修 士 論 文

Augmented Reality based Pedestrian Navigation using Smart Glasses and Open Map Source

（スマートグラスとオープンマップソースを活用した拡張現実歩行者ナビゲーションの研究）

指導教員　　上條俊介 准教授

東京大学大学院
情報理工学系研究科
電子情報学専攻

学籍番号・氏名 48-166467　王 涯

Abstract

Pedestrian navigation has become one of the most important services in people’s city lives. The navigation application can be considered as the suggestion system for pedestrian travelling behavior. With the development of intelligent wearable devices, such as smart glasses, more and more scientists and engineers focus their attention on wearable devices based navigation rather than smart phone-based navigation. An accurate positioning technology is fundamental to a satisfied navigation service. Even though various sensors, such as Global Navigation Satellite System receiver, gyroscope, accelerometer and magnetometer sensors, have been integrated into the current smart phones, the performance of positioning in city urban is still not satisfied because of GNSS signals reflections, the high dynamic of pedestrian activities and disturbance of the magnetic field in city environments. On the other hand, visualizing navigation information in first person view can bring much more convenience for pedestrians, in other words, AR(Augmented Reality) based navigation enables pedestrians perceive virtual and real objects as coexisting in the same space.

This work not only proposes to improve the accuracy of the positioning using the camera sensor in smart glasses, but also gives implementation for intersection detection and destination recognition using open map source in commercial urban area. The visual observation for surround environment provided by camera sensor is compared with the available Google Maps Street View to correct positioning errors. By matching the geo-tagged pedestrian’s photo with the reference images from Google Maps Street View, we cannot only recognize which side of the road the pedestrian is in but also achieve 4-meter positioning error. Furthermore, the 2D shape of intersection and footprint of destination provided by open street map can be used to segment the local 3D point cloud which is generated by combining panorama and depth map from Google Maps API. The camera pose and pedestrian’s position suggest to visualize guidance arrow and destination in first person view.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

In this work, we researched on improving the positioning accuracy for pedestrian wearing smart glasses in urban area with the aid of Google Maps Street View and presenting a augmented reality based pedestrian navigation system. This chapter introduces the reader to the background, objective and structure of this thesis.

1.1. Background

In recent years, Pedestrian navigation has become one of the most important services in people’s city lives. The navigation application can be considered as the suggestion system for pedestrian travelling behavior. The quality of navigation service highly depends on the accuracy of pedestrian localization.

Nowadays, smart phones are nowadays far more than merely devices to communicate with and they are products that help to make our work and everyday life easier. However, with the development of intelligent wearable devices, such as smart glasses, more and more scientists and engineers focus their attention on wearable devices based navigation rather than smart phone-based navigation. Among them, smart glasses integrated various sensors--Global Positioning System receiver, wireless receiver, gyroscope, accelerometer, and magnetometer--should be the most useful device for pedestrian navigation. These sensors just meet the requirement of state-of-the-art positioning techniques, which are Global Positioning System positioning, Wi-Fi-based positioning, PDR-based positioning, and vision-based positioning.

Global Positioning System (GPS) is the most developed positioning system in the world and it is widely applied in the smart devices such as smart phone and smart glasses. GPS is capable of providing reliable position information in most cases, which is extremely helpful in the researches and daily applications. Usually, at least 4 satellites are needed to calculate the positioning result in a GPS system and the positioning accuracy will be degrade when the number of received satellite is small or the received signals contained errors. In the open sky condition, GPS system can perform high

accuracy with a standard deviation about 0.3 meter. However, in the urban city, GPS signals are usually influenced by the blockage and the reflection of high buildings as shown in Figure 1.1 [2]. In these kinds of situations, errors called non-line-of-sight (NLOS) and multipath situations will happened [1].

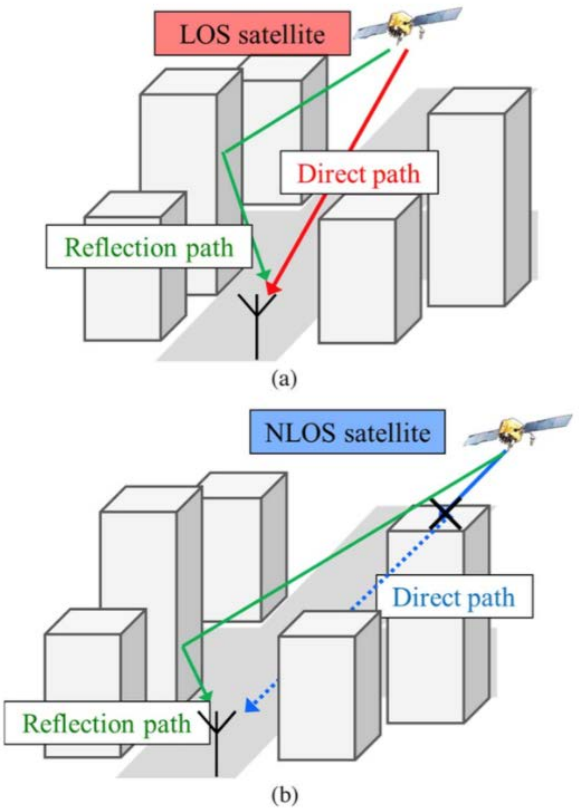


Figure 1.1 The multipath and NLOS effects in an urban canyon. (a) Multipath effect. (b) NLOS propagation.

As a result, in urban canyon environment, the positioning accurate of standalone GPS system is larger than 10 meters. The huge error of GPS based positioning will lead to a mistake in distinguishing the correct side of road and intersection. Even though nowadays we can use Russian GLONESS or Japanese QZSS, China Beidou(etc.) systems to increase the satellite numbers as a whole system of Global Navigation Satellite System(GNSS), it is still suffered the same problem of NLOS and multipath effect as standalone GPS system.

Consider the sensors embedded in the smartphones and smart glasses, there are some other commercialized positioning technologies can be used, including the Wi-Fi-based positioning system and Pedestrian Dead Reckoning (PDR) [3, 5]. Wi-Fi-based

positioning system is a newly attractive method to provide navigation services. Nowadays, Wireless Local Area Network (WLAN) can be found in almost every building. The wide spread infrastructure offers the possibility to using Wi-Fi-based positioning even in outdoor environment at urban canyon. The WPS has advantage over GPS positioning system for its ability to provide seamless navigation service in both indoor and outdoor environment. The Wi-Fi-based positioning is first developed for indoor environment and has been well developed. With the rapid increase in Wi-Fi access points (APs) in metropolitan areas, some researchers and companies implemented Wi-Fi-based positioning in outdoor environment. However, the accuracy of Wi-Fi-based positioning in outdoor environment is still not as accurate as WPS in indoor environment

As for PDR, the system uses the inertial measurement unit (IMU) sensors in the wearable devices, such as accelerometer, magnetometer and gyroscope, and use the circle change of the data from these sensors when the user is walking, to detect every step of the pedestrian and calculated step length and moving direction of the pedestrian. PDR can produce continues and smooth positioning trajectory. However, PDR can only provide relative positioning result but not the absolute positioning result. Therefore, PDR need some other positioning systems to provide an initial position. What's more, PDR will suffer from error accumulation in long distance.

Google Maps Street View is a comprehensive and large database provided by Google, which consists of geo-tagged 360º panoramic images of almost all main streets and roads in a number of countries. The panoramic images of Google Maps Street View are recorded by a spherical arrangement of cameras and the localization comes from the high-performance localization system. Because of its characteristics and quality, Google Maps Street View attracts more and more attentions in the field of computer vision and localization. Beside the panoramic images, we can also get the corresponding depth data from the API which is provided by Google. By combining the panoramic images and depth map information, local 3D models can be created. By using this 3D model, virtual views in all the places can be generated and there will be no limitations from the fixed interval position of the panoramic images.

Considering these limiting problems of GPS and Wi-Fi, with the help of image streams provided by smart glasses, we proposes a pedestrian positioning system with the technique of visual localization, using the matching between the geo-tagged photo

from the camera on the wearable devices and the generated 3D local model from Google Maps Street View.

To get an accurate positioning result, firstly we employ the vanishing point detection to obtain the heading angle of pedestrians, then searching a rough position using a convergent algorithm which keeps updating position according to translation relationship between the query image and reference images for candidate positions. Finally, we generate a local 3D model and find the most matched view to be taken as positioning results.

Once we get an accurate positioning result, we focus on presenting a intuitive navigation system for smart glasses users. Augmented reality (AR) is a live view of a physical world but elements are overlaid by desired geospatial information. There are many applications such as education, art, game, and also navigation. The key to the AR system is how to combine the expanded object with the actual environment. For camera position and heading it is necessary to convert scenes from actual coordinates to camera coordinates and overlay the enlarged objects on the scene. Most of augmented reality researches focus on how to apply the position and heading information of the camera from various sensors integrated on the device. But, as mentioned above, the performance of these devices in urban areas is not accurate. Therefore, in order to provide realistic AR navigation, it is necessary to provide a vanishing point for determining the angle of the road in the pedestrian's field of view. Augmented Reality application, on the other hand, relied on predefined and well-modeled content that only applies to real world conditions that existed during application development. Therefore, in many cases, flexibility is limited because AR applications can only be placed in one physical location. This problem gets worse as environmental changes cause inconsistencies between previously recorded data and the real world. Such a situation may damage the correct function of the application. To solve this problem, we use well-updated and easy-access open map source to present navigation guidance rather than build a database for navigation.

1.2. Objective

The goal of this thesis is to implement a navigation system for pedestrian wearing smart glasses, which supports people in commercial urban area. We can achieve 4-meter positioning accuracy and can also distinguish which side of the road the pedestrian is in. When pedestrian walking towards an intersection, an overlaid arrow guidance will

be shown in the first person view. When the destination is near, it can also be recognized and segmented in first person view. Our work mainly consists of two parts.

1. First, we propose to improve positioning accuracy by using camera sensor. The camera sensor visually checks the surround environment. This observation is compared with the available street view from the Google Maps Street View and the virtual view from the 3D local model and reduces the positioning error. As a results, we can improve the correct side rate to 90% and achieve positioning performance of 4 meters.
2. As for the AR navigation part, we focus on intersection detection and destination recognition. We use intersection shapes and building footprint both from 2D open map to segment local 3D point cloud which is generated by combining panorama and depth map from Google Maps. And by coordinates convection, we can draw arrow guidance and segment destination in first person view. Figure 1.3 shows the sketch map of the AR navigation interface.

Figure 1.2 The sketch map of the AR navigation system interface

1.3. Thesis Structure

This thesis describes the entire process of developing a AR navigation system for pedestrian navigation purposes. It is divided into five parts.

After this introduction chapter, the basic concepts of GPS positioning, PDR-based positioning, Wi-Fi-based positioning and the integrating system are explained in

Chapter 2.

In Chapter 3, firstly we introduce the related works about the vision-based positioning and the Google Maps Street View is also introduced as it is an important source. Then we describe our approach to obtain accurate pedestrian position with the aid of Google Maps Street View in the urban area.

In Chapter 4, at first the related works about intersection detection and destination recognition in first person view are introduced. Then we describe how to use open source to give reliable AR arrow guide in intersections and how to show destinations in pedestrian navigation system.

Finally, Chapter 5 summarizes this work. It gives an overall conclusion, shows the problems that came up and the limitations of the resulting system. In the end of this chapter future tasks to enhance this system are listed.

Chapter 2.

Pedestrian Positioning in Urban Canyon

In this chapter, we will introduce some basic pedestrian positioning methods, including the Global Positioning System (GPS) positioning, Pedestrian Dead Reckoning (PDR) based positioning, Wi-Fi based positioning and their related works. The fundamental of integrated positioning system will also be introduced in this chapter.

2.1. GPS Positioning and Problem

GPS is originally designed to provide position, speed and time information since it was initially developed by America in 1973. Nowadays there has been totally 31 GPS satellites launched into the space and they almost covers the whole planet as shown in Figure 2.1 [4].

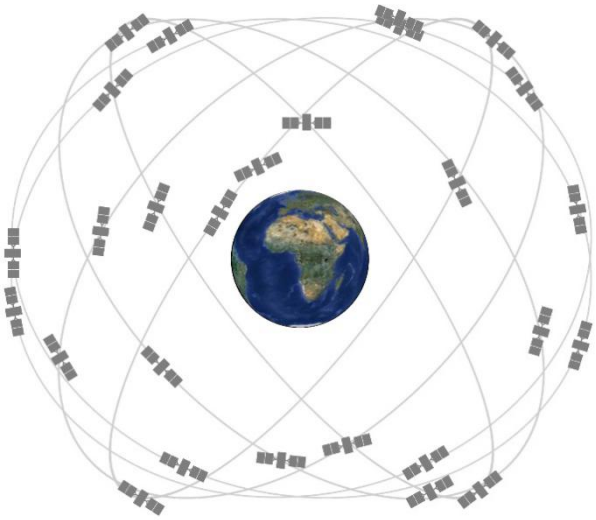


Figure 2.1 The constellation of GPS satellites

Thanks to the high reliability and global coverage, GPS has been opened to public and now GPS is the mainly source for position information worldwide and basically every airplane, vehicle and mobile device has a GPS receiver built inside to provide

customers with position information. Also various kinds of researches have been done in this field

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

2.1.1. Problems of GPS positioning

When computing the pseudorange and GPS positioning, many kinds of errors will influence the result. At first, the satellites will cause the errors of satellite clock bias and satellite coordinate bias. These errors has pattern and could be pre-estimate. The ephemeris information sent by the satellites contains three clock corrections and they can be used to compute the accurate time based on GPS time. Around 5 meters error may be caused from the clock bias [6]. Nowadays, with the development of the technology, satellites has been equipped with new atomic clock. It can be expected that satellite clock bias could be minimized in the future.

Then, when the signal from the satellite is penetrating the earth atmosphere, it has to pass through the ionosphere and troposphere as shown in figure 2.3. The propagation speed of the signal will slow down in the ionosphere and troposphere because of the sun activity, geomagnetic activity and other atmosphere activities. Temperature, humidity, seasons and day night alternation will affect the ionospheric and tropospheric errors so it is difficult to predict the exact time delay. Nevertheless, nowadays, there are Klobuchar model, which could correct 50% of error caused by ionosphere, and Saastamoinen model, which could correct more than 90% of errors caused by troposphere [6].

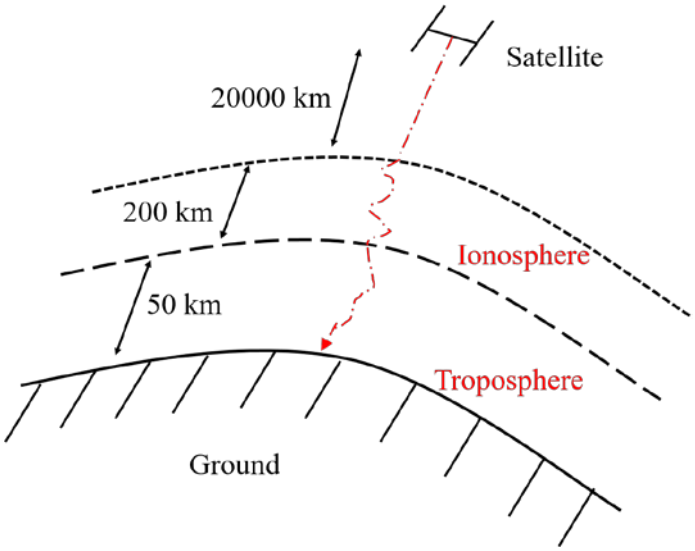


Figure 2.2 The signal propagation through the atmosphere.

Finally, there is also error coming from the receiver's noise. The receiver's noise is caused by antenna, cable, shaking and environmental factors. These errors cannot be modeled but usually they do not affect the positioning result very much. However, as mentioned above, in urban canyon environment, the main error source for GPS is coming from non-line-of-sight (NLOS) and multipath. NLOS and multipath does not have patterns, and cannot be easily corrected.

There are many researches about correcting and evaluating GNSS positioning results have been done. Some of researches are focusing on GPS/GNSS data itself to improve the positioning result. Some others choose to combine other data to improve the accuracy. For example, Miura et al. [2] proposed to use 3D map and ray tracing algorithm to detect whether the signal is blocked or reflected by buildings. The 3D building map can be created based on reliable 2D map and height data of buildings, or by scan the buildings from airplanes or cars, which are equipped with radar or some other sensors. Then, by using the created 3D map, NLOS and multipath signals can be detected from the ray tracing algorithm as shown in Figure2.4 [2].

What’s more, the integration of different positioning system to improve the positioning accuracy is widely used. Then in this chapter, we will introduce two popular positioning method of Pedestrian Dead Reckoning (PDR) based positioning and Wi-Fi based positioning and the fundamental of integrated positioning system.

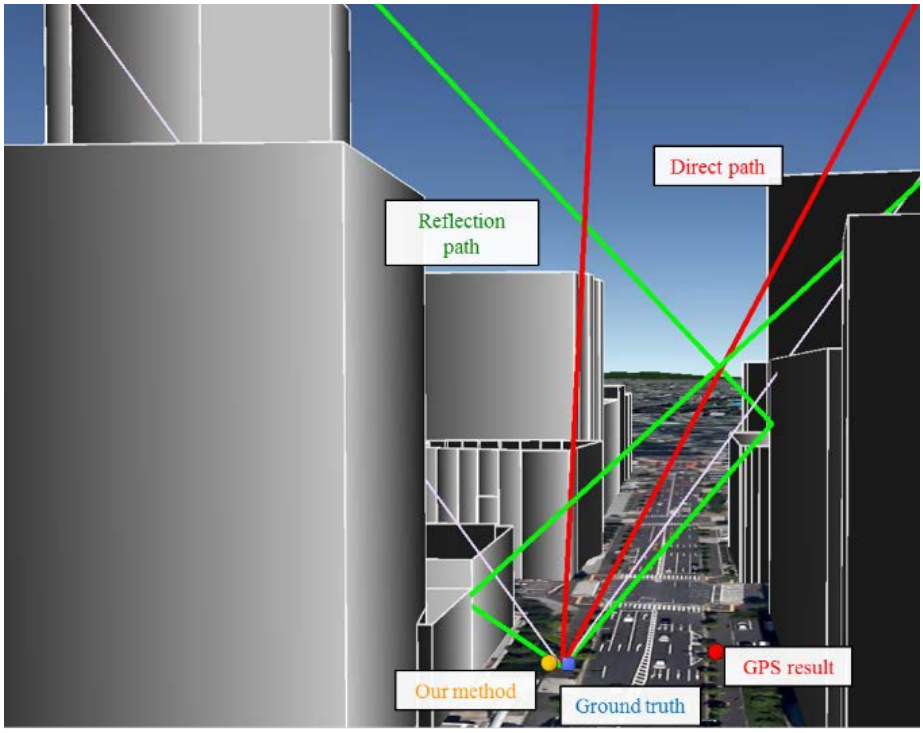


Figure 2.3 Ray-tracing simulation with 3D map.

2.2. PDR-based Positioning

2.2.1. Problems of PDR

As mentioned above, PDR can only provide relative positioning result but not the absolute positioning result. Therefore, PDR need some other positioning systems, such as GPS or Wi-Fi, to provide an initial position. And another big problem of PDR is the error accumulation.

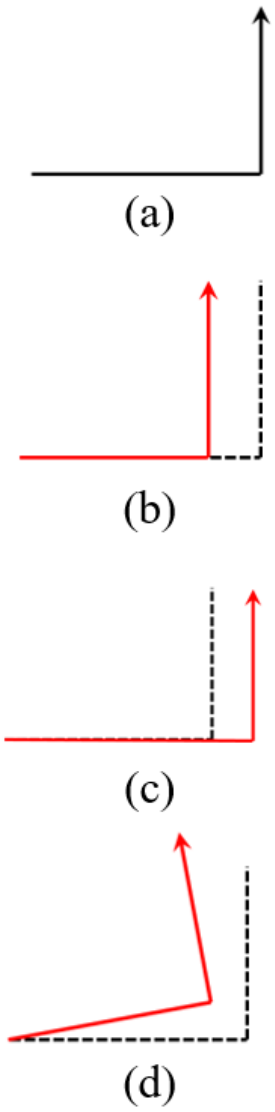


Figure 2.4 Different error accumulation of PDR system.

As shown in Figure 2.8, the error in either the (b, c) stride length estimation or the (d) heading estimation may happen and be accumulated at each step. And this error will also influence the positioning result in the future time. Finally, the PDR trajectory will contain a large deviation.

Therefore, it is difficult to use PDR itself to provide the positioning result and the integration with different positioning system is required.

2.3. Wi-Fi-based Positioning

Nowadays most public buildings already have well-established Wi-Fi

infrastructure, making Wi-Fi becoming an effective aiding resource for indoor navigation. Although some companies have provided Wi-Fi-baesd positioning services in outdoor environment, the accuracy is still not satisfied. Nevertheless, Wi-Fi-based positioning in the urban area is still a research of potential and it is popular to be used in integrated positioning system.

Fingerprinting and trilateration are two main approaches for Wi-Fi-based positioning. Then they will be introduced in this section.

2.3.1. Trilateration

Similar with the trilateration law which GPS fallows as described above, trilateration-based Wi-Fi positioning first calculates the ranges between the device and available Access Points (APs) through the wireless signal propagation model, which can turn measured signal strength to distance. The basic requirements of the method are at least three APs. Then, the device's position is estimated through the use of trilateration.

The distance estimate can be used to generate a circle around each AP on which the device must be. Therefore, the device must be at the position where the circles from each transmitter coincide as shown in Figure 2.9 (a).

However, in fact the circles will not intersect at a single point at all with imperfect information. In this situation an estimate of the position is found by looking for the point that simultaneously minimizes the distance to all circles as shown in Figure 2.9 (b).

Besides, radio signals are extremely variable, particularly indoors, due to being reflected by obstacles or blocked by the walls. Environmental changes such as the people around can also affect the signals. It makes the trilateration extremely unreliable so this method is rarely researched in recent years.

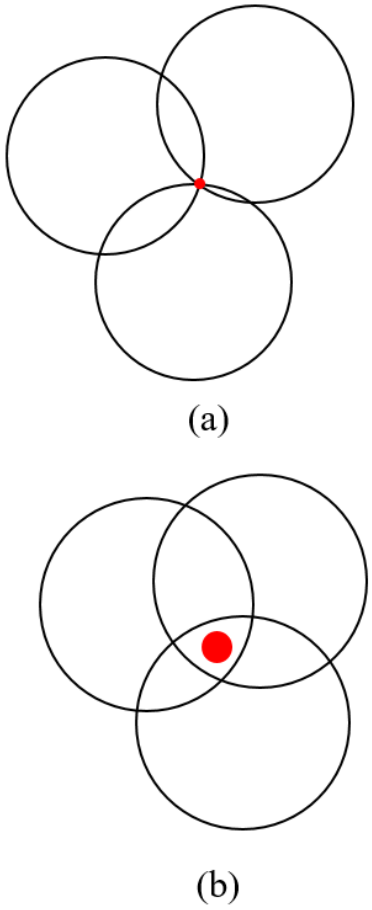


Figure 2.5 Position estimation by trilateration. (a) ideal condition (b) imperfect situation.

2.3.2. Fingerprinting

Wi-Fi fingerprinting is the widely used positioning approach based on Wi-Fi Received Signal Strengths (RSS). This technique does not require knowledge of the positions of the APs and the estimation of distance from AP. Therefore, the environment will not affect the system like the trilateration approach.

Wi-Fi fingerprinting usually has two operating phases: the offline training phase and the online positioning phase, as shown in Figure 2.10.

In the training phase, RSS values from available APs and position information are collected as fingerprints for creating the radio map database. To generate a reliable database, the number of reference points should be big enough to cover the whole area of inter-est. And the RSS from available APs are collected for each reference point. The red part of Figure 2.10 shows the form of the database.

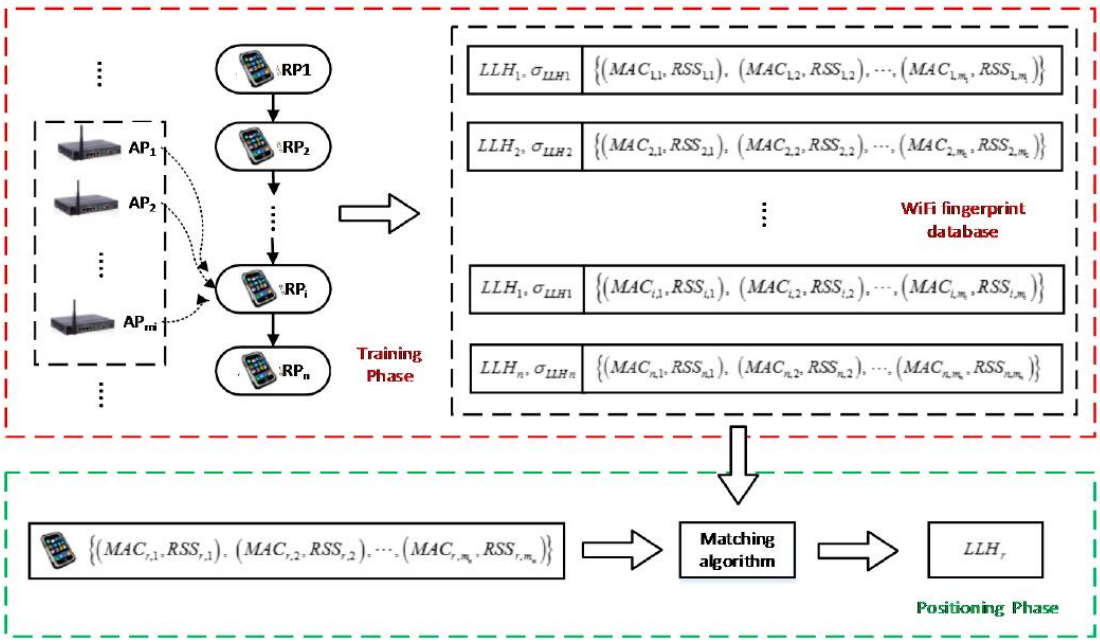


Figure 2.6 Procedure of Wi-Fi fingerprinting.

As shown in Figure 2.10, the fingerprint information at the *i-th* reference point is recorded as:

$$F_i = \left\{ LLH_i, \sigma_{LLHi}, \begin{matrix} (MAC_{i,1}, RSS_{i,1}), (MAC_{i,2}, RSS_{i,2}), \\ \cdots, (MAC_{i,mi}, RSS_{i,mi}) \end{matrix} \right\} \tag{2.6}$$

The state LLH_i and σ_{LLHi} are the coordinate of the *i-th* reference point and its accuracy. And the state $(MAC_{i,j}, RSS_{i,j})$ are the MAC address and RSS of the j-th AP received at this reference point. The m_i is the number of available APs of this reference point.

Then in the positioning phase, the user location will be estimated by comparing the RSS information with that stored in the database.

Zhang et al [12] introduce the nearest neighbor (NN) method. It selects the reference point, which has the minimal signal strength distance as the user's estimated position. The position is calculated as:

$$d_i = \sqrt{\sum_{j=1}^{n_i} \left| SS_{rec,lu}^{j} - SS_{DB,i}^{j} \right|} \text{ , i} \in I_{RP} \tag{2.7}$$

The state d_i is the signal strength distance at reference point RP_i in the database. $SS_{rec,lu}$ is the measured RSS vector at l_u. The $SS_{DB,i}$ is the RSS vector at RP_i. The n_i is the number of Wi-Fi signals received at RP_i. And I_{RP} is the location index set of reference points in the database. Then the coordinates of RP_i which satisfies the condition $d_{i*} = min(d_i | \, i \in I_{RP})$ is determined as the position estimation of l_u.

To optimize Wi-Fi positioning, at first they use a threshold Th_{RSS} to filter out APs with weak RSS. Then, if the minimal signal strength distance at a certain epoch is larger than another given threshold Th_d, the fingerprinting results at this epoch will not be used because the current user location probably has not been stored as a reference points in the database.

Furthermore, to mitigate the error, the k-NN estimation technique is considered. The position is estimated according to k reference points that have the smallest distances, namely, the position estimation is obtained by a weighed sum of the positions of k nearest RPs as:

$$\hat{\boldsymbol{r}} = \sum_{i=1}^{k} \frac{c_i}{C} \boldsymbol{r_i} \tag{2.8}$$

The state $c_i = 1/d_i$ and $C = \sum_{i=1}^{k} c_i$. The $\boldsymbol{r_i}$ is the position of the *i-th* nearest reference point.

Fingerprinting usually provides more accurate position solutions with the cost of survey work in the training phase. However, if the environments changes dramatically, the system may give degraded results and the database needs to be trained again.

Even in the indoor environment, fingerprinting is faced with some problems, such as the multipath effect in some certain environment, or the human body absorb, refract, reflect the signal. And as mentioned above, the outdoor environment will be more complicated and there will be a challenge for Wi-Fi-based positioning to work standalone. Then the integration of different positioning systems will be introduced in next part.

Chapter 3.

Pedestrian Positioning with the Aid of Google Maps Street View

Nowadays, the intelligent mobile devices such as smartphone and smart glasses are widely equipped with embedded camera, commercial level GPS receiver, wireless receiver, and some other sensors such as gyroscope, accelerometer, and magnetometer. It makes visual data associated with geographical tags and camera pose can be easily produced from these devices in our daily lives. However, as mentioned above, the accuracy of these sensor are easily influenced in the urban area and it will be not satisfied if we directly use the data from these sensors for positioning or camera pose estimation. Therefore, to estimate an accurate positioning result, beside the methods to improve the accuracy of positioning performance of them or integrate GPS positioning, Wi-Fi-based positioning, PDR-based positioning these kinds of basic positioning methods, visual information is also an popular source to be used for positioning. Then, in this chapter, at first we will introduce some related works about vision-based positioning method. Because Google Maps Street View is an important source in our proposed method, before introducing our proposed method about pedestrian positioning with the aid of Google Maps Street View, how to get the information and how to use or analyze the information form Google Maps Street View should be introduced.

3.1. Vision-based positioning and related works

Vision-based positioning fallows the same basic principles of landmark-based and map-based positioning. But vision-based positioning relies on optical sensors rather than GPS, wireless, pedestrian dead reckoning and inertial sensors. The advantage of these types of sensors lies in their ability to directly provide position information of distance information. However, as mentioned above, recent techniques based on these types of sensors have several limiting problems in the urban area, such as the NLOS and multipath of GPS, the drift of PDR, and the variation of Wi-Fi signals. At that time,

visual sensing can provide the pedestrian with an incredible amount of reliable information about its environment. Especially the camera on the new generation of intelligent eyeglass-type wearable device named smart glasses, which can provide a same view as what the pedestrian saw in his eyes as shown. Figue 3.1 shows the smart glasses of EPSON Moverio BT-300 which has a camera built into the headset. For video resolution, it supports up to 1920x1080 pixels which can satisfy our requirement. The camera will make the device can recognize the environment in pedestrian's scene and help to generated the AR navigation guide on the screen of the smart glasses.

Figure 3.1 EPSON Moverio BT-300 and the position of camera sensor

Before talking about different vision-based positioning method and sources, the model of the vision sensor, the camera, need to be introduced.

3.1.1. Camera Model

Photometric cameras using an optical lens can be modelled as a pinhole camera. Figure 3.2 shows the geometry of a pinhole camera model and Figure 3.3 shows the process of projecting a feature in the world coordinates $P = (U, V, W)$ into the pixel coordinates (u, v).

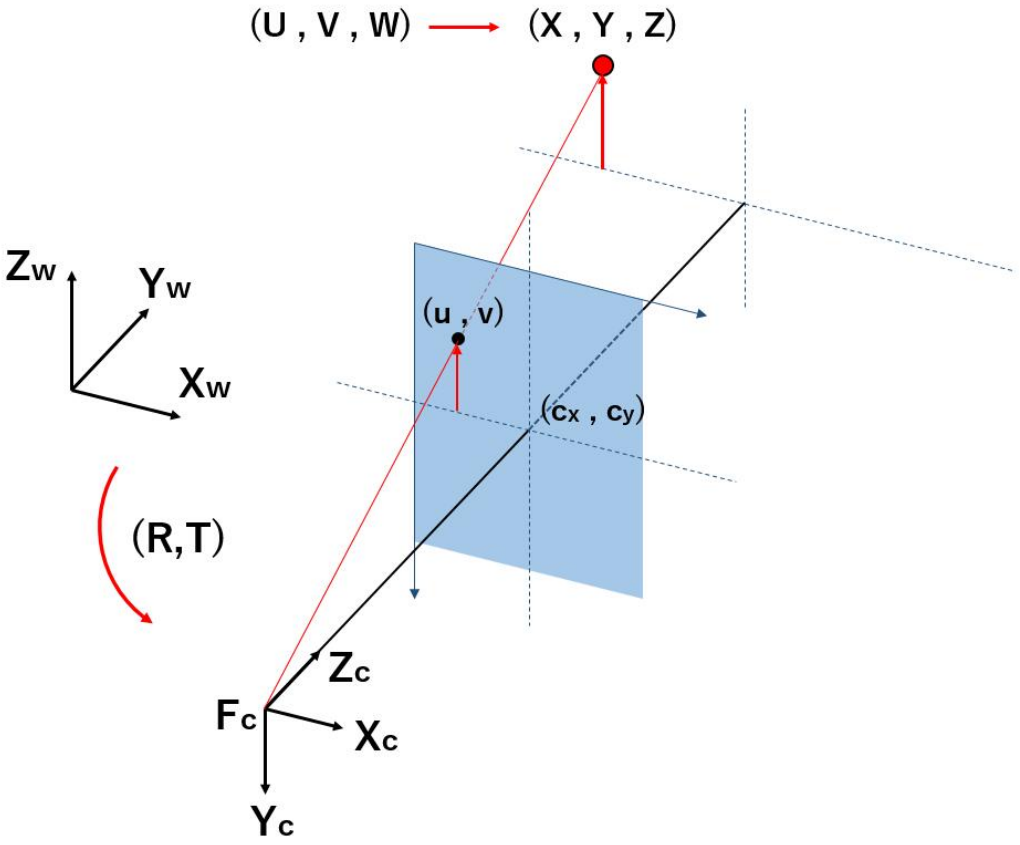


Figure 3.2 The geometry of a pinhole camera model

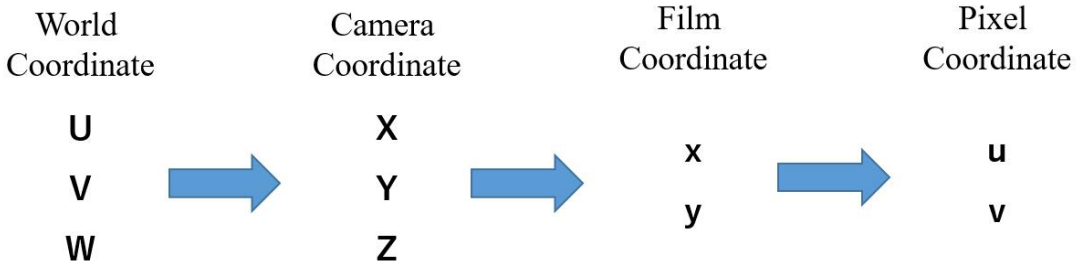


Figure 3.3 The process of projecting a feature in the world coordinates into the pixel coordinates

As shown in Figure 3.2 and Figure 3.3, to project a feature in the world coordinates $P = (U, V, W)$ into the pixel coordinates (u, v), there are many steps [15].

At first, when the camera is moving in the real world, to project a point in the real world to the camera film, the rotation and translation of the camera around the scene in

the world need to be used to transform the coordinates in the world coordinate system to the camera coordinate system as shown in Figure 3.2.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.1}$$

where

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{3.2}$$

is the rotation matrix and

$$t = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{3.3}$$

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

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

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

$$x = f\frac{X}{Z} = \frac{x'}{z'} \tag{3.5}$$

$$y = f\frac{Y}{Z} = \frac{y'}{z'} \tag{3.6}$$

where f is the camera focal length in distance unit.

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

$$u = \frac{1}{s_x} f \frac{X}{Z} + c_x = f_x \frac{X}{Z} + c_x \tag{3.7}$$

$$y = \frac{1}{s_y} f \frac{Y}{Z} + c_y = f_y \frac{Y}{Z} + c_y \tag{3.8}$$

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

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

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.9}$$

where

$$\mathrm{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

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

From Figure3.2, we can see that the range information is lost in this projection, but the angle or orientation of the feature can be obtained if the focal length is known and the lens does not cause distortion. So a single camera only provides only information about the direction of the features exist in the pedestrian's view, while as it doesn't provide any information regarding the depth. To get the three-dimensional location of a feature, multiple images from different viewpoints are required. The multiple images from different viewpoints can be get from a monocular camera while it is moving and structure from motion (SfM) technique can be used to estimate three-dimensional structures from two-dimensional image sequences. And there is a type of

camera with two or more lenses with a separate image sensor or film frame for each lens named stereo camera. This allows the camera to simulate human binocular vision, and therefore gives stereo camera the ability to capture three-dimensional location of the observed features in the environment. Figure 3.4 shows the image from one of the eyes in a stereo camera and the depth map calculated from the images of it.

(a) (b)

Figure 3.4 Stereo camera images. (a) Image from the left camera; (b) Depth map

Although the smart glasses of HoloLens form Microsoft is equipped with stereo camera to recognize the three-dimensional scene and provide high quality AR applications, considering that nowadays most of smartphones and other kinds of smart glasses only have monocular camera, we choose to focusing on the vision-based positioning utilizing the monocular camera.

3.1.2. Simultaneous Localization and Mapping (SLAM)

The problem of vision-based positioning is to determine the position and orientation of the pedestrian camera by matching the sensed visual features in an image or sequence to the object features provided by landmarks or maps. Despite mapping and localization can be performed as independent tasks, they are closely related. In order to build a map, the pose of the structures and the obstacles of the environment needs to be known. On the other hand, during localization, the pose of the agent is computed against a reference map. When there is no map information, the problem becomes to Simultaneous Localization and Mapping (SLAM).

SLAM is widely used in robot navigation. It is the problem for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map [16].

In the SLAM, both the trajectory of the platform and the location of all landmarks are estimated on-line without the need for any a priori knowledge of location.

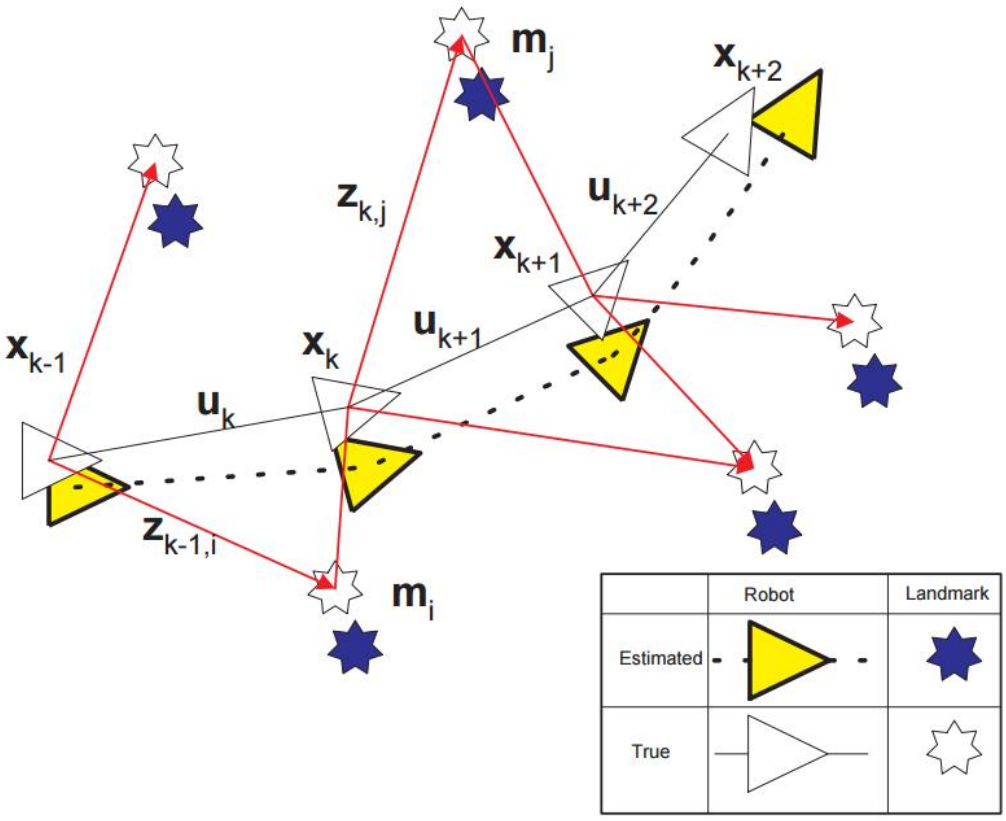


Figure 3.5 Process of the SLAM.

Figure 3.5 [16] shows the process of the SLAM problem. In a SLAM problem, a simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations.

Consider a mobile robot moving through an environment taking relative observations of a number of unknown landmarks using a sensor located on the robot. At a time instant *k*, the location and orientation of the robot is described as x_k. And u_k is the control vector which is applied at time *k*-1 to drive the robot to the state x_k at

time *k*. The vector m_i is used to describe e location of the *i*-th landmark, whose true location is assumed time invariant. And z_{ik} is an observation taken from the vehicle of the location of the *i*-th landmark at time *k*. When there are multiple landmark observations at any one time or when the specific landmark is not relevant to the discussion, the observation will be written simply as z_k. Therefore, the objective of the SLAM problem is to compute the probability distribution at time *k* as:

$$\mathrm{P}(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \tag{3.11}$$

The state $Z_{0:k}$ is the set of all landmark observations and $U_{0:k}$ is the history of control inputs. This probability distribution describes the joint posterior density of the landmark locations and robot state at time *k* given the recorded observations and control inputs up to and including time *k* together with the initial state of the robot. The SLAM algorithm is now implemented in a standard two-step recursive prediction (time-update) correction (measurement-update) form.

The time-update is:

$$\begin{aligned} &\mathrm{P}(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0) \\ &= \int \mathrm{P}(x_k \mid x_{k-1}, u_k) \times \mathrm{P}(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1} \end{aligned} \tag{3.12}$$

And the measurement-update is:

$$\begin{aligned} &\mathrm{P}(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \\ &= \frac{\mathrm{P}(x_k, m \mid x_k, m) \times \mathrm{P}(x_k, m \mid Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1}}{\mathrm{P}(z_k \mid Z_{0:k-1}, U_{0:k})} \end{aligned} \tag{3.13}$$

Like many inference problems, the solutions to inferring the two variables together can be found, to a local optimum solution, by alternating updates of the two beliefs.

3.1.3. Descriptors in vision-based positioning

In order to perform mapping and localization tasks using vision, it is necessary to describe the acquired images and be able to compare these descriptions. Consequently,

the quality of the map and the posterior localization will directly rely on the method used for visually describing the different environment locations. Therefore, according to the description method, different approaches can be classified as approaches based on global descriptors and approaches based on local features [17].

Global descriptors describe the image in a holistic manner, using the full image as input to the process. These descriptors are normally very fast to compute, what simplifies the matching process between images and reduces the computational needs of mapping and localization tasks. This kind of descriptor has been used in several applications comprising scene classification, giving good results in all cases. Histograms, the Gist descriptor, vertical regions and the Discrete Fourier Transform (DFT) and some other approaches can be used as global descriptors.

Global descriptions work well for capturing the general structure of the scene, but they are not able to cope well with several visual problems like partial occlusions or camera rotations. These problems have been addressed more intensively through the recent development of local features.

During the extraction step, a set of distinctive local features, which capture the essence of the image, are detected. These features can be derived from the application of a neighborhood operation or searching for specific structures within the image, such as corners, blobs or regions. Then, a description step is performed, where some measurements are taken from the vicinity of each local feature to form a descriptor. Initially, descriptors were formed as a multi-dimensional floating-point vectors.

In order to identify the same local features in other images, local features need to be invariant to certain properties, such as camera rotations or affine transformations. Therefore, a good feature detector should have the properties of repeatability, distinctiveness, locality, quantity, accuracy and efficiency. The most important property is repeatability, that can be achieved either by invariance, when large deformations are expected because of relevant view changes, or by robustness, in case of relatively small deformations.

Scale-Invariant Feature Transform (SIFT) feature is the most popular feature nowadays. SIFT is widely used in the computer vision and image processing algorithm. And even many other local features are based on SIFT. The SIFT features are local and based on the appearance of the object at particular interest points, and are invariant to image scale and rotation. They are also robust to changes in illumination, noise, and

minor changes in viewpoint. In addition to these properties, they are highly distinctive, relatively easy to extract and allow for correct object identification with low probability of mismatch. They are relatively easy to match against a (large) database of local features but however the high dimensionality can be an issue, and generally probabilistic algorithms such as k-d trees with best bin first search are used. Object description by set of SIFT features is also robust to partial occlusion; as few as three SIFT features from an object are enough to compute its location and pose. Recognition can be performed in close-to-real time, at least for small databases and on modern computer hardware.

Speeded Up Robust Features (SURF) is another famous scale and rotation invariant descriptor for detecting features from image. The detection process is based on the Hessian matrix. SURF descriptors are based on sums of two-dimensional Haar wavelet responses, calculated in a 4×4 subregion around each interest point. It detects region features from an image and obtains the location and the descriptor vector of each interest point.

Harris corner is probably the most widely interesting point detector used due to its strong invariance to scale, rotation and illumination variations, as well as image noise. The basic idea behind this algorithm is to evaluate the derivative of the intensity with respect to the location. The edges are then detected where the derivative gets very large. To cover changes of intensity in each direction, the Harris Corner calculates the derivative in the x-direction and in the y-direction.

Not only these famous features, other feature like line features, edge features or plane features can also be used in the vision-based positioning.

Although the popular features like SIFT is invariant to image rotation, it is still faced with a challenge when the view point changes a lot. In this kind of situation, they cannot find enough matched key features.

Considering that the behaviors of pedestrians are complex and we hope to find a match in most of situations, so the ASIFT method [18] could be a choice. ASIFT simulates a set of sample views of the initial images, obtainable by varying the two camera axis orientation parameters, namely the latitude and the longitude angles. Then it applies the SIFT method itself to all these images to find and match SIFT descriptors. Although ASIFT shows good performance to find key features matches in the situation with a large transition tilts, it is still hard to deal with the transition tilts more than 32.

In the experiment, we found it is difficult to match the reference images with the pedestrian's heading angle changes a lot from it. So we need to change to some other features.

Then Virtual Line Descriptor (kVLD) [19] is found to deal with important changes from viewpoint, illumination and occlusion in urban area. The kVLD is applied to determine the inlier feature point correspondences. It is a SIFT-like descriptor by signing virtual lines to the points with geometrical consistency. The algorithm computes and matches a k connected virtual line graph to reject the outliers. This algorithm evaluates a geometric consistent match by searching for at least K other matches in the neighborhood of that potential match. However, although kVLD deal well with large view changes, it is time-consuming and in our case, a real-time response is desirable.

GMS (Grid-based Motion Statistics) [20][JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan Dat Nguyen, Ming-Ming Cheng, GMS: Grid-based Motion Statistics for Fast, Ultra-robust Feature Correspondence, Conference on Computer Vision and Pattern Recognition (CVPR), 2017] is a simple means of encapsulating motion smoothness as the statistical likelihood of a certain number of matches in a region. It enables translation of high match numbers into high match quality which provides a real-time, ultra-robust correspondence system. Considering the performance for both matching quality and execute efficiency, we choose GMS as our image matching algorithm.

Figure 3.6 Matching between a reference image (left) with the query image (right) based on GMS

Figure 3.6 show one of the matching result in the experiment using GMS, in which situation it is difficult to find enough reliable matched key features from SIFT standalone and ASIFT method. Therefore, GMS was choose in our proposed method.

3.1.4. Related works on positioning with the aid of Google Maps Street View.

After choosing a reliable descriptor, then the map is needed for vision-based positioning as mentioned above.

Google Maps Street View can be considered as a very comprehensive and large database provided by Google, which consists of geo-tagged 360º panoramic images of almost all main streets and roads in a number of countries. The panoramic images of Google Maps Street View are recorded by a spherical arrangement of cameras and the localization comes from the Inertial Measurement Unit (IMU) and GNSS. There is a distance with range of 5 to 20 meters between two successive panoramic images. Because of its characteristics and quality, Google Maps Street View attracts more and more attentions in the field of computer vision, especially for localization. We found that beside the panoramic images, we could also get the depth map of each panoramic images from the API, which is provided by Google. By combining the panoramic images and depth map information, local 3D models can be created. By using this 3D model, virtual views in all the places can be generated and there will be no limitations from the fixed interval position of the panoramic images.

Robertson et al. [26] build a database of views of building facades to determine the pose of a query view provided by the user, using a novel wide-baseline matching algorithm that can identify corresponding building facades in two views despite significant changes of viewpoint and lighting. Figure 3.7 [26] shows the building facades database they build and the transferred building outlines recovered from the matching results. This research suggest us that using the building facades database in the image matching can help us to recognize the shape changing of the building walls in the views when pedestrian is moving and rotating.

Majdik et al. [20, 21] use Street View images to localize a Micro Aerial Vehicle by matching images with strong view point changes by generating virtual affine views to Google Maps Street View images and add 3D models of buildings as input to improve its accuracy. Torii et al. [22] match descriptors computed directly on queried image and multiple Google Maps Street View panoramas with learning a distinctive bag-of-word model to localize. Kim et al. [27] address the problem of accurate

localization of a place depicted in a query image using a large geotagged image database. Features that often mislead to false locations are the ones that are unstable maintain alignment to the corresponding feature in the same object under viewpoint

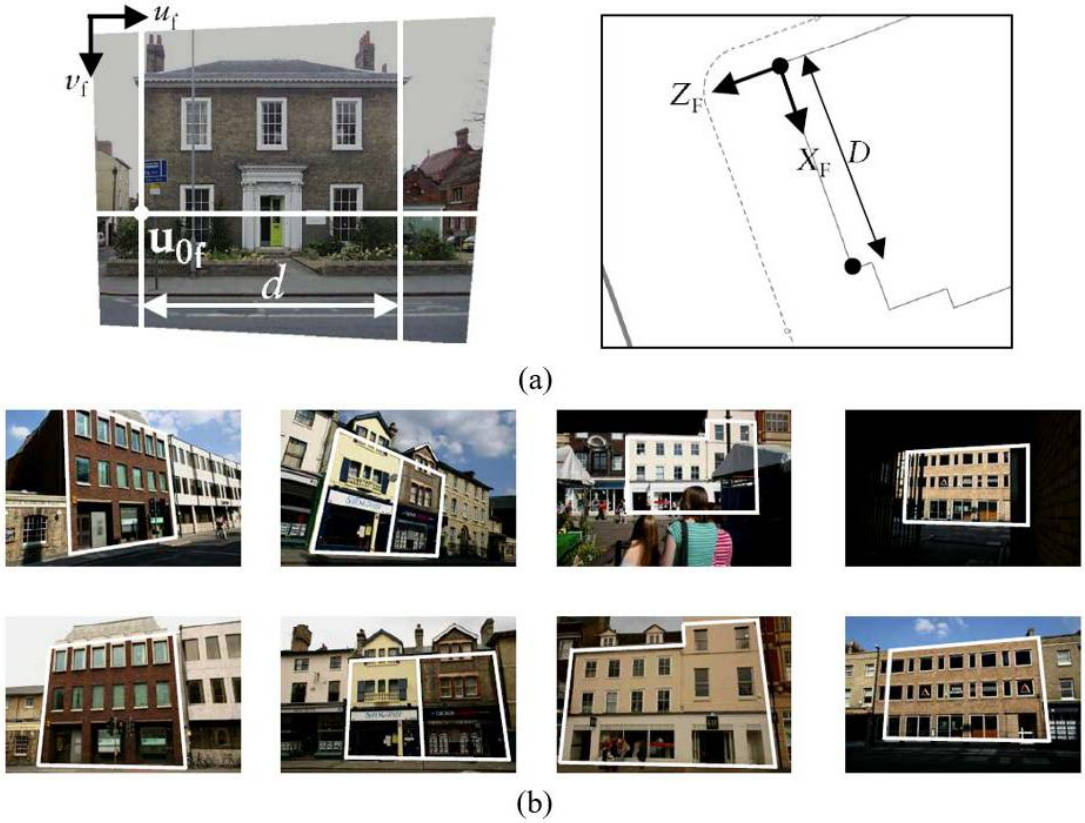


Figure 3.7 Building facades database and matching results. (a) Building facades in the database are associated with a meaningful coordinate system using a map. (b) Illustrative database retrieval results and transferred building outlines.

and illumination changes, or the ones prevalent in space such as features belong to pedestrians and cars. Therefore, they discover features that are both robust and distinctive by performing geometric verification of images depicting the same scene and proposed a novel method for classifying reliable features for place recognition by learning a weighted linear SVM classifier on examples of geometrically verified features and those that are not verified. Knopp et al. [28] also address the key problems in place recognition that the presence of objects such as trees or road markings, which frequently occur in the database and hence cause significant confusion between different places. To avoid features leading to confusion of particular places, they use geotags attached to database images from the Google Maps Street View as a form of supervision and develop a method for automatic detection of image-specific and

spatially-localized groups of confusing features, and demonstrate that suppressing them significantly improves place recognition performance while reducing the database size. However, these methods only solve the place recognition problem and provide topological localization via image matching.

Some other researches choose to focusing on localizing images in large scale metrical maps built from Structure from Motion (SfM). Irschara et al. [23] build accurate point clouds using structure from motion and then compute the camera coordinates of the query image. Zamir et al. [24] leverage a structured dataset of about 100,000 images build from Google Maps Street View as the reference images and proposed a localization method in which the SIFT descriptors of the detected SIFT interest points in the reference images are indexed using a tree. In order to localize a query image, the tree is queried using the detected SIFT descriptors in the query image. A novel GPS-tag-based pruning method removes the less reliable descriptors and a smoothing step with an associated voting scheme is utilized. It allows each query descriptor to vote for the location its nearest neighbor belongs to, in order to accurately localize the query image. A parameter called Confidence of Localization which is based on the Kurtosis of the distribution of votes is defined to determine how reliable the localization of a particular image is. In addition to localizing single images, they propose a novel approach to localize a non-sequential group of images. Agarwal et al. [25] estimate the 3D position of tracked feature points from short monocular camera sequences and then compute the rigid body transformation between the Google Maps Street View panoramas and these estimated points by model the problem as a non-linear least squares problem of two objectives. Sattler et al. [29] also use large scale 3D models of urban scenes for accurate image-based localization and focus on the important bottleneck which is the computation of 2D-to-3D correspondences required for pose estimation. They derive a direct matching framework based on visual vocabulary quantization and a prioritized correspondence search to improve the performance of direct 2D-to-3D matching methods.

Although these works have good performance on image-based positioning, there is still a considerable potential for improving the positioning accuracy performance. Based on the advantages and disadvantages of the dataset generation and positioning method of the related works, we proposed a method to provide accurate pedestrian positioning in urban areas with the aid of visual matching with the dataset from Google

Maps Street View. Considering that Google Maps Street View is the important source in our proposed method, in the next part, we will introduce the pipeline of Google Maps Street View.

3.2. Proposed method

3.2.1. Objective

The geo-tagged images captured by the camera on the smart glasses is the input of our system. But the geographic information is not accurate, the accuracy of commercial GPS is about 20 meters and the error of magnetometer is about 20 degrees. As for the source, we use street view images from the Google Street View Image API and the textured local 3D model from the depth map and panorama image.

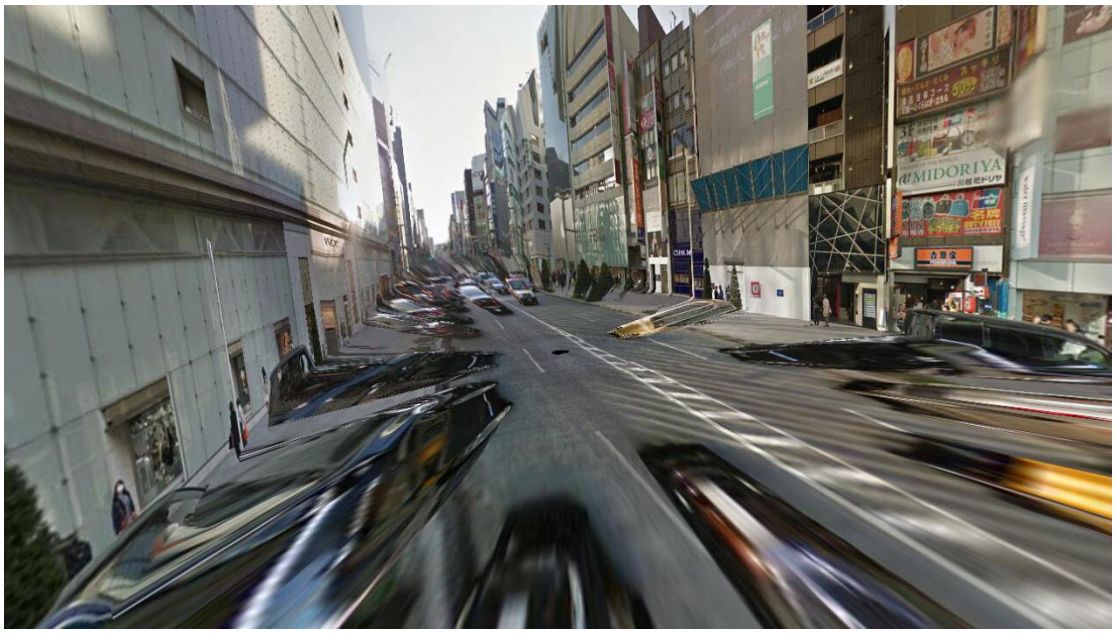


Figure 3.8 Virtual view from textured local 3D model

The objective of our proposed method is to search a virtual view image from Google Street View which is the most similar with the query image from smart glasses. After the best-matched virtual view image is found, the pedestrian’s position can be decided as the location information of this virtual view image, as shown in Figure 3.14.

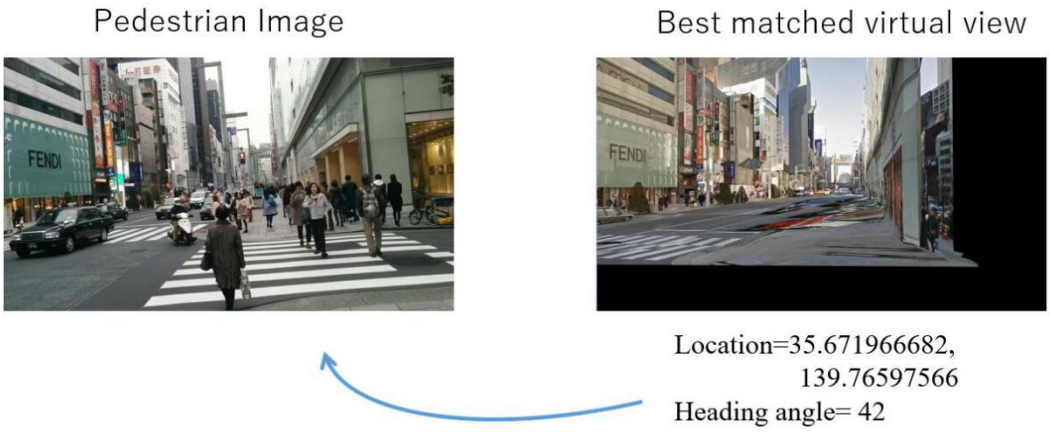


Figure 3.9 Objective of the proposed method

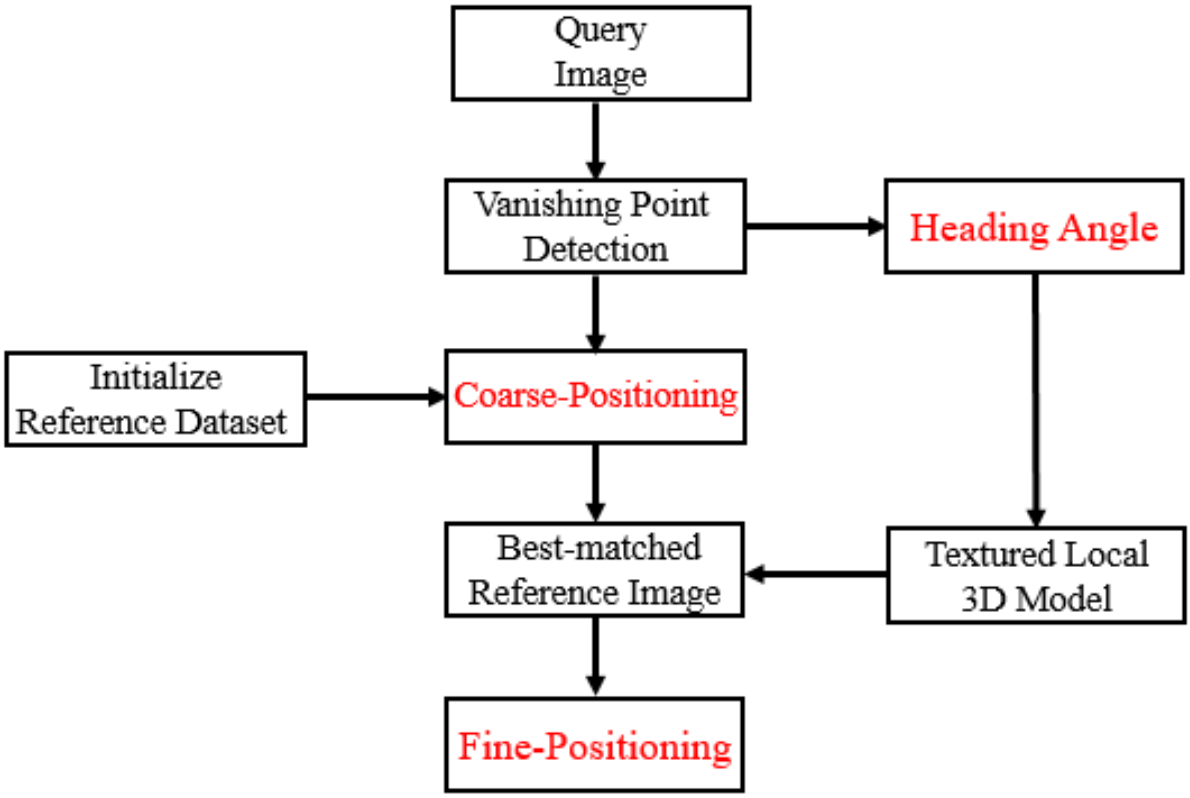


Figure 3.10 The flowchart of our proposed method.

Figure 3.15 shows the flowchart of our proposed method. As can be seen from the figure, there are three main steps, which are the heading angle detection, coarse-positioning and fine-positioning.

In this work, the similarity between a virtual view image and the query image is

evaluated by the average distance of each matched key features. The dissimilarity is formulated as:

$$e = \frac{1}{N}\sum \sqrt{\left(x_{Pedestrian,i} - x_{VirtualView,i}\right)^2 + \left(y_{Pedestrian,i} - y_{VirtualView,i}\right)^2} \quad (3.14)$$

where $\left(x_{Pedestrian,i}, y_{Pedestrian,i}\right)$ is the pixel coordinates of the *i*-th matched features in the pedestrian image and $\left(x_{VirtualView,i}, y_{VirtualView,i}\right)$ is the *i*-th matched features in the virtual view. Therefore, the main purpose is to search a coordinates of $(lat, lon, heading)$ to make the virtual view reach the minimum value of the error *e* as:

$$VirtualView(lat, lon, heading) = argmin\ \mathbf{e} \quad (3.15)$$

To solve this equation, we should decide one parameter at first.

3.2.2. Heading angle detection from vanishing point

Heading angle indicates the compass heading of the camera direction which can be obtained from attitude determination, which is a critical point in navigation systems. In the urban area, the environment will easily influence magnetometer and the error may become 20 degrees sometimes. Kessler et al [32] addressed configuration of projected edges in the camera image, which is shown as Vanishing Points (VPs) and Vanishing Lines (VLs). This method can be used to determine the camera orientation. Figure 3.16 [32] show the projection of a line to the camera image plane.

The vanishing point may also be referred to as the direction point, as lines having the same directional vector, will have the same vanishing point or converge at the same vanishing points. Therefore, as shown in the Figure 3.17 [32], in the environment like urban area, in where the buildings are normally in a same direction next to a road, the vanishing points should be fixed when the heading angle of the camera did not change.

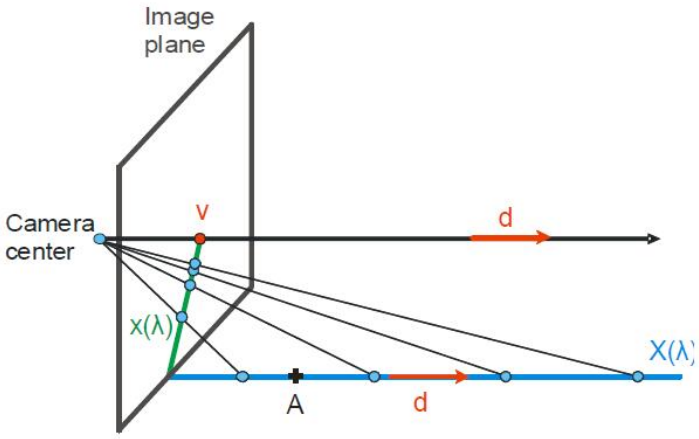


Figure 3.11 Projecction of a line in 3D-space onto the camera image plane

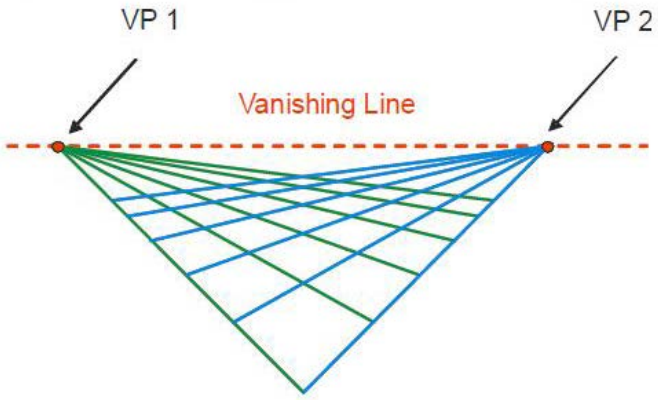


Figure 3.12 Vanishing Line and Vanishing Points within that line in the 3D vision.

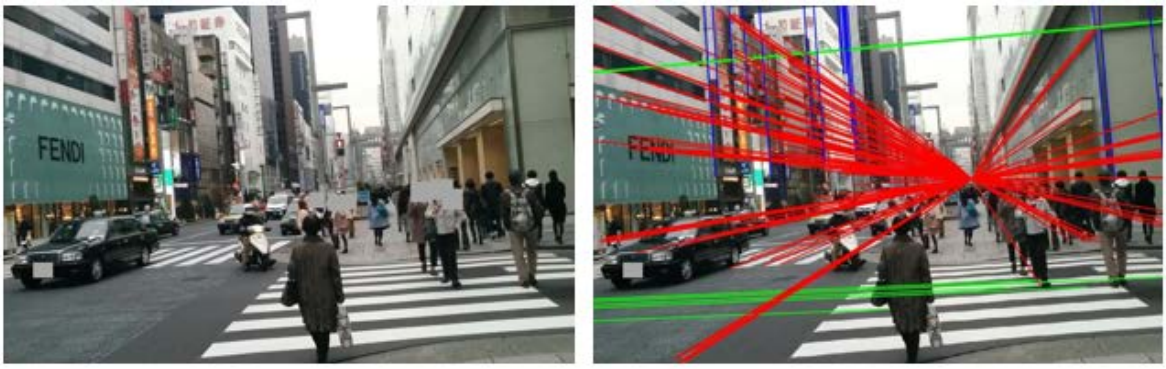
Considering the camera model which was introduces above, the coordinates of the vanishing point in the image plane should be:

$$V = \begin{bmatrix} f_x \sin\theta \\ f_y \sin\varphi \end{bmatrix} \tag{3.16}$$

where θ is the heading angle with 0 and 180 indicating the direction of the road.

In our proposed method, at first a Canny edge detector was used to find local edges in the pedestrian image. Then a Hough line detector was used to detect the lines in the edges detected from Canny. The vanishing points detection and line grouping method from Duan et al. [33] was used to estimate the vanishing point.

Figure 3.13 Process of vanishing point detection



(a) (b)

Figure 3.14 Heading angle detection from vanishing point.

Figure 3.18 shows the process of the vanishing point detection and Figure 3.19

shows the results of lines detection and vanishing points detection. (a) is the query image and (b) shows the results of vanishing point detection. After the vanishing point is detected, we can use equation 3.16 to estimate the camera orientation. Therefore, the problem of equation 3.15 becomes to:

$$VirtualView(lat, lon) = argmin_{heading*}\ \mathbf{e} \qquad (3.17)$$

3.2.3. Coarse-positioning by searching similar street view image

In order to calculate the accurate positioning result, we propose a coarse-to-fine two-stage method. In the coarse positioning step, at first we determine the roads with vertical distance less than 20 meters from its geo-tag. Then we search the candidate positions as shown in Figure 3.21. Rather than creating a building facades datasets with the distance less than 20 meters to the vertical point on each of these roads as the reference dataset, we use a convergence method to reduce the processing time of image matching.

Firstly, considering the maximum error of GPS is 20 meters, we require two locations less than 20 meters to the vertical point on both left and right side as the initial dataset, as and in Figure 3.21. We can get the new reference position based on feature matching between reference image and query images and transition matrix gained from homography matrix. For each candidate position, we collect the reference image with the heading angle of query image. Then we use GMS matching algorithm to evaluate the similarity between the query image and the images from the dataset. In addition, for each reference images, the geographic coordinates and heading angle are tagged, which suggesting the we can get rotation translation matrix from homography matrix and camera intrinsic parameters matrix. Then we keep the more similar one and discard the other location. Once we find the similarity score reaches the highest, that new position will be chosen as the coarse position. We can see the process of it in Algorithm1.

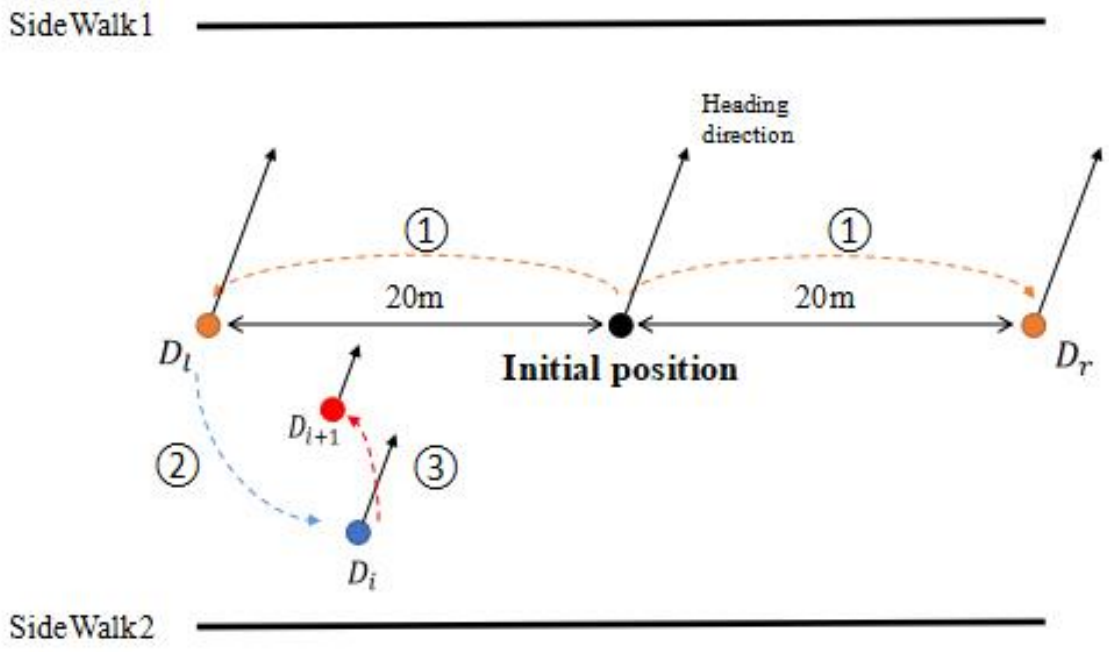


Figure 3.15 Coarse-positioning by searching similar street view image.

Algorithm1: Coarse-positioning by searching similar street image

Input: Geo-tagged query image $\boldsymbol{I}$

Output: Interested area position $\boldsymbol{D}$ **(λ, φ)**

Initialize reference dataset $\boldsymbol{R}$ with 2 street view images $\{\boldsymbol{R_l}, \boldsymbol{R_r}\}$ in position $\boldsymbol{D_l}$ and $\boldsymbol{D_r}$ with heading angle $\boldsymbol{\theta}$

For image **r** $\boldsymbol{\epsilon}$ **R** do

n = GMS_Matches ***(r, I)***

$\boldsymbol{S}$ = GMS_Matches_Scores ***(r, I)***

$\boldsymbol{m}$ = Similarity ***(s, n)***

$\boldsymbol{H}$ = Homography ***(r, I)***

If $\boldsymbol{m}$ < threshold $\boldsymbol{\tau}$ do

Discard $\boldsymbol{r}$

Else

update $\boldsymbol{D}$ $\boldsymbol{D_{new} = f(differ(H_{i-1}, H_i), d_{i-1,i})}$

update $\boldsymbol{R} = \{\boldsymbol{D_{new}}, \boldsymbol{\theta}\}$

End if

If $\boldsymbol{S_{i+1}}$ <= $\boldsymbol{S_i}$ then $\boldsymbol{D_{i+1} = D_i}$

End for

Return $\boldsymbol{D_i(\lambda, \varphi)}$

Both the query image recorded by the camera, the downloaded building facade images from Google Maps Street View, and the virtual view image can be considered as taken from a pinhole camera model.

As we mentioned above, the whole process of the projection can be shown in the equation as:

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.18}$$

Because there is no resize or re-sampling of the images, so the scale factor *s* should be 1 and the equations becomes:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.19}$$

When features in matched image pairs of query image and building façade image, virtual view images and building façade image, we can consider that these matched features are the same feature match $P = (U, V, W)$ in the world coordinates.

Because the camera pose of the building façade images are known in the dataset, we can consider these building façade images as static scenes. So we can get features in the query image as (u_c, v_c):

$$\begin{pmatrix} u_c \\ v_c \\ 1 \end{pmatrix} = K_c(R_c|t_c) \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.20}$$

where K_c is the camera intrinsic parameters matrix of the smart phone camera, which is known. $(R_c|t_c)$ is the rotation-translation matrix from the camera pose of matched building façade image to the camera pose of query image.

Also, in a same way, we can get the matched features in the generated virtual view

images as (u_v, v_v):

$$\begin{pmatrix} u_v \\ v_v \\ 1 \end{pmatrix} = K_v(R_v|t_v)\begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.21}$$

where K_v is the camera intrinsic parameters matrix of the virtual camera. Because we calibrate the parameters of the virtual camera with the camera parameter of our real camera.

$$K_v = K_c \tag{3.22}$$

$(R_v|t_v)$ is the rotation-translation matrix from the camera pose of matched building façade image to the virtual view image.

Because the reference dataset is composed of images perpendicular to the building's walls, to simplify the problem, we can assume that all the features are places on a plane, whose $W = 0$. Then the equation 3.20 becomes:

$$\begin{pmatrix} u_c \\ v_c \\ 1 \end{pmatrix} = K_c \begin{pmatrix} r_{c11} & r_{c12} & r_{c13} & t_{c1} \\ r_{c21} & r_{c22} & r_{c23} & t_{c2} \\ r_{c31} & r_{c32} & r_{c33} & t_{c3} \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} U \\ V \\ 0 \\ 1 \end{pmatrix} \tag{3.23}$$

$$\begin{pmatrix} u_c \\ v_c \\ 1 \end{pmatrix} = K_c \begin{pmatrix} r_{c11} & r_{c12} & t_{c1} \\ r_{c21} & r_{c22} & t_{c2} \\ r_{c31} & r_{c32} & t_{c3} \end{pmatrix}\begin{pmatrix} U \\ V \\ 1 \end{pmatrix} \tag{3.24}$$

$$\begin{pmatrix} u_c \\ v_c \\ 1 \end{pmatrix} \sim K_c(r_{c1}, r_{c2}, t_c)\begin{pmatrix} U \\ V \\ 1 \end{pmatrix} \tag{3.25}$$

where r_{c1}, r_{c2} are the first and second column of the rotation matrix. Our main goal is to estimate the position of the query image from the translation matrix and this assumption does not affect the values in the translation matrix. So this simplification can work in our research.

Also, in a same way, we can get the equation 3.21 becomes:

$$\begin{pmatrix} u_v \\ v_v \\ 1 \end{pmatrix} \sim K_v(r_{v1}, r_{v2}, t_v) \begin{pmatrix} U \\ V \\ 1 \end{pmatrix} \tag{3.26}$$

Because the building façade image is considered as the static scene, there will be no parameter of $(R_g|t_g)$. Therefore, features in the reference image can be projected as (u_g, v_g):

$$\begin{pmatrix} u_g \\ v_g \\ 1 \end{pmatrix} \sim K_g \begin{pmatrix} U \\ V \\ 1 \end{pmatrix} \tag{3.27}$$

where K_g is the camera intrinsic parameters matrix of Google Maps Street View. Since Google didn't provide the intrinsic parameters of their cameras, we use offline calibration to get the focal lengths and select the center of the image as the principal point.

In the image matching, if multiple features pairs has been matched, the perspective transformation between two images can be calculated as a Homography matrix H. So we can associate (u_g, v_g) and (u_c, v_c) by the 3×3 matrix H_c as:

$$\begin{pmatrix} u_c \\ v_c \\ 1 \end{pmatrix} = H_c \begin{pmatrix} u_g \\ v_g \\ 1 \end{pmatrix} \tag{3.28}$$

We also can associate (u_g, v_g) and (u_v, v_v) by the matrix H_v as:

$$\begin{pmatrix} u_v \\ v_v \\ 1 \end{pmatrix} = H_v \begin{pmatrix} u_g \\ v_g \\ 1 \end{pmatrix} \tag{3.29}$$

So, by jointing equations 3.25, 3.27 and 3.28, we can get the rotation-translation matrix of query image as

$$(r_{c1}, r_{c2}, t_c) = K_c^{-1} H K_g \tag{3.30}$$

By jointing equations 3.26, 3.27 and 3.29, we can get the rotation-translation

matrix of virtual view image as

$$(r_{v1}, r_{v2}, t_v) = K_v^{-1} H K_g \tag{3.31}$$

We can estimate a pair of latitude and longitude from the relative offset between the query image and a matched building facade image and a pair of latitude and longitude between the virtual view image and the same building facade image.

Because the parameter of K_v and K_g are manually decide so we cannot directly use the decomposed translation matrix to estimate the position of query image and virtual. Bus the difference between the estimated translation matrixes t_c and t_v can be used to suggest which side of road and the distance we should move.

3.2.4. Fine-positioning by matching virtual view with query image

After we get the coarse position, a reference virtual view will be generated in the center of the road near that coarse position by using the textured local 3D model from the depth map and panorama image provided by Google Street View API.

Similar with the coarse position estimation, we use Algorithm 1 to find the best-matched candidate in 3D model.

Figure 3.16 GMS based matching result.

3.3. Experiment Result

We selected the Ginza area in Tokyo as the experiment spot. Because of the density of the tall and similar buildings, pedestrians are often confused to find their position and destination in this area. Our goal is to estimate an accurate positioning result in this

kind of areas. In the experiment, the smart glasses Epson Moverio BT-300 was used. The image matching was conducted based on images captured from the camera on this smart glasses. To simulate the situation when a visitor is walking on the sidewalk in a new environment, we looked around while walking in the trajectory to capture images from the camera and collect GNSS positioning data simultaneously.

As shown in Figure 3.26, there are 5 routes in our experiment. The blue lines with green dots are our positioning results while the purple dots are the ground truth. The ground truth is obtained by experimenter walking along the pre-decided routes, the

average error of ground truth is no more than 14.2cm.

Figure 3.17 Walking trajectory of the experiments in Ginza: walking trajectory (blue line), positioning result (green dot), and ground truth (purple dot)

Figure 3.18 Errors in virtual view images

We adopt the distance error, correct side rate and the positioning availability to evaluate the performance of the proposed method. Positioning availability indicates the successful positioning rate of our proposed method. From Table 1, we can see our results shows very high correct side rate in each test. But the positioning availability rate of Route 3 and Route 4 is lower than other routes. This problem is coming from the error in the depth map from Google Maps Street View. This error leads to the failure of image matching. Thus, the positioning result cannot be provided in these cases. Table 2 shows the distance error of the proposed method in each test route. We can see that the lateral error is lower than longitude error, the cause is the virtual view images in longitude direction are very similar. When we move a small distance, the detected features of images won’t change a lot, which may cause a relative large error.

Table 3 compares our results with others. Same with us, Zamir et al [22] also used images from Google Map Street View as the dataset, but our matching algorithm is more effective and we use a coarse-to-fine positioning method which including the generation of virtual view. Sattler et al. [13] used large scale 3D models of urban scenes for accurate image-based localization, which reduces the computation time but the accuracy is relatively low. Deng et al.[23] built scene 3D point cloud which requires more computation resources and they reached a mean error of 4.729m. Table III shows that the accuracy of our method outperforms others, compared with other state-of-the-art works, our average accuracy increases at least 0.4m.

From Table 1, we can see that the positioning availability is not 100%, because the images captured from some of the places cannot find a match from the references images. The reason in most of this kind of situations is coming from the quality of the

reference. When the camera of Google is too close facing a wall, there will be less interest features to match with the query image. In addition, if either the query image or the reference image is in a strong sunlight or a deep shadow, the image matching will also be influenced. Also from Table 1, we found that the positioning availability of Route 3 and Route 4 is lower than the positioning availability in other routes. This problem is coming from the blur errors in the virtual view generated from Google Maps Street View, as shown in Figure 3.27. The red boxes show the errors in the generated virtual view images, which lead to the deformation or blur on some of the building walls. This deformation or blur will make it hard to find a correct virtual view.

Table 1 Correct Side Rate and Availability.

	Route 1	Route 2	Route 3	Route 4	Route 5
Correct Side Rate	98.20%	100%	92.39%	91.75%	100%
Positioning Availability	92.00%	86.36%	78.57%	76.92%	90.91%

Table 2 Distance Error of the Proposed Method

	Mean Error(m)	Lateral Error(m)	Longitude Error(m)	Std (m)
Route1	2.893	1.542	2.448	1.227
Route2	3.233	1.259	2.914	1.450
Route3	3.541	1.405	3.169	2.692
Route4	3.550	1.406	3.223	1.191
Route5	1.859	0.763	1.638	0.607

Table 3 The Comparison between Our Method and Others

	Our method	Zamir et al.[22]	Sattler et al.[13]	Deng et al.[23]
Mean error(m)	4.37	12.43	14.9	4.729

Chapter 4.

Augmented Reality based Pedestrian Navigation

Nowadays AR is becoming a very handy tool for various aspects of life such as medical, education, gaming, engineering, and mobile applications that become a very powerful tool if combined with AR. An AR based navigation application can superimpose computer-generated images such as direction guidance, distance information and building information on top of view of reality, thus provide straightforward navigation service for pedestrians. In this chapter, at first the related works about intersection detection and destination recognition in first person view are introduced. Then we describe how to use open source to give reliable AR arrow guide in intersections and how to show destinations in pedestrian navigation system.

4.1. Related works

4.1.1. Smart phone based navigation

Smartphone-based measurement system for road vehicle traffic monitoring and usage-based insurance (UBI) attracts peoples' attentions[36]. Through the aid of a hierarchical model to modularize the description, the functionality is described as spanning from sensor-level functionality and technical specification to the topmost business model. The designer of a complex measurement system has to consider the full picture from low-level sensing, actuating, and wireless data transfer to the topmost level, including enticements for the individual smartphone owners, i.e., the end users who are the actual measurement probes. The measurement system provides two data streams: a primary stream to support road vehicle traffic monitoring and a secondary stream to support the UBI program. The former activity has a clear value for a society and its inhabitants, as it may reduce congestion and environmental impacts. The latter data stream drives the business model and parts of the revenue streams, which ensure the funding of the total measurement system and create value for the end users, the

service provider, and the insurance company. In addition to the presented framework, outcome from a measurement campaign is presented, including road vehicle traffic monitoring (primary data stream) and a commercial pilot of UBI based on the driver profiles (secondary data stream). The measurement system is believed to be sustainable due to the incitements offered to the individual end users, in terms of favorable pricing for the insurance premium. The measurement campaign itself is believed to have an interest in its own right, as it includes smartphone probing of road traffic with a number of probes in the vicinity of the current state of the art, given by the Berkeley Mobile Millennium Project. During the ten-month run of the project, some 4500 driving h/250 000 km of road vehicle traffic data were collected.

4.1.2. Driving navigation

Recently, there has been suggested a device that detects a decrease and the like in the concentration level of a driver who drives a vehicle and that informs the driver of the decrease in the concentration level. For example, in Patent Literature 1, there is disclosed a device that predicts the future occurrence of sleepiness, fatigue, and the like of a driver until the driver reaches a predetermined position on the basis of biometric information such as heart rate, respiratory rate, blink speed, and the like of the driver of a vehicle and on the basis of load exerted on the driver by a road until the vehicle reaches the predetermined position. The device of Patent Literature 1 calls to the attention of the driver in advance on the basis of the sleepiness of the driver at the predetermined position.

In the above technology, however, attention of the driver may be called even when calling attention of the driver and the like are not necessarily required. Thus, the driver may feel inconvenienced.

One embodiment of the present invention is devised with consideration of the problem above, and an object thereof is to provide a driving assistance device, a driving assistance method, an information-providing device, an information-providing method, a navigation device, and a navigation method, in which a driver feels less inconvenienced by performing driving assistance that is more appropriate for a situation where a vehicle is traveling.

4.2. Intersection detection

4.2.1. Objective

Figure 19 Objective of intersection detection

When pedestrians walk along way to their destination, there is high possibilities that they will make a turn at an road intersection. And it is easy to be confused for people who are not familiar with surrounding environment, which is the scenario we want to solve. So if we can detect there is an intersection and know the direction of links for the intersection in front of a pedestrian, it will help us simply the navigation problem. When the route is given, our objective is to present intuitive arrow guidance in the gap of the intersection in the first person view, as Figure 4.1 shows.

4.2.2. Generating point clouds with Google Street View

The overall generation process can be divided into three parts: composing a panorama, computing the depth map and creating the point cloud. Firstly we have to deal with retrieving the panorama image closest to the specified initial position. By making a call to Google Maps REST API at the following address:
https://maps.google.com/cbk?output=json&hl=x-local&ll=LAT,LNG&cb_client=maps_sv&v=3

it is possible to obtain the unique identifier of the panorama. Google Street View Service Javascript API allows us to retrieve some information to be used in the next

step: the panorama identifier, the available resolutions for the whole panorama, the resolution of the single tiles composing it, the world heading, the real coordinates and eventual neighboring panoramas. Despite Google Street View may provide resolutions up to 13312x6656 pixels, we considered the resolution of 3328x1664 pixels more than sufficient for our purposes. In this particular case, the objective is to compose in a single image 7x4 tiles having a resolution of 512x512 pixels, since Google doesn't allow to download directly the panoramic image. So, we need to download each of the 28 tiles by using the REST API:

https://cbks2.google.com/cbk?cb_client=maps_sv.tactile&authuser=0&hl=en&panoid=PANORAMAID&output=tile&zoom=QUALITY&x=XPOS&y=YPOS&TIMESTAMP

where in our case QUALITY is 3 and XPOS and YPOS correspond respectively to the position of the desired square tile in the 7x4 image grid. After having combined all of them in a single image by partially overlapping their borders, we obtain a 3328x1664 RGB image corresponding to the desired panorama.

Now that we have the panorama image, we need to retrieve its corresponding depth map. Google Maps REST API allows us to download a compressed JSON representation of the depth image from the url:

http://maps.google.com/cbk?output=json&cb_client=maps_sv&v=4&dm=1&pm=1&ph=1&hl=en&panoid=PANORAMAID

which contains the distance from the camera to the nearest surface at each pixel of the panorama. After having decoded from Base64 the data and having converted it to an array of unsigned 8-bit integers, we can fetch its header information obtaining useful values, like the number of referenced planes. In fact, each pixel in a grid of 512x256 pixels is corresponding to one of several planes, which are given by its normal vector and its distance to the camera. Therefore, in order to calculate the depth at a pixel, we have to determine the intersection point of a ray starting at the center of the camera and the plane corresponding to the pixel. Iterating for all the planes, we can then populate our depth map as 32-bit float array of 512x256 elements - which is much lower than the resolution of our RGB panorama image. As for the computation, for each point we consider its associated plane and we compute its distance as

Algorithm2: Depth map computation

For all: indeces *x, y* **do**

planeIndex ← indices[y *width+x]

$\varphi \leftarrow \frac{w-x-1}{w-1} * 2\pi + \frac{\pi}{2}$

$\theta \leftarrow \frac{h-y-1}{h-1} * \pi$

$v \leftarrow [sin\theta\ cos\varphi,\ sin\theta\ sin\varphi,\ cos\varphi]$

If planeIndex > 0 **then**

plane ← planes[planeIndex]

$t = \frac{plane.d}{v * plane.n}$

$v \leftarrow v * t$

$depth[y * w + (w - x - 1)] \leftarrow \sqrt{v * v}$

else

$depth[y * w + (w - x - 1)] \leftarrow \infty$

end

end

where indices is an array containing the plane associated to each pixel.

Now that we have the depth information for each pixel, we need to create our point cloud and map each point to its original color in the panorama obtained at step 1. Considering n_{points} = ***w*h*** points, we define two n_{points} *3 float arrays containing the 3D space position and the color of each point. Then we have to consider that the points of the panorama image originally belonged to a spherical image, so we have to reproject them in space by using the following algorithm:

Algorithm3: Point cloud generation

For $y_{depth} \leftarrow 0$ ***to*** h_{depth} **do**

$$lat \leftarrow \frac{y_{depth}}{h_{depth}} - 90$$

$$r \leftarrow \cos\frac{lat * \pi}{180}$$

For $x_{depth} \leftarrow 0$ **to** w_{depth} **do**

$$depth \leftarrow depthMap\,[\, y_{depth} * w_{depth} + (w_{depth} - x_{depth})\,]$$

$$\ln g \leftarrow (1 - \frac{x_{depth}}{w_{depth}}) * 360 - 180$$

$$x_{pos} \leftarrow -r * \cos\frac{\ln g * \pi}{180}$$

$$y_{pos} \leftarrow \sin\frac{lat * \pi}{180}$$

$$z_{pos} \leftarrow r * \sin\frac{\ln g * \pi}{180}$$

$$pos_{point} \leftarrow [x_{pos}, y_{pos}, z_{pos}] * depth$$

$$x_{norm} \leftarrow = \frac{1 - x}{w}$$

$$y_{norm} \leftarrow = \frac{y}{h}$$

$$x_{color} \leftarrow \text{int}(x_{norm} * w_{color})$$

$$y_{color} \leftarrow \text{int}(y_{norm} * w_{color})$$

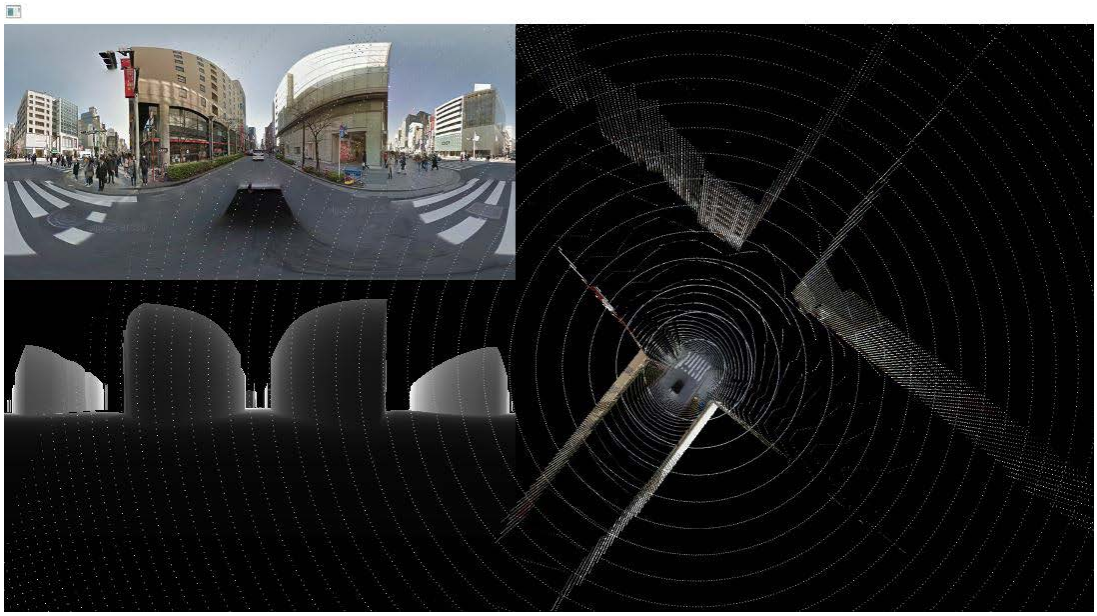
$$color_{index} \leftarrow y_{color} * w_{color} * 4 + x_{color} * 4$$

$$color_{point} \leftarrow image_{color}[color_{index}] / 255$$

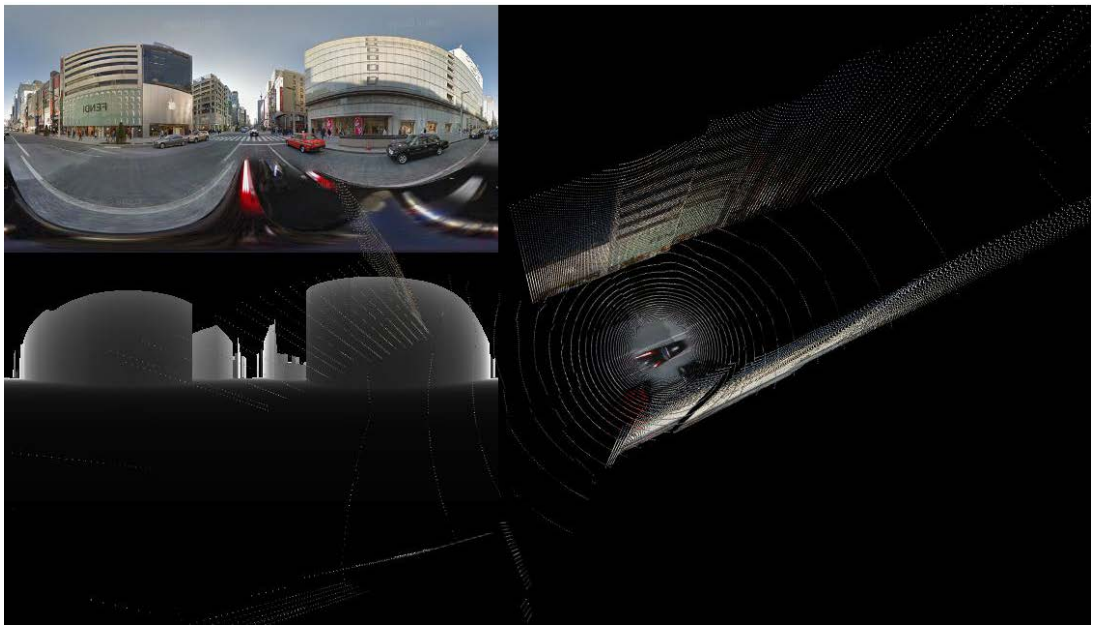
end

end

Notice that, in addition to the reprojection, it was necessary to normalize the pixels positions in 2D in order to retrieve colors from the panoramic colored image, which has a different resolution.



(a)



(b)

Figure 20 Local 3D point cloud for an intersection in Ginza

We can see the generated point cloud in Figure 43, both (a) and (b) are point clouds for the same intersection, but center of the depth map is different. We can see the point cloud is well-formed, the links and gaps are clear and easy to distinguish.

4.2.3. Segment intersection from point cloud using OpenStreetMap

Since Google Maps API do not provide the 2D intersection geographic

information, we use OpenStreetMap to obtain the shape and coordinates information of intersection. We can extract the shape of intersection from OSM as Figure 44 shows. We can segment the intersection in point cloud using the shape and coordinates provided by OSM, Figure 45 shows the result after segmentation.

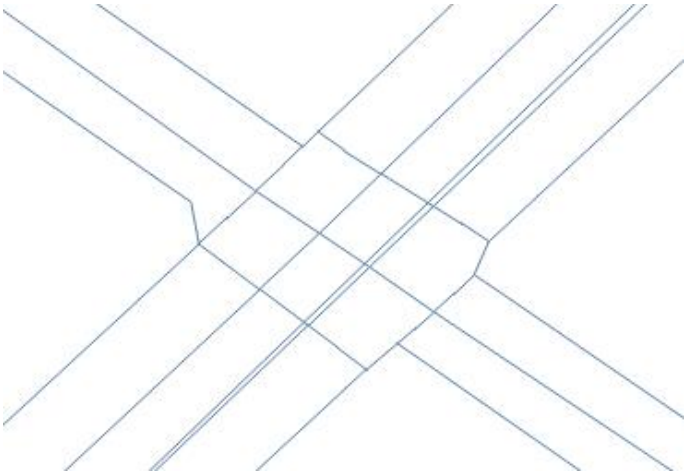


Figure 21. An example of intersection shape from OSM.

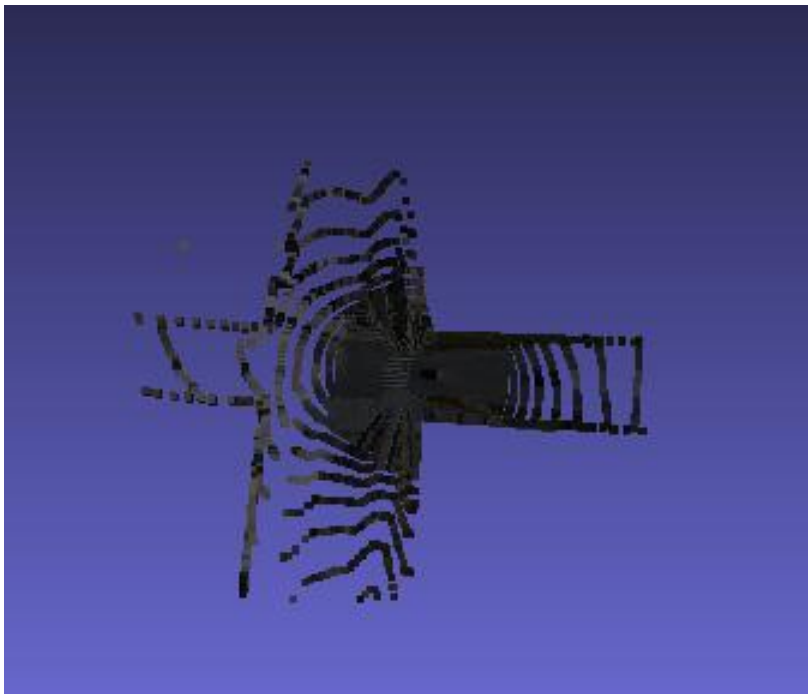
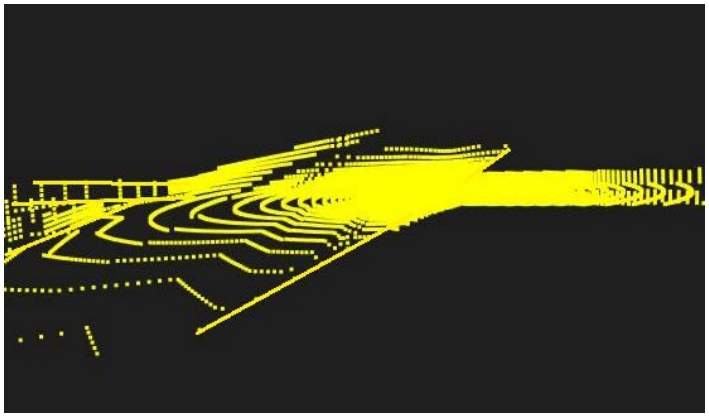


Figure 22 Intersection segmented from local point cloud by 2D map

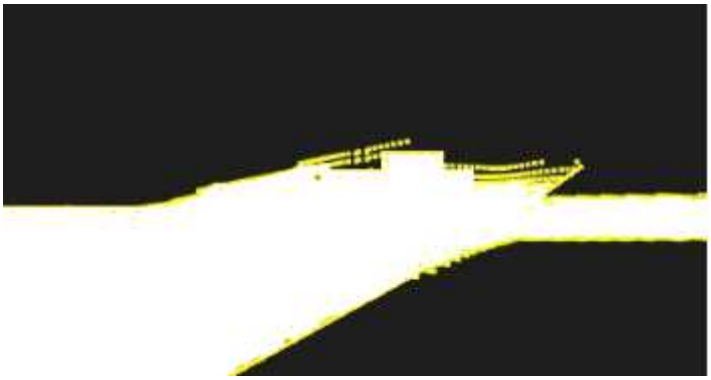
4.2.4. Visualization for intersection detection

As we already know the camera pose and pedestrian position, we can get the camera view of the intersection in 3D local point cloud, like Figure 46 (a) shows. Then we can extract the contour as (b) shows. And finally, we overly the contour of

intersection in first person view as (c) shows.



(a)



(b)

(c)

Figure 23 Process of intersection detection visualization in first person view.

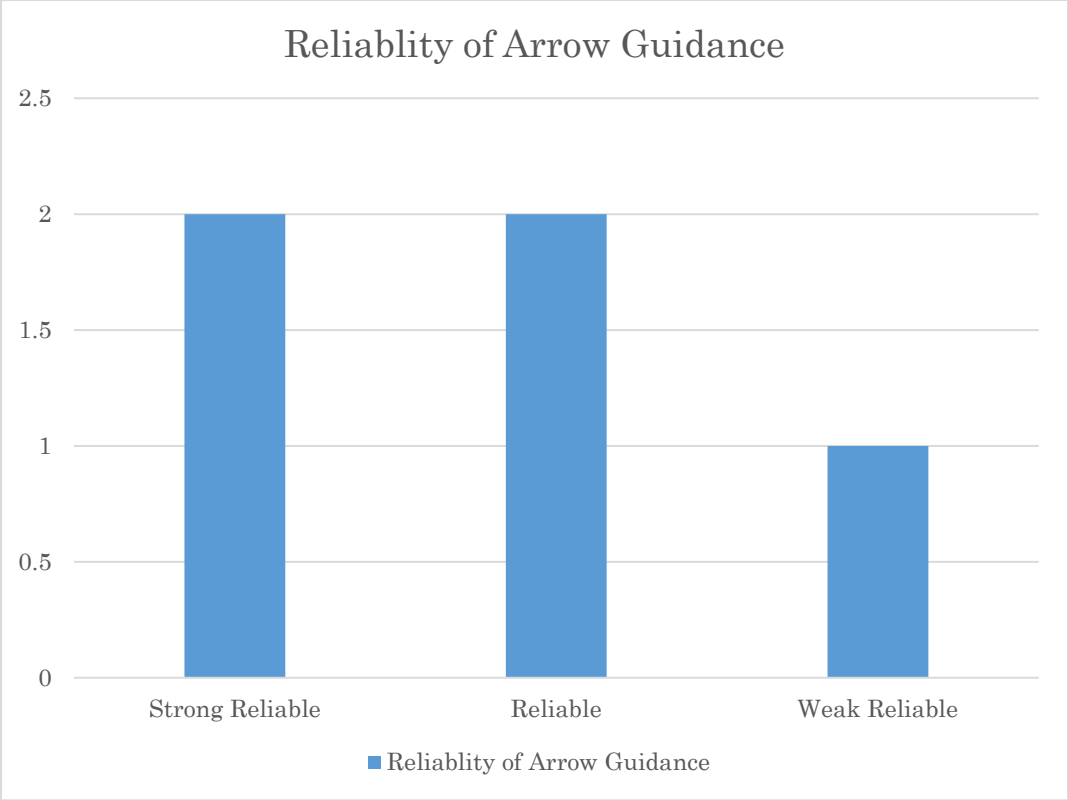
4.2.5. Results of intersection detection

We compose the guidance arrow in the gap in an intuitive way, the results are show in Figure 47. When the pedestrian’s heading angle is not in the desired direction, the system will show an alarm arrow icon toward the right direction, as shown in Figure 48(d), (e), the icon is placed in the center of opposite gap when the desired gap is not

in the view.

Figure 24 Results of intersection detection

To evaluate our results, we ask 5 volunteers to experience the navigation process. Two of them think the arrow guidance is easy to understand, two of them think for more than half of total cases they can follow the guidance, and one volunteer gave feedback that less than half of total cases the guidance is reliable.



4.3. Destination recognition

4.3.1. Objective

When pedestrian moves towards the destination, he or she may do not know what the destination building looks like. This fact motivate us to visualize segmentation of destination in the first person view. When the pedestrian can see the destination through smart glasses, we want to segment it using semi-transparent colors, as Figure shows.

Figure 25 Goal of destination recognition[35]

4.3.2. Proposed method

As the same approach in intersection detection part, we also employ point cloud and OpenStreetMap to visualize destination segmentation in first person view.

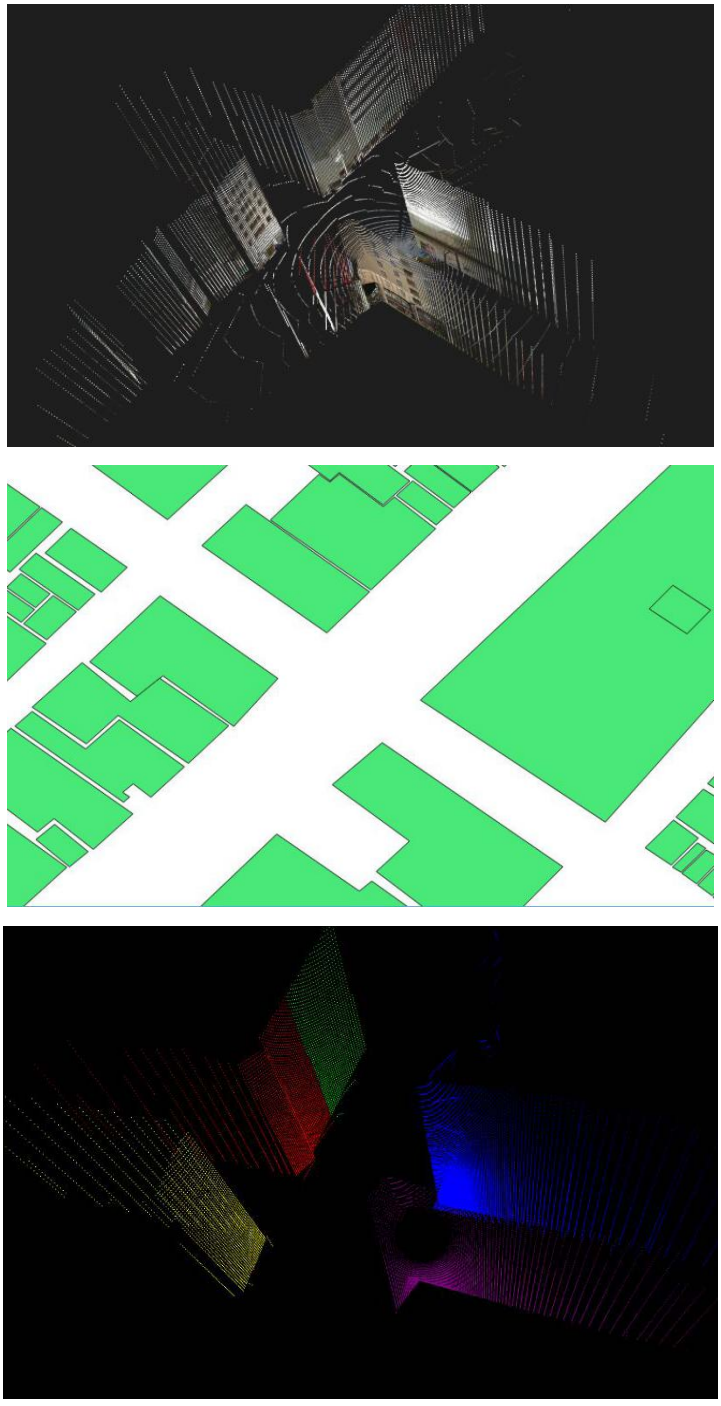
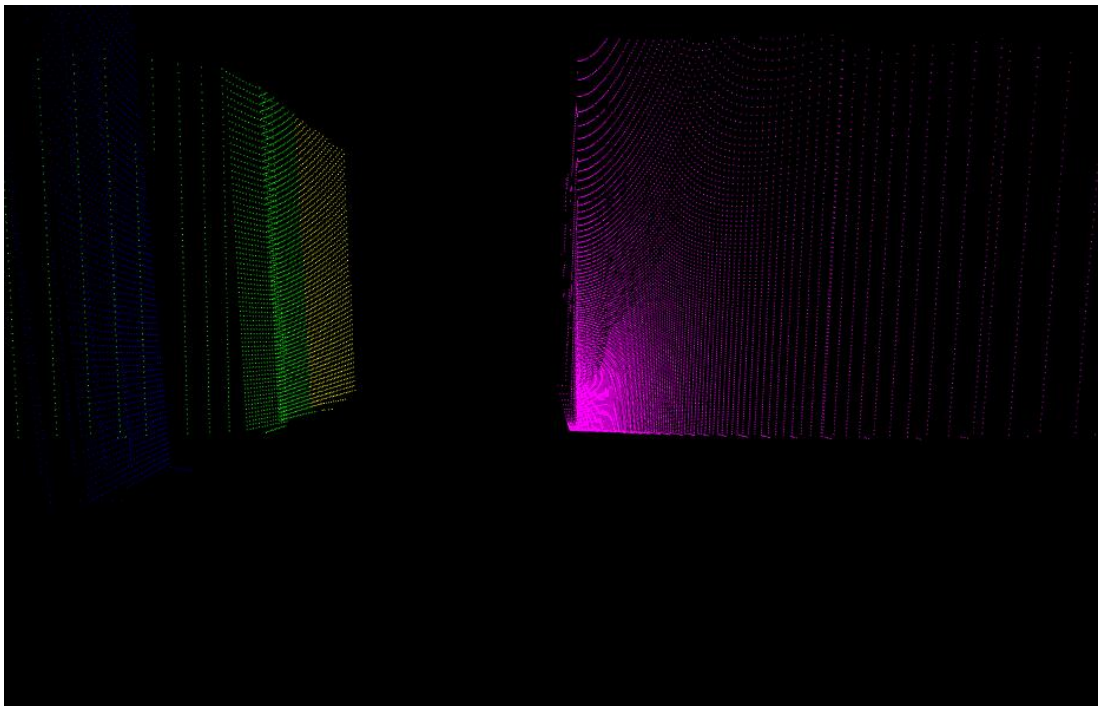


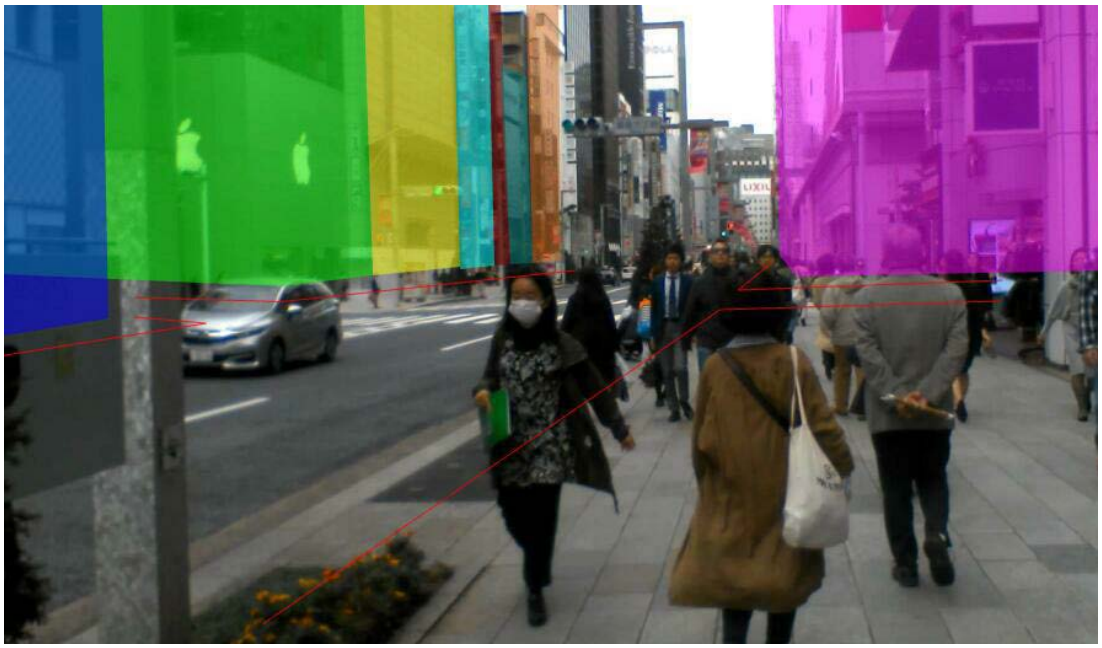
Figure 26 Process of building segmentation in local point cloud using OSM

Figure 49 shows the process of building segmentation in local point cloud. First

of all, we can generate point cloud using panorama and its corresponding depth data provided by Google Maps API, then we can obtain the shapefile of buildings in the same area from OSM, we can see the segmented results in the last image.



(a)



(b)

(c)

Figure 27 Visualization of destination recognition in first person view

Figure 50 shows the visualization of destination recognition in first person view, as we know the position of camera and camera pose, we can get the camera view in the segmented point cloud as (a), so that we can project the 3D coordinates of building into the 2D image, as shown in (b). Also we can list all the building names and their corresponding color in the view as (c) shows. Note that the interface does not support Japanese, so some names of buildings cannot be shown.

4.3.3. Experiment results

Figure 28 Experiment results for destination recognition

Figure 51 shows the visualized destination recognition results. We employ the percentage of overlay area between our results to evaluate our method. We use the OSM building shapefile as our groundtruth, as Figure 52 shows. If the overlaid percentage is higher than 80%, we take it as a successful result; if lower than 80%, we take it as a unsuccessful result. The results can be seen in table 4.

Table 4 Success Rate for Destination Recognition

	Route1	Route2	Route3	Route4	Route5
Success rate	73.35%	69.45%	55.26%	58.17%	65.97%

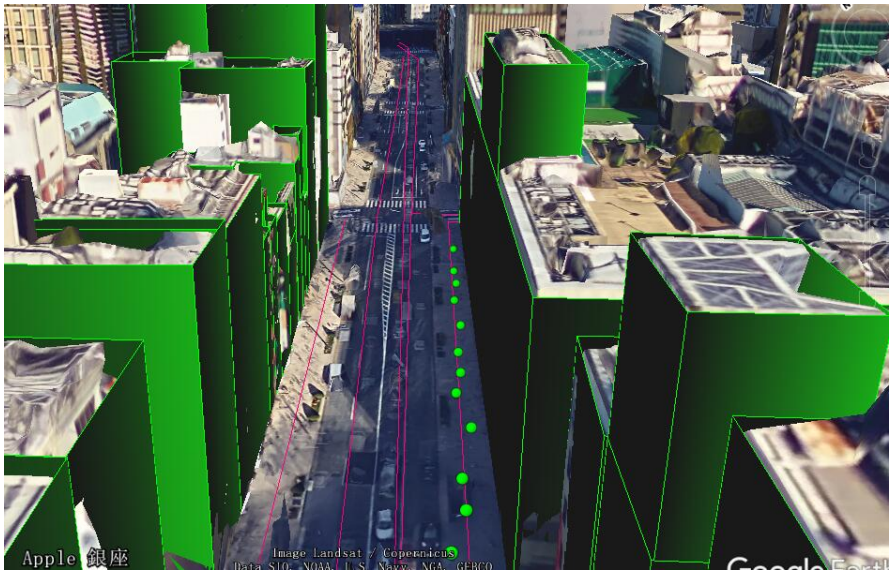


Figure 29 Groundtruth for destination recognition.

Chapter 5.

Conclusions

In this thesis, we present a study on pedestrian navigation system with image-based positioning method and the aid of Google Maps Street View in urban canyon environment.

This observation is compared with the available virtual views and interested area datasets from Google Maps Street View in order to correct positioning errors.

At first we use vanishing point to estimate the heading angle of pedestrian query image.

Then we build a building facade images dataset from Google Street View Image API and the building facade images dataset is used to search the interested area with the geo-tagged query image.

Shape and size of the matched interested building wall is used to suggest us which side of road and the distance we should move to estimate the lateral position of the query image.

Finally, the translation matrix decomposed from Homography matrix will suggest us which direction on the road and the distance we should move to find a virtual view with minimum e as the position of the positioning result of the query image.

With the visual matching between the geo-tagged pedestrian's photo and the reference virtual views from Google Maps Street View, we can improves the correct side rate to 90% and achieves 4-meter positioning performance.

From the positioning result, we get the conclusion that the positioning accuracy in our proposed method rely on the quality of the 3D model. When the 3D model or the texture is not correct, it will make the input query image from the pedestrian unable to get an accurate positioning result or even cannot find a matched virtual, then the positioning will be lost. Also the 3D model should contain the textures in different kinds situations such as both daytime and night view.

After getting a reliable positioning result, to provide a more intuitive navigation service to the pedestrians, we considered to introduce the Augmented Reality (AR) into

the system

We divide objective into two kinds of situations: pedestrian is far from next checkpoint and pedestrian is near to the checkpoint.

When pedestrian is far from next checkpoint, in order to provide a realistic and intuitive AR navigation, we use vanishing point to estimate the directions of roads in the pedestrian's view and use the image matching of interested area to recognize where we need to turn. Then the guide arrow can be drown on the scene.

When pedestrian is near to the checkpoint, we directly project the guidance route in the matched virtual view to the pedestrian view.

The research should continue to focusing on how to make a more correct and natural arrow and guidance.

Reference

[1] L.-T. Hsu, Y. Gu, Y. Huang, and S. Kamijo, Urban pedestrian navigation using smartphone-based dead reckoning and 3D maps aided GNSS, IEEE Sensors Journal, 16(5), 1281-1293, 2016.

[2] .S. Miura, L.-T. Hsu, F. Chen, and S. Kamijo, "GPS Error Correction With Pseudorange Evaluation Using Three-Dimensional Maps," IEEE Transactions on Intelligent Transportation Systems, vol. PP, no. 99, pp. 1-12, 2015.

[3] N. Kakiuchi, K. Sunagawa, and S. Kamijo, Pedestrian dead reckoning for mobile phones using magnetic deviation map, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E98-A(1), 313-322, 2015.

[4] "http://www.gps.gov/"

[5] Y. Huang, L.-T. Hsu, Y. Gu, H. Wang, and S. Kamijo, Assessment of outdoor Wi-Fi fingerprint calibration using different GNSS approaches, In International Symposium on GNSS 2015, 2015

[6] E. D. Kaplan and C. J. Hegarty, Understanding GPS: principles and applications. Artech house, 2005.

[7] N. Kakiuchi and S. Kamijo, "Pedestrian dead reckoning for mobile phones through walking and running mode recognition," in 2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC), pp. 261{267, Oct. 2013.

[8] Leppäkoski, H., J. Collin, and J. Takala. "Pedestrian navigation based on inertial sensors, indoor map, and WLAN signals." Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE, 2012.

[9] Deng, Zhi-An, et al. "Heading estimation for indoor pedestrian navigation using a smartphone in the pocket." Sensors 15.9 (2015): 21518-21536.

[10] Steinhoff, Ulrich, and Bernt Schiele. "Dead reckoning from the pocket-An experimental study." Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on. IEEE, 2010.

[11] Chen, Zhenghua, et al. "Fusion of WiFi, smartphone sensors and landmarks using the Kalman filter for indoor localiza-tion." Sensors 15.1 (2015): 715-732.

[12] Zhang, Peng, et al. "Collaborative WiFi fingerprinting using sensor-based navigation on smartphones." Sensors 15.7 (2015): 17534-17557.

[13] Zhuang, Yuan, et al. "PDR/INS/WiFi integration based on handheld devices for

indoor pedestrian navigation." Micromachines 6.6 (2015): 793-812.

[14]R. E. Kalman, "A new approach to linear filtering and prediction problems," Journal of Fluids Engineering, vol. 82, no. 1, pp. 35-45, 1960.

[15]" http://www.cse.psu.edu/~rtc12/CSE486/ ".

[16]H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping (SLAM): part I The Essential Algorithms,” Robotics & Automation Magazine, vol. 2, pp. 99–110, 2006.

[17] Garcia-Fidalgo, Emilio, and Alberto Ortiz. "Vision-based topological mapping and localization methods: A survey." Robotics and Autonomous Systems 64 (2015): 1-20.

[18] Yu, Guoshen, and Jean-Michel Morel. "ASIFT: An algorithm for fully affine invariant comparison." Image Processing On Line 1 (2011): 11-38.

[19] Z. Liu and R. Marlet, “Virtual line descriptor and semi-local matching method for reliable feature correspondence,” in British Machine Vision Conference 2012, 2012, pp. 16‑1.

[20] A. L. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza. MAV urban localization from Google street view data. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3979–3986, 2013.

[21] A. L. Majdik, D. Verda, Y. Albers-Schoenberg, and D. Scaramuzza. Micro air vehicle localization and position tracking from textured 3d cadastral models. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 920–927, 2014.

[22] A. Torii, J. Sivic, and T. Pajdla. Visual localization by linear combination of image descriptors. In Computer Vision Workshops (ICCV Workshops), 2011.

[23] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pages 2599–2606, 2009.

[24] A. R. Zamir and M. Shah. Accurate image localization based on google maps street view. In Proceedings of the European Conference on Computer Vision (ECCV), pages 255–268, 2010.

[25] Agarwal P, Burgard W, Spinello L. Metric localization using google street view[C]//Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International

Conference on. IEEE, 2015: 3111-3118.

[26]Robertson, Duncan P., and Roberto Cipolla. "An Image-Based System for Urban Navigation." BMVC. 2004.

[27]Jin Kim, Hyo, Enrique Dunn, and Jan-Michael Frahm. "Predicting good features for image geo-localization using per-bundle vlad." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[28] Knopp, Jan, Josef Sivic, and Tomas Pajdla. "Avoiding confusing features in place recognition." European Conference on Computer Vision. Springer Berlin Heidelberg, 2010.

[29] Sattler, Torsten, Bastian Leibe, and Leif Kobbelt. "Fast image-based localization using direct 2d-to-3d matching." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[30] Google Street View Image API, "https://developers.google.com/maps/documentation/streetview/".

[31]M. Andriluka, S. Roth, and B. Schiele, “Monocular 3D pose estimation and tracking by detection,” in 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 623–630.

[32] Kessler, Christoph, et al. "Vision-based attitude estimation for indoor navigation using vanishing points and lines." Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION. IEEE, 2010.

[33] Duan, Wenting, and Nigel M. Allinson. "Vanishing points detection and line grouping for complex building facade identification." (2010).

[34] Google Directions API, "https://developers.google.com/maps/documentation/directions/intro"

[35] Li, Y.; Hu, Q.; Wu, M.; Liu, J.; Wu, X. Extraction and Simplification of Building Façade Pieces from Mobile Laser Scanner Point Clouds for 3D Street View Services. ISPRS Int. J. Geo-Inf. 2016, 5, 231.

[36] Branislav Micusik, Jana Kosecka, Piecewise Planar City 3D Modeling from Street View Panoramic Sequences, IEEE Conference on Computer Vision and Pattern Recognition, 2009

[37] Jay Bolter, Maria Engberg, Blair MacIntyre Media Studies, Mobile Augmented Reality, and Interaction Design, ACM Interactions, February 2013

[38]Zeljko Medenica, Andrew L. Kun, Tim Paek, Oskar Palinko Augmented Reality

vs. Street Views: A Driving Simulator Study Comparing Two Emerging Navigation Aids, MobileHCI ’ 11 Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, 2011

[39]. Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stephane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, Josh Weaver, Google Street View: Capturing the World at Street Level, Computer, vol. 42, 2010

[40] Kotaro Hara, Victoria Le, Jon E. Froehlich, Combining Crowdsourcing and Google Street View to Identify Street-level Accessibility Problems, CHI ’ 13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013

[41] Stefaan Ternier, Roland Klemke, Marco Kalz, Patricia, Ulzen, Marcus Specht, ARLearn: Augmented Reality Meets Augmented Virtuality, J-jucs journal vol. 18, August 2012

[42]Mark Graham, Matthew Zook, Andrew Boulton, Augmented reality in urban places: contested content and the duplicity of code, Transactions of the Institute of British Geographers, vol. 38, July 2013

[43]R. Diaconu, J. Keller, E. Triponez, HybridEarth: Social Mixed Reality at Planet Scale, Consumer Communications and Networking Conference (CCNC), 2014 IEEE

[44] Käshammer, P.; Nüchter, A. Mirror identification and correction of 3D point clouds. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2015, 40, 109.

[45] Rodríguez-Cuenca, B.; García-Cortés, S.; Ordóñez, C.; Alonso, M.C. Morphological operations to extract urban curbs in 3D MLS point clouds. ISPRS Int. J. Geo-Inf. 2016, 5, 93.

[46]IPA Room Segmentation. Available online: http://wiki.ros.org/ipa_room_segmentation (accessed on 12 February 2017).

[47]I. Lipschutz E. Gershikov B. Milgrom "New Methods for Horizon Line Detection in Infrared and Visible Sea Images", International Journal of Computational Engineering Research, vol. 3 no. 3 pp. 226-233 2002.

[48] Cho SW SS Huh HC Shim HS Choi "An Image Processing Algorithm for Detection and Tracking of Aerial Vehicles in Short-Range" Journal of the Korean Society for Aeronautical and Space Sciences vol. 39 pp. 1115-1123

2011.

[49] H Wang Z Wei S Wang Kah Chek SO KT Ho B. Feng L Zhou "Real-time obstable detection for unmanned surface vehicle", Defense Science Research Conference and Expo, pp. 21-24 2011.

[50] Dünser, A., Billinghurst, M., Wen, J., Lehtinen, V., Nurminen, A.: Exploring the use of Handheld AR for Outdoor Navigation. Computers & Graphics 36, 1084‑1095 (2012)

[51]Liarokapis, F., Mountain, D., Papakonstantinou, S., Brujic-okretic, V., Raper, J.: Mixed Reality For Exploring Urban Environments. In: 1st International Conference on Computer Graphics Theory and Applications (2006)

[52]May, A.J., Ross, T., Bayer, S.H., Tarkiainen, M.J.: Pedestrian navigation aids: information requirements and design implications. Personal and Ubiquitous Computing 7(6), 331‑338 (2003)

[53]Millonig, A., Schechtner, K.: Developing Landmark-Based Pedestrian-Navigation Systems. IEEE Transactions on Intelligent Transportation Systems 8(1), 43‑49 (2007)

[54]Mulloni, A.: Enhancing Handheld Navigation Systems with Augmented Reality. In: Mobile HCI, pp. 5‑8 (2011)

[55]Rehrl, K., Häusler, E., Steinmann, R., Leitinger, S., Bell, D., Weber, M.: Pedestrian Navigation with Augmented Reality, Voice and Digital Map: Results from a Field Study assessing Performance and User Experience. In: Proceedings of the 8th International Symposium on Location-based Services, pp. 3‑20 (2011)

[56]Willis, K.S., Hölscher, C., Wilbertz, G., Li, C.: A comparison of spatial knowledge acquisition with maps and mobile maps. Computers Environment and Urban Systems 33(2), 100‑110 (2009),

[57] Wen, J., Helton, W.S., Billinghurst, M.: Classifying Users of Mobile Pedestrian Navigation Tools. In: Oz CHI, pp. 1‑5 (2013)

[58]V Paul J. Michael "Rapid object detection using a boosted cascade of simple features", Proceedings of IEEE conference on computer vision and pattern recognition, vol. 1 pp. 511-518 2001.

Thanks

This thesis is under the instruction of Professor Kamijo. He gives me the direction and advices not only on the researches but also on my life. Ms. Miwa and Ms. Kosaka, Professor Kamijo's ex-secretary and secretary, also provided huge help to my research life in the lab. Also researcher Gu are the people who gives me a lot of important advice on my researches. Nothing in this thesis can be achieved without their help. To the other members in the lab, Liu, Bao, Mahdi, Ehsan, Li, Wang, Kitamura, I want to thank them for accompanying me through the 2 years and make all these time more meaningful time. Lastly, I want to thank my family and friends for their supporting during this two years.

Publication List

研究会発表:

[1] Ya Wang, Haitao Wang, Yanlei Gu, and Shunsuke Kamijo, "Vision-based Pedestrian Positioning and Navigation in Urban Area Using Smart Glasses", in Proceedings of Pattern Recognition and Media Understanding (PRMU), Tokyo, Japan, December 16-17, 2017