VLM을 활용한 객체 인식 및 Grasping을 위한 알고리즘 보고서

1. 서론

본 보고서는 자연어로 기술된 객체(예: "red cups")를 실시간으로 인식하고, manipulator arm을 통해 grasping하는 시스템을 개발하기 위한 최적의 알고리즘을 선정하는 것을 목표로 합니다. 사용자는 Jetson AGX Orin 64GB, RGBD 카메라, ROS2 Humble 환경을 사용하며, 실시간 성능(5 FPS 이상)과 zero-shot learning을 통해 학습되지 않은 객체를 인식할 수 있는 알고리즘을 요구했습니다. 또한, MoveIt2를 활용한 모션 플래닝을 위해 객체의 정확한 위치 정보(3D 위치, 6D pose, 또는 3D bounding box)가 필요합니다. 본 보고서는 사용자가 제공한 알고리즘 목록(SAM2, ZegCLIP, SCLIP, CLIP-SEG, NanoOWL, Semantic-SAM, Semantic-Segment-Anything, EVLA, CLIP-RT, NanoSAM, grounding dino, grounded-sam-2, openvla, MobileSAMv2, ESAM)과 추가 조사를 바탕으로 최적의 알고리즘을 추천합니다.

2. 요구사항 분석

사용자의 주요 요구사항은 다음과 같습니다:

- **실시간 성능**: 최소 5 FPS 이상.
- **Zero-shot learning**: 학습되지 않은 객체를 자연어 프롬프트로 인식.
- **Instance segmentation**: 다중 객체를 구분하여 grasping에 필요한 개별 객체의 위치 정보 제공.
- **하드웨어 및 소프트웨어**: Jetson AGX Orin 64GB, RGBD 카메라, ROS2 Humble.
- **출력 형식**: 3D 위치, 6D pose, 또는 3D bounding box.
- **오픈소스 및 사용성**: GitHub에서 제공되는 패키지를 설치 및 실행 가능.
- **ROS2 통합**: 가능하면 ROS2와의 통합이 용이해야 함.

이러한 요구사항을 충족하기 위해, 각 알고리즘의 기능, 성능, Jetson 호환성, ROS2 통합 가능성을 조사했습니다.

3. 알고리즘 조사

3.1 NanoOWL + NanoSAM

- **설명**: NanoOWL은 OWL-ViT를 기반으로 한 open-vocabulary object detection 모델로, 언어 프롬프트를 통해 객체를 탐지합니다. NanoSAM은 SAM의 경량화 버전으로, bounding box 또는 point 프롬프트를 사용하여 instance segmentation을 수행합니다. 두 모델은 TensorRT를 통해 Jetson에 최적화되었습니다.
- **입력**: RGB 이미지, 텍스트 프롬프트.
- **출력**: NanoOWL은 2D bounding box를 제공하며, NanoSAM은 instance segmentation mask를 생성합니다. RGBD 카메라의 depth 정보를 결합하여 3D 위치 또는 bounding box를 추정할 수 있습니다.
- **Zero-shot 성능**: NanoOWL은 CLIP 기반으로 zero-shot object detection을 지원하며, LVIS 데이터셋에서 31.2 AP_r을 달성했습니다.
- **실시간 성능**: Jetson AGX Orin에서 NanoOWL은 ViT-B/32 모델로 95 FPS, NanoSAM은 PPHGV2-B1 모델로 약 104 FPS를 달성합니다(NanoSAM Hugging Face).
- **오픈소스**: GitHub에서 제공되며, 설치 및 실행 예제가 포함되어 있습니다.
- **ROS2 통합**: NanoOWL은 ROS2-NanoOWL node를 제공하며, NanoSAM은 Python API를 통해 사용자 정의 node로 통합 가능합니다.

- **장점**: Jetson에 최적화된 실시간 성능, ROS2 통합 용이, instance segmentation 제공.
- **단점**: 두 모델을 조합해야 하며, 설정이 약간 복잡할 수 있음.

3.2 Grounded-SAM

- **설명**: Grounded-SAM은 Grounding DINO로 open-set object detection을 수행하고, SAM으로 instance segmentation을 수행합니다. Grounding DINO는 텍스트 프롬프트를 통해 임의의 객체를 탐지합니다.
- **입력**: RGB 이미지, 텍스트 프롬프트.
- **출력**: 2D bounding box 및 instance segmentation mask. RGBD 카메라로 3D 위치 또는 bounding box 추정 가능.
- **Zero-shot 성능**: Grounding DINO는 COCO zero-shot에서 52.5 AP를 달성했습니다(Grounding DINO arXiv).
- **실시간 성능**: Jetson에서의 정확한 FPS는 확인되지 않았으나, NVIDIA NGC 및 Jetson Platform Services에서 지원되며, TensorRT 최적화를 통해 실시간 성능 달성이 가능할 것으로 보입니다(NVIDIA NGC).
- **오픈소스**: GitHub에서 제공되며, Hugging Face에서도 지원됩니다.
- **ROS2 통합**: 공식 ROS2 node는 없으나, Python API를 통해 통합 가능.
- **장점**: 높은 정확도, 널리 사용됨, Jetson 지원.
- **단점**: 실시간 성능을 위해 추가 최적화 필요, ROS2 통합에 노력 요구.

3.3 Semantic-SAM

- **설명**: Semantic-SAM은 SA-1B 데이터셋으로 훈련된 universal image segmentation 모델로, 언어 프롬프트를 통해 다양한 granularity에서 instance segmentation 및 semantic labeling을 수행합니다.
- **입력**: RGB 이미지, 텍스트 또는 클릭 프롬프트.
- **출력**: instance segmentation mask 및 semantic labels. RGBD 카메라로 3D 정보 추정 가능.
- **Zero-shot 성능**: SA-1B 및 semantic 데이터셋으로 훈련되어 강력한 zero-shot 성능을 제공합니다.
- **실시간 성능**: Jetson에서의 성능은 확인되지 않았으나, SwinT backbone(28M parameters)을 사용하여 최적화 가능성이 있습니다.
- **오픈소스**: GitHub에서 제공되며, 데모 및 훈련 코드 포함.
- **ROS2 통합**: 공식 ROS2 node는 없으나, Python API로 통합 가능.
- **장점**: multi-granularity segmentation, semantic labeling 제공.
- **단점**: Jetson에서의 실시간 성능 미확인, 최적화 필요.

3.4 CLIP-SEG

- **설명**: CLIP-SEG은 CLIP에 minimal decoder를 추가하여 텍스트 프롬프트로 segmentation mask를 생성합니다. zero-shot 및 one-shot segmentation을 지원합니다.
- **입력**: RGB 이미지, 텍스트 프롬프트.
- **출력**: binary segmentation mask. instance segmentation은 지원하지 않음.
- **Zero-shot 성능**: Pascal-VOC에서 mIoU_S 35.7, mIoU_U 43.1을 달성했습니다(CLIP-SEG arXiv).
- **실시간 성능**: Jetson에서의 정확한 FPS는 미확인이나, CLIP 기반 모델이 Jetson에서 실행 가능하므로 최적화 시 실시간 성능 가능성 있음.
- **오픈소스**: GitHub 및 Hugging Face에서 제공.
- **ROS2 통합**: 공식 ROS2 node 없음, Python API로 통합 가능.
- **장점**: 간단한 사용, 직접적인 텍스트 기반 segmentation.
- **단점**: instance segmentation 미지원, 다중 객체 구분 어려움.

3.5 MobileSAMv2

- **설명**: MobileSAMv2는 SAM의 경량화 버전으로, SegEvery를 효율적으로 수행합니다. object-aware prompt sampling을 통해 mask 생성 속도를 향상시켰습니다.
- **입력**: RGB 이미지.
- **출력**: 모든 객체에 대한 instance segmentation mask. 언어 기반 필터링 필요.
- **Zero-shot 성능**: LVIS 데이터셋에서 mask AR@K 기준 42.5%를 달성했습니다(MobileSAMv2 arXiv).
- **실시간 성능**: Mac i5 CPU에서 약 3초 소요, Jetson에서 최적화 시 실시간 성능 가능성 있음.
- **오픈소스**: GitHub에서 제공.
- **ROS2 통합**: 공식 ROS2 node 없음, Python API로 통합 가능.
- **장점**: 경량화, 모든 객체 segmentation 가능.
- **단점**: 언어 기반 객체 선택을 위해 추가 처리 필요.

3.6 기타 알고리즘

- **SAM2**: SAM2는 이미지 및 비디오 segmentation을 위한 모델로, 44 FPS를 달성하나, Jetson AGX Orin에서 2 FPS로 성능이 낮습니다. 언어 프롬프트 미지원.
- **ZegCLIP**: ZegCLIP은 zero-shot semantic segmentation을 수행하며, VOC에서 9 FPS, COCO에서 6.7 FPS를 달성했습니다. instance segmentation 미지원.
- **Grounded-SAM-2**: Grounding DINO와 SAM2를 결합한 모델로, 비디오 추적에 중점을 두며, Jetson에서의 성능은 확인되지 않았습니다.
- **Semantic-Segment-Anything**: SAM의 mask에 semantic label을 추가하는 annotation engine으로, 직접적인 segmentation 모델이 아님.
- **SCLIP, CLIP-RT, NanoOWL, EVLA, openvla, ESAM**: 충분한 정보 부족으로 상세 조사 미완료.

4. 알고리즘 비교

알고리즘	Zero-shot 성능	실시간 성능 (Jetson)	출력 형식	ROS2 통합	오픈소스
NanoOWL + NanoSAM	LVIS 31.2 AP_r	95 FPS (NanoOWL), 104 FPS (NanoSAM)	2D bbox, instance mask, 3D 추정 가능	NanoOWL node 제공	GitHub 제공
Grounded-SAM	COCO 52.5 AP	최적화 필요, 가능성 있음	2D bbox, instance mask, 3D 추정 가능	사용자 정의 필요	GitHub 제공
Semantic-SAM	SA-1B 기반, 강력	미확인, 최적화 필요	instance mask, semantic labels	사용자 정의 필요	GitHub 제공
CLIP-SEG	Pascal-VOC mIoU 35.7	미확인, 가능성 있음	binary segmentation mask	사용자 정의 필요	GitHub 제공
MobileSAMv2	LVIS 42.5% AR@K	CPU 3초, Jetson 가능성 있음	instance mask, 언어 필터링 필요	사용자 정의 필요	GitHub 제공

5. 추천 알고리즘

5.1 NanoOWL + NanoSAM

NanoOWL과 NanoSAM의 조합은 사용자의 요구사항에 가장 적합합니다. NanoOWL은 언어 프롬프트를 통해 zero-shot object detection을 수행하며, NanoSAM은 bounding box를 프롬프트로 사용하여 instance segmentation을 제공합니다. Jetson AGX Orin에서 실시간 성능(95 FPS 이상)을 달성하며, ROS2 node가 제공되

어 통합이 용이합니다. RGBD 카메라의 depth 정보를 활용하여 3D 위치 또는 bounding box를 추정할 수 있습니다.

5.2 Grounded-SAM

Grounded-SAM은 Grounding DINO의 높은 정확도와 SAM의 강력한 segmentation 능력을 결합합니다. COCO zero-shot에서 52.5 AP를 달성하여 우수한 성능을 보입니다. Jetson에서 실행 가능하며, TensorRT 최적화를 통해 실시간 성능을 달성할 가능성이 있습니다. 그러나 ROS2 통합을 위해 추가 작업이 필요합니다.

5.3 Semantic-SAM

Semantic-SAM은 언어 프롬프트를 통해 다양한 granularity에서 instance segmentation과 semantic labeling을 제공합니다. SA-1B 데이터셋으로 훈련되어 강력한 zero-shot 성능을 보이지만, Jetson에서의 실시간 성능은 확인이 필요합니다. SwinT backbone을 사용하여 최적화 가능성이 있습니다.

5.4 CLIP-SEG

CLIP-SEG은 간단한 텍스트 프롬프트로 segmentation mask를 생성합니다. 사용이 간단하지만, instance segmentation을 지원하지 않아 다중 객체가 있는 환경에서 grasping에 제한이 있을 수 있습니다. Jetson에서의 성능은 최적화 여부에 따라 달라질 수 있습니다.

6. 결론 및 제언

NanoOWL + NanoSAM은 Jetson AGX Orin에서의 실시간 성능, ROS2 통합 용이성, instance segmentation 제공으로 사용자의 요구사항에 가장 적합합니다. Grounded-SAM은 높은 정확도를 제공하나, 실시간 성능과 ROS2 통합을 위해 추가 작업이 필요합니다. Semantic-SAM은 강력한 기능성을 제공하지만, Jetson에서의 성능 확인이 필요합니다. CLIP-SEG은 간단한 프로토타입 테스트에 적합하나, instance segmentation 미지원으로 제한이 있습니다.

제언:

1. **NanoOWL + NanoSAM 테스트**: GitHub에서 제공되는 패키지를 설치하여 실시간 성능과 ROS2 통합을 확인합니다.
2. **Grounded-SAM 최적화**: TensorRT를 활용하여 Jetson에서의 실시간 성능을 테스트합니다.
3. **Semantic-SAM 평가**: SwinT 모델을 Jetson에서 실행하여 FPS를 측정합니다.
4. **MoveIt2 통합**: 각 알고리즘의 출력(3D 위치, bounding box)을 MoveIt2에 입력하여 모션 플래닝을 구현합니다.

이 보고서는 사용자의 프로젝트 성공을 위해 최적의 알고리즘을 선정하는 데 필요한 모든 정보를 제공합니다.

주요 인용

- NanoOWL GitHub Repository
- NanoSAM GitHub Repository
- Grounded-SAM GitHub Repository
- Semantic-SAM GitHub Repository
- CLIP-SEG GitHub Repository
- MobileSAMv2 arXiv Paper
- Grounding DINO arXiv Paper
- CLIP-SEG arXiv Paper

- NVIDIA NGC Grounding DINO Model
- NanoSAM Hugging Face Model
- ROS2-NanoOWL GitHub Repository