공학박사학위논문

Scale-aware Monocular Visual-Inertial Depth Estimation and Odometry using Self-supervised Learning

스케일 예측이 가능한 자가지도식 딥러닝 기반의
단안 시각-관성 깊이 추정 및 오도메트리 기법

2023년 8월

서울대학교 대학원

기계항공공학부

이 충 근

Scale-aware Monocular Visual-Inertial Depth Estimation and Odometry using Self-supervised Learning

스케일 예측이 가능한 자가지도식 딥러닝 기반의
단안 시각-관성 깊이 추정 및 오도메트리 기법

지도교수 김 현 진

이 논문을 공학박사 학위논문으로 제출함

2023년 5월

서울대학교 대학원

기계항공공학부

이 충 근

이충근의 공학박사 학위논문을 인준함

2023년 6월

위 원 장 : _____

부위원장 : _____

위 원 : _____

위 원 : _____

위 원 : _____

Scale-aware Monocular Visual-Inertial Depth Estimation and Odometry using Self-supervised Learning

A Dissertation

by

Lee, Chungkeun

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2023

Scale-aware Monocular Visual-Inertial Depth Estimation and Odometry using Self-supervised Learning

Lee, Chungkeun

Department of Mechanical and Aerospace Engineering

Seoul National University

APPROVED:

Youdan Kim, Chair, Ph.D.

H. Jin Kim, Ph.D.

Chan Gook Park, Ph.D.

Pyojin Kim, Ph.D.

Hyeonbeom Lee, Ph.D.

Abstract

Monocular Visual-Inertial Depth Estimation and Odometry with Scale Prediction using Self-Supervised Deep-Learning

Chungkeun Lee
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

This dissertation addresses deep-learning-based end-to-end self-supervised scale-aware depth estimation and odometry in the visual-inertial system. For real-world applications with a single monocular camera, scale ambiguity is an important issue. Because self-supervised data-driven approaches that do not require additional data containing scale information cannot avoid the scale ambiguity, state-of-the-art deep-learning-based methods address this issue by learning the scale information from additional sensor measurements. In that regard, inertial measurement unit (IMU) is a popular sensor for various mobile platforms due to its lightweight and inexpensiveness. However, unlike supervised learning which can learn the scale from the ground-truth information, learning the scale from IMU is challenging in self-supervised setting.

In this dissertation, deep-learning-based scale-aware self-supervised monocular visual-inertial depth estimation and odometry method is proposed. I focus on overcoming the scale ambiguity in the self-supervised setting. For the training data, the sequence of images and raw IMU measurements are utilized and neither ground-truth depth nor the stereo image pairs are provided. The proposed method works in an end-to-end manner and does not rely on the classical visual-inertial navigation to learn the scale. For that, I design the IMU preintegration loss which integrates IMU measurements and some regulation losses to predict the scale-aware ego-motion. Next, the network is proposed receiving IMU measurements as an input estimating the bias of the IMU and the gravity in the body coordinate

to perform IMU preintegration from raw IMU measurements. Lastly, a data augmentation technique is proposed, which is compatible with the visual-inertial system. The proposed algorithm is validated in comparison with state-of-the-art algorithms in the KITTI dataset and the indoor experiment, by demonstrating its comparable performance.

Keywords: Deep Learning, Depth Estimation, Visual-Inertial Odometry, Self-supervised Learning.

Student Number: 2014-22512

Table of Contents

Page

Abstract v

Table of Contents vii

List of Tables ix

List of Figures x

Chapter

1 Introduction 1

1.1 Literature Survey 3

1.2 Contributions 17

1.3 Outline 18

2 Preliminaries 19

2.1 Motion Stereo 20

2.2 Self-supervised Monocular Depth Estimation 25

2.3 IMU preintegration 27

3 Self-supervised Monocular Visual-Inertial Depth Estimation and Odometry . . . 28

3.1 Overview 28

3.2 Loss Function 30

3.3 Network Architecture 35

3.4 Data augmentation 41

4 Experimental Validation 44

4.1 Performance indices 44

4.2 Experimental Validation in the KITTI dataset 47

4.3 Experimental Validation in the Indoor Dataset 81

5 Conclusion 95

References 96
Abstract (*in Korean*) 106

List of Tables

1.1 The category of the monocular depth estimation using training data. . . . 3
1.2 The summary of the literature survey of monocular depth estimation. . . . 12
1.3 The summary of the literature survey of deep-learning-based visual-inertial methods. . . . 16

3.1 The detail of the inertial encoder. . . . 37
3.2 The detail of the fusion part. . . . 38
3.3 The detail of the decoder. . . . 39

4.1 The depth performance for the ablation study about preintegration loss function. . . . 50
4.2 The scale prediction results from the depth map 55
4.3 Depth estimation performance in KITTI Eigen split (80m cap). . . . 57
4.4 Pose estimation performance in KITTI odometry dataset. . . . 66
4.5 The scale prediction results from the pose 69
4.6 The runtime of the proposed method during inference. . . . 78
4.7 The list of specifications of the device to collect the data. . . . 81

List of Figures

2.1 Pinhole camera model 20
2.2 Epipolar geometry for the stereo system 22
2.3 The overview of the unsupervised monocular depth estimation. 25

3.1 The overview of the proposed algorithm. 29
3.2 The overview of the proposed odometry network. 36
3.3 The flowchart of the inertial encoder. 37
3.4 The flowchart of the feature fusion part. 38

4.1 The histogram of the scale of the ablation study about preintegration loss function. 51
4.2 Top-down trajectory for the ablation study about the bias regulation loss . 52
4.3 The bias prediction result for the ablation study about the bias regulation loss (odometry 09) 53
4.4 The bias prediction result for the ablation study about the bias regulation loss (odometry 10) 54
4.5 The histogram of the scale from the predicted depth. 56
4.6 The depth prediction result on the Eigen split for qualitative comparison. . 64
4.7 The histogram of the predicted scales from the pose in odometry 09. 67
4.8 The histogram of the predicted scales from the pose in odometry 10. 68
4.9 The top-down view of the estimated trajectory of the KITTI odometry dataset. 72
4.10 Relative pose estimation of linear motion at the beginning of the driving in odometry 09. 73
4.11 Relative pose estimation of angular motion in odometry 09. 74
4.12 Relative pose estimation of linear motion in odometry 09. 75

4.13 Relative pose estimation of angular motion in odometry 10. 76
4.14 Relative pose estimation of linear motion in odometry 10. 77
4.15 The velocity prediction result using the proposed method. 79
4.16 The gravity prediction result using the proposed method. 80
4.17 The device to collect the data and example of collected data of the indoor dataset. 81
4.18 The reference trajectory of the indoor dataset. 83
4.19 The predicted trajectory in the indoor dataset (whole). 85
4.20 The predicted trajectory in the indoor dataset (first part: B2, B2→B3). . . 86
4.21 The predicted trajectory in the indoor dataset (second part: B3, B3→B2). 87
4.22 The gravity prediction result in the indoor dataset. 88
4.23 The depth prediction result on the indoor dataset. 94

1 Introduction

Ego-motion estimation and 3d reconstruction with a monocular camera have broad applicability because a monocular camera is inexpensive and lightweight. Especially, data-driven monocular depth estimation has received attention because they give a dense depth map from an image and ground-truth depth in a supervised manner by training a deep neural network [1, 2, 3].

To avoid the cost of collecting the ground-truth depth with an additional device, self-supervised monocular depth estimation has been proposed. State-of-the-art self-supervised methods jointly train the depth map and ego-motion for structure from motion during the training step, so it requires the sequences of monocular images during the training step [4, 5, 6, 7].

Nevertheless, self-supervised methods have scale ambiguity originating from the nature of the monocular camera because they have no criteria about the scale information unlike supervised methods with ground-truth depth information. In general, an additional sensor is introduced to address the scale ambiguity issue. In that regard, IMU is a popular sensor for various mobile platforms because of its lightweight and inexpensiveness. In classical

vision, state-of-the-art visual-inertial methods predict the ego-motion with scale prediction using the sequences of monocular images and IMU measurements [8, 9].

Because the deep-learning-based approach has a capability of predicting a dense depth map from a *single* image, some researchers have incorporated IMU into the deep-learning like the classical visual navigation literature [10, 11, 12, 13, 14, 15]. However, learning the scale from IMU is challenging in self-supervised setting. To overcome this issue, the training concept to learn the scale from the classical visual-inertial navigation was introduced [16, 17, 18], but it highly relies on the performance of the classical navigation.

In this dissertation, I focus on overcoming the scale ambiguity in the self-supervised setting. For the training data, the sequence of images and raw IMU measurements are utilized and neither ground-truth depth nor the stereo image pairs are provided. The proposed method works in an end-to-end manner, and does not rely on the classical visual-inertial navigation to learn the scale. For that, I design the IMU preintegration loss which integrates IMU measurements to predict the scale-aware ego-motion. The proposed algorithm is validated in comparison with state-of-the-art algorithms in the KITTI dataset and the indoor experiment, by demonstrating its comparable performance.

1.1 Literature Survey

In this section, I introduce a literature survey about deep-learning-based monocular depth estimation, classical visual-inertial navigation and deep-learning-based visual-inertial methods.

1.1.1 Monocular depth estimation

Monocular depth estimation aims to estimate the dense depth map from a single RGB image. I categorize monocular depth estimation based on the training data. The supervised method utilizes the ground-truth depth to train the network. The unsupervised method receives all other data except the ground-truth depth. The self-supervised method utilizes sequences of images. In this dissertation, the self-supervised method is distinguished as a separate category from the unsupervised method even if the self-supervised method is included in the unsupervised one. Table 1.1 summarizes the category of monocular depth estimation based on the training data. Here, the self-supervised method with monocular sequences only needs the monocular images for training. Thus, it has the advantage that the device setup of the training and inference is same.

Method	**Input**	**Output**	**Train Data**
Advantages/Disadvantages			
Supervised	single image	depth map	ground-truth dense depth
Supervised methods can predict the accurate depth map in comparison with others. Supervised methods require the depth ground-truth for training the network.			
Unsupervised	single image	depth map	anything except ground-truth (generally stereo)
Self-supervised	single image consecutive images	depth map +ego-motion	sequences of monocular images (or those of stereo pairs)
For monocular, the network can be trained using the data collected during the inference. For monocular, scale ambiguity issue exists, so no scale is predicted.			

Table 1.1: The category of the monocular depth estimation using training data.

Supervised monocular depth estimation

The supervised monocular depth estimation aims to construct the network predicting the depth map from a single RGB image and train this network from ground-truth depth information. Commonly, supervised methods construct the convolutional neural network emitting the dense map as an input of the single RGB image. The regression problem is formulated to train the network. The network is trained by minimizing the difference between the estimated depth map and the ground truth depth map.

Reference [1] is the first approach to solving the monocular depth estimation with the regression problem, to the author's best knowledge. They proposed the course-fine network architecture; the course network estimates the smoothed depth map and the fine network refines the smoothed depth map to get the final depth map. Since the global scale is variable across the dataset, they design the scale-invariant loss function to express the difference with no scale information as

$$\mathbf{L} = \frac{1}{n}\sum_{i}(\log d_i - \log \hat{d}_i)^2 - \frac{1}{n^2}(\sum_{i}\log d_i - \log \hat{d}_i)^2 \tag{1.1}$$

where $d_i, \hat{d}_i$ is the ground truth depth and predicted depth.

In the supervised depth estimation research field, many researchers have proposed to increase the depth estimation performance. Most approaches focus on the enhancement of the network architecture, the design of additional constraints about the depth, the novel problem formulation of the regression problem or the application in the real world.

Reference [2] adopted the novel network architecture to the depth estimation problem. They adopted the fully convolutional neural network as in [19] with the residual neural network as in [20]. In addition, they adopted the reversed Huber norm $\mathfrak{B}$ as the loss

function to get better performance than the L2 norm as

$$\mathfrak{B}(d,\hat{d}) = \begin{cases} \left|d-\hat{d}\right| & \text{if } \left|d-\hat{d}\right| \geq c \\ \frac{\left(d-\hat{d}\right)+c^2}{2c} & \text{otherwise} \end{cases} \tag{1.2}$$

Reference [3] adopted the ordinal regression into the deep-learning-based depth estimation formulation. They performed space-increasing discretization(SID) in log-scale as an expansion of the uniform discretization to adapt to the distribution of each depth value. Then, they design the ordinal loss function with a fully differentiable form for back-propagation. They proposed the network with a single encoder and multiple decoders to understand the variable scene selecting the final depth by ordinal regression.

Reference [21] proposed the separate depth estimation of the category and the objects. They segmented the image and performed the depth estimation for each category. Therefore, they explicitly considered the characteristic of each object.

Reference [22] proposed the decomposed formulation of the depth map based on the relative depth and estimated each decomposed part. The full-resolution depth map D_n can be decomposed as

$$\log D_n = \log U^n(D_0) + \sum_{i=1}^{n} \log U^{n-i}(F_i) \tag{1.3}$$

where $U(\bullet)$ is the upsampling operation, $F_n = D_n \oslash U(D_{n-1})$ is the fine detail map defined as the elemental-wise division ($\oslash$) from D_n to $U(D_{n-1})$.

Unsupervised monocular depth estimation with stereo cue

Unsupervised monocular depth estimation aims to train the depth estimation network with additional data except for ground-truth depth. In this section, I handle unsupervised approaches which train the network from stereo images. Remark that the self-supervised method such as [4] is differently categorized in this dissertation.

Reference [23] proposed the reconstruction error, so-called reconstruction loss or photometric consistency loss, to train the network from a stereo image pair. They predicted the dense depth map of a left image by the convolutional neural network. During the training step, the left image was warped into the right camera frame using the predicted depth map and epipolar geometry, and the depth was trained to minimize the difference between the warped image and the right image.

Reference [24] extended the [23] into three parts. Firstly, they adopted the bilinear sampling proposed in [25] to obtain a fully differentiable warped image when calculating gradient of the photometric consistency loss. Second, they adopted the loss function from the image restoration proposed in [26] to enhance the depth prediction performance using photometric consistency loss. Lastly, they simultaneously estimated both the left and right disparity using the left image and design the consistency between the two disparities. They showed the best performance when they were proposed.

Reference [27] formulated the unsupervised monocular depth estimation problem with stereo cue into the stereo matching problem with two steps. Firstly, the view synthesis network generates the synthesized right image from the left image like [28]. Then, the stereo matching network estimates the disparity between the left image and the generated right image like [29].

Self-supervised monocular depth estimation with image sequences

In a strict definition, the self-supervised method is categorized if additional information is predicted to predict the depth. However, in this dissertation, I only handle the ego-motion case.

Reference [4] firstly proposed a framework of the self-supervised monocular depth estimation with no additional train data. They estimated the depth from the target image using one network and the ego-motion from the target and nearby images using another network. During the training step, both networks are jointly trained using the photometric consistency loss like the unsupervised method with the stereo cue. In addition, they pro-

posed the explainability mask predicted from another network to prevent optimizing the difference of the irregular pixels. When generating view synthesis from the sequential images, the existence of the irregular pixel is inevitable due to the moving object or occlusion. Therefore, they designed the photometric consistency loss masked by the explainability as

$$\mathbf{L} = \sum_{p} \hat{E}(p) \left| I(p) - \hat{I}(p) \right| \tag{1.4}$$

where $\hat{E}$ is explainability mask, I is the target image and $\hat{I}$ is the warped image. Here, they also designed the regulation loss of the explainability to avoid the trivial solution of the explainability mask.

Reference [30] proposed the training concept which contains both unsupervised and self-unsupervised methods when sequences of stereo pairs are provided as training data. For a sequence of the stereo pairs, they warp the right image based on the stereo setup to generate the photometric consistency between the left and right image. Simultaneously, they warp the left image from the next frame based on the predicted ego-motion like [4]. Thus, they could train the network with both spatial and temporal information.

Reference [31] proposed the 3D ICP loss to constrain the structured point cloud between two images. From the depth prediction result, each point cloud is calculated, and one point cloud is warped using the predicted ego-motion. The iterative closest point(ICP) method matches the warped point cloud and predicted point cloud generating the rigid body motion T_t' and residuals r_t. Since two point clouds should be equal, 3D ICP loss is given as

$$\mathbf{L} = \|T_t' - I\|_1 + \|r_t\|_1 \tag{1.5}$$

where I is the identity matrix.

Reference [5] adopted the optical flow estimation network to design additional constraints about the optical flow in the self-supervised monocular depth estimation. In addition to other self-supervised methods, they predicted rigid flow from the estimated depth

and ego-motion and refined it to predict the optical flow.

Reference [7] proposed appearance loss with per pixel minimum reprojection loss and auto masking to enhance the depth estimation performance. For the source image I_t, the previous image I_{t-1} and the next image I_{t+1} are provided as the target images. For each reprojection loss from the previous and next image respectively, the final reprojection loss was proposed as the minimum value between each reprojection loss for each pixel; thus, the proposed loss rejects high-error pixels which are likely to be occluded. In addition, the auto-mask μ is given as

$$\mu = \begin{cases} 1 & \text{if } pe(I_t, I'_{t'->t}) \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

where $I'_{t'->t}$ is the warped image, $pe(\bullet, \bullet)$ is the distance function between two images for reprojection loss.

Remark that self-supervised depth estimation methods using sequences of monocular images simultaneously learn the depth map and the ego-motion, and they suffer the scale ambiguity issue. They could not learn the real-world scale because of the physical problem of the monocular camera. On the other hand, the supervised methods learn the real-world scale from the ground-truth depth information, and the unsupervised or self-supervised methods using stereo images learn the real-world scale from the length of the stereo camera.

Other monocular depth estimation methods

Reference [32] proposed the depth estimation when the stereo image and sparse ground-truth depth are given as the train data, which is called the semi-supervised method. They combined the photometric consistency loss function from the stereo images, and the supervised loss only from sparse points.

Reference [33] additionally received the velocity cue to predict the real-world scale from the monocular self-supervised setup. From the velocity cue, velocity supervision loss was

proposed as in

$$\mathbb{L} = |\|t\| - |v|\Delta T| \tag{1.7}$$

where t is the translation part of the relative pose, v is the ground truth velocity and ΔT is the time difference between two frames. Due to the real-world scale information of the velocity, the network learned the real-world scale. However, according to the theory of relativity, sensing the velocity in the robot is hard.

Recurrent neural network

For the mobile robot, the monocular images are provided sequentially during the motion. Considering the temporal information is one of the solutions to enhance the depth estimation performance. In that idea, the recurrent neural network has been selected to learn temporal information.

Reference [34] adopted the recurrent neural network containing convolutional LSTM for the supervised monocular depth estimation. They showed this recurrent approach helps to increase the depth estimation performance.

Reference [6] expanded the recurrent approach into the self-supervised depth estimation. They constructed the network based on the convolutional LSTM to apply the recurrent effect with conserving the convolutional effect. During the training step, they calculated the photometric consistency loss along both forward and backward directions.

Generative adversarial network

A generative adversarial network is one of the famous training concepts, the generator generates fake information which looks real and the discriminative model distinguishes the fake information [35]. Both networks are adversarial trained for the generative model to generate indistinguishable information.

In the monocular depth estimation, the generative model predicts the depth map from

a single image and generates the right image based on the predicted depth map. The discriminative model distinguishes the warped image and the right image. This framework was proposed with a depth generation model using stereo data in [36] and both depth and pose generation models using stereo data in [37]. For the self-supervised method, reference [38] performed using monocular sequences and [39] performed using stereo sequences with stack GAN proposed in [40].

Reference [41] adopted both the generative adversarial network and the recurrent neural network. For the image sequence, the consecutive images are coded by the optical flow and refined by the LSTM layers. The depth network predicts the depth map from the refined code, and the generative adversarial network is designed similarly to other methods.

Reference [42] proposed the framework for unlabeled images using the generative adversarial network. They designed one generative model which predicts a depth map and two discriminative networks. One discriminative network distinguishes the ground truth depth and the generated depth from the generative model. The other discriminative network distinguishes the depth map of the paired data is fake or not.

Real-time approach

Some methods focus on the real-time application on the mobile platform. In general, the onboard computer for the mobile platform has approximately ten-percent computational power compared with the desktop computer. Thus, common methods cannot work in real-time on the mobile platform.

Reference [43] proposed the PyD-Net which is enough lightweight for inference on the mobile platform with only CPU equipped computer. They validated the PyD-Net with unsupervised learning following [24]; they showed unsupervised depth estimation works in the lightweight network with reasonable performance. Similarly, [44] proposed MiniNet with the self-supervised approach.

Reference [45] proposed the U-net [46] style network architecture with mobileNet[47] to achieve real-time performance during the inference step. They provided about 175Hz depth

with small performance degradation.

Summary

Deep-learning-based depth estimation has been researched with various types of training data. Table 1.2 is provided as a summary of the survey. The self-supervise method with monocular sequences has been interesting because additional training data are not necessary and the network for odometry is also trained.

	Output	Train Data	Contribution
[1]	-	image + depth	First approach of deep-learning-based depth estimation
[2]	-	image + depth	Adoption of fully convolutional neural network
[3]	-	image + depth	Adoption of the ordinal regression. Best performance when proposed.
[21]	-	image + depth	Semantic-aware depth estimation.
[22]	-	image + depth	Decomposing the depth component based on the relative depth.
[23]	-	stereo	First approach of photometric consistency loss.
[24]	-	stereo	Design of left-right disparity consistency. Best performance when proposed within unsupervised methods with stereo cues.
[27]	-	stereo	Reformulating the problem as stereo matching.
[4]	pose	mono seq.	First approach of the self-supervised method with monocular sequences.
[30]	pose	stereo seq.	Approach of the self-supervised method with stereo sequences.
[31]	pose	mono seq.	Design of 3D ICP loss.
[5]	pose + optical flow	mono seq.	Prediction of the optical flow in the self-supervised problem.
[7]	pose	mono/stereo seq.	Design of appearance loss for the self-supervised method.
[32]	-	stereo + sparse depth	Design the training concept with sparse depth.
[33]	pose	mono seq. + velocity	Overcoming the scale ambiguity using the velocity information.
[34]	-	image + depth	Adoption of the recurrent neural network in the supervised setup.
[6]	pose	mono/stereo seq.	Adoption of the recurrent neural network in the self-supervised setup.
[36]	-	stereo	Adoption of GAN in the unsupervised setup with the stereo cue.
[37]	pose	stereo	Adoption of GAN using stereo data like self-supervised framework with pose generation.
[38]	pose	mono seq.	Adoption of GAN in the self-supervised setup.
[39]	pose	stereo seq.	Adoption of GAN in the self-supervised setup.
[41]	pose	mono seq.	Adoption of GAN and RNN in the self-supervised setup.
[42]	-	image + partially labelled depth	Adoption of GAN when depth ground-truth is partially provided.

Table 1.2: The summary of the literature survey of monocular depth estimation. For the column of *output*, the depth map is not described.

1.1.2 Classical visual-inertial navigation

Classical visual-inertial navigation, including odometry and simultaneous localization and mapping (SLAM), has been widely researched.

With the long history of classical visual-inertial navigation, some survey papers provide a good review of its long history [48, 49, 50]. Thus, in this dissertation, I only mention the common characteristics and a few state-of-the-art monocular visual-inertial navigation methods.

Classical navigation methods can be categorized into two types based on the problem formulation: filtering-based and optimization-based methods. Filtering-based methods design the filter which expresses the state and measurements of the robot, respectively. Then, the designed filter is operated during inference [51, 52, 53, 54]. Optimization-based methods construct the objective function from the camera geometry and perform non-linear optimization [8, 9, 55].

Reference [51] is one of the filtering-based visual-inertial odometry methods. They set the filter state by concatenating the robot state, IMU bias, extrinsic calibration parameter between the image and IMU, feature points, and the distance of each feature point as

$$x = (r, v, q, b_f, b_\omega, c, z, \mu_0, \cdots, \mu_N, \rho_0, \cdots, \rho_N) \tag{1.8}$$

where (r, v, q) is the position, velocity or attitude of IMU expressed in IMU coordinate, (b_f, b_ω) is the biases of IMU sensor, (c, z) is the translation and rotation part of the extrinsic parameter, and (μ_i, ρ_i) is the feature point and its distance.

Then, the state propagation model could be expressed from the robot dynamics with the IMU measurements for (r, v, q), the random acceleration assumption for (b_f, b_ω, c, z) and the pinhole camera model with geometric consistency for (μ_i, ρ_i). They update the filter using the extended Kalman filter.

Reference [9] is one of the optimization-based visual-inertial odometry methods. From the IMU measurements, the preintegration is performed to get the ego-motion using IMU

information. From the image, the feature points are tracked, and then bundle adjustment is performed to obtain the ego-motion from image information. During the optimization step, the IMU residual from the IMU preintegration and the image residual from the geometric consistency are minimized.

For the optimization-based methods, IMU initialization is required at the beginning, and the vehicle/robot should generate the acceleration and the tilting motion in the roll and pitch directions for the monocular case. In automobile environments, IMU initialization may fail, because the dominant motion of the car is yaw direction.

1.1.3 Deep-learning-based visual-inertial methods

Some researchers tried the visual-inertial approach in the deep-learning-based methods. In this section, I introduce deep-learning-based visual-inertial methods.

Reference [10] proposed the supervised visual-inertial odometry, which is the first approach of the visual-inertial method to the best of the author's knowledge. They designed the visual-inertial network with a combination of the convolutional neural network for the images and the recurrent neural network for the IMU measurements. The pose was predicted from the fusion of each network.

Reference [11] proposed the novel network architecture for supervised visual-inertial odometry focusing on robust prediction. They designed the feature fusion by selecting either visual or inertial features to handle the unexpected behavior of either sensor. They showed robust ego-motion estimation even if the sensor emits irregular data.

Reference [15] proposed the RGB-D visual-inertial odometry framework using deep-learning-based approaches. They stacked several layers which generate the error Jacobian of the previous level to predict the transform. This method, however, needs the depth information.

Reference [12] proposed the self-supervised visual-inertial odometry from stereo sequences. The flow network estimates the optical flow from the images, and the IMU preintegration network estimates the ego-motion from the IMU measurements; then, the VI fusion

network estimates the final ego-motion from the optical flow and the ego-motion from the IMU network. During the training step, they utilize the cue from the stereo images to learn each network.

References [16], [17] formulated the deep-learning-based visual-inertial method into the combination of the classical visual-inertial odometry with sparse depth estimation and deep-learning-based depth completion. Due to the sparse depth from the classical visual-inertial odometry which contains the real-world scale, the estimated depth and ego-motion contain the real-world scale. Those works, nonetheless, assume the navigation system always provides a sparse depth for depth prediction, so they cannot predict the depth from a single image.

Reference [18] proposed the transfer learning of the depth estimation network by the teaching of the classical visual-inertial method to learn the scale of the new environment. The depth network S is trained similarly to the self-supervised method, but the ego-motion is provided from the SLAM algorithm instead of training and predicting ego-motion. For transfer learning, distillation loss is introduced as the difference between two depth maps with unit scale to learn the relative depth from the pre-trained network T. In addition, scale consistency loss is introduced to conserve the scale between frames. They, nevertheless, learn the real-world scale from the output of another SLAM method. Since these methods require classical navigation during the training step, whenever the classical method fails, so do they.

Reference [13] proposed self-supervised visual-inertial depth estimation and odometry with the generative adversarial network. They adopted the network architecture from [11] for the generative model. Then, they designed a discriminative model with the convolutional neural network to train the network. They showed good odometry performance, but they have no real-world scale like the self-supervised monocular depth estimation.

Reference [14] proposed unsupervised visual-inertial depth estimation and odometry with the intra-inter optimization technique. Intra-window optimization is performed as the unsupervised monocular depth estimation. For inter-window optimization, the relative

pose is integrated along the entire sequence and additionally optimized using geometric and trajectory consistency. Intra-inter optimization helps to conserve the scale information across the whole trajectory but cannot predict the metric scale.

Methods	**Output**	**Training**	**Limitation**
VINet [10] Chen et al. [11]	pose	sup(pose)	Ground-truth pose is required for training.
VIOLearner [15]	pose	unsup	RGB-D image is required, so the dense depth map is required as an input.
DeepVIO [12]	depth+pose	selfsup (stereo)	Stereo data is required for training.
Wong et al. [17] Sartipi et al. [16]	depth+pose	teaching+ semisup	Classical VIO should be operated during training and inference.
SelfTune [18]	depth	teaching+ selfsup(mono)	Classical VIO is necessary for training.
SelfVIO [13] Wei et al.[14]	depth+pose	selfsup(mono)	No scale is predicted. The relative depth and ego-motion are given.
Proposed	depth+pose	selfsup(mono)	

Table 1.3: The summary of the literature survey of deep-learning-based visual-inertial methods. For **training** column, *sup* is the supervised method, *unsup* is the unsupervised method, *selfsup(·)* is the self-supervised method with · data, and *teaching* is the teaching of the classical visual-inertial navigation.

Table 1.3 summarizes the literature survey of deep-learning-based visual-inertial methods. No related work achieves the unsupervised learning of the depth or pose containing real-world scale with no teaching from another method. [10, 11] require the ground-truth pose information during the training step, [15] needs the depth information as an input data, [12] requires the sequence of the stereo images, [17, 16, 18] require the result from the classical visual-inertial navigation and [13, 14] contain no real-world scale information.

1.2 Contributions

The main contribution of the proposed method is overcoming the scale ambiguity of the self-supervised monocular depth estimation with IMU measurements with the end-to-end concept. State-of-the-art self-supervised methods using monocular sequences suffer the scale ambiguity issue. To solve the scale ambiguity issue, the network requires a supervisory signal, stereo signal or the teaching of classical visual-inertial navigation method to learn real-world scale information. The proposed method, nevertheless, needs neither supervisory nor stereo signal and is learned end-to-end.

Moreover, the proposed method is validated in the KITTI [56] dataset in comparison with state-of-the-art deep-learning-based methods and classical navigation methods. Additionally, another validation is performed on the indoor dataset collected at the indoor underground parking lots, to show that the proposed method is not overfitted in the KITTI dataset. In comparison, the proposed algorithm is shown to have comparable performance compared with other methods.

1.3 Outline

In this dissertation, I handle deep-learning-based scale-aware self-supervised monocular depth estimation and odometry with visual-inertial data. In chapter 2, I provide the preliminary as three sections: motion stereo, self-supervised monocular depth estimation, and IMU preintegration. Then, the proposed method is handled in chapter 3. Starting from the overview, I describe the designed loss function, the designed network, and the augmentation detail. Validation result follows in chapter 4.

2

Preliminaries

The proposed method originated from self-supervised monocular depth estimation and was upgraded to learn the scale by integrating IMU measurements.

In this chapter, I handle three topics as preliminaries. The first topic is motion stereo. In this section, I describe how to extract the depth information from a monocular camera with motion, so-called motion stereo, based on the camera geometry.

The second topic is deep-learning-based self-supervised monocular depth estimation. In this section, I describe the common method of state-of-the-art deep-learning-based self-supervised monocular depth estimation.

The last topic is IMU preintegration. IMU preintegration is the step of integrating IMU measurements to estimate the ego-motion of the robot. Some classical visual-inertial odometry and SLAM perform IMU preintegration, and I describe about those preintegration techniques.

2.1 Motion Stereo

2.1.1 Camera Geometry

Pinhole Camera Model

For the monocular camera, the pinhole camera model as fig. 2.1 has been widely utilized. For the point P in the real world (x_p, y_p, z_p), the camera captures the point $Q = (x_q, y_q)$ as

$$\begin{pmatrix} x_q \\ y_q \end{pmatrix} = -\frac{f}{z_p} \begin{pmatrix} x_p \\ y_p \end{pmatrix} \tag{2.1}$$

where f is the focal length of the camera.

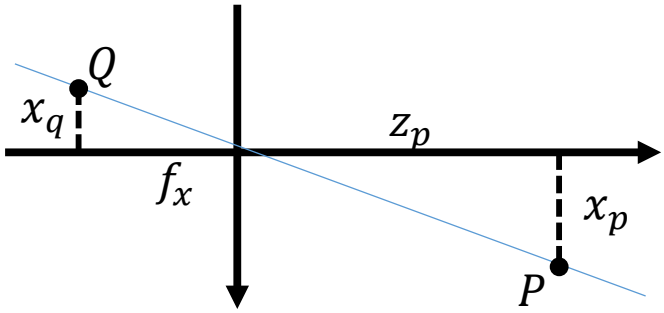


Figure 2.1: Pinhole camera model

For convenience, the pixel coordinate is defined in the normalized image coordinate, and the virtual coordinate with focal length is one, with the origin at the left-top point. The point in the pixel coordinate $U = (u, v)$ is defined as

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z_p} K \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} \tag{2.2}$$

where K is the intrinsic parameter defined as

$$K = \begin{pmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

where f_x, f_y is the focal lengths, γ is skew coefficient, and c_x, c_y is the principal point.

The camera captures the 2-dimensional normalized point, and the depth z_p cannot be known from the single measurement of the camera. One of the solutions is the stereo system, two monocular cameras with known distance are simultaneously equipped and capture images.

Epipolar Geometry

For the stereo system, the epipolar geometry is constructed as in fig. 2.2. From two cameras L and r with the rigid body motion (R, t), the epipolar line $l_\bullet$ is given as the intersection of the projected point of another camera denoted as $e_\bullet$ and the projected point on the normalized image plane of the target point $P_\bullet$ denoted as $p_\bullet$. Then, the epipolar plane containing two epipolar lines can be uniquely defined, and the target point should be on the epipolar plane.

Here, according to the epipolar geometry, the essential matrix $E = R\,[t]_X$ always exists holding,

$$p_L^T E p_r = 0 \tag{2.4}$$

where (R, t) is the rigid body motion between two cameras and $[\bullet]_X$ is the matrix representation of the cross product.

In pixel coordinate, equation (2.4) could be expressed as

$$U_L^T F U_r = 0 \tag{2.5}$$

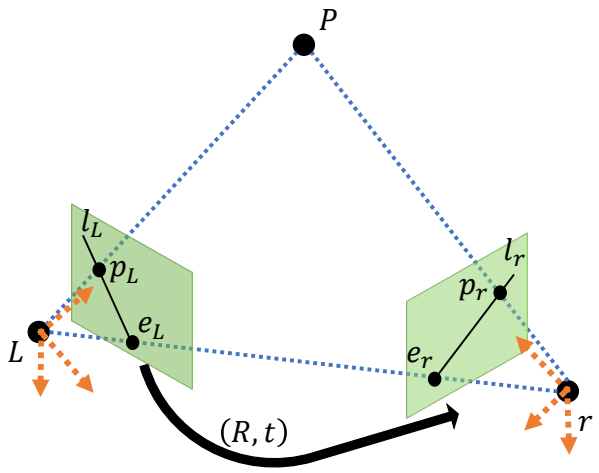


Figure 2.2: Epipolar geometry for the stereo system

where $F = K_L^{-T} E K_r^{-1}$ is the fundamental matrix with the intrinsic parameter $K_\bullet$, and $u_\bullet$ is the projected point in pixel coordinate with homogeneous form.

2.1.2 Motion Stereo

Motion stereo is the method to predict depth from a single monocular camera. When the camera is moving, the epipolar geometry is constructed based on the motion of the camera. This dissertation only addresses the case when the ego-motion of the camera is unknown, so the ego-motion should be simultaneously calculated for the depth estimation.

Most of the methods first find several matched points in both images and then calculate the depth of those points and the ego-motion. First, I extract feature points easily to be matched between frames like [57, 58, 59], and then match the extracted features between two images by proper visual tracker or descriptor based on the feature extraction method.

To calculate the depth and ego-motion, two relations are mostly adapted. One is the epipolar geometry in equation (2.4), and the other is the pixel coordination based on equation (2.1). For epipolar geometry, the fundamental matrix is given to find the ego-

motion between two images by minimizing the (2.4) as

$$F = \underset{F}{\operatorname{argmin}}(\sum_{i} u_{i_L}^T F u_{i_r}) \tag{2.6}$$

with the least square form as

$$\begin{bmatrix} u_{1_L}u_{1_r} & u_{1_L}v_{1_r} & u_{1_L} & v_{1_L}u_{1_r} & v_{1_L}v_{1_r} & v_{1_L} & u_{1_r} & v_{1_r} & 1 \\ u_{2_L}u_{2_r} & u_{2_L}v_{2_r} & u_{2_L} & v_{2_L}u_{2_r} & v_{2_L}v_{2_r} & v_{2_L} & u_{2_r} & v_{2_r} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{n_L}u_{n_r} & u_{n_L}v_{n_r} & u_{n_L} & v_{n_L}u_{n_r} & v_{n_L}v_{n_r} & v_{n_L} & u_{n_r} & v_{n_r} & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ \cdots \\ F_{33} \end{bmatrix} = 0 \tag{2.7}$$

where F_{ij} is the (i, j) component of the fundamental matrix F. It is noted that the fundamental matrix is the form of the ego-motion. Therefore, the fundamental matrix could be transformed into the relative pose.

From the relations, I could warp the pixel coordinate U_L as in the r pixel coordinate as

$$d_r U_r = K\,[R|t]\,K^{-1} d_L U_L \tag{2.8}$$

Here, U_r is the homogeneous formulation, so U_r is uniquely determined if a right term in equation (2.8) is determined. Thus, I could establish the optimization problem concerning the depth d and the pose (R, p) when the matched points are given.

2.1.3 Scale ambiguity in motion stereo

Motion stereo with a monocular camera suffers scale ambiguity due to the physical limitation of the monocular camera. Here, the fundamental matrix is included in a null space as in equation (2.7); thus, if F is the solution of the least square solution, λF is also the solution for all λ. As the definition of the fundamental matrix, if t is the predicted translation of the ego-motion, λt could also be the predicted translation.

In short, if the translation of the ego-motion t and the depth z_L is predicted from the

motion stereo, for any positive value λ, scaled translation λt and depth z_L/λ could be predicted as well. Thus, I estimate the depth ratio to the reference and cannot estimate the ego-motion or depth as a meter scale.

2.2 Self-supervised Monocular Depth Estimation

For self-supervised monocular depth estimation, the depth map and the ego-motion are jointly predicted. Thus, two convolutional neural networks are constructed: one is the depth network predicting the dense depth map from a single RGB image, and the other is the pose network predicting the relative pose from a pair of consecutive images.

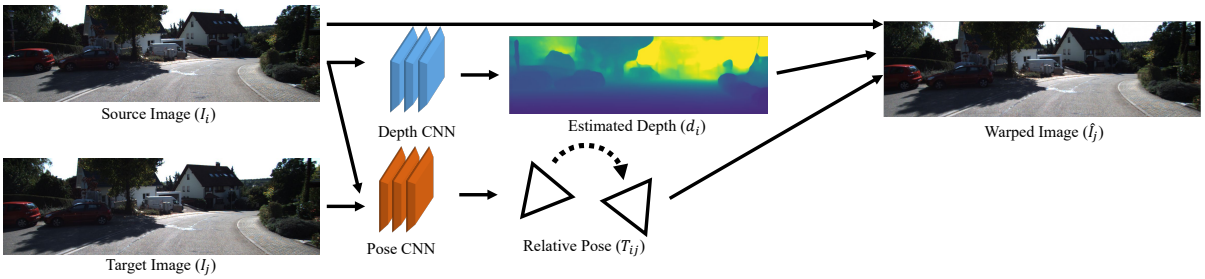


Figure 2.3: The overview of the unsupervised monocular depth estimation. From the source image I_i, a convolutional neural network, denoted as depth CNN, estimates the full depth map $\hat{d}_i$ of the source image. Additionally, from a pair of consecutive images I_i, I_j, another convolutional neural network, denoted as pose CNN, estimates the relative pose between two images T_{ij}.

With dense depth map D_n and relative pose $T_{n\to n+1}$, next pixel coordinates $\hat{u}_{n+1}$ from current pixel coordinates u_n can be obtained as in equation (2.8). Then, the warped image $\hat{I}_{n+1}$ can be obtained from the current image I_n by warping pixels. While warping the pixels, the differentiable bilinear sampling mechanism [25] is utilized to generate a differentiable signal during backpropagation.

The photometric consistency loss [26] is formulated as the distance between the target image I_{n+1} from the dataset and the warped image $\hat{I}_{n+1}$ as

$$\mathbf{L}_{\text{photo}} = \frac{1}{N}\sum_{U}\left[\text{dist}_{(I_{n+1},\hat{I}_{n+1})}(U)\right] \tag{2.9}$$

where $\text{dist}_{(I_{n+1},\hat{I}_{n+1})}(U)$ is the distance between I_{n+1} and $\hat{I}_{n+1}$ at pixel point U.

For the distance function of the photometric consistency loss, [26] shows that the linear

combination of the L1 norm and structural similarity index measure(SSIM) is good to train the network for the image restoration problem. This loss could be extended into the photometric consistency as

$$\mathrm{SSIM}_{(x,y)}(U) = \frac{2\mu_x(U)\mu_y(U) + C_1}{\mu_x^2(U) + \mu_y^2(U) + C_1} \frac{2\sigma_{xy}(U) + C_2}{\sigma_x^2(U) + \sigma_y^2(U) + C_2} \tag{2.10}$$

$$\mathrm{dist}_{(x,y)}(U) = \alpha\,|x(U) - y(U)| + (1-\alpha)\frac{1 - \mathrm{SSIM}_{(x,y)}(U)}{2} \tag{2.11}$$

where $\mu_\bullet, \sigma_\bullet^2, \sigma_{\bullet\bullet}$ is the average, variance, and covariance of $\bullet$ around U with the window size $M \times M$, C_1, C_2 is the fixed value to avoid computational instability, α is the scalar constant for linear weighting of two distance metric. In the implementation, I adopt the parameter from [7], $C_1 = 0.01^2, C_2 = 0.03^2, \alpha = 0.15, M = 3$.

In addition, for regulation, edge-aware depth smoothness is minimized [24] as

$$\mathbf{L}_{\mathrm{smooth}} = |\partial_u D_n| e^{-|\partial_u I_n|} + |\partial_v D_n| e^{-|\partial_v I_n|} \tag{2.12}$$

where $\partial_\bullet$ is the partial derivative respective to $\bullet$ direction.

To reject the effect of the occluded or moving pixel which breaks the photometric consistency, per-pixel mask $\mu \in [0, 1]$ is multiplied by the distance between images for the photometric consistency loss as

$$\mathbf{L}_{\mathrm{photo}} = \frac{1}{|\mathfrak{I}|} \sum_{U \in \mathfrak{I}} \left[\mu_{(I_n, I_{n+1})} \times \mathrm{dist}_{(I_{n+1}, \hat{I}_{n+1})}(U) \right] \tag{2.13}$$

In this dissertation, I adopt the auto-mask [7], binary masking method calculated from source and target images as

$$\mu_{(I_n, I_{n+1})}(U) = \begin{cases} 1 & \text{if } \mathrm{dist}_{(I_{n+1}, \hat{I}_{n+1})}(U) > \mathrm{dist}_{(I_{n+1}, I_n)}(U) \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

2.3 IMU preintegration

IMU preintegration aims to integrate raw IMU measurements to obtain the ego-motion between two image frames. In other words, the rotation R_N, velocity v_N and translation p_N at the next frame ego-motion ($\bullet_N$) should be formulated with the current ego-motion ($\bullet_0$) and IMU measurements containing the acceleration $\tilde{a}_i$ and the angular velocity $\tilde{\omega}_i$.

In this dissertation, the relative pose will be provided in the Lie algebra of the special Euclidean group $se(3)$. Thus, the IMU measurements are preintegrated on the manifold, so I adopted [60, 61] for IMU preintegration. The elapsed time between two IMU measurements is denoted as Δt_i from i-th measurement to $i+1$-th measurement. Then, the ego-motion of i-th frame on inertial coordinates is expressed as

$$R_i = R_0 \prod_{k=0}^{i-1} \exp\left(\omega_k \Delta t_k\right) \tag{2.15}$$

$$v_i^G = v_0^G + \sum_{k=0}^{i-1} (R_k a_k - g^G) \Delta t_k \tag{2.16}$$

$$p_i^G = p_0^G + \sum_{k=0}^{i-1} v_k \Delta t_k + \frac{1}{2} \sum_{k=0}^{i-1} (R_k a_k - g^G) \Delta t_k^2 \tag{2.17}$$

where $\omega_k = \tilde{\omega}_i - b_k^\omega - \eta_k^\omega$ is the unbiased angular velocity with bias b_k^ω and noise η_k^ω, $a_k = \tilde{a}_i - b_k^a - \eta_k^a$ is the unbiased acceleration with bias b_k^a, and the gravity g^G. exp is an exponential map of the Lie algebra of the special orthogonal group $so(3)$.

Commonly, several IMU measurements are measured between two images, and IMU is not observed at the same time as the observed images. In this dissertation, the latest measurement before the first image frame is set to $(\tilde{a}_i, \tilde{\omega}_i)$. Also, Δt_0 is defined as the elapsed time between the first image frame and the first IMU measurement, and Δt_N is defined as the elapsed time between the last IMU measurement and the second image frame.

3

Self-supervised Monocular Visual-Inertial Depth Estimation and Odometry

3.1 Overview

The proposed method originates from self-supervised monocular depth estimation described in section 2.2 and is upgraded to learn the scale by integrating IMU measurements described in section 2.3.

In this section, I describe the proposed method which learns the real-world scale from the IMU measurements as in fig. 3.1. For that, I formulate the proposed method into three parts. The first part is the loss function to train the network, which generates the relation about the scale from the IMU measurements during the training step. The second part is the network architecture, which is proper to optimize the proposed loss function. The last part is the data argumentation for the visual-inertial extension of the state-of-the-art monocular method.

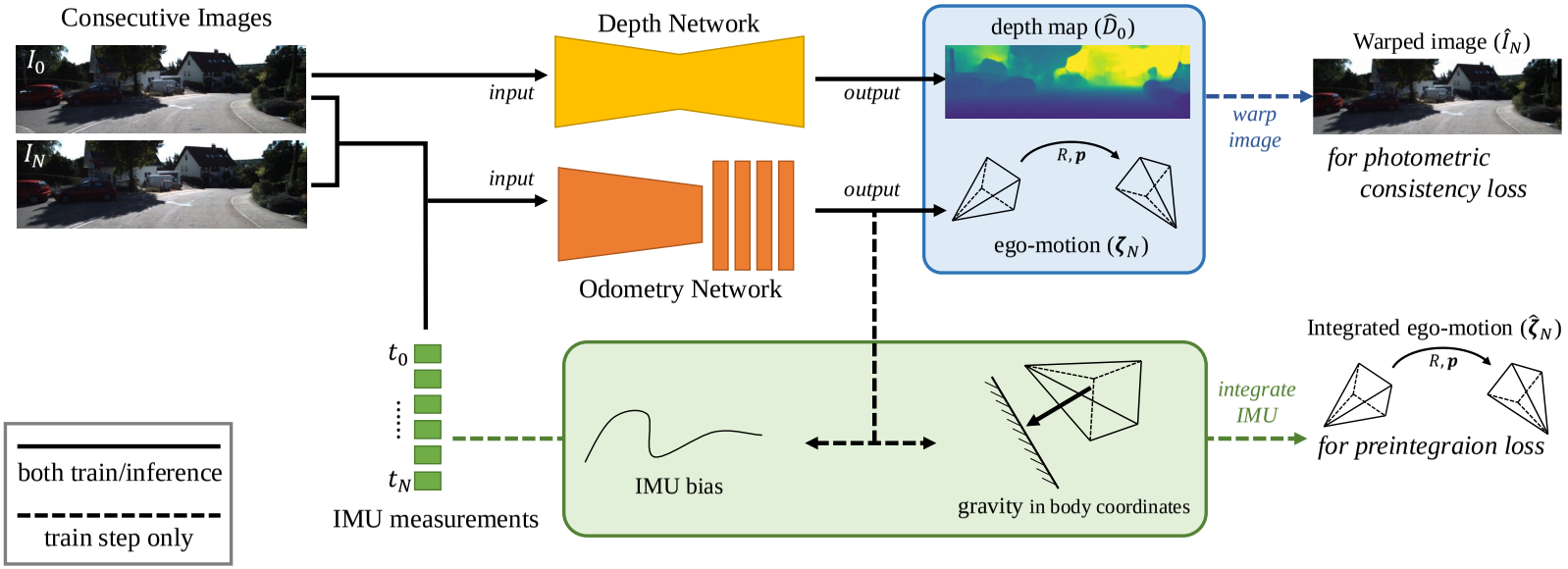


Figure 3.1: The overview of the proposed algorithm. During the inference step, a dense depth map and ego-motion are predicted from a pair of consecutive images and IMU measurements. During the training step, the gravity direction in the body coordinate and the bias of IMU are additionally predicted to learn the real-world scale for the preintegration loss function.

3.2 Loss Function

In this section, I describe each loss function of the proposed method, and the final loss is described at the end of this section. The proposed methods utilize three loss functions to optimize networks: photometric consistency loss function from state-of-the-art self-supervised monocular depth estimation to learn scale-unaware depth and ego-motion, preintegration loss function to learn the scale of ego-motion from the IMU measurements, and regulation loss function about the gravity direction and the bias to regulate the effect of predicted gravity direction and the bias.

3.2.1 Photometric consistency loss

The photometric consistency loss has been widely employed to optimize the depth map and the ego-motion from consecutive images. This loss expresses the epipolar geometry structure from motion. I adopt the photometric consistency loss (2.13) and the depth smoothness loss (2.12) like most state-of-the-art self-supervised methods described in section 2.2.

3.2.2 Preintegration loss

The preintegration loss obtains the scale-aware ego-motion by integrating IMU measurements like IMU preintegration described in section 2.3 and compares predicted ego-motion with the obtained ego-motion. The main role of the preintegration loss is to learn the scale from IMU measurements by integrating the IMU measurements and correcting integrated and predicted ego-motion. Since the photometric consistency loss does not incorporate the scale, only preintegration loss contributes to the learning of the scale.

For the predicted relative pose in the body coordinate $\zeta_N = (w_N, z_N)$ defined on $se(3)$ at time N, I *define* the relative form of the rotation ΔR_N, the velocity Δv_N and the

translation Δp_N respectively as

$$\Delta R_N \coloneqq R_0^T R_N = \exp w_N \tag{3.1}$$

$$\Delta v_N \coloneqq R_0^T \left(v_N^G - v_0^G\right) = \Delta R_N \, v_N^B - v_0^B \tag{3.2}$$

$$\Delta p_N \coloneqq R_0^T \left(p_N^G - p_0^G - v_0^G \Delta T_N\right) = \; p_N^B - v_0^B \Delta T_N \tag{3.3}$$

where $\bullet^G$ is the parameter in the inertial coordinate, $\bullet^B$ is the parameter in the body coordinate, ΔT_N is the elapsed time between two image frames, and $p_N^B = J_{w_N} z_N$ is the translation part of the relative pose with the left Jacobian of w_N denoted as J_{w_N}.

Using IMU preintegration as in equations (2.15)-(2.17), $\Delta\bullet_N$ can be expressed as

$$\Delta\hat{R}_N = \prod_{k=0}^{N-1} \exp\left(\hat{\omega}_k \Delta t_k\right) \tag{3.4}$$

$$\Delta\hat{v}_N = \sum_{k=0}^{N-1} (\Delta\hat{R}_k \hat{a}_k - g_N^B)\Delta t_k \tag{3.5}$$

$$\Delta\hat{p}_N = \sum_{k=0}^{N-1} \Delta\hat{v}_k \Delta t_k + \frac{1}{2} \sum_{k=0}^{N-1} (\Delta\hat{R}_k \hat{a}_k - g_N^B)\Delta t_k^2 \tag{3.6}$$

where $\hat{\omega} = \omega - \eta^\omega$ is the measured angular velocity, $\hat{a} = a - \eta^a$ is the measured acceleration, and $g_N^B = R_N g^G$ is the gravity in the body coordinate. In this dissertation, I denote $\Delta\bullet_N$ as $\bullet_N$ from the relative pose and $\Delta\hat{\bullet}_N$ as that from the IMU measurements to distinguish those.

From equations (3.1)-(3.6), three equality constraints can be generated: $\Delta\bullet_N = \Delta\hat{\bullet}_N$ where $\bullet = R, v, p$. To solve those equality constraints, however, the velocity v_0^B should be known. From the equality of the translation p, the closed-form of the velocity can be obtained as

$$v_0^B = \frac{1}{\Delta T_N}\left(p_N^B - \Delta\hat{p}_N\right) \tag{3.7}$$

Then, the Δv_N is reformulated as

$$\Delta v_N = \Delta R_N v_N^B - \frac{1}{\Delta T_N}\left(p_N^B - \Delta \hat{p}_N\right) \tag{3.8}$$

The calculated velocity v_0^B may be noisy because equation (3.7) is the finite difference of the predicted pose p_N^B. Therefore, I smooth the translation p_N^B by moving the average filter along the temporal axis when calculating the velocity to suppress the noise in the implementation.

Finally, I obtain the preintegration loss as the norm of two remaining equality constraints as

$$\mathbf{L}_{\text{rot}} = \lambda_{\text{rot}} \left\| \Delta R_N - \Delta \hat{R}_N \right\| \tag{3.9}$$

$$\mathbf{L}_{\text{vel}} = \lambda_{\text{vel}} \left\| \Delta v_N - \Delta \hat{v}_N \right\| \tag{3.10}$$

where $\lambda_\bullet$ is the hyperparameter for weighting the loss functions and $\left\|\bullet\right\|$ is a norm function. I adopt the logcosh as a norm function in the implementation to suppress the effect of the outlier for fast convergence.

3.2.3 Regulation loss

In addition to the scale and ego-motion, both predicted gravity and bias affect the IMU preintegration loss. If no regulation is performed, gravity and bias can be freely regressed, so the scale, gravity, and bias may be wrongly estimated. Therefore, I carefully design the regulation loss of the gravity direction and the bias of IMU.

Gravity regulation

The gravity in the inertial coordinate is assumed constant, and the gravity in the body coordinate and the ego-motion are coupled. Hence, I design the gravity regulation loss to express the gravity in the body coordinate using the predicted ego-motion. In the body

coordinate, the gravity at time N ($\hat{g}_N^B$) can be estimated from the gravity predicted by the network at time 0 (g_0^B) and the rotational ego-motion (ΔR_N) as

$$\hat{g}_N^B = \Delta R_N g_0^B \tag{3.11}$$

Since the magnitude of gravity is constant, I adopt the geodesic distance on the surface of the sphere as the loss function between the gravity predicted by the network at time N (g_N^B) and the estimated gravity $\hat{g}_N^B$ from (3.11):

$$\mathbf{L}_{\text{grav}} = \lambda_{\text{grav}} \arctan \frac{\left| g_N^B \times \hat{g}_N^B \right|}{g_N^B \cdot \hat{g}_N^B} \tag{3.12}$$

where the symbols $\cdot$ and $\times$ are the inner and outer products defined in $\mathbb{R}^3$.

The gravity regulation loss has a problem in that it may regulate the ego-motion $\Delta R_N = \exp w_N$. To avoid this problem, from the equality constraint from preintegration formulation, I exchange ΔR_N into $\Delta \hat{R}_N$ as in equation (3.4). In short, gravity regulation loss affects the gravity direction due to g_0^B and the bias of the angular velocity due to $\Delta \hat{R}_N$.

Bias regulation

It is known that the bias varies slowly, so most classical visual-inertial navigation methods construct the bias model as a constant with Gaussian noise. Similarly, I regulate both angular and linear bias by minimizing the bias difference among adjacent frames as

$$\mathbf{L}_{\text{bdiff}} = \lambda_{\text{bdiff}\omega} \left\| b_N^\omega - b_0^\omega \right\|_2^2 + \lambda_{\text{bdiff}a} \left\| b_N^a - b_0^a \right\|_2^2 \tag{3.13}$$

In addition to the regulation among adjacent frames, I also regulate the magnitude of the bias term to avoid bias prediction that is too large. This regulation is expressed as

$$\mathbf{L}_{\text{bmag}} = \lambda_{\text{bmag}\omega} \left\| b_N^\omega \right\|_2^2 + \lambda_{\text{bmag}a} \left\| b_N^a \right\|_2^2 \tag{3.14}$$

To avoid irregular prediction of the bias, λ_{grav} should be small enough in implementation.

3.2.4 Total loss

The total loss function is the linear combination of the above losses: the photometric consistency loss (2.13) with the smoothness loss (2.12), the preintegration loss of the rotational part (3.9) and the velocity part (3.10), the gravity regulation loss (3.12), and the bias regulation loss (3.13)-(3.14), as

$$\begin{aligned} \mathbf{L} = \mathbf{L}_{\text{photo}} + \mathbf{L}_{\text{smooth}} + \mathbf{L}_{\text{rot}} + \mathbf{L}_{\text{vel}} + \\ \mathbf{L}_{\text{grav}} + \mathbf{L}_{\text{bdiff}} + \mathbf{L}_{\text{bmag}} \end{aligned} \tag{3.15}$$

3.3 Network Architecture

The proposed approach estimates the depth map, the ego-motion, the gravity direction, and IMU bias from images and IMU measurements. I design two networks: a depth network and an odometry network. The depth network estimates the depth map from a single RGB image. The odometry network estimates the relative pose between two frames, gravity direction, and IMU bias from the two consecutive images and IMU measurements between images.

Depth network

I adopt the depth network proposed in [7]. The network has the U-Net structure [46], which is a fully convolutional encoder-decoder structure with skip-connection. I select ResNet18 [20] as an encoder. The depth network receives a single image, and no IMU information is received. The proposed depth network can estimate the scale by learning the scale using the preintegration loss during the training step.

Odometry network

I design an odometry network emitting relative pose, IMU bias, and gravity direction. Fig. 3.2 shows the outline of the proposed odometry network. The odometry network consists of a visual encoder, inertial encoder, feature fusion and several decoders.

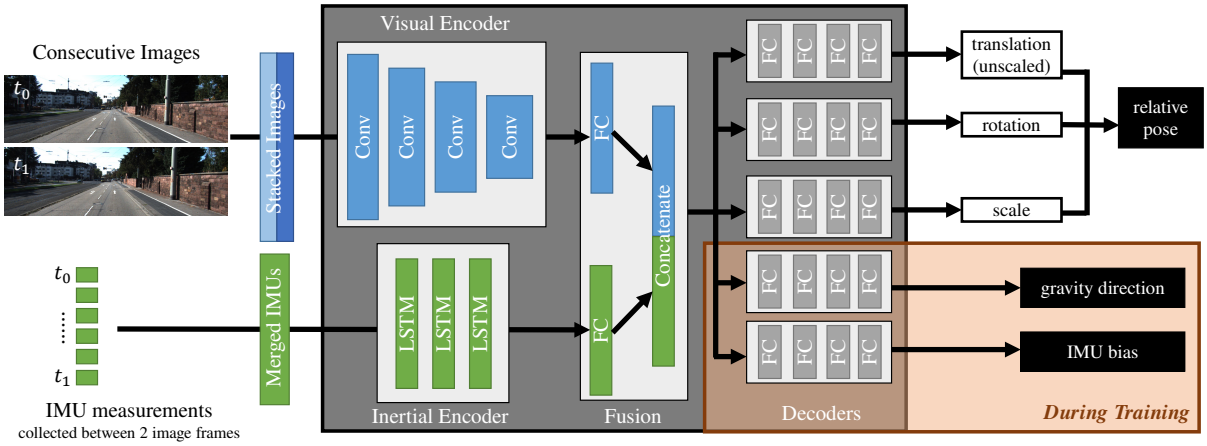


Figure 3.2: The overview of the proposed odometry network. The network receives a pair of consecutive images and IMU measurements between images. Then, the network emits the relative pose between images, the direction of gravity in the body coordinate, and the bias of the IMU measurements.

Visual encoder: I select ResNet18 [20] as a visual encoder, which is almost the same as that of the depth network. As input, two consecutive images stacked along the channel axis are provided.

Inertial encoder: I adopt bidirectional LSTM for encoding the IMU measurements. Because IMU measurements are stacked between two image frames, they have a temporal meaning, so a recurrent neural network is selected. Fig. 3.3 and table 3.1 show the flowchart and detail of the inertial encoder.

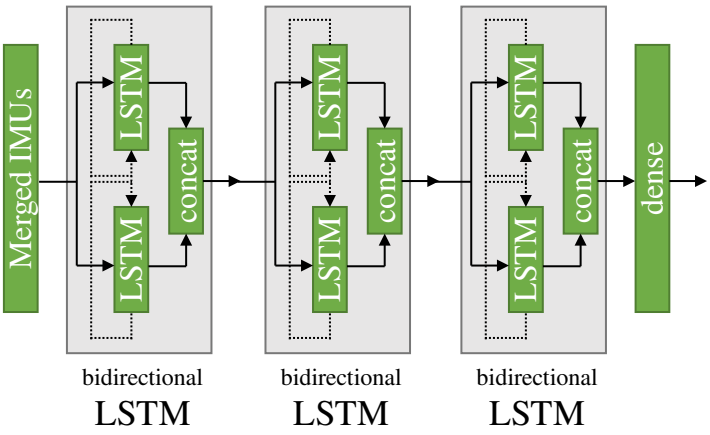


Figure 3.3: The flowchart of the inertial encoder. The inertial encoder has 3 bidirectional LSTM and 1 dense layer. Each bidirectional LSTM layer has two LSTM layers with reversal directional flow.

layer	**chns**	**out**	**input**
→imu	-	BxTxn	-
bilstm1	128	BxTx128	imu
bilstm2	128	BxTx128	bilstm1
bilstm3	128	Bx128	bilstm2
dense	128	Bx128	bilstm3
norm	-	Bx128	dense
relu→	-	Bx128	norm

Table 3.1: The detail of the inertial encoder. **layer** is the name of layer, **chns** is the number of channels, **out** is the shape of output and **input** is the name of input layer. $\rightarrow (\cdots)$ is the input of this network, $(\cdots) \rightarrow$ is the output of this network, B is the batch size and T is the time step of IMU.

Feature Fusion: The feature fusion part of the network aims to merge visual and inertial features provided by each encoder. I focus on balancing each feature to avoid a feature being ignored. Firstly, I add a single dense layer for each network. Then, I normalize each feature by layer normalization [62] for each feature to have a similar magnitude. Then, both features are concatenated to be fused as a single feature. Fig. 3.4 shows the flowchart of the fusion, and table 3.2 shows the detail of the fusion.

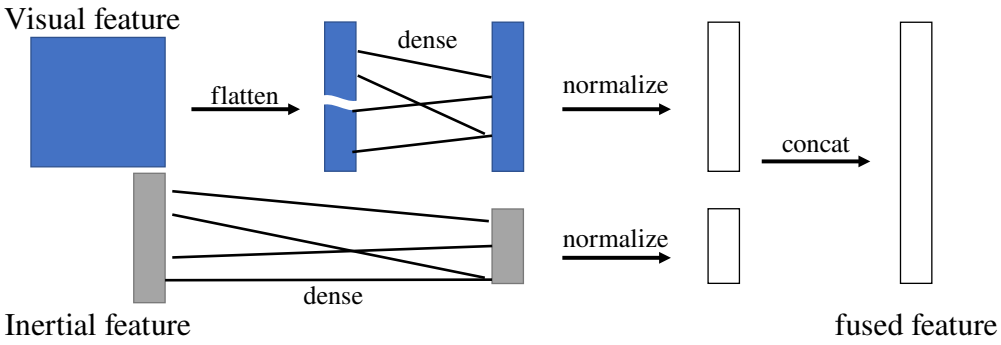


Figure 3.4: The flowchart of the feature fusion part. Each feature passes the dense layer and is normalized. Then, two features are concatenated.

layer	**chns**	**out**	**input**
→visual_f	-	BxHxWx512	-
flatten	-	Bx512HW	visual_f
dense_v	256	Bx256	flatten
norm_v	-	Bx256	dense_v
relu_v	-	Bx256	norm_v
→inertial_f	-	Bx128	-
dense_i	128	Bx128	inertial_f
norm_i	-	Bx128	dense_i
relu_i	-	Bx128	norm_v
concat→	-	Bx384	relu_v relu_i

Table 3.2: The detail of the fusion part. **layer** is the name of layer, **chns** is the number of channels, **out** is the shape of output and **input** is the name of input layer. $\rightarrow(\cdots)$ is the input of this network, $(\cdots)\rightarrow$ is the output of this network, B is the batch size and W, H is width and height of the visual feature.

Decoder: Several decoders receive the same feature from the feature fusion part and emit the final output, respectively. Each decoder contains 7 dense layers with the same channels except the last layer to decode the fused feature. Each decoder has an additional activation to properly constrain the output. In general, the input and output of the network have a magnitude of around 1, so the scalar hyperparameter may be multiplied into the output. Additionally, special activation is applied to some outputs if necessary. Table 3.3 shows the detailed architecture of the decoder.

layer	**chns**	**out**	**input**	**act**
→feature	-	Bx384	-	-
dense1	256	Bx256	feature	relu
dense2	256	Bx256	dense1	relu
dense3	128	Bx128	dense2	relu
dense4	128	Bx128	dense3	relu
dense5	64	Bx64	dense4	relu
dense6	64	Bx64	dense5	relu
dense7→	D	BxD	dense6	*custom*

Table 3.3: The detail of the decoder. **layer** is the name of layer, **chns** is the number of channels, **out** is the shape of output, **input** is the name of input layer and **act** is the activation function of the layer. $\rightarrow (\cdots)$ is the input of this network, $(\cdots) \rightarrow$ is the output of this network, B is the batch size. The number D and the *custom* activation function are different according to the type of output, i.e., $D = 6$ and *custom* is relu for bias.

For the gravity direction, I regress a 2-dof vector on the spherical coordinate because the magnitude of the gravity is fixed. For fast convergence of gravity prediction, the gravity direction is converted considering the nominal gravity direction in the robot platform by initializing the network bias as zero. For instance, if the nominal gravity direction heads the z-axis like the KITTI dataset, the activation for the gravity direction is

$$g^B = \left\| g \right\| \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix} \tag{3.16}$$

where $\left\| g \right\|$ is the magnitude of gravity implemented as 9.81.

For the relative pose, I regress 7-dof vector of logarithm forms of the translation $\tilde{z}$, rotation ω and the pseudo-scale s for the ego-motion (z, ω) on the Lie algebra of the special Euclidean group as

$$z = \tilde{z} \times \exp s \tag{3.17}$$

Here, the pseudo-scale $\exp s$ is not a real-world scale because the magnitude of the translation $\tilde{z}$ is not constrained.

3.4 Data augmentation

For deep-learning applications, data augmentation has been widely performed to generate additional data from a given dataset. In this section, I describe our data augmentation.

3.4.1 Image augmentation

I perform image augmentation at 50% probability, by changing the brightness, contrast, saturation, and hue. If image augmentation is performed, I randomly select values from uniform distribution: brightness $\in [0.8, 1.2]$, contrast $\in [0.8, 1.2]$, saturation $\in [0.8, 1.2]$ and hue $\in [-36, 36]$. Then, all the images in the sequence are converted with the same type of augmentation.

3.4.2 Left-right flip augmentation

Flip augmentation is a common augmentation method for deep-learning-based visual applications. Because the gravity usually heads downwards in the camera view, only a left-right direction flip is performed to conserve the nominal direction of gravity.

Unlike image augmentation, IMU measurements should also be converted for this flip since the ego-motion in the coordinate of the flipped camera is changed. To handle this issue, I convert IMU measurements to justify the ego-motion obtained from the integration of IMU measurements in the flipped coordinate.

Ego-motion in the flipped coordinate

I revisit the epipolar geometry generated from the motion in equation (2.4) to get the ego-motion in the flipped coordinate. For arbitrary 2-dimensional similarity transformation, V in a homogeneous notation, the epipolar geometry of the warped coordinate is given as

$$(V\tilde{p}_L)^T R \left[t\right]_X V\tilde{p}_R = 0 \tag{3.18}$$

where $\tilde{p}$ is the point in the normalized image coordinates of the warped image. For simplicity, let's assume V has no translation. Then, since $VV^T = I \det V$, the epipolar geometry could be expressed as

$$\tilde{p}_L^T \left(V^{-1}RV\right) \left(V^{-1} [t]_X V\right) \tilde{p}_R = 0 \tag{3.19}$$

Thus, the ego-motion in the flipped coordinate $(\tilde{R}, \tilde{t})$ is given as

$$\tilde{R} = V^{-1}RV \tag{3.20}$$

$$\tilde{t} = \kappa V^{-1}t \tag{3.21}$$

where κ is any positive scalar value to express the scale factor.

IMU measurement in the flipped coordinate

For notation simplicity, Let's assume the IMU measurements are aligned to the camera coordinates using the extrinsic calibration parameter.

For rotation, from equation (3.20),

$$\exp(\tilde{w}) = V \exp(\tilde{w}) V^{-1} \tag{3.22}$$

to satisfy equation (2.15) with arbitrary Δt_k. Then,

$$\tilde{w} = \text{unhat}(V\text{hat}(w)V^{-1}) \tag{3.23}$$

where hat($\bullet$) and unhat($\bullet$) are hat operation about the cross product and inverse operation of the hat operation.

Because V is a *left-handed* rotation matrix, $\tilde{w} \neq V^{-1}w$. Instead, when $V = \text{diag}(-1, 1, 1)$, the expression is simplified as

$$\tilde{w} = -Vw \tag{3.24}$$

For translation, I merge equations 2.16 and 2.17 for simplicity as

$$p_i^G = p_0^G + \mathfrak{N} v_0^G + \frac{1}{2} \sum_{k=0}^{i-1} \mathfrak{N}_i (R_k a_k - g^G) \Delta t_k^2 \tag{3.25}$$

where $\mathfrak{N}$ and $\mathfrak{N}_i$ is some scalar constant. Then, from equation (3.21) with $\kappa = 1$,

$$\tilde{a} = Va \tag{3.26}$$

$$\tilde{g}^G = V^{-1} g \tag{3.27}$$

for arbitrary Δt_k and R.

For the camera-IMU extrinsic parameter T, the IMU measurement is converted as

$$\tilde{w} = -T^T V T w \tag{3.28}$$

$$\tilde{a} = T^T V T a \tag{3.29}$$

4

Experimental Validation

I first perform the validation in the KITTI dataset. Firstly, I perform two ablation studies to show whether the proposed loss function contributes to learning the scale. Then, I show the depth performance based on the Eigen split and the pose performance based on the odometry split. Next, I validate the proposed method in indoor environments with the automobile platforms at the underground parking lots.

4.1 Performance indices

In this section, I describe the performance indices for depth and odometry validation in the XY-plane considering automobile applications. For monocular methods that cannot predict the scale, the scale is directly taken from the *ground-truth*.

4.1.1 Depth Validation

For the depth validation, five performance indices have been widely reported: absolute relative error (Abs Rel), square relative error (Sq Rel), root mean square error (RMSE),

log scale root mean square error (RMSE log), and accuracy (δ). The ratio of pixels is reported whose accuracy is less than 1.25, 1.25^2, and 1.25^3, respectively.

$$\text{Abs Rel} = \mathbf{E}\left(\frac{\left|d - \hat{d}\right|}{d}\right) \tag{4.1}$$

$$\text{Sq Rel} = \mathbf{E}\left(\frac{\left(d - \hat{d}\right)^2}{d}\right) \tag{4.2}$$

$$\text{RMSE} = \sqrt{\mathbf{E}\left(\left(d - \hat{d}\right)^2\right)} \tag{4.3}$$

$$\text{RMSE log} = \sqrt{\mathbf{E}\left(\left(\log d - \log \hat{d}\right)^2\right)} \tag{4.4}$$

$$\text{accuracy}(\delta) = \max\left(\frac{d}{\hat{d}}, \frac{\hat{d}}{d}\right) \tag{4.5}$$

where d is ground-truth depth, $\hat{d}$ is predicted depth, and $\mathbf{E}(\bullet)$ is the average of $\bullet$.

For the methods with no scale prediction, the scale is taken from the ground-truth depth in the same manner as [4], which is the ratio of estimated median depth to ground-truth median depth:

$$s_{\text{depth}} = \frac{\text{median}(d)}{\text{median}(\hat{d})} \, . \tag{4.6}$$

4.1.2 Pose Validation

In the odometry validation, average trajectory error (ATE) and relative pose error (RPE) are some famous performance indices as

$$\mathrm{ATE}_i = P_i^{-1} S \hat{P}_i \tag{4.7}$$

$$\mathrm{RPE}_i = \left(P_i^{-1} P_{i+1}\right)^{-1} \left(\hat{P}_i^{-1} \hat{P}_{i+1}\right) \tag{4.8}$$

where $P_i, \hat{P}_i$ are the ground-truth and the predicted pose in the inertial coordinate, and S is a time-invariant rigid body transformation for the alignment. I compute RMSE of the translation and the average of the rotation part as

$$*_{tr} = \mathrm{argmin}_S \sqrt{\frac{1}{N} \sum_i \|\bullet\|_{tr}^2} \tag{4.9}$$

$$*_{rot} = \frac{1}{N} \sum_i \|\bullet\|_{rot} \tag{4.10}$$

where $*$ is ATE or RPE, $\|\cdot\|_{tr}$ is the Euclidean 2-norm of the translation part of the rigid body matrix, and $\|\cdot\|_{rot}$ is the Euclidean 2-norm of the logarithm of the rotation matrix of the rigid body matrix.

For the methods with no scale prediction, a single scale value is taken from the ground-truth ego-motion across the whole trajectory by the least square solution minimizing the translation part of RPE as

$$s_{\mathrm{pose}} = \frac{\sum_i P_i \cdot \hat{P}_i}{\sum_i P_i \cdot P_i} \tag{4.11}$$

where $\cdot$ is a standard inner product in $\mathbb{R}^3$.

Since the experimental validation is performed on the automobile applications, the odometry performance is validated in the xy-place for the qualitative analysis.

4.2 Experimental Validation in the KITTI dataset

4.2.1 KITTI Dataset

I validate the proposed algorithm in the KITTI dataset [56]. The KITTI dataset is a widely used dataset that contains stereo images, IMU/GPS navigation, and LIDAR information. The KITTI dataset collects the dataset by the car in the town. Thus, the dominant motion of the vehicle is forward-directional linear motion and yaw-directional rotational motion.

In this dissertation, raw data from the KITTI dataset is collected like most deep-learning-based monocular depth estimation methods. KITTI provides raw data in two types: unsynced+unrectified and synced+rectified. In general, the synced+rectified dataset is selected since it provides a pair of rectified stereo images, the ego-motion, and the Velodyne points at a single time point with 10Hz frequency.

However, the synced+rectified dataset provides 10Hz IMU measurements, which is the same frequency as the images. Since high-frequency IMU data is required to integrate IMU measurements for ego-motion estimation, IMU data in synced+rectified is unable to utilize for visual-inertial odometry. Therefore, I collect the IMU data from the unsynced+unrectified dataset provided by 100Hz frequency. Since those IMU measurements are raw data, it is possible to collect less or more than 10 measurements between two images. In that case, I insert dummy IMU measurements with zero elapsed time or merge several IMU measurements to make exactly 10 measurements between images.

To validate the learning-based algorithm, I should split the dataset into the train split and test split. For the depth estimation, I follow the Eigen split [1], one of the most famous splits for monocular depth estimation. For the pose estimation, I collect the KITTI odometry dataset [63]. I divide the 00-08 sequences as a train set and the 09-10 sequences as a test set. I drop the 03 sequence since no raw data is provided, so no high-frequency IMU measurements are provided.

4.2.2 Implementation Detail

In this section, I describe the detailed hyperparameter about the networks, the loss functions, and the optimization setup in the implementation for experimental validation using the KITTI dataset.

Network detail

As mentioned in section 3.3, each output of the decoder is multiplied by the following heuristic value.

- Angular part of the ego-motion: 1e-2
- Translation part of the ego-motion: 1e-2
- Pseudo-scale of the ego-motion: 1e0
- Gravity direction as angles: 3e-1
- IMU bias (angular part): 1e-2
- IMU bias (acceleration part): 1e-1

The proposed odometry network has three separate decoders emitting the angular part of the ego-motion, the translation part of the ego-motion, and the pseudo-scale of the ego-motion. Due to this separated decoder strategy, tuning the heuristic values is almost not necessary.

Loss detail

To train the network, I use the ADAM [64] optimizer. The learning rate is 4e-5 at the beginning of the training and decreased by the inverse time policy with 0.98 ratios. Additionally, each loss function has the weighting parameter determined by the heuristic way as

- Photometric consistency loss: 1
- Depth smoothness loss: 1e-2
- Preintegration loss (angular part): 4e3
- Preintegration loss (velocity part): 4e1
- Gravity regulation loss: 4e0
- Bias difference regulation loss (angular part): 1e2
- Bias difference regulation loss (acceleration part): 1e2
- Bias magnitude regulation loss (angular part): 1e-2
- Bias magnitude regulation loss (acceleration part): 1e-2

Dataset detail

I collect all train data and randomly select approximately 1,100 sequences for each epoch. Each sequence consists of 8 consecutive images and 10 IMU measurements observed between two consecutive images. I iterate 200 epochs to train the networks, which takes approximately 40 hours in titan Xp GPU environments.

4.2.3 Ablation study

In the ablation study, I check the effectiveness of the proposed loss function. The first ablation study is intended to check whether the preintegration loss contributes to estimating a real-world scale. The second ablation study focuses on the bias regulation loss function described.

Ablation Study: Preintegration loss

In this ablation, I employ the proposed network architecture which receives both image and IMU as input but turn off the preintegration loss function.

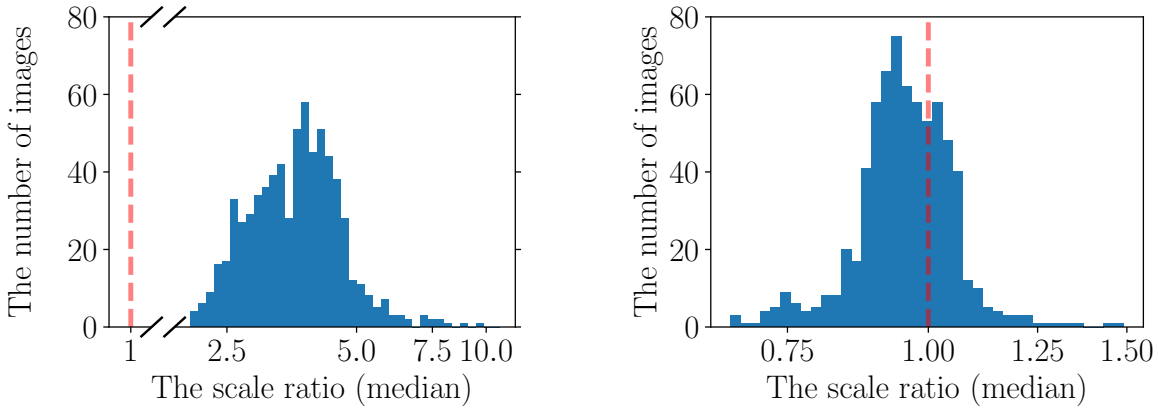
Table 4.1 shows the depth performance result. For no preintegration case, the relative depth seems good if the scale is taken from the ground-truth, but the raw depth is quite bad. On the other hand, the proposed method shows reasonable performance even in the raw depth case.

Methods	Scale[a]	Abs Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$
No preint	RAW	2.5265	35.208	0.0129	0.0349
	from GT	0.1560	6.9432	0.7758	0.9157
proposed	RAW	0.1408	5.4352	0.8038	0.9421
	from GT	0.1252	5.1737	0.8579	0.9530

[a]Scale column represents whether the method obtains the real-world scale from the ground-truth or not: 'RAW' means the real-world scale is estimated and 'from GT' means the scale is taken from the *ground-truth depth* as in (4.6) for each frame.

Table 4.1: The depth performance for the ablation study about preintegration loss function.

Fig. 4.1 is the scale prediction result of this ablation study, which shows that only the proposed method converges to the real-world scale value. I can conclude that without the preintegration loss, the relative depth can be trained due to the photometric consistency loss like self-supervised monocular methods, but the real-world scale is not learned.



(a) No preintegration loss

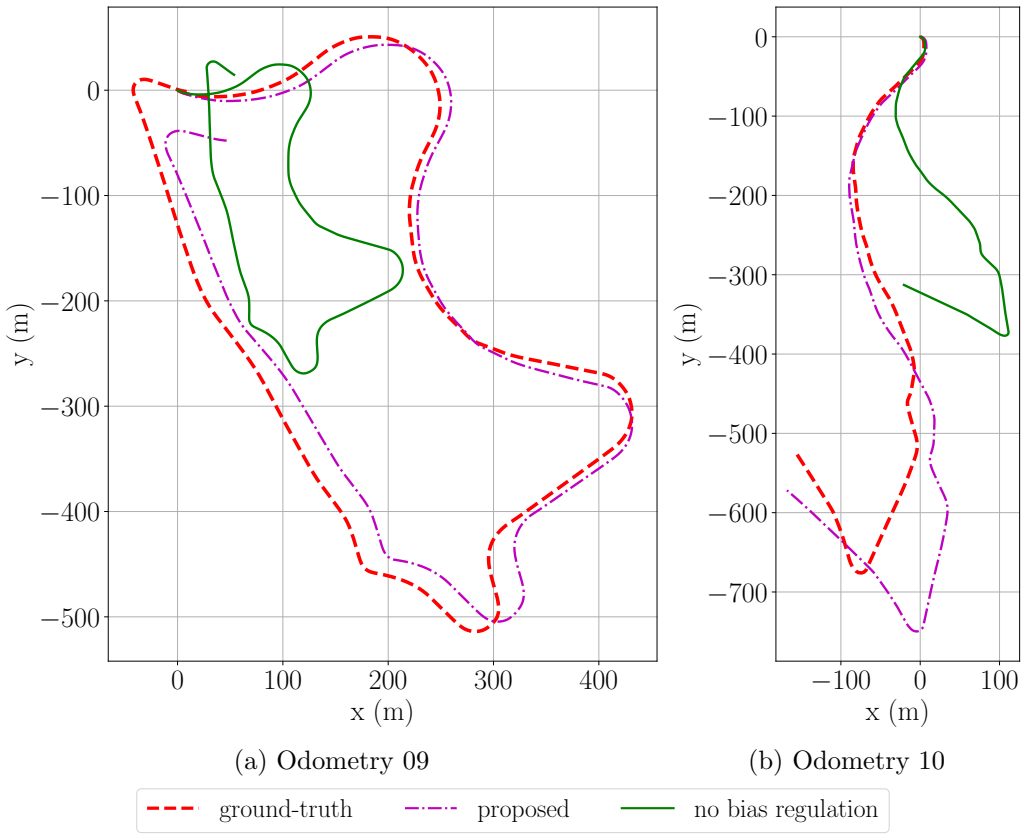
(b) proposed

Figure 4.1: The histogram of the predicted scale from the predicted depth among frames calculated by equation (4.6) for the ablation study about the preintegration loss function. If the method estimates the real-world scale, the value should be one. $(\cdot)$ next to denotes the training data: (M) is monocular sequence, (S) is stereo sequence and (MI) is monocular sequence with IMU measurement.

Ablation Study: Bias regulation loss

In this ablation, I utilize the proposed network architecture which receives both image and IMU as inputs, but I turn off both bias regulation loss functions.

Fig. 4.2 shows the top-down view of the predicted trajectory depending on whether the bias regulation loss is on. Without the bias regulation loss, the scale prediction is wrong. As shown in figs. 4.3-4.4, the magnitude of the bias is too large, and the bias tends to follow the motion of the vehicle.



(a) Odometry 09 (b) Odometry 10

Figure 4.2: Top-down view of the predicted trajectory of KITTI odometry split from the trained network for the ablation study concerning the bias regulation loss function.

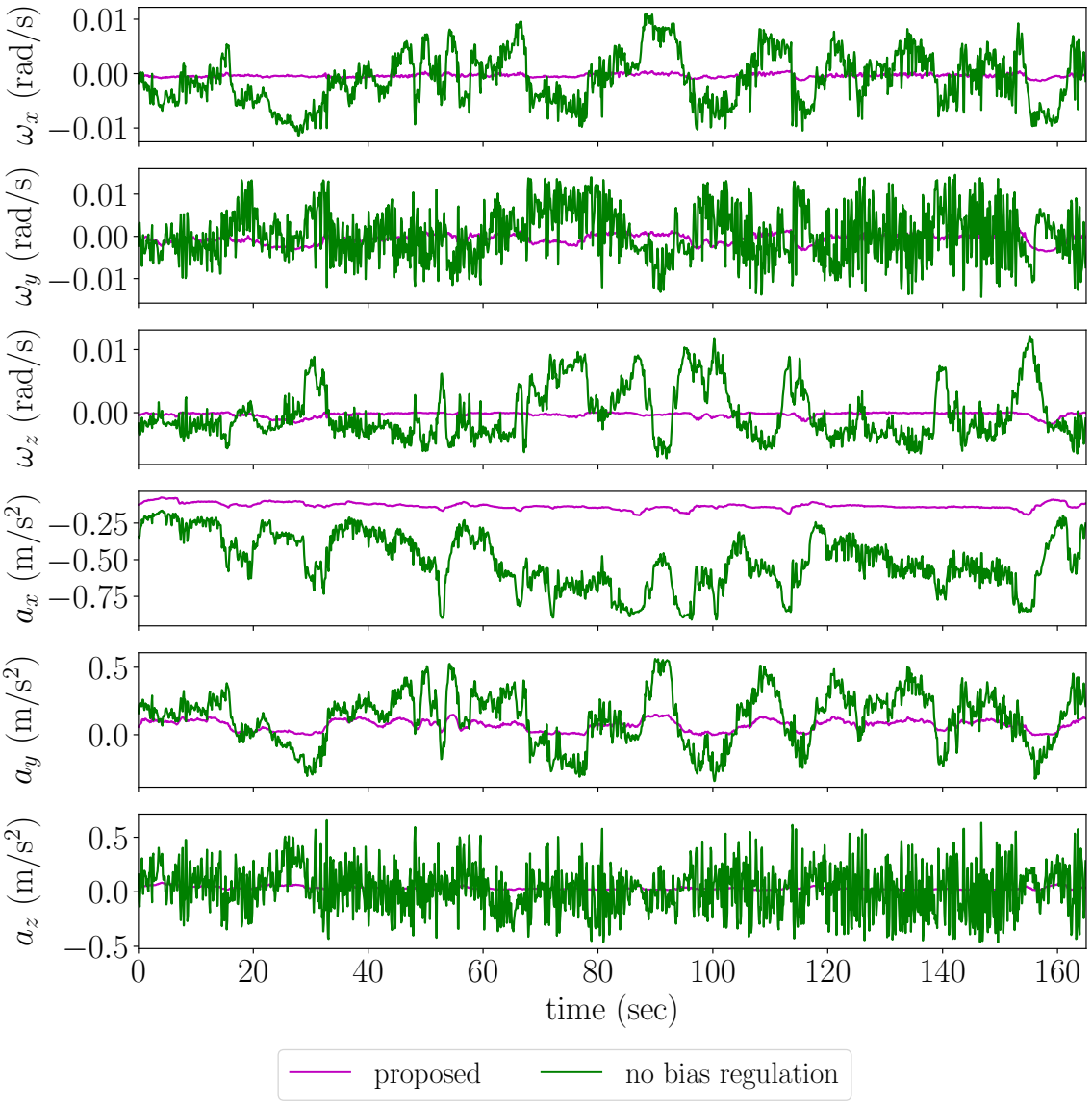


Figure 4.3: The bias prediction result at the KITTI odometry 09 for the ablation study about the regulation loss function.

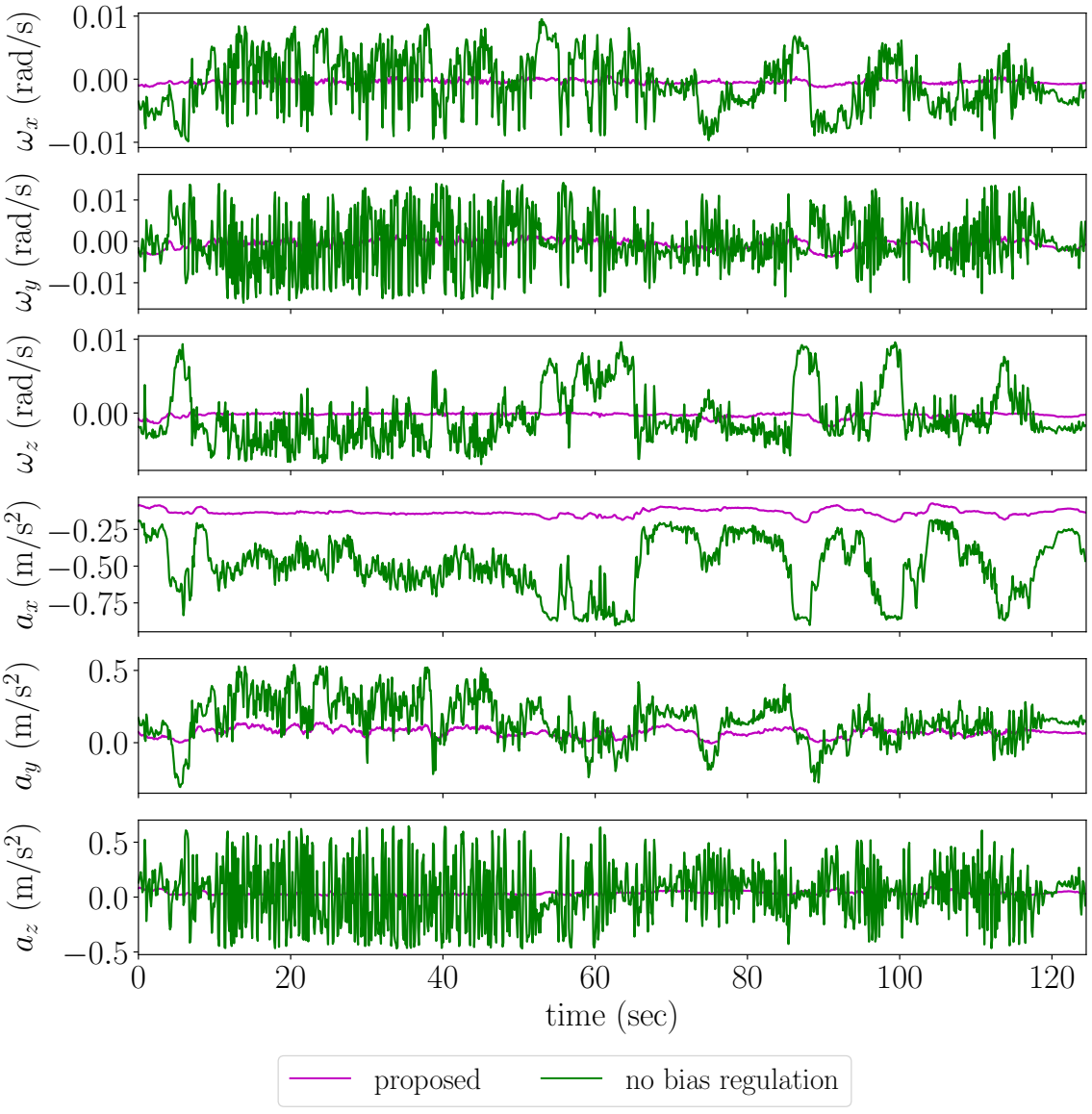


Figure 4.4: The bias prediction result at the KITTI odometry 10 for the ablation study about the regulation loss function.

4.2.4 Depth performance validation

I validate the performance of depth prediction in the KITTI Eigen split. KITTI Eigen split has 697 images as the test image.

Quantitative Analysis

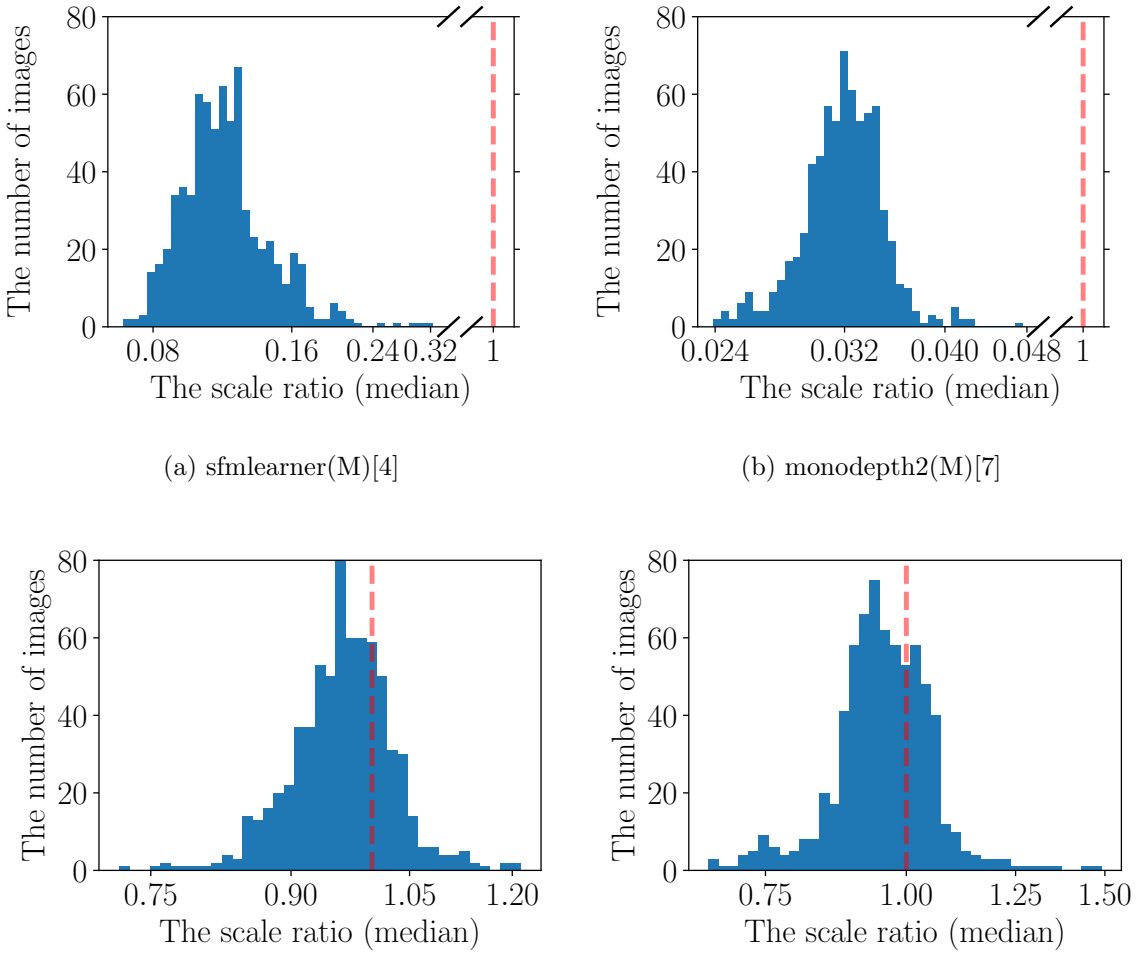
For quantitative analysis, I calculate the performance indices based on the ground-truth depth from Velodyne points for each test image. Since the Velodyne points provide sparse depth, I compare the pixels whose depth is available.

Table 4.3 shows the depth prediction performance of the proposed algorithm and state-of-the-art algorithm. The proposed method is less accurate than the state-of-the-art methods. However, it should be noted that the proposed method runs on monocular sequences with IMU measurements, which can be more easily collected than the stereo methods. Furthermore, the proposed method can predict the scale, which cannot be done by self-supervised monocular methods.

Fig. 4.5 shows the scale prediction result from the predicted depth with the statistic information in table 4.2. In this figure, the scale can be predicted by the proposed method and monodepth(S) that is the self-supervised stereo method. On the other hand, the self-supervised monocular methods, i.e., monodepth2(M) and sfmlearner(M), cannot estimate the scale.

Methods	$\mathbb{E}(s)$	$\sigma(s)$	$\mathbb{E}(\log(s))$	$\sigma(\log(s))$
sfmlearner(M)	0.1178	0.0303	-2.1656	0.2219
monodepth2(M)	0.0320	0.0029	-3.4462	0.0891
monodepth2(S)	0.9642	0.0629	-0.0385	0.0656
proposed(MI)	0.9569	0.0960	-0.0490	0.1001

Table 4.2: The scale prediction result among frames calculated by equation (4.6). For scale s, $\mathbb{E}(\bullet)$ is the average and $\sigma(\bullet)$ is the standard derivation.



(a) sfmlearner(M)[4]

(b) monodepth2(M)[7]

(c) monodepth2(S)[7]

(d) proposed(MI)

Figure 4.5: The histogram of the scale from the predicted depth among frames calculated by (4.6). If the method estimates the real-world scale, so the value should be one. $(\cdot)$ next to denotes the training data: (M) is monocular sequence, (S) is stereo sequence and (MI) is monocular sequence with IMU measurement.

Methods	Scale[a]	Train[b]	Abs Rel	Sq Rel	RMSE	$\text{RMSE}^{\log}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al. [1]	Predicted	Sup(D)	0.190	1.515	7.156	0.270	0.692	0.899	0.967
DORN [3]	Predicted	Sup(D)	0.072	0.307	2.727	0.120	0.932	0.984	0.994
BTS [65]	Predicted	Sup(D)	0.059	0.245	2.756	0.096	0.956	0.993	0.998
PWA [66]	Predicted	Sup(D)	0.060	0.221	2.604	0.093	0.958	0.994	0.999
monodepth [24]	Predicted	Unsup(S)	0.148	1.344	5.927	0.247	0.803	0.922	0.964
MonoGAN [36]	Predicted	Unsup(S)	0.119	1.239	5.998	0.212	0.846	0.940	0.976
SVS [27]	Predicted	Unsup(S)	0.094	0.626	4.252	0.177	0.891	0.965	0.984
monodepth2 [7]	Predicted	Selfsup(S)	0.106	0.818	4.750	0.196	0.874	0.957	0.979
Featdepth [67]	Predicted	Selfsup(S)	0.099	0.697	4.427	0.184	0.889	0.963	0.982
HR-Depth [68]	Predicted	Selfsup(S)	0.101	0.716	4.395	0.179	0.899	0.966	0.983
sfm-learner [4]	N/A	Selfsup(M)	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Vid2Depth [31]	N/A	Selfsup(M)	0.163	1.240	6.221	0.250	0.762	0.916	0.968
Geonet [5]	N/A	Selfsup(M)	0.155	1.296	5.857	0.233	0.793	0.931	0.973
SAVO [41]	N/A	Selfsup(M)	0.150	1.127	5.564	0.229	0.823	0.936	0.975
GANVO [38]	N/A	Selfsup(M)	0.150	1.141	5.448	0.216	0.808	0.939	0.975
SIGNet [69]	N/A	Selfsup(M)	0.133	0.905	5.181	0.208	0.825	0.947	0.981
DualNet [70]	N/A	Selfsup(M)	0.121	0.837	4.945	0.197	0.853	0.955	0.982
monodepth2 [7]	N/A	Selfsup(M)	0.115	0.913	4.873	0.193	0.877	0.959	0.981
Featdepth [67]	N/A	Selfsup(M)	0.104	0.729	4.481	0.179	0.893	0.965	0.984
PackNet-SfM [33]	N/A	Selfsup(M)	0.107	0.802	4.538	0.186	0.889	0.962	0.981
HR-Depth [68]	N/A	Selfsup(M)	0.104	0.727	4.410	0.179	0.894	0.966	0.984
SCSI [71]	N/A	Selfsup(M)	0.109	0.779	4.641	0.186	0.883	0.962	0.982
RM-Depth [72]	N/A	Selfsup(M)	0.108	0.710	4.513	0.183	0.884	0.964	0.983
proposed	Taken from GT	Selfsup(MI)	0.125	1.058	5.174	0.202	0.858	0.953	0.979
selfVIO [13]	N/A	Selfsup(MI)	0.127	1.018	5.159	0.226	0.844	0.963	0.984
proposed	Predicted	Selfsup(MI)	0.141	1.117	5.435	0.223	0.804	0.942	0.977

[a]Scale column represents whether the method predicts the real-world scale or not: 'Predicted' means the real-world scale is estimated and 'N/A' means the scale cannot be predicted, so the scale is taken from *the ground-truth depth* as in (4.6) for each frame to obtain the performance indices. 'Taken from GT' means the scale is taken from the ground-truth for relative depth, although the scale is predicted.

[b]Train column represents the training methods with the data: 'Sup(D)' is the supervised method with depth ground-truth, 'Unsup(S)' is the unsupervised method with stereo image pairs, 'Selfsup(S)' is the self-supervised method with sequences of stereo image pairs, 'Selfsup(M)' is the self-supervised method with sequences of monocular images, and 'Selfsup(MI)' is the self-supervised method with monocular image and the IMU measurement between consecutive images.

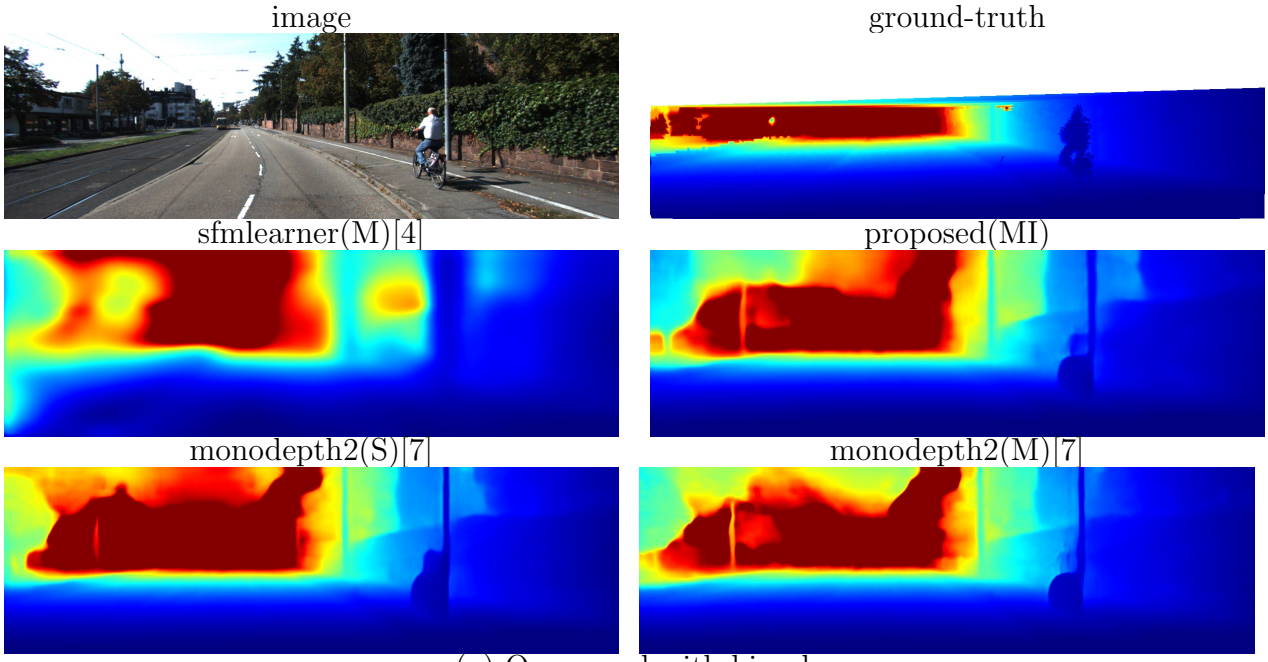
* The performance of other methods comes from each reference paper.

Table 4.3: Depth estimation performance in KITTI Eigen split (80m cap).

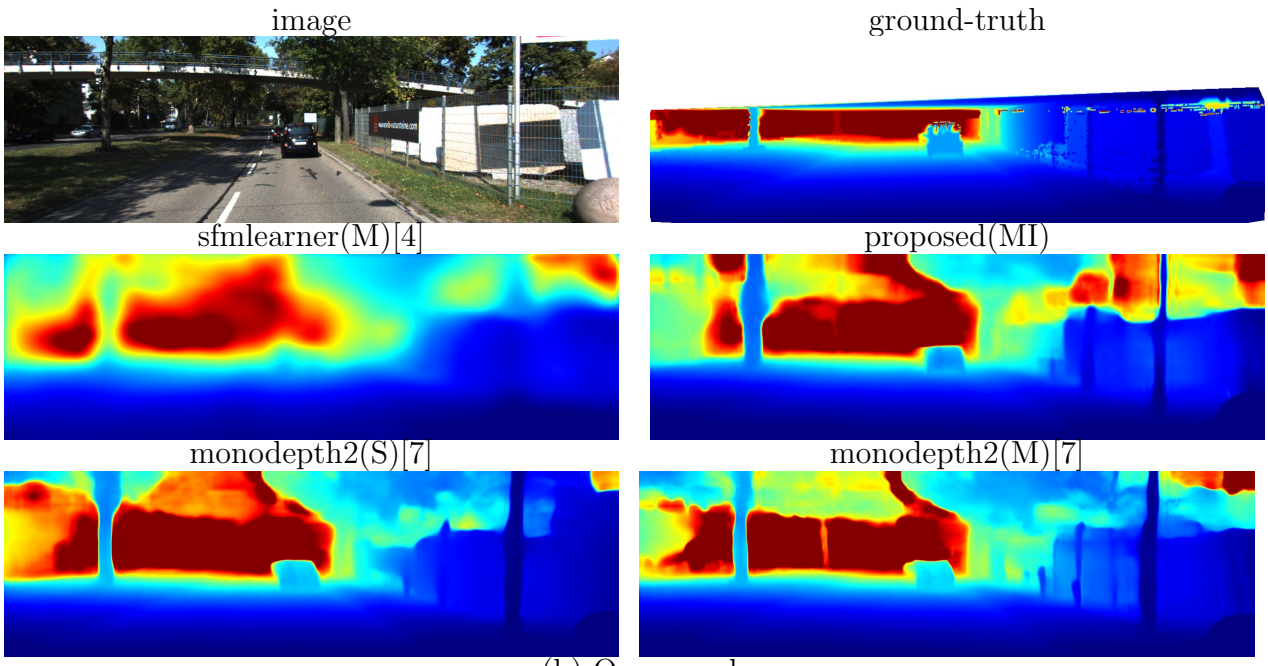
Qualitative analysis

In qualitative analysis, I compare the proposed method with other methods: monodepth2(M) from [7] with the self-supervised approach using monocular sequences, monodepth2(S) from [7] using stereo sequences and sfmlearner(M) from [4]. Since monodepth2(M) and sfmlearner(M) have no scale information, their scale is taken from the ground truth as in equation (4.6). For the ground-truth depth data, I perform bilinear interpolation for better visualization. For that reason, some depths in ground-truth seem to be irregular.

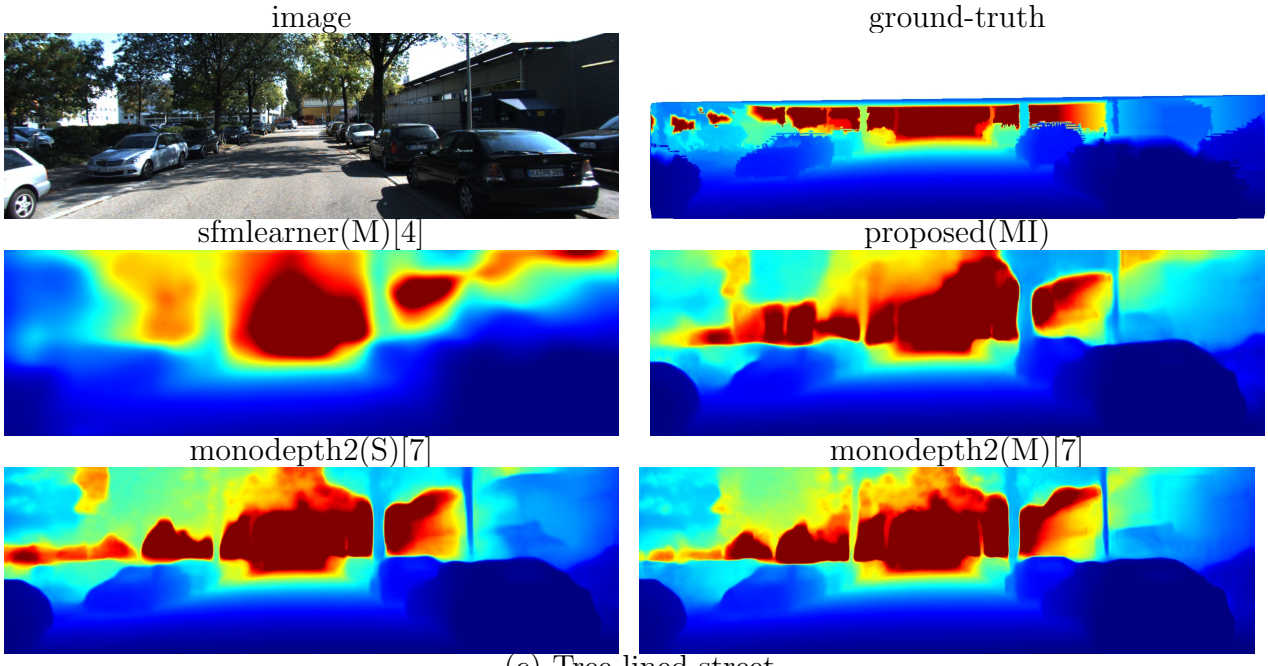
Fig. 4.6 shows the depth map predicted from a single image. In this figure, the results of the proposed method, monodepth2(M), and monodepth(S) correctly capture cars, trees, buildings, etc. In addition, they are qualitatively similar to the ground-truth color. Here, it should be noted that monodepth2(M) and sfmlearner(M) yield no scale information, so the scale used in those methods is taken from the *grond-truth depth*. On the other hand, no ground-truth information is given to the proposed method and monodepth2(S).



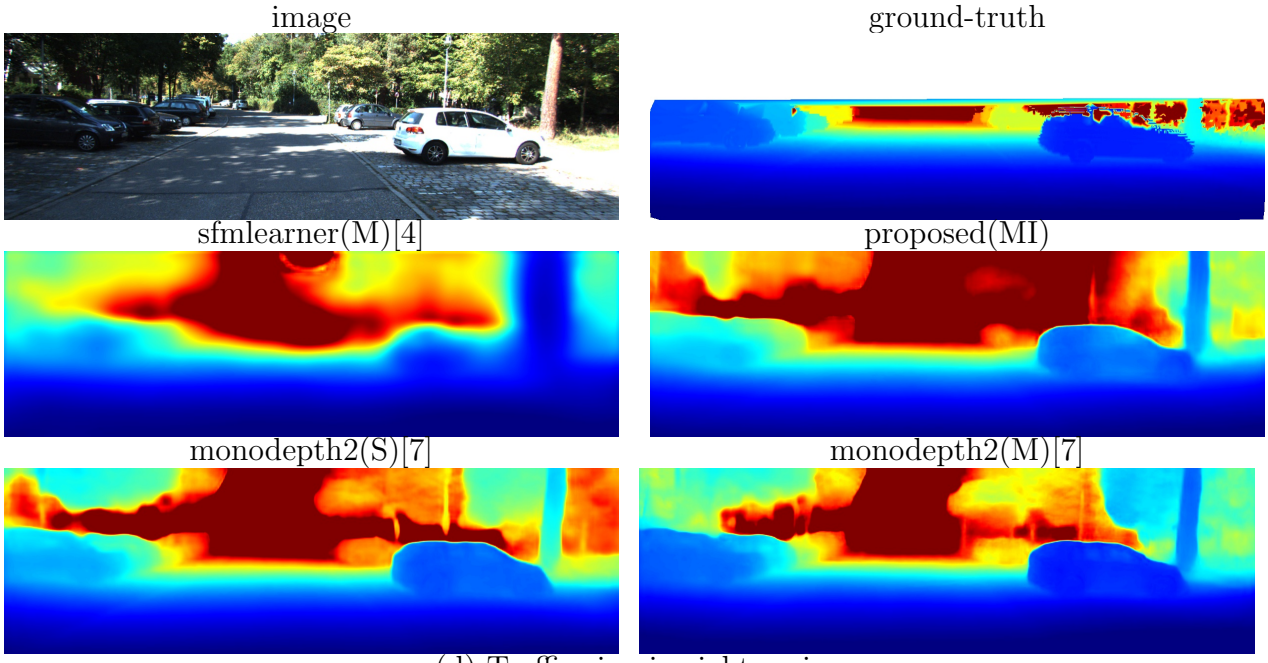
(a) Open road with bicycle



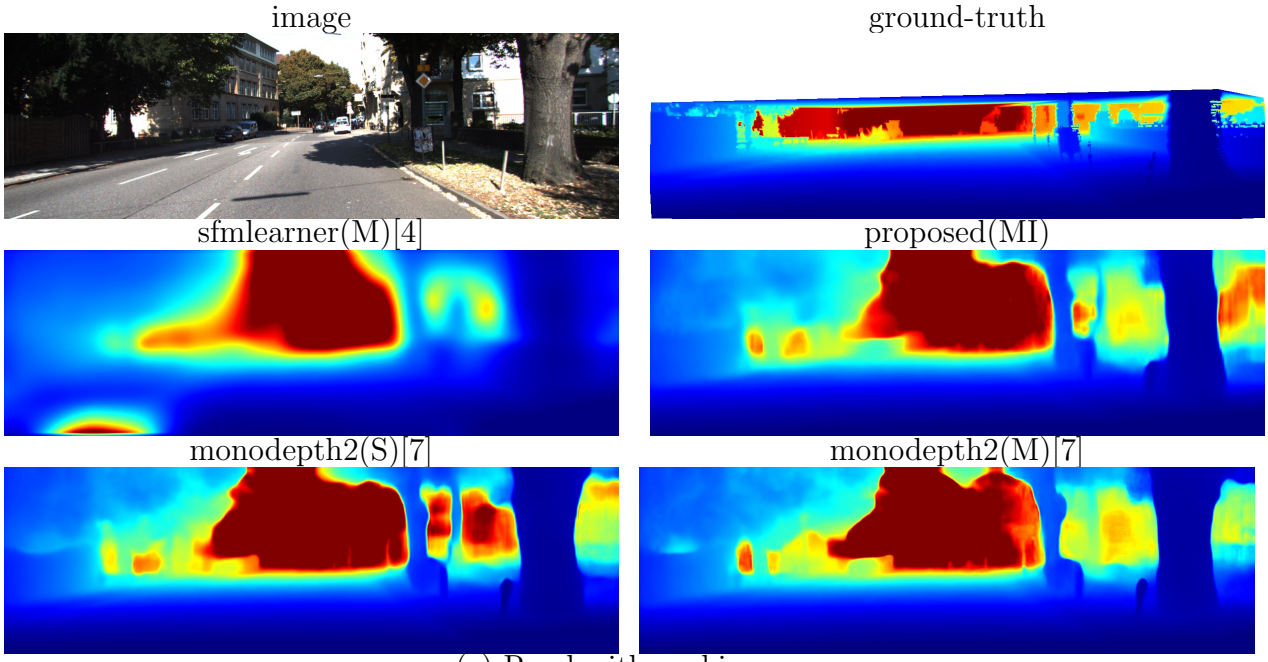
(b) Open road



(c) Tree-lined street



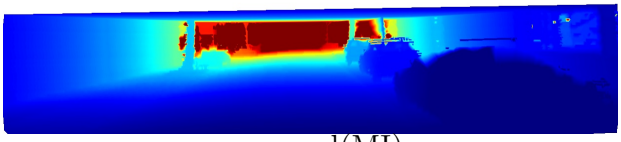
(d) Traffic sign in right region



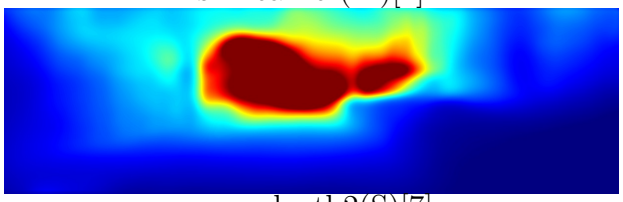
(e) Road with parking cars

image

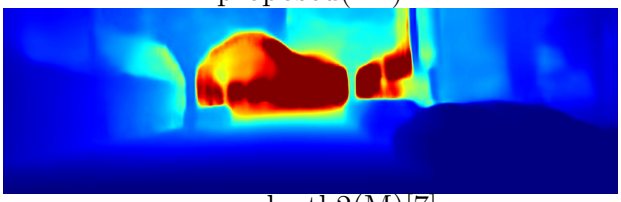
ground-truth



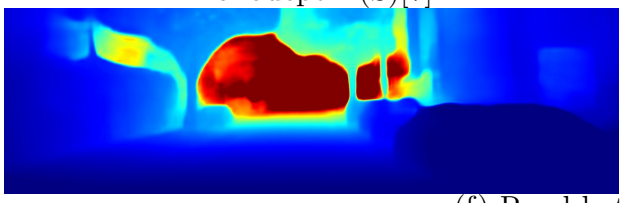
sfmlearner(M)[4]



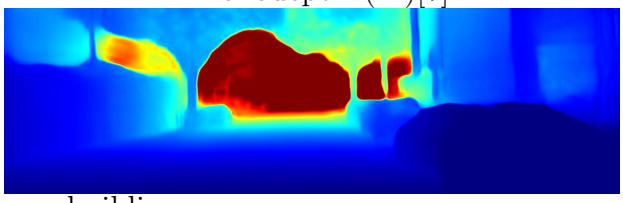
proposed(MI)



monodepth2(S)[7]



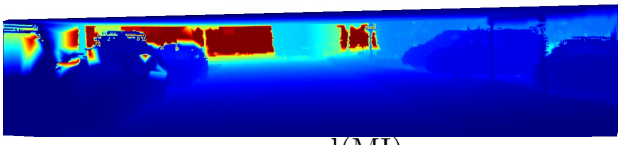
monodepth2(M)[7]



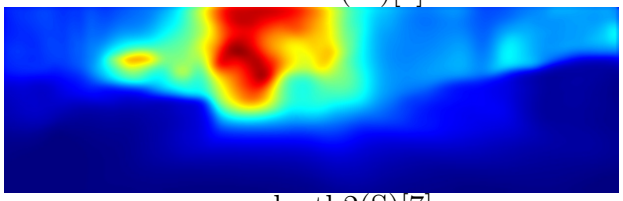
(f) Road between buildings

image

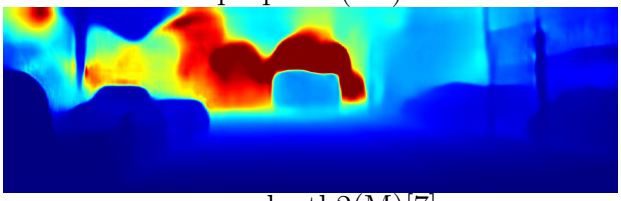
ground-truth



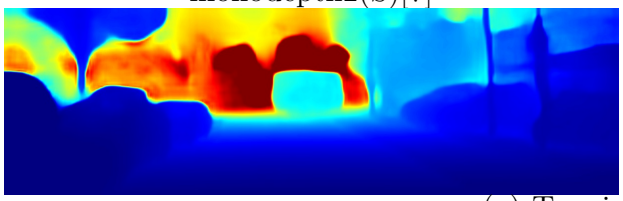
sfmlearner(M)[4]



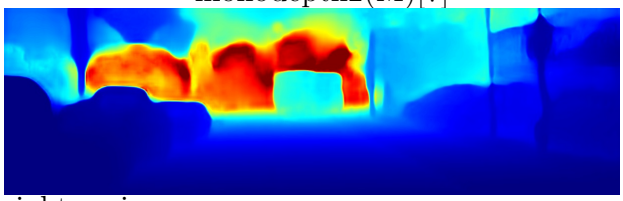
proposed(MI)



monodepth2(S)[7]



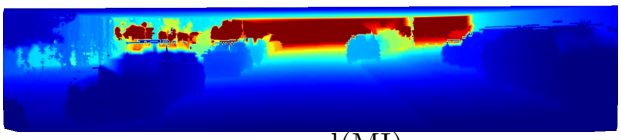
monodepth2(M)[7]



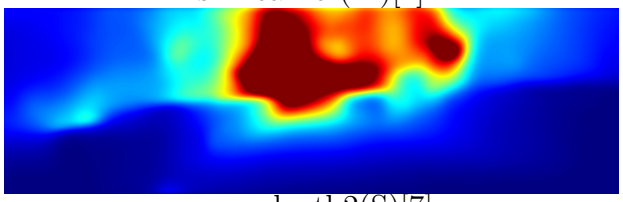
(g) Tree in right region

image

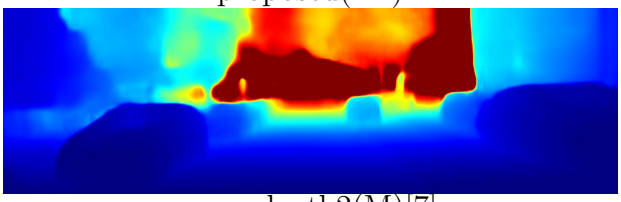
ground-truth



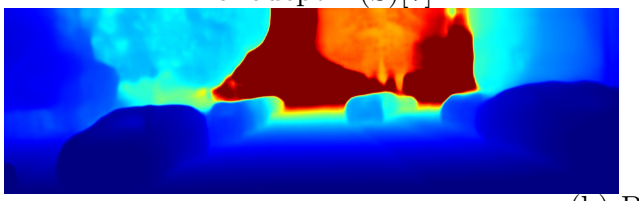
sfmlearner(M)[4]



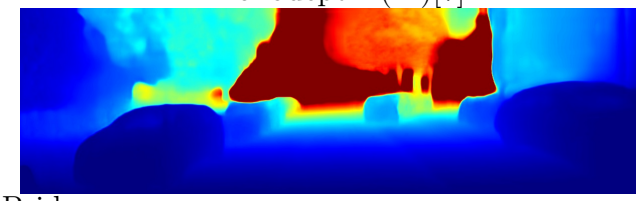
proposed(MI)



monodepth2(S)[7]

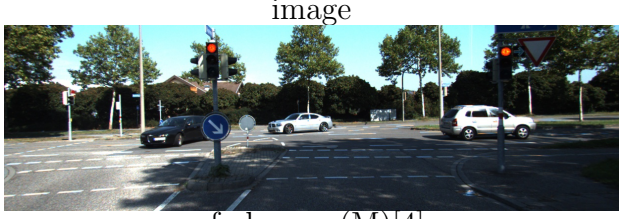


monodepth2(M)[7]

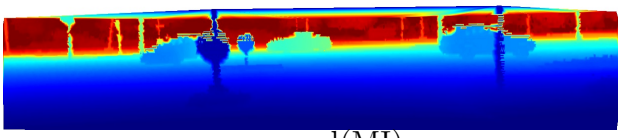


(h) Bridge

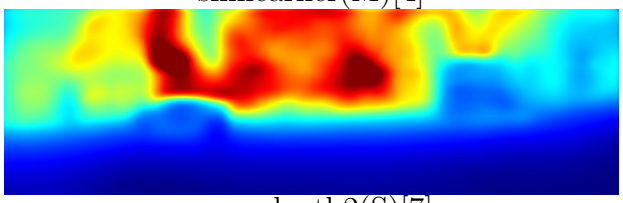
image



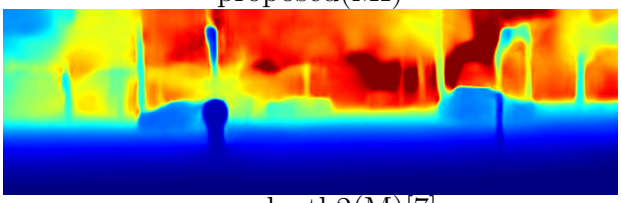
ground-truth



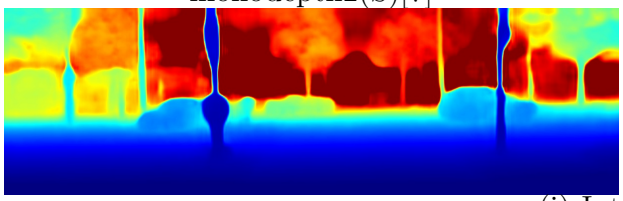
sfmlearner(M)[4]



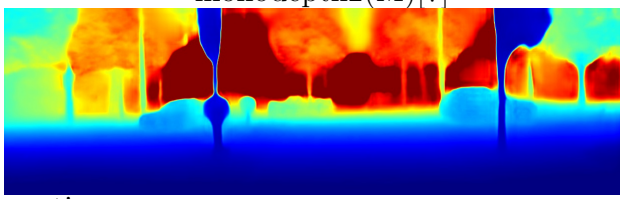
proposed(MI)



monodepth2(S)[7]



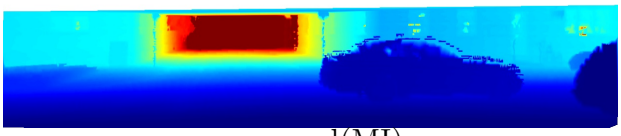
monodepth2(M)[7]



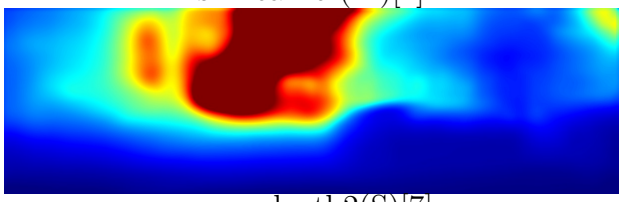
(i) Intersection

image

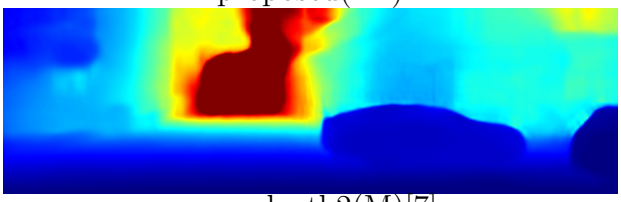
ground-truth



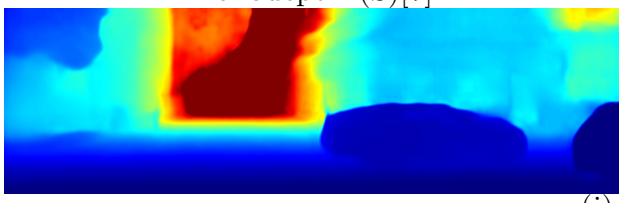
sfmlearner(M)[4]



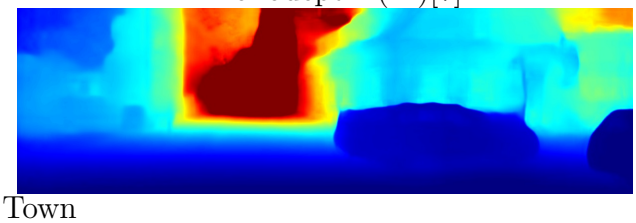
proposed(MI)



monodepth2(S)[7]



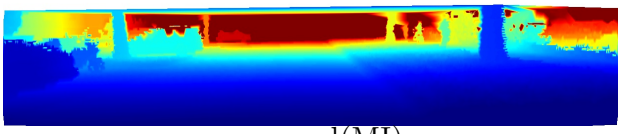
monodepth2(M)[7]



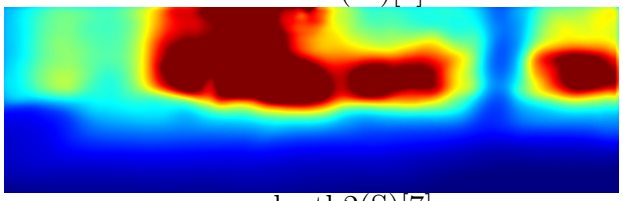
(j) Town

image

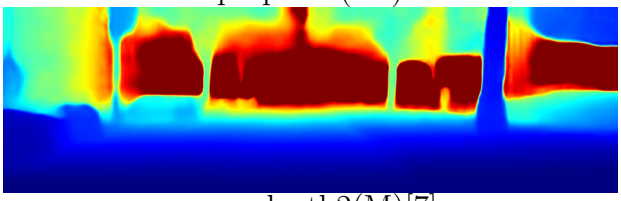
ground-truth



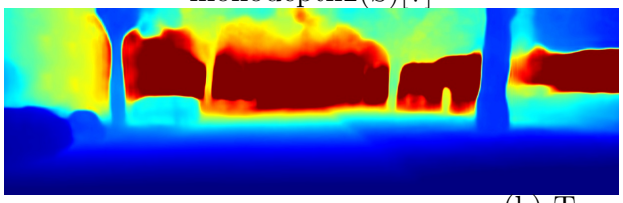
sfmlearner(M)[4]



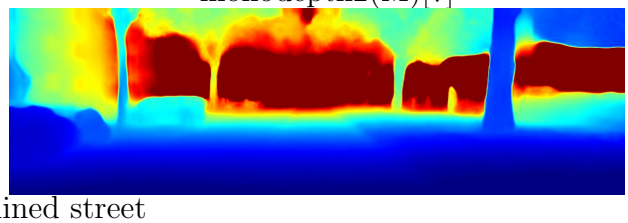
proposed(MI)



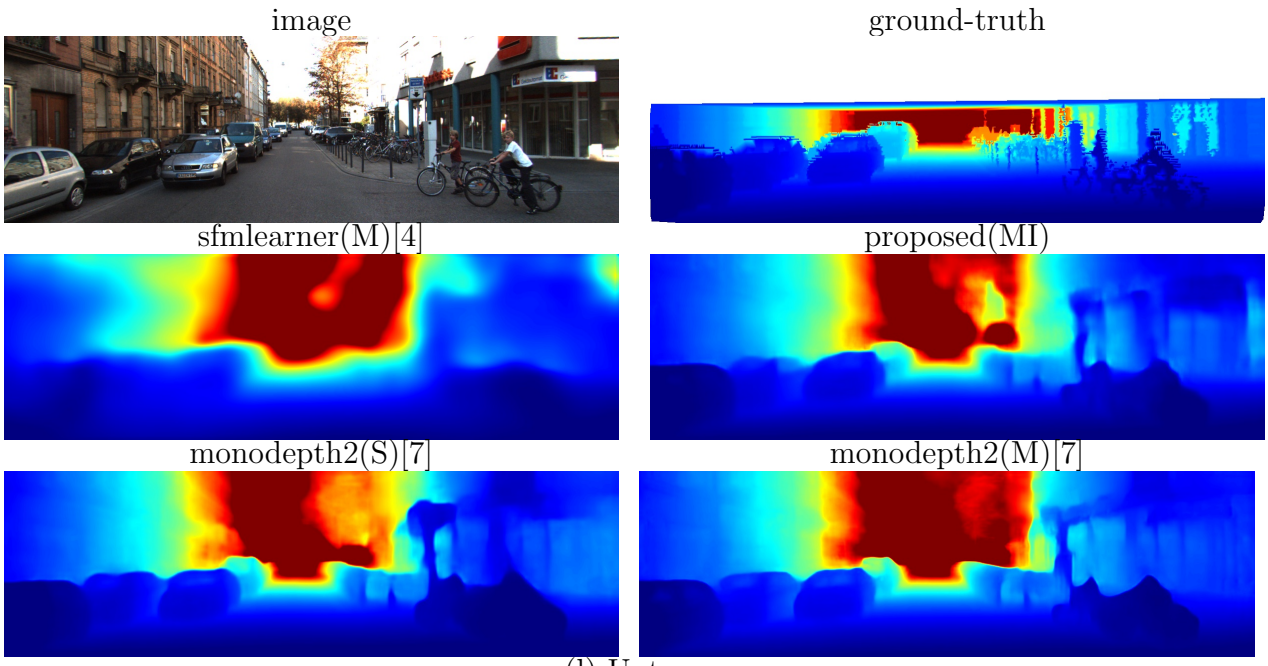
monodepth2(S)[7]



monodepth2(M)[7]



(k) Tree-lined street



(l) Uptown

Figure 4.6: The depth prediction result on the Eigen split for qualitative comparison. The depth of ground-truth is generated from the LiDAR data with bilinear interpolation, and other methods are collected from the results provided by the authors of [4, 7]. (·) next to denotes the training data: (M) is monocular sequence, (S) is stereo sequence, and (MI) is monocular sequence with IMU measurement. For the methods with (M), because no scale is estimated, the scale is taken from the *ground-truth depth* as in (4.6).

4.2.5 Pose performance validation

I test the pose prediction in the KITTI odometry split. The odometry 00-08 are collected as the train data, and 09-10 are collected as the test data. In this dissertation, odometry 03 is dropped since high-frequency IMU data is not provided.

Like depth performance, I perform two validations. One is the quantitative analysis based on the performance indices to show the overall performance of pose estimation. The other is the qualitative analysis by comparing the estimated top-down trajectory result.

Quantitative analysis

In table 4.4, the proposed method shows comparable performance to other state-of-the-art methods, considering that self-supervised monocular methods and classical monocular visual navigation methods cannot predict the scale, so the scale from the *ground-truth pose* was used.

For the scale analysis using the predicted pose, I define frame-wise scale using the relative pose as

$$s_i \coloneqq \frac{\|z_i\|}{\|\hat{z}_i\|} \tag{4.12}$$

where s_i is the scale at i-th frame, $\zeta_i = (w_i, z_i)$ is the relative pose represented as $se(3)$ with rotation w_i and translation z_i.

Figs. 4.7-4.8 and tables 4.5 show the scale prediction result from the predicted pose. For odometry 09, both visual-inertial methods show accurate scale prediction results. In contrast, the scale prediction at odometry 10 is not good. The proposed method seems to be good concerning the histogram in Fig. 4.8, but the average of the scale is bad. It is because the proposed method degrades the ego-motion prediction at 110 seconds, as in Fig. 4.14.

Methods	Scale[a]	Data[b]	ATE(m)	ATE(°)	RPE(m)	RPE(°)
[d]monodepth2[7]	N/A	Selfsup(M)	147.750	21.2581	0.0821	0.0500
[d]sfmlearner[4]	N/A	Selfsup(M)	136.811	24.5313	0.1496	0.0678
[d]monodepth2[7]	Predicted	Selfsup(S)	133.274	19.0951	**0.0701**	0.0528
[d]FeatDepth[67]	Predicted	Selfsup(S)	72.149	10.6586	0.0655	0.0421
[e]ORB-SLAM3[8]	N/A	M[c]	**22.463**	**0.2975**	0.2781	**0.0202**
[e]VINS-MONO[9]	Predicted	MI	30.142	0.9943	0.1251	0.0267
proposed	Predicted	Selfsup(MI)	26.241	2.1733	0.1705	0.0380

(a) Odometry 09

Methods	Scale[a]	Data[b]	ATE(m)	ATE(°)	RPE(m)	RPE(°)
[d]monodepth2[7]	N/A	Selfsup(M)	132.529	26.5673	0.0897	0.0527
[d]sfmlearner[4]	N/A	Selfsup(M)	174.491	35.8786	0.1834	0.1038
[d]monodepth2[7]	Predicted	Selfsup(S)	149.966	30.4696	0.0747	0.0608
[d]FeatDepth[67]	Predicted	Selfsup(S)	131.944	25.3016	0.0788	0.0586
[e]ORB-SLAM3[8]	N/A	M[c]	**20.040**	**2.7212**	**0.0462**	0.0607
[e]VINS-MONO[9]	Predicted	MI	108.240	3.4176	0.1864	0.0704
proposed	Predicted	Selfsup(MI)	63.664	6.9489	0.2493	**0.0427**

(b) Odometry 10

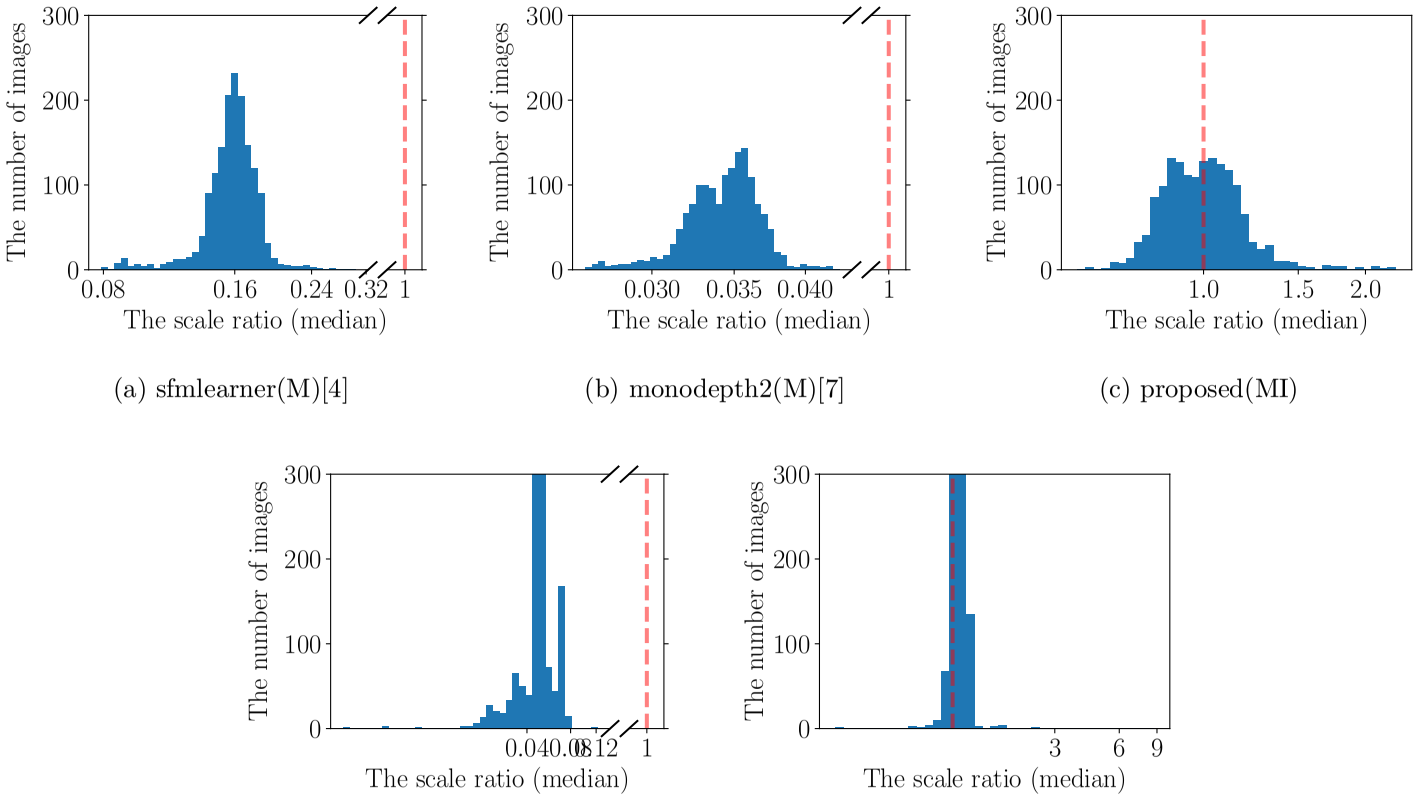
[a]Scale column represents whether the method predicts the real-world scale or not: 'Predicted' means that the real-world scale is estimated and 'N/A' means that the scale cannot be predicted, so the scale is taken from the *ground-truth pose* as in (4.11) across the trajectory.

[b]Data column represents the training method and utilized data: 'Selfsup(M)' is the self-supervised learning method with monocular sequences, 'Selfsup(MI)' is the self-supervised learning method with monocular sequences and the IMU measurements, 'M' is classical monocular navigation, and 'MI' is classical monocular-inertial navigation.

[c]Due to the IMU initialization issue, monocular visual-inertial odometry failed. So instead, the monocular visual odometry is performed.

[d]The performance is obtained based on the weight provided by the authors of [4, 24, 67].
[e]The performance is obtained based on the code provided by the authors of [8, 9].

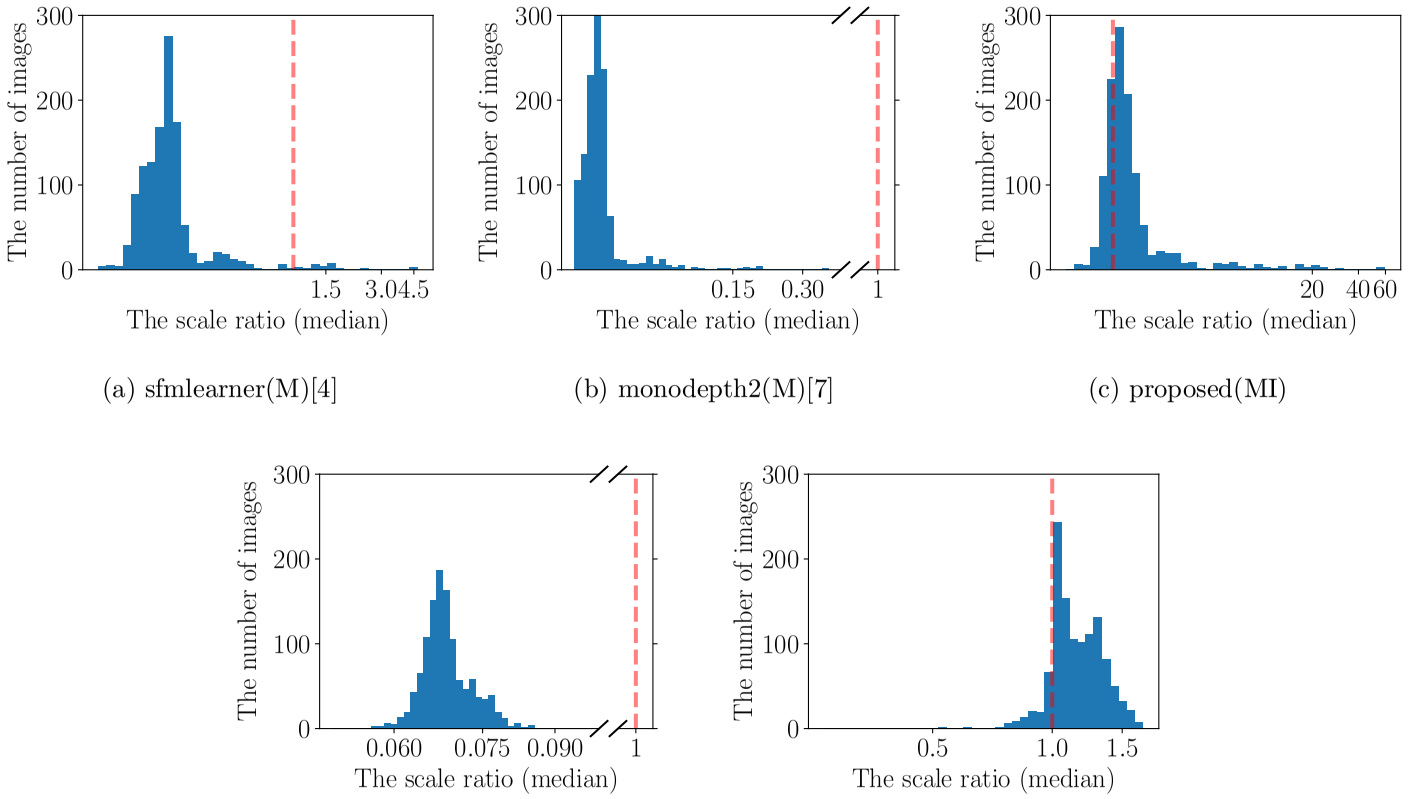
Table 4.4: Pose estimation performance in KITTI odometry dataset.



(a) sfmlearner(M)[4] (b) monodepth2(M)[7] (c) proposed(MI)

(d) ORBSLAM3(M)[8] (e) VINSMONO(MI)[9]

Figure 4.7: The histogram of the scale in odometry 09 from the predicted pose among frames calculated by equation (4.12). If the method estimates the real-world scale, the value should be one. $(\cdot)$ denotes the inference data: M is monocular image and MI is monocular image with IMU measurement.



(a) sfmlearner(M)[4] (b) monodepth2(M)[7] (c) proposed(MI)

(d) ORBSLAM3(M)[8] (e) VINSMONO(MI)[9]

Figure 4.8: The histogram of the scale in odometry 10 from the predicted pose among frames calculated by equation (4.12). If the method estimates the real-world scale, the value should be one. $(\cdot)$ denotes the inference data: M is monocular image and MI is monocular image with IMU measurement.

Methods	$\mathbb{E}(s)$	$\sigma(s)$	$\mathbb{E}(\log(s))$	$\sigma(\log(s))$
sfmlearner(M)	0.1586	0.0222	-1.8519	0.1477
monodepth2(M)	0.0343	0.0023	-3.3751	0.0698
ORBSLAM3(M)	0.0482	0.0120	-3.0730	0.3298
VNISMONO(MI)	1.0631	0.2229	0.0530	0.1127
proposed(MI)	1.0119	0.1957	-0.0044	0.1750

(a) odometry 09

Methods	$\mathbb{E}(s)$	$\sigma(s)$	$\mathbb{E}(\log(s))$	$\sigma(\log(s))$
sfmlearner(M)	0.2658	0.3491	-1.5290	0.4811
monodepth2(M)	0.0437	0.0273	-3.1957	0.2895
ORBSLAM3(M)	0.0686	0.0045	-2.6812	0.0643
VNISMONO(MI)	1.1592	0.1782	0.1352	0.1617
proposed(MI)	2.0088	4.3792	0.2942	0.6159

(b) odometry 10

Table 4.5: The scale prediction result from the predicted pose among frames calculated by equation (4.12). For the scale s, $\mathbb{E}(\bullet)$ is the average and $\sigma(\bullet)$ is the standard derivation.

Qualitative analysis

For qualitative analysis, I provide two types of information. The first one is the top-down trajectory of the whole trajectory. Since the KITTI dataset contains the movement of the car, major motion is generated on the XY-plane. The second one is the relative pose estimation result represented as $se(3)$ group.

Fig. 4.9 shows the top-down view of the predicted result. Since some classical methods provide no odometry information at the first step, I discard first few frames for fair comparison. The proposed method shows reasonable performance when compared with other methods, especially deep-learning-based methods.

Figs. 4.11 - 4.14 show the relative pose estimation of odometry 09 and 10. All methods follow the tendency of the ground-truth trajectory. In odometry 09, ORBSLAM3(M) follows the trajectory but vibrates the relative ego-motion estimation along the x-axis. In the odometry split 10, learning-based methods except the proposed method tend to underestimate the yaw directional rotation at 90 seconds.

When I focus on the beginning of the trajectory, VINSMONO [9] shows poor performance before 5 seconds, as in fig. 4.10, perhaps due to the IMU initialization issue. For a car-driving case like the KITTI dataset, it is difficult to initialize IMU with a monocular camera because the motion of the vehicle is homogeneous. For this reason, the error of VINSMONO is large in the beginning, and ORB3-SLAM [8] with the monocular-inertial mode fails in this example.

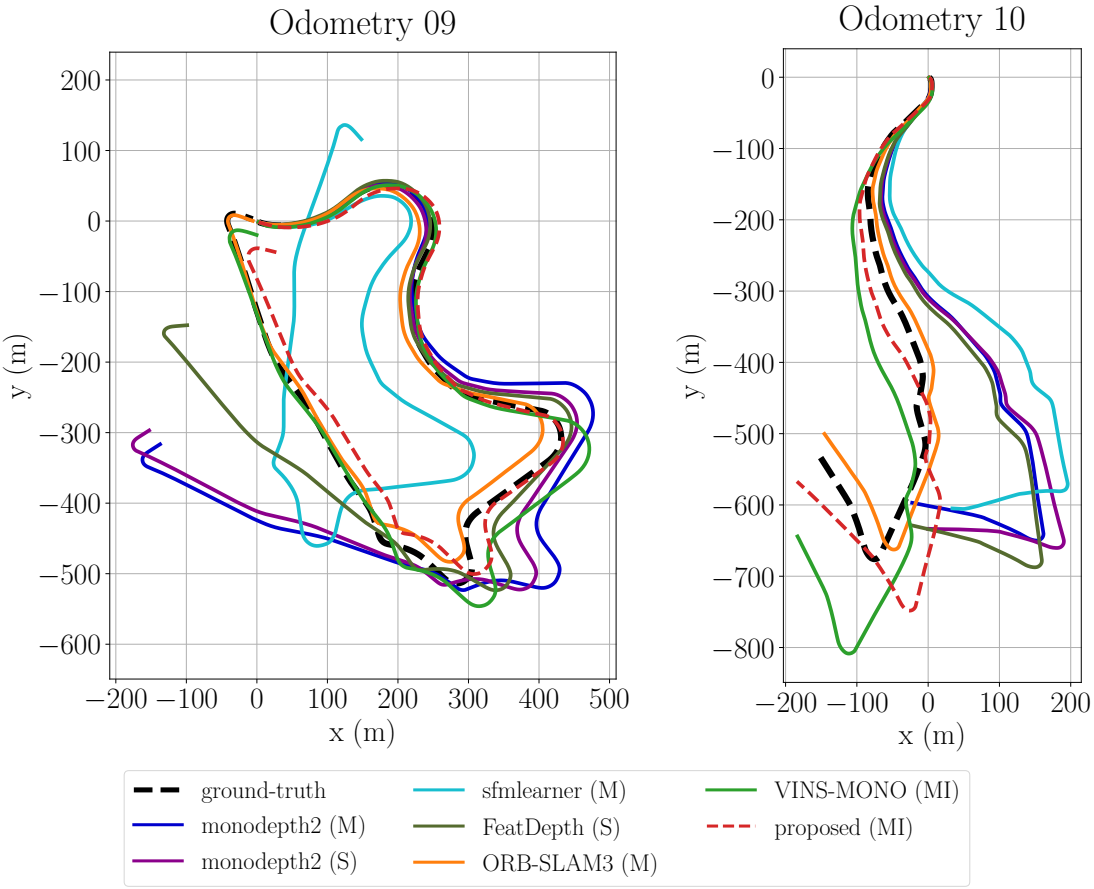


Figure 4.9: The top-down view of the estimated trajectory of the KITTI odometry dataset. $(\cdot)$ next to the method denotes the training data for the deep-learning-based method or the operating data for the classical navigation: (M) is monocular sequence, and (MI) is monocular sequence with IMU measurements. For the methods with (M), because no scale is estimated, the scale is taken from the *ground-truth* as in (4.11). Each trajectory is aligned by fixing the initial point at origin.

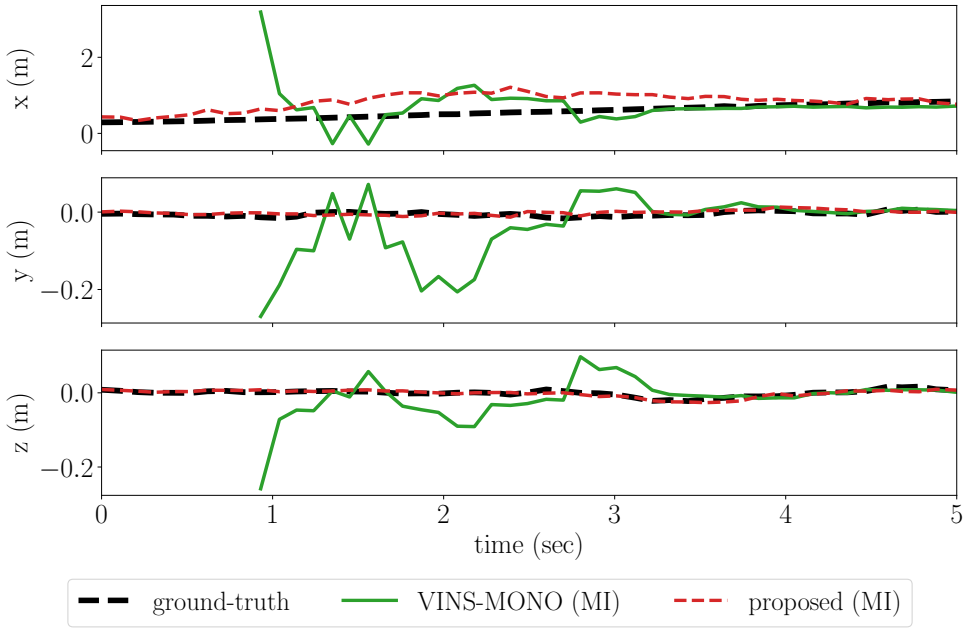


Figure 4.10: Relative pose estimation of linear motion at the beginning of the driving in odometry 09.

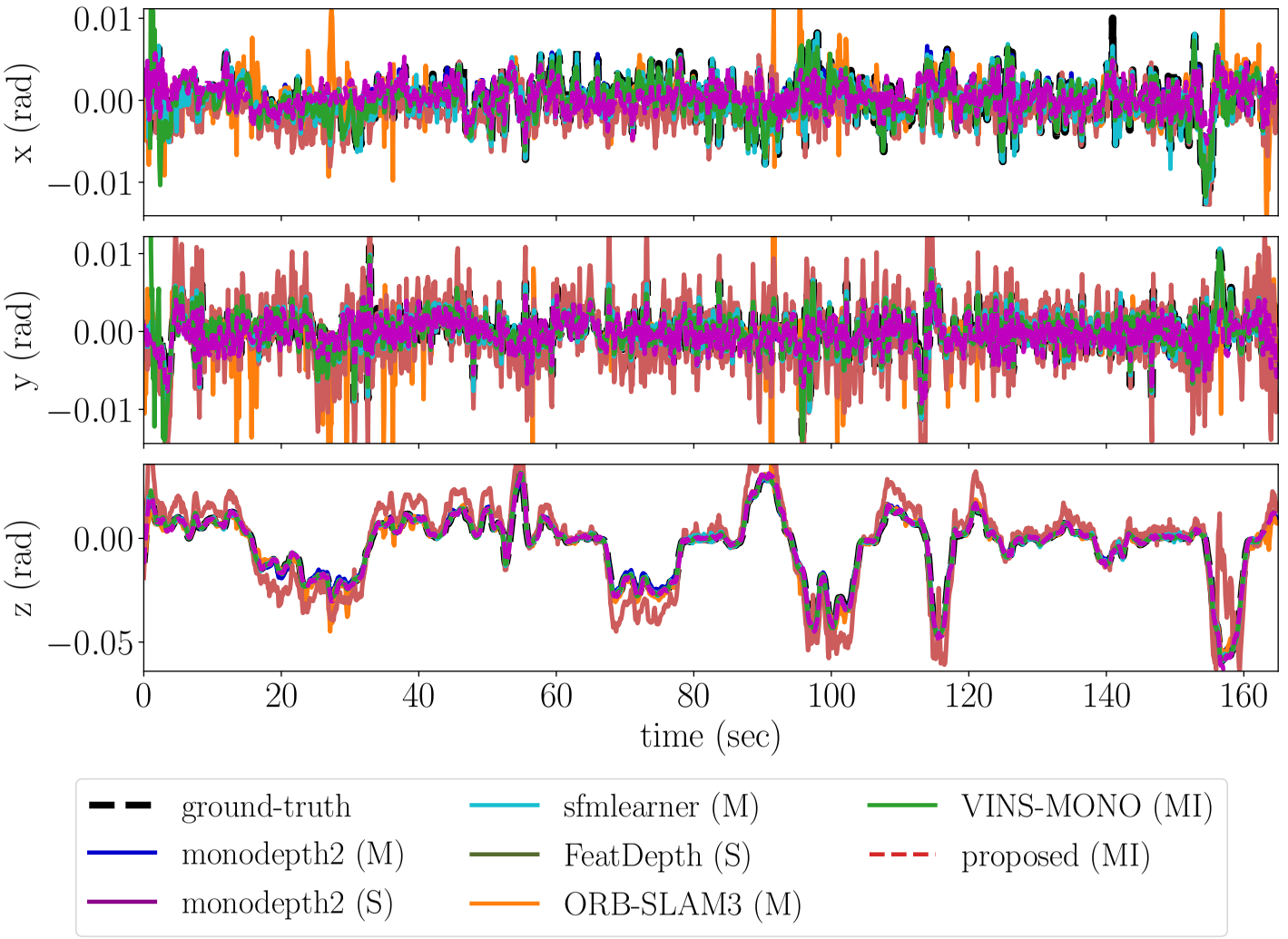


Figure 4.11: Relative pose estimation of angular motion in odometry 09.

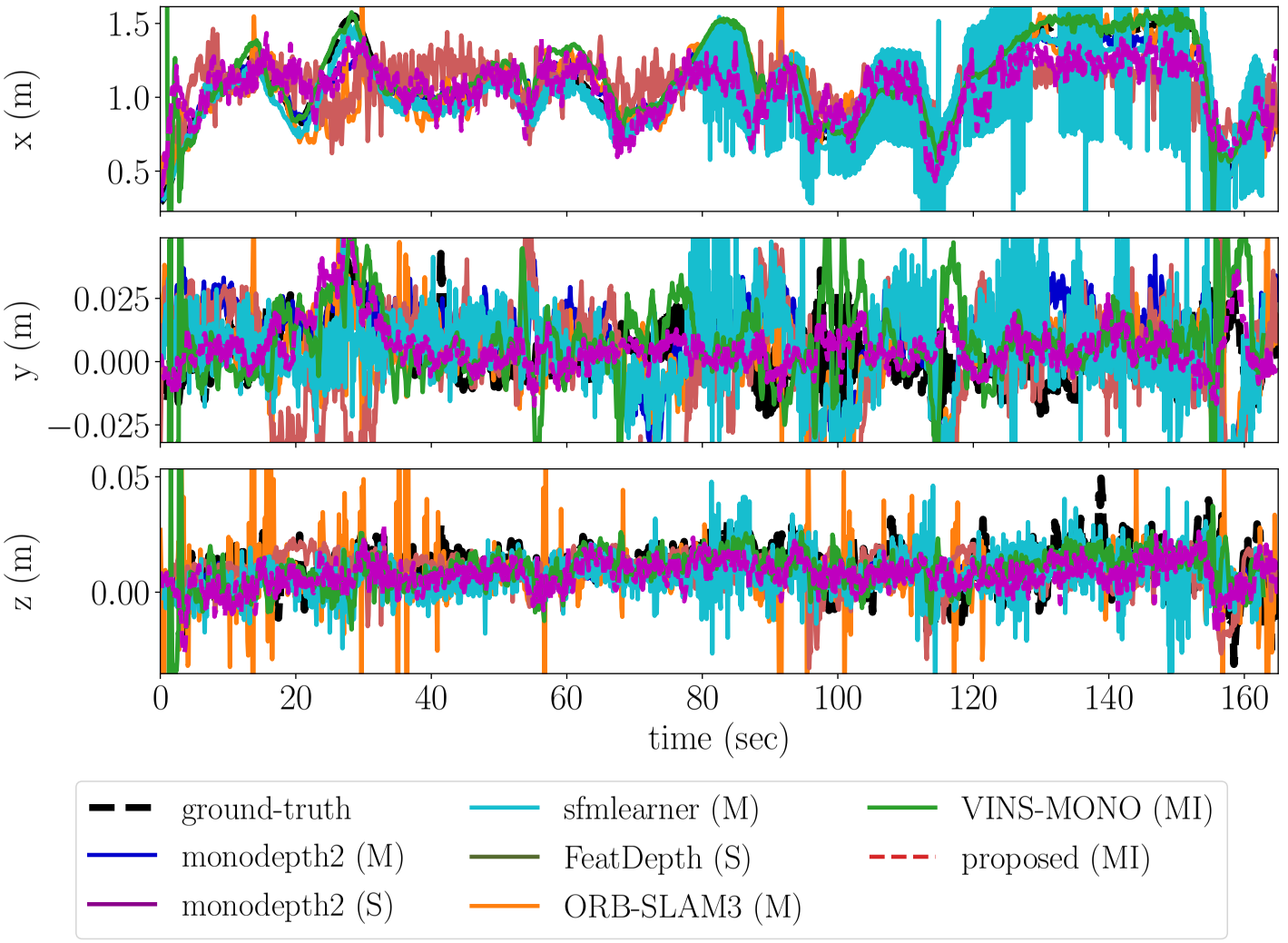


Figure 4.12: Relative pose estimation of linear motion in odometry 09.

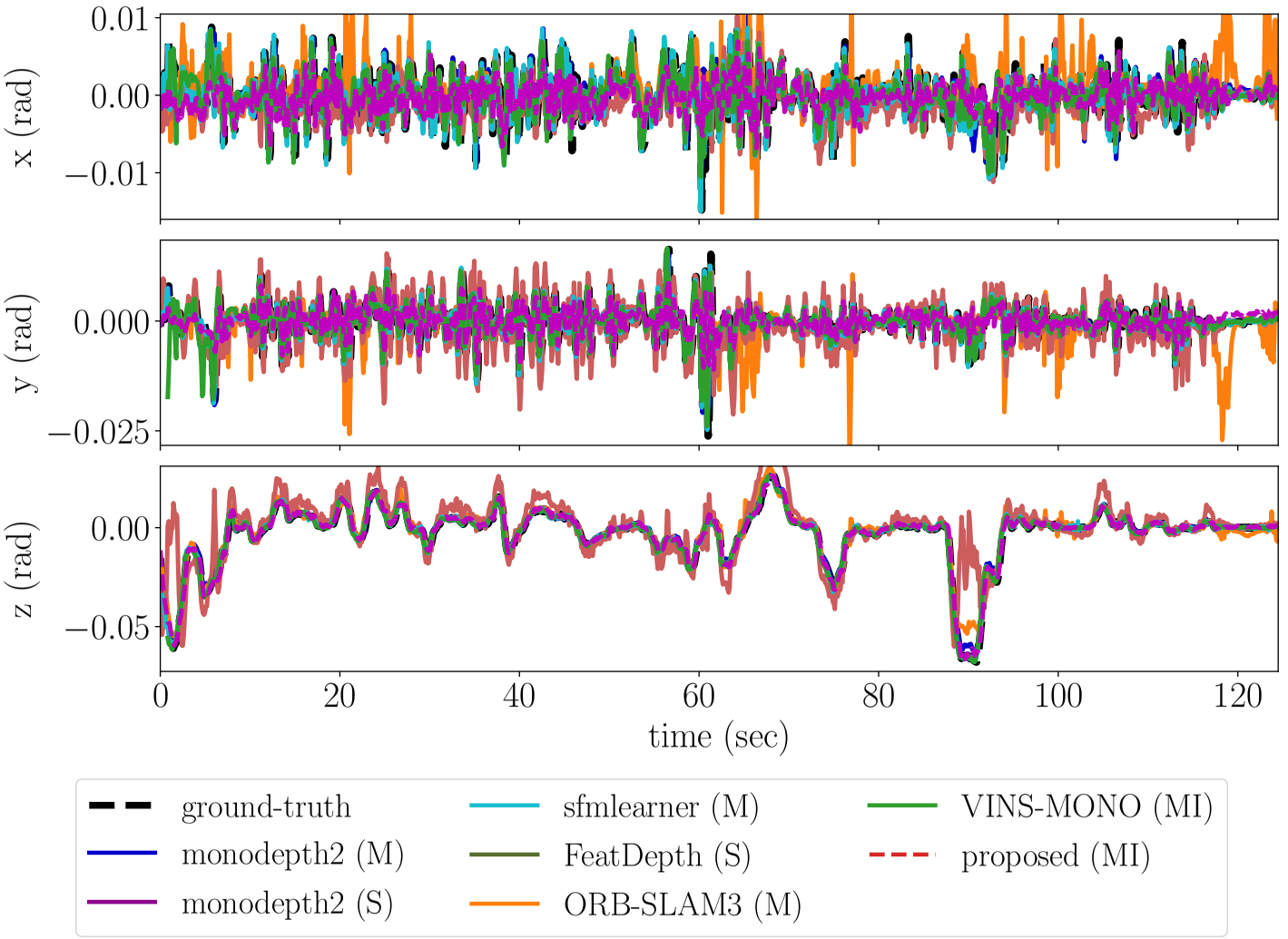


Figure 4.13: Relative pose estimation of angular motion in odometry 10.

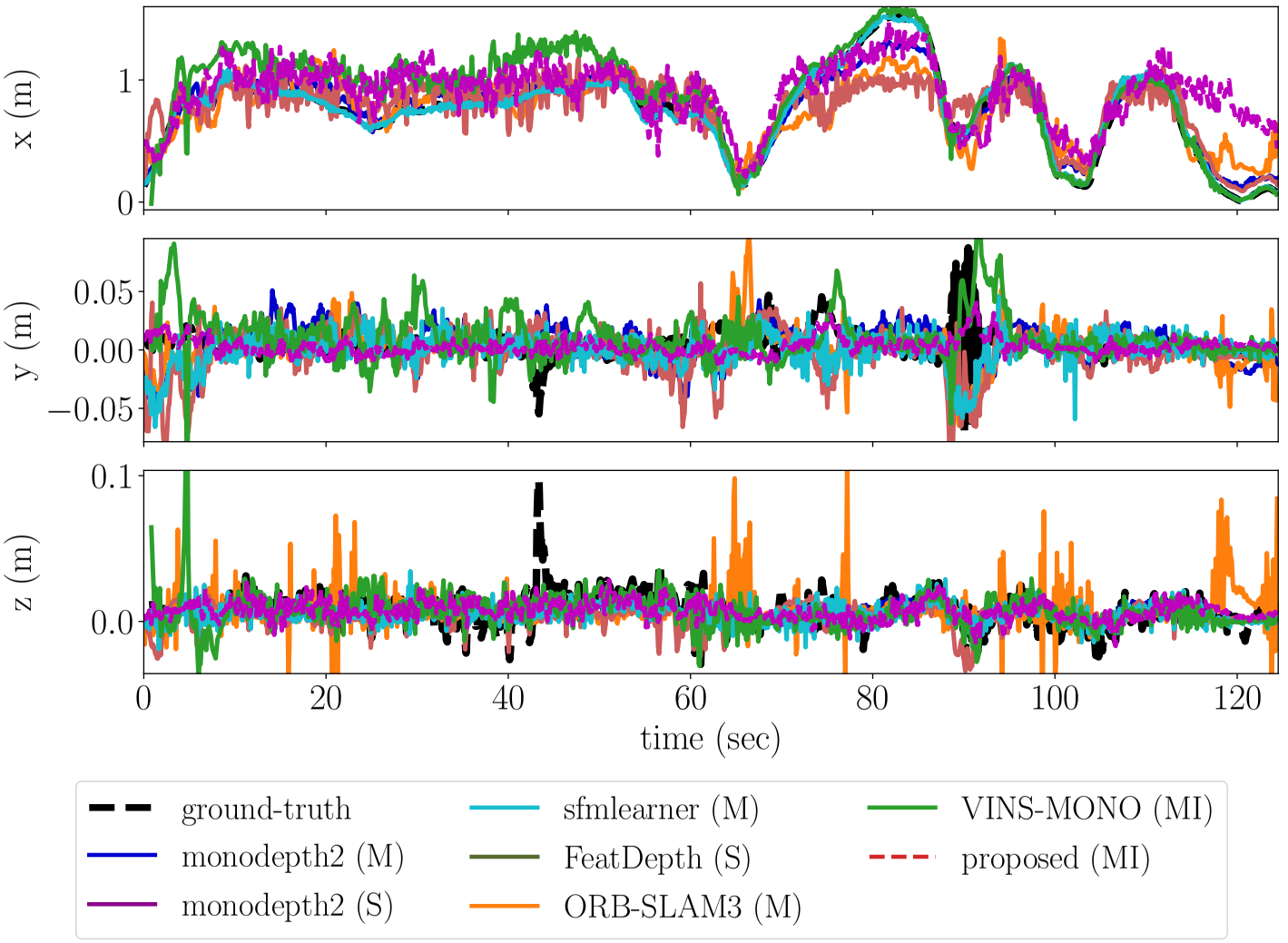


Figure 4.14: Relative pose estimation of linear motion in odometry 10.

Velocity prediction

The proposed method predicts the velocity using the finite differential as in equation (3.7). In this section, I calculate the velocity using the equation (3.7) using the trained pose network assuming trained. To check the effect of the smoothness, I calculate the velocity both at the single point and moving average concept like training step. In fig. 4.15, the forward and sideway directional velocity seems to be good. The prediction performance is degraded at 0-5 seconds and 120-160 seconds of the odometry 09 and 110-125 seconds of the odometry 10. In that time, the ego-motion prediction performance is also degraded, as in fig. 4.12 and 4.14. In addition, the raw velocity prediction result, the blue line in fig. 4.15 is a little noisy, although the ego-motion is converged.

Gravity prediction

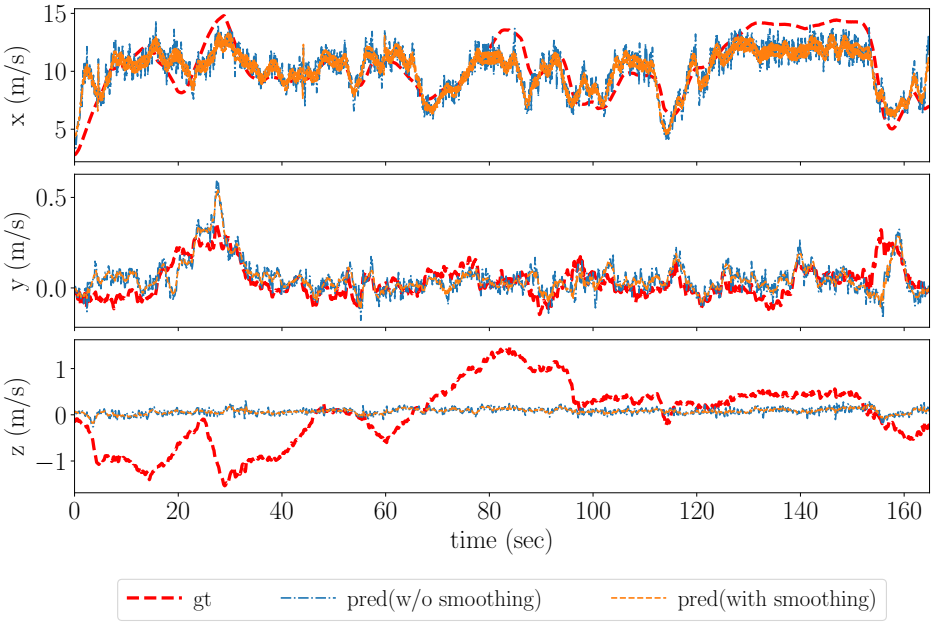
The proposed method predicts the gravity in the body coordinate. In this section, I compare the ground-truth gravity calculated from the ground-truth ego-motion of the vehicle. Fig 4.16 shows the gravity prediction result.

Computational time

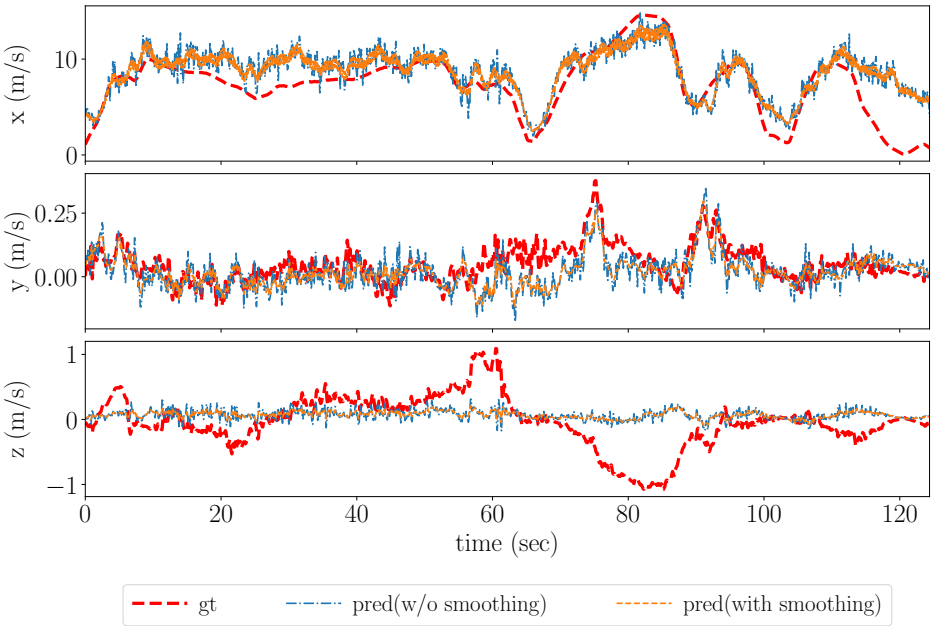
I calculate the computational time in the proposed method. I use the 512×160 image as an input. The computational power is i7-5790K@3.50GHz and GTX2080ti. The proposed method is implemented as the tensorflow library. To check the bottleneck of the parallel computation, the batch size is set as 1 and 32.

	# of params	GPU+CPU		CPU only	
		batch/1	batch/32	batch/1	batch/32
depth	14,338,836	9.70 ms	1.77 ms	39.75 ms	37.07 ms
odometry	24,050,511	13.86 ms	3.03 ms	30.35 ms	19.31 ms
pose	23,371,463	12.79 ms	3.14 ms	30.08 ms	19.35 ms

Table 4.6: The runtime of the proposed method during inference. Pose network is the odometry network that removes the decoder of gravity and bias.

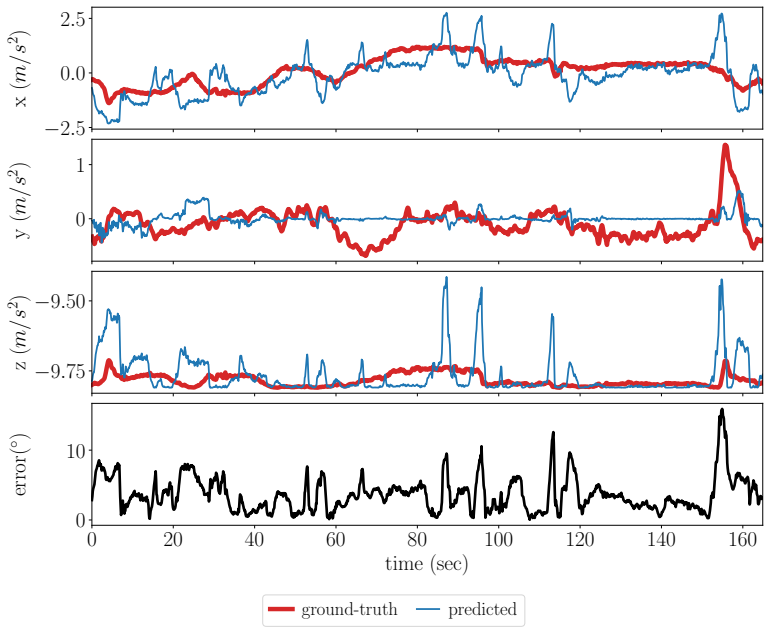


(a) Odometry 09

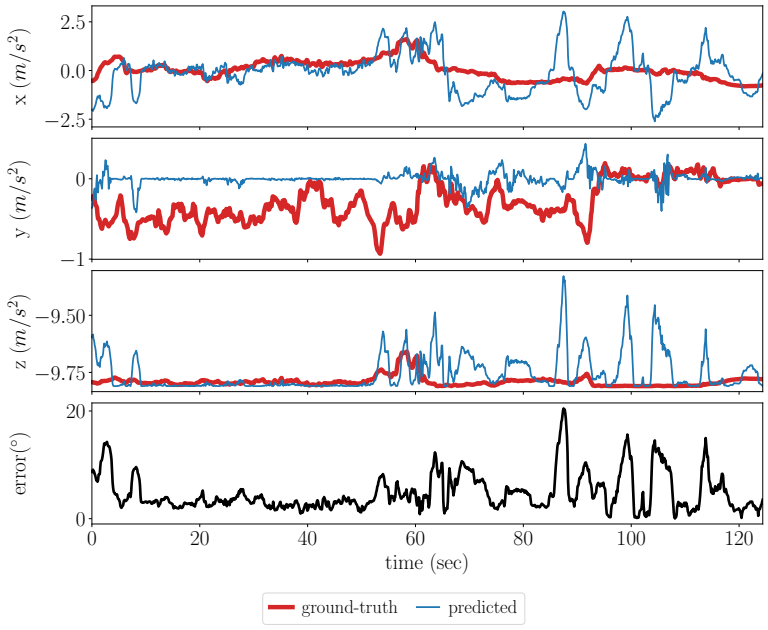


(b) Odometry 10

Figure 4.15: The velocity prediction result using the proposed method in the IMU body coordinates. (x: forward direction, y: left direction, z: upward direction)



(a) Odometry 09



(b) Odometry 10

Figure 4.16: The gravity prediction result using the proposed method. The error is the angle of predicted gravity and ground-truth gravity. (x: forward direction, y: left direction, z: upward direction)

4.3 Experimental Validation in the Indoor Dataset

4.3.1 Dataset collection

In addition to the KITTI dataset, I validate the proposed algorithm in the indoor environment. For that, the device was built as in fig. 4.17 and equipped on the car. The car was driving in the underground parking lots to collect the dataset. The detailed specifications of the device are in table 4.7.

(a) The device for data collection

(b) The example of the collected image

Figure 4.17: The device to collect the data and example of collected data of the indoor dataset.

	Model name	**Freq.**	**etc**
Camera	mvBlueCOUGAR	10Hz	Stereo system
IMU	Microstrain 3DM-G3X-25	250Hz	
LiDAR	Velodyne VLP32C UltraPuck	10Hz	

Table 4.7: The list of specifications of the device to collect the data.

Like the KITTI dataset, I utilize a single monocular camera and IMU sensor during both the training and inference step for the proposed method. The stereo and LiDAR are utilized to generate an accurate depth and pose information to validate the proposed method.

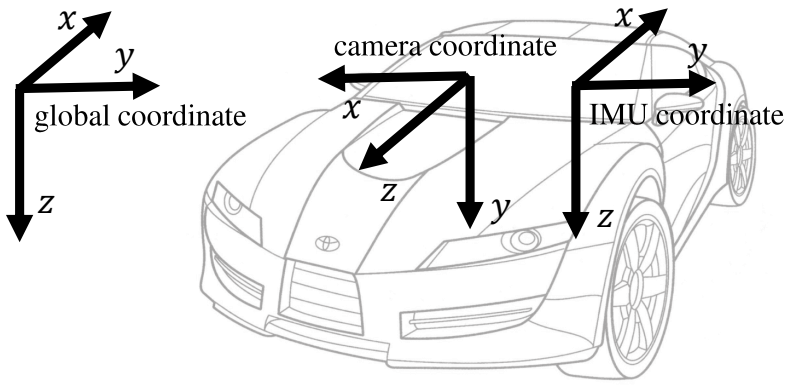
I would like to express my appreciation to my colleague Changhyeon Kim to provide great dataset without any hesitation before the dataset is published.

4.3.2 Dataset detail

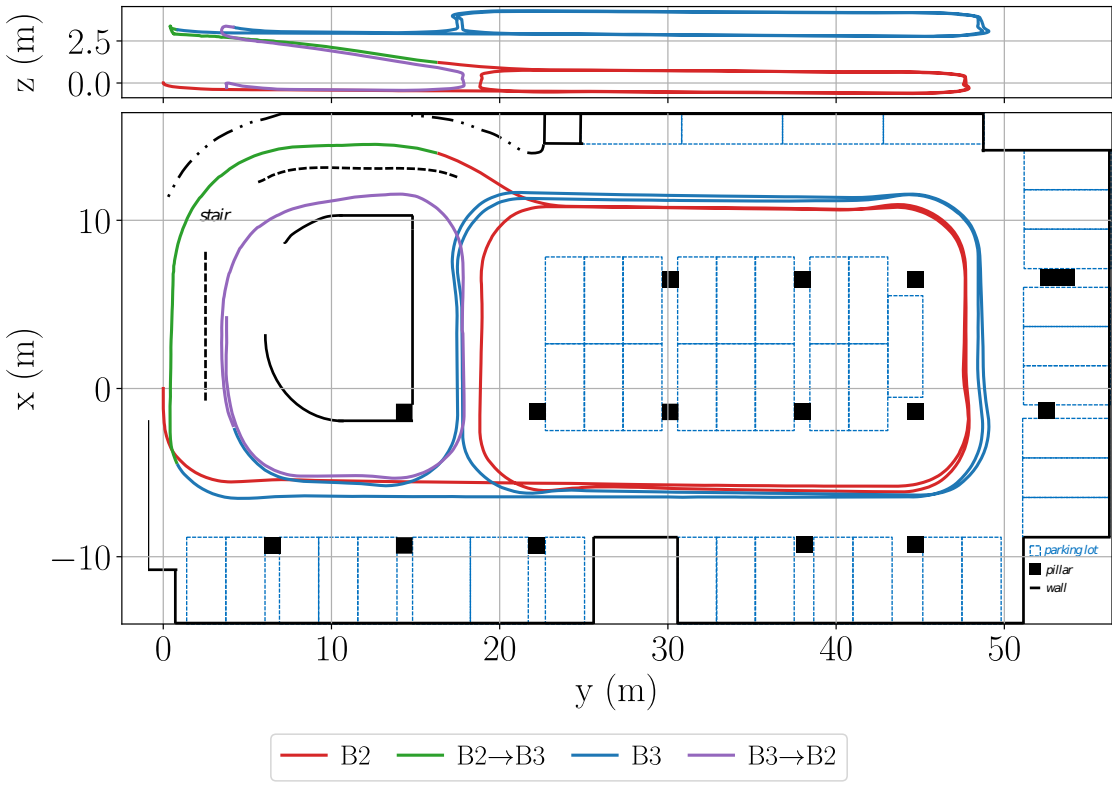
Since no ground-truth data exists, I generate the reference information using the stereo/LiDAR sensors. For the reference trajectory, the stereo visual-inertial SLAM algorithm is utilized. In detail, I tried stereo-inertial ORBSLAM3, stereo ORBSLAM3, and stereo-inertial VINS-MONO using various parameters. Then, I select the best trajectory as a reference trajectory. In addition, the obtained trajectories are plotted on the floor plan of the building for qualitative evaluation.

The car attached to the device was driven in the underground parking area of Seoul National University, building 39. For training purposes, four driving data at the underground parking area were additionally collected in building 39. Each driving data contains 1000-1600 images in approximately 2 minutes. Fig. 4.18 shows the trajectory of the test driving dataset.

For depth data, LiDAR data is utilized to generate the reference depth for validation. Since the collected LiDAR and image are not captured at the same time, I compensate for the time difference between the LiDAR and the image using the reference trajectory obtained above. However, due to the error of the trajectory prediction, some drift of depth prediction exists. In addition, the depth of the moving object may be inaccurate. Thus, in this experiment, I only perform a qualitative study about depth prediction.



(a) The definition of coordinates



(b) The reference trajectory

Figure 4.18: The reference trajectory of test driving in the underground parking area of Seoul National University, building 39. The vehicle starts at $(0, 0, 0)$, circles two times around at B2 (red), goes to B3 (green), circles two times around at B3 (yellow), and goes back to B2 (purple). In the floor plane, the black rectangle box is a pillar, a blue dotted line is a parking area, and the solid black line is a wall.

4.3.3 Pose performance validation

For validation, the trajectory generated by stereo-inertial ORBSLAM3 is provided for reference trajectory. For the comparison with the classical navigation methods, ORBSLAM3[8] and VINSMONO[9] are performed with monocular-visual mode. I turn on/off loop-closing module of both methods since this dataset repeatedly moves across some regions. For comparison with the learning-based method, I train monodpeth2[7], both monocular denoted as monodepth2(M) and stereo denoted as monodepth2(S), using the code provided by the author of [7] in Github with the default parameter. During training, same data are used as the proposed method, but IMU measurements are dropped for both monodepth2(M) and monodepth2(S), and stereo sequences are given instead of monocular sequences for monodepth2(S).

Fig. 4.19 shows the overall trajectory. It seems that classical methods have better prediction result compared with the learning-based method. All monocular-inertial methods, including the proposed method, can predict the scale of the trajectory.

For deep analysis, I split the trajectory into two parts. The first trajectory starts from B2, rotates B2 two times, and then goes into B3, as in fig. 4.20. The last trajectory starts from B3, rotates B3 two times, and then goes into B2, as in fig. 4.21.

For the first trajectory as in fig. 4.20, the proposed method shows drifts at the end of the trajectory, in which the car moves from B2 to B3. In the other regions, the proposed method shows a competitive result compared with classical monocular visual-inertial methods.

For the second trajectory as in fig. 4.21, the proposed method has continual drifts along the z-axis. However, it shows comparable results compared with classical monocular visual-inertial methods considering the top-down view. It is because the proposed method drifts along the pitch angle with comparable prediction along other axes.

In comparison with monodepth2(M) and monodepth2(S), the proposed method shows more accurate trajectory prediction, especially yaw-direction prediction. In addition, the scale of the proposed method is accurate compared with the monodepth2(S).

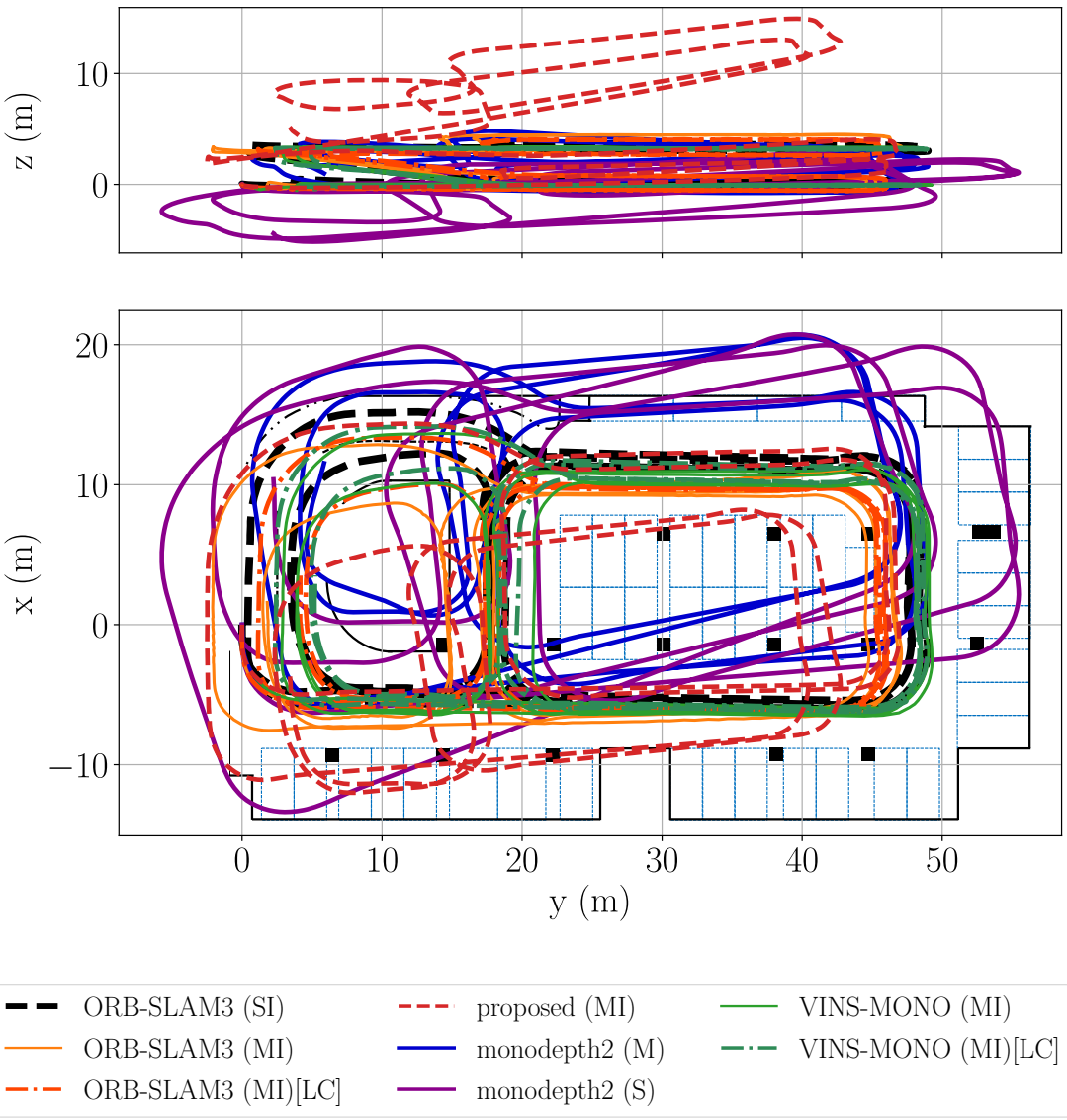


Figure 4.19: The predicted trajectories in the indoor dataset (whole). (·) next to the method denotes the training data for the deep-learning-based method or the operating data for the classical navigation: (SI) is stereo sequence with IMU measurements, (S) is stereo sequence, (MI) is monocular sequence with IMU measurements, and (M) is monocular sequence. The method with [LC] has loop closing ability. For monodepth2(M), because no scale is estimated, the scale is taken from *ORB-SLAM3(SI)* as in (4.11). Each trajectory is aligned by fixing the initial point at origin.

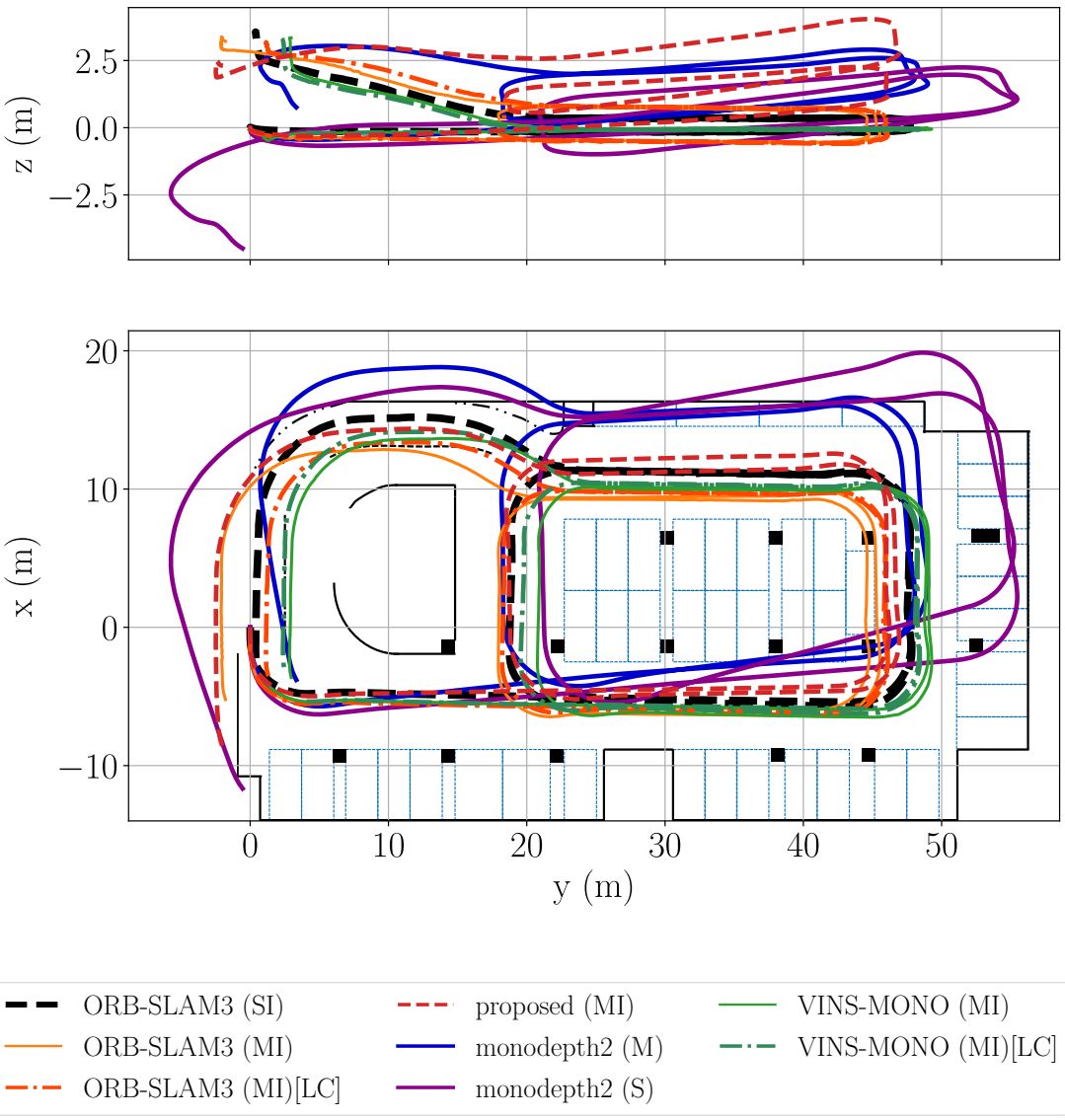


Figure 4.20: The predicted trajectories in the indoor dataset (first part: B2, B2→B3). (·) next to the method denotes the training data for the deep-learning-based method or the operating data for the classical navigation: (SI) is stereo sequence with IMU measurements, (S) is stereo sequence, (MI) is monocular sequence with IMU measurements, and (M) is monocular sequence. The method with [LC] has loop closing ability. For monodepth2(M), because no scale is estimated, the scale is taken from *ORB-SLAM3(SI)* as in (4.11). Each trajectory is aligned by fixing the initial point at origin.

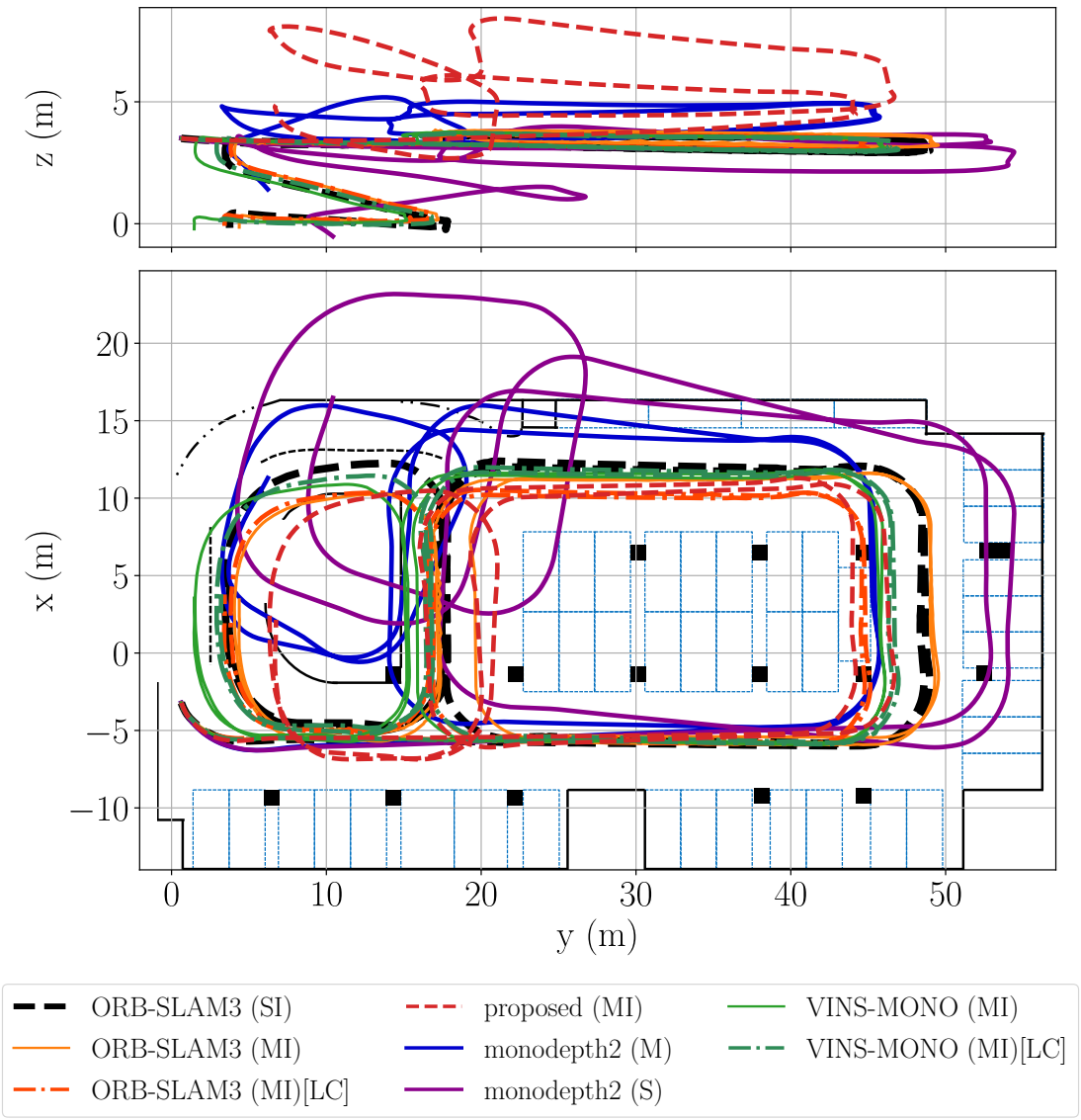


Figure 4.21: The predicted trajectories in the indoor dataset (second part: B3, B3→B2). (·) next to the method denotes the training data for the deep-learning-based method or the operating data for the classical navigation: (SI) is stereo sequence with IMU measurements, (S) is stereo sequence, (MI) is monocular sequence with IMU measurements, and (M) is monocular sequence. The method with [LC] has loop closing ability. For monodepth2(M), because no scale is estimated, the scale is taken from *ORB-SLAM3(SI)* as in (4.11). Each trajectory is aligned by fixing the initial point at the prediction point of ORB-SLAM3(SI).

Gravity prediction

The proposed method predicts the gravity in the body coordinate. In this section, I compare the reference gravity calculated from the reference ego-motion obtained by stereo-inertial SLAM. Fig 4.22 shows the gravity prediction result. The proposed method seems to follow the tendency of the reference gravity, but the error grows at 175-200 seconds. At that time, the vehicle moves from B3 to B2, the purple line in fig. 4.18.

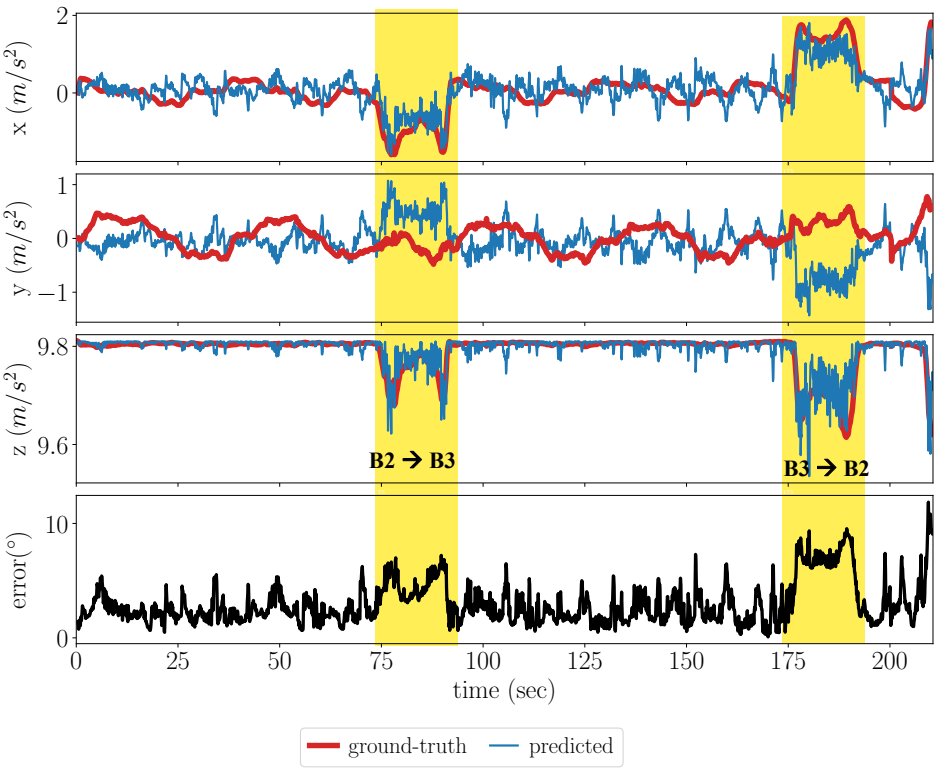
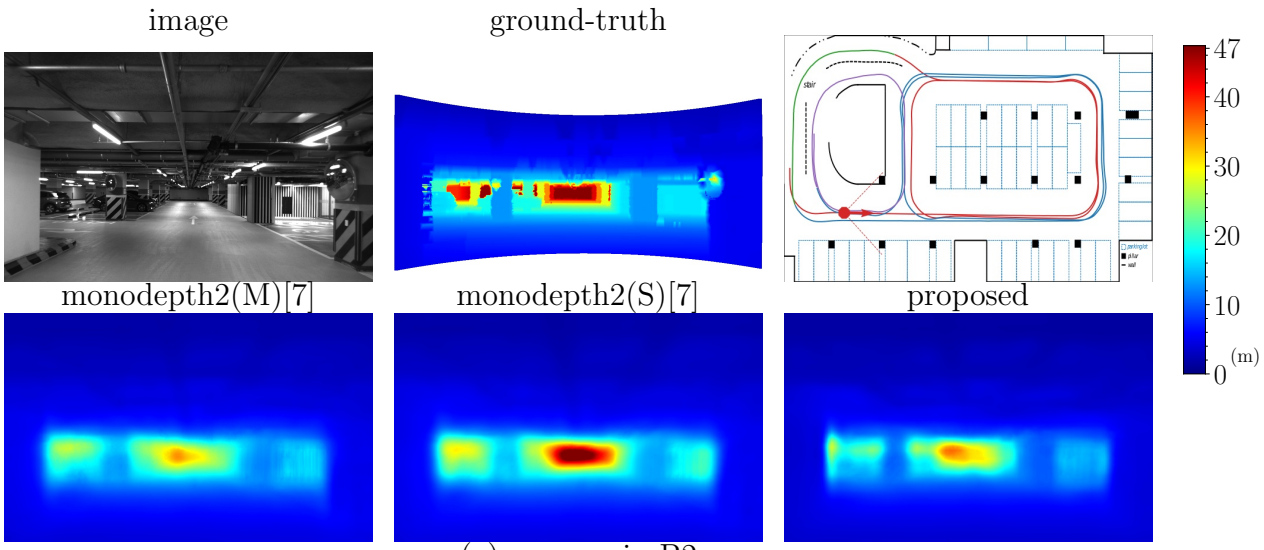


Figure 4.22: The gravity prediction result in the indoor dataset.

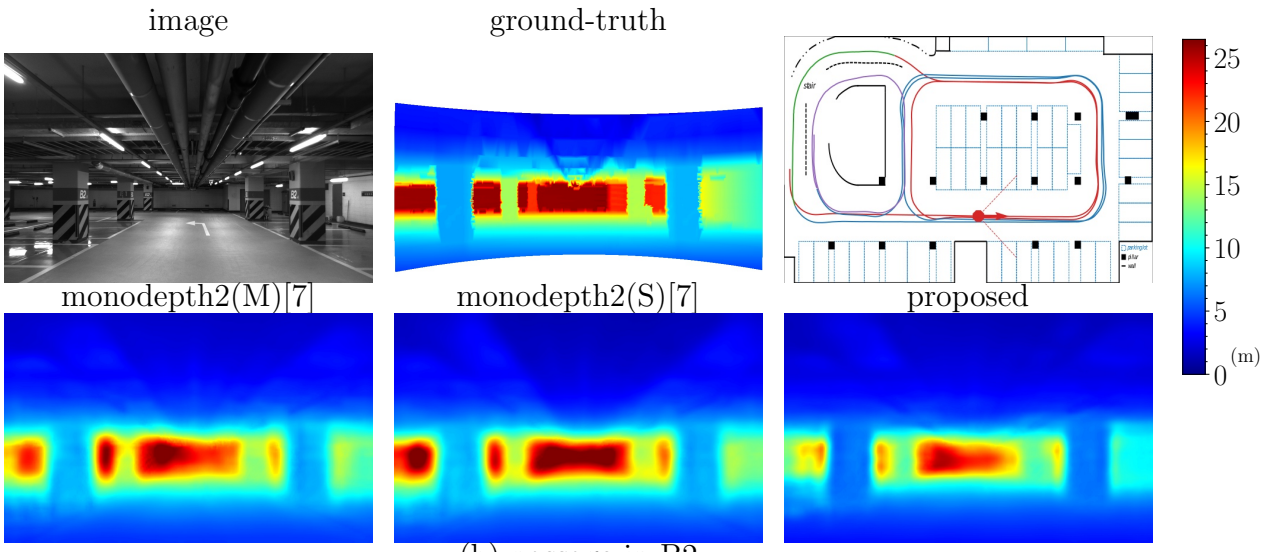
4.3.4 Depth performance validation

For validation, I process the LiDAR information to obtain the sparse depth. For better visualization, I perform bilinear interpolation using the sparse depth, so some points seem to be irregular. For comparison, I learn monodpeth2[7]. I trained the network of monodepth2 using the author's provided code with the default parameter and the same dataset for both monocular and stereo methods.

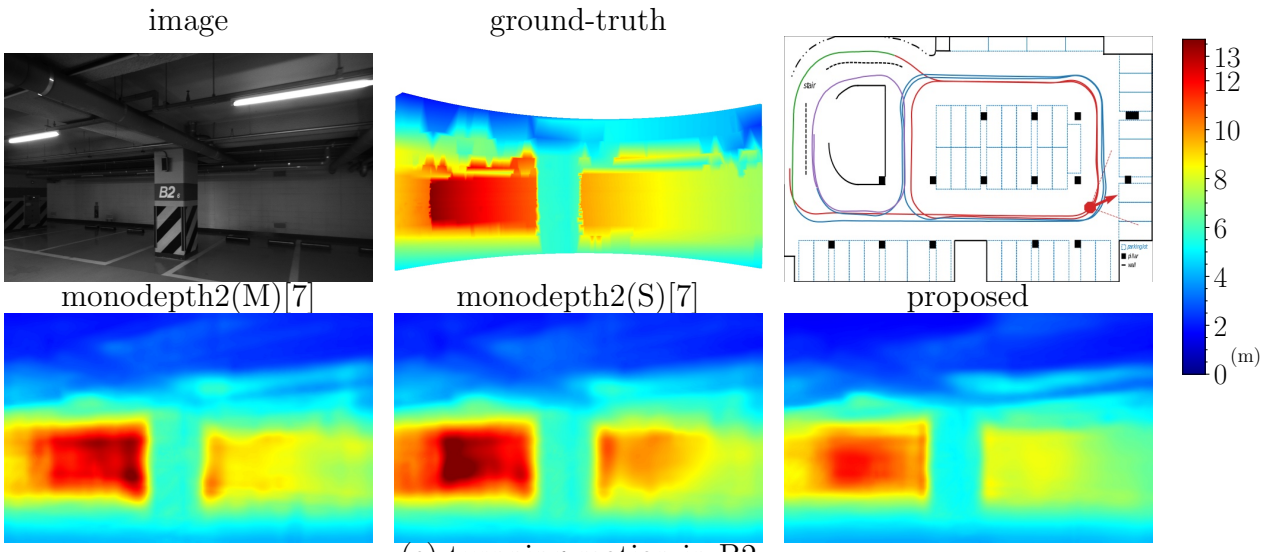
Fig. 4.23 shows the depth prediction result of the proposed method and monodepth2 provided by the authors of [7]. Since monodepth2 with monocular training denoted as monodepth2(M) cannot predict the metric scale, I *take the metric scale* from the LiDAR measurement as in (4.6). The proposed method can predict geometric information such as pillars (a)/(b)/(c)/(h)(i), cars (h), entrances/exits (e)/(k), and walls (d)/(f) like monodepth2. The proposed method predicts the scale from monocular images and IMU measurements, but monodepth2 needs *stereo* images during the training step, otherwise it cannot predict the scale information.



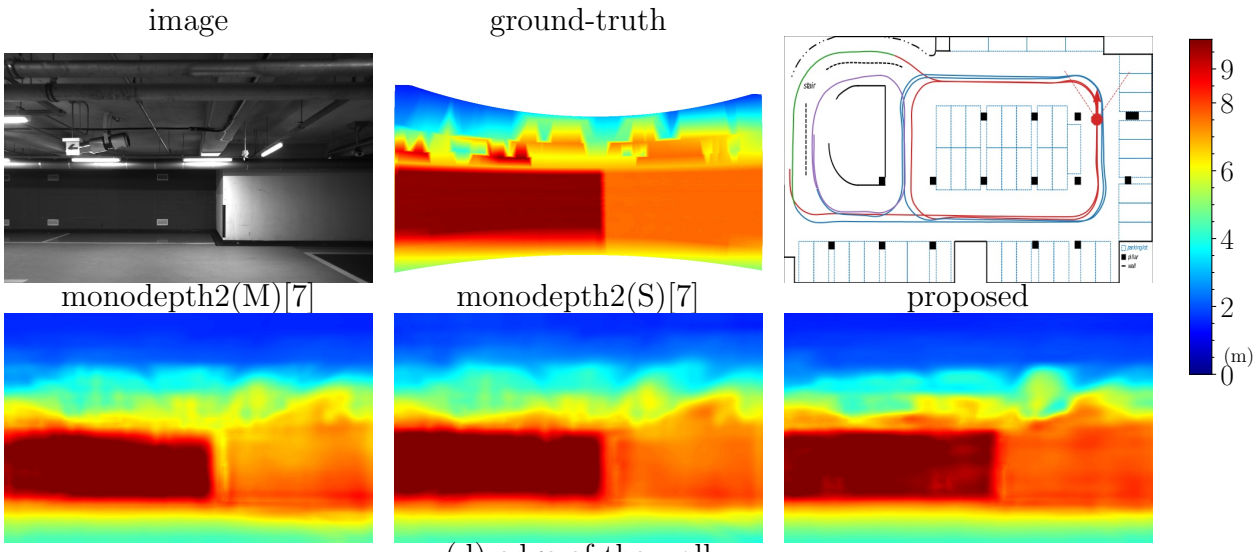
(a) passage in B2



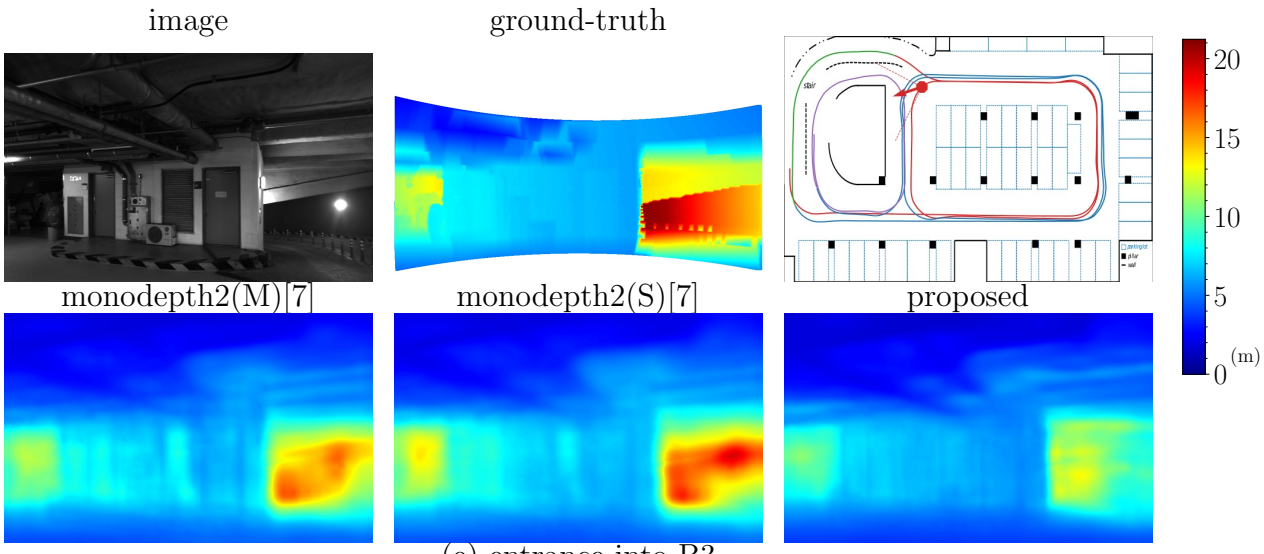
(b) passage in B2



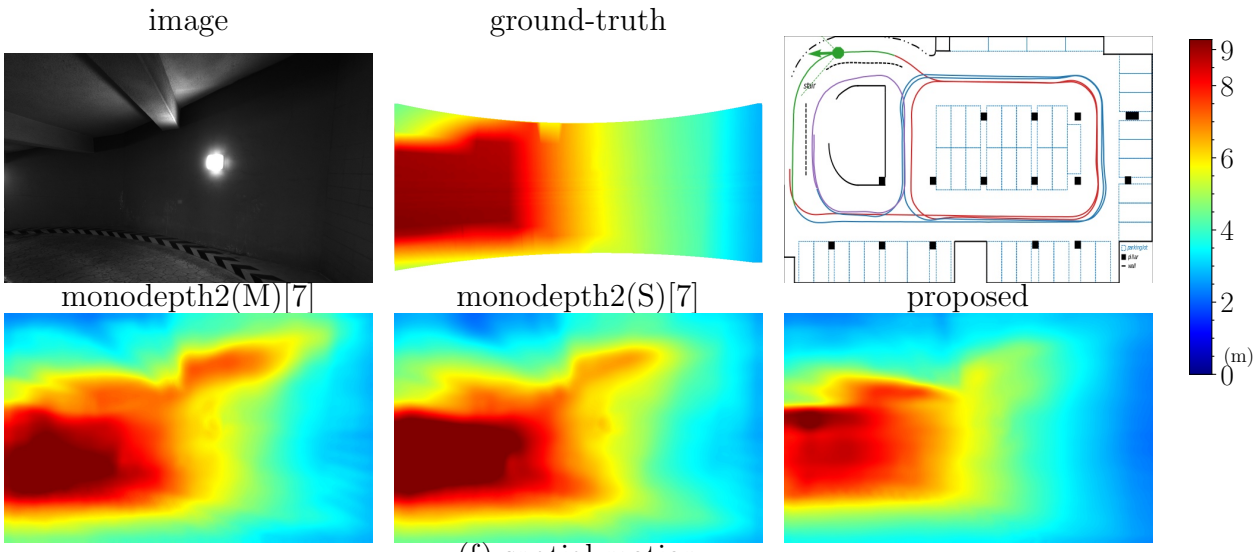
(c) turnning motion in B2



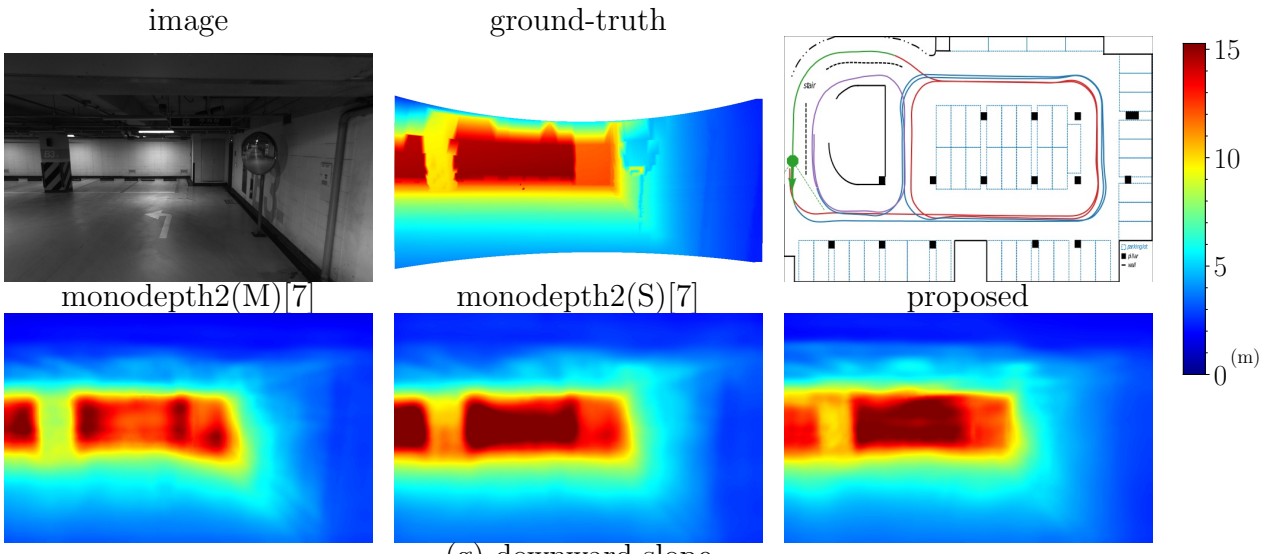
(d) edge of the wall



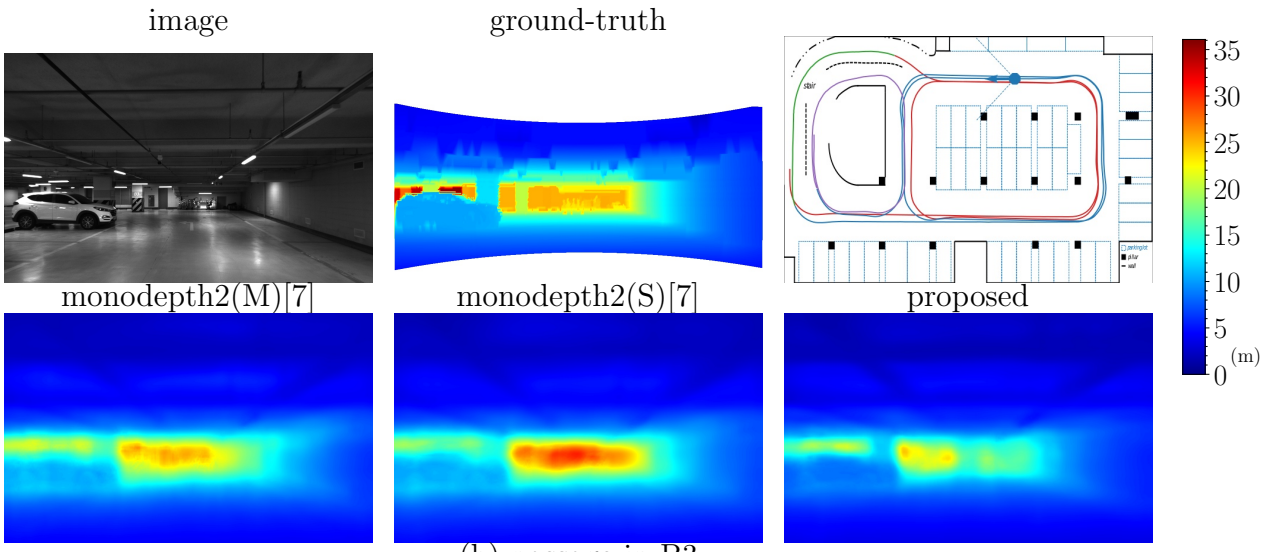
(e) entrance into B3



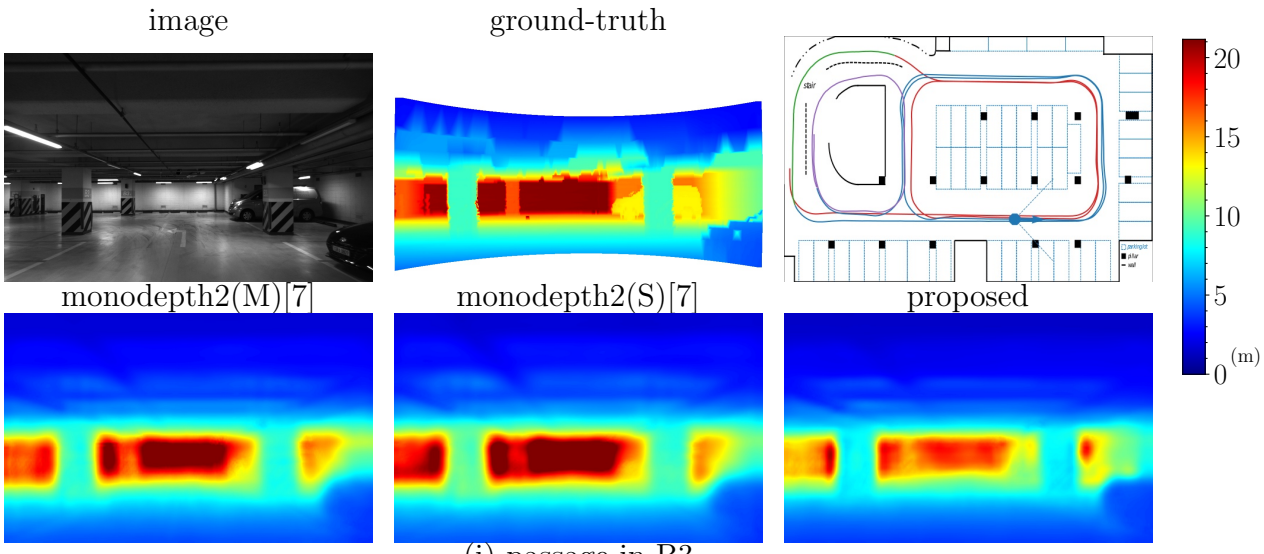
(f) spatial motion



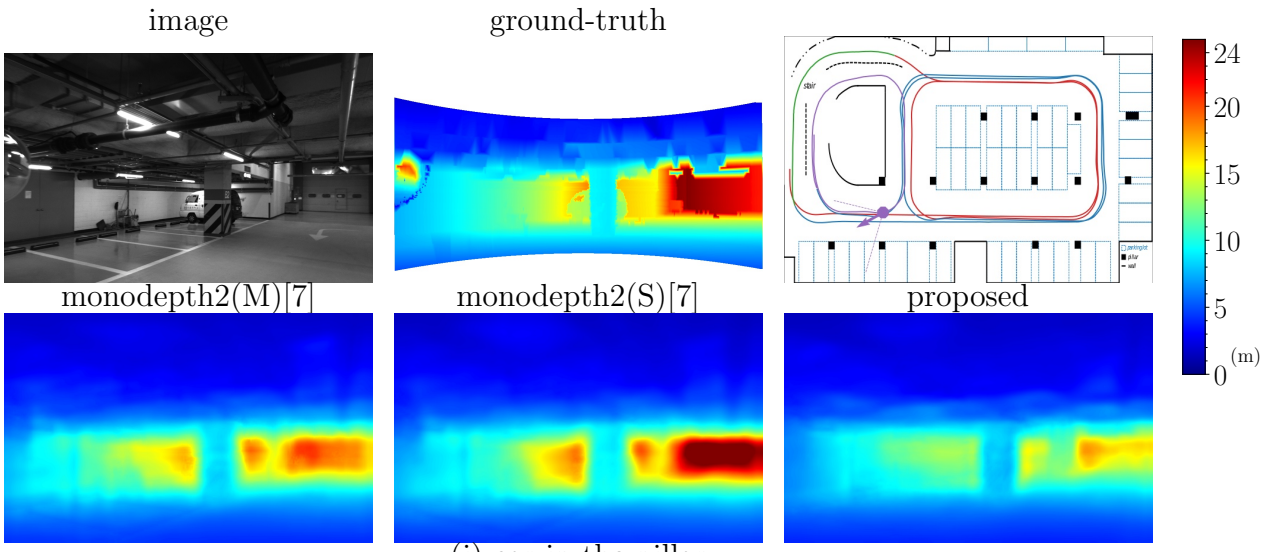
(g) downward slope



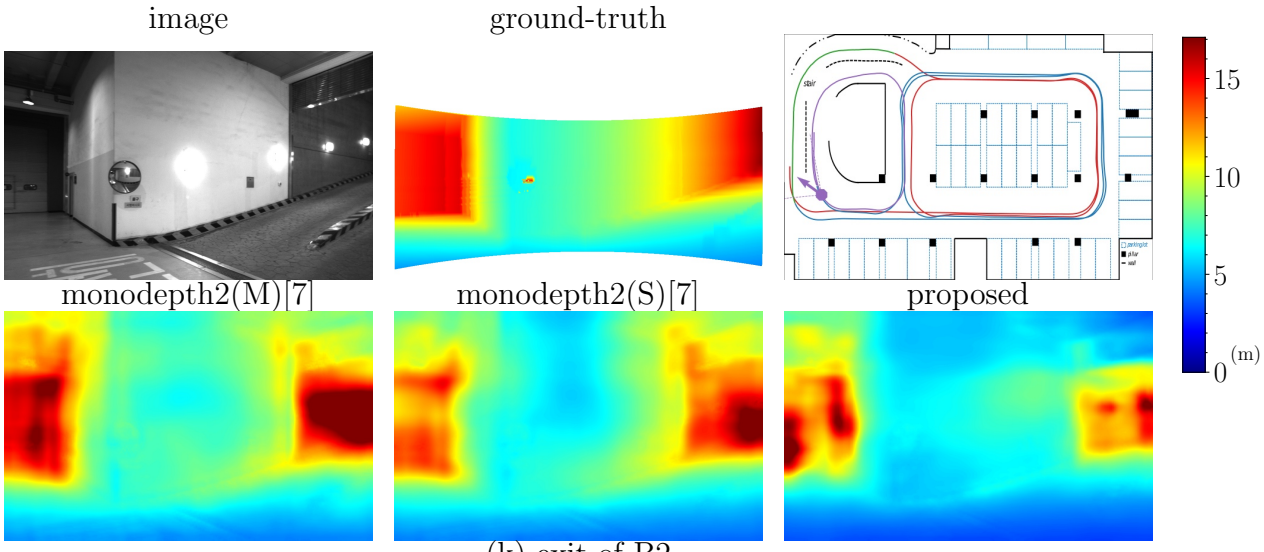
(h) passage in B3



(i) passage in B3



(j) car in the pillar



(k) exit of B2

Figure 4.23: The depth prediction result on the indoor dataset. The depth of ground-truth is generated from the LiDAR data with bilinear interpolation, and monodepth2 is trained by me using the author's provided code. $(\cdot)$ denotes the training data: M is monocular image, S is stereo image, and MI is monocular image with IMU measurement. Since monodepth2(M) estimates no scale information, the scale of it is *taken from the LiDAR* as in equation (4.6).

5

Conclusion

In this dissertation, I propose self-supervised monocular depth estimation and odometry which addresses the scale ambiguity issue with raw IMU measurements. I design the loss function and network architecture to learn the scale information from IMU measurements. I show that the proposed method provides the estimated scale with comparable performance in the KITTI dataset and the additional experiment using an actual vehicle.

The proposed method, like self-supervised monocular methods, can train the network using the same type of data used for the inference, i.e., the proposed method can train from the data collected during the inference. This suggests the possibility to extend the proposed method to the online learning framework, in which the robot/vehicle learns the surrounding environments by itself during the inference step without additional device setup. These characteristics can help improve the estimation performance especially when the robot confronts new environments.

References

[1] 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*, vol. 27, pp. 2366–2374, 2014.

[2] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[3] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2002–2011.

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.

[5] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1983–1992.

[6] R. Wang, S. M. Pizer, and J.-M. Frahm, "Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5555–5564.

[7] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, 2021.

[9] 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.

[10] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[11] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni, "Selective sensor fusion for neural visual-inertial odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 542–10 551.

[12] L. Han, Y. Lin, G. Du, and S. Lian, "Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6906–6913.

[13] Y. Almalioglu, M. Turan, M. R. U. Saputra, P. P. de Gusmão, A. Markham, and N. Trigoni, "Selfvio: Self-supervised deep monocular visual–inertial odometry and depth estimation," *Neural Networks*, vol. 150, pp. 119–136, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608022000752

[14] P. Wei, G. Hua, W. Huang, F. Meng, and H. Liu, "Unsupervised monocular visual-inertial odometry network." in *IJCAI*, 2020, pp. 2347–2354.

[15] E. J. Shamwell, K. Lindgren, S. Leung, and W. D. Nothwang, "Unsupervised deep visual-inertial odometry with online error correction for rgb-d imagery," *IEEE Trans-*

actions on Pattern Analysis and Machine Intelligence, vol. 42, no. 10, pp. 2478–2493, 2020.

[16] K. Sartipi, T. Do, T. Ke, K. Vuong, and S. I. Roumeliotis, "Deep depth estimation from visual-inertial slam," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 038–10 045.

[17] A. Wong, X. Fei, S. Tsuei, and S. Soatto, "Unsupervised depth completion from visual inertial odometry," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1899–1906, 2020.

[18] J. Choi, D. Jung, Y. Lee, D. Kim, D. Manocha, and D. Lee, "Selftune: Metrically scaled monocular depth estimation through self-supervised learning," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6511–6518.

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] L. Wang, J. Zhang, O. Wang, Z. Lin, and H. Lu, "Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 541–550.

[22] J.-H. Lee and C.-S. Kim, "Monocular depth estimation using relative depth maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9729–9738.

[23] R. Garg, V. K. B.G., G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Computer Vision – ECCV 2016*, B. Leibe,

J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 740–756.

[24] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 270–279.

[25] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2015, pp. 2017–2025.

[26] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on computational imaging*, vol. 3, no. 1, pp. 47–57, 2016.

[27] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, and L. Lin, "Single view stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 155–163.

[28] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 842–857.

[29] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[30] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 340–349.

[31] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[32] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[33] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3d packing for self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[34] A. CS Kumar, S. M. Bhandarkar, and M. Prasad, "Depthnet: A recurrent neural network architecture for monocular depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[36] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, "Generative adversarial networks for unsupervised monocular depth prediction," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[37] A. CS Kumar, S. M. Bhandarkar, and M. Prasad, "Monocular depth prediction using generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 300–308.

[38] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni, "Ganvo: Unsupervised deep monocular visual odometry and depth estimation with

generative adversarial networks," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 5474–5480.

[39] T. Feng and D. Gu, "Sganvo: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4431–4437, 2019.

[40] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5077–5086.

[41] S. Li, F. Xue, X. Wang, Z. Yan, and H. Zha, "Sequential adversarial learning for self-supervised deep visual odometry," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2851–2860.

[42] R. Ji, K. Li, Y. Wang, X. Sun, F. Guo, X. Guo, Y. Wu, F. Huang, and J. Luo, "Semi-supervised adversarial monocular depth estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2410–2422, 2019.

[43] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "Towards real-time unsupervised monocular depth estimation on cpu," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5848–5854.

[44] J. Liu, Q. Li, R. Cao, W. Tang, and G. Qiu, "Mininet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 166, pp. 255–267, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271620301544

[45] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6101–6108.

[46] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[48] J. Gui, D. Gu, S. Wang, and H. Hu, "A review of visual inertial odometry from filtering and optimisation perspectives," *Advanced Robotics*, vol. 29, no. 20, pp. 1289–1301, 2015. [Online]. Available: https://doi.org/10.1080/01691864.2015.1057616

[49] C. Chen, H. Zhu, M. Li, and S. You, "A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives," *Robotics*, vol. 7, no. 3, 2018. [Online]. Available: https://www.mdpi.com/2218-6581/7/3/45

[50] G. Huang, "Visual-inertial navigation: A concise review," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 9572–9582.

[51] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 298–304.

[52] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[53] A. Zihao Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5391–5399.

[54] P. Kim, H. Lim, and H. J. Kim, "Visual inertial odometry with pentafocal geometric constraints," *International Journal of Control, Automation and Systems*, vol. 16, no. 4, pp. 1962–1970, 2018.

[55] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. [Online]. Available: https://doi.org/10.1177/0278364914554813

[56] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[57] J. Shi *et al.*, "Good features to track," in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.

[58] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[59] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[60] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation." Georgia Institute of Technology, 2015.

[61] ——, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[62] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[63] 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*. IEEE, 2012, pp. 3354–3361.

[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[65] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[66] S. Lee, J. Lee, B. Kim, E. Yi, and J. Kim, "Patch-wise attention network for monocular depth estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 1873–1881.

[67] C. Shu, K. Yu, Z. Duan, and K. Yang, "Feature-metric loss for self-supervised learning of depth and egomotion," in *European Conference on Computer Vision*. Springer, 2020, pp. 572–588.

[68] X. Lyu, L. Liu, M. Wang, X. Kong, L. Liu, Y. Liu, X. Chen, and Y. Yuan, "Hr-depth: High resolution self-supervised monocular depth estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 2294–2301.

[69] Y. Meng, Y. Lu, A. Raj, S. Sunarjo, R. Guo, T. Javidi, G. Bansal, and D. Bharadia, "Signet: Semantic instance aided unsupervised 3d geometry perception," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2019, pp. 9810–9820.

[70] J. Zhou, Y. Wang, K. Qin, and W. Zeng, "Unsupervised high-resolution depth learning from videos with dual networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6872–6881.

[71] L. Wang, Y. Wang, L. Wang, Y. Zhan, Y. Wang, and H. Lu, "Can scale-consistent monocular depth be learned in a self-supervised scale-invariant manner?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 12 727–12 736.

[72] T.-W. Hui, "Rm-depth: Unsupervised learning of recurrent monocular depth in dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 1675–1684.

국 문 초 록

본 논문은 영상 및 관성 정보를 활용하여 딥러닝 기반의 스케일 예측 깊이 및 항법 추정을 자가지도식으로 학습하는 기법을 다룬다. 단안 카메라를 활용한 실제 활용에 있어서, 스케일 모호성은 중요한 문제이다. 추가적인 데이터를 활용하지 않는 자가지도식 학습 기법은 자가지도식 학습 기법은 스케일 모호성을 피할 수 없기 때문에, 최신 딥러닝 기반 기법은 이를 추가 센서 정보로 부터 스케일 정보를 학습하는 방식으로 해결해 왔다. 이러한 측면에서, 관성항법센서는 가볍고 저렴하다는 측면에서 다양한 이동형 플랫폼에서 많이 사용되어 오고 있다. 그러나, 자가지도식 학습 세팅에서 IMU로 부터 스케일을 학습하는 것은, 참값 정보로 부터 스케일을 학습하는 지도식 학습과는 다르게 도전적인 문제이다.

본 논문에서는 딥러닝 기반으로 스케일 정보를 추정할 수 있는 자가지도식 단안 영상관성 깊이 추정 및 항법 시스템을 제안한다. 특히, 자가지도식 세팅에서 단안 영상의 스케일 모호성을 해결하는 방법에 집중하여, 참값 깊이나 스테레오 이미지가 학습 데이터로 주어지지 않고 단순히 단안 영상 및 관성센서만 주어진 상황에서 스케일을 포함한 깊이 및 위치 추정 기법을 수행하였다. 제안한 기법은 학습 과정에서 end-to-end로 동작하며, 기존의 영상관성 항법 시스템의 도움을 받지 않아도 된다. 이를 위하여, 관성센서 값을 적분하는 손실 함수를 설계하고, 관성 센서를 입력으로 받아 관성센서의 바이어스 및 중력의 방향을 추정하는 네트워크를 설계한다. 또한, 관성센서 정보가 있는 상황에서도 활용할 수 있는 데이터 보강 기법을 제안하였고, 유명한 데이터셋인 KITTI 데이터셋에서 최신 학습기반 및 기존 관성영상항법 알고리즘과의 비교 및 추가 실험을 통해 비교할만한 성능을 보임을 검증하였다.

주요어: 기계학습, 깊이 추정, 영상관성항법, 자가지도식 학습

학 번: 2014-22512