공학석사 학위논문

관성측정장치를 이용한 모바일 로봇의 학습 기반 오도메트리 예측

Learning-based odometry estimation of mobile robot using IMU

2022 년 8 월

서울대학교 대학원

지능정보융합학과

김 명 수

공학석사 학위논문

관성측정장치를 이용한 모바일 로봇의 학습 기반 오도메트리 예측

Learning-based odometry estimation of mobile robot using IMU

2022 년 8 월

서울대학교 대학원

지능정보융합학과

김 명 수

관성측정장치를 이용한 모바일 로봇의 학습 기반 오도메트리 예측

Learning-based odometry estimation of mobile robot using IMU

지도 교수 박 재 흥

이 논문을 공학석사 학위논문으로 제출함
2022 년 7 월

서울대학교 대학원
지능정보융합학과
김 명 수

김명수의 공학석사 학위 논문을 인준함
2022 년 6 월

위 원 장	곽 노 준	(인)
부위원장	박 재 흥	(인)
위 원	전 동 석	(인)

초 록

모바일 로봇을 제어하여 물건을 옮기는 등의 여러 가지 작업의 성공률을 높이기 위해서는 오도메트리를 이용한 정확한 위치 추정이 필요하다. 오도메트리는 센서 정보를 기반으로 상대적인 위치를 계산하는 것을 말한다. 모바일 로봇은 보통 바퀴 속도를 수치 적분하여 위치를 계산하므로, 주행거리가 늘어날수록 실제 위치와의 오차가 커진다. 그뿐만 아니라 바퀴와 바닥의 상호작용으로 인한 로봇의 미끄러짐은 정확한 모델링이 힘들기 때문에 더 큰 오차를 유발한다. 이렇게 모델링하기 힘든 오차를 최소화하기 위해 본 논문에서는 LSTM 기반의 학습 모델을 이용하여 오도메트리를 예측하고자 한다. 바퀴 엔코더와 IMU의 데이터를 제안된 학습 모델의 입력값으로 이용하여 보정된 속도와 각속도를 출력해 이를 적분하여 회전 방향과 위치를 구한다. 실험 결과 다양한 바닥 상황에서 제안된 모델이 동일한 데이터를 사용하는 기존 방법들보다 오도메트리 예측 정확도가 더 높음을 보여준다.

주요어: 관성측정장치, 모바일 로봇, 바퀴, 오도메트리, 학습 기반

학 번: 2020-25247

목 차

표 목차

그림 목차

제 1 장 서 론

제 1 절 연구 배경

모바일 로봇의 작업 성공률을 높이기 위해서는 오도메트리를 이용한 정확한 위치 추정이 필요하다. 오도메트리는 엔코더와 같은 센서의 정보를 이용해 위치의 변화를 계산하는 것으로, 이를 누적시켜 시작 위치에서부터 현재 위치를 예측할 수 있다. 오도메트리를 이용해 위치를 예측하는 경우 센서값을 이용해 변화량을 계산하므로 이동 거리가 늘어날수록 ground truth와의 오차가 증가하게 된다.

모바일 로봇의 오도메트리 예측 시 발생할 수 있는 오차의 원인으로는 시스템적 오차와 비시스템적 오차가 있다. 시스템적 오차는 서로 다른 바퀴의 지름, 오정렬된 바퀴와 같은 기구학적 모델링과 관련돼있기에 기구학적 보정을 통해 모델링 오차를 보정할 수 있다 [1–3]. 비시스템적 오차는 로봇이 주행하는 바닥의 재질이나 상태에 따른 바퀴의 미끄러짐 등 로봇과 주행 환경 간의 상호작용으로 발생한다 [4,5]. 이를 보상하기 위해 외부 센서를 부착해 오차를 보정한다 [6–9]. 카메라와 IMU 센서를 같이 사용하여 오도메트리를 예측하기도 하고 [6,7], 라이다와 IMU 센서를 같이 사용하여 오도메트리를 예측하기도 한다 [8,9].

카메라나 라이다를 사용하여 오도메트리를 예측할 경우 정확도가 뛰어나지만, 외부가 빛이 없는 상황이거나 안개가 껴있는 등의 외부 상황에 영향을 받아 오도메트리 예측 시 어려움이 발생하기도 한다. 그렇기에 외부 상황에 따라 사용하는 센서를 변경하여 오도메트리를 예측하는 연구도 존재한다 [10]. IMU 센서 데이터를 기본으로 상황에 따라 카메라, 라이다 데이터를 추가로 활용하는 방식을 취한다.

이렇듯 카메라나 라이다를 쓰더라도 IMU의 경우 측정 주기가 다른 센서들에 비해 높기 때문에 같이 사용하는 경우가 많다. IMU는 오도메트리 예측 정확성이 다른 센서에 비해 부족할 수 있지만 외부 상황에 영향을 적게 받고 높은 측정 주기로 데이터를 처리할 수 있는 장점이 있다.

따라서 본 연구에서는 IMU 센서만을 추가로 이용하여 Long Short-Term Memory(LSTM) 네트워크를 통해 학습된 모델을 기반으로 모바일 로봇의 오도메트리 예측 성능을 향상했다.

제 2 절 연구 동향

전통적으로, 오도메트리의 정확도를 향상시키기 위해 채택된 방법은 IMU를 이용해 Extended Kalman Filter(EKF)를 사용하는 것이였다 [11]. 이는 바퀴 데이터만을 이용해 오도메트리를 예측하는 것보다는 오차가 적지만, IMU 센서에 noise나 bias 뿐만 아니라 모델링 하기 힘든 외부 환경 요인으로 인해 정확도를 향상시키는 데에 한계가 존재하였다.

최근에는 다양한 분야에서 오도메트리나 회전 방향의 예측 정확도를 높이기 위해 학습 모델을 이용하기 시작했다 [12–19]. WhONet [12]의 경우 자동차의 바퀴 엔코더 데이터를 입력값으로 ground truth 위치와의 오차를 출력값으로 내보내는 학습 모델을 이용한다. 이 경우 내재한 센서 데이터만을 사용하기에 모델링이 되지않는 오차를 파악하는 데에 한계가 있다. AI-IMU [13]는 IMU 데이터만 이용하였지만, 학습 모델을 EKF의 파라미터를 학습하는 데 사용하였다. [14, 15]은 IMU 데이터를 이용하여 회전 방향을 예측하는 데 학습 모델을 이용하였다. [16,17]의 경우 IMU 데이터를 입력값으로 이용해 ResNet 학습 모델을 통해서 바로 위치를 뽑아내는 end-to-end 학습 방법을 사용한다. 이 방법의 경우 학습 모델 내부에

위치를 계산하기 위한 적분식이 포함되게 된다. [18]에서 적분식이 학습 모델에 포함될 경우 오도메트리 예측 성능이 떨어짐을 보였다. 따라서 [18]에서는 end-to-end 방식이 아닌 입력값을 보정하는데 LSTM 학습 모델을 결합하였다. 하지만 IMU 데이터만을 이용해 오도메트리를 예측할 경우 IMU와 바퀴 엔코더 데이터를 이용한 기존의 EKF 방식보다 오도메트리 정확도가 떨어졌다. LWOI [19]는 바퀴 엔코더 데이터와 gyro, 그리고 IMU 데이터를 사용하여 오도메트리 예측을 진행한다. Gaussian Process(GP)의 파라미터를 학습시키는 데 학습 모델을 이용하였다. GP를 통과시킨 데이터를 이용해 EKF로 오도메트리를 예측한다. 이 경우 기존의 EKF보다는 오도메트리 오차가 적었지만 미끄러짐이 잘 발생하는 바닥 상황에서는 여전히 오차가 크게 나타나는 것을 볼 수 있었다.

제 3 절 연구 목표 및 방법

제안하는 방법은 카메라보다는 외부 상황에 독립적으로 데이터를 취득할 수 있는 IMU를 추가로 이용한다. 또한, LSTM을 기반으로 한 학습 모델을 통해 속도와 각속도 데이터를 구한 후 적분을 통해 오도메트리를 구하게 된다. 이때, 손실 함수를 회전 방향과 오도메트리가 누적된다는 특징을 이용하여 설계하였다. 학습된 모델은 다양한 바닥 상황에 대해서도 미끄러짐에 대한 특성을 잘 반영하여 오도메트리뿐만 아니라 회전 각도에 대해서 정확도를 향상했다.

연구 방법으로는 학습 모델에 사용될 데이터셋에 대한 설명 후, 학습 모델이 어떻게 구성돼있는지 설명한다. 그리고 실험을 통해 기존 오도메트리 예측 방식과 성능을 비교 후 결과를 분석하고 결론으로 이어진다.

제 2 장 데이터셋 제작

제 1 절 기존 데이터셋

기존에 오도메트리 학습을 위한 다양한 종류의 데이터셋이 존재한다 [20–24]. [20,21]의 경우 차량을 이용해 실외 주행 데이터를 모아두었고, [22]는 모바일 로봇을 이용해 실내외 주행 데이터를 모았다. [23]는 Micro Air Vehicle(MAV)을 이용해 실내 비행 데이터를 모아두었고, [24]는 측정 장치를 손으로 들고 다니며 실내외에서 보행 데이터를 모아두었다. 본 연구에서는 바퀴 엔코더 데이터가 필요했기 때문에 기존 데이터셋 중 차량이나 모바일 로봇 데이터셋을 쓸 수 있었다. 하지만 이 경우 ground truth로 사용하는 데이터가 Global Positioning System (GPS)이나 Simultaneous Localization and Mapping (SLAM)을 이용하여 제작했기에 정확도가 떨어진다는 단점이 있다. [23]의 경우 모션 캡처 데이터를 ground truth로 사용하였기에 높은 정확도를 가질 수 있었지만, 바퀴 엔코더가 없는 MAV를 이용했기에 사용할 수 없었다. 따라서 직접 모션 캡처 장비를 이용해 모바일 로봇의 데이터셋인 Husky 데이터셋을 제작하여 본 연구에서 사용하고자 했다. 또한 기존 주행 데이터셋들은 다양한 바닥 상황에 대한 데이터를 만들어두지 않았기에, Husky 데이터셋에서는 미끄러짐이 크게 발생하지 않는 평평한 바닥과 미끄러짐이 잘 발생하는 울퉁불퉁한 바닥 모두에서 데이터를 모아 제작하였다.

제 2 절 로봇 및 모션캡처 시스템 구축

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

제 3 절 Husky 데이터셋 생성

Husky 데이터셋은 모바일 로봇의 바퀴 속도, IMU의 각속도와 가속도, 그리고 모션캡처를 통해 측정한 평면상 위치 데이터로 이루어져 있다. Husky 로봇 위에 4개의 모션캡처 측정용 마커를 붙여 위치를 측정한 후 이를 이용해 로봇의 방향과 회전, 속도 등의 데이터를 얻어낼 수 있다.

IMU와 모션 캡처 데이터의 측정 주기는 100Hz이고 Husky 로봇의 바퀴 엔코더의 측정 주기는 25Hz이다. 학습에 사용하기 위해 바퀴 데이터를 100Hz로 보간하였다. 또한 모든 데이터의 시간을 동기화시켜 사용하였고, 데이터셋을 training, validation, testing 데이터셋으로 나눠 학습에 사용하였다.

두 가지 바닥 상황에 대해서 데이터셋을 생성하였다. 그림 2.1a과 같이 평평한 바닥에서 다양한 움직임을 담은 데이터셋과 그림 2.1b과 같이 바퀴와 바닥 사이에 미끄러짐이 잘 발생할 수 있는 환경에서 다양한 움직임을 담은 데이터셋을 만들었다. 총 27개의 데이터셋을 생성하였으며, 이에 대한 정보는 표 2.1에 표기하였다. 또한, 몇 가지 데이터셋 (Even01, Even02, Uneven01, Uneven02)의 경로를 그림

(a) Even ground environment (b) Uneven ground environment

그림 2.1: Various ground environments

2.2에 나타내었다.

표 2.1: Information of Dataset

Dataset			
Name	Path length (m)	Duration (s)	Usage
Even01	31.60	111.86	Training
Even02	34.12	134.63	Training
Even03	79.01	300.28	Training
Even04	76.70	303.62	Training
Even05	25.15	83.21	Testing
Even06	43.52	167.72	Testing
Uneven01	21.42	70.89	Training
Uneven02	25.69	88.04	Training
Uneven03	25.00	86.57	Training
Uneven04	30.04	102.03	Training
Uneven05	25.93	87.96	Training
Uneven06	26.60	89.95	Training
Uneven07	28.72	100.94	Training
Uneven08	25.91	91.24	Training
Uneven09	19.33	75.52	Training
Uneven10	37.41	115.48	Training
Uneven11	31.26	106.11	Training
Uneven12	39.92	111.21	Training
Uneven13	41.07	120.70	Training
Uneven14	38.68	115.34	Validation
Uneven15	25.91	91.24	Validation
Uneven16	32.09	101.77	Validation
Uneven17	24.34	83.11	Testing
Uneven18	37.45	112.41	Testing
Uneven19	43.20	121.07	Testing
Uneven20	35.03	109.44	Testing
Uneven21	42.20	117.24	Testing

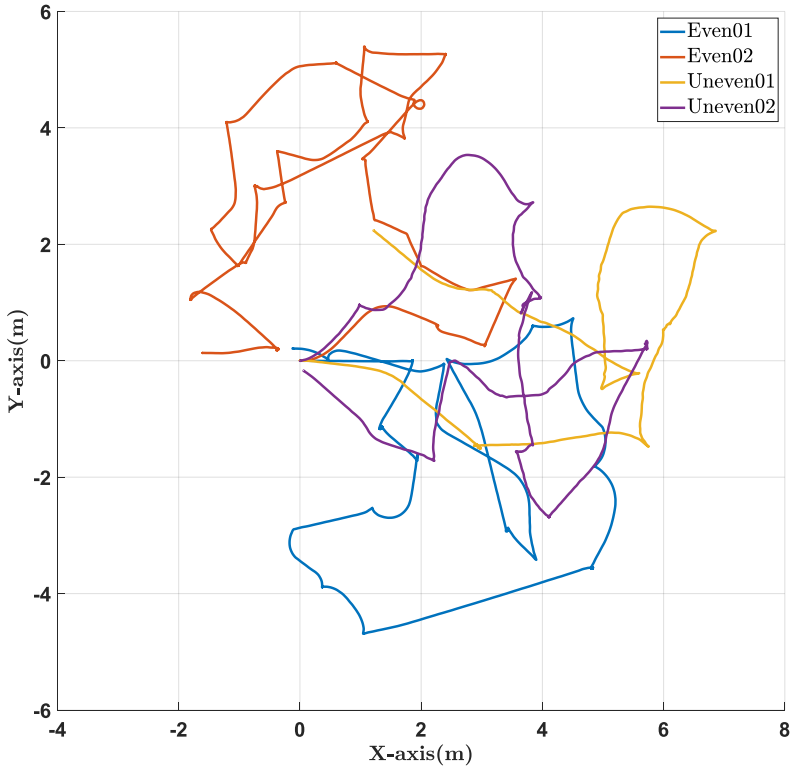


그림 2.2: Trajectory of dataset (Even01, Even02, Uneven01, Uneven02)

제 3 장 학습 방법

제 1 절 네트워크 구조

제안하는 네트워크는 stacked unidirectional LSTM을 사용한다. LSTM의 경우 레이어는 3개, 각 레이어 당 크기는 120으로 돼 있다. 네트워크를 이용한 오도메트리를 예측하는 전체적인 과정은 그림 3.1에 나와 있다. 입력값으로 바퀴 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. 이를 적분을 통해 위치 변화값을 구하고, 값을 누적시켜 오도메트리 예측하게 된다.

$$\mathrm{R}_n = \mathrm{R}_{n-1} exp(\omega_n dt) \tag{3.1}$$

$$\hat{\mathrm{R}}_n = \hat{\mathrm{R}}_{n-1} exp(\hat{\omega}_n dt) \tag{3.2}$$

식 3.1은 ground truth 각속도 ω를 이용해 회전 행렬 R을 구하는 식이다. exp는 exponential map으로 3자유도 각도값을 회전 행렬로 변환시켜준다. 식 3.2는 네트워크의 출력값 각속도 $\hat{\omega}$를 이용해 회전 행렬 $\hat{\mathrm{R}}$를 구하는 식이다.

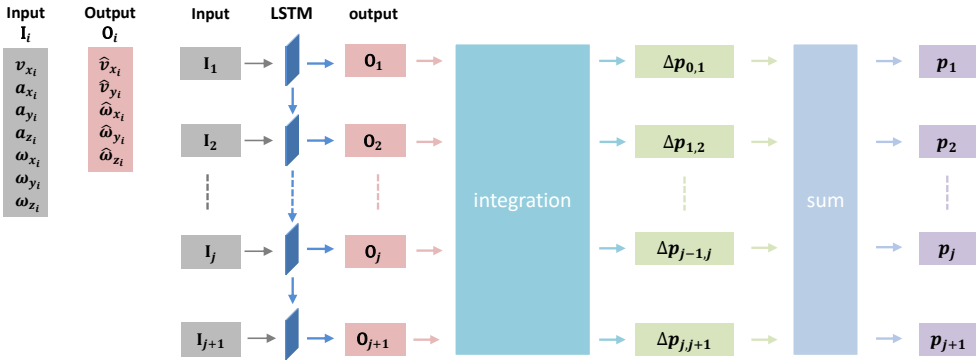


그림 3.1: Proposed architecture

$$p_n = p_{n-1} + \mathrm{R}_{n-1}(v_{n-1}dt) \tag{3.3}$$

$$\hat{p}_n = \hat{p}_{n-1} + \hat{\mathrm{R}}_{n-1}(\hat{v}_{n-1}dt) \tag{3.4}$$

식 3.3에서는 ground truth의 회전 행렬 R과 속도 v를 이용하여 적분을 통해서 위치 p를 구하는 식이다. 식 3.4는 네트워크 출력값의 회전 행렬 $\hat{\mathrm{R}}$과 속도 $\hat{v}$를 이용하여 적분을 통해서 위치 $\hat{p}$를 구하는 식이다.

제 2 절 손실 함수

$$\mathcal{L} = k * \mathcal{L}_w + \mathcal{L}_p \tag{3.5}$$

손실 함수를 어떻게 만드느냐에 따라 학습 모델의 성능이 달라질 수 있다. 학습 모델의 출력으로 로봇의 속도 $\hat{v}_x, \hat{v}_y$, 로봇의 각속도 $\hat{w}_x, \hat{w}_y, \hat{w}_z$를 뽑아내므로 각각에 대해서 손실 함수를 만들 수 있다. 손실 함수 $\mathcal{L}$는 식 3.5와 같이 회전 손실

함수 $\mathcal{L}_w$와 위치 손실 함수 $\mathcal{L}_p$를 더해서 사용한다. 각속도를 이용해 회전 손실 함수 $\mathcal{L}_w$를 구하고, 속도를 이용해 위치 손실 함수 $\mathcal{L}_p$를 구한 후 합하여 사용한다. 회전 손실 함수 $\mathcal{L}_w$에 k를 곱해 사용하는 이유는 회전 손실 함수 $\mathcal{L}_w$와 위치 손실 함수 $\mathcal{L}_p$의 크기가 다르기 때문에 학습 시에 한 손실 함수에 편향되는 것을 막기 위해서이다. 본 논문에서는 $k = 1500$을 이용하였다.

$$\rho(a) = \begin{cases} \frac{1}{2}a^2, & if\ |a| < \delta \\ \delta * (|a| - \frac{1}{2} * \delta), & otherwise \end{cases} \tag{3.6}$$

손실을 계산할 때는 Huber loss function인 식 3.6를 이용하였다.

$$\delta \mathrm{R}_{i,i+j} = \mathrm{R}_i^T \mathrm{R}_{i+j} \tag{3.7}$$

$$\delta \hat{\mathrm{R}}_{i,i+j} = \hat{\mathrm{R}}_i^T \hat{\mathrm{R}}_{i+j} \tag{3.8}$$

식 3.7과 식 3.8은 각각 ground truth와 네트워크의 출력값으로 구한 j간격 동안 회전한 회전 행렬값들이다.

$$\mathcal{L}_{w,j} = \sum_{i=0}^{N/j} \rho(log(\delta \hat{\mathrm{R}}^T_{j\times i,j\times i+j} \delta \mathrm{R}_{j\times i,j\times i+j})) \tag{3.9}$$

$$\mathcal{L}_{w,1} = \sum_{i=0}^{N} \rho(log(\delta \hat{\mathrm{R}}^T_{i,i+1} \delta \mathrm{R}_{i,i+1})) \tag{3.10}$$

$$\mathcal{L}_{w,16} = \sum_{i=0}^{N/16} \rho(log(\delta \hat{\mathrm{R}}^T_{16i,16i+16} \delta \mathrm{R}_{16i,16i+16})) \tag{3.11}$$

회전 행렬의 경우 시간이 지남에 따라 누적할 수 있는 특징이 있다. 따라서 식 3.9는 N개의 데이터가 있을 때 j개의 간격만큼 회전 행렬을 누적시킨 후의 $\delta\hat{\mathrm{R}}$과 $\delta\mathrm{R}$을 이용하여 손실을 계산하는 식이다. log는 SO(3)에서의 logarithm map으로 회전 행렬 R을 각도 $\phi \in \mathbb{R}^3$로 변환시켜준다.

$$\mathcal{L}_w = \mathcal{L}_{w,1} + \mathcal{L}_{w,2} + \mathcal{L}_{w,4} + \mathcal{L}_{w,8} + \mathcal{L}_{w,16} \tag{3.12}$$

최종적인 회전 손실 함수 $\mathcal{L}_w$는 식 3.10이다. 식 3.10처럼 j가 작았을 때는 각 데이터의 회전 행렬이 ground truth 값을 잘 따라갔지만, 누적시켰을 경우 오차가 크게 발생하였다. 반대로 식 3.11처럼 j가 클 때 누적시켰을 경우 오차가 크진 않았지만 각 데이터의 회전 행렬값에서 오차가 발생하였다. 따라서 식 3.12과 같이 j가 작을 때와 클 때 모두를 이용하여 손실 함수를 사용할 때 각 데이터와 누적 데이터의 오차가 적었다.

$$\acute{p}_n = \acute{p}_{n-1} + \mathrm{R}_{n-1}(\hat{v}_{n-1}dt) \tag{3.13}$$

식 3.13는 위치 $\acute{p}$를 구하는 식이다. 위치 $\acute{p}$를 구하기 위한 속도는 출력값 $\hat{v}$을 이용하지만, 회전 행렬은 ground truth R을 이용한다는 특징이 있다. 학습 시에 식 3.4의 $\hat{p}$는 출력값의 속도와 회전 행렬 모두에 영향을 받지만, $\acute{p}$는 출력값의 속도에 대해서만 영향을 받는다. 또한 $\hat{\mathrm{R}}_n$은 출력값을 누적시키는 계산이 필요하지만, ground truth R_n은 이미 알고 있는 데이터이기에 위치를 구할 때 계산 시간이 줄어드는 장점이 있다. 따라서 위치 손실 함수 $\mathcal{L}_p$에서 출력값의 속도에 대해서만 손실을 계산하기 위해 $\hat{p}$ 대신 $\acute{p}$를 사용한다.

$$\delta p_{i,i+j} = p_{i+j} - p_i \tag{3.14}$$

$$\delta \acute{p}_{i,i+j} = \acute{p}_{i+j} - \acute{p}_i \tag{3.15}$$

식 3.14와 식 3.15는 각각 p와 $\acute{p}$를 이용해 구한 j개의 간격동안 이동한 위치 변화값들이다.

$$\mathcal{L}_{p,j} = \sum_{i=0}^{N/j} \rho(\delta \acute{p}_{j\times i,j\times i+j} - \delta p_{j\times i,j\times i+j}) \tag{3.16}$$

$$\mathcal{L}_{p,1} = \sum_{i=0}^{N} \rho(\delta \acute{p}_{i,i+1} - \delta p_{i,i+1}) \tag{3.17}$$

$$\mathcal{L}_{p,4} = \sum_{i=0}^{N/4} \rho(\delta \acute{p}_{4i,4i+4} - \delta p_{4i,4i+4}) \tag{3.18}$$

위치의 경우 회전 행렬과 같이 시간이 지남에 따라 누적할 수 있다. 따라서 식 3.16는 N개의 데이터가 있을 때 j개의 간격만큼 위치를 누적시킨 후의 $\delta \acute{p}$과 δp을 이용하여 손실을 계산하는 식이다.

$$\mathcal{L}_p = \mathcal{L}_{p,1} + \mathcal{L}_{p,2} + \mathcal{L}_{p,4} \tag{3.19}$$

최종적인 위치 손실 함수 $\mathcal{L}_p$는 식 3.19이다. 위치 손실 함수 $\mathcal{L}_p$도 회전 손실 함수 $\mathcal{L}_w$과 같이 하나의 j만을 이용하지 않고 여러 개를 사용하였다. 식 3.17과 같이 j가 작을 경우 각 데이터의 속도값은 오차가 작았지만, 속도값을 이용해 위치를 구했을 경우엔 오차가 크게 발생하였다. 반대로 식 3.18와 같이 j가 클 경우

위칫값은 오차가 작았지만, 각 데이터의 속도값은 오차가 상대적으로 컸다. 따라서 식 3.19와 같이 j를 작을 때와 클 때 모두 이용해 손실 함수를 만들어 각 데이터의 속도와 위치 데이터의 오차를 줄였다.

제 4 장 오도메트리 예측 실험

제 1 절 학습 모델 파라미터 설정

PyTorch를 통해 학습을 진행하였다. Optimizer는 Adam을 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.

학습의 성능을 높이기 위해 training 데이터셋을 8초 시간 간격으로 무작위로 추출하여 섞은 후 학습에 이용하였다. 또한 효율적으로 학습하기 위해 batch 크기를 64로 두고 학습을 진행하였다.

제 2 절 비교 대상

제안하는 방식과 오도메트리 예측 성능을 비교할 대상으로 EKF를 이용한 방법과 기존 학습 논문 LWOI [19]를 사용하였다. Husky 데이터셋을 이용하여 성능 비교를 하였고, 제안하는 방식과 비교 대상들 모두 바퀴 데이터와 IMU의 각속도, 가속도 데이터를 이용하였다. LWOI [19]의 경우 자이로 센서를 추가로 이용하지만, 성능 비교를 위해 IMU의 각속도 데이터를 이용해 자이로 데이터(w_x, w_y, w_z)를 대체하여 사용했다.

제 3 절 실험

각 방법에 대해서 test 데이터셋인 Even05, 06와 Uneven17, 18, 19, 20, 21에 대해서 위치에 대한 평가와 회전 방향에 대한 평가를 진행한다. 오도메트리 예측 성능 평가 지표는 [25]를 참고하였다. $\mathbf{T}_{i,i+j} \in \mathrm{SE}(3)$는 ground truth의 회전 행렬 R과 위치 p로 이루어진 i와 $i+j$ 사이에서의 변환 행렬을 의미한다. $\hat{\mathbf{T}}_{i,i+j} \in \mathrm{SE}(3)$는 네트워크의 출력값 회전 행렬 $\hat{\mathrm{R}}$과 위치 $\hat{p}$로 이루어진 i와 $i+j$ 사이에서의 변환 행렬을 의미한다.

$$\mathbf{e}_{ATE} \triangleq \frac{1}{2}\sum_{i=1}^{N} \|log(\hat{\mathbf{T}}_{0,i}^{-1}\mathbf{T}_{0,i})\| \tag{4.1}$$

식 4.1는 Absolute Trajectory Error (ATE) $\mathbf{e}_{ATE} \in \mathbb{R}^6$를 구하기 위한 식이다. $\mathbf{e}_{ATE}$는 시작 지점부터 도착 지점까지의 네트워크의 출력값 $\hat{\mathbf{T}}$과 ground truth $\mathbf{T}$의 차이를 평균 낸 오차이다. 여기서 쓰인 log는 SE(3)에서의 logarithm map으로 $\mathbf{T}$를 $\xi \in \mathbb{R}^6$로 변환시켜준다. ξ의 첫 3개 항은 위치 $\rho \in \mathbb{R}^3$, 나머지 3개 항은 각도 ϕ를 의미한다.

$$\mathbf{e}_{RTE} \triangleq \frac{1}{2}\sum_{i=0}^{N-k} \|log(\hat{\mathbf{T}}_{i,i+k}^{-1}\mathbf{T}_{i,i+k})\| \tag{4.2}$$

식 4.2는 Relative Trajectory Error (RTE) $\mathbf{e}_{RTE} \in \mathbb{R}^6$를 구하기 위한 식이다. 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 Husky 데이터셋을 이용했으므로 60초 간격으로 RTE를 구하기 위해 $k = 6000$를 이용하였다.

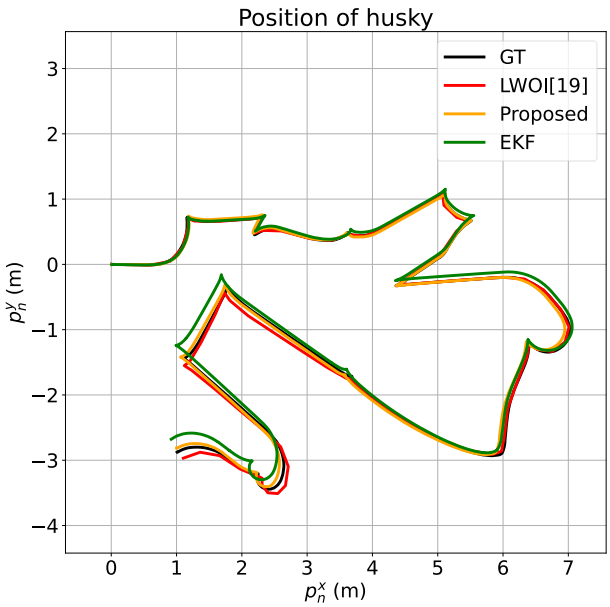


그림 4.1: Even05 position graph

그림 4.1는 Even05에 대한 위치 그래프이다. 표 4.1를 보면 ATE translation과 RTE translation의 경우 제안한 방법이 0.021 m과 0.036 m로 오차 가장 적었다. ATE rotation과 RTE rotation의 경우 LWOI [19]가 0.77 deg와 0.69 deg로 오차가 가장 적었다.

표 4.1: Even05 ATE & RTE

Comparison methods	EKF	LWOI [19]	Proposed
ATE translation (m)	0.045	0.032	**0.021**
RTE translation (m)	0.089	0.048	**0.036**
ATE rotation (deg)	0.78	**0.77**	0.080
RTE rotation (deg)	0.95	**0.69**	1.09

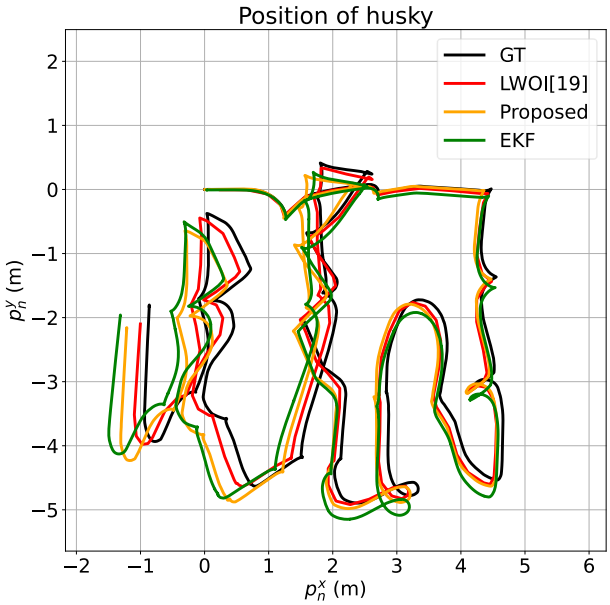


그림 4.2: Even06 position graph

그림 4.2는 Even06에 대한 위치 그래프이다. 표 4.2를 보면 ATE translation과 RTE translation의 경우 LWOI [19]가 0.084 m과 0.056 m로 오차 가장 적었다. ATE rotation의 경우 EKF가 0.96 deg로 오차가 가장 적었고, RTE rotation의 경우 제안하는 방법이 0.60 deg로 오차가 가장 적었다.

표 4.2: Even06 ATE & RTE

Comparison methods	EKF	LWOI [19]	Proposed
ATE translation (m)	0.132	**0.084**	0.117
RTE translation (m)	0.117	**0.056**	0.094
ATE rotation (deg)	**0.96**	0.98	1.03
RTE rotation (deg)	0.75	0.61	**0.60**

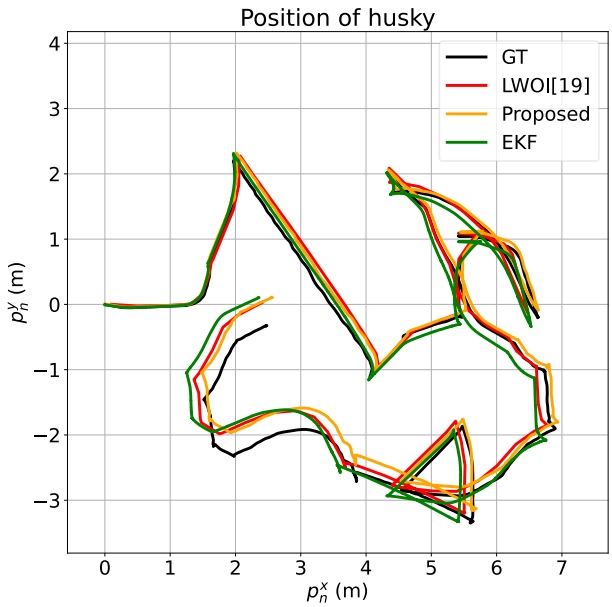


그림 4.3: Uneven20 position graph

그림 4.3는 Uneven20에 대한 위치 그래프이다. 표 4.3를 보면 ATE translation과 RTE translation 모두 제안한 방법이 0.065 m과 0.093 m로 오차가 가장 적었다. ATE rotation과 RTE rotation의 경우도 제안하는 방법이 0.91 deg와 1.27 deg로 오차가 가장 적었다.

표 4.3: Unven20 ATE & RTE

Comparison methods	EKF	LWOI [19]	Proposed
ATE translation (m)	0.077	0.096	**0.065**
RTE translation (m)	0.138	0.123	**0.093**
ATE rotation (deg)	1.51	1.16	**0.91**
RTE rotation (deg)	1.59	1.34	**1.27**

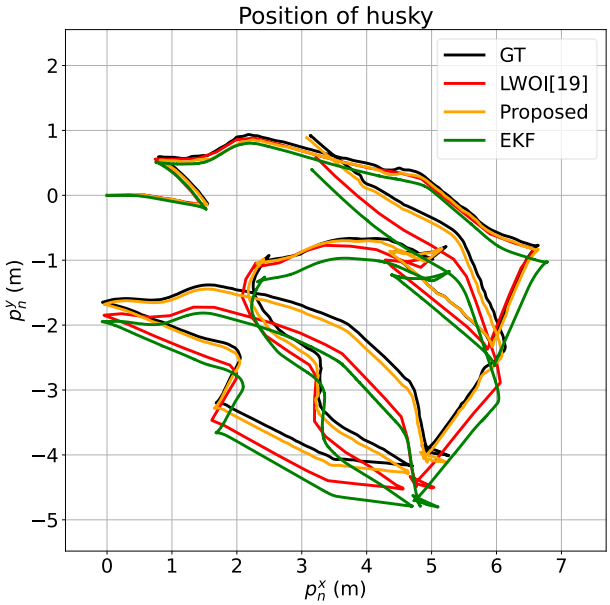


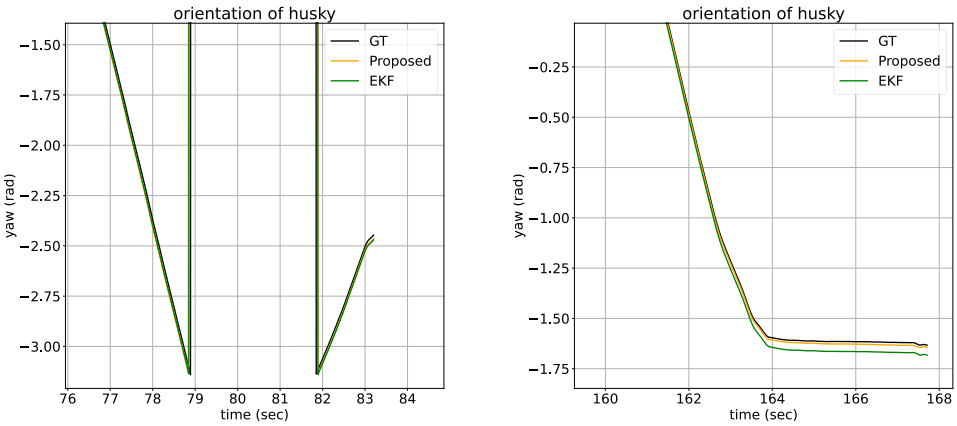
그림 4.4: Uneven21 position graph

그림 4.4는 Uneven21에 대한 위치 그래프이다. 표 4.4를 보면 ATE tranlsation과 RTE translation 모두 제안한 방법이 0.035 m과 0.044 m로 오차 가장 적었다. ATE rotation과 RTE rotation의 경우도 제안하는 방법이 0.50 deg와 0.67 deg로 오차가 가장 적었다.

표 4.4: Unven21 ATE & RTE

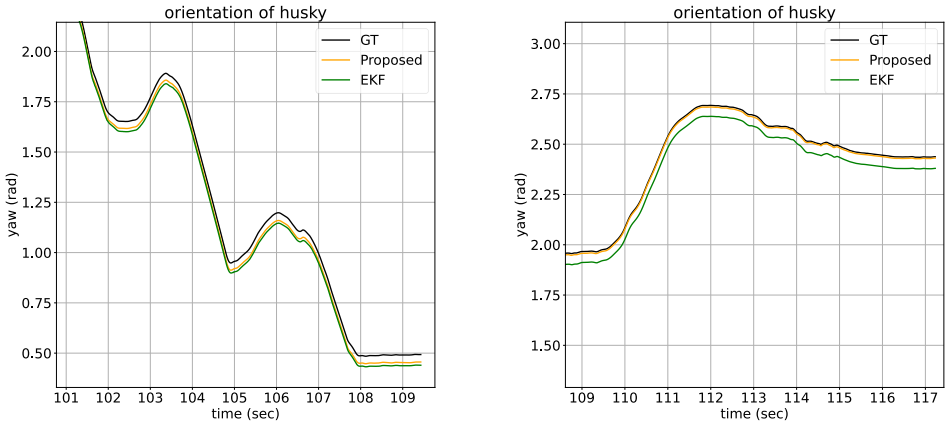
Comparison methods	EKF	LWOI [19]	Proposed
ATE translation (m)	0.193	0.170	**0.035**
RTE translation (m)	0.175	0.179	**0.044**
ATE rotation (deg)	1.56	1.40	**0.50**
RTE rotation (deg)	1.95	2.17	**0.67**

이러한 결과를 통해, 제안하는 방법은 평평한 지면에서는 LWOI [19]과 비슷한 성능을 보였지만, 울퉁불퉁한 지면에서는 제안하는 방법이 LWOI [19]보다 성능이 훨씬 좋은 것을 볼 수 있다.



(a) Even05 (b) Even06

그림 4.5: Even ground orientation graphs



(a) Uneven20 (b) Uneven21

그림 4.6: Uneven ground orientation graphs

그림 4.5과 4.6를 통해 ground truth, EKF, 제안하는 방법의 yaw 회전 누적값을 확인할 수 있다. LWOI [19]의 경우 출력값의 데이터 크기가 작아 직접적인 비교는 힘들어 그래프에서는 제외하였다. 지면의 상태와 상관없이 제안하는 방법이 EKF보다 ground truth에 가까운 것을 알 수 있다.

제 4 절 결과

1) 위치 예측 평가

표 4.5: Average translation error

Comparison methods	ATE translation (m)	RTE translation (m)
EKF	0.117	0.126
LWOI [19]	0.101	0.112
Proposed	**0.067**	**0.076**

표 4.5를 통해 test 데이터셋의 전체 평균 ATE translation과 RTE translation 값을 확인할 수 있다. LWOI [19]와 제안하는 방법이 EKF보다 ATE와 RTE 모두 수치상으로 오차가 적은 것을 확인할 수 있고, 제안하는 방법이 LWOI [19]보다 오차가 적다. 그림 4.7를 통해 각 데이터셋에 대한 RTE translation 값을 확인할 수 있다. Even06에서 제안하는 방법보다 LWOI [19]의 오차가 적게 나왔지만, Uneven17, 20, 21에서 LWOI [19]는 EKF보다 오차가 크게 발생하였다. 평균적인 오차는 LWOI [19]가 EKF보다 작았지만, 울퉁불퉁한 바닥 데이터들에서는 오차가 큰 것으로 보아 바닥 미끄러짐을 효과적으로 감지해낸다고 볼 수 없다. 반면에 제안하는 방법은 평평한 바닥과 울퉁불퉁한 바닥 모두 골고루 EKF에 비해 오도메트리 예측 성능이 향상된 것을 알 수 있다.

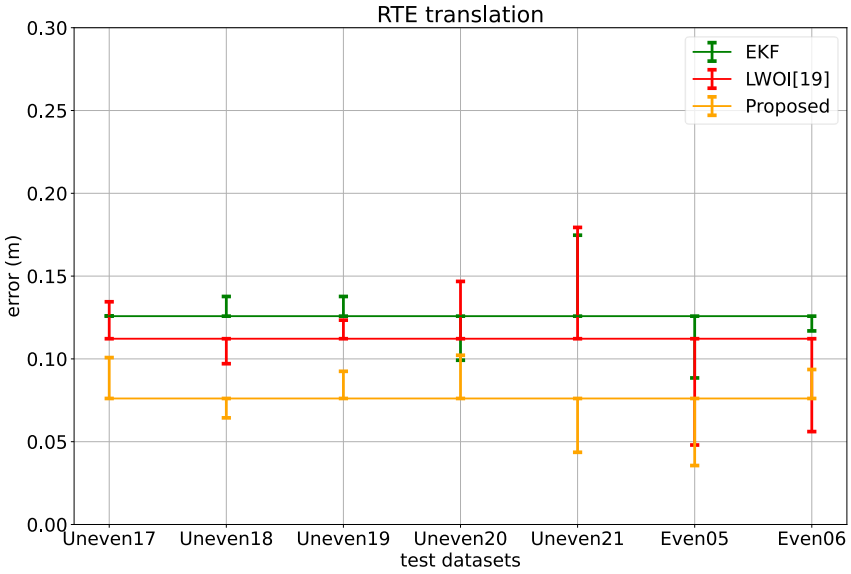


그림 4.7: RTE translation

2) 회전 방향 예측 평가

표 4.6: Average rotation error

Comparison methods	ATE rotation (deg)	RTE rotation (deg)
EKF	1.19	1.40
LWOI [19]	1.01	1.23
Proposed	**0.83**	**0.95**

표 4.6를 통해 test 데이터셋의 전체 평균 ATE rotation과 RTE rotation 값을 확인할 수 있다. 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. Rotation의 경우에도 translation과 같이 EKF,

LWOI [19], 제안하는 방법 순으로 평균 오차가 적은 것을 확인할 수 있다. 그림 4.8를 통해 각 데이터셋에 대한 RTE rotation 값을 확인할 수 있다. Translation 과는 다르게 LWOI [19]가 EKF에 비해 전체적으로 오차가 적었다. Even05에서는 LWOI [19]가 제안하는 방법보다 오차가 적었지만 다른 test 데이터셋에서는 제안하는 방법의 오차가 LWOI [19]보다 더 적었다.

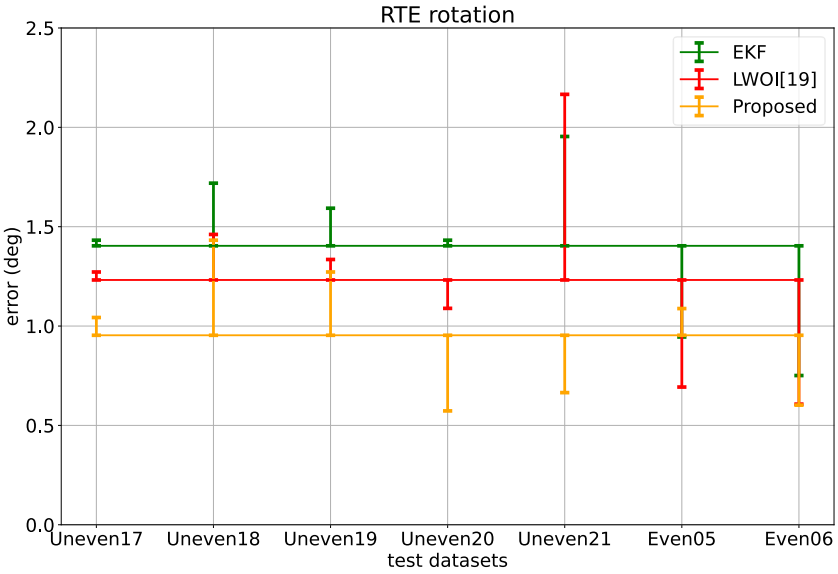
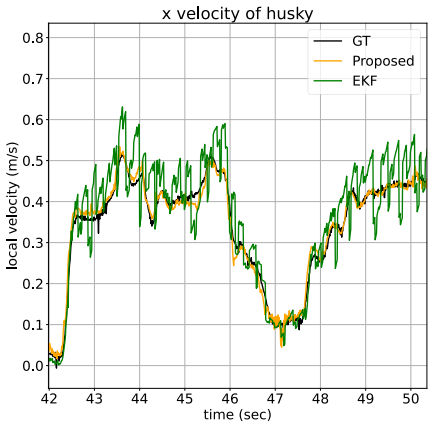


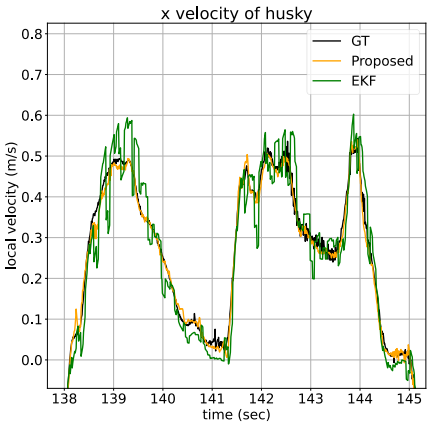
그림 4.8: RTE rotation

3) 속도 데이터 분석

그림 4.9, 4.10, 4.11, 4.12를 통해서 미끄러짐이 얼마나 발생하는지와 속도 예측 성능을 확인할 수 있다. 여기서도 LWOI [19]의 출력값 데이터의 크기가 작아 직접 적인 비교가 힘들어 LWOI [19]는 제외하였다. 그림 4.10에서 속도 v_y가 0이 아니므로 평평한 바닥에서도 로봇이 옆으로 미끄러짐을 알 수 있고, 그림 4.12

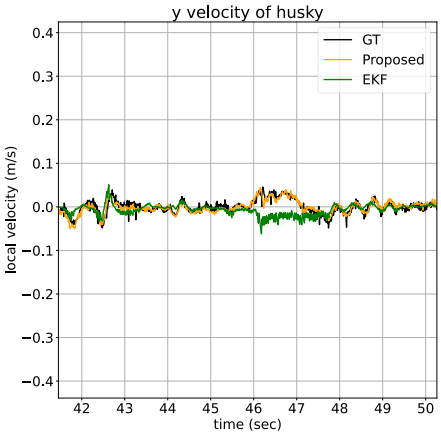


(a) Even05

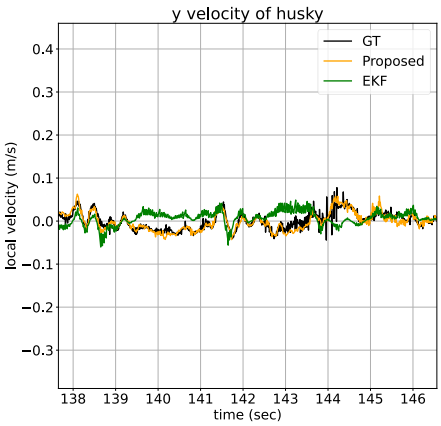


(b) Even06

그림 4.9: Even ground x velocity graphs

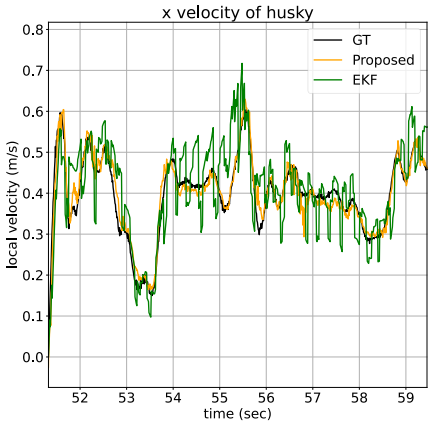


(a) Even05

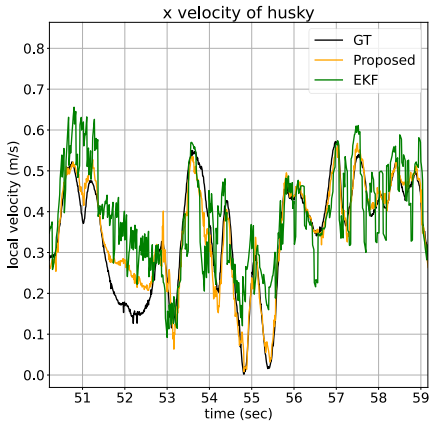


(b) Even06

그림 4.10: Even ground y velocity graphs

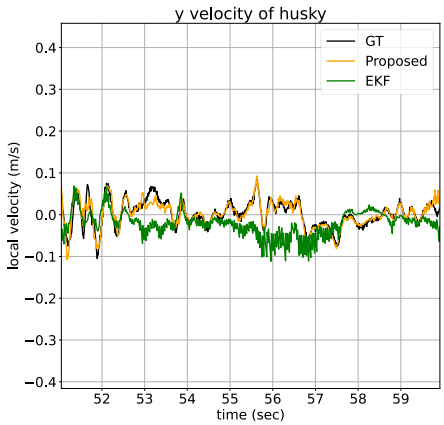


(a) Uneven20

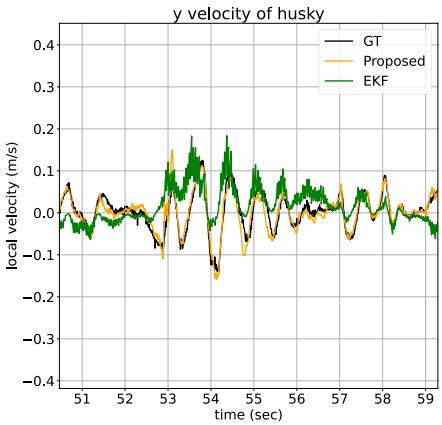


(b) Uneven21

그림 4.11: Uneven ground x velocity graphs



(a) Uneven20



(b) Uneven21

그림 4.12: Uneven ground y velocity graphs

를 보면 속도 v_y가 크게 변하는 것을 확인할 수 있어 울퉁불퉁한 바닥에서 미끄러짐이 크게 발생하는 것을 알 수 있다. 또한 그림 4.9, 4.10, 4.11, 4.12을 통해서 제안하는 방식과 EKF의 데이터를 ground truth와 비교하였을 때 모든 바닥 상황에서 제안하는 방식이 EKF보다 ground truth에 가까운 것을 볼 수 있다. 이는 제안하는 방식이 미끄러짐을 효과적으로 파악하여 오도메트리 예측 성능을 올릴 수 있었음을 의미한다.

제 5 장 결　　론

본 논문은 LSTM 학습 모델을 사용하여 모바일 로봇의 오도메트리를 예측하는 알고리즘을 제안한다. 모바일 로봇의 바퀴 데이터와 IMU 데이터를 입력값으로 이용하여 학습 모델을 통해 보정된 속도와 각속도를 얻을 수 있다. 이를 적분하여 오도메트리를 예측할 수 있고 기존 방법들과 비교해서 다양한 바닥 상황에 대해서 오도메트리 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 오도메트리를 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 오도메트리 예측 성능을 확인해보고자 한다.

참고 문헌

[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, 1996.

[2] K. S. Chong and L. Kleeman, "Accurate odometry and error modelling for a mobile robot," in *Proceedings of International Conference on Robotics and Automation*, vol. 4, pp. 2783–2788, IEEE, 1997.

[3] A. Martinelli, N. Tomatis, and R. Siegwart, "Simultaneous localization and odometry self calibration for mobile robot," *Autonomous Robots*, vol. 22, no. 1, pp. 75–85, 2007.

[4] A. Angelova, L. Matthies, D. Helmick, and P. Perona, "Slip prediction using visual information," *Robotics : Science and Systems*, 2007.

[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, 2006.

[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, 2018.

[7] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3662–3669, IEEE, 2018.

[8] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," *Robotics: Science and Systems*, vol. 2, no. 9, pp. 1–9, 2014.

[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," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5135–5142, IEEE, 2020.

[10] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8729–8736, IEEE, 2021.

[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, 2009.

[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, p. 104421, 2021.

[13] M. Brossard, A. Barrau, and S. Bonnabel, "Ai-imu dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[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, 2019.

[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, 2020.

[16] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3146–3152, IEEE, 2020.

[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, 2020.

[18] M. Zhang, M. Zhang, Y. Chen, and M. Li, "Imu data processing for inertial aided navigation: A recurrent neural network based approach," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3992–3998, IEEE, 2021.

[19] M. Brossard and S. Bonnabel, "Learning wheel odometry and imu errors for localization," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 291–297, IEEE, 2019.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361, IEEE, 2012.

[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, 2019.

[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, 2016.

[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, 2016.

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

[25] 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, 2017.

Abstract

Learning-based odometry estimation of mobile robot using IMU

Myeongsoo Kim

Department of Intelligence and Information

The Graduate School of Convergence Science and Technology

Seoul National University

In order to control the mobile robot and perform various tasks such as moving objects with the robot, position estimation using odometry is required. Odometry calculates a relative position based on sensor information. The mobile robot usually calculates the position by integration of the wheel speed, so as the driving distance increases, the error with the actual position increases. In addition, the slipping of the robot due to the interaction between the wheels and the floor causes a larger error because accurate modeling is difficult. To minimize the error that is difficult to modeling, this paper introduces a Long Short-Term Memory(LSTM)-based learning model to predict odometry. The proposed learning model uses the data of the wheel encoder and Inertial Measurement Unit(IMU) as input values. The outputs of the model are the corrected

speed and angular speed. The rotation direction and position are obtained by integrating them. Experimental results demonstrate that the proposed learning model has higher accuracy of odometry estimation than the existing methods using the same data under various ground conditions.

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

Student Number: 2020-25247