Ph.D. Dissertation

Fusion of Image and Point-cloud Data for Autonomous Driving in Diverse Environments

다양한 환경에서의 자율 주행을 위한 이미지와 포인트 클라우드 데이터 융합 기술

February 2023

Department of Electrical and Computer Engineering
Seoul National University

Yurim Jeon

Ph.D. Dissertation

Fusion of Image and Point-cloud Data for Autonomous Driving in Diverse Environments

다양한 환경에서의 자율 주행을 위한 이미지와 포인트 클라우드 데이터 융합 기술

February 2023

Department of Electrical and Computer Engineering
Seoul National University

Yurim Jeon

Fusion of Image and Point-cloud Data for Autonomous Driving in Diverse Environments

다양한 환경에서의 자율 주행을 위한 이미지와 포인트 클라우드 데이터 융합 기술

지도교수 서 승 우

이 논문을 공학박사 학위논문으로 제출함

2023년 2월

서울대학교 대학원

전기 정보 공학부

전 유 림

전유림의 공학박사 학위 논문을 인준함

2023년 2월

위 원 장:	조 남 익
부위원장:	서 승 우
위 원:	김 성 우
위 원:	김 영 민
위 원:	김 창 수

Abstract

Autonomous driving is one of the main topics of robotics research and is becoming part of our lives as they are used in automobiles, indoor robots, military robots, and drones. Perception, which is the starting point of driving intelligence, is a process of generating the knowledge necessary for driving by interpreting data collected from sensors. The data collected by the sensors varies depending on the robot platform, terrain, and light or weather conditions. For autonomous driving, perception must stably generate accurate knowledge by receiving data that changes according to these various conditions. However, many existing studies have focused on benchmark competition to achieve the best performance under limited conditions. We break away from these laboratory-only studies and focus on the real world to develop robust algorithms that can operate in various conditions.

Sensors widely used in autonomous driving include cameras that collect images and LiDARs that collect point clouds. Images contain high-resolution color information and are sensitive to changes in illumination (e.g., sunlight and weather). Point clouds are robust to changes in illumination and provide distance information of 3D space but are of low resolution. The image and point cloud have complementary characteristics. Therefore, the fusion of two data can produce enhanced information that compensates for the shortcomings of each data.

We propose a fusion framework of images and point clouds for autonomous driving in diverse environments. The first step of the fusion framework is a one-to-one match of the two data. This is called image-to-point cloud registration and aims to align a 2D image with a 3D point cloud. The proposed algorithm is designed for robust operation under various terrain conditions. Therefore, it was demonstrated in automobiles on paved roads and UGVs on off-road driving.

The second step is to generate enhanced data using the two matched data. This is

called depth completion, which generates a depth image with high-resolution depth information using an image with high-resolution color information and a point cloud with low-resolution depth information. The proposed algorithm is designed to work reliably in various light and weather conditions, from sunrise to sunset, fog and rain, and camera corruption.

The last step is to estimate the traversability from the observed image and point cloud. This is called traversability estimation, and traversability is predicted by learning the robot's driving style. The proposed algorithm is designed to estimate different traversabilities with respect to the driving style of the robot platform, from large ATVs to small UGVs.

In this dissertation, we present an image and point cloud fusion framework that can operate in diverse real-world conditions. We evaluate the proposed algorithm through experiments on various platforms (automobile, large and small UGV), terrain (paved road, open fields, mountain), and light and weather conditions (morning, sunset, rain, fog). The proposed framework serves as a buffer to ensure the stability of further applications by receiving data that fluctuates depending on diverse conditions and generating stable and enhanced knowledge. Therefore, we expect this framework to be a foundation for robust autonomous driving.

Keyword: Autonomous driving, Sensor fusion, Deep learning, Image-to-point cloud registration, Depth completion, Traversability estimation

Student Number: 2017-25223

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivations

Are we ready for autonomous driving [2]? Autonomous driving has shown up around us before we could notice it, from self-driving cars that introduced the term "autonomous driving" to the robot vacuum cleaners we use every day and military robots that navigate the battlefield. Hence, it is important to understand autonomous driving. Autonomous driving supports various tasks (driving a car, cleaning the room, exploring the battlefield) by observing the surrounding environment, interpreting the observed data, and then planning actions and controlling robots without human intervention. There are five core modules for autonomous driving: sensing, perception, localization, planning, and control. A sensing module uses sensors to collect data on the surrounding environment. The most widely used sensors are cameras that collect images, LiDARs that collect point clouds, inertial measurement units (IMUs) that detect linear and rotational accelerations, and global positioning systems (GPS) that measure an robot's position. Perception and localization modules generate essential knowledge for autonomous driving by interpreting data collected from sensors. The perception module detects signs and objects and segments the data into meaningful clusters. The localization module builds a map from the data and uses it to estimate the robot's cur-

rent location. The planning module uses interpreted information to plan actions for a target task. Finally, a control module controls the robot to perform the planned action.

Most existing studies on autonomous driving have been conducted in limited conditions (e.g., data collected on a clear and bright day, machine platform driving on paved roads without a tremor). However, we may encounter noisy data, malfunctioning sensors, and dynamic natural conditions in the real world. Therefore, algorithms that are guaranteed to work under limited conditions cannot be used for autonomous driving in the real world. For an algorithm to be ready for autonomous driving, it must be able to operate robustly in various real-world conditions. Therefore, our goal is to develop algorithms that can work on a car driving on paved roads, on a UGV driving off-road, in morning and night conditions, from dusk to dawn, and on foggy and rainy days.

Our research starts with the data obtained from sensors, especially images and point clouds. The image obtained from the camera sensor consists of high-resolution RGB color information. However, it is vulnerable to changes in illuminance; hence, different data may be collected depending on sunlight or weather changes in the same scene. The point cloud obtained from the LiDAR sensor is a set of 3D points. It is less affected by changes in illuminance and provides distance information of 3D space. However, this data has inherently low resolution. The image and point cloud have distinct characteristics. The fusion of the two data is a powerful solution because it can generate accurate and robust integrated information that compensates for the shortcomings of each data.

The first step of fusion is the one-to-one matching of the two data. This is called image-to-point cloud registration, and its goal is to estimate a transformation matrix that aligns the image and point cloud in the same coordinate system. Next, improved data can be generated using the two matched data, which is called depth completion. The goal of depth completion is to generate a depth image with high-resolution depth information using an image with high-resolution color information and a point cloud

with low-resolution depth information. Finally, the space that the robot can traverse can be estimated by integrating the image and point cloud, which is called traversability estimation. In the integration process, semantic information is extracted from the image and surface slope information is extracted from the point cloud, and traversability is estimated based on this information.

In this dissertation, we propose a fusion framework of image and point cloud for autonomous driving in diverse environments. The proposed framework consists of three algorithms. The first is an image-to-point cloud registration algorithm that is feasible under different terrain conditions. The algorithm can be run on automobiles on paved roads and UGVs on off-roads and has been validated in two registration applications: image-based localization and camera-LiDAR extrinsic calibration. The second is a depth completion algorithm that can operate reliably in different light and weather conditions. This algorithm can operate stably from morning to night, in fog and rain, and with camera corruptions. The last is a traversability estimation algorithm that can learn the driving style of the robot platform. This algorithm can estimate the appropriate traversability based on the physical characteristics of the robot platform and desired driving style (e.g., dirt priority driving, pothole avoidance).

1.2 Contributions and Outline of the Dissertation

1.2.1 EFGHNet: A Versatile Image-to-Point Cloud Registration Network for Extreme Outdoor Environment

We present an accurate and robust image-to-point cloud registration method that is viable in urban and off-road environments. Existing image-to-point cloud registration methods have focused on vehicle platforms along paved roads. Therefore, image-to-point cloud registration on UGV platforms for off-road driving remains an open question. Our objective is to find a versatile solution for image-to-point cloud registration. We present a method that stably estimates a precise transformation between an image

and a point cloud using a two-phase method that aligns the two input data in the virtual reference coordinate system (*virtual-alignment*) and then compares and matches the data to complete the registration (*compare-and-match*). Our main contribution is the introduction of divide-and-conquer strategies to image-to-point cloud registration. The *virtual-alignment* phase effectively reduces relative pose differences without cross-modality comparison. The *compare-and-match* phase divides the process of matching the image and point cloud into the rotation and translation steps. By breaking down the registration problem, it is possible to develop algorithms that can robustly operate in various environments. We performed extensive experiments on four datasets (Rellis-3D, KITTI odometry, nuScenes, and KITTI raw). Experiments cover a variety of situations in which image-to-point cloud registration is applied, from image-based localization in off-road environments to camera-LiDAR extrinsic calibration in urban environments. The experiments demonstrate that the proposed method outperforms the existing methods in accuracy and robustness.

1.2.2 ABCD: Attentive Bilateral Convolutional Network for Robust Depth Completion

We propose a point-cloud-centric depth completion method called attentive bilateral convolutional network for depth completion (ABCD). The proposed method uses LiDAR data and camera data to improve the resolution of the sparse depth information. Color images, which have been seen as fundamental to depth completion tasks, are inevitably sensitive to light and weather conditions. We designed an attentive bilateral convolutional layer (ABCL) to build a robust depth completion network under diverse environmental conditions. An ABCL efficiently learns geometric characteristics by directly leveraging a 3D point cloud and enhances the representation capability of sparse depth information by highlighting the core while suppressing clutter. The ABCD, with an ABCL as a building block, stably fills the void in sparse depth images even under unfamiliar conditions with minimum dependency on unstable camera sensors. There-

fore, the proposed method is expected to be a solution to depth completion problems caused by changes in the environment in which images are captured. Through comparative experiments with other methods using the KITTI [3] and VirtualKITTI2 [4] datasets, we demonstrated the outstanding performance of the proposed method in diverse driving environments.

1.2.3 Traversability Estimation Based on Footprint Supervision in an Off-road Environment

Traversability estimation is the task of estimating an area that the robot can travel. This task plays an important role in preventing the robot from crashing or overturning, particularly in off-road environments in the presence of scattered obstacles. Existing studies consider traversability to be explicit and approach a task by estimating a predefined traversable space. However, this approach cannot be a viable solution to the problem of finding robot-centric traversability in off-road environments. Off-road traversability cannot be explicitly defined and is implicitly determined by three factors: surface slope, semantic information, and robot platform. In this study, the footprints of the robot in expert driving are used, in which a human manually controls the robot. For this, we design a footprint-supervision module to predict the entire traversable space based on the footprints of the robot. Simultaneously, a new dynamic filter layer called the inter-modality joint-control kernel layer (IJKL) is proposed that self-manages the effects of the two inputs of the layer, RGB and the surface normal image, to extract features optimized for traversability estimation. The performance of the method is demonstrated through experiments on multiple datasets and the proposed method is confirmed to be a feasible traversability estimation method optimized for off-road environments.

Chapter 2

EFGHNet: A Versatile Image-to-Point Cloud Registration Network for Extreme Outdoor Environment

2.1 Introduction

Image-to-point cloud registration aims to find a transformation that aligns a 2D image and a 3D point cloud. To integrate 2D and 3D data, it is essential to understand the transformation between the two. Because of its importance, image-to-point cloud registration has been applied in various fields such as computer vision, robotics, and autonomous driving. One representative application is image-based localization, which estimates the pose of an image sensor under the world coordinate system of a point cloud. The other is camera-LiDAR extrinsic calibration, which estimates the relative pose between an image sensor and point cloud sensor mounted on the same platform. The main difference between the two tasks is that, in the case of image-based localization, we assume that the point cloud is pre-collected, the image sensor is mounted on the moving platform, and the goal is to estimate the location of an image sensor. However, in the case of camera-LiDAR extrinsic calibration, the aim is to match the image and point cloud by estimating the relative pose between the two sensors.

A classic approach to image-to-point cloud registration is to use matching algo-

rithms in the feature space of two data [5], [6]. These features can be descriptors (SIFT [7], DAISY [8]) or key points from both data. Perspective-n-point (PnP), iterative closest point (ICP), and RANSAC are commonly used matching algorithms. However, reliable estimation is difficult because the performance of the matching algorithm is strongly affected by the initial seed, which is the initial pose of the image and the point cloud. A recently emerged learning-based approach involves regressing transformations using convolutional neural networks (CNNs). Most studies use point clouds converted into depth images to take advantage of CNNs optimized for grid-format information processing [9], [10]. Therefore, CNNs can efficiently process cross-modality inputs and learn the associations between depth and RGB information. However, if the difference between the initial pose of the image and the point cloud is large, the depth image is not properly generated, making an accurate estimation difficult. To solve this problem, DeepI2P [11] interpreted the registration problem as a classification problem for classifying the point cloud projected on the camera frustum. However, this method is still not a reliable solution for large differences in the initial pose of the image and the point cloud.

The situations of a large difference in initial pose values often occur in off-road environments. Existing image-to-point cloud registration methods have focused on vehicle platforms along paved roads, and it is possible to assume a small oscillation in the platform. However, the methods under this assumption did not perform well in the case of an unmanned ground vehicle (UGV) platform driving off-roads, where the platform experiences large tremors. In such an off-road UGV environment, the image or point cloud collected from the onboard sensor undergoes large rotations or movements. Existing registration methods cannot adequately cope with such large noises and cause registration failures. This limitation led to the goal of finding a versatile solution for image-to-point cloud registration. We aim to develop an algorithm that can operate in various situations that robots may experience in an outdoor environment, from image-based localization of UGVs driving off-road to on-the-fly camera-LiDAR

extrinsic calibration of autonomous vehicles.

In this study, we propose an image-to-point cloud registration method called EFGH-Net (Fig. 2.1). We build a two-phase method that aligns the two input data in the virtual reference coordinate system (*virtual-alignment*) and then compares and matches the data to complete the registration (*compare-and-match*). The *virtual-alignment* phase exploits the unique features of data. The horizon is used to align the image to the virtual reference coordinate system, that is, the image is aligned by matching an estimated horizon to a standard basis vector $e_2 = [0, 1, 0]$ (T_H). Similarly, a ground normal vector is used to align the point cloud. The point cloud is firstly aligned by matching an estimated ground normal vector to a standard basis vector $e_3 = [0, 0, 1]$ (T_E). The *compare-and-match* phase compares the two data in the virtual reference coordinate system. It is assumed that the forward axis of the image is $e_1 = [1, 0, 0]$. The area of overlap of the point cloud with respect to the image is estimated, and the point cloud is rotated to match its forward axis to e_1 (T_F). Next, the registration process is completed by estimating the displacement of the image and the point cloud and using it to match the origin of both coordinate systems (T_G).

We experimentally demonstrate the performance of the proposed method. First, we demonstrate that our method can work reliably even in large turbulences through image-based localization experiments using Rellis-3D [12], which is an off-road dataset collected on the UGV platform. Second, we compare the image-based localization performance with existing methods using the KITTI odometry [2] and nuScenes [13] datasets. Finally, we compare the camera-LiDAR extrinsic calibration performance with existing methods using the KITTI raw [14] dataset. The experiments demonstrate that the proposed method outperforms the existing methods in both localization and calibration tasks in terms of both accuracy and robustness.

The key contributions of our paper are:

- We propose a novel image-to-point cloud registration network named EFGH-Net, which provides a versatile solution to the image-to-point cloud registration

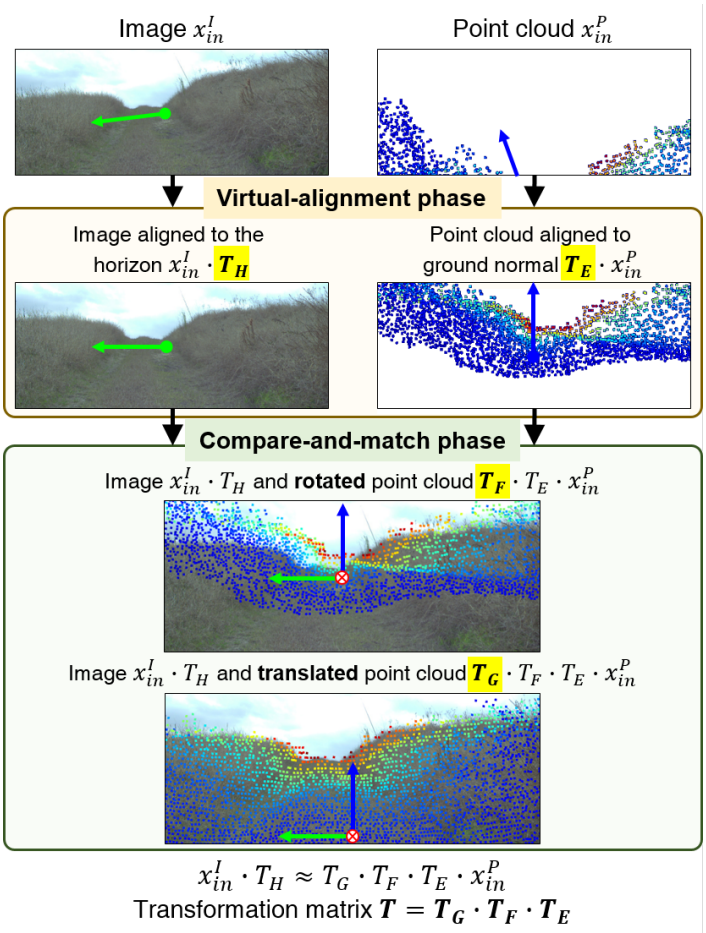


Figure 2.1: **Overview of EFGHNet.** The proposed method aims to determine a transformation that matches an image x_{in}^{I} and a point cloud x_{in}^{P}. In the *virtual-alignment* phase, each of the two input data is aligned to a virtual reference coordinate system. In the *compare-and-match* phase, two aligned data are compared and matched to complete the registration. The final output T is the product of the results from each step.

problem.

- Our method has a two-phase structure using a divide-and-conquer strategy, so it can reliably estimate transformations even with large differences in initial pose, and at the same time have high accuracy.
- We verify the performance of the proposed method in various situations through extensive experiments using four datasets in different platforms and environments.

2.2 Related Work

2.2.1 Image-based Localization

Image-based localization estimates an image sensor's pose under a world coordinate system of a point cloud. One of the early image-based localization approaches utilizes feature descriptors. There are methods of extracting feature descriptors from a given image and point cloud and finding the correlation between the two data in the feature space [5], [6]. Another method extracts descriptors from images and matches descriptor-point pairs at high speed using kd-tree and lookup tables [15]. Another image-based localization approach generates a pseudo image from a point cloud. There are methods to generate synthetic images from a 3D prior map and compare them with images observed by the onboard camera [16], [17], [18]. Conversely, there is an approach to reconstructing a pseudo point cloud from image features. A localization method uses visual odometry to generate 3D points from an image and matches generated 3D points with a 3D prior map [19]. Another localization method using vision-laser matching that generates a sparse keypoint map using vision and IMU data and matches it with a dense LiDAR map [20]. Finally, a learning-based approach has been proposed. CMRNet [9] is an early study on the introduction of CNNs to image-based localization and has proposed a network that learns how to match images to 3D

maps. This method uses an initial pose matrix to generate a depth image from a point cloud and processes the depth and RGB images as a network to estimate a 6-DOF matrix. 2D3D-Matchnet [21] extracts descriptors from image patches and local point clouds through neural networks and estimates the camera's pose using a PnP algorithm, based on the correspondence between the descriptors. DeepI2P [11] solves the image-to-point cloud registration problem as a classification problem. DeepI2P classifies points existing in the image frustum through the neural network and estimates the optimal camera pose by solving the optimization problem for the classified points and image.

2.2.2 Camera-LiDAR Extrinsic Calibration

The camera-LiDAR extrinsic calibration estimates the relative pose between an image sensor and a point cloud sensor mounted on the same platform. In the target-based approach, both sensors measure a calibration target simultaneously, and the algorithm estimates a calibration matrix based on the measured target. Boards with checkerboards [22], [23], and circular [24] [25] markers are often used in this approach. Ordinary boxes are used as targets [26] to reduce the effort required to set up a calibration system. The other approach is a targetless approach that eliminates the need for a target during the calibration process. Instead, this approach uses additional information such as external sensors or time-series data. Methods using inertial measurement units (IMUs) [27], navigational sensors [28], and omnidirectional cameras [29] have been proposed. The targetless approach has come a long way with the advent of deep learning. RegNet [10] is the first attempt to regress rigid transformation parameters using a convolutional neural network (CNN). CalibNet [30] uses geometric supervision to minimize the photometric and point cloud distance errors in the training of the CNN. RGGNet [31] is an online tolerance-aware extrinsic calibration method that introduces Riemannian geometry and the concept of tolerance to solve a specific calibration problem. Another method uses road markers for extrinsic calibra-

tion between nonoverlapping cameras and LiDAR [32]. SOIC [33] is a semantic-based automatic camera–LiDAR extrinsic calibration method. In this method, the calibration process is initialized using newly introduced semantic centroids (SCs) obtained from semantic segmentation results. CRLF [34] entails the extraction of straight-line features from image–point cloud pairs, estimation of the calibration matrix by solving the perspective-3-lines (P3L) problem, and devising a cost function to refine the estimated calibration results. A method has been proposed for the automatic calibration of multiple LiDARs and cameras [35]. In this method, multiple LiDAR sensors are first calibrated through bundle adjustment, and adaptive voxelization is then used to accelerate the process of extrinsic calibration between multiple LiDARs and cameras.

2.3 Methods

We define an image-to-point cloud registration problem as follows. The inputs are an image $x_{in}^{I} \in \mathbb{R}^{H \times W \times 3}$ and a point cloud $x_{in}^{P} \in \mathbb{R}^{N \times 3}$, and the output is the transformation matrix T. The overall registration process is divided into four steps, handled in subnetworks E3, Forward-axis, Gather, and Horizon. Each subnetwork estimates a transformation matrix, that is, T_E, T_F, T_G, and T_H. The final result of the registration is the product of the matrices $T = T_G \cdot T_F \cdot T_E$, indicating the transformation from the point cloud sensor to the image sensor.

In the following sections, p refers to the prediction, y refers to the ground truth, superscript I denotes the image, and P denotes the point cloud. The reference frame consists of e_1, e_2, and e_3.

2.3.1 E3 Network

The first step involves aligning the input point cloud to the virtual reference coordinate system (Fig. 2.2). The E3 network estimates the ground normal vector p_{gn} from the point cloud x_{in}^{P}. The output of the network T_E is a transformation matrix that rotates

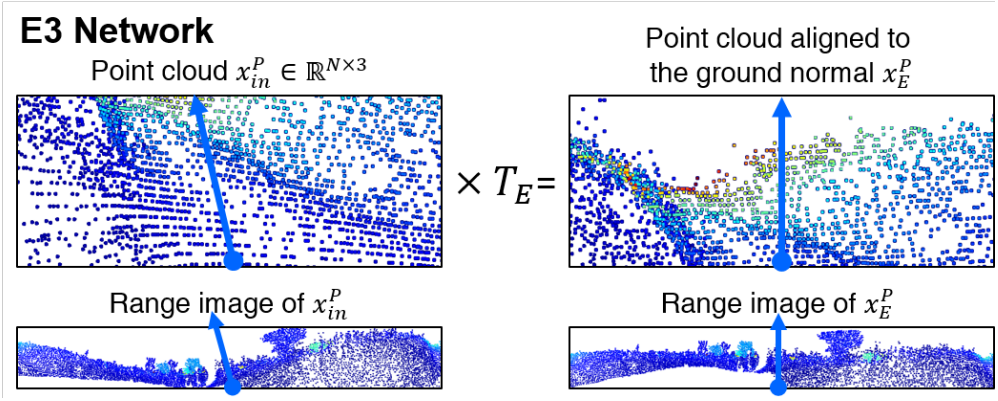


Figure 2.2: Process of E3 network.

p_{gn} to fit the standard basis vector $e_3 = [0, 0, 1]$.

We implement the E3 network using the DownBCL block, which processes the point cloud and extracts features. A bilateral convolutional layer (BCL) is used to process 3D information contained in a point cloud [36], [1], [37]. DownBCL, proposed in [37], expands the receptive field of BCL. Therefore, by stacking this layer, the DownBCL block can learn point cloud information distributed over a large area. The extracted features of DownBCL are fed into the rotation head.

We design a rotation head to estimate the rotation vector p_r using the spherical regression framework proposed in [38], which is a general solution to the regression of the n-sphere problem. The rotation head makes two predictions: the absolute value $p_{r_{abs}} \in \mathbb{R}^{1\times r_{dim}}$ and sign value $p_{r_{sgn}} \in \mathbb{R}^{1\times 2^{r_{dim}}}$ of p_r. The sign prediction is $2^{r_{dim}}$-dim, because the + and - signs of p_r are encoded as a one-hot vector. In E3 network, the rotation head estimates the ground normal vector $p_{gn} \in \mathbb{R}^{1\times 3}$, where $r_{dim} = 3$.

The loss function $\mathcal{L}_r$ of the rotation head consists of two parts: the absolute and sign parts.

$$\mathcal{L}_r = -\frac{y_{r_{abs}} \cdot p_{r_{abs}}}{||y_{r_{abs}}||_2 \cdot ||p_{r_{abs}}||_2} + CE(y_{r_{sgn}}, p_{r_{sgn}}) \tag{2.1}$$

where the loss for the absolute part is the cosine proximity loss and that for the sign

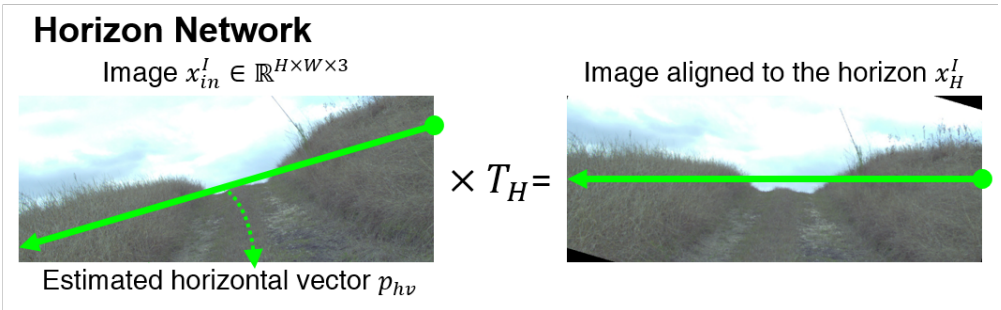


Figure 2.3: Process of Horizon network.

part is the cross-entropy loss (denoted as CE).

2.3.2 Horizon Network

In the second step, the input image is aligned with the virtual reference coordinate system (Fig. 2.3). The horizon network estimates the horizontal vector p_{hv} from image x_{in}^{I}. A horizontal vector is defined as a vector that is parallel to the horizon in the image. When the right-end pixel of the horizon is $[u_r, v_r]$ and the left-end is $[u_l, v_l]$, the horizon is defined as $y_{hv} = [u_l - u_r, v_l - v_r, 0]$. The output of the network T_H rotates p_{hv} to fit $e_2 = [0, 1, 0]$.

The Horizon network consists of a VGG [39] network used to extract features from x_{in}^{I} and the rotation head to estimate the horizontal vector $p_{hv} \in \mathbb{R}^{1 \times 3}$, where $r_{dim} = 3$ with the third element fixed at 0.

2.3.3 Forward-axis Network

In the third step, the Forward-axis network matches the forward axis of the point cloud with that of the image. It is assumed that the vector entering the image plane is the forward axis of the image coordinate $e_1 = [1, 0, 0]$, and that the estimation target is the forward axis of the point cloud p_{fwd}. Output matrix T_F rotates p_{fwd} to fit e_1 to match the two forward axes.

The two inputs of the Forward-axis network are an image aligned to the horizon ${x^I}_H = x^I_{in} \cdot T_H$, and a range map $x^R \in \mathbb{R}^{H \times \lambda W \times 4}$ converted from the point cloud ${x^P}_E = T_E \cdot x^P_{in}$, as shown in

$$
\begin{aligned}
r &= \sqrt{x^2 + y^2 + z^2} \\
h &= \frac{VF^P_U - \sin^{-1}(z/r)}{VF^P_U - VF^P_L} \cdot H \\
w &= \frac{\pi - \tan^{-1}(y/x)}{2\pi} \cdot \lambda W \\
x^R(h, w) &= [x, y, z, r]
\end{aligned}
\tag{2.2}
$$

where VF^P_U is the upper bound of the vertical field-of-view (FoV) of the point cloud, VF^P_L is the lower bound, and H and W are the height and width of image x^I_{in}, respectively. In addition, λ is the ratio of the horizontal FoV of the point cloud to that of the image, as shown in

$$
\lambda = (HF^P_U - HF^P_L)/(HF^I_U - HF^I_L) \tag{2.3}
$$

where HF^P_U and HF^P_L is the upper bound and lower bound of the horizontal FoV of the point cloud, and HF^I_U and HF^I_L is the upper and lower bound of the image. We assume that $\lambda \geq 1$ because the horizontal FoV of the point cloud ($= 2\pi$) is wider than the image.

We design a method for predicting the correlation score map between two inputs (Fig. 2.4), inspired by the cross-view localization method [40]. The image ${x^I}_H$ and range map x^R are processed using independent CNNs. We use the VGG [39] network and add a simple convolution layer before and after the network to reshape the features. Each CNN generates a feature map from an image $f^I \in \mathbb{R}^{H' \times W' \times C}$ and a range map $f^R \in \mathbb{R}^{H' \times \lambda W' \times C}$. The network computes the correlation score map $p_{cs} \in \mathbb{R}^{\lambda W'}$ between the feature maps along the image width, as shown in

$$
p_{cs}(i) = \sum_{c=1}^{C} \sum_{h=1}^{H'} \sum_{w=1}^{W'} f^R(h, (i+w)\%\lambda W', c) \cdot f^I(h, w, c) \tag{2.4}
$$

where $\%$ is the modulo operation.

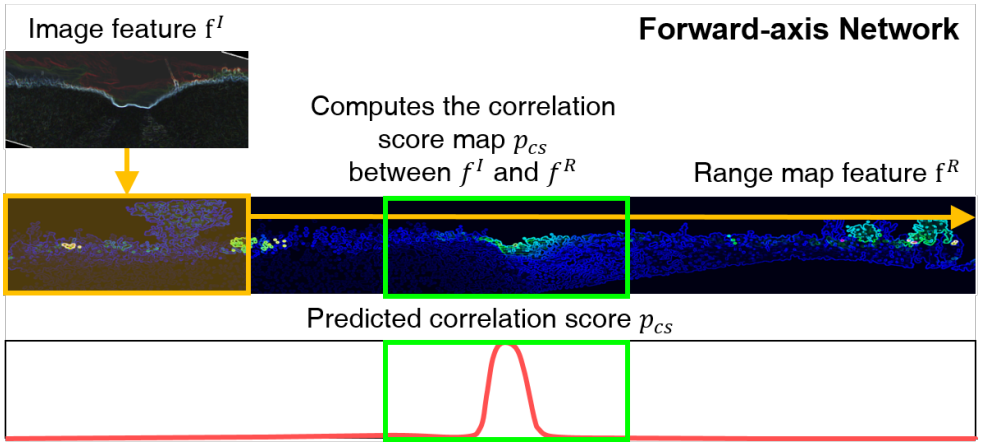


Figure 2.4: **Process of Forward-axis network.** The feature map of the image f^I and the feature map of the range map f^R are used to compute the correlation score p_{cs}. The area with the highest correlation score (green box) indicates the area of the image that overlaps the range map.

The forward axis of the point cloud can be calculated using the w-index of the highest correlation score as

$$p_{rad} = \pi - \frac{2\pi \cdot_w (p_{cs})}{\lambda W'} \tag{2.5}$$

$$p_{fwd} = [\cos(-p_{rad}), \sin(-p_{rad}), 0] \tag{2.6}$$

The ground truth $y_{cs} \in \mathbb{R}^{\lambda W'}$ is set to one for the pixel corresponding to the ground truth y_{rad} value and n pixels around it, and all other pixels are set to zero. Binary cross-entropy loss and hard-negative mining are used for the training.

2.3.4 Gather Network

The fourth and final step is to gather the previous results and estimate the displacement of the image and point cloud (Fig. 2.5). The two inputs of the Gather network are an image ${x^I}_H$ and depth image $x^D \in \mathbb{R}^{H \times W \times 4}$ converted from the point cloud ${x^P}_{EF} =$

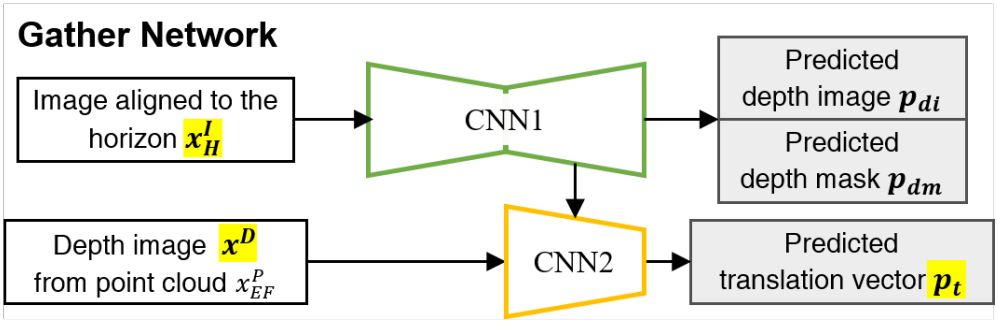


Figure 2.5: Process of Gather network.

$T_F \cdot T_E \cdot {x^P}_{in}$:

$$[u, v, w]^T = K_{init} \cdot [x, y, z, 1]^T$$
$$x^D(u/w, v/w) = [x, y, z, w] \tag{2.7}$$

where K_{init} denotes the initial calibration matrix. The estimated translation vector p_t is used to generate an output transformation matrix T_G, which moves the origin of the point cloud coordinates $(0, 0, 0)$ to p_t to match the origin of the image coordinate system.

The image ${x^I}_H$ is passed through an encoder-decoder network (CNN1) to predict the pseudo-depth image $p_{di} \in \mathbb{R}^{H \times W \times 1}$ and depth mask $p_{dm} \in \mathbb{R}^{H \times W \times 1}$. A pseudo-depth image is an image with depth value for each pixel in the image ${x^I}_H$. The ground truth is a projection of the ground truth point cloud onto the image plane. A depth mask is a binary image for each pixel in the image ${x^I}_H$ that has a value of 1 if the point cloud is projected, and 0 otherwise. The feature map of the decoder and depth image x^D are input into ResNet [41] (CNN2), and the translation vector $p_t \in \mathbb{R}^{1 \times 3}$ is estimated. The loss function for the pseudo-depth image p_{di} is the mean squared error, for the depth mask p_{dm} is the cross-entropy error, and for p_t is the $L1$-loss.

2.4 Experiments

2.4.1 Implementation Details

The weights of the network are initialized to a normal distribution with a mean of 0.0 and a standard deviation of 10^{-3}. The Adam optimizer is used for network optimization with an initial learning rate of 10^{-4}. The learning rate is multiplied by 0.7 for every 50000 iterations.

2.4.2 Test Set Configurations

The test set configurations are listed in Tab. 2.1. The image-based localization test sets are denoted as *Test1*. A pair of image and point cloud is captured at different times, within $10m$. We tested single image and single point cloud pairs (*Test1-A,B,E,F*) and single image and accumulative point cloud pairs (*Test1-C,D,G,H*). The accumulative point cloud is generated by accumulating data from five frames before and after the selected point cloud. The camera-LiDAR extrinsic calibration test sets are denoted as *Test2*. A pair of image and point cloud is captured at the same time step.

We also add noise to the image and point clouds to simulate the tremors of the platform (image-based localization, *Test1*) and a miscalibrated state (camera-LiDAR extrinsic calibration, *Test2*). The noise range is $\pm\alpha^\circ$ for the image rotation. The input image is generated by rotating the image and cropping it using the original image size. The noise range for the point cloud is $\pm\beta^\circ$ for rotation and $\pm\gamma m$ for translation. The input point cloud is generated by rotating the point cloud in the roll, pitch, and yaw directions and translating it.

2.4.3 Image-based Localization

Experiment Setup

Dataset We perform extensive experiments on three large-scale outdoor datasets: the Rellis-3D [12], KITTI odometry [2], and nuScenes [13] datasets. The Rellis-3D dataset

Table 2.1: Test set configurations

Name	**Image**	**Point Cloud**	α	β	γ
Test1-A	single	single	$15°$	$15°$	$2m$
Test1-B	single	single	$30°$	$30°$	$2m$
Test1-C	single	accumulative	$15°$	$15°$	$2m$
Test1-D	single	accumulative	$30°$	$30°$	$2m$
Test1-E	single	single	$0°$	$0°$	$0m$
Test1-F	single	single	$10°$	$10°$	$0m$
Test1-G	single	accumulative	$0°$	$0°$	$0m$
Test1-H	single	accumulative	$10°$	$10°$	$0m$
Test2-A	single	single	$10°$	$10°$	$0.5m$
Test2-B	single	single	$15°$	$15°$	$2m$
Test2-C	single	single	$20°$	$20°$	$1m$
Test2-D	single	single	$30°$	$30°$	$2m$

is an off-road dataset collected using the Clearpath Robotics Warthog UGV platform. It provides 1920×1200-sized RGB images from a Basler acA1920-50gc camera, point clouds from a 64-channel Ouster OS1 LiDAR, calibration, and GPS information. We followed the official data split for training, validation, and testing, which consists of 7800 training samples, 2413 validation samples, and 3343 test samples. The KITTI odometry dataset provides 11 sequences (00 to 10) of images, point clouds, calibration data, and ground truth vehicle poses. The image data are 1392×512-sized RGB images collected using Point Grey Flea 2, and the point cloud data were collected using Velodyne HDL-64E. Sequences 00 to 06 are used for training (15237 samples), 07 to 08 for validation (5172 samples), and 09 to 10 for testing. The nuScenes [13] dataset provides a total of 1000 scenes (850 for trainval split, 150 for test split) of images with a pixel resolution of 1600×900 and point clouds collected by a 32-channel LiDAR. In the trainval set, the first 700 scenes are used for training (13979 samples), and 150

scenes are used for validation (3003 samples) with 20 frame stride.

Evaluation Metrics We used the average relative rotation error (RRE) and average relative translational error (RTE) to evaluate the accuracy.

Result

Experiment on Rellis-3D dataset As a proof of concept, we perform experiments on a fixed input with different models, including E3, Forward-axis, Gather, and Horizon networks. In Tab. 2.2, we list the performance using different models, where EH represents the combination of the E3 network and the Horizon network, EFH represents the EH plus the Forward-axis network, and EFGH represents the final model, including the Gather network. According to the results, the average RRE and RTE gradually decreased with the addition of each subnetwork. The results show that EFH achieves much lower RRE than EH, and EFGH has a certain amount of improvement in RTE through translational alignment than EFH. These results prove the validity of our primary idea that, through the divide-and-conquer strategy to solve the image-based localization problem, even a large range of errors can be reliably controlled.

Comparison to baseline methods To demonstrate the effectiveness of our method, we compare the performance of EFGHNet with two baseline methods. BANet+ICP is a classical ICP matching algorithm using input point clouds and pseudo-points generated by the depth estimation network BANet [42] from input images. DeepI2P [11] is a state-of-the-art learning-based image-to-point cloud registration method. We trained the network using published code on the KITTI odometry and nuScenes datasets.

The results are presented in Tab. 2.3. Our method is robust for diverse conditions, including different ranges of noise and types of point clouds (single or accumulative). In the Test1-E set on KITTI odometry, the difference in RRE between BANet+ICP and the proposed method is $0.3°$. However, in the Test1-F set, this difference increases to $5.85°$. This trend is also observed in comparison with DeepI2P and nuScenes dataset.

Table 2.2: Image-based localization performance on Rellis-3D dataset

Rellis-3D								
Testset	**Test1-A**		**Test1-B**		**Test1-C**		**Test1-D**	
Models	RRE (°)	RTE (m)	RRE (°)	RTE (m)	RRE (°)	RTE (m)	RRE (°)	RTE (m)
Initial State	20.02 ± 20.33	4.67 ± 2.82	32.89 ± 19.42	4.65 ± 2.78	20.02 ± 20.33	4.67 ± 2.82	32.89 ± 19.42	4.65 ± 2.78
EH	15.49 ± 22.20	4.67 ± 2.82	21.21 ± 21.31	4.65 ± 2.78	16.48 ± 22.58	4.67 ± 2.82	20.89 ± 21.44	4.65 ± 2.78
EFH	14.92 ± 21.40	4.67 ± 2.82	16.98 ± 23.48	4.65 ± 2.78	14.85 ± 21.42	4.67 ± 2.82	17.07 ± 23.25	4.65 ± 2.78
EFGH	$\mathbf{14.92 \pm 21.40}$	$\mathbf{4.47 \pm 2.92}$	$\mathbf{16.98 \pm 23.48}$	$\mathbf{4.46 \pm 2.87}$	$\mathbf{14.85 \pm 21.42}$	$\mathbf{4.43 \pm 2.92}$	$\mathbf{17.07 \pm 23.25}$	$\mathbf{4.42 \pm 2.86}$

Table 2.3: Image-based localization performance comparison on KITTI odometry and nuScenes datasets

KITTI Odometry								
Testset	**Test1-E**		**Test1-F**		**Test1-G**		**Test1-H**	
Methods	RRE (°)	RTE (m)	RRE (°)	RTE (m)	RRE (°)	RTE (m)	RRE (°)	RTE (m)
BANet [42] + ICP	5.17 ± 6.75	5.05 ± 3.24	10.14 ± 7.31	5.02 ± 3.21	5.03 ± 6.74	4.97 ± 3.16	10.02 ± 7.32	4.96 ± 3.12
DeepI2P [11]	8.49 ± 8.02	6.19 ± 3.14	13.28 ± 8.41	6.20 ± 3.16	8.33 ± 8.08	6.19 ± 3.16	13.25 ± 8.31	6.31 ± 3.17
EFGHNet	$\mathbf{4.87 \pm 8.02}$	$\mathbf{4.79 \pm 2.93}$	$\mathbf{4.29 \pm 9.31}$	$\mathbf{4.87 \pm 2.90}$	$\mathbf{4.89 \pm 8.76}$	$\mathbf{4.79 \pm 2.93}$	$\mathbf{4.28 \pm 8.55}$	$\mathbf{4.87 \pm 2.90}$
nuScenes								
Testset	**Test1-E**		**Test1-F**		**Test1-G**		**Test1-H**	
Methods	RRE (°)	RTE (m)	RRE (°)	RTE (m)	RRE (°)	RTE (m)	RRE (°)	RTE (m)
DeepI2P [11]	9.21 ± 7.82	6.51 ± 3.15	13.45 ± 6.97	6.52 ± 3.17	6.94 ± 7.04	6.26 ± 3.09	12.49 ± 6.34	6.37 ± 3.12
EFGHNet	$\mathbf{5.77 \pm 5.40}$	$\mathbf{4.46 \pm 3.05}$	$\mathbf{4.73 \pm 5.41}$	$\mathbf{4.49 \pm 3.05}$	$\mathbf{5.26 \pm 5.38}$	$\mathbf{4.47 \pm 3.05}$	$\mathbf{4.09 \pm 5.37}$	$\mathbf{4.50 \pm 3.05}$

That is, neither the classical method (BANet+ICP) nor the learning-based method (DeepI2P) could estimate a reasonable transformation. In contrast, our method showed similar errors for all four test sets. These results quantitatively demonstrate that our method can effectively control noise and reliably operate image-to-point cloud registration compared with other methods. The performance of the proposed method can also be qualitatively verified from Fig. 2.6. EFGHNet is the only method that correctly aligns the image and point clouds.

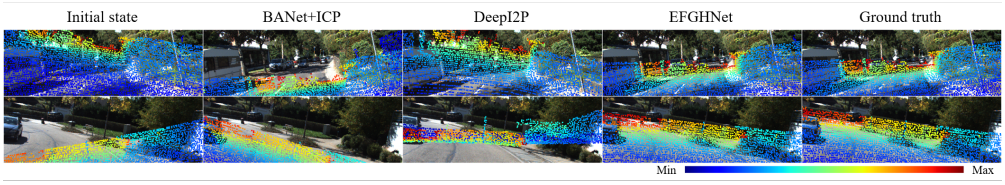


Figure 2.6: Qualitative results of image-based localization experiments on KITTI odometry dataset.

2.4.4 Camera-LiDAR Extrinsic Calibration

Experiment Setup

Dataset The KITTI raw [14] dataset provides image, point cloud, and calibration data. The image data are 1392×512-sized RGB images collected using Point Grey Flea 2, and the point cloud data were collected using Velodyne HDL-64E. The data split for training and testing follows that of a previous study [10], where data collected on 09/26/2011, excluding sequences 05 and 70, are used for training (15,306 image-point cloud pair samples), and sequences 05 and 70 are used for validation (574 samples). For testing, sequence 28 from 09/30/2011 (5177 samples) is used, which is a dataset collected on a different day.

Evaluation Metrics We used the quaternion distance (QD) to evaluate the rotation accuracy and the mean absolute error (MAE) to evaluate the translation accuracy, as shown in

$$QD = 2atan(\sqrt{q_i^2 + q_j^2 + q_k^2}/q_r)) \text{ and } q = q_y q_p^{-1} \tag{2.8}$$

$$MAE = mean(||t_y - t_p||) \tag{2.9}$$

where the subscript p represents the prediction and y represents the ground truth.

Table 2.4: Camera-LiDAR extrinsic calibration performance comparison for KITTI raw dataset

KITTI Raw								
Testset	**Test2-A**		**Test2-B**		**Test2-C**		**Test2-D**	
Methods	QD(°)	MAE(m)	QD(°)	MAE(m)	QD(°)	MAE(m)	QD(°)	MAE(m)
CalibNet [30]	5.10 ± 2.86	0.25 ± 0.09	9.52 ± 4.40	1.00 ± 0.34	12.79 ± 5.85	0.50 ± 0.18	23.12 ± 8.91	1.00 ± 0.34
RGGNet [31]	1.79 ± 1.04	$\mathbf{0.12 \pm 0.06}$	7.79 ± 6.08	0.90 ± 0.35	5.64 ± 5.34	0.41 ± 0.17	17.73 ± 12.27	0.96 ± 0.34
LCCNet [43]	11.62 ± 6.50	0.61 ± 0.29	13.31 ± 7.51	0.79 ± 0.40	14.58 ± 7.79	0.59 ± 0.27	21.29 ± 14.09	0.85 ± 0.40
EFGHNet	$\mathbf{1.54 \pm 1.79}$	0.17 ± 0.07	$\mathbf{1.89 \pm 2.66}$	$\mathbf{0.48 \pm 0.22}$	$\mathbf{1.65 \pm 2.36}$	$\mathbf{0.27 \pm 0.12}$	$\mathbf{2.12 \pm 3.57}$	$\mathbf{0.48 \pm 0.22}$

Result

To validate our method, we compared its performance with those of the latest methods, *CalibNet* [30], *RGGNet* [31], and *LCCNet* [43]. The quantitative results are summarized in Tab. 2.4. The QD difference between our method and *RGGNet*, which has the lowest error among the baseline methods, is $0.25°$ in Test2-A, but this difference increases to $15.61°$ in the Test2-D set. Based on the results, we can verify that our method is capable of stable operation under various experimental conditions. These results also prove that our method has comparable or superior performance to existing methods in camera-LiDAR extrinsic calibration, which requires high accuracy.

2.5 Conclusions

We present an accurate and robust image-to-point cloud registration method that is viable in urban and off-road environments. Our main contribution is the introduction of divide-and-conquer strategies to image-to-point cloud registration. We build a two-phase method that aligns the two input data in the virtual reference coordinate system (*virtual-alignment*) and then compares and matches the data to complete the registration (*compare-and-match*). We performed extensive experiments on four datasets (Rellis-3D [12], KITTI odometry [2], nuScenes [13], and KITTI raw [14]). The experiments demonstrate that the proposed method outperforms existing methods in terms

of accuracy and robustness.

Our experiments focused on existing datasets with simulated noise. Future work should explore the performance of the proposed method in real off-road robots. Another important area of future research is to optimize the proposed network to be able to run at high speed on mobile robots. EFGHNet can become the cornerstone of these researches.

Chapter 3

ABCD: Attentive Bilateral Convolutional Network for Robust Depth Completion

3.1 Introduction

LiDAR is a laser-based sensor that measures the distance to a target using a laser beam. LiDAR has been adopted in a wide range of research areas such as robotics, autonomous driving, and drones. The data obtained from a LiDAR sensor are essentially sparse and, therefore, are often referred to as *sparse* depth information. The low resolution of the data degrades the performance of tasks such as 3D object detection, semantic segmentation, and object tracking. This limitation of LiDAR has led to considerable research into what has been called the depth completion task. A depth completion task typically uses LiDAR and camera data together to improve the resolution of *sparse* depth information to produce *dense* depth information.

Color camera images have been key elements in depth completion research over the years. This is because color images can provide precise boundary information of objects, whereas LiDAR data lack such knowledge. Therefore, most depth completion networks generate dense depth information based on features extracted from color images. However, because cameras are vulnerable and subject to lighting and weather

conditions, a strong dependency on the camera can render depth completion algorithms unstable. To be useful in real-world applications, depth completion algorithms must provide robust and accurate depth information.

To construct efficient depth completion networks, many existing depth completion approaches adopt convolutional neural networks (CNNs), which have been widely used for image processing [44],[45],[46],[47],[48]. However, standard CNNs do not consider the spatial adjacency of LiDAR data during feature extraction. Therefore, to improve the performance of depth completion networks, it is necessary to develop CNNs optimized for LiDAR data.

A 3D CNN for depth completion must process a wide range of point sets simultaneously while preserving detailed point information. To achieve this goal, we adopt the bilateral convolutional layer (BCL), which has been previously proposed for image filtering [36], 3D segmentation [1], and scene flow estimation [37] of LiDAR point cloud data. The BCL can process the entire 3D point cloud of a scene, without using pre-processing steps such as sub-sampling. It is due to this property of the BCL that enables the network to preserve the characteristics of each point and provide a suitable architecture for depth completion. However, the BCL handles the points uniformly, and thus, it lacks the ability to differentiate points for objects at diverse distances for depth completion.

To overcome this problem and retrieve accurate depth information, we designed an attentive bilateral convolutional layer (ABCL) in this study. We introduced an attention mechanism [49] that refines the learned 3D information, highlights specific channels and areas that contain key information for the task, and suppresses clutter using a self-generated attention map. With the ABCL as a building block, we propose an end-to-end depth completion network called the attentive bilateral convolutional network for depth completion (ABCD), which shows robust performance even under harsh environmental conditions. The proposed depth completion network overcomes the limitations of camera-dependent depth completion methods, which are vulnera-

ble to environmental changes, and, hence, it can be applied to a variety of real-world conditions. We use the ABCL to build a point encoder that extracts 3D spatial features from a point cloud. The 3D information encoded via the point encoder is passed to the image plane through a feature projection process, which enables our network to use multimodal inputs to efficiently integrate information from different domains. We succeeded in building a remarkably robust network by adding a point encoder to the image-centric encoder-decoder structure, which has been a commonly employed structure in existing depth-completion networks. An overview of the proposed method is shown in Fig. 3.1.

We evaluated our approach using two datasets: KITTI [3] and VirtualKITTI2 [4]. The KITTI depth completion dataset contains $93k$ training samples under bright and clear daytime conditions. VirtualKITTI2 is a synthetic dataset that contains data generated under various lighting and weather conditions. The average performance of the proposed method on these two datasets was comparable to that of the state-of-the-art methods. Furthermore, under harsh weather and lighting conditions (shaded, corrupted, morning, fog, overcast, rain, and sunset), the proposed method outperformed the others by significant margins.

The contributions of this study are as follows:

- We designed and implemented a novel 3D convolutional layer called ABCL, which operates directly on point cloud data and exploits a wide range of 3D spaces.
- We developed a robust depth completion network named ABCD that, using an effective 3D processing method, functions well under unpredictable real-world driving conditions.
- Through comparative experiments with other methods using the KITTI and VirtualKITTI2 datasets, we demonstrated the outstanding performance of the proposed method in diverse driving environmental conditions.

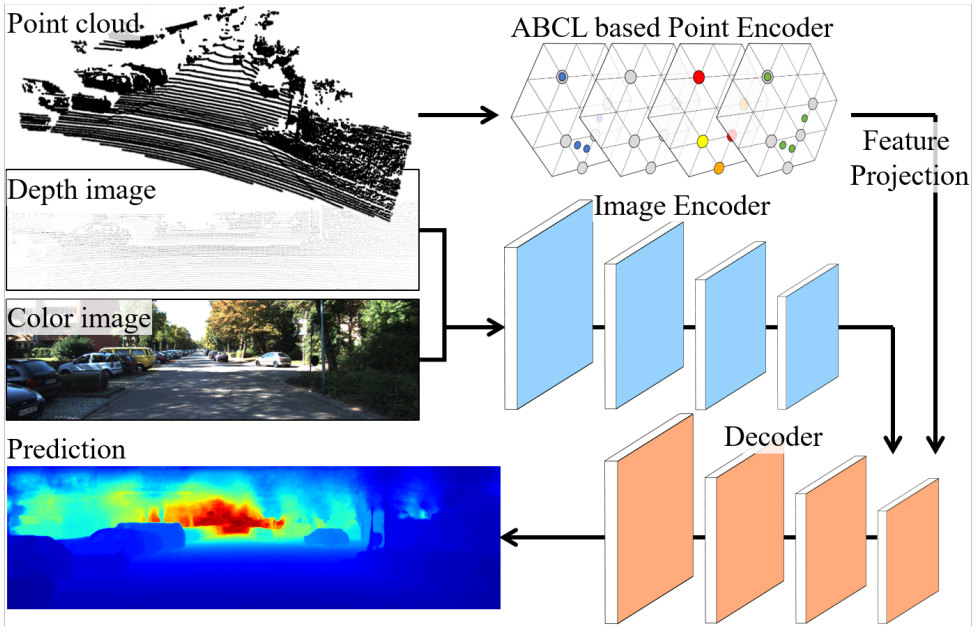


Figure 3.1: **Overview of the proposed method.** The proposed method uses sparse depth information measured by LiDAR sensor in the point cloud and depth image formats. While previous methods utilized only two inputs, depth image and color image, the proposed method leverages the 3D feature information extracted directly from 3D point cloud through the ABCL as the third input.

3.2 Related Work

3.2.1 Depth Completion

The emergence of deep learning methods has led to considerable progress in depth completion research. One study applied standard CNNs, which have been widely used for image processing, to the depth completion task [44]. Furthermore, surface normal information is generated and combined with color images to predict the dense depth image [48], [50]. Studies have also attempted to improve performance by introducing new concepts to depth completion: affinity [51], uncertainty [52], and confidence [53]. A depth completion study proposed an approach for learning the adaptive convolution kernel size and number of propagation iterations [54]. A recent study [55] improved this method [54] and applied it to a depth refinement module. The two-stage depth completion framework, which improved the limitations of the existing one-stage framework, achieved high performance on both indoor and outdoor datasets [56]. Guide image filtering, which is used to predict the weights of guided kernels and extract features for completing sparse depth images, has been used in depth completion studies [47]. Furthermore, a study that introduced repetitive designs to image-guided filtering was proposed [57].

3.2.2 3D Deep Learning

Three-dimensional (3D) point cloud data are used in a wide range of fields, including robotics, autonomous vehicles, and 3D mapping; therefore, the utilization of these data for deep learning has also been widely studied. Certain approaches divide 3D space into regular grids, where the points are mapped to a uniform voxel grid, and these grid-format data are used as inputs to CNNs [58], [59], [60]. However, processing 3D data in the voxel format incurs problems such as the loss of original information and an inefficient use of memory owing to the sparsity of the data. A novel method, named PointNet [61], was proposed to directly handle the point cloud data using a multi-layer

perceptron (MLP), rather than converting it to a grid format. Additionally, the use of a hierarchical summary of the features from the point cloud was proposed in a subsequent study [62]. Tangent convolution [63] processes 3D data using a set of tangent images, which is the result of projecting the local surface geometry on the tangent planes around all points. PointCNN [64] introduced convolution to learn a χ-transformation from points. Certain approaches [65], [66], [67] use a continuous convolution kernel at neighboring points to learn 3D features. Therefore, a depth completion method [45] using continuous convolution [66] was proposed. Compared to our approach, continuous convolution [66] is unable to expand the receptive field, which is desired for 3D deep learning for depth completion.

3.3 Methods

3.3.1 Preliminary: Bilateral Convolutional Layer

A BCL uses a high-dimensional Gaussian kernel with learnable weights. The input of the BCL consists of a position vector $p_{in} \in \mathbb{R}^{N \times 3}$ and a value vector $v_{in} \in \mathbb{R}^{N \times d}$ for N points with a feature length of d. The entire BCL operation is performed on a permutohedral lattice and consists of three steps: *splat*, *convolve*, and *slice* [36],[1] (Fig. 3.2).

The *splat* step involves rearranging the input into the lattice. The *convolve* step involves a convolution operation using a learnable Gaussian kernel on the permutohedral lattice. In the *slice* step, the result of the convolution is projected back from the permutohedral lattice to the original point.

The downsampling and upsampling operators of the BCL, named DownBCL and UpBCL, are described in [37]. DownBCL consists of two steps: *splat* and *convolve*, in which a fine lattice is downsampled to a coarser lattice, producing fewer outputs than input points. UpBCL consists of a *convolve* and *slice* step that produces more output than input points by upsampling the features from a coarse lattice to a finer lattice.

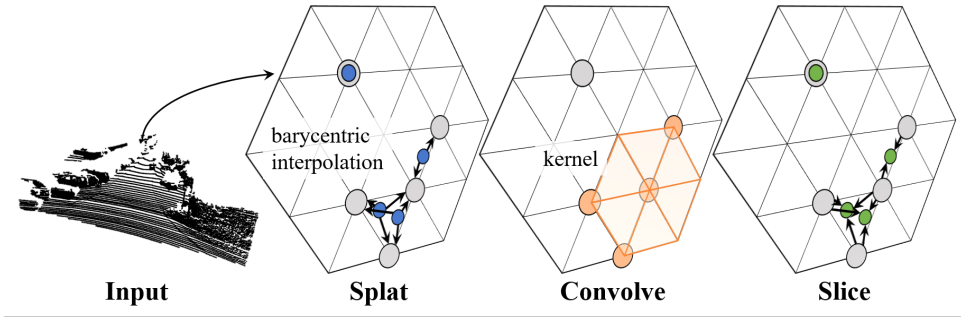


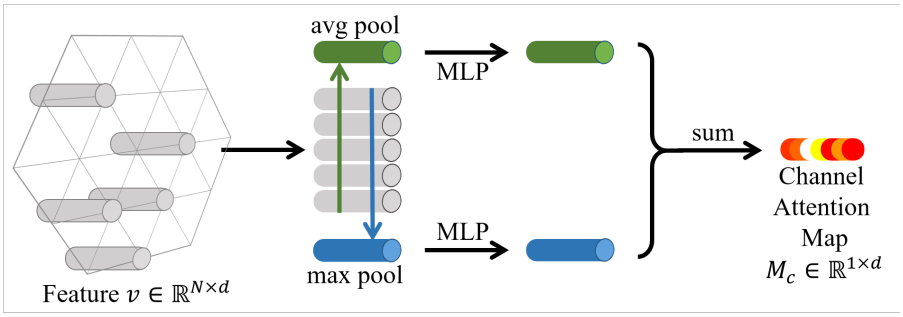
Figure 3.2: Bilateral convolutional layer (BCL) [1].

3.3.2 Attentive Bilateral Convolutional Layer

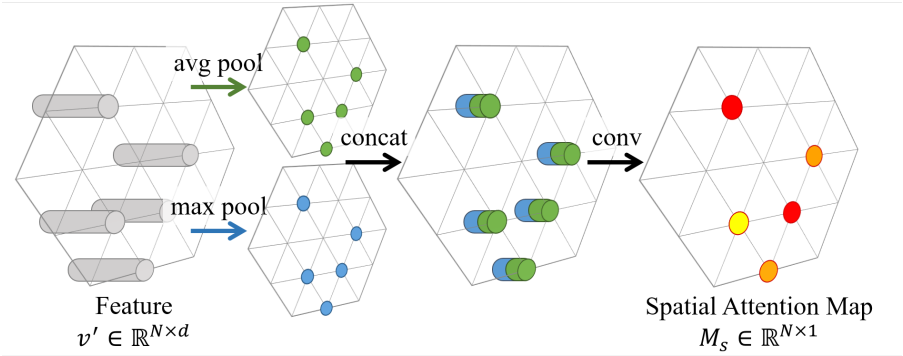
The ABCL consists of four steps: *splat, convolve, attention* (which is a newly proposed step), and *slice*. The *attention* step is an implementation of the attention mechanism CBAM [49] onto a permutohedral lattice, thus refining the results of the convolution. We applied this idea to UpBCL and DownBCL to design an UpABCL consisting of a *convolve-attention-slice* and a DownABCL consisting of a *splat-convolve-attention*.

The input of the attention step is a value vector, $v \in \mathbb{R}^{N \times d}$, which is the result of the *convolve* step. N is the number of occupied lattice points and d is the size of the channel. The channel attention map $M_c \in \mathbb{R}^{1 \times d}$ squeezes the spatial dimension of the value vector in one dimension, and the spatial attention map $M_s \in \mathbb{R}^{N \times 1}$ summarizes the channel dimension of the value vector in one dimension (Fig. 3.3). The output of the attention step v'' is the result of the refinement of the value vector v using these two attention maps.

A channel attention map was used to determine *what* to focus on. This helps the network focus on important channels among the d channels of the value vector. The process of generating the channel attention map involves two pooling layers (average and max), an MLP, and a sigmoid activation function. It has been reported that squeezing the input feature map using a pooling layer is an efficient way to create an attention map. In earlier studies, the average pooling layer effectively aggregated the



(a) Channel attention map



(b) Spatial attention map

Figure 3.3: Two attention maps generated in attention step.

spatial information of the feature map [68], [69]. CBAM [49] has empirically demonstrated that max pooling has important interpretive capabilities for distinctive features to generate fine channel attention, and thus has shown that using average pooling and max pooling together can generate more effective attention maps. Based on this idea, we used a combination of average and max pooling to create an attention map for the channel. The two pooling layers spatially summarize the input value vector to create a channel attention map. The value vector is refined by adding new weights to the channel through the element-wise multiplication of the channel attention map and the value vector. For the value vector, pooling layers aggregate N lattice points and generate a $1 \times d$-sized vector. Using the MLP, a $1 \times d$-sized vector is reduced by the reduction ratio, r, and restored. Eq. 3.1 represents the process of generating a channel attention map. v_i represents the value vector of the i-th lattice point, $w_0 \in \mathbb{R}^{\frac{d}{r} \times d}$ and $w_1 \in \mathbb{R}^{d \times \frac{d}{r}}$ are the parameter matrices of the MLP, and σ represents the sigmoid function.

$$M_c = \sigma(w_1(w_0(\frac{1}{N}\sum_i v_i + MAX_i v_i)^T))^T \tag{3.1}$$

A spatial attention map was used to determine *where* to focus. It assists the network in concentrating on meaningful points among N lattice points. The process for generating the spatial attention map involves average and max pooling layers, a 1×1 convolutional layer, and a sigmoid activation function. It has been reported in one study [70] that applying a pooling layer on the channel dimension helps to highlight important spatial regions. Therefore, we created a spatial attention map by squeezing the channel information of the input value vector using two pooling layers. The d channels of the value vector are aggregated to 1 through two pooling layers, and the results are two $N \times 1$-sized vectors. The outputs from the two pooling layers are concatenated and passed through a 1×1 convolutional layer and sigmoid function. The process for generating a spatial attention map is formulated in Eq. 3.2, where v'_{ij} indicates the j-th element in the value vector of the i-th lattice point.

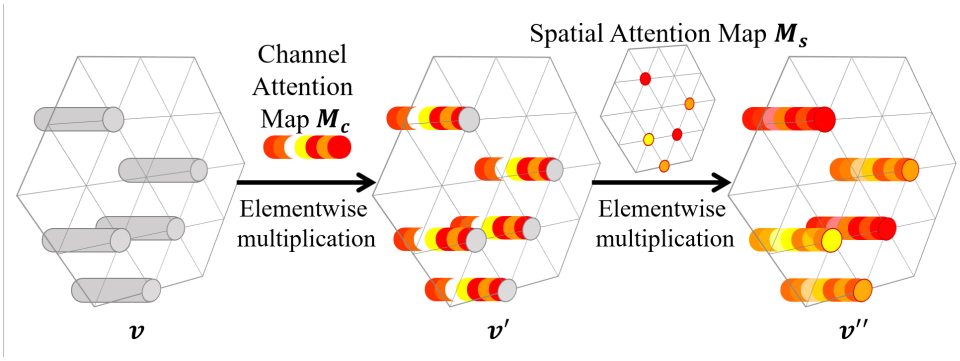


Figure 3.4: Attention step for attentive bilateral convolutional layer (ABCL).

$$M_s = \sum_i \sigma(\frac{1}{d}\sum_j v'_{ij} + MAX_j v'_{ij}) \tag{3.2}$$

The two attention maps weight the input value vector v using element-wise multiplication. First, the element-wise multiplication of M_c and v is performed, and then the result v' is element-wise multiplied by M_s. The entire process of the attention step is shown in Fig. 3.4.

3.3.3 Feature Projection

Feature projection is the process used to transfer the point features into the image domain. The point feature $x_{3d} = (p_{3d}, v_{3d})$ has a position vector $p_{3d} \in \mathbb{R}^{N\times 3}$ and a value vector $v_{3d} \in \mathbb{R}^{N\times d}$. Using a LiDAR-camera calibration matrix, T_{calib}, the 3D coordinates of the position vector p_{3d} are converted into 2D camera coordinates. Then, they are multiplied by the scale factor s to obtain the new position vector p_{2d} in scale s. The entire process is expressed in Eq. 3.3. This process enables the conversion of 3D point features into multi-scale feature maps in the 2D image domain.

$$p_{2d} = (p_{3d} \cdot T_{calib}) \cdot s,\ p_{2d} \in \mathbb{R}^{N\times 2} \tag{3.3}$$

The value vector is assigned to the location corresponding to p_{2d} on the image plane. When two different points in 3D space are on the same pixel in the image, we

assigned the value vector of the point closer to the camera, taking into account the distance between the point and the camera.

3.3.4 Network Overview

The proposed method uses an encoder-decoder structure, with an encoder consisting of an image encoder and a point encoder. The image encoder creates image features, and a point encoder consisting of an ABCL generates point features. After the feature projection process, the decoder takes the resulting features from both encoders and predicts the dense depth image. For training the network, we used the $L2$ loss between the predicted depth image and ground truth. The proposed network is shown in Fig. 3.5, where conv1×1 denotes a convolutional layer with a 1×1 kernel and 1 stride followed by a leaky ReLU ($\alpha = 0.1$), and conv3×3 denotes a convolutional layer with a 3×3 kernel and 1 stride, batch normalization, and a leaky ReLU ($\alpha = 0.2$). The transpose block consists of a transposed convolutional layer with a 3×3 kernel and 2 strides, batch normalization, a leaky ReLU ($\alpha = 0.2$), and conv3×3.

Image Encoder

The image encoder, which learns features using color and depth images, is based on the network described in [44]. And has ResNet [41] as its basic structure. It handles two inputs through additional convolutional layers. After passing through the first convolutional layer, the features from the two input images are concatenated and used as the input to ResNet. The four residual blocks that make up ResNet downsize the input twice.

Point Encoder

The point encoder takes input $x_{in} \in \mathbb{R}^{N \times 3}$ and finds a point feature $x_{out} \in \mathbb{R}^{N \times d}$. Input x_{in} is a point cloud that contains N 3D points represented by xyz coordinates. This information passes through five DownABCLs on a gradually decreasing scale,

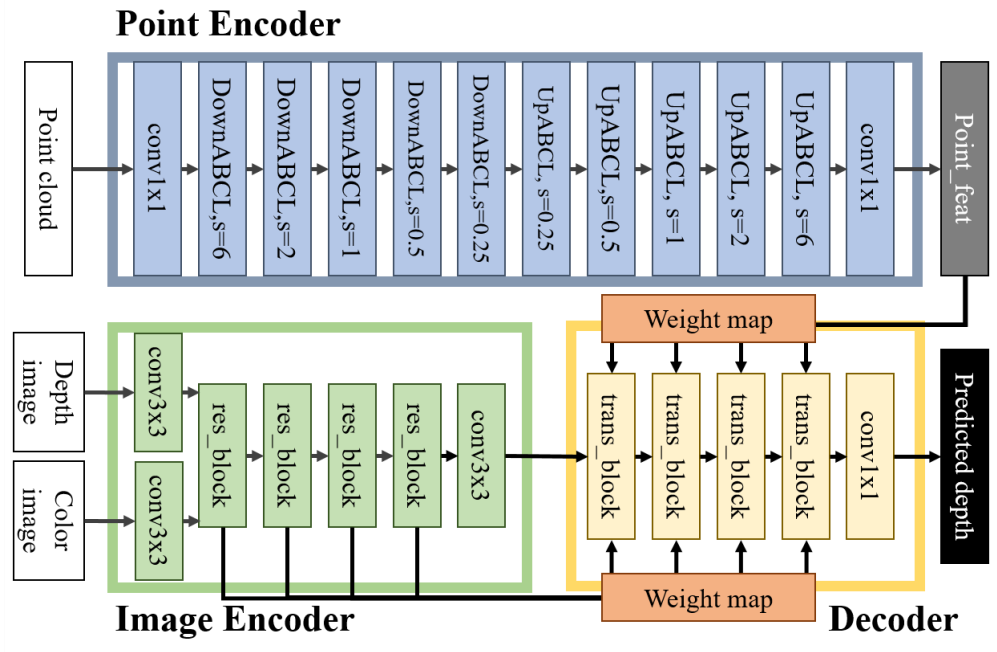


Figure 3.5: ABCD architecture.

and then five UpABCLs on an increasing scale corresponding to the DownABCLs. Each UpABCL takes the previous layer result and DownABCL result of the same scale. By connecting the high-resolution features of a DownABCL to an UpABCL via a skip link, we prevent the early features from disappearing in deep networks. The result of the last UpABCL is sliced into the input point cloud, and a d-dimensional feature vector is created for the input of each of the N points.

Decoder

The decoder uses multiscale image features from the image encoder and point features from the point encoder. The four transpose blocks that make up the decoder consist of a transposed convolutional layer and a convolutional layer. The transposed convolutional layer upsamples the image features twice. The point features are then projected onto each transpose block with the same scale as the image features through a feature

projection process. The image features and point features pass through a 1×1 conv layer and a softmax layer to create a weight map. Next, the weighted feature maps are input as a transpose block. The output of the last transpose block is processed through a convolutional layer, resulting in the prediction of a dense depth image.

3.3.5 Implementation Details

The proposed network was implemented using PyTorch [71]. The weights of the network were initialized to a normal distribution with a mean of 0.0 and standard deviation of 10^{-3}. The Adam optimizer was used for network optimization, with an initial learning rate of 10^{-4}. The learning rate was multiplied by 0.7 every two epochs. We used one NVIDIA GeForce GTX 1080 Ti for training. Our model converges after $2400k$ iterations, which took approximately 3 weeks.

3.4 Experiments

3.4.1 Experimental Setup

The KITTI depth completion dataset [3] was divided into a training set, validation set, and test set. The training set included $92,750$ samples, and the validation set contains $1,000$ samples with color, sparse depth images, and dense depth images. The dense depth images were used as the ground truth and had a depth value of up to approximately 80 m. The test set included $1,000$ samples of sparse depth and color images, excluding the ground truth.

The VirtualKITTI2 [4] dataset is a photo-realistic synthetic dataset, which is a reconstruction of the KITTI video dataset [2] in a virtual environment. This dataset consists of a set of color and dense depth images with a maximum depth of 655.35 m. Because no sparse depth images were provided, we used the raw LiDAR data from the KITTI video dataset to generate a sparse depth image using only the depth values of the pixels where the LiDAR data points are projected onto the image plane. There

Table 3.1: Quantitative comparison with the state-of-the-art methods on the KITTI and VirtualKITTI2 dataset in RMSE (meter)

Method			KITTI				VirtualKITTI2					
Name	Params (M)	Runtime (sec)	Test	Shaded	Corrupted 100/300	Corrupted 300/500	Clone	Fog	Morning	Overcast	Rain	Sunset
GuideNet [47]	62.62	0.120	**0.736**	0.771	0.932	1.077	147.31	124.17	135.83	132.51	131.40	146.82
ACMNet [72]	1.350	0.483	0.744	1.193	1.287	1.421	24.22	24.13	23.89	25.54	26.58	24.44
DeepLiDAR [48]	144.0	0.462	0.758	2.418	1.227	1.961	17.75	19.18	17.99	17.94	17.77	17.70
Van Gansbeke [52]	2.545	0.207	0.772	0.812	1.015	1.193	18.51	22.18	18.60	19.12	19.10	18.35
Sparse-to-Dense [44]	26.11	0.117	0.814	0.883	1.334	1.933	18.69	19.12	18.68	18.44	18.58	18.68
ABCD	32.93	0.248	0.764	**0.755**	**0.865**	**1.052**	**16.27**	**17.17**	**16.55**	**16.37**	**16.29**	**15.94**

are six conditions for the color images: *clone*, *fog*, *morning*, *overcast*, *rain* and *sunset*, each of which contains $4,244$ samples. The *clone* is a synthetic reconstruction of the real world, and the others are variants that differ in lighting and weather conditions for the same scene.

We used the root mean squared error (RMSE) between the prediction and ground truth as a measure to evaluate the accuracy of depth completion. The measurement of runtime is executed on one NVIDIA GeForce GTX 1080 Ti with the Intel Core i7-7700K CPU @ 4.20GHz.

3.4.2 Evaluation on the KITTI Dataset

We compared the proposed method with other state-of-the-art depth completion methods using the KITTI depth completion dataset, as shown in Table 3.1. First, we submitted our depth completion result for the test set to the KITTI evaluation server to compare its performance with other methods on the KITTI leaderboard. Next, to compare the robustness of the proposed method, we conducted two different experiments by intentionally damaging the input camera image of the KITTI validation set. In the first experiment, we reduced the RGB value to half to assume a situation in which the camera image was shaded. In the second experiment, we blocked a part of the input camera image using a randomly generated black box. The height and width of the

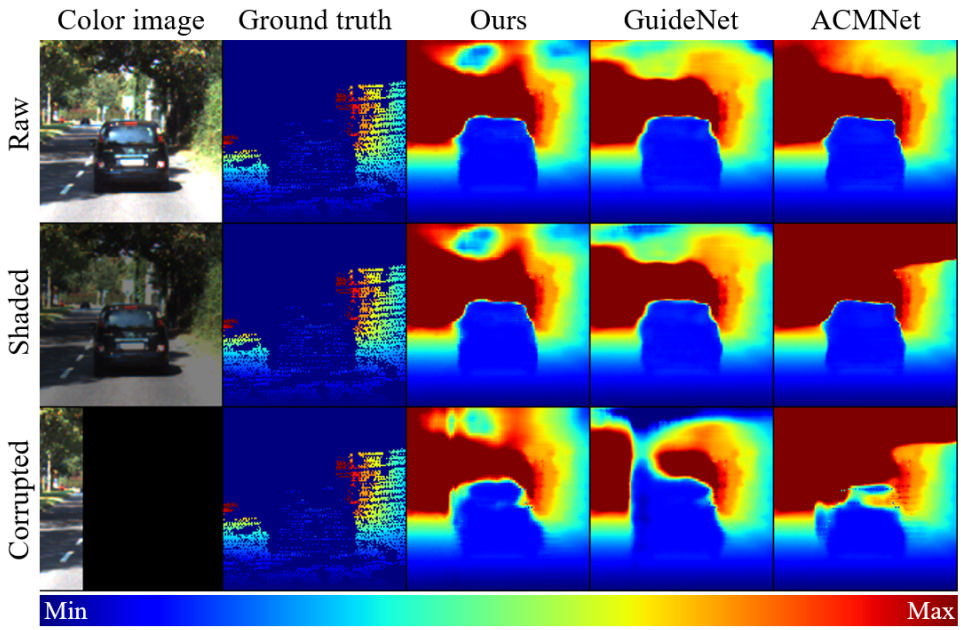


Figure 3.6: **Qualitative comparison with state-of-the-art methods on the KITTI validation set.** The first column shows the input color image and the second column shows the ground truth. The results of our method and other methods are shown in subsequent columns.

black box were determined by uniform sampling in the specified range (represented by minimum/maximum values in the table), and the position of the box was also randomly determined. The experiments used predefined corrupted images for each set of ranges such that all methods could compare their performances under the same input. The prediction results for each condition are presented in Fig. 3.6.

Compared with GuideNet [47], which showed the best performance on the KITTI test set, the RMSE of our method was only 3% higher. However, under various environmental and lighting conditions, the ABCD showed a lower RMSE (from 2% to 68%) compared with other state-of-the-art methods.

3.4.3 Evaluation on the VirtualKITTI2 Dataset

Applications for autonomous vehicles should be robust under a variety of conditions that are likely to be encountered by vehicles. We measured the robustness of the proposed depth completion method under various light and weather conditions using the VirtualKITTI2 dataset. In the training phase, our method and other state-of-the-art methods used only the training set of the KITTI depth completion dataset. Therefore, the model described here is the same as that used in Sec. 3.4.2, without any additional fine-tuning. To ensure a fair comparison, the other methods used in the experiment only applied the trained model published by the author. In the inference phase, we used the VirtualKITTI2 dataset.

The quantitative results for this dataset are listed in Table 3.1. Our ABCD method outperformed the other methods under all conditions. GuideNet [47], ACMNet [72], and DeepLiDAR [48], which achieved lower errors (from 0.7% to 3%) than the proposed method on the KITTI test set, had higher errors (from 8% to 89%) on the VirtualKITTI2 dataset. This indicates that the proposed method does not overfit and has good generalization performance, even though it is trained with limited conditioned data. The qualitative results are presented in Fig. 3.7.

3.4.4 Ablation Study

In this section, we quantitatively evaluate the contributions of each network component. The quantitative results of the KITTI validation set are presented in Tab. 3.2. M0 is a Sparse-to-Dense [44] network, which denotes the framework using only RGB and sparse depth images as inputs. Notably, the BCL-based point encoder improved the performance in RMSE by approximately 10.5% (M0 and M1), which demonstrates that the direct use of point information in depth completion studies is crucial. In addition, the channel attention module provided a 1.2% (M1 and M2) improvement through channel-side emphasis and suppression. Furthermore, the spatial attention module further improved the performance by approximately 2.0% (M2 and M4).

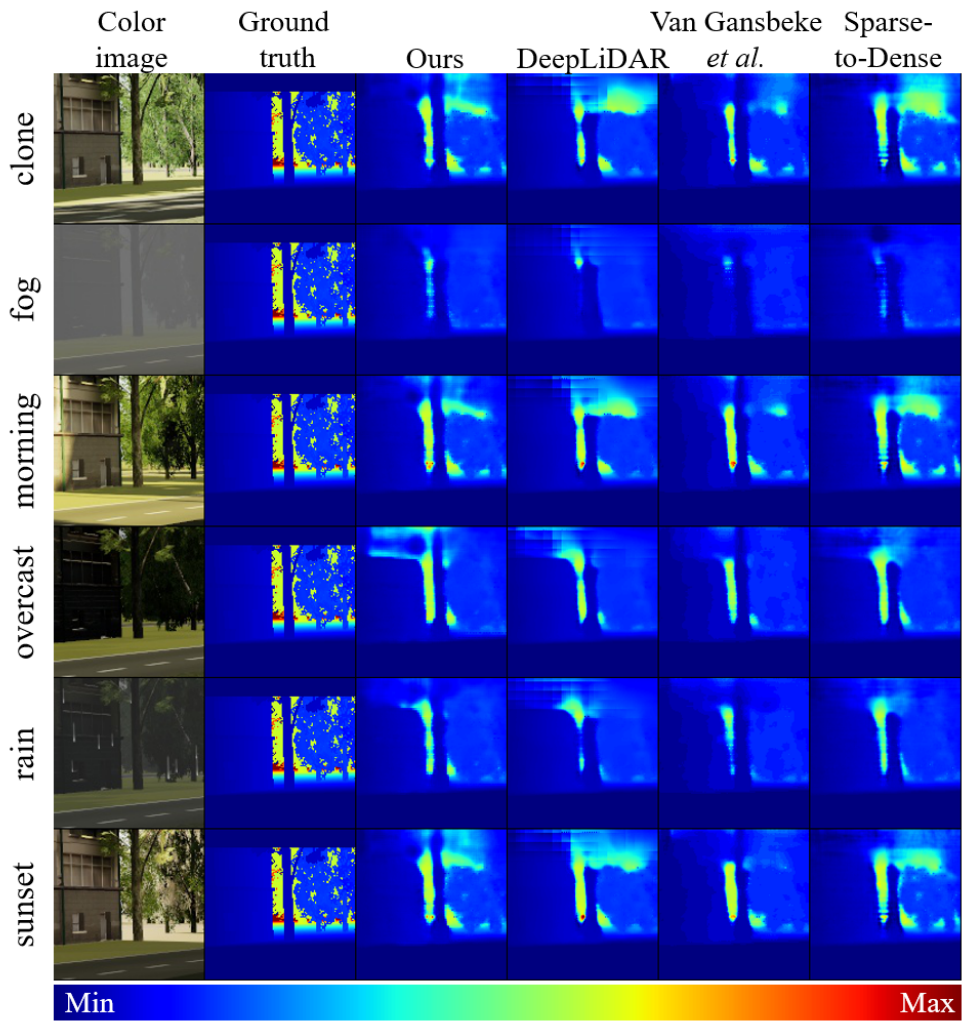


Figure 3.7: **Qualitative comparison with the state-of-the-art methods on the VirtualKITTI2 dataset.** The first column shows the input color image and the second column shows the ground truth. The results of our method and other methods are shown in subsequent columns.

Table 3.2: Quantitative comparison between variants of our model on the KITTI validation set in RMSE (meter)

Name	Params (M)	RGB& Depth	BCL	Attention (Channel)	Attention (Spatial)	RMSE
M0 [44]	26.107	✓				0.8582
M1	32.8925	✓	✓			0.7677
M2	32.9322	✓	✓	✓		0.7583
M3	32.8927	✓	✓		✓	0.7543
M4	32.9324	✓	✓	✓	✓	0.7434

Notably, both the channel and spatial attention modules only increased the number of parameters by only 0.1%.

3.5 Discussion

In research on depth completion, the performance of an algorithm is evaluated based on its accuracy on the KITTI dataset. However, in real-world driving conditions, robustness against changes in the external environment is equally important. We conducted experiments to evaluate the robustness of various methods, where the proposed ABCD outperformed the other algorithms. We analyzed the factors of ABCD's high performance in terms of two aspects.

3.5.1 ABCL-based Point Encoder

The first factor is the ABCL-based point encoder. We analyzed the effect of the point encoder on the robustness through an experiment. In Fig. 3.8, we compare the prediction results of the two networks without using camera images. Unlike the result of M0 [44], the proposed M4 can accurately predict the contours of cyclists. This

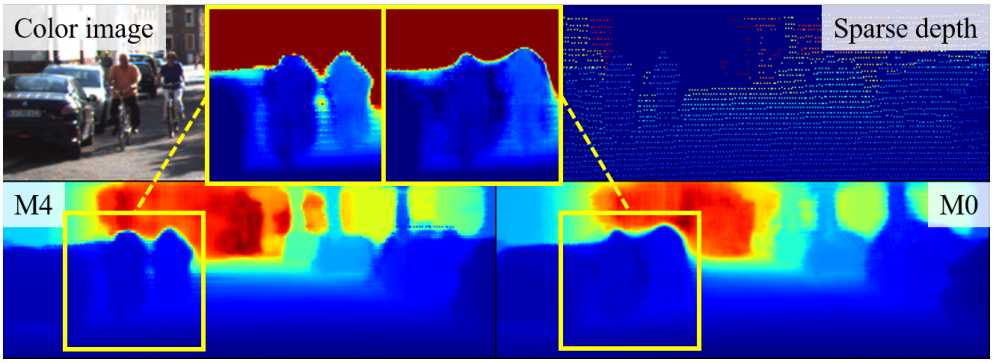


Figure 3.8: **Depth completion result comparison in terms of the point encoder aspect when there is no camera image input.** M4 is a network with an ABCL-based point encoder added to the M0.

indicates that the ABCL-based point encoder is the main factor in making stable predictions even when images are corrupted.

3.5.2 Weight Map

The second factor is the weight map, which determines the proportion of features used in the decoder. A comparison of the two weight maps, shown in Fig. 3.9, shows that the weight map puts more weight on the point feature when no camera data are used. This adaptation minimizes the accuracy degradation of depth completion. However, when image information can improve the prediction of the depth map, the weight map increases the ratio of the image features. In particular, the ratio of image features is high at the boundary of the object, where there are few measured points.

3.6 Conclusions

In this study, we proposed a depth completion network, namely ABCD, that is robust to changes in weather and light conditions that are typically encountered in real-world

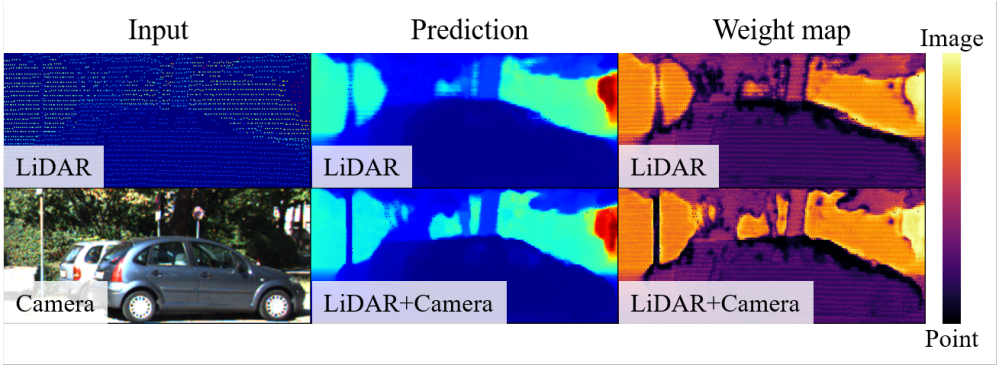


Figure 3.9: **Depth completion results and weight maps of ABCD with and without a camera image.** The first column shows the two input data of the ABCD. The second column shows the prediction result and the third column shows the weight map. In the weight map, yellow indicates a high ratio of the image encoder feature and black indicates a high ratio of the point encoder feature.

driving situations. Furthermore, we introduced ABCL, which can efficiently process point clouds distributed over a large area and refine important features using an attention mechanism. The experimental results show that the proposed method outperforms other state-of-the-art methods qualitatively and quantitatively under harsh environmental conditions.

Chapter 4

Traversability Estimation Based on Footprint Supervision in an Off-road Environment

4.1 Introduction

Traversability estimation is a task that estimates the area a robot can drive based on data collected through the onboard sensor of the robot. This task plays a key role in helping the robot in maneuvering obstacles and drive safely. In particular, in an off-road environment with scattered obstacles and no paved road, estimating the traversable space is essential to prevent the robot from crashing or overturning.

Existing studies consider traversability to be explicit and approach the task by estimating a predefined traversable space. An earlier study introduced the use of superpixels, extracted appearance features from superpixels, and classified the traversability of superpixels based on the extracted features [73]. Inspired by the promising results of convolutional neural networks (CNNs) in computer vision, recent studies adopt CNNs for traversability estimation. GONet [74] proposed a deep learning network that can classify traversable and nontraversable images. GONet synthesizes traversable images using a generative adversarial network (GAN) and classifies the traversability of observed images by comparing the synthesized and observed images. Another study

leveraged CNNs to estimate various traversable routes from observed images [75]. In this study, the observed image was divided into k bins, and the traversability score was estimated for each bin using the encoder-decoder structured network.

However, this approach cannot be a viable solution to the problem of finding robot-centric traversability in off-road conditions. The biggest challenge in estimating off-road traversability is that traversability cannot be explicitly defined. Off-road traversability is implicitly determined by three factors: surface slope, semantic information, and robot platform. For example, if the slope of the surface exceeds the maximum climb slope of the robot, the area becomes nontraversable. However, if the semantics of an area can be penetrated, such as a bush, that area is traversable. Finally, even in the same area, traversability differs depending on the robot platform. All-terrain vehicles (ATVs) can travel over bushes and puddles, whereas small unmanned ground vehicles (UGVs) cannot.

We propose the idea of expert driving, in which a human manually controls the robot. Because the data collected through expert driving represents the driving style of the robot, this information can be used to access the robot's implicit traversability. Specifically, we aim to use the footprints of the robot, which is the route the robot traversed during expert driving. The footprint is a subset of the total traversable space; therefore, with footprint-based supervised learning, we can learn characteristics shared in traversable spaces. Simultaneously, an RGB image and a surface normal image are used to consider semantic and slope information. To effectively integrate the information obtained from the two images, a dynamic filter layer is used. In contrast to a standard convolutional layer, which uses the same kernel for convolution throughout the entire image, the dynamic filter layer generates spatially variant kernels based on the input image and uses them for convolution operations.

In this study, a novel neural network is designed to address the problem of off-road traversability estimation. The proposed network comprises two main elements. First, a footprint-supervision module that predicts the entire traversable space using the foot-

prints of the robot is proposed. Inspired by scribble-supervised semantic segmentation [76], this module predicts the entire traversable space by propagating footprints to neighboring spaces with similar characteristics. Second, the network is developed using a newly proposed dynamic filter layer, named as inter-modality joint-control kernel layer (IJKL), which self-manages the effects of two inputs of the layer, RGB, and surface normal on kernel generation. Thus, the network can effectively integrate the semantic information contained in the RGB image and the surface slope contained in the surface normal image. Using the above two elements, the proposed network effectively considers all three factors that determine traversability: surface slope, semantic information, and robot platform. An overview of the proposed method is shown in Fig. 4.1.

The performance of the method is demonstrated through experiments using RELLIS-3D [12], a public off-road dataset and a custom dataset for mountainous regions. Our method successfully estimated the traversable space, reflecting the driving style of the robot in an off-road environment. The results demonstrate that the proposed method is a feasible traversability estimation solution optimized for off-road environments.

The key contributions of this study are as follows.

- A new concept for traversability estimation is proposed that considers the driving style of a robot through expert driving.
- A footprint-supervision module is developed that can estimate the entire traversable space based on the footprints of the robot.
- A new dynamic filter layer is designed that effectively integrates information from different modalities by self-managing the effects between the two inputs, RGB and surface normal image.
- The performance of our method is demonstrated through experiments using datasets from various environments.

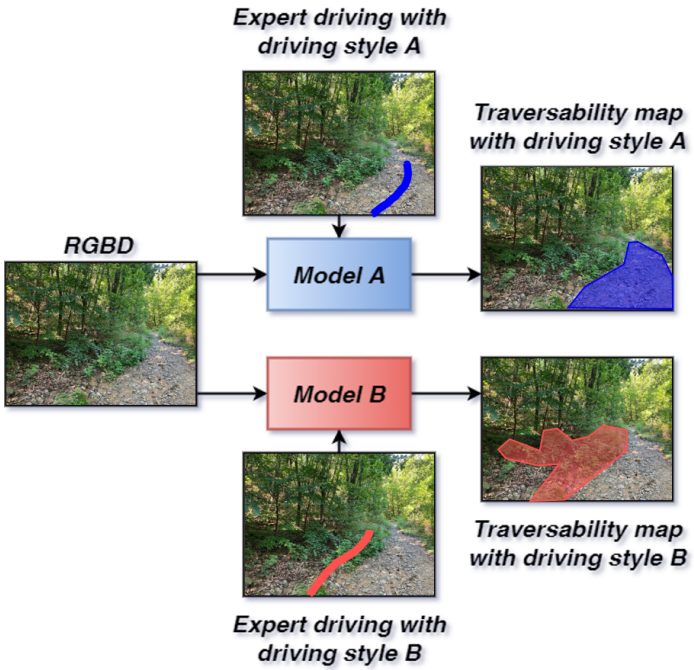


Figure 4.1: We use expert driving to access the driving style of the robot in traversability estimation. In this paper, we propose a method to estimate the entire traversable space by learning the implicit traversability represented by the footprints of robots in expert driving.

4.2 Related Work

4.2.1 Traversability Estimation

Previous studies on traversability estimation defined traversable spaces as obstacle-free spaces. Earlier works dealt with the physical properties of terrain, such as roughness, slope, discontinuity, and hardness and generated a rule-based fuzzy traversability index to classify terrain traversability [77],[78]. Another study on unknown, unstructured outdoor environments included foliage, leaves, and dense vegetation [73]. This study decomposed images into superpixels obtained through oversegmentation, extracted appearance features from superpixels, and classified the traversability of superpixels based on the extracted features.

The introduction of 3D data observed from laser sensors helped to estimate the traversability of an outdoor, unstructured environment. A study on rough terrain [79] used 2D vision and 3D laser sensors, independently built probability maps from the data collected from each sensor, and fused the two maps using Bayes' rule to complete the final traversability map. Another study measured an unstructured environment using 3D range data [80]. In this study, data were collected from human-operated robots, and the collected robot footprints were used as positive labels. The traversability estimation problem was then formulated as PU learning, which used only positive labels and unlabeled data for learning.

Inspired by the promising results of convolutional neural networks (CNNs) in computer vision, recent studies have adopted CNNs for traversability estimation. GONet [74] proposed a deep learning network that can classify traversable and nontraversable images. The generative adversarial network (GAN) constituting GONet synthesizes a traversable image and when the synthesized image matches the observed image, the observed image is classified as traversable; otherwise, it is classified as nontraversable. VUNet [81] improved GONet by introducing a combination of static and dynamic transformation modeling. The proposed network synthesized future images depending

on camera poses and dynamic objects. Similar to GONet, traversability at the current time is classified by comparing the synthesized future image with the currently observed image. Another study estimated various traversable routes from observed images [75]. In this study, the image was divided into k bins, and the traversability score was estimated for each bin using an encoder-decoder structure. The proposed network supports domain adaptation; therefore, the generalization between different domains is possible. In addition, an early stopping tactic was used to ensure the safety of the robot.

4.2.2 Dynamic Filter Layer

The dynamic filter layer generates spatially variant kernels based on the input image and uses them for convolution operations. The dynamic filter layer can improve network representation by overcoming the limitations of the standard convolutional layer, which uses the same kernel for convolution throughout the entire image. A previous study generated vertical and horizontal convolutional kernels from the inputs [82]. These kernels contain information regarding the translation probabilities required to estimate the next image from the input image sequence. Another study proposed an early model for a recent dynamic filter layer [83]. The proposed dynamic filter module is composed of a filter-generating network that generates a kernel from an input and a dynamic filtering layer that applies the generated kernel to the input. A previous study proposed a dynamic filter layer optimized for graphs [65]. In the proposed layer, the generated kernels are conditioned for an input graph and applied to the graph using a convolution-like operation to solve the point cloud classification problem.

Subsequently, various methods have been proposed to improve the early dynamic filter layer. LS-DFN [84] considers the features of adjacent regions in the kernel generation process for an extended receptive field. In addition, in the fusion of features, an attention mechanism is used to assign weights to the feature maps. Several studies have used decomposed filters to solve the large computation problem, which is a limitation

of the early dynamic filter layers. A decoupled dynamic filter (DDF) [85] successfully reduced the amount of computation by decoupling the dynamic filter into spatial and channel filters. C^{2}DFNet [86] uses the DDF to independently process RGB and depth data and proposes scene-aware dynamic filters for inter modality feature interaction. A previous study applied a dynamic filter layer to depth completion [47]. Multimodal features are fused by predicting kernel weights from guidance RGB images and using them for the convolution operation on depth images.

4.3 Methods

4.3.1 Inter-modality Joint-control Kernel Layer

Preliminary: dynamic filter layer

A standard convolutional layer applies the same convolution kernel to each pixel in the image. In contrast, a dynamic filter layer generates different input-conditioned kernels for each pixel in the image. These spatially variant and content-dependent kernels can improve the representation of networks. The convolution operation of a dynamic filter layer can be formulated as

$$F'(i,j) = K_{ij} \otimes F(i,j), \tag{4.1}$$

where $F \in \mathbb{R}^{c \times h \times w}$ is the input feature map, and $F' \in \mathbb{R}^{c' \times h' \times w'}$ is the output feature map. $K_{ij} \in \mathbb{R}^{c' \times c \times k \times k}$ is a generated kernel of size k for pixel (i,j), and $\otimes$ denotes the convolution operation. The dynamic filter layer guarantees improved performance compared to a standard convolutional layer but results in increased memory consumption. For example, a standard convolutional layer requires only one kernel with $c' \times c \times k \times k$ parameters, whereas a dynamic filter layer requires kernels for every pixel in the input feature map. Therefore, a $c' \times c \times k \times k \times h \times w$ parameters are required. To solve this problem, a method for decomposing the kernel was proposed. The decomposed dynamic filter layer proposed in [47] is formulated as follows:

$$
\begin{aligned}
F'(i,j) &= K'_{ij} \otimes F(i,j) \\
F''(i,j) &= K''_{ij} \otimes F'(i,j),
\end{aligned}
\tag{4.2}
$$

where $F \in \mathbb{R}^{c \times h \times w}$ is the input feature map, and $F' \in \mathbb{R}^{c \times h' \times w'}$ is the intermediate feature map obtained by applying the spatially variant kernel $K'_{ij} \in \mathbb{R}^{1 \times 1 \times k \times k}$. $F'' \in \mathbb{R}^{c' \times h' \times w'}$ is the final feature map obtained by applying the content-dependent kernel $K''_{ij} \in \mathbb{R}^{c' \times c \times 1 \times 1}$.

Thus, $c' \times c \times k \times k$ parameters per kernel can be reduced to $c' \times c + k \times k$, and the decomposed dynamic filter layer effectively reduces the computational capacity.

Inter-modality joint-control kernel layer

The dynamic filter layer can improve the performance of the standard convolutional layer; however, there remains a limitation. The limitation occurs when the dynamic filter layer is used for guided image filtering. In guided image filtering, the guidance image, which is the image used to generate the kernel, differs from the convolve image, to which the generated kernel is applied. Because the generated kernel only considers the guidance image, it may fail to extract the optimal features from the convolve image. This can hinder the integration of both images.

A new dynamic filter layer, named as inter-modality joint-control kernel layer (IJKL), is designed, which controls the effect of the guidance image and convolve image on the generation of the kernel. The generated kernel can extract task-optimized features by considering both the guidance image and convolve image. This allows IJKL to effectively integrate information from different modalities. The structure of the IJKL is shown in Fig. 4.2.

Two inputs of IJKL, the guidance image, $x_{guide} \in \mathbb{R}^{c \times h \times w}$, and convolve image, $x_{conv} \in \mathbb{R}^{c \times h \times w}$, are given as input to the confidence generating layer. The confidence generating layer consists of a standard convolutional layer and softmax layer. As an output, a confidence score, $w_{conf} \in \mathbb{R}^{2 \times h \times w}$, is generated. The two inputs, x_{guide} and

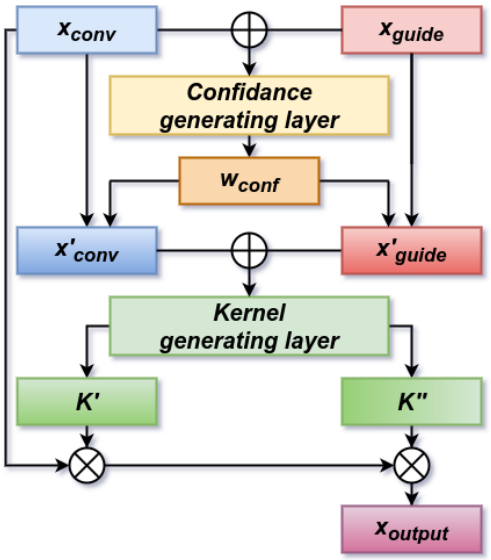


Figure 4.2: **Structure of IJKL.** The guidance image x_{guide} and convolve image x_{conv}, the two inputs of IJKL, are the first inputted to the confidence generating layer, and a confidence score, w_{conf}, is generated. Next, the weighted guidance image x'_{guide} and convolve image x'_{conv} are inputted to the kernel generating layer, and two decomposed kernels K' and K'' are obtained. The two kernels are used sequentially for the convolution operation with x_{conv}.

x_{conv}, are element-wise multiplied by w_{conf} to produce the weighted inputs, x'_{guide} and x'_{conv}.

The weighted guidance, x'_{guide} and convolve, x'_{conv}, images are inputted to the kernel generating layer to generate the kernel. The kernel generating layer generates two types of kernels per pixel. The first is a spatially variant kernel, $K'_{ij} \in \mathbb{R}^{1\times1\times k\times k}$, and the second is a content-dependent kernel, $K''_{ij} \in \mathbb{R}^{c'\times c\times1\times1}$. The two kernels generated here are sequentially used for the convolution operation with x_{conv}, according to Eq. 4.2.

4.3.2 Footprint Supervision

The footprints of the robot from expert driving provide access to implicit traversability. Therefore, a footprint-supervision module is designed that uses the footprints of the robot to predict the entire traversable space by propagating footprints to neighboring spaces with similar characteristics.

In the study of semantic segmentation, scribble supervision is explored as an alternative approach to solve the labor-intensive annotation problem of existing full supervision. Scribble supervision does not use annotations for every pixel; rather, it uses partial annotations known as scribbles. This has applicability in our work, as we need to estimate the entire traversable space using footprints, which are partial annotations. Therefore, a footprint-supervision module based on a scribble-supervised semantic segmentation study [76] is developed. The proposed footprint-supervision module consists of the following elements.

Random walk

Random walk is a widely used method in graph theory [87]. In a graph G having n nodes, an affinity matrix $A \in \mathbb{R}^{n \times n}$ is defined, where A_{ij} represents the affinity between nodes i and j. Using A, the spread of information from a specific node in G can be simulated. If the information distribution of n nodes at time t is defined as $v_t \in \mathbb{R}^{n \times 1}$, then the information distribution at the next time step $t+1$ can be expressed as $v_{t+1} = Av_t$. The diffusion of information over time can be modeled by repeating matrix multiplication.

The equation for applying the random walk operation to the neural network is as follows:

$$F' = \alpha AF + F, \tag{4.3}$$

where F denotes the input feature map for the random walk, and F' denotes the output. Affinity matrix A indicates the probability of a random walk, defined as $A =$

$softmax(F^T F)$. Finally, α represents a learnable parameter that controls the degree of random walk.

Self-supervised loss

Self-supervised loss is widely used in unsupervised learning to produce consistent network outputs. It is defined as the difference between the transformation of the neural representation generated from input x and the neural representation generated from the transformation of x, expressed as follows:

$$L_{ss} = dist(Tr(F(x)), F(Tr(x))), \tag{4.4}$$

where $F(x)$ is the feature map from input x, and Tr is the transformation function. $dist$ is the distance metric between the two neural representations. In this study, Tr is used as the horizontal flip and translation, and $dist$ is used as the mean squared error.

Soft entropy loss

In information theory, a large entropy indicates a large uncertainty and vice versa. Therefore, a prediction with low entropy, which is the desired output of the network, implies a prediction with low uncertainty. This can be achieved by including an entropy term in the loss function.

However, in image segmentation, entropy tends to increase near the boundary where semantic classes are divided. To eliminate the negative effect of entropy increase near the boundary, a method of excluding the boundary entropy from the loss calculation is proposed. This method is called the soft entropy loss [76] expressed as follows:

$$L_{se} = -\frac{1}{|I'|} \sum_{(h,w) \in I'} \sum_{c} p(h,w,c) \cdot log(p(h,w,c)), \tag{4.5}$$

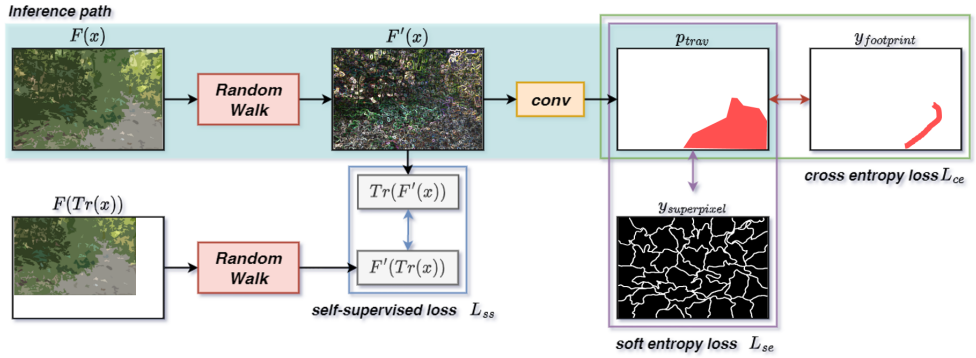


Figure 4.3: **Overview of the footprint-supervision module.** The footprint-supervision module consists of four components: the random walk module, self-supervised loss L_{ss}, soft entropy loss L_{se}, and cross-entropy loss L_{ce}. The light blue box indicates the inference path.

where p denotes the prediction of the network, and $p(h, w, c)$ denotes the value for class c, at location (h, w). I' is a set of pixels that excludes the boundary. Here, the excluded boundary is the boundary of the superpixel generated by SLIC [88].

Footprint-supervision module

The proposed footprint-supervision module is illustrated in Fig. 4.3. First, F is the input of the random walk module, and F' is generated as an output. Inside the random-walk module, α of Eq. 4.3 is used to control the degree of diffusion. F' is fed into the standard convolutional layer and softmax layer to predict traversability map, p_{trav}.

Three loss functions are used in the footprint-supervision module. The first is self-supervised loss, L_{ss}, which is determined as the distance between the transform $Tr(F'(x))$ of the neural representation $F'(x)$ and the neural representation $F'(Tr(x))$ of the transformed input $Tr(x)$. The second is soft entropy loss L_{se}, which is the loss between the superpixel boundary map $y_{superpixel}$ and p_{trav}. The third is cross-entropy

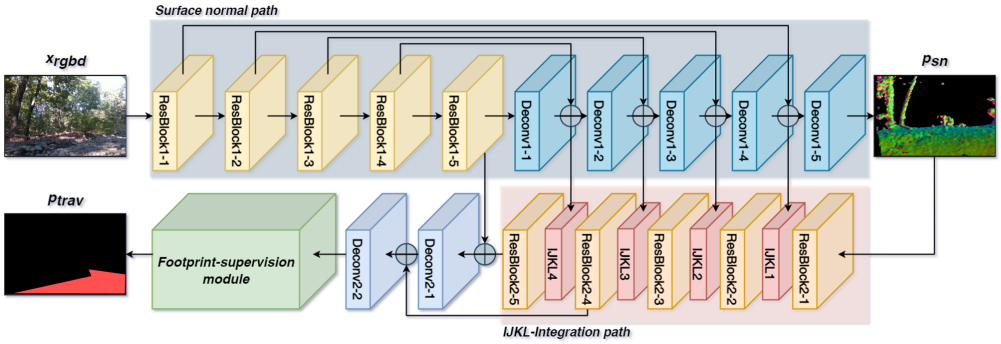


Figure 4.4: **Overview of the proposed network.** The network takes the RGB-D image x_{rgbd} as input, estimates the surface normal image p_{sn} inside the network, and uses IJKL and the footprint-supervision module to predict a traversability map, p_{trav}, as the final result.

loss L_{ce}, which is the cross-entropy between the footprints of the robot $y_{footprint}$ and p_{trav}. Negative labels in $y_{footprint}$ contain both positive and negative samples. Therefore, we set a weight of 1 for losses on positive labels, and 0.1 for losses on negative labels.

4.3.3 Network Architecture

The overall network architecture is illustrated in Fig. 4.4. An RGB-D image is used as input, x_{rgbd}, where D can be a point cloud projected onto the image plane or depth image collected from an RGB-D sensor. The input passes through a series of ResNet [41] blocks (ResBlock1-*) and deconvolution layers (Deconv1-*) to estimate the surface-normal image, p_{sn}. A surface normal image has a three-dimensional surface normal value for each pixel of the image. Here, a skip connection is applied to the input of each deconvolution layer, except for the first layer. The output of the ResNet block is concatenated to the output of the previous deconvolution layer and fed into the next deconvolution layer.

Next, the surface normal image, p_{sn}, is fed into a series of ResNet blocks (ResBlock2-*) and IJKL. Here, IJKL receives two inputs. The first is the output of the previous ResNet block as a guidance image that contains surface normal information, and the second is the output of the deconvolution layer as a convolve image that contains RGB-D information. Through IJKL, the network simultaneously considers the slope contained in the surface normal and semantic information contained in the RGB-D. The feature map passed through the deconvolution layer (Deconv2-*) is input to the footprint-supervision module, and a traversability map, p_{trav}, is generated as the final output.

There are a total of 4 losses used in the network. The self-supervised loss L_{ss}, soft entropy loss L_{se}, and cross-entropy loss L_{ce} are the loss functions used in the footprint-supervision module, and the mean squared error is used as a loss for the surface normal image, L_{sn}.

4.4 Experiments

4.4.1 Dataset

The Rellis-3D [12] dataset is an off-road dataset obtained using the Clearpath Robotics Warthog UGV platform. It provides 1920×1200-sized RGB images from a Basler acA1920-50gc camera, point clouds from Ouster OS1, and GPS data. The official data split was followed for classifying training and validation data, into 7800 and 2413 samples, respectively. In experiments using the Rellis-3D dataset, a depth image was generated by projecting a point cloud onto the image plane using the provided calibration matrix.

In addition, we created a custom dataset. The custom dataset is an off-road dataset collected from mountainous terrain using the Leo Rover UGV platform. It provides 640×480-sized RGB-D images from Intel Real Sense Depth Camera D435i and IMU data. We created three sets to represent different driving styles. The first set represents

driving with priority on dirt roads (driving style A) with 263 samples; the second set represents driving with priority on bushes (driving style B) with 76 samples, and the last set consists of driving data on all terrain (driving style C) with 186 samples. We also created a test set that included various elements that the robot could encounter in the mountains, such as dirt roads, stone roads, fallen leaves, bushes, and trees.

The footprints of the robot were generated using the pose information of the robot from GPS or IMU data. First, the route the robot traversed was generated as a point cloud line in 3D space. Next, the point cloud was projected onto the image plane. The projected point cloud was converted to a binary image, one for the pixel on which the point cloud is projected and zero otherwise. This binary image was used as the footprints of the robot.

4.4.2 Experiments on the Rellis-3D Dataset

A total number of 353 samples were randomly selected from the Rellis-3D test set and manually labeled for quantitative evaluation. Mean intersection-of-union (mIoU) was used as a metric for evaluation. The quantitative results are reported in Tab. 4.1. Here, "Layer" represents the type of basic building blocks of the network: "Base" is a basic dynamic filter layer, and "IJKL" is a proposed dynamic filter layer. Under the same loss function conditions, IJKL always performed better than the Base. In addition, the results show that L_{ce} has the largest effect on network performance, followed by L_{ss}, L_{sn}, and L_{se}.

The spatially variant kernels, K', are depicted in Fig. 4.5 to demonstrate the dependence of generated kernels on the image. The visualization method used in [83] was implemented to visualize the kernels. Here, the network produces kernels with different values for traversable and nontraversable bushes based on semantic information and surface slope.

Table 4.1: Quantitative results on Rellis-3D dataset

Name	Layer	L_{ss}	L_{se}	L_{ce}	L_{sn}	mIoU
Base-A	Base		✓	✓	✓	0.655
Base-B	Base	✓		✓	✓	0.586
Base-C	Base	✓	✓		✓	0.030
Base-D	Base	✓	✓	✓		0.614
Base-E	Base	✓	✓	✓	✓	0.671
IJKL-A	IJKL		✓	✓	✓	0.680
IJKL-B	IJKL	✓		✓	✓	0.705
IJKL-C	IJKL	✓	✓		✓	0.031
IJKL-D	IJKL	✓	✓	✓		0.692
IJKL-E	IJKL	✓	✓	✓	✓	**0.725**

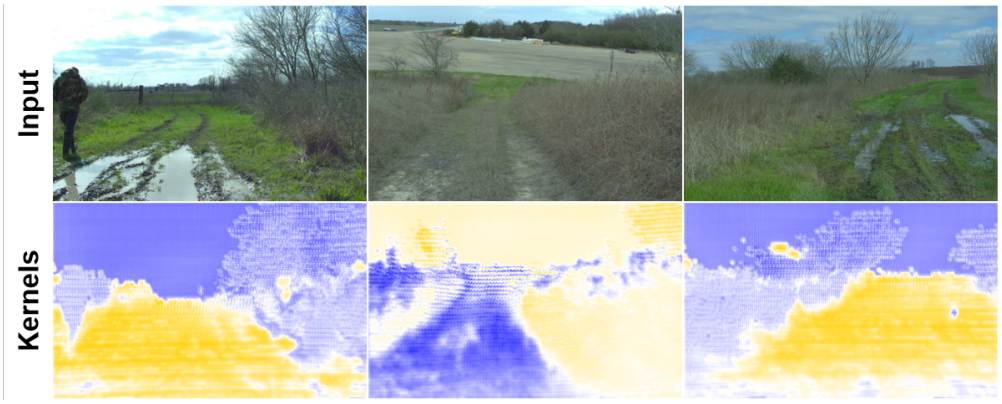


Figure 4.5: **Visualization of the generated kernels.** The similar colors in the image indicate similar kernels.

4.4.3 Experiments on Custom Dataset

Three custom datasets were used to generate three models that reflect different driving styles. The results are presented in Fig. 4.6. The results show that different driving styles predict different traversable spaces for the same scene. Style A predicts dirt roads as traversable spaces, style B predicts bushes as traversable spaces, and style C predicts both dirt and bushes as traversable spaces. This distinct tendency shows that our method estimates traversability by reflecting the driving style implied in expert driving.

4.5 Conclusions

In this study, three factors are considered that determine implicit traversability in an off-road environment: the robot platform, semantic information, and surface slope. Based on this idea, the proposed method estimates the traversability by considering the footprints of the robot obtained from expert driving, the RGB image representing the semantic information, and the surface normal image representing the surface slope. The proposed method is confirmed to be a feasible traversability estimation method optimized for off-road environments through experiments.

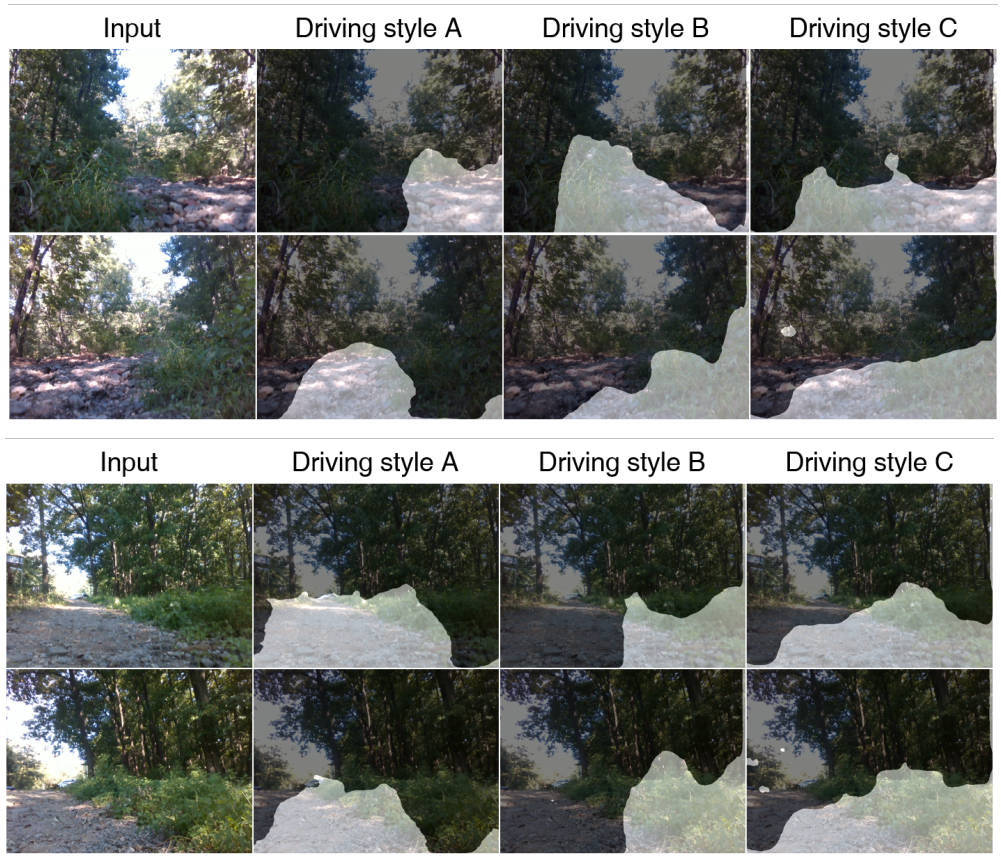


Figure 4.6: **Qualitative results on custom dataset.** The predicted traversable space is highlighted. The results show that different driving styles predict different traversable spaces in the same scene.

Chapter 5

Conclusion

In this dissertation, we propose a framework that combines image and point cloud for autonomous driving in diverse environments. Existing studies optimized under limited conditions showed state-of-the-art performances but often failed to reproduce these performances in the real world. Motivated by this, we aimed to develop a framework that combines image and point cloud that is robust under diverse conditions encountered in the real world.

In Chapter 2, we proposed an image-to-point cloud registration algorithm that estimates a one-to-one match between an image and a point cloud. The algorithm can operate in various terrains, from automobiles on paved roads and UGVs on off-road conditions.

In Chapter 3, we proposed a depth completion algorithm that can generate high-resolution depth information that compensates for the shortcomings of the image and point cloud. This algorithm can operate reliably in a variety of light and weather conditions, from morning to night, in fog and rain, and under camera corruptions.

In Chapter 4, we proposed a traversability estimation algorithm that estimates robot-centered traversability from observed images and point clouds. This algorithm can learn the driving style of the robot platform, from large ATVs to small UGVs, and estimate traversability based on the learned style.

This thesis was completed after exploring and examining the conditions that could be experienced in the real world, which were not considered in previous studies. We believe that this research will serve as a foundation to expand the capabilities of autonomous driving. Based on this, we hope that autonomous driving will contribute to the realization of a better world.

Bibliography

[1] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2530–2539.

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.

[4] Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," 2020.

[5] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1744–1756, 2016.

[6] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3d point clouds," in *European conference on computer vision*. Springer, 2012, pp. 15–29.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[8] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 815–830, 2009.

[9] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard, "Cmrnet: Camera to lidar-map registration," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1283–1289.

[10] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 1803–1810.

[11] J. Li and G. H. Lee, "Deepi2p: Image-to-point cloud registration via deep classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 960–15 969.

[12] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," 2020.

[13] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

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

[15] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele, "Real-time image-based 6-dof localization in large-scale environments," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 1043–1050.

[16] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 176–183.

[17] P. Neubert, S. Schubert, and P. Protzel, "Sampling-based methods for visual navigation in 3d maps by synthesizing depth images," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2492–2498.

[18] G. Pascoe, W. Maddern, and P. Newman, "Direct visual localisation and calibration for road vehicles in changing city environments," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 9–16.

[19] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3d lidar maps," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1926–1931.

[20] A. Gawel, T. Cieslewski, R. Dubé, M. Bosse, R. Siegwart, and J. Nieto, "Structure-based vision-laser matching," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 182–188.

[21] M. Feng, S. Hu, M. H. Ang, and G. H. Lee, "2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4790–4796.

[22] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2301–2306.

[23] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3936–3943.

[24] H. Alismail, L. D. Baker, and B. Browning, "Automatic calibration of a range sensor and camera system," in *2012 Second International Conference on 3D*

Imaging, Modeling, Processing, Visualization & Transmission. IEEE, 2012, pp. 286–292.

[25] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 1–6.

[26] Z. Pusztai and L. Hajder, "Accurate calibration of lidar-camera systems using ordinary boxes," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 394–402.

[27] S. Bileschi, "Fully automatic calibration of lidar and video streams from a vehicle," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2009, pp. 1457–1464.

[28] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor arrays," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4843–4850.

[29] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012.

[30] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1110–1117.

[31] K. Yuan, Z. Guo, and Z. J. Wang, "Rggnet: Tolerance aware lidar-camera online calibration with geometric deep learning and generative model," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6956–6963, 2020.

[32] J. Jeong, Y. Cho, and A. Kim, "The road is enough! extrinsic calibration of non-overlapping stereo camera and lidar using road information," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2831–2838, 2019.

[33] W. Wang, S. Nobuhara, R. Nakamura, and K. Sakurada, "Soic: Semantic online initialization and calibration for lidar and camera," *arXiv preprint arXiv:2003.04260*, 2020.

[34] T. Ma, Z. Liu, G. Yan, and Y. Li, "Crlf: Automatic calibration and refinement based on line feature for lidar and camera in road scenes," *arXiv preprint arXiv:2103.04558*, 2021.

[35] X. Liu, C. Yuan, and F. Zhang, "Fast and accurate extrinsic calibration for multiple lidars and cameras," *arXiv preprint arXiv:2109.06550*, 2021.

[36] V. Jampani, M. Kiefel, and P. V. Gehler, "Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4452–4461.

[37] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, "Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3254–3263.

[38] S. Liao, E. Gavves, and C. G. Snoek, "Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9759–9767.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] Y. Shi, X. Yu, D. Campbell, and H. Li, "Where am i looking at? joint location and orientation estimation by cross-view matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4064–4072.

[41] 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.

[42] S. Aich, J. M. U. Vianney, M. A. Islam, and M. K. B. Liu, "Bidirectional attention network for monocular depth estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 746–11 752.

[43] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "Lccnet: Lidar and camera self-calibration using cost volume network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2894–2901.

[44] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3288–3295.

[45] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2d-3d representations for depth completion," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 023–10 032.

[46] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 103–119.

[47] J. Tang, F.-P. Tian, W. Feng, J. Li, and P. Tan, "Learning guided convolutional network for depth completion," *IEEE Transactions on Image Processing*, vol. 30, pp. 1116–1129, 2020.

[48] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3313–3322.

[49] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[50] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse lidar data with depth-normal constraints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2811–2820.

[51] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. S. Kweon, "Non-local spatial propagation network for depth completion," *arXiv preprint arXiv:2007.10042*, 2020.

[52] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and noisy lidar completion with rgb guidance and uncertainty," in *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE, 2019, pp. 1–6.

[53] A. Eldesokey, M. Felsberg, and F. S. Khan, "Confidence propagation through cnns for guided sparse depth regression," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[54] X. Cheng, P. Wang, C. Guan, and R. Yang, "Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion." in *AAAI*, 2020, pp. 10 615–10 622.

[55] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong, "Penet: Towards precise and efficient image guided depth completion," *arXiv preprint arXiv:2103.00783*, 2021.

[56] L. Liu, X. Song, X. Lyu, J. Diao, M. Wang, Y. Liu, and L. Zhang, "Fcfr-net: Feature fusion based coarse-to-fine residual learning for depth completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 2136–2144.

[57] Z. Yan, K. Wang, X. Li, Z. Zhang, B. Xu, J. Li, and J. Yang, "Rignet: Repetitive image guided network for depth completion," *arXiv preprint arXiv:2107.13802*, 2021.

[58] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[59] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[60] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5648–5656.

[61] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[62] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[63] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3887–3896.

[64] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, pp. 820–830, 2018.

[65] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[66] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2589–2597.

[67] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[68] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[69] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[70] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[71] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[72] S. Zhao, M. Gong, H. Fu, and D. Tao, "Adaptive context-aware multi-modal network for depth completion," *arXiv preprint arXiv:2008.10833*, 2020.

[73] D. Kim, S. M. Oh, and J. M. Rehg, "Traversability classification for ugv navigation: A comparison of patch and superpixel representations," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3166–3173.

[74] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese, "Gonet: A semi-supervised deep learning approach for traversability estimation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3044–3051.

[75] S. Palazzo, D. C. Guastella, L. Cantelli, P. Spadaro, F. Rundo, G. Muscato, D. Giordano, and C. Spampinato, "Domain adaptation for outdoor robot traversability estimation from rgb data with safety-preserving loss," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 014–10 021.

[76] Z. Pan, P. Jiang, Y. Wang, C. Tu, and A. G. Cohn, "Scribble-supervised semantic segmentation by uncertainty reduction on neural representation and self-supervision on neural eigenspace," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7416–7425.

[77] A. Howard and H. Seraji, "Real-time assessment of terrain traversability for autonomous rover navigation," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 1. IEEE, 2000, pp. 58–63.

[78] A. Howard, H. Seraji, and E. Tunstel, "A rule-based fuzzy traversability index for mobile robot navigation," in *Proceedings 2001 ICRA. IEEE International Con-*

ference on Robotics and Automation (Cat. No. 01CH37164), vol. 3. IEEE, 2001, pp. 3067–3071.

[79] J. Sock, J. Kim, J. Min, and K. Kwak, "Probabilistic traversability map generation using 3d-lidar and camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5631–5637.

[80] B. Suger, B. Steder, and W. Burgard, "Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3d-lidar data," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3941–3946.

[81] N. Hirose, A. Sadeghian, F. Xia, R. Martín-Martín, and S. Savarese, "Vunet: Dynamic scene view synthesis for traversability estimation using an rgb camera," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2062–2069, 2019.

[82] B. Klein, L. Wolf, and Y. Afek, "A dynamic convolutional layer for short range weather prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4840–4848.

[83] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," *Advances in neural information processing systems*, vol. 29, 2016.

[84] J. Wu, D. Li, Y. Yang, C. Bajaj, and X. Ji, "Dynamic filtering with large sampling field for convnets," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 185–200.

[85] J. Zhou, V. Jampani, Z. Pi, Q. Liu, and M.-H. Yang, "Decoupled dynamic filter networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6647–6656.

[86] M. Zhang, S. Yao, B. Hu, Y. Piao, and W. Ji, "C^{2} dfnet: Criss-cross dynamic filter network for rgb-d salient object detection," *IEEE Transactions on Multimedia*, 2022.

[87] L. Lovász, "Random walks on graphs," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1-46, p. 4, 1993.

[88] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

초 록

자율 주행은 로봇 연구의 주요 화두 중 하나로 자동차, 실내로봇, 군용로봇, 드론 등 다양한 플랫폼에 사용되어 우리 삶의 일부가 되고 있다. 그 중 주행 지능의 출발점인 “인식”은 센서에서 수집한 데이터를 해석해 주행에 필요한 지식을 생성하는 과정이다. 센서가 수집하는 데이터는 로봇 플랫폼, 지형, 빛이나 날씨 조건에 따라 달라진다. 자율 주행을 위한 인식 기술은 이와 같은 다양한 조건에 따라 변동하는 데이터를 입력받아 안정적으로 정확한 지식을 생성할 수 있어야 한다. 그러나 기존의 연구는 대부분 제한된 조건에서 최고의 성능을 달성하기 위한 벤치마크 경쟁에 초점을 맞추었다. 우리는 이러한 실험실 전용 연구에서 벗어나 실제 세계로 뛰어들어 다양한 조건에서 작동할 수 있는 강력한 알고리즘을 개발하고자 한다.

자율 주행에 널리 사용되는 센서로는 이미지를 수집하는 카메라와 포인트 클라우드를 수집하는 라이다(LiDAR) 가 있다. 이미지는 고해상도의 색상 정보를 포함하고 있으며 조도 변화(햇빛, 날씨 등) 에 민감하다. 포인트 클라우드는 조도 변화에 강하고 3차원 공간의 거리 정보를 제공하지만 낮은 해상도를 가진다. 이렇듯, 이미지와 포인트 클라우드는 보완적인 특성을 가지고 있으므로 우리는 두 데이터의 융합을 통해 각 데이터의 단점을 보완하는 보강된 정보를 생성할 수 있다.

우리는 다양한 환경에서의 자율 주행을 위한 이미지와 포인트 클라우드의 융합 프레임워크를 제안한다. 융합 프레임워크의 첫 번째 단계는 두 데이터의 일대일 일치이다. 이를 이미지-투-포인트 클라우드 레지스트레이션 (Image-to-point cloud registration) 이라고 하며 이 단계의 목표는 2D 이미지를 3D 포인트 클라우드와 정렬하는 것이다. 제안하는 알고리즘은 다양한 지형 조건에서 강건하게 동작하도록 설계되었으며, 포장 도로를 주행하는 자동차와 오프로드를 주행하는 UGV에서 그

성능이 검증되었다.

두 번째 단계는 정렬된 이미지와 포인트 클라우드를 바탕으로 하는 보강된 데이터의 생성이다. 이를 뎁스 컴플리션 (Depth completion) 이라고 하며, 이 단계의 목표는 고해상도의 색상 정보를 가진 이미지와 저해상도의 거리 정보를 가진 포인트 클라우드를 이용하여 고해상도의 거리 정보를 가진 뎁스 이미지를 생성하는 것이다. 제안하는 알고리즘은 아침부터 밤, 안개와 비, 카메라 손상 등 다양한 빛과 날씨 조건에서 안정적으로 작동하도록 설계되었다.

마지막 단계는 관찰된 이미지와 포인트 클라우드를 바탕으로 하는 주행 가능성의 추정이다. 이를 주행 가능성 추정 (Traversability estimation) 이라고 하며, 로봇의 주행 스타일을 학습하여 횡단성을 예측한다. 제안하는 알고리즘은 대형 ATV에서 소형 UGV까지 다양한 로봇 플랫폼의 주행 스타일을 반영한 주행 가능성을 추정할 수 있도록 설계되었다.

본 논문에서, 우리는 다양한 실제 세계의 조건에서 동작할 수 있는 이미지와 포인트 클라우드의 융합 프레임워크를 제시한다. 우리는 제안하는 알고리즘을 다양한 플랫폼 조건 (자동차, 대형 및 소형 UGV), 다양한 지형 조건(포장 도로, 들판, 산), 다양한 빛과 날씨 조건(아침, 해질녘, 비, 안개) 에서의 실험을 통해 검증했다. 제안하는 프레임워크는 다양한 조건에 따라 변동하는 데이터를 입력받고 안정적이고 보강된 지식을 생성하여 후속 응용 프로그램의 안정성을 보장하는 완충 역할을 수행할 수 있다. 따라서 우리는 이 프레임워크가 강력한 자율주행을 위한 초석이 되리라 기대한다.

주요어: 자율 주행, 센서 융합, 딥 러닝, 이미지-투-포인트 클라우드 레지스트레이션, 뎁스 컴플리션, 주행 가능성 추정

학번: 2017-25223