Doctor of Electrical Engineering

Location Estimation Algorithm Based on the Particle Aided Unscented Kalman Filter for Autonomous Vehicle

The Graduate School
of the University of Ulsan

Department of
Electrical/Electronic and
Computer Engineering
Ming Lin

Location Estimation Algorithm Based on the Particle Aided Unscented Kalman Filter for Autonomous Vehicle

Supervisor: Byeongwoo Kim

A Dissertation

Submitted to
The Graduate School of the University of Ulsan
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Electrical Engineering

by

Ming Lin

Department of
Electrical/Electronic and
Computer Engineering
University of Ulsan, Korea
February 2021

Location Estimation Algorithm Based on the Particle Aided Unscented Kalman Filter for Autonomous Vehicle

This certificates that the dissertation thesis of Ming Lin is approved.

_____

Committee Chair Professor. Han-sil Kim

_____

Committee Member Professor. Byeong-Woo Kim

_____

Committee Member Dr. Su-jin Kwag

_____

Committee Member Dr. Won-Seok Lee

_____

Committee Member Dr. Sung-hyo Son

Department of Electrical/Electronic and Computer Engineering

University of Ulsan, Korea

February 2021

ACKNOWLEDGEMENT

Many people helped me in this thesis work. I want to give sincere gratitude to their kindness and intelligence.

At first, I want to express my thanks to the professors in the Department of Electrical/Electronic and Computer Engineering. They taught me a lot during the whole course. They are so kind, patient, and of course full of knowledge.

Secondly, I am very thankful to my advisor Professor Byeongwoo Kim who provides the opportunity to study in his laboratory. He kindly supported me in financial, spirit motivation, and academic guidance during the whole master and doctoral combined course.

Here I want to say thanks to Jae woo, Yoon. He is really smart and gives me the proper guidance in my thesis work. His positive motivation helped me get through a hard time. Without his kindness, I think it is impossible to complete this thesis work in time.

Then I want to express my greatness to my laboratory members. They always provide kind help when I get into trouble in daily life and always make the laboratory like a big warm home. We take a class and discuss it together. It is so nice and lucky for me to work with every one of you.

Last but not least, I am very grateful to my family and friends. Your kind and positive encouragement help me to complete the thesis. Thanks to my parents with their selfless love, thanks to my wife with always patience accompanied and thank my friends with kind encouragement.

Thanks to my wife(Guohua, Sun) and my parents(Jihong, Jin, Qingsong, Lin, Jinliang, Liu)

ABSTRACT(논문 요약)

200년 전에, 엔진을 가진 자동차가 발명되었다. 이때부터 차량은 더욱 편리 해지고 안전 해졌다.이동체 및 이동 로봇과 같은 지상 이동체가 개발되면서, 이동체에 대한 연구가 많이 진행 되고 있다. 오늘날에는 컴퓨팅 능력과 센싱 기술의 발달로 이동체은 사람뿐만 아니라 알고리즘에 의해서도 스스로 제어 되여 특정 임무를 수행하게 되였다.

알고리즘 기반으로 이동체를 정확하게 제어하려면 이동체의 상태를 정확하게 인식해야 한다. 이동체 상태의 가장 중요한 파라미터 중의 하나는 이동체 위치이다. 일반적으로 위치는 GPS(위성항법장치) 신호를 직접 사용해 취득할 수 있다. 그러나, 도시 환경에서 다양한 노이즈가 GPS신호에 영향을 주고 GPS 신호 다중 경로 라우팅 때문에, 이동체 위치 정밀도 요구에 만족하는 위치 정보를 취득하기 어렵다. 따라서, 이동체는 맵의 도움받아 이동체와 주변 요소의 기하학적 관계를 이용하여 자기 이동체의 위치를 추정할 수 있다. 그러나 맵에도 측정 노이즈가 존재하고, 이동체에 탑재된 센서도 노이즈가 존재할뿐만 아니라 노이즈 특성(가우시안, 비가우시안 등)도 다양하다. 따라서 센서의 가우시안 노이즈, 비가우시안 노이즈 필터링와 다양한 센서 정보를 융합하여 이동체의 위치를 추정하는 알고리즘이 필요하다.

본 연구에서는 센서 융합 알고리즘인 PAUKF를 제안하였다. 맵 정보, 이동체위치 예측 모델, 측정 모델, 다양한 센서를 기반으로 이동체위치를 추정한다. PAUKF의 주요 부분은 두 부분으로 나눌 수 있다. 하나는 PF이고 다른 하나는 UKF이다. PF는 이론적 모든 종류의 노이즈을 처리할 수 있는 Monte-Carlo 방법론의 응용이다. PF는 가상의 파티클을 이용하여, 이동체센서에서 측정한 값과의 오차를 기반으로 자기 이동체 위치를 추정한다. UKF는 개선된 Kalman 필터의 일종이다. UKF는 여러 시그마 포인트를 생성하고 이러한

시그마 포인트를 사용하여 가우시안 및 비가우시안 노이즈를 필터링한다. PAUKF는 PF와 UKF의 특성을 융합하여 비가우시안 노이즈를 효과적으로 필터링하고 차량의 위치를 정확하게 추정할 수 있다. PF 기반으로 맵 매칭 진행하여 먼저 자신의 위치를 추정하고, UKF에서 PF에서 처리한 정보를 이용하여 자신의 위치를 최종으로 추정한다. PAUKF는 맵 정보를 이용하기 때문에, PAUKF는 GPS없는 환경에서도 연속적으로 정확한 위치정보를 취득할 수 있다.

PAUKF는 일종의 일반적인 센서 퓨전 알고리즘으로서, 특정 타겟 하드웨어에만 적용할 수 있는 것이 아니다. 하드웨어 동특성에 따라서, PAUKF의 예측모델만 수정하면 알고리즘을 어떻한 하드웨어 플랫폼에도 다 사용가능하다. 따라서, PAUKF알고리즘은 2가지 방법으로 검증하였다. 한가지는 ROS(로봇 작동 시스템)를 기반으로 실제 UGV를 이용하여 검증하였다. UGV 기반의 실험은 이동체가 UGV 기반의 예측모델을 PAUKF에 적용한 다음, 실내환경에서 제한된 특징점을 이용하여 자기 위치를 교정하는 것이다. 다른 한 가지는 차량 형태의 이동체에 PAUKF를 적용하여 특징점이 많고, 속도가 높은 환경에서 시뮬레이션 검증 진행하여 알고리즘의 성능 검증하였다. 두가지 방법으로 검증한 결과 및 기타 연구 결과와 비교를 통해, PAUKF의 위치 추정 성능을 확인 할 수 있었다.

TABLES OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 The autonomous vehicle

The autonomous vehicle becomes a popular research area in the whole transportation system[1]. The autonomous vehicle refers to the vehicle that moves by its algorithm. Both the unmanned ground vehicle(UGV) and the self-driving car can be regarded as autonomous vehicles as Figure 1 shows.

Figure 1 Autonomous vehicle (UGV, Self-driving car)

The field of automated vehicles is multidisciplinary. It involving transportation systems, automotive engineering, human factors, information technology, control, robotics, communications, energy, security, and social sciences[2]–[5]. The unmanned ground vehicle is developed many years ago. The unmanned ground vehicle is mainly used to convey the material in a modernizing self-service factory and warehouse. By using the unmanned ground vehicle, the factory and warehouse can work without human help. The unmanned ground vehicle is not famous because the unmanned ground vehicle is working in the warehouse or factory. In the contrast, the self-driving car like the autonomous vehicle is coming into the spotlight because it is tested in the public road. The milestone of the autonomous vehicle is the DARPA challenge. The DARPA Grand Challenge is a prize competition for American autonomous vehicles, funded by the Defense Advanced Research Projects Agency, the most prominent research organization of the United States Department of Defense As the DARPA challenge started in 2004, the research the autonomous vehicle is accelerated.

The team from Stanford University won the 2005 contest, and the team from Carnegie Mellon University(CMU) won the 2007 contest. The autonomous vehicle of Google which separated as Waymo was initially developed based on the Stanford and CMU[6]. By the beginning of 2017, the United States National Highway Traffic Safety Administration is expected to authorize the adoption of Vehicle-to-Vehicle technology on all new vehicles, whether autonomous or not. Almost every famous automobile manufacturers like Mercedez-Benz, Tesla, Volvo, GM, BMW, VW, Audi, Toyota, Nissan, Hyundai, and Tier 1 suppliers like Bosch investing much capital in developing fully autonomous vehicle technology[7], [8]. Because the autonomous vehicle is driven by the algorithm, therefore the IT corporations play a main role in the autonomous vehicle algorithm development. The autonomous vehicle from Waymo, the open platform APOLLO which is developed by Baidu, and the open-source autonomous vehicle platform from the autoware foundation accelerates the research of the autonomous vehicle[9]–[15].

Even the unmanned ground vehicle and the car-like autonomous vehicle is different in appearance, the main algorithm of the autonomous vehicle is the same. To guarantee the safety and the convenience of the autonomous vehicle, a top-level algorithm needs to know the information of the ego vehicle and the surrounding environment. If the vehicle is controlled by a human, then all of the driving missions are done by the human. The human driver needs to see the scenes, figures out where the vehicle is memorizes the geometry of the road, and remember the environment. Once the vehicle is driven by an algorithm, the algorithm needs to recognize the state of the vehicle and the surrounded environment to control the vehicle appropriately. To recognize the state of the vehicle and environment, the autonomous vehicle is equipped with several sensors. Usually, the autonomous vehicle is equipped camera sensor, lidar sensor, radar sensor, global navigation satellite system(GNSS) like GPS, GLONASS, Galileo, Beidou, and other regional systems. The state of the vehicle can be estimated by using the mounted sensors mentioned above. The algorithm of the autonomous vehicle can be classified into 4 parts roughly. The 4 parts are localization, perception, planning, and control. The perception module is used for detecting the object and environment, the planning module is used for local path planning and global path planning based on the vehicle position, the control module is

used for publishing the final control value to the vehicle. The control module receives the path from the planning module. The planning module makes local and global path plan based on the location of the vehicle and the surrounding information from the perception module. Therefore, it can be found the location of the vehicle is critical to the autonomous vehicle. Without precise location data, the autonomous vehicle can not work appropriately. The typical autonomous vehicle system is shown in Figure 2.

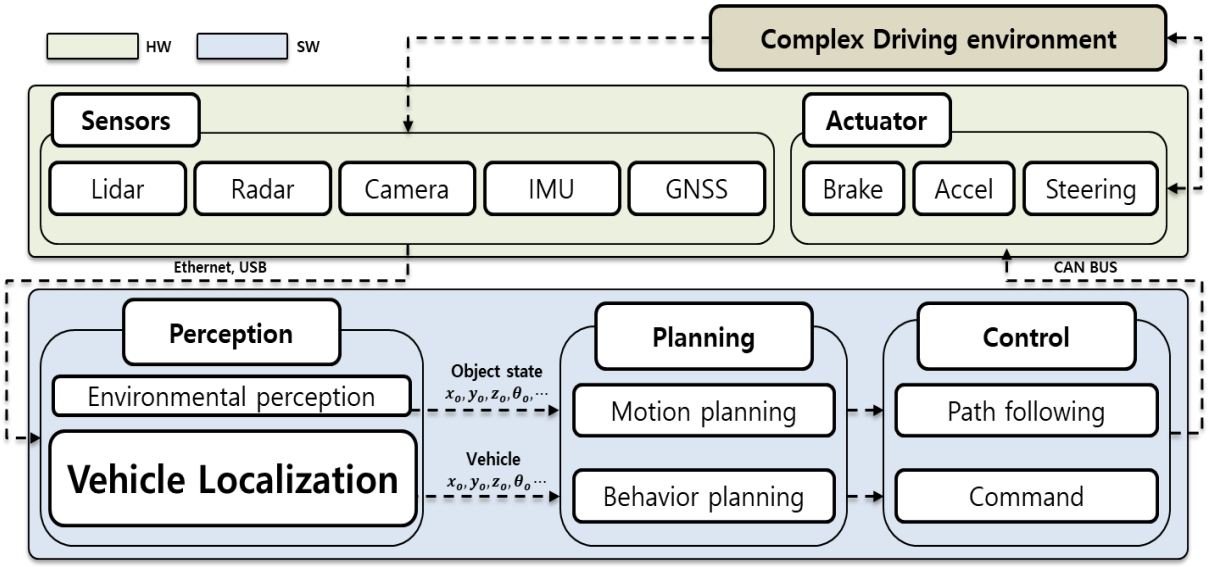


Figure 2 The typical autonomous vehicle system

Thus, extracting the location of the autonomous vehicle is meaningful and critical. Because of the importance of the location data, there are many types of research about the localization algorithm.

The GNSS is the abbreviation of the global navigation satellite system. The GNSS is the standard generic term for satellite navigation systems that provide autonomous geo-spatial positioning with global coverage. This term includes such as the GPS(global positioning system) from the United States, GLONASS from Russia, Galileo from Europe, Beidou from China, and other this kind of system[16]–[21]. GPS is one of the most famous GNSS. Take the GPS as an example. The GPS defines the coordinate of the earth and satellites, then locates the position of the receiver based on the flying time of the signal. The usage of GPS is everywhere in current days. Almost every smartphone

contains a GPS receiver to use the location-based application. Despite the GPS receiver is mounted in the smartphone, however, the precision of the GPS is varied depends on the environment. If the smartphone is used in the tunnel or subway, the receiver can not receive the signal from the satellite. It is not a critical problem for the smartphone user, however, it is critical for the autonomous vehicle. For the same reason, the GPS signal can be worse in the urban city were full of high-rise buildings. The signal of the GPS can be reflected randomly. It means the fly time of the signal changes which also means the position contains much noise. This phenomenon is called multipath error[22]. The update frequency of the GPS is 10Hz usually[23]. If the autonomous vehicle localizes itself with GPS only, then if the velocity is 30m/s(108Km/h), the autonomous vehicle locates itself with an interval of 3 meters. It means even the vehicle changes into the side lane, the algorithm still considers the vehicle is in the same lane. Thus, the GPS is combined with IMU(inertial measurement unit) generally. The update frequency of IMU is 100Hz or higher. It means the algorithm could calculate the position of the vehicle based on the dead reckoning in the interval of the GPS[24]–[26]. However, the data of the IMU sensor accumulates the noise which makes the estimation value worse again. If the GPS signal is good enough, then GPS/IMU combined is enough for localizing the vehicle with a real-time kinematic signal. However, GPS/IMU combined solution still can not works correctly when the vehicle loses a GPS signal for a long time. Therefore, an efficient way that localizes the global position of an autonomous vehicle without a GPS signal for a long time is meaningful research.

The camera sensor is one of the most inexpensive sensors in the autonomous vehicle. The camera sensor provides the RGB value of the real world. In intuition, the camera sensor is the eye of the autonomous vehicle. The object detection based on the mono-camera, dual camera, and multiple cameras has been studied for a long time. The computer vision is used to detect the traffic sign, pedestrian, and other vehicles based on the features and geometry of the object's pixels[27]–[32]. The brightness, reflection, distortion, shelter from other vehicles, and the various objects shape increases the complexity of the detection. The limitation of the camera is the uncertainty of the detection result. The detection result changes a lot based on the light condition and reflection. Besides, because the camera monitors all the scenes on the road in high frequency, the phantom phenomenon can affect the

detection result[33]. The problem of the pure computer vision-based detection is that the relative distance is calculated based on the features in the image. The small noisy movement of the pixels will obtain different results completely. As a result, a reliable approach to detection is needed.

The range sensor provides the relative distance between the sensor coordinate and object. The precision of the relative distance obtained by the sensor like lidar is 1cm. Therefore combining the ability of the object detection of the camera and relative distance measurement from lidar is a suitable approach. Thus camera and lidar fusion-based approaches are studied[34]–[41]. The fusion of the lidar and camera is the most reliable method to detect the elements in the surrounding environment in current days.

Once the perception module extracts the road elements effectively, then the location of the ego vehicle also can be extracted by matching with the map[42]–[46]. The map is made by using the SLAM(simultaneous localization and mapping) algorithm and manufactured by the map engineer with high precision measuring equipment. The map contains every detail of the road, such as the slope of the road, the width of the lane, the position of the traffic sign, the elements of the road, and so on. Every element contains a position labeled value defined by the map manufacturer. Thus the position of the ego vehicle can be obtained by matching the object's position in the map and the detected objects from the ego vehicle perception module. Since the accuracy of the commercial map is verified by the map engineer, thus the precision of the map is reliable. Therefore, if the perception data and the noises can be handled appropriately, the position of the vehicle can be extracted accurately based on the map. However, if the algorithm estimates the position of the ego vehicle only by using the map matching, the jump of the position will happen because of the noise in the map, perception algorithm, and sensor itself. So the localization algorithm should use the information of the vehicle to limit the jump of the estimated location. Usually, the vehicle model has been used to constraint the boundary of the jump of the location. Since the location is estimated based on map matching, the algorithm can estimates the location of the ego vehicle continuously even missing the GPS for a long time. Thus a reliable method that can fuse the elements in the road, elements in the map, and the constraints of the vehicle model is needed.

1.2 Estimate the vehicle location based on sensor fusion

The autonomous vehicle is an extended research area of the unmanned ground vehicle. The location estimation method used in the unmanned ground vehicle is also can be applied to the autonomous vehicle. autonomous vehicle location estimation algorithm level, there are no significant differences between the autonomous vehicle and unmanned ground vehicle except the structure. The only difference between the autonomous vehicle and unmanned ground vehicle is the model of the vehicle. An autonomous vehicle system must control numerous parameters, including speed, orientation, acceleration, and maneuvering. All of these control parameters are controlled by the decision-making module, which handles all perception data from the vehicle and sensors. The perception module determines the relationship between the ego vehicle and the surrounding environment. One of the most important algorithm modules is vehicle localization because all the sensors sense the environment based on local vehicle coordinates[47]. As illustrated in section 1.1, the typical perception sensors of an autonomous vehicle are the camera, range sensor, GPS[48]. GPS is the most commonly used navigation system in an autonomous vehicle. However, because of issues with multipath routing and poor signal availability in cities, relying entirely on GPS is not suitable for localizing vehicles in urban environments. Although differential GPS systems can be used, the high cost and size of these systems limit their implementation[49]. Furthermore, GPS systems cannot be used in tunnels or indoor environments. Vision-based localization has been proposed as a method for localizing vehicles using a low-cost camera. However, the vision-based localization algorithm is easily affected by weather and light conditions, leading to insufficient accuracy and stability[50]–[54] Range sensor-based map matching can yield highly precise results with the help of the map. By contrast, matching requires the environment to be accurately mapped such that the point cloud of the environment does not change. Point cloud matching is expensive and requires considerable power and computation resources[55]–[62]. Thus, developing a localization method that can utilize the vehicle's sensors and road features to achieve precise and stable location performance is necessary for furthering research on the autonomous vehicle. One of the localization solutions that can be used in complex urban environments is vehicle localization based on local sensor systems and information

from the map. Matching an entire point cloud with a map is inefficient; therefore, only the ground truth location map of features is used in this study, for computational efficiency. In previous research, vehicle localization based on vehicle-to-vehicle (V2V) and vehicular ad-hoc network (VANET) communication was proposed. The basic condition is that these algorithms require surrounding vehicles to be equipped with V2V communication equipment, which are then referred to for the infrastructures[63]–[69]. The communication system based localization is not accurate enough and highly depends on the V2V signal. The sigma can be disconnected because of the environment. Thus in this study, we choose matching the detected features with the map for a precise, stable, and communication signal independent approach. Because the vehicle data contain a large amount of noise, and an efficient filtering algorithm is needed to obtain precise localization results.

The methods for vehicle localization have improved considerably over the years. The primary methodology that was used is the probabilistic approach. The Kalman filter (KF) is an optimal estimator that is designed for processing Gaussian noise with mean and variance, and it is an important component in several such approaches[70]. One of the assumptions of the KF is that the noise should be Gaussian. However, in practice, a function like the trigonometric filter renders the Gaussian noise non-Gaussian. Therefore, an extended Kalman filter (EKF), which uses a low order Taylor expansion to linearize the nonlinear (e.g., trigonometric) function, has been proposed. It uses a partial derivative to represent the rate of change of the nonlinear functions, which aims to keep the noise Gaussian. If the state is a vector, then the partial derivative parameters can be assembled into a new matrix, which is called a Jacobian matrix. Generally, to localize the vehicle's position, researchers derive the Jacobian matrix based on the transition and measurement models for handling the vehicle's noisy sensor data[71]–[76]. If an EKF based on a Jacobian matrix approximates a nonlinear function using a high order of Taylor series, it also works well in transforming nonlinear functions into linear ones. The critical problem, however, is that the Jacobian matrix is difficult to derive for complex dynamics. Therefore, a new, sample region-based Kalman filter, which is called the unscented Kalman filter (UKF), was proposed. The UKF performs better than the EKF and KF when the system model is highly nonlinear[77]–[79]. The UKF uses some key points, which are

called sigma points, to approximate the non-Gaussian noise into Gaussian based on the unscented transform. In this way, it can properly capture the nonlinearity. Furthermore, because the UKF approximates the non-Gaussian noise with sigma points, it is easy to combine other information when selecting the sigma points and there is no need to calculate the Jacobian matrix.

The basic assumption of the Kalman filter family is that noise is Gaussian. In the real world, most noise does not have a Gaussian property. For processing non-Gaussian noise, a Monte Carlo-based localization approach, called particle filter (PF), has been proposed [80], [81] The particle filter uses several samples, referred to as particles, to approximate the non-Gaussian property. Because the particles are generated randomly, they can represent the properties of non-Gaussian noise precisely if there are sufficient numbers. However, a vehicle has limited computational resources; therefore, it cannot allow the particle filter to approximate the number of particles. Therefore, there is a trade-off between precision and computational resources when generating an effective particle-based system model. Thus, an extended Kalman filter-aided particle filter, called an extended particle filter (EPF), and an unscented particle filter (UPF), called the Kalman filter-aided particle filter, have been proposed [82]–[84]. Both the EPF and the UPF use system models to generate and update the particles. It should be noted that each particle should compute the sigma points or Jacobian matrix; therefore, both the EPF and UPF are computationally inefficient and difficult to implement [85]–[87].

In this study, a new method, the particle-aided unscented Kalman filter (PAUKF) is proposed for vehicle localization. PAUKF is a kind of hybrid filter framework based algorithm like previous works of literature [88]–[93]. The Nan hu, Chengdong Wu, Tong Jia, and Peng Ji proposed a hybrid filter localization algorithm based on the selection mechanism. They hybrids the extended Kalman filter and H-infinity filter(HEKHF). The limitation of their research is the EKF is expanded in one order of the Tayler series and the EKF can not process the noise with the multi-model property. Amir Panah and Karim Faez proposed a hybrid filter based simultaneous localization and mapping for a mobile robot. They merged the UKF and multi-layer perceptron for the SLAM application. The multi-layer perceptron is used to learning the used as a universal approximator. Compare to this approach,

PAUKF uses the PF as the pre-process by using the detected features and ground truth position of the features in the map. Thus the PAUKF can provide the global location of the autonomous vehicle and correct the state of the vehicle. Inam Ullah, Yu Shen, Xin Su, Christian Esposito, and Chang Choi proposed a localization based on unscented Kalman filter and particle filter localization algorithms. They studied the localization algorithm based on the UKF and PF, however, UKF and PF in their research work independently. Compare to their research, the PAUKF combines the PF and UKF. Seong Jin Kim and Byung Kook Kim proposed a dynamic ultrasonic hybrid localization system for indoor mobile robots. They fused the ultrasonic distance measurement unit based on the EKF algorithm. The EKF has limitations at the noise with multi-modal and highly Non-Gaussian. The Mohammad A. Al-Khedher proposed a hybrid GPS-GSM localization of automobile tracking system. This approach fuse the GPS receiver based on the Kalman filter. The vehicle model and features are not used. What is more, their hybrid GPS-GSM localization algorithm relies on the GPS signal. Compare to this hybrid algorithm, the PAUKF cant provides the precise estimation of location even there is no GPS signal. Carsten Fritsche, Anja Klein, and Dominique Wurtz also proposed a hybrid GPS/GSM localization of mobile terminals using the extended Kalman filter. The limitation of their research is also from the EKF and relies on GPS.

The PAUKF utilizes the particle filter and unscented Kalman filter at the application level. The PAUKF is not a localization algorithm that is derived from the mathematical like the creation of the Kalman filter. The novelty of the PAUKF is it fuses the vehicle model and detected features and the ground truth position of the features in the map. Because the PAUKF fuses the PF and UKF, the PAUKF contains the good property of the PF and UKF. The property of the PF makes PAUKF can handle almost every kind of noise and nonlinearity and the UKF measurement update based on the PF make the final result is smooth and precise. As a result, PAUKF can estimate a system with high nonlinearity and various sources of nonlinear noise more precisely based on the detected features measurement and ground truth of the features comparing to the previous literature. Because each particle does not have to update the sigma points or share the prediction model, this method requires fewer computational resources. The computational burden and precision of PAUKF can be easily

tuned by tuning the quantity of the sigma points and particles. The global position of the vehicle is estimated based on the particle filter, and the unscented Kalman filter carries out the measurement update by using the estimation from the particle filter. The PAUKF considers the geometry affection of the perception and calculates the weight of each particle by using a multivariable normal distribution in three dimensions. We also found that the PAUKF provides feasibility to fuse multisource perception data into the PAUKF framework by weighing the particles.

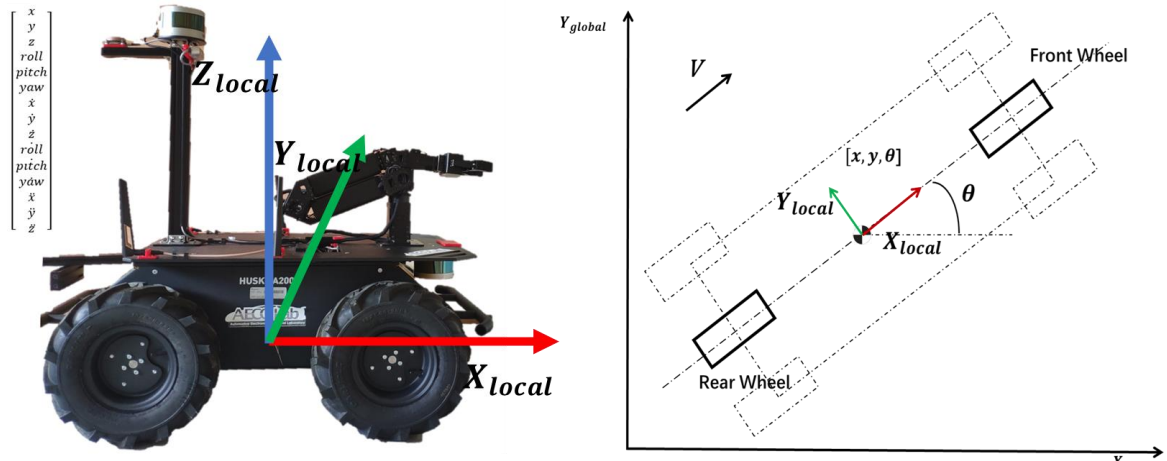
1.3 Outline of the thesis

Section 2 illustrates the methodology of the PAUKF. The PAUKF is evaluated in two kinds of platforms. One is based on a wheeled skid-steer ground vehicle with experiment and another one is based on the car like the autonomous vehicle in the simulation environment. Therefore, section 3 details the experiment environment configurations and analysis of the PAUKF experiment results based on an unmanned ground vehicle. Section 4 details the simulation environment configuration and result analysis of the PAUKF based on the car-like autonomous vehicle. Finally, Section 5 presents the conclusion of this thesis.

Chapter 2 The particle aided unscented Kalman filter based localization

2.1 The PAUKF algorithm

This section describes the implementation of the PAUKF, including particle implementation and PAUKF implementation. Both the PF and UKF are Bayesian-based filters, and the environment is assumed to be Markov, which means that the PAUKF also has a Markov assumption. The PAUKF is a general algorithm that independent of the platform. For example, the PAUKF used in the unmanned ground vehicle and the car-like autonomous vehicle is the same except for the prediction model(vehicle model) and measurement model(sensors used). Therefore, PAUKF can be applied to different kinds of hardware platforms. In this paper, the performance of PAUKF is verified in unmanned ground vehicle platforms with experiments and the car-like autonomous vehicle with simulation as Figure 3 shows.



<Unmanned ground vehicle> <Car like autonomous vehicle>

Figure 3 Different kinds of the verification platform

2.1.1 Particle filter based pre-processing

The particle filter is a Monte Carlo-based method that can handle both Gaussian noise and Non-Gaussian noise[94]. The intuition of the particle filter is shown in Figure 4. The magnetic field is

a kind of probability distribution. No matter how the magnetic field changes, as long as the random iron particle is used, the properties of the magnetic field can be approximated.

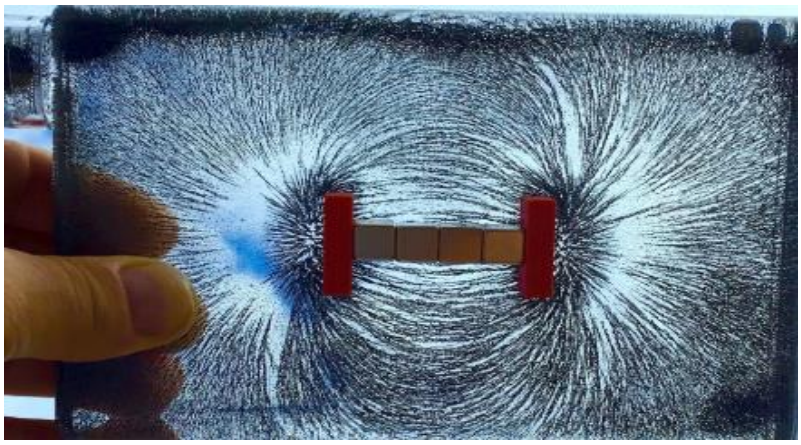


Figure 4 Intuition of the particle filter

Therefore, compared to the widely used filter like Kalman filters, particle filters approximate a large range of probability distributions, not just normal distribution. Once a robot's belief is focused on a subspace of the space of all poses, particle filters are computationally efficient, since they focus their resources on regions in the state space with high likelihood. How to make the algorithm "throw" the particles at the appropriate position for better distribution is an important process. The method which combines particle filter combined with probabilistic models of vehicle perception and motion is called Monte Carlo localization. The particle filter based preprocessing follows the Monte Carlo localization idea.

For better understanding, it is necessary to briefly derive the basics, begging with the Bayes filters. Bayes filters address the problem of estimating the state x of a dynamical system from sensor measurements. For example, in autonomous vehicle localization, the dynamical system is an autonomous vehicle and its environment, the state is the vehicle's pose which usually specified by a position in a two-dimensional Cartesian space and the vehicle's heading direction. The state can be varied according to the specific application. The measurement may include range measurements, camera images, odometry readings, and measurement from the IMU sensor. The Bayesian filters

assume that the environment is Markov which means, past and future data are independent if one knows the current state. The important idea of the Bayes filtering is to estimate the posterior probability density over the state space conditioned on the data. The posterior is typically called as belief. Thus the belief of the x_t is shown as Equation (2.1).

$$Bel(x_t) = p(x_t|y_t, u_{t-1}, y_{t-1}, u_{t-1} \cdots, u_0, y_0) \tag{2.1}$$

The y_t represented as the perceptual data like laser range measurements, and odometry data or controls which carry information about robot motion is represented as u_{t-1}. Bayes filters estimate the belief recursively. The initial belief characterizes the initial knowledge about the system state.

To derive a recursive update equation, the Equation (2.1) can be transformed by Bayes rule to (2.2) where the $d_{0,\ldots t-1}$ represent the data starting at time 0 up to time t.

$$Bel(x_t) = \frac{p(y_t|x_t, u_{t-1}, \ldots, y_0)p(x_t|u_{t-1}, \ldots, y_0)}{p(y_t|u_{t-1}, \ldots, y_0)} = \frac{p(y_t|x_t, u_{t-1}, \ldots, y_0)p(x_t|u_{t-1},\ldots,y_0)}{p(y_t|u_{t-1},d_{0\ldots t-1})} \tag{2.2}$$

By using the Markov assumption, the measurement y_t are conditionally independent of past measurements and odometry readings are given knowledge of the state x_t.

$$p(y_t|x_t, u_{t-1}, \ldots y_0) = p(y_t|x_t) \tag{2.3}$$

Equation(2.3) allows it to conveniently simplify Equation (2.2) into Equation(2.4).

$$Bel(x_t) = \frac{p(y_t|x_t)p(x_t|u_{t-1},\ldots,y_0)}{p(y_t|u_{t-1},d_{0\ldots t-1})} \tag{2.4}$$

To obtain the final recursive form, now have to integrate out the pose x_{t-1} at time t-1, which represents as equation (2.5).

$$Bel(x_t) = \frac{p(y_t|x_t)}{p(y_t|u_{t-1},d_{0\ldots t-1})} \int p(x_t|x_{t-1},u_{t-1},\ldots,y_0)p(x_{t-1}|u_{t-1},\ldots y_0)dx_{t-1} \quad (2.5)$$

The Markov assumption also implies that given knowledge of x_{t-1} and u_{t-1}, the state x_t is conditionally independent of past measurements $y_1\ldots, y_{t-1}$ and odometry readings $u_1\ldots, u_{t-2}$ up to time t-2, that shown as equation (2.6)

$$p(x_t|x_{t-1},u_{t-1},\ldots,y_0) = p(x_t|x_{t-1},u_{t-1}) \quad (2.6)$$

By using the definition of the belief *Bel*, a recursive estimator known as Bayes filter is shown as Equation (2.7). η is the normalizing constant.

$$\begin{aligned} Bel(x_t) &= \frac{p(y_t|x_t)}{p(y_t|u_{t-1},d_{0\ldots t-1})} \int p(x_t|x_{t-1},u_{t-1})Bel(x_{t-1})dx_{t-1} \\ &= \eta p(y_t|x_t) \int p(x_t|x_{t-1},u_{t-1})Bel(x_{t-1})dx_{t-1} \end{aligned} \quad (2.7)$$

Equation (2.7) is the basic idea of various Monte Carlo localization algorithms. The following particle filter also follows the basic idea of the Bayes filter.

To implement Markov localization, we have to know three distributions: the initial belief Bel(x_0), the next state probability p($x_t|x_{t-1},u_{t-1}$) which usually called the motion model, and the perceptual likelihood p($y_t|x_t$) which is usually called a perceptual or measurement model. If the state space is continuous, as is the case in mobile robot localization, implementing the belief update Equation (2.7) is not a trivial matter particularly if one is concerned about efficiency. The idea of MCL(and other particle filter algorithms) is to represent the belief Bel(x) by a set of N weighted samples distributed according to Bel(x) as the Equation (2.8) shows.

$$Bel(x) = \{x^i, w^i\}_{i=1\ldots N} \quad (2.8)$$

Here each x^i is a sample(a state), and w^i are non-negative numerical factors called importance factors, which sum up to numerical value one. In this thesis, the important factors are also called the weight of each sample(particle). In global vehicle localization, the initial belief is a set of poses drawn according to a uniform distribution over the vehicle's universe, annotated by the uniform importance factor $\frac{1}{N}$. The recursive update is realized in three steps, computing the expression in (2.7) from the right to the left.

1. Sample a state x_{t-1} from Bel(x_{t-1}), by drawing a random x_{t-1}^i from the sample set representing Bel(x_{t-1}) according to the (discrete)distribution defined through the importance factors w_{t-1}^i.
2. Use the sample x_{t-1}^i and the action u_{t-1} to sample x_t^i from the distribution p($x_t|x_{t-1}, u_{t-1}$). The predictive density of x_t^i is now given by the product p($x_t|x_{t-1}, u_{t-1}$). The predictive density of x_t^i is now given by the product p($x_t|x_{t-1}, u_{t-1}$)$Bel(x_{t-1})$.
3. Finally, weight the sample x_t^i by the (non-normalized) importance factor p($y_t|x_t^i$), the likelihood of the sample x_t^i given the measurement y_t.

After the generation of N samples, the new importance factors are normalized so that they sum up to 1 (hen define a probability distribution). This process in fact implements (2.7), using an approximate sample-based representation. It should be mentioned that the idea of the particle filter based localization could be the same. However, the implementation of the idea is different case by case.

In this PAUKF approach, the particle filter is used as matching the detected features and ground truth features for estimating the location of the vehicle. There are many sources of noise. The Monte Carlo property makes particle filter can handle the different kinds of noise. The different kinds of noises exist in the map build step, sensor's detection algorithm module error, and the inherent noises of the sensors themselves. There is no way that the noise property always has a Gaussian character. Thus it is essential for the vehicle can handle the different kinds of noises.

The particle filter for localization in this thesis can be mainly divided into five parts as Figure 5 shows. The main process is particle initialization, particle prediction based on the vehicle model, receive the measurement value, transform the measurement based on the particles, and weight update & normalization and resampling of the particles.

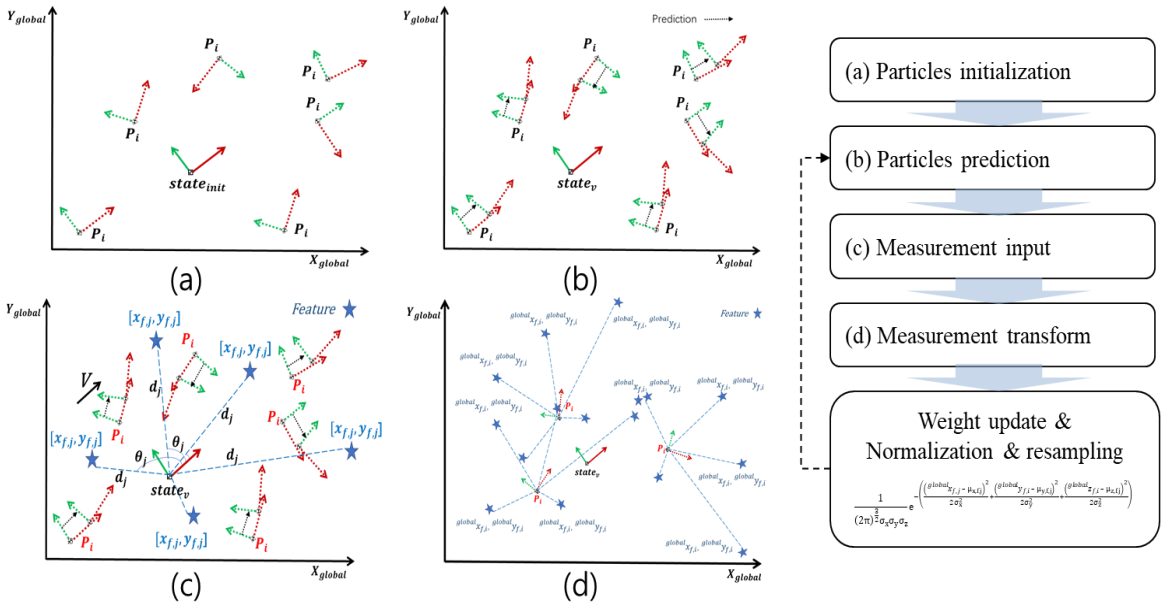


Figure 5 The process of the PF based pre-processing

The details of the particle filter are introduced in the following subsections.

2.1.1.1 Initialization

To localize the vehicle in global coordinates, it is essential to provide an initial location to the vehicle. Otherwise, the vehicle will search for its position over the entire world. Therefore, a sensor that can provide the initial global position is needed. The GPS sensor is used for particle initialization usually. Even though the GPS signal is poor due to multipath and blocking issues, it still provides a limited area for the vehicle to localize. Of course, the initial position also can be given by a human. When a particle filter receives the initial position, it generates N random particles for initialization as Equation (2.9) and Equation (2.10).

$$Particle_i = P_i = State_{init} + Noise_{init} \tag{2.9}$$

$$Particles = [particle_1, particle_2 \cdots particle_{N-1}, particle_N] \tag{2.10}$$

Figure 6 is a diagram of the particles' initialization step. The initialized particles P_i coordinate is represented as dotted lines where the actual initial position is represented as a solid line.

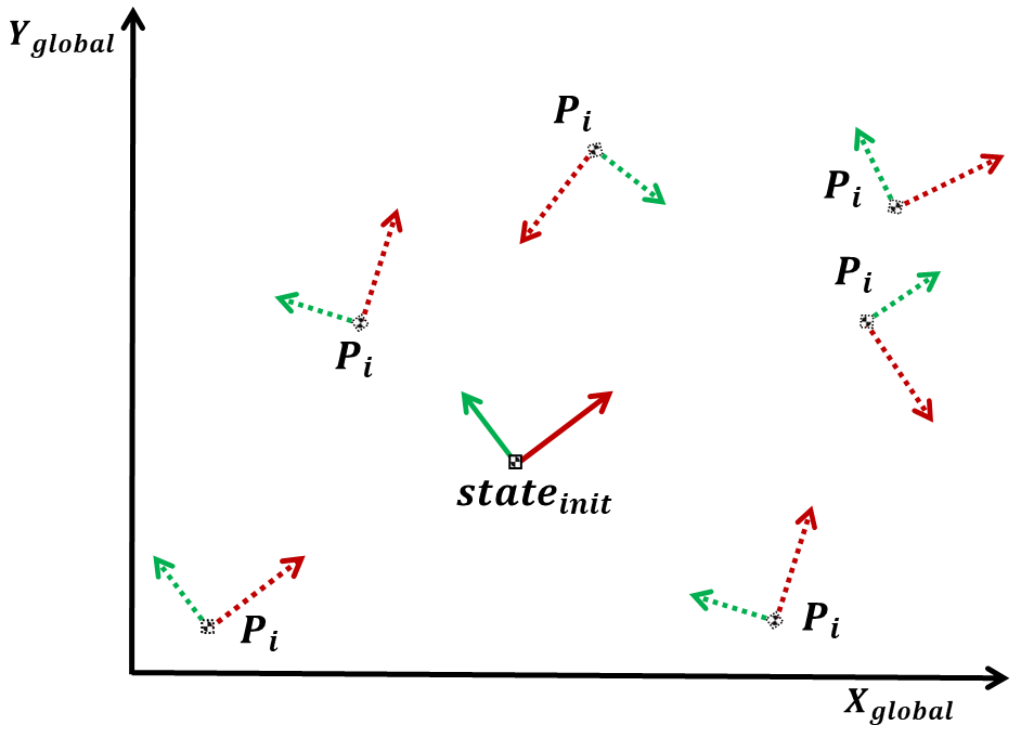


Figure 6 Particles initialization

2.1.1.2 Prediction

To obtain the prior distribution, each particle at timestamp k−1 should predict the current state based on the system prediction model. The prediction model was constructed based on the vehicle model usually. The vehicle model is different according to the structure of the vehicle. For example,

the UGV with a skid-steering wheeled structure can be modeled as Thomas Moore, Daniel Stouch research[95]. For the car-like autonomous vehicle, the model that can be selected is the kinematic model and dynamic model. It should be mentioned that whatever the model is, as long as the state of the vehicle can be calculated from time k-1 to time k, that is enough for prediction. The role of the vehicle model is to provide constraints of the prior distribution. Since the posterior distribution is obtained based on the prior distribution, it is necessary to improve the appropriate prior distribution for better estimation. The state of each particle should be predicted for obtaining the prior distributions at the next timestamp. Equation (2.11) shows the prediction step of particles. The vehicle model used in the prediction step depends on the structure of the selected autonomous vehicle. The timestamp k means current time, and the k-1 means the previous timestamp. The timestamp k-1, k, k+1 is a relative concept. Thus, in some of the literature, the timestamp k+1 and k are also used to represent the relationship of the timestamp.

$$for\ every\ particle : \hat{x}_k^- = prediction\ model\left(\hat{x}_{k-1}\right) + process\_noise \tag{2.11}$$

As Equation (2.11) shows, the prediction model is shown as a concept. This is because the prediction model is different depends on the platform that is selected. For better readability, the details of the prediction models will be explained when illustrating the specific vehicle platform. The prediction models usually contain several trigonometric functions, which correspond to a highly nonlinear prediction model. The heading angle of each particle is critical because it changes according to the local vehicle coordinates. Figure 7 is a diagram of the particle prediction step. The dotted line in black represents the predicted movement depends on Equation (2.11).

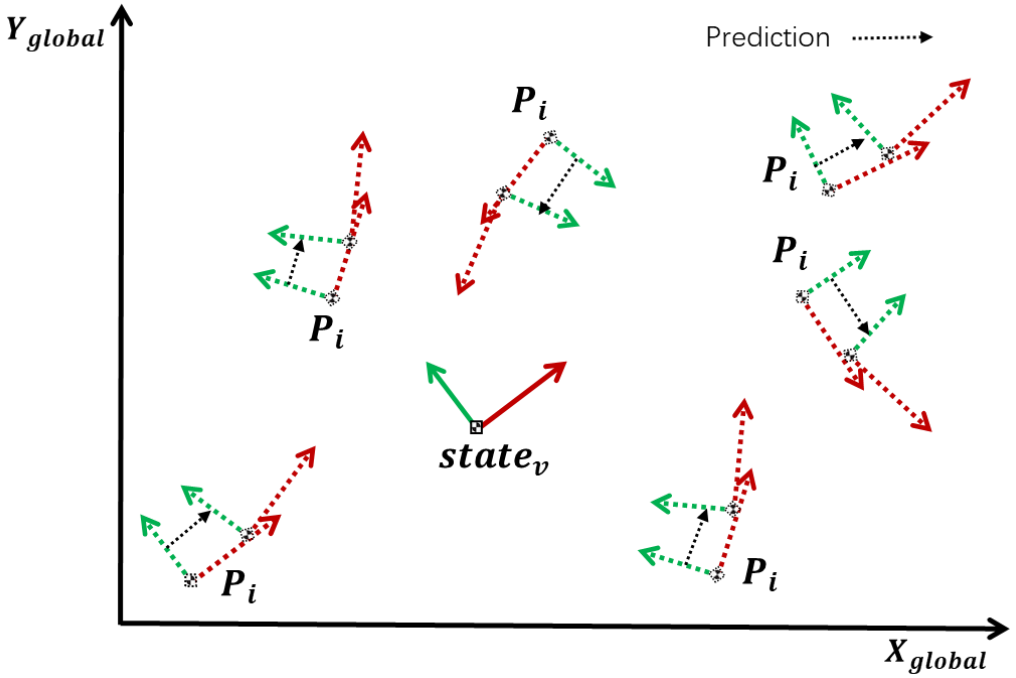


Figure 7 Particles prediction

2.1.1.3 Weight calculation

Weight is an important factor that can heavily influence the performance of the particle filter. By using the measurement data based on the vehicle sensor and the pre-saved features information, the weight of each particle is calculated. In this approach, the vehicle is assumed that can receive the features' data from the map, and the distance & bearing angle value of the vehicle, and every feature j that can be received by the on-vehicle sensor as Figure 8 shows. The black-colored parameters $\boldsymbol{\theta_j, d_j, v}$ is the measurement data from on-vehicle sensors, the red $\boldsymbol{P_i}$ means the predicted particle and the $[\boldsymbol{x_{f,j}, y_{f,j}}]$ in blue means the relative distance of features and ego vehicle.

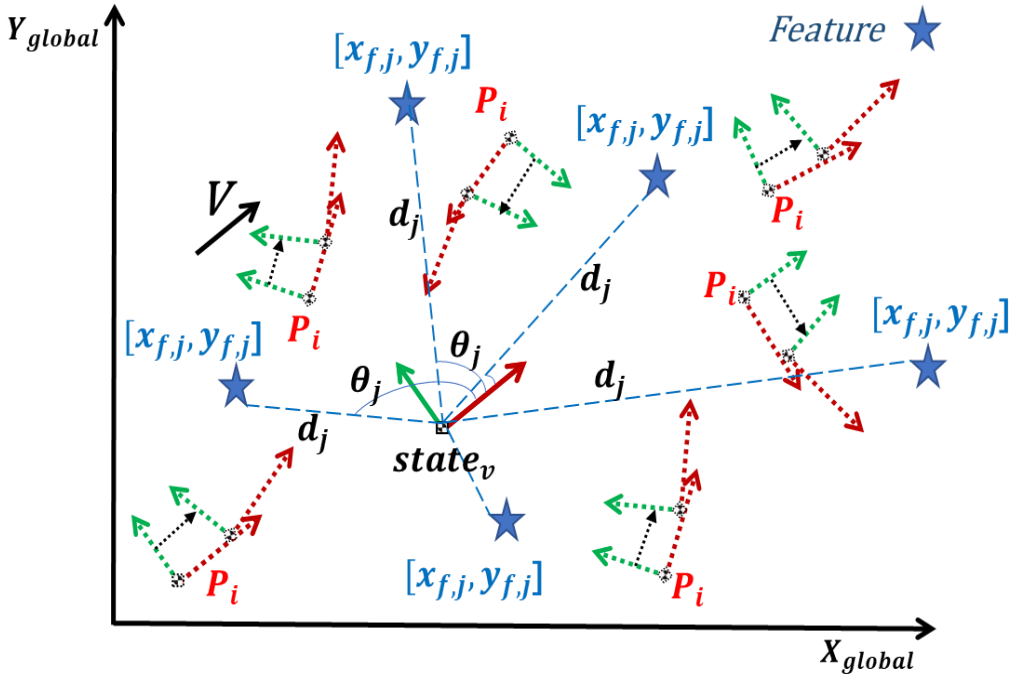


Figure 8 The vehicle, the particles, and the features

The measurement values of z are the distance from the origin of the vehicle coordinate to the feature j, and the bearing angle of the vehicle local coordinate to feature j as Equation (2.12) shows. In reality, there are different kinds of noise are exist in the sensed data. The measurement value is the same for all the particles.

$$z_k = \begin{bmatrix} \mathrm{d_j} \\ \theta_\mathrm{j} \end{bmatrix}_\mathrm{k} \tag{2.12}$$

Since the measurement is relative distance and angle of the ego vehicle and target feature, the geometry of the detection should be considered as Figure 9 shows. The geometry of the measurement is affecting the uncertainty of the particles, thus the measurement should be decomposed when executing transformation based on the predicted particles in the following contents.

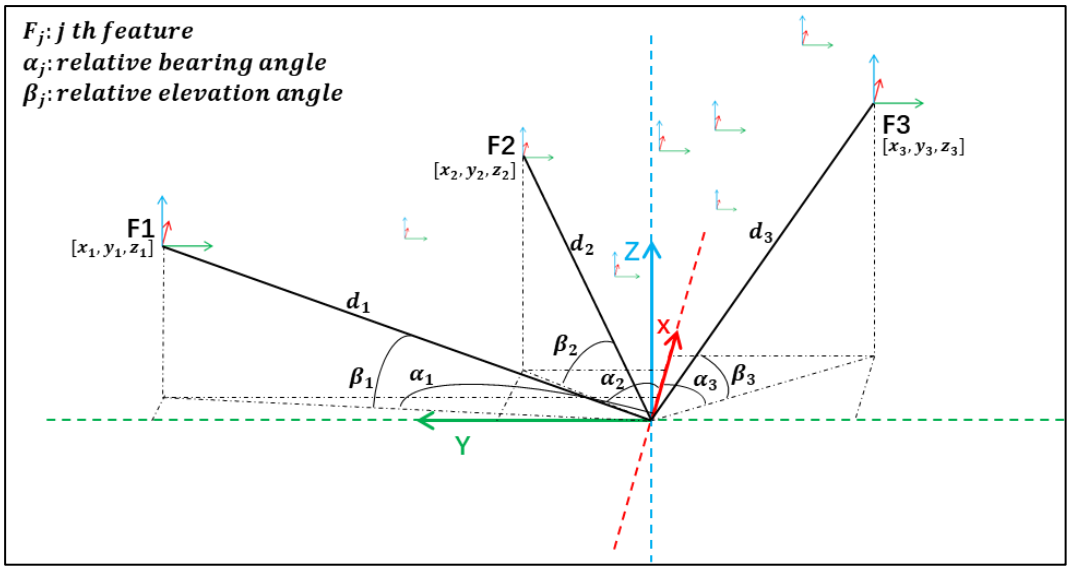


Figure 9 Measurement geometry affect uncertainty

It should be mentioned there are three coordinates used. One of the coordinates is the global coordinate which is fixed. The second coordinate is the vehicle local coordinate and the last coordinate is the sensor coordinate. Usually, the data from the sensor transforms to a fixed point of the vehicle. Therefore, this approach assumes the coordinate of the sensor and vehicle is overlapped meaningful.

Next, to matching the detected features and the features in the map, the measurement should be incorporated into the particles for evaluating each of the particles. In the previous initialization step and prediction step, the N particles are generated and predicted. Every particle has a state that contains position x, position y, and yaw angle of the vehicle in global coordinate. Since particles are generated randomly, the position x, the position y, and the yaw of particles are different. In physical meaning, the initialization step means the algorithm generates the N number of the virtual vehicle near the initial position. The prediction step means the algorithm tries to calculate the position of each particle at the next timestamp. In the current step, the x, y, and yaw values of each particle are different. It means every coordinate of the virtual vehicle is different. However, the sensed distance data and relative bearing angle data is from the vehicle coordinate. Therefore, to calculate the weight of each particle, the measurement values should be transformed into every particle's coordinate as

Figure 10 shows. The ultimate goal of the transformation is to transform the features into global coordinates based on each particle.

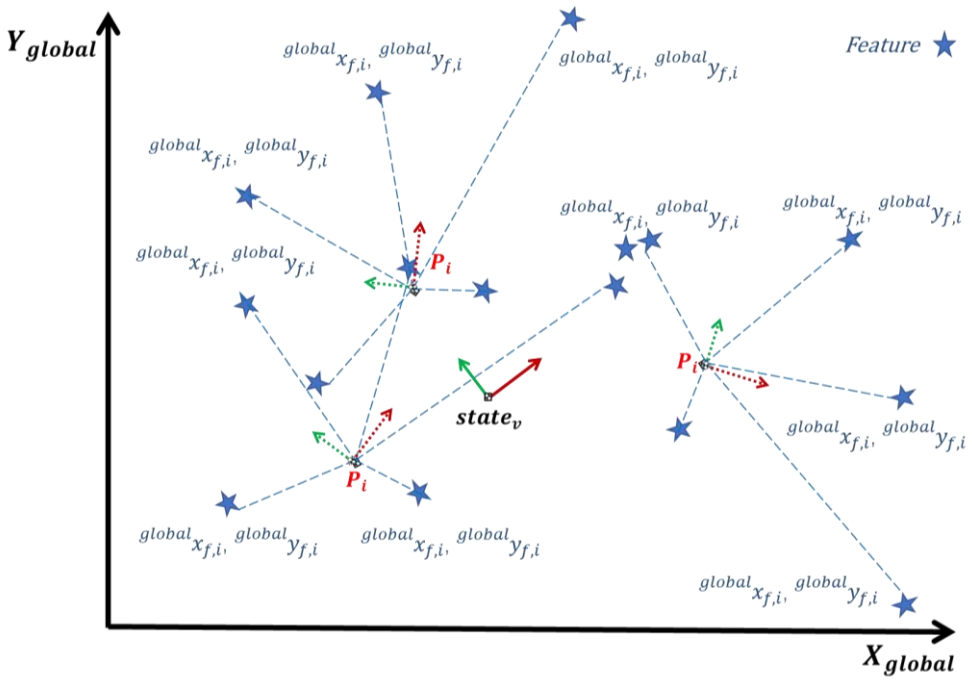


Figure 10 Transform the measurement into each particle

The translation of the measurement of the features based on each particle is implemented as Equation (2.13). The $\theta_{i,v}$ is the vehicle's yaw angle of the particles in global coordinate, x_i, y_i is the position of ith particles in the global coordinate. By transformed into Equation (2.13), $^{global}x_{f,j}$ and $^{global}y_{f,j}$ is calculated based on each of the particles.

$$for\ every\ particle: \begin{bmatrix} ^{global}x_{f,j} \\ ^{global}y_{f,j} \end{bmatrix} = \begin{bmatrix} \cos(\theta_j + \theta_{i,v}) & \sin(\theta_j + \theta_{i,v}) \\ -\sin(\theta_j + \theta_{i,v}) & -\cos(\theta_j + \theta_{i,v}) \end{bmatrix} \begin{bmatrix} d_j \cos(\theta_j) \\ d_j \sin(\theta_j) \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (2.13)$$

To evaluate the weight of each particle, a multivariable normal distribution function was used to assess the importance of each particle. Thus, a multivariable normal distribution function returns the

weight of a particle-based on the newest sensor measurement values and the predicted values from the model as Equation (2.14). The $w_{i,j}$ represent the weight of particle I with feature j, the σ_x, σ_y and σ_z represent the detection of uncertainty in x, y, and z-direction. The μ_x, μ_y, μ_z is the expected value of the features' global position which values are the ground truth data of features in the map. In physical meaning, the larger the weight is, the probability of being in the right position of the vehicle is the highest. After weighing all the particles, the weight with the largest particle's state is treated as the best estimation result of the particle filter. The illustration diagram is shown in Figure 11.

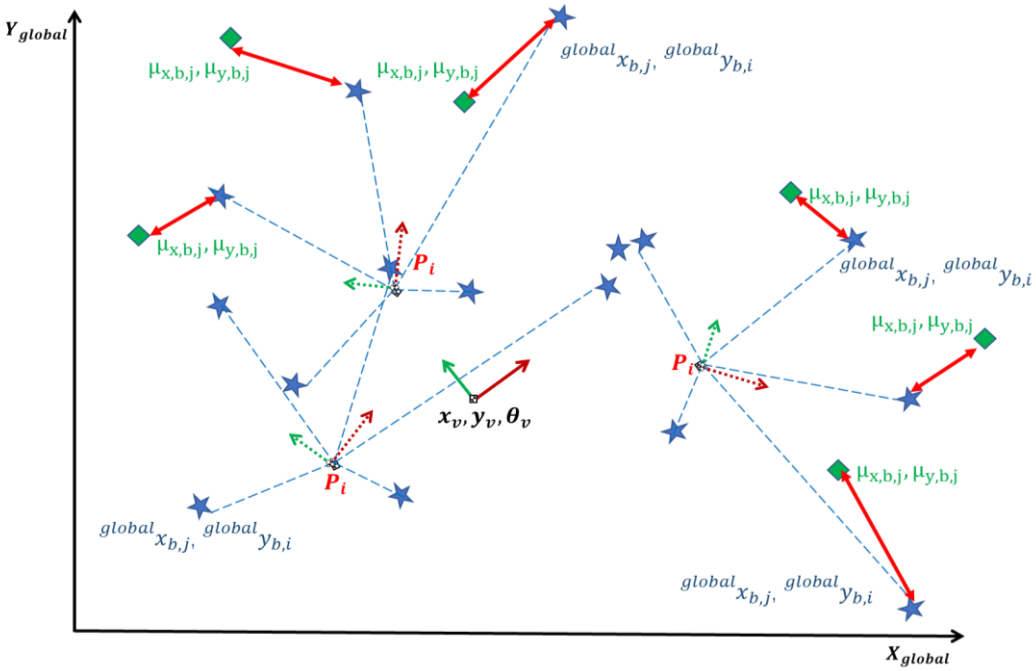


Figure 11 Weight update scheme diagram

$$w_{i,j} = \frac{1}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z} e^{-\left(\frac{\left({}^{global}x_{f,j} - \mu_{x,f,j}\right)^2}{2\sigma_x^2} + \frac{\left({}^{global}y_{f,\,i} - \mu_{y,f,j}\right)^2}{2\sigma_y^2} + \frac{\left({}^{global}z_{f,\,i} - \mu_{z,f,j}\right)^2}{2\sigma_z^2}\right)}, \; i = 1, 2 \ldots N \quad (2.14)$$

The weight w_i of the ith particle is calculated based on the product over every detected measurement and ground truth data in the map as Equation (2.15) shows.

$$w_i = \prod_{j=1}^{N_f} w_{i,j} \tag{2.15}$$

Once the weight of each particle is an update based on the map matching process, the w_i needs to be normalized as Equation (2.16) shows. The reason why the algorithm needs normalization is to prevent the particle group from polarization.

$$w_i = \frac{w_i}{\sum_{j=1}^{N_b} w_i} \tag{2.16}$$

Next, to keep the diversity of the particle group, a resampling process is needed as Equation (2.17) shows. This process draws N particles from the current particle set with probabilities proportional to their weights and replaces the current particle set with this new one. In this thesis, the resampling process happens in every iteration.

$$Prob^{i}_{prob\_selected} \propto w_i \tag{2.17}$$

Then the final output of the particle filter is the state of the particle with largest weight.

$$\hat{x}_{PF} = P_{\max weight}(x, y, \theta) \tag{2.18}$$

The whole pseudo code of the particle filter based pre-processing is shown in Table 1.

Table 1 Pseudo code of the particle filter based pre-processing

Order	Process
1	Start one sample time iteration
2	Initialization $X_{1,2\ldots N}$ particles
3	For 1 to N do
4	$\hat{x}_k^- = \text{Nonlinear prediction model}(\hat{x}_{k-1}) + \text{noise}$
5	End for
6	$z_k = \text{measurement input}$
7	$w_{[1,2\ldots N]} = \text{multivariable normal distribution}(\hat{X}_{i\ldots N}^-, z_k, \sigma_{x,y,z}, \mu_{x,y,z})$
8	Return $\hat{x}_{PF} = \underset{w_{[1,2\ldots N]}}{\text{argmax}}\, P(x_{p,i}, y_{p,i}, z_{p,i}\theta_{p,i})$
9	End one sample time iteration

Based on the multivariable distribution framework, if there are different sources of perception data, they can be fused simply. When the vehicle receives multiple perception data about one feature, the belief of each perception data can be calculated based on the multivariable normal distribution Equation, as Equation (2.14) shows. After calculating the probability based on each sensor, the final weight of each particle can be calculated, as Equation (2.19) shows. This provides a feasible multisource data fusion method based on the PAUKF. Please note that the feasibility is not verified. In this thesis, we only considered a single range sensor case.

$$w_i = P_{LiDAR}(x, y, z) * P_{RADAR}(x, y, z) * P_{Camera}(x, y, z) * \ldots P_{\text{perception source}}(x, y, z) \quad (2.19)$$

2.1.2 Particle-Aided Unscented Kalman Filter Algorithm

The particle filter based pre-processing algorithm is introduced in Section 2.1.1. The particle estimates the position of the vehicle by using the range sensor. The final results for each particle contain information about the surrounding features and the position of the ego vehicle. Therefore, it can be concluded that this estimation from the particle sensor becomes a virtual sensor that provides accurate location results. When the UKF estimates the state, the results from the particle filter will be

the measurement value of the vehicle. Subsequently, the PAUKF can extract a more precise result based on the particle filter estimation results. A flowchart of the PAUKF is shown in Figure 12.

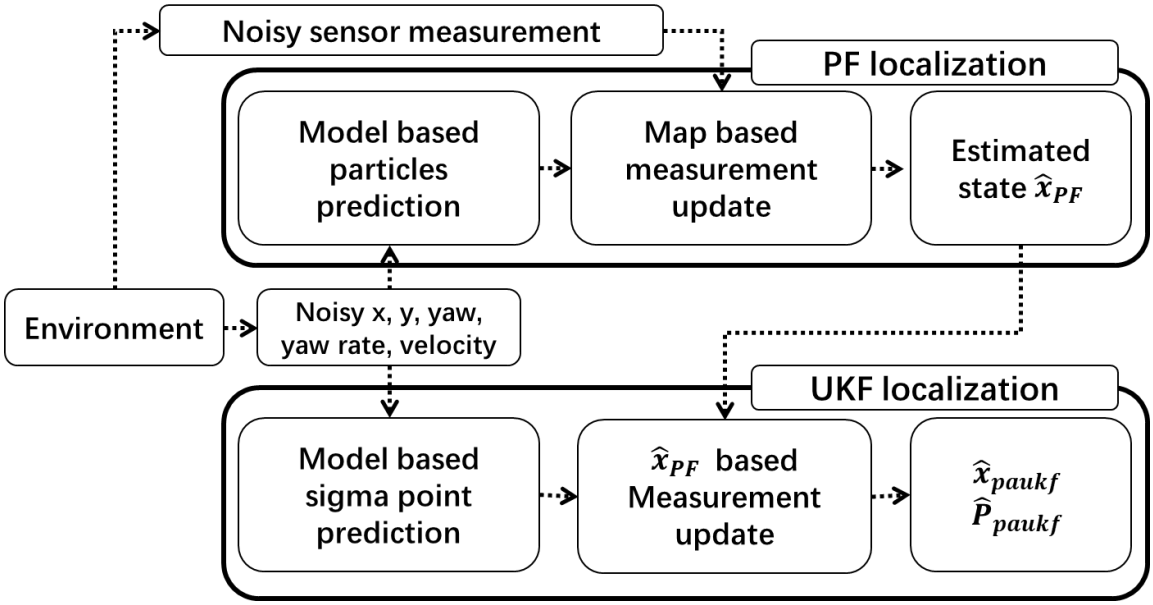


Figure 12 Particle-aided unscented Kalman filter algorithm flowchart

Before the PAUKF, the basic EKF framework of the state estimation process should be illustrated. The basic framework for the EKF involves the estimation of the state of a discrete-time nonlinear dynamic system.

The x_k represent the unobserved state of the system and z_k is the only observed signal. The process noise v_k drives the dynamic system, and the observation noise is given by n_k. The system dynamic model F and H are derived based on the target hardware platform and the sensors that be selected. Consider the basic state-space estimation framework as Equations (2.20) and (2.21).

$$X_k = F(x_{k-1}, v_k) \tag{2.20}$$

$$z_k = H(x_k, n_k) \tag{2.21}$$

Given the noisy observation z_k, a recursive estimation for x_k can be expressed in the Equation (2.22).

$$\hat{x}_k = (prediction\ of\ x_k) + K_k[z_k - (prediction\ of\ z_k] \tag{2.22}$$

This recursion provides the optimal minimum mean-squared error estimate for x_k assuming the prior estimate $\hat{x}_{k-1}$ and current observation z_k are Gaussian random variables. For nonlinear models, the EKF approximates the optimal terms. In other words, in the EKF the state distribution is approximated by a Gaussian random variable which is then propagated analytically through the "first-order" linearization of the nonlinear system. These approximations, however, can introduce large errors in the true posterior mean and covariance of the transformed random variable, which may lead to sub-optimal performance and sometimes divergence of the filter.

The UKF provides another solution to approximation issues of the EKF. The state distribution is approximate by using a minimal set of carefully chosen sample points which is called sigma point. These sample points completely capture the true mean and covariance of the Gaussian random noise, and when propagated through the true non-linear system, captures the posterior mean and covariance accurately to the 3rd order(Taylor series expansion) for any nonlinearity. Therefore, the combination of the PF and UKF is a good combination for processing different kinds of noises.

As illustrated above, PAUKF is Instead of using a Taylor expansion, a UKF deterministically extracts the mean and covariance using the sigma points as Figure 13 shows. The sigma points are generated near the state which is also the mean of the random variables. The PAUKF also follows the 3 basic processes of the Bayes filter, the initial belief Bel(x_0), the next state probability p($x_t|x_{t-1}, u_{t-1}$) which usually called the motion model, and the perceptual likelihood p($y_t|x_t$) which is usually called a perceptual or measurement model. At the final measurement update step, the estimation results z_k from the particle filter are used for the measurement update as Equation (2.20).

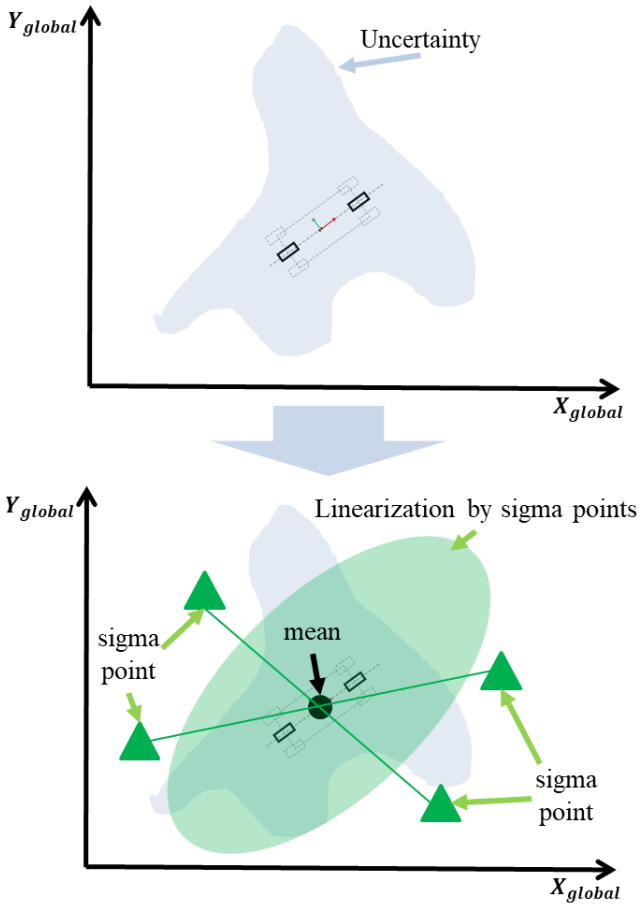


Figure 13 Linearization of the Non-Gaussian noise with sigma points

The sigma points are predefined by an empirical parameter λ that is calculated using Equation (2.23). The diagram of the sigma points generation is shown in Figure 14. The physical meaning of the λ is the spread area of the sigma point near the mean value. The sigma point is an asymmetrical region around the mean value. $P_{k|k}$ is the covariance matrix of the state, which updates at every iteration. The state vector of the vehicle is x_{paukf}, which is changed depends on the platform that we selected as shown in Equation (2.24). The state of the UGV and the car-like autonomous vehicle is different, therefore, it details in the evaluation step with a specific vehicle platform in the following chapters. The state vector value is the mean of the sigma matrix. n_x is the quantity of the state vector. Then, the sigma points are generated using Equation (2.25).

$$\lambda = 3 - n_x \tag{2.23}$$

$$x_{paukf} = State\ of\ the\ target\ platform \quad (2.24)$$

$$\boldsymbol{X}_{paukf,k-1} = (x_{paukf,k-1},\ x_{paukf,k-1} + \sqrt{(\lambda + n_x)P_{k-1}},\ x_{paukf,k-1} - \sqrt{(\lambda + n_x)P_{k-1}}) \quad (2.25)$$

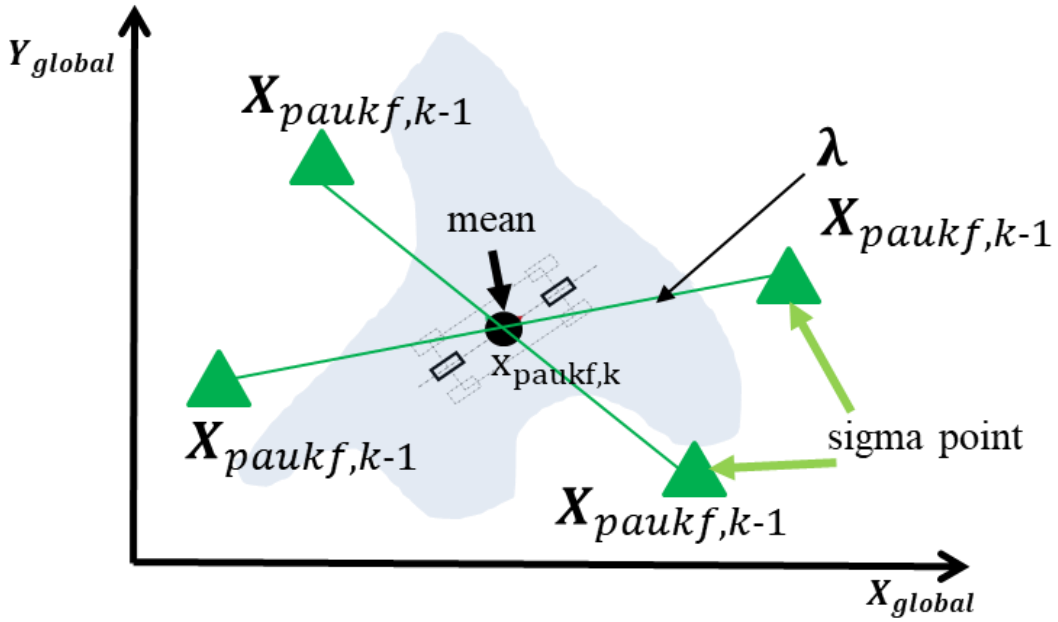


Figure 14 The generated sigma points

The weight of each sigma point is can be calculated based on the Equation (2.26) and (2.27). The weight of the sigma point represents importance. The diagram of the weights of the sigma points is shown in Figure 15.

$$w_{paukf,i} = \frac{\lambda}{\lambda + n_x}, \text{when } i = 0 \quad (2.26)$$

$$\boldsymbol{W}_{paukf,i} = \frac{1}{2(\lambda + n_x)}, \text{when } i = 1 \ldots n_x \quad (2.27)$$

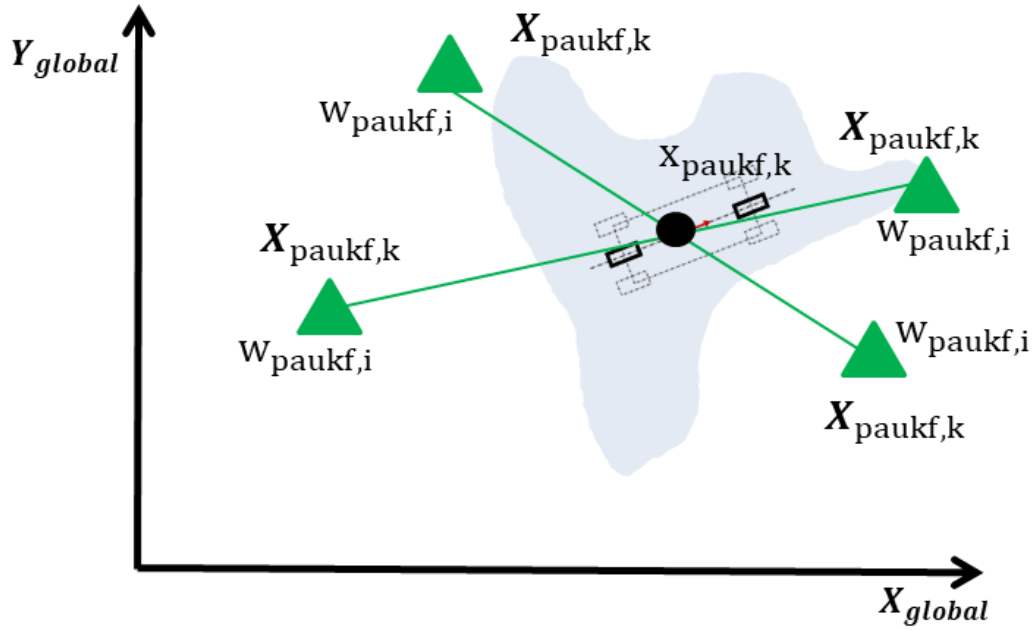


Figure 15 Calculate the weights of sigma points

After it generates the sigma points, a UKF needs a prediction model to calculate the position of the sigma points at the next timestamp as Equation (2.28).

$$\hat{\boldsymbol{X}}^{-}_{paukf,k} = Prediction\ model(\boldsymbol{X}_{paukf,k-1}) \tag{2.28}$$

The diagram of the prediction step of the sigma points is shown in Figure 16.

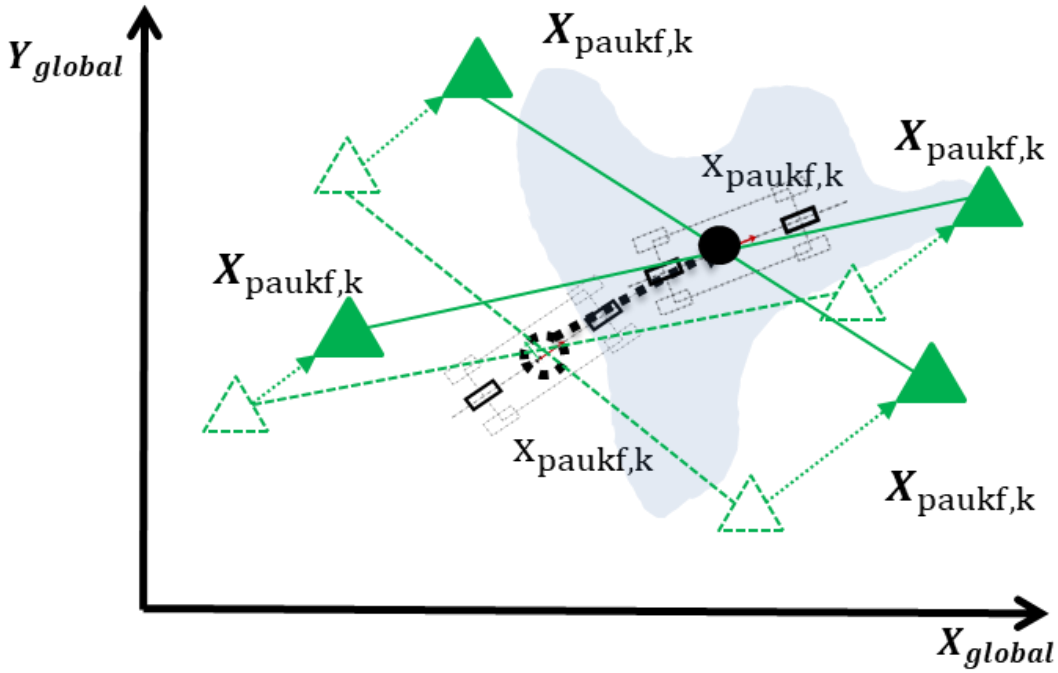


Figure 16 Predict the sigma points

Then the predicted state and covariance matrix of the vehicle can be calculated by using the generated weights and sigma points as Equation (2.29) and (2.30) show.

$$\hat{x}^{-}_{paukf,k} = \sum_{i=0}^{2n_x} \mathbf{W}_{paukf,i} \mathbf{X}_{paukf,k|k-1,i} \tag{2.29}$$

$$P^{-}_{paukf,k} = \sum_{i=0}^{2n_x} \mathbf{W}_{paukf,i} (\mathbf{X}_{paukf,k|k-1,i} - \hat{x}^{-}_{paukf,k}) (\mathbf{X}_{paukf,k|k-1,i} - \hat{x}^{-}_{paukf,k})^{T} \tag{2.30}$$

Figure 17 shows the diagram of the predicted state based on the sigma points.

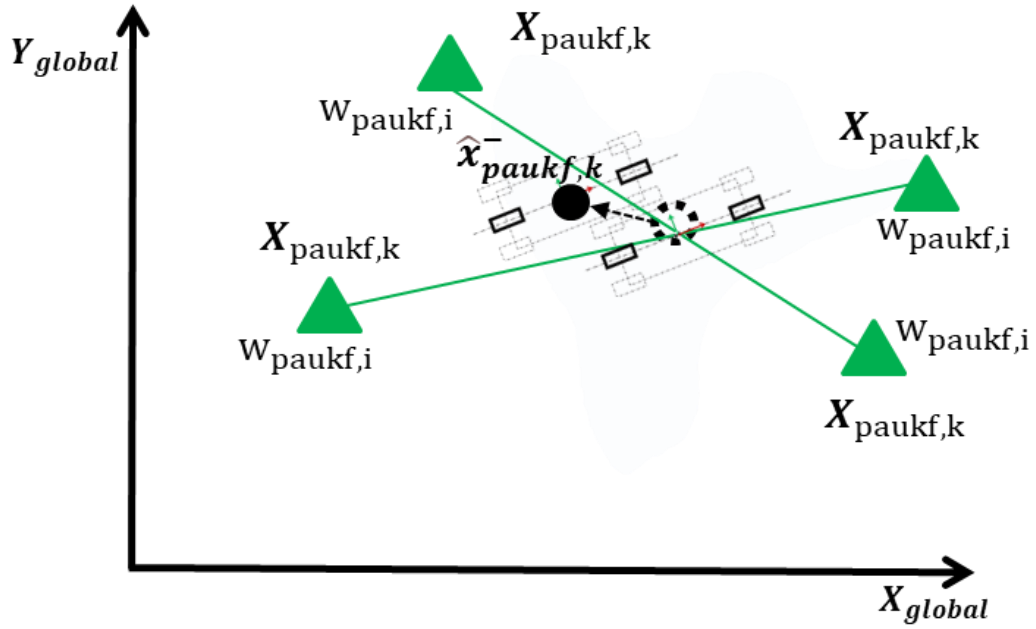


Figure 17 Predicted state based on the sigma points

After predicting the new mean and covariance matrix based on the sigma point, the group of the sigma points represents the prior distribution. Until now, the prior probability was calculated based on sigma points. When using a Bayesian filter, a measurement update can be implemented. Instead of using the original range sensor with noise from the vehicle, $\hat{x}_{PF}$ is used as the measurement in this step. This means that $\hat{x}_{PF}$ becomes a virtual sensor, which is more precise than the original sensor. Since $\hat{x}_{PF}$ already includes sensor information based on the particle filter, it optimally provides a more precise belief of the state. The measurement vector depends on the sensors that be selected. The measurement estimation from the particle filter provides position x,y, and the yaw angle of the vehicle. Thus the measurement is shown in Equation (2.31).

$$z = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.31}$$

The measurement model based on the sigma points can be calculated as Equations (2.32) shows. The particle filter measurement provides estimated x, y, and yaw data to the UKF. Since the x, y, and

yaw angle can be measured directly, the value in the matrix A becomes 1 at the corresponding position to the selected state. Figure 18 shows the diagram of the measurement prediction based on the sigma points.

$$\mathbf{Z}_{\text{paukf,k|k-1}} = \text{A}\mathbf{X}_{\text{paukf,k|k-1}} \tag{2.32}$$

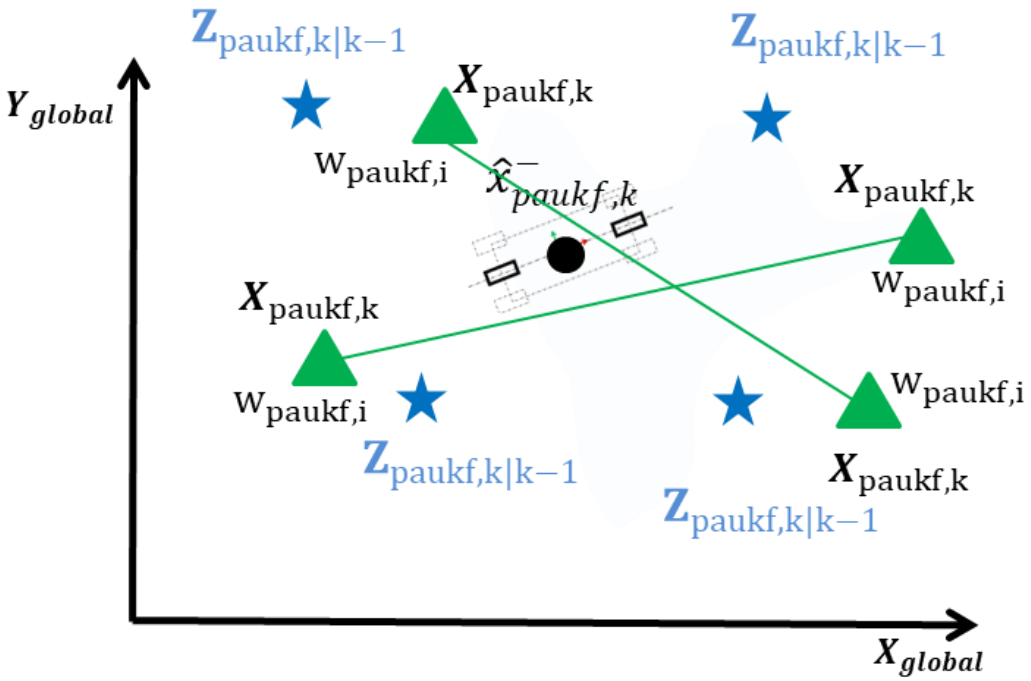


Figure 18 The measurement prediction based on sigma points

The predicted measurement means is calculated based on the weight of each measurement's sigma points, as shown in Equation (2.33).

$$\hat{z}^{-}_{paukf,k} = \sum_{\text{i=1}}^{\text{n}_\text{x}} \text{W}_{\text{paukf,i}} \mathbf{Z}_{\text{paukf,k|k-1,i}} \tag{2.33}$$

The predicted measurement covariance is calculated using Equation (2.34). R is the measurement noise covariance, as shown in Equation (2.35). The covariance is tuned according to the particle filter estimation results.

$$S_{\hat{z}_k\hat{z}_k} = \sum_{i=0}^{2n_x} \mathbf{W}_{paukf,i} (\mathbf{Z}_{paukf,k|k-1,i} - \hat{z}^-_{paukf,k})(\mathbf{Z}_{paukf,k|k-1,i} - \hat{z}^-_{paukf,k})^T + R \quad (2.34)$$

$$R = \begin{bmatrix} {\sigma_{x_{PF}}}^2 & 0 & 0 \\ 0 & {\sigma_{y_{PF}}}^2 & 0 \\ 0 & 0 & {\sigma_{\theta_{PF}}}^2 \end{bmatrix} \quad (2.35)$$

At this time, a measurement value is needed to calculate the posterior probability. The update step is similar to that of the Kalman filter. The only difference is that the UKF needs to calculate the cross-correlation value, according to Equation (2.36), between the sigma points in the state space and the measurement space.

$$T_{\hat{x}_k\hat{z}_k} = \sum_{i=0}^{2n_x} \mathbf{W}_{paukf,i} (\mathbf{X}_{paukf,k|k-1,i} - \hat{x}^-_{paukf,k})(\mathbf{Z}_{paukf,k|k-1,i} - \hat{z}^-_{paukf,k})^T \quad (2.36)$$

Based on the cross-correlation matrix and the measurement covariance, the Kalman gain is then calculated as Equation (2.37).

$$K = T_{\hat{x}_k\hat{z}_k} S^{-1}_{\hat{z}_k\hat{z}_k} \quad (2.37)$$

The state is updated using the measurement value $\hat{x}_{PF}$, which is obtained from the particle filter estimation as Equation (2.38). The measurement value is from the estimation result of the particle filter, thus the z_k of the PAUKF can be represented as the Equation (2.39) shown.

$$z_{paukf,k} = \hat{x}_{PF} \quad (2.38)$$

$$\hat{x}_{paukf,k} = \hat{x}^-_{paukf,k} + K_{k+1|k}(z_{paukf,k} - \hat{z}^-_{paukf,k}) \quad (2.39)$$

Figure 19 shows the state update based on the measurement from the particle filter. This is the place where the fusion of PF and UKF.

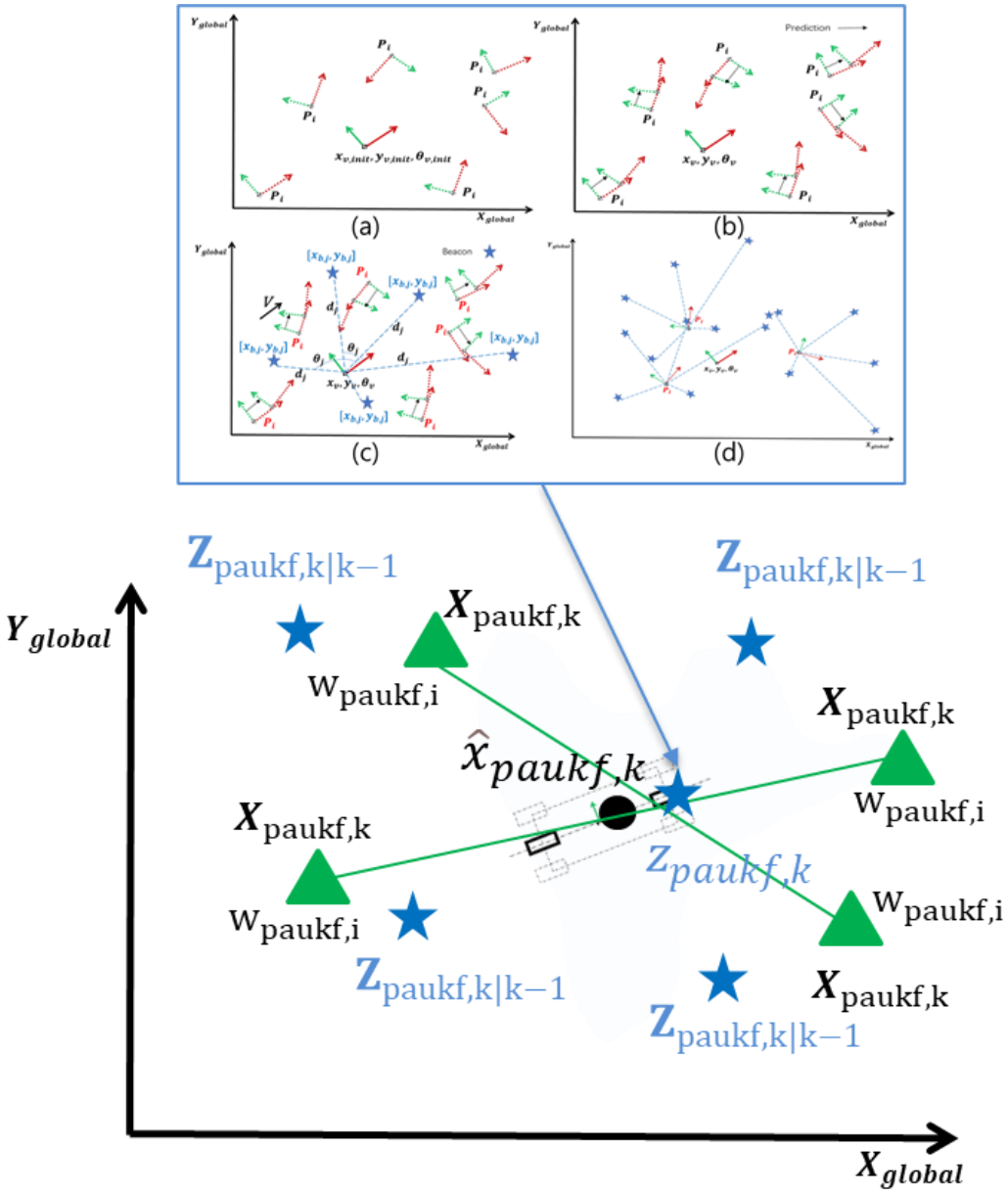


Figure 19 The state update based on the measurement from the particle filter

The covariance matrix is then updated based on the updated Kalman gain and the measurement covariance matrix as Equation (2.40).

$$P_{paukf,k} = P^{-}_{paukf,k} - KS_{\hat{z}_k\hat{z}_k}K^T \tag{2.40}$$

The terms $\hat{x}_{paukf,k}$ and $P_{paukf,k}$ are the final estimation results of the PAUKF, which combines the vehicle model, particles estimation, position of features, the ground truth of position data in the map, and unscented Kalman filter-based estimation. The different kinds of vehicle models can be added to the prediction model. However, the research about the detail of the vehicle dynamic is not the main interest of this thesis.

For easier understanding, the complete pseudo-code of the PAUKF algorithm is shown in Table 2.

Table 2 Pseudocode of the particle-aided unscented Kalman filter (PAUKF)

Order	Process
1	Start one sample time iteration
2	Initialization $X_{1,2\ldots N}$ particles
3	For 1 to N do
4	$\overline{X}_{k+1} =$ prediction model
5	End for
6	$\hat{x}_{PF} = P_{max\,weight}(x, y, \theta)$
7	For 1 to n_{aug} do
8	$\mathbf{X}_{paukf,k-1}$ = unscented transform $(\lambda, x_{paukf,k-1}, \sigma)$
9	$\hat{\mathbf{X}}^{-}_{paukf,k}$ = Model-based prediction
10	$\mathbf{Z}_{paukf,k\|k-1}$ = A($\hat{\mathbf{X}}^{-}_{paukf,k}$) for measurement transition
11	$\hat{x}_{paukf,k}, P_{paukf,k} =$ update($T_{\hat{x}_k\hat{z}_k}, S_{\hat{z}_k\hat{z}_k}, \hat{z}^{-}_{paukf,k}, \hat{x}^{-}_{paukf,k}, z_{paukf,k}, R$)
12	End one sample time iteration

The total formula based PAUKF iteration diagram is organized as Figure 20.

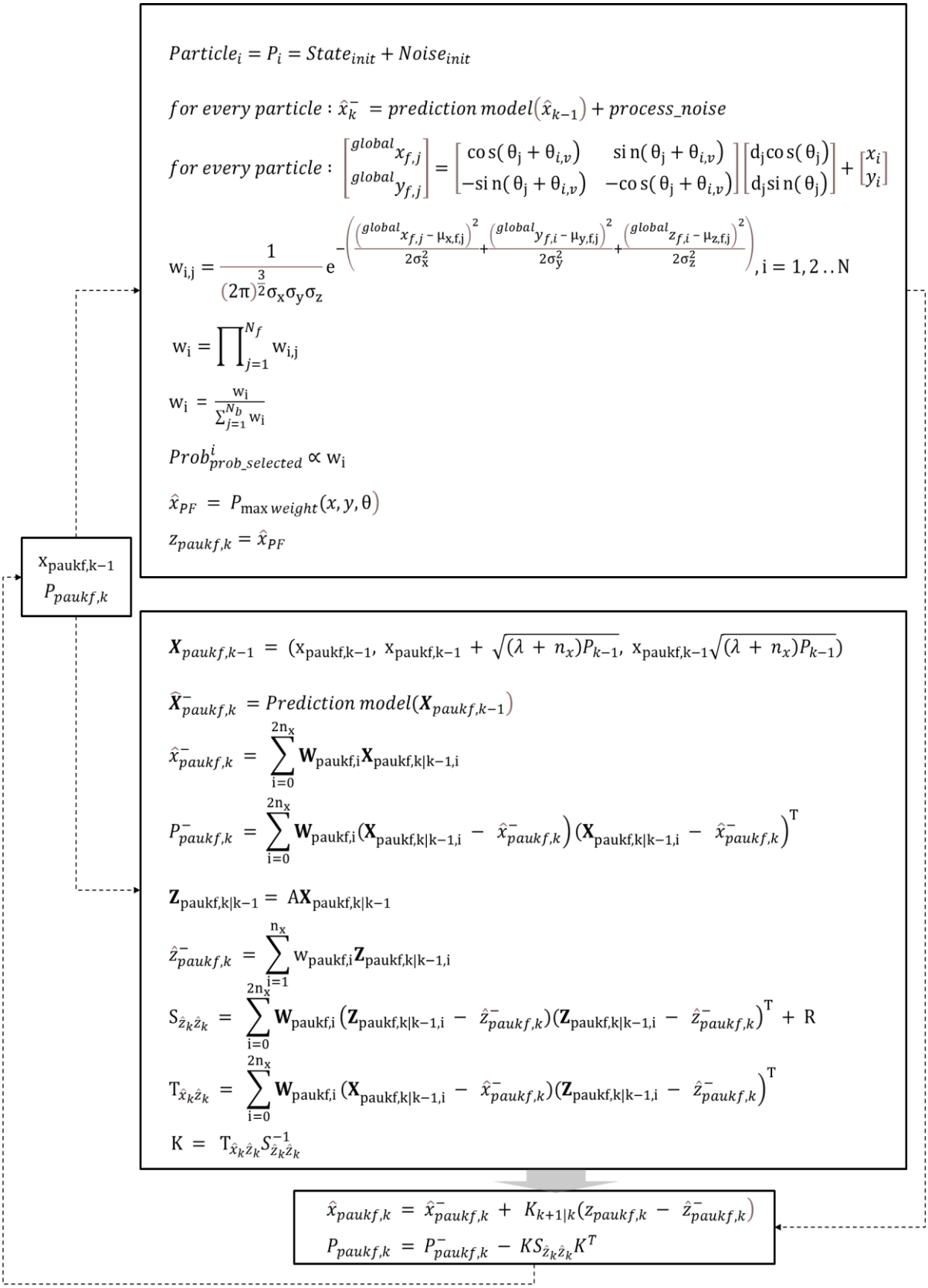


Figure 20 The PAUKF algorithm flow

Chapter 3 PAUKF experiment in the UGV platform

In the previous section, the concept of PAUKF is illustrated in detail. The PAUKF uses the particle filter for processing features and features information by using the vehicle model, and the UKF is in charge of filtering the state based on the data from the vehicle itself based on the prediction model. The evaluation of the PAUKF algorithm is carried out based on the experiment. To evaluate the performance of the PAUKF algorithm, it is needed to be verified in the real vehicle, real hardware, and real perception environment. With the real vehicle test, the performance of the PAUKF could be verified appropriately.

Since the PAUKF is matching the perceived features data and the features ground truth data in the map, the precision of the ground truth data and the quantity and precision of the detected features are critical for the performance of the PAUKF. The ultimate goal of the PAUKF experiment is to figure out the location estimation performance and the correction behavior based on the limited features.

As illustrated in chapter 2, the prediction model of the PAUKF should be changed according to the specific hardware platform. The UGV used in this experiment is a skid-steered mobile robot, thus the prediction model of the PAUKF should be modified to adapt to the real vehicle.

The ideal real test environment is receiving the data from CAN-bus of the vehicle, detecting the features from the surrounded environment, getting the ground truth data of features from HD-map from the commercial map provider and the PAUKF algorithm is loaded to the vehicle computing unit and execute the algorithm in the vehicle computing unit.

However, because of the limited test environment and perception algorithm, a differential wheeled unmanned ground vehicle(UGV) called AECOBOT is used to verify the PAUKF. The UGV equips one lidar sensor(model name: Velodyne VLP-16) for making a simplified HD-map, one depth camera(Intel RealsenseD435) for detecting the features around the UGV. The total algorithm implementation is based on the robot operating system(ROS).

Since the UGV under test is different from the vehicle that is illustrated in the previous sections, the PAUKF algorithm should be modified to adapt to the UGV. The change of the vehicle leads to the

change of the prediction model. The prediction model of particle filter and unscented Kalman filter is referred to as the research of T. Moore and D. Stouch, and other researches[96]–[98]. The coordinate definition follows the ROS standard REP-103 and REP-105. The state of the UGV is defined as Equation (2.41) based on the coordinate shown in Figure 21.

$$X_{UGV} = [x \quad y \quad z \quad roll \quad pitch \quad yaw \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{roll} \quad \dot{pitch} \quad \dot{yaw} \quad \ddot{x} \quad \ddot{y} \quad \ddot{z}] \quad (2.41)$$

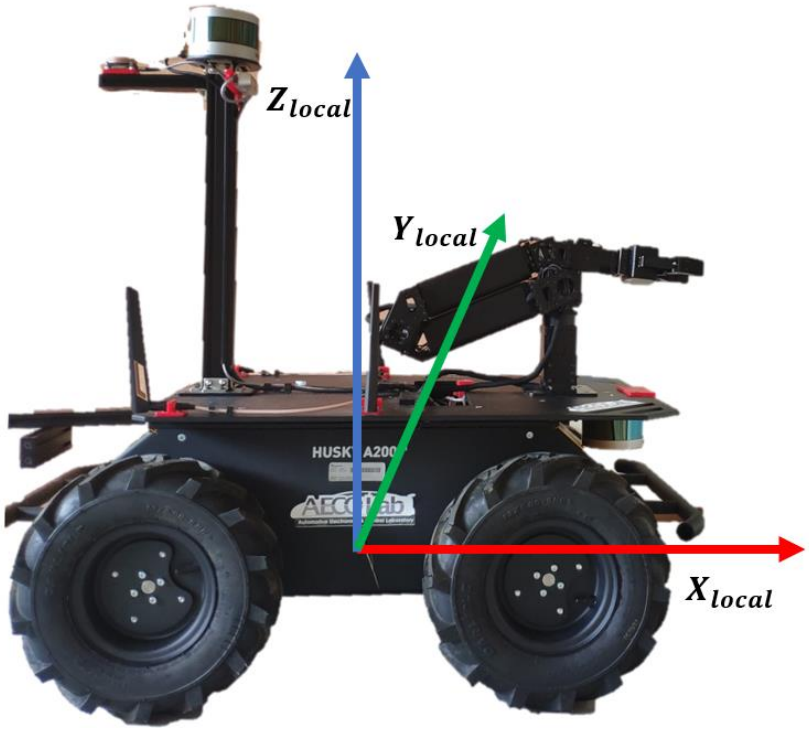


Figure 21 The state and coordinate of the UGV

The prediction model is represented in the matrix form. Despite the environment of the experiment is planar, to improve the expansibility of the algorithm, 15 dimensions of the state and the prediction model are shown below. The transfer function is initialized with the identity matrix which size of column and row is 15 as Equation (2.42) shows.

$$Transferfunction = identity\_matrix(15,15) \quad (2.42)$$

Then the prediction model of each state is defined as Equation (2.43-2.73) shows. The dt is the sample time of the ROS.

$$sr, cr \ : \sin(roll), \cos(roll) \tag{2.43}$$

$$sp, cp: \sin(\text{pitch}), \cos(\text{pitch}) \tag{2.44}$$

$$sy, cy : \sin(yaw), \cos(yaw) \tag{2.45}$$

$$Transferfunction(0,6) = cy * cp * dt \tag{2.46}$$

$$Transferfunction(0,7) = (cy * sp * sr - sy * cr) * dt \tag{2.47}$$

$$Transferfunction(0,8) = (cy * sp * cr - sy * sr) * dt \tag{2.48}$$

$$Transferfunction(0,12) = \frac{1}{2} * Transferfunction(0,6) * dt \tag{2.49}$$

$$Transferfunction(0,13) = \frac{1}{2} * Transferfunction(0,7) * dt \tag{2.50}$$

$$Transferfunction(0,14) = \frac{1}{2} * Transferfunction(0,8) * dt \tag{2.51}$$

$$Transferfunction(1,6) = sy * cp * dt \tag{2.52}$$

$$Transferfunction(1,7) = (sy * sp * sr + cy * cr) * dt \tag{2.53}$$

$$Transferfunction(1,8) = (sy * sp * cr - cy * sr) * dt \quad (2.54)$$

$$Transferfunction(1,12) = \frac{1}{2} * Transferfunction(1,6) * dt \quad (2.55)$$

$$Transferfunction(1,13) = \frac{1}{2} * Transferfunction(1,7) * dt \quad (2.56)$$

$$Transferfunction(1,14) = \frac{1}{2} * Transferfunction(1,8) * dt \quad (2.57)$$

$$Transferfunction(2,6) = -sp * dt \quad (2.58)$$

$$Transferfunction(2,7) = cp * sr * dt \quad (2.59)$$

$$Transferfunction(2,8) = cp * cr * dt \quad (2.60)$$

$$Transferfunction(2,12) = \frac{1}{2} * Transferfunction(2,6) * dt \quad (2.61)$$

$$Transferfunction(2,13) = \frac{1}{2} * Transferfunction(2,7) * dt \quad (2.62)$$

$$Transferfunction(2,14) = \frac{1}{2} * Transferfunction(2,8) * dt \quad (2.63)$$

$$Transferfunction(3,9) = dt \quad (2.64)$$

$$Transferfunction(3,10) = sr * tp * dt \quad (2.65)$$

$$Transferfunction(3,11) = cr * tp * dt \tag{2.66}$$

$$Transferfunction(4,10) = ct * dt \tag{2.67}$$

$$Transferfunction(4,11) = -sr * dt \tag{2.68}$$

$$Transferfunction(5,10) = sr * \frac{1}{cp} * dt \tag{2.69}$$

$$Transferfunction(5,11) = cr * \frac{1}{cp} * dt \tag{2.70}$$

$$Transferfunction(6,12) = dt \tag{2.71}$$

$$Transferfunction(7,13) = dt \tag{2.72}$$

$$Transferfunction(8,14) = dt \tag{2.73}$$

Then predicted state can be calculated as Equation (2.74) both in particle filter and UKF.

$$X_{UGV,k+1} = Transferfunction * X_{UGV,k} + Noise_{UGV} \tag{2.74}$$

The measurement is based on the open-source object detection algorithm ar_track_alvar based on ROS. By combining the RealsenseD435 depth camera and ar_track_alvar, the 3-dimensional features with position value stamped can be obtained. The RealsenseD435 provides the relative distance of the sensor coordinate and detected object in x, y, z-direction, the measurement model becomes The UGV provides raw odometry data which including position, velocity, and angular velocity in x, y, z

three axes with covariance stamped. The PAUKF receives the data from raw odometry and Realsense D435 depth camera and localizes the UGV in the “odom” coordinate which is defined in the REP-105.

3.1 Experiment environment

As illustrated in section 2.4, the total evaluation of the PAUKF is based on the robot operating system(ROS). The ROS is robotics middleware. Although ROS is not an operating system like Windows from Microsoft, it provides services from low-level control, device communication, package management, and convenient packages distribution. The ROS is designed to promote code sharing and enable the development of open-source robotics commons. Therefore, many open-sourced packages were developed and shared based on ROS. Because ROS needs a lot of open-source libraries, the operation system of the ROS is based on the Linux framework. There are a lot of packages that are developed and being updated based on ROS[99]–[101]. Developers can use open packages to realize the specialized functionality of the robot.

Some many robots and sensors support ROS. One of the most famous models is the husky UGV made by Clearpath. The husky supports the ROS officially. Thus all the data from husky can be obtained based on the ROS. It means, the developed code can access most of the data and modify them. There are no sensors mounted on the original husky. Thus, several modifications are needed to meet the evaluation request. The ultimate coordinate relationship and diagram of the PAUKF experiment are shown in Figure 22.

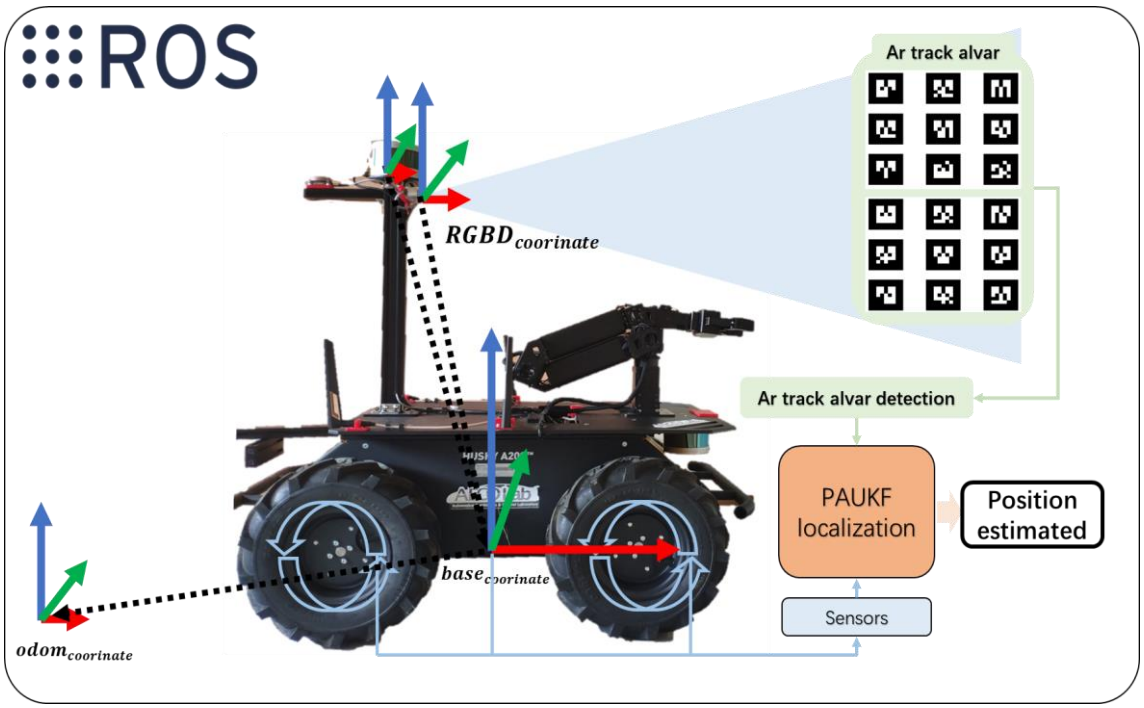


Figure 22 Evaluation environment configuration of PAUKF

To better readability, several coordinates should be introduced first.

3.2 Coordinate of devices

Before introducing the detail of the test environment configuration, the REP-105 should be reviewed since this is the standard coordinate that is defined in the ROS. In this evaluation process, the “odom” frame, “base_link” and “map” frames are used.

3.2.1 Odom

The “odom” frame means the odometry frame. The coordinate frame called Odom is world-fixed. The pose of a mobile platform in the Odom frame can drift over time, without any bounds. This drift makes the Odom frame useless as a long-term global reference. However, the pose of a robot in the Odom frame is guaranteed to be continuous, meaning that the pose of a mobile platform in the Odom frame always evolves smoothly, without discrete jumps. In a typical setup, the Odom frame is computed based on an odometry source, such as wheel odometry, visual odometry, or an inertial

measurement unit. The Odom frame is useful as an accurate, short-term local reference, but drift makes it a poor frame for long-term reference[102].

The Odom frame is a “relative” frame in which the translation of the coordinate is calculated based on the UGV’s state. The state of UGV accumulated and transformed into the “odom” coordinate.

3.2.2 base_link

The coordinate frame called base_link is rigidly attached to the mobile robot base. The base_link can be attached to the base in any arbitrary position or orientation. For every hardware platform, there will be a different place on the base that provides an obvious point of reference. The base_link of the UGV is defined by the manufacture which is min the middle of the UGV. The RGB-D camera and Lidar sensor should transform the data from the local coordinate to the base_link coordinate.

3.2.3 map

The coordinate frame called map is a world fixed frame, with its Z-axis pointing upwards. The pose of a mobile platform, relative to the map frame, should not significantly drift over time. The map frame is not continuous, meaning the pose of a mobile platform in the map frame can change in discrete jumps at any time. In a typical setup, a localization component constantly re-computes the robot pose in the map frame based on sensor observations, therefore eliminating drift, but causing discrete jumps when new sensor information arrives. The map frame is useful as a long-term global reference, but discrete jumps in position estimators make it a poor reference frame for local sensing and acting.

The map frame is used to extract the features in the global coordinate. By using the open-sourced simultaneous localization and mapping(SLAM) algorithm which generates coordinate based on the fixed map frame. By transforming the features in the map coordinate, the “ground truth” global

position of the features can be obtained. The gmapping SLAM algorithm is based on the lidar sensor and odometry data. It should be mentioned that the precision of the ground truth position of the features is important to the performance of the PAUKF.

3.3 The process of the PAUKF evaluation

The PAUKF is implemented based on the robot localization framework which is a famous open-sourced localization package. The PAUKF is evaluated in two scenarios.

The evaluation process of the PAUKF in scenario 1 is illustrated. The process of the PAUKF evaluation in scenario 1 could be divided into two parts. First, the features' position in the global map frame should be obtained with a relatively accurate sensor and algorithm. Usually, in the commercial autonomous vehicle, manufacturing the map is an expensive, long production cycle and complex project based on the SLAM algorithm. Manufacturing this kind of commercial map for evaluating the PAUKF is not an appropriate solution. Thus, gmapping SLAM algorithm(use lidar sensor) and ar_track_alvar(use RGB-D sensor) package based simplified map is generated alternatively. The RGB-D camera also replaces the detection result that fusing the multiple sensors in the commercial autonomous vehicle about the surrounded road. Second, when the experiment, the lidar, and the SLAM algorithm not used. The PAUKF estimates the location of the vehicle only based on the RGB-D camera and the raw odometry.

The evaluation process of the PAUKF in scenario 2 is illustrated. In scenario 2, the quantity of the features(ar markers) is increased. And because of the shifting phenomenon of the gmapping, the ground truth value of the ar markers are measured manually for better precision. The total moving distance is decreased compared to scenario 1 which increases the precision of the raw odometry directly. The position of the features(ar markers) set in wall planar which makes it is easy to figure out the performance of the PAUKF. The lidar sensor is only used to run the LeGO-LOAM algorithm to provides a reference and generates the precise geometry of the corridor. The PAUKF only uses raw odometry and information about the features(ar markers) based on the depth camera. The result of the PAUKF is logged directly from the experiment not based on the rosbag.

3.3.1 Experiment scenario 1 setting

As illustrated above, the SLAM algorithm “gmapping” is used to provide the transformation of the vehicle. The SLAM algorithm fuses the distance data from the lidar sensor and the odometry data from UGV.

The test environment is the corridor of the 6th floor of the 7th department at the University of Ulsan. The initial view of the UGV and the panoramic photo of the test environment is shown in Figure 23.

Figure 23 Test scenario of PAUKF

Then the UGV moves slowly to accumulate the distance data from the lidar sensor and odometry data. If the UGV moves fast, the SLAM results become worse which means the accuracy of the location of UGV decreases.

The ar_track_alvar is used as feature detection. The ar_track_alvar detects the position based on

the specialized images which are shown in Figure 24. Each of the images has specialized ID. Therefore, there are no repeated recognition features in this test scenario.

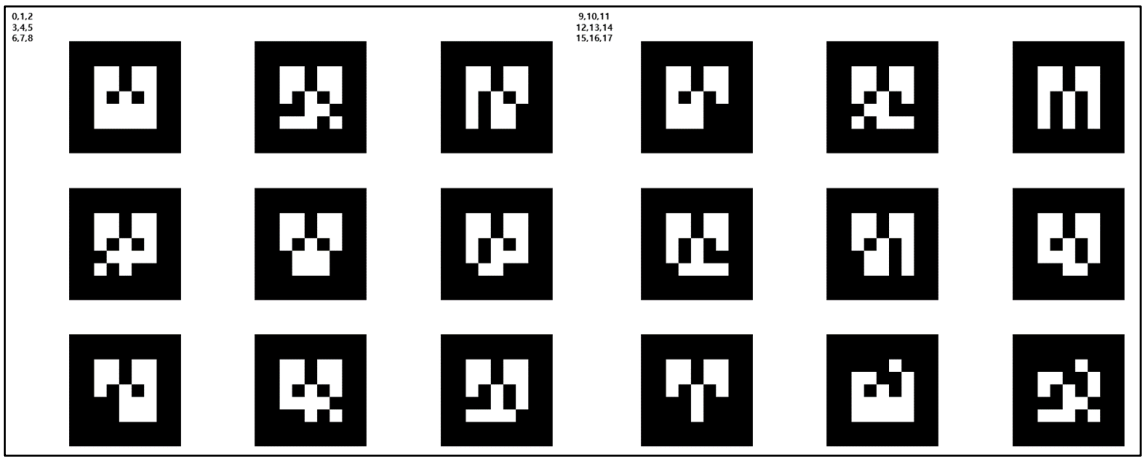


Figure 24 Ar track alvar package specialized image

The specialized images are stuck at the center of the corridor as Figure 25.

Figure 25 The randomly stuck specialized image

Then the RGB-D camera is used to detect the specialized images and transform them into the map frame as Figure 26 shows.

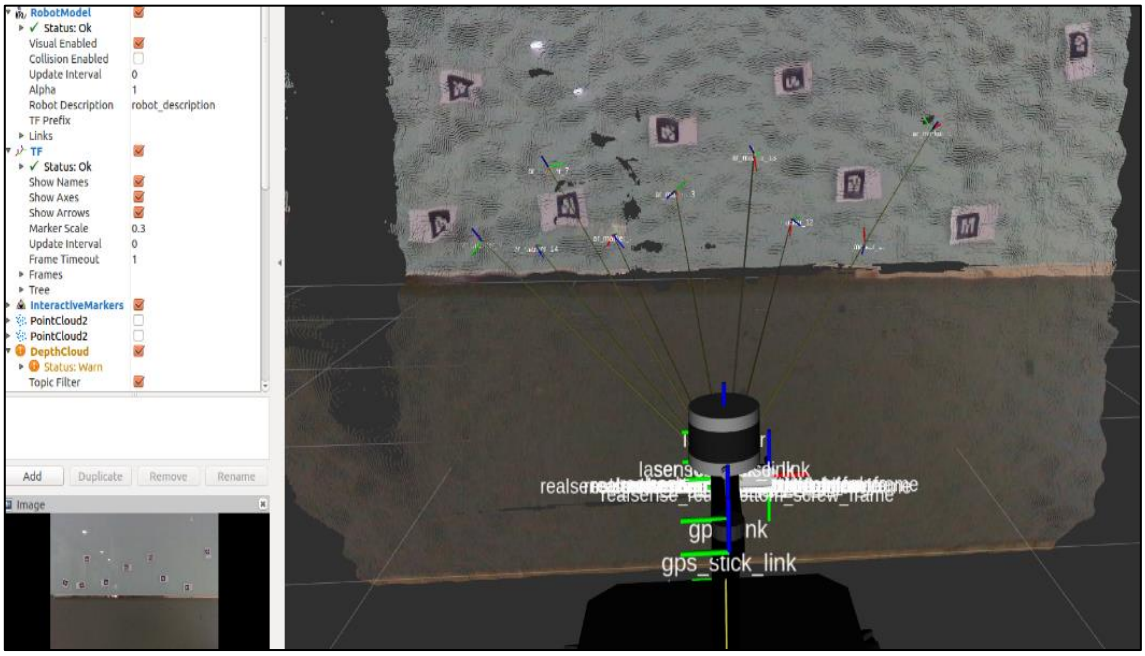


Figure 26 Detect the features based on RGB-D

The detected geometry of the corridor by the SLAM algorithm is shown in Figure 27. From Figure 27, it can be found that the corridor is bent as the UGV moves. It means, the detected features' position data also contains the shifting noise. The features' ground truth(GT) data can be measured by hand one by one. However, measured by hand is not a good solution to handle this problem. Because measures the features manually work in the small scale scenario. Whereas, the autonomous vehicle moves long distance usually. Considering the test in the outside in the future, the SLAM based feature detection and transformation algorithm was made. The slow movement and lidar sensor-based SLAM algorithm provide accurate features position in the map coordinate relatively. Therefore, the feature position data generated by the SLAM algorithm are treated as ground truth data.

From the generated map, it can be found, the map also contains accumulated orientation error. Thus, the "ground truth" position of the ar markers is a relatively precise position data. It means, even the PAUKF estimates perfectly, the precision of the estimation is limited by the precision of the ground truth data of the features.

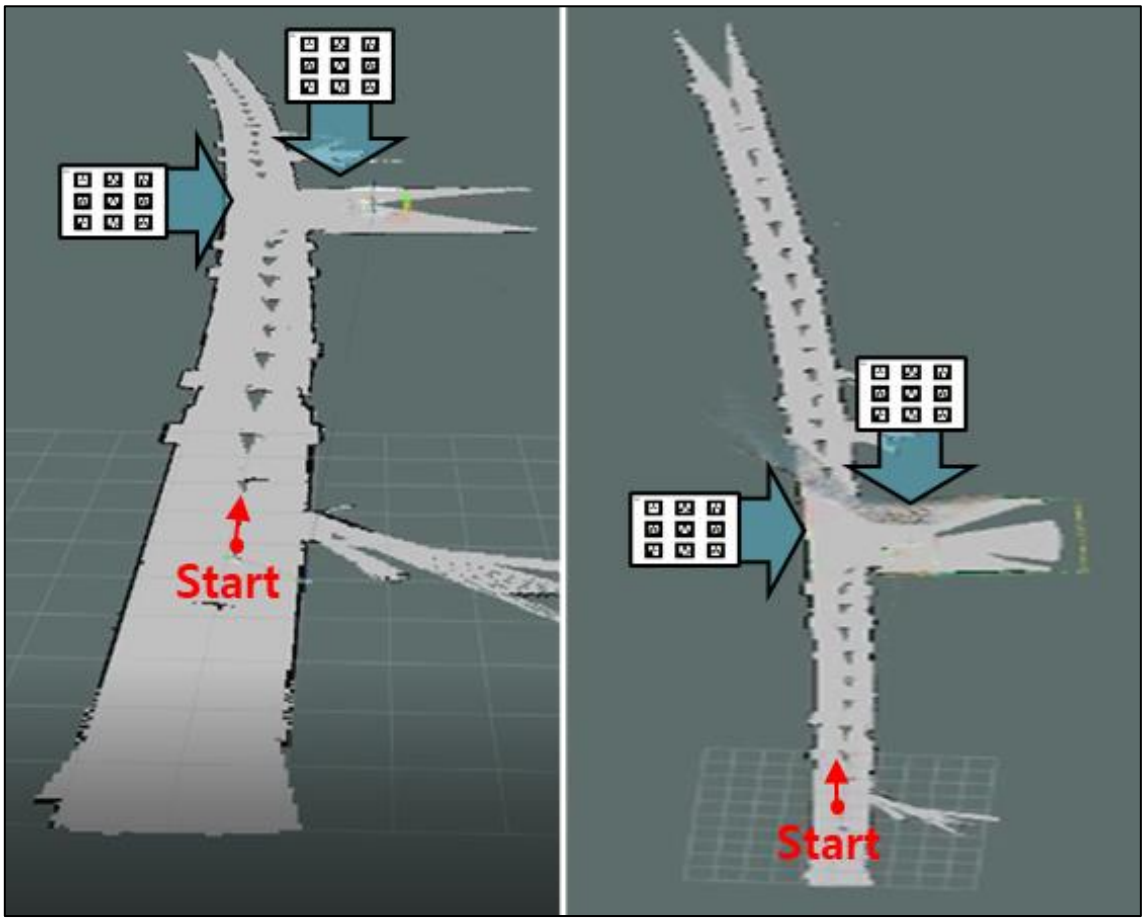
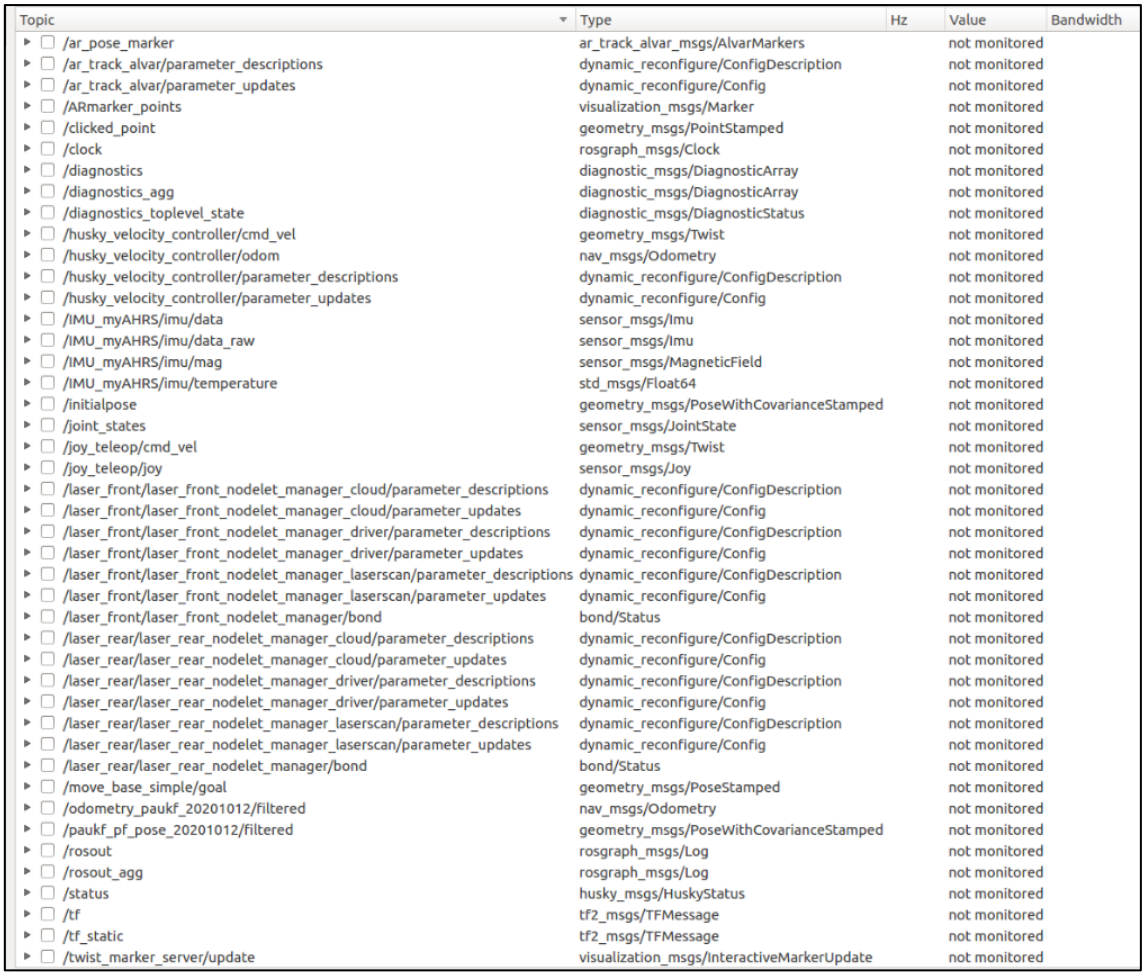


Figure 27 Features GT position obtained based on the SLAM algorithm

3.3.1.1 Recording the ROS bag for the test

The ground truth data of the features are obtained as section 3.4.2.1 introduced. The evaluation scenario of PAUKF is the same as the previous section. The difference between section 3.4.2.1 is the UGV localize the vehicle only with raw odometry and features. The lidar is not used in this test.

Before introducing the test of the PAUKF, one concept “ROS bag” should be mentioned first. ROS bag is a tool of the ROS system. ROS bag record subscribes to topics and writes a bag file with the contents of all messages published on those topics. Topic means the data stream of the ROS. The file contains interlaced, serialized ROS messages dumped directly to a single file as they come in over the ROS. After the record, the ROS bag, the data stream that is saved in the ROS bag can be played. It means the data from the ROS bag is the same as the real test completely as Figure 28 shows.



Topic	Type	Hz	Value	Bandwidth
/ar_pose_marker	ar_track_alvar_msgs/AlvarMarkers		not monitored	
/ar_track_alvar/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/ar_track_alvar/parameter_updates	dynamic_reconfigure/Config		not monitored	
/ARmarker_points	visualization_msgs/Marker		not monitored	
/clicked_point	geometry_msgs/PointStamped		not monitored	
/clock	rosgraph_msgs/Clock		not monitored	
/diagnostics	diagnostic_msgs/DiagnosticArray		not monitored	
/diagnostics_agg	diagnostic_msgs/DiagnosticArray		not monitored	
/diagnostics_toplevel_state	diagnostic_msgs/DiagnosticStatus		not monitored	
/husky_velocity_controller/cmd_vel	geometry_msgs/Twist		not monitored	
/husky_velocity_controller/odom	nav_msgs/Odometry		not monitored	
/husky_velocity_controller/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/husky_velocity_controller/parameter_updates	dynamic_reconfigure/Config		not monitored	
/IMU_myAHRS/imu/data	sensor_msgs/Imu		not monitored	
/IMU_myAHRS/imu/data_raw	sensor_msgs/Imu		not monitored	
/IMU_myAHRS/imu/mag	sensor_msgs/MagneticField		not monitored	
/IMU_myAHRS/imu/temperature	std_msgs/Float64		not monitored	
/initialpose	geometry_msgs/PoseWithCovarianceStamped		not monitored	
/joint_states	sensor_msgs/JointState		not monitored	
/joy_teleop/cmd_vel	geometry_msgs/Twist		not monitored	
/joy_teleop/joy	sensor_msgs/Joy		not monitored	
/laser_front/laser_front_nodelet_manager_cloud/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/laser_front/laser_front_nodelet_manager_cloud/parameter_updates	dynamic_reconfigure/Config		not monitored	
/laser_front/laser_front_nodelet_manager_driver/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/laser_front/laser_front_nodelet_manager_driver/parameter_updates	dynamic_reconfigure/Config		not monitored	
/laser_front/laser_front_nodelet_manager_laserscan/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/laser_front/laser_front_nodelet_manager_laserscan/parameter_updates	dynamic_reconfigure/Config		not monitored	
/laser_front/laser_front_nodelet_manager/bond	bond/Status		not monitored	
/laser_rear/laser_rear_nodelet_manager_cloud/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/laser_rear/laser_rear_nodelet_manager_cloud/parameter_updates	dynamic_reconfigure/Config		not monitored	
/laser_rear/laser_rear_nodelet_manager_driver/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/laser_rear/laser_rear_nodelet_manager_driver/parameter_updates	dynamic_reconfigure/Config		not monitored	
/laser_rear/laser_rear_nodelet_manager_laserscan/parameter_descriptions	dynamic_reconfigure/ConfigDescription		not monitored	
/laser_rear/laser_rear_nodelet_manager_laserscan/parameter_updates	dynamic_reconfigure/Config		not monitored	
/laser_rear/laser_rear_nodelet_manager/bond	bond/Status		not monitored	
/move_base_simple/goal	geometry_msgs/PoseStamped		not monitored	
/odometry_paukf_20201012/filtered	nav_msgs/Odometry		not monitored	
/paukf_pf_pose_20201012/filtered	geometry_msgs/PoseWithCovarianceStamped		not monitored	
/rosout	rosgraph_msgs/Log		not monitored	
/rosout_agg	rosgraph_msgs/Log		not monitored	
/status	husky_msgs/HuskyStatus		not monitored	
/tf	tf2_msgs/TFMessage		not monitored	
/tf_static	tf2_msgs/TFMessage		not monitored	
/twist_marker_server/update	visualization_msgs/InteractiveMarkerUpdate		not monitored	

Figure 28 Data stream output from ROS bag

In other words, the ROS bag allows the PAUKF can be tested in the lab only with one data log. Thus the test scenario illustrated in the following content is the test that logged for final estimation for this thesis. The test data is logged with the wired connection to the UGV. The command that sends to UGV without data loss.

The features detected from ar_track_alvar package are critical to the final estimation result. Therefore, the performance of feature detection should be mentioned first. In the real test, the specialized images are stuck at the wall of the center of the corridor as Figure 25 shows. The performance of the detection is shown in Figure 29 shows. The local coordinate is the same as the previous sections illustrated. The forward direction is the +x axis. The red dashed circle is the perception error, and the yellow dashed rectangle is the correct perception result. The detected relative

distance value is evaluated manually, and the perception error to sensor coordinate is 1-10cm roughly. Figure 29(a) and Figure 29(b) show the perception module detects the images which do not exist and obtained the position of the image is not exist. Figure29(c) also shows the perception module detects the image that does not exist. Compare to the (a) and (b), the detected feature's position is close to the correct perception results which makes it hard to distinguish the correct features and false detected features. Figure 29(d) shows the position of the detected value is jumped randomly. The detected feature ID is correct, however, the relative distance changes randomly. This is the most tricky type of false detection. The reason for this kind of detection error could be happened because of the detection algorithm, the light condition of the scenario, the functional error of the sensor itself. This phenomenon also happens in the real autonomous vehicle in which the scenario is much more complex and hard to perceive. Therefore, the particles are used to handle this tricky problem.

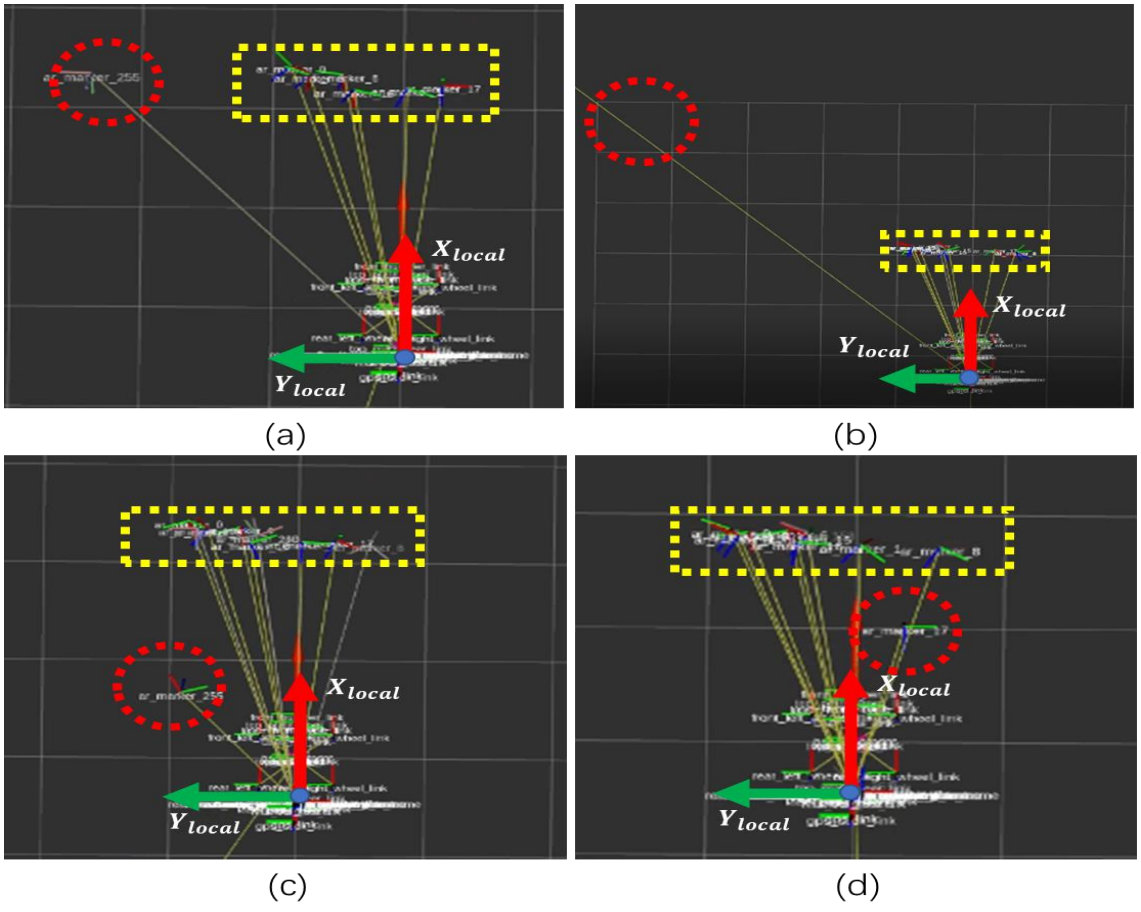


Figure 29 The perception errors(yellow: normal, red circle: unexpected error)

The command value of linear velocity in the x-direction and angular velocity in the z-direction is shown in Figure 30.

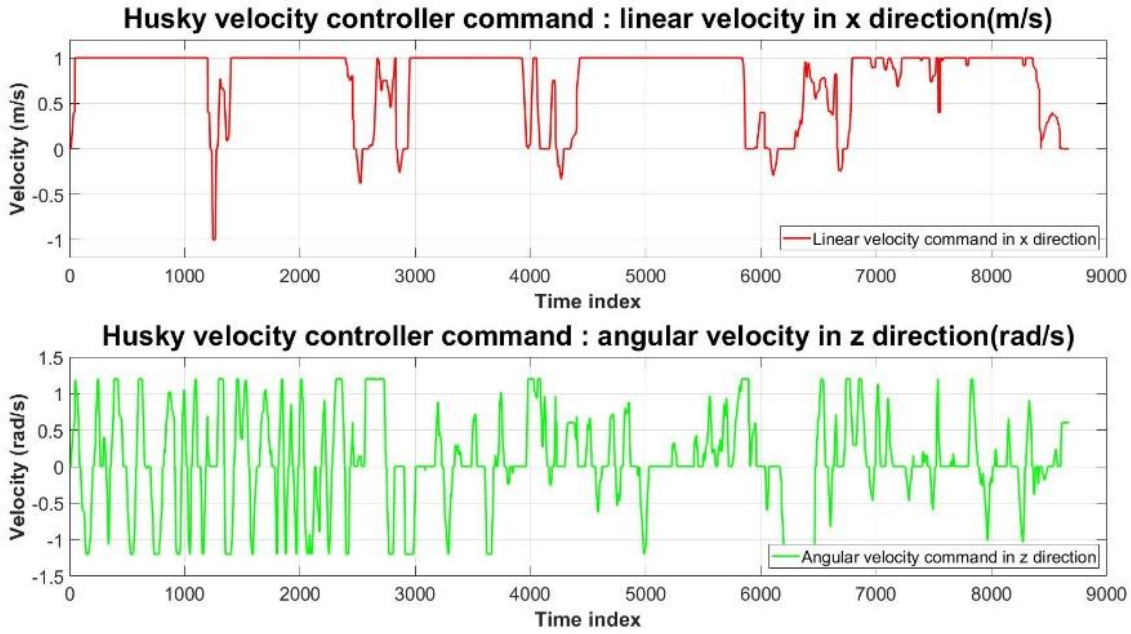


Figure 30 Command send to the UGV

It can be found the maximum velocity of the UGV is 1m/s and it changes a lot. The angular velocity in z-direction which is the yaw angle of the UGV is also changed a lot. This command lets UGV's trajectory looks like an S shape most of the time. The features' position and the light condition are the same as in section 3.4.2.1. The detected features are not transformed into the map coordinate anymore. PAUKF uses these features to localize itself in the Odom frame.

The movement of the UGV is illustrated in Figure 31. Figure 31 is only to demonstrates the movement of the UGV. The arrows in Figure 31 are not the ground truth trajectory of the UGV. As illustrated above, the ar markers are used as features for the particle filter. When the UGV starts to move, it is controlled by the command that sends to the UGV. Before the UGV detects the ar markers for the first time, the PAUKF only has the measurement from the raw odometry which is from the differential wheel of the UGV. It means the PAUKF has to believe the raw odometry from the UGV. By using the raw odometry of the UGV, the PAUKF only can estimate the location based on the dead

reckoning. Once reach the center of the corridor where the ar markers are stuck, the relocation and correction behavior should happen when UGV detects the ar markers based on the depth camera. The UGV only can detect the ar markers 2 times when it far away, and detects the ar markers 2 times when it comes back to the start point. It means the relocation and correction behavior of PAUKF estimation should happen 2 times relatively.

The IMU sensor also can be used, however, it only can provide a relative measurement. In this thesis, the main contribution of the PAUKF is that the PAUKF can extract the global location data based on the map matching. Of course, the usage of the IMU can improve the performance, however, that is not the main interest in this thesis. In this experiment, to figure out the global relocation performance of PAUKF is the main purpose. Therefore, the main performance of the PAUKF is evaluated only based on the raw odometry measurement, and at the last of chapter 3, the performance of the PAUKF with IMU is illustrated for reference of using the IMU measurement.

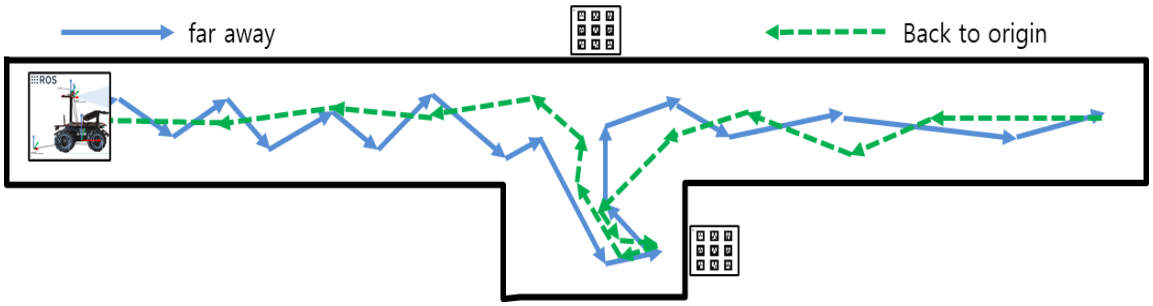


Figure 31 The movement diagram of experiment 1

Figure 32 shows the controller and computing devices when experiments with the PAUKF with UGV. The mini PC is used as collecting the sensor data for the first step. Then all the data should be transformed into the base_link frame of the robot in the transformation tree by using the tf library of the ROS. Then all the data including the RGB-D camera data, IMU data, transformation data, communication data are uploaded to the ROS master node. By join the ROS network based on the ethernet cable, the notebook can access the data of the data in the mini PC. Then the main algorithm of the PAUKF runs on the notebook and the command to the UGV is based on the joystick.

Figure 32 Controller and computing devices used in the experiment

3.3.2 Experiment scenario-2 setting

The trajectory of the UGV is overlapped because of the location of the ar markers position. This will make it hard to figure out the relocation and correction phenomenon of the PAUKF. Compare to environment 1, the ar markers are distributed in four different positions to provide position measurement. The ground truth position of the ar markers in scenario 2 is measured by the ruler. This is because the ground truth position of ar markers from the gmapping algorithm tends to shift as Figure 27 shows. It makes the estimation error bigger when estimation has not yet started. The scenario is the same corridor as experiment 1. The difference is the UGV moves more straightly and the ar markers are more distributed in four positions as Figure 33 shows. The selection of the ar marker position is arbitrarily selected.

Figure 33 The ar markers used in experiment 2

Figure 34 shows the movement diagram of the UGV in experiment 2. The UGV moves more straight forward and there are no markers in the far away process. The ar markers are only detected when the UGV comes back to the start position. The black dot in Figure 34 is the accumulated point cloud extracted based on the LeGO-LOAM algorithm[103]. It should be mentioned that the gray line and green dashed line in Figure 34 is just for illustration of the trajectory of the UGV, it is not the actual trajectory. There is no accurate position measurement device to get the ground truth trajectory of the UGV.

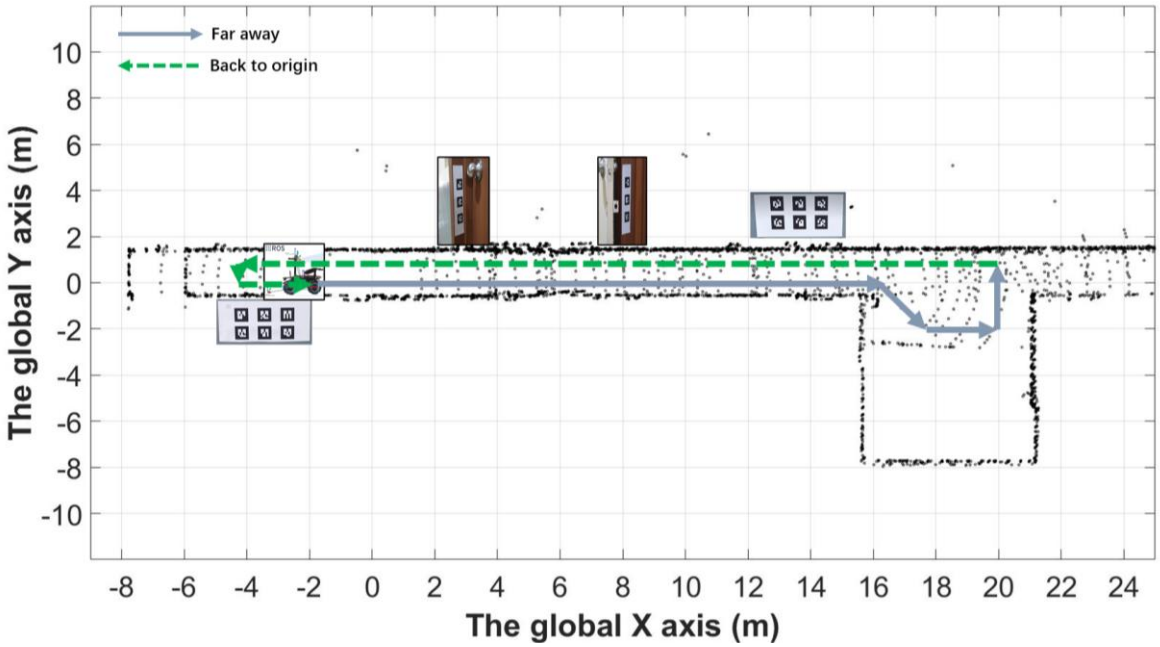


Figure 34 The movement diagram of experiment 2

The detection error phenomenon also happens as in Figure 29, therefore, the detection of the ar marker also contains uncertainty. It will affect the performance of the PAUKF. It will be discussed in the next section. The command to the UGV is the velocity in the x-direction and the angular velocity in the z-direction as the Figure 35 shows. As illustrated in the above contents, the movement of the UGV in experiment scenario 2 is straight forward. Therefore, the velocity in the x-direction is no negative value and the angular velocity in the z-direction is not fluctuate as in experiment scenario 1. The rosbag was also recorded for repeatable monitoring and results extracting.

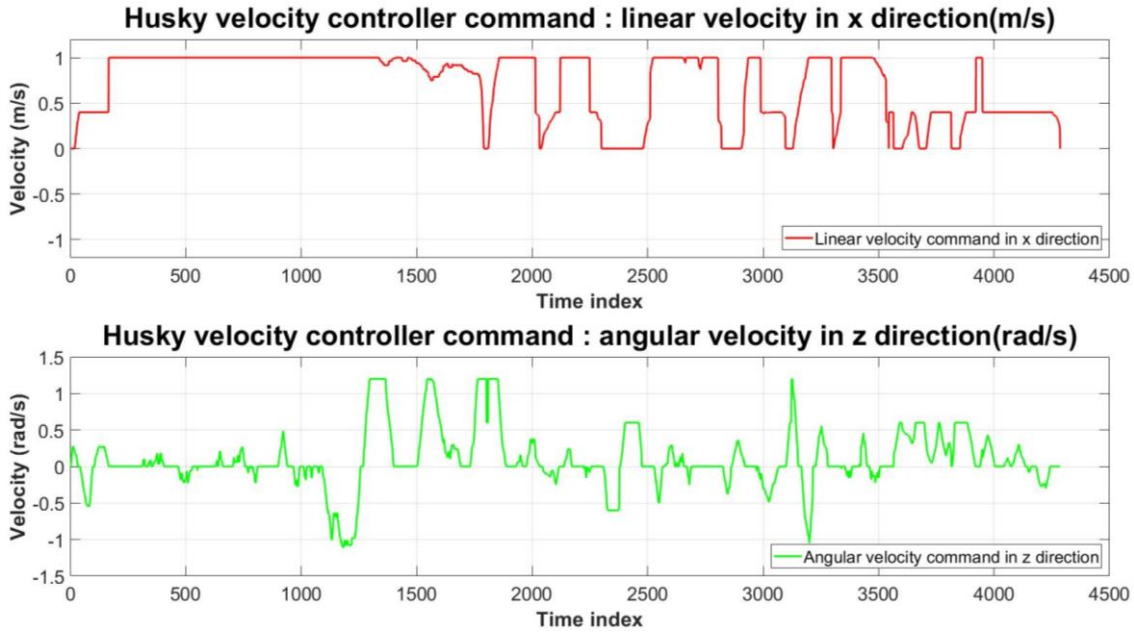


Figure 35 The command to the UGV in experiment 2

3.4 Experiment result analysis

This section illustrates the test result of the PAUKF with the real unmanned ground vehicle. For better readability, the PAUKF is represented as short as PAUKF. As introduced in section 3, the raw odometry of the UGV and the features collected by the depth camera is used. The test data is saved by using the ROS bag with a wired connection. The command to the UGV is sent by the controller which is connected to the notebook. The main algorithm run in the notebook and the basic devices launch files are run in the mini PC which is mounted in the UGV. The ROS master is the UGV and the notebook becomes the client of the ROS network. It means the computation burden of the mini PC mainly from the basic launch files that activate devices and sensors. The object detection is based on the ar track alvar packages, SLAM algorithm when collects ground truth data of features, and the PAUKF is all run in the notebook.

The specification of the notebook is shown in Table 3.

Table 3 The specification of the notebook

Name	Specification
Operation System	Ubuntu 16.04
Robot control framework	ROS kinetic
Processor	Intel 8th i7
Graphic card	NVIDIA GTX1050
Memory	12GB

The frequency and bandwidth of the data stream on the ROS network are shown as Table 4. The data stream contains the data which type is the data type column. The /ar_pose_marker contains the detected markers data stream, /husky_velocity_controller/odom contains the raw odometry provided by the UGV, /joy_teleop/cmd_vel contains the command send to the UGV, /tf contains the transformation between all the coordinates defined, /pose_from_PF contains the estimated UGV's position based on the ar marker and /odometry_paukf/filtered contains final UGV odometry data which is estimated by the PAUKF. The frequency of the data stream also can be increased, 10Hz is selected as the frequency of the PAUKF according to the perception module. From Table 4, it can be found that the frequency of the data in the ROS network is different, the ROS mechanism makes PAUKF receives the latest measurement.

Table 4 The frequency, data type and bandwidth of each data

Datastream name	Data type	Frequency(Hz)	Bandwidth
/ar_pose_marker	ar_track_alvar_msgs/AlvarMarkers	8	5.34KB/s
/husky_velocity_controller/odom	nav_msgs/Odometry	10	7.17KB/s
/joy_teleop/cmd_vel	geometry_msgs/Twist	81.93	3.88KB/s
/tf	tf2_msgs/TFMessage	37.81	9.50KB/s
/pose_from_PF	Geometry_msgs/PosewithCovarianceStamped	10	3.67KB/s
/odometry_paukf/filtered	nav_msgs/Odometry	20	14.21KB/s

3.4.1 The results analysis based on scenario1

The total test was done in the indoor scenario, there is no ground truth for calculating the RMSE value as in the outdoor(for example RTK-GPS). Thus one appropriate way to how to access the accuracy of the filtered estimation results are the differences between the boundary of the scenario and odometry data. In this thesis, the boundary of the scenario is a closed corridor. Therefore, the geometry of the wall and the UGV's odometry data can be used to compare the algorithm's performance. In the following section, a new word PAEKF means the particle aided extended Kalman filter which has the same framework as PAUKF.

The experiment diagram is shown in Figure 36. The data used in the 3.4.1.1-3.4.1.3 section are only raw odometry and ar markers. Section 3.4.1.4 shows the performance of the PAUKF with IMU for reference.

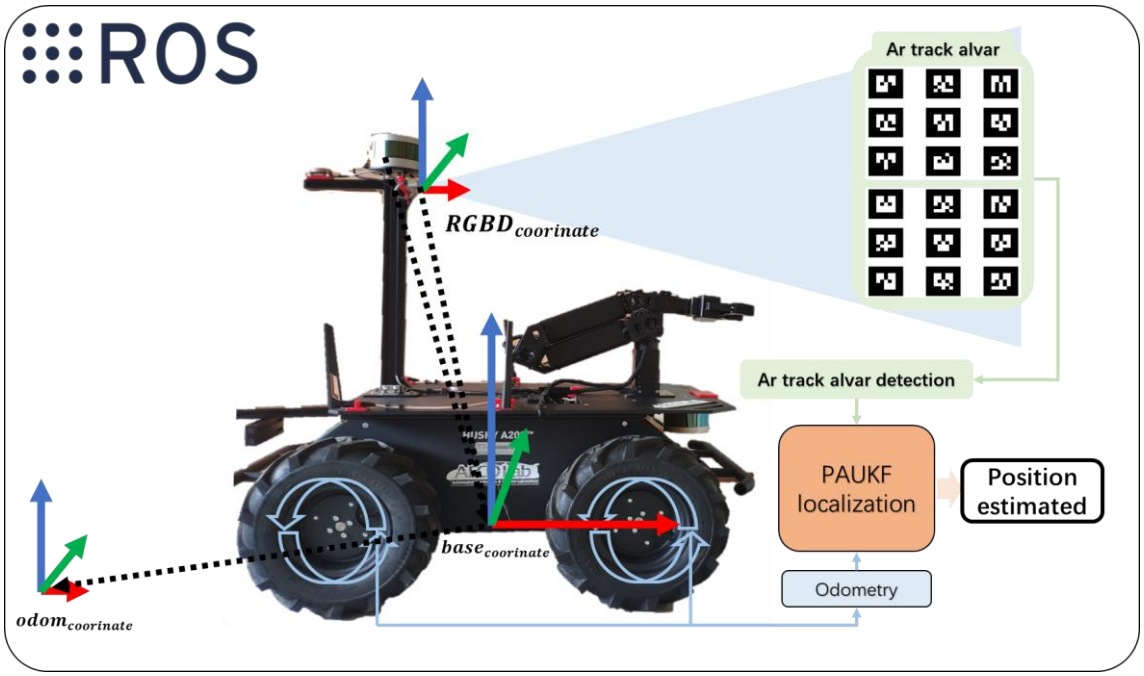


Figure 36 Experiment diagram of the PAUKF with raw odometry and ar markers

3.4.1.1 The trajectory of raw odometry, EKF, and UKF

To compare the performance of the filters, the trajectory of the raw odometry from UGV, EKF

filtered odometry and UKF filtered odometry are shown in Figure 37. There are four different kinds of lines in Figure 37. The solid line with black color is the boundary of the corridor. The geometry of the corridor is measured manually. The solid line with red color is the raw odometry data from the UGV. The dotted line with the black circle is the odometry data filtered by the EKF. The dotted line with purple color is the odometry data filtered by the UKF. There is no measurement value like ar marker is used in this estimation results. From Figure 37, it can be found, all the odometry data from different sources are shifted from the boundary of the corridor. At first, all the odometry data is in the boundary of the corridor at the start point. However, as the UGV moves, the accumulated noise affects the odometry data. All of the odometry data start to shifts to the left side of the UGV. Since there is no efficient relocation algorithm based on the ar markers, the EKF and UKF have to believe the odometry data from UGV and filtering the noise based on the model of the UGV. As a result, the estimation result of the EKF and UKF also shifted as the raw odometry data. At the end of the odometry, it can be found all the odometry from the different sources is far away from the start position. The UGV moved around in the corridor and moved back to the start point. It means the final location estimation error is 3.892m in the x-axis, -10.93m in the y-axis with raw odometry, 4.795m in the x-axis, -11.89m in the y-axis with EKF filtered, and 4.153m in the x-axis, -11.82m in the y-axis with UKF filtered respectively as Table 5. The performance of the UKF and EKF is worse than the raw odometry in this scenario. The enlarged trajectories at the center of the corridor are shown in Figure 38. The performance of the EKF and UKF is worse is not because of the filter itself. The reason why the EKF and UKF work worse is they only have the measurement with raw odometry.

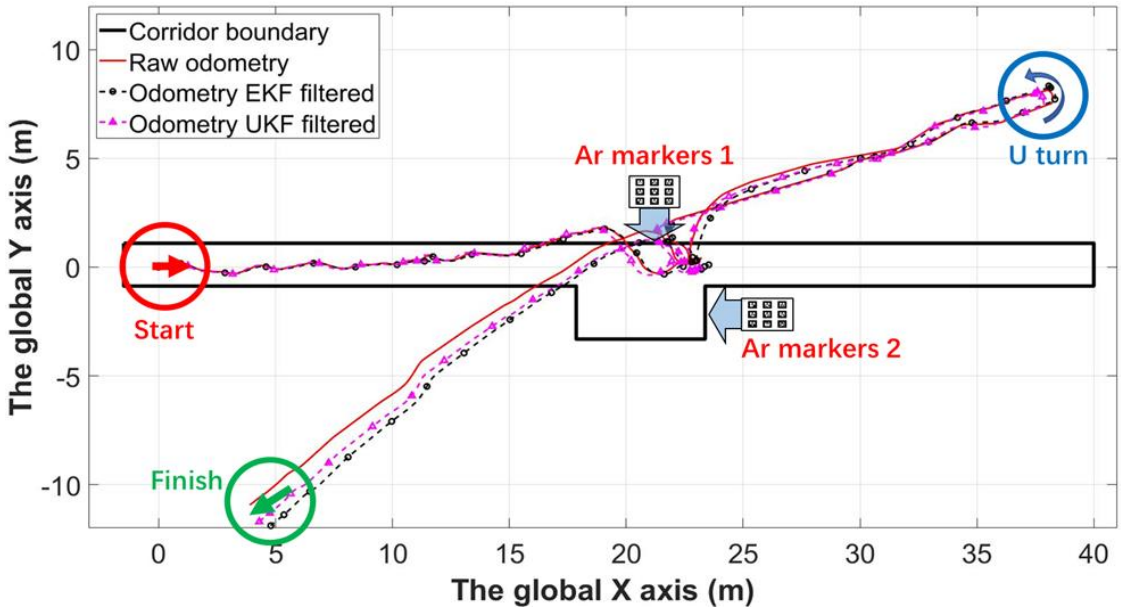


Figure 37 The trajectories of the raw odometry, EKF and UKF

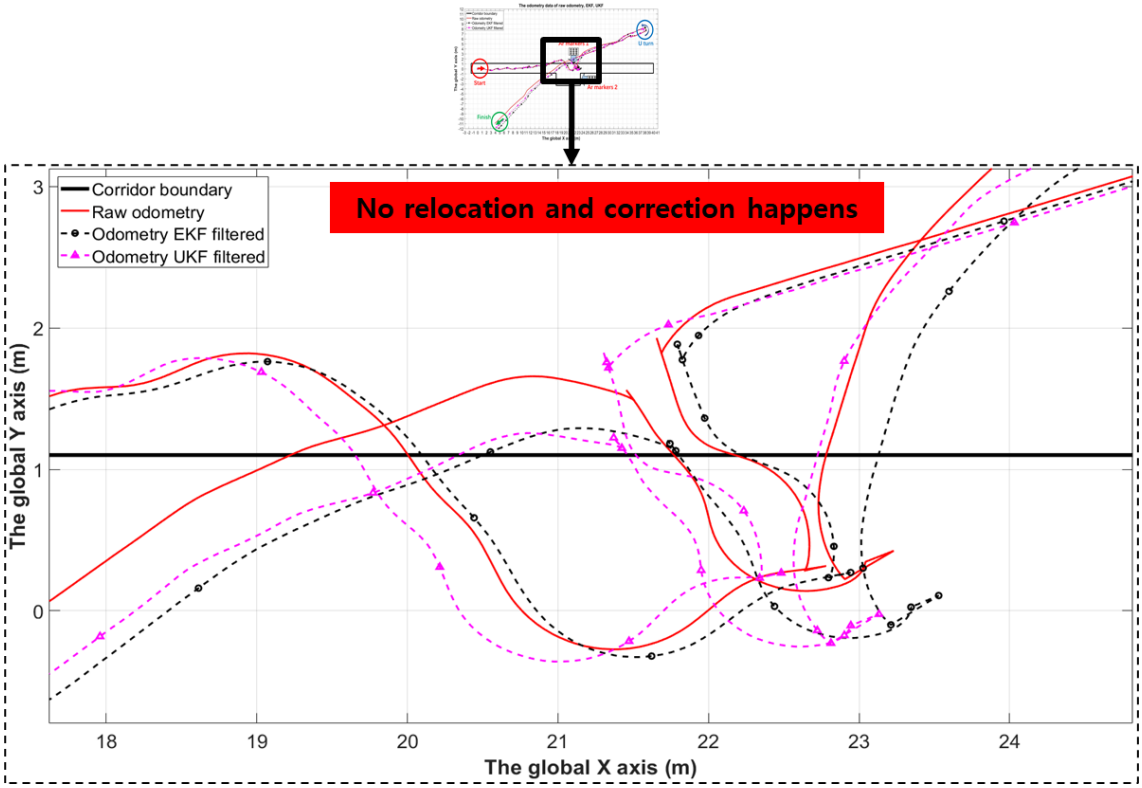


Figure 38 Enlarged trajectories in the center of the corridor

Table 5 Error of the different estimation method at the final position

Estimation method	Longitudinal error(m)	Lateral error(m)
Raw odometry	3.892	-10.93
EKF	4.795	-11.89
UKF	4.153	-11.82

3.4.1.2 The trajectory of raw odometry, PAUKF with different particles

This section illustrates the trajectory of the raw odometry and PAUKF with different particles in section 4.4.2. Figure 39 shows the trajectories of the PAUKF with different particles and Figure 40 shows the enlarged plot of the PAUKF in the center of the corridor. The number of the particles of PAUKF is set as 100, 500, 1000, and 2000 particles. Relocation happened when the UGV detects the ar markers. The error of the final position is referred to as the evaluation parameter. The error of the different estimation methods in the final position is shown in Table 6. From Table 6, a phenomenon that can be found in the estimation precision of the PAUKF is also in positive proportion to the number of the used particles. The final error of the UGV back to the start point is 0.552m in the longitudinal direction, and -1.643m in the lateral direction. The improvement of the precision is increased proportionally according to the number of the particle.

The upper limit of the precision is decided by the accuracy of the ground truth position of the ar markers. The more the ground truth position of the ar markers is, the preciser the PAUKF will be. The error of the PAUKF is only based on the raw odometry and the 4 times of the ar markers detection without additional sensor. The result shows the advancement of the PAUKF.

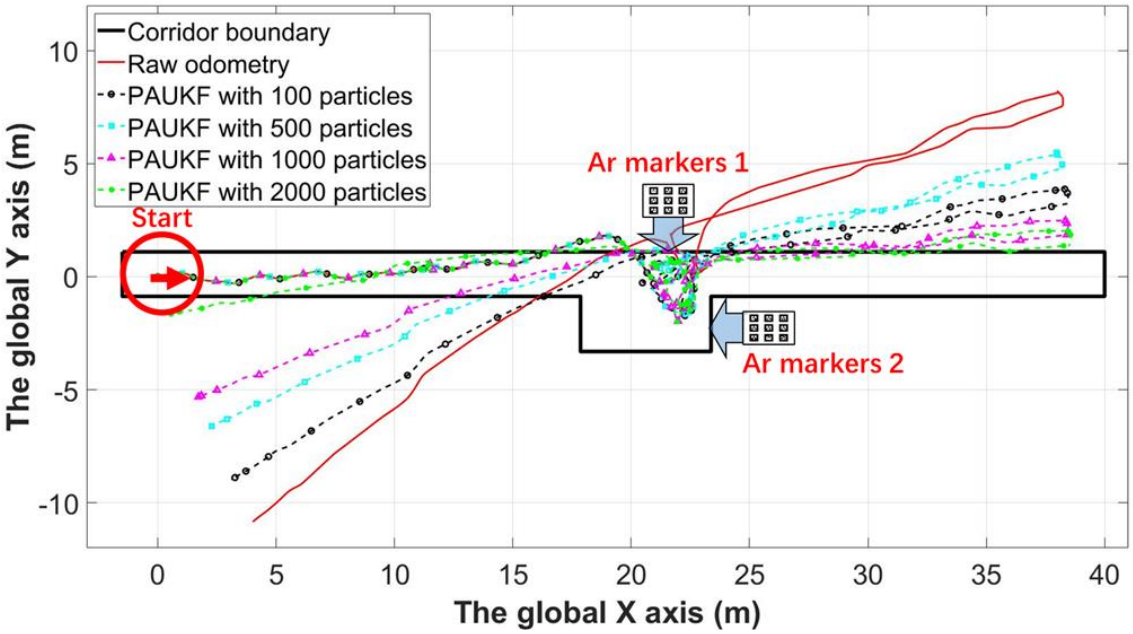


Figure 39 The trajectories of the PAUKF with different particles

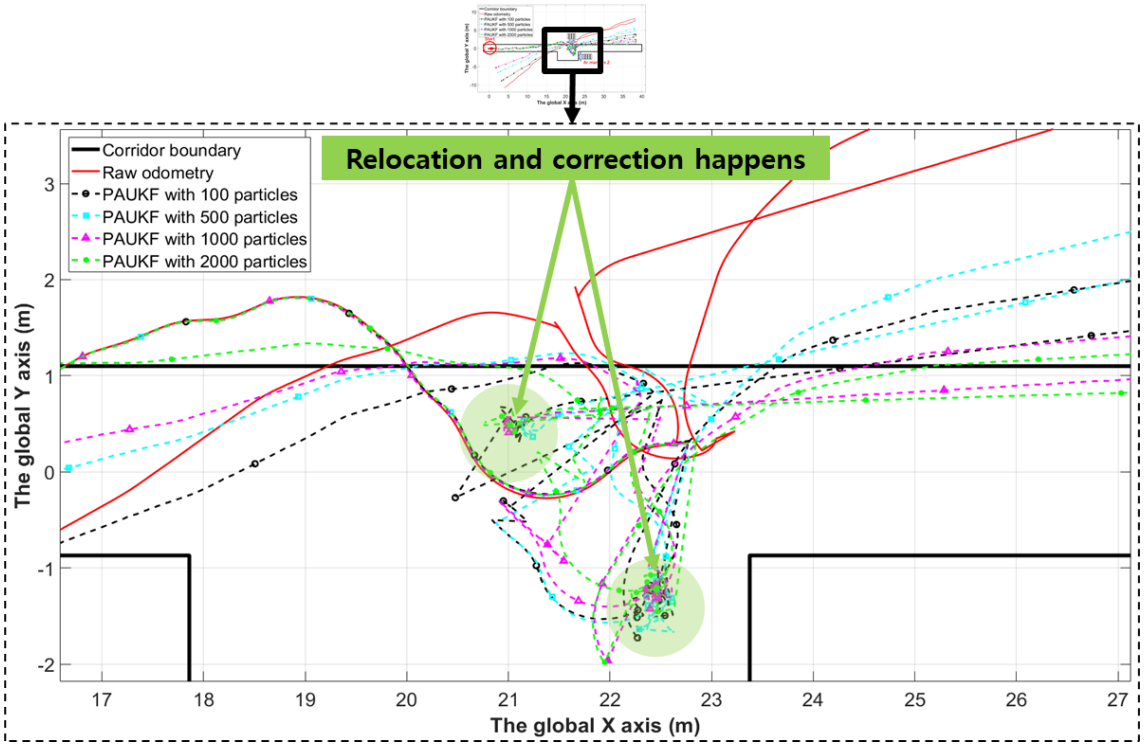


Figure 40 Enlarged trajectories in the center of the corridor

Table 6 Error of the different estimation method at the final position

Estimation method	Longitudinal error(m)	Lateral error(m)
Raw odometry	3.892	-10.930
PAUKF with 100 particles	3.160	-8.946
PAUKF with 500 particles	2.172	-6.648
PAUKF with 1000 particles	1.642	-5.342
PAUKF with 2000 particles	0.552	-1.643

3.4.1.3 The trajectory of raw odometry, EKF, UKF, PAEKF and PAUKF

The comparison of the different estimation results is shown in Figure 41. The PAEKF means the PF is used as the pre-processing of the EKF. The raw odometry, EKF filtered without PF, UKF filtered without PF, PAEKF with 2000 particles, and PAUKF with 2000 particles. From enlarged Figure 42 and Table 7, it can be found the PAUKF with 2000 particles performs best. Despite the performance of the EKF and UKF shows little differences the performance of the EKF and the UKF is random since there is no appropriate way to receive a global position to correct. Only if the algorithm run many loops, then the performance of the filters can be normalized.

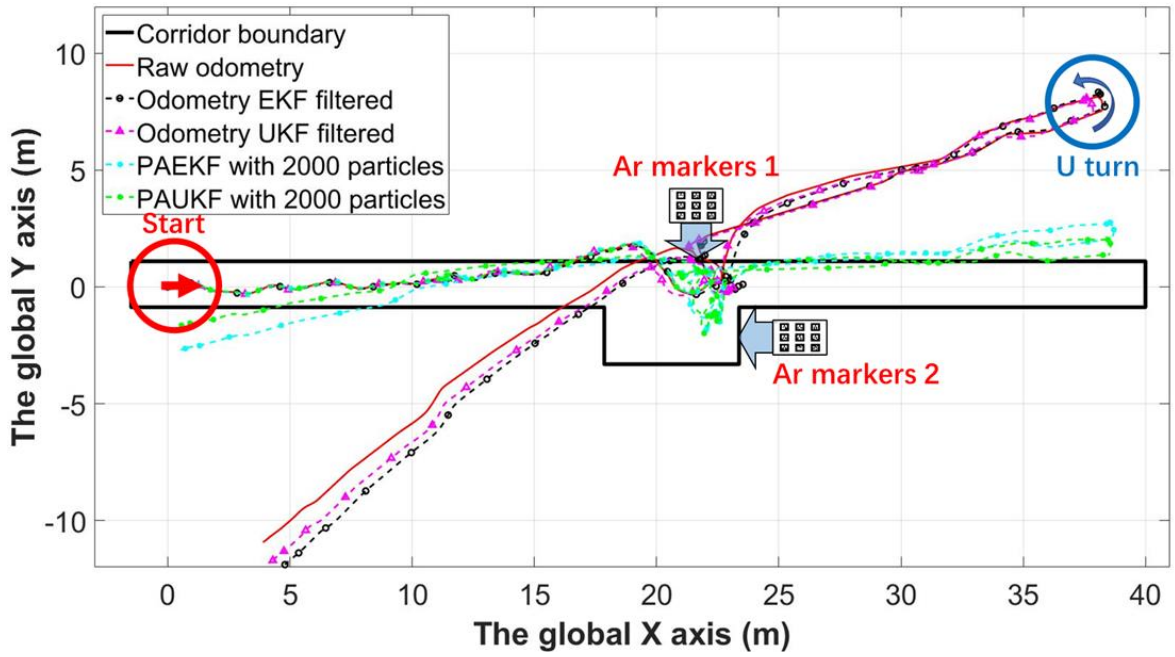


Figure 41 The trajectories of the raw, EKF, UKF, PAEKF, and PAUKF

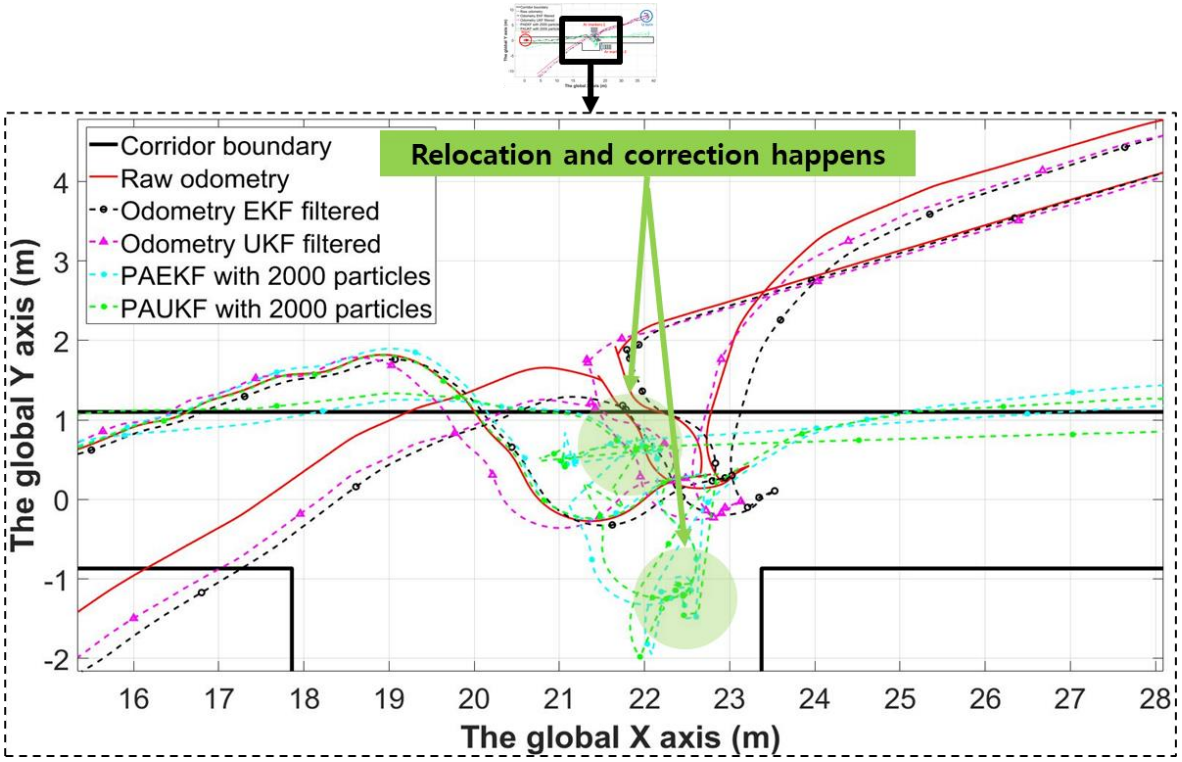


Figure 42 Enlarged trajectories in the center of the corridor

Table 7 Error of the different estimation method at the final position

Estimation method	Longitudinal error(m)	Lateral error(m)
Raw odometry	3.892	-10.93
EKF	4.795	-11.89
UKF	4.153	-11.82
PAEKF with 2000 particles	0.52	-2.688
PAUKF with 2000 particles	0.5519	-1.643

3.4.1.4 The performance of the PAUKF with IMU sensor

The performance of the filtering algorithm can be improved by using more sensors. However, the ultimate goal of the experiment is to figure out the performance of PAUKF localization rather than prove the performance of the sensor itself. The relocation and correction behavior is the focus and novelty of the PAUKF. Therefore, in previous sections, the PAUKF is evaluated only based on the raw odometry.

To gives a reference performance of PAUKF with more sensors, the PAUKF is evaluated by

using the raw odometry of the UGV, ar markers, and additional IMU sensor. The raw odometry provides linear velocity and yaw angle measurement based on the differential wheels and the IMU provides additional linear acceleration and angular velocity based on the devices in the IMU. It should be noticed that the raw odometry data type provides from the UGV contains pose and twist data. However, since the UGV uses the movement of differential wheels for calculating the pose and twist, the pose and twist data are calculated from the linear velocity and angular velocity. It means the effective data source of the UGV is only linear velocity and angular velocity. The pose of the UGV is calculated from the linear velocity and angular velocity. The position of the UGV is generated by using the yaw angle and linear velocity in the linear velocity in the X-direction. The UGV only can move in the X-direction and the rotation angle(yaw) is generated by the differential wheels. Thus the location of the UGV is decided only by using the velocity in the X-direction and the angular velocity in the Z-direction. So the velocity in the X-direction and the angular velocity in the Z-direction are used as the measurement of the state of UGV. Of course, the data from UGV's raw odometry can be used as a measurement, however, that is inefficient since the data sources are duplicated. The IMU provides the acceleration measurement in the X direction and the angular velocity of the UGV in the Z direction. The yaw angle of the IMU is generated by accumulating the angular velocity in the Z direction. For intuition, the yaw angle of the IMU should be used as the measurement value. However, because of the sudden acceleration and changeable moving behavior, the yaw angle from the IMU is too worse to be used. Thus, the IMU only provides the rotation velocity in the Z direction and acceleration measurement in the X-direction. Of course, the measurement of the IMU in the x and y-axis can be used. But since the corridor is a planar scenario, the measurement of angular velocity and linear velocity in the x and y-axis will only add additional noises to the estimation. The diagram of the experiment of the PAUKF with IMU is shown in Figure 43. The position of the IMU is attached at the bottom of the UGV. The coordinate of the IMU frame is integrated into the full coordinate system of the UGV. It means, all the data from the different kinds of sensors is transformed by using the tf library of the ROS into the base link which is an ideal point of the UGV.

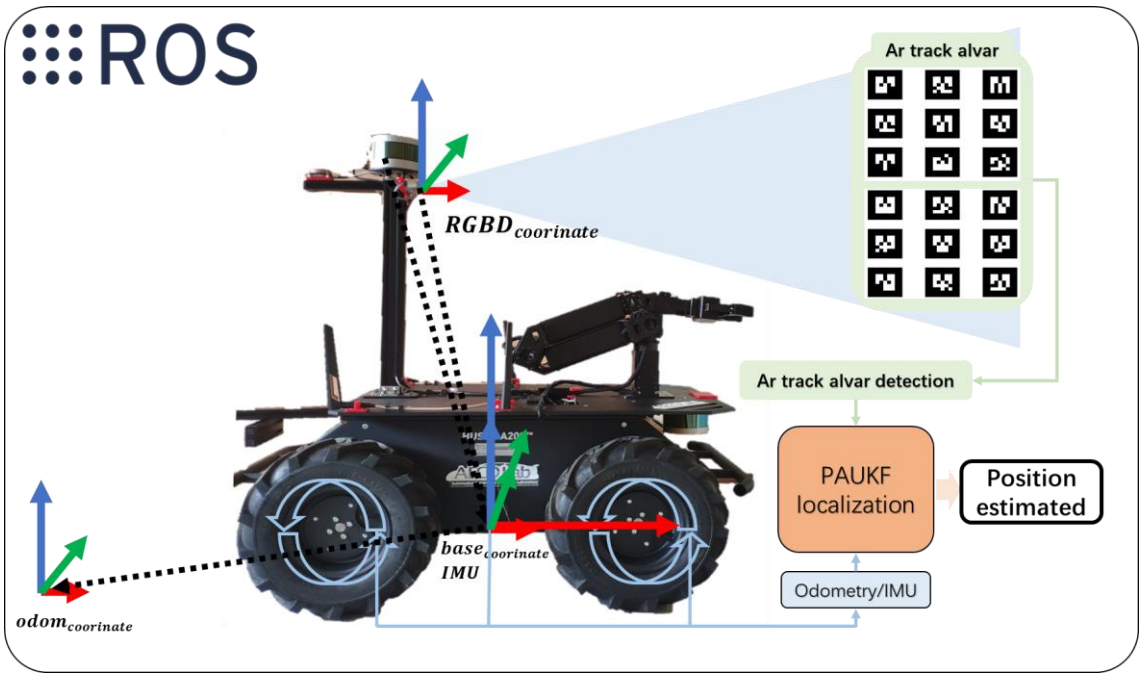


Figure 43 Experiment of the PAUKF with raw odometry, IMU, and ar markers

The trajectories of the raw odometry, PAUKF with raw odometry and ar markers, and PAUKF with raw odometry, ar markers, and IMU are shown in Figure 44. The difference is the usage of the IMU sensor. Both of the PAUKF trajectories use the ar marker. The red line is the trajectory of the raw odometry, the green dashed line with the dot is the trajectory of the PAUKF without IMU and the purple dashed line with the triangle is the trajectory of PAUKF with IMU measurement. Both of the PAUKF estimations use the ar marker, thus the filtered trajectories are close to the corridor.

Compare to the PAUKF without the IMU, the PAUKF with IMU shows better estimation results. When the UGV moves from the start point on the left side of the corridor, the estimations of the PAUKF start to performs differently. The estimation of the PAUKF without the IMU becomes shifting to the left side of the UGV's moving direction. This phenomenon is happening because the yaw angle from the raw odometry accumulates the error from the differential wheels. Compare to the PAUKF without IMU, the PAUKF with IMU estimates the movement better by using the linear acceleration and angular velocity measurement. Thus the estimated trajectory of the PAUKF before the center of the corridor is always in the boundary of the corridor. When UGV moves to the center of the corridor,

both of the PAUKF estimations relocate the position of the UGV and correct the state based on the ar markers. Thus the relocated position of the UGV is the same as both of the PAUKF estimation. After both of the PAUKF relocation and correction, the UGV moves to the left side of the corridor.

When the UGV passes the U-turn, the estimation of the PAUKF becomes different again. The PAUKF with IMU tends to correct the yaw angle to the right side of the UGV's moving direction, and the PAUKF without IMU tends to the left side. This phenomenon always exists in the whole track of the UGV. Thus, when the UGV passes the center of the corridor, relocation, correction, the phenomenon happens again. As a result, the final position of the PAUKF with IMU is estimated at the left side of the start point and the final position of PAUKF without IMU is estimated at the right side of the start point. The PAEKF(EKF with PF pre-processing) with IMU is omitted because the performance trend is the same with PAUKF.

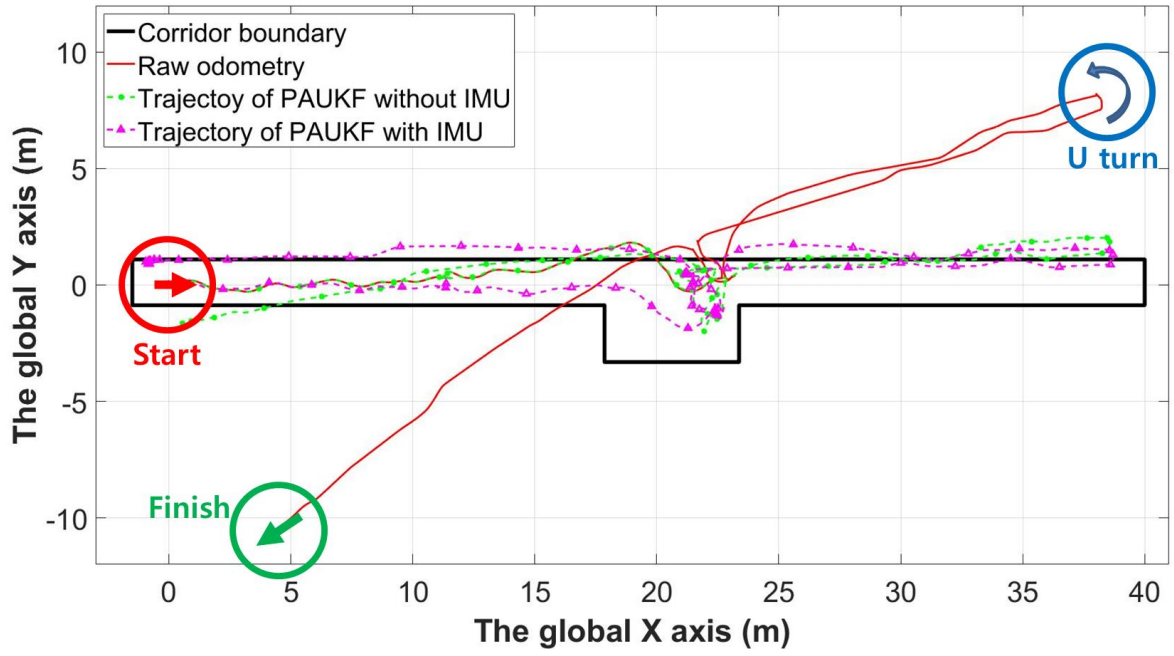


Figure 44 The trajectories of PAUKF with IMU and others

Figure 45 is the enlarged estimated trajectories. It can be found clearly that the PAUKF with IMU estimated the position better when UGV moves to the center of the corridor. In the contrast, the PAUKF without the IMU sensor shifted to the left side of the UGV moving direction and the

estimated trajectory goes over the boundary of the corridor. When UGV detects the ar markers, both of the PAUKF relocate the position and correct orientation of the UGV.

Therefore, a conclusion can be made is adding additional sensors can improve the performance of the estimation. However, without the particle filter based pre-processing the information of the surrounded features can not be used appropriately. The key feature of the PAUKF is fusing multiple data sources. The usage of the IMU can provide relative dead reckoning performance, however, it still can't provide the relocation and correction functions.

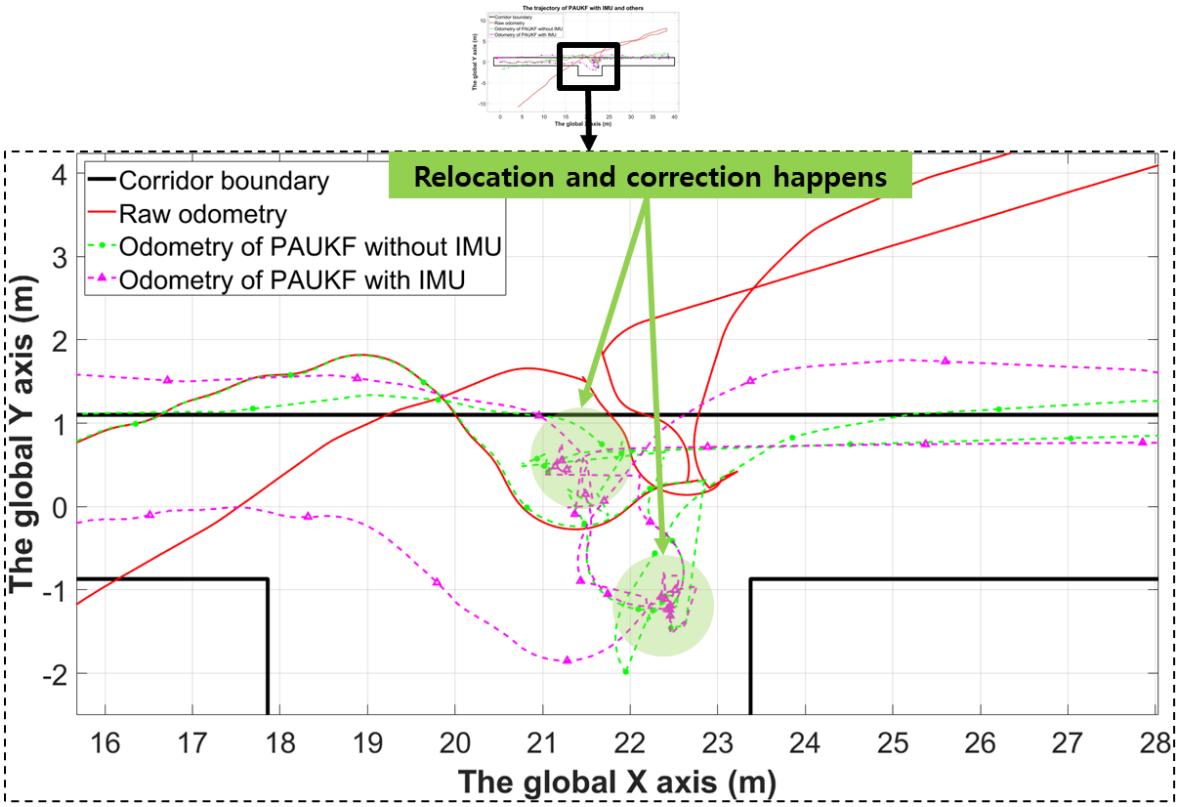


Figure 45 Enlarged trajectories in the center of the corridor

Table 8 shows the error of the different estimation methods in the final position. Compare to the PAUKF without IMU, the longitudinal error changes from 0.552m to the -0.784m, the lateral error changes from -1.643m to 0.906m. The point to point distance of start point and finish point of PAUKF without IMU is 1.733m, and the PAUKF with IMU is 1.435m. The usage of the IMU improves the

precision of the PAUKF by about 17.20%, decreases the point to point error by 0.298m. There is another phenomenon that should be noticed is the usage of the IMU makes the estimated velocity is faster than the UGV's real velocity. As a result, the estimated final position of the PAUKF with IMU behind at the start point. The conclusion that can be found is that adding a sensor could improve the performance of the filter. However, this also adds additional noise to the estimation process.

Table 8 Error of the different estimation method at the final position

Estimation method	Longitudinal error(m)	Lateral error(m)
Raw odometry	3.892	-10.93
PAUKF without IMU	0.552	-1.643
PAUKF with IMU	-0.784	0.906

From the experiment results of this section, it can be found the usage of the IMU can improve the performance of the PAUKF. However, the usage of the IMU still only can provide relative measurement and IMU also contains the noises. Therefore, no matter what kinds of sensors that be used, the PAUKF localization framework can provide the relocation and correction function based on the map.

3.4.2 The results analysis based on scenario2

In the previous section, the trajectory of the UGV is overlapped. Therefore, it is not easy to confirm the relocation and correction phenomenon. In this section, the performance of the representational filters is analyzed based on scenario2 which has a more clear trajectory.

The performance of the raw odometry of the UGV, EKF with IMU sensor, the LeGO-LOAM, and PAUKF(without IMU) are compared. The other filters were explained in previous contents except for LeGO-LOAM. Therefore, it is necessary to explain the LeGO-LOAM algorithm. The full name Lego-LOAM is "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain". The performance is verified in the KITTI dataset. The LeGO-LOAM extracts the odometry position of the sensor by calculating the edge features and planar features. Also, it

transforms the Lidar data into the odom frame. The accumulated points cloud is used to represent the geometry of the corridor. The EKF approach fuses the raw odometry data and IMU, the LeGO-LOAM approach uses the lidar sensor and the PAUKF approach fuses the information of the ar marker based on the depth camera and the raw odometry.

Figure 46 shows the trajectories of different approaches. The black dot is the points transformed to the start position by the LeGO-LOAM algorithm. It can be found a lot of black dots accumulated is the wall of the corridor. The red line is the trajectory from the raw odometry of the UGV. From the end position of the raw odometry, it can be found the position error is not as large as the raw odometry in scenario 1. This is because the UGV moves straightly most of the time and the moving distance is shorter compare to scenario1. The cyan dashed line with the cyan circle is the trajectory of the EKF with the IMU sensor. It can be found the lateral error of the EKF with IMU is decreases compare to the raw odometry. This is because the IMU sensor provides additional yaw rate measurement. The purple dashed line with the purple triangle is the trajectory of the LeGO-LOAM. From the accumulated point cloud, it can be found the trajectory of the LeGO-LOAM is always in the wall which means it works well. The trajectory of the LeGO-LOAM is close to the real trajectory. Since the lateral position can always be extracted by using the planar features, the estimation of the LeGO-LOAM is precisely in the lateral direction. However, the estimation of LeGO-LOAM shows a large error in the longitudinal direction when it comes back to the start position. The green dashed line with the green dot is the trajectory of the PAUKF. In this scenario, the PAUKF only uses ar markers and raw odometry data. The lateral/longitudinal error of the PAUKF is smallest when UGV moves back to the start position intuitionally. The detailed analysis is going to be illustrated in the following manuscripts.

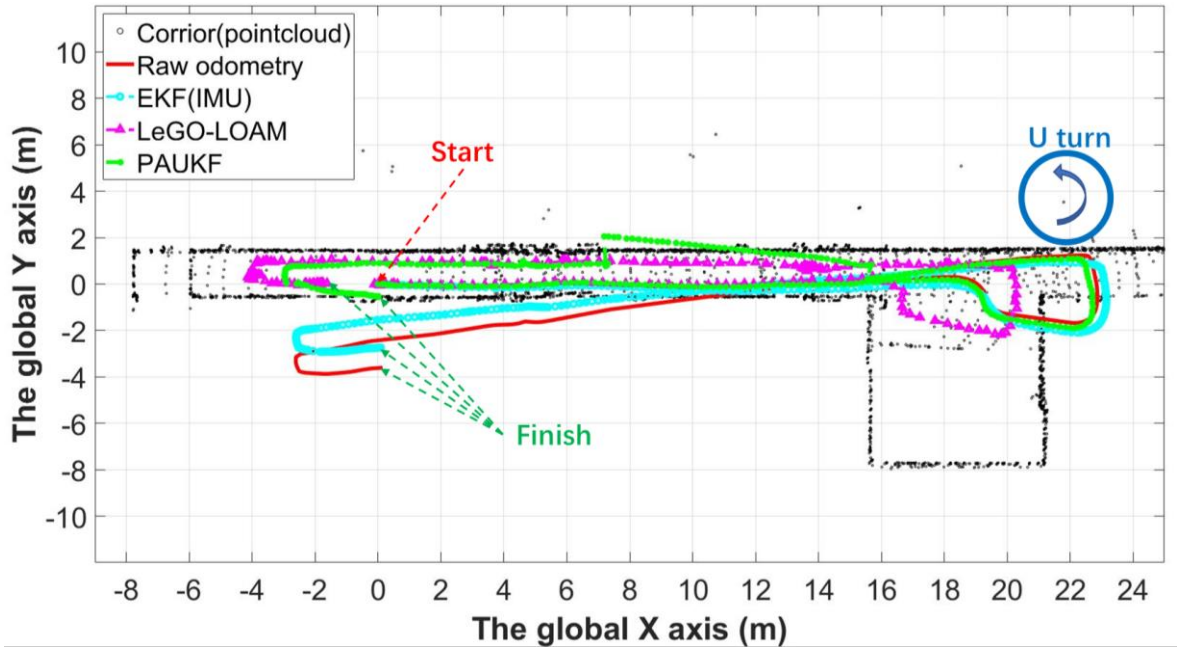


Figure 46 The trajectory of raw odometry, EKF, LeGO-LOAM and PAUKF

Figure 47 is the enlarged figure of the UGV when it moves at the center of the corridor. Since the point cloud is obtained based on the lidar, the accumulated point cloud is considered precise enough to represent the real geometry of the corridor. From the figure, it can be found the trajectory of the LeGO-LOAM is on the left side of the wall. In contrast, the trajectories of raw odometry, EKF with IMU, and PAUKF are on the right side of the wall which means this estimation is erroneous. This phenomenon happens because of the accumulated error in the raw odometry and IMU. The PAUKF in this process did not detect any markers for relocation and correction. Therefore, the raw odometry, EKF with IMU, and PAUKF only can estimate the position by dead reckoning. Since the UGV is controlled with the linear velocity in the x-direction and angular velocity in the z-direction, the position only extracts based on those parameters. Compare to the other estimation, the LeGO-LOAM provides a precise location estimation based on the relative distance to the surrounded planar features(wall). The error of the LeGO-LOAM is significantly smaller compared to the other approaches.

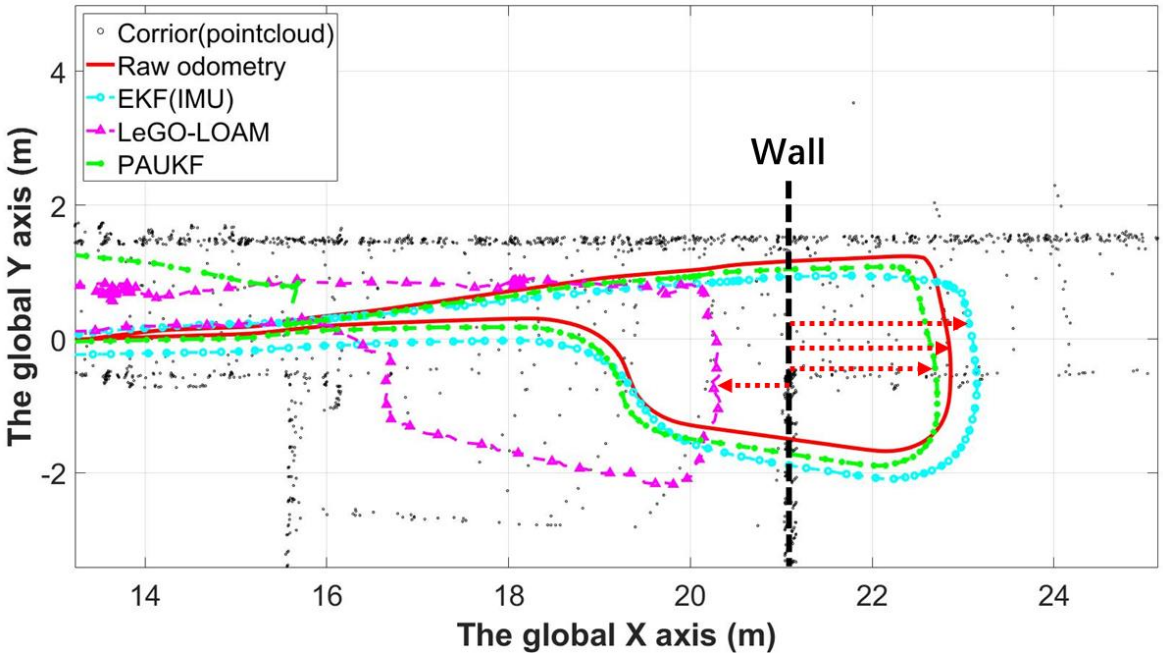


Figure 47 The trajectory of raw odometry, EKF, LeGO-LOAM, and PAUKF at center

Figure 48 is the enlarged figure of the trajectories near the start location where also is the final location. The UGV moves back to the start location. Since the UGV can not move back to the start location perfectly, the offset of the UGV return to the start position is calculated based on the points cloud. The lateral offset of start location and finish position is nearly 0.01m and the lateral offset of start location and finish position is nearly 0.03m. It can be found the offset of the start location and finish location has very small. Thus the effect of the offset can be ignored safely and it doesn't affect the final comparison of the different approaches. From figure 48, it can be found some intuitionally results. Since the UGV moves back to the start position with a very small longitudinal and lateral offset, the error of the final position of the different approaches estimation and the start position can be used as the evaluation parameter. The error of the raw odometry has the biggest error from the start position. The EKF with IMU shows better precision compare to the raw odometry. However, since the IMU also accumulates the velocity error and yaw angle error, the effect of the IMU is limited. Next, the LeGO-LOAM provides very accurate lateral position estimation, but it performs worse in the longitudinal position estimation. The numerical error will be compared in the following manuscript, but it can be found directly the longitudinal error of LeGO-LOAM is bigger than other approaches.

The PAUKF performs best in these approaches. The longitudinal error and lateral error estimated by the PAUKF is the smallest. This is because the PAUKF relocates the vehicle position and corrects the yaw angle based on the surrounded features simulated with ar markers.

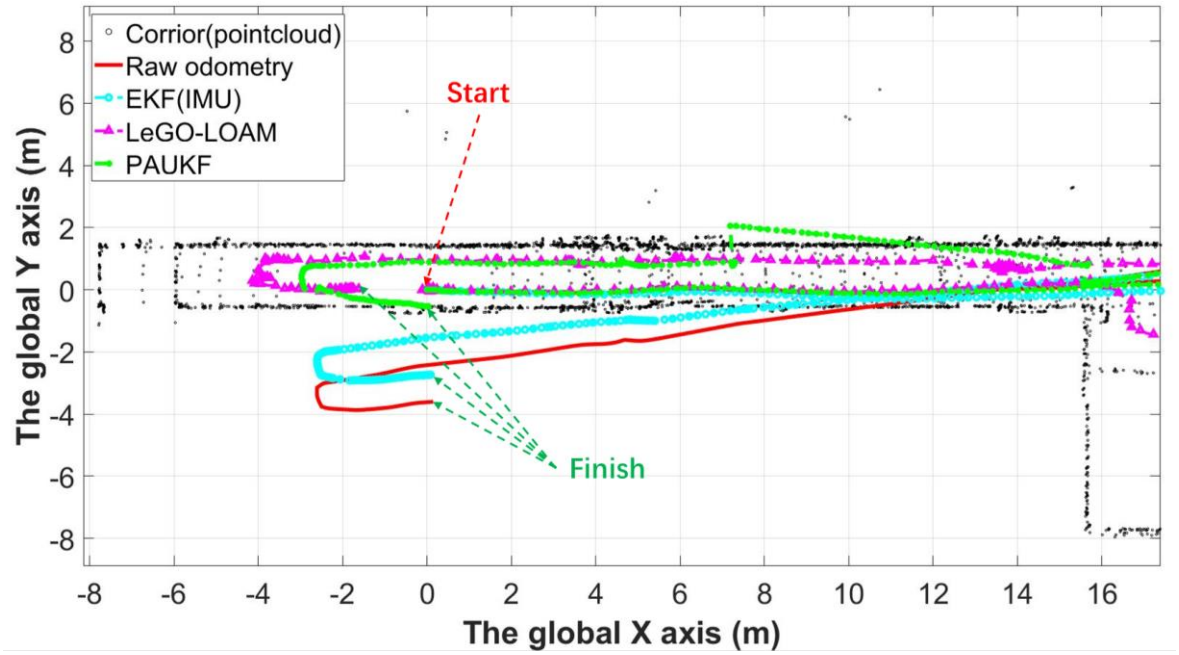


Figure 48 The trajectory of raw odometry, EKF, LeGO-LOAM, and PAUKF at start position

Figure 49 shows the ar markers position in the corridor. The A, B, C, and D with black circles show the position of the ar markers. The red dashed rectangle shows the trajectory of relocation and correction. The UGV detects these ar markers with a depth camera which means the UGV can extract the relative distance between these ar markers. In this figure, we put the focus on the blue dotted line which represents the estimated trajectory of the PAUKF. It can be found the PAUKF relocates the position of the UGV and corrects the yaw angle when the UGV detects the markers A. Then the PAUKF estimates the position based on the relocated position and corrected the yaw angle. However, in this step, because of some reasons, such as perception algorithm error, hardware error, erroneous ground truth position of the ar markers, the corrected yaw angle contains a small error. This error makes the estimated position out of the wall which means it is shifted out the boundary. Then the

UGV detects the markers B and C, in these two points, the PAUKF extracts the correct position and yaw angle. As a result, the position estimation of the PAUKF is almost the same as LeGO-LOAM which uses a Lidar sensor. Then PAUKF continues the dead reckoning based on the raw odometry and detects the markers at the D position. Because of the relocation and correction at the B, C position, the PAUKF contains a little error. Then it relocates and corrects the UGV again at position D. Like in the detection at position A, the PAUKF estimates the yaw angle contains a small error. This small error at the yaw angle increases the error of the PAUKF in the lateral direction. With this experiment, a fact that can be found in the performance of the PAUKF is affected by the number of the features(ar markers) and the precision of the features detection. The more features it measures, the more accurate the PAUKF will be.

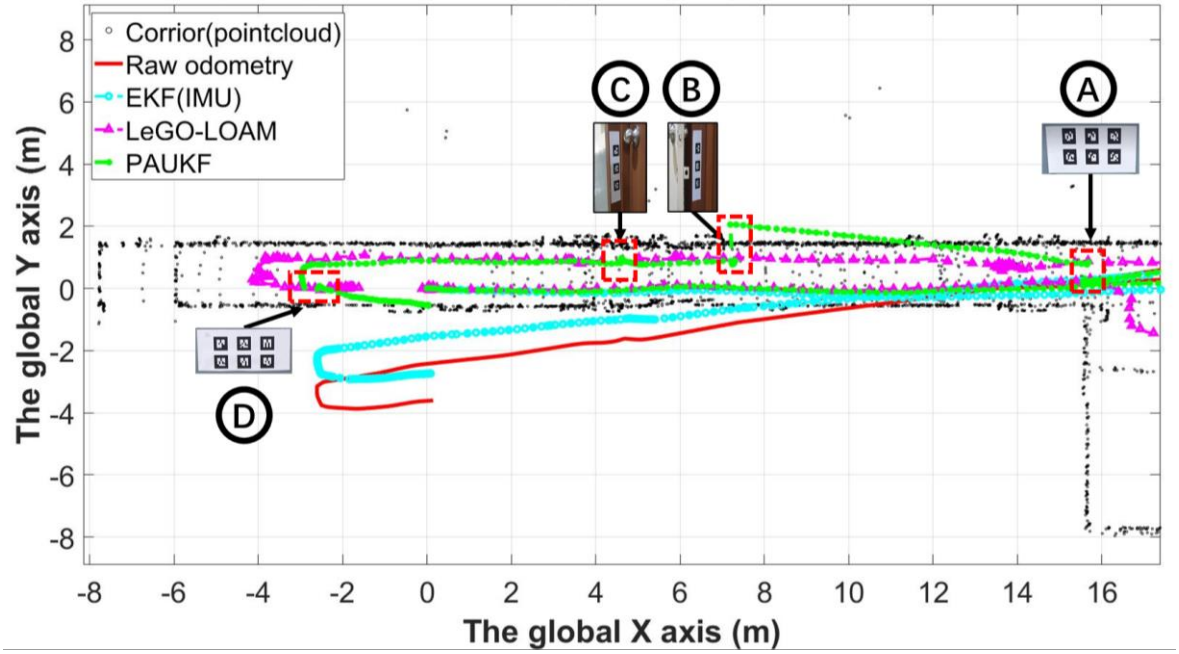


Figure 49 The relocation and correction phenomenon happened at four locations

Table 9 shows the comparison of the longitudinal error and lateral error of different approaches. The data of the error is the differences between the last estimated position and the start position by each approach. The forward-moving direction is the positive direction of x, and the lateral is follows the right-hand coordinate based on the positive x-direction. From Table 9, it can be found directly that

the estimation performance of the PAUKF is the best. The results are analyzed in the longitudinal direction and lateral direction.

The longitudinal error of the raw odometry, EKF with IMU and PAUKF is small compare to the LeGO-LOAM. The estimation of LeGO-LOAM in the longitudinal direction is not good enough. The longitudinal error of LeGO-LOAM is -1.853m. It means the LeGO-LOAM can not extract enough features to relocate the UGV and the loop closure detection is erroneous. Compare to the LeGO-LOAM, other approaches perform well in the longitudinal direction. The longitudinal error of raw odometry, EKF with IMU, PAUKF is 0.1413m, 0.078m, and 0.025m respectively. However, it does not mean the longitudinal estimation of those approaches is precise. From Figure 47, it can be found the longitudinal estimation of those approaches is not as good as LeGO-LOAM. The reason for the raw odometry and EKF with imu that contains small longitudinal is that the error is counteracting in the longitudinal direction. The longitudinal error of the EKF with IMU is smaller than the raw odometry means that the measurement of IMU is helpful to the estimation. Compare to the other approaches, the PAUKF reduces the error in longitudinal based on the features(ar markers). It relocates and corrects the estimation when the UGV detects the surrounding features(ar markers). The IMU is not used in The performance of PAUKF is nearly the same as the EKF with IMU through PAUKF does not use the IMU.

Then the lateral error of different approaches is analyzed. The lateral error is offset of the start position and finish position in y positive direction. From Table 9, it can be found the lateral error of LeGO-LOAM is 0.03m which is the smallest in all approaches. Since the LeGO-LOAM extracts the planar features based on the lidar sensor, it can extract the precise lateral position of the UGV. Compare to the LeGO-LOAM, the raw odometry, EKF with IMU gives a lateral error -3.606m and -2.737m respectively. The lateral error of the raw odometry and EKF with IMU is because of the accumulated error in the yaw angle. Despite the EKF with IMU obtains a relatively small error compared to the raw odometry, it has no way to handle the accumulated error in yaw angle. Compare to the raw odometry and EKF with IMU, the PAUKF gives the lateral error only with -0.546m. The PAUKF does not use the Lidar sensor but uses the surrounded features(ar markers). When PAUKF

detects the ar markers at position D, it relocates the position and corrects the yaw angle. Therefore, the lateral error of PAUKF is smaller than the raw odometry and EKF with IMU.

The integrated error is also calculated. The integral error of raw odometry, EKF with IMU, LeGO-LOAM, and PAUKF is 3.609m, 2.738m, 1.853m, and 0.547m respectively. The PAUKF shows significant precision compare to the other approaches. The advancement of the PAUKF is it can relocate and corrects the pose of the UGV based on matching the position of features(simulated as ar markers) and the ground truth position data. Once the PAUKF receives more features, the precision of the PAUKF should be increased directly. The limitation of the PAUKF is that it can be affected by the error of the detection module. This also can be solved by detecting more features and tuning the covariance of the measurement of the PAUKF.

Table 9 Error of the different estimation method at the final position

Estimation method	Longitudinal error(m)	Lateral error(m)	Integral error(m)
Raw odometry	0.141	-3.606	3.609
EKF with IMU	0.078	-2.737	2.738
LeGO-LOAM	-1.853	0.003	1.853
PAUKF	**0.025**	**-0.546**	**0.547**

3.5 Summary of the experiment results of PAUKF in UGV

As illustrated above, the performance of the particle filter is in positive proportion to the number of the used particles. Thus in fact the performance of the PF is not stable. Because the PF provides the global position and orientation, it makes the estimated position jump around. In the simulation evaluation process, the PF performs relatively smooth compare to the real test. This phenomenon happened because of the number of features. In the real test, the ar markers are only stuck in two areas of the corridor. Thus, PF only works when UGV detects the ar markers. If more markers can be provided, the performance of the PAUKF could be better. Therefore, treat the features detected by the computer vision and lidar-based detection as the references and compare them to the pre-build map is meaningful. Then no matter where the autonomous vehicle is, it can localize itself correctly based on the map and surrounded features. Because perception precision is not the research interest of this

thesis, the ar track alvar package is used for provides features. Thus the number of the ar markers can not compare to the number of the features detected by the perception module as the commercial autonomous vehicles.

From the real test, a fact that can be found is that the PAUKF works and get the best localization accuracy to compare to the other estimation method. After the performance of the PAUKF is evaluated based on the raw odometry and ar markers, the PAUKF is evaluated based on the raw odometry, ar markers, and the IMU sensor. The results of the PAUKF with IMU sensor show the IMU sensors can improve the performance of the PAUKF. Of course, the more sensors that be used, the more accurate the estimation is. However, it should be noticed that the usage of other sensors also includes new noises in the estimation process. Even the IMU provides the additional measurement of the state of the UGV, it still provides the relative measurement. In scenario 2, the PAUKF is compared with EKF with IMU and LeGO-LOAM. The PAUKF does not use an IMU sensor and Lidar sensor. The smallest integrated error of the PAUKF indicates the advancement of the proposed algorithm. The main contribution of the PAUKF is to provide global location information to correct the estimation. The experiment results show the advancement of the PAUKF based localization algorithm in both scenarios.

In chapter 3, the experiment of the PAUKF is introduced with differential wheeled UGV with slow velocity. As illustrated in chapter 2, the PAUKF is a general localization framework that does not depend on the specific hardware. What is more, based on the experiment in scenario1,2, we found the quantity and the detection precision of the features affect the performance of the PAUKF directly. Because of the lack of a detection algorithm, the ar markers are used to simulate the features. However, it is not easy to set enough ar markers for evaluating the performance of the PAUKF. To evaluate the performance of PAUKF on more level, the PAUKF is implemented into a car-like vehicle with high speed and more features around. Because of the limited experiment condition, the evaluation is based on the simulation. The interest of evaluation is not the effect of the vehicle model but the performance of the PAUKF with more features with high speed. Since the ground truth trajectory of the vehicle can be logged in the simulation environment, it is convenient to analyze the

estimation performance of the different approaches.

The simulation in the next chapter is trying to provide a reference of the PAUKF in more features and high-speed conditions with the car-like autonomous vehicle.

Chapter 4 The PAUKF evaluation with car-like autonomous vehicle in simulation

In chapter 3, the PAUKF is implemented into the UGV for evaluation. From the experiment results from scenario1,2, it can be found that the PAUKF corrects the yaw angle of the UGV and make an estimation of UGV relocates to the correct position. The performance is compared based on the relative error of the UGV at the start point and finished point.

However, it is not enough for PAUKF evaluation. The evaluation process needs a ground truth trajectory for comparing the error and it should be proved that the PAUKF based localization method is independent of the specific vehicle. What is more, the features only can be detected 4 times while the whole experiment and the UGV move slowly. The advancement of the PAUKF is to combine the vehicle data and the nearby feature.

However, in the real UGV test, the limited quantity of the features and the distribution of the features are too concentrated. The setting of the scenario makes the PAUKF can not detect enough features for correcting the UGV position.

Despite the estimation of PAUKF is better than the other approaches, it still needs to figure out the real performance of the PAUKF with sufficient features. So, to figure out the performance of the PAUKF with sufficient features with high velocity, the PAUKF is implemented into a car-like autonomous vehicle. This evaluation also proves the PAUKF is a general localization algorithm that independent from the specific target vehicle and evaluates the performance of the PAUKF with ground truth which is known already.

In this chapter, the PAUKF is evaluated based on the simulation with a car-like autonomous vehicle.

4.1 The model of the car-like autonomous vehicle

As illustrated in chapter 2, the model of the PAUKF should be changed based on the target platform. In chapter 3, the UGV model is introduced. Therefore, to evaluate the PAUKF in the car-like autonomous vehicle, the model of the vehicle should be used. The prediction model was constructed

based on the vehicle model. Complex models, such as a dynamic model considering the force of tires, can also be included. However, complex vehicle models reduce computation efficiency. Such models also require detailed vehicle parameters, which are difficult to set. Incorrect parameters can cause noisy estimations. Considering the computational burden and precision, a kinematic model was used in this study. The slip angle is ignored since in the simulation environment, there is no slip.

It should be mentioned that the vehicle dynamic should be improved if the PAUKF is implemented into the real vehicle. However, the goal of this thesis is not to evaluate the effection of the selected model on the PAUKF. The state of each particle should be predicted for obtaining the prior belief at the next timestamp. Equation (2.75) shows the prediction model and the Equation (2.76) shows the movement of the vehicle is modeled as noise since the environment is assumed as planar.

$$for\ every\ particle: \hat{x}_k^- = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \\ \bar{\theta} \end{bmatrix}_k = \begin{bmatrix} \frac{v}{\dot{\theta}}[\sin(\theta + \dot{\theta} \times \Delta t) - \sin(\theta)] \\ \frac{v}{\dot{\theta}}[\cos(\theta) - \cos(\theta + \dot{\theta} \times \Delta t)] \\ z_v \\ \dot{\theta} \times \Delta t \end{bmatrix}_{k-1} + \begin{bmatrix} x \\ y \\ 0 \\ \theta \end{bmatrix}_{k-1} \tag{2.75}$$

$$z_v \sim N(0, \sigma_{v_z}^2) \tag{2.76}$$

When the yaw rate equals zero, Equation (2.75) can be shown to become infinite. A different prediction model should therefore be used when the yaw rate is zero as shown in Equation (2.77).

$$for\ every\ particle: \hat{x}_k^- = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \\ \bar{\theta} \end{bmatrix}_k = \begin{bmatrix} v\cos(\theta)\,\Delta t \\ v\sin(\theta)\,\Delta t \\ z_v \\ 0 \end{bmatrix}_{k-1} + \begin{bmatrix} x \\ y \\ 0 \\ \theta \end{bmatrix}_{k-1} \tag{2.77}$$

The roll, pitch are ignored and the movement in the Z direction is modeled as noise since the vehicle is assumed as moving on a planar in this thesis. Considering the underlying structure of the vehicle for a real-world test, the acceleration and yaw acceleration is considered to be noise. If the

acceleration and yaw acceleration noise effect are additive, then the covariance term can be added directly. However, the acceleration and yaw acceleration noise effects are nonlinear. Therefore, the process noise cannot be handled by addition alone. To handle nonlinear noise, it is considered to be a state, as shown in Equation (2.78). The state of the UKF part in the simulation environment is shown as Equation (2.78). The state considers the nonlinear affection of the noise is called an augmented state.

$$x_{paukf,k,aug} = [x \quad y \quad v \quad \theta \quad \dot{\theta} \quad w_{linacc} \quad w_{anlacc}]^T \tag{2.78}$$

The process noises in linear acceleration in the x-direction w_{linacc} and noises in angular acceleration in z-direction w_{angacc} are set as normal Gaussian distributions with variances of ${\sigma_{linacc}}^2$ and ${\sigma_{angacc}}^2$, respectively, as shown in Equations (2.79) and (2.80).

$$w_{velacc} \sim N(0, {\sigma_{linacc}}^2) \tag{2.79}$$

$$w_{yawacc} \sim N(0, {\sigma_{angacc}}^2) \tag{2.80}$$

The covariance matrix P_k is also augmented into $P_{k,aug}$, which has a size of 7 × 7, as shown in Equation (2.81).

$$P_{k,aug} = \begin{bmatrix} P_k & 0 & 0 \\ 0 & {\sigma_{linacc}}^2 & 0 \\ 0 & 0 & {\sigma_{angacc}}^2 \end{bmatrix} \tag{2.81}$$

The prediction model that considers the augmented noise affection is shown as Equation (2.82). It can be found the noise is affected by the nonlinear function.

$$\widehat{\boldsymbol{X}}_{paukf,k}^{-} = \boldsymbol{X}_{paukf,k-1} + \begin{bmatrix} \frac{v}{\dot{\theta}}\left[\sin\left(\theta_k + \dot{\theta}_k \Delta t\right) - \sin(\theta_k)\right] \\ \frac{v}{\dot{\theta}}\left[\cos(\theta_k) - \cos\left(\theta_k + \dot{\theta}_k \Delta t\right)\right] \\ 0 \\ \dot{\theta}_k \Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \cos(\theta_k) \cdot \sigma_{linacc} \\ \frac{1}{2}\Delta t^2 \sin(\theta_k) \cdot \sigma_{linacc} \\ \Delta t \sigma_{linacc} \\ \frac{1}{2}\Delta t^2 \cdot \sigma_{angacc} \\ \Delta t \cdot \sigma_{angacc} \\ \sigma_{angacc} \\ \sigma_{angacc} \end{bmatrix} \quad (2.82)$$

Then the sigma points generation and the following processes are the same as chapter 2 illustrated.

4.2 PAUKF evaluation based on simulation

The simulation environment is based on the MATLAB autonomous driving toolbox. The whole algorithm is made in MATLAB .m file. The vehicle model in MATLAB toolbox is a simple vehicle model based on the x,y, and yaw angle. The noise of the vehicle, detection, markers is all generated by merging the ground truth data and random noise. The generated noise term and other parameters that are used in each scenario are going to be explained respectively sections.

The simulation results are compared to evaluate the performance of the PAUKF. The evaluation parameter is based on the Root Mean Square Error (RMSE) as Equation (2.83) shows. We choose RMSE as an assessment parameter because the estimation performance of the filter can be compared intuitively by the numerical value of RMSE alone. In Equation (2.83), N indicates the number of data points. The trajectory of the estimated results and the ground truth of the vehicle's trajectory are compared to verify the algorithm. The effect of the yaw angle is considered for both the x and y directions; therefore, there is no additional comparison of the yaw angle. The unit for all position parameters is "meter".

$$\begin{bmatrix} \mathrm{RMSE}_{\mathrm{est}} \\ \mathrm{RMSE}_{\mathrm{noise}} \end{bmatrix} = \begin{bmatrix} \sqrt{\left[\sum_{i=1}^{N}\left(\mathrm{Position}_{\mathrm{est}_i} - \mathrm{Position}_{\mathrm{mean\_est}_i}\right)^2\right]/N} \\ \sqrt{\left[\sum_{i=1}^{N}\left(\mathrm{Position}_{\mathrm{noise}_i} - \mathrm{Position}_{\mathrm{mean\_noise}_i}\right)^2\right]/N} \end{bmatrix} \quad (2.83)$$

4.2.1 Performance of PAUKF with 2D features

First of all, the setting of the environment is illustrated. The simulated geometry of the road is an S curve as shown in Figure 50. The 12 features can be detected all-time in the 2D features environment. The two-dimensional features are fixed near the road in Figure 50. The 2D features mean the vertical position of the features is zero. Not only the features are 2D, but the movement of the vertical movement of the vehicle is also ignored in the 2D features scenario. This means there is no noise effect of the vertical direction.

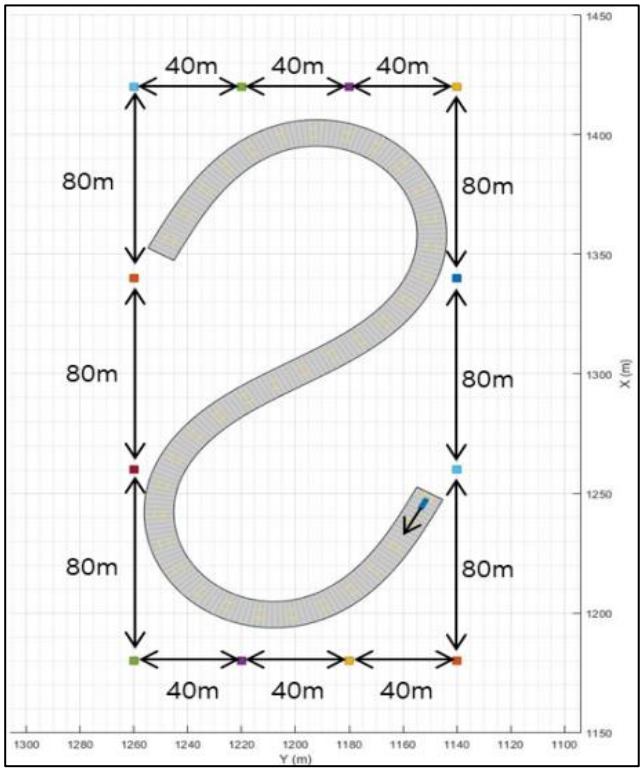


Figure 50 2D features around the road

The noise setting parameters are shown in Table 10. The modeled GPS sensor data is used for the vehicle position initialization and particle initialization. This initial position of the vehicle also can be given with manual input. However, in general, the initial position is measured by the GPS even the signal contains errors. The noise is generated by the normal distribution and transformed with the sinusoidal function for modeling the Non-Gaussian noise according to the previous literature

[104][105]. The literature “Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments” by J. K. Suhr, J. Jang, D. Min, and H. G. Jung is selected for comparison because of the similar approach. J. K. Suhr, J. Jang, D. Min, and H. G. Jung uses particle filter and features of lane line(2D) to relocate and correct the vehicle position. Thus the noise term is generated refer to this literature and compare the result with it.

The onboard noise values are empirical. The perception error is modeled as a normal distribution in x, y-directions. In this thesis, we assumed that the perception was performed by a sensor that can extract the relative distance. These simulation noise parameters are configured with experimental and only provide reference results. Once the PAUKF algorithm is implemented into the real vehicle, then the parameters of the algorithm should be configured according to the appropriate test. The perception error depends on the algorithm of the perception module; therefore, it should also be tuned to the actual sensors. The velocity was assumed to be constant. The test velocities of the vehicle were 60 km/h, 70 km/h, 80 km/h, 90 km/h, 100 km/h, 110 km/h, and 120 km/h.

As illustrated in the previous chapter, the kinematic model is not enough for modeling the motion of the vehicle. The purpose of simulating with different velocities is for evaluating the performance of the PAUKF rather than the effect of the vehicle model. In the real world, the vehicle model should be changed into a dynamic model or fused vehicle model for better results. In this thesis, the goal of the simulation is to evaluate the PAUKF algorithm rather than the model of the vehicle. Thus, the results of the simulation provide the reference performance of the PAUKF. The random seed was fixed to 50 for repeatable simulation. The sample time was set as 0.01s. These parameters provide a reference for the performance of the PAUKF algorithm. These empirical noise parameters should therefore be tuned with the actual sensor characteristics.

Table 10 The parameters of the simulation (2D features)

Parameter Name	Generate Method
vehicle x-axis error(m)(Gaussian)	N(9.65, 12.20)[105]
vehicle y-axis error(m)(Gaussian)	N(9.65, 12.20)[105]
vehicle x-axis error(m)(Non-Gaussian)	~15sin(N(0, 1)) + N(9.65, 12.20) + 5
vehicle y-axis error(m)(Non-Gaussian)	~15sin(N(0, 1)) + N(8.34, 12.33) + 5
Velocity error(m/s)	$\sim\sin(\mathrm{N}(0, 0.09))$
Yaw error(degree)	$\sim\sin(\mathrm{N}(0, 0.09))$
Yaw rate error($\mathrm{degree/s^2}$)	$\sim\sin(\mathrm{N}(0, 0.09))$
Feature x, y error(m)	$\sim\mathrm{N}(0, 0.09)$
Range sensor bearing error(degree)	$\sim\mathrm{N}(0, 0.09)$

Figure 51 shows the trajectory results of the PF, UKF, and PAUKF, and noise in the S-shaped road with 2D features. In the case of the 2D features, the uncertainty of the features in the Z direction is ignored. It means PAUKF processes the features with less noise. As the legend shows, the green line with a green circle is the ground truth trajectory, the dashed line with a red upward-pointing triangle is the noisy vehicle trajectory, the black dashed line with a black square is the PF estimated trajectory, the blue dashed line with a blue square is the UKF estimated trajectory, and the yellow dashed line with the yellow star marker is the PAUKF estimated trajectory.

The data in Figure 51 are generated when the vehicle velocity is 60 km/h, and the noise is Gaussian, as shown in Table 10. The sample time is 0.01s in this simulation. The PF estimated trajectory is near the ground truth trajectory. However, the PF-estimated trajectory is not smooth, and the error is still large. This is because the PF localizes the vehicle position with noisy relative distance to each feature and noisy vehicle data. Since there is no other measurement, it must be considered that the measurement is correct. Compared to that with the PF, the UKF-estimated trajectory is relatively smooth; however, it cannot filter the noise of the GPS data. Because the GPS measurement of the UKF has high variance and the UKF does not use range sensor data, the UKF believes the vehicle model more than the measurement. The noisy measurement also makes the UKF less sensitive to the changes in the position and yaw. Compared to that with the PF and UKF, the trajectory estimated by the PAUKF is more accurate and smoother. As it combines the smoothness of the UKF and the accuracy of the PF, the PAUKF reacts more quickly and precisely when the position and yaw change.

Moreover, the PAUKF does not depend completely on either of the filters, trades off the filters, and generates even better results.

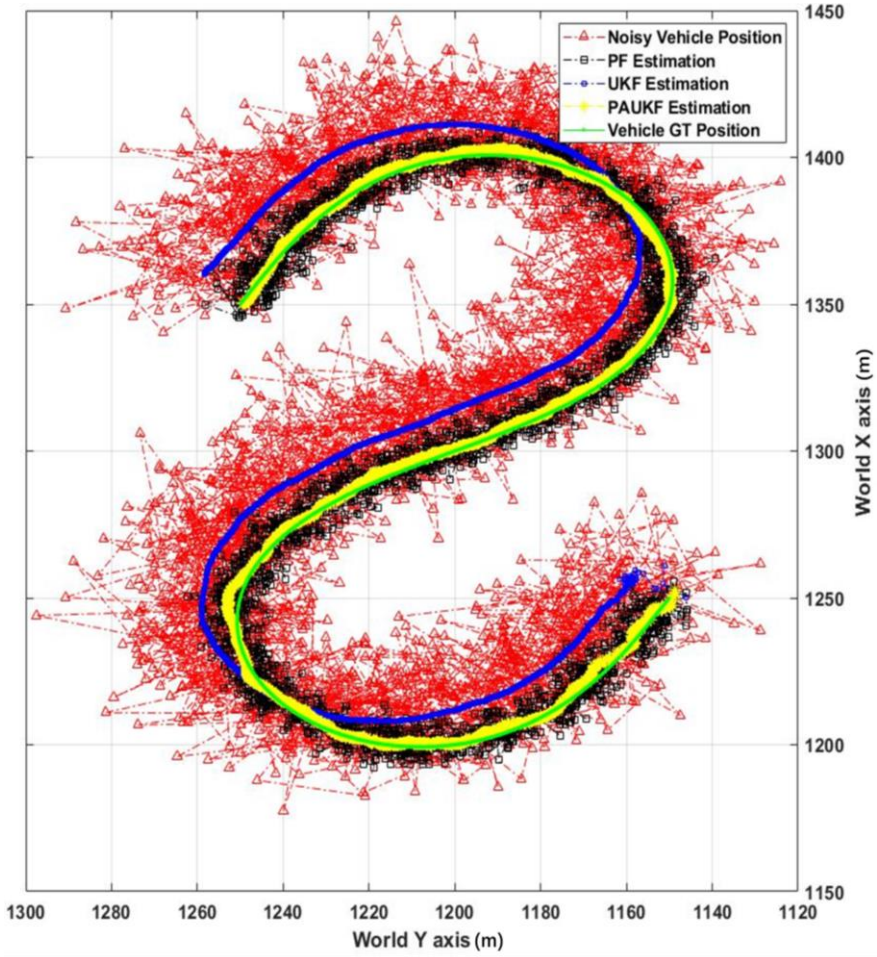


Figure 51 Estimation trajectory of filters with 2D features

The filter performance results are shown in Table 11. Since the UKF does not use range sensor information, it is not appropriate to compare it with the PF and PAUF. Thus, there are no RMSEs for the UKF in Table 11. To determine the performance of the filters in an extreme environment, the algorithm is tested under different velocity and noise environments. As mentioned in Section 3.1, even if the random seed is the same, the random number still changes depending on the number of times it has been called. Therefore, we analyzed the trend of every filter. It can be observed that the PF and PAUKF estimation errors increase slightly when the velocity increases. However, if we consider the

magnitude of the RMSE of the changes in noise from 21.336 to 21.712 m, it can be found that the RMSE of the estimation error does not change even when the velocity increases from 60 to 120 km/h. Compared to the Gaussian noise, the non-Gaussian noise generated a larger mean value. Even so, the precision of the PF does not change even when the noise increases and the precision is almost the same as the RMSE range of 5.489−5.959 m. The PAUKF has an RMSE range of 1.440−1.772 m, even when the noise increases and velocity increases. This is because PAUKF takes the PF estimation results as input and tradeoff the measurement and predicted value from the UKF. The trade-off is done using the cross-correlation function in Equation (2.24). Therefore, the PAUKF combines the recursiveness of the UKF and the location information of the features based on the PF. The PAUKF improves the accuracy by 4.028–4.049 m compared to the PF.

Table 11 Total RMSE of filters in different conditions

	With Gaussian Noise			With Non-Gaussian Noise		
Velocity	**Noise(m)**	**PF(m)**	**PAUKF(m)**	**Noise(m)**	**PF(m)**	**PAUKF(m)**
60 km/h	21.336	5.634	1.451	29.796	5.959	1.655
70 km/h	21.339	5.747	1.624	29.912	5.599	1.409
80 km/h	21.310	5.600	1.651	29.730	5.579	1.440
90 km/h	21.510	5.594	1.742	29.661	5.652	1.423
100 km/h	21.154	5.720	1.501	29.430	5.631	1.616
110 km/h	21.530	5.626	1.625	29.546	5.544	1.485
120 km/h	21.712	5.800	1.772	29.934	5.489	1.454
Mean	**21.413**	**5.674**	**1.624**	**29.716**	**5.636**	**1.497**

To figure out the performance of the filters more clearly, the RMSE and mean value are calculated in the longitudinal direction and lateral direction respectively. The Table 12 is the RMSE and Mean of the PAUKF estimation in the longitudinal direction and lateral direction. The parameters are also divided into two groups based on the character of the noise. It should be emphasized again that the amplitude of Gaussian noise and Non-Gaussian noise is different. Compare to the Gaussian noise, the Non-Gaussian noise larger as Table 10 shows. The average RMSE of PAUKF estimation in the longitudinal direction is 1.097m and the average mean is 0.493m, and the average RMSE of PAUKF estimation in the lateral direction is 1.196m and the average mean is -0.153m with Gaussian

noise. The average RMSE of PAUKF estimation in the longitudinal direction is 0.946m and the average mean is 0.304m, and the average RMSE of PAUKF estimation in the lateral direction is 1.155m and the average mean is -0.061m with Non-Gaussian noise.

Table 12 The RMSE(Mean) of PAUKF estimation in longitudinal/lateral direction

	With Gaussian Noise		With Non-Gaussian Noise	
Velocity	**Longitudinal(m)**	**Lateral(m)**	**Longitudinal(m)**	**Lateral(m)**
60 km/h	0.990(0.526)	1.060(-0.111)	1.113(0.275)	1.225(0.132)
70 km/h	1.089(0.333)	1.206(-0.068)	0.954(0.270)	1.037(-0.165)
80 km/h	1.139(0.548)	1.194(-0.169)	0.970(0.565)	1.064(0.041)
90 km/h	1.269(0.509)	1.193(-0.310)	0.797(0.187)	1.179(-0.328)
100 km/h	0.986(0.571)	1.132(0.123)	1.123(0.305)	1.162(-0.184)
110 km/h	1.058(0.308)	1.233(-0.181)	0.789(0.352)	1.258(0.266)
120 km/h	1.145(0.659)	1.353(-0.352)	0.873(0.179)	1.163(-0.190)
Mean	**1.097(0.493)**	**1.196(-0.153)**	**0.946(0.304)**	**1.155(-0.061)**

To compare the result, the literature “Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments” by J. K. Suhr, J. Jang, D. Min, and H. G. Jung is selected for comparison as illustrated. The maximum velocity of the vehicle is 60km/h, therefore, the estimation of PAUKF in 60km/h is selected for comparison. The Gaussian noise term is generated based on the parameters described in the literature at GANGNAM. Therefore the term of noise could be considered as similar between the simulation environment of the PAUKF and the literature.

Table 13 shows the mean and variance of the estimation of the target literature and proposed PAUKF. The velocity of each approach is 60km/h and the features are all 2 dimensional. The integral mean error and the variance of the error of the target literature are 1.69m and 1.63m respectively. The integral mean error and the variance of the error of the target literature are 1.08m and 0.71m respectively. By comparing the mean and the variance of the two approaches, it can be found the PAUKF improves the precision by 36% in the mean dimension and improves the precision by 56% in the variance dimension. Compare to the target literature, PAUKF not only estimates the surrounded features by using the particle filter as literature, and it also uses UKF for backend filtering. The usage of the UKF behind the PF makes the PAUKF compensating for the jump phenomenon of the

estimation from PF. Therefore, the variance of the PAUKF is decreased compared to the target literature. This comparison data shows the advantage of the PAUKF.

Table 13 The comparison of PAUKF and the literature with similar noise

	The error of the mean(m)	**The error of the variance(m)**
Estimation of literature[105]	1.69	1.63
Estimation of PAUKF	1.08	0.71
Improvement	**36%**	**56%**

The comparison result in Table 13 shows the advancement of the PAUKF. The PAUKF shows better performance compare to the previous research in the Gaussian noise environment.

As Table 10 shows the environment of Non-Gaussian contains a larger amplitude of the noise. This is because we want to find out the performance of the PAUKF in an extremely noisy environment. The extremely noisy signal makes it is hard to find the correct location. Therefore, the error of the PAUKF shows a relatively big error. The extreme noisy position and the estimation of the PAUKF will be shown in the following sections. Since the advancement of the PAUKF is proved in the above section, in the following sections, the estimation data of the PAUKF are all obtained in the Non-Gaussian noise environment. This is for figuring out the performance of the PAUKF in the large Non-Gaussian error environment.

4.2.2 Performance of PAUKF with 3D features

The geometry of the perception data is critical to the precision of the localization. Once the geometry of the perception data is not processed appropriately, the estimation result becomes even worse. Table 14 shows the configuration parameters of the simulation(3D features). Compare to the 2D features, the noise of movement of the vehicle, and the noise of features in the vertical direction are added.

Table 14 The parameters of the simulation (3D features)

Parameter Name	Generate Method
vehicle x-axis error(m)(Gaussian)	N(9.65, 12.20)[105]
vehicle y-axis error(m)(Gaussian)	N(9.65, 12.20)[105]
vehicle x-axis error(m)(Non-Gaussian)	~15sin(N(0, 1)) + N(9.65, 12.20) + 5
vehicle y-axis error(m)(Non-Gaussian)	~15sin(N(0, 1)) + N(8.34, 12.33) + 5
vehicle z-axis error(m)	$\sim N(0, 0.09)$
Velocity error(m/s)	$\sim \sin(N(0, 0.09))$
Yaw error(degree)	$\sim \sin(N(0, 0.09))$
Yaw rate error($degree/s^2$)	$\sim \sin(N(0, 0.09))$
Feature x, y, z error(m)	$\sim N(0, 0.09)$
Feature z position(m)	Random(0,10)
Range sensor bearing error(degree)	$\sim N(0, 0.09)$
Range sensor elevation error(degree)	$\sim N(0, 0.09)$

The three-dimensional features are generated randomly along the road and the S shape road with the 50 m range of features perception limit as Figure 52 shows. For the case of the 3D features, the vehicle always can detect 10+ features in the perception range.

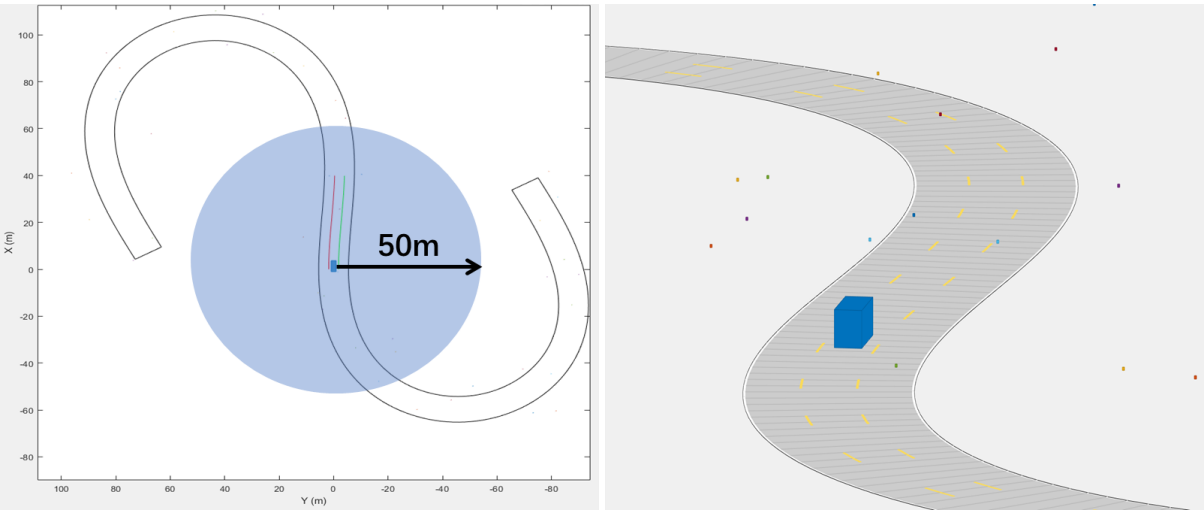


Figure 52 The perception range and the randomly generated 3D features

Figure 53 shows the position estimation result of the PAUKF does not consider the geometry of the perception data at 60 km/h, and the position estimation result of the PAUKF considers the geometry of the perception data at 60 km/h, respectively. The sample time here is 0.05s. In both

figures, the blue line with the circles is the ground truth trajectory of the vehicle, the red line with the triangles is the noisy vehicle position, the yellow line with the square is the particle filter's estimation result, and the black line with the circle is the final estimated vehicle position. The estimated trajectory of the PAUKF does not consider the geometry of the perception data showed a significant error compare to the ground truth trajectory, as shown in Figure 53 (a). This happens because the algorithm did not consider the geometry effect of perception. As a result, it cannot generate and select appropriately weighted particles. The PAUKF by contrast achieved better performance, by not only improving the calculation of the geometry but also by considering the vertical noise effect on the weight generation and the selection scheme. The trajectory of the PAUKF was very close to the ground truth data, as shown in Figure 53 (b). The data is obtained under Non-Gaussian error.

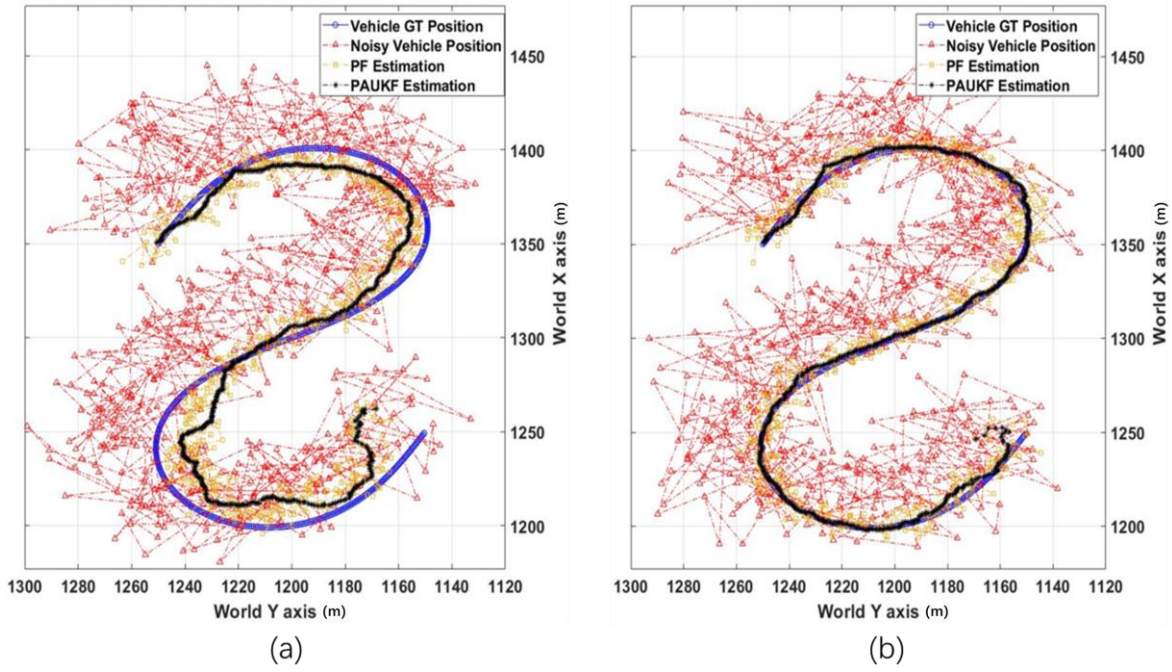
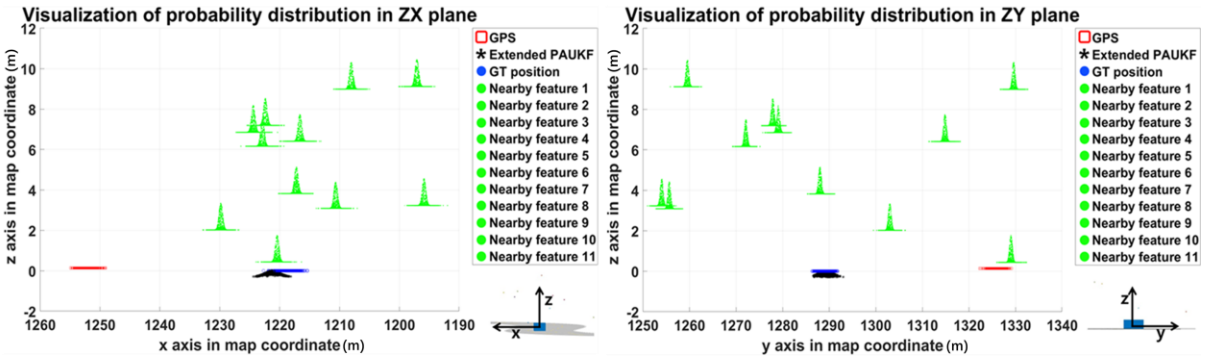


Figure 53 The comparison of the PAUKF considers the geometry or not

Figure 54 shows the visualization result of the probability distribution of the features' position, the ground truth position, the position from the GPS, and the estimated position of PAUKF considering the geometry of the perception data at 250th sample time. To make the figure easy to understand, the origin coordinate of probability distributions translates into the position of features,

GPS, and vehicle. One thousand random numbers were used for visualizing the probability of each parameter. In Figure 54, the red square value is the position from the GPS, the black color with the start marker is the position that is estimated from the PAUKF considering the geometry of the perception data, and the blue data with the square is the ground truth data of the vehicle. Figure 54 shows that all the features have different vertical values. The position from the GPS has a large error compare to the ground truth data. The estimated position from the PAUKF considering the geometry of the perception data is closest to the ground truth data. The PAUKF fused the surrounding feature position information to localize the vehicle itself in the map coordinates. This shows the effectiveness of the PAUKF and the importance of considering the geometry of the perception data.



(a) (b)

Figure 54 The probability distribution at 250th sample time

Table 15 shows the performance of PAUKF when it considers 2D and 3D features geometry effection at a different velocity condition. The data in column "2D" means the PAUKF does not consider the geometry of the perception data in the 3D features environment, and the data in column "3D" means the PAUKF considers the geometry of the perception data in the 3D features environment. The velocity of the vehicle was set at intervals of 10, from 60 km/h to 120 km/h. The performance of the filter changed slightly because of random environmental effects. The RMSE change of noise shows that the vehicle position noise was almost the same at all velocity conditions. This suggests that both filters were in similar environments. From the RMSE of PF, it can be concluded that the previous basic particle filter estimation's mean RMSE is 11.002 m and the particle filter estimation's mean

RMSE is 6.201 m. The estimation RMSE of PF decreased by about 43.64% compared to the previous PF which means the precision of the PF increases by 43.64%. The final estimation result is shown in the right column of the PAUKF in Table 15. The mean of RMSE of the PAUKF that does not consider perception effection of geometry was 10.295 m and that of the PAUKF that considers perception effection of geometry was 2.696 m. The PAUKF decreased the RMSE by about 73.81% by considering the effection of perception geometry. It means, the PAUKF increases the precision by about 73.81% by considering the perception effection of geometry. The PAUKF considered the geometry of perception and improved the weight generation method and selection method. The main improvement happens at the particle filter based pre-processing step. By considering the perception effection of geometry, the PAUKF may therefore estimate the trajectory precisely.

Table 15 The RMSE of the PAUKF depends on the features

	Noisy Position of Vehicle		**PF Estimation**		**PAUKF Estimation**	
Velocity	**2D(m)**	**3D(m)**	**2D(m)**	**3D(m)**	**2D(m)**	**3D(m)**
60 km/h	30.426	29.465	11.057	6.317	10.199	2.478
70 km/h	30.176	29.351	10.973	6.238	10.314	1.767
80 km/h	29.883	28.319	10.989	6.265	10.458	2.303
90 km/h	29.025	29.919	11.052	5.861	10.627	2.169
100 km/h	29.349	29.981	11.094	6.324	10.893	2.833
110 km/h	28.365	29.144	10.851	6.303	9.751	3.441
120 km/h	29.932	30.072	10.999	6.098	9.826	3.881
Mean	**29.571**	**29.465**	**11.002**	**6.201**	**10.295**	**2.696**
RMSE Change	**-0.36%**		**-43.64%**		**-73.81%**	

Table 16 shows the RMSE and Mean of the PAUKF considering the geometry of the perception in the longitudinal direction and lateral direction. The average RMSE of PAUKF estimation in the longitudinal direction is 1.800m and the average mean is 0.259m, and the average RMSE of PAUKF estimation in the lateral direction is 1.993m and the average mean is -0.516m. The estimated error in longitudinal and lateral is nearly 2m which looks like not good enough. However, as illustrated in the previous section, the large estimation error is because we want to figure out the PAUKF in the extreme Non-Gaussian noise environment. The mean RMSE of the noise is 29.571m which is larger

than the usual noise(nearly 10m). The advancement of the PAUKF is already shown in the previous section by comparison to the other literature with the same noise condition. Therefore, the large error of the PAUKF in the longitudinal and lateral directions is meaningful. This is to provide a reference to the PAUKF in the extremely noisy environment.

Table 16 The RMSE(Mean) of PAUKF estimation in longitudinal/lateral direction

Velocity	Longitudinal(m)	Lateral(m)
60 km/h	1.467(0.287)	1.997(-0.107)
70 km/h	1.088(0.397)	1.393(-0.435)
80 km/h	1.724(0.722)	1.527(-0.103)
90 km/h	1.562(0.490)	1.506(-0.597)
100 km/h	2.098(-0.012)	1.904(-0.456)
110 km/h	1.970(0.151)	2.822(-1.218)
120 km/h	2.688(-0.224)	2.799(-0.696)
Mean	**1.800(0.259)**	**1.993(-0.516)**

To figure out the performance of the PAUKF in a huge perception error environment, we changed the perception noise from $N(0, 0.09)$ to $N(3, 9)$ in the x, y, z-directions, meaning that the minimum error of relative distance of every feature and vehicle was always larger than 5.196 m. In this noisy environment, with a velocity of 60 km/h, the RMSE of the total RMSE of the PAUKF was 5.771 m. Since the measurement variances were not changed, the PAUKF still attempted to rely on the measurement from the PF. This simulation provides a reference performance of the PAUKF in a huge perceived noise environment. The algorithm should perform better if we tune the variance of the measurement matrix.

For PAUKF, selects an appropriate covariance parameter is important. The performance of the PAUKF changes according to the selection of the covariance matrix. Especially, in the PAUKF localization framework, the measurement covariance is not easy to decide. It is hard to decide the uncertainty of the PF based on the map matching. However, tuning the measurement covariances is time-consuming work and the results are usually not optimal. Therefore, to obtains better performance of the PAUKF, the measurement covariance matrix is trained with the coordinate descent algorithm.

Figure 55 shows the estimation results of the different approaches where the PAUKF runs with the well-trained measurement covariance matrix.

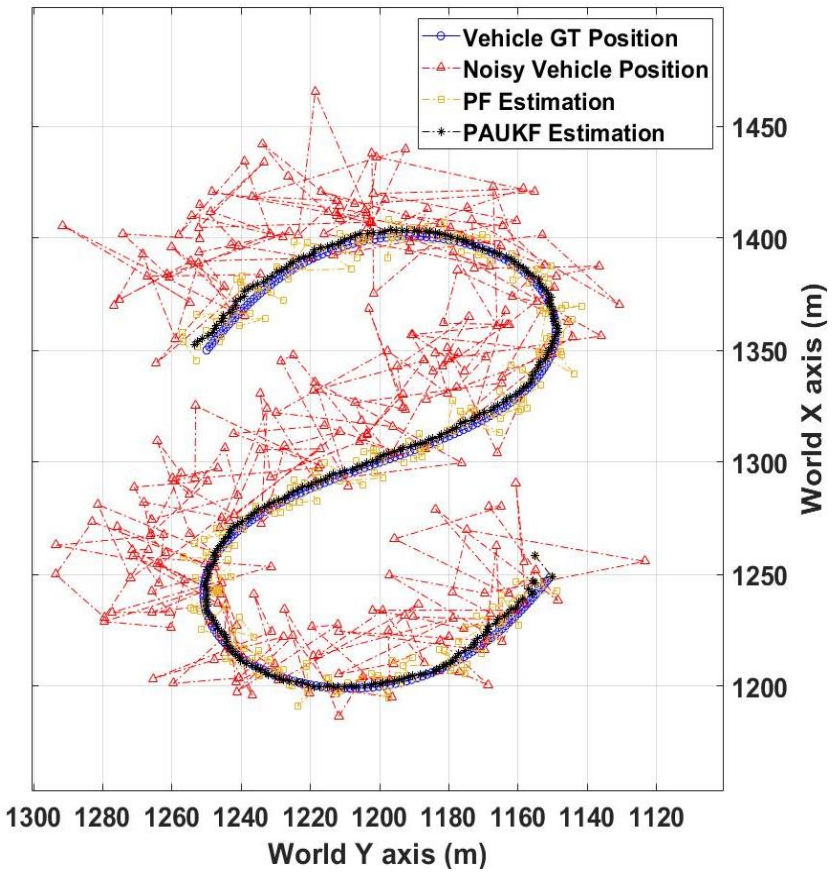


Figure 55 The estimation of PAUKF with trained covariance

Table 17 shows the performance of the PAUKF using the trained measurement covariance matrix. The RMSE change parameter of "Noisy Position of Vehicle" and "PF Estimation" equals to zero means that the noise term and the precision of the PF based pre-processing are not changed at all. The only difference is the selection of the measurement covariance matrix of the PAUKF. From Table 17, it can be found the RMSE of the PAUKF with trained measurement covariance decreases 0.423m. It means compared to the manual tuned PAUKF, the PAUKF with well-trained measurement covariance increases the precision by about 15.7% without adding any computational burden.

Table 17 RMSE of the manual tuned and covariance trained PAUKF

	Noisy Position of Vehicle		PF Estimation		PAUKF Estimation	
Velocity	**Manual(m)**	**Trained(m)**	**Manual(m)**	**Trained(m)**	**Manual(m)**	**Trained(m)**
60 km/h	29.465	29.465	6.317	6.317	2.478	2.651
70 km/h	29.351	29.351	6.238	6.238	1.767	1.510
80 km/h	28.319	28.319	6.265	6.265	2.303	2.645
90 km/h	29.919	29.919	5.861	5.861	2.169	2.270
100 km/h	29.981	29.981	6.324	6.324	2.833	2.166
110 km/h	29.144	29.144	6.303	6.303	3.441	2.574
120 km/h	30.072	30.072	6.098	6.098	3.881	2.093
Mean	**29.465**	**29.465**	**6.201**	**6.201**	**2.696**	**2.273**
RMSE Change	**0%**		**0%**		**-15.70%**	

Table 18 shows the RMSE and Mean of the PAUKF with well-trained measurement covariances in the longitudinal direction and lateral direction. The average RMSE of PAUKF estimation in the longitudinal direction is 1.363m and the average mean is 0.348m, and the average RMSE of PAUKF estimation in the lateral direction is 1.797m and the average mean is -0.228m. It can be found the error in the longitudinal and lateral direction are decreased by well-trained measurement covariances. This phenomenon shows the selection of the measurement covariances of the PAUKF should be well configured for better estimation performance. The selection of the measurement covariances is affected by the precision of the PF based pre-processing step.

Table 18 The RMSE(Mean) of PAUKF estimation in the longitudinal/lateral direction

With Non-Gaussian Noise		
Velocity	**Longitudinal(m)**	**Lateral(m)**
60 km/h	1.402(0.222)	2.250(-0.066)
70 km/h	0.922(0.455)	1.196(-0.280)
80 km/h	1.873(0.553)	1.868(-0.047)
90 km/h	1.507(0.894)	1.698(-0.616)
100 km/h	1.556(-0.170)	1.507(-0.172)
110 km/h	1.102(0.310)	2.326(-0.337)
120 km/h	1.177(0.175)	1.731(-0.080)
Mean	**1.363(0.348)**	**1.797(-0.228)**

Figure 56 shows the longitudinal error of the noisy GPS and the estimation of the PAUKF when the velocity is 60km/h. From Figure 56, it can be found the amplitude of the GPS error in the longitudinal direction is from -39m up to 60m and the RMSE of the GPS error in the longitudinal direction is 18.768m. Under the large noisy environment, the estimation RMSE of the PAUKF is only 1.402m.

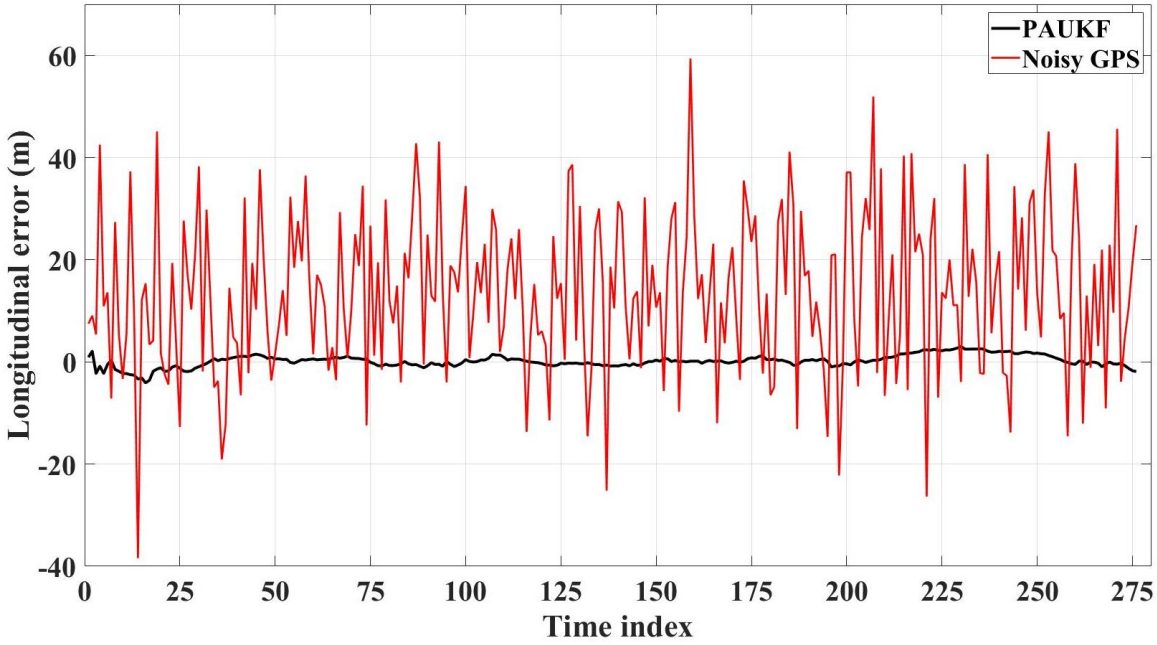


Figure 56 Longitudinal error of PAUKF and noisy GPS

Figure 57 shows the lateral error of the noisy GPS and the estimation of the PAUKF when the velocity is 60km/h. From Figure 57, it can be found the amplitude of the GPS error in the lateral direction is from -30m up to 68m and the RMSE of the GPS error in the lateral direction is 22.715m. Under the large noisy environment, the estimation RMSE of the PAUKF is only 1.797m.

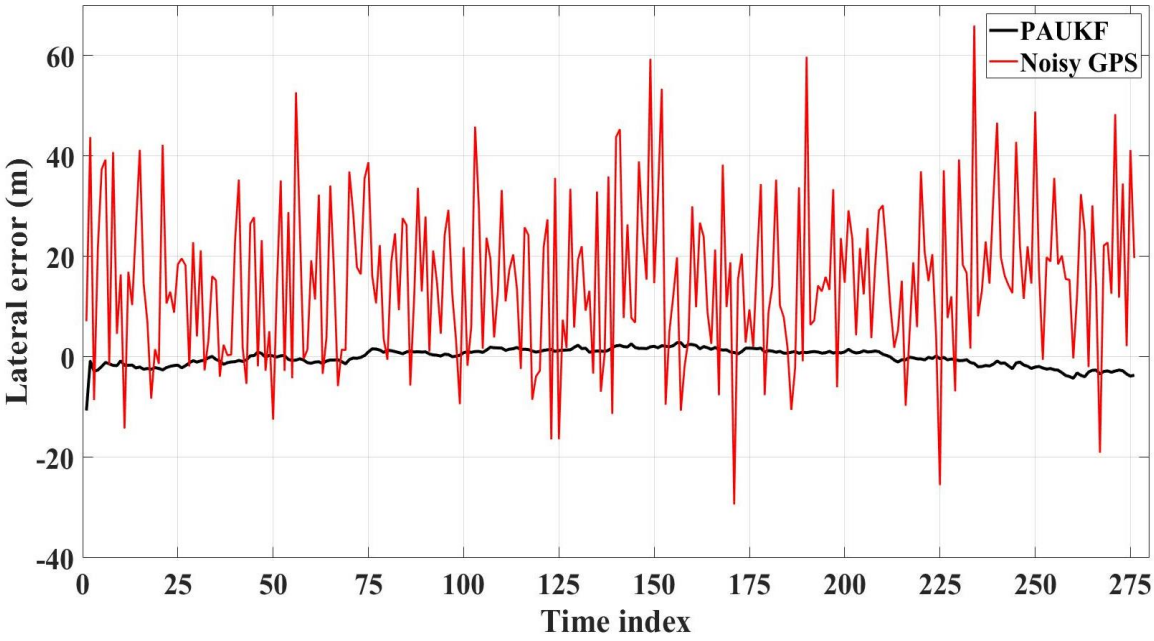


Figure 57 Lateral error of PAUKF and noisy GPS

Figure 58 and 59 shows the trajectories of the different kinds of sources at the start point of the S curve. Figure 58 shows the trajectories of the vehicle with 60km/h and figure 59 shows the trajectories of the vehicle with 120km/h. The blue line with the blue dot is the ground truth data of the vehicle from the simulation. The red dashed line with the red triangle is the position measurement data from the noisy GPS. The yellow dashed line with the yellow square is the estimated position data from the particle filter. The black dashed line with the black dot is the estimated position data of the PAUKF. The PAUKF used in the simulation is the same except for the velocity.

The movement of the estimation data is affected by the noisy vehicle information like velocity and yaw angle. From Figure 58 and Figure 59, it can found the interval of the ground truth position is different. The interval of the position becomes larger as the velocity grows. As intuition, the estimation error should increase as the velocity increase. The prediction step could generate more errors because of the velocity. If the vehicle only uses the information from the on-vehicle sensor, the error could be increase as the velocity increases. However, in the PAUKF localization framework, the vehicle corrects the position and the yaw angle based on the surrounded features. It means, at every timestamp the vehicle predicts the state at the next timestamp. If the nearby features are not used for

correction, the error should increase as the velocity increases. Therefore, as long as the features exist near the vehicle, the PAUKF based localization could guarantee precision that is not affected by the velocity.

Figure 58 and 59 also shows the importance of the initial estimation. Figure 58 shows the PAUKF started from an initial position which contains a large error. The result of the bad initial position makes the PAUKF converges slowly to the ground truth trajectory. Figure 59 shows the PAUKF started from an initial position close to the ground truth position. Therefore, the PAUKF converges to the ground truth trajectory faster than the PAUKF in Figure 58. This explains why the results of the PAUKF in the previous tables are different. The convergence can be faster if the PAUKF run with appropriate covariances. Once the PAUKF uses worse covariance, then even the convergence speed can be accelerated, however, the total precision of the estimation could decrease.

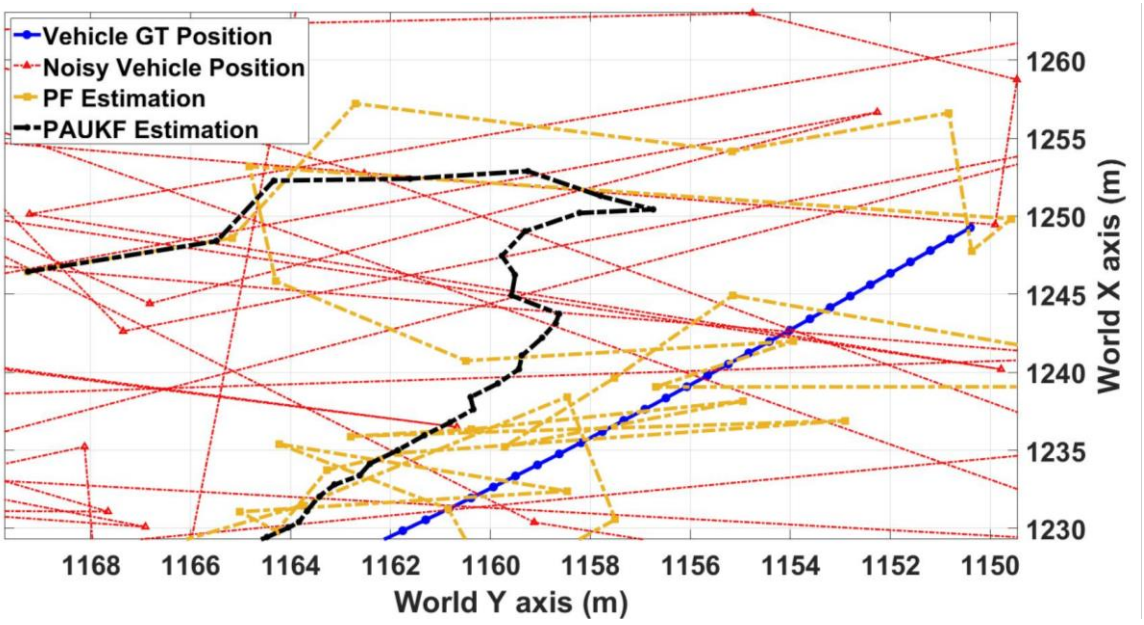


Figure 58 The trajectories at the start point

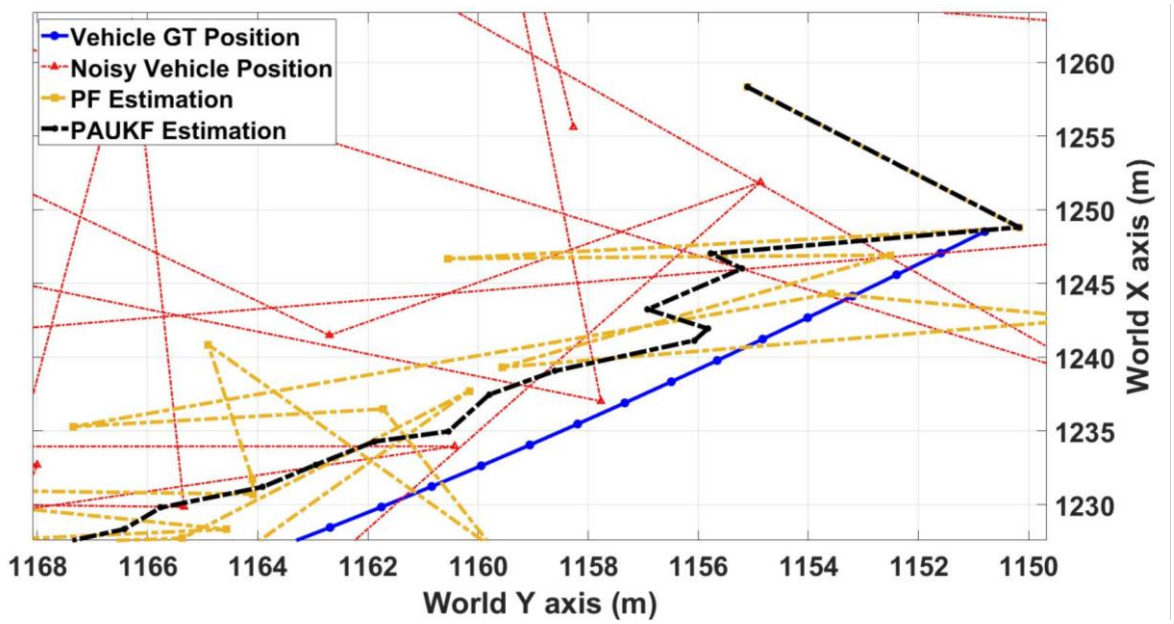


Figure 59 The trajectories at the start point

4.3 Summary of the simulation results of PAUKF in car-like vehicle

In this chapter, the PAUKF is implemented into the car-like vehicle. Because of the limited experiment environment and the hardware, the evaluation process of the PAUKF is done based on the simulation with the Matlab toolbox. The ultimate goal of the PAUKF simulation is to figure out the performance of the PAUKF with more features surrounded and faster movement. As illustrated in chapter 3 and 4, the PAUKF is a general framework of vehicle localization that does not depend on the specific target vehicle. The vehicle model used in the simulation is a simplified model which is called a bicycle model. The bicycle model is not enough for representing the real world dynamic in the high speed. However, the goal of the simulation is not to figure out the differences between the vehicle model and the dynamics of the vehicle in the real world. The goal of the simulation in chapter 4 is to try to provide a reference of the PAUKF with more features and speed. If someone wants to implement the PAUKF into a real car-like vehicle, there are a lot of parameters that should be defined far more than the simulation. From the simulation result, it can be found the PAUKF works precisely in the Gaussian and Non-Gaussian noise environment.

To compare with other literature, we calculate the mean value of estimation. The mean estimation error for the PAUKF is 1.08 m and the variance is 0.7147 m, which is more precise than

the mean of 1.69 m and variance of 1.63 m obtained by GANGNAM for similar noise[105]. To figure out the performance of the PAUKF in the extremely noisy environment, the large Non-Gaussian noise is added based on the simulation. The simulation shows the room for improvement of the PAUKF with preciser perception and the optimized covariances. The effect of the initial position on the PAUKF is analyzed for explaining the randomness of the results. The evaluation of the PAUKF with the car-like platform is done in simulation, therefore the hardware specification is not considered. As a result, the PAUKF performs a character that is not affected by the velocity. It should be mentioned, once the PAUKF is implemented into the real vehicle, the precision of the PAUKF should be affected by the detection rate of the hardware. In summary, the precision of the PAUKF is strongly affected by the precision of the perception module and the quantity of the features near the vehicle. The simulation results show the performance of the PAUKF.

Chapter 5 Conclusion

The autonomous vehicle is an important and challenging issue of the transportation system. With the development of computing capability and sensing technology, the autonomous vehicle like UGV and the car-like autonomous vehicle starts to appear in our life.

The brain of the autonomous vehicle, the path planning module, and the decision-making module work depend on the perception data for path planning and control of the vehicle. It means the perception data affects the output of the planning module directly. Because of the direct effect of the perception module, it is important to filter the perception data robustly and precisely. Almost every autonomous vehicle is equipped with several cameras, lidars, radars, GPS receivers, and other sensors for better perception performance. The location of the vehicle is one of the most important perception data. Despite the autonomous vehicle equipped with lots of sensors, locating the position of the vehicle is still a challenging issue. Because not only every single sensor has its sensing limitations but also the sensed data contains a lot of different kinds of noise. Therefore, a robust and precise localization algorithm is needed.

In this thesis, the particle aided unscented Kalman filter(PAUKF) based localization algorithm is proposed. The algorithm takes advantage of the concept of particle filter and the framework of the unscented Kalman filter for fusing the different kinds of data. Since the noise parameters in the real world are not always Gaussian form, the algorithm should process the Non-Gaussian noise effectively too. The PAUKF not only fusing the noise effectively but also easy to combine vehicle model, motion model. The performance of the algorithm is verified based on the unmanned ground vehicle with real hardware(odometry, RGB-D camera, and IMU) and the car-like vehicle in the simulation environment. From the UGV and the car-like vehicle simulation results, it can be found that the estimated localization of the PAUKF is effective and precise. The algorithm used in the real-world test is based on the robot operating system which means the PAUKF is easy to implemented and used by other researchers. The ground truth data of the specific features are transformed into the map coordinate based on the SLAM algorithm by using the lidar sensor. The precision of the ground truth data of the features affects the estimation precision of the PAUKF. The open-sourced object detection package ar

track alvar(ar markers) is used as the features detection. In the experiment of the PAUKF, the estimated trajectories are generated based on the raw odometry and the ar markers. In the experiment, the performance of the EKF, UKF, PAEKF, PAUKF, PAUKF with IMU, EKF(IMU), LeGO-LOAM are tested. The EKF and UKF are good filter algorithm. However, both of the filters have no additional measurement but the noisy raw odometry of the unmanned ground vehicle. In contrast, the PAUKF uses the information of the features and the pre-saved GT position of the features to relocate the position of the vehicle and correct the state of the vehicle. The experiment shows the PAUKF can relocate the vehicle well and improve the estimation precision.

Next, the PAUKF is evaluated based on the simulation in a car-like vehicle. Compare with other literature, the mean estimation error for the PAUKF is 1.08 m and the variance is 0.7147 m, which is more precise than the mean of 1.69 m and variance of 1.63 m obtained by GANGNAM for similar noise[105].

In this thesis, the algorithm is verified by using simulation and a low-speed robot in a limited environment. In the future, the algorithm should be extended and evaluated in the real autonomous vehicle with real sensors for the actual performance. Additionally, the distribution of the particles should be adapted to the uncertainty of the PAUKF in the previous sample time for better distribution.

References

[1] H. Marzbani, H. Khayyam, C. N. To, D. V. Quoc, and R. N. Jazar, “Autonomous Ve hicles: Autodriver Algorithm and Vehicle Dynamics,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3201–3211, 2019.

[2] M. Da Lio *et al.*, “Artificial co-drivers as a universal enabling technology for future in telligent vehicles and transportation systems,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 244–263, 2015.

[3] Q. Zhao, C. Xu, and S. jin, “Traffic Signal Timing via Parallel Reinforcement Learnin g,” *Smart Innov. Syst. Technol.*, vol. 149, no. 3, pp. 113–123, 2019.

[4] T. Qu, H. Chen, D. Cao, H. Guo, and B. Gao, “Switching-based stochastic model pred ictive control approach for modeling driver steering skill,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 365–375, 2015.

[5] G. Xiong *et al.*, “Cyber-physical-social system in intelligent transportation,” *IEEE/CAA J. Autom. Sin.*, vol. 2, no. 3, pp. 320–333, 2015.

[6] R. Okuda, Y. Kajiwara, and K. Terashima, “A survey of technical trend of ADAS and autonomous driving,” *Proc. Tech. Progr. - 2014 Int. Symp. VLSI Technol. Syst. Appl. VLSI-TSA 2014*, pp. 6–9, 2014.

[7] K. Bimbraw, “Autonomous Cars : Past , Present and Future,” *12th Int. Conf. Informatic s Control. Autom. Robot.*, vol. 01, pp. 191–198, 2015.

[8] T. Bagdonat, “Keynote lecture: Automated driving at Volkswagen Group Research — F uture challenges for environment perception,” pp. vii–viii, 2017.

[9] S. Kato *et al.*, “Autoware on Board: Enabling Autonomous Vehicles with Embedded S ystems,” *Proc. - 9th ACM/IEEE Int. Conf. Cyber-Physical Syst. ICCPS 2018*, pp. 287–296, 2018.

[10] A. Carballo, D. Wong, Y. Ninomiya, S. Kato, and K. Takeda, “Training Engineers in Autonomous Driving Technologies using Autoware,” *2019 IEEE Intell. Transp. Syst. Co nf. ITSC 2019*, pp. 3347–3354, 2019.

[11] V. M. Raju, V. Gupta, and S. Lomate, “Performance of Open Autonomous Vehicle Pla tforms: Autoware and Apollo,” *2019 IEEE 5th Int. Conf. Converg. Technol. I2CT 2019*, pp. 5–9, 2019.

[12] W. N. Tun, S. Kim, J. W. Lee, and H. Darweesh, “Open-Source Tool of Vector Map for Path Planning in Autoware Autonomous Driving Software,” *2019 IEEE Int. Conf. Big Data Smart Comput. BigComp 2019 - Proc.*, pp. 2019–2021, 2019.

[13] V. M. Raju, V. Gupta, and S. Lomate, “Performance of Open Autonomous Vehicle Pla tforms: Autoware and Apollo,” *2019 IEEE 5th Int. Conf. Converg. Technol. I2CT 2019*, pp. 5–9, 2019.

[14] J. Xu *et al.*, “An Automated Learning-Based Procedure for Large-scale Vehicle Dynami cs Modeling on Baidu Apollo Platform,” *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 5049 –5056, 2019.

[15] V. M. Raju, V. Gupta, and S. Lomate, “Performance of Open Autonomous Vehicle Pla tforms: Autoware and Apollo,” *2019 IEEE 5th Int. Conf. Converg. Technol. I2CT 2019*, pp. 5–9, 2019.

[16] A. Broumandan, A. Jafarnia-Jahromi, S. Daneshmand, and G. Lachapelle, “Overview of Spatial Processing Approaches for GNSS Structural Interference Detection and Mitigati on,” *Proc. IEEE*, vol. 104, no. 6, pp. 1246–1257, 2016.

[17] H. C. Yang, Y. Fan, and X. Y. Liu, “A Compact Dual-Band Stacked Patch Antenna With Dual Circular Polarizations for BeiDou Navigation Satellite Systems,” *IEEE Anten nas Wirel. Propag. Lett.*, vol. 18, no. 7, pp. 1472–1476, 2019.

[18] S. Tabibi, F. Geremia-Nievinski, and T. Van Dam, “Statistical Comparison and Combin ation of GPS, GLONASS, and Multi-GNSS Multipath Reflectometry Applied to Snow Depth Retrieval,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3773–3785, 2 017.

[19] A. Proia *et al.*, “Performance results of the Galileo Precise Timing Facility,” *2014 Eur. Freq. Time Forum, EFTF 2014*, pp. 463–467, 2015.

[20] X. Li *et al.*, “Multi-GNSS Meteorology: Real-Time Retrieving of Atmospheric Water V apor from BeiDou, Galileo, GLONASS, and GPS Observations,” *IEEE Trans. Geosci.*

Remote Sens., vol. 53, no. 12, pp. 6385–6393, 2015.

[21] Q. Wenqi, Z. Qingxi, G. Chang, and L. Chade, “Fine Doppler shift acquisition algorithm for BeiDou software receiver by a look-up table,” *J. Syst. Eng. Electron.*, vol. 31, no. 3, pp. 612–625, 2020.

[22] D. N. Aloi and F. Van Graas, “Ground-multipath mitigation via polarization steering of GPS signal,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 40, no. 2, pp. 536–552, 2004.

[23] P. Zabalegui, G. De Miguel, A. Perez, J. Mendizabal, J. Goya, and I. Adin, “A Review of the Evolution of the Integrity Methods Applied in GNSS,” *IEEE Access*, vol. 8, pp. 45813–45824, 2020.

[24] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, “A high integrity IMU/GPS navigation loop for autonomous land vehicle applications,” *IEEE Trans. Robot. Autom.*, vol. 15, no. 3, pp. 572–578, 1999.

[25] G. Wang, X. Xu, Y. Yao, and J. Tong, “A Novel BPNN-Based Method to Overcome the GPS Outages for INS/GPS System,” *IEEE Access*, vol. 7, pp. 82134–82143, 2019.

[26] W. J. Park *et al.*, “MEMS 3D DR/GPS Integrated System for Land Vehicle Application Robust to GPS Outages,” *IEEE Access*, vol. 7, pp. 73336–73348, 2019.

[27] J. Choi *et al.*, “Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 974–982, 2012.

[28] Z. Chen and X. Huang, “Pedestrian detection for autonomous vehicle using multi-spectral cameras,” *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 211–219, 2019.

[29] A. Almagambetov and S. Velipasalar, “Autonomous tracking of vehicle taillights and alert signal detection by embedded smart cameras,” *Distrib. Embed. Smart Cameras Archit. Des. Appl.*, vol. 9781461477, no. 6, pp. 121–150, 2014.

[30] W. Song, Y. Yang, M. Fu, Y. Li, and M. Wang, “Lane Detection and Classification for Forward Collision Warning System Based on Stereo Vision,” *IEEE Sens. J.*, vol. 18, no. 12, pp. 5151–5163, 2018.

[31] S. D. Pendleton *et al.*, “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, vol. 5, no. 1, pp. 1–54, 2017.

[32] A. Mukhtar, L. Xia, and T. B. Tang, “Vehicle Detection Techniques for Collision Avoidance Systems: A Review,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2318–2338, 2015.

[33] B. Nassi, D. Nassi, R. Ben-netanel, Y. Mirsky, O. Drokin, and Y. Elovici, “Phantom of the ADAS : Phantom Attacks on Driver-Assistance Systems,” *Cryptol. ePrint Arch.*, no. Report 2020/085, pp. 1–17, 2020.

[34] A. Rangesh and M. M. Trivedi, “No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras & lidARs,” *arXiv*, vol. 4, no. 4, pp. 588–599, 2018.

[35] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, “Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications,” *IEEE Sens. J.*, vol. 20, no. 9, pp. 4901–4913, 2020.

[36] S. Xie, D. Yang, K. Jiang, and Y. Zhong, “Pixels and 3-D Points Alignment Method for the Fusion of Camera and LiDAR Data,” *IEEE Trans. Instrum. Meas.*, vol. 68, no. 10, pp. 3661–3676, 2019.

[37] Q. Li, L. Chen, M. Li, S. L. Shaw, and A. Nüchter, “A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios,” *IEEE Trans. Veh. Technol.*, vol. 63, no. 2, pp. 540–555, 2014.

[38] K. Banerjee, D. Notz, J. Windelen, S. Gavarraju, and M. He, “Online Camera LiDAR Fusion and Object Detection on Hybrid Data for Autonomous Driving,” *IEEE Intell. Veh. Symp. Proc.*, vol. 2018-June, no. Iv, pp. 1632–1638, 2018.

[39] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, “LiDAR and camera detection fusion in a real-time industrial multi-sensor collision avoidance system,” *Electron.*, vol. 7, no. 6, 2018.

[40] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, “LIDAR–camera fusion for road detection using fully convolutional neural networks,” *Rob. Auton. Syst.*, vol. 111, pp. 125–131, 2019.

[41] L. Xiao, R. Wang, B. Dai, Y. Fang, D. Liu, and T. Wu, “Hybrid conditional random field based camera-LIDAR fusion for road detection,” *Inf. Sci. (Ny).*, vol. 432, pp. 543–558, 2018.

[42] J. Lee *et al.*, “Automated algorithm for removing clutter objects in mms point cloud for 3D road mapping,” *Sensors (Switzerland)*, vol. 20, no. 15, pp. 1–11, 2020.

[43] J. Min Kang, T. S. Yoon, E. Kim, and J. B. Park, “Lane-level map-matching method for vehicle localization using GPS and camera on a high-definition map,” *Sensors (Switzerland)*, vol. 20, no. 8, pp. 1–22, 2020.

[44] Z. Xiao, D. Yang, T. Wen, K. Jiang, and R. Yan, “Monocular localization with vector HD map (MLVHM): A low-cost method for commercial IVs,” *Sensors (Switzerland)*, vol. 20, no. 7, pp. 1–24, 2020.

[45] H. Cai, Z. Hu, G. Huang, D. Zhu, and X. Su, “Integration of GPS, monocular vision, and high definition (HD) map for accurate vehicle localization,” *Sensors (Switzerland)*, vol. 18, no. 10, 2018.

[46] K. Jo, C. Kim, and M. Sunwoo, “Simultaneous localization and map change update for the high definition map-based autonomous driving car,” *Sensors (Switzerland)*, vol. 18, no. 9, 2018.

[47] J. Levinson, M. Montemerlo, and S. Thrun, “Map-based precision vehicle localization in urban environments,” *Robot. Sci. Syst.*, vol. 3, pp. 121–128, 2008.

[48] S. Noh, K. An, and W. Han, “High-Level Data Fusion Based Probabilistic Situation Assessment for Highly Automated Driving,” *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2015-Octob, pp. 1587–1594, 2015.

[49] M. Agrawal and K. Konolige, “Real-time localization in outdoor environments using stereo vision and inexpensive GPS,” *Proc. - Int. Conf. Pattern Recognit.*, vol. 3, pp. 1063–1068, 2006.

[50] N. Mattern and G. Wanielik, “Vehicle localization in urban environments using feature maps and aerial images,” *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 1027–1032, 2011.

[51] C. Li, B. Dai, and T. Wu, “Vision-based precision vehicle localization in urban enviro nments,” *Proc. - 2013 Chinese Autom. Congr. CAC 2013*, pp. 599–604, 2013.

[52] I. Parra, M. A. Sotelo, D. F. Llorca, and M. Ocaa, “Robust visual odometry for vehicl e localization in urban environments,” *Robotica*, vol. 28, no. 3, pp. 441–452, 2010.

[53] I. Parra, M. A. Sotelo, D. F. Llorca, and M. Ocaa, “Robust visual odometry for vehicl e localization in urban environments,” *Robotica*, vol. 28, no. 3, pp. 441–452, 2010.

[54] R. Vivacqua, R. Vassallo, and F. Martins, *A low cost sensors approach for accurate v ehicle localization and autonomous driving application*, vol. 17, no. 10. 2017.

[55] Y. Yu, J. Li, H. Guan, F. Jia, and C. Wang, “Three-dimensional object matching in m obile laser scanning point clouds,” *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 3, pp. 492–496, 2015.

[56] A. Y. Hata and D. F. Wolf, “Feature Detection for Vehicle Localization in Urban Env ironments Using a Multilayer LIDAR,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 420–429, 2016.

[57] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using pr obabilistic maps,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 4372–4378, 2010.

[58] J. Castorena and S. Agarwal, “Ground-Edge-Based LIDAR Localization Without a Refl ectivity Calibration for Autonomous Driving,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 344–351, 2018.

[59] H. Kim, B. Liu, C. Y. Goh, S. Lee, and H. Myung, “Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity informa tion for a large scale urban area,” *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1518–1 524, 2017.

[60] A. Y. Hata and D. F. Wolf, “Feature Detection for Vehicle Localization in Urban Env ironments Using a Multilayer LIDAR,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 420–429, 2016.

[61] R. W. Wolcott and R. M. Eustice, “Visual localization within LIDAR maps for automa ted urban driving,” *IEEE Int. Conf. Intell. Robot. Syst.*, no. Iros, pp. 176–183, 2014.

[62] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, “A S urvey of the State-of-the-Art Localization Techniques and Their Potentials for Autonom ous Vehicle Applications,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, 2018.

[63] S. Fujii *et al.*, “Cooperative vehicle positioning via V2V communications and onboard sensors,” *IEEE Veh. Technol. Conf.*, pp. 0–4, 2011.

[64] M. Rohani, D. Gingras, V. Vigneron, and D. Gruyer, “A new decentralized Bayesian a pproach for cooperative vehicle localization based on fusion of GPS and VANET base d inter-vehicle distance measurement,” *IEEE Intell. Transp. Syst. Mag.*, vol. 7, no. 2, p p. 85–95, 2015.

[65] N. Mattern, M. Obst, R. Schubert, and G. Wanielik, “Co-operative vehicle localization algorithm - Evaluation of the CoVeL approach,” *Int. Multi-Conference Syst. Signals De vices, SSD 2012 - Summ. Proc.*, pp. 1–5, 2012.

[66] R. H. Ordóñez-Hurtado, E. Crisostomi, W. M. Griggs, and R. N. Shorten, “An Assess ment on the Use of Stationary Vehicles as a Support to Cooperative Positioning,” no. February, 2015.

[67] K. Golestan, S. Seifzadeh, M. Kamel, F. Karray, and F. Sattar, “Vehicle localization in VANETs using data fusion and V2V communication,” *DIVANet’12 - Proc. ACM Wor k. Des. Anal. Intell. Veh. Networks Appl.*, no. October, pp. 123–130, 2012.

[68] L. Altoaimy, I. Mahgoub, and M. Rathod, “Weighted localization in Vehicular Ad Hoc Networks using vehicle-to-vehicle communication,” *2014 Glob. Inf. Infrastruct. Netw. Symp. GIIS 2014*, pp. 1–5, 2014.

[69] L. Altoaimy and I. Mahgoub, “Fuzzy logic based localization for vehicular ad hoc net works,” *IEEE SSCI 2014 2014 IEEE Symp. Ser. Comput. Intell. - CIVTS 2014 2014 I EEE Symp. Comput. Intell. Veh. Transp. Syst. Proc.*, pp. 121–128, 2015.

[70] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *J. Fluids Eng. Trans. ASME*, vol. 82, no. 1, pp. 35–45, 1960.

[71] S. Diego, *Stochastic Processes and Filtering Theory*. CA, USA: Academic Press, 1970.

[72] H. Ahmad and T. Namerikawa, “Extended Kalman filter-based mobile robot localization

with intermittent measurements," *Syst. Sci. Control Eng.*, vol. 1, no. 1, pp. 113–126, 2013.

[73] Y. Xu, Y. S. Shmaliy, C. K. Ahn, G. Tian, and X. Chen, "Robust and accurate UWB -based indoor robot localisation using integrated EKF/EFIR filtering," *IET Radar, Sonar Navig.*, vol. 12, no. 7, pp. 750–756, 2018.

[74] L. Yahui, J. Ping, H. Wenwu, and X. Yanqun, "Application of EKF in Laser/Inertial S ensors localization system," *Proc. - Int. Conf. Electr. Control Eng. ICECE 2010*, pp. 3 369–3372, 2010.

[75] J. woo Yoon and B. woo Kim, "Vehicle position estimation using nonlinear tire model for autonomous vehicle," *J. Mech. Sci. Technol.*, vol. 30, no. 8, pp. 3461–3468, 2016.

[76] S. Zhao, J. Gu, Y. Ou, W. Zhang, J. Pu, and H. Peng, "IRobot Self-localization using EKF," *2016 IEEE Int. Conf. Inf. Autom. IEEE ICIA 2016*, no. 61273335, pp. 801–80 6, 2017.

[77] Y. Xiao, Y. Ou, and W. Feng, "Localization of indoor robot based on particle filter w ith EKF proposal distribution," *2017 IEEE Int. Conf. Cybern. Intell. Syst. CIS 2017 IE EE Conf. Robot. Autom. Mechatronics, RAM 2017 - Proc.*, vol. 2018-Janua, pp. 568–5 71, 2018.

[78] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proc. I EEE*, vol. 92, no. 3, pp. 401–422, 2004.

[79] S. Y. Chen, "Kalman filter for robot vision: A survey," *IEEE Trans. Ind. Electron.*, vo l. 59, no. 11, pp. 4409–4420, 2012.

[80] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear syste ms," *Signal Process. Sens. Fusion, Target Recognit. VI*, vol. 3068, p. 182, 1997.

[81] R. van der Merwe, A. Doucet, N. De Freitas, and E. Wan, "A Tutorial on Particle Fil tering and Smoothing," *Proc. Neural Inf. Process. Syst.*, no. December, pp. 4–6, 2000.

[82] S. Thrun, "Particle Filters in Robotics," in *In Proceedings of Uncertainty in AI (UAI) 2002*, 2002.

[83] D. F. N. and G. N. Doucet A, *Sequential Monte Carlo Methods in Practice*. 2001.

[84] P. Aggarwal, Z. Syed, and N. El-Sheimy, "Hybrid extended particle filter (HEPF) for i ntegrated inertial navigation and global positioning systems," *Meas. Sci. Technol.*, vol. 20, no. 5, 2009.

[85] P. Aggarwal, D. Gu, S. Nassar, Z. Syed, and N. El-Sheimy, "Extended Particle Filter (EPF) for INS/GPS land vehicle navigation applications," *20th Int. Tech. Meet. Satell. Div. Inst. Navig. 2007 ION GNSS 2007*, vol. 3, no. September, pp. 2619–2626, 2007.

[86] Y. Wan, S. Wang, and X. Qin, "IMM iterated extended ∞ particle filter algorithm," *M ath. Probl. Eng.*, vol. 2013, 2013.

[87] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle fi lter," *Adv. Neural Inf. Process. Syst.*, 2001.

[88] A. Panah and K. Faez, "Hybrid filter based simultaneous localization and mapping for a mobile robot," in *Lecture Notes in Computer Science (including subseries Lecture No tes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. LNCS 69 79, pp. 343–352.

[89] H. Nan, W. Chengdong, J. Tong, and J. Peng, "Hybrid Filter Localization Algorithm B ased on the Selection Mechanism," 2015, pp. 1128–1131.

[90] I. Ullah, Y. Shen, X. Su, C. Esposito, and C. Choi, "A Localization Based on Unscent ed Kalman Filter and Particle Filter Localization Algorithms," *IEEE Access*, vol. 8, pp. 2233–2246, 2020.

[91] S. J. Kim and B. K. Kim, "Dynamic ultrasonic hybrid localization system for indoor mobile robots," *IEEE Trans. Ind. Electron.*, vol. 60, no. 10, pp. 4562–4573, 2013.

[92] M. A. Al Khedher, "Hybrid GPS-GSM Localization of Automobile Tracking System," *I nt. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 6, pp. 75–85, 2011.

[93] C. Fritsche, A. Klein, and D. Würtz, "Hybrid GPS/GSM localization of mobile termina ls using the extended Kalman filter," *Proc. - 6th Work. Positioning, Navig. Commun. WPNC 2009*, pp. 189–194, 2009.

[94] W. Yu, J. Peng, X. Zhang, S. Li, and W. Liu, “An adaptive unscented particle filter a lgorithm through relative entropy for mobile robot self-localization,” *Math. Probl. Eng.*, vol. 2013, 2013.

[95] T. Moore and D. Stouch, “A generalized extended Kalman filter implementation for th e robot operating system,” *Adv. Intell. Syst. Comput.*, vol. 302, pp. 335–348, 2016.

[96] T. Moore and D. Stouch, “A generalized extended Kalman filter implementation for th e robot operating system,” *Adv. Intell. Syst. Comput.*, vol. 302, pp. 335–348, 2016.

[97] O. Gougeon and M. H. Beheshti, “2D Navigation with a Differential Wheeled Unmann ed Ground Vehicle *,” no. June 2017, 2018.

[98] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, “Kinematic modeling an d analysis of skid-steered mobile robots with applications to low-cost inertial-measurem ent-unit-based motion estimation,” *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 1087–1097, 2009.

[99] G. Grisetti, C. Stachniss, and W. Burgard, “Improving grid-based SLAM with Rao-Blac kwellized particle filters by adaptive proposals and selective resampling,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2005, pp. 2432–2437, 2005.

[100] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with Rao-Blackwellized particle filters,” *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, 200 7.

[101] S. Zaman, W. Slany, and G. Steinbauer, “ROS-based mapping, localization and autono mous navigation using a Pioneer 3-DX robot and their relevant issues,” *Saudi Int. Elec tron. Commun. Photonics Conf. 2011, SIECPC 2011*, pp. 0–4, 2011.

[102] A. Desai and D. J. Lee, “Visual Odometry Drift Reduction Using SYBA Descriptor an d Feature Transformation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1839–1851, 2016.

[103] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odo metry and Mapping on Variable Terrain,” *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4758 –4765, 2018.

[104] S. Minfen, F. H. Y. Chan, and P. J. Beadle, “A method for generating non-Gaussian n oise series with specified probability distribution and power spectrum,” *Proc. - IEEE In t. Symp. Circuits Syst.*, vol. 2, pp. 5–8, 2003.

[105] J. K. Suhr, J. Jang, D. Min, and H. G. Jung, “Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments,” *IEEE Trans. Intell. Transp. S yst.*, vol. 18, no. 5, pp. 1078–1086, 2017.

Nomenclature

Parameter	Description
$Particle_i,\ P_i$	The ith particle
x, y	Vehicle position in the x, y dimension
θ	Yaw angle
x_{random_noise}	The random noise of x
y_{random_noise}	The random noise of y
z_{random_noise}	The random noise of z
$Particles$	The group of the particles
$\overline{X}_{k+1}$	Predicted state at sample time k + 1
K	Timestamp
$\bar{x}$	Predicted X position of the vehicle
$\bar{y}$	Predicted Y position of the vehicle
$\bar{\theta}$	Predicted yaw angle of the vehicle
v	The velocity of the vehicle
$\dot{\theta}$	Yaw rate of the vehicle
Δt	Sample time
x_v	The vehicle x position in the map coordinate
y_v	The vehicle y position in the map coordinate
θ_v	The yaw angle of the vehicle in the map coordinate
$\theta_{i,v}$	The yaw angle of the ith particle
θ_j	The relative angle of the vehicle coordinate and feature j
d_j	The relative distance of ego vehicle and feature j
$x_{b,j}$	The relative x distance of ego vehicle and feature j
$y_{b,j}$	The relative y distance of ego vehicle and feature j
z_{k+1}	Measurement vector at time k + 1
N	Number of the used particles
w_i	Weight of the ith particle
$w_{[1,2\ldots N]}$	The weight group of each particle
x_i	The x value of the ith particle
y_i	The y value of the ith particle
${}^{global}x_{b,j}$	The x position of ith feature transformed by the particle
${}^{global}y_{b,j}$	The y position of ith feature transformed by the particle
σ_x	The standard deviation of the noise in the x-direction
σ_y	The standard deviation of the noise in the y-direction
$p(x_i, y_i)$	Multivariable normal distribution
$\mu_{x,b,j}$	The position x of the jth feature in the pre-saved map

$\mu_{y,b,j}$	The position y of the jth feature in the pre-saved map
$x_{p,i}$	The position x value of the ith particle
$y_{p,i}$	The position y value of the ith particle
$\hat{X}_{PF}$	The estimated result of the particle filter
λ	The designed hyperparameter of UKF
n_x	The number of the state of UKF
$x_{paukf,k}$	The state of the PAUKF at time k
μ_k	The mean of the state
P_k	Covariance matrix at time k
$x_{paukf,k,aug}$	The state of PAUKF at time k
w_{velacc}	The noise of vehicle acceleration
w_{yawacc}	The noise of vehicle yaw acceleration
σ_{velacc}	The standard deviation of the noise of vehicle acceleration
σ_{velacc}	The standard deviation of the noise of vehicle yaw acceleration
$x_{paukf,k+1}$	The state of the PAUKF at time k+1
$x_{paukf,k+1,aug}$	The augmented state of the PAUKF at time k+1
$\mu_{paukf,k,aug}$	The augmented mean value of the PAUKF at time k
$n_{x,aug}$	The number of the augmented state of PAUKF
$P_{paukf,k,aug}$	The augmented covariance matrix at time k of PAUKF
$w_{paukf,i}$	Weight of ith sigma point
$\bar{x}_{paukf,k+1\|k}$	Predicted state based on the weight of sigma points and states
$\bar{P}_{k+1\|k}$	Predicted variance based on sigma points and predicted state mean
$\omega_{paukf,k+1}$	Measurement noise of PAUKF.
$Z_{paukf,k+1\|k,i}$	Measurement prediction based on sigma points.
$X_{paukf,k+1\|k,i}$	Sigma points of the state
A	Measurement transition model.
$z_{paukf,k+1\|k}$	Predicted measurement based on sigma points and weights
$S_{k+1\|k}$	Predicted measurement covariance matrix.
R	Variance matrix of the measurement noise.
$\sigma_{x_{pf}}$	The standard deviation of PF estimation in the x dimension
$\sigma_{y_{pf}}$	The standard deviation of PF estimation in the y-dimension
$T_{k+1\|k}$	Cross-correlation matrix of PAUKF
$K_{k+1\|k}$	Kalman gain of PAUKF
$\hat{x}_{PAUFK}$	Final state estimation of PAUKF.
$\hat{P}_{PAUFK}$	Final state variance matrix of PAUKF
$RMSE_{est}$	The RMSE of the estimation result
$RMSE_{noise}$	The RMSE of the noise
$Position_{est_i}$	The ith estimated position value in the recorded data

$\text{Position}_{\text{mean}_{\text{est}_i}}$	The ith expectation estimated position value in the record data
$\text{Position}_{\text{noise}_i}$	The ith noisy position value in the recorded data
$\text{Position}_{\text{mean}_{\text{noise}_i}}$	The ith expectation noisy position value in the recorded data
σ_{v_z}	The standard deviation of vehicle noise in the vertical direction
$\overline{X}_{k+1}$	Predicted state at sample time k+1
z_v	The movement of the vehicle in the Z direction
F_j	The jth feature
α_j	The relative bearing angle to jth feature
β_j	The relative elevation angle to jth feature
$x_{p,i}, y_{p,i}, z_{p,i}$	The x, y, z value of the ith particle
${}^{global}x_{f,j}$	The x position of ith feature transformed by the particle
${}^{global}y_{f,j}$	The y position of ith feature transformed by the particle
μ_z	The expectation value of the z
$P(x_{p,i}, y_{p,i}, z_{p,i})$	Probability when the particle is $x_{p,i}, y_{p,i}, z_{p,i}$
$\sigma_x, \sigma_y, \sigma_z$	Standard deviation in the x, y, z-direction
$\mu_{x,f,j}$	The position x of the jth feature in the pre-saved map
$\mu_{y,f,j}$	The position y of the jth feature in the pre-saved map
$\mu_{z,f,j}$	The position z of the jth feature in the pre-saved map
$P_{\text{LiDAR}}(x, y, z)$, $P_{\text{RADAR}}(x, y, z)$, $P_{\text{Camera}}(x, y, z)$, $P_{\text{perception source}}$	The probability of perception based on different sensors
$h()$	Projection of the estimated value to the high-accuracy measurement
P	Variance matrix of the high-accuracy measurement
μ_t	The final estimated result of the PAUKF
γ	The random noise of the high-accuracy measurement
$z_{h,1:T}$	Whole data of the high-accuracy measurement
$x_{paukf,1:T}$	Whole data of the estimated value of the PAUKF
y_t	Measurement value from a highly accurate sensor
R_{op}	Optimal measurement variances
a, b, c	Tuning hyper-parameters
EC	Evaluation criteria
$x_{UGV,k+1}$	The predicted state at k+1
$x_{UGV,k}$	The state at k
$Noise_{UGV}$	The noise of the UGV

ABSTRACT

200 years ago, the vehicle with an engine is invented. From that time, the technologies used in the vehicle become more convenient and safe. In current days, with the development of computing ability and sensing technology, the vehicle starts to be controlled not only by the human but also controlled by the algorithm. Therefore, how to make the vehicle drive safely and correctly becomes a hot issue of current research.

To control the autonomous vehicle correctly, the vehicle needs to recognize the surrounding environment precisely. One of the most important parameters of a vehicle is the location of the vehicle's global coordinate. In general, the location can be calculated by using the GPS(global positioning system) signal directly. However, Owing to the noisy global positioning system (GPS) signal and multipath routing in urban environments, a novel, practical approach is needed. Since the GPS can not provide precise location information, researchers have to find other methods. For estimating the state of vehicles precisely, researchers developed a map that contains precise road elements like a road sign, traffic sign, and surrounded buildings. With the map′s help, the algorithm can localize the vehicle by using the geometry relationship of the vehicle and surrounding road elements. However, not only the map has different kinds of noise, but also all the sensors mounted on the vehicle have different kinds of noise. There is no way the noises are all in Gaussian form. So a sensor fusion algorithm that can handle the Non-Gaussian noise in-vehicle information and sensor information is needed.

In this study, a sensor fusion algorithm, PAUKF(particle aided unscented Kalman filter) is proposed. PAUKF trying to handle non-Gaussian noise and localizes the vehicle based on the map information. By using the information of the ground truth of the features and the detected features based on the sensors in the vehicle and the data of the vehicle, the PAUKF can estimate the global location of the vehicle precisely. The global position is estimated based on the features, thus the PAUKF does not reply to the signal of the GPS. The main part of the PAUKF can be divided into two parts. One is PF(particle filter) and the other one is UKF(unscented Kalman filter). The particle filter

is an application of the Monte-Carlo methodology that can process all kinds of noise. The particle filter generates particles and compares the particles with the measurement from the sensors of the vehicle. The UKF is a version of the improved Kalman filter. The UKF generates several sigma points and processes the non-Gaussian noise by using these sigma points. By combining the PF and UKF as PAUKF, the PAUKF can filter the Non-Gaussian noise effectively and localizing the vehicle precisely and smoothly. The performance of PAUKF is evaluated based on the two kinds of platforms. One is the experiment based on the unmanned ground vehicle by using ROS, and the other one is based on the car-like autonomous vehicle based on the simulation. Both results of the simulation and real test prove that the effectiveness of the PAUKF compares to the other works of literature.

Keywords: particle filter; sensor fusion; Unmanned ground vehicle; autonomous vehicle; unscented Kalman filter; ROS;