End-to-End Learning-based Visual SLAM

2024년도 동계 세미나

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

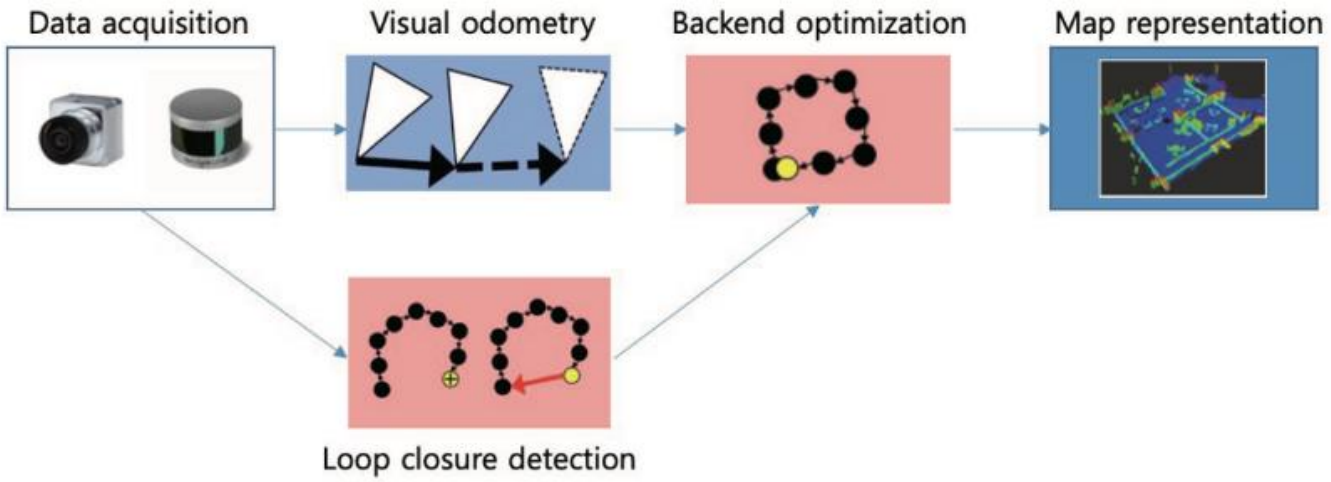
Presented By
안성욱

Outline

- Introduction
  - About Visual-SLAM
- NICE-SLAM: Neural Implicit Scalable Encoding for SLAM
  - CVPR 2022 Oral
- Point-SLAM: Dense Neural Point Cloud-based SLAM
  - ICCV 2023

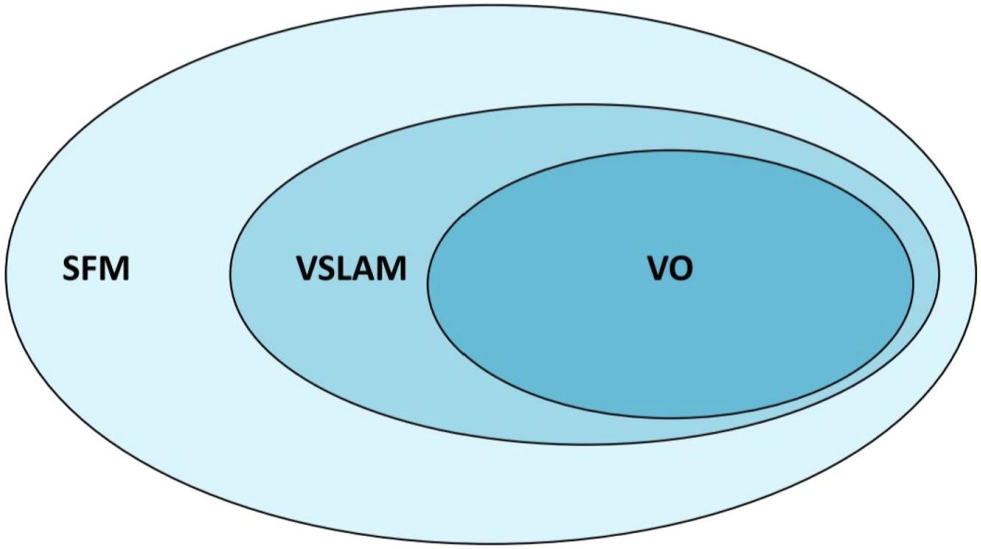
Introduction

- Simultaneous Localization And Mapping
  - 움직이는 물체에 센서를 장착하여 사전 지식 없이 위치 예측, 주변 환경의 지도 작성
- SLAM
  - Pipeline
    - Visual odometry
      - 시작 시점에서부터 실시간으로 자신의 위치 추정
    - Loop closure detection
      - 현재 위치가 이전에 왔던 지점임을 인지
    - Backend optimization
      - Noise 처리



Introduction

- Visual SLAM vs. SfM
  - 공통점
    - 3D mapping과 카메라 pose 추정이 가능함
  - SfM
    - 다수의 이미지로부터 카메라의 pose를 추정하고 3D 지도 작성
    - 이미지의 순서와 실시간성을 고려하지 않음
  - Visual SLAM
    - 순차적으로 들어오는 이미지 sequence로부터 실시간으로 pose 추정과 mapping수행



Introduction

- Learning based Visual SLAM
  - Traditional SLAM
    - BAD-SLAM, ORB-SLAM
    - 오랫동안 SOTA를 유지함
  - Learning-based 방식의 출현
    - PoseNet(ICCV 2015)
      - SLAM의 mapping과 tracking process 중 tracking에 deep learning 적용
        - ✓Deep learning으로 6DOF camera pose를 추정
      - 새로운 환경에서 정확도가 급격히 떨어지는 문제점이 존재 (overfitting)
      - Deep learning이 Visual-SLAM에도 적용될 수 있음을 알림
  - DROID-SLAM(NeurIPS 2021)
    - End-to-end learning-based Visual SLAM의 시작
    - Large scene으로의 확장과 noise 처리에 용이함

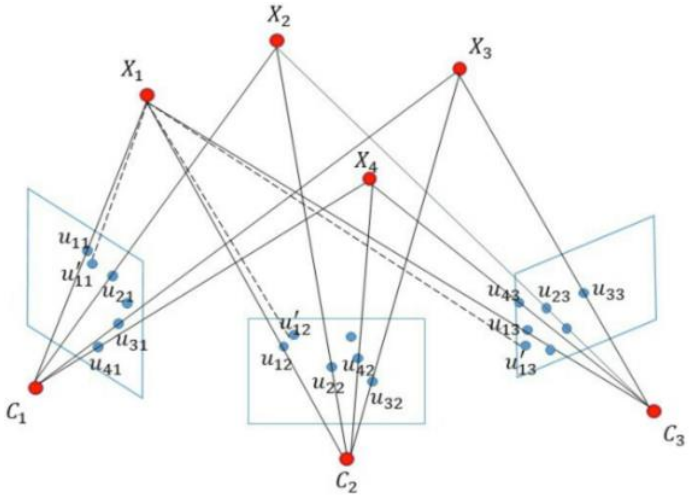
NICE-SLAM: Neural Implicit Scalable Encoding for SLAM

NICE-SLAM[1)]

- Background
  - iMAP (ICCV 2021)
    - 본 논문의 baseline
    - MLP를 real-time SLAM system에 적용
    - 주어진 RGB-D sequence를 이용하여 real-time dense SLAM 수행
    - 전체 scene에 대한 compact한 표현을 위해 single MLP 사용
      - Single MLP의 한계로 인해 detail한 scene geometry와 정확한 camera tracking이 불가능함.
  - Neural implicit representations
    - 데이터에 대한 위치를 입력으로 받아서 해당 점의 값을 반환하는 함수를 학습시키는 딥러닝 기법
      - Point 좌표를 parameterize
    - 최근 여러 task에서 유의미한 결과를 냄
      - Geometry representation, Scene completion, Novel view synthesis

NICE-SLAM[1)]

- Background
  - Local bundle adjustment
    - Bundle adjustment (BA)
      - 다수의 frame에 존재하는 keypoint들의 위치를 기반으로 3D landmark의 위치와 카메라 frame 사이의 motion을 동시에 최적화하는 과정
    - Local BA
      - 이미지의 전체 pixel이 아닌 특정 이미지 영역에 대해서만 BA를 수행함
      - BA에 비해 computational cost가 현저히 감소함



< Bundle adjustment >

NICE-SLAM[1)]

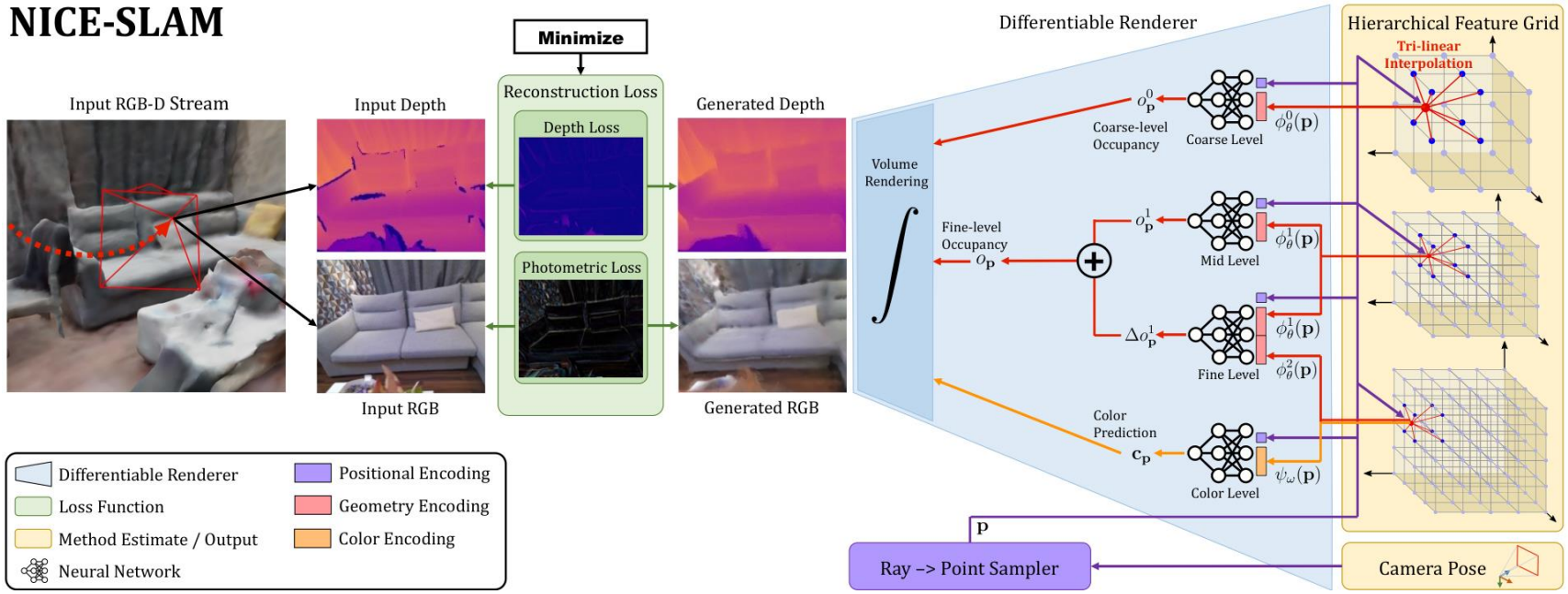
- Background
  - Color representation
    - Scene geometry가 main이지만, camera tracking에 추가 신호 제공을 위해 RGB image를 rendering함
      - Feature grid ψ_ω와 decoder g_ω를 사용

$$\mathbf{c_p} = \mathbf{g}_\omega(\mathbf{p}, \psi_\omega(\mathbf{p}))$$

  - Keyframe selection
    - 첫 번째 frame을 keyframe으로 선정함
    - 이후부터 연속된 frame들 중 유사도가 적은 frame들을 선택하여 저장
      - 모든 frame을 저장하고 유지할 때 보다 메모리를 효율적으로 활용할 수 있음
      - 실시간 정보 처리를 가능하게 함

NICE-SLAM[1)]

- Overview
  - Contribution
    - Large indoor scene에서 detail한 reconstruction을 가능하게 함
      - ‘Hierarchical scene representation’을 통해 multi-level local 정보를 사용
  - Hierarchical scene representation
    - Geometry의 implicit latent embedding을 feature grid에 저장



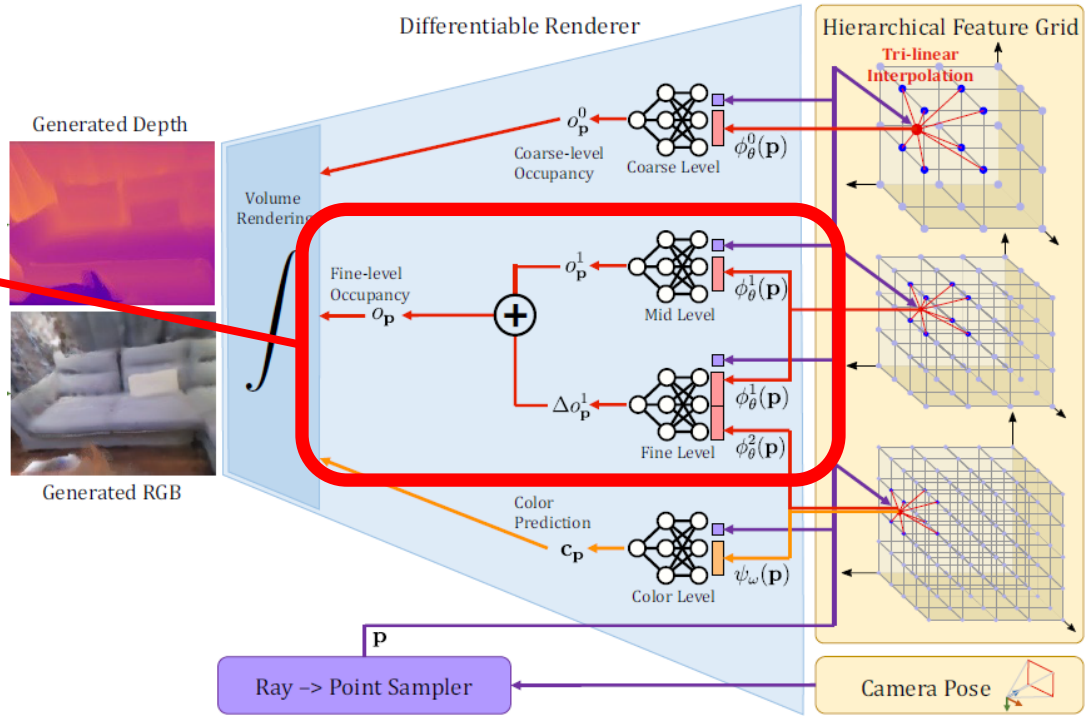
NICE-SLAM[1)]

- Architecture
  - Mid- & Fine-level Geometric Representation
    - Mid-level grid
      - Edge size: 32cm
    - Fine-level grid
      - Edge size: 16cm
    - $\boldsymbol{\phi}_\theta^l$: l번 째 feature grid

$$o_{\mathbf{p}}^1 = f^1(\mathbf{p}, \phi_\theta^1(\mathbf{p}))$$

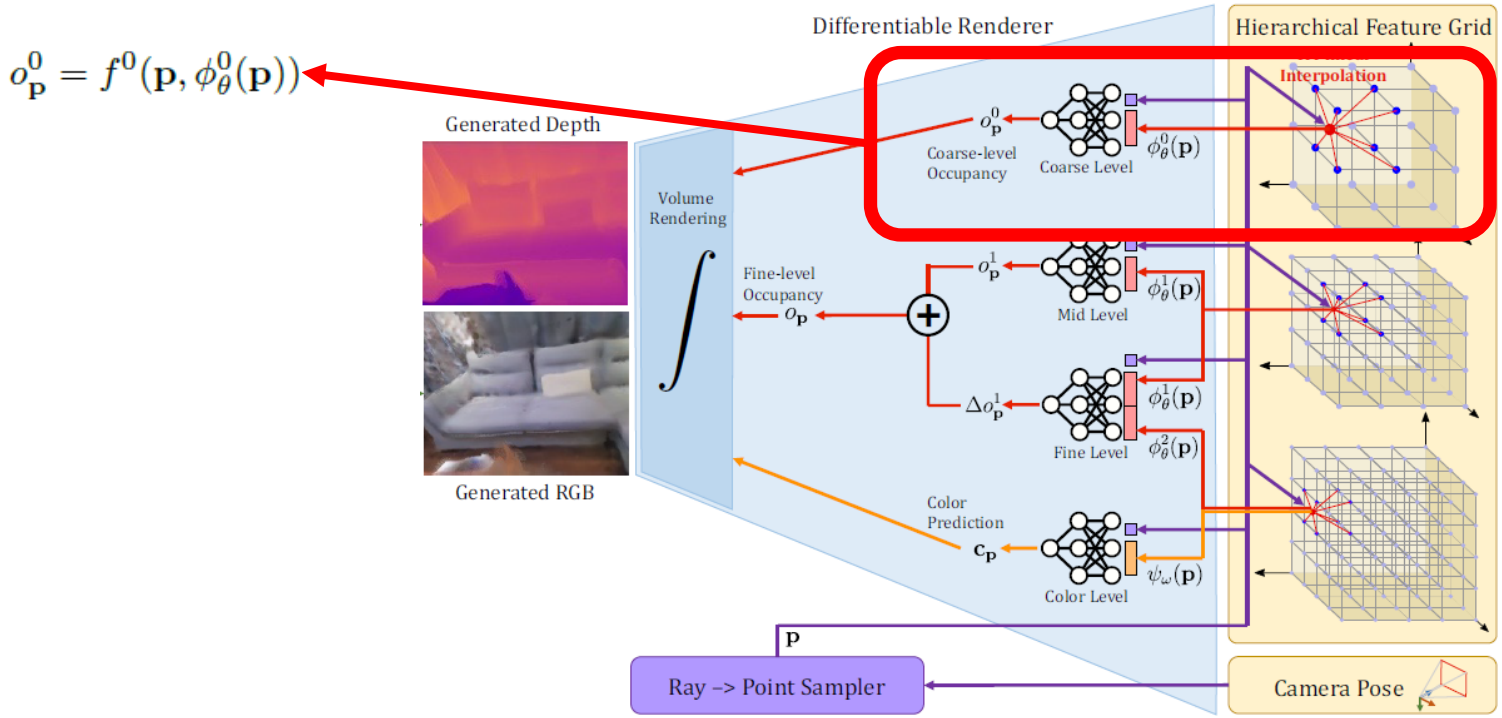
$$\Delta o_{\mathbf{p}}^1 = f^2(\mathbf{p}, \phi_\theta^1(\mathbf{p}), \phi_\theta^2(\mathbf{p}))$$

$$o_{\mathbf{p}} = o_{\mathbf{p}}^1 + \Delta o_{\mathbf{p}}^1$$



NICE-SLAM[1)]

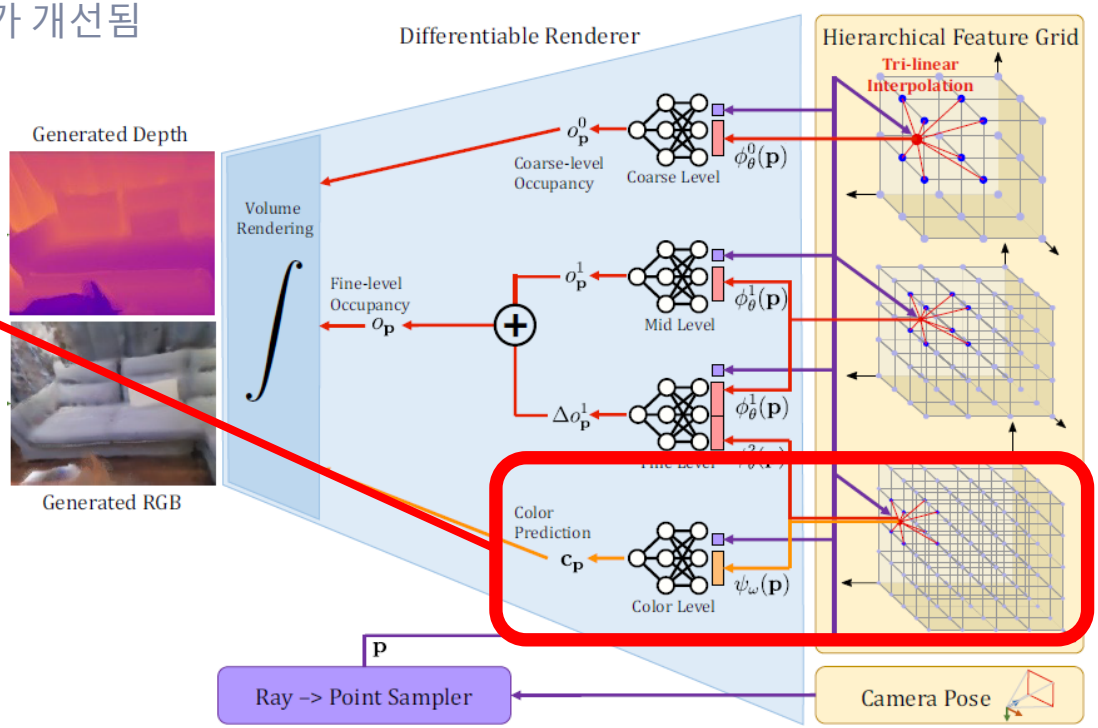
- Architecture
  - Coarse-level Geometric Representation
    - Edge size: 2m
    - High-level geometry(e.g. walls, floors, etc.)와 같은 기하학적인 형상을 포착하기 위함
    - Partially observed area에 대해 occupancy value 예측 가능



NICE-SLAM[1)]

- Architecture
  - Color Representation
    - Fine level grid를 사용함
    - ω는 learnable parameter
    - 강력한 prior knowledge를 가진 geometry와는 다르게 feature grid와 decoder를 함께 최적화함
      - Tracking performance가 개선됨

$$\mathbf{c_p} = \mathbf{g}_\omega(\mathbf{p}, \psi_\omega(\mathbf{p}))$$



NICE-SLAM[1)]

- Architecture
  - Depth and Color Rendering
    - Occupancy 값들과 color prediction 값을 이용하여 weight 생성
    - Coarse level weight

$$w_i^c = o^0_{\mathbf{p}_i} \prod_{j=1}^{i-1} (1 - o^0_{\mathbf{p}_j})$$

    - Fine level weight

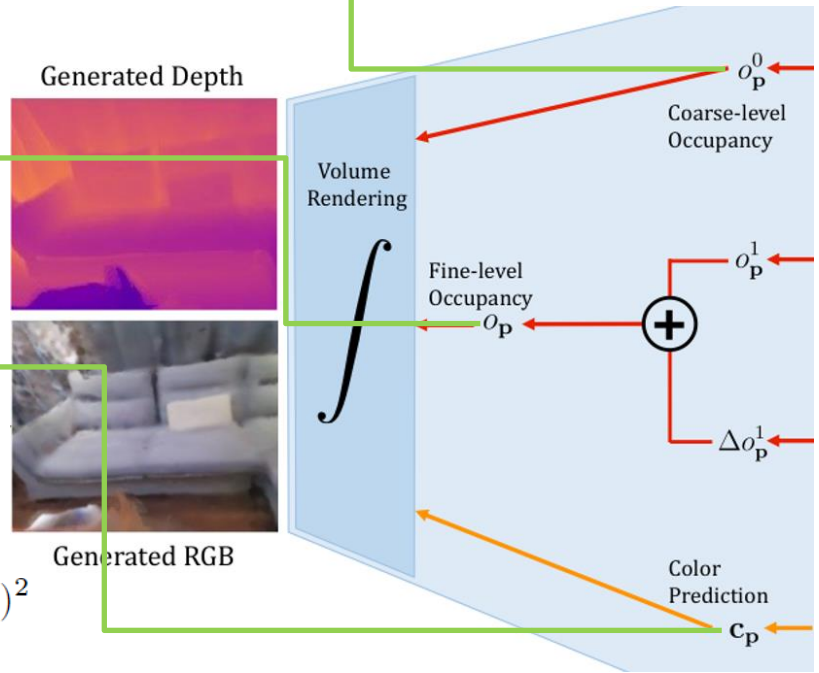
$$w_i^f = o_{\mathbf{p}_i} \prod_{j=1}^{i-1} (1 - o_{\mathbf{p}_j})$$

    - Rendering

$$\hat{D}^c = \sum_{i=1}^{N} w_i^c d_i, \quad \hat{D}^f = \sum_{i=1}^{N} w_i^f d_i, \quad \hat{I} = \sum_{i=1}^{N} w_i^f \mathbf{c}_i$$

    - Ray를 따라 depth variance 값도 계산

$$\hat{D}^c_{var} = \sum_{i=1}^{N} w_i^c (\hat{D}^c - d_i)^2 \quad \hat{D}^f_{var} = \sum_{i=1}^{N} w_i^f (\hat{D}^f - d_i)^2$$



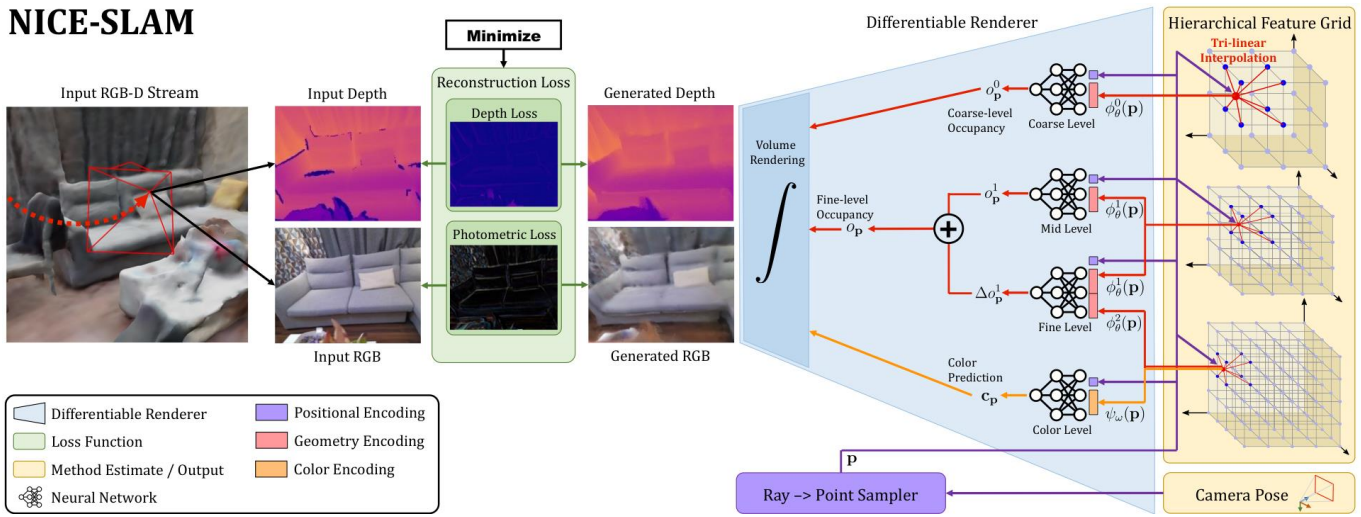
NICE-SLAM[1)]

- Mapping
  - Feature grid에 저장된 scene geometry θ와 appearance ω를 단계적으로 최적화
    - Geometric loss를 이용하여 mid-level feature grid $\boldsymbol{\phi}_\theta^l$ 최적화

$$\mathcal{L}_g^l = \frac{1}{M}\sum_{m=1}^{M}\left|D_m - \hat{D}_m^l\right|, \quad l \in \{c, f\}$$

    - Mid- & fine-level feature $\boldsymbol{\phi}_\theta^1, \boldsymbol{\phi}_\theta^2$를 함께 최적화
    - Local bundle adjustment를 이용해서 K개의 keyframe에 대해 jointly optimize

$$\min_{\theta,\omega,\{\mathbf{R}_i,\mathbf{t}_i\}} \left(\mathcal{L}_g^c + \mathcal{L}_g^f + \lambda_p \mathcal{L}_p\right), \quad \mathcal{L}_p = \frac{1}{M}\sum_{m=1}^{M}\left|I_m - \hat{I}_m\right|$$



NICE-SLAM[1)]

- Camera Tracking
  - Modified geometric loss

$$\mathcal{L}_{g\_var} = \frac{1}{M_t}\sum_{m=1}^{M_t}\frac{\left|D_m - \hat{D}_m^c\right|}{\sqrt{\hat{D}_{var}^c}} + \frac{\left|D_m - \hat{D}_m^f\right|}{\sqrt{\hat{D}_{var}^f}}$$

    - Coarse feature grid가 tracking에도 유의미한 정보를 제공함
  - Camera extrinsic parameter 최적화

$$\min_{\mathbf{R},\mathbf{t}} \left(\mathcal{L}_{g\_var} + \lambda_{pt}\mathcal{L}_p\right), \quad \mathcal{L}_p = \frac{1}{M}\sum_{m=1}^{M}\left|I_m - \hat{I}_m\right|$$

  - Robustness to Dynamic objects
    - 특정 pixel의 loss가 현재 frame 내 모든 pixel value loss의 $(10 \times median)$보다 큰 경우에는 최적화 도중에 pixel을 제거함

NICE-SLAM[1)]

- Experiments

	FLOPs [$\times 10^3$]↓	Tracking [ms]↓	Mapping [ms]↓
iMAP [47]	443.91	101	448
NICE-SLAM	**104.16**	**47**	**130**

Table 4. **Computation & Runtime.** Our scene representation does not only improve the reconstruction and tracking quality, but is also faster. The runtimes for iMAP are taken from [47].

Figure 3. **Reconstruction Results on the Replica Dataset [45].** iMAP* refers to our iMAP re-implementation.

	fr1/desk	fr2/xyz	fr3/office
iMAP [47]	4.9	2.0	5.8
iMAP* [47]	7.2	2.1	9.0
DI-Fusion [16]	4.4	2.3	15.6
NICE-SLAM	2.7	1.8	3.0
BAD-SLAM [43]	1.7	1.1	1.7
Kintinuous [60]	3.7	2.9	3.0
ORB-SLAM2 [27]	**1.6**	**0.4**	**1.0**

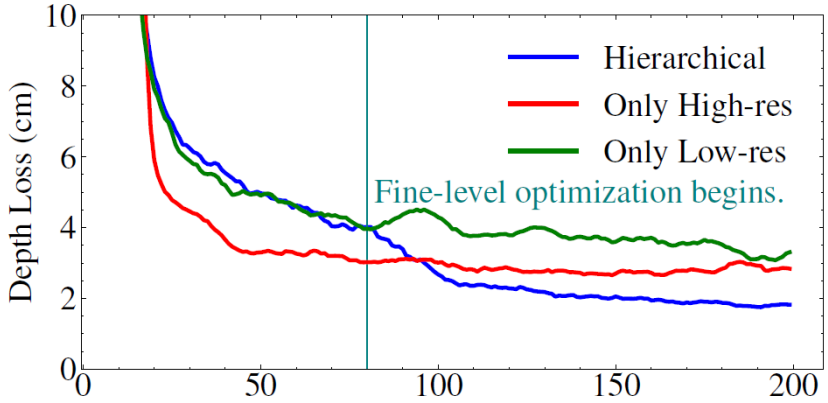
Table 2. **Camera Tracking Results on TUM RGB-D [46].** ATE RMSE [cm] (↓) is used as the evaluation metric.

	TSDF-Fusion [11]	iMAP* [47]	DI-Fusion [16]	NICE-SLAM
Mem. (MB) ↓	67.10	**1.04**	3.78	12.02
Depth L1 ↓	7.57	7.64	23.33	**3.53**
Acc. ↓	**1.60**	6.95	19.40	2.85
Comp. ↓	3.49	5.33	10.19	**3.00**
Comp. Ratio ↑	86.08	66.60	72.96	**89.33**

Table 1. **Reconstruction Results for the Replica Dataset [45] (average over 8 scenes).**

NICE-SLAM[1)]

- Ablation Study
  - Hierarchical Architecture
    - Hierarchical feature grid를 사용했을 때와 사용하지 않았을 때



  - Local BA, Color representation, Keyframe selection
    - 각각을 제외했을 때 ATE RMSE

ATE RMSE (↓)	w/o Local BA	w/o $\mathcal{L}_p$	w/ iMAP keyframes	Full
Mean	37.74	32.02	12.10	9.63
Std.	30.97	21.98	3.38	0.62

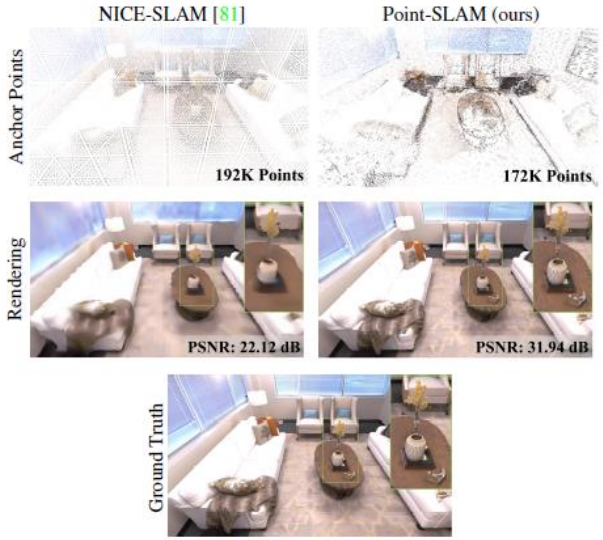
Point-SLAM: Dense Neural Point Cloud-based SLAM

Point-SLAM[1)]

- Background
  - Scene Representations
    - Grid-based representation
      - NICE-SLAM이 대표적인 예시
      - Dense grid, hierarchical octrees, voxel hashing등을 사용함
      - 장점
        - ✓주변 검색과 context 집계가 빠르게 일어남
      - 단점
        - ✓Grid의 resolution이 미리 정해져야 하고, reconstruction 도중에 바꿀 수 없음
    - Point-based representation
      - Point density를 미리 정할 필요가 없으며 scene에 따라 density가 달라짐
      - Free space를 modeling하는데 memory를 낭비하지 않기 위해 surface 주위를 focus
      - 다만 point set에는 연결성 있는 구조가 부족하기 때문에 neighborhood search가 어려움
        - ✓본 논문에서는 adaptive feature point를 사용하여 빠른 neighborhood search 가능

Point-SLAM[1)]

- Overview
  - Contribution
    - 기존 dense neural SLAM 방법들은 scene feature를 sparse grid에 고정하였으나, 본 논문에서는 dynamic point를 사용하여 더 적은 detail로 dense한 공간을 표현함
      - Runtime과 memory usage 방면에서 효과적임
    - Point-based scene representation이 mapping과 tracking에 효과적으로 사용됨을 증명함
  - Dynamic point density
    - Depth를 추정할 때 생기는 noise를 고려하여 anchor point의 위치를 조정함



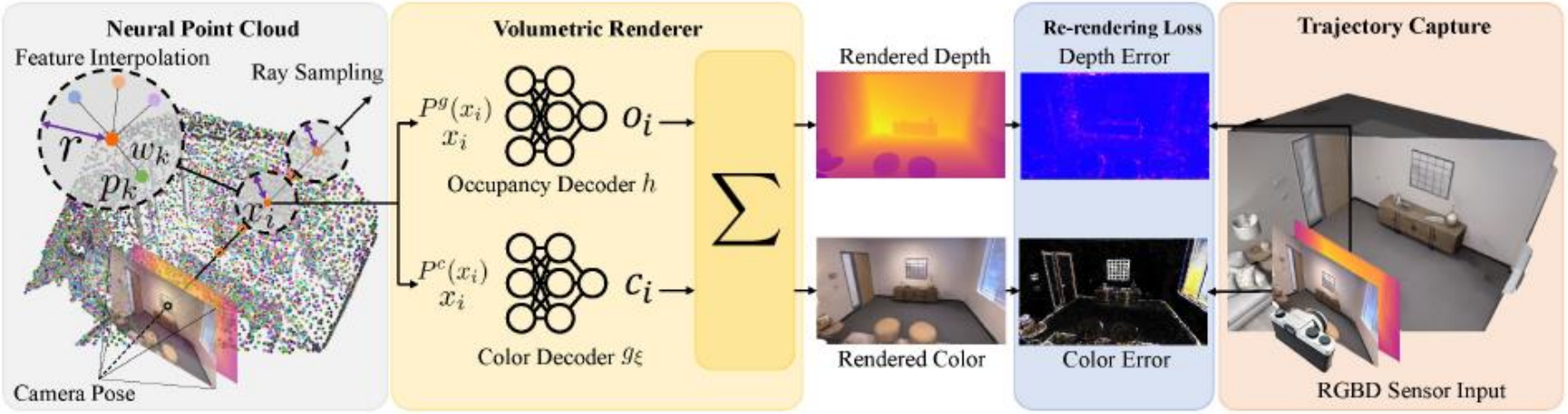
Point-SLAM[1)]

- Architecture
  - Neural point cloud representation
    - Neural point cloud
      - N개의 neural points로 이루어진 하나의 set으로 정의함

$$P = \{(p_i, f_i^g, f_i^c) \mid i = 1, \cdots, N\}$$

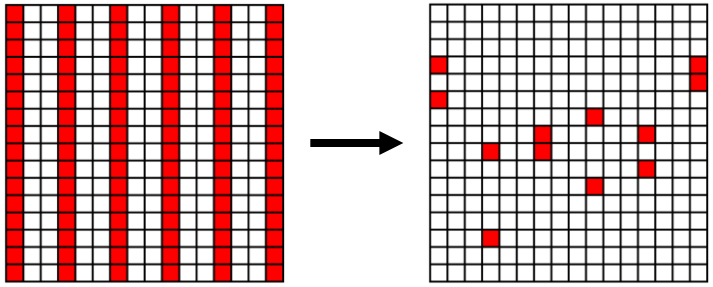
$$p_i \in \mathbb{R}^3, \quad f_i^g \in \mathbb{R}^{32}, \quad f_i^c \in \mathbb{R}^{32}$$

✓각각 location, geometric feature descriptor, color feature descriptor

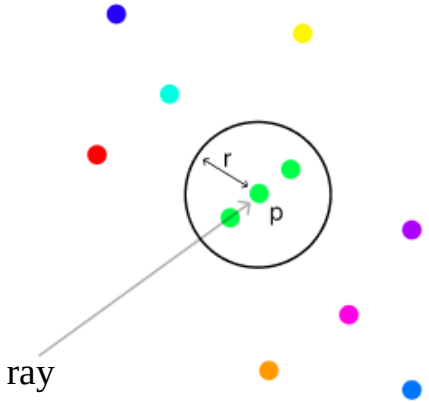


Point-SLAM[1)]

- Architecture
  - Neural point cloud representation
    - Point Adding Strategy
      - Image plane에서 일정 간격으로 X축의 pixel을 sampling하고, 그 중에서 Y축을 따라 가장 높은 color gradient magnitude를 가진 5Y개의 pixel 중 Y개의 pixel을 sampling

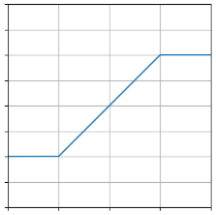


      - Depth information을 사용하여 sampling된 points를 3D로 unprojection
      - 3D point를 중심으로 하는 구의 반지름 r 내의 neighbor를 찾고, 발견되지 않으면 ray를 따라 2개의 neural point를 추가함
        - ✓Depth D 에 대하여 $(1-\rho)D, (1+\rho)D$에 위치한 point
          - $\rho \in (0,1)$, expected depth noise



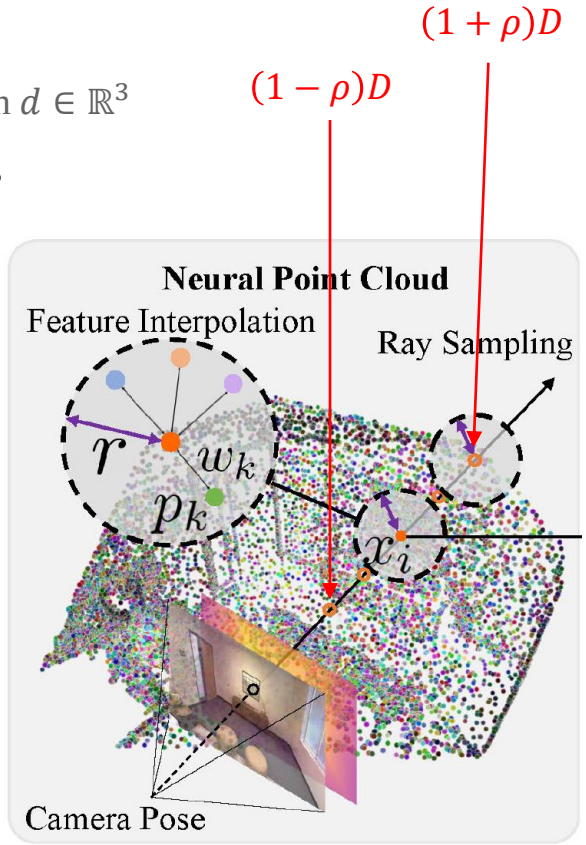
    - Dynamic Resolution
      - Scene 전체에 dynamic point density 적용
      - Color gradient에 기초하여 clamped linear mapping으로 r을 정의함

$$r(u,v) = \begin{cases} r_l & \text{if } \nabla I(u,v) \geq g_u \\ \beta_1 \nabla I(u,v) + \beta_2 & \text{if } g_l \leq \nabla I(u,v) \leq g_u \\ r_u & \text{if } \nabla I(u,v) \leq g_l \end{cases}$$



Point-SLAM[1)]

- Architecture
  - Rendering
    - 앞서 생성된 point cloud를 기반으로 Volume rendering 방식 사용
      - $x_i = \boldsymbol{O} + z_i \boldsymbol{d}, \quad i \in \{1, \dots, M\}$
        - ✓Camera pose의 원점 $\boldsymbol{O}$, point depth $z_i \in \mathbb{R}$, ray direction $d \in \mathbb{R}^3$
    - $(1-\rho)D, (1+\rho)D$ 사이 5개의 point를 같은 간격으로 sampling
    - 반지름 r 내에서 2~8개의 point를 가지고 feature interpolation
      - 2개 미만의 neighbor point를 가진다면 occupancy는 0
  - Geometric feature $\boldsymbol{P^g}$
    - $P^g(x_i) = \sum_k \frac{w_k}{\sum_k w_k} f_k^g \quad \text{with } w_k = \frac{1}{\|p_k - x_i\|^2}$
  - Color feature $\boldsymbol{P^c}$
    - $P^c(x_i) = \sum_k \frac{w_k}{\sum_k w_k} f_{k,x_i}^c, \quad f_{k,x_i}^c = F_\theta(f_k^c, p_k - x_i)$
      - F는 θ로 parameterize된 one-layer MLP

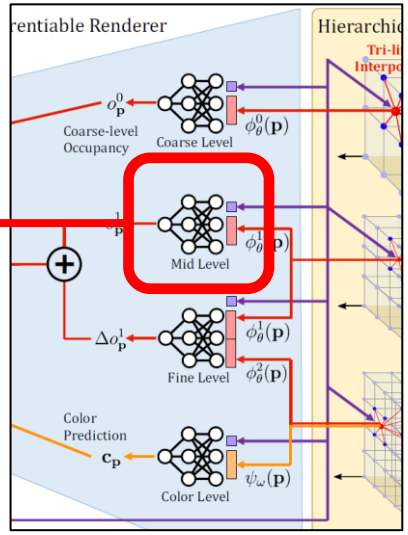


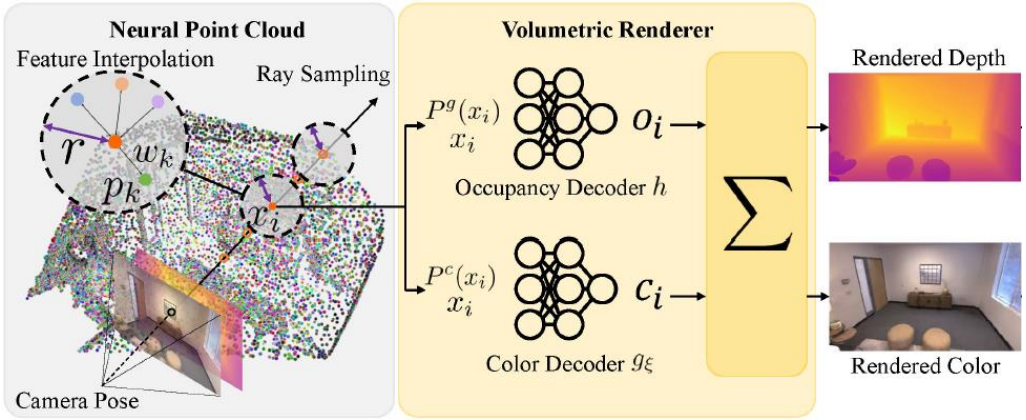
Point-SLAM[1)]

- Architecture
  - Rendering
    - $o_i = h(x_i, P^g(x_i)), \quad c_i = g_\xi(x_i, P^c(x_i)), \quad \alpha_i = o_{p_i} \prod_{j=1}^{i-1}(1 - o_{p_j})$

Positional encoding 적용

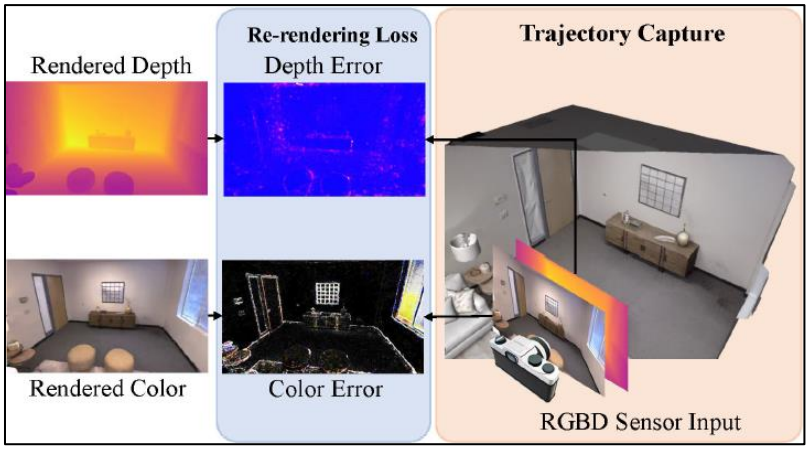
      - h, g_ξ는 각각 geometry decoder와 color decoder를 의미함
        - ✓h는 NICE-SLAM의 middle geometric decoder와 동일
        - ✓ξ는 learnable parameter
    - Depth와 color rendering
      - $\hat{D} = \sum_{i=1}^{N} \alpha_i z_i, \quad \hat{I} = \sum_{i=1}^{N} \alpha_i c_i$
    - Variance
      - $\hat{S}_D = \sum_{i=1}^{N} \alpha_i (\hat{D} - z_i)^2$





Point-SLAM[1)]

- Mapping
  - Loss function
    - $\mathcal{L}_{map} = \sum_{m=1}^{M} |D_m - \hat{D}_m|_1 + \lambda_m |I_m - \hat{I}_m|_1$
      - Geometric L_1 depth loss와 hyperparameter λ_m을 곱한 color L_1 loss 사용
      - Feature descriptor f^g, f^c와 learnable parameter θ, ξ optimize
- Tracking
  - Loss function
    - $\mathcal{L}_{track} = \sum_{m=1}^{M_t} \frac{|D_m - \hat{D}_m|_1}{\sqrt{\hat{S}_D}} + \lambda_t |I_m - \hat{I}_m|_1$
      - Depth loss를 standardization하여 사용
      - 각 frame에서 camera extrinsic $\{\boldsymbol{R}, \boldsymbol{t}\}$ optimize



Point-SLAM[1)]

- Experiments

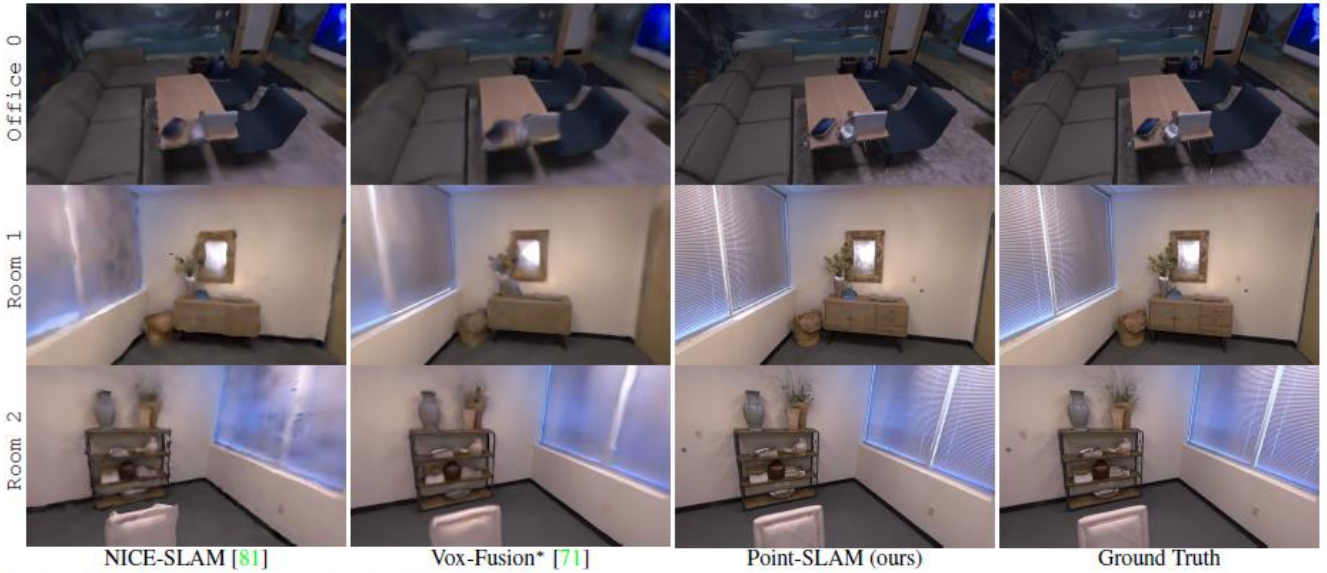


Figure 4: **Rendering Performance on Replica [53].**

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM [81]	Depth L1 [cm] ↓	1.81	1.44	2.04	1.39	1.76	8.33	4.99	2.01	2.97
	Precision [%] ↑	45.86	43.76	44.38	51.40	50.80	38.37	40.85	37.35	44.10
	Recall [%] ↑	44.10	46.12	42.78	48.66	53.08	39.98	39.04	35.77	43.69
	F1 [%] ↑	44.96	44.84	43.56	49.99	51.91	39.16	39.92	36.54	43.86
Vox-Fusion* [71]	Depth L1 [cm] ↓	1.09	1.90	2.21	2.32	3.40	4.19	2.96	1.61	2.46
	Precision [%] ↑	75.83	35.88	63.10	48.51	43.50	54.48	69.11	55.40	55.73
	Recall [%] ↑	64.89	33.07	56.62	44.76	38.44	47.85	60.61	46.79	49.13
	F1 [%] ↑	69.93	34.38	59.67	46.54	40.81	50.95	64.56	50.72	52.20
ESLAM [29]	Depth L1 [cm] ↓	0.97	1.07	1.28	0.86	1.26	1.71	1.43	1.06	1.18
Ours	Depth L1 [cm] ↓	**0.53**	**0.22**	**0.46**	**0.30**	**0.57**	**0.49**	**0.51**	**0.46**	**0.44**
	Precision [%] ↑	**91.95**	**99.04**	**97.89**	**99.00**	**99.37**	**98.05**	**96.61**	**93.98**	**96.99**
	Recall [%] ↑	**82.48**	**86.43**	**84.64**	**89.06**	**84.99**	**81.44**	**81.17**	**78.51**	**83.59**
	F1 [%] ↑	**86.90**	**92.31**	**90.78**	**93.77**	**91.62**	**88.98**	**88.22**	**85.55**	**89.77**

Figure 3: **Reconstruction Performance on Replica [53].**

Method	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM [81]	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.06
Vox-Fusion [71]	0.40	0.54	0.54	0.50	0.46	0.75	0.50	0.60	0.54
Vox-Fusion* [71]	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
ESLAM [29]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	**0.63**	0.63
Point-SLAM (ours)	**0.61**	**0.41**	**0.37**	**0.38**	**0.48**	**0.54**	**0.69**	0.72	**0.52**

Table 1: **Tracking Performance on Replica [53]** (ATE RMSE ↓ [cm]).

first second third

감사합니다