The Journal of the Convergence on Culture Technology (JCCT)
Vol. 10, No. 6, pp.597-604, November 30, 2024. pISSN 2384-0358, eISSN 2384-0366

http://dx.doi.org/10.17703/JCCT.2024.10.6.597
JCCT 2024-11-72

역기구학과 강화 학습을 활용한 리드 클라이밍 루트 파인딩 시뮬레이션 개발

Development of a Lead Climbing Route-Finding Simulation Using Inverse Kinematics and Reinforcement Learning

노승현*, 진성아**

Seunghyun Noh*, Seongah Chin**

요 약 본 연구는 올림픽 정식 종목 중 하나인 리드 클라이밍에 초점을 맞춰서 루트 파인딩 시뮬레이션을 개발하는 것을 목표로 한다. Unity를 활용하여 장면에 오브젝트를 배치하고 변수들을 정의하여 클라이밍 환경을 구성하고, 클라이밍 자세에 따른 상태를 생성하여 클라이머 모델의 관절 위치를 FABRIK 알고리즘으로 계산하였다. 그리고 ML-Agents를 활용하여 벡터 관측 선택과 이산적인 행동을 정의하고, CCW 알고리즘을 이용한 안정적인 자세를 정의하여 행동 마스킹과 에이전트의 행동 선택에 따른 처리 함수를 구현하였다. 모의실험 결과, 탑 홀드까지 루트 파인딩이 가능함을 확인하였고, 이러한 시뮬레이션이 리드 클라이밍 훈련을 진행하는 선수들에게 많은 도움을 줄 것으로 기대된다.

주요어 : 리드 클라이밍, 루트 파인딩, 역기구학, 강화 학습

Abstract This study aims to develop a route-finding simulation focused on lead climbing, an official Olympic discipline. Objects were strategically using Unity placed and variables defined to construct a realistic climbing environment. Various climbing postures were generated, and the joint positions of the climber model were calculated using the FABRIK algorithm. ML-Agents were utilized to define vector observations and discrete actions, with stable postures determined using the CCW algorithm. Functions for behavior masking and agent action selection were implemented. Simulation results confirmed the feasibility of route-finding to the top hold, showing promise in enhancing training for lead climbers.

Key words : Lead Climbing, Route-Finding, Inverse Kinematics, Reinforcement Learning

Ⅰ. 서 론

새로운 정책을 설계하거나 제안한 시스템을 도입하기에 앞서 시뮬레이션 연구가 수행되어 왔다[1-2]. 2020 도쿄 올림픽에서는 스포츠 클라이밍이 정식 종목으로 채택되었으며, 스피드, 볼더링, 리드 세 가지 종목의 순위를 종합하는 컴바인 방식으로 진행되었다[3].

이에 따라 클라이밍에 대한 사람들의 관심이 꾸준히 증가하고 있으며, 클라이밍 선수들이 올림픽에서 높은 성과를 달성하기 위해 많은 시간을 클라이밍 연습에 투

*정회원, 성결대학교 미디어소프트웨어학과 XICOM LAB. 연구보조원
**성결대학교 미디어소프트웨어학과 XICOM LAB. 정교수(교신저자)
접수일: 2024년 8월 12일, 수정완료일: 2024년 9월 7일
게재확정일: 2024년 11월 1일

Received: August 12, 2024 / Revised: September 7, 2024
Accepted: November 1, 2024
*Corresponding Author: solideochin@gmail.com
Dept. of Business Administration , Univ. of Seoul, Korea

자하고 있다.

클라이밍 연습에서 중요한 부분 중 하나는 클라이밍 루트를 관찰하고 분석하여 시작 지점에서 목표 홀드까지 빠르고 안정적으로 도달할 수 있는 루트를 찾고, 그에 맞는 클라이밍 자세를 취하며 올라가는 것이다. 이러한 연습에 도움이 되는 다양한 클라이밍 루트 파인딩 시뮬레이션들이 개발되었는데, Back Tracking 방식으로 후보 지점들을 저장하여 루트를 찾는 시뮬레이션[4], 자세 그래프에서 A* Prune 알고리즘으로 볼더링 루트를 찾는 시뮬레이션[5] 등이 있다. 또한 Unity의 ML-Agents를 활용한 강화 학습으로 루트 파인딩을 진행하는 시뮬레이션도 등장하였다[6].

본 논문에서는 올림픽 정식 종목 중 하나인 리드 클라이밍에 초점을 맞춰 클라이밍 시뮬레이션을 개발하고자 한다. 리드 클라이밍은 15m 이상의 클라이밍 벽에서 진행되며, 선수는 지면에서 발이 떨어지는 순간 경기를 시작한다. 순위는 주어진 시간 내에 누가 높이 오르는가로 결정되고, 탑 홀드에 도달하면 종료된다. 이러한 요소들을 가진 클라이밍 환경에서 효과적으로 연습하는 데 도움이 되는 루트 파인딩 시뮬레이션은 아직 부족하며, 이를 이용한 연구 및 개발이 필요한 상황이다.

본 논문의 구성은 다음과 같다. 먼저 역기구학과 CCW 알고리즘, 강화 학습 등에 대하여 설명하며 관련 연구와 기술을 소개한다. 다음으로 클라이밍 환경을 구성하기 위해 오브젝트를 배치하고, 필요 변수들과 계산 방법을 정의한다. 이어서 강화 학습 환경을 구성하고, 마지막으로 테스트 진행과 본 연구의 의의를 정리한다.

II. 본 문

1. 관련 연구 및 기술

1) 역기구학과 FABRIK 알고리즘

강체 다물체 시스템(Rigid Multi-body System)은 링크(Link)라고 불리는 강체들이 연결된 시스템을 의미한다. 이 링크들은 조인트(Joint)로 서로 연결되며, 강체 다물체의 움직임을 제어하기 위해 역기구학(Inverse Kinematics)을 사용한다. 역기구학 문제는 엔드 이펙터(End Effector)라 불리는 링크의 끝 지점이 목표 위치(Target Position)에 도달하도록 각 조인트의 각도를 조절하는 문제이다[7].

FABRIK은 역기구학 문제를 휴리스틱(Heuristic) 방법으로 해결하는 알고리즘으로, 최근에 계산된 조인트들의 위치를 이용하여 Forward And Backward Reaching 반복 과정을 거쳐서 각 조인트의 위치를 갱신한다[8]. 처음에는 루트 조인트와 목표 지점 사이의 거리와 링크들의 길이 합을 비교하여 엔드 이펙터가 목표 지점에 도달 가능한지 판별한다. 만일 도달 가능하다면, 각 반복 과정에서 엔드 이펙터를 목표 위치로 놓고 각 조인트의 위치를 갱신하는 Forward Reaching 단계, 루트 조인트를 초기 위치로 놓고 다시 각 조인트의 위치를 갱신하는 Backward Reaching 단계를 거친다.

2) CCW 알고리즘

CCW(Counter-Clockwise) 알고리즘은 평면상에서 두 벡터의 방향성을 판별하는 알고리즘이다[9]. 방향 선분 $\overrightarrow{ab}$를 기준으로 방향 선분 $\overrightarrow{ac}$가 반시계 방향에 있는지를 결정하기 위해 벡터 곱 $\overrightarrow{ab} \times \overrightarrow{ac}$를 계산한다. 여기서 벡터의 z 성분이 양수이면 반시계 방향, 음수이면 시계 방향, 0이면 일직선상으로 볼 수 있다.

3) 강화 학습과 Unity ML-Agents

강화 학습(Reinforcement Learning)은 순차적인 행동 선택을 통해 보상을 최대화하는 의사 결정 전략인 순차적 결정 문제를 해결하는 방법이다. 강화 학습 환경에서 에이전트는 환경 정보를 관측하고, 특정 행동을 수행하여 보상을 획득한다. 이 과정에서 에이전트는 시행착오를 거치며 학습하고, 장기적으로 보상을 최대화하는 최적의 행동 정책을 학습하게 된다.

ML-Agents는 Unity에서 강화 학습을 진행할 수 있는 오픈 소스 툴킷이다[10]. PyTorch와 Unity 사이의 통신을 담당하며, 에이전트의 행동을 환경으로 전달하고, 변화한 상태와 보상을 다시 에이전트에게 전달해 주는 역할을 한다.

2. 클라이밍 환경 구성

1) Unity 장면 구성과 오브젝트 배치

강화 학습을 진행하기에 앞서 클라이밍 환경을 구성하기 위해 Unity 장면에 배치할 오브젝트들을 [표 1]과 같이 선정하여 배치하였다. 클라이밍 벽에는 홀드들을 배치하여 클라이밍장을 제작하였고, 학습을 진행할 에이전트인 클라이머는 시작 홀드들을 선정하여 클라이

밍 루트 파인딩의 초기 지점을 설정하였다.

표 1. Unity 장면에 배치한 오브젝트 목록
Table 1. The list of objects placed in a Unity scene

이미지	오브젝트명	설명
	Wall	클라이밍 벽
	Hold	홀드
	Climber	클라이머

2) 변수 정의

클라이밍 자세 S는 클라이머가 어느 홀드를 잡거나 밟고 있는지를 나타내며, Iterable 구조체로 정의하였다. S의 변수들은 [표 2]와 같이 순서대로 정의하였다.

표 2. 클라이밍 자세 S의 변수 목록
Table 2. Variables of S for climbing postures

순서	변수	설명
1	h_{lh}	왼손으로 잡고 있는 홀드 번호
2	h_{rh}	오른손으로 잡고 있는 홀드 번호
3	h_{lf}	왼발로 밟고 있는 홀드 번호
4	h_{rf}	오른발로 밟고 있는 홀드 번호

클라이밍 상태 T는 클라이밍 자세 S를 취할 때 클라이머의 각 관절이 어느 위치 또는 방향에 있어야 하는지를 나타내며, 구조체로 정의하였다. T의 변수들은 [표 3]과 같이 정의하였고, 클라이머 모델에 각 변수를 표시하면 [그림 1]과 같다.

그 외 클라이밍 무브 또는 에이전트 학습에 사용되는 변수들은 [표 4]와 같이 정의하였다.

표 3. 클라이밍 상태 T의 변수 목록
Table 3. Variables of T for climbing states

변수	설명
$t_{lh}, t_{rh}, t_{lf}, t_{rf}, t_b, t_e$	'왼손, 오른손, 왼발, 오른발, 몸 무게 중심, 시점'의 목표 지점
$p_{lh}, p_{rh}, p_{lf}, p_{rf}$	'왼손, 오른손, 왼발, 오른발'의 Pole 위치
$f_{lh}, f_{rh}, f_{lf}, f_{rf}, f_b$	'왼손, 오른손, 왼발, 오른발, 몸 무게 중심'의 전향 벡터
u_b, u_e	'몸 무게 중심, 시점'의 상향 벡터
a_l, a_r	'왼쪽 팔 윗부분, 오른쪽 팔 윗부분'의 위치
l_l, l_r	'왼쪽 다리 윗부분, 오른쪽 다리 윗부분'의 위치

3. 클라이밍 무브 구현

1) 클라이밍 상태 생성

클라이밍 자세 S를 취할 때의 클라이밍 상태 T를 생성하기 위해 [표 5]와 같이 어떤 순서로 T의 변수들을 계산해야 하는지를 정의하였다. 여기서 a는 클라이머와 오브젝트 사이가 겹치지 않도록 거리를 보정하는 상수이고, $v.n$은 벡터 v의 정규화된 벡터를 의미한다.

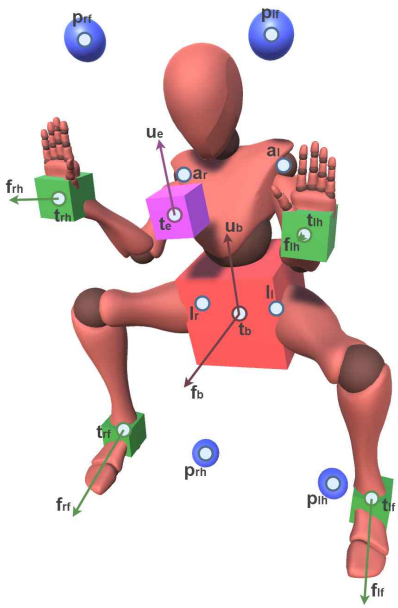


그림 1. 클라이머 모델에 표시한 클라이밍 상태 T의 변수
Figure 1. Variables representing climbing states of T displayed on the climber model

표 4. 그 외 변수 목록
Table 4. Other variables

변수	설명
N	홀드의 수
h_{top}	탑 홀드 번호
s_{top}	탑 홀드의 위치
$s_{lh}, s_{rh}, s_{lf}, s_{rf}$	S의 '왼손, 오른손, 왼발, 오른발' 홀드의 위치
$v_{lh}, v_{rh}, v_{lf}, v_{rf}$	S의 '왼손, 오른손, 왼발, 오른발' 홀드의 전향 벡터
R	남은 보상
D	남은 거리
A	팔 윗부분과 아랫부분의 길이 합
L	다리 윗부분과 아랫부분의 길이 합

2) 클라이머 모델의 팔다리 위치 계산

클라이밍 상태 T가 주어졌을 때, FABRIK 알고리즘을 사용하여 클라이머 모델의 팔다리 위치를 계산하였다. 팔의 위치 계산을 중점으로 살펴보면, [그림 2]와 같이 팔 윗부분은 루트 조인트, 손은 엔드 이펙터인 강체 다물체 시스템을 구성하였고, Pole의 방향으로 회전 제약을 유지하면서 Forward And Backward Reaching 반복 과정으로 손이 목표 지점으로 이동했을 때의 각 조인트의 위치를 갱신하였다.

표 5. 클라이밍 상태 T의 변수 계산 방법 정의
Table 5. Defining the method of T for calculating variables of climbing states

순서	해당 변수	계산 방법
1	$t_{lh}, t_{rh}, t_{lf}, t_{rf}$	$t_i = s_i - f_i a$ (for $i = lh, rh, lf, rf$)
2	$f_{lh}, f_{rh}, f_{lf}, f_{rf}$	$f_i = s_i - v_i a$ (for $i = lh, rh, lf, rf$)
3	f_b	$f_b = ((t_{rf} - t_{lf}) \times (t_{rh} - t_{lf})).n$
4	u_b	$u_b = ((t_{lh} + t_{rh})/2 - (t_{lf} + t_{rf})/2).n$
5	t_b	$t_b = (t_{lh} + t_{rh} + t_{lf} + t_{rf})/4 - f_b a$
6	u_e	$u_e = u_b$
7	t_e	$t_e = (t_{lh} + t_{rh})/2$
8	p_{lh}, p_{rh}	$p_{ih} = (a_i + t_{ih})/2 - u_b$ (for $i = l, r$)
9	p_{lf}, p_{rf}	$p_{if} = (l_i + t_{if})/2 + u_b$ (for $i = l, r$)

4. 강화 학습 구성

1) 벡터 관측 선택

벡터 관측값은 현재 위치해 있는 홀드의 위치, 탑 홀드의 위치를 기준으로 루트 파인딩을 진행하므로 $s_{lh}, s_{rh}, s_{lf}, s_{rf}, s_{top}$ 총 5개의 변수를 선택하였다.

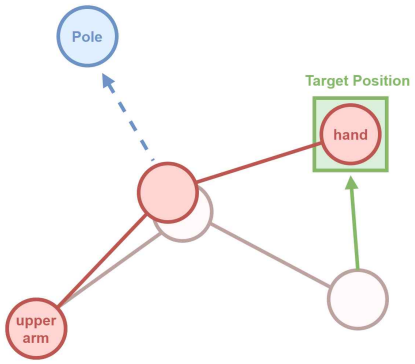


그림 2. FABRIK 알고리즘을 이용한 팔의 위치 계산
Figure 2. Calculating arm positions using the FABRIK algorithm

2) 이산적인 행동 정의

홀드의 수 N에 대해, Branch 0의 이산적인 행동 범위를 [0, 4N]으로 설정하고, [표 6]과 같이 다시 5개의 범위로 나누어 각 행동 범위에 대한 의미를 정의하였다. 이렇게 하면 에이전트는 하나의 행동으로 하나의 홀드만 선택하거나, 아무것도 선택하지 않게 된다.

표 6. 이산적인 행동 정의
Table 6. Discrete behavior definition

이산적인 행동 번호	설명
[0, N-1]	왼손으로 [0, N-1]번 홀드를 잡는다.
[N, 2N-1]	오른손으로 [0, N-1]번 홀드를 잡는다.
[2N, 3N-1]	왼발로 [0, N-1]번 홀드를 밟는다.
[3N, 4N-1]	오른발로 [0, N-1]번 홀드를 밟는다.
4N	아무런 행동도 하지 않는다.

5. 안정적인 클라이밍 자세 정의

에이전트가 다음 행동을 잘못 선택하면, 몸이 뒤틀린 자세, 물구나무서기 자세 등 잘못된 클라이밍 자세가 나올 수 있다. 이를 방지하기 위해 다음과 같이 안정적인 클라이밍 자세를 정의하였다.

(1) 클라이밍 상태 T에서 손발이 각각 목표 지점에 도달 가능해야 한다. 이는 FABRIK 알고리즘에서 언급한 도달 가능성 판별을 통해 확인할 수 있다.

(2) 손의 위치가 발의 위치보다 위에 있어야 한다.

(3) 손 홀드 번호와 발 홀드 번호가 서로 일치하면 안 된다.

(4) 삼지점 또는 사지점 자세여야 한다.

(5) 만일 삼지점 자세이면, [표 7]의 조건 중 하나를

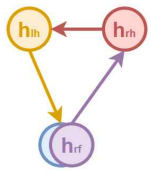
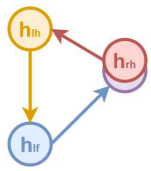
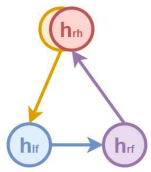
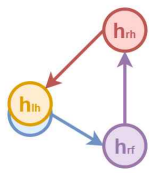
만족해야 한다.

(6) 만일 사지점 자세이면, [표 8]의 조건 중 하나를 만족해야 한다.

여기서 클라이밍 벽의 각 표면 기울기는 x축 또는 y축 방향으로 구간 (-90, 90) 사이의 각도만큼만 회전된 상태라 가정하였고, 네 홀드의 위치 $h_{lh}, h_{lf}, h_{rf}, h_{rh}$를 $z = 0$인 평면상에 평행으로 투상하여 CCW 알고리즘을 수행할 수 있도록 구현하였다.

반시계 방향인 $h_{lh}, h_{lf}, h_{rf}, h_{rh}$ 순서로 순회하면서 CCW 알고리즘을 수행한다고 가정했을 때, [표 8]에서 α는 $CCW > 0$의 개수, β는 $CCW < 0$의 개수, γ는 $CCW = 0$의 개수를 의미한다.

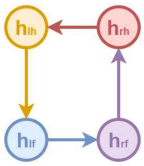
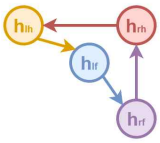
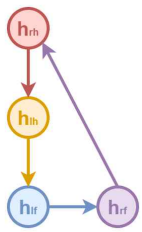
표 7. 올바른 삼지점 자세 선택
Table 7. Selecting the correct three-point stance

예시 이미지	조건
	$h_{lf} = h_{rf}$ $((s_{rh} - s_{lf}) \times (s_{lh} - s_{lf})).z > 0$
	$h_{rf} = h_{rh}$ $((s_{lh} - s_{rf}) \times (s_{lf} - s_{rf})).z > 0$
	$h_{rh} = h_{lh}$ $((s_{lf} - s_{rh}) \times (s_{rf} - s_{rh})).z > 0$
	$h_{lh} = h_{lf}$ $((s_{rf} - s_{lh}) \times (s_{rh} - s_{lh})).z > 0$

6. OnEpisodeBegin 함수 구현

OnEpisodeBegin 함수는 에피소드가 시작될 때마다 호출되는 함수로, [알고리즘 1]과 같이 구현하였다. 먼저 사용자가 지정한 시작 자세로 초기화하고, 자세에 따른 클라이밍 상태, 남은 보상, 남은 거리 순으로 초기화하였다.

표 8. 올바른 사지점 자세 선택
Table 8. Selecting the correct four-point stance

예시 이미지	조건
	$\alpha = 4$ $\beta = 0$ $\gamma = 0$
	$\alpha = 3$ $\beta = 1$ $\gamma = 0$
	$\alpha = 3$ $\beta = 0$ $\gamma = 1$

알고리즘 1. 에피소드가 시작될 때 초기화하는 알고리즘
Algorithm 1. Algorithm for initializing episodes at the start

```
1  Initialize S
2  T = Generate state when assuming S
3  R = 1
4  d_l = distance from S[h_lh] to h_top
5  d_r = distance from S[h_rh] to h_top
6  D = min(d_l, d_r)
```

7. WriteDiscreteActionMask 함수 구현

WriteDiscreteActionMask 함수는 에이전트가 이산적인 행동 중 일부분을 선택하지 못하도록 마스킹하는 함수로, [알고리즘 2]와 같이 구현하였다. 에이전트가 현재 자세에서 임의의 홀드 하나를 선택하여 다음 자세를 취한다고 가정했을 때, 그 자세가 안정적인 자세가 아니면 그에 해당하는 행동을 마스킹하였다.

8. OnActionReceived 함수 구현

OnActionReceived 함수는 에이전트가 선택한 이산적인 행동을 처리하는 함수로, [알고리즘 3]과 같이 구현하였다.

알고리즘 2. 이산적인 행동을 마스킹하는 알고리즘
Algorithm 2. Algorithm for masking discrete actions

```
for i = 0, 1, ..., N−1 do
  S′ = S
  j = 0
  for each reference variable h in S′ do
    h_tmp = h
    h = i
    if S′ is not stable state then
      Disable action with index i + jN in branch 0
    h = h_tmp
    j = j + 1
```

알고리즘 3. 에이전트의 이산적인 행동을 처리하는 알고리즘
Algorithm 3. Algorithm for processing the agent's discrete actions

```
Add reward of −0.01
a = discrete action selected by agent in branch 0
// 현재 자세에서 다음 자세로 취하고, 상태를 갱신한다.
S′ = S
j = 0
for each reference variable h in S′ do
  if jN ≤ a ≤ (j+1)N−1 then
    h = a − jN
T = Generate state when assuming S′
S = S′
// 만일 다음 자세가 현재 자세보다 탑 홀드에 더 근접하면,
   보상을 계산하여 추가하고 남은 보상과 거리를 갱신한다.
d_l = distance from S′[h_lh] to h_top
d_r = distance from S′[h_rh] to h_top
d = min(d_l, d_r)
if D > d then
  r = R(D − d)/D
  Add reward of r
  R = R − r
  D = d
// 만일 탑 홀드에 도달하면, 에피소드를 종료한다.
if S′[h_lh] = h_top or S′[h_rh] = h_top then
  End episode
```

9. 하이퍼파라미터 설정

mlagents-learn을 통한 학습을 수행하기 위해 yaml 파일의 하이퍼파라미터 일부를 조정하여 사용하였다([표 9]). 여기서 batch_size는 에이전트가 이산적인 행동을 선택하므로 32로 조정하였고, buffer_size는 적은 메모리로 학습할 수 있도록 2048로 조정하였다.

표 9. yaml 파일의 하이퍼파라미터 설정
Table 9. Hyperparameter configuration in a YAML file

파라미터		설정
trainer_type		ppo
hyperparameters	batch_size	32
	buffer_size	2048
	learning_rate	0.0003
	beta	0.005
	epsilon	0.2
	lambd	0.95
	num_epoch	3
	learning_rate_schedule	linear

III. 실험 및 결과

리드 클라이밍 시뮬레이션이 제대로 동작하는지 확인하기 위해 [표 10]과 같이 총 4개의 강화 학습 환경을 구성하고, 테스트를 진행하였다.

표 10. 강화 학습 환경 구성
Table 10. Setting up a reinforcement learning environment

	s1	s2	s3	s4
홀드의 수	43	43	86	31
이산적인 행동의 수	173	173	345	125
벽의 상태	수직	수직	수직, 가로 2배	불규칙한 표면
에이전트 모델명	X Bot	Remy	X Bot	X Bot
Max Step	1,500			
Steps of Final Checkpoint	300,000			

mlagents-learn을 통한 강화 학습을 진행한 결과, [표 11]에서 네 환경 모두 시작 지점에서 탑 홀드까지의 루트 파인딩이 가능함을 직접 확인할 수 있었다. 또한 ML-Agents의 TensorBoard를 분석해 본 결과, 학습이 진행됨에 따라 [그림 3]에서 네 환경의 Cumulative Reward 그래프가 상승하는 것을 확인할 수 있었고, 반대로 [그림 4]에서 네 환경의 Episode Length 그래프가 줄어드는 모습을 관찰할 수 있었다. 이를 통해 에이전트가 탑 홀드까지 빠르고 정확하게 올라가는 방법을 효

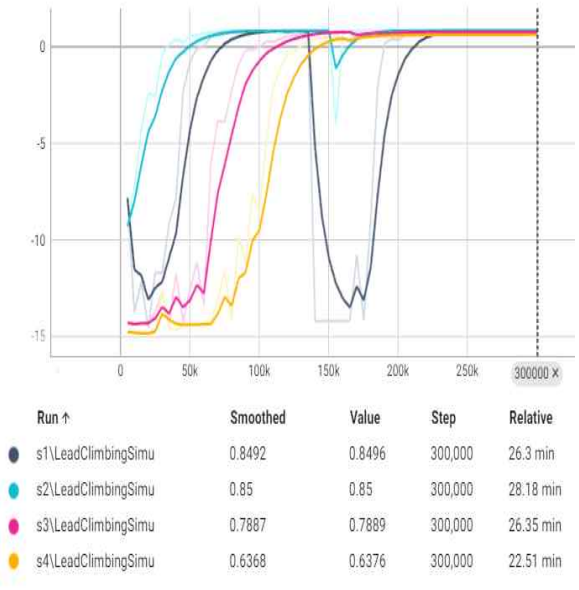
과적으로 학습했음을 확인할 수 있었다.

Ⅳ. 결 론

본 논문에서는 리드 클라이밍에 초점을 맞춘 역기구학과 강화 학습 기반의 루트 파인딩 시뮬레이션을 개발하고, 테스트를 진행하였다. 이 시뮬레이션을 통해 탑 홀드까지 빠르고 안정적으로 도달하기 위한 최적의 클라이밍 루트를 확인할 수 있으며, 클라이밍 자세를 시각적으로 확인해 볼 수 있다.

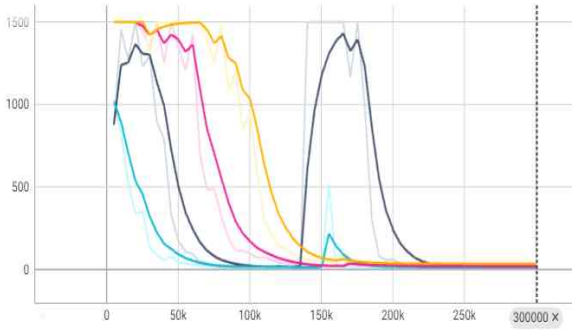
하지만 현재 시뮬레이션은 팔다리가 겹치거나 몸 뒤쪽으로 다리가 나오는 등의 문제점이 있어, 이를 해결하기 위해 다양한 방법으로 관절 계산을 시도하여 자세를 개선할 필요가 있다. 또한 리드 클라이밍의 핵심 요소 중 하나인 앵커가 구현되어 있지 않아, 향후 앵커에 로프를 걸며 진행하는 기능을 추가하여 시뮬레이션의 완성도를 높이는 것이 필요하다.

그럼에도 이와 같은 시뮬레이션 구현을 통해 리드 클라이밍을 준비하는 선수들에게 맞춤형 가이드를 제공하고, 루트 파인딩 훈련에 큰 도움이 될 것으로 기대된다.



Run ↑	Smoothed	Value	Step	Relative
s1\LeadClimbingSimu	0.8492	0.8496	300,000	26.3 min
s2\LeadClimbingSimu	0.85	0.85	300,000	28.18 min
s3\LeadClimbingSimu	0.7887	0.7889	300,000	26.35 min
s4\LeadClimbingSimu	0.6368	0.6376	300,000	22.51 min

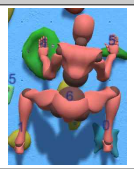
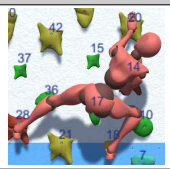
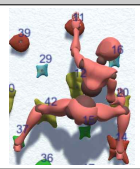
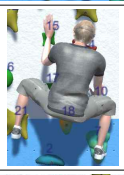
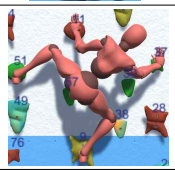
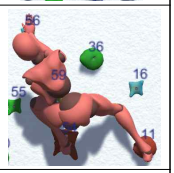
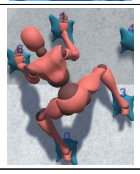
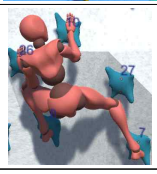
그림 3. TensorBoard의 Cumulative Reward 비교
Figure 3. Comparing Cumulative Rewards in TensorBoard



Run ↑	Smoothed	Value	Step	Relative
s1\LeadClimbingSimu	14.0786	14.036	300,000	26.3 min
s2\LeadClimbingSimu	14.0019	14	300,000	28.18 min
s3\LeadClimbingSimu	20.1274	20.1097	300,000	26.35 min
s4\LeadClimbingSimu	35.3172	35.2391	300,000	22.51 min

그림 4. TensorBoard의 Episode Length 비교
Figure 4. Comparing Episode Length in TensorBoard

표 11. 클라이밍 루트별 자세 비교
Table 11. Comparing climbing techniques for different routes

	에피소드 시작	에피소드 진행 중	에피소드 종료
s1			
s2			
s3			
s4			

References

[1] S.K. Lee. B.M KIM, and H. Heon, “Development of Control Method for Self-Driving Roller Conveyor Based on 3D Simulation Self-Driving Roller Conveyor, Manufacturing Automation, Simulation, Unity,” The Journal of the

Convergence on Culture Technology, Vol. 10, No. 3, pp. 861-864, 2024, DOI: 10.17703/JCCT.2024.10.3.861
[2] S.H. Lee, K,H. KIM and B.S. Jeong, "Design and Simulation of RFID Tag for Container-Grown Seedlings System," The International Journal of Advanced Culture Technology, Vol.10 No.2, pp. 292-299, 2022, DOI: 10.17703/IJAC T.2022.10.2.292
[3] KAFTV, "Sports Climbing Basics - Understanding Combined Competition Format(Speed, Bouldering, Lead)," YouTube, February 7, 2020. https://youtu.be/4sbSO5yzC4c
[4] H.K. Kim, and Y.K. Lee, "Route-finding simulator for Sports Climbing Education," Proceedings of the Korean Association of Computer Education Conference, Vol. 17, No. 2, pp. 273-277, 2013
[5] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Discovering and synthesizing humanoid climbing movements," ACM Transactions on Graphics (TOG), Vol. 36, No. 4, pp. 1-11, 2017. DOI:10.1145/3072959.3073707
[6] K. Naderi, A. Babadi, S. Roohi, and P. Hämäläinen, "A reinforcement learning approach to synthesizing climbing movements," 2019 IEEE Conference on Games (CoG), pp. 1-7, 2019. DOI:10.1109/CIG.2019.8848127
[7] S.R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods," Technical report, Univ. of California, 2009. https://scholar.google.co.kr/scholar?q=Introduction+to+Inverse+Kinematics+with+Jacobian+Transpose,+Pseudoinverse+and+Damped+Least+Squares+methods&hl=ko&as_sdt=0&as_vis=1&oi=scholart
[8] A. Aristidou, and J. Lasenby, "FABRIK: A fast, iterative solver for the Inverse Kinematics problem," Graphical Models, Vol. 73, No. 5, pp. 243-260, 2011. DOI:10.1016/j.gmod.2011.05.003
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to algorithms, MIT press, pp. 1014-1017, 2009. https://books.google.co.kr/books?hl=ko&lr=&id=RSMuEAAAQBAJ&oi=fnd&pg=PR13&dq=introduction+to+algorithms&ots=a3i0Y00DVN&sig=b3aSr9LEnO4kz6KYm5WelcozqUo#v=onepage&q=introduction%20to%20algorithms&f=false
[10] Unity Technologies, "Unity ML-Agents Toolkit," GitHub, October 29, 2023. https://github.com/Unity-Technologies/ml-agents/blob/643df3599f81ad472a1f7b46e8d59984ef62c39a/docs/Readme.md