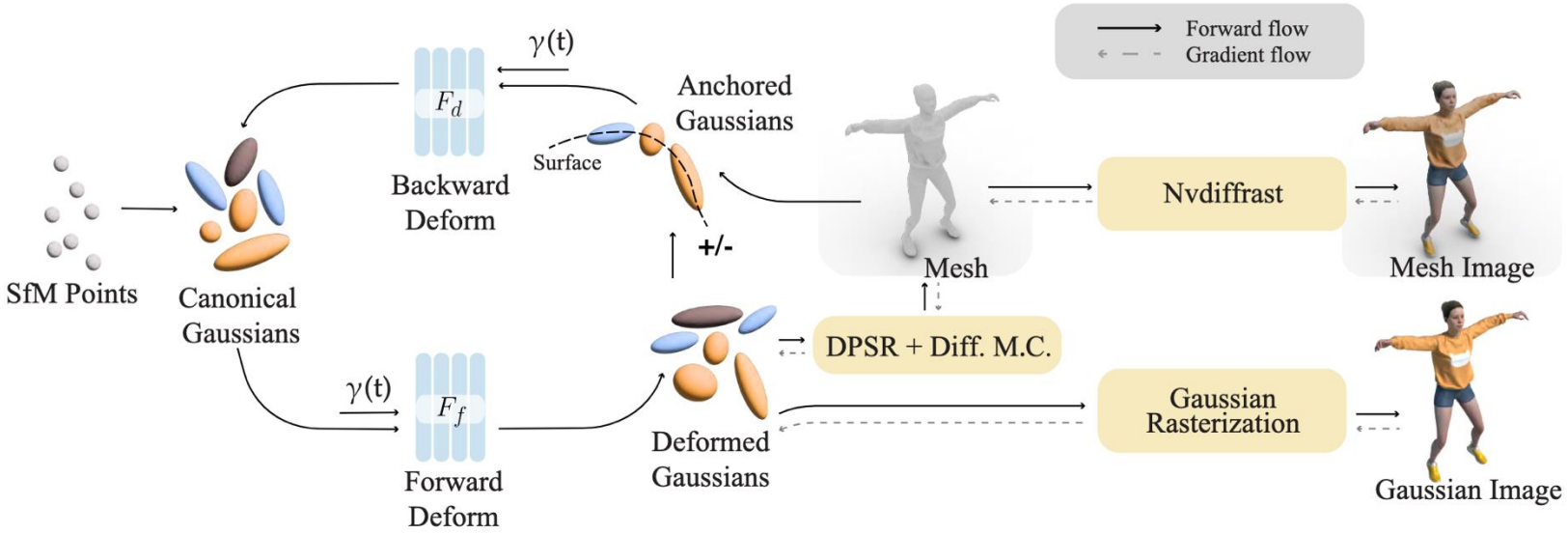
Dynamic Gaussian Mesh

- Method Overview
  - Canonical Gaussian을 각 time step의 Deformed Space로 변환
    - 변형된 Gaussian을 Oriented Point Cloud로 사용
  - DPSR + Differentiable Marching Cubes로 Mesh를 추출
    - Gaussian / Mesh rendering loss로 Appearance와 Geometry를 함께 최적화
  - Gaussian-Mesh Anchoring으로 Gaussian을 Mesh face에 균일하게 배치
    - Backward Deformation으로 수정된 Gaussian을 Canonical Space에 반영



Dynamic Gaussian Mesh

- Method
  - Preliminary: Deformable 3D Gaussian Splatting
    - Canonical Space에서 Gaussian $G = (\mu, r, s, \alpha)$를 학습
    - Deformation Network가 parameter 변화량을 예측
      - $F_f(\gamma(x), \gamma(t)) = (\delta x, \delta r, \delta s, \delta \alpha)$
      - 변화량을 Canonical Gaussian에 적용하여 G_t를 생성
      - 동일한 Gaussian을 변형하므로 frame 간 point correspondence 유지

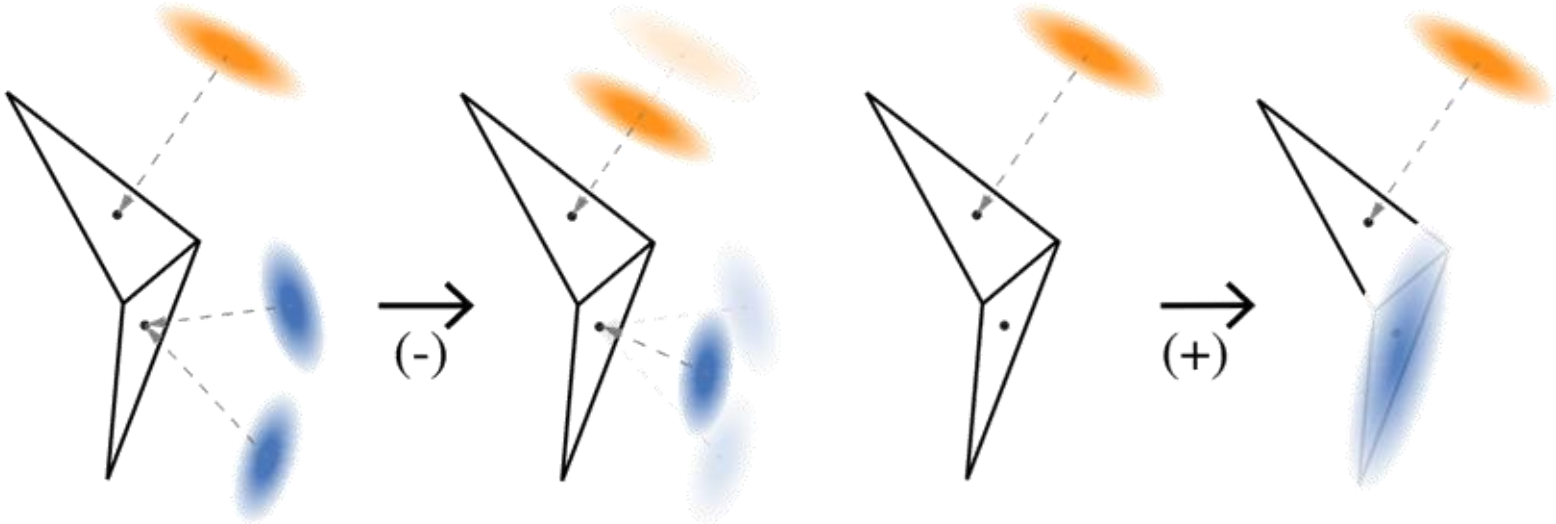


Dynamic Gaussian Mesh

- Method
  - Gaussian-Mesh Anchoring
    - 빈틈없이 Mesh 면을 채우도록 강제하도록 Anchor Loss 제안
      - GS 최적화 특성상 복잡한 부분에 GS 수가 많아지는 경향이 있음
      - Mesh 면에 대해 부족한 부분은 GS 생성, 초과 부분은 Pruning

$$n_{x_i} = \arg\min_f \|x_i - f\|, f \in \{f_j\}_{j=0}^{M}, \qquad \mathcal{L}_{anchor} = \frac{1}{n}\sum_{i=0}^{n}\|x_i - n_{x_i}\|^2,$$

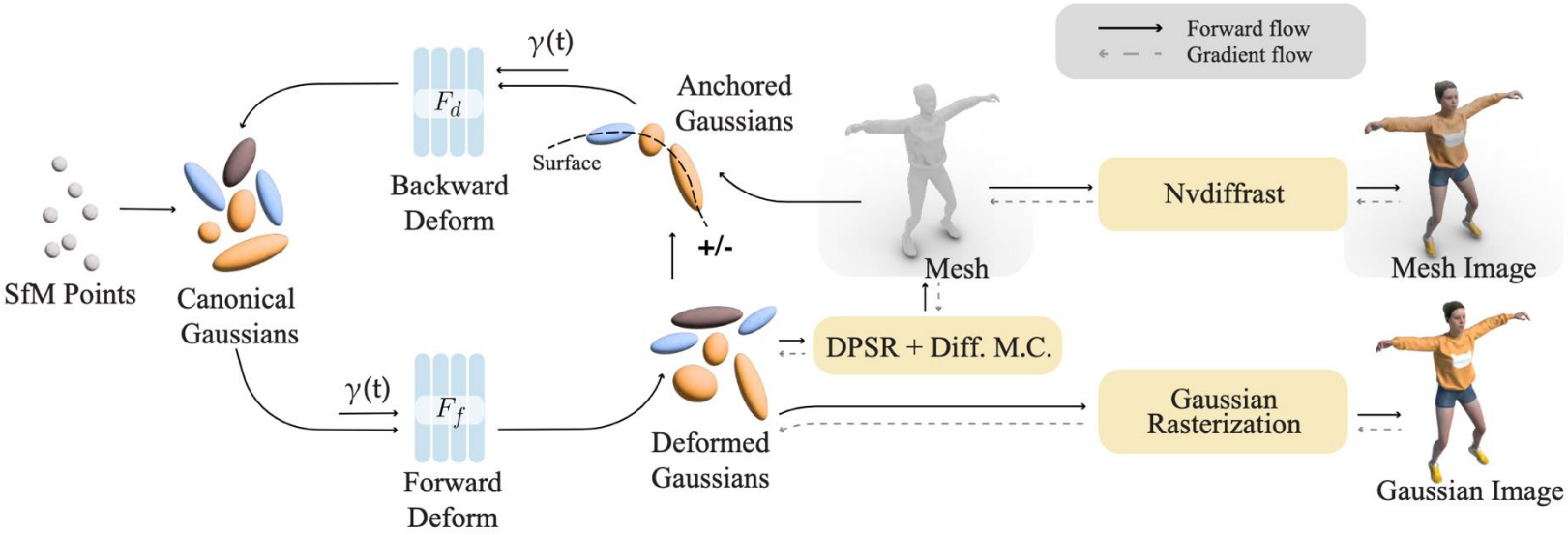
      - f:Mesh Centroid, $\mathrm{G}(x)$:Deformed 3DGS, x: 3DGS Position



Dynamic Gaussian Mesh

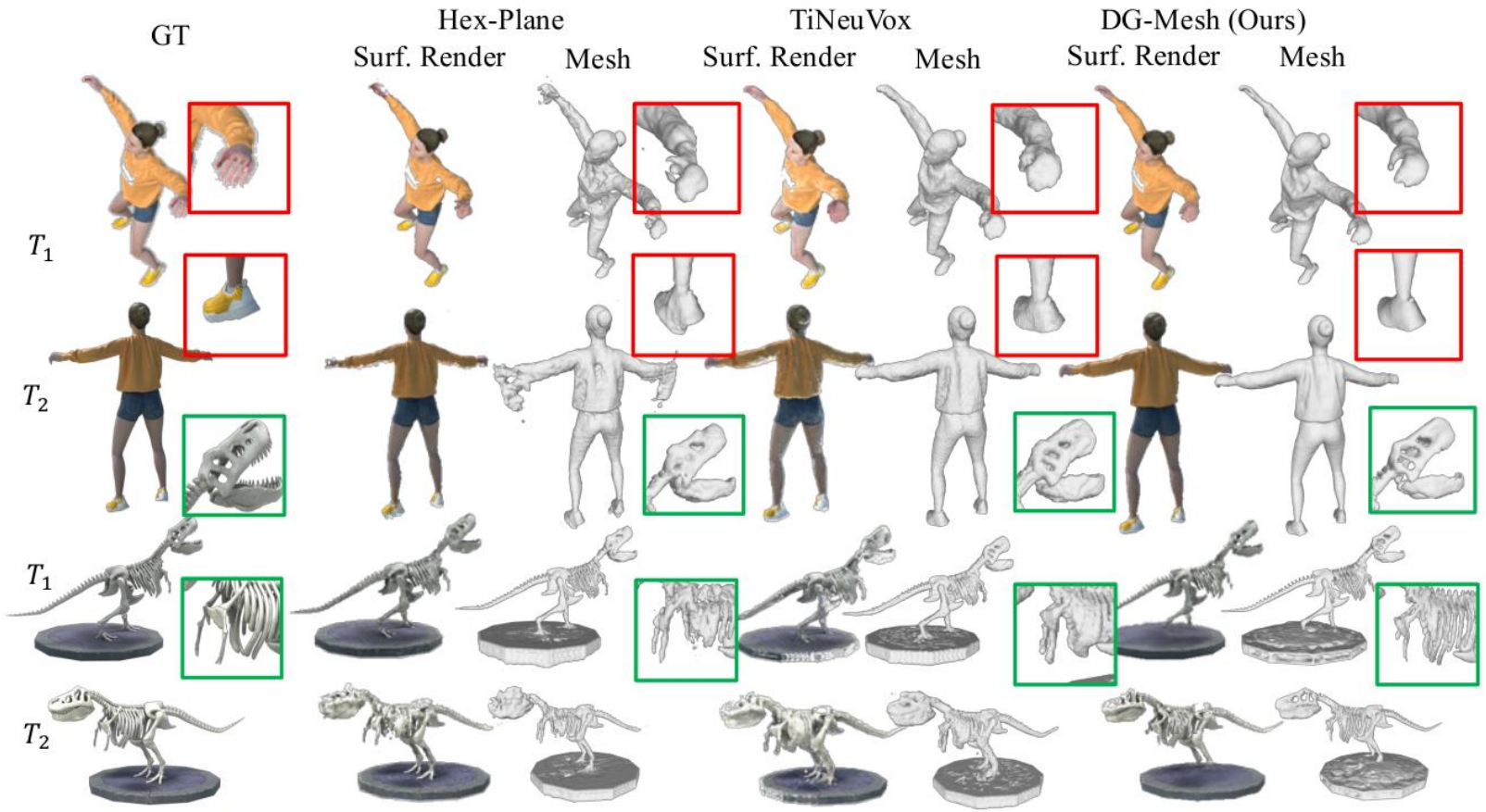
- Method
  - Cycle-Consistent Deformation
    - Gaussian-Mesh Anchoring으로 수정한 G'가 Deformation Network $\mathcal{F}_f$ 에 대해 출력될 수 있도록 Canonical Space에 Mapping하는 별도의 Backward Deformation Network $\mathcal{F}_b$구성

$$\mathcal{F}_f(G,t) = -\mathcal{F}_b(G',t),$$
$$\mathcal{L}_{\text{cycle}} = \left\| \mathcal{F}_f(G,t) - \left(-\mathcal{F}_b(G',t)\right) \right\|_1$$



Dynamic Gaussian Mesh

- Experiments
  - Qualitative Results



Dynamic Gaussian Mesh

- Experiments
  - Mesh Reconstruction Quality (DG-Mesh Dataset)
    - GT Mesh를 포함한 합성 데이터셋에서 CD, EMD로 Geometry 정확도 평가
      - Chamfer Distance(CD): GT, Predicted Mesh 에서 Point Sampling하여 point간 distance 평균
      - Earth Mover's Distance(EMD): Predict Mesh의 Point를 옮겨 GT Point로 옮길 때, 최소로 이동한 총 거리.
        - ✓점이 한 곳에 몰려 있어 균질하지 않으면 높은 값을 가짐(안좋음)

Method	Duck			Horse			Bird		
	CD ↓	EMD ↓	$PSNR_m$ ↑	CD ↓	EMD ↓	$PSNR_m$ ↑	CD ↓	EMD ↓	$PSNR_m$ ↑
D-NeRF	0.934	0.073	23.019	1.685	0.280	17.381	1.532	0.163	19.573
K-Plane	1.085	0.055	20.372	1.480	0.239	21.629	0.742	0.131	19.559
HexPlane	2.161	0.090	27.945	1.750	0.199	22.395	4.158	0.178	20.595
TiNeuVox-B	0.969	0.059	22.073	1.918	0.246	18.156	8.264	0.215	19.844
4D-GS	1.134	0.111	-	1.500	0.272	-	2.311	0.187	-
Deformable-GS	2.366	0.115	-	1.510	0.217	-	1.358	0.141	-
4DGS	5.895	0.169	-	0.895	0.340	-	20.426	0.277	-
SC-GS	1.306	0.097	-	0.897	0.177	-	0.897	0.166	-
DG-Mesh (Ours)	**0.782**	**0.047**	**32.757**	**0.297**	**0.164**	**30.865**	**0.510**	**0.125**	**28.085**

Dynamic Gaussian Mesh

- Experiments
  - Mesh Rendering Quality (D-NeRF Dataset)
    - Mesh Rendering 기준 PSNRm / SSIM / LPIPS에서 Baseline 대비 최고 성능
    - Mesh 기반이므로 Rendering이 효율적
      - Nerfies 기준 Mesh Query 10.9 FPS (Deformable-GS = 0.10 FPS)

Method	Bouncingballs			Jumpingjacks			Mutant			Trex		
	PSNR$_m$ ↑	SSIM ↑	LPIPS ↓	PSNR$_m$ ↑	SSIM ↑	LPIPS ↓	PSNR$_m$ ↑	SSIM ↑	LPIPS ↓	PSNR$_m$ ↑	SSIM ↑	LPIPS ↓
D-NeRF	23.398	0.899	0.157	22.255	0.914	0.103	21.070	0.906	0.077	22.594	0.908	0.085
K-Plane	23.307	0.935	0.109	25.240	0.937	0.068	23.226	0.923	0.064	23.093	0.921	0.075
HexPlane	25.389	0.957	0.069	27.078	0.954	0.052	26.811	0.953	0.045	26.629	0.953	0.046
TiNeuVox-B	24.819	0.947	0.101	23.621	0.932	0.075	22.967	0.925	0.064	24.219	0.927	0.070
DG-Mesh (Ours)	**29.145**	**0.969**	**0.099**	**31.769**	**0.977**	**0.045**	**30.400**	**0.968**	**0.055**	**28.951**	**0.959**	**0.065**

Dynamic Gaussian Mesh

- Experiments
  - Ablation Study
    - Gaussian-Mesh Anchoring: 적용 시 CD 0.685 → 0.516, PSNRm 31.28 → 32.73
      - 균일한 Gaussian 분포가 DPSR의 가정을 만족시켜 Geometry 품질에 핵심적

Method	Gaussian-Mesh Anchoring		
	CD ↓	EMD ↓	PSNR$_m$ ↑
w/o Anchoring	0.685	0.146	31.283
w/ Anchoring	**0.516**	**0.128**	**32.734**

Dynamic Gaussian Mesh

- Experiments
  - Limitation
    - 여전히 전경에 해당하는 object에 대해서 수행한다.