Combining Lidar and Camera via Attention Mechanism for Simultaneous Localization and Mapping in Outdoor Environments

LAM KHA HAN

Japan Advanced Institute of Science and Technology

Master Thesis

Combining Lidar and Camera via Attention Mechanism for Simultaneous Localization and Mapping in Outdoor Environments

LAM KHA HAN

Supervisor : Atsuo YOSHITAKA

Graduate School of Advanced Science and Technology

Japan Advanced Institute of Science and Technology

Information Science

November, 2022

Abstract

Autonomous vehicles are self-control of moving on their own without human intervention. Autonomous vehicles are important in reducing pressure and distraction for drivers to ensure safety when participating in traffic. The area of autonomous vehicles is attractive to researchers. Many problems need to be solved to develop autonomous vehicles, such as navigation, path planning, obstacle avoidance, etc.

Simultaneous Localization and Mapping (SLAM) aims to estimate the position and reconstruct the map of the autonomous vehicle. SLAM is vital in autonomous vehicles and other areas such as drones, recuse systems, robot navigation, etc. SLAM consists of two main modules: front-end and back-end. The front-end module with the other name is odometry, which aims to localize the vehicle's pose and draw a map of the unknown environment by information collected by sensors mounted in this vehicle. Some external factors (e.g., persons, cars) affect odometry estimation. Therefore, the back-end module optimizes the odometry module result. This thesis focus on the odometry module of the SLAM component.

The purpose of odometry is to estimate the autonomous vehicle's path while it is on the fly. Because of work in outdoor environments, autonomous vehicles do not know spatial information or surrounding objects. So, the odometry depends on data collected by sensors. Several sensors can be applied to this problem. Cameras are commonly used since they are inexpensive and convenient. However, RGB cameras without depth information are insufficient for estimating the position, while RGB-D cameras are sensitive to light in outdoor environments. An alternative is to use LiDAR sensors to capture depth information in outdoor environments. This thesis combines visual and depth information from the camera and LiDAR for the odometry of SLAM in outdoor environments.

The main challenge when designing an odometry system is combining information from

sensors. Traditional methods use a handcraft feature extracting and matching pipeline to estimate the vehicle's pose. These methods archive good performance but depend on handcraft feature extraction designs. Deep learning-based approaches have devised breakthroughs in various topics in recent years. The learning-based approach applies in the multi-stage of the odometry framework. The advantage of learning-based methods is independent of the geometric model or handcrafted features. Therefore, this thesis follows the learning-based odometry approach.

The purpose of this work is to combine information from the camera and LiDAR in odometry. The thesis's main contribution is to propose an end-to-end Visual-LiDAR odometry via an attention-driven mechanism. The proposed method extracts the essential regions from RGB images by a self-attention layer instead of feature points in previous works. A region on an image consists of many adjacent points, which is more robust than individual feature points. In previous works, the depth information from the 3D point cloud is used to complement the feature from 2D images. Guided attention is applied to emphasize the interaction between depth and visual information.

The proposed method is evaluated using the KITTI dataset, a data set in traffic with pre-collected image sequences, 3D points cloud from Lidar, and vehicle GPS - considered as ground truth. The estimation result is compared with other learning-based odometry methods.

Keywords: SLAM, **LiDAR**, **Odometry**, **Robot Localization**, **Deep Learning**

Acknowledgments

First, I would like to thank my supervisor, Assoc. Prof. Atsuo Yoshitaka of the School of Information Science, Japan Advanced Institute of Science and Technology. I am grateful to my supervisor for giving me a chance to come to JAIST and work on an exciting project. I wouldn't finish my master's thesis without his support, patience, and motivation for my study.

Second, I thank my friends and all my labmate at Yoshitaka Laboratory for their interest and encouragement during this thesis. Special thanks to Pho-senpai and Tran-senpai for supporting me greatly during my study and life at JAIST.

Finally, I thank my parents for their encouraging and unfailing support throughout my journey.

Ishikawa, Japan

Lam Kha Han

Contents

This thesis was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and VNU - HCMC University of Sciences.

List of Figures

List of Tables

Terms and Abbreviations

SLAM : Simultaneous Localization And Mapping

LiDAR : Light Detection And Ranging

VO : Visual Odometry

LO : LiDAR Odometry

Chapter 1

Introduction

1.1 Background

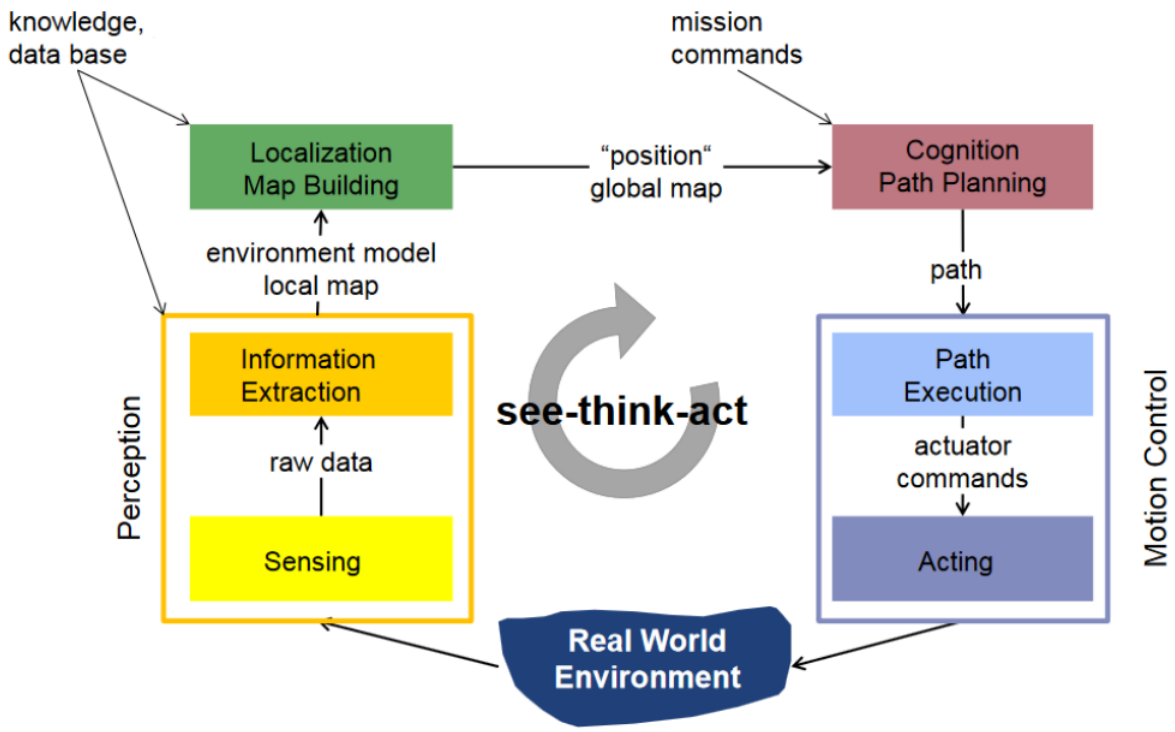

Figure 1.1: The see-think-act diagram[1]

In recent years, autonomous robots have been growing up continuously because of those practical applications in the real world. They can apply in many fields, such as recuse systems, autonomous vehicles, UAVs, etc. For example, autonomous vehicles must work

by sensor capture from sensors without the navigation of humans. In other words, self-driving cars have to be capable of seeing, thinking, and acting.

Figure 1.1 shows a see-think-act diagram [1]. This figure means self-driving cars must first determine their current position in the environment by observations (i.e., seeing). Once they know their position on the global map and the surrounding environment, they can think about path planning (i.e., thinking). After thinking, the vehicle decides its direction to achieve the goal - this is "control." The vehicle will stop, continue, or move to reach the goal (i.e., acting) based on the goal's direction. "Perception" is the first block in the diagram. In this block, raw data from sensors are extracted to be helpful information and understandable by the system. That information is used to build local mapping and localization. Because localization and mapping are estimated simultaneously, this task is called SLAM, which stands for **S**imultaneous **L**ocalization **A**nd **M**apping. This is an essential task in autonomous vehicles.

In recent years, deep learning has received much attention from the scientific community. For robots to think and act like humans, traditional algorithms have yet to be able to meet. Deep learning affects many areas, including autonomous vehicles and SLAM. Many works have used Deep learning in computer vision, and SLAM problems, such as [8], [9], etc. In the past, geometry algorithms have received more attention because training deep learning models requires a lot of data. Currently, the data sets available are many and can be used for many different problems, including SLAM and odometry. Therefore, our proposed method follows a learning-based approach.

SLAM consists of two main modules: front-end and back-end (figure 1.2). The front-end module has the other name, odometry. Odometry is the module that uses information collected from sensors to estimate the vehicle's path and the vehicle's pose in this path. This module depends on sensors that take in vehicles, such as cameras, LiDAR, IMU, GPS, etc. However, the pose estimation is affected by external factors (moving objects like cars or persons). Therefore, the back-end module uses to optimize the result of estimation. This module not only uses information from the environment to pose estimation but also uses knowledge from the database to build a global map.

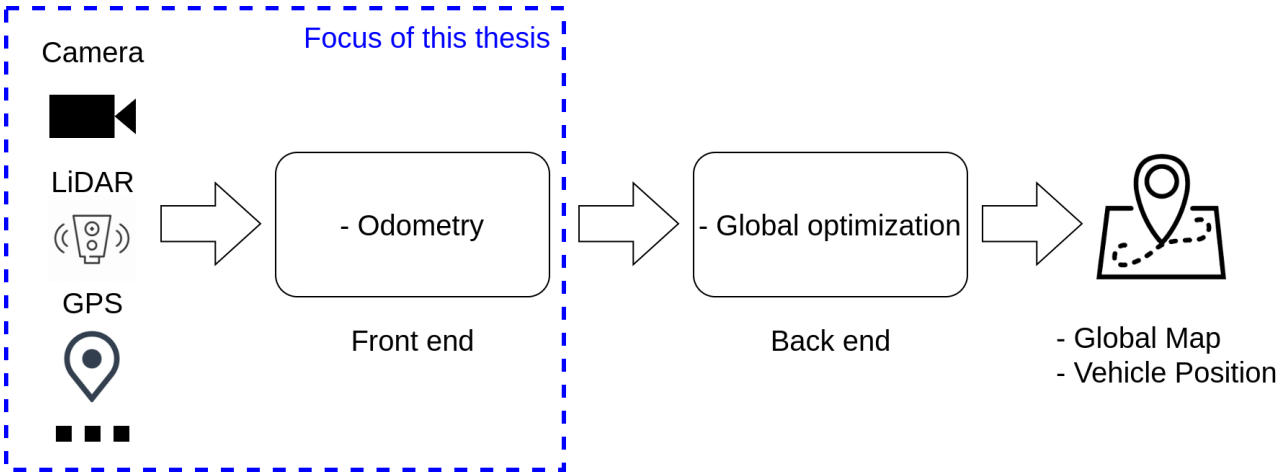

Figure 1.2: The SLAM components

This work aim to combine the sensor's information for pose estimation, focusing on the odometry in SLAM components. Autonomous vehicles work in the environment without any information about the surrounding objects. Odometry is the task that builds the map and localization at the same time in unknown environments. Odometry not only has applications in autonomous vehicles but also has applications in rescue systems or mapping buildings. The data from sensors is significant for self-driving and odometry. In other words, the sensors are "the eyes" of this vehicle. Odometry depends on the sensor used to collect data from posing estimation. Many kinds of sensors can be used, such as cameras, LiDAR, GPS, IMU, etc. They have different advantages and disadvantages.

Figures 1.3 and 1.4 show the autonomous vehicles and their sensors of Tesla and Google, respectively. Autonomous cars have many sensors equipped. If one sensor fails, the remaining sensors can be used instead. Therefore, odometry is a task that needs to be performed on an autonomous vehicle. It helps self-driving cars be less dependent on GPS signals in case they cannot work effectively.

1.2 Problem statement

The purpose of odometry is to estimate the path of the autonomous vehicle while it is moving. Figure 1.5 demonstrates the odometry problem. In this figure, the blue line is the path of the vehicles, and triangles represent for vehicle's poses at the time step

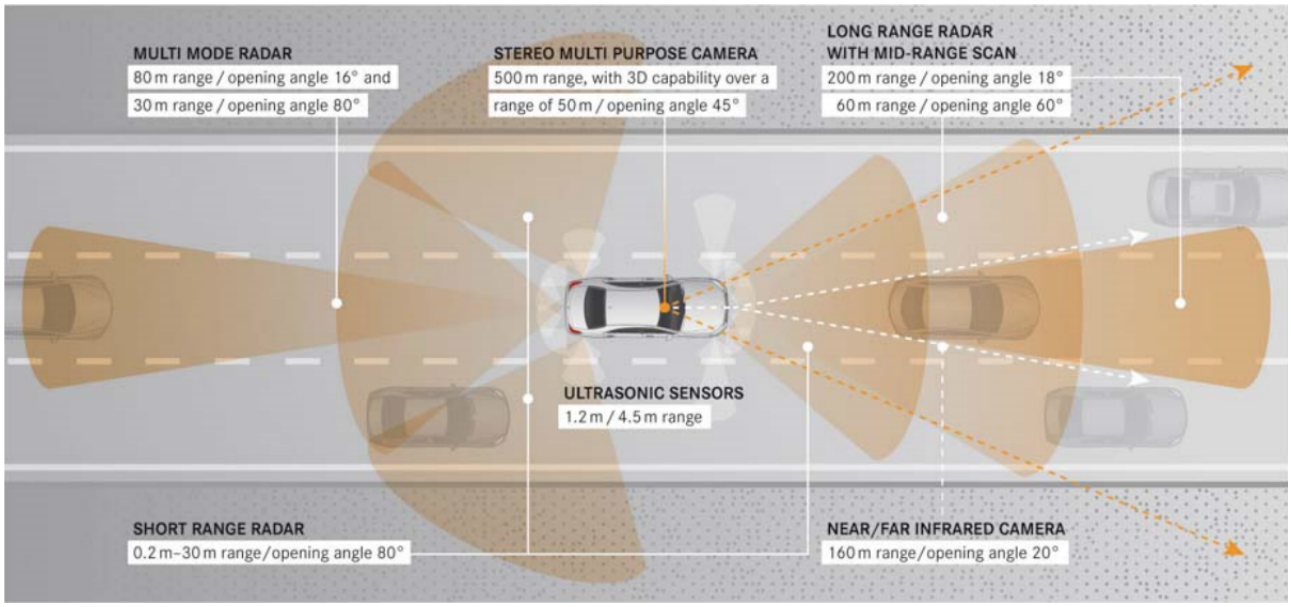

Figure 1.3: Tesla Model S and their model equipped [2]

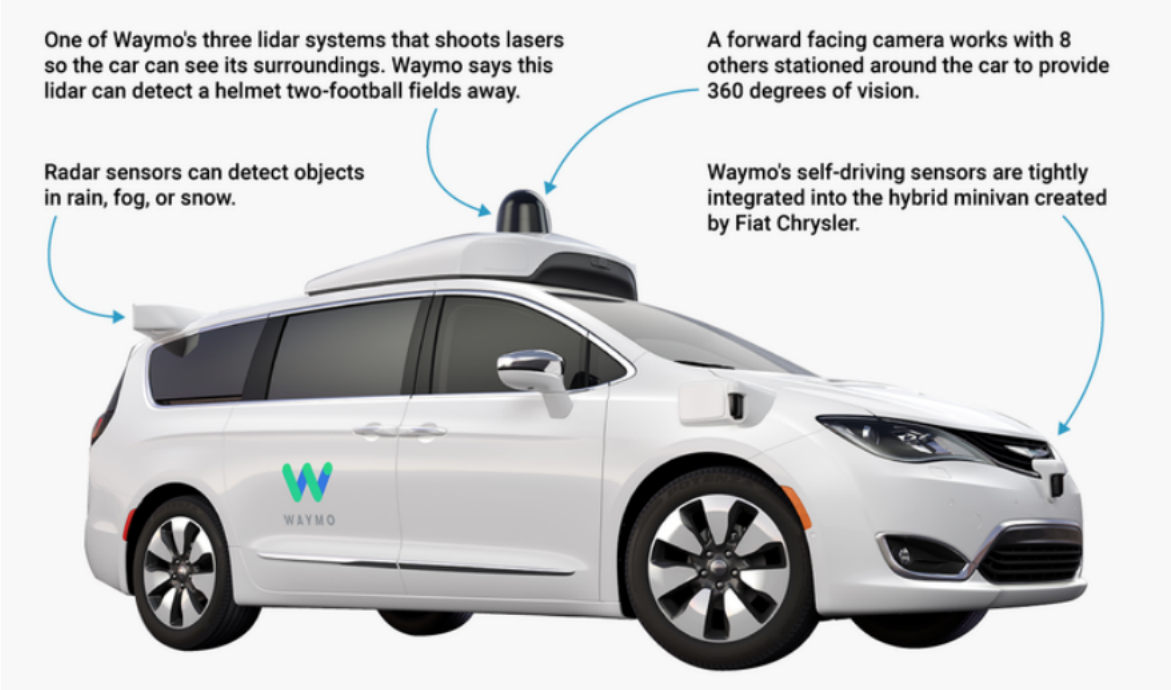

Figure 1.4: Waymo autonomous car of Google and their model equipped [2]

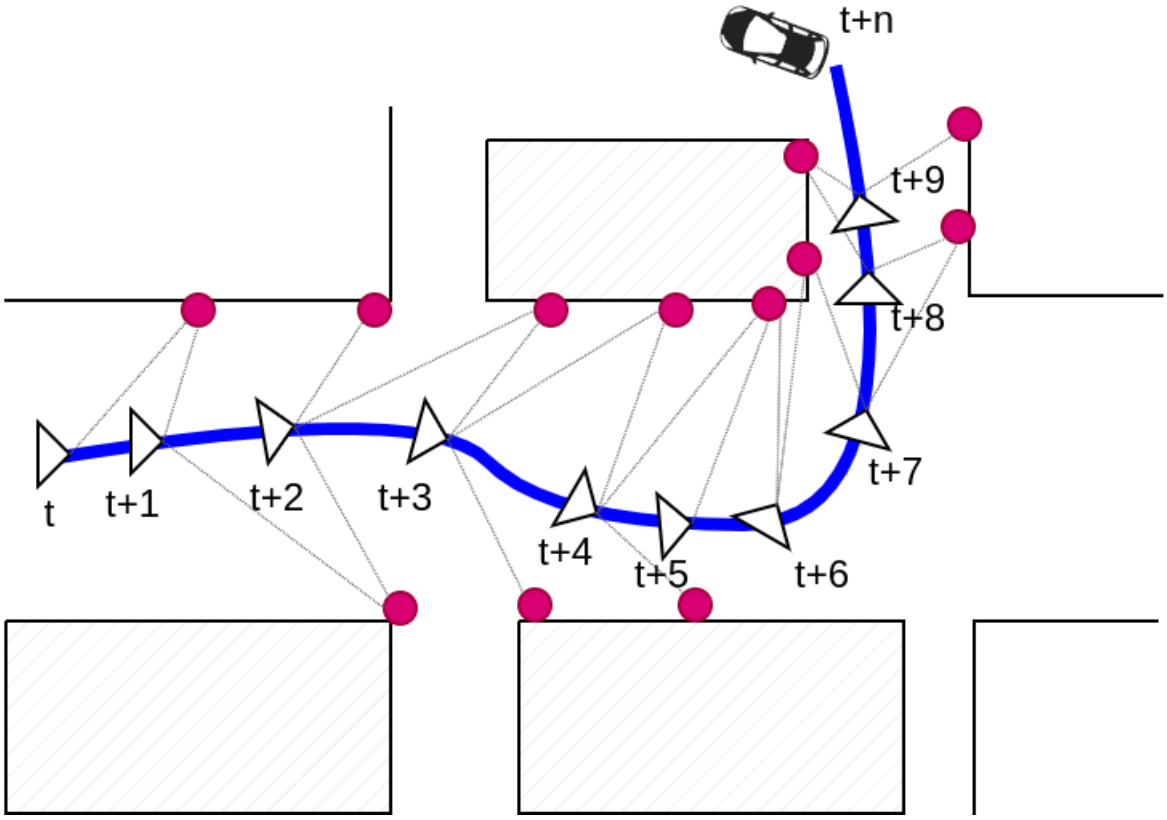

Figure 1.5: Demonstration of odometry for SLAM

$t, t+1, ..., t+n$. The red dots represent landmarks which are observations of sensors at each time. The landmarks are essential points that the autonomous vehicle can observe. The vehicle's positions are estimated based on the change of that landmark's position in each frame.

Autonomous vehicles work in unknown environments, so odometry uses data obtained from sensors mounted on the autonomous vehicle to estimate the vehicle's path. Many kind of sensors can be used to perform this task. The camera is a common type of sensor because it is convenient and easy to use. However, the data obtained from the RGB camera without depth information will cause inconvenience in estimating the 3D position. Besides, RGB-D interferes with the light, so it cannot be used in outdoor environments.

Another kind of sensor also common in autonomous vehicle systems is LiDAR. The advantage of LiDAR is that it uses lasers to measure the distance to objects in the surrounding environment. Therefore, the data obtained from LiDAR has depth information

which is very useful for estimating the position of autonomous vehicles. However, the raw 3D point clouds are too large to process directly and maybe contains useless information.

Visual information from the camera and depth information from LiDAR compensate each other to enhance accuracy for odometry. This study will combine LiDAR and a camera for odometry in the SLAM problem.

1.3 Research Motivation

Data collected by sensors are essential information for the SLAM problem because autonomous vehicles work in unknown environments. LiDARs and cameras have become popular sensors in the robotics research area. They have their advantage and disadvantages. Researchers have made many efforts for sensor fusion because it brings many benefits to various problems. For example, when there is an image and a 3D point cloud of the object. That information is combined to 3D reconstruct that object.

Information fusion is a hot topic in research because the amount of data is increasing day by day. Depending on the research area, the kind of information fused varies. Information fusion is necessary for SLAM and other research areas such as object detection, visual question answering, etc. In computer vision, the kind of information includes depth or visual information. This thesis's research motivation is to combine LiDAR and camera information for odometry in SLAM.

1.4 Technical Challenges

Odometry is a challenging research topic. There are several challenges for odometry in outdoor environments. This thesis focus on 3 challenges:

- **Feature selection from input:**

  One of the main challenges of odometry is the information extraction from data collected by sensors. The vehicle's movement in 3D space is calculated based on the change in position of each key point in the captured viewpoints. Vehicle displace-

ment is estimated based on the change in the position of key points between frames. Therefore, extracting key points from input data is essential in odometry. Keypoint selection must ensure that the number of points is enough for the algorithm to estimate efficiently.

- **Removing dynamic objects in frames:**

  Outdoor environments include dynamic and static objects. Key points extracted from those static objects are helpful for odometry, while dynamic things affect the transition estimation. If the key points are on a dynamic object, the change in position of the feature between frames is not consistent. Therefore, moving objects must be removed from the frame to improve accuracy is also a challenge in odometry.

- **Reducing the size of the 3D point cloud from LiDAR:**

  3D point cloud provides valuable information to the system to estimate the vehicle's position. Because it uses laser beams to measure distance, LiDAR is able to obtain information about objects farther away than the camera can provide. Therefore, such information is considered unnecessary and should be removed. The 3D point cloud needs to be projected to another representation to reduce the size of the 3D point cloud.

1.5 Contribution

The main contributions of this thesis are summarized as follows:

- Based on the deep learning approach, an end-to-end Visual-LiDAR odometry for SLAM is proposed. Inputs include the image sequence and the corresponding 3D point cloud sequence. In the method, sub-modules are implemented to solve the odometry problem.

- Essential regions are extracted instead of features or key points from RGB images. The self-attention mechanism is applied in the essential regions extraction module.

- Guided attention is applied to that information for depth and visual information fusion. The attention mechanism is used to find the correlation between input entities. When applying it to the model, the correlation between the information is also determined.

1.6 Thesis Organization

The structure of this master thesis is as follows:

- **Chapter 1: Introduction**

  This chapter introduces the background, SLAM components, problem statement, research motivation, technical challenge, contribution, and thesis organization. This chapter also gives a brief proposed method overview.

- **Chapter 2: Related Works**

  This chapter describes some related studies about odometry in SLAM. These include some approaches using camera, LiDAR, and Camera-LiDAR fusion.

- **Chapter 3: Proposed Method**

  The detail of the components in the proposed method's network structure is described. The effects of the attention mechanism are also given in this chapter.

- **Chapter 4: Experiment**

  This chapter shows the experimental result of our proposed method on the KITTI dataset. In addition, some estimation results compared with the ground truth are illustrated. Some explanations about the experiment are also given.

- **Chapter 5: Conclusion and Future Work**

  Finally, a summary of the present method is given in this chapter. Besides, this chapter discusses some possible improvements for the proposed method.

Chapter 2

Related works

Odometry is a task that estimates the autonomous vehicle's path based on data collected from sensors. Because autonomous vehicles or mobile robots work in an unknown environment, odometry has to be solved based on data collection from vehicle sensors. Some previous SLAM works are also referred to find out the odometry research status because odometry is the first module in the SLAM. Because autonomous vehicles or mobile robots work in an unknown environment, odometry has to be solved based on data collection from vehicle sensors. This chapter briefly reviews some approaches related to this study. There are Visual Odometry (VO), LiDAR Odometry (LO), and Visual-LiDAR Odometry (VLO), which are different kinds of sensors.

2.1 Visual Odometry

Visual Odometry (VO) aims to estimate the camera's pose based on an image sequence. VO is one of the first approaches in odometry problems because the camera is easy to use and low cost. Figure 2.1 shows the VO principle. The transitions are estimated based on the change of points' position in 2 continuous frames.

In general, the traditional VO approach includes many modules continuously. Figure 2.2 shows the VO pipeline. Input is the image sequence, and output is the camera's pose. The feature extraction module extracts key points on each image input. Next, feature

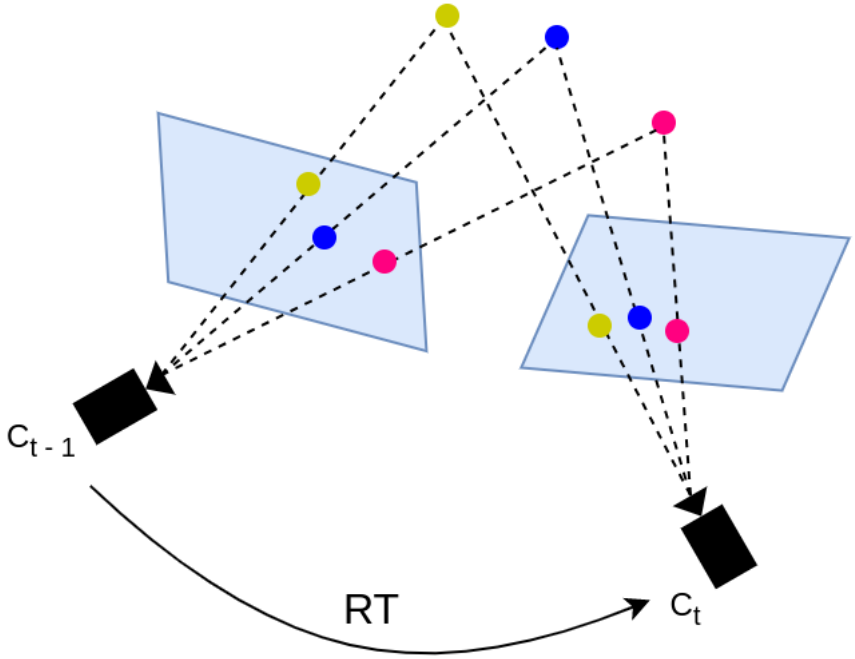

Figure 2.1: The Visual Odometry principle

matching is applied to calculate these key points' movement between frames (figure 2.3). The last module estimates the camera poses based on this movement's change. Because the camera is mounted on the vehicle, the camera's position is also the vehicle's. The traditional VO approach also calls the geometry-based approach because they rely on a change in the position of the feature in the image.

Several well-known methods exist, such as MonoSLAM [10], PTAM [11], and ORB-SLAM [12]. MonoSLAM [10] is one of the first monocular SLAM methods. Shi-Tomasi features [13] are extracted from images and used for pose estimation. Other methods apply different features to VO, such as ORB [14], SIFT [15], and SURF [16]. However, the real world is in 3D dimension, and monocular cameras deliver 2D images. Therefore, ORB-SLAM2 [17] expands the single camera to stereo cameras and RGB-D cameras. Stereo cameras estimate the depth by the triangular algorithm but have high computational costs. Besides, light conditions affect RGB-D cameras; RGB-D cameras do not work well in outdoor environments.

In recent years, the learning-based approach for VO also has been attended by researchers because of those advantages. In the feature extraction module, Bruno et al. [18]

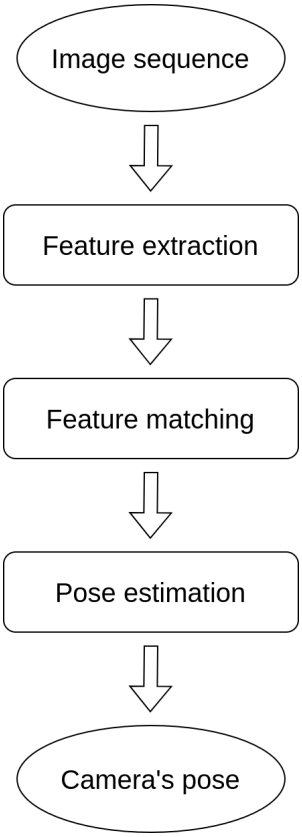

Figure 2.2: The Visual Odometry pipeline

applies the LIFT features [9] instead of hand-crafted features and use them to solve the VO problem. Besides, Handa et al. [19] applies VGG16 [20] to extract deep features. In the pose estimation model, the authors [19] use photometric consistency between frames to learn the change in the camera's position. Some researchers propose the learning-based end-to-end approach for odometry. PoseNet [21] applies a CNN to estimate the 6DoF of image sequence input. Wang et al. [8] use CNN for feature extraction and an RNN to estimate the change of position of the sequence. DeepSLAM [22] expands stereo images in the training phase to estimate the depth information of images and uses a single image in the testing phase.

Moving object tracking is a technical challenge for SLAM and Odometry. DS-SLAM [4] is a method that works in dynamic environments with moving objects in image sequences. This framework is shown in figure 2.4. They use an RGB-D camera to collect data. They apply a CNN model to segment objects in the image. Also, they extract the ORB feature in the image and check moving objects. The moving points are considered an outlier, and

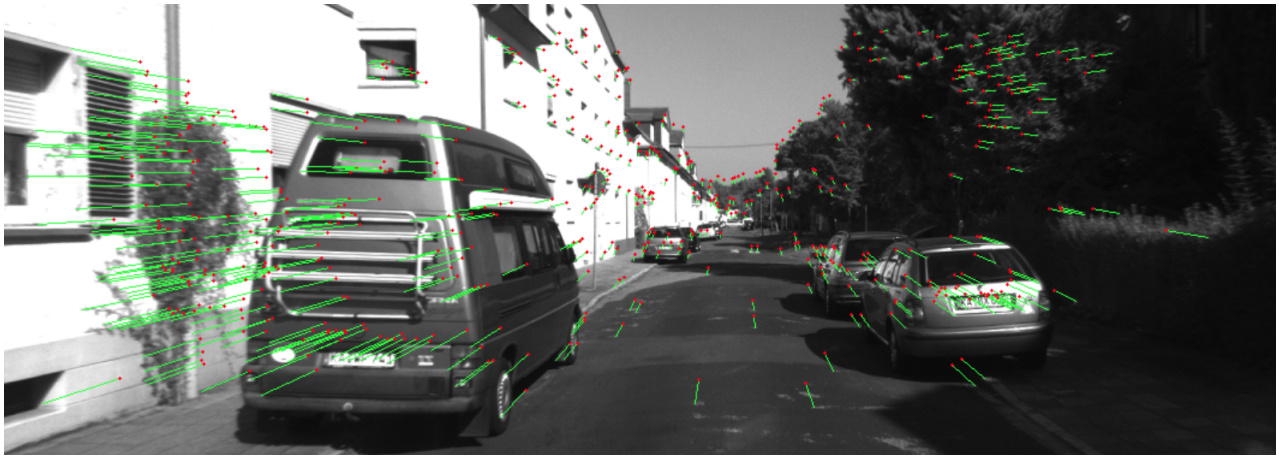

Figure 2.3: Motion estimation of the feature in the image [3]

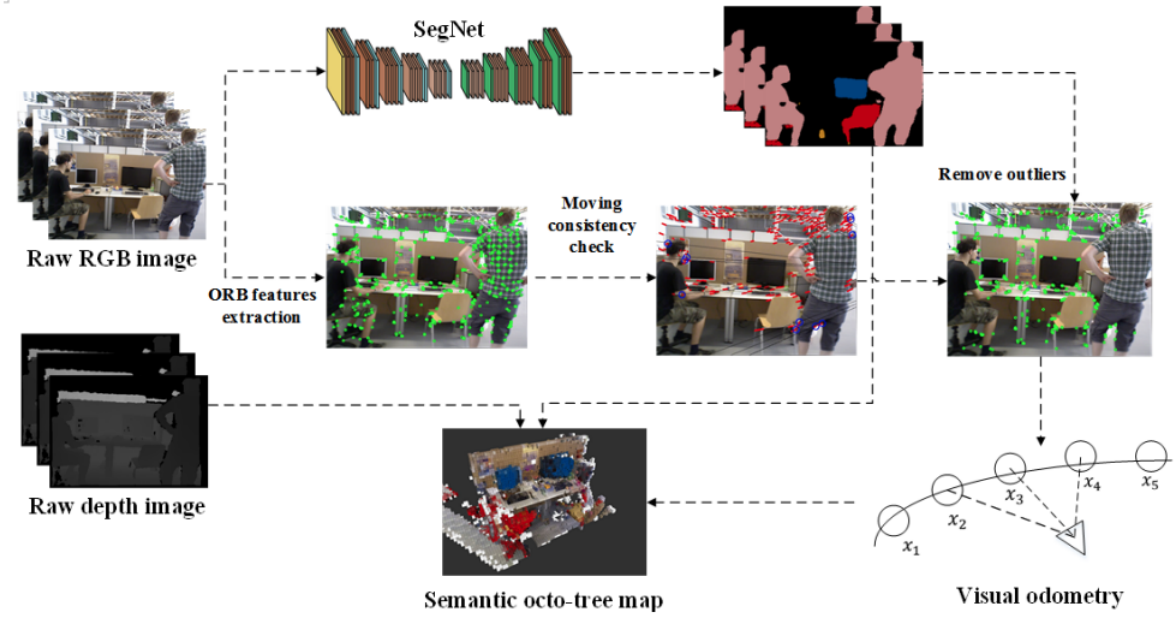

Figure 2.4: The DS-SLAM pipeline [4]

the objects containing those points are removed, then pose estimation. Then based on the pose, depth image, and semantic segmentation results, the semantic map is built.

2.2 LiDAR Odometry

LiDAR (Light Detection And Ranging) is a popular sensor in autonomous vehicles. LIDAR emits laser light to its surroundings and detects the returning laser. Then, it calculates the distance between it and objects in the surrounding environment by measuring

the time between when the lasers are emitted and back. The distance between LiDAR and objects in the surrounding environment is proportional to the time it takes to transmit and receive laser beams. The formula for calculating the distance is:

$$d = \frac{c_0.t}{2} \tag{2.1}$$

where d is the distance (in meters), c_0 is the speed of light and t is the transmission and reception time. Every second, LiDAR emits millions of lasers to collect 3D data of the surrounding environment [23]. The advantages of the 3D point cloud are long-range and unaffected by lighting conditions. Therefore, LiDAR odometry is also a research direction of interest to researchers.

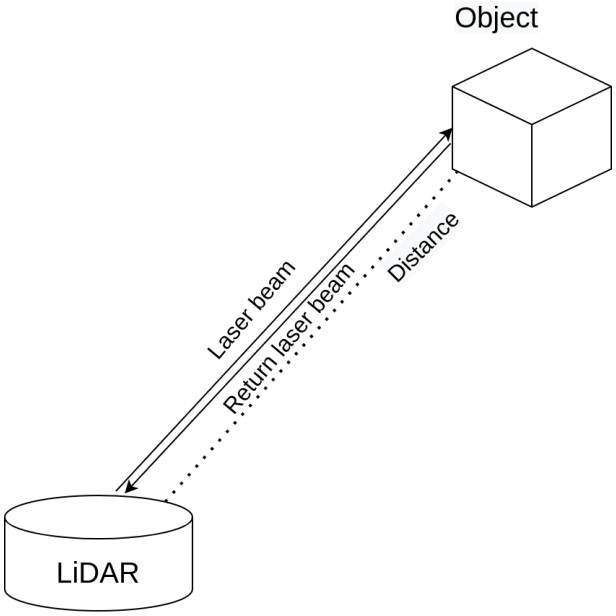

Figure 2.5: The mechanism of the LiDAR sensor

ICP [24] is one of the first methods for LiDAR odometry. The ICP method calculates the change between 3D point clouds and estimates the vehicle's path. LiDAR odometry And Mapping (LOAM) [25] simultaneously estimate pose and map construction. First, this algorithm extracts plane and edge from 3D point cloud input. Then, the transition is estimated between two scans using these features. After estimating the transition between frames, they use this position and feature to build and update the 3D map. This method also uses ICP for estimation.

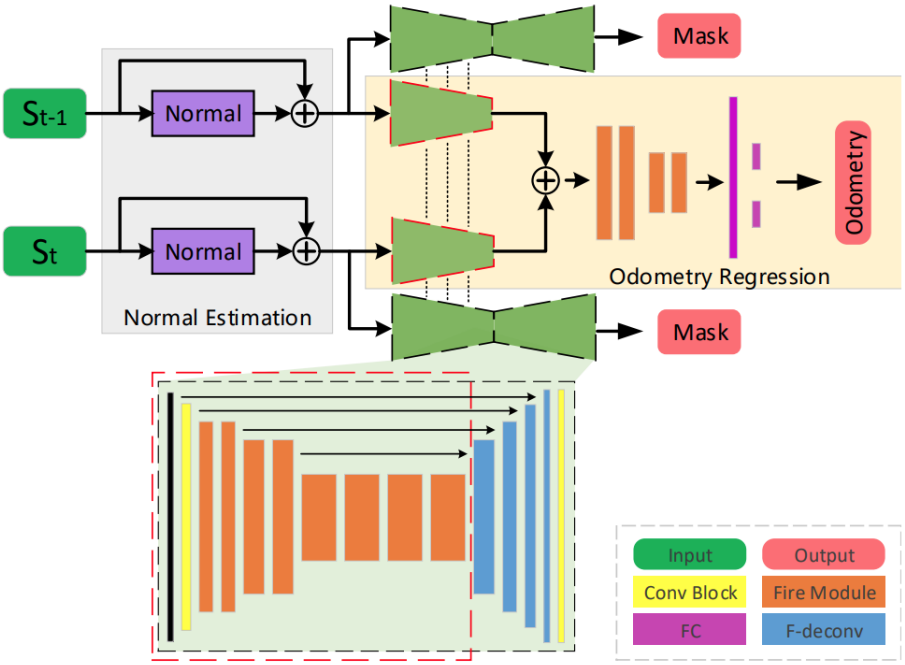

Figure 2.6: Lo-Net network architecture [5]

The LO-Net [5], a famous learning-based LiDAR odometry method, was introduced by Li et al. in 2019 (figure 2.6). Figure 2.6 The input includes consecutive LiDAR scans S_{t-1} and S_t, and the output is the 6DoF value between scans and mask of the moving object in this scan. Each scan S is normalized before pushing it into the deep neural network. This network also extracts the moving objects' mask of each scan.

DeepLO [26], an end-to-end LiDAR odometry, include FeatNet [27] and PoseNet [21]. FeatNet [27] extracts the 3D point cloud to feature vector for a compact representation. Each feature vector between time t and $t+1$ is forwarded to PoseNet [21] to estimate the relative motion between two frames.

Besides, the researchers try to segment the semantic information from 3D point clouds. Li et al. [28] proposed a semantic LiDAR SLAM, which includes a point cloud semantic segmentation network. This semantic information is beneficial for pose estimation tasks and map reconstruction. L3-Net [29] applies PointNet [30] to extract 3D key points from LiDAR data. In [31], CAE-LO [31] extract features from the 3D point cloud and estimation by RANSAC [32].

2.3 Visual-LiDAR Odometry

Besides using the mono sensor, the sensors are also combined for the odometry problem. Because the real world is the 3D dimension, LiDAR information has a good representation of the real world. 3D point clouds can represent surrounding environments. However, as discussed in the previous section, raw LiDAR data is informative but very large for direct processing. In addition, 3D point clouds have no semantic information; all objects are represented as point sets. The 2D image from the camera does not have the depth information of the environment. Still, the visual information from the image brings many benefits to odometry. Therefore, 3D point clouds can combine with the images from the camera.

V-LOAM [25] is one of the earliest methods for Visual-LiDAR odometry. This method uses VO to estimate the camera pose and apply it to a LiDAR scan to correct the point cloud. After correcting the pose of the 3D point cloud, this method estimates the relative pose between two frames. This relative pose between two 3D point clouds corrects the visual estimated position for VO.

LIMO [33] considers data from LiDAR for depth extraction. The 3D point cloud from LiDAR is translated to the image coordinate and extracts the depth of visual features. These features with depth information are applied to camera pose estimation.

Shi et al. [6] proposed a method for LiDAR-mono odometry using line and point features. After extracting point and line features from RGB images, they use these features to pose estimation. 3D point cloud from LiDAR adds depth information for features from images. Then, 3D features are updated to landmarks for global optimization. Figure 2.7 shows the illustration for point and line depth extraction. After the LiDAR data (grey points) is converted to the image plane, the depth information of points and lines are extracted.

Another method - DVL SLAM [34] - include RGB image and LiDAR scan as an input for each frame. The authors use image sequences for motion estimation and use this estimation to correct the distortion of LiDAR.

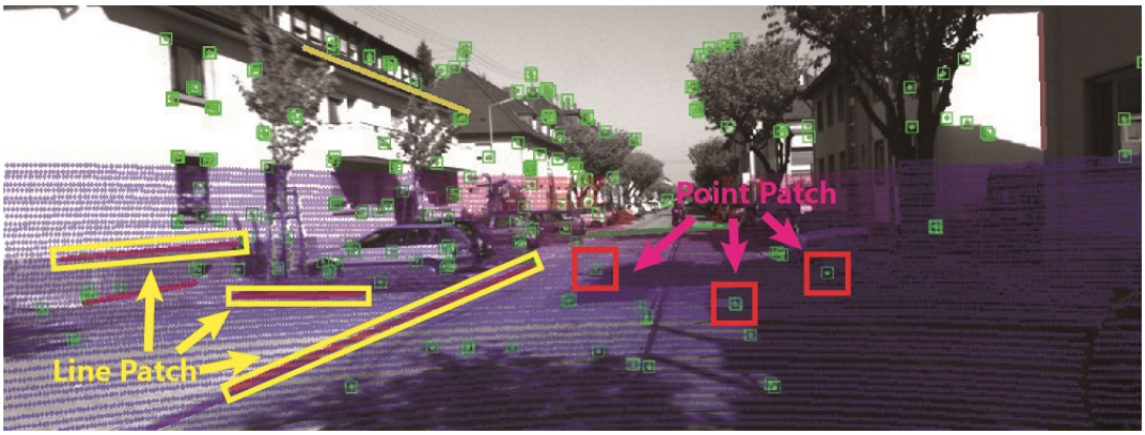

Figure 2.7: Point and line depth extraction of [6]

Youngwoo et al. [35] combine LiDAR and camera information for odometry differently. This method built two separate maps based on information from 2 sensors, including a LiDAR map and a Visual Map. Then, they apply the visual feature to the LiDAR map for pose tracking. The visual map is used to normalize the estimation results.

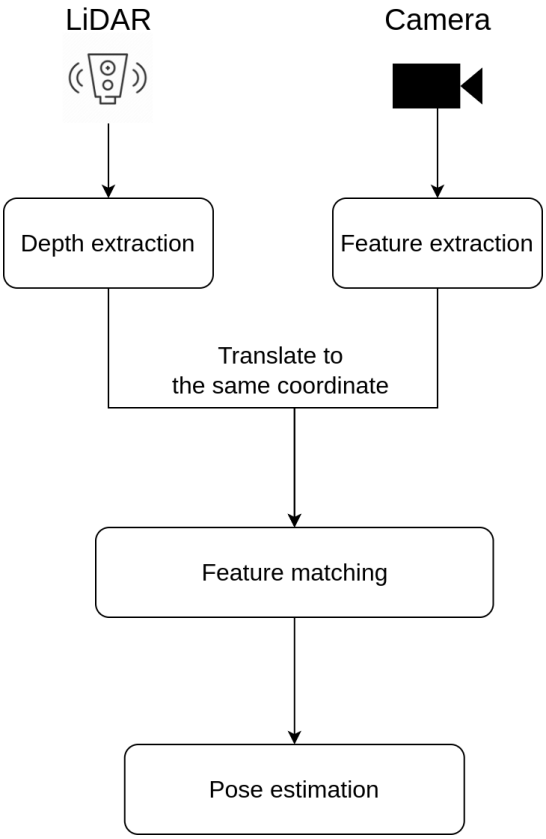

Figure 2.8: The pipeline of Visual-LiDAR Odometry

The general pipeline of Visual-LiDAR odometry is shown in figure 2.8. This approach

mainly extracts information from images and 3D point clouds. Then the features are brought to the same coordinate and estimate pose.

2.4 Summary

This chapter brings an overview of odometry - the front-end module of the SLAM problem. These include the Visual Odometry approach, LiDAR Odometry approach, and Visual-LiDAR Odometry approach. This section discusses the advantages and disadvantages of the previous approaches. A describes the difference between the proposed method and previous works also is given.

- **Visual Odometry (VO):**

  VO is one of the first approaches and receives much attention from researchers. VO mainly uses the change between frames to find the camera's position. Besides geometric features, learning-based features are also applied for motion estimation. The camera is a sensor that is easy to use, convenient, and inexpensive. However, the image from the camera is a 2D image, so it does not have depth information.

- **LiDAR Odometry (LO):**

  LO approach also gets a lot of attention because its accuracy is higher than VO. Because data from LiDAR is a 3D point cloud, it is a more accurate representation of the natural world because the real world is also a 3D dimension. However, 3D point clouds do not contain visual information. The previous works only estimate based on the change of point position.

- **Visual-LiDAR Odometry (VLO):**

  The VLO approach combines VO and LO to take advantage of those two methods. Previous studies extract features in images and fusion depth for this feature by the 3D point cloud. However, the previous methods still process 3D point clouds directly.

The main difference between our proposed method and previous works is feature extraction and information fusion. Previous studies often extract points or lines as features. Instead of extracting feature points, our method extracts essential regions via the self-attention mechanism. A region on an image consists of many adjacent points, which is more robust than individual key points.

The moving object in the frame can be considered noise. Previous works applied a CNN to predict potentially moving objects and remove them. Besides, preparing annotations for static objects in video sequences and the corresponding LiDAR point clouds requires massive human effort. Therefore, this thesis aims to extract the essential region that does not include moving objects.

In previous works, the depth information from the 3D point cloud is used to complement the feature from 2D images. This work apply guided attention for information fusion instead of directly as they do. The attention mechanism emphasizes the interaction between objects in RGB images and 3D point clouds.

Chapter 3

Proposed Method

3.1 Problem description

Given the RGB image sequence and 3D point cloud correspondence as input, the output is the path of this vehicle (figure 3.1). The path is define as a sequence:

$$P = (P_{t-2,t-1}, P_{t-1,t}, P_{t,t+1}..., P_{t+n-1,t+n})$$

The transition $P_{t-1,t}$ between frames $t-1$ and t is represented as Homogeneous Transformation Matrices (3.1):

$$P_{t-1,t} = \begin{bmatrix} R_{t-1,t} & T_{t-1,t} \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

where $T_{t-1,t}$ and $R_{t-1,t}$ are translation and rotation value between two frame $t-1$ and t. For detail, with an RGB image I_t and a 3D point cloud D_t, the system estimates the matrix C_t representing the vehicle position at the time t. The camera's pose is also the vehicle position because the camera is mounted on the car. In general, formula to calculate the position of the camera C_t at time t when the position of the camera at time $t-1$ is available as follows:

$$C_t = R_{t-1,t} * C_{t-1} + T_{t-1,t} \tag{3.2}$$

where:

$$R = \begin{bmatrix} \cos\phi\cos\kappa & \cos\omega\sin\kappa + \sin\phi\sin\omega\cos\kappa & \sin\phi\sin\kappa - \cos\omega\sin\phi\cos\kappa \\ -\cos\phi\sin\kappa & \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa & \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa \\ \sin\phi & -\sin\omega\cos\phi & \cos\omega\cos\phi \end{bmatrix} \tag{3.3}$$

is the rotation matrix around the axes Ox, Oy, Oz in 3D dimension by the angles ω, ϕ, κ (roll, pitch, yaw) and $T_{t-1,t.} = (T_x, T_y, T_z)$ is the robot translation vector at time $t-1$ to time t.

Therefore, the 6-DoF pose of the sensor between continuous frames need to be recover, represented by:

$$E_{t-1,t} = (t_x, t_y, t_z, \omega, \phi, \kappa) \tag{3.4}$$

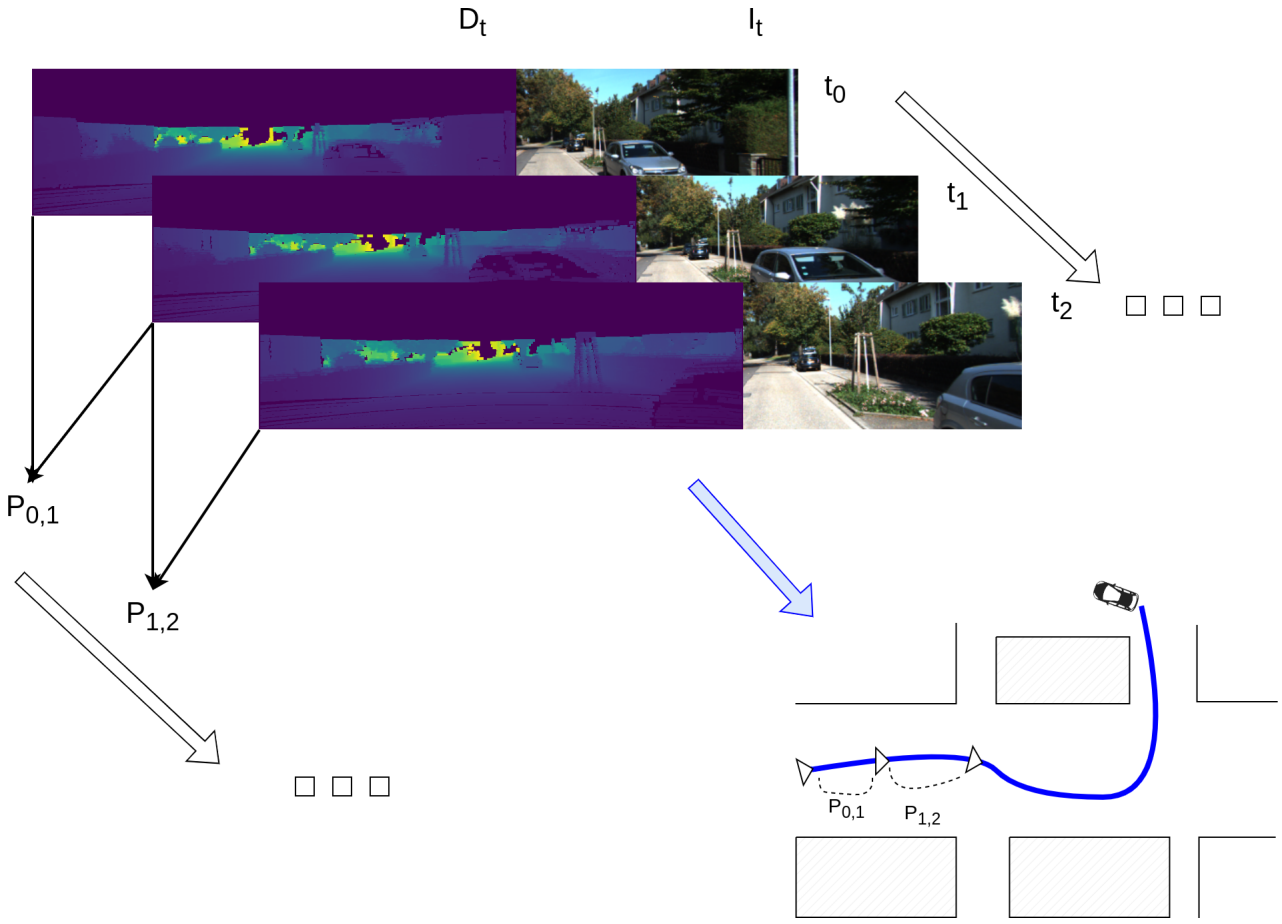

Figure 3.1: Odometry problem description

3.2 3D point cloud preprocessing

3.2.1 LiDAR-camera calibration

LiDAR-Camera calibration is a task of converting a 3D point cloud and a 2D image at the same time t to the same coordinate system 3.2. While the camera captures the visual information, the LiDAR captures the surrounding environment's 3D point cloud - including depth information. Those data are on LiDAR and camera's coordinate systems. In this study, the vehicle's position is as the camera's. The 3D point cloud from LiDAR coordinate system need to convert to the camera coordinate system.

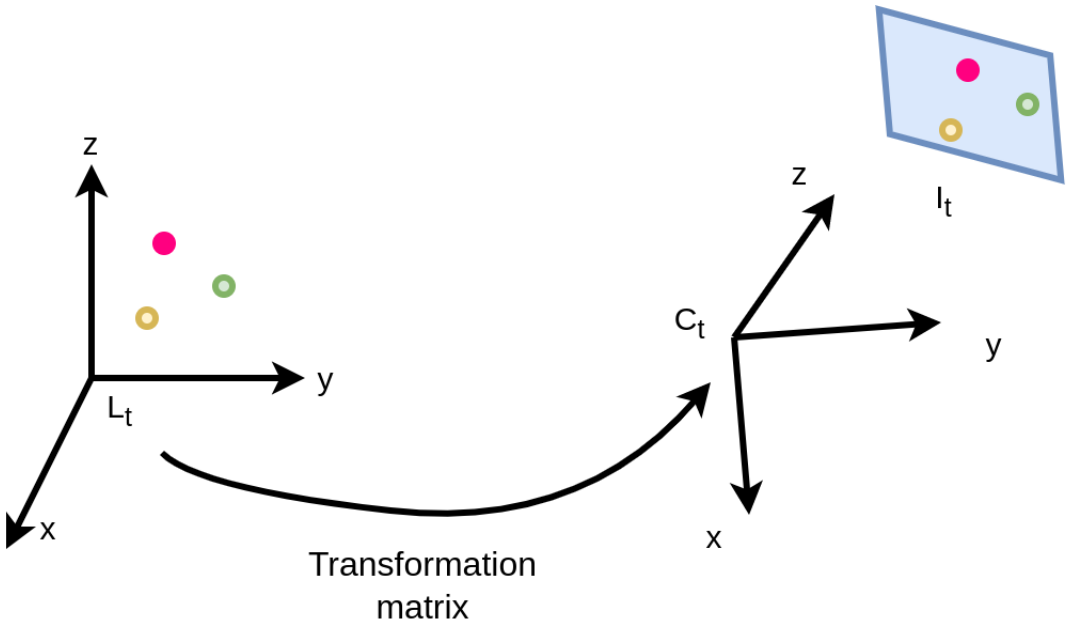

Figure 3.2: LiDAR-camera calibration example

To translate the 3D point cloud from the LiDAR coordinate to the Camera coordinate, the coordinate system transformation matrix between the camera and LiDAR must be know. The prosed in the [36] approach determines the transformation matrix. LiDAR-camera transformation matrix is:

$$P_L^C = \begin{pmatrix} R_L^C & t_L^C \\ 0 & 1 \end{pmatrix} \tag{3.5}$$

Here, L represents the LiDAR coordinate, and C represents the Camera coordinate.

R_L^C and t_L^C are rotation matrices and the translation vector from LiDAR coordinate to the camera coordinate. 3D point cloud in camera coordinate D_t^C at the time t is translated by 3D point cloud in LiDAR coordinate D_t^L at the same time is:

$$D_t^C = P_L^C D_t^L \tag{3.6}$$

3.2.2 3D point cloud to range image

3D point cloud from LiDAR brings high-accuracy surrounding information and is unaffected in light conditions. The large-volume data obtained by LiDAR is inconvenient when practicing such as storage or real-time processing. Because it emits lasers to get environmental information, the 3D point cloud contains depth information that the camera cannot capture. This thesis aim to fuse depth information and visual information; the depth information not in visual information is redundant. The 3D point clouds are projected into 2D range images in this work. 3D point cloud representation by 2D image is a common but effective approach for LiDAR information preprocessing. The 3D point-based representation is a formula follow:

$$D = \{p_1, p_2, \ldots, p_n\} \tag{3.7}$$

where each point have the value $p_i = (x_i, y_i, z_i)$ in LiDAR coordinate. Each point p_i is converted to the image coordinate, as defined by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - arctan(y, x)\pi^{-1}] * w \\ [1 - (arcsin(z, r) + f_{down})f^{-1}] * h \end{pmatrix} \tag{3.8}$$

where the range value $r = \sqrt{x^2 + y^2 + z^2}$ is the Euclidean distance from this point to the LiDAR origin coordinate. $f = f_{up} + f_{down}$ is the vertical field-of-view (FOV) of this LiDAR sensor. w and h are the weight and height of the range image.

Besides the 3D point cloud size reduction by range image representation, the number of points in the 3D point cloud is also decreased. The robot only needs to observe how

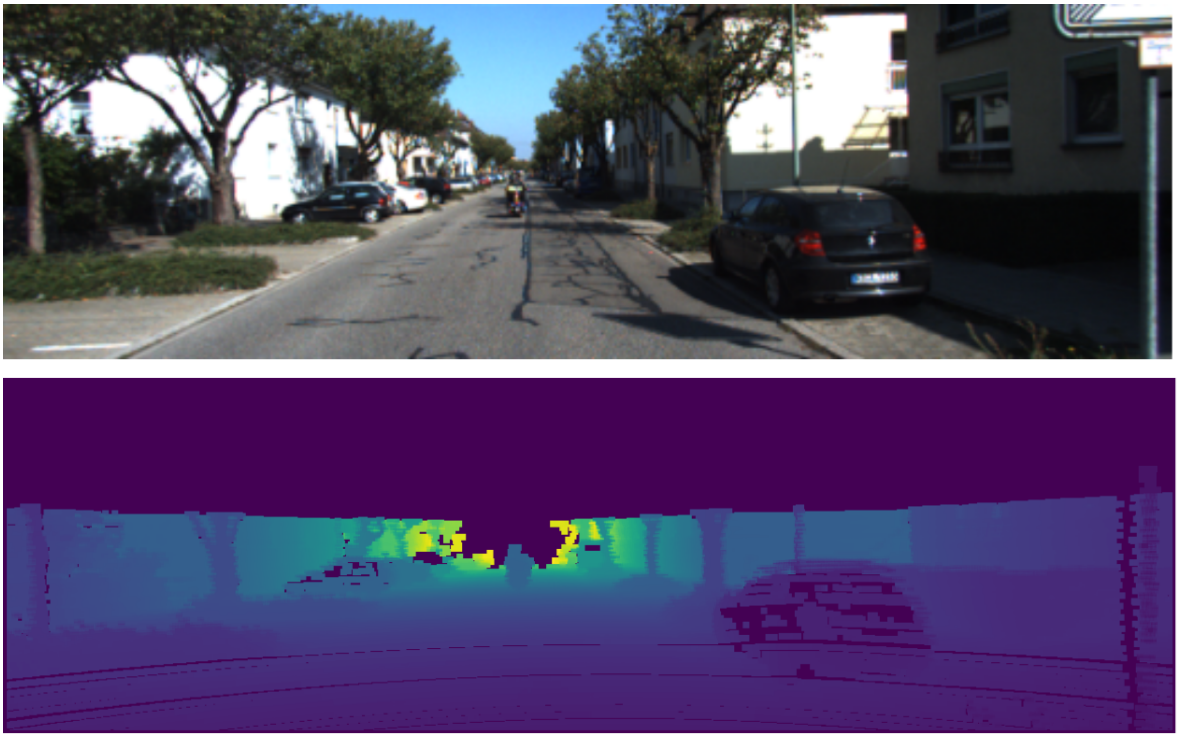

Figure 3.3: RGB image (top) and range image (bottom) at the same time

the position of the objects in front of them changes compared to the previous frame for motion estimation. Based on this assumption, the autonomous vehicle only needs to know the information in the front-of-view to determine its pose. Because the 3D point cloud contains 360-degree information, the size of the 3D point cloud is reduced by removing points which out of the front view of the camera. In this way, the 3D point cloud size is decreased and ensures depth information fits with visual information.

RGB and range images are shown in figure 3.3. The pixel's depth is brought only with calculations on the 2D image instead of the 3D point cloud based on equation (3.8). Computing 2D images will help save computational and system storage costs. Furthermore, 2D range images can inversely represent the 3D point cloud. Therefore, we choose the range images for data from LiDAR representation.

3.3 Network architecture

Figure 3.4 show the proposed network architecture. At each time t, the input includes an RGB image and 3D point cloud, and the transition consists of translation and rotation

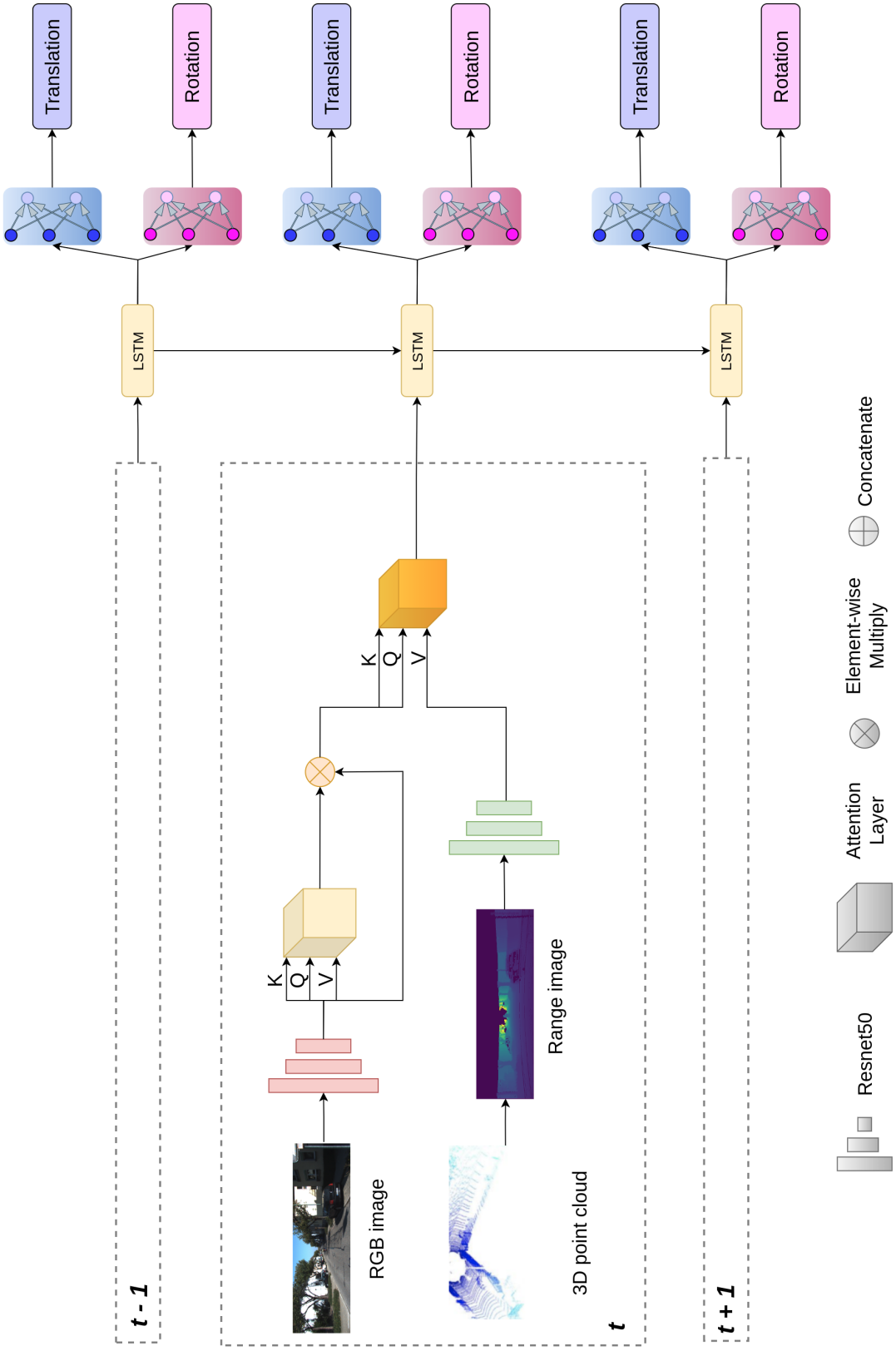

Figure 3.4: The proposed network architecture

between frames at the time t and $t-1$ is the output. Resnet50 [37] model extracts the feature from RGB image. Then, the feature vector is fed into the self-attention layer to generate a mask for the important region. Next, $Element-wiseMultiply$ is applied to fuse the important region's mask with this feature vector to extract the important region in the RGB image. The 3D point cloud, after representing it as a 2D range image, also uses Resnet50 for feature extraction. A guided attention layer fuses visual and depth information. Then, this output is put into the LSTM network [38] for sequential information learning. Finally, fully connected layers extract the translation and rotation between this and the previous frame $t-1$.

3.3.1 Feature extraction by ResNet50

Image feature extraction aims to extract higher information in raw pixels and reduce the data dimension from input images of different sizes. Resnet is a well-known model because of its outperforming results in the image classification topic. ResNet has more variants, such as ResNet18, ResNet50, and ResNet101, with different numbers of layers.

ResNet50 takes as input a resized image with size 224*224. The model includes four convolution blocks for feature extraction. The end of the ResNet50 model is an average pooling and fully connected layer for image classification.

This work removes the average pooling and the fully connected layer for the feature extraction task. The different networks are applied for RGB images and range images. Figure 3.5 shows the Resnet50 model architecture used in this network. The network accepts an RGB image as input. The first is a 7×7 kernel convolution with 64 kernels and a 2-sized stride. The next is the max pooling layer with a 2-sized stride. After the first two layers, the subsequent layers are defined as blocks as follows:

- **Block 1:** the first layer is 3×3,64 kernel convolution, next is 1×1 with 64 kernels and 1x1, 256 kernels. This block is repeated three times, total 9 layers.
- **Block 2:** 1×1,128 kernel convolution in the first, 3×3,128 kernels, and 1×1,512 kernels. This block is repeated four times, 12 layers.

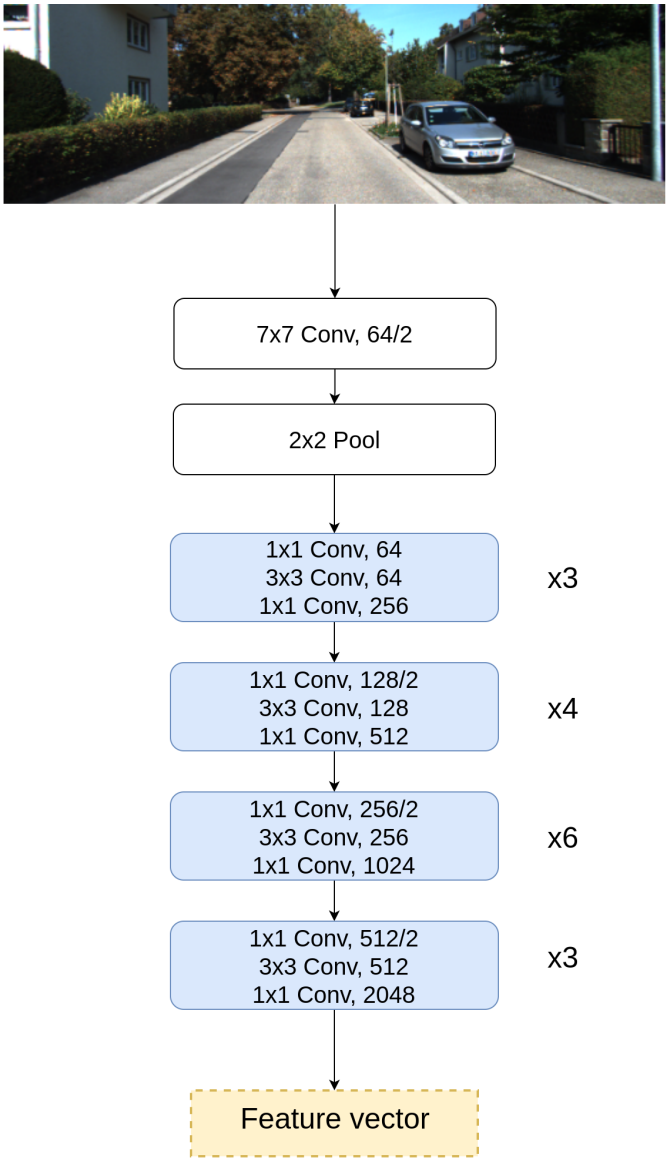

Figure 3.5: The feature extraction based on Resnet50 architecture

- **Block 3:** with 1×1,256 kernel, 3×3,256 and 1×1,1024, iterated six times, the total is 18 layers.
- **Block 4:** with 1×1,512 kernel, 3×3,512 and 1×1,2048; include 9 layers with three times repeated.

To better represent important information, ResNet50 is also applied to the range image from LiDAR. Since vehicle work in outdoor environments, a pre-trained model of ResNet50 on the ImageNet dataset, which includes objects in outdoor environments, is used for model training.

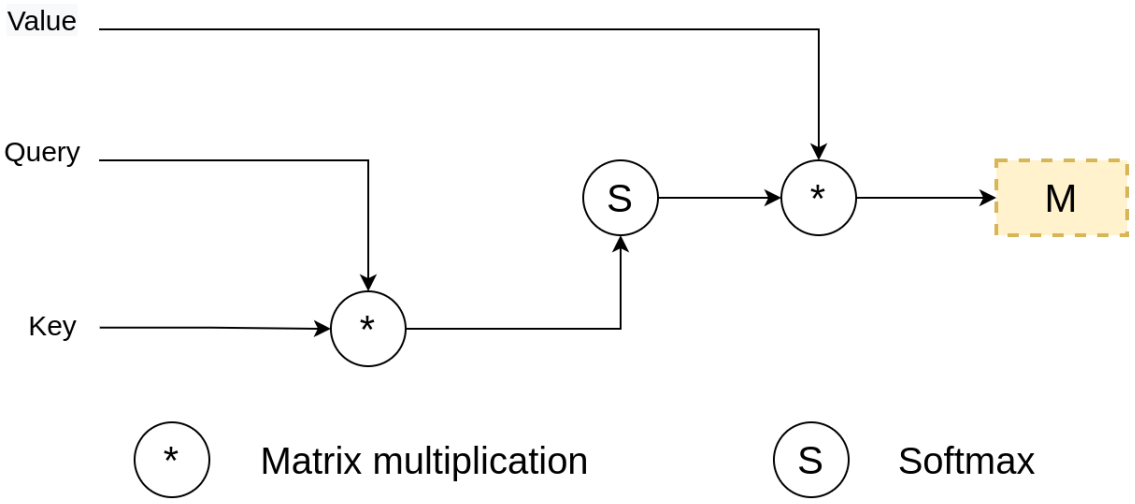

Figure 3.6: The attention mechanism visulization

3.3.2 Essential Region Selection via Attention-driven

The main difference between the proposed method and previous works is essential region selection via the Attention mechanism [39]. As mentioned in the technical challenges section, moving objects will adversely affect estimation results. Therefore, the motion estimation of the autonomous vehicle needs to be calculated on the static regions in the frame. In this thesis, essential regions are considered static regions in the frame. However, there are some static objects of no value in estimation, such as the sky or trees. Therefore, this thesis applies an attention mechanism to extract the frames' essential regions based on the above assumption.

When observing, humans tend to focus on the essential objects in sight. Based on that assumption, the Attention mechanism focuses on the essential regions on the input RGB image. The general attention calculation equation is as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.9}$$

Q, K and V are Query, Key, and Value, respectively.

Figure 3.6 describes the working of the attention mechanism. The Key and Query calculation of the attention score and the value vector use this score to determine the relationship between entities' input. Moreover, entities' relationships in the frame are found

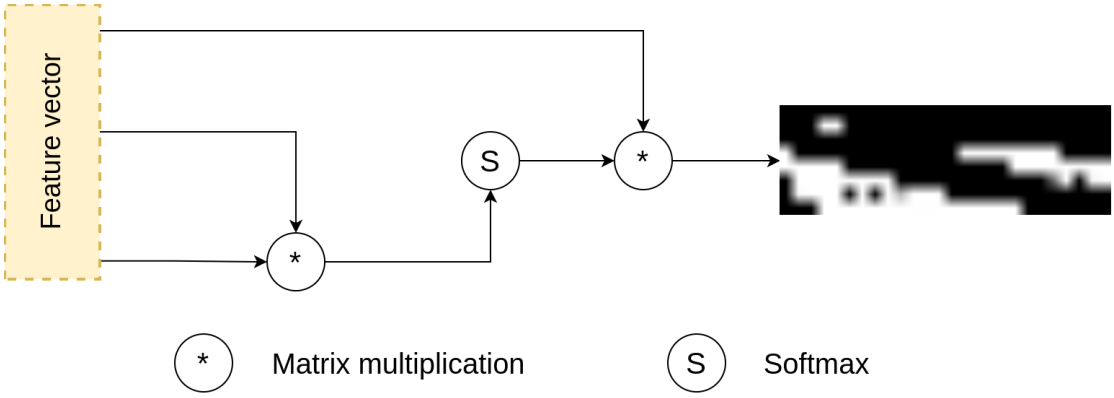

Figure 3.7: The self-attention layer

by applying Multi-head Attention. Equation 3.10 is the Multihead Attention equation.

$$MultiheadAttention(Q, K, V) = [head_1, head_2, ..., head_n]W^O \tag{3.10}$$

where W^O is the trainable matrix for all the heads. $head_j, j \in [1; n]$ determined based on user considerations. The details of each head are:

$$head_j = softmax(\frac{QW_j^Q(KW_j^K)^T}{\sqrt{d_k}})VW_j^V \tag{3.11}$$

In 3.11, W_j^Q, W_j^K, and W_j^V are the weight of Query, Key, and Value of $head_j$, respectively. These weights are updated during training.

In this work, the essential region in the RGB image is fused with depth information from the 3D point cloud by the attention mechanism.

First,self-attention generate the essential regions a from the feature vector I, extracted by the Resnet50 layer (3.12).

$$a = \text{SelfAttention}(I) = \text{MultiheadAttention}(I, I, I) \tag{3.12}$$

The self-attention layer in this work is shown in figure 3.7. The feature vector of the RGB image in previous layers is treated as the input to the self-attention mechanism. The output is a mask of essential regions a of the RGB image.

The previous work often applies attention maps to update the feature vectors. Unlike

them, this work filter the essential regions a by a convolutional layer and one kernel to generate the attention mask $\hat{a}$ with one dimension (3.13)

$$\hat{a} = \text{conv2D}_{\text{num\_kernel}=1}(a) \tag{3.13}$$

This one-channel attention mask plays the role of the essential region in the input image. The element-wise multiply operator $\odot$ is applied to keep only the essential regions of the image feature vector I (3.14).

$$s = I \odot \hat{a} \tag{3.14}$$

The purpose of essential regions includes key points that reside on static objects. An explanation is that the key points extracted from the static object will have more information. Static objects are considered landmarks to estimate the vehicle's change from the previous frame. However, data labeling for static and moving objects requires massive efforts from the human. Therefore, self-attention is used to learn to identify essential regions in an image without data labeling. Figure 3.8 shows some examples of region selection masks. The first two images are the input RGB images. The two middle images are the essential mask produced by self-attention driven, and the last two images result from applying the generated mask to input images. The self-attention layer focuses on statics region instead of dynamic objects such as people or vehicles.

Next, the Guilded Attention is applied for the fusion module, which fuses the visual and depth information for the Camera and LiDAR. The essential regions s is fused with depth feature vector L in (3.15).

$$g = \text{GuidedAttention}(L, s) = \text{MultiheadAttention}(L, s, s) \tag{3.15}$$

Figure 3.8: The essential region visualization

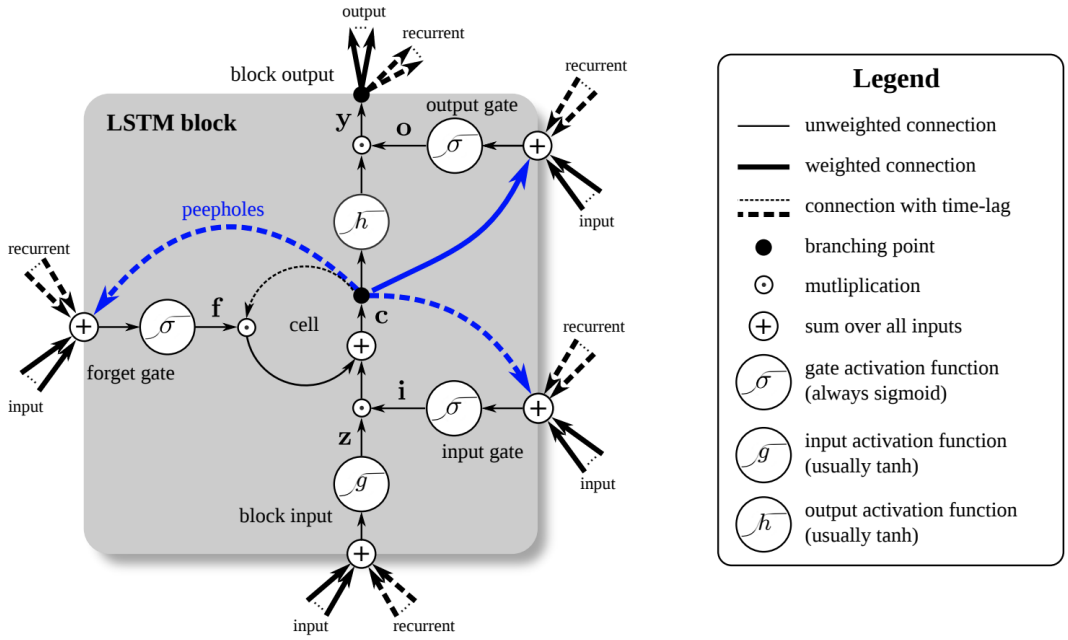

Figure 3.9: The Long Short-Term Memory block [40]

3.3.3 The pose estimation

Transition is estimation from fused information g in the previous layer. The transition of each frame includes translation and rotation values. An LSTM is applied to learn the sequential information from image sequence input. LSTM block has a memory cell, a self-hidden cell with a recurrent connection, and two gating units, including input and output gates (figure 3.10).

They control the information accessible to the memory cell. In addition, the LSTM has a forget gate, which learns the behavior of memory self-reset. Therefore, the LSTM model is suitable for our problem. The LSTM layer in our proposed method is formula (3.16). Figure 3.10 shows this work's visualization of the pose estimation module.

$$l_t, h_t = LSTM(g_t, h_{t-1}) \tag{3.16}$$

where h_t is the hidden state, including sequential information. l_t is the output of LSTM layer. The number of sequence lengths is 3 for the LSTM input. Besides, bidirectional LSTM [38] is applied instead of feed-forward LSTM. The bidirectional LSTM learns the

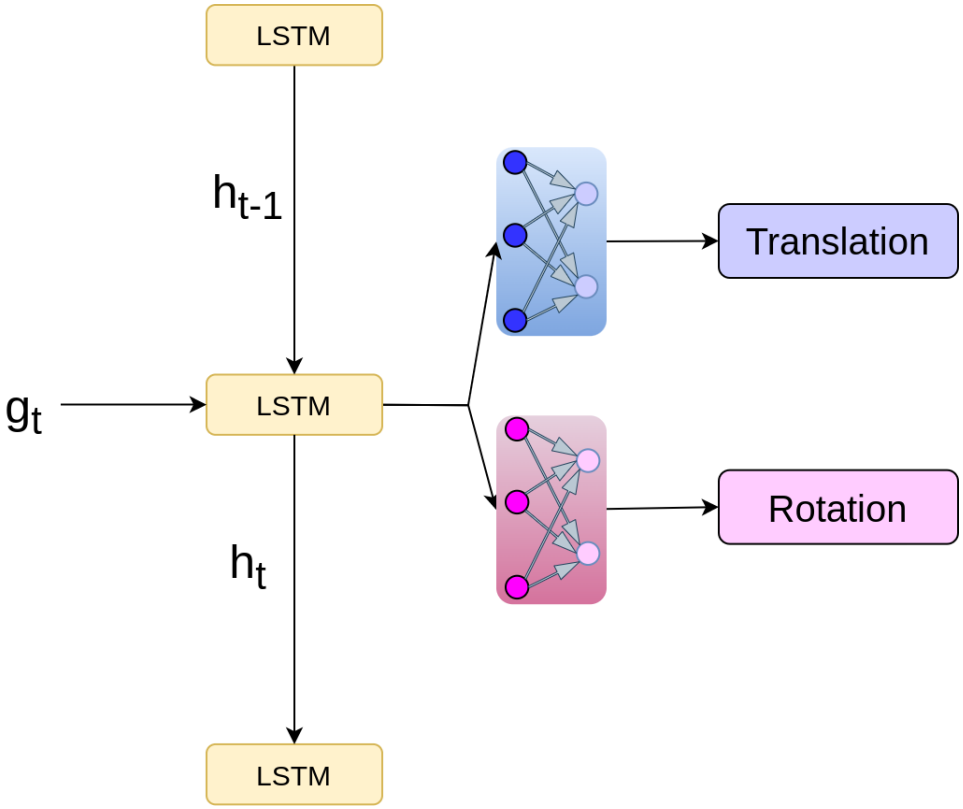

Figure 3.10: The pose estimation module

sequential information in both directions, forward and backward. Translation and rotation extraction sub-networks are built after the LSTM. The output of each sub-network is three output nodes corresponding to the three values of rotation or translation.

3.3.4 The loss function

The joint loss function for network training of this thesis as follow:

$$\mathcal{L} = w_{\mathcal{L}_{trans}} \mathcal{L}_{trans} + w_{\mathcal{L}_{trans}} \mathcal{L}_{rot} \tag{3.17}$$

In there, $\mathcal{L}_{trans}$ and $\mathcal{L}_{rot}$ are the loss of translation and rotation, and $w_{\mathcal{L}_{trans}}$ and $w_{\mathcal{L}_{rot}}$ are their weights, respectively. The mean squared error (MSE) is the loss function to calculate the dissimilarity between the estimated transition and the ground truths.

The translation loss function for each estimated between frame $t-1$ and t is:

$$\mathcal{L}_{trans} = \frac{1}{N}\sum_{i}^{N}(t_i - \widehat{t_i})^2 \tag{3.18}$$

Similar to the translation loss function, the rotation loss function is defined as:

$$\mathcal{L}_{rot} = \frac{1}{N}\sum_{i}^{N}(r_i - \widehat{r_i})^2 \tag{3.19}$$

where t_i and r_i are the ground truth value of translation and rotation. The $\widehat{t_i}$ and $\widehat{r_i}$ are the estimated value of translation and rotation. To scale normalization between translation and rotation value, $w_{\mathcal{L}_{trans}}$ is set 1 $w_{\mathcal{L}_{trans}}$ is set as 100.

Chapter 4

Experiments

4.1 Experimental setting

The system used to experiment has a single NVIDIA GeForce RTX 3090, 3.9 GHz CPU, and 24 GB RAM with Ubuntu 18.04 Operating system. The programming language is Python and is implemented based on the Pytorch framework. The ResNet50 pre-trained weight is used in the training phase. The batch size is 8, and the number of iterations is 5000/*epoch*. The learning rate is initially 0.00001 and decreases a half after every 10 epochs. Adam optimizer set the $\beta = 0.9$ for loss function optimization during training.

4.2 Dataset and Evaluation Metrics

The proposed method is evaluated using the KITTI dataset. The KITTI dataset is a popular dataset used to evaluate models related to autonomous vehicles. This dataset includes a lot of information based on the sensors placed on the vehicle (figure 4.1) This study uses the ground truth of RGB image sequences, 3D point clouds, and vehicle GPS. The sequences 00 to 08 are taken for the model training and sequences 09 and 10 for testing, similar to previous works. The evaluation metric followed the KIITI benchmark, including the mean square error for translation (measured in percent) and rotation (degrees).

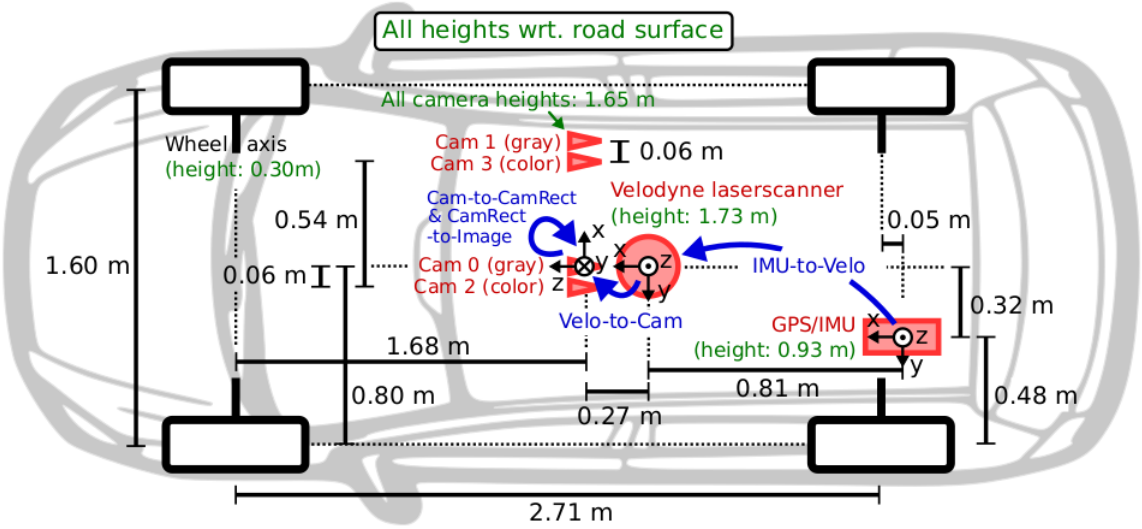

Figure 4.1: The sensor system is used to collect the KITTI dataset [7]

4.3 Data Augmentation

In deep learning, having more data helps the model learn more during training. In this work, the RGB and corresponding range images are flipped horizontally for data augmentation. Figure 4.2 shows the RGB and depth image and their flipping images, respectively. The left images are the original, and the flipping images are on the right.

Note that the camera's intrinsic matrix K is changed to (4.1) when flip the image with w as the weight of this image.

$$K^f = \begin{bmatrix} f_x & 0 & w - c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

In the training phase, training data include the 3D point cloud and image as input for the model and vehicle GPS for the label. The ground truth label needs to be changed for each flip sequence. Some original and flipping trajectories are shown in 4.3. The original image sequence $\langle I_t, I_{t+1} \rangle$ have the pose value $E = [t_x, t_y, t_z, r_x, r_y, r_z]$ corresponding. With flip image sequence $\left\langle I_t^f, I_{t+1}^f \right\rangle$, the flip pose value as the equation:

$$E^f = [t_x^f, t_y^f, t_z^f, r_x^f, r_y^f, r_z^f] = [-t_x, t_y, t_z, r_x, -r_y, -r_z] \tag{4.2}$$

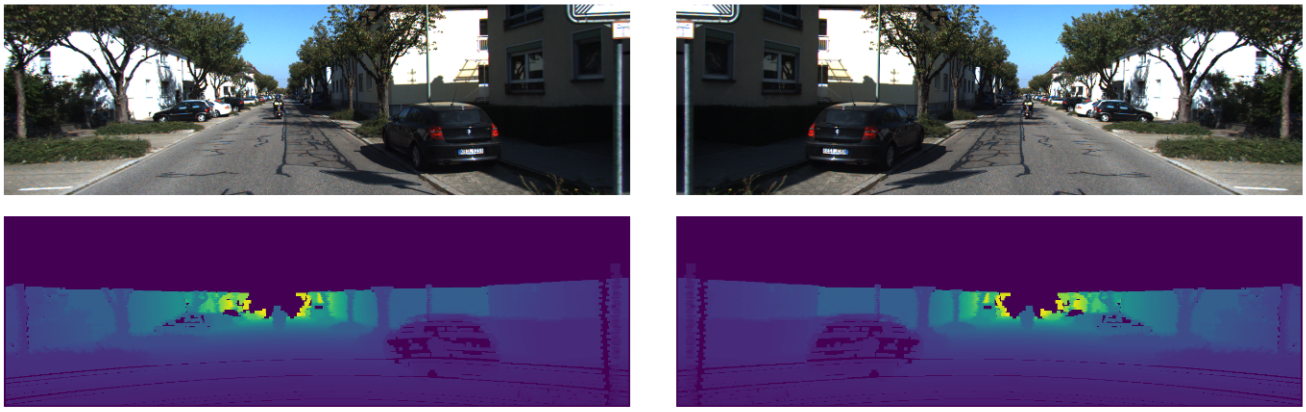

Figure 4.2: The results of data augmentation in RGB and range image

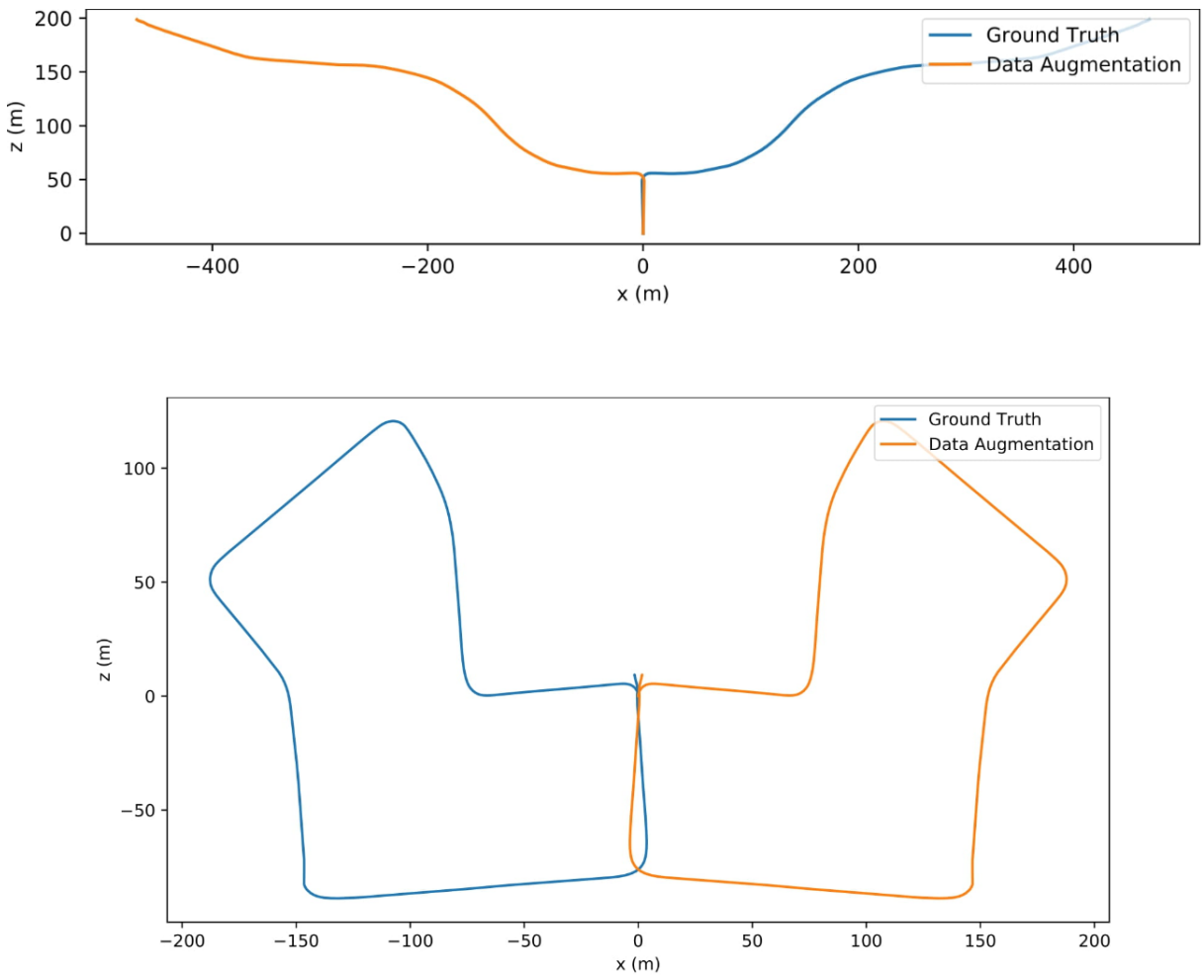

Figure 4.3: The flipping trajectories example

4.4 Experimental Results

The experimental results are shown in 4.1. t_{error} is the translational error and r_{error} is the rotational error. The lower, the better. The proposed method is compared with some recent learning-based methods for odometry. The proposed method gives a promising result compared to previous works. This thesis experiments with Unidirectional LSTM and Bidirectional LSTM models. Experimental results show that the error is reduced in the Bidirectional LSTM model. It shows the better effect of learning two-way information about the LSTM network. The proposed method's error rate is lower than most works. In sequence 09, the proposed method outperforms those produced by Zhan and SfMLearner. The translation error (t_{error}) and rotation error (r_{error}) of the proposed method are cut down to 10.04 and 2.41 in MSE, respectively. Although our translation error is higher than UnDeepVO, the rotation error of the proposed method is lower than UnDeepVO.

Table 4.1: Comparision of the proposed method with another learning-based method

Method	Sequence 09		Sequence 10	
	t_{error}	r_{error}	t_{error}	r_{error}
Deep-VO [8]			**8.11**	8.83
UnDeepVO [41]	**7.01**	3.16	10.63	4.65
SfMLearner [42]	18.77	3.21	14.33	**3.30**
Zhan et al. [43]	11.92	3.60	12.62	3.43
Proposed method (Unidirectional LSTM)	12.79	3.12	11.80	5.18
Proposed method (Bidirectional LSTM)	10.04	**2.41**	10.13	4.33

In sequence 09, the proposed method's rotation error is the lowest compared to previous methods. Translation error and rotation similarity along the vehicle's speed are shown in figure 4.4. The interesting point is that the higher the speed, the smaller the rotation error; the opposite is the translation error. Because of the higher vehicle speed, the rotation error is decreased in sequence 09.

Figure 4.6 shows each frame's translation value and rotation. The estimated values are in the same direction as the ground truth values, although the magnitudes are not exact. It proves that the attention-driven mechanism contributes significantly to the effectiveness

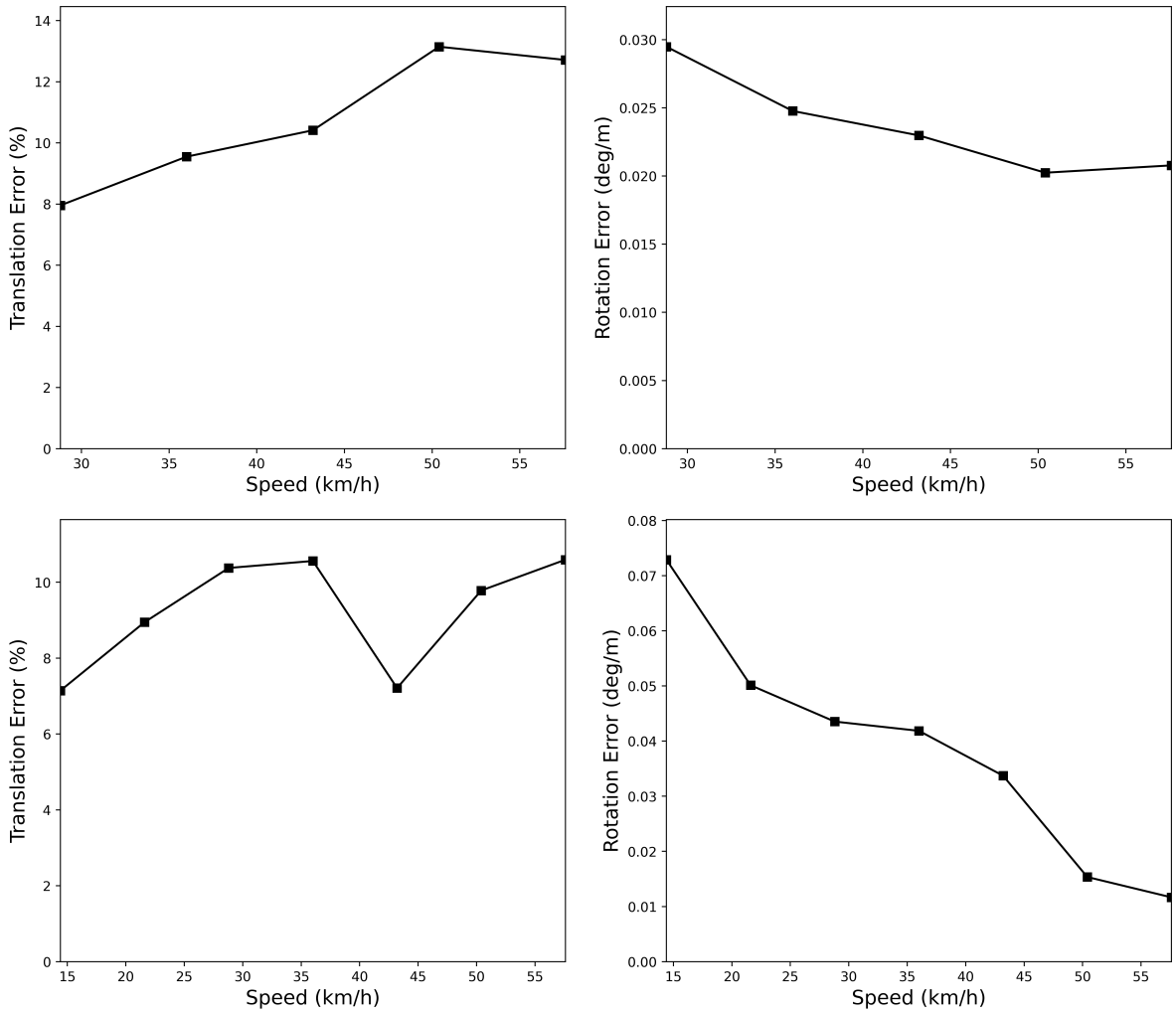

Figure 4.4: Translational and rotational errors on sequences 09 (top) and 10 (bottom) with different speed

of the proposed.

The estimated trajectory is visualized and compared with the ground truth trajectory. Figure 4.5 shows this visualization of sequences 09 and 10. The estimated values shown in figure 4.6 have a direction close to the GPS value. Therefore, the estimated trajectories' direction is almost the same as the ground truth. However, the experimental results of task mapping are not good because the error accumulates during the vehicle's moving. Because this thesis focus on odometry, the global optimization module has not been applied to improve the pose estimation. Therefore, the mapping estimations differ from the ground truth.

Figure 4.7 shows an example of the feature selection module. The attention masks are

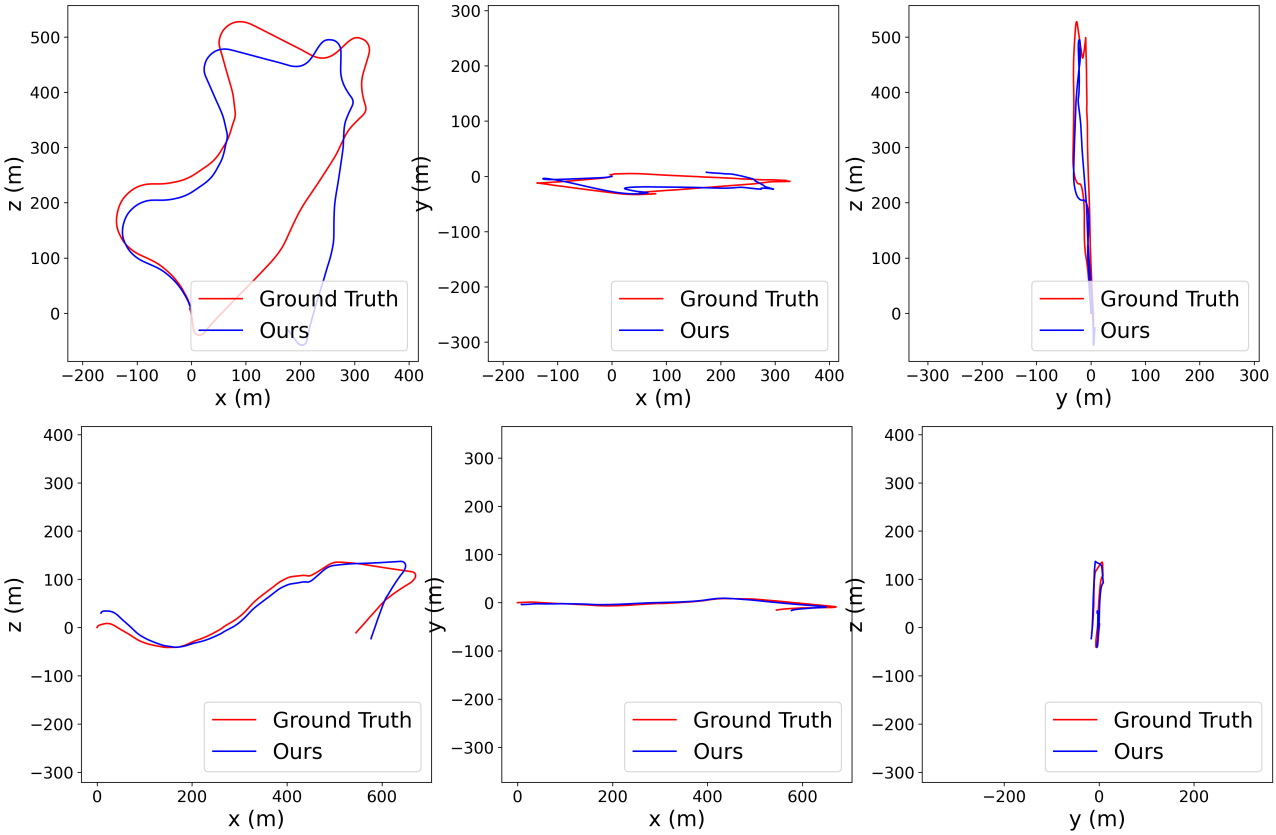

Figure 4.5: The trajectories of the vehicle in sequences 09 (top) and 10 (bottom) in xz-planes, xy-planes, and yz-planes

projected onto corresponding RGB images. A sequence of consecutive frames is shown in 4.7, in which a moving car appears in this frame. The attention-driven mechanism focus on static objects (the landmarks), as highlighted by the green masks. It does not focus on this moving car. This example demonstrates that the proposed method's self-attention mechanism has determined a static region in RGB images. However, only some static objects are essential regions. The essential region must provide more information for the system and robustness. Static objects that yield much essential information, such as traffic signs or roads. However, the proposed method also focuses on static regions but yields less information than roadside leaves. Based on this observation, some ideas can be used to improve the proposed method.

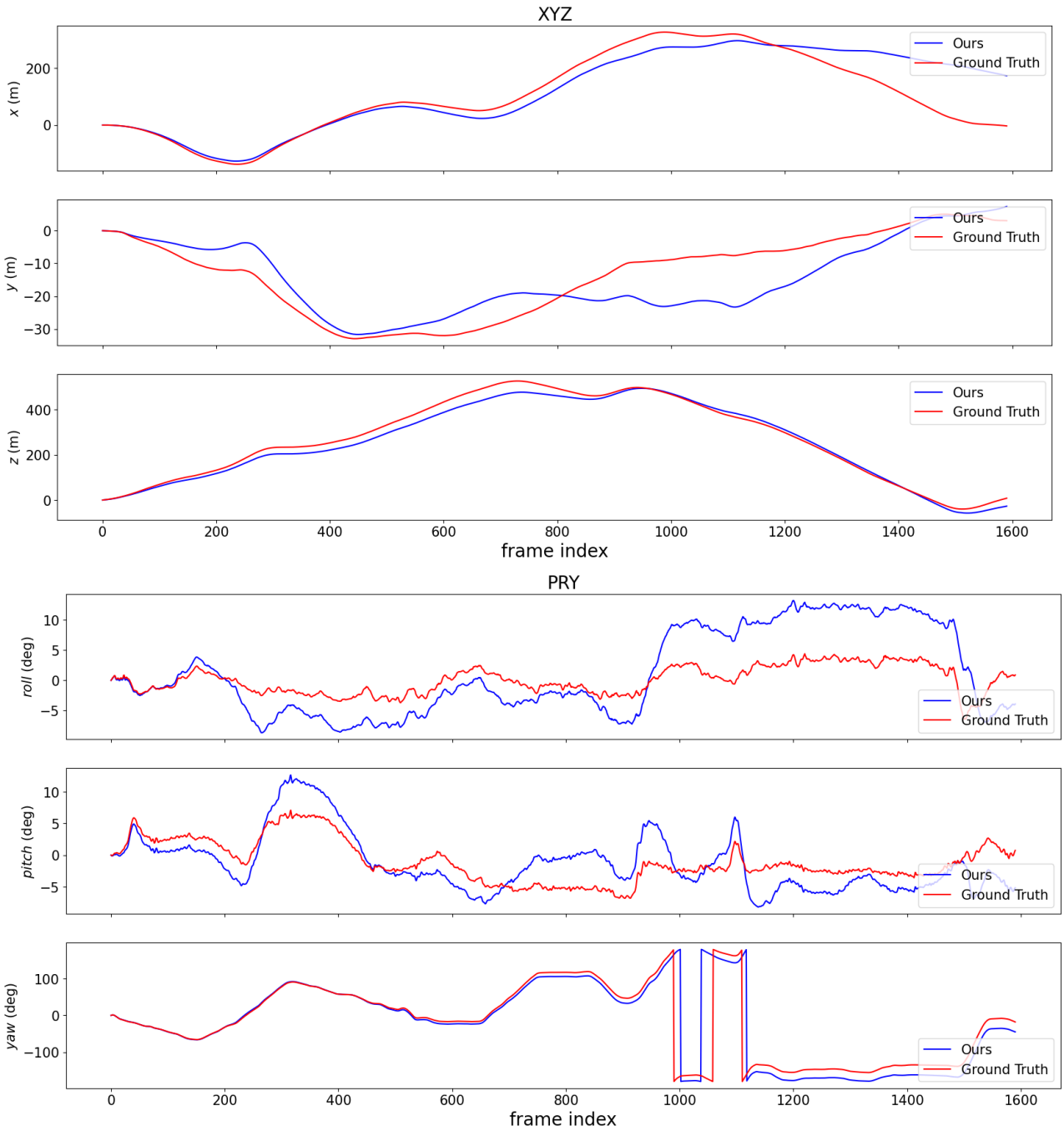

Figure 4.6: The translational and rotational value of estimation and ground truth in each frame of sequences 09

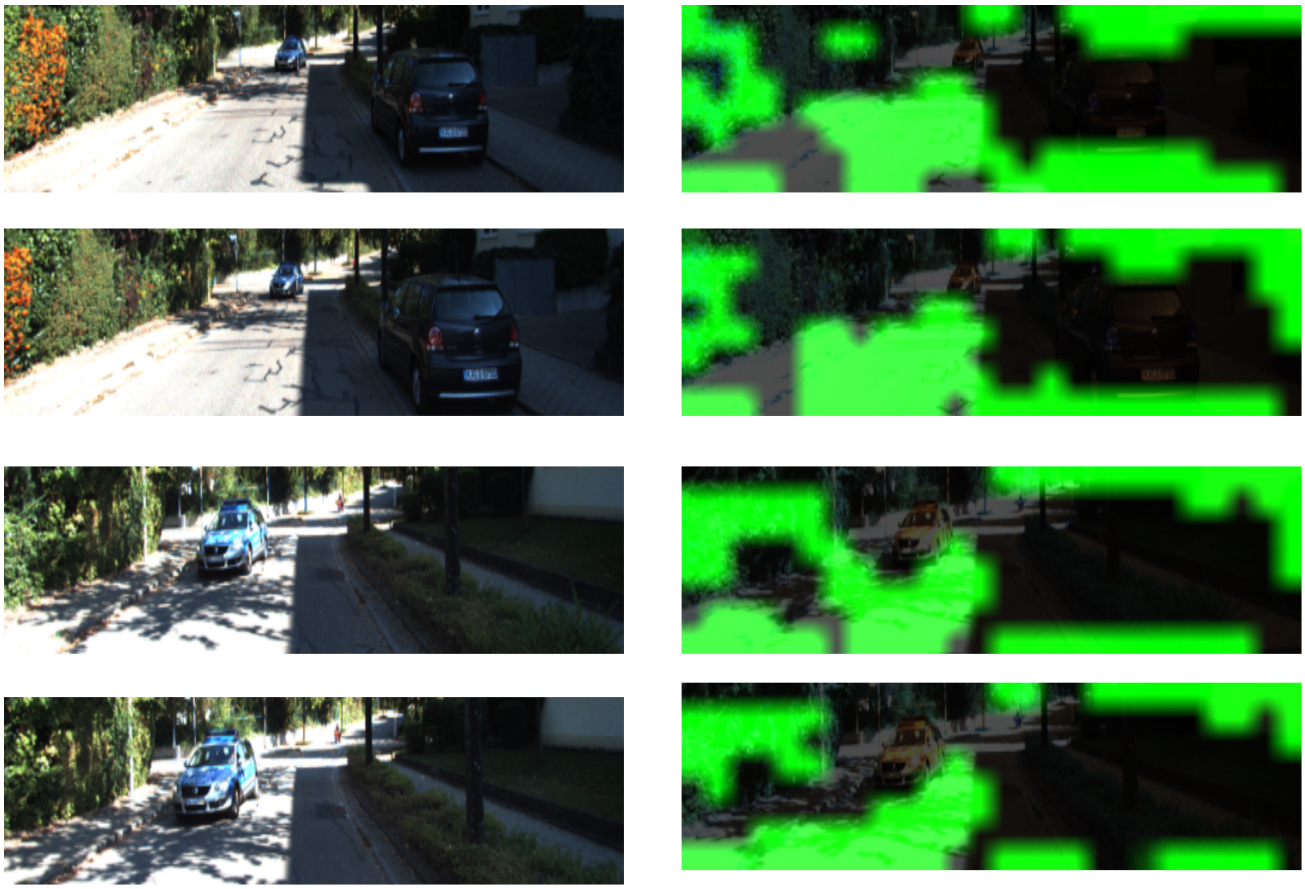

Figure 4.7: The result of attention mask in feature selection module

Chapter 5

Conclusions and Future Work

5.1 Conclusion

The research question of this thesis is: "How can we compensate for the depth and visual information for odometry?" Following this research question, the subquestions are answered:

- **What kind of representation for LiDAR information should be used?**
  LiDAR information, which has a large size, is a technical challenge for odometry. The 3D point cloud must project to another representation to reduce the system's computational cost. In this work, the point cloud is projected from 3D to 2D dimension based-on range images. Compared to 3D point clouds, 2D ranger images retain depth information but have a smaller size.

- **How do we extract the essential information in RGB images?**
  Essential regions include static regions in images. Moving objects such as persons or cars affect the pose estimation. An extraction module is proposed based on the deep learning method. By essential region mask generation by self-attention, this method extracts the essential region which does not include moving objects.

- **How to fuse information from the camera and LiDAR effectively?**
  After extracting information from the camera and LiDAR, depth and visual infor-

mation are fused to exploit their advantages. This thesis applies guided attention to information fusion. The attention mechanisms aim to intensify the relationship between entities. Therefore, the attention mechanism can find the correlation between depth features and visual information.

Odometry is an exciting topic in the field of computer vision. Because using information obtained from the sensor, the data collected is very important in solving the odometry problem. This thesis proposes an end-to-end model with data captured by sensors, and the output is the vehicle's position as input. Feature extraction is the focus of this thesis. Besides, this work fuse visual and depth information via an Attention-driven mechanism. The thesis has achieved a specific result, and there will be more development directions in the future.

5.2 Future Work

The proposed method can be improved in the future with some potential ideas as follows:

- **Essential region segmentation**

  The essential and informative regions for the estimation are the static object containers. However, not all regions containing static objects can be considered essential regions. Suppose the trees or the sky do not move through the frames, but it cannot be considered an essential region. Therefore, a classifier can be built to look for essential or non-essential regions. The question is how each feature class affects the experimental results using the object segmentation algorithms in the image. Then, this classifier determines whether the object belongs to the essential region or not. Finally, the method will remove non-essential regions to improve accuracy.

- **Feature discriminate**

  The assumption for the discriminative idea is that the closer a region is to an autonomous vehicle, the more informative it is. Depth information from LiDAR will help us know the distance from the vehicle to the surroundings. A region

distance condition will be applied. The closer the regions are to the vehicle, the higher the confidence level and vice versa. In other words, the weights will be added to the essential regions based on the depth information.

Bibliography

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[2] D. N. S. D. A. Salleh, "Study of vehicle localization optimization with visual odometry trajectory tracking," Ph.D. dissertation, Université Paris Saclay (COmUE), 2018.

[3] M. H. Mirabdollah and B. Mertsching, "Fast techniques for monocular visual odometry," in *German Conference on Pattern Recognition*. Springer, 2015, pp. 297–307.

[4] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[5] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "Lo-net: Deep real-time lidar odometry," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8465–8474.

[6] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Lidar-monocular visual odometry using point and line features," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1091–1097.

[7] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[8] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual

odometry with deep recurrent convolutional neural networks," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[9] Y. K. Moo, T. Eduard, L. Vincent, and F. Pascal, "Lift: Learned invariant feature transform," in *Computer Vision - ECCV*, 2016, pp. 467–483.

[10] A. J. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 1052–1067, 2007.

[11] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] J. Shi and Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *International Conference on Computer Vision*, 2011, pp. 2564–2571.

[15] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

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

[17] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[18] H. M. S. Bruno and E. L. Colombini, "Lift-slam: A deep-learning feature-based monocular visual slam method," *Neurocomputing*, vol. 455, pp. 97–110, 2021.

[19] A. Handa, M. Bloesch, V. Patraucean, S. Stent, J. McCormac, and A. J. Davison, "Gvnn: Neural network library for geometric computer vision," in *Computer Vision - ECCV Workshops*, 2016, pp. 67–821.

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

[21] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization." in *International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946.

[22] R. Li, S. Wang, and D. Gu, "Deepslam: A robust monocular slam system with unsupervised deep learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2021.

[23] Y. Lyu, L. Bai, and X. Huang, "Chipnet: Real-time lidar processing for drivable region segmentation on an fpga," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1769–1779, 2018.

[24] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[25] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2174–2181.

[26] Y. Cho, G. Kim, and A. Kim, "Deeplo: Geometry-aware deep lidar odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2145–2152.

[27] C. Xu, Z. Feng, Y. Chen, M. Wang, and T. Wei, "Featnet: large-scale fraud device detection by network representation learning with rich features," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 57–63.

[28] R. Li, D. Gu, Q. Liu, Z. Long, and H. Hu, "Semantic scene mapping with spatio-temporal deep neural network for robotic applications," *Cognitive Computation*, vol. 10, no. 2, pp. 260–271, 2018.

[29] L. Weixin, W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based lidar localization for autonomous driving," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6389–6398.

[30] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[31] D. Yin, Q. Zhang, J. Liu, X. Liang, Y. Wang, J. Maanpää, H. Ma, J. Hyyppä, and R. Chen, "Cae-lo: Lidar odometry leveraging fully unsupervised convolutional auto-encoder for interest point detection and feature description," *arXiv preprint arXiv:2001.01354*, 2020.

[32] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[33] J. Graeter, A. Wilczynski, and M. Lauer, "Limo: Lidar-monocular visual odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7872–7879.

[34] Y.-S. Shin, Y. S. Park, and A. Kim, "Dvl-slam: Sparse depth enhanced direct visual-lidar slam," *Autonomous Robots*, vol. 44, no. 2, pp. 115–130, 2020.

[35] Y. Seo and C.-C. Chou, "A tight coupling of vision-lidar measurements for an effective

odometry," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1118–1123.

[36] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *IEEE international conference on robotics and automation*. IEEE, 2012, pp. 3936–3943.

[37] 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.

[38] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5998–6008.

[40] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[41] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7286–7291.

[42] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858.

[43] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 340–349.