PA3 Report

20251155 박찬영

1. Overview

- Input: sparse_depth, normal, rgb
- Output: final_refirned_depht
- Goal: Sparse depth를 여러 모듈과 입력을 통해, dens depth를 산출하는게 이번 PA3의 목표이다.

- Step1: HoleFilling.py
    - ./data/data_example/sparse_depth.npy 를 입력으로 받아, initial depth를 구하기
    - 7*7 커널 사이즈를 갖는 patch를 convolution하여, sparse 픽셀을 채워주는 모듈
- step2: unet_model_final.py[1]
    - initial depth와 rgb를 연결하여, (1, 4, H, W)인 Unet을 구하고, 이 unet_input을 입력으로 넣어, (1, 1, H, W) predicted depth를 구하였다. 추후에 step4를 통해서 이 모델이 학습된다.
    - step2는 Unet paper와 거의 유사하게 설계하였고, size와 scale 값을 작게 보정하였다.
- step3: Detph2Normal.py
    - predicted_depth를 입력으로 받아, carmera 좌표계로 좌표변환을 하고, 좌표변환된 predicted_depth map을 통해, normal map을 구하였다.
    - gt.py(ground truth depth 맵)을 사용하여 검증하였다.
- step4: step4_train.py
    - Unet 모델을 학습하는 train 모듈로서, sparse depth와 predicted depth로 L1 손실 함수로, predicted normal와 GT normal L2 or cosine loss로 학습을 진행한다.

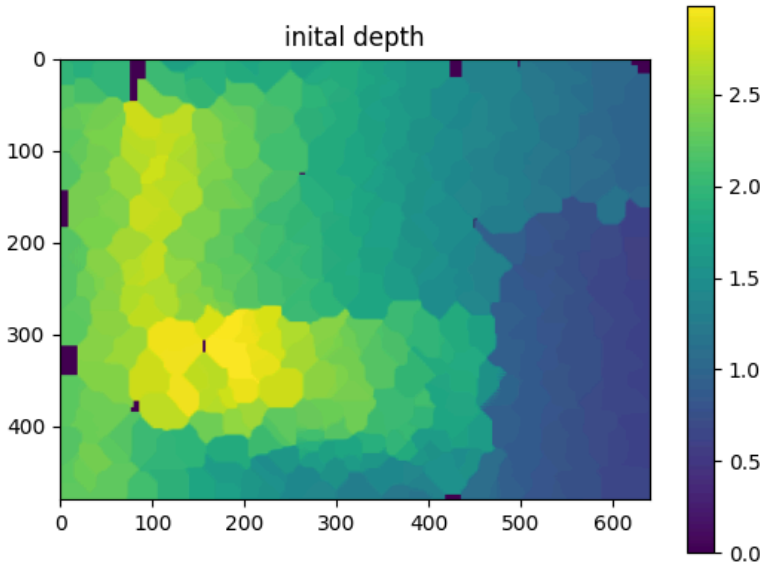
2. Method & Experiment

2.1 HoleFilling.py

- Goal: 본 단계의 목표는 zero값이 대부분인 sparse depth map을 보완하여, initial depth를 생성하는 것이다.
- Input: sparse_depth
    - torch.tensor, shape(1, 480, 640)
    - 유효하지 않는 픽셀이 대부분인 sparse depth map
- Output: inital_depth
    - torch.tensor, shape(1, 1, 480, 640)
    - 0값이 대부분 채워진 initial depth
- 동작 원리
    - 초기 설정

[1] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer international publishing, 2015.

        - 커널 사이즈와 iteration 횟수를 정해준다.
        - 입력을 Conv2d에 사용할 수 있게, unsqueeze하여 (1, 1, H, W)형태로 만들고, input과 output 사이즈를 같게 하기위해서 padding 사이즈를 padding = (kernel_size-1)//2 로 정해준다.
    - 평균 필터로 convolution하여 보간 수행하기
    - 초기 설정한 값들로 Conv2d를 설정해준다. 이때 필터의 가중치는 모두 1로 고정하고, 학습되지 않도록 설정한다.
    - 평균 필터를 활용하여, 주변값 평균을 계산한다. 특히 이때, 주의할 점은 평균을 모든 픽셀에 넣으면 안되고, 이미 유효한 픽셀은 제외하고, 비어있는 픽셀에 그 값을 채워준다.
- 주의할 점
    - 앞서 말한것처럼 처음에 모든 픽셀에 평균값을 채워 넣었더니, 이상한 initial depth가 나왔었다. 즉, 픽셀이 0인 값에만 평균값을 넣어줘야한다.
    - Conv2D는 bool 타입은 사용할 수 없음으로 float으로 변환해야하는 것도 주의해야한다.
    - 그리고 평균을 구할때, 0으로 나누면 안되는 것도 주의해야한다.
    - 커널 사이즈와 iteration 크기를 처음에 작게 하여, initial depth값을 출력하였는데, 유효하지 않은 값이 많아 점점 사이즈를 키워서 수정하였다.



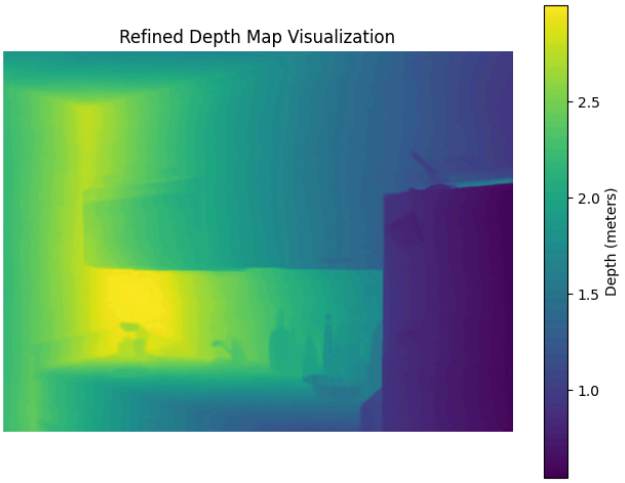
2.2 Unet[2]

- Goal: step1에서 생성한 initial depth map은 sparse 정보를 주변 픽셀 평균으로 보완하여 생성되므로, 객체 경계나 구조적 특성이 손실되어 있다. 이를 해결하기 위해, UNet 구조를 활용해 initial depth map을 개선하고, GT Depth에 보다 가까운 predicted map을 생성하는 것을 목표로한다.
- Input: RGB+Initial Depth (1, 4, H, W)
- Output: predicted Depth (1, 1, H, W)

---

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer international publishing, 2015.

- 동작 원리
  - 각주를 달아 놓은 것처럼, Unet Paper와 거의 유사하게 설계하였다. 기존 Unet에 비해 레이어(4 → 2)와 채널수(64 →16)를 축소하였다. 각 구조에 대한 설명은 다음과 같다.
    - 인코더: Conv → BN → Relu를 두 겹으로 연속으로 쌓아 DoubleConv를 설계하고, DoubleConv + Maxpool 구조를 하나의 layer로 설계하여 총 2층의 layer로 설계였다. 인코더는 down sampling하면서 고수준 feature를 추출한다.
    - Bottleneck: 가장 해상도가 낮은 공간에서, 고수준 feature를 생성한다. 해상도는 가장 작지만 채널 수는 가장 크므로 의미 정도가 밀집되어있다.
    - 디코더: ConvTranspose2d로 feature map를 2배로 증가시키면서, 동일 해상도의 인코더 레이어의 출력값과 concat하여 skip connection으로 이어 붙인다. 그리고 DoubleConv로 정제 시켜준다. 디코더도 마찬가지로 2 layer로 구성된다.
- Skip connection의 역할
  - Skip connection은 U-Net 구조의 핵심으로, 로컬 정보 손실을 보완하고, 세밀한 출력 예측을 가능하게 한다.
  - 이 역할 때문에, 아래 사진같이 경계선, 얇은 구조등이 더 정교하게 복원되었다.



initial+rgb를 입력으로 받고**, gt(ground truth depth)**를 감독신호로 사용하고 **criterion** 파이토치 손실 함수를 이용해서 얻은 **predicted depth map, unet**을 레이어와 채널을 축소해도 **PA3 task**에는 아무 문제 없다는 것을 알 수 있다.

2.3 Depth2Normal.py

- Goal: Depth map을 입력으로 받아, 3D surface normal map 추정하는 것이 목적이다. 즉, step3은 step4에서 GT normal로 학습을 하기 위한 준비 단계이다.
- Input: predicted_depth map (1, 1, H, W)
- Output: 3D predicted_normal map (1, 3, H, W)
- 동작 원리
  - Step4는 PA3.pdf에 각주로 달린 블로그[3]를 바탕으로 설계하였다.
  - 먼저, Image coordinate system를 Camera coordinate system로 변환한다. camera intrinsic matrix는 아래식과 같다. 내부 파라미터는 과제에 주어진

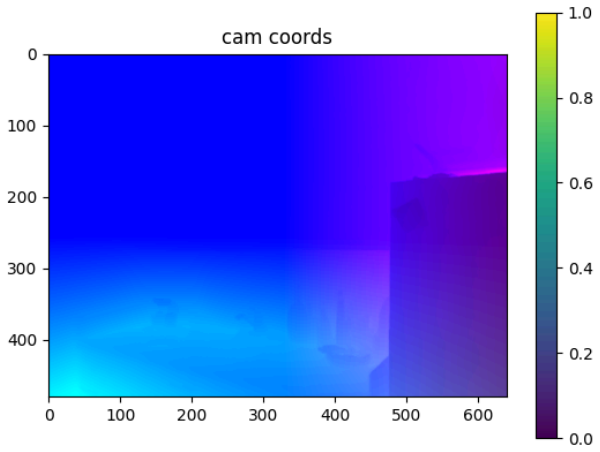
[3] https://darkpgmr.tistory.com/32

파라미터를 이용하였다. s는 scale factor로 여기서는 depth라고 보면 된다.

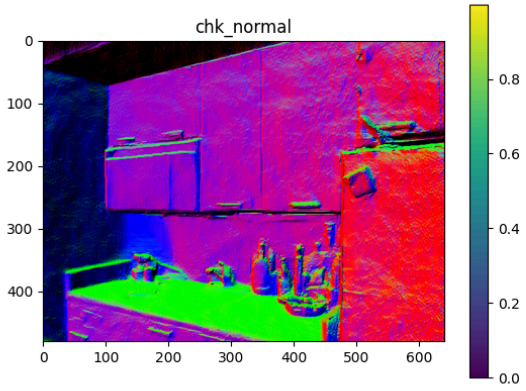
$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}.$$

[4]

- 코드도 위식에서 K^{-1}하고, 행렬곱하여 3D map을 구하였다. 단, 역행렬을 행렬곱하기전에 scale값인 depth를 곱해주었다. 그래서 얻은 중간 결과값은 아래 사진이다.



- 이후에는 PA3.pdf에서 명시한 과정대로 경계값 누락을 방지하기 위해, padding을 작업을 한후, vertical vector와 horizontal vector를 구해서 cross product를 하고, normalization하여 normal map을 구했다. 그 값은 아래 사진과 같다. 참고로 이 모듈의 설계 검증을 위해, gt.npy (Ground truth depth)를 사용하였다.

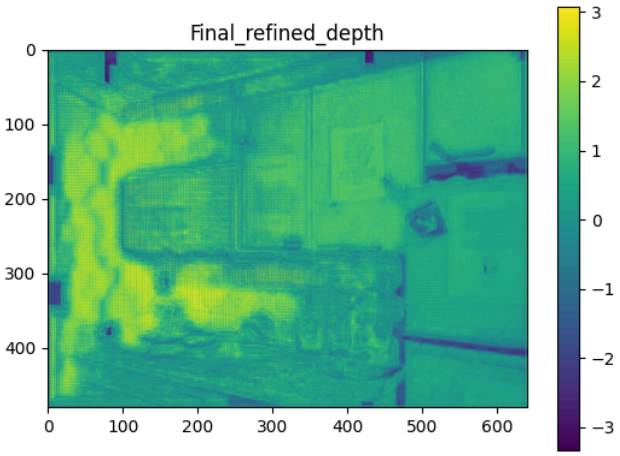


- 주의할점
  - cross product할때, 방향을 가지기 때문에 순서가 중요하다. 하지만 값이 이상하게 나오면, 순서를 바꿔서 시행하는 것을 주의하면된다.

[4] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html?utm_source=chatgpt.com

2.4 step4_train.py

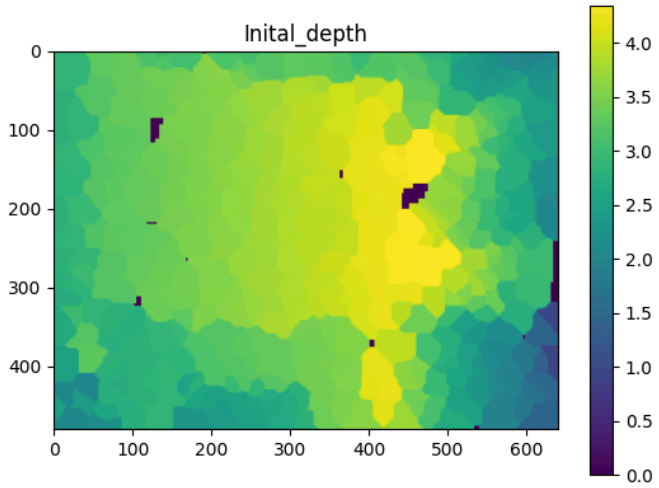
- Goal: Unet 모델을 학습시키는 것이 목표이다. 그래서 최종 refined depth map을 예측하는 것이 목표이다.
- Input
  - train 모듈에 필요한 데이터를 위해 step4 data loader를 설계하였다. data loader 모듈에서는 rgb, sparse depth, GT normal, holefilling으로 구한 initial deph를 전처리한다.
  - Unet에 들어가는 unet input은 train 모듈에서 설계를 하였다.
- Output
  - 학습한 unet의 가중치값인 unet_trained.pth에 저장하였다. 추후에 main모듈에서 사용하였다.
- Loss Function
  - 총 3가지의 loss 설계하였고, 실험을 통해 2가지를 선택하였다.
    - sparse_depth_loss: sparse depth에서 유효한 픽셀 값(0이 아닌 픽셀)을, predicted depth와 비교하는 손실함수이다. 유효 픽셀을 판별해주는 mask를 구한후, L1 기반으로 손실함수를 구했다.
    - normal_l2_loss: predicted normal과 GT normal을 비교하는 것이 목적인 손실함수로서, L2 기반으로 구햇다.
    - normal_cosine_loss: 위 normal loss를 cosine loss 기반으로 구하였다. 하지만 실험결과 L2 loss보다 결과값이 좋지 않아 사용하지 않았다.
  - 앞서 설계한 sparse_depth_loss와 normal_l2_loss를 각각 alpha와 beta를 곱해서 더해주어 최종 loss = alpha * sparse_depth + beta * loss와 normal_l2_loss를 구했다.
- 주의할점
  - alpha, beta, 학습율, epoch, batch 모두 조절하여, 실험을 하였지만 좋지않은 결과값을 얻었다. 하지만 파라미터를 분석한 내용은 아래와 같다.
  - alpha: sparse depth loss는 유효한 픽셀 값이 매우 작기 때문에, alpha를 beta에 비해 많이 키워야한다.
  - beta: alpha 대비 beta 비율에 따라, 결과값이 크게 바껴서 조금씩 수정하여 실험하였다.
  - 하지만, 아래 사진처럼 좋지 못한 결과를 얻었다.



2.5 Submission data

submission data로 얻은 결과는 다음과 같다.

1. **Initial Depth**



2. **Final_reifned_depth.npy**