'LumiAR' 상세 설계서

컴퓨터공학 종합설계[202502-CSE4205-001]

그리마	팀장	정현민(12201392)
	팀원	최민석(12201812)
	팀원	박준희(12201833)

목차

1. 개요

1.1 문서 목적

1.1.1 본 문서는 팀 그리마의 “3DGS 기반 AR 시스템” LumiAR 의 상세설계서이다. 소수의 사용자 이미지로부터 COLMAP 기반의 카메라 포즈·희소 포인트를 추정하고, 이를 경량화한 3DGS 로 최적화하여 실시간 렌더링 가능한 스플랫 씬을 구성한다. 모바일 AR Foundation 환경에서 SH(L0~L2) 기반 IBL-lite 조명 추정을 주기적으로 반영해 현실 공간과 조화를 이루는 합성 결과를 제공하는 것이 목적이다.

1.2 설계 범위 및 목표

1.2.1 입력: 서로 다른 시점의 사용자 촬영/업로드 이미지 4+장(JPG/PNG).

1.2.2 재구성: COLMAP SfM 으로 초기 포즈/포인트 산출 후, 경량 3DGS(iteration cap, adaptive pruning, tile-visibility)로 스플랫 파라미터(위치, 공분산, 불투명도, SH 계수)를 최적화. COLMAP 은 공개 SfM/MVS 파이프라인으로 대규모 데이터셋에서 검증된 절차를 제공한다.

1.2.3 조명: 카메라 프레임으로부터 저해상도 환경맵을 누적하여 **구면조화(SH) 9 계수(L0~L2)**를 추정, Lambertian 근사에서 저차 SH 만으로도 유효하다는 고전 결과에 근거해 실시간 IBL-lite 로 적용.

1.2.4 AR 합성: Unity/AR Foundation 에서 평면 탐지·앵커·제스처 조작 및 'LightEstimation(ambient SH, main light)'을 병합 활용하여 scene 조도를 갱신.

1.2.5 출력: AR 합성 캡처(이미지/영상), **SplatPack**(.splat/.ply) 및 옵션으로 .glb 파일.

1.3 주요 기능 요약

1.3.1 **소수 프레임 3D 재구성(경량 3DGS)**

1.3.1.1 COLMAP SfM 으로 초기 카메라 파라미터/희소 포인트를 추정하고, 이를 초기값으로 *3D 가우시안(위치/공분산/opacity/SH)*을 최적화한다. 학습 중 visibility-aware 라스터를 사용해 역전파를 가속하고, iteration cap + adaptive pruning + tile-based visibility 로 시간을 단축한다. 본 설계는 과제 환경에 맞게 파이프라인을 경량화 할 수 있다.

1.3.2 **IBL-lite 실시간 조명 동기화**

1.3.2.1 카메라 프레임으로 64×32 수준의 저해상도 환경맵을 누적한 뒤 SH 9 계수를 최소자승으로 추정한다. Lambertian 근사에서 2 차(9 계수)만으로 주요 조명 성분을 포착할 수 있음을 근거로 하며, Unity 셰이더에서 노멀 기반 SH 방사 조도를 즉시 평가해 베이스

라이팅으로 사용한다. AR Foundation 의 ambient SH / main light 방향·세기도 함께 사용 가능하다.

1.3.3 **AR 배치/합성 및 사용자 상호작용**

1.3.3.1 AR Foundation 으로 평면 탐지·앵커 고정, 스플랫 프리팹(SplatRenderer) 배치, 스케일·회전 제스처 및 Shadow Matte 를 통한 바닥 접지 그림자를 제공한다. 라이트 추정 갱신과 스플랫 렌더링은 프레임 루프에서 저지연 갱신된다. (ARKit/ARCore Light 추정을 Unity 에서 일관 인터페이스로 제공)

1.3.4 **결과물 관리/공유**

1.3.4.1 스플랫 번들(SplatPack)을 헤더(JSON) + 바이너리 블롭 구조로 저장하여 재사용·전송을 용이하게 하고, 썸네일/메타데이터를 DB 에 인덱싱한다. 필요 시 .glb 변환, AR 합성 캡처(이미지/영상) 저장을 지원한다.

2. 시스템 설계

2.1 전체 아키텍처

2.1.1 계층 개요

2.1.1.1 **클라이언트**(모바일/태블릿, Unity/AR Foundation): 이미지 업로드, AR 평면 탐지·앵커, 스플랫 렌더링(SplatRenderer), 실시간 조명 적용(IBL-lite, SH9).

2.1.1.2 **재구성 서비스**(Python/CUDA): COLMAP SfM → 경량 3DGS 학습(visibility-aware, pruning, iteration cap) → SplatPack 산출.

2.1.1.3 **라이트 추정**(온디바이스 기본, 서버 옵션): 카메라 프레임 누적으로 저해상도 환경맵 생성 → SH(L0~L2, 9 계수) 추정 → 셰이더 상수 업데이트(≤200ms 목표).

2.1.1.4 **스토리지/메타데이터**: 결과물(SplatPack, 썸네일)과 프로젝트 메타를 저장·색인(포맷 호환성 확보: splat/ply/glb/obj).

2.2 모듈 상세 설계

2.2.1 **입력 모듈** (Image Intake & Precheck)

기능: 이미지 수집, 형식·해상도·용량 검사, 기초 전처리(정규화/감마/노이즈 경감).

입력/출력: JPG/PNG → 정규화 이미지 텐서 + 메타(JSON: 초점거리/EXIF/촬영순서).

오류 처리: 해상도 부족/중복 시점/흔들림 감지 시 재촬영 안내.

UI 연계: 업로드 진행도, 실패 사유 명시, 다음 단계 활성화 조건.

2.2.2 **3DGS 재구성 모듈** (SfM → GS Training → SplatPack)]

2.2.2.1 **SfM(경량 COLMAP)**:

ORB 특징·근접 프레임 매칭·RANSAC 으로 포즈/희소점 복원.

2.2.2.2 **GS 학습(경량화 스케줄)** :

Iteration cap: 15k~25k 내 학습 종료.

Visibility-aware raster: front-to-back 누적 투과율 T로 기여도 $vk = T\alpha k$ 산출. 실제 화면에 기여한 가우시안만 역전파(불필요 업데이트 차단)

Adaptive pruning: 평균 가시성/opacity/기울기 기준 하위 10~15% 주기 제거.

Tile-based visibility: 스크린을 16×16 타일로 분할, 타일에 겹치는 스플랫만 정렬/합성

2.2.2.3 **산출물**: SplatPack(header.json + binary blobs),

런타임은 SplatLoader → GPU Buffers → SplatRenderer 로 소비한다.

2.2.3 **Lighting & IBL 모듈** (SH9 Estimation)

2.2.3.1 **입력:** 카메라 프레임(저해상도 64×32 LatLong 누적), AR Foundation light estimation(메인 라이트 방향/강도).

2.2.3.2 **처리:** LDR 감마 보정 → SH(L0~L2, 9 계수×RGB) 최소자승 → 지수평활 → outlier clamp → ≤200ms 내 셰이더 상수 갱신.

2.2.3.3 **출력:** _SH[0..8](float3), _Exposure, _ShadowStrength 상수 버퍼.

2.2.4 **AR 렌더링 모듈** (Anchoring & SplatRenderer)

2.2.4.1 평면 탐지/앵커·좌표 동기화, 탭 배치/드래그/스케일/회전 제스처 제공.

2.2.4.2 **렌더링 파이프라인:** SplatLoader 가 StructuredBuffer 업로드 → URP 패스에서 screen space Rasterize + 깊이/알파 합성 → Shadow Matte 로 접지 그림자 표현

2.2.4.3 **성능 대처:** FPS 저하 시 LOD/프루닝 강도 상향, 타일 크기 조절(16↔32).

2.2.5 **출력/저장 모듈 (Artifacts & Project Index)**

2.2.5.1 **포맷:** SplatPack(.splat/.ply), 옵션 .glb 변환, AR 합성 캡처(이미지/영상).

2.2.5.2 **메타/색인:** 썸네일, 생성일, 가우시안 수, 품질지표(PSNR/SSIM 추정) 저장.

2.2.5.3 **호환성 설계:** .glb/.obj 등 범용 포맷 병행 지원.

2.2.6 구현 책임 구분(Build/Modify/Existing)

모듈	책임 구분	상세 범위
입력 모듈	**Build**	해상도/형식 검사, EXIF 추출, 블러/중복 시점 휴리스틱
SfM 래퍼	Existing	COLMAP CLI/py 바인딩 호출, 결과 파싱, 실패 재시도 로직
3DGS 학습 스케줄러	Modify(경량화)	iteration cap, visibility-aware raster, adaptive pruning, 타일 가시성
SplatPack 포맷 & 로더	**Build**	header(JSON) + blobs 정의, Unity 런타임 로더
Lighting/IBL-lite(SH9)	**Build**	저해상도 envmap 누적, SH(L0~L2) 최소자승, 지수평활/클램프
AR 렌더링 (Anchoring &SplatRenderer)	**Build**	AR Foundation 제스처, URP 커스텀 패스(스플랫 래스터/알파합성), Shadow Matte
서버 /API(FastAPI)	**Build**	업로드/재구성/상태/다운로드/로그

2.3 처리 흐름

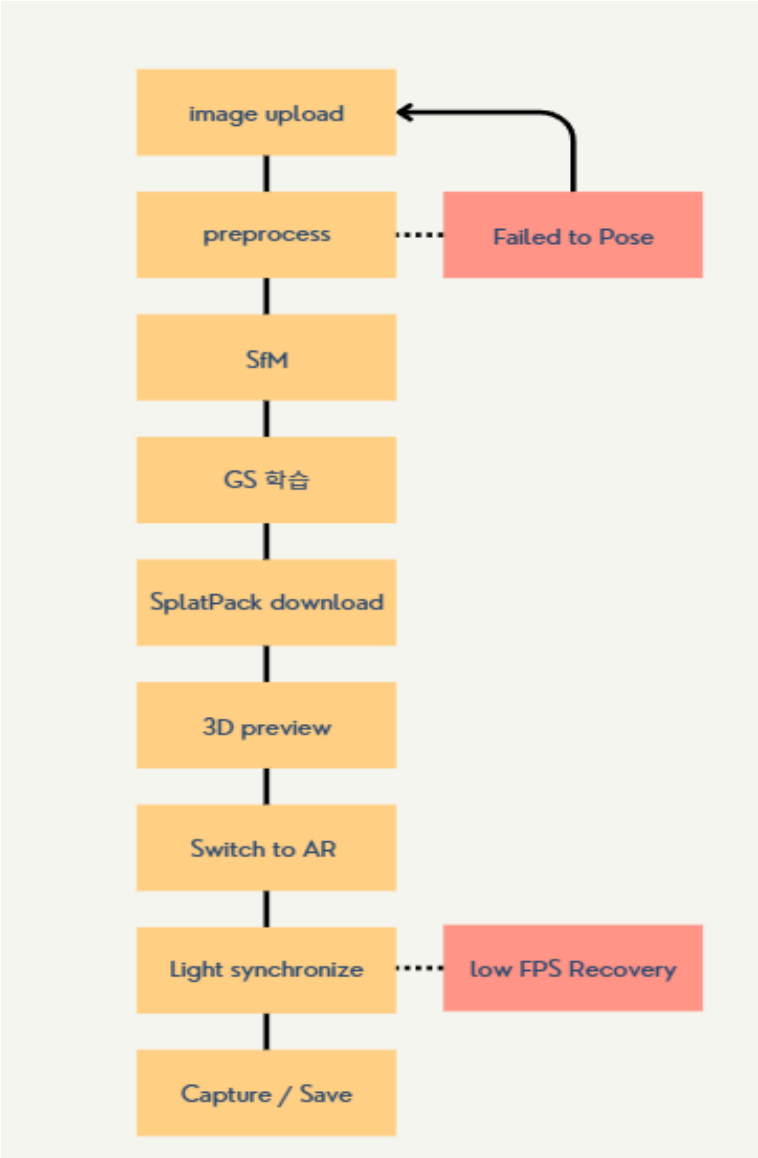


image upload
preprocess
Failed to Pose
SfM
GS 학습
SplatPack download
3D preview
Switch to AR
Light synchronize
low FPS Recovery
Capture / Save

3. 데이터 및 API 설계

3-1. 입력 데이터 구조

구분	항목명	타입	설명
ImageInput	image_id	String	업로드 이미지 고유 ID
	file_path	String	로컬 저장 경로 (임시)
	upload_time	Timestamp	업로드 시각
	camera_intrinsic	JSON	카메라 내부 파라미터 (focal length, cx, cy 등)
	device_info	String	촬영 디바이스 정보 (Android/iOS 모델명)

3-2. 2D → 3D 재구성 데이터 (Gaussian Splatting)

구분	항목명	타입	설명
Gaussian Point	position	Float[3]	(x, y, z) 위치
	color	Float[3]	RGB 색상
	opacity	Float	투명도 (α 값)
	scale	Float[3]	각 축 방향의 분산 (σx, σy, σz)
	rotation	Float[4]	쿼터니언 기반 회전값
	sh_coefficients	Float[16]	0~3차 Spherical Harmonics 계수
	normal	Float[3]	법선 벡터 (선택적, Lighting 보정용)

3-3. Lighting / SH 계수 데이터 구조

구분	항목명	타입	설명
LightingEstimation	sh_order	Int	SH 차수
	sh_coefficients	Float[9~16]	전역 광원 Spherical Harmonics 계수
	light_direction	Float[3]	주요 광원 방향 (디버깅용)
	intensity	Float	광원 세기 (0~1 정규화)
	timestamp	Timestamp	추정 시점 (Frame 기준)

3-4. 배치(Placement) 데이터 구조

구분	항목명	타입	설명
PlacementAnchor	anchor_id	String	Unity AR Anchor 고유 ID
	model_id	String	배치할 3D 모델 ID
	position	Float[3]	월드 좌표계 위치
	rotation	Float[4]	회전값 (Quaternion)
	scale	Float[3]	스케일
	light_ref	String	연결된 조명 환경 ID

3.2 DB 및 파일 저장 구조

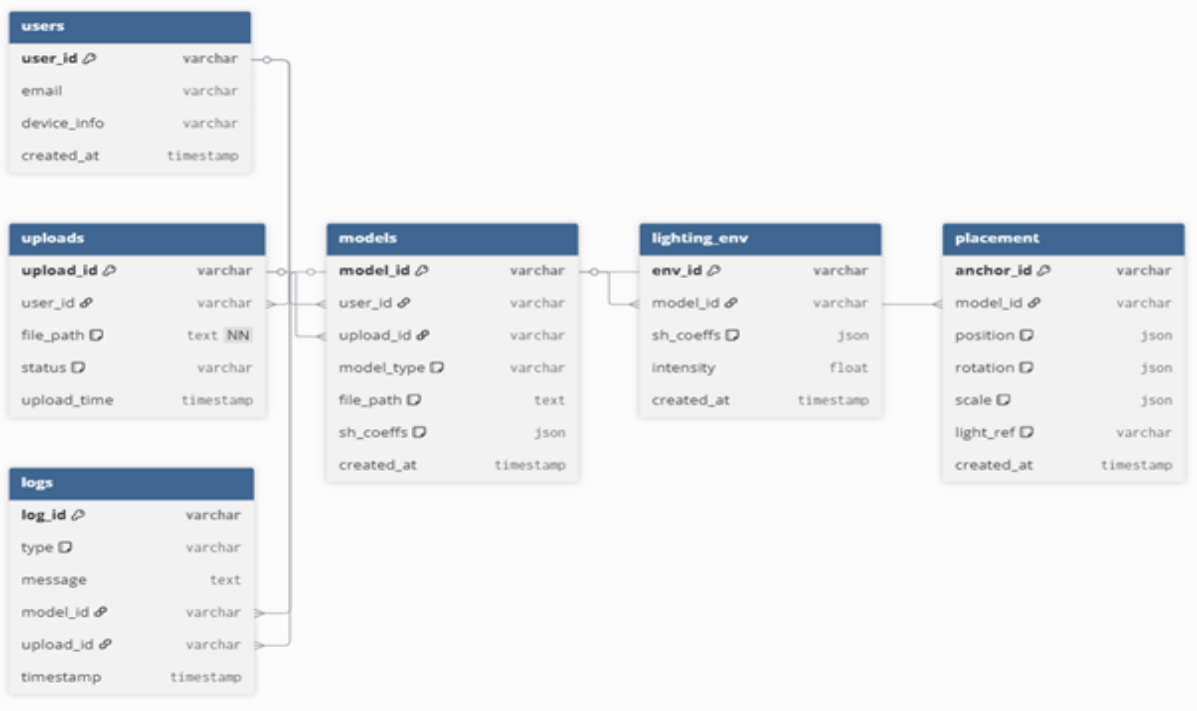
(1) Database 개요

- **DBMS:** PostgreSQL
- **사용 목적:** 메타데이터 관리, 모델 경로 및 사용자 작업 상태 관리

(2) 테이블 구조

테이블명	주요 필드	설명
users	user_id (PK), email, device_info, created_at	사용자 식별 및 환경 정보
uploads	upload_id (PK), user_id (FK), file_path, status, upload_time	입력 이미지 업로드 기록
models	model_id (PK), user_id (FK), model_type (3DGS/OBJ), file_path, sh_coeffs, created_at	생성된 3D 모델 메타데이터
lighting_env	env_id (PK), model_id (FK), sh_coeffs, intensity, created_at	Lighting 환경 추정 결과
placement	anchor_id (PK), model_id (FK), position, rotation, scale, light_ref	배치된 모델 정보
logs	log_id (PK), type, message, timestamp	오류 및 상태 로그

(3) ERD



(4) 파일 저장 구조

/LumiAR/

├── uploads/

│ ├── {user_id}/input_{timestamp}.jpg

├── models/

│ ├── {user_id}/model_{uuid}.ply (또는 .splat)

├── lighting/

│ ├── {model_id}/sh_{timestamp}.json

├── previews/

│ ├── {model_id}/thumbnail.png

3.3 주요 API 명세

[API-001] 이미지 업로드

- **Method:** POST
- **Endpoint:** /api/upload
- **설명:** 사용자가 촬영한 이미지를 업로드하고 서버로 전달
- **Request Body:**

```
{
  "user_id": "user_123",
  "image_file": "<binary>",
  "device_info": "Galaxy S23"
}
```

- **Response:**

```
{
  "upload_id": "up_001",
```

```
  "status": "accepted",
  "upload_time": "2025-10-13T10:22:30"
}
```

[API-002] 3DGS 모델 생성 요청

- **Method:** POST
- **Endpoint:** /api/reconstruct
- **설명:** 업로드된 이미지를 기반으로 3DGS 모델 생성 요청
- **Request:**

```
{
  "upload_id": "up_001",
  "method": "gaussian_splatting"
}
```

- **Response:**

```
{
  "model_id": "mdl_001",
  "status": "processing",
  "estimated_time": "300s"
}
```

[API-003] 모델 생성 상태 조회

- **Method:** GET
- **Endpoint:** /api/reconstruct/status/{model_id}
- **Response:**

{

"model_id": "mdl_001",

"status": "completed",

"model_path": "s3://LumiAR/models/user123/model_mdl_001.splat"

}

[API-004] Lighting 추정 요청

- **Method:** POST
- **Endpoint:** /api/lighting/estimate
- **설명:** 카메라 프레임을 이용해 조명 SH 계수를 추정
- **Request:**

  {

  "model_id": "mdl_001",

  "frame_data": "<binary>",

  "timestamp": "2025-10-13T11:00:00"

  }
- **Response:**

  {

  "env_id": "env_101",

  "sh_coefficients": [0.42, -0.13, 0.08, ...],

  "intensity": 0.92

  }

[API-005] 모델 배치 요청

- **Method:** POST
- **Endpoint:** /api/placement
- **설명:** Unity에서 AR Anchor 정보를 이용하여 모델을 배치
- **Request:**
- {

  "model_id": "mdl_001",

  "anchor": {

  "position": [0.2, 0.0, -1.1],

  "rotation": [0, 0, 0, 1],

  "scale": [1, 1, 1]

  },

  "lighting_env": "env_101"

  }

- **Response:**

  {

  "anchor_id": "anc_001",

  "status": "placed"

  }

[API-006] 모델 미리보기 및 삭제

- **GET** /api/model/{model_id}/preview
  → 모델의 썸네일 및 메타데이터 조회
- **DELETE** /api/model/{model_id}
  → 모델 및 관련 파일 삭제 요청

4. 사용자 인터페이스 설계

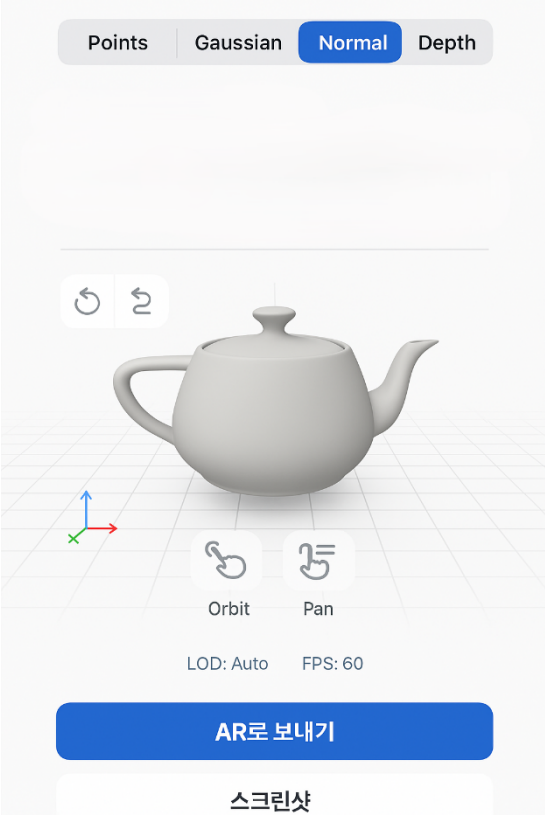
4.1. 홈

와이어 프레임	기능 상세
	앱 실행 직후, 사용자는 **새 프로젝트 시작(이미지 업로드)** 또는 **최근 프로젝트 이어하기** 중 하나를 선택할 수 있다. **1. 앱 소개 표시** ● 홈 화면 상단 중앙에 앱 로고와 앱 소개 문구를 표시한다. **2. 시작하기(이미지 업로드) 버튼** ● 홈 화면 중앙에 "시작하기" 버튼을 배치한다. ● 사용자가 버튼을 클릭하면 기기 내 파일 탐색기가 실행된다. ● 허용 파일 형식은 .jpg, .png이며, 선택된 이미지 파일을 불러와 새 프로젝트를 생성한다. **3. 최근 프로젝트 이어하기 버튼** ● "최근 프로젝트 이어하기" 버튼을 클릭하면, 기기 내에 저장된 프로젝트 목록 창이 표시된다. ● 목록은 프로젝트명, 마지막 수정일 등을 포함하며, 사용자가 항목을 선택하면 해당 프로젝트를 불러온다. **4. 로딩 상태 표시** ● "시작하기" 또는 "최근 프로젝트 이어하기" 기능 실행 시, 회전 아이콘과 "불러오는 중..." 텍스트를 화면 중앙에 표시한다.

4.2. 이미지 업로드

와이어 프레임	기능 상세
	사용자는 3D 재구성을 위해 서로 다른 각도에서 촬영된 4장의 이미지를 업로드해야 한다. 입력된 이미지는 **형식 및 용량 기준을 검증**하며, 모든 검증이 완료되면 다음 단계로 진행할 수 있다. **1. 이미지 촬영 및 업로드** ● 화면 중앙에 "이미지 추가" 버튼을 배치한다. ● 사용자가 버튼을 클릭하면 선택 창이 표시되며, 카메라 촬영 또는 기기 파일 탐색기에서 업로드 중 하나를 선택할 수 있다. ● 선택된 이미지는 업로드 슬롯에 미리보기 형태로 표시된다. **2. 이미지 권장 각도 안내** ● 총 4개의 이미지 업로드 슬롯을 제공하며, 각 슬롯에는 권장 촬영 각도(정면, 좌측, 우측, 상단 등)를 시각적으로 안내하는 아이콘 또는 텍스트를 표시한다. **3. 용량 및 형식 체크** ● 이미지 형식 및 파일 용량을 확인한다. **4. 진행도 표시** ● 화면 하단에 진행도 바(progress bar)를 배치하여 업로드 진행도(0~100%)를 시각적으로 표시한다.

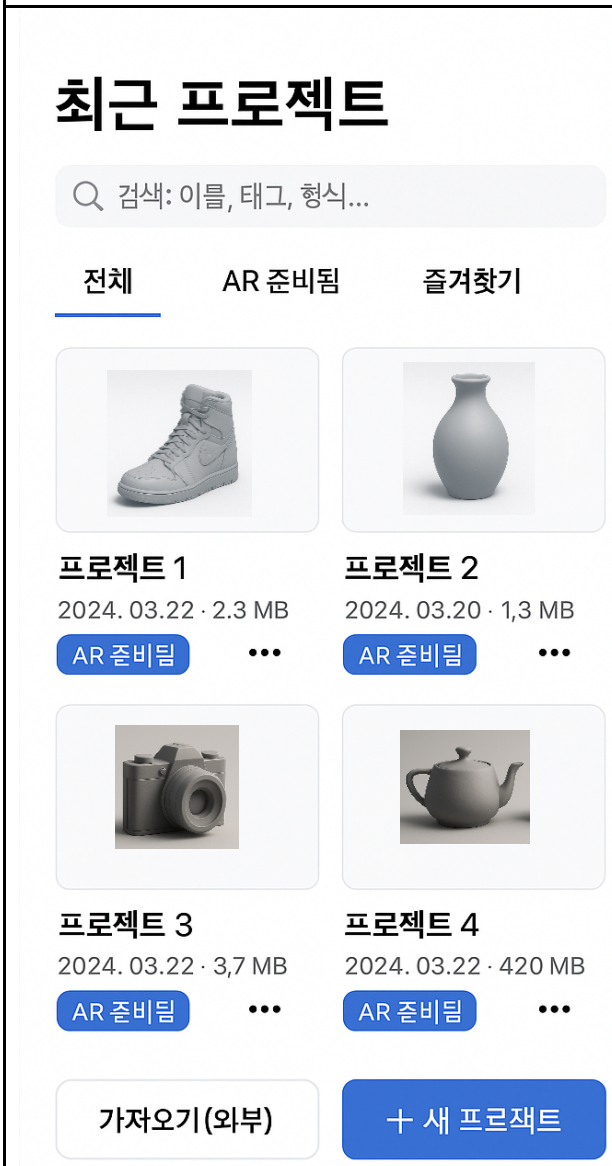
4.3. 3D 미리보기

와이어 프레임	기능 상세
	사용자는 3D 재구성된 결과물을 **모바일 화면에서 실시간으로 확인**할 수 있으며, **카메라 조작을 통해 다양한 각도에서 모델을 관찰**할 수 있다. 또한 모델의 **품질 정보를 시각적으로 확인**할 수 있고, 결과물을 **AR 환경으로 전송하거나 스크린샷으로 저장**할 수 있다. **1. 모델 로드 및 렌더링** ● 3D 재구성 과정이 완료되면 자동으로 3D 미리보기 화면으로 전환된다. ● 전환 시, 로딩 인디케이터가 표시되며 모델 데이터가 로드 완료되면 실시간 렌더링이 시작된다. **2. 카메라 조작** ● 드래그(Drag): 모델 회전 ● 핀치(Pinch): 확대/축소 (Zoom In/Out) ● 두 손가락 이동: 화면 이동 **3. 모델 정보 및 품질 표시** ● 화면 우측 상단 또는 별도의 정보 패널에 모델 품질 관련 데이터를 표시한다. **4. 스크린샷 및 AR 전송 기능** ● 스크린샷 버튼 클릭 시: 현재 시점에서 렌더링된 이미지를 캡처하여 기기 앨범에 저장한다.

4.4. AR 배치

와이어 프레임	기능 상세
	사용자는 **3D 모델을 AR 환경에 배치**할 수 있다. 모델은 탐지된 평면 위에 고정되며, 사용자는 **제스처를 통해 모델의 위치, 크기, 회전**을 조정할 수 있다. 또한 **실제 환경의 조명과 그림자 정보가 모델에 실시간 반영**되어야 하며, 사용자는 **화면 캡처 및 조명 설정 기능**을 통해 결과를 조정하고 저장할 수 있다. **1. 평면 탐지 및 Anchor 고정** ● AR 모듈 실행 시, AR SDK를 이용하여 바닥, 테이블 등 수평 평면을 자동 탐지한다. ● 사용자가 모델을 배치하면, 해당 지점에 Anchor가 생성되고 모델의 월드 좌표계와 AR 좌표계가 동기화된다. **2. 배치 및 제스처 조작** ● 드래그(Drag) 제스처로 모델의 위치를 평면 위에서 이동할 수 있다. ● 핀치(Pinch) 제스처로 모델의 크기(스케일)를 확대 또는 축소할 수 있다. ● 두 손가락 회전(Rotate) 제스처로 모델의 회전(Orientation)을 변경할 수 있다. **3. 실시간 조명 반영** ● 카메라와 센서를 통해 측정된 환경광 및 방향광 정보를 기반으로 모델의 조명 상태를 동적으로 조정한다. ● 모델의 재질은 PBR기반으로, 밝기, 색온도, 그림자 방향 등이 실시간으로 반영된다.

4.5. 최근 프로젝트

와이어 프레임	기능 상세
	사용자는 **저장된 3D 프로젝트 결과물**을 **리스트 형태로 확인**할 수 있으며, **검색·정렬·필터 기능**을 통해 원하는 프로젝트를 빠르게 찾을 수 있다. 각 프로젝트는 **썸네일 형태의 미리보기 이미지**로 시각적으로 식별 가능하며, 프로젝트를 선택하면 해당 데이터를 불러와 **재사용 또는 편집**할 수 있다. **1. 저장된 프로젝트 리스트 표시** ● 화면에는 기기 내 저장된 프로젝트가 카드 형태로 나열된다. ● 각 카드에는 프로젝트명, 생성일, 마지막 수정일, 미리보기가 표시된다. **2. 검색 / 정렬 / 필터 기능** ● 화면 상단에 검색 창을 배치하여, 프로젝트 검색이 가능하다. ● 정렬 옵션 버튼을 통해 생성일, 수정일, 이름 순으로 리스트를 재정렬할 수 있다. ● 필터 기능을 제공하여 AR 준비됨, 즐겨찾기, 최근 수정 조건으로 프로젝트를 분류해 표시한다. **3. 미리보기 및 가져오기 기능 제공** ● 사용자가 특정 프로젝트 카드를 클릭하면 미리보기 창이 열린다. ● 미리보기 창에는 해당 프로젝트의 3D 썸네일 이미지와 기본 정보가 표시된다. ● 불러오기 버튼을 클릭하면 해당 프로젝트가 로드되어 3D 미리보기 화면으로 전환된다.

5. 비기능 설계

5.1. 렌더링 FPS·해상도·예산

5.1.1.목표 FPS(모바일, URP): ≥ 30 FPS

기본 해상도 1080p, 성능 저하 시 720p 로 자동 다운스케일.

Gaussian 수 목표/상한: 장치 프로파일별 100k·200k·300k 단계(LOD)타일 크기: 16×16(기본) / 32×32(저성능 폴백).

URP 최적화 기본값: 추가광 그림자 비활성, 그림자 거리/캐스케이드 축소, 카메라 수 최소화.

5.2. 지연 (Latency)

5.2.1. **재구성**(4 장 입력, 데스크톱 GPU 기준): ≤ 5 분

COLMAP SfM(특징 추출/매칭/포즈): 1.5 분 이내.

경량 3DGS 학습(Iter cap 15k–25k, pruning/타일 가시성): 3.5 분

5.2.2. 조명 반영(IBL-lite/SH): ≤ 200 ms 내로 SH(L0~L2, 9 계수) 재추정 및 셰이더 상수 갱신(5–10 Hz).

5.3. 안정성 (런타임)

5.3.1. **목표:** 30 분 연속 시연 도중 크래시 발생 X

5.3.2. **메모리/VRAM 예산(모바일):** 200k splats 기준 VRAM ~120MB 전후(Half 정밀도 사용 시 감소), 재구성 결과 메타·썸네일 포함 앱 메모리 < 1.2GB

5.3.3. 폴백 시나리오:

FPS 하락 → LOD 하향(가우시안 수/해상도/타일 크기 조정), 섀도우·HDR·추가광 단계적 비활성. (URP 권장사항 준용)

조명 급변 플리커 → SH 지수평활 가중↑, 이상치 clamp, 메인 라이트만 임시 사용. (AR 라이트 에스티메이션 제공 범위)

5.4. 성능 지표 표준

항목	목표	허용치	측정 방법
렌더링 FPS(모바일)	≥ 30	≥ 24	Unity Profiler, GPU FrameTime
재구성 Latency	≤ 5분	≤ 7분	서버 로그(phase 타임스탬프)
SH 갱신 지연	≤ 200 ms	≤ 300 ms	카메라프레임→셰이더상수 반영 시간
VRAM(모바일, 200k splats)	≤ 120MB	≤ 180MB	RenderDoc/Profiler
앱 메모리	≤ 1.2GB	≤ 1.5GB	OS/Profiler
시연 안정성	30분 무크래시	1회 이하 경미 오류	크래시/ANR 로그

6. 개발 환경 및 일정

6.1. 사용 기술 스택

영역	기술/도구	버전	목적/비고
엔진/AR	Unity, URP, AR Foundation(ARCore/ARKit)	Unity: 2022.3.62f2 URP: 14.0.12 ARF: 5.1.6	모바일 실시간 렌더링 + 크로스 AR SDK 일원화
재구성(SfM)	COLMAP(CUDA 빌드)	3.12.6	포즈/희소점 복원 (ORB/RANSAC)
3DGS 학습	Python, PyTorch, NumPy/OpenCV	Python: 3.12.12 PyTorch: 2.5.1 (CUDA 12.1 빌드)	경량 학습(visibility-aware, pruning)
라이트 추정	SH(L0~L2) 피팅 모듈 (NumPy/SciPy), AR LightEstimation	내부모듈	IBL-lite(9계수) + 플랫폼 라이트 추정 병합
서버/API	FastAPI 또는 Flask, Uvicorn/Gunicorn, NGINX	LTS	/recon/*업로드/상태/다운로드
스토리지	S3 호환 스토리지, PostgreSQL/SQLite(메타)	managed/로컬	SplatPack/썸네일/메타 아카이빙
CI/CD	GitHub Actions, pytest/unittest, black/isort/mypy	-	품질/빌드 자동화
프로파일링	Unity Profiler, RenderDoc, nvprof/Nsight	-	FPS/VRAM/프레임 타임 추적

6.2. 개발 환경 구성

6.2.1. 하드웨어 / OS 기준

6.2.1.1. 개발 PC: Windows 11 / Ubuntu 22.04, NVIDIA GeForce RTX 3070

6.2.1.2. 모바일 단말: Android(ARCore 지원) 1기 이상

6.2.1.3. 저장소: 산출물(SplatPack) 보관용 로컬 SSD + S3 버킷

6.3. 주차별 개발 일정

주차	핵심 작업	세부 내용	산출물 / QG
6W	SRS/설계, 환경 셋업	Unity 2022.3 LTS/URP/AR Foundation 프로젝트 생성, Python/Colmap/CUDA 셋업, 리포 구조 확정	프로젝트 스캐폴딩, 실행 체크리스트
7W	입력/전처리 & COLMAP 래퍼	업로드/전처리(해상도/감마/메타), ORB/RANSAC 파이프라인 스크립트화	/recon/upload베타, SfM 결과 시연
8W	3DGS 학습(경량 스케줄)	iteration cap/visibility-aware/프루닝/타일 가시성 1차 구현	SplatPack v0(헤더/블롭), QG1: 1장면 수렴
9W	Unity SplatLoader/Renderer	StructuredBuffer 업로드, 기본 Raster/알파 합성, 3D 프리뷰 화면	3D 프리뷰 시연, FPS 로그
10W	IBL-lite(SH9) 온디바이스	카메라 프레임 누적→SH9 추정→셰이더 상수 업데이트(≤200ms)	라이트 반영 데모, QG2: 플리커< 허용

11W	AR 배치/제스처/앵커	평면 탐지, 배치/스케일/회전, Shadow Matte 합성	AR 합성 시연 (30FPS 목표)
12W	품질/성능 튜닝 라운드	LOD/프루닝/타일 크기 자동화, VRAM/프레임타임 측정	프로파일(저/중/고), QG3: 성능표
13W	서버 안정화/다운로드	/recon/status, /recon/download안정화, 청크/재시작	대용량 산출 다운로드, 실패 복구 시나리오
14W	저장/최근 프로젝트/썸네일	S3 업로드, 메타/썸네일/색인, 최근 프로젝트 화면	저장/가져오기 UX, QG4: 안정성 30분
15W	통합 테스트/버그픽스	회귀 테스트(샘플 씬 세트), 에러 메시지/가이드 보강	Test Report, 사용자 가이드 초안
16W	최종 발표	최종 시연, 발표자료 완성	최종 빌드/영상