2023 전기 졸업과제 중간보고서

NeRF 를 이용한 중고시장 플랫폼 개발

팀명 : 에듀윌

201824540 은승우

202055511 권재섭

201824439 김성현

목차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1. 요구조건 및 과제목표

차세대 기술로 주목받고 있는 NeRF(Neural Radiance Fields for View Synthesis)를 이용하여 사용자가 촬영한 상품의 2D 이미지들을 NeRF기술을 사용해 3d로 변환해서 제공하는 중고 거래 플랫폼을 만드는 것을 목표한다. 이로 인해 중고거래 중 불확실한 제품을 사진으로만 판단하는 것 보다, AR을 통해 실제로 제품을 확인할 수 있도록 하여 제품을 설계하도록 한다.

1.2. 제약 사항 및 변경 사항

- 서버의 구조 변경

➔ Spring Boot를 사용해 서버를 개발하였고, 이 서버는 Docker Container의 Linux 환경에서 동작한다.

- Android 어플을 통한 프런트 개발

➔ Kotlin을 이용하여 Android를 개발하기로 결정하였으며, Unity와의 원활한 연동을 위해 Android Target API 31 (Android 12)를 적용

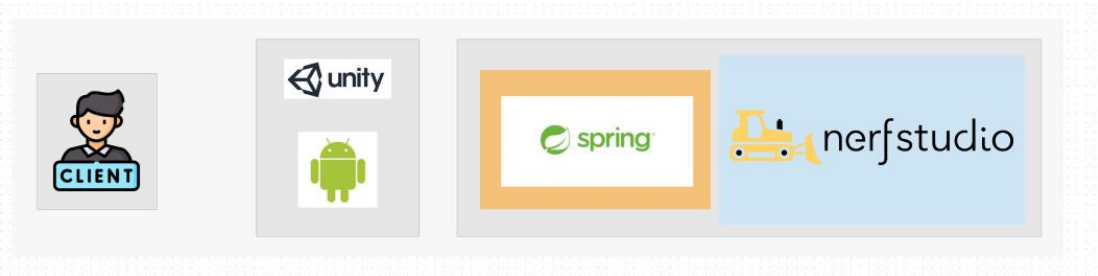
- AR개발툴 확정

➔ Unity에서 동작하는 Unity AR Core를 이용하여 안드로이드에서의 AR 기능 구현

제2장 설계 상세화 및 변경 내역

2.1 모델 설계

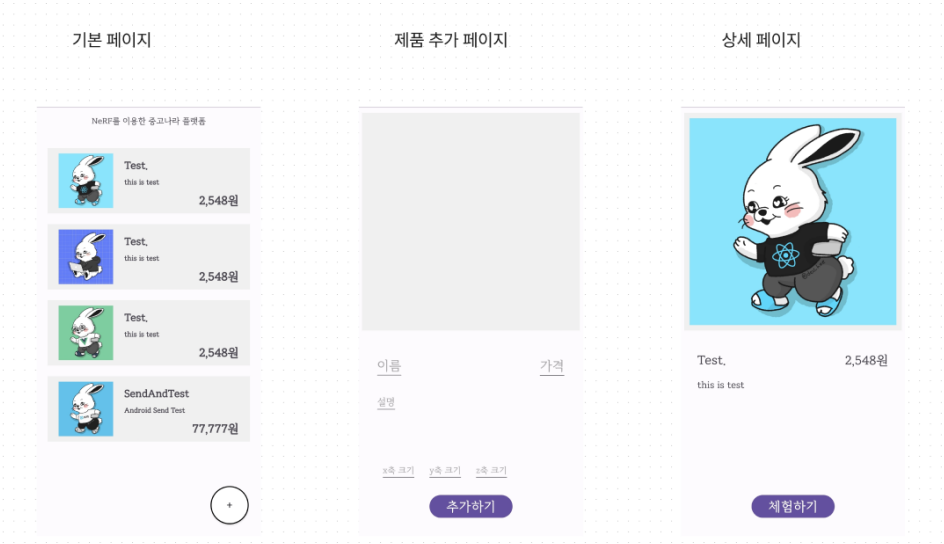
2.1.1 수정된 모델 설계도



판매자는 동영상 파일, 사진 파일, 제품의 설명과 가격들을 어플리케이션에 전달하는 과정을 거친다. 그 이후, 어플리케이션이 서버에 그 데이터들을 전송하며, 판매 글이 서버에 저장된다. 이때, 서버는 동영상파일을 Input으로 이용하여 NeRFStudio를 통해 mesh로 변환한다. 이때의 Output은 obj, mlt, png 파일이며, 이 파일들은 DB에 저장된다.

2.2 모델 구조 및 구현

2.2.1 어플리케이션 구조



어플리케이션을 시작하면 어플은 기본페이지를 보여주며 서버에 get요청으로 전체 데이터의 리스트를 받아온다. 이때, 받아온 데이터를 기반으로 RecyclerView를 만들어 구매자가 찾는 물품을 골라서 보게 한다.

이때, 만약 판매자가 상품을 등록하고 싶다면, 기본페이지 오른쪽 하단에 플러스 모양의 버튼을 누르면 상품 등록페이지로 화면이 전환된다. 이때, 등록하고 싶은 데이터를 EditText에 입력한 후, 추가하기 버튼을 누르면 서버에 Post 요청으로 페이로드를 전달하여 서버에 데이터 저장한다.

만약 구매자가 상품정보를 보고싶다면, 기본페이지의 RecyclerView를 선택하면 화면이 전환되며, 아이템의 상세 페이지로 이동한다. 이때 서버에서 제공되는 사진 이미지, 이름, 가격, 부가 설명 등이 업로드 된다.

2.2.2 NeRF Studio 이용

Nerfstudio는 Data Preprocess, Data Loader, Model Training, Visualizing, Rendering을 API형태로 제공하는 Framework로 Mip-NeRF, instant-NGP, Nerfacto 등 다양한 NeRF model을 지원해서, 여러 model들을 nerfstudio만으로 손쉽게 사용해 볼 수 있다.

NeRF는 입력 값으로 사진과 해당 사진을 찍은 카메라의 위치, 방향 정보를 요구하는데, 일반적인 동영상이나 사진의 경우 카메라의 위치, 방향정보가 없기 때문에 COLMAP을 사용한 데이터 전처리과정이 필요하다. COLMAP은 2d 이미지들로 3d 정보, 즉 카메라의 위치, 방향 정보를 얻어내는 기술로 대략적인 과정은 아래 그림과 같다.

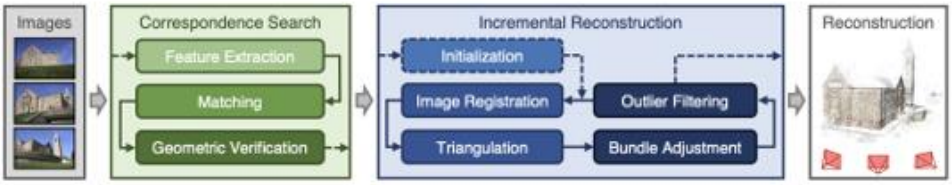


Figure 2. Incremental Structure-from-Motion pipeline.

<COLMAP 과정>

이렇게 전처리된 데이터를 입력 값으로 해서 학습하고, 학습결과로부터 mesh를 추출해 낸다.

이 과정들은 모두 nerfstudio에서 지원하는 기능이고, shell command를 통해 실행되는데, shell command들을 shell script로 작성하여 손쉽게 실행시킬 수 있도록 하였다.

```
!/bin/bash

id=$1

echo "start sh"

#process
echo "start process"
ns-process-data video --data /workspace/video/$id.mp4 --output-dir /workspace/result/data/$id

#train
echo "start train"
ns-train nerfacto --data /workspace/result/data/$id --output-dir /workspace --experiment-name result --method-name train --timestamp $id --
pipeline.model.predict-normals True --viewer.quit-on-train-completion True --logging.steps-per-log 30001

#export
echo "start export"
ns-export poisson --load-config /workspace/result/train/$id/config.yml --output-dir /workspace/result/mesh/$id --target-num-faces 50000 --n
um-pixels-per-side 2048 --normal-method model_output --num-points 1000000 --remove-outliers True --use-bounding-box True --bounding-box-min
 -1 -1 -1 --bounding-box-max 1 1 1

#HTTP message(inform server nerf is done)
echo "send message"
curl -X GET "http://localhost:3389/finishNerf?id=$id"

echo "All done"
```

<nerfstudio 실행을 위한 shell script>

```
start sh
start process
Number of frames in video: 1713
Number of frames to extract: 343
[22:06:08] Done converting video to images.                    process_data_utils.py:173
[22:07:37] Done downscaling images.                            process_data_utils.py:361
[22:08:01] Done extracting COLMAP features.                        colmap_utils.py:131
[22:10:32] Done matching COLMAP features.                          colmap_utils.py:145
[22:17:25] Done COLMAP bundle adjustment.                          colmap_utils.py:167
[22:17:26] Done refining intrinsics.                               colmap_utils.py:176
           All DONE                                video_to_nerfstudio_dataset.py:118
           Starting with 1713 video frames              video_to_nerfstudio_dataset.py:121
           We extracted 343 images                      video_to_nerfstudio_dataset.py:121
           We downsampled the images by 2x, 4x and 8x   video_to_nerfstudio_dataset.py:121
           Colmap matched 10 images                     video_to_nerfstudio_dataset.py:121
```

<shell script실행결과 - 데이터 전처리>

```
29980 (99.93%)      26.938 ms           538.763 ms          154.76 K
29990 (99.97%)      27.575 ms           275.751 ms          151.66 K
29999 (100.00%)
--------------------------------------------------------------------------------------------------------
Viewer at: https://viewer.nerf.studio/versions/23-05-15-1/?websocket_url=ws://localhost:7007
                                Training Finished
╭──────────────────────────────────────────────────────────────────────────────────────────────────────╮
│ Config File           │ /workspace/result/train/20/config.yml                                        │
│ Checkpoint Directory  │ /workspace/result/train/20/nerfstudio_models                                 │
╰──────────────────────────────────────────────────────────────────────────────────────────────────────╯
Printing profiling stats, from longest to shortest duration in seconds
Trainer.train_iteration: 0.0268
VanillaPipeline.get_train_loss_dict: 0.0163
```

<shell script실행결과 – 학습>

```
start export
[22:31:19] Auto image downscale factor of 2                    nerfstudio_dataparser.py:336
           Skipping 0 files in dataset split train.            nerfstudio_dataparser.py:163
           Skipping 0 files in dataset split test.             nerfstudio_dataparser.py:163
Setting up training dataset...
Caching all 9 images.
Setting up evaluation dataset...
Caching all 1 images.
Loading latest checkpoint from load_dir
✅Done loading checkpoint from /workspace/result/train/20/nerfstudio_models/step-000029999.ckpt
Checking that the pipeline has a normal output.
Computing Point Cloud ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 00:03
✅Cleaning Point Cloud
✅Generated PointCloud with 1004089 points.
Computing Mesh... this may take a while.
[WARNING] /root/Open3D/build/poisson/src/ext_poisson/PoissonRecon/Src/FEMTree.IsoSurface.specialized.inl (Line 1858)
          Extract
✅Computing Mesherage roots: 2
✅Saving Mesh.
Running meshing decimation with quadric edge collapse
Texturing mesh with NeRF
Unwrapping mesh with xatlas method... this may take a while.
✅Unwrapped mesh with xatlas method━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 01:06
Creating texture image by rendering with NeRF...
Writing relevant OBJ information to files...
Writing vertices to file ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% ? 00:00
Writing vertices to file ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% ? 00:00
Writing vertex normals to file ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% ? 00:00
Writing faces to file ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% ? 00:00
                                All DONE
                    Unwrapped mesh using xatlas method.
                    Mesh has 28058 vertices and 49996 faces.
          Length of rendered rays to compute texture values: 0.03581223264336586
                    OBJ file saved to /workspace/result/mesh/20/mesh.obj
                    MTL file saved to /workspace/result/mesh/20/material_0.mtl
          Texture image saved to /workspace/result/mesh/20/material_0.png with resolution 2048x2048 (WxH)
```

<shell script실행결과 – mesh 추출>

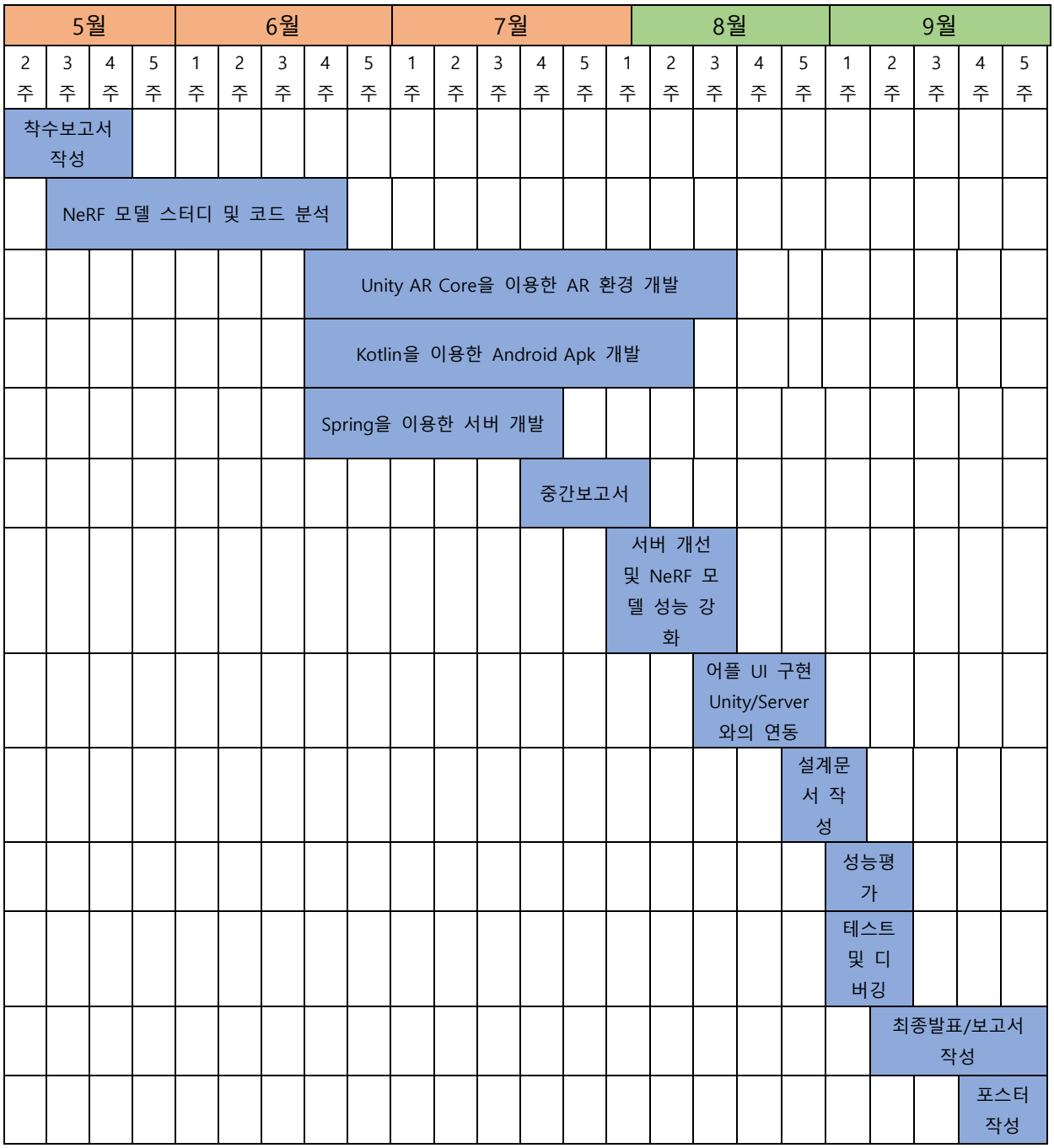
2.2.3 Unity AR Core 이용

어플리케이션에서 체험하기를 누르면, 사진에 나와있는 물체를 3D AR로 확인하기 위해 Unity어플과의 연동이 이루어진다. 이때, 휴대폰 후방 카메라를 이용한 AR화면이 나오며, 원하는 위치에 터치하거나, 특정 이미지를 카메라가 인식하면 보고 있었던 물체의 3D 모델이 실제 자기 위치에 AR로 생성된다. 이때, 생성된 3D모델은 서버에 있는 obj, mtl, png 파일을 이용하여 생성되며, 이 오브젝트에 회전가능한 기능을 추가하여, 자기가 원하는 위치에 확실하게 고정할 수 있게 도와준다. 또한, Occlusion기능도 추가시켜 현실감을 더욱 더 증가시키는 기능도 추가하였다. 밑의 그림은 실제 어플리케이션이 어떻게 동작하는지 보여주는 예시이다.

<Unity AR Core를 이용한 Android AR APK>

제3장 개발 일정 및 역할 분담

3.1 개발 일정



3.2 구성원별 진척도 및 계획

이름	역할분담
은승우	- Unity AR Core 개발 - Mesh 전처리 - Occlusion/택스쳐 개발
김성현	- 안드로이드 프런트 개발 - 안드로이드/유니티 연동화 - 입력 데이터관리
권재섭	- shell script를 통한 nerfstudio 자동화 - spring boot를 사용해 서버 개발 - 서버의 http request 보안 강화

제4장 보고 시점까지의 과제 수행 내영 및 중간 결과

4.1 NeRF model 비교

NeRF model	Instant-NGP	Mip-NeRF	Nerfacto
학습량(기본값)	30,000 iterations	1,000,000 iterations	30,000 iterations
학습소요시간	약13분	24시간이상	약13분

<NeRF model별 학습량, 학습소요시간>

NeRF Studio는 몇몇개의 NeRF 모델을 제공하며, 각각 간의 장단점이 존재한다. 대표적으로 자주 사용하는 3가지의 모델을 대상으로 테스트를 해봤으며, 결과는 위와 같다.

instant-NGP는 속도향상을 목표로 한 model로 학습소요시간은 약 13분으로 매우 빠른 편이다.

Mip-NeRF는 속도향상이 아닌 Anti-Aliasing을 목표로 한 model이다보니 결과의 품질은 좋으나, 학습요구량이 많고, 각 iteration의 소요시간 또한 다른 두 model에 비해 길어서 학습소요시간이 매우 길다.

Nerfacto는 nerfstudio에서 추천하는 model로 여러 model을 통합시킨 model이다. 학습소요시간은 약 13분으로 instant-NGP와 거의 동일하고, mesh 추출의 소요시간은 instant-NGP보다 3배이상 빠르다. 또한, Nerfacto의 경우 다른 두 model과 달리 학습단계에서 normal을 예측하도록 옵션을 줄 수 있는데, 이렇게 할 경우 학습시간이 1.2~1.3배로 늘어나기는 하지만 추출되는 mesh의 품질이 매우 좋아진다. 시현 결과는 밑과 같다.

(좌)instant-NGP mesh 추출 결과

(우)Nerfacto(학습시 normal 예측) mesh 추출 결과

4.2 중간 결론

NeRF 모델의 선정을 완료하였으나, 아직까지 Mesh의 정밀도가 완벽하지 않아 그것에 대한 성능 개선이 필요하다. 또한, Android 와 Unity AR Core를 연동을 할때의 호환이 잘 되지 않아 그 부분에서의 연구도 더 필요할 예정이다. 또한, AR에서 OBJ를 생성할 때 터치로 생성할 것인지, 이미지 트래커를 이용하여 이미지를 생성할 것 인지에 대한 논의가 더 필요하며, Occlusion을 적용할 것 인지에 대한 논의도 필요하다. 추가로, 공간에 볼 때, 조명에 따라 OBJ의 모습이 차이 날 수 있음으로, 이것의 택스쳐를 어떻게 수정할 것인지에 대한 논의도 필수적이다.