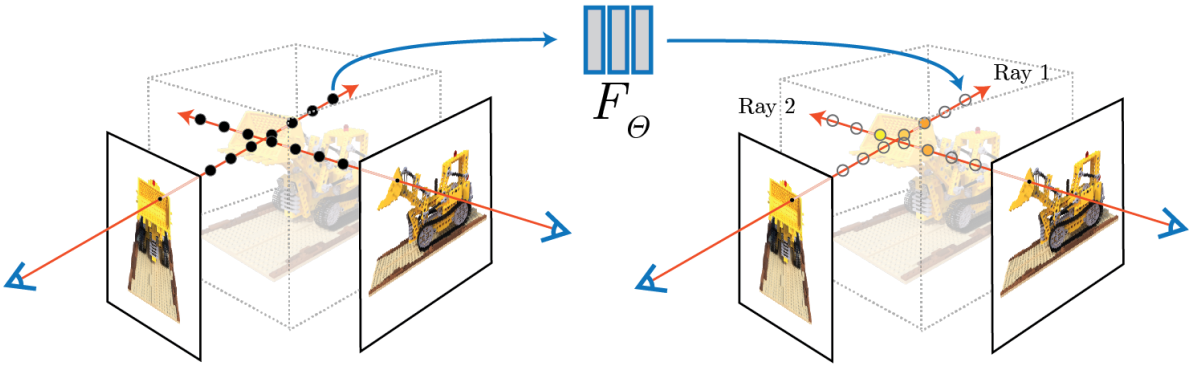
39-1_AI모델 환경 설치가이드



- **레퍼런스 논문(ECCV 2020 Oral)** : https://www.matthewtancik.com/nerf
- **레퍼런스 소스코드(pytorch)** : https://github.com/yenchenlin/nerf-pytorch
- **학습 모델 :** Novel view synthesis
- **모델 :** NeRF
- **성능 지표 :** PSNR,SSIM
- **과제명** : NIA 39-1 3D 에셋-이미지쌍 데이터 (2023.01.31)
- **개발 내용**
  - 구축된 작업물의 최상, 상, 중, 하, 최하 난이도의 Potable, Non-Potable 3D 에셋 데이터를 샘플링
  - 총 10개 데이터셋에 대한 View Synthesis 모델을 개발하여 유효성 검증을 진행함.
  - 24개 시점의 RGB 이미지를 학습한 모델을 통해 새로운 뷰의 RGB 이미지를 생성할 수 있음
- **응용서비스**
  - 가상 카메라를 활용하여 3D 형상 데이터를 여러 방향에서 촬영(24장 이상)
  - 텍스처 데이터가 입혀진 3D 형상 데이터를 메타버스 공간상에서 로드
  - 추후 제공된 코드를 활용하여 오토 렌더링, 3D 스캐닝서비스 개발가능

작업환경

```
CPU : AMD EPYC 7543 32 core * 128개
RAM : 1TB
GPU : NVIDIA A100 * 8 개
CUDA 버전 : 11.3
OS : Ubuntu v18.04
Python : 3.8
Pytorch 딥러닝 프레임워크 버전 : 1.11.0+cu113
```

학습 환경 설치 방법

0. 데이터셋 목록

```
최상
- Po : 1-5-23_20_minicooper-luggage-minicar_5_1_1_nurbs
- Np : 2-16-2_24_leather-white-wood-3+-seater-recliner_5_5_1_polygon

상
- Po : 1-7-21_9_pink-cylinder-rice-cooker_4_2_1_nurbs
- Np : 2-17-6_15_silver-round-streetlight_4_4_1_polygon

중
- Po : 1-5-16_12_knitted-toy-cat-stand_3_2_1_polygon
- Np : 2-21-2_7_retro-style-mint-oven_3_3_1_polygon

하
- Po : 1-7-20_20_beige-body-juicer_2_2_1_nurbs
- Np : 2-16-12_22_square-pillow-bean-bag_2_4_1_polygon

최하
- Po : 1-11-16_17_white-vivid-sport-goggles_1_2_1_polygon
- Np : 2-14-10_24_Yellow-TopBottom-Solid-Square-Stool_1_2_1_nurbs
```

---

1. 개발 환경 구축

1.1 도커 이미지 생성 및 컨테이너 구동

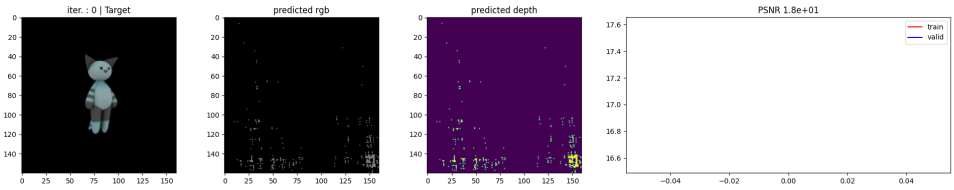
```
# 관련 코드 및 라이브러리 다운로드
git clone https://github.com/Narnialabs/NIA_39-1-2_NeRF.git

# 1.2. 도커 배치 파일 명령어 실행
bash docker_run.sh
```

1.2. 도커 컨테이너 내 모델 실행 방법

1.2.1. 도커 실행 및 패키지 설치

```
docker start pytorch_nerf # pytorch_nerf 컨테이너가 실행하지 않은 경우
docker exec -it pytorch_nerf bash # 컨테이너를 설 실행후 해당 명령어로 접속함
#python 확인 후 버전이 2.x 버전일 시 아래와 같이 명령어를 입력함.
alias python=python3
cd /home/Nia_AI/
#파이썬 라이브러리 다운로드 받음
pip install -r requirements.txt
# 데이터셋 압축 풀기
unzip dataset.zip
```

1.2.2. AI 모델실행 방법



<그림.checkpoint>

```
# --gpu_num은 내가 사용하고자 하는 gpu의 번호
# --config 타겟 에셋의 환경설정 파일
# --training 학습 시 True로 설정함
# --testing 학습 된 모델로 시험 시 True로 설정함
# --rendering 학습된 모델로 렌더링시 True로 설정함
# "asset_x_x.yaml" 파일에 들어가서 학습하고자하는 에셋의 환경 설정을 변경할 수 있음

# 모델 학습시
python run_nerf.py  --gpu_num 0 --config ./config/asset_1_1.yaml --training True
# 학습된 모델 테스트시
python run_nerf.py  --gpu_num 0 --config ./config/asset_1_1.yaml --testing True

# 학습된 모델로 새로운 뷰 예측 및 렌더링
(optional) python run_nerf.py  --gpu_num 0 --config ./config/asset_1_1.yaml --rendering True
```

1.3. Output

```
#학습이 완료된 후 로그폴더가 자동으로 생성도니 후 모델, 체크포인트, 테스팅, 렌더링 폴더에 결과물이 저장된것을 확인할 수 있습니다.
└── logs
    ├── model
    │   └── tgt_class_model.tar # 학습 완료된 모델 저장
    ├── checkpoint
    │   └── tgt_class_folder
    │       ├── result_0.png # 500 번에 한번씩 체크포인트 결과 저장
    │       └── result_500.png
    ├── testing
    │   └── tgt_class_folder
    │       ├── TestSet_result.png # 테스트 결과 저장
    │       └── ValSet_result.png # 검증 결과 저장
    ├── rendering
    │   └── tgt_class_video.png # 360 각도 뷰 렌더링 결과 저장
```

---

2. Utils

```
# (1) 정사각 이미지가 아닌경우 crop_images.py 를 이용해서 정사각 이미지로 처리 할 수 있음.
python image_processing.py
\ --src_path [이미지소스경로] --tgt_asset [타겟에셋명] (크롭할경우[default './dataset/data_square/'폴더에 저장됨])
\ --cropping True --black True (블랙 배경으로 바꾸고 싶은 경우 True로 설정하면됨 안설정하면 하얀색 배경임.)

# (2) colmap 으로 이미지를 추정해서 pose를 추정한 경우 colmap2nerf.py를 이용해서 nerf가 읽어들일수있는 데이터로 처리해줌.
# 사전에 colmap_text 폴더를 만들어서 colmap의 output인 cameras.txt, images.txt를 넣고, images 폴더를 만들어서 이미지를 넣어줌
python colmap2nerf.py
# 결과물로 transforms.json
```