ASE: Large-Scale Reusable Adversarial Skill Embedding for Physically Simulated Characters

Paper Review

ASE: Large-Scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters

XUE BIN PENG, University of California, Berkeley, USA and NVIDIA, Canada
YUNRONG GUO, NVIDIA, Canada
LINA HALPER, NVIDIA, Canada
SERGEY LEVINE, University of California, Berkeley, USA
SANJA FIDLER, University of Toronto, Canada and NVIDIA, Canada

Introduction

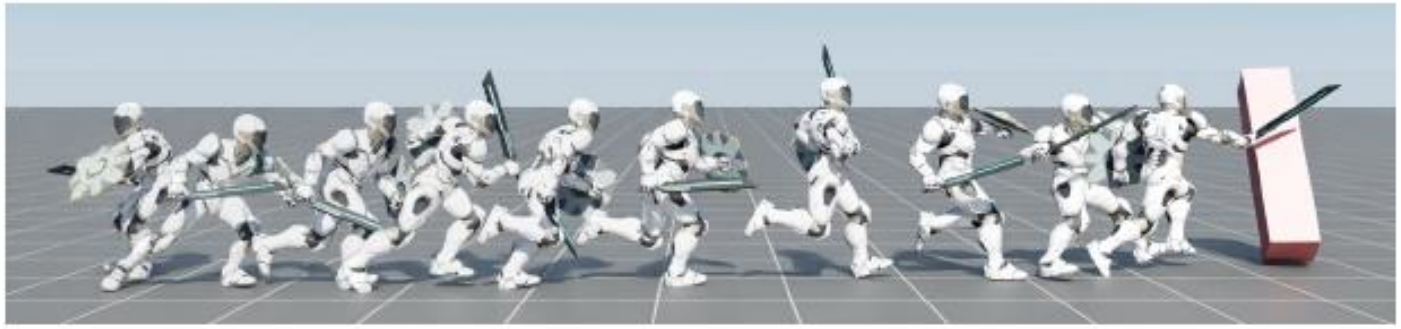


Fig. 1. Our framework enables physically simulated characters to learn versatile and reusable skill embeddings from large unstructured motion datasets, which can then be applied to produce life-like behaviors for new tasks. Here, a character is utilizing behaviors from a learned skill embedding in order to run to a target and knock it over.

◎ 다재다능(versatile)하고 재사용가능(reusable)한 control의 필요성

- 인간은 일생동안 물리환경에서 상호작용으로 다양한 움직임을 습득하고 새로운 태스크에 적용하는 방식을 사용
- 필요한 스킬을 그때그때 학습하는 것이 아니라, 원래 인간이 갖고있던 General-purpose 스킬을 이용해서 그것이 재사용 되는 방식
- 반면, 최근 물리기반 캐릭터 애니메이션은 하나의 특정 스킬을 익히기 위해 from scratch로 학습 진행하는 편

◎ 제안 방법론 – Adversarial Skill Embeddings (ASE)

- 비정형의 대형 motion clip 데이터 셋으로 부터, 일반적인 행동양식을 먼저 습득한다. (low-level)
- Low-level 컨트롤에서 학습된 모션은 하나의 특정 motion clip을 표방하진 않는다.
- Low-level 컨트롤에선 이후 High-level 컨트롤에서 사용될 추상적인 액션공간을 정의하는 용도.

Framework Overview

◎ Goal of this paper

▶ **다재다능(versatile) 하고 다양한(diverse) 스킬을 Control policy를 학습**

→ 특정 모션에만 국한하지 않고 Motion dataset의 일반적인 특징(general characteristics)을 표현하는 모션의 학습

◎ Two-stage approach

▶ **Pre-Training stage**

→ 특정 task를 타겟으로 학습이 아닌, 데이터셋의 일반적인 모션을 학습

→ 궁극적으로 **Low-level policy**를 학습 하는 것을 목표

(Mapping: latent z |-> motion in the dataset)

→ 학습 대상: Latent encoder, Discriminator, Low-level policy 모두 학습하는 단계

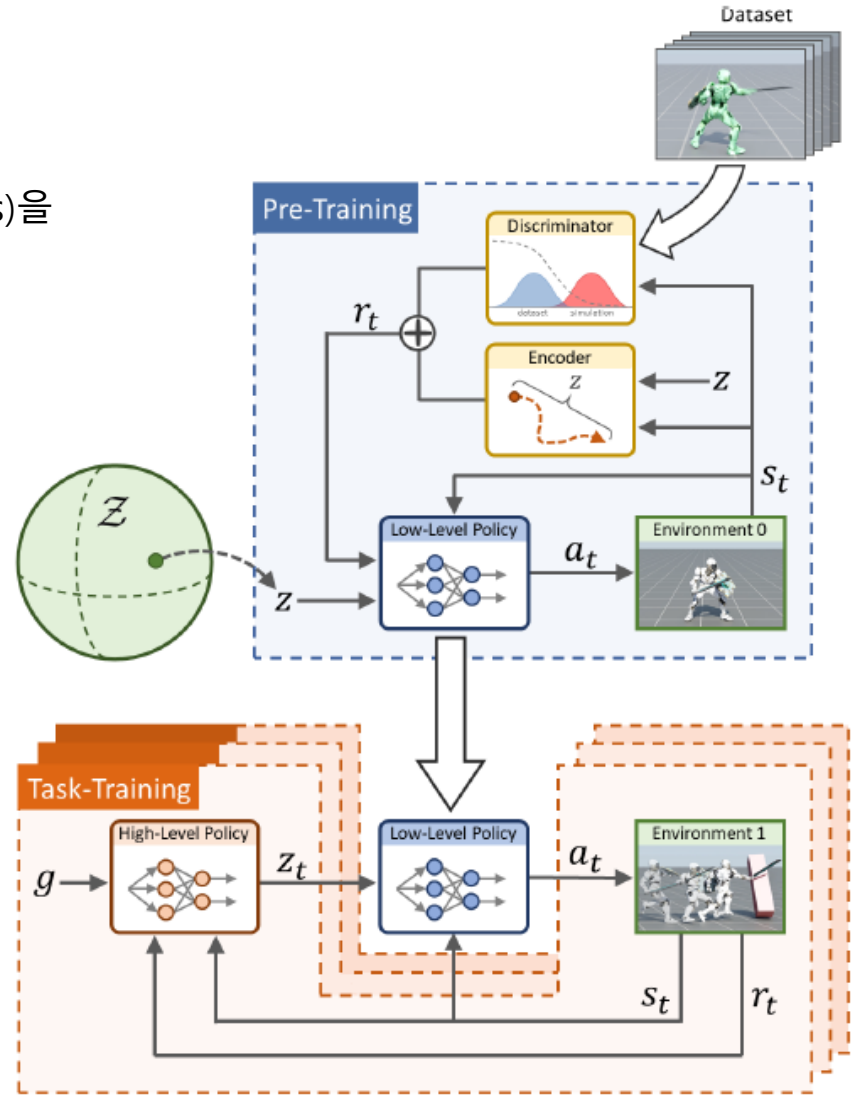
▶ **Task-Training stage**

→ g라는 특정 task에 특화된 policy를 학습

→ **High-level policy**를 학습 하는 것을 목표

→ 앞서 학습한 Low-level policy를 이용

→ 학습 대상: High-level policy 단 한개



Low-level Policy and Skill Embeddings

◎ Goal of low-level policy

▶ **지시 가능하고 (Directable) 일반화 된(Generalized) 정책**

→ Directable: latent 변수 z의 값에 따라 구분 가능하고 예측 가능한 동작이 나오게끔 policy가 학습이 되어야 함

→ Generalized: 개별 모션 보다 dataset에 드러난 모션의 전체 분포를 익혀야 함 [Adversarial Skill Embedding]

◎ Objective of Pre-training

Imitation Objective / Skill Discovery Objective

$$\max_{\pi} \; -D_{\mathrm{JS}}\left(d^{\pi}(s,s') \,\|\, d^{\mathcal{M}}(s,s')\right) + \beta\, I\left(s,s';z|\pi\right)$$

State transition induced by policy / State transition from dataset

▶ **Imitation Objective**

→ 이론상 Jensen-Shannon divergence 를 이용하여 데이터셋 M과 최대한 비슷한 모션이 생성되게끔 (JS는 계산이 쉽지 않음..)

→ 실제로는 AMP에 나왔던 것과 비슷한 결의 discriminator 사용: $\min_{D} \; -\mathbb{E}_{d^{\mathcal{M}}(s,s')}\left[\log\left(D\left(s,s'\right)\right)\right] - \mathbb{E}_{d^{\pi}(s,s')}\left[\log\left(1-D\left(s,s'\right)\right)\right]$

→ reward per timestep: $-\log\left(1-D\left(s_t,s_{t+1}\right)\right)$

▶ **Skill Discovery Objective**

→ 단순히 dataset 따라하는 Imitation Objective 만으로는 하위 태스크 (downstream task) 에서 재사용이 가능한 low-level policy 만들 수 없음

$I(s,s';z|\pi) = \mathcal{H}(s,s'|\pi) - \mathcal{H}(s,s'|z,\pi)$ ➔ 최종 reward는 $r_t = -\log\left(1-D(s_t,s_{t+1})\right) + \beta \log q\left(z_t|s_t,s_{t+1}\right)$

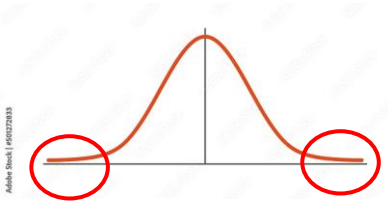
이 term을 **최대화** 함으로써 state transition이 좀 더 다양하게 만들 수 있음

이 term을 **최소화** 함으로써 특정한 스킬 z가 주어졌을때, 다른 스킬들과 구분이 되는 state transition을 만들 수 있게 한다. (주어진 z가 만들수 있는 state transition의 불확실성 감소)

Entropy를 직접 계산 사용하지 않기 위해 mutual information의 variational lower bound사용

Skill Encoder and Discriminator

◎ Latent Space 소개 및 Skill Encoder 학습

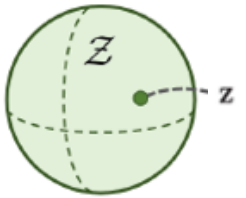


▶ Latent distribution p(z) 의 선택

→ Gaussian이 일반적이지만, unbounded하고 원점에서 먼 점이 샘플 될 경우, 비정상적인 모션으로 이어질 수 있음.

→ Bounded된 분포가 필요하고, Uniform distribution[-1,1] 같은 것이 이상적임.

→ ASE논문에서는 Hypersphere {z : ||z|| =1} 상에서 uniform한 점을 추출하는 latent 분포로 디자인 ➔ 그러면 bounded 함



→ Sampling 트릭: 정규 Gaussian 분포에서 한점 샘플 후 normalize함으로써, 마치 sphere 에서 샘플된 것 처럼 만들 수 있음:

$\bar{z} \sim \mathcal{N}(0, I)$ ▶ $z = \bar{z}/\|\bar{z}\|$ (Box-Muller 변환의 n차원 버전??)

▶ Skill Encoder의 학습

→ 인코더는 von Mises-Fisher 분포를 이용해 모델링 됨: **sphere상의 Gaussian** $q(z|s,s') = \frac{1}{Z}\exp\left(\kappa \mu_q(s,s')^T z\right)$ (분포의 평균값)

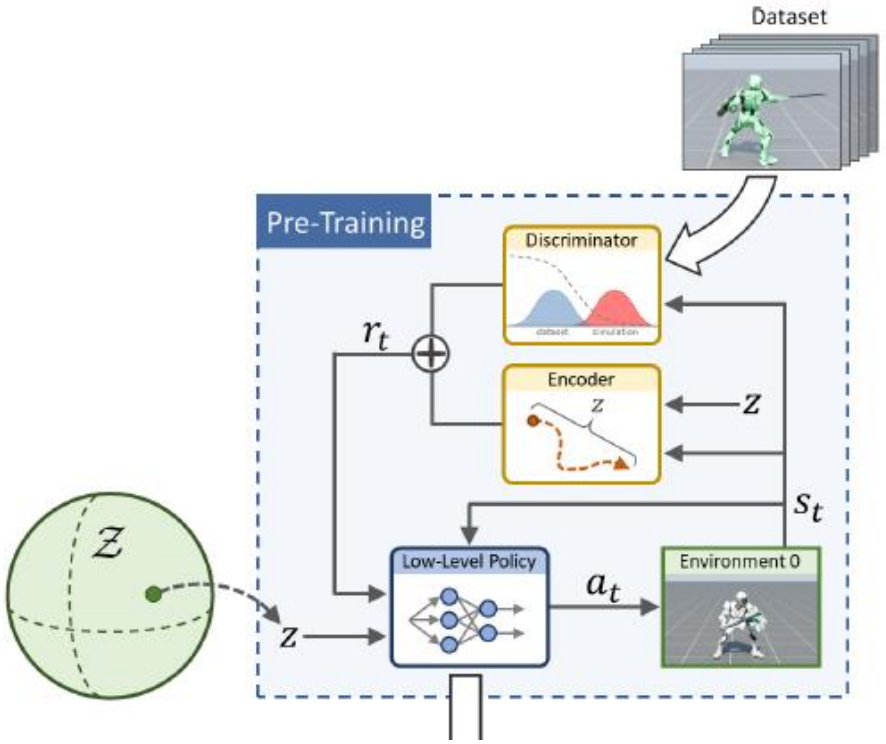
→ (z,s,s')값이 주어지면, Maximum log likelihood로 인코더 학습: $\max_q \; \mathbb{E}_{p(z)}\mathbb{E}_{d^\pi(s,s'|z)}\left[\kappa\, \mu_q(s,s')^T z\right]$

◎ Discriminator 학습

$$\min_D \; -\mathbb{E}_{d^{\mathcal{M}}(s,s')}\left[\log\left(D\left(s,s'\right)\right)\right] - \mathbb{E}_{d^{\pi}(s,s')}\left[\log\left(1-D\left(s,s'\right)\right)\right]$$
$$+ w_{gp}\,\mathbb{E}_{d^{\mathcal{M}}(s,s')}\left[\left\|\nabla_\phi D(\phi)\Big|_{\phi=(s,s')}\right\|^2\right]$$

AMP에도 등장했던 gradient penalty : training의 안정성 보장

Pre-Training Algorithm



ALGORITHM 1: ASE Pre-Training

input $\mathcal{M}$: dataset of reference motions
$D \leftarrow$ initialize discriminator
$q \leftarrow$ initialize encoder
$\pi \leftarrow$ initialize policy
$V \leftarrow$ initialize value function

while not done **do**
  $\mathcal{B} \leftarrow \emptyset$ initialize data buffer
  for trajectory $i = 1, ..., m$ **do**
    $Z \leftarrow$ sample sequence of latents $\{z_0, z_1, ..., z_{T-1}\}$ from $p(z)$
    $\tau^i \leftarrow \{s_0, a_0, s_1, ..., s_T\}$ collect trajectory with π and Z
    record Z in τ^i
    for time step $t = 0, ..., T-1$ **do**
      $r_t \leftarrow -\log(1 - D(s_t, s_{t+1})) + \beta \log q(z_t | s_t, s_{t+1})$
      record r_t in τ^i
    end for
    store τ^i in $\mathcal{B}$
  end for

  Update encoder:
  for update step = 1, ..., n **do**
    $b^\pi \leftarrow$ sample batch of K transitions $\{(s_j, s'_j, z_j)\}_{j=1}^K$ from $\mathcal{B}$
    update q according to Equation 13 using b^π
  end for

$$\max_q \quad \mathbb{E}_{p(z)} \mathbb{E}_{d^\pi(s,s'|z)} \left[\kappa \, \mu_q(s, s')^T z \right]$$

  Update discriminator:
  for update step = 1, ..., n **do**
    $b^\mathcal{M} \leftarrow$ sample batch of K transitions $\{(s_j, s'_j)\}_{j=1}^K$ from $\mathcal{M}$
    $b^\pi \leftarrow$ sample batch of K transitions $\{(s_j, s'_j)\}_{j=1}^K$ from $\mathcal{B}$
    update D according to Equation 14 using $b^\mathcal{M}$ and b^π
  end for

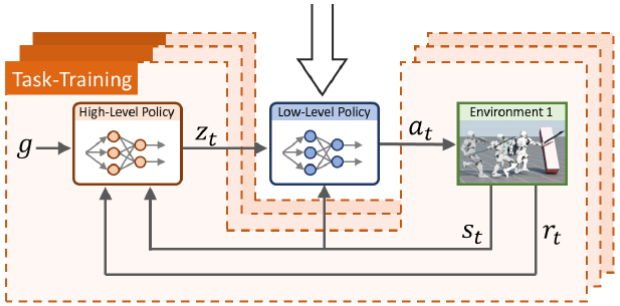
$$\min_D \quad -\mathbb{E}_{d^\mathcal{M}(s,s')} \left[\log(D(s, s')) \right] - \mathbb{E}_{d^\pi(s,s')} \left[\log(1 - D(s, s')) \right] + w_{gp} \mathbb{E}_{d^\mathcal{M}(s,s')} \left[\left\| \nabla_\phi D(\phi) \big|_{\phi=(s,s')} \right\|^2 \right]$$

    update V and π according to Equation 15 using data from $\mathcal{B}$
end while

Reward (PPO사용)

High Level Policy

◎ High-level action space



▶ **High-level policy: input (goal and state) & output (latent):** $\omega(\mathbf{z}|\mathbf{s},\mathbf{g})$

→ 특이하게도 action space가 (unnormalized) latent space: $\bar{\mathbf{z}} \sim \mathcal{N}(0, I)$

▶ **학습 경과에 따른 skill sample pool의 차이**

→ 학습 초반에는 exploration 중시: 다양한 skill을 uniform하게 시도해본다. (*그림 –좌상단*)

→ 학습이 진행될 수록 주어진 태스크에 더 맞는 쪽으로 이동 (*그림 –우상단*) ➔ assign more likelihoods

→ 학습 막바지에는 exploitation 중시: origin에서 굉장히 멀어지고 (*그림-우하단 혹은 좌하단*), 특정 기술에 oriented됨

▶ **Low-level policy와의 관계**

→ low-level policy의 인풋으로 들어가기 전 normalize 해준다: $\mathbf{z} = \bar{\mathbf{z}}/||\bar{\mathbf{z}}||$

◎ Motion Prior

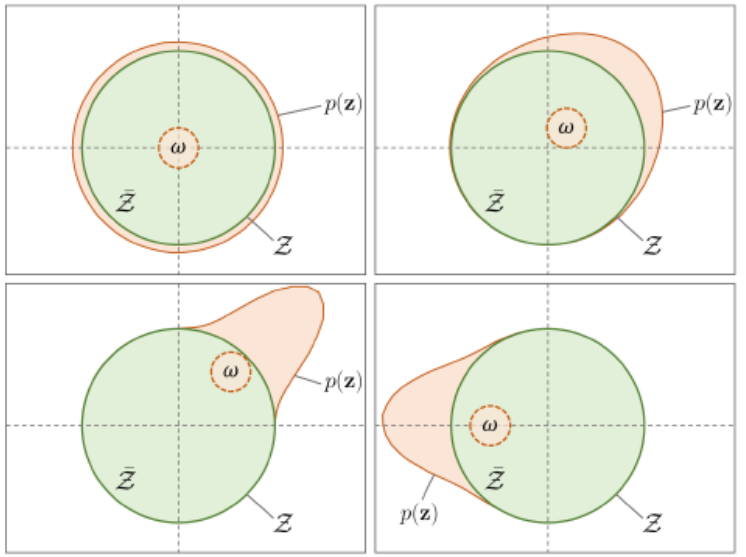
▶ **Low-level discriminator 재사용**

→ Low-level policy가 어떤 z가 들어오더라도 자연스러운 동작을 생성하도록 학습되었으나,
(데이터셋에서 보지 못한) 새로운 동작을 학습할 경우, 자연스럽지 못한 동작이 도출될 수 있음

→ 특히나, timestep이 바뀜에 따라 ω 값이 급격하게 바뀔 경우 이 현상이 두드러짐

→ 이를 방지하기 위해 pre-training이후 Discrimiator의 파라미터를 freeze하고 task-training에 사용

→ 그러므로, 새로운 task를 학습할 때, 모션 데이터가 필요 없음



Reinforcement Learning Settings

◎ States and actions

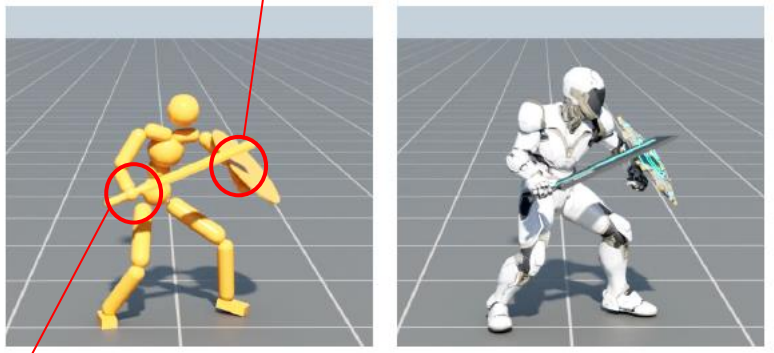
▶ **States (from 37 DoF Humanoid Char. like AMP)**

- Height of the root from the ground.
- Rotation of the root in the character's local coordinate frame.
- Linear and angular velocity of the root in the character's local coordinate frame. (AMP)
- Local rotation of each joint. (AMP)
- Local velocity of each joint. (AMP)
- Positions of the hands, feet, shield, and tip of the sword in the character's local coordinate frame. (AMP)

ASE에서 추가됨

➔ 총 120D의 state space

방패(shield)는 fixed joint를 통해 왼손에 부착



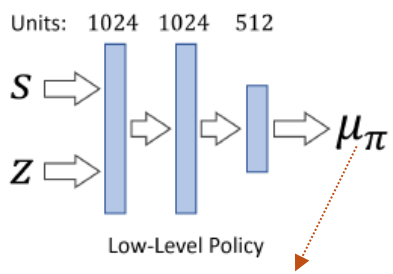
(a) Simulation Model (b) Visualization Model

칼(sword)은 3D spherical joint를 통해 오른손에 부착

▶ **Actions**

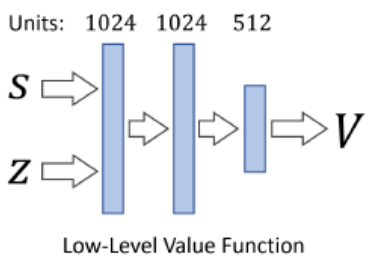
→ target rotations for PD controllers positioned at each of the character's joint (31D action space)

▶ **Neural Nets Architecture**

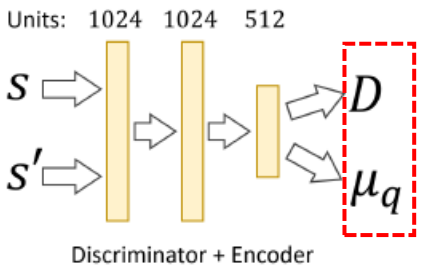


Low-Level Policy

$$\pi(\mathbf{a}|\mathbf{s},\mathbf{z}) = \mathcal{N}(\mu_\pi(\mathbf{s},\mathbf{z}), \Sigma_\pi)$$

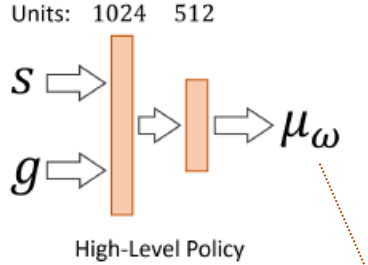


Low-Level Value Function



Discriminator + Encoder

Discriminator와 encoder mean은 조인트하게 학습이 되고 아웃풋만 다르다.



High-Level Policy

$$\omega(\bar{\mathbf{z}}|\mathbf{s},\mathbf{g}) = \mathcal{N}(\mu_\omega(\mathbf{s},\mathbf{g}), \Sigma_\omega)$$

Results

◎ Experimental Setup

- **Dataset : Reallusion의 187개 모션 클립**
- **Hardware Environment: NVIDIA V100 GPU / Software Environment: IsaacGym, Pytorch**

◎ Result of Low-level Skills

Fig. 6. Simulated character performing skills generated by random samples from the latent space. The low-level policy learns to model a diverse array of skills.

- Low-level 만으로도 다양한 자연스러운 모션 학습이 가능함
- 앞서 소개한 skill discovery의 영향력이 큼
- 여러가지 모션이 짬뽕된 데이터셋을 썼지만 **구조화 된 스킬 임베딩으로** 학습 되었음을 의미

◎ Task 수행 능력

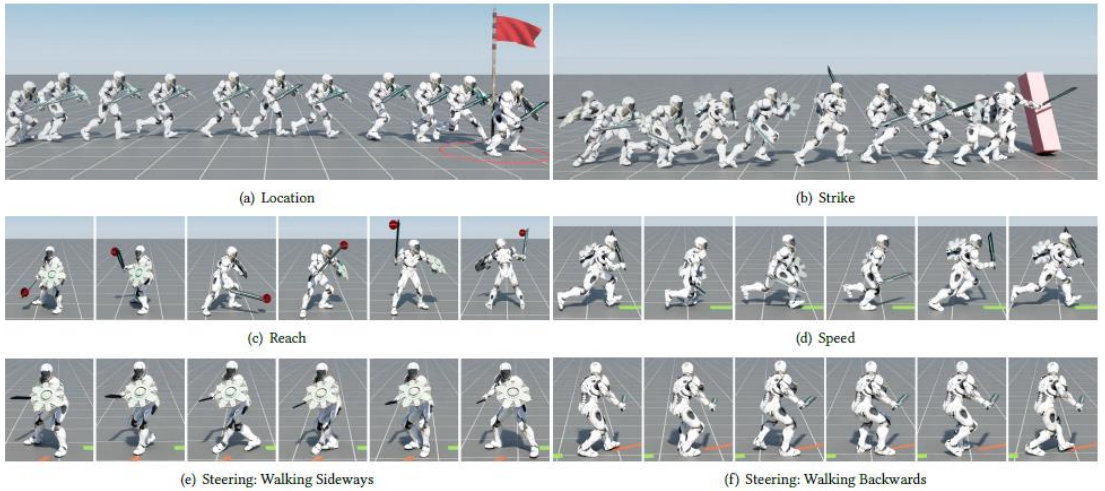
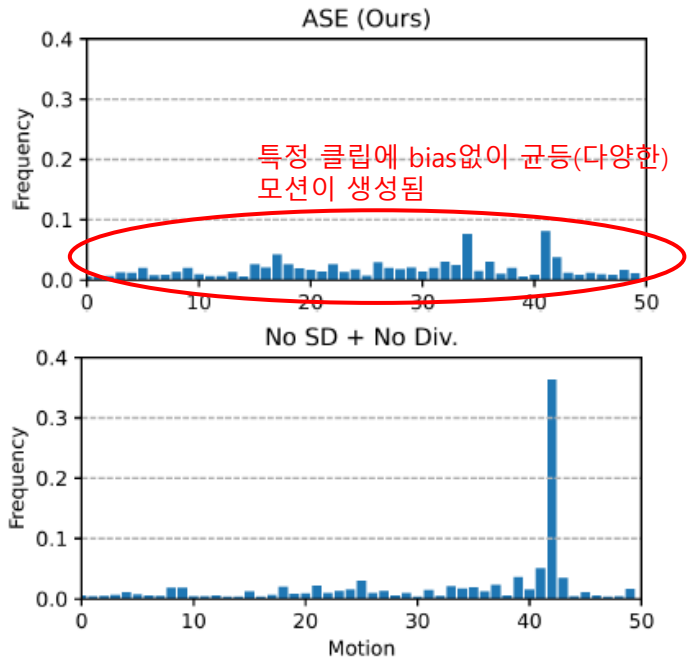


Fig. 7. Simulated character performing tasks using skills from a pre-trained low-level policy. The character can be directed to perform various tasks using simple reward functions, and the low-level policy then enables the character to achieve the task objectives by using naturalistic behaviors.

- 각 task 별 reward를 추가 해주는 것 만으로도 다양하고 자연스러운 모션의 컨트롤이 가능함

Ablation Study

◎ Task 다양성 측정



$$m^* = \arg\min_{m^i \in \mathcal{M}} \min_{(\mathbf{s},\mathbf{s}') \in m^i} ||\bar{\mathbf{s}}_t - \bar{\mathbf{s}}||^2 + ||\bar{\mathbf{s}}_{t+1} - \bar{\mathbf{s}}'||^2$$

- 랜덤한 trajector생성 (random z사용)후, 각 state transition에 대해서 가장 비슷한 transitio 을 갖고있는 모션 클립을 찾은 후 비율 비교 (Top 50)
  ➔ 앞서 설명한 Skill discovery objective 덕택

◎ Training Time

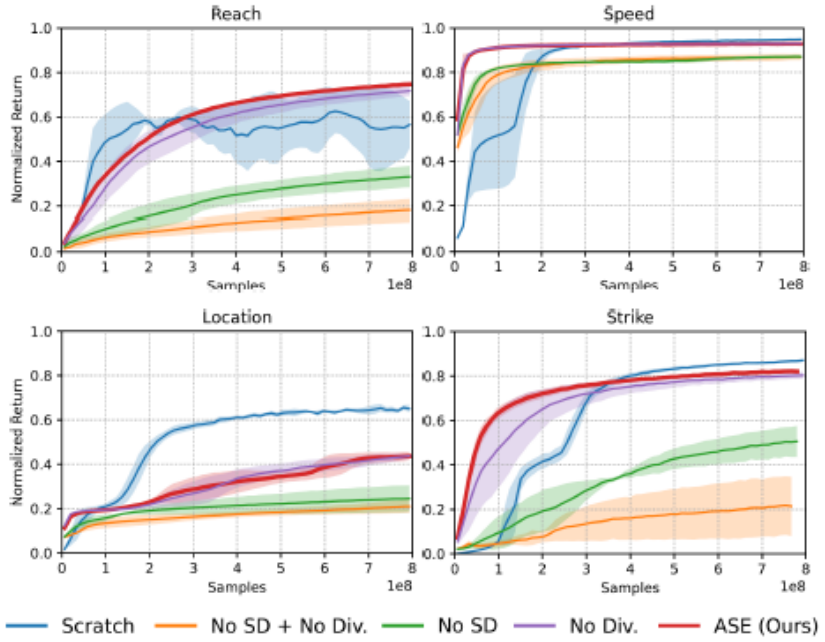


Fig. 11. Learning curves comparing performance on downstream tasks using different low-level policies. We compare ASE to policies that are trained from scratch for each tasks (Scratch), as well as to low-level policies trained without the skill discovery objective (No SD), without the diversity objective (No Div.), and with both objectives disabled (No SD + No Div.). The skill discovery objective is crucial for learning effective skill representations. The policies trained from scratch often achieve higher returns by exploiting unnatural behaviors (see Figure 8)

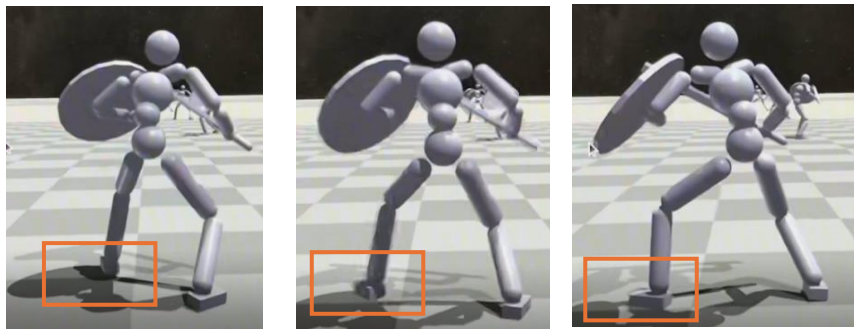
간단 실험

◎ Latent 별로 실제 모션이 얼마나 달라지는가? 아무값이나 사용해 볼 경우..

```
#################### (Myungin Test starts) [24/11/4] ########################

    def create_fixed_latent(self, latent_dim, device):
        """Helper function to create a test latent vector"""
        # You can modify these values for different behaviors
        z = torch.zeros(latent_dim, dtype=torch.float32, device=device)
        z[0] = 1.0
        return z
```

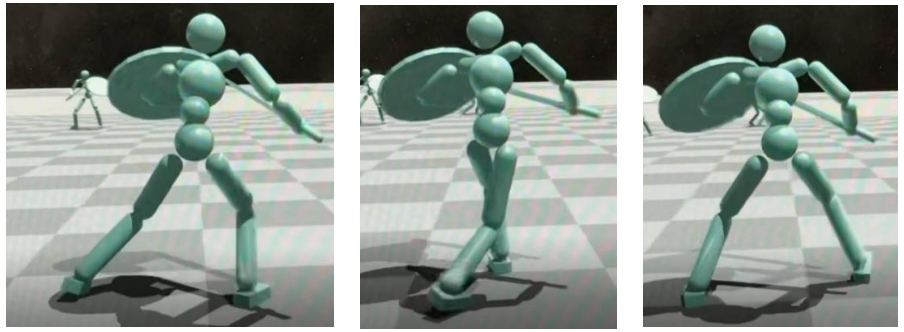
Z = [1,0,0,0,0,0, ...0] 사용



"왼발을 바닥에 고정하고 사주경계 "

```
def create_fixed_latent(self, latent_dim, device):
    """Helper function to create a test latent vector"""
    # You can modify these values for different behaviors
    z = torch.zeros(latent_dim, dtype=torch.float32, device=device)
    z[3] = 1.0
    return z
```

Z = [0,0,0,1,0,0, ...0] 사용



"전방 주시하면서 옆으로 걷기 "