1-1-2016

Towards Autonomous Unmanned Vehicle Systems

Sheng Cai

Towards autonomous unmanned vehicle systems

By

Sheng Cai

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2016

Towards autonomous unmanned vehicle systems

By

Sheng Cai

Approved:

J. Patrick Donohoe
(Major Professor)

Sherif Abdelwahed
(Committee Member)

James E. Fowler
(Committee Member and Graduate Coordinator)

Pan Li
(Committee Member)

Jason M. Keith
Dean
Bagley College of Engineering

Name: Sheng Cai

Date of Degree: December 9, 2016

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: Dr. J. Patrick Donohoe

Title of Study: Towards autonomous unmanned vehicle systems

Pages of Study: 97

Candidate for Degree of Doctor of Philosophy

As an emerging technology, autonomous Unmanned Vehicle Systems (UVS) have found not only many military applications, but also various civil applications. For example, Google, Amazon and Facebook are developing their UVS plans to explore new markets. However, there are still a lot of challenging problems which deter the UVS's development.

We study two important and challenging problems in this dissertation, i.e. localization and 3D reconstruction. Specifically, most GPS based localization systems are not very accurate and can have problems in areas where no GPS signals are available. Based on the Received Signal Strength Indication (RSSI) and Inertial Navigation System (INS), we propose a new hybrid localization system, which is very efficient and can account for dynamic communication environments. Extensive simulation results demonstrate the efficiency of the proposed localization system. Besides, 3D reconstruction is a key problem in autonomous navigation and hence very important for UVS. With the help of high-speed In-

ternet and powerful cloud servers, the light-weight computers on the UVS can now execute computationally expensive computer vision based algorithms. We develop a 3D reconstruction scheme which employs cloud computing to perform realtime 3D reconstruction. Simulations and experiments show the efficacy and efficiency of our scheme.

Key words: UVS, Localization, Computer Vision, Cloud Computing

DEDICATION

To DingDingLa.

ACKNOWLEDGEMENTS

I want to thank my advisor Dr. J. Patrick Donohoe and Dr. Pan Li who gave me financial support and research guide to finish my PhD Dissertation. In addition, I want to thank our NEST group who gave me a lot of help. Furthermore, I want to thank my school and department that provided me with a comfortable environment to finish my PhD degree. I also want to thank my committees for their comments on this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the development of computer, network and sensor, Unmanned Vehicle Systems (UVS) are used in more and more important situations including environment monitoring [73], exploration of Mars [39] and industrial production [13]. Google developed and tested their unmanned car in the real roads [63]. Amazon has already used the robots to manage their warehouses [48], and in a short time, they will use the Unmanned aerial vehicles (UAV) to deliver the goods to the customs' houses [43]. Facebook started their great plan to deploy the solar powered UAVs to build the wireless network in the areas where the normal ways cannot be done [49]. To enable these important applications, many important technologies need to be developed and improved.

1.1 Overview

In this dissertation, we study two key problems for UVS. The first one is an adaptive localization system without using GPS, and the other is 3D reconstruction based on cloud computing. In particular, a good Localization system keeps UVS safe and controllable in complex environments. Without the UVS's position information, operations like navigation and tracking cannot be done. GPS is a well developed technology for this purpose. However, GPS has its own limits, and is not always working. To build a robust localization

system, the new technologies are needed. Besides, in order to enable UVS to navigate, track targets, etc, autonomously, 3D reconstruction needs to be performed efficiently at the UVS [39]. However, the high computational complexity limits the realtime 3D reconstruction at the UVS. With the help of cloud computing, the most expensive computation can be outsourced to the cloud, and 3D reconstruction can be enabled.

1.2 Organization of Disstation

In Chapter 2, we propose a new localization system. To the best of our knowledge, this method is the first to propose an efficient and adaptive localization system, which can adapt to dynamic communication environments quickly and effectively. Extensive simulation results demonstrate that the proposed localization system CRIL is able to track both slow changes and sudden changes of the channel model in dynamic environments. Besides, the proposed CRIL can perform accurate localization in the simulations with estimation errors up to $1\,\mathrm{m}$, while previous schemes' localization errors are up to several meters or even tens of meters.

In Chapter 3, we propose a 3D reconstruction scheme based on cloud computing. Basically, we can reconstruct scene geometry and camera motion from two or more images [22] in four steps [18]: track 2D features, estimate 3D points, optimize: bundle adjust and fir surfaces. Each step requires large computational resource, where is difficult to realize in small UVS. High speed Internet and cloud computing service give us a chance to get enough power to do this. A good outsourcing scheme can accelerate the whole processing, and enable the other high level tasks. We will continue investing at this problem.

In Chapter 4, we conclude this dissertation and introduce my future work. Basically, we propose a new localization system and a new 3D reconstruction outsourcing scheme to solve two key technology problems for UVS. In the future, we will complete our previous work, try to implement them on the real UVS, and show their values.

CHAPTER 2

CRIL: AN EFFICIENT ONLINE ADAPTIVE LOCALIZATION SYSTEM

2.1 Introduction

Indoor localization or indoor positioning systems find their use in many important applications including augmented reality [1], guided tours [26] (e.g., in museums, shopping malls), tracking and monitoring [15, 73], and situational awareness [19]. For example, social networks help people find friends at a party based on their location information [1]. To enable effective response in disaster rescue, accurate and reliable location information is indispensable as well. However, since GPS signals are usually poor in indoor environments, how to design an accurate indoor localization system is a challenging problem.

Due to the widespread adoption of wireless local area networks (WLANs) and mobile devices, WiFi signals easily become an alternative of GPS signals for indoor localization. Some works propose indoor localization techniques based on Angle of Arrival (AOA) [66], Time Difference of Arrival (TDOA) [77], Time of Arrival (TOA) [38], etc., of the WiFi signals. These methods require extra hardware to measure such data, which is not practical for common mobile devices like smart phones or tablets. Some other works develop schemes by taking advantage of the received signal strength indicator (RSSI), which can be easily obtained by almost every wireless device. Generally, RSSI based localization can be classified into two categories: *fingerprinting or mapping based schemes* [76, 72, 15, 73], and

channel modeling based schemes [36, 58, 67]. The first kind of schemes rely on building comprehensive data maps of RSSI, which requires significant efforts and needs recalibration whenever the environment changes. The second kind of schemes attempt to construct an accurate channel model to estimate the distances between a receiver and several known transmitters, then employ estimation algorithms such as the circular positioning and the hyperbolic positioning [67, 37] to find the receiver's location. Although the channel modeling based schemes do not require as much preparatory work as RSSI data mapping based schemes, they need an accurate channel model, which is difficult to get due to its dynamic nature in indoor environments. Several calibration methods such as [5, 7] are proposed to estimate the parameters in the channel model. Unfortunately, some of them like [5] are passive offline schemes, while the online schemes like [7] require much extra communication and computation cost.

One way to improve the performance of channel modeling based localization systems is to integrate it with an Inertial Navigation System (INS) [74]. In particular, an INS uses an Inertial Measurement Unit (IMU) to estimate an object's position with high update rate without any other side information. However, the error of the IMU usually gets accumulated very fast, especially for those cheap IMUs on mobile devices, which renders the stand-alone INS impractical. Previous works [12, 16, 80, 28, 68] propose to couple these two systems by data fusion technologies, but they cannot account for the dynamic channel models in dynamic communication environments.

In this paper, we propose an efficient coupled RSSI and INS localization system called CRIL, which can adapt to dynamic communication environments quickly and effectively.

Specifically, we employ a Kalman filter to fuse the localization results obtained from the RSSI and INS systems. Moreover, we introduce a recursive process in the update phase of the Kalman filter to update the parameters of the channel model utilizing the fused results. In so doing, our system can well model the dynamic communication channels in realtime without much additional calibration or overhead. In addition, we notice that there is always noise and measurement error in our localization system. To avoid unstable estimation results, we carefully design a stop mechanism to terminate the recursive process. Our recursive method of estimating parameters can be applied to other filters or data fusion technologies to model the changing of a subsystem's parameters. Furthermore, we have developed schemes to reduce outliers in and update the covariance of RSSI measurements. Our simulation and experiment study shows that the proposed CRIL system can 1) proactively track both gradual and abrupt changes in the channel model in realtime, 2) effectively account for uncertainties in RSSI measurements, and 3) lead to very small localization errors (on the scale of meters) compared to large errors (up to tens of meters) in previous schemes.

The rest of the paper is organized as follows. Section 2.2 reviews previous work on localization systems. In Section 2.3, we present the system models. Section 2.4 describes our proposed CRIL system, which is evaluated in Section 2.5 through extensive simulations and in Section 2.6 through real experiments. Finally, we conclude the paper in Section 2.7.

2.2 Related work

The rising interests in the indoor localization problems have drawn intensive attention. We introduce the most related work below.

Evennou and Marx [15] and Woodman and Harle [73] propose RSSI fingerprinting mapping based localization schemes. In particular, [15] develops a system where an accelerometer can count the number of walking steps of the user, a gyroscope can tell the orientation of the user, and the RSSI fingerprinting based scheme provides the location of the user. [73] does similar work, in which an RSSI mapping scheme is developed to find the location of a user.

Some previous works [53, 67, 37, 7, 5] design localization schemes through channel modeling utilizing RSSI. The path loss model with log-normal shadowing is commonly adopted. For instance, to build a more accurate model, Bernardos *et al.* [7] and Barsocchi *et al.* [5] formulate optimization problems to calibrate the channel model, which needs extra communication and time. As mentioned before, these schemes cannot account for dynamic communication channels, which is usually the case in indoor environments where both the objects under observation and the surrounding people/things may move around.

Data fusion algorithms have been designed to integrate RSSI systems with INS systems. [12, 16, 80] develop loosely-coupled estimation algorithms to fuse the two systems, which means that each of the two subsystem outputs an estimated position and the high level system fuses these results. In contrast, [28, 68] design tightly-coupled estimation algorithms to fuse both systems, i.e., the high level system will fuse the measurements (angular, velocity, RSSI, etc.) directly from the two subsystems to calculate the final position

and the subsystems will not output their own estimated positions. The former estimation algorithms have lower computational complexity and easy to be designed, while the latter estimation algorithms have better performance. Our proposed CRIL system takes advantage of both of them and can have better performance.

In addition, there are some other localization systems such as [28, 54, 27] which utilize both Ultra-Wide Band (UWB) systems and INSs. The use of UWB enables the measurement of AOA, TDOA, TOA, etc., but currently UWB radio interfaces are not very common on mobile devices and would be more expensive.

2.3 System Models

2.3.1 System Architecture

As shown in Fig. Figure 2.1, we consider an indoor environment where we intend to track a moving pedestrian's (or object's) two-dimensional location on a certain floor of a building[1]. There are a number of WiFi anchors in known positions in this space. The object carries a mobile device with a WiFi radio and an IMU. Thus, the object can receive WiFi signals from the anchors and can measure the RSSI of each of the signal. More importantly, we consider a dynamic indoor environment, which implies the dynamic nature of the communication channels therein.

2.3.2 Channel model

One of the most commonly used channel models is the log-normal path loss model [53], which has a direct relationship between the distance and the received signal strength.

[1] In this paper, we use pedestrian and object interchangeably.

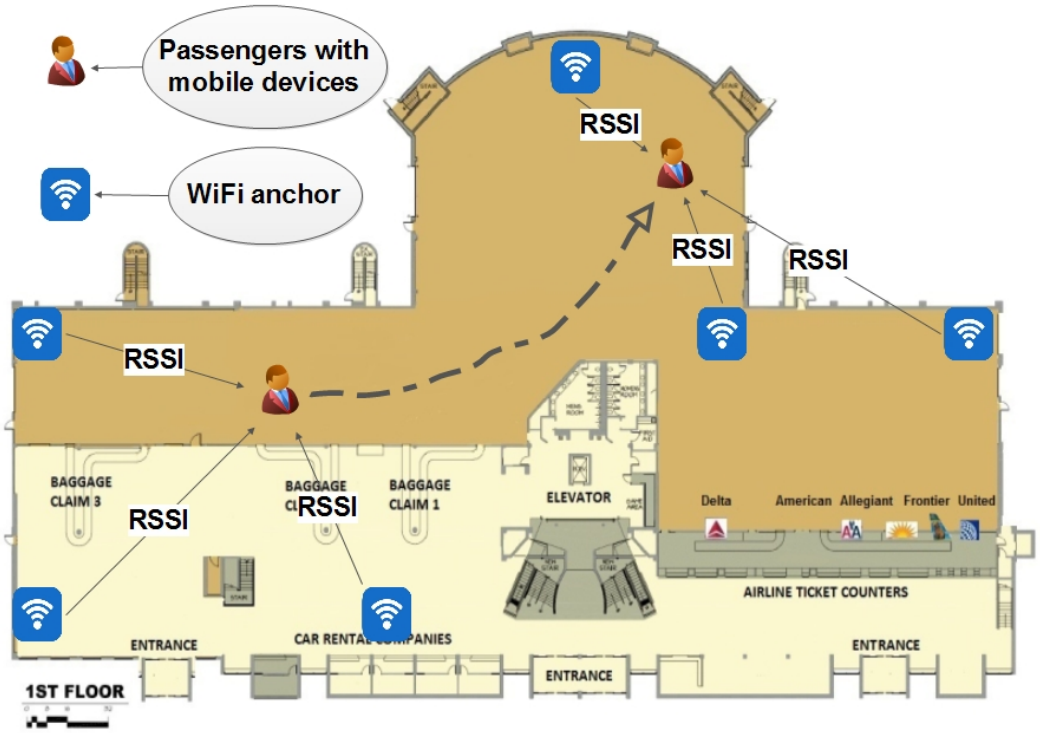

Figure 2.1

A typical scenario of indoor localization.

Specifically, the RSSI (in dBm) of a signal received at a receiver, denoted by P_{RX}, can be calculated as:

$$P_{RX}(dBm) = A - 10\eta \log_{10}(d/d_0) + N_0 \tag{2.1}$$

where A is the received signal strength at a reference distance d_0, η is the path loss exponent, and N_0 is the noise in the environment. The value of A depends on the transmitted signal power P_{TX} and the antenna gains of the transmitter and the receiver. The noise N_0 is usually defined as a zero-mean Gaussian random variable $\mathcal{N}(0, \sigma)$.

Rewriting (2.1), the distance d between the transmitter and the receiver can be calculated by:

$$d = d_0 \cdot 10^{\frac{A - P_{RX} + N_0}{10\eta}}. \tag{2.2}$$

Note that η typically varies between 2 and 4 in outdoor environments, and can range from 4 to 6 in indoor environments [67].

Obviously, time-consuming experiments need to be conducted in order to calibrate the value of η before we use this channel model to estimate the distance d. The overhead becomes more intolerable in dynamic environments where channel models constantly change due to its sensitivity to surrounding movements, temperature, air pressure, air moisture, etc. [52]. Therefore, a realtime calibration process is indispensable to track the changes in the path loss exponent η and hence perform more accurate localization.

2.3.3 Observation Model

The observation model defines the relation between the measurements and the actual states of the observed object. The proposed CRIL system includes two subsystems: an

RSSI localization system and an INS localization system, both of which will output their observation results, i.e., measurement results. Let $\boldsymbol{Z}$ denote the measurement of the observed object's location $\boldsymbol{X} = [x, y]^\top \in \Re_{2\times 1}$. Note that the object's position in the z-axis is denoted by z_* and does not change, which does not need to be estimated. Then, we can have the following observation model:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}_{INS} \\ \\ \boldsymbol{Z}_{RSSI} \end{bmatrix} = \boldsymbol{C}\boldsymbol{X} + \boldsymbol{\Upsilon} \tag{2.3}$$

where $\boldsymbol{Z}_{INS}$ and $\boldsymbol{Z}_{RSSI}$ are the measurements of the object's position from the INS system and RSSI system, respectively, and $\boldsymbol{\Upsilon}$ is the measurement noise. Note that $\boldsymbol{C}$ is the observation matrix which is defined as:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} \\ \\ \boldsymbol{I}_{2\times 2} \end{bmatrix} \tag{2.4}$$

where $\boldsymbol{I}$ is the identity matrix.

As shown in (2.3), the observation error affects the measurement results. Thus, the object's position cannot be directly obtained, but need to be estimated based on the measurements.

2.4 CRIL: A Coupled RSSI and INS Localization System

In this section, we detail our proposed online adaptive localization system: coupled RSSI and INS localization (CRIL). In particular, the RSSI system has bounded error but low accuracy, while the INS has high accuracy in the short run but large drift error in the long run. Thus, CRIL couples these two systems in order to obtain better localization

performance. One salient feature of CRIL is that it can fuse the results from RSSI and INS and in return update the channel model in RSSI in realtime. In so doing, CRIL can quickly and efficiently track the dynamic channel model, and better fuse the results from RSSI and INS to provide more accurate localization results. In what follows, we first describe the RSSI and INS systems, and then the proposed CRIL localization system.

2.4.1 RSSI Localization System

Theoretically, we can determine the location of the object based on the distances between the object and three anchors through triangulation algorithms. However, because of the inaccuracy of the measurement results, the triangulation algorithms cannot be used directly. In the literature, there are mainly two kinds of estimation algorithms that address the inaccuracy problem: the circular positioning algorithm [67], and the hyperbolic positioning algorithm [37]. The circular positioning algorithm minimizes the sum of the squared errors between the real and estimated distances from the tracked object to the different chosen anchors. The hyperbolic positioning algorithm uses a least squares estimation method to solve a linear problem in order to estimate the object's position. Although the circular positioning algorithm has a better performance than the hyperbolic positioning algorithm, it has much higher computation cost. Considering that the object is usually a mobile device with limited computation capability and energy, we employ the hyperbolic positioning approach.

Specifically, consider that we choose N anchors, and anchor 1 is at the reference spot $(0, 0, 0)$. The square of the distance between the object and an arbitrary anchor i ($1 \leq i \leq N$), denoted by d_i^2, can be expressed as:

$$d_i^2 = (x_i - x)^2 + (y_i - y)^2 + (z_i - z_*)^2. \tag{2.5}$$

Therefore, subtracting (2.5) when $i = 1$ from that when $i \neq 1$, we get

$$d_i^2 - d_1^2 = x_i^2 - 2x_i x + y_i^2 - 2y_i y + z_i^2 - 2z_i z_*. \tag{2.6}$$

Then, we rewrite (2.6) in the following matrix form:

$$\begin{bmatrix} 2x_2 & 2y_2 \\ \vdots & \vdots \\ 2x_N & 2y_N \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_2^2 + y_2^2 + z_2^2 - 2z_2 z_* - d_2^2 + d_1^2 \\ \vdots \\ x_N^2 + y_N^2 + z_N^2 - 2z_N z_* - d_N^2 + d_1^2 \end{bmatrix}. \tag{2.7}$$

Because of the measurement noise and errors, we can only have the estimated distances of d_i, denoted by $\tilde{d}_i$, according to (2.2). Thus, (2.7) becomes

$$\boldsymbol{H}\hat{\boldsymbol{X}} = \tilde{\boldsymbol{b}} \tag{2.8}$$

where

$$\boldsymbol{H} = \begin{bmatrix} 2x_2 & 2y_2 \\ \vdots & \vdots \\ 2x_N & 2y_N \end{bmatrix}$$

$$\tilde{\boldsymbol{b}} = \begin{bmatrix} x_2^2 + y_2^2 - \tilde{d}_2^2 + \tilde{d}_1^2 \\ \vdots \\ x_N^2 + y_N^2 - \tilde{d}_N^2 + \tilde{d}_1^2 \end{bmatrix}$$

and $\hat{\boldsymbol{X}} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$ is the estimated position of the object.

The least squares solution to this equation is:

$$\hat{\boldsymbol{X}} = (\boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{H}^T \tilde{\boldsymbol{b}}. \tag{2.9}$$

Note that this least squares solution (2.9) assigns the same weight to different estimated distances $\tilde{d}_i^2$. However, since the channel model (2.1) is nonlinear, the same gaussian distributions of the RSSI measurement error will lead to different distributions of the distance measurement error in the case of different transmission distances. Intuitively, the larger the distance d_i, the larger is the distance error caused by the same RSSI error. We employ a weighted hyperbolic algorithm [67] to solve this issue. The algorithm assigns larger weights to those estimated distances with higher accuracy, under the assumption that the shorter estimated distances have better accuracy as mentioned above. The modified weighted least squares solution is as follows:

$$\hat{\boldsymbol{X}} = (\boldsymbol{H}^T \boldsymbol{S}^{-1} \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{S}^{-1} \tilde{\boldsymbol{b}}. \tag{2.10}$$

Here $\boldsymbol{S}$ is the estimated covariance matrix of $\tilde{\boldsymbol{b}}$, which can be estimated by

$$\boldsymbol{S} = \begin{bmatrix} \tilde{d}_1^4 + \tilde{d}_2^4 & \tilde{d}_1^4 & \cdots & \tilde{d}_1^4 \\ \tilde{d}_1^4 & \tilde{d}_1^4 + \tilde{d}_3^4 & \cdots & \tilde{d}_1^4 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{d}_1^4 & \tilde{d}_1^4 & \cdots & \tilde{d}_1^4 + \tilde{d}_N^4 \end{bmatrix}. \tag{2.11}$$

Note that N should be at least 3 to conduct the least squares estimation in order to have a fairly good estimation of $\boldsymbol{X}$. The complexity of this algorithm is low since N can be a small number in real implementations.

2.4.2 Inertial Navigation System

An Inertial Navigation System (INS) uses an Inertial Measurement Unit (IMU) to estimate positions and is widely used as the navigation system for airplane, ships, rockets, etc. [74]. Advances of the Micro Electro Mechanical System (MEMS) technology lead to cheap and small IMUs for common mobile devices like smart phones and tablets. The main advantage of IMUs is that they need no external inputs for measuring their positions and can be used in indoor environments or wherever satellite signals are not available.

Normally, an IMU includes a gyroscope and an accelerometer. In particular, the gyroscope outputs the angular velocity of the object and the accelerometer outputs the linear acceleration. The angular velocity of an object gives its orientation and attitude. Based on the orientation, attitude, and other information, we can transfer the linear acceleration in inertial reference coordinates into navigation reference coordinates. Then, based on the

Newton's laws of motion, the position of the object can be calculated after two integrations. More detailed descriptions on INS systems can be found in [74].

An IMU position estimation system can be modeled by the following linear equations [16]:

$$\bar{\boldsymbol{\phi}}_{j+1} = \bar{\boldsymbol{\phi}}_j + \bar{\boldsymbol{W}}_{j+1}\Delta t + \boldsymbol{\Gamma}_{\boldsymbol{\phi}} \tag{2.12}$$

$$\bar{\boldsymbol{V}}_{j+1} = \bar{\boldsymbol{V}}_j + \bar{\boldsymbol{A}}_{j+1}\Delta t + \boldsymbol{\Gamma}_{\boldsymbol{V}} \tag{2.13}$$

$$\bar{\boldsymbol{X}}_{j+1} = \bar{\boldsymbol{X}}_j + \bar{\boldsymbol{V}}_{j+1}\Delta t + \boldsymbol{\Gamma}_{\boldsymbol{X}} \tag{2.14}$$

where Δt is the IMU's sampling period, and $\bar{\boldsymbol{\phi}}$, $\bar{\boldsymbol{W}}$, $\bar{\boldsymbol{A}}$, $\bar{\boldsymbol{V}}$, and $\bar{\boldsymbol{X}}$ are the attitude vector, angular velocity vector, acceleration vector, velocity vector, and position vector, respectively, in the navigation reference coordinates. $\boldsymbol{\Gamma}_{\boldsymbol{\phi}}$, $\boldsymbol{\Gamma}_{\boldsymbol{V}}$, and $\boldsymbol{\Gamma}_{\boldsymbol{X}}$ are the noises in $\bar{\boldsymbol{\phi}}$, $\bar{\boldsymbol{V}}$, and $\bar{\boldsymbol{X}}$, respectively, and assumed to follow Gaussian distribution.

One problem of the INS system is that the accumulated error, which is well-known as INS drift, can increase very fast as time goes by. To be more prominent, as shown in equations (2.12)-(2.14), the error introduced in the angular velocity measurement will be propagated into the estimated attitudes, and the error introduced in the linear acceleration measurement will be propagated into both the estimated velocity and the estimated objects' position. Moreover, since the $(j+1)$th estimated position is based on the previous estimated position $\bar{\boldsymbol{X}}_j$, the previous errors of the INS system will be accumulated into the future position estimations [74]. Therefore, after a few position updates, the accumulated errors may become non-eligible. This problem becomes even more serious in MEMS IMUs.

The reason is that the thermo-mechanical white noise and the bias errors account for a significant fraction of the measurement error [74].

To address the above problem, we propose to employ the Pedestrian Dead-Reckoning (PDR) method [64, 60, 40], which obtains a pedestrian's position based on the number of steps, step length, and orientation. There are three processes in PDR: step detection, stride length estimation, and orientation estimation.

2.4.2.1 Step detection

In this process, the algorithm counts the number of steps by detecting the number of swing phases. Particularly, we first calculate the magnitude of the acceleration a_{j+1} with the three-axis accelerometer[60]:

$$a_{j+1} = \sqrt{a_{x_{j+1}}^2 + a_{y_{j+1}}^2 + a_{z_{j+1}}^2}, \tag{2.15}$$

and then identify a swing phase whenever the magnitude a_{j+1} is larger than a threshold T_{acc}.

2.4.2.2 Stride length estimation

The Zero Velocity Updates (ZUPT) Stride Length (SL) algorithm is widely used with high accuracy for most statuses of a user (walk or run) [60]. In particular, when the user is in the stance phase, the velocity is zero. Using this information, the algorithm can correct the drifts of the accelerometer, and decrease the stride length errors in the swing phases when we employ (2.12)-(2.14).

There are three steps to estimate the length of the ith stride (or the ith swing phase) denoted by SL_i[60]. First, we use equation (2.13) to calculate the linear velocities in the duration of swing phase i. Note that a swing phase usually contains a number of IMU's sampling periods. Second, we correct the drifts as follows: the velocities are decreased by the linear interpolation of μ_i, i.e., the mean velocity at the stance phase i, and μ_{i-1} [60]. Third, we carry out the integration of (2.13) to get the position increment, whose abstract value is the stride length SL_i.

2.4.2.3 Orientation estimation

There are two main methods to estimate the orientation of the pedestrian in the ith stance phase (before the ith swing phase), denoted by θ_{stance_i}, based on the IMU on his/her mobile device: the gyroscope method and the accelerometer method. In the gyroscope method, by using equation (2.12), the INS system can output the orientation $\theta_{i,gyro}$. Although the gyroscope method can give accurate results in a short time period, the drift will increase with time. On the other hand, the accelerometer method is less accurate but does not accumulate the errors as time goes by. Thus, we combine these two methods as follows by introducing a control parameter γ_i $(0 \leq \gamma_i \leq 1)$:

$$\theta_{stance_i} = (1 - \gamma_i) \cdot \theta_{i,acc} + \gamma_i \cdot \theta_{i,gyro} \tag{2.16}$$

Consequently, when we have all the above results, we can update the user's position after every m steps. Particularly, the $(k+1)$th estimated position, denoted by $\bar{\boldsymbol{X}}_{k+1}$, can be calculated as:

$$X^x_{k+1} = X^x_k + \sum_{i=1}^{m} SL_i \cdot \cos(\theta_{stance_i}), \tag{2.17}$$

$$X^y_{k+1} = X^y_k + \sum_{i=1}^{m} SL_i \cdot \sin(\theta_{stance_i}). \tag{2.18}$$

2.4.3 Coupling RSSI and INS Systems through a Kalman Filter

In this section we develop a new Kalman filter that can well integrate the above RSSI and INS systems to estimate the object's position.

Specifically, Kalman filters are widely used in data fusion and state estimation [35], and many Kalman filters [32, 29] have been proposed to solve different problems [71, 17, 31, 14]. These different forms of Kalman filters follow the same two general steps: *the prediction (or propagation) step* and *the update step*. In the *prediction step*, the system states are propagated from the last iteration to the predicted states in the current iteration and the prior distribution is generated by using the estimated model of the system. In the *update step*, the filter has two kinds of information: the prior distribution from the *prediction step* and the measurement information. By using the measurement information and the prior distributions of the states, the filter can estimate the posterior distribution of the states and generate the current estimated states.

In what follows, we detail our Kalman filter design. Note that in the following equations, the index k denotes the iteration number, the index $k|k-1$ denotes the predicted parameters, and the $k|k$ denotes the estimated parameters.

First, in the *prediction step*, the filter uses the estimation model (2.14) to get a predicted state $\boldsymbol{X'}_{k|k-1}$ and the predicted estimated state covariance $\boldsymbol{P}_{k|k-1}$ (or the prior distributions of the states):

$$\boldsymbol{X'}_{k|k-1} = \boldsymbol{X'}_{k-1|k-1} + \bar{\boldsymbol{V}}\Delta T \tag{2.19}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{P}_{k-1|k-1} + \boldsymbol{Q}_k \tag{2.20}$$

where ΔT is the update period of the Kalman filter, $\boldsymbol{P}_{k-1|k-1}$ is the covariance matrix of the estimated state vector $\boldsymbol{X'}_{k-1|k-1}$ at time $k-1$, and $\boldsymbol{Q}_k$ is the covariance matrix of the current process noise's distribution. $\boldsymbol{Q}_k$ can be calculated as [68]:

$$\boldsymbol{Q}_k = \boldsymbol{I} \cdot (\frac{1}{2}\sigma_a \Delta T^2)^2 + \boldsymbol{I} \cdot (\sigma_v \Delta T)^2 \tag{2.21}$$

where $\boldsymbol{I}$ is the identity matrix, σ_a and σ_v are the standard deviation of the acceleration and of the velocity of the IMU, respectively.

Then, when the system receives enough measurement information from both of the subsystems, the Kalman filter estimates the posterior distribution of the states and generates the current estimated states. Specifically, the *update step* generates the Kalman gain, estimated state, and covariance matrix of the estimated state for the next iteration. Such updates can be made by:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{C}(\boldsymbol{C}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T + \boldsymbol{R}_k)^{-1} \tag{2.22}$$

$$\boldsymbol{X'}_{k|k} = \boldsymbol{X'}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{Z}_k - \boldsymbol{C}\boldsymbol{X'}_{k|k-1}) \tag{2.23}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{C})\boldsymbol{P}_{k|k-1} \tag{2.24}$$

where $\boldsymbol{Z}_k$ is the observation results from the RSSI system and INS system as shown in (2.10) and (2.17)-(2.18) respectively, $\boldsymbol{K}_k$ is the optimal Kalman gain, and $\boldsymbol{R}_k$ is the covariance matrix of the measurement error $\boldsymbol{\Upsilon}_k$:

$$\boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{Q}_{k/INS} & 0 \\ 0 & \boldsymbol{R}_{k/RSSI} \end{bmatrix}. \tag{2.25}$$

The entry $\boldsymbol{Q}_{k/INS}$ corresponding to the INS system is the same as $\boldsymbol{Q}_k$, and $\boldsymbol{R}_{k/RSSI}$ is the covariance matrix of the RSSI system's results. The initial $\boldsymbol{R}_{0/RSSI}$ can be given by:

$$\boldsymbol{R}_{0/RSSI} = \boldsymbol{I} \cdot \sigma_R^2 \tag{2.26}$$

where σ_R is standard deviation of the RSSI system's results that could be predetermined by **a few previous runs** a few experiments. We will see later that $\boldsymbol{R}_{k/RSSI}$ will be updated in the algorithm.

Note that as shown in (2.23), we can fuse the results from the RSSI and INS systems by using the Kalman gain $\boldsymbol{K}_k$ (2.22).

2.4.4 Realtime Update of the Channel Model

As indicated in Section 2.3, the RSSI system's accuracy is directly influenced by the measurement accuracy of the path loss exponent η. However, this value is difficult to obtain in a dynamic environment, where η needs to be obtained experimentally every time we need an estimated position from the RSSI system. Thus, even if η can be accurately calibrated at the beginning of the estimation, η will continue to change because of the dynamic nature of the environment. Some prior works [7, 5] claim that their systems can update the path loss exponent η by using some fixed WiFi anchors and reference points,

however, their update is not realtime or accurate enough to catch the changing of the actual parameters.

Different from the previous works, we update the path loss exponent η by recursively using the results of the proposed Kalman filter. The basic idea is to use the Kalman filter's estimation results as the position state input to calibrate the new path loss exponent η. In particular, according to (2.1), we can easily calculate η as follows:

$$\eta_{new} = \frac{(A - P_{RX})}{10 \log(\left\| \boldsymbol{X'}_{k|k} - \boldsymbol{X}_{anchor-i} \right\|_2 / d_0)} \tag{2.27}$$

where $\boldsymbol{X'}_{k|k}$ and $\boldsymbol{X}_{anchor-i}$ are the estimated position obtained from the Kalman filter and the position of anchor i respectively, $\|\|_2$ is the Euclidean norm, and η_{new} is the updated path loss exponent based on the newly obtained estimated position from the Kalman filter. After we have an updated η_{new}, we employ the Kalman filter again to update $\boldsymbol{X'}_{k|k}$, which is then used to update η_{new} again. This process proceeds recursively until we have a good enough estimate of the path loss exponent, which is finally used to update η. Moreover, as will be shown in simulations, the iteration number in the recursive process is very small (mostly no more than 3), which means that this proposed calibration is very efficient.

2.4.4.1 Stop condition for the recursive update process for η

In the proposed scheme, it is important to find the correct condition to stop the recursive steps for updating η, so that we can avoid the influence of the noise and measurement errors. In particular, as shown in the the path loss function (2.1), there is always some noise that affect the accuracy of the RSSI system. Besides, the received signal strength P_{RX} may not be very accurate either. Similarly, there is noise and measurement error in

the INS system as well. The estimated position $\boldsymbol{X'}$ by the coupled Kalman filter thus still contains uncertainty. Since the new pass loss exponent η is calculated according to (2.27), it is influenced by these errors too. Thus, we decide to set a suitable threshold to avoid getting an unstable η in this recursive process.

Particularly, if the newly estimated position $\boldsymbol{X'}_{k|k}^{new}$ based on the new path loss exponent η in any iteration is away from the previously estimated position $\boldsymbol{X'}_{k|k}^{old}$ by a threshold T, it means that the pass loss exponent η did change and the update of η is necessary. Otherwise, the change of the pass loss exponent η may be due to the noise and measurement errors, and the system will ignore this update and terminate the recursive process of updating η. We notice that as shown in (2.24), the covariance matrix $\boldsymbol{P}_{k|k}$ of the estimated states can be used to obtain a confident range of the estimated position. Therefore, the threshold T can be set to:

$$T = \alpha\sqrt{({\sigma_x}^2 + \sigma_y^2)/2} \tag{2.28}$$

where σ_x^2 and σ_y^2 are the covariances of the estimated position's coordinates x and y in $\boldsymbol{P}_{k|k}$ respectively, and α is a coefficient needed to be tuned. In so doing, our recursive algorithm can generate a threshold online to control the recursive level, so that the update of the pass loss exponent η will converge.

2.4.4.2 Online updating of the RSSI measurement error covariance

Moreover, after the pass loss exponent η is finally updated as mentioned above, some other parameters in the Kalman filter should be updated as well.

Specifically, since the channel model is updated, the measurement covariance $\boldsymbol{R}_k$ needs to be updated accordingly. Rewriting the path loss function (2.1), we can know that the measurement $\tilde{d}_i$ is a random variable:

$$\tilde{d}_i = d_i \cdot 10^{\frac{\mathcal{N}(0,\sigma)}{10\eta}} = 10^{\mathcal{N}(\log_{10} d_i, \frac{\sigma}{10\eta})}. \tag{2.29}$$

From (2.29), we find that when the pass loss exponent η is updated, the covariance of the Gaussian distribution changes, and are inversely proportional. Therefore, when the proposed Kalman filter changes the pass loss exponent η, the new variance of the RSSI measurement error is updated as follows:

$$\sigma_{R/new} = \frac{\eta_{old}}{\eta_{new}} \cdot \sigma_{R/old}. \tag{2.30}$$

After obtaining these new variances, the Kalman gain $\boldsymbol{K}_k$ and the estimated position $\boldsymbol{X'}_{k|k}$ will also be updated.

2.4.5 Further Improvement

2.4.5.1 Online Reduction of RSSI Signal Outliers via L_1 Regression

We notice that RSSI measurements tend to have outliers because of unstable communications or disturbances caused by obstacles [34]. The performance of the Kalman filter will be seriously degraded by the outliers, so many robust schemes have been proposed to reduce them[30, 70, 59, 2, 41]. In this paper, we employ an L_1 regression based approach [30, 41] due to their convenient implementation and low computation complexity.

Specifically, in the proposed Kalman filter, we consider that the RSSI measurement error $\boldsymbol{v}_{k/RSSI}$ is given by:

$$\boldsymbol{v}_{k/RSSI} = \boldsymbol{r}_{k/RSSI} + \boldsymbol{o}_{k/RSSI} \tag{2.31}$$

where $\boldsymbol{r}_{k/RSSI}$ and $\boldsymbol{o}_{k/RSSI}$ are the Gaussian noise and outlier in the step k, respectively.

Then, we can estimate $\boldsymbol{o}_{k/RSSI}$ by solving the following optimization problem [41]:

$$\begin{aligned} &\min_{\boldsymbol{o}_{k/RSSI}} (\boldsymbol{v}_{k/RSSI} - \boldsymbol{o}_{k/RSSI})^T \boldsymbol{W}_k (\boldsymbol{v}_{k/RSSI} - \boldsymbol{o}_{k/RSSI}) \\ &+ \lambda \left\| \boldsymbol{o}_{k/RSSI} \right\|_1 \end{aligned} \tag{2.32}$$

where

$$\boldsymbol{W}_k = (\boldsymbol{I} - \boldsymbol{K}_k)^T \boldsymbol{R}_{k/RSSI}^{-1} (\boldsymbol{I} - \boldsymbol{K}_k) + \boldsymbol{K}_k^T \boldsymbol{P}_{k|k-1}^{-1} \boldsymbol{K}_k,$$

$\|\cdot\|_1$ is a L_1 norm, and λ is a regularization parameter which is needed to get the solution.

The solution to (2.32) can be obtained by [69]:

$$\boldsymbol{o}_{k/RSSI} = \begin{cases} \boldsymbol{v}_{k/RSSI} - \frac{\lambda}{2\boldsymbol{W}}, \text{ if } \boldsymbol{v}_{k/RSSI} > \frac{\lambda}{2\boldsymbol{W}} \\ 0, \text{ if } -\frac{\lambda}{2\boldsymbol{W}} \leq \boldsymbol{v}_{k/RSSI} \leq \frac{\lambda}{2\boldsymbol{W}} \\ \boldsymbol{v}_{k/RSSI} + \frac{\lambda}{2\boldsymbol{W}}, \text{ if } \boldsymbol{v}_{k/RSSI} < -\frac{\lambda}{2\boldsymbol{W}} \end{cases} \tag{2.33}$$

where $\boldsymbol{v}_{k/RSSI} = \boldsymbol{Z}_{k|RSSI} - \boldsymbol{X'}_{k|k-1}$, and $\frac{\lambda}{2\boldsymbol{W}}$ is used as a threshold to cut the outlier. We set $\frac{\lambda}{2\boldsymbol{W}}$ to the standard deviation of $\boldsymbol{v}_{k/RSSI}$ which is defined by:

$$\boldsymbol{\Sigma}^2_{\boldsymbol{v}_{k/RSSI}} = \boldsymbol{P}_{k|k-1} + \boldsymbol{R}_{k/RSSI}. \tag{2.34}$$

Therefore, λ can be set to:

$$\boldsymbol{\lambda} = 2\boldsymbol{W_k}\boldsymbol{\Sigma}_{\boldsymbol{v}_{k/RSSI}}. \tag{2.35}$$

By calculating the outliers, we can reduce them from the RSSI measurements $\boldsymbol{Z}_{k/RSSI}$.

2.4.5.2 Online Estimation for the RSSI Measurement Covariance

As mentioned above, the RSSI measurements have outliers which can now be removed. This enables us to have a more accurate estimate of the RSSI measurement error covariance $\boldsymbol{R}_{k/RSSI}$. Besides, the change in the environment may lead to different $\boldsymbol{R}_{k/RSSI}$'s. So before we update $\boldsymbol{R}_{k/RSSI}$ using (2.30) when we conduct realtime update of the channel model as described in Section 2.4.4, we need to obtain $\boldsymbol{R}_{k/RSSI}$ by using the RSSI measurements without outliers.

In particular, let $\boldsymbol{v}'_{j/RSSI}$ be equal to $\boldsymbol{v}_{j/RSSI}$ minus the outlier. Based on the adaptive Kalman filter (AKF) from [42], the covariance of $\boldsymbol{v}'_{k/RSSI}$ can be estimated by:

$$\boldsymbol{C}_{\boldsymbol{v}'_{k/RSSI}} = \frac{1}{N} \sum_{j=k-N+1}^{k} \boldsymbol{v}'_{j/RSSI} \boldsymbol{v}'_{j/RSSI}{}^{T} \tag{2.36}$$

where N is the estimation window size. Then, from (2.34), the RSSI measurement error covariance can be calculated as:

$$\hat{\boldsymbol{R}}_{k/RSSI} = \boldsymbol{C}_{\hat{\boldsymbol{v}}_{k/RSSI}} - \boldsymbol{P}_{k|k-1}, \tag{2.37}$$

and then we can update $\boldsymbol{R}_{k/RSSI}$ as follows:

$$\boldsymbol{R}_{k/RSSI} = (1-\beta)\boldsymbol{R}_{k-1/RSSI} + \beta\hat{\boldsymbol{R}}_{k/RSSI} \tag{2.38}$$

where β is a control parameter.

The complete algorithm description for CRIL is detailed below:

1: **procedure** CRIL($\boldsymbol{X'}_{0|0}$, $\boldsymbol{P}_{0|0}$, $\boldsymbol{Q}_1$, $\boldsymbol{R}_1$)

2: Collect the position result form the INS system by (2.17) and (2.18).

3: Collect the position result form the RSSI system by (2.10).

Deduce the RSSI outlier $\boldsymbol{o}_{k/RSSI}$ and estimate the RSSI measurement covariance $\boldsymbol{R}_{k/RSSI}$ by (2.33) and (2.38), respectively.

Denote their own position results as $\boldsymbol{Z}_k^{old}$. Compute (2.19), (2.20) to estimate predicted states $\boldsymbol{X'}_{k|k-1}$ and predicted covariance $\boldsymbol{P}_{k|k-1}$.

Calculate Kalman gain K_k by (2.22) and estimate the estimated position $\boldsymbol{X'}_{k|k}^{old}$ by (2.23) by using the measurement $\boldsymbol{Z}_k^{old}$.

Use the estimated position $\boldsymbol{X'}_{k|k}^{old}$ to get the new path loss exponent η^{new} by (2.27), and the new RSSI position result $\boldsymbol{Z}_{k/RSSI}$.

Calculate the new estimated position $\boldsymbol{X'}_{k|k}^{new}$ with the new RSSI results and new η^{new} by (2.23).

while $\left\| \boldsymbol{X'}_{k|k}^{new} - \boldsymbol{X'}_{k|k}^{old} \right\|_2 > T$ **do**

    Set $\boldsymbol{X'}_{k|k}^{old} \leftarrow \boldsymbol{X'}_{k|k}^{new}$

    Update the path loss exponent η, and RSSI position result $\boldsymbol{Z}_{k/RSSI}$.

    Calculate the new estimated position $\boldsymbol{X'}_{k|k}^{new}$ by (2.23).

end while

Update the RSSI measurement covariance $\sigma_{R/new}$ by (2.30).

Update Kalman gain K_k and $\boldsymbol{P}_{k|k}$, and calculate final estimated position $\boldsymbol{X'}_{k|k}^{Final}$.

end procedure

2.5 Simulation Results

In this section, we analyze the performance of our proposed CRIL system through extensive simulations and validate the proposed scheme by comparing it with the following two previous methods:

1. RSSI localization system (RSSI)[67]: With a fixed time step, the RSSI localization system estimates the object's position based on the RSSI values only.
2. A hybrid RSSI and INS system with Kalman filter without channel model update (KF)[80]: Kalman filter is employed to fuse the results from the RSSI system and the INS system, but there is no channel model update process.

As will be seen later, our proposed system gives very accurate and stable localization results even in dynamic environments.

2.5.1 Simulation Environment and Parameter Setting

In the simulations, we consider that there are three WiFi anchors located at three fixed points, respectively, in a room of $50\,\mathrm{m}\times 50\,\mathrm{m}$. The tracked object moves around under the coverage of all three anchors, and they are all on the same plane. These anchors send WiFi signals to the tracked object at a fixed rate of one per $0.5\,\mathrm{sec}$. The speed of the object is fixed at $1\,\mathrm{m/s}$.

At the start point $(5,\ 5,\ 0)$, the object uses its exact position to initialize the parameters and position states in the proposed Kalman filter. The values of σ_a and σ_v in the covariance matrix $\boldsymbol{Q}_k$ of the process noise $\boldsymbol{\Gamma}_{\mathbf{X}}$ are set to $0.1\,\mathrm{m/s^2}$ and $0.1\,\mathrm{m/s}$. In the covariance matrix of RSSI measurement noise, i.e., $\boldsymbol{R}_{RSSI}$, the initial value of σ_R is set to $2\,\mathrm{m}$. As explained in the previous section, this $\boldsymbol{R}_{k/RSSI}$ will be updated online to enhance the accuracy of our algorithm.

The log-normal shadowing path loss model is used as the signal propagation model, where $\sigma = 2\,\mathrm{dBm}$, and $A = -38.0460\,\mathrm{dBm}$.

2.5.2 Results under Constant Path Loss Exponent

In this simulation, the real path loss exponent is set to a constant value of 4. As shown in the description of the CRIL algorithm, the noise influences the accuracy of the estimated path loss exponent η. In this special situation where the real path loss exponent η is not changing, the proposed recursive update process for η should keep the estimated η unchanged. The small variation of the estimated η caused by the noise should be ignored by the recursive update process. That means the new CRIL system should have the same performance as the KF method [80].

Specifically, we can see in Figure 2.2 that the estimated path loss exponent η (i.e., blue circles in Figure 2.2) is unchanged. This is because the proposed recursive update process for η ignores the influence of noise, and the updated η_{new} obtained in the recursive update process (i.e., blue squares in Figure 2.2, only one iteration) is not accepted by the system. The results in Figure 2.2 validate the design of the proposed threshold T for controlling the recursive level of the update process, and shows that we can have a stable and converging update process in this environment with a constant path loss exponent.

In Figure 2.3 and Figure 2.4, we show the estimated positions by our proposed CRIL and by RSSI [67] and KF [80], respectively. We can see that both our CRIL and KF [80] can track the objects' real positions well, with localization errors on the scale of centimeters. The localization errors of RSSI [67] are larger, up to **2.5** 1.5 meters. Figure 2.2

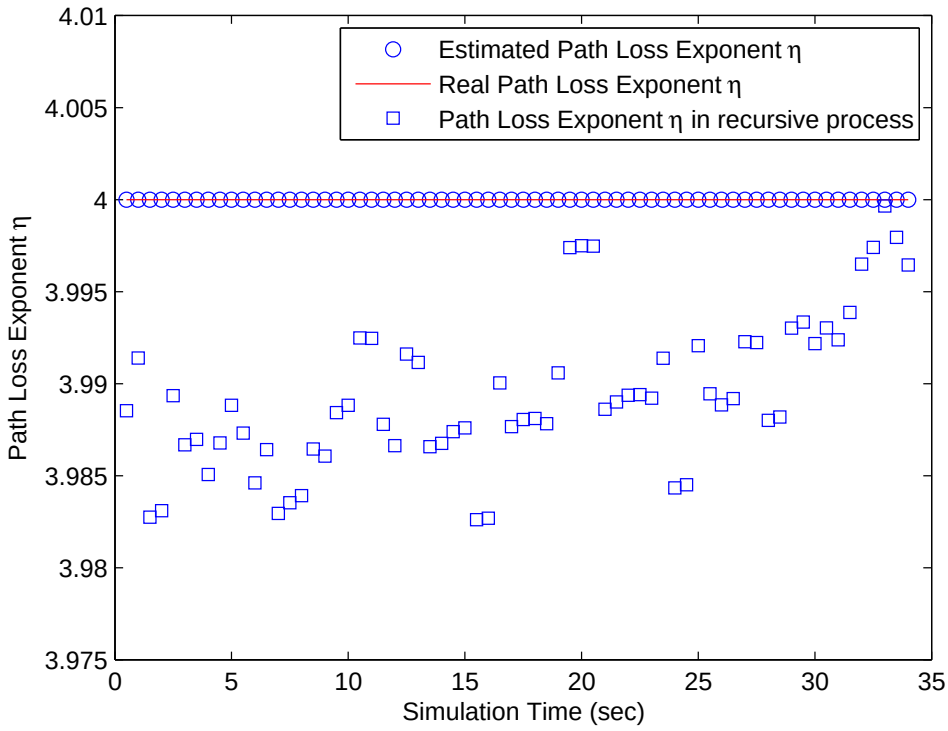

Figure 2.2

Estimated path loss exponent η by CRIL

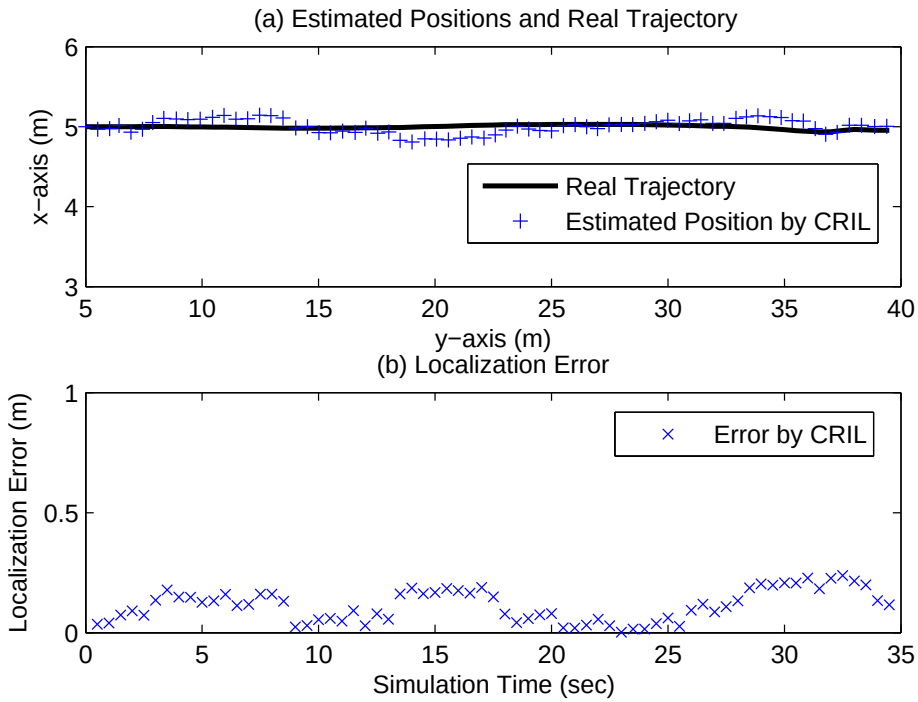

Figure 2.3

Comparison of estimated positions by CRIL and the real trajectory

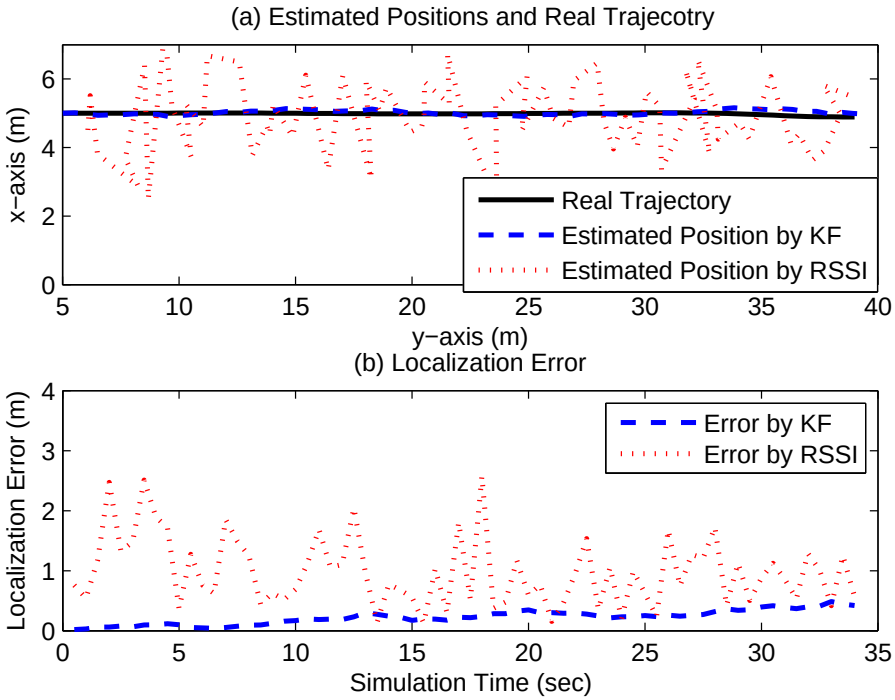

Figure 2.4

Comparison of estimated positions by RSSI [67], KF [80] and the real trajectory

and Figure 2.3 together demonstrate that the proposed recursive update process for η can have a good performance, even when the noise and the measurement errors influence the estimation of η.

2.5.3 Results under Dynamic Path Loss Exponent

Previous results show that the proposed CRIL system can obtain the same accuracy level as the normal Kalman filter when the path loss exponent is a constant. On the other hand, one of CRIL's biggest advantages is that it can estimate and update the path loss exponent η through realtime calibration. This is one main reason that the proposed system can have high localization accuracy compared to previous RSSI alone localization systems and hybrid localization systems.

In what follows, two situations are discussed and investigated: first, the value of the real path loss exponent η slowly changes with time; second, the value of the real path loss exponent η has a sudden change, i.e., jumps form one value to anther. In the next two subsections, we show that the proposed CRIL can achieve accurate localization in both situations.

2.5.3.1 Path loss exponent η is slowly changing

In this simulation, the real path loss exponent η slowly changes form 4 to 4.35. This can happen when the humidity in the room is slowly increasing [52]. In Figure 2.5[2], the estimation error of the path loss exponent is very small, and most of the errors are below 0.02. The estimated path loss exponent η can follow the slow changing of the real

[2]Comparison of estimated path loss exponent η by CRIL and the real path loss exponent η which is slowly changing.

path loss exponent η. Although there are still some small errors caused by the noise and measurement errors, the estimated path loss exponent η is very accurate and can give very good performance.

Figure 2.6[3] and Figure 2.7 demonstrate the localization performance of CRIL and of RSSI [67] and KF [80], respectively. Specifically, in Figure 2.6, the proposed CRIL system achieves similar performance under dynamic path loss exponents to that in Figure 2.3 where the path loss exponent is a constant. The variation of the real path loss exponent η does not influence the accuracy of the proposed CRIL system much. This is because the recursive update process for η can detect this variation efficiently and accurately. We also notice that our proposed scheme's performance (with localization errors up to 0.4m) is much better than the RSSI [67] and KF [80] localization systems, as shown in Figure 2.7. The KF scheme leads to localization errors of a few meters, which cannot be used. The RSSI method is even worse than the KF scheme and has localization errors more than 20m, due to the inaccurate channel model. From these results, we can see that the proposed CRIL system can fully utilize the fused results of INS and RSSI systems and detect the variations of the path loss exponent η quickly, thus improving the localization performance. Moreover, as shown in Figure 2.8, we notice that the number of iterations numbers in the recursive update process for η are no more than 3, which demonstrates that the proposed update process is very efficient.

[3]Comparison of estimated positions by CRIL and the real trajectory with the slowly changing path loss exponent η

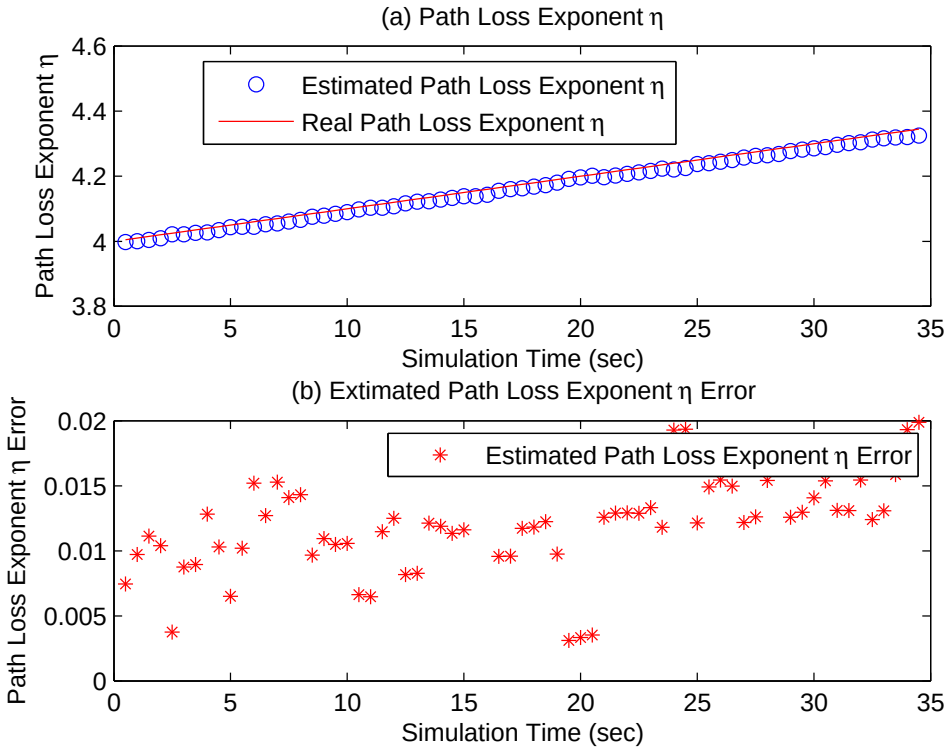

Figure 2.5

Comparison of estimated path loss exponent η

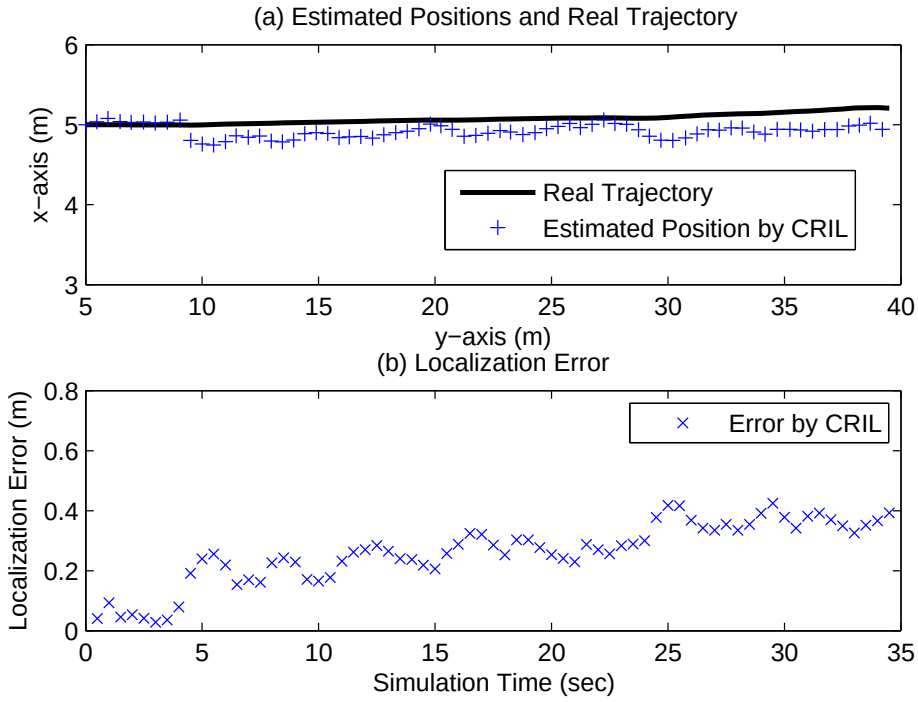

Figure 2.6

Comparison of estimated positions by CRIL and the real trajectory.

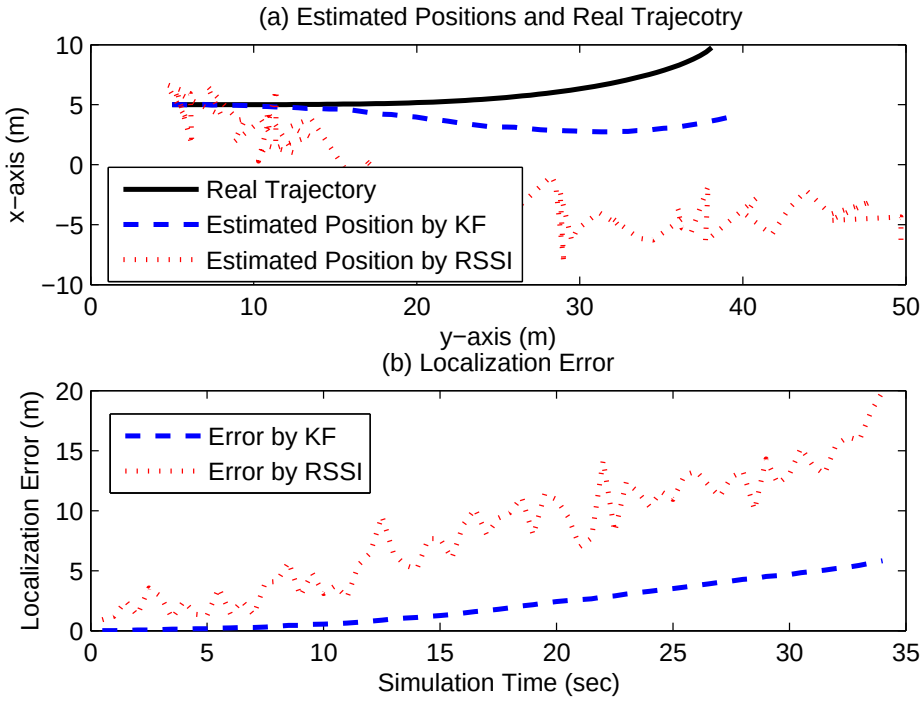

Figure 2.7

Comparison of estimated positions by KF[80] and RSSI[67], and the real trajectory.

2.5.3.2 Path loss exponent η is suddenly changed

In this simulation, the value of the path loss exponent η jumps form 4.0 to 4.2 at about $t = 26\,\mathrm{sec}$. This may happen when the mobile object enters one room form another [52]. From Figure 2.9, we can find that the estimation error of the path loss exponent η is also very small as that in Figure 2.3 and Figure 2.5, and most of the errors are less than 0.015. We can easily see that the estimated path loss exponent η can track the suddenly changed real path loss exponent η very fast. It is important to notice that the estimation error after $t = 26\,\mathrm{sec}$ is not much different, compared with that before $t = 26\,\mathrm{sec}$. The small error of η caused by the noise and measurement error will not influence the performance of the proposed system too much. Figure 2.10 shows that the performance of the proposed CRIL system is very good and not affected by the sudden change of the real path loss exponent η.

Particularly, in Figure 2.10, because the successful detection of the sudden jump of the real path loss exponent η, the proposed system can still output estimated positions with high accuracy. The performance in this scenario is similar to those in the previous simulations where the real path loss exponent η is a constant or slowly changing. There is not much difference in the estimated error before and after the jump of η, which mostly remains below 0.2m. This means the performance of the proposed CRIL system is smooth and stable when the parameters are changed suddenly. This is desirable in real-world applications, where a localization system should be adaptive to these changes and update its own parameters effectively. We notice in Figure 2.11 that both the RSSI system's and the KF system's localization errors are several meters. Thus, the RSSI [67] and KF [80]

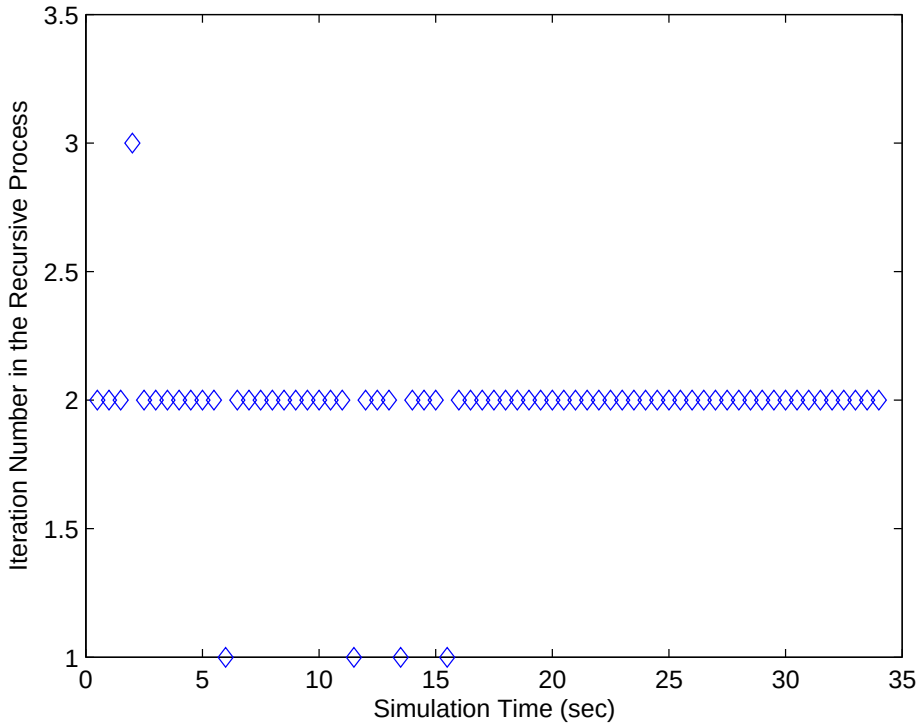

Figure 2.8

The number of iterations in the recursive update process for η.

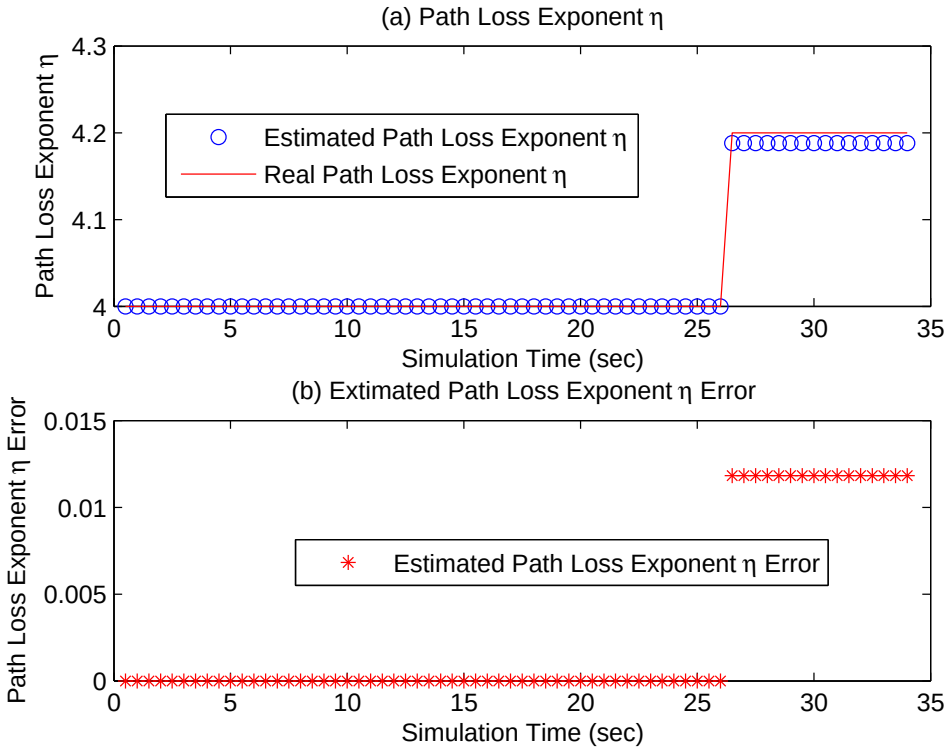

Figure 2.9

Comparison of estimated path loss exponent η by CRIL.

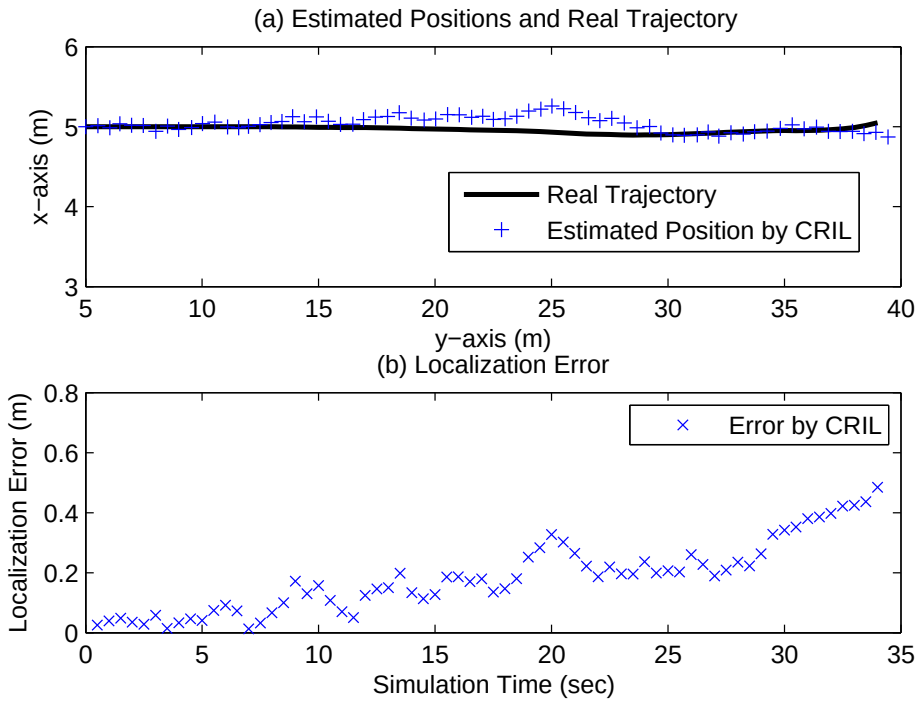

Figure 2.10

Comparison of estimated positions by CRIL and the real trajectory.

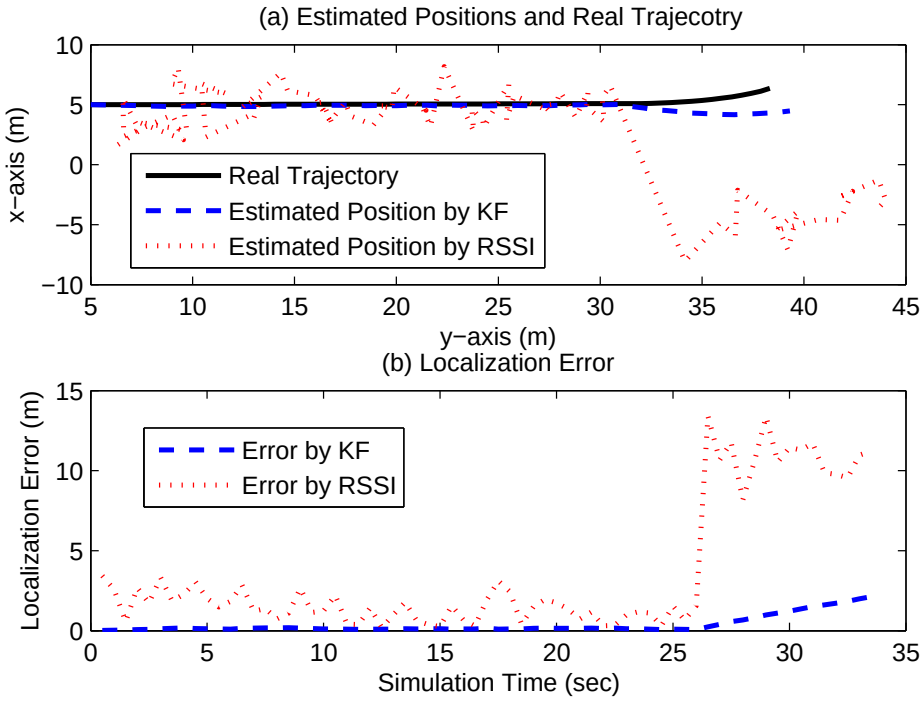

Figure 2.11

Comparison of estimated positions by KF[80], RSSI[67] and the real trajectory.

systems cannot give good localization results because of the sudden changing of the path loss exponent.

2.5.4 Results under Dynamic RSSI Environments

In the previous experiments, the RSSI values is following a fixed gaussian distribution, and there are no strong outliers in the values. However, based on previous research [67, 52], these are not true in practical environments. In what follows, we discuss and investigate three situations: first, the covariance of RSSI values changes; second, outliers are randomly added into the RSSI values; third, both of the covariance changes and outliers are added into the RSSI values. We show that our CRIL with robust scheme can achieve accurate localization in these situations.

2.5.4.1 Covariance of RSSI values changes

In this simulation, the σ of the RSSI values changes from $2\,\mathrm{dBm}$ to $4\,\mathrm{dBm}$. In Figure 2.12[4], the estimation errors in the case of CRIL without RSSI covariance update is much bigger than the results in the previous simulations. In contrast, the estimation errors in the case of CRIL with RSSI covariance can give small errors, although they are a little bigger than the results in the fixed RSSI covariance case.

[4]Comparison of localization errors by CRIL with RSSI covariance update and CRIL without RSSI covariance update when the covariance of RSSI values changes.

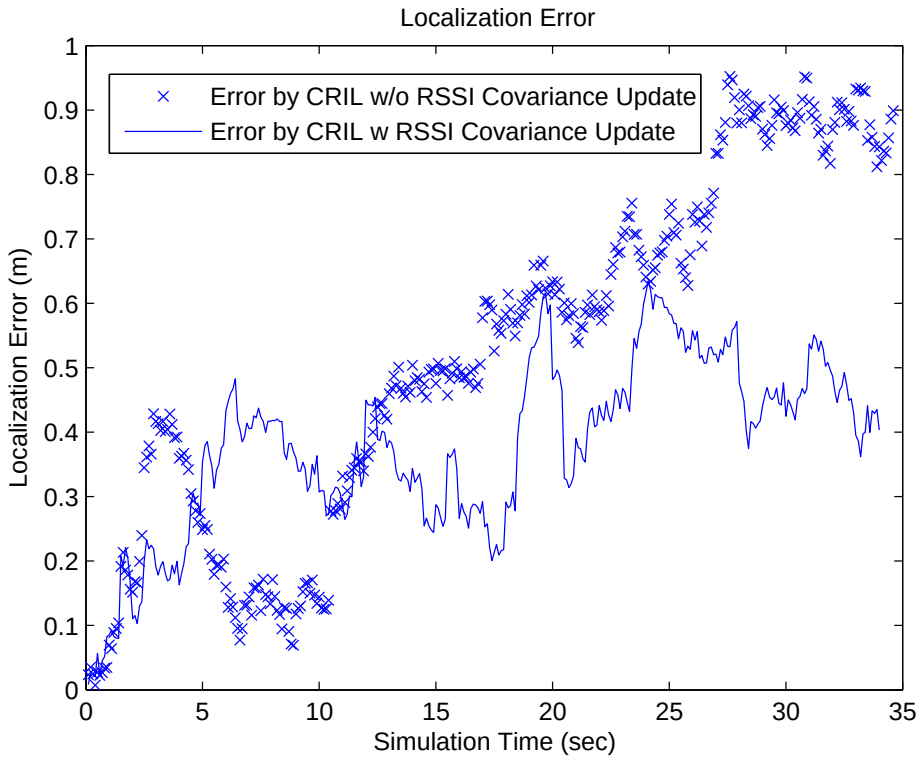

Figure 2.12

Comparison of localization errors

2.5.4.2 Outliers randomly are added into the RSSI values

In this simulation, the outliers of $15\,\mathrm{dBm}$ are randomly added into the RSSI values with the probability of 0.2. In Figure 2.13[5], these outliers lead to big localization errors in the case of CRIL without outlier reduction. However, the localization errors in the case of CRIL with outlier reduction demonstrate its effect, which give us much better performance.

2.5.4.3 Both of the covariance variance and outliers are added

In this simulation, both the covariance changes and outliers like those in 1) and 2) are added into the RSSI measurement values. In Figure 2.14[6], the localization errors in the case of CRIL with both RSSI covariance update and outlier reduction are much smaller than those in the case of CRIL without both RSSI covariance update or outlier reduction. This figure demonstrates that the proposed CRIL scheme can well deal with the uncertainty in RSSI values.

2.6 Experiment Results

We conduct experiments on a mobile device (iPhone 5S) to evaluate the performance of our proposed CRIL, which has a tri-gyro and a tri-accelerameter. The IMU data is obtained through the App "Sensor Monitor", and the sampling rate is $120\,\mathrm{Hz}$. The four WiFi anchors are Linksys WRT54GL routers with OpenWrt system. The experiments are done in our lab of $15\,\mathrm{meters}$ by $10\,\mathrm{meters}$. The WiFi anchors are located at the four corners of the room

[5]Comparison of localization errors by CRIL with outlier reduction and CRIL without outlier reduction when the outliers are randomly added into the RSSI values.

[6]Comparison of estimated position errors by CRIL with RSSI covariance update and outlier reduction and CRIL without RSSI covariance update or outlier reduction when both the covariance changes and outliers are present.

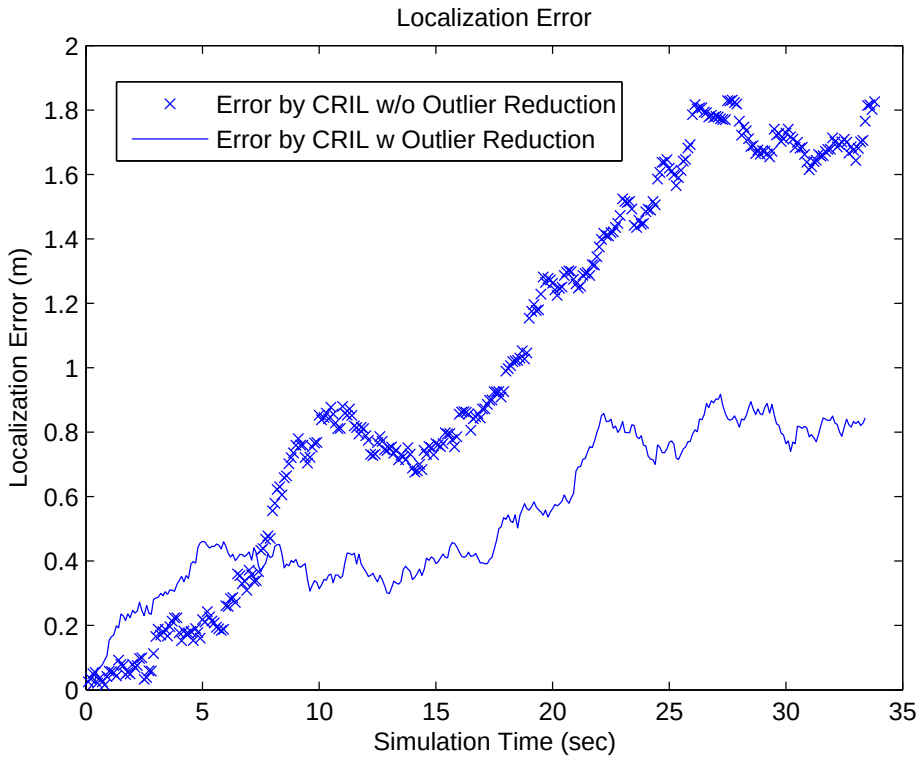

Figure 2.13

Comparison of localization errors

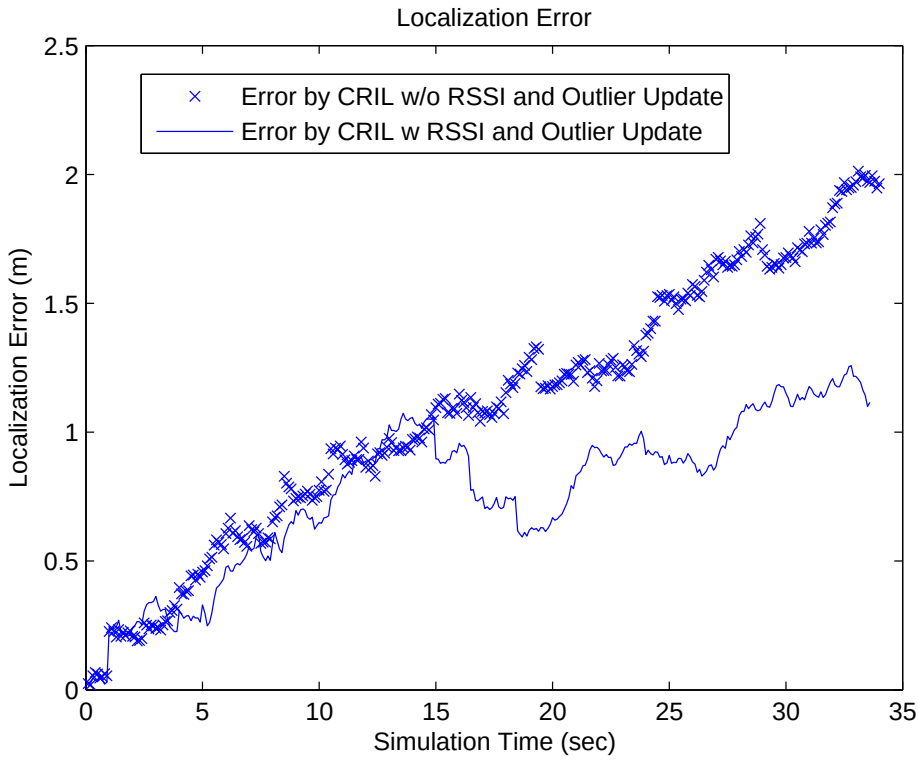

Figure 2.14

Comparison of estimated position errors

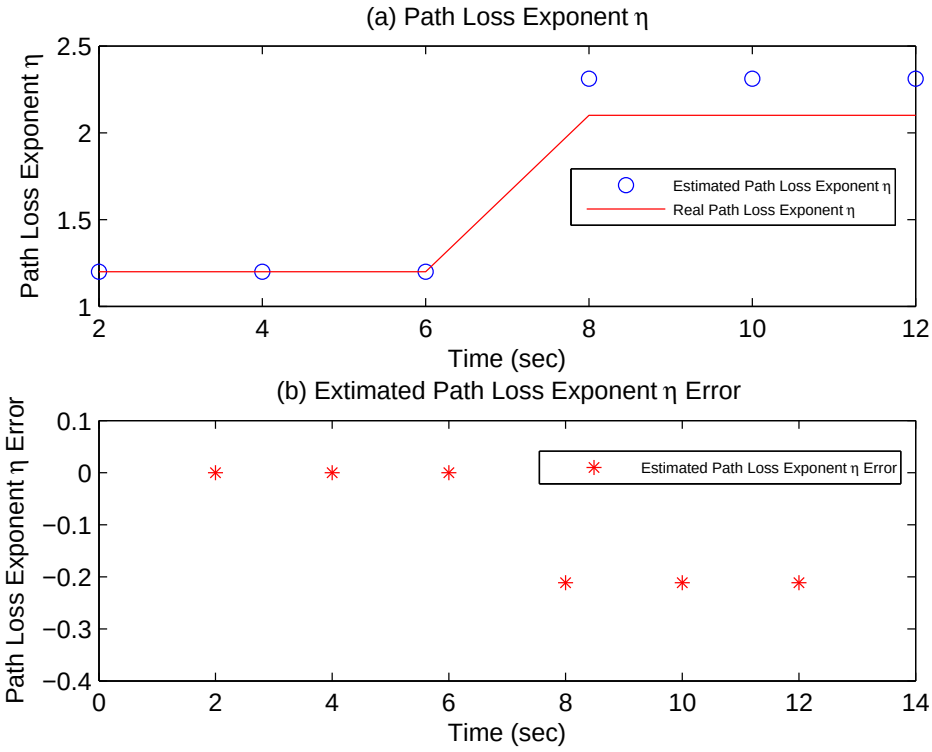

Figure 2.15

Comparison of the estimated path loss exponent η by CRIL.

and they all cover the whole room. A basic fingerprint system based on k-closest neighbors method [76] is implemented as a comparison of our proposed CRIL system.

In the first experiment we test our CRIL system with the path loss exponent η jumping from 1.2 to 2.1 at $t = 7\,\mathrm{sec}$. We make it happen by creating many blocks in the room [52]. From Figure 2.15, we can find that the estimation error of the path loss exponent η is very small like those in the simulations, and most of the errors are less than 0.25. We can easily see that the estimated path loss exponent η can track the changed real path loss exponent η very fast. The error of η caused by the noise and measurement error will not influence the performance of the proposed system too much.

Besides, Figure 2.16[7] shows that the localization performance of the proposed CRIL system is very good and not affected by the sudden change of the real path loss exponent η. In particular, because of the successful detection of the sudden change in the real path loss exponent η, the proposed CRIL system can still output estimated positions with high accuracy. There is not much difference in the estimated error with and without the jump of η, which mostly remains below $3\,\mathrm{m}$. This shows that the performance of the proposed CRIL system is good and stable.

In the second experiment, we implement the fingerprint system when the value of the path loss exponent η is 1.20. The same as above, one test is carried out when the environment does not change, and the other is performed when we change the path loss exponent η from 1.2 to 2.1. We notice in Figure 2.17[8] that in the static environment, the average

[7]Comparison of the estimated positions by CRIL in the static environment and in the dynamic environment.

[8]Comparison of the localization errors by the fingerprint system in the static environment and in the dynamic environment.

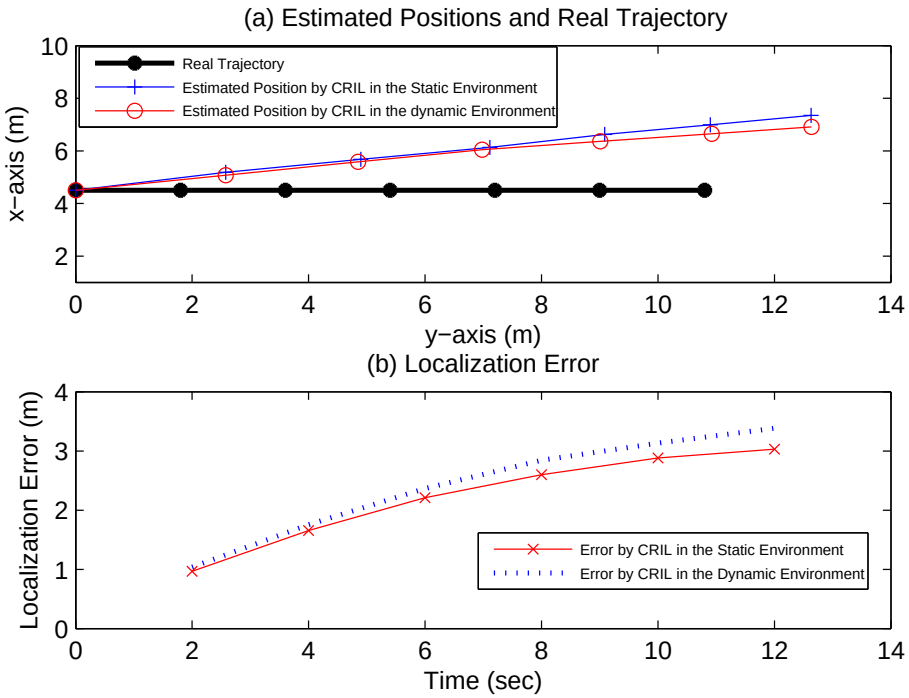

Figure 2.16

Comparison of the estimated positions

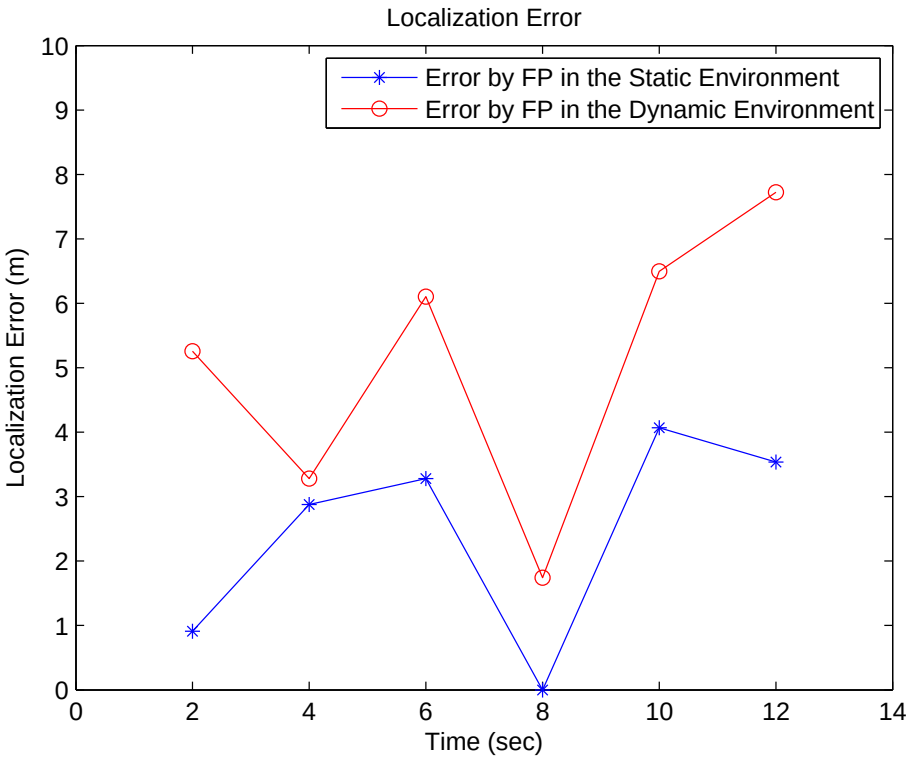

Figure 2.17

Comparison of the localization errors

error is $2.4\,\mathrm{m}$, and in the changed environment, the average error is $5.1\,\mathrm{m}$. Obviously, although in the static environment the fingerprint system have a comparable performance with our CRIL system, in the dynamic environment our CRIL system can maintain similar performance and outperform the fingerprint system.

2.7 Conclusion

Involved in many emerging applications, indoor localization systems have attracted intense research interests recently. This paper has proposed an efficient, adaptive and robust indoor localization system: Coupled RSSI and INS Localization (CRIL). Generally, the calibration of the channel model in previous methods is complex, not realtime, and time-consuming. The proposed CRIL system utilizes the fused results from both RSSI and INS systems through a newly designed Kalman filter to estimate the object's location while calibrating the channel model in realtime. Extensive simulation and experiment studies demonstrate that the proposed system can accurately localize mobile objects. Compared with previous methods which may not converge and whose localization errors are several meters or even tens of meters, simulations show that CRIL has much better performance in dynamic environments with localization errors up to $1\,\mathrm{m}$. In particular, CRIL is able to detect the changes in the environment and adapt to dynamic environments quickly and effectively. Moreover, with outlier reduction and RSSI covariance update, CRIL can still give a good performance when there are more uncertainties in the RSSI values. Finally, through experiments, we prove that our CRIL system works well in practice.

CHAPTER 3

REALTIME3D: A NEW REALTIME 3D RECONSTRUCTION SCHEME

3.1 Introduction and background

As a promising technology, Unmanned Vehicle Systems (UVS) find more and more applications including environment monitoring [73], industrial production [13], and even exploration of Mars [39]. Except for those large and expensive UVS which are used by governments and research institutes, a lot of small-sized commercial UVS have emerged recently, because they are cost-efficient and easy to operate. Such small-sized UVS can facilitate many commercial and personal applications [79], such as building maintenance, gaming, filming and education. To enable these non-professional users to easily operate the UVS and utilize them for different applications, many important technologies need to be developed, one of which is autonomous navigation.

There are several small popular UVS on the market. Particularly, Parrot AR.Drone [10] is a cheap and small quadrotor UVS whose mechanical structure comprises four rotors. It has a vertical camera and a horizontal camera which can transfer UVLC (MJPEG-like) and P264 (H.264-like) videos at a rate of 15 frames per second, and a set of ultrasonic transceiver and receiver which can measure the flight height. These sensors enable it to have the hovering function. In the meantime, DJI Phantom series [79] target the middle to high end markets. With higher prices, DJI uses cameras with higher quality and stabilized

control systems to support more complex flight tasks, e.g. collision avoidance and target tracking. However, none of these low-price UVS can realize the complete autonomous navigation, because of the limited on-board computational ability and power.

To enable the real autonomous navigation, one approach is to obtain a realtime local 3D map, so that a UVS can find a candidate path in it. Researchers have proposed a few technologies utilizing different sensors and devices to perform 3D space reconstruction. For example, Microsoft Kinect [78] and Asus Xtion [20] use infrared rays to scan the objects, and estimate their depth based on the reflection. Google's self-driving car [21] has a Velodyne 64-beam laser scanner to generate the 3D models of the surrounding environment. These active sensor methods have high accuracy and good performance without prior assumptions and requirements. However, such designs are based on certain invisible light which requires high power and extra costs. In contrast, 3D models can also be reconstructed using the visible light. A popular way is to use the stereo cameras [3]. Particularly, two lens are placed apart by a certain distance in a stereo camera. Then simple geometric equations can be used to calculate the differences between the left image and the right image, and then obtain the depth of the front objects. Such stereo cameras systems need a highly accurate synchronization scheme to make sure that the two images from two lenses are subject to exposures at the same time. Besides, the installation error will propagate into the 3D estimation process. To address these problems, single visible light camera based systems are developed, e.g. [51] and [46]. In these designs, a single well calibrated camera (even a cheap web camera) is enough to conduct 3D reconstruction with well designed algorithms. Unfortunately, such computer vision algorithms involve many procedures which

are highly computationally intensive. Thus, most previous systems cannot work in a real-time fashion, or have degraded results due to the tradeoff between the performance and the computing efficiency.

To improve computing efficiency of single visitable camera based systems, several attempts have been made in the literature with parallel computing and GPU computing. For example, Klein and Murray [33] design a Parallel Tracking and Mapping system (PTAM) by utilizing multi-core to improve the computing speed. GPU computing is also well-known for processing the images and float computing, because of GPU's special structures. Newcombe and Davison [51] developed GPU-based variational optical flow to reduce their system's computational complexity. We notice that these systems enhance the computational efficiency not by simplifying the algorithms themselves, but by increasing the hardware's usage efficiency. Such a methodology may not be appropriate for those light-weight small UVS for personal users, since these systems are not equipped with expensive and powerful multi-core GPUs or high-capacity batteries.

On the other hand, cloud computing emerges as a cost-effective and powerful computing paradigm, which serves as an ideal technology to support computationally intensive with algorithms on light-weight UVS. Delay is obviously an issue in cloud computing based systems. Simply outsourcing all the data and compute to the cloud would leave to significant communication delay. Unfortunately, although so far there have been several designs that utilize the cloud computing for robot control and realtime image processing systems [57], [65], [62], most of them simply upload all the raw images to the cloud, without carefully investigating an outsourcing scheme. In fact, a well designed outsourcing

scheme can significant reduce the communication delay while keeping low computational complexity locally. Besides, today's small UVS also offer multi-core CPUs. Although they are not as powerful as those in expensive computer server or GPUs, they can help with certain preprocessing and enable parallel computing in the pipeline of computing vision algorithms, while the other most heavy tasks are outsourced to the cloud.

In this paper, we develop a new realtime 3D reconstruction system called Realtime3D based on a single visible-light camera, which can enable a UVS to generate 3D scenes of its environment in realtime. Based on these 3D scenes, a lot of further processing can be conducted to realize autonomous navigation, realtime VR, 3D mapping, etc.. The camera we need can be any common off-the-shelf camera without any special requirement. A web camera or a mobile phone's camera can also be used in our system. Besides, the power consumption of these cameras is much lower than that of the active cameras like infrared and laser cameras. By using the camera's raw video as the system input, our system intelligently outsources the most expensive computations of th Structure from Motion (SfM) scheme [50] to the cloud, and further improve the computing efficiency through parallel computing and preporcessing specifically, the proposed Realtime3D first extracts image features and estimates the raw 3D points of the objects in the environment by improving the SfM algorithm design and utilizing parallel computing. These raw 3D points are further optimized through a bundle adjustment (BA) processing [75] in order to obtain a global sparse 3D point cloud. The traditional BA optimization is highly computationally expensive, and hence common on-board computers cannot process it in real-time. In our proposed system, we improve the efficiency of this process by outsoucing it to the cloud

and limiting the amount of data transmitted to the cloud as well. Moreover, we carefully select certain raw data and transmit from the UVS to the cloud, which can help transform the sparse 3D point cloud into a dens 3D point cloud. The proposed Realtime3D system is implemented and validated by extensive real experiments.

The rest of the chapter is organized as follows. In Section 3.2, we present the system models and fundamentals of 3D reconstruction. Section 3.3 describes our proposed Realtime3D system, which is evaluated in Section 3.4 through extensive real experiments. Finally, we conclude the chapter in Section 3.5.

3.2 System Models

3.2.1 System Architecture

As shown in Figure 3.1, we consider an unknown environment where we intend to fly a UVS to explore. The unknown environment has many 3D objects in it which the UVS need to discover. As described in Figure 3.2, the UVS carries an onboard single visible-light camera, a WiFi radio and an onboard computer. The onboard computer has a multi-core CPU, which can conduct simple processing on the input video but is not strong enough to build realtime 3D models of the environment by itself. Our propsed Realtime3D system processes the live streaming videos from the onboard camera which has been calibrated already. The WiFi radio on the UVS connects it to a pre-configured cloud server. Thus, the UVS can conduct the realtime 3D reconstruction by taking advantage of cloud computing. Furthermore, the 3D map generated by our scheme can be further processed for autonomous flight control purposes.

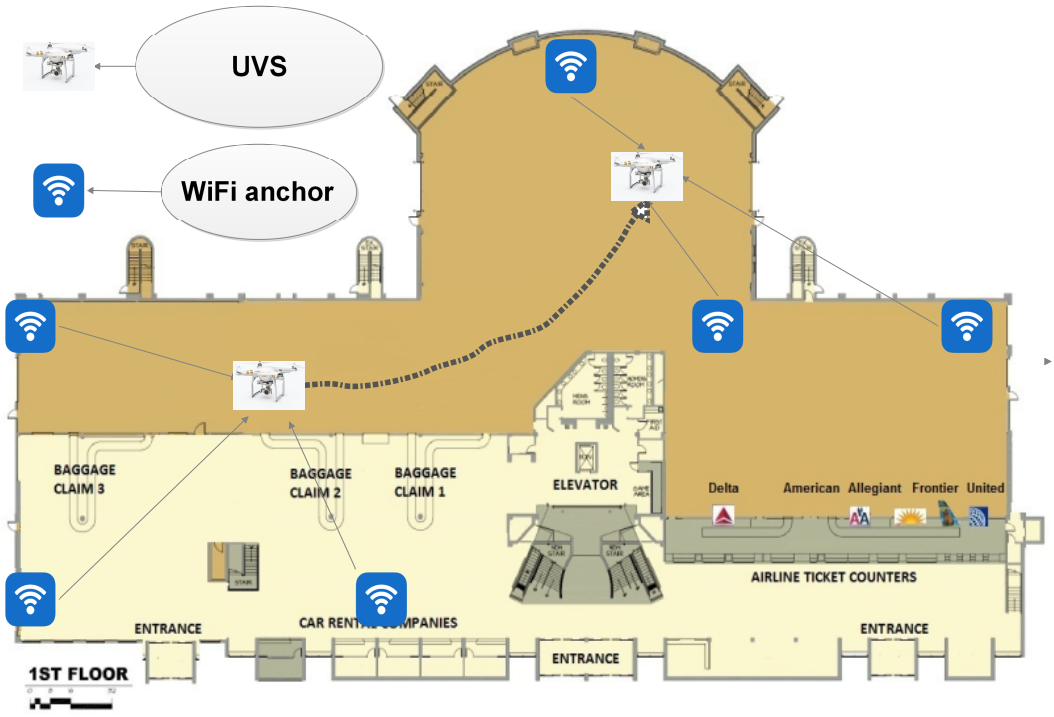

Figure 3.1

A typical scenario of environment exploration using a UVS.

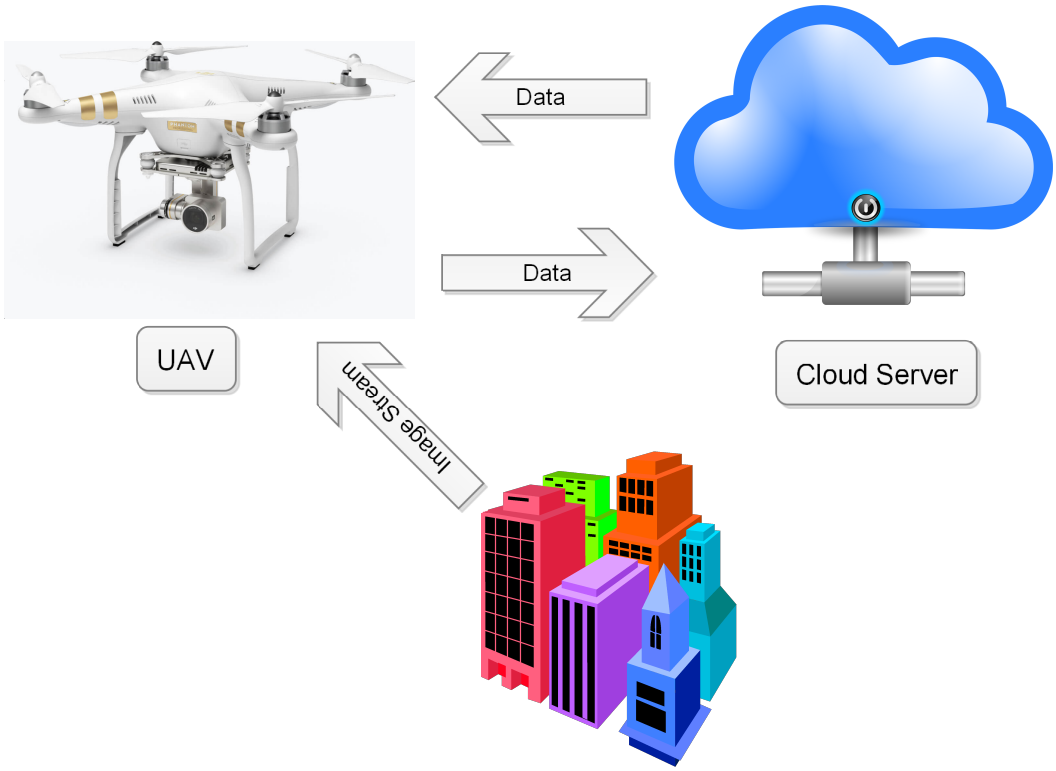

Figure 3.2

The architecture of our Realtime3D system

3.2.2 Camera Calibration

Cheap onboard cameras usually have serious distortion problems [25]. In particular, radial distortion and tangential distortion are two major distortions which have great influence on images' quality.

Radial distortions change straight lines into curves, especially those lines in the center of images. This problem can be resolved into the following equations [25]:

$$\begin{aligned} x_{corrected} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corrected} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3.1}$$

where (x, y) is the position of one pixel in the distorted image, $(x_{corrected}, y_{corrected})$ is the corrected or calibrated position, and $k_i, (1 \leq i \leq 3)$ are the distortion coefficients regarding the radius r, i.e. the distance from the center of the image to (x, y).

Similarly, tangential distortion makes some points in the images closer than they should be, because the lens in a camera are not in parallel with the image plane. The following equations can resolve this problem:

$$\begin{aligned} x_{corrected} &= x(2p_1 xy + p_2(r^2 + 2x^2)) \\ y_{corrected} &= y + (p_1(r^2 + 2y^2) + 2p_2 xy) \end{aligned} \tag{3.2}$$

where p_1 and p_2 are the distortion coefficients regarding the radius r.

In summary, we need five distortion coefficients to recover the undistorted input images; i.e., k_1, k_2, k_3, p_1, p_2.

Moreover, to perform 3D reconstruction, we need the intrinsic matrix K of a camera, which is also called the "camera matrix" and camera-dependent. Particularly, camera ma-

trix K information includes such as focus length (f_x, f_y) and optical center (c_x, c_y), which is a 3×3 matrix as follows [25]:

$$K = \begin{bmatrix} f_x & c_x & 0 \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

To determine all the above camera parameters, we need to conduct calibrations. One common calibration method is using several standard chessboard images as the input, which is supported by libraries such as OpenCV [9]. After the calibration process, we can retrieve the undistort images.

3.2.3 Epipolar Geometry

Before we detail our 3D reconstruction scheme, we introduce some basic concepts about epipolar geometry in the following. Specifically, the basic idea of reconstructing 3D models from 2D images is to estimate the depth information, which is lost when we take the images, based on images from multiple views. From [9], we know that at least two images are needed to construct what is called "stereo vision".

Let us take Figure 3.3 as an example, we have two cameras centered at O and O' respectively, which can each take an image of the current scene. From these two images (two blocks in the figure), we can estimate a point x's coordinates in the 3D space, i.e., X, centered at point O. Particularly, only with the left image, we cannot obtain X, because it can be any point along the line Ox. By considering the right image as well, we can determine X since it is obviously the intersection of Ox and $O'x'$.

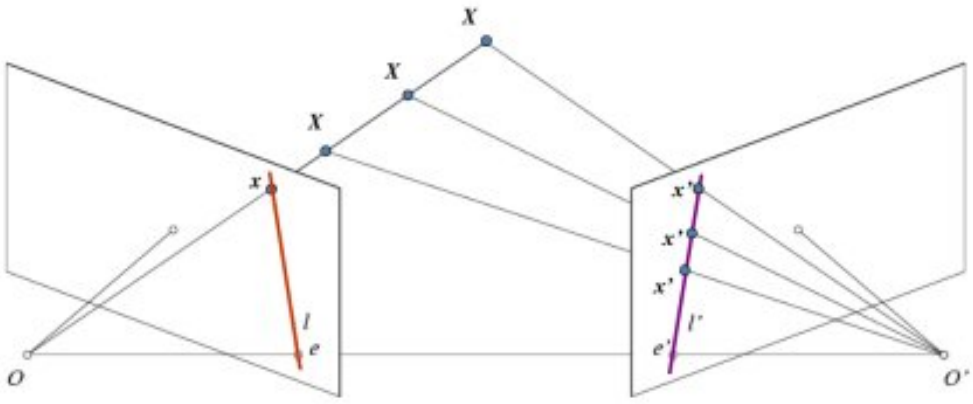

Figure 3.3

Two cameras taking images of the same scene [9]

The whole process of determining a point x's coordinates in the 3D space can be formulated as follows. There are two important matrices: the fundamental matrix denoted by F and the essential matrix denoted by E, which are both 3×3 matrices. By using either one of these two matrices, we can find one image's relative position to the other, and then project the 2D points to 3D points.

Specifically, as shown in Figure 3.4, to obtain the fundamental matrix F, we can construct the following equation involving the point x in the left image and the corresponding point x' in the right image:

$$\begin{bmatrix} x^T & 1 \end{bmatrix} F \begin{bmatrix} x' \\ \\ 1 \end{bmatrix} = 0 \tag{3.4}$$

where x and x' are 2D coordinates in images centered at O_l and O_r, respective. To solve for F, we need at least 8 pairs of x and x' to form 8 equations of Equation (3.4), because F

has 8 degrees-of-freedom. What is more, the fundamental matrix (F) and essential matrix (E) are related as follows:

$$F = K^T E K \tag{3.5}$$

where K is the camera matrix in Equation (3.3).

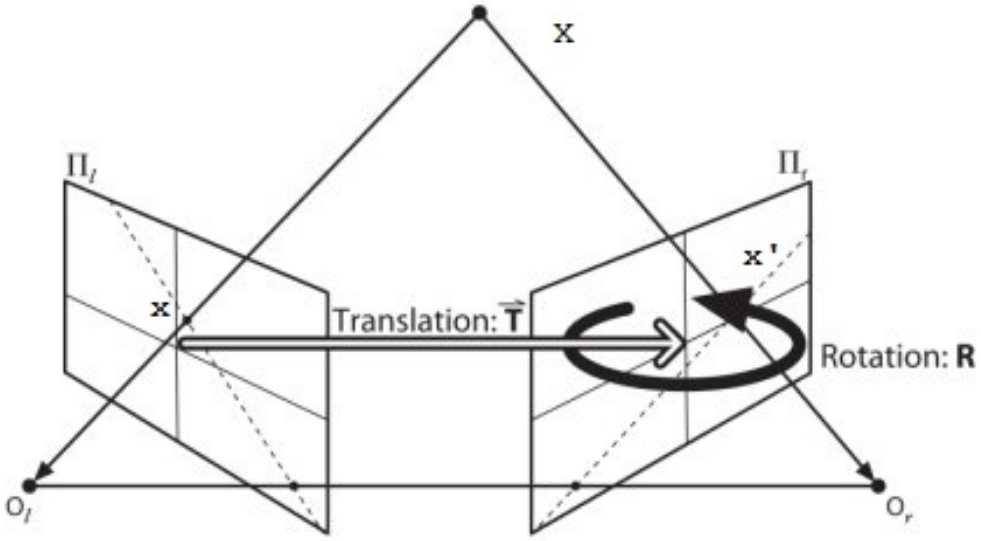

Figure 3.4

Camera relative position [9]

Then, we take the SVD decomposition of E, i.e.,

$$SVD(E) = U \cdot S \cdot Vt \tag{3.6}$$

and construct a rotation matrix R of 3×3 and a translation matrix T of 3×1 as follows:

$$R = U \cdot W \cdot Vt, \quad T = U \cdot [0, 0, 1]^T, \tag{3.7}$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

After that, we can obtain the relative camera pose matrix P between two images. In particular, by considering one image as the reference with no rotation and translation, matrix P can then be composed of R and T in Equation (3.7):

$$P = [R|T]. \tag{3.8}$$

Finally, based on matrix P, we can estimate the 3D position X of the 2D point x via triangulation. The basic ideas is to find the intersection of two rays, i.e., $O_l x$ and $O_r x'$, as shown in Figure 3.4 which go through the cameras' centers based on the projection equation [23]:

$$\begin{bmatrix} x' \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ 1 \end{bmatrix} \tag{3.9}$$

where X is the 3D coordinates and a 3×1 vector.

The above process is about the reconstruction based on two images. Given a group of images $\{I_0, I_1, \cdots, I_t\}$ from time 0 to time t, we need to find the camera's pose matrices $\{P_1, \cdots, P_t\}$ between two consecutive images. Note that we usually set $P_0 = [I|0]$.

3.3 Description of Our Proposed Realtime3D System

In this section, we present the details of our realtime 3D reconstruction scheme called Realtime3D. The main idea of Realtime3D is to intelligently outsource the most expensive computation of a 3D reconstruction scheme, which is chosen as SfM [50] in this study.

Specifically, SfM reconstructs a 3D space from two or more images [22] in four steps [18]: 2D feature extraction, 3D points estimation, bundle adjustment, and dense cloud expansion. However, the traditional SfM is an off-line scheme, and difficult to directly serve as a realtime tool to deal with live images from the onboard cameras. We modify and improve the original scheme for our UVS application, and utilize parallel computing and cloud computing to accelerate the computing speed. In what follows, we first describe the key processes in each step of the proposed Realtime3D system, and then summarize the whole system.

3.3.1 2D Feature Extraction

As mentioned in Section 3.2.3, we first need to determine the fundamental matrix F between two images in order to reconstruct the 3D space which can be obtained by finding at least eight feature point pairs in the two images. Although it might seem easy to identify the same features by human eyes on two images, due to the camera's rotation, translation, noise, and exposure, the same features may have very different shapes and colors in two images. There are several feature selection technologies in the literature, including FAST [55], ORB [56], SIFT [47] and SURF [6], and we need to find a scheme with balanced performance and computational effciency. Among these different technologies, FAST works very fast, but it is not very accurate; ORB is a little slower than FAST but more accurate; SIFT is the slowest but has the best performance, which has been widely used in the original SfM. In contrast, SURF is slower than FAST and ORB, but much faster than SIFT.

In the meantime, SURF's performance is a little worse than SIFT's, but much better than FAST's and ORB's. In this study, we develop our scheme based on SURF.

Specifically, to reduce the feature selection time, not all input images will be processed. The difference of two consecutive frames will be calculated, and only the images which have enough difference will be chosen for feature extraction. In so doing, we can prevent the system from reconstructing the same scene repeatedly when the UVS is static. Besides, instead of always employing feature selection algorithms, another approach is to use optical flow to track the features in the live image stream. Optical flow refers to the movement of the features in the consecutive frames because of the cameras' and/or the features movements. It can track the features more efficiently and accurately. One popular scheme is Lucas-Kanade method [8] which is a least squares fitting technology. The disadvantage of this method is that it gives poor results when the motion is too fast or some features disappear. We propose to combine optical flow and feature selection together to better extract features. In particular, we first employ a feature selection algorithm, i.e., SURF, to extract the features of an image, and then adopt Lucas-Kanade method method to track the features in the following images until the performance does not meet a certain threshold. In that case, we repeat the above process again and proceed.

3.3.2 3D Point Estimation

After extracting the features, the next step is to link the corresponding features in two images, and then recover the 3D points.

Speciafically, in the case that features were obtained by the Lucas-Kanade approach, the feature points in different images are already linked. Otherwise, i.e., when features were extracted by using the SURF algorithm, we need to find the matching pairs between these features in two images based on their descriptors, which is called feature matching. We employ SURF [6] descriptor in our scheme. It is calculated based on Haar wavelet responses of square regions around the features. One popular feature matching is FLANN [45], short for Fast Library for Approximate Nearest Neighbors. It is much faster than the native brute force search, especially when the number of features is large. FLANN mainly consists of three steps

- Build a K-Dimension(KD) tree for image $I_j's$ features.
- For each feature in image I_i, find K neighbors in the KD-tree of I_j based on the distance between their descriptors.
- Find the feature out of K features that has the shortest distance from the feature in image I_i and make them a pair.

After feature matching, we have the paired features, base on which we can find the fundamental matrix F. However, due to the noise and possible mismatching of the features, common methods like [9] can lead to inaccurate results. We apply another method called Random Sample Consensus (RANSAC) [11], which can well account for noisy raw data with a lot of outliers. This main idea of RANSAC is as follows:

- Initiate an F by randomly selecting 8 feature pairs, and set it to the optimal $F_{optimal}$.
- Calculate the overall error of the other pairs with F.
- Find another F' by selecting 8 feature pairs and calculate the overall error of this F'.
- If the error of this F' can beat that of the optimal $F_{optimal}$, set the optimal $F_{optimal}$ to F'.

- Jump to Step 3 until the $F_{optimal}$ converges or the iteration number reaches a predefined number.

Morever, we parallelize this process since step 3 can be conducted in parallel, which can speed up the computation.

Finally, based on F, we can calculate the pose matrix P based on SVD decompostion [9] and solve for the 3D space coordinates of the matched feature pairs, i.e., x in Equation (3.9) by the linear least squares(Linear-LS) method [23].

3.3.3 Bundle Adjustment

After the previous two steps, we can add more and more 3D points and form a sparse 3D point cloud. However, errors and noises usually make new points' coordinates not well aligned with the previous points', which results in distorted 3D models. Therefore, to project all the whole 3D points into a global coordinate system and optimize the related parameters, we apply a refinement of the reconstruction [4], which is called bundle adjustment. In this refinement, the object is to minimize the re-projection error from 3D space to 2D space. Assume that we have processed m images to get n 3D points. Then the cost function of reprojection is defined as

$$\min_{P_j, X_i} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} w_{ij} d(\Pi(P_j, X_i), x_{ij})^2 \tag{3.10}$$

where $\Pi(\cdot)$ is the re-projection function of point i on image j defined as Equation (3.9), d is the Euclidean distance function between the re-projected 2D position of the 3D point X_i and the corresponding 2D observed point x_{ij}, and w_{ij} is an indicator variable which equals

1 if point i is visible in image j and 0 otherwise which can account for the missing 2D points in some images.

The above optimization can easily involve tens of thousands or even more nonlinear equations which is very time-consuming. We employ the Levenberg-Marquardt (LM) [44] algorithm to solve this nonlinear least squares problem, which can coverage fast with the 3D points obtained in the previous section being the initial starting point [22]. In particular, the problem can be formulated as [75]:

$$(P_0^*, ..., P_{m-1}^*, X_0^*, ..., X_{n-1}^*) = \arg\min_{(P_j, X_i)} \sum_{j=0}^{m-1} f_j(P_j, X_0, ..., X_{n-1}) \tag{3.11}$$

where $(P_0^*, ..., P_{m-1}^*, X_0^*, ..., X_{n-1}^*)$ are the estimated pose matrices and 3d positions, and f_j is the overall re-projection error on image j of all the 3D points regarding the pose matrix P_j

$$f_j(P_j, X_0, ..., X_{n-1}) = \sum_{i=0}^{n-1} \omega_{ij} d^2(\Pi(P_j, X_i), x_{ij}). \tag{3.12}$$

We further define a $\mathbf{f}$ which is a vector of re-projection errors over all the images:

$$\mathbf{f}(\{P_j\}, \{X_i\}) = \left[f_0(P_j, \{X_i\}), \cdots, f_{m-1}(P_j, \{X_i\}) \right]. \tag{3.13}$$

Then, we can solve this optimality problem Equation (3.11) in an iteration way by updating the previous parameters by δ^*, i.e.,

$$(\{P_j\}, \{X_i\}) \leftarrow (\{P_j\}, \{X_i\}) + \delta^*. \tag{3.14}$$

δ^* can be obtained by solving the following linear least squares optimization problem:

$$\delta^* = \arg\min_{\delta} \|\mathbf{J}\delta + \mathbf{f}\|^2 + \lambda \|D\delta\| \tag{3.15}$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathbf{f}$, D is a non-negative diagonal matrix which is the square root of the diagonal of $J^T J$, and λ is a nonnegative parameter to control the optimization. The above equation can be solved below:

$$(J^T J + \lambda D^T D)\delta = -J^T f \tag{3.16}$$

This is a large linear system that can be solved by using the iterative Conjugate Gradient (CG) approach. This CG method can also be implemented in a parallel way. Furthermore, we can simplify the CG approach by using the implicit Schur algorithm, because the augmented hessian matrix is sparse and can be transferred to a Schur complement format.

Although we consider that the onboard CPU has multi-core and hence parallel computing capability, such a bundle adjustment optimization problem is still very expensive and time-consuming which does not satisfy our realtime requirement. Therefore, we outsource this problem to the cloud. What is more, we notice that complete bundle adjustment over all the input images is not always necessary, because the current scene is more important for UVS. Thus, a rough 3D scene e.g., for autonomous navigation, can be constructed by several hundreds of features, which normally can be obtained from only a few images within a certain time period. Based on such observations, we can significantly reduce the size of the problem and make it a realtime process. Furthermore, the number of the iterations in the LM method is also considered to save the computing time. Particularly, the re-projection errors are usually significantly reduced after a few of iterations. Some small errors will not influence the performance of applications like autonomous navigation too much.

3.3.4 Dense Cloud Expansion

After bundle adjustment, we can now get a sparse point cloud which describes the raw structure of the front scene of the camera. The sparse point cloud has limitations, and it cannot give the details of the scene. Maybe it is sufficient for applications like autonomous navigation, but further processing is needed for other tasks like 3D modeling. We can expand a sparse poind cloud to a dense point cloud, and then conduct polygonal surface reconstruction to get a more fine-grained 3D sparse model. Note that only a small number of images need to be sent from the UVS to the cloud for dense cloud expansion, which are determined based on the features' movements and the current network speed. In particular, in the cloud, we employ an algorithm based on patch-based multi-view reconstruction [18], where the main idea is as follows:

- Matching: match the small size (2*2) feature patches to those in the sparse 3D model.
- Expansion: iteratively add new neighbors to existing patches until they cover the surfaces visible in the scene.
- Filtering: Outliers lying outside (left) or inside (right) the correct surface are filtered.
- Polygonal surface reconstruction: turn our collection of patches into surface meshes for image-based modeling applications.

3.3.5 Summary of the Proposed Realtime3D

In this section, we give a brief summary of the proposed Realtime3D scheme. Specifically, the onboard computer receives live images from its onboard camera. First, Realtime3D performs preprocessing to decrease the noises and enhance the edges in an image by employing the bilateral filter [24] and the Sobel operator method [61]. Second, Realtime3D selects key frames based on the camera' movements and the time. The scheme

chooses the first key frame as the initial view, uses it to help build the global coordinates, and employs SURF to extract the features on this initial key frame. Third, the scheme employs the Lucas-Kanade algorithm to track these SURF features in the following new images. If there are not enough matched features found by the Lucas-Kanade algorithm, Realtime3D uses SURF to extract the features and SURF feature descriptor to conduct feature matching based on FLANN, and employs RANSAC to find F subsequently. Otherwise, Realtime3D checks if the UVS has moved enough distance. If so, it adopts RANSAC to find the fundamental matrix F. If not, the scheme processes the next key image. After F is found, Realtime3D utilizes the linear lease squares method to project the 2D points into a 3D space.

The above computations are done locally. Next, Realtime3D sends the 3D points to the cloud, where bundle adjustment is conducted with the Levenberg-Margquardt algorithm. Parallel computing is designed to speed up the process, so that a sparse point cloud can be constructed efficiently. Furthermore, the scheme selects very crucial images and transmit them from the UAS to the cloud, which are used to improve the sparse point cloud and construct a dense point cloud. The detailed flow chart of Realtime3D is shown in Figure 3.5.

3.4 Experiment Results

In this section, we analyze the performance of our proposed Real3D system through extensive experiments. As will be seen later, our proposed system gives very accurate and stable 3D point cloud results.

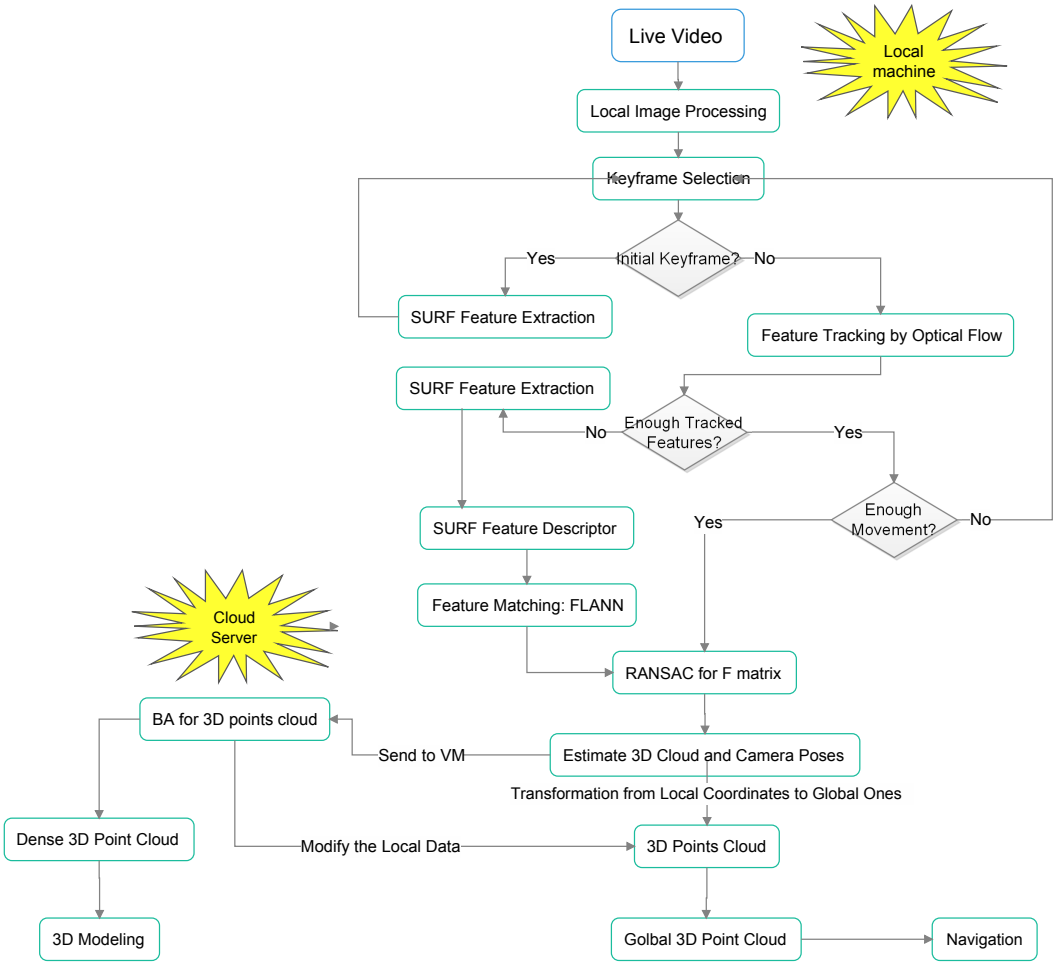

Figure 3.5

Flowchat of 3D Estimation

3.4.1 Experiment Environment and Setting

We implement both the local computing and the cloud computing parts of Realtime3D in Python 2.7, and most of the computer vision algorithms such as SURF, LK, BA are implemented by OpenCV 3.0. We run the local computing part on a laptop with a dual-core 2.53GHz i5 CPU, 4GB RAM memory, and a 320GB hard disk at 5,400RPM which simulates the onboard computer of a UVS. The local computing part receives live images from a Logitech web camera which simulates the onboard camera of a UVS, and has an image resolution of 480. The cloud computing part is implemented on a virtual machine. In particular, we use another laptop to simulate a real cloud server which has quad-core 2.6GHz i7 CPU, 16 GB RAM memory. It hosts a virtual machine by using Virtual Box which has a quad-core CPU and 8 GB RAM memory. The first laptop uses TCP to connect to the virtual machine via a wireless link.

3.4.2 Performance of Feature Selection

In this section, we test four feature extraction algorithms: FAST [55], ORB [56], SIFT [47] and SURF [6], and show their performance in Table 3.1. we can see that FAST is the fastest algorithm, and SIFT is the slowest one. The features of an image selected by these four algorithm are shown in Figure 3.6[1]. We can observe that 1) although FAST generates the most large features, a lot of them are located in the small same areas; 2) ORB cannot generate enough features, although it is fast; 3) SIFT gives the fewest features which have

[1]Top Left: FAST Features. Top Right: ORB Features. Bottom Left: SIFT Features. Bottom Right: SURF Features

high quality, but it is the slowest; 4) SURF results in a good number of features and can keep a good balance between the computing time and the performance.

Table 3.1

Performance of Feature Extraction Algorithms

	Time(Sec)	Number of Features
FAST	0.00465835929975	1103
ORB	0.0197941781448	500
SIFT	0.261150506978	321
SURF	0.190460370521	875

3.4.3 Performance of 3D Point Estimation

First, in Table 3.2, we compare three feature descriptors' matching performance when we use FLANN as the matching algorithm. Note that the features are all obtained by the SURF feature selection algorithm. According to the result, SIFT descriptor can find the fewest matched features, but takes the longest time, i.e., about 24 seconds. So it cannot be used in a realtime application. BRIEF descriptor is the fastest one, but the quality is very low. As shown in Figure 3.7, we can see that SIFT descriptor gives the best matching accuracy, and the BRIEF descriptor is the worst. Among them, SURF descriptor again keeps the balance between the computing time and the matching accuracy. Although SURF is not very fast for a realtime application, in our proposed Realtime3D, we delete the low quality features before we perform feature matching so that the computing time can be reduced.

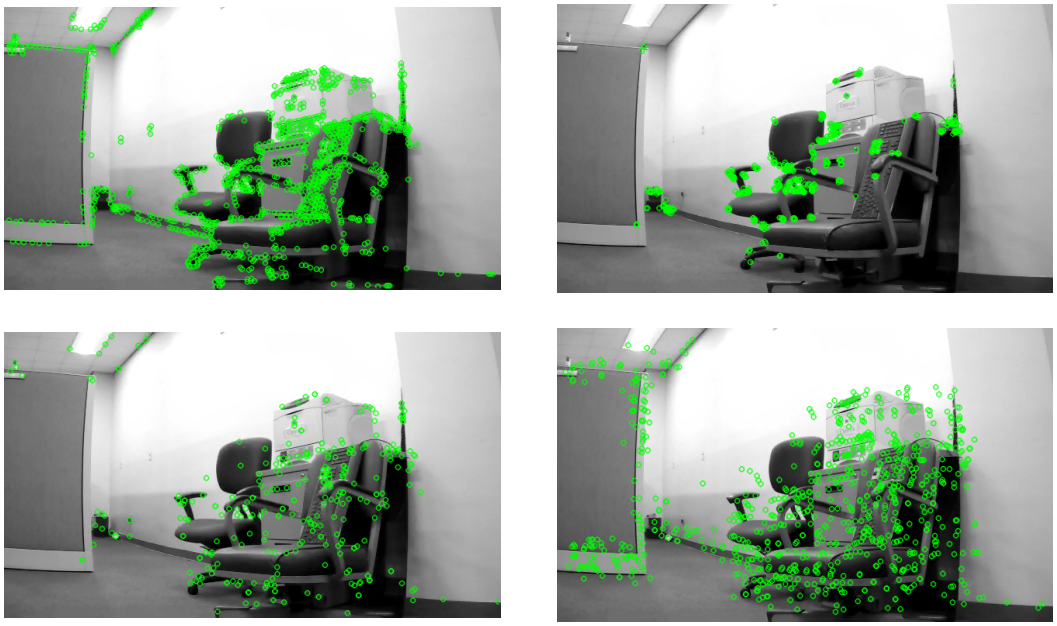

Figure 3.6

Different Kinds of Features

Table 3.2

FLANN based Matcher

	Time(Second)	Number of Matched Features
SIFT Descriptor	24.0184239771	119
SURF Descriptor	1.86921224786	602
BRIEF Descriptor	0.331263716699	453

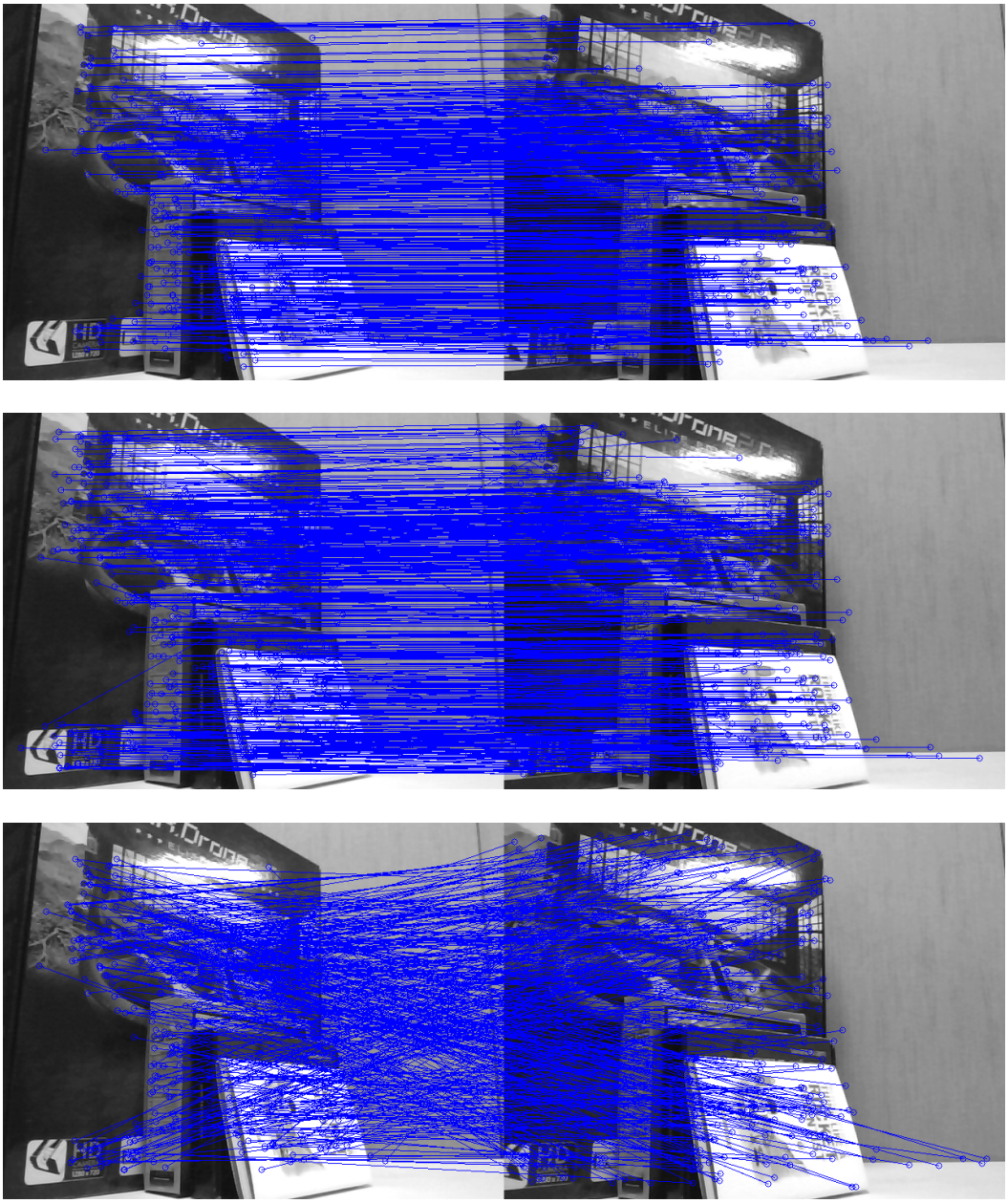

Figure 3.7

Top: SIFT descriptor. Center: SURF Descriptor. Bottom: BRIEF Descriptor

Second, we show that feature tracking with the Lucas-Kanade method is faster than feature matching. The matching performance is also shown in Figure 3.8. In particular, the time needed for the feature tracking is 0.172945539169 seconds with 1695 matched features, which is much faster than feature matching as shown in Table 3.2. This is because feature tracking does not need to calculate the feature descriptors and perform global searching. However, as mentioned before, its disadvantage is that the movement between two consecutive images cannot be too far. Otherwise, mismatch or feature loss will happen. When this situation is detected, the feature matching algorithm has to be applied.

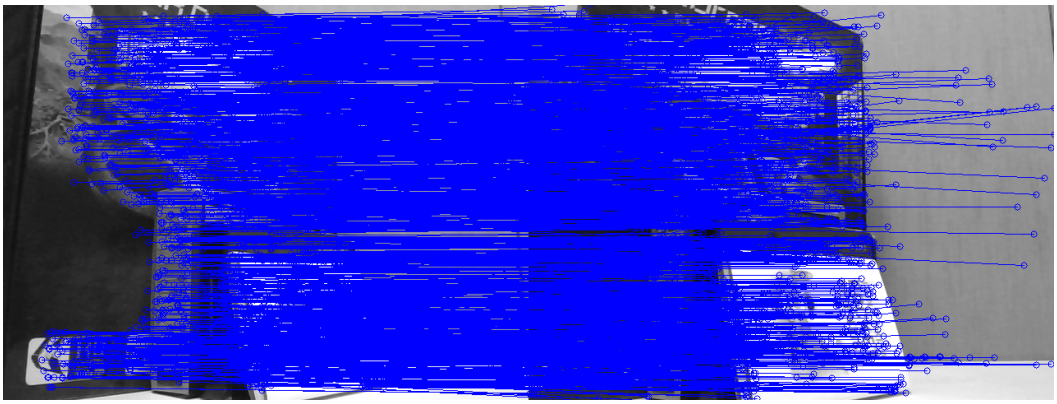

Figure 3.8

Optical Flow Feature Tracking

Third, based on the matched features, we can go ahead to calculate the fundamental matrix F. After that, the camera pose matrix and the 3D point cloud can be obtained. Note that here no bundle adjustment is done yet. As shown in Figure 3.9, we build a sparse 3D

point cloud from two images: Fountain 1 and Fountain 2. The fundamental matrix F is estimated as:

$$F = \begin{bmatrix} 2.87970795e^{-08} & -2.55772771e^{-06} & 8.65910956e^{-04} \\ 6.78077872e^{-06} & 9.22882695e^{-07} & 1.27240360e^{-02} \\ -1.9832146e^{-03} & -1.51672530e^{-02} & 1 \end{bmatrix}$$

.

We can then estimate the relative movement between Fountain 1 and Fountain 2 from this F:

$$R = \begin{bmatrix} 0.9903295 & -0.0217032 & -0.1370275 \\ 0.01594782 & 0.99895005 & -0.04294715 \\ 0.13781564 & 0.04034654 & 0.98963579 \end{bmatrix}$$

$$T = \begin{bmatrix} -0.9777178 & 0.00372538 & 0.20989051 \end{bmatrix}$$

Two views of the 3D points are shown in Figure 3.9[2]. We can see that the basic structure is clear. Note that these 3D points are obtained based on two images. When more images are processed, we need to perform bundle adjustment to transform them into the same coordinate system.

3.4.4 Performance of Bundle Adjustment algorithm

Here, we evaluate the performance of the bundle adjust algorithm. Consider that we have three images: Fountain 1, Fountain 2 and Fountain 3, from which we can have two

[2]Top Left: Fountain 1. Top Right: Fountain 2. Bottom Left: 3D Features View 1. Bottom Right: 3D Features View 2

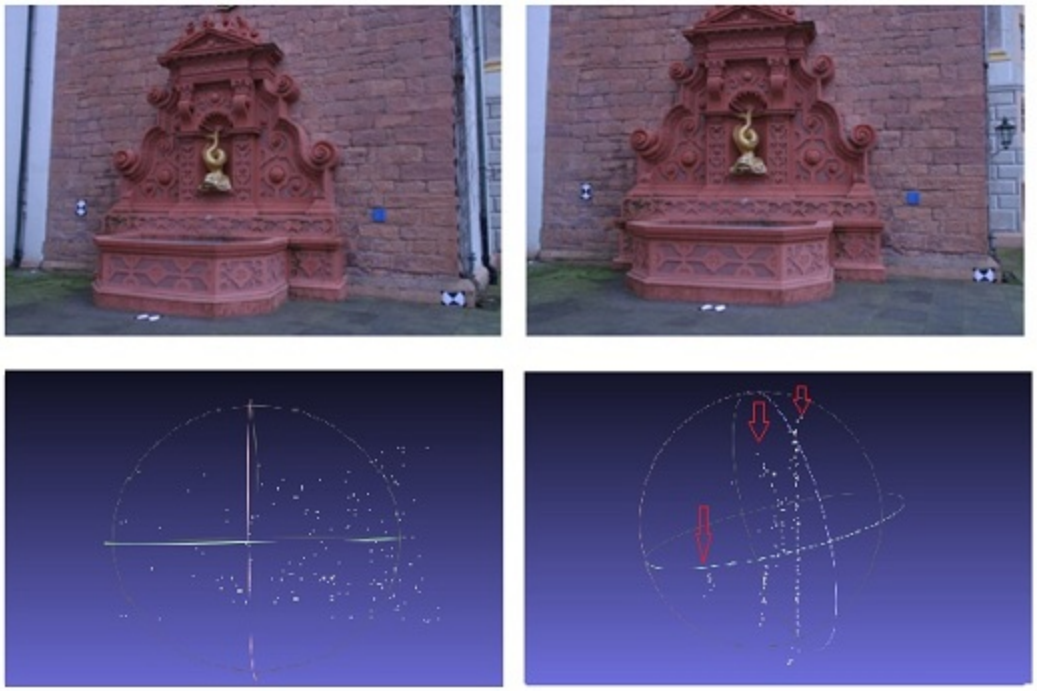

Figure 3.9

Fountains and 3D Features Views 1

pairs: Fountain 1 and Fountain 2, and Fountain 2 and Fountain 3. We can generate two groups of 3D points, which are then transformed into the global coordinations.

The 3D points clouds before and after bundle adjustment are shown in Figure 3.10: Top Left: 3D Features View 1 without bundle adjustment; Top Right: 3D Features View 2 with bundle adjustment; Bottom Left: fountain 1; Bottom Center: fountain 2; Bottom Right: fountain 3. It is clear that bundle adjustment increases the accuracy of the 3D point cloud. Particularly, from the two views on the top, we can see that there are still two point layers before bundle adjustment. After bundle adjustment, this problem gets resolved as shown in the two views at the bottom. However, such bundle adjustment processing is very time consuming. In our scheme, we outsource the bundle adjustment processing to the cloud. Assuming that there are 1000 features points in each of the 5 images, we evaluate the running time of this process both locally and in the cloud sever, by taking into account the uploading time and downloading time. From Table 3.3,we can see that we can save a lot of time by outsourcing bundle adjustment to the cloud, even under very conservative assumptions on the network connection speed.

Table 3.3

Performance of Different Computers for BA

	Time(Second)
Local Computer	23.080846
VM with 4 cores	8.195623 for computation + 5 for communication with 25KBps
VM with 4 cores	8.195623 for computation + 2.5 for communication with 50KBps
VM with 4 cores	8.195623 for computation + 1 for communication with 125KBps

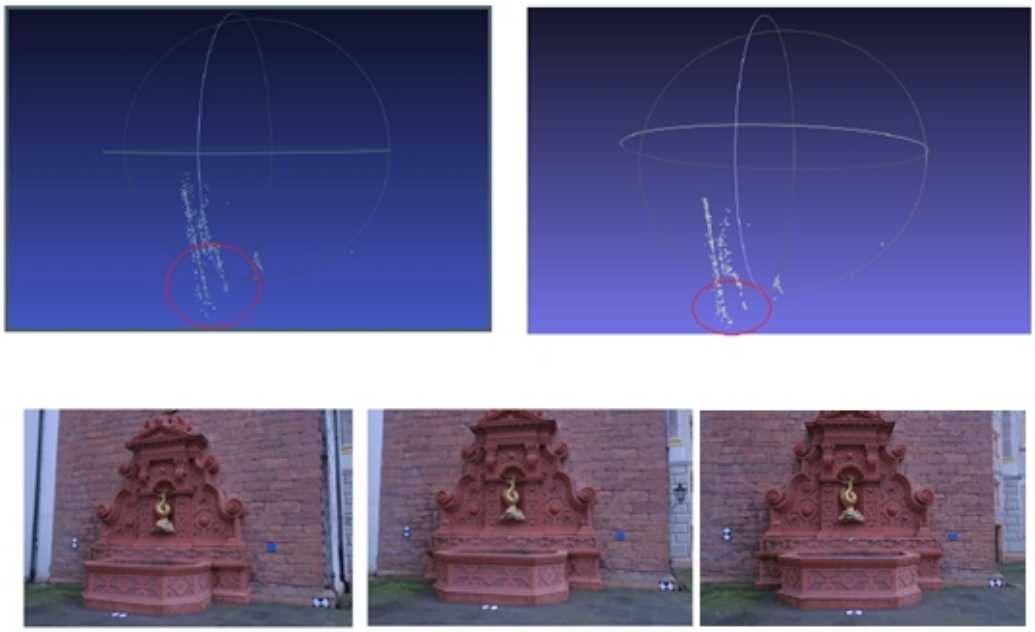

Figure 3.10

3D Features Views and Fountains

3.4.5 Performance of the Realtime3D scheme

In this section, we evaluate the performance of the overall Realtime3D scheme. We set the video to 15 frames per second, and the camera is moving at a constant speed. Besides, about one key frame every four seconds is captured by the Realtime3D scheme to generate the 3D point cloud. Based on our observations, 5 images (4 image pairs) generally can give enough sparse points to represent the front objects in the camera. As shown in Table 3.4, each 3D estimation takes about 3 seconds. It means that if we do not perform bundle adjustment, the scheme can handle 3D estimation in realtime. On the other hand, if we do perform bundle adjustment locally, there will be about 8.76 seconds additional delay and the local computer cannot deal with the new incoming images; in contrast, if we perform

bundle adjustment in the cloud, the delay is less much, i.e., 1.66 seconds. What is more, the local computer and the cloud can do 3D estimation at the same time. In particular, the 3D point cloud and the related parameters can be sent back to the local computer when the system is performing the next frame's processing and not transmitting. Therefore, the proposed Realtime3D can work in realtime with well controlled short delay.

Table 3.4

Time for Each Computation Part

	Time(Second)
Raw 3D estimation for Pair 1	2.43003312859
Raw 3D estimation for Pair 2	3.1542101909
Raw 3D estimation for Pair 3	2.91904988978
Raw 3D estimation for Pair 4	3.2178513536
Bundle adjustment at the local machine	8.75868840451
Bundle adjustment in the Cloud	1.65724234489
Average time for each uploading	0.5
Average time for downloading	1.25

Some detailed graphical results are discussed as follows. In Figure 3.11, Top Left: Features View 1 by Optical Flow; Top Right: Features View 2 by Optical Flow; Bottom Left: Features View 3 by Optical Flow; Bottom Right: Features View 4 by Optical FlowFrom Figure 3.11, we can see four key frames' features and their optical flows from the previous key frames. There are enough number of matching features and very few mismatches. In Figure 3.12, Top Left: 3D View 1; Top Right: 3D View 2; Bottom Left: 3D View 3; Bottom Right: 3D View 4. From Figure 3.12, we can see the sparse 3D point cloud, and know

that there is an object in front of the wall. This sparse 3D point cloud can be used by the onboard navigation system to detect the obstacles.

Moreover, in the cloud, further processing can be conducted to generate the dense 3D point cloud which can be used for other applications like virtual reality. Particularly, each key frame is less than $100\,\mathrm{KB}$. So if the Internet speed is higher than $125\,\mathrm{KBps}$ which is reasonable in the current wireless networks, these key frames can be uploaded to the cloud in 1 second. Four key frames can be used to generated a dense 3D point cloud as shown in Figure 3.13 which takes about $20\,\mathrm{seconds}$. We can easily observe that we have many more details about the surface of the books and the wall.

3.5 Conclusion

Realtime 3D space reconstruction system design emerges as a very interesting and important problem because it has very wide applications. In this paper, we design a realtime 3D reconstruction system called Realtime3D by taking advantage of cloud computing, parallel computing and image processing. In our extensive experiments, the results show that our system can generate good sparse and dense 3D point clouds in realtime.

Figure 3.11

Features views

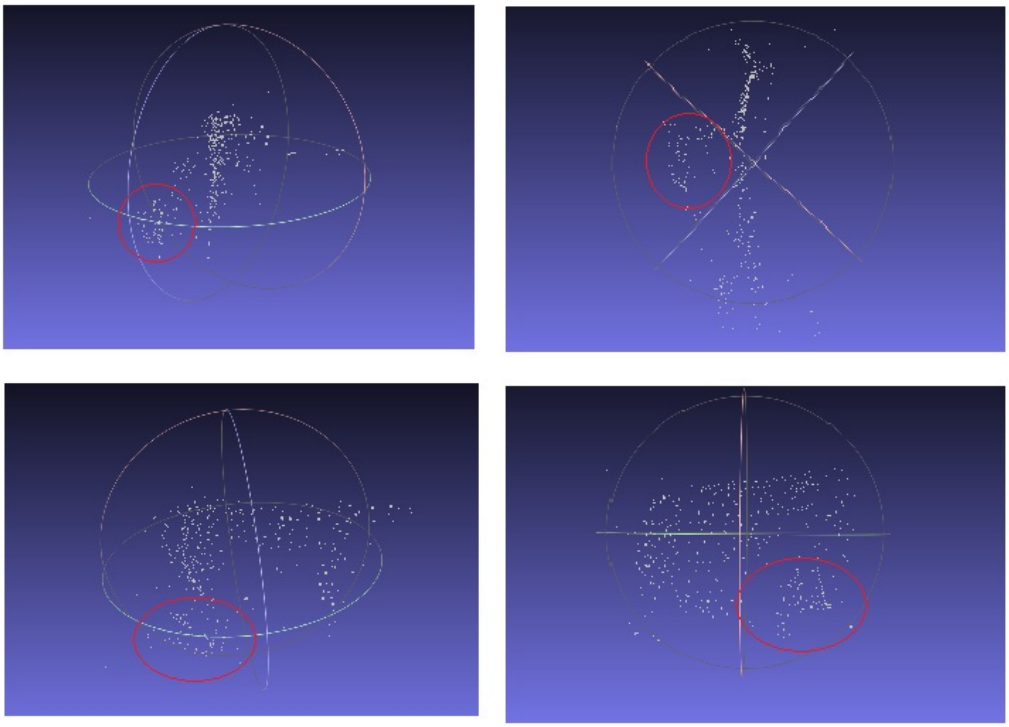

Figure 3.12

3D views

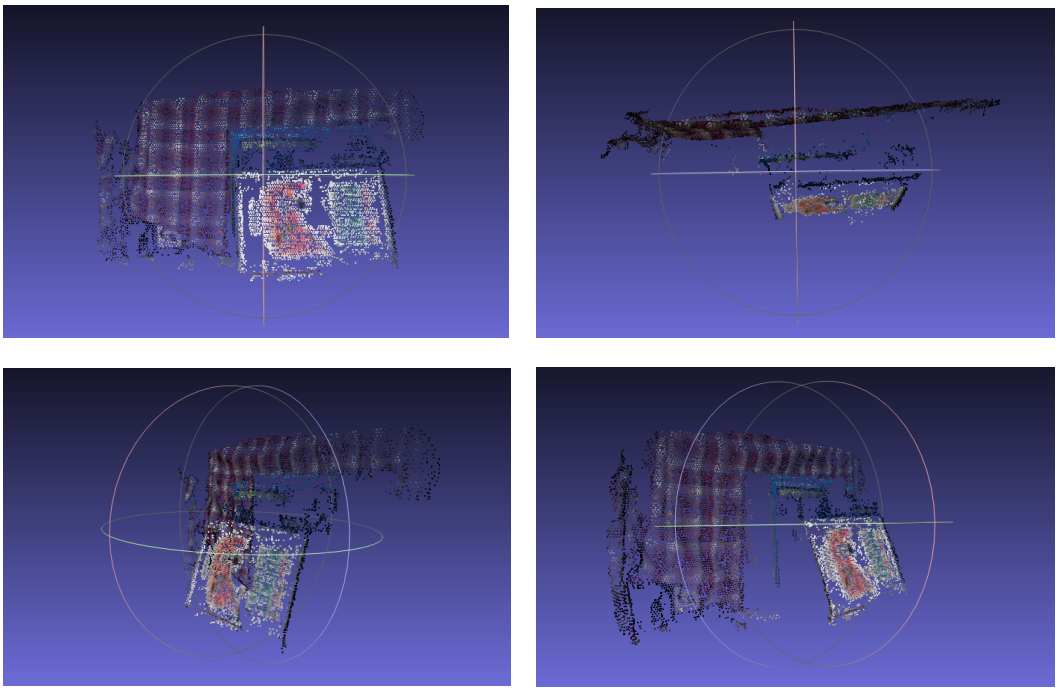

Figure 3.13

Dense 3D Point Cloud

CHAPTER 4

CONCLUSIONS

This dissertation studies some technologies which are key to develop UVS. The first topic is a localization system. When the UVSs cannot utilize the GPS signal, we propose a new system to localize the UVS with high accuracy. Without additional devices or specific modules, the new propose system fuse Received Signal Strength Indication (RSSI) and Inertial Navigation System (INS) which are normally mountable on the UVS. We design an adaptive and efficient algorithm which can be used in the dynamic communication environments. Extensive simulation results prove such a conclusion. In the second topic, we propose a 3D reconstruction scheme based on cloud computing by UVS onboard devices without extra ones. Basically, we can reconstruct scene geometry and camera motion from two or more images[22] in four steps[18]. Such work needs a lot of computational resource, where is difficult to realize in small UVS. High speed Internet and cloud computing service give us a chance to get enough power to do this. A good outsourcing scheme can accelerate the whole processing and have good performance, and enable the other high level tasks.

4.1 Contributions

To further extend the work in the proposal, we finish some improvements and experiments here. Based on the proposal, the accumulated errors becomes even more serious in MEMS IMUs. The reason is that the thermo-mechanical white noise and the bias errors account for a significant fraction of the measurement error [74]. Although there are a lot of methods which strike minimizing the impact of the measurement error, it still remains a big problem for the MEMS IMUs. The experiments shows that IMU in the smart phone (iPhone 5s) cannot give good performance when it uses the ordinal equations (2.12)-(2.14), even we use the proposed scheme in this dissertation. The reason is that the magnitude of the noise generated from the low cost MEMS chips is very large, and the error of the position will be accumulated over $16\,\mathrm{m}$ even in $30\,\mathrm{sec}$. We use some well designed solutions which are employing some heuristic methods like gait tracking algorithm based on the papers [60, 40] adaptive Kalman filter (AKF) from [42], and L_1 regression based approach [30, 41] to handle such problems. Real environment experiments is designed and the results is compared with the other previous systems including a basic fingerprint system based on k-closest neighbors method [76] and a hybrid RSSI and INS system with Kalman filter without channel model update (KF)[80].

Another topic is 3D reconstruction based on cloud computing. There are four steps in the whole scheme: 1, Track 2D features;2, Estimate 3D points; 3, Optimize: Bundle Adjust;4, Fir Surfaces. The new idea is to outsource the most expensive computations to the cloud servers, while the local machine deal with the rest tasks. In this scheme, we identify the particular task: BA for cloud servers. The native way of sending all raw data

to the cloud is not efficient for a UVS. Since the raw data's size is huge, and the Internet connection cannot afford such a high speed all the time. Besides, Internet disconnection may get the UVS into the troubles without enough data update. The proposed scheme aims to keep the system work even while the connection is down, or the Internet speed is low. The performance and the robustness are issues that we carefully consider. The related simulation and tests is scheduled after the theoretic design is done. Compared with previous method which may not be realtime or use some extra devices, simulations show that Real3D has much better performance in time constrain with result errors is acceptable with single ordinary cameras.

4.2 For Further Research

In the future, we will complete our previous work, try to implement them on the real UVS, and show their values. Some improvements and experiments are needed to raise the systems' performance, and we will find current designs' defects. The real UVS have more limitation on the safety and robustness issues. Solving those problems will be milestones towards Autonomous Unmanned Vehicle Systems.

REFERENCES

[1] "Junaio 2.0 First Indoor Social Augmented Reality App at SXSW With Developers API,", Available from http://arabcrunch.com/2010/03/junaio-2-0-first-indoor-social-augmented-reality-app-at-sxsw-with-developers-api.html, 10 Mar, 2010.

[2] G. Agamennoni, J. I. Nieto, and E. M. Nebot, "An outlier-robust Kalman filter," *IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011.

[3] P. F. Alcantarilla, C. Beall, and F. Dellaert, "Large-scale dense 3D reconstruction from stereo imagery," *5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV13)*, 2013.

[4] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*, Packt Publishing Ltd, 2012.

[5] P. Barsocchi, S. Lenzi, S. Chessa, and G. Giunta, "A Novel Approach to Indoor RSSI Localization by Automatic Calibration of the Wireless Propagation Model," *IEEE 69th Vehicular Technology Conference, 2009*, Barcelona, Spain, April 2009.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, 2008, pp. 346–359.

[7] A. M. Bernardos, J. R. Casar, and P. Tarrío, "Real time calibration for rss indoor positioning systems," *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, Sept. 2010.

[8] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, 2001, p. 4.

[9] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, " O'Reilly Media, Inc.", 2008.

[10] P.-J. Bristeau, F. Callou, D. Vissière, and N. Petit, "The Navigation and Control Technology Inside the AR. Drone Micro UAV," *18th IFAC World Congress*, 2011, pp. 1477–1484.

[11] O. Chum and J. Matas, "Matching with PROSAC-progressive sample consensus," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 1, pp. 220–226.

[12] P. Coronel, S. Furrer, W. Schott, and B. Weiss, "Indoor location tracking using inertial navigation sensors and radio beacons," *The Internet of Things*, Springer, 2008, pp. 325–340.

[13] B. S. Dhillon, *Robot reliability and safety*, Springer Science & Business Media, 2012.

[14] A. K. E. A. Hansen, J. Shi, "A POMDP Approach to Influence Diagram Evaluation," *25th International Joint Conference on Artificial Intelligence(IJCAI-16)*, New York, 2016.

[15] F. Evennou and F. Marx, "Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning," *Eurasip journal on applied signal processing*, vol. 2006, 2006, pp. 164–164.

[16] L. Fairfax and F. Fresconi, "Loosely-coupled GPS/INS state estimation in precision projectiles," *IEEE/ION Position Location and Navigation Symposium (PLANS)*, Myrtle Beach, SC, April 2012.

[17] Y. Fu, C. Wang, W. Tian, and M. Shahidehpour, "Integration of Large-Scale Offshore Wind Energy via VSC-HVDC in Day-Ahead Scheduling," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 2, 2016, pp. 535–545.

[18] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 8, 2010, pp. 1362–1376.

[19] S. M. George, W. Zhou, H. Chenji, M. Won, Y. O. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, "DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response," *IEEE Communications Magazine*, vol. 48, no. 3, 2010, pp. 128–136.

[20] H. Gonzalez-Jorge, B. Riveiro, E. Vazquez-Fernandez, J. Martínez-Sánchez, and P. Arias, "Metrological evaluation of microsoft kinect and asus xtion sensors," *Measurement*, vol. 46, no. 6, 2013, pp. 1800–1806.

[21] E. Guizzo, "How googles self-driving car works," *IEEE Spectrum Online, October*, vol. 18, 2011.

[22] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[23] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, 1997, pp. 146–157.

[24] K. He, J. Sun, and X. Tang, "Guided image filtering," *European conference on computer vision*. Springer, 2010, pp. 1–14.

[25] J. Heikkila and O. Silvén, "A four-step camera calibration procedure with implicit image correction," *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997, pp. 1106–1112.

[26] S. Hemachandra, T. Kollar, N. Roy, and S. Teller, "Following and interpreting narrated guided tours," *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[27] E. P. Herrera, R. Quirós, and H. Kaufmann, "Analysis of a kalman approach for a pedestrian positioning system in indoor environments," *Euro-Par 2007 Parallel Processing*, Springer, 2007, pp. 931–940.

[28] J. D. Hol, F. Dijkstra, H. J. Luinge, and P. J. Slycke, "Tightly coupled UWB/IMU pose estimation system and method,", June 2012, US Patent 8,203,487.

[29] B. Jia, S. Cai, Y. Cheng, and M. Xin, "Stochastic collocation method for uncertainty propagation," *AIAA/AAS Astrodynamics Specialist Conference, Minneapolis, Minnesota*, 2012.

[30] Y. Kaneda, Y. Irizuki, and M. Yamakita, "Design method of robust Kalman filter via l_1 regression and its application for vehicle control with outliers," *38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012.

[31] A. Kargarian, Y. Fu, P. Liu, and C. Wang, "A system of systems engineering approach for unit commitment in multi-area power markets," *2014 IEEE PES General Meeting— Conference & Exposition*. IEEE, 2014, pp. 1–5.

[32] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume III: Practical Algorithm Development*, vol. 3, Pearson Education, 2013.

[33] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[34] K. Kurashiki, T. Fukao, K. Ishiyama, T. Kamiya, and N. Murakami, "Orchard traveling UGV using particle filter based localization and inverse optimal control," *IEEE/SICE International Symposium on System Integration (SII)*, Shendai, Japan, Dec 2010.

[35] J. Li and D. Xiu, "A Generalized Polynomial Chaos Based Ensemble Kalman Filter with High Accuracy," *Journal of Computational Physics*, vol. 228, no. 15, 2009, pp. 5454–5469.

[36] X. Li, "RSS-Based Location Estimation with Unknown Pathloss Model," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 12, December 2006, pp. 3626–3633.

[37] B.-C. Liu, K.-H. Lin, and J.-C. Wu, "Analysis of hyperbolic and circular positioning algorithms using stationary signal-strength-difference measurements in wireless communications," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 2, March 2006, pp. 499–509.

[38] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, 2007, pp. 1067–1080.

[39] T. Lozano-Perez, I. J. Cox, and G. T. Wilfong, *Autonomous robot vehicles*, Springer Science & Business Media, 2012.

[40] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics (ICORR)*, ETH Zurich Science City, Switzerland, June, 2011.

[41] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, 2010, pp. 50–61.

[42] R. K. Mehra, "Approaches to adaptive filtering," *Automatic Control, IEEE Transactions on*, vol. 17, no. 5, 1972, pp. 693–698.

[43] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "UAVs for smart cities: Opportunities and challenges," *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 267–273.

[44] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," *Numerical analysis*, Springer, 1978, pp. 105–116.

[45] M. Muja and D. G. Lowe, "Scalable Nearest Neighbor Algorithms for High Dimensional Data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.

[46] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1498–1505.

[47] P. C. Ng and S. Henikoff, "SIFT: Predicting amino acid changes that affect protein function," *Nucleic acids research*, vol. 31, no. 13, 2003, pp. 3812–3814.

[48] S. OConnor, "Amazon unpacked," *Financial Times Magazine*, vol. 8, 2013.

[49] S. Perez, "Facebook Looking Into Buying Drone Maker Titan Aerospace [Electronic resource]/Sarah Perez, Josh Constine," *TechCrunch/AOL Inc.–Mar*, vol. 3, 2014.

[50] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, et al., "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, 2008, pp. 143–167.

[51] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche, "MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera," *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 83–88.

[52] C. C. Pu, S. Y. Lim, and P. C. Ooi, "Measurement arrangement for the estimation of path loss exponent in wireless sensor network," *7th International Conference on Computing and Convergence Technology (ICCCT)*, Seoul, Korea, December 2012.

[53] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[54] V. Renaudin, B. Merminod, and M. Kasser, "Optimal data fusion for pedestrian navigation based on UWB and MEMS," *IEEE/ION Position, Location and Navigation Symposium IEEE/ION*, Monterey, CA, May 2008.

[55] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV 2006*, Springer, 2006, pp. 430–443.

[56] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[57] J. Salmeron-Garcia, P. Inigo-Blasco, F. Diaz-del Rio, and D. Cagigas-Muniz, "A tradeoff analysis of a cloud-based robot navigation assistant using stereo image processing," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, 2015, pp. 444–454.

[58] T. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma, "A survey of various propagation models for mobile communication," *IEEE Antennas and Propagation Magazine*, vol. 45, no. 3, June 2003, pp. 51–82.

[59] S. Sarkka and A. Nummenmaa, "Recursive noise adaptive Kalman filtering by variational Bayesian approximations," *Automatic Control, IEEE Transactions on*, vol. 54, no. 3, 2009, pp. 596–600.

[60] F. Seco, C. Prieto, J. Guevara, et al., "A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU," *IEEE International Symposium on Intelligent Signal Processing*, Budapest, Hungary, August, 2009.

[61] I. Sobel, "History and definition of the sobel operator," *Retrieved from the World Wide Web*, 2014.

[62] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," *Computers and Communications (ISCC), 2012 IEEE Symposium on*. IEEE, 2012, pp. 000059–000066.

[63] N. Spinrad, "Google car takes the test," *Nature*, vol. 514, no. 7523, 2014, pp. 528–528.

[64] R. G. Stirling, *Development of a pedestrian navigation system using shoe-mounted sensors*, Master of science, University of Alberta, 2003.

[65] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys, "Live metric 3d reconstruction on mobile phones," *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 65–72.

[66] L. Taponecco, A. D'Amico, and U. Mengali, "Joint TOA and AOA estimation for UWB localization applications," *IEEE Transactions on Wireless Communications*, vol. 10, no. 7, 2011, pp. 2207–2217.

[67] P. Tarrío, A. M. Bernardos, and J. R. Casar, "Weighted least squares techniques for improved received signal strength based localization," *Sensors*, vol. 11, no. 9, 2011, pp. 8569–8592.

[68] P. Tarrio, J. Besada, and J. Casar, "Fusion of RSS and inertial measurements for calibration-free indoor pedestrian tracking," *16th International Conference on Information Fusion (FUSION)*, Cairns, Queensland, Australia, July 2013.

[69] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, 1996, pp. 267–288.

[70] J.-A. Ting, E. Theodorou, and S. Schaal, "A Kalman filter for robust outlier detection," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, USA, 2007.

[71] C. Wang and Y. Fu, "Fully Parallel Stochastic Security-Constrained Unit Commitment," *IEEE Transactions on Power Systems*, vol. PP, no. 99, 2015, pp. 1–11.

[72] J. Wang and D. Katabi, "Dude, where's my card?: RFID positioning that works with multipath and non-line of sight," *Proceedings of the ACM SIGCOMM*, Hong Kong, China, August 2013.

[73] O. Woodman and R. Harle, "Pedestrian Localisation for Indoor Environments," *Proceedings of the 10th International Conference on Ubiquitous Computing*, New York, NY, USA, September 2008, UbiComp '08.

[74] O. J. Woodman, "An introduction to inertial navigation," *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, vol. 14, 2007, pp. 15–42.

[75] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3057–3064.

[76] J. Yin, Q. Yang, and L. Ni, "Learning Adaptive Temporal Radio Maps for Signal-Strength-Based Location Estimation," *IEEE Transactions on Mobile Computing*, vol. 7, no. 7, July 2008, pp. 869–883.

[77] C. Zhang, M. Kuhn, B. Merkl, A. Fathy, and M. Mahfouz, "Accurate UWB indoor localization system utilizing time difference of arrival approach," *IEEE Radio and Wireless Symposium*, San Diego, CA, Oct 2006.

[78] Z. Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE*, vol. 19, no. 2, 2012, pp. 4–10.

[79] G. Zhou, A. Liu, K. Yang, T. Wang, and Z. Li, "An Embedded Solution to Visual Mapping for Consumer Drones," *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 670–675.

[80] L. Zwirello, X. Li, T. Zwick, C. Ascher, S. Werling, and G. F. Trommer, "Sensor data fusion in UWB-supported inertial navigation systems for indoor navigation," *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, May 2013.