2026 겨울 세미나

2026.02.13

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
김우영

Outline

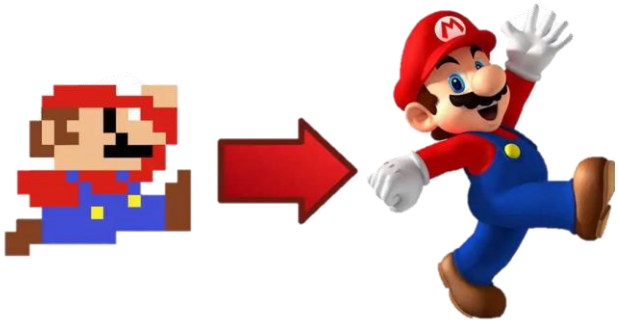
- Background
  - 3D(Point cloud) Anomaly detection
  - VLA(Vision Language Action)
- Paper 1
  - Towards Zero-shot Point Cloud Anomaly Detection: A Multi-View Projection Framework [IEEE TSMC 2025]
- Paper 2
  - CoT-VLA: Visual Chain-of-Thought Reasoning for Vision-Language-Action Models [CVPR 2025]

Background

- Anomaly Detection
  - 정상 데이터 분포를 학습하고, 이를 벗어나는 샘플을 이상으로 탐지하는 문제

- 2D AD의 한계
  - 이미지 및 영상은 깊이, 기하학적 구조와 같은 3D 정보 소실과 occlusion에 취약

- 3D AD
  - Data type: point cloud, depth map, mesh …
  - 2D 모달리티 대비 데이터 수집의 어려움
  - 이에 따른 few/zero-shot model 필요



=> Towards Zero-shot Point Cloud Anomaly Detection: A Multi-View Projection Framework

	Category	Real Size [mm]			Attribute	Training	Test		Total	Anomaly Point Ratio
		Length	*Width*	*Height*		*Normal*	*Normal*	*Abnormal*		Δ
	Airplane	34.0	14.2	31.7	Transparency	4	50	50	104	1.18%
	Car	35.0	29.0	12.5	Transparency	4	50	50	104	1.99%
	Candybar	33.0	20.0	8.0	Transparency	4	50	50	104	2.37%
	Chicken	25.0	14.0	20.0	White	4	52	54	110	4.39%
	Diamond	29.0	29.0	18.7	Transparency	4	50	50	104	5.41%
	Duck	30.0	22.2	29.4	Transparency	4	50	50	104	2.00%
	Fish	37.7	24.0	4.0	Transparency	4	50	50	104	2.86%
	Gemstone	22.5	18.8	17.0	Transparency	4	50	50	104	2.06%
	Seahorse	38.0	11.2	3.5	Transparency	4	50	50	104	4.57%
	Shell	21.7	22.0	7.7	Transparency	4	52	48	104	2.25%
	Starfish	27.4	27.4	4.8	Transparency	4	50	50	104	4.47%
	Toffees	38.0	12.0	10.0	Transparency	4	50	50	104	2.46%
	Mean	30.9	20.3	13.9	—	4	50	50	104	3.00%
	Total	—	—	—	—	48	604	602	1254	—

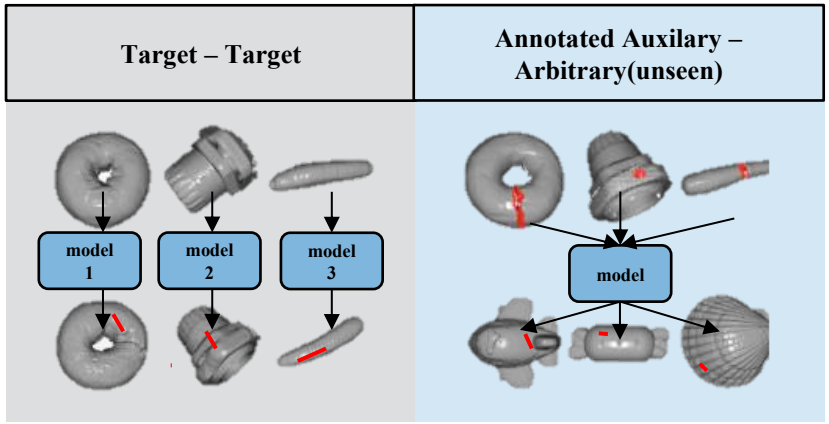
Towards Zero-shot Point Cloud Anomaly Detection: A Multi-View Projection Framework

[IEEE TSMC 2025]

Introduction

- Unsupervised 3D Anomaly Detection
  - Data Collection: 2D image 대비 point cloud data를 수집하는데 비용 큼
  - Cold Start Problem: Normal data 조차 수집할 수 없는 경우 존재
  - Separate Model Training: 학습에 사용한 단일 class에 대해서만 적용 가능

- Zero-Shot 3D Anomaly Detection
  - Zero-Shot Image AD에서는 VLM을 활용한 method 제안
  - Point cloud AD에서도 이를 활용하기 위해 MVP framework을 제안



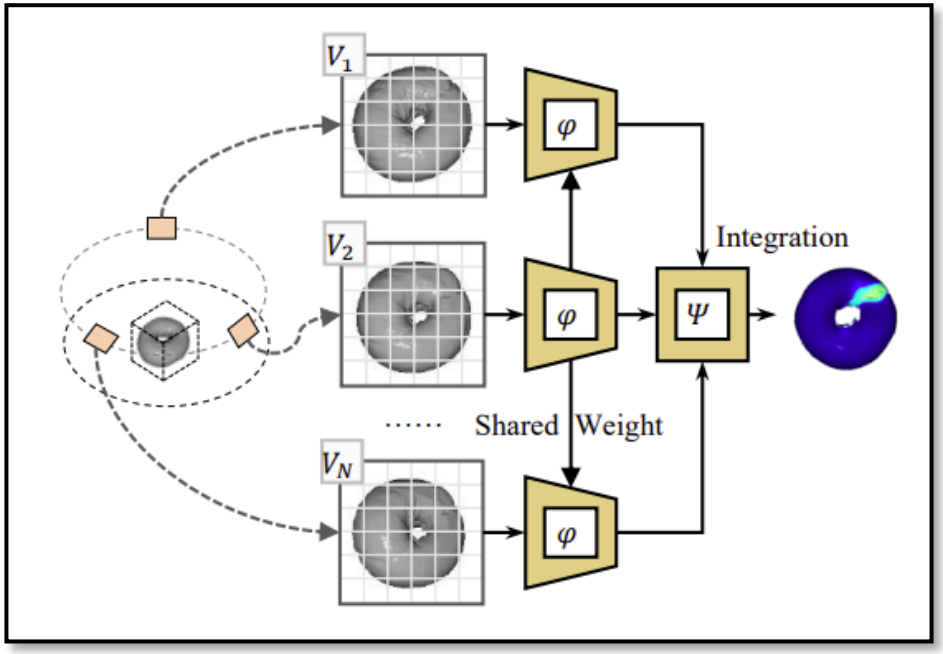
[Figure 1: Unsupervised vs. Zero-Shot]

MVP Framework

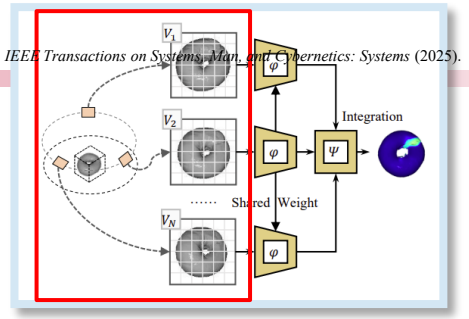
3D point cloud를 바로 VLM에서 다룰 수 없기 때문에 depth image를 생성해서 anomaly score/map 계산

- Overall Process

  0. Point cloud data alignment
  1. Multi-view projection: Pcd를 여러 각도에서 투영시켜 N개의 2D depth image 생성
  2. Single-view anomaly detection: depth image의 anomaly score 계산 (shared model φ)
  3. Integration: N개의 image output 결과를 통합하여 최종 score 계산



[Figure 2: MVP Framework]



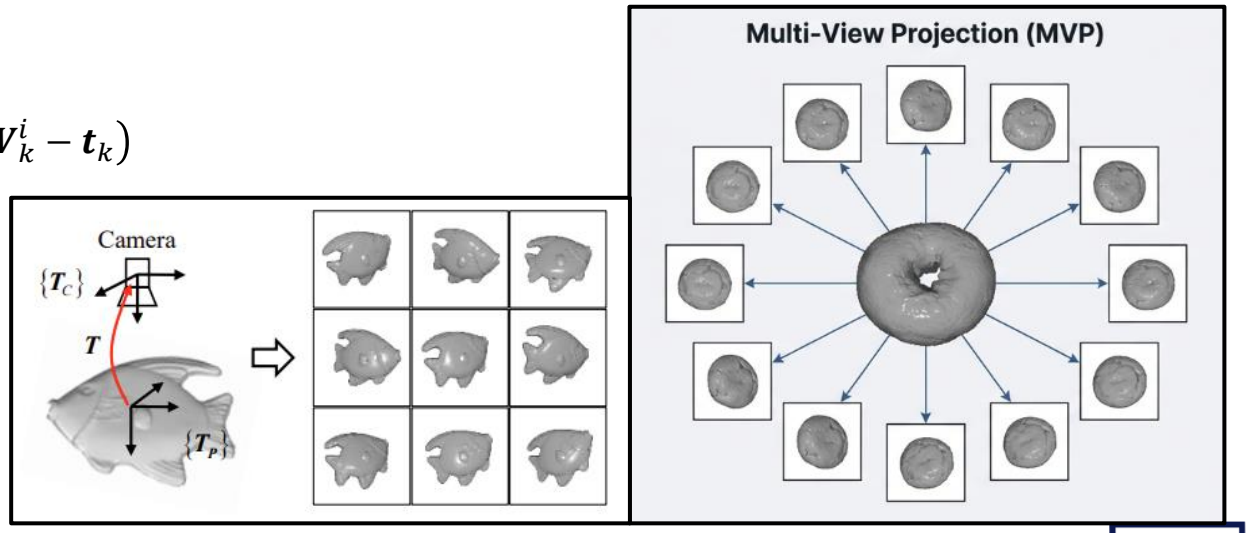
MVP Framework

- Problem Definition
  - $\boldsymbol{P} \in R^{n\times 3}$: point cloud → $\boldsymbol{A} \in R^{n\times 1}, \boldsymbol{\xi} \in [0,1]$: point/object-wise anomaly score
  - Supervised training: Annotated auxiliary point cloud dataset
  - $\boldsymbol{\chi}_{\text{train}} - \{\boldsymbol{A}\}_{\text{train}} / \{\boldsymbol{\xi}\}_{\text{train}}$

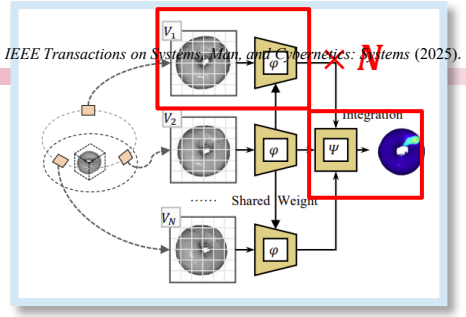
- Point Cloud Multi-View Projection (Camera Calibration)
  - i_{th} point를 각각의 pose $\boldsymbol{T}$ 에서 projection (K: camera intrinsic) [1]

$$\left[\boldsymbol{V}_k^i, 1\right]^T = (1/z_i)\boldsymbol{K}\boldsymbol{T}_k\left[\boldsymbol{P}^i, 1\right]^T$$

$$\boldsymbol{P} = Pos_k(\boldsymbol{V}_k) = \boldsymbol{R}_k^{-1}\left(z_i\boldsymbol{K}^{-1}\boldsymbol{V}_k^i - \boldsymbol{t}_k\right)$$



[Figure 3: Multi-View Projection Visualization]

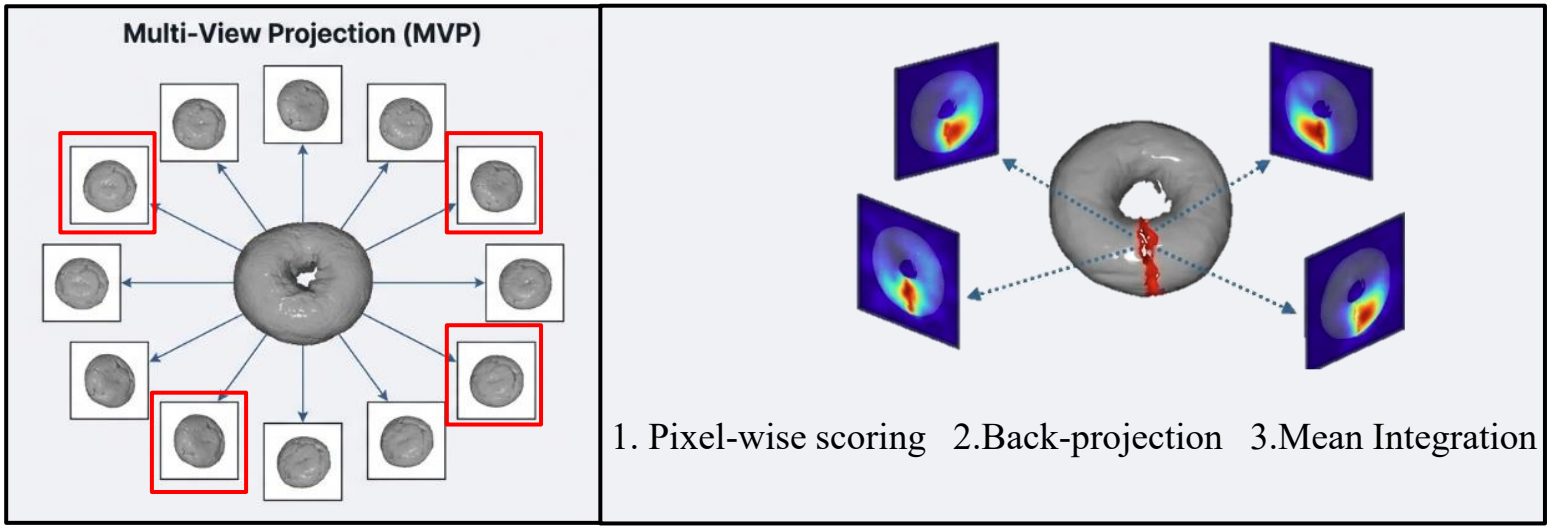


MVP Framework

- Single View Anomaly Detection
  - i_{th} depth image $\boldsymbol{V}_i$를 shared zero-shot AD model에 입력하여 detection result $\boldsymbol{C}_i$ 출력

    $\boldsymbol{C}_i = \boldsymbol{\varphi}(\boldsymbol{V}_i)$
- Integration
  - N개의 output $\{C_1, C_2, \ldots C_N\}$을 fuse(Ψ)하여 최종 object/point level score 계산

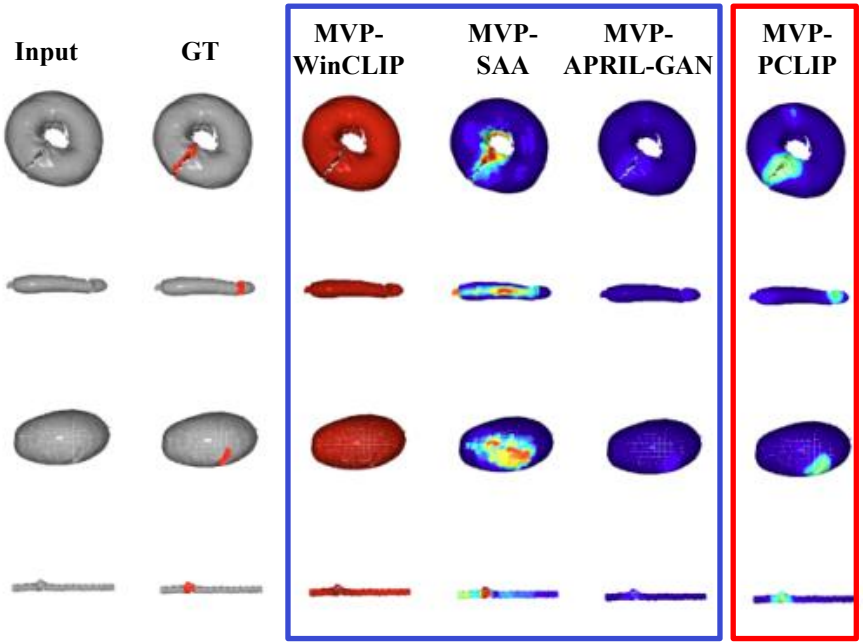
    $\xi, \mathbf{A} = \Psi(\{\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots \boldsymbol{C_N}\})$



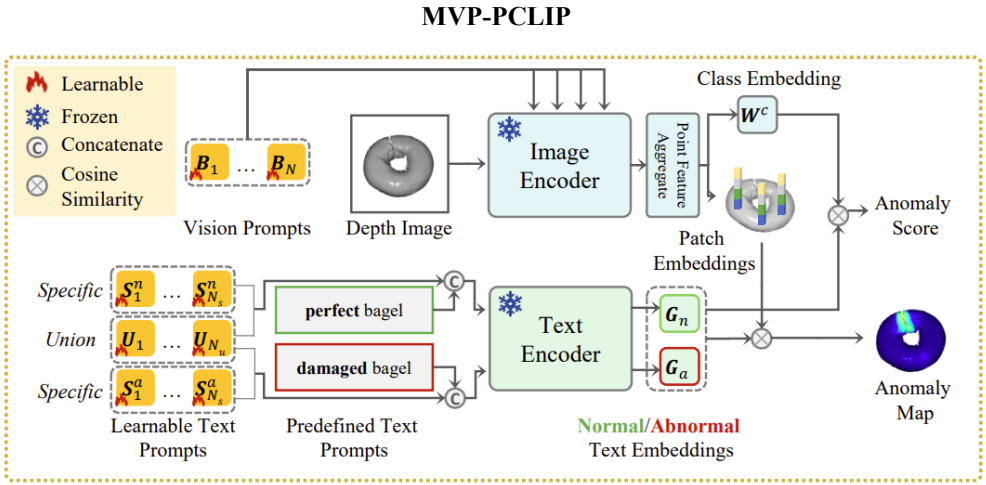
[Figure 4: Multi-View Score Integration]

MVP Framework to MVP-PCLIP

- Why MVP-PCLIP?
  - MVP framework을 기존 2D AD 모델들에 적용했을 때 domain gap 존재
  - Domain gap을 줄이기 위해 Key Layer Visual Prompt(KLVP)와 Adaptive Text Prompt(ATP)를 CLIP에 추가한 MVP-PCLIP 제안



[Figure 5: Domain Gap Visualization]



[Figure 6: MVP-PCLIP Architecture]

MVP-PCLIP

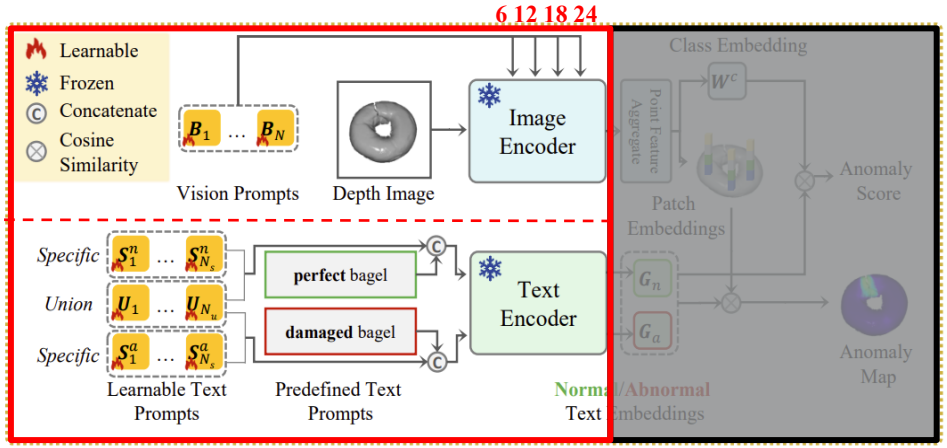
- Key Layer Visual Prompt (KLVP)
  - Key layer $\{k_1, k_2, \ldots, k_m\}$에 대해 이전 layer의 token output에 learnable token B_j를 더해서 well guide 된 feature token 추출 (6, 12, 18, 24)

$$\begin{cases} \boldsymbol{W}_j = L_j(\boldsymbol{W}_{j-1}), & \text{if } j \notin \{k_1, k_2, \cdots, k_m\} \\ \tilde{\boldsymbol{W}}_j = L_j(\boldsymbol{W}_{j-1}, \boldsymbol{B}_j), & \text{if } j \in \{k_1, k_2, \cdots, k_m\} \end{cases}$$

- Adaptive Text Prompt (ATP)
  - $T_{prompt} = \boldsymbol{h}(cat(\boldsymbol{U}, \boldsymbol{S}, State, CLS))$

$$t_n = [U_1] \cdots [U_{n_u}][S_1^n] \cdots [S_{n_s}^n] \text{ perfect bagel}$$
$$t_a = [U_1] \cdots [U_{n_u}][S_1^a] \cdots [S_{n_s}^a] \text{ damaged bagel}$$

* State ∈ [perfect, normal, damaged …]
* CLS ∈ {bagel, carrot …]
* $\boldsymbol{U}$: union learnable
* $\boldsymbol{S}$: specific learnable



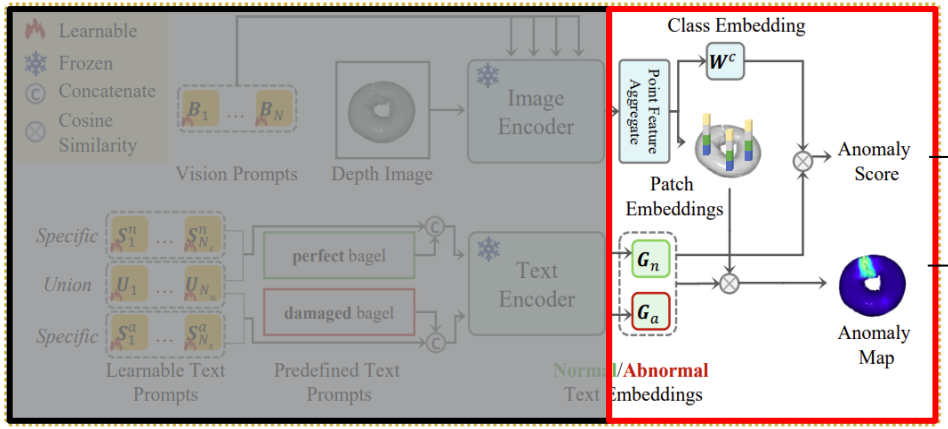
[Figure 7: MVP-PCLIP Prompt Learning]

MVP-PCLIP

- Point Cloud Anomaly Detection (point-wise)
  - Image encoder의 key layer $\boldsymbol{k}$에서 얻은 각 feature token을 point level로 반환
  - N개의 multi-view image와 각 key layer에서 얻은 point level feature를 합쳐 최종 point feature 반환
  - 이를 text feature $\boldsymbol{G}$와 similarity 연산을 통해 score 산출

$$\boldsymbol{P}_{jk}^{i} = Pos_j(\boldsymbol{F}_{jk}^{i}) \qquad \tilde{\boldsymbol{P}}^{\boldsymbol{i}} = \frac{1}{mN}\sum_{j}^{N}\sum_{l}^{m}\boldsymbol{P}_{jk_l}^{i} \qquad \boldsymbol{G} = g(t_n, t_a)$$

- Point Cloud Anomaly Detection (object-wise)
  - Image encoder에서 반환된 N개의 class token $\boldsymbol{W}_i^{clc}$의 평균과 text feature $\boldsymbol{G}$와 연산



$$\xi = softmax((\frac{1}{N}\sum_{i}^{N}\tilde{\boldsymbol{W}}_i^{clc})\boldsymbol{G}^{\mathrm{T}})$$

$$\boldsymbol{A} = softmax(\tilde{\boldsymbol{P}}\boldsymbol{G}^{\mathrm{T}})$$

$$\mathcal{L} = IoU(\boldsymbol{A}, \boldsymbol{A}_{gt}) + Focal(\boldsymbol{A}, \boldsymbol{A}_{gt}) + CrossEntroy(\xi, \xi_{gt})$$

[Figure 8: MVP-PCLIP Anomaly Scoring]

Experiment

- Quantitative Results
  - MVP-PCLIP이 zero-shot setting에서 가장 우수하며, Real3D-AD의 경우 unsupervised setting의 CPMF도 상회

Category	CPMF (Unsupervised) [9] *PR'2024*	PointMAE [42] *ECCV'2022*	MVP-WinCLIP [13] *CVPR'2023*	MVP-SAA [15] *CVPRW'2023*	MVP-APRIL-GAN [14] *CVPRW'2023*	MVP-PCLIP
Bagel	(96.6±0.0, 53.0±0.0, 55.2±0.0)	(50.0±0.0, 2.6±0.0, 1.3±0.0)	(58.1±0.0, 3.4±0.0, 1.7±0.0)	(90.1±0.0, 32.3±0.0, 25.5±0.0)	(95.9±0.1, 38.8±0.8, 31.6±1.2)	(**97.1**±0.1, **41.7**±0.6, **35.0**±0.8)
Cable_Gland	(92.8±0.0, 34.5±0.0, 30.6±0.0)	(50.0±0.0, 3.1±0.0, 1.6±0.0)	(61.4±0.0, 5.8±0.0, 2.7±0.0)	(56.9±0.0, 4.1±0.0, 1.8±0.0)	(76.8±1.8, 15.0±1.5, 7.4±1.6)	(**82.2**±0.5, **20.4**±0.8, **12.1**±0.7)
Carrot	(95.5±0.0, 49.2±0.0, 48.3±0.0)	(50.2±0.0, 5.7±0.0, 3.0±0.0)	(73.5±0.0, 12.4±0.0, 8.7±0.0)	(88.7±0.0, 36.5±0.0, 29.4±0.0)	(91.9±0.3, 36.6±0.3, 29.9±0.2)	(**94.4**±0.2, **40.7**±0.5, **34.2**±0.7)
Cookie	(92.8±0.0, 51.9±0.0, 47.0±0.0)	(50.0±0.0, 4.6±0.0, 2.4±0.0)	(71.8±0.0, 11.2±0.0, 6.0±0.0)	(76.0±0.0, **39.7**±0.0, **32.2**±0.0)	(**85.4**±0.3, 33.0±1.2, 27.1±2.1)	(84.7±0.3, 31.8±1.4, 25.2±1.4)
Dowel	(89.0±0.0, 36.0±0.0, 29.0±0.0)	(50.0±0.0, 4.2±0.0, 2.1±0.0)	(49.9±0.0, 5.4±0.0, 2.3±0.0)	(68.9±0.0, 14.3±0.0, **8.4**±0.0)	(**74.0**±0.9, **15.0**±0.9, 7.8±0.8)	(72.0±0.8, 14.8±0.9, 7.5±0.6)
Foam	(79.0±0.0, 35.4±0.0, 29.5±0.0)	(50.0±0.0, 1.2±0.0, 0.6±0.0)	(50.2±0.0, 1.3±0.0, 5.0±0.0)	(62.5±0.0, 13.1±0.0, 6.0±0.0)	(67.8±1.9, 20.1±0.3, 9.7±0.1)	(**70.2**±1.0, **23.8**±0.1, **12.3**±0.2)
Peach	(98.7±0.0, 56.4±0.0, 55.9±0.0)	(50.0±0.0, 2.6±0.0, 1.3±0.0)	(77.8±0.0, 11.0±0.0, 5.3±0.0)	(94.4±0.0, 46.4±0.0, 30.6±0.0)	(96.8±0.1, 40.8±0.4, 31.0±1.0)	(**98.5**±0.1, **47.0**±0.7, **40.3**±1.2)
Potato	(97.2±0.0, 43.2±0.0, 39.3±0.0)	(48.4±0.0, 3.6±0.0, 1.8±0.0)	(71.1±0.0, 9.2±0.0, 4.4±0.0)	(96.3±0.0, **52.1**±0.0, 42.7±0.0)	(96.8±0.1, 47.0±0.8, 47.3±2.0)	(**98.2**±0.0, 49.0±0.4, **49.9**±0.3)
Rope	(96.8±0.0, 51.2±0.0, 48.5±0.0)	(53.9±0.0, 10.6±0.0, 3.8±0.0)	(85.8±0.0, 24.8±0.0, 14.7±0.0)	(68.6±0.0, 14.1±0.0, 8.0±0.0)	(94.7±0.1, 47.2±0.5, 39.7±0.5)	(**96.6**±0.1, **51.5**±0.5, **44.7**±0.4)
Tire	(96.9±0.0, 42.4±0.0, 38.1±0.0)	(50.0±0.0, 1.6±0.0, 0.8±0.0)	(47.8±0.0, 1.8±0.0, 7.0±0.0)	(73.3±0.0, 5.3±0.0, 2.2±0.0)	(78.5±0.5, 11.3±0.3, 4.5±0.2)	(**81.5**±0.5, **15.7**±0.2, **7.5**±0.2)
Mean	(93.5±0.0, 45.3±0.0, 42.1±0.0)	(50.3±0.0, 4.0±0.0, 1.9±0.0)	(64.7±0.0, 8.6±0.0, 4.7±0.0)	(77.6±0.0, 25.8±0.0, 18.7±0.0)	(85.9±0.3, 30.5±0.3, 23.6±0.4)	(**87.5**±0.2, **33.6**±0.1, **26.9**±0.3) (↑1.6, ↑3.1, ↑3.3)

[Table 1: MVTec3D-AD Point-wise Results]

Category	CPMF (Unsupervised) [9] *PR'2024*	PointMAE [42] *ECCV'2022*	MVP-WinCLIP [13] *CVPR'2023*	MVP-SAA [15] *CVPRW'2023*	MVP-APRIL-GAN [14] *CVPRW'2023*	MVP-PCLIP
Airplane	(61.8±0.0, 2.3±0.0, 1.0±0.0)	(50.0±0.0, 1.6±0.0, 0.8±0.0)	(42.4±0.0, 2.6±0.0, 0.7±0.0)	(73.7±0.0, 12.6±0.0, 2.8±0.0)	(78.2±0.7, 9.5±2.6, 3.3±0.6)	(**81.7**±0.5, **16.4**±1.8, **10.1**±1.6)
Candybar	(83.6±0.0, 13.5±0.0, 6.4±0.0)	(50.0±0.0, 2.7±0.0, 1.4±0.0)	(57.5±0.0, 5.8±0.0, 1.9±0.0)	(70.4±0.0, 6.1±0.0, 2.4±0.0)	(76.6±1.7, 8.6±1.5, 3.1±0.6)	(**79.8**±0.4, **13.9**±2.3, **6.4**±0.8)
Car	(73.4±0.0, 10.7±0.0, 5.0±0.0)	(50.0±0.0, 2.3±0.0, 1.2±0.0)	(50.8±0.0, 7.0±0.0, 1.8±0.0)	(58.6±0.0, 4.0±0.0, 1.6±0.0)	(62.8±2.4, 4.3±0.7, 1.4±0.3)	(**69.5**±1.0, **7.4**±0.6, **4.5**±0.7)
Chicken	(55.9±0.0, 7.1±0.0, 3.1±0.0)	(50.0±0.0, 5.5±0.0, 2.8±0.0)	(60.3±0.0, 8.6±0.0, 3.9±0.0)	(**90.6**±0.0, **41.6**±0.0, **32.2**±0.0)	(87.3±0.9, 29.3±1.7, 21.6±2.7)	(88.8±0.3, 32.2±0.7, 27.8±0.9)
Diamond	(75.3±0.0, 14.9±0.0, 7.4±0.0)	(50.0±0.0, 5.5±0.0, 2.8±0.0)	(68.5±0.0, 9.1±0.0, 4.7±0.0)	(90.7±0.0, 31.3±0.0, 18.8±0.0)	(90.0±2.0, 38.0±1.4, 33.5±2.2)	(**95.3**±0.9, **48.5**±2.0, **47.9**±2.3)
Duck	(71.9±0.0, 4.2±0.0, 1.8±0.0)	(50.0±0.0, 1.9±0.0, 1.0±0.0)	(44.3±0.0, 2.0±0.0, 0.8±0.0)	(80.5±0.0, 11.1±0.0, 3.7±0.0)	(86.6±0.6, 10.4±1.1, 6.4±1.3)	(**88.1**±0.7, **14.0**±1.4, **8.2**±0.6)
Fish	(98.8±0.0, 58.2±0.0, 55.9±0.0)	(50.0±0.0, 2.7±0.0, 1.4±0.0)	(71.4±0.0, 15.4±0.0, 6.3±0.0)	(**93.1**±0.0, **67.9**±0.0, **68.5**±0.0)	(82.4±2.4, 28.6±0.7, 17.4±0.2)	(85.8±1.6, 18.6±1.7, 11.0±1.1)
Gemstone	(44.9±0.0, 2.0±0.0, 0.7±0.0)	(50.0±0.0, 1.8±0.0, 1.8±0.0)	(66.5±0.0, 3.1±0.0, 1.3±0.0)	(79.5±0.0, 7.9±0.0, 2.8±0.0)	(87.9±1.0, 24.0±2.0, 13.3±1.0)	(**91.5**±0.4, **28.0**±0.5, **18.2**±0.8)
Seahorse	(96.2±0.0, 61.5±0.0, 63.6±0.0)	(50.0±0.0, 4.9±0.0, 4.8±0.0)	(65.3±0.0, 8.7±0.0, 4.1±0.0)	(64.3±0.0, 16.1±0.0, 11.1±0.0)	(**81.5**±1.9, **24.8**±4.2, **18.3**±3.1)	(80.5±1.2, 21.7±3.4, 17.1±3.1)
Shell	(72.5±0.0, 5.2±0.0, 2.5±0.0)	(50.0±0.0, 2.3±0.0, 2.3±0.0)	(62.0±0.0, 9.4±0.0, 2.7±0.0)	(87.3±0.0, 19.1±0.0, **17.0**±0.0)	(89.1±1.4, 26.3±1.6, 12.5±1.0)	(**90.1**±0.7, **28.1**±1.4, 16.4±1.0)
Starfish	(80.0±0.0, 20.2±0.0, 12.8±0.0)	(50.0±0.0, 4.5±0.0, 4.5±0.0)	(63.1±0.0, 7.1±0.0, 3.7±0.0)	(59.7±0.0, 14.7±0.0, 6.0±0.0)	(**75.4**±0.6, **22.2**±2.6, **13.4**±2.0)	(70.9±1.2, 19.4±0.8, 12.1±0.5)
Toffees	(95.9±0.0, 46.0±0.0, 39.1±0.0)	(50.0±0.0, 2.2±0.0, 2.2±0.0)	(48.1±0.0, 3.0±0.0, 1.1±0.0)	(87.4±0.0, 18.8±0.0, 8.6±0.0)	(**94.8**±0.3, **41.1**±5.6, **37.5**±3.7)	(93.4±0.5, 38.7±3.7, 34.7±2.5)
Mean	(75.8±0.0, 20.5±0.0, 16.6±0.0)	(50.0±0.0, 3.2±0.0, 2.3±0.0)	(58.4±0.0, 6.8±0.0, 2.7±0.0)	(78.0±0.0, 20.9±0.0, 14.6±0.0)	(82.7±1.0, 22.3±1.2, 15.1±1.0)	(**84.6**±0.3, **23.9**±0.4, **17.9**±0.1) (↑1.9, ↑1.6, ↑2.8)

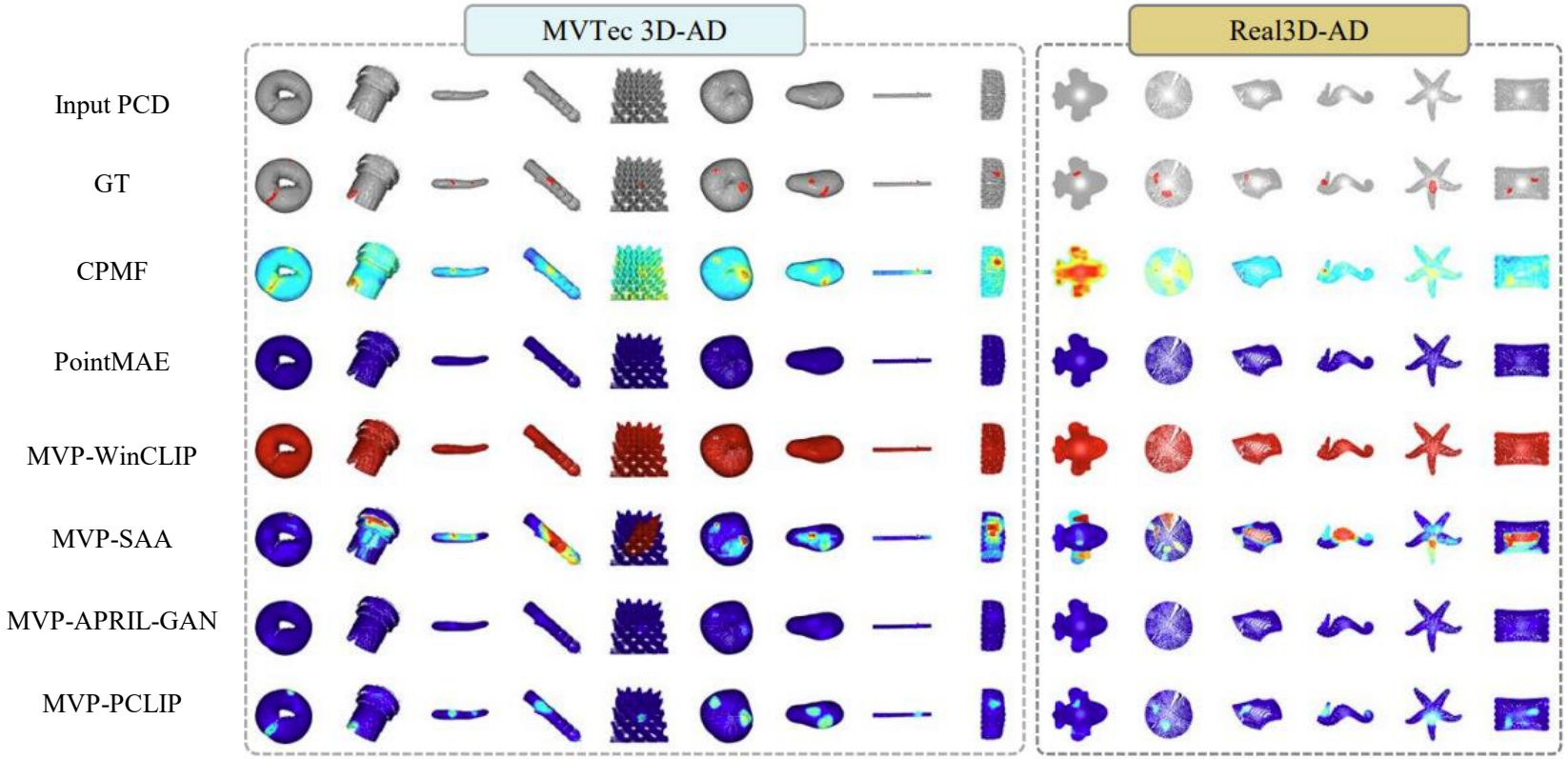
[Table 2: Real3D-AD Point-wise Results]

Experiment

- Quantitative & Qualitive Results

Category	MVTec 3D-AD	Real3D-AD
CPMF (Unsupervised) [9] *PR'2024*	(94.6±0.0, 98.6±0.0, 95.2±0.0)	(62.5±0.0, 64.2±0.0, 72.3±0.0)
PointMAE [42] *ECCV'2022*	(4.7±0.0, 88.2±0.0, 78.5±0.0)	(51.4±0.0, 67.8±0.0, 54.7±0.0)
MVP-WinCLIP [13] *CVPR'2023*	(71.0±0.0, 89.1±0.0, 89.2±0.0)	(54.1±0.0, 68.5±0.0, 54.5±0.0)
MVP-SAA [15] *CVPRW'2023*	(54.7±0.0, 88.4±0.0, 81.8±0.0)	(49.1±0.0, 69.2±0.0, 54.5±0.0)
MVP-APRIL-GAN [14] *CVPRW'2023*	(66.9±8.5, 88.2±0.5, 82.6±4.2)	(51.2±4.0, 68.1±1.0, 54.2±2.8)
MVP-PCLIP	(**71.9**±2.4, **90.7**±0.1, **89.6**±1.8) (↑0.9, ↑1.6, ↑0.4)	(**55.3**±0.8, **69.5**±0.2, **57.5**±1.4) (↑1.2, ↑0.3, ↑2.8)

[Table 3: Object-wise Results]



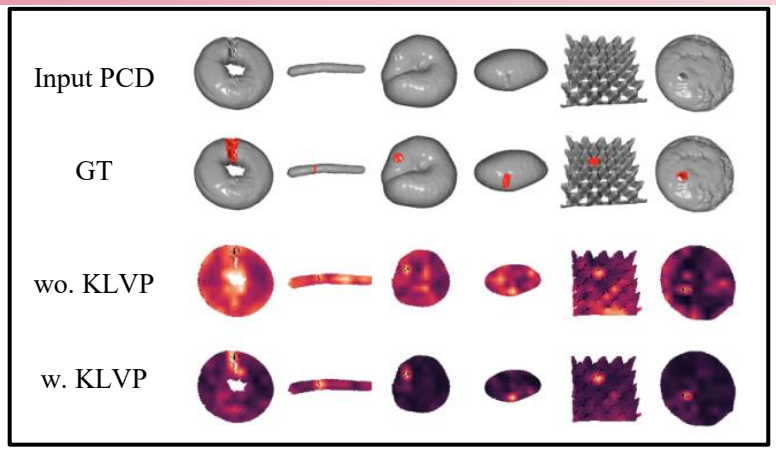
[Figure 9: Anomaly Map Visualization]

Ablation Studies

- ATP & KLVP(P-R, P-F, P-P)

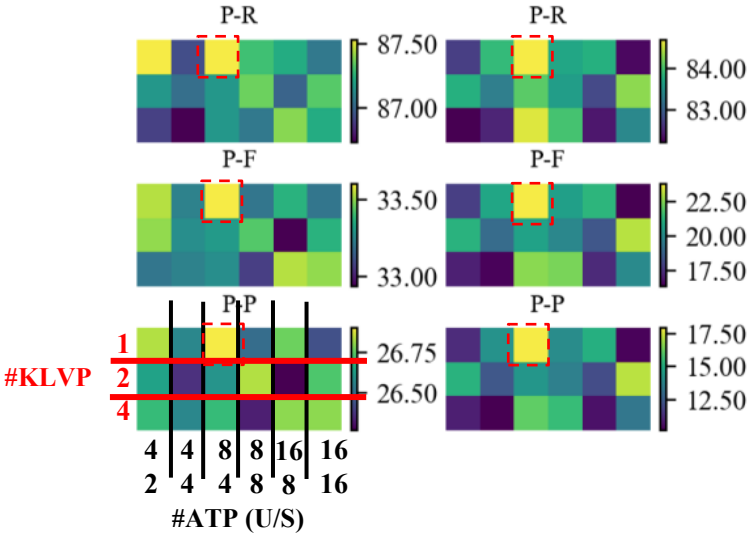
Base	ATP	KLVP	MVTec 3D-AD	Real3D-AD
✓			(86.8, 32.6, 25.4)	(83.1, 21.8, 15.5)
✓	✓		(87.2, 33.3, 26.3)	(84.3, 23.3, 17.2)
✓		✓	(87.1, 32.8, 25.7)	(83.8, 22.4, 16.2)
✓	✓	✓	**(87.5, 33.6, 26.9)**	**(84.6, 23.8, 17.9)**

[Table 4: 모듈 별 Ablation Study]



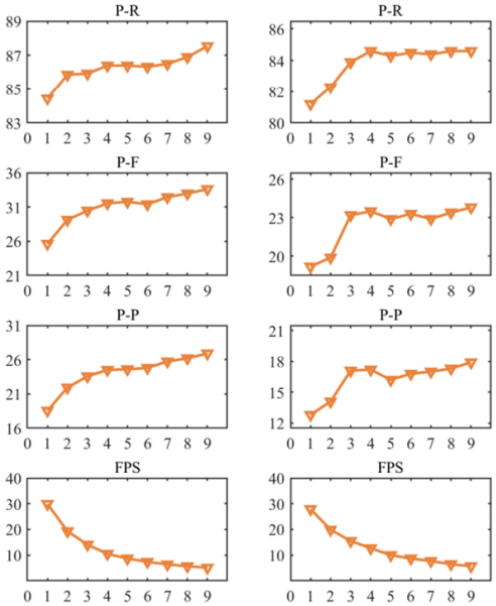
[Figure 10: Image Encoder Feature Comparision]

- Number of Learnable Parameters
  - KLVP=1 / ATP(U/S) = 8, 4 일 때 가장 성능이 좋음



[Figure 11: Parameter Ablation Study]

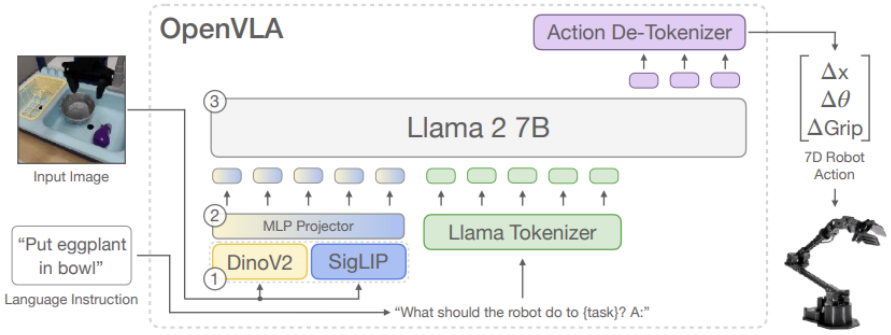
- Number of Rendering Views
  - View = 9 일 때 FPS와 성능 간의 sweet point



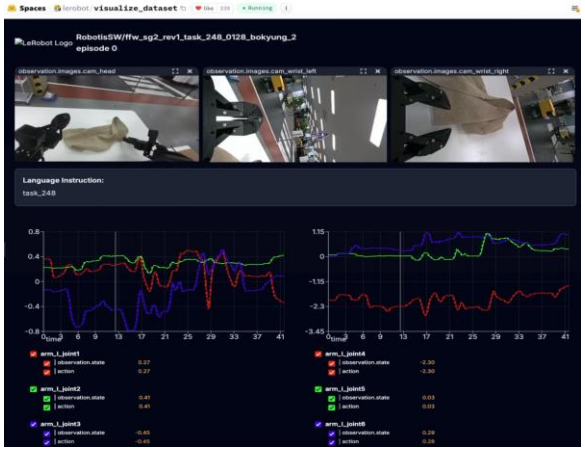
[Figure 12: Optimal Rendering View]
(Left) MVTec3D-AD (Right) Real3D-AD

Background

- VLA?
  - Vision-Language-Action: 시각 입력과 자연어 지시를 받아, 로봇 행동을 생성
- Why VLA?
  - Rule-based 등의 기존 방식들은 제한된 환경에서만 작동 가능
  - Robot task에서도 pretrain된 VLM을 활용하여 일반화된 성능을 확보하자
- Scaling Law?
  - LLM을 통해 모델의 성능이 모델의 크기와 데이터의 양에 비례함을 확인
  - Robotics에서는 text나 image와 달리 action label이 달린 데이터를 수집하는 것은 매우 어려움 (ex. Teleoperation, Simulation)



[Figure 13: OpenVLA]



[Figure 14: Action Data 예시]

CoT-VLA: Visual Chain-of-Thought Reasoning for Vision-Language-Action Models

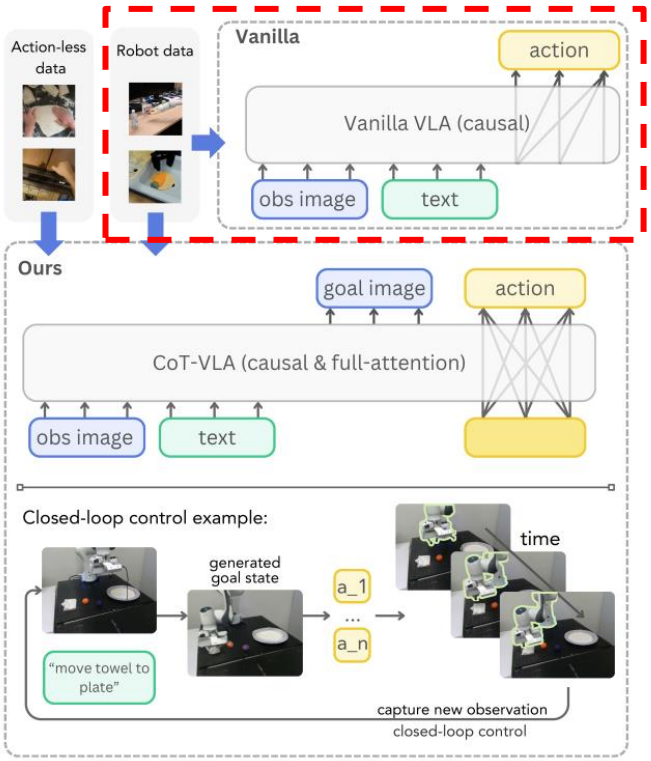
[CVPR 2025]

Introduction

- Vanilla VLA의 한계
  - Direct input - output mapping에 따른 중간 단계의 추론 부재
  - Text 기반 CoT는 공간에 대한 정보를 소실
  - Action label이 있는 robot demonstration 데이터에 의존

- Contribution
  - Subgoal image를 생성하는 visual CoT를 제안
  - 중간 단계의 image를 autoregressive하게 생성하여 공간 정보와 함께 추론 능력 향상
  - Image generation task로의 전환을 통해 action-less 데이터로도 학습 가능

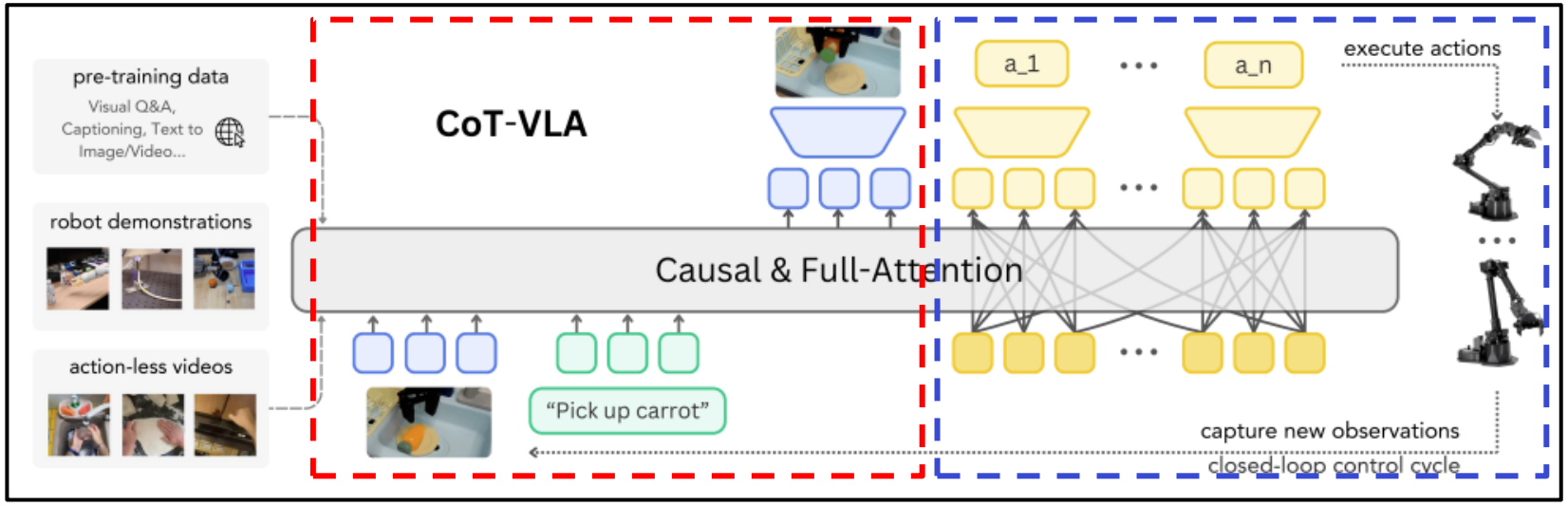
"먼저 앞으로 어떻게 될지 미래 장면을 시각적으로 그린 다음, 그 장면에 도달하기 위한 행동을 계획하자"



[Figure 15: Vanilla VLA CoT-VLA 비교

CoT-VLA

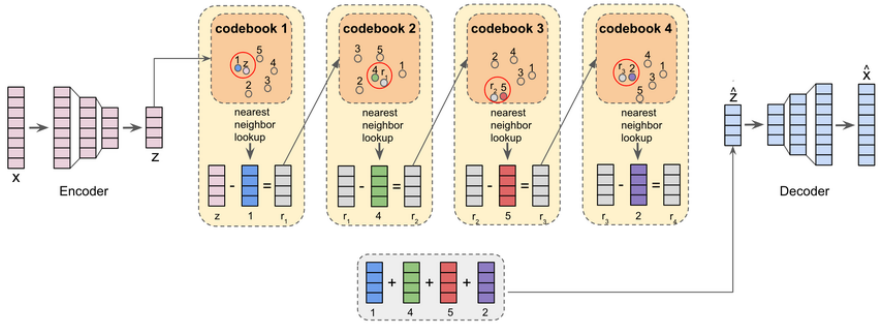
- Overview
  - 기존 VLA와 달리 2-stage에 걸쳐 action 생성
  - Stage-1: 현재 관측 이미지(s_t)와 언어 지시(l)를 입력으로 subgoal image 생성
  - Stage-2: Subgoal image를 기반으로 행동 sequence ($a_1 \ldots a_n$) 생성 (=action chunking)



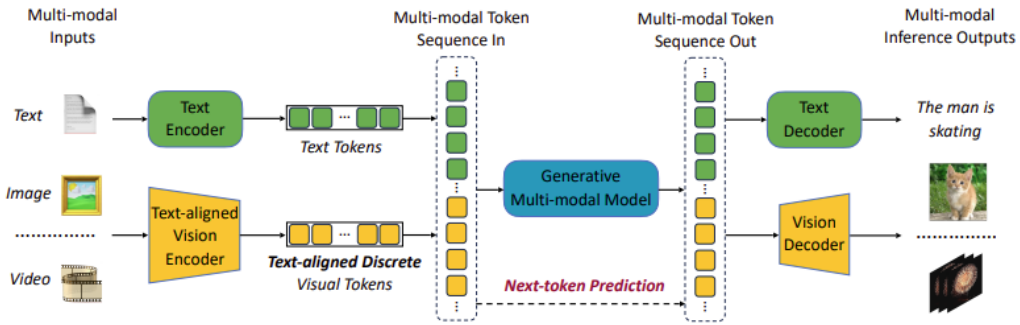
Stage-1 [Figure 16: CoT-VLA 모델 구조] Stage-2

VILA-U

- Base Vision-Language Model
  - Image를 language model이 text처럼 다루기 위해선 discrete token으로 다뤄야 함
  - RQ-VAE (d=4) 를 통해 이미지를 한 번에 토큰화하지 않고 점진적으로 잔차를 줄여가며 기존 VQ-VAE보다 풍부한 정보를 가지는 discrete token 생성[3]
  - Image(256x256)를 discrete token(16x16x4) 분해하여 Multi-Modal framework 달성



[Figure 17: RQ-VAE]

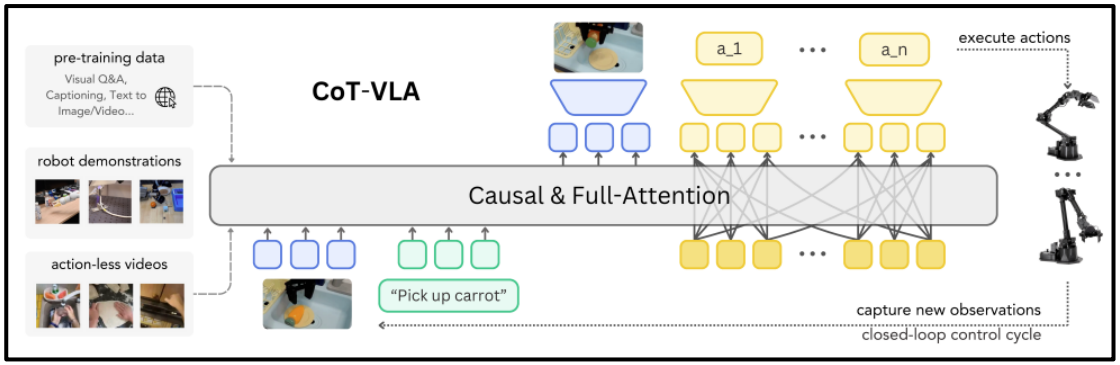


[Figure 18: VILA-U 모델 구조]

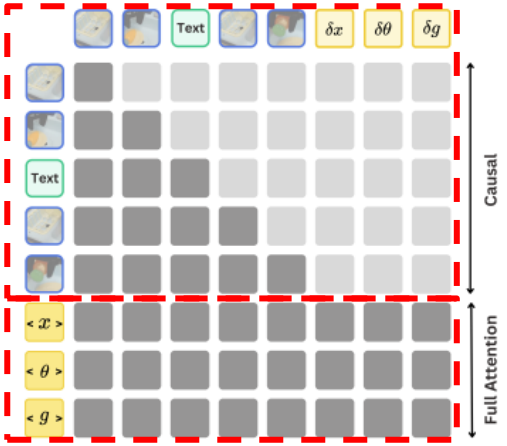
Hybrid Attention

- Causal Attention
  - Subgoal image 생성 시에는 현재까지의 token에 대해서만 attention 수행(masked)

- Full Attention
  - Action sequence 생성 시에는 앞뒤 모든 token에 대해서 attention 수행
  - Action (pos, rot, grip)은 서로 유기적으로 맞물려야 안정적인 action을 생성



[Figure 16: CoT-VLA 모델 구조]

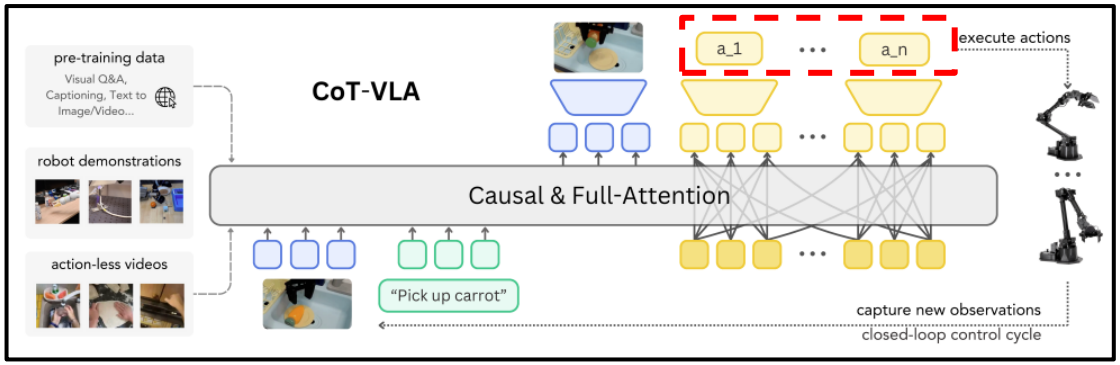


[Figure 19: Hybrid Attention]

Action Tokens

- Action Discretization
  - 연속적인 action 값을 모델이 다루기 편하도록 action distribution을 256 bin으로 나눠 각 action token을 256개의 정수로 표현
  - Action a_i는 7개($pos_{x/y/z}$, $rot_{x/y/z}$, grip)의 token으로 구성되며, Text tokenizer 내 가장 덜 쓰이는 256개의 token을 action용 vocabulary로 사용

- Action Chunking
  - 매 step마다 하나의 action 을 취하는 것이 아닌 n개의 step 분량의 행동 덩어리를 한 번에 예측하여 움직임의 일관성 확보



[Figure 16: CoT-VLA 모델 구조]

[Figure 19: Hybrid Attention]

Training

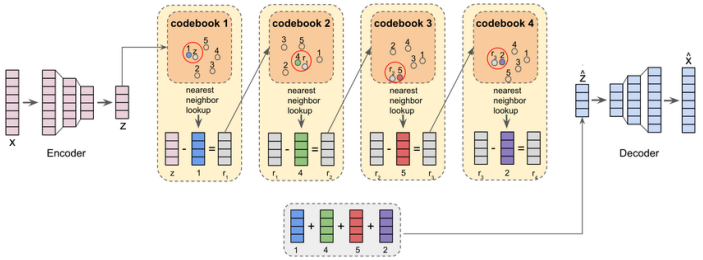
- Problem Definition
  - VLA: $\hat{\mathbf{a}}_t \sim P_\theta(\mathbf{a}_t|\mathbf{s}_t, l)$ (1)
    - 매 step t마다 현재 state s_t와 instruction l을 받아 action a_t를 생성 (1)
  - CoT-VLA : $\hat{\mathbf{s}}_{t+n} \sim P_\theta(\mathbf{s}_{t+n}|\mathbf{s}_t, l)$ (2) $\{\hat{\mathbf{a}}_t, ..., \hat{\mathbf{a}}_{t+m}\} \sim P_\theta(\{\mathbf{a}_t, ..., \mathbf{a}_{t+m}|\mathbf{s}_t, l, \mathbf{s}_{t+n}\})$ (3)
    - 현재 state s_t와 instruction l을 받아 n frame 뒤 subgoal image s_{t+n} 생성 (2) – visual CoT
    - 이후 해당 subgoal state를 달성하기 위해 m개의 action sequence 생성 (3) – action chunking
  - Dataset
    - Robot demonstration dataset $\boldsymbol{D_r} = \{(\boldsymbol{l}, \boldsymbol{a_{1...T}}, \boldsymbol{s_{1...T}})\}$
    - Action-less video dataset $\boldsymbol{D_v} = \{(\boldsymbol{l}, \boldsymbol{s_{1...T}})\}$
    - CoT-VLA는 subgoal image prediction(Eq.2)에서 $\boldsymbol{D_r}$뿐 아니라 $\boldsymbol{D_v}$ 모두 활용 가능
    - Eq.3에서는 D_r로만 학습
    - $\boldsymbol{D_r}$: Open X-Embodiment dataset
    - $\boldsymbol{D_v}$: EPIC-KITCHENS, Something-Something V2

* $\boldsymbol{l}$: 자연어 지시 명령문
* $\boldsymbol{a_{1...T}} = \{\boldsymbol{a_1}, ..., \boldsymbol{a_T}\}$: 행동 label sequence
* $\boldsymbol{s_{1...T}} = \{\boldsymbol{s_1}, ..., \boldsymbol{s_T}\}$: 시각 image sequence

Training Procedures

- Visual Token Prediction
  - j_{th} visual token에 대해 d=1~D까지의 residual token을 잘 예측하기 위한 NLL loss
  - 즉, subgoal image를 더 잘 생성하기 위함

$$\mathcal{L}_{\text{visual}} = -\sum_{j}\sum_{d=1}^{D} \log P_\delta(k_{jd}|k_{j,<d})$$



[Figure 17: RQ-VAE]

- Action Token Prediction
  - 지시문과 현재 관찰 s_t와 subgoal image s_{t+n} 을 바탕으로 올바른 action sequence를 예측하기 위한 cross entropy loss

$$\mathcal{L}_{\text{action}} = -\sum_{i=1}^{m} \log P_\theta(\mathbf{a}_t ... \mathbf{a}_{t+m}|l, s_t, s_{t+n})$$

- Overall Training Objectives
  - 미래를 정확히 상상하고(L_{visual}), 그 미래에 도달하기 위해 잘 움직이도록(L_{action}) 학습

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{visual}}$$

Experiments

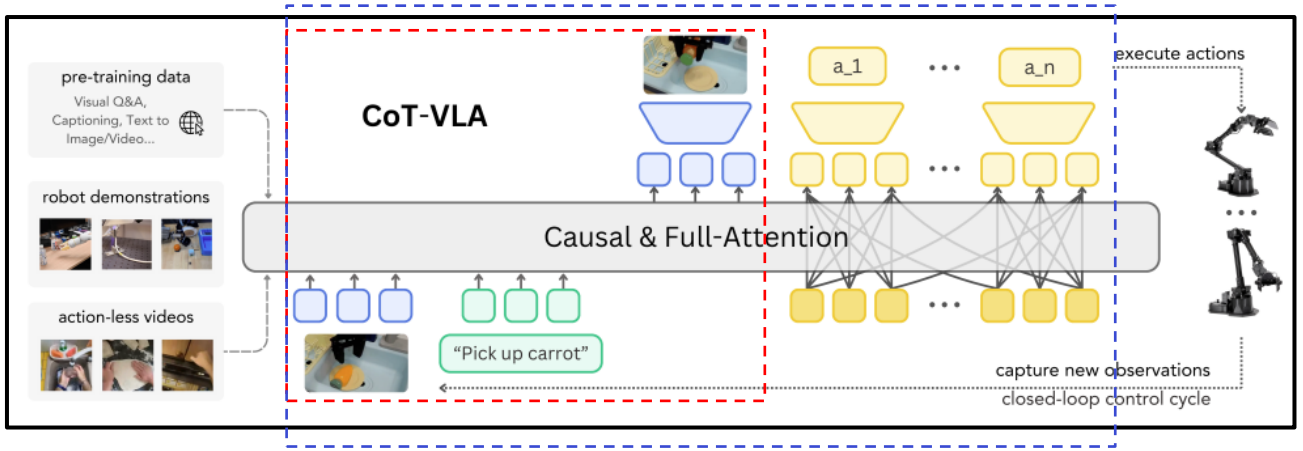
- Test-Time Closed Loop Control
  - 현재 state와 지시문을 입력으로 받아 n frame 뒤 subgoal image 생성
  - 해당 subgoal image를 달성하기 위한 m개의 action sequence 생성
  - m개의 action sequence를 수행 후 새로운 상태를 관측하여 최종 goal을 달성하기 까지 위 과정을 반복

Algorithm 1 CoT-VLA test-time closed-loop control

Require: CoT-VLA Model P_θ, initial state s_0^{obs}, language instruction l

$t \leftarrow 0$
while True **do**
  sample $\hat{\mathbf{s}}_{t+n} \sim P_\theta(\mathbf{s}_{t+n} \mid l, \mathbf{s}_t^{\text{obs}})$
  sample $[\hat{\mathbf{a}}_t, .., \hat{\mathbf{a}}_{t+m}] \sim P_\theta(\mathbf{a}_t, .., \mathbf{a}_{t+m} \mid l, \mathbf{s}_t^{\text{obs}}, \mathbf{s}_{t+n})$
  for $j = 0$ to m **do**
    execute $\hat{\mathbf{a}}_{t+j}$
  end for
  $t \leftarrow t + m + 1$
  $\mathbf{s}_t^{\text{obs}} \leftarrow$ robot observation
end while

[Figure 20: CoT-VLA test time 수도 코드]



[Figure 16: CoT-VLA 모델 구조]

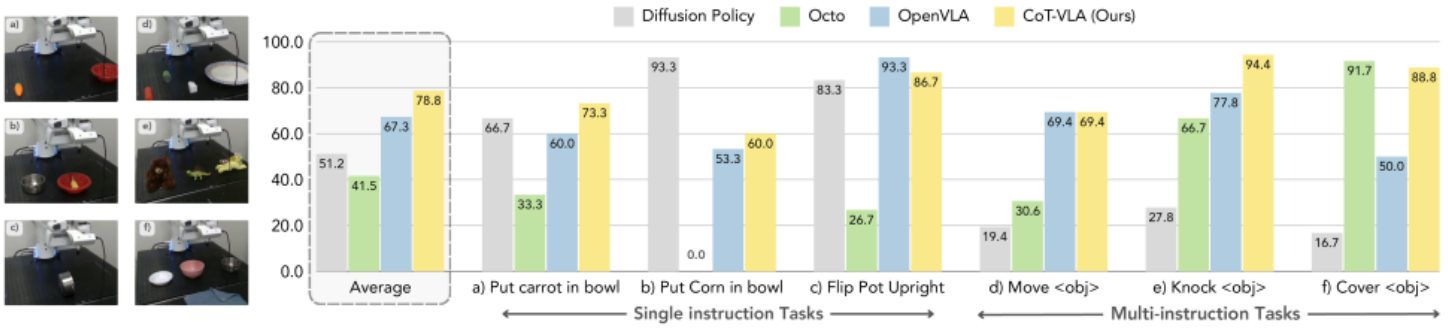
Experiments

- LIBERO benchmark results
  - Simluation 환경에서의 benchmark로 4가지 task로 구성
  - Visual CoT를 바탕으로 중간 단계의 추론 능력이 향상됨에 따라 Goal과 Long task에서 큰 성능 향상

	Average (↑)	Spatial (↑)	Object (↑)	Goal (↑)	Long (↑)
Diffusion Policy	72.4 ± 0.7%	78.3 ± 1.1%	**92.5 ± 0.7%**	68.3 ± 1.2%	50.5 ± 1.3%
Octo fine-tuned	75.1 ± 0.6%	78.9 ± 1.0%	85.7 ± 0.9%	84.6 ± 0.9%	51.1 ± 1.3%
OpenVLA fine-tuned	76.5 ± 0.6%	84.7 ± 0.9%	88.4 ± 0.8%	79.2 ± 1.0%	53.7 ± 1.3%
CoT-VLA-7B (ours)	**81.13 ± 0.6 %**	**87.5 ± 1.4%**	91.6 ± 0.5%	**87.6 ± 0.6%**	**69.0 ± 0.8%**

[Table 5: LIBERO Benchmark Results]

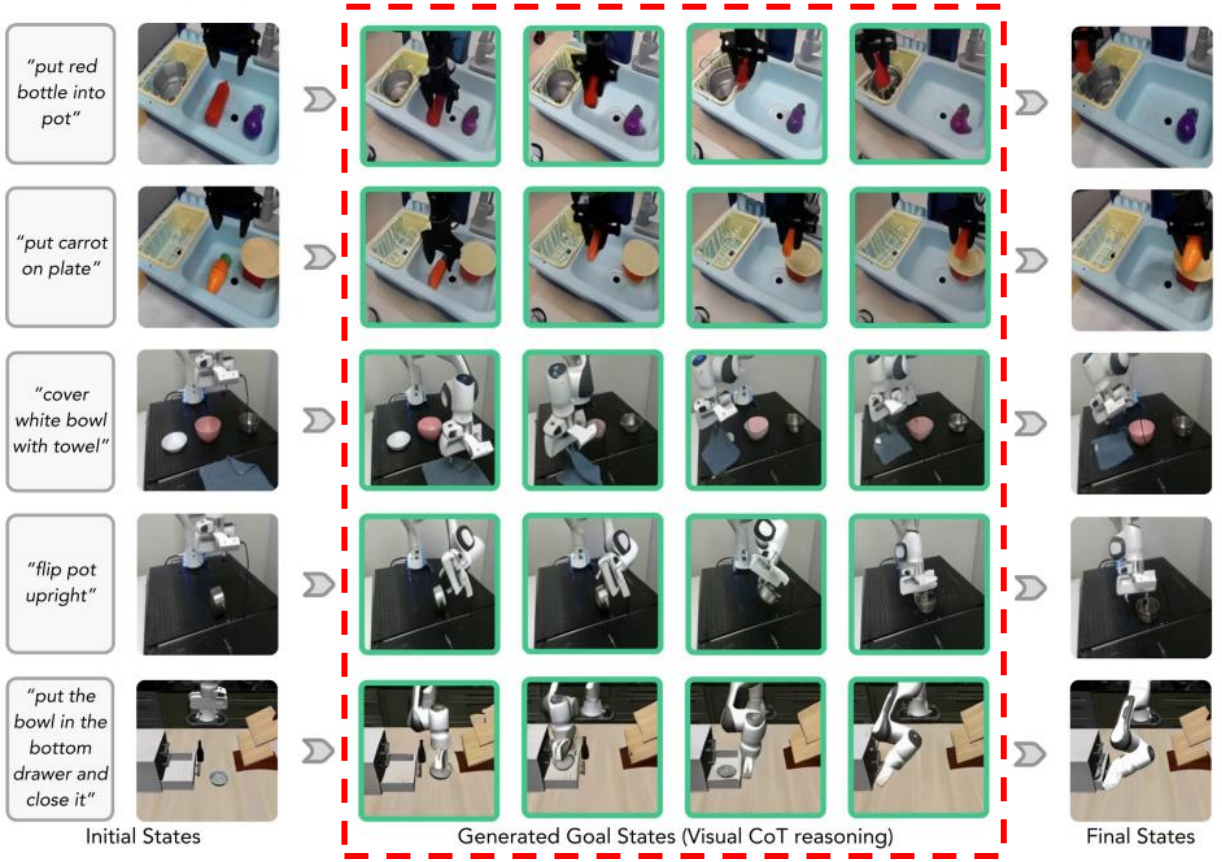
- Franka-Tabletop benchmark results
  - Franka라는 7-DoF 로봇 팔에 대한 real-world benchmark
  - CoT-VLA가 multi-instruction task와 같은 복잡한 task에서 비교 모델 대비 우수한 성능 유지



[Figure 21: Franka-Tabletop benchmark results]

Experiments

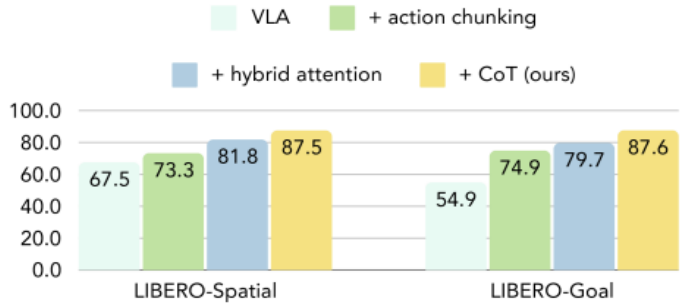
- Visual CoT visualization
  - Visual CoT를 통해 중간 단계의 subgoal image를 생성함으로써 추론 과정의 오류나 action 생성의 근거를 확인할 수 있음



[Figure 22: CoT-VLA Subgoal Visualization]

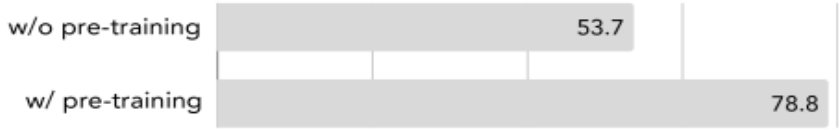
Ablation Studies

- Module 별 ablation



[Figure 23: 모듈 별 Ablation Study]

- Video Data Pretraining
  - Video data의 다양한 상황을 통해 subgoal image 예측 성능 향상
    => action-less data의 활용 가능성 제시



[Figure 24: Video Data Pretraining 성능 비교]

- Better Visual Reasoning Helps
  - 직접 생성한 subgoal image보다 GT subgoal image를 입력으로 넣었을 때 성능 향상
  - 더 정확한 subgoal image 생성이 robotic task의 성능에 직접적으로 영향을 미침
    => (improved vision reasoning == improved action execution)

	Sub-task 1	Sub-task 2
Generated Goal Images	20%	0%
Ground-truth Goal Images	**60%**	**40%**

[Table 6: Visual CoT with GT Image]

Thank You

Supplement [1]

$$[V_k^i, 1]^T = (1/z_i)\boldsymbol{K}\boldsymbol{T}_k[\boldsymbol{P}^i, 1]^T$$

```python
def rotate_render(self, pcd, rotate_angle, ref_points):
    '''
    Rotate a point cloud with the desired angle and then render it to image
    Args:
        pcd:
        rotate_angle:

    Returns:

    '''
    # rotate pcd
    R = o3d.geometry.get_rotation_matrix_from_xyz(rotate_angle)
    pcd_temp = copy.deepcopy(pcd)
    pcd_temp.rotate(R, pcd_temp.get_center())

    ref_points_temp = copy.deepcopy(ref_points)
    ref_points_temp.rotate(R, ref_points_temp.get_center())

    vis = self.vis
    # render and calculate 3d to 2d pairs
    vis.add_geometry(pcd_temp)
    image = vis.capture_screen_float_buffer(do_render=True)
    points2d = calculate_points2d(vis, np.asarray(ref_points_temp.points).T)
    vis.clear_geometries()

    # convert to rgb
    image = cv2.cvtColor(np.asarray(image) * 255, cv2.COLOR_RGB2BGR)

    return image, points2d
```

```python
def calculate_points2d(vis, pcd):
    '''
    Project a point cloud into an image plane,
    Args:
        vis:
        pcd:

    Returns:

    '''
    ctr = vis.get_view_control()
    param = ctr.convert_to_pinhole_camera_parameters()
    intrinsics = param.intrinsic.intrinsic_matrix
    extrinsics = param.extrinsic

    rvec = cv2.Rodrigues(extrinsics[:3, :3])[0]
    tvec = extrinsics[:3, 3:]

    points2d, _ = cv2.projectPoints(pcd, rvec, tvec, intrinsics, None)

    return points2d[:, 0, :].T
```

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \text{skew\_c} f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= A[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

Supplement [2]

VDSL_UY > MVP-PCLIP > param > candybar.txt

```
347 -1 0 1 2
354 1 0 1 2
361 -1 0 1 2
368 -1 0 1 2
375 -1 0 1 2
382 1 0 1 2
389 1 0 1 2
396 1 0 1 2
403 1 0 1 2
410 -1 0 1 2
417 1 0 1 2
424 1 0 1 2
431 1 0 1 2
438 1 0 1 2
445 -1 0 1 2
452 -1 0 1 2
459 1 0 1 2
466 -1 0 1 2
473 -1 0 1 2
480 1 0 1 2
494 1 0 1 2
508 1 0 1 2
515 -1 0 1 2
522 -1 0 1 2
529 -1 0 1 2
```

```python
para_file = os.path.join('./param', name + '.txt')
if os.path.exists(para_file):
    input_points = np.loadtxt(para_file, dtype=np.int64)
    orders = {input_points[i, 0]: input_points[i, 2:5] for i in range(len(input_points))}
    directions = {input_points[i, 0]: input_points[i, 1] for i in range(len(input_points))}
else:
    orders = {}
    directions = {}

txt_paths = sorted(glob.glob(os.path.join(gt_path, "*_cut.txt")))
```

투영 방향 X

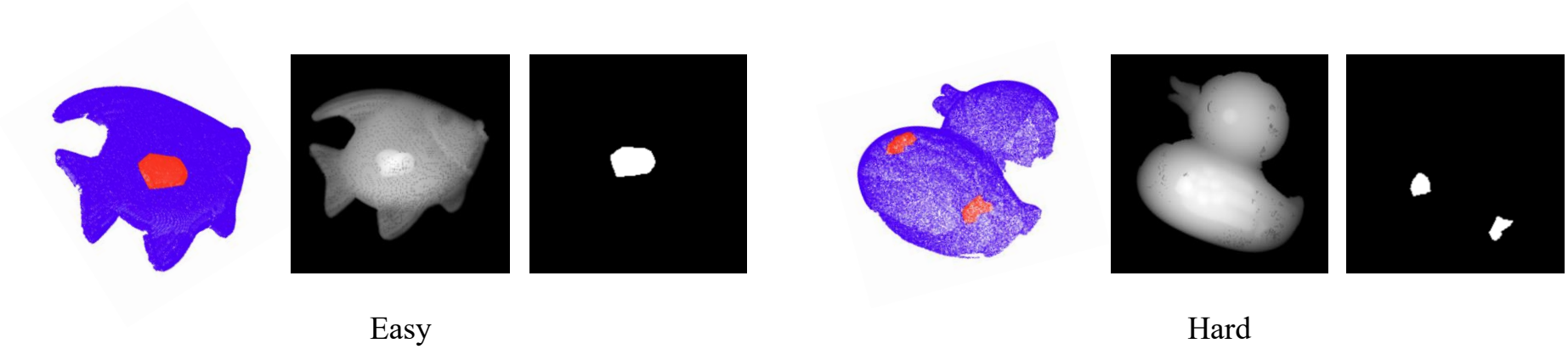
```
> ls *_cut*
137_good_cut.pcd  165_good_cut.pcd  200_good_cut.pcd  242_good_cut.pcd  270_good_cut.pcd  312_good_cut.pcd  340_good_cut.pcd
144_good_cut.pcd  172_good_cut.pcd  207_good_cut.pcd  249_good_cut.pcd  284_good_cut.pcd  319_good_cut.pcd  347_bulge_cut.pcd
151_good_cut.pcd  179_good_cut.pcd  228_good_cut.pcd  256_good_cut.pcd  298_good_cut.pcd  326_good_cut.pcd  354_bulge_cut.pcd
158_good_cut.pcd  186_good_cut.pcd  235_good_cut.pcd  263_good_cut.pcd  305_good_cut.pcd  333_good_cut.pcd  361_bulge_cut.pcd

~/VDSL_UY/datasets/Real3D-AD-PCD/candybar/test
368_bulge_cut.pcd   396_bulge_cut.pcd   424_bulge_cut.pcd   452_bulges_cut.pcd  480_sink_cut.pcd    522_sink_cut.pcd
375_bulge_cut.pcd   403_bulge_cut.pcd   431_bulge_cut.pcd   459_bulge_cut.pcd   494_sink_cut.pcd    529_sinks_cut.pcd
382_bulge_cut.pcd   410_bulge_cut.pcd   438_bulge_cut.pcd   466_bulges_cut.pcd  508_sinks_cut.pcd
389_bulges_cut.pcd  417_bulge_cut.pcd   445_bulges_cut.pcd  473_bulge_cut.pcd   515_sink_cut.pcd

ok  PCLIP py  09:08:13 PM
```

Limitation

1. Data preprocessing
   - Multi-View Projection 과정에서 어느 방향으로 볼지 직접 정해야 됨[2]

2. 이상 영역 미탐 문제
   - 초기 align 및 rendering 과정에서 anomaly area를 못 잡는 경우 해당 object가 abnomal이여도 normal로 인식

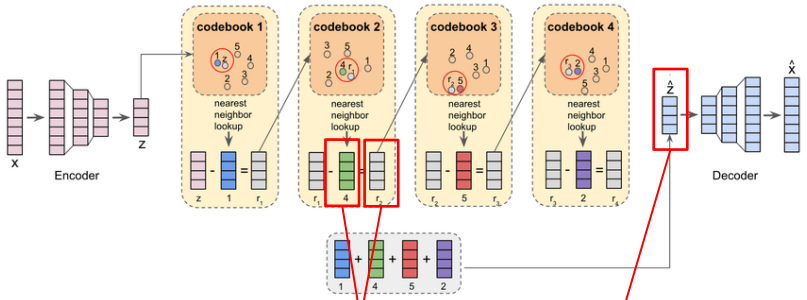
3. 정보 소실
   - 3D를 2D image로 변환했을 때 이상 영역 구분의 난이도 증가



Easy

Hard

Supplement [3]



Residual Vector Quantization. Our visual features are discretely quantized, so their representation ability heavily depends on the code size used in our quantizer. Since we hope they contain both high-level and low-level features, we need more capacities in their vector feature space, making a larger code size necessary for good performance in downstream tasks. However, too many codes for each image will result in too many tokens for LLM to produce in the visual generation process, incurring much latency. So in an attempt to increase the vector feature capacity and meanwhile maintain a reasonable number of tokens for LLM, we adopt a residual vector quantization method following RQ-VAE (Lee et al., 2022) to discretize a vector $\mathbf{z}$ as D discrete codes:

$$\mathcal{RQ}(\mathbf{z}; \mathcal{C}, D) = (k_1, \cdots, k_D) \in [K]^D, \tag{2}$$

where $\mathcal{C}$ is the codebook, $K = |\mathcal{C}|$ and k_d is the code of $\mathbf{z}$ at depth d. Starting with $\mathbf{r}_0 = \mathbf{z}$, we recursively perform vector quantization by

$$\begin{aligned} k_d &= \mathcal{Q}\left(\mathbf{r}_{d-1}, \mathcal{C}\right), \\ \mathbf{r}_d &= \mathbf{r}_{d-1} - \mathbf{e}\left(k_d\right), \end{aligned} \tag{3}$$

for each depth $d = 1, 2, \cdots, D$, where $\mathbf{e}$ is the codebook embedding table and $\mathcal{Q}$ is the standard vector quantization:

$$\mathcal{Q}(\mathbf{z}; \mathcal{C}) = \underset{k \in [K]}{\arg\min} \|\mathbf{z} - \mathbf{e}(k)\|_2^2. \tag{4}$$

The quantized vector for $\mathbf{z}$ is the sum over the depth dim: $\hat{\mathbf{z}} = \sum_{i=1}^{D} \mathbf{e}\left(k_i\right)$. Intuitively, in each depth we choose a code to reduce the quantization error. So compared to the standard vector quantization methods, we have D codes to quantize one vector, allowing for finer approximation and larger feature space. During multi-modal training and inference, LLM only needs to predict the code embedding, with codes in different depth sequentially produced by a depth transformer taking the code embedding as the initial input, as we will introduce in Section 3.2. So with this residual quantization, we can enhance the representation capability of our vision tower while incurring little latency.