SLAM with 3DGS

1. SplaTAM:Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM [CVPR 2024]

2. Gaussian Splatting SLAM [CVPR 2024 (Highlight)]

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

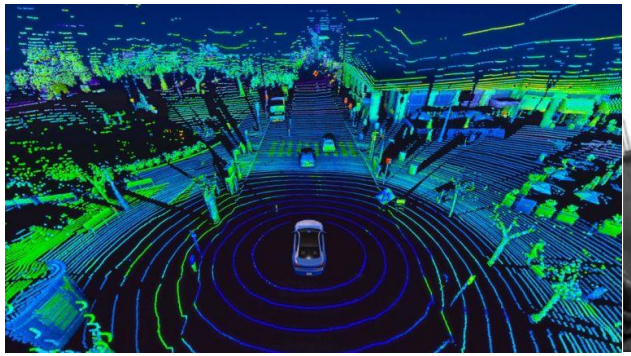
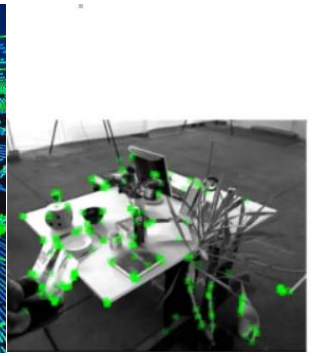
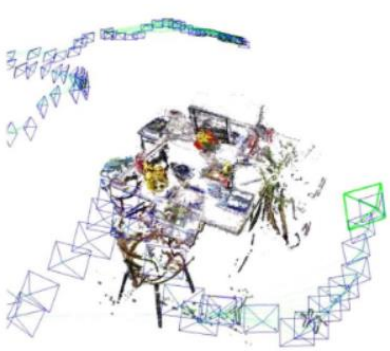
Presented By
120240320 신은호

Contents

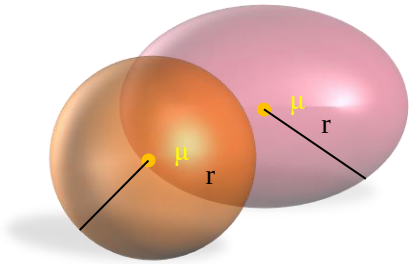
- Background
  - SLAM
  - 3DGS
- SplaTAM [CVPR 2024]
- Gaussian Splatting SLAM [CVPR 2024 (Highlight)]

Background

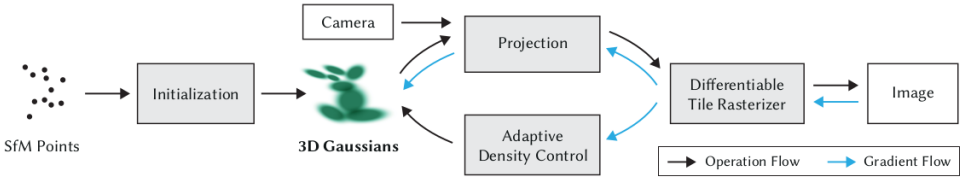
- SLAM(Simultaneous Localization and Mapping):동시적 위치추정 및 지도작성
  - 목적
    - SLAM 알고리즘이 탑재된 로봇은 바퀴 회전 수, 카메라 및 기타 영상 센서에서 가져온 데이터를 사용하여 필요한 움직임의 양을 파악
    - 실시간으로 컴퓨터가 공간에 대한 정보를 이해하고 사용할 수 있도록 가공
    - 이동 로봇의 길 찾기, 자율 주행 차량의 빈 공간 주차, 미지의 환경에서의 드론 조종, AR/VR contents 제작 등 활용
  - 연구 필요성
    - 기존 Hand-crafted, Rule base 방식의 문제 상황 직면
      - 밝기 변화와 Dynamic/Poorly 텍스쳐 환경 문제로 Deep Learning 기반 방식 대두

Background



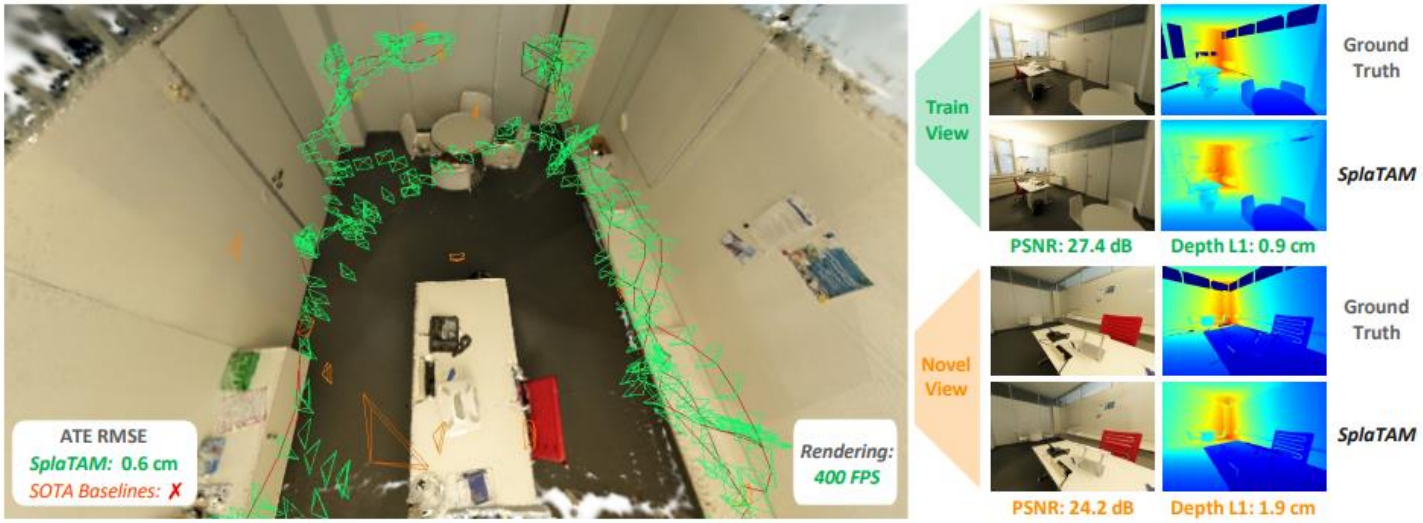
- 3DGS(3D Gaussian Splatting [SIGGRAPH 2023])
  - 목적
    - 각 point cloud를 중심으로 하는 3D Gaussians를 α-blending 하면 image 생성 가능
    - Multiview를 통해 Gaussians의 parameter(μ, ∑, scale)를 update
  - 주요 수식
    - $\sum = RSS^T R^T, \sum' = JW\sum W^T J^T$
      - Projective Transformation의 Jacobian(편미분) J , World2Camera W
      - $\sum'$는 $\sum$의 2D projection
    - $C(p) = \sum_{i=1}^{n} c_i f_i(p) \prod_{j=1}^{i-1}(1 - f_i(p))$, $f = \exp(-\frac{1}{2}(x)^T \sum^{-1}(x))$
  - 결과
    - 빠른 학습속도 및 Rendering 속도
    - 명시적인 공간 표현으로 확장 가능



SplaTAM

- Introduction
  - 3D Gaussian 기반 공간 표현 방식 이용 첫 SLAM 연구
    - 3DGS의 높은 품질의 reconstruction과 real-time rendering 성능 이용
    - 단일 unposed monocular RGB-D camera 이용 3D Gaussian 장면 구성 가능
    - Radiance field 기반 표현 방식(implicit representation) 한계점과의 비교
      - 계산 복잡성, 어려운 edit, 기하적 표현 명시성 부족, 'catastrophic forgetting' 발생

        SLAM의 후반 frame 학습 과정중 전반 frame 잊음 및 혼동

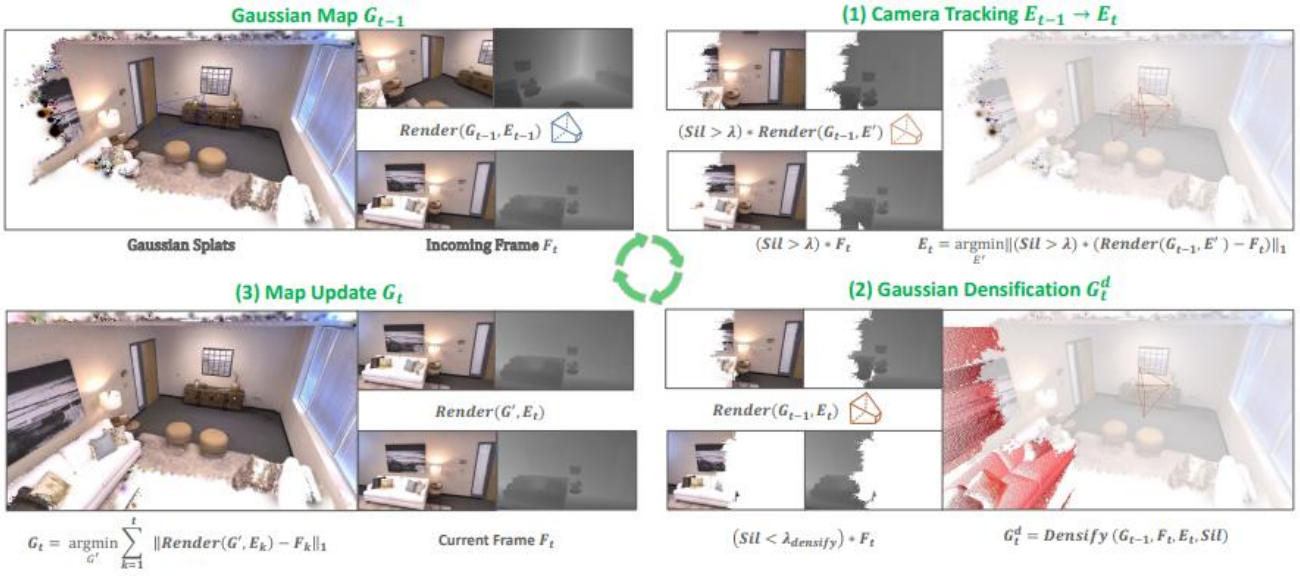


SplaTAM

- Main Contribution
  - Fast rendering and rich optimization
    - 400FPS 속도의 rendering 가능
    - Multiview 대상 학습 방식이었던 기존 GS와 다른 SLAM 고유의 학습법
    - Dense photometric loss 수행 가능
  - Maps with explicit spatial extent
    - Gaussian을 추가함으로써 이전에 관측되었던 장면 수정 손쉽게 가능
    - 새로운 image frame이 제공될 경우, silhouette rendering 통해 new content 탐색 가능
      - Implicit의 경우 global changes 발생으로 unmapped 영역 지속적 변경
  - Explicit map
    - Map의 수용 능력을 Gaussian을 추가함으로써 flexible하게 조정 가능
    - 장면의 특정 영역 edit 가능

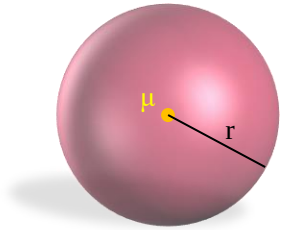
SplaTAM

- Main Contribution
  - 동작 구조
    - Top-Left : 해당 frame RGB-D 입력에 대하여 3D Gaussian map 표현
    - Top-Right : 새로운 입력 이미지에 대한 카메라 포즈를 추정
    - Bottom-Right : silhouette과 input depth 기반으로 새로운 Gaussians를 추가함으로써 Map capacity 확장
    - Bottom-Left : Differentiable rendering을 통해 Gaussian map update



SplaTAM

- Main Contribution
  - Gaussian Map representation
    - 각 Gaussian은 8개 parameter로 구성
      - RGB c 3개, center position $\mu \in R^3$, 반지름 r, opacity $o \in [0,1]$
    - 3차원의 점 x에 대한 $opacity$: $f(x) = o \exp\left(-\frac{\|x-\mu\|}{2r^2}\right)$
    - $C(p) = \sum_{i=1}^{n} c_i f_i(p) \prod_{j=1}^{i-1}(1 - f_i(p))$, $\mu^{2D} = K\frac{E_t\mu}{d}, r^{2D} = \frac{fr}{d}, where\ \ d = (E_t\mu)_z.$
      - K : known camera intrinsic matrix
      - E_t: extrinsic matrix (rotation & translation) at frame t
      - f : focal length, d : camera coordinate에서의 depth
    - $D(p) = \sum_{i=1}^{n} d_i f_i(p) \prod_{j=1}^{i-1}(1 - f_i(p))$, $S(p) = \sum_{i=1}^{n} f_i(p) \prod_{j=1}^{i-1}(1 - f_i(p))$
      - Depth rendering & Silhouette rendering



SplaTAM

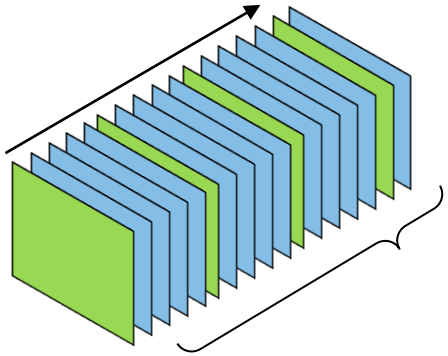
- Main Contribution
  - Camera Tracking
    - 현재 입력된 RGB-D image를 통해 camera pose 추정
    - $E_{t+1} = E_t + (E_t - E_t)$
      - 새로운 timestep의 pose는 이전 이동 방향을 유지한다고 가정
    - RGB rendering을 통한 Gradient-based 최적화
    - $L_t = \sum_p (S(p) > 0.99)(L_1(D(p)) + 0.5L_1(C(p))$
      - 앞서 계산된 Silhouette 기반 minimize Loss
      - Depth와 color render에 대한 L1 loss 수행
      - S(p)>0.99를 통해 아직 충분히 학습된 영역만을 통해 pose estimation
        - ✓기존 Gaussian으로 비교 가능한 영역만을 한정하므로 최적화 성능 향상

SplaTAM

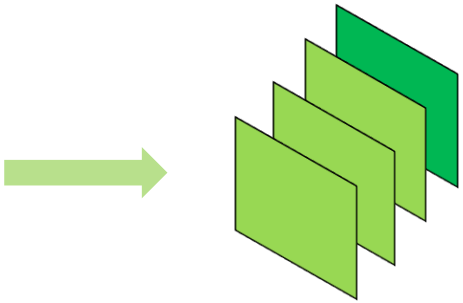
- Main Contribution
  - Gaussian Densification
    - Densification Mask 생성 후 해당 영역에 대하여 Gaussian densify 진행
    - 충분히 dense한 영역에 대해서는 Gaussian 추가하지 않음
    - $M(p) = (S(p) < 0.5) + (D_{GT}(p) < D(p))(L_1(D(p)) > 50MDE)$
      - Dense하지 않은 영역 S<0.5
        - ✓기존 Gaussian이 차지하지 않는 영역
      - GT Depth가 예측된 Depth보다 앞에 있으며 median depth error의 50배 이상인 경우
        - ✓Occlusion 혹은 새로운 물체가 발생한 경우
    - $r = \frac{D_{GT}}{f}$
      - First frame initialization과 동일한 방식으로 new Gaussian 생성

SplaTAM

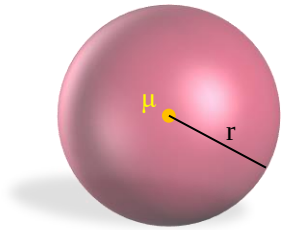
- Main Contribution
  - Gaussian Map Updating
    - 추정된 online 카메라 포즈 집합을 바탕으로 3D Gaussian Map 파라미터 업데이트
    - Pose Estimation과 달리 고정된 pose에 대하여 Gaussian Parameter Update
    - 이미 알려진 posed image에 radiance field를 fitting하는 “classic”한 방식과 2가지 차이점
      - 기존의 Multi-view 대상 3DGS 등
    - 1. 가장 최근에 구축된 Map에서 최적화 시작,
    - 2. 모든 keyframes에 대해 수행하지 않고 새로 추가된 Gaussian에 영향을 미칠 가능성이 있는 프레임 선택 후 Update



각 N번째 frame들을 keyframe으로 선택



현재 frame, 직전 keyframe, 현재 frame과 overlap이 큰 k-2개 keyframe 선택



Parameter Update

SplaTAM

- Datasets
  - Replica
    - 합성 장면 dataset으로 높은 정확도와 완성된 depth map
    - 연속된 frame간 적은 움직임으로 simplest benchmark
  - TUM-RGBD & ScanNet
    - 낮음 품질의 RGB와 Depth image
    - Depth의 경우 missing information 다수 존재
    - Color의 경우 motion blur 다량 존재
  - ScanNet++
    - DSLR로 취득된 장면, dense한 trajectories
    - Color와 depth image의 높은 품질
    - 훈련 과정과 별도로 촬영된 novel hold-out view 존재
    - Replica의 약 30배 정도의 카메라 이동 정도

SplaTAM

- Result & Discussion
  - Quantitative result
    - Camera-Pose Estimation Result

Methods	Avg.	S1	S2
Point-SLAM [30]	343.8	296.7	390.8
ORB-SLAM3 [3]	158.2	156.8	159.7
SplaTAM	**1.2**	**0.6**	**1.9**

ScanNet++ [48]

Methods	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
Vox-Fusion [45]	3.09	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94
NICE-SLAM [53]	1.06	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13
ESLAM [12]	0.63	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63
Point-SLAM [30]	0.52	0.61	0.41	0.37	**0.38**	0.48	0.54	0.69	0.72
SplaTAM	**0.36**	**0.31**	**0.40**	**0.29**	0.47	**0.27**	**0.29**	**0.32**	**0.55**

Replica [35]

Methods	Avg.	fr1/ desk	fr1/ desk2	fr1/ room	fr2/ xyz	fr3/ off.
Kintinuous [41]	4.84	3.70	7.10	7.50	2.90	3.00
ElasticFusion [42]	6.91	2.53	6.83	21.49	1.17	2.52
ORB-SLAM2 [23]	**1.98**	**1.60**	**2.20**	**4.70**	**0.40**	**1.00**
NICE-SLAM [53]	15.87	4.26	4.99	34.49	31.73	3.87
Vox-Fusion [45]	11.31	3.52	6.00	19.53	1.49	26.01
Point-SLAM [30]	8.92	4.34	4.54	30.92	1.31	3.48
SplaTAM	5.48	3.35	6.54	11.13	1.24	5.16

TUM-RGBD [36]

Methods	Avg.	0000	0059	0106	0169	0181	0207
Vox-Fusion [45]	26.90	68.84	24.18	8.41	27.28	23.30	9.41
NICE-SLAM [53]	**10.70**	12.00	14.00	**7.90**	**10.90**	13.40	**6.20**
Point-SLAM [30]	12.19	**10.24**	**7.81**	8.65	22.16	14.77	9.54
SplaTAM	11.88	12.83	10.10	17.72	12.08	**11.10**	7.46

Orig-ScanNet [5]

SplaTAM

- Result & Discussion
  - Quantitative result
    - Train View Rendering Performance on Replica

Methods	Metrics	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
Vox-Fusion [45]	PSNR ↑	24.41	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21
	SSIM ↑	0.80	0.68	0.75	0.80	0.86	0.88	0.79	0.80	0.85
	LPIPS ↓	0.24	0.30	0.27	0.23	0.24	0.18	0.24	0.21	0.20
NICE-SLAM [53]	PSNR ↑	24.42	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94
	SSIM ↑	0.81	0.69	0.76	0.81	0.87	0.89	0.80	0.80	0.86
	LPIPS ↓	0.23	0.33	0.27	0.21	0.23	0.18	0.24	0.21	0.20
Point-SLAM [30]	PSNR ↑	**35.17**	32.40	**34.08**	**35.50**	**38.26**	39.16	**33.99**	**33.48**	**33.49**
	SSIM ↑	**0.98**	0.97	**0.98**	**0.98**	**0.98**	**0.99**	0.96	**0.96**	**0.98**
	LPIPS ↓	0.12	0.11	0.12	0.11	0.10	0.12	0.16	0.13	**0.14**
SplaTAM	PSNR ↑	34.11	**32.86**	33.89	35.25	**38.26**	**39.17**	31.97	29.70	31.81
	SSIM ↑	0.97	**0.98**	0.97	**0.98**	**0.98**	0.98	**0.97**	0.95	0.95
	LPIPS ↓	**0.10**	**0.07**	**0.10**	**0.08**	**0.09**	**0.09**	**0.10**	**0.12**	0.15

    - Novel & Train View Rendering on ScanNet++

Methods	Metrics	Novel View			Training View		
		Avg.	S1	S2	Avg.	S1	S2
Point-SLAM [30]	PSNR [dB] ↑	11.91	12.10	11.73	14.46	14.62	14.30
	SSIM ↑	0.28	0.31	0.26	0.38	0.35	0.41
	LPIPS ↓	0.68	0.62	0.74	0.65	0.68	0.62
	Depth L1 [cm] ↓	✗	✗	✗	✗	✗	✗
SplaTAM	PSNR [dB] ↑	**24.41**	**23.99**	**24.84**	**27.98**	**27.82**	**28.14**
	SSIM ↑	**0.88**	**0.88**	**0.87**	**0.94**	**0.94**	**0.94**
	LPIPS ↓	**0.24**	**0.21**	**0.26**	**0.12**	**0.12**	**0.13**
	Depth L1 [cm] ↓	**2.07**	**1.91**	**2.23**	**1.28**	**0.93**	**1.64**

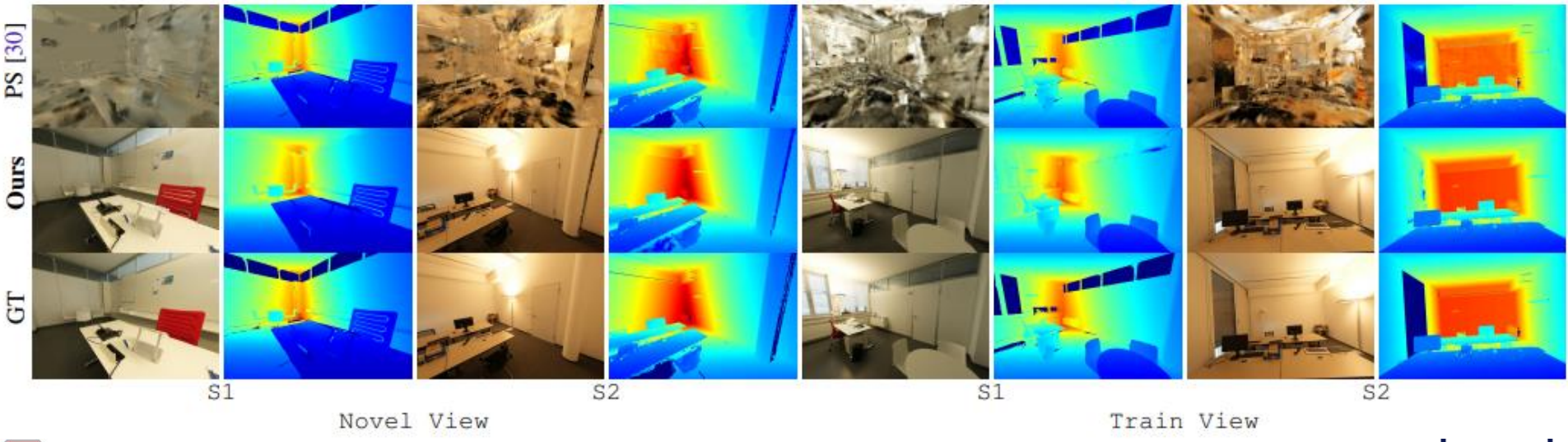
Rendering performance가 입력으로 사용된 training view에서 평가되기 때문에 overfitting 의심 가능

따라서, novel view rendering 평가 수행

SplaTAM

- Result & Discussion
  - Qualitative result
    - PS(Point-SLAM)의 경우 GT depth를 이용하여 rendering
    - GT depth를 이용하였음에도 불구하고, Tracking자체를 실패 하여, Novel view Rendering 실패

Methods	Avg.	S1	S2
Point-SLAM [30]	343.8	296.7	390.8
ORB-SLAM3 [3]	158.2	156.8	159.7
SplaTAM	**1.2**	**0.6**	**1.9**



SplaTAM

- Result & Discussion
  - (본문)Limitations & Future Work
    - Motion blur, large depth noise, aggressive rotation에 민감한 성능 변화 확인
    - 시간을 고려한 모델을 통해 해당 문제 해결 완화 기대
    - 대규모 공간으로의 연구 확장
    - Camera의 intrinsic 설정 값과 dense depth 가 SLAM 동작을 위해 필요
  - 추가 Limitations
    - Rendering 속도가 빠른 것으로 설명했으나, 상대적으로 느린 학습 속도
    - 타 GS 기반 SLAM과 비교하였을 경우 상대적으로 낮은 성능

Gaussian Splatting SLAM

- Introduction
  - 3D Gaussian Splatting in Monocular SLAM
    - 3 fps로 실시간 수행
    - 3D representation을 위해 Gaussian만을 이용
      - Accurate, efficient tracking, mapping, and high-quality rendering
    - 외부 깊이 센서가 있는 경우 RGB-D SLAM으로 확장 가능
  - 3D Gaussian Splatting 한계 해결
    - Structured from Motion을 통해 시스템의 정확한 포즈를 요구하는 기존 알고리즘 해결
      - 직접 최적화를 사용하는 3DGS를 위한 카메라 추적 공식화
    - Explicit한 Gaussian의 특징을 활용하여 기하학적 검증 및 정규화 도입
    - 작은 물체와 투명한 물체의 재구성에서도 sota 성능 달성

Gaussian Splatting SLAM

- Main Contribution
  - Gaussian Splatting
    - $C(p) = \sum_{i=1}^{n} c_i f_i(p) \prod_{j=1}^{i-1}(1 - f_i(p))$
      - N개의 Gaussians를 Blending 하여 pixel color C 출력
        - ✓SH(Spherical Harmonics)아닌 RGB color 이용
    - $\mu_i = \pi(T_{CW} \cdot \mu_W),\ \sum_I = JW\sum_W W^T J^T$
      - 3D Gaussian $N(\mu_W, \sum_W)$ in world coordinates
        2D Gaussian $N(\mu_I, \sum_I)$ in image plane
      - π : projection operation, $T_{CW} \in SE(3)$: camera pose of the viewpoint
    - 위의 공식을 통해 3D Gaussians는 미분 가능한 blending 동작 수행 가능
    - First-order gradient descent 통해 Gaussians의 optic & geometric params update

Gaussian Splatting SLAM

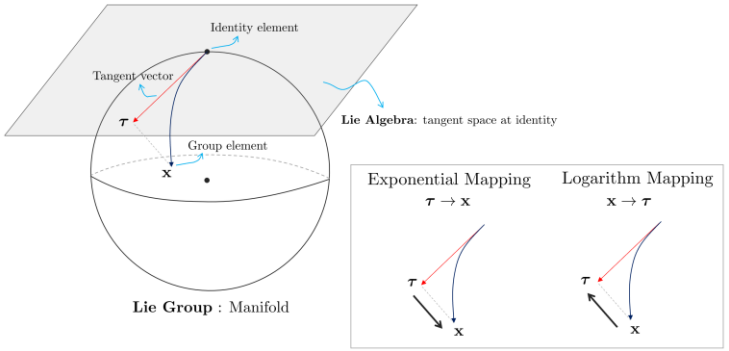
- Main Contribution
  - Camera Pose Optimization
    - For Accurate Tracking,
      - 일반적으로 50 iterations gradient descent per frame 필요
      - 높은 효율의 view synthesis와 gradient computation 동작 중요
    - Parameter calculated explicitly
      - Automatic differentiation의 높은 overhead를 피하기 위해, 3DGS에서 rasterization CUDA를 통해 명시적으로 모든 파라미터 미분 계산
      - 이와 유사하게 camera Jacobians 명시적으로 계산

$$-\frac{\partial \mu_I}{\partial T_{CW}} = \frac{\partial \mu_I}{\partial \mu_C}\frac{D\mu_C}{DT_{CW}}, \frac{\partial \Sigma_I}{\partial T_{CW}} = \frac{\partial \Sigma_I}{\partial J}\frac{\partial J}{\partial \mu_C}\frac{\partial \mu_C}{\partial T_{CW}} + \frac{\partial \Sigma_I}{\partial W}\frac{\partial W}{\partial T_{CW}}$$

      - Camera pose T_{CW}로 미분할 때, chain rule

$$\frac{\mathcal{D}f(\boldsymbol{T})}{\mathcal{D}\boldsymbol{T}} \triangleq \lim_{\tau \to 0} \frac{\mathrm{Log}(f(\mathrm{Exp}(\tau) \circ \boldsymbol{T}) \circ f(\boldsymbol{T})^{-1})}{\tau},$$

$$\frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\boldsymbol{T}_{CW}} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{\mu}_C^{\times} \end{bmatrix}, \frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\boldsymbol{T}_{CW}} = \begin{bmatrix} \mathbf{0} & -\mathbf{W}_{:,1}^{\times} \\ \mathbf{0} & -\mathbf{W}_{:,2}^{\times} \\ \mathbf{0} & -\mathbf{W}_{:,3}^{\times} \end{bmatrix}$$



Gaussian Splatting SLAM

- Main Contribution
  - SLAM(Tracking)
    - Tracking 과정에서 현재 frame camera pose를 optimize

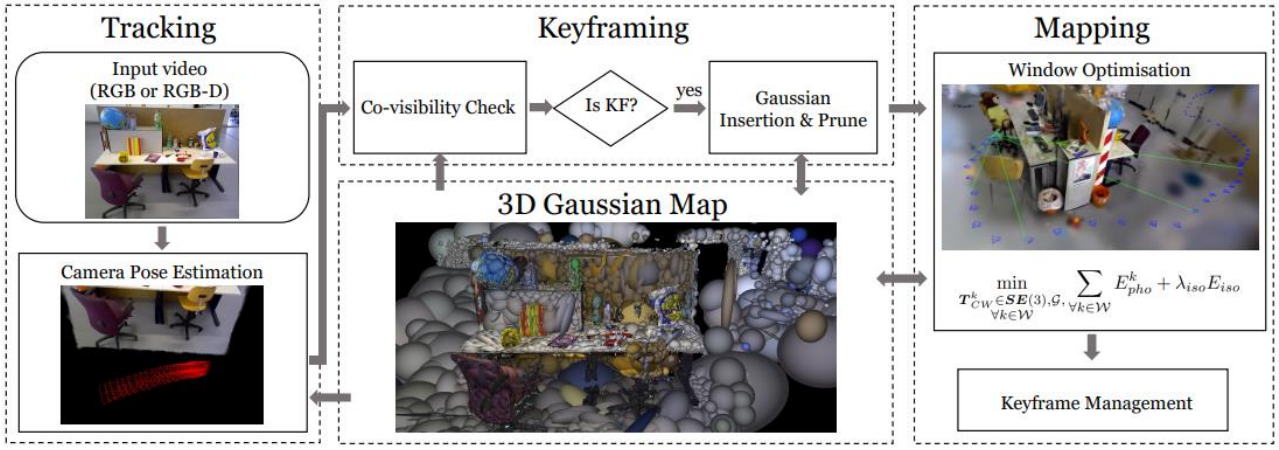
$\therefore E_{pho} = \left\| I(G, T_{CW}) - \bar{I} \right\|_1$

✓$I(G, T_{CW})$는 렌더링된 이미지, $\bar{I}$는 관측된 이미지

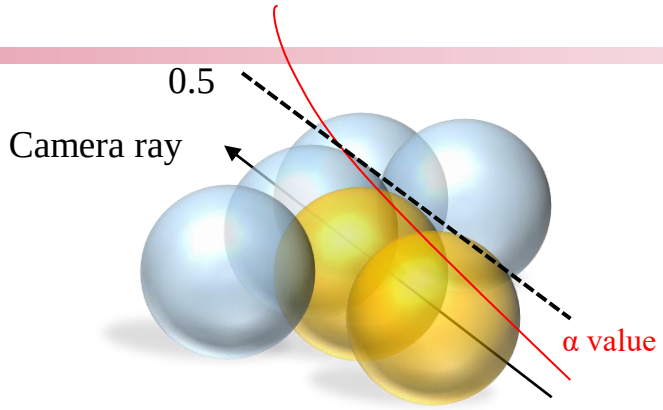
$\therefore E_{geo} = \left\| D(G, T_{CW}) - \bar{D} \right\|_1$

✓Depth 관측 가능한 경우 geometric residual 도입

✓Gaussian initialize를 위해 사용하지 않음



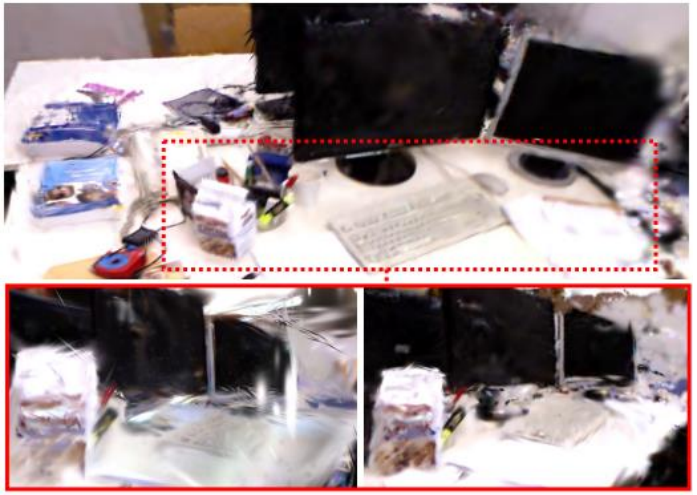
Gaussian Splatting SLAM



- Main Contribution
  - SLAM(Keyframing)
    - Selection and Management
      - 현재 frame i와 이전 keyframe j 간의 관찰된 Gaussians의 교집합 비율을 측정하여 가시성 평가
      - 가시성이 threshold 미만으로 떨어지거나, translation이 일정 크기 이상일 경우 keyframe으로 등록
    - Gaussian Covisibility
      - 3D Gaussian이 camera 위치로 부터 sorting 되기 때문에 visibility 판별 이용
      - 해당 시점으로부터 rasterization을 수행 할 때, α(opacity)값이 0.5 미만인 영역
    - Gasussian Insertion and Pruning
      - 모든 keyframe에 대해 새로운 Gaussian이 삽입되어 새로 보이는 장면을 포착하고 세부사항 정제
      - Monocular 상황에서 Gaussian의 새로운 위치는 부정확할 것이지만 multi-view consistency를 통해 최적화

Gaussian Splatting SLAM

- Main Contribution
  - Mapping
    - Isotropic regularization
      - $E_{iso} = \sum_{i=1}^{|G|} \| s_i - \tilde{s}_i \cdot 1 \|_1$
        - ✓Scaling paramters 타원의 신축에 mean값과의 차이로 penalizing
        - ✓구형성을 장려, 시점에 따른 artifact 억제
      - $\min_{T_{CW}^k \in SE(3), G, k \in W} \sum_{k \in W} E_{pho}^k + \lambda_{iso} E_{iso}$
        - ✓Depth 관측이 가능한 경우 E_{geo}추가



Top : Rendering close to training view
Bottom : Rendering far from training view

Gaussian Splatting SLAM

- Experimental Setup
  - Datasets
    - TUM RGB-D dataset, Replica dataset(Synthetic)
      - Replica의 경우 challenging한 purely rotational camera motions 포함
        - ✓RGB-D evaluation 수행
      - TUM RGB-D 의 경우 RGB, RGB-D evaluation 수행
    - Self-captured real-world sequences by Intel Realsense d455
  - Metrics
    - RMSE of Absolute Trajectory Error(ATE)
    - PSNR, SSIM, LPIPS
      - 매 5th frame 마다 평가

Gaussian Splatting SLAM

- Results
  - Tracking
    - Monocular 설정에서, depth predictor or tracking modules등 deep prior 없이 높은 성능 확인
    - 기본적인 SLAM 표현을 탐구함으로써 성능 개선의 잠재력이 남아있음 확인 가능

Input	Loop-closure	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Monocular	w/o	DSO [5]	22.4	**1.10**	9.50	11.0
		DROID-VO [38]	5.20	10.7	7.30	7.73
		DepthCov-VO [4]	5.60	1.20	68.8	25.2
		Ours	**3.78**	4.60	**3.50**	**3.96**
	w/	DROID-SLAM [38]	**1.80**	**0.50**	2.80	1.70
		ORB-SLAM2 [21]	1.90	0.60	**2.40**	**1.60**
RGB-D	w/o	iMAP [35]	4.90	2.00	5.80	4.23
		NICE-SLAM [48]	4.26	6.19	3.87	4.77
		DI-Fusion [8]	4.40	2.00	5.80	4.07
		Vox-Fusion [45]	3.52	1.49	26.01	10.34
		ESLAM [9]	2.47	**1.11**	2.42	2.00
		Co-SLAM [41]	2.40	1.70	2.40	2.17
		Point-SLAM [29]	4.34	1.31	3.48	3.04
		Ours	**1.50**	1.44	**1.49**	**1.47**
	w/	BAD-SLAM [31]	1.70	1.10	1.70	1.50
		Kintinous [42]	3.70	2.90	3.00	3.20
		ORB-SLAM2 [21]	**1.60**	**0.40**	**1.00**	**1.00**

Camera tracking result on TUM for monocular and RGB-D

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
iMAP [35]	3.12	2.54	2.31	1.69	1.03	3.99	4.05	1.93	2.58
NICE-SLAM	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.07
Vox-Fusion [45]	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
ESLAM [9]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	**0.63**	0.63
Point-SLAM [29]	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72	0.53
Ours	0.44	0.32	0.31	0.44	0.52	**0.23**	0.17	2.25	0.58
Ours (sp)	**0.33**	**0.22**	**0.29**	**0.36**	**0.19**	0.25	**0.12**	0.81	**0.32**

Camera tracking result on Replica for RGB-D
(sp)는 single process 타 Method아 동일 iteration

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	w/o E_{iso}	4.16	4.66	5.73	4.83
	w/o kf selection	13.2	**4.36**	8.65	8.73
	Ours	**3.78**	4.60	**3.50**	**3.96**
RGB-D	w/o E_{geo}	2.39	**0.62**	4.98	2.66
	w/o kf selection	1.64	1.49	2.60	1.90
	Ours	**1.50**	1.44	**1.49**	**1.47**

Ablation Study on TUM RGB-D
isotropic regularization, geometric residual, keyframe selection

Gaussian Splatting SLAM

- Results
  - Novel view rendering
    - 타 방식보다 수백배 빠른 FPS로 rendering 가능
      - Real-time map interaction 가능
    - NeRF 기반 Point-SLAM 과 비교하여 fine detail capture에 장점을 가짐
      - 우측 결과 이미지

Method	PSNR[db]↑	SSIM↑	LPIPS↓	Rendering FPS
NICE-SLAM[48]	24.42	0.809	0.233	0.54
Vox-Fusion[45]	24.41	0.801	0.236	2.17
Point-SLAM [29]	35.17	**0.975**	0.124	1.33
ours	**38.94**	0.968	**0.070**	**769**

Average rendering performance on Replica(RGB-D)

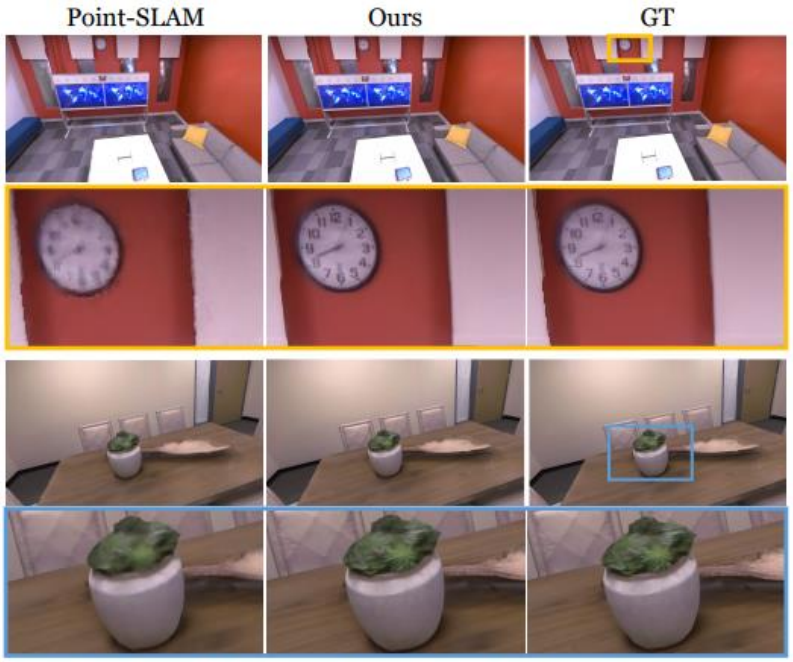
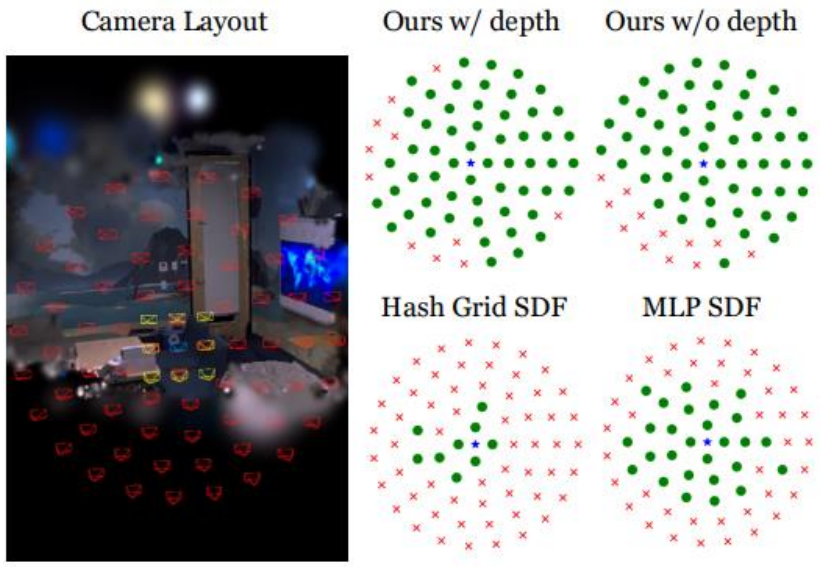


Figure 4. **Rendering examples on Replica.** Point-SLAM struggle with rendering fine details due to the stochastic ray sampling.

Gaussian Splatting SLAM

- Results
  - Convergence basin analysis
    - Camera Localization 능력 평가
      - Co-SLAM(Hash Grid SDF)
      - iMAP(MLP SDF)
        - ✓Co-SLAM 방식에서 loss 추가

Method	seq1	seq2	seq3	Avg.
Neural SDF (Hash Grid)	0.13	0.15	0.16	0.14
Neural SDF (MLP)	0.40	0.38	0.22	0.33
Ours w/o depth	0.82	0.91	**0.65**	0.79
Ours w/ depth	**0.83**	**1.0**	**0.65**	**0.82**



Train views : 노란색 Target view : 파란색 Test views : 빨간색
train 약 0.5m 영역, 0.2m ~ 1.2m 영역 Localization test 수행

- 결과 해석
  - Hash Grid SDF & MLP SDF(Depth 이용)보다 더 좋은 수렴 능력 확인 가능
  - Hashing 및 positional encoding이 signal conflict 가능성 존재
  - Anisotropic Gaussian은 smooth gradient 형성

Gaussian Splatting SLAM

- Conclusion
  - 3D Gaussians를 SLAM 표현 방법으로 사용한 첫 연구
  - Volume rendering을 통해 다양한 object materials를 live SLAM에서 재현
  - Monocular와 RGB-D cases에서 state-of-the-art 달성
- Future research
  - Large-scale scenes을 위한 Loop closure와의 통합
  - Surface normal과 같은 geometry를 Gaussian을 통해 획득하는 방법