Journal of Korea Robotics Society (2023) 18(4):427-435
https://doi.org/10.7746/jkros.2023.18.4.427 ISSN: 1975-6291 / eISSN: 2287-3961

모바일 로봇을 위한 학습 기반 관성-바퀴 오도메트리

Learning-based Inertial-wheel Odometry for a Mobile Robot

김 명 수[1] · 장 근 우[2] · 박 재 흥[†]
Myeongsoo Kim[1], Keunwoo Jang[2], Jaeheung Park[†]

Abstract: This paper proposes a method of estimating the pose of a mobile robot by using a learning model. When estimating the pose of a mobile robot, wheel encoder and inertial measurement unit (IMU) data are generally utilized. However, depending on the condition of the ground surface, slip occurs due to interaction between the wheel and the floor. In this case, it is hard to predict pose accurately by using only encoder and IMU. Thus, in order to reduce pose error even in such conditions, this paper introduces a pose estimation method based on a learning model using data of the wheel encoder and IMU. As the learning model, long short-term memory (LSTM) network is adopted. The inputs to LSTM are velocity and acceleration data from the wheel encoder and IMU. Outputs from network are corrected linear and angular velocity. Estimated pose is calculated through numerically integrating output velocities. Dataset used as ground truth of learning model is collected in various ground conditions. Experimental results demonstrate that proposed learning model has higher accuracy of pose estimation than extended Kalman filter (EKF) and other learning models using the same data under various ground conditions.

Keywords: IMU, Learning, Mobile Robot, Odometry, Wheel

1. 서 론

바퀴 기반 모바일 로봇은 현재 실내와 실외에서 주행을 하면서 안내를 하거나 물건을 배달하는 등 여러 작업을 진행하고 있다. 이때 오도메트리(odometry)를 이용한 정확한 자세(pose) 예측이 중요하다. 오도메트리는 센서 정보를 기반으로 상대적인 위치를 계산하는 것을 말한다. 모바일 로봇의 자세 예측 시 발생할 수 있는 오차의 원인으로는 시스템적 오차와 비시스템적 오차가 있다. 먼저, 기구학적 모델링과 관련된 시스템적 오차는 기구학적 보정[1-3]을 통해 오차를 잡을 수 있다. 하지만 미끄러짐 등의 로봇과 주행 환경 간의 상호작용으로 인한 비시스템적 오차는 예측하기 어렵다[4,5]. 보통 이러한 오차를 줄이기 위해 바퀴 엔코더 뿐만 아니라 IMU, 카메라, 라이다 등의 다른 외부 센서를 추가적으로 이용하는 방법들이 제안되었다[6-10]. 하지만 이러한 연구들도 모델링하기 힘든 비시스템적 요인으로 인해 오차를 줄이는 데에는 한계가 있다. 따라서, 본 연구에서는 모바일 로봇의 바퀴 엔코더와 IMU 데이터를 이용하여 Long Short-Term Memory (LSTM) 네트워크를 통해 학습된 모델을 기반으로 모바일 로봇의 위치와 방향 예측 성능을 향상하는 방법을 제안한다.

1.1 관련 연구

전통적으로, 모바일 로봇의 자세 예측 정확도를 향상시키기 위해 채택된 방법은 바퀴 엔코더와 IMU 데이터를 이용해 Extended Kalman Filter (EKF)를 사용해 예측하는 것이었다[11]. 이는 바퀴 엔코더만을 이용해 자세를 예측하는 것보다는 오차가 적지만, IMU 센서에 noise나 bias 뿐만 아니라 모델링 하기

Received : May. 27. 2023; Revised : Jul. 10. 2023; Accepted : Aug. 30. 2023
※ This work was supported by Imdang Scholarship & Cultural Foundation
1. M.S Degree, Graduate School of Convergence Science and Technology, Seoul National University, Seoul, Korea (kimms74@snu.ac.kr)
2. Researcher, Artificial Intelligence and Robotics Institute, Korea Institute of Science and Technology, Seoul, Korea (jang90@kist.re.kr)
† Professor, Corresponding author: Graduate School of Convergence Science and Technology, ASRI, RICS, Seoul National University, Seoul, Korea; Advanced Institutes of Convergence Technology, Suwon, Korea (park73@snu.ac.kr)

힘든 외부 환경 요인으로 인해 정확도를 향상시키는 데에 한계가 존재하였다.

최근에는 다양한 분야에서 위치나 회전 방향의 예측 정확도를 높이기 위해 학습 모델을 이용하기 시작했다[12-19]. WhONet[12]의 경우 자동차의 바퀴 엔코더 데이터를 입력값으로 이용하여 위치를 출력값으로 내보내는 학습 모델을 이용한다. 하지만, 바퀴 엔코더 데이터만을 이용할 경우 시스템적 오차인 기구학적 모델링과 관련된 오차만 파악이 가능하다. 비시스템적 오차를 파악할 수 있는 IMU, 카메라, 라이다 등의 외부 센서를 이용하지 않았기에 모델링 되지 않은 오차를 잡기에는 한계가 있다.

이와 달리 IMU 데이터만을 이용해 위치나 회전 방향을 예측하는 연구들이 있다[13-18]. AI-IMU[13]는 IMU 데이터를 학습 모델의 입력값으로 이용하여, EKF의 파라미터를 학습하는 데 사용하였다. Orinet[14]과 M. Brossard의 연구[15]는 오직 IMU 데이터를 이용하여 회전 방향을 예측하는 데 학습 모델을 이용하였다. Ronin[16]과 Tlio[17]의 경우에도 IMU 데이터를 입력값으로 이용해 ResNet 학습 모델을 통해서 위치를 계산하는 end-to-end 학습 방법을 사용한다. 이 방법은 학습 모델 내부에 위치를 계산하기 위한 적분식이 포함되게 된다. M. Zhang의 연구[18]에서 적분식이 학습 모델에 포함될 경우 자세 예측 성능이 떨어짐을 보였다. 따라서 M. Zhang의 연구[18]에서는 입력값을 IMU 데이터로 이용하여 end-to-end로 위치를 출력하는 것이 아닌 입력값을 보정하는데 LSTM 학습 모델을 결합하였다[18]. 하지만 IMU 데이터만을 이용해 자세를 예측할 경우 IMU와 바퀴 엔코더 데이터를 이용한 기존의 EKF 방식보다 자세 예측 정확도가 낮았다.

LWOI는 바퀴 엔코더 데이터와 자이로, 그리고 IMU 데이터를 사용하여 자세 예측을 진행한다. Gaussian Process (GP)의 파라미터를 학습시키는데 학습 모델을 이용하였다. GP를 통과시킨 데이터를 이용해 EKF로 자세를 예측한다.

1.2 연구 기여

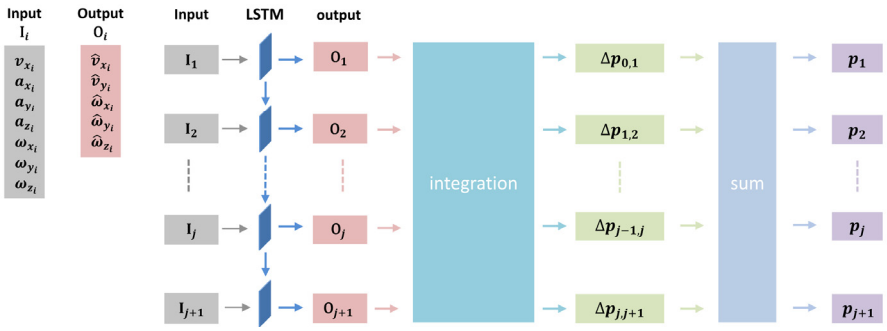
제안하는 방법은 LSTM을 기반으로 한 학습 모델을 통해 속도와 각속도 데이터를 구한 후 적분을 통해 자세를 구하게 된다. 속도 데이터는 바퀴 엔코더와 외부 환경의 변화에 영향을 적게 받으면서 데이터를 얻을 수 있는 IMU를 추가로 이용하여 취득하였다. LSTM의 손실 함수는 회전 방향과 위치가 적분을 통해 계산되므로 시간이 지남에 따라 누적된다는 특징을 이용하여 설계하였다. 학습된 모델은 다양한 바닥 상황에 대해서도 미끄러짐에 대한 특성을 잘 반영하여 위치 뿐만 아니라 회전 각도에 대해서 정확도를 향상했다.

2. 학습 방법

2.1 네트워크 구조

본 연구에서 사용하는 신경망은 LSTM을 이용한다. LSTM의 경우 이전 정보를 가지고 있으면서 새로운 정보를 받아 학습을 진행하는 특징이 있다. 로봇의 자세는 속도 v와 각속도 ω를 계산을 통해 누적해 얻어지는 위치와 회전 방향이다. 로봇의 자세 또한 이전 정보의 누적으로 구성됐기에 LSTM의 학습 방식이 적절하다고 생각하였다. 또한 미끄러짐이 한 순간의 데이터에만 영향을 받는 것이 아닌 그 시점 이전의 데이터들도 영향을 끼칠 것이라 생각하여 LSTM을 이용하였다.

제안하는 네트워크는 stacked unidirectional LSTM을 사용한다. LSTM의 경우 레이어는 3개, 각 레이어 당 크기는 120으로 돼 있다. 네트워크를 이용한 자세를 예측하는 전체적인 과정은 [Fig. 1]에 나와 있다. 입력값으로 바퀴 엔코더 데이터를 이용해 구한 모바일 로봇의 직진 방향 속도 v_x, IMU의 각속도 $\omega_x,\omega_y,\omega_z$, 가속도 a_x,a_y,a_z를 사용한다. 입력값이 LSTM을 통과하면 출력값으로 보정된 속도 $\hat{v}_x,\hat{v}_y$, 보정된 IMU 각속도 $\hat{\omega}_x,\hat{\omega}_y,\hat{\omega}_z$가 나오게 된다. v_x가 들어가서 $\hat{v}_x,\hat{v}_y$가 나오는 이유는 본 연구에서 사용하는 로봇은 입력으로 직진 속도 v_x만 주어지고 실제 로봇의 속도는 미끄러짐으로 인한 v_y도 존재하기 때문이다. 출력값을 적분하여 평면 상의 위치 변화값을 구하고, 값들을 누적시켜 자세를 예측하게 된다. $\hat{v}_z$를 예측하지 않



[Fig. 1] Proposed pose estimation architecture

는 이유는 z축에 대한 데이터는 IMU의 가속도 a_z 뿐이므로 가속도 데이터만으로 속도를 예측하게 되면 네트워크 내부에 적분식이 포함돼 성능이 떨어지게 되기 때문이다.

$$p_n = p_{n-1} + \mathrm{R}_{n-1}(v_{n-1}\mathrm{dt}) \tag{1}$$

$$\hat{p}_n = \hat{p}_{n-1} + \hat{\mathrm{R}}_{n-1}(\hat{v}_{n-1}\mathrm{dt}) \tag{2}$$

p는 ground truth의 회전 행렬 R 과 속도 v를 이용하여 적분을 통해서 구한 위치다. $\hat{p}$는 네트워크 출력값의 회전 행렬 $\hat{\mathrm{R}}$ 과 속도 $\hat{v}$를 이용하여 적분을 통해서 구한 위치다.

$$\mathrm{R}_n = \mathrm{R}_{n-1}\exp_{\mathrm{SO}(3)}(\omega_n \mathrm{dt}) \tag{3}$$

$$\hat{\mathrm{R}}_n = \hat{\mathrm{R}}_{n-1}\exp_{\mathrm{SO}(3)}(\hat{\omega}_n \mathrm{dt}) \tag{4}$$

R 은 ground truth 각속도 ω를 이용해 구한 회전 행렬이다. $\hat{\mathrm{R}}$는 네트워크의 출력값 각속도 $\hat{\omega}$를 이용해 구한 회전 행렬이다. $\exp_{\mathrm{SO}(3)}$는 SO(3) 에서의 exponential map으로 3자유도 각도값을 회전 행렬로 변환시켜준다. 식 (3)과 식 (4)의 경우 매번 회전 행렬끼리 곱셈을 진행하게 될 때 아주 작은 오차들이 누적되면서 값이 틀어지는 현상이 발생했다. 따라서 매번 행렬 곱셈을 할 때마다 특이값 분해 (SVD)를 이용해 회전 행렬을 정규화 작업을 하였고, 식 (3)의 R 을 ground truth 회전 행렬 R 과 비교했을 때 일치하는 것을 확인하였다.

Ground truth의 p와 R 를 데이터셋에서 측정된 값을 사용하지 않고 계산을 통해서 구하는 이유는 학습 시 overfitting을 막으면서 학습 성능을 높여주기 위해 training 데이터셋을 재정렬하는 작업을 진행하기 때문이다. 매번 같은 training 데이터를 이용해 학습을 진행하면 overfitting이 발생할 가능성이 높아진다. 따라서 일정한 시간 간격으로 나눈 training 데이터셋을 매번 학습 시 재정렬한다. 재정렬된 데이터의 속도 v와 각속도 ω에 따라 회전 방향과 위치가 바뀌게 돼 ground truth에서 구한 회전 방향과 위치를 그대로 사용할 수 없다. 따라서 식 (2)와 식 (4)와 같이 p와 R 를 계산하여 사용한다.

2.2 손실 함수

손실 함수를 어떻게 설계하느냐에 따라 학습 모델의 성능이 달라질 수 있다. 학습 모델은 로봇의 속도 $\hat{v}_x, \hat{v}_y$, 로봇의 각속도 $\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z$를 출력하므로 각각에 대해서 손실 함수를 만들 수 있다.

$$L = k * L_\omega + L_p \tag{5}$$

손실함수 L는 회전 손실 함수 L_ω와 위치 손실 함수 L_p를 더해서 구한다. 각속도를 이용해 회전 손실 함수 L_ω를 구하고, 속도를 이용해 위치 손실 함수 L_p를 구한 후 합하여 사용한다. 손실 함수를 두 개로 나누어 진행한 이유는 네트워크에서 출력된 각속도와 속도를 같이 이용해 위치를 계산하여 학습을 진행할 경우 학습 성능이 떨어졌기 때문이다. 이는 속도와 각속도를 적분해 위치를 구하게 되면 두 데이터가 위치 예측에 영향을 끼치는 정도가 다르므로 한쪽에만 우세하게 학습이 진행된다. 따라서 속도와 각속도 각각에 대해서 위치 손실 함수 L_p와 회전 손실 함수 L_ω를 만들어 가중치 파라미터 k를 이용해 둘 다 적절히 학습이 될 수 있도록 하였다. 회전 손실 함수의 경우 각도 단위가 radian으로 되어있어 손실의 크기가 작았고, IMU의 각속도 데이터로 인한 오차가 바퀴 엔코더 속도로 인한 오차에 비해 적었기에 회전 손실 함수에 k를 곱하여 사용하였다. 본 논문에서는 $k = 1500$을 이용하였다.

회전 행렬과 위치의 경우 적분을 통해 계산되므로 시간이 지남에 따라 누적할 수 있는 특징이 있다. 이 특징을 이용하여 회전 손실 함수와 위치 손실 함수를 정의하였다. 먼저 회전 손실 함수는 다음과 같이 정의된다:

$$L_w = L_{\omega,1} + L_{\omega,2} + L_{\omega,4} + L_{\omega,8} + L_{\omega,16} \tag{6}$$

$$\begin{aligned} L_{\omega,j} &= \textstyle\sum_{\mathrm{i}=0}^{\mathrm{N}/j} \rho(\log_{\mathrm{SO}(3)}(\delta\hat{\mathrm{R}}^{\mathrm{T}}_{j\times i, j\times i+j}\,\delta\mathrm{R}_{j\times i, j\times i+j})) \\ L_{\omega,1} &= \textstyle\sum_{\mathrm{i}=0}^{\mathrm{N}} \rho(\log_{\mathrm{SO}(3)}(\delta\hat{\mathrm{R}}^{\mathrm{T}}_{i,i+1}\,\delta\mathrm{R}_{i,i+1})) \\ L_{\omega,16} &= \textstyle\sum_{\mathrm{i}=0}^{\mathrm{N}/16} \rho(\log_{\mathrm{SO}(3)}(\delta\hat{\mathrm{R}}^{\mathrm{T}}_{16i,16i+16}\,\delta\mathrm{R}_{16i,16i+16})) \end{aligned} \tag{7}$$

$$\delta\mathrm{R}_{i,i+j} = \mathrm{R}_i^{\mathrm{T}}\mathrm{R}_{i+j} \tag{8}$$

$$\delta\hat{\mathrm{R}}_{i,i+j} = \hat{\mathrm{R}}_i^{\mathrm{T}}\hat{\mathrm{R}}_{i+j} \tag{9}$$

$$\rho(a) = \begin{cases} \dfrac{1}{2}\|a\|^2, & if\|a\| < \delta \\ \delta^*(\|a\| - \dfrac{1}{2}*\delta), & otherwise \end{cases} \tag{10}$$

$L_{\omega,\mathrm{j}}$는 학습에 이용되는 N 개의 총 데이터에서 j 의 간격만큼 회전 행렬을 누적시킨 후의 손실을 나타낸다. ρ는 Huber 손실 함수를 나타내고 $\log_{\mathrm{SO}(3)}$는 SO(3) 에서의 logarithm map으로 회전 행렬 R 을 각도 $\phi \in \mathbb{R}^3$로 변환시켜준다. $\delta\mathrm{R}_{i,i+j}$와 $\delta\hat{\mathrm{R}}_{i,i+j}$은 각각 ground truth와 네트워크의 출력값으로 구한 j 간격 동안 회전한 회전 행렬들이다.

회전 손실 함수 L_ω에는 하나의 $L_{\omega,\mathrm{j}}$만 사용하지 않고 여러 개를 더하여 사용한다. 이는 $L_{\omega,\mathrm{j}}$에서 j가 작을 때와 클 때의 단점을 상쇄시키기 위한 방식이다. $L_{\omega,1}$처럼 j가 작을 때는 매

순간의 출력 회전 행렬이 ground truth에 비해 오차가 적게 발생하였지만, 회전 행렬을 누적시킨 후 비교하였을 때 출력 회전 행렬이 ground truth에 비해 오차가 크게 발생하였다. 반대로 $L_{\omega,16}$처럼 j가 클 때는 매 순간의 출력 회전 행렬이 ground truth에 비해 오차가 많이 발생했지만, 회전 행렬을 누적시킨 후 비교하였을 때 출력 회전 행렬이 ground truth에 비해 오차가 적게 발생하였다. 따라서 L_{ω}과 같이 j가 작을 때와 클 때 모두를 이용하여 손실 함수를 사용할 때 매 순간의 데이터와 누적 데이터의 오차가 적었다.

위치 손실 함수 L_p는 다음과 같이 정의된다:

$$L_p = L_{p,1} + L_{p,2} + L_{p,4} \tag{11}$$

$$L_{p,j} = \sum_{i=0}^{N/j} \rho(\delta\bar{p}_{j\times i, j\times i+j} - \delta p_{j\times i, j\times i+j}) \tag{12}$$
$$L_{p,1} = \sum_{i=0}^{N} \rho(\delta\bar{p}_{i,i+1} - \delta p_{i,i+1})$$
$$L_{p,4} = \sum_{i=0}^{N/4} \rho(\delta\bar{p}_{4i,4i+4} - \delta p_{4i,4i+4})$$

$$\delta p_{i,i+j} = p_{i+j} - p_i \tag{13}$$

$$\delta\bar{p}_{i,i+j} = \bar{p}_{i+j} - \bar{p}_i \tag{14}$$

$$\bar{p}_n = \bar{p}_{n-1} + \mathrm{R}_{n-1}(\hat{v}_{n-1}\mathrm{d}t) \tag{15}$$

$L_{p,j}$는 N개의 데이터가 있을 때 j개의 간격만큼 위치를 누적시킨 후의 $\delta\bar{p}$과 δp을 이용하여 구한 손실이다. $\delta p_{i,i+j}$와 $\delta\bar{p}_{i,i+j}$는 각각 p와 $\bar{p}$를 이용해 구한 j개의 간격동안 이동한 위치 변화값들이다. 위치 손실 함수 L_p도 회전 손실 함수 L_w과 같이 하나의 $L_{p,j}$만을 이용하지 않고 여러 개를 더하여 사용하였다. $L_{p,1}$과 같이 j가 작을 경우 매 순간의 출력 속도값은 ground truth와의 오차가 작았지만, 속도값을 이용해 누적된 위치를 구했을 경우엔 오차가 크게 발생하였다. 반대로 $L_{p,4}$와 같이 j가 클 경우 매 순간의 출력 속도값은 ground truth와의 오차가 컸지만, 속도값을 이용해 누적된 위치를 구했을 경우엔 오차가 적게 발생하였다. 따라서 L_p와 같이 j를 작을 때와 클 때 모두 이용해 손실 함수를 만들어 매 순간의 출력 속도와 누적 위치 데이터의 오차를 줄였다.

위치 손실을 계산할 때는 $\hat{p}_n$ 대신 $\bar{p}_n$을 이용한다. $\bar{p}_n$에서 속도는 네트워크의 출력값 $\hat{v}$을 이용하지만, 회전 행렬은 식 (3)의 R 을 이용한다는 특징이 있다. 학습 시에 $\hat{p}$는 출력값의 속도와 회전 행렬 모두에 영향을 받지만, $\bar{p}$는 출력값의 속도에 대해서만 영향을 받는다. 또한 $\hat{\mathrm{R}}_n$은 출력값을 누적시키는 계산이 필요하지만, R_n은 이미 알고 있는 데이터이기에 위치를 구할 때 계산 시간이 줄어드는 장점이 있다. 따라서 위치 손실 함수 L_p에서 출력값의 속도에 대해서만 손실을 계산하기 위해 $\hat{p}$ 대신 $\bar{p}$를 사용한다.

3. 실험 및 결과

3.1 데이터셋 제작

3.1.1 기존 데이터셋

기존에 오도메트리 학습을 위한 다양한 종류의 데이터셋이 존재한다[20-24]. 차량을 이용해 실외 주행 데이터를 모아두기도 하고[20,21], 모바일 로봇을 이용해 실내외 주행 데이터를 모으기도 한다[22]. Micro Air Vehicle (MAV)을 이용해 실내 비행 데이터를 모아두기도 하고[23], 측정 장치를 손으로 들고 다니며 실내외에서 보행 데이터를 모으기도 하였다[24]. 본 연구에서는 바퀴 엔코더 데이터가 필요했기 때문에 기존 데이터셋 중 차량이나 모바일 로봇 데이터셋을 사용할 수 있었다. 하지만 차량 데이터셋의 경우 도로 주행이었고, 모바일 로봇 데이터셋의 경우에도 평평한 바닥에서의 주행이었다. 이 경우 각각의 데이터셋이 다양한 바닥 상황을 반영하지 못하였다. 평평한 바닥 뿐만 아니라 미끄러짐이 잘 발생할 수 있는 바닥에 대해서도 데이터가 필요했기에 직접 모션 캡처 장비를 이용해 모바일 로봇의 데이터셋을 제작하였다. 제작된 데이터셋에서는 미끄러짐이 크게 발생하지 않는 평평한 바닥과 미끄러짐이 잘 발생하는 울퉁불퉁한 바닥 모두에서 데이터를 모아 제작하였다.

3.1.2 로봇과 모션 캡처 시스템의 구성

데이터셋을 만들기 위한 로봇으로 Clearpath 회사의 Husky 모바일 로봇을 이용하였다. Skid steering 방식의 로봇으로 왼쪽 바퀴 2개, 오른쪽 바퀴 2개가 같이 움직여 차동 구동(differential-drive) 방식의 로봇에 비해 이동 시 미끄러짐이 잘 발생한다. IMU 센서는 Microstrain 회사의 3dm-gx4-25를 사용하였다. 모션 캡처는 VICON 회사의 T160 카메라 20대를 이용하여 모바일 로봇의 좌표를 측정하였다.

3.1.3 제작한 데이터셋

새로 제작된 데이터셋은 모바일 로봇의 바퀴 속도, IMU의 각속도와 가속도, 그리고 모션 캡처를 통해 측정한 3차원 위치 데이터로 이루어져 있다. Husky 로봇 위 상판의 네 모퉁이에 4개의 모션캡처 측정용 마커를 붙여 위치를 측정하였다. 이를 이용해 로봇의 방향과 회전, 속도 등의 데이터를 얻었다.

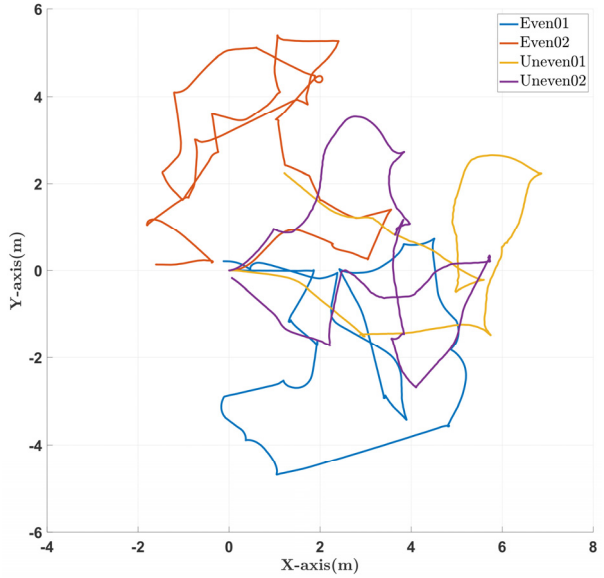
IMU와 모션 캡처 데이터의 측정 주기는 100 Hz이고 Husky 로봇의 바퀴 엔코더의 측정 주기는 25 Hz이다. 학습에 사용하기 위해 바퀴 엔코더 데이터를 100 Hz로 보간 하였고, 모든

데이터의 시간을 동기화 시켜 사용하였다.

두 가지 바닥 상황에 대해서 데이터셋을 생성하였다. [Fig. 2]와 같이 평평한 바닥에서의 움직임을 담은 데이터셋과 바퀴와 바닥 사이에 미끄러짐이 잘 발생할 수 있는 환경에서의 움직임을 담은 데이터셋을 만들었다. 총 27개의 데이터셋을 생성하였으며, 이에 대한 정보는 [Table 1]에 표기하였다. 또한, 몇 가지 데이터셋(Even 01, Even 02, Uneven 01, Uneven 02)의 경로를 [Fig. 3]에 나타내었다.

직접 제작한 데이터셋에서 바닥 상황에 따른 미끄러짐에 대한 정의가 필요하다. 본 연구에서는 미끄러짐에 대한 정의를 바퀴 엔코더로 구한 로봇의 속도와 모션 캡처를 통해 구한 로봇의 속도 차이가 모션 캡처 로봇 속도의 20%가 넘을 경우로 가정하였다. 이때 느리게 움직이는 경우, 즉 매 데이터셋의 평균 속도의 20% 이하인 데이터들은 미끄러짐 판단에서 제외하였다. 미끄러짐 정도 또한 [Table 1]에 slip rate로 표기하였다. 평평한 바닥의 데이터셋인 Even과 울퉁불퉁한 바닥의 데이터셋인 Uneven의 slip rate를 비교하면 Uneven이 Even보다 미끄러짐이 심한 것을 알 수 있다.

[Fig. 2] Even ground environment for Husky dataset (left) and uneven ground environment for Husky dataset (right)



[Fig. 3] Trajectory of Husky Dataset (Even01, Even02, Uneven01, Uneven 02)

제작한 데이터셋은 학습 시에 training, validation, testing 데이터셋으로 나눠 학습에 사용하였다. 보통 각각에 대한 비율은 7:1:2로 가져가기에 그와 비슷한 비율로 데이터셋을 구분하였다. 또한 다양한 바닥 상황에 대해서 미끄러짐에 대한 학습을 잘하기 위해 Even과 Uneven 데이터셋을 골고루 분배하였다. 다만 validation 데이터셋은 Uneven 데이터셋으로만 구성했는데 Even의 경우 전체적인 데이터 개수가 적었고, 오차가 Uneven의 경우보다 작았기에 validation에는 Uneven으로만 구성하였다.

[Table 1] Information of Produced Dataset

Produced Dataset				
Name	Path length (m)	Duration (s)	Slip rate (%)	Usage
Even 01	31.60	111.86	12.0	Training
Even 02	34.12	134.63	8.7	Training
Even 03	79.01	300.28	16.5	Training
Even 04	76.70	303.62	14.3	Training
Even 05	25.15	83.21	12.4	Testing
Even 06	43.52	167.72	13.0	Testing
Uneven 01	21.42	70.89	19.3	Training
Uneven 02	25.69	88.04	25.9	Training
Uneven 03	25.00	86.57	34.8	Training
Uneven 04	30.04	102.03	32.0	Training
Uneven 05	25.93	87.96	29.8	Training
Uneven 06	26.60	89.95	31.6	Training
Uneven 07	28.72	100.94	26.1	Training
Uneven 08	25.91	91.24	33.8	Training
Uneven 09	19.33	75.52	25.0	Training
Uneven 10	37.41	115.48	28.5	Training
Uneven 11	31.26	106.11	23.1	Training
Uneven 12	39.92	111.21	23.6	Training
Uneven 13	41.07	120.70	23.2	Training
Uneven 14	38.68	115.34	25.3	Validation
Uneven 15	25.91	91.24	26.5	Validation
Uneven 16	32.09	101.77	29.9	Validation
Uneven 17	24.34	83.11	30.5	Testing
Uneven 18	37.45	112.41	25.1	Testing
Uneven 19	43.20	121.07	29.4	Testing
Uneven 20	35.03	109.44	27.2	Testing
Uneven 21	42.20	117.24	27.2	Testing

3.2 학습을 위한 설정

PyTorch[25]를 통해 학습을 진행하였다. Optimizer는 Adam[26]을 사용하였고, 학습률은 0.02로 시작하였다. 학습을 진행할 때 validation 손실 값이 50 epochs동안 줄어들지 않으면 학습률에 0.75를 곱해 크기를 낮춰 학습을 진행하였다. 총 epochs는 2000번으로 잡았고, dropout의 경우 없을 때 성능이 더 좋게 나와 dropout 없이 사용하였다.

매 학습 시 training 데이터셋을 8초 시간 간격으로 나눈 후 그 중 일부를 무작위로 재정렬하여 새로운 입력 데이터셋을 만들었다. 이를 통해 입력 데이터셋의 양을 증가시켜 학습에 이용하였고 학습의 성능을 향상시켰다.

3.3 자세 예측 비교 대상

자세 예측 성능을 비교할 대상으로는 제안하는 방식과 같이 바퀴 엔코더와 IMU 데이터를 둘 다 사용하여 자세를 예측하는 EKF와 기존 학습 방법인 LWOI를 사용하였다. 제작된 데이터셋을 이용하여 성능 비교를 하였다. LWOI의 경우 자세를 예측할 때 IMU 뿐만 아니라 추가적인 자이로 센서를 이용하지만, 본 연구에서 제작한 데이터셋의 경우 IMU의 각속도 데이터만 가지고 있다. 따라서 같은 데이터셋을 이용한 자세 예측 성능 비교를 위해 LWOI에서 필요한 자이로 데이터 $(\omega_x,\omega_y,\omega_z)$를 IMU 데이터의 각속도로 대체하여 사용하였다.

3.4 제작한 데이터셋에 대한 평가

각 방법에 대해서 test 데이터셋인 Even 05, 06와 Uneven 17, 18, 19, 20, 21에 대해서 위치에 대한 평가와 회전 방향에 대한 평가를 진행한다. 자세 예측 성능 평가 지표는 V. Peretroukhin의 연구[27]를 참고하였다.

$$\boldsymbol{e}_{\mathrm{ATE}} \triangleq \frac{1}{2}\sum_{i=1}^{N}\left\|\log_{\mathrm{SE}(3)}\left(\hat{\mathbf{T}}_{0,i}^{-1}\mathbf{T}_{0,i}\right)\right\| \tag{16}$$

$$\boldsymbol{e}_{\mathrm{RTE}} \triangleq \frac{1}{2}\sum_{i=0}^{N-k}\left\|\log_{\mathrm{SE}(3)}\left(\hat{\mathbf{T}}_{i,i+k}^{-1}\mathbf{T}_{i,i+k}\right)\right\| \tag{17}$$

$\boldsymbol{e}_{\mathrm{ATE}}\in\mathbb{R}^6$는 Absolute Trajectory Error (ATE)이다. $\boldsymbol{e}_{\mathrm{ATE}}$는 시작 지점부터 도착 지점까지의 네트워크의 출력값 $\hat{\mathbf{T}}$과 ground truth $\mathbf{T}$의 차이를 평균 낸 오차이다. $\mathbf{T}_{\mathrm{i,i+j}}\in\mathrm{SE}(3)$는 ground truth의 회전 행렬 R과 위치 p로 이루어진 i와 $i+j$ 사이에서의 변환 행렬을 의미한다. $\hat{\mathbf{T}}_{\mathrm{i,i+j}}\in\mathrm{SE}(3)$는 네트워크의 출력값 회전 행렬 $\hat{\mathrm{R}}$과 위치 $\hat{p}$로 이루어진 i와 $i+j$ 사이에서의 변환 행렬을 의미한다. 여기서 쓰인 $\log_{\mathrm{SE}(3)}$는 $\mathrm{SE}(3)$에서의 logarithm map으로 T를 $\xi\in\mathbb{R}^6$로 변환시켜준다. ξ의 첫 3개 항은 위치 $\varphi\in\mathbb{R}^3$, 나머지 3개 항은 각도 $\phi\in\mathbb{R}^3$를 의미한다.

$\boldsymbol{e}_{\mathrm{RTE}}\in\mathbb{R}^6$는 Relative Trajectory Error (RTE)이다. ATE의 경우 경로가 겹치면 오차가 줄어들 가능성이 존재한다. 이를 방지하기 위해 k 만큼의 시간 간격 동안 움직인 이동 경로의 차이를 구한 뒤 평균 내 오차를 구하는 방법이 RTE다. 고정된 시간 간격을 이용하여 상대적인 경로 오차를 확인하는 방법이다. 본 실험에서는 제작된 데이터셋을 이용했으므로 60초 간격으로 RTE를 구하기 위해 $k=6000$를 이용하였다.

3.4.1 위치 예측 평가

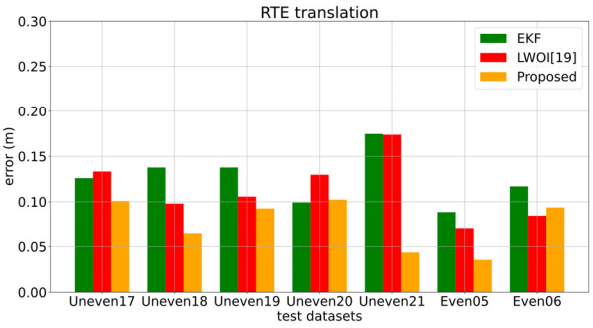
[Table 2]를 통해 test 데이터셋의 전체 평균 ATE translation과 RTE translation 값을 확인할 수 있다. LWOI와 제안하는 방법이 EKF보다 ATE와 RTE 모두 수치상으로 오차가 적은 것을 확인할 수 있고, 제안하는 방법이 LWOI보다 오차가 적다. [Fig. 4]를 통해 각 데이터셋에 대한 RTE translation 값을 확인할 수 있다.

3.4.2 회전 예측 평가

모바일 로봇이 작업을 하기 위해서는 정확한 위치를 예측하는 것 뿐만 아니라 로봇의 회전 방향도 정확히 예측해야 하므로 ATE와 RTE의 rotation 값도 성능 평가에 중요한 요소이다. [Table 3]를 통해 test 데이터셋의 전체 평균 ATE rotation과 RTE rotation 값을 확인할 수 있다. Rotation의 경우에도 translation과 같이 EKF, LWOI, 제안하는 방법 순으로 평균 오

[Table 2] Average translation error

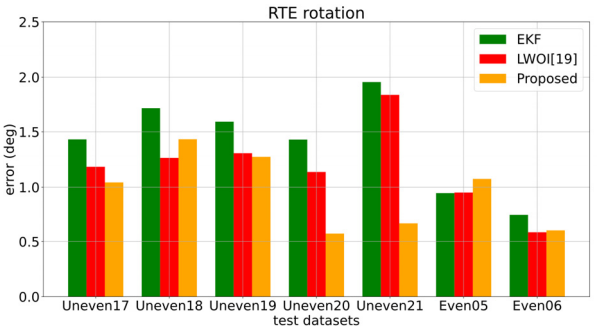
Comparison methods	ATE translation (m)	RTE translation (m)
EKF	0.112	0.126
LWOI	0.103	0.114
Proposed	**0.067**	**0.076**



[Fig. 4] Relative trajectory error translation

[Table 3] Average rotation error

Comparison methods	ATE rotation (deg)	RTE rotation (deg)
EKF	1.19	1.40
LWOI	1.00	1.18
Proposed	**0.83**	**0.95**



[Fig. 5] Relative trajectory error rotation

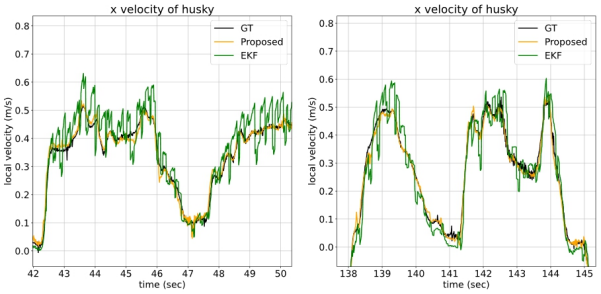
차가 더 적은 것을 확인할 수 있다. [Fig. 5]를 통해 각 데이터셋에 대한 RTE rotation 값을 확인할 수 있다.

3.4.3 자세 예측 평가

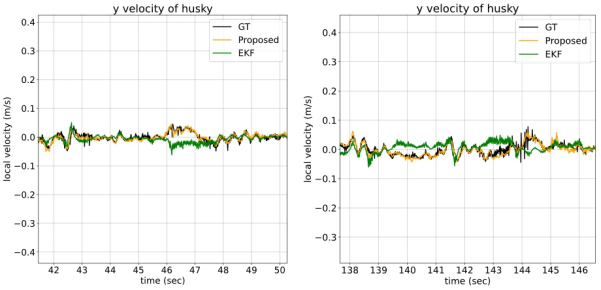
평균적인 오차를 확인했을 때 위치 예측과 회전 예측 모두 LWOI와 제안하는 방법이 EKF보다 성능이 뛰어났다. 특히 제안하는 방법은 Uneven18의 RTE rotation 예측과 Uneven20의 RTE translation 예측을 제외하고는 EKF와 LWOI보다 모든 Uneven test 데이터셋에서 위치와 회전 예측 성능이 뛰어났다. 이는 미끄러짐이 잘 발생하는 바닥에 대해서 제안하는 방식이 예측을 잘 한다고 볼 수 있다. 다만 LWOI의 경우 RTE translation에서 Uneven17과 20에서 EKF보다 성능이 떨어졌고, 제안하는 방법의 경우 RTE rotation에서 Even05에서 EKF보다 성능이 떨어졌다. 일부 test 데이터셋에서 두 학습 방법이 EKF보다 성능이 떨어진 것은 각각의 방식이 모든 상황에 대해서 학습되지 않았다는 것을 보여준다. 이는 학습에 사용된 데이터셋이 더 다양한 상황을 담고 있지 못하여 학습하지 못한 부분이 존재한다고 생각한다.

3.4.4 속도 분석

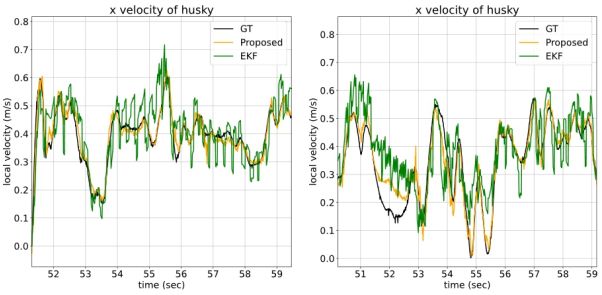
[Fig. 6]~[Fig. 9]를 통해서 미끄러짐이 얼마나 발생하는지와 속도 예측 성능을 확인할 수 있다. 모션 캡쳐 데이터와 같이 출력값이 100 Hz로 나오는 EKF와 제안하는 방식과는 다르게 LWOI의 출력값은 25 Hz로 나오기에 속도 데이터의 직접적인 비교가 힘들어 LWOI는 속도 비교에서 제외하였다. 속도 v_x와 v_y는 로봇 좌표계 기준으로 표현된 것으로, 미끄러짐이 발생하지 않았다면 속도 v_y는 0이라고 볼 수 있다. [Fig. 7]에서 속



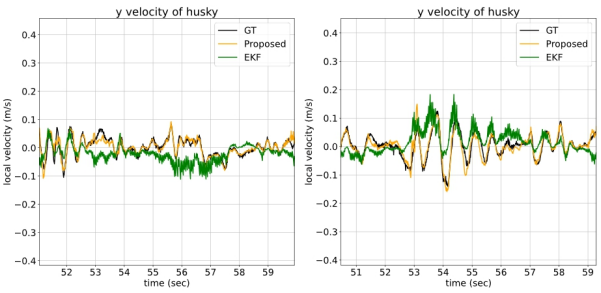
[Fig. 6] v_x in Even 05 (left) and v_x in Even 06 (right)



[Fig. 7] v_y in Even 05 (left) and v_y in Even 06 (right)



[Fig. 8] v_x in Uneven 20 (left) and v_x in Uneven 21 (right)



[Fig. 9] v_y in Uneven 20 (left) and v_y in Uneven 21 (right)

도 v_y가 0이 아니므로 평평한 바닥에서도 로봇이 옆으로 미끄러짐을 알 수 있고, [Fig. 9]를 보면 속도 v_y가 크게 변하는 것을 확인할 수 있어 울퉁불퉁한 바닥에서 미끄러짐이 크게 발생하는 것을 알 수 있다. 또한 [Fig. 6]~[Fig. 9]를 통해서 제안하는 방식과 EKF의 데이터를 ground truth와 비교하였을 때 모든 바

닥 상황에서 제안하는 방식이 EKF보다 ground truth에 가까운 것을 볼 수 있다. 이는 제안하는 방식이 미끄러짐을 효과적으로 파악하여 자세 예측 성능을 올릴 수 있었음을 의미한다.

4. 결 론

본 논문은 LSTM 학습 모델을 사용하여 모바일 로봇의 자세를 예측하는 알고리즘을 제안한다. 모바일 로봇의 바퀴 엔코더와 IMU 데이터를 입력값으로 이용하여 학습 모델을 통해 보정된 속도와 각속도를 얻을 수 있다. 이를 적분하여 자세를 예측할 수 있고 기존 방법들과 비교해서 다양한 바닥 상황에 대해서 자세 예측 오차가 적은 것을 확인할 수 있다. 특히 울퉁불퉁한 지면에서 다른 방법들에 비해 미끄러짐을 잘 파악하였다. 따라서 외부 환경에서 주행하는 모바일 로봇을 사용할 경우 제안하는 학습 모델을 이용해 자세를 예측하는 것이 효과적이라 볼 수 있다.

본 논문에서는 실험 환경 구축 한계상 두 가지 바닥 상황 데이터셋을 이용하여 학습을 진행하였지만, 추후 연구를 진행하게 된다면 모랫바닥이나 잔디 등 좀 더 다양한 바닥 환경을 구현하여 제안하는 방법의 자세 예측 성능을 확인해보고자 한다.

References

[1] J. Borenstein and L. Feng, “Measurement and correction of systematic odometry errors in mobile robots,” *IEEE Transactions on robotics and automation*, vol. 12, no. 6, pp. 869-880, Dec., 1996, DOI: 10.1109/70.544770.

[2] K. S. Chong and L. Kleeman, “Accurate odometry and error modelling for a mobile robot,” *IEEE International Conference on Robotics and Automation (ICRA)*, Albuquerque, USA, vol. 4, pp. 2783-2788, 1997, DOI: 10.1109/ROBOT.1997.606708.

[3] A. Martinelli, N. Tomatis, and R. Siegwart, “Simultaneous localization and odometry self calibration for mobile robot,” *Autonomous Robots*, vol. 22, pp. 75-85, 2007, DOI: 10.1007/s10514-006-9006-7.

[4] A. Angelova, L. Matthies, D. Helmick, and P. Perona, “Slip prediction using visual information,” *Robotics: Science and Systems*, Pennsylvania, USA, 2006, DOI: 10.15607/RSS.2006.II.014.

[5] D. M. Helmick, S. I. Roumeliotis, Y. Cheng, D. S. Clouse, M. Bajracharya, and L. H. Matthies, “Slip-compensated path following for planetary exploration rovers,” *Advanced Robotics*, vol. 20, no. 11, pp. 1257-1280, Apr., 2012, DOI: 10.1163/156855306778792470.

[6] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004-1020, Aug., 2018, DOI: 10.1109/TRO.2018.2853729.

[7] T. Qin and S. Shen, “Online temporal calibration for monocular visual-inertial systems,” *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, pp. 3662-3669, 2018, DOI: 10.1109/IROS.2018.8593603.

[8] J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time,” *Robotics: Science and Systems*, Berkeley, USA, pp. 1-9, 2014, DOI: 10.15607/RSS.2014.X.007.

[9] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping,” *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, USA, pp. 5135-5142, 2020, DOI: 10.1109/IROS45743.2020.9341176.

[10] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, “Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments,” *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, pp. 8729-8736, 2021, DOI: 10.1109/IROS51168.2021.9635862.

[11] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, “Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation,” *IEEE transactions on robotics*, vol. 25, no. 5, pp. 1087-1097, Oct., 2009, DOI: 10.1109/TRO.2009.2026506.

[12] U. Onyekpe, V. Palade, A. Herath, S. Kanarachos, and M. E. Fitzpatrick, “Whonet: Wheel odometry neural network for vehicular localisation in gnss-deprived environments,” *Engineering Applications of Artificial Intelligence*, vol. 105, pp. 104421, Oct., 2021, DOI: 10.1016/j.engappai.2021.104421.

[13] M. Brossard, A. Barrau, and S. Bonnabel, “Ai-imu dead-reckoning,” *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585-595, Dec., 2020, DOI: 10.1109/TIV.2020.2980758.

[14] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, “Orinet: Robust 3-d orientation estimation with a single particular imu,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 399-406, Apr., 2020, DOI: 10.1109/LRA.2019.2959507.

[15] M. Brossard, S. Bonnabel, and A. Barrau, “Denoising imu gyroscopes with deep learning for open-loop attitude estimation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4796-4803, Jul., 2020, DOI: 10.1109/LRA.2020.3003256.

[16] S. Herath, H. Yan, and Y. Furukawa, “Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods,” *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, pp. 3146-3152, 2020, DOI: 10.1109/ICRA40945.2020.9196860.

[17] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, “Tlio: Tight learned inertial odometry,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653-5660, Oct., 2020, DOI: 10.1109/LRA.2020.3007421.

[18] M. Zhang, M. Zhang, Y. Chen, and M. Li, “IMU data processing for inertial aided navigation: A recurrent neural network based approach,” *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi’an, China, pp. 3992-3998, 2021, DOI: 10.1109/ICRA48506.2021.9561172.

[19] M. Brossard and S. Bonnabel, "Learning wheel odometry and imu errors for localization," *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, pp. 291-297, 2019, DOI: 10.1109/ICRA.2019.8794237.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, USA, pp. 3354-3361, 2012, DOI: 10.1109/CVPR. 2012.6248074.

[21] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 642-657, Apr., 2019, DOI: 10.1177/0278364919843996.

[22] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023-1035, Dec., 2015, DOI: 10.1177/0278364915614638.

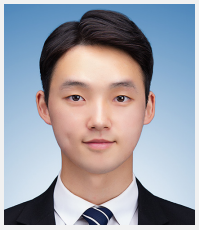
[23] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157-1163, Jan., 2016, DOI: 10.1177/0278364915620033.

[24] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, pp. 1680-1687, 2018, DOI: 10.1109/IROS.2018.8593419.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019, [Online], paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, Dec., 2014, DOI: 10.48550/arXiv.1412.6980.

[27] V. Peretroukhin and J. Kelly, "DPC-Net: Deep pose correction for visual localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2424-2431, Jul., 2018, DOI: 10.1109/LRA.2017.2778765.



김 명 수

2020 성균관대학교 기계공학부(공학사)

2022 서울대학교 지능정보융합학과 (공학석사)

관심분야: Mobile Manipulation, Planning

장 근 우

2016 고려대학교 기계공학과(공학사)

2023 서울대학교 융합과학부(공학박사)

2023~현재 한국과학기술연구원 AI로봇연구소 연구원

관심분야: Mobile Manipulator, Planning, Optimization

박 재 흥

1995 서울대학교 항공우주공학과(공학사)

1997 동 대학원 항공우주공학과(공학석사)

2006 Stanford University Aero/Astro (공학박사)

2009~현재 서울대학교 융합과학기술 대학원 교수

관심분야: Robot-environment Interaction, Multi Contact Control, Whole Body Control