修 士 論 文

Pedestrian Navigation in Urban Area Using Wearable Camera Combined with Digital Map and Street View

（ディジタル地図および Street View
とウェアラブルカメラを連携した都市部における
ナビゲーションに関する研究）

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 48-156454　**王 海涛**

© Copyright 2016, WANG HAITAO.

All rights reserved.

Abstract

Pedestrian navigation has become one of the most used services in people's city lives. Not only smartphone based navigation, but also the application in the next generation of intelligent wearable devices, such as smart glasses, attract attentions from both scientists and engineers. The satisfied navigation service requires an accurate positioning technology. Even though the current smartphones and smart glasses have integrated various sensors, such as Global Positioning System receiver, gyroscope, accelerometer, magnetometer sensors and camera, the performance of positioning in city urban is still not satisfied. The reasons of the errors include GPS signals reflections, high dynamic of pedestrian activities and disturbance of the magnetic field in city environments. This paper proposes to utilize the camera sensor for improving the accuracy of the positioning. The camera sensor provides the visual observation for surround environment.

This observation is compared with the available Google Maps Street View in order to correct positioning errors. With the visual matching between the geo-tagged pedestrian's photo and the reference images from Google Maps Street View, we expect to reduce the positioning error into 4 meters, and further recognize which side of the road the pedestrian is in. Moreover, based on our positioning result, Google Directions API combined with digital map information are used to provide the Augmented Reality navigation to guide the user to the destination. Experiments are conducted at Ginza area of Tokyo to prove effectiveness of the proposed methods.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

In this research, we studied how to improve the positioning accuracy of pedestrian in the urban area with the aid of Google Maps Street View and how to provide a pedestrian navigation system with Augmented Reality. In this chapter, the background, objective and structure of this thesis are discussed.

1.1. Background

Nowadays, the pedestrian navigation application becomes one of the most used and convenient service in people's city lives. The quality of navigation service significantly depends on the accuracy of pedestrian localization.

In recent years, most of people have been equipped with smartphone and it has become an important part of citizen's daily life. Not only the smartphone, more and more kinds of next generation of intelligent wearable devices have coming out, such as smart glasses, smart watch and smart bracelet. To support different kinds of application software, these wearable intelligent devices have integrated various sensors, such as Global Positioning System receiver, wireless receiver, gyroscope, accelerometer, and magnetometer. And the smartphone and smart glasses even have been equipped with camera. 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 smartphone 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].

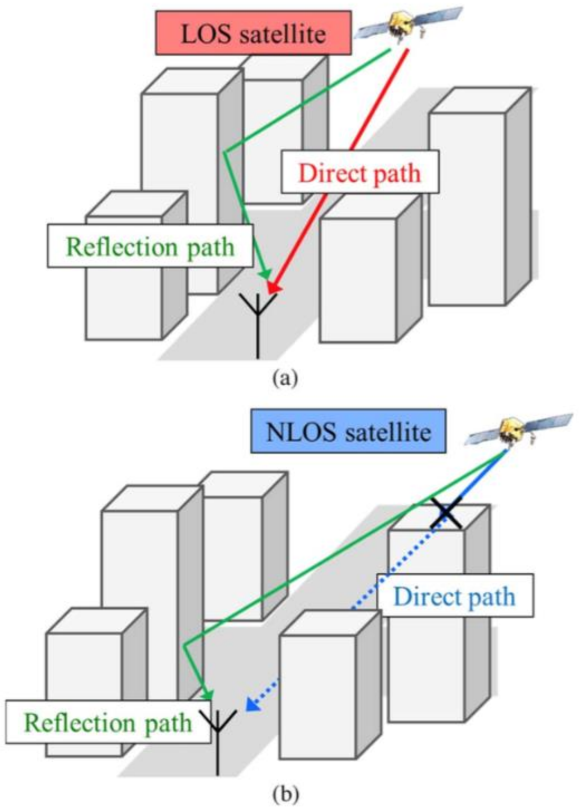


Figure 1.1 The multipath and NLOS effects in an urban canyon. (a) Multipath effect. (b) NLOS propagation.

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.

Figure 1.2 shows a typical conventional GPS positioning result (yellow line) using commercial level GPS receiver. We make this test by taking the Google Nexus 5 smartphone, in which there have been a commercial level GPS receiver, and walking alone the ground truth route, which is shown in the blue line. Because of the NLOS and Multipath effect, the positioning accuracy is around 26 meters. As can be observed, we cannot distinguish which side of road we were with the GPS positioning result in most of cases.

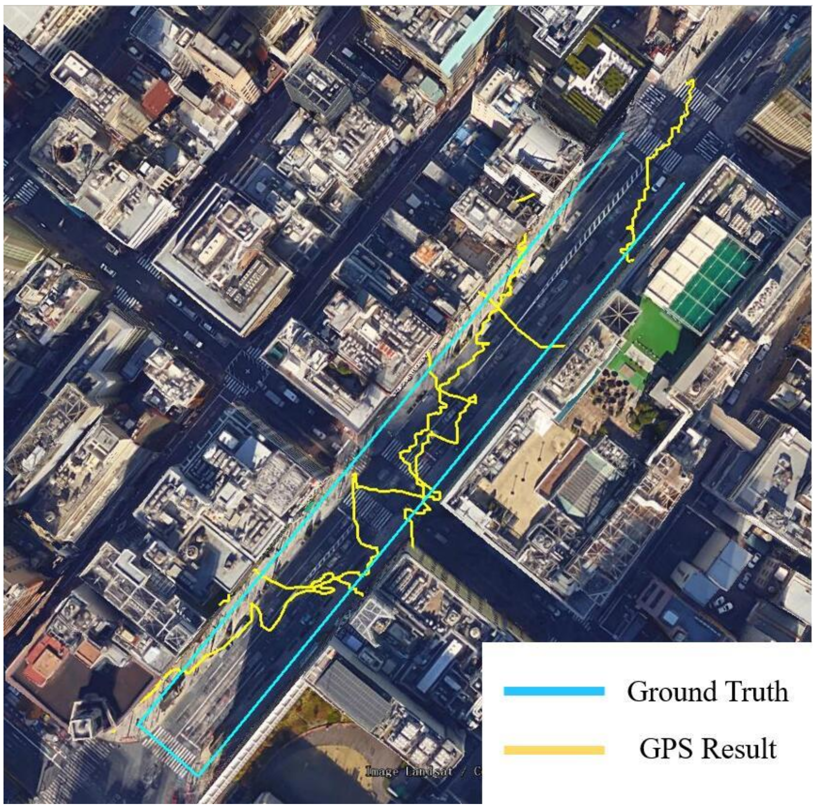


Figure 1.2 Conventional GPS result in urban area.

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.

Considering these several limiting problems of GPS, Wi-Fi and PDR when they works standalone, integration of different positioning systems becomes an effective solution. The integration of Wi-Fi-based position and PDR is usually used in indoor environment and the integration of GPS and PDR is popular in outdoor environment.

Considering the smartphone and smart glasses have been equipped with camera, therefore, beside these positioning method, image information also can be used to recognize the environment and provide positioning result.

In this paper, 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 virtual views from Google Maps Street View.

Google Maps Street View is a comprehensive and large database provided by Google, which consists of geo-tagged 360^{o} 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 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. After deciding the heading angle of the pedestrian's camera utilizing the vanishing point detection in the image, we can use the difference of shapes of a same interested wall in both the pedestrian image and the generated virtual view to estimate which side of road the pedestrian is in. Then the position of the interested area can help us to move the virtual view to the position near the pedestrian's position. Once an almost same virtual view of the pedestrian can be found, the pedestrian positioning result can be decided from the position where the virtual view was generated.

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. AR is attracting more and more attentions in recent years, especially after the coming out of the new generation of eyeglass-type intelligent wearable devices. AR can be used in many kinds of fields, such as education, art, game, and also navigation. The key of an AR system is how to combine the amplification objects with the actual environment. The position and heading of the camera are required to convert the scene from the real-world coordinate to the camera coordinate and superimpose the amplification objects into the scene. Many researches on AR navigation rely on the position and heading information of the camera getting from GNSS, IMU sensors and magnetometer. However, as mentioned previously, the performance of these devices in the urban area are not accurate. Therefore, in order to provide a realistic AR navigation, we directly use the image information from the camera, including the vanishing point to decide the angle of the road in the pedestrian's view, and the interested area matching to find where to turn.

1.2. Objective

In this research, we studied the pedestrian navigation system in a typical urban canyon environment. We tried to improve the positioning accuracy with the aid of Google Maps Street View. We expect to reduce the positioning error into 4 meters and further recognize which side of the road or which corner of the crossroads the pedestrian is in. We also want to provide a reliable AR navigation guide. There are mainly two works be done in this thesis.

1. Firstly, we proposes to utilize the camera sensor for improving the accuracy of the positioning. The camera sensor provides the visual observation for surround 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. 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.

2. 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. Figure 1.3 shows the sketch map of the AR navigation interface.

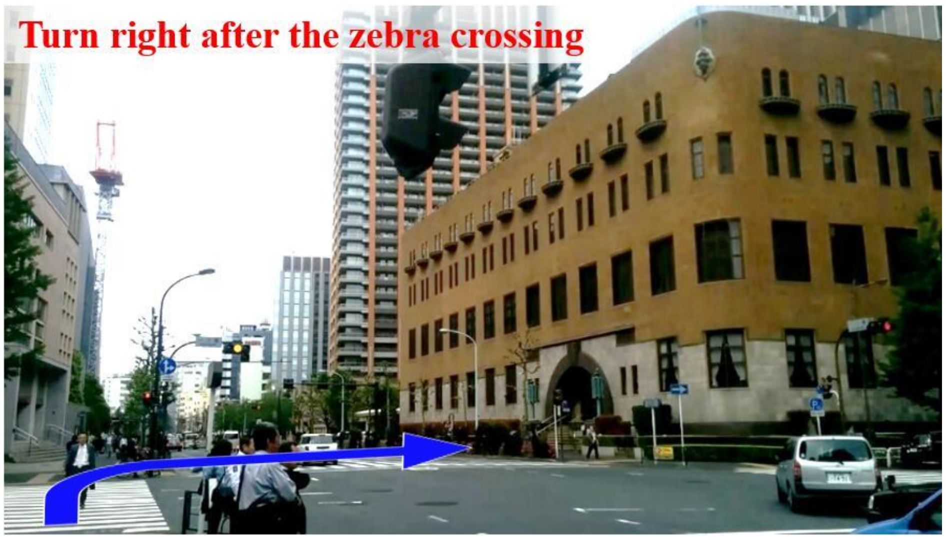


Figure 1.3 The sketch map of the AR navigation system interface

1.3. Thesis Structure

The Structure of this paper listed as follows:

In Chapter 2, the basic concepts of GPS positioning, PDR-based positioning, Wi-Fi-based positioning and the integrating system are introduced.

In Chapter 3, at first we introduce the related works about the vision-based positioning and the Google Maps Street View pipeline is introduced as an important source. Then we introduce our propose method about how to estimate pedestrian position with the aid of Google Maps Street View in the urban area.

In Chapter 4, how to provide reliable AR navigation guide in pedestrian navigation system is talked.

In Chapter 5, the conclusions are presented.

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].

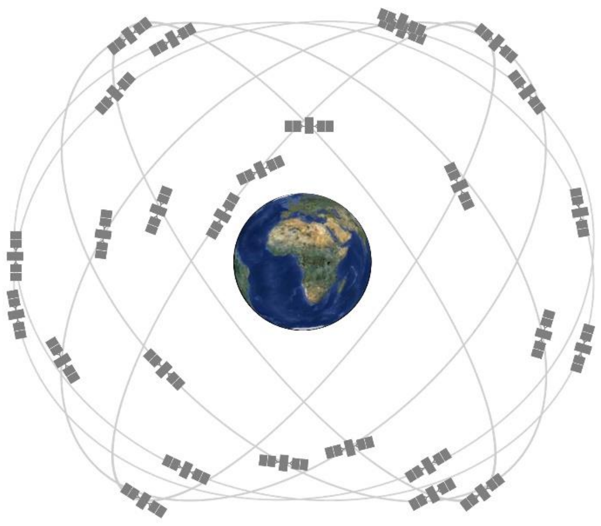


Figure 2.1 The constellation of GPS satellites

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. The fundamental principle of GPS

GPS is composed of the space segments include satellites and satellite launchers, the control segments include several monitor stations around the world and user segments. To positioning the users, GPS fallow the law of trilateration as shown in Figure 2.2 [6].

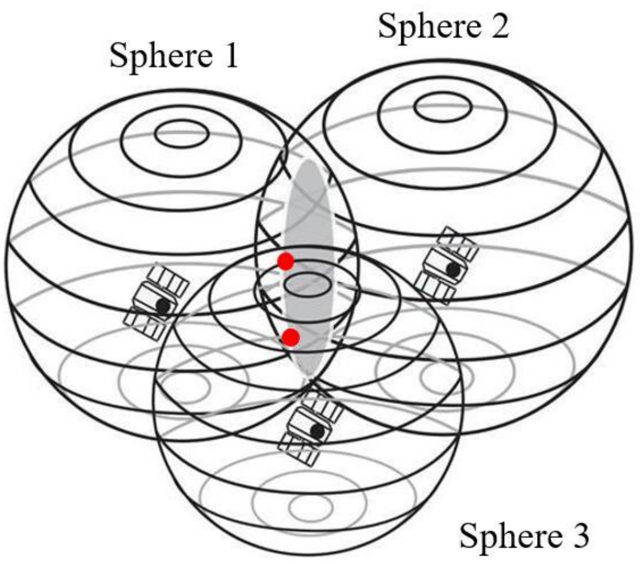


Figure 2.2 GPS trilateration positioning.

When a GPS receiver can receive the signals from three GPS satellites, if we can know the coordinates of these satellites, and the distances form each satellites to the receiver, then we can draw three spheres as shown in Figure 2.2. The intersection of sphere 1 and sphere 2 can be found as a circle as shown in the gray. And there will be two intersects from sphere 3 and this intersection circle as shown in the 2 red points. Usually, one of the red points should be near the ground and another is in the space. So only one of them is reasonable and the position of the user can be determined.

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

2.1.2. 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].

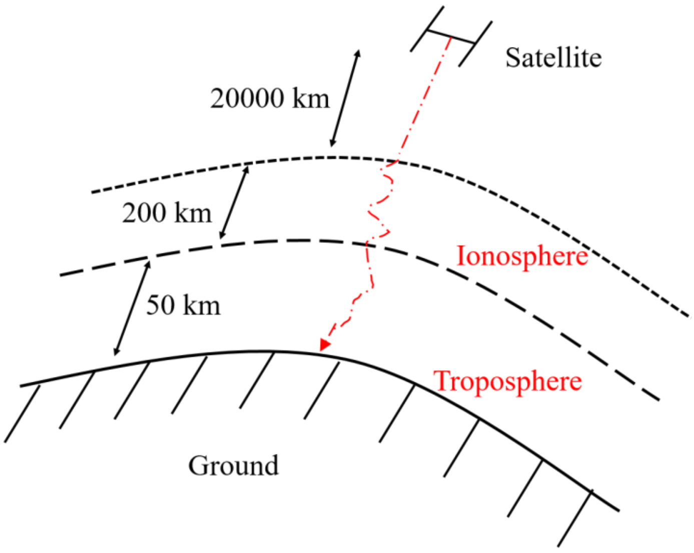


Figure 2.3 The signal propagation through the atmosphere.

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.

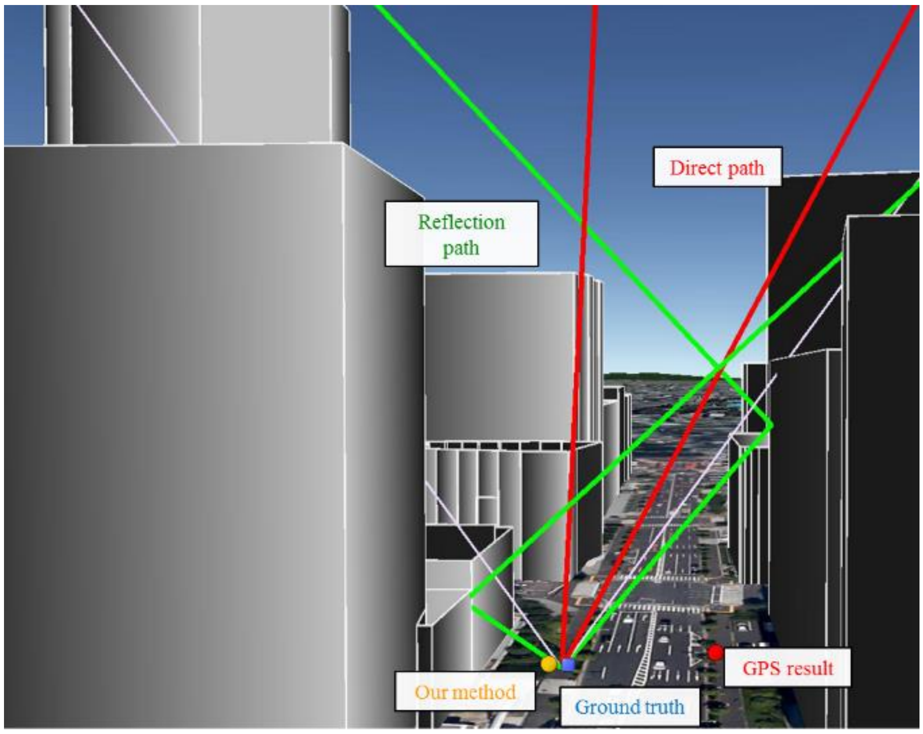


Figure 2.4 Ray-tracing simulation with 3D map.

2.2. PDR-based Positioning

2.2.1. General introduction of PDR

Dead Reckoning (DR) is a relative navigation technique, which is usually used for vehicle navigation by determining current vehicle position from the knowledge of the previous position and the measurements of motion direction and traveled distance. Pedestrian Dead Reckoning (PDR) applies the traditional DR algorithm for personal navigation applications. PDR system uses the sensors in the wearable devices, such as accelerometer, magnetometer and gyroscope. And it is often applied to obtain the personal position information at each step of the personal movement. Because the walking of the pedestrian is a cycle pattern, if we can detect it when each step happened and the walking direction and step length can be estimated, the trajectory of pedestrian can be determined. The PDR mechanization equation is given by:

$$x_k = x_{k-1} + l_k \begin{pmatrix} \cos\theta_k \\ \sin\theta_k \end{pmatrix} \tag{2.1}$$

The state x_k is the position after the kth step in the east and north coordinates relative to the starting point. And the stride length l_k and the moving direction θ_k are obtained in the PDR system. In this case, PDR is studied for step detection, and the traveled distance equals to step length these three parts as shown in Figure 2.5 [7].

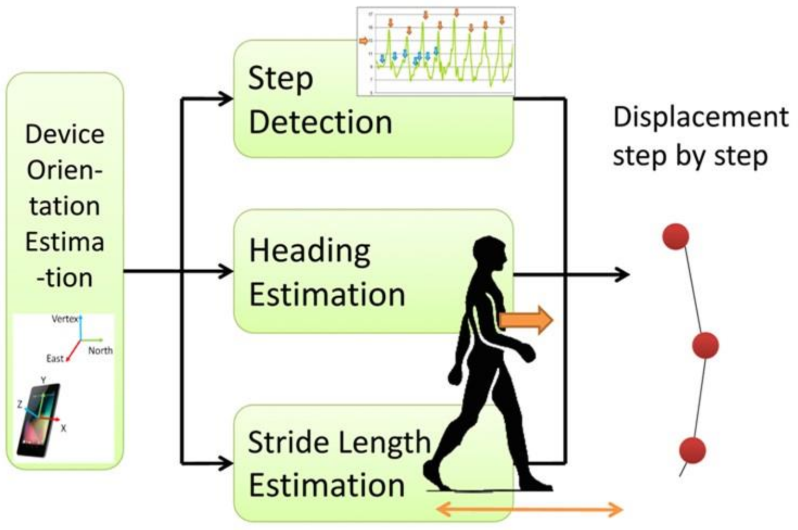


Figure 2.5 Overview of a typical PDR system

In a PDR system MEMS(micro-electro-mechanical-system) of the Inertial Measurement Unit (IMU) or the Inertial Navigation System(INS) can be can be placed on many space of the pedestrian's body, such as foot, waist, back, head, in the pocket and hold on the hand. It allow PDR can be used in many kinds of wearable devices such as smartphone, smart glasses, smart watch and smart bracelet, which are popular nowadays. Especially the smart phone in the pocket is a hot research topic in past few years [1, 3, 7, 11]. Then the three main parts of PDR will be discussed in turn.

2.2.2. Step detection

Steps can be detected based on the periodic pattern of vertical acceleration during walking. However, in real situations, the raw data of acceleration are noisy. In fact, many kinds of smoothing functions such as low-pass filter or moving average can be chosen to reduce the noise effect. What's more, because the coordinate system for

MEMS's local coordinate is different from the Global coordinate system, we need to transform measured data in MEMS's local coordinate to Global E-N-U coordinate. If gyroscope can be used and the initial positioning and placement information of MEMS can be known, the roll, yaw, pitch angle of the device can be accumulated. And there is another method to calculate rotation matrix to convert the data from local coordinate system to global coordinate system. By using acceleration data g and magnitude data m_N, m_U , we can get equations as:

$$[0 \quad 0 \quad g]^T = \boldsymbol{R} \times gravity \tag{2.2}$$

$$[0 \quad m_N \quad m_U]^T = \boldsymbol{R} \times geomagnetic \tag{2.3}$$

The *gravity* and *geomagnetic* are the gravity measurement and geomagnetic measurement component in local coordinate system and $\boldsymbol{R}$ is the rotation matrix, which can be determined by solving these two equations.

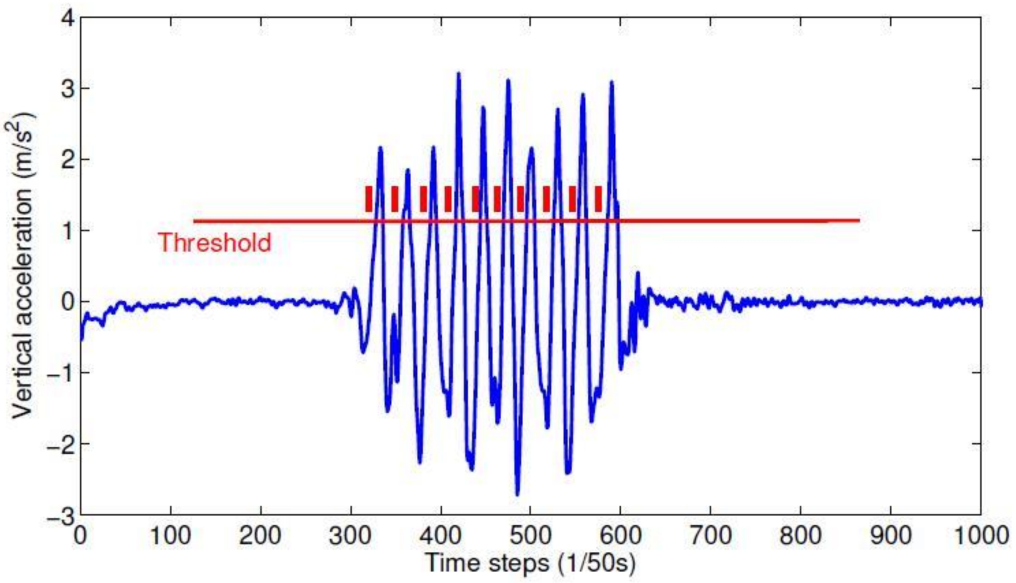


Figure 2.6 Step detection based on acceleration

Then the step can be easily detected when the vertical acceleration norm positive or negative going crosses a threshold as shown in Figure 2.6 [11].

2.2.3. Stride length estimation

The step length of different pedestrian varies a lot. Even for the same person, the step length changes dramatically during walking and is influenced by the different places where the device is holed.

Some researches just only consider the normal walking model. And there is also some researches have complex classifications in order to get more accurate estimation. For example, classify the motions into symmetrical motion modes, such as texting and phoning, as well as asymmetrical motion modes, such as hand swinging and trouser pocket. Kakiuchi et al. [7] at first use a decision tree classifier generated with the C4.5 training algorithm. From the acceleration and GPS speed values, the user's state of motion is continu-ously classified into "walking," "running," "stationary" and updated periodically as a system state information.

Then the stride length is estimated using two different models. In the walking mode, the stride length *l* is given by:

$$l = K_{\omega}\left(a_{v,max} - a_{v,min}\right)^{\frac{1}{4}} \tag{2.4}$$

The state a_v is the vertical acceleration in a period of step. K_{ω} and K_r below are the coefficients that need to be adjusted for different subjects. This equation is widely used in many subsequent PDR systems.

In the running mode, however, *l* is estimated from the average of magnitude of horizontal acceleration $a_h = \sqrt{a_e^2 + a_n^2}$ in N samples:

$$l = K_r\left(\frac{\Sigma_i\sqrt{a_{e,i}^2 - a_{n,i}^2}}{N}\right)^{0.7} \tag{2.5}$$

2.2.4. Heading estimation

The most difficult part in the PDR system is heading estimation. Some of the researches directly use the gyroscope [8] and magnitude [11] data to get the walking direction. And principle component analysis (PCA) is another popular choice in many researches [7, 9, 10]. The back and forth movement of the leg produces a large variance of horizontal acceleration in a direction parallel to the anteroposterior axis. Therefore, PCA is applied in the sequence of east and north components of acceleration (a_e and a_e), which can be estimated when transforming acceleration measured data in MEMS's local coordinate to Global E-N-U coordinate.

Kakiuchi et al. [7] apply eigenvalue decomposition of the covariance matrix to a_e and a_e after smoothing these two sequences with a moving average. Then the resulting first eigenvector gives an estimation of the direction that is parallel to the antero-posterior axis. The cyclic patterns of vertical and forward acceleration have a temporal relation such that a vertical dip is closely followed by a forward peak.

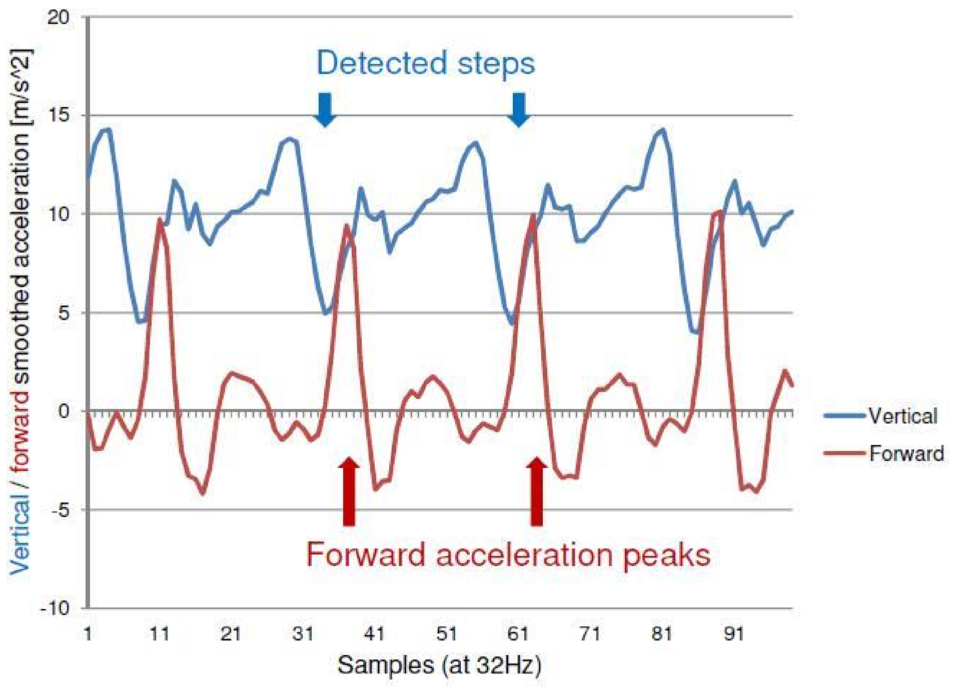


Figure 2.7 Vertical and forward smoothed acceleration.

So, to determine the forward direction from the antero-posterior acceleration, which is the horizontal acceleration projected onto the first eigenvector, they detect the peak and dip from the antero-posterior acceleration after the detected vertical dip. Therefore, as shown in Figure 2.7 [7], if the peak of the antero-posterior acceleration is earlier, the eigenvector is determined as the forward direction. If not, the direction opposite to the eigenvector is determined as the forward direction.

2.2.5. 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.

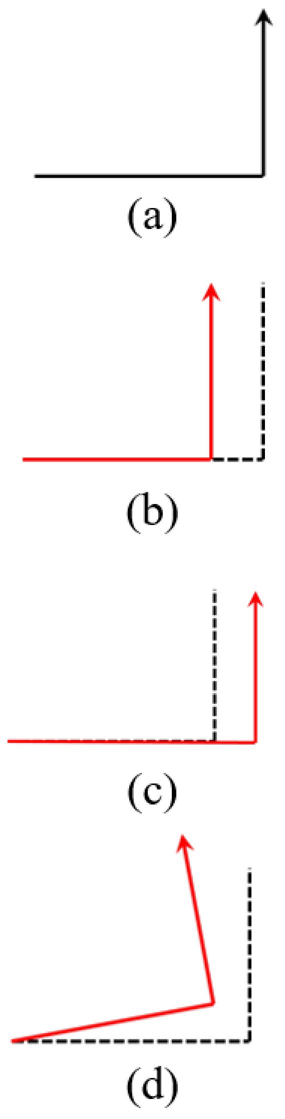


Figure 2.8 Different error accumulation of PDR system.

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.

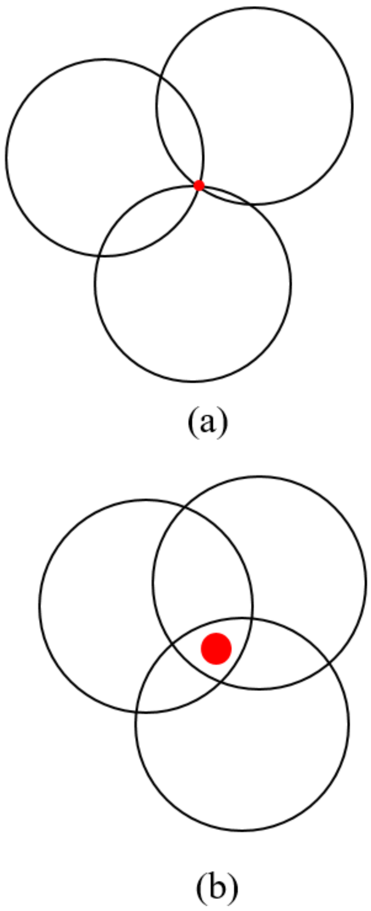


Figure 2.9 Position estimation by trilateration. (a) ideal condition (b) imperfect situation.

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.

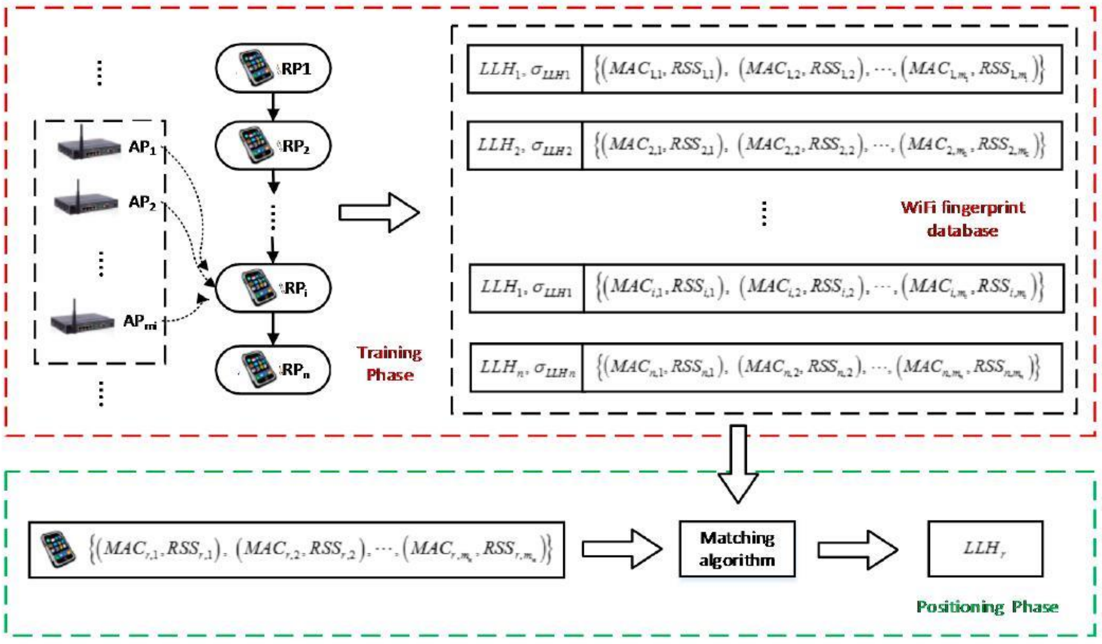


Figure 2.10 Procedure of Wi-Fi fingerprinting.

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

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

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:

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

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.

2.4. Fundamental of Integrated Positioning System

As mentioned above, recent techniques based on GPS, Wi-Fi and PDR standalone have several limiting problems, such as the NLOS and multipath of GPS, the drift of PDR, and the variation of Wi-Fi signals. Therefore, to solve the problems, the integration of different positioning systems becomes an effective way. There have been many focusing on the integration of Wi-Fi and PDR in the indoor environment [8, 11, 12, 13] and the integration of PDR and GPS in the outdoor environment [1]. In addition, with the rapid increase in Wi-Fi APs in outdoor areas, some researchers start to implement Wi-Fi-based positioning in outdoor environment and the integration of Wi-Fi and GNSS also comes out [5]. In these researches, Kalman filter is the most used integrating framework.

The basic idea to use Kalman filter is to update the state iteratively. It is a mathematical optimization algorithm that makes an estimation of an observed variable using a predation and update equations. This method performs much better than using the update value which is using signal measurement alone. Traditional Kalman filter can only be applied to liner system and the pedestrian's walking pattern just could be approximate by a liner model. Therefore, PDR system is suitable for being the state model in Kalman filter. Take the integration between PDR and Wi-Fi-based positioning as an example. Assume X_t is the 2D coordinate of the pedestrian and $d_t = l_t(\cos\theta_t, \sin\theta_t)^T$ is the input from PDR, where l_t is the step length and θ_t is the moving direction at the time step t. Then, based on PDR, the state transition function of the sensor fusion framework is shown as:

$$X_t = FX_{t-1} + Gd_t + v \tag{2.9}$$

The *F*, *G* are identity matrices and v denotes the Gaussian noise of the motion model with zero mean and covariance matrix M [11]. The observation function can be obtained based on the Wi-Fi-based positioning result Z_t at time step t as follows:

$$Z_t = HX_t + p \tag{2.10}$$

The *p* denotes the Gaussian noise of Wi-Fi-based positioning with zero mean and covariance matrix N. Since it is a direct observation problem, *H* is an identity matrix. Then there are the predicting and updating processes in the algorithm.

The predation equations are:

$$Z_{t|t-1} = FX_{t-1|t-1} + Gd_t \tag{2.11}$$

$$P_{t|t-1} = FX_{t-1|t-1}F^T + M \tag{2.12}$$

Where P is the measured covariance of the variable of interest.

The update equations are:

$$K_t = P_{t|t-1}H^T\left(HP_{t|t-1}H^T + N\right)^{-1} \tag{2.13}$$

$$X_{t|t} = X_{t|t-1} + K_t\left(Z_t - HX_{t|t-1}\right) \tag{2.14}$$

$$P_{t|t} = (I - K_tH)P_{t|t-1} \tag{2.15}$$

The K is the Kalman gain. The prediction equation find the priori distribution by predicting the mean and variance of the next value of the observed variable. The update equations calculate the posterior distribution by using the new weighted measurements.

Similar with Wi-Fi-based positioning, GPS can also be used to integrate with PDR. The basic idea of integrating PDR and GPS under Kalman filter framework is shown in Figure 2.11. As shown in Figure 2.11, PDR is used as state model while GPS is used as obsercation. And Finally, the integrated positioning system can performs much better result than using the update value using each of the method standalone.

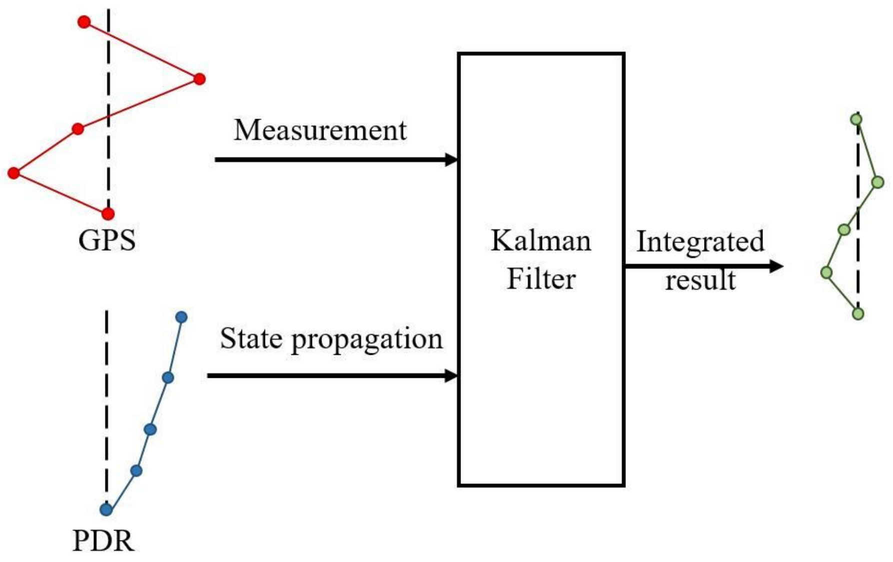


Figure 2.11 Fusion of GPS and PDR.

Not only GPS, PDR and Wi-Fi can be integrated with each other, some other information can also be used to help to improve the positioning accuracy.

For example, map information can also be used in the integrated positioning system [8, 11]. With the help of map information and map matching algorithm, the space where the pedestrian can move to will be limited, and the pedestrian moving trajectory can be corrected at some corner or near some wall.

Considering that nowadays in the next generation of intelligent devices, many of them have been equipped with high-resolution cameras, image information also can be introduced into the integrated positioning system. Then in next chapter, we will introduce the related work about image-based positioning and our proposed method to provide pedestrian positioning with the aid of pedestrian’s geo-tagged image and Google Maps Street View.

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-200 which our lab has and the position of the camera sensor. 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.

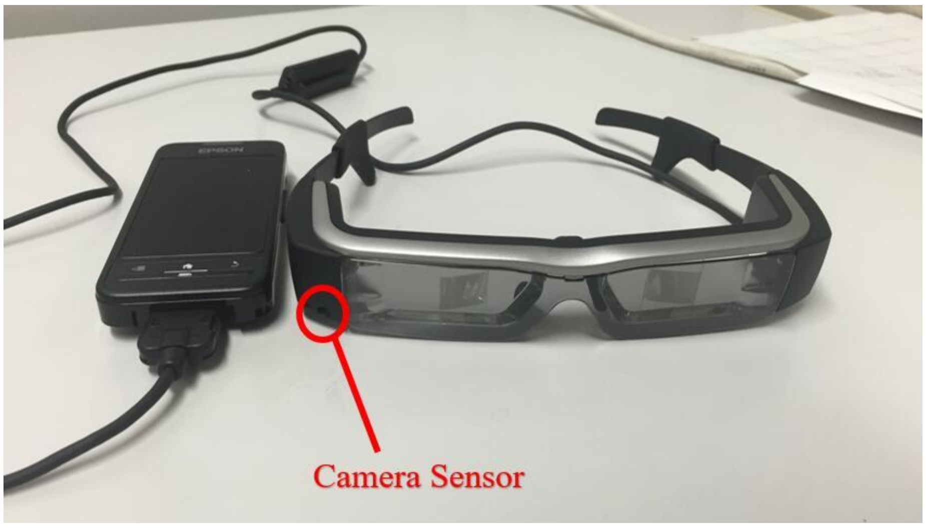


Figure 3.1 EPSON Moverio BT-200 and the position of camera sensor

Of course, smartphone can also be used when the pedestrian hold the smartphone to take the video and AR navigation guide can also be shown on the screen combined with the camera scene. Unfortunately, the resolution of the embedded camera on EPSON Moverio BT-200 is poor so there will be a huge noise and distortion on the image or video taken from it. In this situation, our experiments are based on the device of the Google Nexus 5 smartphone and we hold the smartphone in front of the eyes to take video and collecting data to simulate the situation of a wearable camera.

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).

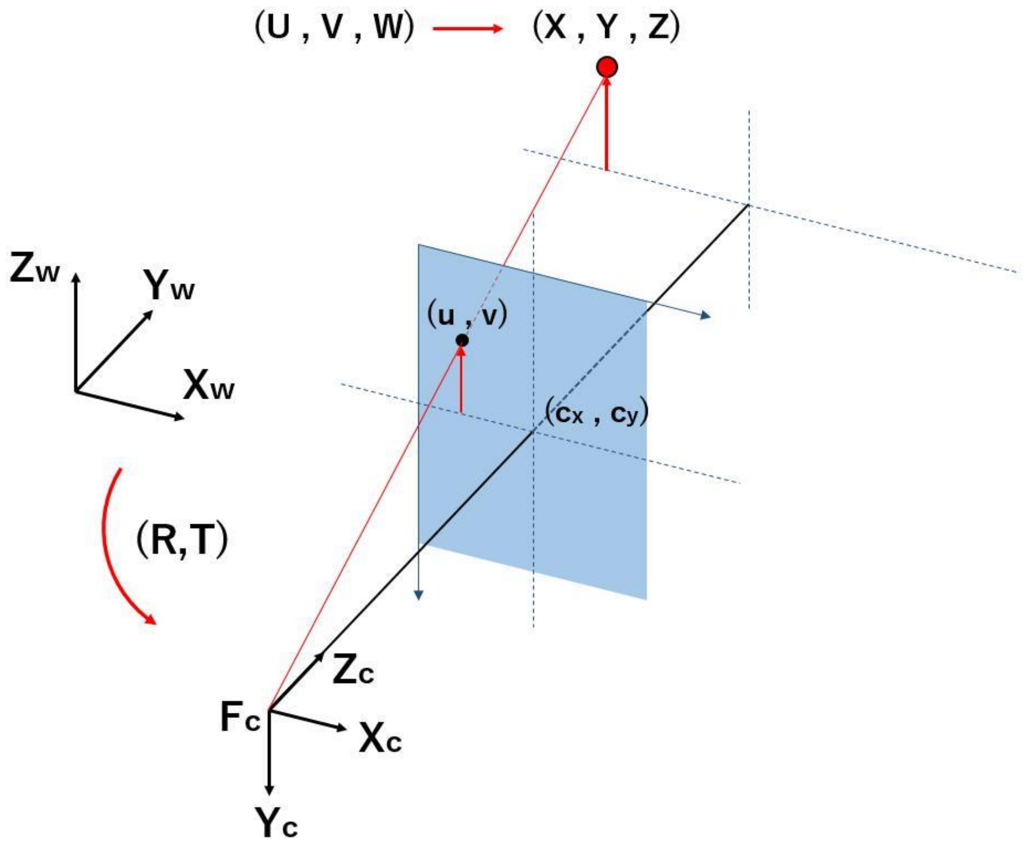


Figure 3.2 The geometry of a pinhole camera model

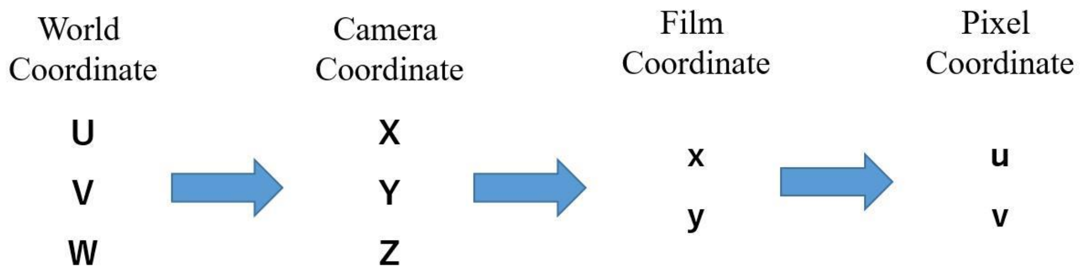


Figure 3.3 The process of projecting a feature in the world coordinates into the pixel coordinates

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.

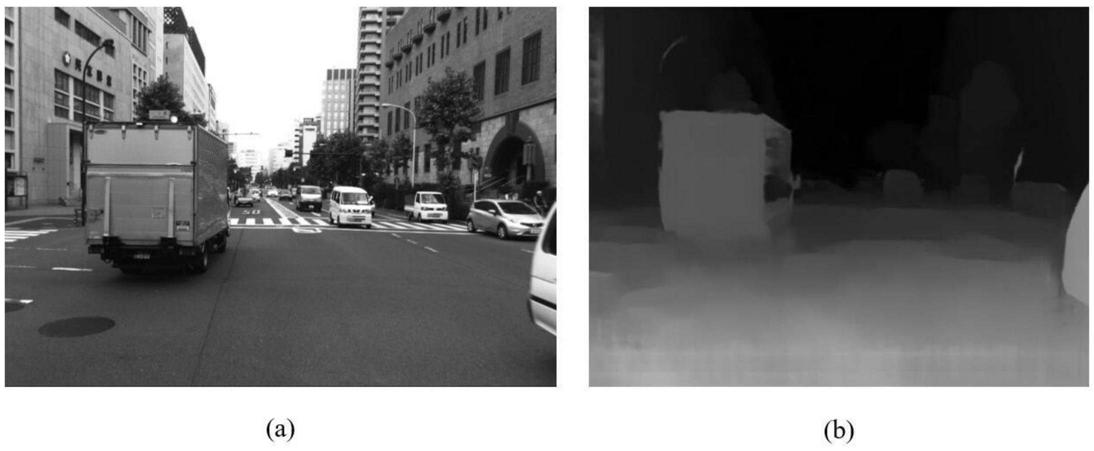


Figure 3.4 Stereo camera images. (a) Image from the left camera; (b) Depth map

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.

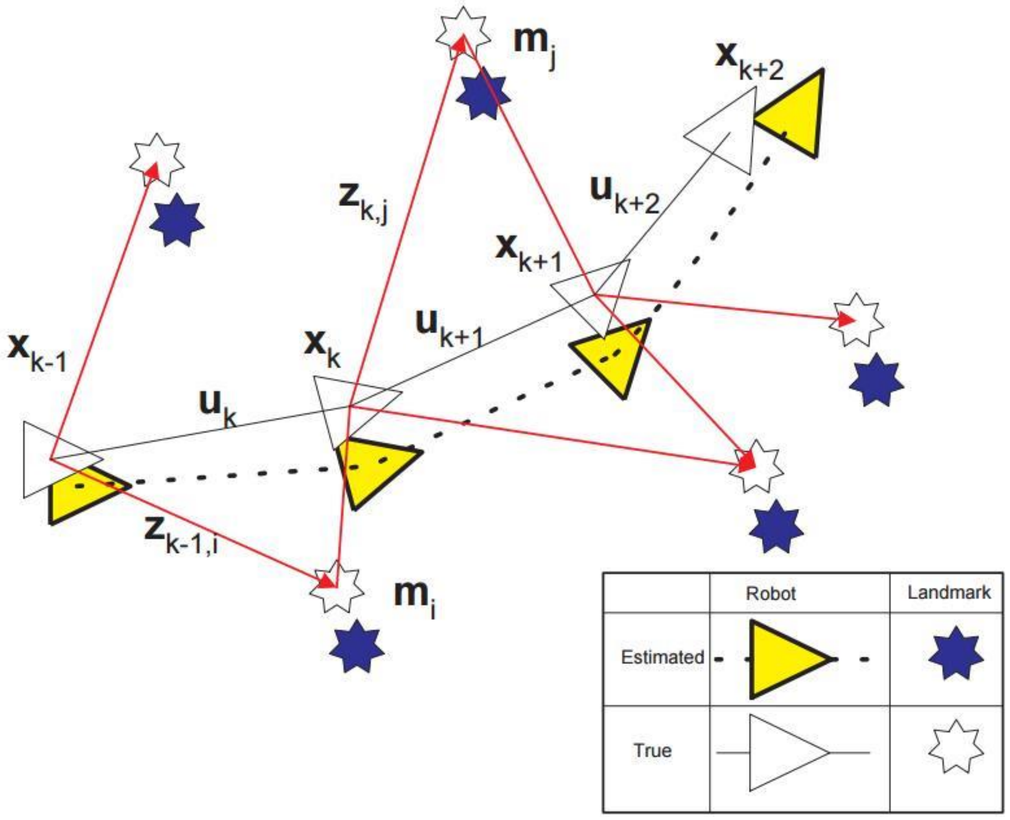


Figure 3.5 Process of the SLAM.

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.

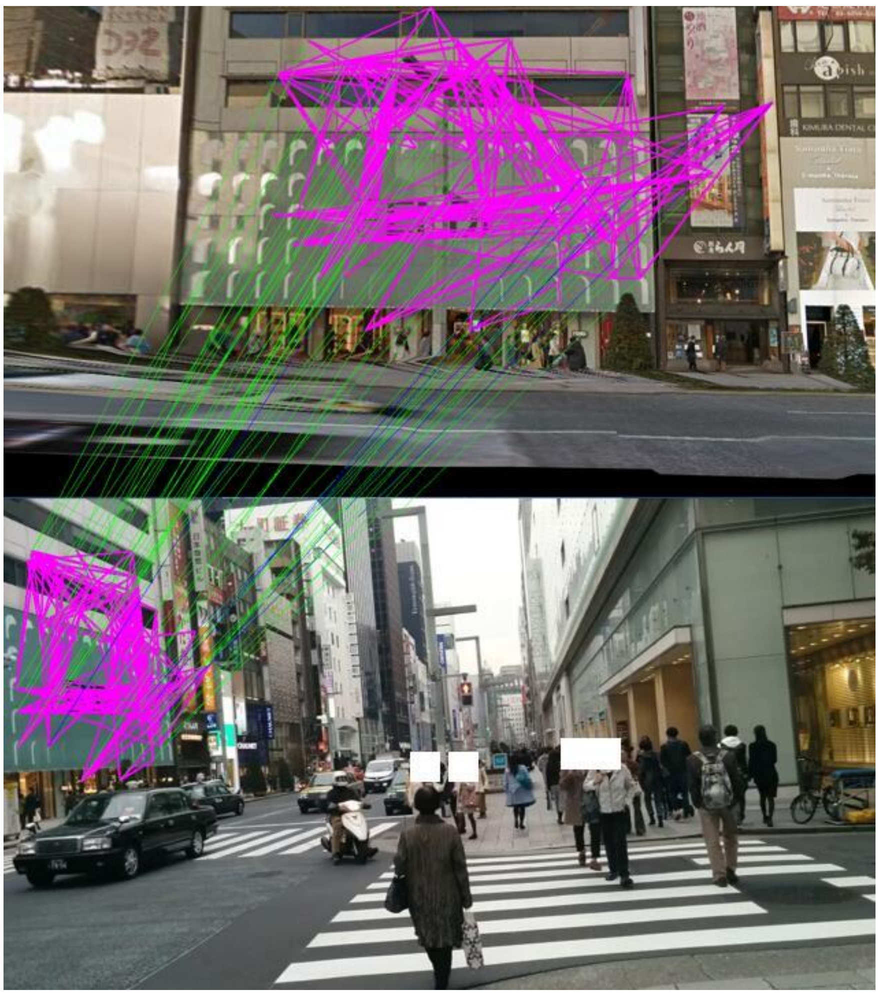


Figure 3.6 Matching between a reference image (left) with the que-ry image (right) based on kVLD

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

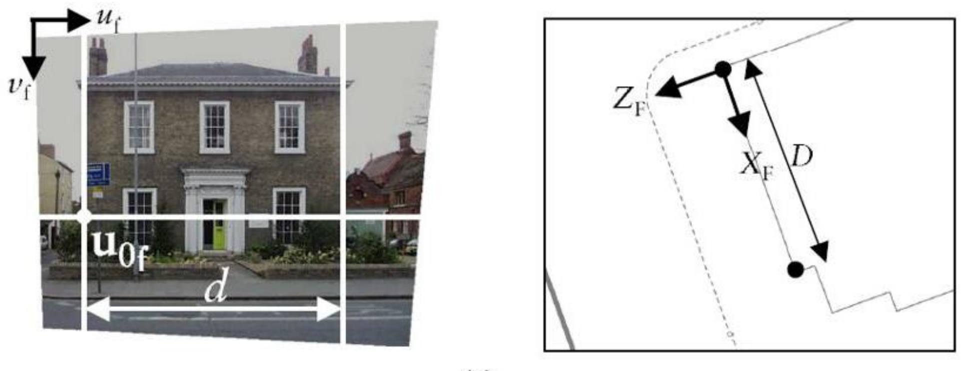
3.1.4. Related works on positioning with the aid of Google Maps Street View.

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

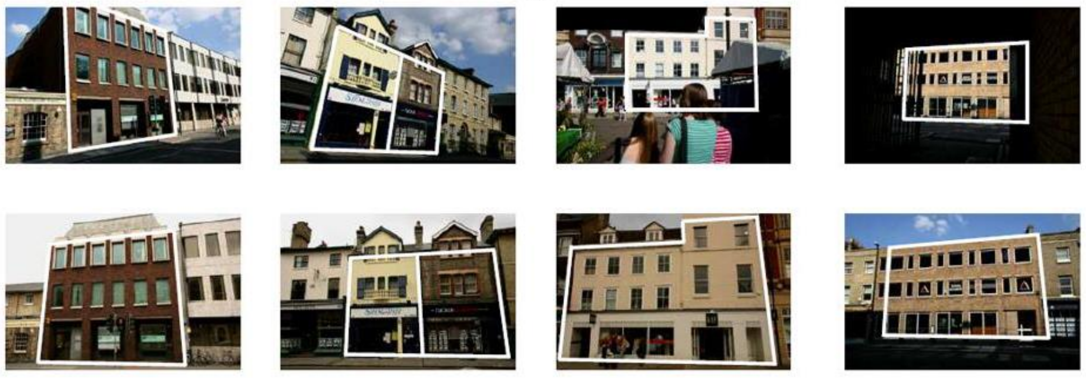
Google Maps Street View can be considered as a very comprehensive and large database provided by Google, which consists of geo-tagged 360^o panoramic images of almost all main streets and roads in a number of countries. The panoramic images of Google Maps Street View are recorded by a spherical arrangement 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



(a)



(b)

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.

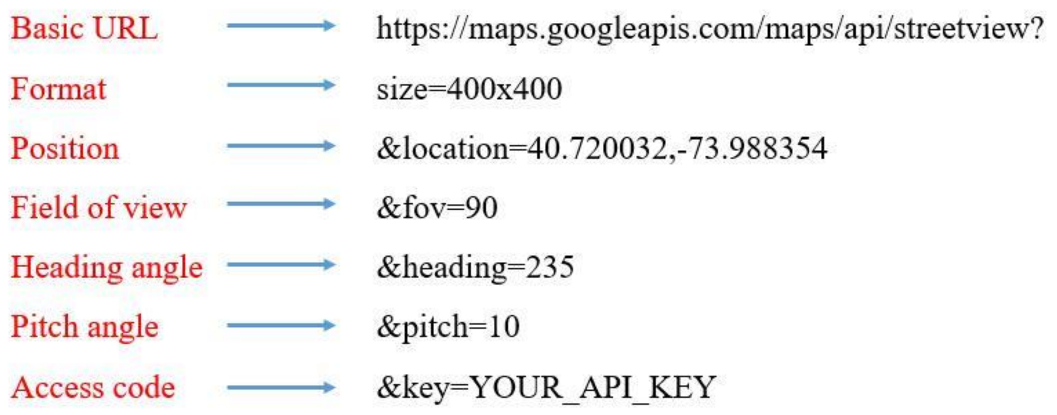
Some other researches choose to focusing on localizing images in large scale metrical maps built from Structure from Motion (SfM). Irschara et al. [23] build accurate point clouds using structure from motion and then compute the camera coordinates of the query image. Zamir et al. [24] leverage a structured 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. The Google Maps Street View Pipeline

At first, the Google Street View Image API is used to download Street View images from specific locations. The viewport is defined with URL parameters sent through a standard HTTP request, and is returned as a static image [30]. Figure 3.8 shows the parameters of the URL link to access to an image.



Basic URL	https://maps.googleapis.com/maps/api/streetview?
Format	size=400x400
Position	&location=40.720032,-73.988354
Field of view	&fov=90
Heading angle	&heading=235
Pitch angle	&pitch=10
Access code	&key=YOUR_API_KEY

Figure 3.8 Description of different parameters in the URL of the Google Street View Image API

The image is specified using request parameters as shown in Figure 3.8. As is standard in URLs, all parameters are separated using the ampersand character "&". Allowed parameters and their possible values are listed below:

Format: The parameter *size* specifies the output size of the image in pixels. Size is specified as $width \times height$. For the free version of the Google Street View Image API, a maximum size of 640×640 can be loaded. This parameter is one of the required parameters which must be included in the URL.

Position: There are two kinds of parameters can be choose to use as the position parameter. One is the parameter *location*, which is in form of latitude and longitude coordinates. Note that this does not load the image with this exact position, but searches a 50 meter radius and output the closest image to the input position that is available in the Google Maps Street View database. Another choice is the parameter *pano* which is a specific generally stable ID for the panorama. This parameter is also one of the required parameters which must be included in the URL.

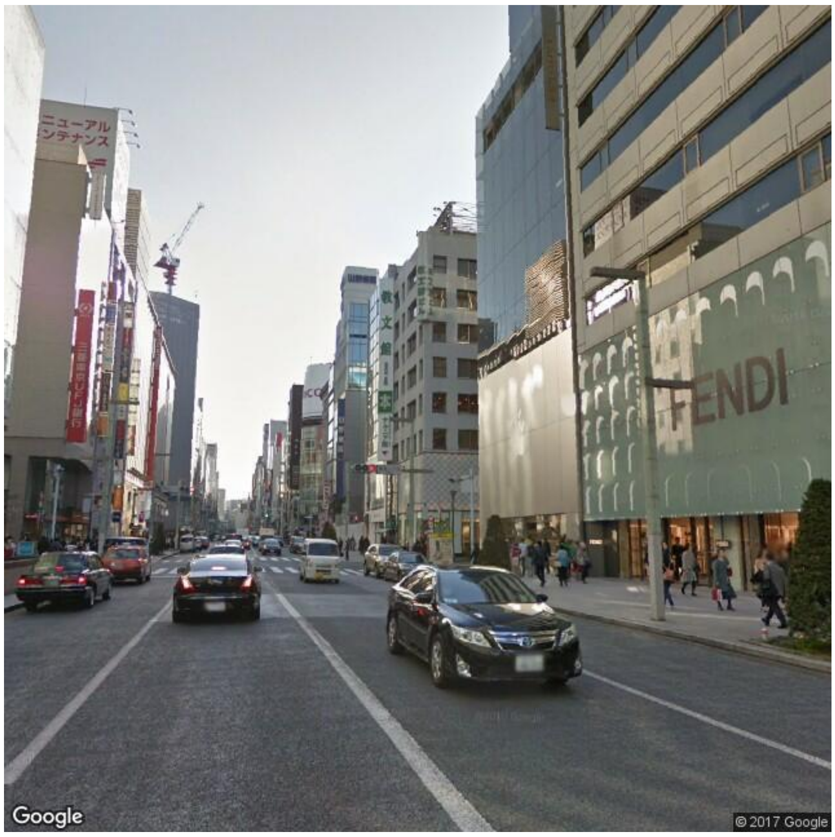


Figure 3.9 Street View image downloaded from the URL link of API

However, there is a problem coming from the position parameter. If we use the parameter *location*, because the Google Street View Image API output the image with the position most close to the input position, it means the position of the output Street View image is not taken from the position we required and we even cannot know the true position of the output image. And if we want to use the parameter *pano*, the panorama ID information cannot be directly found in the Google Street View Image API. Therefore, some other API from Google is need to require the true position of a panorama or the panorama ID information.

Access code: The parameter *key* is necessary to sign the URL. It can be easily applied for a free version key from the Google Street View Image API website.

Field of view: The parameter *fov* determines the horizontal field of view of the image. The field of view is expressed in degrees, with a maximum allowed value of 120. When dealing with a fixed-size viewport, as with a Street View image of a set size, field of view in essence represents zoom, with smaller numbers indicating a higher level of zoom. The parameter *fov* is one of the optional parameters and its default value is 90. But to match with the camera we used, we set the parameter value as 60.

Heading angle: The parameter *heading* indicates the compass heading of the camera. Accepted values are from 0 to 360 which is indicating North. And heading angle of 90 indicate East, and 180 indicate South. Although heading angle is also one of the optional parameters, but this is an important parameter in our proposed method because we need to tag each images in our dataset with their heading angles for positioning.

Pitch angle: The parameter *pitch* specifies the up or down angle of the camera relative to the Street View vehicle. Positive values angle the camera up with 90 degrees indicating straight up, and negative values angle the camera down with -90 indicating straight down.

After setting the parameters above, we can download images from Google Maps Street View and Figure 3.9 shows a sample of downloaded image.

As mentioned above, use 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. Therefore, we build a datasets of building facades images as shown in Figure 3.10.

As mentioned above, If we use the parameter *location*, because the Google Street View Image API output the image with the position most close to the input position, it means the position of the output Street View image is not taken from the position we required and we even cannot know the true position of the output image. And if we want to use the parameter *pano*, the panorama ID information cannot be directly found in the Google Street View Image API. Therefore, some other API from Google is need to require the true position of a panorama or the panorama ID information. Therefore, the Google Street View Image API metadata requests can be used to find the information of the Street View panoramas. By using the metadata we can find out if a Street View image is available at a given location, as well as getting programmatic access to the latitude and longitude, the panorama ID, the date the photo was taken, and the copyright information for the image.

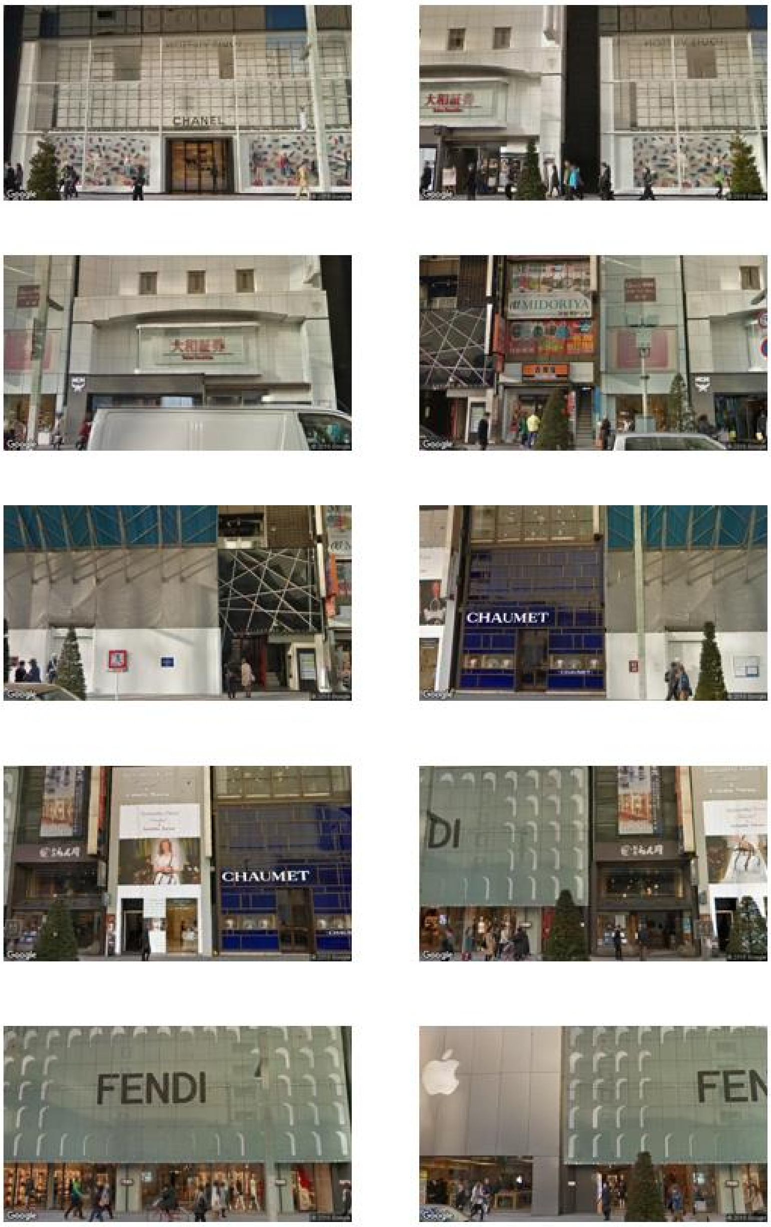


Figure 3.10 Building facades datasets in Ginza area.

A Street View image metadata request is an HTTP URL as:

https://maps.googleapis.com/maps/api/streetview/metadata?parameters

The metadata requests accept the same URL parameters as the Street View Image API imagery requests, although only the parameters of *location* and *key* are required. The parameter of *size*, *heading*, *fov*, and *pitch* can also be included in the metadata request but these parameters do not influence the data about the panorama, or which panorama is found. The API allows the inclusion of the same parameters as the imagery request to make it easier to construct a metadata request related to a specific imagery request, but for metadata requests, the API ignores the optional parameters and their values. And the metadata responses are returned in JSON format as:

```
{
    "copyright" : "© 2017 Google",
    "date" : "2016-03",
    "location" : {
        "lat" : 35.672411638183,
        "lng" : 139.7662904016759
    },
    "pano_id" : "IHH_-WgFIpu1N5S-pw0HMg",
    "status" : "OK"
}
```

By using the metadata, we can easily tag each images in the building facades datasets with their true position, panorama ID information and orientation angles.

However, Google Street View Image API only provide the street view images in specific positions. Even though we can use the parameters of orientation to control the camera to change the view angle, the position of the camera in the scene is still fixed and cannot move freely. Therefore, to generate the virtual view in any place we want, the 3D information of the Google Maps Street View needs to be known.

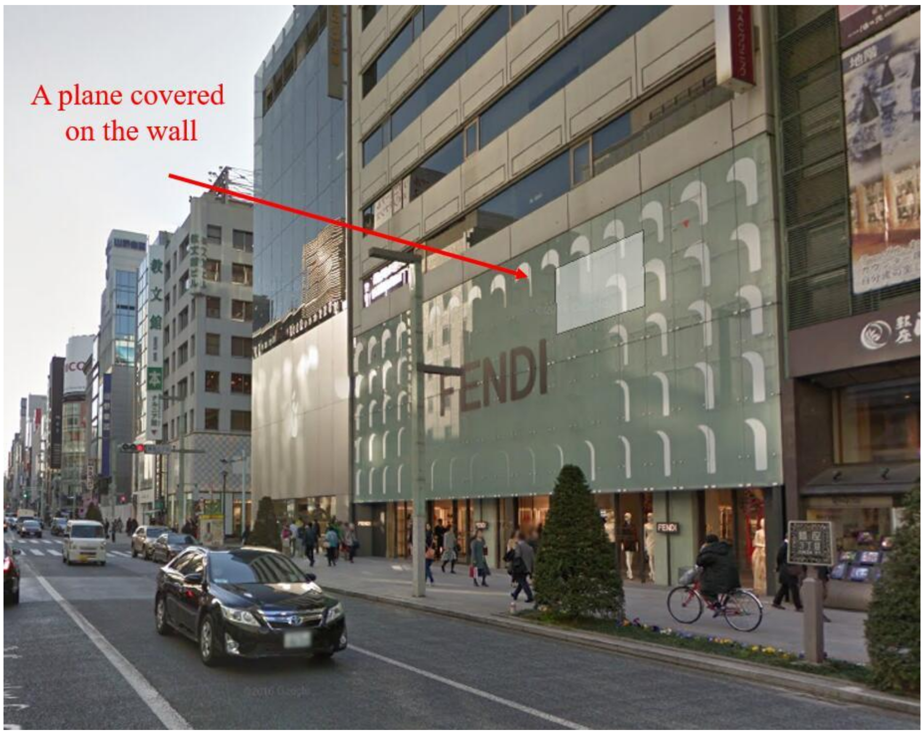


Figure 3.11 3D information in the Google Maps Street View

As shown in Figure 3.11, when we move the pointer to the wall in a Google Maps Street View scene, there will be a plane correctly covered on the wall. It means the Google Maps Street View should contain the 3D model information in it.

Fortunately, we found that the 3D information of Google Maps Street View can be request from another URL as:

http://cbk0.google.com/cbk?

Similar with Google Street View API, there are two input parameters. One is the parameter *output*, from which we can select the output data in XML format or JSON format. Another parameter is the *location* or *pano*.

The output of this URL request contain the basic information of the panorama similar with the metadata. It also provide the ID of neighborhood panoramas which is linked with current panorama. And most importantly, this request provide the depth map of current panorama.

However, the response of the depth map is encoded in base64 and we need to decode it before using. After decoding the depth map, the data format is shown as:

Header size: The length of the header parameters.

Number of planes: There are how many planes are modeled in the current panorama.

Map size: The width and height of the depth map.

Depth map indices: For each pixel in the depth map, which plane it belongs to.

Depth map planes: The normal vectors and the distances from the original point in the scene to each plane in the current panorama.

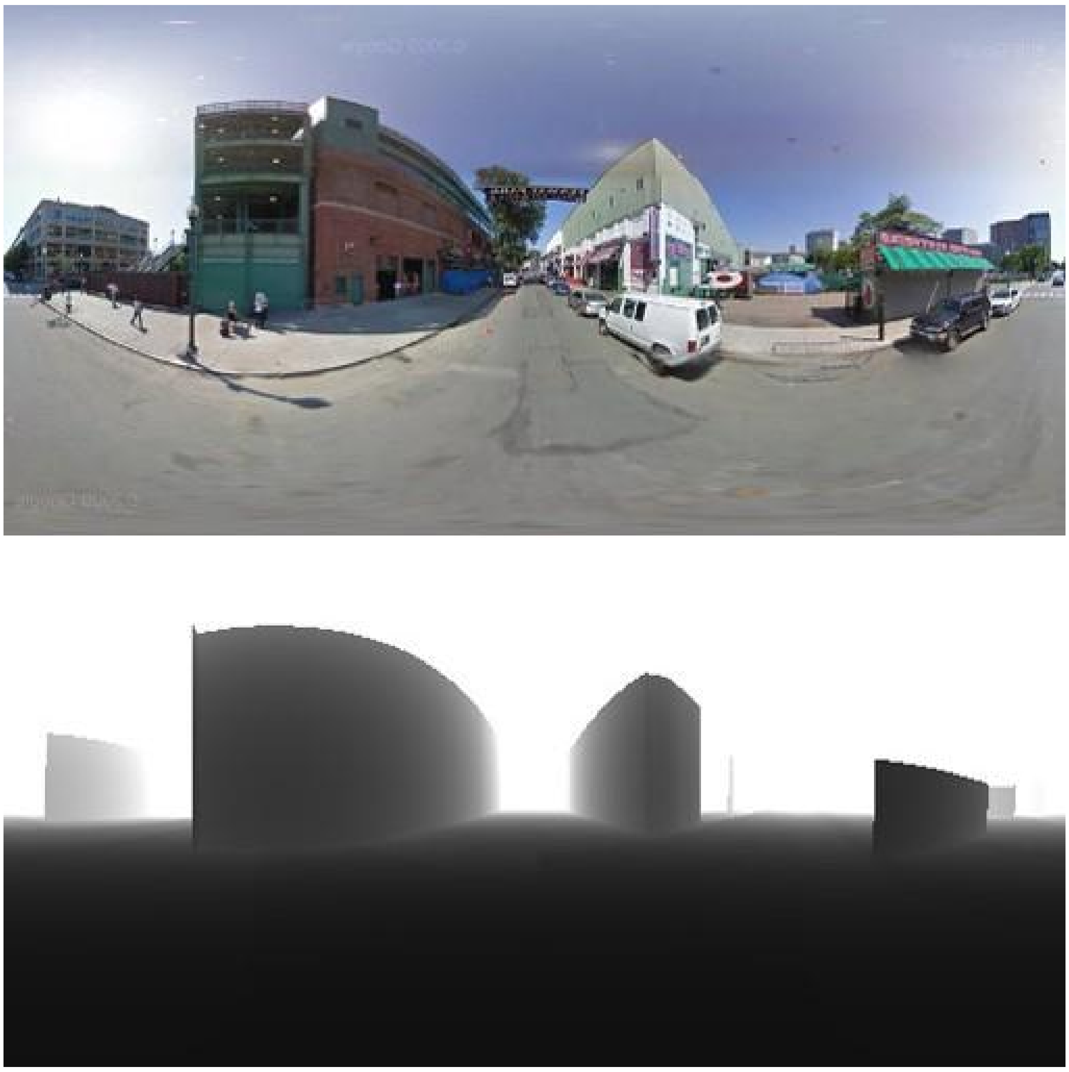


Figure 3.12 A sample of panorama and the depth map

By using the panorama image, depth map and the planes information in it, then we can construct a textured local 3D model as shown in Figure 3.13.

Although the car is textured on the ground so it looks strange, but we can see that the texture of the walls are almost correct. With the help of this texture map, we can easily generate virtual view in any places in the scene.

3.3. Proposed method

3.3.1. Objective

Now, the input of our system should be the image from the wearable camera. And this image should be tagged with the geographical position information from the commercial GPS with about 20 meter error and the heading angle from the magnetometer in the device with about 20 degree error.

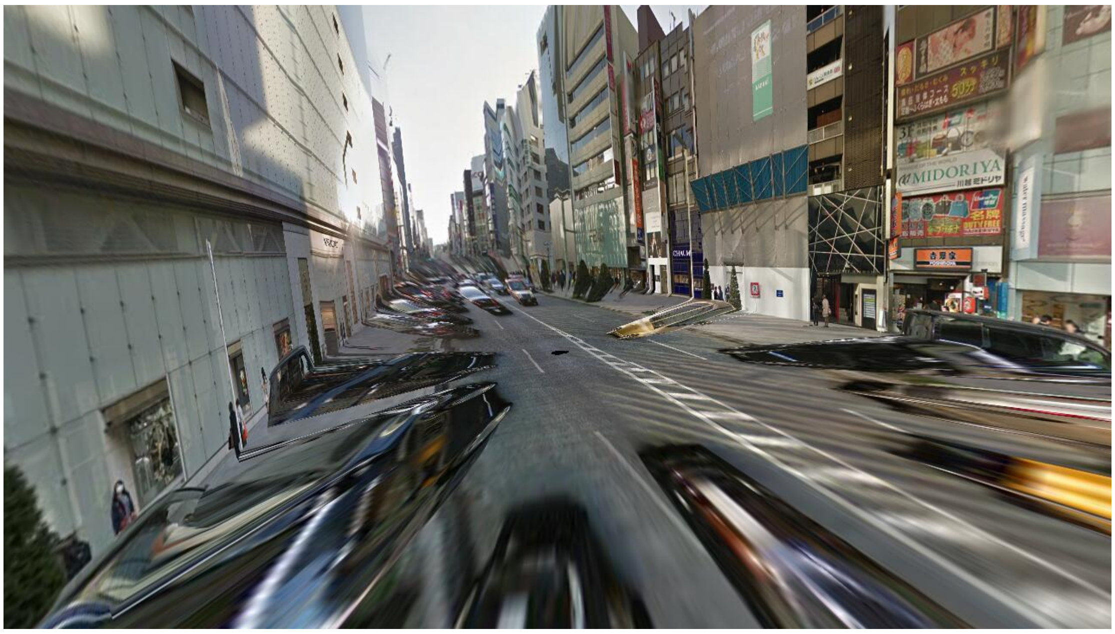


Figure 3.13 Textured local 3D scene made from panorama image and depth map.

And the source we can use include the building façades datasets from the Google Street View Image API and the textured local 3D model from the depth map and panorama image.

The objective of our proposed method is to search a virtual view which is almost totally matched with the pedestrian's input image. After the best matched virtual view can be searched, the pedestrian's position should be decided from the location information of this virtual as shown in Figure 3.14.

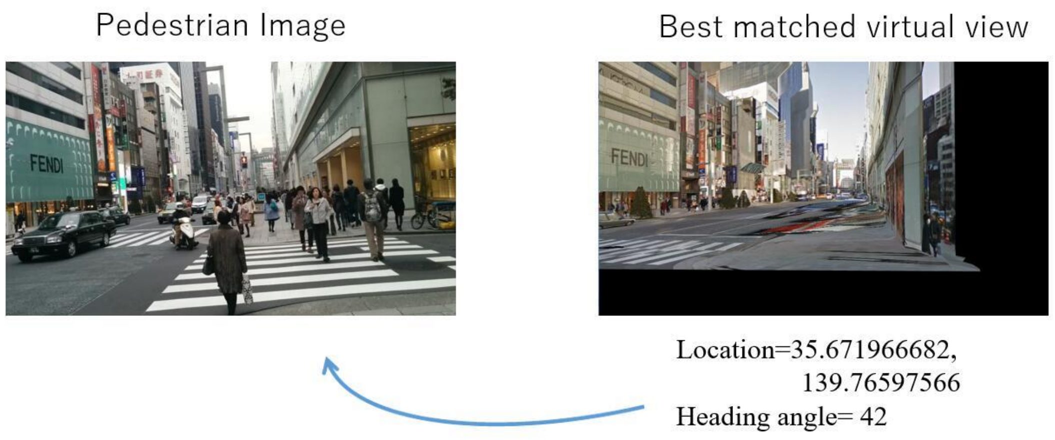


Figure 3.14 Objective of the proposed method

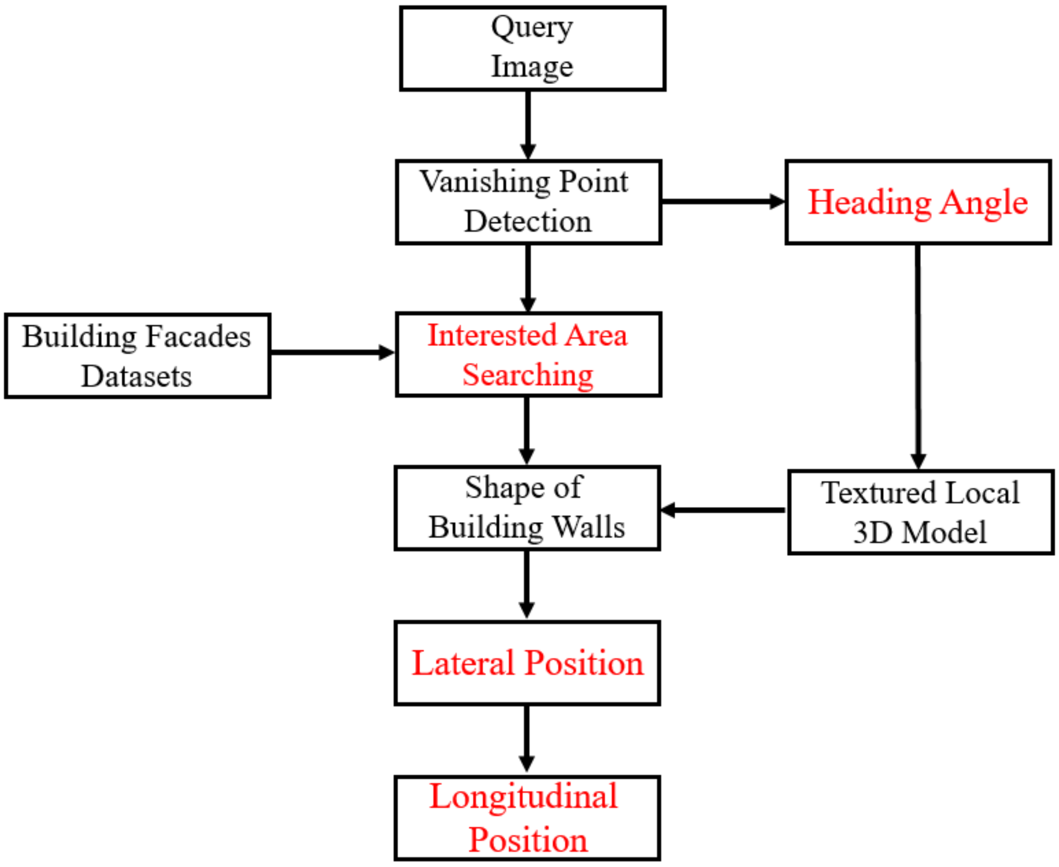


Figure 3.15 The flowchart of our proposed method.

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 estimation, lateral position estimation and longitudinal position estimation.

Because our objective is to search an almost totally matched virtual view with the input pedestrian image, we decide to evaluate a virtual view with the average distance between each matched key features in kVLD method 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 problem, one of the parameters should be fixed at first.

3.3.2. Vanishing point based heading angle estimation

Attitude determination is a critical point in navigation systems. In the urban area, the environment will easily influence magnetometer and the error may become 20 degree 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), can be used to determine the camera attitude. 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.

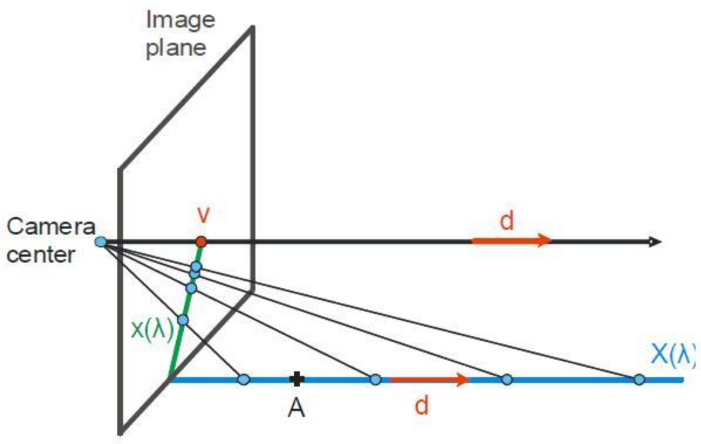


Figure 3.16 Projecction of a line in 3D-space onto the camera image plane

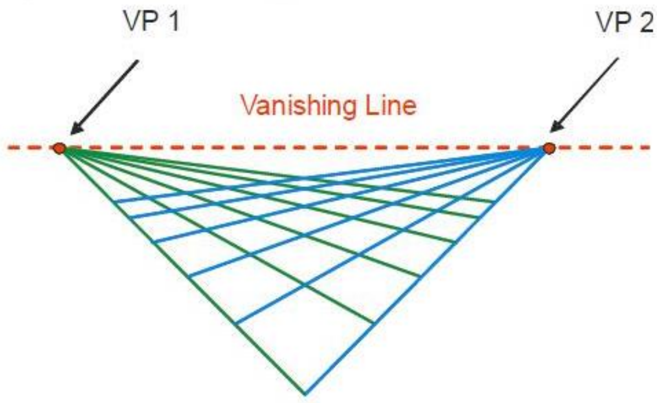


Figure 3.17 Vanishing Line and Vanishing Points within that line in the 3D vision.

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.

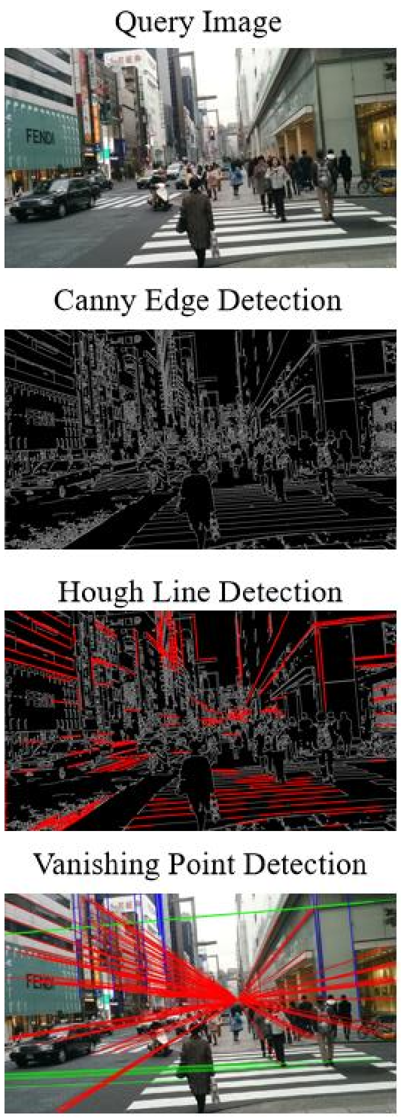


Figure 3.18 Process of vanishing point detection

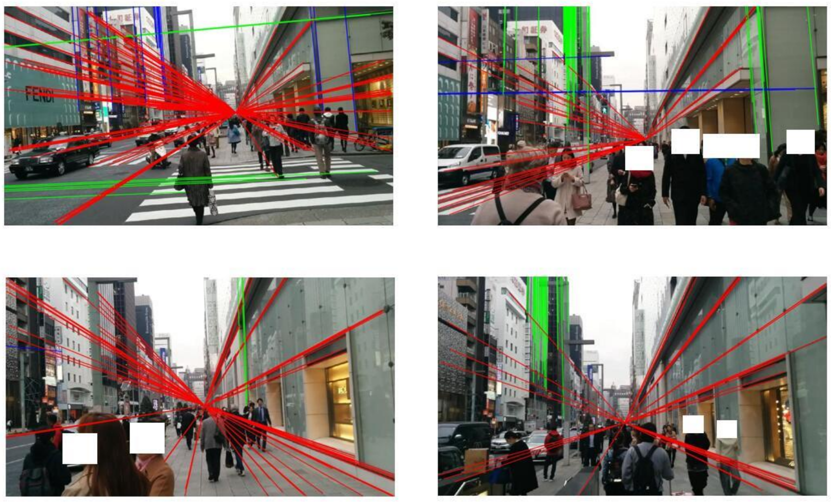


Figure 3.19 Vanishing points in the pedestrian image.

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.

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:

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

3.3.3. Lateral position estimation

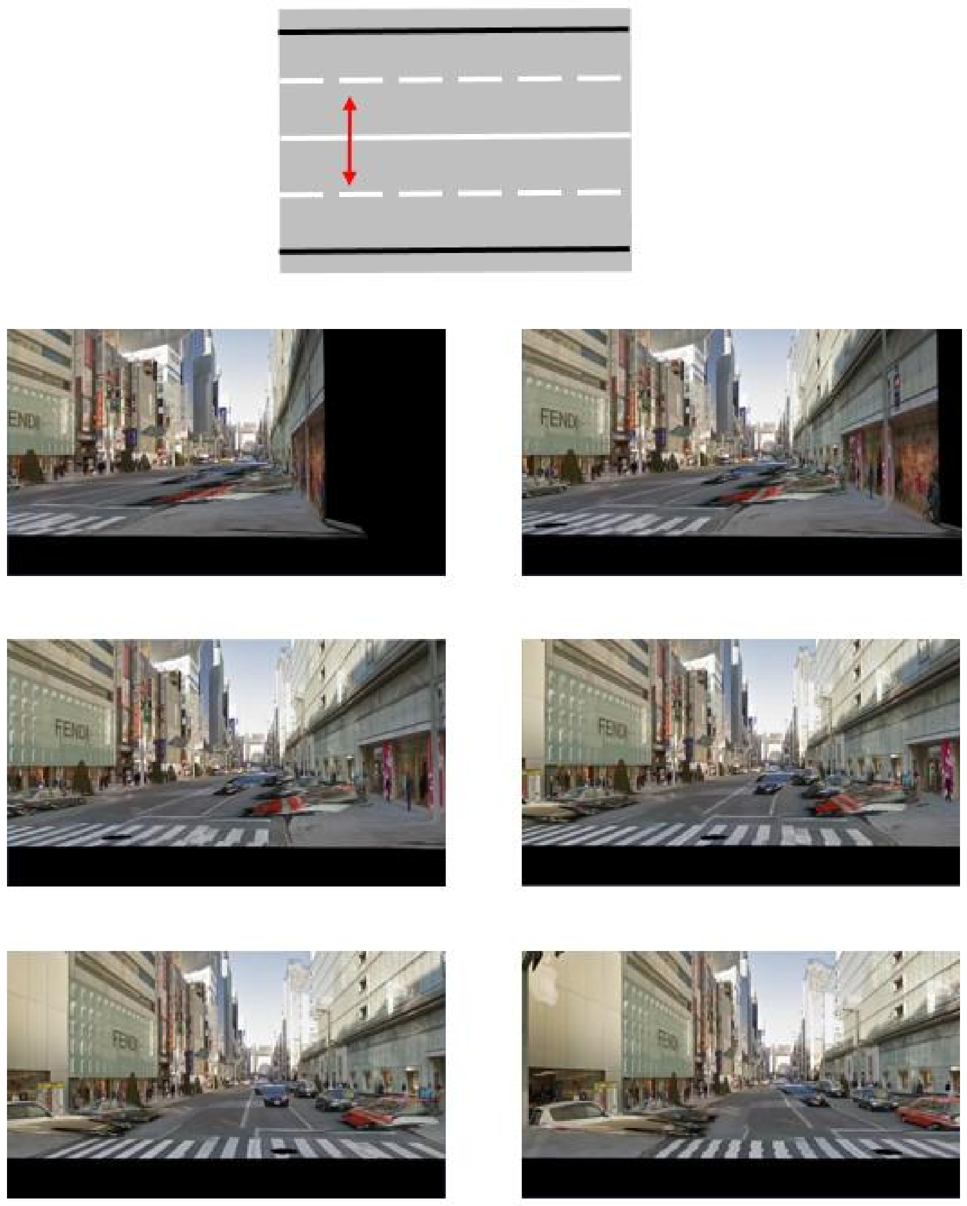


Figure 3.20 Virtual views with different lateral position.

From Figure 3.19, we can found that, when the heading angle is fixed and the camera is moving alone the lateral direction, the building wall will be in different shape. This characteristic can be used to guide us move the camera of virtual view to the correct side of road.

As shown in Figure 3.20, for each input query image, at first we determine the roads with vertical distance less than 20 meters from its geo-tag. Then we use the reference images from the building facades datasets with the distance less than 10 meters to the vertical point on each of these roads as the reference dataset for this query image to search the interested areas. In addition, the geographic coordinates and heading angle of each reference images is tagged.

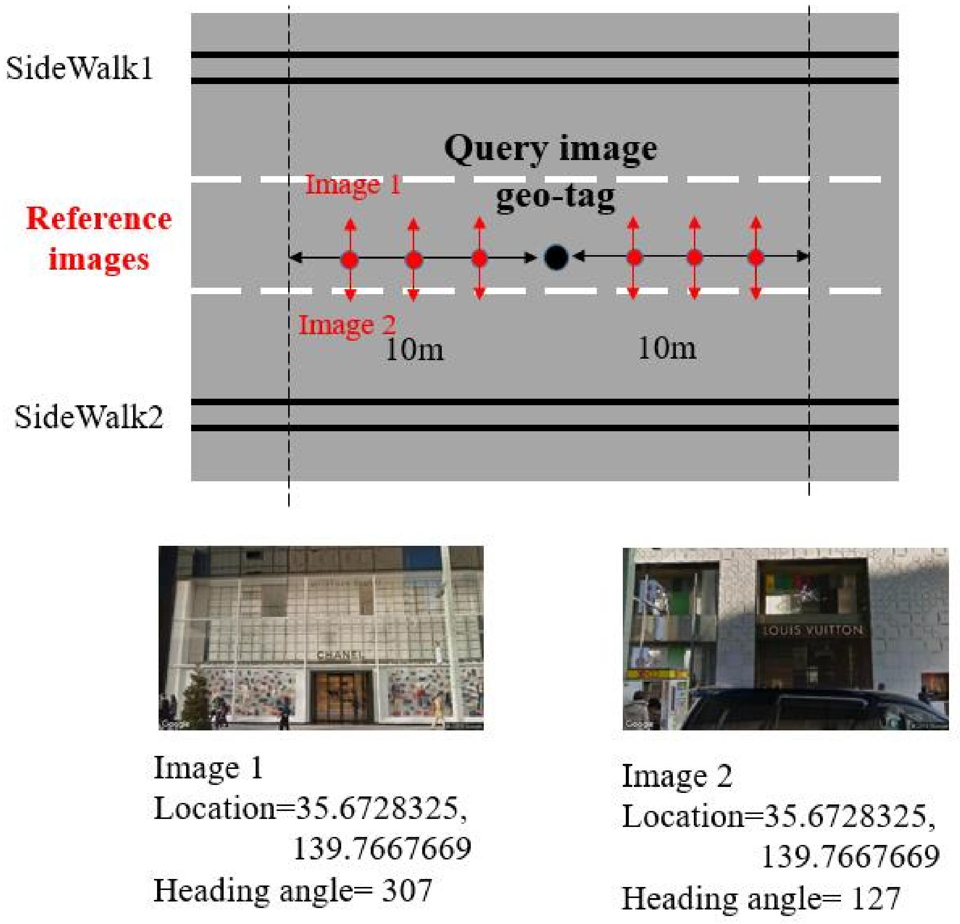


Figure 3.21 Illustration of the interested areas searching from the building facades datasets.

Once an image from the building facades datasets can be matched with the query image, the projected shape of the wall can be draw on the query image by using the

Homography matrix estimated from the matching result. Also, a reference virtual view will be generated in the center of the road near the searched interested area as shown in Figure 3.21.

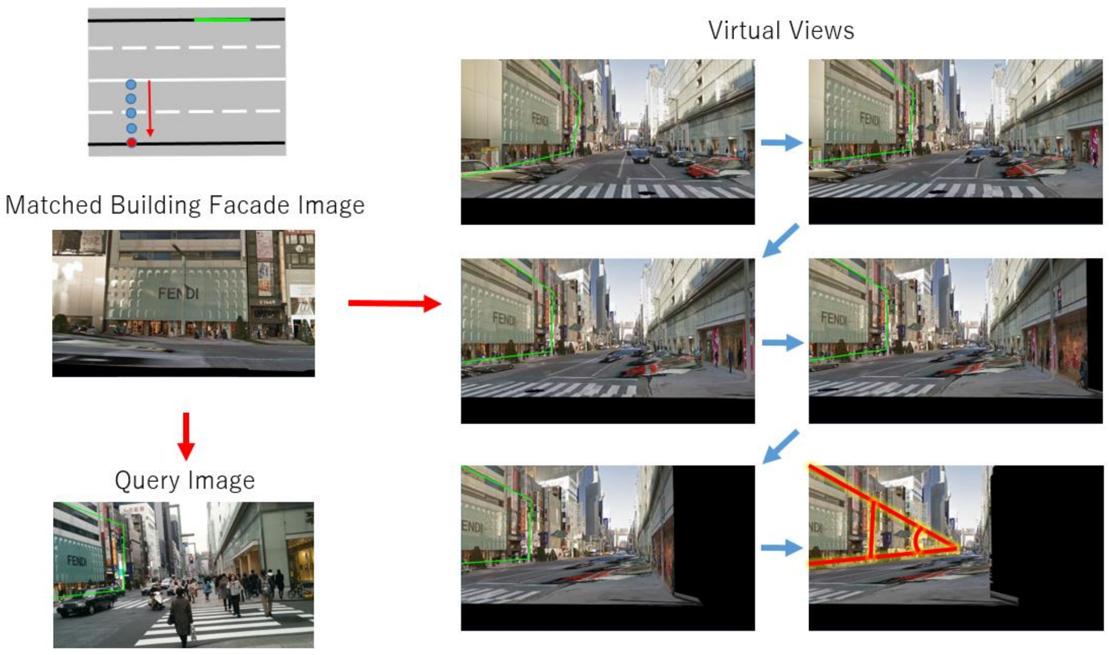


Figure 3.22 Interested area searching result and generated reference virtual view. The green windows shows the projection of the interested area.

From Figure 3.19, we can found when we move close to the interested area, the angle between the up and down edges and the size of the projection window will become bigger. And when we move far away from the interested area, the angle and the projection window will become smaller. Therefore, we can use the difference of this angle and the projection window size between the pedestrian query image and the reference virtual to move the camera alone the lateral direction of the road. When the angle and the up and down edges of the projection windows of a same interested area can become coincidence in the pedestrian query image and the generated virtual view, the lateral position of the pedestrian will be determined.

Then, how to decide which side of road and the distance need to move will be discussed in detail.

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 smartphone's 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. The algorithm is shown as Figure 3.23.

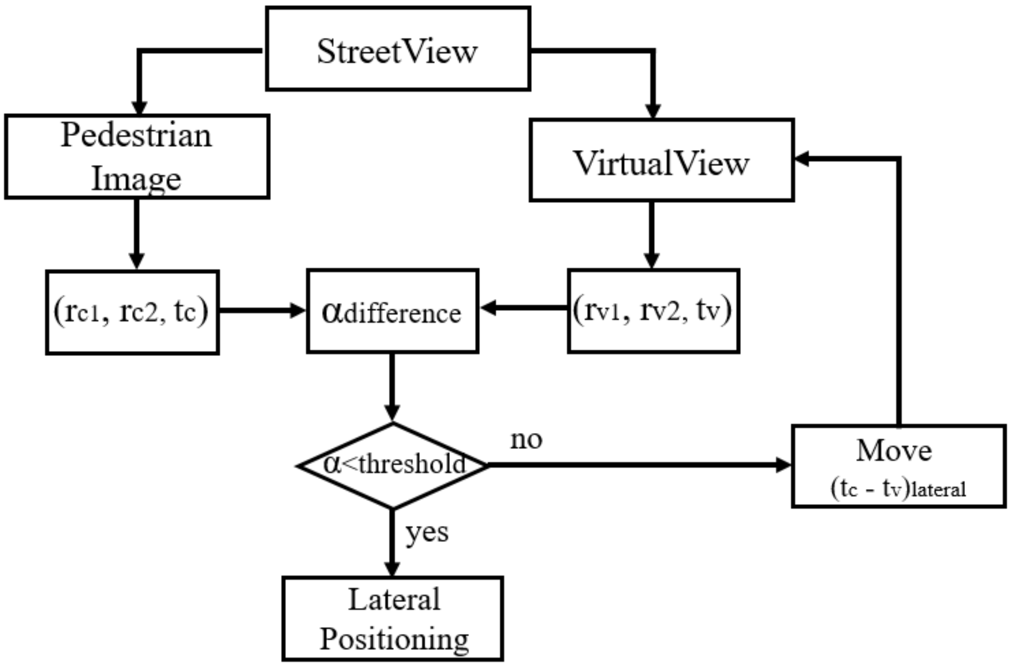


Figure 3.23 The flowchart of lateral position estimation.

As shown in Figure 3.23, we use the difference of this angle between the pedestrian query image and the reference virtual to decide if we need to move and use the difference between t_c and t_v to suggest which side of road and the distance we should move the camera alone the lateral direction of the road.

3.3.4. Longitudinal position estimation

After the lateral position of the pedestrian was determined, the final step is to determine the longitudinal position.

Similar with the lateral position estimation, the difference in the position of the projection windows between the query image and reference image can inform us which longitudinal direction of the road the camera in the virtual view should move to.

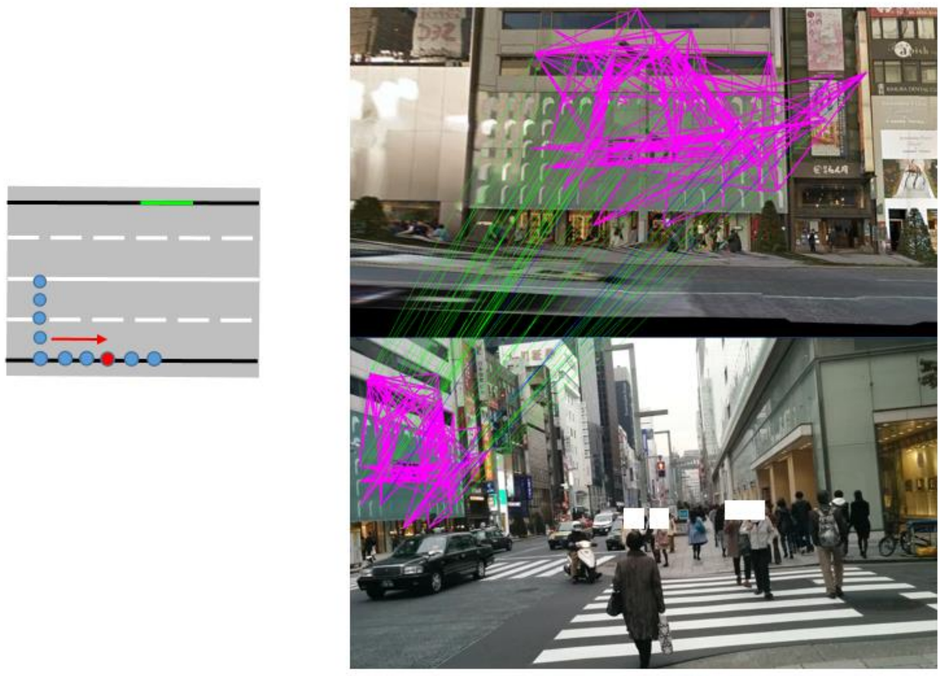


Figure 3.24 Longitudinal position estimation and kVLD based matching result.

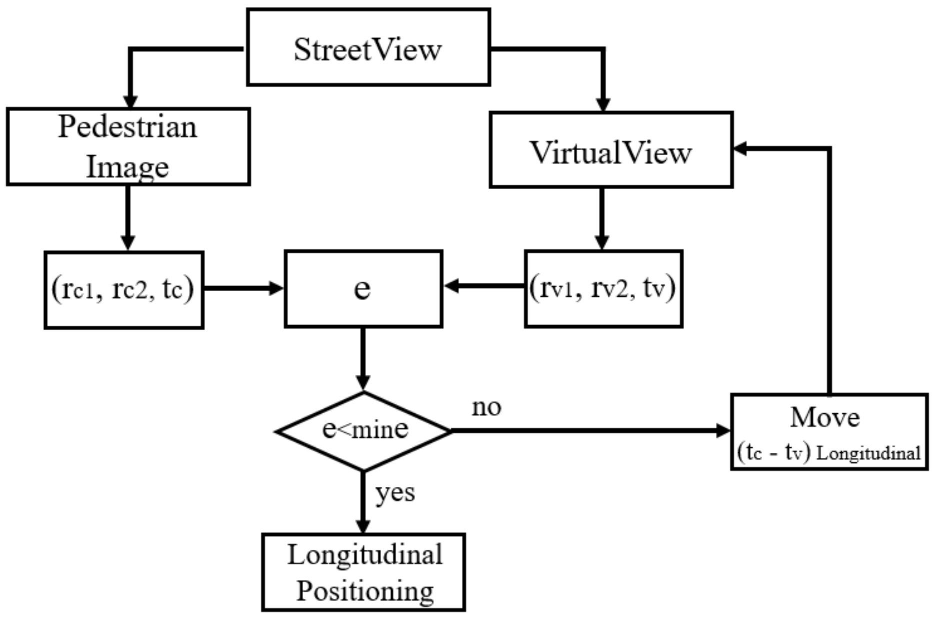


Figure 3.25 The flowchart of longitudinal position estimation

As shown in Figure 3.24, at this time, while the camera of virtual view is moving alone the longitudinal direction of the road, the generated virtual view will match with the query image using kVLD. And the error *e* will also be calculated. Finally, the position of the pedestrian will be determined when the minimum value of error *e* is found.

3.4. Experiment Result

This research 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 addition, we will focus on the research of pedestrian navigation in our future work. Shopping malls like Ginza is full of new visitors so a correct navigation service is need. Therefore, Ginza area just conform our research.

In the experiment, the smart phone Google Nexus 5 was used. The image matching was conducted based on images captured from the camera on this smart phone. To simulate the situation when a visitor is walking on the sidewalk in a new environment, we looked around while walking in the trajectory as shown in Figure 3.23 to capture images from the camera and collect GNSS positioning data simultaneously. The system time of the smart phone was used for synchronization. In pedestrian navigation, it is more important to distinguish which side of the road the pedestrian is on. This paper adopts the distance error and correct side rate to evaluate the performance of the proposed method.

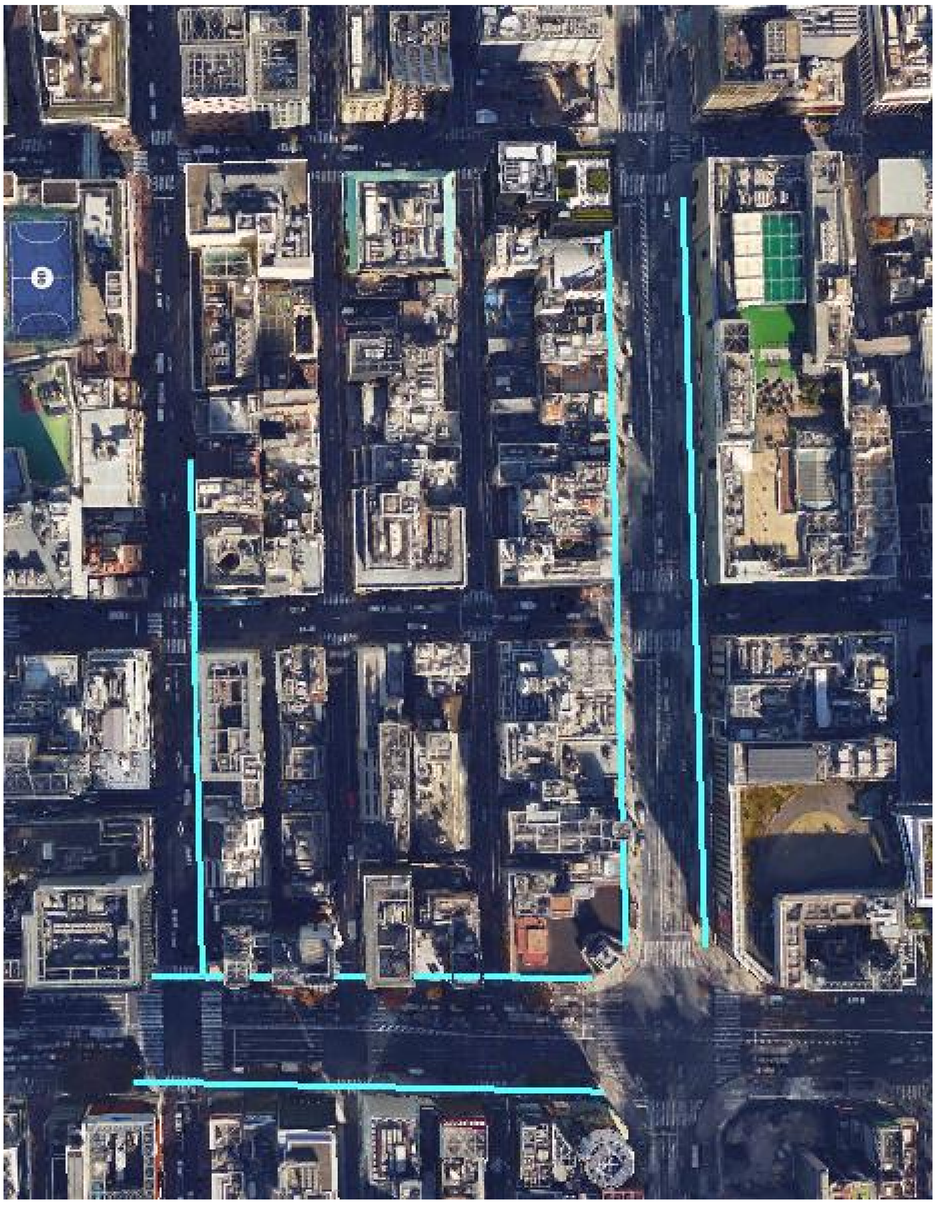


Figure 3.26 Walking trajectory of the experiments in Ginza

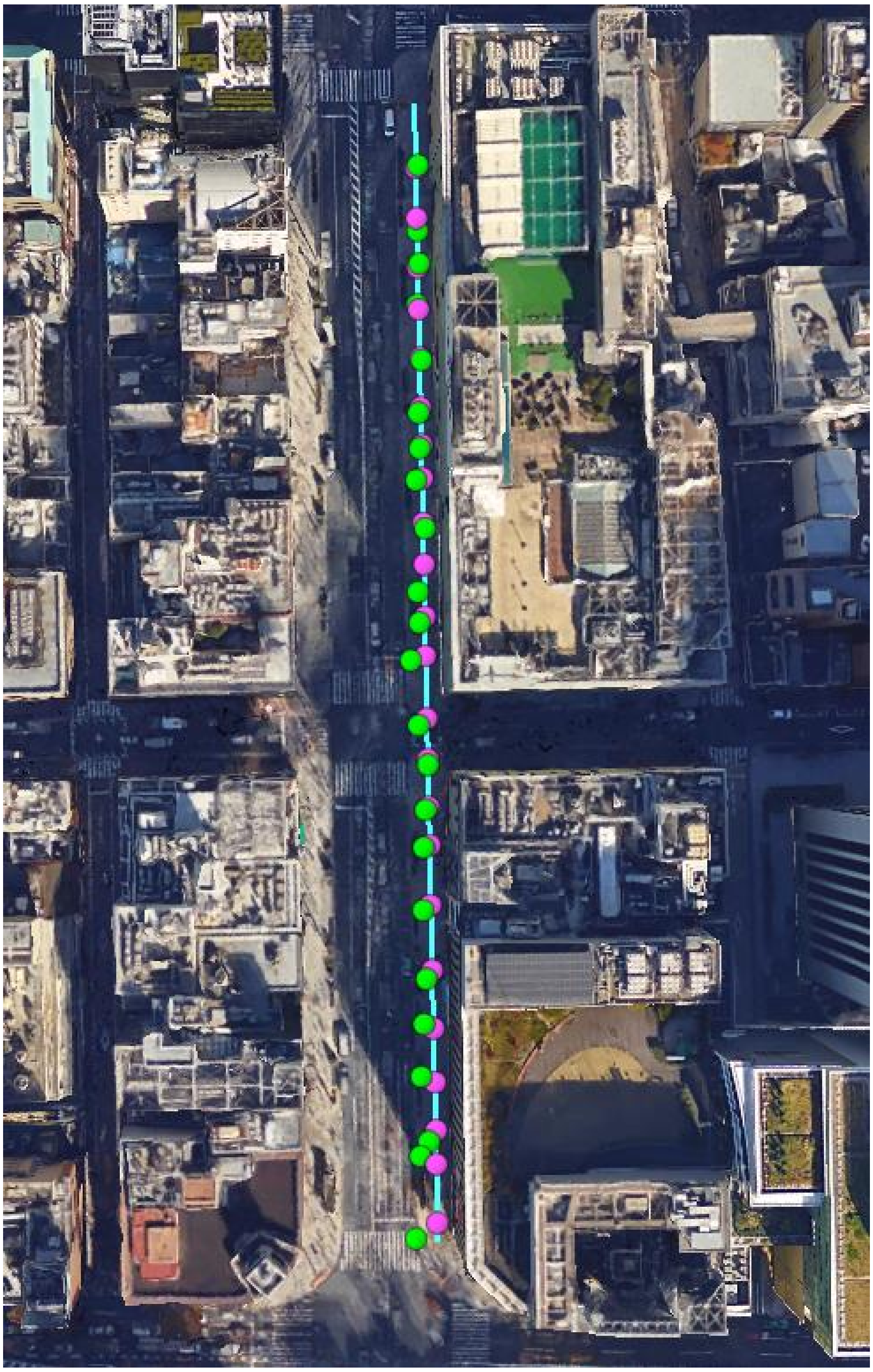


Figure 3.27 Visualization of results of route 1: walking trajectory (blue line), positioning result (green dot), and ground truth (purple dot).

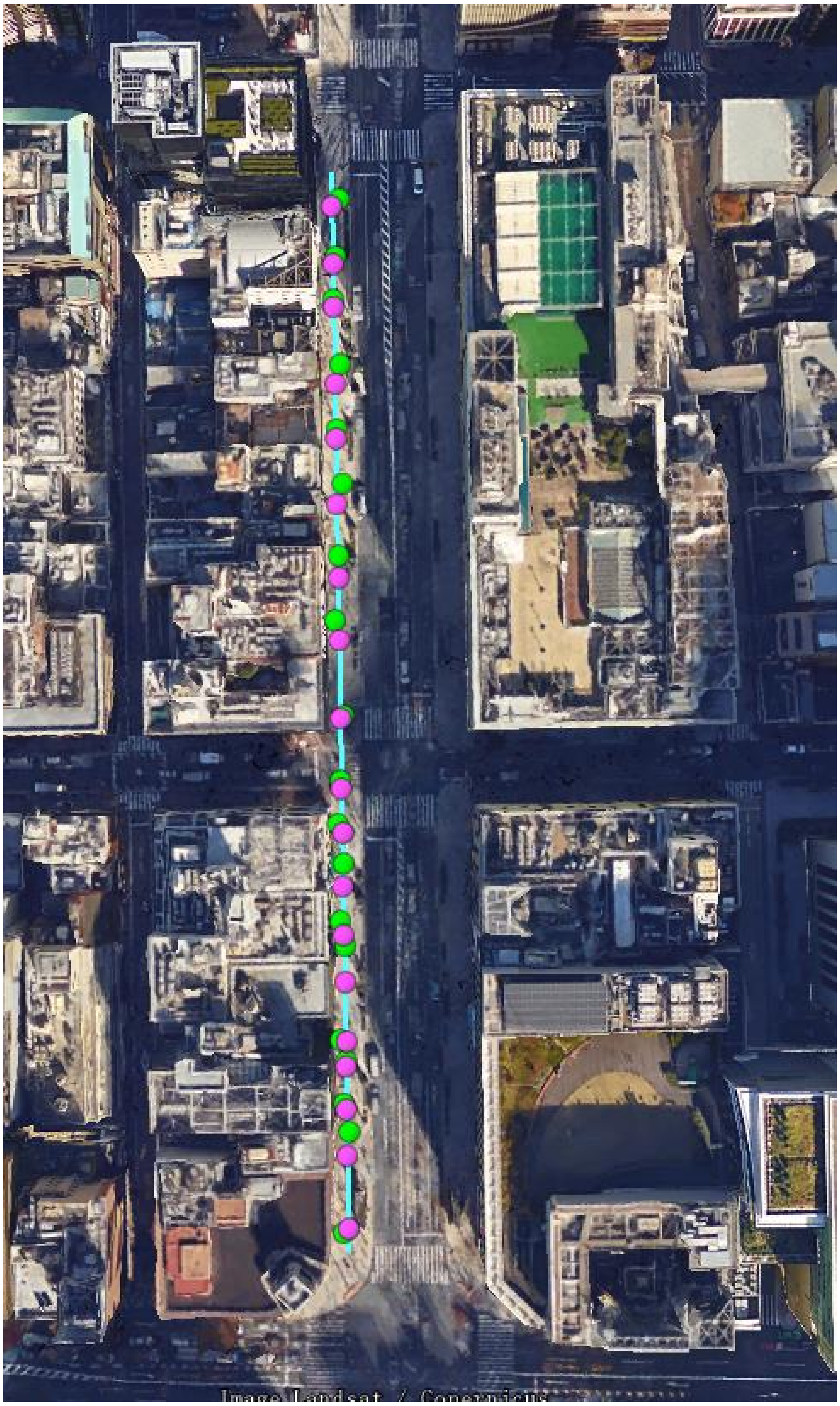


Figure 3.28 Visualization of results of route 2: walking trajectory (blue line), positioning result (green dot), and ground truth (purple dot).

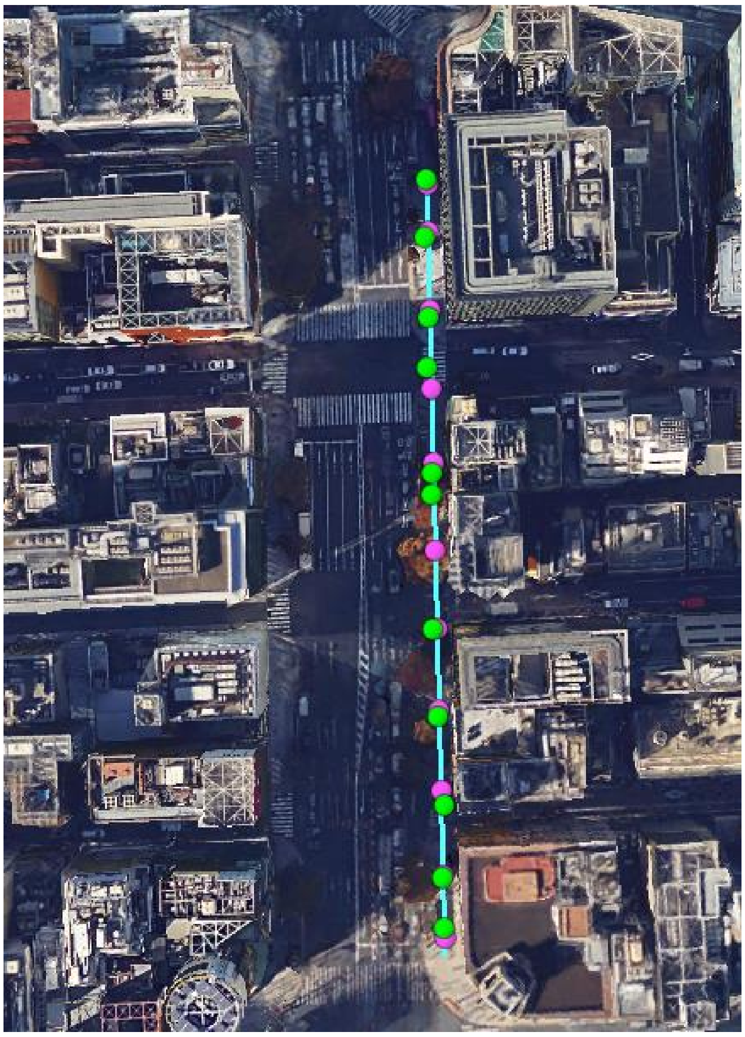


Figure 3.29 Visualization of results of route 3: walking trajectory (blue line), positioning result (green dot), and ground truth (purple dot).

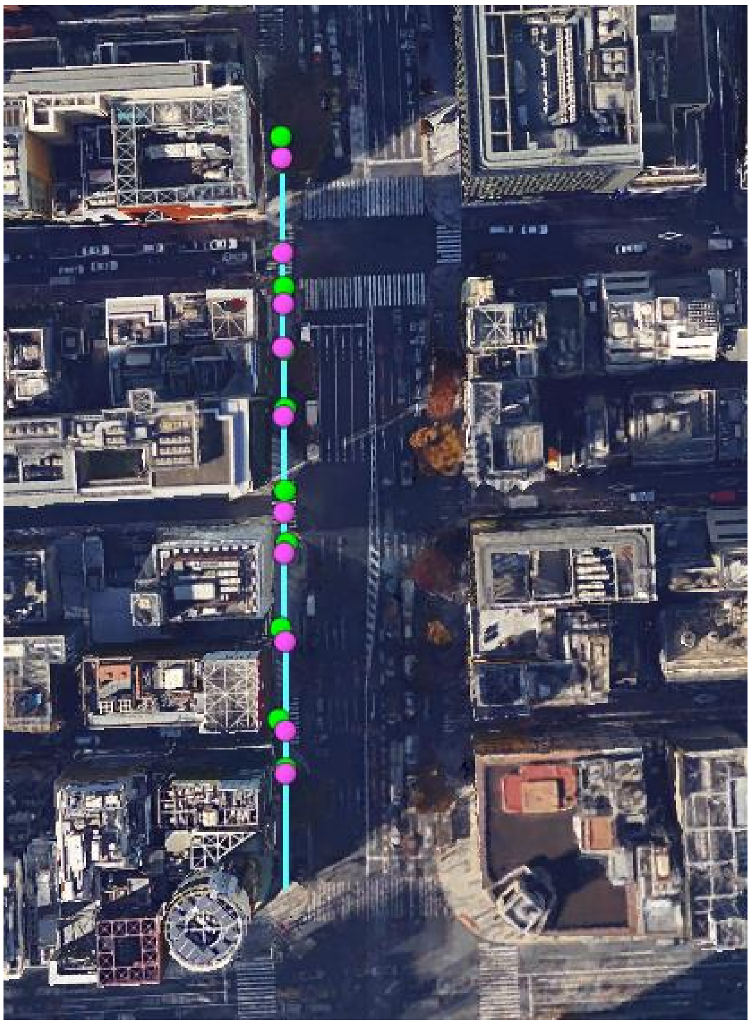


Figure 3.30 Visualization of results of route 4: walking trajectory (blue line), positioning result (green dot), and ground truth (purple dot).

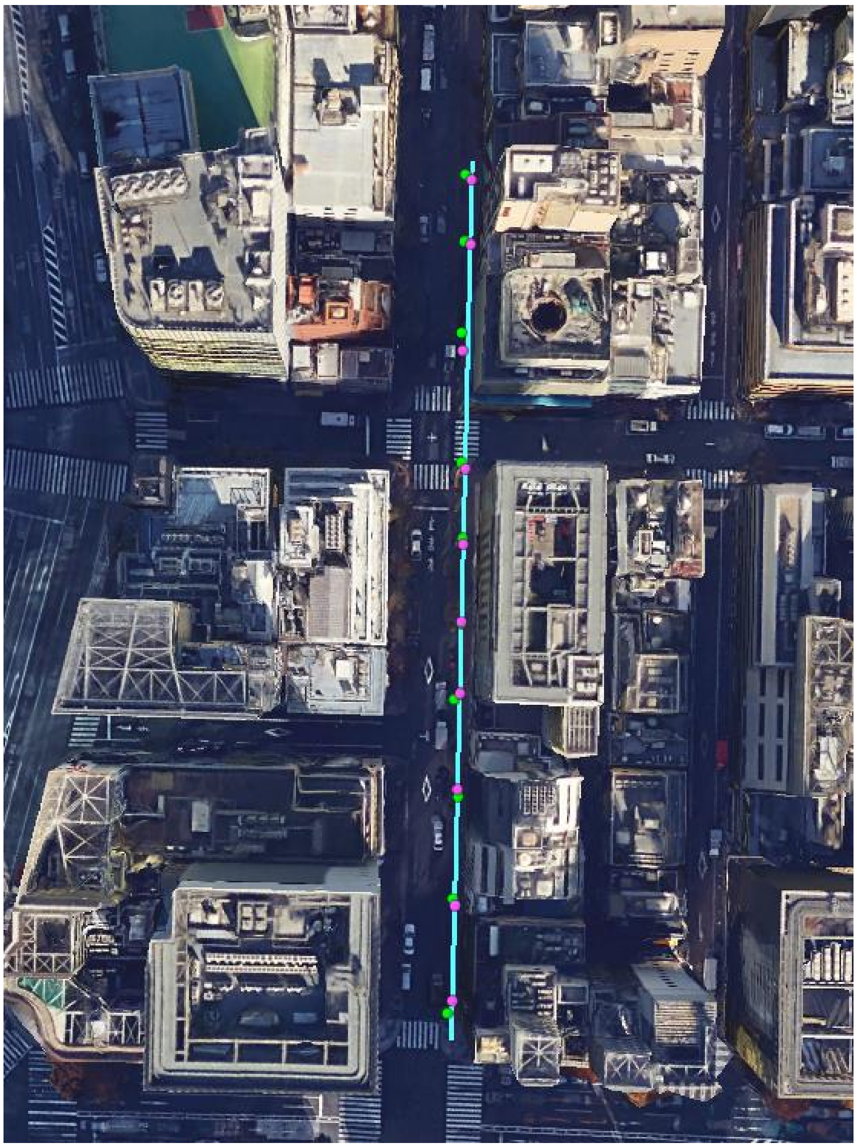


Figure 3.31 Visualization of results of route 5: walking trajectory (blue line), positioning result (green dot), and ground truth (purple dot).

Figure 3.24 to Figure 3.28 shows the positioning results in the experiment. The white line is the ground truth route and the purple dots are the ground truth positions where we captured images. The yellow line is the GNSS positioning result directly collected by the smart phone, from which we cannot distinguish which side of road we were. Green dots are the positioning result based on this paper.

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.

Table 1 Evaluation of the proposed method.

	Mean Error (meter)	Lateral Error (meter)	Longitudinal Error (meter)	Std (meter)	Positioning Availability
Route1	2.893	1.542	2.448	1.227	92.00%
Route2	3.233	1.259	2.914	1.450	86.36%
Route3	3.541	1.405	3.169	2.692	78.57%
Route4	3.550	1.406	3.223	1.191	76.92%
Route5	**1.859**	**0.763**	**1.638**	**0.607**	**90.91%**

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 error in the depth map from Google Maps Street View.

Route 3 and Route 4 are different sides of the Harumi road in Ginza as shown in Figure 3.32

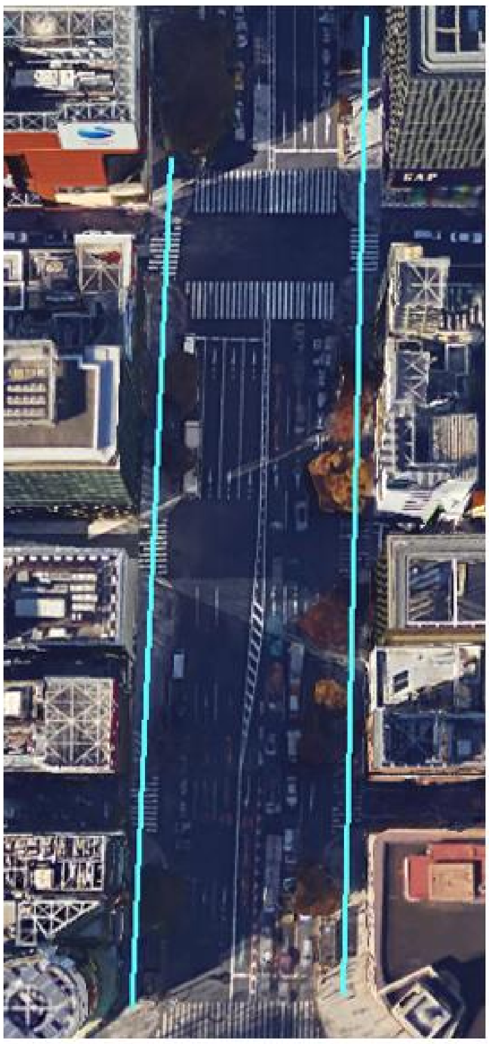
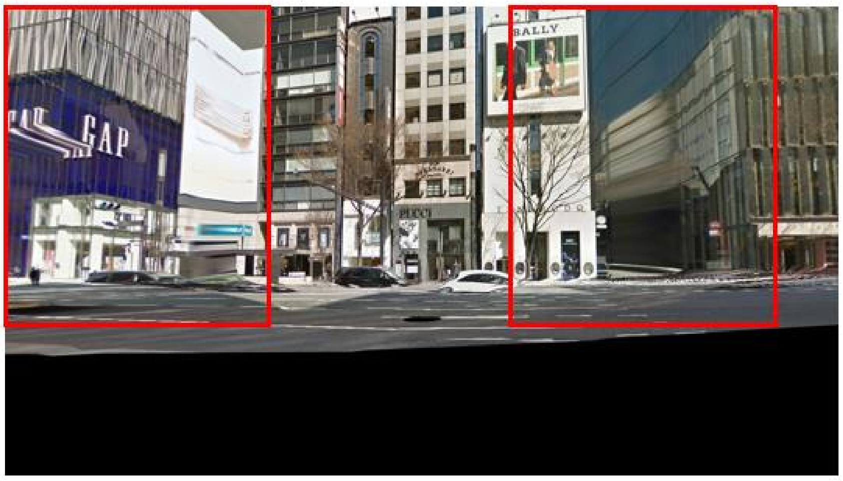


Figure 3.32 Route 3 and Route 4 are different sides of a same road—Harumi road in Ginza



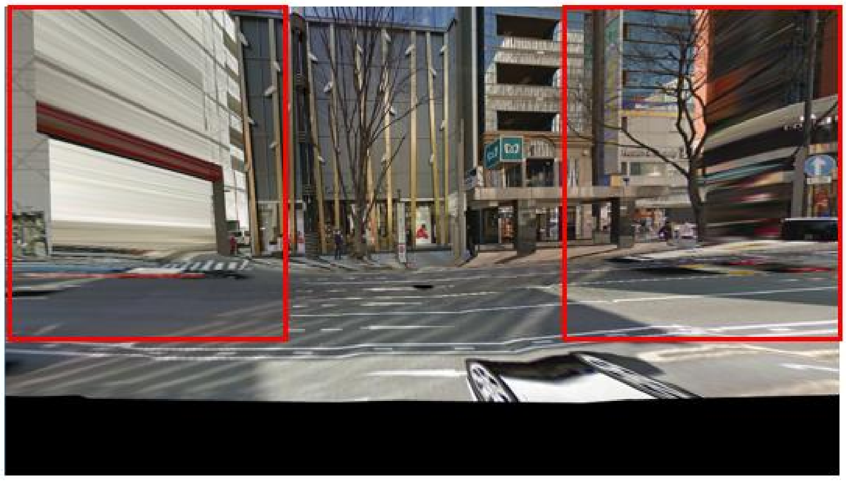


Figure 3.33 Errors in virtual view images

As shown in Figure 3.33, when there the depth map from Google Maps Street View is not correct in some place, there will be 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 finally make the query image hard to find a correct virtual view. Therefore, an accurate and reliable 3D map is required.

Also, Google Maps Street View only provide the daytime view in most of the roads. It makes our proposed method hard to provide the positioning and navigation in the night. Therefore, a reliable 3D map also need to prepare datasets in different situations.

Chapter 4.
Augmented Reality for Pedestrian Navigation

In this chapter, we will introduce how to provide guidance in the pedestrian view to guide the user to their destination.

4.1. Proposed method

4.1.1. Objective

To simplify the problem, we divide our 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.

4.1.2. Google Directions API

To find the route from the start point where the pedestrian is to the destination, we use Google Directions API.

The Google Maps Directions API is a service that calculates directions between locations using an HTTP request [33].

A Google Directions API request is an HTTP URL as:

http://maps.googleapis.com/maps/api/directions/outputFormat?parameters

Allowed parameters and their possible values are listed below：

Start position: The parameter *origin* specifies the start point to calculate the directions. The parameter can be input as the address, latitude and longitude coordinates, or place ID.

Destination position: The parameter *destination* specifies the start point to calculate the directions. Similar with the parameter *origin*, the parameter can be input as the address, latitude and longitude coordinates, or place ID.

Motion mode: The parameter *mode* specifies the mode of transport to use when calculating directions, including *walking*, *driving*, *bicycling* and *transit*.

Waypoints: The parameter *waypoints* specifies an array of waypoints. Waypoints alter a route by routing it through the specified locations.

Avoid: The parameter *avoid* indicates that the calculated routes should avoid the indicated features, including *tolls*, *highways*, *ferries* and *indoor*.

Access code: The parameter *key* is necessary to sign the URL. It can be easily applied for a free version key from the Google Street View Image API website.

By using the Google Directions API, we can set the start point, destination and motion mode. The API will provide us position of each checkpoint to the destination. Also text instructions on each part of the route will be provided.

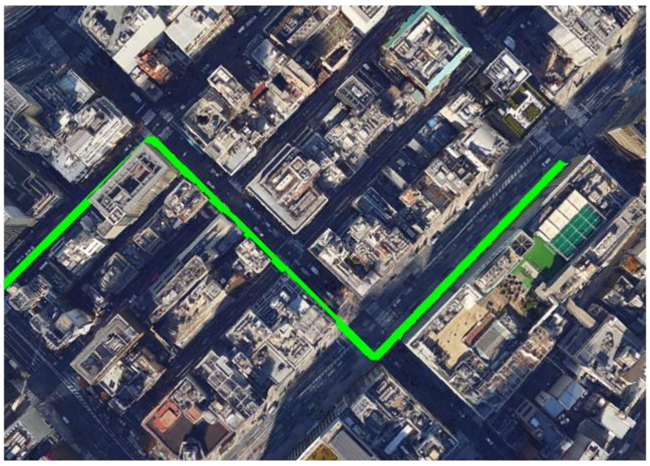


Figure 4.1 Sample of a route from the Google Directions API

Figure 4.1 shows a sample of route getting from Google Directions API in Ginza area.

4.1.3. Far from next checkpoint

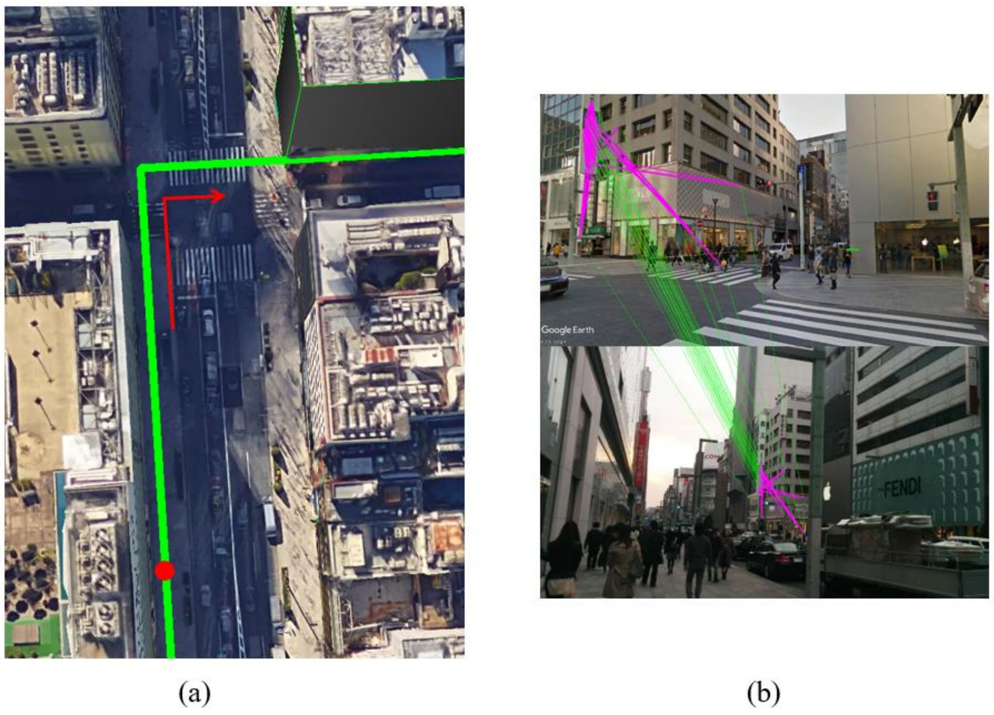


Figure 4.2 Situation when pedestrian is far from next checkpoint. (a) Current pedestrian position and the route; (b) match between the interested building and current pedestrian image.

Figure 4.2 shows the scene when pedestrian is far from next checkpoint. Red dot shows current pedestrian position, green lines is the route from Google Directions API, and red arrow is the direction need to go.

In this situation, the building in the right up corner in Figure 4.2 (a) will be choose as the interested building. When the interested building is found in pedestrian's scene, then we can draw an arrow to show where the pedestrian should turn.

And we also use the vanishing point to find the direction of the road and draw a arrow on the road to guide pedestrian which direction to go as shown in Figure 4.3.

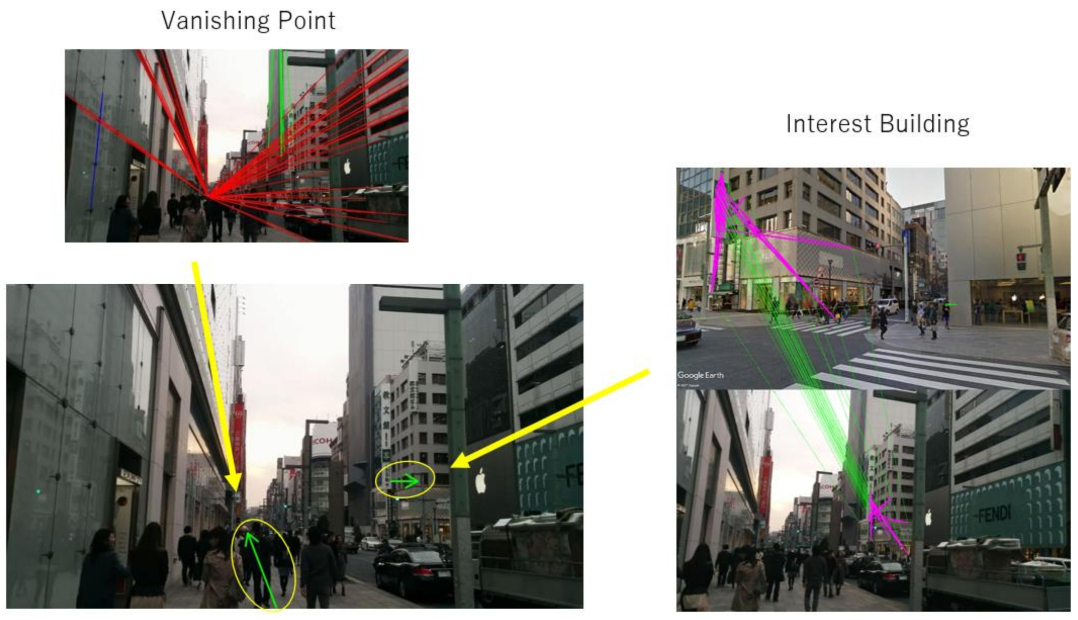


Figure 4.3 AR guidance when pedestrian is far from next check point

4.1.4. Near to next checkpoint

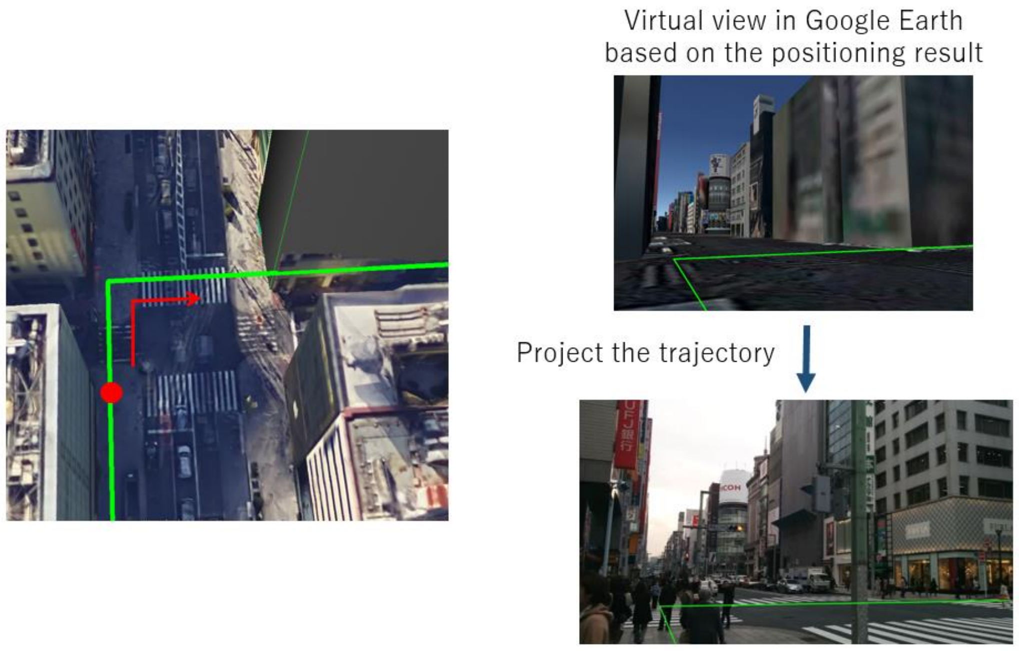

Virtual view in Google Earth based on the positioning result
Project the trajectory

Figure 4.4 Situation when pedestrian is near to next checkpoint.

As shown in Figure 4.4, when pedestrian is near to next checkpoint, we directly project the trajectory which is drawn in the virtual view to the pedestrian scene as the guidance.

4.2. Experiment Result

Demo of the AR navigation when the pedestrian is far from the checkpoint is shown as below.

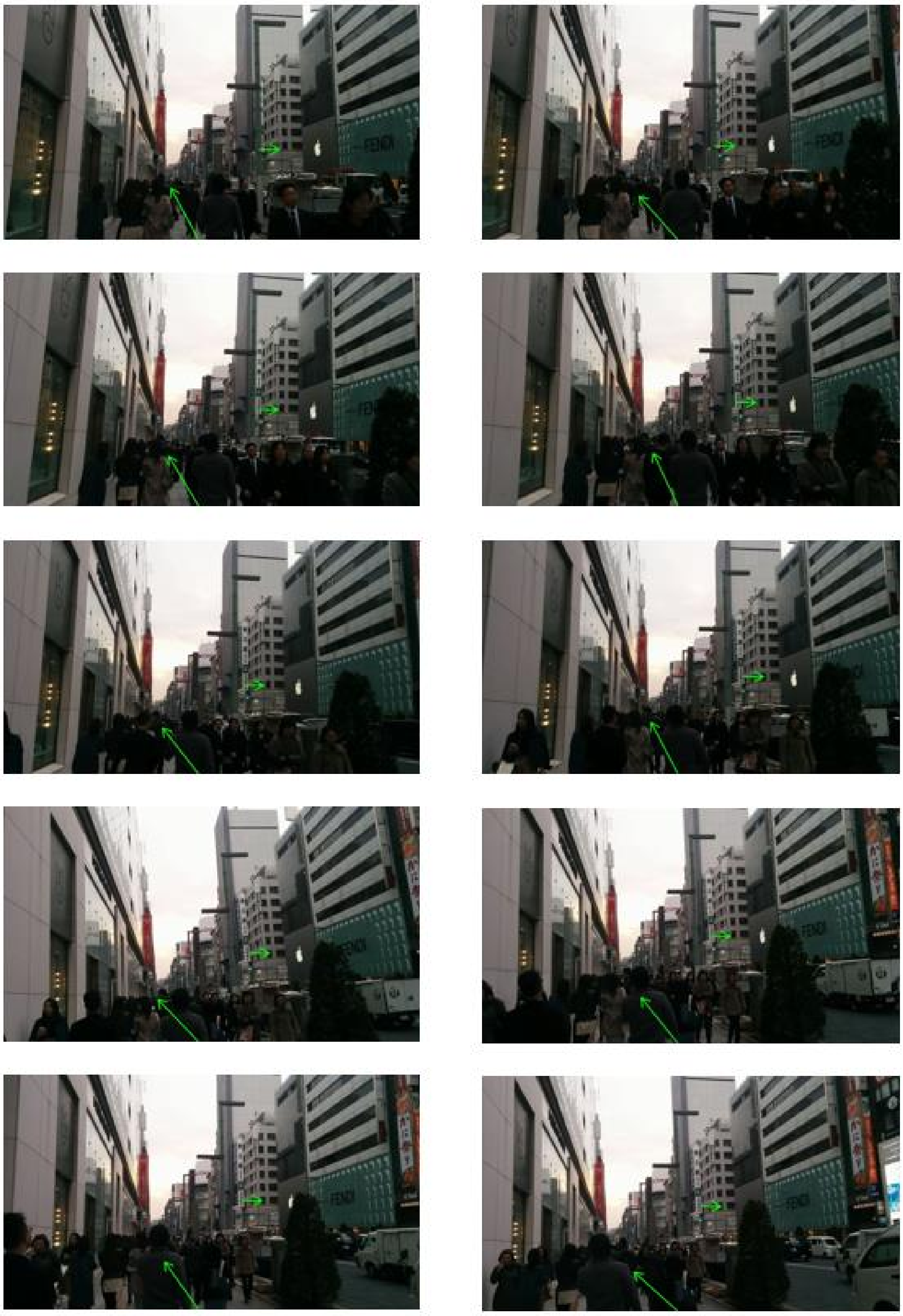


Figure 4.5 AR guidance when pedestrian is far from the checkpoint: scene 1, when the road and checkpoint are in the scene

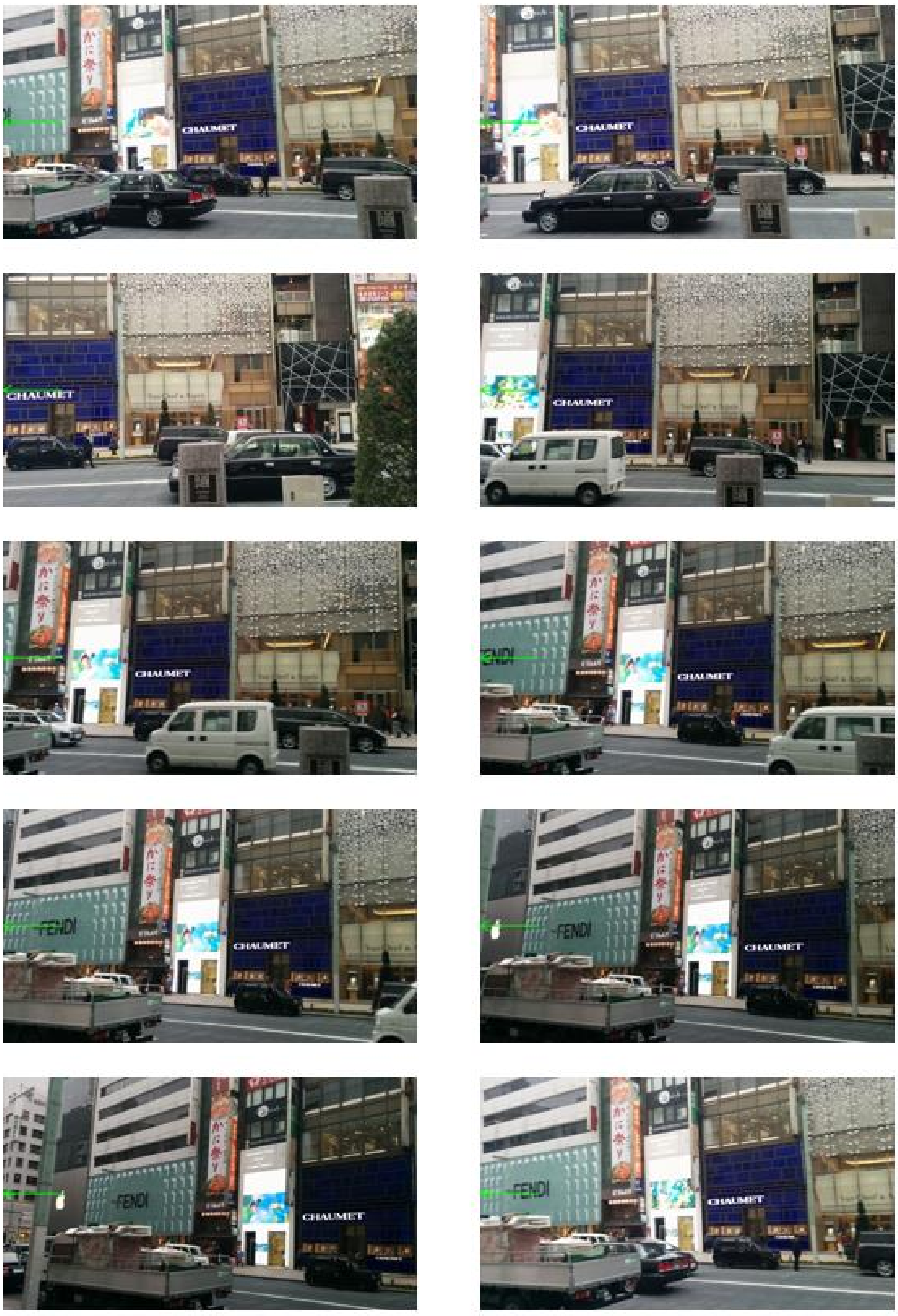


Figure 4.6 AR guidance when pedestrian is far from the checkpoint: scene 2, when the road and checkpoint are out of the scene

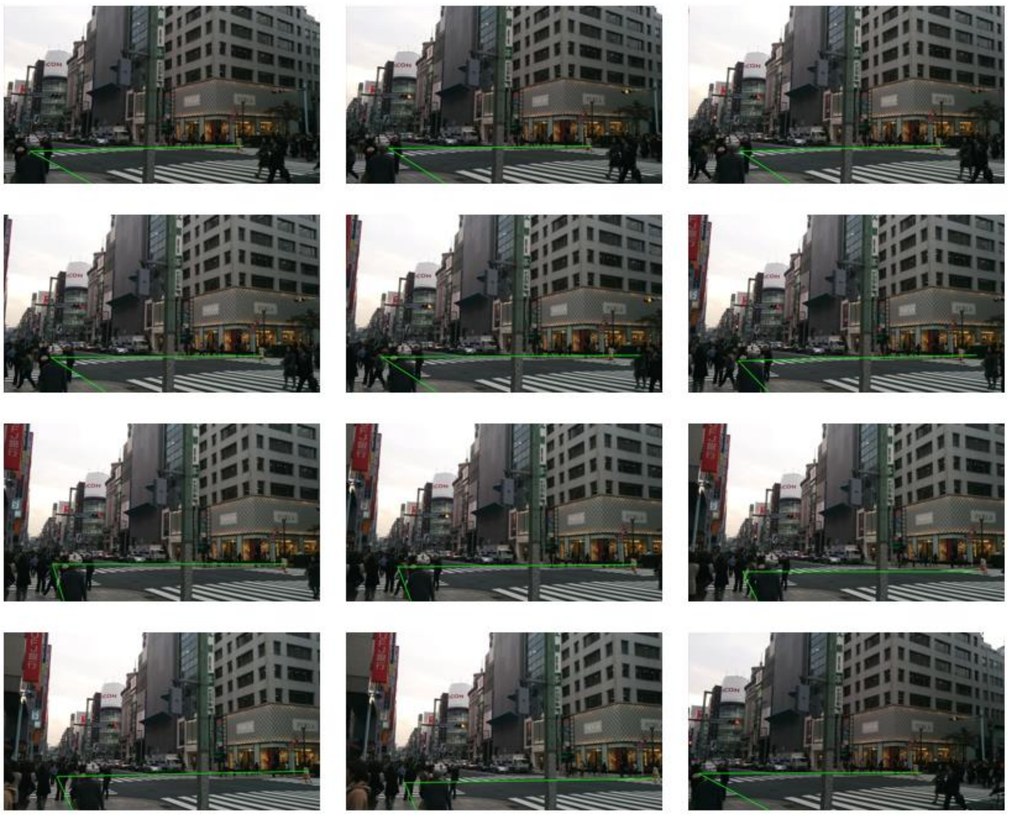


Figure 4.7 AR guidance when pedestrian is near to the checkpoint

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"

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, Professor Kamijo's 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 researchs. Nothing in this thesis can be achieved without their help. To the other members in the lab, Liu, Bao, Mahdi, Ehsan, Huang, Wada, Hashimoto, Xie, Prathana, Wang, I want to thank them for companying 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] Haitao Wang, Yanlei Gu, Shunsuke Kamijo, "Pedestrian Positioning with the aid of Google Earth and Google Maps Street View", 信学技報, vol. 116, no. 338, ITS2016-34, pp. 113-118, 2016 年 12 月.

国際学会:

[2] Yuyang Huang, Li-Ta Hsu, Yanlei Gu, Haitao Wang and Shunsuke Kamijo "Assessment of Outdoor Wi-Fi Fingerprint Calibration using Different GNSS Approaches" International Symposium on GNSS 2015 Kyoto, Japan. September 16-19, 2015.

[3] Haitao Wang, Yanlei Gu, Shunsuke Kamijo, "Pedestrian Positioning in City Urban with the Aid of Google Maps Street View", IAPR International Conference on Machine Vision Applications, 2017 (December 2016 submitted)