# Human Tracking Technologies for City-scale Trip Estimation

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2020

Yuma YAMADA

# List of Publications

## Related Journal Articles

1. Yuma Yamada, Akira Uchiyama, Akihito Hiromori, Hirozumi Yamaguchi, Teruo Higashino, “Proposal of a Travel Estimation Method Using Control Signal Records in Cellular Networks and Geographical Information” *IPSJ Journal*, vol. 57, no. 8, pp. 1826–1834, August 2016. (in Japanese)

2. Yuma Yamada, Akihito Hiromori, Hirozumi Yamaguchi, Teruo Higashino, “Development of Bus Passenger Counter Using LiDAR Sensors” *IPSJ Journal*, vol. 60, no. 3, pp. 934–944, March 2019. (in Japanese)

## Related Conference Papers

1. Yuma Yamada, Akira Uchiyama, Hiromori Akihito, Hirozumi Yamaguchi, Teruo Higashino, “Travel estimation using control signal records in cellular networks and geographical information” in *Proceedings of 9th IFIP Wireless and Mobile Networking Conference (WMNC’16)*, pp. 138–144, July 2016.

## Other Journal Articles

1. Hikaru Yoshisada, Yuma Yamada, Akihito Hiromori, Hirozumi Yamaguchi, Teruo Higashino, “Relative Position Estimation of Laser Range Scanners in Indoor Environments” *IPSJ Journal*, vol. 59, no. 8, pp. 1485–1498, August 2018. (in Japanese)

## Other Conference Papers

1. Hikaru Yoshisada, Yuma Yamada, Akihito Hiromori, Hirozumi Yamaguchi, Teruo Higashino, “Indoor Map Generation from Multiple Point Clouds” in *Proceedings of 4th IEEE International Conference on Smart Computing (SMARTCOMP 2018)*, pp. 73–80, June 2018.

2. Katsuya Ogura, Yuma Yamada, Shugo Kajita, Hirozumi Yamaguchi, Teruo Higashino, Mineo Takai, “Ground Object Recognition from Aerial Image-based 3D Point Cloud” in *Proceedings of the 11th International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2018)*, October, 2018.

# Abstract

In recent years, people's mobility data by city traffic has become increasingly necessary as primary data for MaaS (Mobility as a Service) for city communities. In MaaS, it is important to understand the specific traffic demand of people at a micro-level and to construct a multimodal transportation system efficiently. Besides, if a specific demand for travel can be grasped, it will contribute to effective measures to realize a safer and more prosperous community (e.g., optimizing the route and operation of existing transportation systems, and activating the use of unprofitable routes by holding events along the route). In major cities such as the Tokyo metropolitan area, a person trip survey (PT survey) is conducted to check people's movements and grasp the actual state of transportation in major cities once every ten years to grasp people's movements. However, the PT survey takes much effort to carry out, and it is difficult to grasp the movements of people that change from moment to moment according to time, day of the week, and season. Furthermore, in local cities, suburbs, and depopulated areas, PT surveys themselves have not been carried out, and grasping human behavior in local cities has become a major issue. In recent years, it is not impossible to conduct PT surveys, including arrivals and departures, using traffic IC card usage history. However, because people move not only by rail but also by various means such as walking, bicycles, cars, etc., it is not enough to grasp the movement of people in the whole city. Besides, especially in regional traffic with deficit routes, IC cards are often not introduced for management reasons, and even if they are introduced, they are usually limited to some routes. Besides, it is impossible to measure passengers who do not have an IC card.

This dissertation presents a technology to grasp trips of people who move automatically between cities without having to install a dedicated app by users. The goal of this dissertation is to use various sensors to grasp the trips of people, such as trains and buses. Specifically, I address this issue by sensing people using laser range sensors, RGB cameras, and CSR logs that are recorded when a smartphone communicates with a base station. In this dissertation, I make the following three primary contributions to embody this idea.

Firstly, I propose two methods for estimating travel from CSRs for two different scenarios according to the availability of cooperative users. The first method estimates travel modes and trajectories from CSRs, focusing on travel by trains and automobiles that are majorities of travel modes. To overcome coarse-grained location information of CSRs, I fully exploit Spatio-temporal features such as average speeds, train timetables, station locations, and geographic information of railway networks and road

networks. However, its performance is degraded in the real urban environment due to complex radio wave propagation caused by many buildings. To overcome this problem, I propose another travel estimation method using CSRs combined with GPS traces provided by cooperative users. Complex radio propagation is directly learned from CSRs and GPS traces of the cooperative users. In this method, I focus on train passengers because the number of travel route candidates is enormous for other travel modes, which requires a huge amount of training data. Nevertheless, I cannot expect enough volume of the training data to generate a radio propagation map. Therefore, it uses the communication interval between the connection with two different BSs during travel as a related feature. In evaluation, I have modeled the complex BS selection pattern as a radio propagation model to reproduce CSRs and integrated the model into the Scenargie network simulator. I conduct a simulation experiment reproducing an area around Shinjuku station. The results show that train passengers are identified with 53% recall, and 87% precision and automobile passengers are identified with 76% recall and 78% precision. It also achieves 67% accuracy for automobile trajectory estimation. The simulation results also show that the method with GPS traces successfully identifies train passengers with 90% recall and 85% precision for data sets with the expected connection interval of 30 seconds.

Secondly, I propose a method for estimating the number of passengers on a route bus. This method uses a laser range sensor (LiDAR; Light Detection and Ranging) and Raspberry Pi to reduce the labor and cost barriers for the introduction and realize high-precision detection. In the proposed method, sensor correction can be performed simply by installing the system at an appropriate location and acquiring background data when the door is open, and when the door is closed. After detecting the opening of the door, the point cloud corresponding to the object surface is extracted using the background subtraction method, and the point cloud corresponding to the human body surface is detected from the point cloud. Based on this, the number of passengers is measured by tracking the point cloud of each passenger and detecting the passage of doors. In evaluation, I conducted experiments with a bus running between two campuses of my university and achieved 5.3% or lower error (the average absolute error was 0.94%) for coming passengers and 7.5% (the average was 1.9%) for leaving ones. The average processing time per frame was 3.4 ms in Raspberry Pi 3 Model B. This has shown the capability of my algorithm for real-time measurement.

Thirdly, the LiDAR-based method mentioned above can measure the number of passengers at each stop, but cannot track each passenger's on/off locations. I call this "origin-destination (OD)" of the passenger, which is informative for improvement of route planning. Therefore, I propose a method for estimating the OD of bus passengers. This method uses an RGB camera to estimate the OD of passengers, which could not be achieved by using a LiDAR. Specifically, by recognizing passengers in the car during operation between each stop, and comparing passengers in the car before and after the stop, passengers in the car that got off at that stop and new passengers who got on that stop to understand. This makes it possible to measure the OD of each passenger. In order to evaluate the effectiveness of the proposed method, the performance was evaluated using camera images taken in an environment that reproduced an actual route bus. As a result of creating 50 kinds of learning models and evaluating the performance using Raspberry Pi 3, the best model achieved AUC 0.854, processing time 0.70s and accuracy comparable to conventional face image identification system. It was shown that it could be processed at high speed.

Through these contributions, it will be shown that my methods can grasp the movement of people moving between cities. This dissertation has established a method for grasping trips of people moving on trains, buses, cars, etc. without requiring users to install dedicated apps.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years, people's mobility data by local transportation has become increasingly necessary as basic data for MaaS (Mobility as a Service) for local communities. In MaaS, it is important to understand the specific traffic demand of people at a micro-level and to construct a multimodal transportation system efficiently. Besides, if a specific demand for travel can be grasped, it will contribute to effective measures to realize a safer and more prosperous community (e.g., optimizing the route and operation of existing transportation systems, and activating the use of unprofitable routes by holding events along the route). In major cities such as the Tokyo metropolitan area, a person trip survey (PT survey) [3] is conducted to check people's movements and grasp the actual state of transportation in major cities once every ten years to grasp people's movements. However, the PT survey takes much effort to carry out, and it is difficult to grasp the movements of people that change from moment to moment according to time, day of the week, and season. Furthermore, in local cities, suburbs, and depopulated areas, PT surveys themselves have not been carried out, and grasping human behavior in local cities has become a major issue. In particular, it is extremely important to understand the use of route buses, which are the main axis of public transport, in all regions such as local cities, in order to understand the use of local public transport and to realize a correct improvement process. The number of passengers getting on and off the route bus is valuable data for transportation operators who cannot accurately grasp the number of passengers. Besides, for example, a large number of passengers getting off at one time on crowded routes, the stopping time at the stop increases, many tourists are confused by cash usage and payment methods, and the getting off time per person increases. This phenomenon can be grasped by measuring and analyzing the time required for getting on and off. Such data can be an indicator of whether the smooth operation has been realized, and can be used as the basic data for business operators to make improvements.

In recent years, it is not impossible to conduct PT surveys, including arrivals and departures, using traffic IC card usage history. However, because people move not only by rail but also by various means such as walking, bicycles, cars, etc., it is not enough to grasp the movement of people in the whole city. Besides, especially in regional traffic with deficit routes, IC cards are often not introduced for management reasons, and even if they are introduced, they are usually limited to some routes. Besides, it is impossible to measure passengers who do not have an IC card.

Ref. [4] estimates the movements of people from location information collected by the Global Positioning System (GPS). However, each user is required to install an application that sends the user's location periodically, which leads to a small amount in the data volume and may cause bias in users' attributes. Call Detail Record (CDR) [5] is an alternative that has the potential to overcome the problems since CDR is generated on the network operator's side whenever mobile phone users make phone calls. However, the accuracy and frequency of its location information are low and sparse.

On the other hand, the rapid spread of smartphones has spawned a vast amount of mobile communication traffic, leading to a growing potential of Control Signal Records (CSRs) for travel estimation. CSRs are CDRs generated when a mobile phone sends a BS any signal for communication, handover, and location management and contains a timestamp and a BS ID connected by the phone. Since mobile communication occurs more often than phone calls, CSRs are generated more frequently than CDRs, while their location accuracy is still the same as CDRs. Since the location accuracy depends on the BS cell size, it is much lower than Wi-Fi and GPS positioning and is not enough to estimate people's travel. To overcome this problem, Ref. [6] proposes an approach using CSRs and geographic information for estimating trajectories of daily people's behavior. However, it aims at the estimation of daily people's travels, which is different from my goal.

However, the location error of CSR depends on the cell size of the base station. In urban areas where base stations are densely installed, CSR can be used. However, in local cities, the location error of CSR is often several kilometers. In local cities, route buses are the primary means of transportation, and there is a need for a system that measures passengers on route buses. However, when the location error is several kilometers, it is often difficult to estimate the trips. A laser range sensor (LiDAR; Light Detection and Ranging) is a sensor that measures the distance of surrounding objects by irradiating the surrounding area with infrared laser light and receiving reflected waves. Despite being very small, the LiDAR is capable of sensing a wide range with a small error, with a measurement distance of several tens of meters, a scanning angle of about 270 degrees, and a ranging accuracy of several tens of millimeters. Besides, the sensing data obtained from the LiDAR is excellent because it is point cloud information representing the position of surrounding objects, and privacy is taken into consideration. Human tracking is possible by detecting the human body using this LiDAR. The second purpose of this dissertation is to estimate passengers of local city buses using LiDARs.

This dissertation presents a method to grasp trips of people who move automatically between cities without having to install a dedicated app by users. The goal of this dissertation is to use various sensors to grasp the trips of people, such as trains and buses. I define a "trip" as a trip that travels between cities, such as a distance of several kilometers. Specifically, I address this issue by sensing people using LiDARs, RGB cameras, and CSR logs that are recorded when a smartphone communicates with a base station. In this dissertation, I make the following three primary contributions to embody this idea.

Firstly, I propose two methods for estimating travel from CSRs for two different scenarios according to the availability of cooperative users. The first method estimates travel modes and trajectories from CSRs, focusing on travel by trains and automobiles that are majorities of travel modes. To overcome coarse-grained location information of CSRs, I fully exploit Spatio-temporal features such as average speeds, train timetables, station locations, and geographic information of railway networks and road networks. However, its performance is degraded in the real urban environment due to complex radio wave propagation caused by many buildings. To overcome this problem, I propose another travel estimation method using CSRs combined with GPS traces provided by cooperative users. Complex radio propagation is directly learned from CSRs and GPS traces of the cooperative users. In this method, I focus on train passengers because the number of travel route candidates is enormous for other travel modes, which requires a huge amount of training data. Nevertheless, I cannot expect enough volume of the training data to generate a radio propagation map. Therefore, it uses the communication interval between the connection with two different BSs during travel as a related feature. In evaluation, I have modeled the complex BS selection pattern as a radio propagation model to reproduce CSRs and integrated the model into the Scenargie network simulator [7]. I conduct a simulation experiment reproducing an area around Shinjuku station. The results show that train passengers are identified with 53% recall, and 87% precision and automobile passengers are identified with 76% recall and 78% precision. It also achieves 67% accuracy for automobile trajectory estimation. The simulation results also show that the method with GPS traces successfully identifies train passengers with 90% recall and 85% precision for data sets with the expected connection interval of 30 seconds.

Secondly, I propose a method for estimating the number of passengers on a route bus. This method uses a LiDAR and Raspberry Pi to reduce the labor and cost barriers for the introduction and realize high-precision detection. In the proposed method, sensor correction can be performed simply by installing the system at an appropriate location and acquiring background data when the door is open (opened status) and when the door is closed (closed status). It uses the background subtraction method to automatically detect the opening and closing of the route bus door. After detecting the opening of the door, the point cloud corresponding to the object surface is extracted using the background subtraction method, and the point cloud corresponding to the human body surface is detected from the point cloud. Based on this, the number of passengers is measured by tracking the point cloud of each passenger and detecting the passage of doors. Because it automatically detects the opening and closing of the doors, it can simultaneously measure the time and number of passengers required at each stop, and can also determine the arrival time and stop time of each stop, so it can also be used for route bus operation planning be able to. In evaluation, I conducted experiments with a bus running between two campuses of my university and achieved 5.3% or lower error (the average absolute error was 0.94%) for coming passengers and 7.5% (the average was 1.9%) for leaving ones. The average processing time per frame was 3.4 ms in Raspberry Pi 3 Model B. This has shown the capability of my algorithm for real-time measurement.

Thirdly, the LiDAR-based method mentioned above can measure the number of passengers at each stop, but cannot track each passenger's on/off locations. I call this "origin-destination (OD)" of the passenger, which is informative for improvement of route planning. Therefore, I propose a method for estimating the OD of bus passengers. This is a method that uses an RGB camera to estimate the OD of bus passengers, which could not be achieved by using a LiDAR. Specifically, by recognizing passengers in the car during operation between each stop, and comparing passengers in the car before and after the stop, passengers in the car that got off at that stop and new passengers who got on that stop to understand. This makes it possible to measure the OD of each passenger. In order to evaluate the effectiveness of the proposed method, the performance was evaluated using camera images taken in an environment that reproduced an actual route bus. As a result of creating 50 kinds of learning models and evaluating the performance using Raspberry Pi 3, the best model achieved AUC 0.854, processing time 0.70s, and accuracy comparable to conventional face image identification system. It was shown that it could be processed at high speed.

Through these contributions, it will be shown that my methods can grasp the movement of people moving between cities. This dissertation has established a method for grasping trips of people moving on trains, buses, cars, etc. without requiring users to install dedicated apps.

The rest of this dissertation is organized as follows. Chapter 2 reviews related work on person trip survey techniques. Chapter 3 proposes a method for grasping trips of people who move between cities in a multimodal manner using CSR. Chapter 4 proposes a method for measuring the number of passengers on a route bus using LiDARs. Chapter 5 proposes a method for measuring passenger bus OD using RGB cameras. Finally, Chapter 6 summarizes and concludes this dissertation.

# Chapter 2

# Related Work

## 2.1 City Trop Estimation

As a position estimation technique, there is a positioning system GPS using an artificial satellite [8–10]. The GPS estimates the position by determining the position on the earth from the distance from three or more satellites. GPS positioning accuracy is within 10 m in the environment not surrounded by high-rise buildings. It is used in car navigation and mobile phones, etc. There is also research [11] to improve position estimation accuracy itself by GPS. For example, D-GPS [12] takes into consideration the propagation delay caused by atmospheric fluctuations such as the ionosphere and is trying to improve accuracy by using features similarly appearing in nearby receivers. In addition, considering the GPS fingerprint information, a method [12] that can perform highly accurate position estimation even in areas with large position errors such as building streets has been studied.

Besides GPS surveying, there is a positioning method based on fingerprint of Wi-Fi radio information. Ref. [13] proposes a position estimation system Place Lab based on Wi-Fi wireless beacon, and uses location combination with Place Lab, laptop, PDA and mobile terminal. With this method, the accuracy of error of 20 - 30 m is obtained within almost 100% range, and sufficient performance is achieved by position estimation in an outdoor environment. In addition, for the purpose of improving the accuracy of position estimation based on this Wi-Fi information, it is common to use a trajectory estimation technique based on Dead Reckoning [14–16] in many cases. In Ref. [17], the acceleration and the electronic compass installed in a commercially available smartphone are used to estimate the amount of change in direction and direction, and the relative position from the starting point is estimated. In addition, there is a method of estimating the position of the portable terminal from the received signal strength (RSS) transmitted from the mobile phone base station. Ref. [18] proposes a position estimation method using RSS indoors and outdoors, achieving an error within 100 m at a rate of 78% of the whole. Ref. [19] performs position estimation on a 50 m grid by introducing a Cramer-Rao boundary [20] in addition to RSS measurement values.

Floating car data is recently used to estimate the traffic condition on the road network based on speeds, positions, directions of driving cars [21]. Floating car data is collected from driving cars with wireless sensing devices such as mobile phones. For example, Ref. [22] proposed a method to estimate and predict the traffic conditions on the highway loop in Rome by using floating car data collected from over 600,000 cars every 3 minutes. By applying pattern matching based on neural networks, the method successfully predicts average speeds after 30 minutes with 3.5sim9.5 [km/h] error. Ref. [23] also estimates the traffic volumes from floating car data. According to the result, the estimation error is approximately 20% when 40% cars are equipped with devices to collect floating car data. Ref. [24] uses floating car data from taxis to estimate movement of taxi passengers.

Similarly to floating car data, GPS traces of mobile phone users are exploited to estimate people travels in urban areas [4, 5, 25–27]. Ref. [4] accurately estimates movement of evacuees during 1 month after the 2011 Great East Japan earthquake through analysis of over 9.2 billion GPS samples. The relationship between people movement and popular events such as concerts and sport games is also found by analyzing GPS traces in Ref. [25]. Interestingly, Ref. [28] proposed a method to estimate user locations by analyzing twitter texts, focusing on connections between users in the social network.

Compared to the above data sets, CSRs are a huge data set because they are collected from all mobile phone users. Instead, location information of CSRs is coarse (e.g., more than 100m), which is often difficult to pinpoint user locations. Therefore, I need a careful design to overcome the problem.

CDRs and CSRs have an advantage in the data volume since their recording does not need any instruction by users and they are automatically generated on the network operator's side. Ref. [5] predicts locations of users in near future from current phone calls by analyzing the relationship between occurrences of phone calls, locations and time. Ref. [29] succeeded in estimating a train used by a railway passenger from CSRs. However, it relies on a railway passenger identification proposed in Ref. [30] based on machine learning, which means a large volume of training data is necessary. Similarly to my approach, Ref. [6] combines CSRs and geographic information for estimating trajectories of people. To do this, it first estimates user-dependent locations such as a home and an office from CSRs recorded over a long term. Since its goal is to estimate daily habits, the method has difficulty in identifying unexpected events that do not appear in daily CSRs. Different from the previous works, I aim at estimating not only daily travels but also unexpected events such as concerts, sightseeing and shopping.

## 2.2 Passenger Counter on Route Bus

There are many researches to grasp the traffic of people inside and outside using various sensors [31–34]. As a representative study, there are studies using infrared sensors, cameras, and range sensors, and these are introduced below.

### 2.2.1 Passenger Counter Using Infrared Sensor

Several methods have been proposed to measure passersby using inexpensive infrared sensors [35, 36]. Ref. [35] proposes an inexpensive system that combines an infrared sensor and a mat-type pressure sensor to measure the number of passengers and transit time at a security inspection area, which is a typical bottleneck in an airport. Ref. [36] proposes a system in which a pyroelectric infrared sensor is installed at the gate and the number and direction of people passing the gate are measured. In evaluation experiments in which several people pass at intervals of 30 cm, it is possible to detect human traffic with an accuracy of almost 100%, and in evaluation experiments targeting people entering and leaving the lecture room show that the number of people entering and leaving can be measured with an accuracy of 88% or more. In addition, a system for measuring the number of people passing by using an infrared sensor has been commercialized [37]. In this product, two infrared sensors are installed on the wall, and the direction in which people pass is judged from the difference in sensor detection that occurs when a person passes. These methods use a small infrared sensor and can be easily installed in various places, but the detection range of the infrared sensor is limited, so measurement in a wide area is not suitable. In addition, it is difficult to measure getting on and off the bus because it is not possible to distinguish individual people's traffic. For example, in recent years, non-step buses with wide entrances are widely used, and not only situations where each passenger gets on and off one after another, but there are also many cases where multiple passengers get on and off in a row. In addition, there are many situations in which the passenger stops near the entrance due to the acquisition of the numbered ticket and the contact of the IC card, and it is difficult to detect these behaviors with only two sensors. Therefore, passenger detection based on the difference in the traffic detection time between sensors is not always an effective method.

### 2.2.2 Passenger Counter Using Camera

Methods using RGB cameras and stereo cameras have also been proposed to measure passers [38, 39]. In these methods, the number of passers is measured by installing a camera on the ceiling to properly grasp the position of the person in the target area. In Ref. [39], a method is proposed to measure the number of passers based on the video taken by a surveillance camera installed on the ceiling. In this method, pedestrians in the video are detected based on the HOG descriptor, and the direction of movement of the pedestrians is detected by the Kalman filter, and the passenger getting on and off is measured. This method achieves an estimation accuracy of 91% or more. In addition, a passenger count system using an RGB camera has also been commercialized, achieving an estimated accuracy of about 95% under the recommended environment [40]. Similarly, systems using infrared LEDs and stereo cameras have also been commercialized [41]. In these methods, while it is possible to measure the number of passers with high accuracy, it is necessary to install the camera in a position where the target area can be seen, such as installing the camera on the ceiling. Furthermore, because it uses image processing, it has limitations such as the need for a high-performance computer.

### 2.2.3 Face Identification

Individual tracking is possible by identifying individual faces using multiple fixed-point cameras installed indoors and outdoors. It is possible to estimate the passenger getting on/off bus stop by grasping the passengers in the car using the in-vehicle camera of the route bus. In recent years, recognition accuracy has been greatly improved by using deep learning [42–49]. In Ref. [43], using the Generative Adversarial Network (GAN), the discrimination accuracy has reached 93.0%. In Ref. [44], the recognition accuracy has reached 97.9% by using the deep convolutional neural network (DCNN).

However, although these methods are very accurate, the amount of calculation is huge, so the introduction cost per system is very high. Therefore, there is a drawback that it is difficult to install in multiple places when there is a budget restriction, such as in local cities. Therefore, in order to reduce the cost of the system, I propose a face recognition method that operates in real-time on the low-priced Raspberry Pi 3. It can be said that this low-cost computer-operated face recognition system is a new approach that has never existed.

# Chapter 3

# City-scale Trip Estimation Using LTE Control Signal Records

## 3.1 Introduction

In this chapter, I propose two methods for estimating travel from CSRs for two different scenarios according to availability of cooperative users. The first method estimates travel modes and trajectories from CSRs, focusing on travels by trains and automobiles that are majorities of travel modes. To overcome coarse grained location information of CSRs, I fully exploit spatio-temporal features such as average speeds, train timetables, station locations and geographic information of railway networks and road networks. I first estimate stay periods and locations from CSRs to identify travel periods between stay locations. Then, travel modes (i.e., trains, automobiles or others) are estimated assuming average speeds of each travel mode. It estimates speeds of users from CSRs although its accuracy is not high due to coarse location information. Therefore, it also uses typical trajectories that move along with railway lines for train passenger identification. Train passengers are firstly identified since their travels are relatively unique. It estimates the train with the target user (the boarded train) as well as its trajectory according to timetables. Automobile travels are identified from estimated speeds and trajectories which are different from trains. Maximum Likelihood Estimation (MLE) is applied to estimate trajectories and boarded trains. In contrast to Ref. [6], my approach does not assume any daily behavior and therefore is capable of estimating various travels including unexpected events.

However, its performance is degraded in the real urban environment due to complex radio wave propagation caused by many buildings. To overcome this problem, I propose another travel estimation method using CSRs combined with GPS traces provided by cooperative users. Complex radio propagation is directly learned from CSRs and GPS traces of the cooperative users. In this method, I focus on train passengers because the number of the travel route candidates is enormous for other travel modes, which requires a huge amount of training data. Nevertheless, I cannot expect the enough volume of the training data to generate a radio propagation map. Therefore, it uses the communication interval between connection with two different BSs during a travel as a related feature.

For evaluation, I have modeled the complex BS selection pattern as a radio propagation model to reproduce CSRs and integrated the model into Scenargie network simulator [7]. I conduct a simulation experiment reproducing an area around Shinjuku station. The results show that train passengers are identified with 53% recall and 87% precision and automobile passengers are identified with 76% recall and 78% precision. It also achieves 67% accuracy for automobile trajectory estimation. The simulation results also show that the method with GPS traces successfully identifies train passengers with 90% recall and 85% precision for data sets with the expected connection interval of 30 seconds.

## 3.2 Control Signals in Cellular Networks

### 3.2.1 BS selection model

In a cellular network, the locations of cellular phones are managed by Home Location Registers (HLR) according to Signaling System 7 (SS7) [50]. HLRs store location areas where cellular phones exist. Since each location area is constructed by several base stations, the cellular network does not know the exact locations of the cellular phones. However, a part of SS7 contains the location information of the cellular phones as base station IDs. Therefore, Such location information with BS IDs is recorded at the following three situations: (i) a cellular phone establishes a communication channel to its BS, (ii) a user leaves its location area and moves to a different location area, and (iii) a user stays in the same location area for a long time with no communication. Recently, smartphones handle not only phone calls but also many applications, and always communicate in the background because of the applications. Thus, the signals categorized into (i) can be tracked more very frequently, and I can expect to derive the locations of the cellular phones by utilizing the signals. In urban areas, many Wi-Fi networks are available, and I may not rely on cellular networks frequently. However, I can still expect a large volume of network traffic in a cellular network because smartphones automatically switch two different networks according to the quality of the networks. Table 3.1 shows a sample signal data set containing Mobile Station (MS) ID, timestamp and BS ID. The existing works [6, 51] utilized the data in order to discover frequent mobility patterns and profiles. I also assume that the position of each BS is known. In addition, since the cellular phone usually communicates via its nearest BS, I can expect that the users stay near the connected BSs.

Table 3.1: A Control Signal Set

| MS ID | timestamp | BS ID |
|---|---|---|
| 123456 | 2014-04-01 07:00:10 | AABBAA |
| 090909 | 2014-04-01 07:00:14 | DDEEEE |
| 123456 | 2014-04-01 07:01:45 | CCCCCC |
| 90abcd | 2014-04-01 07:02:22 | AABBAA |

In this thesis, I estimate a connected BS for a user based on the following signal strength model because MSs usually connect BSs based on the quality of the radio communication. The received signal strength (RSS) in free space decreases inversely proportional to the square of the distance between the transmitter and the receiver [52]. Thus, the $RSS_i$[W] from the $BS_i$ can be estimated by the following equation, where $P_{t_i}$ is transmission power of $BS_i$ and $d_i$ is distance between $BS_i$ and a user.

$$RSS_i(d_i) \propto P_{t_i} \cdot \frac{1}{d_i^2} \tag{3.1}$$

Therefore, I can represent the probability $P(X = i)$ that $BS_i$ is selected for a user as the following equation.

$$\begin{aligned} P(X = i) &= \frac{RSS_i(d_i)}{\sum_{j=1}^{N} RSS_j(d_j)} \\ &= \frac{P_{t_i} \cdot 1/d_i^2}{\sum_{j=1}^{N} P_{t_j} \cdot 1/d_j^2} \end{aligned} \tag{3.2}$$

The estimated accuracy of the location information depends on the deployment of the location areas. Thus, the position error would be more than 100 m even in urban areas with many base stations.

### 3.2.2 Handover Model

The associated BS is determined according to the received signal strength. However due to radio wave fluctuation, it is not always the nearest BS. In this thesis, I estimate a connected BS for a user based on the following signal strength model. I use the handover model described in Ref. [53]. First, the strongest received signal strength is taken as a reference value. Then, a limit value is defined as a value obtained by subtracting a fixed value determined by each mobile phone operator from the reference value. If the received signal strength of the currently communicating BS falls below this limit value, MS disconnects communication with that BS and connects to the BS having the strongest received signal strength at the time. Therefore, as shown in Fig. 3.1, there may be cases where users at the same location are associated with different BSs. Moreover, although the received signal strength changes according to the distance from the BS in Fig. 3.1, in fact, due to the influence of the building the received signal strength from a distant base station becomes strongest. Therefore, the proposed method considers such complex signal propagation in estimating users' travel.

## 3.3 Proposed Method

### 3.3.1 Travel Estimation Without GPS Traces

#### 3.3.1.1 Overview

As shown in Fig. 3.2, people travel between different stay locations. My method first estimates stay periods from CSR. Then, I identify modes (trains, automobiles or others) of estimated travels $M_x$ from

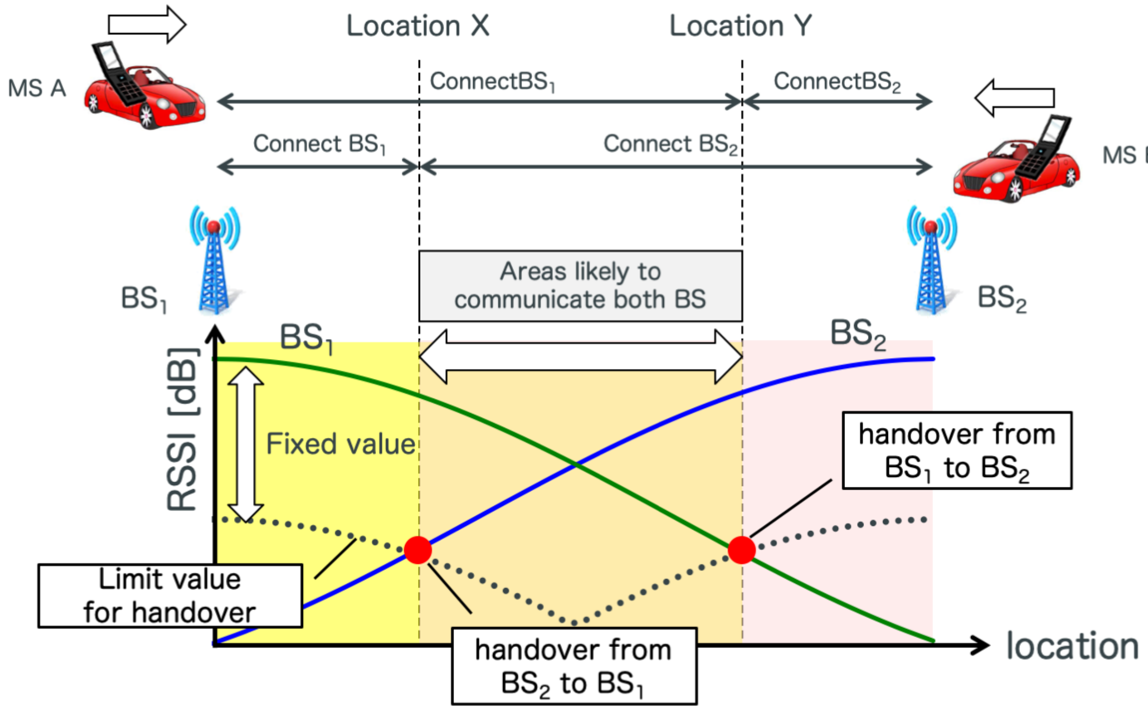


Figure 3.1: Example of handover

stay location $SL_x$ to the next stay location $SL_{x+1}$. I further estimate trajectories of automobiles and boarded trains of railway passengers. In the following sections, I describe the details of the estimation algorithm.

#### 3.3.1.2 Stay Period Estimation

As shown in Fig. 3.3, a user is more likely to be connected to the same BS when the user stays at a location. This means that a stay location is identified if a user connects to the same BS for a long time. However, a user may also be connected to different BSs even when he is at the same location depending on RSS as described in Sec. 3.2.1. Therefore, I identify that a user stays around a BS if the user is connected to the BS frequently in a fixed time window.

I define the whole sequence $P_i$ of CSRs of a user $i$ as shown in Eq. (3.3), where $p_i(x)$ denotes a BS of the $x$-th CSR of $i$.

$$P_i = \{p_i(1), p_i(2), \ldots, p_i(m)\} \tag{3.3}$$

I also let $t_i(x)$ be the timestamp of the $x$-th CSR of $i$. For a sub-sequence $\{p_i(x), \ldots, p_i(y)\}(1 \leq x < y \leq m)$ of $P_i$, my method identifies a stay period if the following condition (3.4) is satisfied.

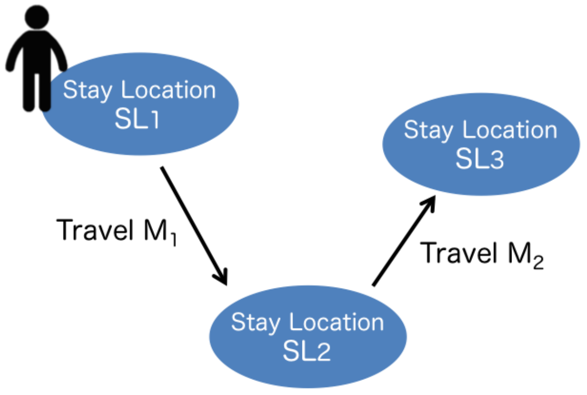


Figure 3.2: Stay locations and travels

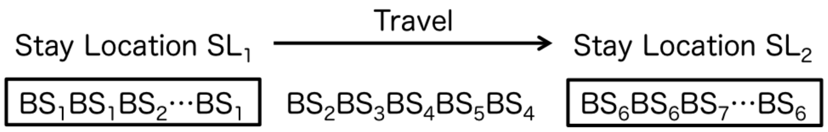


Figure 3.3: BS sequence of travel between $SL_1$ and $SL_2$

$$\begin{aligned} & t_i(y) - t_i(x) > \text{StayTime} \\ & \wedge \frac{\sum_{k=x}^{y} \delta_{p_i(x)p_i(k)}}{y - x + 1} > \text{StayFreq} \end{aligned} \tag{3.4}$$

$$\delta_{ij} = \begin{cases} 1 & (i = j) \\ 0 & (\text{otherwise}) \end{cases} \tag{3.5}$$

Here, $StayTime$ and $StayFreq$ are thresholds for the length of a stay period and the ratio of the same BS in CSRs during a stay, respectively. $\delta_{ij}$ is the Kronecher delta function defined by Eq. (3.5). All the possible sub-sequences of $P_i$ are tested by the Eq. (3.4)[1]. Note that the location of the most frequent BS appeared in the stay period is regarded as the location of the stay.

#### 3.3.1.3 Travel Mode Estimation

I estimate travel modes for each travel $M_x$ identified by Eq. (3.4). Since train travels are relatively unique in terms of trajectories, I start with checking if the travel mode of $M_x$ is train or not. If not, I further check if the mode is automobile or not.

[1]If consecutive stay periods with the same stay location are found, I concatenate those stay periods into a single stay period at the location.

**Train Travel Estimation**

Trains move between stations along with railway lines. Therefore, locations of stations are keys to identify train travels. To do this, my method defines a set $S_i$ of BSs that are likely to be connected at train station $i$. As I mentioned in Sec. 3.2.1, I cannot uniquely define a single BS that a user is connected to at a location. Therefore, $S_i$ is defined as below, where $d(i, BS_j)$ is the Euclidean distance between $i$ and BS $j$. $N_{BS}$ is the total number of BSs in the target area.

$$S_i = \{BS_j | d(i, BS_j) \leq 2 \times \min_{k \in [1, N_{BS}]} d(i, BS_k)\} \tag{3.6}$$

The above equation means that $S_i$ includes BSs whose distances to $i$ are equal to or less than the double distance between $i$ and its nearest BS[2]. Figure 3.4 illustrates an example of BS sets of three stations where $S_1 = \{BS_1, BS_4\}, S_2 = \{BS_2, BS_6\}, S_3 = \{BS_3\}$.

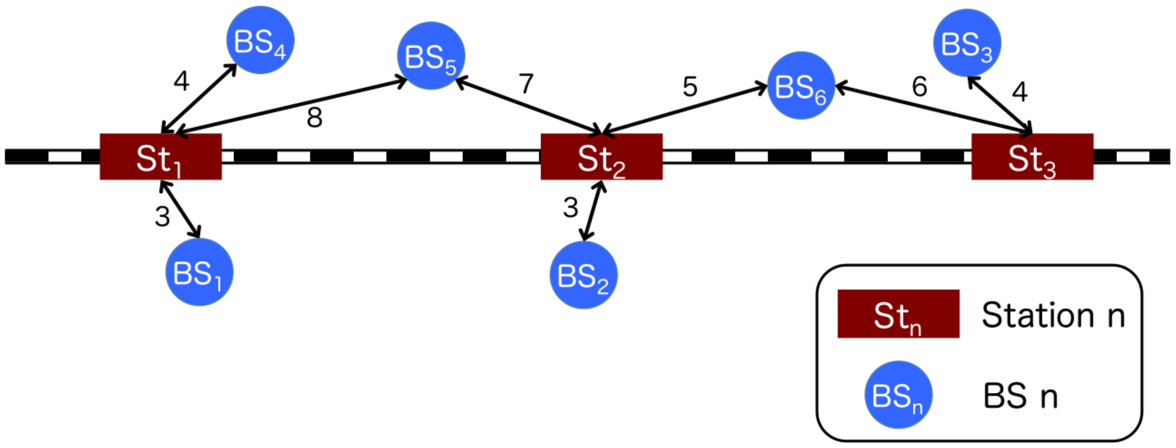


Figure 3.4: Example of BS sets

Finally, I identify train travels focusing on their trajectories that are spatially and temporally unique. I use the following condition (3.7) to represent spatially unique train trajectories, where $(S_{j-1}, S_j)$ and $(S_j, S_{j+1})$ are geographically neighboring stations directly connected by a railway line.

$$\begin{aligned} \exists x \exists y \exists z \big( (p_i(x) \in S_{j-1}) \wedge (p_i(y) \in S_j) \\ \wedge (p_i(z) \in S_{j+1}) \wedge (x < y < z) \big) \end{aligned} \tag{3.7}$$

This condition represents user's consecutive CSRs contain BSs along with a train trajectory.

In addition, train travels should follow the timetables, which is temporally unique. The condition (3.8) below indicates user $i$ moves from station $j-1$ to the station $j+1$ after the next station $j$ within a time period $t(j-1, j+1)$ defined by the timetable.

$$\begin{aligned} \exists x \exists y \big( (p_i(x) \in S_{j-1}) \wedge (p_i(y) \in S_{j+1}) \\ \wedge (t_i(y) - t_i(x) \leq t(j-1, j+1)) \wedge (x < y) \big). \end{aligned} \tag{3.8}$$

[2]If a BS is included in multiple sets, the BS is removed from the sets except the nearest station's set.

Note that I do not rely on directly neighboring stations because their distances are relatively short, leading to location error due to coarse location information of CSRs. Then, my method finally identifies a train travel if it satisfies both conditions (3.7) and (3.8).

**Automobile Travel Estimation**

I identify automobile travels from those which are not identified as train with the assumption that speeds of automobiles are much faster than the other possible travel modes such as walking, running and bicycles. Given road networks representing locations of intersections and their connections, I estimate average speeds of unidentified travels. Assuming a user is at the intersection $IS_x$ nearest to a stay location $SL_x$ during a stay, I regard the shortest distance between $IS_x$ and $IS_{x+1}$ on the road network as a distance of a travel $M_x$. Then, the speed of $M_x$ is calculated as the travel distance divided by the travel time. $M_x$ is identified as automobile if the speed exceeds 30[km/h][3].

#### 3.3.1.4 Boarded Train Estimation

For travels identified as trains, I further estimate trains used in the travels by applying maximum likelihood estimation. I estimate positions of trains at each time according to the timetables assuming a train moves with a constant speed between two stations. Then, I define the likelihood $\mathcal{L}_i(M_x)$ that a train $i$ is used by a travel $M_x$ as below.

$$\mathcal{L}_i(M_x) = \frac{\sum_{t \in TS(M_x)} w_i^t}{\sum_{j=1}^{N_{tr}} w_j^t} \tag{3.9}$$

where $TS(M_x)$ represents a set of timestamps of CSRs in a travel $M_x$ and $N_{tr}$ is the total number of trains operated during $M_x$. The weight $w_i^t$ is defined by either of the following (1)-(4) depending on the distance between BS $j(t)$'s location $l_{j(t)}^t$ of a CSR with a timestamp $t$ and the estimated position $l_i^t$ of train $i$ at $t$.

(1) $w_i^t = 1/d(l_i^t, l_{j(t)}^t)$

(2) $w_i^t = 1/d(l_i^t, l_{j(t)}^t)^2$

(3) $w_i^t = 1/d(l_i^t, l_{j(t)}^t)^3$

(4) $w_i^t = \begin{cases} 1 & (\text{if } d(l_i^t, l_{j(t)}^t) = \min_{k \in [1, N_{tr}]} d(l_k^t, l_{j(t)}^t)) \\ 0 & (\text{otherwise}) \end{cases}$

Intuitively, a likelihood of a train $i$ is higher if $i$'s locations at each time are closer to the BS locations in CSRs. In the evaluation in Sec. 3.4, I compare the above four weighting functions.

[3]The average speed of automobiles in urban areas in Japan is 30sim40[km/h].

#### 3.3.1.5 Automobile Trajectory Estimation

For automobile travels, I estimate their trajectories under the assumption that a user generally prefers choosing the shortest path to the destination on the road network. Given $SL_x$ and $SL_{x+1}$, I define a set $PATH$ of $N_{auto}$ candidates of travel $M_x$'s trajectory as paths between $IS_x$ and $IS_{x+1}$ with lengths which are not longer than the $N_{auto}$-th shortest path. Then, maximum likelihood estimation is applied to determine the trajectory of $M_x$ among the candidates.

The $i$-th candidate path $Path_i$ consists of a sequence of road segments connecting intersections. According to the distance between a road segment $R_k$ and a BS $j$, I define a posterior probability $P(R_k|BS_j)$ that a user is on the road $R_k$ when it is connected to BS $j$. Note that the distance between $R_k$ and BS $j$ is defined as the distance between the center location of $R_k$ and BS $j$'s location. According to Bayes' theorem, $P(R_k|BS_j)$ is defined as:

$$P(R_k|BS_j) = \frac{P(BS_j|R_k)P(R_k)}{\sum_{r\in Path_i} P(BS_j|r)P(r)}. \tag{3.10}$$

$P(BS_j|R_k)$ is the posterior probability that a user is connected to $j$ when he is on $R_k$ and given by the Eq. (3.2). $P(R_k)$ is the prior probability that a user is on $R_k$. I can assign $P(R_k)$ according to prior knowledge such as the amount of car traffic from person trip survey. If no prior knowledge is given, I simply assign all $P(R_k)$ uniform probabilities.

The likelihood of $Path_i$ is calculated as below. Since mobile communication occurs at non-periodic timings, no CSR is generated on some roads in a path. Therefore, I calculate all of valid communication timings regarding CSRs of a travel $M_x$. For each CSR of $M_x$, I assign a road in a path $Path_i$ so that the order of assigned roads completely follows the order of roads in $Path_i$. Let $\pi(p_j(x))$ denote the assignment where a road assigned to CSR $p_j(t)$ is the $\pi(p_j(t))$-th road in the road sequence of $Path_i$. Then, a condition of valid assignments is described as:

$$\forall t_1 \forall t_2 \big(\mathrm{succ}(\pi(p_j(t_1)), \pi(p_j(t_2))) = \mathrm{succ}(t_1, t_2)\big) \tag{3.11}$$

where $\mathrm{succ}(t_1, t_2)$ is an order testing function defined as below.

$$\mathrm{succ}(t_1, t_2) = \begin{cases} 1 & (t_1 \le t_2) \\ 0 & (\text{otherwise}) \end{cases} \tag{3.12}$$

The likelihood $\mathcal{L}_i(a)$ of $Path_i$ with road assignment $a$ is defined as:

$$\begin{aligned} \mathcal{L}_i(a) = &\log_{10} \prod_{t\in TS(M_x)} P(R_{\pi(p_j(t))}|p_j(t)) \\ &+ |TS(M_x)| \log_{10} \frac{\min_{k\in[1,N_{auto}]} |Path_k|}{|Path_i|} \end{aligned} \tag{3.13}$$

where $|TS(M_x)|$ and $|Path_i|$ are the number of CSRs in $M_x$ and the length of $Path_i$, respectively. Note that the second term is a penalty for a longer path.

The likelihood $\mathcal{L}_i$ of $Path_i$ is defined by the average of top-k likelihoods among all valid assignments[4]. Finally, I estimate the trajectory of a automobile travel $M_x$ by choosing $Path_i$ that has the maximum likelihood among $N_{auto}$ candidate paths.

### 3.3.2 Travel Estimation Using GPS Traces

#### 3.3.2.1 Overview

As shown in Fig. 3.5, even train users on the same route may have different BS transition patterns depending on the positions on a train and signal propagation. Moreover, since the communication frequency greatly differs depending on the user, the handover to the adjacent BS is not necessarily limited. Therefore, since the number of combinations of BS transition patterns enormous, a large amount of learning data is needed to learn transition patterns. In this thesis, these problems are solved by estimating the travel based on the "communication interval". Here, "communication interval" is defined as the time from one communication to the other communication in two different communication records of the user.

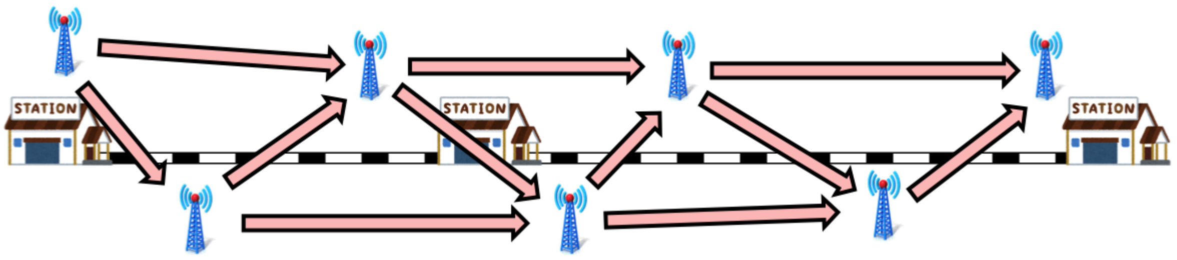


Figure 3.5: Example of BS transition patterns of users moving between stations

I consider training data where the user moved from station $X$ to station $Y$ as shown in Fig. 3.6. From this training data, I can see that during a travel from station $X$ to station $Y$, it communicates with BS C after 200 seconds since the travel started. I can also notice that it communicates with with BS B after 50 seconds since communication with BS A. My method learns the communication interval of these two different BSs and uses it for travel estimation. Then, using the learning result, my method calculates the likelihood of the travel and estimates the train user along with its travel route. In the following sections, I describe detail learning and estimation algorithms.

#### 3.3.2.2 Learning of BS transition patterns

I define the learning table $Tbl$ as the learning result of the communication interval that can occur in two different BSs. $Tbl$ is defined for each travel attribute, and the learning table of the travel from the departure station $i$ to the terminal station $j$ is defined as $Tbl^{(i \to j)}$. In addition, each element

[4] I empirically set k as 5.

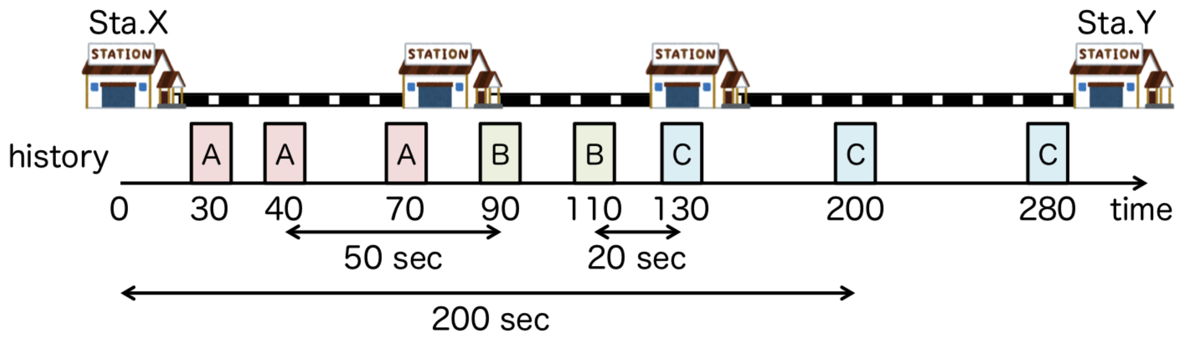


Figure 3.6: Example of training data obtained from GPS user

$Tbl^{(i\to j)}(m,n)$ of the learning table $Tbl^{(i\to j)}$ stores the learning result of the communication interval when communicating with the BS $n$ after the BS $m$. The element $Tbl^{(i\to j)}(m,n)$ represents an interval consisting of the minimum and maximum communication interval as shown in Eq. (3.14).

$$Tbl^{(i\to j)}(m,n) = \left[Tbl^{(i\to j)}_{\min}(m,n),\ Tbl^{(i\to j)}_{\max}(m,n)\right] \tag{3.14}$$

I define that the training user group used for learning is $\mathcal{TU}$, the set of travels of user $u$ is $\mathcal{T}_r^{(u)}$, the departure station and terminal station of travel $T_r$ is $st_{orig}(T_r)$ and $st_{dest}(T_r)$ and departure time and arrival time is $t_{orig}(T_r)$ and $t_{dest}(T_r)$. Then, the training data $\mathcal{TD}^{(i\to j)}$ of the travel attribute from the station $i$ to the station $j$ can be defined as the Eq. (3.15).

$$\mathcal{TD}^{(i\to j)} = \{\mathcal{C}_u^{(t_{orig}(T_r),t_{dest}(T_r))} \mid st_{orig}(T_r) = i \land st_{dest}(T_r) = j,\ T_r \in \mathcal{T}_r^{(u)},\ u \in \mathcal{TU}\} \tag{3.15}$$

where $\mathcal{C}_u$ represents the sequence of the communication history of the user $u$. $\mathcal{C}_u^{(t_1,t_2)}$ represents the sequence of the communication history generated between time $t_1$ and $t_2$ defined as:

$$\mathcal{C}_u^{(t_1,t_2)} = \{c \mid t_1 \le t(c) \le t_2, c \in \mathcal{C}_u\} \tag{3.16}$$

where $t(c)$ represents the time when communication $c$ occurred.

Then, learning table $Tbl^{(i\to j)}$ of travel from station $i$ to station $j$ is learned from training data $\mathcal{TD}^{(i\to j)}$. Based on the two communication tuple $c_1, c_2$ $(t(c_1) < t(c_2))$ of the communication history $\mathcal{C}$ belonging to the given training data $\mathcal{TD}^{(i\to j)}$, it learns the time from communication with $BS(c_1)$ to communication with $BS(c_2)$ and updates the learning table $Tbl^{(i\to j)}(BS(c_1), BS(c_2))$. Therefore, learning table $Tbl^{(i\to j)}(m,n)$ stores the shortest time and the longest time required to communicate with the BS $n$ after communicating with the BS $m$ in the training data. The detail of the learning algorithm is shown in Algorithm 1, where $t(c_1,c_2)$ represents the time from communication $c_1$ occurrence until communication $c_2$ occurrence. Furthermore, it learns the time from the start of movement to communication with the BS $m$ and from communication with BS $m$ to end of movement. These elements of the learning table are represented as $Tbl^{(i\to j)}(O,m)$, $Tbl^{(i\to j)}(m,D)$.

**Algorithm 1** Learning Travels

```
Input: $\mathcal{TD}^{(i\to j)}$
Output: $Mat^{(i\to j)}$
  /* initialization section */
  while $Mat^{(i\to j)}_{\min}(m,n) \in Mat^{(i\to j)}_{\min}$ do
    $Mat^{(i\to j)}_{\min}(m,n) \leftarrow +\infty$
  end while
  while $Mat^{(i\to j)}_{\max}(m,n) \in Mat^{(i\to j)}_{\max}$ do
    $Mat^{(i\to j)}_{\max}(m,n) \leftarrow -\infty$
  end while
  /* learning section */
  while $\mathcal{C} \in \mathcal{TD}^{(i\to j)}$ do
    while $c_1 \in \mathcal{C}$ do
      $m \leftarrow BS(c_1)$
      while $c_2 \in \mathcal{C}$ do
        if $BS(c_1) \neq BS(c_2) \wedge t(c_1,c_2) > 0$ then
          $n \leftarrow BS(c_2)$
          $Mat^{(i\to j)}_{\min}(m,n) \leftarrow \min(t(c_1,c_2), Mat^{(i\to j)}_{\min}(m,n))$
          $Mat^{(i\to j)}_{\max}(m,n) \leftarrow \max(t(c_1,c_2), Mat^{(i\to j)}_{\max}(m,n))$
        end if
      end while
      $Mat^{(i\to j)}_{\min}(O,m) \leftarrow \min(t(c_1) - t_{orig}, Mat^{(i\to j)}_{\min}(O,m))$
      $Mat^{(i\to j)}_{\max}(O,m) \leftarrow \max(t(c_1) - t_{orig}, Mat^{(i\to j)}_{\max}(O,m))$
      $Mat^{(i\to j)}_{\min}(m,D) \leftarrow \min(t_{dest} - t(c_1), Mat^{(i\to j)}_{\min}(m,D))$
      $Mat^{(i\to j)}_{\max}(m,D) \leftarrow \max(t_{dest} - t(c_1), Mat^{(i\to j)}_{\max}(m,D))$
    end while
  end while
```

#### 3.3.2.3 Extraction of Train Travel Candidates

Based on the learning table created in Section 3.3.2.2, I calculate the likelihood $l^{(i\to j)}_{u,t}$ that the user $u$ departed from station $i$ at time $t$ and travelled to station $j$. In the element $Tbl^{(i\to j)}(m,n)$ of the learning table, the communication interval from the BS $m$ to the BS $n$ is recorded. However, learned communication intervals is not perfectly covered by training data. Therefore, the communication interval from the BS $m$ to the BS $n$ in the communication history of the estimation target user $u$ does not necessarily coincide with the learning interval. For this reason, the likelihood should be defined so as to allow some difference between the interval of the target user's record and the interval learned from the training records.

Based on this fact, $l^{(i\to j)}_{u,t}$ is calculated using Algorithm 2. The function $p(t,m)$ in Algorithm 2 is defined as Eq. (3.17) , which calculates the likelihood from the communication interval $t$ and the learning interval $m$. As shown in Fig. 3.7, the function $p(t,m)$ is 1 if the communication interval $t$ is the median value of the learning interval, or $\exp(-\frac{1}{2})$ if it is at the end of the learning interval. It is defined so that the return value becomes smaller with the increase of the gap between $t$ and $m$.

$$p(t,m) = \exp\left(-\frac{(t - \frac{m_{\min}+m_{\max}}{2})^2}{2 \cdot (\frac{m_{\min}-m_{\max}}{2})^2}\right) \tag{3.17}$$

In each communication tuple, the likelihood of the communication interval is calculated based on the function $p(t,m)$. If the product of likelihoods is greater than or equal to the threshold, it is added to the travel candidates $\mathcal{TC}^{(u)}$ of the user $u$ as travel from the station $i$ to the station $j$ with the departure time $t$. However, in Algorithm 2, the likelihood $l_{u,t}^{(i\to j)}$ becomes 1 when communication does not occur or a user $u$ communicates with a single BS during the verification period. Since these cases are not appropriate for the extraction of travel candidates, they are removed from the candidates. Here, the verification period $[t, t+\Delta T]$ represents a period for estimating whether it is a travel or not, and $\Delta T$ represents the time required for a travel obtained by different learning for each travel.

**Algorithm 2** Calculate Likelihood

```
Input: $C_u^{(t,t+\Delta T)}, Tbl^{(i\to j)}$
Output: $l_{u,t}^{(i\to j)}$
  /* initialization */
  $l_{u,t}^{(i\to j)} \leftarrow 1.0$
  /* calculate likelihood */
  while $c_1 \in C_u^{(t,t+\Delta T)}$ do
    $m \leftarrow BS(c_1)$
    while $c_2 \in C_u^{(t,t+\Delta T)}$ do
      if $BS(c_1) \neq BS(c_2) \land t(c_1,c_2) > 0$ then
        $n \leftarrow BS(c_2)$
        $l_{u,t}^{(i\to j)} \leftarrow l_{u,t}^{(i\to j)} \times p\left(t(c_1,c_2), Tbl^{(i\to j)}(m,n)\right)$
      end if
    end while
    $l_{u,t}^{(i\to j)} \leftarrow l_{u,t}^{(i\to j)} \times p\left(t(c_1) - t, Tbl^{(i\to j)}(O,m)\right)$
    $l_{u,t}^{(i\to j)} \leftarrow l_{u,t}^{(i\to j)} \times p\left((t+\Delta T) - t(c_1), Tbl^{(i\to j)}(m,D)\right)$
  end while
```

The aforementioned likelihood threshold is defined by $\{\exp(-\frac{1}{2})\}^N$, where $N$ represents the number of combinations of communication occurred during the verification period of the estimation target user. This threshold is defined so that travel likelihood always exceeds it if the communication interval of all communication tuples in the verification period are within the learning interval. Even in the case where there is a communication tuple whose communication interval is out of the learning interval, if the other communication tuple is close to the median value of the learning interval, the likelihood exceeds the threshold. Therefore, it is possible to prevent omission of travel detection.

#### 3.3.2.4 Travel Estimation

In this section, I describe a method to estimate all travels of users based on the travel candidate $\mathcal{TC}^{(u)}$ detected in section 3.3.2.3. Depending on the user's BS transition pattern, multiple travel candidates

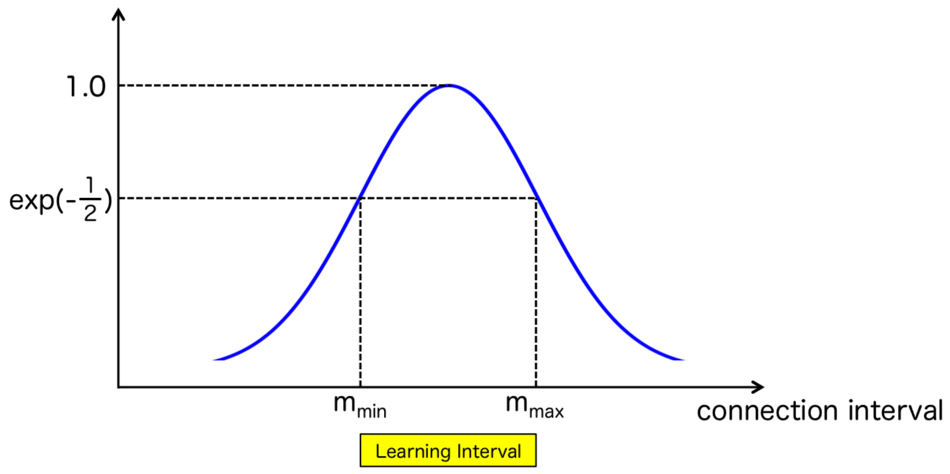


Figure 3.7: Relationship between learning interval and likelihood function $p$

may be detected at the same time. However, since a single user never makes multiple travels at the same time, I resolve such conflicts according to the likelihood of the travel candidate. The travel with the highest likelihood among the travel candidates is sequentially allocated as a travel of the user $u$ and removed from the candidates. This process is repeated until travel allocation becomes impossible. In this allocation, as shown in Fig. 3.8, if another travel already exists in the allocation period, the travel candidate is excluded without allocation. In this way, out method estimates the all travel of the user.

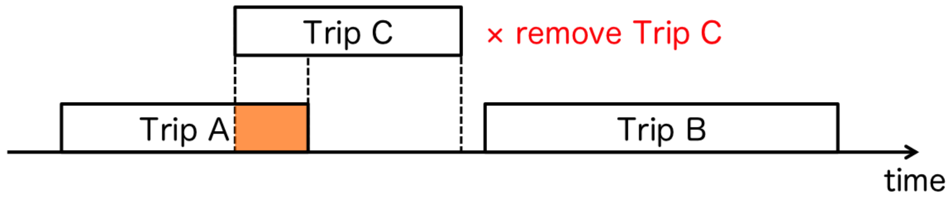


Figure 3.8: Example of estimating a travel of a day from a travel candidates

## 3.4 Evaluation

### 3.4.1 Simulation Settings

To evaluate the performance of the proposed methods, I have conducted simulation experiments using a network simulator Scenargie [7]. In the simulations, I have introduced not only control signals in cellular networks but also mobility of users to confirm that the proposed method can estimate travels from the control signals.

First, I run the following simulation scenario with 2,000 users. The simulation time is from 7 am to 7 pm, and the users move in the area with 6,000m x 6,000m shown in Fig. 3.9. These users have several travels according to their attributes. 750 users are categorized into (a) train commuter. The users move to their offices by trains and on foot. Their offices are randomly selected from the buildings colored pink in the map. They are randomly placed in either X station or Y station at around 7 am. The trains depart from X with 5 minutes interval, and arrive at Y after 12 minutes through three stops. The trains from Y to X run in same way. At first, they move to one of three stations located in the middle of the map from the starting station by train. After that, they go to their offices from the stations on foot. When they arrive at the offices, they stay in the offices until around 5 pm and go back to the starting station on foot and by train. Thus, each user has two travels by train and two travels on foot in this category. 150 users are categorized into (b) car commuter, and are randomly placed in the residential area colored green in the map. They go to their offices directory by cars at around 9 am, and work at the office until around 5 pm. They also go back to their home by cars. Another 600 users are categorized into (c) bus commuter, and are randomly placed in the residential area as same as (b). Six bus lines are operated in the map. The buses in line A run from bus stop a to g through b, c, d, e, and f in 21 minutes. The buses in line C run from bus stop h to m through i, j, k and l in 20 minutes. The buses in line B and D run in the opposite directions of line A and C, respectively. The buses in line E run from bus stop n to q through o and p in 10 minutes. The buses in line F run from bus stop r to u through s and t in 11 minutes. They move to their offices from their homes as well as the users in (a) and (b) in around 9 am on foot and by bus, and stay in the offices selected randomly until around 5 pm. Then, they visit one of the stores colored orange in the map, and spend around 30 minutes there. Finally, they go back to their homes. The rest 500 users are categorized into (d) student. They are also randomly placed in the residential area, and move to the nearest school colored blue in the map at around 9 am on foot and by bus. They stay in the schools until around 5 pm, and move to the nearest park colored green in the map. They also stay there in around 30 minutes, and go back to their home. Thus, each user has three travels on foot in the category.

In addition, I reproduced the real map and run the following simulation scenario with 100,000 users. The simulation time is 4 hours, and the users move in the 10km x 8km area shown in Fig. 3.10. The 247 BSs (colored red) are deployed based on Ktai-lab (BS search service) [54]. The building height is 20 m, the BS height is 50 m and The user's MS height is 1.5m. Each user is randomly placed in building (colored gray). Simultaneously with the start of the simulation, the user travels to randomly selected building. The 20% users arrive at the destination by automobile. The remaining 80% of users travel to their destination by train or on foot. In addition, I reproduce the 34 train routes and 113 stations (colored blue) based on actual geography, and the trains are operated at 3 - 8 minutes interval based on the actual timetable.

Their network communication intervals follow an exponential distribution shown in the Eq. (3.18), where $\alpha$ is the expected interval, $x$ $(0 < x < 1)$ is a uniform random value. In the performance

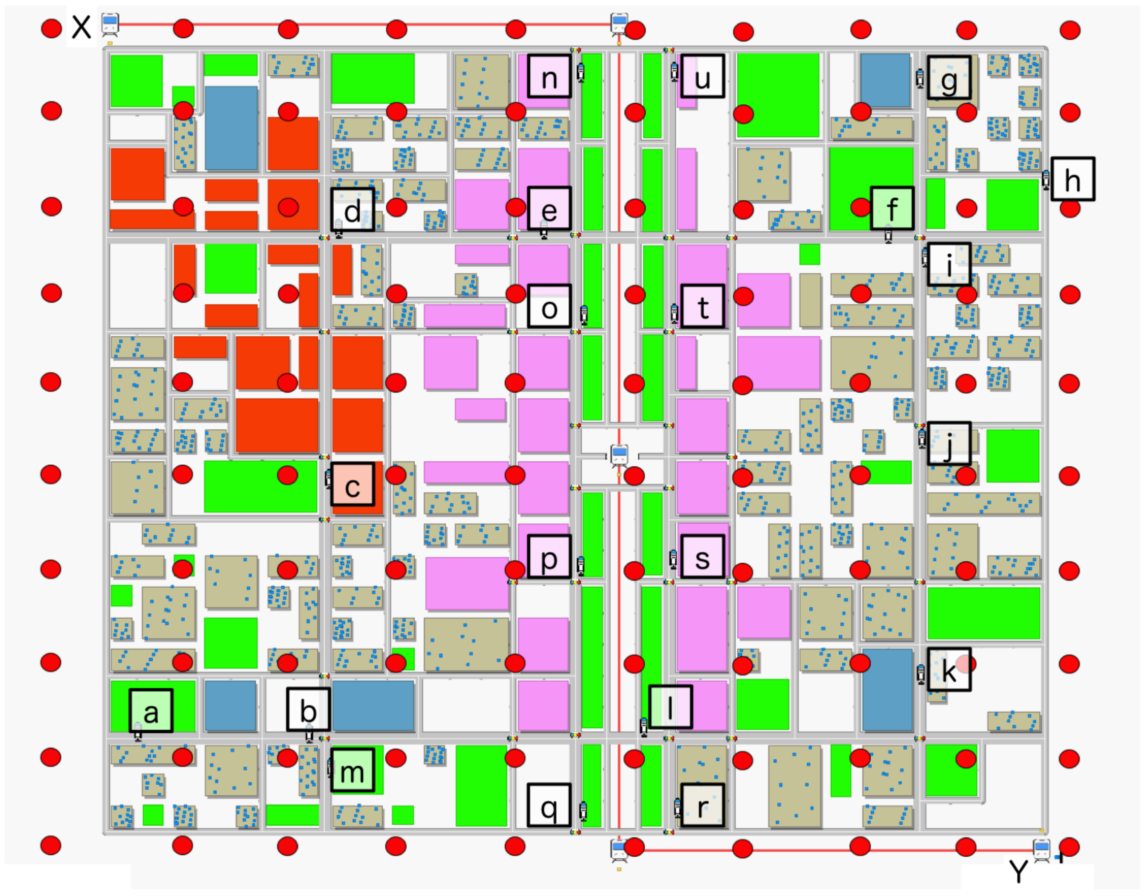


Figure 3.9: Manhattan Map of simulation

evaluation, I changed the expected interval and verified the difference in accuracy according to communication intervals. The control signals are generated at the BSs according to the model in Section 3.2.2.

$$T = -\alpha \ln x \tag{3.18}$$

### 3.4.2 Train Travel Estimation

#### 3.4.2.1 Manhattan Map

At first, I have evaluated the accuracy of the estimation for train travels. Fig. 3.11 shows the precision and recall of Manhattan Map according to the number of control signals for a user. The horizontal axis represents the number of the control signals, and the vertical axis represents the precision and recall, respectively. I omitted the users that sent only one or two control signals during the simulations due to the limitation of the proposed method as stated in the previous section. As shown in the figure, as the number of the control signals increases, the recall increases and is over 90% in the cases with

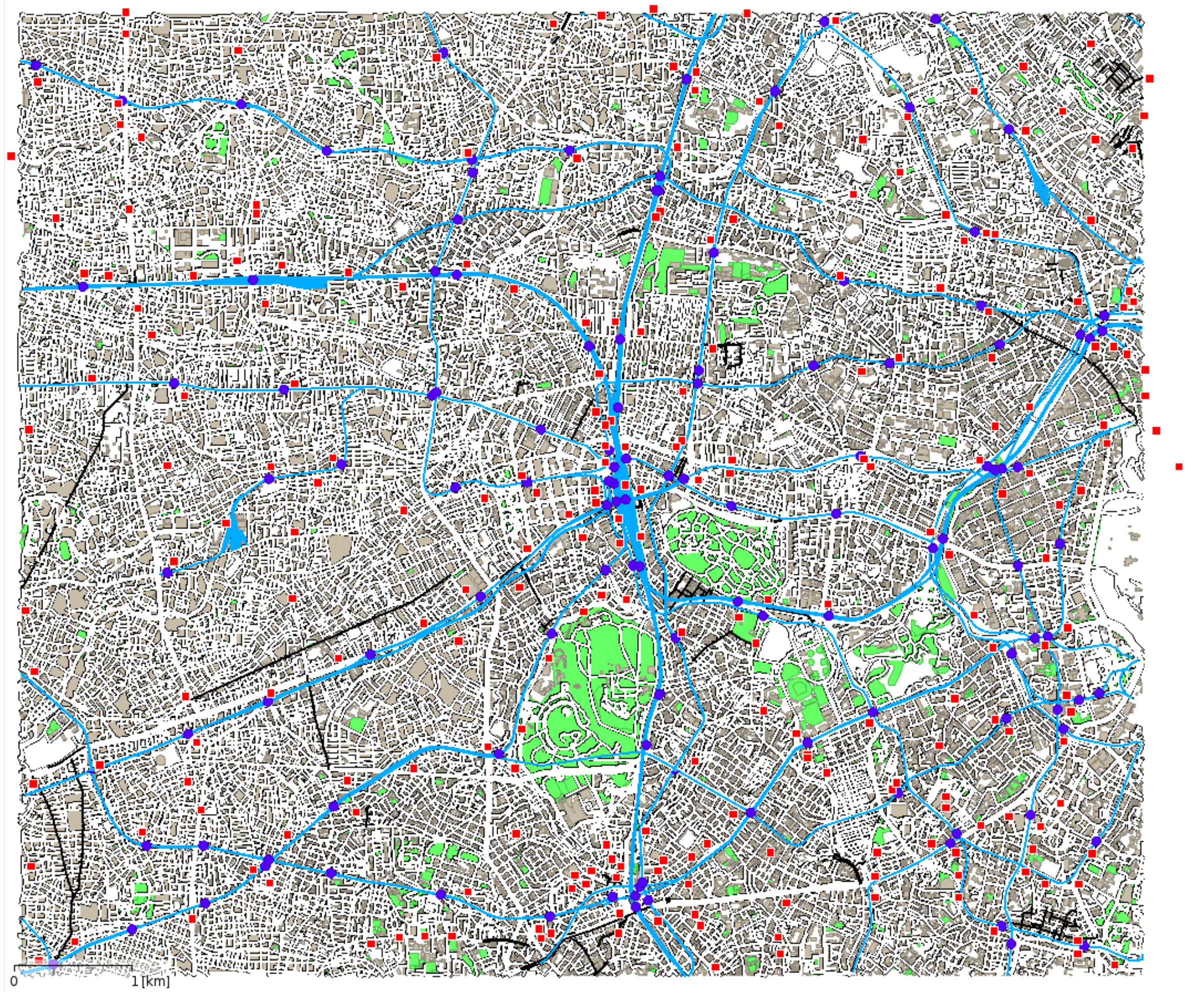


Figure 3.10: Real Map of simulation (OSM of Shinjuku station area) [1]

more than 15 control signals. Also, the accuracy of the estimation is around 90% in all cases. Thus, I can see that the proposed method estimated train travels with high accuracy from the simulations.

In addition, I have also evaluated whether the proposed method could estimate the correct train for each user according to the timetables. As show in the Sec. 3.3.1.4, I have introduced four different schemes to calculate the likelihood to trains. The results for the schemes are shown as a box-and-whisker diagram in Fig. 3.12. From the figure, I can see that the scheme (2), (3) and (4) achieved about 80% accuracy. However, the scheme (1) could not estimate trains correctly because this scheme did not match the BS selection model used in the simulations. The results show that I have to choose a scheme for each area carefully according to the radio characteristics such radio propagation, radio fading and so on.

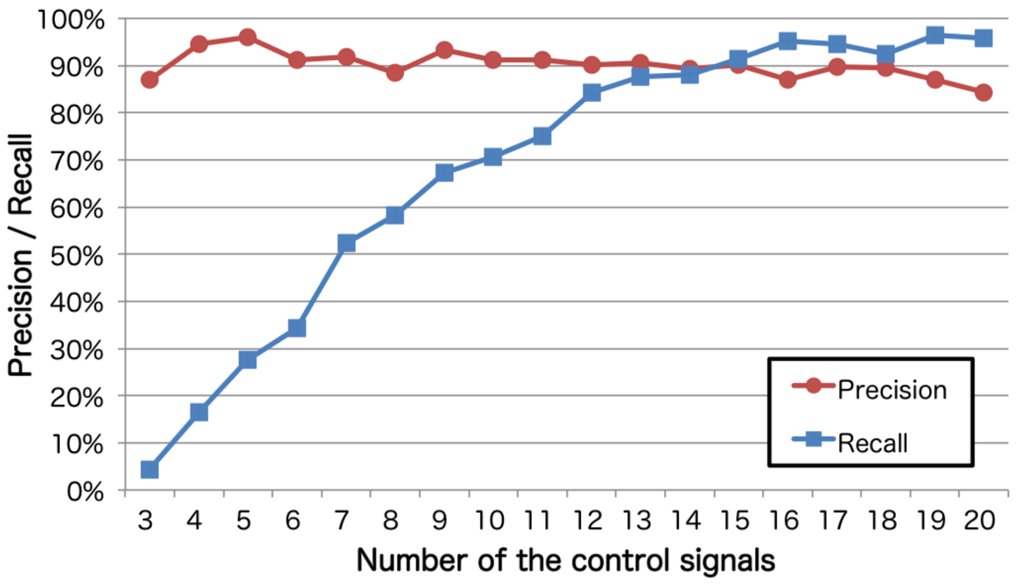


Figure 3.11: Precision and recall for railway passenger estimation (Manhattan Map)

#### 3.4.2.2 Real Map

I have evaluated the accuracy of the estimation for train travels. In order to verify the difference in estimation accuracy due to communication intervals, the performance is evaluated on six expected intervals (30sec, 60sec, 90sec, 120sec, 150sec and 180sec). In NoGPS, $\mathrm{StayTime} = 30\mathrm{min}, \mathrm{StayFreq} = 0.5$ in Eq. (3.4) are applied to for all users. In WithGPS, the number of GPS provided users for each route is 20. Fig. 3.13 shows the precision, recall and F-score of NoGPS (broken line) and WithGPS (solid line) for each expected intervals. The horizontal axis represents the expected connection intervals, and the vertical axis represents the precision, recall and F-score, respectively. As shown in the figure, although accuracy is almost the same for both methods, it can be seen that WithGPS is superior in recall. Since WithGPS estimates using the communication interval, it is considered that WithGPS estimates the travel accurately even in the case of no communication in the middle.

However, in Fig. 3.13, WithGPS is superior, but when the expected connection interval exceeds 90 seconds, the recall is less than 60%. One reason for this is that train passengers who hardly communicated with simulation data were included. Because it is impossible to estimate the travel of users who do not communicate, I evaluate the accuracy except for users with 3 or fewer communications while on the train. The result is shown in Fig. 3.14. This result shows that even when the communication interval is long, it achieves a high recall of 80% or more, and the effectiveness of WithGPS is shown.

In addition, in order to verify the difference in accuracy of WithGPS due to the amount of training data, I evaluate performance according to the amount of training data of 5 cases (each route 10, 15,

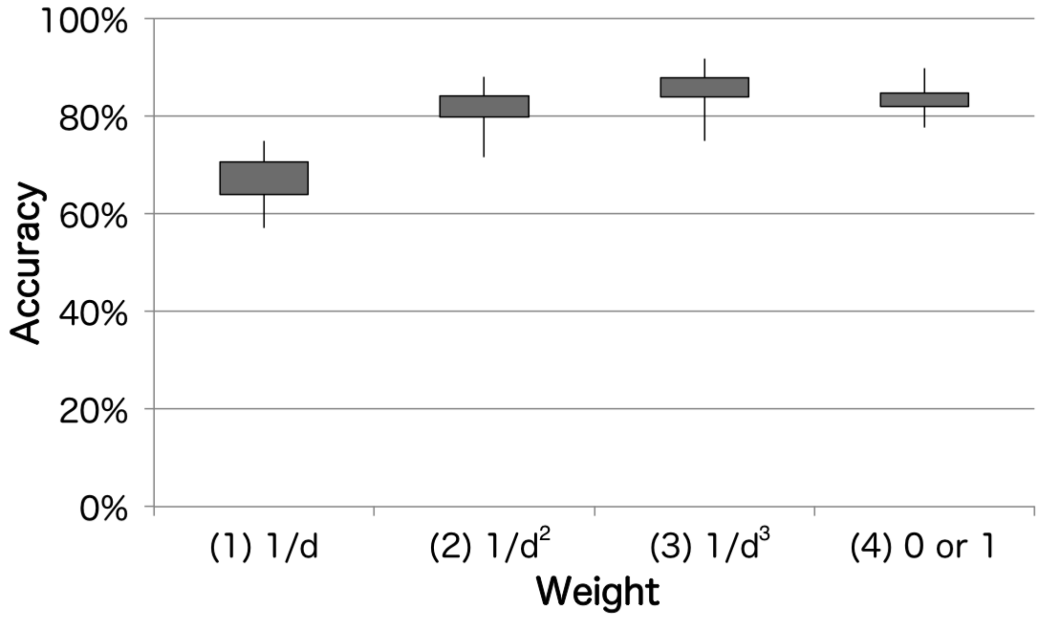


Figure 3.12: Accuracy of Boarded Train Estimation (Manhattan Map)

20, 25 and 30 users). Also, the user's expected connection intervals is set to 60 seconds. The results are shown in Fig. 3.15. From this result, it can be seen that the recall increases as the number of data sets increases. This is because the learning interval of the communication interval becomes wider as the number of training data increases, and the detection omission of train passengers is prevented. In addition, even when the number of training data is 10, WithGPS achieves F-score of 80% or more, indicating that performance can be demonstrated with less training data.

### 3.4.3 Automobile Travel Estimation

#### 3.4.3.1 Manhattan Map

In this section, I show the performance of the estimation for automobile travels. The precision and recall for 10 travel distance ranges are shown in Fig. 3.16. The prior probability $P(R_k)$ passing through the road $R_k$ was a uniform distribution in the simulations. This means that no prior knowledge is given. The average of the precision and recall are 70% and 75%, respectively. Thus means that F-value is 0.851. As shown in the figure, the proposed method could not estimate the short travels under 2.0 km. The travel speed was not estimated correctly in these cases because these travels were over several BSs and the position error are relatively large for the travel distances. On the other hand, longer travels contains more than 10 BSs and such position errors were mitigated.

I have also evaluated travel paths for the automobile travels estimated correctly in the previous process. $N_{auto}$ shown in Sec. 3.3.1.5 was 50. I introduce a following metric for a path to measure how

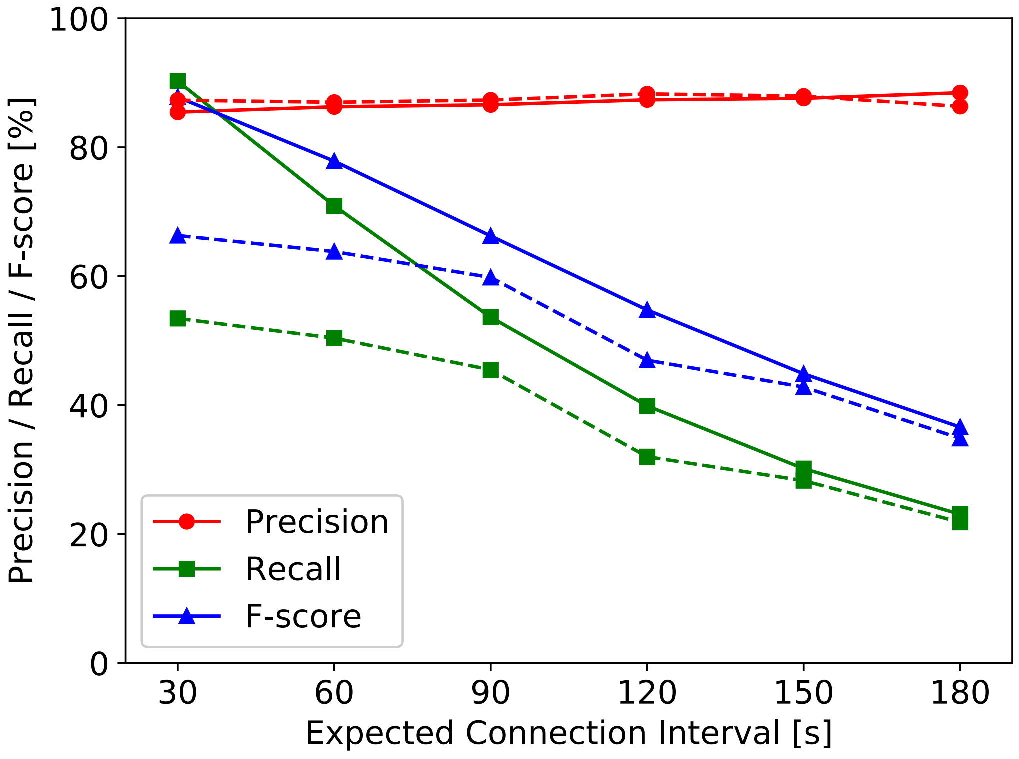


Figure 3.13: Precision and recall for train passenger estimation (Real Map)

an estimated path matches to an actual path for a travel. The metric are shown in as Eq. (3.19), where $IS_t$ is sequences of intersection automobile passengers have traversed, $IS_t = \{IS_{t,1}, IS_{t,2}, \ldots, IS_{t,P}\}$, $IS_e$ is sequences of intersection of the estimated path traversed, $IS_e = \{IS_{e,1}, IS_{e,2}, \ldots, IS_{e,Q}\}$, $\mathrm{LevDist}(IS_t, IS_e)$ is Levenshtein distance between $IS_t$ and $IS_e$.

$$AC = 1 - \frac{\mathrm{LevDist}(IS_t, IS_e)}{\max(P, Q)} \tag{3.19}$$

The Levenshtein distance [55] is one of the edit distance used as an index indicating the similarity of the strings. It is defined by the minimum number of editing required to deform one string to another string. The editing is an operation to one character inserted at any position and rewrite any character in one character deletion or another character. The evaluation expressed its similarity to a string a sequence of intersection.

The results are shown in Fig. 3.17 (left) according to the metric. The mean value and median of the overall accuracy are 0.644 and 0.688. My method could estimate roughly correct paths for the travels. However, as shown in the figure, the accuracy for 25% of automobile travels is less than 50.0%. In these cases, my proposed method could not estimate the stay points correctly for the travels because

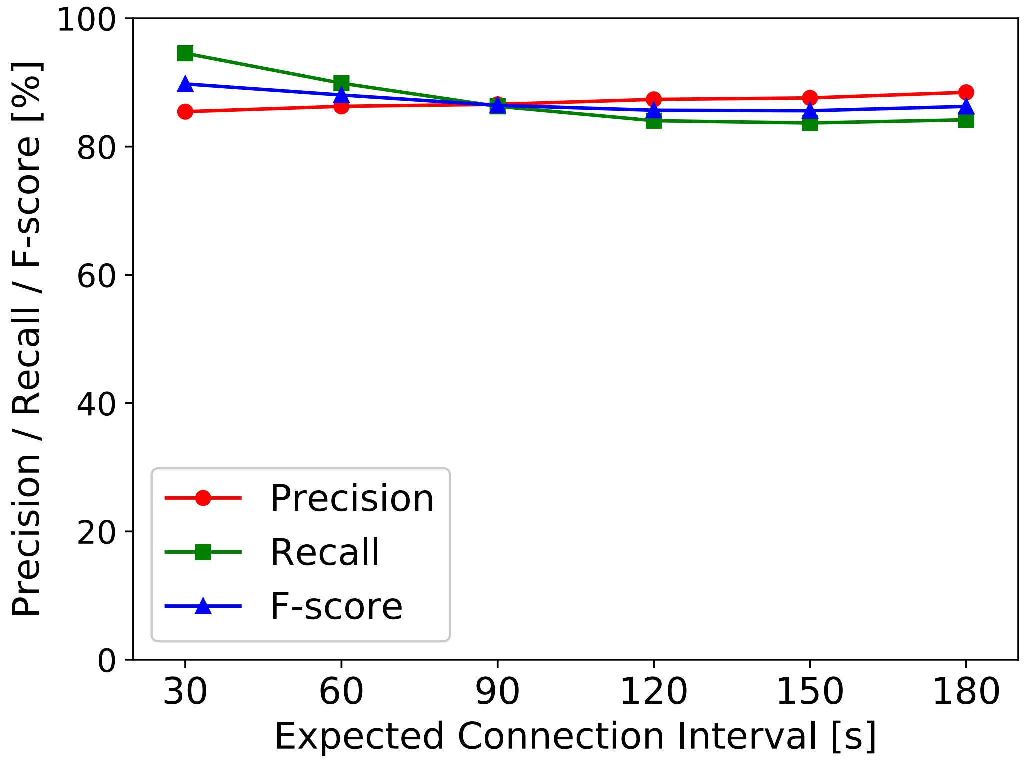


Figure 3.14: Precision and recall for train passenger estimation (exclude users with 3 or fewer communications)

the method derive their location in BS level. However, if I could estimated the stay points accurately, the path estimation could be drastically improved. I have conducted another experiments given such stay points. The result is shown in Fig. 3.17 (right). The mean value and median are 0.838 and 0.894. This means that I could get better results for travels with several anchor points because my method could derive different but similar paths.

#### 3.4.3.2 Real Map

In this section, I show the performance of the estimation for automobile travels. The precision and recall for 10 travel distance ranges are shown in Fig. 3.18. The prior probability $P(R_k)$ passing through the road $R_k$ was a uniform distribution in the simulations. This means that no prior knowledge is given. The average of the precision and recall are 65% and 69%, respectively. As shown in Fig. 3.18, the proposed method could not estimate the short travels under 2.0 km. The travel speed was not estimated correctly in these cases because these travels were over several BSs and the position error are relatively large for the travel distances. On the other hand, longer travels contains more than 10

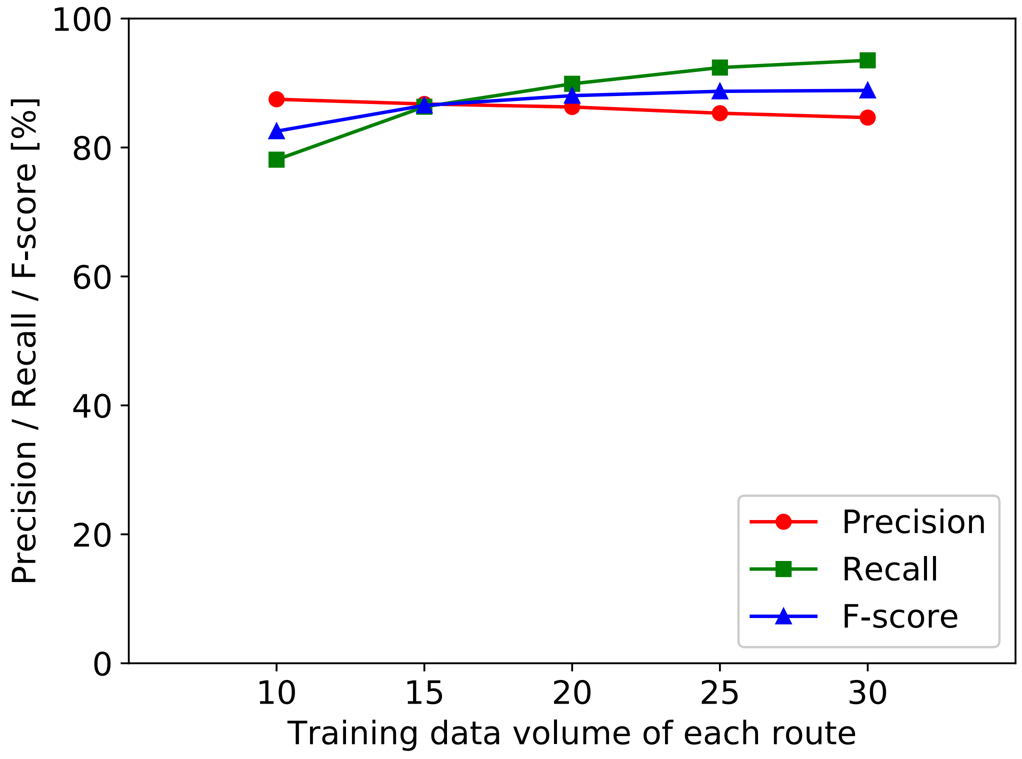


Figure 3.15: Precision and recall according to the amount of training data

BSs and such position errors were mitigated. The average precision and recall of users with distance of 2 km or more are 76% and 78%. Therefore, my method is suitable for the long distance mobile users.

I have also evaluated travel paths for the automobile travels estimated correctly in the previous process. The results are shown in Fig. 3.19 (left) according to the metric. The mean value and median of the overall accuracy are 0.67 and 0.77. My method could estimate roughly correct paths for the travels. However, as shown in the figure, the accuracy for 25% of automobile travels is less than 50.0%. In these cases, my proposed method could not estimate the stay points correctly for the travels because the method derive their location in BS level. However, if I could estimated the stay points accurately, the path estimation could be drastically improved. I have conducted another experiments given such stay points. The result is shown in Fig. 3.19 (right). The mean value and median are 0.76 and 0.84. This means that I could get better results for travels with several anchor points because my method could derive different but similar paths.

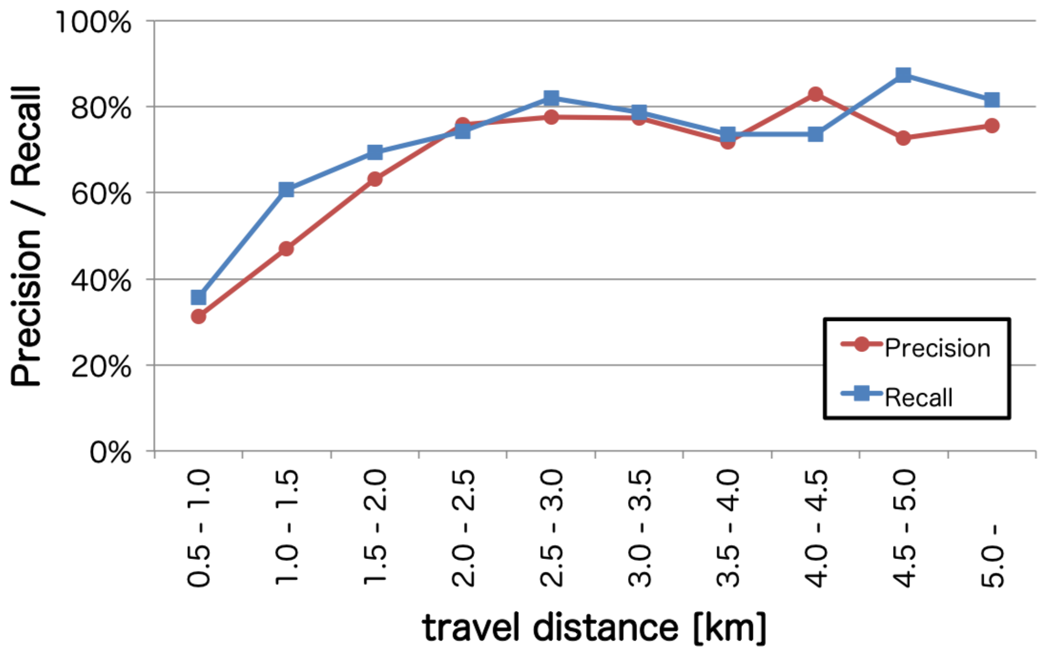


Figure 3.16: Precision and recall for automobile travel estimation (Manhattan Map)

## 3.5 Conclusion

In this study, I have proposed two methods to estimate travel modes and trajectories of people in urban areas. To estimate travels of people with mobile phones, I use Control Signal Records (CSRs) that are generated on the network operator's side when a user sends BSs any signal. To overcome the problem of spatially coarse grained locations provided by CSRs. It fully exploits geographical information such as stations, railway networks and road networks. The other method uses CSRs combined with GPS traces provided by cooperative users in order to consider complex radio propagation. Since CSRs with the ground truth of people movement are difficult to collect in practice, I have conducted a simulation study by modelling a BS selection characteristic focusing on distances from BSs. The simulation results have shown the proposed methods achieve fairly high accuracy.

My future work includes evaluating the accuracy of my methods in various realistic scenarios. I am also planning to evaluate the accuracy for unexpected events such as disasters.

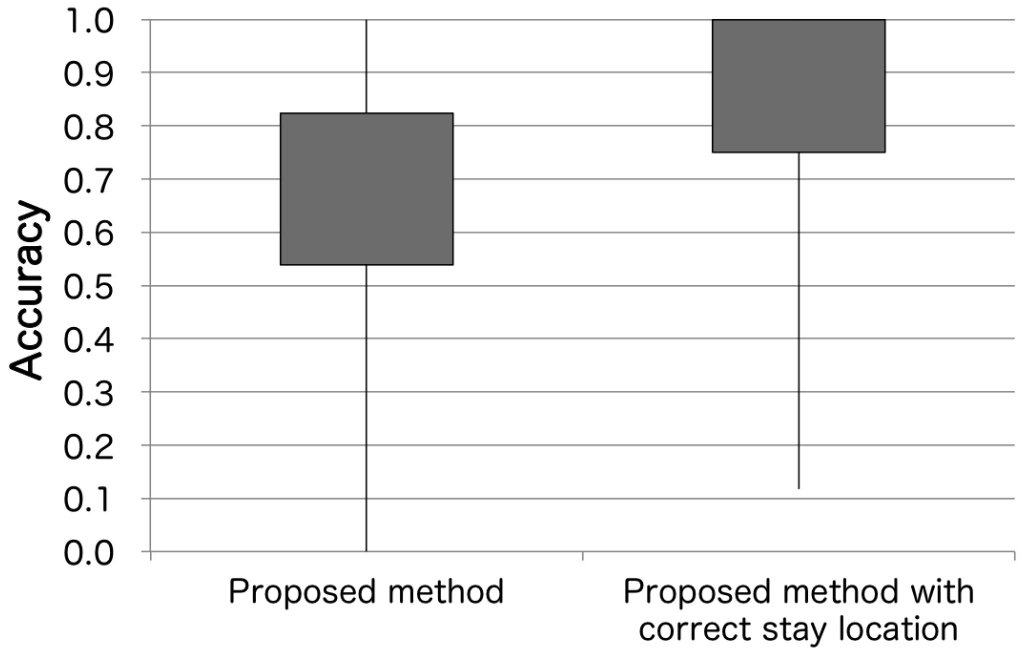


Figure 3.17: Automobile travel path estimation (Manhattan Map)

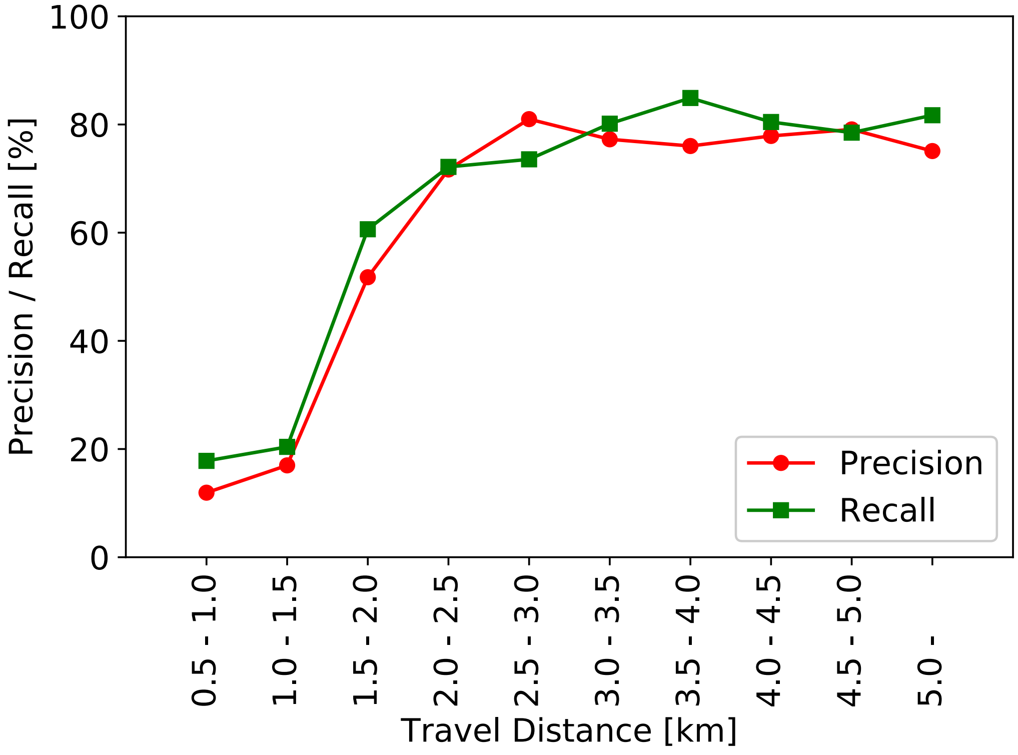


Figure 3.18: Precision and recall for automobile travel estimation (Real Map)

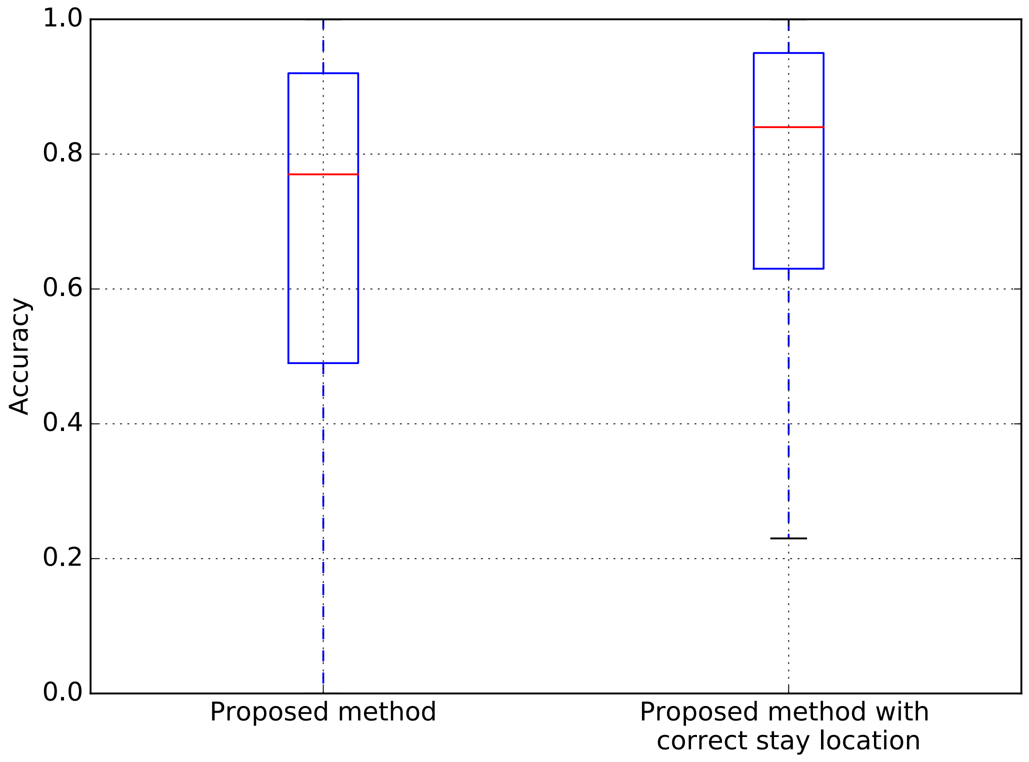


Figure 3.19: Automobile travel path estimation (Real Map)

# Chapter 4

# Route Bus Passenger Counter Using LiDAR Sensors

## 4.1 Introduction

Grasping the behavior of people by regional traffic such as trains and route buses is important to realize a safe and affluent society, such as regional revitalization by traffic optimization, streamlining, and disaster simulation. In particular, grasping the actual conditions of use of the route bus, which is the main of public transport in all regions is extremely important for the understanding of the use situation of local public transport and the realization of the correct improvement process. Data on the number of passengers is valuable data for transportation operators who can not accurately grasp the number of passengers. Besides, by measuring and analyzing the time required for getting on and off, it is possible to grasp a phenomenon in which a large number of passengers get off at one time on a congested route and the stop time at a stop increases. Such data can be used as an indicator that smooth operation can be realized, and based on that data, it can be basic data for the company to improve. Besides, if the number of passengers in each bus can be grasped in real-time, it can be used for evacuation guidance plans and rescue team planning when a passenger is locked up in a bus, when sudden disasters such as earthquakes occur. Not only that, when an event is held, it can be used to determine the number of additional buses while checking the number of passengers in real-time.

In major cities, The Ministry of Land, Infrastructure, Transport and Tourism in Japan conducts a questionnaire survey called a person-trip survey [3] every 10 years. However, the survey is difficult to conduct frequently and precisely due to huge effort involved while movements of people often changes depending on time, days and events. Person-trip surveys are not conducted in local cities. Some public transports collect Origin-Destination (OD) data of passengers recorded by IC cards. However, there are many cases in which IC cards have not been introduced, especially in regional traffic with deficit routes. Even if it has been introduced, it is often limited to some routes. In addition, it is impossible to count passengers who do not have an IC card. In the alternative method, it is possible to grasp the number of passengers by taking the numbered ticket when the passenger gets on the bus and collecting the numbered ticket at the time of getting off. However, when a passenger forgets to take a numbered ticket, it can not be counted, and the thorough taking the numbered ticket is an problem. Furthermore, because there is no real-time capability, there is also the problem that it can not cope with the sudden disaster occurrence or holding an event as mentioned above.

A system using infrared sensors have been developed to count the number of passengers [37]. The system uses two infrared sensors side by side, and counts the number of passengers by using the difference between their detection times. However, when two or more passengers pass the sensors at the same time, it can not be counted accurately, and can not measure the time required for getting on and off. In addition, a system for measuring passengers by using RGB cameras installed on the ceiling of the bus entrance has been developed [39]. Such a system can avoid occlusion due to overlapping of passengers in the sensor field of view, and achieve more accurate passenger detection. However, the cost of installing and operating the system can not be ignored in terms of the need for ceiling installation and the need for high-load image analysis. Furthermore, in Japan, because taking images for purposes other than safety surveillance is a risk of privacy violation, it has the disadvantage that it is not easy to introduce.

In this chapter, I propose a lightweight and high-performance measurement system that processes LiDAR's (Light Detection and Ranging) raw data in real-time with a single board computer (ex: Raspberry Pi 3 Model B) in order to reduce the labor and cost barriers for implementation. Specifically, the counting passengers error is targeted at 20%, which is the guaranteeing accuracy of the person-trip survey, and the processing time per scan is within 100 ms, which is the scanning time of LiDAR. LiDAR is a sensor that can accurately measure the distance to surrounding objects over a wide range. LiDAR of Hokuyo Automatic Co., Ltd. can scan a wide area at high speed and accurately, with a detection guarantee distance of 5.6 m, a scanning angle of 240 degrees, a ranging accuracy of 30 mm, and a scanning time of 100 ms/scan. The power consumption is also very low, with a maximum of 2.5W. In addition, the raw data of LiDAR is only the position information of the measurement object that can be expressed by the angle and distance from the sensor. Therefore, it has also the advantage that there is little risk of violating people's privacy. In the proposed method, sensor calibration can be performed only by getting background data of the opened and closed door state (ODS/CDS) with the system installed at an appropriate location. It use the background difference method for automatically detecting the opening and closing door of the bus. After ODS detection, the proposed method extracts the point cloud corresponding to the object surface using the background subtraction method and detects the point cloud corresponding to the human surface from the point cloud. The proposed method tracks the point cloud of each passenger and realizes the count of the number of passengers by detecting the passage of the door border. My method automatically detects the opening and closing of the door, so the getting-on/off time and the number of passengers at each stop can be measured simultaneously, and the arrival time and stopping time for each stop can be grasped.

Therefore, it can also be used for route bus operation planning.

In evaluation, I conducted experiments with a bus running between two campuses of my university and achieved 5.3% or lower error (the average absolute error was 0.94%) for coming passengers and 7.5% (the average was 1.9%) for leaving ones. The average processing time per frame was 3.4 ms in Raspberry Pi 3 Model B. This has shown the capability of my algorithm for real-time measurement.

## 4.2 Environment

### 4.2.1 System Overview

Fig. 4.1 shows a proposed system overview of this research. The measurement system is equipped with a laser range sensor, a single-board computer, and a GPS sensor, and it is installed at two locations front and rear the route bus based on the installation conditions described later in Section 4.2.3. In each measurement system, a laser range sensor and a computer are connected, and the sensor data is sent to the computer sequentially. The computer that received the sensor data performs passer measurements in real-time. Besides, the bus stop is detected from the GPS sensor, and the bus stop and measurement information (the number of passenger and door opening time) are recorded together. Every time a certain amount of measurement information is accumulated, the measurement information is uploaded to the data server, and the measurement information of each bus stop is aggregated. Moreover, it is possible to distribute the position information by grasping the position of the bus in real-time, and there is an advantage that not only the delay of the bus itself can be grasped, but also the position of the vehicle can be specified immediately at the time of disaster occurrence.

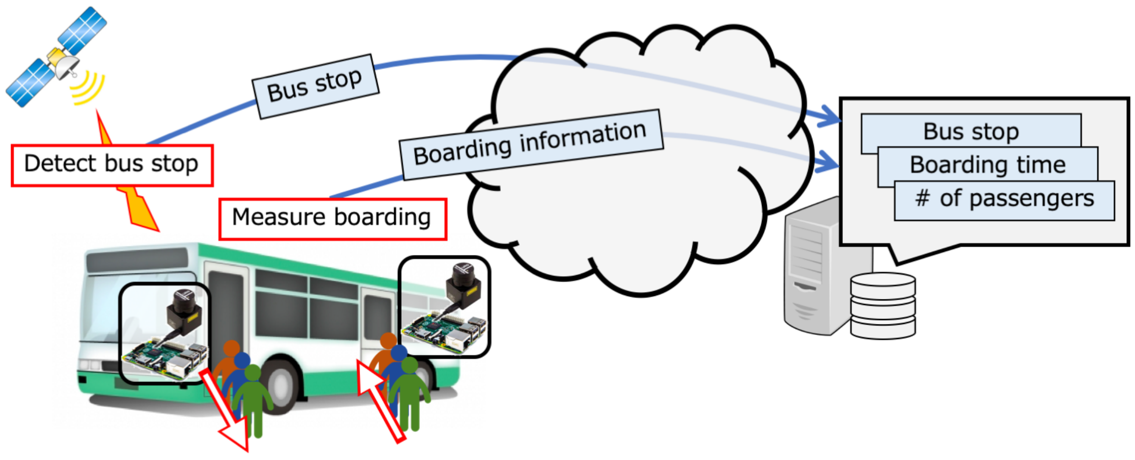


Figure 4.1: System overview

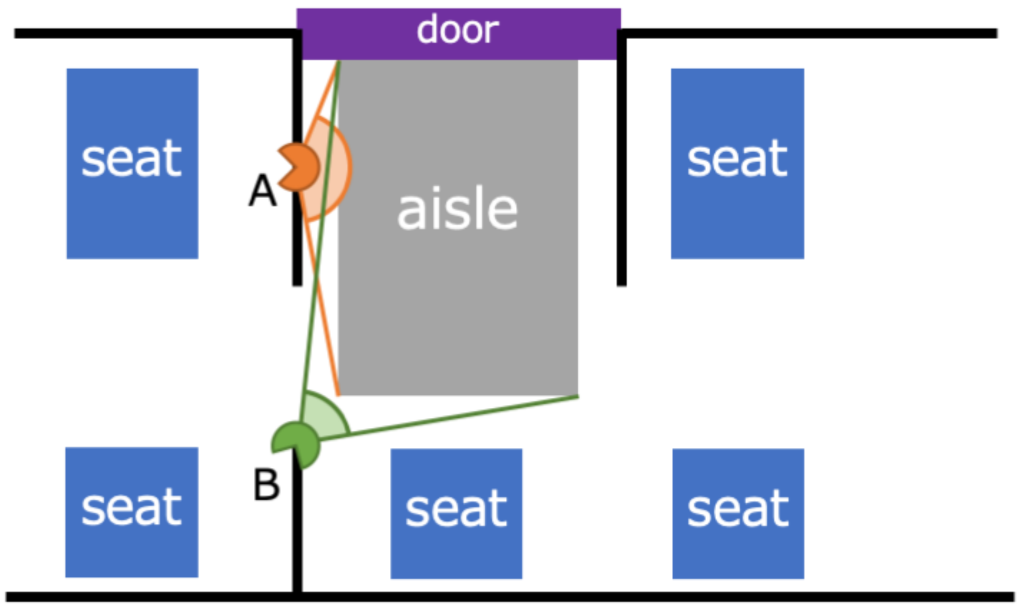
### 4.2.2 Gate of Route Bus

In Japan, route buses are divided into non-step and one-step buses. The non-step bus has a low floor structure with a floor height of 350 mm and is designed to be easily accessible to the elderly and people with disabilities. Although the number of non-step buses is increasing recently, the introduction rate as of 2016 has not reached half at 37.6% [56]. On the other hand, the one-step bus has a floor height of about 530 mm and currently occupies more than half of the total. In the one-step bus, due to the difference in height between the ground and the floor, when the passenger is on the ground, the torso of the passenger is sensed. However, when the passenger is on the bus, the legs of the passenger are sensed. I need to design an appropriate algorithm considering the height difference.

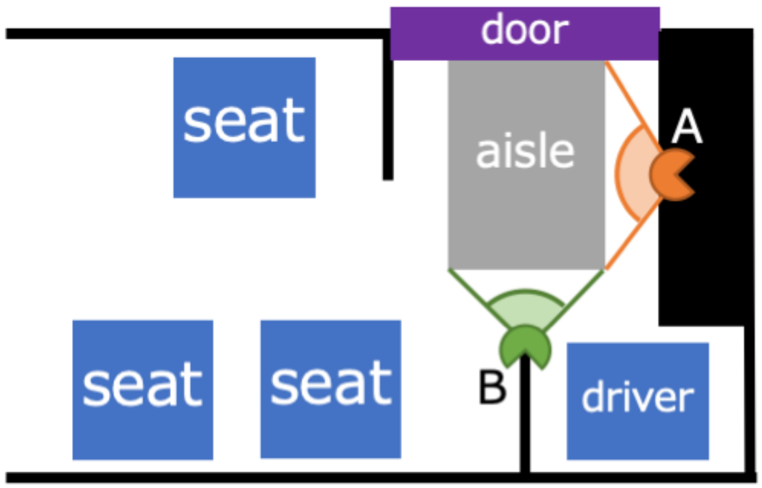
### 4.2.3 LiDAR Installation Conditions

In order to simplify the measurement process, the range sensor is assumed to be installed horizontally to the floor. Since the proposed method uses the background subtraction method, the range sensor is completely fixed so that it does not move during the measurement. In order to detect the opening and closing of the door, suppose that the measurement range of the range sensor includes the door. The range sensors are installed at a position where the central angle of the measurement surface with respect to the entrance/exit passage is as large as possible in order to reduce the influence of occlusion that obstructs the sensor eld of view between passengers. Figure 4.2 shows an example of a range sensor installation. Positions A and B are candidates for range sensor installation positions. In both cases of Fig. 4.2(a) and Fig. 4.2(b), sensor A has a larger central angle of the measurement surface of the passage, it can be said that it is desirable to install the range sensor at position A in the example of Fig. 4.2. The range sensor will be installed at a height that allows both passengers on the ground and floor to sense regardless of the height of the passenger.

In actual installations, it is necessary to carefully select an installation location that does not hinder passengers getting on and off and that does not interfere with safety even during sudden braking. In the demonstration experiment conducted in this study, a place that does not affect safety was selected in consultation with the operator. Also, the laser beam emitted by the range sensor used is classified as “FDA Laser Hazard Class 1 (Considered non-hazardous)”, so there is no problem even if it enters the passenger’s eyes.



(a) rear door



(b) front door

Figure 4.2: Example of the LiDAR sensor position

## 4.3 Proposed Method

Figure 4.3 shows an overview of the proposed method. First, in the introduction phase, devices are installed and background data in the route bus is scanned. At this time, background data at the time of opening and closing is scanned and stored on a computer. Next, in the getting-on/off measurement phase, the opening state is detected using two background data. If the opening door can be detected, the background data at the time of opening the door is used to track passengers and measure the number of passengers. After that, if the closing state is detected, the time required for getting on and off and the number of passengers are recorded on the server. If the information of the bus opening/closing switch can be acquired, the opening/closing state of the door can be acquired. However, as with the GPS, it is more reasonable to detect the opening/closing of the door in consideration of wiring and construction work cost. Therefore, in my method, the door opening/closing is detected using a range sensor.

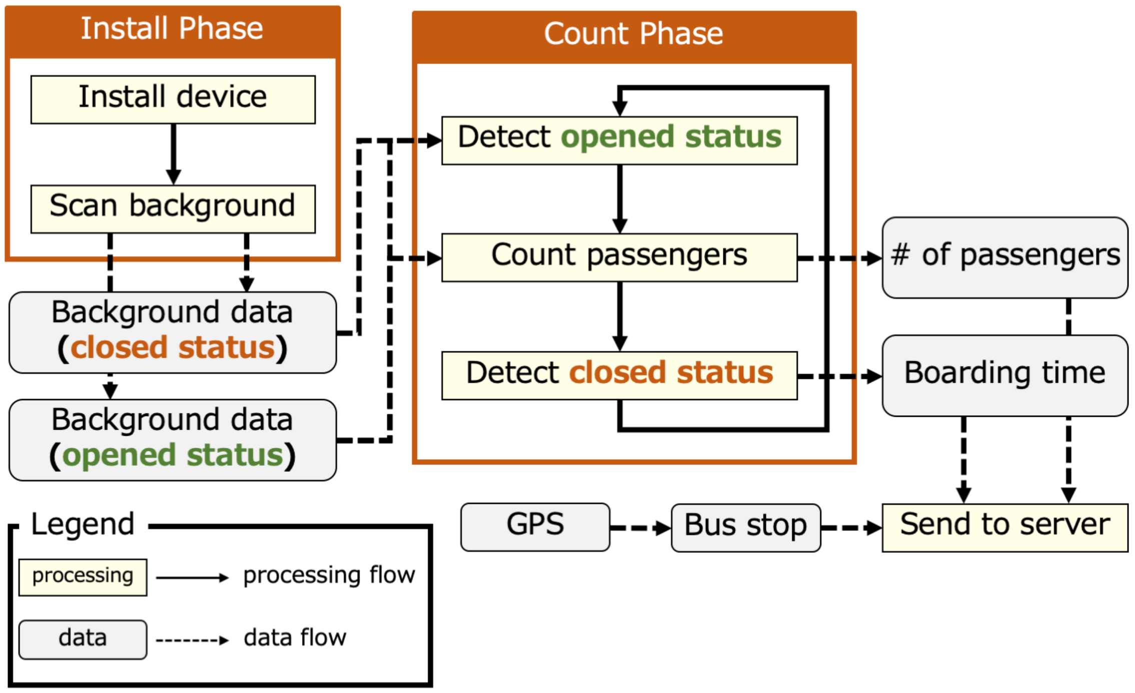


Figure 4.3: Proposed method overview

### 4.3.1 Laser Range Sensor Characteristics

This section describes the characteristics of the laser range sensors used in this study and the representation method of the measurement data. As an example of a range sensor, the specifications of URG-04LX-UG01 [2] made by Hokuyo Automatic Co., LTD. used in this study are shown in Table 4.1. The range sensor is a sensor that measures the distance from the range sensor to the object using the propagation time until the laser beam is reflected back to the object. As can be seen from Table 4.1, although the range sensor is small, it can measure a wide range with a detection distance of 5.6 m and a scanning range of 240 degrees with a tiny error (several centimeters or less). Besides, the measurement data obtained from the range sensor has only the position information calculated from the direction and distance of the object and has the advantage that the data size is small and easy to handle. However, it is impossible to measure the distance of objects that transmit laser light, such as glass. Besides, clothes of black or dark blue with a low reflectance have a shorter detection distance, as a result of investigating the detection distance of the clothes in advance, the measurement distance of black clothes is at least 3 m. However, the width of the bus entrance/exit is only 2 m. It can be said that there is no influence on clothes.

Table 4.1: Specification of URG-04LX-UG01 [2]

| Item | Spec |
|---|---|
| Measuring Distance | 0.02 - 5.6m |
| Accuracy | 0.06 - 1m: ±30mm |
| | 1 - 3m: 3% of measurement |
| Measuring Angle | 240° |
| Angular Resolution | 0.36° |
| Scanning Time | 100 ms/scan |
| Size | W50 × D50 × H70 mm |
| Weight | 160g |
| Light Source | Semiconductor laser diode ($\lambda = 785$nm) |
| Safety | FDA Laser Hazard Class 1 (Considered non-hazardous) |

In this thesis, $\mathcal{D}$ represents the data of one measurement obtained by the range sensor (measured once in all directions at the sensor's viewing angle). When the eld of view of the range sensor is scanned counter clockwise, the first measurement data obtained is zeroth, and the $i$-th distance data is expressed as $\mathcal{D}(i)$. In this paper, this i is called an orientation step. The measurement data $\mathcal{D}$ is expressed as a point $(x, y)$ on the $x$-$y$ plane, and the coordinate system of the range sensor is shown in Fig. 4.4. The position of the range sensor corresponds to the origin of the coordinate plane, and the scan range angle is $2\Phi$, the sector range is $\pm\Phi$ with respect to the positive $x$-axis direction. If the number of data obtained from one measurement is $N_{scan}$, the angular resolution $\Delta\theta$ is expressed as $\dfrac{2\Phi}{N_{scan} - 1}$. Therefore, the azimuth angle $\theta_i$ of azimuth step $i$ can be expressed as $-\Phi + i\Delta\theta$. Therefore, the point group $\mathcal{P}$ on the $x$-$y$ coordinate plane can be obtained from the measurement data of the range sensor using Eq (4.1).

$$\mathcal{P} = \{p_i = (x_i, y_i) \mid x_i = d_i \cos\theta_i,\ y_i = d_i \sin\theta_i,\ i \in \mathbb{Z} \cap [0, N_{scan})\} \tag{4.1}$$

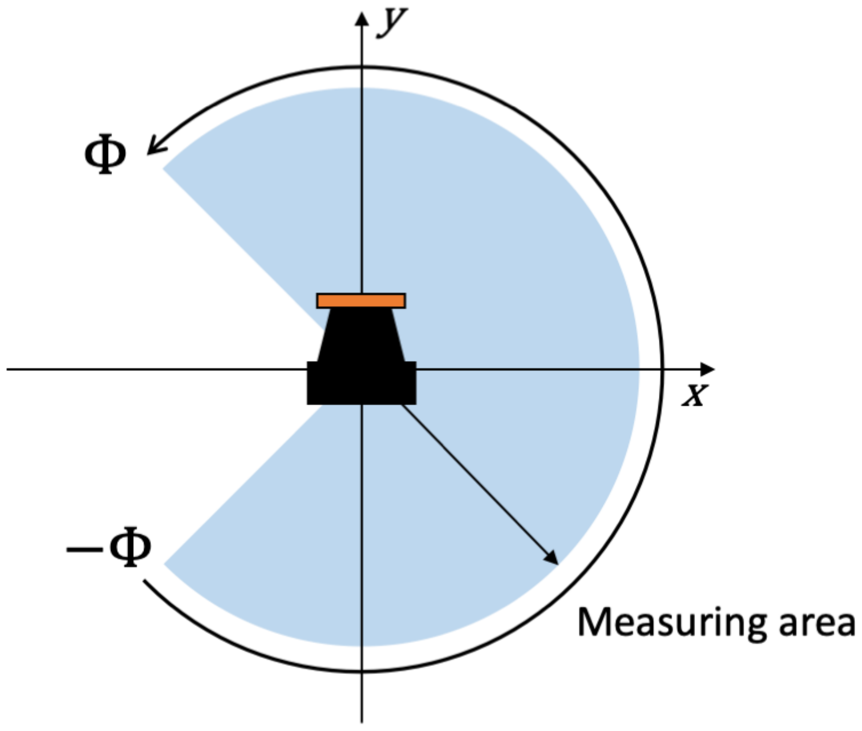


Figure 4.4: Coordinate system of LiDAR sensor

### 4.3.2 Door Status Detection

As described above, this method uses the background difference to detect the opening and closing of the door. In order to grasp the position of the door, the background data at the time of opening and closing when the passenger is not present is acquired, and these measurement data are expressed as $\mathcal{D}_{open}$ and $\mathcal{D}_{close}$, respectively. Then, as shown in Fig. 4.5, by calculating the difference between the measurement data $\mathcal{D}_{close}$ when the door is closed and the measurement data $\mathcal{D}_{open}$ when the door is opened, the set $\mathcal{I}_{door}$ of the azimuth steps existing in the door is calculated. The distance data $\mathcal{D}(i)$ $(i \in \mathcal{I}_{door})$ in the door direction when the door is opened is much larger than when the door is closed, so $\mathcal{I}_{door}$ is defined as shown in Equation (4.2). Here, $th_{door}$ is the threshold value of the difference in distance data in the door direction when the door is opened and when the door is closed, in evaluation, $th_{door} = 200\text{mm}$.

$$\mathcal{I}_{door} = \{i \mid \mathcal{D}_{open}(i) - \mathcal{D}_{close}(i) \geq th_{door}\} \tag{4.2}$$

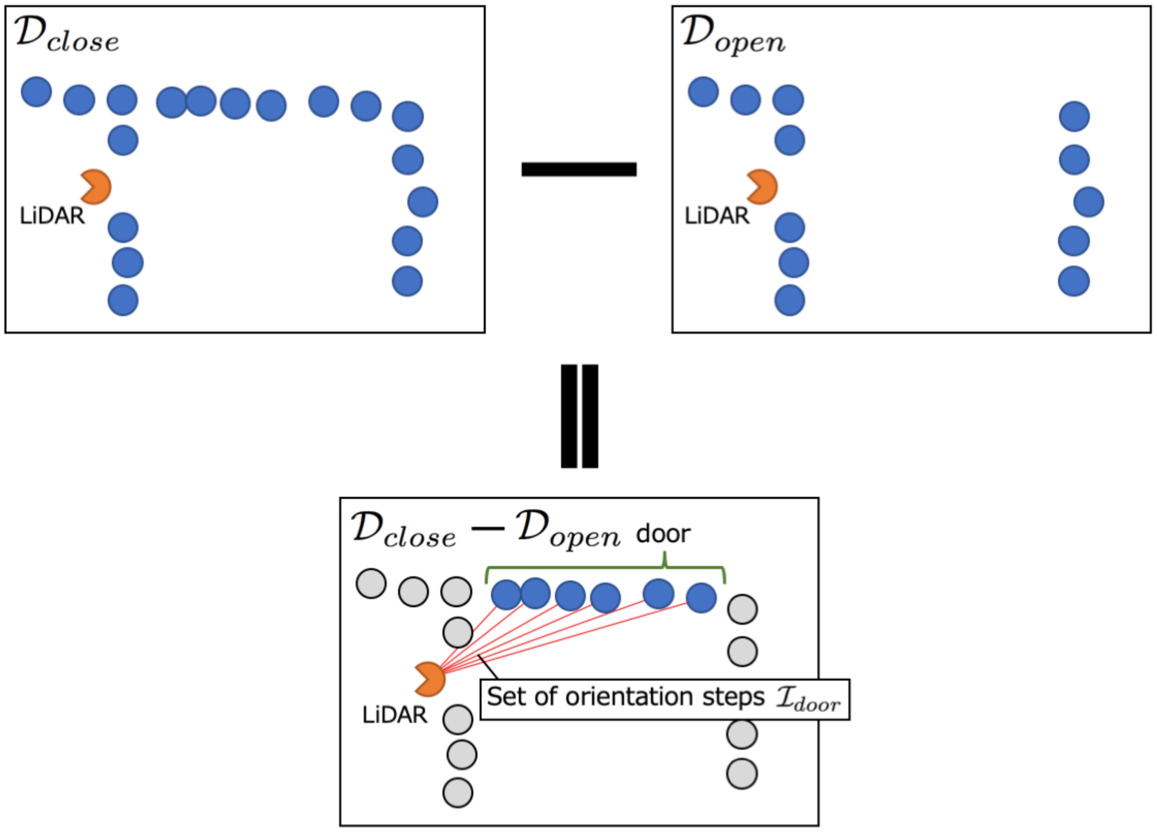

$\mathcal{D}_{close}$
LiDAR
$\mathcal{D}_{open}$
LiDAR
$\mathcal{D}_{close} - \mathcal{D}_{open}$
door
Set of orientation steps $\mathcal{I}_{door}$
LiDAR

Figure 4.5: Example of calculating $\mathcal{I}_{door}$

Using the feature that many point clouds outside the vehicle are detected when the door is open, door open/close detection is performed based on the number $N_{out}$ of point clouds outside the vehicle. Define the number of point clouds $N_{out}$ outside the vehicle of measurement data $\mathcal{D}$ using Eq. (4.3). Here, U(x) is a unit step function that is 1 for $x \geq 0$, and 0 for $x < 0$. $th_{outside}$ is the threshold value of the difference between the distance to the door and the distance to the detected point. If the measurement distance in the door direction is equal to or greater than the sum of the distance to the door when closed and $th_{outside}$, the measurement point is considered to be outside the vehicle.

$$N_{out} = \sum_{i \in \mathcal{I}_{door}} U\left((\mathcal{D}(i) - \mathcal{D}_{close}(i)) - th_{outside}\right) \tag{4.3}$$

The door state is expressed as the state transition between two states, “opened door state” and “closed door state”. As shown in Fig. 4.6, the number of points outside the vehicle tends to increase when the door is opened, and the number of points outside the vehicle tends to decrease when the door is closed. When the number is decreased, the door is opened or closed by making a transition to the closed state. However, as shown in Fig. 4.7, when passing through a person’s door when the door is opened, the number of point clouds outside the vehicle decreases, and there is a possibility that the door is erroneously recognized as closed. For this reason, there is no way to detect the opening and closing of the door using the number of point clouds outside the vehicle at the time of a certain measurement, and to reduce misrecognition by using the average over the past few seconds. On the other hand, if the averaging period is longer, the recognition error will decrease or decrease, but there will be a delay in the open / close detection. Does the detection delay when closing the door hinder the measurement of passengers? Does the detection delay when opening the door cause detection omission of people who get on and off during the period from when the door is actually opened to when the door is detected? is there. Besides, there is almost no measurement of the point cloud outside the vehicle when the door is closed, and no misrecognition of the transition to the open state occurs. This is set separately for detection, and the time span is greatly shortened when opening is detected, thus eliminating the above-mentioned problems.

The door state is expressed as the state transition between two states, “opened door state” and “closed door state.” As shown in Fig. 4.6, the number of points outside the vehicle tends to increase when the door is opened, and the number of points outside the vehicle tends to decrease when the door is closed. When the number is decreased, the door is opened or closed by making a transition to the closed state. However, as shown in Fig. 4.7, when passing through a person’s door when the door is opened, the number of point clouds outside the vehicle decreases, and there is a possibility that the door is erroneously recognized as closed. For this reason, I use a way to detect the opening and closing of the door using the number of point clouds outside the vehicle at the time of a certain measurement and to reduce misrecognition by using the average over the last few seconds. On the other hand, misrecognition decreases as the averaging period become longer, but there is a problem of delay in opening and closing detection. The detection delay of the closing of the door does not interfere with passenger measurement, but the detection delay of the opening of the door may cause omission of detection of the person who gets on and off between the time when the door is actually opened and the time when the opening is detected. The detection delay of the closing of the door does not interfere with passenger measurement. However, the detection delay of the opening the door may cause omission of detection of the person who gets on and off between the time when the door is actually opened and the time. Besides, there is almost no measurement of the point cloud outside the vehicle when the door is closed, and no misrecognition of the transition to the open state occurs. This is set separately for detection, and the time span is greatly shortened when the opening is detected, thus eliminating the above-mentioned problems.

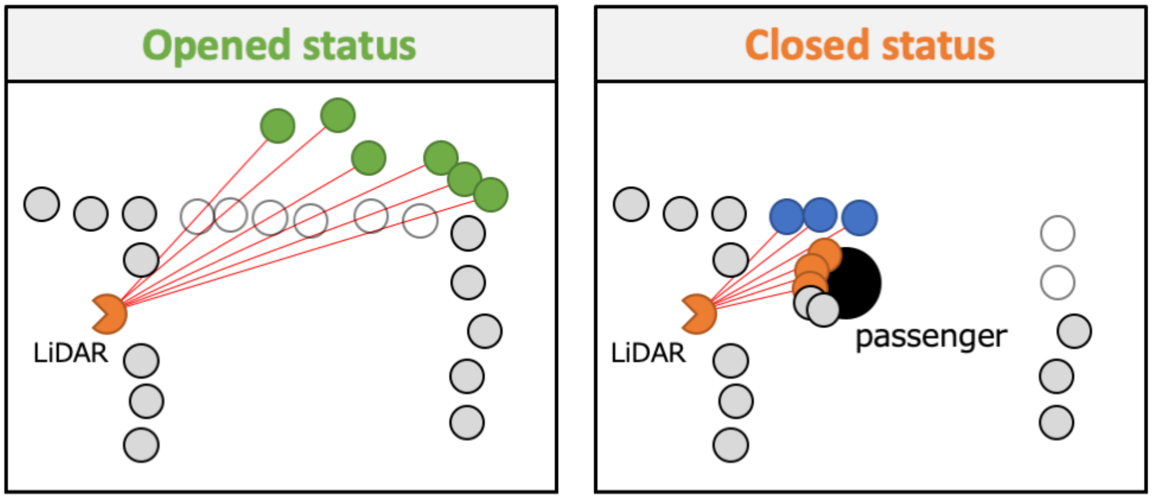


Figure 4.6: Example of measurement data when opening and closing the door

A concrete door open / close detection method is shown in Algorithm 3. $R_{open},\ R_{close} \in (0,1)$ is a coefficient that is the threshold value of the number of point clouds outside the vehicle, $N_{out}^{(t)}$ is $N_{out}$ at time $t$, and $N_{open}$ / $N_{close}$ is the average number of point clouds outside the vehicle when opening / closing is detected Represents the number of past frames. The performance evaluation was $R_{open} = 0.50,\ R_{close} = 0.25,\ N_{open} = 10$ (1sec. $\times$ 10Hz), $N_{close} = 50$ (5sec. $\times$ 10Hz).

A concrete door open/close detection method is shown in Algorithm 3. $R_{open},\ R_{close} \in (0,1)$ is a coefficient that is the threshold value of the number of point clouds outside the vehicle, $N_{out}^{(t)}$ is $N_{out}$ at time $t$, and $N_{open}/N_{close}$ is the number of frames in the past that averages the number of point clouds outside the vehicle when door opening/closing is detected. In evaluation, the evaluation was performed with $R_{open} = 0.50,\ R_{close} = 0.25,\ N_{open} = 10$ (1sec. $\times$ 10Hz), $N_{close} = 50$ (5sec. $\times$ 10Hz).

### 4.3.3 Passenger Measurement

This section describes a method for measuring passengers from the obtained point cloud data. When measuring passengers, it is necessary to detect a person in each frame. First, calculate the background difference of the measurement data $\mathcal{D}_{open}$ when the door is opened from the obtained measurement data $\mathcal{D}$, and let $\mathcal{D}'$ be the measurement data excluding the background. As shown in Fig. 4.7, laser beams that are adjacent to each other in the azimuth angle hit the same person, so points that are similar in azimuth angle or adjacent measurement distance are considered to represent the same person.

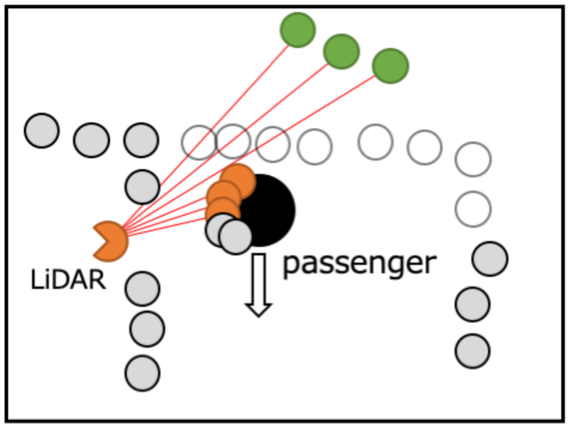


Figure 4.7: Example of false recognition occurring at door opening

However, as described in Section 4.2.2, the positioning surface of the range sensor is not necessarily located higher than the waist of the passenger, and there are cases where both feet are detected, as shown in Fig. 4.8. Therefore, there is a risk that a single passenger will be mistakenly recognized as two people, so it is necessary to allow a certain degree of azimuth between adjacent point clouds. For this reason, grouping measurement data that is somewhat similar in azimuth and similar in measurement distance, and if the number of point groups that have been grouped exceeds the threshold, this point group is regarded as a human point cloud. A specific grouping method is shown in Algorithm 4. Here, $th_D$ and $N_{skip}$ are the threshold values for the difference in measurement distance and direction step that are considered to be the same person, $th_{groupNum}$ is the threshold value for the number of point groups that are considered as humans, and $\mathrm{maxDist}(\mathcal{G})$ is a function that returns the maximum distance between given points, $th_{humanSize}$ is the minimum size to be considered as a person, $\mathcal{H}$ is a set of detected persons, and the elements of $\mathcal{H}$ represent the set of orientation steps that detected those persons. In evaluation, the evaluation was performed with $th_D = 200\text{mm}$, $N_{skip} = 5$, $th_{groupNum} = 10$, $th_{humanSize} = 200mm$.

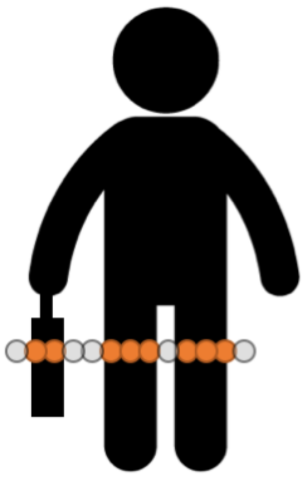
Then, my method tracks the person by using the detected set $\mathcal{H}$ of people. It is assumed that the position of the center of the point cloud obtained from the set $\mathcal{G}(\in \mathcal{H})$ of the azimuth steps is the position of the person. When the movement speed of the person is $v$ and the scanning time of the range sensor is $\Delta t$, the movement amount of the person between two consecutive frames is $v\Delta t$. In the set of people $\mathcal{H}^{(t)}$ and $\mathcal{H}^{(t+1)}$ between two consecutive frames, if the distance between the people $\mathcal{G}(\in \mathcal{H}^{(t)})$ and $\mathcal{G}'(\in \mathcal{H}^{(t+1)})$ detected in each frame is less than or equal to the threshold th, $\mathcal{G}$ and $\mathcal{G}'$ are considered to be the same person. If there are multiple people within the thtrack at this time, the person closest to the distance is regarded as the same person. If the line connecting the centroids of $\mathcal{G}$ and $\mathcal{G}'$, which are considered to be the same person, intersects the bus entrance/exit, the

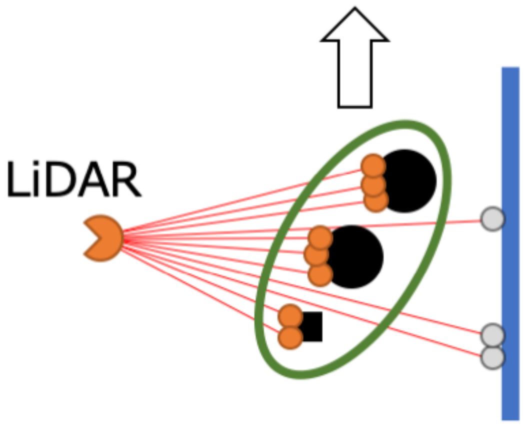
**Algorithm 3** Door status(opened/closed) detection

```
Require: $\mathcal{D}$, $\mathcal{I}_{door}$, $N_{out}$, $PrevState$
Ensure: $State$
  /* Initialization */
  $State \leftarrow PrevState$
  $N_{door} \leftarrow |\mathcal{I}_{door}|$

  /* Detection */
  if State = "Opened" then
    if $R_{close} N_{door} > \frac{1}{N_{close}} \sum_{k=0}^{N_{close}} N_{out}^{(t-k)}$ then
      $State \leftarrow$ "Closed"
      Stop ingress and egress measurement
    end if
  else if State = "Closed" then
    if $R_{open} N_{door} < \frac{1}{N_{open}} \sum_{k=0}^{N_{open}} N_{out}^{(t-k)}$ then
      $State \leftarrow$ "Opened"
      Start ingress and egress measurement
    end if
  end if
```

boarding/exiting is counted according to the direction of travel. Besides, when a person stops near the door, the same person may go back and forth many times due to the noise of the measurement. Even in such a case, in order to avoid counting again, it is necessary to give all tracking persons whether or not they have passed the entrance/exit. It is possible to prevent the above-mentioned false detection by not counting even if counted passenger passes. It is possible to prevent the above-mentioned false detection by not counting even if it passes. In the performance evaluation, the moving speed of the person in the bus was assumed to be 1m/s, and the tracking threshold was set to $th_{track} = 300$mm considering the scanning time of 100ms and measurement noise.


(a) Side View



(b) Overhead View

Figure 4.8: State of measuring point of leg

**Algorithm 4** Human Detection

```
Require: $\mathcal{D}, \mathcal{D}'$
Ensure: $\mathcal{H}$
  /* Initialization */
  $\mathcal{G} \leftarrow \emptyset$ /* Temporary New Group */
  $PrevI \leftarrow$ NULL
  for all $i$ such that $0 \leq i < N_{scan}$ do
    if $\mathcal{D}(i) \notin \mathcal{D}'$ then
      /* $\mathcal{D}(i)$ is background */
      continue;
    end if

    if $PrevI \neq \text{NULL} \ \wedge \ |\mathcal{D}(PrevI) - \mathcal{D}(i)| < th_D \ \wedge \ (i - prevI) \leq N_{skip}$ then
      /* add to $\mathcal{G}$, if measurement distance and azimuth are similar */
      /* add previous group */
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{i\}$
      $PrevI \leftarrow i$
    else
      /* $\mathcal{D}(i)$ is new group */
      if $|\mathcal{G}| \geq th_{groupNum} \wedge \text{maxDist}(\mathcal{G}) \geq th_{humanSize}$ then
        /* $\mathcal{G}$ is human */
        $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{G}\}$
      end if
      /* create new group */
      $\mathcal{G} \leftarrow \{i\}$
      $PrevI \leftarrow i$
    end if
    if $|\mathcal{G}| \geq th_{groupNum} \wedge \text{maxDist}(\mathcal{G}) \geq th_{humanSize}$ then
      /* $\mathcal{G}$ is human */
      $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{G}\}$
    end if
  end for
```

## 4.4 Evaluation

### 4.4.1 Environment

A demonstration experiment was conducted on the Osaka University campus shuttle bus operated by Hankyu Bus Co., Ltd., and the performance of the proposed method was evaluated. As indicated by the red circles in Fig. 4.9, measuring devices were installed near the front, and rear entrances of the shuttle bus and the state of getting on and off ware measured for 3 days (145 times of getting on and off). When getting on a connecting bus, only the entrance at the rear is used, and when getting off, both entrances are used. The laser range sensor is URG-04LX-UG01 made by Hokuyo Automatic Co., LTD. The Raspberry Pi 3 Model B of Raspberry Pi Foundation was used for the single-board computer. Table 4.2 shows the performance of Raspberry Pi 3 Model B. As a performance evaluation, the estimated value of the number of passengers in each period was compared with the ground truth, with one period from the time the door of the shuttle bus opens until it closes. As a comparison method, I compare it with a naive method that imitates the method using two infrared sensors side by side. In the comparison method, two measurement data of the azimuth angle $\pm 10^\circ$ of the range sensor used in the experiment was used as the measurement data of the infrared sensor. When the two measurement data detect an object within 0.5 seconds, the boarding or getting off is measured according to the azimuth angle at which the detection time was earlier. As an evaluation of the computational complexity of the proposed method, the processing time of each frame was measured, and it was verified whether it was within 100ms, which is the scanning time per measurement of the range sensor.

Table 4.2: Spec of Raspberry Pi 3 Model B

| Item | Spec |
|---|---|
| OS | Raspbian 8.0 Jessie |
| CPU | Quad Core 1.2GHz Broadcom BCM2837 64bit |
| RAM | 1GB |
| Power Consumption | 7W (Max: 12.5W) |
| Weight | 45g |

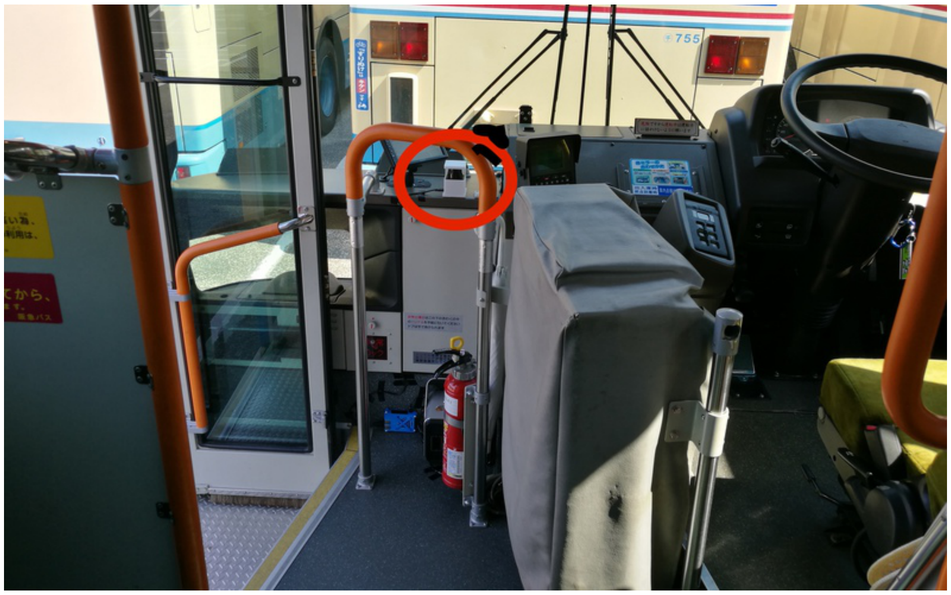
### 4.4.2 Results

Figures 4.10 and 4.11 show the results of estimating the number of passengers for each period using the proposed method and the comparison method. The horizontal axis of the graph represents the ground truth, the vertical axis represents the estimated value, and the red line is the straight line "ground truth = estimated value." As shown in Fig. 4.11, in the comparison method using an infrared sensor, the same person was measured twice under the influence of passengers who stopped near the door, and the estimation was more than the ground truth overall. Besides, the error rate is 25% in the estimation of the number of passengers, and the error rate is 17% in the estimation of the number of passengers, so it cannot be said that the number of passing passengers can be estimated accurately. On the other hand, as shown in Fig. 4.10, in the proposed method, most of the plots are near the straight line, indicating that they can be estimated accurately. The number of passengers can be estimated with high accuracy, with an error rate of 5.3% and an average absolute error of 0.94, and an estimated passenger count of 7.5% and an average absolute error of 1.9 people. The PT survey described in Section 4.1 statistically guarantees a relative error of 20% or less, and the proposed method dramatically exceeds that. Therefore, the proposed method is sufficiently useful. However, in the estimation of the number of people getting off the rear door (Fig. 4.10(b)), the estimated value when the number of people getting off was 66 people was 55 people, resulting in an error of 11 people. In this period, there were more than 80 passengers in the car, and the passengers were crowded. Figure 4.12 shows the state before the bus stops, and the door opens. A red square represents the position of the laser range sensor, a blue point represents the background data measured by the laser range sensor, a red point represents the person, and a green point represents the door. In addition, a purple line represents the outline of the bus estimated from these results is represented, an orange line represents the rear door, and a yellow circle represents the person riding near the door. In this way, in a situation where a large number of passengers are on board, the passengers near the door get off at once when the rear door opens. Therefore, the proposed method recognizes two close persons as one person, and it is considered that a detection failure occurred. The phenomenon of passengers close to the doors is likely to occur when the bus is full and gets off, but it is unlikely to be full when there are passengers near the doors. However, it can be considered that there is no problem in practical use because it is only necessary to grasp that the number of passengers getting on and off is low so that it can not be measured accurately at a stop where a large number of passengers get on and off. On the other hand, when getting on a bus, there are many cases where people line up one by one, and there are few false detections.

Table 4.3 shows the processing time per frame of the proposed method. The maximum processing time for one frame was 12.5ms, but the scanning time of the range sensor was less than 100ms. This result shows that real-time measurement is sufficiently possible even with an inexpensive and small single-board computer such as Raspberry Pi 3 Model B. The frame with the longest processing time was characterized by a large number of passengers and a small number of background point clouds. This is because when the point cloud of objects other than the background increases, it takes time to detect and track passengers. However, the demonstration experiment includes a boarding/alighting case in a full car, and the processing time will not increase any further. From the results of these evaluation experiments, this system achieves high estimation accuracy with a low processing load and can be said to be a highly practical approach.

Table 4.3: Result of one frame processing time

| Item | Result |
|---|---|
| Maximum processing time | 12.5[ms] |
| Average processing time | 3.4[ms] |


(a) Front door

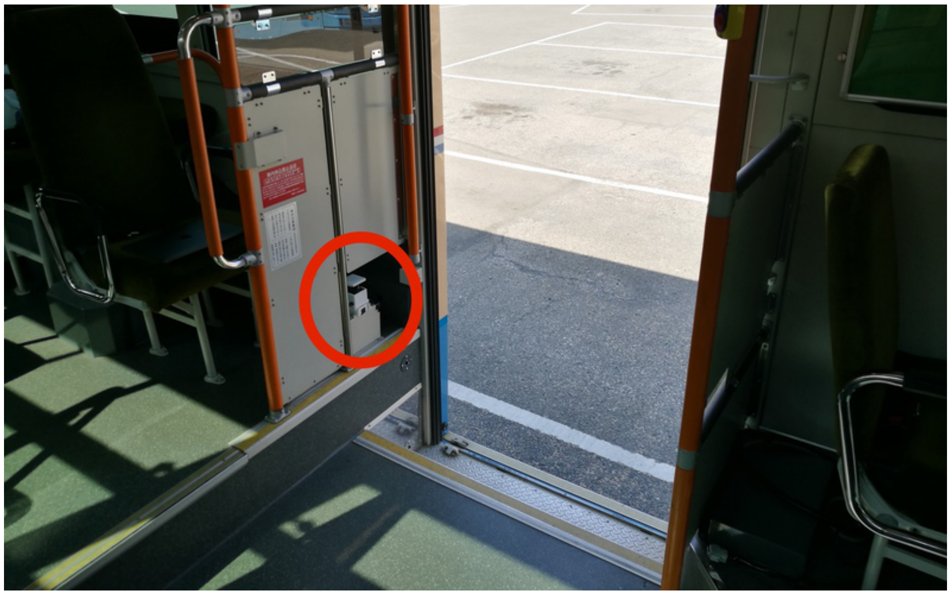
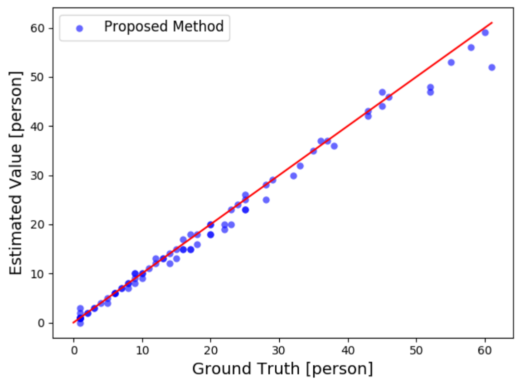
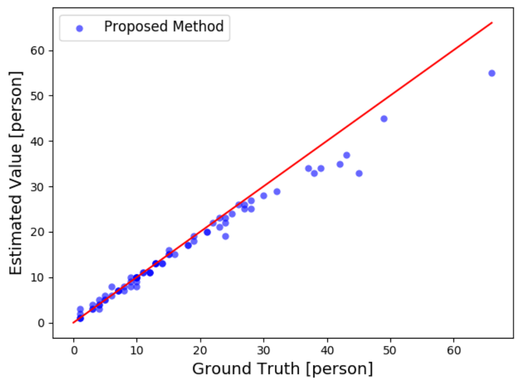

(b) Rear door

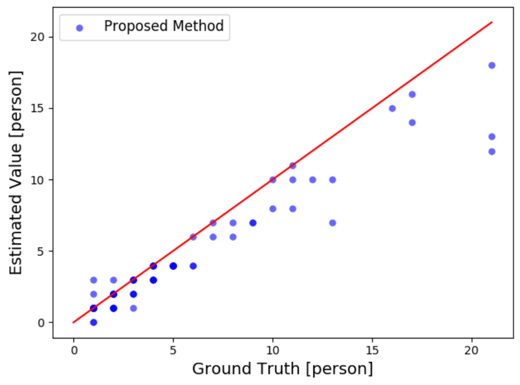
Figure 4.9: State of installation of measuring system



(a) Getting on (Rear door)

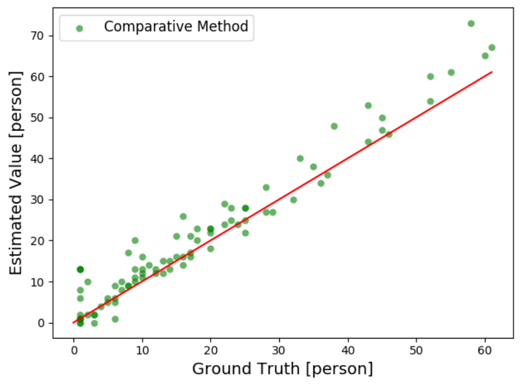


(b) Getting of (Rear door)

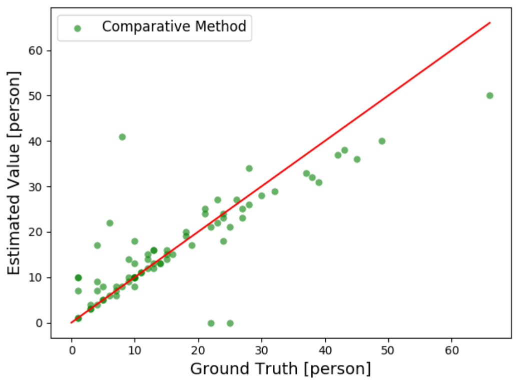


(c) Getting of (Front door)

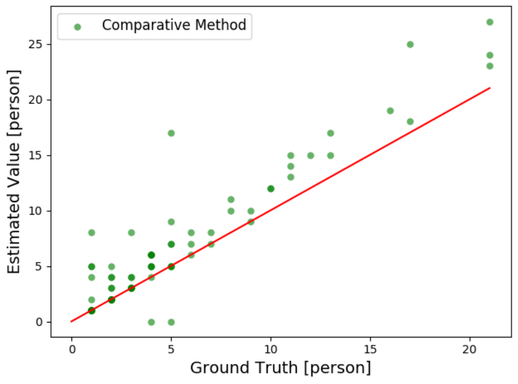
Figure 4.10: Result of passenger estimation by proposed method



(a) Getting on (Rear door)



(b) Getting of (Rear door)



(c) Getting of (Front door)

Figure 4.11: Result of passenger estimation by comparative method

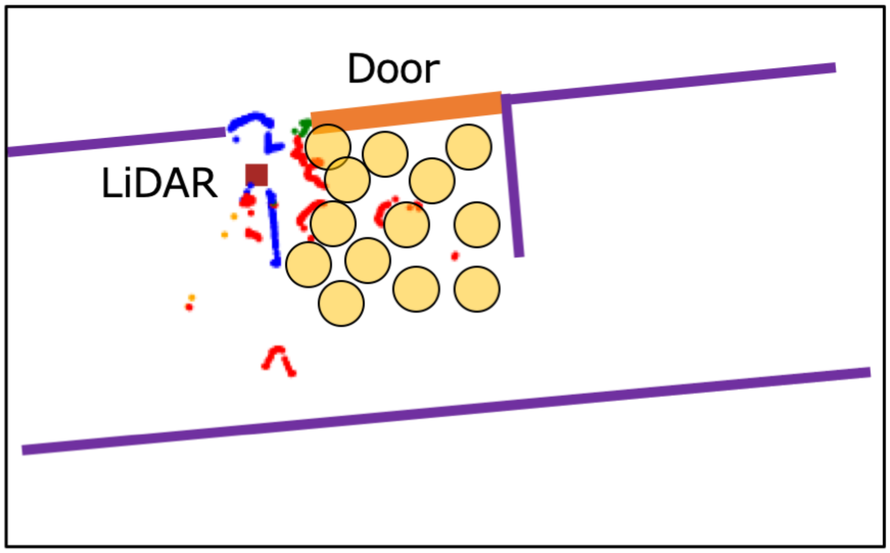


Figure 4.12: State where many passengers are densely near the entrance

## 4.5 Conclusion

In this chapter, in order to reduce the labor and cost barriers related to the introduction and realize high-accuracy detection, I proposed a high-accuracy route bus boarding / alighting measurement system that uses real-time processing of measurement data obtained from range sensors on an inexpensive single-board computer. I have established a method for accurately detecting people and measuring boarding/exiting situations under various situations where the body part to be measured depending on the floor on which the passenger stands. In addition, as a result of conducting a 3-day demonstration experiment on a bus that is actually operating, an error rate of 5.3% (average absolute error of 0.94 people) was achieved, and an error rate of 7.5% (average absolute error of 1.9 people) was achieved when estimating the number of people getting off. From this result, it was shown that the proposed method could measure boarding/exiting with very high accuracy compared with the collateral accuracy of 20% in the PT survey. Furthermore, since this method is excellent in portability and can be used both indoors and outdoors, it can be expected to be used for measuring the number of visitors to the event hall in addition to measuring passengers on the route bus.

A future issue is the construction of a technique that is resistant to occlusion caused by overlapping passengers. In the proposed method, the occurrence of occlusion is reduced by devising the installation position of the range sensor, but the occurrence of occlusion is inevitable when the interior of the vehicle is very crowded. For this reason, in human tracking estimation, I consider a method to prevent occlusion detection omission by complementing the position of a person who can no longer be detected by occlusion in consideration of the direction and speed of movement of the person.

In this thesis, the performance evaluation was conducted for university students, but performance evaluation for various passengers such as parents with children, people with baby strollers, walking sticks, etc. has not been performed. Therefore, I are currently conducting a demonstration experiment on an actual route bus for actual operation.

# Chapter 5

# Route Bus OD Measurement System Using Camera Image

## 5.1 Introduction

In recent years, people's movement information by city traffic has become increasingly necessary as primary data for MaaS (Mobility as a Service) for city communities. In MaaS, it is important to understand the specific traffic demand of people at a micro-level and to construct a multimodal transportation system efficiently. In addition, if a specific demand for travel can be grasped, it will contribute to effective measures to realize a safer and more prosperous community (e.g., optimizing the route and operation of existing transportation systems, and activating the use of unprofitable routes by holding events along the route).

In particular, it is extremely important to understand the use of route buses, which are the central axis of public transport, in local cities, suburbs, and depopulated areas, in order to understand the use of local public transport and to realize correct improvement processes. In particular, measurement of passenger arrival and departure points is valuable data for transport operators who cannot accurately grasp the number of passengers. Besides, for example, a large number of passengers get off at a time on a congested route, and the stoppage time at the stop increases, or many tourists are confused by cash use and payment methods, and the time of getting off per person increases. It can be grasped by measuring and analyzing the time required for getting on and off. Such data can be an indicator of whether a smooth operation has been realized, and can be used as the primary data for business operators to make improvements.

In major cities, person trip surveys (PT surveys) are conducted about once every 10 years in order to grasp the actual use of people's transportation systems. However, PT surveys rely on human resources. The cost is high, and it is difficult to grasp the behavior of people that change every moment according to time, day of the week, and season. In local cities, the PT survey itself is not implemented. On the other hand, it is not impossible to conduct a bus trip survey that includes the origin-destination

(OD) by using the usage history of the transportation IC card. However, especially in regional traffic with deficit routes, IC cards are often not introduced for management reasons, and even if they are introduced, they are often limited to some routes. Besides, it is impossible to measure passengers who do not have an IC card.

A system using an infrared sensor has been developed to measure passengers getting on and off the route bus [37]. By using two infrared sensors side-by-side and using the difference between their passage detection times, it is possible to count the number of passengers getting on and off for each ride. However, when multiple people pass through the sensor at the same time, it cannot be measured accurately, and the time required for getting on and off cannot be measured. My research group has proposed a lightweight and high-accuracy route bus getting on/off measurement system that processes the detection data of LiDAR (Light Detection and Ranging) in real-time with a Raspberry Pi [57]. However, methods that use infrared rays, including this system, are not intended for OD measurement.

In this chapter, I propose a low-cost OD (getting-on/off the stop) measurement system that detects the OD of each passenger in real-time using only an inexpensive single-board computer (e.g., Raspberry Pi 3) and an image of a single-camera installed at a position overlooking the interior of a bus such as a route bus. The proposed method recognizes passengers in the vehicle during driving between the stops (hereinafter referred to simply as "driving") and compares passengers in the vehicle before and after the stops. Identify new passengers aboard the stop. This makes it possible to measure the OD of each passenger.

The main problem with using camera images is the occlusion problem where passengers far from the camera cannot be seen due to overlapping passengers. Also, detection and recognition may fail to depend on the face angle and light. Therefore, the proposed method suppresses these effects as much as possible by using multiple camera images taken at regular intervals (several seconds) during each operation. On the other hand, since it is necessary always to identify the same person appearing between multiple camera images, I propose a lightweight and highly accurate algorithm to determine whether two-person images are the same person. The algorithm constructs a feature extraction model that calculates feature vectors from human face images using deep learning using public datasets. The person's face image shown in the camera image is detected, and the feature vector of the face image is compared to determine whether or not they are the same person.

As a related study, a system has been developed and commercialized to detect passengers by installing an RGB camera on the ceiling near the bus entrance and processing images [40]. In such a system, since the camera is installed from the ceiling to the floor, it is possible to avoid occlusion caused by overlapping passengers in the camera view. However, these are different from the proposed method because the purpose is to measure the number of passengers getting on and off and not OD measurement. As far as I know, a system that realizes OD measurement in real-time with a single camera has not been proposed.

In order to evaluate the effectiveness of the proposed method, the performance was evaluated using camera images taken in an environment that reproduced an actual route bus. As a result of creating 50 kinds of learning models and evaluating their performance using Raspberry Pi 3, the best model achieved AUC = 0.854 and processing time 0.70s. From this result, it was shown that it could be processed at the same high speed and accuracy as the conventional face image identification system.

## 5.2 System Overview

Figure 5.1 shows an overview of the system proposed in this study. A combination of an RGB camera, a single board computer (hereinafter simply called a computer for simplicity), and a GPS sensor is used as one measurement system. The measurement system is installed at an angle so that it can be seen from the front ceiling of the route bus to the rear. In order to reduce the amount of data transferred, RGB cameras and computers are connected to each measurement system. Record the estimated OD measurement information after image processing by computer. In addition, a bus stop is detected from a GPS sensor, the bus stop, and OD information are uploaded to the data server, and the getting on/off information for each bus stop is aggregated.

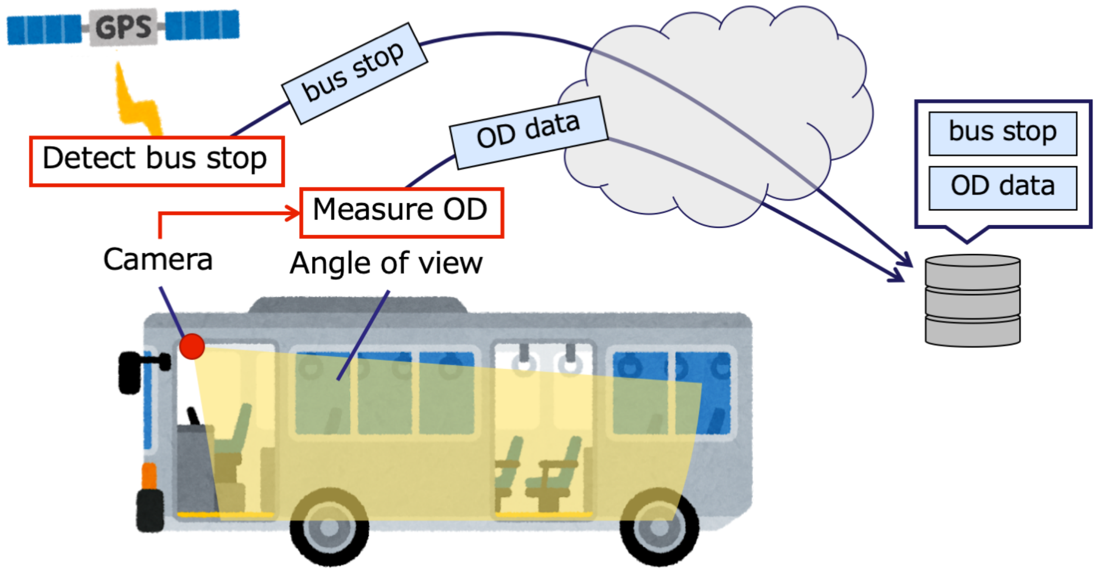


Figure 5.1: System Overview

## 5.3 Proposed Method

Figure 5.2 shows an overview of the proposed method. First, the door open/close state is obtained from the bus vehicle. If it is closed, the bus is considered to be driving (moving between stops), and the passenger detector is activated. In the passenger detector, the in-vehicle image is acquired from the RGB camera installed on the route bus. Then, a face detector is applied to the in-car image to extract all the face images of the person (passenger) in the car and the position on the image, and the extracted face image information is stored in the face image memory (RAM on the computer). This series of operations, from acquiring the in-vehicle image to adding it to the face image memory is repeated while the bus driving flag is ON. When the door of the bus is opened and getting on/off is started, all face image data detected during operation stored in the face image memory is input to the person classifier, and the face image of the same person is input. Each group is grouped, and the result is recorded in the measurement data memory (RAM) together with the stop at that time. Then, when the operation of one route is completed, the measurement data stored in the measurement data memory is input to the OD aggregator, and the OD of each passenger (the stop where the passenger got on and the stop where the passenger got off) is calculated. Send it to the data server as OD information. The following sections describe the details of each mechanism.

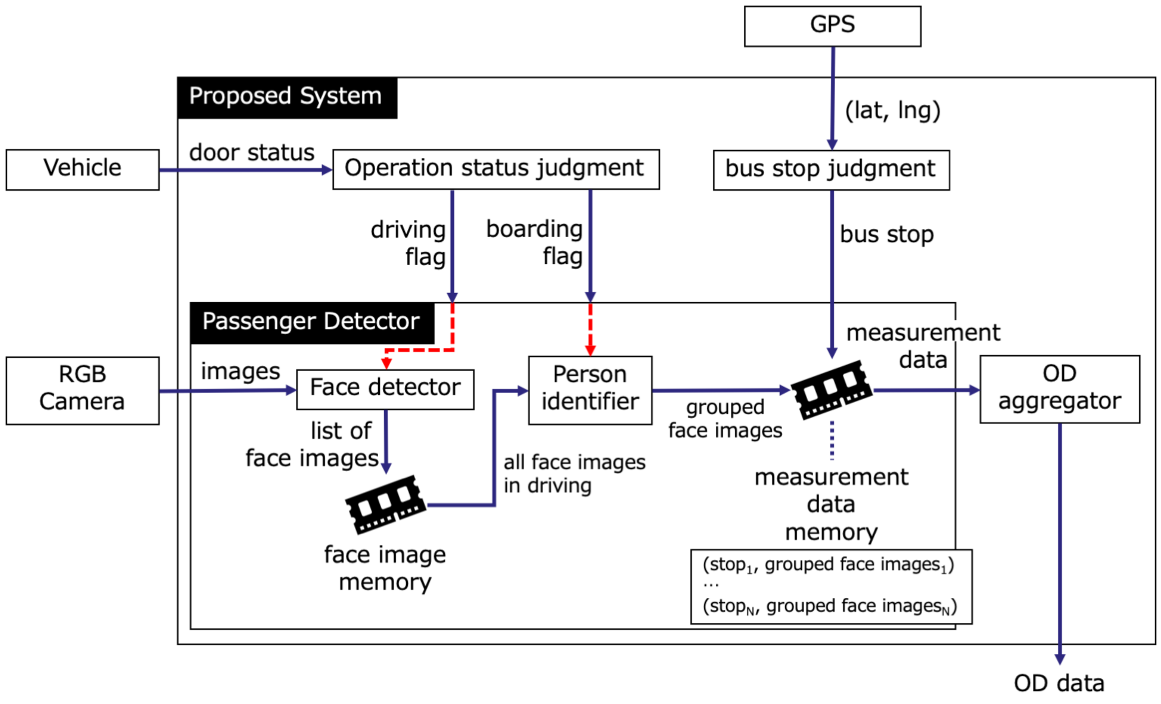


Figure 5.2: Overview of proposed method

### 5.3.1 Face Image Similarity Calculation Method

In this section, I describe a method of calculating the similarity used to determine whether or not a person is the same from two face images. Figure 5.3 shows the similarity calculation model proposed by this method. First, the two input face images are input to the feature extraction model, and the feature vector is calculated. Then, the two feature vectors are compared, a feature comparator that calculates the similarity is applied, and the feature quantities of the two face images are calculated. Section 5.3.1.1 describes the feature extraction model, and Section 5.3.1.2 describes the details of the feature comparator.

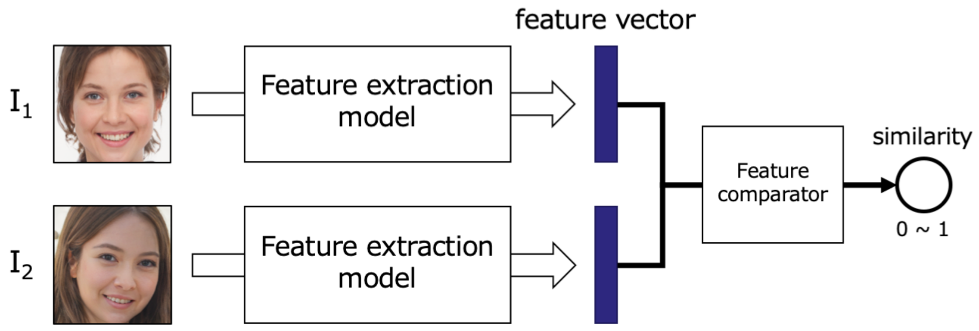


Figure 5.3: Face image similarity calculation method model

#### 5.3.1.1 Feature Extraction Model

This section describes a model that extracts the features of the input facial image. In order to operate on an inexpensive computer, it is necessary to build a lightweight model while maintaining a certain degree of accuracy. Therefore, in this study, a feature extraction model is realized by deep learning using CNN. Figure 5.4 shows the realized network model. The white layer represents the convolutional layer of kernel size $3 \times 3$ + activation function (ReLU), the red layer represents the Max Pooling layer, and the blue layer represents the fully connected layer + activation function (ReLU). The feature extraction model itself expresses features by connecting three feature extraction layers that connect the convolutional layer and the MaxPooling layer, and connecting all connected layers to the lowest layer. In Fig. 5.4, $L_i$ $(i \in 1,2,3,4)$ represents the number of convolutional layers and all coupling layers, and $D_i$ $(i \in 1,2,3,4)$ represents the number of channels in each layer. In the performance evaluation in Section 5.4, several parameters were created by changing these parameters, and the performance was evaluated.

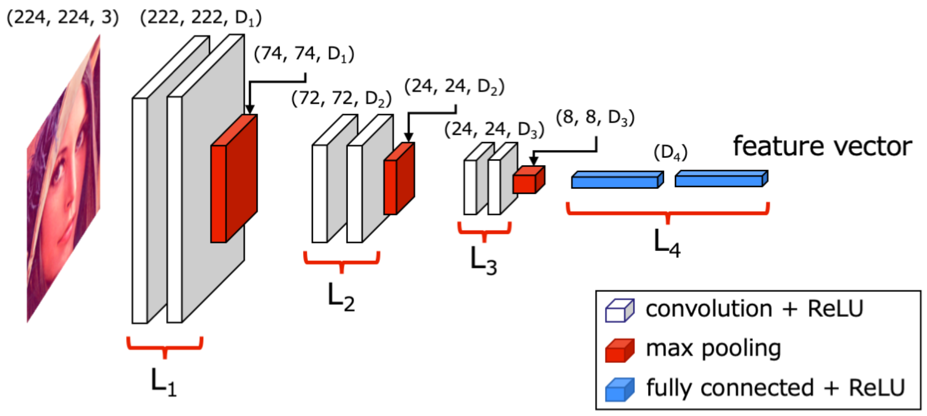


Figure 5.4: Feature Extraction Model

#### 5.3.1.2 Feature comparator

The two feature comparators employed in this study are shown in Fig.5.5(a), (b). The model in Fig.5.5(a) is a model that applies cosine similarity that represents the similarity of vector space models. Cosine similarity expresses the closeness of the angle between two vectors and is often used for classification problems such as sentences. The model in Fig.5.5(b) is a DNN model that expresses the similarity by applying two fully connected layers to the absolute value of each feature difference. By learning the parameters of this fully connected layer at the same time as the feature extraction model in Section 5.3.1.1, a comparator specialized for comparing facial images can be expected.

### 5.3.2 Person identifier

In this section, I describe a method for grouping and outputting face images detected from multiple in-vehicle images obtained during operation. The number of images (total number of frames) obtained during operation is $T$, the number of people detected in each frame $t$ $(0 \leq t < T)$ is $N^{(t)}$, and each face image is $I_i^{(t)}$ $\left(0 \leq i < N^{(t)}\right)$. Ideally, it is desirable from the aspect of accuracy to compare the similarity of the feature values of the face images of all frames obtained during operation. However, in this method, the emphasis is placed on reducing the calculation time. Only the face images detected in two consecutive frames are compared, and the similarity is calculated. This is performed between all consecutive frames, and finally, the face images of all frames are grouped. In the following, I describe a method for matching face images detected in two consecutive frames $t-1$ and $t$. First, the similarity between all face images detected in the previous and next frames is calculated. Since the passenger does not move significantly in the bus car in operation, the position of the detected face image hardly changes. Therefore, the similarity is calculated by combining the similarity of the face image itself and the similarity of the position in the frame described later (Section 5.3.1). As a result, even if the accuracy of the face image similarity calculator is somewhat rough, the accuracy of the system is improved by correcting the similarity calculation by using position information. Where the similarity of the face image is $\mathrm{sim}_f(I_a^{(t-1)}, I_b^{(t)})$ and the similarity of the position in the frame is $\mathrm{sim}_l(I_a^{(t-1)}, I_b^{(t)})$, the composite similarity $\mathrm{sim}(I_a^{(t-1)}, I_b^{(t)})$ of the face images $I_a^{(t-1)}$ and $I_b^{(t)}$ is defined as in equation (5.1). Here, parameter $\alpha (0 \leq \alpha \leq 1)$ represents the weight of facial image similarity.

$$\mathrm{sim}(I_a^{(t-1)}, I_b^{(t)}) = \alpha \mathrm{sim}_f(I_a^{(t-1)}, I_b^{(t)}) + (1-\alpha)\mathrm{sim}_l(I_a^{(t-1)}, I_b^{(t)} \tag{5.1}$$

Also, the similarity $\mathrm{sim}_l(I_a^{(t-1)}, I_b^{(t)})$ of the face image position in the frame is defined as shown in Equation (5.2) using the distance $\mathrm{dist}(I_a^{(t-1)}, I_b^{(t)})$ between the two face images.

$$\mathrm{sim}_l(I_a^{(t-1)}, I_b^{(t)}) = 1 - \frac{\mathrm{dist}(I_a^{(t-1)}, I_b^{(t)})}{\max_{I_i^{(t-1)}, I_j^{(t)} \in I}(\mathrm{dist}(I_i^{(t-1)}, I_j^{(t)}))} \tag{5.2}$$

Based on Eq. (5.1), the composite similarity between all face images is calculated, and the face images between the previous and next frames are matched so that the sum of the similarities is the largest. The grouping of face images of the same person is completed by aggregating the matched images between all previous and next frames. Finally, each grouped face image is output, and the grouping information is stored in the measurement data memory in Fig. 5.2.

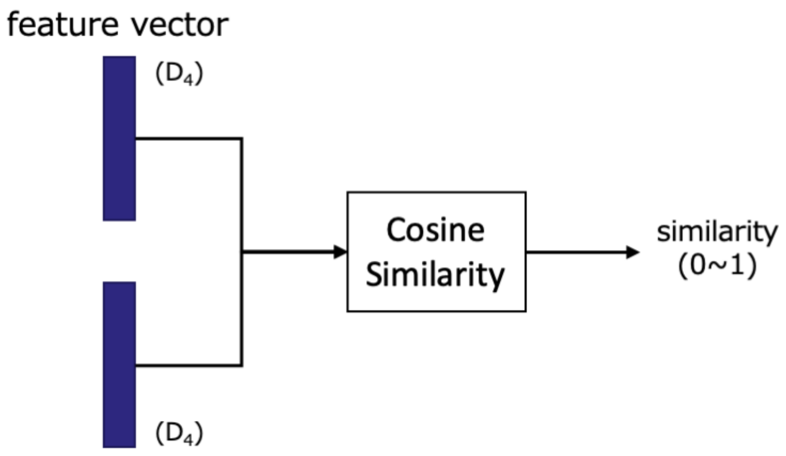
### 5.3.3 OD Estimation

This section describes the method for estimating the final passenger OD. Figure 5.6 shows an overview of OD estimation. By calculating the difference between passengers in the vehicle before and after getting on and off at any stop, it is possible to identify passengers who got on and off at that stop.

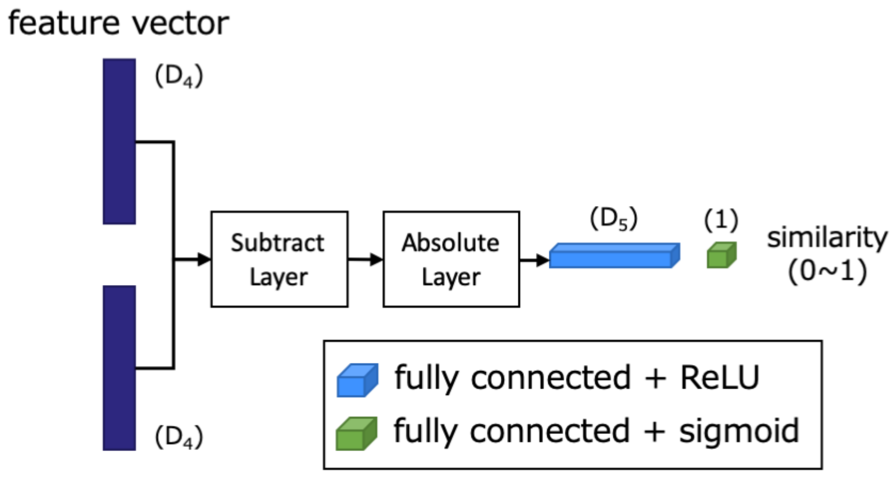
A specific method for OD estimation is described. I define the similarity score $\mathrm{sim}(p, q)$ of the two people as in Equation (5.3), where $p$ is any passenger before getting on and off and $q$ is any passenger after getting on and off. Passengers here represent the passengers grouped in section 5.3.2, and $p_{last}, q_{first}$ represent the last facial image and the first facial image in time among the grouped facial images. $\mathrm{FaseSim}(p, q)$ is a function that returns the similarity of face images in section 5.3.1, and $th$ is the similarity threshold that is considered the same person.

$$\mathrm{sim}(p, q) = \begin{cases} \mathrm{FaseSim}(p_{last}, q_{first}) & (> th) \\ -1 & (\text{otherwise}) \end{cases} \tag{5.3}$$

Let $P^{(s-1)}, P^{(s)}$ be the set of passengers in the vehicle before and after getting on and off at the stop $s$. Based on equation (5.3), the similarity score between each passenger is calculated, and a similarity score matrix $M$ of $\left|P^{(s-1)}\right|$ rows and $\left|P^{(s)}\right|$ columns is created. Then, select the cell that maximizes the sum of the cells under the constraint of at most one cell from each row and column of the matrix $M$. At this time, if the cell in the $i$-th row is not selected, it means that the passenger $p_i \in P^{(s-1)}$ has got off at the stop $s$, and if the cell in the $j$-th row is not selected, the passenger $p_j \in P^{(s)}$ got in at the stop $s$. It means that. If cell $(i, j)$ is selected, it means that passengers $p_i \in P^{(s-1)}$ and $p_j \in P^{(s)}$ are the same person. The problem of maximizing the sum of similarities is equivalent to the assignment problem, and can be solved using the Hungarian method (computation $O(\max\left(\left|P^{(s-1)}\right|, \left|P^{(s)}\right|\right)^3))$.



(a) Cosine similarity



(b) DNN

Figure 5.5: Feature comparison model

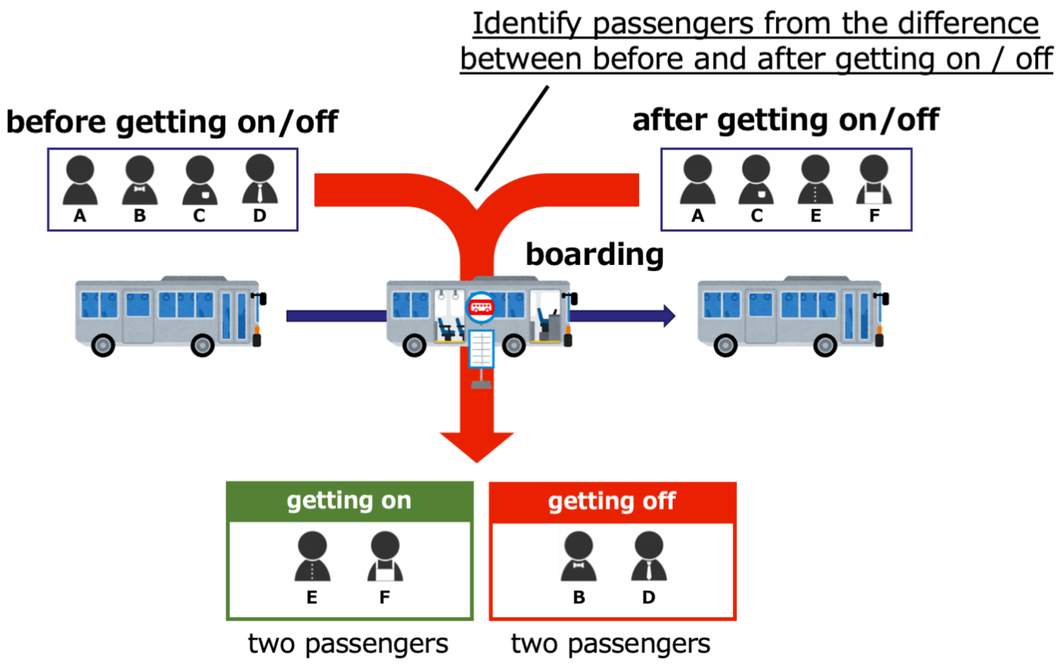


Figure 5.6: Overview of OD estimation

## 5.4 Evaluation

### 5.4.1 Dataset

#### 5.4.1.1 Training Dataset

The purpose of this study is to propose a facial image comparison model for a large number of unspecified people. Therefore, VGFace2 [45], which includes various ages and races, was adopted as training data. From the VGGFace2 data set, I randomly extracted 500 people with more than 100 face images. I extracted 12,800 positive examples with two face images of the same person as input and 12,800 negative examples with two face images of different persons as input, and used this as a training data set. Besides, the brightness of the inside of a route bus varies greatly depending on the presence or absence of external light such as sunlight. Therefore, a gamma correction that adjusts the brightness of the image was applied to the training data image at random in the parameter $\gamma$ range of $0.8 - 1.25$.

#### 5.4.1.2 Validation Dataset

A camera was attached to the route bus, the same situation as an actual route bus was reproduced, an experiment was performed, and a verification data set was generated using the camera image obtained in the experiment. For the performance evaluation of the facial image similarity calculation method (Section 5.4.3.1), the performance evaluation was performed using 269,745 combinations of all 13 face images (735 images) obtained as a verification data set. Figure 5.7 shows the experimental scene.

### 5.4.2 Verification Model

As the face detector in Fig. 5.2, I used YOLO Face [58] with YOLOv3 [59] tuned for face detection. Tables 5.1 and 5.2 show the parameters $L$ and $D$ of the feature extraction model of the proposed method described in Section 5.3.1.1. In addition to the 25 combinations of all these $L$ and $D$, a total of 50 types of feature extraction models were applied that applied two dropout rates of $0.10, 0.15$ to prevent overlearning.

Table 5.1: Parameter $L$ of feature extraction model

| Model Type | $L_1$ | $L_2$ | $L_3$ | $L_4$ |
|---|---|---|---|---|
| A | 2 | 2 | 2 | 2 |
| B | 1 | 2 | 2 | 2 |
| C | 1 | 1 | 2 | 2 |
| D | 1 | 1 | 1 | 2 |
| E | 1 | 1 | 1 | 3 |

Besides, the cosine similarity and DNN feature comparison models were used as the feature comparator described in Section 5.3.1.2. In the DNN feature comparison model, the number of channels $D_5$ in the total coupling layer is the same as the parameter $D_4$.

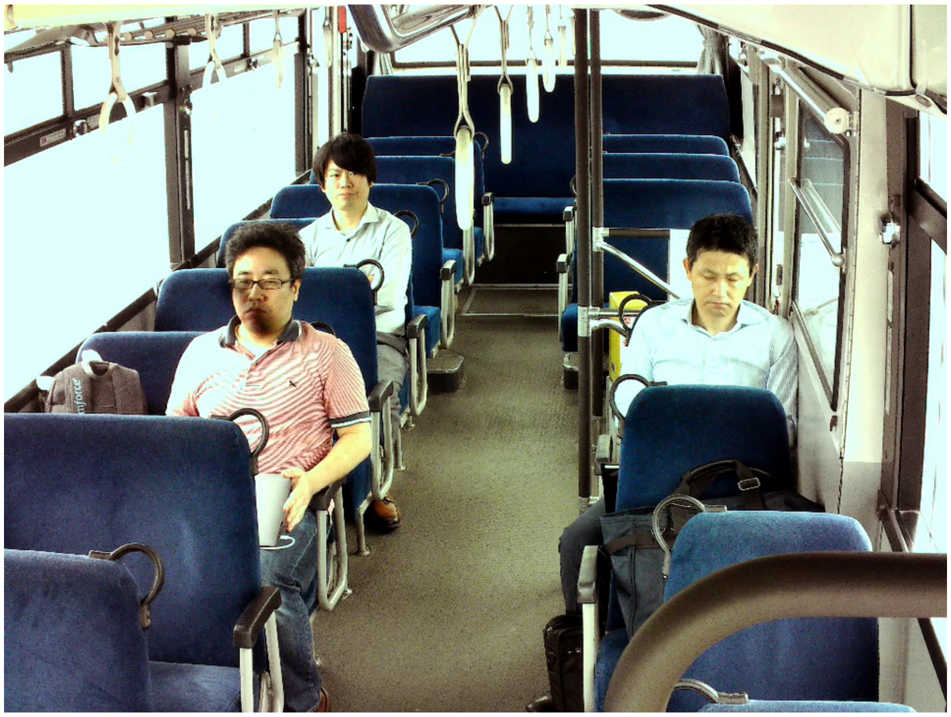


Figure 5.7: Experimental scene (Subject is a collaborator and does not include general passengers)

Table 5.2: Parameter $D$ of feature extraction model

| Channel Type | $D_1$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|
| I | 128 | 128 | 128 | 128 |
| II | 64 | 128 | 128 | 128 |
| III | 64 | 64 | 128 | 128 |
| IV | 64 | 64 | 64 | 128 |
| V | 64 | 64 | 64 | 64 |

For the evaluation method, the similarity was predicted for all the validation data sets described in Section 5.4.1.2, and the performance of each model was compared in the lower right area of the ROC curve (AUC; Area Under the Curve). An ROC curve is a graph in which the vertical axis represents the True Positive Rate (TPR), and the horizontal axis represents the False Positive Rate (FPR), and is often used to judge the usefulness of binary classification.

In addition, Raspberry Pi Foundation's Raspberry Pi 3 Model B was used to predict the verification data, and the time required for similarity prediction was measured for each model. Table 5.3 shows the performance of Raspberry Pi 3 Model B.

Finally, I evaluate the performance of the human classifier proposed in Section 5.3.2. In order to verify the performance in terms of the number of passengers, when the number of passengers in the bus was $3, 5, 7, 9$ people, the method of section 5.3.2 was applied to verify the grouping accuracy rate.

Table 5.3: Spec of Raspberry Pi 3 Model B

| Item | Spec |
|---|---|
| OS | Raspbian 8.0 Jessie |
| CPU | Quad Core 1.2GHz Broadcom BCM2837 64bit |
| RAM | 1GB |
| Power Consumption | 7W (Max: 12.5W) |
| Weight | 45g |

The grouping accuracy rate *acc* is defined as in equation (5.4). Here, $G_{tp}^{(i)}$ represents the number of grouping true positive in group $i$ (passenger $i$), and $G^{(i)}$ represents the number of images classified into group $i$.

$$\text{acc} = \frac{\sum G_{tp}^{(i)}}{\sum G^{(i)}} \tag{5.4}$$

### 5.4.3 Evaluation Results

#### 5.4.3.1 Face Image Similarity Calculation

First, the two feature comparators proposed in Section 5.3.1.2 are evaluated. As shown in Fig. 5.4, AUC was evaluated by learning a deep model in which the outputs of the 50 types of feature extraction models described in Section 5.4.2 were input to two comparators. Figure 5.8 shows the AUC distribution for each comparator. The average AUC of each model using cosine similarity and DNN is 0.758 and 0.805, indicating that the accuracy of the model using DNN is overwhelmingly good. The reason for the poor accuracy of cosine similarity is that the dimensionality of the feature vector has increased, and the curse of the dimension has occurred, making it difficult to separate the similarity. As a solution to this problem, it is conceivable to reduce the number of dimensions of the feature vector. However, reducing the number of dimensions makes it difficult to represent various features of the facial image, so cosine similarity is not suitable for this model. Therefore, DNN is used as a feature comparator in the subsequent performance evaluation.

The performance of each model is evaluated. Fig. 5.9 shows the ROC curve for each model parameter $L$, and Fig. 5.10 shows the AUC distribution for each parameter $L$. The legend in Fig. 5.9 shows the "model type-channel type-dropout rate." Based on this result, the AUC of model type D is stable at $0.82 - 0.83$ in all channel types. Therefore, model type D is considered suitable for facial image feature extraction. In the best model (D-I/dropout: 0.10), TPR = 0.820, FPR = 0.238 are achieved. The accuracy of about 80% is achieved for both positive and negative cases, and this model is fully operational.

Next, I evaluate the relationship between the time required to extract facial image features and AUC. Figure 5.11 shows a plot of the average processing time and AUC per face image for each model.

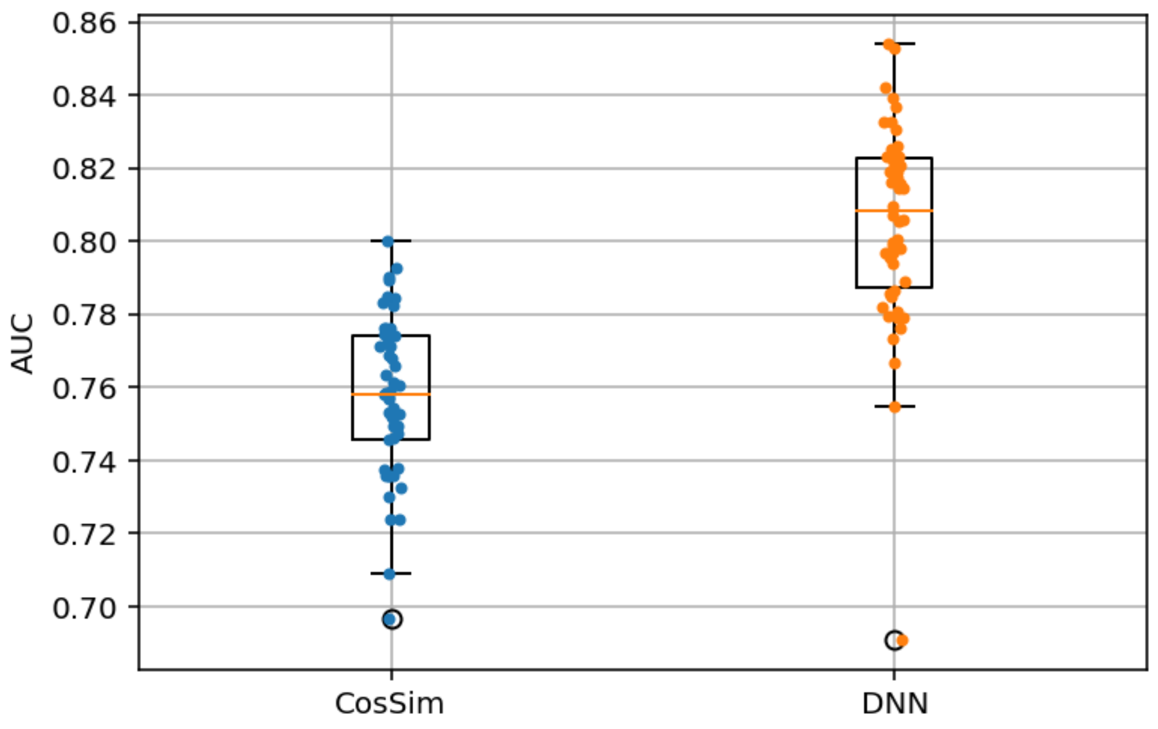
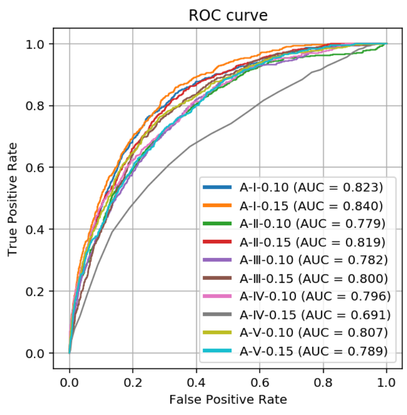


Figure 5.8: Distribution of AUC for each feature comparator

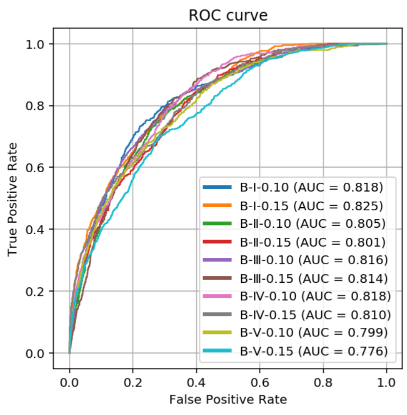
The marker color represents the model type and the shape represents the channel type. The average processing time of D-I, which had the highest accuracy, was 0.70s, and good results were obtained. The models A-I and A-Ⅱ, which have the most CNN layers, have a good AUC, but the processing time is very long.

#### 5.4.3.2 Person Identifier

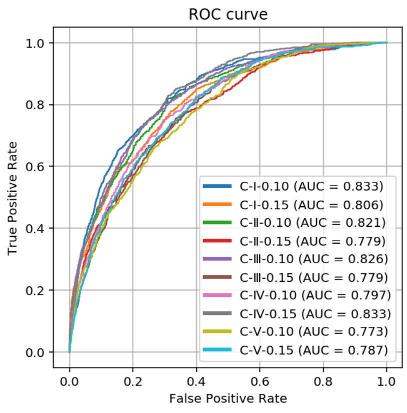
I evaluated the performance of facial image grouping using D-I, which had the best accuracy in Section 5.4.3.1. When there were $3, 5, 7, 9$ passengers in the route bus, I applied the method in Section 5.3.2 and evaluated the accuracy of grouping. Note that the weight parameter $\alpha$ of the face similarity and the position similarity is in the range of $0.0 - 1.0$ in increments of 0.2. Figure 5.12 shows the evaluation results for each number of passengers in the car, with the horizontal axis representing the parameters $\alpha$ and the vertical axis representing the accuracy rate. The parameter $\alpha = 1.0$ was the result of person identification using only the face image comparison model, but the correct answer rate was $60 - 80\%$. However, the accuracy rate improves as the weight of position similarity increases, and the accuracy rate of about 80-100% is achieved when $\alpha = 0.2, 0.4$. From this, accuracy can be improved by adding position information in the frame to the face image comparison model, and it can be said that sufficiently practical accuracy was obtained even when a lightweight face image comparison model was used.



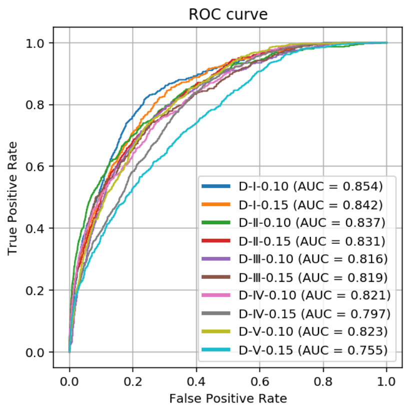
(a) Model Type A



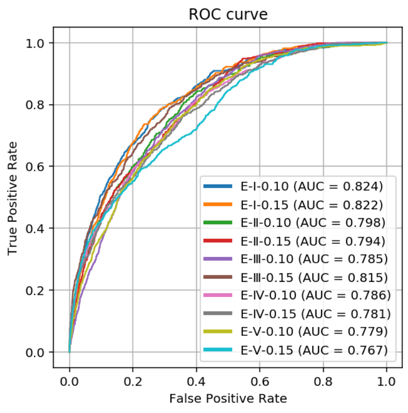
(b) Model Type B



(c) Model Type C



(d) Model Type D



(e) Model Type E

Figure 5.9: ROC curve for each model type

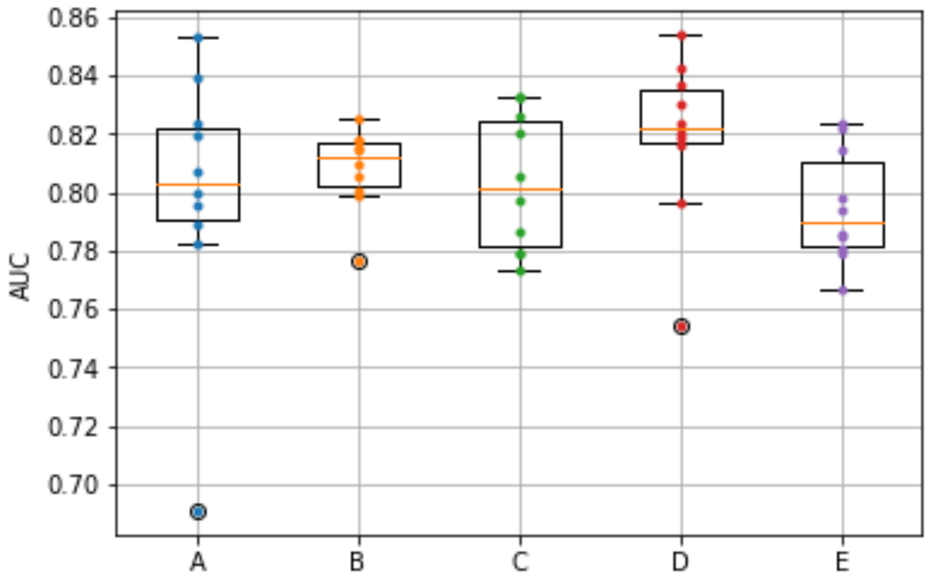


Figure 5.10: AUC for each model type

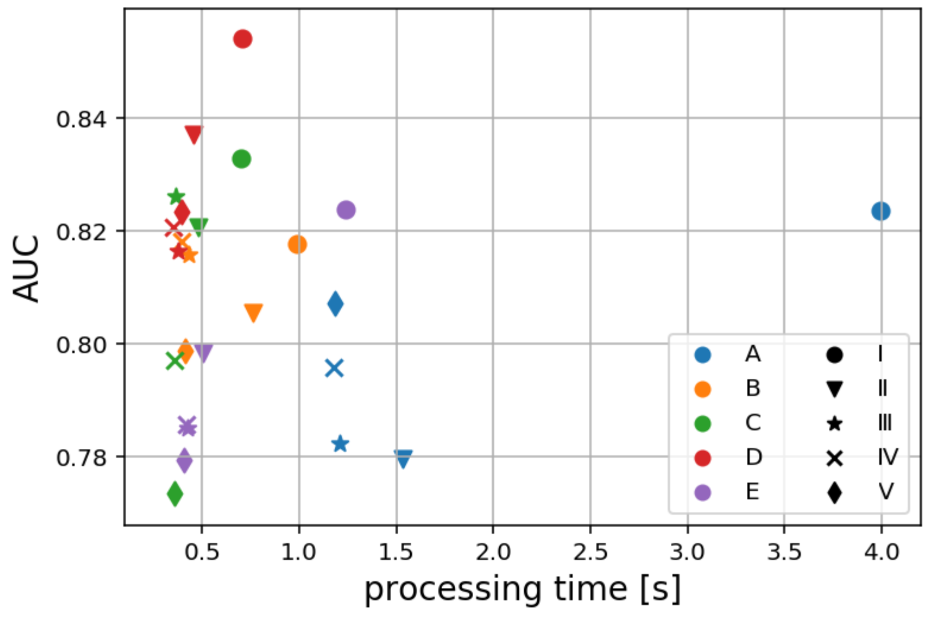


Figure 5.11: Relationship between processing time and AUC

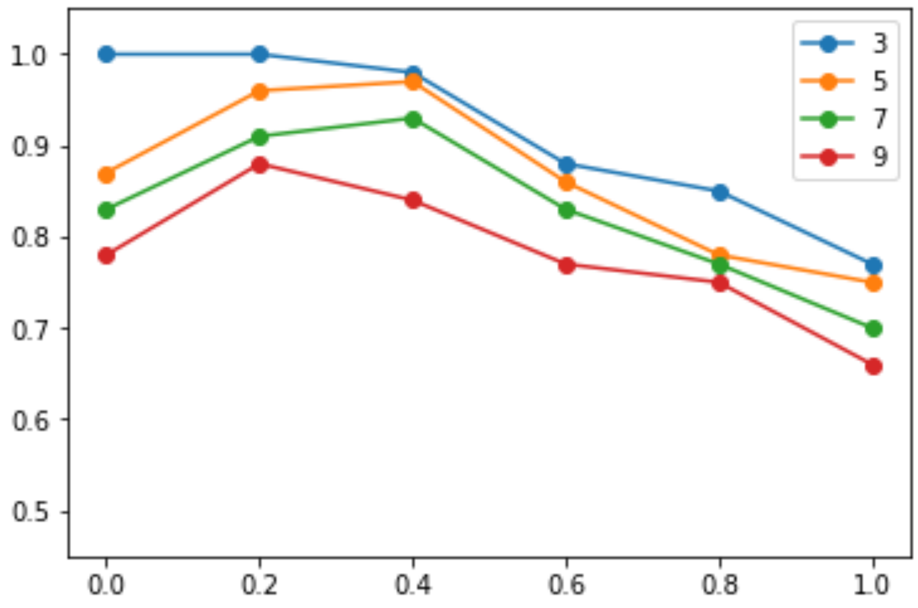


Figure 5.12: Face image grouping accuracy by number of passengers

## 5.5 Conclusion

In this chapter, I proposed a method for estimating the similarity of facial images, which is important in creating an OD measurement system for route buses using camera images. A face image feature extraction model based on CNN was established to reduce the computational complexity of the deep learning model so that an inexpensive single-board computer can perform real-time processing. As a result of performance evaluation in an environment that reproduced an actual route bus, the best model achieved AUC 0.854 and processing time per person 0.70s. This can be said to be accuracy and processing time that can be sufficiently realized in actual operation.

In this system, the face image can be discarded on the in-vehicle terminal at the point where the passenger gets off, and the face image is not stored in the permanent memory as data. For this reason, although there is little concern about privacy infringement, face images are included in personal data (personal information), so operators are required to be cautious in actual operation. Specifically, it is necessary to indicate that it is for a trip survey and that personal data is immediately anonymized and that it is not recorded, including the data after anonymization, and also the opt-out method. I think that it is necessary to ask passengers to understand local transportation improvement in cooperation with local governments.

As a future work, this paper proposed a similarity estimation method using a face image as a single input. However, in addition to the face image, the face image, such as the eyes and nose, can be separately inputted to improve accuracy and reduce the model.

# Chapter 6

# Conclusion

This dissertation has presented a method to grasp trips of people who move automatically between cities without having to install a dedicated app by users. This dissertation presents a method to grasp trips of people who move automatically between cities without having to install a dedicated app by users. The goal of this dissertation is to use various sensors to grasp the destinations of multimodal people such as trains and buses. Specifically, I address this issue by sensing people using laser range sensors, RGB cameras, and CSR logs that are recorded when a smartphone communicates with a base station. In this dissertation, I have made the following three primary contributions to embody this idea.

Firstly, I have proposed two methods for estimating travel from CSRs for two different scenarios according to availability of cooperative users. The first method estimates travel modes and trajectories from CSRs, focusing on travels by trains and automobiles that are majorities of travel modes. To overcome coarse grained location information of CSRs, I fully exploit spatio-temporal features such as average speeds, train timetables, station locations and geographic information of railway networks and road networks. However, its performance is degraded in the real urban environment due to complex radio wave propagation caused by many buildings. To overcome this problem, I propose another travel estimation method using CSRs combined with GPS traces provided by cooperative users. Complex radio propagation is directly learned from CSRs and GPS traces of the cooperative users. In this method, I focus on train passengers because the number of the travel route candidates is enormous for other travel modes, which requires a huge amount of training data. Nevertheless, I cannot expect the enough volume of the training data to generate a radio propagation map. Therefore, it uses the communication interval between connection with two different BSs during a travel as a related feature. In evaluation, I have modeled the complex BS selection pattern as a radio propagation model to reproduce CSRs and integrated the model into Scenargie network simulator [7]. I conduct a simulation experiment reproducing an area around Shinjuku station. The results show that train passengers are identified with 53% recall and 87% precision and automobile passengers are identified with 76% recall and 78% precision. It also achieves 67% accuracy for automobile trajectory estimation. The simulation results also show that the method with GPS traces successfully identifies train passengers with 90% recall and 85% precision for data sets with the expected connection interval of 30 seconds.

Secondly, I have proposed a method for estimating the number of passengers on a route bus. This method uses a laser range sensor (LiDAR) and Raspberry Pi to reduce the labor and cost barriers for introduction and realize high-precision detection. In the proposed method, sensor correction can be performed simply by installing the system at an appropriate location and acquiring background data when the door is open and when the door is closed. It uses the background subtraction method to automatically detect the opening and closing of the route bus door. After detecting the opening of the door, the point cloud corresponding to the object surface is extracted using the background subtraction method, and the point cloud corresponding to the human body surface is detected from the point cloud. Based on this, the number of passengers is measured by tracking the point cloud of each passenger and detecting the passage of doors. Because it automatically detects the opening and closing of the doors, it can simultaneously measure the time and number of passengers required at each stop, and can also determine the arrival time and stop time of each stop, so it can also be used for route bus operation planning be able to. In evaluation, I conducted experiments with a bus running between two campuses of my university and achieved 5.3% or lower error (the average absolute error was 0.94%) for coming passengers and 7.5% (the average was 1.9%) for leaving ones. The average processing time per frame was 3.4 ms in Raspberry Pi 3 Model B. This has shown the capability of my algorithm for real-time measurement.

Thirdly, I have proposed a method for estimating the OD of bus passengers. This is a method that uses an RGB camera to estimate the OD of bus passengers, which could not be achieved by using a LiDAR. Specifically, by recognizing passengers in the car during operation between each stop, and comparing passengers in the car before and after the stop, passengers in the car that got off at that stop and new passengers who got on that stop to understand. This makes it possible to measure the OD of each passenger. In order to evaluate the effectiveness of the proposed method, the performance was evaluated using camera images taken in an environment that reproduced an actual route bus. As a result of creating 50 kinds of learning models and evaluating the performance using Raspberry Pi 3, the best model achieved AUC 0.854, processing time 0.70s, and accuracy comparable to conventional face image identification system It was shown that it can be processed at high speed.

In conclusion, this dissertation has established technologies that can be applied in the cases shown in Table 6.1. In major cities, it is possible to estimate trips for people of all modes of transportation using the method using the CSRs described in Chapter 3. In local cities, I focused on local buses, which are the mainstay of local city transport. The trip estimation of the route bus became possible by the method using the laser range sensor and the RGB camera shown in Chapters 4 and 5. This dissertation has contributed to urban development planning and evacuation guidance in the event of a disaster by establishing a real-time trip estimation technology that does not require the user to install a dedicated application.

However, as can be seen from Table 6.1, my technologies cannot measure trains, cars, and people walking on foot in local cities. Future work is to establish a method to measure these people. Specifically, cameras can be installed at ticket gates at railway stations and intersections on roads to track them. However, the viewpoint of user privacy and the size of the installation scale (particularly at intersections) are issues. It can be said that how to solve this problem is important.

Table 6.1: Coverages of my technologies

| | Major Cities | Local Cities |
|---|---|---|
| Train | Chapter 3 | x |
| Bus | Chapter 3 | Chapter 4,5 |
| Car/Taxi | Chapter 3 | x |
| Walk | Chapter 3 | x |

# Acknowledgement

First of all, I would like to gratefully acknowledge the enthusiastic supervision and great support of Professor Teruo Higashino during this research. I also express my appreciation for his great encouragement, support, and backup in my Ph.D student life.

Secondly, I am very grateful to Professor Masayuki Murata, Professor Takashi Watanabe, Professor Toru Hasegawa and Professor Morito Matsuoka of Osaka University for their invaluable comments and helpful suggestions concerning this thesis.

Thirdly, I much thank Associate Professor Hirozumi Yamaguchi for his precious advice and technical discussions provided throughout the research.

In addition, I would like to be thankful very much to Associate Professor Akihito Hiromori for giving me helpful advice. And I would like to thank Associate Professor Takaaki Umedu at Shiga University for many precious pieces of advice and support. Also, I am grateful very much to Assistant Professor Akira Uchiyama, Specially Appointed Assistant Professor Teruhiro Mizumoto and Specially Appointed Assistant Professor Viktor Erdélyi of Osaka University for many impressive comments.

Special thanks to Hankyu Bus Co., Ltd., Zentan Bus Co., Ltd., and Toyooka City. They provided me with a demonstration experiment field.

Finally, I would like to thank every one of Higashino laboratories for their feedback, encouragement, and kind support in my research. To my fiance Megumi, thank you for your assistance, your encouragement, and your love.

# Bibliography

[1] "OpenStreetMap," http://www.openstreetmap.org/ (ref. 2017-01-20).

[2] L. Hokuyo Automatic Co., "Scanning rangefinder distance data output/urg-04lx-ug01," https://www.hokuyo-aut.jp/search/single.php?serial=166 (ref. 2017-06-22).

[3] *Results from the 4th Nationwide Person Trip Survey*, Ministry of Land, Infrastructure and Transport of Japan, 2007, http://www.mlit.go.jp/crd/tosiko/zpt/pdf/zenkokupt_gaiyouban_english.pdf.

[4] T. Horanont, A. Witayangkurn, Y. Sekimoto, and R. Shibasaki, "Large-Scale Auto-GPS analysis for discerning behavior change during crisis," *IEEE Intelligent Systems*, vol. 28, no. 4, pp. 26–34, 2013.

[5] D. Zhang, M. Chen, M. Guizani, and H. Xiong, "Mobility prediction in telecom cloud using mobile calls," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 26–32, 2014.

[6] H. Kanasugi, Y. Sekimoto, M. Kurokawa, T. Watanabe, S. Muramatsu, and R. Shibasaki, "Spatiotemporal route estimation consistent with human mobility using cellular network data," in *Proceedings of 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2013, pp. 267–272.

[7] Space-Time Engineering, LLC, *Scenargie*, http://www.spacetime-eng.com/.

[8] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and gis information," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 2011, pp. 54–63.

[9] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on gps data for web applications," *ACM Trans. Web*, vol. 4, no. 1, pp. 1:1–1:36, 2010.

[10] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in *Proceedings of the 17th International Conference on World Wide Web.* ACM, 2008, pp. 247–256.

[11] G. M. Djuknic and R. E. Richton, "Geolocation and assisted gps," *Computer*, vol. 34, no. 2, pp. 123–125, 2001.

[12] R. Koshirai, *Easy GPS survey (in Japanese).* Ohmsha, Ltd., 2010.

[13] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, *et al.*, "Place Lab: Device positioning using radio beacons in the wild," in *Proceedings of IEEE Pervasive Computing*, 2005, pp. 116–133.

[14] B. Krach and P. Robertson, "Integration of foot-mounted inertial sensors into a bayesian location estimation framework," *In Proceeding of 5th Workshop on Positioning, Navigation and Communication, (WPNC '08)*, pp. 55–61, 2008.

[15] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proceedings of the 10th International Conference on Ubiquitous computing (UbiComp '08)*, 2008, pp. 114–123.

[16] U. Steinhoff and B. Schiele, "Dead reckoning from the pocket an experimental study," in *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom '10)*, 2010, pp. 162–170.

[17] A. Agarwal and S. R. Das, "Dead reckoning in mobile ad hoc networks," in *Proceedings of 2003 IEEE Wireless Communications and Networking (WCNC '03)*, vol. 3, 2003, pp. 1838–1843.

[18] J. Zhu and G. D. Durgin, "Indoor/outdoor location of cellular handsets based on received signal strength," *Electronics letters*, vol. 41, no. 1, pp. 24–26, 2005.

[19] A. J. Weiss, "On the accuracy of a cellular location system based on RSS measurements," *IEEE Transactions on Vehicular Technology*, vol. 52, no. 6, pp. 1508–1518, 2003.

[20] H. L. Van Trees, *Detection, estimation, and modulation theory.* John Wiley & Sons, 2004.

[21] T. Toledo and T. Kolechkina, "Estimation of dynamic origin - destination matrices using linear assignment matrix approximations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 618–626, 2013.

[22] C. de Fabritiis, R. Ragona, and G. Valenti, "Traffic estimation and prediction based on real time floating car data," in *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems (ITSC 2008)*, 2008, pp. 197–203.

[23] B. R. Hellinga, "Estimating dynamic origin-destination demands from link and probe counts," Ph.D. dissertation, Queen's University, 1994.

[24] R. Ganti, M. Srivatsa, A. Ranganathan, and J. Han, "Inferring human mobility patterns from taxicab location traces," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013, pp. 459–468.

[25] F. Calabrese, F. C. Pereira, G. Di Lorenzo, L. Liu, and C. Ratti, “The geography of taste: analyzing cell-phone mobility and social events,” in *Proceedings of IEEE Pervasive Computing*, 2010, pp. 22–37.

[26] J. McInerney, J. Zheng, A. Rogers, and N. R. Jennings, “Modelling heterogeneous location habits in human populations for location prediction under data sparsity,” in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing.* ACM, 2013, pp. 469–478.

[27] Q. Lv, Y. Qiao, N. Ansari, J. Liu, and J. Yang, “Big data driven hidden markov model based individual mobility prediction at points of interest,” *IEEE Transactions on Vehicular Technology*, 2016.

[28] C. A. Davis Jr, G. L. Pappa, D. R. R. de Oliveira, and F. de L. Arcanjo, “Inferring the location of twitter messages based on user relationships,” *Transactions in GIS*, vol. 15, no. 6, pp. 735–751, 2011.

[29] T. Kanno, H. Kanasugi, Y. Sekimoto, and R. Shibasaki, “Real-time passenger location estimation using cdrs and train objects generated from crowdsourced timetables,” in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, 2015, pp. 1197–1205.

[30] H. Ishizuka, N. Kobayashi, S. Muramatsu, and C. Ono, “Detecting train commuters using cdrs and gis information,” in *NetMob 2015 book of abstracts::posters*, 2015, pp. 39–41.

[31] U. Scheunert, H. Cramer, B. Fardi, and G. Wanielik, “Multi sensor based tracking of pedestrians: a survey of suitable movement models,” in *IEEE Intelligent Vehicles Symposium, 2004*, June 2004, pp. 774–778.

[32] R. Greene-Roesel, M. C. Diogenes, D. R. Ragland, and L. A. Lindau, “Effectiveness of a commercially available automated pedestrian counting device in urban environments: Comparison with manual counts,” *Safe Transportation Research & Education Center*, 2008.

[33] H. Yang, K. Ozbay, and B. Bartin, “Investigating the performance of automatic counting sensors for pedestrian traffic data collection,” in *Proceedings of the 12th World Conference on Transport Research, Lisbon, Portugal*, vol. 1115, 2010.

[34] H.-b. Qian and H. Han, “The applications and methods of pedestrian automated detection,” in *Proceedings of Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on*, vol. 3. IEEE, 2010, pp. 806–809.

[35] D. Bauer, M. Ray, and S. Seer, “Simple sensors used for measuring service times and counting pedestrians: Strengths and weaknesses,” *Transportation Research Record: Journal of the Transportation Research Board*, no. 2214, pp. 77–84, 2011.

[36] T. Hata, “Detecting numbers and directions of pedestrians walking through a one-person gate with a pyroelectric infrared sensor,” *Japan Society for Fuzzy Theory and Intelligent Informatics*, pp. 887–898, 2016.

[37] Fanbright, “Moving counter,” http://www.fanbright.jp/service/movingcounter/ (ref. 2017-06-22).

[38] L. D. Pizzo, P. Foggia, A. Greco, G. Percannella, and M. Vento, “A versatile and effective method for counting people on either rgb or depth overhead cameras,” in *Proceedings of 2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2015, pp. 1–6.

[39] S. D. Pore and B. F. Momin, “Bidirectional people counting system in video surveillance,” in *Proceedings of 2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2016, pp. 724–727.

[40] L. Giken Trastem Co., “Passenger counter,” http://www.trastem.co.jp/product/passenger_counter.html (ref. 2017-06-22).

[41] Eurotech, “Pcn-1001,” https://www.eurotech.com/en/products/PCN-1001 (ref. 2017-06-22).

[42] R. Ranjan, C. D. Castillo, and R. Chellappa, “L2-constrained softmax loss for discriminative face verification,” *Computing Research Repository (CoRR)*, vol. abs/1703.09507, 2017.

[43] J. Zhao, L. Xiong, P. Karlekar Jayashree, J. Li, F. Zhao, Z. Wang, P. Sugiri Pranata, P. Shengmei Shen, S. Yan, and J. Feng, “Dual-agent gans for photorealistic and identity preserving profile face synthesis,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 66–76.

[44] L. Xiong, J. Karlekar, J. Zhao, J. Feng, S. Pranata, and S. Shen, “A good practice towards top performance of face recognition: Transferred deep feature fusion,” *Computing Research Repository (CoRR)*, vol. abs/1704.00438, 2017.

[45] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “Vggface2: A dataset for recognising faces across pose and age,” in *Proceedings of International Conference on Automatic Face and Gesture Recognition*, 2018.

[46] J. Yang, P. Ren, D. Chen, F. Wen, H. Li, and G. Hua, “Neural aggregation network for video face recognition,” *Computing Research Repository (CoRR)*, vol. abs/1603.05474, 2016.

[47] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, "An all-in-one convolutional neural network for face analysis," *Computing Research Repository (CoRR)*, vol. abs/1611.00851, 2016.

[48] N. Crosswhite, J. Byrne, O. M. Parkhi, C. Stauffer, Q. Cao, and A. Zisserman, "Template adaptation for face verification and identification," *Computing Research Repository (CoRR)*, vol. abs/1603.03958, 2016.

[49] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa, "Triplet probabilistic embedding for face verification and clustering," *Computing Research Repository (CoRR)*, vol. abs/1604.05417, 2016.

[50] T. Sinclair and D. Ghosal, "An enhanced network architecture to support replicated hlr databases-prototype design and experimental performance analysis," in *Proceedings of 1999 IEEE International Conference on ICC'99*, vol. 2, 1999, pp. 1367–1373.

[51] M. A. Bayir, M. Demirbas, and N. Eagle, "Mobility profiler: A framework for discovering mobility profiles of cell phone users," *Pervasive and Mobile Computing*, vol. 6, no. 4, pp. 435 – 454, 2010.

[52] A. Goldsmith, *Wireless Communications.* Cambridge University Press, 2005.

[53] N. Nakajima, T. Arita, and K. Higuchi, *Why do mobile phones connect (in Japanese).* Nikkei BP, 2012.

[54] ktailab.net, "Ktai-lab," http://ktailab.net/ (ref. 2017-01-20).

[55] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.

[56] T. Ministry of Land, Infrastructure and Tourism, "About the results of the smoothing of car traffic-related movement at the end of 2015 - changes in the number of vehicles such as non-step buses (reference 1)," p. 1, 2016.

[57] Y. Yamada, A. Hiromori, H. Yamaguchi, and T. Higashino, "Development of bus passenger counter using lidar sensors," vol. 60, no. 3, pp. 934–944, 2019.

[58] Thanh Nguyen, "YOLOFace," https://github.com/sthanhng/yoloface (ref. 2019-08-20).

[59] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767