

Monocular Depth Estimation and Evaluation using LiDAR

Team 더:위사냥(hunting higher)

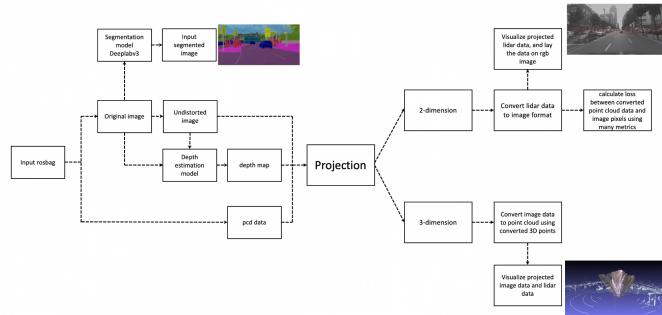
요약

미리 학습된 깊이 추정 모델(Depth Former, DPT, BTS, Adabins)을 통해 생성된 Depth Map과 왜곡이 보정된 Lidar 데이터를 통해 추출한 depth ground truth 를 선정된 평가 방법으로 오차를 계산하고, 결과 비교를 통해 Custom 데이터셋에 최적의 모델을 제안한다. 또한, 데이터를 시각화해볼 수 있는 모듈을 함께 제안한다.

프로젝트 소개

직접 개발한 깊이 추정 알고리즘을 통해 추출된 깊이 데이터를, 관측한 LiDAR 데이터와 비교하여 알고리즘을 평가하고자 한다. 자신만의 평가 알고리즘을 개발하고, 깊이 추정 모델을 활용한 추가적인 작업을 진행한다.

1. 프로젝트 개요



먼저 rosbag 파일에서 ‘sensor_msgs/Image’ 토픽과 ‘rslidar_points’ 토픽을 각각 png, pcd 형태로 변환한다. 그리고, 모든 이미지는 카메라 렌즈에 의한 왜곡이 존재하므로, calibration을 통해 카메라의 왜곡을 보정해준다. 왜곡이 보정된 이미지를 깊이 추정 모델에 넣어서 픽셀 단위의 depth map을 얻는다. depth map은 RGBA 4채널로 구성되어 있는데, RGB는 동일한 값을 가진다. 추후 depth map 이미지를 불러올 때는 1채널로 불러오게 된다.

그 후 왜곡 보정된 이미지와 depth map, LiDAR 센서로 측정한 point cloud data를 통해 차원 투영을 수행한다. 차원 투영이란 이미지는 Depth 정보가 손실된 2차원 데이터이고, LiDAR는 x,y,z 3차원의 데이터가 담겨져 있기 때문에, 두 데이터 간의 정보 격차가 존재한다. 이를 calibration을 통해 얻은 intrinsic matrix, extrinsic matrix,

projection matrix를 통해 카메라 좌표계로 변환하고, rotation matrix, translation matrix를 통해 실제 좌표계로 변환한다.

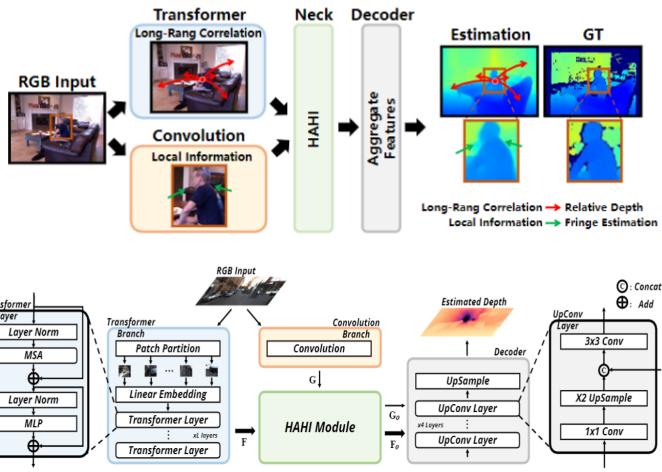
차원 투영은 2가지 방식으로 수행했다. 하나는 point cloud data를 2차원 형태로 투영하는 것이고, 다른 하나는 RGB image data와 depth map을 가지고, 3차원 형태로 투영하는 것이다. 2차원의 경우 LiDAR 데이터를 2차원 이미지 형태로 변환하고, 이를 이미지 위에 표시하여 시각화했고, 3차원의 경우 이미지 데이터를 depth map을 활용하여 3차원으로 변환하여 LiDAR 데이터와 함께 표시하여 시각화했다.

마지막으로 알고리즘을 평가하기 위해 2차원으로 변환한 LiDAR 데이터를 깊이 추정 모델로 추출한 depth map을 다양한 기법으로 오차를 구한다. 구하는 방법은 가장 통용적인 RMSE, 변동성에 강인한 RMSElog, 임계값을 활용하여 적절한 값의 개수를 구하는 δ1, δ2, δ3 등을 사용했다. 추가적으로 RGB 이미지 대신 segmentation을 수행한 이미지를 입력으로 사용할 때 추가적인 이점이 있을 것으로 예상하여 사용해보았으나 가시성 이외의 이점을 활용하기 어려워 사용하지 않았다.

2. 모델 소개

2.1. DepthFormer

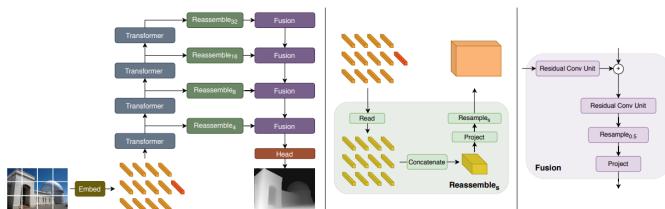
DepthFormer(DepthFormer: Exploiting Long-Range Correlation and Local Information for Accurate Monocular Depth Estimation)[1] 의 architecture는 다음과 같다.



Depth Former는 global 정보를 위한 transformer layer와 local 정보를 위한 convolution branch를 사용했다. transformer와 CNN의 융합을 위해 HAHI module을 사용했다. HAHI module은 transformer + CNN에 대한 HA(hierarchical aggregation)와 attention 자동화를 위한 HI(heterogeneous interaction)로 구성되어 있다. HAHI는 입력 크기가 고정되어 있지 않으며 LIDAR와 융합한다면 성능이 좋아진다고 한다. transformer의 경우 Swin-transformer layer 사용하고, CNN의 경우는 ResNet 50을 사용한다.

2.2. DPT

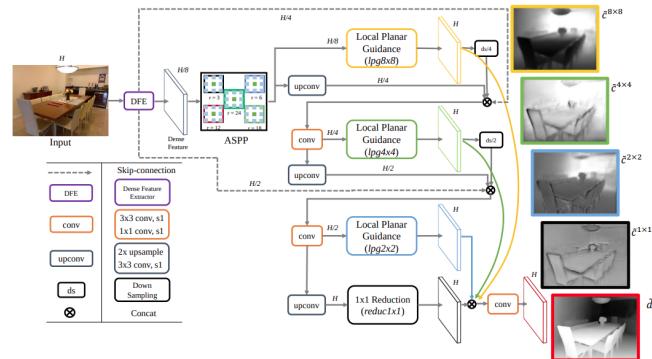
DPT(dense prediction transformer)의 architecture는 아래와 같다.



DPT는 ResNet50을 적용하여 vision transformer를 backbone으로 활용한 인코더와 convolutional 디코더 구조로 되어있다. 이러한 구조는 dense한 구조, 대용량 데이터 처리 작업에서 좋은 성능을 보여준다. [2]

2.3. BTS

BTS(From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation) 의 architecture는 아래와 같다.



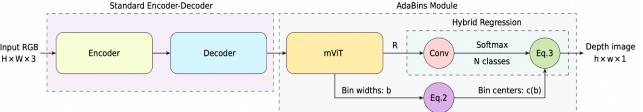
BTS 모델은 dense feature extractor 위한 인코더와 prediction을 위한 디코더 구조로 되어있다.

인코딩 과정에서 ResNet-101, ResNext-101, DenseNet-161 backbone을 사용해 추출된 feature에 atrous spatial pyramid pooling layer(ASPP) 과정을 거친다. ASPP는 Segmentation 분야에서 Detail을 살려주기 위한 기법으로 많이 사용되며, Multi-scale에 잘 대응할 수 있는 Pooling 기법이다.

디코딩 과정에서는 Local Planar Guidance Layer(LPG layer) 방식이 적용된다. LPG layer를 이용하여 단계적으로 Upsampling을 하면서 reconstruction을 진행한다. [3]

2.4. Adabins

Adabins(Adabins:Depth Estimation using Adaptive Bins) [4] 의 architecture는 다음과 같다.



간격 ‘D’를 N개의 bin으로 구별할 수 있는 Adaptive bins를 처음 제안했다. Adabins에서 bin의 넓이 b 는 각 이미지마다 알맞게 계산된다. 깊이 값을 부드럽게 추정하기 위한 bin의 중앙의 선형 결합을 진행한다. attention block을 생성하여 더 나은 결과를 예측하도록 한다. Adabins architecture를 가능한 한 단순하게 구축하기 위하여 EfficientNet B5를 인코더의 백본으로 사용하는 최신 인코더디코더를 기반으로 구축하였다.

2.5. 모델 설정

모델을 설정할 때 가장 먼저 보았던 것은 편리성이다. 모델을 사용함에 있어서 실행하기 편하고, 응용하기 편한 모델이 가장 사용하기 적합하다 판단했다. 특히 깃허브에 설명서가 잘 적혀있는지, 코드가 모듈화가 잘 되어 있는지 확인했다.

또한, GT data가 존재하면서 강남 야외 환경과 비슷한 데이터를 제공해주는 KITTI data를 이용하고 한정된 컴퓨터 자원을 해결하고자 Supervised Monocular Depth Estimation 모델을 선택하게 되었으며, Transformer와 같은 최신 기법이 사용되는 모델은 하나 정도 선정하고자 했다. 너무 오래된 모델 또는 너무 최신 모델은 후보에서

제외했다. 그렇게 총 Depth Former, DPT-Hybrid, BTS, Adabins를 선택했다.

4개의 모델을 사용하는 이유는 각 모델마다 weight가 다르고, 구조가 다르므로 깊이 추정값에도 차이가 있을 것이다. 그 차이를 보기 위해 4개의 모델을 사용하고자 했다.

먼저 Depth Former를 선정한 이유는 최신 트렌드인 **Transformer**와 **CNN**을 함께 사용하고 있기 때문에 선정하였다.

DPT의 경우도 Transformer의 한 종류인 Vision Transformer를 사용하고 있는데, Transformer는 fully convolutional network와 비교하여 최대 28% 성능이 향상된다고 한다. 8개의 다양한 데이터셋을 사용할 수 있는 모델로, 추론에 강하다 판단되어 선정하였다.

BTS의 경우 DPT와 동일한 encoder-decoder 구조를 가지면서 feature를 추출하고 예측하는 방식에 차이를 두고 있어 DPT와의 비교와 **Self-Supervised, Supervised, Semi-Supervised learning**을 지원하기 때문에 추후 다른 learning 방법도 적용하고자 선정하였다.

Adabins의 경우 **Adaptive bins** 기법은 최근 기법들에서도 많이 사용되는 기법이다. 이 후 많은 아키텍처에서 적용하는 기법이므로 선정하였다.

3. 결과

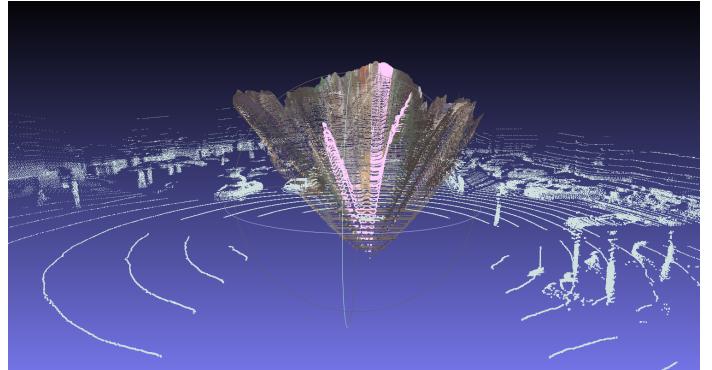
3.1. 시각화 결과

3.1.1. 3d ply 시각화

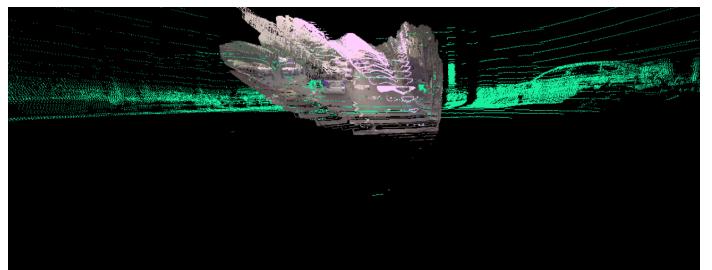
위에서 차원 투영하는 2가지 방식에 대해 설명했다. 여기서는 3차원으로 투영하는 것에 대해 설명하고자 한다. 이미지를 3차원으로 투영하는 과정으로는 다음과 같다.

- 1) 먼저 RGB 이미지와 depth map 이미지를 불러온다.
- 2) RGB 이미지와 depth map 이미지의 크기를 맞춰준다.
- 3) 불러온 depth map 이미지 각 픽셀마다 calibration 정보를 통해 3차원(x,y,z) 점으로 변환한다.
- 4) RGB 이미지의 해당 픽셀의 색상과 3차원 점을 각각 리스트에 저장한다.
- 5) LiDAR와 변환한 3차원 점을 병합하여 시각화한다.

3D point cloud에 대한 시각화는 이미지 형태를 함께 확인하기 위해 ply 포맷으로 사용하였다. pcd 포맷으로도 사용해본 결과 화면 위치가 고정되어 있어 사용하기에 다소 불편함이 있었다.



ply format



pcd format

3.1.2 2d image 시각화

[figure 1](#)에서 첫 번째 이미지는 원본, 두 번째는 깊이 추정 모델로부터 추출된 depth map, 세 번째는 lidar data를 이미지로 투영한 이미지이다. [figure 1](#)에서 확인할 수 있듯이 depth map에 비해 lidar 데이터가 매우 sparse한 특성을 가지고 있다. 따라서 이러한 특성을 개선해준다면 더 나은 결과를 얻을 수 있을 것이다.

LiDAR 데이터를 이미지로 투영했을 때의 투영 정확도를 높이기 위해 다음과 같은 조건을 충족하는 데이터는 필터링했다.

- 1) Lidar 센서 뒤에 위치하는 데이터
- 2) x 방향으로 너무 멀리 떨어져있는 데이터
- 3) y 방향으로 너무 멀리 떨어져있는 데이터

점과 이미지 사이의 관계를 판단했을 때, 두 데이터가 일치하지 않고 있다는 것을 확인할 수 있는데, 이는 캘리브레이션 정보가 오차가 있다고 판단할 수 있다.

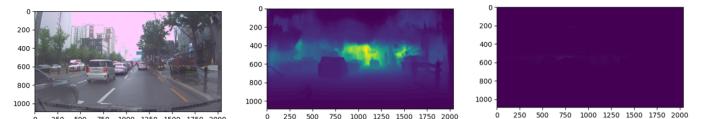


figure 1



figure 2

3.1.3. Calibration 파일 시각화 결과

더 정확한 캘리브레이션 정보를 확인해보기 위해 calibration 폴더에 있는 파일을 투영해보았다. **figure 3**에서는 라이다 데이터를 시각화한 모습이다. **Figure 4**에서는 LiDAR데이터를 이미지로 투영한 모습이다. **Figure 4**에서 볼 수 있듯이 캘리브레이션 정보 자체가 잘못되어 있다는 것을 확인할 수 있다. **Figure 5**는 라이다 데이터를 이미지에 직접 투영해본 이미지이다.

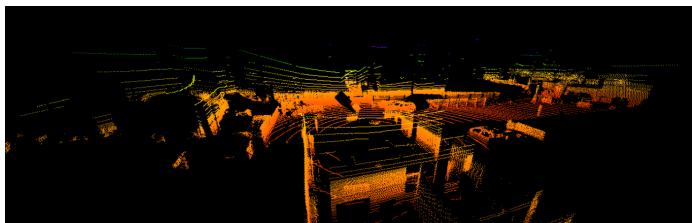


figure 3

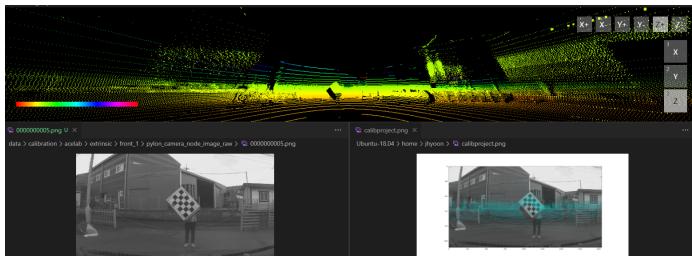


figure 4

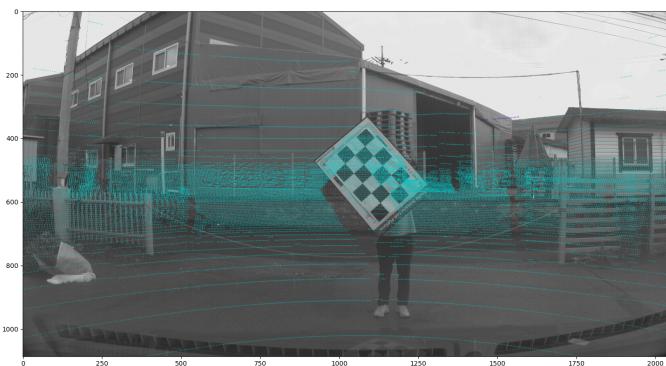


figure 5

3.2. 평가

3.2.1. 평가 전략

모델 4개를 사용해서 각 모델마다 depth map과 lidar 데이터를 이미지 포맷으로 변형한 GT 데이터를 비교한다. 다양한 기법을 사용해보고, 가장 작게 나오는 지표, 가장 크게 나오는 지표를 골라 그 이유를 서술하고자 했다.

3.2.2. 평가 지표

3.2.2.1 δ $\delta_1, \delta_2, \delta_3$

정확도 지표로 δ 가 많이 활용된다. 정확도 δ 는 Eq. (10)과 같이 정의되며, 일반적으로는 <1.25 , $<1.25^2$, $\delta < 1.25^3$ 인 픽셀의 비율을 제시한다. 이 세 지표는 높을수록 성능이 좋다는 의미이다.

$$\max\left(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}\right) = \delta < \text{thr} \text{ for } \text{thr} = 1.25, 1.25^2, 1.25^3$$

3.2.2.2 RMSE

Root Mean Square Error 의 약자로, 평균 제곱근 오차를 의미한다. MSE 값에서 제곱근을 취하여 구할 수 있으며, 오차값에 대해 MSE 값에 비해 이상치에 덜 민감하다는 특징을 가지고 있다.

$$\sqrt{\frac{1}{n} \sum_p^n (y_p - \hat{y}_p)^2}$$

3.2.2.3 RMSE log

RMSE를 구할 때 각각의 값에 log를 취하여 구할 수 있다. 상대적 error를 측정할 수 있다는 특징이 있고, 아웃라이어가 있더라도 값의 변동폭이 크지 않는 강건한 특징을 가지고 있다.

$$\sqrt{\frac{1}{n} \sum_p^n \| \log y_p - \log \hat{y}_p \|^2}$$

3.2.2.4 Abs REL

Abs REL은 Average relative error, 절대 상대 오차를 의미한다.

Abs REL의 식은 다음과 같다.

$$\frac{1}{n} \sum_p^n \frac{|y_p - \hat{y}_p|}{y}$$

3.2.3. 평가 결과

	d1	d2	d3	RMSE
Adabins	0.022	0.023	0.024	8.9375
Depthformer	0.583	0.585	0.586	43.445
BTS	0.024	0.026	0.026	12.325
DPT	0.000	0.000	0.000	60.515

[Table 1.] 모델에 따른 평가 결과

BTS와 Adabins는 delta 부분에서는 거의 동일한 지표를 가진다. 그러나 Depth Former와는 20배 가량 높은 것을 확인할 수 있다. RMSE에서는 Adabins가 BTS보다 조금 더 좋은 성능을 보이며, Depth Former와는 약 5배 가량 좋은 성능을 보이고 있다.

DPT의 경우는 모든 성능 부분에서 좋지 않은 결과를 보였다.

depth former를 활용한 알고리즘이 delta 1,2,3에서 가장 좋은 성능을 보여주고 있으나, RMSE를 함께 고려한다면, Adabins를 활용한 알고리즘이 가장 좋은 결과를 가진다고 판단할 수 있다.

4. 결론 및 향후 연구

시각화를 통한 데이터셋의 구성을 확인하고, 학습된 모델을 통해 추출된 데이터와 LiDAR 데이터를 비교해본 후 평가를 분석하였다. 그 결과, Adabins를 사용하는 것이 우리의 데이터셋에 적합하다.

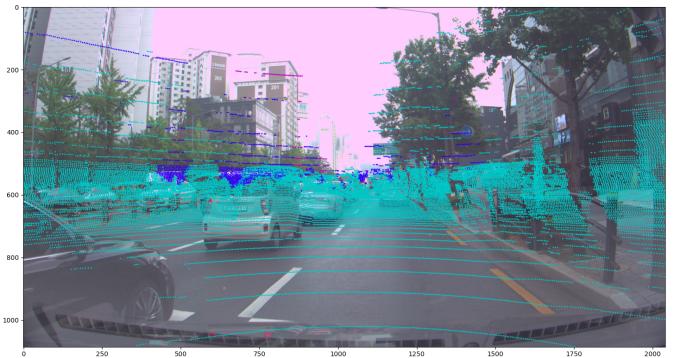
paperswithcode 홈페이지에서는 Depthformer가 좋은 성능을 보이고 있으므로, 원본 그대로 연산을 하는 것이 아닌 전처리를 수행해야 정확한 비교가 될 듯하다. 예를 들어, Depth Former의 경우 depth map을 추출해보면 반전되어 있는 듯한 모습을 볼 수 있다. 따라서 이러한 반전을 전처리하는 과정이 중요하다. 또한, DPT의 경우도 Table 1에서 지표가 좋지 않은 것도 이러한 이유에서 일 것이다.

KITTI 데이터셋을 활용했을 때는 Table 1에서의 결과보다는 더 좋은 결과를 보이는 것으로 보아, 학습되지 않은 데이터셋을 활용하여 추론했을 때는 학습된 이미지에 비해 다소 성능이 떨어지는 것을 확인할 수 있다. 그러므로 학습 데이터셋인 KITTI와 함께 NYU, nuScenes, BDD100K, cityscapes 등 다양한 데이터셋을 사용하여 개선해볼 필요가 있다.

추가로 LiDAR 데이터는 sparse다는 특징이 있으므로, 이를 보간법(interpolation)을 활용하여 점 개수를 늘릴 수 있다.



[original LiDAR data]



[Interpolation 후]

한 점을 기준으로 상/하/좌/우의 점을 채워주는 방식으로 interpolation을 수행했을 때 약 10만개의 점이 추가로 생성되었다.



[Interpolation 전/후 depth gt]

LiDAR 데이터를 이미지로 투영했을 때도 차이를 확인할 수 있었다.

그러나 평가 지표를 적용했을 때는 결과가 거의 동일한 모습이었다. 따라서 추후 조금 더 나은 interpolation 기법을 개발하여 적용하면 평가지표에도 영향을 줄 수 있을 것이라 예상한다.

5. 참고 문헌

- [1] Zhenyu Li, Zehui Chen ,Xianming Liu ,Junjun Jiang.(2022). DepthFormer: Exploiting Long-Range Correlation and Local Information for Accurate Monocular Depth Estimation
- [2] Rene Ranftl, Alexey Bochkovskiy,Vladlen Koltun, Intel Labs.(2021).Vision Transformers for Dense Prediction
- [3] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko and Il Hong Suh Department of Electronics and Computer Engineering, Hanyang University.(2021).From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation
- [4] Shariq Farooq Bhat,Ibraheem Alhashim,Peter Wonka.(2020). Adabins:Depth Estimation using Adaptive Bins
- [5] Chungkeun Lee, Dongseok Shim, H. Jin Kim.Department of Aerospace Engineering, Seoul National University, Seoul 08826, Korea(2021).Deep Learning Based Monocular Depth Estimation: Survey