修 士 論 文

Mapping of Road Facilities and Information on High Definition Maps using Mobile Mapping System

（モービルマッピングシステムを用いた路側構造物と道路情報の高精度 3D マップへの自動マッピング）

指導教員　　上條俊介　准教授

東京大学大学院
情報理工学系研究科
電子情報学専攻

学籍番号・氏名 48-176452　張　越

Abstract

Autonomous driving has become a hot and exciting research field in recent years. We can benefit greater road safety, reduced congestion and more productivity from these highly automated technologies. There are several on-board sensors on autonomous vehicles which can be used for environmental perception. Since tasks like navigation, path planning cannot be achieved by only on-board sensors, High Definition (HD) Maps, which have high accuracy of object locations and important roadway attributes are also significant components for autonomous or assistant driving. And semantic information of traffic rules, which are important for motion planning of vehicles, should be also contained in HD maps. However, semantic traffic regulations are now putted into maps manually, not automatically in industry. Motivated by this, this research aims to find an automatic way generating HD map with the semantic information.

In this research, we extract positional and semantic information of traffic facilities and road markings in the traffic scene based on camera images and laser scanning point cloud data from mobile mapping system. And we integrate the extracted information into our high definition map towards the automated construction of high-definition maps with semantic traffic regulations.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

In this research, we studied how to construct the High Definition Map (HD Map) in an automatic way instead of current manual way for self-driving vehicles based on sensor fusion. In this chapter, the background, objective and structure of this thesis are discussed.

1.1. Background

As road transportation which unites the world by connecting big and small parts with each other, plays an important role in our everyday life and production, autonomous driving has become a hot and exciting research field in recent years. We can benefit greater road safety, reduced congestion and more productivity from these highly automated technologies.

The main components of the autonomous vehicle can be broadly categorized into hardware and software. Hardware splits broadly into sensors, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) technology. And software is composed of the following functional modules: perception, prediction, planning and control [1]. The perception task is to do detection and tracking with the objects of interest (e.g. vehicles) in the scene. The prediction module estimates the intentions and trajectories of all actors into the future. Planning is responsible for making safe and efficient decisions for the autonomous vehicle, while control outputs the commands which are necessary for the autonomous vehicle to execute such decision.

Sensors are indispensable components which allow the autonomous vehicle to get raw information about the surroundings. The main sensors in autonomous vehicles include camera, Light Detection and Ranging (LiDAR), radar, GPS (Global Positioning System) and Inertial Measurement Unit (IMUs). Since each of these sensors have their respective advantages and disadvantages, sensor fusion is often applied to reduce the uncertainty of data derived from disparate sources.

However, environment beyond cannot be apperceived by only on-board sensors. And tasks such as navigation, path planning cannot be achieved by only on-board sensors. So, in order to make reliable driving decisions, besides on-board sensors, maps, which are also essential components of the intelligent transportation systems, would be used for providing applications include self-localization, path planning, and perception to make the cars safer and more comfortable.

The definition of a regular map is a visual representation of an entire area or a part of an area, typically represented on a flat surface. Regular maps are not sufficient for autonomous vehicles. They do not have information about traffic lanes, traffic signs, and lights, position, and height of the curbs. Also, they are useless for localization since they are designed for humans, not for autonomous vehicles. These maps can have location errors up to a few meters. Therefore, maps with high accuracy of object locations and important roadway attributes which are necessary for vehicle positioning and control need to be made for autonomous or assistant driving. This kind of map is what we called High Definition Map (HD Map).

Moreover, dynamic information such as statues of traffic lights, road condition of the path can make vehicles safer and more reliable. The combination of dynamic information with our usual static map is called Dynamic Map (DM) [2]. DM which manages various data from vehicles and their surroundings distributed widely and provides their data for ITS applications, is one of the key technologies towards autonomous driving. DM can be used to provide vehicles with information such as positions and mobility of cars and pedestrians nearby, statues of traffic lights, and traffic jams or accidents on the road, helping not only autonomous driving but also assistance driving for self-localization, path planning and decision making. From permanent static data to dynamic data, DM consists of four types of data layers, as shown in Figure 1.1. The lowest layer is common map with permanent static data. The second layer adds transient static information (road markings, roadside infrastructures etc.) based on the first layer. Transient dynamic information like traffic congestion, signal phase and slippery road are included in the third layer. And the top layer, which includes highly dynamic data (information of vehicles and pedestrians), is the final goal towards DM.

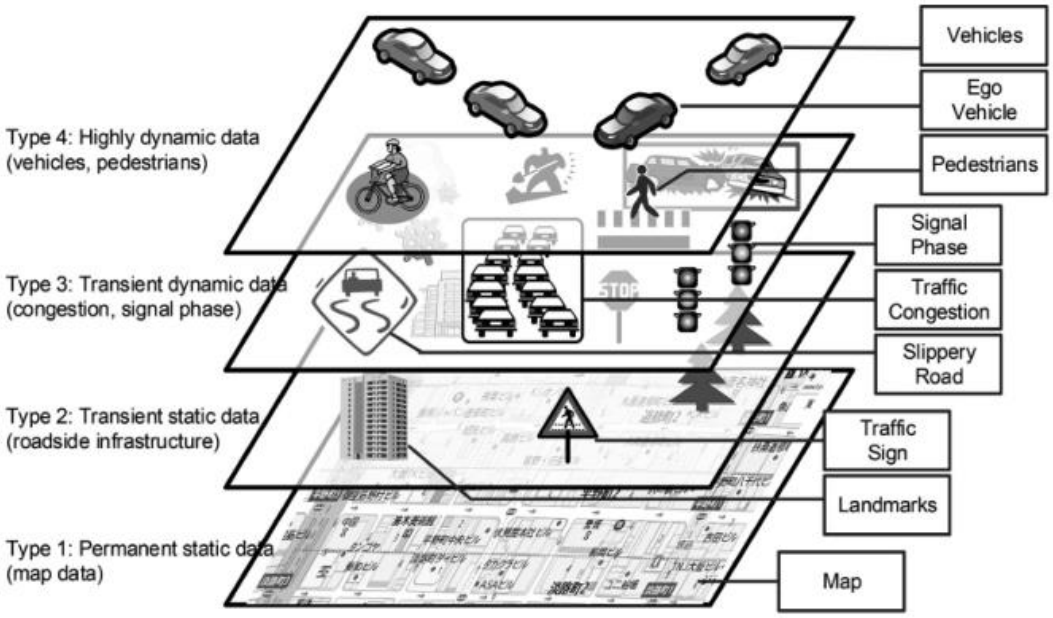


Figure 1.1. Hierarchy of Dynamic Map [2].

HD map is corresponding to the second layer of the DM. Since the construction of DM needs the highly developed Internet of Things (IoT) technologies. Current researches focus on construction of HD maps. HD maps enable vehicles to see beyond the driver's field of view, providing an accurate representation of the road ahead and information on the surrounding environment. And as an integral part of the system, High Definition (HD) Maps bring functions such as high-precision localization, environment perception, planning and decision making, and real-time navigation cloud services to autonomous vehicles. Typically, there are several roadway components can be included in the HD map. Lane lines can be included in the map as they allow for precise positioning of vehicle and other detected objects; 3D surrounding geometry can be included in the map for it delivering realistic modelling of surroundings and driving scenarios, and enabling slope and curvature values to be derived for Advanced Driver Assistance Systems (ADAS) applications; Including lane markings in the map would help ensure the vehicle adheres to the traffic rules and assist path planning; Adding traffic signs and traffic lights would provide semantic information of traffic rules for the vehicle; And Road borders and guardrails can be contained in the map to help find drivable areas of the road and deliver improved driving scenarios. Table 1.1 lists components usually contained in HD maps.

Table 1.1. Components and their Usage in HD Map

Components	Usage
3D Surrounding Geometry	Deliver realistic modelling of surroundings and driving scenarios, and enabling slope and curvature values to be derived for ADAS applications
Lane Lines	Allow for precise positioning of vehicle and other detected objects
Lane Markings	Help ensure the vehicle adheres to the traffic rules and assist path planning
Road Borders and Guardrails	Help find drivable areas of the road and deliver improved driving scenarios
Traffic Signs and Lights	Provide semantic information of traffic rules for the vehicle

Many self-driving car companies like Waymo and companies specialize in providing mapping services (e.g. HERE, TomTom, DeepMap, lvl5 etc.) create their own HD maps recently. Here, we will take HERE and TomTom's HD maps as examples to make an introduction of recent HD maps in industry.

The "HERE HD Live Map" [3] of the map company HERE, demonstrated in Figure 1.2, contains information of objects, lanes, signs, road geometry to help precise localization for vehicles. Besides, in order to deal with the dynamic and unpredictable thing of the roadway, HERE's map utilizes data from the cloud to compare current and historical map data. Map errors can be noticed if the vehicles use the map to report any disagreements between their perception and the map back to the mapping service. When a vehicle detects a difference in the road and the map, algorithms are set into motion to evaluate changing road conditions, and distribute those changes in near-real time to other cars on the road. This information may be not precise and trustworthy enough to update the map, but at least it can inform quality metadata and guide the mapping company to collect updated information in that location.

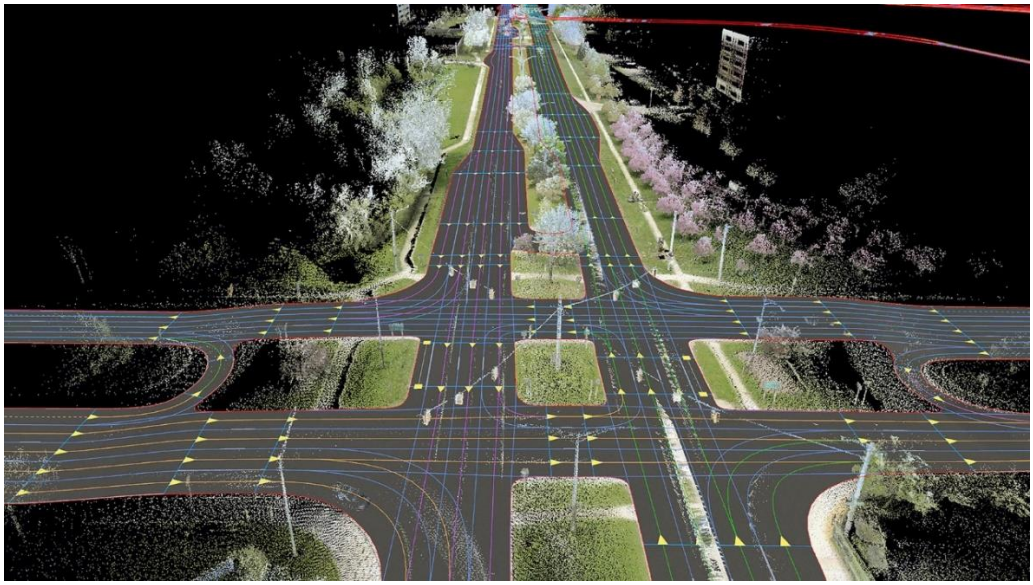


Figure 1.2. HERE HD Live Map [3].

The HD Map made by TomTom [4] takes all the important roadway attributes necessary for vehicle positioning and control. The content that TomTom provides helps enable Open Street Map (OSM) to accurately determine the current and upcoming precise location of the vehicle. TomTom's new concept "RoadDNA" allows for precise lateral and longitudinal positioning in a storage and processing friendly format while still using the power of a detailed LIDAR point cloud data, which assists with providing necessary content and libraries to work with in-vehicle hardware to efficiently determine a vehicles position. Figure 1.3 shows an example of detailed intersection data in TomTom's HD Map.

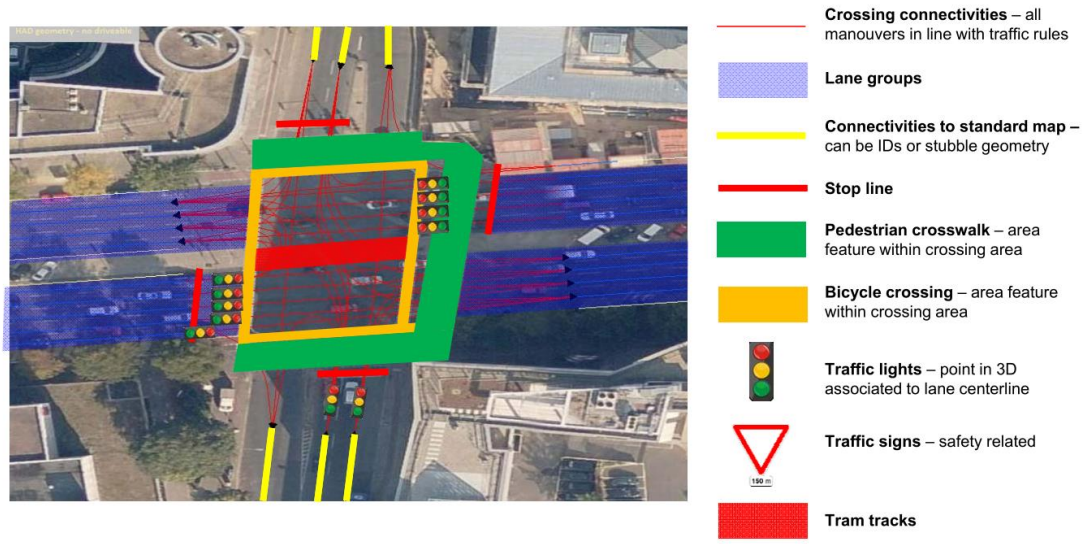


Figure 1.3. Detailed intersection data in TomTom's HD Map [4].

However, to the best of our knowledge, there are very little work on methods for constructing HD Maps automatically. Attempts to automatically generate lane-level precision maps for vehicles have been made in some researches. [5] presents a method that combines coarse, inaccurate prior maps from OSM with local sensor information from 3D laser scanner and a positioning system. They formulate a probabilistic model of lane structure using information from prior maps, and develop a number of tractable inference algorithms to leverage the coarse and structural information present in OSM, and integrates it with the highly accurate local sensor measurements. [6] solves the problem of inferring the position and relevant properties of lanes of urban roads with poor or absent horizontal signalization by deep neural networks (DNN) based segmentation approach. laser remission grid maps are segmented into road grid maps which contain all information about the road-lanes required for the operation of their autonomous car. However, information of semantic traffic regulations derived from traffic facilities (traffic lights, traffic signs) and road markings are not in their lane-level precision maps. As for industrial HD Maps, semantic traffic regulations are putted into maps manually, not automatically. Motivated by this, this research aims to find an automatic way generating HD map with the semantic information.

1.2. Objective

In this research, we propose a system towards the automated construction of high-definition maps with semantic traffic regulations. Figure 1.4 demonstrates the flowchart of proposed system. There are mainly three works be done in this thesis.

1. Firstly, we utilize point cloud data derived by mobile mapping system to extract road facility poles and road markings, from which we can know the accurate position of them in the map.
2. Since images can provide semantic meanings of these traffic facilities, we apply LaneNet to segment road lines and YOLOv3 to detect traffic lights, traffic signs and other road markings from images.
3. Finally, we apply coordinate transforming to integrate the information from both point cloud and images into our high definition map.

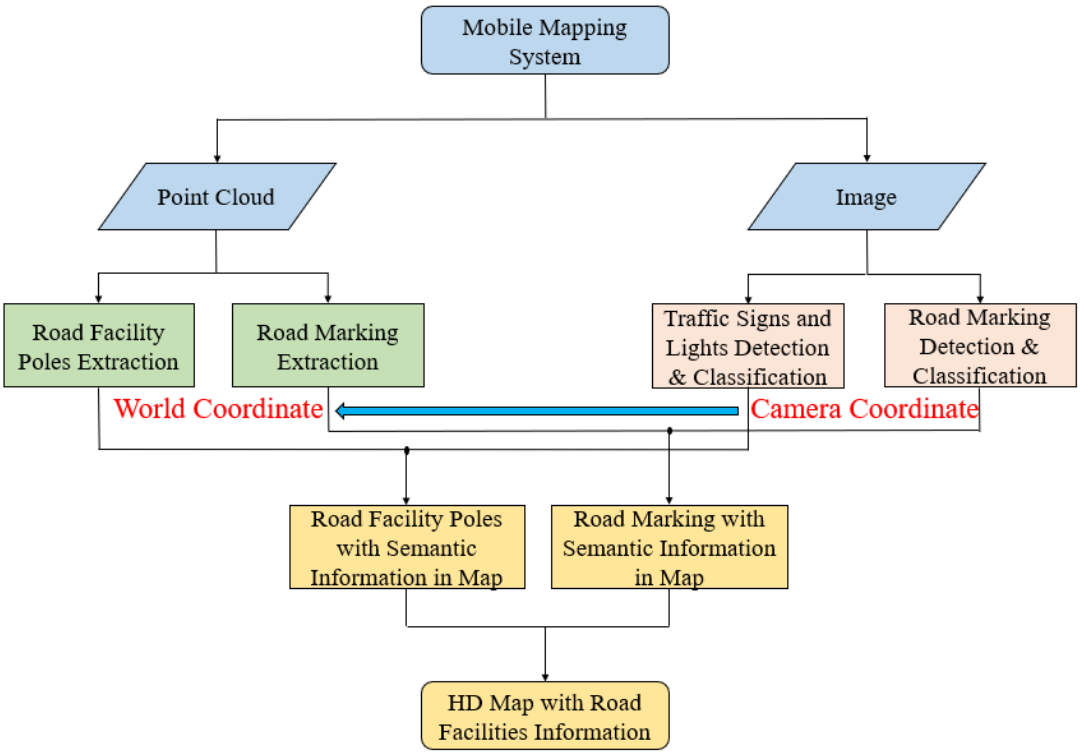


Figure 1.4. Flowchart of proposed system.

1.3. Thesis Structure

The Structure of this paper listed as follows:

In Chapter 2, the basic knowledge in the mapping process and the application of maps will be given.

In Chapter 3, we will make an introduction of Convolutional Neural Network.

In Chapter 4, we will discuss methods of road facilities extraction from laser scanning data

In Chapter 5, the way to understand semantic information of road facilities by image will be presented.

In Chapter 6, we will give the integration of information from laser scanners and images, and demonstrate the result of our map.

In Chapter 7, we will draw the conclusion of this thesis and talk about future work.

Chapter 2.

Foundation and Application of Mapping

From the previous chapter we know that regular maps are not sufficient for autonomous vehicles. In this chapter, we will introduce the foundation and application of mapping for autonomous vehicles. Components of the mapping system and detailed principles about the core component of the mapping system are introduced firstly. Then, we discuss the process of constructing 3D maps. Finally, applications of these created maps are described.

2.1. Foundation of Mobile Mapping System

In this section, we will give an introduction of the foundation of Mobile Mapping System (MMS). We will introduce the components of MMS firstly. As the positioning system and laser scanners are the core part of MMS in mapping, their principles of working will be described in detail.

2.1.1. Components of Mobile Mapping System

The MMS is usually mounted on a car, van or other vehicles that can move with traffic speed over roads and highways for the measurement of 3D coordinate data and the acquisition of sequence of images of the road and its surroundings. An MMS usually consists of a positioning and orientation system (POS), an Inertial Measurement Unit (IMU), an odometer system, one or more laser scanners, one or more digital cameras and a control unit. The part of the MMS without cameras is called Mobile Laser Scanning (MLS) system. The POS continuously acquires data for calculating the exterior orientation parameters (three coordinates and three attitude angles) using a Global Navigation Satellite System (GNSS) receiver and the IMU, often complemented with a wheel rotation counter. The odometer system captures even the minutest vehicle movement. In locations which Global Positioning System (GPS) measurements may be blocked, such as in tunnels, this equipment ensures that measurement accuracy is

maintained. Each laser scanner emits pulses, presently up to one million pulses per second, to capture road surfaces and objects above and alongside the road. Images acquired from cameras enables efficient mapping by compensating for the visual shortcomings of laser point clouds when the angle of the laser scanner and camera is adjusted by rigorous calibration, and calculation is conducted by precisely overlapping laser point clouds with images. And the control unit to which all devices are connected and from which they are steered eases life for the operator during data capture.

Figure 2.1 [7] shows the Mitsubishi Electric's MMS. We can see this system includes a unit with three GPS antennas, an IMU, cameras and laser scanners are mounted on the roof, and the odometer system is mounted in the wheel.

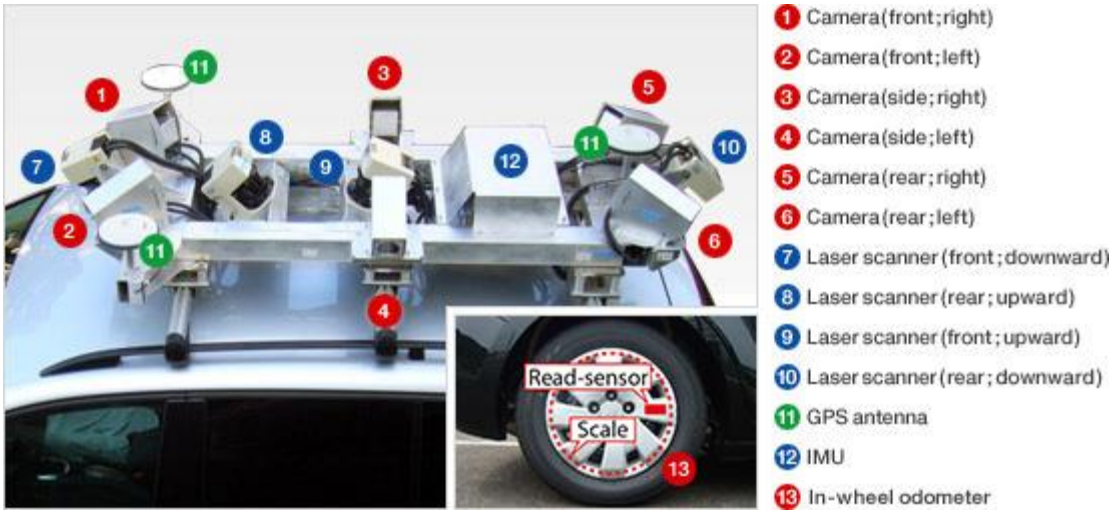


Figure 2.1. Mobile Mapping System developed by Mitsubishi Electric [7].

2.1.2. GPS Positioning

GPS is originally designed to provide position, speed and time information since it was initially developed by America in 1973. Nowadays there has been totally 31 GPS satellites launched into the space and they have almost covered the whole planet as shown in Figure 2.2 [8].

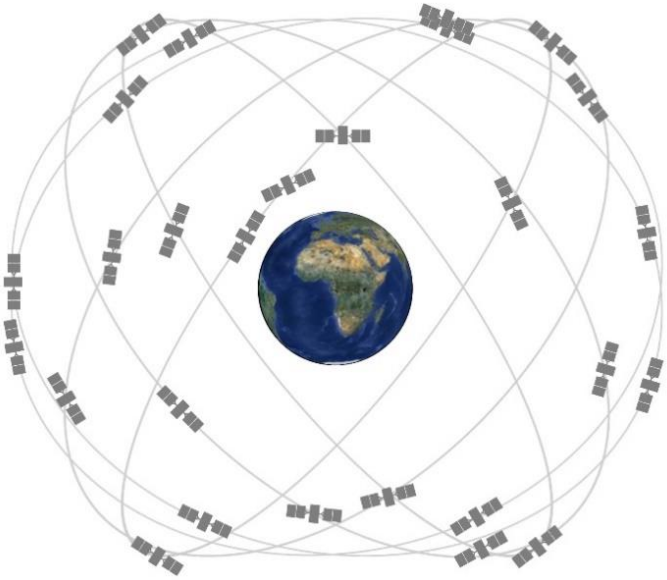


Figure 2.2. The constellation of GPS satellites [8].

Thanks to the high reliability and global coverage, GPS has been opened to public and now GPS is the mainly source for position information worldwide and basically every airplane, vehicle and mobile device has a GPS receiver built inside to provide customers with position information. Various kinds of researches have been done in this field as well.

Beside American GPS, other countries also launched their own satellites navigation system and all of these satellites has formed the modern GNSS. For example, Russia has GLONESS, Europe has Galileo, China has BeiDou and Japan has QZSS. Before the year of 2020, all the navigation satellite systems will be fully developed and the overall satellites in the orbit will be around 80, which means the position information will be much easier to get and more accurate in the future. Most of these satellite systems have the same fundamental positioning algorithm and we will mainly introduce the fundamental algorithm of GPS in this section.

GPS is composed of the space segments include satellites and satellite launchers, the control segments include several monitor stations around the world and user segments. To positioning the users, GPS fallow the law of trilateration [9] as shown in Figure 2.3.

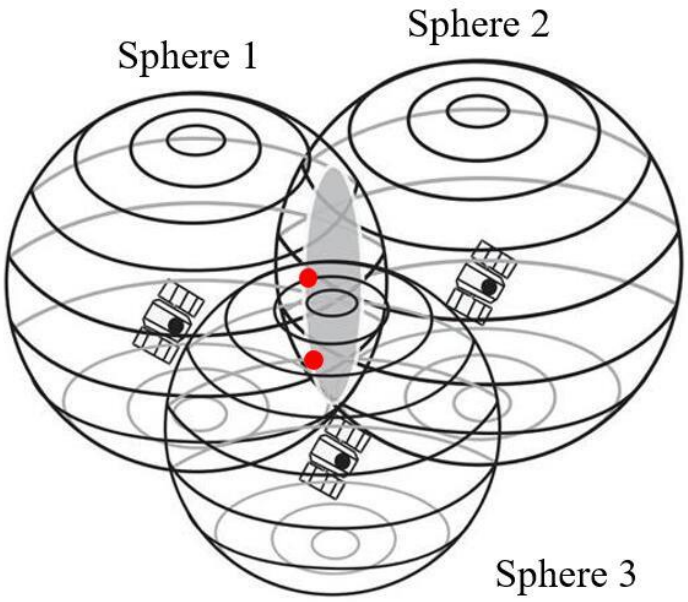


Figure 2.3. GPS trilateration positioning [9].

When a GPS receiver can receive the signals from three GPS satellites, if we can know the coordinates of these satellites, and the distances form each satellite to the receiver, then we can draw three spheres as shown in Figure 2.3. The intersection of sphere 1 and sphere 2 can be found as a circle as shown in the gray. And there will be two intersects from sphere 3 and this intersection circle as shown in the 2 red points. Usually, one of the red points should be near the ground and another is in the space. So only one of them is reasonable and the position of the user can be determined.

Therefore, in the ideal case, a reliable positioning result can be provided by using three GPS satellites. However, in the real world, at least four satellites are needed to calculate the user positioning result. The distances from the satellites to the user are defined as "pseudo range" and is computed by using the propagation time calculated based on the time the signal is sent and the time the signal is received. In order to get the time, clocks are built inside of the receiver. Because the clock in the GPS is not synchronized with the clock in the receiver, there will be a time shift and it will bring errors to the measured distance. Therefore, beside the altitude, longitude and latitude of the user, the clock bias becomes the fourth unknown parameter, and at least four equations are needed to solve four identically unknown parameters. This is why at least four satellites are required to calculate the user positioning result.

2.1.3. Laser Scanning and Prior Maps

Lidar (Light Detection And Ranging) is a form of laser scanning. It's similar to radar (Radio Detection And Ranging) but uses light instead of radio waves to detect objects and their distances. Figure 2.4 displays the HDL-64E Lidar made by Velodyne. the Lidar sensors measure precisely the road and its surroundings, including curbs, drainage channels, even potholes — to within millimeters.

Figure 2.4. HDL-64E Lidar made by Velodyne [10].

Lidar emits light in the form of a pulsed laser to measure the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to generate precise 3-D representations of the target. It primarily measures: (1) range, which means distance from the sensor to the first surface hit by the laser pulse (2) scan angle and (3) intensity, which is the reflective luminance of each point, represented as a digital number in the range from 0 to 255. Combining these measurements with those from a GNSS receiver, IMU and wheel counter provides 3D coordinates of millions or even billions of points in a local or national reference system. Usually the sensors are integrated on one rigid platform of which the mutual offsets have to be calibrated [11]. This is usually done by the manufacturer. Vibrations during the survey, shocks due to holes in the pavement and sudden slowdowns will cause mutual displacements of the sensors and other disturbances. To warrant high-precision surveys all the time regular recalibration is required.

In addition to the 3D coordinates, one or more attributes may be assigned to each point. The attributes may be directly measured by the sensor, this relates particularly to the intensity of the return, computed from a neighborhood of points or obtained from

other sources. RGB values, which may act as features for automatic object recognition. may be assigned to laser points using the simultaneously recorded digital images or image sources having other time stamps. Attributes of collected points create point clouds data, which is assembled into “prior maps”. A prior map is a rich, 3-D map of the road which can be used to help autonomous vehicles know precisely where it is at any time. An example of raw data collected by HDL-64E Lidar is shown in Figure 2.5.

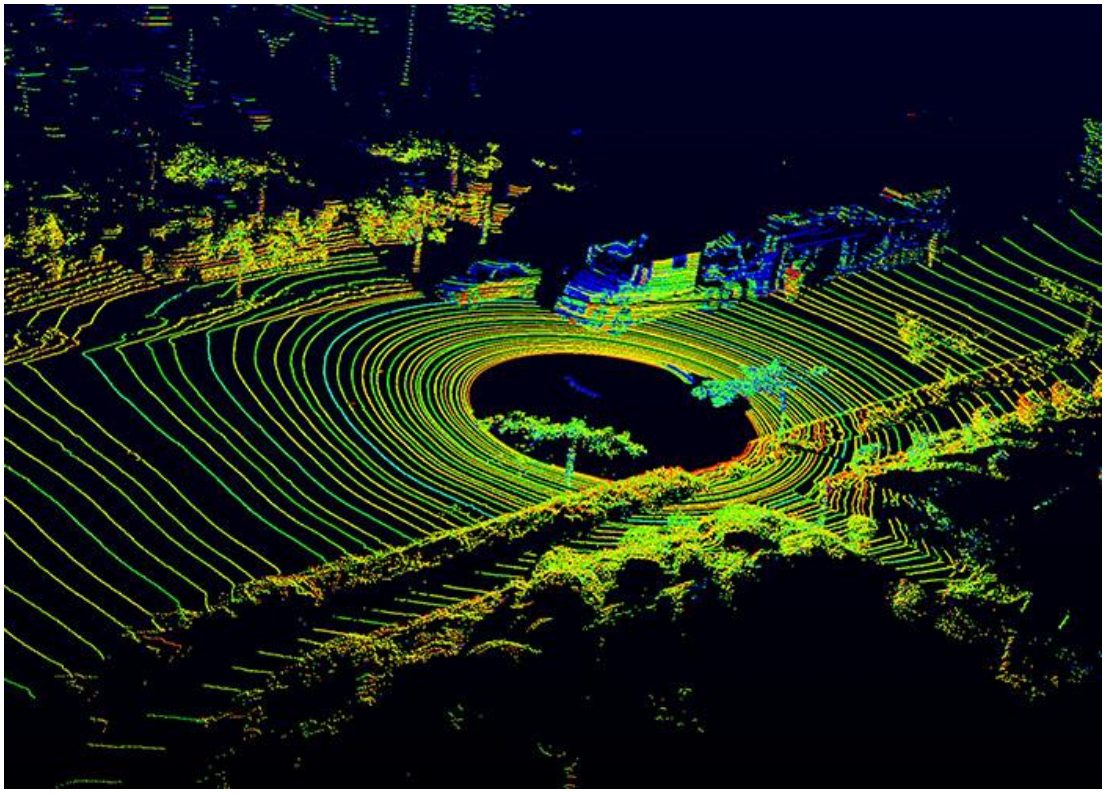


Figure 2.5. Example of raw data collected by HDL-64E Lidar [10].

For area within the measurement range of the scanner, collected point cloud data can be used as the raw map directly. However, if the working area is larger than the scanner’s measurement range, multiple stations are required, from where the separate point clouds are collected, then these clouds have to be unified. This can be solved by three kinds of methods [11]: manually select natural points being identifiable in multiple point clouds; using artificial markers (e.g. spheres) during scanning, and applying automatic or manual marker selection; or computationally intensive technologies for automatic point cloud matching.

2.2. 3D Map Construction by MMS

The main process of construction of 3D maps for autonomous vehicles is shown in Figure 2.6. In the pipeline, Ground Control Points (GCPs), which are acquired by actual measurement, are marked points on the ground used for the adjustment of MMS measurement data.

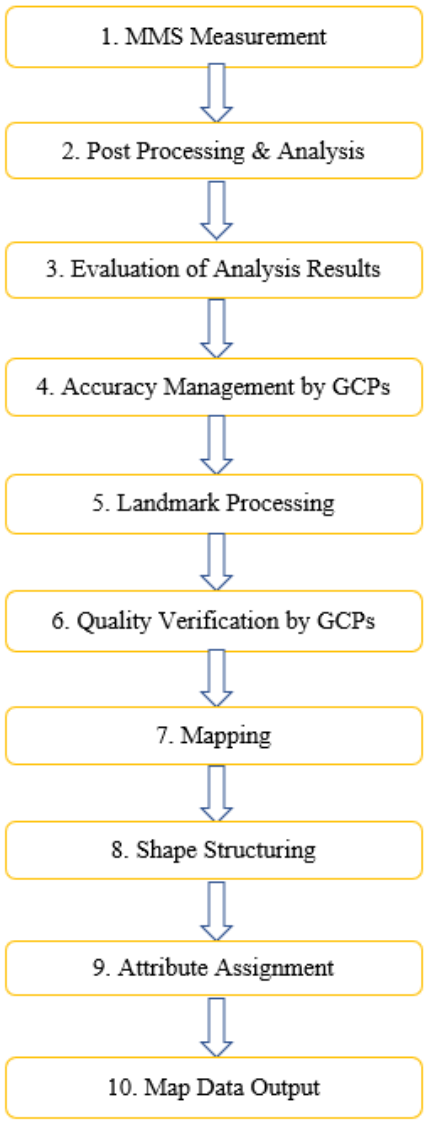


Figure 2.6. Pipeline of 3D map construction.

We can see there are ten steps in total. We will make a brief introduction of these steps. The first step is the MMS measurement which we have discussed in the previous section. The second step is to analysis the position and orientation of the vehicle by

obtained MMS data. And according to the calibrated vehicle's position and orientation, position and orientation of laser scanners and cameras can be calculated. The third and fourth step do evaluation of obtained data based on GCPs within limits of accuracy. When significant deviation occurs comparing to the GCPs data, step five needs to be done to improve the accuracy based on the GCPs data. In the sixth step, if there are GCPs that were not used for landmark processing, then using these GCPs to verify the accuracy. Based on the verified data, points, lines, and planes of prescribed objects or structures should be obtained in step seven. Relationships among the obtained shapes should be given in the eighth step. In the ninth step, we annotate attributes of the prescribed objects or structures using information derived from on-board camera and field survey. And the output map data with high accuracy of object locations and important roadway attributes, in prescribed format for autonomous vehicles, can be acquired finally.

2.3. Applications of 3D Maps

Various applications can be done based on the created 3D maps. In this section, we will make a brief introduction of some examples about applications made by the constructed 3D maps.

Vehicle self-localization is one of the significant applications. The main vehicle-based positioning technologies, such as GNSS, 3D Lidar, and vision-based systems, can be assisted by access to a prior map. A 3D building map helps GNSS address the signal blockage and reflection problems caused by tall buildings in urban areas [12]. Another well studied positioning technique for urban area is map matching, which can be done based on Lidar [13] or vision sensors [14]. [13] proposed a localization method based on self-adaptive multi-layered scan matching and road line segment matching to adapt the situation changes among different heights. This paper effectively matches the features observed from different heights and improves the results by applying the line segment matching in certain scenes. [14] used image data and HD map data to mitigate the lateral error for lane level localization. In this paper, the image content of a camera mounted on the platform is utilized to detect the road boundaries in the image. They apply color masks to detect the road marks, do the Hough transform to fit lines to the left and right road boundaries, find the corresponding road segment in the database, estimate the homography transformation between the global and image coordinates of

the road boundaries, and obtain the camera pose with respect to the global coordinate system.

There are many other applications based on constructed 3D maps. 3D city modeling is one of the applications. [15] investigated methods for 3D building modelling in complex urban scenarios with the support of oblique. The terrestrial map data collected by MMS is used to complement building models generated by airborne data. Other applications include lane-level integrity provision for navigation and map matching with GNSS, dead reckoning, and enhanced maps [16], 3D city visualizations and metrology [17], road asset inventories [18], urban forest inventories [19], and lane departure warnings [20] for vehicles. It would take too much space for us to present a comprehensive discussion. If you are interested in these topics, please refer to the original paper. We will talk more about pole-like objects detection and classification based on point cloud map in Chapter 4.

Chapter 3.

Foundation of Convolutional Neural Network

In this chapter, we will explain the foundation of Convolutional Neural Network (CNN). We will make a brief introduction of CNN in Section 3.1. In Section 3.2, we will discuss several basic components of CNN including 4 kinds of layers and the non-linear activation function. Then, training methods of CNN will be talked about in Section 3.3. Finally, in Section 3.4, we will introduce several most famous and important architectures which are LeNet, AlexNet, VGGNet, GoogleNet, ResNet, and DenseNet in chronological order.

3.1. Origin of Convolutional Neural Network

It is estimated that the average human brain contains 86 billion neurons [21]. Together they form a huge network. Inspired by biological nervous systems, neural network is a mathematical algorithm that simulates the behavioral characteristics of human brain neural network and performs distributed parallel information processing. Even though we have not been able to replicate the brain so far, the field of artificial intelligence offers very effective solutions to many problems by simulating the observations of biological research of various nervous systems.

Generally, neural network consists of a set of artificial neurons. Formally, an artificial neural has n inputs represented as a vector $\vec{x} \in R^n$. Inputs in an artificial neuron correspond to the dendrites in a biological neuron, while a single output of an artificial neuron corresponds to the axon in a biological neuron, which is depicted in Figure 3.1. Each input i, $1 \leq i \leq n$, has an assigned weight $w_1, w_2, \ldots, w_n$, and bias $b_1, b_2, \ldots, b_n$. Weighted input values are combined and followed by a non-linear activation, as shown in Figure 3.2.

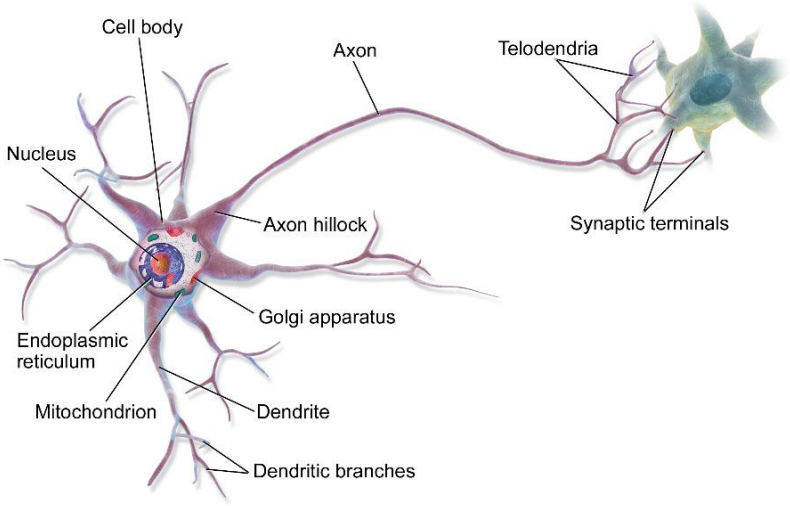


Figure 3.1. Model of biological neuron [21].

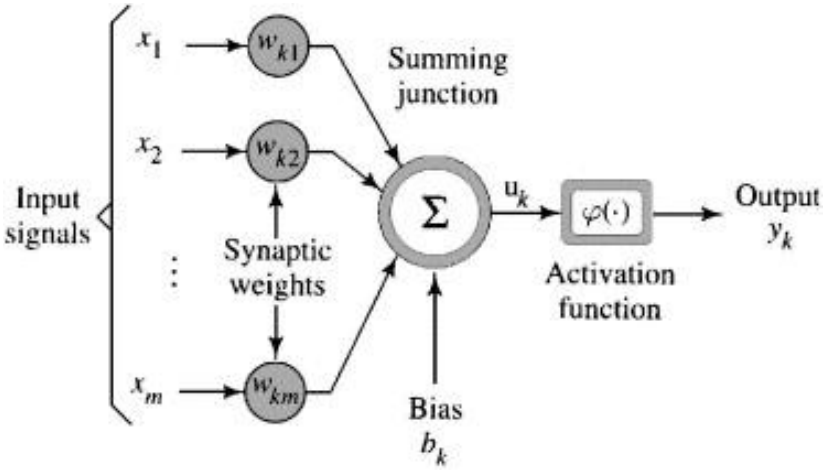


Figure 3.2. Model of an artificial neuron.

In 1959 Hubel & Wiesel [22] inserted a thin metal electrode into the cat's visual cortex to see how a single neuron in the cat's brain responded to the image on the screen. They found that neurons in the frontal area of the visual system reacted strongly to specific light signals, but did not respond to any other pattern at all. This part of visual system was called the Primary Visual Cortex (V1). Figure 3.3 [23] explains the cognitive process of the human visual system. Information flows from the retina to V1 and then to the Secondary Visual Cortex (V2), next is V4 and last is IT (Inferior Temporal Gyrus). The visual system is hierarchical and progressive, with each layer handling a higher level of information than the previous layer.

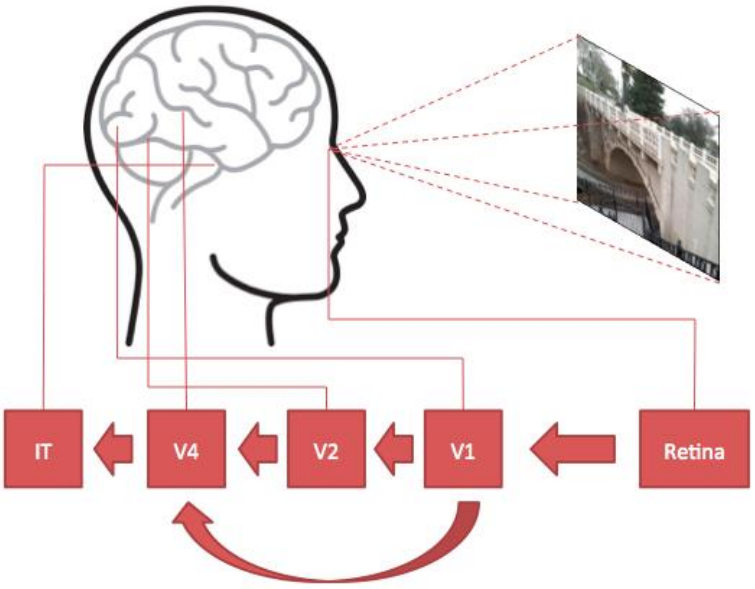


Figure 3.3. A brief illustration of ventral stream of the visual cortex in human vision system [23].

The characteristics of V1 layer has inspired the structure of Convolutional Neural Network (CNN). Since V1 layer has a function that it is able to transform 3D information coming from outside into 2D information, the layers in a CNN are based on a two-dimensional spatial structure. There are many simple cells in V1 layer with local receptive field to detect low-level features, which inspired the design of convolution kernels (filters) in CNN. There are also many complex cells in V1 layer that are used to respond to the low-level feature detection of simple cells, which inspired the design of a pooling layer in a CNN. The visual layers behind V1 have the same principle as V1. By stacking these layers, features detection and pooling strategy repeatedly executed. Finally, a human visual system formed. Similarly, the architecture of CNN is also a repetitive overlay of convolutional and pooling layers, forming a deep hierarchy and extracting high-level features.

3.2. Basic Components of Convolutional Neural Network

Even though there are many different architectures for CNN in the literature, the majority of them can be built by stacking four main type of layers in different combinations. Namely, the fully connected layer, the convolutional layer, pooling layer and batch normalization layer. And the activation function, which is the non-linear

factor, enables CNN to approximate any function. In this section, we will explain these components.

3.2.1. Convolutional Layer

Convolution layers are an important layer in CNN. The purpose of convolutional layers is to detect the features in the presented images. It consists of multiple feature maps, each recognizing certain specific feature. The feature recognition can be thought of as running the sub-image through a filter. The filtering is essentially done through weight adjustments. Regular Neural Networks, only made of linear and activation layers, do not scale well to full images. For instance, images of size 3 × 224 × 224 (3 color channels, 224 width, 224 height) would necessitate a first linear layer having 3 × 224 × 224 + 1 = 150 parameters for a single neuron (e.g. output). However, convolution layers take advantage of the fact that their input (e.g. images or feature maps) exhibits many spatial relationships. In fact, neighboring pixels should not be affected by their location within the image. Thus, a convolutional layer learns a set of N_k filters $F = f_1, f_2, \ldots, f_{N_k}$, which are convolved spatially with input image x, to produce a set of N_k 2D features maps z:

$$z_k = f_k \otimes x \tag{3.1}$$

where $\otimes$ is the convolution operator. When the filter correlates well with a region of the input image, the response in the corresponding feature map location is strong. Unlike conventional linear layer, weights are shared over the entire image reducing the number of parameters per response and equivariance is learned (i.e. an object shifted in the input image will simply shift the corresponding responses in a similar way). Figure 3.4 shows how a convolution operation works.

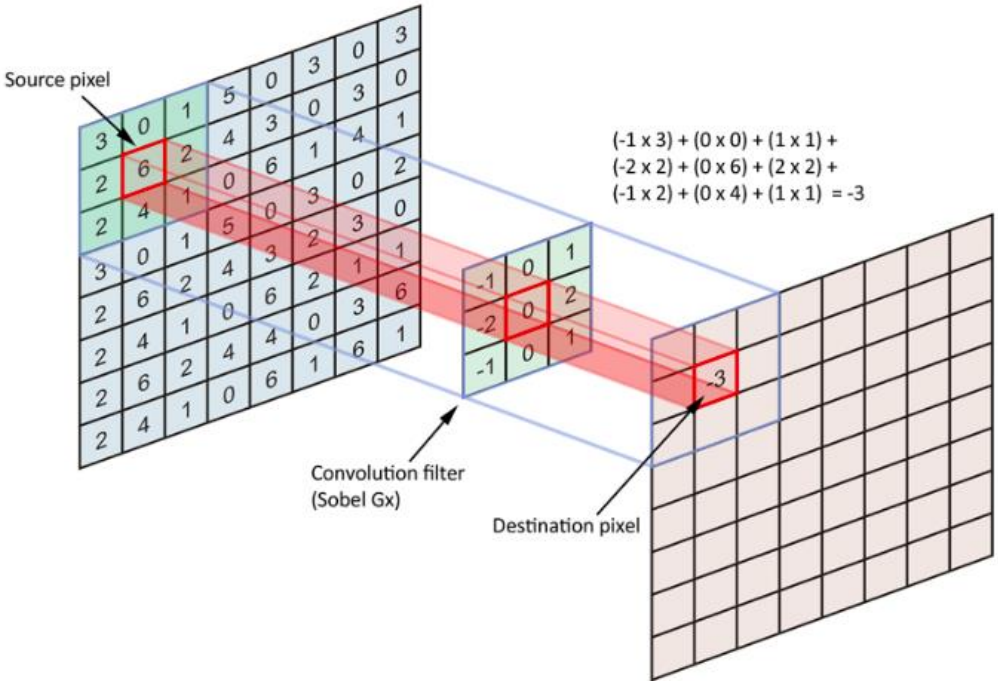


Figure 3.4. The work flow of a convolution operation.

3.2.2. Pooling Layer

Convolution layer can extract the features of images. If these features are directly using for image classification, GPU will face a huge challenge due to the amount of computation and the model will have a high probability to be overfitting. The pooling layer, which is a form of non-linear down-sampling, serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. There are two main kinds of pooling layers which are usually used:

Max pooling layer: Selecting the largest value in the pooling window as the sample value.

Mean pooling layer: Averaging all the values in the pooling window and this average value is chosen as sample value.

Figure 3.5 shows how max pooling layer and mean pooling layer works.

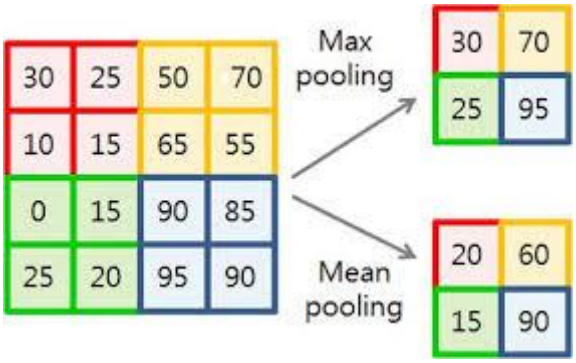


Figure 3.5. Example of max pooling and mean pooling.

3.2.3. Batch Normalization Layer

Batch normalization [24] is a technique for improving the performance and stability of artificial neural networks. It adds a normalization step (shifting inputs to zero mean/unit variance) to make the inputs of each trainable layers comparable across features. By doing this it ensures a high learning rate while keeping the network learning.

3.2.4. Fully Connected Layer

Fully connected layer usually appears at the end of CNN. The function of the fully connected layer is to map the feature maps to the label. Mathematically, we can think of a linear layer as a function which applies a linear transformation on a vectoral input of dimension I and output a vector of dimension O. Usually the layer has a bias parameter:

$$f(x) = W \cdot x + b \tag{3.2}$$

The features learned by previous convolution layer are generally local features. These local features will be connected as a global feature through the fully connected layer. Fully connected layer is not necessary. For the reason that the fully connected layer has a large number of parameters, which will affect the training speed.

3.2.5. Activation Functions

Convolution layer is a kind of linear operation. Stacking convolution layers make the CNN be a linear system. It means that the CNN will only be able to solve linear problems and make nonsense. Hence CNN needs non-linear factors in order to bring in

non-linearity property that enables them to approximate any function. This non-linear factor is called activation function. Every kind of activation function takes a vector and performs a certain fixed point-wise operation on it. Mainly, all the activation functions can be divided into two groups, which are non-parametric activation function and parametric activation functions.

non-parametric activations: Sigmoid function, Tanh function, ReLu function, and LeakyReLu function [25] are commonly used in practice. The mathematical forms and graphs of them are shown as follows:

Sigmoid Function:

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \tag{3.3}$$

Tanh Function:

$$f(x) = tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.4}$$

ReLu Function:

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{3.5}$$

LeakyReLu Function:

$$f(x) = \begin{cases} x & x > 0 \\ \lambda x & x \leq 0 \end{cases} \tag{3.6}$$

where λ is a small value, often set to 0.01.

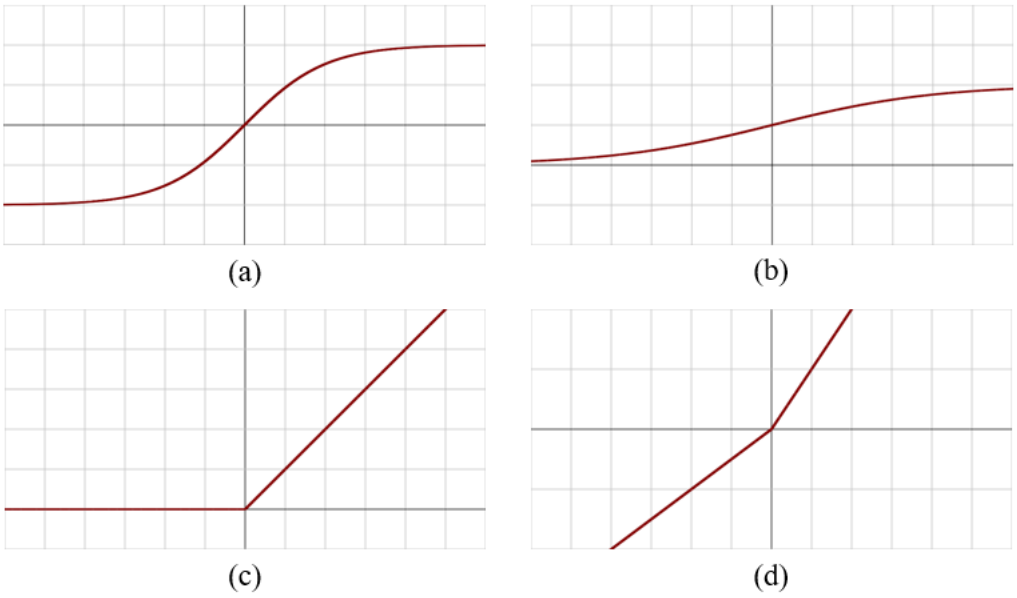


Figure 3.6. Graphes of activation functions: (a) Sigmoid function; (b) Tanh function; (c) ReLu function; (d) LeakyReLu function [26].

parametric activation functions: The typical parametric activation function is PReLu [27] (Parametric Rectified Linear Unit). The mathematical form is:

$$y_i = \begin{cases} x_i & x_i > 0 \\ \lambda x_i & x_i \leq 0 \end{cases} \tag{3.7}$$

As its name suggests, λ is the parameter need to be learned. If we set λ to 0, then PReLu becomes ReLu. If we set λ to a small fixed value, it becomes LeakyReLu.

3.3. Training of Convolutional Neural Network

The ability to learn is the key concept of neural networks. The aim of the process is to find the optimal parameters (and structure) of the network for solving the given task. Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Learning is then carried out on the training set by means of feeding the training data through the network. The networks will be trained based on gradient back propagation [28]. Namely, by computing the difference between the output of a neuron network and the expected output, we keep propagating the error back through the whole network guiding the parameters updating. The network is considered to be trained after reaching the target performance on the training data.

3.3.1. Initialization

All the network parameters are generally initialized with Layer-sequential unit-variance (LSUV) (e.g. parameters as Gaussian random variables with mean 0 and standard deviation 1 and biases are initialized to 0). Since the LSUV initialization works under assumption of preserving unit variance of the input, pixel intensities are given after subtracting the mean and dividing by the standard deviation.

3.3.2. Loss function

As a significant part in training CNN, loss function is used to quantify the capacity of the network to approximate the ground truth labels for all training inputs, which takes as inputs the weights, biases, and examples from the training data. It is a non-negative value, where the robustness of model increases along with the decrease of the value of loss function. Mean Square Error, Cross Entropy and Kullback Leibler Divergence are often used as loss function in tanning.

Mean Square Error (MSE): It is a multi-class loss widely used in linear regression as the performance measure.

$$Loss(x, y) = \frac{1}{n}\sum_{i}|x_i - y_i|^2 \tag{3.8}$$

where x is the vector of n predictions, and y is the binary vector full of 0 besides a 1 in the corresponding class dimension.

Cross Entropy: It is commonly-used multi-class loss in binary classification (labels are assumed to take values 0 or 1) as a loss function. Cross entropy measures the divergence between two probability distribution, if the cross entropy is large, which means that the difference between two distribution is large, while if the cross entropy is small, which means that two distribution is similar to each other. It is computed as follows:

$$Loss(x, y) = -\sum_{i} y_i * \log(\frac{e^{x_i}}{\sum_j e^{x_j}}) \tag{3.8}$$

where x is the vector of n predictions, and y is the binary vector full of 0 besides a

1 in the corresponding class dimension.

Kullback Leibler Divergence (KL divergence): It is also called relative entropy, information divergence/gain, which is a measure of how one probability distribution diverges from a second expected probability distribution. The mathematical form is:

$$Loss(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i \cdot \log x_i) - \frac{1}{n}\sum_{i=1}^{n}(x_i \cdot \log y_i) \tag{3.9}$$

where x is the vector of n predictions, and y is the binary vector full of 0 besides a 1 in the corresponding class dimension. We can see the first term is entropy and the second is cross entropy. The KL divergence of 0 indicates that we can expect similar behavior of two different distributions, while a KL divergence of 1 indicates that the two distributions behave in such a different manner that the expectation given the first distribution approaches 0.

3.3.3. Backpropagation

Backpropagation, short for "backward propagation of errors" is an algorithm for supervised learning of artificial neural networks using gradient descent. Backpropagation is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data (in image recognition, multiple images) is processed.

For each training data, we compute the prediction and its associated loss. We sum up all the loss to compute the final error. Then we use the backpropagation algorithm to propagate the error in order to compute the partial derivatives of the cost function for all weights and bias. In this paper, since our goal is not to explain in details how works the backpropagation algorithm. We advise the curious reader to find details about backpropagation in [29].

3.3.4. Optimization

Once all the derivatives are computed, we should find parameters of a neural network that significantly reduce a cost function, which typically includes a performance measure evaluated on the entire training data. We then iterate the

predication (e.g. forward pass), the backpropagation of errors (e.g. backward pass) and the optimization until convergence hopping to find a local minimum low enough to ensure good predictions. The most fundamental algorithm is Gradient Decent, and advanced algorithms include Stochastic Gradient descent (SGD), Momentum, AdaGrad, RMSProp and Adam. Details can be checked in [29].

3.3.5. Regularization

Deep and large enough neural networks can memorize any data. During training, their accuracy on the training dataset typically converges towards perfection while it degrades on the test dataset. This phenomenon is called overfitting. To avoid overfitting, regularization has been imported. We will make a brief introduction on several regularization approaches.

L2 Regularization: The first main approach to overcome overfitting is the classical weight decay, which adds a term to the cost function to penalize the parameters in each dimension, preventing the network from exactly modeling the training data and therefore help generalize to new examples:

$$Err(x, y) = Loss(x, y) + \sum_i \theta_i^2 \tag{3.10}$$

here θ means the vector containing all the network parameters.

Data augmentation: It is a method of boosting the size of the training dataset so that the model cannot memorize all of it. This can take several forms depending of the dataset. For instance, if the objects are supposed to be invariant to rotation, it is well suited to apply different kind of rotations to the original images.

Dropout [30]**:** The idea is to randomly set a certain percentage of the activations in each layer to 0. During the training, neurons must learn better representations without co-adapting to each other being active. During the testing, all the neurons are used to compute the prediction and Dropout acts like a form of model averaging over all possible instantiations of the model.

3.4. Famous Architectures in Convolutional Neural Network

In this section, we will introduce several important modern CNN architectures. LeNet [31] which was developed by Yann LeCun in 1990, is the first successful application of CNN in digit recognition. LeNet consists of a sequence of convolutional layers and max pooling layers followed by a fully connected layer. AlexNet [32] is the milestone for deep learning in computer vision which occupied the first place on the ImageNet ILSVRC [33] challenge in 2012, showing significant gains in performance comparing to previous methods. The network has similar architecture to LeNet but is deeper and bigger and features convolutional layers stacked on top of each other. VGG16 [34], which was the runner-up in ILSVRC 2014 with almost 140 million of parameters, demonstrated the importance of depth as a critical component to good performance. The architecture of VGG16 consists of stacked convolutional layers and max pooling layers, with increasing depth and it uses a large number of parameters due to the final fully connected layers. GoogLeNet [35] was the winner of ILSVRC 2014 challenge. The architecture includes inception modules which dramatically reduce the number of parameters, it uses multi-scale 3x3, 5x5 convolutional filters including 1x1 convolutions for dimensionality reduction. ResNet [36] won the ILSVRC 2015 challenge, which introduces skip connections for easier training that enable very deep architectures and makes use of batch normalization. Different from ResNet, which draws representational power from extremely deep network architectures, DenseNet [37] designed the dense blocks to improve the model from the perspective of feature maps reusing. We will make an introduction for these famous structures in the following subsections.

3.4.1. LeNet

The LeNet architecture is an excellent “first architecture” for CNN (especially when trained on the MNIST dataset, an image dataset for handwritten digit recognition).

LeNet is small and easy to understand, yet large enough to provide interesting results. This network was the origin of much of the recent architectures, and a true inspiration for many people in this field. Figure 3.7 shows the architecture of LeNet-5. LeNet-5 features can be summarized as:

- Sequence of 3 layers: convolution, pooling, non-linearity.

- Inputs are normalized using mean and standard deviation to accelerate training.
- Sparse connection matrix between layers to avoid large computational cost.
- Hyperbolic tangent or sigmoid as non-linearity function.
- Trainable average pooling as pooling function.
- Fully connected layers as final classifier.
- Mean squared error as loss function.

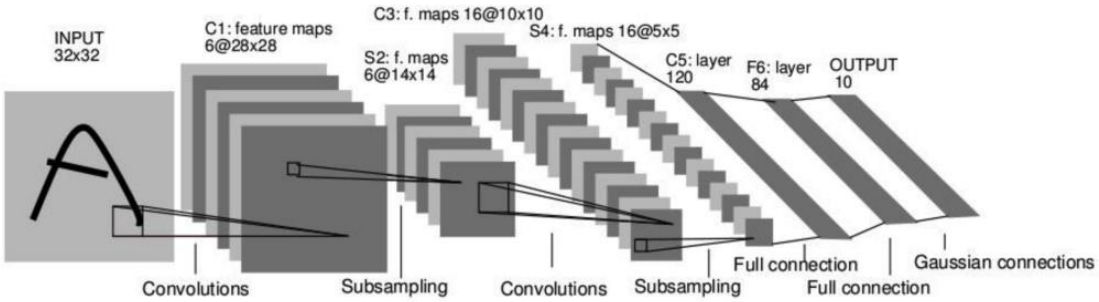


Figure 3.7. Architecture of LeNet-5 [31].

3.4.2. AlexNet

This paper, titled “ImageNet Classification with Deep Convolutional Networks”, has been cited over 30,000 times and is widely regarded as one of the most influential publications in the field of computer vision. This architecture was applied to win the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge), which achieved a top 5 test error rate of 15.4% (Top 5 error is the rate at which, given an image, the model does not output the correct label with its top 5 predictions), far exceeded the next best entry with an error of 26.2%. This was an astounding improvement that pretty much shocked the computer vision community.

The layout of AlexNet is relatively simple compared to modern architectures. The network was comprised of 5 convolutional layers, 5 max-pooling layers, 5 dropout layers, and 3 fully connected layers. The network they designed was used for classification with 1000 possible categories. And the authors provided a multi-GPUs implementation in CUDA to bypass the memory needs. Figure 3.8 shows the architecture of AlexNet. The Main points of AlexNet includes:

- Trained the network on ImageNet data, which contained over 15 million annotated images from a total of over 22,000 categories.

• Used data augmentation techniques that consisted of image translations, horizontal reflections, and patch extractions.

• Implemented dropout layers in order to combat the problem of overfitting to the training data.

• Trained the model using batch stochastic gradient descent, with specific values for momentum and weight decay.

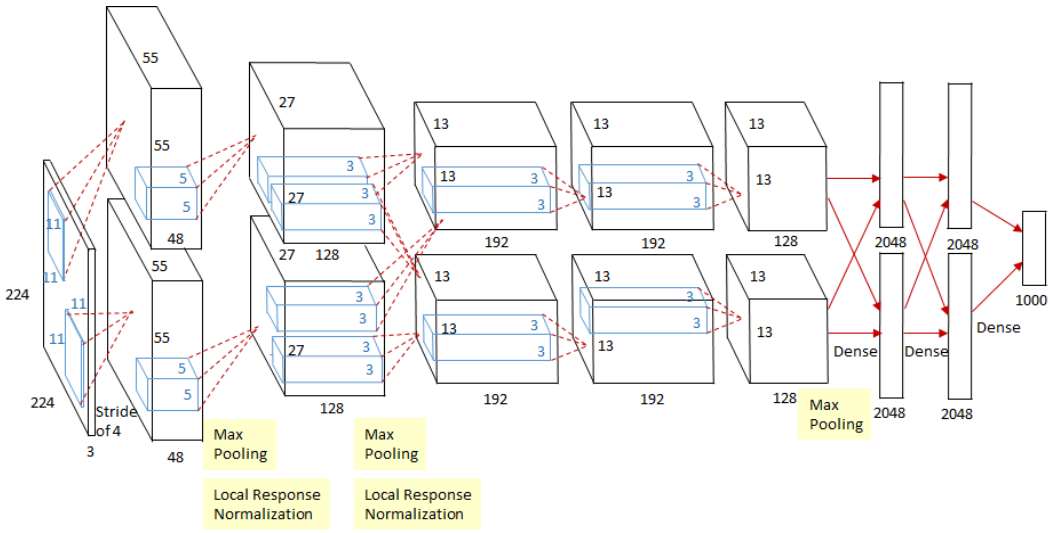


Figure 3.8. An illustration of the architecture of AlexNet [32].

This was the first time a model performed so well on a historically difficult ImageNet dataset. Techniques such as data augmentation and dropout still play an important role in CNN architectures today. AlexNet illustrated the advantages of CNN architectures in contrast with traditional methods in the field of computer vision.

3.4.3. VGGNet

The main contributions of VGGNet are showing that depth is a critical component for good performance, to use much smaller 3 × 3 filters in each convolutional layer, and also to combine them as a sequence of convolutions. Figure 3.9 demonstrates the 6 different architectures of VGGNet. The great advantage of VGGNet was the insight that multiple 3×3 convolution in sequence can emulate the effect of larger receptive fields, for examples 5 × 5 and 7 × 7. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes. One of the benefits is a decrease in the number of parameters. And as the spatial size of the input volumes at each layer decrease (result

of the convolutional and pooling layers), the depth of the volumes increase due to the increased number of filters when going down the network. These ideas were also used in more recent network architectures as GoogLeNet and ResNet.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

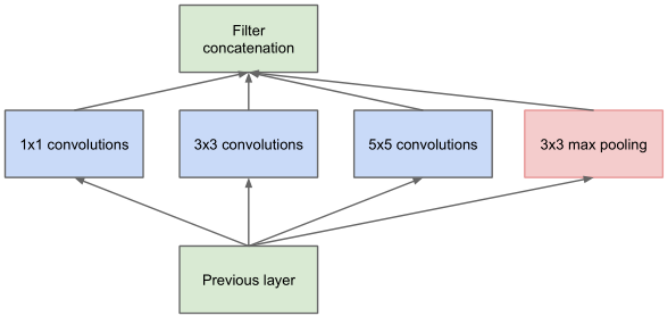
Figure 3.9. The 6 different architectures of VGGNet [34].

3.4.4. GoogLeNet

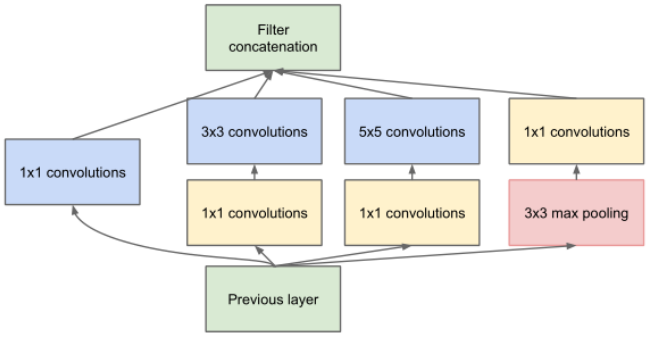
The main contribution of GoogLeNet is the development of the Inception module which dramatically reduced the number of parameters (The authors of the paper emphasized that this new model places notable consideration on memory and power usage). The architecture of the Inception module is shown in Figure 3.10. GoogLeNet is a 22 layers CNN and was the winner of ILSVRC 2014 with a top 5 error rate of 6.7%. This work was one of the first CNN architectures that really strayed from the general approach of simply stacking convolution and pooling layers on top of each other in a sequential structure.

In previous CNN architecture, you have to make a choice of whether to have a

pooling operation or a convolution operation for each layer. The most fantastic thing in GoogLeNet is that Inception module allows you to do is perform all of these operations in parallel. An Inception module consists of a medium sized filter convolution, a large-sized filter convolution, and a pooling operation. (There are activation operations by ReLu after each convolutional layer) Basically, the model is able to perform the functions of these different operations while still remaining computationally considerate.



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Figure 3.10. Architectures of Inception module [35].

3.4.5. ResNet

Before ResNet, CNN models improve network performance mainly by increasing the depth and width of the network or by reducing the size of filters. However, with the network depth increasing, there is a problem that accuracy gets saturated and then degrades rapidly and the same phenomena also appears in training when adding more layers to a suitably deep model [36]. To solve this problem, Kaiming He et al. proposed ResNet, which is a new 152-layer (the deepest version) network architecture that set new records in classification, detection, and localization through one incredible architecture. Aside from the new record in terms of the number of layers, ResNet won

ILSVRC 2015 with an incredible error rate of 3.6% comparing to humans who generally hover around a 5-10% error rate.

ResNet addressed the degradation problem by introducing a new module called Residual Block [36] see in Figure 3.11. Instead of hoping each few stacked layers directly fit a desired underlying mapping, ResNet explicitly let these layers fit a residual mapping. Formally, denoting the desired underlying mapping as $H(x)$, we let the stacked nonlinear layers fit another mapping of $F(x) = H(x) - x$. The original mapping is recast into $F(x) + x$. So, instead of just computing that transformation from x to $F(x)$, we compute the term that you have to add, which is a slightly altered representation (Comparing to traditional CNN, going from x to $F(x)$ which is a completely new representation that does not keep any information about the original input x).

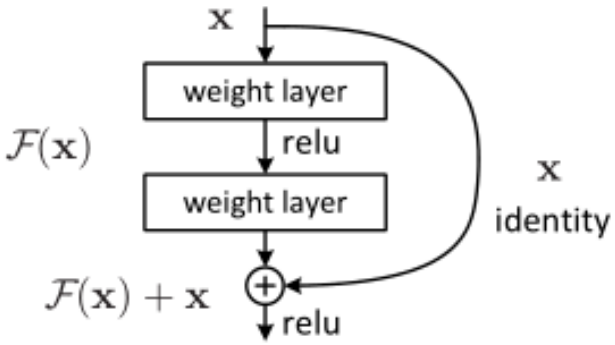


Figure 3.11. Residual learning: a building block [36].

3.4.6. DenseNet

ResNet solved the gradient vanishing problem based on the idea that creating short paths from previous layers to later layers. However, there is a drawback of ResNet that the shallow feature maps can not be passed to deep layers, which makes it hard to explore new features. In fact, simply increasing the number of filters in each layer of ResNet can improve its performance provided the depth is sufficient [38]. Inspired by this, instead of drawing representational power from extremely deep or wide architectures, DenseNet exploits the potential of the network through feature reuse, yielding condensed models that are easy to train and highly parameter-efficient. DenseNet shows significant improvements over the state-of-the-art on most of previous architectures, whilst requiring less computation to achieve high performance.

The feature maps reusing is done by the designed dense block, whose number of L layers connections is $L(L+1)/2$, comparing to the traditional CNN which contains L connections in L layers. Namely, the input for each layer in a dense block comes from the output of all the previous layers. The connections in the dense block makes the transfer of features and gradient more efficient. Due to the design of such a block, the model also become easier to train. The architecture of a 5-layer dense block is displayed in Figure 3.12, where the growth rate $k = 4$ indicates that each layer outputs 4 feature maps.

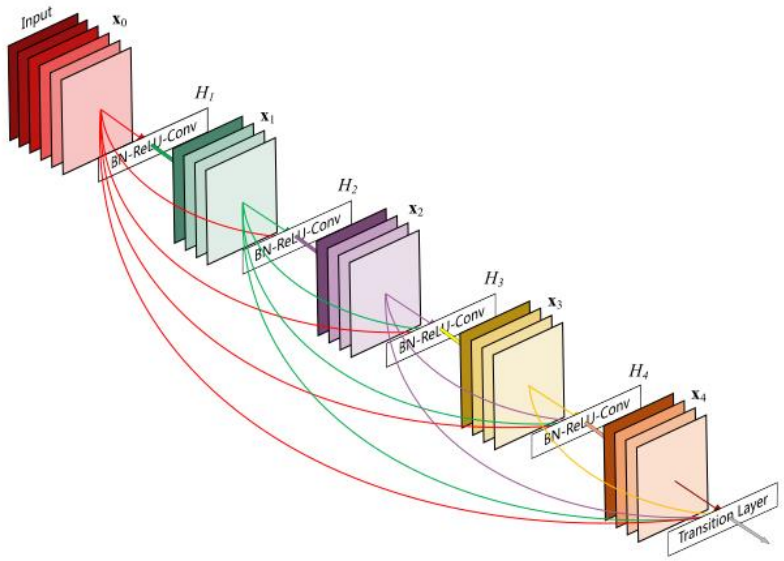


Figure 3.12. The architecture of a 5-layer dense block with a growth rate of $k = 4$ [37].

Chapter 4.

Road Facility Extraction from Laser Scanning Data

In this chapter, we will give the details on how to extract road facilities from laser scanning data. In section 4.1, we will discuss pole-like traffic facilities extraction and classification, and describe our method and experimental results. Road marking extraction will be introduced in section 4.2.

4.1. Pole-like Traffic Facilities Extraction and Classification

In this section, we will firstly talk about related works of pole-like traffic facilities extraction and classification. Then, we will describe our proposed method of pole-like traffic facilities extraction and classification. Finally, the experimental results will be given.

4.1.1. Related Works

For pole-like objects possess a cylinder like structure, cylindrical shape-based method is a kind of method to do poles extraction. [39] detects poles from ranged laser data by using Hough voting to detect circles. [40] introduces an algorithm which divides the point cloud in slices and compares projective parameters of pillar objects in adjacent slices to extract the pillar features of Chinese ancient buildings. However, there is a problem for this kind of method that the circular characteristic for poles is obvious only for indoor environments or close-range scans.

Slicing-based Method is also an often-used approach to extract pole-like structures, which slice the point cloud data for finding vertical objects, and to reduce the influence of structures attached to the vertical trunk. Pu et al. [41] extend the work of [40] and propose a percentile-based method to detect the pole-like object. However, the validation method of these works is mainly based on the deviation of the neighboring

subparts, which is capable of dealing with pole-like objects with attachments only in the bottom or on the top of the trunks. In order to detect pole-like objects with rich attachments in the middle part, Huang et al. [42] take advantage of the unique characteristics of pole-like objects. They propose the localization algorithm based on slicing, clustering, pole seed generation and bucket augmentation. Then they integrate the bucket-shaped neighborhood of the segments and trim with region growing algorithms to obtain pole-like objects.

A segmentation procedure is also routinely used for extracting poles. Segmentation is the process of grouping the points of the cloud into segments: points in the same region are given the same category and treated as a set. Since the grounds or façades which connect other objects in a huge cluster, filtering and segmentation of point cloud data is usually needed. In the paper of [43], they assume that ground points are already removed from the point cloud data, so they do not need to filter the ground area. Golovinskiy et al. [44] use iterative plane fitting to remove the ground, and separate the foreground from the background by a graph-cut-based method. Tombari et al. [45] apply a RANSAC-like iterative algorithm to remove all planar surfaces.

Before the pole-like objects classification, point classification need to be done to analyze the composition of the point cloud. By applying Gaussian Mixture Model (GMM) with Expectation Maximization algorithm, Lalonde et al. [46] get a model of the three saliency features (linear, planar and volumetric) derived from the eigenvalues of the local covariance matrix of the points. In the paper of [45], they propose a histogram of scalar products to classify the vertical pole points using Support Vector Machine (SVM).

Since the usage and shapes of the pole-like objects are quite diverse, the usage, shapes and even height are often used to classify poles in existing works. Pu et al. [41] make a detailed classification on the signs according to the planar shapes. However, they do not distinguish between non-planar poles (e.g. lights and utility poles). Golovinskiy et al. [44] make a mixed categorization of usage and height, resulting in seven categories including short post, lamp post, sign, light standard, traffic light, tall post and parking meters, while the utility poles are not reported. Yokohama et al. [43] classify the poles into three categories: street lights, utility poles and signs. By distinguishing between wire points and the general linear points, Huang et al. [42] make a more delicate classification of four categories: street lights, utility poles, signs and

non-pole category. In this thesis, since c is very high, we also utilize the feature of reflectance to help the classification.

4.1.2. System Overview

The procedure of our proposed pole-like road facilities extraction and classification system is demonstrated in Figure 4.1. Input of the system is a large-scale point cloud data of an urban area, and the outputted results are all possible pole-like structures which are classified into 4 categories: utility poles, traffic signs, street lights, and others (non-traffic facility poles). The system mainly consists of two stages in processing. The first stage is to extract the pole-like objects from input data. We remove the outlier points before clustering. Then Euclidean clustering is applied. Next, we take advantage of the unique characteristic of pole-like structures (the local parts of the pole-like structures are also pole-like even they are broken down) to stitch clusters and filter them with their height. After that, z-slicing is used to analysis the shape feature of every cluster. And finally pole-like candidates are detected after thresholding. The second stage is to classify these candidates. We compute the statistical attributes for each candidate based on the extended distribution features of pole components and classify the candidates with a support vector machine. The classification step also help filters out non-traffic facility objects which also contain pole-like parts like trees and part of buildings. We will discuss details of these two stages in the following subsections.

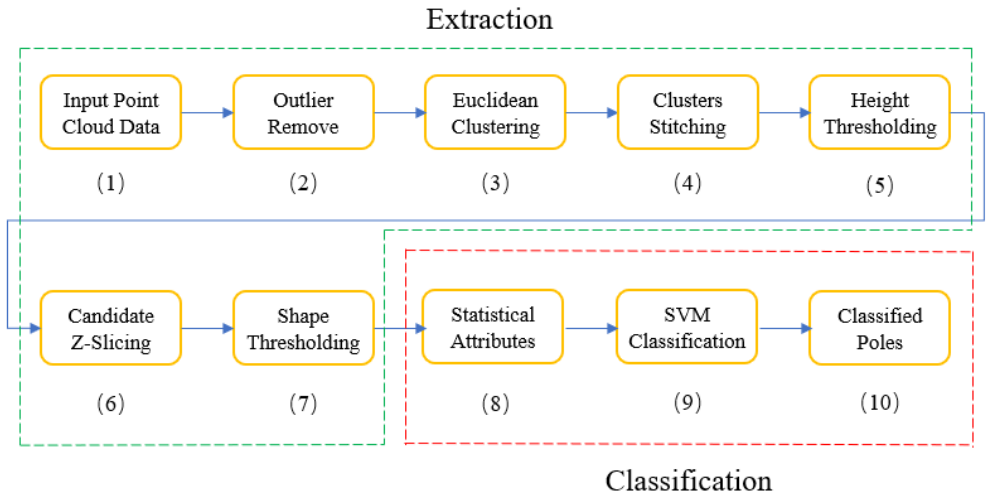


Figure 4.1. The procedure of the proposed pole-like road facilities extraction and classification system.

4.1.3. Pole-like Structures Extraction

In our practice, we utilize two laser scanners, with 180° field of view, one configured to point up (pitch: +25°) and the other down (pitch: −25°) in our Mobile Mapping System. And the data collected by the upper one is applied as the input data in our pole-like traffic facilities extraction and classification, in which there is no ground area point. So, we can skip the step of removing ground area points from given point clouds.

Since laser scans typically generate point cloud datasets of varying point densities. And additionally, measurement errors lead to sparse outliers which corrupt the results even more. It is necessary for the input point cloud data to remove noisy measurements before clustering. We exclude the outlier points by mean k-nearest neighbor distances for each point, and set the parameter $k = 10$.

Then we do the clustering for the point cloud data. A clustering method needs to divide an unorganized point cloud model into smaller parts so that the overall processing time for is significantly reduced. We use Euclidean cluster extraction algorithm to extract clusters and a k-d tree structure [47] is applied for finding the nearest neighbors. The algorithmic steps for that are as follows:

Algorithm4.1: Euclidean cluster extraction

Input: point cloud data $\boldsymbol{P}$

Output: point clusters $\boldsymbol{C}$

1: create a k-d tree representation for the input point cloud data $\boldsymbol{P}$;

2: set up an empty list of clusters $\boldsymbol{C}$, and a queue of the points that need to be checked $\boldsymbol{Q}$;

3: then for every point $\boldsymbol{p_i} \in \boldsymbol{P}$, perform the following steps:

- add $\boldsymbol{p_i}$ to the current queue $\boldsymbol{Q}$;
- for every point $\boldsymbol{p_i} \in \boldsymbol{Q}$, do:
  - search for the set $\boldsymbol{P_i^k}$ of point neighbors of $\boldsymbol{p_i}$ in a sphere with radius $\boldsymbol{r < d_{th}}$;
  - for every neighbor $\boldsymbol{p_i^k} \in \boldsymbol{P_i^k}$, check if the point has already been processed, and if not add it to $\boldsymbol{Q}$;

- when the list of all points in $\boldsymbol{Q}$ has been processed, add $\boldsymbol{Q}$ to the list of clusters $\boldsymbol{C}$, and reset $\boldsymbol{Q}$ to an empty list.

4: the algorithm terminates when all points $\boldsymbol{p_i} \in \boldsymbol{P}$ have been processed and are now part of the list of point clusters $\boldsymbol{C}$.

We consider that facility poles should obtain a certain amount of points in point cloud data. clusters which are less than 60 or more than 8000 points are excluded.

Some of the traffic facilities poles would be divided into two or more parts due to the sparsity of points and the choice of radius in the clustering step. To deal with this situation, the stitch of clusters is applied. If two clusters have a relatively big distance mainly in the z direction, we consider them as one cluster. Suppose the centroid point of 10 lowest (in z direction) points in higher cluster is $p_h = (x_h, y_h, z_h)$, the centroid of 10 highest (in z direction) points in lower cluster is $p_l = (x_l, y_l, z_l)$. And the vector from p_l to p_h is $\vec{v}_{dif}$. The mathematical form of the condition is:

$$\begin{cases} |z_h - z_l| < d_{th} \\ \left| \dfrac{\vec{v}_{dif}}{\|\vec{v}_{dif}\|} \cdot \vec{z}_{unit} \right| < \theta_{th} \end{cases} \tag{4.1}$$

Where $\vec{z}_{unit}$ means the unit vector in z direction, d_{th} is the threshold distance in z direction which we set a value of 1 meter, and θ_{th} is the threshold angle between the two centroid points which we set a value of 0.35.

Since the pole-like traffic facilities we want to extract (utility poles, traffic signs and street lights) have to be created according to some standards. Therefore, the height of potential facility poles should be in a certain range. Based on the statistics in our dataset, we set the range of the height filter from 1.5 meters to 9 meters.

One of the important properties of the poles' trunk part in the point cloud is that, if it's horizontally sliced, each of its slices would still be a trunk-like segment (though some of the slices have an attachment part). Given a horizontal slice of a pole-like object, there are two obvious characteristics: first, the cross section of the slice is relatively small; second, the length of each slice is long enough. Moreover, the bounding box of a piece of a slice part is very close to the original shape. Therefore, we have the following two criteria regarding the cross area and the segment length based on the bounding box property of each candidate cluster:

$$\begin{cases} \max(B_x, B_y) < L \\ B_z \geq k \cdot H_{slice} \end{cases} \tag{4.2}$$

Note that the width, depth and height of the bounding box are B_x, B_y and B_z, respectively. L means the maximum value of a cross section that would be considered as a trunk, H_{slice} is the slicing height, and k is the ratio coefficient of the length. We empirically set $L = 0.4m$, $H_{slice} = (z_{max} - z_{min})/number\ of\ slices$ and $k = 0.8$. Based on this property, we slice the input clusters along the z (upright) direction into 10 pieces and count the number of slices satisfied the two criteria. Since some of the slices have an attachment part, most but not all of the slices should satisfy the two criteria for pole-like traffic facilities. In our case, we consider the cluster to be a candidate of pole-like traffic facility when more than half of the slices satisfy the two criteria, and not all of the slices satisfy the first criteria.

4.1.4. Pole-like Structures Classification

Due to the previous step of shape thresholding are relatively not so strict in order to extract more potential pole-like traffic facilities as possible. As a side effect, many candidates are actually trees with trunks or parts of buildings with pole-like structures. So, Classification aims to exclude these pole-like structures need to be done after extraction. The ones classified as trees or others would be removed from the candidates. Meanwhile, we should identify the detected traffic facility poles whether they are belongs to utility poles, traffic signs, or street lights.

Some attributes from the candidates should be extracted before applying the classifier. The most straightforward attribute about a candidate is the height $h = z_{max} - z_{min}$. Since the pole-like traffic facilities we would like to extract are man-made objects. therefore, they have fixed height for certain sub-categories, which can be applied as a classify feature. Another significant attribute is the high intensity (which we set as bigger than 240) point rate of a candidate. For the intensity of a traffic sign would be high when the scanning faces to the sign. This can be also used as a classify feature to classify facing traffic signs.

Since the usage of a traffic facility pole is highly dependent on its local shapes or attachments, which can be categorized as three kinds of components: linear, planar and volumetric, these features can be applied to do the classification. As all pole-like

candidates contain a vertical linear trunk. Besides that, traffic lights could contain linear branches or volumetric bulbs, signs contain planar components, and utility poles contain linear wires, and sometimes contain planar signs and volumetric parts. These components are further composed of local point-level patches of the same property, so we just need to make a classification on the neighborhood of each point. Principal Component Analysis (PCA) is applied here to classify every point. Suppose that $M_{\square}$ is the variance-covariance matrix of the point i. The mathematical form is:

$$M_i = \frac{1}{|S_i|}\sum_{j\epsilon S_i}(p_j - \bar{p_i})(p_j - \bar{p_i})^T \tag{4.3}$$

where S_i is the set of neighbor points which lie in the sphere area with radius $r_S = 0.5m$, centered at the point i, and $\bar{p_i}$ is the centroid of S_i. We denote eigenvalues of M_i by λ_1, λ_2, and λ_3 (where $\lambda_1 \geq \lambda_2 \geq \lambda_3$), the corresponding eigenvectors are $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$ respectively. The distribution of neighbors of point i can be indicated by the magnitude relation of the eigenvalues [46]. When the maximum eigenvalue λ_1 is much larger than λ_2 and λ_3, it means that there is one principal direction and the distribution is linear; When λ_2 is approximate to λ_1 and they are very large compared with λ_3, it represents that there are two principal directions and the distribution is planar; When λ_1, λ_2 and λ_3 are approximate to each other, it suggests that there's no obvious principal direction and the distribution is thus volumetric.

There are multiple kinds of mathematical forms to determine which criterion is satisfied [43, 46]. We use the equation of distribution features of [43]:

$$\begin{cases} S_1 = \lambda_1 - \alpha\lambda_2 \\ S_2 = \lambda_2 - \lambda_3 \\ S_3 = \beta\lambda_3 \end{cases} \tag{4.4}$$

Note that α and $\square$ are the adjustment coefficient which should be set according to the data. And to find feature is the most significant, we apply the dimensionality feature:

$$d = \arg_{i\in\{1,2,3\}} \max S_i \tag{4.5}$$

When the surrounding structure of a point is linear or planar or volumetric, the value of d becomes 1, 2, and 3 respectively. [43] uses the endpoint preserving Laplacian Smoothing in the cluster in order to recognize linear shapes of different radii so that it applies the parameter of α = 10 and β = 100. The smoothing has the problem

that it could cause problems in case of sparse regions, and is time-consuming [42]. Since a smaller α and β is enough for recognizing most linear, planar and volumetric structures, in our case, we use $\alpha = 5$ and $\beta = 4$.

In our dataset, street lights without traffic signs can sometimes be confused with other pole-like structure since they both consists of linear and volumetric points with the similar proportion. Therefore, we introduce the class of horizontal linear points besides linear points. There are two main differences between horizontal linear points and other linear points. First is that the horizontal linear structure in traffic facilities are thinner than other linear parts, where we set the adjustment coefficient α in (4.4) a larger value to represent the significant linearity of the structure. Second is that the directions are typically horizontal, where we apply a threshold angle to determine whether the eigenvector $\vec{v}_1$ corresponding to λ_1 is perpendicular to the upright direction.

The horizontal linear point is defined as:

$$\begin{cases} S_1' = \lambda_1 - \gamma\lambda_2 > S_2, S_3 \\ \left| \dfrac{\vec{v}_1}{\|\vec{v}_1\|} \cdot \vec{z}_{unit} \right| < \theta_h \end{cases} \tag{4.6}$$

Where γ is the adjustment coefficient which we set as 10, and θ_h is the threshold angle which we set the value as 0.4. We can see a point classification result of different pole-like structures in Figure 4.2.

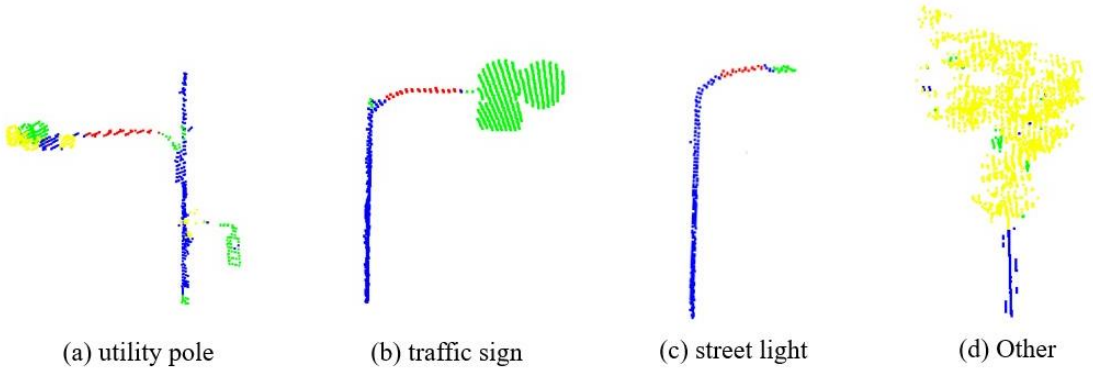


Figure 4.2. Point classification results. The colors of blue, red, green, and yellow represents normal linear, horizontal linear, planar, and volumetric points, respectively.

Classification of these traffic facility poles can be carried out based on the result of point classification. In our case, we classify these poles into 4 categories which are

utility poles, traffic signs, street lights and others as shown in Figure 4.2. Components of each point class in these 4 categories are different: utility poles can contain all 4 kinds of point class where the class of normal linear is main; traffic signs mainly contain the normal linear and planar points, and sometimes contain horizontal linear points; the main part of street lights is normal linear points with few horizontal linear and planar points; as for non-traffic facility objects, the volumetric points would be the main part (tree), or one kind of points occupies the absolute dominate position. Moreover, since the reflectance of faced traffic signs are very high, the number of high reflective points which are close to the number of in traffic signs, can be an effective feature to classify traffic signs. And the height range of poles, which are different among these pole categories, can also be a classification feature. We applied the method of Support Vector Machine (SVM) based on 6 kinds of feature do the 4-class classification.

4.1.5. Experimental Results

We use the MMS-K320 system, developed by Mitsubishi Electrics, to collect data. The MMS system is shown in Figure 4.3. As we see the system is equipped with two single-layer laser scanners and three cameras to perform a 3D measurement of the surroundings. The two laser scanners have a 180° field of view. One is configured to point up (pitch: +25°) and one down (pitch: −25°). The localization of the car is obtained by one dual frequency GPS receiver, two single frequency GPS receivers, the odometer and high-end IMU.

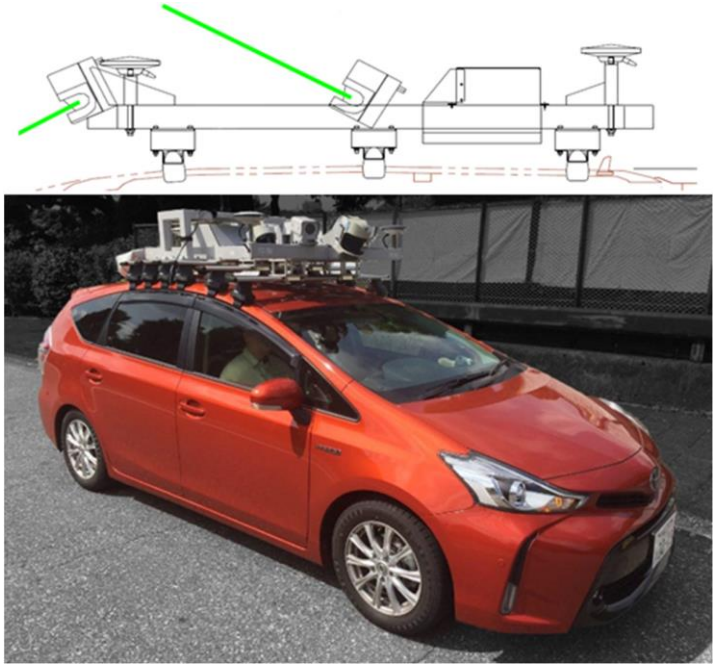


Figure 4.3. Our MMS system: Mitsubishi Electric's MMS-K320 (bottom); and the configuration of two SICK LMS-511 laser scanners and RTK GPS receivers (top).

Our experiment data is acquired around Hitotsubashi intersection, which is a dense urban area in the Chiyoda-ku area of central Tokyo, Japan. Streets around this area are surrounded by tall buildings, trees, and traffic facilities. Our experimental routes are displayed in Figure 4.4.

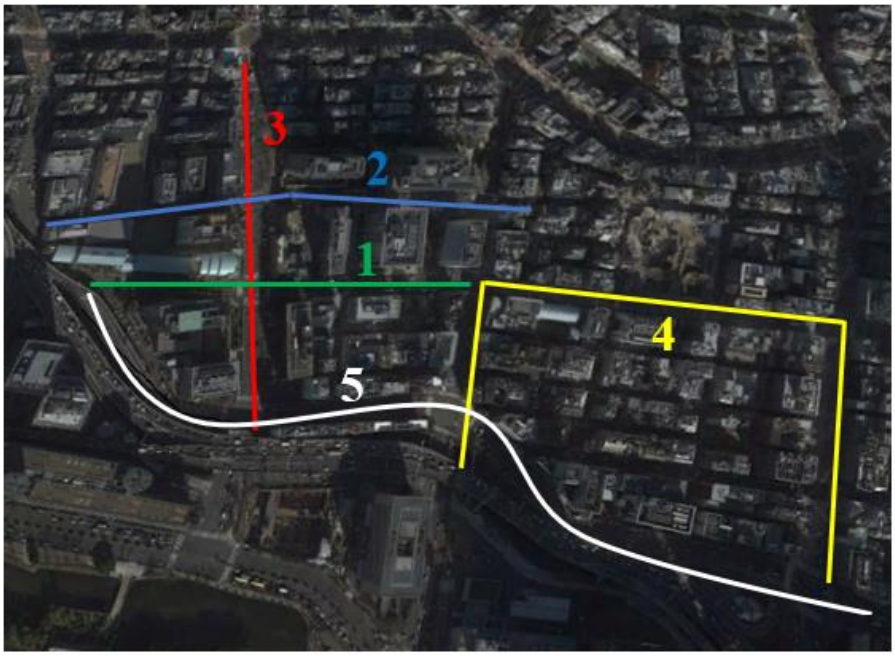


Figure 4.4. Our experimental routes in Hitotsubashi area.

From the laser scanning data of our experimental routes, 287 candidate pole-like

structures are extracted by our method. We manually labeled traffic facility poles in these routes to evaluation of our proposed method. We apply precision and recall criterion to evaluate our method. The computational formula of precision is:

$$Precision = \frac{TP}{TP + FP} \tag{4.7}$$

Where *TP* and *FP* mean true positive and false positive, respectively. And the computational formula of recall is:

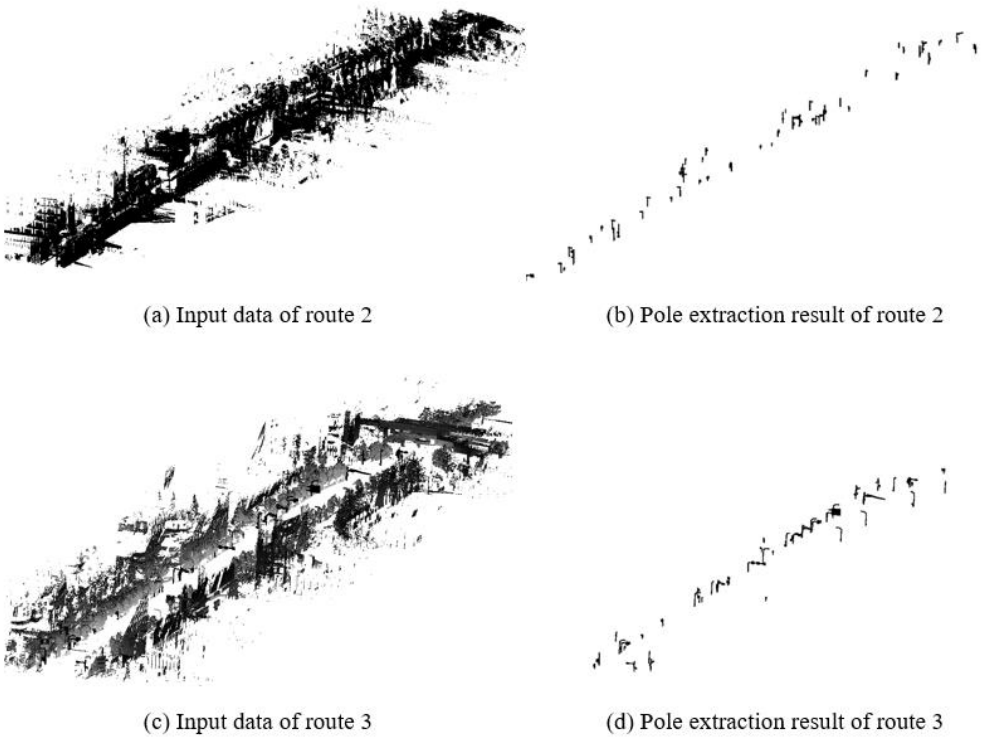
$$Recall = \frac{TP}{TP + FN} \tag{4.8}$$

Where *FN* is false negative. Evaluation of our pole-like structures extraction method is shown in Table 4.1. Among the all 274 poles (utility poles, traffic signs and street lights), our method extracts 227 of them successfully, thus the recall is 82.8%. And since 287 candidate poles are extracted, which means 60 of them are other objects such as trees or sparse building facades. Therefore, the precision is 79.1%.

Table 4.1. Evaluation of our Pole-like Structures Extraction Method

Route	Ground Truth	Extracted	Correct	Precision	Recall
1	38	41	32	78.0%	84.2%
2	46	49	37	75.5%	80.4%
3	47	45	39	86.7%	83.0%
4	68	74	59	79.7%	86.8%
5	75	78	60	76.9%	80.0%
All	274	287	227	79.1%	82.8%

Figure 4.5 gives visualization examples of pole-like structures extraction results in two routes.



(a) Input data of route 2 (b) Pole extraction result of route 2

(c) Input data of route 3 (d) Pole extraction result of route 3

Figure 4.5. Visualization examples of our pole-like structures extraction results.

Figure 4.6 shows some examples of undetected poles by our method. Since we use clustering during the pole-like structures extraction, poles that are very close to other objects would be segmented as a whole so that we can not extract the pole correctly. Besides, traffic signs on overpass cannot be extracted since they cannot be treated as a standard pole structure. Nevertheless, we can infer these signs from images since invariable categories of traffic signs could appear on overpass.

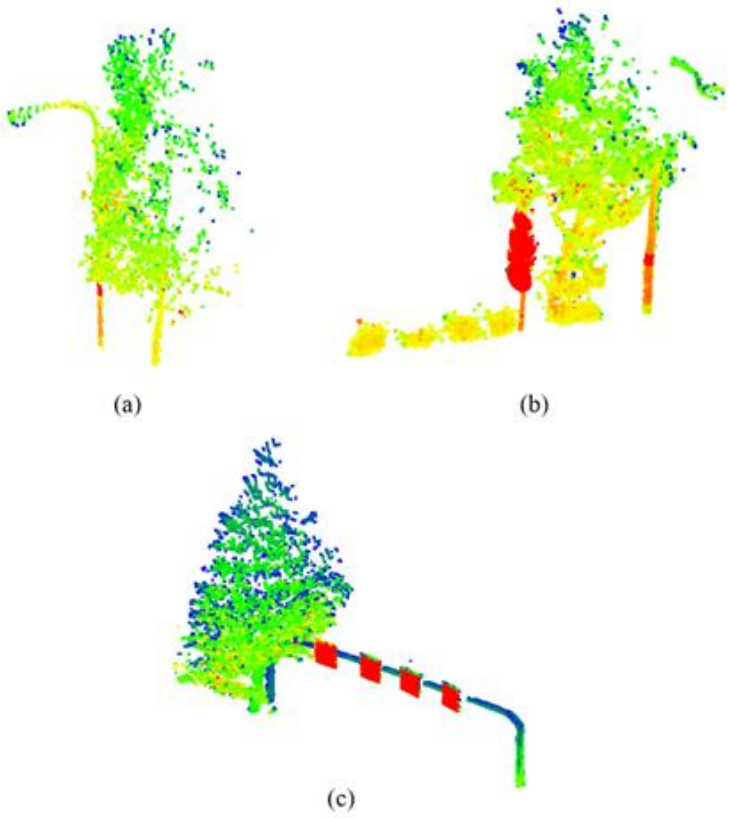
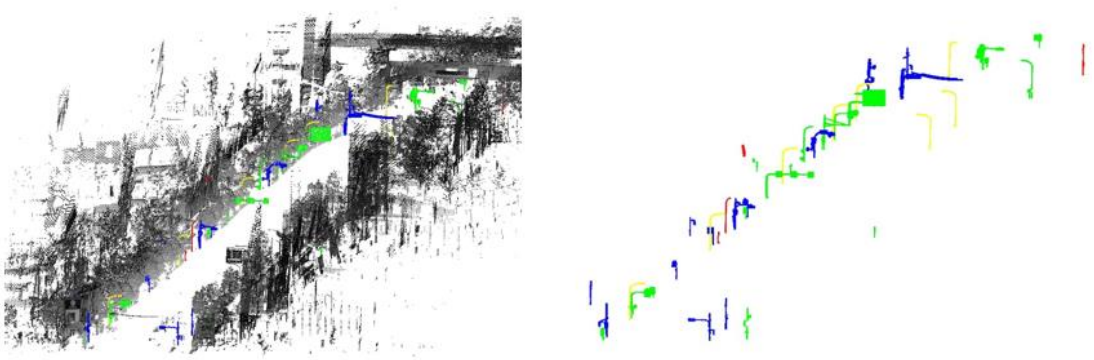


Figure 4.6. Example of undetected poles by our method.

As we mentioned above, we trained an SVM classifier for poles classification. Evaluation of our pole-like structures classification is shown in Table 4.2. And Figure 4.7 shows a visualization of our pole-like structures classification results.

Table 4.2. Evaluation of our Pole-like Structures Classification Method

Category	Classified	Correct	Precision
Utility Poles	97	75	77.3%
Traffic Signs	88	67	76.1%
Street Lights	60	49	81.7%
Others	42	35	83.3%
All	287	226	80.4%



(a) (b)

Figure 4.7. Visualization of our pole-like structures classification results (route 3). The color of blue, green, yellow and red stand for utility poles, traffic signs, street lights and others, respectively.

4.2. Road Marking Extraction

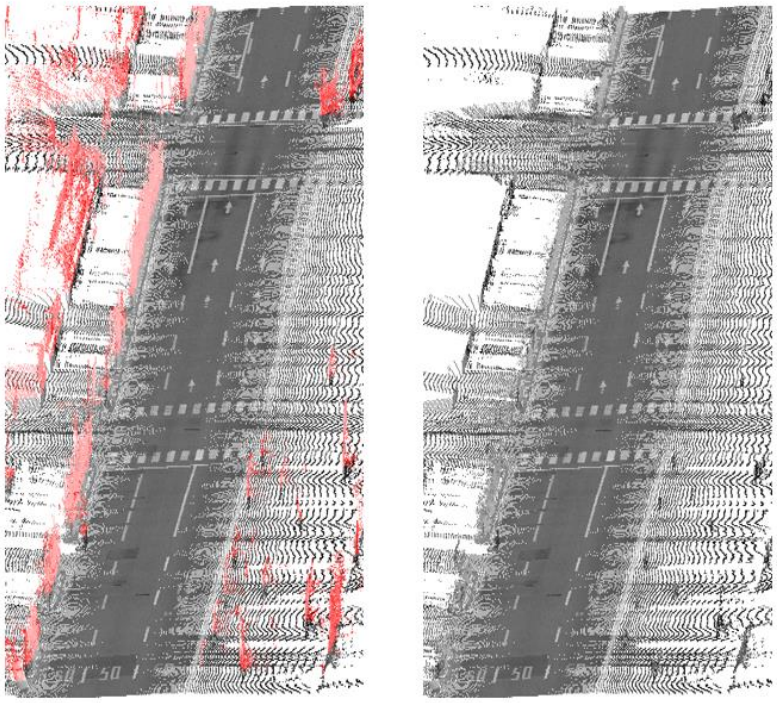
In laser scanning data, road markings painted on road surfaces are highly retroreflective, comparing to the asphalt road, which means that we can use the intensity value to extract the road markings. However, since the intensity value of road markings can be confused with some tall structures, we need to exclude these areas firstly in order to delimit the road markings searching space in the ground area. After that, road makings can be extracted by intensity value thresholding. We will follow these steps to describe details of our method of road markings extraction in this section.

4.2.1. Ground Area Segmentation

Ground area segmentation needs to be done at the first. As mentioned above, tall structure like buildings will confuse the intensity thresholding while sidewalks contain some useful information. So, the method of ground area segmentation we took should segment the road and sidewalk area while excluding buildings and other tall structures. Therefore, we applied a method based on the cloth simulation filter (CSF) [48] for ground area segmentation. Table 4.3 lists the parameters of CSF we applied in our experiment. And a result of ground segmentation in a part of point cloud data is shown in Figure 4.8.

Table 4.3. Parameters of cloth filter simulation applied in our experiment

Parameters	Value	Description
Cloth resolution	2 m	Set empirically
Max iteration	1000	More than 500 is suggested
Classification threshold	20 cm	Small values suit for cloth resolution



(a) (b)

Figure 4.8. Ground area segmentation from point cloud data: (a) original point cloud data; and (b) the result of ground segmentation.

4.2.2. Road Marking Extraction by Intensity Thresholding

After the segmentation of ground area, road markings can be extracted using the adaptive intensity thresholding procedure. Since there are some noise points after thresholding, we use the k-nearest neighbor distances filtering to remove noise points. Figure 4.9 presents a sample of the road markings extracting result. As we can see, the result shows the effectiveness of our method. Some curbs close to the sidewalks are also extracted as road markings, which can be treated as boundaries of road lanes later.

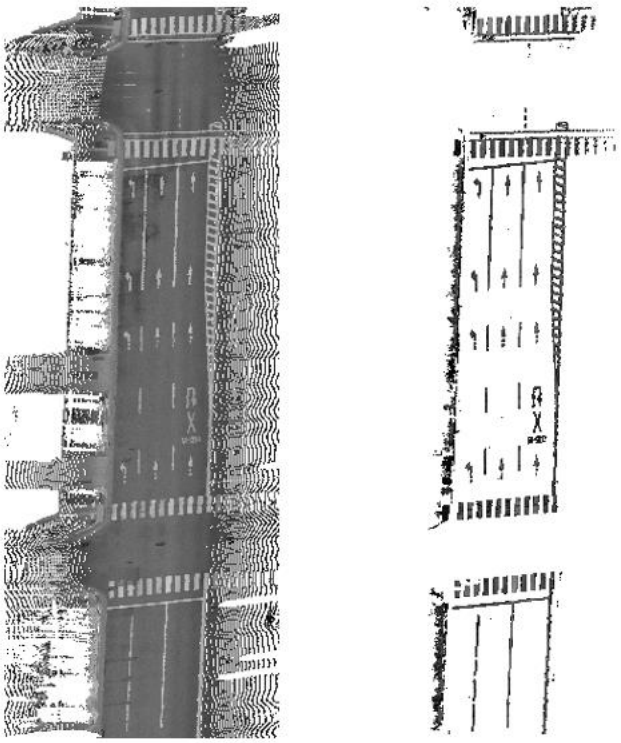


Figure 4.9. The result of road marking extraction: (a) before processing; (b) after thresholding and outlier filtering.

Chapter 5.

Image Based Road Facility Information Understanding

5.1. Road Facility Detection

5.1.1. Related Works of Object Detection

As we talked in chapter 3, the Convolutional Neural Network based deep learning have made more and more new record in many computer vision tasks in recent years. Architectures of CNN have achieved higher and higher accuracy in image classification task. For object detection, CNN based deep learning models also outperformed conventional Histogram of Oriented Gradient (HOG) [49] + SVM method from 2014. Therefore, we apply deep learning model to detect traffic facilities. There are two main kinds of deep learning architecture for object detection:

1. Two stage CNN framework: This kind of model provides region proposals with an algorithm or a light CNN network firstly, then applies a high-quality classifier to classify these proposals. We will make an introduction of RCNN [50], SPP-net [52], Fast-RCNN [54] and Faster-RCNN [55].

2. Single stage CNN framework: This kind of framework consider the task of detection as a regression problem to achieve the real-time speed of object detection. This kind of model includes YOLO [56], YOLOv2 [57], SSD [58], and YOLOv3 [59]. We will discuss the details of these object detection network in this subsection.

Girshick et al. [50] proposed a method named Regions with CNN Feature (R-CNN) in 2014, which achieved excellent object detection accuracy by using a deep Convolutional Neural Network to classify object proposals. Figure 5.1 shows the architecture of R-CNN.

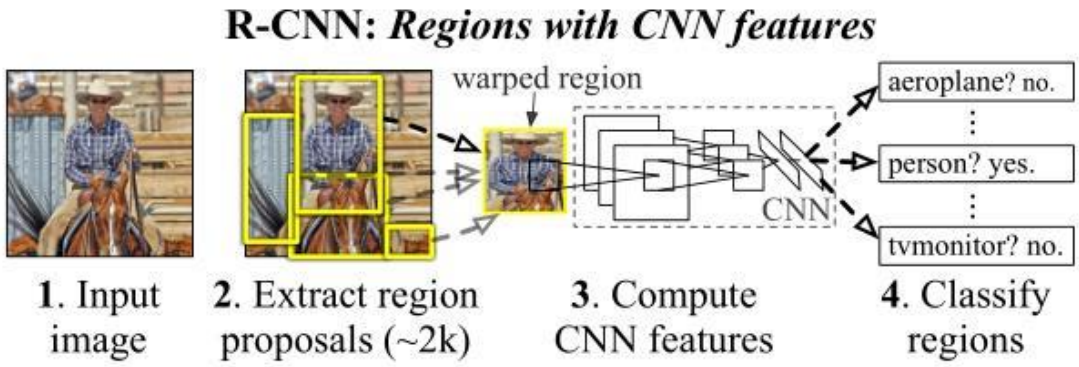


Figure 5.1. Object detection system overview [50].

This system consists of 4 steps. Firstly, an image is inputted into this system. Then it extracts around 2000 bottom-up region proposals by selective search [51], each proposal is warped to the size of 227 × 227. After that, a CNN with 5 convolutional layers and 2 fully connected layers is used to compute features for each proposal. Finally, the linear SVMs are adopted to classify each region and the bounding box regression method is applied to fix localization errors. However, R-CNN has notable drawbacks: (1) Training is a multi-stage pipeline. R-CNN first finetunes a CNN on object proposals. Then, it fits SVM classifier to CNN features. In the third training stage, bounding-box regressors are learned; (2) Training is expensive in space and time. It takes a lot of time with very deep networks, and proposals as well as features require hundreds of gigabytes of storage; (3) Object detection is slow since it performs a CNN forward pass and SVM classification for each object proposal without sharing computation. Detection with VGG16 on a GPU takes 47s per image.

He et al. [52] proposed Spatial pyramid pooling networks (SPP-net) to speed up R-CNN by sharing computation. A CNN mainly consists of two parts: convolutional layers, and fully-connected layers that follow. In fact, convolutional layers do not require a fixed image size and can generate feature maps of any sizes. According to this, they added an SPP layer on top of the last convolutional layer. Figure 3 displays the change of the network architecture by introducing the SPP layer. The SPP layer pools the features and then concatenated as in spatial pyramid pooling [53] to generate fixed-length outputs, which are then fed into the fully-connected layers (see Figure 5.2). SPP-net accelerates R-CNN by 10 to 100× at test time. Training time is also reduced by 3× due to faster proposal feature extraction.

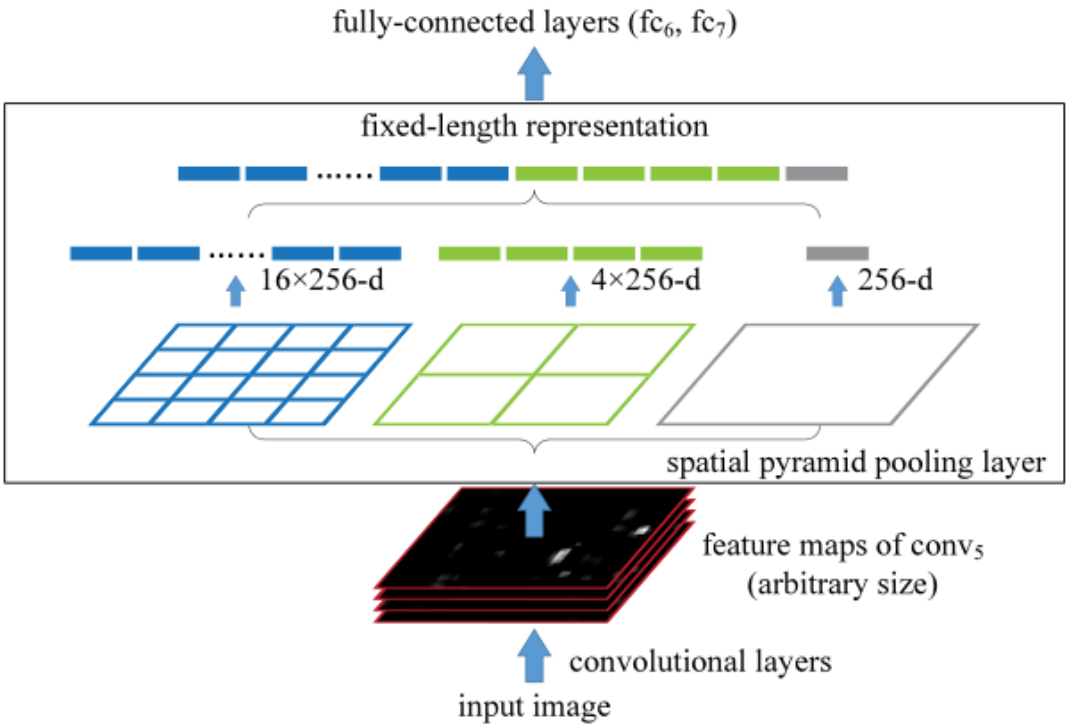


Figure 5.2. A network structure with a spatial pyramid pooling layer. Here 256 is the filter number of the conv 5 layer, and conv 5 is the last convolutional layer [52].

Nevertheless, SPP-net also has notable drawbacks. Training is a multi-stage pipeline which involves 3 stages like R-CNN. And unlike R-CNN, the fine-tuning algorithm proposed in [50] cannot update the convolutional layers, which limits the accuracy of very deep networks.

In order to fix the disadvantages of R-CNN and SPP-net, Girshick et al. [54] proposed the method of Fast R-CNN in 2015. Figure 5.3 presents the architecture of Fast R-CNN. There are 2 main differences we can find between the architecture of Fast R-CNN and R-CNN: (1) Fast R-CNN adds the region of interest (ROI) pooling layer before fully connected layers, using max pooling to down-sample each proposal to a small feature map of 7×7 , which is similar to the spatial pyramid pooling layer in SPP-net; (2) Fast R-CNN applies softmax classifier to take place of SVM, and uses a multi-task loss function which jointly optimizes a softmax classifier and bounding-box regressors, making it a single-stage for the training step (except for the region proposal part. For Fast R-CNN, region proposal still needs to be done for images before inputting into the network structure).

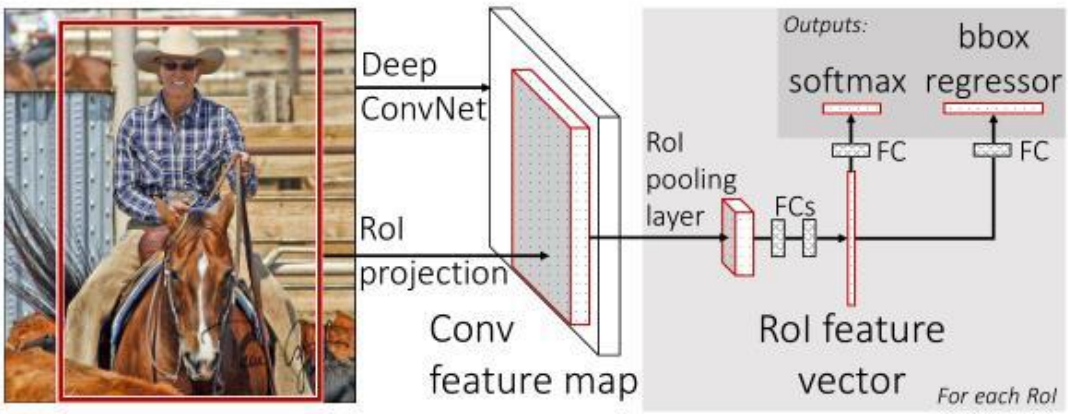


Figure 5.3. Fast R-CNN architecture [54].

However, Fast-RCNN is still not fast enough for the real-time application. The region proposal in Fast-RCNN still uses selective search algorithm, which takes up 90% of the processing time. Now, proposals are the computational bottleneck in state-of-the-art detection systems. Ren et al. [55] proposed Faster R-CNN (See Figure 5.4) in which the selective search has been replaced by a Region Proposal network (RPN). An RPN is a fully-convolutional network that simultaneously predicts object bounds and objectness scores (measures membership to a set of object classes or background) at each position. The advantage of RPN is the region proposal step can be nearly cost-free by sharing convolutional features with Fast R-CNN. RPN can be trained end-to-end to generate high-quality region proposals.

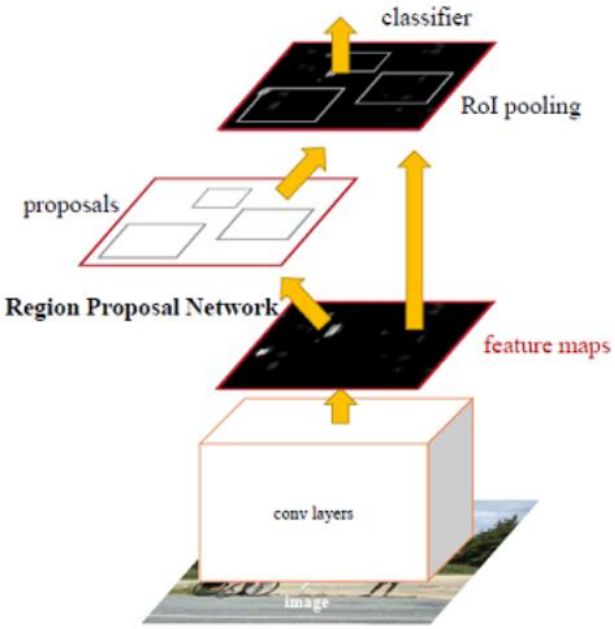


Figure 5.4. Faster R-CNN architecture [55].

The working process of RPN is demonstrated in Figure 5.5. The RPN uses a spatial window to slide feature map output by the last shared convolutional layer, which takes an image feature map as input and outputs *k* rectangular object proposals (coordinates), each with an objectness score (objects or background), where *k* denotes the number of anchor boxes. Each sliding window is mapped to a lower-dimensional vector and this vector is fed into two sibling fully-connected layers: a box-regression (reg) layer and a box-classification (cls) layer. Therefore, the reg layer has 4*k* outputs encoding the coordinates of *k* boxes, and the cls layer outputs 2*k* scores that estimate the probability of object or not object for each proposal.

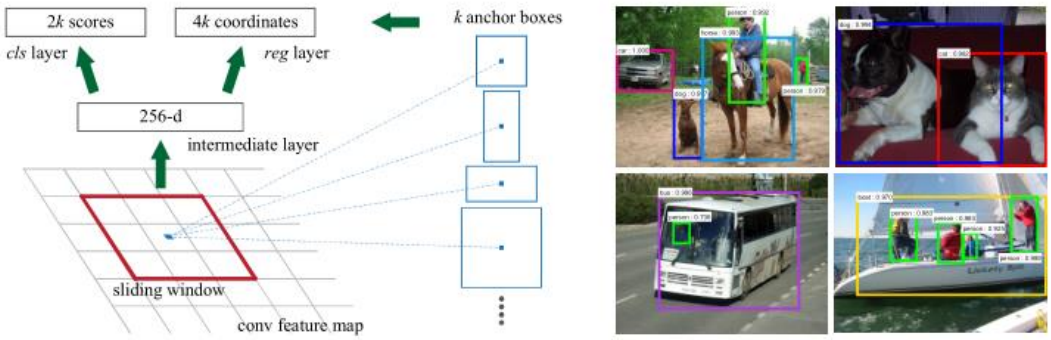


Figure 5.5. Region Proposal Network (RPN). Left: Architecture of RPN. Right: Example detections using RPN proposals on PASCAL VOC 2007 test [55].

Faster RCNN is currently the dominant method for the task of object detection. However, the speed of Faster RCNN is still slow for real-time applications. Redmon et al. [56] proposed a You Only Look Once (YOLO) object detection approach in 2016. The YOLO Detection System is shown in Figure 5.6. First, the input image is resized to 448 × 448; Then runs a single CNN on the image; And the results are obtained by thresholding the model's confidence.

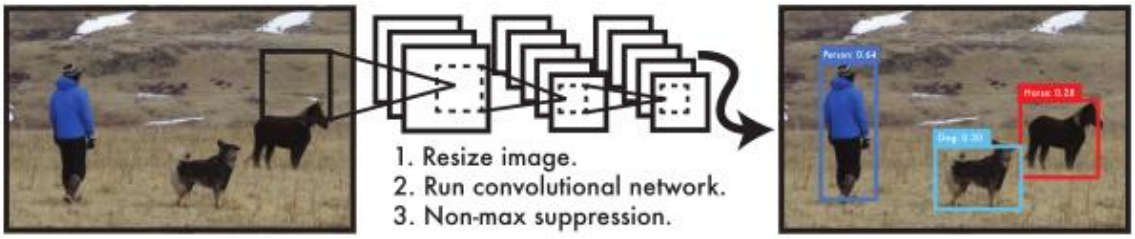


Figure 5.6. The YOLO Detection System [55].

Figure 5.7 describes the prediction method of YOLO. It divides the image into an

$S * S$ grid and predicts B bounding boxes and C conditional class probabilities for each grid cell. These predictions are encoded as an $S \times S \times B \times 5 + C$ tensor. Grid cells are in charge of detecting the object of which center falls into the grid cell. At test time they multiply the conditional class probabilities and the individual box confidence predictions to get the class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

$$Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth} \quad (6.1)$$

In their paper, they applied $S = 7$, $B = 2$. $C = 20$ for evaluating YOLO on PASCAL VOC2012, which reached to 45 frames per second on Titan-X GPU, and achieved mean Average Precision (mAP) of 63.4%.

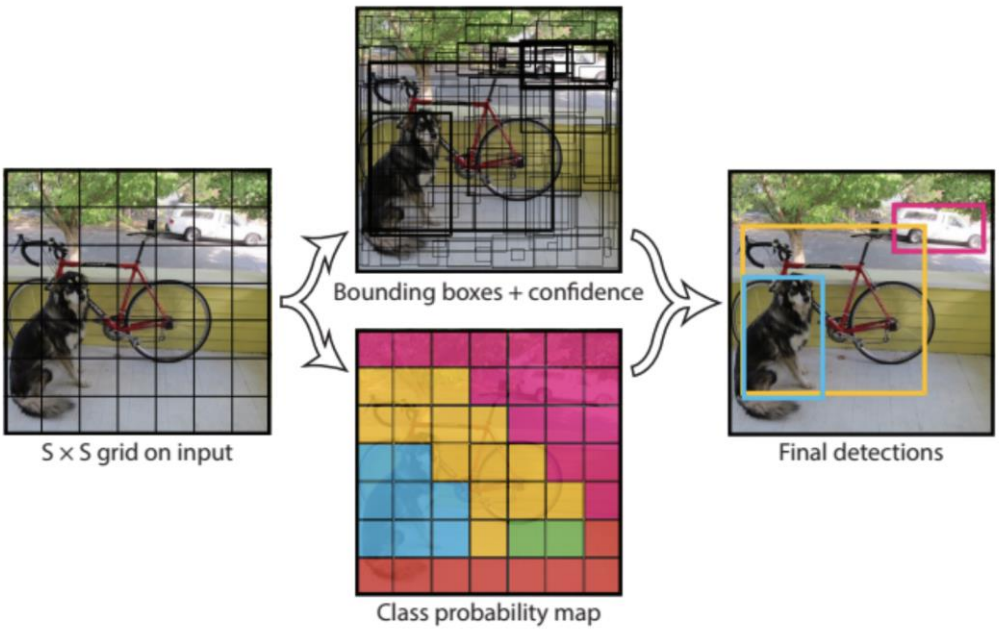


Figure 5.7. The You Only Look Once Model [56].

Although YOLO takes significant speed advantages over Faster-RCNN, YOLO suffers from a significant number of localization errors when compared with Faster-RCNN. And YOLO has relatively low recall compared to region proposal-based methods. Inspired by the improvement of Faster-RCNN via the anchor proposal, Redmon et al. [57] proposed YOLOv2, which is an improved YOLO method. There are mainly 6 improvements from YOLO: (1) YOLOv2 adds batch normalization on all of the convolutional layers; (2) YOLOv2 finetunes the classifier with full resolution of

448×448 for 10 epochs instead of 224×224 in YOLO; (3) Instead of predicting coordinates directly after fully connected layers in YOLO, YOLOv2 removes all of the fully connected layers, and adopts anchor boxes to predict bounding boxes; (4) While Faster R-CNN chooses dimensions of anchor boxes by hand, YOLOv2 applies k-means clustering to find good priors. Along with this, YOLOv2 improves the model instability of anchor by directly predicting the bounding box center location; (5) Unlike Faster R-CNN running proposal networks at various feature maps in the network to get a range of resolution, YOLOv2 designs the passthrough layer to concatenate the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations; (6)YOLOv2 changes input image size every few iterations in training to predict well across a variety of input dimensions.

Another method of object detection with a single deep neural network, named SSD [58] was proposed in 2016. Since YOLO only operates on a single scale feature map, the performance of detecting small objects is not very well. In order to improve this problem, SSD adds convolutional feature layers to the end of the truncated base network. These layers decrease in size progressively and allow predictions of detections at multiple scales, meaning that small objects and big objects are expected to be detected in different layers. A comparison between YOLO and SSD is shown in Figure 5.8. We can see that the Conv5_3 layer is responsible to detect the smallest objects while the Conv11_2 layer is responsible for the biggest objects.

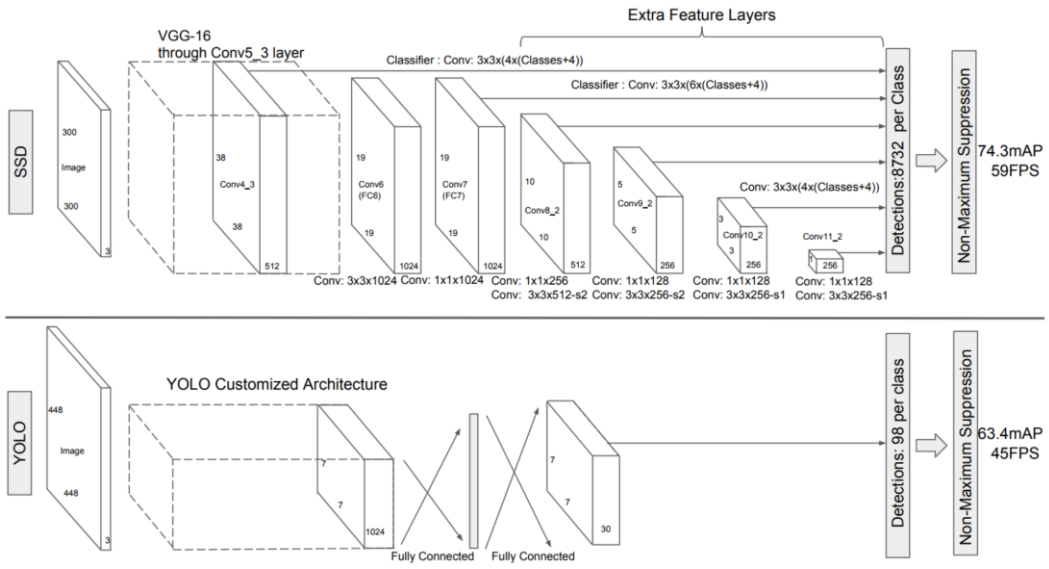


Figure 5.8. A comparison between the model of SSD and YOLO [57].

While the speed (Frames Per Second, FPS) of YOLO and YOLOv2 are faster than SSD with a near size of input images, the accuracy of YOLO-448 (448 is the size of input image) and YOLOv2-416 are both lower than SSD-300 according to experiment results. As a tradeoff of speed and accuracy, YOLOv3 [59] makes improvements on the accuracy with a little increasement in time. There are mainly 3 changes from YOLOv2 to YOLOv3: (1) YOLOv3 applies multilabel classification to predict the class of each box, and they replace softmax by independent logistic classifiers. This helps the detection of overlapping labels; (2) Inspired by feature pyramid networks (FPN) [60], YOLOv3 predicts boxes across 3 different scales, which makes the model more robust to small objects; (3) YOLOv3 designs a new network Darknet-53 for performing feature extraction, which is a hybrid approach between the network used in YOLOv2, Darknet-19, and that newfangled residual network [38]. The architecture of Darknet-53 showed in Figure 5.9, uses successive 3×3 and 1×1 convolutional layers with some shortcut connections as well. This new network is much more powerful than Darknet-19 and more efficient than ResNet-101 or ResNet-152 (While the accuracy is near). Figure 5.10 demonstrated a comparison of several significant detection models on time and accuracy. We can see YOLOv3 is a balanced model for the task of objects detection.

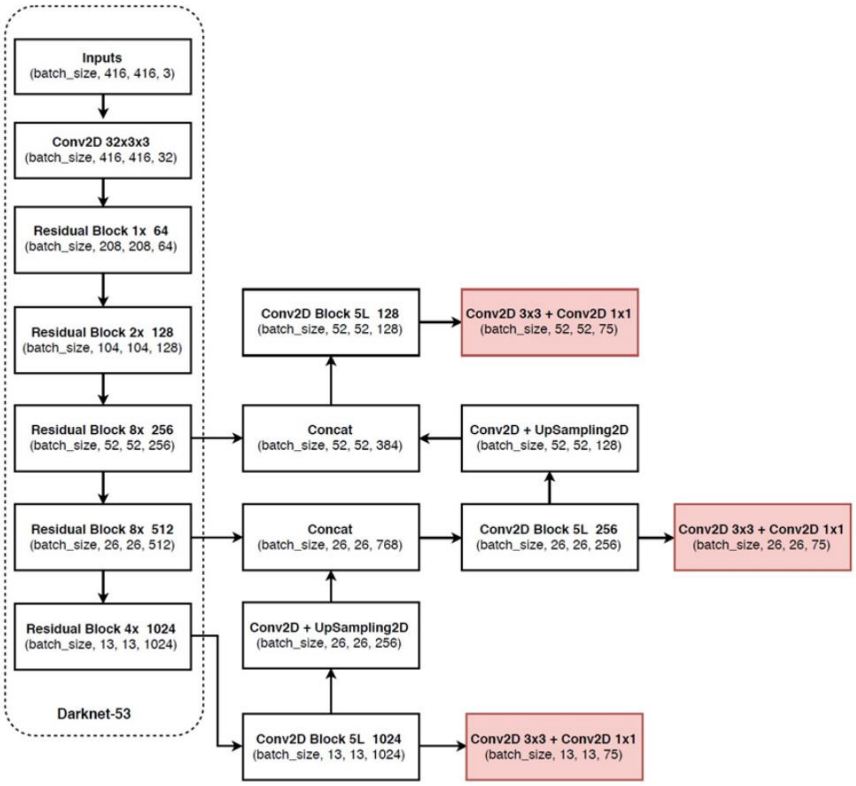


Figure 5.9. The architecture of Darknet-53 [59].

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20

Figure 5.10. A Comparison of significant detection models on time and the mAP at .5 IOU metric [59].

5.1.2. Our Method and Experimental Results

Since road lines are continuous by frames, we apply the segmentation network architecture to extract them instead of the detection network, which will be described in the next section. In our road facilities understanding, we acquire semantic regulations of traffic lights, signs and road markings (except road lines) based on the architecture of YOLOv3. We categorize these detection objects into 49 classes including 4 kinds of traffic lights (red, green, yellow and unknown states), 12 kinds of road markings (including crosswalks, stop lines, direction arrows, speed limitation and no U-turn markings), and 33 kinds of traffic signs (can mainly categorized as speed limitation signs, turning signs, restriction signs, instruction signs, warning signs and other signs).

Since the size of detection object differs a lot in traffic scene, and there would be both small and large objects in the same image, we applied multiscale feature maps for detection, where shallow layers work for small size objects (e.g. traffic lights) detection

and deeper layers work for large size objects (e.g. crosswalks) detection. And we also disabled flip data augmentation for distinguish left and right objects as separate classes (e.g. left and right turn on road signs).

We labeled 4684 on-board camera images derived from Hitotsubashi and Shinjuku intersections in Tokyo. 3607 images are used as training data, and 1077 images as test data. Darknet-53 is used as backbone for training and we trained our model for 100,000 iterations with the input image size of 608×608.

To evaluate our detection results, we follow the PAS-CAL [61] protocol and average by summing in 10% recall steps. The process for computing average precision is given in algorithm 5.1. Only when the overlap region with ground truth above 50% will it be considered as a correct detection.

Algorithm 5.1: Average Precision

```
average precision = 0
for i=0 to 1 step 0.1 do
    p = max(precision(recall>=i))
    average precision = average precision + p / 11
end for
return average precision
```

As our final goal of road facilities detection in image is to integrate their semantic regulation with point cloud map data. And the same target would appear in several sequential frames. Therefore, the semantic regulation of a road facility can be projected to the map as long as it appears at least one time in frames. So, we also take the occurrence probability as a criterion for evaluation, where we count the real number of each object as ground truth, and consider it correct if one object is correct detected at least one time in several frames. Table 5.1 gives the evaluation of our detection results and some testing results are demonstrated in Figure 5.11.

Table 5.1. Evaluation of our road facility detection results

Road Facility Detection	Traffic Light	Road Marking	Traffic Sign	All
Average Precision (%)	90.7	84.9	86.5	86.5
Occurrence Probability (%)	97.0	96.3	96.7	96.6

Figure 5.11. Road facilities detection results in our dataset.

5.2. Road Line Detection

Traditional lane detection methods [62, 63] rely on a combination of highly-specialized, hand-crafted and heuristics features to identify lane segments, which are computationally expensive and prone to scalability due to road scene variations. Neven et al. [64] proposes an instance segmentation network named LaneNet to train for pixel-wise lane segmentation. Figure 5.12 shows the architecture of LaneNet. As we can see from the figure that LaneNet is branched, multi-task network, containing a lane segmentation branch which is the top one and a lane embedding branch which is the bottom one. The lane segmentation branch aims to output a binary segmentation map in which pixels are classified as lane or background. And the embedding branch further disentangles the segmented lane pixels into different lane instances. By means of the multi-task architecture, LaneNet can segment different road lines as different instances without labeling them as different classes. After the pixel embeddings, a clustering loss function is trained to assign id for each lane which is outputted by lane segmentation branch.

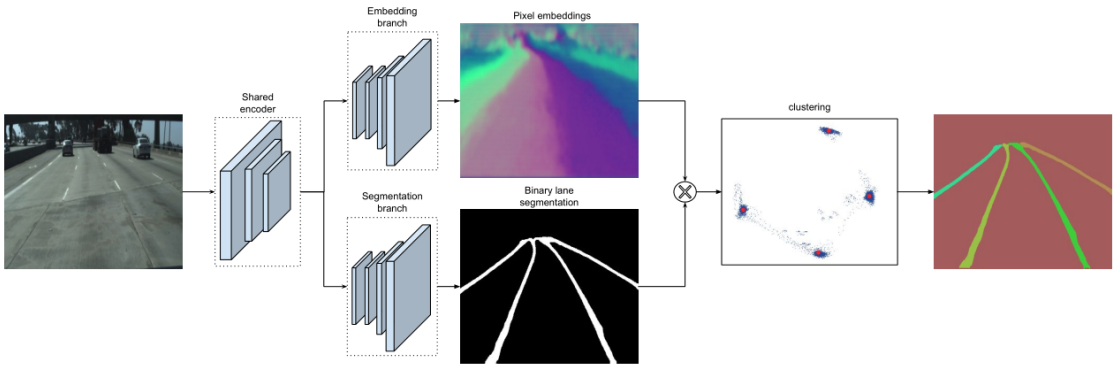


Figure 5.12. LaneNet architecture [64].

We labeled 2342 image frames in our dataset, in which 1208 frames (together with 3626 frames from tuSimple lane dataset [65]) for training, 1134 frames for testing. The training epochs of our experiment are 200,000. We obey the evaluation methodology of tuSimple lane dataset to evaluate our result, which calculates accuracy (acc), false positive (FP), and false negative (FN). The accuracy is calculated by average correct number of points per image:

$$acc = \sum_{img} \frac{C_{img}}{S_{img}} \tag{6.2}$$

where C_{img} is the number of correct points in the image, and S_{img} means the number of ground-truth points in the image. And the false positive and false negative scores are calculated by:

$$FP = \frac{F_{pred}}{N_{pred}} \tag{6.3}$$

$$FN = \frac{M_{pred}}{N_{gt}} \tag{6.4}$$

where F_{pred} is the number of wrongly predicted lanes, N_{pred} is the number of predicted lanes, M_{pred} is the number of missed ground-truth lanes and N_{gt} is the number of all ground-truth lanes. The evaluation of our road line detection results is shown in Table 5.2, and some testing results in our dataset are given in Figure 5.13.

Table 5.2. Evaluation of our road line detection results

acc	FP	FN
90.2%	6.4%	8.2%

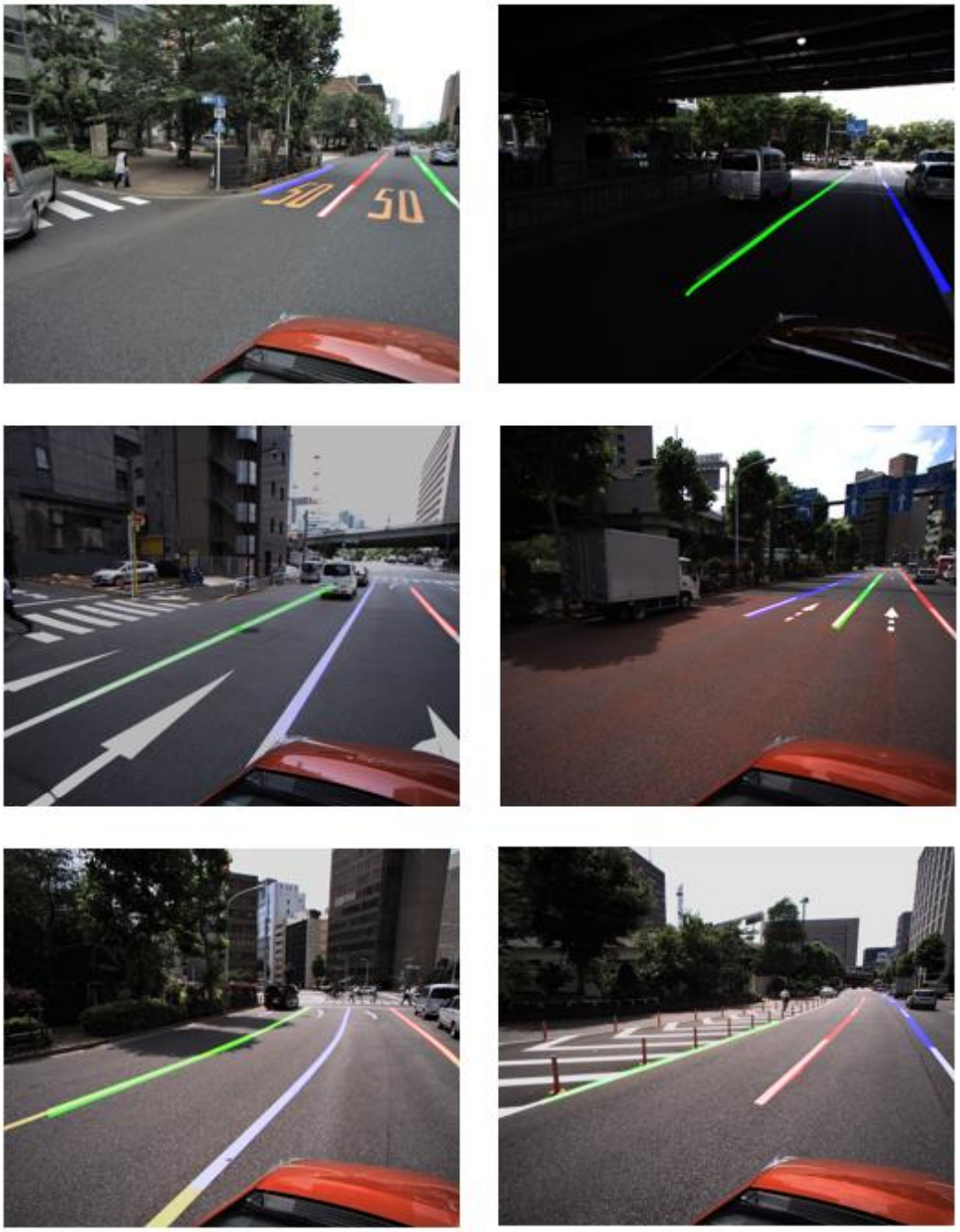


Figure 5.13. Road line detection results in our dataset.

Chapter 6.

Integration of Point Cloud and Image for Mapping

6.1. Camera Model and Coordinate Transforming

In order to matching the semantic information derived from images on the point cloud map, we need to find the coordinate transformation from image pixel coordinates to world coordinates. Therefore, we will make an introduction of the geometry of a pinhole camera model and how to match the coordinates in this section. Photometric cameras using an optical lens can be modelled as a pinhole camera. Figure 6.1 shows the geometry of a pinhole camera model and Figure 6.2 displays the process of projecting a feature in the world coordinates into the pixel coordinates.

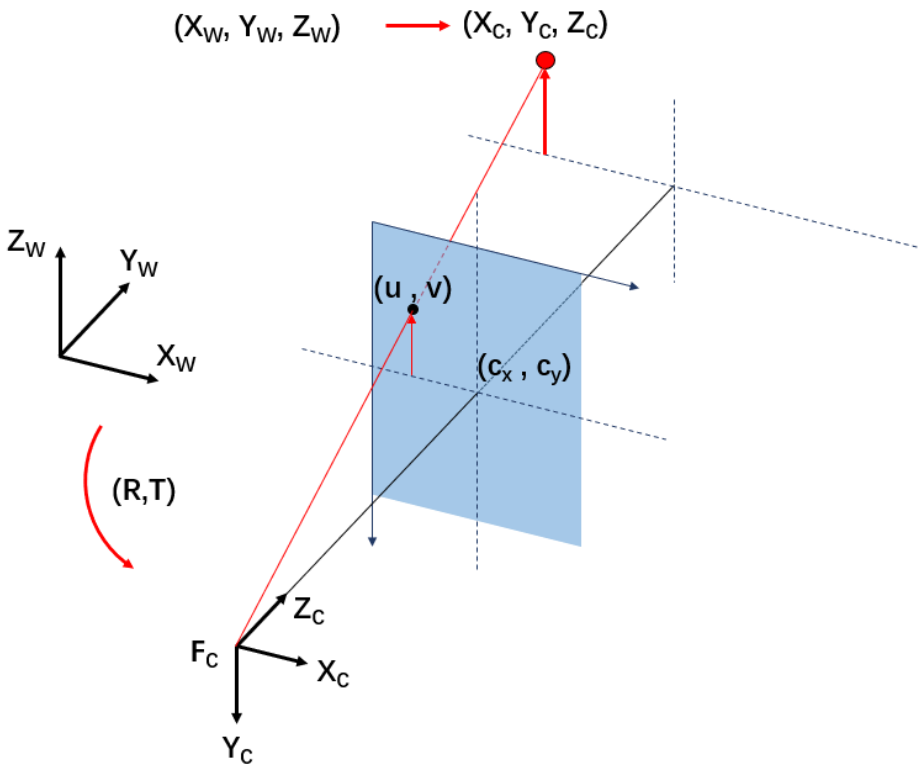


Figure 6.1. The geometry of a pinhole camera model.

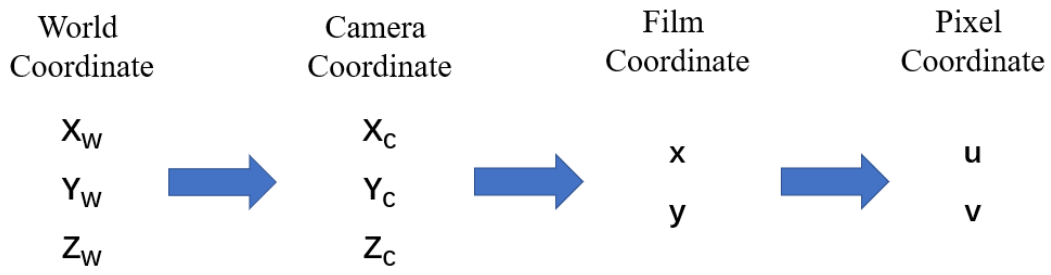


Figure 6.2. Process of coordinate transforming between world coordinates and pixel coordinates.

As we can see in Figure 6.2, since the camera moves in real world, the first step is to project a point in the real world to the camera film by the rotation and translation of the camera. The mathematical form is:

$$\begin{pmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix} \tag{6.1}$$

where

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & 1 \end{pmatrix} \tag{6.2}$$

is the rotation matrix. And

$$T = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{6.3}$$

is the translation matrix. The joint matrix $(R|T)$ is called camera extrinsic parameters. The camera extrinsic parameters are used to describe the camera motion around a static scene.

Then camera coordinates are transformed to the film coordinate by the perspective matrix equation as:

$$Z_C \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{pmatrix} \tag{6.4}$$

where f is the camera focal length in the distance unit.

At last the pixel coordinates comes from the sampling of the CCD sampling.

$$u = \frac{1}{s_x} f \frac{X_C}{Z_C} + c_x = f_x \frac{X_C}{Z_C} + c_x \tag{6.5}$$

$$v = \frac{1}{s_y} f \frac{Y_C}{Z_C} + c_y = f_y \frac{Y_C}{Z_C} + c_y \tag{6.6}$$

where s_x and s_y are the dimension of pixel in frame grabber. The f_x, f_y are the focal lengths expressed in pixel units and (c_x, c_y) is the principal point in the image because normally the principal point in the film coordinates is the upper left corner while in the pixel coordinates the principal point should the center of the image.

Finally, the whole process of the projection can be shown in the equation as:

$$Z_C \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{6.7}$$

where

$$\mathrm{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{6.8}$$

is the camera intrinsic parameters matrix. If an image from the camera is scaled by a factor, all of these parameters should be scaled by the same factor. The matrix of intrinsic parameters does not depend on the scene viewed. So, once estimated, it can be re-used as long as the focal length is fixed

we can see that the range information is lost in this projection, but the angle or orientation of the feature can be obtained if the focal length is known and the lens does not cause distortion. So, a single camera only provides only information about the direction of the features exist in the pedestrian's view, while as it doesn't provide any information regarding the depth. However, our goal is to matching semantic information of traffic lights, traffic signs and road markings from image to the point cloud map. The road surface is always perpendicular to the car's z-axis, meaning that we can matching with road markings only by the x-y plane. And since there would be a distance between two traffic poles (traffic lights, signs), we can also match the them in the x-y plane with a certain range. Since cameras are mounted in a fixed position on the MMS car and we have the intrinsic and extrinsic parameters of the camera, we can

use the Inverse Perspective Mapping (IPM) to transform to transform the pixel coordinates back to the camera coordinates, and get the world coordinates based on the GPS position of the MMS car. Figure 6.3 shows a visualization of coordinate transformation from pixel coordinate to world coordinate.

(a) (b) (c)

Figure 6.3. A Visualization of coordinate transformation from pixel coordinate to world coordinate: (a) original image; (b) image after IPM, which is in camera coordinate; (c) image in world coordinate, where the pink part is road marking extracted from point cloud data.

6.2. Integration of Semantic Information on Map

From above we have talked about how to transform coordinates from pixels in image to the map. In our experiments, we divide our facilities into 5 categories for integration: stop lines, crosswalks, instruction road markings, facility poles (traffic lights and signs), and road lines. Since there are wrong detections for first four categories when the camera is far from the target, which can detect correctly when the target becomes near (as shown in Figure 6.4). To solve this kind of problem, for each object, if it is classified as different categories in several frames, we calculate the probability (percentage) of each category and consider the one with the largest probability as the correct one. Then we obtain the center point of bounding-box in the last correct-detected frame, and transform the center point to world coordinate for integration. The method of integrate these four categories is a little bit different, we will describe in following. And we will also introduce the method of integrate road lines which differs from first four categories.

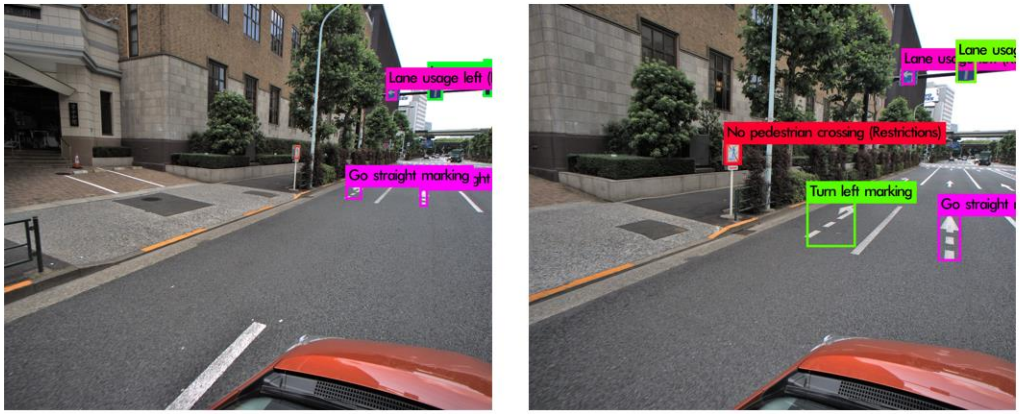


Figure 6.4. Example of wrong detections when car is far from target: (a) the left marking is wrongly detected since it is small in this image; (b) the left marking can be detected correctly when car becomes near.

For stop lines, since they can be considered vertical to the car trajectory when the car is near. Therefore, we transform the center point to world coordinate and fit a straight line along the x-axis. Figure 6.5 displays the process of stop lines integration.

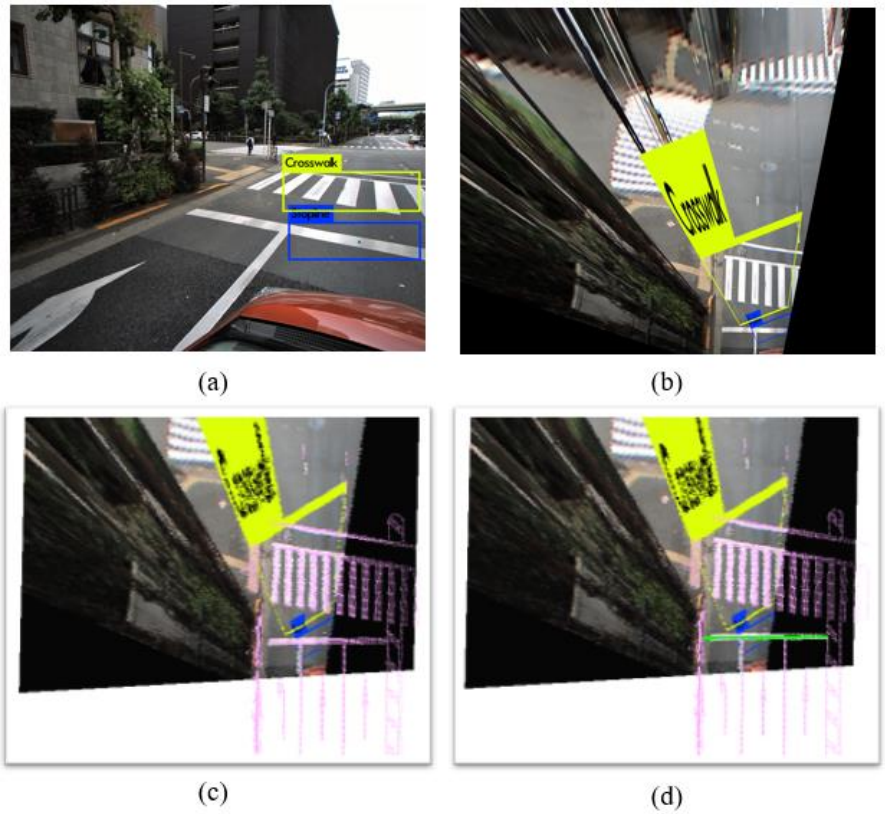


Figure 6.5. Process of stop lines integration, blue point is center point of bounding box: (a) original image; (b) image after IPM; (c) image in world coordinate with point cloud; (d) stop line fitting.

For crosswalk area, since they are usually with a certain range of width and height. Therefore, we generate a square area from the center point to represent a crosswalk area. Figure 6.6 shows the process of the crosswalk area generation.

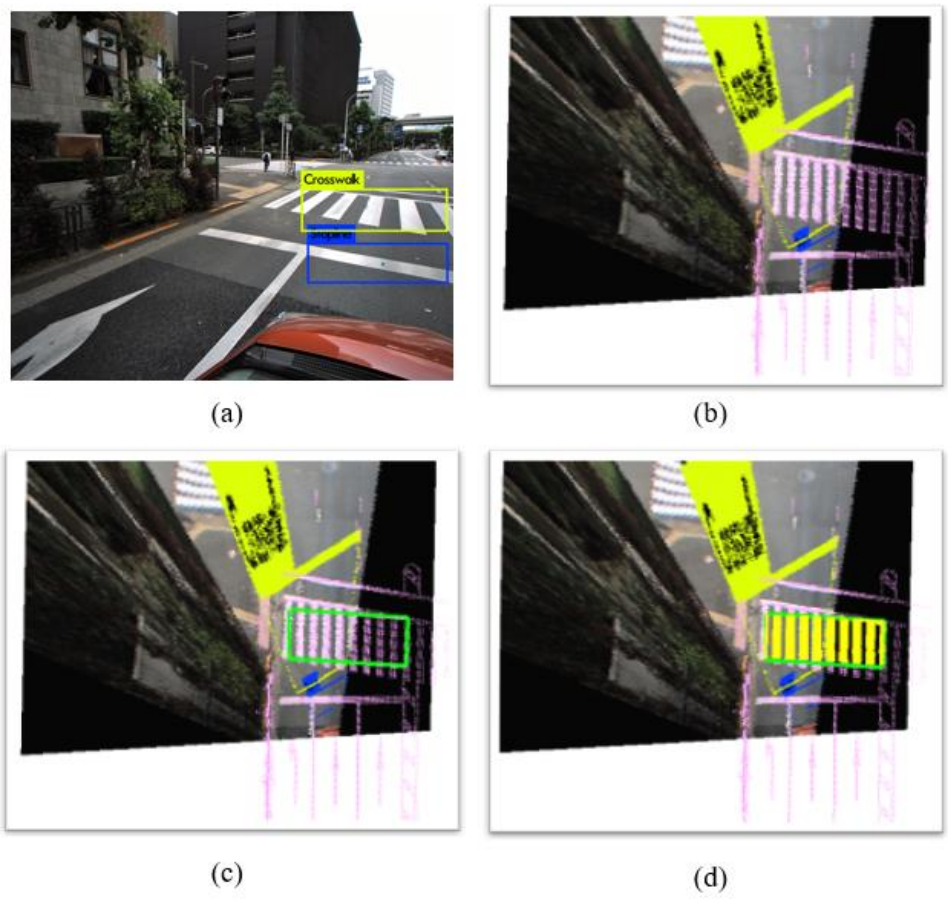
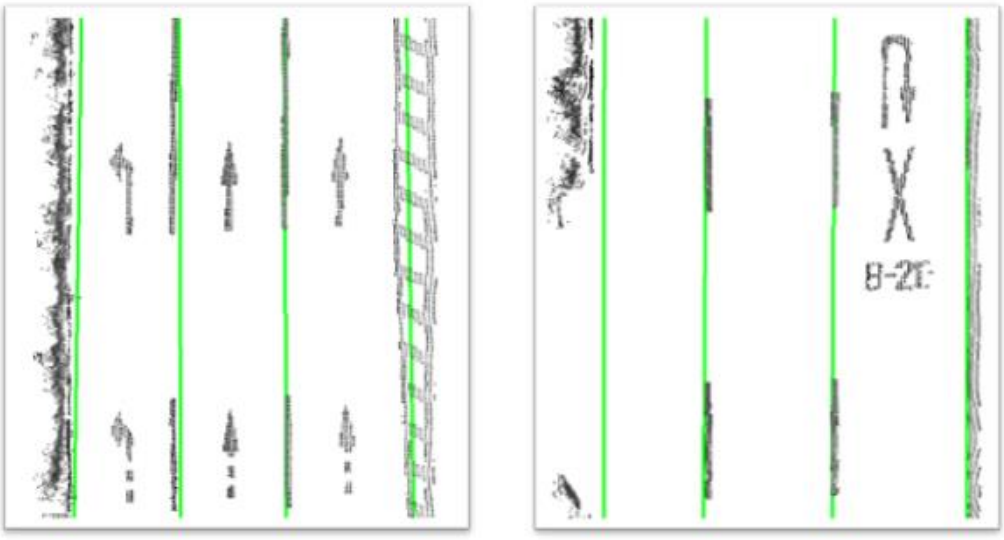


Figure 6.6. Process of crosswalk area generation, blue point is center point of bounding box: (a) original image; (b) image in world coordinate with point cloud; (c) crosswalk area generation; (d) crosswalk area extraction.

For road lines fitting, since the line detection results are pixels frame by frame, we firstly use coordinate transforming to transform pixels of each frame into world coordinate. Then we segment one whole road area part into several segments according to its length (we consider a whole road area part begins and ends with crosswalks, where Figure 6.7 gives an example of a whole road area part). Finally, we apply the Ransac algorithm to fit straight lines in each segmented area. Some road line fitting results are shown Figure 6.8.

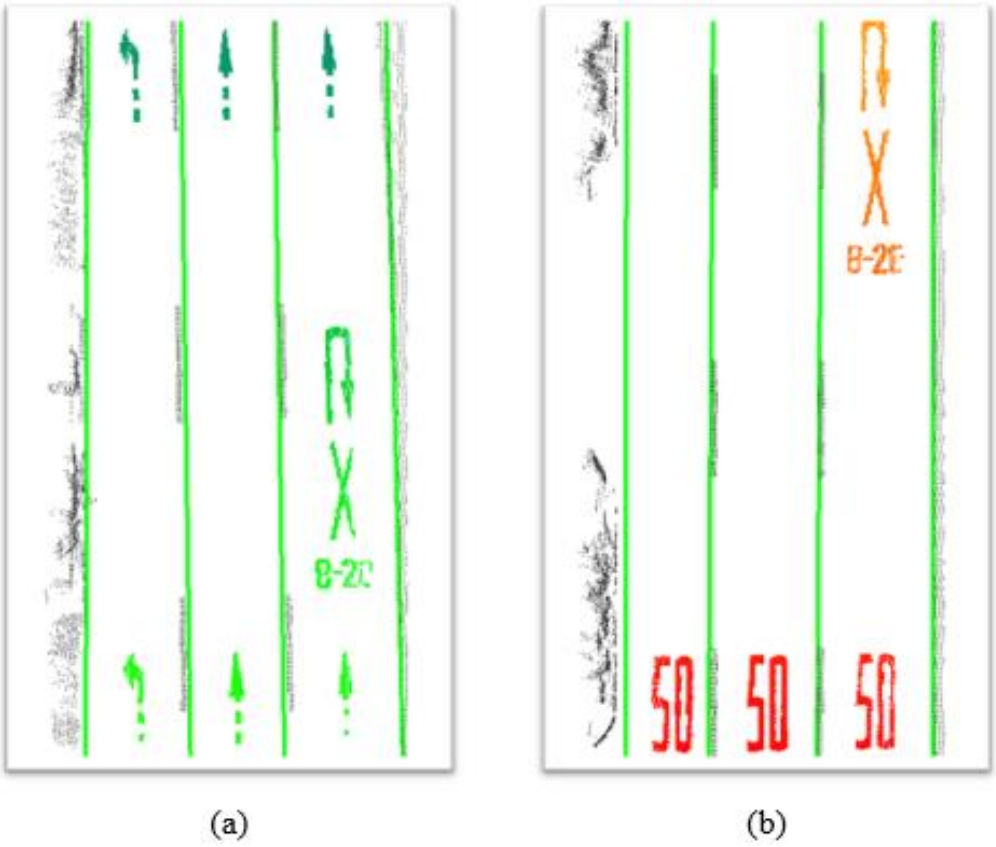
Figure 6.7. Example of a whole road area part.



(a) (b)

Figure 6.8. Some road line fitting results.

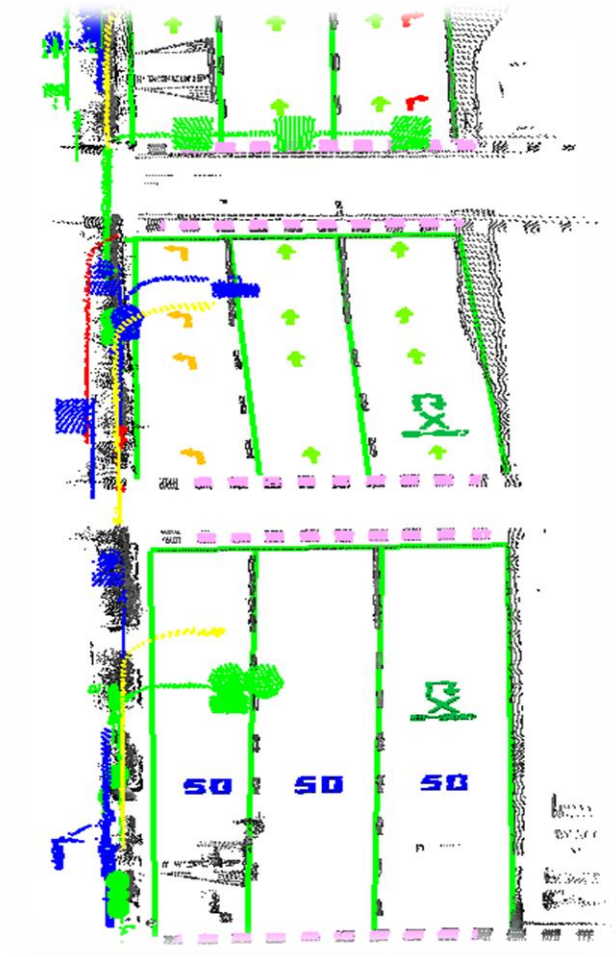
As for instruction road markings, since they are all in road lanes which lie between two road lines. Therefore, we use the information of road lines to restrict the searching area and generate clusters around center points. Figure 6.9 gives integration results of instruction road markings.



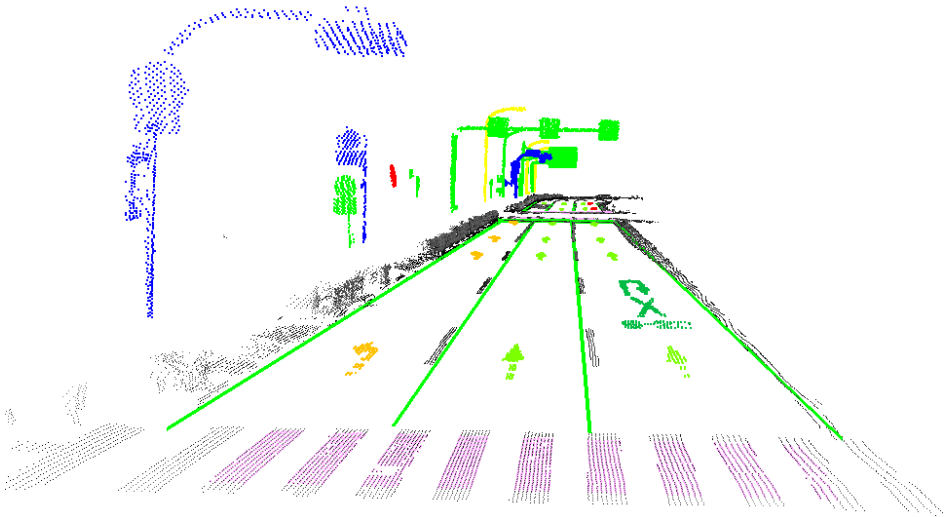
(a) (b)

Figure 6.9. Integration results of instruction road markings, different colors stand for different categories.

After all of the road facilities are added, integration of semantic information on our map are completed. Figure 6.10 demonstrates the final result of our map with semantic information.



(a)



(b)

Figure 6.10. Final result of our map with semantic information.

Chapter 7.

Conclusions

In this thesis, we presented a study on automated mapping of road facilities and their semantic information on HD maps by MMS. Firstly, we utilize point cloud data derived by mobile mapping system to extract road facility poles and road markings, from which we can know the accurate position of them in the map. Then we applied deep-learning model for road facilities detection to obtain traffic regulation information from these facilities. Finally, Integration of information derived from point cloud and image was done by coordinate transforming to construct HD map with semantic information automatically.

For future work, based on derived semantic information, how to generate a lane-level model for division of region with different traffic regulations should be taken into consideration. After traffic regulations are defined on every road of HD map, motion planning of vehicles can be the further step towards higher level autonomous driving. Besides, since extracted traffic poles on the map can provide good features for localization, accurate self-localization of vehicles based on the map can also be a direction for further study.

Reference

[1] Yang, Bin, Ming Liang, and Raquel Urtasun. "HDNET: Exploiting HD Maps for 3D Object Detection." In *Conference on Robot Learning*, pp. 146-155. 2018.

[2] Takada Hiroaki, Sato Kenya. SYSTEMS, CONTROL AND INFORMATION 60(11), pp. 457-462, 2016.

[3] https://www.here.com/products/automotive/hd-maps.

[4] http://proceedings.esri.com/library/userconf/proc16/papers/360_461.pdf.

[5] Joshi, Avdhut, and Michael R. James. "Generation of accurate lane-level maps from coarse prior maps and lidar." IEEE Intelligent Transportation Systems Magazine 7, no. 1 (2015): 19-29.

[6] Carneiro, Raphael V., Rafael C. Nascimento, Rânik Guidolini, Vinicius B. Cardoso, Thiago Oliveira-Santos, Claudine Badue, and Alberto F. De Souza. "Mapping Road Lanes Using Laser Remission and Deep Neural Networks." *arXiv preprint arXiv:1804.10662* (2018).

[7] http://www.mitsubishielectric.com/bu/mms/features/index.html.

[8] http://www.gps.gov/.

[9] Kaplan, Elliott, and Christopher Hegarty. Understanding GPS: principles and applications. Artech house, 2005.

[10] https://velodynelidar.com/hdl-64e.html.

[11] Potó, V., Á. Somogyi, T. Lovas, Á. Barsi, V. Tihanyi, and Z. S. Szalay. "Creating hd map for autonomous vehicles-a pilot study." In 34th international colloquium on advanced manufacturing and repairing technologies in vehicle industry. 2017.

[12] Ellul, C., M. Adjrad, and P. Groves. "The Impact of 3D Data Quality on Improving GNSS Performance Using City Models-Initial Simulations." In ISPRS Annals, vol. 4. International Society for Photogrammetry and Remote Sensing, 2016.

[13] Yoneda, Keisuke, Chenxi Yang, Seiichi Mita, Tsubasa Okuya, and Kenji Muto. "Urban road localization by using multiple layer map matching and l ine segment matching." In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 525-530. IEEE, 2015.

[14] Hosseinyalamdary, S., and M. Peter. "LANE LEVEL LOCALIZATION; USING IMAGES AND HD MAPS TO MITIGATE THE LATERAL ERROR." International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 42 (2017).

[15] Toschi, I., M. M. Ramos, E. Nocerino, F. Menna, F. Remondino, K. Moe, D. Poli, K. Legat, and Francesco Fassi. "Oblique photogrammetry supporting 3D urban reconstruction of complex scenarios." International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 42 (2017): 519-526.

[16] Toledo-Moreo, Rafael, David Bétaille, and François Peyret. "Lane-level integrity provision for navigation and map matching with GNSS, dead reckoning, and enhanced maps." IEEE Transactions on Intelligent Transportation Systems 11, no. 1 (2010): 100.

[17] Paparoditis, Nicolas, Jean-Pierre Papelard, Bertrand Cannelle, Alexandre De vaux, Bahman Soheilian, Nicolas David, and Erwann Houzay. "Stereopolis I I: A multi-purpose and multi-sensor 3D mobile mapping system for street v isualisation and 3D metrology." Revue française de photogrammétrie et de t élédétection 200, no. 1 (2012): 69-79.

[18] Sairam, Nivedita, Sudhagar Nagarajan, and Scott Ornitz. "Development of mobile mapping system for 3d road asset inventory." *Sensors* 16, no. 3 (2016): 367.

[19] Holopainen, Markus, Ville Kankare, Mikko Vastaranta, Xinlian Liang, Yi Lin, Matti Vaaja, Xiaowei Yu et al. "Tree mapping using airborne, terrestrial and mobile laser scanning–A case study in a heterogeneous urban forest." *Urban forestry & urban greening* 12, no. 4 (2013): 546-553.

[20] Jin, Hang, Marc Miska, Edward Chung, Maoxun Li, and Yanming Feng. "Road feature extraction from high resolution aerial images upon rural regions based on multi-resolution image analysis and gabor filters." In *Remote Sensing-Advanced Techniques and Platforms*. InTech, 2012.

[21] Wikipedia, "Biological neural network" https://en.wikipedia.org/wiki/Biological_neural_network.

[22] Hubel, David H., and Torsten N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex." *The Journal of physiology* 195, no. 1 (1968): 215-243.

[23] Wang, Haohan, and Bhiksha Raj. "On the origin of deep learning." *arXiv preprint arXiv:1702.07800* (2017).

[24] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International Conference on Machine

Learning, pp. 448–456, 2015.

[25] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. ICML, vol. 30, 2013.

[26] Wikipedia, "Activation function." https://en.wikipedia.org/wiki/Activation_function.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proceedings ofthe IEEE international conference on computer vision, pp. 1026–1034, 2015.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” nature, vol. 323, no. 6088, pp. 533–536, 1986.

[29] J. Christopher M.Bishop, M.Jordan, Pattern Recognition and Machine Learning. Springer, 2006.

[30] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105, 2012.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., “Imagenet large scale visual recognition challenge,” International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.

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

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9, 2015.

[36] 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, pp. 770–778, 2016.

[37] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700-4708. 2017.

[38] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." *arXiv preprint arXiv:1605.07146* (2016).

[39] Press, P., and David Austin. "Approaches to pole detection using ranged laser data." In Proceedings of Australasian Conference on Robotics and Automation. 2004.

[40] Luo, De-an, and Yan-min Wang. "Rapid extracting pillars by slicing point clouds." In Proc. XXI ISPRS Congress, IAPRS, vol. 37, pp. 215-218. 2008.

[41] Pu, Shi, Martin Rutzinger, George Vosselman, and Sander Oude Elberink. "Recognizing basic structures from mobile laser scanning data for road inventory studies." ISPRS Journal of Photogrammetry and Remote Sensing 66, no. 6 (2011): S28-S39.

[42] Huang, Jing, and Suya You. "Pole-like object detection and classification from urban point clouds." In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 3032-3038. IEEE, 2015.

[43] Yokoyama, Hiroki, Hiroaki Date, Satoshi Kanai, and Hiroshi Takeda. "Detection and classification of pole-like objects from mobile laser scanning data of urban environments." International Journal of Cad/Cam 13, no. 2 (2013): 31-40.

[44] Golovinskiy, Aleksey, Vladimir G. Kim, and Thomas Funkhouser. "Shape-based recognition of 3D point clouds in urban environments." In Computer Vision, 2009 IEEE 12th International Conference on, pp. 2154-2161. IEEE, 2009.

[45] Tombari, Federico, Nicola Fioraio, Tommaso Cavallari, Samuele Salti, Alioscia Petrelli, and Luigi Di Stefano. "Automatic detection of pole-like structures in 3d urban environments." In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pp. 4922-4929. IEEE, 2014.

[46] Lalonde, Jean-François, Nicolas Vandapel, Daniel F. Huber, and Martial Hebert. "Natural terrain classification using three-dimensional ladar data for ground robot mobility." Journal of field robotics 23, no. 10 (2006): 839-861.

[47] Bentley, Jon Louis. A survey of techniques for fixed radius near neighbor searching. No. SLAC-0186. 1975.

[48] Zhang, Wuming, Jianbo Qi, Peng Wan, Hongtao Wang, Donghui Xie, Xiaoyan

Wang, and Guangjian Yan. "An easy-to-use airborne LiDAR data filtering method based on cloth simulation." Remote Sensing 8, no. 6 (2016): 501.

[49] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pp. 886-893. IEEE, 2005.

[50] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.

[51] Uijlings, Jasper RR, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition." International journal of computer vision 104, no. 2 (2013): 154-171.

[52] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37, no. 9 (2015): 1904-1916.

[53] Lazebnik, Svetlana, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories." In null, pp. 2169-2178. IEEE, 2006.

[54] Girshick, Ross. "Fast r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.

[55] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In Advances in neural information processing systems, pp. 91-99. 2015.

[56] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

[57] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." arXiv preprint (2017).

[58] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.

[59] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[60] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In CVPR, vol. 1, no. 2, p. 4. 2017.

[61] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." International journal of computer vision 88, no. 2 (2010): 303-338.

[62] Borkar, Amol, Monson Hayes, and Mark T. Smith. "A novel lane detection system with efficient ground truth generation." IEEE Transactions on Intelligent Transportation Systems 13, no. 1 (2012): 365-374.

[63] Wu, Pei-Chen, Chin-Yu Chang, and Chang Hong Lin. "Lane-mark extraction for automobiles under complex conditions." Pattern Recognition 47, no. 8 (2014): 2756-2767.

[64] Neven, Davy, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. "Towards End-to-End Lane Detection: an Instance Segmentation Approach." arXiv preprint arXiv:1802.05591 (2018).

[65] The tuSimple lane challange, http://benchmark.tusimple.ai/

Thanks

This thesis is under the instruction of Professor Kamijo. He gave me a lot of guidance as an academic advisor to advance this research. Also, researcher Yanlei Gu are the people who gives me a lot of important advice on my research. Ms. Miwa, Professor Kamijo's secretary, also provided huge help to my research life in the lab. I won’t finish my thesis without their meaningful discussions and help.

To the other members in the lab, I want to thank Dailin Li, Yongjie Liu, Qianlong Wang, Ya Wang and Hirotsugu Kitamura when I was new in the laboratory. Mahdi Javanmardi, Ehsan Javanmardi, Jiali Bao, Xiao Wang, I want to thank them for giving me many useful inspirations on my research. I want to thank Huiyang Zhang, Yoshihiko Kamiya, Rin Kawanami, Kazuki Motari, Duyu Chen for accompanying me through my research and making all the time more meaningful time.

Finally, I want to thank my family and friends for their supporting of my research work during the past two years.

Publication List

研究会発表:

[1] Yue Zhang, Ehsan Javanmardi, Mahdi Javanmardi, Yanlei Gu, Shunsuke Kamijo, "Towards ADAS Map: Automatic Extraction and Integration of Semantic Traffic Regulation from MMS Data", IEICE Technical Report, vol. 118, no. 79, ITS2018-2, pp. 7-12, June 12, 2018.