아주 훌륭하게 정리해주셨습니다.

이제 Gen3 Lite 로봇과 책상 위 general 객체에 대해, ① Vision-Language Model 기반 객체 인식 및 MoveIt2를 활용한 grasping, ② 강화학습 기반 grasping 두 가지 방향 모두를 다루는 연구를 위한 심층 리서치를 시작하겠습니다.

영문 논문 제목 및 프로젝트 제목 후보 5개를 작성하고, 이를 기반으로 통합 분석하여 매우 전문적인 30페이지 분량의 한국어 리서치 보고서를 완성해드리겠습니다.

작업이 완료되면 바로 알려드리겠습니다!

1. 연구 주제 및 프로젝트 제목 후보 (영문 / 한국어 번역):

1. *Zero-Shot Vision-Language Robotic Grasping for Everyday Objects* (일상 물체에 대한 **제로샷 비전-언어 로봇 파지** 연구)
2. *Reinforcement Learning-Driven Grasping on Edge Robots* (엣지 로봇 플랫폼에서 **강화학습 기반 파지** 구현)
3. *Vision-Language vs. Reinforcement Learning for Robotic Manipulation: A Comparative Study* (로봇 조작을 위한 **비전-언어 대 강화학습 접근법 비교 연구**)
4. *Real-Time Language-Guided Grasping with VLMs on Jetson AGX Orin* (Jetson AGX Orin 상의 **실시간 VLM 기반 언어 지시 파지 시스템**)
5. *Integrating CLIP and Deep RL for Robust Robotic Grasping* (**CLIP과 심층 강화학습을 통합한 로봇 파지** 기법 연구)

비전-언어 모델과 강화학습을 활용한 로봇 파지: 시스템 설계 및 비교 연구 보고서

서론 (연구 배경, 필요성 및 목표)

현대 로봇공학에서 **로봇 팔의 물체 파지(grasping)**는 핵심 도전 과제 중 하나입니다. 특히 일상 환경의 **비구조화된 실내 공간**에서 사람처럼 다양한 물체를 집어드는 능력을 로봇에게 부여하려면, 로봇이 사전에 학습되지 않은 **새로운 물체**도 인식하고 파지할 수 있어야 하는 필요성이 대두됩니다. 전통적인 컴퓨터 비전 기반 파지 기법들은 대개 사전 정의된 물체나 특징에 의존하여 일반화에 한계가 있고, 강화학습이나 딥러닝 기반 파지 검출 또한 대량의 학습 데이터와 시간, 연산 자원이 필요합니다.

최근 등장한 **비전-언어 모델(VLM, Vision-Language Model)**들은 거대한 이미지-텍스트 데이터셋으로 학습되어 **제로샷(Zero-Shot)**으로 새로운 객체를 인식하거나 설명할 수 있는 능력을 보여주고 있습니다 (CLIP: Connecting text and images | OpenAI) (CLIP: Connecting text and images | OpenAI). 예를 들어 OpenAI의 CLIP 모델은 자연어 supervision만으로 다양한 시각 개념을 습득하여 별도 학습 없이도 새로운 분류 작업에 적용될 수 있음을 증명하였습니다 (CLIP: Connecting text and images | OpenAI). 이러한 **멀티모달 AI**의 발전으로 로봇은 사전에 본 적 없는 물체도 **텍스트로 지시**하면 인식하고 찾아낼 수 있는 가능성이 열렸습니다. 따라서 "빨간 컵을 집어줘"와 같은 **자연어 명령**을 이해하여 해당 물체를 찾아 파지하는 **언어-지시 기반 파지**가 연구의 새로운 방향으로 주목받고 있습니다.

동시에, **강화학습(Reinforcement Learning, RL)**을 이용한 로봇 파지 역시 활발한 연구 분야입니다. 심층 강화학습은 시각 입력을 통해 **폐루프(closed-loop)** 제어 정책을 학습하여 복잡한 파지 동작을 스스로 익힐 수 있고, 기존에 사람이 수동으로 설계하던 파지 동작이나 경로계획을 자동으로 학습하도록 합니다. 예를 들어, 구글의 QT-Opt는 58만 회에 달하는 실제 로봇 팔의 파지 시도를 통해 거대한 Q함수를 학습함으로써 **보지 못한 새로운 객체에 대해서도 96%의 높은 파지 성공률**을 달성하였으며, 재파지(regrasping)나 물체 위치 재조정과 같은 복잡한 동작도 자동으로 습득하였습니다 ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation) ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation). 이러한 **모델 프리(Model-free)** 강화학습 접근법은 인간이 정의하기 어려운 최적의 파지 전략을 로봇이 스스로 발견하게 해주지만, 방대한 학습 시도가 필요하고 학습된 정책이 새로운 환경 변화나 명시적으로 지정된 목표(예: 특정 물체 선택)에 쉽게 대응하지 못한다는 한계도 있습니다.

본 연구의 목표는 주어진 **Kinova Gen3 Lite 로봇팔**과 **Jetson AGX Orin** 컴퓨팅 플랫폼, **RGB-D 카메라** 기반의 책상 위 실내 환경을 대상으로, **비전-언어 모델 기반 파지 방법**과 **강화학습 기반 파지 방법** 두 가지를 **시스템 수준에서 설계 및 구현**하고 성능을 비교 분석하는 데 있습니다. 구체적으로, 사전 학습된 VLM을 활용하여 **레이블되지 않은 일반 물체**를 **자연어**로 지칭하며 파지하는 시스템과, 물체의 종류와 상관없이 **강화학습**으로 파지 동작을 익히는 시스템을 각각 개발합니다. 이를 통해 두 접근법의 **실시간 동작 성능**, **일반화 능력**, **구현 복잡도** 및 **장단점**을 면밀히 평가하고, 향후 **모바일 매니퓰레이터**로의 확장 등 응용 방향을 제안하는 것이 본 연구의 궁극적인 목표입니다.

요약하면 본 연구는 **Vision-Language 기반 파지 시스템**과 **강화학습 기반 파지 시스템**을 ROS2 Humble 환경에서 구현하여, Jetson Orin 상에서의 **실시간 동작 가능성**을 검증하고, 책상 위 다양한 물체들(단순 형태부터 복잡한 형태까지)에 대한 **파지 성공률**과 **효율**을 평가하는 **심층 연구**입니다. 다음 장들에서는 관련 최신 연구 동향과 기술을 살펴보고, 두 접근 방식의 시스템 구조와 알고리즘, 구현 세부사항을 제시한 뒤, 실험 방법과 예측되는 결과를 논의하도록 하겠습니다.

관련 연구 동향 (Vision-Language Models, 강화학습 기반 그래스핑, Jetson 최적화 기술)

본 장에서는 본 연구와 밀접한 관련이 있는 세 가지 기술 영역의 최신 동향을 살펴봅니다: **비전-언어 모델(VLM)**을 활용한 로봇 인지, **강화학습 기반 로봇 파지**, 그리고 **Jetson AGX Orin과 같은 엣지 플랫폼에서의 최적화 기법**입니다.

비전-언어 모델을 활용한 로봇 인지 및 파지

Vision-Language Model(VLM)은 이미지와 텍스트를 공동 학습하여 시각 장면에 대한 높은 수준의 이해를 가능케 하는 **기초 모델(foundation model)**입니다. 대표적으로 CLIP (Contrastive Language–Image Pre-training)은 웹상의 방대한 이미지-텍스트 쌍으로 학습되어, **텍스트 프롬프트만으로 이미지 내 객체를 식별**하거나 분류할 수 있는 강력한 **제로샷** 능력을 보여주었습니다 (CLIP: Connecting text and images | OpenAI). CLIP은 별도의 파인튜닝 없이도 원하는 클래스 이름을 텍스트로 주어 분류 작업을 수행하는 등 범용적인 시각 인식 능력을 보여주었으며, **자연어 지시만으로** 새로운 작업에 적용될 수 있음을 입증했습니다 (CLIP: Connecting text and images | OpenAI). 이러한 능력은 로봇에게도 유용하여, 사람이 일일이 물체를 학습시키지 않더라도 **"노란색 공 집어"**와 같은 명령으로 **새로운 물체**를 찾아낼 수 있는 가능성을 제공합니다. 최근 로봇 분야에서는 이러한 VLM을 파지에 활용하려는 시도가 늘고 있습니다. Shridhar 등 ([자연어 기반 로봇 파지 연구.pdf](file://file-Ueu1VAWN22a4YWEshfqYsx#:~:text=%EC%9E%90%EC%97%B0%EC%96%B4%20%EB%AA%85%EB%A0%B9%20%EA%B8%B0%EB%B0%98%20unlabeled%20%EA%B0%9D%EC%B2%B4,%ED%8F%AC%EC%A6%88%20%EC%B6%94%EC%A0%95%20%ED%9B%84%20%ED%8C%8C%EC%A7%80%20%EA%B3%84%ED%9A%8D))은 CLIP 모델을 로봇에 적용하여 **시각 정보와 언어 정보를 연결**하고, **로봇 팔 제어**와 결합하는 연구 방향을 제시하였습니다. 특히 **CLIPort** (CLIP + Transporter) 등의 시스템은 언어로 지목된 영역을 파지하도록 설계된 예로서, 언어와 시각 피쳐를 결합하여 **특정 객체나 위치**를 골라 조작하는 성능을 향상시켰습니다.

한편, **세그멘테이션** 분야에서도 강력한 기초 모델이 나와 파지에 활용되고 있습니다. Meta AI의 **Segment Anything Model (SAM)**은 쿼리 형태(점, 박스 등) 입력에 대해 이미지 내 **任意의 객체 마스크**를 고품질로 산출할 수 있는 모델로, 1억 개 이상의 마스크로 학습된 덕분에 거의 모든 분할 작업에 **제로샷** 성능을 보입니다 (GitHub - facebookresearch/segment-anything: The repository provides code for running inference with the SegmentAnything Model (SAM), links for downloading the trained model checkpoints, and example notebooks that show how to use the model.). SAM은 본래 범용 세그멘터로 제안되었으나, **Grounding DINO**와 결합한 **Grounded-SAM**과 같이 **텍스트 프롬프트로 목표 객체를 탐지 및 분할**하는 파이프라인도 등장했습니다. Grounding DINO는 CLIP의 이미지-텍스트 임베딩을 활용하여 **텍스트로 지시**

된 객체의 바운딩박스를 찾는 **오픈 vocabulary 객체 탐지** 모델로 ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=Grounding%20DINO%20DINO%20%EF%BF%BD%20CLIP,%ED%94%84%EB%A1%AC%ED%94%84%ED%8A%B8%20%EA%B8%B0%EB%B0%98%20%EB%B0%94%EC%9A%B4%EB%94%A9%20%EB%B0%95%EC%8A%A4%20%EC%83%9D%EC%84%B1)), SAM과 연계하면 **임의의 텍스트로 지칭한 객체 영역**을 분할해낼 수 있습니다. 이는 로봇이 "컵의 손잡이 부분"처럼 구체적인 부위를 인식하는 데에도 응용될 수 있어, **파지 접점 결정에 활용 가능한 시각정보**를 제공합니다. 실제로 **Affordance 기반 파지** 연구에서는 CLIP과 같은 VLM으로부터 얻은 객체의 의미 정보로 해당 물체의 유용한 파지 부위를 추론하기도 합니다. 예를 들어, CLIP으로 "컵의 손잡이"에 해당하는 픽셀들을 인식하고 이를 파지 지점으로 정하면, 단순히 물체의 기하학적 중심을 잡는 것보다 **더 안정적이고 기능적인 파지**가 가능합니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=Affordance%20%EA%B8%B0%EB%B0%98%20%ED%8C%8C%EC%A7%80%20%EA%B2%80%EC%B6%9C%EC%9D%80%20%EA%B0%9D%EC%B2%B4%EC%9D%98,%ED%8C%8C%EC%A7%80%20%EB%AA%A9%ED%91%9C%EB%A1%9C%20%EC%84%A4%EC%A0%95%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)). 이러한 접근은 **도구 사용**이나 **복잡한 형상의 물체**를 다룰 때 특히 유용하며, 인간이 물체를 사용하는 방식을 로봇 파지에 반영한다는 점에서 주목받고 있습니다.

비전-언어-액션(VLA) 모델의 부상도 눈여겨볼 만합니다. 이는 시각 및 언어 정보를 입력으로 받아 **로봇의 동작 시퀀스 출력**까지 통합한 모델로, 일종의 거대 멀티모달 모델을 로봇 제어에 적용하는 개념입니다. 2023년 공개된 **OpenVLA**는 7억개 이상의 파라미터를 가진 **7B 규모의 VLA 모델**로, 실제 로봇 시연 97만 에피소드로 학습되어 다양한 조작 작업을 일반화하도록 시도되었습니다 (OpenVLA: An Open-Source Vision-Language-Action Model - arXiv). OpenVLA와 같이 **대규모 시퀀스 모델**에 언어 지시와 시각 관찰, 로봇 액션을 함께 학습시키는 접근은, 향후 특정 작업에 한정되지 않는 **범용 로봇 비서** 개발의 토대가 되는 방향으로 각광받고 있습니다. 이러한 VLA 모델들은 아직 방대한 학습 데이터와 자원이라는 현실적 제약이 있지만, **Say-Can**(Google), **PALM-E** 등 대형 모델을 활용한 로봇 지능 연구 흐름과 맥을 같이합니다. 본 연구에서는 VLM을 주로 활용하지만, 향후 **VLA**로의 확장 가능성도念頭에 두고 있습니다.

마지막으로, 다양한 VLM 관련 최신 기법들을 살펴보면, **경량화 및 실시간화**가 중요한 트렌드입니다. 대형 모델을 로봇에 쓰려면 제한된 온보드 연산 자원에서 돌아가야 하므로, **효율성에 중점**을 둔 모델 변형이 이루어지고 있습니다 ([여러 모델 비교 분석 리서치.pdf](file://file-2bGr97xpkCw72nuHrBpfyN#:~:text=%EC%9D%B4%20%EB%B3%B4%EA%B3%A0%EC%84%9C%EB%8A%94%20SAM2%2C%20ZegCLIP%2C%20SCLIP%2C,%EB%98%90%ED%95%9C%20%EC%97%90%EC%A7%80%20%EB%B0%B0%ED%8F%AC%EB%A5%BC%20%EC%9C%84%ED%95%B4%20%EC%B5%9C%EC%A0%81%ED%99%94%EB%90%9C)). 예를 들어, **NanoOWL**과 **NanoSAM**은 각각 OWL-ViT 및 SAM을 NVIDIA Jetson급 장치에서 **실시간 추론 가능**하도록 최적화한 모델들로, ResNet18 기반 경량 아키텍처와 TensorRT 엔진 최적화를 통해 **Jetson AGX Orin에서 NanoSAM은 8.1ms 추론 (~123 FPS)**을 달성할 정도로 가볍습니다 ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-5biiWDZCXYfJSFGgtuHEvW#:~:text=%EC%8B%A4%EC%8B%9C%EA%B0%84%20%EC%84%B1%EB%8A%A5%20l%20Jetson%20AGX,%EB%86%92%EC%9D%80%20%ED%94%84%EB%A0%88%EC%9E%84%20%EB%A0%88%EC%9D%B4%ED%8A%B8%20%EC%B2%98%EB%A6%AC%20%EA%B0%80%EB%8A%A5)). NanoOWL은 자연어 프롬프트 기반 객체탐지 모델인 OWL-ViT를 경량화한 것으로, Jetson에서 OWL-ViT (ViT-B/32) 모델을 **약 95 FPS**로 구동 가능하며, 더 큰 ViT-B/16모델도 25 FPS로 구동할 수 있습니다 ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-5biiWDZCXYfJSFGgtuHEvW#:~:text=%EC%8B%A4%EC%8B%9C%EA%B0%84%20%EC%84%B1%EB%8A%A5%20l%20Jetson%20AGX,B%2F16%29%20%EA%B5%AC%EC%84%B1%EC%9C%BC%EB%A1%9C%2025fps%20%EC%84%B1%EB%8A%A5)). NanoOWL+NanoSAM을 조합하면 **제로샷 객체 탐지 및 세그멘테이션**을 동시에 실시간 수행할 수 있어, **Jetson Orin에 최적화된 파지 인식 파이프라인**으로 유망합니다 ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-5biiWDZCXYfJSFGgtuHEvW#:~:text=NanoOWL%20NanoOWL%EC%9D%80%20NVIDIA%EC%97%90%EC%84%9C%20%EA%B0%9C%EB%B0%9C%ED%95%9C%20OWL,%ED%86%B5%ED%95%9C%20%EA%B0%9D%EC%B2%B4%20%ED%83%90%EC%A7%80%EC%97%90%20%EC%B4%88%EC%A0%90%EC%9D%84%20%EB%A7%9E%EC%B6%A5%EB%8B%88%EB%8B%A4)) ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-5biiWDZCXYfJSFGgtuHEvW#:~:text=%EC%8B%A4%EC%8B%9C%EA%B0%84%20%EC%84%B1%EB%8A%A5%20l%20Jetson%20AGX,B%2F16%29%20%EA%B5%AC%EC%84%B1%EC%9C%BC%EB%A1%9C%2025fps%20%EC%84%B1%EB%8A%A5)). 그밖에도 MobileSAM, MobileVLM 등 모델 경량화 연구 ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=MobileVLM%20V2%20ViT%20%EF%BF%BD%20LLaMA,%EA%B2%BD%EB%9F%89%ED%99%94%EB%90%9C%20%ED%94%84%EB%A1%9C%EC%A0%9D%ED%84%B0%EB%A1%9C%20%ED%86%A0%ED%81%B0%20%EC%88%98%20%EA%B0%90%EC%86%8C))가 활발하며, 이러한 기술 동향을 참고하여 **엣지에서 운용 가능한 파지 시스템**을 설계하는 것이 중요합니다.

강화학습 기반 로봇 그래스핑

강화학습을 통한 로봇 파지 학습은 지난 수년간 많은 연구를 통해 발전해왔습니다. 초기에는 물체의 3D 모델 정보나 사전 정의된 그립 포인트를 활용했으나, 딥러닝과 결합된 RL은 **시각 관찰 → 로봇 제어 명령** 간의 매핑을 **엔드투엔드**로 학습함으로써, 사람이 규칙을 설계하지 않아도 로봇이 스스로 **파지 전략**을 터득하게 했습니다. 특히 **심층 Q-러닝** 기반의 접근이 주목받았는데, 앞서 언급한 QT-Opt는 딥마인드 및 구글 연구진에 의해 **실세계 데이터만으로** 학습된 사례입니다 ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation). QT-Opt는 시뮬레이션 없이 실제 로봇으로 58만 회의 시도를 거쳐 학습되었음에도, 미학습 객체에 96% 성공률을 달성하고, **폐루프 제어**로 파지 도중 물체가 밀리거나 실패할 경우 즉각 재시도하는 등 **적응적 행동**을 보여주었습니다 ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation). 이는 강화학습이 충분한 경험을 통해 전통적 방법보다도 **강인하고 유연한 파지 동작**을 얻을 수 있음을 시사합니다. 다만 현실에서 수십만 회의 시도를 실행하는 것은 비용이 매우 크기 때문에, **시뮬레이션**을 활용한 학습이 일반적입니다.

시뮬레이터 + 강화학습의 결합으로, 로봇 파지 학습의 규모를 키우고 속도를 높이는 연구들이 진행되어 왔습니다. 특히 NVIDIA의 **Isaac Gym**은 GPU 상에서 다수의 로봇 환경을 병렬로 시뮬레이션하고 동시에 RL 알고리즘을 학습시킬 수 있는 플랫폼으로, **물리 시뮬레이션과 신경망 학습을 모두 GPU 메모리에서 직접 처리**하여 기존 CPU기반 시뮬레이션 대비 **2~3차원(100~1000배) 속도 향상**을 달성했습니다 ([2108.10470] Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning). 이를 통해 복잡한 로봇 과제도 단일 GPU로 수 시간~수일 내에 정책 학습이 가능해졌습니다. 예를 들어 Isaac Gym을 활용하면 수천 개의 병렬 환경에서 로봇팔 파지 시도를 동시에 수행하여, 실시간에 가까운 속도로 학습 데이터를 모을 수 있습니다. Makoviychuk 등 ([2108.10470] Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning)은 이러한 **대량병렬 RL**을 통해 고난도 로봇 제어 과제를 단분간에 풀어내는 사례들을 보고하였습니다.

강화학습 알고리즘 측면에서는 **정책 경사 방법(Policy Gradient)**과 **오프-폴리시 Off-policy** 알고리즘들이 주로 활용됩니다. **Proximal Policy Optimization (PPO)**은 비교적 안정적으로 학습되는 on-policy 방법으로 로봇 제어에 자주 쓰이며, OpenAI 등이 물체 조작이나 로봇다리 제어 등에 활용한 바 있습니다. **Soft Actor-Critic (SAC)**은 오프폴리시 기법으로 연속 제어 문제에서 **표본 효율이 높고** 안정적이라 평가받으며 (Control strategy of robotic manipulator based on multi-task ...) ([PDF] arXiv:2309.04687v1 [cs.RO] 9 Sep 2023), 특히 로봇 팔 등의 정확한 제어에 적합합니다. 실제 연구에서는 PPO로 기본 정책을 학습한 뒤 SAC로 미세조정하거나, 혹은 Hyperparameter 튜닝을 통해 둘 중 하나의 알고리즘을 선택하는 방식이 흔합니다 (Optimizing Deep Reinforcement Learning for Adaptive Robotic Arm ...) (Robotic Manipulator in Dynamic Environment with SAC Combing ...). 본 연구에서도 **연속 동작 공간**에서 효과적인 알고리즘으로 SAC와 PPO를 후보로 고려하며, 시뮬레이션 상에서 학습 안정성과 수렴 속도를 비교하여 최적 알고리즘을 선택할 것입니다. 또한 파지 성공 시 높은 보상, 물체를 놓치거나 충돌 시 패널티 등을 부여하는 **보상 함수 설계**, 학습 초기의 탐색을 돕기 위한 **행동 노이즈** 추가, 그리고 **도메인 랜덤화**를 통한 **Sim-to-Real** 간극 축소 전략 등도 최신 연구에서 강조되는 기법들입니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=%EC%82%AC%EA%B0%81%ED%98%95%EC%9D%98%20%EC%A4%91%EC%8B%AC%EC%A0%90%EC%9D%84%20%ED%8C%8C%EC%A7%80%20%EB%AA%A9%ED%91%9C%20%EC%9C%84%EC%B9%98%EB%A1%9C,%EC%A0%91%EA%B7%BC%20%EB%B0%A9%ED%96%A5%EC%9D%84%20%EA%

B2%B0%EC%A0%95%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)). 이러한 요소들을 종합하여, 강화학습으로 학습된 파지 정책이 실제 로봇에서도 원활히 동작하도록 할 계획입니다.

Jetson AGX Orin 및 엣지 AI 최적화 기술

Jetson AGX Orin은 NVIDIA의 첨단 엣지 AI 플랫폼으로, 200 TOPS 이상의 연산 성능과 16코어 CPU, 64GB 메모리 등을 갖추고 있어 로봇 내장용 컴퓨터로 각광받고 있습니다. 그러나 여전히 대형 딥러닝 모델을 실시간 구동하기에는 제약이 있으므로, **모델 최적화 및 경량화** 기술이 필수적입니다. 본 연구에서는 Jetson 상에서 **5 FPS 이상**의 추론 속도를 목표로 하고 있는데 ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=%EC%A0%9C%EB%A1%9C%EC%83%B7%20%EA%B0%9D%EC%B2%B4%20%EC%9D%B8%EC%8B%9D%3A%20%EC%82%AC%EC%A0%84%20%ED%95%99%EC%8A%B5%EB%90%98%EC%A7%80,%EC%95%8A%EC%9D%80%20%EA%B0%9D%EC%B2%B4%EB%8F%84%20%ED%85%8D%EC%8A%A4%ED%8A%B8%20%ED%94%84%EB%A1%AC%ED%94%84%ED%8A%B8%EB%A1%9C%20%EC%9D%B8%EC%8B%9D)) ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=%EC%8B%A4%EC%8B%9C%EA%B0%84%20%EC%B2%98%EB%A6%AC%3A%205FPS%20%EC%9D%B4%EC%83%81%EC%9D%98%20%EC%8B%A0%EC%86%8D%ED%95%9C,%EC%B6%94%EB%A1%A0%20%EC%86%8D%EB%8F%84)), 이를 위해 고려하는 최적화 기법들은 다음과 같습니다.

- **TensorRT 최적화**: NVIDIA에서 제공하는 딥러닝 추론 최적화 라이브러리인 TensorRT를 사용하여 모델을 FP16 또는 INT8로 quantization하고, Jetson의 GPU 및 NVDLA 가속기에서 효율적으로 돌아가도록 엔진화합니다. 예를 들어 앞서 언급한 NanoOWL, NanoSAM도 TensorRT를 통해 최적화되었습니다 ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-5biiWDZCXYfJSFGgtuHEvW#:~:text=%EC%9E%A5%EB%8B%A8%EC%A0%90%20l%20%EC%9E%A5%EC%A0%90%3A%20Jetson%20%ED%94%8C%EB%9E%AB%ED%8F%BC%EC%97%90,TensorRT%20%EC%A7%80%EC%9B%90%2C%20%EB%8B%A4%EC%96%91%ED%95%9C%20%EC%82%AC%EC%9A%A9%20%EC%82%AC%EB%A1%80)).
- **경량 모델 사용**: 가능하면 MobileNet, EfficientNet 등 파라미터 수가 적은 backbone을 활용한 모델을 채택합니다. 본 연구에서는 SAM 대신 MobileSAMv2 (경량화 SAM), 거대 언어모델 대신 LLaMA 기반 MobileVLM 등 **Mobile 모델**들을 고려하며 ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=MobileVLM%20V2%20ViT%20%EF%BF%BD%20LLaMA,%EA%B2%BD%EB%9F%89%ED%99%94%EB%90%9C%20%ED%94%84%EB%A1%9C%EC%A0%9D%ED%84%B0%EB%A1%9C%20%ED%86%A0%ED%81%B0%20%EC%88%98%20%EA%B0%90%EC%86%8C)), 실험적으로도 NanoSAM 등은 성능 대비 현저히 가벼워 실시간성에 유리함을 확인했습니다.
- **모델 파이프라인 최적화**: 여러 딥러닝 모듈을 연속 사용할 경우, **GPU 메모리 복사 최소화** 및 **병렬화**가 중요합니다. 예를 들어 객체 탐지 후 곧바로 세그멘테이션을 할 때, **한 프로세스 내에서 연계**하여 실행하거나 TensorRT에서 two-stage pipeline을 하나의 engine으로 합치는 방안을 모색합니다. Grounded-SAM의 경우 Grounding DINO와 SAM을 별개로 돌리지만, **SAM2** 등의 최신 연구에서는 **단일 트랜스포머**로 객체 탐지+분할을 통합하여 중간 과정을 줄이는 시도를 하고 있습니다 ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=%EB%8B%A8%EC%9D%BC%20%ED%8C%8C%EC%9D%B4%ED%94%84%EB%9D%BC%EC%9D%B8%EC%9C%BC%EB%A1%9C%20%ED%86%B5%ED%95%A9%20%EA%B0%9D%EC%B2%B4%20%EC%9D%B8%EC%8B%9D%2F)) ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=Jetson%20AGX%20Orin%EC%97%90%20%EC%B5%9C%EC%A0%81%ED%99%94%EB%90%9C%20NanoOWL%EC%9D%B4,NanoSAM%EC%9D%B4%20%EC%84%B8%EA%B7%B8%EB%A9%98%ED%85%8C%EC%9D%B4%EC%85%98%EC%9D%84%20%EB%8B%B4%20%EB%8B%B9%ED%95%98%EB%8A%94%20%ED%8C%8C%EC%9D%B4%ED%94%84%EB%9D%BC%EC%9D%B8)). 우리 시스템에서도 가능하면 단계를 간소화하여 지연(latency)을 줄일 것입니다.
- **ROS2 실시간 설정**: ROS2에서 실시간 성능을 높이기 위한 설정 (예: realtime 커널, 쓰레드 우선순위 조정, DDS QoS 설정)과 **노드 구성 최적화**도 적용합니다. 영상 처리는 별도 노드에서 GPU를 활용하고, MoveIt2 등의 경로계획은 CPU 스레드에서 병렬로 수행하여 서로 간섭을 최소화합니다. 또한 Jetson에서 **NvJPEG**, **VPI** 같은 하드웨어 가속 라이브러리를 사용해 카메라 영상 전처리를 최적화합니다.

이러한 기술들을 종합하여 Jetson Orin 상에서 **인지-계획-제어 전체 파이프라인**이 원활히 동작하도록 하는 것이 목표이며, 이는 엣지 로봇의 **자율성**을 높이는 중요한 요소입니다. NanoOWL의 ROS2 패키지 활용이나 ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-5biiWDZCXYfJSFGgtuHEvW#:~:text=%EC%B6%94%EC%B2%9C%20%EC%9D%B4%EC%9C%A0%3A%20l%20Jetson%20AGX,95fps%2C%20NanoSAM)), Grounded-SAM의 오픈소스 ROS2 통합 등을 적극 활용하고 필요 시 커스텀 노드를 개발하여, **ROS2 Humble** 기반 시스템에 최적화된 구조를 구현할 것입니다.

시스템 설계 (VLM 기반 파지 시스템 vs. 강화학습 기반 파지 시스템)

본 연구에서는 두 가지 상이한 접근의 파지 시스템을 각각 설계합니다. 하나는 **Vision-Language Model(VLM) 기반 파지 시스템**으로, 사전 학습된 AI 모델을 이용해 **물체 인식** 및 **파지 지점 결정**을 수행하고, **MoveIt2**를 활용해 로봇팔을 제어합니다. 다른 하나는 **강화학습(RL) 기반 파지 시스템**으로, **시뮬레이션**을 통해 로봇팔의 파지 정책을 학습시키고 이를 실로봇에 적용하는 구조입니다. 두 시스템 모두 하드웨어적으로는 Kinova Gen3 Lite 로봇팔과 RGB-D 카메라, Jetson AGX Orin을 사용하며, ROS2 Humble를 미들웨어로 합니다. 아래에서는 각 시스템의 아키텍처와 데이터 흐름을 설명합니다.

VLM 기반 파지 시스템 아키텍처

비전-언어 모델 기반 시스템의 전체 흐름은 **"인지 → 계획 → 실행"**의 단계로 구성됩니다. **그림 1**은 이 시스템의 개략적인 구성도입니다 (텍스트로 서술):

1) 카메라 센싱 및 전처리: RGB-D 카메라 (예: Realsense D435 또는 Azure Kinect)를 통해 **환경의 컬러 영상과 깊이 정보**를 획득합니다. 컬러 영상은 VLM 기반 인지 모듈로 입력되고, 깊이 정보는 탐지된 객체의 3D 위치 산출에 사용됩니다. 카메라로부터 받은 원시 이미지에 대해 왜곡 보정, 색상 보정 등의 전처리를 ROS2 노드에서 수행합니다.

2) 비전-언어 기반 객체 인식: 사전에 학습된 **오픈 vocabulary 객체 탐지 및 세분화** 모델을 사용하여, **관심 물체**의 픽셀 좌표와 마스크를 구합니다. 구체적으로, 사용자가 명령한 대상 객체의 **텍스트 프롬프트**를 VLM에 입력으로 주어, 해당 텍스트에 부합하는 물체를 이미지에서 찾습니다. 예를 들어 명령이 "빨간 컵을 잡아"라면, 텍스트 프롬프트 "red cup"을 사용하여 이미지 내 빨간 컵의 위치를 찾습니다. 모델 구성은 앞서 논의한 여러 후보 중 **NanoOWL + NanoSAM** 조합을 택할 예정입니다. NanoOWL이 **"red cup"**이라는 텍스트를 입력 받아 이미지에서 **해당 객체의 bounding box**를 예측하고 ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=Grounding%20DINO%20DINO%20%EF%BF%BD%20CLIP,%ED%94%84%EB%A1%AC%ED%94%84%ED%8A%B8%20%EA%B8%B0%EB%B0%98%20%EB%B0%94%EC%9A%B4%EB%94%A9%20%EB%B0%95%EC%8A%A4%20%EC%83%9D%EC%84%B1)), 이어서 NanoSAM이 그 박스를 입력받아 정밀한 **객체 마스크**를 생성합니다. 이때 NanoOWL/NanoSAM 모델은 모두 Jetson 상에서 TensorRT로 최적화되어 **실시간 추론**이 가능합니다 (NanoOWL ~95FPS, NanoSAM ~30FPS 성능) ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=%EC%8B%A4%EC%8B%9C%EA%B0%84%20%EC%84%B1%EB%8A%A5%3A%20NanoOWL%EC%9D%80%2095FPS%2C%20NanoSAM%EC%9D%80,30FPS%20%EC%9D%B4%EC%83%81%20%EB%8B%AC%EC%84%B1)) ([Genspark - VLM 알고리즘 비교 분석.pdf](file://file-

5biiWDZCXYfJSFGgtuHEvW#:~:text=%EC%8B%A4%EC%8B%9C%EA%B0%84%20%EC%84%B1%EB%8A%A5%20I%20Jetson%20AGX,B%2F16%29%20%EA%B5%AC%EC%84%B1%EC%9C%BC%EB%A1%9C%2025fps%20%EC%84%B1%EB%8A%A5)). 만약 대상 객체가 다수 존재하거나 여러 후보가 인식되면, 가장 confidence가 높은 것을 선택하거나, 추가 조건 (예: "가장 큰 빨간 컵")이 있다면 이미지 후처리를 통해 조건에 맞는 객체를 고릅니다. 결과적으로 이 단계에서는 **타겟 객체의 픽셀 좌표 영역 (ROI)**과 **픽셀 마스크**가 산출됩니다.

3) 3D 좌표 계산 및 파지 지점 결정: 얻어진 객체의 2D 위치 및 마스크를 **카메라 좌표계의 3차원 좌표**로 변환합니다. 깊이 영상을 이용하여 ROI 영역의 평균 깊이나 마스크의 centroid 깊이를 추출하면 물체의 대략적인 3D 위치 (x, y, z)를 얻을 수 있습니다. 추가로, 마스크로부터 물체의 **orientation**을 추정하거나, **affordance** 정보가 있다면 (예: 컵의 손잡이 부위), 해당 부분의 좌표를 파지 목표로 선택합니다. 간단한 경우, 물체의 중심 상단을 향해 그리퍼를 내려잡는 **top-grasp** 전략을 취할 수 있고, 보다 복잡한 형태의 경우 별도의 **파지 포인트 산출 알고리즘**을 적용할 수 있습니다. 예컨대 **Grasp Pose Detection** 알고리즘이나 사전에 학습된 **Grasping Rectangle** 모델을 사용하여 해당 객체에서 최적 그립 지점을 찾고, 그 결과를 이용할 수 있습니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=Grasping%20Rectangle%20%ED%91%9C%ED%98%84%20%EB%B0%A9%EC%8B%9D%EC%9D%80%20RGBD,%EC%A0%91%EA%B7%BC%20%EB%B0%A9%ED%96%A5%EC%9D%84%20%EA%B2%B0%EC%A0%95%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)) ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=Deep%20Learning%20%EA%B8%B0%EB%B0%98%20%ED%8C%8C%EC%A7%80%20%EA%B2%80%EC%B6%9C,%EB%AA%A9%ED%91%9C%20%EC%9C%84%EC%B9%98%EB%A1%9C%20%EC%84%A4%EC%A0%95%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)). 그러나 본 연구의 기본 시나리오는 일반적인 책상 위 물체이므로, 마스크 중심에 대한 top-down 파지를 1차 접근으로 활용하고, 필요 시 보조적인 개선을 추가할 것입니다.

4) MoveIt2 기반 모션 계획: 결정된 파지 지점(목표 3D 좌표 및 그립 방향)은 **ROS2**를 통해 **MoveIt2**에 전달되어 로봇팔의 모션 계획에 사용됩니다. MoveIt2는 ROS2의 표준 모션 플래닝 프레임워크로, 로봇의 역운동학, 경로계획, 충돌 검사 등을 담당합니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=MoveIt%202%EB%8A%94%20%EB%A1%9C%EB%B4%87%20%ED%8C%94%EC%9D%98%20%EB%AA%A8%EC%85%98,%EA%B3%84%ED%9A%8D%EC%9D%84%20%EC%88%98%EB%A6%BD%ED%95%98%EB%8A%94%20%EB%8D%B0%20%ED%95%84%EC%88%98%EC%A0%81%EC%9D%B8%20%EB%8F%84%EA%B5%AC%EC%9D%B4%EB%8B%A4)). VLM 모듈로부터 나온 파지 목표를 MoveIt2로 입력하는 방법은 여러 가지가 있으나, 가장 표준적인 방식은 **moveit_msgs/Grasp** 메시지를 활용하는 것입니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=%ED%8C%8C%EC%A7%80%20%EC%A7%80%EC%A0%90%20%EC%A0%95%EB%B3%B4%EB%A5%BC%20MoveIt%202%EC%97%90,%EB%8F%99%EC%9E%91%20%EA%B3%84%ED%9A%8D%EC%9D%84%20%EC%88%98%EB%A6%BD%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)). 이 메시지에는 파지 전 그리퍼 자세, 파지 시 그리퍼 자세, 접근 방향, 파지할 목표 pose 등이 포함되며, 앞 단계에서 계산한 **파지 목표 위치/방향**를 grasp_pose 필드에 설정하고 그리퍼의 여닫음 상태를 pre_grasp_posture (열림) 및 grasp_posture (닫힘)에 명시하여 MoveIt2로 보냅니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=%ED%8C%8C%EC%A7%80%20%EC%A7%80%EC%A0%90%20%EC%A0%95%EB%B3%B4%EB%A5%BC%20MoveIt%202%EC%97%90,%EB%8F%99%EC%9E%91%20%EA%B3%84%ED%9A%8D%EC%9D%84%20%EC%88%98%EB%A6%BD%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)). MoveIt2는 이를 받아 **접근 경로**와 **파지 후 후퇴 경로**를 함께 계획하며, 이 과정에서 **Planning Scene**에 대상 객체를 **Collision Object**로 추가하여 로봇이 물체를 잡을 때 물체와의 충돌(실은 파지이지만, 로봇 모델 상 충돌로 간주됨)을 허용하거나 적절히 처리합니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=%EC%9D%B8%EC%8B%9D%EB%90%9C%20%EA%B0%9D%EC%B2%B4%EB%A5%BC%20MoveIt%202%EC%9D%98%20Planning,%EC%95%88%EC%A0%84%ED%95%9C%20%ED%8C%8C%EC%A7%80%20%EC%9E%91%EC%97%85%EC%9D%84%20%EC%9C%84%ED%95%B4%20%ED%95%84%EC%88%98%EC%A0%81%EC%9D%B4%EB%8B%A4)). Planning Scene 갱신을 위해 객체의 3D bounding box를 Collision Object로 설정해두면, MoveIt이 경로를 짤 때 그 객체를 인지하게 할 수 있습니다. 이러한 설정으로 **안전하고 충돌없는 경로**를 찾아 그리퍼를 목표 지점까지 이동시키도록 계획됩니다.

5) 로봇 제어 실행: 계획된 경로가 도출되면, ROS2의 FollowJointTrajectory 액션 등을 통해 Kinova Gen3 Lite 로봇팔을 제어하여 그리퍼를 움직입니다. Kinova Gen3 Lite는 6자유도 경량 매니퓰레이터로, ROS2용 드라이버 (Kinova Gen3 ROS2 패키지)를 통해 제어 명령을 받을 수 있습니다. MoveIt2가 산출한 trajectory를 **Joint Position or Velocity Command** 형태로 로봇에게 전송하여 순차적으로 관절들을 구동합니다. 목표 지점에 도달하면 그리퍼를 **폐합**하여 물체를 파지하고, 후퇴 경로를 따라 들어올립니다. 이러한 일련의 실행 과정 동안, 실시간으로 혹시 모를 오차에 대비해 카메라 피드백을 모니터링하거나 (본 연구 기본 시나리오는 정적인 환경으로 간주하여 폐루프로 제어하지는 않지만), 필요한 경우 중간에 경로 재계획을 할 수 있도록 구성합니다.

6) 완료 및 후속처리: 물체를 쥐어 성공적으로 들어올렸다면, 목표 높이까지 리프트한 후 동작을 종료합니다. 만약 파지에 실패했다면 (예: 물체를 놓쳤거나 미끄러짐), VLM 인지 모듈의 결과를 갱신하고 재시도하는 루프를 수행할 수도 있습니다. 예를 들어 한 번 실패 시 그리퍼 자세를 미세 조정하여 재시도하거나, 다른 파지 지점을 선택하는 휴리스틱을 둘 수 있습니다.

이 전체 과정에서 중요한 것은 **인지 모듈 (VLM)**과 **플래닝/제어 모듈 (MoveIt2)** 간의 원활한 통신과 **좌표계 정합**입니다. 카메라의 좌표로 얻은 3D 좌표를 로봇 베이스 좌표계로 변환해야 하며, 이를 위해 카메라-로봇 간 **캘리브레이션**이 선행되어야 합니다. 또한 ROS2 상에서 영상 인지 노드와 MoveIt2 노드가 **동기화**되어야 하므로, 메시지 타임스탬프와 TF 변환 관리가 필요합니다. 이러한 시스템 통합 이슈를 모두 해결함으로써, VLM 기반 로봇 파지 시스템이 **실시간으로 언어 지시에 따라 물체를 인지하고 집어낼 수 있도록** 하는 것이 설계의 핵심입니다.

강화학습 기반 파지 시스템 아키텍처

강화학습 기반 파지 시스템은 **훈련 단계**와 **실행 단계**로 크게 나누어 볼 수 있습니다. **훈련 단계**에서는 물체 파지 과제를 가상 환경에서 시뮬레이션하고, RL 알고리즘으로 로봇의 파지 정책을 학습시킵니다. **실행 단계**에서는 학습된 정책을 실 로봇에 이식하여 파지 작업을 수행합니다. 각 부분을 자세히 설명하면 다음과 같습니다.

1) 시뮬레이션 환경 구축 (Isaac Gym): 현실의 책상 위 물체 파지 시나리오를 모사하기 위해, NVIDIA Isaac Gym (또는 최근의 Isaac Sim 강화학습 환경)을 사용하여 **가상 환경**을 설정합니다. Kinova Gen3 Lite 로봇팔 모델과 여러 가지 가상의 물체 (박스, 공, 컵 등)를 시뮬레이터에 배치합니다. 물체들은 무작위 크기, 모양, 위치로 생성하여 **다양한 파지 상황**을 구성합니다. Isaac Gym의 GPU 가속을 활용하여 수백~수천 개의 병렬 환경을 동시에 실행함으로써, 로봇 정책 학습에 필요한 **다양한 경험을 병렬 수집**합니다 ([2108.10470] Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning). 이러한 병렬 환경에서는 각기 다른 초기 물체 배치, 마찰 계수, 조명 조건 등을 무작위화(domain randomization)하여, 학습된 정책이 한정된 조건에 과적합되지 않고 **일반적인 파지 능력**을 갖추도록 합니다.

2) RL 문제 설정: 파지 과제를 강화학습 문제로 정의합니다. 에이전트(agent)는 로봇팔이며, 환경 상태(state)는 로봇 관절각도, 관절속도, 그리고 **시각 관찰**로 구성됩니다. 시각 관찰은 카메라 영상을 그대로 신경망에 입력으로 줄 수도 있고, 혹은 **PointCloud**나 **Depth image** 형태로 줄 수도 있습니다. 선택지로, 전면 카메라 RGB 이미지와 뎁스 이미지를 신경망에 쌓아 입력으로 활용하거나, 시뮬레이터 내 물체의 위치 정보를 직접 사용하는 것도 고려할 수 있습니다 (후자는 학습 난이도를 크게 낮추지만, Sim-to-Real을 위해서는 최종적으로 카메라 인풋을 사용하는 것이 바람직합니다). 행동(action)은 로봇팔의 관절 속도 또는 목표 위치 명령입니다. 6개 관절에 대해 연속적인 값 ($-\Delta\theta \sim +\Delta\theta$ 범위)으로 움직이거나, 또는 **엔드이펙터(end-effector)의 3D 이동 + 그리퍼 개폐** 명령을 직접 정의할 수도 있습니다. 본 연구에서는 직관적으로 학습되도록, **엔드이펙터**의 Δx, Δy, Δz 이동 및 Δroll, Δpitch, Δyaw (혹은 quaternion) 변화를 action으로 설정하고, 별도로 그리퍼 개폐 (이산 2값: 열림/닫힘)를 추가로 다루는 방식을 고려합니다. 이렇게 하면 학습이 로봇팔의 운동학을 직접 찾아가기보다, 손목 위치를 어떻게 움직일지에 초점을 맞춰 **탐색 공간**을 줄일 수 있습니다.

3) 보상 함수 설계: 강화학습의 성능은 **보상 함수**에 크게 좌우되므로, 파지 과제에 적합한 보상 설계를 합니다. 기본적으로 물체를 성공적으로 집었을 때 **높은 양의 보상**을 부여합니다. 구체적으로, 에피소드 내에서 그리퍼가 물체를 들어올렸을 때 +1의 보상을 주고 에피소드를 종료시킬 수 있습니다. 또한 학습을 촉진하기 위해 **부분 보상 shaping**을 활용합니다: 예를 들어, 그리퍼 끝과 목표 물체 사이 거리가 줄어들수록 작은 양의 보상을 주거나, 그리퍼가 닫혔을 때 물체에 어느 정도 접촉했으면 보상을 줘

서 **파지 시도를 장려**합니다. 반대로, 로봇이 일정 높이 이상 내려갔는데도 물체를 잡지 못했거나, 물체를 밀어내 책상에서 떨어뜨렸다면 **페널티**를 부여합니다. 충돌(로봇 팔이 책상이나 주변과 부딪힘) 역시 -1의 큰 페널티와 함께 에피소드를 종료시켜, 안전한 경로만 탐색하도록 유도합니다. 이러한 shaping은 학습 초기 난제를 완화하는데 도움을 주지만, 잘못하면 정책이 애매하게 수렴할 위험도 있으므로 적절한 상수를 정하는 데 유의합니다.

4) 신경망 정책 및 알고리즘: 로봇의 정책을 근사할 신경망 구조를 정합니다. 일반적으로 **컨벌루션 신경망(CNN)**이 시각 입력을 처리하고, 물체와 로봇 상태를 결합한 feature를 **다층 퍼셉트론(MLP)**으로 처리하여 action을 출력합니다. 예를 들어 RGB-D 이미지를 입력으로 받는 ResNet 기반 CNN과, 로봇 관절 상태 (혹은 엔드이펙터 상대 위치)를 입력으로 받는 작은 MLP의 출력을 concat하여 다시 MLP로 처리, 최종 6DOF delta pose와 그리퍼 상태를 예측하도록 할 수 있습니다. 강화학습 알고리즘으로는 앞서 언급한 **SAC**를 1차 선택합니다. SAC는 off-policy 방법이므로 **리플레이 버퍼**를 사용해 과거 경험을 샘플링하며 학습할 수 있어 **시뮬레이션 병렬화**와도 잘 맞습니다. 또한 SAC의 entropy 보상으로 **탐색을 유지**하면서 **안정적 수렴**이 가능하리라 기대합니다. 비교를 위해 **PPO** 알고리즘도 병행 실험하여 성능을 측정합니다. PPO는 on-policy이지만, Isaac Gym같이 대량 병렬 환경에서는 배치 데이터를 크게 모아 업데이트함으로써 효율을 낼 수 있습니다. 두 알고리즘의 파라미터 (학습률, entropy 계수 등)는 기본값으로 시작해, **학습 곡선**을 관찰하며 튜닝합니다.

5) 학습 과정 모니터링: 수십만 step에 걸친 학습 동안, **에피소드 당 성공률**, **평균 에피소드 보상**, **그립 강도** 등의 지표를 로깅하고 모니터링합니다. 목표는 시뮬레이터 상에서 성공률 90% 이상을 달성하는 정책을 얻는 것입니다. 만약 학습이 정체되면 보상 함수를 수정하거나, 신경망 구조를 변경하거나, 객체 초기 조건을 바꾸는 등 개입을 합니다. 또한 학습된 정책이 특정 유형의 물체에 치우쳐 성능을 내지 않는지, 예컨대 둥근 공만 잘 집고 얇은 판형 물체는 못 집는 것은 아닌지 등을 확인합니다. 그런 경우 추가적인 트레이닝 (特定 물체 비중을 늘림 등)을 수행합니다.

6) 학습 정책의 실세계 적용 (Sim-to-Real): 시뮬레이션에서 얻어진 최종 정책은 로봇에 이식됩니다. 이때 바로 쓰기보다, **도메인 랜덤화로 인해** 시뮬레이터와 현실 간 차이를 줄였다고 하더라도, 일부 갭이 존재할 수 있음을 감안합니다. 예를 들어 시뮬레이터의 카메라 영상과 실제 카메라 영상의 노이즈/감마 차이나, 물체 물리 특성 차이 등이 있습니다. 이를 보완하기 위해 **fine-tuning** 단계를 둘 수 있습니다: 로봇 실제 환경에서 몇 차례 시도를 해보고 그 데이터를 가지고 policy를 미세 조정하거나, Human-in-the-loop으로 약간의 보정을 가할 수 있습니다. 하지만 가능하면 추가 학습 없이도 동작하도록, 처음부터 **현실감 있는 시뮬레이션**과 **충분한 랜덤화**를 적용했을 것입니다.

실행 단계에서, 학습된 policy는 기본적으로 **관찰 → 행동**을 매핑하는 함수이므로, ROS2 노드로 이식을 합니다. 카메라로부터 이미지를 받아 전처리 후 신경망에 넣고, 출력으로 로봇 제어 명령을 실시간 생성합니다. 이때 정책이 **초당 수십 Hz**로 출력할 수 있으므로, 제어 주기는 10~30Hz 수준으로 설정합니다. RL 정책은 *피드백 제어* 형태로 동작하므로, 실행 도중에도 카메라를 계속 참조하며 물체가 예상대로 따라오지 않으면 그에 맞게 경로를 수정해갑니다. 이는 앞서 VLM+MoveIt 방식이 **한 번 경로 계획 후 실행**하는 *개방루프*(open-loop)인 것과 대비되는 점입니다. RL 정책은 매 시점 센서 피드백에 반응하므로, 파지 시 물체가 움직이거나 살짝 위치가 달라져도 추적하여 대응할 가능성이 높습니다.

7) 안전 및 종료 조건: 실 로봇에서 정책을 실행할 때도 안전장치를 둡니다. 예컨대 정책이 잘못되었을 경우 이상한 방향으로 관절을 움직일 수 있는데, 이를 위해 **소프트 리미트**나 **가상 울타리**를 설정하고, 카메라로 모니터링하여 물체를 지나치게 밀거나 떨어뜨리면 사람 개입으로 정지할 수 있도록 합니다. 또한 학습된 정책이 성공/실패 여부를 스스로 인지하지 못하므로, **후처리**로 그리퍼의 힘이나 물체의 이동 여부를 체크하여 성공 시 동작을 멈추고 실패 시 재시도하게끔 상위 레벨에서 제어합니다 (상위 레벨 제어는 간단히 rule-based로 구현할 수 있습니다).

정리하면, 강화학습 기반 시스템은 **시뮬레이션 학습기**와 **실시간 제어기**의 조합으로 구성됩니다. 이는 VLM 기반 시스템에 비해 학습에 시간이 들지만, **로봇 제어의 저수준 세부까지 자동으로 최적화**된 정책을 제공하며, 특히 **센서-액츄에이터 사이클을 빠르게 폐루프로** 돌릴 수 있다는 장점이 있습니다. 반면 특정 물체를 골라 잡는 등의 **고수준 인지 능력**은 부족하므로, 필요 시 VLM의 인지 모듈과 RL 정책을 결합하여 "이 물체를 잡아"라는 명령에 대응하는 **하이브리드 시스템**도 고려할 수 있을 것입니다.

하드웨어 및 소프트웨어 구성 (Gen3 Lite, ROS2, MoveIt2, Jetson AGX Orin, RGB-D 카메라)

이 절에서는 본 연구에 사용되는 하드웨어와 소프트웨어 스택의 구성을 정리합니다.

- **로봇 플랫폼: Kinova Gen3 Lite** – Kinova사의 Gen3 Lite는 6-자유도 경량 로봇팔로, 최대 가반하중 약 2kg, 작동 범위 可及半径 ~ 590 mm를 갖습니다. 비교적 컴팩트하며, 각 관절에 torque sensor가 있어 impedance 제어도 가능하지만, 본 연구에서는 주로 위치 제어를 사용합니다. 이 로봇팔은 **Kinova Kortex API**로 제어할 수 있고 ROS2 드라이버 패키지가 제공되어, ROS2 액션/토픽 인터페이스를 통해 **관절 제어** 및 **그리퍼 개폐** 명령을 줄 수 있습니다. 그리퍼는 2-finger 형태로 병진 개폐하며, 물체를 잡기에 적당한 힘 제어가 가능합니다.

- **컴퓨팅: NVIDIA Jetson AGX Orin** – 로봇 제어 및 AI 연산을 담당하는 온보드 컴퓨터로, 2022년 출시된 Jetson AGX Orin 모듈을 사용합니다. Orin은 12-core Arm Cortex CPU와 2048-core Ampere GPU (Orin 32GB 모델 기준), 그리고 2개의 NVDLA(딥러닝 가속기)를 포함하고 있어, 병렬 연산에 강점이 있습니다. 또한 10Gbps CSI 카메라 입력 등을 지원하여 RGB-D 카메라를 직접 연결할 수 있습니다. 본 연구에서는 Jetson Orin에 **Ubuntu 22.04 LTS**와 **ROS2 Humble Hawksbill**을 설치하고, CUDA, TensorRT, cuDNN 등 GPU 가속 라이브러리를 세팅합니다. Jetson은 저전력으로 동작하기 때문에 발열관리에 주의하며, **MAXN 모드**로 성능을 최대로 활용할 예정입니다.

- **RGB-D 카메라: Intel RealSense D435 (예시)** – 실험 환경에서 물체 인지를 위한 깊이 카메라로 Intel RealSense D400 시리즈를 사용합니다. 이 카메라는 1280x720 색상 영상과 정밀한 깊이 영상을 실시간 제공하며, ROS2용 드라이버 (realsense2_camera 패키지)가 있어 쉽게 토픽을 통해 영상을 획득할 수 있습니다. 깊이 센싱 범위는 0.2m ~ 10m 정도로 책상 위 물체 (~0.5m 거리)에는 충분한 정밀도를 제공하며 깊이 오차는 수 mm 수준입니다. 카메라는 로봇팔이 내려다보는 각도로 고정하거나, 로봇 베이스 혹은 외부 삼각대에 장착하여 책상 상부를 시야에 담도록 배치합니다. 카메라 좌표와 로봇 좌표의 정합(calibration)은 ROS2 tf 시스템에 statically 세팅하고, 필요 시 Hand-eye 캘리브레이션 절차를 통해 개선합니다.

- **ROS2 Humble 및 주요 패키지:** ROS2는 분산 시스템으로 각 컴포넌트를 노드로 실행하고 DDS를 통해 통신하므로, 우리 시스템의 **인지 노드**, **모션계획 노드**, **로봇드라이버 노드** 등이 동시에 동작합니다. 주요 패키지로, MoveIt2 (moveit_ros 플레임워크), realsense2_camera, rviz2 (시각화) 등을 사용합니다. 또한 앞서 언급한 NanoOWL, NanoSAM 등의 AI 모델을 돌리기 위해 **ros2_nanollm** 프로젝트의 일부를 활용할 계획입니다 (ROS2 Nodes for Generative AI - NVIDIA Jetson AI Lab). ros2_nanollm은 NVIDIA에서 제공하는 ROS2용 LLM/VLM 패키지로, NanoOWL의 ROS2 wrapper (`ros2_nanoowl`)와 같은 노드를 포함하고 있어 VLM 추론 결과를 ROS 토픽으로 받을 수 있습니다 (NVIDIA-AI-IOT repositories - GitHub). 만약 해당 패키지가 성숙하지 않다면, Python으로 ONNX 혹은 TensorRT 엔진을 불러오는 자체 노드를 작성할 것입니다. 강화학습 쪽으로는, Isaac Gym은 Python 환경에서 학습 단계에 사용되고, 실제 로봇 적용 시에는 학습된 모델을 불러오는 ROS2 노드를 작성해야 합니다. 이 부분은 PyTorch로 정책을 저장하고, ROS2에서 C++보다는 Python 노드로 불러와 실행할 가능성이 높습니다 (Jetson에서 Python GPU 실행도 가능하지만, GIL로 인해 멀티스레딩에 유의).

- **MoveIt2 & Planning**: MoveIt2는 별도 구성 파일을 통해 Kinova Gen3 Lite의 URDF 모델과 Kinematics, Joint limits 등을 세팅하고 사용합니다. 또한 grasp 동작을 위해 MoveIt2의 **Grasp Execution Pipeline**을 활용하는데, 이는 detect된 객체를 가상으로 Planning Scene에 삽입하고, Grasp 메시지를 통해 plan을 생성한 후, 실행(Execution)까지 해주는 파이프라인입니다. MoveIt2는 기본적으로 OMPL (Open Motion Planning Library)의 RRT, PRM 알고리즘을 사용해 경로를 탐색하며, 필요시 제약조건 설정 (예: 엔드이펙터 평행하게 유지) 등을 적용할 수 있습니다.

- **기타 센서/구성:** 기본 시나리오에는 카메라 한 대와 로봇팔만 있지만, 향후 확장을 고려해 로봇 베이스(모바일 플랫폼)나 추가 센서(IMU 등)가 연동될 수 있습니다. 소프트웨어적으로는 이를 위해 Nav2 (자율주행 이동)나 SLAM 툴박스 등의 ROS2 패키지도 염두에 둘 수 있습니다. 그러나 본 연구의 범위에서는 주로 고정된 로봇팔+카메라 환경에 집중합니다.

以上 하드웨어/소프트웨어 구성 요소들을 통합하여, 실험을 위한 시스템을 구성합니다. 설치된 환경에서 각 노드들의 주기를 최적화하고, 데이터 흐름을 원활히 함으로써 **실시간 처리**와 **정확한 제어**가 모두 만족되도록 하는 것이 핵심입니다.

알고리즘 선택 및 비교 분석 (VLM 모델 vs. RL 알고리즘)

이 절에서는 앞서 설계한 두 접근 방법에 사용될 **핵심 알고리즘들을 선정**하고, 그 특징을 비교합니다. 구체적으로 VLM 기반 시스템에 적합한 **시각-언어 모델 조합**과 RL 기반 시스템에 적합한 **강화학습 알고리즘**을 결정합니다. 또한 두 접근의 장단점을 종합 비교합니다.

VLM 모델 선정 및 구성

비전-언어 기반 파지에서 요구되는 기능은 (a) **텍스트 질의에 따른 객체 인식**과 (b) **해당 객체의 정확한 위치/형상 파악**입니다. 이를 만족하기 위해 **객체 탐지 모델**과 **세그멘테이션 모델**의 조합이 필요하며, 추가로 **언어 임베딩**을 처리할 수 있어야 합니다. 후보 기술로 1) CLIP 기반 방법, 2) OWL-ViT 기반 방법, 3) Grounding DINO 기반 방법 등을 검토했습니다. 각 접근의 장단점을 표로 정리하면 다음과 같습니다:

접근 방법 (모델 조합)	기술 특징
NanoOWL + NanoSAM *(OWL-ViT + SAM 경량화 버전)*	제로샷 텍스트 기반 객체탐지 + 고속 세그멘테이션 (CLIP계열 ViT 백본) ([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=Grounding%20DINO%20DINO%20%EF%BF%BD%20CLIP,%ED%94%84%EB%A1%AC%ED%94%84%ED%8A%B8%20([실시간 비전-언어 모델 기반 객체 인식 및 그래스핑 시스템 설계 레포트.pdf](file://file-5XdocsrELM1XboiJw6drtR#:~:text=MobileSAMv2%20%EA%B2%BD%EB%9F%89%ED%99%94%20SAM%20%EA%B2%BD%EB%9F%89%ED%99%94%EB%
Grounding DINO + MobileSAMv2	텍스트 프롬프트 기반 바운딩박스 예측 + 경량 세그멘테이션 (ViT-H 기반)
MobileVLM v2 + MobileSAMv2 *(경량 멀티모달 + SAM)*	ViT + 경량 LLM 조합으로 복잡한 언어 명령 처리 가능 + 세그멘테이션
Grounded SAM (1단계 파이프라인)	Grounding DINO + SAM 통합 (joint 모델) – **오픈소스 구현 이용**

위 표에서 보듯이, **NanoOWL + NanoSAM 조합**이 Jetson Orin 환경에서 가장 **실시간 처리**에 유리하고, ROS2 통합도 수월하므로 우선적 선택입니다. 이 조합은 OWL-ViT 기반이어서 CLIP의 강력한 언어-시각 임베딩을 활용하며, SAM 기반 분할로 픽셀 정확도를 확보합니다. Grounding DINO + MobileSAM도 정확도 면에서 매력적이지만, NanoOWL 대비 속도가 떨어질 우려와 구현 복잡도가 있습니다. Grounded-SAM 등의 통합 모델은 아직 연구 프로토타입 수준으로 성능 검증이 충분치 않아, 부차적으로 고려합니다.

따라서 **최종적으로 VLM 기반 인지 모듈은 NanoOWL (텍스트 기반 탐지)과 NanoSAM (세그멘테이션)**으로 구성하고, 필요시 NanoOWL 대신 Grounding DINO를 fallback으로 사용할 수 있도록 할 것입니다.

또한 **CLIP 자체를 활용한 접근**도 간단히 비교하면, CLIP 모델로 이미지의 영역 특징을 임베딩하고 텍스트 임베딩과 비교하여 해당 물체를 찾는 **Zeroshot Detection** 연구들이 있습니다 (예: RegionCLIP, ZegCLIP 등 ([여러 모델 비교 분석 리서치.pdf](file://file-2bGr97xpkCw72nuHrBpfyN#:~:text=%EC%9D%B4%20%EB%B3%B4%EA%B3%A0%EC%84%9C%EB%8A%94%20SAM2%2C%20ZegCLIP%2C%20SCLIP%2C,%EB%98%90%ED%95%9C%20%EC%97%90%EC%A7%80%20%EB%B0%B0%ED%8F%AC%EB%A5%BC%20%EC%9C%84%ED%95%B4%20%EC%B5%9C%EC%A0%81%ED%99%94%EB%90%9C)). 그러나 이런 접근은 2-stage (지역 제안 + 임베딩 비교)로 속도가 느릴 수 있고, 구현 복잡성이 높아, 완성도 있는 오픈소스인 OWL-ViT나 Grounding DINO쪽을 선택했습니다.

강화학습 알고리즘 및 설정 비교

강화학습 부분에서는 **SAC vs. PPO** 두 알고리즘을 중점적으로 고려하고 있습니다. 이 둘은 로봇 제어에 많이 사용되는 대표 알고리즘으로, 각기 장단점이 있습니다:

- **SAC (Soft Actor-Critic)**: Off-policy 최대엔트로피 RL 알고리즘입니다. 경험을 효율적으로 재사용하여 **표본 효율(sample efficiency)**이 높고, Q함수 학습을 병행해 **안정적**인 성능을 보이는 경우가 많습니다. 또한 자동 온도 조절로 탐험-이용 균형을 자동으로 맞춰줍니다. 단, Off-policy이므로 hyperparameter에 민감할 수 있고, 정책과 가치함수 신경망 학습이 복잡하여 tuning이 어렵다는 지적도 있습니다. 로봇 연속제어 문제에서 SAC는 DDPG, TD3 등을 제치고 좋은 성과를 내는 것으로 보고되었습니다 (Control strategy of robotic manipulator based on multi-task ...) ([PDF] arXiv:2309.04687v1 [cs.RO] 9 Sep 2023).

- **PPO (Proximal Policy Optimization)**: On-policy 정책경사 기법으로, trust-region 개념을 접목하여 update의 안정성을 높인 알고리즘입니다. 샘플 효율은 SAC보다 낮을 수 있지만, 단순한 구조와 적은 hyperparameter로 **현장에서 많이 활용**됩니다. 병렬환경에서 대량의 샘플을 모아 학습할 경우 충분한 성능을 낼 수 있고, 특히 이산 동작이나 제한된 스킴에서는 SAC 못지않게 성과가 좋습니다. 다만 exploration 측면에서 entropy 보상을 인위적으로 조절해야 할 수 있고, sparse reward 문제에서 학습이 느릴 수 있습니다.

본 연구의 파지 과제는 연속 제어 + sparse reward 특성이 있으므로, **SAC**가 유리할 것으로 예상합니다. 시뮬레이션 속도도 매우 빠르기 때문에 sample 효율이 다소 낮아도 PPO도 활용 가능하지만, 정책의 품질을 높이려면 SAC를 기본으로 선택합니다.

또 다른 고려사항으로, **학습 파이프라인**을 결정해야 합니다. Isaac Gym에서는 병렬 환경으로 on-policy PPO를 학습하는 예제가 풍부하지만, off-policy 알고리즘 적용도 가능합니다 (AI4Finance-Foundation/ElegantRL: Massively Parallel Deep ...). Off-policy 학습 시 하나의 GPU에서 4096 환경을 돌리면서 경험을 쌓고 학습하면, 1억 step 정도까지도 몇 시간~하루 내로 가능하다고 보고됩니다 ([PDF] PARALLEL Q-LEARNING: SCALING OFF-POLICY RE - OpenReview). 이 정도면 충분히 로봇 파지 정책을 학습시킬

수 있을 것으로 보입니다. 또한 **학습 안정화 기법**으로 reward normalization, observation normalization 등을 적용하여, 학습 초반 수렴을 돕습니다 (Robotic Manipulator in Dynamic Environment with SAC Combing ...).

비교군으로 **전통적 파지 기법**과도 장단점을 분석해볼 수 있습니다. 딥러닝 기반 **Grasp Detection** (예: GG-CNN 등) 방법은 한 장의 RGB-D 영상에서 바로 그립 위치를 회귀하여 제시하는 방식인데, 이는 **지도학습**으로 특정 데이터 분포에 최적화되며, 우리 연구의 VLM이나 RL 방식과는 접근이 다릅니다. Grasp Detection은 현재도 산업 로봇에 널리 쓰이며 구현이 비교적 간단하지만, **새로운 목표(특정 물체 선택)** 같은 요구에는 약합니다. 반면 VLM 방식은 **목표 지정 유연성**이 높고, RL 방식은 **상황대응 및 최적화된 동작** 측면에서 강점이 있습니다.

아울러, VLM과 RL 접근의 **내재된 장단점**을 요약 비교하면 다음과 같습니다:

- **일반화 측면:** VLM 기반은 CLIP 등의 사전학습 덕분에 **미학습 객체**도 인식 가능하고, 언어로 지칭만 하면 잡을 수 있는 **범용성**이 있습니다. 반면 RL 기반은 학습한 물체들(혹은 그 형태 유사체)에 대해서만 성공률이 높으며, 새로운 유형의 물체나 전혀 다른 모양에는 학습을 다시 해야 할 수 있습니다.
- **실시간 성능:** RL 정책의 추론은 단순 신경망 forward이므로 매우 빠르고 (수 ms 이하), 센서 수신부터 제어 출력까지 짧은 주기로 돌릴 수 있습니다. VLM 기반은 인지 단계에 복잡한 모델이 있어 수십 ms~수백 ms 이상 걸릴 수 있고, 또 MoveIt 경로계획도 수백 ms 정도 소요될 수 있어, **즉각 대응성**은 RL보다 떨어집니다.
- **정확도와 성공률:** 이는 구현에 따라 다르지만, VLM기반은 인지 정확도가 높다면 적절한 파지 지점을 선택할 가능성이 높습니다. 다만 파지 동작 자체는 MoveIt의 **기하학적** 계획에 의존하므로, 미세한 조정이나 실패 시 재시도 등의 **적응적 스킬**은 부족할 수 있습니다. RL 기반은 학습된 정책이 때때로 **예상 밖의 전략** (예: 물체를 굴려서 잡기)을 구사할 수 있고, 실패 시 바로 다른 접근을 시도하는 유연성이 있습니다 ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation). 그러나 학습이 충분치 않으면 엉뚱한 곳을 잡으려 하거나 불안정하게 잡을 위험도 있습니다.
- **개발 난이도:** VLM 방식은 강력한 사전 학습 모델들을 사용하지만, 그것들을 **로봇 시스템에 통합**하는 공학적 작업이 필요합니다. 반면 RL 방식은 학습 파이프라인 구축과 시뮬레이터 설계에 큰 노력이 들고, 특히 보상 튜닝 등이 **시도 과정**을 많이 요구합니다. 둘 다 난이도 측면의 도전이 다르나, RL 쪽이 **계측/평가/반복**이 오래 걸린다는 점에서 연구 개발 일정에 영향을 줄 수 있습니다.

이러한 점을 고려하여, 본 연구에서는 **VLM 기반 방법**과 **RL 기반 방법** 모두를 구현/실험함으로써, 각 접근의 실제 성능을 체계적으로 비교 평가할 계획입니다. 이는 해당 영역 연구에 기여하는 바도 있으며, 궁극적으로 두 접근의 **장점을 결합**하는 방향 (예: VLM으로 목표 인식 + RL 정책으로 제어)도 모색해볼 수 있을 것입니다.

VLM 기반 Grasping 방법론 (MoveIt2 연동 중심)

이 장에서는 Vision-Language Model을 활용한 파지 시스템의 **구체적인 방법론**을 상세히 설명합니다. 앞서 시스템 설계장에서 전체적인 파이프라인을 개관하였으므로, 여기서는 특히 **MoveIt2와의 연동 방식**과 **핵심 알고리즘/모듈의 동작**을 단계별로 명확히 기술하고자 합니다.

객체 인식 및 파지 후보 결정

VLM 기반 grasping의 성능은 첫 단계인 **객체 인식 결과**에 크게 좌우됩니다. 따라서 신뢰도 높고 정밀한 객체 인식을 위해 **다단계 인지 알고리즘**을 적용합니다.

1. **텍스트 프롬프트 입력:** 사용자로부터 또는 상위 명령으로부터 **파지할 대상**을 나타내는 문장이 주어집니다. 예) "잡지책을 집어 들어" 또는 "빨간색 공을 집어." 여기서 주요 키워드를 추출하여 **텍스트 프롬프트**로 사용합니다. 위 예에서 "잡지책", "빨간색 공" 등이 프롬프트가 됩니다. 만약 문장이 복잡하여 추가 단어 (예: "오른쪽에 있는") 등이 포함되면, 이를 파싱하여 조건으로 활용할 수도 있으나, 1차 버전에서는 단순 객체명 위주로 처리합니다.

2. **오픈 어휘 객체 탐지 (NanoOWL):** 추출된 텍스트 프롬프트를 NanoOWL 모듈에 입력합니다. NanoOWL은 CLIP과 동일한 ViT 백본을 이용해 이미지 전체를 **한 번에 처리**하면서, 주어진 텍스트에 해당하는 **바운딩 박스**를 출력합니다. 이때 Confidence score도 함께 나오며, 만약 top-1 결과의 신뢰도가 임계치보다 낮다면 "대상 객체 없음"으로 처리할 수 있습니다. NanoOWL이 여러 객체를 탐지할 경우 (예: 같은 카테고리 여러 개), top-N 박스를 출력받습니다.

3. **세그멘테이션 (NanoSAM):** 탐지된 ROI에 대하여 NanoSAM을 적용합니다. NanoSAM (혹은 MobileSAMv2)은 바운딩 박스 또는 가운데 포인트를 프롬프트로 받아 해당 영역의 **픽셀 정밀 마스크**를 생성합니다. 마스크 획득으로 객체의 **정확한 윤곽과 픽셀 단위 위치**를 파악할 수 있습니다. 만약 NanoOWL에서 여러 후보를 받은 경우, 각 후보별로 NanoSAM을 수행하여 모든 마스크를 얻습니다.

4. **3D 포인트 산출:** 각 마스크에 대해, 카메라의 뎁스 데이터를 조회하여 마스크 영역의 **3차원 좌표들** (point cloud)를 얻습니다. 이를 통해 물체의 **중심 좌표 (x,y,z)**를 계산하고, 또한 **기본 법선 방향**을 추정합니다 (평균 노멀 또는 주성분분석으로 방향 결정). 기본적으로 물체의 표면 노멀 방향이 위쪽을 향하면 top-grasp가 용이하나, 책과 같이 평평이 놓인 경우 옆면을 잡아야 할 수도 있습니다. 이때 물체의 기울기 등을 고려해 잡기 좋은 방향을 heuristics로 정합니다. 예를 들어 공이나 컵같이 위가 열려있지 않은 물체는 윗면 어느 방향이나 비슷하므로 top-down, 책이나 마우스같이 납작한 물체는 옆면을 잡도록 파지 방향을 결정합니다.

5. **후보 파지 지점 추출:** 마스크와 3D 정보로부터 **후보 파지 지점**을 정합니다. 가장 단순히는 물체의 **겉 중심부**를 그리퍼 중앙이 가도록 하는 접근입니다. 그리퍼의 너비는 물체 크기에 따라 조절해야 하므로, 마스크의 bounding box 크기를 픽셀단위로 잰 뒤 카메라 깊이를 활용하여 실제 물체 폭을 추정합니다. 이 폭보다 약간 큰 수준으로 그리퍼를 벌려서 접근하게 하면 기본적인 파지가 됩니다. 그러나 모든 물체를 이렇게만 하면 놓칠 수 있으므로, **특정 형태별 세부 규칙**을 추가합니다:

   - 긴 막대형 물체(예: 펜)인 경우 중앙보다 한쪽 끝을 잡는 것이 안정적이므로, 마스크를 긴 방향으로 2등분하여 1/4, 3/4 지점을 후보로 고려합니다.
   - 컵이나 주전자처럼 **손잡이**가 있는 물체는, CLIP 등으로 손잡이 부분을 인식해 해당 부위를 파지점으로 삼습니다. (예: "컵 손잡이"를 텍스트로 주어 NanoOWL→NanoSAM 수행하면 손잡이 마스크를 얻을 수도 있습니다).
   - 책이나 얇은 물체는 위에서 집기 어려우므로 측면을 파지점으로 선택하며, 이때 로봇 End-effector pose도 옆에서 접근하는 자세를 취하도록 설정합니다.
   - 공처럼 둥근 물체는 어느 방향이나 유사하므로, 그냥 top-down으로 중심을 잡되, 미끄럼 방지를 위해 최대한 중앙을 관통하도록 alignment합니다.

   이러한 규칙 기반 보완은 추후 ML기반 Grasp Planner로 대체할 수 있으나, 현 단계에서는 안전빵으로 적용합니다.

6. **최종 파지 대상 선정:** 만약 인식 결과 다수의 객체 (예: 동일한 컵 2개)에서 후보 파지 지점이 나왔다면, **사용자 명령**이나 **상황 맥락**을 고려하여 하나를 선택합니다. 예를 들어 "왼쪽 빨간 컵 잡아" 같은 명령에서는 이미지 좌우를 분석해 더 왼쪽에 있는 것을 선택합니다. 맥락이 모호하면 임의로 하나를 고르되, 사용자에게 Confirm 절차(해당 안하지만 미래 시스템에선 필요)를 둘 수도 있습니다.

이상으로, 하나의 대상 객체와 그에 대한 **구체적인 파지 지점 (3D 좌표 + 그립 자세)**이 결정되었습니다.

MoveIt2를 통한 파지 동작 계획 및 실행

MoveIt2 연동은 앞서도 다뤘듯이, **Grasp 메시지**와 **Planning Scene**을 사용하는 것이 핵심입니다. 이를 구현 단계별로 기술합니다.

1. **Planning Scene 설정:** ROS2에서 MoveIt2를 구동하면 기본적으로 로봇 URDF 기반 장면이 생성됩니다. 여기에, 인식된 객체를 Collision Object로 추가합니다. 구체적으로, moveit_msgs::CollisionObject 메시지에 객체의 primitive shape (예: box)나 mesh를 넣어 게시하면 MoveIt2가 장면에 반영합니다. 우리 경우 정확한 물체 모델이 없으므로, **bounding box 크기의 직육면체**로 근사 추가합니다. 이것은 로봇팔이 파지 시 자기손(그리퍼)와 물체의 충돌을 무시하도록 설정하기 위해서인데,

MoveIt2의 grasp plan 시 CollisionObject와 그리퍼 링크 간 충돌은 grasp 시 허용으로 설정할 수 있습니다. 한편, CollisionObject를 추가해두면 파지 후 물체를 들었을 때 MoveIt이 그 물체와 주변 환경 충돌도 고려하게 되어, 더 안전한 후퇴 경로를 생성합니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=%EC%9D%B8%EC%8B%9D%EB%90%9C%20%EA%B0%9D%EC%B2%B4%EB%A5%BC%20MoveIt%202%EC%9D%98%20Planning,%EC%95%88%EC%A0%84%ED%95%9C%20%ED%8C%8C%EC%A7%80%20%EC%9E%91%EC%97%85%EC%9D%84%20%EC%9C%84%ED%95%B4%20%ED%95%84%EC%88%98%EC%A0%81%EC%9D%B4%EB%8B%A4)).

2. **Grasp 메시지 구성:** moveit_msgs::Grasp 타입의 메시지를 채웁니다. 필수 필드로:
   - `grasp_pose`: 로봇이 실제 파지할 목표 pose (geometry_msgs::PoseStamped). 앞서 결정한 3D 파지 지점과 접근 orientation을 여기에 넣습니다. 좌표는 **세계 좌표계** 기준으로 변환한 값이어야 합니다 (카메라 TF → base_link TF 변환 적용).
   - `pre_grasp_approach` 및 `post_grasp_retreat`: 접근 및 후퇴 경로를 지정합니다. 보통 접근 방향은 grasp_pose를 향해 **앞에서 10cm 거리** 정도 떨어진 지점을 시작점으로 하고, 후퇴는 반대로 위쪽으로 10-15cm 이동 등으로 지정합니다. 이 길이와 방향은 상황에 맞게 조절하되, 기본적으로 수직 위 방향(z+)으로 후퇴하도록 설정합니다.
   - `pre_grasp_posture`: 파지 전 그리퍼 모양, 즉 벌어진 상태의 JointState. Kinova 그리퍼 관절 각도를 최대 벌림 각도로 설정.
   - `grasp_posture`: 파지 시 그리퍼 모양, 즉 닫힌 상태 JointState. 물체를 잡았을 때의 힘 조절이 필요하지만 MoveIt 수준에서는 각도만 지정 (예: 완전 닫힘에 가깝게).
   - `grasp_quality`: 이 grasp의 점수 (0~1). 하나만 줄 경우 1로 설정.

   이러한 Grasp 메시지를 하나 원소로 가진 vector를 MoveIt2의 pick() 함수에 넘기면, MoveIt2가 내부적으로 위 정보를 활용하여 trajectory를 계산합니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=%ED%8C%8C%EC%A7%80%20%EC%A7%80%EC%A0%90%20%EC%A0%95%EB%B3%B4%EB%A5%BC%20MoveIt%202%EC%97%90,%EB%8F%99%EC%9E%91%20%EA%B3%84%ED%9A%8D%EC%9D%84%20%EC%88%98%EB%A6%BD%ED%95%A0%20%EC%88%98%20%EC%9E%88%EB%8B%A4)). MoveIt2는 자체적으로 IK를 풀어 grasp_pose를 도달 가능한지 확인하고, 여러 관절해 중 가장 적절한 것을 택합니다.

3. **Trajectory Planning:** MoveIt2가 제공하는 Planner (OMPL)로 Grasp trajectory를 생성합니다. Plan 시 고려사항:
   - pre-grasp에서 grasp_pose까지 직선 경로 (approach)와, grasp_pose 이후 post_grasp까지 경로 (retreat)을 우선적으로 맞추려 시도.
   - 위 과정에서 CollisionObject(물체)를 grasp 시에는 허용하도록, MoveIt2의 AllowedCollisionMatrix를 조정. 즉, 그리퍼(fin_{links})와 해당 CollisionObject간 충돌을 무시.
   - 다른 충돌 (책상, 로봇자신)은 회피.
   - 제한 시간 (planning_time)을 1~2초로 설정 (실시간성을 위해 너무 오래 탐색하지 않도록).
   - 성공 못하면, 재시도 횟수 (max planning attempts) 설정하여 몇 번 더 시도.

   이러면 대체로 feasible한 path를 찾을 수 있습니다. 만약 IK 불가 등 실패하면, grasp_pose 자체를 수정해야 합니다 (다른 approach 각도 등). 이 경우 VLM 인지단계에서 orientation 후보를 여러 개 넣어둘 수도 있지만, 우선 simple approach로 충분할 것으로 가정합니다.

4. **Trajectory 실행:** Plan 성공 시, MoveIt2의 MoveGroupInterface를 통해 trajectory 실행 명령을 보냅니다. 그러면 ros2_control을 통해 Kinova 드라이버가 각 관절을 시퀀스대로 움직이게 됩니다. 이 동안 피드백으로 현재 로봇 상태를 모니터링하며, 만약 중간에 문제 발생 (충돌 감지, 목표 미달)하면 정지합니다.

5. **Gripper 동기화:** MoveIt2 plan에는 그리퍼 동작도 타임라인에 포함될 수 있습니다 (grasp_posture가 특정 시점에 닫히도록). 하지만 Kinova gripper는 속도가 느리므로, 보통 grasp_pose 도달 직전에 그리퍼 닫힘을 트리거하고, 충분히 잡을 시간을 잠시 둡니다. MoveIt2 pick() 함수는 이 과정을 자동화하지만, 세밀히 제어하려면 trajectory 중간에 서버 측에서 gripper action을 호출해야 할 수도 있습니다. 실험을 통해 타이밍을 조정할 예정입니다.

6. **파지 완료 처리:** 로봇이 물체를 잡고 후퇴 위치까지 이동 완료하면, pick 동작이 끝난 것입니다. 이후 상위 로직에서 "성공" 상태를 기록하고, 필요하면 후속 동작 (예: 특정 위치에 내려놓기)으로 넘어갑니다. 만약 pick 실패 (ex. MoveIt plan 실패, 혹은 잡았는데 물체를 놓침)하면, MoveIt2는 에러를 리턴하거나 로봇센서(그리퍼 폭 변화)를 통해 알 수 있습니다. 이때 VLM 인지 재시작 → pick 재시도를 할 수 있습니다.

정리하면, VLM 기반 grasping 방법론은 주로 **정확한 인지와 기계적 계획**으로 구성됩니다. 이는 이미 검증된 MoveIt2 라이브러리를 활용하기 때문에 모션 안정성은 높지만, 인지가 틀리면 아무 소용없고, 또한 정적인 목표에 대해서만 동작한다는 제한이 있습니다. 그러나 본 연구 범위인 책상 위 정적인 물체 파지에는 잘 부합하며, 특히 **언어 지시에 따른 임의 물체 파지**라는 목표 기능을 충실히 수행할 수 있을 것입니다.

추가로, VLM 기반 방법론의 변형으로 **affordance 인식**을 응용하는 아이디어도 있습니다. 예를 들어, CLIP으로 "잡을 곳"을 이미지 내 찾아내도록 훈련하거나, 또는 Semantic Segmentation 모델 (SAM2, Semantic-SAM 등)로 물체의 부위별 마스크를 얻어, 그중 "잡기 좋은 부분"을 선택하는 것입니다 ([자연어 기반 로봇 파지 연구.pdf](file://file-Ueu1VAWN22a4YWEshfqYsx#:~:text=4,%ED%95%99%EC%8A%B5%20%EB%82%9C%EC%9D%B4%EB%8F%84%20%EC%B8%A1%EB%A9%B4%EC%97%90%EC%84%9C%EC%9D%98%20%EB%B9%84%EA%B5%90%20%EB%B6%84%EC%84%9D)). 이는 아직 연구 단계지만 향후 파지 성공률을 높이는 방향으로 고려될 수 있습니다.

강화학습 기반 Grasping 방법론 (Isaac Gym 및 RL 세팅)

이번 장에서는 강화학습 기반으로 로봇 파지 정책을 학습하고 적용하는 구체적인 절차와 기법들을 다룹니다. 앞서 시스템 설계 및 알고리즘 비교 부분에서 개요를 제시했으므로, 여기서는 **Isaac Gym 환경 설정**, **학습 과정 상세**, **성능 향상 기법**, 그리고 **실제 적용시 고려사항**을 중심으로 기술합니다.

시뮬레이션 환경 세부 설정 (Isaac Gym)

Isaac Gym을 사용하여 수천 개의 병렬 환경을 구동하기 위해, 먼저 파지 환경을 하나 정의하고 이를 다중 복제하도록 합니다.

- **로봇 모델 불러오기:** Kinova Gen3 Lite의 URDF 또는 USD 모델을 불러옵니다. Isaac Gym에는 예제로 Franka Emika Panda, UR5 등 몇 가지 로봇이 있지만 Kinova는 custom 추가가 필요합니다. URDF를 파싱해 물리 속성을 세팅하거나, Gym CPU pipeline으로 불러 geometry와 관절 제약 등을 정의합니다. 로봇 베이스를 world에 고정하고, 각 관절에 모터(토크 제어) 또는 position drive를 설정합니다. Gripper는 1-DOF (두 finger linkage 연동)로 모델링하여, 닫힘 시 충분한 힘으로 물체를 잡도록 effort 한계를 설정합니다.
- **물체 생성 및 배치:** 다양한 **훈련용 객체**를 준비합니다. 단순하게는 primitive shape (박스, 공, 원기둥)을 사용하고, 추가로 YCB dataset 같은 공개 3D 모델(컵, 병, 택배상자 등)을 일부 활용할 수 있습니다. Isaac Gym에서는 기본 도형은 쉽게 생성 가능하며, 복잡한 mesh는 물리엔진 상에서 계산 비용이 조금 늘 수 있지만 허용 범위입니다. 각 environment마다 하나의 물체를 테이블 위에 놓는데, 초기 위치는 로봇 기준 좌표계에서 x: 0.3~0.5m, y: -0.2~0.2m (팔 정면 범위 내 랜덤), z: 테이블 높이 (0.0m) 위로 살짝 띄워 (drop test) 안정적으로 배치합니다. 초기 orientation도 무작위. 테이블은 얇은 판으로 static 설정.

    - **도메인 랜덤화:** 각 에피소드마다 물체의 크기(10~30cm 범위), 질량(0.1~2kg), 마찰계수(0.5~1.0) 등을 랜덤화합니다 ([로봇 파지 연구 자료 조사.pdf](file://file-No9vxQcXh7E7rCsH9LroUn#:~:text=39,07394)). 또 조명, 카메라 각도도 여러 환경에서 바꾸어, 시각 입력 다양성을 확보합니다.
    - **멀티-객체 시나리오:** 추후 확장으로 한 환경에 2~3개의 물체를 둘 수도 있지만, 그러면 정책이 어느걸 집어야 할지 애매해지므로, 학습 목표가 불명확해집니다. 따라서 기본은 한 번에 하나의 물체만 놓고 집도록 합니다.

- **관찰 (Observation) 설계:** 에이전트에게 주는 관찰 정보는 몇 가지 모달리티로 구성됩니다.

    1. 로봇 자신의 상태: 6개 관절 각도 및 속도. (필요시 end-effector pose로 변환하여 줘도 됨)
    2. 시각 관찰: 전면 카메라 이미지 (앞쪽 45° 위에서 로봇과 테이블을 함께 보는 시야). RGB 128x128 또는 256x256으로 제공. 심층 강화학습에서 고해상도는 불필요한 연산을 늘리므로 적당히 축소. 딥마인드 QT-Opt 경우 어깨 너머 카메라 RGB를 472x472 grayscale 변환해 사용하기도 했습니다 ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation), 우리도 단색 or depth만 써볼 수도 있습니다. Depth map은 물체와 테이블 구분이 쉬워 유용할 수 있지만, 잡으려는 대상 하나만 있는 경우 RGB도 충분합니다. 일단 **RGB-D 둘 다** 스택하여 CNN 입력으로 넣을 계획입니다.
    3. 목표 관련 정보: 우리 작업은 "아무거나 집어"라 가정하여 특정 목표 ID는 없습니다. 만약 여러 물체 중 특정 하나만 잡도록 바꾸려면, 관찰에 목표의 one-hot이나 언어 embedding 등을 넣어줘야 하지만, 여기서는 불필요합니다.

    관찰 텐서는 위 요소들을 정규화하여 하나로 concatenate합니다. 관절 각도 등은 [–1,1]로 normalize, 이미지 픽셀도 [0,1] or [-1,1] 스케일로. 최종 관찰 크기는 (C x H x W + joint_dim), 예: RGB-D (4x128x128=65536) + 12 = 65548 차원 정도. CNN에서 convolution으로 차원 감소를 하므로 실제 네트워크 파라미터는 manageable합니다.

- **행동 (Action) 설계:** 앞서 설명한대로 end-effector 3D delta pose + gripper open/close로 설계합니다. 그러나 시뮬레이션에서는 굳이 delta pose가 아니어도, 6관절 모터 torque 직접 출력도 가능하긴 합니다. 하지만 torque 제어는 학습 난이도가 높고, 위치기반 제어보다 불안정할 수 있습니다. 반면 delta pose 방식은 일종의 Cartesian impedance 컨트롤처럼 동작하여, 정책이 안정적으로 공간에서 동작하도록 합니다. 구현적으로는, action 6개를 받아 이를 IK로 joint velocity command로 변환합니다. 또는, 6개 관절 속도를 직접 action으로 하는 것도 고려됩니다.

    - Gripper action: 1개 이산값 (0=open, 1=close)로 간주해도 되고, continuous 0~1로 출력하여 thresholding해도 됩니다. RL에서는 이산이 더 자연스러울 수 있어 multi-head output으로 actor의 일부는 gripper open/close 확률을 소프트맥스로 내보내 2-category 분류처럼 취급합니다. Critic네트워크는 연속+이산 혼합 action도 문제없이 다룰 수 있도록 구성해야 하므로, SAC의 경우 gripper를 continuous 0~1로 두고 reward계산만 별도로 해주는 편이 단순할 수 있습니다.

- **Dynamics and Timestep:** 시뮬레이션의 한 스텝당 dt를 0.05~0.1초로 세팅합니다. RL 알고리즘의 action 빈도와 일치하게 됩니다. 각 에피소드 길이는 최대 3~5초 (즉 60~100 steps)로 두고, 그 안에 집으면 성공, 못 집으면 실패로 종료합니다. PyTorch로 학습 시 4096 env * 100 steps = 409600 transitions per episode rollout, 상당히 많지만 GPU 상에서 한 번에 계산하므로 감당 가능합니다.

강화학습 학습 과정 상세

환경이 준비되었으니, SAC 알고리즘을 실제 적용하는 과정을 설명합니다:

1. **초기화:** Actor 네트워크 π(s), Critic 네트워크 Q(s,a) 두 개 (또는 double Q로 2개) 가중치를 랜덤 초기화. 리플레이 버퍼 생성 (정크 크기: 1e6 transitions 저장 등).

2. **Warm-up 탐색:** 초기 정책은 랜덤이므로, 우선 일정 시간 epsilon-greedy 혹은 Gaussian noise 높은 상태로 rollouts을 수행합니다. 이때 대부분 실패하지만, 다양한 시도가 버퍼에 쌓입니다. 물체를 전혀 못잡는 상태라 reward=0인 에피소드가 많을 것이므로, shaped reward (거리 짧아지면 +r 등)로 약간의 gradient가 생기게 합니다.

3. **Experience Collection:** 각 병렬 환경에서 policy에 따라 action을 수행하고, 그 결과 next state, reward, done을 얻어 버퍼에 저장합니다. 이 과정을 GPU 상에서 vectorize하여 매우 빠르게 합니다. Isaac Gym에서는 step마다 모든 환경 관찰을 tensor로 업데이트하고, policy inference도 tensor연산으로 합니다. (PyTorch나 JAX로 작성)

4. **Policy Update (SAC):** 일정 수의 step마다 (e.g. every 4 steps) SAC gradient update를 실행합니다. 버퍼에서 배치 (e.g. 1024 samples) 랜덤 추출 -> Q-loss 계산 (벨만: r + γ * min(Q_target(s', π(s'))) - Q(s,a)) -> 업데이트; 그리고 policy loss 계산 (entropy regularized: grad log π(a|s) * (α*logπ - Q)) -> 업데이트. 타겟 네트워크 soft update. 이러한 업데이트를 한 iteration에 여러 번 (e.g. 2) 수행합니다. 이때 GPU 병렬로 워낙 많은 데이터를 쌓기에, 보통 environment step 수 >> update step 수가 됩니다.

    - **학습율 및 안정화:** 초기엔 exploration이 중요하므로, α (entropy weight)를 높게 유지하거나, Gaussian noise를 policy output에 추가합니다 (SAC는 본래 entropy로 exploration하지만, entropy 타겟팅이 있긴 합니다). 학습이 진행되며 α가 자동 조절되면서 exploitation 쪽으로 기울게 됩니다.

5. **Evaluation:** 주기적으로 (say every 10k steps) 현재 policy로 deterministic하게 여러 에피소드 시뮬레이션 돌려보고 성공률 계산, 텐서보드 등에 로깅합니다. 또 reward 평균, Q-value 평균, policy entropy 등의 커브를 기록해 학습 상황을 파악합니다.

6. **수렴 조건:** 성공률이 일정 기준 (ex. 90%) 넘고 plateau 지속되면 학습 종료를 판단합니다. 또는 지정된 최대 step (ex. 5 million steps) 도달 시 끝냅니다. 충분히 수렴했다면, Actor 네트워크 파라미터를 파일로 저장합니다.

7. **Policy Refinement (선택사항):** 만약 특정 object에서 실패율 높거나, 편향이 보이면, 그 사례만 모아 fine-tune하는 등의 기법을 쓸 수 있습니다. 하지만 과도한 fine-tune은 다른 성능 떨어뜨릴 수 있으니 신중히.

성능 향상 및 보정 기법

학습을 원활히 하거나 정책 성능을 높이기 위한 추가 기법들:

- **Heristic Pre-grasp**: 완전 처음부터 학습하기 어렵다면, 초기 몇 에피소드에서 물체 근처까지 로봇을 움직이는 행동을 휴리스틱으로 삽입해, 거기서부터 학습하게 할 수도 있습니다. 이를 Hindsight Experience Replay (HER)나 Curriculum으로 볼 수도 있는데, 첫 단계부터 완전 목표를 이루는건 아닌 incremental 목표를 부여해 학습하는 방식입니다.
- **Hindsight Experience Replay (HER)**: 에피소드가 실패로 끝나더라도, 마지막에 로봇이 어떤 위치까지 손을 뻗었다면, 그걸 목표로 하는 가상 목표로 바꿔 성공 경험으로 추가하는 HER 기법을 적용할 수 있습니다. 그러나 여기서는 목표가 특정 물체이므로 일반 HER은 부적합, 대신 "결과적으로 잡은 물체가 목표"라 가정해 추가 학습하는 식의 HER 변형은 생각해볼 수 있습니다.
- **Multi-Task Learning**: 향후 여러 객체 종류별로 정책을 조금 다르게 해야 할 필요가 생기면, 하나의 거대 정책 대신 object embedding을 입력으로 받아 동적으로 다른 동작을 하는 정책을 학습하는 방법도 연구되고 있습니다. (이번 범위에는 해당 안됨)

- **Sim-to-Real Adaption**: 도메인 랜덤화는 기본이고, 추가로 **System Identification** 파라미터 튜닝이나, **Visual Domain Randomization** (텍스처, 조명)으로 현실 차이를 줄입니다. 필요시 **Real data fine-tune** – 예: 실제 로봇에서 random policy로 시도한 작은 데이터 + imitation loss로 보정 – 을 고려할 수 있지만, 우선은 하지 않습니다.

실환경 적용과 로봇 통합

학습이 완료되면, Jetson에 학습된 모델을 배포합니다:

- PyTorch 모델을 TorchScript로 export하거나, ONNX 변환 후 TensorRT 최적화도 가능하지만, 정책 네트워크는 작은편이라 그냥 PyTorch로 CPU inferencing도 실시간은 가능할 겁니다 (또는 Jetson GPU 사용할 수도).
- ROS2 Node로 만들고, 구독하는 토픽: 카메라 이미지, 로봇 상태. 발행: 로봇 제어 명령 (geometry_msgs/Twist 혹은 control_msgs/JointJog 등 실시간 제어 메시지).
- RL 정책은 매주기 (20Hz 등) 카메라 이미지 + 현재 관절각을 받아 신경망 forward -> delta end-effector 움직임 산출. 이를 IK 계산하여 6관절 속도 명령으로 변환, ROS2로 publish. (Kinova ROS2 드라이버에 joint velocity control 인터페이스가 있어야 함. 없다면, MoveIt2의 jog_arm 패키지를 이용해 Twist -> joint velocity 변환 가능)
- Gripper 제어: RL policy가 잡기 시도를 하려 할 때(close action=1 출력), 이를 감지하여 gripper closed-loop로 꽉 쥐도록 명령. 그리고 일정 힘 또는 위치에 도달하면 유지.

실행 도중, 만약 물체를 잡았으면 (그리퍼 힘센서, 혹은 깊이 변화로 감지 가능), 정책 출력을 무시하고 후속 동작(들기 등)을 상위 레벨에서 지시할 수 있습니다. 하지만 RL 정책이 알아서 들도록 학습되어 있을 수도 있으므로, 최대한 자율에 맡깁니다. (QT-Opt의 경우, grasp 후 lifting도 포함한 장시간 horizon을 학습했음 ([1806.10293] QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation). 우리도 episode 종료조건을 들어올리기까지로 했다면 정책이 당연히 lift 함)

마지막으로, RL 기반 방법의 신뢰성 위해 **안전장치**를 점검합니다. 시뮬레이터에서는 제한했지만 실제에서는 예기치 않은 행동(빠르게 휘두르거나 테이블 강타 등)하지 않도록, 제어 명령에 속도/가속도 제한을 겁니다. 또 운영 중 사람이 접근하면 정지하는 emergency stop 등을 병렬 구현합니다.

실험 설계 및 시나리오 (책상 위 단순 객체 및 복합 객체 파지 테스트)

이제 개발된 두 가지 파지 시스템(VLM 기반, RL 기반)의 성능을 평가하기 위한 **실험 시나리오**를 설계합니다. 실험은 실제 Kinova Gen3 Lite 로봇팔과 RGB-D 카메라로 구성된 테스트베드에서 이루어지며, 책상 위에 다양한 물체를 놓고 파지 테스트를 수행합니다. 각 접근법별로 성능을 측정하고 비교하기 위해, 아래와 같이 단계적인 시나리오를 구성합니다:

실험 환경 세팅

- **장소:** 실내 사무실 환경의 책상 위. 책상 크기 120x60cm, 높이 약 75cm. 로봇팔은 책상 한 쪽 모서리에 고정되어 베이스가 책상면과 같은 높이에 위치.
- **조명:** 일반 실내 조명 (형광등)으로 일정한 밝기 확보. 필요시 스탠드 조명 추가. VLM 인지에 지장없도록 그림자 최소화.
- **카메라 위치:** 책상 전면 상부에 RGB-D 카메라 장착 (로봇팔 관절에 달지 않고 별도 스탠드에 고정하여 흔들림 배제). 카메라 시야에 책상 위 전체 영역이 들어오게 1m 높이에서 약 45° 각도로 내려봄.
- **물체 풀(pool):** 실험에 사용할 **테스트 물체**들을 선정.
  - *단순 형태 객체*: 정육면체 나무 블록, 공 모양 장난감, 원통 (캔), 컵, 볼펜 등 **단일 부품**으로 된 간단한 모양들.
  - *복합 형태 객체*: 손잡이 달린 머그컵, 책(얇고 평평), 가위(이질 부품 구성), 작은 장난감 로봇(복잡한 형상) 등 **파지가 상대적으로 까다로운** 것들.
  - *총 개수*: 최소 10종 (단순 5, 복합 5) 정도 준비.
- **객체 크기:** 소형~중형 (대략 5cm ~ 15cm 크기). 너무 큰 것은 그리퍼가 못 잡으므로 제외.
- **배치 방식:** 각 시나리오마다 책상 위 특정 위치에 특정 객체들을 배치. 배치 변인: 객체 수 (1 vs 여러 개), 객체 간 거리 (충분히 떨어짐 vs 다소 인접), 물체 놓인 자세 (정상 vs 옆으로 눕힘), 등.

실험 시나리오

1. **시나리오 A – 단일 단순 객체 파지:**
   - 한 번에 **한 개의 단순 형태 물체**를 책상 위 임의 위치에 놓고, 로봇이 그 물체를 파지하도록 명령.
   - 이 경우 VLM 접근에서는 "해당 물체 집어" 명령을 주면 되며 (예: "블록을 집어"), RL 접근에서는 카메라에 그 물체만 보이므로 정책이 자연히 그것을 잡을 것.
   - 모든 단순 객체에 대해 각 5회 이상 반복 테스트하여 **성공률(%)**, **평균 소요 시간** 등을 측정.
   - 기대: 두 접근 모두 높은 성공률 (>90%) 달성. 시간은 VLM은 인지+경로계획으로 3~5초, RL은 2~3초 등 차이날 수 있음.
2. **시나리오 B – 단일 복합 객체 파지:**
   - **한 개의 복합/어려운 형태 물체** (머그컵, 책 등) 대상으로 테스트. 각 물체 5회 이상 시도.
   - 머그컵의 경우 손잡이 파지 vs 몸통 파지 여부 관찰. 책의 경우 옆면 파지 시도 확인.
   - 성공률이 떨어질 가능성 있으므로, 실패 원인 (미끄러짐, 인식 오류 등) 기록.
   - 이 결과로 각 접근법의 **어려운 물체 대응력** 평가.
3. **시나리오 C – 다중 객체 중 특정 물체 선택 (VLM 특화):**
   - 책상 위에 **두 개 이상의 서로 다른 물체**를 동시에 배치 (예: 공 + 컵, 책 + 블록 등 조합).
   - VLM 접근: 명령으로 특정 물체 지목 ("공을 집어") → 해당 물체만 집었는지 확인.
   - RL 접근: (학습 정책은 '아무거나' 잡도록 되어 특정 선택 능력 없음) 일단 눈앞에 보이는 아무 물체나 잡을 것이므로, 어떤 걸 잡았는지 기록. RL은 보통 더 가까운/큰 걸 우선할 것으로 예상.
   - 비교 포인트: VLM은 **정확히 지시된 것**을 집는지 (정확도), RL은 **랜덤/우선순위** 어떻게 되는지.
   - 10회 이상 다양한 조합으로 실시.
4. **시나리오 D – 잡기 어려운 위치/상황:**
   - 물체가 **테이블 모서리에 걸쳐 있거나**, **벽에 인접**, **다른 물체 옆에 바짝 붙은** 경우 등 난이도 조건 부여.
   - 이런 경우 VLM+MoveIt은 충돌회피로 인해 경로를 크게 우회하거나 파지 자체를 포기할 수도 있음. RL 정책은 훈련에 없던 경우 당황하여 실패할 수 있음.
   - 각 접근 5회씩 테스트, 성공여부와 동작 패턴 관찰.
5. **시나리오 E – 실시간 변화 대응 (추가 테스트):**

- (이것은 stretch goal) 실험 중 사람이 물체를 약간 밀거나 위치를 변경했을 때 시스템 대응.
- VLM+MoveIt은 현재 구현상 바로 대응 못하고 실패할 가능성 높음 (센싱 ~ 계획 사이 갭). RL은 피드백 제어로 어느정도 따라갈 수 있을지 관찰.
- 이 테스트는 정량평가보다는 동작 관찰 위주.

각 시나리오에서 **평가 지표**는 다음과 같이 정의합니다:

- **파지 성공률:** N번 시도 중 성공한 횟수/N * 100%. (성공 기준: 물체를 확실히 집어 들어올려 5cm 이상 높이로 2초 이상 유지)
- **평균 파지 시간:** 성공한 경우에 한해, "시작 명령 시점"부터 "물체 들어올려 안정화 시점"까지 측정한 시간 (초). 실패한 경우 별도로 표시.
- **계획/추론 시간** (VLM 전용): 명령 입력 후 인지+경로계획 완료되기까지 시간.
- **정확한 대상 파지 비율:** (시나리오 C의 경우) 지시한 물체를 잡은 비율.
- **실패 유형 분류:** 실패마다 원인을 분류:
  - 인식 실패 (오인식, 놓친 경우),
  - 파지 동작 실패 (접近 불안정, 미끄러짐),
  - 경로계획 실패 (충돌로 인한 미수행) 등.
- **부수적 관찰:** 파지 품질 (잡은 후 흔들림 여부), 로봇 모션의 부드러움, 시스템의 재현성 등 주관적 평가 메모.

데이터 수집 및 분석 방법

각 실험에서 나온 로그 데이터를 체계적으로 수집합니다. ROS2 bag에 토픽 (로봇 joint states, 카메라 영상**1. 연구 주제 및 프로젝트 제목 후보 (영문 / 한국어 번역):**

1. *Zero-Shot Vision-Language Robotic Grasping for Everyday Objects* (일상 물체에 대한 **제로샷 비전-언어 로봇 파지** 연구)
2. *Reinforcement Learning-Driven Grasping on Edge Robots* (엣지 로봇 플랫폼에서 **강화학습 기반 파지** 구현)
3. *Vision-Language vs. Reinforcement Learning for Robotic Manipulation: A Comparative Study* (로봇 조작을 위한 **비전-언어 대 강화학습 접근법 비교 연구**)
4. *Real-Time Language-Guided Grasping with VLMs on Jetson AGX Orin* (Jetson AGX Orin 상의 **실시간 VLM 기반 언어 지시 파지 시스템**)
5. *Integrating CLIP and Deep RL for Robust Robotic Grasping* (**CLIP과 심층 강화학습을 통합한 로봇 파지** 기법 연구)

비전-언어 모델과 강화학습을 활용한 로봇 파지: 시스템 설계 및 비교 연구 보고서

서론 (연구 배경, 필요성 및 목표)

현대 로봇공학에서 **로봇 팔의 물체 파지(grasping)**는 핵심 도전 과제 중 하나입니다. 특히 일상 환경의 **비구조화된 실내 공간**에서 사람처럼 다양한 물체를 집어드는 능력을 로봇에게 부여하려면, 로봇이 사전에 학습되지 않은 **새로운 물체**도 인식하고 파지할 수 있어야 하는 필요성이 대두됩니다. 전통적인 컴퓨터 비전 기반 파지 기법들은 대개 사전 정의된 물체나 특징에 의존하여 일반화에 한계가 있고, 강화학습이나 딥러닝 기반 파지 검출 또한 대량의 학습 데이터와 시간, 연산 자원이 필요합니다.

최근 등장한 **비전-언어 모델(VLM, Vision-Language Model)**들은 거대한 이미지-텍스트 데이터셋으로 학습되어 **제로샷(Zero-Shot)**으로 새로운 객체를 인식하거나 설명할 수 있는 능력을 보여주고 있습니다 (CLIP: Connecting text and images | OpenAI) (CLIP: Connecting text and images | OpenAI). 예를 들어 OpenAI의 CLIP 모델은 자연어 supervision만으로 다양한 시각 개념을 습득하여 별도 학습 없이도 새로운 분류 작업에 적용될 수 있음을 증명하였습니다 (CLIP: Connecting text and images | OpenAI). 이러한 **멀티모달 AI**의 발전으로 로봇은 사전에 본 적 없는 물체도 **텍스트로 지시**하면 인식하고 찾아낼 수 있는 가능성이 열렸습니다. 따라서 "빨간 컵을 집어줘"와 같은 **자연어 명령**을 이해하여 해당 물체를 찾아 파지하는 **언어-지시 기반 파지**가 연구의 새로운 방향으로 주목받고 있습니다.

동시에, **강화학습(Reinforcement Learning, RL)**을 이용한 로봇 파지 역시 활발한 연구 분야입니다. 심층 강화학습은 시각 입력을 통해 **폐루프(closed-loop)** 제어 정책을 학습하여 복잡한 파지 동작을 스스로 익힐 수 있고, 기존에 사람이 수동으로 설계하던 파지 동작이나 경로계획을 자동으로 학습하도록 합니다. 예를 들어, 구글의 QT-Opt는 58만 회에 달하는 실제 로봇 팔의 파지 시도를 통해 거대한 Q함수를 학습함으로써 **보지 못한 새로운 객체에 대해서도 96%의 높은 파지 성공률**을 달성하였으며, 재파지(regrasping)나 물체 위치 재조정과 같은 복잡한 동작도 자동으로 습득하였습니다. 이러한 **모델 프리(Model-free)** 강화학습 접근법은 인간이 정의하기 어려운 최적의 파지 전략을 로봇이 스스로 발견하게 해주지만, 방대한 학습 시도가 필요하고 학습된 정책이 새로운 환경 변화나 명시적으로 지정된 목표(예: 특정 물체 선택)에 쉽게 대응하지 못한다는 한계도 있습니다.

본 연구의 목표는 주어진 **Kinova Gen3 Lite 로봇팔**과 **Jetson AGX Orin** 컴퓨팅 플랫폼, **RGB-D 카메라** 기반의 책상 위 실내 환경을 대상으로, **비전-언어 모델 기반 파지 방법**과 **강화학습 기반 파지 방법** 두 가지를 **시스템 수준에서 설계 및 구현**하고 성능을 비교 분석하는 데 있습니다. 구체적으로, 사전 학습된 VLM을 활용하여 **레이블되지 않은 일반 물체**를 **자연어**로 지칭하며 파지하는 시스템과, 물체의 종류와 상관없이 **강화학습**으로 파지 동작을 익히는 시스템을 각각 개발합니다. 이를 통해 두 접근법의 **실시간 동작 성능**, **일반화 능력**, **구현 복잡도** 및 **장단점**을 면밀히 평가하고, 향후 **모바일 매니퓰레이터**로의 확장 등 응용 방향을 제안하는 것이 본 연구의 궁극적인 목표입니다.

요약하면 본 연구는 **Vision-Language 기반 파지 시스템**과 **강화학습 기반 파지 시스템**을 ROS2 Humble 환경에서 구현하여, Jetson Orin 상에서의 **실시간 동작 가능성**을 검증하고, 책상 위 다양한 물체들(단순 형태부터 복잡한 형태까지)에 대한 **파지 성공률**과 **효율**을 평가하는 **심층 연구**입니다. 다음 장들에서는 관련 최신 연구 동향과 기술을 살펴보고, 두 접근 방식의 시스템 구조와 알고리즘, 구현 세부사항을 제시한 뒤, 실험 방법과 예측되는 결과를 논의하도록 하겠습니다.

관련 연구 동향 (Vision-Language Models, 강화학습 기반 그래스핑, Jetson 최적화 기술)

본 장에서는 본 연구와 밀접한 관련이 있는 세 가지 기술 영역의 최신 동향을 살펴봅니다: **비전-언어 모델(VLM)**을 활용한 로봇 인지, **강화학습 기반 로봇 파지**, 그리고 **Jetson AGX Orin과 같은 엣지 플랫폼에서의 최적화 기법**입니다.

비전-언어 모델을 활용한 로봇 인지 및 파지

Vision-Language Model(VLM)은 이미지와 텍스트를 공동 학습하여 시각 장면에 대한 높은 수준의 이해를 가능케 하는 **기초 모델(foundation model)**입니다. 대표적으로 CLIP (Contrastive Language–Image Pre-training)은 웹상의 방대한 이미지-텍스트 쌍으로 학습되어, **텍스트 프롬프트만으로 이미지 내 객체를 식별**하거나 분류할 수 있는 강력한 **제로샷** 능력을 보여주었습니다 (CLIP: Connecting text and images | OpenAI). CLIP은 별도의 파인튜닝 없이도 원하는 클래스 이름을 텍스트로 주어 분류 작업을 수행하는 등 범용적인 시각 인식 능력을 보여주었으며, **자연어 지시만으로** 새로운 작업에 적용될 수 있음을 입증했습니다 (CLIP: Connecting text and images | OpenAI). 이러한 능력은 로봇에게도 유용하여, 사람이 일일이 물체를 학습시키지 않더라도 **"노란색 공 집어"**와 같은 명령으로 **새로운 물체**를 찾아낼 수 있는 가능성을 제공합니다. 최근 로봇 분야에서는 이러한 VLM을 파지에 활용하려는 시도가 늘고 있습니다. Shridhar 등은 CLIP 모델을 로봇에 적용하여 **시각 정보와 언어 정보를 연결**하고, **로봇 팔 제어**와 결합하는 연구 방향을 제시하였습니다. 특히 **CLIPort** (CLIP + Transporter) 등의 시스템은 언어로 지목된 영역을 파지하도록 설계된 예로서, 언어와 시각 피처를 결합하여 **특정 객체나 위치**를 골라 조작하는 성능을 향상시켰습니다.

한편, **세그멘테이션** 분야에서도 강력한 기초 모델이 등장하여 파지에 활용되고 있습니다. Meta AI의 **Segment Anything Model (SAM)**은 포인트, 박스 등의 입력 프롬프트에 대해 이미지 내 **임의의 객체 마스크**를 고품질로 산출할 수 있는 모델로, 11억 개 이상의 마스크로 학습된 덕분에 거의 모든 분할 작업에 **제로샷** 성능을 보입니다. SAM은 본래 범용 세그멘터로 제안되었으나, **Grounding DINO**와 결합한 **Grounded-SAM**과 같이 **텍스트 프롬프트로 목표 객체를 탐지 및 분할**하는 파이프라인도 등장했습니다. Grounding DINO는 CLIP의 이미지-텍스트 임베딩을 활용하여 **텍스트로 지시된 객체의 바운딩박스**를 찾는 **오픈 어휘 객체 탐지** 모델로, SAM과 연계하면 **임의의 텍스트로 지칭한 객체 영역**을 분할해낼 수 있습니다. 이는 로봇이 "컵의 손잡이 부분"처럼 구체적인 부위를 인식하는 데에도 응용될 수 있어, **파지 접점 결정에 활용 가능한 시각정보**를 제공합니다. 실제로 **Affordance 기반 파지** 연구에서는 CLIP과 같은 VLM으로부터 얻은 객체의 의미 정보를 활용해 해당 물체의 유용한 파지 부위를 추론하기도 합니다. 예를 들어, CLIP으로 "컵의 손잡이"에 해당하는 픽셀들을 인식하고 이를 파지 지점으로 정하면, 단순히 물체의 기하학적 중심을 잡는 것보다 **더 안정적이고 기능적인 파지**가 가능합니다. 이러한 접근은 **도구 사용**이나 **복잡한 형상의 물체**를 다룰 때 특히 유용하며, 인간이 물체를 사용하는 방식을 로봇 파지에 반영한다는 점에서 주목받고 있습니다.

비전-언어-액션(VLA) 모델의 부상도 눈여겨볼 만합니다. 이는 시각 및 언어 정보를 입력으로 받아 **로봇의 동작 시퀀스 출력**까지 통합한 모델로, 일종의 거대 멀티모달 모델을 로봇 제어에 적용하는 개념입니다. 2023년 공개된 **OpenVLA**는 70억 개 이상의 파라미터를 가진 **7B 규모의 VLA 모델**로, 실제 로봇 시연 97만 에피소드로 학습되어 다양한 조작 작업을 일반화하도록 시도되었습니다. OpenVLA와 같이 **대규모 시퀀스 모델**에 언어 지시와 시각 관찰, 로봇 액션을 함께 학습시키는 접근은, 향후 특정 작업에 한정되지 않는 **범용 로봇 비서** 개발의 토대가 되는 방향으로 각광받고 있습니다. 이러한 VLA 모델들은 아직 방대한 학습 데이터와 자원이라는 현실적 제약이 있지만, **Say-Can**(Google), **PaLM-E**(Google) 등 대형 멀티모달 모델을 활용한 로봇 지능 연구 흐름과 맥을 같이합니다. 본 연구에서는 VLM을 주로 활용하지만, 향후 **VLA**로의 확장 가능성도念頭에 두고 있습니다.

마지막으로, 다양한 VLM 관련 최신 기법들을 살펴보면, **경량화 및 실시간화**가 중요한 트렌드입니다. 대형 모델을 로봇에 쓰려면 제한된 온보드 연산 자원에서 돌아가야 하므로, **효율성에 중점**을 둔 모델 변형이 이루어지고 있습니다. 예를 들어, **NanoOWL**과 **NanoSAM**은 각각 OWL-ViT 및 SAM을 NVIDIA Jetson급 장치에서 **실시간 추론 가능**하도록 최적화한 모델들로, ResNet18 기반 경량 아키텍처와 TensorRT 엔진 최적화를 통해 **Jetson AGX Orin에서 NanoSAM은 8.1ms 추론 (~123 FPS)**을 달성할 정도로 가볍습니다. NanoOWL은 자연어 프롬프트 기반 객체탐지 모델인 OWL-ViT를 경량화한 것으로, Jetson에서 OWL-ViT (ViT-B/32) 모델을 **약 95 FPS**로 구동 가능하며, 더 큰 ViT-B/16 모델도 25 FPS로 구동할 수 있습니다. NanoOWL+NanoSAM을 조합하면 **제로샷 객체 탐지 및 세그멘테이션**을 동시에 실시간 수행할 수 있어, **Jetson Orin에 최적화된 파지 인식 파이프라인**으로 유망합니다. 그밖에도 MobileSAM, MobileVLM 등 모델 경량화 연구가 활발하며, 이러한 기술 동향을 참고하여 **엣지에서 운용 가능한 파지 시스템**을 설계하는 것이 중요합니다.

강화학습 기반 로봇 그래스핑

강화학습을 통한 로봇 파지 학습은 지난 수년간 많은 연구를 통해 발전해왔습니다. 초기에는 물체의 3D 모델 정보나 사전 정의된 그립 포인트를 활용했으나, 딥러닝과 결합된 RL은 **시각 관찰 → 로봇 제어 명령** 간의 매핑을 **엔드투엔드**로 학습함으로써, 사람이 규칙을 설계하지 않아도 로봇이 스스로 **파지 전략**을 터득하게 했습니다. 특히 **심층 Q-러닝** 기반의 접근이 주목받았는데, 앞서 언급한 QT-Opt는 딥마인드 및 구글 연구진에 의해 **실세계 데이터만으로** 학습된 사례입니다. QT-Opt는 시뮬레이션 없이 실제 로봇으로 58만 회의 시도를 거쳐 학습되었음에도, 미학습 객체에 96% 성공률을 달성하고, **폐루프 제어**로 파지 도중 물체가 밀리거나 실패할 경우 즉각 재시도하는 등 **적응적 행동**을 보여주었습니다. 이는 강화학습이 충분한 경험을 통해 전통적 방법보다도 **강인하고 유연한 파지 동작**을 얻을 수 있음을 시사합니다. 다만 현실에서 수십만 회의 시도를 실행하는 것은 비용이 매우 크기 때문에, **시뮬레이션**을 활용한 학습이 일반적입니다.

시뮬레이터 + 강화학습의 결합으로, 로봇 파지 학습의 규모를 키우고 속도를 높이는 연구들이 진행되어 왔습니다. 특히 NVIDIA의 **Isaac Gym**은 GPU 상에서 다수의 로봇 환경을 병렬로 시뮬레이션하고 동시에 RL 알고리즘을 학습시킬 수 있는 플랫폼으로, **물리 시뮬레이션과 신경망 학습을 모두 GPU 메모리에서 직접 처리**하여 기존 CPU 기반 시뮬레이션 대비 **2~3자릿수(100~1000배) 속도 향상**을 달성했습니다. 이를 통해 복잡한 로봇 과제도 단일 GPU로 수 시간~수일 내에 정책 학습이 가능해졌습니다. 예를 들어 Isaac Gym을 활용하면 수천 개의 병렬 환경에서 로봇팔 파지 시도를 동시에 수행하여, 실시간에 가까운 속도로 학습 데이터를 모을 수 있습니다. Makoviychuk 등은 이러한 **대량 병렬 RL**을 통해 고난도 로봇 제어 과제를 단시간에 풀어내는 사례들을 보고하였습니다.

강화학습 알고리즘 측면에서는 **정책 경사 방법(Policy Gradient)**과 **오프-폴리시(Off-policy)** 알고리즘들이 주로 활용됩니다. **Proximal Policy Optimization (PPO)**은 비교적 안정적으로 학습되는 on-policy 방법으로 로봇 제어에 자주 쓰이며, OpenAI 등이 물체 조작이나 로봇 다리 제어 등에 활용한 바 있습니다. **Soft Actor-Critic (SAC)**은 off-policy 기법으로 연속 제어 문제에서 **표본 효율이 높고** 안정적이라 평가받으며, 특히 로봇 팔 등의 정확한 제어에 적합합니다. 실제 연구에서는 PPO로 기본 정책을 학습한 뒤 SAC로 미세 조정하거나, 혹은 하이퍼파라미터 튜닝을 통해 둘 중 하나의 알고리즘을 선택하는 방식이 흔합니다. 본 연구에서도 **연속 동작 공간**에서 효과적인 알고리즘으로 SAC와 PPO를 후보로 고려하며, 시뮬레이션 상에서 학습 안정성과 수렴 속도를 비교하여 최적 알고리즘을 선택할 것입니다. 또한 파지 성공 시 높은 보상, 물체를 놓치거나 충돌 시 패널티 등을 부여하는 **보상 함수 설계**, 학습 초기의 탐색을 돕기 위한 **행동 노이즈** 추가, 그리고 **도메인 랜덤화**를 통한 **Sim-to-Real** 간극 축소 전략 등도 최신 연구에서 강조되는 기법들입니다. 이러한 요소들을 종합하여, 강화학습으로 학습된 파지 정책이 실제 로봇에서도 원활히 동작하도록 할 계획입니다.

Jetson AGX Orin 및 엣지 AI 최적화 기술

Jetson AGX Orin은 NVIDIA의 첨단 엣지 AI 플랫폼으로, 200 TOPS 이상의 연산 성능과 12-core ARM CPU, 최대 2048-core GPU, 32~64GB 메모리 등을 갖추고 있어 로봇 내장용 컴퓨터로 각광받고 있습니다. 그러나 여전히 대형 딥러닝 모델을 실시간 구동하기에는 제약이 있으므로, **모델 최적화 및 경량화** 기술이 필수적입니다. 본 연구에서는 Jetson 상에서 **5 FPS 이상의 추론 속도**를 목표로 하고 있는데, 이를 위해 고려하는 최적화 기법들은 다음과 같습니다.

- **TensorRT 최적화**: NVIDIA에서 제공하는 딥러닝 추론 최적화 라이브러리인 TensorRT를 사용하여 모델을 FP16 또는 INT8로 양자화(quantization)하고, Jetson의 GPU 및 NVDLA 가속기에서 효율적으로 돌아가도록 엔진화합니다. 예를 들어 앞서 언급한 NanoOWL, NanoSAM도 TensorRT를 통해 최적화되었습니다.
- **경량 모델 사용**: 가능하면 MobileNet, EfficientNet 등 파라미터 수가 적은 backbone을 활용한 모델을 채택합니다. 본 연구에서는 SAM 대신 MobileSAMv2 (경량화 SAM), 거대 언어모델 대신 LLaMA 기반 MobileVLM 등 **Mobile 모델**들을 고려하며, 실험적으로도 NanoSAM 등은 성능 대비 현저히 가벼워 실시간성에 유리함을 확인했습니다.
- **모델 파이프라인 최적화**: 여러 딥러닝 모듈을 연속 사용할 경우, **GPU 메모리 복사 최소화** 및 **병렬화**가 중요합니다. 예를 들어 객체 탐지 후 곧바로 세그멘테이션을 할 때, **한 프로세스 내에서 연계**하여 실행하거나 TensorRT에서 two-stage pipeline을 하나의 engine으로 합치는 방안을 모색합니다. Grounded-SAM의 경우 Grounding DINO와 SAM을 별개로 돌리지만, **SAM2** 등의 최신 연구에서는 **단일 트랜스포머**로 객체 탐지+분할을 통합하여 중간 과정을 줄이는 시도를 하고 있습니다. 우리 시스템에서도 가능하면 단계를 간소화하여 지연(latency)을 줄일 것입니다.
- **ROS2 실시간 설정**: ROS2에서 실시간 성능을 높이기 위한 설정 (예: 리얼타임 커널, 스레드 우선순위 조정, DDS QoS 설정)과 **노드 구성 최적화**도 적용합니다. 영상 처리는 별도 노드에서 GPU를 활용하고, MoveIt2 등의 경로계획은 CPU 스레드에서 병렬로 수행하여 서로 간섭을 최소화합니다. 또한 Jetson에서 **NvJPEG**, **VPI** 같은 하드웨어 가속 라이브러리를 사용해 카메라 영상 전처리를 최적화합니다.

이러한 기술들을 종합하여 Jetson Orin 상에서 **인지-계획-제어 전체 파이프라인**이 원활히 동작하도록 하는 것이 목표이며, 이는 엣지 로봇의 **자율성**을 높이는 중요한 요소입니다. NanoOWL의 ROS2 패키지 활용이나, Grounded-SAM의 오픈소스 ROS2 통합 등을 적극 활용하고 필요 시 커스텀 노드를 개발하여, **ROS2 Humble** 기반 시스템에 최적화된 구조를 구현할 것입니다.

시스템 설계 (VLM 기반 파지 시스템 vs. 강화학습 기반 파지 시스템)

본 연구에서는 두 가지 상이한 접근의 파지 시스템을 각각 설계합니다. 하나는 **Vision-Language Model(VLM) 기반 파지 시스템**으로, 사전 학습된 AI 모델을 이용해 **물체 인식** 및 **파지 지점 결정**을 수행하고, **MoveIt2**를 활용해 로봇팔을 제어합니다. 다른 하나는 **강화학습(RL) 기반 파지 시스템**으로, **시뮬레이션**을 통해 로봇팔의 파지 정책을 학습시키고 이를 실로봇에 적용하는 구조입니다. 두 시스템 모두 하드웨어적으로는 Kinova Gen3 Lite 로봇팔과 RGB-D 카메라, Jetson AGX Orin을 사용하며, ROS2 Humble를 미들웨어로 합니다. 아래에서는 각 시스템의 아키텍처와 데이터 흐름을 설명합니다.

VLM 기반 파지 시스템 아키텍처

비전-언어 모델 기반 시스템의 전체 흐름은 **"인지 → 계획 → 실행"**의 단계로 구성됩니다. **그림 1**은 이 시스템의 개략적인 구성도입니다 (텍스트로 서술):

1) 카메라 센싱 및 전처리: RGB-D 카메라 (예: Realsense D435 또는 Azure Kinect)를 통해 **환경의 컬러 영상과 깊이 정보**를 획득합니다. 컬러 영상은 VLM 기반 인지 모듈로 입력되고, 깊이 정보는 탐지된 객체의 3D 위치 산출에 사용됩니다. 카메라로부터 받은 원시 이미지에 대해 왜곡 보정, 색상 보정 등의 전처리를 ROS2 노드에서 수행합니다.

2) 비전-언어 기반 객체 인식: 사전에 학습된 **오픈 어휘 객체 탐지 및 세분화** 모델을 사용하여, **관심 물체**의 픽셀 좌표와 마스크를 구합니다. 구체적으로, 사용자가 명령한 대상 객체의 **텍스트 프롬프트**를 VLM에 입력으로 주어, 해당 텍스트에 부합하는 물체를 이미지에서 찾습니다. 예를 들어 명령이 "빨간 컵을 잡아"라면, 텍스트 프롬프트 "red cup"을 사용하여 이미지 내 빨간 컵의 위치를 찾습니다. 모델 구성은 앞서 논의한 여러 후보 중 **NanoOWL + NanoSAM** 조합을 택할 예정입니다. NanoOWL이 **"red cup"**이라는 텍스트를 입력 받아 이미지에서 **해당 객체의 bounding box**를 예측하고, 이어서 NanoSAM이 그 박스를 입력받아 정밀한 **객체 마스크**를 생성합니다. 이때 NanoOWL/NanoSAM 모델은 모두 Jetson 상에서 TensorRT로 최적화되어 **실시간 추론**이 가능합니다 (NanoOWL ~95FPS, NanoSAM ~30FPS 성능). 만약 대상 객체가 다수 존재하거나 여러 후보가 인식되면, 가장 confidence가 높은 것을 선택하거나, 추가 조건 (예: "가장 큰 빨간 컵")이 있다면 이미지 후처리를 통해 조건에 맞는 객체를 고릅니다. 결과적으로 이 단계에서는 **타겟 객체의 픽셀 좌표 영역 (ROI)**과 **픽셀 마스크**가 산출됩니다.

3) 3D 좌표 계산 및 파지 지점 결정: 얻어진 객체의 2D 위치 및 마스크를 **카메라 좌표계의 3차원 좌표**로 변환합니다. 깊이 영상을 이용하여 ROI 영역의 평균 깊이나 마스크의 centroid 깊이를 추출하면 물체의 대략적인 3D 위치 (x, y, z)를 얻을 수 있습니다. 추가로, 마스크로부터 물체의 **orientation**을 추정하거나, **affordance** 정보가 있다면 (예: 컵의 손잡이 부위), 해당 부분의 좌표를 파지 목표로 선택합니다. 간단한 경우, 물체의 중심 상단을 향해 그리퍼를 내려잡는 **top-grasp** 전략을 취할 수 있고, 보다 복잡한 형태의 경우 별도의 **파지 포인트 산출 알고리즘**을 적용할 수 있습니다. 예컨대 **Grasp Pose Detection** 알고리즘이나 사전에 학습된 **Grasping Rectangle** 모델을 사용하여 해당 객체에서 최적 그립 지점을 찾고, 그 결과를 이용할 수 있습니다. 그러나 본 연구의 기본 시나리오는 일반적인 책상 위 물체이므로, 마스크 중심에 대한 top-down 파지를 1차 접근으로 활용하고, 필요 시 보조적인 개선을 추가할 것입니다.

4) MoveIt2 기반 모션 계획: 결정된 파지 지점(목표 3D 좌표 및 그립 방향)은 **ROS2**를 통해 **MoveIt2**에 전달되어 로봇팔의 모션 계획에 사용됩니다. MoveIt2는 ROS2의 표준 모션 플래닝 프레임워크로, 로봇의 역운동학, 경로계획, 충돌 검사 등을 담당합니다. VLM 모듈로부터 나온 파지 목표를 MoveIt2로 입력하는 방법은 여러 가지가 있으나, 가장 표준적인 방식은 **moveit_msgs/Grasp** 메시지를 활용하는 것입니다. 이 메시지에는 파지 전 그리퍼 자세, 파지 시 그리퍼 자세, 접근 방향, 파지할 목표 pose 등이 포함되며, 앞 단계에서 계산한 **파지 목표 위치/방향**을 grasp_pose 필드에 설정하고 그리퍼의 여닫음 상태를 pre_grasp_posture (열림) 및 grasp_posture (닫힘)에 명시하여 MoveIt2로 보냅니다. MoveIt2는 이를 받아 **접근 경로**와 **파지 후 후퇴 경로**를 함께 계획하며, 이 과정에서 **Planning Scene**에 대상 객체를 **Collision Object**로 추가하여 로봇이 물체를 잡을 때 물체와의 충돌(실은 파지이지만, 로봇 모델 상 충돌로 간주됨)을 허용하거나 적절히 처리합니다. Planning Scene 갱신을 위해 객체의 3D bounding box를 Collision Object로 설정해두면, MoveIt이 경로를 짤 때 그 객체를 인지하게 할 수 있습니다. 이러한 설정으로 **안전하고 충돌 없는 경로**를 찾아 그리퍼를 목표 지점까지 이동시키도록 계획됩니다.

5) 로봇 제어 실행: 계획된 경로가 도출되면, ROS2의 FollowJointTrajectory 액션 등을 통해 Kinova Gen3 Lite 로봇팔을 제어하여 그리퍼를 움직입니다. Kinova Gen3 Lite는 6자유도 경량 매니퓰레이터로, ROS2용 드라이버 (Kinova Kortex ROS2 패키지)를 통해 제어 명령을 받을 수 있습니다. MoveIt2가 산출한 trajectory를 **Joint Position or Velocity Command** 형태로 로봇에게 전송하여 순차적으로 관절들을 구동합니다. 목표 지점에 도달하면 그리퍼를 **폐합**하여 물체를 파지하고, 후퇴 경로를 따라 들어올립니다. 이러한 일련의 실행 과정 동안, 실시간으로 혹시 모를 오차에 대비해 카메라 피드백을 모니터링하며, 필요한 경우 중간에 경로 재계획(replanning)을 할 수 있도록 구성합니다 (다만 본 연구 기본 시나리오는 정적인 환경으로 간주하여 추가 replanning 없이 일회 실행함).

6) 완료 및 후속처리: 물체를 쥐어 성공적으로 들어올렸다면, 목표 높이까지 리프트한 후 동작을 종료합니다. 만약 파지에 실패했다면 (예: 물체를 놓쳤거나 미끄러짐), VLM 인지 모듈의 결과를 갱신하고 재시도하는 루프를 수행할 수도 있습니다. 예를 들어 한 번 실패 시 그리퍼 자세를 미세 조정하여 재시도하거나, 다른 파지 지점을 선택하는 휴리스틱을 둘 수 있습니다.

이 전체 과정에서 중요한 것은 **인지 모듈 (VLM)**과 **플래닝/제어 모듈 (MoveIt2)** 간의 원활한 통신과 **좌표계 정합**입니다. 카메라의 좌표로 얻은 3D 좌표를 로봇 베이스 좌표계로 변환해야 하며, 이를 위해 카메라-로봇 간 **Hand-Eye 캘리브레이션**이 선행되어야 합니다. 또한 ROS2 상에서 영상 인지 노드와 MoveIt2 노드가 **동기화**되어야 하므로, 메시지 타임스탬프와 TF 변환 관리가 필요합니다. 이러한 시스템 통합 이슈를 모두 해결함으로써, VLM 기반 로봇 파지 시스템이 **실시간으로 언어 지시에 따라 물체를 인지하고 집어낼 수 있도록** 하는 것이 설계의 핵심입니다.

강화학습 기반 파지 시스템 아키텍처

강화학습 기반 파지 시스템은 **훈련 단계**와 **실행 단계**로 크게 나누어 볼 수 있습니다. **훈련 단계**에서는 물체 파지 과제를 가상 환경에서 시뮬레이션하고, RL 알고리즘으로 로봇의 파지 정책을 학습시킵니다. **실행 단계**에서는 학습된 정책을 실 로봇에 이식하여 파지 작업을 수행합니다. 각 부분을 자세히 설명하면 다음과 같습니다.

1) 시뮬레이션 환경 구축 (Isaac Gym): 현실의 책상 위 물체 파지 시나리오를 모사하기 위해, NVIDIA Isaac Gym (또는 최근의 Isaac Sim 강화학습 환경)을 사용하여 **가상 환경**을 설정합니다. Kinova Gen3 Lite 로봇팔 모델과 여러 가지 가상의 물체 (박스, 공, 컵 등)를 시뮬레이터에 배치합니다. 물체들은 무작위 크기, 모양, 위치로 생성하여 **다양한 파지 상황**을 구성합니다. Isaac Gym의 GPU 가속을 활용하여 수백~수천 개의 병렬 환경을 동시에 실행함으로써, 로봇 정책 학습에 필요한 **다양한 경험을 병렬 수집**합니다. 이러한 병렬 환경에서는 각기 다른 초기 물체 배치, 마찰 계수, 조명 조건 등을 무작위화(domain randomization)하여, 학습된 정책이 특정 조건에 과적합되지 않고 **일반적인 파지 능력**을 갖추도록 합니다.

2) RL 문제 설정: 파지 과제를 강화학습 문제로 정의합니다. 에이전트(agent)는 로봇팔이며, 환경 상태(state)는 로봇 관절각도, 관절속도, 그리고 **시각 관찰**로 구성됩니다. 시각 관찰은 카메라 영상을 그대로 신경망에 입력으로 줄 수도 있고, 혹은 **PointCloud**나 **Depth image** 형태로 줄 수도 있습니다. 선택지로, 전면 카메라 RGB 이미지와 뎁스 이미지를 신경망에 stack하여 입력으로 활용하거나, 시뮬레이터 내 물체의 위치 정보를 직접 사용하는 것도 고려할 수 있습니다 (후자는 학습 난이도를 크게 낮추지만, Sim-to-Real을 위해서는 최종적으로 카메라 인풋을 사용하는 것이 바람직합니다). 행동(action)은 로봇팔의 관절 속도 또는 목표 위치 명령입니다. 6개 관절에 대해 연속적인 값 ($-\Delta\theta \sim +\Delta\theta$ 범위)으로 움직이거나, 또는 **엔드이펙터(end-effector)의 3D 이동 + 그리퍼 개폐** 명령을 직접 정의할 수도 있습니다. 본 연구에서는 직관적으로 학습되도록, **엔드이펙터**의 Δx, Δy, Δz 이동 및 Δroll, Δpitch, Δyaw (혹은 quaternion) 변화를 action으로 설정하고, 별도로 그리퍼 개폐 (이산 2값: 열림/닫힘)를 추가로 다루는 방식을 고려합니다. 이렇게 하면 학습이 로봇팔의 운동학을 직접 찾아가기보다, 손끝(E.E.) 위치를 어떻게 움직일지에 초점을 맞춰 **탐색 공간**을 줄일 수 있습니다.

3) 보상 함수 설계: 강화학습의 성능은 **보상 함수**에 크게 좌우되므로, 파지 과제에 적합한 보상 설계를 합니다. 기본적으로 물체를 **성공적으로 집었을 때** 큰 양의 보상 (예: +1)을 부여합니다. 구체적으로, 에피소드 내에서 그리퍼가 물체를 들어올렸을 때 +1의 보상을 주고 에피소드를 종료시킬 수 있습니다. 또한 학습을 촉진하기 위해 **부분 보상 shaping**을 활용합니다: 예를 들어, 그리퍼 끝과 목표 물체 사이 거리가 줄어들수록 작은 양의 보상을 주거나, 그리퍼가 닫혔을 때 물체에 어느 정도 접촉했으면 보상을

줘서 **파지 시도를 장려**합니다. 반대로, 로봇이 일정 높이 이상 내려갔는데도 물체를 잡지 못했거나, 물체를 밀어내 책상에서 떨어뜨렸다면 **페널티**를 부여합니다. 충돌(로봇팔이 책상이나 주변과 부딪힘) 역시 -1의 큰 페널티와 함께 에피소드를 종료시켜, 안전한 경로만 탐색하도록 유도합니다. 이러한 shaping은 학습 초기 난제를 완화하는 데 도움을 주지만, 잘못하면 정책이 애매하게 수렴할 위험도 있으므로 적절한 상수 값을 정하는 데 유의합니다.

4) 신경망 정책 및 알고리즘: 로봇의 정책을 근사할 신경망 구조를 정합니다. 일반적으로 **컨벌루션 신경망(CNN)**이 시각 입력을 처리하고, 물체와 로봇 상태를 결합한 feature를 **다층 퍼셉트론(MLP)**으로 처리하여 action을 출력합니다. 예를 들어 RGB-D 이미지를 입력으로 받는 ResNet 기반 CNN과, 로봇 관절 상태 (혹은 엔드이펙터 상대 위치)를 입력으로 받는 작은 MLP의 출력을 concatenation하여 다시 MLP로 처리, 최종 6DOF delta pose와 그리퍼 상태를 예측하도록 할 수 있습니다. 강화학습 알고리즘으로는 앞서 언급한 **SAC**를 1차 선택합니다. SAC는 off-policy 방법이므로 **리플레이 버퍼**를 사용해 과거 경험을 샘플링하며 학습할 수 있어 **시뮬레이션 병렬화**와도 잘 맞습니다. 또한 SAC의 entropy 보상으로 **탐색을 유지**하면서 **안정적 수렴**이 가능하리라 기대합니다. 비교를 위해 **PPO** 알고리즘도 병행 실험하여 성능을 측정합니다. PPO는 on-policy이지만, Isaac Gym같이 대량 병렬 환경에서는 배치 데이터를 크게 모아 업데이트함으로써 효율을 낼 수 있습니다. 두 알고리즘의 파라미터 (학습률, entropy 계수 등)는 기본값으로 시작해, **학습 곡선**을 관찰하며 튜닝합니다.

5) 학습 과정 모니터링: 수십만~수백만 step에 걸친 학습 동안, **에피소드 당 성공률**, **평균 에피소드 보상**, **그리퍼 사용 빈도** 등의 지표를 로깅하고 모니터링합니다. 목표는 시뮬레이터 상에서 성공률 90% 이상을 달성하는 정책을 얻는 것입니다. 만약 학습이 정체되면 보상 함수를 수정하거나, 신경망 구조를 변경하거나, 객체 초기 조건을 바꾸는 등 개입을 합니다. 또한 학습된 정책이 특정 유형의 물체에 치우쳐 성능을 내지 않는지, 예컨대 둥근 공만 잘 집고 얇은 판형 물체는 못 집는 것은 아닌지 등을 확인합니다. 그런 경우 해당 경우를 추가로 훈련 데이터에 반영하거나 curriculum을 조정하여 **균형 잡힌 성능**을 도모합니다.

6) 학습 정책의 실세계 적용 (Sim-to-Real): 시뮬레이션에서 얻어진 최종 정책은 로봇에 이식됩니다. 이때 바로 쓰기보다, **도메인 랜덤화**로 인해 시뮬레이터와 현실 간 차이를 줄였다고 하더라도, 일부 갭이 존재할 수 있음을 감안합니다. 예를 들어 시뮬레이터의 카메라 영상과 실제 카메라 영상의 노이즈/감마 차이나, 물체 물리 특성 차이 등이 있습니다. 이를 보완하기 위해 **파인튜닝(fine-tuning)** 단계를 둘 수 있습니다: 로봇 실제 환경에서 몇 차례 시도를 해보고 그 데이터를 가지고 policy를 미세 조정하거나, Human-in-the-loop으로 약간의 보정을 가할 수 있습니다. 하지만 가능하면 추가 학습 없이도 동작하도록, 처음부터 **현실감 있는 시뮬레이션**과 **충분한 랜덤화**를 적용했을 것입니다.

실행 단계에서, 학습된 policy는 기본적으로 **관찰 → 행동**을 매핑하는 함수이므로, ROS2 노드로 이식을 합니다. 카메라로부터 이미지를 받아 전처리 후 신경망에 넣고, 출력으로 로봇 제어 명령을 실시간 생성합니다. 이때 정책이 **초당 수십 Hz**로 출력할 수 있으므로, 제어 주기는 10~30Hz 수준으로 설정합니다. RL 정책은 *피드백 제어* 형태로 동작하므로, 실행 도중에도 카메라를 계속 참조하며 물체가 예상대로 따라오지 않으면 그에 맞게 경로를 수정해갑니다. 이는 앞서 VLM+MoveIt 방식이 **한 번 경로 계획 후 실행**하는 *개방루프*(open-loop)인 것과 대비되는 점입니다. RL 정책은 매 시점 센서 피드백에 반응하므로, 파지 시 물체가 움직이거나 위치가 약간 달라져도 추적하여 대응할 가능성이 높습니다.

7) 안전 및 종료 조건: 실 로봇에서 정책을 실행할 때도 안전장치를 둡니다. 예컨대 정책이 잘못되었을 경우 이상한 방향으로 관절을 움직일 수 있는데, 이를 위해 **소프트 리미트**나 **가상 울타리(virtual fence)**를 설정하고, 카메라로 모니터링하여 물체를 지나치게 밀거나 떨어뜨리면 사람이 개입하여 정지할 수 있도록 합니다. 또한 학습된 정책이 성공/실패 여부를 스스로 인지하지 못하므로, **후처리**로 그리퍼의 힘이나 물체의 이동 여부를 체크하여 성공 시 동작을 멈추고 실패 시 재시도하게끔 상위 레벨에서 제어합니다 (상위 레벨 제어는 간단히 rule-based로 구현 가능).

정리하면, 강화학습 기반 시스템은 **시뮬레이션 학습기**와 **실시간 제어기**의 조합으로 구성됩니다. 이는 VLM 기반 시스템에 비해 학습에 시간이 들지만, **로봇 제어의 저수준 세부까지 자동으로 최적화**된 정책을 제공하며, 특히 **센서-액추에이터 사이클을 빠르게 폐루프로** 돌릴 수 있다는 장점이 있습니다. 반면 특정 물체를 골라 잡는 등의 **고수준 인지 능력**은 부족하므로, 필요 시 VLM의 인지 모듈과 RL 정책을 결합하여 "이 물체를 잡아"라는 명령에 대응하는 **하이브리드 시스템**도 추후 모색해볼 수 있을 것입니다.

하드웨어 및 소프트웨어 구성 (Gen3 Lite, ROS2, MoveIt2, Jetson AGX Orin, RGB-D 카메라)

이 절에서는 본 연구에 사용되는 하드웨어와 소프트웨어 스택의 구성을 정리합니다.

- **로봇 플랫폼: Kinova Gen3 Lite** – Kinova사의 Gen3 Lite는 6-자유도 경량 로봇팔로, 최대 가반하중 약 2kg, 작동 범위 (Reach) 약 590 mm를 갖습니다. 비교적 컴팩트하며, 각 관절에 torque sensor가 있어 impedance 제어도 가능하지만, 본 연구에서는 주로 위치 제어를 사용합니다. 이 로봇팔은 **Kinova Kortex API**로 제어할 수 있고 ROS2 드라이버 패키지가 제공되어, ROS2 액션/토픽 인터페이스를 통해 **관절 제어** 및 **그리퍼 개폐** 명령을 줄 수 있습니다. 그리퍼는 2-finger 형태로 병진 개폐하며, 물체를 잡기에 적당한 힘 제어가 가능합니다.

- **컴퓨팅: NVIDIA Jetson AGX Orin** – 로봇 제어 및 AI 연산을 담당하는 온보드 컴퓨터로, 2022년 출시된 Jetson AGX Orin 모듈을 사용합니다. Orin은 12-core Arm Cortex CPU와 2048-core Ampere GPU (Orin 32GB 모델 기준), 그리고 2개의 NVDLA(딥러닝 가속기)를 포함하고 있어, 병렬 연산에 강점이 있습니다. 또한 10Gbps CSI 카메라 인터페이스 등을 지원하여 RGB-D 카메라를 직접 연결할 수 있습니다. 본 연구에서는 Jetson Orin에 **Ubuntu 22.04 LTS**와 **ROS2 Humble Hawksbill**을 설치하고, CUDA, TensorRT, cuDNN 등 GPU 가속 라이브러리를 세팅합니다. Jetson은 저전력으로 동작하기 때문에 발열 관리에 주의하며, **MAXN 모드**로 성능을 최대로 활용할 예정입니다.

- **RGB-D 카메라: Intel RealSense D435 (예시)** – 실험 환경에서 물체 인지를 위한 깊이 카메라로 Intel RealSense D400 시리즈를 사용합니다. 이 카메라는 1280x720 색상 영상과 정밀한 깊이 영상을 실시간 제공하며, ROS2용 드라이버 (`realsense2_camera` 패키지)가 있어 쉽게 토픽을 통해 영상을 획득할 수 있습니다. 깊이 센싱 범위는 0.2m ~ 10m 정도로 책상 위 물체 (~0.5m 거리)에는 충분한 정밀도를 제공하며 깊이 오차는 수 mm 수준입니다. 카메라는 로봇팔이 내려다보는 각도로 고정하거나, 로봇 베이스 혹은 외부 삼각대에 장착하여 책상 상부를 시야에 담도록 배치합니다. 카메라 좌표와 로봇 좌표의 정합(calibration)은 ROS2 tf 프레임워크에 statically 세팅하고, 필요 시 Hand-eye 캘리브레이션 절차를 통해 보정합니다.

- **ROS2 Humble 및 주요 패키지:** ROS2는 분산 시스템으로 각 컴포넌트를 노드로 실행하고 DDS를 통해 통신하므로, 우리 시스템의 **인지 노드**, **모션계획 노드**, **로봇드라이버 노드** 등이 동시에 동작합니다. 주요 패키지로 MoveIt2 (moveit_ros 플레임워크), realsense2_camera, rviz2 (시각화) 등을 사용합니다. 또한 앞서 언급한 NanoOWL, NanoSAM 등의 AI 모델을 구동하기 위해 **ros2_nanollm** 프로젝트의 일부를 활용할 계획입니다. `ros2_nanollm`은 NVIDIA에서 제공하는 ROS2용 LLM/VLM 패키지로, NanoOWL의 ROS2 wrapper (`ROS2-NanoOWL`)와 같은 노드를 포함하고 있어 VLM 추론 결과를 ROS 토픽으로 받을 수 있습니다. 만약 해당 패키지가 성숙하지 않다면, Python으로 ONNX 혹은 TensorRT 엔진을 불러오는 자체 노드를 작성할 것입니다. 강화학습 쪽으로는 Isaac Gym은 Python 환경에서 학습 단계에 사용되고, 실제 로봇 적용 시에는 학습된 모델을 불러오는 ROS2 노드를 작성해야 합니다. 이 부분은 PyTorch로 policy를 저장하고, ROS2에서 C++보다는 Python 노드로 불러와 실행할 가능성이 높습니다 (Jetson에서 Python GPU 실행도 가능하지만, GIL로 인해 멀티스레딩에 유의).

- **MoveIt2 & Planning**: MoveIt2는 별도 설정 파일(`.srdf`, `.yaml` 등)을 통해 Kinova Gen3 Lite의 URDF 모델과 Kinematics, Joint limits 등을 세팅하고 사용합니다. 또한 grasp 동작을 위해 MoveIt2의 **Grasp Execution Pipeline**을 활용하는데, 이는 인식된 객체를 가상으로 Planning Scene에 삽입하고, Grasp 메시지를 통해 plan을 생성한 후, 실행(Execution)까지 해주는 파이프라인입니다. MoveIt2는 기본적으로 OMPL (Open Motion Planning Library)의 RRT, PRM 알고리즘을 사용해 경로를 탐색하며, 필요시 제약조건 설정 (예: 엔드이펙터 자세 고정) 등을 적용할 수 있습니다.

- **기타 센서/구성:** 기본 시나리오에는 카메라 한 대와 로봇팔만 있지만, 향후 확장을 고려해 로봇 베이스(모바일 플랫폼)나 추가 센서(IMU 등)가 연동될 수 있습니다. 소프트웨어적으로는 이를 위해 Nav2 (자율주행 이동)나 SLAM 툴박스 등의 ROS2 패키지도 염두에 둘 수 있습니다. 그러나 본 연구의 범위에서는 주로 고정된 로봇팔+카메라 환경에 집중합니다.

以上 하드웨어/소프트웨어 구성 요소들을 통합하여, 실험을 위한 시스템을 구축합니다. 설치된 환경에서 각 노드들의 주기를 최적화하고, 데이터 흐름을 원활히 함으로써 **실시간 처리**와 **정확한 제어**가 모두 만족되도록 하는 것이 핵심입니다.

알고리즘 선택 및 비교 분석 (VLM 모델, RL 알고리즘)

이 절에서는 앞서 설계한 두 접근 방법에 사용될 **핵심 알고리즘들을 선정**하고, 그 특징을 비교합니다. 구체적으로 VLM 기반 시스템에 적합한 **시각-언어 모델 조합**과 RL 기반 시스템에 적합한 **강화학습 알고리즘**을 결정합니다. 또한 두 접근의 장단점을 종합 비교합니다.

VLM 모델 선정 및 구성

비전-언어 기반 파지에서 요구되는 기능은 (a) **텍스트 질의에 따른 객체 인식**과 (b) **해당 객체의 정확한 위치/형상 파악**입니다. 이를 만족하기 위해 **객체 탐지 모델**과 **세그멘테이션 모델**의 조합이 필요하며, 추가로 **언어 임베딩**을 처리할 수 있어야 합니다. 후보 기술로 1) CLIP 기반 방법, 2) OWL-ViT 기반 방법, 3) Grounding DINO 기반 방법 등을 검토했습니다. 각 접근의 장단점을 표로 정리하면 다음과 같습니다:

접근 방법 (모델 조합)	기술 특징	실시간성	ROS2 통합	장점 및 비고
NanoOWL + NanoSAM *(OWL-ViT + SAM 경량화 버전)*	제로샷 텍스트 기반 객체탐지 + 고속 세그멘테이션 (CLIP 계열 ViT 백본 활용)	**매우 높음** (탐지 ~95 FPS, 분할 ~123 FPS)	**용이** (공식 ROS2 패키지 제공)	Jetson에 최적화되어 **실시간 성능** 우수, 오픈 어휘 지원. ★ **추천 조합:** Jetson 상 탁월한 속도로 본 시스템에 최적.
Grounding DINO + MobileSAMv2	텍스트 프롬프트 기반 바운딩박스 예측 + 경량 세그멘테이션 (MobileSAM)	**중간~높음** (DINO 수 Hz, MobileSAM ~30 FPS 추정)	**보통** (ROS2 노드 커스텀 작성 필요)	**정확한 객체 위치**(6-DoF 포즈) 추정 가능, 세분화 품질 양호. 복잡한 형상 및 명시적 지칭에 강점.
MobileVLM v2 + MobileSAMv2 *(경량 멀티모달 + SAM)*	ViT + 경량 LLM 조합으로 복잡한 언어 명령 처리 + 세그멘테이션	**높음** (ViT/LLM 추론 수십 ms, SAM ~30 FPS)	**보통** (커스텀 노드 필요)	**긴 문장 이해** 등 언어처리 우수. 여러 객체 중 **상황에 따른 선택** (예: "가장 높은 병") 가능.
Grounded SAM (단일 파이프라인)	Grounding DINO + SAM 통합 (joint 모델) – **오픈소스 구현 이용**	**중간** (~5 FPS 수준 보고됨)	**양호** (ROS2 노드 존재 가능)	탐지와 분할이 **단일 모델**로 처리되어 구현 간결. 다중 객체 **동시 인식** 용이.

위 표에서 보듯이, **NanoOWL + NanoSAM 조합**이 Jetson Orin 환경에서 가장 **실시간 처리**에 유리하고, ROS2 통합도 수월하므로 우선적 선택입니다. 이 조합은 OWL-ViT 기반이어서 CLIP의 강력한 언어-시각 임베딩을 활용하며, SAM 기반 분할로 픽셀 정확도를 확보합니다. Grounding DINO + MobileSAMv2도 정확도 면에서 매력적이지만, NanoOWL 대비 속도가 떨어질 우려와 구현 복잡도가 있습니다. Grounded-SAM 등의 통합 모델은 아직 연구 프로토타입 수준으로 성능 검증이 충분치 않아, 부차적으로 고려합니다.

따라서 **최종적으로 VLM 기반 인지 모듈은 NanoOWL (텍스트 기반 탐지)과 NanoSAM (세그멘테이션)**으로 구성하고, 필요시 NanoOWL 대신 Grounding DINO를 fallback으로 사용할 수 있도록 할 것입니다.

추가로, **CLIP 자체를 활용한 접근**도 간단히 언급하면, CLIP 모델로 이미지의 지역 특징을 임베딩하고 텍스트 임베딩과 비교하여 해당 물체를 찾는 **제로샷 탐지** 연구들이 있습니다 (예: RegionCLIP, ZegCLIP 등). 그러나 이런 접근은 2-stage (지역 제안 + 임베딩 비교)로 속도가 느릴 수 있고, 구현 복잡성이 높아, 완성도 있는 오픈소스인 OWL-ViT나 Grounding DINO 쪽을 선택했습니다.

강화학습 알고리즘 및 설정 비교

강화학습 부분에서는 **SAC vs. PPO** 두 알고리즘을 중점적으로 고려하고 있습니다. 이 둘은 로봇 제어에 많이 사용되는 대표 알고리즘으로, 각기 장단점이 있습니다:

- **SAC (Soft Actor-Critic)**: Off-policy 최대 엔트로피 RL 알고리즘입니다. 경험을 효율적으로 재사용하여 **표본 효율(sample efficiency)**이 높고, Q함수 학습을 병행해 **안정적**인 성능을 보이는 경우가 많습니다. 또한 자동 온도 조절(α 파라미터)로 탐험-이용 균형을 자동으로 맞춰줍니다. 단, Off-policy이므로 하이퍼파라미터에 민감할 수 있고, 정책과 가치함수 신경망 학습이 복잡하여 튜닝이 어렵다는 지적도 있습니다. 로봇 연속제어 문제에서 SAC는 DDPG, TD3 등을 제치고 좋은 성과를 내는 것으로 보고되었습니다.
- **PPO (Proximal Policy Optimization)**: On-policy 정책경사 기법으로, trust-region 개념을 접목하여 update의 안정성을 높인 알고리즘입니다. 샘플 효율은 SAC보다 낮을 수 있지만, 단순한 구조와 적은 하이퍼파라미터로 **현장에서 많이 활용**됩니다. 병렬 환경에서 대량의 샘플을 모아 학습할 경우 충분한 성능을 낼 수 있고, 특히 이산 동작이나 제한된 범위의 연속 동작에서는 SAC 못지않게 성과가 좋습니다. 다만 exploration 측면에서 entropy 보상을 인위적으로 조절해야 할 수 있고, sparse reward 문제에서 학습이 느릴 수 있습니다.

본 연구의 파지 과제는 연속 제어 + sparse reward 특성이 있으므로, **SAC**가 유리할 것으로 예상합니다. 시뮬레이션 속도도 매우 빠르기 때문에 샘플 효율이 다소 낮아도 PPO로 대응 가능하지만, 정책의 품질을 높이려면 SAC를 기본으로 선택합니다.

또 다른 고려사항으로, **학습 파이프라인**을 결정해야 합니다. Isaac Gym에서는 병렬 환경으로 on-policy PPO를 학습하는 예제가 풍부하지만, off-policy 알고리즘 적용도 가능합니다. Off-policy 학습 시 하나의 GPU에서 4096개 환경을 돌리면서 경험을 쌓고 학습하면, 1억 step 수준까지도 몇 시간~하루 내로 가능하다고 보고됩니다. 이 정도면 충분히 로봇 파지 정책을 학습시킬 수 있을 것으로 보입니다. 또한 **학습 안정화 기법**으로 reward normalization, observation normalization 등을 적용하여 학습 초반 수렴을 돕습니다.

비교군으로 **전통적 파지 기법**과도 장단점을 분석해볼 수 있습니다. 딥러닝 기반 **파지 검출(Grasp Detection)** (예: GG-CNN 등) 방법은 한 장의 RGB-D 영상에서 바로 그립 위치를 회귀하여 제시하는 방식인데, 이는 **지도학습**으로 특정 데이터 분포에 최적화되며, 우리 연구의 VLM이나 RL 방식과는 접근이 다릅니다. Grasp Detection은 현재도 산업 로봇에 널리 쓰이며 구현이 비교적 간단하지만, **새로운 목표(특정 물체 선택)** 같은 요구에는 약합니다. 반면 VLM 방식은 **목표 지정의 유연성**이 높고, RL 방식은 **상황 대응 및 최적화된 동작** 측면에서 강점이 있습니다.

아울러, VLM과 RL 접근의 **내재된 장단점**을 요약 비교하면 다음과 같습니다:

- **일반화 측면:** VLM 기반은 CLIP 등의 사전학습 덕분에 **미학습 객체**도 인식 가능하고, 언어로 지칭만 하면 잡을 수 있는 **범용성**이 있습니다. 반면 RL 기반은 학습한 물체들(혹은 그 형태 유사체)에 대해서만 성공률이 높으며, 새로운 유형의 물체나 전혀 다른 모양에는 학습을 다시 해야 할 수 있습니다.
- **실시간 성능:** RL 정책의 추론은 단순 신경망 forward이므로 매우 빠르고 (수 ms 이하), 센서 수신부터 제어 출력까지 짧은 주기로 돌릴 수 있습니다. VLM 기반은 인지 단계에 복잡한 모델이 있어 수십 ms~수백 ms 이상 걸릴 수 있고, 또 MoveIt 경로계획도 수백 ms 정도 소요될 수 있어, **즉각 대응성**은 RL보다 떨어집니다.
- **정확도와 성공률:** 이는 구현에 따라 다르지만, VLM 기반은 인지 정확도가 높다면 적절한 파지 지점을 선택할 가능성이 높습니다. 다만 파지 동작 자체는 MoveIt의 **기하학적** 계획에 의존하므로, 미세한 조정이나 실패 시 재시도 등의 **적응적 스킬**은 부족할 수 있습니다. RL 기반은 학습된 정책이 때때로 **예상 밖의 전략** (예: 물체를 굴려서 잡기)을 구사할 수 있고, 실패 시 바로 다른 접근을 시도하는 **유연성**이 있습니다. 그러나 학습이 충분치 않으면 엉뚱한 곳을 잡으려 하거나 불안정하게 잡을 위험도 있습니다.
- **개발 난이도:** VLM 방식은 강력한 사전 학습 모델들을 사용하지만, 그것들을 **로봇 시스템에 통합**하는 공학적 작업이 필요합니다. 반면 RL 방식은 학습 파이프라인 구축과 시뮬레이터 설계에 큰 노력이 들고, 특히 보상 튜닝 등이 **시행착오 과정**을 많이 요구합니다. 둘 다 난이도 측면의 도전이 다르나, RL 쪽이 **계측/평가/반복**이 오래 걸린다는 점에서 연구 개발 일정에 영향을 줄 수 있습니다.

이러한 점을 고려하여, 본 연구에서는 **VLM 기반 방법**과 **RL 기반 방법** 모두를 구현/실험함으로써, 각 접근의 실제 성능을 체계적으로 비교 평가할 계획입니다. 이는 해당 분야 연구에 기여하는 바도 있으며, 궁극적으로 두 접근의 **장점을 결합**하는 방향 (예: VLM으로 목표 인식 + RL 정책으로 제어)도 모색해볼 수 있을 것입니다.

VLM 기반 Grasping 방법론 (MoveIt2 연동 중심)

이 장에서는 Vision-Language Model을 활용한 파지 시스템의 **구체적인 방법론**을 상세히 설명합니다. 앞서 시스템 설계 장에서 전체적인 파이프라인을 개관하였으므로, 여기서는 특히 **MoveIt2와의 연동 방식**과 **핵심 알고리즘/모듈의 동작**을 단계별로 명확히 기술하고자 합니다.

객체 인식 및 파지 후보 결정

VLM 기반 grasping의 성능은 첫 단계인 **객체 인식 결과**에 크게 좌우됩니다. 따라서 신뢰도 높고 정밀한 객체 인식을 위해 **다단계 인지 알고리즘**을 적용합니다.

1. **텍스트 프롬프트 입력:** 사용자로부터 또는 상위 시스템으로부터 **파지할 대상**을 나타내는 명령 문장이 주어집니다. 예) "잡지책을 집어 들어" 또는 "빨간색 공을 집어." 여기서 주요 키워드를 추출하여 **텍스트 프롬프트**로 사용합니다. 위 예에서 "잡지책", "빨간색 공" 등이 프롬프트가 됩니다. 만약 문장이 복잡하여 추가 단어 (예: "오른쪽에 있는") 등이 포함되면, 이를 파싱하여 조건으로 활용할 수도 있으나, 1차 버전에서는 단순 객체명 위주로 처리합니다.

2. **오픈 어휘 객체 탐지 (NanoOWL):** 추출된 텍스트 프롬프트를 NanoOWL 모듈에 입력합니다. NanoOWL은 CLIP과 동일한 ViT 백본을 이용해 이미지 전체를 **한 번에 처리**하면서, 주어진 텍스트에 해당하는 **바운딩 박스**를 출력합니다. 이때 confidence score도 함께 나오며, 만약 top-1 결과의 신뢰도가 임계치보다 낮다면 "대상 객체 없음"으로 처리할 수 있습니다. NanoOWL이 여러 객체를 탐지할 경우 (예: 동일 분류의 물체 여러 개), top-N 박스를 출력받습니다.

3. **세그멘테이션 (NanoSAM):** 탐지된 ROI에 대하여 NanoSAM을 적용합니다. NanoSAM (또는 MobileSAMv2)은 바운딩 박스 또는 해당 영역의 포인트를 프롬프트로 받아 해당 객체의 **픽셀 정밀 마스크**를 생성합니다. 마스크 획득으로 객체의 **정확한 윤곽과 픽셀 단위 위치**를 파악할 수 있습니다. 만약 NanoOWL에서 여러 후보를 받은 경우, 각 후보별로 NanoSAM을 수행하여 모든 마스크를 얻습니다.

4. **3D 포인트 산출:** 각 마스크에 대해, 카메라의 뎁스 데이터를 조회하여 마스크 영역의 **3차원 좌표들**(point cloud)을 얻습니다. 이를 통해 물체의 **중심 좌표 (x, y, z)**를 계산하고, 또한 **표면 법선 방향**을 추정합니다 (마스크의 convex hull 등에 대해 PCA 수행 등으로 근사). 기본적으로 물체의 표면 노멀 방향이 위쪽을 향하면 top-grasp가 용이하나, 책과 같이 평평이 놓인 경우 옆면을 잡아야 할 수도 있습니다. 이때 물체의 기울기 등을 고려해 잡기 좋은 방향을 heuristics로 결정합니다. 예를 들어 공이나 컵같이 윗면이 열려있지 않은 물체는 윗면 어느 방향이나 비슷하므로 top-down, 책이나 마우스같이 납작한 물체는 옆면을 잡도록 파지 방향을 결정합니다.

5. **후보 파지 지점 추출:** 마스크와 3D 정보로부터 **후보 파지 지점**을 정합니다. 가장 단순히는 물체의 **겉 중심부**를 그리퍼 중앙이 가도록 하는 접근입니다. 그리퍼의 너비는 물체 크기에 따라 조절해야 하므로, 마스크의 bounding box 크기를 픽셀 단위로 측정한 뒤 카메라 깊이를 활용하여 실제 물체 폭을 추정합니다. 이 폭보다 약간 큰 수준으로 그리퍼를 벌려서 접근하게 하면 기본적인 파지가 됩니다. 그러나 모든 물체를 이렇게만 하면 놓칠 수 있으므로, **특정 형태별 세부 규칙**을 추가합니다:

   - 긴 막대형 물체(예: 펜)의 경우 중앙보다 한쪽 끝을 잡는 것이 안정적이므로, 마스크를 긴 방향으로 2등분하여 1/4, 3/4 지점을 후보로 고려합니다.
   - 컵이나 주전자처럼 **손잡이**가 있는 물체는, CLIP 등으로 손잡이 부분을 인식해 해당 부위를 파지점으로 삼습니다. (예: "컵 손잡이"를 텍스트로 주어 NanoOWL→NanoSAM 수행하면 손잡이 마스크를 얻을 수도 있습니다).
   - 책이나 얇은 판형 물체는 위에서 집기 어려우므로 측면을 파지점으로 선택하며, 이때 로봇 End-effector pose도 옆에서 접근하는 자세를 취하도록 설정합니다.
   - 공처럼 둥근 물체는 어느 방향이나 유사하므로, 그냥 top-down으로 중심을 잡되, 미끄럼 방지를 위해 최대한 중앙을 관통하도록 alignment합니다.

   이러한 규칙 기반 보완은 추후 ML 기반 Grasp Planner로 대체할 수 있으나, 현 단계에서는 안전빵으로 적용합니다.

6. **최종 파지 대상 선정:** 만약 인식 결과 다수의 객체 (예: 동일한 컵 2개)에서 후보 파지 지점이 나왔다면, **사용자 명령**이나 **상황 맥락**을 고려하여 하나를 선택합니다. 예를 들어 "왼쪽 빨간 컵 잡아" 같은 명령에서는 이미지 좌우를 분석해 더 왼쪽에 있는 것을 선택합니다. 맥락이 모호하면 임의로 하나를 고르되, 사용자에게 확인 (confirmation) 절차(해당 안하지만 미래 시스템에선 필요)를 둘 수도 있습니다.

이상으로, 하나의 대상 객체와 그에 대한 **구체적인 파지 지점 (3D 좌표 + 그립 자세)**이 결정되었습니다.

MoveIt2를 통한 파지 동작 계획 및 실행

MoveIt2 연동은 앞서도 다뤘듯이, **Grasp 메시지**와 **Planning Scene**을 사용하는 것이 핵심입니다. 이를 구현 단계별로 기술합니다.

1. **Planning Scene 설정:** ROS2에서 MoveIt2를 구동하면 기본적으로 로봇 URDF 기반 장면이 생성됩니다. 여기에 인식된 객체를 Collision Object로 추가합니다. 구체적으로, `moveit_msgs::CollisionObject` 메시지에 객체의 primitive shape (예: box)나 mesh를 넣어 게시하면 MoveIt2가 장면에 반영합니다. 우리 경우 정확한 물체 모델이 없으므로, **바운딩 박스 크기의 직육면체**로 근사 추가합니다. 이는 로봇팔이 파지 시 자기 손(그리퍼)과 물체의 충돌을 무시하도록 설정하기 위해인데, MoveIt2의 grasp plan 시 CollisionObject와 그리퍼 링크 간 충돌은 grasp 시 허용으로 설정할 수 있습니다. 한편, CollisionObject를 추가해두면 파지 후 물체를 들었을 때 MoveIt이 그 물체와 주변 환경 충돌도 고려하게 되어, 더 안전한 후퇴 경로를 생성합니다.

2. **Grasp 메시지 구성:** `moveit_msgs::Grasp` 타입의 메시지를 채웁니다. 주요 필드로:

- `grasp_pose`: 로봇이 실제 파지할 목표 자세 (position + orientation, `PoseStamped`). 앞서 결정한 3D 파지 지점과 접근 orientation을 여기에 넣습니다. 좌표는 **세계좌표계** 기준으로 변환한 값이어야 합니다 (카메라→로봇 base의 TF 변환 적용).
- `pre_grasp_approach` 및 `post_grasp_retreat`: 접근 및 후퇴 경로를 지정합니다. 보통 접근 방향은 grasp_pose를 향해 **앞쪽으로 10cm 거리** 정도 떨어진 지점을 시작점으로 하고, 후퇴는 반대로 위쪽으로 10-15cm 이동 등으로 지정합니다. 이 길이와 방향은 상황에 맞게 조절하되, 기본적으로 수직 위 방향(z+)으로 후퇴하도록 설정합니다.
- `pre_grasp_posture`: 파지 전 그리퍼 모양, 즉 벌어진 상태의 JointState. Kinova 그리퍼 관절 각도를 최대 벌림 각도로 설정.
- `grasp_posture`: 파지 시 그리퍼 모양, 즉 닫힌 상태 JointState. 물체를 잡았을 때의 힘 제어가 필요하지만 MoveIt 수준에서는 각도만 지정 (예: 완전 닫힘에 가깝게).
- `grasp_quality`: 이 grasp의 점수 (0~1). 하나만 줄 경우 1로 설정.

이러한 Grasp 메시지를 하나 원소로 가진 vector를 MoveIt2의 `pick()` 함수에 넘기면, MoveIt2가 내부적으로 위 정보를 활용하여 trajectory를 계산합니다. MoveIt2는 자체적으로 IK를 풀어 grasp_pose에 도달 가능한지 확인하고, 여러 관절 해 중 가장 적절한 것을 택합니다.

3. **Trajectory Planning:** MoveIt2가 제공하는 Planner (OMPL 등)로 Grasp trajectory를 생성합니다. Plan 시 고려사항:

   - Pre-grasp 위치에서 grasp_pose까지 **직선 경로**(approach)와, grasp 후 post_grasp까지 **후퇴 경로**를 우선적으로 맞추려 시도.
   - 위 과정에서 CollisionObject(물체)를 grasp 시에는 허용하도록, MoveIt2의 AllowedCollisionMatrix를 조정 (즉, 그리퍼와 해당 CollisionObject 간 충돌 무시).
   - 다른 충돌 (책상, 로봇자신)은 회피.
   - 제한 시간 (planning_time)을 1~2초로 설정 (실시간성을 위해 너무 오래 탐색하지 않도록).
   - 성공 못하면 재시도 횟수 (`max_planning_attempts`)를 3회 정도로 설정하여 몇 번 더 시도.

   이러면 대체로 feasible한 path를 찾을 수 있습니다. 만약 IK 불가 등 실패하면, grasp_pose 자체를 수정해야 합니다 (다른 approach 각도 등). 이 경우 VLM 인지 단계에서 orientation 후보를 여러 개 넣어둘 수도 있지만, 우선 simple approach로 충분할 것으로 가정합니다.

4. **Trajectory 실행:** Plan 성공 시, MoveIt2의 MoveGroupInterface를 통해 trajectory 실행 명령을 보냅니다. 그러면 ros2_control을 통해 Kinova 드라이버가 각 관절을 시퀀스대로 움직이게 됩니다. 이 동안 피드백으로 현재 로봇 상태를 모니터링하며, 만약 중간에 문제 발생 (충돌 감지, 목표 미도달)하면 정지합니다.

5. **Gripper 동기화:** MoveIt2의 plan에는 그리퍼 동작도 타임라인에 포함될 수 있습니다 (grasp_posture가 특정 시점에 닫히도록). 하지만 Kinova 그리퍼는 속도가 느리므로, 보통 grasp_pose 도달 직전에 그리퍼 닫힘을 트리거하고, 충분히 잡을 시간을 잠시 둡니다. MoveIt2의 `pick()` 함수는 이 과정을 자동화하지만, 세밀히 제어하려면 trajectory 중간에 서버 측에서 gripper action을 호출해야 할 수도 있습니다. 실험을 통해 타이밍을 조정할 예정입니다.

6. **파지 완료 처리:** 로봇이 물체를 잡고 후퇴 위치까지 이동 완료하면, pick 동작이 끝난 것입니다. 이후 상위 로직에서 "성공" 상태를 기록하고, 필요하면 후속 동작 (예: 특정 위치에 내려놓기)으로 넘어갑니다. 만약 pick 실패 (예: MoveIt plan 실패, 혹은 잡았는데 물체를 놓침)하면, MoveIt2는 에러를 리턴하거나 로봇 센서(그리퍼 폭 변화) 등을 통해 알 수 있습니다. 이때 VLM 인지 재시작 → pick 재시도를 할 수 있습니다.

정리하면, VLM 기반 grasping 방법론은 주로 **정확한 인지**와 **기계적 계획**으로 구성됩니다. 이는 이미 검증된 MoveIt2 라이브러리를 활용하기 때문에 모션 안정성은 높지만, 인지가 틀리면 아무 소용없고, 또한 정적인 목표에 대해서만 동작한다는 제한이 있습니다. 그러나 본 연구 범위인 책상 위 정적인 물체 파지에는 잘 부합하며, 특히 **언어 지시에 따른 임의 물체 파지**라는 목표 기능을 충실히 수행할 수 있을 것입니다.

추가로, VLM 기반 방법론의 변형으로 **affordance 인식**을 응용하는 아이디어도 있습니다. 예를 들어, CLIP으로 "잡을 곳"을 이미지 내 찾아내도록 훈련하거나, 또는 Semantic Segmentation 모델 (Semantic-SAM 등)로 물체의 부위별 마스크를 얻어, 그중 "잡기 좋은 부분"을 선택하는 것입니다. 이는 아직 연구 단계지만 향후 파지 성공률을 높이는 방향으로 고려될 수 있습니다.

강화학습 기반 Grasping 방법론 (Isaac Gym 및 RL 세팅)

이번 장에서는 강화학습 기반으로 로봇 파지 정책을 학습하고 적용하는 구체적인 절차와 기법들을 다룹니다. 앞서 시스템 설계 및 알고리즘 비교 부분에서 개요를 제시했으므로, 여기서는 **Isaac Gym 환경 설정**, **학습 과정 상세**, **성능 향상 기법**, 그리고 **실제 적용 시 고려사항**을 중심으로 기술합니다.

시뮬레이션 환경 세부 설정 (Isaac Gym)

Isaac Gym을 사용하여 수천 개의 병렬 환경을 구동하기 위해, 먼저 파지 환경을 하나 정의하고 이를 다중 복제하도록 합니다.

- **로봇 모델 불러오기:** Kinova Gen3 Lite의 URDF 또는 USD 모델을 로드합니다. Isaac Gym에는 예제로 Franka Emika Panda, UR5 등 몇 가지 로봇이 있지만 Kinova는 custom 추가가 필요합니다. URDF를 파싱해 물리 속성을 세팅하거나, Gym API로 geometry와 관절 제약 등을 정의합니다. 로봇 베이스를 world에 고정하고, 각 관절에 모터(토크 제어) 또는 position drive를 설정합니다. 그리퍼는 1-DOF (두 finger 링크 연동)로 모델링하여, 닫힘 시 충분한 힘으로 물체를 잡도록 effort 한계를 설정합니다.

- **물체 생성 및 배치:** 다양한 **훈련용 객체**를 준비합니다. 단순하게는 primitive shape (박스, 구체, 원기둥)을 사용하고, 추가로 YCB dataset 같은 공개 3D 모델(컵, 병, 도구 등)을 일부 활용할 수 있습니다. Isaac Gym에서는 기본 도형은 쉽게 생성 가능하며, 복잡한 mesh는 물리엔진 상 계산 비용이 조금 증가할 수 있으나 허용 범위입니다. 각 environment마다 하나의 물체를 테이블 위에 놓는데, 초기 위치는 로봇 기준 좌표계에서 x: 0.3~0.5m, y: -0.2~0.2m (팔 정면 범위 내 랜덤), z: 테이블 높이(~0.0m) 위로 살짝 놓습니다. 초기 orientation도 무작위 설정합니다. 테이블은 얇은 판으로 static 설정.

  - **도메인 랜덤화:** 각 에피소드마다 물체의 크기(예: 5~15cm), 질량(0.1~2kg), 마찰계수(0.5~1.0) 등을 랜덤화합니다. 또 조명, 카메라 각도도 환경마다 다르게 하여, 시각 입력 다양성을 확보합니다.
  - **멀티-객체 시나리오:** 추후 확장으로 한 환경에 2~3개의 물체를 둘 수도 있지만, 그러면 정책이 어느 것을 집어야 할지 모호해집니다. 따라서 기본은 한 번에 하나의 물체만 놓고 집도록 합니다.

- **관찰 (Observation) 설계:** 에이전트에게 주는 관찰 정보는 몇 가지 모달리티로 구성됩니다.

  1. 로봇 자체 상태: 6개 관절 각도 및 속도 (필요시 end-effector pose로 변환하여 사용).
  2. 시각 관찰: 전면 카메라 이미지 (앞쪽 45° 위에서 로봇과 테이블을 함께 보는 시야). RGB 128x128 또는 256x256으로 제공. 심층 강화학습에서 고해상도는 불필요하게 연산을 늘리므로 적당히 축소. 딥마인드 QT-Opt의 경우 어깨 너머 카메라 RGB를 472x472 grayscale로 사용했습니다; 우리도 필요에 따라 grayscale이나 depth만 활용하는 방안도 고려합니다. Depth map은 물체와 테이블 구분이 쉬워 유용할 수 있지만, 잡으려는 대상 하나만 있는 경우 RGB도 충분합니다. 일단 **RGB-D 둘 다** 스택하여 CNN 입력으로 넣을 계획입니다.
  3. 목표 정보: 우리 작업은 "아무 물체나 집어"이므로 특정 목표 ID는 없습니다. 만약 여러 물체 중 특정 하나만 잡도록 목표를 지정하려면, 관찰에 목표의 one-hot 또는 언어 embedding 등을 넣어줘야 하지만, 여기서는 불필요합니다.

관찰 텐서는 위 요소들을 정규화하여 하나로 합칩니다. 관절 각도 등은 [–1,1]로 normalize, 이미지 픽셀도 [0,1] 또는 [-1,1] 스케일로. 최종 관찰 벡터 크기는 (C x H x W + joint_dim), 예: RGB-D (4x128x128=65,536) + 12 ≈ 65,548 차원. CNN에서 convolution으로 차원 감소를 하므로 실제 네트워크 파라미터 수는 감당 가능 수준입니다.

- **행동 (Action) 설계:** 앞서 설명한 대로 end-effector 3D delta pose + gripper open/close로 설계합니다. 그러나 시뮬레이션에서는 굳이 delta pose가 아니어도, 6관절 모터 torque를 직접 출력하는 것도 가능합니다. 하지만 torque 제어는 학습 난이도가 높고, 위치 기반 제어보다 불안정할 수 있습니다. 반면 delta pose 방식은 일종의 Cartesian impedance control처럼 동작하여, 정책이 안정적으로 공간에서 동작하게 합니다. 구현적으로는 action 6개를 받아 이를 IK를 통해 joint velocity 명령으로 변환합니다. 또는 6개 관절 속도를 직접 action으로 하는 것도 고려됩니다.
    - Gripper action: 1개 이산값 (0=open, 1=close)로 간주해도 되고, continuous 0~1로 출력하여 thresholding해도 됩니다. RL에서는 이산이 다루기 쉽기 때문에, multi-head output으로 actor의 일부는 gripper open/close 확률을 softmax로 내보내 2-class 분류처럼 취급합니다. Critic 네트워크는 연속+이산 혼합 action도 다룰 수 있도록 구성해야 하므로, SAC의 경우 gripper를 continuous 0~1로 두고 reward 계산만 별도로 해주는 편이 단순할 수 있습니다.
- **시간 간격 및 에피소드 구성:** 시뮬레이션의 1 timestep당 dt를 0.05~0.1초로 설정합니다. RL 알고리즘의 action 빈도와 일치하게 됩니다. 각 에피소드 길이는 최대 3~5초 (즉 60~100 steps)로 두고, 그 안에 집으면 성공, 못 집으면 실패로 종료합니다. PyTorch 기반 학습 시 4096 env * 100 steps = 409,600 transitions per batch rollout, 상당히 많지만 GPU 상에서 한 번에 계산하므로 충분히 처리 가능합니다. 이렇게 대량 병렬 roll-out으로 신속히 경험을 축적합니다.

강화학습 학습 과정 상세

환경이 준비되었으니, SAC 알고리즘을 실제 적용하는 과정을 설명합니다:

1. **초기화:** Actor 네트워크 π(s), Critic 네트워크 Q(s,a) 두 개 (또는 Double Q로 2개) 가중치를 랜덤 초기화합니다. 리플레이 버퍼 생성 (용량: 1e6 transitions 등).
2. **탐색 초기화:** 초기 정책은 랜덤이므로, 우선 일정 시간 epsilon-greedy 혹은 높은 탐색 노이즈로 rollouts을 수행합니다. 이때 대부분 실패하지만, 다양한 시도가 버퍼에 쌓입니다. 물체를 전혀 못 잡는 상태라 reward=0인 에피소드가 많을 것이므로, shaped reward (거리 짧아지면 +r 등)로 약간의 gradient가 생기게 합니다.
3. **경험 수집:** 각 병렬 환경에서 policy에 따라 action을 수행하고, 그 결과 next state, reward, done을 얻어 버퍼에 저장합니다. 이 과정을 GPU 상에서 벡터화하여 매우 빠르게 처리합니다. Isaac Gym에서는 step마다 모든 환경 관찰을 tensor로 업데이트하고, policy inference도 tensor 연산으로 수행합니다 (PyTorch 등으로 구현).
4. **정책 업데이트 (SAC):** 일정 수의 step마다 (예: 매 4 step) SAC의 gradient update를 실행합니다. 버퍼에서 미니배치 (예: 1024 샘플) 랜덤 추출 -> Q-loss 계산 (벨만: r + γ * min(Q_target(s', π(s'))) - Q(s,a)) -> 업데이트; 그리고 policy loss 계산 (entropy regularized: ∇θ [α * logπ(a|s) - Q(s,a)]) -> 업데이트. 타겟 네트워크 soft update. 이러한 업데이트를 한 iteration에 여러 번 (예: 2~4회) 수행합니다. 이때 GPU 병렬로 워낙 많은 데이터를 쌓기에, 일반적으로 environment step 수 >> update step 수가 됩니다.
    - **학습률 및 안정화:** 초기엔 exploration이 중요하므로, α (entropy weight)를 높게 유지하거나, Gaussian noise를 policy output에 추가합니다 (SAC는 본래 entropy로 exploration하지만, entropy 타겟팅이 진행됩니다). 학습이 진행되며 α가 자동 조절되면서 exploitation 쪽으로 기울어집니다.
5. **평가:** 주기적으로 (예: 매 10k step) 현재 policy로 deterministic하게 여러 에피소드를 시뮬레이션 돌려보고 성공률을 계산하며, TensorBoard 등에 로깅합니다. 또 reward 평균, Q-value 평균, policy entropy 등의 추이를 기록해 학습 상황을 파악합니다.
6. **수렴 판단:** 성공률이 일정 기준 (예: 90%)을 넘고 plateau가 지속되면 학습 종료를 판단합니다. 또는 지정된 최대 step (예: 5 million steps) 도달 시 중지합니다. 충분히 수렴했다면, Actor 네트워크 파라미터를 파일로 저장합니다.
7. **정책 개선 (선택사항):** 만약 특정 object에서 실패율이 높거나 정책 편향이 보이면, 그 사례만 모아 fine-tune하거나, 보상 구조를 수정하는 등의 추가 조치를 고려합니다. 그러나 과도한 fine-tune은 다른 성능을 저해할 수 있으므로, 균형을 맞춥니다.

성능 향상 및 보정 기법

학습을 원활히 하거나 정책 성능을 높이기 위한 추가 기법들:

- **휴리스틱 Pre-grasp 도입:** 완전히 처음부터 학습하기 어렵다면, 초반 몇 에피소드에서 물체 근처까지 로봇을 움직이는 기본 동작을 수동 제공하고 그 시점부터 학습하게 할 수도 있습니다. 이를 Curriculum Learning의 초기 단계로 볼 수 있는데, 첫 단계부터 최종 목표를 요구하기보다, 점진적 목표를 부여해 학습하는 방식입니다.
- **Hindsight Experience Replay (HER):** 에피소드가 실패로 끝나더라도, 마지막에 로봇이 어떤 위치까지 손을 뻗었다면, 그걸 목표로 하는 가상의 성공으로 간주하여 경험을 버퍼에 추가하는 방법입니다. 그러나 여기서는 목표가 특정 물체이므로 일반적인 HER은 부적합하고, 대신 "결과적으로 잡힌 물체가 목표"라고 post-hoc 가정하여 학습에 활용하는 변형된 HER을 고려할 수 있습니다.
- **다중 작업 학습:** 향후 여러 종류 객체별로 정책을 조금 다르게 해야 할 필요가 생기면, 하나의 거대 정책 대신 object embedding을 입력으로 받아 동적으로 다른 동작을 하는 정책을 학습하는 방법도 연구되고 있습니다 (본 범위에서는 제외).
- **Sim-to-Real 적응:** 도메인 랜덤화는 기본이고, 추가로 **System Identification** 기반으로 시뮬레이터 파라미터 (마찰, 질량 등)을 실세계와 맞추는 노력, 또는 **Visual Domain Randomization** (텍스처, 노이즈)으로 현실 차이를 줄입니다. 필요시 **실제 데이터 미세조정** – 예: 실제 로봇에서 random policy로 시도한 작은 데이터에 imitation loss로 보정 – 을 고려할 수 있지만, 우선은 하지 않습니다.

실환경 적용과 로봇 통합

학습이 완료되면, Jetson에 학습된 모델을 배포합니다:

- PyTorch 모델을 TorchScript로 export하거나, ONNX 변환 후 TensorRT 최적화도 고려합니다. 다만 정책 네트워크는 비교적 작으므로 그냥 PyTorch로 CPU inferencing만 해도 실시간에 가깝게 동작할 수 있고, 원한다면 Jetson의 GPU를 사용하도록 할 수도 있습니다.
- ROS2 Node로 만들고, 구독하는 토픽: 카메라 이미지, 로봇 상태. 발행하는 토픽: 로봇 제어 명령 (`geometry_msgs/Twist` 혹은 `control_msgs/JointJog` 등 실시간 제어 메시지).
- RL 정책은 매 주기 (예: 20Hz) 카메라 이미지 + 현재 관절각을 받아 신경망 forward -> delta end-effector 움직임 산출. 이를 IK 계산하여 6관절 속도 명령으로 변환, ROS2로 publish. (Kinova ROS2 드라이버에 joint velocity control 인터페이스가 있어야 함. 없다면, MoveIt2의 `jog_arm` 패키지를 이용해 Twist -> joint velocity 변환이 가능합니다.)
- Gripper 제어: RL policy가 잡기 시도를 하려 할 때(close action=1 출력), 이를 감지하여 gripper를 닫도록 명령합니다. 그리고 일정 힘 또는 위치에 도달하면 유지(maintain) 모드로 전환합니다.

실행 도중, 만약 물체를 잡았으면 (그리퍼 힘센서, 혹은 depth 이미지 변화로 감지 가능), 정책 출력을 무시하고 후속 동작(예: 들어올리기)을 상위 레벨에서 지... (이어서 작성) ...

실험 설계 및 시나리오 (책상 위 단순 객체 및 복합 객체 파지 테스트)

두 가지 파지 시스템(VLM 기반, RL 기반)의 성능을 비교 평가하기 위해 현실 환경에서 다양한 **실험 시나리오**를 설계합니다. 실험은 실제 Kinova Gen3 Lite 로봇팔과 RGB-D 카메라로 구성된 테스트베드에서 수행되며, 책상 위에 여러 가지 물체를 놓고 파지 작업을 테스트합니다. 각 접근법별로 성능을 정량/정성 평가하고 비교하기 위해 아래와 같은 단계적인 시나리오를 구성하였습니다.

실험 환경 세팅

- **장소:** 실내 사무실 환경의 책상 위 공간. 책상 크기는 약 120cm x 60cm, 높이 75cm 정도이며, 로봇팔은 책상 한쪽 모서리에 베이스가 고정되어 있습니다 (로봇 베이스 좌표계를 책상 모서리에 위치시킴).
- **조명 조건:** 일반 실내 조명 (형광등)으로 일정한 밝기를 유지합니다. VLM 인지가 조명에 민감할 수 있으므로, 그림자를 최소화하고, 필요하면 추가 조명(스탠드 조명)을 배치합니다.
- **카메라 위치:** RGB-D 카메라를 로봇팔 어깨 부근 또는 별도 스탠드에 장착하여 책상 위를 내려다보는 시야를 확보합니다. 카메라의 시야각 내에 로봇팔과 책상 위 전체 영역이 포함되도록 조정합니다. (카메라와 로봇 좌표계의 변환은 보정 완료된 상태로 가정)
- **테스트 물체 세트:** 다양한 형태와 크기의 **테스트 대상 물체**들을 선정합니다.
  - *단순 형상 객체:* 정육면체 나무 블록, 공 모양 장난감, 원기둥(캔), 컵, 볼펜 등 구조가 단순한 물체들.
  - *복합 형상 객체:* 손잡이가 달린 머그컵, 얇고 평평한 책, 가위 (여러 부품으로 구성), 작은 장난감 로봇 등 파지 난이도가 높은 물체들.
  - *물체 크기:* 대략 5cm에서 15cm 범위의 소형~중형 물체. (그리퍼 크기 제한으로 너무 큰 물체 제외)
  - 각 물체는 표면 재질도 다양하게 (플라스틱, 금속, 천 등) 선정하여 인지/파지의 일반화 성능을 시험합니다.
- **배치 방식:** 시나리오별로 책상 위 물체들의 배치를 다르게 합니다. 단일 물체 실험의 경우 물체를 책상 중앙 근처에 놓고, 다중 물체 실험의 경우 두세 개를 서로 약간 떨어뜨려 놓습니다. 또한 특정 시나리오에서는 물체를 책상 모서리나 벽 가까이 배치하거나, 물체들 간 거리를 좁혀 난이도를 조절합니다.

실험 시나리오별 구성

시나리오 A – 단일 단순 객체 파지:

- 한 번에 **하나의 단순 형태 물체**만 책상 위에 올려놓고 파지 시도.
- 예: 나무 블록 하나를 임의 위치에 놓고, VLM 시스템에 "블록 집어" 명령을 주거나 RL 시스템을 동작시켜 블록을 잡도록 함.
- 각 단순 물체(블록, 공, 컵 등)에 대해 5회 이상 반복 시도하여 **파지 성공률**을 측정합니다.
- 또한 **평균 파지 시간**(명령 후 물체를 들어올릴 때까지 걸린 시간)을 기록합니다.
- 기대: 두 시스템 모두 단순 물체에 대해서는 높은 성공률(>90%)을 보일 것으로 예상됩니다. 시간 측면에서는, VLM+MoveIt은 인지+경로계획으로 약 3~5초 소요, RL 정책은 즉각 제어로 2~3초 내에 파지 완료하는 차이가 있을 것으로 봅니다.

시나리오 B – 단일 복합 객체 파지:

- **하나의 복잡한 형태 물체**(머그컵, 책, 가위 등)를 대상으로 파지 시도.
- 각 물체별로 5~10회 반복하여 성공률을 측정하고, 어떤 **파지 접근 방식**을 취했는지 관찰합니다.
- 머그컵의 경우 VLM 기반은 손잡이 vs 컵 몸통 어느 쪽을 잡는지, RL 기반은 임의로 어느 부분을 잡는지 확인합니다. 책의 경우 VLM은 옆면 파지를 계획할 가능성, RL은 책 위를 덮치려다 실패할 수 있는지 등을 관찰합니다.
- 이 시나리오는 **어려운 물체에 대한 대응력**을 평가하며, 실패 발생 시 그 원인(인식 오류, 그립 실패 등)을 기록해둡니다.

시나리오 C – 다중 객체 중 특정 물체 선택 (VLM 특화 실험):

- 책상 위에 **두 개 이상의 서로 다른 물체**를 동시에 배치하고, 특정 물체만 집도록 지시.
- 예: 공과 컵을 동시에 놓고, "공을 집어" 라고 명령.
- VLM 기반 시스템: 명령된 대상(공)만 정확히 집는지 확인. 이때 다른 방해물(컵)이 있어도 잘 무시하는지 평가.
- RL 기반 시스템: (학습 정책에는 특정 물체 구분 개념이 없으므로) 시각적으로 보이는 둘 중 임의의 것에 접근할 것으로 예상됨. RL은 보통 가까이에 있는 물체나 큰 물체를 우선 집으려 할 수 있습니다. 어떤 물체를 잡는지 기록합니다.
- 평가 포인트: VLM 시스템의 **목표 지정 정확도** (지시한 물체를 집었는지 여부), RL 시스템의 **임의 동작 경향** (특정 상황에서 어느 물체를 선호하는지).
- 10가지 조합 정도 (예: 공+컵, 책+블록, 컵+가위 등)로 실험하여 통계적 경향을 파악합니다.

시나리오 D – 파지 난이도 조건 부여:

- 물체가 **잡기 어려운 위치나 자세**에 있는 경우를 시험합니다.
- 예: (1) 물체가 책상 **모서리**에 일부 걸쳐있어 완전히 평면에 놓이지 않은 경우, (2) **벽에 인접**하게 놓인 경우 (로봇 접근 방향 제한), (3) 두 물체가 매우 근접해 붙어있는 경우 등.
- 이러한 조건에서 VLM+MoveIt은 충돌 회피 경로를 크게 우회하거나 파지 자체를 포기할 수 있고, RL 정책은 학습되지 않은 상황에 직면하여 실패할 수 있습니다.
- 각 경우 5회 이상 시도하여 성공/실패를 기록하고, 시스템들의 행동 패턴을 정성적으로 관찰합니다 (예: VLM 시스템이 아예 plan 실패를 리턴하는지, RL이 물체를 밀어내 버리는지 등).

시나리오 E – 실시간 변화 대응 (동적 상황 실험): *(선택 실험)*

- 파지 시도 중에 사람이 개입하여 **물체의 위치를 약간 이동**시키는 등 동적 변화를 주었을 때 시스템의 대응 능력 평가.
- VLM+MoveIt의 경우 현재 구현상 정적인 가정이라 변화에 즉각 대응하지 못하고 실패할 가능성이 큽니다 (change detection 후 재계획 필요). RL 시스템은 폐루프 제어로 어느 정도 변화에 대응할 수 있을지 관찰합니다.
- 이 실험은 정량화보다는 **동작 관찰** 위주로 실시하고, 필요하면 비디오로 기록하여 후분석합니다.

측정 항목 및 데이터 수집

각 실험 시나리오에서 **평가 지표**를 아래와 같이 정의하고 측정합니다:

- **파지 성공률:** N회 시도 중 성공 횟수 / N * 100%. (성공 기준: 물체를 확실히 그리퍼로 집어서 들어올려 5cm 이상 높이 유지)

- **평균 파지 수행 시간:** 성공한 시도에 한해, 명령 입력 시점부터 물체를 들어 안정적으로 든 시점까지의 시간(초)을 측정하여 평균 계산.
- **경로계획 시간 (VLM 시스템 전용):** VLM 기반 시스템에서, 명령 입력 후 객체 인지 완료 및 MoveIt2 경로계획이 완료될 때까지 걸린 시간. (ROS2 노드 타임스탬프 로깅)
- **정확한 대상 파지 비율:** (시나리오 C의 경우) 지시한 물체를 정확히 집은 비율. VLM 시스템은 이 값이 100%에 근접해야 성공적이며, RL 시스템은 별도의 목표식별이 없으므로 해당 없음.
- **실패 유형 분석:** 실패한 각 사례에 대해 원인을 분류:
  - *인지 실패*: 잘못된 객체를 탐지하거나 위치 오인 (VLM).
  - *계획 실패*: MoveIt2가 경로를 찾지 못함 또는 충돌로 중지 (VLM).
  - *파지 동작 실패*: 물체를 놓침, 미끄러짐, 힘이 부족함 등 (양 시스템 공통).
  - *정책 오류*: RL 정책이 엉뚱한 곳을 시도하거나 물체를 밀어냄 등 (RL).
- **파지 안정성 및 품질:** 주관적으로 파지 후 흔들림이 없는지, 그리퍼의 잡는 위치가 적절한지 등을 관찰하여 메모합니다. (예: 컵을 잡았을 때 기울어지지 않고 수직으로 잡았는지 등)

데이터 수집은 ROS2의 rosbag을 활용하여 모든 토픽 (카메라 영상, 인식 결과, 로봇 상태, 명령 등)을 기록합니다. 특히 **MoveIt2 출력 로그**, **RL 노드 로그**에 성공/실패 플래그와 이유를 기록하도록 합니다. 실험 후 해당 로그와 rosbag 데이터를 분석 스크립트를 통해 처리하여 위 지표들을 산출합니다.

예상 결과 및 분석 계획

위 실험들을 통해 얻은 데이터로 다음과 같은 결과와 분석을 기대합니다:

- **성공률 비교:** 시나리오 A (단순 객체)에서는 VLM 기반과 RL 기반 모두 거의 **100%에 가까운 파지 성공률**을 보일 것으로 예상됩니다. 반면 시나리오 B (복잡 객체)에서는 VLM 기반 시스템이 상대적으로 높은 성공률을 유지하는 반면, RL 기반은 객체 형태 적응에 어려움을 겪어 다소 낮은 성공률을 나타낼 가능성이 있습니다. 예컨대 머그컵의 손잡이 잡기는 VLM이 손잡이를 인식하여 접근하면 성공하겠지만, RL은 손잡이보다는 몸통을 잡으려다 미끄러질 수 있습니다.

- **실행 시간:** VLM 시스템은 인지+플래닝 단계로 인한 초기 지연이 존재하므로, **평균 파지 시간**이 RL에 비해 길 것입니다. 예를 들어 VLM 방식 4초, RL 방식 2초라면, **실시간성 측면**에서 RL 접근이 유리함을 보여줍니다. 그러나 VLM도 Jetson 최적화로 전체 5초 내외로 동작하면, 일반적인 서비스 로봇 응용에 수용 가능한 수준입니다. 경로계획 시간 세부 측정으로, MoveIt2가 복잡한 상황(시나리오 D 등)에서 소요 시간이 늘어나는지도 확인할 것입니다.

- **정확도 및 목표식별:** 시나리오 C 결과를 통해, **VLM 기반 시스템이 명시된 대상만 파지하는 정확도**를 평가할 수 있습니다. 기대상 정확히 지시된 물체만 집어야 하나, 만약 VLM 인지가 오탐하거나 혼동하면 다른 물체를 잡는 실수가 있을 수 있습니다. 반면 RL 기반은 구분 없이 가까운 물체를 잡을 것이며, 이는 **범용성 한계**로 분석됩니다. 이 비교를 통해 VLM 접근의 **고수준 명령 이행 능력**이 강조될 것입니다.

- **적응성:** 시나리오 D와 E를 통해 **동적 적응성**을 평가한 결과, VLM+MoveIt은 주로 정적 환경에 가정되어 있어 이러한 변화에 약하며, RL 정책은 일부 대응 능력을 보일 수 있지만 한계가 있을 것입니다. 예컨대 물체 위치를 살짝 바꾸면 RL은 따라가는 경향을 보이겠지만 큰 변화에는 실패할 것입니다. 이는 VLM+MoveIt에 **폐루프 보정**이 필요하고, RL에 **추가 학습**이나 **안전장치**가 필요함을 시사합니다.

- **실패 원인 분석:** 수집된 로그를 통해 실패 케이스들을 분류하면, VLM 시스템의 실패는 주로 인식 오류(예: 유사한 객체 잘못 탐지)나 경로계획 충돌 문제로 나뉠 것으로 보입니다. RL 시스템의 실패는 대부분 파지 자체 실패(그립 미스) 또는 정책의 엉뚱한 시도로 인한 것으로 예상됩니다. 이 분석은 **개선 방향**에 인사이트를 줍니다. 예를 들어, VLM 인지 실패율이 높다면 더 나은 모델(예: Grounding DINO 사용)로 개선하거나 데이터 증강이 필요하고, RL 파지 실패가 잦다면 보상함수나 관찰 정보 보강이 필요함을 알 수 있습니다.

분석 계획으로는, 각 시나리오별로 위 지표들을 표와 그래프로 정리하여 두 시스템을 비교합니다. 예를 들어 **성공률 비교 막대그래프**, **평균 수행 시간 박스플롯** 등을 작성합니다. 또한 사례별 사진을 캡처하여, 성공/실패의 상황을 시각적으로 제시할 것입니다. 통계적으로 유의미한 차이가 있는지는 시나리오별 시도횟수가 충분하지 않을 수 있으나, 경향 파악 위주로 서술하고 필요시 카이제곱 테스트 등 검정을 간략 실시합니다.

본 연구는 이처럼 **정량적 평가**와 **정성적 고찰**을 모두 포함하여, 두 접근법의 성능 특성을 다각도로 비교 분석할 계획입니다.

결론 및 향후 확장 연구 방향 (모바일 매니퓰레이터 확장)

결론: 본 연구에서는 **비전-언어 모델 기반 파지 시스템**과 **강화학습 기반 파지 시스템**을 설계하고, 실제 로봇팔 플랫폼에서 구현 및 비교 실험을 수행했습니다. VLM 기반 접근은 사전 학습된 거대 멀티모달 모델을 활용하여 **새로운 객체에 대한 제로샷 인지**와 **자연어 명령 이행**이 가능하고, MoveIt2 통합을 통해 **안정적인 모션 계획**을 달성하였습니다. 강화학습 기반 접근은 시뮬레이션으로 학습된 정책을 통해 **폐루프 제어**와 **적응적 파지 동작**을 보여주었으며, 사람의 수동 개입 없이 **저수준 제어를 최적화**하는 잠재력을 확인했습니다. 실험 결과, 단순한 환경에서는 두 접근 모두 높은 성능을 보였으나, **목표 지정 정확성**과 **상황 적응성** 측면에서 상호 보완적인 특징을 나타냈습니다. 이는 로봇 파지 문제에 단일 기법으로 완벽히 대응하기 어렵다는 점을 시사하며, **하이브리드 방법**의 가능성을 제기합니다.

한계: VLM 기반 시스템은 실시간성에서 약간 부족하고 동적 변화에 약하다는 한계를 보였고, RL 기반 시스템은 목표 선택 기능이 없고 충분한 학습을 거치지 않은 상황에서는 오작동 가능성이 있음을 알았습니다. 또한 Jetson AGX Orin 상에서 두 시스템을 구동하며 **연산 자원의 제약**을 실감하였는데, 추론 최적화와 경량화의 중요성을 재확인하였습니다.

향후 연구 방향: 이러한 결과를 바탕으로 다음과 같은 확장 연구를 제안합니다:

- **VLM-RL 하이브리드 시스템:** VLM의 고수준 인지 능력과 RL의 저수준 제어 능력을 결합하는 방향입니다. 예를 들어, VLM으로 목표 물체를 인식하고 대략적 파지 지점을 지정한 후, 세부 접근 동작은 RL 정책이 실시간 조정하는 시스템을 구상할 수 있습니다. 이는 **Vision-Language-Action** 모델의 일종으로 볼 수 있으며, 최근 등장한 OpenVLA 등의 접근과도 연계됩니다.
- **모바일 매니퓰레이터로 확장:** 현재는 고정된 로봇팔이지만, 이를 **이동 로봇 베이스**와 결합한 **모바일 매니퓰레이터**로 확장하면 적용 범위가 크게 넓어집니다. 향후에는 로봇이 방 안을 자율 주행하여 물체가 있는 곳으로 이동하고, 매니퓰레이터로 집어오는 시나리오 (예: "주방에서 빨간 컵 가져와")를 고려합니다. 이를 위해서는 SLAM, 경로 계획 (Nav2)과 본 연구의 파지 시스템을 통합해야 합니다. VLM 기반 인지는 **환경 맥락 이해**(어느 방에 무엇이 있는지 등)로까지 확장되어야 하고, RL 정책은 움직이는 베이스에 대응하도록 추가 센서 정보(IMU 등)를 받아야 할 것입니다.
- **보다 복잡한 상호작용**: 단순 파지를 넘어서 **물체 조작**(manipulation)으로 연구를 확장할 수 있습니다. 예를 들어 서랍을 열고 물건을 집는 복합 과제, 두 손으로 물체를 옮기는 작업 등에 VLM과 RL 접근을 적용하는 것입니다. 이러한 과제에서는 **계획(plan)과 제어(control)의 계층적 조합**이 중요해지므로, 계층적 강화학습이나 멀티모달 플래너 등의 주제가 추가로 연구될 것입니다.
- **성능 개선 및 최적화:** 소프트웨어적으로, Jetson 플랫폼에서 더욱 높은 FPS를 달성하기 위해 TensorRT 엔진 최적화, 파이프라인 병렬화, 모델 경량화(MMDetection 등 활용) 등을 지속 탐구합니다. 또한 실험적으로 부족했던 부분, 예를 들어 RL 정책의 학습 다양성을 늘리기 위해 **Issac Orbit** 등 상위 환경이나 **Auto-Curriculum** 전략을 적용할 수 있습니다.

- **사용자 상호작용 및 학습:** 최종적으로 로봇이 사용자와 함께 작업하려면, 사용자 피드백으로 파지 실패를 교정하는 **온사이트 학습**이나, **자연어로 설명**하며 로봇이 학습하는 기법 (예: "이건 무거우니 두 손으로 잡아") 등 인간-로봇 상호작용 측면 연구도 발전시킬 수 있습니다.

본 연구의 결과는, 향후 가정용 서비스 로봇이나 산업용 로봇에서 **다양한 물체를 유연하게 다룰 수 있는 파지 시스템** 구축에 기여할 수 있을 것입니다. **비전-언어 인공지능과 강화학습의 융합**은 로봇의 지능적 행동을 향상시키는 유망한 방향이며, 본 연구를 통해 얻은 통찰이 그 발전에 밑거름이 되기를 기대합니다.

참고 문헌 및 관련 링크

1. Alec Radford et al., *"Learning Transferable Visual Models From Natural Language Supervision" (CLIP)*, ICML 2021 (CLIP: Connecting text and images | OpenAI) (CLIP: Connecting text and images | OpenAI).
2. Shridhar et al., *"CLIPort: What and Where Pathways for Robotic Manipulation"*, CoRL 2022. (자연어-시각 정보 결합 로봇 파지 연구).
3. Alexander Kirillov et al., *"Segment Anything"*, arXiv 2023. (Meta AI의 SAM 모델 소개 및 코드).
4. Liu et al., *"Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection"*, arXiv 2023. (텍스트 기반 객체 탐지 모델).
5. Dmitry Kalashnikov et al., *"QT-Opt: Scalable Deep RL for Vision-Based Robotic Manipulation"*, CoRL 2018 (Google 연구, 실세계 데이터로 학습한 파지 RL).
6. Makoviychuk et al., *"Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning"*, arXiv 2021 (대량 병렬 RL 시뮬레이션 플랫폼).
7. NVIDIA Jetson AI Lab, *ROS2 NanoML Packages*, 2023. (ROS2용 NanoOWL, NanoSAM 패키지).
8. MoveIt 2 Documentation – Grasping, Pick and Place Tutorials.
9. Robotics Data Set (YCB Dataset) – 대학/연구소 공개 물체 모델 세트.
10. 기타: 그 외 본 보고서에 인용된 SAM2, MobileSAM, OpenVLA 등은 최신 arXiv 논문 및 GitHub 자료를 참고하였으며, 본 연구 맥락에서 요약하여 활용했습니다.