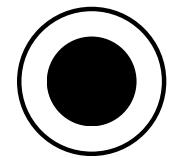


# 찜콩 : 보여줘! 홈즈

CV-05

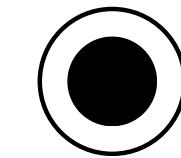
팀원: 양우희 이영진 조민지 조수민 조창희 한상범

# INDEX



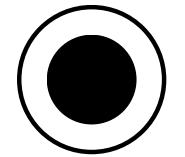
## Intro

팀, 프로젝트 소개 및 타임라인



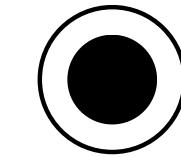
## Model

공간 및 가구 재구성 모델 설명



## Product Serving

전체 서비스 아키텍쳐 및 구현



## Conclusion

시연 영상 및 자체 평가 의견

# 1. Intro

- 
1. 팀 소개
  2. 프로젝트 주제
  3. 프로젝트 타임라인

# TEAM. 짬콩의 기술력

## 양우희

가구 모델러, 모델 서빙, 기획

역할 :

openLRM 모델 기반 커스텀 데  
이터 학습 및 가구 재구성



## 이영진

3D 렌더링

역할 :

모델 결과 전시 및 위치 초기화  
공간 추론 결과 최적화



## 조민지

공간 모델러, 모델 서빙

역할 :

Nerfacto 모델 기반 공간 재구  
성 및 전후처리 최적화



## 조수민

3D 렌더링, 디자인, 모델 성능 개선 연구

역할 :

모델 결과 전시 및 배치 시뮬레이터 구현  
서비스 디자인, 가구 모델 성능 개선 연구



## 조창희

풀스택 개발, 모델 서빙

역할 :

전체 아키텍쳐 설계  
클라이언트/서버 로직 구현 및  
분산 추론 환경 연동



## 한상범

백엔드 개발, 모델 서빙, 일정관리

역할 :

클라우드 기반 서버 로직 구현 및  
분산 추론 환경 연동



## 프로젝트 기획배경

도면 기반의 기존 서비스는 사용자가 직접 그리려면 툴 사용을 위한 일정 수준의 도메인 지식을 요구하고, 도면 상으로는 알 수 없는 구조 변경 및 실제 방 분위기가 반영되지 않는 한계점 존재

## 프로젝트 목적

사용자가 쉽게 인테리어 요소(공간, 가구)를 촬영하여 업로드하면 딥러닝 모델이 3차원 정보를 예측, 예측 결과를 렌더링하여 현실감 넘치는 가상 인테리어로 시뮬레이션하는 서비스 제작

## 프로젝트 기대효과

일반 사용자 대상 가상 인테리어 서비스, 공유 갤러리 제공

기업 사용자 대상 가상 쇼룸, 홍보 서비스 제공

# 프로젝트 타임라인

| 내용          | 1월 1주차 | 1월 2주차 | 1월 3주차 | 1월 4주차 | 2월 1주차 | 2월 2주차 | 2월 3주차 | 2월 4주차 | 3월 1주차 | 3월 2주차 | 3월 3주차 | 3월 4주차 |
|-------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 기획          |        |        |        |        |        |        |        |        |        |        |        |        |
| 마이디어 기획     |        |        |        |        |        |        |        |        |        |        |        |        |
| 모델 조사       |        |        |        |        |        |        |        |        |        |        |        |        |
| 웹 페이지 디자인   |        |        |        |        |        |        |        |        |        |        |        |        |
| 1차 데모 구현    |        |        |        |        |        |        |        |        |        |        |        |        |
| 퍼블리싱        |        |        |        |        |        |        |        |        |        |        |        |        |
| REST API 구현 |        |        |        |        |        |        |        |        |        |        |        |        |
| 렌더링 구현      |        |        |        |        |        |        |        |        |        |        |        |        |
| GPU Pool 세팅 |        |        |        |        |        |        |        |        |        |        |        |        |
| 모델 추론 환경 세팅 |        |        |        |        |        |        |        |        |        |        |        |        |
| 최종 구현       |        |        |        |        |        |        |        |        |        |        |        |        |
| 가구 배치 기능 구현 |        |        |        |        |        |        |        |        |        |        |        |        |
| 로드 밸런싱      |        |        |        |        |        |        |        |        |        |        |        |        |
| CI/CD 구축    |        |        |        |        |        |        |        |        |        |        |        |        |

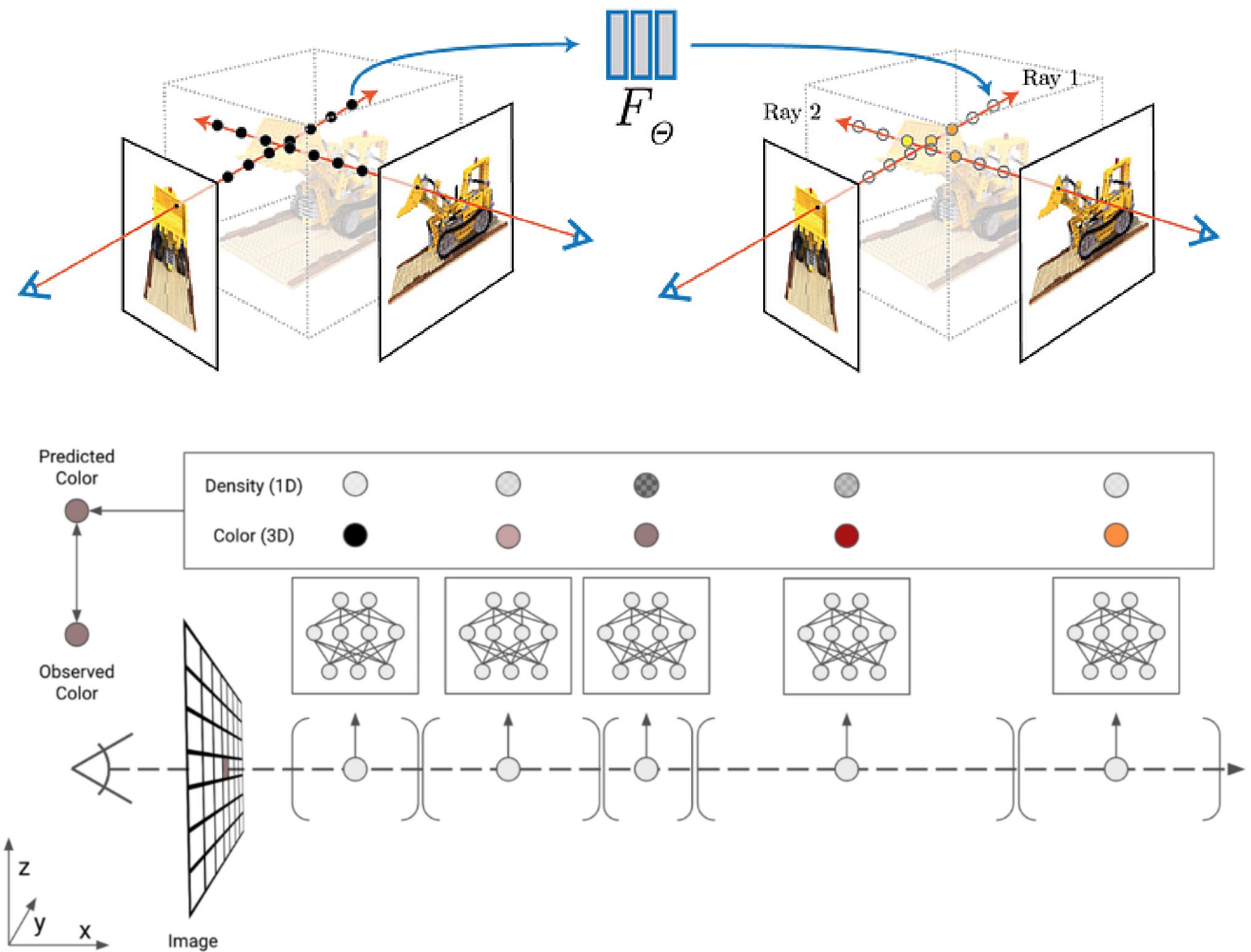


# 2. Model

- 
1. 사용자의 공간을 재구성 하는 공간 모델
  2. 인테리어 할 가구를 재구성 하는 가구 모델

# 이미지 기반 시점 합성

- 3D 모델을 카메라 시점에 따라 투사하여 2D 시점 영상을 구하는 렌더링 과정을 역으로 수행
- ⇒ 여러 시점에서 찍은 2D 영상들로 특정 3D 장면을 학습
- 이미지와 이미지를 찍은 카메라 시점인 **카메라 포즈**를 입력으로 딥러닝 모델이 장면의 3차원 구조를 학습

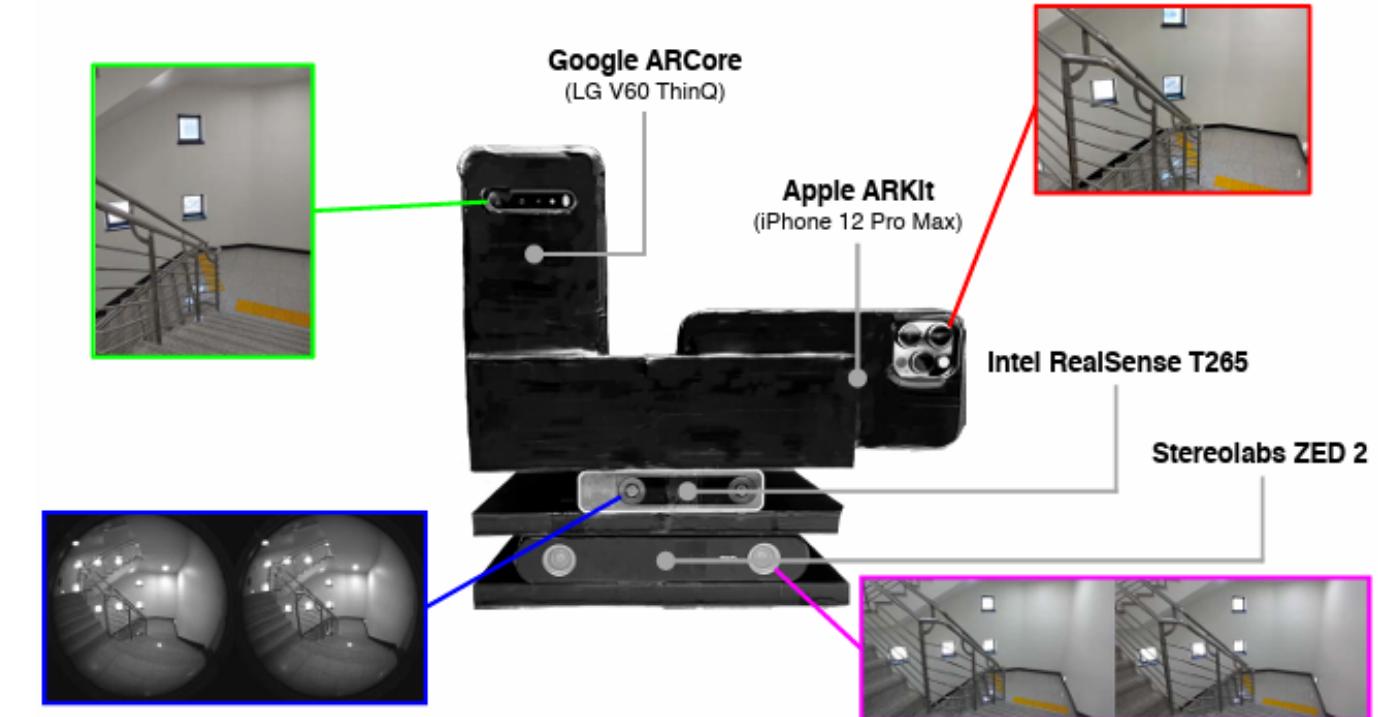
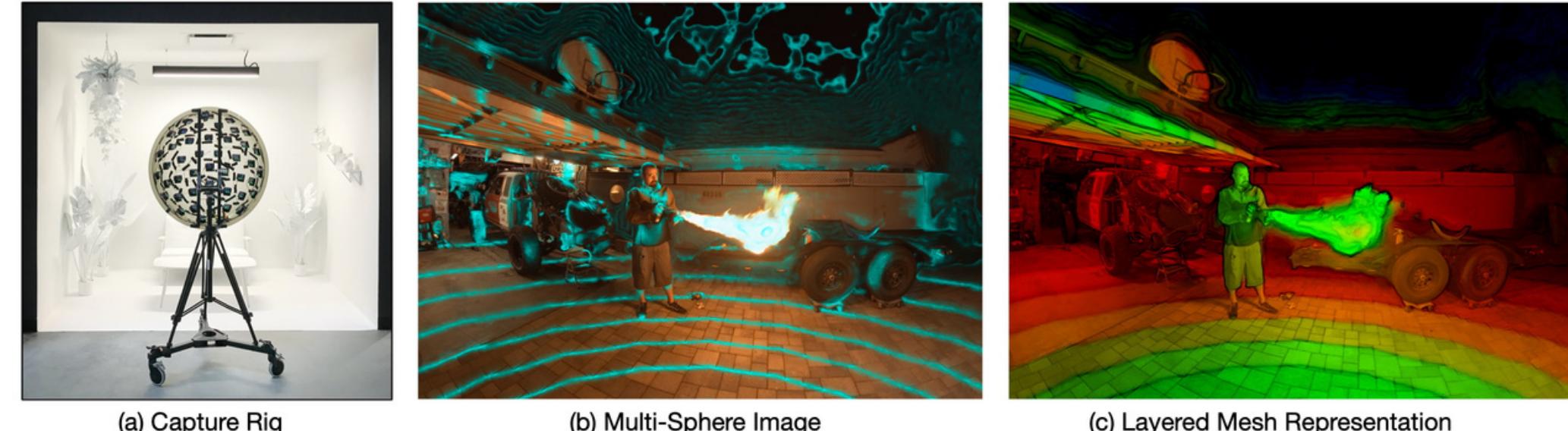


# 이미지 기반 시점 합성

- 이미지 & 정답 카메라 포즈를 입력으로 주어야 함
- 보통은 딥러닝 모델 학습에 이미지 & 정답 카메라 포즈가 포함

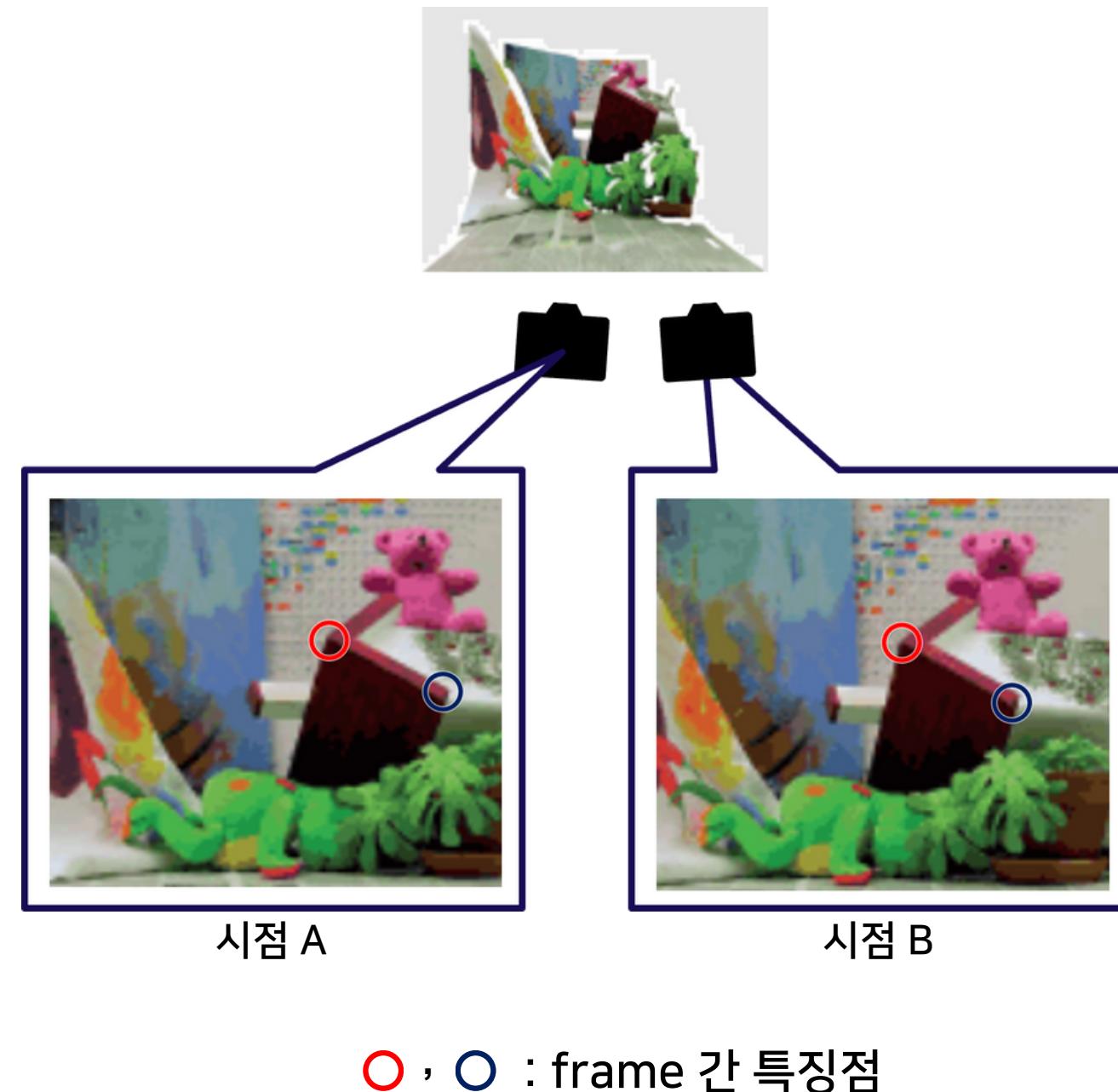
된 벤치마크 데이터를 사용

- 정답 카메라 포즈를 구하기 위해서 사용자에게 ARkit과 같은 추가 api 사용을 요구할 순 없음



# 사용자 입력 데이터 정의

- 사용자는 원하는 공간의 영상을 찍어서 업로드, 영상의 frame들을 여러 시점 이미지로써 추출
- 영상 frame 간 특징점 매칭으로 카메라 포즈 추정
  - 특징점은 서로 다른 시점 이미지 속 같은 점



# 사용자 입력 데이터 정의

- 영상 frame 간 겹치는 영역이 많아야 매칭된 특징점을 활용하여 정확한 카메라 포즈의 예측이 가능함
- 이를 위해 적합한 촬영 가이드를 제시
  - 사용자가 쉽게 따라할 수 있도록, step by step 8자 촬영

영법 제안

**Step 1/3. 공간 준비하기**



- 인테리어 하고자 하는 공간이 포함된 벽면 하나를 정해주세요.
- 촬영 할 때 사람, 반려 동물, 움직이는 물체가 포함되지 않아야 합니다.
- 카메라 초점이 촬영자의 시야 중앙과 일치하도록 맞춰주세요.

**Step 2/3. 카메라 설정하기**

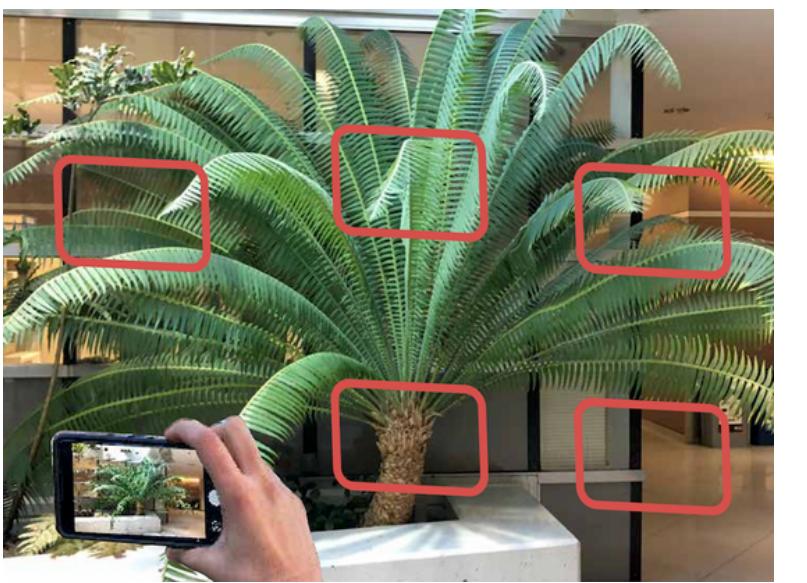


- 스마트폰 기종에 따라 카메라 설정을 다음과 같이 맞춰주세요.
- [삼성 갤럭시]
  - 화면 비율을 1:1 (FHD 30)로 맞춰주세요.
  - 좌측 상단 설정에서 동영상 손떨림 보정, 자동 HDR, 대상 추적 AF 기능을 꺼주세요.
- [아이폰]
  - 해상도는 720p HD를 선택해주세요.
  - 설정 > 카메라 > 비디오 녹화로 이동한 후 설정 가능한 모든 기능(안정화 활성, 자동 FPS, HDR 비디오 등)을 꺼주세요.

**Step 3/3. 공간 촬영하기**

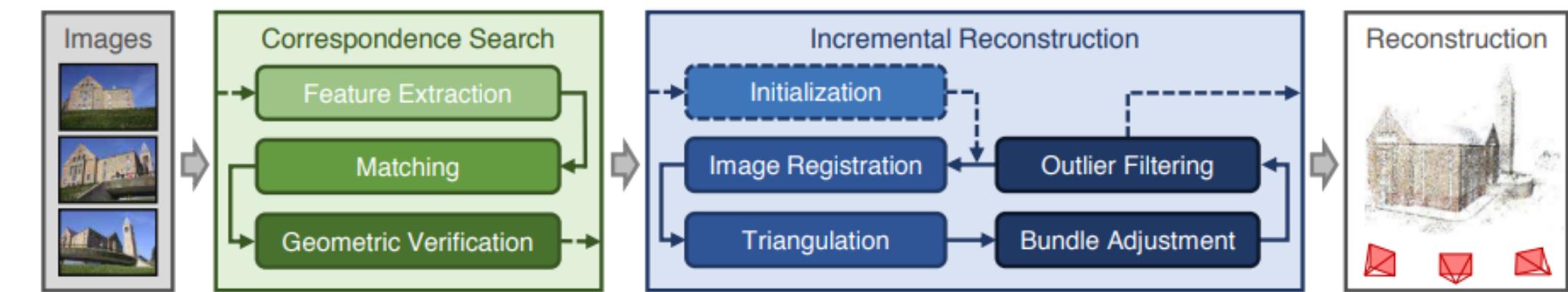


- 공간 중앙에서 카메라를 천천히 8자 모양으로 그리며 20초 간 공간을 촬영합니다.
- 중앙에서 왼쪽 또는 오른쪽으로 조심스럽게 한 걸음 움직이세요.
- 다시 한번 카메라를 천천히 8자 모양으로 그리며 20초 간 공간을 촬영합니다.
- 수고하셨습니다 😊 아래 버튼을 눌러 업로드 해보세요!



# DL을 활용한 데이터 전처리

- 카메라 포즈 추정에 Structure-from-Motion (SfM) 기법 사용
- 주로 COLMAP이라는 rule-based 기법이 많이 활용됨
  - SIFT로 특징점을 추출하고, RANSAC 최적화로 적합한 매칭쌍을 찾음
  - PnP로 카메라 포즈를 추정하고, BA는 reprojection error가 줄어들도록 포즈를 refine



# DL을 활용한 데이터 전처리

- 적합한 매칭쌍을 찾는 최적화 기법 RANSAC이 같은 입력에 대

해서 항상 같은 결과를 보장하지 않음

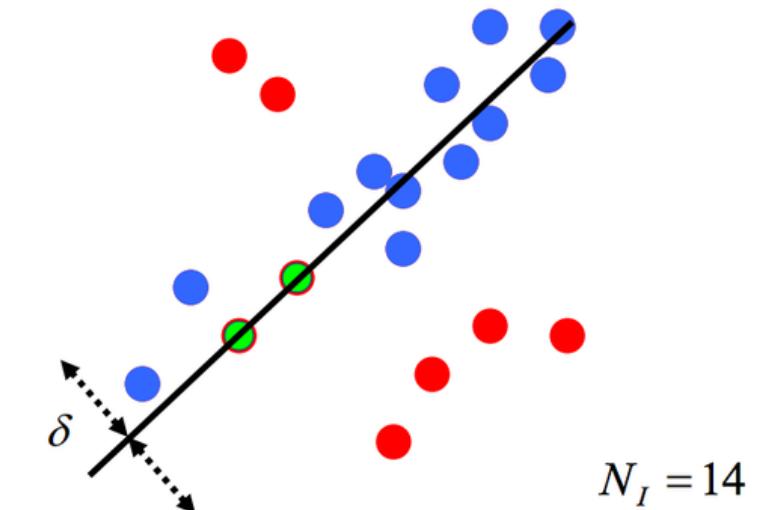
- Outlier 필터링에 사용되는 랜덤 샘플링 결과 상이

- 매칭 결과가 불안정한 문제 발생

⇒ 다른 시행에서 100% 매칭이 가능했던 영상에 대해

5%의 매칭 결과를 내기도 함

RANSAC



Algorithm:

1. Sample (randomly) the number of points required to fit the model (#=2)
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

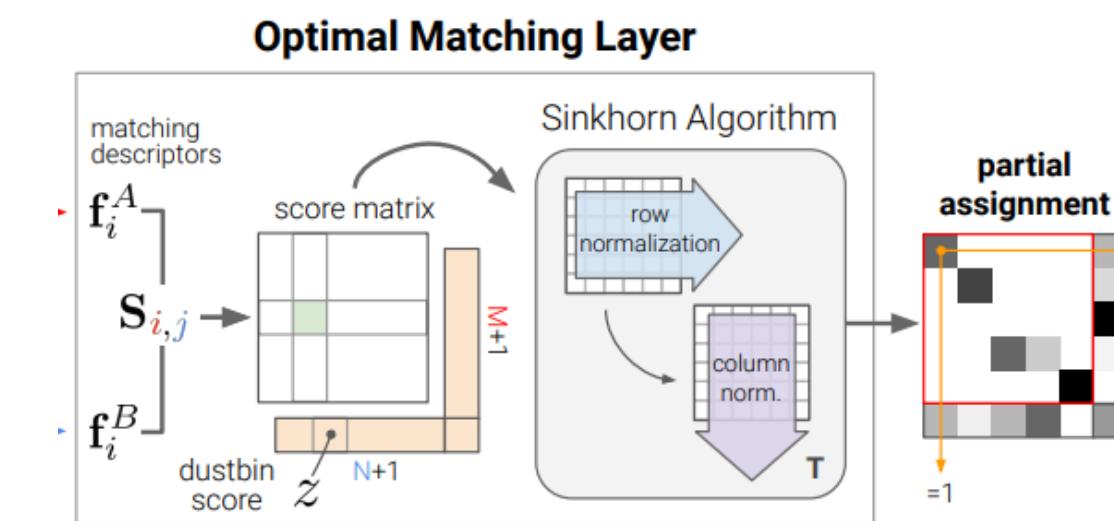
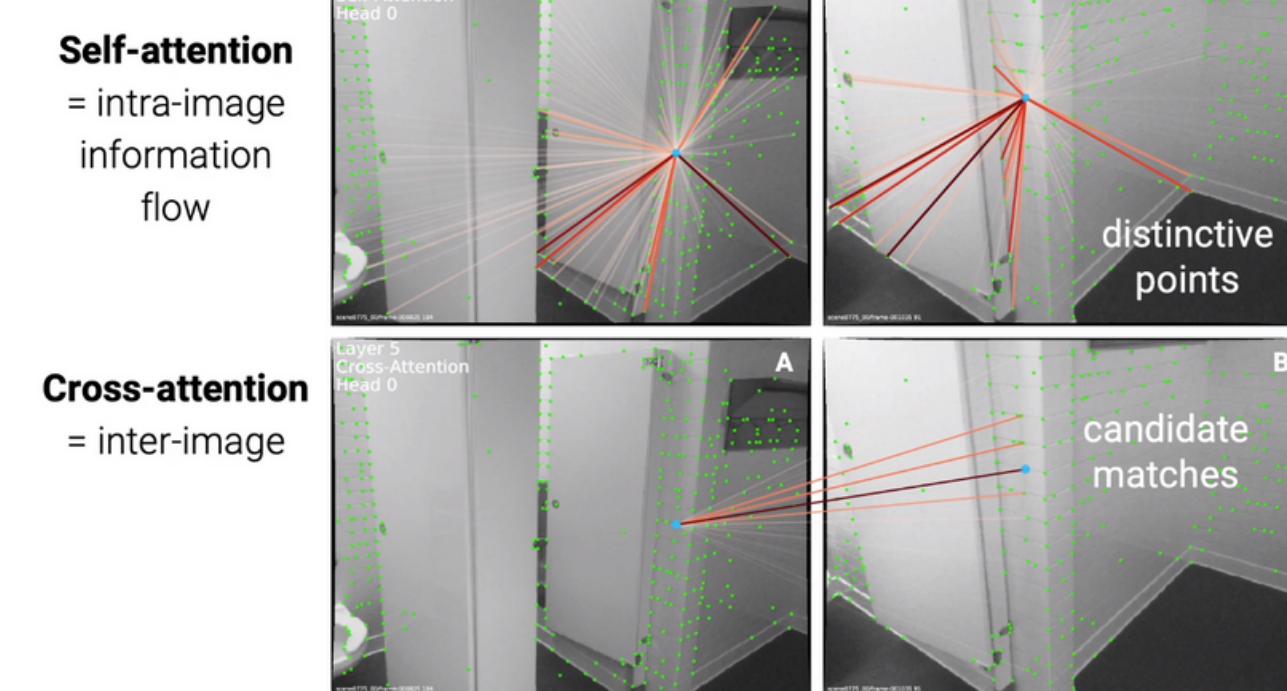
Repeat 1-3 until the best model is found with high confidence

RANdom SAmple Consensus 최적화

| 같은 입력 영상에 대한 RANSAC 매칭 결과 |  |
|---------------------------|--|
| 시행 A                      | COLMAP found poses for <b>all</b> images, CONGRATS!                  |
| 시행 B                      | COLMAP only found poses for <b>5.76%</b> of the images. This is low. |

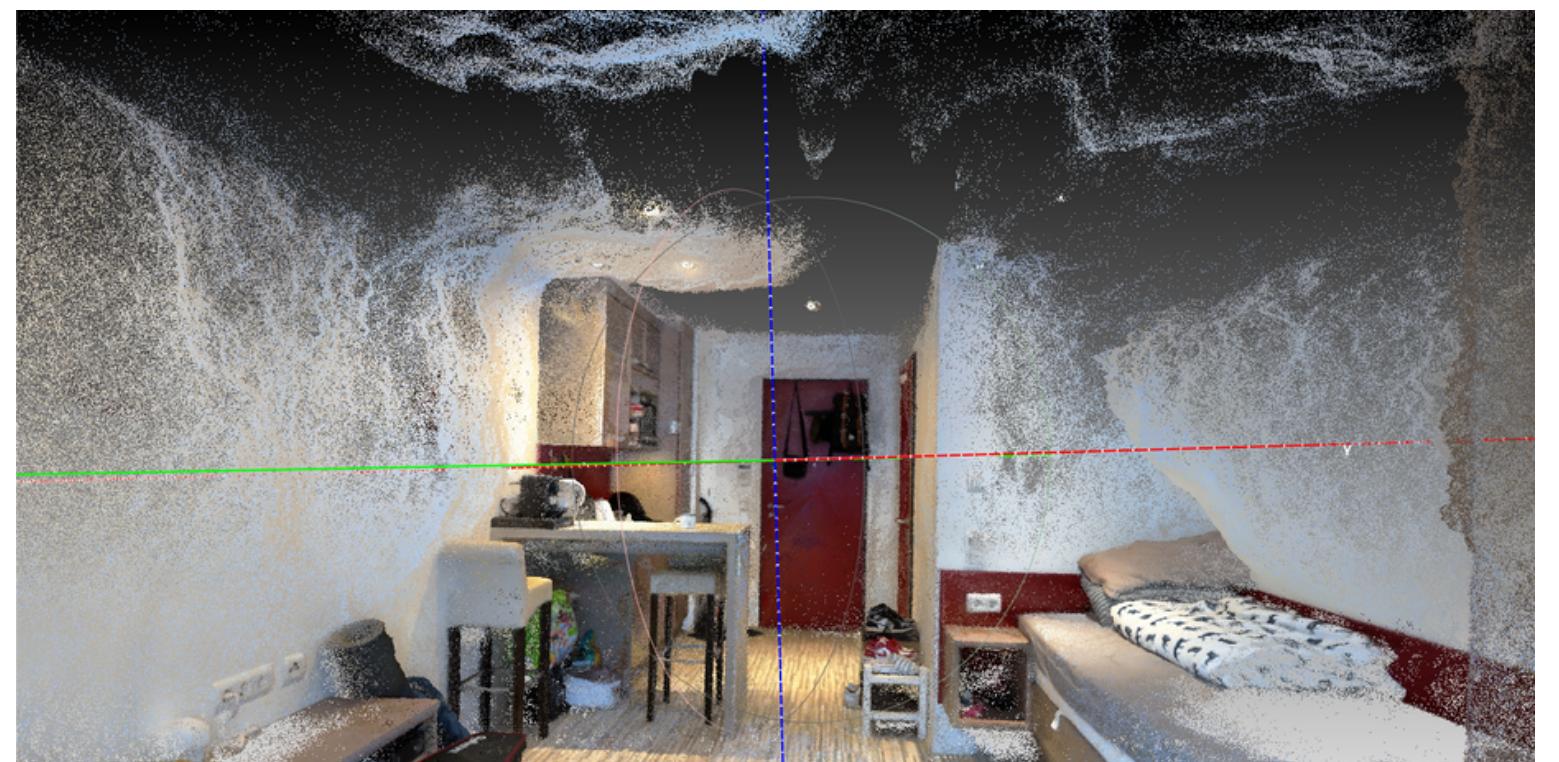
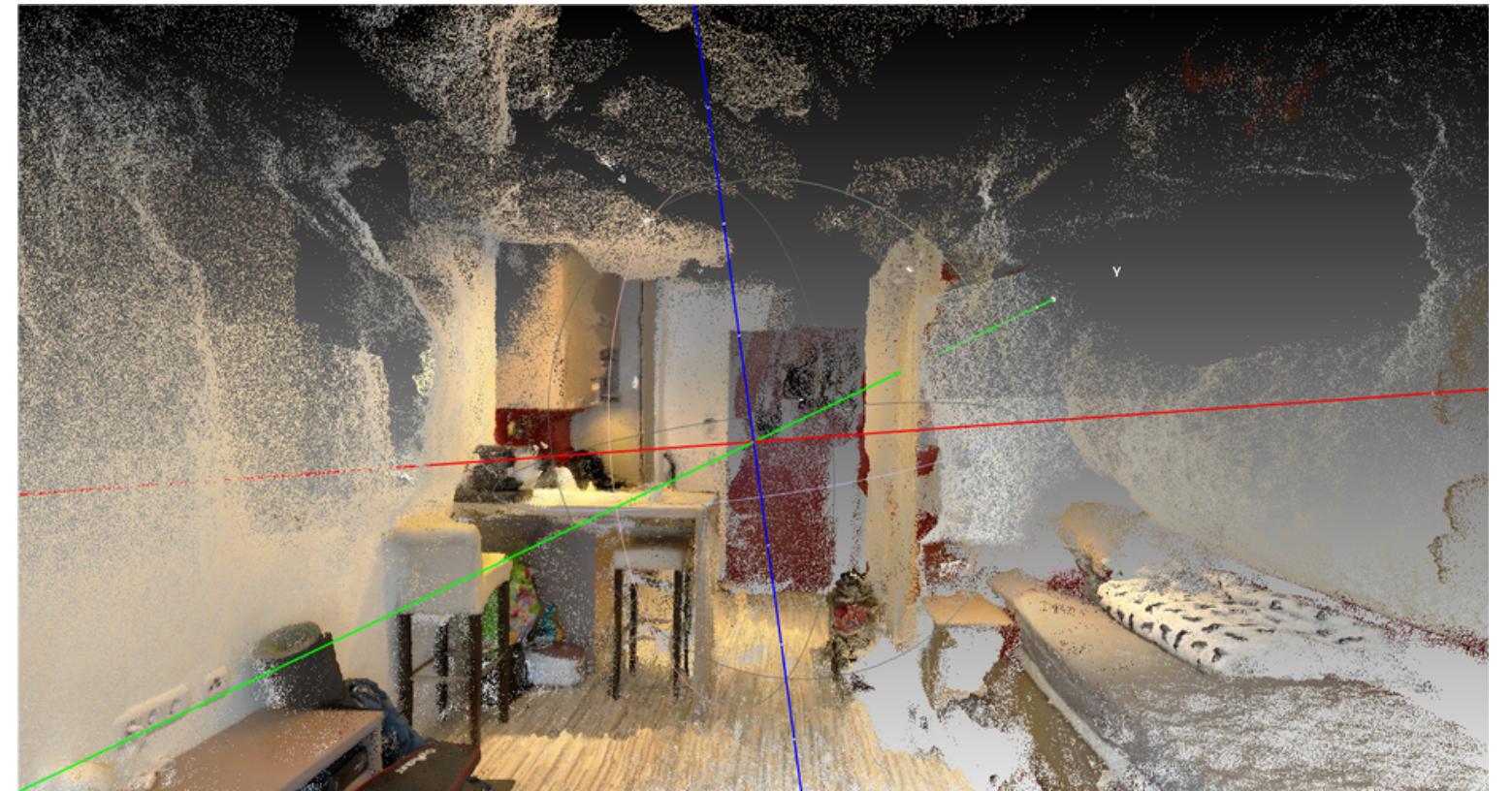
# DL을 활용한 데이터 전처리

- 카메라 포즈 추정 기법을 딥러닝 기반 기법으로 대체
- 딥러닝 기반 SfM: Hierarchical localization (HLOC)
  - 매칭이 잘 이루어질 수 있는 특징점들을 추출하는 SuperPoint
  - Attention 기반으로 특징점들을 매칭하는 SuperGlue
    - 적합한 매칭쌍을 찾는 연산에 Sinkhorn 알고리즘이 적용되어 GPU 상에서도 연산 진행 가능



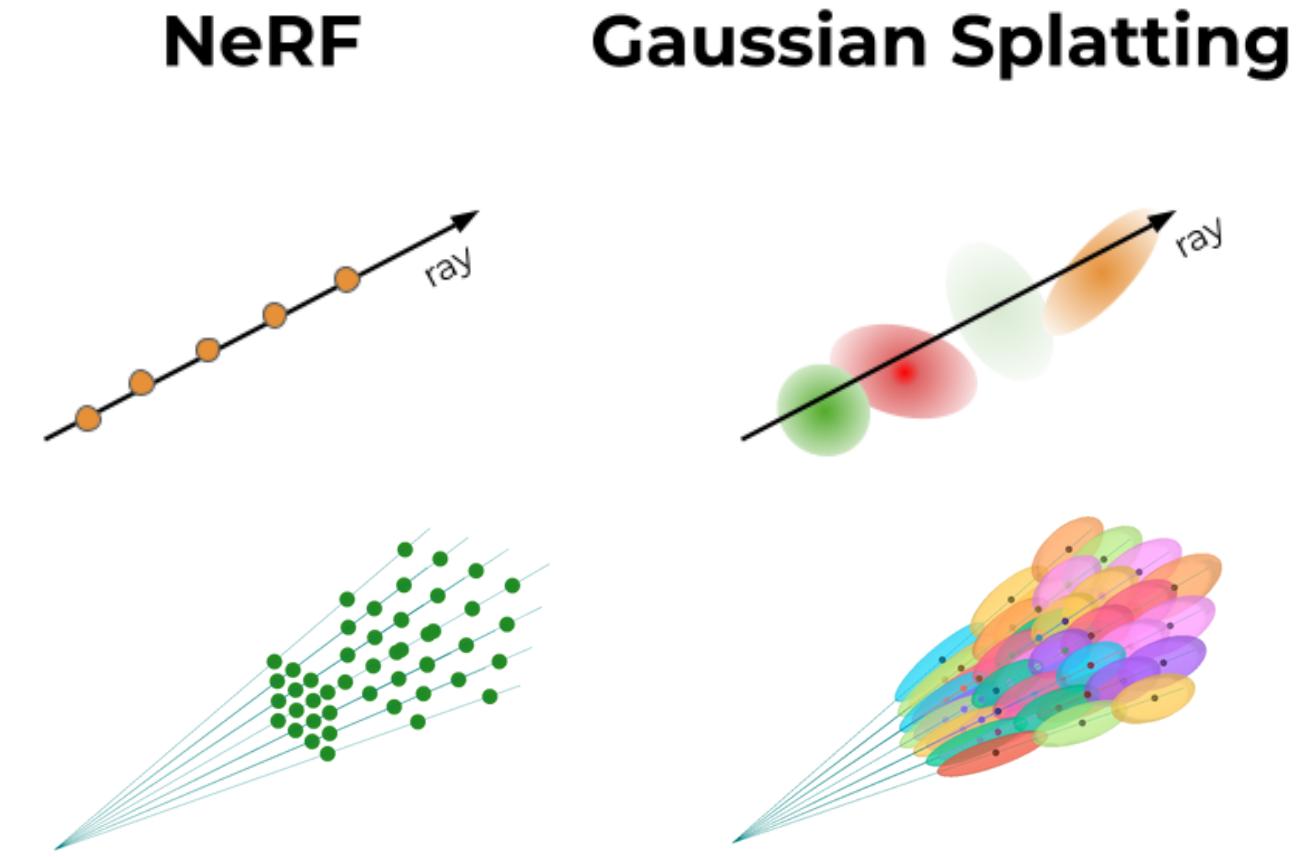
# DL을 활용한 데이터 전처리

- 실내 공간 벤치마크 데이터셋 ScanNet++의 valid scene 30  
개로 유효성 검증
- 딥러닝 기반 SfM인 HLOC로 변경 후 성능 향상 효과
  - SfM 매칭률 평균 27%  $\Rightarrow$  98%
  - HLOC 전처리 후 공간 재구성 모델 학습 시, 정량적인 지표 향상, 정성적인 결과 개선
    - PSNR(↑) 13%, SSIM(↑) 15% 증가
    - LPIPS(↓) 7% 감소



# 노이즈 데이터에 적합한 공간 모델 선정

- 공간 모델 선정에 있어 생성 퀄리티, 런타임 모두에 강점을 가지는 최신 모델들로 후보를 추림
  - 3차원 구조를 point로 정의하는 NeRF 구조의 Nerfacto
  - 3차원 구조를 gaussian으로 정의하는 gaussian splatting 구조의 Splatfacto
- 노이즈가 적은 벤치마크 데이터셋 기준으로 두 모델의 퀄리티는 비슷, 런타임은 Splatfacto가 더 유리

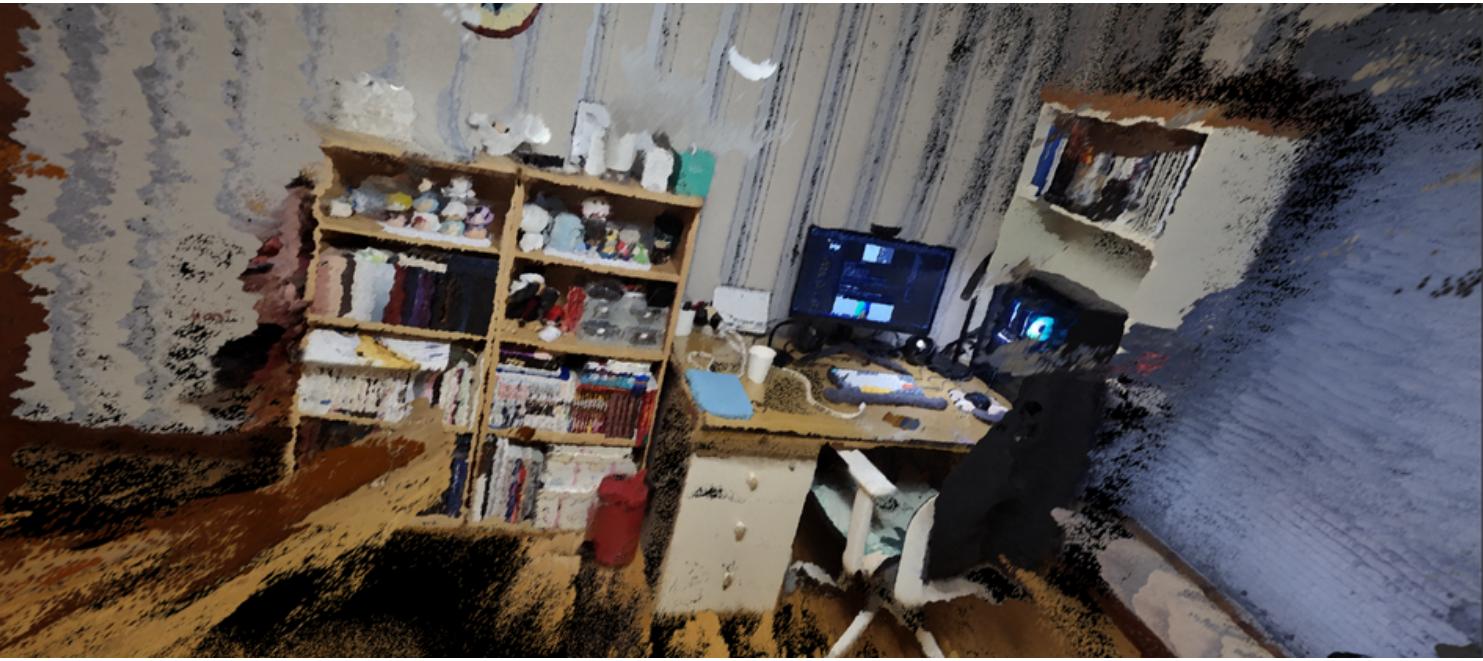


| per<br>1 scene train    | Nerfacto             | Splatfacto      |
|-------------------------|----------------------|-----------------|
| 생성 퀄리티<br>(PSNR / SSIM) | <b>27.98</b> / 0.800 | 27.21 / 0.815   |
| 런타임                     | ~ 20 min             | <b>~ 10 min</b> |

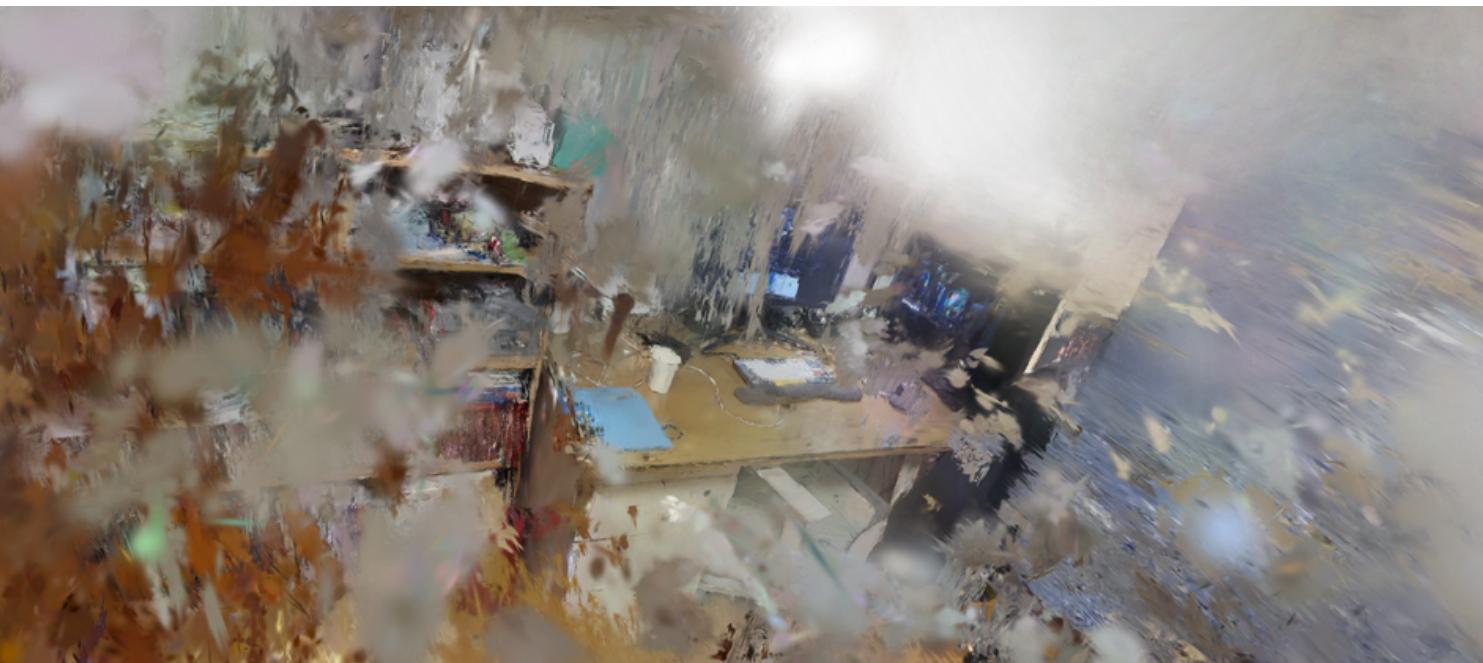
# 노이즈 데이터에 적합한 공간 모델 선정

- SfM으로 추정한 pseudo 포즈를 사용함에 따라 올바르지 않은 geometry가 학습되어 3차원 공간 상 떠다니는 점과 같은 floater 발생
- Splatfacto (Gaussian 구조)에서 floater의 영향이 더 크게 나타나 최종 재구성 결과 퀄리티 저하  
⇒ Point 형태로 3차원 구조를 정의하는 NeRF 구조 모델  
mipNeRF360의 구현체인 Nerfacto 사용

**NeRF**

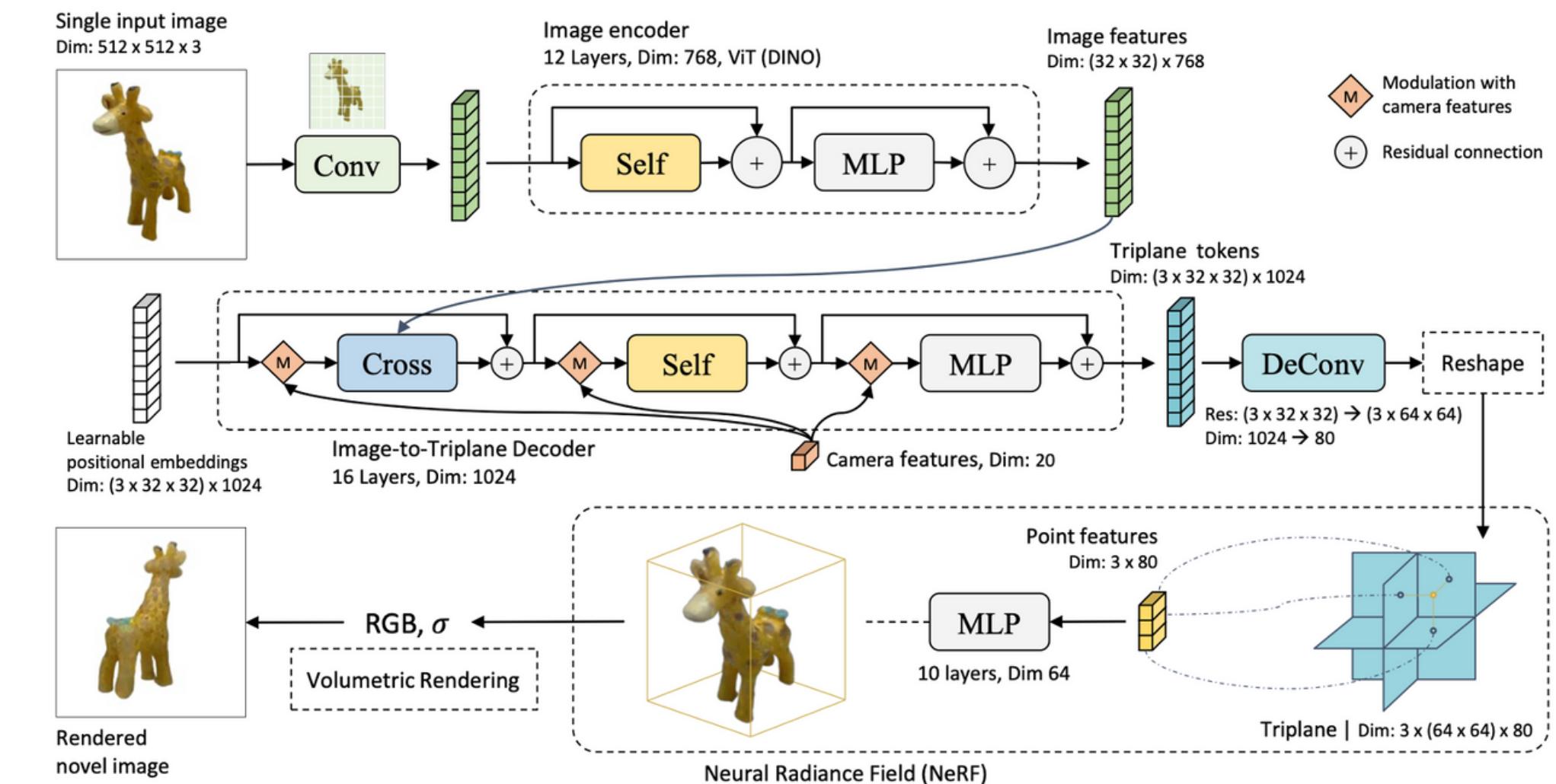


**Gaussian  
Splatting**



# 2D 이미지를 3D Triplane 변환

- 사전 학습된 DINO(Decomposition Into Image and Noise)를 이용하여 이미지 patch별 특징 벡터로 변환
- Transformer decoder를 통해 이미지 특징을 cross-attention 기법으로 3D triplane(3D 공간에서 객체의 형태를 표현하는 평면)과 연관성을 학습
- => 이미지 특징을 학습 가능한 공간과 위치 정보를 가지는 벡터로 변환하여 이를 triplane 표현으로 변환



# 입력 데이터 정의 및 전처리

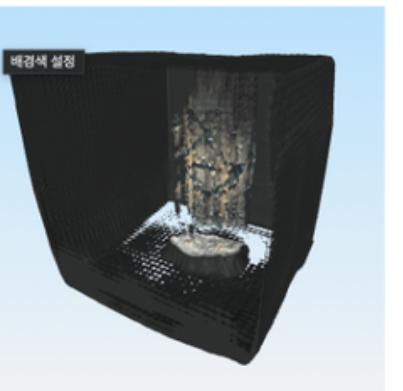
- 단일 이미지로 3D 모형 예측 시, 가장 결과가 좋았던 정면  
+-45도 까지만 회전된 물체만 사용하길 권장
- 다양한 기종의 스마트폰에서 촬영하고 바로 올릴 수 있도록 함
- 이미지 내 여러 물체 중 핵심 물체만 탐지하기 위해 YOLO를 이용한 객체 탐지 및 rembg를 활용한 배경제거

- 3D로 생성할 가구를 정중앙에 두고 사진을 찍어주세요.
- 되도록 뒤의 배경에 다른 사물이 없도록 해주세요.  
(한번에 한 가구만 바꿀 수 있습니다.)

## 촬영 주의사항 안내



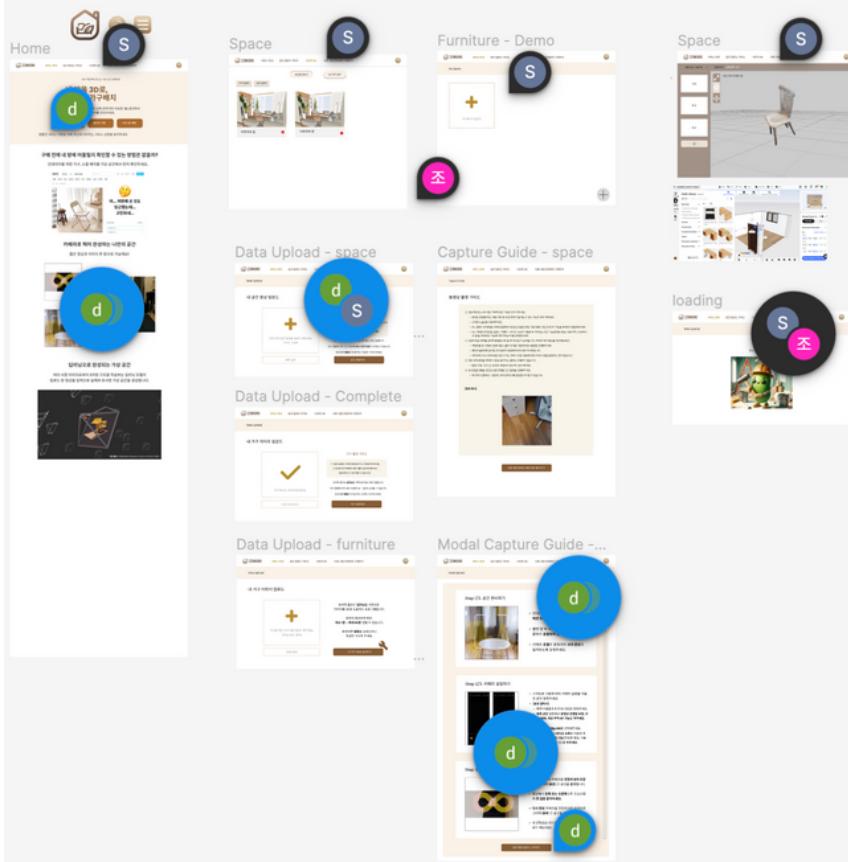
전처리 X  
→



## 실제 변환 결과

# 3. Product Serving

- 
1. 프론트엔드
  2. 백엔드
  3. 서비스 아키텍쳐

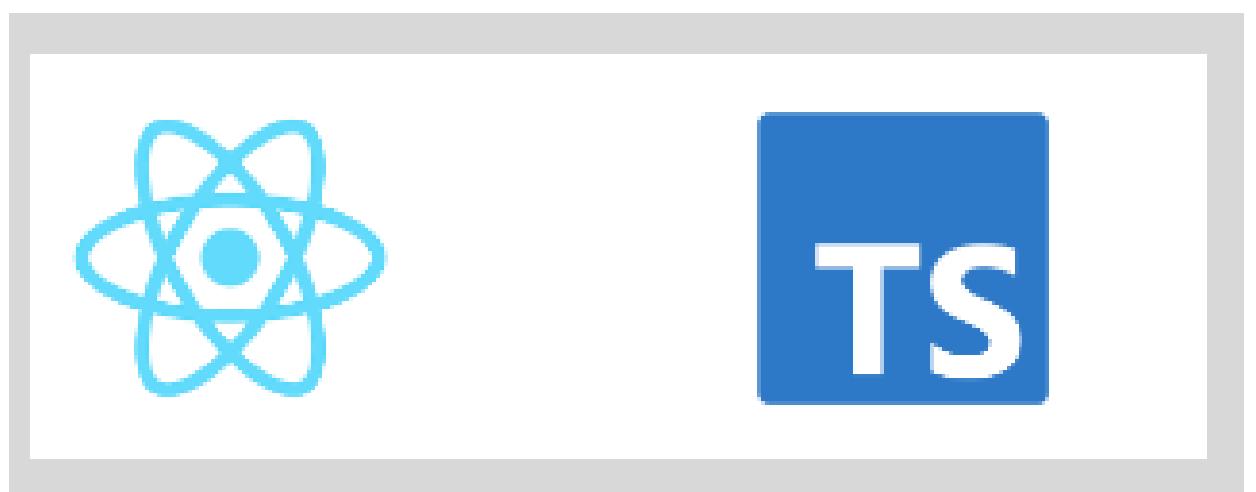
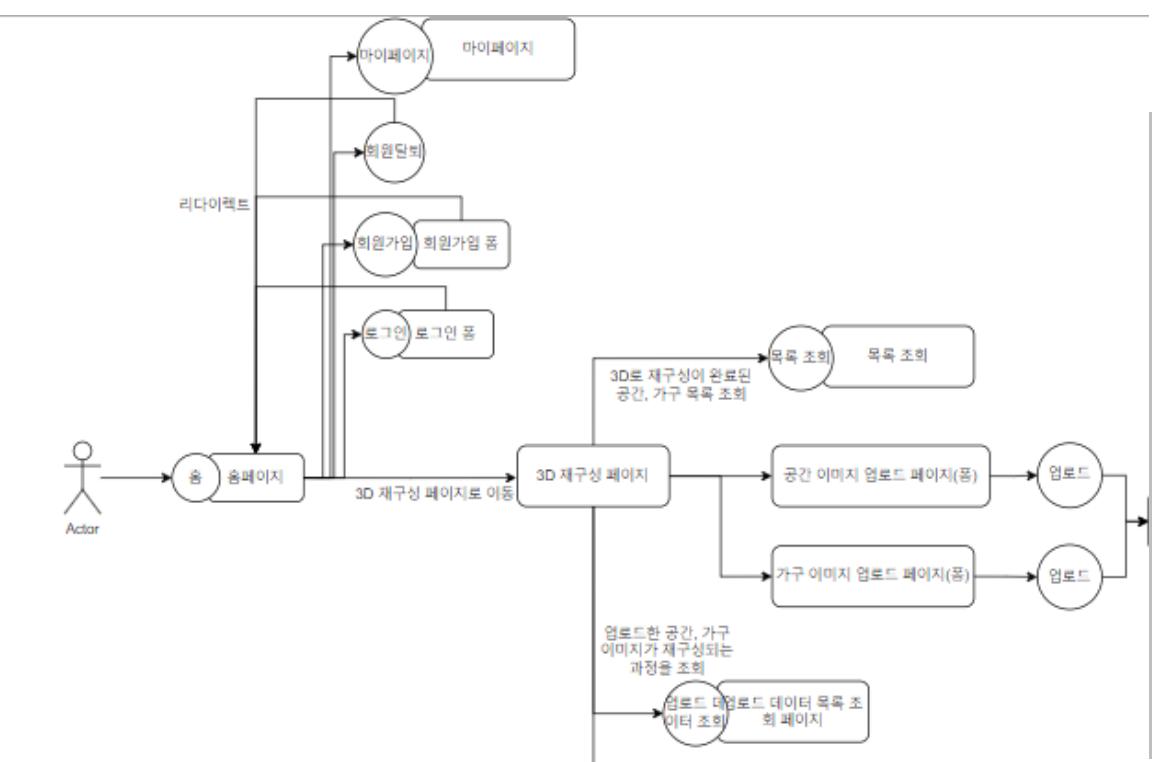


#### figma를 이용한 기획, 디자인

사용자 측면을 고려하여 심플하고 직관적인 디자인 설계

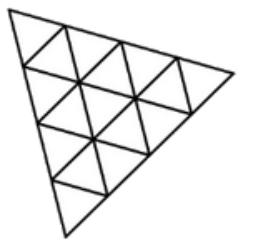
#### 페이지 이동 설계

유저 시나리오를 시스템 흐름에 맞게 설계



# Rendering

WebGL 기반의 three.js 이용



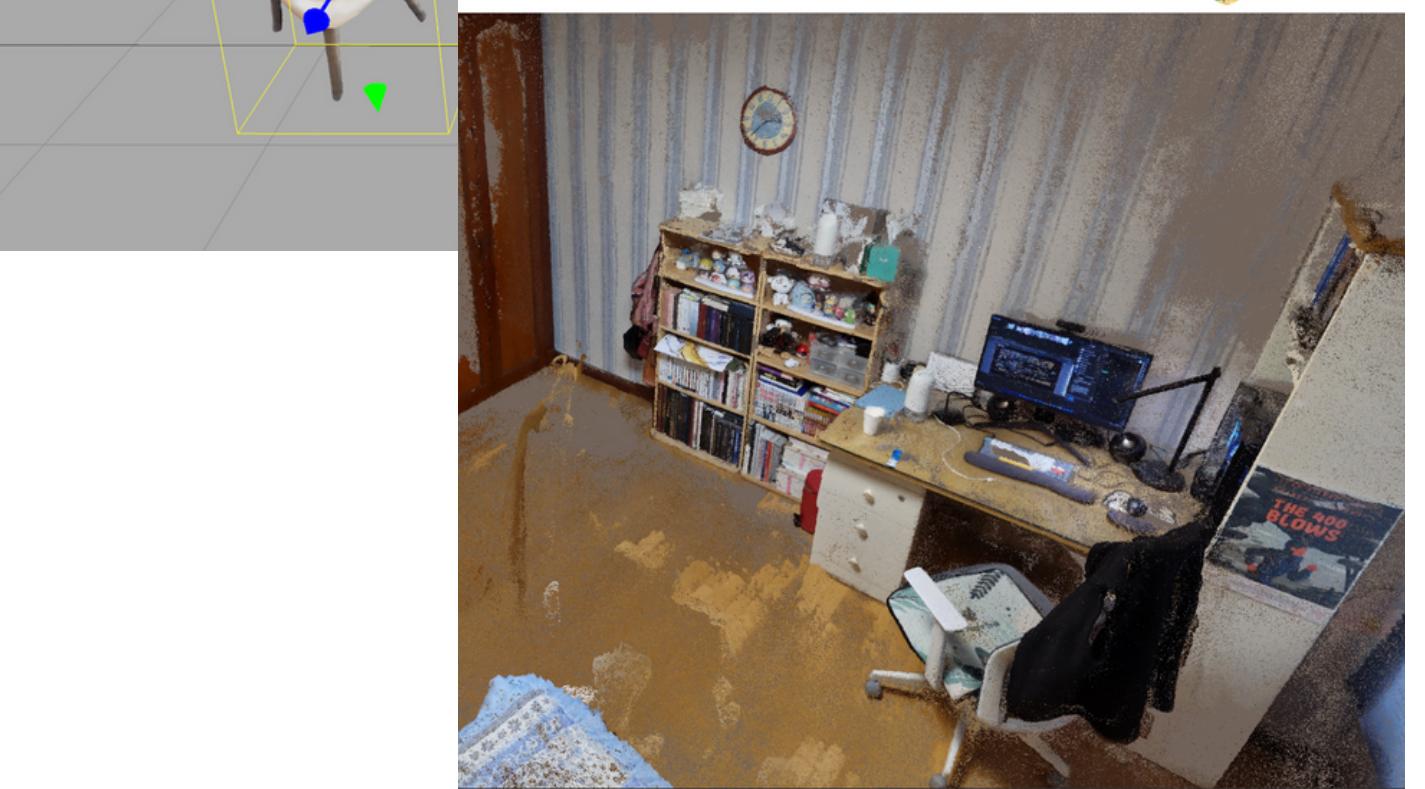
Three.js

모델의 인퍼런스 결과인 3D 오브젝트 ply 파일을 웹 상에서 렌더링 하여 보여주는 것이 필요

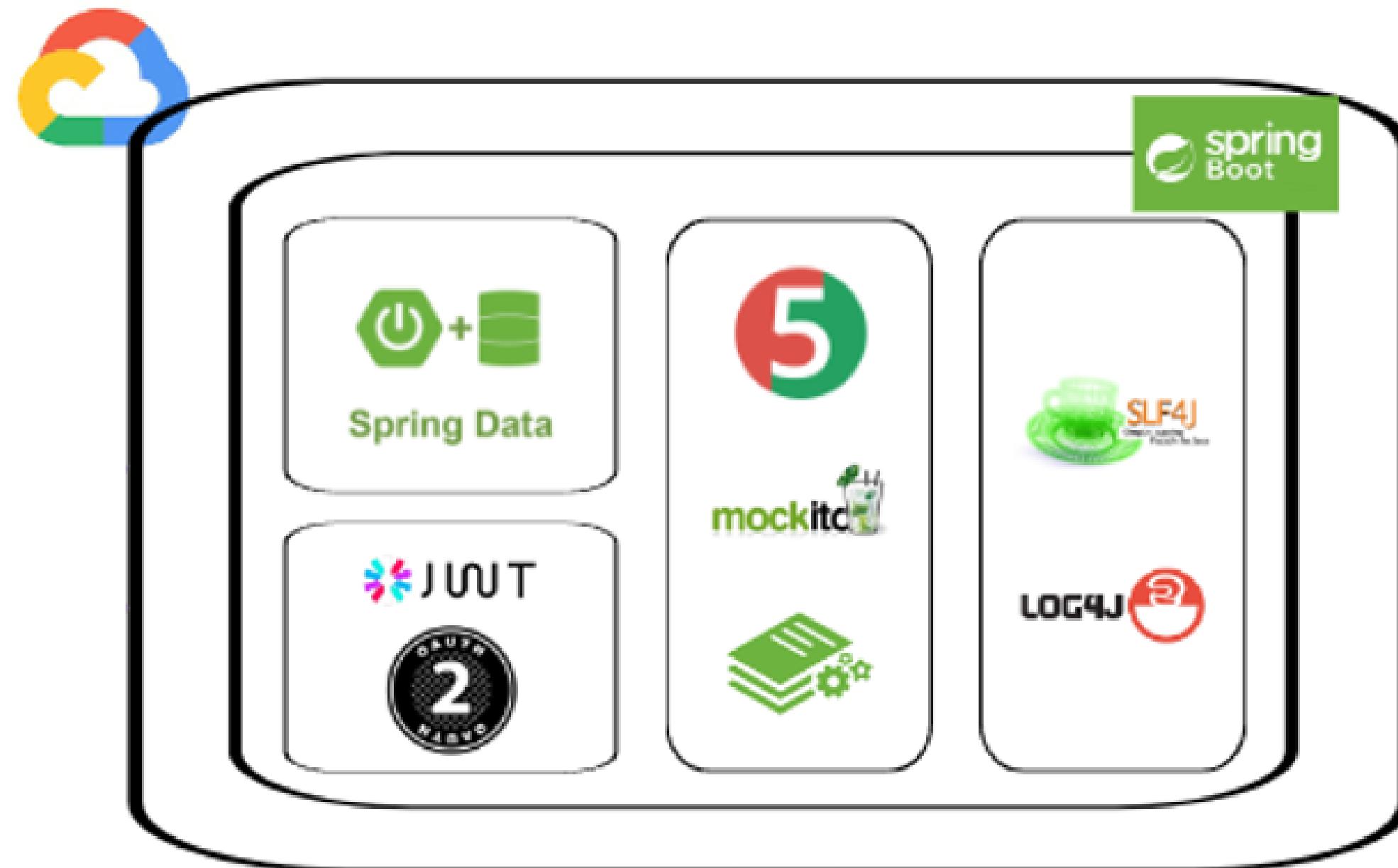
- 다양한 각도에서 결과를 확인할 수 있도록 컨트롤 설정

유저의 공간에 다양한 가구 오브젝트를 배치하도록 인테리어 배치 시뮬레이터 구현

- 공간의 인퍼런스 결과가 xyz축에 평행하지 않을 수도 있기에 방의 바닥 예측 필요
- 가구를 유저의 임의로 움직이거나 크기 변화, 회전이 가능하도록 컨트롤 기능 추가
- 몰입감과 편리함을 위해 1인칭 시점 뷰어 설정

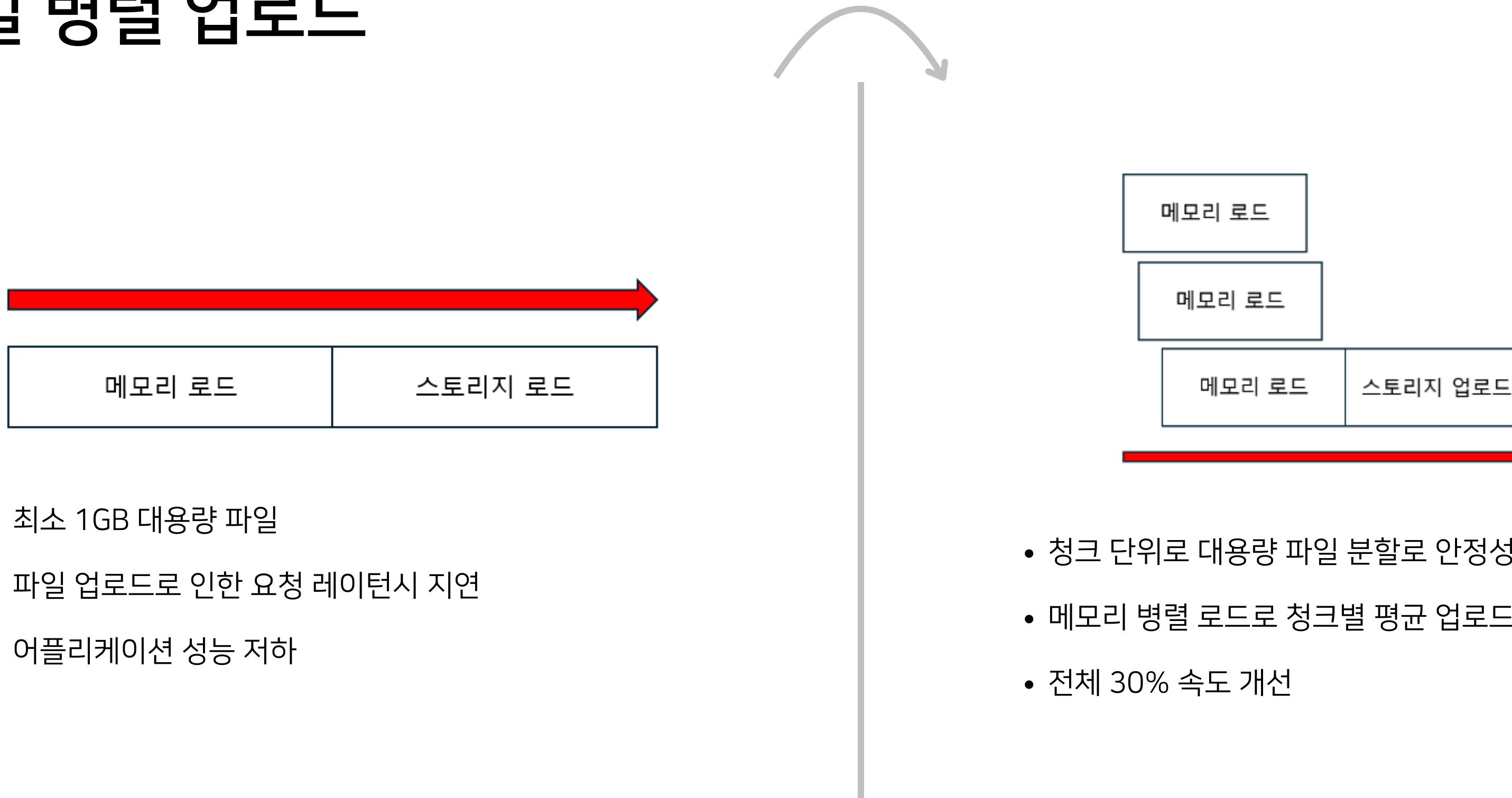


# Server FrameWork



어플리케이션 성능을 위한 컴파일 언어 웹 프레임워크 사용

# 파일 병렬 업로드



# 배치 스케줄링을 활용한 메일 일괄 푸시



- 재구성이 완료된 항목 존재 시 배치 스케줄링을 활용해 일괄 메일 시스템 구현
- 특정 단위 시간에만 메일을 보내도록 설계하여 어플리케이션 성능에 영향을 줄임
- 이때 발생하는 여러 건의 update 쿼리는 벌크 연산으로 성능 개선
- 서버 용량 부족, 부적절한 업로드 파일 등 서버 오류나 사용자 오류에 의한 에러 모니터링

# 사용자 선호도 파악을 위한 로깅



- 유저 로그인 시간 + 로그아웃 시간
- 세션 timeout 여부 → 자의로 나간 것인지 판단 여부
- 가구 및 공간 데이터 접근 시간 및 횟수 카운팅
- 추론 서버별 추론 시간 로깅 → 평균 추론 시간 파악 및 오류난 서버 파악

- 서비스 활용도를 파악하기 위한 로그 체크리스트 체크
- 사용자 패턴 및 선호도를 파악하여 커뮤니티 활성화 및 모델 학습 강화

```

GET /api/noticeboards/1 args=[LoginMemberRequest(id=4), 1]
----> NoticeBoardController.findNoticeBoard(..) args=[LoginMemberRequest(id=4),
--<-- NoticeBoardController.findNoticeBoard(..) time=58ms
GET /api/noticeboards/1 args=[LoginMemberRequest(id=4), 1]
----> NoticeBoardController.findNoticeBoard(..) args=[LoginMemberRequest(id=4),
--<-- NoticeBoardController.findNoticeBoard(..) time=17ms
GET /api/noticeboards/2 args=[LoginMemberRequest(id=4), 2]

```

- 스프링 AOP를 활용하여 단건 조회 메서드만 추적  
(일간 가장 인기 있는 가구, 개인별 선호 가구 체크 가능)
- 유저 트랜잭션, 메서드 실행 시간 체크
- 스프링 log4j2를 활용한 로깅 성능 개선

## 모델 서빙

- 6대의 분산 추론 서버 존재
- 각각 NCP 클라우드에 적재 후 사용

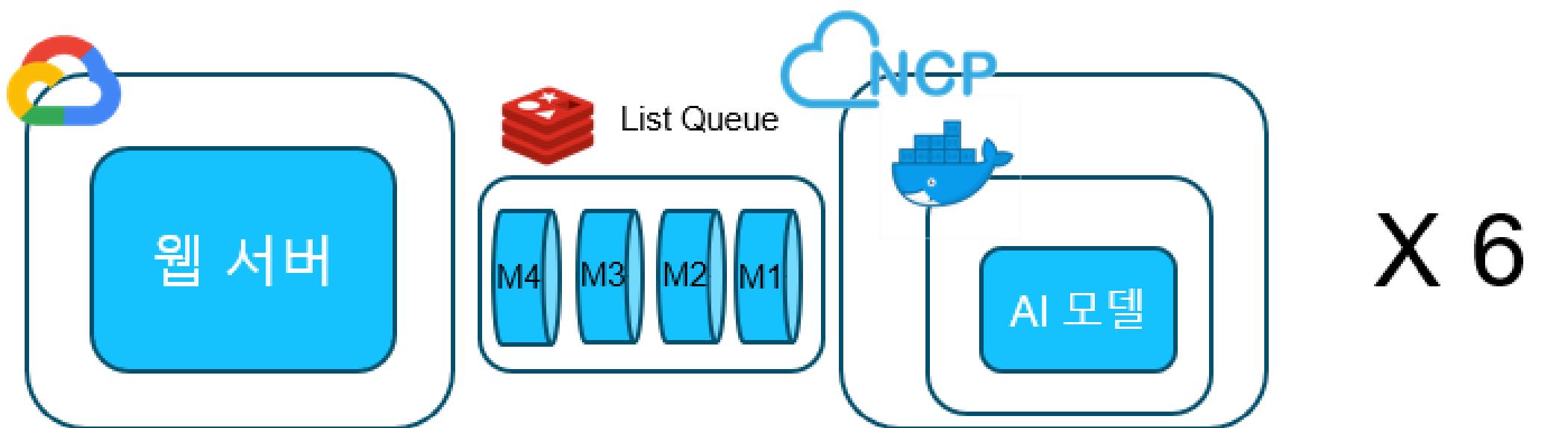


## 분산 환경 GPU 스케줄러 구현

- 웹 서버의 요청을 실시간으로 처리하는 스트림 서빙 상황
- 요청과 반환, 대기의 반복 수행



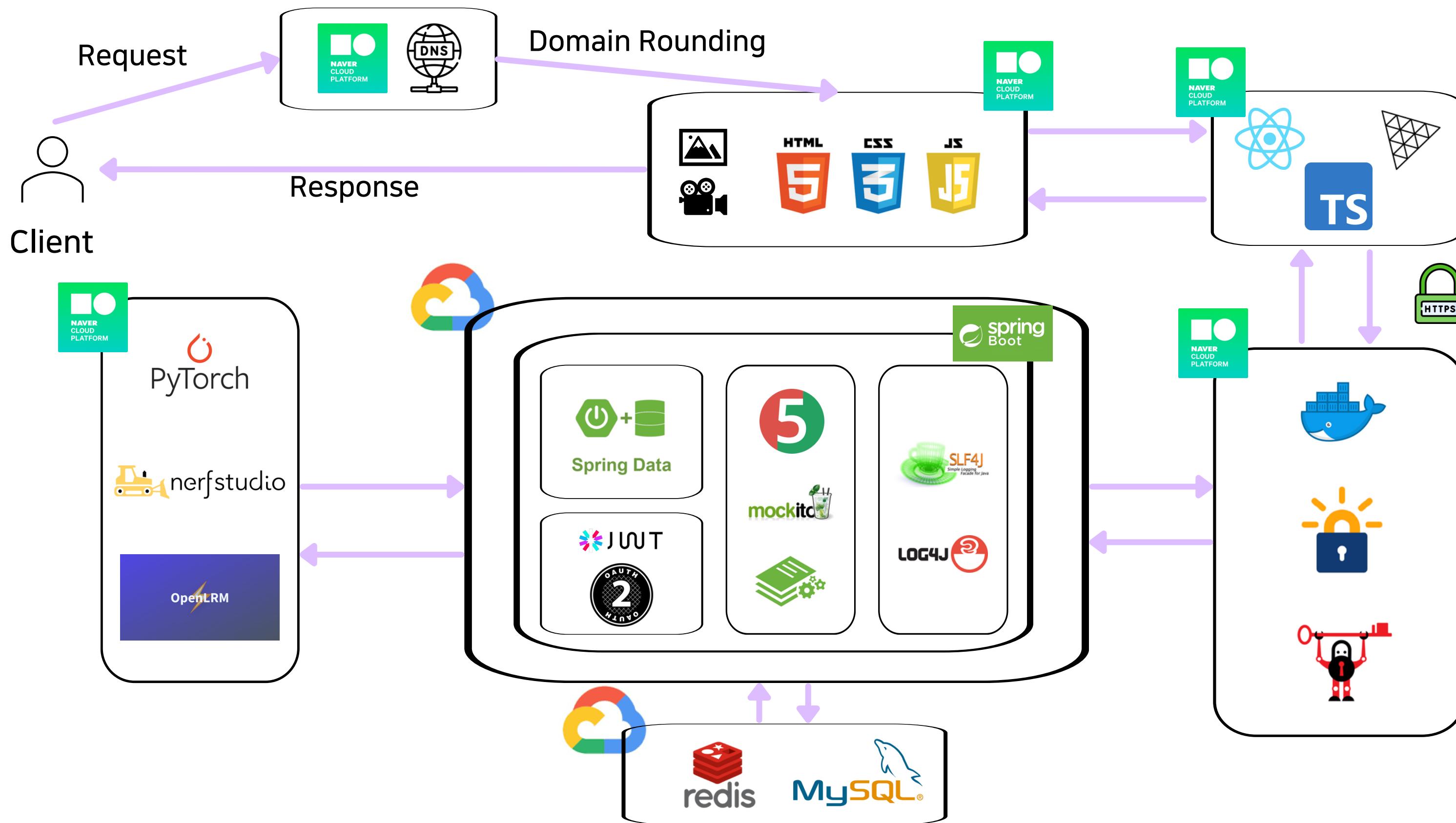
# 분산 환경 GPU 스케줄러 구현

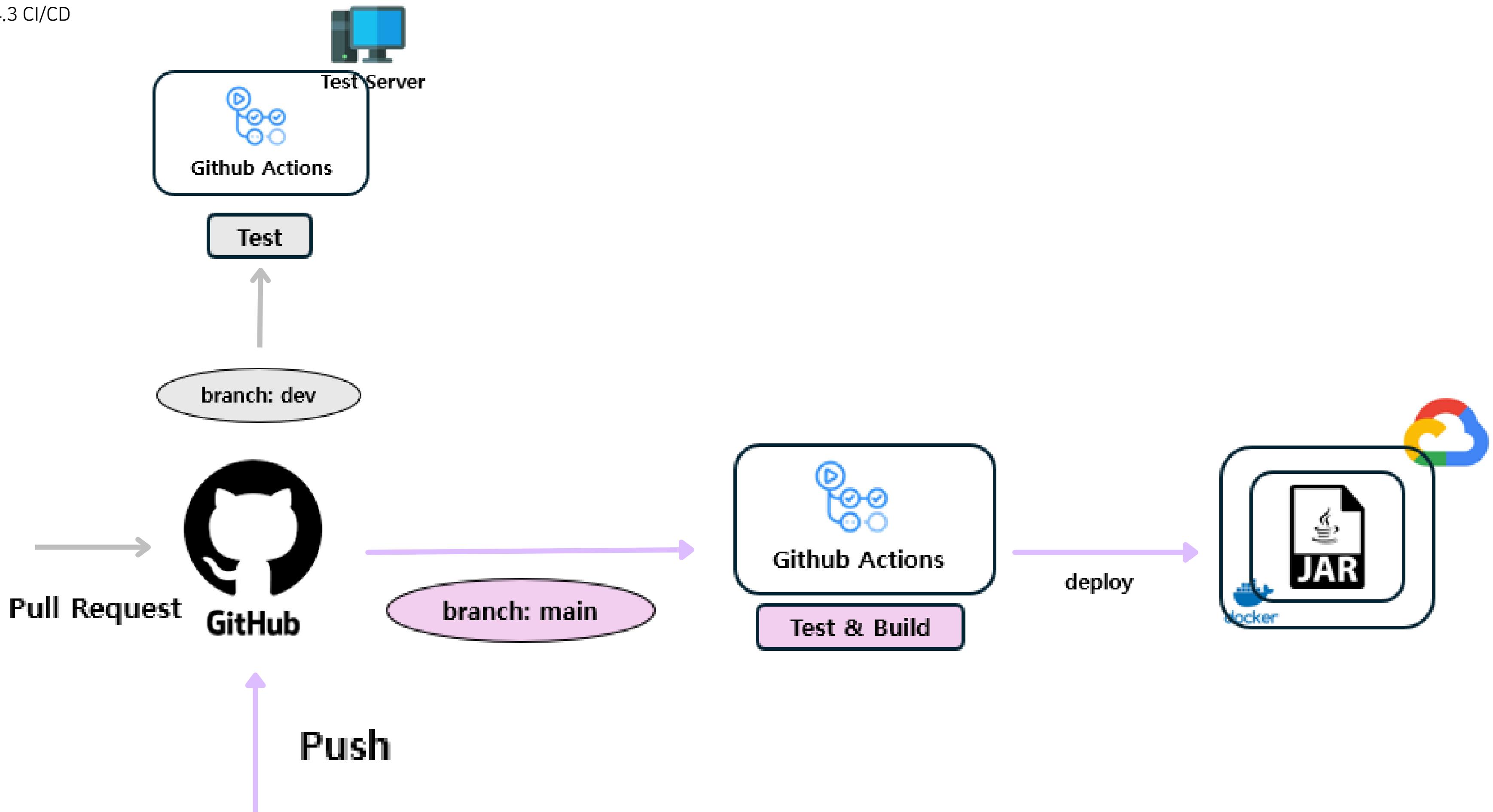


&lt;Queue Message&gt;

```
config: {
    id: number, #학습 테이블 고유번호
    objectType: bool, #가구인지, 공간인지
    src: string, #동영상, 이미지 파일 주소
}
```

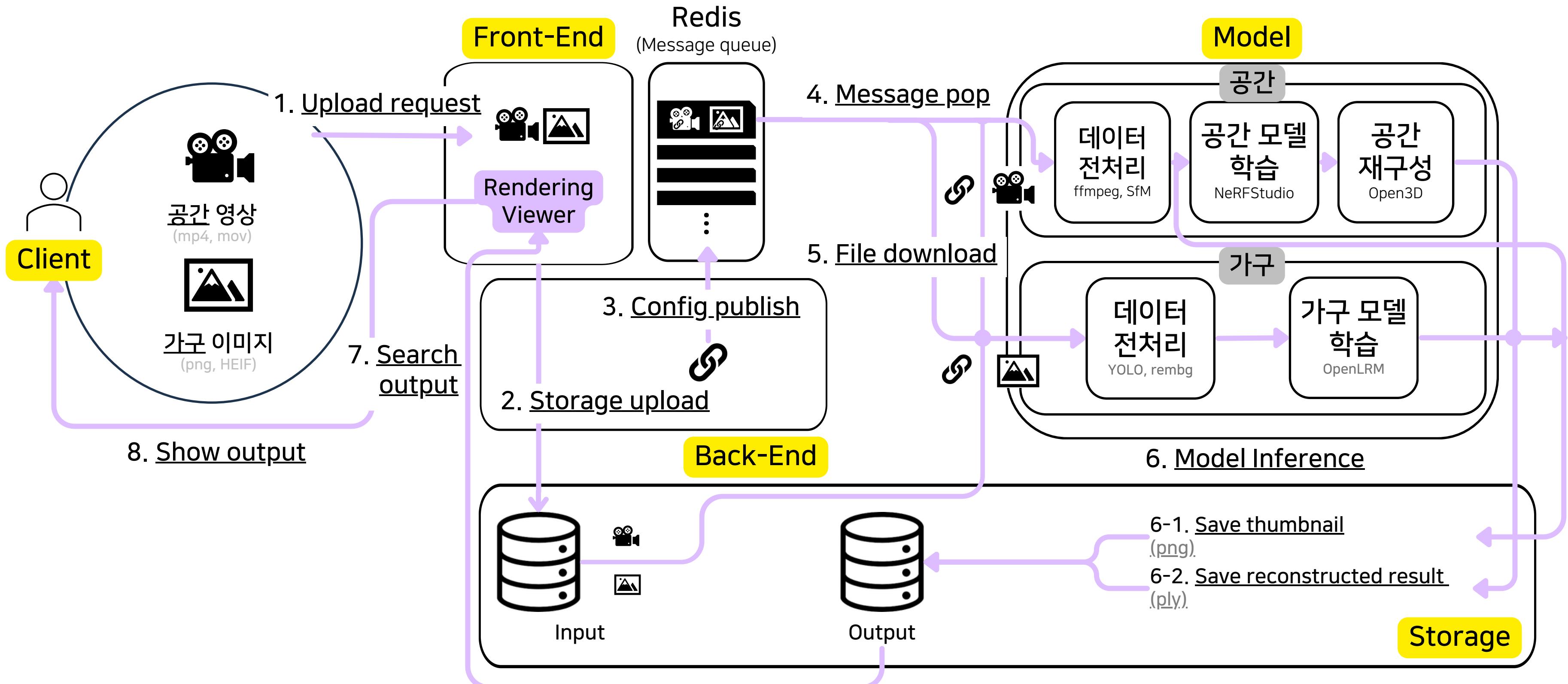
- Redis List를 활용하여 스케줄러 구현
- 단일 쓰레드 읽기와 누적 메모리 효과로 분산 환경에 적절하다고 판단
- 가구와 공간 모델 포맷에 맞는 메세지 pop & push로 스트림 서빙 수행





# 4. Conclusion

- 
1. 서비스 파이프라인
  2. 서비스 아키텍쳐
  3. 데모 시연 영상
  4. 자체 평가 의견



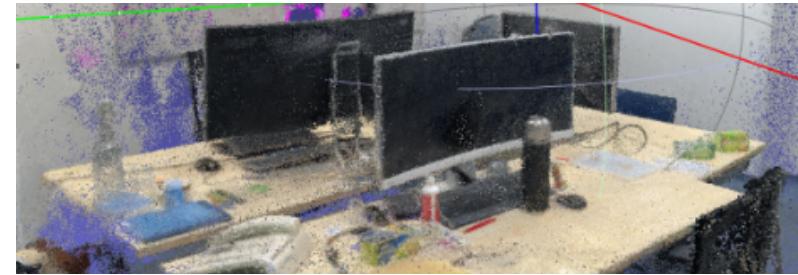
## Model

- 공간 모델

- Point 기반의 공간 모델을 사용하다 보니 point cloud 대비 재구성된 mesh의 퀄리티가 아쉬웠다.
- ⇒ Mesh refine을 위한 방법론 추가 검토를 통한 퀄리티 개선이 필요하다.

- 가구 모델

- Gaussian Splatting기반 모델 적용이 어려워 아쉬웠다.
- ⇒ Point to Mesh로 변환할 수 있는 간단한 기법 연구가 더 필요하다.



Point cloud



재구성된 mesh

## Product Serving

- FE, Rendering

- Mesh의 Face 및 방의 벽 정보가 있다면 더나은 뷰어를 만들 수 있을 듯 하다.

- BE

- MLOps를 위한 CT를 시도하지 못한 것이 아쉽다.
- ⇒ Airflow와 MLflow로 스케줄링 최적화 검토가 필요하다.

---

**End of Document**

**Thank You.**