Monocular Depth Estimation and Evaluation using LiDAR

Team 씨리스랩

2022.07.15.

자율주행 데브코스 3기 최종 프로젝트 발표

Contents

programmers 자율주행 데브코스 3기

Part 01

프로젝트 개요

프로젝트 목표

수행 계획

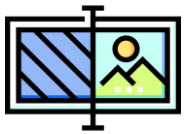
01 프로젝트 목표

Monocular Depth Estimation and Evaluation using LiDAR

단일 이미지로부터의 깊이 추정과 라이다 데이터를 이용한 평가

단일 이미지를 이용한 깊이 추정 알고리즘 개발

Monocular Depth Estimation Algorithm



LiDAR 데이터를 활용하여 깊이 추정 결과 평가

Evaluation Algorithm using LiDAR Data

보고서 작성 및 발표

Report and Presentation

02 수행 계획

역할 분배

한은기	이현진	유희평
▪ 보고서 작성 및 발표 ▪ 프로젝트 관리 및 협업툴 운용 ▪ Architecture 설계 및 2D Projection	▪ 학습 및 평가 Data 획득 및 가공 ▪ LiDAR를 이용한 평가 알고리즘 설계 ▪ 2D Projection 및 시각화	▪ Monocular Depth Estimation 모델 비교 및 깊이 추정치 도출 ▪ 깊이 추정 결과 시각화

중점사항

- 프로젝트 일정 준수
- 원활한 커뮤니케이션
- 유연한 역할 분배

- CI: Formatting & UnitTest
- 단계적 구현: 기본 → 심화

02 수행 계획

프로젝트 일정

프로젝트 시작 | 데이터 공개 | 중간발표

Task	27 MON	28 TUE	29 WED	30 THU	01 FRI	02 SAT	03 SUN	04 MON	05 TUE	06 WED	07 THU	08 FRI	09 SAT	10 SUN	11 MON	12 TUE	13 WED	14 THU
▼ 깊이 추정																		
추정 모델 스터디	■	■	■	■	■	■	■	■										
모델 개선									■	■	■	■	■	■	■	■		
깊이 추정 시행										■	■	■	■	■	■	■	■	
시각화 및 데이터 추출														■	■	■	■	■
▼ 추정 평가																		
LiDAR & 평가 스터디				■	■	■	■	■										
2D Projection								■	■	■	■	■	■	■	■	■		
평가 알고리즘 구현									■	■	■	■	■	■	■	■	■	
추정 평가 시행										■	■	■	■	■	■	■	■	■
시각화 및 데이터 추출														■	■	■	■	■
▼ 프로젝트 관리																		
보고서 작성				■	■	■	■	■		■	■	■	■	■	■	■	■	■
GitHub 점검				■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
발표 대비															■	■	■	■

Part 02

프로젝트 수행

프로젝트 파이프라인

(1) 사전 준비

(2) LiDAR 데이터 변환

(3) 단일 이미지 깊이 추정

(4) 추정 결과 평가

00 프로젝트 파이프라인

01 사전 준비

- Dataset 구성 및 형식 확인
- Calibration 정보 획득

02 LiDAR 데이터 변환

- Point Cloud Data 획득
- 이미지 평면으로의 투영
- 투영 결과 저장

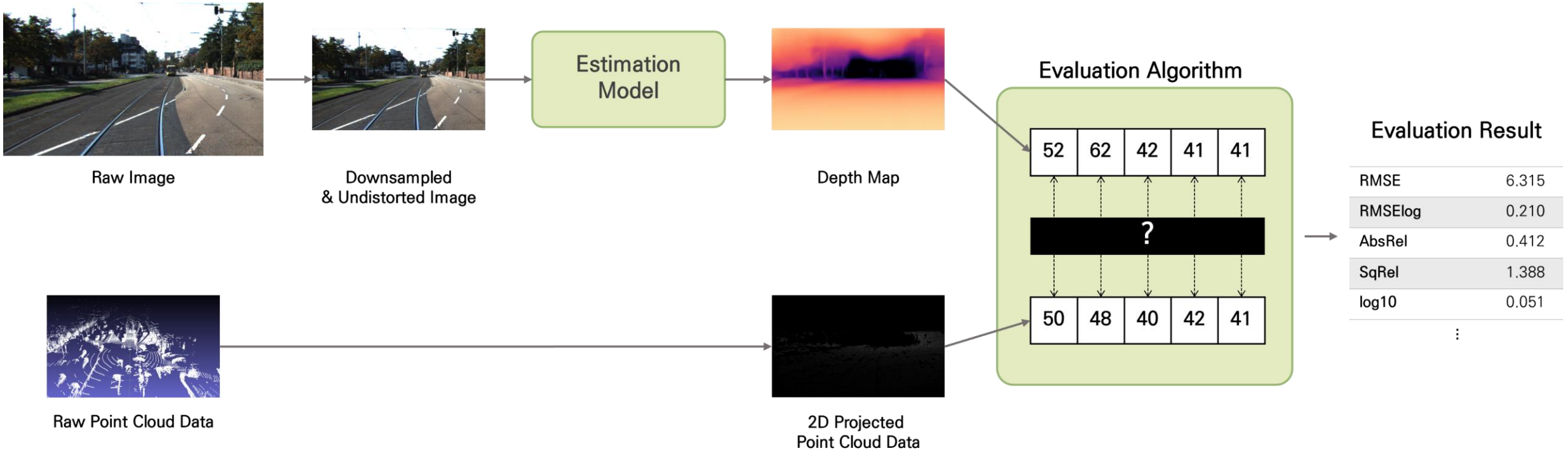
03 단일 이미지 깊이 추정

- Image 획득 및 왜곡 보정
- 모델을 사용한 깊이 추정
- 깊이 추정 결과 저장

04 추정 결과 평가

- LiDAR data 및 추정 결과를 이용한 정확도/에러 평가
- 평가 결과 출력 및 저장
- 추정 및 평가 반복

00 프로젝트 파이프라인



Evaluation Result

RMSE	6.315
RMSElog	0.210
AbsRel	0.412
SqRel	1.388
log10	0.051
⋮	

01 사전준비

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가

확인해야 하는 정보

Basic Info	▢ 데이터 수집 지역 및 시점 ▢ 사용된 센서 및 그 특성
Format & Size	▢ Image & Point Cloud format ▢ 전체 Dataset 및 개별 데이터 크기
Calibration	▢ 좌표계 및 변환 관계 ▢ Notations
Etc.	▢ Directory 구조 ▢ 포함된 기타 정보 등

획득해야 할 정보 및 형태

Image	▢ PNG 형식의 이미지 파일 ▢ 왜곡 보정하기
Point Cloud	▢ NumPy 배열 파일 (*.npy)로 변환해 저장 ▢ 파일을 읽어 모듈 내에서 배열로 변환
rosbag	▢ Synchronize 방식 ▢ 실시간 처리 기능 구현 여부 ▢ Image 및 Point Cloud 저장 format

02 LiDAR 데이터 변환

파일	형태 / 특징
*.bin / *. bag	Point cloud raw data
*.npy	NumPy array Raw data [x, y, z] for one point
*.npy & *.png	Projected Points [x, y, depth] for one point

LiDAR Point Cloud를 이미지 평면으로 투영하는 이유

- 차원을 축소하는 연산 (2D → 3D)이 차원을 확장시키는 연산 (3D → 2D) 보다 비교적 단순함
- LiDAR – Camera 혹은 World – Camera 등의 정보와 연산식이 제공됨

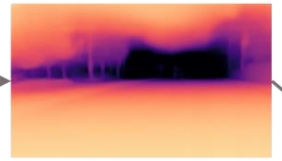
03 단일 이미지 깊이 추정 Monocular Depth Estimation

01 사전 준비 ✓ — 02 LiDAR 데이터 변환 ✓ — 03 단일 이미지 깊이 추정 ○ — 04 추정 결과 평가 ○

Raw Image

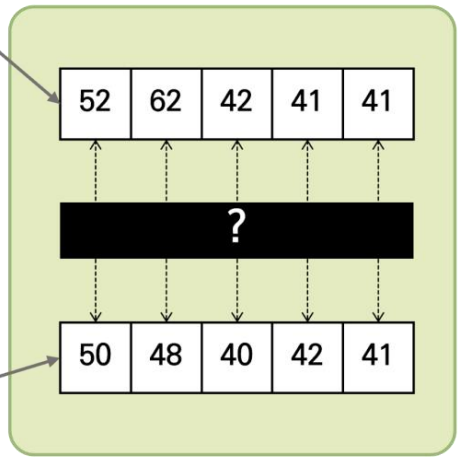
Downsampled
& Undistorted Image

Estimation Model



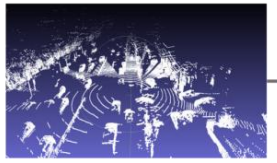
Depth Map

Evaluation Algorithm



Evaluation Result

RMSE	6.315
RMSElog	0.210
AbsRel	0.412
SqRel	1.388
log10	0.051
⋮	

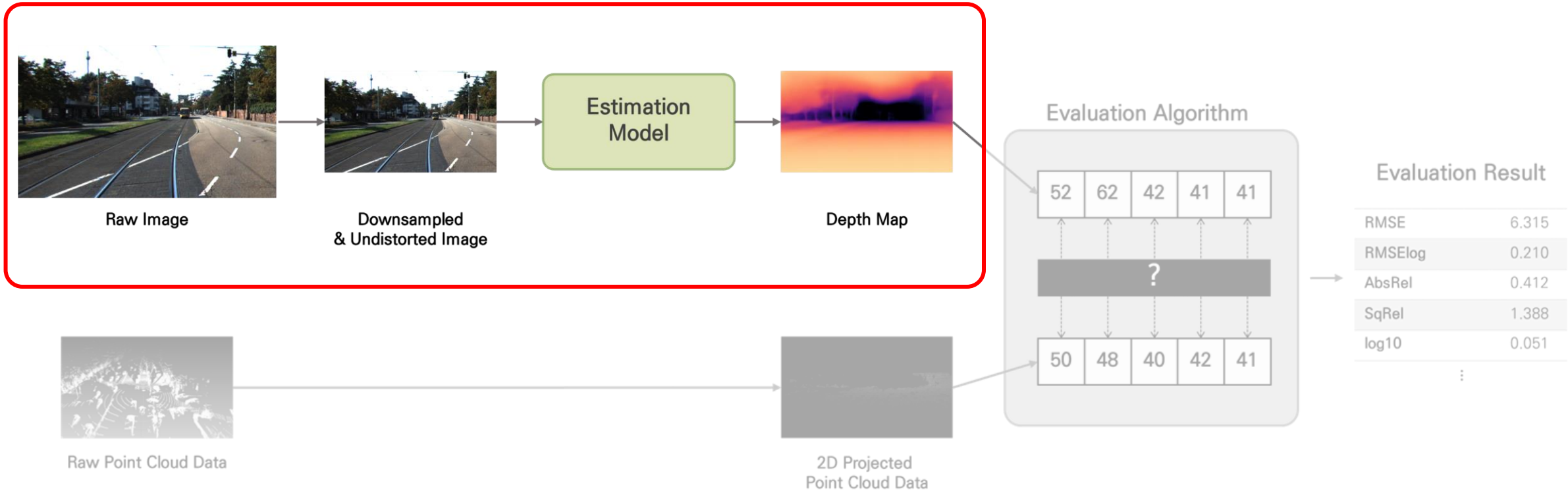


Raw Point Cloud Data

2D Projected
Point Cloud Data

03 단일 이미지 깊이 추정 Monocular Depth Estimation

01 사전 준비 ✓ — 02 LiDAR 데이터 변환 ✓ — 03 단일 이미지 깊이 추정 ○ — 04 추정 결과 평가 ○



03 단일 이미지 깊이 추정 Monocular Depth Estimation

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가

Monocular Depth Estimation Task

- 하나의 이미지 내 각 픽셀 위치에서의 깊이를 추정하는 문제
- 단일 이미지에서 깊이 추정의 어려움

  이미지 내 깊이 추정에 사용할 수 있는 사용할 수 있는 단서(cue) 부족

  scale ambiguity 문제가 존재

  반투명하거나 반사되는 재질에서 추정의 정확도 하락
- 최근에는 심층학습 기반의 깊이 추정 방식들이 주로 연구됨

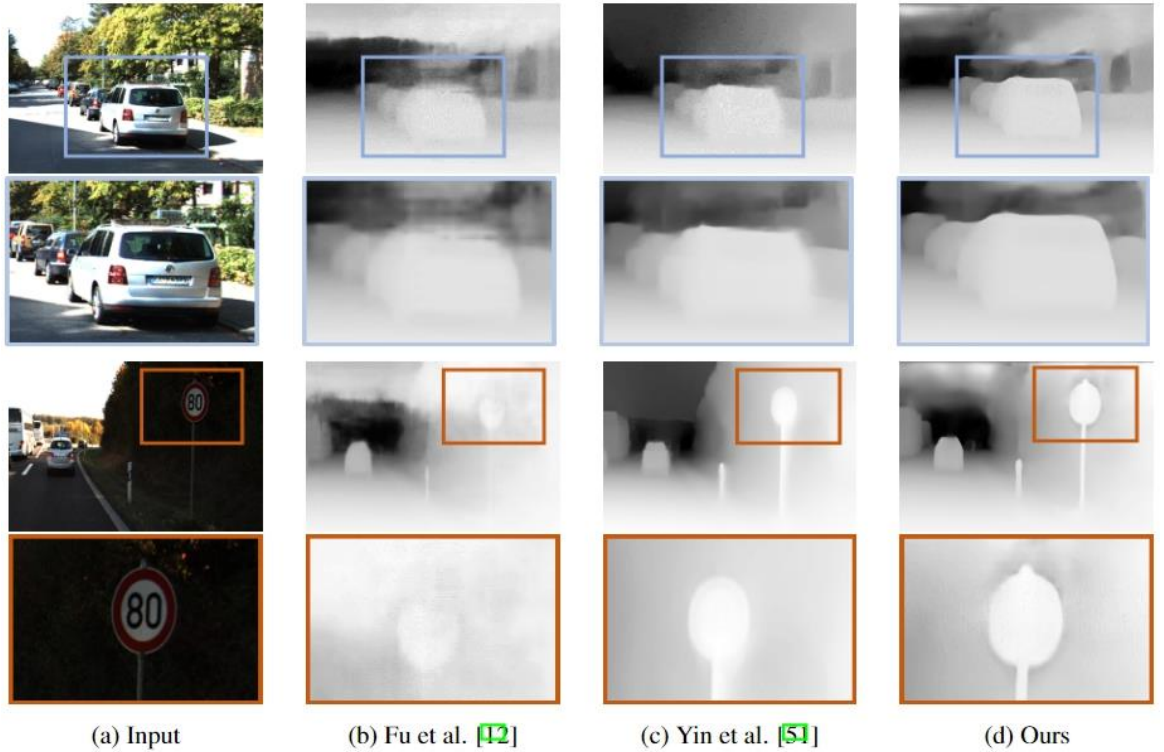
Supervised
- Depth 값인 ground truth 정보 필요
- CNN 방식 + Encoder-Decoder 구조
- Attention Mechanism과 Transformer
- BTS, DPT, AdaBins, DepthFormer 등

Unsupervised
- Self-supervised 방식이라고도 함
- GT 정보 없이, stereo image pair만을 이용
- Image Reconstruct 문제로 전환

03 단일 이미지 깊이 추정 Monocular Depth Estimation

BTS (Lee et al., 2019)	DPT (Ranftl et al., 2021)	AdaBins (Bhat et al., 2021)	DepthFormer (Li et al., 2022)	MonoDepth (Godard et al., 2017)

- Supervised Learning 방식, Encoder-Decoder scheme 사용
- Dense Feature Extractor를 Backbone으로 Feature Map 생성
- ASPP(Atrous Spatial Pyramid Pooling) layer를 사용해 contextual information을 추출함
- Decoding 단계에서 각 multiple stages에 LPG(Local Planar Guidance) layer를 사용해 large scale variation을 파악할 수 있고, 해상도를 입력 이미지 크기로 복원함
- Decoding에선 각 레이어들이 4차원 평면 coefficients를 각각 학습하고, 출력을 비선형적으로 결합하여 깊이 추정치를 반환함



Depth map result for KITTI Dataset

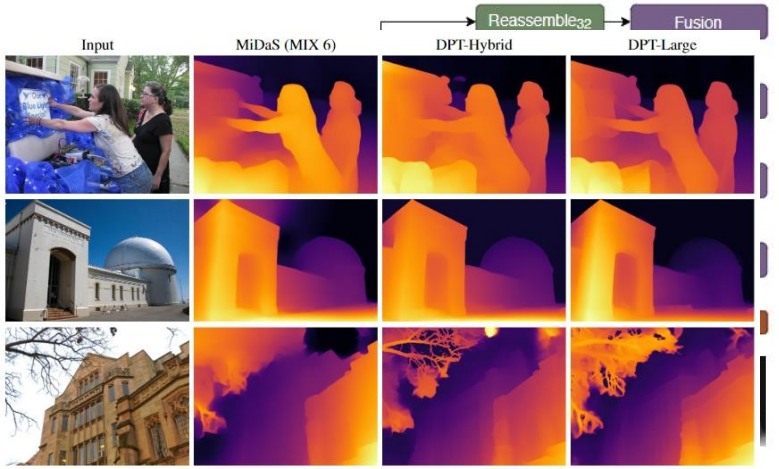
Lee, J. H., Han, M. K., Ko, D. W., & Suh, I. H. (2019). From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326.*

03 단일 이미지 깊이 추정 Monocular Depth Estimation

BTS (Lee et al., 2019)	DPT (Ranftl et al., 2021)	AdaBins (Bhat et al., 2021)	DepthFormer (Li et al., 2022)	MonoDepth (Godard et al., 2017)

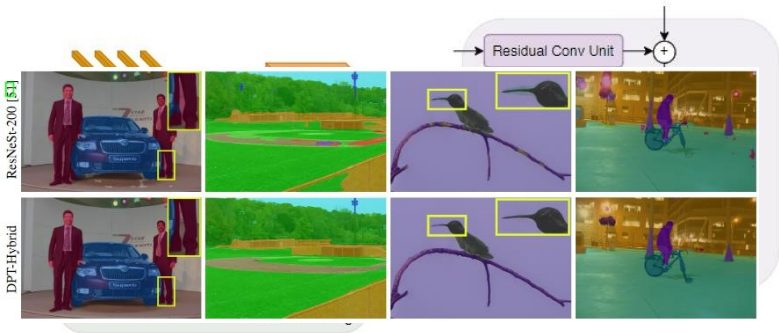
- Supervised Learning 방식, Encoder-Decoder scheme 사용
- Dense Prediction Transformer (DPT) 제안, Backbone으로 Vision Transformer(ViT) 사용
  - Embed — 입력 이미지를 Tokens으로 바꿈
  - Transformer — 여러 transformer stages에서 특징을 추출함
  - Reassemble — 여러 resolutions에서 서로 다른 stages의 tokens를 합쳐 image-like feature maps로 만듦
  - Fusion — Residual Convolutional Units를 사용해 feature maps를 fuse하고, upsampling하여 예측값 생성

  모든 stages에서 global receptive field를 가지고 있어 globally coherent한 깊이 예측 가능
- Monocular depth estimation 뿐만 아니라 Semantic segmentation에서도 적용 가능



DPT network architecture

Depth Estimation



Semantic Segmetation

Reassemble Module

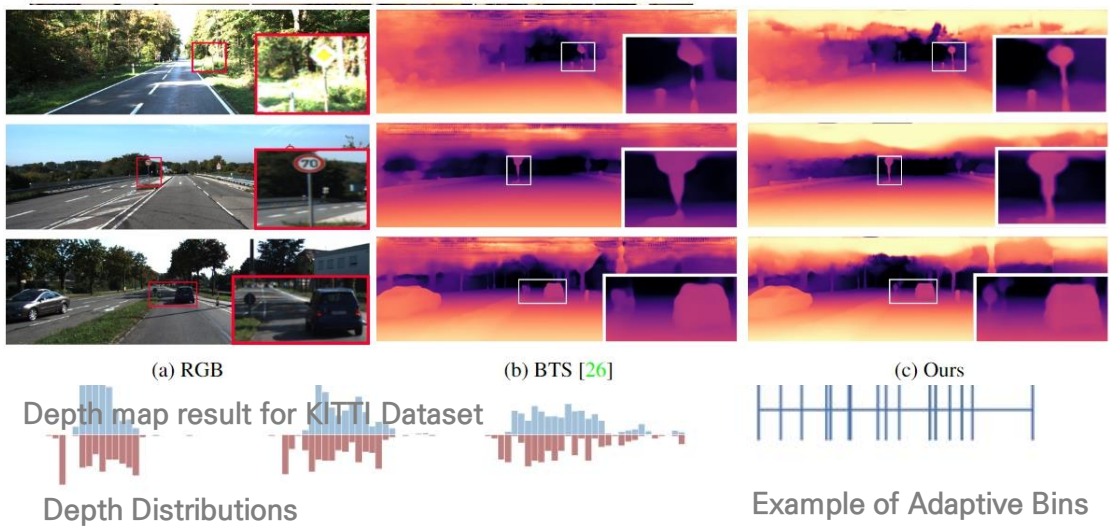
Fusion Module

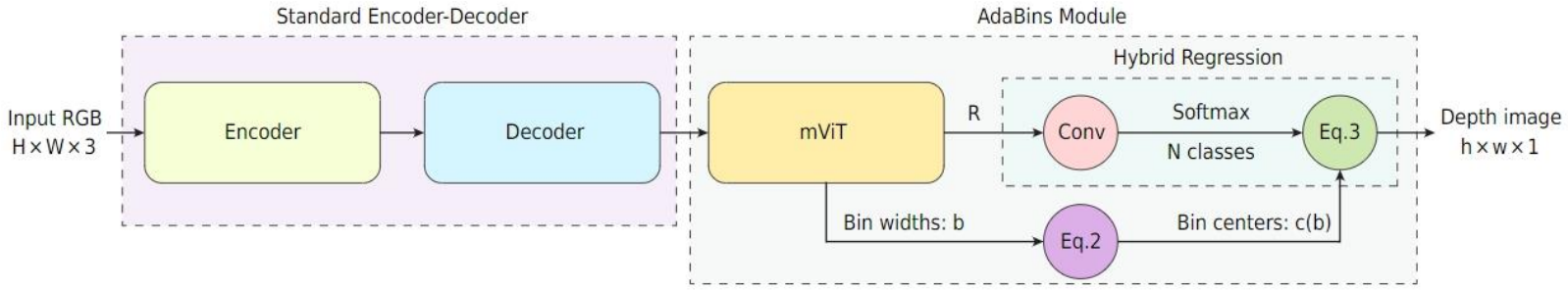
Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 12179-12188).

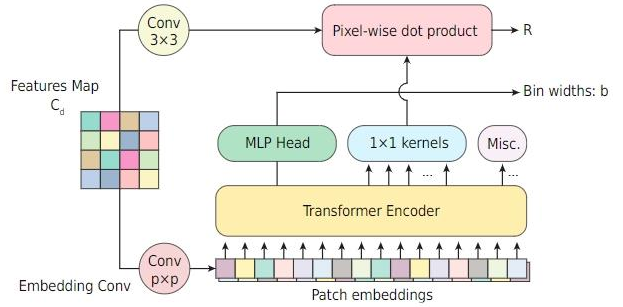
03 단일 이미지 깊이 추정 Monocular Depth Estimation

BTS (Lee et al., 2019)	DPT (Ranftl et al., 2021)	AdaBins (Bhat et al., 2021)	DepthFormer (Li et al., 2022)	MonoDepth (Godard et al., 2017)

- Supervised Learning 방식, Regression이 아닌 Classification으로 접근
- Encoder (feature extractor) – Decoder(upsampling) Block
- AdaBins (Adaptive Bin-width Estimator) 모듈 제안
  - Mini-ViT를 사용해 bin width b 와 Range-Attention Maps $\mathcal{R}$ 계산
  - 각 scene의 features에 따라 bins가 적응적으로 바뀜(Adaptive)
  - Classification을 사용하여, depth values의 discretization을 만듦
  - Bin centers의 선형 결합으로 최종 깊이 예측값 생성







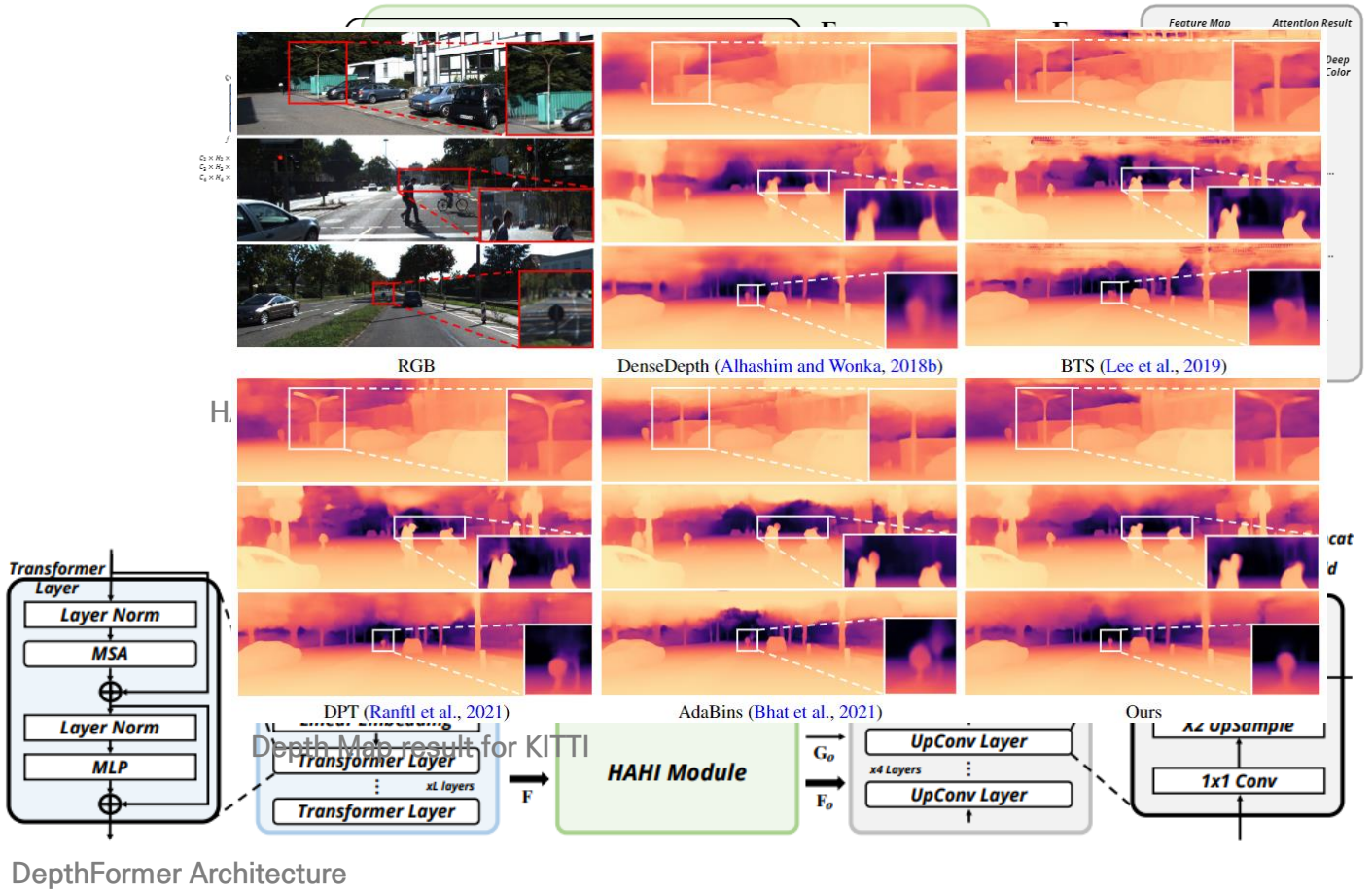
(*Left*) AdaBins Architecture Overview (*Right*) mini-ViT Overview

Bhat, S. F., Alhashim, I., & Wonka, P. (2021). Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4009-4018).

03 단일 이미지 깊이 추정 Monocular Depth Estimation

BTS (Lee et al., 2019)	DPT (Ranftl et al., 2021)	AdaBins (Bhat et al., 2021)	DepthFormer (Li et al., 2022)	MonoDepth (Godard et al., 2017)

- Supervised Learning 방식, Encoder-Decoder 사용
- Encoder = Transformer Branch + Convolution Branch
  - 두 Branch로 나누어 Long-range correlation과 Local information을 탐색하고, Spatial Inductive Bias 결핍을 보완
  - Swin Transformer(ViT)를 사용해 Hierarchical feature 제공 및 연산 복잡도 개선
- 두 Branch를 결합하기 위한 HAHI(Hierarchical Aggregation and Heterogeneous Interaction) 모듈을 사용해 feature와 model affinity의 enhancement 도모



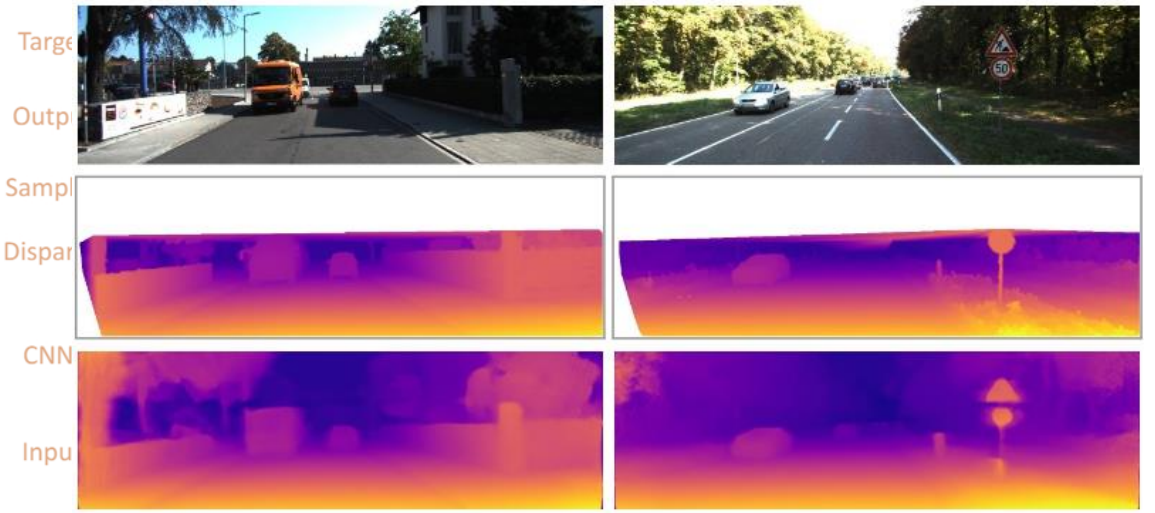
DepthFormer Architecture

Li, Z., Chen, Z., Liu, X., & Jiang, J. (2022). DepthFormer: Exploiting Long-Range Correlation and Local Information for Accurate Monocular Depth Estimation. *arXiv preprint arXiv:2203.14211.*

03 단일 이미지 깊이 추정 Monocular Depth Estimation

BTS (Lee et al., 2019)	DPT (Ranftl et al., 2021)	AdaBins (Bhat et al., 2021)	DepthFormer (Li et al., 2022)	MonoDepth (Godard et al., 2017)

- **Unsupervised** Learning 방식
- Aligned Ground Truth depth 정보 대신, Binocular Stereo data를 사용함
  - Rectified stereo image pair 간의 pixel-level correspondence를 예측
- Depth Estimation 문제를 Reconstruction 문제로 여김
  - 한 이미지에서 다른 이미지로 reconstruct함
    - 왼쪽 이미지에서 depth를 추정하여 오른쪽 이미지를 reconstruct
    - Disparity를 만들고 이를 모델이 학습함
- 새 Training loss를 제안해, 추정한 depth map 간 consistency enforcement



Sampling Method (*Left*) and Loss module (*Right*)

Godard, C., Mac Aodha, O., & Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 270-279).

03 단일 이미지 깊이 추정 Monocular Depth Estimation

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가

Model Selection

DepthFormer

Supervised vs Unsupervised
ACE Lab 데이터는 단안 카메라에서 얻은 single image이므로 supervised 선택

Performance
KITTI Eigen Split Benchmark(야외 도로상황)에서 AbsRel 기준 3위(0.052) 기록

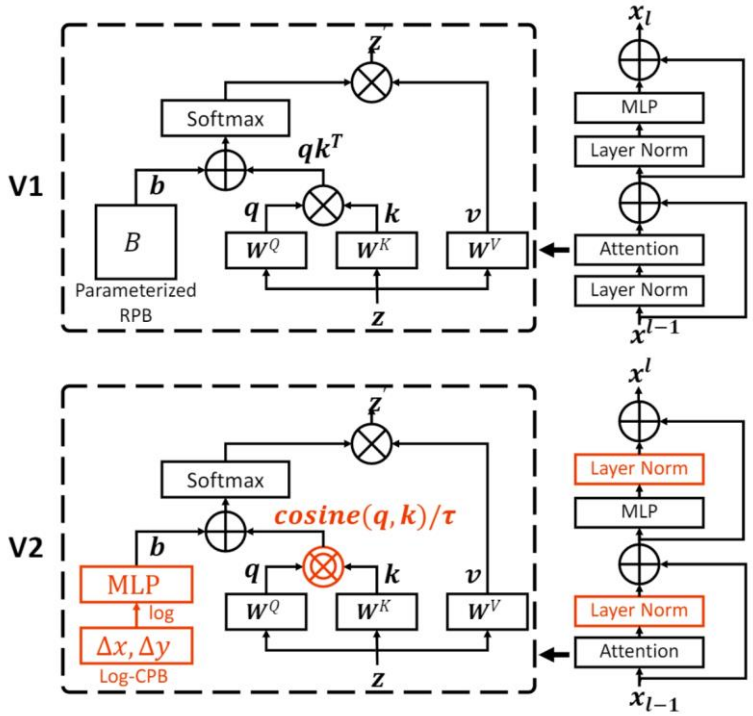
Usage
KITTI Dataset 및 ACE Lab 데이터를 적용하여 모델에서 출력을 얻기에 성공

Model Improvement
기존에 Swin Transformer를 사용하고 있던 ViT를 다른 Transformer로 변경 가능성

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가

Model Modification

- 기존의 DepthFormer에서는 Swin Transformer V1을 사용
- 최근 발표된 Swin Transfomer V2의 특징
  - residual-post-norm 및 scaled cosine attention 사용
    - ➡ 모델 용량이 깊어짐에 따라 활성함수의 값의 급증 현상 완화
  - 로그 공간의 continuous position bias를 사용
    - ➡ window 간 부드러운 relative position bias의 전이를 도움
- Swin Transformer를 V2로 변경함으로써 더 좋은 성능의 pre-trained model 이용 가능



Swin Transformer V1과 V2의 비교

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., ... & Guo, B. (2022). Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12009-12019).

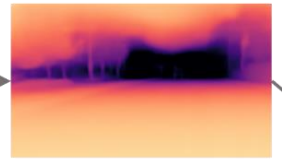
04 추정 결과 평가 Evaluation with LiDAR

01 사전 준비 ✓ — 02 LiDAR 데이터 변환 ✓ — 03 단일 이미지 깊이 추정 ✓ — 04 추정 결과 평가 ○

Raw Image

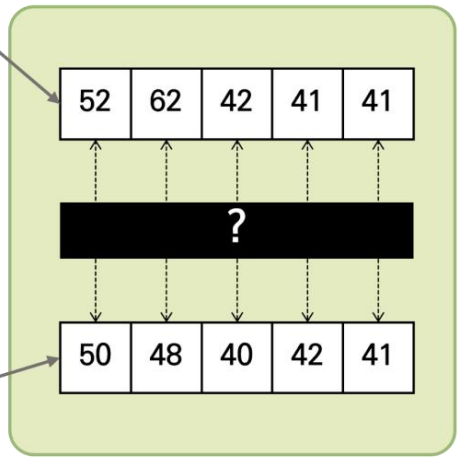
Downsampled
& Undistorted Image

Estimation Model



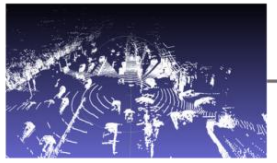
Depth Map

Evaluation Algorithm



Evaluation Result

RMSE	6.315
RMSElog	0.210
AbsRel	0.412
SqRel	1.388
log10	0.051
⋮	

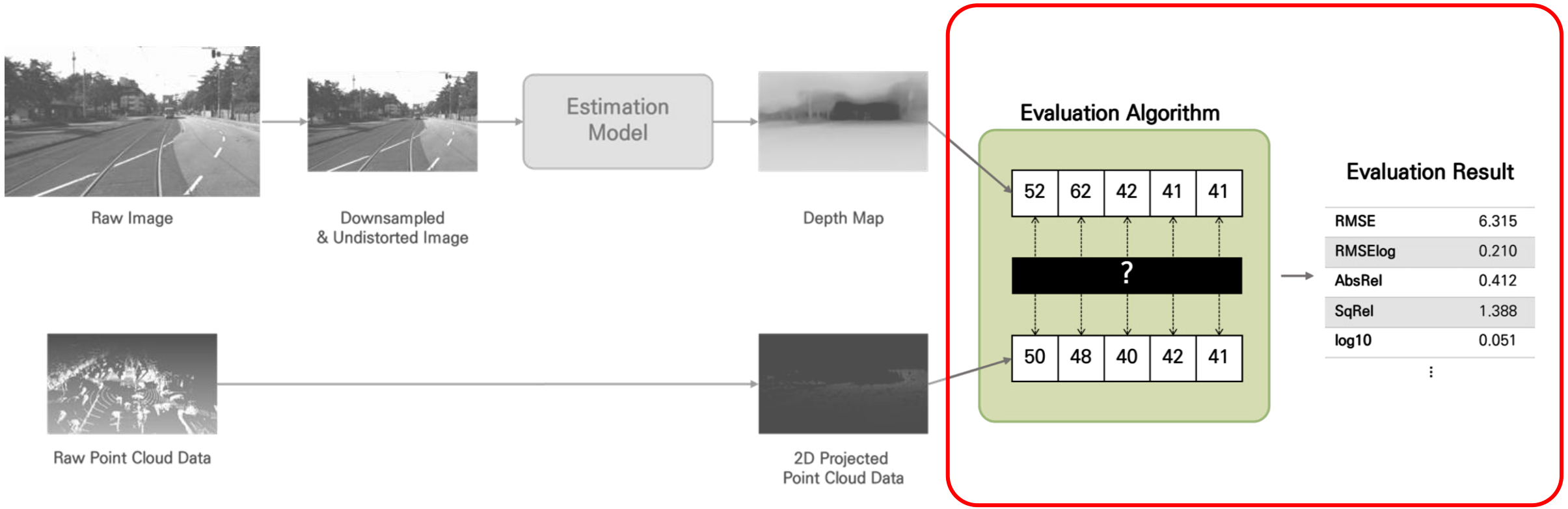


Raw Point Cloud Data

2D Projected
Point Cloud Data

04 추정 결과 평가 Evaluation with LiDAR

01 사전 준비 ✓ — 02 LiDAR 데이터 변환 ✓ — 03 단일 이미지 깊이 추정 ✓ — 04 추정 결과 평가 ○



04 추정 결과 평가 Evaluation with LiDAR

RMSE / RMSElog

Root Mean Squared Error (Linear) / (Log space)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^*)^2}$$

$$\text{RMSElog} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log y_i - \log y_i^*)^2}$$

AbsRel / SqRel

Absolute / Sqared Relative Error

$$\text{AbsRel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - y_i^*|}{y_i^*}$$

$$\text{SqRel} = \frac{1}{N}\sum_{i=1}^{N}\frac{\|y_i - y_i^*\|^2}{y_i^*}$$

SILog

Scale-invariant Mean Squared Log Error

$$\text{SILog} = \frac{1}{2N}\sum_{i=1}^{N}\left(\log y_i - \log y_i^* + \alpha\left(y, y^*\right)\right)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N} d_i^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N} d_i\right)^2$$

이때,

$$\alpha\left(y, y^*\right) = \frac{1}{n}\sum_{i=1}^{N}\left(\log y_i^* - \log y_i\right), d_i = \log y_i - \log y_i^*$$

Threshold

$$\delta = \%\ \text{of}\ y_i \quad \text{s.t.} \quad \max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) = \delta < thr$$

단, δ_k일 때 $thr = 1.25^k \ \ (k = 1, 2, 3)$

Log10

Mean Log_{10} Error

$$\log_{10} = \frac{1}{N}\sum_{i=1}^{N}\left|\log_{10} y_i^* - \log_{10} y_i\right|$$

04 추정 결과 평가 Evaluation with LiDAR

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가

no.	thr1	thr2	thr3	RMSE	RMSElog	SILog	AbsRel	SqRel	log_10
0001	0.978	0.989	0.995	10.040	0.457	45.433	0.314	3.066	0.144
0002	0.978	0.990	0.995	10.667	0.465	46.456	0.336	3.493	0.143
0003	0.977	0.990	0.995	10.777	0.456	45.599	0.323	3.320	0.145
0004	0.975	0.990	0.996	10.895	0.459	45.907	0.330	3.388	0.148
0005	0.975	0.989	0.995	11.311	0.470	46.885	0.324	3.430	0.149

...

no.	thr1	thr2	thr3	RMSE	RMSElog	SILog	AbsRel	SqRel	log_10
0900	0.979	0.992	0.997	7.660	0.373	37.332	0.280	2.074	0.125
0901	0.975	0.992	0.996	7.608	0.383	38.179	0.302	2.143	0.132
0902	0.974	0.991	0.996	7.408	0.390	38.694	0.316	2.267	0.134
0903	0.972	0.991	0.996	7.673	0.406	40.068	0.343	2.520	0.143
0904	0.972	0.990	0.996	8.012	0.417	41.176	0.351	2.639	0.147
0905	0.973	0.991	0.996	8.361	0.423	42.209	0.339	2.736	0.147
0906	0.974	0.989	0.995	8.736	0.440	43.880	0.333	2.895	0.150
0907	0.972	0.988	0.994	8.874	0.463	45.835	0.351	3.140	0.160

Final Report

Bag	thr1	thr2	thr3	RMSE	RMSElog	SILog	AbsRel	SqRel	log_10
01	0.976	0.988	0.993	12.063	0.541	52.444	0.353	4.332	0.171

각 프레임(이미지) 당 평가 결과

지표 별 평균

데이터셋 전체 평가 결과

Part 03

실험 결과

(1) 사전 준비
(2) LiDAR 데이터 변환
(3) 단일 이미지 깊이 추정
(4) 추정 결과 평가

01 ACE Lab Dataset: 사전 준비

Basic Info

- **수집 지역 |** 서울 강남 일대
- **사용 센서 |** 80ch 3D LiDAR, HD Camera

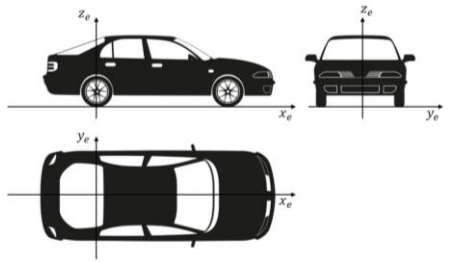
Format & Size : ROS bag X 7개

No.	Duration (s)	Size (GB)
1	90.0	9.5
2	37.1	3.9
3	49.0	5.2
4	44.6	4.7
5	44.2	4.6
6	53.7	5.6
7	79.0	8.4

- **Image |** PNG
  sensor_msgs/Image
  about 30 Hz
- **Point cloud |** NPY
  Sensor_msgs/PointCloud2
  about 10 Hz

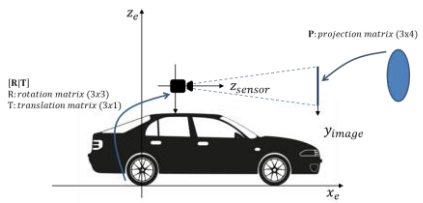
➡ **LiDAR의 timestamp에 맞춰 Camera topic 저장**

Calibration Info

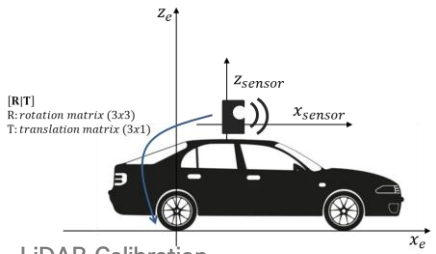


Ego Vehicle Coordinate

Sensor	Notation	Descrip.
Camera	D	Distortion coefficients
	K	Intrinsic Matrix
	P	Projection Matrix
	R	Rotation Matrix
	t	Translation vector
	size	Size of image
LiDAR	R	Rotation Matrix
	t	Translation vector



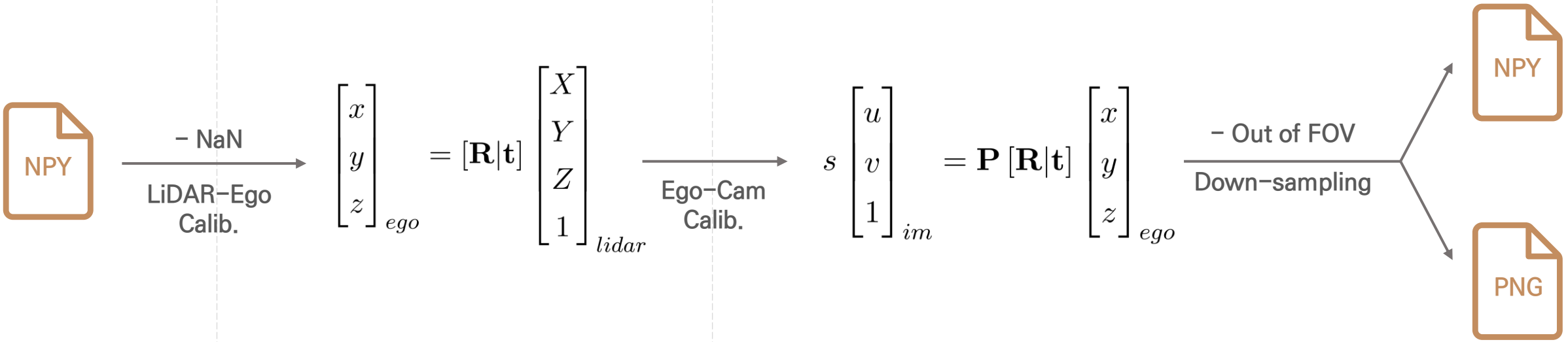
Camera Calibration



LiDAR Calibration

02 ACE Lab Dataset: LiDAR 데이터 변환

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가



LiDAR Coord.

Vehicle Coord.

Camera Coord.

03 ACE Lab Dataset: 단일 이미지 깊이 추정

01 사전 준비 ✓ — 02 LiDAR 데이터 변환 ✓ — 03 단일 이미지 깊이 추정 ○ — 04 추정 결과 평가 ○

Down-Sampling

- ACE Lab 제공 rosbag 내 이미지 원본 데이터는 DepthFormer의 메모리 부족 이슈 발생
- 가로와 세로 크기를 ½배 하여 down-sampling
- 이미지는 OpenCV의 cv2.resize()로 진행
  Point Cloud Data는 투영 이후 (u, v) 좌표를 ½배

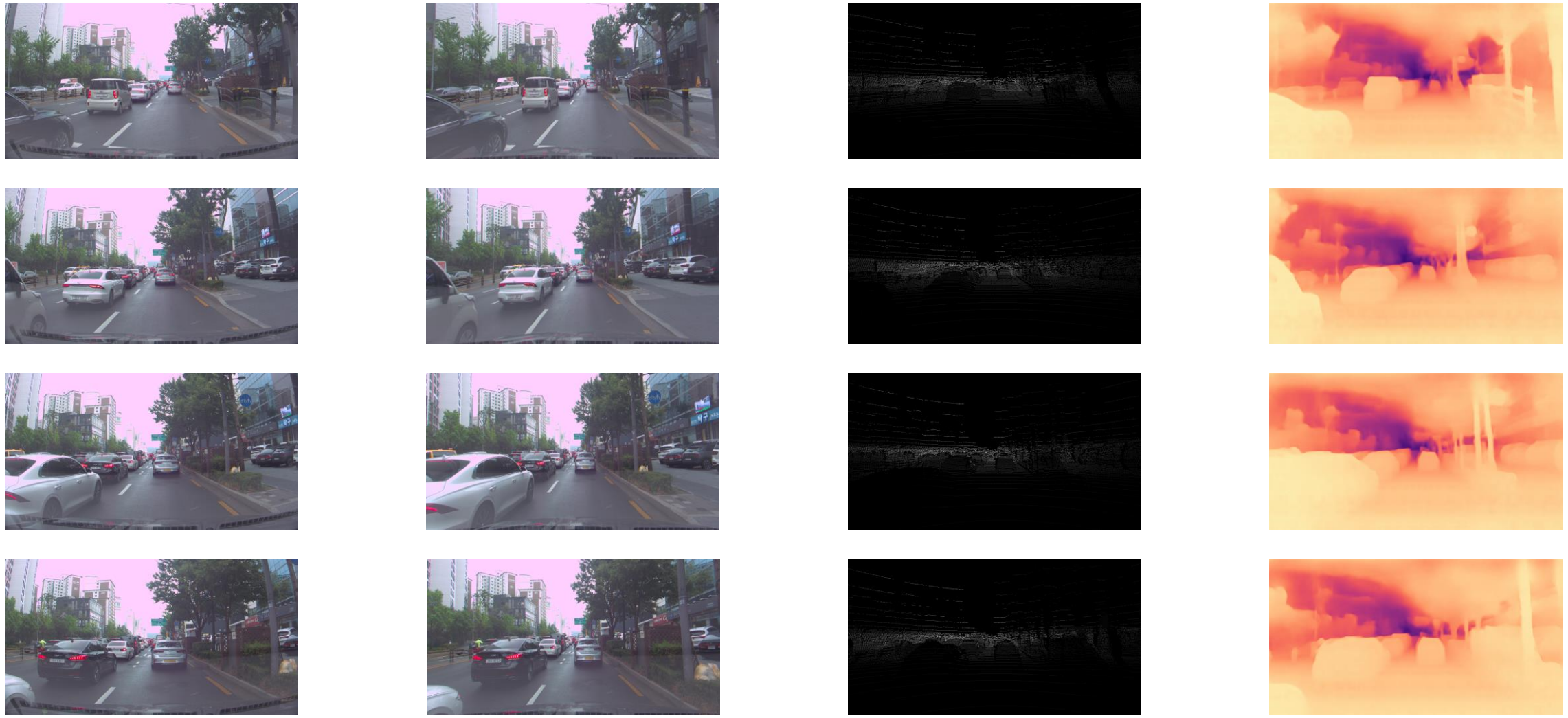
강남 일대 도심 도로와
KITTI 수집 장소(유럽 교외) 간 환경 차이 존재

vs 야외 환경이라는 공통점이 있음

Model Training

- KITTI Eigen Split dataset 사용
  - Stereo Camera & 3D Laser Scanner
  - 깊이 추정을 위하여 가공된 ground truth 정보 제공
- 훈련용 26K, 검증용 697개의 이미지 사용
- 학습 환경
  - Nvidia GTX 1080 ti 2 way: 약 12h (38,000 iter)
  - Nvidia RTX 3080 12GB: 약 15h (120,000 iter)
- 다만 Swin Transformer V2(V2-B)를 적용한 모델의 추정 정확도가 V1에 비해 낮아 추정에는 V1 사용

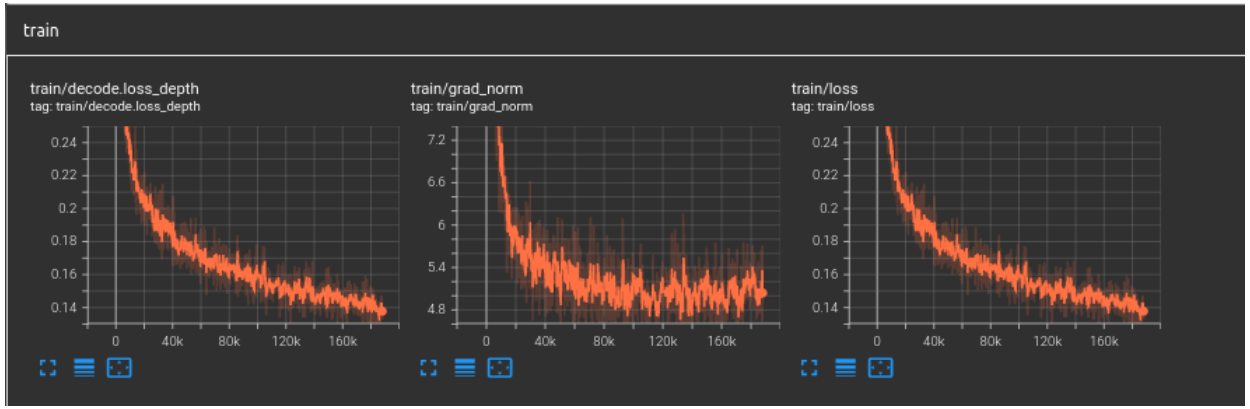
03 ACE Lab Dataset: 단일 이미지 깊이 추정



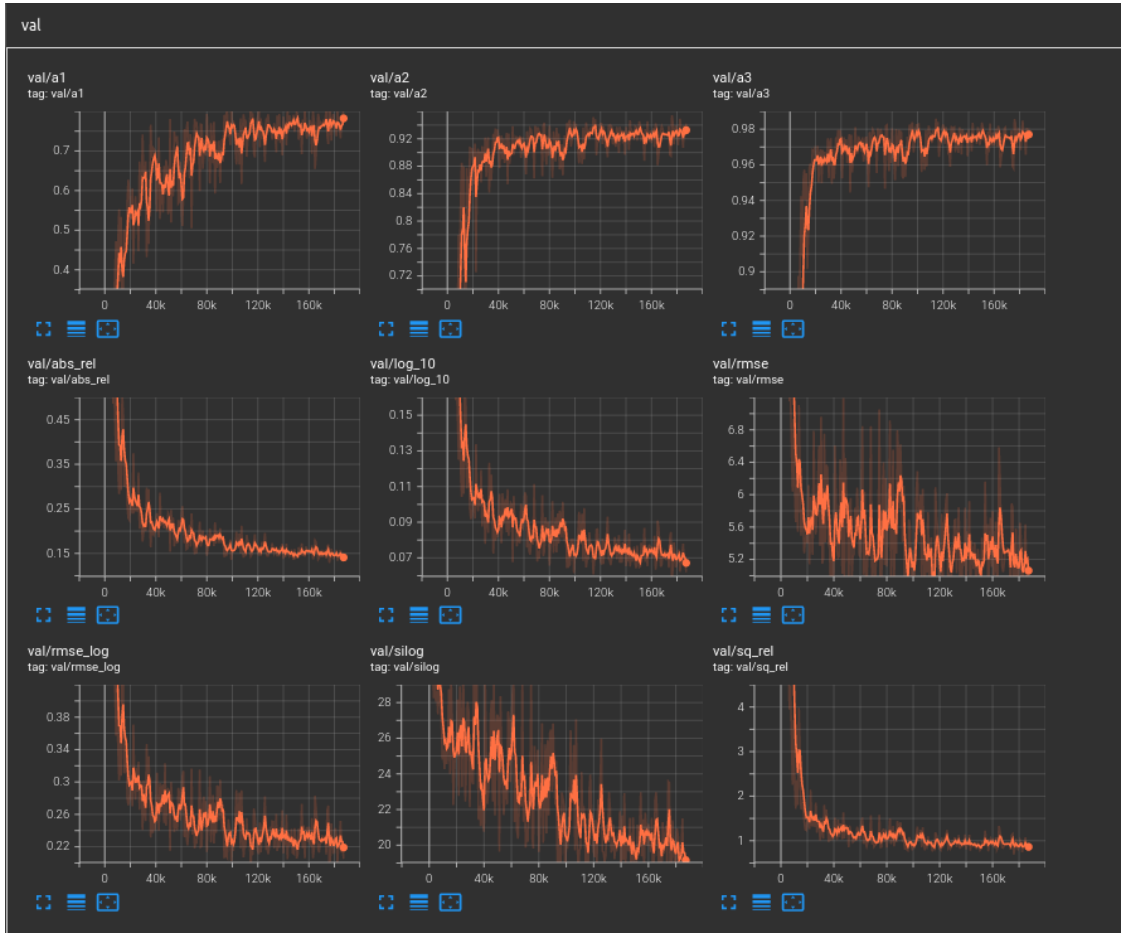
Raw Image | Undistorted Image | Point Cloud (GT) | Depth Map

03 ACE Lab Dataset: 단일 이미지 깊이 추정

01 사전 준비 — 02 LiDAR 데이터 변환 — 03 단일 이미지 깊이 추정 — 04 추정 결과 평가



변경한 모델의 학습 중 지표 결과 변화 추이



04 ACE Lab Dataset: 추정 결과 평가

01 사전 준비 ✓ — 02 LiDAR 데이터 변환 ✓ — 03 단일 이미지 깊이 추정 ✓ — 04 추정 결과 평가 ○

bag file	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	RMSE ↓	RMSElog ↓	SILog ↓	AbsRel ↓	SqRel ↓	log_10 ↓
gangnam_1.bag	0.583	0.787	0.872	12.080	0.542	52.714	0.357	4.406	0.172
gangnam_2.bag	0.558	0.843	0.938	9.953	0.408	39.977	0.320	3.249	0.140
gangnam_3.bag	0.663	0.881	0.940	9.381	0.446	43.600	0.290	2.668	0.142
gangnam_4.bag	0.572	0.831	0.930	11.396	0.463	45.319	0.333	3.891	0.154
gangnam_5.bag	0.556	0.728	0.831	13.487	0.582	54.104	0.355	4.999	0.187
gangnam_6.bag	0.546	0.842	0.934	11.132	0.398	38.742	0.301	3.437	0.135
gangnam_7.bag	0.665	0.805	0.874	16.061	0.497	47.352	0.263	4.907	0.139

각 bag file 별 추정치 검증 결과

Part 04

결론

실험 결과 해석
추가 학습 및 연구
프로젝트 회고

01 실험 결과 해석

bag file	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	RMSE ↓	RMSElog ↓	SILog ↓	AbsRel ↓	SqRel ↓	log_10 ↓
gangnam_1.bag	0.583	0.787	0.872	12.080	0.542	52.714	0.357	4.406	0.172
gangnam_2.bag	0.558	0.843	0.938	9.953	0.408	39.977	0.320	3.249	0.140
gangnam_3.bag	0.663	0.881	0.940	9.381	0.446	43.600	0.290	2.668	0.142
gangnam_4.bag	0.572	0.831	0.930	11.396	0.463	45.319	0.333	3.891	0.154
gangnam_5.bag	0.556	0.728	0.831	13.487	0.582	54.104	0.355	4.999	0.187
gangnam_6.bag	0.546	0.842	0.934	11.132	0.398	38.742	0.301	3.437	0.135
gangnam_7.bag	0.665	0.805	0.874	16.061	0.497	47.352	0.263	4.907	0.139

Ground Truth의 정확도

- Point cloud data가 Image에 비해 sparse함
- 투영 과정의 투영 오차 (소수점 버림 등)
- 애초에 정답값이 잘못되었는지 추정값이 잘못되었는지 판단 불가

Down-Sampling의 영향

- 보간법 이용함에도 발생하는 정보의 손실
- Point Cloud Data 투영 위치를 단순히 ½ 연산함

훈련용 / 검증용 데이터셋 차이

- 훈련용 데이터셋:
- KITTI dataset (유럽, 교외)

검증용 데이터셋:

ACE Lab dataset (서울 강남, 도심)

02 추가 학습 및 연구

01
Convolution 연산의 한계

Convolution 연산은
spatial inductive bias 부족

원거리 객체의 깊이값
추정 정확도 하락 우려

깊이 range 별 평가

DepthFormer 저자들과 같이
깊이 range를 나누어
추정 결과 평가를 따로 진행

모델이 근/원거리 거리 추정에
약점을 보이는지 확인 가능

02
훈련/검증 데이터셋 상이

훈련 데이터셋(KITTI)은
서양 교외 지역에서 구축됨

검증 데이터셋(ACE Lab)은
한국 도시 도로 상황 대상

다른 dataset 확보

한국이나 도시 도로 교통 상황을
포함한 Dataset으로
모델의 추가적 학습을 진행해
정확도 향상을 도모

03
Ground Truth 확보 비용

Supervised Learning 방식은
이미지 픽셀에 대응하는
Ground truth 데이터를
확보에 큰 비용 소요

Unsupervised

Stereo camera에서 얻은
image pair를 사용하며
ground truth가 불필요한
Unsupervised learning 모델
(e.g. MonoDepth)

04
Point Cloud Data 이상치

LiDAR에서 얻은
point cloud data는
노이즈 및 이상치를 포함

Filter 추가

Gaussian 분포를 이용하여
Point cloud data 중
이상치를 제거하고
정확한 ground truth 확보
(Statistical Outlier Removal)

03 프로젝트 회고: 안정성 및 협업 효율성 확보

CI(Continuous Integration) with GitHub Actions

- Python의 Code Formatter인 Black과 isort 적용
- 자동으로 code format을 검사하고 교정하여 저장

문서 및 함수마다 Documentation 작성

- 함수의 용도 및 반환형, 파라미터 설명 및 자료형 등
- 코드 파악을 용이하게 하고 협업 효율성 확보

Unit Test 활용

- Python의 Unit Test 라이브러리 PyTest 활용
- 함수 구현 중 예외사항, 안정적 작동 여부 확인

PyTest 를 이용한 Unit Test 결과 출력 예시

```
platform win32 -- Python 3.9.7, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- C:\Users\lumos\anaconda3\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lumos\Desktop\depth-estimation-with-lidar, configfile: pytest.ini
plugins: anyio-2.2.0
collected 4 items

test/test_code.py::TestCalib::test_convert_pcd_to_xyz
INFO     2022-07-08 13:41:36 logger::test_code.py:test_convert_pcd_to_xyz:93: test_convert_pcd_to_xyz
PASSED
test/test_code.py::TestCalib::test_in_image
INFO     2022-07-08 13:41:36 logger::test_code.py:test_in_image:98: test_in_image
PASSED
test/test_code.py::TestCalib::test_projection
INFO     2022-07-08 13:41:36 logger::test_code.py:test_projection:103: test_projection
INFO     2022-07-08 13:41:36 logger::test_code.py:test_projection:108: input point(xyz) : [-10.85400009 -14.15200043  -1.55200005]
INFO     2022-07-08 13:41:36 logger::test_code.py:test_projection:109: output point(xy) : [-170.  455.]
INFO     2022-07-08 13:41:36 logger::test_code.py:test_projection:110: depth: -12.345935455122923
FAILED
test/test_code.py::TestCalib::test_project_all_points
INFO     2022-07-08 13:41:36 logger::test_code.py:test_project_all_points:116: test_project_all_points
        logger.info("output point(xy)\t: " + str(projected_pos[:-1]))
        logger.info("depth: " + str(projected_pos[-1]))

>       assert in_image(projected_pos, cam_calib["size"])  # 범위 벗어나면 Fail
E       AssertionError: assert False
E        +  where False = in_image(array([-170.        ,  455.        , -12.34593546]), {'height': 1086, 'width': 2040})

test\test_code.py:112: AssertionError
-------------------------------- Captured log call --------------------------------
INFO     logger:test_code.py:103 test_projection
INFO     logger:test_code.py:108 input point(xyz)       : [-10.85400009 -14.15200043  -1.55200005]
INFO     logger:test_code.py:109 output point(xy)       : [-170.  455.]
INFO     logger:test_code.py:110 depth: -12.345935455122923
=========================== short test summary info ===========================
FAILED test/test_code.py::TestCalib::test_projection - AssertionError: assert False
========================= 1 failed, 3 passed in 9.22s =========================
```

함수 Documentation 예시

```
134             * KITTI Calib
135             * cam to cam: 4개의 카메라가 있어 서로 바뀌는 calib 정보가 있음
136             * velo to cam: 라이다를 카메라(0번)로 바로 투영
137         """
138         lidar = np.append(lidar_point, [1], axis=0)  # [x(forward) y(left) z(up) 1]
139         lidar = np.transpose(lidar)  # (4, )
140
141         if dataset == "ace":
142             # lidar coordinate -> ego coordinate
143             ego = np.concatenate([lidar_calib["R"], lidar_calib["t"]], axis=1)  # [R|t] matrix
144             ego = np.concatenate([ego, [[0, 0, 0, 1]]], axis=0)  # (3, 4) -> (4, 4)
145             ego = np.matmul(ego, lidar)  # [R|t] X [x y z 1]^T -> [x(forward) y(left) z(up)] (3,)
216             # cam_calib["S"] = calib_info["size"] # image size {'height': 1086, 'width': 2040}
217             # TODO ace의 K와 shape 맞나 확인
218             calib_info = yaml.load(f, Loader=yaml.FullLoader)
219 -           cam_calib["P"] = np.reshape([float(x) for x in calib_info["P_rect_00"].split(' ')], (3, 4)) # projection matrix (3, 4)
220 -           cam_calib["R"] = np.reshape([float(x) for x in calib_info["R_rect_00"].split(' ')], (3, 3)) # rotation matrix (3, 3)
221 -           cam_calib["T"] = np.reshape([float(x) for x in calib_info["T_00"].split(' ')], (3, 1)) # translation (3, 1) # TODO ace의 T와 shape 맞나 확인
222 +           cam_calib["P"] = np.reshape(
223 +               [float(x) for x in calib_info["P_rect_00"].split(" ")], (3, 4)
224 +           ) # projection matrix (3, 4)
225 +           cam_calib["R"] = np.reshape(
226 +               [float(x) for x in calib_info["R_rect_00"].split(" ")], (3, 3)
227 +           ) # rotation matrix (3, 3)
228 +           cam_calib["T"] = np.reshape(
229 +               [float(x) for x in calib_info["T_00"].split(" ")], (1, 1)
230 +           ) # translation (3, 1) # TODO ace의 T와 shape 맞나 확인
231
232         return cam_calib
```

Code Formatter 적용 결과 상세 출력

03 프로젝트 회고: 소감

한은기

Project Manager & Architecture

- 수많은 선행 연구를 조사하고, 그중 프로젝트에 적합한 내용을 선별하는 값진 경험이었다.
- 프로젝트 수행을 위한 프로세스를 직접 설계하고, 이를 모듈화시킬 수 있었다.
- 지난 프로젝트들에서 하지 못해 아쉬움이 남았던 코드 포매팅 자동화, unit test 활용 등을 이번 프로젝트에서 이루었다.

이현진

Evaluation with LiDAR

- LiDAR 데이터가 매우 많아 CPU 연산 수행 시간이 길었다. 따라서 voxel grid 등 down-sampling의 이유를 체감했다.
- 요구조건에 맞게 튜닝하는 데도 많은 어려움이 존재함을 배웠다.
- 기존 라이브러리나 코드를 활용하기보다는, 직접 구현하는 과정에서 코딩 공부를 많이 하게 되었다.

유희평

Depth Estimation

- 복잡한 구조의 모델을 이해하고 다루려 코드와 논문을 찾아 읽으며, 딥러닝 기법과 용어를 이해하고 코드에 적용되는 방식을 이해할 수 있었다.
- 모델 수정을 더 일찍 끝내고, 학습을 위한 시간을 더 확보했으면 좋았을 것이다.
- 모델 성능 개선의 테크닉보다는 구체적 이유를 정확히 알고 싶어, 설명 가능한 AI에 관심이 생겼다.

경청해주셔서 감사합니다!

Team 씨리스랩

2022.07.15.

자율주행 데브코스 3기 최종 프로젝트 발표