2025 겨울 세미나

Hand Pose Generation / Egocentric View Generation

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

Presented By
MinSuh Song

Outline

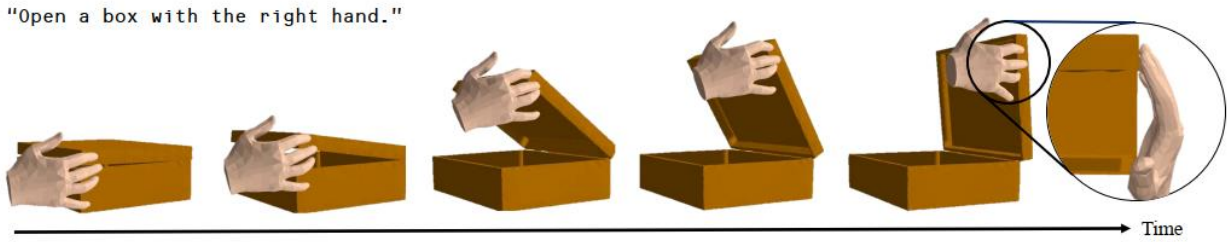
- Junuk Cha, Jihyeon Kim et al. **“Text2HOI: Text-guided 3D Motion Generation for Hand-Object Interaction.”** CVPR, 2024

- Bolin Lai, Xiaoliang Dai, et al. **“LEGO: Learning Egocentric Action Frame Generation via Visual Instruction Tuning.”** ECCV, 2024

Text2HOI: Text-guided 3D Motion Generation for Hand-Object Interaction

Text2HOI

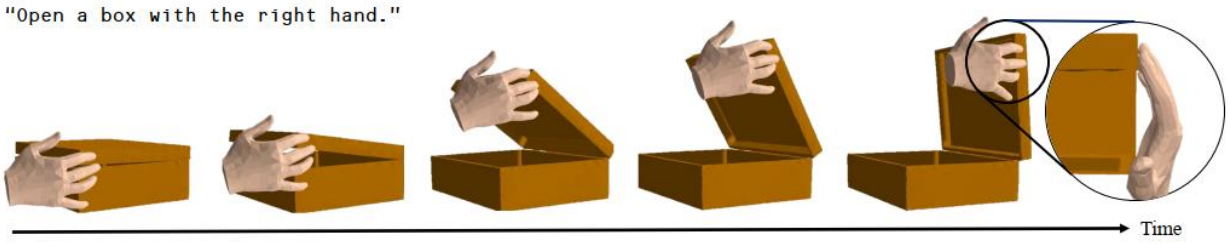
- Introduction
  - Text와 object mesh를 입력 받아 3D hand-object interaction을 생성하는 논문
  - Hand-object interaction은 많은 parameter에 의해 결정
    - Hand type: Left, Right
    - Object category
    - Object shape, scale
    - Contact regions
  - 이런 복잡한 연산을 요구하는 motion을 생성하기 위해 두 subtask로 분리
    - Object contact map generation
    - Hand-object motion generation



<Text2HOI의 정성적 결과>

Text2HOI

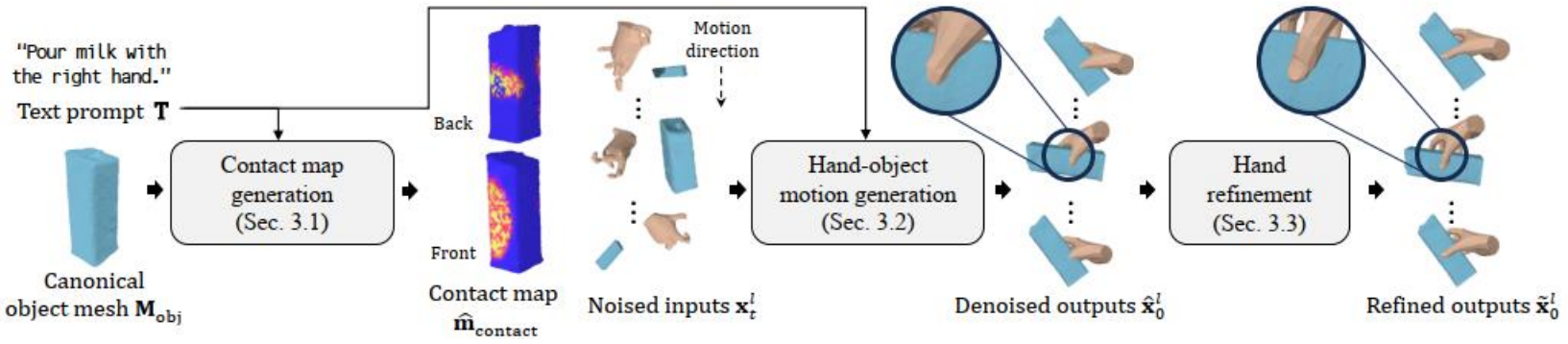
- Introduction
  - 3D Hand-Object Motion Generation



<Text2HOI의 정성적 결과>

Text2HOI

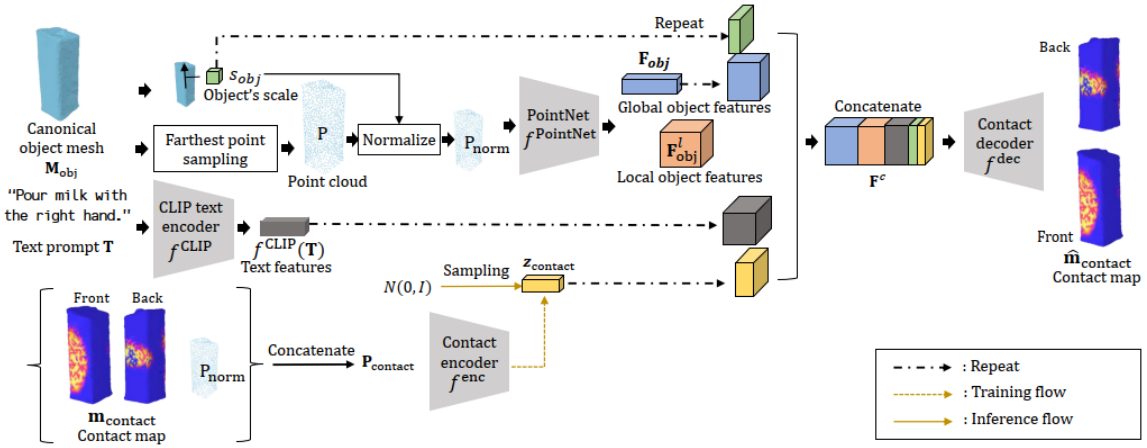
- Introduction
  - Contact map generation
    - Text prompt와 canonical object mesh를 입력 받아, mesh와 손이 접촉하는 contact map을 생성
  - Hand-object motion generation
    - 예측된 contact map과 text prompt를 이용해 noised input에서 noise를 제거함으로써 denoised output을 생성
  - Hand refinement
    - 손과 object가 서로 penetrate하지 않고 적당한 거리에서 접촉하도록 refine



<Text2HOI의 전체적인 pipeline>

Text2HOI

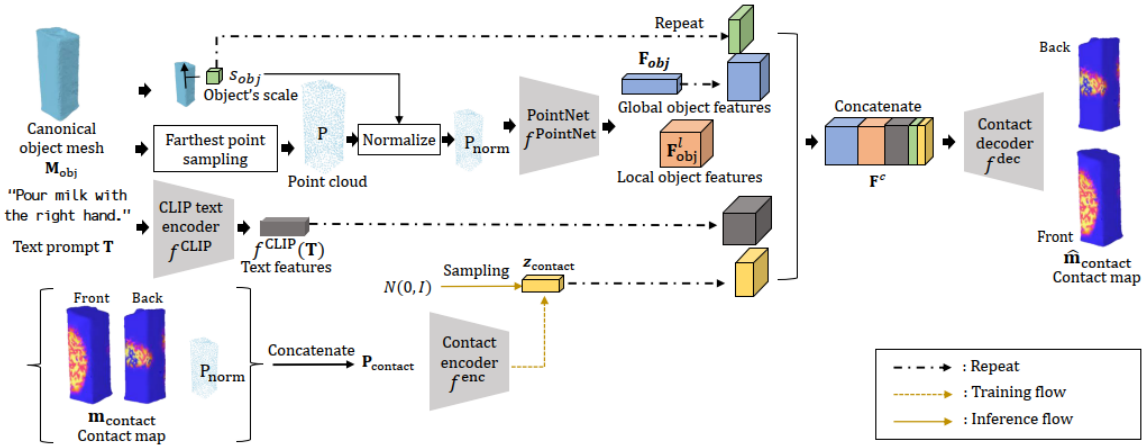
- Contact map generation
  - Object의 표면의 각 지점에 대해 손과 접촉할 확률을 나타내는 3D probability map
    - 손과 object가 상호작용하는 위치와 방식을 정의, 이후의 motion generation 과정에서 guidance의 역할을 함
  - Input으로 object의 canonical mesh M_{obj}와 text prompt T가 입력
    - M_{obj}로부터 물체의 크기를 나타내는 scale s_{obj}를 계산



< Contact map generation의 구조 >

Text2HOI

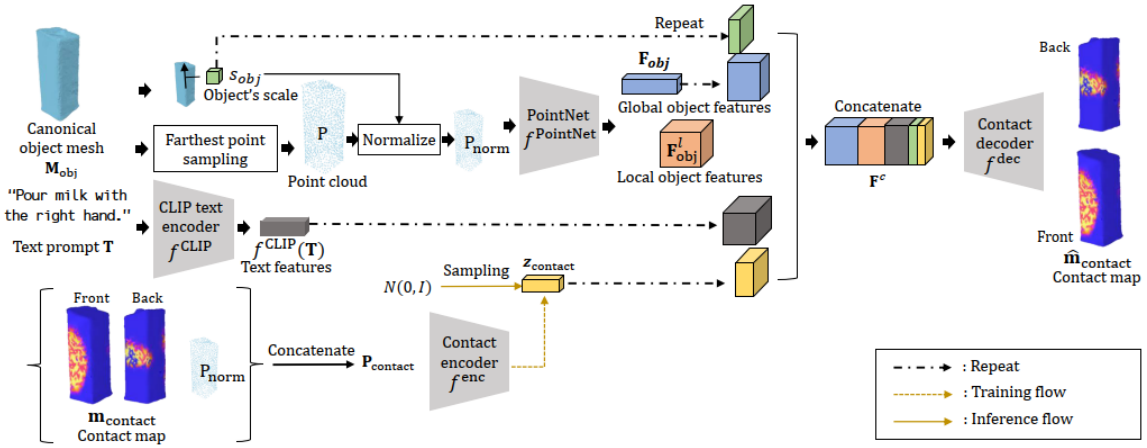
- Contact map generation
  - Step 1: Object의 3D mesh에서 Farthest Point Sampling (FPS) 알고리즘을 사용해 N개의 points(P)를 sampling
    - Object의 크기에 따라 point cloud P를 $P_{norm} = \frac{P}{s_{obj}}$로 normalize
      - s_{obj}는 object mesh의 중심으로부터 가장 먼 vertex까지의 거리
  - Step 2: Text prompt는 CLIP encoder를 사용해서 text feature vector로 변환



< Contact map generation의 구조 >

Text2HOI

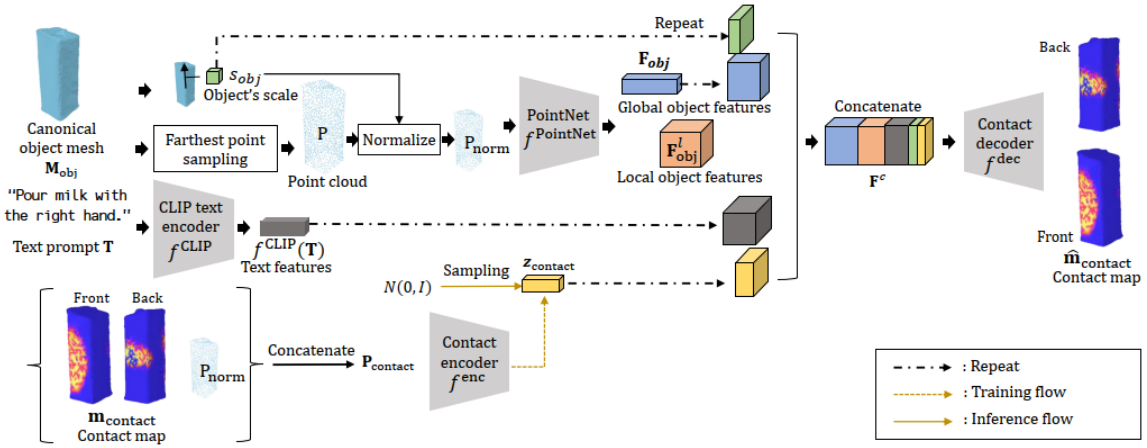
- Contact map generation
  - Step 3: PointNet[1)]을 사용하여 sampling 된 point cloud (P_{norm})으로부터 local feature와 global feature를 추출
    - Local object feature (F_{obj}^{local})
      - Object 표면에서 각 point에 대한 지역적 특징을 추출
    - Global object feature (F_{obj})
      - Object의 전체적인 구조에 대한 feature 추출



< Contact map generation의 구조 >

Text2HOI

- Contact map generation
  - Step 4: object feature (F_{obj}^{local}, F_{obj}), text feature vector, noise vector $z_{contact}$를 입력 받아서 VAE 기반 contact map prediction network에 입력
    - $z_{contact}$는 ground truth contact map과 P_{norm}을 이용해서 학습
    - Inference 할 때는 Gaussian Distribution에서 sampling된 $z_{contact}$를 input에 추가
  - Concatenate된 input F^C는 decoder f_{dec}를 통해 최종 contact map $\hat{m}_{contact}$을 생성

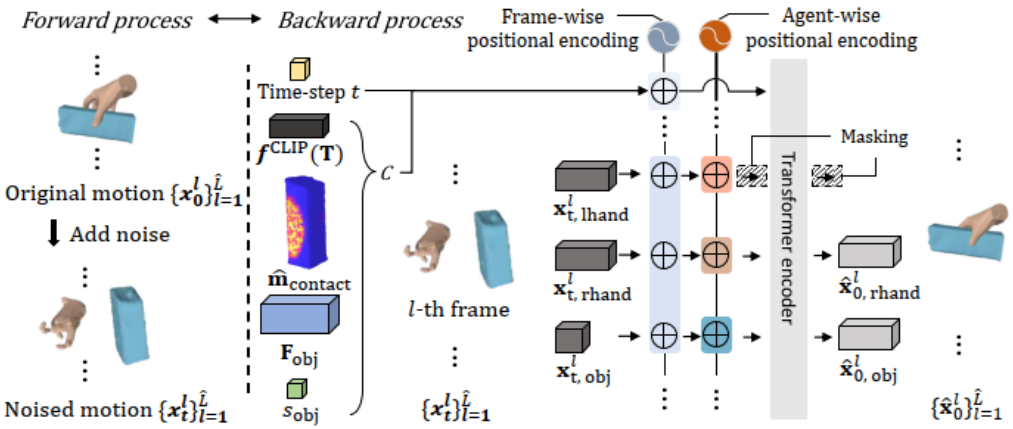


< Contact map generation의 구조 >

Text2HOI

- Motion Generation
  - Text prompt와 생성된 contact map을 기반으로 3D hand-object motion을 생성
    - Motion Generation은 diffusion process를 기반으로 작동하며, forward process와 backward process로 나뉨
  - Forward process
    - Original motion(ground truth) $\{x_0^l\}_{l=1}^{\hat{L}}$에 noise를 추가함으로써 noised motion $\{x_t^l\}_{l=1}^{\hat{L}}$을 생성

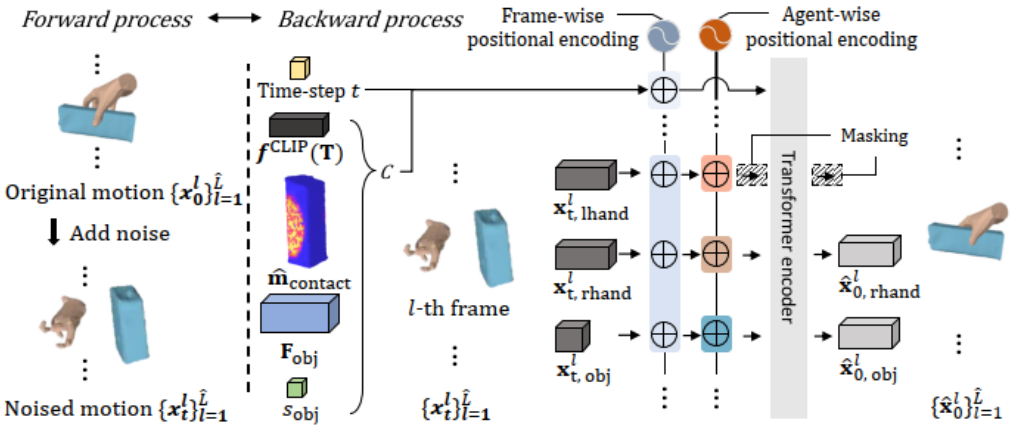
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$



< Motion Generation의 전체적인 pipeline >

Text2HOI

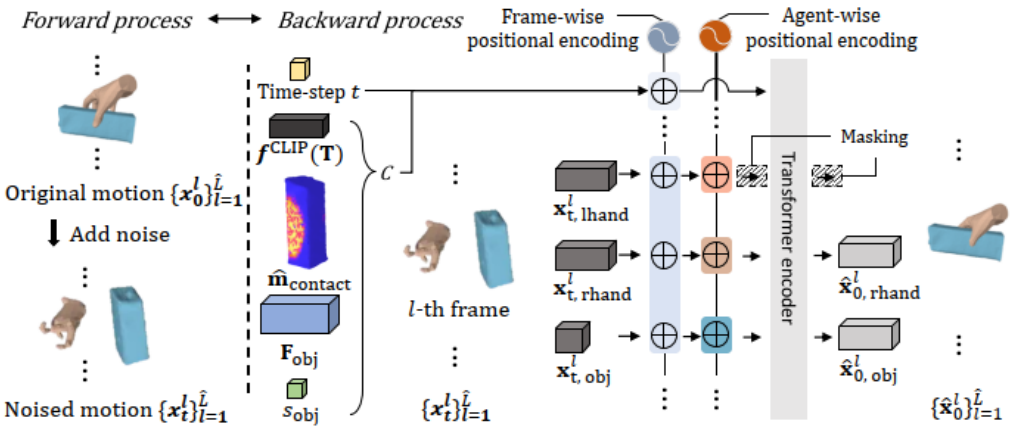
- Motion Generation
  - Backward Process
    - Forward process에서 noise가 추가된 x_t를 역으로 복원하여 재구성
    - Motion x_t와 conditions c = $\{f^{CLIP}(T), \hat{m}_{contact}, F_{obj}, s_{obj}\}$를 고려하여 denoising 과정을 진행
    - Transformer Input Generation
      - Noise가 추가된 motion x_t를 transformer 모델에서 학습 가능한 형식으로 변환
    - Conditional Input Generation
      - Conditions c를 생성하여 motion이 text prompt 등 조건에 맞도록 보장



< Motion Generation의 전체적인 pipeline >

Text2HOI

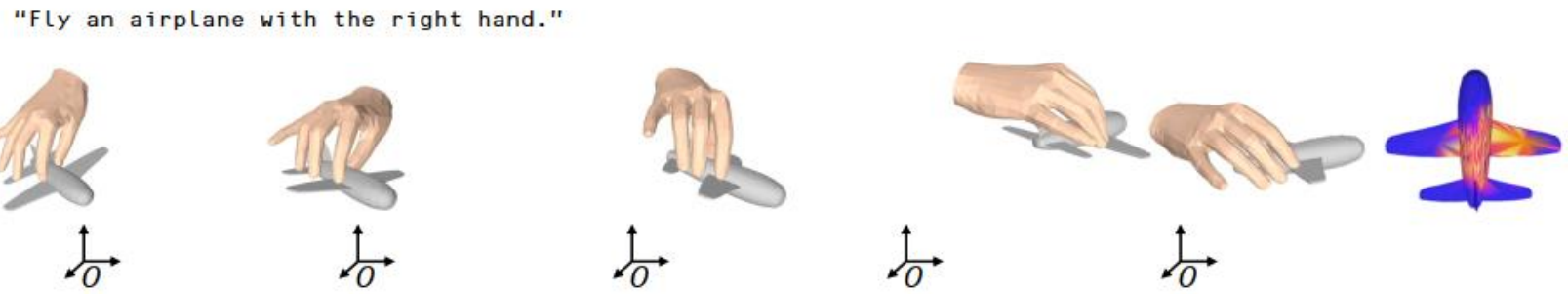
- Motion Generation
  - Backward Process - Transformer Input Generation
    - 3D hand-object motion은 $x_t = \{x^l_{t,lhand}, x^l_{t,rhand}, x^l_{t,object}\}$로 표현
      - l번째 frame의 왼손, 오른손, 물체의 feature를 의미
    - 각 time step에서 transformer model이 입력(x_t)을 받아서 다음 step (x_{t-1}) 을 생성
      - Frame-wise positional encoding
      - Agent-wise positional encoding



< Motion Generation의 전체적인 pipeline >

Text2HOI

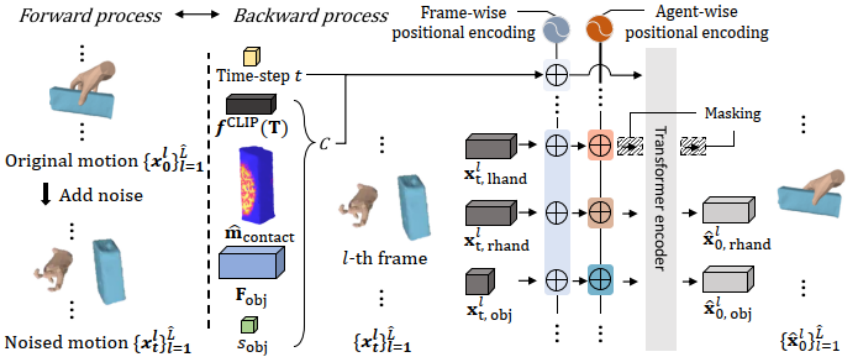
- Motion Generation
  - Backward Process - Transformer Input Generation
    - Frame-wise positional encoding
      - Frame 간 시간적 순서 정보를 encoding
      - 그림에서 시간 순서가 transformer model에 명확히 전달 되도록 상대적 거리를 수학적으로 encoding
    - Agent-wise positional encoding
      - Transformer가 agent(오른손, 왼손, 물체) 사이의 관계를 학습하기 위함
      - 각 agent에 고유한 positional encoding value를 부여하여 input data를 agent 별로 구분할 수 있도록 함



<Text2HOI의 정성적 결과 예시 >

Text2HOI

- Motion Generation – 전체적 흐름
  - Forward process에서 x_0에 noise를 추가하여 x_t를 생성
    - $x_t = \{x^l_{t,lhand}, x^l_{t,rhand}, x^l_{t,object}\}$
  - Conditional Input Generation을 통해 text와 object 정보를 바탕으로 conditions c 생성
    - $c = \{f^{CLIP}(T), \hat{m}_{contact}, F_{obj}, s_{obj}\}$
  - Transformer Input Generation에서 positional encoding
    - $x_{t,input} = x_t + PE_{frame} + PE_{agent}$
  - Noised motion $x_{t,input}$와 conditions c를 사용해 denoising 진행
    - $x_{t-1} = f_{THOI}(x_{t,input}, t, c)$



< Motion Generation의 전체적인 pipeline >

Text2HOI

- Hand Refinement Network
  - 생성된 hand-object interaction의 물리적 타당성과 자연스러움을 향상
  - 생성된 motion에서 발생할 수 있는 문제점
    - Hand가 object를 penetrate하거나 제대로 접촉하지 않는 비현실적인 결과 발생 가능
    - 관절 제한 및 자연스러운 motion이 유지되지 않을 수 있음
  - $L_{refine} = L_{simple} + L_{penet} + \lambda L_{contact}$
    - λ: Contact loss의 가중치 (λ=5)

  - Simple L2 loss
    - 모델 출력과 실제 hand motion 사이의 차이를 최소화
    - $L_{simple} = \|\tilde{x}_{hand} - x_{hand}\|^2$
      - $\tilde{x}_{hand}$: predicted hand motion
      - x_{hand}: ground-truth hand motion

Text2HOI

- Hand Refinement Network
  - Penetration loss
    - 손의 표면이 물체 내부로 penetrate하지 않도록 학습
    - $L_{penet} = l_{left} \cdot \left\| d(v_{lhand}, \hat{p}_{obj}) \right\|^2 + l_{right} \cdot \left\| d(v_{rhand}, \hat{p}_{obj}) \right\|^2$
      - v_{lhand}, v_{rhand}: 물체 표면 내부로 penetrate한 손의 표면 정점
      - $l_{left},\ l_{right}$: 왼손, 오른손을 판별하는 indicator function
      - $\hat{p}_{obj}$: 물체의 가장 가까운 표면의 점
  - Contact loss
    - 손과 물체 사이의 접촉 영역을 강화
    - $L_{contact} = l_{left} \cdot \left\| d(j_{lhand}, \hat{c}_{obj}) \right\|^2 + l_{right} \cdot \left\| d(j_{rhand}, \hat{c}_{obj}) \right\|^2$
      - j_{lhand}, j_{rhand}: 물체와 threshold τ보다 거리가 가까운 손의 관절
      - $\hat{c}_{obj}$: 손과 가장 가까운 물체의 점

Text2HOI

- Experiments
  - Datasets
    - H2O, GRAB
      - Hand-object interaction dataset
      - Two hands and one object
      - Dataset에서 주어진 action label을 이용하여 “{action} {object category} with {hand type}”를 자동 생성
        - ✓“Open a box with right hand”
    - ARCTIC
      - RGB 이미지에서 손과 물체의 3D reconstruction을 위해 공개된 dataset
      - Action label이 주어지지 않기에 직접 수동으로 text prompt를 작성
    - 모든 dataset은 MANO hand parameters, object meshes, object의 rotation과 translation 정보를 제공

Text2HOI

- Experiments
  - Evaluation metrics
    - Accuracy
    - Frechet Inception Distance (FID)
      - Feature space에서 실제와 생성된 motion 간의 거리(차이)
    - Diversity
      - 생성한 모든 motion sample 간의 차이를 기반으로 다양성 평가
    - Multimodality
      - 특정 text prompt에 대해 다양한 결과를 생성할 수 있는 model의 능력을 평가
    - Physical Realism
      - 생성된 motion이 얼마나 현실적인지 평가
      - ManipNet[1)]에서 사용된 physical model에 따라 각 프레임에 대해 real(1), unreal(0) 부여

Text2HOI

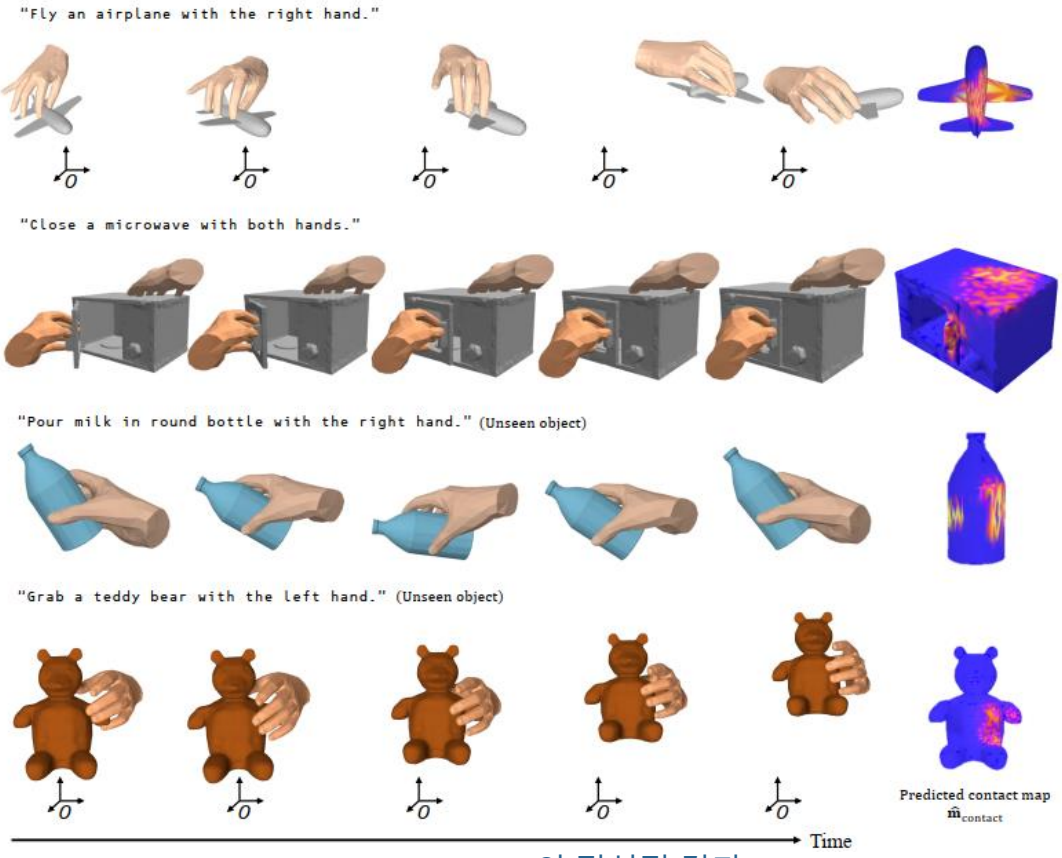
- Experiments
  - Results

Method	Accuracy (top-3) ↑	FID ↓	Diversity →	Multimodality ↑	Physical realism ↑
			H2O		
GT	0.9920 ± 0.0003	-	0.6057 ± 0.0050	0.2067 ± 0.0024	0.4790 ± 0.0002
T2M† [8]	0.6463 ± 0.0014	0.3439 ± 0.0006	0.3475 ± 0.0040	0.0634 ± 0.0022	0.3890 ± 0.016
MDM† [27]	0.5832 ± 0.0011	0.3015 ± 0.0011	0.5127 ± 0.0054	0.1738 ± 0.0049	0.5572 ± 0.0013
IMOS† [6]	0.5518 ± 0.0026	0.2945 ± 0.0011	0.4076 ± 0.0056	0.1798 ± 0.0115	0.3532 ± 0.0026
Ours	**0.8295 ± 0.0015**	**0.1744 ± 0.0013**	**0.5365 ± 0.0073**	**0.2469 ± 0.0081**	**0.7574 ± 0.0022**
			GRAB		
GT	0.9994 ± 0.0001	-	0.8557 ± 0.0054	0.4390 ± 0.0045	0.8084 ± 0.0002
T2M† [8]	0.1897 ± 0.0007	0.7886 ± 0.0005	0.5712 ± 0.0078	0.0964 ± 0.0027	0.5844 ± 0.0002
MDM† [27]	0.5127 ± 0.0009	0.6023 ± 0.0011	0.8012 ± 0.0054	0.5194 ± 0.0145	0.7382 ± 0.0004
IMOS† [6]	0.4097 ± 0.0005	0.6147 ± 0.0003	0.6861 ± 0.0060	0.2845 ± 0.0036	0.6418 ± 0.0014
Ours	**0.9218 ± 0.0010**	**0.3017 ± 0.0004**	**0.8351 ± 0.0061**	**0.5216 ± 0.0131**	**0.8839 ± 0.0005**
			ARCTIC		
GT	0.9997 ± 0.0001	-	0.5916 ± 0.0037	0.3279 ± 0.0038	0.9573 ± 0.0000
T2M† [8]	0.5234 ± 0.0015	0.3599 ± 0.0005	0.3301 ± 0.0023	0.0849 ± 0.0017	0.0143 ± 0.0001
MDM† [27]	0.5572 ± 0.0012	0.3025 ± 0.0006	0.4984 ± 0.0039	0.2632 ± 0.0065	0.7043 ± 0.0009
IMOS† [6]	0.8190 ± 0.0039	0.1826 ± 0.0005	0.5702 ± 0.0039	0.2741 ± 0.0049	0.7569 ± 0.0023
Ours	**0.9205 ± 0.0012**	**0.1329 ± 0.0006**	**0.5758 ± 0.0042**	**0.3170 ± 0.0068**	**0.8760 ± 0.0009**

< Text2HOI와 기존 methods와의 성능 비교표 >

Text2HOI

- Experiments
  - Results

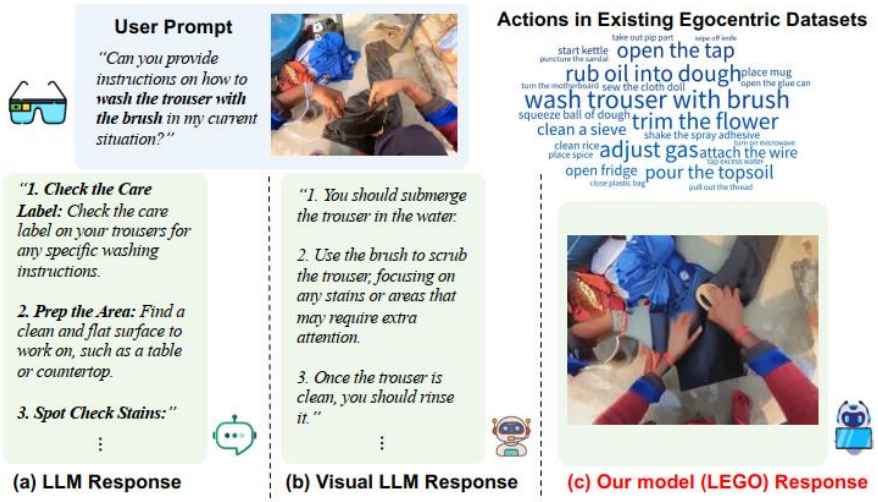


< Text2HOI의 정성적 결과 >

LEGO: Learning Egocentric Action Frame Generation via Visual Instruction Tuning

LEGO

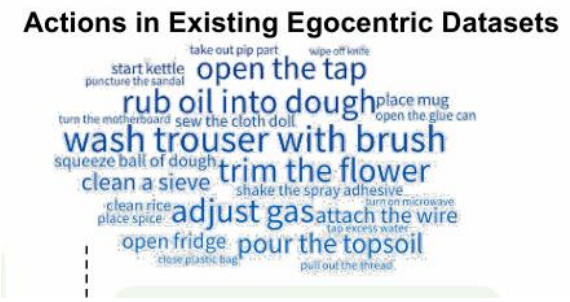
- Introduction
  - User에게 특정 task의 수행 방법을 알려주는 데 기존 LLM 모델은 한계점이 존재
    - LLM은 task의 일반적인 방법은 알려주지만, user의 상황에 맞는 지침은 제공하지 못함
    - VLLM의 경우 상황에 맞는 description을 알려주지만, 단어 기반이라 직관적이지 못함
      - 텍스트는 이미지보다 뇌에서 처리 속도가 느림
  - 본 논문은 user의 현재 시점에서 행동을 시각적으로 설명하는 egocentric hand image를 생성하여, 사용자가 행동을 이해하고 task를 수행하는 과정을 더 직관적으로 지원



<LLM, VLLM, LEGO의 답변 비교>

LEGO

- Introduction
  - Input
    - 특정 action을 수행하는 방법을 묻는 input query
    - 현재 상황의 egocentric view image
  - 한계점
    - 현재 egocentric datasets의 action annotation은 명사나 동사와 같은 단어로만 이루어짐
      - 이런 단순한 label만으로는 세부적인 작업이나 객체의 움직임을 학습하기 어려움
    - 현재 diffusion model들은 exocentric한 이미지들로 학습
      - Egocentric 이미지는 시각적 구조나 표현 방법이 다르므로, 기존 모델이 이를 정확히 이해하고 생성하는데 어려움



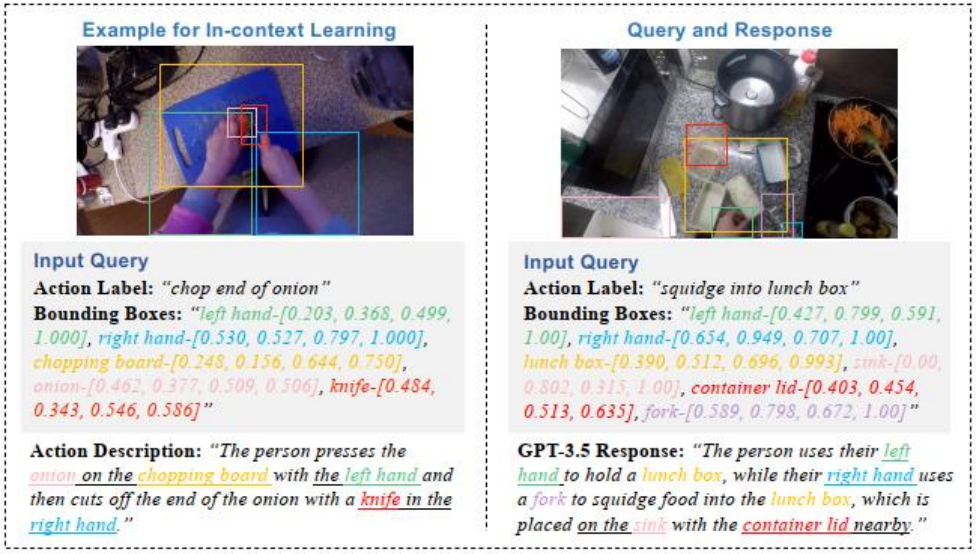
< 기존 Ego dataset의 action annotation >

LEGO

- Introduction
  - 본 논문이 한계점을 해결하는 방법
    - Data Curation
      - Visual Large Language Model (VLLM)을 visual instruction tuning을 통해 학습시켜, 기존 단순한 label에서 detailed한 작업 설명을 생성
      - “Chop onion” -> “The person presses the onion on the chopping board with the left hand then cuts off the end of the onion with a knife right hand.”
    - Visual Instruction Tuning
      - Finetuned VLLM을 통해 생성된 image 및 text embedding을 diffusion model에 입력하여 domain gap 해소

LEGO

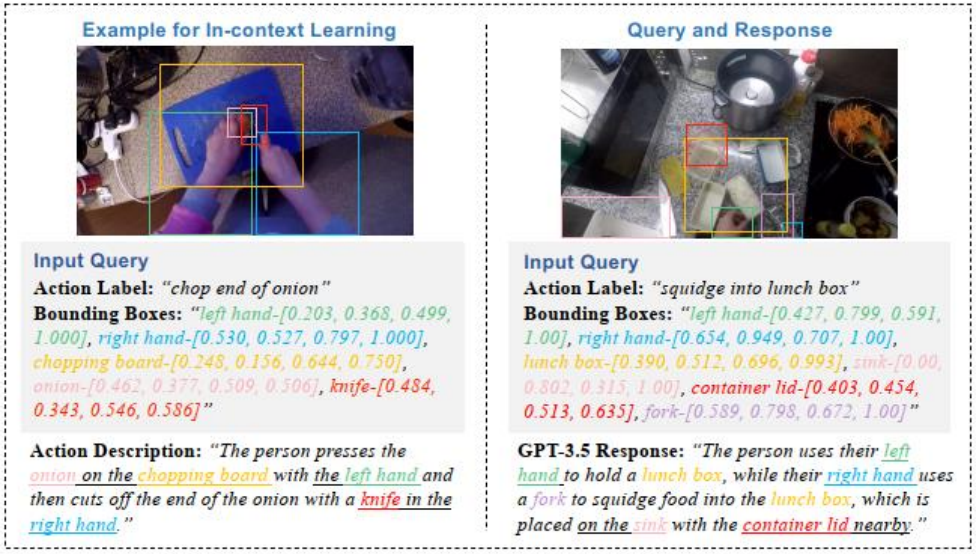
- Egocentric Visual Instruction Tuning
  - Data Curation for Visual Instruction Tuning
    - GPT-3.5를 사용하여 egocentric image에 대한 action label과 객체의 bounding box를 입력 받아 더 세부적이고 풍부한 action description을 생성
      - 기존 Ego4D나 Epic-Kitchen dataset의 경우, 매우 간단한 action label, 객체의 bounding box만을 제공
      - GPT-3.5를 원하는 출력이 나오도록 In-context learning을 사용



< GPT-3.5를 이용한 data curation >

LEGO

- Egocentric Visual Instruction Tuning
  - Data Curation for Visual Instruction Tuning
    - In-context learning
      - 몇 가지의 예제를 입력해서 GPT가 유사한 스타일과 구조로 출력을 생성하도록 유도하는 방법
      - In-context learning을 위한 초기 action description은 직접 작성하여 사용



< GPT-3.5를 이용한 data curation >

LEGO

- Egocentric Visual Instruction Tuning
  - Data Curation for Visual Instruction Tuning
    - In-context learning
      - 아래 그림과 같은 prompt를 이용하여 action label, object bounding box, manually annotated action description을 GPT에 입력
      - In-context learning 결과 생성된 action description을 이용해 visual instruction tuning에 사용

System: *You are an AI assistant that provides a description of an image based on the action and object context. The action consists of a verb and nouns. Each object location is represented by a bounding box. For each bounding box, four numbers are provided in brackets – they are [x-coordinate of top-left, y-coordinate of top-left, x-coordinate of bottom-right, y-coordinate of bottom-right]. The origin is at the top-left of each frame. The x-axis is on the top and the y-axis is on the left. All coordinates are normalized to the range from 0 to 1. This information can be used to infer the spatial relation of hands and objects. Note that the detailed narration is in a natural and holistic style. Please add more details in the action. For example, try to describe which hand is used in each action like "with right hand" or "using left hand". Try to describe the spatial relation of these objects like "on the right", "on the left", "from ... to ..." or use some spatial words like "in", "on", "out", "front", "back", etc. Describe the image in only one sentence. Do not describe objects or actions that are not presented in action or objects locations context. Many examples are provided for learning and an additional example is provided for inference.*

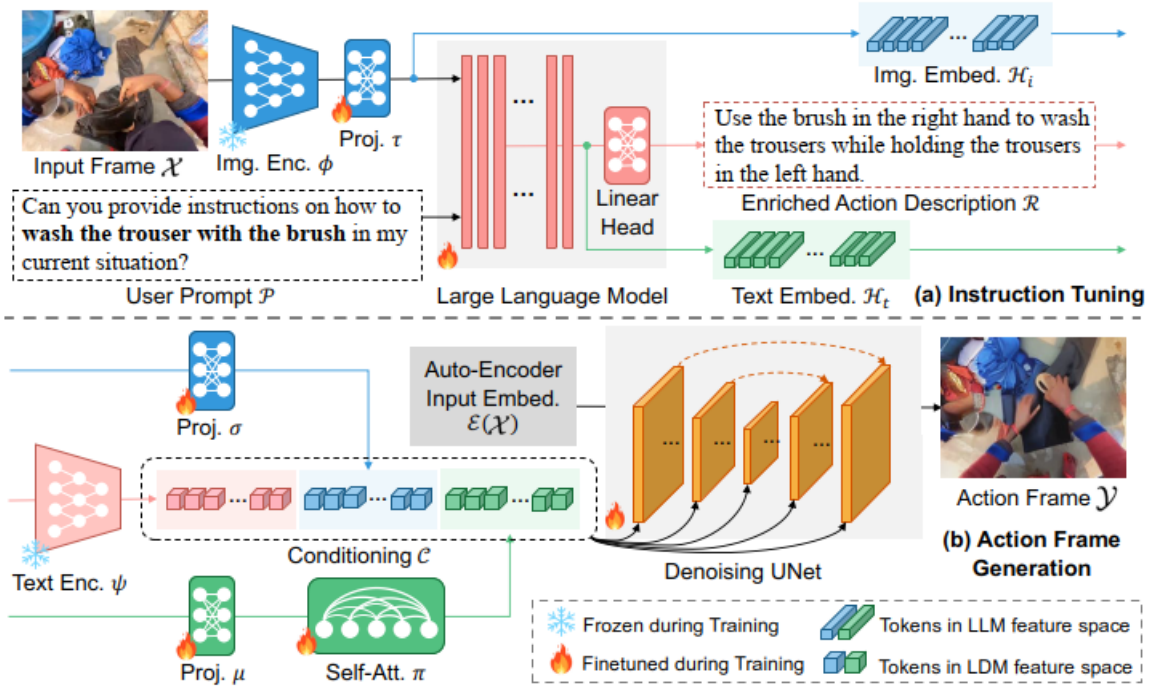
User: *Examples for learning: (1) {Example-1} (2) {Example-2} ... (12) {Example-12}*

User: *Example for inference: {Inference Example}*

< In-context learning을 위해 GPT의 입력하는 prompt >

LEGO

- Egocentric Visual Instruction Tuning
  - Visual Instruction Tuning
    - Egocentric image와 GPT-3.5의 action description을 이용하여 세부 설명 작업을 생성하도록 VLLM을 학습
      - 본 논문에서는 $LLaVA^{1)}$를 baseline으로 함

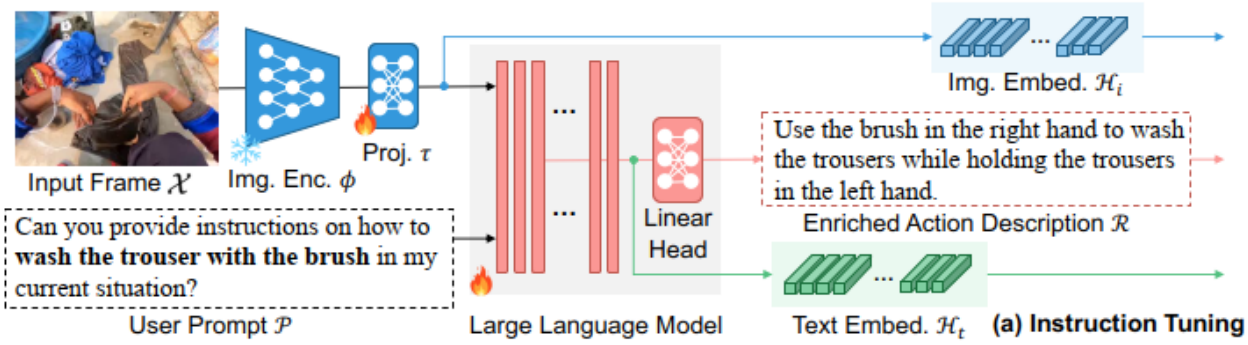


< LEGO의 전체적인 구조 >

LEGO

- Egocentric Visual Instruction Tuning
  - Visual Instruction Tuning
    - Egocentric input frame χ는 CLIP image encoder ϕ를 사용하여 feature를 추출
    - 이후 linear projection τ를 시켜서 image embedding H_i을 text embedding 공간으로 project

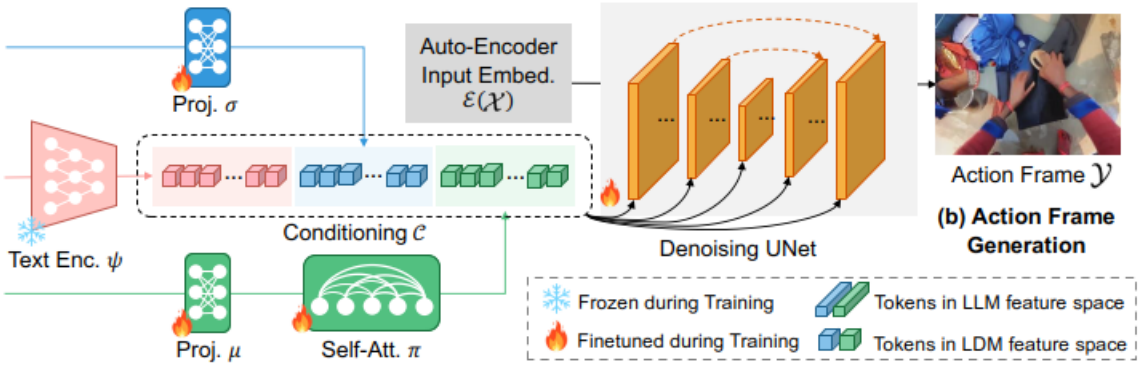
      $H_i = \tau(\phi(X))$
    - Prompt template에 action label annotation을 추가하여 user prompt P를 생성
    - 이후 tokenize하여 image embedding H_i와 함께 LLM의 input으로 입력
    - LEGO의 LLM은 GPT-3.5의 action description을 ground truth로 하여 학습



< Visual Instruction Tuning >

LEGO

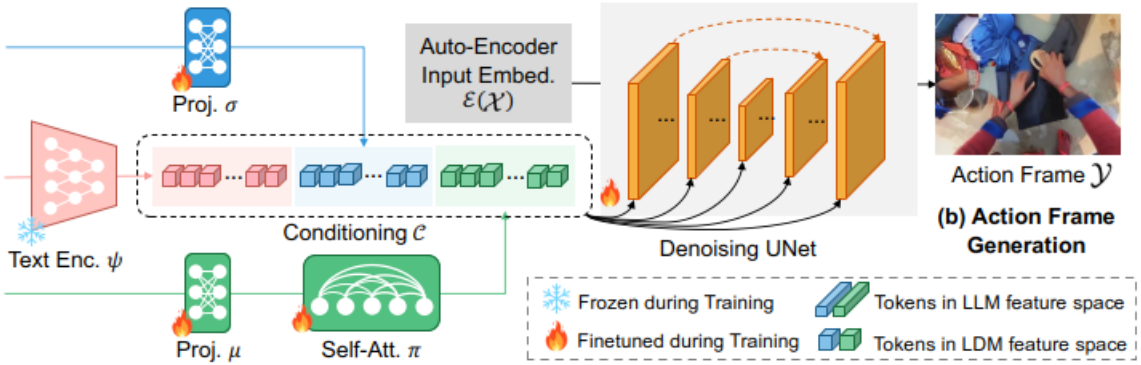
- Egocentric Action Frame Generation
  - Diffusion model의 egocentric domain gap issue를 해결하기 위해 LDM conditioning 추가
    - 본 논문은 Latent Diffusion Model (LDM)을 사용
    - Finetuned VLLM에서 생성된 detailed action description $\mathcal{R}$을 CLIP text encoder ψ로 추출 $\psi(\mathcal{R})$
    - Image embedding H_i를 linear layer σ를 거쳐서 LDM 입력에 적합한 차원으로 설정 $\sigma(H_i)$
    - Text embedding H_t를 projection layer μ를 거친 후, 전체적인 text의 의미를 보장하기 위해서 self-attention layer π를 통과 시킴 $\pi(\mu(H_t))$
    - 결과적으로 LDM conditioning $C = [\psi(\mathcal{R}), \sigma(H_i), \pi(\mu(H_t))]$



< Action Frame Generation >

LEGO

- Egocentric Action Frame Generation
  - 기존 LDM의 특성에 맞게 초기 egocentric image χ를 latent space로 encoding
    - Pre-trained autoencoder $\mathcal{E}$를 사용 $\mathcal{E}(\chi)$
  - Diffusion의 denoising Unet에서 cross-attention 진행
    - Query: $\mathcal{E}(\chi)$
    - Key: LDM conditioning $C = [\psi(\mathcal{R}), \sigma(H_i), \pi(\mu(H_t))]$
  - 최종적으로 pre-trained decoder를 통해 action frame $\mathcal{Y}$ 생성



< Action Frame Generation >

LEGO

- Experiments
  - Datasets
    - Ego4D
      - Egocentric view video dataset
      - 인간의 일상활동을 다양한 background에서 촬영한 dataset
    - Epic-Kitchens-100
      - Egocentric view video dataset
      - 요리 활동을 중심으로 egocentric 시점에서 촬영한 dataset
    - 데이터 셋에서 특정 행동이 시작되기 전 프레임과 행동이 종료된 후의 프레임을 추출
      - 행동이 종료된 후의 프레임을 ground truth로 사용

LEGO

- Experiments
  - Evaluation metrics
    - Egocentric Video-Language Pre-training (EgoVLP)
      - Egocentric video와 text간의 관계를 학습하는 pre-trained multimodal
      - Text prompt와 생성된 이미지 사이의 일치도를 egocentric 관점에서 평가
    - EgoVLP+
      - EgoVLP의 확장 버전, text-video 사이의 관계를 더 잘 modeling함
    - CLIP
      - CLIP 모델을 사용하여 text와 image 사이의 의미적 유사성을 측정
    - Frechet Inception Distance (FID)
      - 생성된 이미지와 실제 이미지 간의 차이를 측정하는 지표
    - Peak Signal-to-Noise Ratio (PSNR)
      - Image의 재구성 품질을 평가하기 위한 지표
    - Learned Perceptual Image Patch Similarity (LPIPS)
      - 두 이미지 간의 차이를 통해 시각적 유사성을 측정하는 지표

LEGO

- Experiments
  - Results
    - LEGO가 기존 모델들보다 text prompt와 더 적합한 이미지를 생성함
      - 실제 image와 상대적으로 더 유사한 이미지를 생성하고, 오차도 적음

	Methods	EgoVLP	EgoVLP$^+$	CLIP	FID ↓	PSNR	LPIPS ↓
Ego4D	ProxEdit [26]	44.51	72.68	68.17	33.01	11.88	40.90
	SDEdit [59]	50.07	72.90	73.35	33.35	11.81	41.60
	IP2P [6]	62.19	78.84	78.75	24.73	12.16	37.16
	LEGO	**65.65**	**80.44**	**80.61**	**23.83**	**12.29**	**36.43**
E-Kitchens	ProxEdit [26]	32.27	52.77	65.80	51.35	11.06	46.35
	SDEdit [59]	33.84	56.80	74.76	27.41	11.30	43.33
	IP2P [6]	42.97	61.06	77.03	**20.64**	11.23	40.82
	LEGO	**45.89**	**62.66**	**78.63**	21.57	**11.33**	**40.36**

< LEGO와 기존 model들의 성능 비교표 >

LEGO

- Experiments
  - Results
    - Ego4D test

< LEGO와 기존 model들의 정성적 평가 >

LEGO

- Experiments
  - Results
    - Epic-Kitchens test

< LEGO와 기존 model들의 정성적 평가 >

감사합니다