Depth Anything 3: Recovering the Visual Space from Any Views

GSPS
Graduate Student Paper Seminar

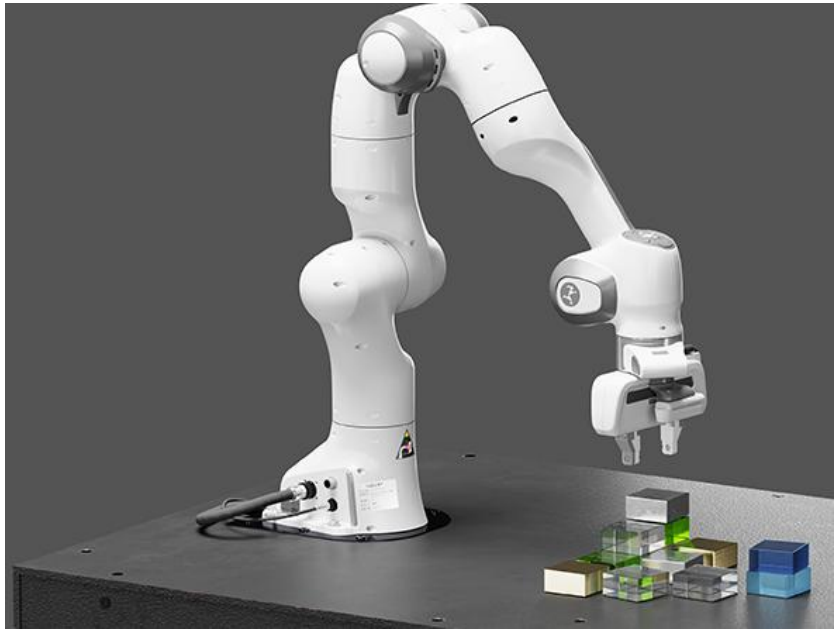
(Haotong Lin∗, Sili Chen∗, Jun Hao Liew∗, Donny Y. Chen∗, Zhenyu Li, Guang Shi, Jiashi Feng, Bingyi Kang∗,†, in ICLR 2026 under review)

Department of Metaverse Convergence at Chung-Ang University
Myungjun Yun

ICLR
International Conference On Learning Representations

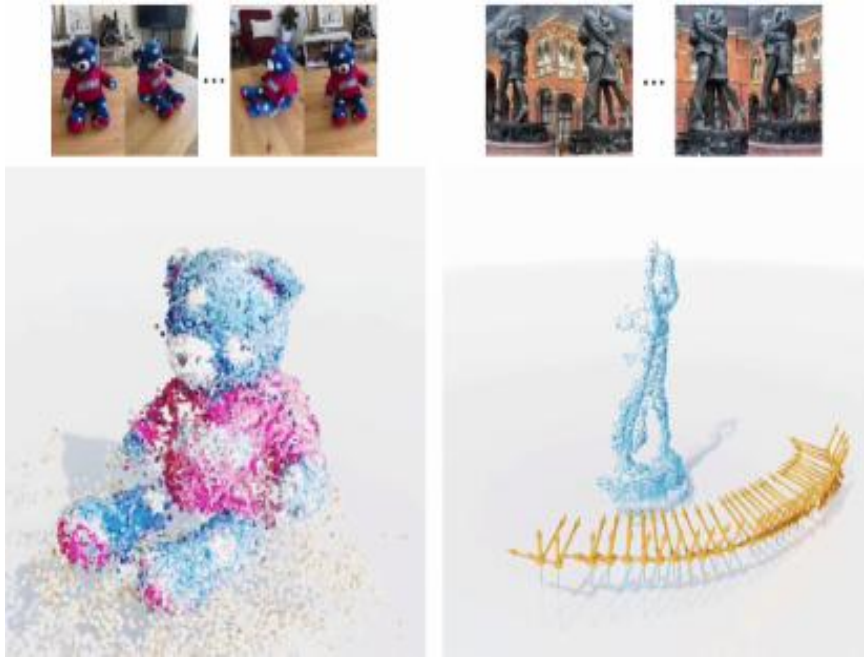
Introduction

- 3D 공간 정보를 visual input으로부터 인지하고 이해하는 능력이 인간의 spatial intelligence의 핵심
- 위 능력은 로봇 공학 및 혼합 현실과 같은 응용 분야에 필수적이며 다양한 3D vision task로 발전함



Introduction

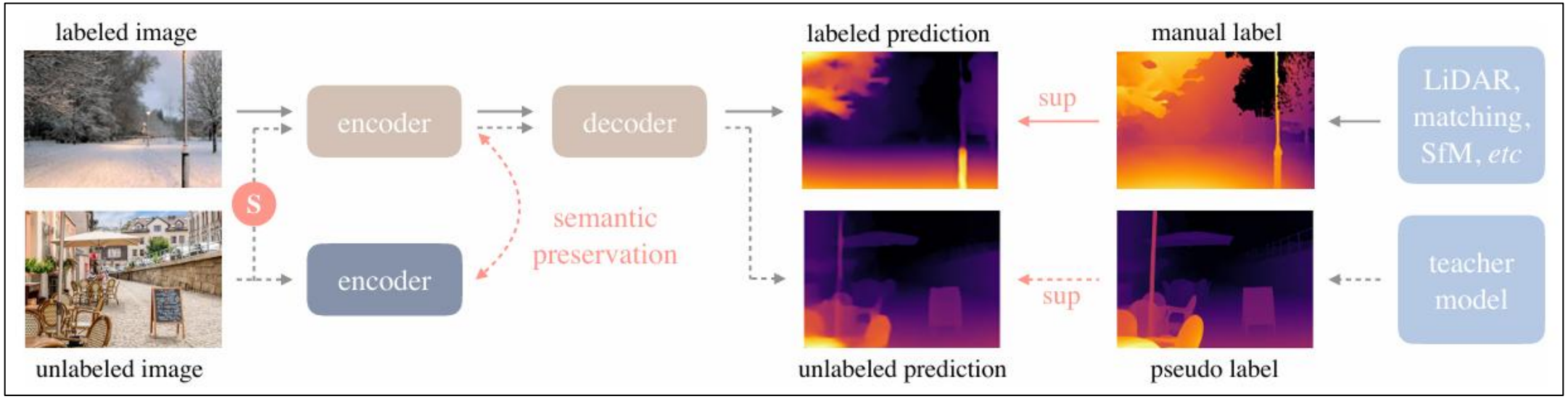
1. Monocular Depth Estimation
    - Single 2D image → Depth map
2. Structure from Motion
    - Multi 2D image → Camera pose, Sparse 3D point cloud



Introduction

Depth Anything V1

- Monocular Depth Estimation foundation model
- 어떤 상황(Lighting condition, noise, unseen)에서도 모든 image에 대해 high-quality depth map을 출력
- Unlabeled Datasets을 구축해서 학습



Introduction

Depth Anything V1

- 기존에 Depth dataset을 구축하는 방법은 sensor, sfm등을 통해 depth data를 얻음

  → **high cost, time-consuming**

- 이 문제를 해결하기 위해 Unlabeled Data 사용

  → **low cost, time-saving, generalization ability**

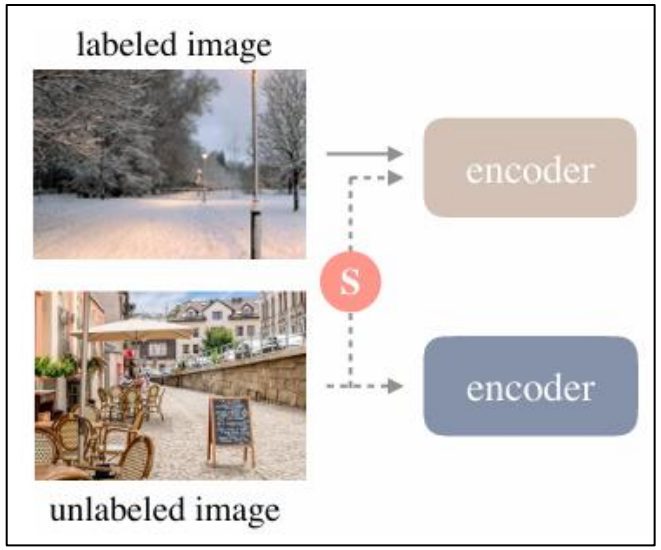
Introduction

Depth Anything V1

- Teacher model은 labeled data로 training
- Student model은 labeled, unlabeled data로 training

  → unlabeled data는 학습된 teacher model로 pseudo label생성
- Student model을 training 할 때 이미 충분한 labeled data가 있어서 unlabeled data에서 얻는 visual knowledge 제한

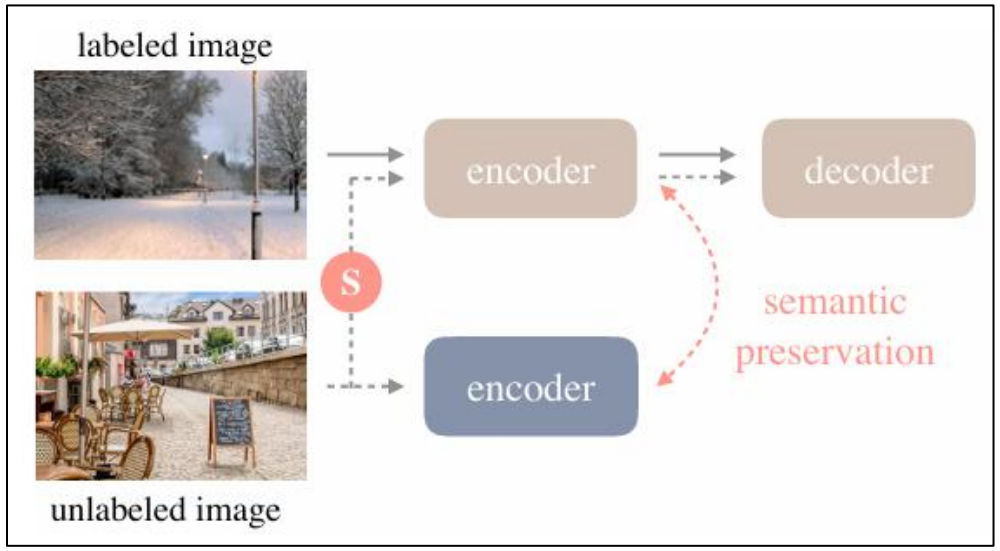
  → unlabeled image에 perturbations(GaussianBlur, ColorJitter) 적용



Introduction

Depth Anything v1

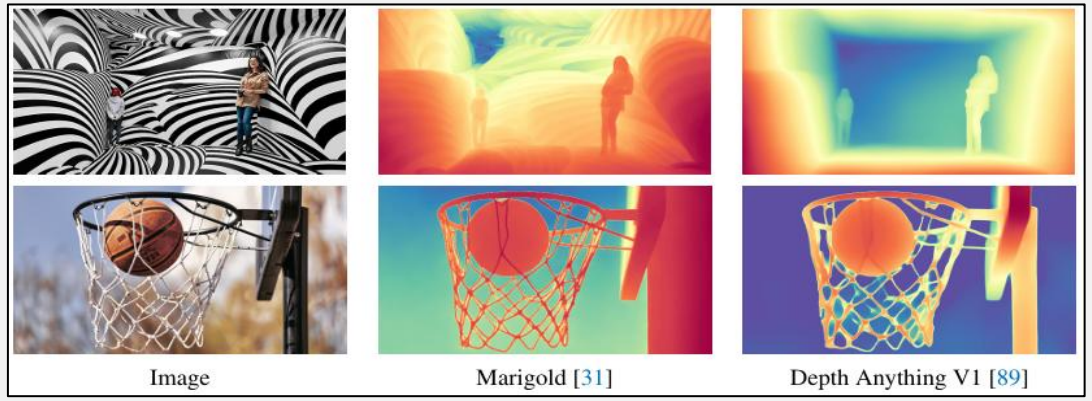
- DINOv2의 semantic prior를 supervision으로 받아 depth 모델의 성능 향상
  → pseudo label의 오차, perturbations로 인한 변환에 robust
- Student model이 supervision을 과도하게 따라가지 않도록 제약을 줌



Introduction

Depth Anything v2

- Model architecture 측면에서 MDE는 두 그룹으로 나눌 수 있음
  1. Discriminative model
     - ➢ 복잡한 장면에 대해서 더 정확
  2. Generative model
     - ➢ Detail에 강점
- 두 그룹의 장점을 결합하고 단점을 해결하는 Model 제안



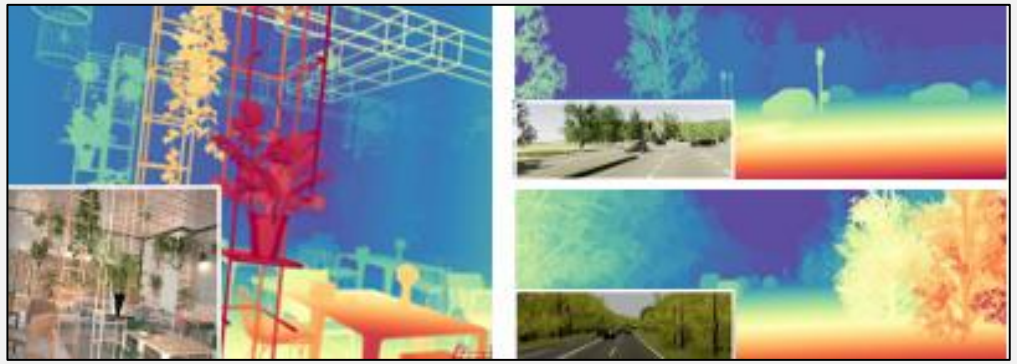
Preferable Properties	Fine Detail	Transparent Objects	Reflections	Complex Scenes	Efficiency	Transferability
Marigold [31]	✓	✓	✓	✗	✗	✗
Depth Anything V1 [89]	✗	✗	✗	✓	✓	✓
Depth Anything V2 (Ours)	✓	✓	✓	✓	✓	✓

Introduction

Depth Anything V2

- MDE의 본질은 discriminative task, Depth Anything V1의 장점을 유지하고 단점을 개선하는 것을 목표로 함
- Technique 보다는 dataset에 집중
- Real-world data → noise, inaccurate labels
- Synthetic data → detail, accurate labels

Real-world data



Synthetic data

Introduction

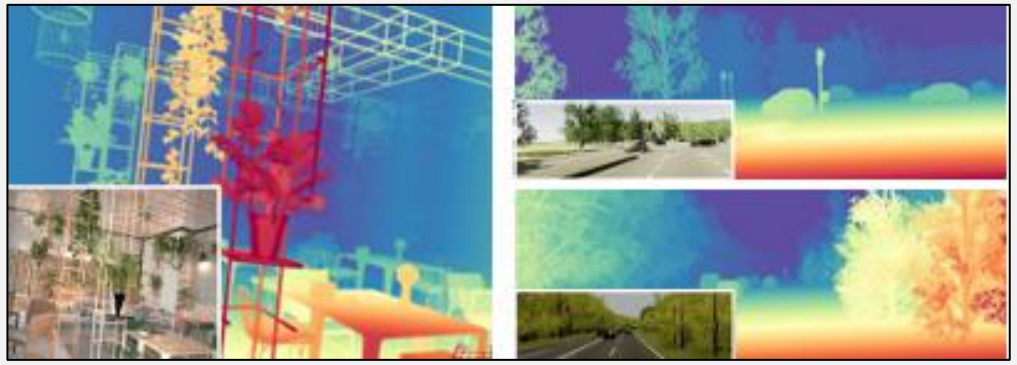
Depth Anything V2

Why not use synthetic data?

- Synthetic 이미지는 색상 및 noise, blur가 없고 clean
- Real-world data는 noise, blur, light condition등 다양한 변수가 있음

  → synthetic data로만 학습하면 real-world data에서는 성능이 안 좋음

Real-world data

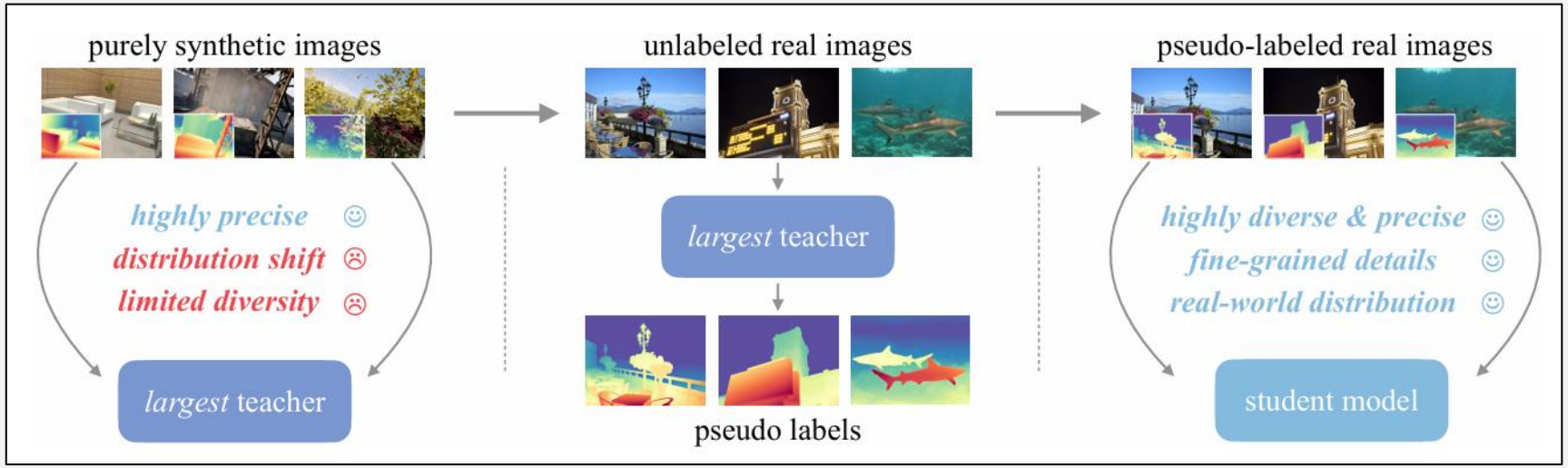


Synthetic data

Introduction

Depth Anything V2

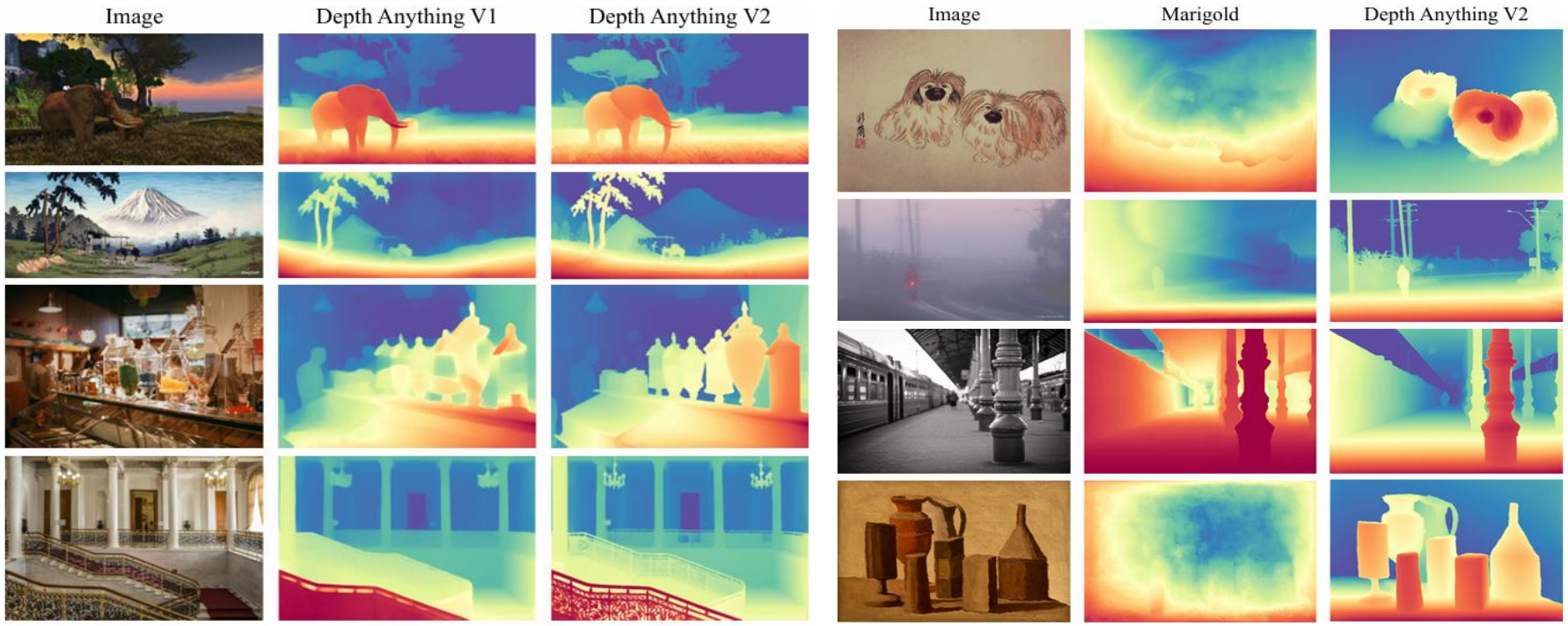
- Teacher model에서 synthetic dataset으로 training
- Unlabeled real image에 pseudo labels 생성
- Pseudo-labeled real image로 student model training



Introduction

Depth Anything V2

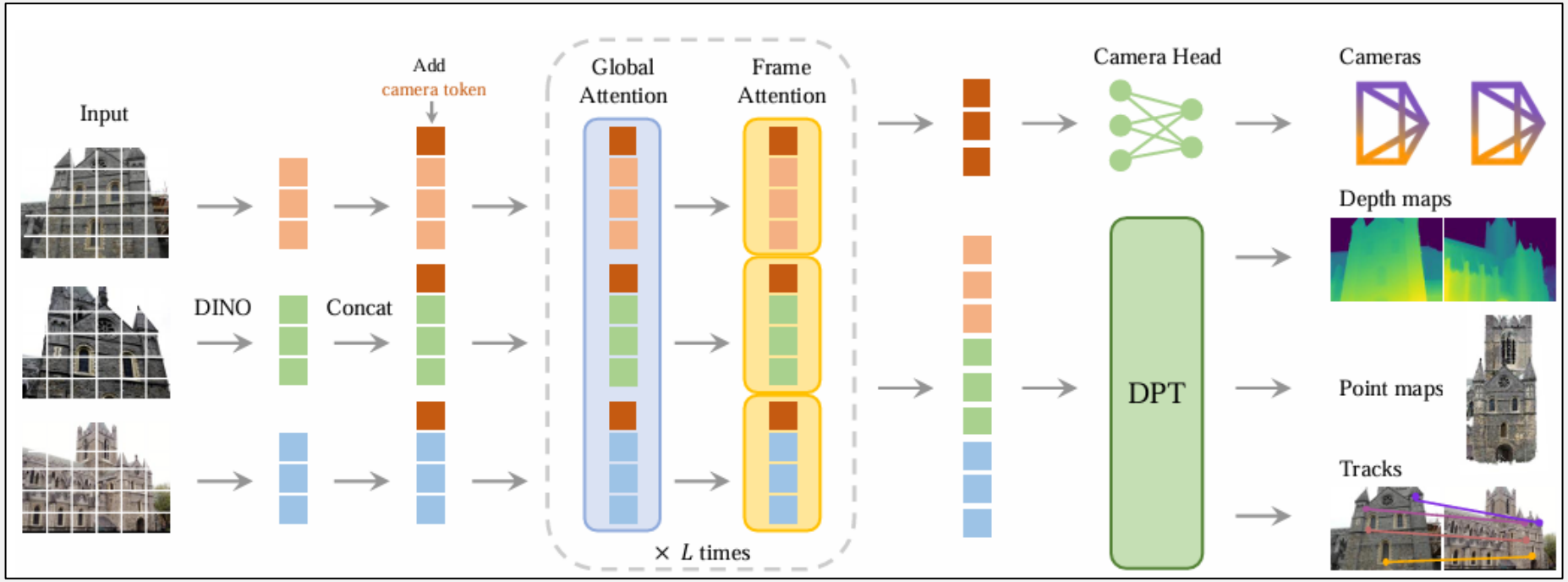
- Depth Anything V1의 단점을 해결(detail, inaccurate of transparent object)



Introduction

VGGT: Visual Geometry Grounded Transformer

- Single ~ hundreds images를 input으로 받아 여러 3D task를 수행하는 model

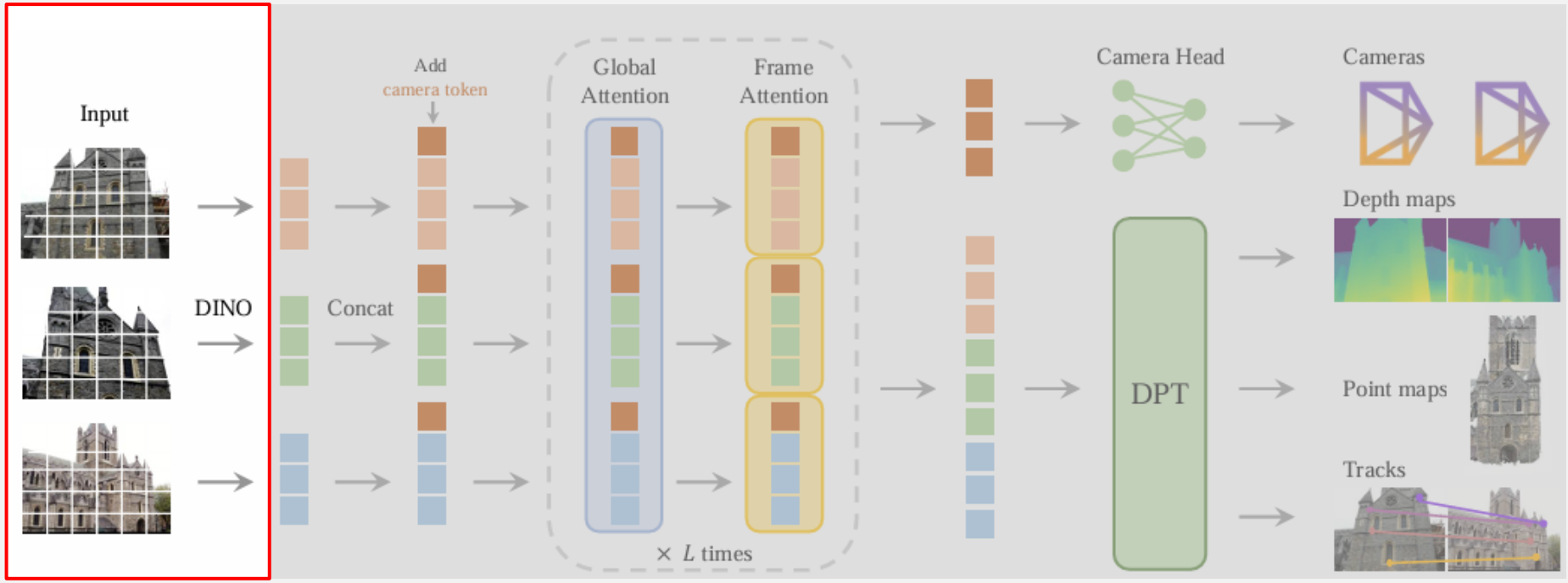


Introduction

VGGT: Visual Geometry Grounded Transformer

❖ Step 1

- Input images are converted to tokens through Pre-trained DINOv2

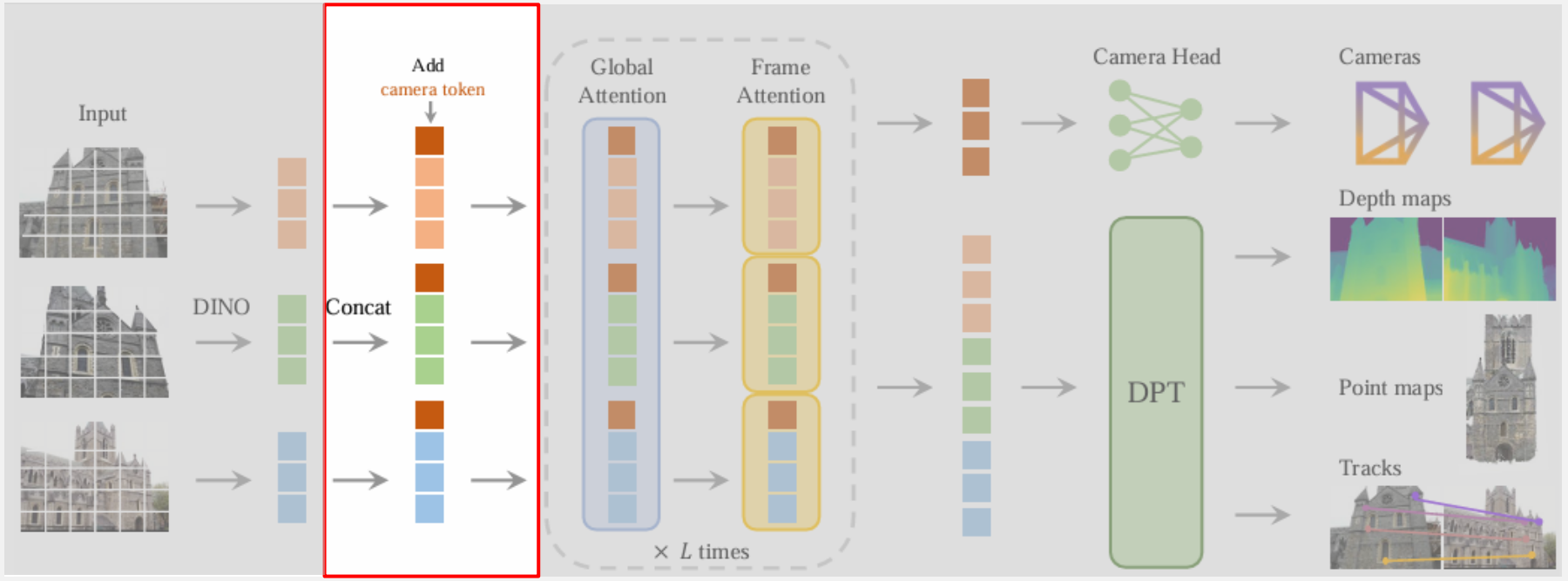


Introduction

VGGT: Visual Geometry Grounded Transformer

❖ Step 2

- Camera token을 image마다 add
  - camera token: frame의 camera 정보를 담는 token
  - register token: 첫번째 frame과 나머지 frame을 구분, 첫번째 frame을 월드 좌표계 기준으로 삼음

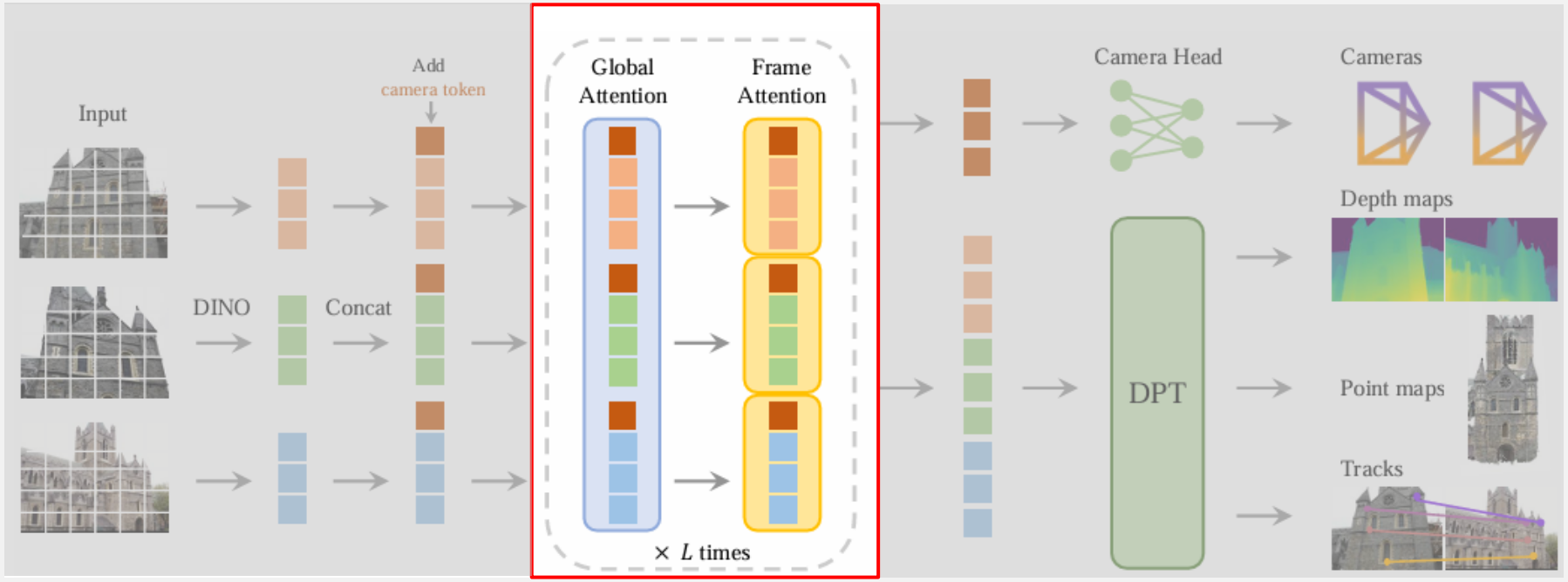


Introduction

VGGT: Visual Geometry Grounded Transformer

❖ Step 3

- Alternating Attention
  - ➢ Frame Attention → 각 frame내의 token 연산
  - ➢ Global Attention → 모든 frame의 token 연산

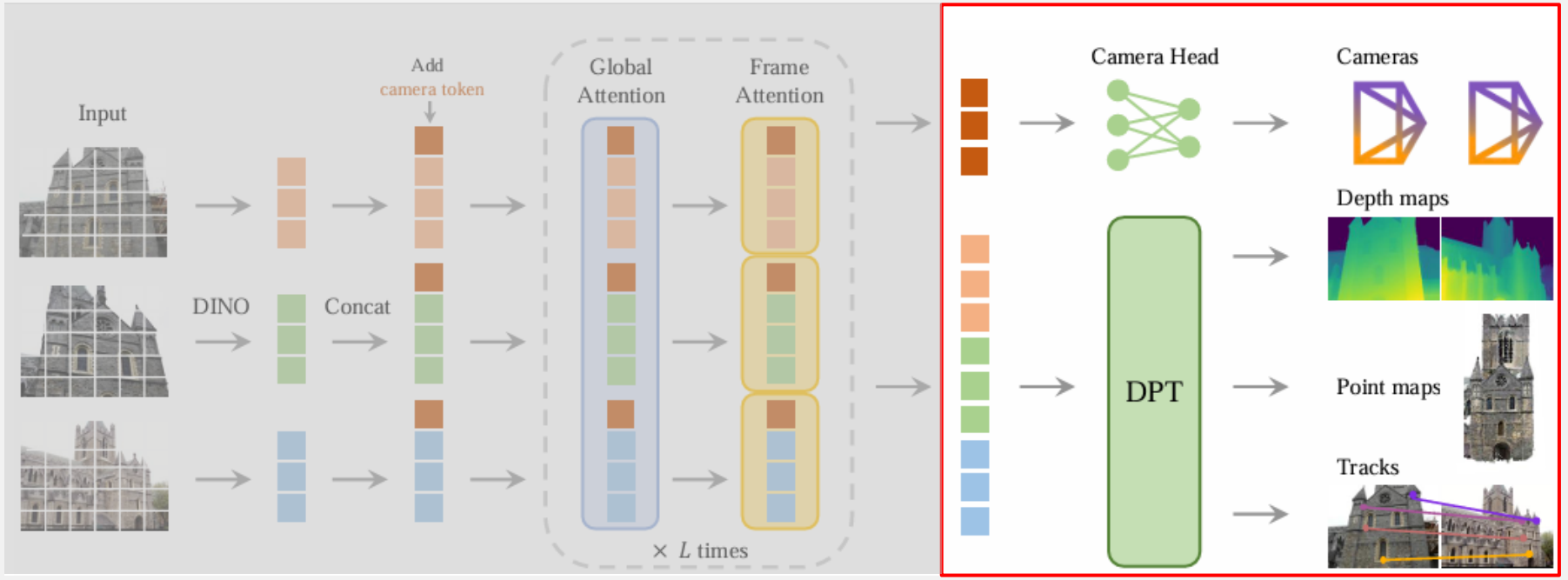


Introduction

VGGT: Visual Geometry Grounded Transformer

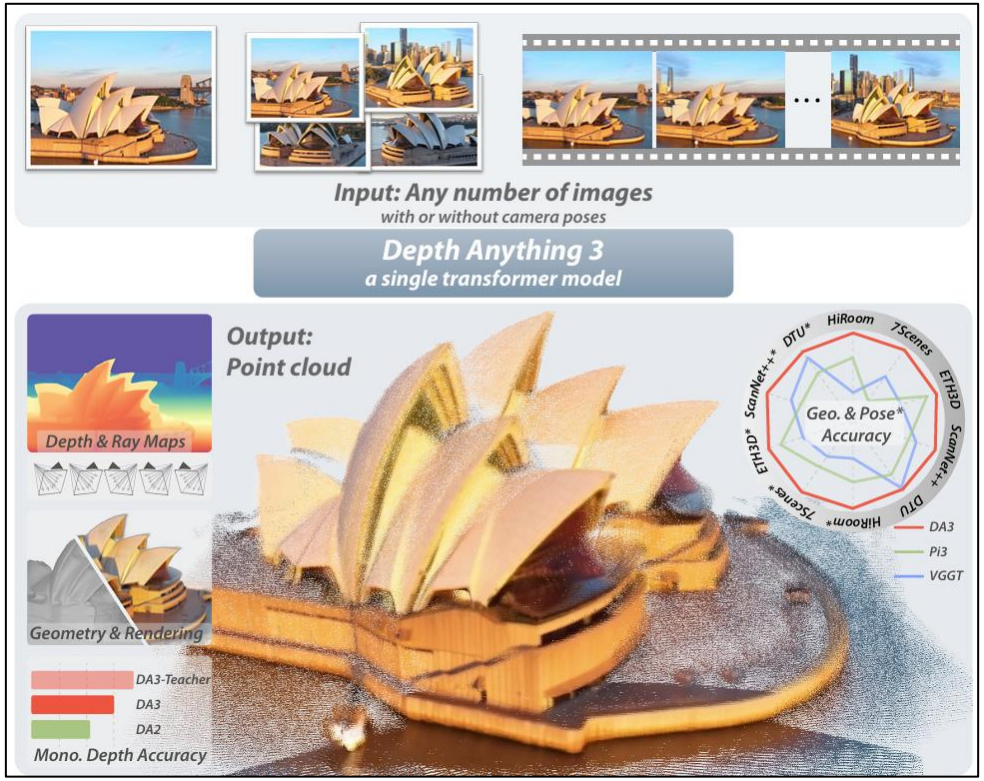
❖ **Step 4**

- Camera Head → Camera parameters prediction
- DPT → Depth map, point map, tracking prediction



Introduction

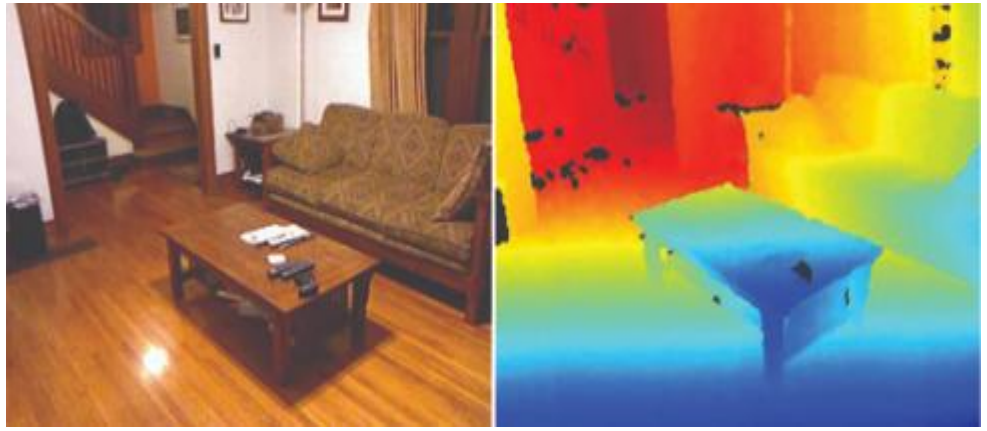
- 최근 연구에서 여러 Task를 동시에 처리하기 위한 통합 model 개발
- 하지만 복잡한 구조의 Architecture 사용, 여러 task에 대해 joint optimization을 통해 training 해서 cost, time-consuming
- 이런 문제를 해결하고자 복잡한 Architecture 엔지니어링을 포기하고 minimal modeling strategy 추구

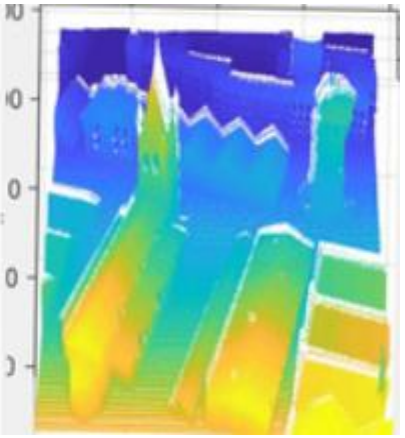


Introduction

- ❖ Depth map?
  - 각 Pixel이 camera로부터 얼마나 떨어져 있는지 나타내는 map
  - Depth map을 통해 각 pixel을 3D 공간의 point로 나타낼 수 있음
- ❖ Camera Pose?
  - Single image에서 depth estimation을 통해 point cloud를 얻을 수 있지만 부정확
  - Multi-view image를 통해 더 정확한 point cloud를 얻을 수 있음
  - Image에서 생성된 point cloud는 local coordinate system에 있어 global coordinate system 변환 필요

    → 이때 Camera pose가 필요



Introduction

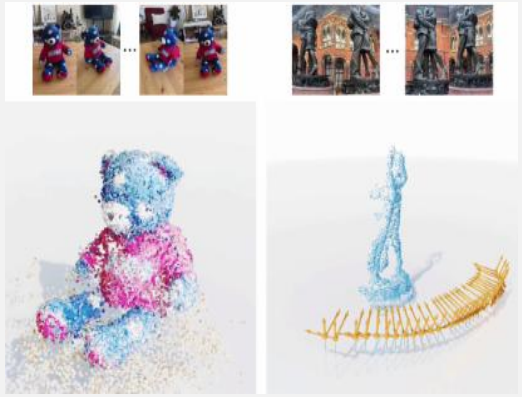
Depth Anything V3는 Depth, Pose estimation을 통해 3D reconstruction 목표 → Dense prediction

❖ Sparse prediction?

- SfM: Sparse feature points를 추출, correspondence를 찾아 sparse 3D point를 reconstruction함
- Dense prediction보다 가벼워서 real-time 분야에서 사용
- Sparse하기 때문에 정교한 3D reconstruction은 어려움

❖ Dense prediction?

- Input image의 모든 pixel에 대해 prediction 진행
- 정교한 3D reconstruction 가능
- Dense하기 때문에 무거워서 real-time 제한



Formulation

- Input image는 Eq. 1 같이 표기하고 N_v=1인 경우 Monocular image, N_v >1 이면 video 또는 multi-view image임
- Image는 depth, camera extrinsics, intrinsics parameters를 가지고 pixel p는 Eq. 2에 의해 3D point로 projects
- 이 수식을 통해 Depth, camera의 extrinsics, intrinsics을 알면 image의 한 점이 3D 공간의 어떤 지점에 해당하는지 계산 가능

$$\mathcal{I} = \{\mathbf{I}_i\}_{i=1}^{N_v} \text{ with each } \mathbf{I}_i \in \mathbb{R}^{H \times W \times 3} \quad (1)$$

I: model의 input image로 이루어진 집합
I_i: i번째 input image
H: Height
W: Width

$$\mathbf{P} = \mathbf{R}_i\left(\mathbf{D}_i(u,v)\,\mathbf{K}_i^{-1}\mathbf{p}\right) + \mathbf{t}_i, \quad (2)$$

P: world coordinate system에서의 3D point
p: pixel
R_i: i번째 camera의 rotation matrix
$D_i(u,v)$: i번째 image의 pixel u,v에 해당하는 depth value
K_i^{-1}: i번째 camera의 intrinsics matrix K의 역행렬
t_i: i번째 camera의 translation vector

Formulation

Depth-ray representation

- Orthogonality constraint으로 인해 rotation matrix를 예측하는건 어려움
- 이를 피하기 위해 Input image 및 depth map에 정렬된 per-pixel ray map을 사용해 카메라 pose를 암시적으로 표현
- 각 Pixel에 대해 camera ray $\mathrm{r} \in \mathbb{R}^6$은 원점 $\mathrm{t} \in \mathbb{R}^3$, 방향 $\mathrm{d} \in \mathbb{R}^3$로 정의 됨
  - $\mathrm{r}(t, d)$, ray의 방향 d는 p를 camera frame으로 back projection하고 이를 world frame으로 rotation해서 얻음
- Dense ray map은 모든 pixel에 대해 위에 parameter를 저장
- 따라서 World coordinate system의 3D point Eq. 4로 표현 됨

$$d = RK^{-1}p \quad (3)$$

d: ray의 방향
R: camera의 rotation matrix
K^{-1}: camera의 intrinsics matrix K의 역행렬

$$P = t + D(u, v) \cdot d \quad (4)$$

P: world coordinate system에서의 3D point
$D(u, v)$: image의 pixel u,v에 해당하는 depth value
t: camera의 translation vector
d: ray의 방향

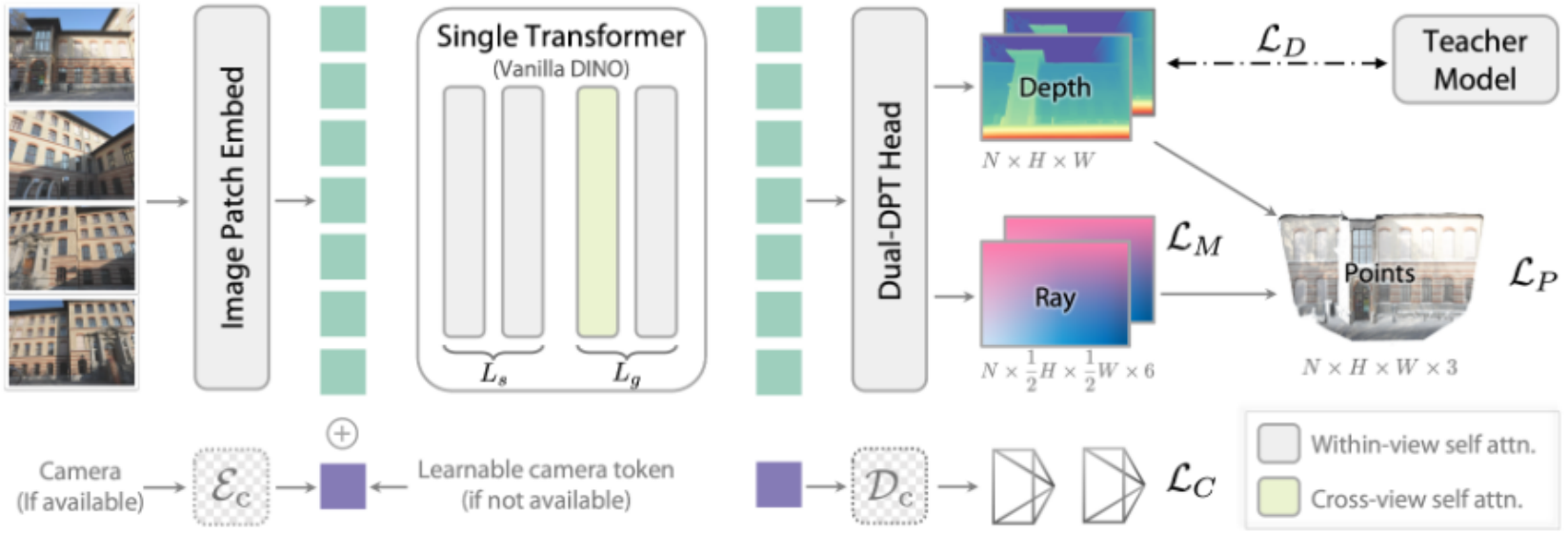
Formulation

Minimal prediction targets

- 최근 연구들은 3D 재구성을 위해 camera pose, local/global point map, depth 등 여러 task을 동시에 예측하는 multi-task learning 방식을 사용
- 이런 방법은 Model에서 더 많은 supervision을 제공하고 camera pose accuracy를 향상시키는데 도움이 될 수 있음
- 하지만 여러 task을 동시에 training하면 entanglement 문제가 발생할 수 있음
- 이런 문제를 극복하기 위해 Minimal prediction targets 전략을 사용함
  - Ray representation을 통해 depth, pose estimation
  - But.., ray map에서 camera pose를 복구하는 것은 계산 비용이 많이 드는 문제가 발생
    이를 해결하고자 lightweight camera head인 D_c를 추가

Architecture

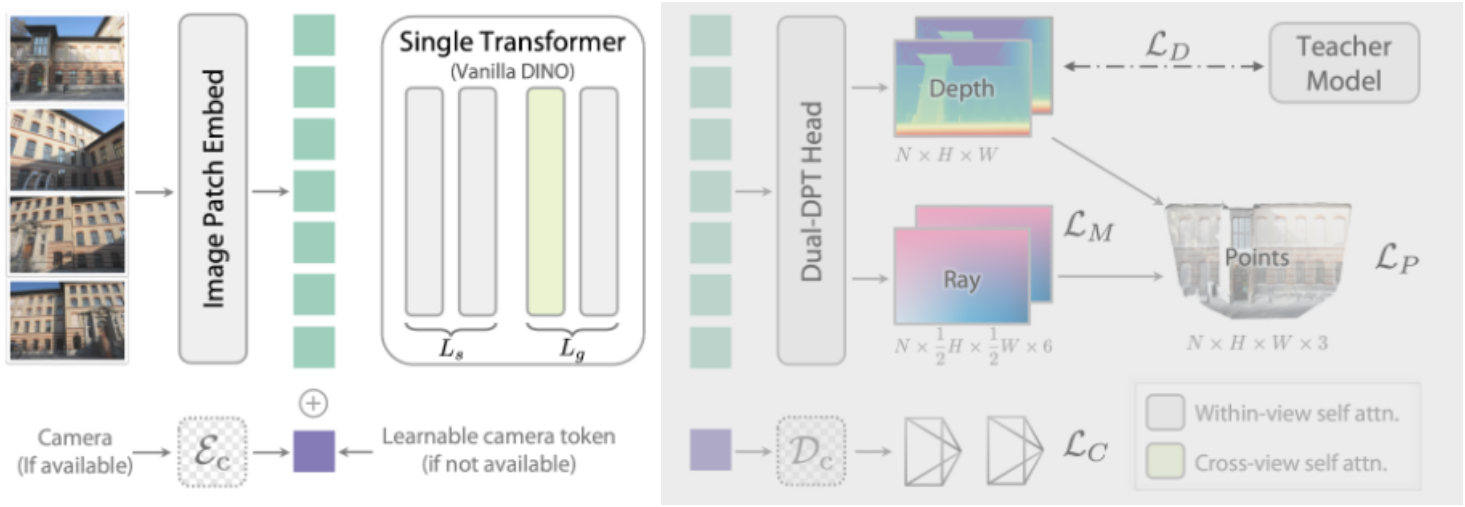
1. Single transformer model
2. Optional camera encoder
3. Dual-DPT head



Architecture

Single transformer model

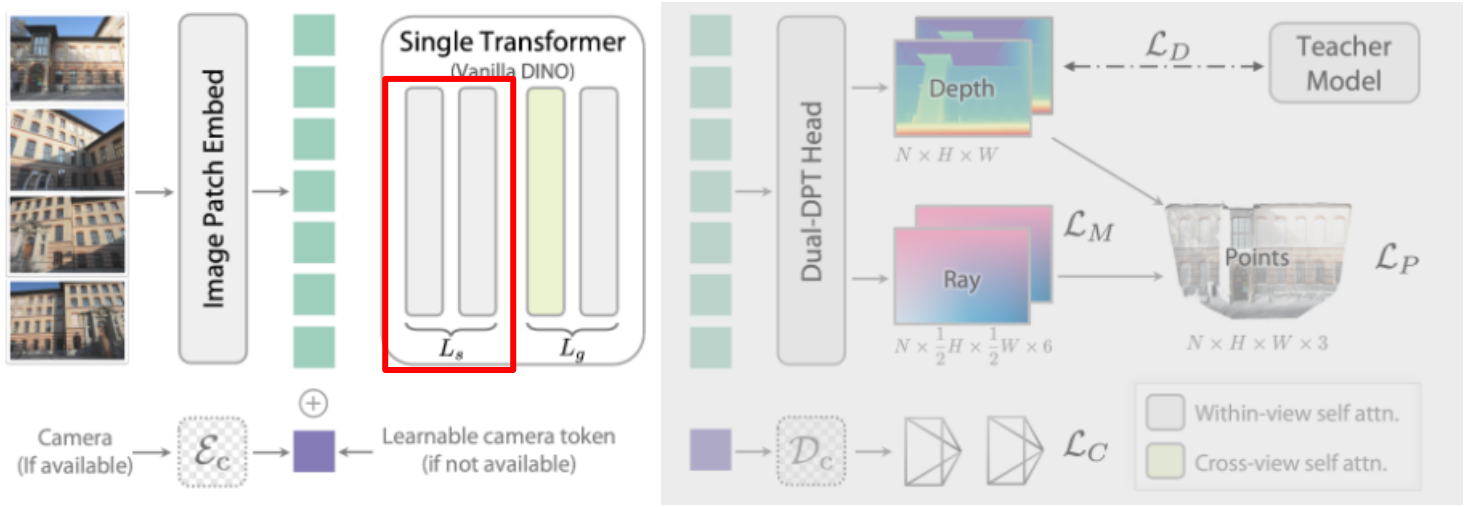
- Backbone으로 Vanilla DINOv2 사용
  - ➢ 저자가 제안한 것처럼 복잡한 모듈 추가 없이 Pre-trained Vanilla DINOv2를 사용
- L_s, L_g layer로 분할



Architecture

Single transformer model

- Backbone으로 Vanilla DINOv2 사용
  - ➢ 저자가 제안한 것처럼 복잡한 모듈 추가 없이 Pre-trained Vanilla DINOv2를 사용
- L_s, L_g layer로 분할
  - ➢ L_s: 각 image 내에서만 self-attention을 적용

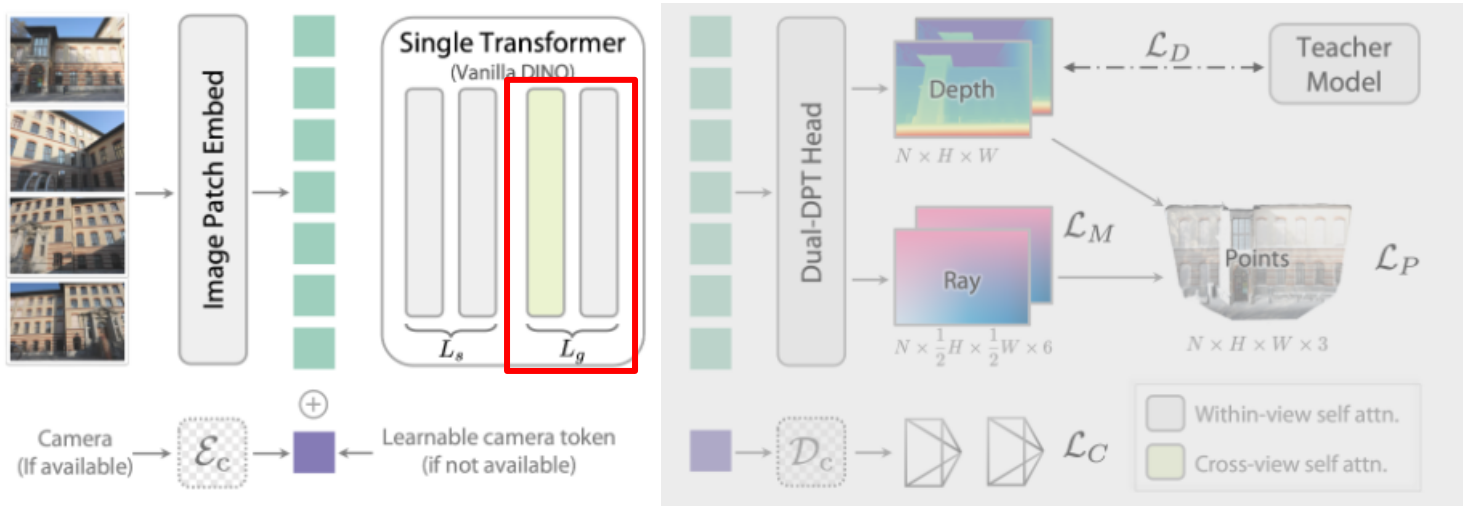


Architecture

Single transformer model

- Backbone으로 Vanilla DINOv2 사용
  - ➢ 저자가 제안한 것처럼 복잡한 모듈 추가 없이 Pre-trained Vanilla DINOv2를 사용
- L_s, L_g layer로 분할
  - ➢ L_g: cross-view attention과 within-view attention을 번갈아가며 적용

    $L_s : L_g = 2:1$ 비율로 적용
- Input-adaptive 설계를 통해 모델이 single image만 입력으로 받으면 MDE mode로 전환



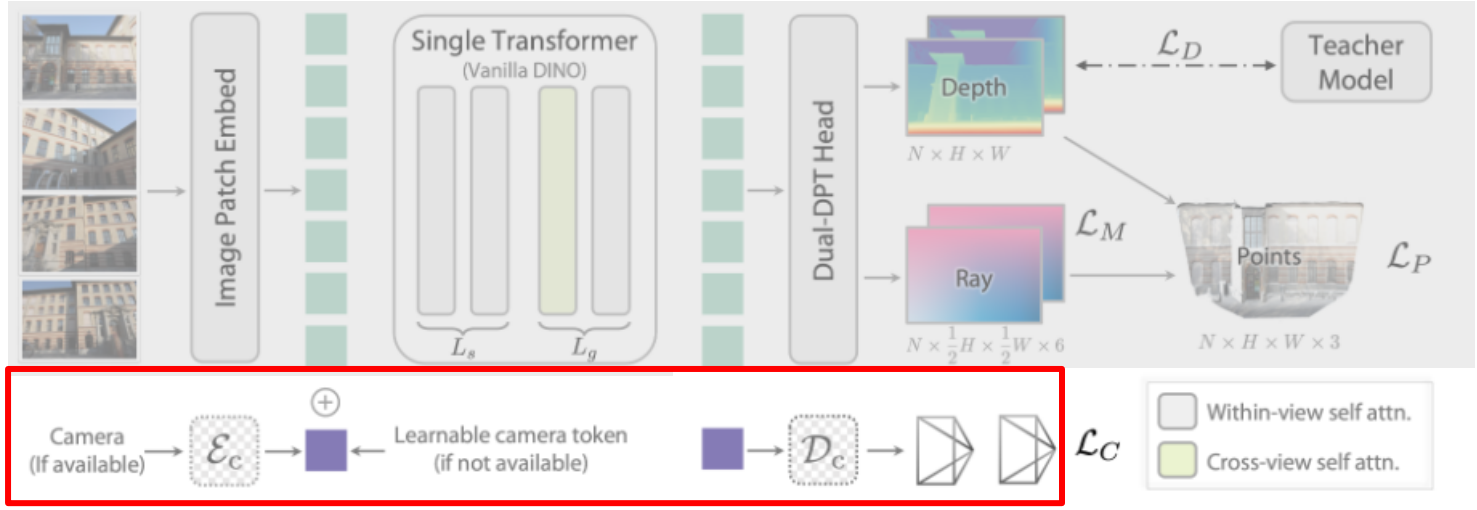
Architecture

Camera condition injection

- Camera pose가 있는 입력과 없는 입력 모두 처리하기 위해 camera token c_i를 추가
  - ➢ Camera pose가 주어지면 lightweight MLP을 통해 ε_c를 얻음
  - ➢ 그렇지 않은 경우 learnable token c_l를 사용

    learnable token c_l는 모델에게 Camera pose가 주어지지 않았다는 것을 알려주면서 placeholder 역할을 함

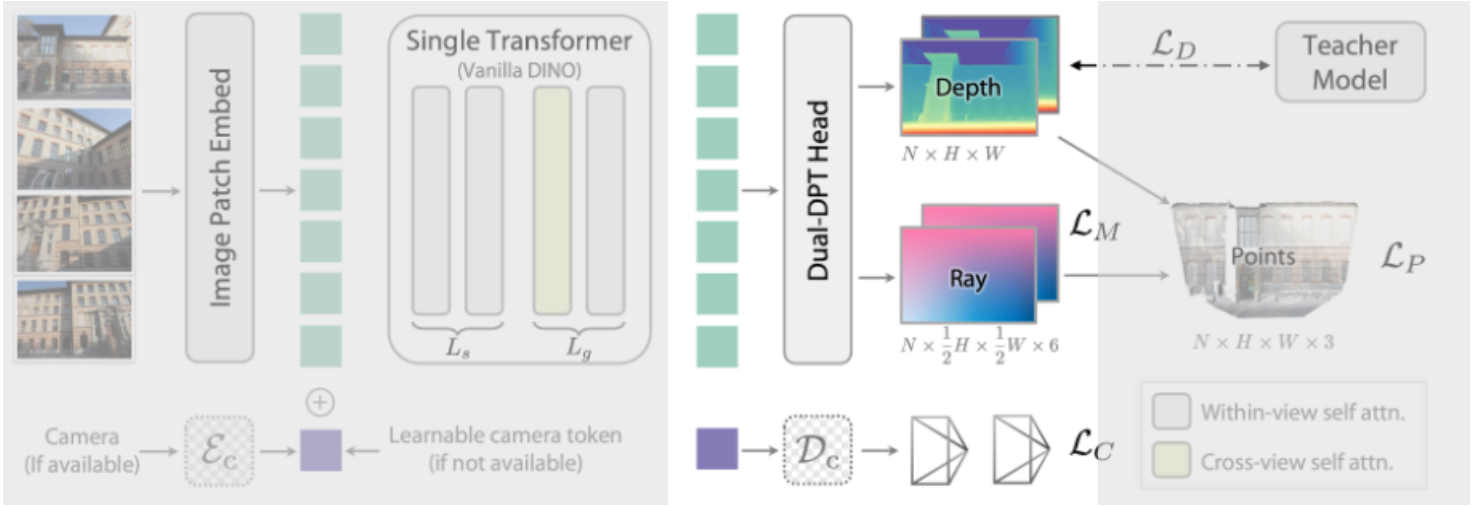
    또한 Camera pose가 주어지지 않으면 camera pose를 estimation → lightweight camera head인 $\mathcal{D}_c$

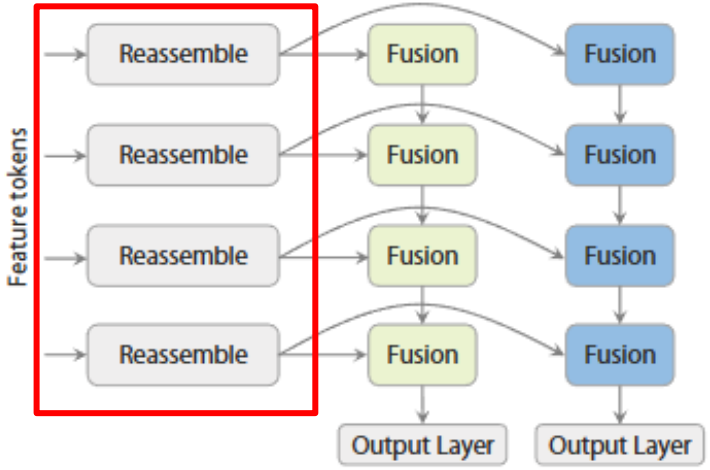


Architecture

Dual-DPT head

- Dense depth, ray value를 같이 출력하는 Dual-DPT head를 도입
  - ➢ Model의 backbone에서 추출된 feature는 reassemble 모듈을 거침

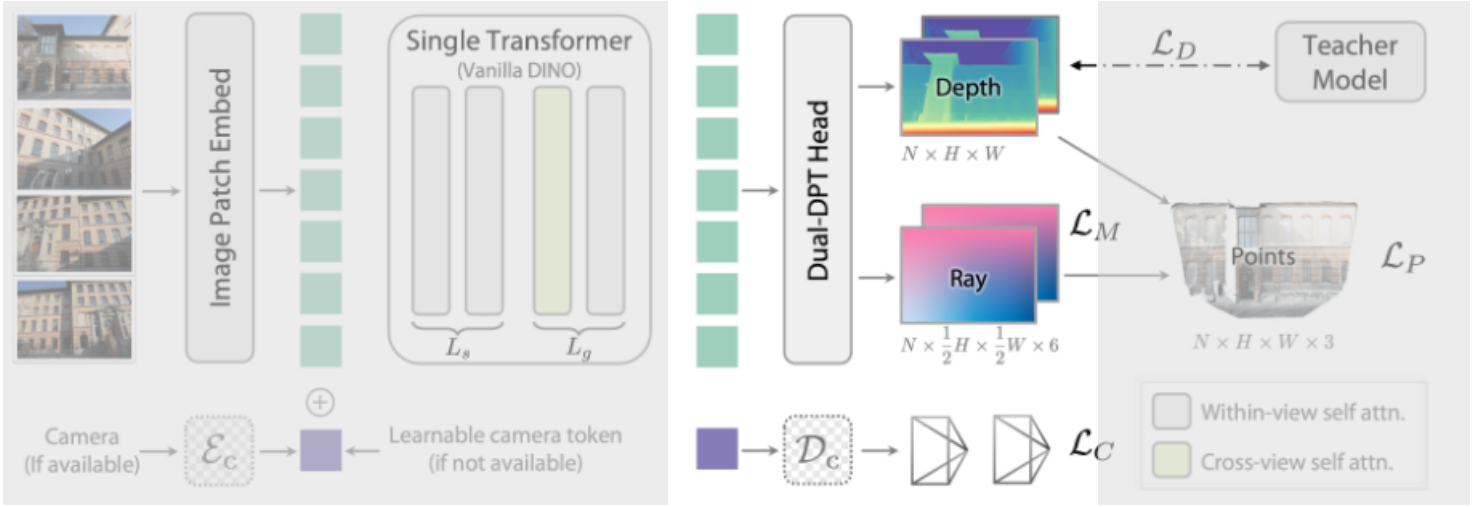


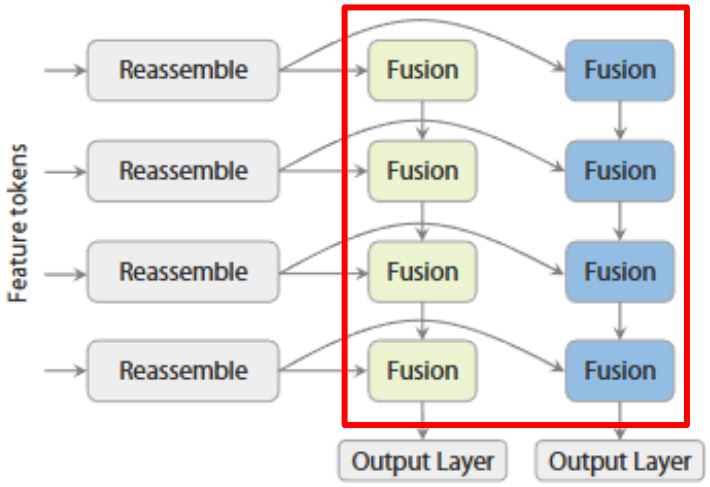


Architecture

Dual-DPT head

- Dense depth, ray value를 같이 출력하는 Dual-DPT head를 도입
  - ➢ Model의 backbone에서 추출된 feature는 reassemble 모듈을 거침
  - ➢ 그후 두개의 branch로 나뉘어 output layer을 통해 각각 Depth, ray map 출력

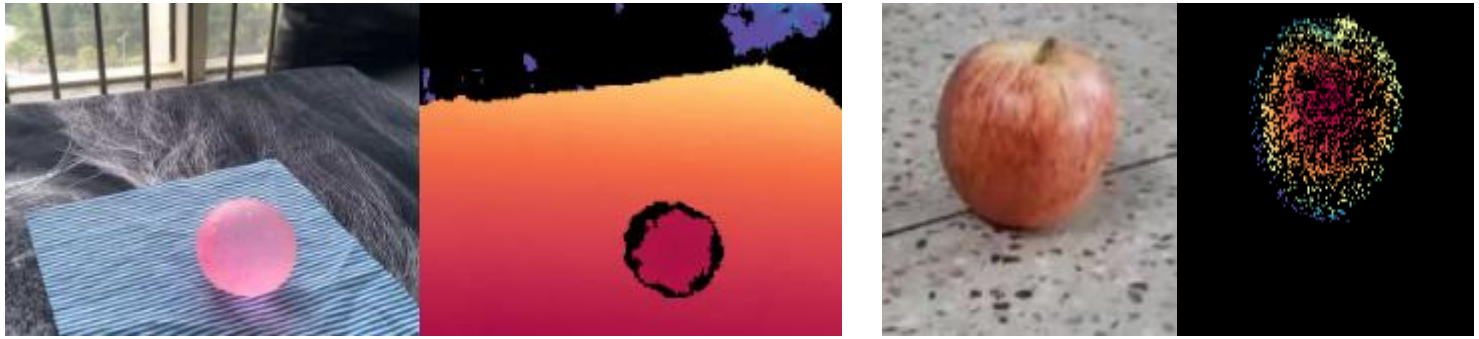




Training

Teacher-student learning paradigm

- Training data는 real-world, synthetic datasets을 사용
  - ➢ Real-world data는 noise, incomplete해서 supervision으로 사용하기 제한점이 있음
  - ➢ 이를 완화하기 위해 synthetic datasets으로 teacher model을 학습하고 real-world input을 받아 pseudo-depth map 생성
  - ➢ 생성된 pseudo-depth map은 RANSAC least squares를 통해 원래 sparse하거나 noise가 많은 ground truth와 정렬



Result

- HiRoom: Blender로 렌더링된 synthetic datasets → 저자들이 만든 dataset(미공개)
- ETH3D: 실내 및 실외 환경 real-world dataset
- DTU: 실내 real-world dataset
- 7Scenes: 7개의 실내 장면으로 구성 → 움직임이 많아 pose 추정이 어려운 dataset
- ScanNet++: 실내 real-world dataset

Table 1: **Comparisons with SOTA methods on pose accuracy.** We report both Auc3 ↑ and Auc30 ↑ metrics. The top-3 results are highlighted as first , second , and third .

Methods	HiRoom		ETH3D		DTU		7Scenes		ScanNet++	
	Auc3	Auc30	Auc3	Auc30	Auc3	Auc30	Auc3	Auc30	Auc3	Auc30
DUSt3R	17.6	54.3	4.30	27.3	4.00	74.3	6.90	61.6	8.10	33.9
Fast3R	25.9	77.0	8.10	44.4	9.50	79.1	19.0	78.6	17.9	72.5
MapAnything	17.9	82.8	19.2	77.4	6.50	72.7	12.6	79.7	20.2	84.1
Pi3	67.0	94.8	35.2	87.3	62.5	94.9	25.5	86.3	50.7	92.1
VGGT	49.1	88.0	26.3	80.8	79.2	99.8	23.9	85.0	62.6	95.1
DA3-Giant	81.7	96.4	39.3	90.6	85.6	94.9	29.2	86.8	83.2	98.0
DA3-Large	37.9	84.5	19.0	81.7	58.4	95.3	25.1	85.4	46.9	92.1
DA3-Base	12.8	79.8	13.6	74.0	31.4	90.8	17.2	81.1	16.2	77.5
DA3-Small	3.40	64.6	4.89	51.9	9.46	82.2	6.19	71.8	2.86	51.8

Result

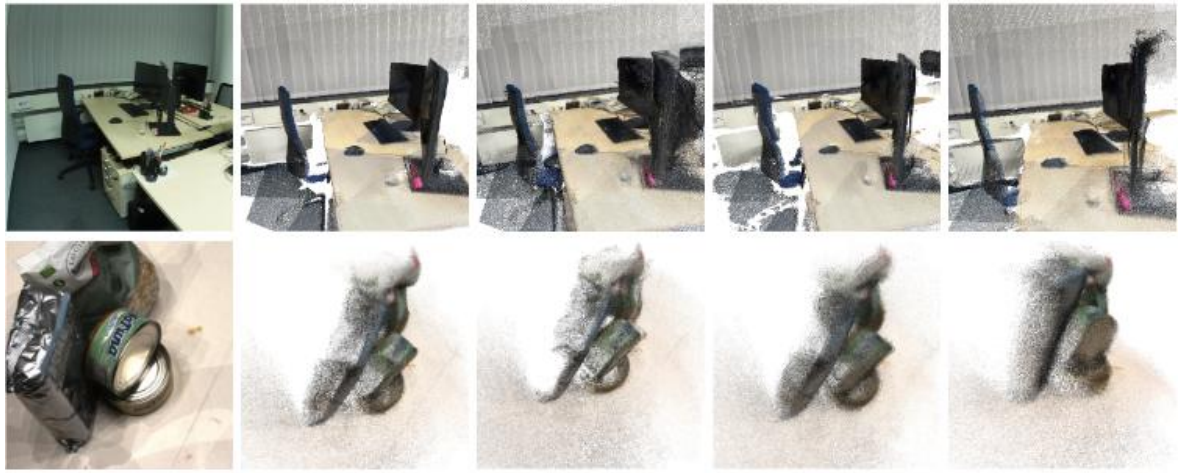
- HiRoom: Blender로 렌더링된 synthetic datasets → 저자들이 만든 dataset(미공개)
- ETH3D: 실내 및 실외 환경 real-world dataset
- DTU: 실내 real-world dataset
- 7Scenes: 7개의 실내 장면으로 구성 → 움직임이 많아 pose estimation이 어려운 dataset
- ScanNet++: 실내 real-world dataset

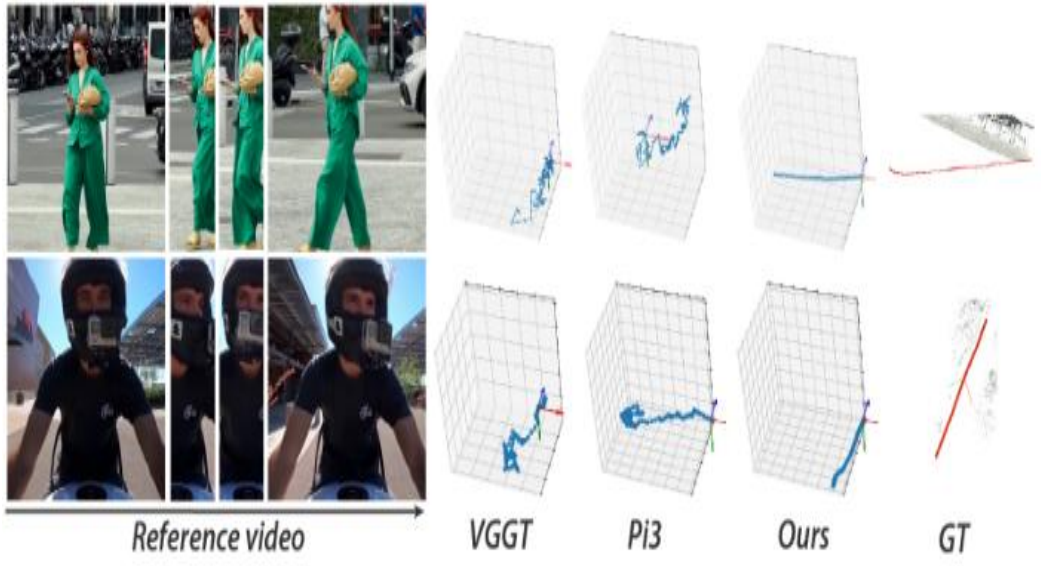
Methods	HiRoom		ETH3D		DTU		7Scenes		ScanNet++	
	w/o p.	w/ p.	w/o p.	w/ p.	w/o p.	w/ p.	w/o p.	w/ p.	w/o p.	w/ p.
DUSt3R	30.1	39.5	19.7	18.8	7.60	7.97	26.6	39.8	18.9	27.3
Fast3R	40.7	48.2	38.5	50.3	6.88	8.20	41.0	49.8	37.1	53.7
MapAnything	32.4	69.2	54.8	71.9	7.91	3.97	44.8	55.2	39.4	71.3
Pi3	75.8	85.0	72.7	80.6	3.28	1.72	44.2	57.5	63.1	73.3
VGGT	56.7	70.2	57.2	66.7	2.05	1.44	47.9	51.4	66.4	70.7
DA3-Giant	89.3	95.2	74.4	85.8	1.92	0.91	52.0	52.3	76.4	79.2
DA3-Large	48.2	85.7	57.3	79.1	3.45	2.48	48.7	48.7	58.9	72.9
DA3-Base	18.6	71.7	52.8	66.6	5.14	1.99	37.8	47.2	39.7	66.3
DA3-Small	12.9	43.1	39.4	58.2	5.12	4.05	30.8	39.5	24.2	45.7

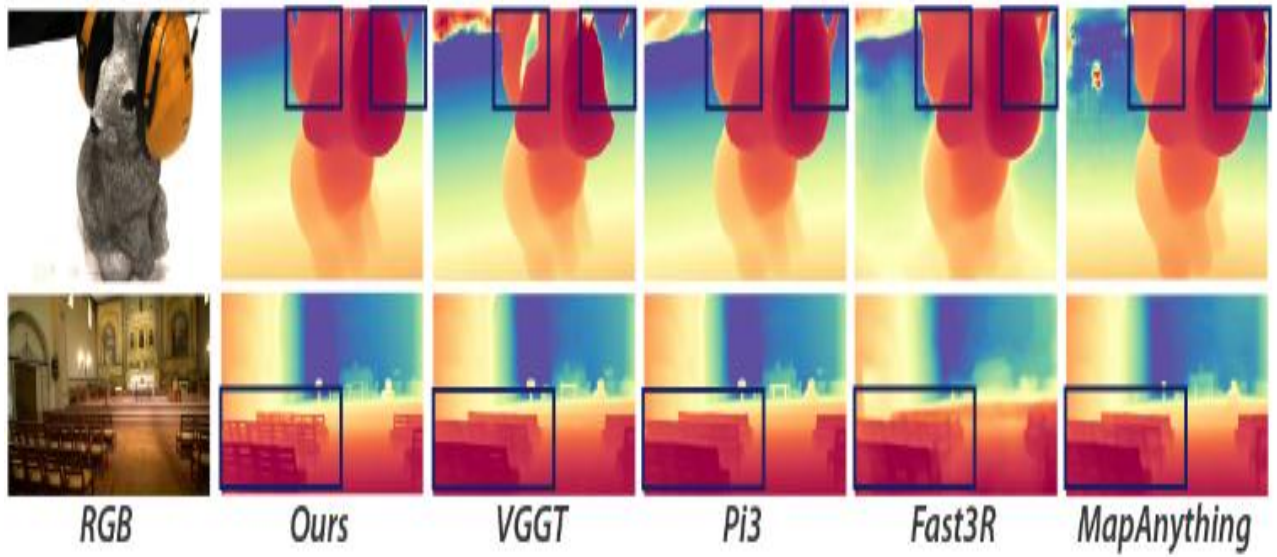
Result

Table 3: **Monocular depth comparisons.** $\delta_1 \uparrow$

Method	KITTI	NYU	SINTEL	ETH3D	DIODE	Rank
DA2	94.6	97.9	77.2	86.5	95.2	2.60
VGGT	91.7	97.9	67.9	97.5	95.3	3.75
DA3	95.3	97.4	75.5	98.6	95.4	2.20
Teacher	97.2	97.9	81.4	99.8	96.6	1.00







Discussion

- 복잡한 모듈 추가 없이 SOTA 달성
- Depth-ray representation을 통해 기존 방법보다 정확한 pose estimation
- Camera Head parameters가 무시할만한 수준이라고 했는데 0.48B는 무시할만한 수준이 아닌 것 같음
- Teacher model이 synthetic dataset으로만 학습해서 real-world dataset에서 성능이 안 좋아야 할 것 같은데 좋은 이유를 모르겠음

Model	Max # of Images	Parameters			Running Speed
		Backbone	DualDPT	CameraHead	
VGGT(Reference)	400-500	0.91B	0.064B	0.22B	34.1 FPS
DA3-Giant	900-1000	1.130B	0.050B	0.48B	37.6 FPS
DA3-Large	1500-1600	0.300B	0.047B	0.21B	78.37 FPS
DA3-Base	2100-2200	0.086B	0.045B	0.12B	126.5 FPS
DA3-Small	4000-4100	0.022B	0.043B	0.03B	160.5 FPS

Table 3: **Monocular depth comparisons.** $\delta_1 \uparrow$

Method	KITTI	NYU	SINTEL	ETH3D	DIODE	Rank
DA2	94.6	97.9	77.2	86.5	95.2	2.60
VGGT	91.7	97.9	67.9	97.5	95.3	3.75
DA3	95.3	97.4	75.5	98.6	95.4	2.20
Teacher	97.2	97.9	81.4	99.8	96.6	1.00

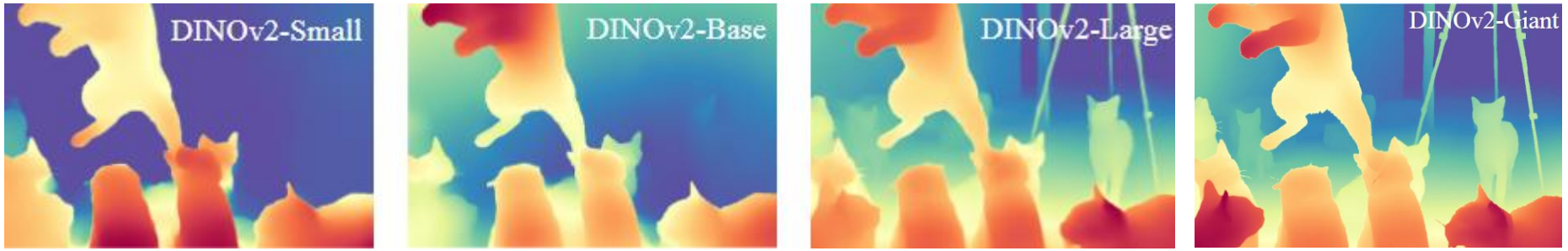
Discussion

- VGGT를 비판한게 복잡하고 transformer model 여러 개 사용했다고 했는데 그 비교 보다는 parameters수에서 비교한 것 같음
- depth+ray가 가장 좋은데 저자들은 depth+ray+cam 사용

  → depth+ray에서 camera pose 뽑으면 연산량 높아서 비교적 적은 연산량 가지는 cam 써서 어느정도 trade-off 관계 쓴듯함

Methods	HiRoom		ETH3D		DTU		7Scenes		ScanNet++	
	Auc3↑	F1↑	Auc3↑	F1↑	Auc3↑	CD↓	Auc3↑	F1↑	Auc3↑	F1↑
a. Proposed Arch.	**39.2**	**47.0**	**21.0**	**55.4**	**45.8**	**3.82**	**26.2**	47.6	**30.3**	**51.1**
b. VGGT Style	3.72	14.5	2.31	27.4	1.38	6.93	0.97	21.4	2.03	12.2
c. Full Alt.	24.7	29.3	13.1	51.9	44.6	4.23	21.1	**48.6**	27.7	47.5

Methods	HiRoom		ETH3D		DTU		7Scenes		ScanNet++	
	Auc3↑	F1↑	Auc3↑	F1↑	Auc3↑	CD↓	Auc3↑	F1↑	Auc3↑	F1↑
depth + pcd + cam	9.1	12.8	19.0	60.4	42.3	4.918	20.8	43.4	22.0	43.0
depth + cam	10.8	16.5	9.9	48.0	23.3	5.316	13.0	38.5	13.3	41.0
depth + ray	**48.7**	**60.3**	**25.5**	**65.4**	46.5	3.919	24.0	**46.5**	**35.5**	53.4
depth + ray + cam	37.2	45.4	22.3	59.4	**56.3**	**3.066**	**25.7**	45.6	34.1	**56.5**



감사합니다.