'LumiAR' 최종 보고서

컴퓨터공학 종합설계[202502-CSE4205-001]

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

목차

1. 개요

1.1 문서 목적

본 문서는 팀 그리마의 "3DGS 기반 AR 시스템" LumiAR의 최종 보고서이다.

1.2 개발 배경 및 필요성

최근 Apple Vision Pro, Meta Quest 3 등 XR 기기의 보급으로 고품질 3D 콘텐츠에 대한 수요가 급증하고 있다. 그러나 기존의 3D 에셋 제작 방식인 Photogrammetry는 연산 시간이 오래 걸리고, NeRF(Neural Radiance Fields)는 렌더링 속도가 모바일 환경에 적합하지 않은 한계가 있다.

2023년 발표된 3D Gaussian Splatting(3DGS은 실시간 렌더링과 고품질 복원을 동시에 달성했으나, 원본 알고리즘은 배경(Background)과 객체(Object)를 분리하지 못해 AR 환경에 배치 시 불필요한 배경 노이즈가 포함되는 치명적인 단점이 존재한다.

1.3 프로젝트 목표

본 프로젝트 LumiAR 은 사용자가 스마트폰으로 촬영한 2D 동영상을 입력하면, 배경이 완벽하게 제거된 고품질 3D 객체(Point Cloud/Mesh)로 자동 변환하여 AR 환경(Unity)에 즉시 배치할 수 있는 **End-to-End 자동화 파이프라인**을 구축하는 것을 목표로 한다.

2. 시스템 설계

2.1 전체 시스템 구조

본 시스템은 **Client-Server 구조**로 설계되었으며, 고사양 연산이 필요한 3DGS 학습은 Local GPU Server에서 수행하고, 결과물은 모바일 클라이언트에서 소비하는 형태이다.

- **Client (Mobile):** 동영상 촬영 및 업로드, 생성된 3D 에셋 목록 조회 및 AR 배치 (Unity ARFoundation).
- **Server (Backend):** FastAPI 기반의 REST API 서버. 작업 대기열(Queue) 관리 및 파이프라인 오케스트레이션 수행.
- **Core Pipeline:** FFmpeg, COLMAP, Python(PyTorch) 기반의 3DGS 학습 및 후처리 모듈.

2.2 기술 스택

구분	상세 기술	비고
OS	Windows 11	PowerShell CLI 환경 최적화
Backend	Python 3.9+, FastAPI, SQLite	비동기 작업 처리 및 상태 관리
Network	Ngrok	로컬 서버 외부 접속 터널링
3D Vision	COLMAP, 3D Gaussian Splatting	SfM 및 3D Scene Reconstruction
AI/ML	Rembg (U-2-Net), PyTorch	객체 마스킹 및 학습
Engine	Unity (ARFoundation)	Aras Gaussian Splatting Renderer 활용

3. 핵심 구현 내용

3.1 Core Pipeline

LumiAR 파이프라인은 총 7단계의 공정으로 이루어지며, 각 단계는 기존 오픈소스의 한계를 극복하기 위해 독자적인 알고리즘으로 개선되었다. 주요 기술을 위주로 서술한다.

3.1.1 Stage 1~2: 영상 전처리 및 최적화 SfM

▷ **Sequential Matching 도입:** 일반적인 이미지 세트와 달리, 동영상은 프레임 간 연속성이 존재한다. 이를 활용해 Sequential Matcher를 적용, Feature Matching 속도를 3배 이상 향상시키고 False Matching을 최소화했다.

▷ **Adaptive Frame Extraction:** FFmpeg를 제어하여 영상의 길이에 따라 FPS를 자동 조절, 최적의 데이터셋(100~300장 내외)을 구축하도록 설계했다.

3.1.2 Stage 3: Hybrid Mask Cleaning

Rembg(AI 마스킹)가 생성한 초기 마스크는 노이즈가 많아 3D 품질을 저하시킨다. 이를 해결하기 위해 **3단계 하이브리드 필터링**을 구현했다.

▷ **Robust Z-score Filter:** 전체 프레임 중 객체 크기 비율이 통계적으로 비정상적인 (너무 크거나 작은) 프레임을 자동 제거.

▷ **Dust Filter:** 렌즈 먼지나 작은 부유물을 제거하기 위해 절대 픽셀 면적 기반의 필터링 수행.

▷ **Center-Weighted Repair:** 중앙 집중 가중치(Gaussian Weight)를 적용하여, 화면 중앙에 위치한 주 객체는 살리고 주변 배경 노이즈는 침식(Erosion) 연산으로 제거한 뒤 다시 복원(Dilation)하는 알고리즘 적용.

3.1.3 Stage 3.5: Voting-based Visual Hull

3DGS의 고질적인 문제인 "Sparse Point Cloud(점군 부족)로 인한 학습 실패"를 방지하기 위해, 오브젝트 내부의 초기 포인트 클라우드를 인위적으로 증강하는 **Visual Hull** 기법을 도입했다.

▷ **Consensus Voting:** 3D 공간의 격자(Voxel) 점들을 카메라에 투영(Projection)했을

때, 유효한 뷰의 70% 이상이 마스크 내부라고 판단하는 경우에만 점을 생성한다.

▷ **효과:** COLMAP이 특징점을 찾지 못한 텍스처가 없는 영역(예: 흰색 도자기 등)도 형상을 성공적으로 복원함.

3.1.4 Stage 4: Mask-Supervised 3DGS Training

▷ **Loss Function 개선:** 기존 RGB Loss 외에 **Mask Loss**를 추가하여, 배경 영역의 Gaussian 투명도(Alpha)를 0으로 수렴시키도록 학습 루프를 수정했다.

▷ **White Background Adaptation:** 투명 배경 처리를 위해 렌더링 시 배경색을 동적으로 제어하며 학습 안정성을 확보했다.

3.5. Stage 5: Iterative Inpainting & Normalization

3DGS 결과물 바닥면의 구멍(Hole)을 메우고, Unity 엔진 좌표계에 맞추는 최종 후처리 단계이다.

▷ **Iterative Inpainting:** 구멍 난 영역을 탐지하여 주변 텍스처 색상으로 2단계(Edge -> Deep)에 걸쳐 점진적으로 채워넣는 알고리즘을 구현했다.

▷ **Radial Crop & Normalization:** 객체 중심으로부터의 거리 통계를 기반으로 Outlier를 잘라내고(Percentile Crop), Unity의 Unit Scale(1.0)에 맞춰 크기와 좌표를 정규화했다.

3.2 문제 해결 및 성능 최적화

3.2.1 CUDA OOM (Out Of Memory) 문제

▶ 현상: 고해상도 이미지 처리 시 VRAM 부족으로 파이프라인 중단.

▶ 해결: try-except 블록을 통해 OOM 발생 감지 시 자동으로 CPU Fallback 모드로 전환하거나 배치 사이즈를 동적으로 조절하는 로직 구현.

torch.cuda.empty_cache() 및 명시적 GC(Garbage Collection) 호출로 메모리 누수 방지.

3.2.2. 배경 노이즈(Floater) 제거

▶ 현상: 학습 후에도 공중의 미세한 점(Floater) Splat들이 AR 몰입감을 저해.

▶ 해결: Statistical Outlier Removal (SOR) 필터를 3DGS 포맷(PLY)에 맞게 포팅하여 적용. 특히, stage7a_cleanup 스크립트에서 DBSCAN 군집화 알고리즘을 사용하여 메인 객체와 떨어진 군집을 과감하게 삭제.

3.2.3. Unity 연동 호환성

▶ 현상: Python에서 생성한 PLY 파일이 Unity의 Aras Importer에서 색상이 이상하거나 회전되어 로드됨.

▶ 해결: SH(Spherical Harmonics) 계수의 순서 및 차수(Degree 2 vs 3)를 Unity 포맷에 맞춰 재정렬.

좌표계 변환(Y-up vs Z-up) 행렬을 적용하여 내보내기 단계에서 자동 보정.

4. 기대효과 및 활용 방안

4.1. 기술적 기대 효과

▶ 자동화: 전문가의 수작업(Cleanup) 없이도 평균 15분 이내(RTX 3070기준)에 AR용 3D 에셋 생성 가능.

▶ 일관성: 영상 흔들림이나 일부 마스킹 오류가 있어도 Voting 기반 초기화와 후처리 필터를 통해 일관된 품질 보장.

4.2. 활용 분야

▶ **E-Commerce:** 상품 360도 뷰어 및 AR 피팅 서비스.

▶ **Digital Archiving:** 박물관 유물 등의 개인화된 디지털 소장.

▶ **UGC Gaming:** 사용자가 직접 스캔한 물건을 게임 아이템으로 활용.

5. 결론

5.1. 결론

LumiAR 프로젝트는 3D Gaussian Splatting 기술을 실제 서비스 레벨로 끌어올리기 위한 파이프라인 최적화에 집중했다. 특히, 영상 데이터의 특성을 고려한 SfM 최적화, 하이브리드 마스크 정제, 그리고 Visual Hull 기반의 초기화 기법은 기존 오픈소스 프로젝트 대비 월등한 객체 분리 성능을 보여주었다.

본 시스템은 차세대 3D 콘텐츠 제작의 진입 장벽을 획기적으로 낮추는 데 기여할 것으로 기대된다.

5.2. 별첨

시연 영상: https://www.youtube.com/watch?v=4QHBN2c--FQ

Unity code github: https://github.com/hyeonnmin/3DGS-AR-project

참고 논문: https://arxiv.org/pdf/2308.04079