Journal of Korea Robotics Society (2023) 18(2):143-154
https://doi.org/10.7746/jkros.2023.18.2.143 ISSN: 1975-6291 / eISSN: 2287-3961

심층 강화학습을 이용한 휠-다리 로봇의 3차원 장애물극복 고속 모션 계획 방법

Fast Motion Planning of Wheel-legged Robot for Crossing 3D Obstacles using Deep Reinforcement Learning

정 순 규[1] · 원 문 철†
Soonkyu Jeong[1], Mooncheol Won†

Abstract: In this study, a fast motion planning method for the swing motion of a 6x6 wheel-legged robot to traverse large obstacles and gaps is proposed. The motion planning method presented in the previous paper, which was based on trajectory optimization, took up to tens of seconds and was limited to two-dimensional, structured vertical obstacles and trenches. A deep neural network based on one-dimensional Convolutional Neural Network (CNN) is introduced to generate keyframes, which are then used to represent smooth reference commands for the six leg angles along the robot's path. The network is initially trained using the behavioral cloning method with a dataset gathered from previous simulation results of the trajectory optimization. Its performance is then improved through reinforcement learning, using a one-step REINFORCE algorithm. The trained model has increased the speed of motion planning by up to 820 times and improved the success rates of obstacle crossing under harsh conditions, such as low friction and high roughness.

Keywords: Wheel-Legged Robot, Motion Planning, Obstacle Crossing, Reinforcement Learning, Behavioral Cloning

1. 서 론

이전 논문[1]에서 경로최적화(trajectory optimization) 기법으로 모션 계획 및 제어(motion planning and control)를 수행하는 방법을 제안하였다. 경로최적화 기법은 모션을 계획하기 위하여 수 초에서 수십 초가 소요되기 때문에 실제 로봇에 적용하는 데에는 한계가 있다. 또한 볼록하지 않은 최적화 문제(non-convex optimization problem)이므로 수학적 수렴성이 보장되지 않아 2차원의 정형화된 장애물에 대한 최적화에는 적용이 가능하나 3차원 비정형 장애물에 대해서는 적용하기 어려운 문제가 있다. 본 연구에서는 이를 해결하기 위한 방법으로 심층신경망(deep neural network, DNN)을 이용하여 3차원 비정형 장애물에 대하여 고속으로 모션 계획 및 제어를 구현하는 기법을 제안한다.

최근 강화학습을 이용한 모션 계획 및 제어 방법이 많이 연구되고 있다[2-4]. 하지만 고차원 연속 상태·행동공간(high dimensional continuous state-action space)을 갖는 로봇의 경우 샘플복잡도(sample complexity)가 높아 학습에 많은 시간이 소요되며 하이퍼 파라미터(hyper-parameter) 조정에 시행착오를 거쳐야 한다. 특히 강화학습은 보상함수(reward function) 설계에 따른 성능차이가 확연하기 때문에 보상함수 최적화에도 많은 노력이 필요하다. 그럼에도 불구하고 학습과정에서 DNN 모델 파라미터 공간의 지역최적점(local optimum)에 함몰되어 최적성능에 도달하지 못하는 결과가 발생하기도 한다[5].

본 연구에서는 이러한 문제점을 해결하기 위하여 강화학습을 수행하기 이전에 경로최적화 기법으로 얻은 결과를 이용하여 모방학습(behavioral cloning)[6]을 수행하였다. 모방학습은

Received : Feb. 17. 2023; Revised : Feb. 26. 2023; Accepted : Feb. 27. 2023
1. Ph.D Candidate, Department of Mechatronics Engineering, Chungnam National University, Daejeon, Korea (reingel@o.cnu.ac.kr)
† Professor, Corresponding author: Department of Mechatronics Engineering, Chungnam National University, Daejeon, Korea (mcwon@cnu.ac.kr)

주어진 상태와 행동의 쌍(state-action pairs)을 이용하여 DNN 모델을 학습하는 일종의 지도학습(supervised learning) 방법이다. 모방학습을 이용하여 학습할 경우 학습에 사용된 환경에서는 잘 동작하지만 환경이 변경되면 성능이 낮아지는 공변량 변화(covariant shift) 문제가 있을 수 있다[7]. 따라서 모방학습 후에 강화학습으로 강건성을 높여주는 것이 바람직하다.

고차원 연속 상태·행동공간을 갖는 로봇의 모션 계획 및 제어를 위하여 강화학습으로 DNN 모델을 학습할 때 방대한 상태·행동공간을 탐색(exploration)해야 한다. 이를 위하여 DNN 모델에서 출력된 행동을 가우스 분포(Gaussian distribution) 등으로 확률분포화하여 샘플링하거나 랜덤 노이즈를 더하는 방법을 사용한다. 이로 인해 로봇에 진동이 발생할 수 있어 로봇의 거동, 특히 각속도와 가속도 등 시간미분 신호에 적지 않은 노이즈가 발생한다. 이러한 신호를 보상함수에 사용하고자 하거나 경로 계획(path planning)이나 모션 계획 등 로봇의 부드러운 움직임이 요구되는 경우에는 로봇 상태나 행동을 필터링하거나 스플라인 곡선을 사용하는 등의 추가적인 작업이 필요하다[8-10].

이와 관련하여 기 수행된 연구결과는 다음과 같다. 먼저 족형 로봇 또는 애니메이션 캐릭터의 고속 모션 계획과 제어를 위하여 강화학습을 적용한 사례가 다수 존재한다. Tsounis는 4족 로봇의 비정형 노면 보행을 위한 계획과 제어를 두 개의 신경망 모델로 구성하여 온라인으로 수행할 수 있는 방법을 제시하였다[3]. Rudin은 대규모 병렬 강화학습을 이용하여 4족 로봇이 비정형 노면에서 보행할 수 있는 에이전트(agent)를 수십 분 내에 학습시키는 결과를 보여주었다[4]. Peng은 계층적 강화학습을 이용하여 2족 캐릭터의 복잡한 움직임을 자동생성하는 기법을 제시하였다[2].

연속 행동공간을 스플라인 곡선으로 파라미터화하여 경로계획 또는 모션 계획을 수행하는 연구가 있었다. Jazayeri와 Hasanzade는 각각 차량[9] 또는 항공기[10]의 경로를 스플라인 곡선으로 표현하고 강화학습으로 최적경로을 도출하였다. Peters는 로봇의 움직임을 3차 스플라인으로 표현하고 모방학습으로 초기성능을 구현하고 강화학습으로 성능을 향상시키는 방법을 제시하였다[8].

본 연구에서는 로봇이 부드럽게 운동하는 모션을 계획하고 제어하기 위하여 DNN 모델이 키프레임(keyframe)을 출력하도록 하고 이로부터 장애물 통과 전 구간에 대한 구동 명령을 스플라인 곡선(spline curve)으로 표현하는 방법을 제안한다. 탐색을 위하여 키프레임에 노이즈를 추가하더라도 스플라인 곡선에는 노이즈가 없어 학습과정에서도 로봇의 부드러운 운동을 기대할 수 있다. 또한 스플라인 곡선을 최적화하는 강화학습을 위하여 장애물극복 에피소드(episode) 전과정을 시뮬레이션하여 획득한 시계열(time series) 데이터로 계산한 비용(cost)과 장애물극복 성공 또는 실패에 따른 보상(reward)을 조합한 보상함수를 제시한다. 즉, 장애물극복 에피소드 전체가 하나의 상태가 되며 강화학습 대상은 단일 상태 MDP (one state Markov decision process) 또는 MAB (multi-armed bandit)가 된다. 이는 경로최적화 기법[1]에서 장애물극복 에피소드 전체를 대상으로 설계변수를 구성하고 비용함수를 최소화하는 최적 설계변수를 찾는 방법과 유사하다. 또한 3차원 비정형 장애물 형상을 자동으로 생성하는 방법도 제시한다. 정리하자면 본 연구는 심층신경망 모델을 활용하여 휠-다리 로봇이 3차원 비정형 수직장애물과 참호를 통과하기 위한 최적 스윙 모션을 고속으로 계획하고 제어하는 기법을 제안한다. 기존 연구에서 제안한 경로최적화 결과를 이용하여 모방학습을 수행함으로써 초기성능을 구현하고 단일 상태 MDP로 시스템을 모델링한 후 강화학습을 수행함으로써 복잡한 형상의 3차원 수직장애물과 참호를 높은 성공률로 부드럽게 통과할 수 있는 방법을 제시한다. 본 논문에서 사용하는 기호(symbol)와 의미를 [Table 1]에 나타내었다.

[Table 1] Nomenclature

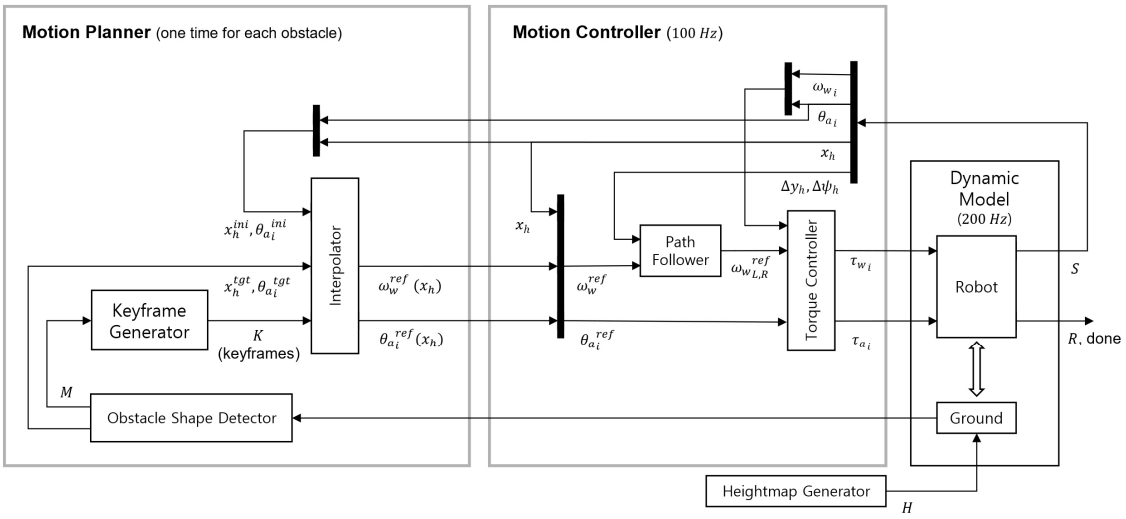
Symbol	Meaning
L_d	obstacle detection length
L_f, L_r	front and rear length of obstacle shape detection
M	obstacle shape vector
x_0	obstacle starting location
x_h, y_h, z_h	x, y, z location of center of body (hull)
ψ_h	yaw angle of body
θ_{a_i}	i-th leg (arm) angle
ω_w	angular velocity of wheel
x_h^{ini}, x_h^{tgt}	initial and target location of motion control
x_i^{up}, x_i^{dn}	lift-off and touch-down location of i-th wheel
R_w	wheel radius
N_a	the number of legs
τ_{a_i}, τ_{w_i}	torque of i-th leg and wheel
κ, K	keyframe and keyframe vector
R, c	reward and cost

2장에서는 모션 계획 및 제어 시스템 구조를 설명하고, 3장과 4장에서는 각각 모방학습과 강화학습 방법 및 결과를 기술한다.

2. 모션 계획 및 제어 시스템 구조

2.1 전체 구조

휠-다리 로봇의 고속 장애물극복 모션 계획 및 제어를 위한 시스템 전체 구조는 [Fig. 1]과 같다. 모션계획부(Motion Planner)



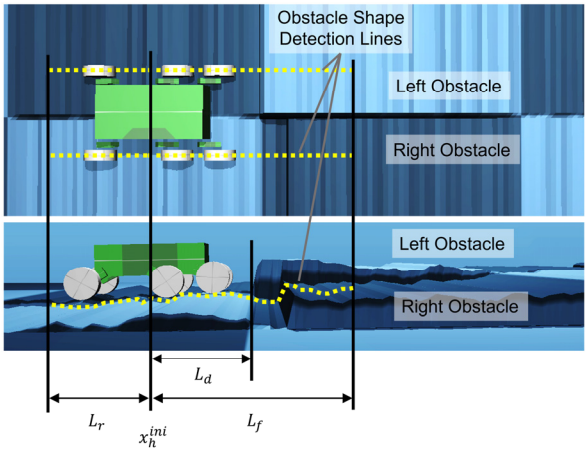
[Fig. 1] Overview of motion planning and control system for 6x6 wheel-legged robot

와 모션제어부(Motion Controller)가 좌측에 위치하며 동역학 모델(dynamic model)이 우측에 위치한다. 모션계획부는 장애물을 넘기 직전에 한 번만 실행되고 모션제어부는 장애물을 통과하는 동안 100 Hz로 실행된다. 동역학 모델은 항상 200 Hz로 실행된다. 노면생성기(Heightmap Generator)는 3차원 노면의 형상을 랜덤하게 생성하며 학습(training) 또는 평가(evaluation)시 모션계획부, 모션제어부와는 별도로 주기적으로 실행된다.

2.2 모션계획부

모션계획부는 키프레임 생성기(Keyframe Generator)와 장애물 형상 감지기(Obstacle Shape Detector), 보간기(Interpolator)로 구성된다. 키프레임 생성기는 모션계획부의 핵심으로써 장애물 형상 벡터 M으로부터 키프레임을 생성하는 DNN 모델이다. 장애물 형상 감지기는 장애물의 형상을 획득한다. 장애물 형상 감지기는 실제 로봇에 적용될 때 라이다(LiDAR) 등의 환경인식 센서로 대체되어야 할 부분이며 본 연구에서는 시뮬레이션 상에서 가상의 센서로 여러 지점에서의 노면 높이를 측정하는 방법으로 구현하였다. 환경인식 센서가 측정한 장애물 형상 데이터에는 노이즈가 포함되어 모션 계획의 성능을 저하시킬 수 있다. 본 연구에서는 이와 같이 sim2real에서 발생할 수 있는 문제를 대비하기 위하여 강화학습 시 노이즈를 추가한 노면을 반복적으로 생성하며 학습함으로써 강건성(robustness)을 높이고자 한다.

[Fig. 2]와 같이 휠-다리 로봇이 수직장애물 또는 참호가 시작되는 위치로부터 장애물감지거리 L_d 만큼 떨어진 위치(모션계획 위치 또는 모션제어 시작위치 x_h^{ini})에 도달하면 장애물



[Fig. 2] Obstacle shape detection

형상 감지기가 동체중심(center of body, CoB)으로부터 후방감지거리 L_r 만큼 떨어진 위치부터 전방감지거리 L_f 위치까지 좌우 휠 중심선을 따라 각각 N_s개(총 $2N_s$개)의 노면 높이를 샘플링하여 벡터 M을 구성한다. 후방의 노면 형상도 필요한 이유는 다리 회전각도를 계산하는 데에 있어서 현재 위치부터 목표 위치까지 이동하는 동안 모든 휠이 접촉하는 전 구간의 노면 형상이 영향을 주기 때문이다. 이러한 장애물 형상 감지기 측정범위를 실제 로봇으로 구현할 때 환경인식 센서의 시야범위가 제한되므로 주로 로봇의 전방 노면 형상만을 파악할 수 있지만 측정된 형상을 메모리에 저장하고 로봇이 전방으로 이동하는 만큼 수평이동시키면서 신규로 측정되는 전방 노면 형상을 정합하면 상기 장애물 형상 감지기의 동작을 구현할 수 있을 것이다.

장애물 형상 감지기는 측정한 장애물 형상으로부터 급격하게 상승 또는 하강하는 지점을 찾아내고 이로부터 수직장애물

상승, 하강 또는 참호인지 파악하고 모션 계획이 종료되는 위치에서의 동체중심 위치인 목표 동체중심 위치 x_h^{tgt}와 목표 다리 회전각 $\theta_{a_i}^{tgt}$을 생성한다. x_h^{tgt}는 수직장애물인 경우 장애물이 시작하는 위치 x_0에 L_d를 더한 위치이며 참호인 경우 L_t+L_d를 더한 위치이다. 여기서 L_t는 참호길이이다. 장애물 시작위치 x_0는 수직장애물 상승의 경우 노면 높이가 급격히 상승하는 위치, 수직장애물 하강이나 참호는 급격히 하강하는 위치로 정의한다. 본 연구에서 고려하는 장애물은 좌우 형상이 다른 3차원 장애물이므로 좌우 장애물이 시작하고 끝나는 위치가 서로 다를 수 있다. 따라서 모션계획 위치 또는 모션제어 시작위치 x_h^{ini}는 식 (1a)와 같이 좌우 장애물이 시작하는 위치 x_{0L}, x_{0R}중 작은 값에서 L_d를 뺀 값이며 모션제어 종료 위치 또는 목표 동체중심 위치 x_h^{tgt}는 식 (1b)와 같이 좌우 장애물이 시작하는 위치 중 큰 값에 L_d를 더한 값으로 정한다. 단, 참호인 경우 식 (1c)와 같이 좌우 참호가 끝나는 위치 중 큰 값에 L_d를 더한 값이다. 본 연구에서 사용한 모션 계획에 관련된 파라미터들을 [Table 2]에 정리하였다.

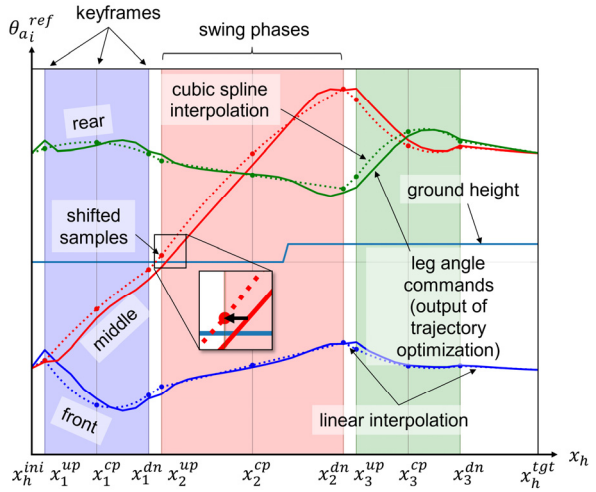
$$x_h^{ini} = \min(x_{0L}, x_{0R}) - L_d \tag{1a}$$

$$x_h^{tgt} = \max(x_{0L}, x_{0R}) + L_d \text{ for step up/down} \tag{1b}$$

$$x_h^{tgt} = \max(x_{0L} + L_{tL}, x_{0R} + L_{tR}) + L_d \text{ for trench} \tag{1c}$$

[Table 2] Parameters for motion planning

Parameter	Value	Parameter	Value
L_d	40 cm	L_f	80 cm
L_r	40 cm	N_s	150



[Fig. 3] Keyframes and optimal trajectories of leg angles

키프레임 생성기는 장애물 형상 M을 입력받아 키프레임을 출력한다. 키프레임은 모든 구간에서의 데이터(영상, 신호 등)를 재현할 수 있는 특정 지점(제어점, control point)에서의 데이터값을 의미한다. 본 연구에서는 식 (2)와 같이 휠-다리 로봇의 위치와 자세를 결정하기 위한 최소한의 조합으로 키프레임 κ를 구성하며 여기에는 동체중심의 x축 좌표와 전방, 중간, 후방 다리 회전각을 포함한다.

$$\kappa = [x_h, \theta_{a_1}, \theta_{a_2}, \theta_{a_3}]^T \tag{2}$$

[Fig. 3]은 하나의 장애물을 통과할 때 CoB의 x축 좌표 x_h에 대한 전방, 중간, 후방 다리의 회전각 명령 $\theta_{a_1}^{ref}$, $\theta_{a_2}^{ref}$, $\theta_{a_3}^{ref}$을 나타낸 그림이다. 실선으로 나타낸 세 개의 곡선은 경로최적화 기법[1]으로 생성한 2차원 로봇의 다리 회전각 명령이다. 전방, 중간, 후방 세 개의 다리가 스윙하는 구간(swing phases)이 존재하며 각 스윙 구간은 휠 분리(lift-off) 지점 x_i^{up}에서 시작하여 착지(touch-down) 지점 x_i^{dn}에서 끝난다. 각 스윙 구간에서 분리와 착지 두 지점만을 키프레임으로 사용하면 선형보간만 가능하므로 비선형 보간을 수행하기 위해 추가적인 키프레임이 필요하다. 본 연구에서는 분리와 착지 지점 사이에 하나의 키프레임 x_i^{cp}을 추가한 후 식 (3a)의 3차 자연 스플라인(natural cubic spline)으로 보간할 경우 경로최적화 기법으로 생성한 결과와 유사한 곡선을 얻을 수 있음을 확인하였다.

$$\theta_{a_i}^{ref}(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 \tag{3a}$$

$$\frac{d\theta_{a_i}^{ref}}{dx}(x) = c_1 + 2c_2 x + 3c_3 x^2 \tag{3b}$$

3차 스플라인을 사용하면 부드러운 운동을 표현할 수 있는 장점 외에도 식 (3b)와 같이 해석적 미분을 구할 수 있는 장점도 있다. 해석적 미분을 사용하면 식 (4)와 같이 다리 회전속도가 최대값을 넘지 않는 최대 휠 속도를 계산하기 용이하다.

$$\omega_w^{ref}(x_h) = \frac{1}{R_w} \min \left(\frac{\dot{\theta}_{a_{\max}}}{\max_i \frac{d\theta_{a_i}^{ref}}{dx}(x_h)}, v_{\max} \right) \tag{4}$$

앞에서 구한 3차 스플라인을 그대로 이용하여 다리 회전각을 제어하는 방법도 가능하지만 제어점을 앞으로, 즉, [Fig. 3]에서 좌측으로 약간 이동(shift)한 후 3차 스플라인을 구하면 시뮬레이션 테스트를 통하여 장애물극복 성공률이 향상되는 것을 알 수 있었다. 이는 다리 회전각 제어 시 비례제어 알고리즘으로 인하여 상시 오차가 존재하기 때문으로 판단된다. 경험적

으로 약 1.6 cm 이동할 경우 다양한 장애물 형상에 대하여 장애물극복 성공률이 전반적으로 상승하는 것을 확인할 수 있었다.

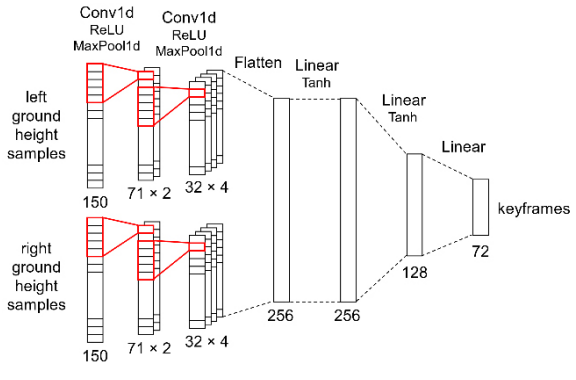
인접한 두 스윙구간 사이는 거리가 넓지 않아 선형보간으로 충분하므로 키프레임을 추가하지 않는다. 결과적으로 9개의 키프레임이 있으면 3개의 스윙 모션과 스윙간 모션을 기술할 수 있다.

모션제어 시작 위치 x_h^{ini}로부터 종료 위치 x_h^{tgt}까지 전구간에 걸쳐 제어명령을 생성해야 하므로 모션제어를 시작하고 종료하는 지점의 키프레임이 추가적으로 필요하다. 모션제어 시작위치의 키프레임은 로봇이 모션계획 지점에 도달했을 때의 로봇 상태로부터 얻을 수 있고 모션제어 종료위치의 키프레임은 목표 동체중심 위치에서의 로봇 목표상태를 사용하여 도출할 수 있다. 추가된 두 개의 키프레임은 각각 x_1^{up} 및 x_3^{dn}과 선형보간하는 데에 사용된다.

3차원 노면 형상의 장애물을 극복하기 위하여 좌우측 다리 회전각 명령을 독립적으로 생성할 수 있어야 하며 좌측과 우측의 각 휠-다리가 스윙을 시작하고 종료하는 시점이 다르기 때문에 [Fig. 3]의 키프레임이 좌우 독립적으로 존재해야 한다. 즉, 좌측과 우측에 각각 9개의 키프레임이 필요하며 총 키프레임 수는 18개, 데이터 수로는 72개가 된다. 결론적으로 키프레임 생성기는 크기가 300인 벡터를 입력받아 크기가 72인 벡터를 출력한다.

키프레임 생성기는 경로최적화 방법[1]에서 얻은 결과를 재현하는 것이 목적이다. 경로최적화는 로봇의 초기 상태와 목표상태, 그리고 장애물 형상이 주어지면 로봇이 장애물을 극복하기 위한 다리 회전각과 휠 속도 명령이 출력되는 구조이다. 키프레임 생성기에 장애물 형상을 입력하여 키프레임을 생성하고 생성된 키프레임과 로봇의 초기 상태 및 목표 상태를 조합하여 연속된 다리 회전각과 휠 속도 명령을 재현할 수 있다.

키프레임 생성기는 [Fig. 4]의 심층신경망 모델로 구현한다. 본 모델은 장애물 형상 감지기가 획득한 길이 300의 벡터를 입력받아 길이 72의 벡터를 출력한다. 300개의 벡터는 150개의 벡터 두 개로 분리되어 각각 두 단계의 1차원 CNN (convolutional neural network) 레이어에 입력되어 피처벡터(feature vector)로 변환되고 다시 하나의 벡터로 통합된다. 활성화 함수(activation function)로는 ReLU를 사용한다. 이 후 세 단계의 FC (fully connected) 레이어를 거쳐 최종적으로 크기가 72인 벡터로 변환된다. FC 레이어의 활성화 함수는 tanh를 사용한다. DNN 모델의 세부 파라미터는 [Table 3]에 정리하였다.



[Fig. 4] DNN model structure of keyframe generator

[Table 3] DNN model parameters

Layer	Parameter	Value
Conv1d	kernel	5
	stride	2
	channel	1, 2, 4
MaxPool1d	kernel	3
	stride	1

보간기는 키프레임 생성기로부터 키프레임 벡터를, 모션계획 위치에서의 로봇 상태로부터 초기 키프레임 κ^{ini}을, 장애물 형상 감지기로부터 최종 키프레임 κ^{tgt}을 받아 [Fig. 3]과 같이 3차 스플라인 보간과 선형보간을 사용하여 CoB 위치 x_h에 대한 다리 회전각 명령 함수를 출력한다. 또한 식 (3b)와 (4)를 이용하여 휠 속도 명령 함수를 유도하여 출력한다.

2.3 모션제어부

모션제어부는 모션계획부에서 생성된 다리 회전각 및 휠 속도 명령 함수를 이용하여 현재 위치 x_h에서의 다리 회전각 명령과 휠 속도 명령을 도출하는 부분과 로봇 주행경로를 추종하기 위한 경로추종기(Path Follower), 그리고 어깨 관절과 휠 회전축의 토크를 제어하기 위한 토크제어기(Torque Controller)로 구성된다.

다리 회전각 명령은 토크제어기에 그대로 입력되며, 휠 속도 명령은 경로추종기에 입력된다. 경로추종기는 로봇의 중심이 경로로부터 이탈한 수직 방향 거리 Δy_h와 경로접선과 로봇 진행방향 벡터가 이루는 요각 $\Delta\psi_h$을 이용하여 필요한 좌우 휠 속도 차이를 계산하고 이를 각 휠 속도 명령에 더하여 보완된 휠 속도 명령 $\omega_{w_{L,R}}^{ref}$을 출력한다. 경로추종기는 일종의 조향제어기로써 식 (5a)-(5b)와 같이 비례제어 알고리즘으로 구현한다. 이탈거리와 요각에 대한 비례제어 게인은 각각 1.0 (m/sec)/m, 1.0 (m/sec)/rad이다.

$$\omega_{w_L}^{ref}=\omega_w^{ref}+(P_y\Delta y_h+P_\psi\Delta\psi_h)/R_w \tag{5a}$$

$$\omega_{w_R}^{ref}=\omega_w^{ref}-(P_y\Delta y_h+P_\psi\Delta\psi_h)/R_w \tag{5b}$$

토크제어기는 식 (6a)-(6b)와 같이 다리 회전각 및 휠 속도 명령과 그에 상응하는 실제 값과의 차이를 이용한 비례제어를 수행하여 어깨와 휠 구동기의 토크 명령을 출력한다. 제어게인 P_a, P_w은 이전 논문과 동일한 값인 각각 10 Nm/deg, 0.2 Nm/(deg/sec)로 설정하였다.

$$\tau_{a_i} = P_a(\theta_{a_i} - \theta_{a_i}^{ref}) \quad (6a)$$

$$\tau_{w_i} = P_w(\omega_{w_i} - \omega_{w_i}^{ref}) \quad (6b)$$

3. 모방학습

3.1 데이터셋 생성

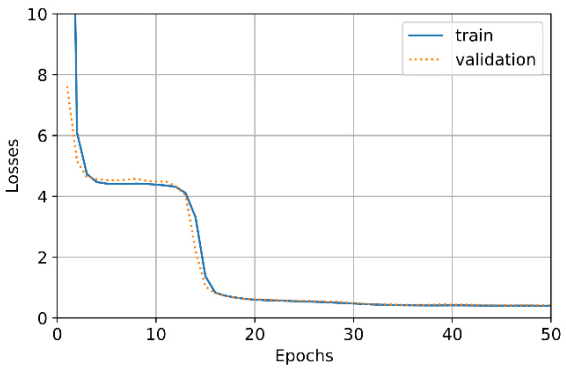
앞에서 설명한 바와 같이 DNN으로 구성한 키프레임 생성기의 초기성능을 구현하기 위하여 모방학습을 수행한다. 모방학습을 위한 데이터는 숙련자의 조작 데이터 또는 시뮬레이션 결과로부터 수집하는 것이 일반적이다. 본 연구에서는 이전 논문의 결과인 높이와 길이가 랜덤한 정형화된 수직장애물과 참호를 대상으로 경로최적화를 1000회 반복하여 획득한 데이터를 사용한다. 입력 데이터는 장애물 형상을 감지한 벡터이며 출력 데이터는 [Fig. 3]의 점으로 나타낸 바와 같이 전방, 중간, 후방 다리가 스윙을 시작하는 지점(휠 분리 지점) x_i^{up}과 종료하는 지점(착지 지점) x_i^{dn} 그리고 두 지점의 중간지점에서 추출한 동체중심 위치와 다리 회전각 명령을 통합하여 구성한 키프레임 벡터이다. 이 때 제어성능을 높이기 위하여 다리 회전각 명령을 앞으로 약 1.6 cm 이동한다. 이전 논문은 2차원 모델을 사용하므로 좌우 동일한 값으로 반복해주어야 온전한 입출력 벡터가 구성된다. 이렇게 구성된 입출력 벡터들을 저장하여 데이터셋을 생성한다.

3.2 학습방법

생성된 데이터셋을 7:3 비율로 나누어 각각 학습(training)과 검증(validation)에 사용한다. 나누어진 데이터는 셔플(shuffle)한 후 미니배치 100개씩 나누어 학습을 시킨다. 손실함수는 DNN 모델에서 출력된 벡터와 데이터셋의 출력벡터로 계산한 MSE (mean squared error)를 사용한다 학습률(learning rate)은 0.001이며 에폭(epoch) 수는 50이다.

3.3 학습결과

상기의 방법으로 모방학습을 수행한 후 손실함수 변화를



[Fig. 5] Train and validation losses of behavioral cloning

[Fig. 5]에 나타내었다. 50번 반복학습 후에 손실함수가 큰 변화없이 유지되어 학습결과가 양호함을 확인하였다.

4. 강화학습

4.1 학습 알고리즘

앞에서 설명한 바와 같이 본 연구에서는 장애물극복 에피소드 전체를 하나의 상태로 간주하는 단일 상태 MDP를 대상으로 강화학습을 수행한다. 단일 상태이므로 에피소드 종료 후 획득하는 보상이 리턴과 동일하다. 본 연구에서 사용한 강화학습 알고리즘을 [Algorithm 1]에 나타내었으며 [Table 4]에 학습에 사용한 하이퍼 파라미터를 제시하였다. 학습효율을 높이기 위하여 한 번 생성된 랜덤 노면에 대하여 세 장애물 중 하나를 랜덤하게 선택하고 장애물극복 시뮬레이션과 DNN 모델 파라미터를 업데이트하는 작업을 10번 반복한다.

키프레임 생성기에서 키프레임 벡터 K를 얻은 이후에 탐색을 위하여 평균이 K이고 동체중심 위치와 다리 회전각에 대한 표준편차가 각각 σ_{x_h}, σ_{θ_a}인 정규분포에서 샘플링을 한다. 두 표준편차 σ_{x_h}, σ_{θ_a}는 학습이 진행됨에 따라 [Fig. 6]과 같이 점차 감소시켜 탐색범위를 줄이도록 한다.

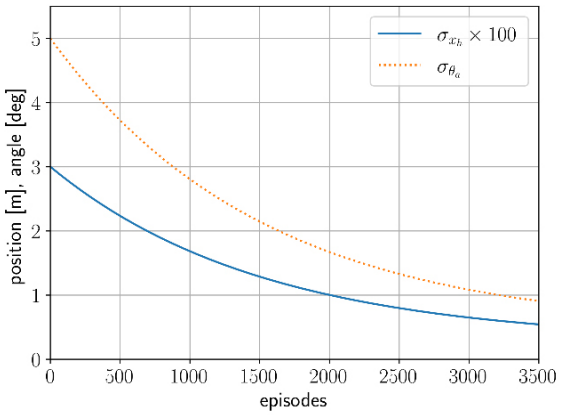
DNN 모델에서 출력하는 좌우 각 9개의 키프레임에 포함된 동체중심 위치들이 순차적으로 증가해야 하지만 학습초기에는 그렇지 않은 경우가 발생할 수 있다. 이러한 현상을 방지하기 위하여 좌우측 키프레임의 $x_h^{ini}, x_h, x_h^{tgt}$가 각각 단조증가하는지 점검하고 그렇지 못한 경우 이전 값보다 0.02 m 큰 위치로 보정한다.

[Table 4] Hyperparameters for DNN model training

Spec.	Value	Spec.	Value
N_t	350	N_u	10
α	10^{-5}		

[Algorithm 1] One Step REINFORCE

Input: Pre-trained DNN model parameters Θ
for $i = 1, 2, \ldots, N_t$ **do**
  Generate a random heightmap
  Set a random friction coefficient
  for $j = 1, 2, \ldots, N_u$ **do**
    $t \leftarrow 0.0$
    Choose randomly one out of three obstacles
    Reset environment
    Detect obstacle shape M via Obstacle Shape Detector
    Obtain keyframe vector K via Keyframe Generator
    $K' \sim \mathcal{N}(K, [\sigma_{x_h}^2, \sigma_{\theta_a}^2])$
    Make x_h^{ini}, x_h, x_h^{tgt} monotonously increase
    Convert K into cubic splines via Interpolator
    loop forever
      Obtain torque commands via Path Follower and Torque Controller
      Simulate one time step
      Obtain the next state, reward, and done
      Calculate and store an instantaneous cost $c(t)$
      if $x_h > x_h^{tgt}$ **then**
        $R \leftarrow +10 - \Sigma_t\, c(t)$
        break loop
      else if done is true **then**
        $R \leftarrow -1 - \Sigma_t\, c(t)$
        break loop
      end if
      $t \leftarrow t + \Delta t$
    end loop
    $\Theta \leftarrow \Theta - \alpha R \nabla \ln \pi\,(K|M, \Theta)$
  end for
end for
return Θ



[Fig. 6] Standard deviation schedule for robot position and leg angles

4.2 보상함수

보상함수는 식 (7)과 같이 장애물극복 성공에 대한 보상 r_s와 로봇 거동에 대한 비용함수로 구성된다. 성공보상은 로봇이 정해진 시간 내에 장애물을 극복하고 목표위치에 도달하면 +10, 그렇지 못하면 -1이다. 시간을 초과하거나 에피소드 중단조건이 참인 경우 장애물극복이 실패로 판정된다. 비용함수는 장애물을 통과하는 시간을 늘리지 않으면서 로봇의 거동을 부드럽게 만들기 위한 항목이며 동체 가속도, 각도, 각속도를 줄이기 위한 항, 다리 회전속도와 좌우 다리 회전각 차이를 줄이기 위한 항, 그리고 장애물 통과시간을 줄이기 위한 항으로 구성한다. 이는 경로최적화의 비용함수를 참조하여 설계하였다. 장애물 통과시 피치각의 증가는 필수적이므로 피치각을 줄이기 위한 항은 제외하였다. 식 (7)에서 ϕ_h, ψ_h는 각각 동체의 롤각과 요각을 의미하며 N_a는 다리 개수를 의미하며 T는 에피소드 종료시간이다. 각 항의 가중치 $\alpha_j (j = 1, \ldots, 6)$는 10, 16, 2, 0.1, 0.1, 0.02이다.

$$R = r_s - \Sigma_{t=0}^{T} [\alpha_1 (\ddot{x}_h^2 + \ddot{y}_h^2 + \ddot{z}_h^2) + \alpha_2 (\phi_h^2 + \psi_h^2) + \alpha_3 (\dot{\phi}_h^2 + \dot{\theta}_h^2 + \dot{\psi}_h^2) + \alpha_4 \Sigma_i^{N_a} \dot{\theta}_{a_i}^{\ 2} + \alpha_5 \Sigma_k^{\frac{N_a}{2}} (\theta_{a_{2k}} - \theta_{a_{2k+1}})^2 + \alpha_6 \cdot 1] \quad (7)$$

4.3 에피소드 중단조건

에피소드 진행 중 피해야 할 현상이 발생하면 시뮬레이션을 중단하고 done을 true로 반환한다. 본 연구에서는 에피소드 시간이 15초를 초과하거나 로봇이 5초간 진행하지 못하거나 동체 또는 다리가 장애물에 접촉하거나 휠과 휠이 접촉하거나 동체중심이 경로를 50 cm 이상 벗어나거나 요각 오차가 45도를 초과하거나 전방 다리 회전각이 -150도와 50도 범위를 벗어나거나 후방 다리 회전각이 -50도와 150도 범위를 벗어나면 중단조건이 만족되도록 한다.

4.4 3차원 비정형 장애물 생성

실제 환경에서의 장애물을 극복하고 환경인식 센서의 노이즈에 대비하기 위하여 강화학습 시 [Algorithm 2]를 이용하여 3차원 랜덤 노면형상을 생성하며 학습을 진행한다.

[Algorithm 2]에서 U, clip, linspace, interp1d는 각각 균등분포(uniform distribution), 첫 번째 인자를 두 번째와 세 번째 인자 사이의 값으로 강제로 제한하는 클립함수, 첫 번째 인자부터 두 번째 인자까지 등간격으로 증가하되 원소개수가 세 번째 인자인 벡터를 생성하는 함수, 그리고 1차원 선형보간 함수를 의미한다. 랜덤 노면형상 생성 시 필요한 파라미터는 [Table 5]와 같다. $\eta_{\max}$가 0이면 평평한 노면이 생성되며 값이 클수록 노면의 거칠기(roughness)가 증가한다.

[Algorithm 2] Random Heightmap Generation

Input: minimum and maximum increment $\Delta x_{\min}$, $\Delta x_{\max}$, $\Delta z_{\max}$, max noise height $\eta_{\max}$, ground length L_g, min & max position of two blocks $x^{b1}_{\min}$, $x^{b1}_{\max}$, $x^{b2}_{\min}$, $x^{b2}_{\max}$, heights of two blocks h_{b1}, h_{b2}, the number of samples N_g

Initialize x_0, $z_0 \leftarrow 0$, $k \leftarrow 0$

while $x_k < L_g$ **do**

  $\Delta x_k \sim \mathrm{U}(\Delta x_{\min}, \Delta x_{\max})$

  $\Delta z_k \sim \mathrm{U}(-\Delta z_{\max}, \Delta z_{\max})$

  $x_{k+1} \leftarrow x_k + \Delta x_k$

  $z_{k+1} \leftarrow \mathrm{clip}\,(z_k + \Delta z_k, -\eta_{\max}, \eta_{\max})$

  $k \leftarrow k+1$

  if $x^{b1}_{\min} < x_k < x^{b1}_{\max}$ **then**

    $z_k \leftarrow z_k + h_{b1}$

  else if $x^{b2}_{\min} < x_k < x^{b2}_{\max}$ **then**

    $z_k \leftarrow z_k + h_{b2}$

  end if

end while

$\boldsymbol{x} \leftarrow [x_0, x_1, \dots, x_K]^T$

$\boldsymbol{z} \leftarrow [z_0, z_1, \dots, z_K]^T$

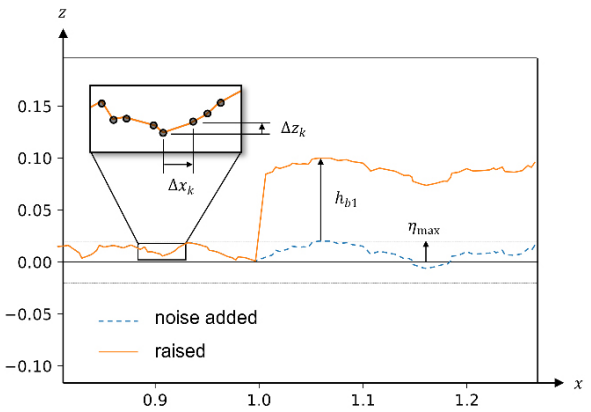
$\hat{\boldsymbol{x}} \leftarrow \mathrm{linspace}(0, L_g, N_g)$

$\hat{\boldsymbol{z}} \leftarrow \mathrm{interp1d}(\hat{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{z})$

return $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}$

[Table 5] Parameters for generating random obstacle shape and random friction coefficients

Parameters	Value	Parameters	Value
$\Delta x_{\min}$	0.1 cm	$\Delta x_{\max}$	1 cm
$\Delta z_{\max}$	5 cm	$\eta_{\max}$	2 cm
L_g	10 m	N_g	1000
$\mu_{\min}$	0.3	$\mu_{\max}$	1.0



[Fig. 7] Example of random obstacle generation

[Algorithm 2]를 이용하여 좌측과 우측의 노면형상을 독립적으로 생성하면 3차원 비정형 장애물을 생성할 수 있다. 랜덤 노면을 생성한 예를 [Fig. 7]에 나타내었다.

실제 장애물에 가깝게 모사하기 위하여 휠-노면 마찰계수도 식 (8)과 같이 랜덤한 값으로 변경한다.

$$\mu \sim \mathrm{U}(\mu_{\min}, \mu_{\max}) \tag{8}$$

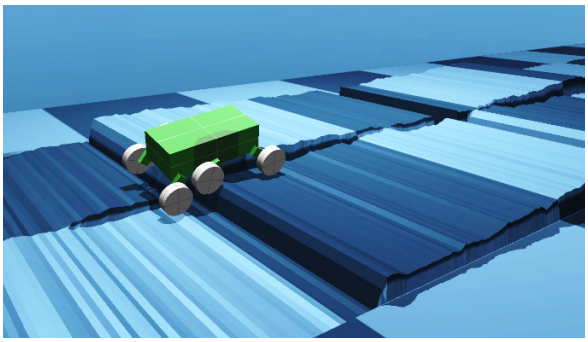
4.5 학습환경

제안한 강화학습으로 DNN 모델을 학습하기 위하여 [Fig. 8]과 같이 MuJoCo[11]를 이용한 학습 환경을 구성하였다. C++ 언어를 이용하여 MuJoCo의 C API를 클래스로 추상화하여 초기화, 로봇 상태 획득, 제어입력 적용, 시뮬레이션 가시화 등이 가능하도록 하였다. 특히 학습시간을 감소시키기 위하여 보유하고 있는 컴퓨터의 CPU 코어수(16개)만큼 MuJoCo 환경을 복수로 생성하여 동시에 실행할 수 있도록 벡터화된 환경(vectorized environment)을 구성하였다. DNN 모델 학습 프레임워크로 Pytorch를 사용하였고 Pybind11[12]과 Eigen3[13]를 이용하여 C++ 코드를 파이썬과 바인딩함으로써 Pytorch와 상호 연동되도록 하였다.

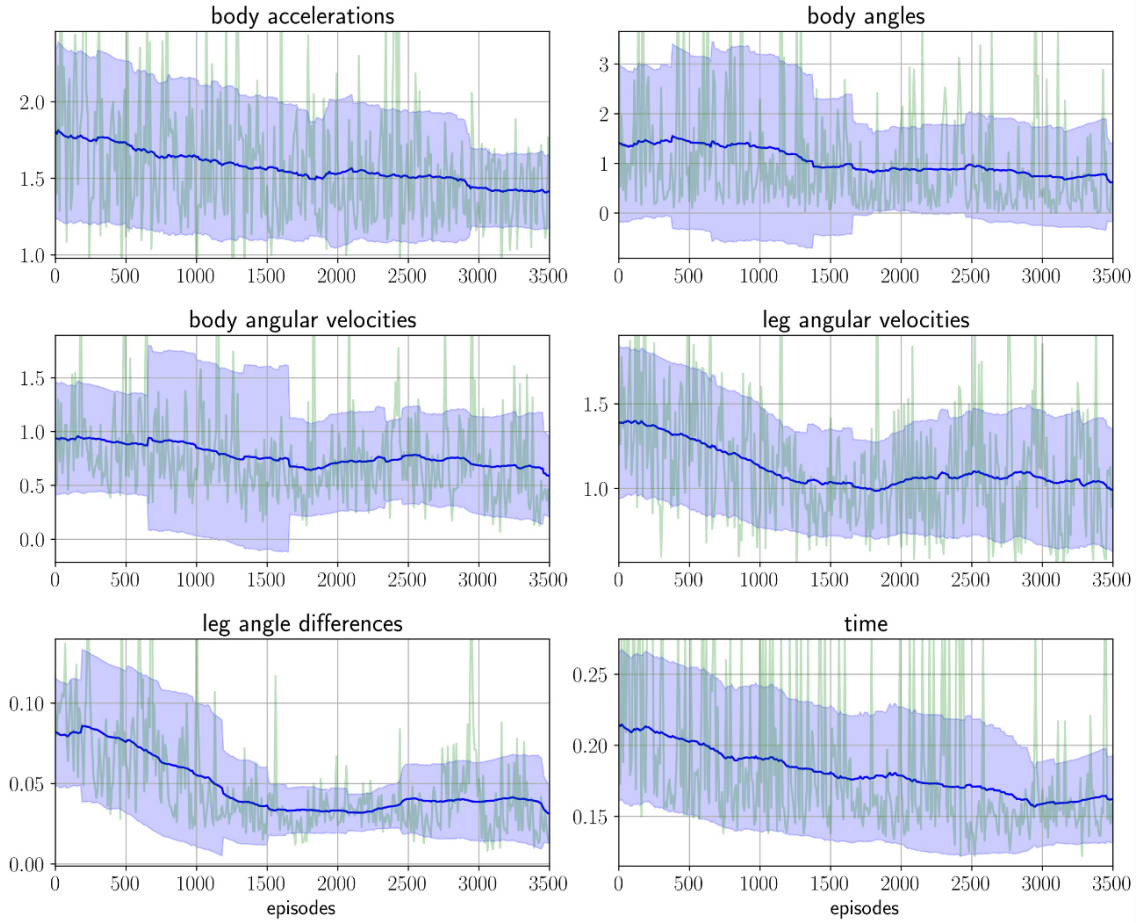
상기의 강화학습은 Ubuntu 20.04, AMD Raizen 3700X CPU, 16 GB RAM으로 구성된 컴퓨터에서 수행하였다.

4.6 강화학습 결과

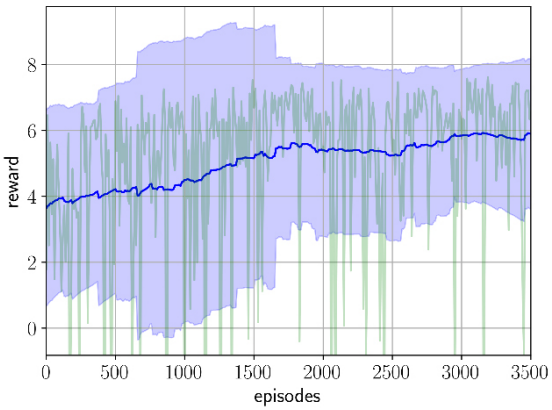
강화학습 시 에피소드가 증가함에 따라 변화하는 비용함수 세부항목과 보상함수 값을 [Fig. 9]와 [Fig. 10]에 나타내었다. 동일한 노면에 대하여 10회 반복하여 학습하기 때문에 10회당 마지막 1회의 에피소드 결과만 나타내었다. 부드러운 선은 미가공 데이터(raw data)에 윈도우를 이동하며 구한 이동평균(moving average, 기준 위치로부터 ±50개 범위의 데이터를 이용하여 계산한 평균)이다. 반투명으로 표시한 영역은 이동평균을 기준으로 1시그마(1 sigma) 범위를 나타낸다. 여기서 표준편차는 이동평균을 계산한 동일한 윈도우 데이터를 사용하



[Fig. 8] MuJoCo simulation of a 6×6 wheel-legged robot with 3D randomized obstacles



[Fig. 9] Cost learning curves: Smooth lines show the averaged values using the window of size ±50. Non-smooth lines are raw data. Filled areas represent the range of 1 sigma

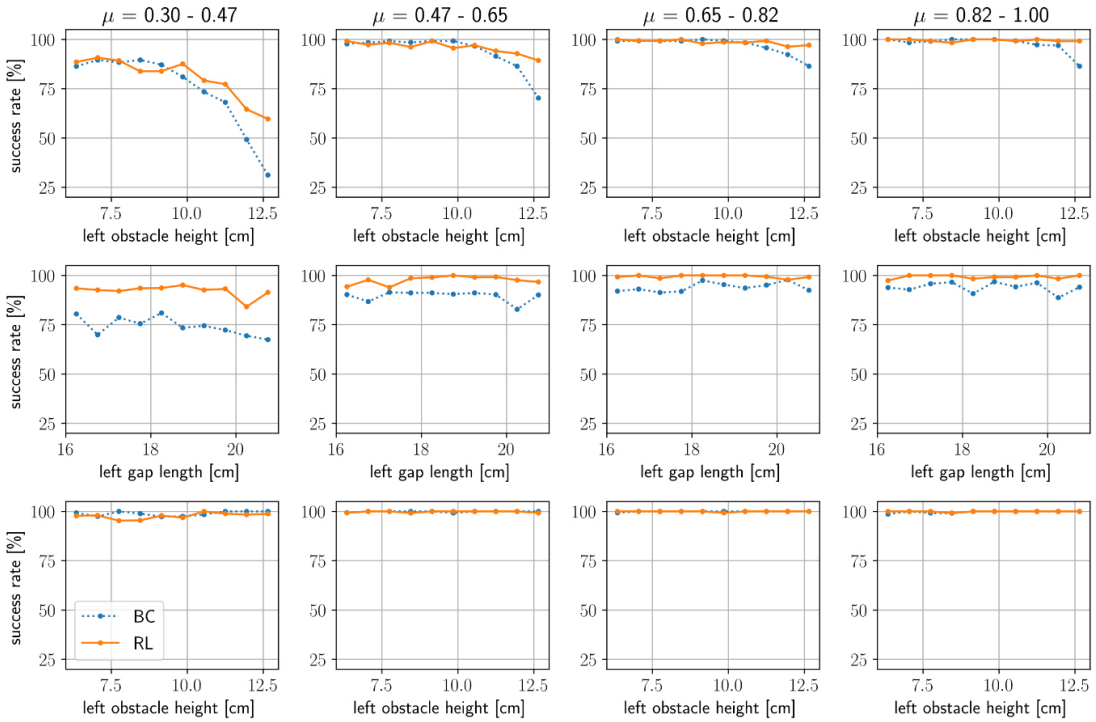


[Fig. 10] Reward learning curve: The meaning of each line and area are the same as [Fig. 9]

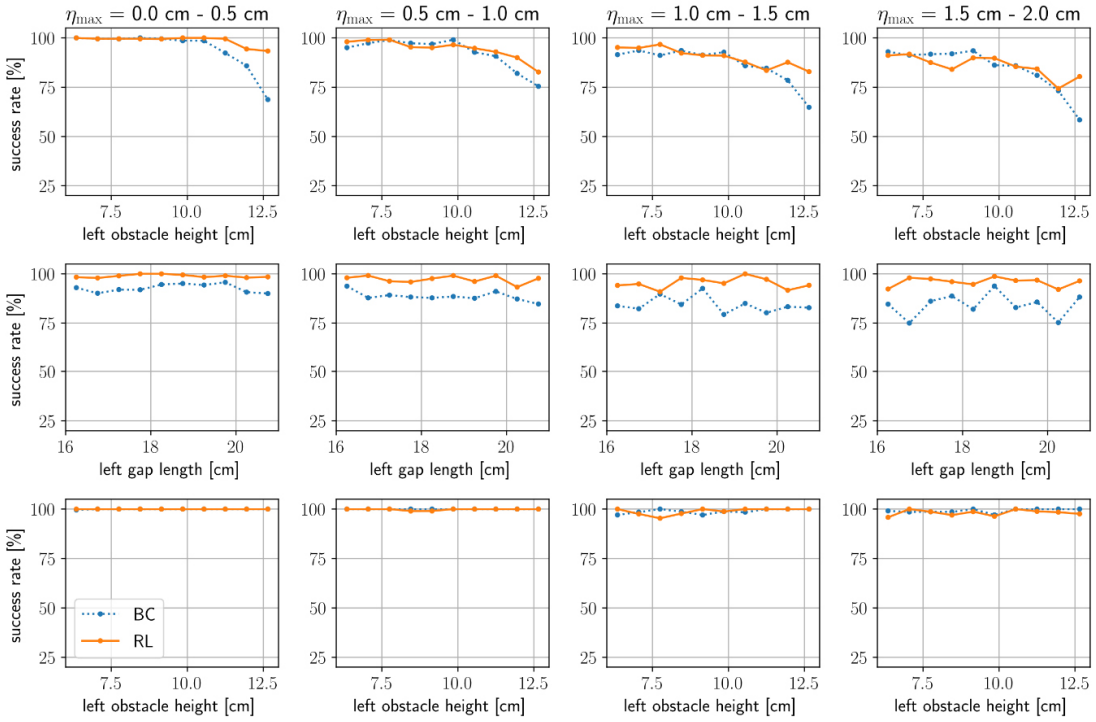
였다. 비용마다 차이는 있지만 강화학습 초기 약 1500회까지는 비용이 감소하고 보상은 상승함을 확인할 수 있다. 약 1500회부터는 비용과 보상에 큰 변화가 없다.

강화학습을 수행하기 전후의 DNN 모델 성능을 비교하기 위하여 탐색을 하지 않는 결정론적 DNN 모델을 이용하여 수직장애물 상승, 참호 통과, 수직장애물 하강 순으로 장애물을 극복하는 시뮬레이션을 5000번씩 수행하였다. 매번 장애물 형상을 랜덤하게 생성하고 노면 마찰력을 랜덤하게 설정하였다.

[Fig. 11]과 [Fig. 12]는 장애물극복 성공률을 장애물의 크기와 노면 마찰계수 또는 노면 거칠기에 따라 분류하여 비교한 결과이다. 두 그림에서 세 개의 행(row)은 위로부터 각각 수직장애물 상승, 참호 통과, 수직장애물 하강에 대한 성공률을 나타낸다. 네 개의 열(column)은 [Fig. 11]의 경우 노면 마찰계수의, [Fig. 12]는 노면 거칠기의 네 개 구간별 성공률을 나타낸 결과이다. 각 그래프의 가로축은 수직장애물의 경우 좌측 장애물의 최대 높이, 참호인 경우 좌측 장애물의 최대 길이를 나타낸다. 좌우측 장애물의 높이 또는 길이를 평균하면 중심극한정리(central limit theory)에 의하여 낮거나 높은 구간의 데이터 수가 감소하기 때문에 한 쪽 길이를 사용하였다. 세로축은 모두 장애물극복 성공률을 의미한다. 점선은 모방학습만을 수행한 모델의 성능이며 실선은 강화학습까지 수행한 모델의 성능이다. 수직장애물 하강인 경우 모방학습만으로도 대부분 90% 이상의 성공율을 보이기 때문에 강화학습의 효과가 크지



[Fig. 11] Success rates of obstacle crossing with DNN models trained by behavioral cloning and reinforcement learning. Results for step up, trench, and step down are shown in the top, middle, and bottom rows, respectively. Columns represent results for four friction coefficient ranges



[Fig. 12] All settings are the same as [Fig. 11] except columns represent results for four ground roughness ranges

않다. 하지만 수직장애물 상승과 참호 통과의 경우 모방학습만 수행하면 장애물의 크기가 크고 마찰력이 작으며 거칠기가 클수록 성공률이 저조하며 강화학습을 통하여 많게는 28.5%p 향상시킬 수 있다. 단, 수직장애물 상승의 일부 구간에서는 강화학습을 수행하면 성공률이 다소 낮아지는 형상을 보이기도 하였다.

[Table 6] Comparison of calculation time for trajectory optimization and deep neural network

[sec]	step up	trench	step down
TO	23.9 (max) 6.79 (mean) 1.61 (min)	12.3 3.33 0.91	12.2 3.73 1.30
DNN	0.0210 0.00820 0.00592	0.0221 0.00809 0.00591	0.0206 0.00794 0.00591

본 연구의 중요한 목적인 모션 계획 속도향상에 대한 효과를 확인하기 위하여 경로최적화(TO)와 DNN을 이용하여 각 장애물에 대한 모션 계획을 각각 1000번씩 수행하며 모션 계획 시간을 측정한 결과의 최대값, 평균값, 최소값을 [Table 6]에 나타내었다. DNN을 사용함으로써 평균속도 기준으로 모션 계획 속도가 약 410배에서 820배 향상되었다.

5. 결 론

본 연구에서는 6×6 휠-다리 로봇의 장애물극복을 위하여 기 제안한 경로최적화 기반 모션 계획 및 제어 기법의 한계를 극복하기 위하여 DNN 모델을 이용하여 모션 계획 속도를 증대하고 장애물극복 성공률을 높이는 방법을 제시하였다. DNN 모델이 모션 계획을 위한 키프레임을 출력하고 이를 기반으로 스플라인 곡선으로 제어명령을 생성하도록 구성한 후 모방학습으로 초기성능을 구현하고 강화학습으로 성능을 극대화하는 방법을 사용하였다. 학습된 모델을 사용하면 경로최적화 기법으로는 성공하기 어렵거나 많은 계산시간이 소요될 3차원의 복잡한 형상을 갖는 장애물도 높은 성공률로 통과하는 결과를 보여주었다. 또한 강화학습을 통하여 저마찰 구간 등의 성공률을 모방학습 대비 최대 28.5%p까지 향상시킬 수 있었다.

후속 연구로는 실제 로봇을 제작하고 노면형상 감지기는 라이다 등 환경센서로 구현하고 키프레임 생성기는 소형 딥러닝 연산기에 적용하는 등 본 연구에서 제안한 방법을 실제 로봇에 적용하는 sim2real 연구를 고려하고 있다.

References

[1] S. Jeong and M. Won, “Motion Planning and Control of Wheel-legged Robot for Obstacle Crossing,” *The Journal of Korea Robotics Society*, vol.17, no.4, pp. 500-507, Dec., 2022, DOI: 10.7746/jkros.2022.17.4.500.

[2] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, “DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning,” *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1-13, Jul., 2017, DOI: 10.1145/3072959.3073602.

[3] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, “DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699-3706, Apr., 2020, DOI: 10.1109/LRA.2020.2979660.

[4] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, “Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning,” *Robotics*, 2021, DOI: 10.48550/arXiv.2109.11978.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: a brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38, Nov., 2017, DOI: 10.1109/MSP.2017.2743240.

[6] M. Bain and C. Sammut, “A framework for behavioural cloning,” *Machine Intelligence 15*, 1995, [Online], https://www.semanticscholar.org/paper/A-Framework-for-Behavioural-Cloning-Bain-Sammut/1f4731d5133cb96ab30e08bf39dffa874aebf487, Accessed: Feb., 13, 2023.

[7] J. D. Chang, M. Uehara, D. Sreenivas, R. Kidambi, and W. Sun, “Mitigating covariate shift in imitation learning via offline data with partial coverage,” *Machine Learning*, 2021, DOI: 10.48550/arXiv.2106.03207.

[8] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural networks*, vol. 21, no. 4, pp. 682-697, May, 2008, DOI: 10.1016/j.neunet.2008.02.003.

[9] M. S. Jazayeri and A. Jahangiri, “Utilizing b-spline curves and neural networks for vehicle trajectory prediction in an inverse reinforcement learning framework,” *Journal of Sensor and Actuator Networks*, vol. 11, no. 1, Feb., 2022, DOI: 10.3390/jsan11010014.

[10] M. Hasanzade and E. Koyuncu, “A dynamically feasible fast replanning strategy with deep reinforcement learning,” *Journal of Intelligent & Robotic Systems*, vol. 101, 2021, DOI: 10.1007/s10846-020-01274-1.

[11] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” *IEEE International Workshop on Intelligent Robots and Systems (IROS)*, Vilamoura-Algarve, Portugal, 2012, DOI: 10.1109/IROS.2012.6386109.

[12] pybind11 − Seamless operability between C++11 and Python, [Online], https://github.com/pybind/pybind11, Accessed: Nov. 1, 2022.

[13] Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms, [Online], https://eigen.tuxfamily.org, Accessed: Nov. 1, 2022.

정 순 규

1996 한양대학교 기계설계공학과(학사)
1998 GIST 기전공학과(석사)
2020~현재 충남대학교 메카트로닉스공학과 박사과정

관심분야: Wheel-legged robot, Legged robot, Trajectory optimization, Reinforcement learning

원 문 철

1983 서울대학교 조선공학과(학사)
1985 서울대학교 조선공학과(석사)
1995 U.C. Berkeley Mechanics(Ph.D.)
1987~1990 한국기계연구원
1995~현재 충남대학교 메카트로닉스공학과 교수

관심분야: Control of vehicles and mechatronics systems, AI, Self-driving vehicles, Robots, Computer vision