工學碩士 學位論文

9-Axis IMU와 뉴럴 네트워크 기반 Modified IONet을 이용한 실시간 3차원 위치추정에 관한 연구

A Study on real-time 3D Position Estimation based on 9-Axis IMU and Neural Network using Modified IONet

指導敎授 徐 東 煥

2020年 8月

韓國海洋大學校 大學院

電氣電子工學科

徐 弘 一

本 論文을 徐弘一 工學碩士 學位論文으로 認准함

委員長 : 工學博士 金 載 熏 ㊞

委 員 : 工學博士 徐 東 煥 ㊞

委 員 : 工學博士 朱 良 翊 ㊞

2020年 **7**月

韓國海洋大學校 大學院

목 차

그림 및 표 목차

<그림 목차>

<표 목차>

A Study on real-time 3D Position Estimation based on 9-Axis IMU and Neural Network Using Modified IONet

by Hong-Il, Seo

Department of Electrical & Electronics Engineering
Graduate School of Korea Maritime & Ocean University
Busan, Republic of Korea

Abstract

There are several studies to use sensor data measured in mobile devices as the efficient deep learning framework that can be applied to embedded systems has been developed. Especially, study on location estimation is being focused using acceleration, gyroscope, and geomagnetic embedded in a smartphone. In this paper, we propose three Modified Inertial Neural Networks (Modified IONet) based on IMU and neural network. The proposed network, 9-Axis IONet, corrected the inertial system drift by adding gravity acceleration and

geomagnetic, which are data including attitude information, to the 6-Axis IONet that estimates trajectory only with free acceleration and gyro. In addition, this paper presents a fast 6-Axis Ionet and fast 9-Axis Ionet, which are replaced by GRUs that can reduce the amount of operation by about 30% lower than LSTM, and analyzes the effect of parameter numbers on the trajectory estimation performance, thereby suggesting a method to apply IONet to embedded and mobile environments. The proposed network has improved the trajectory estimation performance by 9.08% compared to the 6-Axis IONEt by learning and evaluating OxIOD, which is a data set constructed using low-cost IMU.

9-Axis IMU와 뉴럴 네트워크 기반
Modified IONet을 이용한 실시간 3차원 위치추정에 관한 연구

서 홍 일

대한민국, 부산
한국해양대학교 대학원
전기전자공학과

초록

최근 임베디드 시스템에 적용될 수 있는 효율적인 딥러닝 프레임워크가 개발됨에 따라 모바일 기기에서 측정되는 센서 데이터를 사용하기 위한 연구가 활발히 진행 중이다. 특히 스마트폰에 내장되어 있는 가속도 센서, 자이로 센서, 지자기 센서를 이용한 위치추정에 대한 연구가 집중적으로 이루어지고 있다. 따라서 본 논문에서는 IMU와 뉴럴 네트워크 기반 3가지 Modified Inertial Neural Network(Modified IONet)를 제안한다. 제안하는 네트워크인 9-Axis IONet는 자유가속도와 자이로만으로 궤적을 추정하는 6-Axis IONet

에 자세정보가 포함된 데이터인 중력가속도와 지자기를 추가하여 Inertial System Drift을 보정하였다. 또한 LSTM에 비해 파라미터 수가 약 30% 낮아 연산량을 감소시킬 수 있는 GRU로 대체된 fast 6-Axis IONet과 fast 9-Axis IONet을 제시하여 파라미터 수가 궤적 추정성능에 미치는 영향을 분석함으로써 임베디드 및 모바일 환경에 IONet을 적용할 수 있는 방법을 제시하였다. 제안하는 네트워크는 저가형 IMU를 이용하여 구축된 데이터셋인 OxIOD를 학습 및 평가를 진행하여 6-Axis IONet에 비해 궤적추정 성능이 9.08% 향상되었다.

제 1 장 Introduction

미국의 Global Positioning System(GPS), 일본의 Quasi-Zenith Satellite System(QZSS), EU의 Galileo Positioning System(Galileo) 등 세계 각국의 Global Navigation Satellite System(GNSS)는 인공위성 기반 지상의 사용자에게 고정밀 위치정보와 속도정보를 제공한다. 그러나 터널, 지하도로, 건물 등 위성의 가시성을 확보할 수 없는 공간에서는 GNSS의 정확도가 급격하게 낮아지므로 가속도 센서, 자이로 센서 등 다양한 관성 센서 융합을 통해 움직이는 물체의 위치, 자세, 속도 추정이 가능한 Inertial Navigation System(INS) 구축이 필요하다[1, 2]. INS는 Inertial Measurement Unit(IMU)에서 측정된 데이터를 상대적 위치변화 추정 프로세스인 Odometry에 입력하여 궤적을 추정한다. 이러한 INS의 정확도를 향상시키기 위해서 IMU 기반 Odometry에 대한 연구가 활발히 진행되고 있다.

IMU 기반 Odometry에 대한 연구는 선형가속도의 이중적분을 통해 위치를 추정하는 Strapdown Inertial Navigation System(SINS)[3-5]와 보행자의 걸음과 같은 행동 패턴을 이용한 관성항법인 Pedestrian Dead Reckoning(PDR)[6-8]이 있다. SINS는 IMU에서 측정된 가속도와 자이로를 통해 자세를 추정한 후 중력성분을 제거한 가속도를 적분하여 속도 및 위치 추정이 가능하지만 피드백 기능이 없는 Open-Loop 시스템이다. 따라서 측정 기기의 내외재적 문제에 의해 발생하는 Inertial System Drift의 누적을 보상해줄 수 없으므로 시간이 지남에 따라 위치가 발산한다[5]. 또 다른 방법은 사용자의 발, 손목, 머리, 가슴 등 신체에 부착된 IMU를 사용하여 Inertial System Drift를 보상하는 PDR이다[6]. PDR은 걸음 검출과 보폭 추정으로 자세를 갱신함으로써 2차원 지도에서 보행자의 궤적을 추정한다. 그러나 이는 2차원 궤적을 추정하는 3-DOF(Degree of Freedom) Odometry로 일반적인 3차원 궤적인 6-DOF 환경에 적용하기 어렵다.

최근에는 컴퓨팅 파워가 증가됨에 따라 Odometry에 대한 연구는 방대한 정보를 포함하고 있는 이미지로 입력데이터를 확장하였고, 연산량이 방대하지만 우수한 성능을 보여주는 학습 기반 알고리즘에 대한 연구가 활발하게 진행되고 있다. 이에 해당하는 방법으로는 이미지를 사용하는 Visual Odometry(VO), 이미지와 센서 데이터를 사용하는 Visual Inertial Odometry(VIO), 센서 데이터만을 사용하는 Inertail Odometry(IO)가 있다. 순차적 이미지에서 카메라 움직임에 의한 변화를 분석하여 위치와 방향을 추정하는 VO는 NASA의 화성 탐사선의 위치를 추정하는 것[9, 10]을 시작으로 이미지 내 기하학적 특징을 이용하여 제한된 자세를 추정하는 방향[11, 12]으로 연구가 진행되어 왔다. 하지만 시스템 모델 기반인 VO는 이미지만을 사용하여 기준좌표계 설정을 위해 적어도 3개의 교차 평면이 필요하기 때문에 실제 환경에 적용 시 한계가 존재한다. 이를 해결하기 위해 이미지의 Depth Map 생성을 통해 위치와 자세를 추정하는 VO에 대한 연구가 [13, 14]와 같이 진행되었고 VO의 기반기술인 Depth Map 추정에 대한 연구[15, 16] 또한 관심도가 높다.

한편 빛, 그림자, 카메라 내부 문제 등에 취약한 이미지와 주변환경에 의한 영향은 작지만 측정시간이 길어짐에 따라 성능이 저하되는 IMU 센서의 융합을 통해 두 센서의 문제점을 보완하여 안정적인 성능을 목표로 하는 VIO에 대한 연구[17-21]와 이에 학습기법을 적용한 연구[22, 23]가 진행되었다. 이러한 VO와 VIO는 우수한 성능을 보이지만, 입력데이터인 이미지는 큰 데이터 크기로 인해 연산량이 기하급수적으로 증가하고 학습되지 않은 환경에서는 낮은 성능을 보인다. 반면에 IO은 실제 환경에 Odometry를 구현하기 위한 조건인 입력데이터를 축소시키며 학습되지 않은 환경도 적용이 가능하다. 또한 IO의 성능향상은 VIO의 성능향상을 의미하므로 IO에 대한 연구가 각광받고 있다.

IO는 IMU에서 측정되는 데이터를 사용한 Odometry로써 시작점으로부터의 상대적인 위치와 자세 추정이 가능하다. IO에 사용되는 IMU는 자유가속도와 중력가속도의 합인 가속도와 자이로 측정이 가능한 6축 IMU와 추가적으로 지자기 측정이 가능한 9축 IMU 두 종류가 있고, IO의 입력데이터는 자유가속도, 자이로, 중력가속도, 지자기 네 가지로 구성된다. Solin의 연구에서 모바일 기기로 획득된 IMU 데이터에 Extended Kalman Filter Framework를 적용한 연구가 진행되었다[24]. 최근 자유가속도와 자이로를 입력으로 위치 및 자세 변화량 추정을 통해 궤적을 복원하는 학습모델이 제안되고 있다.

Yan의 연구는 VO의 낮은 추정속도를 개선하기 위한 학습기반 IO로 6축 IMU 데이터를 입력하여 SVM(Support-Vector Machine)을 통해 자세를 추정함으로써 부착된 위치를 분류한 후, 이에 따른 속도를 SVR(Support-Vector Regression)으로 추정한다[25]. 이 방법은 PDR의 제한조건인 IMU의 위치가 고정된다는 문제점을 해결하여 실제 환경 적용에 대한 확장성을 향상시켰지만, 2차원 궤적만을 추정하는 3-DOF Odometry이고 Regression을 사용하여 추정오차가 크다.

Chen의 연구는 매 시점의 자유가속도와 중력가속도를 입력으로 FC(Fully Connected Neural Network)와 Bi-LSTM(Bidirectional Long Short-Term Memory)을 사용하여 극좌표와 자세를 추정하는 모델이다[26]. 이는 IMU의 위치 및 움직임에 대한 정보를 사용하지 않고 측정된 데이터만을 이용하여 위치를 추정한다. 그러나 자세에 따라 진행방향이 결정된 3-DOF Odometry이므로 진행방향 외 움직임에 대한 추정이 어렵다.

Lima의 연구는 잡음과 노이즈를 포함한 6-Axis IMU 데이터를 입력으로 CNN과 Bi-LSTM을 사용하여 3차원 속도와 자세변화를 추정하는 6-Axis IONet이다[27]. 그러나 자유가속도와 자이로만이 입력된 네트워크

의 출력인 자세변화량을 적분하여 자세를 추정하는 Open-Loop 시스템이므로 지속적으로 Inertial System Drift가 발생한다. 이로 인해 정확한 자세변화량 추정이 어려우므로 이동방향에 대한 오차가 누적된다.

본 논문에서는 Inertial System Drift를 해결하기 위해 자세정보를 포함한 데이터를 추가적으로 입력하는 Modified Inertial Odometry Network(Modified IONet)을 제안한다. 제안하는 네트워크는 기존 6-Axis IONet에 IMU의 자세각 정보를 포함하는 중력가속도와 자세각을 통해 방위각을 계산할 수 있는 지자기를 추가하여 궤적 추정성능을 향상시킨 9-Axis IONet이다. 이를 임베디드 및 모바일 환경에 적용하기 위해 LSTM Layer를 GRU로 대체한 fast 6-Axis IONet과 fast 9-Axis IONet을 제시하고 이에 대한 분석을 진행하였다[28].

제 2 장 Related Work

2.1 Pose Measurement based on Gravity Acceleration

중력가속도는 지구 중력에 의해 물체가 받는 가속도이며 지면과 수직한 방향으로 작용한다. 지구는 완벽한 구가 아니므로 지역별로 차이가 있으나 평균적으로 $9.8 m/s^2$의 값을 가진다.

$$\vec{g} = \overrightarrow{A_x} + \overrightarrow{A_y} + \overrightarrow{A_z} \qquad (2.1)$$

중력가속도 $\vec{g}$는 식 (2.1)과 같이 $\overrightarrow{A_x}$, $\overrightarrow{A_y}$, $\overrightarrow{A_z}$의 합성벡터로 이루어지며 IMU의 자세에 따라 다르게 나타난다. 그림 1은 x축 회전각인 Roll각 θ만큼 기울어졌을 시 IMU가 받는 힘을 나타낸 것이다. x축이 회전하므로 $\overrightarrow{A_y}$와 $\overrightarrow{A_z}$의 크기를 이용하여 θ를 식 (2.2)와 같이 계산할 수 있다. 그리고 y축 회전각인 Pitch각 ϕ는 식 (2.3)과 같이 $\overrightarrow{A_x}$와 $\overrightarrow{A_z}$를 통해 나타낼 수 있다.

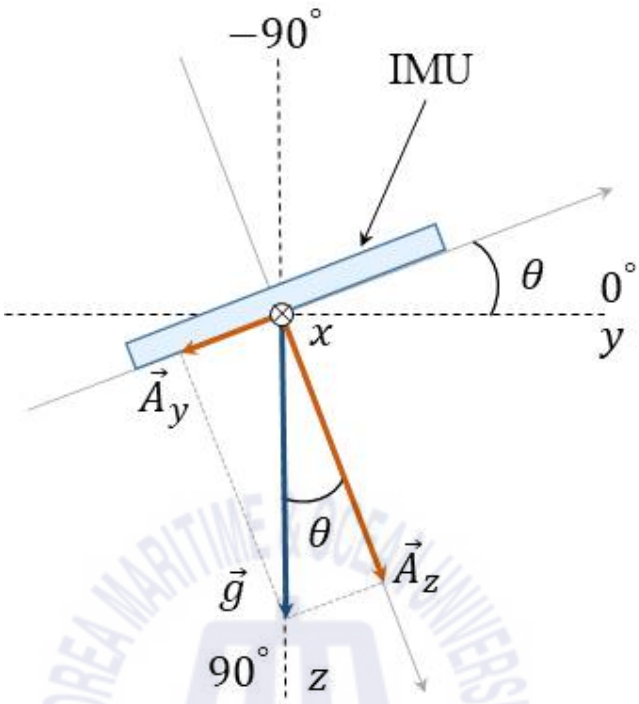


그림 2.1 중력가속도 벡터의 구성요소

Fig. 2.1 The components of gravity accelerometer vectors

$$\theta = \tan^{-1}\frac{\overrightarrow{A_x}}{\overrightarrow{A_z}} \tag{2.2}$$

$$\varnothing = \tan^{-1}\frac{\overrightarrow{A_y}}{\overrightarrow{A_z}} \tag{2.3}$$

그러나 측정되는 데이터에는 노이즈가 포함되어 있으므로 한 시점의 데이터를 이용하면 정확한 자세를 계산하기 어렵다. 따라서 정확한 자세를 얻기 위해서는 안정적인 데이터 입력이 필수적이다.

2.2 Azimuth Measurement based on Geomagnetic

지구의 자기벡터는 그림 2.2과 같이 7가지로 구성된다. X는 지구 자전축 북쪽 끝에 있는 지리적 북쪽(진북), Y는 지리적 서쪽, Z는 지구 천저를 향하는 연직자기력, F는 지구자기력, H는 수평자기력(자북 방향), I는 지구자기력과 수평면 사이의 각(복각), D는 진북과 자북 사이의 각(편각)이다.

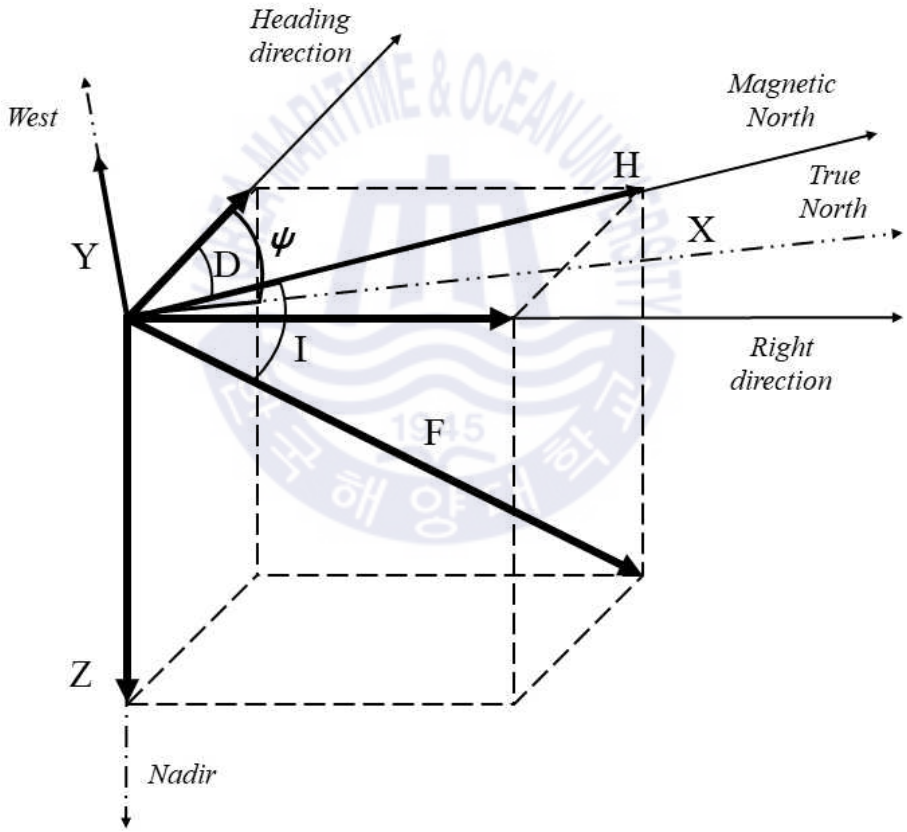


그림 2.2 지자기 벡터의 구성요소

Fig. 2.2 The components of magnetic vectors

식 (2.4)은 지구자기력 F를, 식 (2.5)는 수평자기력 H를 나타내는 식이다.

$$F = \sqrt{Hd^2 + Rd^2 + Nd^2} \tag{2.4}$$

$$H = \sqrt{Hd^2 + Rd^2} \tag{2.5}$$

식 (2.4), (2.5)에서 Hd, Rd, Nd는 각각 진행방향(Heading Direction), 우측 방향(Right Direction), 천저 방향(Nadir Direction)이다. 자세각을 계산하기 위해 현재 자세에 따른 수평 좌표계 변환이 필수적이며 식 (2.6)과 같다.

$$\begin{pmatrix} X_H \\ Y_H \\ Z_H \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varnothing & -\sin\varnothing \\ 0 & \sin\varnothing & \cos\varnothing \end{pmatrix} \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} \tag{2.6}$$

θ와 ϕ는 롤각(Roll), 피치각(Pitch)이고 m_x, m_y, m_z는 측정된 지구 자기장이며 X_H, Y_H, Z_H는 자세가 보상된 자기벡터값이다. 이를 이용하여 방위각 Ψ를 식 (2.7)와 와 같이 계산할 수 있다.

$$\psi = \tan^{-1}\left(\frac{m_y\cos\theta - mzsin\varnothing}{m_x\cos\theta + m_y\sin\varnothing\sin\varnothing + m_z\cos\varnothing\sin\theta}\right) + L \tag{2.7}$$

방위각은 자세 예측에 필수적인 요소이며, 고정된 위치일 때 방위각에 따른 이상적인 지자기 벡터는 그림 2.3과 같이 원의 형태로 표현된다. 그러나 지자기 센서의 내부적으로는 전기적 내재오차인 Soft-Iron Effect가 그림 2.4와 같이 발생하여 지자기 벡터 형상의 중심이 임의로 이동되고, 외부적으로는 철근 구조물 및 자성체에 따른 오차인 Hard-Iron Effect가 발생하여 지자기 벡터 형상의 중심이 그림 2.5와 같이 임의의 타원형태로 변화한다[29]. 따라서 서로 다른 위치에서 측정된 지자기 벡터로는 정확한 방위각을 추정하기 어렵다.

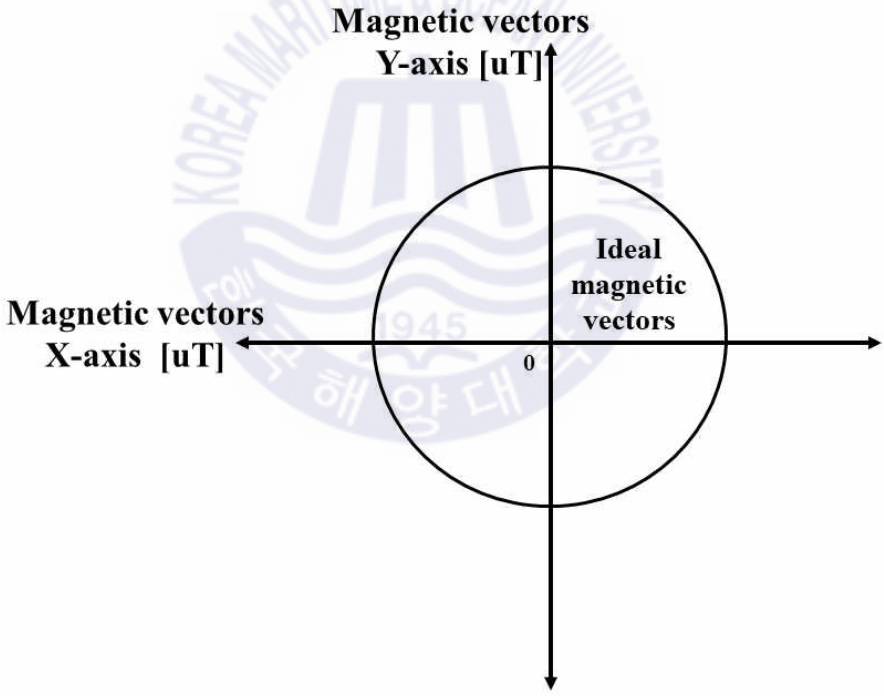


그림 2.3 이상적인 자기벡터의 특성
Fig. 2.3 Ideal Magnetic Vectors

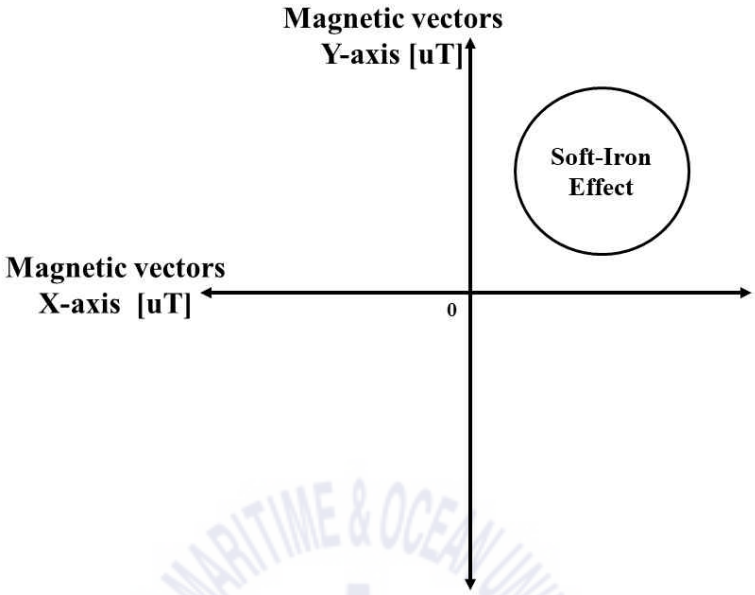


그림 2.4 센서의 내재적 오차

Fig. 2.4 Soft-Iron Effect

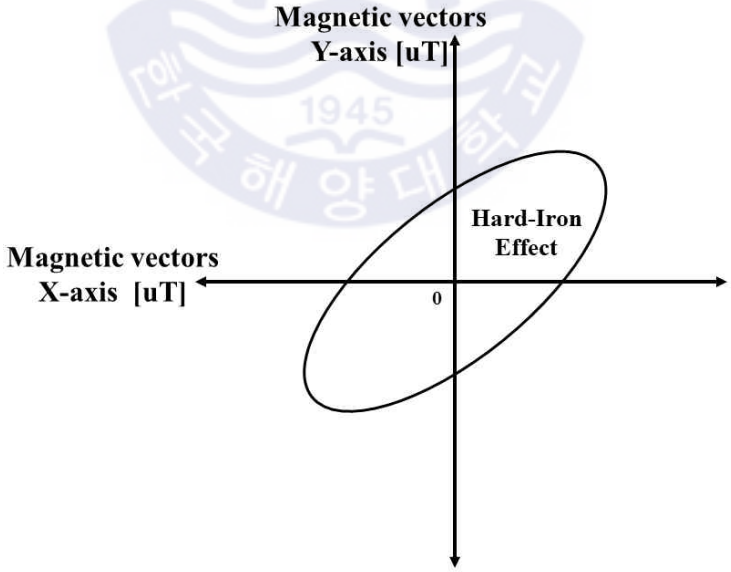


그림 2.5 센서의 외재적 오차

Fig. 2.5 Hard-Iron Effect

2.3 Convolutional Neural Network

CNN(Convolutional Neural Network)는 데이터의 특징 벡터를 추출하는 네트워크로 객체 분류 및 검출과 같은 이미지가 입력데이터인 모델에서 우수한 성능을 보인다. 최근 신호처리 분야에서는 IMU, Lidar 등 센서에서 출력되는 시계열 데이터를 보정하기 위한 방법으로 단방향 필터를 사용하는 1D CNN을 사용한다. 그림 2.6은 1D CNN 모델 구조를 나타낸 것이다. 일반적인 1D CNN은 Convolutional Layer와 Pooling Layer로 이루어져 있으며, 시계열 데이터의 특징추출을 목적으로 하여 이후 Recurrent Neural Network 계열의 순차적인 정보를 처리하는 학습 알고리즘과 융합하여 사용한다.

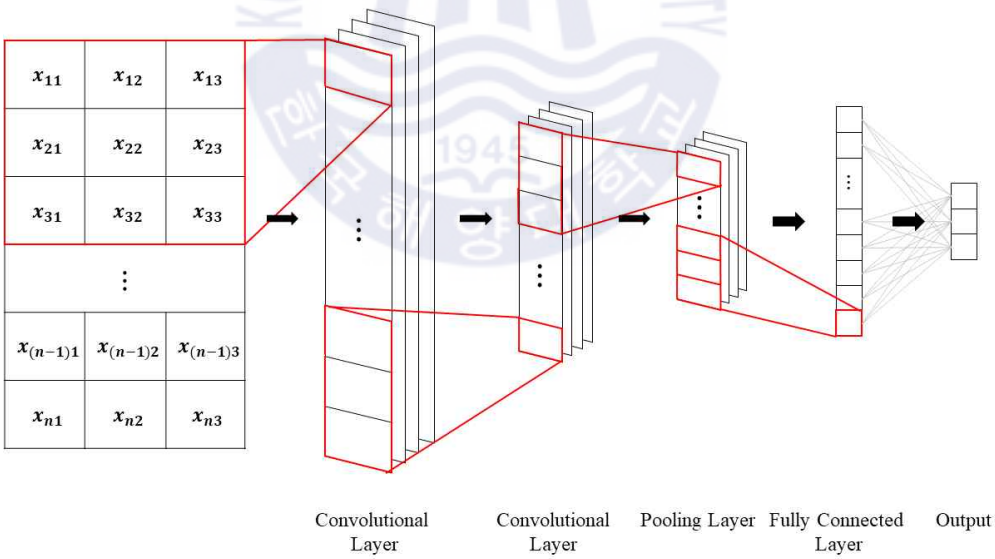


그림 2.6 1D Convolutional Neural Network의 구조

Fig. 2.6 The Structure of 1D Convolutional Neural Network

Convolutional Layer는 Filter가 이동하면서 입력데이터와의 연산을 통해 Feature Map을 생성한다. 이때 입력하는 시계열 데이터수를 Window Size라 하는데, 목적에 맞게 Window Size를 조정하여 입력데이터로 사용한다. 1D Convolution Layer는 2D Convolution Layer와는 달리 Filter를 단방향으로만 이동시켜 합성곱 연산을 함으로써 Feature Map의 순서에 따라 시간적 특성을 반영한다. Feature 추출에 사용되는 Filter의 크기는 클수록 세밀한 Feature 추출이 어렵고, 작을수록 데이터 잡음에 취약하고 연산량이 많아 목적을 고려하여 구성해야 한다.

Pooling Layer는 Convolutional Layer 이후 생성되는 Feature Map의 Overfitting을 방지하고 연산시간을 감소시키기 위한 과정이다. Convolutional Layer의 결과물인 Feature Map은 입력데이터와 비교했을 때 크기가 방대해지는데 이를 그대로 사용할 경우 Overfitting이 발생할 가능성이 높고 모든 Feature에 대해 연산을 진행하여 연산시간이 길어진다. 이를 해결하기 위해 Feature Map 압축 역할을 하는 Pooling Layer를 사용하는데 Max Pooling과 Average Pooling을 많이 사용한다. Max Pooling은 CNN에서 가장 많이 사용하는 Pooling Layer로 일정한 범위에 존재하는 특징들에 대한 최대값을 선택하여 유리한 Invariance를 얻을 수 있고, Average Pooling은 평균값을 선택하여 잡음이 많은 데이터에서 안정적인 Feature Map을 출력할 수 있다.

2.4 Recurrent Neural Network

RNN(Recurrent Neural Network)은 시계열 데이터와 같은 순차적 정보처리가 필요한 데이터 학습에 사용하는 알고리즘이다. 그림 2.7은 RNN의 구조를 나타낸 것이다. RNN은 시간적으로 연결된 입력데이터 x_t, 이전 시점의 모든 입력에 대한 함수인 RNN Cell, Hidden Layer h_t, 그리고 출력데이터 y_t로 구성된다. x_t는 시점에 따라 순차적으로 Cell로 입력되고, h_t와 y_t가 나오게 된다. h_t는 이전 시점의 특성을 지속적으로 입력으로 보내는 역할을 수행한다. 이러한 RNN의 구조는 모든 시점이 영향을 주지만 입력데이터 초기 시점에 대한 영향력이 낮아지는 장기 의존성 문제(Long Term Dependency Problem)를 발생시킨다. 이를 해결하기 위해 RNN Cell의 구조를 개선한 LSTM과 RNN의 구조를 개선한 Bi-RNN을 사용한다.

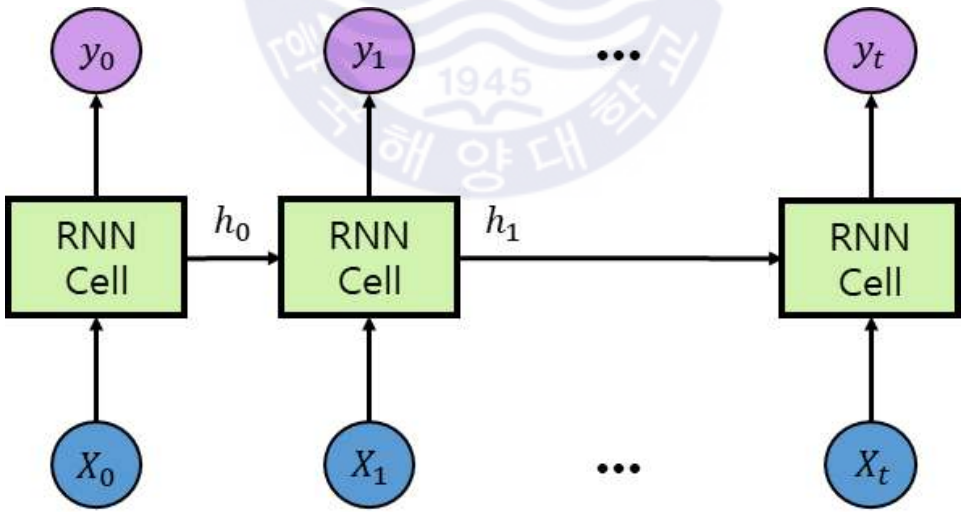


그림 2.7 Recurrent Neural Network 구조

Fig. 2.7 The structure of Recurrent Neural Network

2.5 Long Short-Term Memory

LSTM(Long Short-Term Memory)은 많은 시점의 데이터를 한번에 입력할 경우 초기 입력데이터에 대한 반영이 되지 않는 RNN의 문제점을 해결하기 위해 개발된 알고리즘이다[30]. LSTM의 기본적인 동작방식은 RNN과 유사하나 입력, 망각, 출력 게이트를 추가함으로써 장기 의존성 문제를 해결하였다. 그림 2.8은 LSTM의 Cell 구조를 나타낸 것이다.

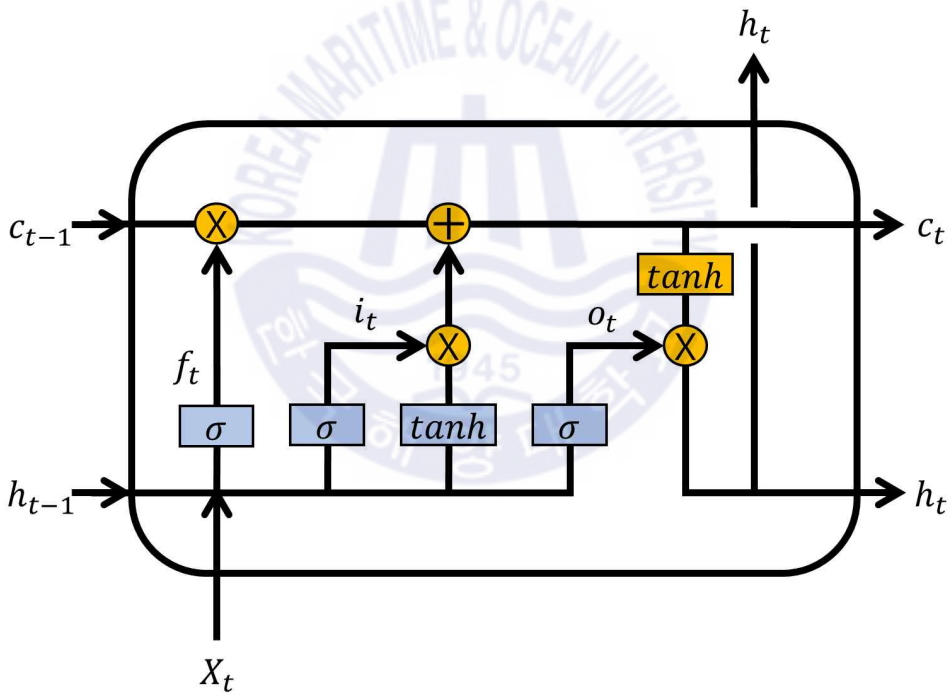


그림 2.8 Long Short-Term Memory Cell 구조

Fig. 2.8 The structure of Long Short-Term Memory Cell

그림 2.8에서 c_t는 Cell의 상태, h_t는 현재 Cell의 출력, i_t는 입력게이트, f_t는 망각 게이트, o_t는 출력 게이트를 의미한다. LSTM은 이전 셀 상태에서 망각 게이트를 통해 셀 상태 데이터의 일정량을 소멸시키고, 이전 출력값과 현재의 입력값을 입력 게이트의 출력과 곱하여 입력데이터를 조절함으로써 현재 셀 상태를 갱신한다. 그리고 셀의 출력과 출력 게이트를 곱함으로써 출력 데이터를 조절한다. 이와 같이 LSTM은 이전 셀 상태를 얼마만큼 망각하고 새로운 입력데이터를 어느 정도 받아들일지 조절함으로써 현재의 셀 상태를 갱신하기 때문에 RNN에서 발생하는 Gradient Vanishing 현상을 개선하여 긴 시퀀스 데이터도 학습이 가능하다. 그러나 LSTM은 복잡한 연산과정 때문에 임베디드 시스템에 적용하기 어렵다. 이를 개선하기 위해 빠른 속도와 준수한 성능을 보이는 GRU를 많이 사용한다.

2.6 Gated Recurrent Unit

그림 2.10는 Gated Recurrent Unit(GRU)의 구조를 나타낸다. GRU는 입력, 망각, 출력 게이트를 가지는 LSTM의 많은 연산량을 감소시키기 위한 RNN의 변형 모델로 입력 및 망각 게이트를 하나의 업데이트 게이트로 통합하였다[31]. 이는 LSTM을 단순화 함으로써 연산량을 감소시켰고 LSTM과 성능이 유사하기 때문에 다양한 분야에서 적용되고 있다. GRU는 데이터가 작고 특성이 빈번하지 않은 작업에서 LSTM보다 높은 성능을 보이지만 파라미터 수가 작다. 따라서 제안하는 모델의 결과는 미분된 데이터를 가지므로 높은 성능을 기대하기 어렵다. 그러나 LSTM보다 연산량이 약 30% 작으므로 연산량에 한계가 있는 임베디드 시스템 적용 시 필수적이다.

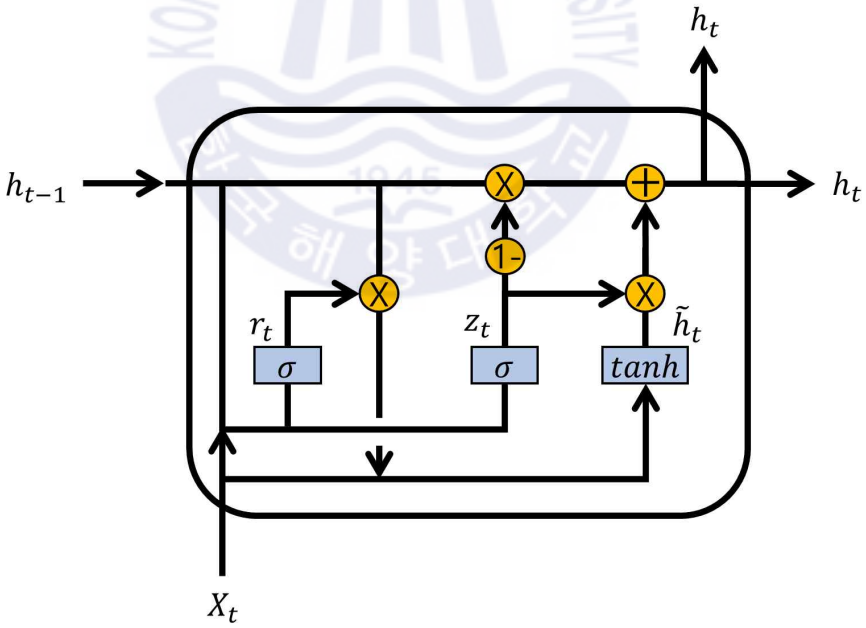


그림 2.9 Gated Recurrent Unit Cell 구조

Fig. 2.9 The structure of Gated Recurrent Unit Cell

2.7 Bidirectional Recurrent Neural Network

Bi-RNN(Bidirectional Recurrent Neural Network)은 순차적 데이터 내에서 이전 시점뿐만 아니라 이후 시점의 데이터까지 고려하는 RNN의 구조적 확장 모델이다[32]. 그림 2.10는 Bi-RNN의 구조를 나타낸다.

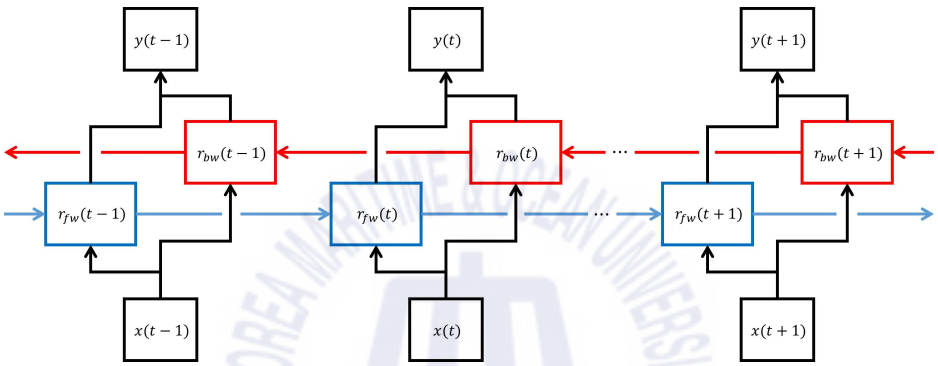


그림 2.10 Bidirectional Recurrent Neural Network 구조

Fig. 2.10 The structure Bidirectional Recurrent Neural Network

$x(t)$는 입력, $r_{fw}(t)$는 순방향 출력, $r_{bw}(t)$는 역방향 출력 그리고 $y(t)$는 Bi-RNN의 출력이고 식 (2.8)과 같이 나타낼 수 있다.

$$y(t) = [r_{fw}{}^{T}(t), r_{bw}{}^{T}(t)]^{T} \quad (2.8)$$

Bi-RNN은 순방향 RNN과 역방향 RNN이 병렬로 존재하며 두 RNN의 출력을 합산하여 최종출력으로 사용한다. 이는 학습하는 데이터의 양을 늘려줌으로써 기존 RNN에 비해서 한정된 데이터를 효율적으로 사용한 학습 방법이고 순차적 데이터를 입력으로 하는 센서 데이터 처리에서 우수한 성능을 보인다.

3. Proposed Methods

본 논문은 9-Axis IMU에서 측정된 자유가속도, 자이로, 중력가속도, 지자기를 입력으로 속도, 자세변화량를 추정하는 3가지 Modified IONet(Modified Inertial Odometry Network)을 제안한다. Modified IONet은 IMU에서 측정되는 가속도에서 중력가속도를 제거하여 Global 좌표계 내 이동변화량을 나타내는 자유가속도, 자세변화량을 측정하는 자이로, 지구중력을 측정하여 IMU의 자세각에 대한 정보를 포함하고 있는 중력가속도, 지구자기장을 측정하여 방위각에 대한 정보를 포함하고 있는 지자기를 입력으로 하며, CNN과 Bi-LSTM을 이용하여 속도와 자세변화량을 추정하는 9-Axis IONet이다. 이는 기존 자유가속도와 자이로만을 입력으로 한 6-Axis IONet에 비해 자세각과 방위각에 대한 정보를 추가적으로 입력함으로써 정확성을 향상시킨다. 또한 제안하는 네트워크 내 IMU의 순차적 데이터를 처리하기 위한 Layer인 LSTM은 우수한 성능을 보이지만 방대한 연산량으로 인해 많은 추정시간이 소요되므로 이를 개선하기 위해 Bi-LSTM Layer를 Bi-GRU Layer로 대체하는 fast 6-Axis IONet과 fast 9-Axis IONet 실험을 진행하였다.

3.1 9-Axis IONet

그림 3.1은 9-Axis IONet의 구조를 나타낸다. 9-Axis IONet의 입력데이터는 자유가속도 a, 자이로 w, 중력가속도 g, 지자기 m이며, 네트워크는 지역적인 특징을 추출하기 위한 CNN과 순차적인 특징을 추출하기 위한 Bi-LSTM으로 구성된다. 자유가속도는 IMU에서 측정된 가속도 데이터에서 중력가속도를 뺀 값으로 Global 좌표계에서의 이동변화량이며 자이로는 자세변화량이다. 중력가속도는 IMU가 받는 중력을 3-Axis로 측정하여 자세각 Roll각 θ와 Pitch각 ϕ을 계산할 수 있다. 지자기 데이터는 자기장을 측정하는 값으로 중력가속도를 통해 계산된 자세각을 이용하면 방위각 Yaw ψ를 계산할 수 있다. 이러한 각 데이터는 6-Axis IONet[27]과 같이 매 10 stride마다 200 Frame씩을 하나의 입력데이터로 생성하며 추정 대상인 위치변화량 $\triangle p$와 자세변화량 $\triangle q$는 데이터 1세트 내 95번째 데이터와 105번째 데이터 사이의 차로 계산된다. 또한 네 가지 데이터 간 간섭을 최소화하기 위해 분리하여 입력한다.

제안하는 네트워크는 Convolutional Layer, Bi-LSTM Layer로 구성되어있으며 이를 이용하여 $\triangle p$와 $\triangle q$를 추정한다. Convolutional Layer는 다양한 필터를 이용하여 데이터의 인접한 시간동안 IMU 자세 및 위치 변화에 대한 Feature Map을 생성하며 MaxPooling Layer를 통해 이를 강인하게 한다. 이때, 입력이 2차원 시계열 데이터이므로 1D Convolutional Layer, 1D MaxPooling Layer를 사용한다. Bi-LSTM Layer는 순차적인 데이터에 대한 특징을 추출하며 순방향 LSTM과 역방향 LSTM을 병렬로 활용하여 최종적으로 $\triangle p$와 $\triangle q$를 추정한다.

9-Axis IONet은 6-Axis IONet[27]에 자세와 명확한 연관성을 가진 중력 가속도와 지자기를 추가적으로 입력받음으로써 Inertial System Drift가 작은 자세를 학습하여 우수한 궤적 추정성능을 보인다.

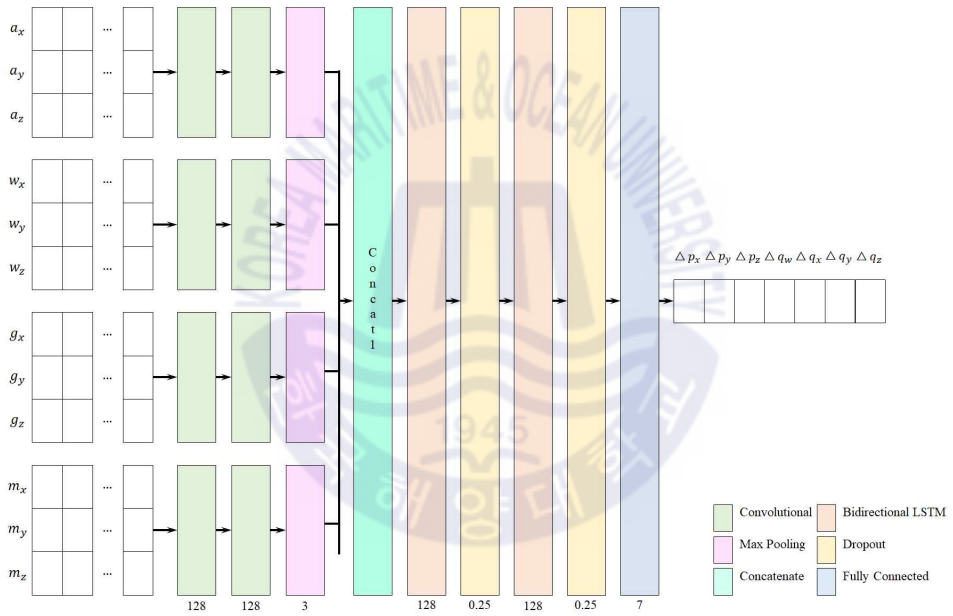


그림 3.1 9-Axis IONet 구조

Fig. 3.1 The structure of 9-Axis IONet

3.2 fast 6-Axis IONet

컴퓨팅 파워의 제약이 있는 임베디드 시스템에 IONet을 적용하기 위해서는 LSTM에 비해 연산에 사용되는 파라미터 수가 약 30% 작은 GRU를 사용한 네트워크 적용이 필수적이다. 그림 3.2는 6-Axis IONet의 Bi-LSTM Layer를 Bi-GRU Layer로 대체한 fast 6-Axis IONet이다. fast 6-Axis IONet은 파라미터 수가 감소됨으로써 6-Axis IONet에 비해 낮은 성능을 보이지만 연산속도에서 강점을 보인다. 이는 임베디드 시스템, 모바일 시스템 등 낮은 성능의 GPU를 사용하는 환경에 적용 시 유효하다.

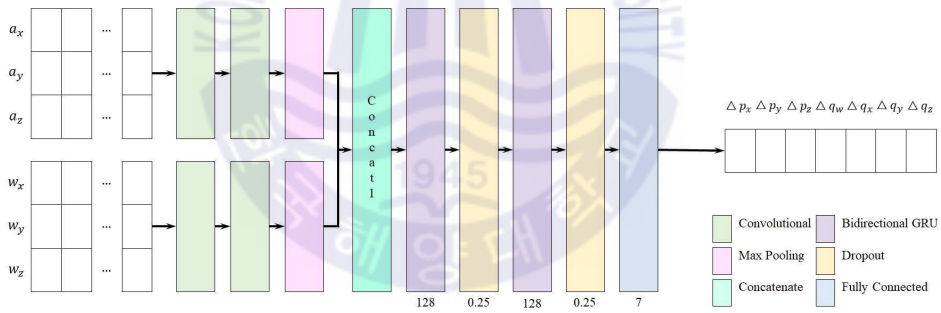


그림 3.2 fast 6-Axis IONet 구조

Fig. 3.2 The structure of fast 6-Axis IONet

3.3 fast 9-Axis IONet

IONet에 사용되는 두 Layer 중 LSTM Layer는 Cell 내부 연산이 복잡하므로 전체 네트워크 연산속도를 증가시키기 위해서는 이에 대한 개선이 필수적이다. 그림 3.3은 9-Axis IONet의 Bi-LSTM을 Bi-GRU로 대체하여 연산량을 감소시킨 네트워크이다. 중력가속도와 지자기가 입력으로써 추가되면서 Bi-LSTM Layer에 입력되는 Feature Map의 크기가 커지면고 이에 따라 파라미터 수가 기하급수적으로 증가하게 된다. 이러한 경우 LSTM Layer를 GRU Layer로 대체하는 것은 현재 뉴럴 네트워크 기반 추정기술 적용을 위해서는 필수적인 과정이다.

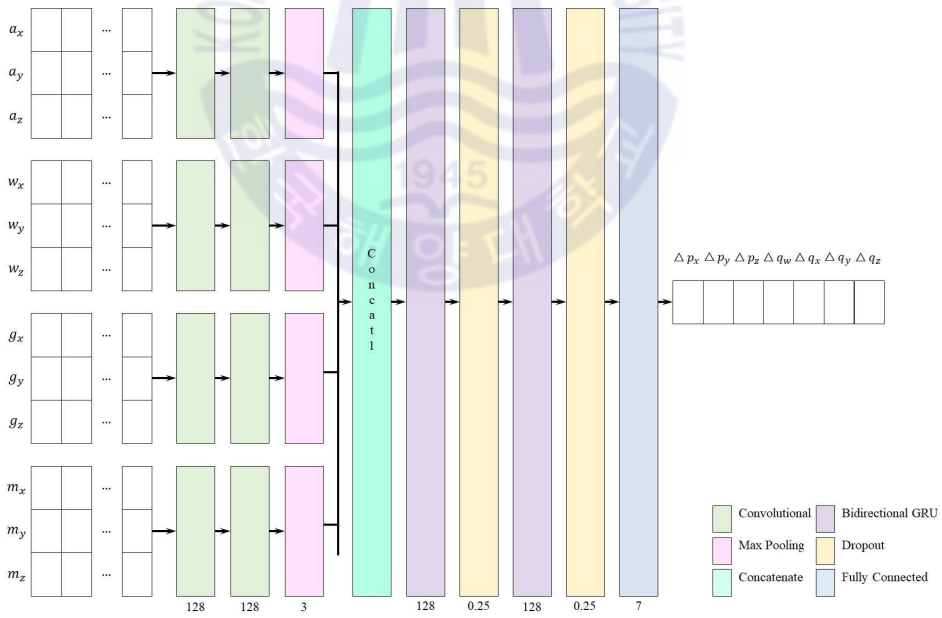


그림 3.3 fast 9-Axis IONet 구조

Fig. 3.3 The structure of fast 9-Axis IONet

3.4 6-DOF Pose Representation

6-DOF 자세를 표현은 다양한 방식으로 가능하지만, 본 논문은 6-Axis IONet과 같이 모든 자세에 대해서 6-DOF 표현이 가능한 3D translation vector $\triangle p$와 unit quaternion $\triangle q$ 그리고 자세 표현이 가능한 이전 자세 q_{t-1}을 사용한다[27]. 이 방법은 식 (3.1)와 같이 이전 위치와 자세에 단위 시간 간 변화량을 더해서 현재 위치와 자세를 도출한다.

$$\begin{cases} p_t = p_{t-1} + R(q_{t-1})\triangle p \\ q_t = q_{t-1} \otimes \triangle q \end{cases} \tag{3.1}$$

p_t는 현재 위치이며 이전 위치 p_{t-1}에 q만큼의 회전을 의미하는 행렬 $R(q)$와 $\triangle p$의 곱인 단위시간 사이의 위치 변화량을 합하여 계산한다. 현재 자세 q_t는 이전 자세인 q_{t-1}와 $\triangle q$의 Hamilton product $\otimes$를 통해 계산된다. Hamilton Product는 두 Quaternion 간 자세 합을 의미한다[33]. 뉴럴 네트워크로부터 추정된 Quaternion은 자세를 표현하는 Unit Quaternion의 형태를 유지하기 위해 정규화한다.

3.5 Loss Function

제안하는 네트워크는 △p, △q 그리고 q에서 발생하는 Loss를 최소화하기 위해 학습하며, 실수의 범위를 가지는 p와 복소수의 범위를 가지는 q를 모두 Loss Function에 반영하기 위해 식 (3.2)와 같이 서로 다른 수체계를 통합할 수 있는 Multi-task Learning 기법을 사용한다[34].

$$L_{total} = \sum_{i=1}^{n} \exp(-\log \sigma_i^2) L_i + \log \sigma_i^2 \tag{3.2}$$

σ_i^2과 L_i는 i번째 task의 분산와 Loss Function이다. L_{total}에는 분산을 회귀하는 것보다 0에 의한 Loss 분열을 피할 수 있어 안정적인 Log 분산 예측을 사용한다. L_{total} 연산에는 세 가지 Loss Function이 사용된다.

$$[L_1, L_2, L_3] = [L_{DPMAE}, L_{DQME}, L_{QME}] \tag{3.3}$$

식(3.4)는 Delta Position Mean Absolute Error L_{DPMAE}이고, 식(3.5)는 Delta Quaternion Mean Error L_{DQME}이다.

$$L_{DPMAE} = \| \triangle \hat{p} - \triangle p \|_1 \tag{3.4}$$

,

$$L_{DQME} = 2^* \| imag(\triangle \hat{q} \otimes \triangle q^*) \|_1 \tag{3.5}$$

L_{DPMAE}와 L_{DQME}는 IONet의 속도와 자세변화량에 대한 Loss Function으로 Ground Truth ($\triangle p$, $\triangle q$)와 Model의 Predicted Pose ($\triangle \hat{p}$, $\triangle \hat{q}$) 간 기하학적 변화량을 학습시킨다. $\triangle p$는 궤적의 미분값이고, $\triangle q$는 IMU의 자세변화량을 표현하는 값이므로 3D Pose graph SLAM error의 절대값을 Loss Function으로 사용한다[27]. 이때 $\triangle q^*$은 $\triangle q$의 켤레복소수로 반대 위상을 의미하므로 $\triangle \hat{q}$와 $\triangle q^*$의 Hamilton Product $\otimes$는 $\triangle \hat{q}$와 $\triangle q$의 차를 의미하며, 뉴럴 네트워크로부터 추정된 $\triangle \hat{q}$는 크기가 1인 Unit Quaternion의 형태이다.

제 4 장 Experiment

4.1 OxIOD Dataset

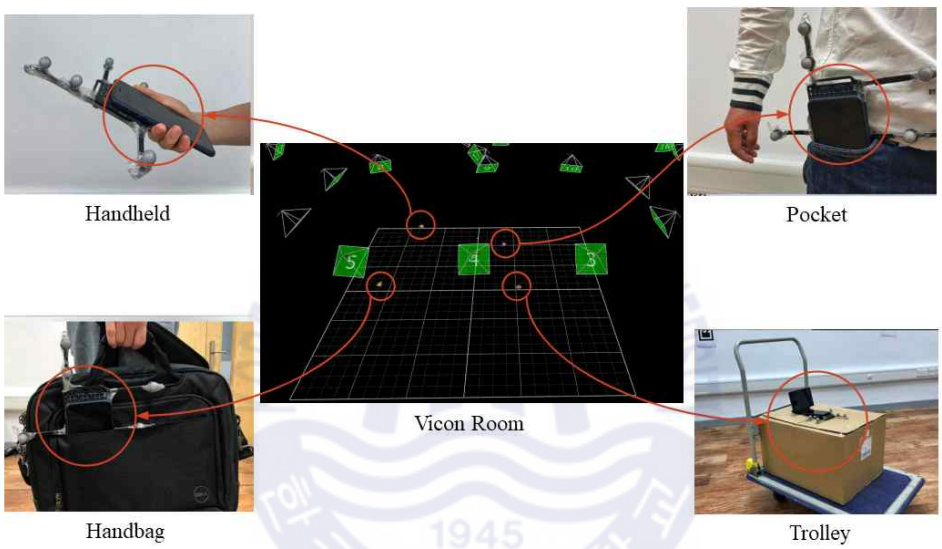


그림 4.1 데이터 수집 환경 및 부착위치

Fig. 4.1 Environment collected Data and Attachment Location

Oxford Inertial Odometry Dataset[35]은 그림 4.1과 같이 Vicon Motion Capture System을 사용하여 Ground Truth를 제작되었고, 저가형 IMU와 지자기 센서가 내장된 iPhone 7 plus를 네 가지 부착위치에서 측정된 데이터이다. 실험에는 Handheld 데이터를 이용하였으며 이는 100Hz로 서로 다른 조건에서 측정된 24개의 데이터로 구성되며 표 4.2과 같이 17개의 Training Data와 7개의 Testing Data로 분리하였고 Ground Truth의 측정 노

이즈 때문에 각 데이터의 초기 12초와 마지막 3초를 소거한 후 실험을 진행하였다. 1개의 입력데이터는 200 프레임이고 데이터 간 10프레임의 Stride를 두었으며 55,003개의 입력데이터가 존재한다. 네트워크에 대한 평가는 Testing Data의 초기 20초를 사용하여 Output 데이터를 얻은 후 실제 이동 궤적에 관한 Qualitative 평가 및 실제 이동 궤적과의 차를 통한 Quantitative 평가를 진행한다. 또한 각 네트워크의 속도를 확인하기 위해 100초간 데이터의 처리속도와 네트워크의 파라미터 수를 확인한다.

표 4.1 벤치마크 데이터셋의 분류

Table 4.1 Categorizing of the Benchmark Dataset

Training Data	Testing Data
data1/seq1	data1/seq2
data1/seq3	data1/seq5
data1/seq4	data1/seq6
data1/seq7	data3/seq1
data2/seq1	data4/seq1
data2/seq2	data4/seq3
data2/seq3	data5/seq2
data3/seq2	
data3/seq3	
data3/seq4	
data3/seq5	
data4/seq2	
data4/seq4	
data4/seq5	
data5/seq2	
data5/seq3	
data5/seq4	

4.2 Detail of Training

학습의 Framework는 Keras 2.2.5와 TensorFlow 1.14.0을 사용하였고, GPU는 NVIDIA RTX TITAN을 사용하였다. 학습 시 학습률 0.0001의 Adam Optimizer를 사용했고 Validation Data는 10%로 설정하였으며 Network의 학습횟수는 500회이다. Batch Size는 32로 설정하였다. 그림 4.2은 LSTM을 사용한 9-Axis IONet의 Loss 변화, 그림 4.3, 그림 4.4는 GRU를 사용한 fast 6-Axis IONet과 fast 9-Axis IONet의 Loss 변화이다.

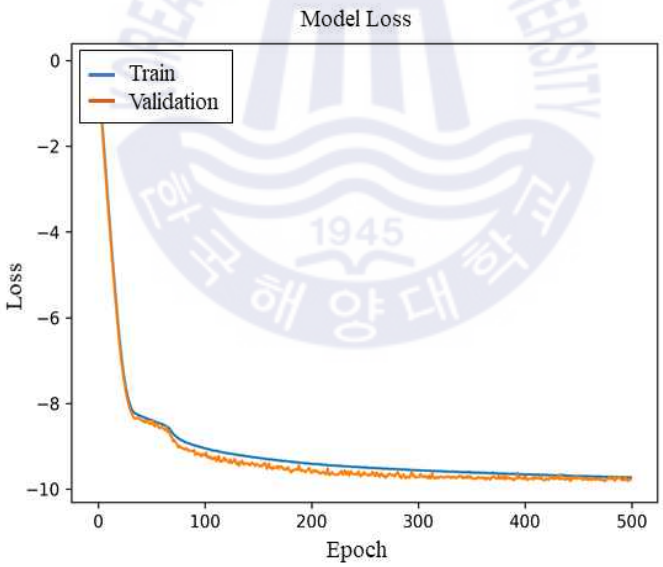


그림 4.2 9−Axis IONet의 훈련 및 검증 Loss
Fig. 4.2 Training and Validation Loss of 9−Axis IONet

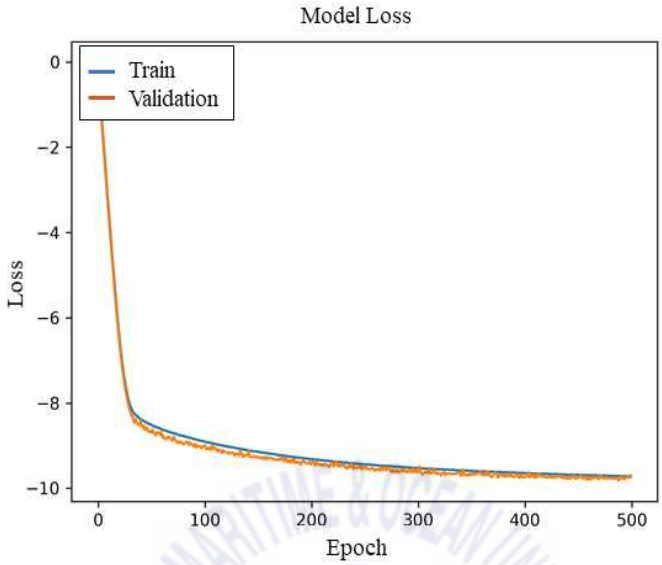


그림 4.3 fast 6−Axis IONet의 훈련 및 검증 Loss

Fig. 4.3 Training and Validation Loss of fast 6−Axis IONet

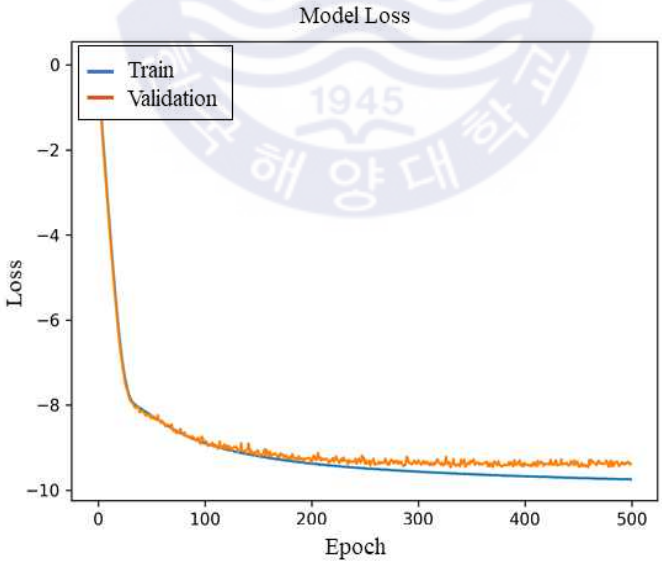


그림 4.4 fast 9−Axis IONet의 훈련 및 검증 Loss

Fig. 4.4 Training and Validation Loss of fast 9−Axis IONet

9-Axis IONet과 fast 6-Axis IONet은 400회에서 수렴하는 것으로 판단되며 fast 9-Axis IONet은 300회에서 수렴하며 이후 Overfitting이 발생하여 위치 추정 시 이상치가 발견될 가능성이 높을 것으로 판단된다.

4.3 Quantitative Evaluation

각 네트워크의 Quantitative 평가는 표 4.2과 같이 7개의 Test 데이터셋을 입력으로 각 IONet 출력과 Ground Truth와의 RMSE(Root-Mean-Square Error)를 이용하여 평가한다. 또한 표 4.3에서는 4가지 IONet의 파라미터 수를 비교하여 연산량을 확인한다.

표 4.2 네트워크별 위치추정 RMSE 비교

Table 4.2 Comparison of trajectory RMSE by Network

	6-Axis IONet	9-Axis IONet	fast 6-Axis IONet	fast 9-Axis IONet
handheld/data1/seq2	0.531	**0.363**	0.508	0.400
handheld/data1/seq5	0.624	**0.214**	0.537	0.513
handheld/data1/seq6	**0.299**	0.366	**0.299**	0.344
handheld/data3/seq1	0.530	**0.438**	0.550	0.451
handheld/data4/seq1	0.340	0.598	**0.338**	1.612
handheld/data4/seq3	**0.307**	0.581	0.419	0.514
handheld/data5/seq2	0.396	**0.192**	0.312	0.283
Mean	0.432	**0.393**	0.423	0.588

표 4.3 네트워크별 파라미터 수

Table 4.3 Parameter number by Network

	6-Axis IONet	9-Axis IONet	fast 6-Axis IONet	fast 9-Axis IONet
Parameter	1,161,735	1,793,289	964,105	1,530,121

6-Axis IONet은 7가지 실험 중 2가지에서 가장 우수한 성능을 보였으며, 9-Axis IONet은 4가지에서, fast 6-Axis IONet은 2가지에서 가장 낮은 RMSE 결과를 보였다. RMSE의 평균은 9-Axis IONet이 기존 6-Axis IONet에 비해 약 10% 우수한 성능을 보였으며 6축 두 네트워크는 비슷한 성능을 보였다. 또한 fast 9-Axis IONet은 'handheld/data4/seq1'에서 비정상적으로 낮은 성능을 보이지만 이를 제외하면 네 가지 모델 중 두 번째로 높은 성능을 보인다. 이러한 이상치는 fast 9-Axis IONet이 학습되는 과정의 Overfitting에 의한 결과라고 판단된다.

실험 결과, 자세에 대한 정보가 포함된 중력가속도와 지자기 데이터를 입력으로 추가한 것만으로 Inertial System Drift를 보정하여 RMSE를 감소시킬 수 있는 것을 확인하였다. 또한 연산량이 많은 LSTM Layer를 파라미터 수가 약 30% 작은 GRU로 대체하더라도 성능저하가 크지 않은 것을 확인하였다.

4.4 Qualitative Evaluation

Qualitative Evaluation에서는 4개의 데이터를 이용하여 궤적 추정성능을 확인하기 위해 6-Axis IONet[27], 중력가속도와 지자기가 입력 데이터로 추가된 9-Axis IONet, 그리고 경량화 모델인 fast 6-Axis IONet, fast 9-Axis IONet가 추정하는 결과와 실제 이동궤적을 나타내었다. 실험에는 'data1/seq2', 'data1/seq6', 'data4/seq1', 'data5/seq2'가 사용되었으며 Quantitative Evaluation에서 확인하지 못한 네트워크의 특성을 확인하였다. 그림 4.5와 그림 4.6은 'data1/seq2'의 결과이며 입력된 데이터가 많은 2가지의 9축 네트워크가 6축 네트워크에 비해 z축 오차가 작아 성능이 우수한 것을 확인할 수 있다. 이는 파라미터 수와 추정 성능이 비례하는 가장 일반적인 결과이다.

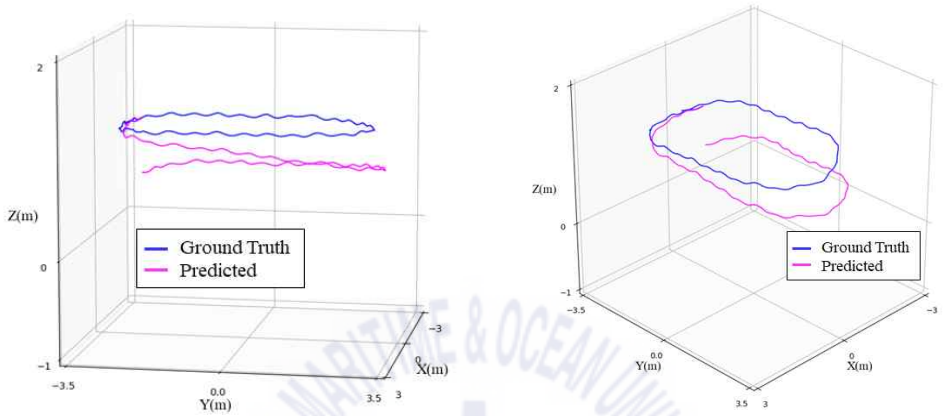
그림 4.7과 그림 4.8은 'data1/seq6'를 입력한 추정 결과이다. 이 결과에서는 각 네트워크의 Quantitative Evaluation 결과가 큰 차이를 보이지 않지만 6축 네트워크의 결과가 가장 우수하다. 네가지 추정궤적의 z축 오차는 큰 차이를 보이지 않으며 xy평면 오차에서 네트워크의 성능 순위가 결정된 것으로 판단된다. 이 결과에서 확인해야하는 점은 그림 4.8의 (b)이다. 6축 네트워크와 9-Axis IONet은 Inertial System Drift가 z축의 음수 방향으로 발생하지만 fast 9-Axis IONet은 양수 방향으로 발생한다. 이는 학습 과정에서 발생한 Overfitting 때문에 발생한 것으로 판단된다.

그림 4.9와 그림 4.10은 'data4/seq1'를 입력한 추정 결과이다. 이 결과에서는 대체로 6축 네트워크가 우수한 성능을 보이며 fast 9-Axis IONet은 RMSE가 1을 초과한 것을 확인할 수 있다. 각 네트워크의 z축 오차는 큰 차이를 보이지 않지만 9-Axis IONet은 xy평면에서 추정궤적이 이동된 것을 확인할 수 있다. 또한 fast 9-Axis IONet은 추정궤적이 xy평면에서도 이

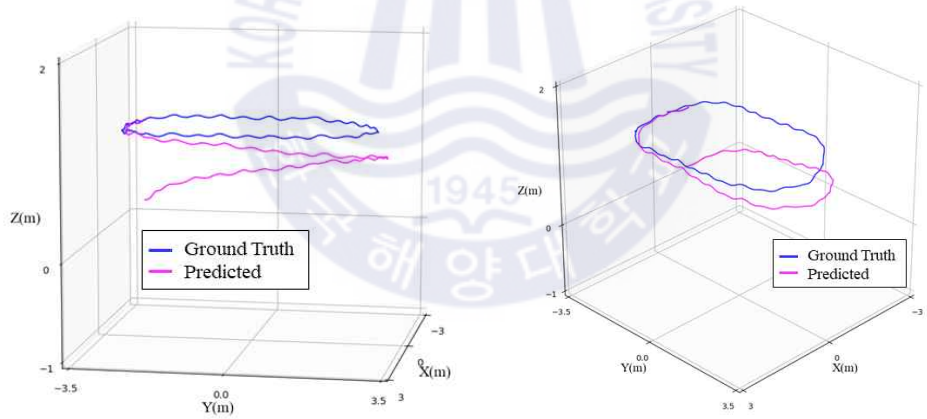
동되고 z축 양수 방향으로 발생하였다. 이는 fast 9-Axis IONet의 학습과정에서 Overfitting 문제가 심각하며, 학습 데이터가 모든 방향에 대한 가속도를 포함하지 못하고 특히 ‘data4/seq1’은 학습 데이터와의 이질적일 것으로 판단된다.

그림 4.11과 그림 4.12는 ‘data5/seq2’를 입력한 추정 결과이며 9-Axis IONet의 성능이 우수하다. 9축 네트워크는 6축 네트워크에 비해 z축 오차가 작으며 또한 xy평면 추정치가 Ground Truth와 매우 유사한 것을 확인할 수 있다.

네 가지 Qualitative Evaluation 결과, 6-Axis IONet과 fast 6-Axis IONet에 비해 9-Axis IONet과 fast 9-Axis IONet의 궤적 추정성능이 우수한 것을 확인하였다. 이는 추가적인 입력데이터가 Inertial System Drift 보정을 통해 자세추정 성능을 향상시키는 것이라 판단된다. 그러나 ‘data4/seq1’ 입력 시 fast 9-Axis IONet은 Overfitting으로 인해 이상치가 발견되었다.



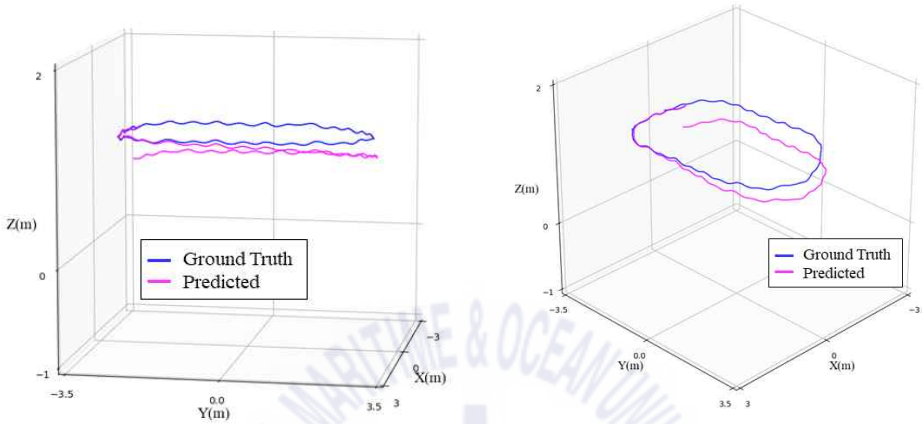
(a) Trajectory predicted by 6-Axis IONet



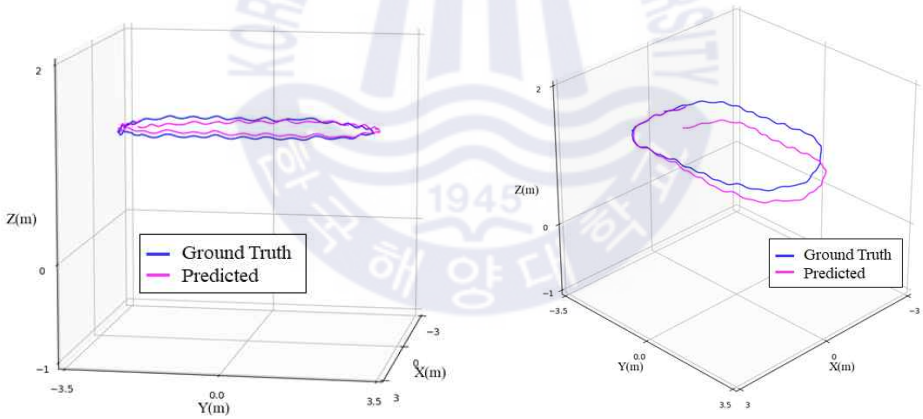
(b) Trajectory predicted by fast 6-Axis IONet

그림 4.5 'data1/seq2'를 6축 입력으로 네트워크의 위치 추정 결과

Fig. 4.5 Positioning result of the network using 'data1/seq2' as 6-axis input



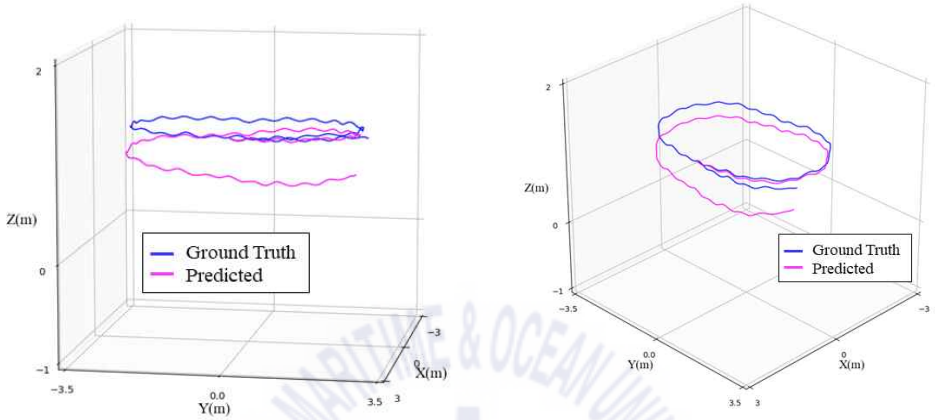
(a) Trajectory predicted by 9-Axis IONet



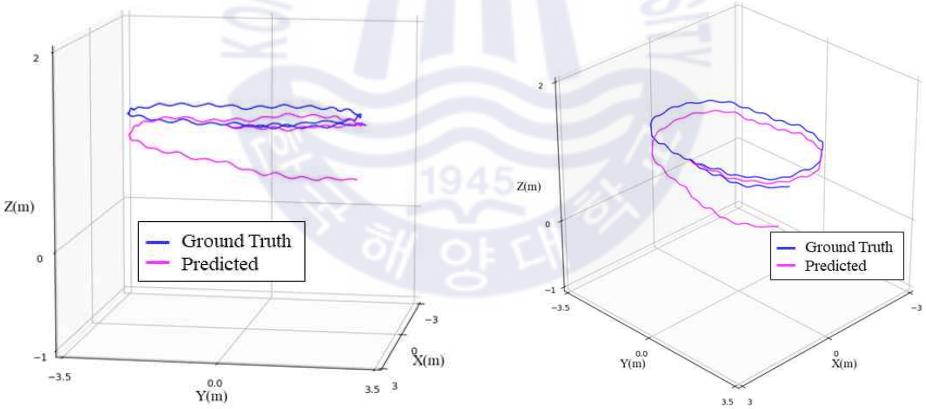
(b) Trajectory predicted by fast 9-Axis IONet

그림 4.6 'data1/seq2'를 9축 입력으로 네트워크의 위치 추정 결과

Fig. 4.6 Positioning result of the network using 'data1/seq2' as 9-axis input



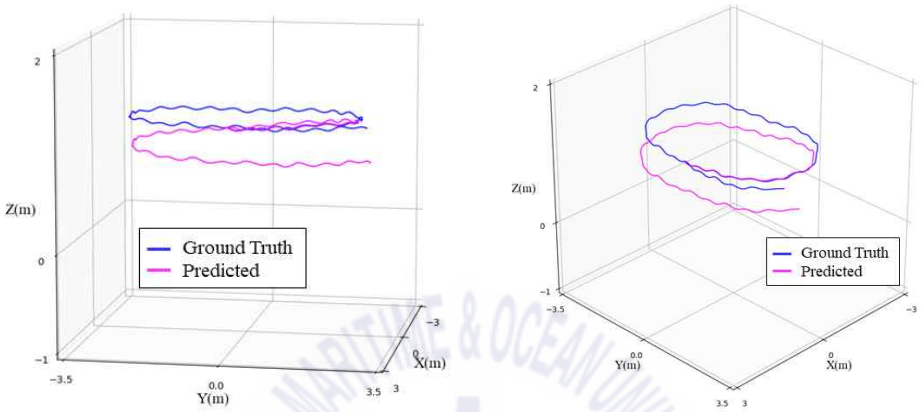
(a) Trajectory predicted by 6-Axis IONet



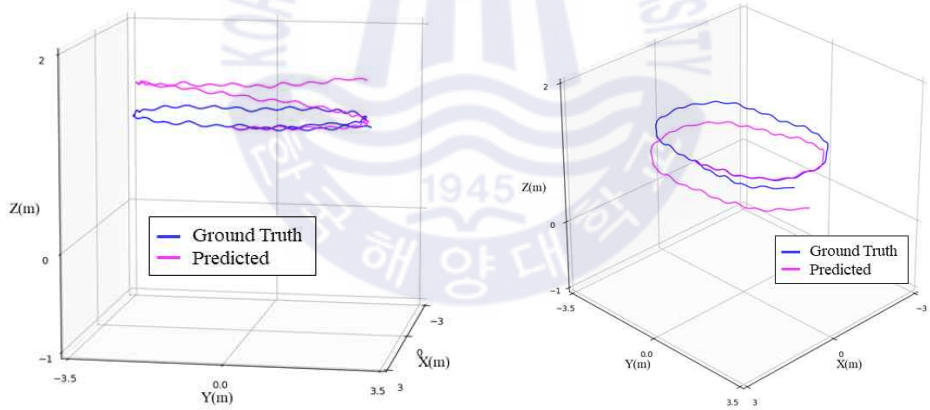
(b) Trajectory predicted by fast 6-Axis IONet

그림 4.7 'data1/seq6'를 6축 입력으로 네트워크의 위치 추정 결과

Fig. 4.7 Positioning result of the network using 'data1/seq6' as 6-axis input



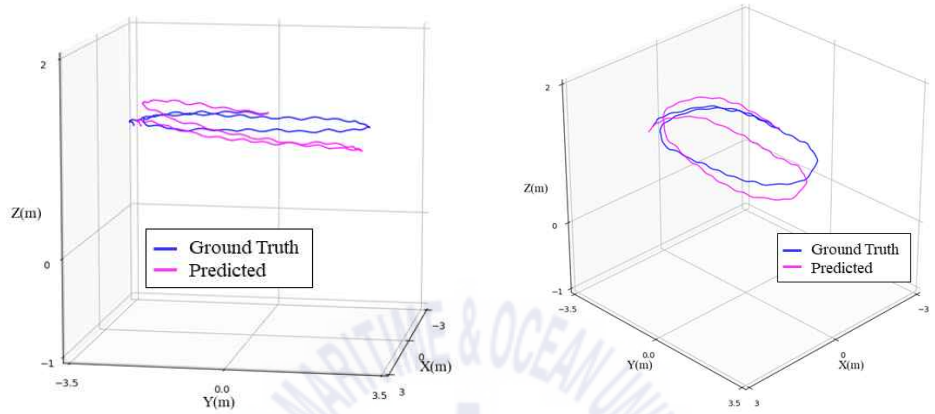
(a) Trajectory predicted by 9-Axis IONet



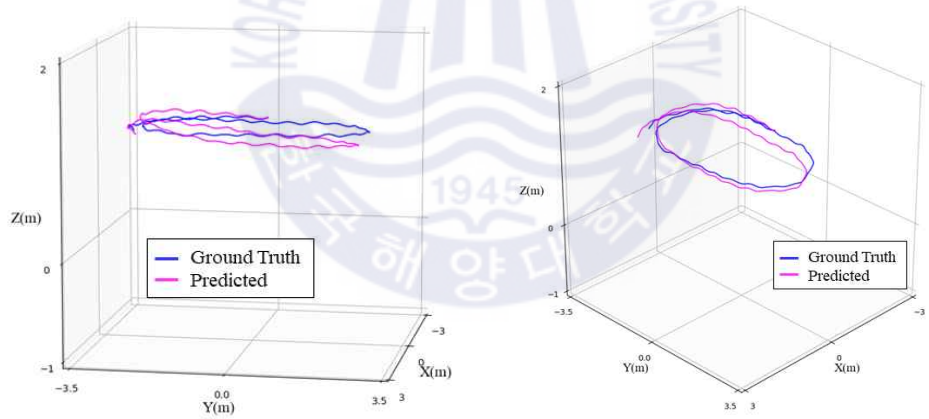
(b) Trajectory predicted by fast 9-Axis IONet

그림 4.8 'data1/seq6'를 9축 입력으로 네트워크의 위치 추정 결과

Fig. 4.8 Positioning result of the network using 'data1/seq6' as 9-axis input



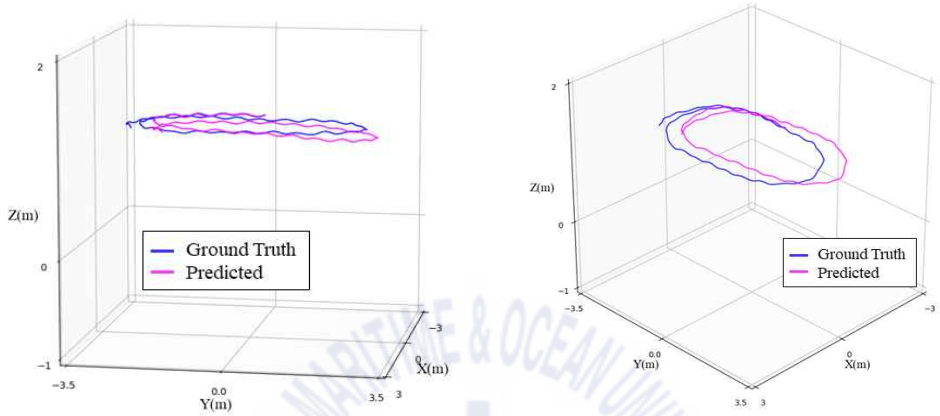
(a) Trajectory predicted by 6-Axis IONet



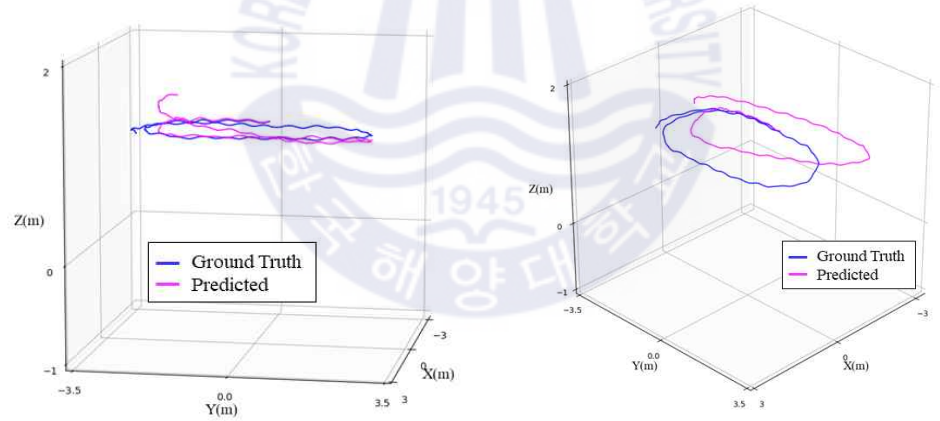
(b) Trajectory predicted by fast 6-Axis IONet

그림 4.9 'data4/seq1'를 6축 입력으로 네트워크의 위치 추정 결과

Fig. 4.9 Positioning result of the network using 'data4/seq1' as 6-axis input



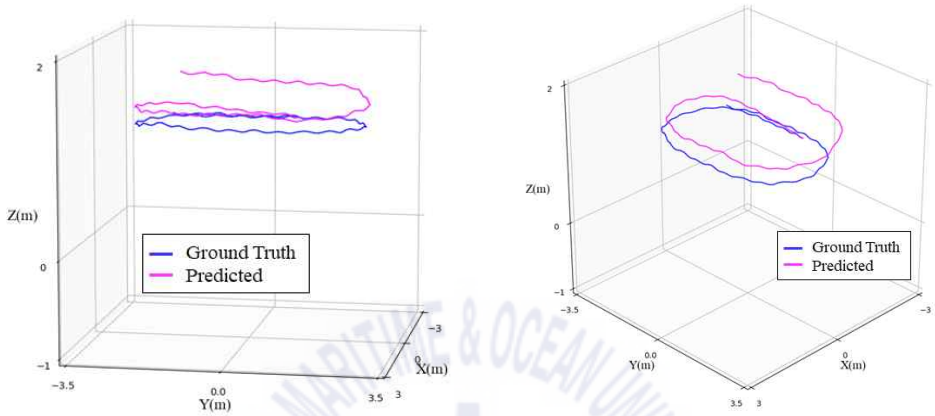
(a) Trajectory predicted by 9-Axis IONet



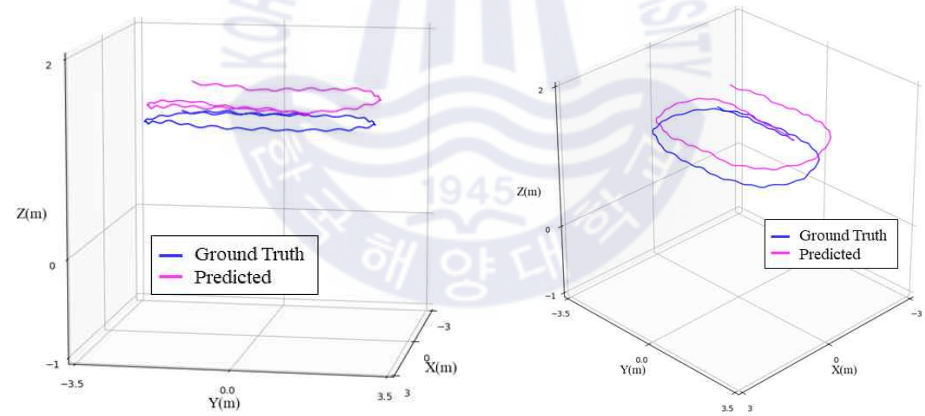
(b) Trajectory predicted by fast 9-Axis IONet

그림 4.10 'data4/seq1'를 9축 입력으로 네트워크의 위치 추정 결과

Fig. 4.10 Positioning result of the network using 'data4/seq1' as 9-axis input



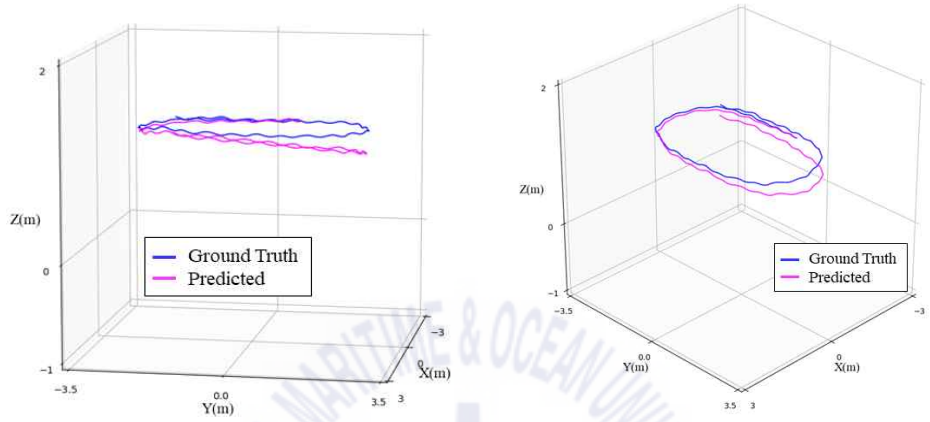
(a) Trajectory predicted by 6-Axis IONet



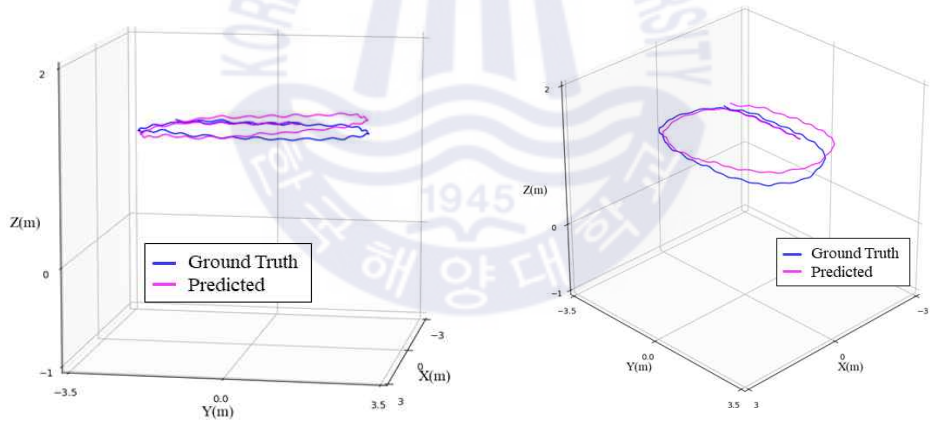
(b) Trajectory predicted by fast 6-Axis IONet

그림 4.11 'data5/seq2'를 6축 입력으로 네트워크의 위치 추정 결과

Fig. 4.11 Positioning result of the network using 'data5/seq2' as 6-axis input



(a) Trajectory predicted by 9-Axis IONet



(b) Trajectory predicted by fast 9-Axis IONet

그림 4.12 'data5/seq2'를 9축 입력으로 네트워크의 위치 추정 결과

Fig. 4.12 Positioning result of the network using 'data5/seq2' as 9-axis input

제 5 장 Conclusion

본 연구는 3차원 궤적 추정 시 발생하는 Inertial System Drift를 보정하기 위해 Modified IONet을 사용한 확장된 접근법을 제시하였다. 특히, 자세정보를 포함하는 기존 6-Axis IONet에서 사용되지 않던 중력가속도와 지자기를 입력데이터로 추가하여 성능을 향상시켰으며, 고사양 GPU가 구비되지 않은 환경에 적용하기 위해 파라미터 수 감소와 정확도 간 관계에 관한 연구를 진행하였다. Quantitative Evaluation 결과, 기존 6-Axis IONet과 비교하여 9-Axis IONet는 약 10% 우수한 성능을 보였으며 Qualitative Evaluation 결과, 추가적인 입력데이터가 궤적추적 안정성을 향상시키는 것을 확인하였다. 또한 순차적 데이터를 처리하는 LSTM Layer를 GRU Layer로 대체함으로써 파라미터 수를 감소시킨 fast 6-Axis IONet과 fast 9-Axis IONet이 6-Axis IONet에 비해 성능저하가 크지 않은 것을 확인하였다.

본 논문은 스마트폰에 사용되는 저비용 IMU를 사용하여 우수한 성능의 6-DOF Odometry Solution을 제시하였으며 향후 용접현장 혹은 철제 구조물과 같은 외란이 심한 환경에서 속도 및 자세의 추정에 대한 연구를 진행할 것이다. 또한 Convolutional Layer와 순차적 데이터를 처리하는 Layer를 모두 경량화된 네트워크로 대체하여 실시간 모바일 환경에서 사용될 수 있는 fast IONet에 대한 연구를 시도할 것이다.

참 고 문 헌

[1] K. R. Britting, “Inertial navigation systems analysis,” 1971.

[2] C. Jekeli, “Inertial navigation systems with geodetic applications,” Walter de Gruyter, 2012.

[3] D. Titterton, J. L. Weston and J. Weston, “Strapdown inertial navigation technology,” vol. 17, IET, 2004.

[4] Y. WU, X. HU, D. HU, T. LI and J. LIAN, “Strapdown inertial navigation system algorithms based on dual quaternions.” IEEE transactions on aerospace and electronic systems, vol. 41, no. 1, pp.110-132, 2005.

[5] Paul G. Savage, “Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms,” Journal of guidance, control, and dynamics, vol. 21, num. 1, pp. 19-28, 1998.

[6] C. Lu, H. Uchiyama, D. Thomas, A. Shimada and R. Taniguchi “Indoor positioning system based on chest-mounted IMU,” Sensors, vol. 19, num. 2, pp. 420, 2019.

[7] J. Nilsson, D. Zachariah, I. Skog and P. Händel, “Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging,” EURASIP Journal on Advances in Signal Processing, vol. 2013, num. 1, pp. 164, 2013.

[8] J.Nilsson, D. Zachariah, I. Skog and P. Händel, “Foot-mounted inertial navigation made easy,” International Conference on Indoor Positioning and Indoor Navigation(IPIN), 2014.

[9] H. P. Moravec, “Obstacle avoidance and navigation in the real world by a seeing robot rover,” no. STAN-CS-80-813, Stanford Univ CA Dept of Computer Science, 1980.

[10] L. Matthies and S. A. Shafer. “Error modeling in stereo navigation,” IEEE Journal on Robotics and Automation, vol. 3, no. 3, pp. 239-248, 1987.

[11] H. Zhang and C. Ye, “An indoor navigation aid for the visually impaired,” IEEE International Conference on Robotics and Biomimetics,

2016.

[12] H. Zhang and C. Ye. “An indoor wayfinding system based on geometric features aided graph SLAM for the visually impaired,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 25, no. 9, pp. 1592-1604, 2017.

[13] T. Zhou, M. Brown, N. Snavely and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[14] R. Li, S. Wang, Z. Long and D. Gu, “Undeepvo: Monocular visual odometry through unsupervised deep learning,” 2018 IEEE international conference on robotics and automation, 2018.

[15] D. Eigen, C. Puhrsch and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” Advances in neural information processing systems, 2014.

[16] C. Godard, O. M. Aodha and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[17] V. Indelmana, S. Williams, M. Kaess and F. Dellaert, “Information fusion in navigation systems via factor graph based incremental smoothing,” Robotics and Autonomous Systems, vol. 61, no. 8, pp. 721-738, 2013.

[18] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige and R. Siegwart, “Keyframe-based visual‐inertial odometry using nonlinear optimization,” The International Journal of Robotics Research, vol. 34, no. 3, pp. 314-334, 2015.

[19] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, “On-Manifold Preintegration for Real-Time Visual-Inertial Odometry,” IEEE Transactions on Robotics, vol. 33, no. 1, 2016.

[20] 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.

[21] G. Yang, L. Zhao, J. Mao and X. Liu, “Optimization-Based, Simplified Stereo Visual-Inertial Odometry with High-Accuracy Initialization,” IEEE

Access, vol. 7, pp. 39054-39068, 2019.

[22] R. Clark, S. Wang, H. Wen, A. Markham and N. Trigoni, “Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem,” Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[23] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham and N. Trigoni, “Selective sensor fusion for neural visual-inertial odometry,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[24] A. Solin, S. Cortes, E. Rahtu and J. Kannala, “Inertial odometry on handheld smartphones,” 21st International Conference on Information Fusion (FUSION), 2018.

[25] H. Yan, Q. Shan and Y. Furukawa, “RIDI: Robust IMU double integration,” Proceedings of the European Conference on Computer Vision(ECCV), 2018.

[26] C. Chen, X. Lu, A. Markham and N. Trigoni, “Ionet: Learning to cure the curse of drift in inertial odometry,” Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[27] S. M. Lima, J. Paulo, H. Uchiyama and R. Taniguchi. “End-to-End Learning Framework for IMU-Based 6-DOF Odometry,” Sensors, vol. 19, no. 17, pp. 3777, 2019.

[28] Y. Cheng, D. Wang and P. Zhou, “A survey of model compression and acceleration for deep neural networks,” arXiv preprint arXiv:1710.09282, 2017.

[29] Ju-Hyeon Seong, Seung-Hyun Lee, Kyoung-Kuk Yoon and Dong-Hoan Seo, “Ellipse coefficient map-based geomagnetic fingerprint considering azimuth angles,” Symmetry, vol. 11, no.5, pp. 708, 2019.

[30] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[31] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv preprint arXiv:1412.3555, 2014.

[32] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,”

IEEE transactions on Signal Processing vol. 45, no. 11, pp. 2673-2681, 1997.

[33] T. Parcollet, M. Morchid and G. Linarès, “A survey of quaternion neural networks,” Artificial Intelligence Review, pp. 1-26, 2019.

[34] A. Kendall, Y. Gal and R. Cipolla, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR), 2018.

[35] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham and N. Trigoni, “Oxiod: The dataset for deep inertial odometry,” arXiv preprint arXiv:1809.07491, 2018.