좋습니다. 주신 추가 정보를 반영하여 본격적으로 심층 조사를 시작하겠습니다.

요약하면, 목표는 Jetson AGX Orin + ROS2 Humble + RGBD camera 환경에서 "자연어로 지정한 객체를 인식하고 (위치 정보 포함) manipulator로 grasping"하는 연구를 위한 최적의 알고리즘 조합을 제안하는 것입니다. 특히:

- 실시간성은 5fps 이상.
- 단순 객체부터 시작하지만 generalization 가능성을 염두.
- 3D bounding box 또는 6D pose 추정 여부를 명시.
- 모델 경량성 고려 (특히 tiny/lightweight 버전 존재 여부 명시).
- ROS2 호환성 여부 명시.
- 오픈소스(github 등) 사용성까지 고려.

주어진 알고리즘 목록 + 추가 가능성 있는 최신 알고리즘까지 종합하여 조사하고, 최종적으로 3~6개의 추천 조합을 제안드리겠습니다. 각 알고리즘별 특성과 추천 사유를 상세히 설명하고, 전문적인 보고서 스타일로 한국어로 작성하겠습니다.

조사가 완료되면 바로 공유드리겠습니다!

Jetson AGX Orin 기반 VLM 객체 인식 알고리즘 연구 보고서

배경 및 목표

Jetson AGX Orin (64GB)과 ROS 2 Humble, RGB-D 카메라 환경에서 **사용자 음성이나 텍스트로 입력된 객체** (예: "red cup")를 인식하고 로봇 팔(manipulator arm)이 잡을 수 있도록 **객체 위치 정보를 추출**하는 것이 목표입니다. 이를 위해 **Vision-Language Model (VLM)** 기반 알고리즘을 활용하여, **텍스트**와 **이미지**를 동시에 처리하고 원하는 객체를 찾아내는 기술을 조사합니다. 구체적인 요구사항은 다음과 같습니다:

- **입력**: 카메라로 촬영된 RGB-D 이미지 + 사용자 자연어 질의(예: "빨간 컵").
- **출력**: 해당 객체의 위치 정보 (카메라 좌표계 또는 월드 좌표계의 3D Bounding Box 혹은 6-자유도 자세).
- **속도**: 실시간성 확보 – **최소 5 FPS 이상**으로 동작할 것.
- **일반화**: 초기에는 컵, 휴지심 등 **단순 일상 물체**로 실험하나, 향후 **임의 객체로 일반화** 가능해야 함 (Open-Vocabulary 인식).
- **경량화**: 임베디드 Jetson 환경을 고려하여 **가능한 경량 모델** 선호 (Tiny/Small 모델 존재 시 고려).
- **ROS 2 연동**: ROS 2 시스템에 통합 가능 여부 (기존 패키지 존재 여부 등) 명시.
- **오픈 소스**: GitHub 등에서 사용 가능한 공개 구현 위주로 검토.

이 보고서에서는 위 요건을 만족할 가능성이 있는 최신 VLM 기반 객체인식 알고리즘들을 깊이 조사합니다. 특히 사용자가 관심을 표명한 후보 알고리즘들(SAM2, ZegCLIP, SCLIP, CLIP-SEG, NanoOWL, Semantic-SAM, Semantic-Segment-Anything, EVLA, CLIP-RT, NanoSAM, Grounding DINO, Grounded-SAM-2, OpenVLA, MobileSAMv2, ESAM)에 대해 각각 **기본 원리, 입출력 형태, 성능(FPS 등), 경량화 옵션, ROS2 통합, 장단점**을 분석합니다. 마지막으로 요구사항에 가장 부합하는 **3~6개의 알고리즘(또는 조합)**을 선정하여 추천 이유를 자세히 기술합니다.

주요 알고리즘 분석

CLIP 기반 Zero-Shot 세그멘테이션 알고리즘

Vision-Language 모델인 **CLIP**을 활용하여 **사용자 텍스트에 대응하는 픽셀 단위 분할(segmentation)**을 수행하는 접근들입니다. 즉, 특정 문구에 해당하는 픽셀들을 찾아 마스크를 출력하는 방식으로, open-vocabulary semantic segmentation 문제를 해결합니다.

- **CLIPSeg (CLIP-SEG)**: *"Image Segmentation Using Text and Image Prompts"* (CVPR 2022) ([2112.10003] Image Segmentation Using Text and Image Prompts) ([2112.10003] Image Segmentation Using Text and Image Prompts)에 소개된 모델입니다. CLIPSeg은 **OpenAI CLIP의 시각 백본**(ViT-B/16)을 사용하고 그 위에 **작은 트랜스포머 디코더**를 얹어 훈련한 segmentation 모델입니다 ([2112.10003] Image Segmentation Using Text and Image Prompts). 텍스트 또는 예시 이미지로 주어진 **프롬프트**에 따라 해당 픽셀 영역을 0/1 마스크로 출력합니다 ([2112.10003] Image Segmentation Using Text and Image Prompts). CLIPSeg은 **freeze된 CLIP 비전 인코더**를 사용하고 소량의 추가 파라미터만 학습하여 효율적으로 구현되었으며, 훈련 시 PhraseCut 등 데이터셋을 활용해 **임의 문구에 대한 분할** 능력을 갖추었습니다 ([2112.10003] Image Segmentation Using Text and Image Prompts). 입력은 RGB 이미지와 텍스트(query)이며, 출력은 해당 객체의 이진 마스크입니다. 성능 면에서, CLIPSeg은 **CLIP 사전학습 덕분에** 일반화 성능이 높고, 작은 ViT-B 기반이라 비교적 가볍습니다 (CLIP ViT-B/16 약 86M param 규모). 논문에 언급된 **실험 설정에서는** 해상도 352px 이미지 기준으로 **실시간에 가까운 속도**를 보입니다. 다만 정확한 FPS는 공개되지 않았으나, 파라미터 규모나 연산량으로 미루어 Jetson에서도 **수 FPS 수준으로 가능**할 것으로 예상됩니다. 경량화 옵션으로 더 작은 CLIP(backbone)을 쓰는 것도 가능하며, 원 논문에서는 CLIP ViT-L은 공개 시점에 사용 불가해 ViT-B로 한정했다고 명시되어 있습니다 ([2112.10003] Image Segmentation Using Text and Image Prompts). **ROS 2 통합** 패키지는 따로 없지만, PyTorch 코드가 공개 ([2112.10003] Image Segmentation Using Text and Image Prompts)되어 있어 자체적으로 노드 작성이 어렵지 않습니다. **장점:** 다양한 텍스트 입력에 유연하고, 별도 학습 없이 **Zero-shot 세그멘테이션**이 가능합니다. **단점:** instance 구분 없이 해당 클래스의 모든 픽셀을 한 마스크로 출력하므로, 동일 클래스 다수가 있는 경우엔 추가 처리가 필요합니다. 또한 출력이 2D 마스크이므로 3D 좌표로 변환하려면 깊이정보 후처리가 필요합니다.

- **ZegCLIP**: *"Towards Adapting CLIP for Zero-shot Semantic Segmentation"* (CVPR 2023)으로, CLIP을 **semantic segmentation**에 직접 응용한 최신 연구입니다. ZegCLIP은 **CLIP의 텍스트-이미지 예측 능력을 한 단계 확장**하여, 픽셀 단위로 **Zero-shot 분류**가 가능하도록 설계되었습니다. 구조적으로, 별도의 두 단계 (예: 영역 제안 후 분류)를 거치지 않고 **원스테이지**로 작동하는 세그멘터입니다 (Towards Adapting CLIP for Gaze Object Prediction - ResearchGate). 구체적으로, CLIP의 이미지 인코더 출력에 **가벼운 디코더**와 **학습가능한 토큰** 등을 추가하여, 텍스트 프롬프트로 지정된 카테고리의 마스크를 바로 예측합니다 ((PDF) ZegCLIP: Towards Adapting CLIP for Zero-shot Semantic ...) (GitHub - jingyi0000/VLM_survey: Collection of AWESOME vision-language models for vision tasks). 입력은 텍스트 리스트(관심 클래스들)와 이미지, 출력은 각 텍스트에 대한 세그멘테이션 마스크입니다. 성능 측면에서, 저자들은 기존 이단계 방식 대비 **약 5배 빠른 추론 속도**를 보고하였으며 (Towards Adapting CLIP for Gaze Object Prediction - ResearchGate), 이는 실시간 응용에 한 걸음 다가선 결과입니다. 다만 절대 FPS 수치는 백본 크기에 따라 다를 것입니다. (예컨대 CLIP ViT-L 기반이면 수 FPS, ViT-B 기반이면 더 빨라질 수 있음). 정확도는 COCO-Stuff 등에서 SOTA에 근접하는 zero-shot 성능을 보였다고 하며 (GitHub - jingyi0000/VLM_survey: Collection of AWESOME vision-language models for vision tasks), 배경/전경 분리 없이 **개방형 클래스 세그멘테이션**이 가능합니다. 경량화 옵션으로 CLIP의 작은 모델(ViT-B/32 등) 사용이나 디코더 경량화가 가능하지만 논문 기본설정은 CLIP-L로 추정됩니다. **ROS2 통합** 구현은 공식 제공되지 않았고, Python 코드로 PyTorch 구현이 제공되는 형태입니다. **장점:** open-vocabulary로 **미학습 카테고리도 바로 분할** 가능하고, end-to-end로 간결하여 효율적입니다. **단점:** 대형 CLIP 백본 사용 시 임베디드에서는 속도가 제한될 수 있고, 또한 문장 단위 복잡한 프롬프트보다는 단일 명사에 최적화되어 있을 수 있습니다.

- **CLIP-ES ("CLIP is Also an Efficient Segmenter", 사용자 정의 SCLIP)**: *"CLIP is Also an Efficient Segmenter: A Text-Driven Approach for Weakly Supervised Semantic Segmentation"* (CVPR 2023) (GitHub - linyq2117/CLIP-ES: [CVPR 2023] CLIP is Also an Efficient Segmenter: A Text-Driven Approach for Weakly Supervised Semantic Segmentation)은 CLIP의 잠재력을 **약한 지도학습**으로 끌어낸 사례입니다. 이 방법은 클래스별 마스크 레이블이 없는 **이미지-텍스트 쌍** 데이터만으로 CLIP 기반 세그멘테이션 모델을 학습하여, **텍스트만으로 픽셀 분할**을 해냅니다. 구조적으로 CLIP-ES도 CLIP 비전 인코더를 사용하며, 그 위에 클래스 마스크를 예측하는 헤드를 약지도 학습합니다. 입력은 이미지와 텍스트(관심 객체 이름)이며, 출력은 해당 객체의 마스크입니다. **성능:** 완전한 Zero-shot보다는 **학습된 범주에 약간 의존**하지만, 텍스트로 세밀 조정이 가능하고 배경 분리 등의 품질이 준수합니다. 다만 해당 연구는 **효율적인 학습**에 초점을 두었지, 경량 모델 구현을 목표로 한 것은 아니어서, 추론 속도는 CLIP 백본 크기에 비례합니다. ViT-B 등으로도 구현 가능하나 정확도는 낮아질 수 있습니다. **ROS 2 패키지**는 제공되지 않으며, 연구용 PyTorch 코드만 있습니다. **장점:** 적은 레이블로도 학습 가능하여 새로운 도메인에 적용하기 용이하고, **텍스트 지시를 통한 세그멘테이션** 성능을 입증했습니다. **단점:** 완전 사전학습된 모델(ZegCLIP 등) 대비 특정 데이터에 최적화된 경향이 있고, open-vocabulary 수준의 범용성은 약간 떨어질 수 있습니다.

(요약: CLIP 기반 세그멘테이션들은 공통적으로 **자연어 질의에 해당하는 픽셀 마스크**를 생성해주므로, RGB-D 카메라 입력을 활용하면 마스크 영역의 **3D 좌표**를 추출 가능하다는 장점이 있습니다. Jetson에서도 ViT-B 수준 모델은 **수 FPS 실시간 처리**가 기대되나, 더 큰 모델은 최적화(예: TensorRT) 없이는 어려울 수 있습니다.**)**

Segment Anything 기반 모델 (SAM2, MobileSAM 등)

Meta AI의 **Segment Anything Model (SAM)**은 사전 학습된 **범용 세그멘테이션** 모델로, 입력 이미지에 대해 **프롬프트(점, 박스 등)**가 주어지면 해당 영역의 마스크를 정밀하게 산출합니다. 본 절에서는 SAM 및 파생 모델들을 다룹니다. SAM 자체는 **텍스트 입력을 직접 받지는 못하기 때문에**, 여기서는 SAM을 **텍스트 지시와 결합**하거나 **경량화**한 연구들을 중점적으로 살펴봅니다.

- **SAM2 (Segment Anything Model 2)**: 이는 2024년에 Meta가 공개한 SAM의 차세대 버전입니다 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium). **SAM2**는 **동영상까지 포함한 범용 세그멘테이션**을 목표로 설계된 모델로, **이미지와 비디오 모두에 단일한 아키텍처**를 적용합니다 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium). 주요 특징은 **실시간 성능(real-time performance)**과 **Zero-shot 일반화 능력**입니다 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium). Meta에 따르면 SAM2는 **실시간 영상 처리**가 가능하고, 새로운 객체 카테고리에 대해 **별도 재학습 없이도** 대응할 수 있다고 합니다 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium). 또한 **대화형 세분화** 기능이 있어, 사용자 입력(텍스트 등)을 통해 마스크를 개선(refine)하는 인터랙티브한 사용도 지원합니다 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium). 입력 프롬프트 형태는 SAM과 유사하게 점/박스 등을 받지만, SAM2에서는 여기에 **텍스트 설명도 일부 활용**되는 것으로 알려져 있습니다 (예: 텍스트로 원하는 클래스를 지정하여 해당 영역을 강조). 출력은 하나 혹은 다수 객체의 정밀한 마스크이며, 비디오의 경우 **프레임 연속성**을 고려한 결과를 얻습니다. 성능 면에서, SAM2의 논문 및 튜토리얼은 **SAM 대비 향상된 복잡한 환경에서의 정확도**와 **실시간 처리속도**를 강조하고 있습니다 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium) (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium). 다만 Jetson에서 실시간이 가능한지는 별도 검증이 필요하며, Meta가 제공한 공식 구현은 주로 서버급 GPU를 대상으로 합니다. 모델 경량화 옵션은 공식으로는 없으나, SAM2 자체가 **효율적 추론**을 고려해 **스트리밍 메모리** 등 기법을 도입한 것으로 보입니다 (GitHub - facebookresearch/sam2: The repository provides code for running inference with the Meta Segment Anything Model 2 (SAM 2), links for downloading the trained model checkpoints, and example notebooks that show how to use the model.). **ROS2 통합**: 현재까지 ROS2 패

키지는 발표되지 않았습니다. 오픈소스 PyTorch 코드 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium)가 있으므로 필요한 경우 ROS2 노드로 래핑해야 합니다. **장점:** SAM의 범용성에 실시간성과 Zero-shot 인식을 더해 **동적 장면의 모든 객체 세그멘테이션**까지 가능하다는 점에서 업계 최첨단입니다. **단점:** 모델이 아직 대규모여서 Jetson에서 효율을 보장하기 어려울 수 있고, 또한 텍스트를 직접 입력받는 방식은 제한적(보조적 역할)이라, **자연어 지시를 1단계에서 처리하는 기능은 제한**적입니다.

- **MobileSAM & MobileSAMv2**: **MobileSAM**은 SAM의 **경량화/고속화 버전**으로 2023년 등장했습니다. 원본 SAM의 비전 트랜스포머(ViT-H, 632M param)가 **지나치게 무거운 점**을 개선하기 위해, MobileSAM은 **지식 증류(knowledge distillation)** 기법으로 작은 백본 모델을 훈련했습니다 ([2312.09579] MobileSAMv2: Faster Segment Anything to Everything). MobileSAMv1은 주로 **SegAny (단일 객체 세그멘트)** 상황에서 **추론속도 향상**을 달성하였고, MobileSAMv2는 더 나아가 **SegEvery (전체 객체 세그먼트)**의 속도를 개선했습니다 ([2312.09579] MobileSAMv2: Faster Segment Anything to Everything) ([2312.09579] MobileSAMv2: Faster Segment Anything to Everything). MobileSAMv2는 SAM의 **mask decoder** 단계에서 불필요한 grid 검색을 제거하고 **object-aware한 프롬프트 샘플링**을 도입하여, **마스크 디코더 연산을 16배 이상 가속**했다고 보고됩니다 ([2312.09579] MobileSAMv2: Faster Segment Anything to Everything). 예를 들어, MobileSAMv2는 **사전에 객체 후보를 발견**하여 그 위치들만 마스크로 세그먼트하여, 원본 SAM처럼 수백 개 점을 찍어보는 과정을 대폭 줄입니다. 결과적으로 **SegEvery 작업에서 전체 성능 3.6% 향상**과 **속도 큰 폭 개선**을 동시에 달성했습니다 ([2312.09579] MobileSAMv2: Faster Segment Anything to Everything). 입력은 SAM과 마찬가지로 point 혹은 box prompt이며, MobileSAMv2의 경우 내부적으로 자동 prompt 생성으로 **이미지 내 모든 객체를 분할**할 수도 있습니다. 출력은 SAM과 동일한 마스크들입니다. 성능 예를 들면, MobileSAM (v1)은 ViT-H 대비 수십 배 빠르게 동작하며, MobileSAMv2 적용 시 **전체 파이프라인 속도가 기존 SAM 대비 월등히 향상**됩니다. MobileSAM 저자들은 **코드와 모델을 공개**하였고 (ChaoningZhang/MobileSAM: This is the official code for ... - GitHub), HuggingFace에도 가중치가 올라와 있습니다. **ROS2 통합** 패키지는 별도로 없으며, 필요시 ROS2 노드에 PyTorch 혹은 ONNXRuntime 형태로 삽입해야 합니다. **장점:** SAM의 강력한 segmentation 능력을 **임베디드 환경에서 활용 가능**하도록 한 점입니다. 특히 MobileSAMv2는 한 장의 이미지에서 **모든 객체를 빠르게 찾는** 기능이 있어, SAM의 활용폭을 넓힙니다. **단점:** MobileSAM 시리즈도 여전히 **텍스트 이해 능력은 없고**, 어디까지나 세그멘테이션 전용이라 원하는 객체를 선택하려면 다른 방법(예: text->point 변환 등)이 필요합니다. 또한 distillation으로 미세 정확도가 소폭 감소할 수 있습니다 (NanoSAM 결과에서 보듯 mIoU 약간 하락) (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT).

- **NanoSAM**: **NVIDIA**가 Jetson 장치 최적화를 위해 개발한 SAM 파생모델입니다. NanoSAM은 **MobileSAM의 이미지 인코더를 distill**하여 얻은 **ResNet18 기반 경량 SAM**입니다 (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT). 즉, 거대 ViT 대신 작은 CNN백본으로 SAM과 유사한 출력을 내도록 한 것입니다. NanoSAM은 TensorRT 등을 통해 극대화된 속도를 보이는데, **Jetson AGX Orin에서 전체 추론 8.1ms**만에 마스크를 얻을 정도입니다 (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT). 이는 **약 125 FPS**에 해당하므로 사실상 실시간을 넘어서는 속도입니다. 정확도도 MobileSAM 대비 소폭만 감소하여, **원본 SAM과 유사한 품질(mIoU 0.80 수준)**을 유지합니다 (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT). 입력/출력 인터페이스는 기본 SAM과 동일하며, point, box prompt 등을 여러 개 넣어 동시에 여러 객체를 세그멘테이션할 수 있습니다 (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT). **ROS2 통합:** NanoSAM 전용 ROS2 패키지는 나오지 않았으나, NanoSAM은 **NanoOWL과의 결합 예제** 등으로 제공되어 NVIDIA의 ROS 데모에서 활용됩니다 (GitHub - NVIDIA-AI-IOT/nanoowl: A

project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (Detection+Segmentation 조합, 후술). **장점: Jetson에서 가장 빠른 SAM 대안**으로, 지연이 매우 낮아 로봇 실시간 제어에도 활용 가능할 수준입니다. **단점:** NanoSAM 자체로는 특정 객체를 식별하지 않으며, 프롬프트로 어떤 픽셀을 자를지 인간이 지정해야 합니다. 따라서 **텍스트 지시를 바로 반영할 수 없고**, 검출 모듈과 연결이 필요합니다.

- **Efficient SAM (Edge SAM, ESAM 등)**: SAM 공개 이후 여러 연구진이 **SAM의 경량화**를 시도했습니다. MobileSAM/NanoSAM 외에도, **FastSAM**, **Efficient-SAM**, **EdgeSAM** 등 다양한 구현이 존재합니다 (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything). 예를 들어 **FastSAM**은 ViT-H 대신 YOLO 스타일 백본을 사용하여 SAM과 유사한 출력을 매우 빠르게 생성하는 모델로 알려져 있고, **EdgeSAM**은 엣지 디바이스(모바일 GPU 등)에서 원본 품질을 내려 최소한의 경량화를 이룬 변형입니다. 이러한 모델들의 공통 목표는 **SAM의 추론 속도나 메모리 사용량을 줄이는 것**이며, 몇몇은 정확도보다 속도에 초점을 맞춰 **CPU상에서도 수백 ms 내 출력**을 내기도 합니다. 다만, 공식 논문이 아닌 구현 수준인 것도 있어 신뢰성과 재현성은 검증이 필요합니다. **ROS2 지원**은 대부분 없으며, 필요에 따라 PyTorch 혹은 ONNX 변환 후 ROS2에서 사용해야 합니다. **장단점:** 실시간성 측면에서 유리하나, **정확도 손실**가 있을 수 있고 SAM처럼 광범위하게 검증되지 않았다는 위험이 있습니다.

(요약: SAM 계열 모델들은 **텍스트 프롬프트 직접 입력 기능은 부족**하지만, **정교한 마스크 생성** 능력이 뛰어납니다. 따라서 **텍스트로 지목된 객체를 정확히 집기 위해**, 검출 단계와 SAM을 **결합**하는 방식으로 활용됩니다. SAM2는 향후 발전 방향을 제시하지만, 현 시점 Jetson에서는 NanoSAM 같은 **경량 SAM**을 검출기와 조합하는 것이 현실적입니다.**)**

개방형 객체 검출 모델 (텍스트 기반 Object Detection)

자연어 텍스트로 질의한 객체의 **2D 경계상자(bounding box)**를 예측하는 알고리즘들입니다. 이러한 **Open-Vocabulary Object Detection** 접근은, 사전에 학습되지 않은 카테고리도 **텍스트 설명만으로 인식**할 수 있다는 장점이 있습니다. 여기서는 사용자가 언급한 **OWL-ViT (NanoOWL)**, **Grounding DINO**, 그리고 관련 개념들(CLIP-RT 등)을 다룹니다.

- **OWL-ViT 및 NanoOWL**: **OWL-ViT**는 Google Research가 2022년에 공개한 **Open-Vocabulary Detection** 모델로, CLIP처럼 **비전-언어 쌍학습**을 통해 **임의의 객체 이름에 대한 Bounding Box**를 찾습니다. ViT 기반의 두 단계 구조(인코더+디코더)로 동작하며, 입력 이미지와 텍스트(예: ["cup", "bottle"] 등)를 받아 각 텍스트에 해당하는 2D 박스를 출력합니다. **NanoOWL**은 NVIDIA가 이 OWL-ViT를 Jetson에 최적화한 프로젝트로, **TensorRT 가속**을 통해 **실시간 추론**을 구현했습니다 (NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT ... - GitHub). NanoOWL은 OWL-ViT의 ViT-B/32, B/16 등 백본에 대하여 엔진을 미리 빌드하여 사용하며, Jetson AGX Orin에서 **ViT-B/32 모델은 95 FPS**, **ViT-B/16 모델도 25 FPS**를 달성했습니다 (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.). 이는 매우 인상적인 속도로, Jetson에서 **텍스트 기반 객체 검출을 25~90FPS 범위**로 실현한 것입니다. NanoOWL의 출력은 각 텍스트 라벨에 대한 **Bounding Box 좌표와 점수**이며, 별도로 마스크는 제공하지 않습니다. 하지만 **검출된 박스를 후처리**하여 ROI의 깊이 값을 추출하면 3D 위치 추정이 가능합니다. 또한 NanoOWL 예제에는 **NanoSAM과 연계**하여 검출된 박스 내부를 정교하게 마스크화하는 활용도 포함되어 있습니다 (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.). **ROS2**

통합: NVIDIA는 NanoOWL의 ROS2 패키지인 **ROS2-NanoOWL**을 제공하고 있습니다 (GitHub - NVIDIA-AI-IOT/ROS2-NanoOWL: ROS 2 node for open-vocabulary object detection using NanoOWL.). 이 패키지를 이용하면 카메라 토픽으로부터 텍스트 질의를 받아 검출 결과를 바로 ROS2 좌표계로 퍼블리시할 수 있습니다. **장점:** NanoOWL (OWL-ViT)은 **임의의 객체 이름으로 탐색**이 가능하며, 특히 **NVIDIA의 최적화로 Jetson에서 실사용 가능한 속도**를 보장하는 거의 유일한 VLM 검출기입니다. **단점:** 마스크나 자세를 직접 주지는 않으므로, **정밀한 Grasping을 위해서는** 추가 단계(SAM으로 마스크 추출 등)가 필요합니다. 또한 텍스트 질의 간 유사도가 높을 경우 (예: 'cup' vs 'mug') 혼동 가능성이 있고, 장면 복잡도가 매우 높을 때는 성능이 떨어질 수 있습니다.

- **Grounding DINO**: OpenGVLab 등이 2023년 공개한 **Grounding DINO**는 DETR 계열의 **Transformer 기반 검출 모델**에 **텍스트 임베딩**을 융합하여 **자유도 높은 질의**에 응답하는 **오픈셋 검출기**입니다 (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything). Grounding DINO는 입력으로 이미지와 프리폼 텍스트(prompt)를 받아, 텍스트와 연관된 모든 객체들의 2D bounding box들과 confidence를 출력합니다. 예를 들어 “red cup on table” 같이 문장 형태도 처리하며, 해당되는 모든 빨간 컵을 찾아냅니다. 이 모델은 **대용량 비전-언어 데이터**로 학습되어 일반적인 사물에 대한 뛰어난 Zero-shot 성능을 보입니다. 성능 측면에서, Jetson AGX Orin에서 **1080p 해상도 영상** 처리 기준 **약 11.8 FPS**가 보고되었습니다 (Grounding DINO (GDINO) — Jetson Platform Services documentation). (NVIDIA Jetson 서비스 문서에 따르면 “car, bike, person, bus” 4개 질의에 11.8fps 달성 (Grounding DINO (GDINO) — Jetson Platform Services documentation)). Orin NX(16GB)에서는 약 4.6 FPS, Orin Nano(8GB)에선 4 FPS 이하로 떨어져 AGX급에서 주로 사용됩니다 (Grounding DINO - NGC Catalog - NVIDIA). Grounding DINO는 추론속도를 위해 Swin-Transformer-B 등 다양한 백본을 쓸 수 있고, 최근 **Grounding DINO 1.5 Edge**라는 경량화 버전이 발표되어 **Orin NX에서 10 FPS (640×640 입력)**를 달성했다고 합니다 (Too many objects, so little time: Open set object detection work by my... | Harry Shum) (Too many objects, so little time: Open set object detection work by my... | Harry Shum). 이를 AGX Orin으로 환산하면 수십 FPS도 가능하여, **향후 더욱 실시간에 가까워질 전망**입니다. 모델 경량화 옵션으로 공식 Edge 모델 외에도, NVIDIA TAO 툴킷으로 가속된 버전이 언급되고 있습니다 (Eyal Enav - Accelerated Grounding Dino with NVIDIATAO - LinkedIn). **ROS2 통합:** 공식 ROS2 패키지는 없으나, NVIDIA에서 Triton Inference Server 기반 마이크로서비스로 GroundingDINO를 컨테이너 제공하고 있고 (Grounding DINO (GDINO) — Jetson Platform Services documentation) (Grounding DINO (GDINO) — Jetson Platform Services documentation), Roboflow 등에서도 Jetson 배포 예제를 제공합니다. ROS2 노드로 활용하려면 Python API 혹은 ONNX 변환 후 wrapping이 필요합니다. **장점:** GroundingDINO는 **텍스트 조건부 검출** 정확도가 매우 높아, COCO/LVIS와 같은 벤치마크의 미등록 카테고리까지 탐지하는 **개방형 인지** 능력을 보여줍니다. 또한 하나의 프롬프트에 여러 객체 키워드를 넣으면 **동시에 여러 종류**를 찾아내는 것도 가능합니다. **단점:** 연산량이 많아 Jetson에서는 **최적화 없이 다소 느릴 수 있고**, 큰 Transformer 모델이라 메모리 사용량이 높습니다. 또한 2D bounding box만 출력하므로, 후속 세그멘테이션이나 3D 추정은 추가로 구현해야 합니다.

- **CLIP-RT (실시간 CLIP 기반 검출)**: 명시적인 알고리즘 이름이라기보다, **실시간에 가까운 CLIP 활용 객체 검출 기법**을 지칭하는 것으로 해석됩니다. 예를 들어, **YOLO 등 경량 검출기의 제안영역을 CLIP으로 재분류**하거나, 앞서 소개한 NanoOWL처럼 **CLIP계열 open-vocabulary 모델을 TensorRT로 최적화**하는 식의 접근이 이에 해당합니다. 실제로 NVIDIA NanoOWL 프로젝트가 **“CLIP 기반 객체 검출을 실시간화”**한 좋은 사례라 볼 수 있습니다. 또한 최근 연구 중에는 **CLIP의 이미지 임베딩과 SAM 등의 결합**으로 특정 텍스트에 해당하는 마스크를 실시간 추출하는 실험들도 있습니다. **CLIP-RT의 구체 사례**로는, Microsoft Research가 소개한 Grounding DINO 1.5 Edge 모델도 CLIP-RT 개념에 부합하는데, 이 모델은 **640×640 입력에 대해 10 FPS**를 Orin NX에서 달성하며, **검출 정확도와 효율을 양립**했습니다 (Too many objects, so little time: Open set object detection work by my... | Harry Shum). 요컨대 “CLIP-RT”는 **open-**

vocabulary 모델의 경량 최적화를 의미하며, **NanoOWL (OWL-ViT)**이나 **Edge GroundingDINO** 등이 현실화된 예입니다. **ROS2 통합:** 개별 구현마다 다르지만, 앞서 언급한 NanoOWL ROS2 패키지가 대표적인 **실시간 CLIP 검출 ROS 통합** 사례입니다 (GitHub - NVIDIA-AI-IOT/ROS2-NanoOWL: ROS 2 node for open-vocabulary object detection using NanoOWL.). **장점: 텍스트 지시를 바로 실행**하여 빠르게 Bounding Box를 얻을 수 있고, 기존 폐쇄형 검출기(클래스 한정)와 달리 **유연한 객체 인식**이 가능합니다. **단점:** 파이프라인이 복잡해질 수 있고, 만약 YOLO 등과 결합 시 **두 모델 추론 지연**이 누적될 수 있습니다. 하지만 이러한 접근은 계속 개선되고 있으며, Jetson에서 **5 FPS 이상** 달성은 충분히 가능하므로 의미 있는 방향입니다.

(요약: 텍스트 기반 객체 검출은 **로봇이 특정 물체를 찾는 1단계**로 매우 유용합니다. NanoOWL (OWL-ViT)과 GroundingDINO는 대표적 방법이며, 특히 NanoOWL은 **Jetson에서 실시간으로 다중 객체 탐지**를 지원하여 유망합니다. Bounding Box 결과에 RGB-D의 깊이를 접목하면 **3D 위치 추정**이 가능하며, 필요시 이어지는 SAM 계열로 **정교한 마스크**를 얻어 후속 작업(그리핑 포인트 계산 등)에 활용할 수 있습니다.**)**

결합/통합 접근: 검출 + 세그멘테이션 + 인식

위에서 개별적으로 살펴본 **텍스트기반 검출**과 **세그멘테이션**을 **조합**하면, 사용자가 원하는 **"자연어로 특정 객체를 인식하고, 그것만 마스크로 분리"**하는 완전한 파이프라인이 구축됩니다. 이러한 통합 접근의 대표적인 예로 **Grounded-SAM**, **Semantic-SAM** 등이 있으며, **Semantic-Segment-Anything (SSA)** 같은 시스템도 존재합니다. 또한 Vision-Language 모델을 로봇 제어까지 확대한 **OpenVLA** 같은 시도도 있습니다. 이 절에서는 이러한 **융합 알고리즘 및 시스템**을 설명합니다.

- **Grounded-SAM (그라운디드 SAM)**: 이는 하나의 모델이 아니라, 2023년 초 커뮤니티에서 고안된 **파이프라인**으로, **텍스트 조건부 검출 모델**과 **Segment Anything**을 결합한 것입니다 (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything). 예컨대 **Grounding DINO로 텍스트 "cup"에 해당하는 bounding box**를 찾고, 그 영역에 **SAM**을 적용하여 픽셀 단위 마스크까지 얻게 됩니다 (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything). 이렇게 하면 사용자는 단순히 "cup"이라고만 명령해도, 이미지 속 모든 컵을 정확히 분할해낼 수 있습니다. **Grounded-SAM 2**는 이러한 파이프라인을 확장/개선한 버전으로, **더 정교한 분할(SAM-HQ 등 활용)**과 **다양한 입력 모달리티**(음성 등) 를 추가한 것을 지칭하는 것으로 보입니다. 실제 GitHub 프로젝트에서 Grounded-SAM은 Stable Diffusion을 결합해 분할 영역을 편집/생성하거나, BLIP/ChatGPT를 연결해 자동 캡셔닝/라벨링하는 등 꾸준히 확장되었습니다 (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything) (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything). **입출력:** 사용자 텍스트 질의 -> 대상 객체의 마스크들. **성능:** Grounded-SAM 조합의 속도는 사용하는 검출기+SAM 모델에 따라 결정됩니다. 기본적으로 **GroundingDINO-L + SAM-H(ViT-H)** 조합은 매우 무겁기 때문에 GPU 워크스테이션에서도 수 FPS 수준이며, Jetson에서는 수백 ms~수 초 단위로 느릴 수 있습니다. 하지만 앞서 소개한 **NanoOWL + NanoSAM**과 같이 **경량 모델 조합을 사용하면 Jetson에서도 실시간 Grounded-SAM 구현**이 가능합니다. 사실 NanoOWL 깃허브 예제는 tree 구조 탐지에 NanoSAM을 써서 **instance segmentation**까지 수행하고 있는데, 이는 **경량 Grounded-SAM**이라고 볼 수 있습니다 (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.). **ROS2 통합:** Grounded-SAM 자체는 스크립트 조합이라 ROS2 패키지는 없지만,

NanoOWL ROS2 노드 결과를 받아 NanoSAM을 처리하는 노드를 추가하는 식으로 구성하면 ROS2 환경에서 동작시킬 수 있습니다. **장점: 텍스트 => 픽셀 마스크**로 바로 이어지는 **end-to-end 인식**을 실현하여, 사용 편의성과 범용성이 최고입니다. 여러 물체를 동시에 찾아 각기 마스크로 구분할 수도 있어 응용력이 높습니다. **단점:** 전체 파이프라인를 구성하는 모델 수가 늘어나 **시스템 복잡성**이 증가하고, 최적화하지 않으면 지연이 길어질 수 있습니다. 또한 각 구성요소(검출기, SAM)의 오차가 누적될 수 있습니다. 그러나 이러한 조합은 이미 많은 실험에서 성공적임이 입증되어, 현재 **가장 주목받는 접근 방식**이라 할 수 있습니다.

- **Semantic-SAM (Segment & Recognize Anything Model)**: 이것은 MSRA(마이크로소프트 리서치 아시아) 등이 2023년에 제안한 **범용 세그멘테이션 모델**로, 앞서 설명한 SAM에 **semantic 인식 능력**을 통합하려는 시도입니다 (Semantic-SAM: Segment and Recognize Anything at Any Granularity). Semantic-SAM (약칭 SEEM으로도 불림)은 **여러 종류의 segmentation 데이터셋**(일반 객체, 파트(part) 분할, SA-1B 등)을 통합하여 거대한 모델을 훈련한 것입니다 (Semantic-SAM: Segment and Recognize Anything at Any Granularity). **특징:** 하나의 모델이 **다양한 그랜율(granularity)**의 마스크를 출력하고, 각 마스크에 대한 **의미있는 레이블**까지 예측합니다 (Semantic-SAM: Segment and Recognize Anything at Any Granularity). 예를 들어, 한 점 프롬프트를 주면 그 위치에 대한 **Instance 수준 마스크**와 **Part 수준 마스크**를 모두 생성하고, 각각 **“person”, “person’s head”**처럼 계층적 라벨을 붙이는 식입니다 ([PDF] Semantic-aware SAM for Point-Prompted Instance Segmentation). 이를 위해 Semantic-SAM은 SAM의 강력한 분할 데이터(SA-1B)를 **COCO, ADE20K 등의 주석 데이터와 함께** 훈련에 활용하였고, Transformer 디코더를 개선하여 **클릭 한 번에 다중 마스크**를 생성하도록 했습니다 (Semantic-SAM: Segment and Recognize Anything at Any Granularity) (Semantic-SAM: Segment and Recognize Anything at Any Granularity). **입출력:** 프롬프트(점, 박스, 텍스트 등 다양하게 지원) -> 여러 마스크 + 각 마스크의 범주 레이블. **성능:** 논문에 따르면 일반적인 semantic/instance segmentation 벤치마크에서 우수한 성능을 보였고, 특히 **한 번의 추론으로 복수 종류의 분할을 해내는 점**이 혁신적입니다 (Semantic-SAM: Segment and Recognize Anything at Any Granularity). 다만 모델 크기가 커서 속도는 ViT-H SAM 수준 (몇 FPS)으로 추정됩니다. 다행히 공개된 체크포인트 중 **SEEM-Tiny**가 존재하여 경량 모델도 실험되고 있습니다 (GitHub - UX-Decoder/Segment-Everything-Everywhere-All-At-Once: [NeurIPS 2023] Official implementation of the paper "Segment Everything Everywhere All at Once"). **ROS2 통합:** 공식 패키지는 없으나, PyTorch 코드와 모델이 공개되어 있어 필요하면 이식해야 합니다. **장점: 한 모델로 시각적 세그멘테이션 거의 모든 것**을 할 수 있는 만능에 가깝습니다. 텍스트 프롬프트도 지원하므로, “cat”이라고 보내면 고양이 마스크를 바로 얻는 식의 활용도 가능하며 (GitHub - UX-Decoder/Segment-Everything-Everywhere-All-At-Once: [NeurIPS 2023] Official implementation of the paper "Segment Everything Everywhere All at Once"), 이는 Grounded-SAM을 단일 모델로 구현한 비전으로 볼 수 있습니다. **단점:** 모델이 크고 복잡하여 **임베디드 실시간성은 확보되지 않았고**, 훈련 데이터에 존재하지 않는 특이한 도메인에 대한 일반화는 미지수입니다. 또한 출력 레이블이 100% 신뢰도 높은 것은 아니므로 (오인식 가능), 로봇에서 바로 semantic 정보를 활용할 때는 주의가 필요합니다.

- **Semantic Segment Anything (SSA)**: 이는 Fudan 대학 Zhang-Vision 그룹이 발표한 **SAM 기반 자동 레이블링 엔진**입니다 (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).). SAM 자체는 어떤 픽셀들이 같은 객체인지 잘라주지만 **그 객체가 무엇인지는 알려주지 않기 때문에**, SSA는 **SAM의 마스크 각각에 의미있는 태그를 붙여주는 파이프라인**입니다 (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).) (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).). SSA의 동작은 다음과 같습니다: 우선 SAM을 사용해 이미지 내 **모든 object 마스크**를 얻습니다. 그 다음, **별도의 이미지-caption 모델**(BLIP 등)이나 **open-**

set 분류기(CLIPSeg 등)를 활용해 각 마스크에 대한 **텍스트 라벨**을 예측합니다 (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).). 예를 들어 한 이미지에 사람, 강아지, 의자가 있다면 SAM은 세 마스크를 내고, SSA 엔진은 이를 "person", "dog", "chair"로 분류해주는 식입니다. 이렇게 얻은 **방대한 자동 라벨 마스크 데이터**를 **SA-1B** 같은 데이터셋 구축이나, Semantic-SAM 모델 학습 등에 활용할 수 있습니다 (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).) (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).). **실시간 활용 측면:** SSA는 본래 **데이터셋 어노테이션 자동화**용도로 제안되었기에, 한 장면을 여러 모델(SAM+CLIPSeg 등)로 처리하므로 **추론 시간이 길고 무겁습니다**. Jetson에서 직접 SSA 파이프라인을 돌려 실시간 인식에 쓰기는 현실적으로 어려우며, 대신 SSA 개념을 응용하여 **사전 수집한 환경 이미지들을 자동 라벨링**해두고, 그 정보를 기반으로 특정 상황에 로봇을 적용하는 방향이 현실적입니다. **ROS2 통합:** 해당 파이프라인은 웹 데모 등으로는 제공되나 ROS2와는 무관합니다. **장점: 휴먼 레이블 없이도 대규모의 의미있는 분할데이터**를 얻을 수 있어, 자체적인 모델 재학습이나 커스텀 인식모델 구축에 도움이 됩니다. **단점:** 온라인 실시간 추론보다는 **오프라인 데이터 준비**에 가깝기 때문에, 즉각적인 로봇 제어에는 직접 쓰기 어렵습니다. 또한 CLIP 등으로 라벨링하므로 **라벨 품질이 완벽하지 않습니다** (필요시 일부 사람의 검수 필요).

- **OpenVLA (Vision-Language Action)**: **OpenVLA**는 앞선 인식 단계를 넘어, **시각-언어 모델로 로봇의 행동 제어까지** 수행하려는 최첨단 연구입니다. NVIDIA Jetson AI Lab에서 정리한 바에 따르면, OpenVLA는 **LLM (거대 언어모델)과 VLM을 결합하여 이미지+텍스트를 입력받아 로봇 제어 명령을 생성하는** 모델입니다 (OpenVLA - NVIDIA Jetson AI Lab) (OpenVLA - NVIDIA Jetson AI Lab). 구체적으로, **Llama 7B**와 **DINOv2 + SigLIP 비전인코더**로 구성된 OpenVLA-7B 모델은 카메라 이미지와 자연어 명령(예: "Grab the red cup on the table")을 입력받아, 바로 **로봇의 joint 움직임 시퀀스**를 토큰 형태로 출력합니다 (OpenVLA - NVIDIA Jetson AI Lab) (OpenVLA - NVIDIA Jetson AI Lab). 이 토큰들은 미리 정한 256개의 특수 토큰(행동 토큰)으로, 각각 end-effector의 Δx, Δy, Δz, Δroll, Δpitch, Δyaw, gripper open/close 등을 표현합니다 (OpenVLA - NVIDIA Jetson AI Lab). 결국 OpenVLA는 **시각-언어 인식 + 정책(policy) 모델**이 합쳐진 형태로, 사용자가 "빨간 컵 집어"라고 지시하면, **중간에 컵을 어디인지 bounding box로 표시하거나 좌표를 계산하는 단계 없이**, 곧장 **그리퍼를 적절히 움직이는 제어명령 시퀀스**를 생성하는 것을 목표로 합니다 (OpenVLA - NVIDIA Jetson AI Lab) (OpenVLA - NVIDIA Jetson AI Lab). 이 모델은 거대한 LLM을 포함하므로 학습 및 추론이 무겁지만, NVIDIA는 INT4/FP8 양자화 및 MLC 가속 등을 통해 Jetson에서도 구동을 실험하고 있습니다 (OpenVLA - NVIDIA Jetson AI Lab). 실제로 OpenVLA-7B 가중치에 대해 **Jetson AGX Orin에서 LoRA 미세튜닝**이나 **시뮬레이터(MimicGen) 검증** 절차도 공개되어 있습니다 (OpenVLA - NVIDIA Jetson AI Lab) (OpenVLA - NVIDIA Jetson AI Lab). **성능:** OpenVLA는 아직 연구 단계라 정량적 FPS보다는, 주어진 시뮬레이션 환경에서 얼마나 성공적으로 물체 조작을 수행하는지가 주요 지표입니다. 7-DoF 로봇팔 기준 80여% 성공률 등의 보고가 있으며, 이는 동종 크기의 LLM 기반 정책 중 우수한 편이라고 합니다. 다만 실시간 FPS는 낮아 수 초에 한 번 명령을 내리는 정도일 수 있습니다. **ROS2 통합:** OpenVLA의 결과물(액션 토큰)을 ROS2 제어 명령으로 변환하는 것은 비교적 단순하므로, 이론적으로는 통합 가능하나, 현재 OpenVLA는 연구 프로토타입으로 별도 ROS2 패키지는 없습니다. **장점: 인지-계획-제어를 단일 거대 모델로 처리**한다는 혁신성이 있으며, 사람의 자연어 지시를 직접 로봇 행동으로 옮기는 궁극적 목표에 가깝습니다. **단점:** 거대한 모델(수십억 매개변수)을 필요로 하므로 임베디드 활용이 아직 초기 단계이고, 미세한 안전성 검증이나 예상치 못한 명령에 대한 대응 등 실용화를 위해 갈 길이 있습니다. 또한 중간에 사람이 결과를 확인/수정할 수 있는 여지가 없다는 것도 단점인데, 이는 반대로 말하면 **전과정 자동화**의 장점이기도 합니다.

(요약: 결합 접근법은 **텍스트 -> 검출 -> 세분화 -> 인식/행동**의 전체 흐름을 최적화하려는 방향입니다. Grounded-SAM과 같은 파이프라인은 현재 가장 실용적이며, NanoOWL+NanoSAM으로 ROS2 로봇에 적용 가능성이 높습니다. 한편 Semantic-SAM 같은 단일 거대모델이나 OpenVLA 같은 end-to-end 접근은 미래 지향적이며 잠재력이 크지만, **현 시점 Jetson에서 즉각 구현하기에는 무겁거나 복잡**하다는 한계가 있습니다.**)**

ROS 2 통합 요약

위에서 각 모델별로 ROS2 지원 여부를 언급하였지만, 주요 사항을 정리하면 다음과 같습니다:

- **NanoOWL (OWL-ViT)**: NVIDIA에서 **공식 ROS2 패키지** 제공 (GitHub - NVIDIA-AI-IOT/ROS2-NanoOWL: ROS 2 node for open-vocabulary object detection using NanoOWL.) – 오픈 vocab 검출 결과를 바로 ROS2 topic으로 받을 수 있음. NanoSAM과 조합은 별도 노드 작성 필요.
- **Grounding DINO**: 공식 ROS2 노드는 없으나, NVIDIA Jetson Platform Services로 REST API 형태 제공 (Grounding DINO (GDINO) — Jetson Platform Services documentation). 필요시 Python 기반 ROS2 노드 커스터마이징.
- **SAM 계열 (MobileSAM, NanoSAM 등)**: ROS2 패키지 없음. 다만 NanoSAM은 NanoOWL 결과와 함께 사용할 수 있도록 Python API 제공됨. ROS2에서 서비스로 mask 생성 기능을 노출하도록 구현 가능.
- **CLIPSeg/ZegCLIP 등**: 모두 연구 코드 수준으로 ROS2 통합은 사용자 몫. 이미지를 받아 mask와 좌표를 퍼블리시하는 간단한 구조이므로 비교적 구축 용이함.
- **Grounded-SAM 파이프라인**: ROS2 통합 시, 검출 노드 (텍스트 입력 -> ROI 출력)와 세그멘테이션 노드 (ROI -> 마스크 출력)를 연결하는 구조로 구성할 수 있음. 예: NanoOWL ROS2 노드 + 자체 SAM 노드.
- **OpenVLA**: 공개 코드가 PyTorch 기반으로 ROS와 무관하지만, 만약 실사용한다면 OpenVLA의 출력 (action 토큰)을 받아 ROS2의 trajectory 혹은 Twist 메시지로 변환하는 노드를 만들어야 함.

즉, **NanoOWL**을 제외하면 대부분 **직접 ROS2 래퍼를 구현**해야 하지만, **충분한 연산 성능과 최적화**를 확보하면 Jetson 상에서도 동작에 문제가 없습니다. ROS2 통합 시에는 **실시간성 보장을 위해 QoS 설정, 입력 지연 관리** 등이 중요하며, 특히 비디오 스트림에서 누락없이 프레임당 하나의 결과를 출력하도록 설계해야 합니다.

추천 알고리즘 및 조합 (TOP 3~6)

以上 여러 후보 중 **Jetson AGX Orin의 실시간 로보틱스 응용**에 특히 적합한 알고리즘들을 선정하면 다음과 같습니다.

1. **NanoOWL (OWL-ViT) + NanoSAM 조합** – **텍스트 기반 검출 + 마스크 분할**의 최적 조합 (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.). Jetson에서 **25~95 FPS**로 동작하는 NanoOWL이 관심 객체 위치를 찾아내고, 초고속 NanoSAM이 해당 객체의 **정확한 마스크**를 산출합니다. 이 방식은 사실상 **Grounded-SAM을 경량 모델로 구현**한 형태로서, 성능과 속도 균형이 뛰어납니다. ROS2 패키지로 NanoOWL이 이미 있고 통합이 수월하며, NanoSAM도 Python API를 통해 바로 사용 가능합니다. **추천 사유:** 실환경에서 다수 객체를 빠르게 인식하고 정확히 분할할 수 있어, **로봇 팔이 목표 물체를 오인없이 집도록 보조**합니다. 또한 모든 부품이 오픈소스이며 NVIDIA에서 최적화되어 **호환성, 유지보수성**도 우수합니다.

2. **Grounding DINO 1.5 Edge 단독 사용** – 최근 개선된 **경량 Grounding DINO** 모델을 활용하여, **텍스트 -> 2D Bounding Box**까지 한 단계로 수행하는 방안입니다. Grounding DINO 1.5 Edge는 Orin NX에서 10 FPS, AGX Orin에서는 추정상 **30 FPS 내외**로 동작 가능하며 (Too many objects, so little time: Open set object detection work by my... | Harry Shum), Zero-shot 검출 정확도가 가장 최신 SOTA 수준이라 **검출 신뢰성**이 높습니다. 마스크 분할이 꼭 필요하지 않은 경우(예: **3D 센서로 포인트클라우드에서 중심만 산출해도 되는 경우**), 이 방법이 파이프라인를 단순화하고 충분한 정확도를 제공합니다. **ROS2 통합**은 직접

구현해야 하지만, NVIDIA 제공 컨테이너 등을 참고하면 비교적 수월합니다. **추천 사유:** 모델 단독으로 **구현이 간결**하고, 다양한 표현의 자연어를 이해해 주는 능력이 강력합니다. Jetson에서 실시간에 근접한 성능을 낼 수 있는 **최신 Transformer 검출기**라는 점도 매력적입니다.

3. **CLIPSeg (텍스트 세그멘테이션) 단독 사용** – 장면 내 객체들이 뚜렷이 구분되고 겹치지 않는 상황이라면, **CLIPSeg처럼 경량 텍스트-세그멘테이션 모델** 하나만으로도 충분히 목적 달성이 가능합니다. 예를 들어 테이블 위에 **컵 하나만 있는 경우** "cup"이라는 질의에 대한 CLIPSeg 출력 마스크로 바로 3D 위치를 계산할 수 있습니다. 이 방법은 파이프라인이 가장 단순하며, CLIP 기반이라 **학습된 클래스 제한 없이** 임의의 물체도 인식 가능합니다. Jetson에서 ViT-B 기반 CLIPSeg는 적절히 최적화 시 **5~10 FPS**도 기대할 수 있어 실시간 요건에도 맞습니다. **추천 사유:** 시스템이 **간결하고 구현 난이도가 낮아**, 빠른 프로토타이핑에 유리합니다. 또한 추론 결과가 곧바로 픽셀 마스크라, 깊이 카메라와 결합하여 바로 **목표 물체의 3D 중심좌표**를 얻기 쉽습니다. 다만, 동일 클래스의 객체 여러 개가 있는 경우엔 한꺼번에 하나의 마스크로 묶어버리는 한계가 있으므로, 이 때는 검출+분할 조합으로 전환해야 합니다.

4. **MobileSAM / NanoSAM + Laser Pointer(**임의**)** – 만약 로봇이 **사용자 지시에 따라 특정 물체를 집는 상황**에서, 사용자가 레이저 포인터나 손가락 등으로 대략적인 위치를 가리킬 수 있다면, **SAM 계열 모델 단독으로도** 인식이 가능합니다. 예를 들어 **NanoSAM**에 **foreground point prompt**를 사용자 클릭 혹은 포인터 검출로 넣으면, 해당 지점 객체의 마스크를 얻을 수 있습니다. 이 방식은 텍스트 매칭 단계를 생략하지만, 사람의 직관을 입력받아 대상 식별을 합니다. Jetson에서 NanoSAM은 실시간이므로, 사용자가 UI나 레이저로 콕 집는 즉시 마스크와 좌표를 얻어 로봇팔을 제어할 수 있습니다. **추천 사유: 텍스트 인식의 애매함(동일 단어 여러 대상 등)을 사용자 입력으로 해소**할 수 있어 오인률이 0%에 가깝습니다. 또한 NanoSAM 마스크는 매우 정확하여, 바로 **그리퍼 접근 방향** 등을 계산하는데 활용할 수 있습니다. 단 이 접근은 사용자의 개입을 필요로 하므로 완전 자동화는 아니지만, **시스템 안정성과 제어 용이성** 측면에서 매력적입니다.

5. **Semantic-SAM (SEEM) 또는 SEEM-Tiny** – 향후를 대비한 전략으로, Microsoft의 **Semantic-SAM 모델**을 테스트해볼 가치가 있습니다. 비록 무겁지만, **한 번의 추론으로 이미지 내 모든 객체를 분할하고 분류(labeling)**할 수 있기 때문에, 환경이 비교적 고정된 경우 미리 한두 장면만 캡처해서 Semantic-SAM으로 처리하면 **어떤 물체들이 어디 있는지** 파악이 가능합니다 (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).). 이를 로봇에게 지식을 주입하거나(예: "table 위에 컵이 2개 있다"를 미리 앎), 또는 Semantic-SAM의 Tiny모델을 로컬에서 돌려 실시간 인식에 도전해볼 수 있습니다. **추천 사유: 종합적인 장면 이해**가 가능하므로, 단순 picking을 넘어 **멀티 객체 시나리오**("책상 위 책은 치우고 컵을 집어")까지 확장할 기반이 됩니다. Jetson 64GB면 Tiny 모델 정도는 로드 가능하므로, 최적화한다면 부분적으로 활용 가능할 것입니다.

각 추천 접근에는 그에 따른 이유와 한계가 있지만, **전반적으로 1번 NanoOWL+NanoSAM 조합**이 현 시점에서는 **가장 구현 난이도 대비 효과가 크고 안정적인 선택**으로 보입니다. 이 조합은 이미 **ROS2 환경과 상용 Jetson 보드에 최적화**되어 있기 때문에 개발 리스크가 낮습니다. 이후 성능 튜닝을 통해 Grounding DINO 1.5 등의 도입이나, 더 상위 개념인 OpenVLA 시도의 접목도 고려할 수 있을 것입니다.

결론 및 향후 전망

본 연구를 통해 Jetson AGX Orin에서 **자연어 지시 기반 객체 인식**을 실현하기 위한 다양한 VLM 알고리즘들을 검토하였습니다. **경량화된 open-vocabulary 검출과 세그멘테이션**의 조합이 현재로서는 가장 현실적 솔루션이며, 특히 NVIDIA의 NanoOWL, NanoSAM 등이 돋보였습니다. 이러한 기술을 활용하면, 사용자가 말로 지목한 물체를 로봇이 **실시간으로 인식하고 좌표를 얻어** 정확히 집을 수 있는 시스템을 구축할 수 있을 것으로 기대됩니다.

향후에는, **더 강력한 멀티모달 모델(SEEM, OpenVLA 등)**이 발전함에 따라 로봇이 장면을 **이해하고 추론**하는 수준까지 나아갈 것입니다. 예를 들어 "책상 위의 빨간 컵을 잡아서 싱크대로 가져다 놔"라는 복합 명령도, 로봇이 **맥락을 파악**하여 수행하게 될 것입니다. 그러한 단계로 가기 위해, 우선 본 보고서에서 논의된 기술들을 통합하고 ROS2 기반으로 **안정적인 perception 모듈**을 구현하는 것이 선결과제입니다.

요약하면, **Jetson AGX Orin + ROS2** 환경에서:

- **NanoOWL + NanoSAM**: 텍스트 지시 객체 검출 및 정밀 분할 (실시간).
- **GroundingDINO Edge**: 단순화된 high-performance 텍스트 검출 (실시간에 근접).
- **CLIPSeg 등**: 단일 물체 시나리오에 효과적인 경량 세그멘테이션. 이들을 적절히 활용하면 요구사항을 충족하는 시스템을 구축할 수 있습니다. 나아가 이러한 모듈을 조합함으로써 로봇의 **인지부터 조작까지** 전체 파이프라인의 지능을 향상시킬 수 있을 것입니다.

以上의 내용을 토대로, 구체적인 구현을 진행할 때에는 각 알고리즘의 **장단점과 시스템 통합 이슈**를 충분히 고려하시기 바라며, 필요시 추가 튜닝(예: TensorRT 최적화, Depth 처리 보정 등)을 병행하는 것을 권장합니다. 이번 조사를 통해 확보한 최신 지식을 바탕으로 성공적인 연구 개발이 이루어지기를 기대합니다.

참고 문헌 및 출처

- Meta AI, *"Segment Anything Model 2"* 소개 튜토리얼 (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium) (SAM 2: Advanced Object Segmentation for Images and Videos | by Abonia Sojasingarayar | Medium)
- NVIDIA, NanoOWL GitHub 리포지토리 및 ROS2 패키지 (GitHub - NVIDIA-AI-IOT/nanoowl: A project that optimizes OWL-ViT for real-time inference with NVIDIA TensorRT.) (GitHub - NVIDIA-AI-IOT/ROS2-NanoOWL: ROS 2 node for open-vocabulary object detection using NanoOWL.)
- NVIDIA, NanoSAM GitHub 및 성능 지표 (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT) (GitHub - NVIDIA-AI-IOT/nanosam: A distilled Segment Anything (SAM) model capable of running real-time with NVIDIA TensorRT)
- Microsoft Research, *"Semantic-SAM: Segment and Recognize Anything"* 논문 (Semantic-SAM: Segment and Recognize Anything at Any Granularity) (Semantic-SAM: Segment and Recognize Anything at Any Granularity)
- Fudan University, *"Semantic Segment Anything (SSA)"* GitHub (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).) (GitHub - fudan-zvg/Semantic-Segment-Anything: Automated dense category annotation engine that serves as the initial semantic labeling for the Segment Anything dataset (SA-1B).)
- OpenGVLab, Grounding DINO Jetson 성능 문서 (Grounding DINO (GDINO) — Jetson Platform Services documentation)
- Harry Shum (LinkedIn), *"Grounding DINO 1.5 Edge"* 발표 내용 (Too many objects, so little time: Open set object detection work by my... | Harry Shum)
- Timo Lüddecke et al., *"CLIPSeg: Image Segmentation Using Text and Image Prompts"* ([2112.10003] Image Segmentation Using Text and Image Prompts) ([2112.10003] Image Segmentation Using Text and Image Prompts)
- Zhou et al., *"ZegCLIP: Towards Adapting CLIP for Zero-shot Semantic Segmentation"* (CVPR 2023) (GitHub - jingyi0000/VLM_survey: Collection of AWESOME vision-language models for vision tasks) (Towards Adapting CLIP for Gaze Object Prediction - ResearchGate)

- Lin et al., *"CLIP is Also an Efficient Segmenter"* (CVPR 2023) (GitHub - linyq2117/CLIP-ES: [CVPR 2023] CLIP is Also an Efficient Segmenter: A Text-Driven Approach for Weakly Supervised Semantic Segmentation)
- 기타: NVIDIA Jetson AI Lab 자료 (OpenVLA 등) (OpenVLA - NVIDIA Jetson AI Lab) (OpenVLA - NVIDIA Jetson AI Lab), Grounded-SAM 프로젝트 (GitHub - IDEA-Research/Grounded-Segment-Anything: Grounded SAM: Marrying Grounding DINO with Segment Anything & Stable Diffusion & Recognize Anything - Automatically Detect , Segment and Generate Anything) 등.